Deep learning methods for long RNA secondary structure prediction

Méthodes de deep learning pour la prédiction de structure secondaire des ARNs longs

Thèse de doctorat de l'université Paris-Saclay

École doctorale n°580 : Sciences et technologies de l'information et de la communication (STIC)
Spécialité de doctorat : Informatique
Graduate School : Informatique et sciences du numérique
Référent : Université d'Évry Val d'Essonne

Thèse préparée dans l'unité de recherche **IBISC** (Université Paris-Saclay, Univ Évry), sous la direction de **Fariza TAHI**, Professeure, le co-encadrement de **Eric ANGEL**, Professeur, et de **Pierre BARTET**, Docteur

Thèse soutenue à Paris-Saclay, le 18 juin 2025, par

Loïc OMNES

THESE DE DOCTORAT

NNT : 2025UPASG032

Titre: Méthodes de deep learning pour la prédiction de structure secondaire des ARNs longs

Mots clés: ARN longs, pseudo-noeuds, prédiction de structure secondaire, apprentissage profond, diviser pour régner

Résumé: Le rôle essentiel des ARNs a été démontré dans divers processus biologiques et maladies. Toutefois, on ignore encore la fonction de nombreux ARNs. Une meilleure connaissance de leur rôle pourrait permettre de découvrir de nouveaux biomarqueurs ou cibles thérapeutiques et ainsi d'améliorer l'efficacité des traitements médicaux. Cependant, la validation expérimentale de leur fonction est très coûteuse, ce qui pose un frein à l'étude de leurs rôles. Il est possible de pallier ce problème grâce à des outils informatiques. En particulier, l'apprentissage profond est aujourd'hui fréquemment utilisé pour l'étude des ARNs. Il permet de découvrir efficacement des motifs récurrents dans de larges jeux de données.

On distingue traditionnellement les ARNs courts et les ARNs longs en fonction d'un seuil de 200 nucléotides. Toutefois, différents seuils ont déjà été proposés. Nous définissons ici ce seuil à 1000 nucléotides. En effet, si les ARNs plus courts que ce seuil ont été étudiés en profondeur aujourd'hui, les ARNs plus longs possèdent des fonctions très variées et sont encore mal caractérisés. La majorité des méthodes existantes se focalisent sur l'étude des ARNs courts et ne permettent pas d'être étendues aux ARNs longs, que ce soit pour des raisons de performance ou de complexité algorithmique.

Les ARNs peuvent être caractérisés notamment par leur structure secondaire, permettant de comprendre leur fonction. Les pseudo-noeuds sont un type de motif biologique particulier au sein de la structure secondaire des ARNs car ils ne sont pas imbriqués dans la structure principale. De ce fait, les pseudo-noeuds permettent un aperçu précieux de la structure des ARNs dans l'espace en trois dimensions et donc de les caractériser plus finement. Toutefois, la détermination des pseudo-noeuds est un problème complexe pour lequel les performances des méthodes actuelles sont encore insatisfaisantes.

Nous utilisons l'apprentissage profond pour déterminer la structure secondaire des ARNs longs, à partir de leur séquence biologique uniquement. Dans cette thèse, nous présentons tout d'abord DivideFold, qui a pour but de prédire la structure secondaire des ARNs longs selon leur séquence biologique. Nous nous basons sur une approche "diviser pour régner" afin de nous adapter à des ARNs plus longs en temps linéaire. Notre algorithme utilise des motifs connus pour représenter l'information dans la séquence, puis divise la séquence récursivement en plusieurs fragments grâce à un réseau de neurones convolutifs à une dimension jusqu'à ce qu'ils soient suffisamment courts pour pouvoir être donnés à une méthode existante de prédiction de structure secondaire. En deuxième lieu, nous proposons une extension de DivideFold permettant la prédiction de structure secondaire avec pseudo-noeuds pour les ARNs longs. En utilisant des fragments suffisamment larges, en les fusionnant, et en utilisant une méthode existante capable de prédire les pseudo-noeuds dans les fragments, il est possible pour DivideFold de reconnaître les pseudo-noeuds dans les ARNs longs, même à longue distance. Enfin, nous proposons de nouvelles fonctions d'augmentation de données pour les séquences et les structures secondaires des ARNs, permettant d'améliorer les performances et les capacités de généralisation des méthodes d'apprentissage en mettant à disposition un jeu de données plus varié. Cela est particulièrement important pour les ARNs longs, pour lesquels la quantité de données de structure secondaire disponibles est très restreinte. De telles méthodes existent déjà pour les séquences d'ARN, mais pas encore pour les données de structure secondaire.

Notre outil DivideFold est mis à disposition de la communauté scientifique sur la plateforme EvryRNA.

université PARIS-SACLAY

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre: Deep learning methods for long RNA secondary structure prediction

Mots clés: long RNA, pseudoknots, secondary structure prediction, deep learning, divide and conquer

Abstract: The essential role of RNAs in various biological processes and diseases has been demonstrated. However, the function of many RNAs is still unknown. A better understanding of the role of RNAs could lead to the discovery of new biomarkers or therapeutic targets to improve the efficacy of medical treatments. However, the experimental validation of the function of RNAs is very costly, which hinders the study of their roles. This problem can be overcome with the help of computational tools. In particular, deep learning is now widely used to study RNAs, enabling accurate and efficient methods for many tasks by discovering recurrent patterns in large datasets.

A distinction is traditionally made between short and long RNAs based on a threshold length of 200 nucleotides. However, different thresholds have been proposed. We define here this threshold at 1,000 nucleotides. Indeed, while RNAs shorter than this threshold have been extensively studied, longer RNAs have a wide range of functions and are not yet well characterized. Most existing methods focus on the study of short RNAs and do not extend to long RNAs, either for reasons of performance or algorithmic complexity.

RNAs can be characterized by their secondary structure, thus allowing us to understand their function. Pseudoknots are a special type of biological motif within the secondary structure of RNAs in that they are not nested within the main structure. As a result, pseudoknots provide valuable insight into the structure of RNAs in three-dimensional space, allowing them to be more finely characterized. However, the determination of pseudoknots is a complex problem for which the performance of current methods still leaves to be desired.

We use deep learning to determine the secondary structure of long RNAs from their biological sequence alone. In this thesis, we first present DivideFold, which aims to predict the secondary structure of long RNAs based on their biological sequence. We rely on a "divide and conquer" approach based on deep learning to process longer RNAs in linear time. Our algorithm uses an insertion of various known motifs to represent the information in the sequence, then recursively divides the sequence into multiple fragments using a one-dimensional convolutional neural network until they are short enough to be passed to an existing secondary structure prediction method. Secondly, we propose to extend DivideFold to secondary structure prediction with pseudoknots for long RNAs. By using sufficiently large fragments, merging them, and using an existing method that is able to predict pseudoknots in fragments, we extend DivideFold to the detection of pseudoknots in long RNAs, even over long distances. Finally, we propose new data augmentation functions for RNA sequences and secondary structures, which help improve the performance and generalization capabilities of learning methods by providing a more diverse data set. This is particularly important for long RNAs, for which the amount of available secondary structure data is very limited. Such methods already exist for RNA sequences, but do not yet extend to secondary structure data.

Our tool DivideFold is made available to the scientific community on the EvryRNA platform.

Contents

List of figures

List of tables

1 Introduction

1.1 Context

Precision medicine aims to tailor treatments to an individual's unique molecular and clinical profile [Jameson, 2015]. A growing aspect of this personalized approach involves exploring the role of RNA molecules in health and disease [Esteller, 2011]. While much attention has historically been directed toward protein-coding sequences, it is now apparent that non-coding RNAs (ncRNAs) are integral to a wide range of biological processes, playing key regulatory roles in cancer and neurological, cardiovascular and developmental diseases [Calin, 2006; Prasanth, 2007; Lewin, 2009; Cech, 2014; Slack, 2019]. Furthermore, alterations in RNA secondary structure can disrupt normal cellular processes. The discovery of disease-causing mutations in RNAs is uncovering a wealth of new therapeutic targets, while advances in RNA biology and chemistry are enabling the development of novel RNA-based tools for developing therapeutics [Cooper, 2009; Aznaourova, 2020].

The secondary structure of RNA has emerged as an important information for understanding RNA function and exploiting it for therapeutic benefit [Wan, 2014; Assmann, 2023; Bose, 2024]. RNA secondary structure is made of complementary bases within the same RNA molecule that pair up, forming stems, loops, bulges, and more complex configurations. These secondary structure patterns often serve as binding interfaces for proteins, small molecules, or other RNAs. For example, a hairpin loop might enable an ncRNA to recruit a protein complex that modifies gene expression. As such, identifying the RNA secondary structure allows researchers to understand how an RNA might act within the cell.

In the context of precision medicine, discovering robust biomarkers is crucial for early diagnosis and patient stratification. Certain ncRNAs exhibit secondary structures that either stabilize or interfere with diseases, and RNA molecules may show altered expression levels in specific disease states [Fannon, 1996; Calin, 2006; Lee, 2013]. Databases such as NONCODE [Liu, 2005] document thousands of long non-coding RNAs across various species, including their associations with human diseases. Incorporating structural biomarkers into patient stratification could improve clinical trial design and therapy selection. Patients carrying an RNA that folds in a way predictive of drug resistance, for example, may need a different treatment regimen. By recognizing these variations early, clinicians can avoid ineffective therapies. Assessing these patient-specific variations could improve prognostic accuracy. By integrating secondary structure prediction with high-throughput sequencing data, researchers can identify diagnostic biomarkers [Slack, 2019] that might be invisible if only the RNA sequence or overall expression level are examined, helping guide more precise diagnostics and adapt treatment plans accordingly.

Pharmaceutical companies are increasingly interested in small molecules that modulate RNA structure. As computational predictions improve, it becomes more feasible to screen libraries of compounds for their affinity to specific RNA folds [Thomas, 2008]. This structure-centric approach, where the drug's design is inherently tied to the patient's molecular profile, strengthens the possibilities of precision medicine. RNA interference strategies, such as small interfering RNAs (siRNAs) or microRNA (miRNA) mimics/inhibitors, are becoming more common in clinical research. However, these therapies often assume straightforward binding to a target RNA. If the target RNA folds in an unexpected manner, it might render certain binding sites inaccessible [Castanotto, 2009]. When knowing the secondary structure though, researchers can design siRNAs and antisense molecules that consider the folded state, thus increasing the efficacy and stability of RNA-based targeted interventions [Burnett, 2012]. Over time, as more RNA structures are cataloged, clinicians could use sequencing data to automatically generate personalized RNA profiles. This could give a comprehensive overview of a patient's regulatory landscape, guiding everything from risk assessment to post-treatment follow-up.

However, while small RNAs are now well characterized, the secondary structure of many long RNAs is still unknown [Wan, 2014; Yankey, 2021]. Furthermore, experimental validation of the function of RNAs is very costly, which hinders the study of their roles. This problem can be overcome with the help of computational tools. In particular, deep learning is now widely used to study RNAs, enabling accurate and efficient methods for many tasks by discovering recurrent patterns in large datasets. AlphaFold 3 [Abramson, 2024] is a prime example of this success and has revolutionized protein structure prediction. Deep learning methods are now being adopted to predict RNA secondary structures more accurately. Models can learn from large datasets of experimentally validated structures, identifying patterns that traditional folding algorithms might miss. By accurately predicting how RNA molecules fold and function within patients, researchers and clinicians can identify novel biomarkers, better stratify disease subtypes, and design more personalized and effective treatments. Yet, processing long RNAs is significantly more difficult as the computational complexity becomes a major obstacle and current prediction methods tend to be less reliable. There is a need for novel methods capable of predicting the secondary structures of long RNAs to expand our knowledge of the involvement of RNAs in biological processes and diseases.

1.2 Thesis objectives

In this thesis, we aim to better determine the structural elements of long RNAs with the use of deep learning approaches. To that end, scalability is a key property in our research. In this thesis, we address three main objectives:

- To develop a deep learning model able to predict the secondary structures of long RNAs from their sequence. While most existing methods focus on the study of small RNAs and do not extend to long RNAs, either for reasons of performance or algorithmic complexity, we propose a novel "divide and conquer" approach based on deep learning, DivideFold, to process longer RNAs in linear time. Our algorithm uses an insertion of various known patterns to represent the information in the sequence, then recursively divides the sequence into multiple fragments using a deep learning-based method until they are shorter than a chosen length threshold. The fragments are then passed to an existing secondary structure prediction method before being reassembled to form the secondary structure prediction for the original sequence. DivideFold, along with all the datasets used for this study, is accessible on the EvryRNA platform at https://evryrna.ibisc.univ-evry.fr/evryrna/dividefold/home. This work has been published in the 2023 IEEE International Conference on Bioinformatics and Bioengineering (BIBE) [Omnes, 2023].

- To design a deep learning model to predict pseudoknots in long RNAs. Indeed, the determination of pseudoknots is a complex problem for which the performance of current methods still leaves much to be desired, especially for long RNAs. By ensuring a sufficiently large fragment size, merging the outer fragments at each cutting step, and using an existing method for predicting pseudoknots in fragments, we extend DivideFold to the detection of pseudoknots in long RNAs, even over long distances. This research has been accepted for publication in the journal PLOS ONE [Omnes, 2025].

- To come up with new data augmentation techniques for RNA sequences and secondary structures. Such methods already exist for RNA sequences, but do not yet extend to secondary structure data. We propose new data augmentation functions for RNA sequences and secondary structures, which improve the performance and generalization capabilities of learning methods by providing a more diverse data set. This is particularly important for long RNAs, for which the amount of available secondary structure data is very limited.

The remainder of this manuscript is structured as follows: Chapter 2 provides an introduction to core concepts in transcriptomics and machine learning essential for understanding this work. Chapter 3 presents the current state-of-the-art in RNA secondary structure prediction. The next chapters detail our contributions: to detail the methodology of DivideFold in Chapter 4, to predict the secondary structure of long RNAs in Chapter 5, to predict their pseudoknots in Chapter 6, and to develop data augmentation techniques for RNA sequences and secondary structures in Chapter 7. Finally, a conclusion is given in Chapter 8 to discuss this thesis and examine future prospects.

2 Background

2.1 RNA

Ribonucleic acid (RNA) is found in all living organisms and plays essential roles in cells, serving not only as a crucial intermediary that conveys genetic instructions from DNA for protein synthesis [Crick, 1970] but also participating in various biological processes. In fact, the diversity of RNA molecules and their functions is considerable, as many RNAs do not code for proteins at all [Cech, 2014]. Although early observations in the 1950s showed that certain RNA species like transfer RNAs (tRNAs) and ribosomal RNAs (rRNAs) play critical roles in protein synthesis [Kim, 1974; Noller, 1991], it was not fully appreciated until much later that non-coding RNAs possess a broad range of functionalities, from guiding essential housekeeping tasks to shaping the regulatory networks that control gene expression [Ponting, 2009]. RNA might also have played a key role in the early history of life on Earth, functioning as both genetic material and biological catalysts before the evolution of DNA and proteins [Gilbert, 1986; Cech, 1986; Moore, 2002]. The ribosome itself, which is responsible for synthesizing proteins, is a ribozyme (a catalytic RNA) [Steitz, 2003], highlighting RNA's pivotal catalytic role even in modern cells.

2.1.1 What is RNA ?

At the chemical level, RNA is made of ribonucleotides, each consisting of a sugar (ribose), a phosphate group, and a nitrogenous base. The four bases in RNA are adenine (A), guanine (G), cytosine (C), and uracil (U). RNA is synthesized in cells through a process called transcription, during which a complementary RNA strand is assembled from the DNA strand, with uracil substituting for the thymine (T) found in DNA. This process ensures that the genetic information stored in DNA is accurately transferred to RNA.

RNA is ubiquitous, present in all cells across different biological environments. It usually exists as a single strand, which allows it to fold into complex shapes through base pairing [Tinoco, 1999] and carry out its roles.

RNAs are categorized into classes, which are groups of RNAs that share certain functional and structural characteristics [Fogel, 2002; Smirnov, 2017]. Among the first and most functionally important RNA classes discovered are mRNAs, tRNAs, and rRNAs, which are all involved in translation. Messenger RNAs carry genetic instructions from DNA to ribosomes, large molecular machines that read the mRNA sequence and assemble amino acids into proteins [Noller, 1991]. However, not all RNA molecules code for proteins. In fact, a vast majority of transcripts are non-coding RNAs, which never get translated into proteins but nonetheless carry out essential cellular functions [Ponting, 2009]. Transfer RNAs are a prime example of ncRNAs that play a central role in translation by delivering specific amino acids to the ribosome [Kim, 1974]. Each tRNA recognizes a particular codon (a sequence of three nucleotides) on the mRNA and brings the corresponding amino acid. Ribosomal RNAs are another type of ncRNAs, found in the ribosome, where they are responsible for its structural integrity and catalytic activity. Notably, rRNAs are vital to translation, catalyzing the formation of peptide bonds between amino acids during protein synthesis [Steitz, 2003], highlighting the enzymatic capacity of RNA.

RNAs are commonly separated between small RNAs and long RNAs. Often, the threshold is set at around 200 nucleotides to label a transcript as either short (under 200 nt) or long (over 200 nt) [Ponting, 2009]. However, this boundary is not absolute. Some RNAs up to 500 nt long are categorized as small RNAs in some contexts [Eddy, 2001; Mattick, 2023]. In this thesis, we define the threshold between small RNAs and long RNAs at 1,000 nucleotides. Indeed, while RNAs shorter than this threshold have been extensively examined, longer RNAs are not yet well characterized and their study is challenging.

2.1.2 Roles and functions of RNA

We give here a more detailed overview of some of the key roles and functions that RNA fulfills across different biological contexts. First and foremost, as mentioned in Section 2.1.1, a crucial function of RNA in eucaryotic cells is to transfer genetic information from DNA to the ribosome in the form of mRNA [Jacob, 1961], aided by tRNA and rRNA to achieve translation and assemble proteins [Watson, 1963; Noller, 1991]. Before an mRNA is fully mature, it often undergoes additional processing. One of the most critical modifications is splicing, in which certain RNA regions called introns are removed while exons are joined together [Sharp, 1985]. The spliceosome, an RNA-protein complex, carries out this task, ensuring that the final mRNA contains the proper sequence for protein synthesis [Will, 2011]. In many genes, alternative splicing can occur, meaning different combinations of exons are stitched together, thus expanding the variety of proteins that can arise from a single gene [Black, 2003].

This flow of genetic information (DNA $\rightarrow$ RNA $\rightarrow$ protein), often referred to as the central dogma, is central to cellular biology [Crick, 1970]. This view placed proteins at the center of molecular function, and it was first thought that the roles of RNA were primarily limited to those of mRNA, tRNA and rRNA. However, a paradigm shift in the 2000s revealed how many other ncRNAs carry out vital regulatory and structural functions within cells and are more that mere byproducts of RNA transcription [Wilusz, 2009; Esteller, 2011]. The recognition of ncRNAs significantly reshaped our understanding of the roles of RNA [Cech, 2014]. Large-scale research programs such as the Human Genome Project [Collins, 1995; Lander, 2001] and the ENCODE project [Dunham, 2012] shed light on a surprisingly abundant universe of ncRNAs. Thousands of RNA families, which are more specific groups of homologous RNAs compared to classes, with similar sequences and well-documented functional attributes [Balakin, 1996; Parker, 2011; Hoeppner, 2012], have been identified and regrouped into the Rfam database [Kalvari, 2021; Ontiveros-Palacios, 2025], showing the complexity and diversity of RNA functions.

Among the first regulatory ncRNAs to be recognized were microRNAs (miRNAs) and small interfering RNAs (siRNAs), small molecules that can influence gene expression by binding to specific mRNAs and regulating whether or not they get translated into proteins [Bartel, 2009]. These molecules often work within specialized protein complexes such as RNA-induced silencing complexes (RISC) [Pratt, 2009]. Another example of regulatory non-coding RNA is riboswitches, RNA segments typically found in the $5'$ untranslated region of an mRNA, that can toggle between different conformations upon binding small ligands [Mandal, 2004]. By switching shapes in response to changing conditions in the cell, these RNA elements regulate gene expression [Winkler, 2005].For instance, when a ligand binds, it may induce a structural rearrangement that either allows or prevents the expression of downstream genes [Breaker, 2012]. Other regulatory roles of ncRNAs include guiding RNA editing [Daniel, 2015] and splicing [Ouyang, 2022].

Ribozymes are catalytic RNAs, demonstrating that RNAs can participate directly in biochemical reactions without requiring a protein enzyme [Doudna, 2002]. Well-known ribozyme classes include rRNAs, RNase P and telomerase RNAs. RNase P process the $5'$ ends of precursor tRNAs, ensuring their maturation, which is vital to protein translation [Reich, 1988]. Telomerase RNAs help preserving DNA integrity by adding repetitive DNA sequences to eukaryotic chromosome ends [Greider, 1985].

Aptamers are RNA (or single-stranded DNA) molecules that bind with high specificity to other targets, including proteins, small molecules, or even nucleic acids [Stoltenburg, 2007], functioning as highly selective ligands. Similarly to antibodies, aptamers can be used to detect or isolate specific molecules, as well as to act as targeting and therapeutic agents [Radom, 2013].

RNA can also be used to store genetic information directly in the case of viruses. Indeed, while most organisms store their genetic information in DNA, many viruses rely on RNA as their hereditary material [Weiss, 1991; Payne, 2017]. This allows RNA viruses to mutate rapidly, due to a lack of the proofreading enzymes that assure fidelity of DNA replication [Holland, 1982]. After infecting a host cell, these viral RNAs serve as templates for producing more copies of the viral genome and, in many cases, for synthesizing viral proteins [Payne, 2017; Wang, 2022]. Retroviruses, like HIV or Rous sarcoma virus for example, use an RNA genome that is reverse-transcribed into DNA inside the host cell, integrating into the host's genome [Temin, 1970; Hu, 2012]. Thus, RNA can act as the fundamental repository of genetic information in certain pathogens.

Studying RNA has practical implications for understanding diseases and developing therapies. Many ncRNAs are now known to participate in cancer progression, metabolic disorders, and various genetic conditions [Esteller, 2011]. Because some ncRNAs are highly tissue-specific or disease-specific, they have become potential targets for diagnostic markers and novel treatments [Slack, 2019]. Technologies such as RNA interference and CRISPR-Cas systems leverage the natural properties of RNA to modulate gene expression [Fire, 1998] or edit genomes [Jinek, 2012], with possible applications to precision medicine, to tailor treatments to patients.

2.1.3 The secondary structure of RNA

While the primary structure of RNA is characterized by its nucleotide sequence, the secondary structure [Holley, 1965] is defined by canonical base pairs [Tinoco, 1999].

Formed from hydrogen bonds, base pairs in RNA can be canonical or non-canonical. Canonical base pairs include the Watson-Crick pairs (A-U and G-C), which are the most stable, and Wobble pairs [Varani, 2000] (G-U), which are a little less stable but still very common in many RNA molecules. Although canonical pairings dominate, there are also non-canonical base pairs that can form through different hydrogen bonding configurations, such as interactions relying on the Hoogsteen or Sugar edges of the bases rather than on the Watson-Crick edges [Leontis, 2001; Nagaswamy, 2000]. Non-canonical base pairs tend to be less stable and are not considered in the secondary structure. The formation of base pairs can also be influenced by external trigger factors such as proteins or temperature [Al-Hashimi, 2008].

Base pairs typically form arrangements of structural motifs, consisting of stems and loops, within the secondary structure. A stem is a double-stranded region where complementary bases are paired, while a loop is a region where nucleotides remain unpaired. Several types of structural motifs commonly occur:

- Hairpin (terminal) loop: occurs when the RNA strand folds back on itself at the end of a stem, creating a loop of unpaired nucleotides [Svoboda, 2006].
- Internal loop: found in the middle of a stem where both sides contain unpaired bases.

- Bulge: a special case of an internal loop where unpaired nucleotides appear on only one side of the stem.
- Multibranch loop: a junction connecting three or more stems.
- Pseudoknot: when nucleotides in a loop pair up with another segment of the RNA outside that loop, leading to more complex folding patterns [Staple, 2005].

We show an example of RNA secondary structure containing the structural motifs described above in Fig 2.1.

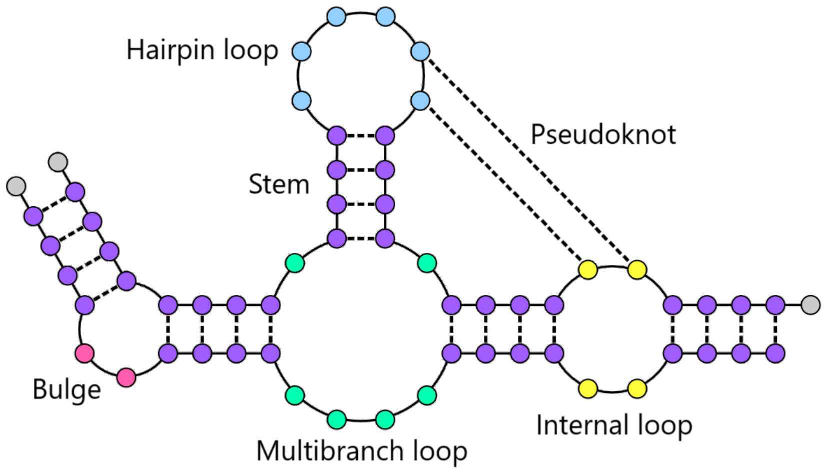

Figure 2.1: **Structural motifs in RNA secondary structure.** The secondary structure is formed by canonical base pairs, which create several structural motifs. The stems are displayed in purple, the internal loops in yellow, the bulges in pink and the multibranch loops in green. A pseudoknot is binding the hairpin loop to the internal loop. Adapted from [Weeks, 2021].

Secondary structure is central to RNA function [Wan, 2014; Assmann, 2023; Bose, 2024]. Many RNAs must form specific folded shapes to carry out their roles (for instance, creating binding sites for proteins, ligands, or other RNAs [Cech, 2014]). Certain viruses contain structured RNA elements called IRES (Internal Ribosome Entry Sites) that recruit the ribosome for protein translation, illustrating how secondary structure can be essential for gene expression [Kieft, 2008]. Disrupting these structures can impair RNA function [Jackson, 2010], which is why researchers often target them in drug design. For example, some antiviral medications bind to these structured regions to block critical viral processes.

Interestingly, the secondary structure is more conserved than the primary sequence in many RNAs. A mutation in one strand of a stem may be tolerated if it is compensated by a corresponding change in the opposite strand, preserving the base pair [Hancock, 1988]. This concept of compensatory mutations explains why structure-based analyses are important for predicting functional regions. Overall, through its secondary structure, RNA folds in a way to acquire a variety of functions that are key to many biological processes and diseases.

2.1.4 Pseudoknots

Pseudoknots are more complex structural motifs that, unlike others, are not nested within the rest of the secondary structure. Pseudoknots are made from at least two stems that are interlaced such that the end of one stem lies between both ends of the other stem [Pleij, 1990]. Essentially, a pseudoknot occurs when nucleotides in a loop are paired to complementary bases in a different segment of the RNA strand. Pseudoknots are more common than once thought and can have profound effects on RNA function and regulation [Staple, 2005]. Pseudoknots are often seen as a structural bridge between the secondary and tertiary structures of RNA, as they involve base pairing but cause the molecule to fold upon itself in three-dimensional space. Despite involving crossing base pairs, pseudoknots occurring from canonical base pairs are generally considered to be part of the secondary structure, as the secondary structure is formed from canonical base pairs.

Different types of pseudoknots have been identified by examining how the stems within these structures are arranged relatively to each other. One commonly encountered classification, proposed in the Pseudobase++ database [Taufer, 2009] cataloging known examples of pseudoknots, distinguishes various topological categories based on how the stems interlace. Among main different forms of pseudoknots in the classification are:

- H-type: This is often considered the "classic" pseudoknot and is seen most frequently in naturally occurring RNAs. In a H-type pseudoknot, a hairpin loop interacts with another unpaired region of the RNA strand.
- HHH-type (kissing hairpin): Here, two hairpin loops interact with each other, creating a distinctive "kissing" arrangement.
- HLout-, HLin- and LL-type: In these, additional stems may branch from or lie within the pseudoknot, altering how the stems interconnect.

A visualization of these different types of pseudoknots is given in Fig 2.2. By studying these confirmed examples, researchers gain insights into the biological significance and structural diversity of pseudoknots. However, although these categories provide a useful way of describing pseudoknots, some RNA pseudoknots can be even more complex, combining features of multiple types.

From a formal standpoint, a pseudoknot is made of at least two nested groups of base pairs where each base pair (i, j) in the first group and each base pair (k, l) in the second group are such that $i < k < j < l$, in a way that one stem "crosses" over the other when

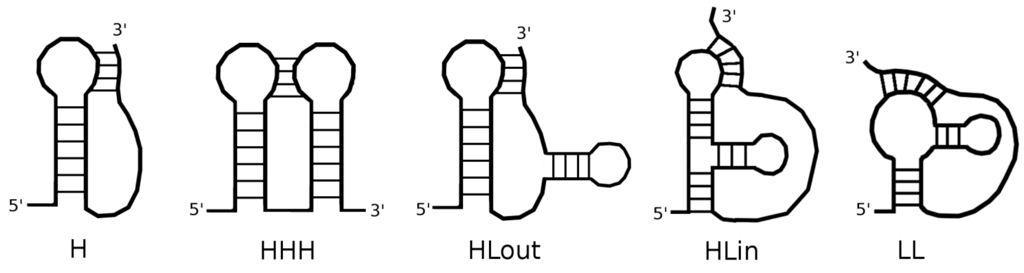

Figure 2.2: **Main types of pseudoknots.** The H-type pseudoknot is the most common, involving two stems. The HHH-type pseudoknot, or kissing hairpin, is a distant binding of two hairpin loops. The HLout-, HLin- and LL-type pseudoknots are examples of more intricate patterns. Adapted from [Legendre, 2018].

the RNA sequence is viewed in a linear representation. A visual example of a H-type pseudoknot, in which two stems are interlaced, is given in Fig 2.3.

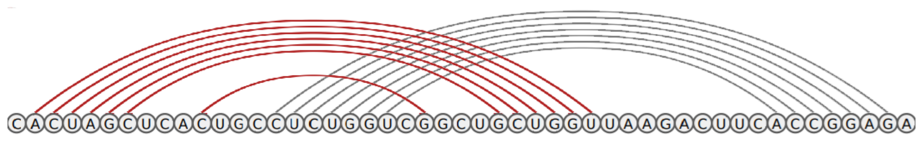

Figure 2.3: **A H-type pseudoknot.** The pseudoknot is formed from two interlaced stems displayed in red and gray respectively.

An important concept linked to pseudoknots is their structural depth, or level. This depth describes how many crossing interactions exist. A pseudoknot with multiple overlapping stems is more "deeply nested" and, accordingly, more complex. Formally, the depth of a pseudoknot can be defined by the smallest number of subsets in the pseudoknot region such that no subset contains further pseudoknots.

Pseudoknots often contribute to essential biological functions. Pseudoknots are observed in some ribozymes. The intricate folding of ribozymes, including pseudoknots, is crucial for proper enzymatic function. A well-known example is the telomerase RNA, which helps maintain the ends of eukaryotic chromosomes [Jeng, 1996; Chen, 2004]. Another example of the functional roles of pseudoknots is ribosomal frameshifting. In certain viral RNAs, a pseudoknot located within the messenger RNA can stall the ribosome during translation. When the ribosome attempts to move past the pseudoknot, it may slip by one or more nucleotides, causing a reading frame shift [Brierley, 1989; Plant, 2005; Namy, 2006]. This shift can dramatically change which protein is produced downstream of the pseudoknot. For example, the HIV-1 genome contains a pseudoknot that causes translation to halt or shift about 10% of the time, leading to the production of a different protein [Biswas, 2004; Wang, 2019b]. Aside from frameshifting, pseudoknots can sometimes slow or block translation entirely if the ribosome cannot efficiently unfold the structure. This is a phenomenon called translation regulation and can serve as a regulatory mechanism, controlling the expression levels of specific genes, especially in viruses [Giedroc, 2000; Romilly, 2020]. Therefore, an antiviral compound that binds to a pseudoknot region might prevent ribosomal frameshifting or translation regulation, or otherwise interfere with viral proteins. For this reason, pseudoknots are studied as potential drug targets. For example, to defend against severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2) variants, CRISPR-Cas13b was used to target the pseudoknot site, resulting in a 99% reduction in viral replication [Yu, 2023]. This highlights how the prediction of pseudoknots is an important step in better understanding the functions of RNA and opening new therapeutic avenues.

2.1.5 The tertiary structure of RNA

The tertiary structure of RNA refers to the arrangement of its nucleotides in the three-dimensional space. In comparison to the secondary structure, the tertiary structure is stabilized by tertiary contacts that bring distant parts of the sequence close together. These interactions allow the RNA molecule to adopt complex three-dimensional shapes necessary for its biological role. Tertiary contacts include examples such as pseudoknots, or base triples, where three nucleotides interact. The tertiary structure is often defined using torsion angles, which describe the rotations between atoms in the RNA in 3D space. An visualization of tertiary structure is given in Fig 2.4.

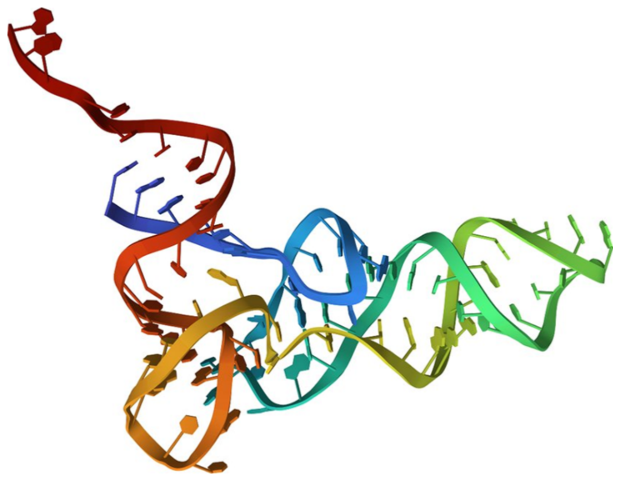

Figure 2.4: **RNA tertiary structure.** The tertiary structure describes how RNA folds in 3D space and is stabilized by long-range interactions that bring distant parts of the molecule together. Adapted from [Budnik, 2024].

The structure of RNA, including the tertiary contacts necessary to the tertiary structure, is shaped by building blocks named modules. Modules are specific forms of the general structural motifs described in Section 2.1.3, depending on their specific length, nucleotide arrangement, or association to other parts of the sequence. Known modules are cataloged in databases such as CaRNAval [Reinharz, 2018] or RNA Bricks [Chojnowski, 2014]. Each module documents which nucleotides are paired and how, typically as a graph. This information can help better understand the mechanisms behind the structure and RNA. It can also be used to perform module insertion in RNA sequences, locating where each module could be found in the sequence. Indeed, knowing which nucleotides interact in a particular module, it is possible to look for matches of such nucleotides in the sequence, indicating that the module may appear at these positions and thus that the considered nucleotides may be paired. Module insertion can be used to represent the sequence information of RNA in a way that is biologically meaningful, leveraging our knowledge of structural motifs [Becquey, 2020]. Examples of possible modules are displayed in Fig 2.5. We show a) a simple stem of length 2, b) a more complex module involving pseudoknots that bind multiple stems, and c) an example of a more sophisticated module containing intricate structural patterns.

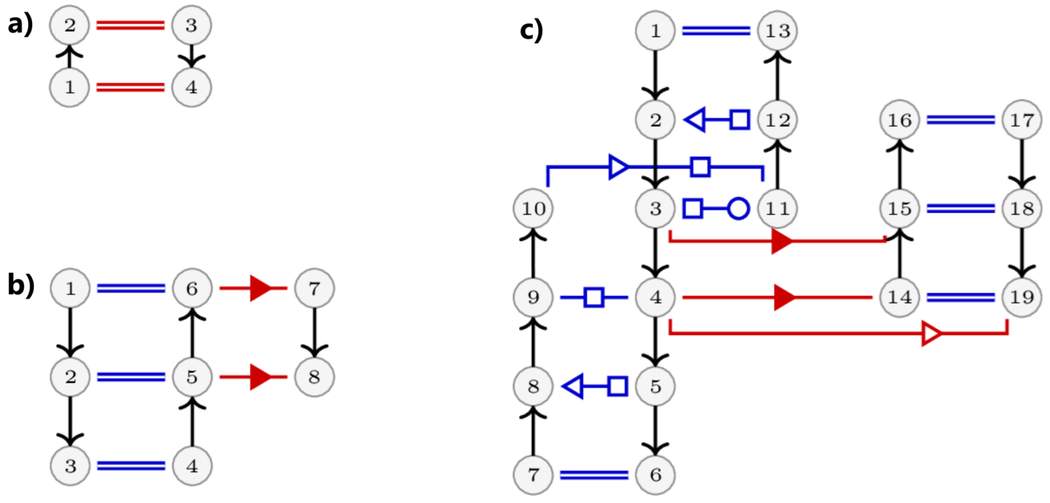

Figure 2.5: **Examples of modules.** Modules can **a)** be as simple as a stem of length 2 and involve only 4 nucleotides, **b)** be more complex and contain pseudoknots that bind different stems, or in other cases **c)** involve more intricate patterns. The modules displayed here are part of the CaRNAval [Reinharz, 2018] database.

2.1.6 Experimental methods for inferring RNA secondary structure

Experimental approaches for investigating RNA secondary structure provide valuable insights into RNA folding and function. These methods generally fall into two main categories: those based on chemical probing (using small molecules that react with exposed nucleotides) and those based on enzymatic cleavage (using enzymes that preferentially cut RNA). Importantly, chemical probing and enzymatic cleavage methods do not determine the secondary structure directly. Instead, they yield nucleotide pairing probabilities, which can then be used to infer plausible structures. In this section, we describe an overview of several key techniques, along with their strengths and limitations.

Starting with chemical probing methods, SHAPE (Selective 2′-Hydroxyl Acylation Analyzed by Primer Extension) uses chemical reagents that attach to nucleotides when they are unpaired [Merino, 2005] and acylate them. Reactivity scores can then be inferred depending on which nucleotides have been modified, and are an indication of the flexibility of nucleotides, reflecting the probability that they are unpaired. SHAPE reactivity depends not only on base pairing but also on local structural contexts.

Another similar method is dimethyl sulfate (DMS), which attaches to certain unpaired adenine and cytosine nucleotides, revealing which nucleotides are likely unpaired [Peattie, 1979]. Other chemicals exist with similar targeting properties, such as kethoxal or CMCT. By measuring which nucleotides were modified, potential base pairings can be inferred.

Lastly, in inline probing, RNA is incubated under conditions that favor spontaneous cleavage [Soukup, 1999]. As nucleotides that are flexible and unpaired are more likely to be cleaved, this is another method that can help estimate which nucleotides are paired [Regulski, 2008].

Regarding enzymatic methods, a common factor is that they often involve two enzymes that exhibit preferences for single- versus double-stranded RNA regions. The Parallel Analysis of RNA Structure (PARS) [Kertesz, 2010] method uses two enzymes: RNase V1, which generally cleaves double-stranded regions, and RNase S1, which cleaves single-stranded regions. By comparing cleavage sites from both enzymes, researchers can estimate which positions are likely paired or unpaired. FragSeq [Underwood, 2010] is another similar method that compares cleavage sites from two enzymes, where the fragments are analyzed by high-throughput sequencing.

Recent years have seen the development of high-throughput approaches (SHAPE-seq, DMS-seq [Ding, 2014], PARS-seq, FragSeq) that combine the classic chemical or enzymatic probing methods with next-generation sequencing. By mapping millions of reads or cleavage fragments back to the transcript of interest, these methods deliver transcriptome-wide structural profiles. This allows researchers to compare RNA folding across different conditions like the presence of a ligand or different cellular contexts. High-throughput methods typically follow a similar pipeline. First, a chemical probing or enzymatic cleavage method is used to mark or cut unpaired regions in the RNA. Then, an enzyme performs reverse transcription to synthesize complementary DNAs (cDNAs), but stalls or introduces mutations at modified sites. Next, the truncated or mutated cDNAs are amplified by polymerase chain reaction (PCR) and prepared for sequencing. Lastly, sequencing reads are aligned to the original RNA sequence, revealing where modifications or cuts occurred and thereby providing a reactivity or cleavage profile for each nucleotide.

However, certain limits to these chemical or enzymatic probing methods must be kept in mind. Certain nucleotides may remain hidden from chemical reagents because they reside in deep pockets or complex tertiary folds, which can lead to incomplete or misleading data. Moreover, reagents may mark any sites that are not engaged in canonical interactions as unpaired and not be able to distinguish between actually unpaired nucleotides and those engaged in higher-order interactions. Methods like SHAPE are also sensitive to local environment and neighboring nucleotides, and small changes in conditions such as temperature, pH, or ion concentration can significantly alter RNA structure. Thus, experimental data often reflect a snapshot of an RNA's fold under a specific set of conditions, rather than a universal structure of that RNA. Finally, raw probing and cleavage data require normalization and analysis pipelines to translate reactivity or cleavage frequencies into meaningful structural models. Different labs may use slightly different normalization protocols, and comparing datasets across experiments can be challenging [Wirecki, 2020].

Chemical and enzymatic probing mostly inform secondary structure, but some other methods also yield clues about tertiary contacts. When the tertiary structure is needed, tools such as X-ray crystallography, cryo-electron microscopy or nuclear magnetic resonance can be used. These methods can locate atoms in 3D space but are more technically demanding, requiring crystallization, specialized equipment, or large amounts of highly purified RNA.

While experimental methods for the identification of RNA secondary structures are crucial to gather real data, they are also costly. The development of computational tools is essential to overcome this obstacle.

2.2 Deep learning

Deep learning is a subfield of machine learning in which computational models, often referred to as neural networks, learn patterns directly from data by tuning parameters in multiple interconnected layers of processing units (or "neurons"), capture increasingly abstract features at each layer [LeCun, 2015]. The origins of deep learning can be traced back to 1958 when Frank Rosenblatt proposed the perceptron [Rosenblatt, 1958].

2.2.1 General framework for evaluating and training a deep learning model

The evaluation of a deep learning model starts with a forward pass through the network. At the input layer, the model receives raw data (such as images, text, or numerical features). As the input propagates through successive hidden layers, the network's parameters transform and combine features into more complex representations. Finally, at the output layer, the model produces a prediction or estimation about the input, such as assigning a class label or generating a numerical value. Once the model generates an output, it is compared to the correct or ground truth label. For a classification problem, the ground truth label might be the true class of the input data; for a regression problem, it could be a numeric value that the model is supposed to predict. The gap between the model's predicted output and the ground truth label is quantified by a loss function or an objective function. The loss function allows to evaluate the quality of a prediction relatively to a ground truth label and to give a metric of the model's performance. The goal is for the model's predictions to be as close as possible to these ground truth labels. A higher loss indicates large errors in prediction; a lower loss indicates that the model is making more accurate predictions.

After completing the forward pass and computing the loss between the model's prediction and the ground truth label, the loss is sent backward through the network via an algorithm called backpropagation [Rumelhart, 1986]. Backpropagation applies the chain rule of calculus to compute partial derivatives of the loss with respect to every parameter in the network, where these parameters are typically the weights and biases of each neuron. Once the gradients (partial derivatives) are computed, they guide how the network's parameters should be updated to reduce the loss on subsequent predictions. This process is performed by the optimizer. There are different existing optimizers, but all are variations of the gradient descent [Cauchy, 1847]. In gradient descent, each parameter θ is adjusted in the opposite direction of its gradient:

$$\theta \leftarrow \theta - \eta \frac{\partial \mathcal{L}}{\partial \theta} \tag{2.1}$$

where θ represents a parameter, and η is the learning rate (a small positive value controlling the step size).

In practice, optimizers can take various forms that extend or modify basic gradient descent, for example by adaptively tuning the learning rate or by incorporating momentum. Popular optimizers include stochastic gradient descent (SGD), AdaGrad, RMSProp or Adam. Due to its rapid convergence, Adam [Kingma, 2017] is widely used for the training of deep learning models. Regardless of the optimizer, the overall idea is always the same: use the gradients to update the model's parameters and systematically lower its loss.

To train a model, the network is typically initialized first with random or strategically chosen parameter values. Then, the training dataset is generally split into small batches of data. This allows faster convergence and better generalization compared to processing the whole dataset at once, and faster computation compared to processing the training examples one at a time [Kandel, 2020]. Then, the steps that we described in Section 2.2.1 are repeated many times. For each batch of data, the forward pass is first performed. The loss is then computed, and finally backpropagated into the network, leading to the update of the parameters from the gradients. For each batch of data, the forward pass is first performed. Afterward, the loss is calculated by comparing the prediction to the ground truth. This loss is then backpropagated through the network, and the model parameters are updated using the gradients to minimize the loss. This iterative process typically continues for multiple epochs, where an epoch is a complete pass over the entire training dataset.

However, in deep learning, we sometimes have to deal with scarce datasets. This is usually very detrimental to the model, as the quantity of available training data is of paramount importance. Yet, other larger datasets can often be found, even if only indirectly linked to the task at hand. A common strategy is to pre-train the model on a large correlated dataset, learning important patterns in the data while minimizing the risks of overfitting. However, since this dataset does not correspond exactly to the actual objective of the model, it needs to be trained a second time on the small dataset that is specific to the considered task in order to learn to solve it. This process is called fine-tuning and enables transfer learning from the large dataset to the small one. By pre-training a model on a large dataset and fine-tuning it to a smaller one that is specific to the task at hand, it is possible to leverage much larger quantities of data, learning patterns that may not be found otherwise and reducing the risks of overfitting, which is usually a major obstacle when dealing with small datasets. Foundation models such as large language models, including GPT models, are very powerful and commonly used applications of this principle. Foundation models are encoders pre-trained on vast datasets to learn various relationships. They can then be combined with a decoder that is fine-tuned to a specific task. This allows to make use of the features learned by the foundation model for a wide range of use cases.

Finally, an evaluation of how well the model is performing can be conducted on a separate validation dataset. The same steps are applied, except for backpropagation and parameter update, as the model should generally not be trained during evaluation. The resulting performance metric gives an indication of whether the network is learning effectively (i.e. if the model has learned to generalize to the validation dataset) or if the model is overfitting. Overfitting is when the model is improving on the training dataset, but its performance is degrading on the validation dataset. This means that the model is picking up on the specificities of the examples in the training dataset instead of learning the general structure of the data, decreasing its generalization capabilities on the validation dataset [Ying, 2019]. This should be avoided since learning the general patterns in a dataset is generally what is wanted from the model, rather than learning specific examples. For this reason, the final evaluation of the model is often done on an independent test dataset to value generalization.

By combining these principles, deep learning methods have shown groundbreaking results for many applications in a variety of fields such as text generation, object recognition, robotics, finance, healthcare, and many more.

2.2.2 Loss functions for regression

Regression tasks involve predicting a continuous output, such as a price, temperature, or probability, given input data. More generally, the predictions and target labels are considered to be vectors of dimension N. To evaluate the quality and usefulness of a prediction, several metrics are commonly used in regression problems. Loss functions are piecewise differentiable functions used to quantify the quality of a single prediction compared to a single ground truth label, and return gradients to the optimizer so that it can update the parameters in the network during training. Evaluation metrics can be more generally used to quantify the quality of an ensemble of predictions compared to an ensemble of ground truth labels, over one or several examples at once. In regression, loss functions and evaluation metrics are generally the same functions. They quantify the error between a prediction $(\hat{y}_i)_{i \in [\![1\ ;\ N]\!]} \in \mathbb{R}^N$ and a ground truth label $(y_i)_{i \in [\![1\ ;\ N]\!]} \in \mathbb{R}^N$, with N the dimension of the target vector space. In this section, we list the most used loss functions for regression, as we will need to describe training for a regression task in Chapter 4.

The most straightforward regression metric is the mean absolute error (MAE), which calculates the average of the absolute differences between a prediction and a ground truth label:

$$MAE(y, \hat{y}) = \frac{1}{N} \|\hat{y} - y\|_1 = \frac{1}{N} \sum_{i=1}^{N} |\hat{y}_i - y_i| \tag{2.2}$$

The MAE allows to quickly see the average difference between predictions and target labels, making it easily interpretable. By not squaring the differences, the MAE is less sensitive to outliers, but it can be too forgiving of large errors.

To answer this, the mean squared error (MSE) calculates the average of the squared differences between the prediction and the ground truth label:

$$MSE(y, \hat{y}) = \frac{1}{N} \|\hat{y} - y\|_2 = \frac{1}{N} \sum_{i=1}^{N} (\hat{y}_i - y_i)^2 \tag{2.3}$$

The MSE heavily penalizes large errors, which can be beneficial if large deviations are especially costly. The MSE is very widely used for regression problems, as a few large errors are often considered more serious than many small errors. This is often done in the interests of designing a more robust model, with more consistent outputs and errors.

The root mean squared error (RMSE) is the square root of the MSE, bringing the error back to the same units as the original target variable:

$$RMSE(y, \hat{y}) = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (\hat{y}_i - y_i)^2} \tag{2.4}$$

Like the MSE, it penalizes large errors more heavily due to squaring differences, but it is easier to interpret because it is in the same scale as the target variable. For this reason, the RMSE is a popular variation of the MSE.

The Huber loss combines features from both the MAE and the MSE. It squares differences up to a certain point, after which the loss increases linearly:

$$\begin{aligned} Huber(y,\hat{y}) &= \frac{1}{N}\sum_{i=1}^{N} H_\delta(\hat{y}_i - y_i) \\ \forall x \in \mathbb{R}, H_\delta(x) &= x^2 \text{ if } |x| \leq \delta \\ &= \delta * (2 * |x| - \delta) \text{ otherwise} \end{aligned} \tag{2.5}$$

The Huber loss allows to give more importance to larger errors compared to the MAE, but it is less sensitive to outliers compared to the MSE since errors are treated linearly after a threshold δ to be determined depending on the dataset.

A different metric might be used as a loss function depending on the problem. In practice, additionally to the loss functions, other evaluation metrics might be used to get a well-rounded understanding of the model's performance.

2.2.3 Evaluation metrics for classification

Classification problems involve predicting a categorical output. In binary classification, the goal is to determine whether an example belongs to the positive class (1) or the negative class (0). For multiclass classification, there can be multiple possible classes, and the model must predict which single class each example belongs to. In multilabel classification, however, an example can belong to multiple classes simultaneously, and the model must handle this possibility by allowing an arbitrary number of predicted classes for a single example. Classification problems often require multiple metrics to fully capture how well a model is performing [Powers, 2020]. While loss functions and evaluation metrics are generally the same for regression, in classification we distinguish loss functions that are piecewise differentiable and quantify the quality of a single prediction compared to a single ground truth, to evaluation metrics which in this case are often non-differentiable and can often only be applied to several examples at once. In this section we cover the evaluation metrics most often used for binary classification, as we will be evaluating a classification task in chapters 5 and 6.

Classification evaluation metrics quantify the error between an ensemble of predictions and an ensemble of ground truth labels. They are generally non-differentiable functions and thus are unsuitable for guiding the learning of a model since they are incapable of propagating gradients to optimize the model's parameters. They are instead used to assess the quality of the predictions given by a model over a dataset, evaluating several examples at once. The most commonly used classification evaluation metrics involve the concepts of true / false and positive / negative samples. True samples are the examples which were correctly predicted, and false samples are the ones that were not, regardless of whether they belong to the positive or the negative class. Additionally, positive samples are ones which were predicted to belong to the positive class, and negative samples are the ones which were predicted to belong to the negative class. From this are derived the groups of true positives, true negatives, false positives and false negatives. The amount of samples in each group are noted TP, TN, FP and FN respectively.

Accuracy represents the proportion of correctly classified instances, both positive and negative, out of all predictions:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.6}$$

Accuracy divides the sum of true examples by the sum of total examples in the dataset, representing the proportion examples that were correctly classified. It is easy to interpret, but can be misleading when the data is imbalanced since a trivial model labeling everything as the dominant class may have a high accuracy, and does not distinguish between the two types of errors, false positives and false negatives.

Specificity, or true negative rate, measures the model's ability to correctly identify cases in the negative class:

$$Specificity = \frac{TN}{TN + FP} \tag{2.7}$$

It is particularly important in scenarios where false positives are costly or disruptive, for example in certain medical tests. However, a model labeling everything as negative will have perfect specificity, and specificity does not address how well the model captures the positive class by itself.

Conversely, recall, or true positive rate, measures the model's ability to correctly identify samples in the positive class:

$$Recall = \frac{TP}{TP + FN} \tag{2.8}$$

It is critical in scenarios where missing a positive case is costly or dangerous, for example failing to diagnose a disease. A high recall means that the model makes few false negative errors. Yet, as for specificity with negative examples, a model can achieve perfect recall simply by labeling everything as positive, and recall does not assess the model's performance on the negative class.

Precision, or positive predictive value, is the proportion of predicted positives that are actually correct:

$$Precision = \frac{TP}{TP + FP} \tag{2.9}$$

A high precision means that the model makes few false positive errors, but does not account for false negatives by itself. Because of this, precision is often paired with recall.

Following this, the F-score is the harmonic mean of precision and recall, balancing both metrics into a single value.

$$F\text{-}score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{2.10}$$

It is helpful when both precision and recall are both important, which is often the case, and punishes extreme trade-offs (e.g. very high precision with very low recall, or vice versa). It also helps in the case of imbalanced data. However, it does not differentiate which is more important (precision or recall), in the considered problem.

Lastly, the Matthews correlation coefficient (MCC) is a correlation coefficient between the true and predicted labels. It ranges from -1 to 1. An MCC of 1 indicates perfect prediction, 0 is equivalent to random guessing and -1 indicates total disagreement.

$$MCC = \frac{(TP * TN) - (FP * FN)}{\sqrt{(TP + FP) * (TP + FN) * (TN + FP) * (TN * FN)}} \tag{2.11}$$

The MCC is considered a balanced metric that works even for imbalanced datasets, as it factors in all four confusion matrix groups (TP, TN, FP and FN). It provides a single measure of overall performance, but is less commonly used and less intuitive to interpret than metrics like precision, recall, or accuracy.

Ultimately, depending on the problem, the severity of false negative errors versus false positive errors may be different, and the dataset may not be well balanced. Because of this, some evaluation metrics may be more suited than others. Additionally, each classification metric helps assess different aspects of the model's performance, but each also has its limitations. Even using a harmonic mean of other metrics, such as the F-score, may not always provide a complete picture of the model's performance. For this reason, at least two complementary metrics are typically used together to provide a more comprehensive understanding of how well the model meets expectations.

2.2.4 Neural network architectures

2.2.4.1 MLP

A multi-layer perceptron (MLP) is one of the most fundamental neural network architectures in deep learning. It was first introduced by Frank Rosenblatt in 1958, laying the groundwork for modern deep learning [Rosenblatt, 1958]. It consists of multiple layers of nodes, referred to as neurons, where each layer is fully connected to the next by a matrix multiplication.

The MLP architecture starts with an input layer which receives a raw numerical observation $X_0 \in \mathbb{R}^{d_0}$ from a dataset, where X_0 is a vector containing d_0 features, with d_0 the dimension of the input vector space, each feature corresponding to a single input neuron.

The input layer is connected to the first hidden layer, and each hidden layer is connected to the next. An MLP operates in a feed-forward fashion, meaning data flows from the input layer to the hidden layers, and then to the output, without any feedback connections. An MLP is made only of fully connected layers, or dense layers, which are defined by the fact that all neurons in the previous layer are connected to all new neurons in the current layer. The MLP network can contain an arbitrary number of hidden layers, depending on the depth needed for the problem, and the number of neurons can vary in each hidden layer. Increasing the depth (i.e., the number of hidden layers) can allow the network to capture more complex patterns, while increasing the number of neurons in each layer gives more representational capacity within each layer. It has been shown that an MLP with even only one hidden layer and a sufficient number of neurons can approximate any continuous function on compact intervals. This is referred to as the "universal approximation theorem" [Hornik, 1991]. Let n be the number of layers in the network (counting the hidden layers and the output layer). Each hidden layer $i \in [\![1, n]\!]$ consists of multiple neurons that transform data from the previous layer through learned parameters, a matrix $W_i \in \mathbb{R}^{d_{i-1} \cdot d_i}$ and a bias $B_i \in \mathbb{R}^{d_i}$, with d_i the dimension of the vector space at layer i. After that, a non-linear activation function $f_i \in \mathbb{R}^{\mathbb{R}}$ is then applied to each element of the transformed data independently (element-wise), before it reaches layer i:

$$\forall i \in [\![1, n]\!], X_i = f_i(X_{i-1} W_i + B_i) \tag{2.12}$$

where X_{i-1} is the vector at the previous layer $i - 1$, W_i and B_i are the learned weights and biases in layer i, f_i is the activation function in layer i, and X_i is the resulting vector in layer i. It is crucial to use a non-linear activation to break the linearity of matrix multiplication. Without the presence of a non-linear activation function, successive layers would ultimately correspond to a single matrix multiplication, i.e. a single layer. Using a non-linear activation function allows to expand the space of learnable functions for the network, making it possible of learning increasingly complex and abstract relationships between the features in the data. Examples of usual activation functions f_i for the hidden layers include the ReLu (Rectified Linear Unit), sigmoid and hyperbolic tangent functions. At the output layer, the activation function depends on the problem and it is common to use sigmoid for a binary or multilabel classification problem, softmax for a multiclass classification problem, and sigmoid or linear for a regression problem depending on the type of target. Using sigmoid is good to direct data to the $[0, 1]$ interval, in the case of probabilities for example. Using a linear activation function is equivalent to using no activation function at all, and should only be allowed at the end of the last layer, as there is no subsequent layer where it is needed to break the linearity.

The last hidden layer is then connected to the output layer. For a network containing n layers, when the input vector X_0 has finished going through the network up to the end of the output layer, it reaches the final vector $X_n = \hat{y} \in \mathbb{R}^N$, where $N = d_n$ is the dimension of the target vector space. This is the forward pass, and the output vector

$\hat{y}$ is the final prediction of the model for the input X_0. The number of neurons in the output layer depends on the task. For example, a single neuron might be used for binary classification tasks, indicating the probability that the example belongs to the positive class, while multiple neurons (one per class) could be used for multiclass or multilabel classification problems. The MLP architecture is depicted in Fig 2.6 with two hidden layers.

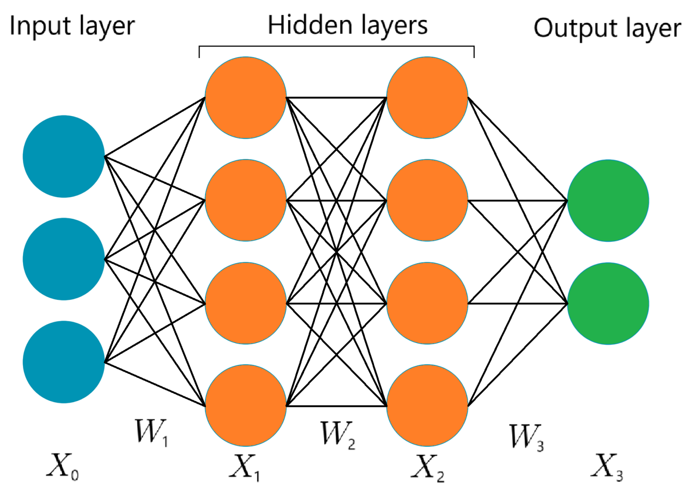

Figure 2.6: **MLP architecture.** The input layer is followed by the hidden layers, then the output layer. In each layer, all neurons are connected to the ones in the next layer. The forward pass is performed from left to right, and the backpropagation from right to left.

This describes the forward pass in the MLP architecture. After this, as has been explained in the previous sections, the model can be trained using a loss function and an optimizer to backpropagate the errors and update the model's parameters.

MLPs are frequently applied to tabular data, simple image classification tasks, or preliminary baselines in many machine learning problems. They are straightforward to implement and easier to interpret than most other deep learning architectures. They can be effective for a wide range of classification and regression tasks, especially if the data is well-structured. Additionally, MLPs provide a foundational understanding for more complex architectures which can account for spacial or sequential relationships in the data, which MLPs are incapable of.

2.2.4.2 CNN

A convolutional neural network (CNN) is a specialized neural network architecture designed primarily for processing data that have spacial structure, most notably images which can be viewed as 2D grids of pixels [LeCun, 2015]. CNNs extend the idea of local connectivity and weight sharing to capture spatial features more efficiently than fully connected networks, which makes them particularly well suited for tasks that require to detect spatial patterns in data, such as image classification, object detection or semantic segmentation [Krizhevsky, 2012]. While 2D CNNs are often used for image processing in computer vision, 1D CNNs are also widely employed in a time series analysis, audio processing and natural language processing tasks.

Unlike a fully connected layer, where every input is connected to every neuron, the neurons in a convolutional layer look at only a small region of the inputs around them at a time. This local connectivity substantially decreases the number of parameters and helps detect small spatial patterns, like edges or corners in images, which can then be combined into more complex patterns at deeper layers.

Another defining characteristic of convolutional layers is that the same set of weights, known as a kernel or filter, is used across different positions in the input. This drastically reduces the number of parameters compared to a fully connected layer, allowing the network to learn filters that are reused in multiple regions.

Weight sharing and local connectivity allow convolutional layers to keep only a fraction of the parameters in the network compared to fully connected layers. This is done under the assumption that relevant features in spatially-structured data should be found only locally, and should not depend on where they're located in the input. For example, if we want to detect a cat's ear, it has a particular shape that will only be found in local groups of pixels, and its shape will remain the same wherever it might be found in the image. Because of those two core characteristics, convolutional layers can detect features regardless of where they appear in the input. A filter that recognizes a particular shape can find that shape in multiple parts of an image without needing separate, position-specific parameters. This gives convolutional layers an important property: shift invariance.

In a convolutional layer, several different learnable filters, or kernels, are convolved across the input, each one producing a feature map that captures the response to that filter at different spatial positions. Common hyperparameters include the kernel size (e.g., 3×3 or 5×5), stride (the step with which the filter moves across the image), and the number of output channels (i.e., the number of filters). A dilation rate can also be defined to make filters evaluate pixels that are further apart, rather than consecutive pixels. An activation function, often ReLu, is applied to the feature maps after they have been computed by convolving the filters.

After a convolutional layer, a pooling layer is often used to reduce the spatial resolution of the feature maps, typically by taking a maximum or average over small patches [Boureau, 2010], respectively called max pooling and average pooling. This is an important step to reduce the dimensionality of the feature maps, making the network more computationally efficient and bringing the encoded representation to the desired size.

Regarding the whole network, most CNNs consist of a stack of alternating convolutional and pooling layers (the encoder), after which the resulting feature maps are usually flattened (turned into a 1D vector) to make the encoded representation, which is then passed to one or more fully connected layers (the decoder). The final layer in the decoder depends on the particular task and might end with a single neuron with sigmoid activation for binary classification, with several neurons with softmax activation for multiclass classification, with several neurons with sigmoid activation for multilabel classification, or with one or more neurons with sigmoid or linear activation for regression tasks. The CNN architecture is depicted in Fig 2.7.

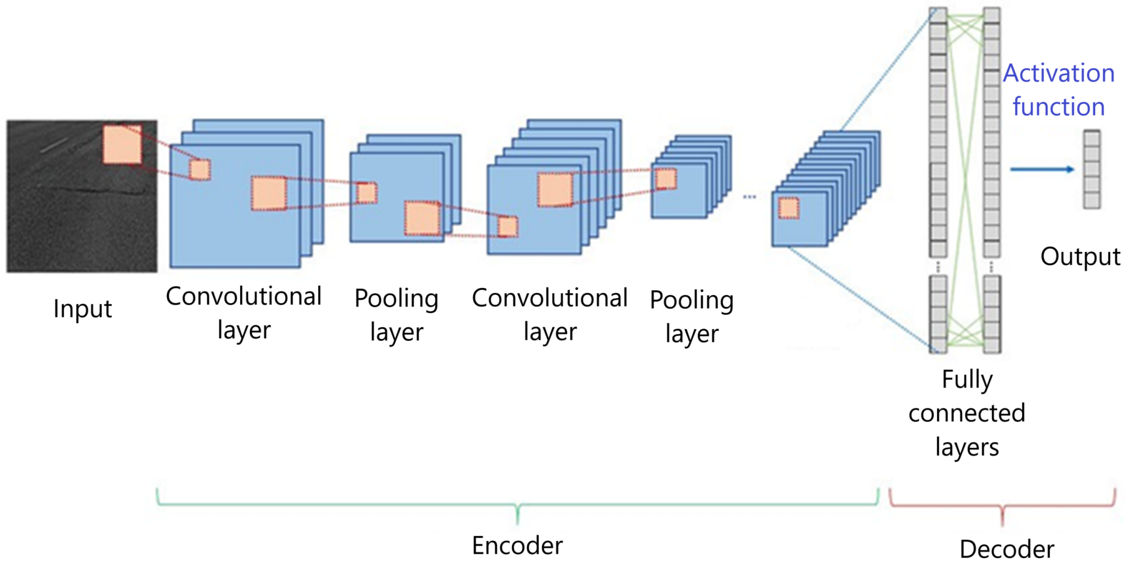

Figure 2.7: **CNN architecture.** The encoder is made of alternating convolutional and pooling layer and transforms the input into abstract features. The decoder then processes these features through fully connected layers to predict the output. Adapted from https://tinyurl.com/26pj9pfw.

More generally, in many different fields and for many different deep learning architectures, it is very common to have an encoder, which learns a relevant high-level abstract representation for the input data, followed by a decoder, which translates this high-level representation into the desired prediction. The decoder and the encoder might be considered together or separately. For example, the same encoder which encodes the input data into a useful representation might be used for a variety of different downstream tasks, each time paired with a different decoder relevant to the particular task [Devlin, 2019]. In representation learning, the goal is only to train a decoder to learn a relevant representation. This encoder can then be used on any particular task by training a decoder on this given task and pairing it with the pre-trained encoder.

Once again, such a CNN architecture can then be trained using a loss function and an optimizer to backpropagate the errors and update the model's parameters. CNNs have become a cornerstone of modern computer vision, providing powerful tools for tasks ranging from basic image classification to advanced applications such as image generation and video analysis.

2.2.4.3 RNN

Recurrent neural networks (RNNs) are a class of neural network architectures specifically designed to handle sequential data, such as time series, text, or any other input where elements follow a logical or temporal order [Rumelhart, 1986; Elman, 1990]. Unlike feed-forward networks that process inputs independently, RNNs incorporate "memory" of what has been processed before, enabling them to capture dependencies over time. In natural language processing, RNNs can process text one token at a time to perform sentiment analysis or machine translation. RNNs can be easily applied to time series analysis for tasks like forecasting stock prices or predicting sensor measurements over time. They are also well suited for other tasks based on sequential data such as audio processing, speech recognition, and video analysis, since all these tasks involve sequences where each element depends on its predecessors.

An RNN processes a sequence of vectors $(X_t)_{t\in[\![1,T]\!]} \in (\mathbb{R}^{d_0})^T$ step by step, with d_0 the dimension of the input vectors in the sequence, and T the length of the sequence. At each time step $t \in [\![1, T]\!]$, the network reads an input X_t and its hidden state $H_{t-1} \in \mathbb{R}^{d_h}$, and produces an output $\hat{y}_t \in \mathbb{R}^N$ and updates its hidden state $H_t \in \mathbb{R}^{d_h}$, with d_h and N the dimensions of the hidden and target vector spaces respectively. This hidden state carries forward information from previous steps, effectively giving the network a memory of past inputs:

$$\forall t \in [\![1, T]\!], H_t = f(H_{t-1}W_{hh} + X_tW_{xh} + B_h) \tag{2.13}$$

$$\forall t \in [\![1, T]\!], \hat{y}_t = g(H_tW_{ho} + B_o) \tag{2.14}$$

where H_{t-1} is the previous hidden state, X_t is the current input at time t, $W_{hh} \in \mathbb{R}^{d_h.d_h}$, $W_{xh} \in \mathbb{R}^{d_0.d_h}$ and $W_{ho} \in \mathbb{R}^{d_h.N}$ are weights, $B_h \in \mathbb{R}^{d_h}$, $B_o \in \mathbb{R}^N$ are biases, and f and g are non-linear activation functions of $\mathbb{R}^{\mathbb{R}}$, applied element-wise to the vectors. This recurrent formula allows the network to incorporate information from past inputs into its current internal state, additionally to the current input, to produce a new sequence of output vectors $(\hat{y}_t)_{t\in[\![1,T]\!]} \in (\mathbb{R}^N)^T$. A core property of RNNs is that the same recurrent cell is repeated at each input in a sequence, meaning that the parameters W_{hh}, W_{xh}, W_{ho}, B_h and B_o are shared among cells through time. This is done under the assumption that the key structural patterns in a given task based on sequential data do not depend on the time step, allowing to significantly reduce the number of parameters. The RNN architecture is depicted in Fig 2.8.

While the foundational RNN architecture uses a simple recurrent update, more advanced cells help tackle issues with training stability and memory retention. Indeed, plain RNNs often struggle to keep information over long sequences. Moreover, vanishing gradients (when the gradients become extremely small, making it difficult for the model to learn long-range dependencies) and exploding gradients (when gradients become very large, destabilizing training) are issues that may arise with RNNs. Solutions include gated RNN cells like Long Short-Term Memory (LSTM) [Hochreiter, 1997] and Gated Recurrent

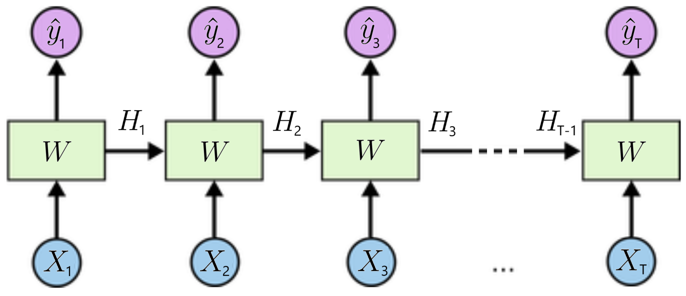

Figure 2.8: **RNN architecture.** At each time t, the input X_t and the current hidden state H_{t-1} are processed by the recurrent cell to produce the output $\hat{y}_t$ and the next hidden state H_t. Adapted from https://tinyurl.com/38kcmftz.

Unit (GRU) [Cho, 2014], which use gating mechanisms to control how information is stored, updated, and retrieved. These specialized cells manage how information flows within the network, helping address the problem of long-range dependencies. The RNN can also be bidirectional [Schuster, 1997], in which case two RNNs are run in opposite directions, combining forward and backward context. The forward hidden state $\overrightarrow{H_t}$ and the backward hidden state $\overleftarrow{H_t}$ are concatenated to form a richer representation H_t at each position t. This is often used when the entire sequence is available in advance, giving access to both left and right context for the prediction.

Additionally, there may be some variations in the architecture. For many-to-one tasks, a sequence of inputs is mapped to a single output, meaning that only the final output $\hat{y}_T$ is considered. An example of this category of problems is sentence analysis. For many-to-many tasks, a sequence of inputs is given and each one has a corresponding output. This is the most general case. An example is sequence labeling, where each word in a sentence can be assigned a label, for example identifying verbs in a sentence, or specific entities like organizations, dates or drugs. For one-to-many, a single input leads to a sequence of outputs. This is the case of text or music generation for example. Moreover, recurrent layers may be stacked in an RNN, meaning that the output vectors produced by a first recurrent layer may be passed as input vectors to a second recurrent layer, and so on. This can happen several times before a final sequence of vectors $(\hat{y}_t)_{t\in[\![1,T]\!]}$ is produced by the last recurrent layer at the end of the RNN.

When training RNNs, instead of computing gradients on a single forward pass as in feed-forward networks, the network must be unrolled over the entire sequence of time steps. Gradients are then backpropagated through time, starting from the last time step and moving backward to the first. At each step, the dependencies of the output on both the current input, parameters, and the hidden state from previous time steps are accounted for. This is called backpropagation through time (BPTT). However, training can be slow, especially when dealing with long input sequences.

2.2.4.4 Self-attention and Transformer

A Transformer is a neural network architecture introduced to handle sequences in a more efficient and scalable way than previous recurrent or convolution-based approaches [Vaswani, 2017]. Originally proposed for machine translation, Transformers have since become the backbone of many state-of-the-art models in natural language processing as well as in other domains such as computer vision, speech processing and multi-modal tasks with the advent of BERT-based models (Bidirectional Encoder Representations from Transformers) [Devlin, 2019].

Instead of relying on recurrence, processing inputs one at a time and carrying hidden states forward, the Transformer uses an attention mechanism to find relationships between any pair of elements in a sequence. Each token in the input sequence can be linked to any other token, regardless of its position, enabling the model to capture both short-range and long-range dependencies more effectively. This is called self-attention [Vaswani, 2017]. Because the Transformer does not process the sequence sequentially like RNNs, it needs a way to represent the order of the tokens in the input sequence. This is done through positional embeddings, which add information about the position of each token in the sequence, usually via sinusoidal encodings or learnable position vectors. The positional embeddings are concatenated to the input vectors at each point in the sequence before they enter the attention step. Let d_0 and d_{att} be the dimensions of the input vector space and the attention hidden vector space respectively. The model learns parameters W_Q, W_K and W_V, all matrices of $\mathbb{R}^{d_0 \cdot d_{att}}$, corresponding to the queries, keys and values respectively. To transform a sequence of input vectors $(X_t)_{t \in [\![1,T]\!]} \in (\mathbb{R}^{d_0})^T$, the queries, keys and values, all vectors of $\mathbb{R}^{d_{att}}$, are first computed at each position in the sequence:

$$\begin{aligned} \forall t \in [\![1,T]\!], Q_t &= X_t W_Q \\ \forall t \in [\![1,T]\!], K_t &= X_t W_K \\ \forall t \in [\![1,T]\!], V_t &= X_t W_V \end{aligned} \tag{2.15}$$

For each token in the input, the query indicates what it is looking for, the key represents what it has to offer to other tokens, and the value conveys its content or representation. Then, to determine how much each token should pay attention to another, the attention scores $(e_{t,u})_{t,u \in [\![1,T]\!]^2} \in \mathbb{R}^{T^2}$ are computed for every pair of tokens t and u with the dot product of the query Q_t and the key K_u, followed by a softmax so that the scores sum to 1:

$$\forall t, u \in [\![1,T]\!]^2, e_{t,u} = \frac{\langle Q_t, K_u \rangle}{\sum\limits_{v=1}^{T} \langle Q_t, K_v \rangle} \tag{2.16}$$

The attention score $e_{t,u}$ quantifies how much of a link there is from token t toward token u, showing if the token t is able to find what it seeks in token u. After that, at each position t, the attention scores $(e_{t,u})_{u \in [\![1,T]\!]}$ are used to weight the value vectors V_u at every position u

in the sequence, producing a weighted sum $Att_t \in \mathbb{R}^{d_{att}}$ that encodes the relevant context:

$$\forall t \in [\![1, T]\!], Att_t = \sum_{u=1}^{T} e_{t,u} V_u \tag{2.17}$$

Finally, a feed-forward step, typically consisting of two fully connected layers with a non-linear activation function in a between, is applied independently to each vector Att_t (meaning that the different positions are not interconnected at this step) to form the encoded vector $H_t \in \mathbb{R}^{d_h}$ at that position, with d_h the dimension of the hidden vector space at the end of the feed-forward step:

$$\forall t \in [\![1, T]\!], H_t = FF(Att_t) \tag{2.18}$$

where $FF \in (\mathbb{R}^{d_h})^{\mathbb{R}^{d_{att}}}$ denotes the feed-forward step. This process encodes the input vectors $(X_t)_{t \in [\![1,T]\!]}$ into a new sequence of hidden vectors $(H_t)_{t \in [\![1,T]\!]}$ which provide a more abstract representation able to capture different short-range and long-range contextual patterns in the sequence.

In practice, the Transformer splits the self-attention process into multiple units called attention heads. In that case, each different head k performs the operation described above independently to learn different aspects of the sequence relationships and produce attention hidden vectors $(Att_{t,k})_{t \in [\![1,T]\!]}$. These heads are then concatenated and combined to form the final attention hidden vectors $(Att_t)_{t \in [\![1,T]\!]}$ for the sequence. This multi-head design allows the model to capture various types of contextual information in parallel.

Additionally, it is possible to stack several alternating attention and feed-forwards steps in the encoder, in a way that the hidden layers $(H_t)_{t \in [\![1,T]\!]}$ obtained the first time are then used as the input sequence in a second same process to yield new hidden vectors at a deeper level for the sequence.

Once the encoder has produced the hidden layers $(H_t)_{t \in [\![1,T]\!]}$, it is followed by a decoder that can use that encoded representations to generate the desired predictions $(\hat{y}_t)_{t \in [\![1,T]\!]} \in (\mathbb{R}^N)^T$ at each point in the sequence depending on the considered task, where N is the dimension of the target vector space. Similarly to RNNs, many-to-one tasks will only require the last prediction $\hat{y}_T$, while one-to-many and many-to-many tasks will make use of the full $(\hat{y}_t)_{t \in [\![1,T]\!]}$ predictions. The Transformer architecture is depicted in Fig 2.9.

This encoder-decoder framework is especially useful in tasks like machine translation where one sequence, for example an english sentence, is encoded into a more universal representation containing the ideas and properties of the sentence independently of language, before being converted by the decoder to a french sentence for example. Some models like BERT consider only the encoder to develop versatile representations that can then be used for a variety of downstream tasks, or be readily employed for simpler tasks such as text classification, while others like GPT focus primarily on the decoder for generative tasks like text completion. Encoder-decoder models like T5 or BART are effective

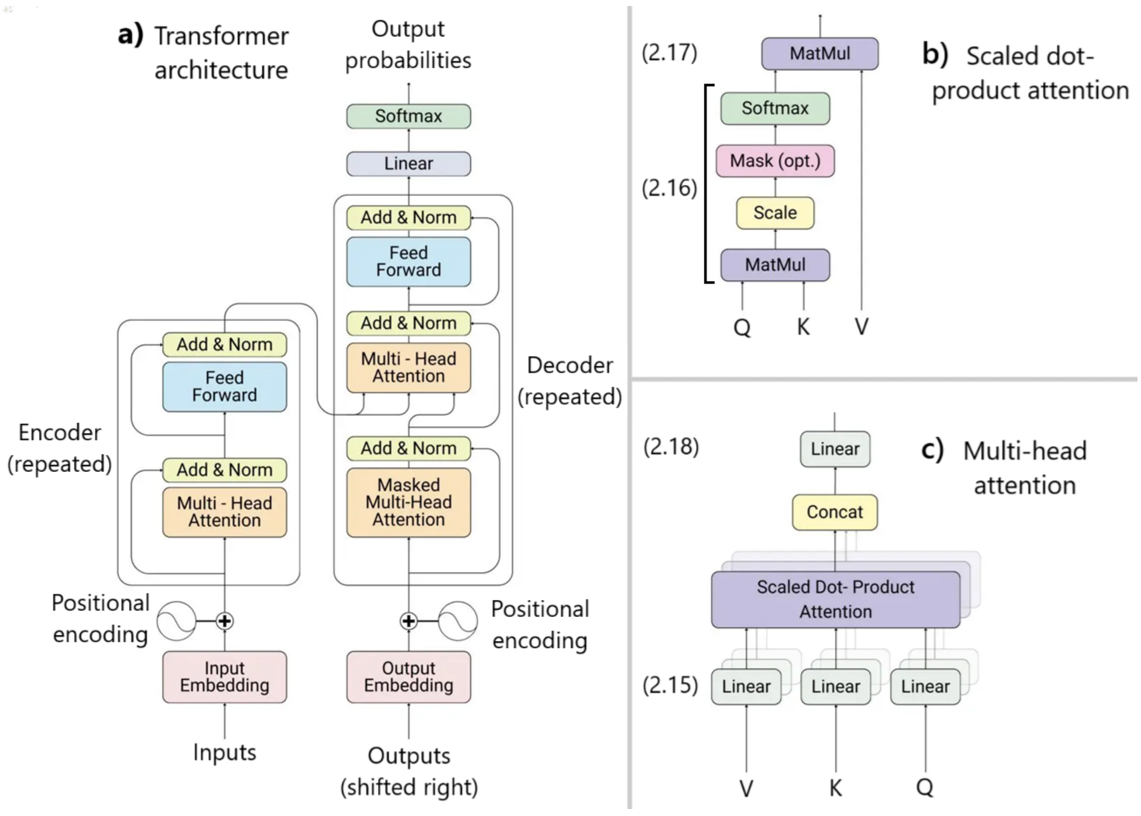

Figure 2.9: **Transformer architecture. a)** The Transformer architecture is made of an encoder and a decoder. Both repeat using **b)** multi-head attention layers, in which values, keys and queries go through linear transformations followed by **c)** scaled dot-product attention. References to equations 2.15, 2.16, 2.17 and 2.18 are indicated. Adapted from [Vaswani, 2017].

for tasks that map one sequence to another, including translation or summarization.

To stabilize training and help with gradient flow, residual connections (or skip connections) and layer normalization are often used. Residual connections consist in adding the input to the output of a layer, meaning that the layer only has to learn the difference to be added to the input instead of the entire output, which can facilitate learning in stacked layers, allowing deeper models to train more effectively and converge more quickly.

As always, a Transformer model might be trained by combining this forward pass process, embedding input tokens and adding positional embeddings before passing them to the encoder and the decoder, with a loss function and an optimizer to backpropagate gradients and optimize the model's parameters. Transformers are highly parallelizable and allow for more computationally efficient operations compared to RNNs, significantly speeding up training on modern hardware like GPUs or TPUs. They are effective in capturing global context and versatile across many tasks with minimal architectural changes, by pairing the same encoder with a different decoder relevant to the considered task.

However, self-attention scales quadratically with sequence length, making it computationally expensive to process very long sequences. Moreover, high amounts of training data are often needed to achieve state-of-the-art performance.

3

State-of-the-art: prediction of RNA secondary structure

3.1 Main types of approaches for RNA secondary structure prediction

In this section, we present the main types of RNA secondary structure prediction methods. These methods can be broadly divided into three groups: the thermodynamic approaches, the comparative analysis approaches and the deep learning approaches. Through these three categories, we aim to provide a comprehensive overview of the strategies employed in RNA secondary structure prediction.

3.1.1 The thermodynamic approach

In general, molecules arrange themselves in such a way as to reach a stable thermodynamic equilibrium, meaning that at a constant pressure and temperature, the molecules tend to no longer carry out reactions. A molecule reaches thermodynamic equilibrium when its free energy is minimized. The thermodynamic approaches, which rely on principles of energy minimization, aim to predict the RNA structure that is most thermodynamically stable. This stability is often determined by either minimizing the free energy directly (MFE methods), or by maximizing the expected accuracy (MEA methods) of the structure, given base-pairing probabilities that are derived from the free energy and the partition function.

3.1.1.1 Thermodynamic approach based on MFE

The first models that were developed for the prediction of RNA secondary structure are based on the concept of minimum free energy (MFE) [Zuker, 1981; Zuker, 1989a; Wuchty, 1999]. Such computational prediction methods rely on the hypothesis that an RNA molecule tends to fold into the structures with the lowest free energy to adopt the most thermodynamically stable configurations. Thus, these approaches seek to find the structure s that minimizes the free energy $E(s)$ in that state. This kind of approach allows to make predictions that are physically meaningful, and to provide an explicit energy value along with the predicted structure.

Several competing factors need to be balanced to minimize the free energy. Modern thermodynamic models decompose the RNA secondary structure into components to account for these different factors, assigning specific energy values to each:

- Base pairings contribute to decrease the free energy and improve the stability of the RNA by releasing energy. They are associated with negative energy parameters. Stronger base pairings, such as Watson-Crick and Wobble base pairs, contribute more to the stability of the RNA structure.
- Stacking base pairs in a stem provide even more stability to the structure and correspond to additional negative energy parameters.
- Loops (non-pairing regions such as bulge, hairpin, internal and multibranch loops) tend to increase the free energy and are therefore associated with positive energy parameters. These positive parameters act as penalties that depend on the loop size.

The parameters for these contributions are derived from experimental measurements and stored in energy tables used by the computational algorithms. The first energy parameters of pairings were determined by Salser in 1978 [Salser, 1978]. A major advancement in MFE estimation came with the introduction of the nearest neighbor model by Turner in 1988 [Turner, 1988], in which the energy parameters of a base pair depend not only on the pair itself, but also on its neighboring base pairs. Energy parameters for stems and loops were also introduced. Energy parameters have since been refined from year to year. New parameters have been added to account for other types of structural motifs and their nucleotide composition [Serra, 1995], in particular for pseudoknots [Andronescu, 2010], RNA duplexes [Xia, 1998] and chemical constraints [Mathews, 2004]. The parameters collected by Turner can be found in the Nearest Neighbor Database (NNDB) [Turner, 2010], which provides several versions of the parameters as well as methods for calculating the free energy of RNA structures.

Once the energy parameters are determined, the free energy of an RNA sequence running from nucleotides i to j can be expressed recursively with an equation of the form:

$$\forall i,j \in [\![1,L]\!]^2, MFE_{i,j} = \min(MFE_{i,j-1}, \min_{i\leq k\leq j-1}(MFE_{i,k-1} + MFE_{k+1,j-1} + \beta_{k,j})) \tag{3.1}$$

where L is the length of the RNA sequence, $MFE_{i,j}$ is the minimum free energy of the subsequence running from nucleotides i to j, and $\beta_{k,j}$ is the energy gained from pairing nucleotides k and j, based on the energy parameters. Assuming that $MFE_{i,j-1}$ is known, the nucleotide j is added to the subsequence. The first term $MFE_{i,j-1}$ corresponds to the case where j is unpaired, and the second term $\min_{i\leq k\leq j-1}(MFE_{i,k-1}+MFE_{k+1,j-1}+\beta_{k,j})$ corresponds to the case where j is paired to a nucleotide k in the subsequence.

Such an equation can be employed to find the structure that minimizes the free energy using dynamic programming algorithms, such as the Zuker algorithm [Zuker, 1981]. After recursively using the previous equation to compute the MFE for the entire sequence, $MFE_{1,L}$, the steps that were taken can be traced back to identify the base pairings that led to the MFE and find the optimal structure. Along with the improvement of the energy parameters, new developments were later also brought to the dynamic programming algorithms used to compute the MFE structure, making them able to consider several suboptimal structures [Zuker, 1989b] instead of only the optimal structure, and decreasing the time complexity to $O(n^2)$ for modern algorithm instead of $O(n^4)$ for the original Zuker algorithm.

3.1.1.2 Thermodynamic approach based on MEA

In 1990, McCaskill introduced a dynamic programming algorithm for computing the partition function of an RNA sequence [McCaskill, 1990], enabling a new kind of approaches based on maximum expected accuracy (MEA) for the prediction of RNA secondary structure. These methods seek to find the structure that maximizes some base-pairing probabilities estimated from the free energy of the structures, reflecting both accuracy and thermodynamic feasibility, unlike the methods based on the MFE that seek to directly find the structure of lowest free energy. Following the Boltzmann distribution in statistical mechanics, the probability of a structure s to occur, $P(s)$, is assumed to be proportional to $e^{-\frac{E(s)}{RT}}$, where $E(s)$ is the free energy in structure s, R is the gas constant and T is the temperature. The first step is to compute the partition function Z, which is a constant that sums this quantity over all possible structures:

$$Z = \sum_{s\in S} e^{-\frac{E(s)}{RT}} \tag{3.2}$$

where S is the set of possible structures for the sequence. Therefore, the probability of a structure s to occur, $P(s)$, is as follows:

$$\forall s \in S, P(s) = \frac{e^{-\frac{E(s)}{RT}}}{Z} \tag{3.3}$$

Then, the probability that two nucleotides i and j are paired, $P_{i,j}$, can be expressed as the sum of $P(s)$ for when i and j are paired in the structure s:

$$\forall i,j \in [\![1,L]\!]^2, P_{i,j} = \sum_{\substack{s\in S\\(i,j)\in s}} P(s) \tag{3.4}$$

where L is the length of the sequence. The probability of a nucleotide i to be unpaired, P_i, becomes:

$$\forall i \in [\![1,L]\!], P_i = 1 - \sum_{\substack{1\leq j\leq L\\ j\neq i}} P_{i,j} \tag{3.5}$$

The expected accuracy $A(s)$ of an RNA sequence in the structure s is the sum of base-pairing probabilities for paired and unpaired nucleotides in the structure s. Once the base-pairing probabilities $(P_{i,j})_{i,j\in[\![1,T]\!]^2}$ have been computed, the expected accuracy can be expressed as:

$$\forall s \in S, A(s) = 2\sum_{\substack{(i,j)\in s\\ i<j}} P_{i,j} + \sum_{i\in \text{unpaired}(s)} P_i \tag{3.6}$$

and following this, the MEA of an RNA subsequence running from nucleotides i to j can be expressed recursively with an equation of the form:

$$\forall i,j \in [\![1,L]\!]^2, MEA_{i,j} = \max(MEA_{i,j-1}+P_j, \max_{i\leq k\leq j-1}(MEA_{i,k-1}+MEA_{k+1,j-1}+2P_{k,j})) \tag{3.7}$$

where $MEA_{i,j}$ is the MEA of the subsequence running from nucleotides i to j. Assuming that $MEA_{i,j-1}$ is known, the nucleotide j is added to the subsequence. The first term $MEA_{i,j-1} + P_j$ corresponds to the case where j is unpaired, and the second term $\max_{i\leq k\leq j-1}(MEA_{i,k-1} + MEA_{k+1,j-1} + P_{k,j})$ corresponds to the case where j is paired to a nucleotide k in the subsequence. A dynamic programming algorithm is then used based on the previous equation to find the MEA of the whole sequence, $MEA_{1,L}$, and the associated optimal structure. Additionally, the first term in the equation can be multiplied by a weight $\gamma > 0$ to adjust the relative importance of paired versus unpaired nucleotides, giving more flexibility to the approach in that it is adaptable to different biological contexts.

Furthermore, the partition function can also be used to sample a wide range of suboptimal structures, providing insights into the various possible conformations of a given RNA, rather than focusing solely on the optimal structure. This is especially valuable since

RNAs are flexible molecules that can adopt diverse conformations depending on the cellular context. Using conditional base-pairing probabilities from the partition function, a recursive sampling process can be employed to statistically sample suboptimal structures from the complete Boltzmann ensemble [Ding, 2003; Mathews, 2006; Waldispühl, 2007].

3.1.2 The comparative analysis approach

The comparative analysis approach to RNA secondary structure prediction relies on examining multiple, evolutionarily related RNA sequences to identify conserved patterns. The underlying assumption is that these patterns that are maintained across organisms point to elements in the sequence that are structurally important. Indeed, if an RNA molecule plays a critical role in the cell, its secondary structure tends to be conserved during evolution. For instance, a functional hairpin loop may remain in the same place throughout various organisms, even though the underlying nucleotide sequences may have undergone mutations.

Comparative sequence analysis begins by collecting homologous RNA sequences from different species or strains. These sequences are then aligned so that corresponding nucleotides, which may serve analogous structural or functional roles, are placed at the same positions. Positions that remain relatively unchanged across all sequences often imply a critical functional or structural role such as a catalytic site. Additionally, when two nucleotides form a base pair, compensatory mutations can occur. This means that a change at one position can lead to a complementary change at the pairing position so that the pair remains stable. For example, if there is a G-C pair between two nucleotides, and if a mutation G $\rightarrow$ A happens at the first position, a complementary mutation C $\rightarrow$ U may happen at the second position so that the pair remains stable in the form of A-U. This pattern of compensatory mutations is strong evidence that those nucleotides are indeed paired in the RNA structure [Hancock, 1988; Knies, 2008]. For this reason, it is also important to identify any two positions in the sequence alignment that vary but do so in a correlated manner showing compensatory mutations, because they are likely an indication of a base pair that is conserved across the alignment and that could have a significant structural role.

Once the alignment is in place, algorithms can identify conserved regions and pairs of alignment columns with correlated mutations [Coventry, 2004; Dutheil, 2012; Eddy, 2014; Rivas, 2021]. Statistical and heuristic methods often quantify how strongly changes in one alignment position correlate with changes in another. A high score suggests that those two positions likely form a base pair. Then, a consensus secondary structure can be built to explain all the conserved and covarying positions that were identified in the sequence alignment.

Because it draws on real evolutionary changes rather than relying solely on thermodynamic folding predictions, and because it uses all the information contained in multiple aligned RNA sequences rather than a single RNA sequence, comparative sequence analysis often yields highly reliable results. However, in order to identify compensatory mutations, enough and sufficiently diverse homologous sequences are required. Evidently, if no homologous sequences are available, comparative analysis methods cannot be employed. The comparative RNA web (CRW) [Cannone, 2002] database compiles high-quality sequence alignments of RNA sequences from various species, along with their secondary structure annotations, enabling researchers to study comparative analysis methods from RNA conservation and variability data across different organisms.

3.1.3 The deep learning approach

Deep neural networks can be used to predict RNA secondary structures by learning complex patterns in large datasets that traditional methods might miss. It is also possible for deep learning methods, regardless of the architecture used, to incorporate thermodynamics features into the loss function during training. This can help guide the model toward predictions that are more physically meaningful, since they account for experimentally validated parameters. Deep learning methods for the prediction of RNA secondary structure generally rely on either RNNs, CNNs, both, or Transformers. In this section, we discuss these different types of deep learning approaches.

3.1.3.1 Using an RNN architecture

RNNs can learn patterns of nucleotide interactions through sequential processing, capturing dependencies that help determine which bases are paired. Each RNA sequence is first encoded into a series of input vectors, where each vector typically corresponds to a nucleotide. Common approaches include a one-hot representation of the nucleotide (A, C, G or U) or an embedding layer that projects each nucleotide into a learnable vector space. Alternatively, the input vectors can be used for encoding k-mers, which are subsequences of length k within the original sequence, rather than the nucleotides. For example, 2-mers of AUCGU are AU, UC, CG and GU. Using 1-mers is equivalent to simply using nucleotides. Depending on the approach, positional embeddings can also be concatenated to the input vectors.

The neural network then processes the input sequence with recurrent layers. Several recurrent layers can be stacked to learn more complex patterns, at the cost of training and prediction speed. Gated recurrent layers such as LSTM [Hochreiter, 1997] or GRU [Cho, 2014] layers are commonly used in place of plain recurrent layers to improve the ability to capture long-range dependencies. Using its hidden state, the model can integrate context from previously seen nucleotides when evaluating the structural role of the current nucleotide. To capture both upstream and downstream context, a bidirectional RNN is often used.

After the RNN encodes context for each nucleotide in the sequence, the model must predict which bases are paired. The most common approach is to predict a pairing probability matrix $P \in \mathbb{R}^{L.L}$, where L is the length of the RNA sequence, such that for every pair of nucleotides i and j in the sequence, P_{ij} is the predicted probability that the two bases are paired. To generate this matrix, for each pair of nucleotides i and j in the sequence, the representations of the two bases encoded by the RNN are concatenated. The resulting vector is processed by a decoder, typically a feed-forward network, to yield the probability P_{ij} that nucleotides i and j are paired. In some cases, the objective can be to only predict which nucleotides are paired or not, but not to predict to which other nucleotides they are paired, for example to generate synthetic SHAPE data [Willmott, 2020]. In this case, the decoder will only have to process the encoded representation at position i to predict the probability P_i that nucleotide i is paired, although this is not enough in itself to predict a secondary structure.

Once the pairing probability matrix P is predicted, a threshold of 0.5 could be applied over the matrix to decide which bases are predicted to be paired or not. However, because an actual contact matrix must follow certain rules, this may not lead to a valid secondary structure. Indeed, although the matrix P gives probabilities, it does not inherently guarantee a valid RNA secondary structure. In particular, (i) the contact matrix must be symmetrical (if i pairs with j, then j also pairs with i), and (ii) any nucleotide should be paired with at most one other nucleotide. To ensure that the predicted secondary structure is valid, the pairing probability matrix P can instead be fed into a constraint-based algorithm, often a Zuker-like dynamic programming algorithm. Such an algorithm is able to maximize a quantity, for example the sum of $\log P_{ij}$ for positive pairs plus the sum of $\log(1 - P_{ij})$ for negative pairs, while ensuring the desired constraints.

3.1.3.2 Using a 2D CNN architecture

Many methods for the prediction of RNA secondary structure use 2D CNNs to take advantage of the fact that secondary structure can be seen as relationships between pairs of nucleotides. Using an approach based on a 2D CNN rather than a RNN allows to handle pairwise relationships directly to capture interactions between distant sequence positions, although it also means that the computation time grows in $O(n^2)$ with sequence length, leading to high computational costs for long RNAs.

First, the sequence needs to be mapped to an input for the model. Since 2D CNNs operate on a 2D grid, a grid of size LxL should be created, with L the length of the sequence, where every cell (i, j) encapsulates features about the relationship between the pair of nucleotides i and j. Formally, a sequence is typically mapped to a one-hot input matrix of size LxLxk for the different pairs of nucleotides, where k is the number of channels for the chosen representation. A commonly chosen representation is to indicate for every pair of nucleotides (i, j) if the pair corresponds to GC, CG, AU, UA, GU, UG, or something else (in which case it would not be able to form a canonical base pair). In this scenario, the input representation would contain 7 channels. Other representations can be employed to account for possible non-canonical base pairs.

A 2D convolutional encoder then processes the input matrix into encoded feature maps. This way, the network is able to learn local, and eventually global, patterns in the pairwise feature map that are indicative of base pairing. Multiple convolutional layers can be stacked in the encoder, generally separated by pooling layers, to expand the receptive field of the network and let it detect more global patterns, for example showing how nucleotides might be arranged in stems or loops. After passing through the CNN, each cell (i, j) is represented by a feature vector containing information on whether the nucleotides i and j likely pair or not.

Then, these feature maps go through the decoder, generally a feed-forward network applied independently in each cell (i, j), to output a pairing probability matrix P of size LxL. This matrix indicates in each cell (i, j) the predicted probability that the nucleotides i and j are paired. Lastly, as has been described previously, it is necessary to process the predicted pairing probability matrix P using a dynamic programming algorithm in order to yield a valid secondary structure for the prediction.

3.1.3.3 Using a combination of RNN and 2D CNN layers

Deep learning architectures for RNA secondary structure prediction can combine recurrent layers and 2D convolutional layers to take advantage of the strengths of both sequential and pairwise modeling. An RNN, usually using bidirectional GRU or LSTM layers, first processes the RNA sequence to capture sequential dependencies. The L resulting embeddings, each representing a nucleotide in the sequence, then form a 2D matrix of size LxLxk where every cell (i, j) contains the concatenation of the embeddings representing the nucleotides i and j, where L is the length of the sequence and k is the obtained number of channels (twice the size of the RNN embeddings).

This matrix is then given as input to the 2D CNN part of the model. The CNN scans this matrix and learns filters that detect spatial patterns indicative of base pairing or structural motifs. As before, the CNN then outputs a predicted pairing probability matrix P, which is finally processed by a dynamic programming algorithm in order to find the closest valid secondary structure. This allows to benefit from the pairwise processing of the 2D CNN approach, and to use informative features obtained sequentially from the RNN as input for the CNN rather than simply one-hot encodings, to capture both the linear context and the pairwise interactions.

3.1.3.4 Using a Transformer architecture

Transformers can be adapted for the prediction of RNA secondary structure, similarly to RNNs. In the same way, a sequence is first mapped to input vectors, each corresponding to a nucleotide or k-mer, using one-hot encodings or learnable embeddings. Compared to the RNN approach, positional embeddings are added to the input vector when using a Transformer to keep track of the order in the sequence. The Transformer then makes use of the self-attention mechanism to encode the input vectors into a more abstract representation, learning queries, keys and values to capture long-range relationships between nucleotides. Several Transformer layers can be stacked. Each layer refines the representation of every nucleotide using context from the entire sequence, progressively capturing more complex structural patterns.

Like in the RNN approach, after the final encoder layer, every pair of the encoded embeddings corresponding to positions i and j are concatenated to form a pairwise matrix representation. A feed-forward network is then applied to each cell (i, j) of this matrix independently, each time yielding a probability P_{ij} that nucleotides i and j are paired, leading to a pairing probability matrix P. Lastly, as with other deep learning methods for RNA folding, the predicted pairing probability matrix P generally requires to be processed by a dynamic programming algorithm to enforce necessary constraints and output a valid secondary structure.

Compared to the RNN approach, a model based on a Transformer architecture could be better at relating distant parts of a sequence, thereby learning long-range interactions, since the self-attention mechanism allows to directly consider relationships between every pair of nucleotides in the sequence. However, this leads to a time complexity of $O(n^2)$, which may pose computational challenges for long RNAs, whereas RNNs can compute embeddings in a $O(n)$ time complexity.

3.2 Tools for RNA secondary structure prediction

Predicting RNA secondary structures involves various methods and tools. Pseudoknots, which give insights into the tertiary structure, were often excluded by early algorithms. This is because accounting for pseudoknots significantly increases the time and space complexity for the algorithms. Early methods also struggled with long RNAs due to computational limitations and a lack of sufficient data. A chronological overview of tools for RNA secondary structure prediction is presented in Fig 3.1. The tools are categorized depending on their methodological approach and whether they predict pseudoknots. While the vast majority of earlier methods relied on thermodynamics, a major shift started in 2019 with the rise of deep learning approaches.

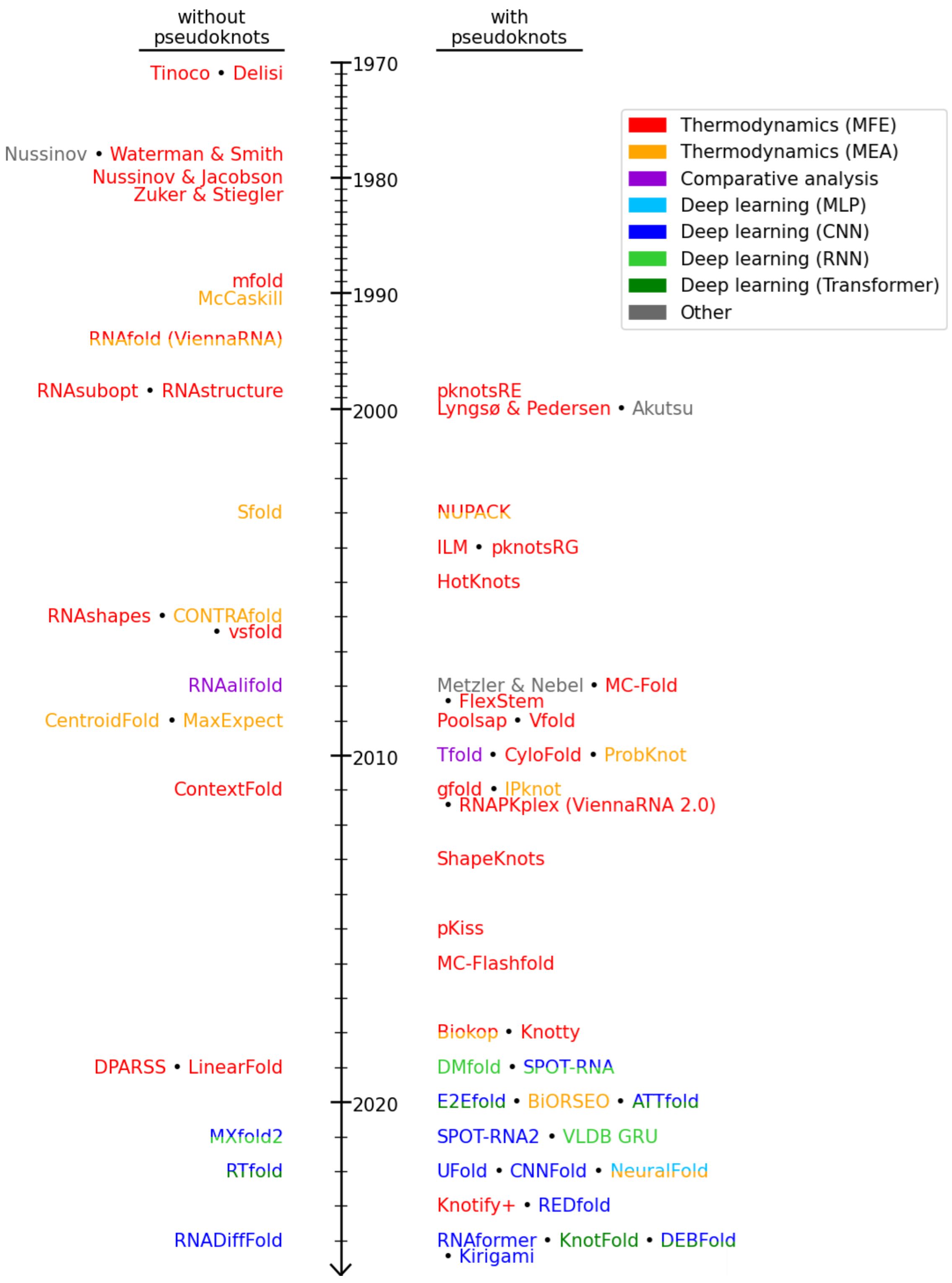

Figure 3.1: **Chronological overview of tools for RNA secondary structure prediction.** Tools that do not predict pseudoknots are displayed on the left, while those that do are shown on the right. The tools are color-coded according to their methodological approach. Tools that combine multiple approaches are indicated using the corresponding colors.

We first review tools designed for secondary structure prediction excluding pseudoknots, and then those that are able to predict pseudoknots. We specify which methods are designed for long RNAs (longer than 1,000 nt).

3.2.1 Tools for RNA secondary structure prediction excluding pseudoknots

Early work on RNA secondary structure starting in the 1950s [Doty, 1959] led to the development of methods that maximize pairings and incorporate thermodynamic principles [Delisi, 1971; Tinoco, 1971; Tinoco, 1973] to estimate an approximate secondary structure. Such approaches based on MFE have continually advanced with the improvement of free energy models and parameters, described in Section 3.1.1.1.

In 1978, Nussinov introduced the first dynamic programming algorithm for predicting RNA secondary structure by recursively maximizing the number of base pairs [Nussinov, 1978]. This algorithm has a time complexity of $O(n^3)$ in regard to the sequence length. This algorithm was further improved in 1980 by Nussinov and Jacobson, incorporating MFE [Nussinov, 1980].

Also in 1978, Waterman and Smith developed an algorithm to predict the MFE structure of RNA [Waterman, 1978]. Up to this point, algorithms do not account for suboptimal structures, only predicting the optimal structure relatively to the quantity to be maximized.

In 1981, Zuker and Stiegler developed a dynamic programming algorithm [Zuker, 1981] to predict the MFE structure of RNA from energy parameters. This dynamic programming algorithm if often described as the Zuker algorithm, or Zuker-style dynamic programming. The energy parameters were later refined, leading to the tool mfold in 1989 [Zuker, 1989a]. The algorithm was further enhanced by Zuker to be the first to include predictions of suboptimal structures [Zuker, 1989b]. In the same year, Jaeger further improved mfold's accuracy [Jaeger, 1989]. Mfold was later made accessible as a web server [Zuker, 2003]. Although the mfold web server is no longer supported, it is now hosted on the UNAFold web server [Markham, 2008].

In 1990, McCaskill introduced an algorithm to calculate the partition function of an RNA sequence [McCaskill, 1990], enabling new MEA methods which seek to determine the structure maximizing expected accuracy from computed pairing probabilities, as described in Section 3.1.1.2.

In 1994, researchers at the University of Vienna introduced the ViennaRNA software package [Hofacker, 1994], which contains tools for predicting RNA secondary structure. This includes RNAfold, which uses a Zuker-style dynamic programming algorithm to find the optimal MFE structure, or the MEA structure using McCaskill's partition function computation algorithm [McCaskill, 1990], depending on the option used. It is also inspired by the mfold algorithm. However, RNAfold does not generate suboptimal structures. Further works were later brought to the ViennaRNA package to include a web server in 2003 [Hofacker, 2003], as well as updated energy parameters and new features in 2011 [Lorenz, 2011].

Also in 1994, the favorable contribution of coaxial stacking of stems in free energy was demonstrated [Walter, 1994]. Accounting for this structural motif in the computation of free energy, which was not considered until then, another RNA folding tool based on the MFE called AllSub [Wuchty, 1999] was developed by Wuchty in 1999 to generate a range of suboptimal structures, and integrated into the ViennaRNA [Hofacker, 1994; Lorenz, 2011] package as the RNAsubopt tool.

RNAstructure [Mathews, 1997; Mathews, 1999] is a software package developed by Mathews and the University of Rochester team. It uses thermodynamics with nearest neighbor parameters from the Turner group. It includes methods for secondary structure prediction and prediction of base pair probabilities. RNAstructure was later further developed to include a user-friendly graphical interface along with updated tools [Reuter, 2010].

In 2003, Ding and Lawrence emphasized that RNA can fold into multiple conformations. They developed an algorithm [Ding, 2003] to sample structures from the Boltzmann ensemble, based on pairing probabilities derived from the partition function in the MEA approach. This algorithm led to the creation of the Sfold tool, which is available as a web server [Ding, 2004]. The algorithm was also added to the stochastic tool from RNAstructure [Mathews, 1997; Mathews, 1999] and to the RNAsubopt tool from ViennaRNA [Hofacker, 1994; Lorenz, 2011].

In 2006, Dawson introduced the tool vsfold [Dawson, 2006] for calculating the free energy of long-distance interactions. This model is based on the cumulative sum of pairing energies, with energy values weighted inversely to the length of the pairing in the RNA sequence. The model has since been refined multiple times [Dawson, 2013; Dawson, 2014; Dawson, 2015] for various improvements, including the ability to identify suboptimal structures.

Developed in 2006, CONTRAfold [Do, 2006] is a tool based on a probabilistic framework known as the conditional log-linear model. This model learns conditional base-pairing probabilities from training data and predicts RNA secondary structure by maximizing the expected accuracy.

Steffen introduced the RNAshapes [Steffen, 2006] software package in 2006, which contains RNA analysis tools based on the abstract shapes approach and can be applied to the secondary structure prediction of RNA using an MFE model. Unlike other tools, RNAshapes groups similar predicted structures together, reducing redundancy in the results.

In 2008, Bernhart introduced RNAalifold [Bernhart, 2008], a comparative sequence analysis algorithm to predict a consensus MFE structure given an ensemble of related RNA sequences. RNAalifold is inspired by covariance analysis between columns in the alignment, which detects conservation and covariation patterns indicative of core structural elements and compensatory mutations.

MaxExpect [Lu, 2009], a tool from the RNAstructure software package was introduced in 2009 by Lu to predict the secondary structure of RNA using the MEA approach, maximizing the expected accuracy. MaxExpect can be used to generate suboptimal structures. This tool shows that the MEA approach performs well and is able to provide better predictions than the historical MFE models.

Following on the encouraging results of the MEA approach, CentroidFold [Hamada, 2009] was developed by Hamada in 2009 and proposes new estimators for the expected accuracy to better predict the secondary structure of a single RNA sequence, or of an alignment of RNA sequences. CentroidFold was later made easily available as a web server [Sato, 2009] to output a secondary structure along with a graphical representation.

In 2011, ContextFold [Zakov, 2011] is introduced by Zakov. ContextFold is an algorithm for learning energy parameters from data. It is shown that a rich parametrization, significantly increasing the number of considered parameters for the different structural patterns compared to the previous approaches, improves the accuracy of the RNA secondary structure prediction.

In 2019, the dynamic programming algorithm DPARSS [Jiang, 2019] was introduced, enabling a broader exploration of the RNA secondary structure search space compared to traditional dynamic programming tools minimizing the free energy. It is particularly efficient for predicting tRNA structures.

Aiming to predict the secondary structures of long RNA sequences, LinearFold [Huang, 2019] is a low complexity model developed in 2019 and based on an enhancement of RNAfold [Hofacker, 1994; Lorenz, 2011] to improve the computation time. It has a time complexity of $O(n)$, allowing it to process long RNAs.

In 2021, MXfold2 [Sato, 2021] was developed by Sato to predict the secondary structure of RNA based on a deep learning model with thermodynamic integration. The model is composed both of recurrent and 2D convolutional layers, combining advantages of both. Thermodynamic features are added to the loss function to guide the training of the model, leading to more physically meaningful predictions.

Developed in 2022, RTfold [Jung,] is another deep learning model based on a combination of 1D convolutional layers and self-attention layers for the prediction of RNA secondary structure. However, RTfold is unable to predict pseudoknots and is intended for RNAs shorter than 500 nt.

In late 2024, following the success of diffusion models in deep learning, RNADiffFold [Wang, 2025] was introduced as a generative prediction approach of RNA secondary structure based on multinomial diffusion. In this model, starting from a noise-infused state, a denoising model is trained to progressively refine the pairing probability matrix to resemble the true contact map, similarly to a pixel-wise segmentation task.

Among secondary structure prediction tools that do not predict pseudoknots, while LinearFold is the only one specifically designed for long RNAs past 1,000 nt, others like RNAfold or MXfold2 still have a computational complexity that is low enough to allow them to be applied to long RNAs.

3.2.2 Tools for RNA secondary structure prediction including pseudoknots

Pseudoknots pose significant computational challenges for RNA secondary structure prediction. Dynamic programming algorithms that successfully handle nested (non-pseudoknotted) structures often come with a higher time complexity when crossing base pairs are included [Rivas, 1999]. Yet, with the gradual improvement in the accuracy and complexity of the models for RNA secondary structure prediction, more and more tools started considering pseudoknots as part the prediction problem. While the prediction of optimal secondary structure including any types of pseudoknots is very computationally expensive, prediction tools circumvent this issue by either restricting themselves to only certain types of pseudoknots (often the H-type) [Rivas, 1999] or using approximate approaches to handle more diverse pseudoknots without exhaustive enumeration.

In 1999, Rivas and Eddy developed a dynamic programming algorithm, PKNOTS, with a time complexity of $O(n^6)$ for the prediction of RNA secondary structure with pseudoknots [Rivas, 1999] based on the MFE. Their approach uses standard thermodynamic energy parameters along with parameters specific to pseudoknots. The class of pseudoknots predicted by PKNOTS is the most general among RNA structure prediction tools and thus the slowest to solve. PKNOTS does not generate suboptimal, structures and is no longer available.

In 2000, Akutsu introduced another dynamic programming algorithm [Akutsu, 2000] with a time complexity of $O(n^4)$, designed to maximize the number of pairings including pseudoknots. The algorithm considers the pseudoknots where the bases in any two stems are located arbitrarily.

In the same year, Lyngsø and Pedersen developed an algorithm [Lyngsø, 2000] based on the same thermodynamic model used by Rivas and Eddy. The algorithm considers a smaller class of pseudoknots and finds the optimal pseudoknotted MFE structure in $O(n^5)$. Lyngsø and Pedersen also show that predicting the MFE secondary structure including any pseudoknots is an NP-complete problem [Lyngsø, 2004].

In 2003, Dirks and Pierce developed an algorithm capable of calculating both the MFE structure including pseudoknots, and the partition function [Dirks, 2003]. The algorithm has a time complexity of $O(n^5)$ and identifies H-type pseudoknots. This algorithm is implemented in the NUPACK software package [Zadeh, 2011] and is able to generate suboptimal structures.

In 2004, Ruan developed a tool called ILM [Ruan, 2004], which predicts RNA secondary structures including pseudoknots. The algorithm is based on the MFE and extends the Nussinov loop matching algorithm [Nussinov, 1978] to incorporate pseudoknot prediction. ILM can be used for single RNA sequences, but can also consider covariation from comparative sequence analysis in the case of sequence alignments.

In 2004, Reeder and Giegerich introduced an algorithm [Reeder, 2004] with a time complexity of $O(n^4)$ for predicting pseudoknots in a specific subset of H-type pseudoknots, known as simple recursive canonicals, under the MFE model. In these structures, the two ends of each stem must be of equal length and free of internal loops, meaning that all possible pairings must occur. The algorithm is implemented in the tool pknotsRG.

HotKnots [Ren, 2005] is a tool developed in 2005 that uses a heuristic approach to predict the secondary structure including pseudoknots. This method iteratively searches for stable stems based on an energy minimization algorithm.

The tool FlexStem [Chen, 2008], introduced in 2008, also employs a heuristic approach based on energy minimization and the RNA folding mechanism. This method simulates the folding process by iteratively adding stems of maximum length.

MC-Fold [Parisien, 2008], also developed in 2008, introduces a novel prediction model in which RNA structure is defined from sets of nucleotide cyclic motifs instead of the Watson-Crick and Wobble base pairs. These motifs offer additional contextual information around each nucleotide and define a new structure evaluation function for free energy minimization. MC-Fold can predict H-type pseudoknots, identify non-canonical pairings, and generate suboptimal secondary structures.

In 2008, Metzler and Nebel developed a sampling algorithm for generating pseudoknotted structures [Metzler, 2008]. Their algorithm uses the Monte Carlo method with Markov chains, enabling them to sample structures based on a distribution computed with a probabilistic out-of-context grammar. This approach allows for the generation of structures without pseudoknots, while also incorporating the ability to add stems, thus including pseudoknots, as based on the method by Cai [Cai, 2003].

Cao and Chen, who had introduced the first version of their Vfold algorithm in 2005 [Cao, 2005], extended it in 2009 to the prediction of pseudoknots [Cao, 2009]. Vfold is based on MFE with a time complexity of $O(n^6)$ that predicts H-type pseudoknots, specifically those connecting loops. In the same year, Chen introduced an algorithm with a time complexity of $O(n^5)$ for predicting H-type and HHH-type pseudoknots [Chen, 2009].

In 2009, Poolsap proposed a linear programming approach that minimizes free energy to predict H-type pseudoknots, even containing stems inside their loops [Poolsap, 2009]. However, the algorithm does not generate suboptimal structures.

In 2010, Bellaousov developed a tool called ProbKnot [Bellaousov, 2010], which is included in the RNAstructure package. This tool uses an algorithm that maximizes expected accuracy and has a time complexity of $O(n^2)$. Compared to existing approaches at the time, the lower time complexity allows to treat much longer sequences. The algorithm works by calculating pairing probabilities using McCaskill's algorithm, then assembling pseudoknot-free structures based on these probabilities. ProbKnot does not generate suboptimal structures.

Tfold [Engelen, 2010], developed in 2010, is a tool able to search for stems and pseudoknots recursively in long RNAs by optimizing criteria of stability, conservation and covariation in sequences that are selected from a large set of homologous sequences. Tfold is based on a divide-and-conquer algorithm with a time complexity of $O(n^2)$. However, it is needed to know homologous sequences in order to use it, which is rare in the case of long regulatory RNAs.

Another tool introduced in 2010, CyloFold [Bindewald, 2010], also simulates the folding process by selecting stable stems to predict secondary structures including pseudoknots minimizing the free energy.

In 2011, Sato developed a tool called IPknot [Sato, 2011], which uses an integer programming approach to maximize the expected accuracy given base-pairing probabilities. IPknot is able to predict pseudoknots, and can also predict consensus structures with pseudoknots for RNA alignments. The algorithm has a time complexity of $O(n^3)$. However, it does not generate suboptimal structures. LinearPartition [Zhang, 2020] was introduced in 2020 to compute the partition function in $O(n)$. This led to an improvement of IPknot in 2022 [Sato, 2022], using base-pairing probabilities from LinearPartition [Zhang, 2020] to predict pseudoknotted RNA secondary structure in $O(n)$ instead of $O(n^3)$, effectively allowing to treat much longer RNA sequences.

In 2011, Reidys developed an algorithm, called gfold [Reidys, 2011], based on a grammar and the topological classification of pseudoknots to determine the minimum free energy structure of an RNA.

In the same year, a tool from the ViennaRNA package, called RNAPKplex [Lorenz, 2011], is created to predict RNA secondary structures including pseudoknots using a heuristic approach. The algorithm has a time complexity of $O(n^4)$ and computes intervals of unpaired and accessible structures, then determines the regions that can form stable base pairs based on the free energy of interactions.

ShapeKnots [Hajdin, 2013] is another tool that uses a dynamic programming algorithm to minimize the MFE and predict secondary structures including pseudoknots. ShapeKnots is able to capture information from additional SHAPE data, enriching the model. On the downside, this means that ShapeKnots requires SHAPE data to reach its full potential.

In 2015, Janssen improved the pknotsRG model with a tool called pKiss [Theis, 2010; Janssen, 2015], able to predict HHH-type pseudoknots by minimizing free energy using dynamic programming. pKiss therefore predicts a broader range of pseudoknots than pknotsRG, which is limited to predicting certain H-type pseudoknots. Both pknotsRG and pKiss can generate multiple suboptimal structures.

MC-Fold was improved in 2016, resulting in a faster tool, MC-Flashfold [Dallaire, 2016].

In 2018, Knotty [Jabbari, 2018] is introduced as an improvement to the algorithm proposed in 2009 by Chen [Chen, 2009]. Knotty allows the identification of more pseudoknot types and provides a more realistic evaluation of the energy of the structures.

BiokoP [Legendre, 2018] is another method introduced in the same year and capable of predicting pseudoknots. It is a bi-objective algorithm, optimizing both the MFE and the MEA, based on integer programming.

In 2019, the first deep learning methods were introduced and quickly gained popularity. Wang introduced a method called DMfold [Wang, 2019a], which is based on deep learning and the maximization of the number of base pairs. The method first predicts structures without pseudoknots, then assembles them to predict pseudoknots in a second step.

In the same year, SPOT-RNA [Singh, 2019] was created as an ensemble of deep learning models based on a combination of CNN and RNN layers to predict a secondary structure including pseudoknots.

Developed in 2020, BiORSEO [Becquey, 2020] uses a combination of MEA optimization and module insertion to find pseudoknots. However, it comes with a high computation time and memory usage.

E2Efold [Chen, 2020] is a deep learning tool introduced in 2020 that is based on a Transformer encoder followed by a CNN decoder to produce a base-pairing probability matrix. That matrix is finally processed by a dynamic programming algorithm to enforce constraints in order to obtain a valid secondary structure including pseudoknots.

Developed in the same year, ATTfold [Wang, 2020] is another tool that also uses a Transformer encoder and a CNN decoder followed by a dynamic programming algorithm to transform the predicted base-pairing probability matrix into a valid pseudoknotted structure.

SPOT-RNA2 [Singh, 2021] is developed in 2021 and is an improvement of SPOT-RNA, based on an ensemble of dilated CNN models to predict RNA secondary structures including pseudoknots.

VLDB GRU [Lu, 2021] is a deep learning method that was introduced in 2021 to predict RNA secondary structure including pseudoknots through a RNN architecture that uses bidirectional GRU layers to produce a base-pairing probability matrix, once again processed by a dynamic programming algorithm to obtain a valid pseudoknotted secondary structure.

In 2022, CNNFold [Saman Booy, 2022] is a deep learning that was developed to predict pseudoknotted RNA secondary structures using a 2D CNN architecture. As for most deep learning approaches, a base-pairing probability matrix is predicted and then processed through a dynamic programming algorithm to yield a valid structure.

NeuralFold [Akiyama, 2022] is another deep learning tool developed in 2022, this time based on an MLP architecture, to predict a matrix base-pairing probabilities. The predicted probabilities are used as in the MEA approach, in place of the usual probabilities computed from the partition function.

UFold [Fu, 2022] is a CNN-based deep learning method released in 2022 that uses a U-Net model to predict a contact map of pairing probabilities, which is then converted to a secondary structure, including pseudoknots, through a post-processing algorithm.

REDfold [Chen, 2023] is based on a 2D CNN architecture resembling a U-net. Once again, a base-pairing probability matrix is predicted and processed by a dynamic programming algorithm to ensure constraints.

Knotify+ [Makris, 2023] is a tool based on context-free grammar to predict H-type pseudoknots in RNA secondary structure. The algorithm uses an objective function which combines maximizing the number of base pairs and minimizing the free energy.

In 2024, RNAformer [Franke, 2024] was released with the goal of predicting the secondary structure of RNA including pseudoknots with a 2D CNN architecture. However, RNAformer is limited to sequences shorter than 200 nucleotides.

DEBFold [Yang, 2024] is another RNA secondary structure prediction deep learning tool able to predict pseudoknots that was introduced in 2024. DEBFold is based on an architecture involving 1D convolutional layers and self-attention mechanisms. Yet, DEBFold is intended for RNAs shorter than 512 nt.

Also released in 2024, Kirigami [Harary, 2024] is another deep learning tool based on a 2D CNN architecture followed by an optimizer that aims to predict the secondary structures of RNA including pseudoknots, but it is intended for RNAs shorter than 500 nt.

KnotFold [Gong, 2024] is a deep learning model based on a Transformer architecture that was introduced in 2024 as well. KnotFold uses a minimum-cost flow algorithm on a potential function that is learned through an attention-based neural network and is able to predict pseudoknotted secondary structures.

Some of these approaches have attempted to focus on handling long RNAs past 1,000 nt. Considering tools for secondary structure prediction including pseudoknots, IPknot is very fast due to its linear complexity and can be easily used for long RNAs. ProbKnot, Tfold and KnotFold are relatively fast and can be applied to long RNAs as well, although Tfold requires homologous sequences.

3.3 Datasets

Historically, the first RNA structure databases used comparative sequence analysis over known sequence alignments to obtain curated secondary structures. Earlier databases often focused on a single class of RNAs, studying the structure of ribonuclease P, SRP RNAs, tRNAs, tmRNAs or rRNAs. Over time, several of such databases started to coexist, and there was a need to centralize data in a single database. To that end, meta-databases were later created to aggregate data from the different existing structure databases to make it more easily accessible.

The Protein Data Bank (PDB) [Hamilton, 1971; Berman, 2000] is a database released in 1971 and originally containing protein structures. However, the PDB has continuously grown since it was first published, and it now contains secondary structures of several biological molecules, including RNAs. Most structures are determined by X-ray diffraction, and some others by nuclear magnetic resonance (NMR). Since 2013, an increasing number of structures are determined by cryo-electron microscopy. The number of structures in the PDB has grown exponentially since its release. As of 2025, the PDB contains the structures of more than 200,000 molecules, including 1,915 RNAs.

The Ribonuclease P (RNase P) database [Brown, 1994] is a compilation of RNase P sequences, sequence alignments and secondary structures introduced in 1994. In addition to information on RNase P proteins, the database also contains the secondary structures of several RNAse P RNAs in various species. It is one of the first RNA structure databases, since few examples of RNAs were yet contained into the PDB at that time. However, it only contains information about the RNase P family.

Presented in 1996, the Signal Recognition Particle Database (SRPDB) [Larsen, 1996] provides alignments of SRP RNA sequences, along with secondary structures that were estimated by comparative sequence analysis. The SRPDB was enriched over the years to include more and more RNA alignments and secondary structures [Samuelsson, 1999; Gorodkin, 2001; Rosenblad, 2003], but it is solely focused on the SRP RNA family.

The tRNA database (tRNAdb) is another RNA structure database that focuses on a single RNA class. It originally started with works from Sprinzl in 1998 to compile and align tRNA sequences [Sprinzl, 1998; Sprinzl, 2005]. The database contains 5800 tRNA sequences from various organisms. The tRNAdb was improved in 2009 [Jühling, 2009] to include 12,000 total tRNA sequences and to provide secondary structures for the alignments from comparative sequence analysis.

Published in 1998 and later improved in 2003, the tmRNA database (tmRDB) [Zwieb, 1998; Zwieb, 2003] compiles data for tmRNA sequences. Like the other RNA structure databases released in the previous years, it focuses on a single RNA class to provides RNA sequences, alignments, and secondary structures through comparative sequence analysis. The rmRDB contains 274 tmRNA sequences, which combine properties of tRNA and mRNA.

In 2002, the Comparative RNA Web (CRW) [Cannone, 2002] was released. Akin to previous RNA structure databases, the CRW catalogues RNA sequences and alignments, and applies comparative sequence analysis to deduce secondary structures through patterns of conservation and covariation in the alignments, in diverse organisms. However, the CRW is among the first more diverse RNA structure databases. Not limiting itself to a single RNA class, the CRW instead considers data for rRNAs (5S, 16S, and 23S), tRNAs, and catalytic intron RNAs. New RNAs and new categories of RNAs are added regularly to the CRW for continuous development.

Rfam [Griffiths-Jones, 2003] is a comprehensive database introduced in 2003 that focuses on RNA families, providing information on non-coding RNAs. It includes curated sequence alignments, consensus secondary structures, and covariance models for the identification of RNA families. The first release of Rfam (1.0) in 2003 contains 25 families including over 50,000 non-coding RNAs. Rfam has been regularly refined over the years to include newly discovered families and to improve the quality of determined consensus secondary structures and RNA families [Nawrocki, 2015; Kalvari, 2021; Ontiveros-Palacios, 2025]. The 2025 release of Rfam (15.0) contains 4178 total RNA families.

The RNA secondary STRucture and statistical ANalysis Database (RNA Strand) [Andronescu, 2008], published in 2008, is a curated meta-database of 4,666 known RNA sequences and secondary structures extracted from other public databases, including the PDB, RNase P database, SRPDB, tRNAdb, tmRDB, CRW, and Rfam 8.1. RNA Strand is collaborative and allows researchers to add newly found secondary structures to the database.

ArchiveII [Sloma, 2016] is presented in 2016 as a database of RNA sequences and their secondary structures, combining several data sources including the RNase P database, SRPDB, tRNAdb, tmRDB, and Rfam 9.1. The sequence alignments are gathered and their secondary structures are determined through comparative sequence analysis. ArchiveII contains 2975 RNA sequences of lengths from 28 to 2968 nucleotides across 10 RNA families.

Similarly, RNAStralign [Tan, 2017] is an RNA alignment and secondary structure database published in 2017 and aggregated from existing databases including the RNase P database, SRPDB, tRNAdb, tmRDB, CRW, and Rfam 12.0. RNAStralign is often considered complementary to ArchiveII since together they cover the majority of existing RNA structure data.

Released in 2018, bpRNA-1m [Danaee, 2018] is another widely used meta-database of RNA sequences and secondary structures. It combines data from the PDB, RNase P database, SRPDB, tRNAdb, tmRDB, CRW, and Rfam 12.2. It comprises over 100,000 RNA sequences and secondary structures, which is significantly larger than other databases. Furthermore, it encompasses all the RNA structure databases mentioned previously. As such, bpRNA-1m is very widely used for RNA secondary structure analysis today. In addition to the sequences and secondary structures, bpRNA-1m also contains annotations of structural motifs such as stems, loops, and pseudoknots, facilitating structure analysis and visualization.

With the advent of data-driven prediction methods, based on a training dataset and evaluated over a test dataset, the two datasets are often separated using sequence similarity. A common approach is to split sequences at an 80% similarity threshold, with a clustering tool such as CDHIT-EST [Li, 2006]. This approach is called sequence-wise evaluation. It is the most common and is sufficient to correctly assess the performance of models in cases where it is not planned to use the tool for unknown families. However, tools are often bound to be applied to unknown families, in which case it is important to evaluate its generalization capabilities to unknown families to avoid overfitting. In this case, it has been shown that splitting data based on sequence similarity is not sufficient, and may hide overfitting [Szikszai, 2022]. Indeed, deep learning models that solely rely on sequence-wise evaluation such as E2Efold [Chen, 2020] and CNNFold [Saman Booy, 2022] were later revealed to be prone to overfitting. Instead, it is necessary to use a test dataset containing exclusively RNA families that are absent from the training dataset, which is called family-wise evaluation. To answer to this need, bpRNA-new [Sato, 2021] was created in 2021 by Sato et al. as a test dataset to bpRNA-1m. As such, bpRNA-new only contains RNA families that are not found in bpRNA-1m. Indeed, since the release of bpRNA-1m in 2018, about 1,500 new RNA families have been discovered up to 2021 and documented in the Rfam database. To enforce that the families in the two dataset are different, bpRNA-new collects data from the families found in Rfam 14.2 that were not already in Rfam 12.2, which is the version that was used to build bpRNA-1m. This ensures that the families in bpRNA-new cannot be found in bpRNA-1m. The authors chose to remove sequences longer than 500 nt from bpRNA-new. Nowadays, many deep learning models perform family-wise evaluation in their benchmark [Sato, 2021; Yang, 2024; Gong, 2024].

3.4 Conclusion

Despite these advancements, substantial challenges persist. Computational complexity is a significant obstacle, particularly in the context of long RNAs and/or pseudoknots. Indeed, dealing with long RNAs can be an issue given the exponential growth in the number of possible secondary structures. In addition, the scarcity of structure data for long RNAs is also problematic for the training of machine learning models. Furthermore, pseudoknot prediction is a challenging and computationally expensive problem. As a whole, existing approaches that are able to predict pseudoknots require a time complexity of at least $O(n^2)$, which makes it difficult to handle long RNAs. One exception to this is IPknot [Sato, 2011; Sato, 2022], which introduced a new version [Sato, 2022] with a time complexity of $O(n)$.

4 DivideFold: a partition method for RNA secondary structure prediction

The accurate prediction of RNA secondary structure is of great importance in understanding the functions of RNAs. However, existing approaches often face computational challenges or lack precision when dealing with long RNA sequences. To address this, we propose a divide-and-conquer partition method based on deep learning, called DivideFold, for predicting the secondary structures of long RNAs. Divide-and-conquer algorithms have been developed previously to predict RNA secondary structure, such as Tfold [Engelen, 2010] and RNA-par [Zhao, 2023], but not for long RNAs using only their sequence. The core concept driving our approach is to recursively separate long RNAs into smaller fragments and use an existing secondary structure prediction tool to estimate the secondary structure of each fragment. This approach has the potential to tackle both computational and accuracy challenges, as shorter fragments are typically faster to process and have simpler secondary structures that are easier to predict.

In this chapter, we present our approach and describe the implementation details of our divide model, as well as the training procedure of its deep neural network.

4.1 General approach

Our goal in the proposed approach is to partition a long sequence into shorter, structurally independent fragments to make use of existing structure prediction models designed for small RNAs and predict their structures. Then, the predicted structures for the different fragments are recombined together to their original locations in the sequence and constitute the global structure prediction for the long sequence. Our approach consists of any combination of a divide model and a structure prediction model. The workflow of Divide-

Fold is depicted in Fig 4.1. While we design the divide model ourselves, we choose the structure model among already existing secondary structure prediction models.

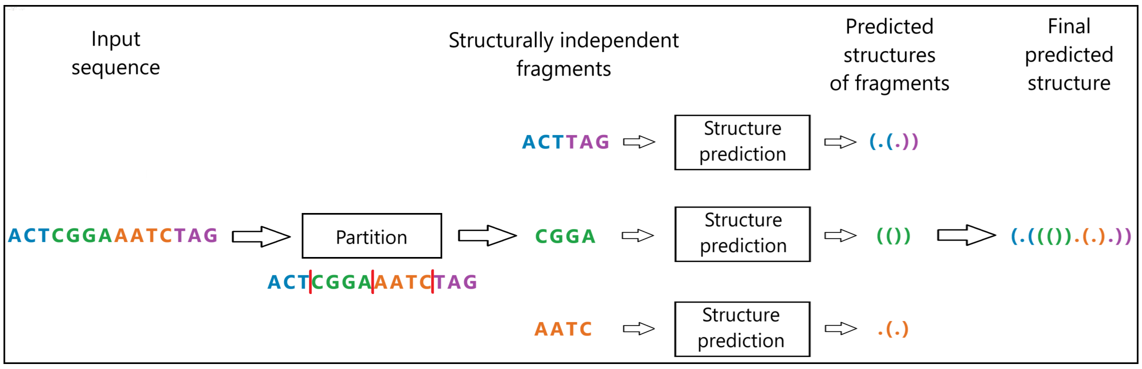

Figure 4.1: **DivideFold workflow.** The sequence is partitioned into several fragments and the different predicted secondary structures are then recombined to the original positions of the fragments in the sequence.

4.2 Partition strategy

A key challenge in partitioning the sequence lies in strategically selecting cut points as to preserve the structural integrity of the RNA as much as possible. Indeed, any base pair occurring between two different fragments will be broken and impossible to recover at the structure prediction step, since each fragment is processed separately. To avoid breaking base pairs during partition, cut points should ideally be placed between the different stems that are minimally nested. This would result in fragments where all base pairs only occur within themselves, meaning that no base pairs are broken during the process except pseudoknots, while being able to partition the sequence as deep as desired. However, for this, it is first necessary to remove pseudoknots from the structure. Indeed, it is only possible to keep locating the regions between the different stems once the structure is non-pseudoknotted, ensuring that there are no conflicts between different stems. It would not be possible to find a cut point that separates two stems forming a pseudoknot, making it impossible to partition the whole region covered by the pseudoknot. For the same reason, it is impossible to find cut points that break no base pairs at all while partitioning the sequence as deep as desired when the structure is pseudoknotted, meaning that breaking pseudoknots is eventually unavoidable. Thus, to break no base pairs except pseudoknots is the best that we can aim for.

Once the cut points have been chosen, we allow the immediate combination of the left-most and right-most parts into a single fragment, creating an artificial fragment that, while not directly continuous in the original sequence, maintains structural coherence. This step is crucial to make it possible to partition further a sequence where a long-range stem contains a large portion of the structure without breaking that stem. Indeed, in this scenario, the stem would inevitably be broken if only continuous fragments are used in the partition. Instead, by combining the external parts in a discontinuous fragment, the sequence is guaranteed to be partitioned into at least two fragments, ensuring that it will be shortened, while having the opportunity to preserve structural information.

We decide to repeat choosing cut points and merging outer parts recursively to allow to focus on the minimally nested regions at each iteration and progressively partition the sequence until all fragments are shorter than a chosen length. This is important because some regions may be more deeply nested than others. The precision of cut point selection is an important issue, as any misstep in this process can significantly compromise the accuracy of the final predicted structure over the iterations. Fig 4.2 provides a visual representation of one partition iteration.

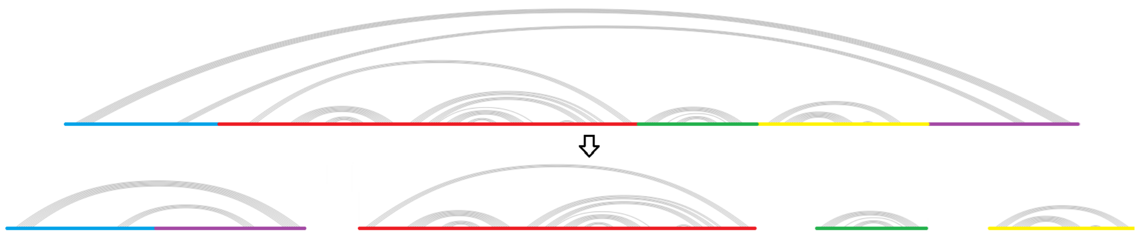

Figure 4.2: **One partition iteration.** The partition process occurs recursively until all fragments are shorter than a chosen length. At each iteration, the left-most and right-most parts are combined to form a single fragment. Note that the red fragment may be further partitioned similarly at the next iteration.

4.3 Divide model

In this section, we present our algorithm for predicting cut points in a given sequence, using only the sequence information. Features are first extracted from the sequence. The deep learning neural network, made of an encoder and a regression head, then generates cutting probabilities for each nucleotide in the sequence. Cut points are finally selected from the predicted probabilities through a peak detection algorithm. This process is repeated recursively until the fragments are shorter than a chosen length. We show the architecture of the divide model in Fig 4.3.

4.3.1 Feature extraction

We have explored three different strategies for the input features: one-hot embedding, DNABERT [Ji, 2021] embedding and module insertion. One-hot embedding is the most standard and direct way to encode the sequence of nucleotides, without additional information.

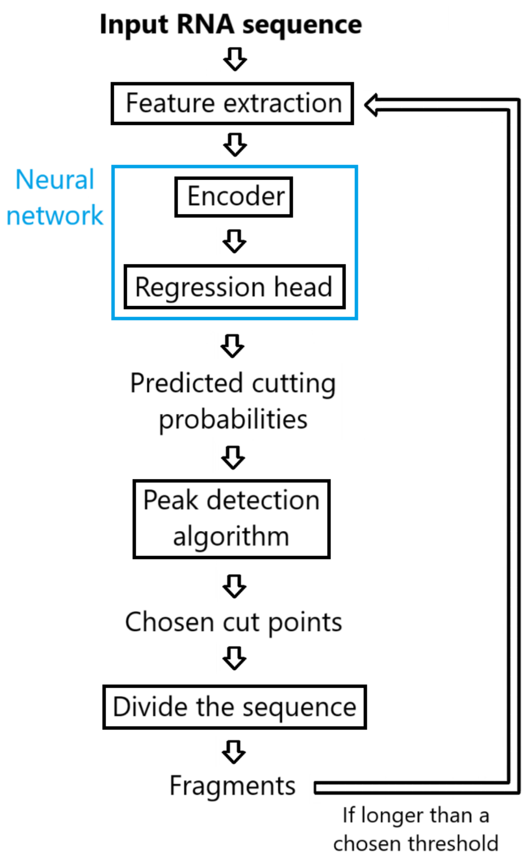

Figure 4.3: **Architecture of the divide model.** A concatenation of one-hot embeddings and module insertion is used for feature extraction, followed by an encoder and a pointwise fully connected regression head with sigmoid activation. A peak detection algorithm is used to extract cut points from the predicted cutting probabilities. The divide model is repeated recursively until all fragments are shorter than a chosen length threshold.

Another way to represent RNA sequence information is using pre-trained encoders, often based on the BERT [Devlin, 2019] Transformer architecture, such as DNABERT [Ji, 2021]. Designed and pre-trained on a large DNA database by Ji et al., DNABERT provides a learned representation to DNA sequences that captures important relationships in the sequence. Since RNA sequences are derived from DNA and share similar nucleotide patterns, DNABERT has also been applied to RNA sequences as a tool to represent sequence information for diverse downstream tasks such as RNA classification [Creux, 2025] or RNA 3D torsion angles prediction [Bernard, 2025]. We also apply DNABERT to RNA sequences to employ it as a feature extraction strategy.

Finally, module insertion involves integrating known modules or structural patterns into the RNA sequence as additional features. We describe modules in Section 2.1.5 and show examples of known modules in Fig 2.5. For module insertion, we use the modules from the CaRNAval [Reinharz, 2018] database. We propose an original approach where for each module, we check for matches in the sequence through regular expressions and count the number of possible configurations for the module to be inserted at each point in the sequence. This is illustrated in Fig 4.4.

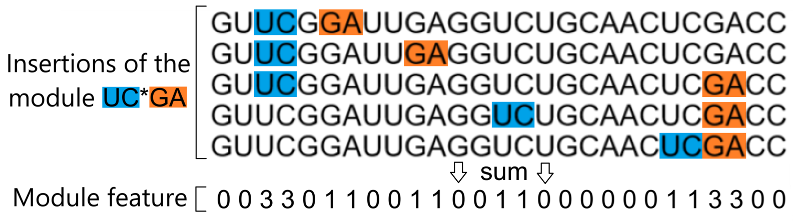
Insertions of the module UC*GA
GUUCGGAUUGAGGUCUGCAACUCGACC
GUUCGGAUUGAGGUCUGCAACUCGACC
GUUCGGAUUGAGGUCUGCAACUCGACC
GUUCGGAUUGAGGUCUGCAACUCGACC
GUUCGGAUUGAGGUCUGCAACUCGACC
sum
Module feature
0 0 3 3 0 1 1 0 0 1 1 0 0 1 1 0 0 0 0 0 0 1 1 3 3 0 0

Figure 4.4: **Illustration of the module insertion.** The insertion of the UC*GA module in a sequence is shown here. The possible configurations of the module are found inside the sequence using regular expressions. Then, at each nucleotide in the sequence, the number of configurations that occur at that position are summed, leading to a feature vector. This process is repeated for each module to be inserted, each time yielding a different feature for the sequence.

While using one-hot embedding alone is enough to access the sequence information, we find that the DNABERT embeddings and module insertion hold valuable additional information about relationships between nucleotides in the sequence. Providing the divide model's deep neural network with either one of the two, in addition to the one-hot embeddings, allows to significantly increase its prediction accuracy compared to using only one-hot embeddings. However, using both DNABERT embedding and module insertion yields no added benefit, indicating that the two are likely highly correlated and that it is only needed to use one of the two. Because DNABERT embeddings are very computationally expensive when applied to long RNAs compared to module insertion, we choose to use a concatenation of one-hot embeddings and module insertion as our input features. Furthermore, it should be noted that some of the modules require more time to be inserted than others and bring very little extra information because they are highly correlated to other lighter modules. For this reason, we choose to remove the heavier modules and observe a further drastic reduction in computation time without degrading performance.

4.3.2 Deep neural network architecture

We compute a cutting probability for each nucleotide in the sequence, using a deep neural network. First the feature extraction occurs, consisting of a concatenation of one-hot embeddings and module insertion as described previously. Then, the sequence is passed to the encoder, whose role is to translate it to relevant features in a vector space at each point in the sequence. Lastly, a fully connected regression head is applied pointwise, meaning that it is applied at each position independently, with a sigmoid activation to compute cutting probabilities at every nucleotide position.

We experimented with three different architectures for the encoder: an MLP encoder, a BiLSTM encoder and a 1D CNN encoder. All three encoders, as well as the regression head, run in $O(n)$ time complexity. The architecture of the three encoder networks are shown in Fig 4.5. The MLP encoder uses a single fully connected layer, applied pointwise with dimension 16 and ReLu activation. For the BiLSTM encoder, we use two bidirectional LSTM layers with dimension 32 and hyperbolic tangent (tanh) activation. Finally, the 1D CNN encoder is made of 10 dilated 1D convolutional layers with kernel size 3, dilation rates decreasing in powers of 2, and ReLu activations. The 1D CNN encoder uses a fixed kernel size with decreasing dilation rates, rather than variable kernel sizes, to capture long-range relationships in the sequence while avoiding the need to evaluate all possible pairs.

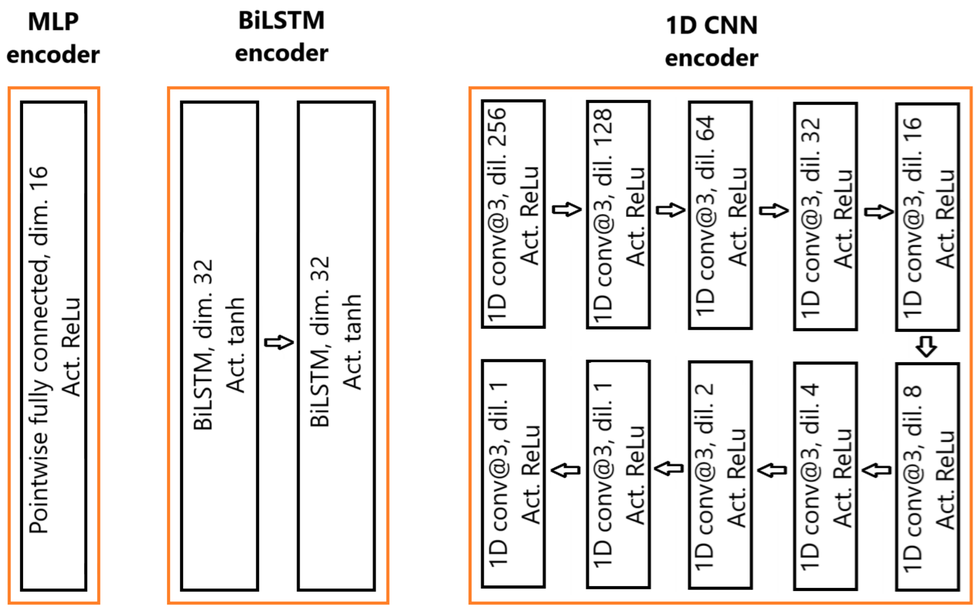
MLP encoder
Pointwise fully connected, dim. 16 Act. ReLu
BiLSTM encoder
BiLSTM, dim. 32 Act. tanh
BiLSTM, dim. 32 Act. tanh
1D CNN encoder
1D conv@3, dil. 256 Act. ReLu
1D conv@3, dil. 128 Act. ReLu
1D conv@3, dil. 64 Act. ReLu
1D conv@3, dil. 32 Act. ReLu
1D conv@3, dil. 16 Act. ReLu
1D conv@3, dil. 8 Act. ReLu
1D conv@3, dil. 4 Act. ReLu
1D conv@3, dil. 2 Act. ReLu
1D conv@3, dil. 1 Act. ReLu
1D conv@3, dil. 1 Act. ReLu

Figure 4.5: **Architecture of the MLP, BiLSTM and 1D CNN encoder networks.** A single pointwise fully connected layer with dimension 16 and ReLu activation is used for the MLP encoder. The BiLSTM encoder consists of two bidirectional LSTM layers with dimension 32 and tanh activation. The 1D CNN encoder uses 10 dilated 1D convolutional layers with kernel size 3, dilation rates decreasing in powers of 2, and ReLu activations.

4.3.3 Removing pseudoknots

As mentioned in Section 4.2, it is necessary to remove pseudoknots in order to find ideal cut points in the sequence. Indeed, two stems forming a pseudoknot cannot be separated with a cut point, as the two stems are entangled. Thus, to be able to find the regions between any two stems, the pseudoknots must be removed from the structure, ensuring that no conflicts occur between the stems. This is important, as we use this procedure to find ideal cut points to generate training labels.

Several methods can be used to remove pseudoknots, which can typically be regrouped in dynamic programming methods, conflict elimination methods, and incremental methods [Smit, 2008]. The first use a dynamic programming algorithm to find the non-pseudoknotted structure maximizing a criterion of choice such as the number of base pairs in the structure. The conflict elimination starts from the pseudoknotted structure and remove conflicting base pairs one by one, until no conflicts remain. A criterion must be chosen to determine which base pair to remove at each step. For example, it can be chosen to remove the base pair that causes the most conflicts, or to remove the conflicting base pair that encompasses the least amount of base pairs in order to keep the longest stems, which may be more structurally important. Similarly, incremental methods start from an empty structure and add base pairs one by one, as long as no conflict occurs. As before, a criterion must be picked to choose which base pair to add at each step. A possible approach is to add base pairs from the $5'$ end to the $3'$ end, or from the $3'$ end to the $5'$ end. Another possibility involves prioritizing the longest stems first, since the stems that comprise more base pairs may contribute more to the structure. Lastly, it is possible to add the short-range interactions first, assuming that they may be more central to the structure than long-range interactions. Importantly, there is no absolute "truth" in pseudoknot removal, and no single approach is inherently better than the others, as the criteria for which pseudoknots should be removed are subjective. Instead, all methods operate differently and some may be more suitable than others depending on the context and the user's goals. To remove pseudoknots, we use implementations from the PyCogent [Knight, 2007] project. We discuss which pseudoknot removal is the most beneficial to the training of the divide model's deep neural network in Section 5.2.3.

4.3.4 Generating training labels

We generate training labels for cut points at ideal positions following the strategy presented in Section 4.2 and represented in Fig 4.2, from the RNA's sequence and its secondary structure. Specifically, we remove pseudoknots from the structure and we position the cut points between the different minimally nested stems. These cut points are guaranteed to preserve all non-pseudoknotted base pairs in the structure, making them relevant training labels for our deep neural network. Because the partition process is recursive, at each iteration the current sequence to be partitioned is used as input and new training labels are generated. This means that for each sequence in the training dataset, when partitioning to generate training labels, each iteration contributes to a new observation for the training of the divide model's deep neural network, thus serving as a valuable data augmentation tool.

4.3.5 Loss function

We adopt a distinctive loss function for the divide model's deep neural network, denoted as $\mathcal{L}$. We design an original ground truth function y (Eq (4.1)) and we set $\mathcal{L}$ as the mean squared error between y and the predicted probabilities $\hat{y}$ (Eq (4.2)):

$$\forall\, i \in [\![1\,;\,N]\!], y_i(C) = \exp^{-\lambda * \min_{c \in C} |i-c|} \tag{4.1}$$

$$\mathcal{L}(\hat{y}, C) = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i(C))^2 \tag{4.2}$$

where $\hat{y}$ is the predicted cutting probabilities at each point in the sequence, C is the set of target cut points, N is the sequence length and $\lambda > 0$ is a hyperparameter relative to how steeply the target score function decreases when getting farther from the labeled cut points.

The training process focuses on optimizing the model parameters to minimize the loss function, which quantifies the differences between the positions of predicted and target cut points. Rather than assigning a value of 1 for an exact prediction of the position of a target cut point and 0 otherwise, y uses an inverse exponential distance to the nearest target cut point, leading to peaks around target cut points. This provides a much more extensive support, with non-zero values everywhere, enabling a more reliable flow of information back to the model. This function emphasizes precise cut point positioning, which is crucial since even minor deviations can accumulate over the iterations, significantly influencing the final predicted RNA structure. Overall, this loss function facilitates the training of the model, offering less sparse information and demonstrating superior performance compared to standard loss functions.

For easier interpretation of the λ hyperparameter, we plot y for different values of λ in Fig 4.6 for an example sequence. Given a set of training labels C, y forms peaks around the target cut points. The width of the peaks is determined by the value of λ. We show here that if λ is too low, peaks may not be well separated as shown in blue. On the other hand, it can be seen in green that if λ is too high, y will rapidly collapse to 0, making the training process harder. We observe in the data that stems can span as little as 12 nt in range. Thus, successive target cut points are at least 12 nt apart. Therefore, we choose a value of 0.5 for λ as we find that it is the lowest value allowing to properly separate any two peaks at least 12 nt apart. At the same time, it allows y to decrease more slowly compared to higher values of λ, leading to an increased support for y and, consequently, an easier training for the divide model's deep neural network.

4.3.6 Training process and fine-tuning

Considering RNA secondary structure, available datasets are relatively small compared to the large amounts of data typically needed to train deep learning models. To mitigate this issue, transfer learning has been used successfully for RNA secondary structure prediction [Singh, 2019; Chaturvedi, 2025]. This challenge is even greater for long RNAs, which are especially rare in datasets. To help alleviate this obstacle, we perform

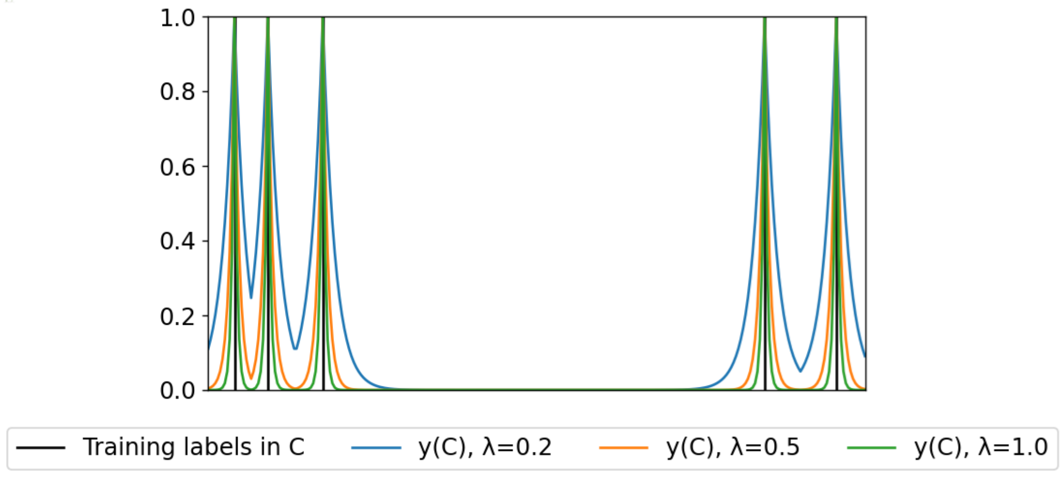

Figure 4.6: **Curve of y for different values of λ.** An example sequence is considered here. The generated training labels can be seen in the vertical black bars, and y forms peaks around them. The width of the peaks depends on the value of λ that was used to compute y.

pre-training and fine-tuning, described in Section 2.2.1. We choose to first pre-train the divide-model on shorter sequences in order to better learn the fundamental structural patterns that can be found in RNA, before fine-tuning it to long RNAs which may involve more intricate relationships. We discuss the specific length thresholds chosen for pre-training and fine-tuning in Section 5.2.4. We use the Adam optimizer [Kingma, 2017] for the training of the divide model's deep neural network. Adam is the most commonly used optimizer in deep learning due to its combination of adaptive learning rates and momentum, which makes it effective across a wide range of tasks.

4.3.7 Peak detection

Yet, predicting cutting probabilities is not enough, as we need to decide at which positions the sequence should actually be cut. For this reason, following the prediction of cutting probabilities at each point in the sequence, we employ a peak detection algorithm to find peaks in the probabilities and select the cut points that will be chosen to partition the sequence. Indeed, we cannot simply select positions where the predicted probabilities are above a threshold, as many very close positions would be selected in a region where the probabilities are high. Instead, a peak detection algorithm allows to select a single position in such a region. Additionally, we specify that two successive peaks should be at least 12 nt apart, as we observe that this is roughly the smallest possible range of a stem as described in Section 4.3.5. This is because cut points should be placed between the different stems, which means that the region between two successive cut points should be long enough to accommodate a stem. Depending on the probabilities predicted by the model, the peak detection algorithm may retain an arbitrary number of cut points. We ensure that at least two cut points are selected, so that the sequence is guaranteed to be partitioned into at least two fragments. We employ the *signal.find_peaks* implementation from the Scipy [Virtanen, 2020] library.

Fig 4.7 illustrates the predicted cutting probabilities and the cut points chosen by the peak detection algorithm for the *Bradyrhizobium* 16S ribosomal RNA gene. In this example, the sequence is partitioned into five parts, with the left-most and right-most parts being combined into a single fragment, resulting in four fragments. We also show the structure of the RNA for reference. It can be seen that the model correctly cuts the sequence between the different parts, and no base pairs are broken. The base pairs connecting the left-most and right-most parts are also conserved since these two parts are combined.

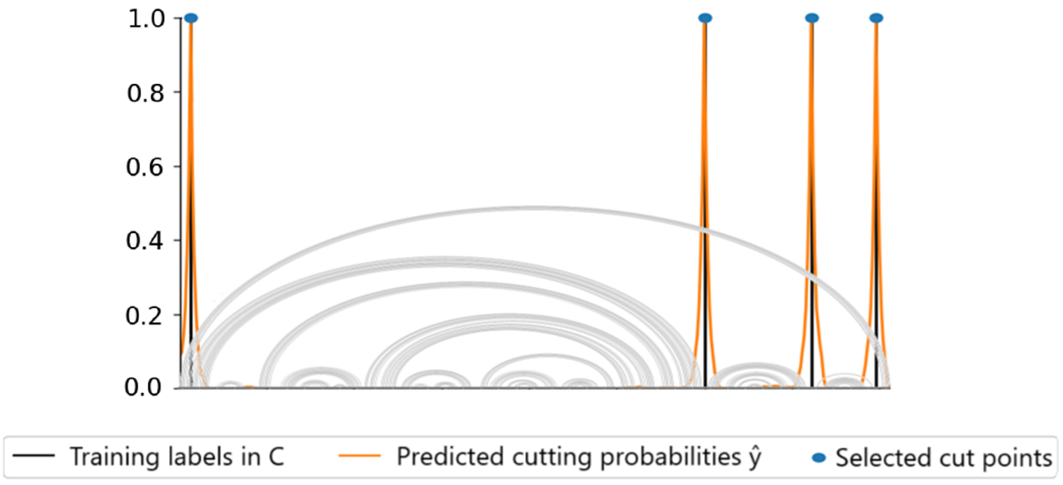

Figure 4.7: **Cut points prediction.** The *Bradyrhizobium* 16S ribosomal RNA is given here as an example. The divide model's deep neural network predicts cutting probabilities, and cut points are then selected using a peak detection algorithm. The selected cut points can be seen in the blue dots, and ideal positions in the black bars. The structure of the RNA is displayed for reference. No base pairs are broken here.

4.4 Time and memory complexity

All encoders considered for the deep neural network (MLP, BiLSTM, and 1D CNN) have a time complexity of $O(n)$, where n denotes the sequence length. All other components of the divide model, including one-hot encoding, module insertion, the regression head, and the peak detection algorithm, also have a time complexity of $O(n)$. Additionally, the space complexity of each of these components is also of $O(n)$. Therefore, a single iteration of the divide model has both time and space complexity of $O(n)$. During the partition step, the divide model is applied recursively and may run multiple times. Consequently, the partition step has a time complexity in $O(d * n)$, where d is the number of times that the divide model is called. However, since memory can be released between iterations, the space complexity of the partition step remains in $O(n)$. In the worst case, the divide model could be applied up to $n - 1$ times, by isolating one nucleotide from the rest of the sequence at each iteration. Since $d \leq n - 1$, it follows that d is in $O(n)$ in the worst case. Therefore, the partition step has a worst case time complexity of $O(n^2)$, while its space complexity remains $O(n)$.

In the structure prediction step, each fragment is processed independently by the secondary structure prediction model. Fragments are guaranteed to be shorter than the maximum fragment length M, so in the worst case, a sequence of length n may be divided into n/M fragments of length M. The total computation time is thus $n/M * T(M)$, where $T(M)$ is the time cost of the structure prediction tool for an input of length M. Since M and $T(M)$ are constants, this results in a time complexity of $O(n)$ for the structure prediction step. Because memory can be refreshed between fragment predictions, the maximum memory usage is bounded by $S(M)$, where $S(M)$ is the space requirement of the prediction tool for an input of length M. As $S(M)$ is constant, the space complexity of the structure prediction step is of $O(1)$.

However, in practice, the number of times d that the divide model is executed is much lower. Specifically, the divide model runs only once in 83% of cases, less than three times in 94% of cases, and never more than 12 times. Thus, in practice, the divide model runs less times than a constant, meaning that the partition step operates in linear time. Furthermore, as shown in section 5.3.3, the partition step is significantly faster than the structure prediction step in practice. Its memory cost is also negligible compared to that of the structure prediction step.

Because of this, DivideFold is dominated by the structure prediction step, running in linear time and using constant memory in practice, regardless of the structure prediction model applied to the fragments. This allows DivideFold to process long RNAs quickly and to avoid memory overflows, even when using a structure prediction tool with high time and space complexity.

4.5 Implementation and availability

We write DivideFold using the coding language Python. DivideFold, along with all the datasets used for this study, is publicly accessible on the EvryRNA platform. It is possible to download the GitHub repository, which hosts all code and data, and install the corresponding Python library for local usage. When used locally, DivideFold is intended for Python version 3.11 or higher, and is compatible with either TensorFlow or PyTorch as the deep learning backend depending on the user's preference. DivideFold can be used alone to provide cut points for an RNA sequence, or combined with a secondary structure prediction tool by passing the structure prediction function as an argument to DivideFold. This allows users to pair DivideFold with any tool installed locally, facilitating its integration into ongoing research. Hyperparameter values, including the maximum fragment length, can be adjusted. For easier usage, DivideFold can also be used as a web server. A graphical interface is provided on EvryRNA, allowing users to easily pass RNA sequences and obtain predictions.

5 Secondary structure prediction excluding pseudoknots of long RNAs

In this chapter, we detail the datasets and methods used in our experiments and we report the results of DivideFold in comparison to the other benchmarked tools for secondary structure prediction excluding pseudoknots. We do not consider the prediction of pseudoknots in this chapter, as we solely focus on the challenge of handling long RNAs. We then discuss the results and opportunities for future improvements.

5.1 Dataset preparation

As discussed in Section 3.3, ArchiveII [Sloma, 2016], RNAStralign [Tan, 2017], and bpRNA-1m [Danaee, 2018] are among the most commonly used datasets for RNA secondary structure, each combining data from several previously established sources. Since the contents of ArchiveII and RNAStralign are already included in bpRNA-1m, we use the bpRNA-1m dataset for our experiments. The structures in the bpRNA-1m database are predominantly determined through comparative sequence analysis from the Comparative RNA Web (CRW) [Cannone, 2002] and Rfam 12.2 [Griffiths-Jones, 2003; Nawrocki, 2015] databases. The bpRNA-1m dataset features 102,318 RNA sequences alongside their corresponding secondary structures, among which 69,969 are unique. We show the distribution of sequence length in bpRNA-1m in Fig 5.1. Although most sequences are shorter than 100 nt, many longer sequences can be found in bpRNA-1m up until 1,600 nt. On the downside, bpRNA-1m contains very few RNAs longer than 1,600 nt, which could make the training of our model difficult for long RNAs past this threshold, but this is the case for every RNA secondary structure dataset.

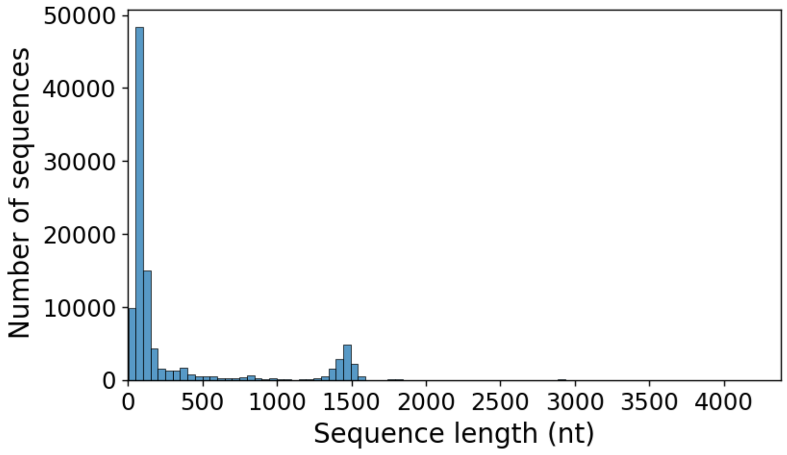

Figure 5.1: **Distribution of sequence length in bpRNA-1m.** Most sequences are shorter than 100 nt, but many longer sequences can be found up to 1,600 nt. However, sequences longer than 1,600 nt are very rare in bpRNA-1m, which is why they are barely visible in this figure.

5.1.1 Sequence-wise data split

When using a deep learning model, data must be split into separate training and test datasets in order to ensure that the model is not evaluated on the same data that it was trained on and to avoid overfitting. Additionally, it is needed to make sure that the samples in the training and test datasets are not too similar, as this would also encourage overfitting. Since the family of the different sequences in bpRNA-1m is not known, we must proceed differently to ensure that the samples are different enough between the two datasets. A common practice for deep learning tools in secondary structure prediction is to cluster the dataset using the CD-HIT-EST [Li, 2006] software at a 80% sequence similarity threshold [Singh, 2019; Singh, 2021; Sato, 2021; Fu, 2022; Saman Booy, 2022; Gong, 2024]. We first remove duplicate sequences from bpRNA-1m. Then, following guidelines from previous research, we also cluster bpRNA-1m using CD-HIT-EST at a 80% sequence similarity threshold. Therefore, any sequences that belong to a different cluster share a similarity no higher than 80%.

We then split bpRNA-1m into a Train, Validation and Test datasets according to the clusters, in a way that each cluster is exclusively assigned to one dataset. This ensures that any two sequences picked from different datasets, thus different clusters, have a similarity no higher than 80%. Consequently, we enforce that the Train, Validation and Test datasets are not too similar. Moreover, we enforce that the TR0 dataset introduced by Sato et al. [Sato, 2021] is included in the Train dataset. This allows to avoid re-training existing tools that were trained on TR0. We use the Train dataset for the training of our model. The Validation dataset is used for the choice of the maximum fragment length. Lastly, we use the Test dataset for evaluating the performances of our method and comparing it to the state-of-the-art in our experiments. Our final Train, Validation and Test datasets contain

57,251, 2,544 and 10,174 RNAs respectively. The number of sequences in each dataset is summarized in Table 5.1.

Table 5.1: **Number of sequences in the datasets originating from bpRNA-1m for the sequence-wise data split.**

Dataset	Number of sequences
bpRNA-1m	102,318
bpRNA-1m, deduplicated	69,969
Train	57,251
Validation	2,544
Test	10,174

The number of sequences in the bpRNA-1m [Danaee, 2018], Train, Validation, and Test datasets are shown. Once deduplicated, bpRNA-1m is split between the Train, Validation and Test datasets following a clustering performed with CD-HIT-EST [Li, 2006].

5.1.2 Family-wise data split

Sequence-wise evaluation is useful, as it is sufficient to properly assess the performance of a tool intended for known RNA families, which is often the case. However, it may fall short when the tool is meant to be used for newly discovered families. As a matter of fact, sequence-wise evaluation has been shown to be insufficient in that case, as it may hide overfitting [Szikszai, 2022]. Instead of splitting data using only sequence similarity, family-wise evaluation is necessary to ensure that the models are able to generalize to unknown families. Deep learning methods often perform family-wise evaluation in their benchmark, ensuring that the training and test datasets exclusively contain different RNA families [Sato, 2021; Yang, 2024; Gong, 2024]. Thus, we also need a separate test dataset to check for the generalization to unseen families. However, as the family of the different sequences in bpRNA-1m is not known, bpRNA-1m cannot be easily split according to RNA family. To answer to this need, the bpRNA-new dataset is released by Sato et al. in 2021 [Sato, 2021]. It contains RNA families introduced in Rfam 14.2 [Kalvari, 2021] which were not yet discovered in Rfam 12.2 [Nawrocki, 2015] when the bpRNA-1m [Danaee, 2018] dataset was introduced. Thus, bpRNA-new only contains different families from bpRNA-1m, and is widely used in the literature as a test dataset for family-wise evaluation [Sato, 2021; Yang, 2024; Gong, 2024]. However, sequences longer than 500 nt were removed from bpRNA-new, making it unsuitable for DivideFold. To solve this, we design a novel family-wise test dataset, which we name bpRNA-NF-15.0. For this, we followed guidelines and replicated the procedure that was used to create bpRNA-new. We gathered RNAs from newly identified families in the latest available version of Rfam at the time of writing, Rfam 15.0 [Ontiveros-Palacios, 2025], that were not yet introduced in Rfam 12.2 when bpRNA-1m was created, ensuring that bpRNA-NF-15.0 only contains different families from the ones in the Train dataset. Then, we deduplicated the dataset and checked for sequence similarity with CD-HIT-EST [Li, 2006] at a 80% similarity threshold, only keeping one representative sequence per cluster. In comparison to bpRNA-new, bpRNA-NF-15.0 is based on Rfam 15.0 rather than Rfam 14.2 and contains twice as many new unseen families. Furthermore, it contains sequences up to 951 nt long. Unfortunately, since the release of Rfam 12.2, no new families including sequences longer than 1,000 nt were discovered. This is a significant issue preventing us from properly studying the generalization capabilities of DivideFold to newly discovered RNA families, since DivideFold specializes in RNAs longer than 1,000 nt. Nonetheless, since bpRNA-NF-15.0 contains enough RNA sequences between 500 and 1,000 nt, we conduct this analysis on sequences within that range. We report the number of sequences and families in each dataset in Table 5.2.

Table 5.2: **Number of sequences and families in the datasets originating from Rfam for the family-wise data split.**

Dataset	**Number of sequences**	**Number of families**
Rfam 15.0	92,781	4178
Rfam 15.0 \ Rfam 12.2	41,143	1609
Rfam 15.0 \ Rfam 12.2, deduplicated	35,863	1609
bpRNA-NF-15.0	8,788	1605

The number of sequences and families in the Rfam 15.0 [Ontiveros-Palacios, 2025], Rfam 15.0 \ Rfam 12.2, and bpRNA-NF-15.0 datasets are shown. Sequences from Rfam 12.2 are first removed from Rfam 15.0. The resulting dataset is then deduplicated and clustered with CD-HIT-EST [Li, 2006], keeping only one representative sequence per cluster to form bpRNA-NF-15.0.

5.2 Hyperparameter tuning

We first conduct preliminary experiments to pick the hyperparameters of DivideFold. The hyperparameters that we evaluate are the secondary structure prediction tool applied to the fragments, the divide model's neural network architecture, the pseudoknot removal function used for generating training labels, the fine-tuning length threshold and the maximum fragment length. Instead of running a full grid search over all hyperparameter combinations, which would have taken over 100 days even with parallelization across 20 GPUs, we evaluate hyperparameters one at a time while keeping the others fixed, reducing computation time to under 3 days. All experiments for selecting DivideFold's hyperparameters are performed on the Validation dataset.

5.2.1 Structure prediction model

We first evaluate the different methods to decide which one we will integrate in our approach as the structure prediction model. Since we ignore pseudoknots in this chapter, we consider tools designed for secondary structure prediction excluding pseudoknots, which we described in Section 3.2.1. We include RNAfold [Hofacker, 1994; Lorenz, 2011] in this experiment, as it is a tool of reference for secondary structure prediction. We also evaluate LinearFold [Huang, 2019] since it is a faster and potentially stronger improvement to RNAfold. We consider MXfold2 [Sato, 2021] as well, as it is frequently cited among deep learning approaches. We also include KnotFold [Gong, 2024], a recent tool designed for RNA secondary structure prediction including pseudoknots, as it has shown very good results on non-pseudoknotted base pairs in addition to pseudoknotted base pairs [Gong, 2024]. We remove the pseudoknots from KnotFold's prediction in this chapter to evaluate it for secondary structure prediction excluding pseudoknots. All methods (RNAfold, LinearFold, MXfold2 and KnotFold) are used with their default parameters. In the case of MXfold2 and KnotFold, we use the weights pre-trained on the TR0 [Sato, 2021] dataset, which is included in the Train dataset, sparing us the need to re-train them. RNAfold and LinearFold are not trained models.

We evaluate RNAfold, LinearFold, MXfold2 and KnotFold for short sequences below 1,000 nt, since we will use the structure prediction model on short fragments. We measure the models' performance for secondary structure prediction using the F-score metric, as well as precision and recall. We describe these classification metrics in Section 2.2.3.

The prediction performances as well as the computation times of each of the four tools are shown in Fig 5.2. KnotFold has the highest performance out of the four methods. It reaches a mean F-score for secondary structure prediction that is substantially better than that of the other three methods for sequences in all considered ranges, in particularly for sequences longer than 200 nt. It requires around 10 seconds to compute a sequence, which is reasonable, albeit slower than RNAfold and LinearFold. In view of these results, we choose KnotFold as the structure prediction model in our approach in this chapter.

5.2.2 Architecture of the divide model's deep neural network

Another important choice for DivideFold is that of the encoder architecture in the divide model's deep neural network. We evaluate three different architectures for the encoder: an MLP, a BiLSTM and a 1D CNN. In each case, the encoder is followed by a pointwise fully connected regression head. In this experiment, we measure the secondary structure prediction performance. We only consider sequences longer than 1,000 nt, as we typically focus on RNAs longer than this threshold in this study. A maximum fragment length of 1,000 nt is used.

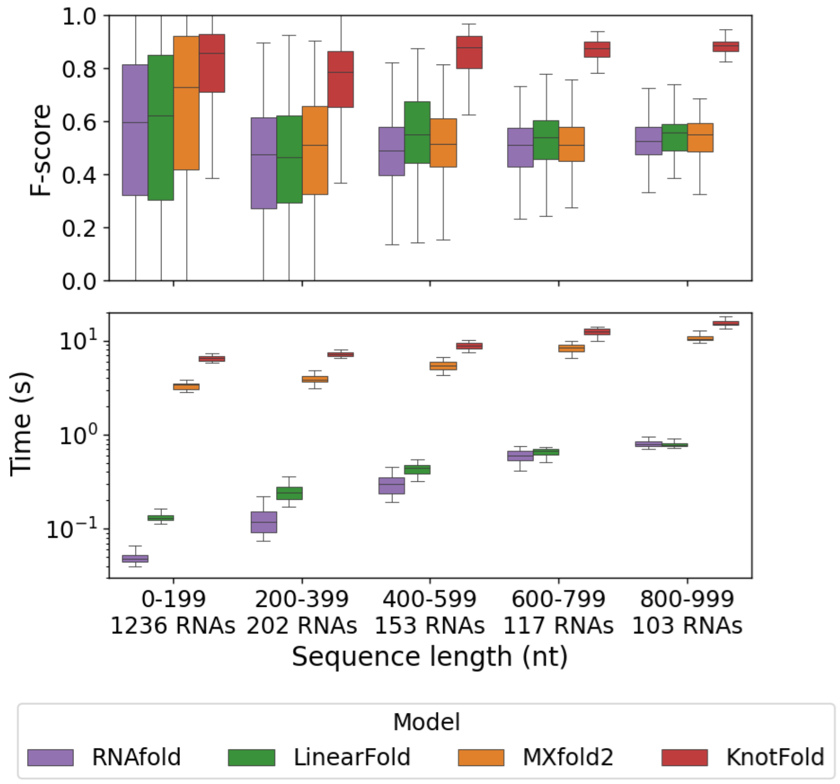

Figure 5.2: **Evaluation of the structure prediction models by sequence length on the Validation dataset.** The F-score for secondary structure prediction depending on the input sequence length for RNAs shorter than 1,000 nt is shown for RNAfold [Hofacker, 1994; Lorenz, 2011], LinearFold [Huang, 2019], MXfold2 [Sato, 2021] and KnotFold [Gong, 2024]. The computation times are displayed for reference in a log scale.

The prediction performances as well as the computation times are shown in Fig 5.3 depending on the sequence length. The MLP encoder performs worse than the other two. This can be explained by the fact that the MLP architecture is applied to each nucleotide independently, meaning that the context information can only be found in the module insertion features. Modules are specific forms of structural motifs that involve several nucleotides interacting with each other. We describe modules in Section 2.1.5. While these features do contain indirect information about the other nucleotides involved in the same modules, it can be lacking to rely only on them to find context information. On the other hand, to encode the information at some position in the sequence, the BiLSTM and 1D CNN encoder architectures allow to use other nucleotides information. The BiLSTM encoder is able to do this as it treats the RNA sequentially, keeping previous nucleotide information in its hidden state, in both directions since it is bidirectional. Considering the 1D CNN encoder, repeated dilated convolutional layers with varying dilation rates allow to access other nucleotides information at different points in the sequence. Out of the three architectures, the 1D CNN encoder yields the best prediction accuracy overall. Furthermore, the 1D CNN encoder results in the fastest computation time, while the BiLSTM is the slowest of the three. Considering this, we choose the 1D CNN architecture for the encoder.

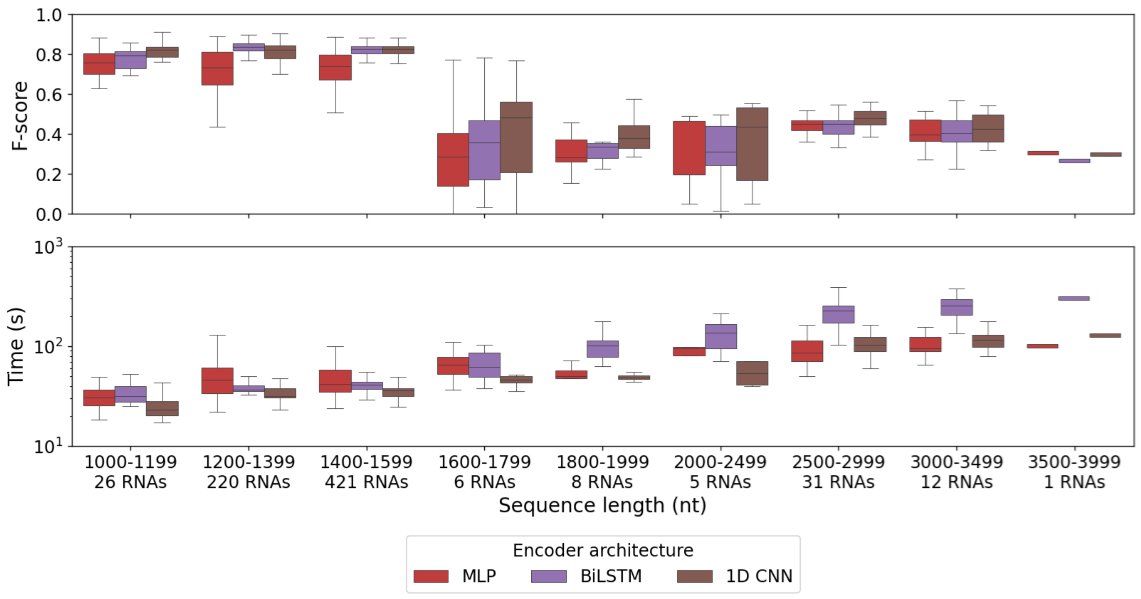

Figure 5.3: **Performance of DivideFold depending on the encoder architecture in the divide model's deep neural network, by sequence length on the Validation dataset.** The F-score of DivideFold is shown for the MLP, BiLSTM and 1D CNN encoders, for secondary structure prediction depending on the input sequence length for RNAs longer than 1,000 nt. KnotFold is used as the structure prediction model for DivideFold. The computation times are displayed for reference in a log scale.

5.2.3 Pseudoknot removal function

We also evaluate which pseudoknot removal function to use for generating training labels during the training of the divide model's deep neural network. Indeed, as mentioned in Section 4.3.3, pseudoknots must be removed for the generation of training labels. Yet, there is no inherent truth as to which base pairs should be removed, and several approaches with varying results can be considered based on what matters most to the user. The choice of the pseudoknot removal function is important as methods that preserve more of the core structural elements are likely to lead to training labels for the cut points that are more biologically meaningful. This, in turn, helps the divide model learn to partition sequences into more relevant fragments more effectively.

We evaluate here the main functions for pseudoknot removal [Smit, 2008]:

- Dynamic programming by base pairs (DPBP). This method uses a dynamic programming algorithm to maximize the total number of base pairs in the non-pseudoknotted structure.

- Conflict elimination by conflicts (CEC). Conflict elimination approaches start from the pseudoknotted structure and iteratively remove conflicting base pairs until no pseudoknots remain. In CEC, base pairs causing the most conflicts are removed first.
- Conflict elimination by length (CEL). Another conflict elimination method, CEL removes the shortest conflicting stems first.
- Incremental by order (IO). Incremental approaches start from an empty structure and add base pairs as long as no pseudoknots appear. In IO, base pairs are added in order from the $5'$ end to the $3'$ end.
- Incremental by length (IL). This is another incremental approach, where the longest stems are added first.
- Incremental by range (IR). This incremental approach prioritizes adding short-range base pairs first.

It should be noted that the length of a stem represents the number of base pairs that it contains, while the range of a base pair stands for the number of nucleotides in the sequence between the two bases.

We display in Table 5.3 the mean recall, precision and F-score of DivideFold depending on the pseudoknot removal function used during training. We only consider sequences longer than 1,000 nt and we use a maximum fragment length of 1,000 nt. KnotFold is used as the structure prediction tool on the fragments. It can be seen that the IO approach performs the worst among considered methods. This makes sense, as there is no evidence that the order of base pairs in a sequence would be linked to the importance of their role in the structure. The DPBP method performs slightly worse than the other approaches. This can also be explained, as maximizing the total number of base pairs does not necessarily mean that the most structurally important elements are kept. The CEC, CEL and IL methods all perform similarly and lead to satisfactory scores, even though the resulting structures differ, indicating that the number of conflicts of a base pair or the length of a stem are valid criteria in determining which base pairs are most essential to the structure. Lastly, the IR approach achieves higher recall, precision, and F-score compared to the other methods. This outcome is expected, as training labels are generated between minimally nested stems, which involve the base pairs with the highest range. Therefore, prioritizing long-range interactions in the structure naturally helps precisely identify the minimally nested stems, and consequently, the training labels. This leads to fragments that are more biologically relevant for the divide model, and ultimately to better predicted secondary structures for DivideFold as a whole. For that reason, we choose to use the IR method for removing pseudoknots when training the divide model's deep neural network.

Table 5.3: **Mean performance of DivideFold depending on the pseudoknot removal function used for training, on the Validation dataset.**

Pseudoknot removal function	**Recall**	**Precision**	**F-score**
Dynamic programming by base pairs (DPBP)	0.723	0.803	0.759
Conflict elimination by conflicts (CEC)	0.727	0.811	0.765
Conflict elimination by length (CEL)	0.727	0.810	0.764
Incremental by order (IO)	0.712	0.798	0.751
Incremental by length (IL)	0.727	0.810	0.764
Incremental by range (IR)	**0.734**	**0.813**	**0.770**

The mean recall, precision and F-score for secondary structure prediction for RNAs longer than 1,000 nt are shown for DivideFold, using KnotFold as the structure prediction model, for different pseudoknot removal functions used for generating training labels during the training of the divide model's deep neural network.

5.2.4 Fine-tuning length threshold

Even though longer RNAs are more relevant for this study, they are lacking in the available RNA secondary structure databases, including bpRNA-1m. Thus, as described in Section 4.3.6, we consider first pre-training the divide model's deep neural network on shorter sequences to learn fundamental patterns, and then fine-tuning it on the few longer training samples that are available above a certain length threshold to learn more complex relationships. For this reason, we choose possible length thresholds for pre-training and fine-tuning from the distribution of sequence lengths in the available training samples. However, as mentioned in Section 4.3.4, the actual number of available training samples is not limited to the original sequences in the Train dataset. Rather, each sequence in the Train dataset is partitioned recursively, yielding a new training sample at each iteration. This allows to substantially increase the amount of available training samples, although most of the new training samples are shorter sequences since they come from the partition of a larger sequence in the Train dataset. This is depicted in Fig 5.4. We can see that his phenomenon creates almost no new training samples longer than 1,000 nt, for the reason stated above. However, it adds many new samples shorter than 600 nt, which are in fact the structural domains of larger RNAs in the Train dataset. We ignore samples shorter than 100 nt for training, as such sequences may not be long enough to accommodate multiple structural domains and, consequently, relevant cut points.

Among the total training samples, 20,424 are between 400 and 1,000 nt long. We consider this a sufficiently large basis for pre-training the divide model's deep neural network on small RNAs and deem it unnecessary to include shorter RNAs. Thus, we decide to pre-train the divide model's deep neural network on sequences longer than 400 nt. This would not have been possible without the data augmentation phenomenon described above, as only 2,312 unique sequences in the Train dataset are between 400 and 1,000 nt, which

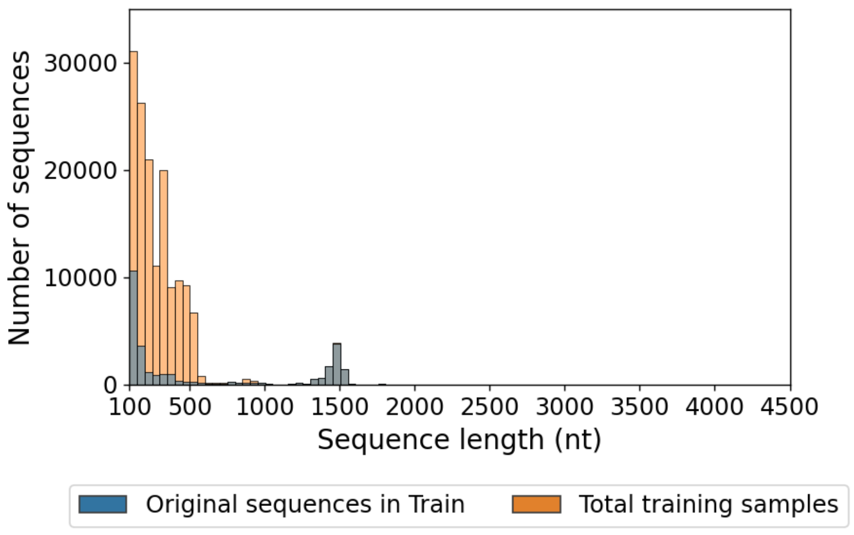

Figure 5.4: **Distribution of sequence length for original training sequences and total training samples.** The original sequences in the Train dataset are shown in blue, and the total training samples after exploiting each iteration of the partition process when generating training labels are shown in orange. Very few extra training samples longer than 1,000 nt are obtained. However, the amount of available training samples shorter than 600 nt increases considerably. Sequences longer than 1,600 nt are very rare, which is why they are barely visible in this figure.

is almost 10 times less than what is now available. This highlights the crucial role of exploiting each iteration in the partition process used for generating training labels, as it significantly expands the training set. We also include long sequences for pre-training the divide model's deep neural network as it can only benefit the model's training.

Regarding the second step of training, i.e. the fine-tuning, we consider three possibilities: (i) not performing any fine-tuning, (ii) fine-tuning the divide model's deep neural network on sequences longer than 1,000 nt since these are the ones we focus on in this study, and (iii) fine-tuning the divide model's deep neural network on sequences longer than 1,600 nt as very few of them are available. Indeed, training samples between 1,600 and 4,500 nt in length constitute valuable but rare data. Although these are the sequences that we are most interested in, only 332 of them are available, representing approximately 1% of the samples used for pre-training. As a result, the divide model's deep neural network may struggle to learn how to handle them during pre-training. Therefore, it is possible that the divide model's deep neural network could benefit from specifically focusing on those sequences to better learn how to partition them.

We display the results for these three fine-tuning strategies in Fig 5.5. The results for the three strategies are similar for sequences shorter than 1,600 nt. However, fine-tuning the divide model's deep neural network on sequences longer than 1,600 nt tends to lead to a better F-score for the secondary structure prediction of such longer sequences. This is expected, as fine-tuning the divide model's deep neural network on sequences longer than 1,600 nt allows it to better partition those sequences, therefore facilitating secondary structure prediction. Importantly, fine-tuning the divide model's deep neural network this way does not degrade its performance on shorter RNAs. In view of these results, we choose to first pre-train the divide model's deep neural network on training samples longer than 400 nt, then fine-tune it on samples longer than 1,600 nt.

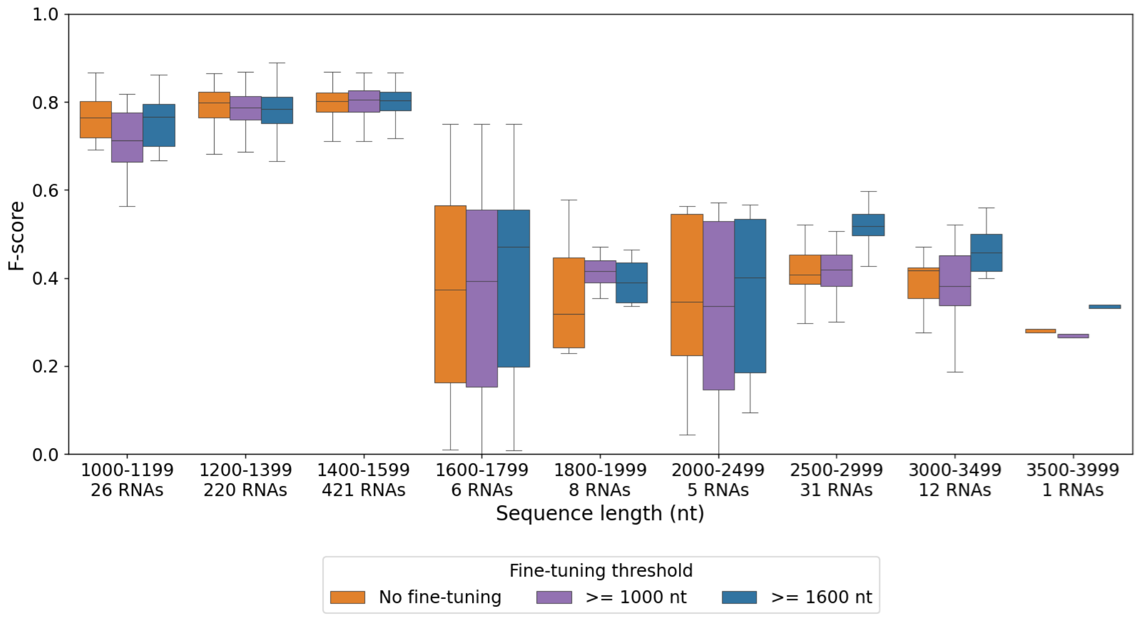

Figure 5.5: **Performance of DivideFold depending on the fine-tuning threshold, by sequence length on the Validation dataset.** The F-score of DivideFold is shown for the different fine-tuning strategies, for secondary structure prediction depending on the input sequence length for RNAs longer than 1,000 nt. KnotFold is used as the structure prediction model for DivideFold.

5.2.5 Maximum fragment length

The maximum length of a fragment is a crucial hyperparameter in our method. If the maximum fragment length is too low, the fragments will be shorter, but at the cost of a higher risk of breaking base pairs, ultimately leading to a drop in performance overall. On the other hand, if the maximum fragment length increases, less base pairs will be broken during partition, but the resulting fragments will be longer, making the structure prediction task harder.

In order to determine the optimal value for the maximum fragment length, we introduce two metrics of interest for measuring the quality of the divide model:

- The break rate, which is the percentage of base pairs that are broken during partition, excluding pseudoknots.

- The compression rate, which reflects the factor by which the fragments are shortened compared to the original sequence:

$$compression = 1 - \sum_{i=1}^{p} \left(\frac{N_i}{N}\right)^2 \tag{5.1}$$

where $N_1, ..., N_p$ are the lengths of the p final fragments and N is the sequence length, such that $\sum_{i=1}^{p} N_i = N$

Generally, predicting the structure of a shorter sequence is easier, meaning that the compression rate should be as high as possible. At the same time, the break rate should be as low as possible, so as to preserve the structure. The two metrics are closely linked because a higher compression rate is due to a deeper partition and will generally lead to a higher break rate. We evaluate our approach for different values of the maximum fragment length up to 1,200 nt. We cannot consider higher values for the maximum fragment length because most RNAs in the dataset are either shorter than this threshold or only slightly longer. Since sequences shorter than the maximum fragment length cannot be partitioned, raising the threshold further would prevent us from partitioning nearly all of these sequences. To ensure that we compare the results on the same data for the different values of the maximum fragment length, we only consider here sequences longer than 1,200 nt This is because shorter RNAs would not be partitioned for the higher values of the maximum fragment length, and all values would not be evaluated on the same data.

We display the mean compression rate and break rate of our divide model for different values of the maximum fragment length in Fig 5.6. The F-score for secondary structure prediction is also indicated for reference. The F-score is highest when the maximum fragment length is between 800 nt and 1,000 nt, even though the variations in F-score are slim. At 1,000 nt for the maximum fragment length compared to 800 nt, the compression rate decreases (67.1% instead of 70.0%) but the break rate is also lower (2.0% instead of 2.8%).

To choose a value for the maximum fragment length, we perform a more detailed analysis. We measure the F-score for the secondary structure prediction and we separate the sequences into groups according to their lengths. We show the results in Fig 5.7. The performance is slightly better for a maximum fragment length of 800 nt compared to 1,000 nt for RNAs shorter than 1,600 nt, but it is slightly worse for RNAs longer than 1,600 nt. As a whole, it appears that a maximum fragment length of 1,000 nt yields the best performance overall when considering the various RNA sequence length ranges studied.

In light of these results, we choose a value of 1,000 nucleotides for the maximum length of a fragment in our approach in this chapter. Therefore, RNAs shorter than these thresholds will not be partitioned in our approach. Because of this, we focus on longer RNAs in the rest of our experiments.

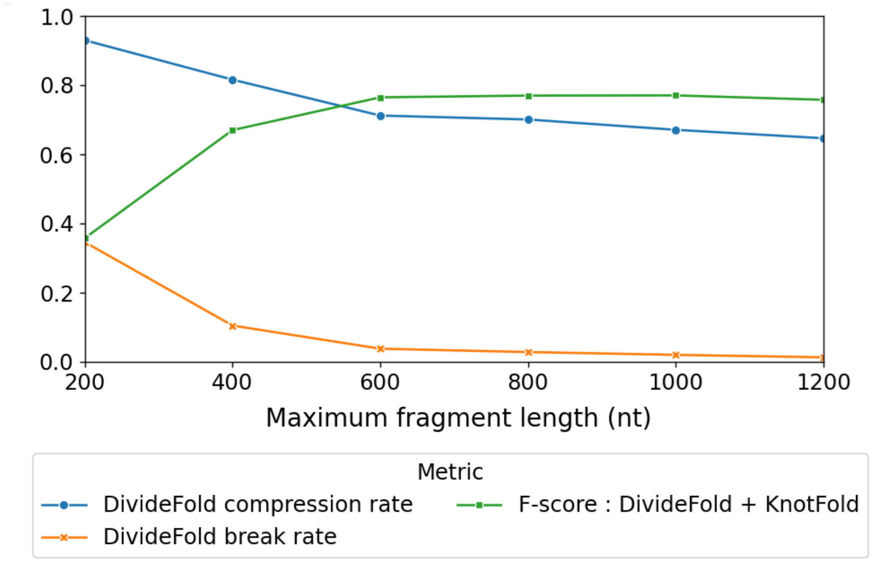

Figure 5.6: **Compression and break rates with respect to the maximum fragment length on the Validation dataset.** The mean compression rate and break rate of our divide model are shown for different values of the maximum fragment length hyperparameter, for sequences longer than 1,200 nucleotides. The F-score for secondary structure prediction is shown for reference.

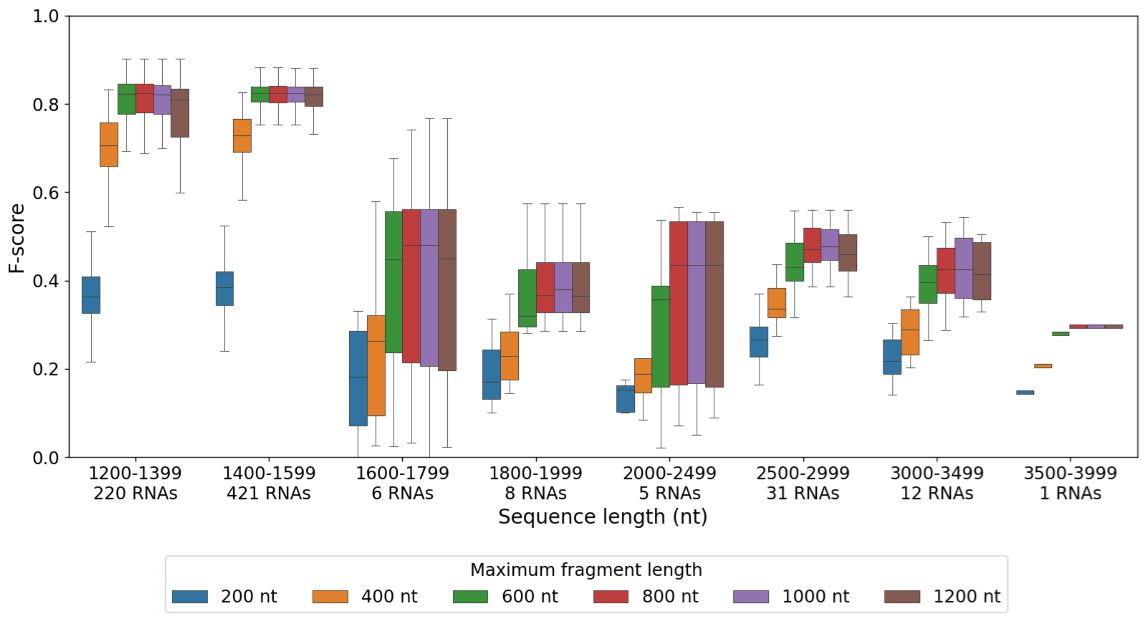

Figure 5.7: **Secondary structure F-score for DivideFold with respect to the maximum fragment length, by sequence length, on the Validation dataset.** The F-score for secondary structure prediction depending on the input sequence length for RNAs longer than 1,200 nt is shown for DivideFold, using KnotFold as the structure prediction model, for different values of the maximum fragment length.

We also show the distribution of fragment length in Fig 5.8, given the hyperparameters chosen in this section, including a maximum fragment length of 1,000 nt. Fragments of different lengths between 6 nt and 1,000 nt are obtained after partition depending on the RNA, although most fragments are shorter than 600 nt.

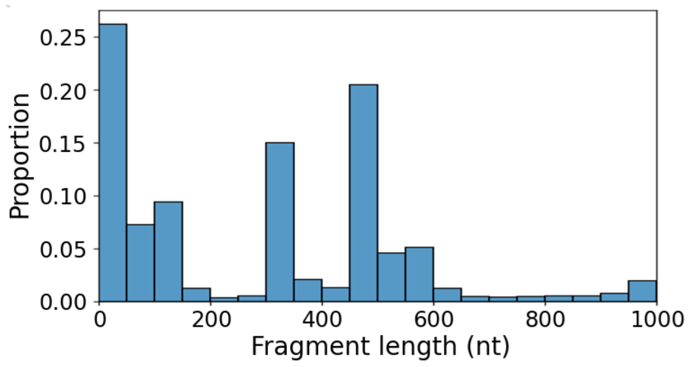

Figure 5.8: **Fragment length distribution.** Given the chosen hyperparameters, considering a maximum fragment length of 1,000 nt, fragments of various lengths between 6 nt and 1,000 nt are found after partition depending on the RNA. The majority of fragments are shorter than 600 nt.

5.3 Results

We now conduct a benchmark comparing DivideFold to some of the different most effective tools from the literature that do not predict pseudoknots. We first evaluate their performance for secondary structure prediction excluding pseudoknots and report their computation time. We then conduct an analysis of the 16S ribosomal RNA as a case study. Next, we assess their performance for RNAs between 500 and 1,000. Finally, this allows us to study the generalization of the benchmarked tools to unknown RNA families. All experiments to assess the performance of the benchmarked methods are performed on the Test dataset, or on bpRNA-NF-15.0 when evaluating generalization to unknown families in Section 5.3.6.

5.3.1 Benchmarked methods

We choose to include in our benchmark the same tools that we have considered for hyperparameter tuning: RNAfold [Hofacker, 1994; Lorenz, 2011], LinearFold [Huang, 2019], MXfold2 [Sato, 2021] and KnotFold [Gong, 2024]. These tools are indeed able to process long RNAs and belong to the state-of-the-art for secondary structure prediction excluding pseudoknots. We do not consider in our benchmark other tools that cannot handle long RNAs. We also ignore Tfold [Engelen, 2010] since it requires homologous sequences.

Additionally, RNA-par [Zhao, 2023] is a tool developed in 2023 that, similarly to DivideFold, predicts a partition to subdivide an RNA sequence into shorter fragments instead of predicting a secondary structure. As in our approach, it is then possible to obtain a predicted structure for the sequence by using an existing structure prediction model on the fragments and recombining them together. RNA-par uses a deep learning architecture based on both 1D convolutional layers and recurrent layers to separate an RNA sequence into continuous fragments. However there are some key differences with our work. (i) RNA-par does not allow external parts to be merged, yielding only continuous fragments. Thus, any long-range stem will cause a large region to be impossible to partition without losing that stem. This will likely happen often, given that these long-range stems are common motifs in RNA structure. (ii) The partition process of RNA-par is only made of one iteration. Yet, more nested structures will require to repeat steps in order to partition deeper while preserving the structure. Thus, it is impossible to further partition large nested structures in a single iteration without breaking base pairs. Because of this, large regions in the sequence may be left to the structure prediction model, instead of short fragments as intended. (iii) RNA-par does not remove pseudoknots beforehand when building training labels, which can also cause large regions to be linked and impossible to partition without breaking base pairs. This can lead to the absence of training labels in large regions of the sequence, and the RNA-par's inability to process such regions as a whole. (iv) RNA-par is solely trained and tested on RNAs shorter than 200 nucleotides and does not consider long RNAs. Nonetheless, given the similarities to DivideFold, RNA-par would be a valuable tool to compare to in our experiments. Unfortunately, we could not include RNA-par in our benchmark because the code provided in the website mentioned in the article could not be used.

As in Section 5.2, all methods are used with their default parameters, and we use MXfold2 and KnotFold with their weights pre-trained on the TR0 [Sato, 2021] dataset, which is included in our Train dataset.

5.3.2 Secondary structure prediction results

We evaluate here the ability of DivideFold to predict the secondary structure of long RNAs. For this analysis, we compare DivideFold to RNAfold [Hofacker, 1994; Lorenz, 2011], LinearFold [Huang, 2019], MXfold2 [Sato, 2021] and KnotFold [Gong, 2024] for sequences longer than 1,000 nt. KnotFold could not process RNAs longer than 2,500 nt in our experiments because of memory overflows. We show the results in Table 5.4. Our approach demonstrates significantly higher average recall, precision, and F-score compared to RNAfold, LinearFold, MXfold2, and KnotFold. DivideFold is the only method to achieve an F-score above 0.75, while the F-scores of the other evaluated tools remain around 0.5. It also demonstrates a recall above 0.7 and a precision exceeding 0.8. In contrast, the other benchmarked tools show both recall and precision values near 0.45, except for the precision of KnotFold, which is approximately 0.6.

Additionally, as the input sequence length is a key factor in our approach, we present

Table 5.4: **Secondary structure prediction performance on the Test dataset.**

Model	**Recall**	**Precision**	**F-score**
DivideFold + KnotFold	**0.737**	**0.813**	**0.772**
KnotFold	0.451	0.609	0.517
MXfold2	0.477	0.429	0.450
LinearFold	0.467	0.454	0.459
RNAfold	0.464	0.390	0.423

The performance of DivideFold, RNAfold [Hofacker, 1994; Lorenz, 2011], LinearFold [Huang, 2019], MXfold2 [Sato, 2021] and KnotFold [Gong, 2024] is reported here for RNAs longer than 1,000 nt.

the secondary structure prediction performance depending on the RNA sequence length in Fig 5.9. The number of RNAs in each length range is indicated below the figure for reference. Our approach exhibits better performance for RNAs up to 2,500 nucleotides in length compared to the other methods. However, DivideFold struggles to accurately predict the secondary structures of longer RNAs, with performance declining as the number of available sequences in a given length range decreases. This drop may be attributed to the scarcity of training data for RNAs longer than 1,600 nucleotides, which likely hampered our ability to derive reliable results for such sequences. Indeed, the availability of sufficient training data is a crucial factor that can significantly impact the performance of deep learning approaches.

LinearFold, which is not based on deep learning, shows better results for sequences longer than 2,500 nt.

5.3.3 Time cost

We run our experiments on the Jean Zay supercomputer[1] using one V100 GPU with 16 Go of memory and we measure the computation time. Our approach remains fast and ensures scalability to long RNAs, as evidenced in Fig 5.10, addressing the computational issues faced by many models, although not as fast as RNAfold or LinearFold. By partitioning the sequences into shorter fragments, our approach allows to decrease significantly the computation time. This is because RNA structure prediction algorithms tend to have a time complexity of at least $O(n^2)$, meaning that it is faster to process several shorter sequences rather than one large sequence. It is confirmed that DivideFold runs in linear time in practice. Furthermore, the partition step is very fast (less than 0.3 seconds on average). Thus, the computation time is almost only due to the structure prediction model itself, which is KnotFold here, and not the divide model. This shows that DivideFold has the potential to be faster than it already is if combined with a fast secondary structure predic-

[1]Jean Zay supercomputer: http://www.idris.fr/eng/jean-zay/jean-zay-presentation-eng.html

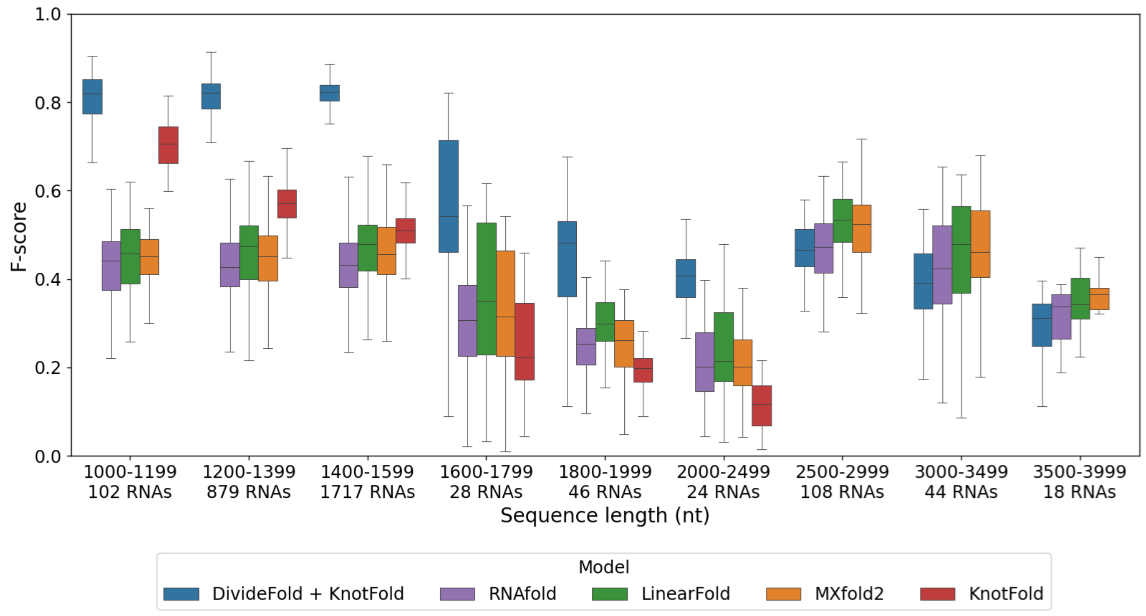

Figure 5.9: **Secondary structure prediction performance by sequence length on the Test dataset.** The F-score is shown depending on the input sequence length for RNAs longer than 1,000 nt. The performance of DivideFold, RNAfold [Hofacker, 1994; Lorenz, 2011], LinearFold [Huang, 2019], MXfold2 [Sato, 2021] and KnotFold [Gong, 2024] is reported here. KnotFold is unable to handle RNAs longer than 2,500 nt.

tion model. LinearFold is the fastest tool among those evaluated here, as it is designed to be an approximate and faster version of RNAfold as seen in Section 3.2.1.

5.3.4 Case study: the 16S ribosomal RNA

We now examine a concrete example to illustrate the partition approach used by DivideFold. We choose the 16S ribosomal RNA, as it is of significant importance and was extensively studied throughout the years. It has a length of 1,512 nucleotides and its structure is well-known. To showcase a practical example, we display its structure in Fig 5.11 and we highlight the final cut points selected by our divide model in red dots. The 16S ribosomal RNA is composed of a central domain, a 5′ domain, a 3′ major domain, and a 3′ minor domain [Hori, 2021]. It can be seen that the RNA is accurately partitioned into its different domains by the chosen cut points. Our divide model does not break any base pairs here excluding pseudoknots. Moreover, it requires only 0.2 seconds to compute the cut points.

The results for the secondary structure prediction of the 16S ribosomal RNA are shown in Table 5.5. DivideFold yields strong performance, surpassing RNAfold, LinearFold, MXfold2 and KnotFold in terms or recall, precision, and F-score, while remaining reasonably fast. It achieves recall, precision, and F-score values near 0.8, whereas the other evaluated tools show values close to 0.45, with the exception of the precision of KnotFold, which is

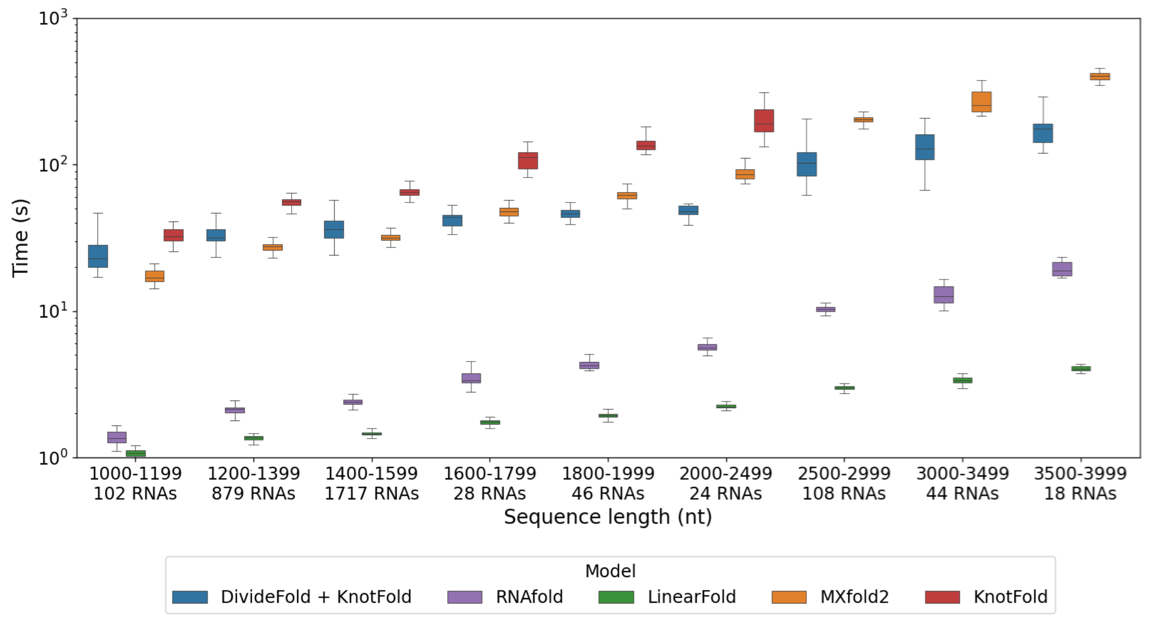

Figure 5.10: **Computation time by sequence length.** The time is displayed in a log scale depending on the input sequence length for DivideFold, RNAfold [Hofacker, 1994; Lorenz, 2011], LinearFold [Huang, 2019], MXfold2 [Sato, 2021] and KnotFold [Gong, 2024] for RNAs longer than 1,000 nt on the Test dataset. KnotFold is unable to handle RNAs longer than 2,500 nt.

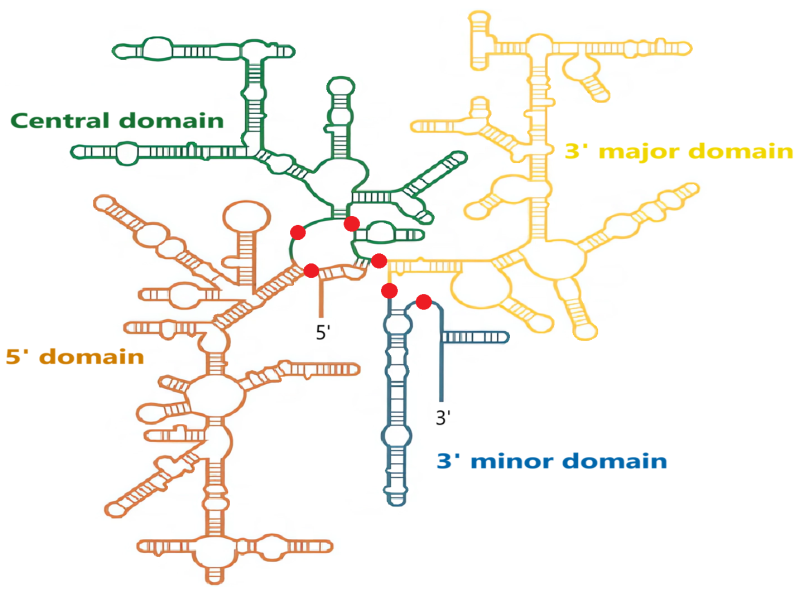

Figure 5.11: **Secondary structure of the 16S ribosomal RNA.** The final cut points chosen by our divide model are shown in red dots. Adapted from [Williamson, 2005].

around 0.55. Even though our approach takes 25 more seconds to compute compared to LinearFold and RNAfold, the prediction is significantly better.

Table 5.5: **Results for secondary structure prediction on the 16S ribosomal RNA.**

Model	Recall	Precision	F-score	Time
DivideFold + KnotFold	**0.785**	**0.869**	**0.825**	0:00:27
KnotFold	0.368	0.564	0.446	0:01:11
MXfold2	0.463	0.453	0.458	0:00:36
LinearFold	0.408	0.411	0.409	**0:00:02**
RNAfold	0.461	0.419	0.439	0:00:03

The performance of DivideFold, RNAfold [Hofacker, 1994; Lorenz, 2011], LinearFold [Huang, 2019], MXfold2 [Sato, 2021] and KnotFold [Gong, 2024] is reported here, as well as the computation time.

5.3.5 Study of RNAs shorter than 1,000 nt

We assessed in the previous sections the performance of DivideFold for RNAs longer than 1,000 nt. Here, we evaluate DivideFold's abilities for RNAs shorter than 1,000 nt. We compare DivideFold to RNAfold, LinearFold, MXfold2 and KnotFold for sequences between 500 nt and 1,000 nt. Since the maximum fragment length must be lower than the sequence length for the sequence to be partitioned, it cannot be higher than 500 nt here as any higher value would prevent partition. As higher values for the maximum fragment length lead to better performance up to 1,000 nt as evidenced in Fig 5.7, we choose a value of 500 nt for this hyperparameter of DivideFold in this experiment.

We show the results for secondary structure prediction depending on the RNA sequence length in Fig 5.12. KnotFold and DivideFold exhibit significantly better performance than RNAfold, LinearFold, and MXfold2 for RNAs between 500 and 1,000 nucleotides. Although DivideFold achieves satisfactory results, it does not outperform KnotFold in this length range. However, its performance progressively approaches that of KnotFold as sequence length increases, eventually surpassing it for RNAs longer than 1,000 nucleotides as displayed in Fig 5.9. This suggests that DivideFold is better suited for RNAs longer than 1,000 nt, which become increasingly difficult to deal with for other tools.

5.3.6 Generalization to unseen families

We assess here the ability to generalize to families unseen in the training dataset for secondary structure prediction. For this analysis, we use the bpRNA-NF-15.0 dataset, which we built to exclusively contain RNA families that do not exist in bpRNA-1m as seen in Section 5.1.2, and thus, in the Train dataset. This enables us to perform family-wise evaluation. Yet, bpRNA-NF-15.0 does not include sequences longer than 1,000 nt as seen in Section 5.1.2. This is an issue as we show above that DivideFold is less suited for RNAs shorter than 1,000 nt, even for sequence-wise evaluation, which is typically an

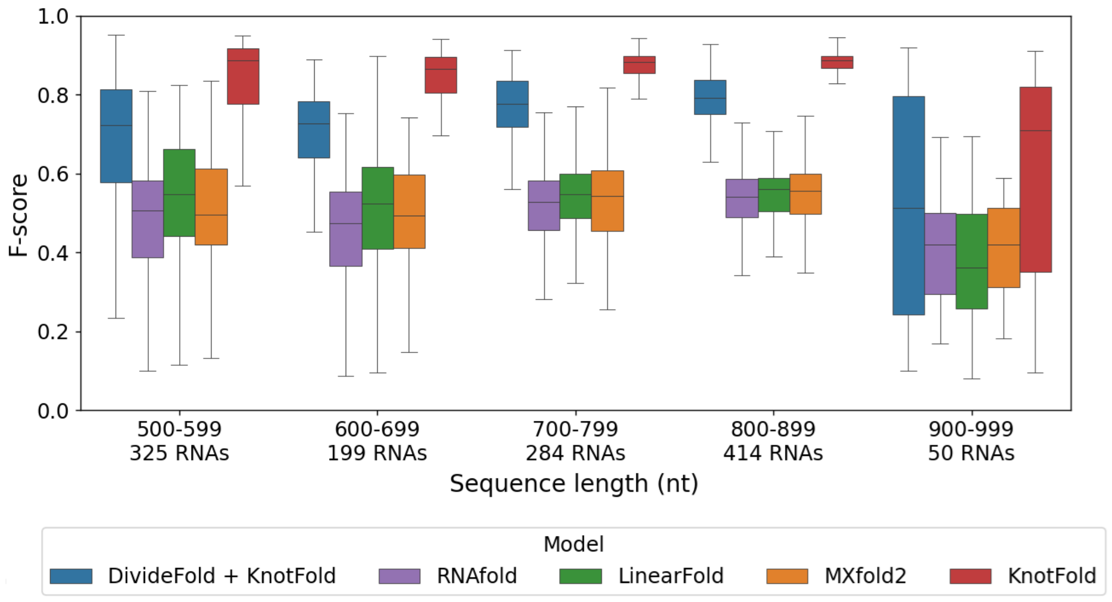

Figure 5.12: **Secondary structure prediction performance by sequence length for RNAs between 500 nt and 1,000 nt on the Test dataset.** The F-score is shown depending on the input sequence length. The performance of DivideFold, RNAfold [Hofacker, 1994; Lorenz, 2011], LinearFold [Huang, 2019], MXfold2 [Sato, 2021] and KnotFold [Gong, 2024] is reported here.

easier task than family-wise evaluation. Thus, we do not expect great results for DivideFold on bpRNA-NF-15.0. Nonetheless, we perform this analysis on bpRNA-NF-15.0 for RNAs between 500 and 1,000 nt, evaluating DivideFold, RNAfold, LinearFold, MXfold2 and KnotFold. As in Section 5.3.5, we use a maximum fragment length of 500 nt for DivideFold to allow partition.

We show the secondary structure prediction results on bpRNA-NF-15.0 in Table 5.6. While DivideFold does not outperform KnotFold, it achieves better results than RNAfold, LinearFold, and MXfold2. This suggests that non-trained methods like RNAfold and LinearFold, though often considered to be more robust to unseen data, may remain less effective than deep learning approaches, even when generalizing to newly discovered RNA families.

In Table 5.7, we compare these results on bpRNA-NF-15.0 to those obtained in Section 5.3.5 on the Test dataset for the same sequence lengths. The performance loss on bpRNA-NF-15.0 is highest for DivideFold and KnotFold, even though this is a little less significant for DivideFold, showing that the generalization to unseen families remains a serious concern for certain deep learning-based approaches. Notice however that the mean performance decreases for all benchmarked methods, including RNAfold and LinearFold which are not trained methods. This highlights the difficulty of handling these newly discovered sequences.

Table 5.6: **Secondary structure prediction performance on bpRNA-NF-15.0.**

Model	**Recall**	**Precision**	**F-score**
DivideFold + KnotFold	0.393	0.462	0.422
KnotFold	0.511	**0.479**	**0.493**
MXfold2	**0.529**	0.339	0.412
LinearFold	0.459	0.412	0.426
RNAfold	0.526	0.319	0.396

The performance of DivideFold, RNAfold [Hofacker, 1994; Lorenz, 2011], LinearFold [Huang, 2019], MXfold2 [Sato, 2021] and KnotFold [Gong, 2024] is reported here for RNAs between 500 and 1,000 nt.

Table 5.7: **Mean performance comparison for secondary structure prediction between the Test and bpRNA-NF-15.0 datasets.**

Model	**F-score on Test**	**F-score on bpRNA-NF-15.0**	**F-score gap**
DivideFold + KnotFold	0.709	0.433	-0.276
KnotFold	**0.821**	**0.493**	-0.328
MXfold2	0.514	0.412	-0.102
LinearFold	0.525	0.426	-0.099
RNAfold	0.499	0.396	-0.103

The mean F-score for secondary structure prediction of DivideFold, RNAfold [Hofacker, 1994; Lorenz, 2011], LinearFold [Huang, 2019], MXfold2 [Sato, 2021] and KnotFold [Gong, 2024] is reported here on the Test and bpRNA-NF-15.0 datasets, for RNAs between 500 and 1,000 nt.

5.4 Discussion

DivideFold, which combines a recursive partition and deep learning techniques, offers a promising direction for enhancing the prediction of secondary structures in long RNAs. Our approach addresses the challenges associated with lengthy RNA sequences by partitioning long sequences into shorter fragments through strategic cut point selection and leveraging existing structure prediction models designed for small RNAs. Our approach runs in linear time and has constant memory usage, making it capable of processing long RNAs.

We perform a benchmark in which we demonstrate the superior performance of our method compared to RNAfold, LinearFold, MXfold2 and KnotFold for RNAs longer than 1,000 nt. DivideFold exhibits a stronger performance overall for RNA secondary structure prediction. Nonetheless, our model struggles to find accurate cut points for long RNAs beyond 2,500 nucleotides.

We also attempt to check for generalization to new RNA families unseen in the Train dataset, but the absence of data for sequences longer than 1,000 nt belonging to newly discovered families prevents us from conducting a proper analysis for DivideFold. Nevertheless, we study this for RNAs between 500 and 1,000 nt and show that the performance drops significantly for all benchmarked tools, suggesting that predicting the structures of RNAs from new families remains a tough challenge.

In the future, acquiring an increased number of long RNA sequences and secondary structures will be essential to enable more accurate predictions and comprehensively evaluate the model's performance. Finally, DivideFold will be able to be paired with any new accurate RNA secondary structure prediction method, benefiting from future research.

6 Secondary structure prediction including pseudoknots of long RNAs

The prediction of pseudoknots is particularly important since they give insights into their folding in three-dimensional space. However, pseudoknot prediction remains a tough challenge. Existing approaches exhibit difficulties in accurately finding the pseudoknots in RNA and often face computational hurdles. This is especially true for long RNAs. In this chapter, we extend DivideFold to the prediction of pseudoknots in long RNAs. By using a secondary structure prediction tool capable of predicting pseudoknots for the fragments, and by ensuring that the fragment size is sufficiently large, DivideFold can partition an RNA sequence and allow the prediction of pseudoknots in the fragments while preserving the structural integrity of the sequence. As a result, DivideFold can predict the secondary structure including pseudoknots of long RNA sequences.

In this chapter, we first discuss the extension of DivideFold to pseudoknot prediction. Then, we report our results for secondary structure prediction including pseudoknots and for pseudoknot prediction, and we discuss the contribution of DivideFold.

Formally, a pseudoknot is made of at least two nested groups of base pairs where each base pair (i, j) in the first group and each base pair (k, l) in the second group are such that $i < k < j < l$. An example is shown in Fig 2.3.

As mentioned in Section 4.2, pseudoknots must be removed before generating training labels. Thus, pseudoknots are ignored at the partition step. Notice nonetheless that by using a structure prediction model that is able to predict pseudoknots, it is possible to find pseudoknots at the structure prediction step. However, it is crucial that the pseudoknots are located within a single fragment to be recovered, since the structure prediction model is applied to each fragment independently. Nevertheless, it is still possible to find long-range pseudoknots, as fragments may be continuous or may be made of several disconnected parts, thereby containing regions that are far apart in the original structure. This is depicted in Fig 6.1. In this example, the RNA sequence includes five pseudoknots. Four of them can be recovered because they occur within a single fragment, continuous (red and yellow fragments) or discontinuous (blue/purple fragment). Therefore, the partition strategy of DivideFold can be leveraged to predict pseudoknots in long RNAs, helping find pseudoknots more easily by separating long RNAs into shorter structural domains, but it is critical that the pseudoknots lie within a single fragment to be recovered.

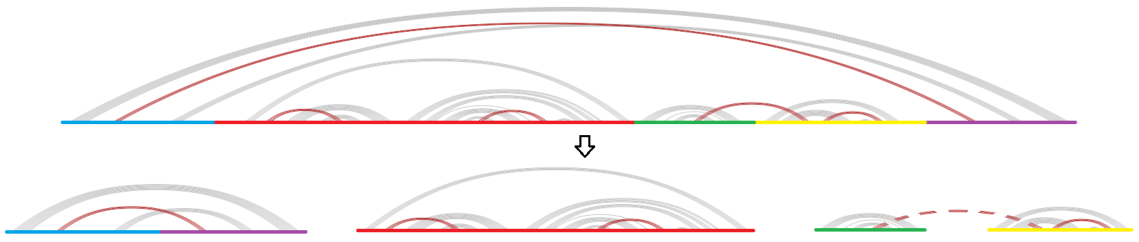

Figure 6.1: **Pseudoknot conservation during partition.** This is an example of a partition iteration where the RNA sequence includes five pseudoknots, displayed in red. Four of them can be recovered because they occur within a single fragment, continuous (red and yellow fragments) or discontinuous (blue/purple fragment). However, the pseudoknot represented in a dashed line occurs between the green and yellow fragments and will be lost.

6.1 Dataset preparation

In this chapter we use the same datasets as in Chapter 5, namely the Train dataset for the training of DivideFold, the Validation dataset for the choice of hyperparameters, and the Test dataset for evaluating the performances of the benchmarked methods. Additionally, we evaluate the generalization to unknown families on bpRNA-NF-15.0. However, we keep track of pseudoknots during evaluation in this chapter, as we aim to assess performance in both the prediction of overall secondary structure, including pseudoknots, and the prediction of pseudoknots specifically. Moreover, we restrict evaluation to RNA sequences that contain pseudoknots. This is because the benchmarked tools often predict many false positive pseudoknots, making it difficult to evaluate performance on sequences without pseudoknots, where only false positives can be observed. Focusing on pseudoknotted sequences allows for a more comprehensive evaluation, capturing true positives, false positives, and false negatives.

6.2 Hyperparameter tuning

As in the previous chapter, we perform experiments to select the hyperparameters of DivideFold, now applied to secondary structure prediction including pseudoknots. We also evaluate hyperparameters one at a time while keeping the others fixed instead of running a full grid search over all hyperparameter combinations, greatly reducing computation time.

We retain the same encoder architecture, pseudoknot removal function, and fine-tuning threshold as in Chapter 5, as these are hyperparameters of the divide model's deep neural network, which is not affected by pseudoknots. Indeed, pseudoknots are discarded during its training, meaning that the neural network does not consider pseudoknots when predicting the location of cut points. Therefore, there is no need to re-evaluate these hyperparameters in this chapter. However, it is necessary to re-evaluate the secondary structure prediction model applied to the fragments, and the maximum fragment length which controls the depth at which the divide model is recursively applied during partitioning. This parameter is critical for pseudoknot prediction, as a pseudoknot must lie within a single fragment to be correctly recovered, making fragment size a decisive factor. All experiments for the choice of DivideFold's hyperparameters are performed on the Validation dataset.

6.2.1 Evaluation of structure prediction models

We first evaluate the different methods to choose which one we will integrate in our approach as the structure prediction model. Since we now examine pseudoknots in the predictions, we consider tools designed for secondary structure prediction including pseudoknots in this chapter, which we described in Section 3.2.2. In this experiment, we include IPknot [Sato, 2011; Sato, 2022] (version 1.1.0), ProbKnot [Bellaousov, 2010] (version 6.4) and pKiss [Theis, 2010; Janssen, 2015] (version 2.2.12) as they are tools of reference in the literature for the prediction of secondary structure including pseudoknots. IPknot is also one of the fastest tools that is able to predict pseudoknots. Additionally, we evaluate UFold [Fu, 2022], which is a deep learning method often included in secondary structure prediction benchmarks and is able to predict pseudoknots. Finally, we consider KnotFold [Gong, 2024] for this experiment as well since it is a recent deep learning method that has shown convincing results for RNA secondary structure prediction, both with and without pseudoknots. All methods (IPknot, ProbKnot, KnotFold, pKiss and UFold) are used with their default parameters. We re-train UFold on the Train dataset. In the case of KnotFold, we use the weights pre-trained on the TR0 [Sato, 2021] dataset, which is included in the Train dataset. IPknot, ProbKnot and pKiss are not trained models.

We evaluate the structure prediction performance including pseudoknots of IPknot, ProbKnot, KnotFold, pKiss and UFold for short sequences below 1,000 nt. The prediction performances and the computation times are shown in Fig 6.2. UFold only accepts sequences shorter than 600 nt. It can be seen that KnotFold performs best out of the five methods on all the considered length ranges. It is also among the fastest tools in this experiment along with IPknot. Considering these results, we choose KnotFold as the structure prediction model for DivideFold in this chapter as well.

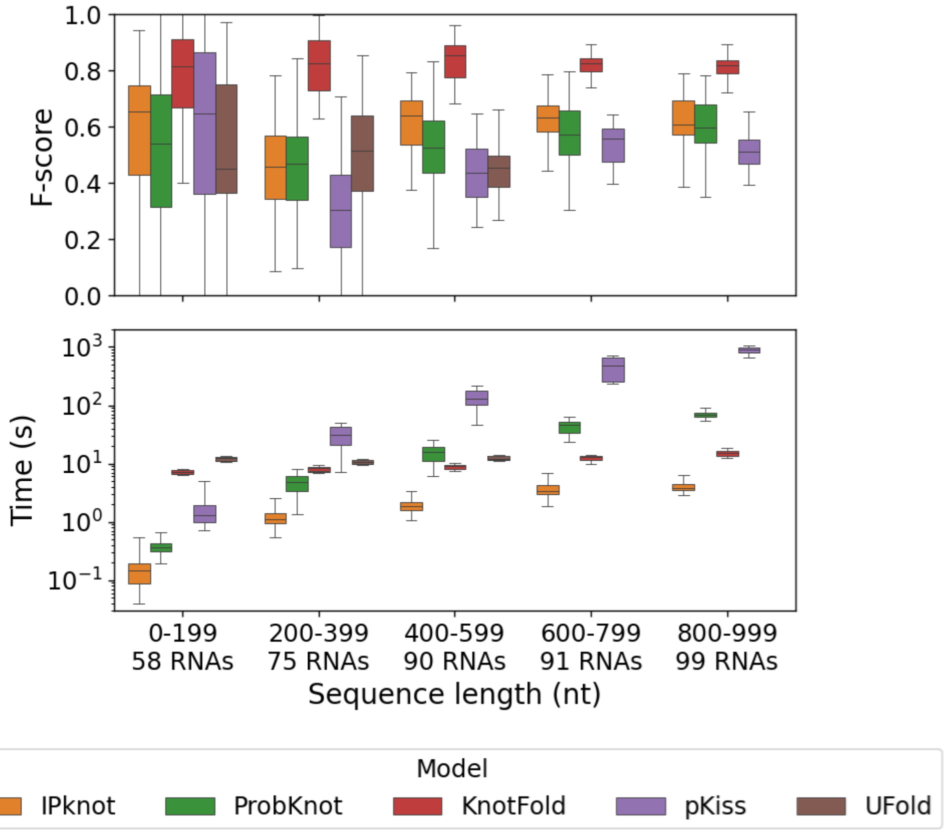

Figure 6.2: **Structure prediction models evaluation including pseudoknots by sequence length on the Validation dataset.** The F-score for secondary structure prediction including pseudoknots depending on the input sequence length for RNAs shorter than 1,000 nt is shown for IPknot [Sato, 2011; Sato, 2022], ProbKnot [Bellaousov, 2010], KnotFold [Gong, 2024], pKiss [Theis, 2010; Janssen, 2015] and UFold [Fu, 2022]. The computation times are displayed for reference in a log scale. UFold is unable to handle RNAs longer than 600 nt.

6.2.2 Maximum fragment length hyperparameter

As before, we evaluate our approach for different values of the maximum fragment length up to 1,200 nt.

The mean compression rate and break rate are the same as in the previous chapter. Indeed, the compression rate and break rate only depend on the divide model, and not on the structure prediction model, meaning that it does not matter whether pseudoknots are considered or not at the structure prediction step. Thus, the same conclusions hold and values of 800 nt and 1,000 nt for the maximum fragment length bring the best tradeoffs between compression rate and break rate.

We then assess the structure prediction performance including pseudoknots depending on the input sequence length to obtain more detailed insights. We separate the sequences into groups according to their lengths and show the results in Fig 6.3. As in the previous chapter, the performance is similar for a maximum fragment length of 1,000 nt compared to 800 nt for RNAs shorter than 1,600 nt, but slightly better for RNAs longer than 1,600 nt, and preferable as a whole.

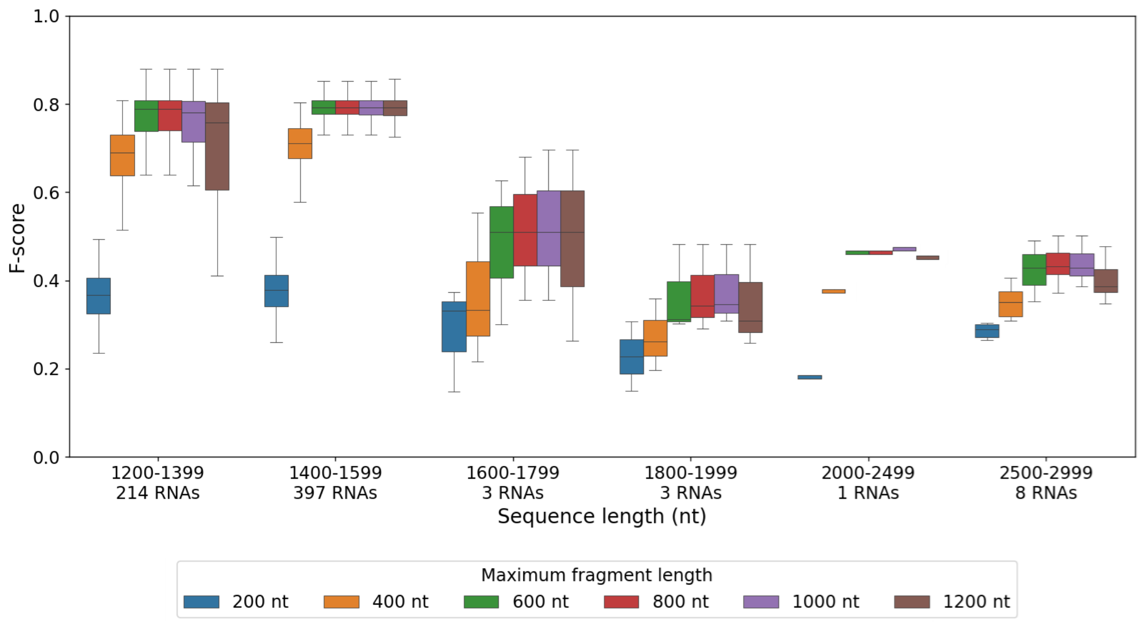

Figure 6.3: **Secondary structure F-score including pseudoknots for DivideFold with respect to maximum fragment length, by sequence length, on the Validation dataset.** The F-score for secondary structure prediction including pseudoknots depending on the input sequence length for RNAs longer than 1,200 nt is shown for DivideFold, using KnotFold as the structure prediction model, for different values of the maximum fragment length.

Taking these results into consideration, we choose a value of 1,000 nucleotides for the maximum length of a fragment for DivideFold in this chapter as well.

6.3 Results

We now perform a benchmark to compare DivideFold to different state-of-the-art tools that predict the secondary structure of RNAs including pseudoknots. We evaluate their performance for secondary structure prediction including pseudoknots, as well as pseudoknot prediction, and report their computation time. As in the previous chapter, we then conduct an analysis of the 16S ribosomal RNA as a case study. Finally, we assess their performance for RNAs between 500 and 1,000 and study their generalization capabilities to unknown RNA families. All experiments are performed on the Test dataset, or on bpRNA-NF-15.0 when evaluating generalization to unknown families in Section 6.3.7.

6.3.1 Benchmarked methods

We consider in our benchmark IPknot [Sato, 2011; Sato, 2022], ProbKnot [Bellaousov, 2010] and KnotFold [Gong, 2024], since they are all able to handle long RNAs and are among the most commonly benchmarked methods in the literature for secondary structure prediction including pseudoknots. We also review pKiss [Theis, 2010; Janssen, 2015] and UFold [Fu, 2022] in our experiments on bpRNA-NF-15.0 since this dataset only contains sequences shorter than 1,000 nt, but not on the Test dataset as pKiss and UFold are unable to deal with long RNAs. Indeed, UFold only accepts RNAs shorter than 600 nt, and pKiss has a high time complexity of $O(n^4)$, preventing us from testing it on RNAs longer than 1,000 nt in our experiments. We do not add to our benchmark other methods that can only handle small RNAs such as ShapeKnots [Hajdin, 2013], BiokoP [Legendre, 2018], BiORSEO [Becquey, 2020], SPOT-RNA [Singh, 2019] and SPOT-RNA2 [Singh, 2021]. We also disregard Tfold [Engelen, 2010] since it requires homologous sequences. Finally, we could not include RNA-par [Zhao, 2023] in our benchmark because the code provided could not be used.

As in Section 6.2, all methods are used with their default parameters. We re-train UFold on the Train dataset and we use the weights pre-trained on TR0 [Sato, 2021] (which is included in the Train dataset) for KnotFold.

6.3.2 Secondary structure prediction results including pseudoknots

We assess here the secondary structure prediction performance including pseudoknots of DivideFold for long RNAs. We compare DivideFold to IPknot [Sato, 2011; Sato, 2022], ProbKnot [Bellaousov, 2010] and KnotFold [Gong, 2024] for sequences longer than 1,000 nt. KnotFold could only process RNAs shorter than 2,500 nt because of memory overflows. We show the results in Table 6.1. DivideFold shows significantly better secondary structure performance including pseudoknots on average compared to IPknot, ProbKnot and KnotFold. Its mean recall, precision and F-score are all above 0.75. In contrast, the other benchmarked tools show values near 0.5 or less.

We display the secondary structure prediction performance including pseudoknots depending on the RNA sequence length in Fig 6.4. DivideFold exhibits better performance for secondary structure prediction including pseudoknots for RNAs up to 2,500 nucleotides long compared to the other methods, but similarly as in the previous chapter, the results are still lacking for RNAs longer than 2,500 nt. IPknot shows better results for sequences longer than 2,500 nt.

Table 6.1: **Secondary structure prediction performance including pseudoknots on the Test dataset.**

Model	**Recall**	**Precision**	**F-score**
DivideFold + KnotFold	**0.777**	**0.750**	**0.762**
KnotFold	0.520	0.330	0.403
IPknot	0.526	0.564	0.542
ProbKnot	0.470	0.431	0.448

The performance of DivideFold, IPknot [Sato, 2011; Sato, 2022],
ProbKnot [Bellaousov, 2010] and KnotFold [Gong, 2024] is reported here for RNAs longer than 1,000 nt.

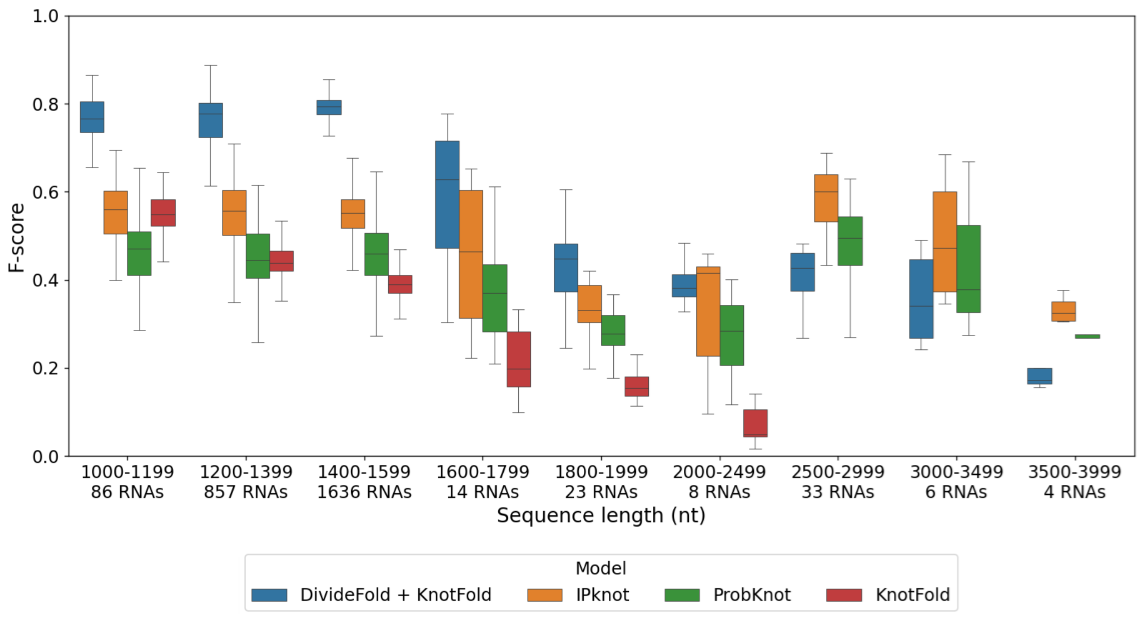

Figure 6.4: **Secondary structure prediction performance including pseudoknots by sequence length on the Test dataset.** The F-score is shown depending on the input sequence length for RNAs longer than 1,000 nt. The performance of DivideFold, IPknot [Sato, 2011; Sato, 2022], ProbKnot [Bellaousov, 2010] and KnotFold [Gong, 2024] is reported here. KnotFold is unable to handle RNAs longer than 2,500 nt.

6.3.3 Pseudoknot prediction results

Since the pseudoknots represent a minority of base pairs in the secondary structures of RNAs, the secondary structure prediction results excluding pseudoknots and those including pseudoknots are close to one another. To better assess the capability of DivideFold to accurately predict the presence of pseudoknots, we evaluate here the pseudoknot prediction alone, without examining the rest of the base pairs. As mentioned in the introduction of this chapter, to find the pseudoknots in the structure, we look for any two nested groups of base pairs where each base pair (i, j) in the first group and each base pair (k, l) in the second group are such that $i < k < j < l$. As such, two crossing stems constitute a single pseudoknot in our experiments, rather than a pseudoknot for every two crossing base pairs between the two stems, which would lead to an artificially high number of pseudoknots. Furthermore, we consider that two pseudoknots are similar if they share at least one common base pair in their first groups and one common base pair in their second groups. A predicted pseudoknot is considered a true positive if there is a similar pseudoknot in the real structure, and a false positive otherwise. Real pseudoknots with no similar pseudoknots in the predicted structure are false negatives. We use the precision, recall and F-score metrics when evaluating pseudoknot prediction.

We compare DivideFold to IPknot [Sato, 2011; Sato, 2022], ProbKnot [Bellaousov, 2010] and KnotFold [Gong, 2024] for sequences longer than 1,000 nt. KnotFold could only process RNAs shorter than 2,500 nt.

We show the mean pseudoknot prediction results in Table 6.2. IPknot and ProbKnot show poor results for pseudoknot prediction, finding less than 1% of pseudoknots. KnotFold performs much better and manages to find 46.4% of pseudoknots on average, although it also predicts many false positives, hence a low precision. In contrast, our approach allows to find up to 81.3% of pseudoknots on average and increases precision by a factor of 23 compared to KnotFold.

Table 6.2: **Pseudoknot prediction performance on the Test dataset.**

Model	**Recall**	**Precision**	**F-score**
DivideFold + KnotFold	**0.813**	**0.069**	**0.126**
KnotFold	0.464	0.003	0.006
IPknot	0.008	0.001	0.002
ProbKnot	0.002	0.001	0.001

The performance of DivideFold, IPknot [Sato, 2011; Sato, 2022], ProbKnot [Bellaousov, 2010] and KnotFold [Gong, 2024] is reported here for RNAs longer than 1,000 nt.

We now inspect the pseudoknot prediction performance depending on the RNA sequence length. We separate the sequences into groups according to their lengths and we show the results in Fig 6.5. DivideFold exhibits significantly superior performance compared to the other benchmarked methods for RNAs up to 1,800 nucleotides in length, and better performance until 3,000 nucleotides in length. IPknot shows better results for RNAs longer than 3,000 nucleotides, but the very limited number of sequences available for evaluation makes this less meaningful. As for the secondary structure prediction, the drop in DivideFold's performance for RNAs beyond 1,600 nt is probably caused by the lack of training data for RNAs longer than 1,600 nucleotides.

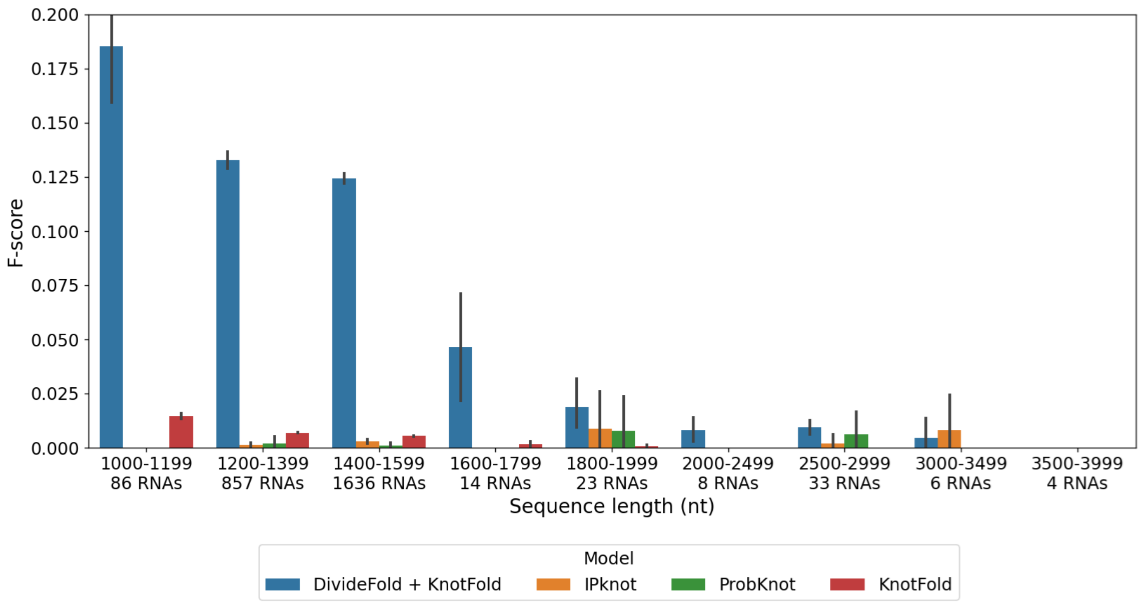

Figure 6.5: **Pseudoknot prediction performance by sequence length on the Test dataset.** The F-score is shown for pseudoknot prediction depending on the input sequence length for RNAs longer than 1,000 nt. Confidence intervals are shown in the black bars. The performance of DivideFold, IPknot [Sato, 2011; Sato, 2022], ProbKnot [Bellaousov, 2010] and KnotFold [Gong, 2024] is reported here. Missing bars mean that the performance is very close to zero and cannot be seen. KnotFold is unable to handle RNAs longer than 2,500 nt.

6.3.4 Time cost

We measure the computation time of DivideFold and compare it to that of IPknot [Sato, 2011; Sato, 2022], ProbKnot [Bellaousov, 2010] and KnotFold [Gong, 2024]. We show the results in Fig 6.6. DivideFold maintains a scalable computation time, taking less than 100 seconds for RNAs up to 4,000 nt. Notably, IPknot [Sato, 2011; Sato, 2022] is very fast in its new version [Sato, 2022].

6.3.5 Case study: the 16S ribosomal RNA

We use the 16S ribosomal RNA as a case study for predicting pseudoknots. We display its structure in Fig 6.7, which includes three pseudoknots shown in purple, and we highlight the cut points selected by our divide model in red dots.

We show the results for the secondary structure prediction including pseudoknots of the 16S ribosomal RNA in Table 6.3. In this experiment we were able to use pKiss [Theis, 2010; Janssen, 2015], even though it takes a long time to compute, since there is only one sequence to process here. DivideFold performs better than IPknot, ProbKnot, KnotFold and pKiss in terms or recall, precision and F-score. DivideFold is also faster than all other

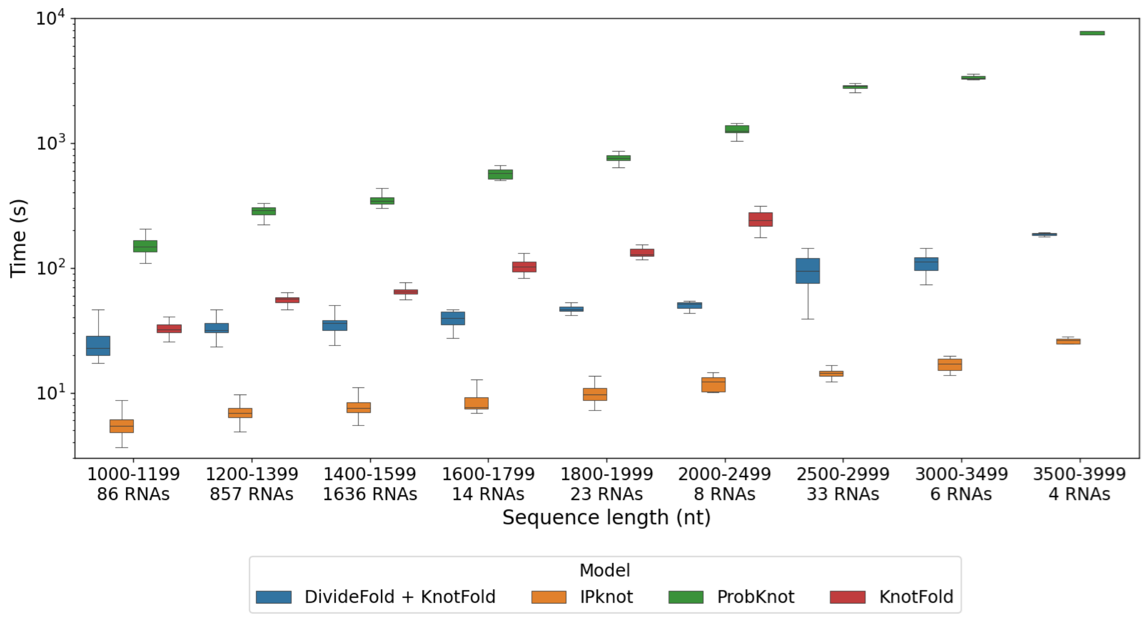

Figure 6.6: **Computation time by sequence length for tools that predict pseudoknots.** The time is displayed in a log scale depending on the input sequence length for DivideFold, IPknot [Sato, 2011; Sato, 2022], ProbKnot [Bellaousov, 2010] and KnotFold [Gong, 2024] for RNAs longer than 1,000 nt on the Test dataset. KnotFold is unable to handle RNAs longer than 2,500 nt.

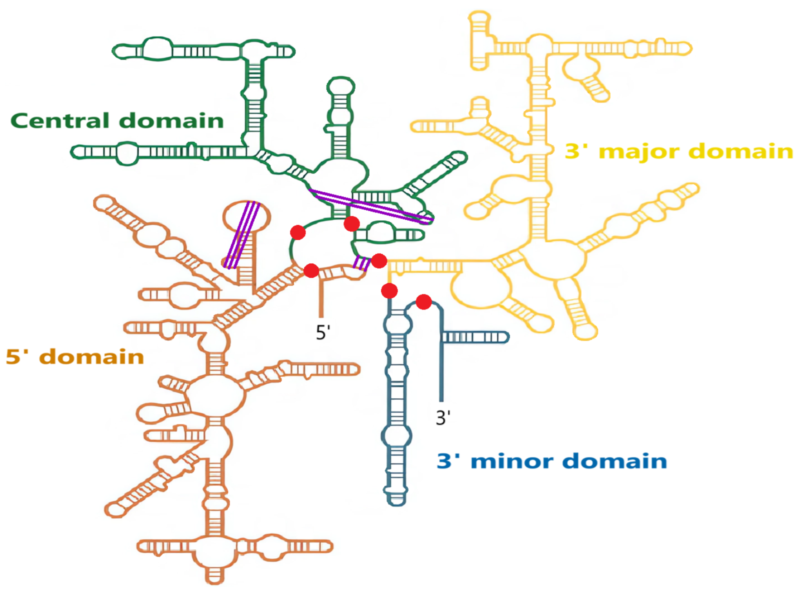

Figure 6.7: **Secondary structure of the 16S ribosomal RNA.** The final cut points chosen by our divide model are shown in red dots. The three pseudoknots are displayed in purple. Adapted from [Williamson, 2005].

benchmarked tools except IPknot.

Additionally, the pseudoknot prediction results for the 16S ribosomal RNA are shown in Table 6.4. DivideFold manages to find two pseudoknots out of three, while KnotFold

Table 6.3: **Results for secondary structure prediction including pseudoknots on the 16S ribosomal RNA.**

Model	Recall	Precision	F-score	Time
DivideFold + KnotFold	**0.770**	**0.848**	**0.807**	0:00:27
KnotFold	0.441	0.315	0.368	0:01:10
IPknot	0.443	0.528	0.482	**0:00:08**
ProbKnot	0.376	0.392	0.384	0:06:33
pKiss	0.277	0.269	0.273	2:37:09

The performance of DivideFold, IPknot [Sato, 2011; Sato, 2022], ProbKnot [Bellaousov, 2010], KnotFold [Gong, 2024] and pKiss [Theis, 2010; Janssen, 2015] is reported here, as well as the computation time.

only finds one, and IPknot, pKiss and ProbKnot find none. Furthermore, DivideFold produces only 14 false positive pseudoknots, while KnotFold generates a much higher number, with 367 false positives. IPknot, the fastest tool, fails to detect any pseudoknots but instead returns 11 false positives.

Table 6.4: **Results for pseudoknot prediction on the 16S ribosomal RNA.**

Model	Pseudoknots found	False positives
DivideFold + KnotFold	**2 / 3**	14
KnotFold	1 / 3	367
IPknot	0 / 3	11
ProbKnot	0 / 3	**4**
pKiss	0 / 3	17

The performance of DivideFold, IPknot [Sato, 2011; Sato, 2022], ProbKnot [Bellaousov, 2010], KnotFold [Gong, 2024] and pKiss [Theis, 2010; Janssen, 2015] is reported here. It is shown how many of the three pseudoknots are correctly found and how many false positive pseudoknots are incorrectly predicted.

We can see in Fig 6.7 that two out of the three pseudoknots (the one on the left and the one at the top right) in the structure are located within the same fragment and can be recovered after the partition. As a matter of fact, both are then successfully predicted at the structure prediction step. However, the third pseudoknot (at the bottom right) occurs between two different fragments (the pseudoknot takes place between the left-most part and an internal part, which are not combined together and yield to two separate fragments) and is irrecoverable.

6.3.6 Study of RNAs shorter than 1,000 nt

We evaluate here the secondary structure prediction performance including pseudoknots, and pseudoknot prediction performance, of DivideFold for RNAs shorter than 1,000 nt. We compare DivideFold to IPknot, ProbKnot, KnotFold, pKiss and UFold for sequences between 500 nt and 1,000 nt. Thus, the maximum fragment length cannot exceed 500 nt since it must remain lower than the sequence length to allow partition. Since it is shown in Fig 6.3 that higher values are better up to 1,000 nt, we choose a value of 500 nt for the maximum fragment length in this experiment.

The results depending on the RNA sequence length are shown in Fig 6.8 and Fig 6.9 for the structure prediction including pseudoknots and pseudoknot prediction respectively. UFold only accepts sequences shorter than 600 nt. Even though DivideFold perform reasonably well on the secondary structure prediction including pseudoknots, it is unable to beat KnotFold for RNAs between 500 nt and 1,000 nt. This confirms that DivideFold is better used for RNAs longer than 1,000 nt. Regarding pseudoknot prediction, DivideFold still yields a better F-score than other considered tools for RNAs between 500 nt and 1,000 nt.

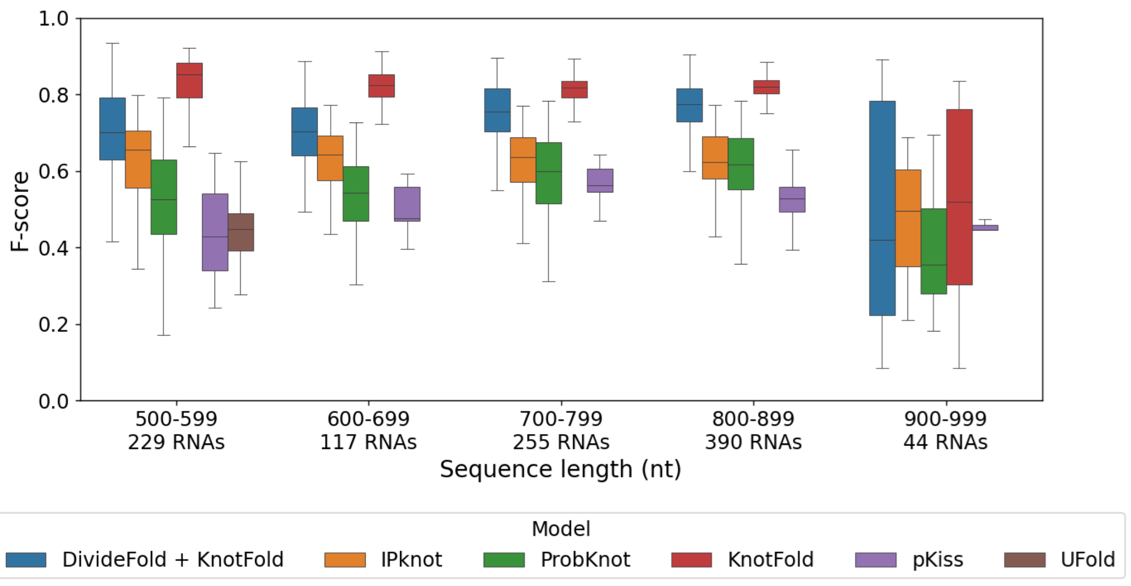

Figure 6.8: **Secondary structure prediction performance including pseudoknots by sequence length for RNAs between 500 nt and 1,000 nt.** The F-score is shown for secondary structure prediction including pseudoknots depending on the input sequence length. The performance of DivideFold, IPknot [Sato, 2011; Sato, 2022], ProbKnot [Bellaousov, 2010], KnotFold [Gong, 2024], pKiss [Theis, 2010; Janssen, 2015] and UFold [Fu, 2022] is reported here for RNAs between 500 and 1,000 nt on the Test dataset. UFold is unable to handle RNAs longer than 600 nt.

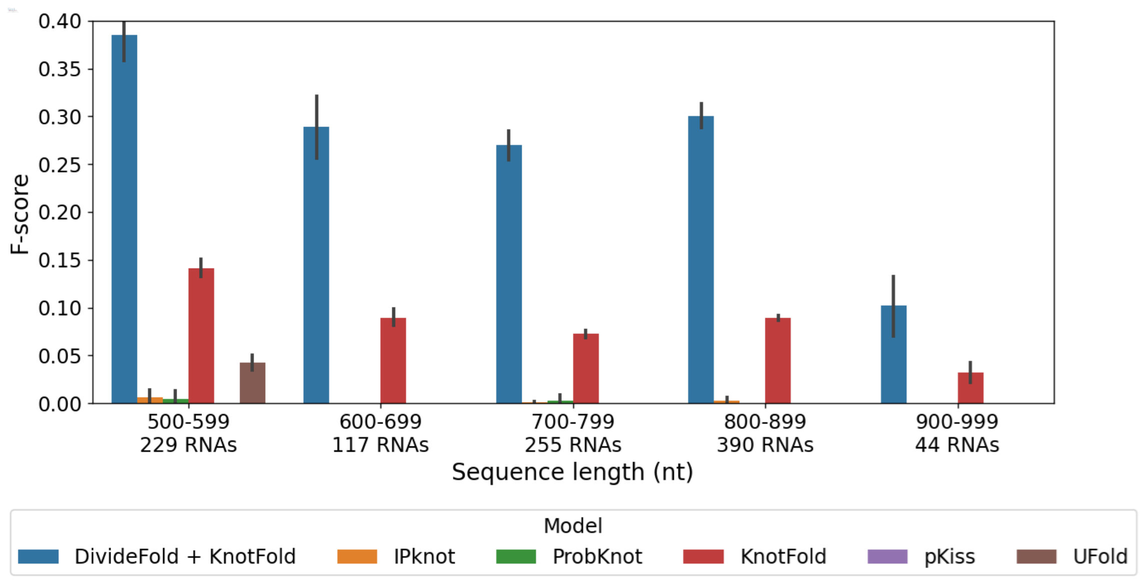

Figure 6.9: **Pseudoknot prediction performance by sequence length for RNAs between 500 nt and 1,000 nt.** The F-score is shown for pseudoknot prediction depending on the input sequence length. The performance of DivideFold, IPknot [Sato, 2011; Sato, 2022], ProbKnot [Bellaousov, 2010], KnotFold [Gong, 2024], pKiss [Theis, 2010; Janssen, 2015] and UFold [Fu, 2022] is reported here for RNAs between 500 and 1,000 nt on the Test dataset. UFold is unable to handle RNAs longer than 600 nt.

6.3.7 Generalization to unseen families

We tackle here the ability to generalize to families unseen in the training dataset for structure prediction including pseudoknots and for pseudoknot prediction. As in Chapter 5, we use the bpRNA-NF-15.0 dataset for this since it only contains RNA families that do not exist in the Train dataset. No sequences longer than 1,000 nt can be found in bpRNA-NF-15.0, which is not ideal for DivideFold, but we perform this analysis regardless. We evaluate DivideFold, IPknot, ProbKnot, KnotFold, pKiss and UFold on bpRNA-NF-15.0 for RNAs between 500 and 1,000 nt. For fair comparison to the other tools, we only display the results of UFold depending on the input sequence length, but not its mean results as UFold can only process RNAs shorter than 600 nt, meaning that we cannot evaluate it on all of bpRNA-NF-15.0. As in Section 6.3.6, we use a maximum fragment length of 500 nt for DivideFold to allow partition.

The mean structure prediction performance performance including pseudoknots on bpRNA-NF-15.0 are shown in Table 6.5. DivideFold does not perform as good as KnotFold and IPknot on bpRNA-NF-15.0, underlying that generalization to unseen families is still insufficient for RNAs shorter than 1,000 nt for structure prediction.

Considering pseudoknot prediction, the mean results on bpRNA-NF-15.0 are displayed in Table 6.6. IPknot, ProbKnot and pKiss show poor results for pseudoknot prediction. KnotFold is able to find 68.8% of the pseudoknots, but also predicts many false positives.

Table 6.5: **Secondary structure prediction performance including pseudoknots on bpRNA-NF-15.0.**

Model	Recall	Precision	F-score
DivideFold + KnotFold	0.429	0.421	0.424
KnotFold	**0.580**	0.382	0.460
IPknot	0.485	**0.455**	**0.464**
ProbKnot	0.493	0.337	0.399
pKiss	0.477	0.328	0.387

The performance of DivideFold, IPknot [Sato, 2011; Sato, 2022], ProbKnot [Bellaousov, 2010], KnotFold [Gong, 2024] and pKiss [Theis, 2010; Janssen, 2015] is reported here for RNAs between 500 and 1,000 nt.

On bpRNA-NF-15.0, our approach leads to a lower recall compared to KnotFold, but a higher precision and a higher F-score overall.

Table 6.6: **Pseudoknot prediction performance on bpRNA-NF-15.0.**

Model	Recall	Precision	F-score
DivideFold + KnotFold	0.606	**0.147**	**0.228**
KnotFold	**0.688**	0.070	0.126
IPknot	0.038	0.069	0.043
ProbKnot	0.007	0.021	0.010
pKiss	0.063	0.038	0.043

The performance of DivideFold, IPknot [Sato, 2011; Sato, 2022], ProbKnot [Bellaousov, 2010], KnotFold [Gong, 2024] and pKiss [Theis, 2010; Janssen, 2015] is reported here for RNAs between 500 and 1,000 nt.

We also inspect the pseudoknot prediction performance depending on the RNA sequence length on bpRNA-NF-15.0 and plot the results in Fig 6.10. As IPknot, ProbKnot and pKiss, UFold is only able to correctly find a minority of the pseudoknots. DivideFold maintains superior performance compared to the other benchmarked methods on bpRNA-NF-15.0, showing that generalization to unseen families is better handled for pseudoknot prediction.

We compare the performance on bpRNA-NF-15.0 to that measured in Section 6.3.6 on the Test dataset for the same sequence lengths. We show the results in Table 6.7 and Table 6.8 for structure prediction including pseudoknots and pseudoknot prediction respectively. Regarding secondary structure prediction, the performance loss on bpRNA-NF-15.0 for secondary structure prediction including pseudoknots is highest for DivideFold and KnotFold, but the mean performance also decreases significantly for IPknot, ProbKnot and pKiss, which are not trained methods, demonstrating that handling newly discovered sequences is a tough challenge in itself.

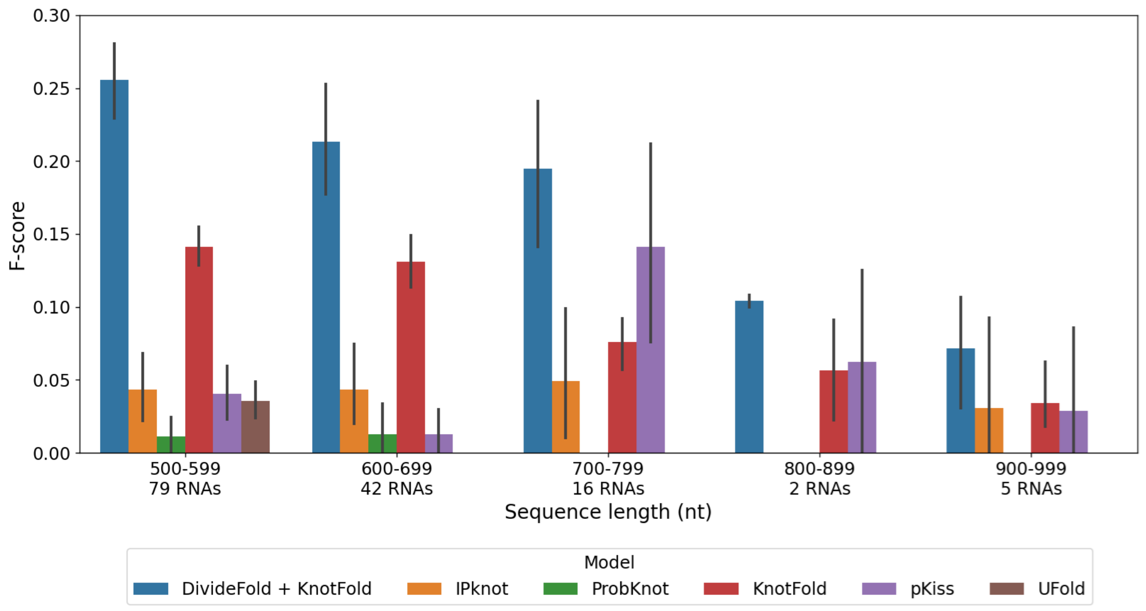

Figure 6.10: **Pseudoknot prediction performance by sequence length on bpRNA-NF-15.0.** The F-score is shown for pseudoknot prediction depending on the input sequence length for RNAs longer than 500 nt. Confidence intervals are shown in the black bars. The performance of DivideFold, IPknot [Sato, 2011; Sato, 2022], ProbKnot [Bellaousov, 2010], KnotFold [Gong, 2024], pKiss [Theis, 2010; Janssen, 2015] and UFold [Fu, 2022] is reported here. Missing bars mean that the performance is very close to zero and cannot be seen. UFold is unable to handle RNAs longer than 600 nt.

Table 6.7: **Mean F-score comparison for secondary structure prediction including pseudoknots between the Test and bpRNA-NF-15.0 datasets.**

Model	**F-score on Test**	**F-score on bpRNA-NF-15.0**	**F-score gap**
DivideFold + KnotFold	0.699	0.424	-0.275
KnotFold	**0.797**	0.460	-0.337
IPknot	0.616	**0.464**	-0.152
ProbKnot	0.567	0.399	-0.168
pKiss	0.493	0.387	-0.106

The mean F-score for secondary structure prediction including pseudoknots of DivideFold, IPknot [Sato, 2011; Sato, 2022], ProbKnot [Bellaousov, 2010], KnotFold [Gong, 2024] and pKiss [Theis, 2010; Janssen, 2015] is reported here on the Test and bpRNA-NF-15.0 datasets, for RNAs between 500 and 1,000 nt.

Considering pseudoknot prediction, the performance of DivideFold decreases on bpRNA-NF-15.0 compared to the Test dataset, but does not fall too far behind. DivideFold still yields the highest mean F-score in comparison to the other benchmarked tools, showing that the prediction of pseudoknots is reasonably well generalized over unseen families.

Table 6.8: **Mean F-score comparison for pseudoknot prediction between the Test and bpRNA-NF-15.0 datasets.**

Model	**F-score on Test**	**F-score on bpRNA-NF-15.0**	**F-score gap**
DivideFold + KnotFold	**0.302**	**0.228**	-0.074
KnotFold	0.094	0.126	0.032
IPknot	0.003	0.043	0.040
ProbKnot	0.002	0.010	0.008
pKiss	0.000	0.043	0.043

The mean F-score for pseudoknot prediction of DivideFold, IPknot [Sato, 2011; Sato, 2022], ProbKnot [Bellaousov, 2010], KnotFold [Gong, 2024] and pKiss [Theis, 2010; Janssen, 2015] is reported here on the Test and bpRNA-NF-15.0 datasets, for RNAs between 500 and 1,000 nt.

6.4 Discussion

We show that DivideFold can be successfully extended to the prediction of pseudoknots for long RNAs. DivideFold still runs in linear time and has constant memory usage when predicting pseudoknots, allowing it to process long RNAs, which is usually an important challenge for pseudoknot prediction.

We demonstrate in our benchmark that DivideFold surpasses IPknot, ProbKnot and KnotFold for the secondary structure prediction including pseudoknots and the pseudoknot prediction of RNAs longer than 1,000 nt. We also show that DivideFold outperforms UFold and pKiss within their respective length limitations. However, all the tools that we evaluated struggle to accurately predict pseudoknots in long RNAs beyond 2,500 nucleotides.

We also show that DivideFold is able to generalize fairly well to unseen RNA families for pseudoknot prediction. Indeed, the pseudoknot prediction performance of DivideFold does not decrease too much for RNAs between 500 and 1,000 nt between the Test and bpRNA-NF-15.0 datasets and remains stronger than that of the other benchmarked tools.

7 Data augmentation for RNA sequence and secondary structure data

7.1 Introduction

Data augmentation is often used in machine learning, using existing data samples to create new plausible ones to alleviate overfitting problems and help improve model generalization [Shorten, 2019]. Data augmentation is widely used for computer vision, natural language processing and time series analysis, in numerous domains including finance, cybersecurity, robotics, healthcare, and many others. In computer vision, common techniques include cropping, flipping, noise injection, or applying a rotation or translation to the images. The colors can also be altered, for example modifying the brightness, contrast or saturation. This helps the model learn important properties such as positional invariance and recognizing an object at various angles or lighting conditions. For text data, data augmentation methods include back translation (translating a sentence to another language, then back to the original language), synonym replacement, word insertion, word deletion, or shuffling the sentences in a text. Data augmentation can also be used for time series data. In acoustic signals, it is possible to change the speed of the sample, to add noise, or to mask certain frequencies. In biological signals, a common approach is to generate synthetic signals by rearranging components of real data. When dealing with imbalanced datasets, over-sampling methods focus on creating new samples in the minority class. One such method is SMOTE [Chawla, 2002] (Synthetic Minority Over-sampling TEchnique), which randomly selects a sample in the minority class and generates new synthetic samples between the sample and its closest neighbors. Another approach to data augmentation involves generating new samples that follow the probability distribution of the original data. In recent years, generative models, and in particular Generative Adversarial Networks (GANs) [Goodfellow, 2014], have been increasingly used across a variety of fields for generating synthetic data. GANs take advantage of a competitive training process between a generator and a discriminator to produce realistic data. Variation AutoEncoders (VAEs) [Kingma, 2022] and diffusion-based models [Sohl-Dickstein, 2015] are other types of generative models that have been successfully used for data augmentation.

When dealing with nucleic acid sequences, typical augmentation techniques involve mutation, insertion, deletion, inversion, translocation (swapping the positions of two segments in the sequence) and reverse complementing to include both strands [Lee, 2023]. A representation of these techniques is given in Fig 7.1. These augmentations are typically applied to RNA sequence data in classification tasks, for example predicting the class of a sequence, the functional consequences of non-coding mutations, or the presence of specific sites of interest such as chromatin accessibility sites [Lee, 2023]. Generative models can also be employed to produce entirely new RNA sequences [Zhao, 2024; Shahgir, 2024].

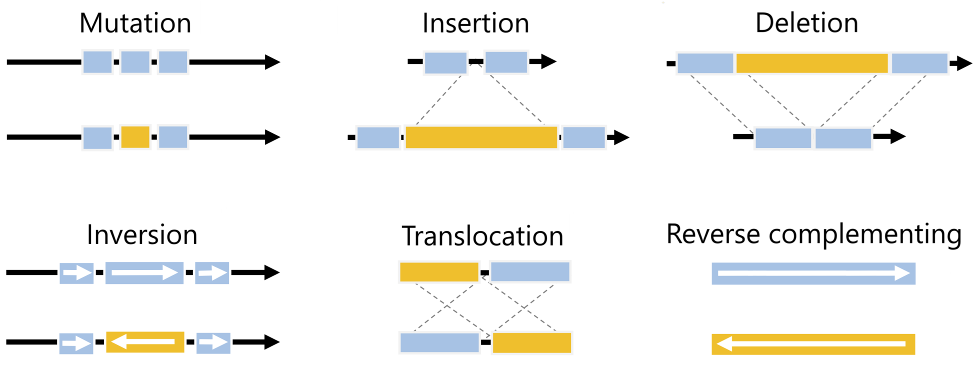

Figure 7.1: **Main augmentation techniques for RNA sequence data.** Typical augmentation techniques for RNA sequence data include mutation, insertion, deletion, inversion, translocation and reverse complementing. Adapted from [Lee, 2023].

However, data augmentation is rarely used for RNA secondary structure data. Indeed, a difficulty lies in maintaining consistency between the RNA sequence and its corresponding structure, as they are closely linked. Any modification to the sequence must be appropriately reflected in the structure to preserve biological coherence. To the best of our knowledge, data augmentation is often overlooked in RNA secondary structure prediction, and when it is used, it typically involves only random mutations applied to the sequence [Fu, 2022; Wang, 2025].

In this chapter, we show that other data augmentations techniques can be used for RNA sequence and secondary structure data, similarly to the augmentations used for RNA sequence data, and allow to enhance the prediction of RNA secondary structure. We disregard generative models since we observe that existing models are trained on short sequences and are unable to generate long meaningful sequences [Zhao, 2024; Shahgir, 2024].

7.2 Method

We consider existing data augmentation techniques for RNA sequence data, and generalize them to RNA sequence and secondary structure data. We detail here our implementation for (i) mutation, (ii) insertion, (iii) deletion, (iv) inversion, (v) translocation and (vi) reverse complementing, which we illustrate in Fig 7.2. For all these data augmentations, the sequence and its secondary remain consistent, and no new pseudoknots are created during the process. Our combined augmentation method randomly selects and applies a subset of these six augmentation techniques to further increase sequence diversity.

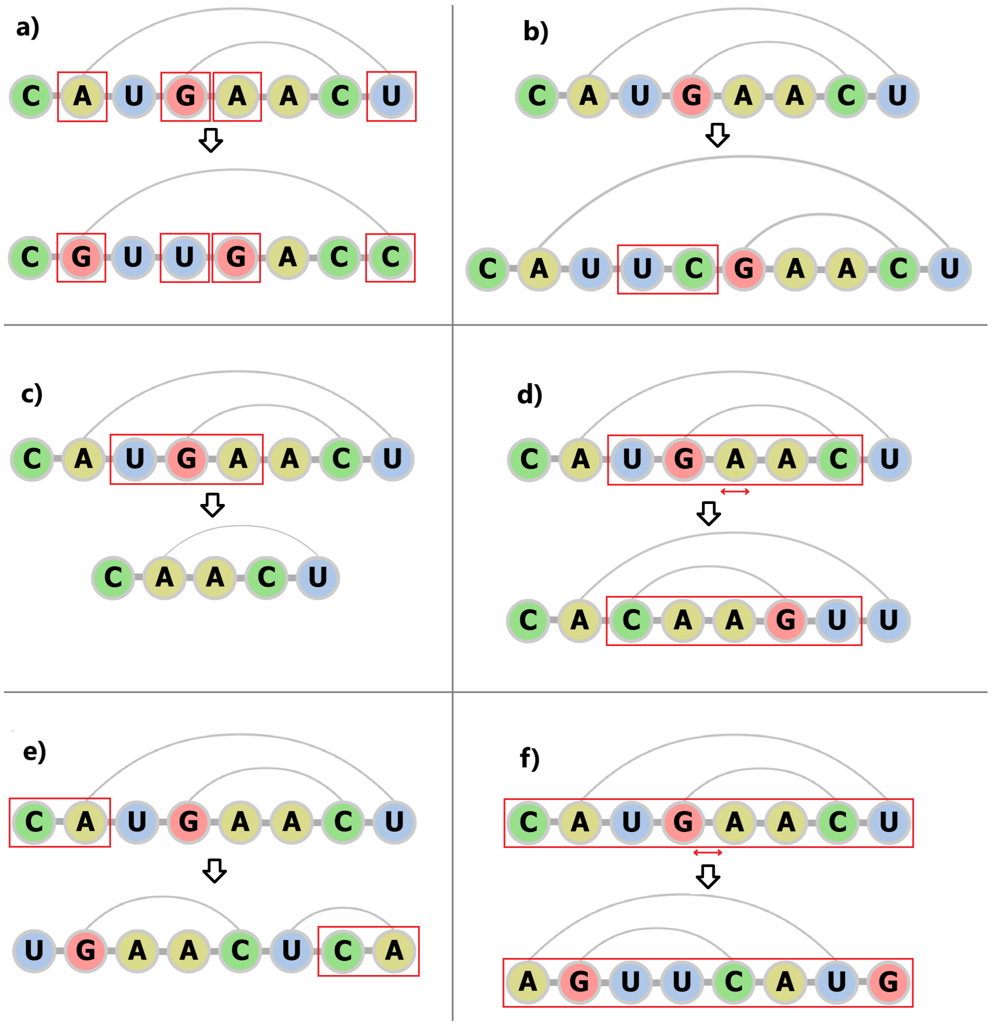

Figure 7.2: **Our combined augmentation method, illustrated on the sequence CAUGAACU.** We implement **a)** mutation, **b)** insertion, **c)** deletion, **d)** inversion, **e)** translocation and **f)** reverse complementing. Our combined augmentation method randomly selects and applies a subset of these six augmentation techniques.

Mutation is the most used form of data augmentation for RNA sequence, and has been occasionally used for RNA secondary structure prediction as well by UFold [Fu, 2022] and RNADiffFold [Wang, 2025]. It simulates a mutation occurring naturally into the sequence. However, it is generally only applied to the sequence, randomly mutating certain nucleotides into other ones, which can result in the associated secondary structure to contain unrealistic base pairs. To tackle this issue, UFold and RNADiffFold use CONTRAfold [Do, 2006] to predict a secondary structure for the mutated sequence, and pick the result as the new ground truth for training. This introduces bias, as the training labels are found by a specific secondary structure prediction tool, rather than trusted comparative analysis methods based on numerous homologous sequences. Instead, we choose to randomly mutate 5% of the nucleotides, and we adjust the secondary structure accordingly. Specifically, base pairs that become non-canonical because of nucleotide mutations are removed.

Insertion mimics the adding of a bulge loop to the sequence. We consider here insertion for RNA sequence and secondary structure data by randomly inserting a segment of unpaired nucleotides into the RNA sequence. Formally, a segment of length between 1 and 20 bases is randomly sampled and inserted at a random position into the sequence. An empty substructure of the same length, signaling a bulge loop, is inserted at the same position into the secondary structure.

Similarly, we consider deletion by removing a segment of length between 1 and 20 bases from the sequence at a random position. The same segment is removed from the secondary structure, and base pairs connecting a deleted nucleotide to another nucleotide are deleted from the secondary structure to ensure coherence.

We consider inversion as an augmentation technique for RNA sequence and secondary structure data as well. To that end, we choose to reverse a segment of length between 2 and 20 bases in the sequence at a random position. Since a reversed segment will conserve the base pairs occurring within itself, we keep the base pairs internal to the segment. However, we remove the base pairs connecting a nucleotide inside the reversed segment and a nucleotide outside of it to avoid creating pseudoknots.

We also apply translocation to RNA sequence and secondary structure data. Translocation is done by moving a segment from the $3'$ end of the sequence to the $5'$ end, or vice versa. Formally, we randomly pick a segment of length between 1 and 20 bases, at either the $3'$ or $5'$ end, move it to the other end, and adjust the secondary structure consequently.

Reverse complementing consists in switching an RNA sequence to its other strand. The sequence is reversed, and every nucleotide is changed to its complementary base (A to U and vice versa, G to C and vice versa). The secondary structure is once again adjusted. Note that this is not the same as inversion since inversion only reverses a part of the sequence, rather than all of it, and does not switch to complement bases.

Finally, for our combined augmentation method, we integrate all of these augmentations techniques. Specifically, we randomly apply two of the six augmentations described above in random order to increase the diversity of augmented sequences. This data augmentation is done online at the data loader step, which makes it more flexible and easy to integrate into the training process compared to offline augmentation, where the augmented training dataset had to be generated first. Using different random combinations of the previously mentioned augmentation techniques allows to generate many other sequences and secondary structures while preserving their biological coherence, thereby increasing the variety of training samples.

7.3 Results

7.3.1 Secondary structure prediction excluding pseudoknots

We evaluate here DivideFold for different data augmentation strategies, for secondary structure prediction excluding pseudoknots on the Test dataset. We compare the performance of DivideFold (i) without data augmentation, (ii) using only a mutation augmentation and (iii) using the combined augmentation comprising mutation, insertion, deletion, inversion, translocation and reverse complementing.

We show the results in Table 7.1. It can be seen that using a mutation augmentation, as is sometimes done for secondary structure prediction, indeed allows to obtain a better performance in regard to the recall, precision and F-score. However, using the combined augmentation increases even further all metrics, demonstrating its efficacy in creating more diverse training data to avoid overfitting.

Table 7.1: **Effect of data augmentation on DivideFold for secondary structure prediction excluding pseudoknots on the Test dataset.**

Augmentation	**Recall**	**Precision**	**F-score**
No augmentation	0.710	0.801	0.751
Mutation	0.719	0.803	0.757
Combined augmentation	**0.737**	**0.813**	**0.772**

The performance of DivideFold is reported here for RNAs longer than 1,000 nt on the Test dataset.

7.3.2 Secondary structure prediction including pseudoknots

We now evaluate the effect of data augmentation strategies on DivideFold for secondary structure prediction including pseudoknots on the Test dataset. As in the previous section, we compare the performance of DivideFold (i) without data augmentation, (ii) using only a mutation augmentation and (iii) using the combined augmentation.

We show the results in Table 7.2. As before, using the combined augmentation is more effective than performing only mutations and leads to a higher recall, precision and F-score on average.

Table 7.2: **Effect of data augmentation on DivideFold for secondary structure prediction including pseudoknots on the Test dataset.**

Augmentation	**Recall**	**Precision**	**F-score**
No augmentation	0.744	0.738	0.740
Mutation	0.755	0.743	0.748
Combined augmentation	**0.777**	**0.750**	**0.762**

The performance of DivideFold is reported here for RNAs longer than 1,000 nt on the Test dataset.

7.3.3 Pseudoknot prediction

Similarly, we evaluate DivideFold for different data augmentation strategies for pseudoknot prediction on the Test dataset. As before, we compare the performance of DivideFold (i) without data augmentation, (ii) using only a mutation augmentation and (iii) using the combined augmentation.

The results are shown in Table 7.3. As for the structure prediction including pseudoknots, we can see that using a mutation augmentation results in an increase of the recall, precision and F-score of DivideFold, but the gap is even larger when using the combined augmentation. This proves the effectiveness of the combined augmentation for pseudoknot prediction as well.

Table 7.3: **Effect of data augmentation on DivideFold for pseudoknot prediction on the Test dataset.**

Augmentation	**Recall**	**Precision**	**F-score**
No augmentation	0.662	0.062	0.112
Mutation	0.725	0.067	0.121
Combined augmentation	**0.813**	**0.069**	**0.126**

The performance of DivideFold is reported here for RNAs longer than 1,000 nt on the Test dataset.

7.3.4 Generalization to unseen families

Lastly, we perform the same analysis on bpRNA-NF-15.0 to study if the combined data augmentation helps improve generalization to unseen RNA families. We compare the performance of DivideFold (i) without data augmentation, (ii) using only a mutation augmentation and (iii) using the combined augmentation. As in Sections 5.3.6 and 6.3.7, we use a maximum fragment length of 500 nucleotides for DivideFold on bpRNA-NF-15.0 to allow partition as sequences can be as short as 500 nucleotides in length, causing any higher value for the maximum fragment length to prevent the partition of such sequences.

The results are shown in Table 7.4, Table 7.5 and Table 7.6 for secondary structure prediction excluding pseudoknots, secondary structure prediction including pseudoknots and pseudoknot prediction respectively. In the context of secondary structure prediction excluding pseudoknots, data augmentation using only mutations does not improve performance. In comparison, the combined augmentation strategy slightly enhances DivideFold's recall. While it can be seen that the effect of data augmentation on the generalization to unseen RNA families remains limited for secondary structure prediction excluding pseudoknots, it still enables to marginally improve performance.

Table 7.4: **Effect of data augmentation on DivideFold for secondary structure prediction excluding pseudoknots on bpRNA-NF-15.0.**

Augmentation	**Recall**	**Precision**	**F-score**
No augmentation	0.384	0.462	0.417
Mutation	0.384	**0.463**	0.417
Combined augmentation	**0.393**	0.462	**0.422**

The performance of DivideFold is reported here for RNAs longer than 500 nt on bpRNA-NF-15.0.

Considering secondary structure prediction including pseudoknots, the mutation-only augmentation strategy does not improve performance, indicating that relying solely on mutations is insufficient to help generalization to unseen RNA families. In contrast, the combined augmentation strategy enhances DivideFold's performance on RNA families not present in the training set, demonstrating its effectiveness.

Table 7.5: **Effect of data augmentation on DivideFold for secondary structure prediction including pseudoknots on bpRNA-NF-15.0.**

Augmentation	**Recall**	**Precision**	**F-score**
No augmentation	0.415	0.411	0.411
Mutation	0.413	0.413	0.411
Combined augmentation	**0.429**	**0.421**	**0.424**

The performance of DivideFold is reported here for RNAs longer than 500 nt on bpRNA-NF-15.0.

Lastly, for pseudoknot prediction, the traditional augmentation strategy using only mutations does not improve performance on unseen RNA families either. In fact, it even decreases DivideFold's performance. This suggests that using only mutations is not enough to create a more diverse and relevant training dataset, and may even introduce sequences that are less biologically meaningful for capturing pseudoknots. Conversely, the combined augmentation strategy significantly improves generalization to unseen RNA families for pseudoknot prediction.

Table 7.6: **Effect of data augmentation on DivideFold for pseudoknot prediction on bpRNA-NF-15.0.**

Augmentation	**Recall**	**Precision**	**F-score**
No augmentation	0.521	0.124	0.193
Mutation	0.484	0.114	0.177
Combined augmentation	**0.606**	**0.147**	**0.228**

The performance of DivideFold is reported here for RNAs longer than 500 nt on bpRNA-NF-15.0.

7.4 Discussion

We present a data augmentation strategy for RNA sequence and secondary structure data that combines not only mutation, but also insertion, deletion, inversion, translocation and reverse complementing. We leverage the augmentation techniques commonly used for RNA sequence data and generalize them to data containing both the RNA sequence and its secondary structure, to provide greater variety in training data for secondary structure prediction and pseudoknot prediction. We design this combined augmentation to be applied online at the data loader step, for any task involving RNA sequence and secondary structure data. This enables the combined augmentation to be seamlessly integrated into the training procedure, making it easy to use. In contrast, offline augmentation requires generating the augmented dataset in advance.

We show that while mutation-based data augmentation enhances the performance of DivideFold, applying combined augmentation further improves its results for secondary structure excluding pseudoknots, secondary structure prediction including pseudoknots, and pseudoknot prediction for RNAs longer than 1,000 nt. This demonstrates that combined augmentation provides greater diversity in training data compared to traditional augmentation strategies based solely on mutation, enabling the model to generalize better to unseen samples and reducing overfitting. We also show that the combined data augmentation improves generalization to RNA families unseen in the training dataset for all three prediction tasks. In contrast, the traditional mutation-based strategy has no effect on the generalization to unseen families for secondary structure prediction, excluding or including pseudoknots, and may even be detrimental to pseudoknot prediction.

8

Conclusion and perspectives

8.1 Conclusion

In this thesis, we have explored novel approaches to tackle the complex problem of predicting the secondary structure of long RNA sequences, which we define in this thesis as those exceeding 1,000 nucleotides in length as shorter RNAs have already been extensively studied. This question is of great importance because understanding the structural characteristics of RNAs plays a key role in identifying their functional roles, and many long RNAs are not yet well characterized. These insights can ultimately lead to the discovery of new biomarkers and therapeutic targets, paving the way for the development of more personalized and effective medical treatments. However, predicting the secondary structure of long RNAs remains a challenging task, primarily due to the high computational cost involved. This is even more difficult to deal with when pseudoknots are taken into account, as these intricate structural motifs further increase the time complexity of prediction algorithms. Therefore, the development of fast and accurate prediction tools is essential. To address this issue, this thesis presents several contributions.

We develop DivideFold, a divide-and-conquer partition method based on deep learning designed to predict the secondary structures of long RNA sequences. DivideFold addresses the computational challenges posed by lengthy RNAs by partitioning them into shorter, more manageable fragments. The workflow of DivideFold consists of two main steps, partitioning the input sequence and predicting the secondary structure of each fragment. In the partition step, using only the RNA sequence information, DivideFold recursively applies a divide model based on deep learning, which selects strategic cut points to split the sequence into shorter segments. To predict these cut points, the sequence information is first represented as a concatenation of one-hot embedding and module insertion. This representation is then processed by a 1D CNN encoder, followed by a pointwise fully connected regression head that estimates the probability of a cut at each nucleotide position. A peak detection algorithm is then applied to these predicted probabilities to identify the final cut points. This process is repeated recursively until all resulting fragments are shorter than a chosen threshold, which we set to 1,000 nucleotides. Each of these shorter fragments is then passed to an existing RNA secondary structure prediction tool designed for short sequences. Finally, the individual fragment predictions are reassembled into a full secondary structure by aligning them back to their original positions in the sequence, reversing the partition step. This approach enables the use of existing structure prediction tools that may perform well on small RNAs but struggle with long RNAs. Furthermore, DivideFold can be paired with any new secondary structure prediction tool and can readily incorporate future advances in the field. DivideFold is fast, running in linear time with respect to the sequence length, regardless of the structure prediction tool used on the fragments.

When predicting RNA secondary structure excluding pseudoknots, we perform a benchmark based on data from the bpRNA-1m and Rfam 15.0 datasets where we evaluate RNAfold [Hofacker, 1994; Lorenz, 2011] as it is a tool of reference for secondary structure prediction, LinearFold [Huang, 2019] since it is a faster and potentially stronger improvement to RNAfold, MXfold2 [Sato, 2021] as it is frequently cited among deep learning approaches, and KnotFold [Gong, 2024] where its predicted pseudoknots are removed, as it has shown very good results on non-pseudoknotted base pairs in addition to pseudoknotted base pairs. We demonstrate that DivideFold outperforms RNAfold, LinearFold, MXfold2 and KnotFold for RNA sequences longer than 1,000 nucleotides. We also assess the generalization capabilities of the evaluated methods to unseen RNA families. Even though publicly available datasets lack sequences longer than 1,000 nucleotides from newly discovered RNA families, we nonetheless perform this analysis for RNAs between 500 and 1,000 nucleotides. The results show a notable drop in performance across all tools, highlighting that predicting the structures of newly discovered sequences remains a significant challenge. We confirm in our experiments that DivideFold runs in linear time with respect to the sequence length. Furthermore, we observe that the partition step is significantly faster than the structure prediction step. As a result, the overall computation time is essentially due to the structure prediction tool applied to the fragments, not to the divide model itself. This highlights the potential for DivideFold to achieve even greater speed when paired with a fast structure prediction method. Additionally, DivideFold has constant memory usage, avoiding memory overflows. This work has led to the publication of a paper in the 2023 IEEE International Conference on Bioinformatics and Bioengineering (BIBE), *Prediction of secondary structure for long non-coding RNAs using a recursive cutting method based on deep learning* [Omnes, 2023].

We then extend DivideFold to the prediction of secondary structure including pseudoknots. Pseudoknots play a crucial role in RNA structure, offering important insights into their three-dimensional folding. Yet, accurately predicting pseudoknots remains a challenging task due to computational constraints, especially for long RNA sequences. To address this, we adapt DivideFold by using at the fragment level a secondary structure prediction tool that supports pseudoknot prediction, and by choosing a fragment size that is sufficiently large so that most pseudoknots lie in a single fragment and can be recovered after partition. This enables DivideFold to predict the secondary structure including pseudoknots of long RNA sequences. DivideFold also merges external parts into a single fragment during each iteration in the partition step, allowing distant pseudoknots to be found. DivideFold still runs in linear time and has constant memory usage with respect to the sequence length when predicting pseudoknots, making it capable of handling long RNA sequences. We evaluate IPknot [Sato, 2011; Sato, 2022], ProbKnot [Bellaousov, 2010], KnotFold [Gong, 2024], pKiss [Theis, 2010; Janssen, 2015] and UFold [Fu, 2022] since they are among the most commonly benchmarked methods in the literature for secondary structure prediction including pseudoknots. We show that DivideFold outperforms IPknot, ProbKnot and KnotFold in both secondary structure prediction including pseudoknots and pseudoknot prediction for RNAs exceeding 1,000 nucleotides. It also surpasses UFold and pKiss within the bounds of their respective length constraints. We further show that DivideFold generalizes well to new RNA families for pseudoknot prediction. Its performance remains relatively stable when predicting the pseudoknots of RNAs from unseen families between 500 and 1,000 nucleotides, and continues to outperform the other benchmarked tools. This research has been accepted for publication in PLOS ONE as *A divide-and-conquer approach based on deep learning for long RNA secondary structure prediction: focus on pseudoknots identification* [Omnes, 2025].

Lastly, we generalize RNA sequence data augmentation techniques to data containing both the sequence and secondary structure of RNAs. Indeed, various data augmentation methods are commonly applied to RNA sequence data to mitigate overfitting and enhance model generalization, including mutation, insertion, deletion, inversion, translocation, and reverse complementing. However, data augmentation for RNA secondary structure prediction is rarely explored, and it has only been applied in the form of random mutations to the sequence itself. When augmenting RNA sequence and secondary structure data, the nucleotide sequence and its corresponding secondary structure must remain coherent. Any modification must respect this relationship to ensure that both data types are aligned. We show that standard RNA sequence data augmentation techniques can be effectively extended to RNA sequence and secondary structure data. Our approach introduces a strategy that combines not only mutation, but also insertion, deletion, inversion, translocation, and reverse complementing, applied jointly to both the sequence and its secondary structure. Furthermore, we demonstrate that the combined data augmentation strategy improves secondary structure prediction excluding pseudoknots, secondary structure prediction including pseudoknots and pseudoknot prediction for long RNA sequences, surpassing usual data augmentation methods that are based solely on sequence mutations. This shows that the combined augmentation provides greater diversity in the training data compared to mutation-only strategies, allowing the model to generalize better to unseen samples and reducing overfitting. We also show that the combined data augmentation improves generalization to RNA families unseen in the training dataset for all three prediction tasks. By comparison, the traditional mutation-based strategy has no effect on the generalization to unseen families for secondary structure prediction, excluding or including pseudoknots, and may even be detrimental to pseudoknot prediction. We designed the combined augmentation strategy to be seamlessly integrated into the training pipeline, applied online. This ensures that it can be easily employed for any task involving RNA sequence and secondary structure data.

DivideFold, along with all the datasets used for this study, is accessible on the EvryRNA platform at https://evryrna.ibisc.univ-evry.fr/evryrna/dividefold/home.

8.2 Perspectives

Several avenues for improvement remain to enhance the work presented in this thesis. One major limitation is the restricted amount of available training data, which is critical for the effectiveness of deep learning methods. RNA secondary structure datasets are generally small, and particularly scarce in long RNA sequences. In the case of DivideFold, both secondary structure prediction and pseudoknot prediction remain challenging for RNAs longer than 2,500 nucleotides. The lack of training data for long RNAs is likely a key factor behind the model's reduced accuracy for such long sequences. Going forward, gathering more long RNA sequences and secondary structures will be essential not only to improve predictive performance but also to better assess the model's capabilities. Additionally, acquiring secondary structure data for long sequences from newly discovered RNA families is crucial. Without such data, it is currently impossible to properly evaluate how well models generalize to previously unseen families for long RNAs. This data is therefore highly needed, as generalization to unknown families is an essential property for any robust RNA structure prediction tool.

DivideFold could also benefit from incorporating additional data types beyond just the nucleotide sequence. Because it estimates the likelihood that a nucleotide is paired or unpaired, SHAPE data is highly relevant when studying RNA secondary structure. Thus, integrating additional data sources could significantly enhance DivideFold's cut point predictions. However, in the case of SHAPE, most data currently available is limited to RNA sequences shorter than 500 nucleotides, making it unusable for DivideFold considering its focus on long RNAs. If data for long RNA sequences were to become available, it could be easily integrated into DivideFold's architecture by concatenating it with the other input features, potentially leading to substantial performance gains.

Beyond data availability, improvements to the model architecture itself are also worth exploring. An enhancement would be the integration of user-defined constraints. In many cases, users possess partial knowledge of the RNA's secondary structure. For example, the user may know the location of some major stems in the sequence, as such long stems are fundamental to the structure. This information could be used in a post-processing step to eliminate any cut points predicted within regions known to form stems, ensuring the known structural elements are preserved. However, this approach would only filter cut points after they have been predicted and could still lead to inconsistencies between user constraints and the rest of the predicted structure. A more robust solution would be to incorporate user information directly into the cut point prediction process by encoding it as additional input features, similar to the approach proposed for SHAPE data. This enables the user information to actively guide the prediction and helps ensure that the predicted cut points remain coherent with one another. However, incorporating user information solely as input features does not guarantee that it will be preserved in the prediction, as was the case with post-processing filtering. Therefore, integrating user knowledge both as additional input features and as a post-processing step could enforce user-specified constraints while maintaining prediction coherence. Integrating reliable structural insights from users has the potential to meaningfully enhance DivideFold's performance for the partition step.

A further potential improvement of DivideFold lies in predicting multiple secondary structures rather than a single one. RNAs are often flexible and may adopt different conformations depending on their molecular environment. Generating multiple predictions could help uncover these alternative structures and offer insight into different potential functional roles. One straightforward approach would be to apply DivideFold multiple times, using a different secondary structure prediction tool on the fragments each time, or even varying tools across fragments. However, even using different prediction tools, the resulting structures would likely remain similar as the partition would remain unchanged, since the partition step largely determines the overall structural organization. Instead, a better alternative would be to generate multiple distinct partitions, each guiding differently the structure prediction downstream. This would require adapting the architecture of the divide model's deep neural network in such a way that it is non-deterministic. Instead of using a peak detection algorithm, this could be done by sampling a single cut point from the distribution of predicted cutting probabilities at each iteration of the divide model. This would allow the model to randomly select cut points across successive iterations, thereby generating a random partition. Running DivideFold multiple times with this approach would effectively enable the prediction of multiple structures.

Additionally, it would be beneficial to further study the impact of our proposed combined data augmentation for different models and other tasks involving RNA sequence and secondary structure data. This is necessary to better assess its efficiency and how well it can be generalized. A more in-depth analysis of the combined data augmentation for various combinations of its parameters would also allow us to potentially improve its efficiency.

In conclusion, while the tools developed in this thesis provide a solid foundation for predicting secondary structures and pseudoknots in long RNAs, several opportunities for improvement remain. These include addressing data limitations, integrating additional data types such as SHAPE, incorporating user guidance, and enabling the prediction of multiple structures. Progress in these areas will not only enhance DivideFold's accuracy and robustness but also deepen our biological understanding of RNA. Ultimately, this could contribute to the discovery of new biomarkers and therapeutic targets, and lead to the development of more personalized and effective treatments in precision medicine.

Bibliography

[Abramson, 2024] Josh Abramson, Jonas Adler, Jack Dunger, et al. “Accurate Structure Prediction of Biomolecular Interactions with AlphaFold 3”. *Nature* 630.8016 (2024), pp. 493–500 (cit. on p. 2).

[Akiyama, 2022] Manato Akiyama, Yasubumi Sakakibara, and Kengo Sato. “Direct Inference of Base-Pairing Probabilities with Neural Networks Improves Prediction of RNA Secondary Structures with Pseudoknots”. *Genes* 13.11 (2022), p. 2155 (cit. on p. 50).

[Akutsu, 2000] Tatsuya Akutsu. “Dynamic Programming Algorithms for RNA Secondary Structure Prediction with Pseudoknots”. *Discrete Applied Mathematics* 104.1 (2000), pp. 45–62 (cit. on p. 46).

[Andronescu, 2008] Mirela Andronescu, Vera Bereg, Holger H. Hoos, et al. “RNA STRAND: The RNA Secondary Structure and Statistical Analysis Database”. *BMC bioinformatics* 9 (2008), p. 340 (cit. on p. 52).

[Andronescu, 2010] Mirela S. Andronescu, Cristina Pop, and Anne E. Condon. “Improved Free Energy Parameters for RNA Pseudoknotted Secondary Structure Prediction”. *RNA (New York, N.Y.)* 16.1 (2010), pp. 26–42 (cit. on p. 34).

[Assmann, 2023] Sarah M Assmann, Hong-Li Chou, and Philip C Bevilacqua. “Rock, Scissors, Paper: How RNA Structure Informs Function”. *The Plant Cell* 35.6 (2023), pp. 1671–1707 (cit. on pp. 1, 9).

[Aznaourova, 2020] Marina Aznaourova, Nils Schmerer, Bernd Schmeck, et al. “Disease-Causing Mutations and Rearrangements in Long Non-coding RNA Gene Loci”. *Frontiers in Genetics* 11 (2020) (cit. on p. 1).

[Balakin, 1996] Andrey G Balakin, Laurie Smith, and Maurille J Fournier. “The RNA World of the Nucleolus: Two Major Families of Small RNAs Defined by Different Box Elements with Related Functions”. *Cell* 86.5 (1996), pp. 823–834 (cit. on p. 7).

[Bartel, 2009] David P. Bartel. “MicroRNAs: Target Recognition and Regulatory Functions”. *Cell* 136.2 (2009), pp. 215–233 (cit. on p. 7).

[Becquey, 2020] Louis Becquey, Eric Angel, and Fariza Tahi. “BiORSEO: A Bi-Objective Method to Predict RNA Secondary Structures with Pseudoknots Using RNA 3D Modules”. *Bioinformatics* 36.8 (2020), pp. 2451–2457 (cit. on pp. 13, 49, 94).

[Bellaousov, 2010] Stanislav Bellaousov and David H. Mathews. “ProbKnot: Fast Prediction of RNA Secondary Structure Including Pseudoknots”. *RNA* 16.10 (2010), pp. 1870–1880 (cit. on pp. 48, 91, 92, 94–104, 115).

[Berman, 2000] Helen M. Berman, John Westbrook, Zukang Feng, et al. “The Protein Data Bank”. *Nucleic Acids Research* 28.1 (2000), pp. 235–242 (cit. on p. 51).

[Bernard, 2025] Clément Bernard, Guillaume Postic, Sahar Ghannay, et al. “RNA-TorsionBERT: Leveraging Language Models for RNA 3D Torsion Angles Prediction”. *Bioinformatics* 41.1 (2025), btaf004 (cit. on p. 58).

[Bernhart, 2008] Stephan H. Bernhart, Ivo L. Hofacker, Sebastian Will, et al. “RNAalifold: Improved Consensus Structure Prediction for RNA Alignments”. *BMC Bioinformatics* 9.1 (2008), p. 474 (cit. on p. 45).

[Bindewald, 2010] Eckart Bindewald, Tanner Kluth, and Bruce A. Shapiro. “CyloFold: Secondary Structure Prediction Including Pseudoknots”. *Nucleic Acids Research* 38.Web Server issue (2010), W368–W372 (cit. on p. 48).

[Biswas, 2004] Preetha Biswas, Xi Jiang, Annmarie L. Pacchia, et al. “The Human Immunodeficiency Virus Type 1 Ribosomal Frameshifting Site Is an Invariant Sequence Determinant and an Important Target for Antiviral Therapy”. *Journal of Virology* 78.4 (2004), pp. 2082–2087 (cit. on p. 11).

[Black, 2003] Douglas L. Black. “Mechanisms of Alternative Pre-Messenger RNA Splicing”. *Annual Review of Biochemistry* 72 (2003), pp. 291–336 (cit. on p. 6).

[Bose, 2024] Ritwika Bose, Irfana Saleem, and Anthony M. Mustoe. “Causes, Functions, and Therapeutic Possibilities of RNA Secondary Structure Ensembles and Alternative States”. *Cell Chemical Biology* 31.1 (2024), pp. 17–35 (cit. on pp. 1, 9).

[Boureau, 2010] Y-Lan Boureau, Francis Bach, Yann LeCun, et al. “Learning Mid-Level Features for Recognition”. *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2010, pp. 2559–2566 (cit. on p. 24).

[Breaker, 2012] Ronald R. Breaker. “Riboswitches and the RNA World”. *Cold Spring Harbor Perspectives in Biology* 4.2 (2012), a003566 (cit. on p. 7).

[Brierley, 1989] I. Brierley, P. Digard, and S. C. Inglis. “Characterization of an Efficient Coronavirus Ribosomal Frameshifting Signal: Requirement for an RNA Pseudoknot”. *Cell* 57.4 (1989), pp. 537–547 (cit. on p. 11).

[Brown, 1994] J. W. Brown, E. S. Haas, D. G. Gilbert, et al. “The Ribonuclease P Database”. *Nucleic Acids Research* 22.17 (1994), pp. 3660–3662 (cit. on p. 51).

[Budnik, 2024] Michał Budnik, Jakub Wawrzyniak, Łukasz Grala, et al. “Deep Dive into RNA: A Systematic Literature Review on RNA Structure Prediction Using Machine Learning Methods”. *Artificial Intelligence Review* 57.9 (2024), p. 254 (cit. on p. 12).

[Burnett, 2012] John C. Burnett and John J. Rossi. “RNA-based Therapeutics: Current Progress and Future Prospects”. *Chemistry & Biology* 19.1 (2012), pp. 60–71 (cit. on p. 2).

[Cai, 2003] Liming Cai, Russell L. Malmberg, and Yunzhou Wu. “Stochastic Modeling of RNA Pseudoknotted Structures: A Grammatical Approach”. *Bioinformatics (Oxford, England)* 19 Suppl 1 (2003), pp. i66–73 (cit. on p. 47).

[Calin, 2006] George A. Calin and Carlo M. Croce. “MicroRNA Signatures in Human Cancers”. *Nature Reviews. Cancer* 6.11 (2006), pp. 857–866 (cit. on pp. 1, 2).

[Cannone, 2002] Jamie J. Cannone, Sankar Subramanian, Murray N. Schnare, et al. “The Comparative RNA Web (CRW) Site: An Online Database of Comparative Sequence and Structure Information for Ribosomal, Intron, and Other RNAs”. *BMC Bioinformatics* 3.1 (2002), p. 2 (cit. on pp. 38, 52, 67).

[Cao, 2005] Song Cao and Shi-Jie Chen. “Predicting RNA Folding Thermodynamics with a Reduced Chain Representation Model”. *RNA* 11.12 (2005), pp. 1884–1897 (cit. on p. 47).

[Cao, 2009] Song Cao and Shi-Jie Chen. “Predicting Structures and Stabilities for H-type Pseudoknots with Interhelix Loops”. *RNA (New York, N.Y.)* 15.4 (2009), pp. 696–706 (cit. on p. 47).

[Castanotto, 2009] Daniela Castanotto and John J. Rossi. “The Promises and Pitfalls of RNA-interference-based Therapeutics”. *Nature* 457.7228 (2009), pp. 426–433 (cit. on p. 2).

[Cauchy, 1847] A. Cauchy. “Methode Generale Pour La Resolution Des Systemes d’equations Simultanees”. *C.R. Acad. Sci. Paris* 25 (1847), pp. 536–538 (cit. on p. 16).

[Cech, 1986] T. R. Cech and B. L. Bass. “Biological Catalysis by RNA”. *Annual Review of Biochemistry* 55 (1986), pp. 599–629 (cit. on p. 5).

[Cech, 2014] Thomas R. Cech and Joan A. Steitz. “The Noncoding RNA Revolution-Trashing Old Rules to Forge New Ones”. *Cell* 157.1 (2014), pp. 77–94 (cit. on pp. 1, 5, 7, 9).

[Chaturvedi, 2025] Mayank Chaturvedi, Mahmood A. Rashid, and Kuldip K. Paliwal. “RNA Structure Prediction Using Deep Learning — A Comprehensive Review”. *Computers in Biology and Medicine* 188 (2025), p. 109845 (cit. on p. 62).

[Chawla, 2002] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, et al. “SMOTE: Synthetic Minority over-Sampling Technique”. *J. Artif. Int. Res.* 16.1 (2002), pp. 321–357 (cit. on p. 105).

[Chen, 2023] Chun-Chi Chen and Yi-Ming Chan. “REDfold: Accurate RNA Secondary Structure Prediction Using Residual Encoder-Decoder Network”. *BMC Bioinformatics* 24.1 (2023), p. 122 (cit. on p. 50).

[Chen, 2004] Jiunn-Liang Chen and Carol W. Greider. “Telomerase RNA Structure and Function: Implications for Dyskeratosis Congenita”. *Trends in Biochemical Sciences* 29.4 (2004), pp. 183–192 (cit. on p. 11).

[Chen, 2009] Ho-Lin Chen, Anne Condon, and Hosna Jabbari. “An O(n(5)) Algorithm for MFE Prediction of Kissing Hairpins and 4-Chains in Nucleic Acids”. *Journal of Computational Biology: A Journal of Computational Molecular Cell Biology* 16.6 (2009), pp. 803–815 (cit. on pp. 48, 49).

[Chen, 2008] Xiang Chen, Si-Min He, Dongbo Bu, et al. “FlexStem: Improving Predictions of RNA Secondary Structures with Pseudoknots by Reducing the Search Space”. *Bioinformatics (Oxford, England)* 24.18 (2008), pp. 1994–2001 (cit. on p. 47).

[Chen, 2020] Xinshi Chen, Yu Li, Ramzan Umarov, et al. *RNA Secondary Structure Prediction By Learning Unrolled Algorithms*. 2020 (cit. on pp. 49, 53).

[Cho, 2014] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, et al. “On the Properties of Neural Machine Translation: Encoder–Decoder Approaches”. *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*. Ed. by Dekai Wu, Marine Carpuat, Xavier Carreras, and Eva Maria Vecchi. Doha, Qatar: Association for Computational Linguistics, 2014, pp. 103–111 (cit. on pp. 27, 38).

[Chojnowski, 2014] Grzegorz Chojnowski, Tomasz Walen, and Janusz M. Bujnicki. “RNA Bricks–a Database of RNA 3D Motifs and Their Interactions”. *Nucleic Acids Research* 42.Database issue (2014), pp. D123–131 (cit. on p. 12).

[Collins, 1995] Francis S. Collins and Leslie Fink. “The Human Genome Project”. *Alcohol Health and Research World* 19.3 (1995), pp. 190–195 (cit. on p. 7).

[Cooper, 2009] Thomas A. Cooper, Lili Wan, and Gideon Dreyfuss. “RNA and Disease”. *Cell* 136.4 (2009), pp. 777–793 (cit. on p. 1).

[Coventry, 2004] Alex Coventry, Daniel J. Kleitman, and Bonnie Berger. “Msari: Multiple Sequence Alignments for Statistical Detection of RNA Secondary Structure”. *Proceedings of the National Academy of Sciences of the United States of America* 101.33 (2004), pp. 12102–12107 (cit. on p. 37).

[Creux, 2025] Constance Creux, Farida Zehraoui, François Radvanyi, et al. “MMnc: Multi-Modal Interpretable Representation for Non-Coding RNA Classification and Class Annotation”. *Bioinformatics* 41.3 (2025), btaf051 (cit. on p. 58).

[Crick, 1970] F. Crick. “Central Dogma of Molecular Biology”. *Nature* 227.5258 (1970), pp. 561–563 (cit. on pp. 5, 7).

[Dallaire, 2016] Paul Dallaire and François Major. “Exploring Alternative RNA Structure Sets Using MC-Flashfold and Db2cm”. *Methods in Molecular Biology (Clifton, N.J.)* 1490 (2016), pp. 237–251 (cit. on p. 49).

[Danaee, 2018] Padideh Danaee, Mason Rouches, Michelle Wiley, et al. "bpRNA: Large-Scale Automated Annotation and Analysis of RNA Secondary Structure". *Nucleic Acids Research* 46.11 (2018), pp. 5381–5394 (cit. on pp. 53, 67, 69).

[Daniel, 2015] Chammiran Daniel, Jens Lagergren, and Marie Öhman. "RNA Editing of Non-Coding RNA and Its Role in Gene Regulation". *Biochimie*. Special Issue: Regulatory RNAs 117 (2015), pp. 22–27 (cit. on p. 7).

[Dawson, 2006] Wayne Dawson, Kazuya Fujiwara, Gota Kawai, et al. "A Method for Finding Optimal Rna Secondary Structures Using a New Entropy Model (Vsfold)". *Nucleosides, Nucleotides & Nucleic Acids* 25.2 (2006), pp. 171–189 (cit. on p. 44).

[Dawson, 2015] Wayne Dawson and Gota Kawai. "A New Entropy Model for RNA: Part IV, The Minimum Free Energy (mFE) and the Thermodynamically Most-Probable Folding Pathway (TMPFP)". *Journal of Nucleic Acids Investigation* (2015) (cit. on p. 44).

[Dawson, 2014] Wayne Dawson, Toshikuni Takai, Nobuharu Ito, et al. "A New Entropy Model for RNA: Part III. Is the Folding Free Energy Landscape of RNA Funnel Shaped?" *Journal of Nucleic Acids Investigation* (2014) (cit. on p. 44).

[Dawson, 2013] Wayne Dawson, Kenji Yamamoto, Kentaro Shimizu, et al. "A New Entropy Model for RNA: Part II. Persistence-related Entropic Contributions to RNA Secondary Structure Free Energy Calculations". *Journal of Nucleic Acids Investigation* (2013) (cit. on p. 44).

[Delisi, 1971] C. Delisi and D. M. Crothers. "Prediction of RNA Secondary Structure". *Proceedings of the National Academy of Sciences of the United States of America* 68.11 (1971), pp. 2682–2685 (cit. on p. 43).

[Devlin, 2019] Jacob Devlin, Ming-Wei Chang, Kenton Lee, et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Ed. by Jill Burstein, Christy Doran, and Thamar Solorio. Minneapolis, Minnesota: Association for Computational Linguistics, 2019, pp. 4171–4186 (cit. on pp. 25, 28, 58).

[Ding, 2004] Ye Ding, Chi Yu Chan, and Charles E. Lawrence. "Sfold Web Server for Statistical Folding and Rational Design of Nucleic Acids". *Nucleic Acids Research* 32.Web Server issue (2004), W135–141 (cit. on p. 44).

[Ding, 2003] Ye Ding and Charles E. Lawrence. "A Statistical Sampling Algorithm for RNA Secondary Structure Prediction". *Nucleic Acids Research* 31.24 (2003), pp. 7280–7301 (cit. on pp. 37, 44).

[Ding, 2014] Yiliang Ding, Yin Tang, Chun Kit Kwok, et al. "In Vivo Genome-Wide Profiling of RNA Secondary Structure Reveals Novel Regulatory Features". *Nature* 505.7485 (2014), pp. 696–700 (cit. on p. 14).

[Dirks, 2003] Robert M. Dirks and Niles A. Pierce. "A Partition Function Algorithm for Nucleic Acid Secondary Structure Including Pseudoknots". *Journal of Computational Chemistry* 24.13 (2003), pp. 1664–1677 (cit. on p. 47).

[Do, 2006] Chuong B. Do, Daniel A. Woods, and Serafim Batzoglou. "CONTRAfold: RNA Secondary Structure Prediction without Physics-Based Models". *Bioinformatics (Oxford, England)* 22.14 (2006), e90–98 (cit. on pp. 44, 108).

[Doty, 1959] P. Doty, H. Boedtker, J. R. Fresco, et al. "SECONDARY STRUCTURE IN RIBONUCLEIC ACIDS". *Proceedings of the National Academy of Sciences of the United States of America* 45.4 (1959), pp. 482–499 (cit. on p. 43).

[Doudna, 2002] Jennifer A. Doudna and Thomas R. Cech. "The Chemical Repertoire of Natural Ribozymes". *Nature* 418.6894 (2002), pp. 222–228 (cit. on p. 7).

[Dunham, 2012] Ian Dunham, Anshul Kundaje, Shelley F. Aldred, et al. "An Integrated Encyclopedia of DNA Elements in the Human Genome". *Nature* 489.7414 (2012), pp. 57–74 (cit. on p. 7).

[Dutheil, 2012] Julien Y. Dutheil. "Detecting Coevolving Positions in a Molecule: Why and How to Account for Phylogeny". *Briefings in Bioinformatics* 13.2 (2012), pp. 228–243 (cit. on p. 37).

[Eddy, 2001] S. R. Eddy. "Non-Coding RNA Genes and the Modern RNA World". *Nature Reviews. Genetics* 2.12 (2001), pp. 919–929 (cit. on p. 6).

[Eddy, 2014] Sean R. Eddy. "Computational Analysis of Conserved RNA Secondary Structure in Transcriptomes and Genomes". *Annual Review of Biophysics* 43 (2014), pp. 433–456 (cit. on p. 37).

[Elman, 1990] Jeffrey L. Elman. "Finding Structure in Time". *Cognitive Science* 14.2 (1990), pp. 179–211 (cit. on p. 26).

[Engelen, 2010] Stéfan Engelen and Fariza Tahi. "Tfold: Efficient in Silico Prediction of Non-Coding RNA Secondary Structures". *Nucleic Acids Research* 38.7 (2010), pp. 2453–2466 (cit. on pp. 48, 55, 80, 94).

[Esteller, 2011] Manel Esteller. "Non-Coding RNAs in Human Disease". *Nature Reviews. Genetics* 12.12 (2011), pp. 861–874 (cit. on pp. 1, 7, 8).

[Fannon, 1996] Michael R. Fannon. "Gene Expression in Normal and Disease States — Identification of Therapeutic Targets". *Trends in Biotechnology* 14.8 (1996), pp. 294–298 (cit. on p. 2).

[Fire, 1998] Andrew Fire, SiQun Xu, Mary K. Montgomery, et al. "Potent and Specific Genetic Interference by Double-Stranded RNA in Caenorhabditis Elegans". *Nature* 391.6669 (1998), pp. 806–811 (cit. on p. 8).

[Fogel, 2002] Gary B. Fogel, V. William Porto, Dana G. Weekes, et al. "Discovery of RNA Structural Elements Using Evolutionary Computation". *Nucleic Acids Research* 30.23 (2002), pp. 5310–5317 (cit. on p. 6).

[Franke, 2024] Jörg K. H. Franke, Frederic Runge, Ryan Köksal, et al. *RNAformer: A Simple Yet Effective Deep Learning Model for RNA Secondary Structure Prediction*. 2024 (cit. on p. 50).

[Fu, 2022] Laiyi Fu, Yingxin Cao, Jie Wu, et al. "UFold: Fast and Accurate RNA Secondary Structure Prediction with Deep Learning". *Nucleic Acids Research* 50.3 (2022), e14 (cit. on pp. 50, 68, 91, 92, 94, 100, 101, 103, 106, 107, 115).

[Giedroc, 2000] David P Giedroc, Carla A Theimer, and Paul L Nixon. "Structure, Stability and Function of RNA Pseudoknots Involved in Stimulating Ribosomal Frameshifting". *Journal of Molecular Biology* 298.2 (2000), pp. 167–185 (cit. on p. 12).

[Gilbert, 1986] Walter Gilbert. "Origin of Life: The RNA World". *Nature* 319.6055 (1986), pp. 618–618 (cit. on p. 5).

[Gong, 2024] Tiansu Gong, Fusong Ju, and Dongbo Bu. "Accurate Prediction of RNA Secondary Structure Including Pseudoknots through Solving Minimum-Cost Flow with Learned Potentials". *Communications Biology* 7.1 (2024), pp. 1–13 (cit. on pp. 50, 53, 68, 69, 71, 72, 80–87, 91, 92, 94–104, 114, 115).

[Goodfellow, 2014] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, et al. *Generative Adversarial Networks*. 2014. arXiv: `1406.2661 [stat]` (cit. on p. 105).

[Gorodkin, 2001] Jan Gorodkin, Bjarne Knudsen, Christian Zwieb, et al. "SRPDB (Signal Recognition Particle Database)". *Nucleic Acids Research* 29.1 (2001), pp. 169–170 (cit. on p. 51).

[Greider, 1985] C. W. Greider and E. H. Blackburn. "Identification of a Specific Telomere Terminal Transferase Activity in Tetrahymena Extracts". *Cell* 43.2 Pt 1 (1985), pp. 405–413 (cit. on p. 7).

[Griffiths-Jones, 2003] Sam Griffiths-Jones, Alex Bateman, Mhairi Marshall, et al. “Rfam: An RNA Family Database”. *Nucleic Acids Research* 31.1 (2003), pp. 439–441 (cit. on pp. 52, 67).

[Hajdin, 2013] Christine E. Hajdin, Stanislav Bellaousov, Wayne Huggins, et al. “Accurate SHAPE-directed RNA Secondary Structure Modeling, Including Pseudoknots”. *Proceedings of the National Academy of Sciences* 110.14 (2013), pp. 5498–5503 (cit. on pp. 49, 94).

[Hamada, 2009] Michiaki Hamada, Hisanori Kiryu, Kengo Sato, et al. “Prediction of RNA Secondary Structure Using Generalized Centroid Estimators”. *Bioinformatics (Oxford, England)* 25.4 (2009), pp. 465–473 (cit. on p. 45).

[Hamilton, 1971] Walter Hamilton. “Crystallography: Protein Data Bank”. *Nature New Biology* 233.42 (1971), pp. 223–223 (cit. on p. 51).

[Hancock, 1988] J M Hancock, D Tautz, and G A Dover. “Evolution of the Secondary Structures and Compensatory Mutations of the Ribosomal RNAs of Drosophila Melanogaster.” *Molecular Biology and Evolution* 5.4 (1988), pp. 393–414 (cit. on pp. 9, 37).

[Harary, 2024] Marc Harary and Chengxin Zhang. *Kirigami: Large Convolutional Kernels Improve Deep Learning-Based RNA Secondary Structure Prediction*. 2024. arXiv: `2406.02381 [q-bio]` (cit. on p. 50).

[Al-Hashimi, 2008] Hashim M. Al-Hashimi and Nils G. Walter. “RNA Dynamics: It Is about Time”. *Current Opinion in Structural Biology* 18.3 (2008), pp. 321–329 (cit. on p. 8).

[Hochreiter, 1997] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-Term Memory”. *Neural Computation* 9.8 (1997), pp. 1735–1780 (cit. on pp. 26, 38).

[Hoeppner, 2012] Marc P. Hoeppner, Paul P. Gardner, and Anthony M. Poole. “Comparative Analysis of RNA Families Reveals Distinct Repertoires for Each Domain of Life”. *PLoS Computational Biology* 8.11 (2012), e1002752 (cit. on p. 7).

[Hofacker, 1994] I. L. Hofacker, W. Fontana, P. F. Stadler, et al. “Fast Folding and Comparison of RNA Secondary Structures”. *Monatshefte für Chemie / Chemical Monthly* 125.2 (1994), pp. 167–188 (cit. on pp. 43–45, 71, 72, 80–87, 114).

[Hofacker, 2003] Ivo L. Hofacker. “Vienna RNA Secondary Structure Server”. *Nucleic Acids Research* 31.13 (2003), pp. 3429–3431 (cit. on p. 44).

[Holland, 1982] J. Holland, K. Spindler, F. Horodyski, et al. “Rapid Evolution of RNA Genomes”. *Science (New York, N.Y.)* 215.4540 (1982), pp. 1577–1585 (cit. on p. 8).

[Holley, 1965] Robert W. Holley, Jean Apgar, George A. Everett, et al. “Structure of a Ribonucleic Acid”. *Science* 147.3664 (1965), pp. 1462–1465 (cit. on p. 8).

[Hori, 2021] Naoto Hori, Natalia A. Denesyuk, and D. Thirumalai. “Shape Changes and Cooperativity in the Folding of the Central Domain of the 16S Ribosomal RNA”. *Proceedings of the National Academy of Sciences* 118.10 (2021), e2020837118 (cit. on p. 83).

[Hornik, 1991] Kurt Hornik. “Approximation Capabilities of Multilayer Feedforward Networks”. *Neural Networks* 4.2 (1991), pp. 251–257 (cit. on p. 22).

[Hu, 2012] Wei-Shau Hu and Stephen H. Hughes. “HIV-1 Reverse Transcription”. *Cold Spring Harbor Perspectives in Medicine* 2.10 (2012), a006882 (cit. on p. 8).

[Huang, 2019] Liang Huang, He Zhang, Dezhong Deng, et al. “LinearFold: Linear-Time Approximate RNA Folding by 5’-to-3’ Dynamic Programming and Beam Search”. *Bioinformatics* 35.14 (2019), pp. i295–i304 (cit. on pp. 45, 71, 72, 80–87, 114).

[Jabbari, 2018] Hosna Jabbari, Ian Wark, Carlo Montemagno, et al. “Knotty: Efficient and Accurate Prediction of Complex RNA Pseudoknot Structures”. *Bioinformatics (Oxford, England)* 34.22 (2018), pp. 3849–3856 (cit. on p. 49).

[Jackson, 2010] Richard J. Jackson, Christopher U. T. Hellen, and Tatyana V. Pestova. “The Mechanism of Eukaryotic Translation Initiation and Principles of Its Regulation”. *Nature Reviews Molecular Cell Biology* 11.2 (2010), pp. 113–127 (cit. on p. 9).

[Jacob, 1961] François Jacob and Jacques Monod. "Genetic Regulatory Mechanisms in the Synthesis of Proteins". *Journal of Molecular Biology* 3.3 (1961), pp. 318–356 (cit. on p. 6).

[Jaeger, 1989] J. A. Jaeger, D. H. Turner, and M. Zuker. "Improved Predictions of Secondary Structures for RNA". *Proceedings of the National Academy of Sciences of the United States of America* 86.20 (1989), pp. 7706–7710 (cit. on p. 43).

[Jameson, 2015] J. Larry Jameson and Dan L. Longo. "Precision Medicine–Personalized, Problematic, and Promising". *The New England Journal of Medicine* 372.23 (2015), pp. 2229–2234 (cit. on p. 1).

[Janssen, 2015] Stefan Janssen and Robert Giegerich. "The RNA Shapes Studio". *Bioinformatics* 31.3 (2015), pp. 423–425 (cit. on pp. 49, 91, 92, 94, 97, 99–104, 115).

[Jeng, 1996] K. S. Jeng, A. Daniel, and M. M. Lai. "A Pseudoknot Ribozyme Structure Is Active in Vivo and Required for Hepatitis Delta Virus RNA Replication". *Journal of Virology* 70.4 (1996), pp. 2403–2410 (cit. on p. 11).

[Ji, 2021] Yanrong Ji, Zhihan Zhou, Han Liu, et al. "DNABERT: Pre-Trained Bidirectional Encoder Representations from Transformers Model for DNA-language in Genome". *Bioinformatics (Oxford, England)* 37.15 (2021), pp. 2112–2120 (cit. on pp. 57, 58).

[Jiang, 2019] Guosong Jiang, Ke Chen, and Jie Sun. "Accurate Prediction of Secondary Structure of tRNAs". *Biochemical and Biophysical Research Communications* 509.1 (2019), pp. 64–68 (cit. on p. 45).

[Jinek, 2012] Martin Jinek, Krzysztof Chylinski, Ines Fonfara, et al. "A Programmable Dual-RNA–Guided DNA Endonuclease in Adaptive Bacterial Immunity". *Science* 337.6096 (2012), pp. 816–821 (cit. on p. 8).

[Jühling, 2009] Frank Jühling, Mario Mörl, Roland K. Hartmann, et al. "tRNAdb 2009: Compilation of tRNA Sequences and tRNA Genes". *Nucleic Acids Research* 37.suppl_1 (2009), pp. D159–D162 (cit. on p. 52).

[Jung,] Andrew J Jung, Leo J Lee, Alice J Gao, et al. "RTfold: RNA Secondary Structure Prediction Using Deep Learning with Domain Inductive Bias" () (cit. on p. 45).

[Kalvari, 2021] Ioanna Kalvari, Eric P Nawrocki, Nancy Ontiveros-Palacios, et al. "Rfam 14: Expanded Coverage of Metagenomic, Viral and microRNA Families". *Nucleic Acids Research* 49.D1 (2021), pp. D192–D200 (cit. on pp. 7, 52, 69).

[Kandel, 2020] Ibrahem Kandel and Mauro Castelli. "The Effect of Batch Size on the Generalizability of the Convolutional Neural Networks on a Histopathology Dataset". *ICT Express* 6.4 (2020), pp. 312–315 (cit. on p. 16).

[Kertesz, 2010] Michael Kertesz, Yue Wan, Elad Mazor, et al. "Genome-Wide Measurement of RNA Secondary Structure in Yeast". *Nature* 467.7311 (2010), pp. 103–107 (cit. on p. 14).

[Kieft, 2008] Jeffrey S. Kieft. "Viral IRES RNA Structures and Ribosome Interactions". *Trends in biochemical sciences* 33.6 (2008), pp. 274–283 (cit. on p. 9).

[Kim, 1974] S. H. Kim, F. L. Suddath, G. J. Quigley, et al. "Three-Dimensional Tertiary Structure of Yeast Phenylalanine Transfer RNA". *Science (New York, N.Y.)* 185.4149 (1974), pp. 435–440 (cit. on pp. 5, 6).

[Kingma, 2017] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs] (cit. on pp. 16, 63).

[Kingma, 2022] Diederik P. Kingma and Max Welling. *Auto-Encoding Variational Bayes*. 2022. arXiv: 1312.6114 [stat] (cit. on p. 106).

[Knies, 2008] Jennifer L. Knies, Kristen K. Dang, Todd J. Vision, et al. "Compensatory Evolution in RNA Secondary Structures Increases Substitution Rate Variation among Sites". *Molecular Biology and Evolution* 25.8 (2008), pp. 1778–1787 (cit. on p. 37).

[Knight, 2007] Rob Knight, Peter Maxwell, Amanda Birmingham, et al. "PyCogent: A Toolkit for Making Sense from Sequence". *Genome Biology* 8.8 (2007), R171 (cit. on p. 61).

[Krizhevsky, 2012] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. *Advances in Neural Information Processing Systems*. Vol. 25. Curran Associates, Inc., 2012 (cit. on p. 24).

[Lander, 2001] Eric S. Lander, Lauren M. Linton, Bruce Birren, et al. “Initial Sequencing and Analysis of the Human Genome”. *Nature* 409.6822 (2001), pp. 860–921 (cit. on p. 7).

[Larsen, 1996] Niels Larsen and Christian Zwieb. “The Signal Recognition Particle Database (SRPDB)”. *Nucleic Acids Research* 24.1 (1996), pp. 80–81 (cit. on p. 51).

[LeCun, 2015] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep Learning”. *Nature* 521.7553 (2015), pp. 436–444 (cit. on pp. 15, 23).

[Lee, 2023] Nicholas Keone Lee, Ziqi Tang, Shushan Toneyan, et al. “EvoAug: Improving Generalization and Interpretability of Genomic Deep Neural Networks with Evolution-Inspired Data Augmentations”. *Genome Biology* 24.1 (2023), p. 105 (cit. on p. 106).

[Lee, 2013] Tong Ihn Lee and Richard A. Young. “Transcriptional Regulation and Its Misregulation in Disease”. *Cell* 152.6 (2013), pp. 1237–1251 (cit. on p. 2).

[Legendre, 2018] Audrey Legendre, Eric Angel, and Fariza Tahi. “Bi-Objective Integer Programming for RNA Secondary Structure Prediction with Pseudoknots”. *BMC bioinformatics* 19.1 (2018), p. 13 (cit. on pp. 11, 49, 94).

[Leontis, 2001] N. B. Leontis and E. Westhof. “Geometric Nomenclature and Classification of RNA Base Pairs”. *RNA (New York, N.Y.)* 7.4 (2001), pp. 499–512 (cit. on p. 8).

[Lewin, 2009] Geertje Lewin, Marek Matus, Abhijit Basu, et al. “Critical Role of Transcription Factor Cyclic AMP Response Element Modulator in Beta1-Adrenoceptor-Mediated Cardiac Dysfunction”. *Circulation* 119.1 (2009), pp. 79–88 (cit. on p. 1).

[Li, 2006] Weizhong Li and Adam Godzik. “Cd-Hit: A Fast Program for Clustering and Comparing Large Sets of Protein or Nucleotide Sequences”. *Bioinformatics* 22.13 (2006), pp. 1658–1659 (cit. on pp. 53, 68–70).

[Liu, 2005] Changning Liu, Baoyan Bai, Geir Skogerbø, et al. “NONCODE: An Integrated Knowledge Database of Non-Coding RNAs”. *Nucleic Acids Research* 33.Database Issue (2005), pp. D112–D115 (cit. on p. 2).

[Lorenz, 2011] Ronny Lorenz, Stephan H. Bernhart, Christian Höner zu Siederdissen, et al. “ViennaRNA Package 2.0”. *Algorithms for Molecular Biology* 6.1 (2011), p. 26 (cit. on pp. 44, 45, 48, 71, 72, 80–87, 114).

[Lu, 2021] Weizhong Lu, Yan Cao, Hongjie Wu, et al. “Research on RNA Secondary Structure Predicting via Bidirectional Recurrent Neural Network”. *BMC Bioinformatics* 22.Suppl 3 (2021), p. 431 (cit. on p. 50).

[Lu, 2009] Zhi John Lu, Jason W. Gloor, and David H. Mathews. “Improved RNA Secondary Structure Prediction by Maximizing Expected Pair Accuracy”. *RNA (New York, N.Y.)* 15.10 (2009), pp. 1805–1813 (cit. on p. 45).

[Lyngsø, 2000] R. B. Lyngsø and C. N. Pedersen. “RNA Pseudoknot Prediction in Energy-Based Models”. *Journal of Computational Biology: A Journal of Computational Molecular Cell Biology* 7.3-4 (2000), pp. 409–427 (cit. on p. 46).

[Lyngsø, 2004] Rune B. Lyngsø. “Complexity of Pseudoknot Prediction in Simple Models”. *Automata, Languages and Programming*. Ed. by Josep Díaz, Juhani Karhumäki, Arto Lepistö, and Donald Sannella. Berlin, Heidelberg: Springer, 2004, pp. 919–931 (cit. on p. 47).

[Makris, 2023] Evangelos Makris, Angelos Kolaitis, Christos Andrikos, et al. “Knotify+: Toward the Prediction of RNA H-Type Pseudoknots, Including Bulges and Internal Loops”. *Biomolecules* 13.2 (2023), p. 308 (cit. on p. 50).

[Mandal, 2004] Maumita Mandal and Ronald R. Breaker. “Gene Regulation by Riboswitches”. *Nature Reviews. Molecular Cell Biology* 5.6 (2004), pp. 451–463 (cit. on p. 7).

[Markham, 2008] Nicholas R. Markham and Michael Zuker. “UNAFold: Software for Nucleic Acid Folding and Hybridization”. *Methods in Molecular Biology (Clifton, N.J.)* 453 (2008), pp. 3–31 (cit. on p. 43).

[Mathews, 1999] D. H. Mathews, J. Sabina, M. Zuker, et al. “Expanded Sequence Dependence of Thermodynamic Parameters Improves Prediction of RNA Secondary Structure”. *Journal of Molecular Biology* 288.5 (1999), pp. 911–940 (cit. on p. 44).

[Mathews, 2006] David H. Mathews. “Revolutions in RNA Secondary Structure Prediction”. *Journal of Molecular Biology* 359.3 (2006), pp. 526–532 (cit. on p. 37).

[Mathews, 1997] David H. Mathews, Troy C. Andre, James Kim, et al. “An Updated Recursive Algorithm for RNA Secondary Structure Prediction with Improved Thermodynamic Parameters”. *Molecular Modeling of Nucleic Acids*. Vol. 682. ACS Symposium Series. American Chemical Society, 1997. Chap. 15, pp. 246–257 (cit. on p. 44).

[Mathews, 2004] David H. Mathews, Matthew D. Disney, Jessica L. Childs, et al. “Incorporating Chemical Modification Constraints into a Dynamic Programming Algorithm for Prediction of RNA Secondary Structure”. *Proceedings of the National Academy of Sciences of the United States of America* 101.19 (2004), pp. 7287–7292 (cit. on p. 34).

[Mattick, 2023] John S. Mattick, Paulo P. Amaral, Piero Carninci, et al. “Long Non-Coding RNAs: Definitions, Functions, Challenges and Recommendations”. *Nature Reviews Molecular Cell Biology* 24.6 (2023), pp. 430–447 (cit. on p. 6).

[McCaskill, 1990] J. S. McCaskill. “The Equilibrium Partition Function and Base Pair Binding Probabilities for RNA Secondary Structure”. *Biopolymers* 29.6-7 (1990), pp. 1105–1119 (cit. on pp. 35, 43).

[Merino, 2005] Edward J. Merino, Kevin A. Wilkinson, Jennifer L. Coughlan, et al. “RNA Structure Analysis at Single Nucleotide Resolution by Selective 2’-Hydroxyl Acylation and Primer Extension (SHAPE)”. *Journal of the American Chemical Society* 127.12 (2005), pp. 4223–4231 (cit. on p. 14).

[Metzler, 2008] Dirk Metzler and Markus E. Nebel. “Predicting RNA Secondary Structures with Pseudoknots by MCMC Sampling”. *Journal of Mathematical Biology* 56.1 (2008), pp. 161–181 (cit. on p. 47).

[Moore, 2002] Peter B. Moore and Thomas A. Steitz. “The Involvement of RNA in Ribosome Function”. *Nature* 418.6894 (2002), pp. 229–235 (cit. on p. 5).

[Nagaswamy, 2000] Uma Nagaswamy, Neil Voss, Zhengdong Zhang, et al. “Database of Non-Canonical Base Pairs Found in Known RNA Structures”. *Nucleic Acids Research* 28.1 (2000), pp. 375–376 (cit. on p. 8).

[Namy, 2006] Olivier Namy, Stephen J. Moran, David I. Stuart, et al. “A Mechanical Explanation of RNA Pseudoknot Function in Programmed Ribosomal Frameshifting”. *Nature* 441.7090 (2006), pp. 244–247 (cit. on p. 11).

[Nawrocki, 2015] Eric P. Nawrocki, Sarah W. Burge, Alex Bateman, et al. “Rfam 12.0: Updates to the RNA Families Database”. *Nucleic Acids Research* 43.Database issue (2015), pp. D130–137 (cit. on pp. 52, 67, 69).

[Noller, 1991] H. F. Noller. “Ribosomal RNA and Translation”. *Annual Review of Biochemistry* 60 (1991), pp. 191–227 (cit. on pp. 5, 6).

[Nussinov, 1980] R. Nussinov and A. B. Jacobson. “Fast Algorithm for Predicting the Secondary Structure of Single-Stranded RNA”. *Proceedings of the National Academy of Sciences of the United States of America* 77.11 (1980), pp. 6309–6313 (cit. on p. 43).

[Nussinov, 1978] Ruth Nussinov, George Pieczenik, Jerrold R. Griggs, et al. “Algorithms for Loop Matchings”. *SIAM Journal on Applied Mathematics* 35.1 (1978), pp. 68–82 (cit. on pp. 43, 47).

[Omnes, 2023] Loïc Omnes, Eric Angel, Pierre Bartet, et al. "Prediction of Secondary Structure for Long Non-Coding RNAs Using a Recursive Cutting Method Based on Deep Learning". *2023 IEEE 23rd International Conference on Bioinformatics and Bioengineering (BIBE)*. Dayton, OH, USA: IEEE, 2023, pp. 14–21 (cit. on pp. 3, 114).

[Omnes, 2025] Loïc Omnes, Eric Angel, Pierre Bartet, et al. "A Divide-and-Conquer Approach Based on Deep Learning for Long RNA Secondary Structure Prediction: Focus on Pseudoknots Identification". *PLOS ONE* 20.4 (2025), e0314837 (cit. on pp. 3, 115).

[Ontiveros-Palacios, 2025] Nancy Ontiveros-Palacios, Emma Cooke, Eric P Nawrocki, et al. "Rfam 15: RNA Families Database in 2025". *Nucleic Acids Research* 53.D1 (2025), pp. D258–D267 (cit. on pp. 7, 52, 69, 70).

[Ouyang, 2022] Jiawei Ouyang, Yu Zhong, Yijie Zhang, et al. "Long Non-Coding RNAs Are Involved in Alternative Splicing and Promote Cancer Progression". *British Journal of Cancer* 126.8 (2022), pp. 1113–1124 (cit. on p. 7).

[Parisien, 2008] Marc Parisien and François Major. "The MC-Fold and MC-Sym Pipeline Infers RNA Structure from Sequence Data". *Nature* 452.7183 (2008), pp. 51–55 (cit. on p. 47).

[Parker, 2011] Brian J. Parker, Ida Moltke, Adam Roth, et al. "New Families of Human Regulatory RNA Structures Identified by Comparative Analysis of Vertebrate Genomes". *Genome Research* 21.11 (2011), pp. 1929–1943 (cit. on p. 7).

[Payne, 2017] Susan Payne. "Introduction to RNA Viruses". *Viruses* (2017), pp. 97–105 (cit. on pp. 7, 8).

[Peattie, 1979] D. A. Peattie. "Direct Chemical Method for Sequencing RNA". *Proceedings of the National Academy of Sciences of the United States of America* 76.4 (1979), pp. 1760–1764 (cit. on p. 14).

[Plant, 2005] Ewan P. Plant and Jonathan D. Dinman. "Torsional Restraint: A New Twist on Frameshifting Pseudoknots". *Nucleic Acids Research* 33.6 (2005), pp. 1825–1833 (cit. on p. 11).

[Pleij, 1990] Cornelis W. A. Pleij. "Pseudoknots: A New Motif in the RNA Game". *Trends in Biochemical Sciences* 15.4 (1990), pp. 143–147 (cit. on p. 10).

[Ponting, 2009] Chris P. Ponting, Peter L. Oliver, and Wolf Reik. "Evolution and Functions of Long Noncoding RNAs". *Cell* 136.4 (2009), pp. 629–641 (cit. on pp. 5, 6).

[Poolsap, 2009] Unyanee Poolsap, Yuki Kato, and Tatsuya Akutsu. "Prediction of RNA Secondary Structure with Pseudoknots Using Integer Programming". *BMC Bioinformatics* 10.Suppl 1 (2009), S38 (cit. on p. 48).

[Powers, 2020] David M. W. Powers. *Evaluation: From Precision, Recall and F-measure to ROC, Informedness, Markedness and Correlation*. 2020. arXiv: `2010.16061 [cs]` (cit. on p. 19).

[Prasanth, 2007] Kannanganattu V. Prasanth and David L. Spector. "Eukaryotic Regulatory RNAs: An Answer to the 'genome Complexity' Conundrum". *Genes & Development* 21.1 (2007), pp. 11–42 (cit. on p. 1).

[Pratt, 2009] Ashley J. Pratt and Ian J. MacRae. "The RNA-induced Silencing Complex: A Versatile Gene-silencing Machine". *The Journal of Biological Chemistry* 284.27 (2009), pp. 17897–17901 (cit. on p. 7).

[Radom, 2013] Filip Radom, Przemysław M. Jurek, Maciej P. Mazurek, et al. "Aptamers: Molecules of Great Potential". *Biotechnology Advances* 31.8 (2013), pp. 1260–1274 (cit. on p. 7).

[Reeder, 2004] Jens Reeder and Robert Giegerich. "Design, Implementation and Evaluation of a Practical Pseudoknot Folding Algorithm Based on Thermodynamics". *BMC Bioinformatics* 5.1 (2004), p. 104 (cit. on p. 47).

[Regulski, 2008] Elizabeth E. Regulski and Ronald R. Breaker. "In-Line Probing Analysis of Riboswitches". *Methods in Molecular Biology (Clifton, N.J.)* 419 (2008), pp. 53–67 (cit. on p. 14).

[Reich, 1988] Claudia Reich, Gary J. Olsen, Bernadette Pace, et al. “Role of the Protein Moiety of Ribonuclease P, a Ribonucleoprotein Enzyme”. *Science* 239.4836 (1988), pp. 178–181 (cit. on p. 7).

[Reidys, 2011] Christian M. Reidys, Fenix W. D. Huang, Jørgen E. Andersen, et al. “Topology and Prediction of RNA Pseudoknots”. *Bioinformatics (Oxford, England)* 27.8 (2011), pp. 1076–1085 (cit. on p. 48).

[Reinharz, 2018] Vladimir Reinharz, Antoine Soulé, Eric Westhof, et al. “Mining for Recurrent Long-Range Interactions in RNA Structures Reveals Embedded Hierarchies in Network Families”. *Nucleic Acids Research* 46.8 (2018), pp. 3841–3851 (cit. on pp. 12, 13, 58).

[Ren, 2005] Jihong Ren, Baharak Rastegari, Anne Condon, et al. “HotKnots: Heuristic Prediction of RNA Secondary Structures Including Pseudoknots”. *RNA (New York, N.Y.)* 11.10 (2005), pp. 1494–1504 (cit. on p. 47).

[Reuter, 2010] Jessica S. Reuter and David H. Mathews. “RNAstructure: Software for RNA Secondary Structure Prediction and Analysis”. *BMC Bioinformatics* 11.1 (2010), p. 129 (cit. on p. 44).

[Rivas, 1999] E. Rivas and S. R. Eddy. “A Dynamic Programming Algorithm for RNA Structure Prediction Including Pseudoknots”. *Journal of Molecular Biology* 285.5 (1999), pp. 2053–2068 (cit. on p. 46).

[Rivas, 2021] Elena Rivas. “Evolutionary Conservation of RNA Sequence and Structure”. *Wiley Interdisciplinary Reviews. RNA* 12.5 (2021), e1649 (cit. on p. 37).

[Romilly, 2020] Cédric Romilly, Anne Lippegaus, and E Gerhart H Wagner. “An RNA Pseudoknot Is Essential for Standby-Mediated Translation of the tisB Toxin mRNA in Escherichia Coli”. *Nucleic Acids Research* 48.21 (2020), pp. 12336–12347 (cit. on p. 12).

[Rosenblad, 2003] Magnus Alm Rosenblad, Jan Gorodkin, Bjarne Knudsen, et al. “SRPDB: Signal Recognition Particle Database”. *Nucleic Acids Research* 31.1 (2003), pp. 363–364 (cit. on p. 51).

[Rosenblatt, 1958] F. Rosenblatt. “The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain”. *Psychological Review* 65.6 (1958), pp. 386–408 (cit. on pp. 15, 21).

[Ruan, 2004] Jianhua Ruan, Gary D. Stormo, and Weixiong Zhang. “An Iterated Loop Matching Approach to the Prediction of RNA Secondary Structures with Pseudoknots”. *Bioinformatics (Oxford, England)* 20.1 (2004), pp. 58–66 (cit. on p. 47).

[Rumelhart, 1986] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. “Learning Representations by Back-Propagating Errors”. *Nature* 323.6088 (1986), pp. 533–536 (cit. on pp. 16, 26).

[Salser, 1978] W. Salser. “Globin mRNA Sequences: Analysis of Base Pairing and Evolutionary Implications”. *Cold Spring Harbor Symposia on Quantitative Biology* 42 Pt 2 (1978), pp. 985–1002 (cit. on p. 34).

[Saman Booy, 2022] Mehdi Saman Booy, Alexander Ilin, and Pekka Orponen. “RNA Secondary Structure Prediction with Convolutional Neural Networks”. *BMC Bioinformatics* 23.1 (2022), p. 58 (cit. on pp. 50, 53, 68).

[Samuelsson, 1999] T Samuelsson and C Zwieb. “The Signal Recognition Particle Database (SRPDB).” *Nucleic Acids Research* 27.1 (1999), pp. 169–170 (cit. on p. 51).

[Sato, 2021] Kengo Sato, Manato Akiyama, and Yasubumi Sakakibara. “RNA Secondary Structure Prediction Using Deep Learning with Thermodynamic Integration”. *Nature Communications* 12.1 (2021), p. 941 (cit. on pp. 45, 53, 68, 69, 71, 72, 80–87, 91, 94, 114).

[Sato, 2009] Kengo Sato, Michiaki Hamada, Kiyoshi Asai, et al. “CentroidFold: A Web Server for RNA Secondary Structure Prediction”. *Nucleic Acids Research* 37.Web Server issue (2009), W277–W280 (cit. on p. 45).

[Sato, 2022] Kengo Sato and Yuki Kato. “Prediction of RNA Secondary Structure Including Pseudoknots for Long Sequences”. *Briefings in Bioinformatics* 23.1 (2022), bbab395 (cit. on pp. 48, 54, 91, 92, 94–104, 115).

[Sato, 2011] Kengo Sato, Yuki Kato, Michiaki Hamada, et al. “IPknot: Fast and Accurate Prediction of RNA Secondary Structures with Pseudoknots Using Integer Programming”. *Bioinformatics* 27.13 (2011), pp. i85–i93 (cit. on pp. 48, 54, 91, 92, 94–104, 115).

[Schuster, 1997] M. Schuster and K.K. Paliwal. “Bidirectional Recurrent Neural Networks”. *IEEE Transactions on Signal Processing* 45.11 (1997), pp. 2673–2681 (cit. on p. 27).

[Serra, 1995] M. J. Serra and D. H. Turner. “Predicting Thermodynamic Properties of RNA”. *Methods in Enzymology* 259 (1995), pp. 242–261 (cit. on p. 34).

[Shahgir, 2024] Haz Sameen Shahgir, Md Rownok Zahan Ratul, Md Toki Tahmid, et al. *RNA-DCGen: Dual Constrained RNA Sequence Generation with LLM-Attack*. 2024 (cit. on p. 106).

[Sharp, 1985] Phillip A. Sharp. “On the Origin of RNA Splicing and Introns”. *Cell* 42.2 (1985), pp. 397–400 (cit. on p. 6).

[Shorten, 2019] Connor Shorten and Taghi M. Khoshgoftaar. “A Survey on Image Data Augmentation for Deep Learning”. *Journal of Big Data* 6.1 (2019), p. 60 (cit. on p. 105).

[Singh, 2019] Jaswinder Singh, Jack Hanson, Kuldip Paliwal, et al. “RNA Secondary Structure Prediction Using an Ensemble of Two-Dimensional Deep Neural Networks and Transfer Learning”. *Nature Communications* 10.1 (2019), p. 5407 (cit. on pp. 49, 62, 68, 94).

[Singh, 2021] Jaswinder Singh, Kuldip Paliwal, Tongchuan Zhang, et al. “Improved RNA Secondary Structure and Tertiary Base-Pairing Prediction Using Evolutionary Profile, Mutational Coupling and Two-Dimensional Transfer Learning”. *Bioinformatics (Oxford, England)* 37.17 (2021), pp. 2589–2600 (cit. on pp. 49, 68, 94).

[Slack, 2019] Frank J. Slack and Arul M. Chinnaiyan. “The Role of Non-coding RNAs in Oncology”. *Cell* 179.5 (2019), pp. 1033–1055 (cit. on pp. 1, 2, 8).

[Sloma, 2016] Michael F. Sloma and David H. Mathews. “Exact Calculation of Loop Formation Probability Identifies Folding Motifs in RNA Secondary Structures”. *RNA (New York, N.Y.)* 22.12 (2016), pp. 1808–1818 (cit. on pp. 52, 67).

[Smirnov, 2017] Alexandre Smirnov, Cornelius Schneider, Jens Hör, et al. “Discovery of New RNA Classes and Global RNA-binding Proteins”. *Current Opinion in Microbiology* 39 (2017), pp. 152–160 (cit. on p. 6).

[Smit, 2008] Sandra Smit, Kristian Rother, Jaap Heringa, et al. “From Knotted to Nested RNA Structures: A Variety of Computational Methods for Pseudoknot Removal”. *RNA* 14.3 (2008), pp. 410–416 (cit. on pp. 61, 73).

[Sohl-Dickstein, 2015] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, et al. *Deep Unsupervised Learning Using Nonequilibrium Thermodynamics*. 2015. arXiv: `1503.03585 [cs]` (cit. on p. 106).

[Soukup, 1999] G. A. Soukup and R. R. Breaker. “Relationship between Internucleotide Linkage Geometry and the Stability of RNA”. *RNA (New York, N.Y.)* 5.10 (1999), pp. 1308–1325 (cit. on p. 14).

[Sprinzl, 1998] M. Sprinzl, C. Horn, M. Brown, et al. “Compilation of tRNA Sequences and Sequences of tRNA Genes”. *Nucleic Acids Research* 26.1 (1998), pp. 148–153 (cit. on p. 52).

[Sprinzl, 2005] Mathias Sprinzl and Konstantin S. Vassilenko. “Compilation of tRNA Sequences and Sequences of tRNA Genes”. *Nucleic Acids Research* 33.Database issue (2005), pp. D139–140 (cit. on p. 52).

[Staple, 2005] David W Staple and Samuel E Butcher. “Pseudoknots: RNA Structures with Diverse Functions”. *PLoS Biology* 3.6 (2005), e213 (cit. on pp. 9, 10).

[Steffen, 2006] Peter Steffen, Björn Voß, Marc Rehmsmeier, et al. “RNAshapes: An Integrated RNA Analysis Package Based on Abstract Shapes”. *Bioinformatics* 22.4 (2006), pp. 500–503 (cit. on p. 44).

[Steitz, 2003] Thomas A. Steitz and Peter B. Moore. “RNA, the First Macromolecular Catalyst: The Ribosome Is a Ribozyme”. *Trends in Biochemical Sciences* 28.8 (2003), pp. 411–418 (cit. on pp. 5, 6).

[Stoltenburg, 2007] Regina Stoltenburg, Christine Reinemann, and Beate Strehlitz. “SELEX–a (r)Evolutionary Method to Generate High-Affinity Nucleic Acid Ligands”. *Biomolecular Engineering* 24.4 (2007), pp. 381–403 (cit. on p. 7).

[Svoboda, 2006] P. Svoboda and A. Di Cara. “Hairpin RNA: A Secondary Structure of Primary Importance”. *Cellular and molecular life sciences: CMLS* 63.7-8 (2006), pp. 901–908 (cit. on p. 8).

[Szikszai, 2022] Marcell Szikszai, Michael Wise, Amitava Datta, et al. “Deep Learning Models for RNA Secondary Structure Prediction (Probably) Do Not Generalize across Families”. *Bioinformatics* 38.16 (2022), pp. 3892–3899 (cit. on pp. 53, 69).

[Tan, 2017] Zhen Tan, Yinghan Fu, Gaurav Sharma, et al. “TurboFold II: RNA Structural Alignment and Secondary Structure Prediction Informed by Multiple Homologs”. *Nucleic Acids Research* 45.20 (2017), pp. 11570–11581 (cit. on pp. 53, 67).

[Taufer, 2009] Michela Taufer, Abel Licon, Roberto Araiza, et al. “PseudoBase++: An Extension of PseudoBase for Easy Searching, Formatting and Visualization of Pseudoknots”. *Nucleic Acids Research* 37.Database issue (2009), pp. D127–D135 (cit. on p. 10).

[Temin, 1970] Howard M. Temin and Satoshi Mizutani. “Viral RNA-dependent DNA Polymerase: RNA-dependent DNA Polymerase in Virions of Rous Sarcoma Virus”. *Nature* 226.5252 (1970), pp. 1211–1213 (cit. on p. 8).

[Theis, 2010] Corinna Theis, Stefan Janssen, and Robert Giegerich. “Prediction of RNA Secondary Structure Including Kissing Hairpin Motifs”. *Algorithms in Bioinformatics*. Ed. by Vincent Moulton and Mona Singh. Berlin, Heidelberg: Springer, 2010, pp. 52–64 (cit. on pp. 49, 91, 92, 94, 97, 99–104, 115).

[Thomas, 2008] Jason R. Thomas and Paul J. Hergenrother. “Targeting RNA with Small Molecules”. *Chemical Reviews* 108.4 (2008), pp. 1171–1224 (cit. on p. 2).

[Tinoco, 1973] I. Tinoco, P. N. Borer, B. Dengler, et al. “Improved Estimation of Secondary Structure in Ribonucleic Acids”. *Nature: New Biology* 246.150 (1973), pp. 40–41 (cit. on p. 43).

[Tinoco, 1999] I. Tinoco and C. Bustamante. “How RNA Folds”. *Journal of Molecular Biology* 293.2 (1999), pp. 271–281 (cit. on pp. 6, 8).

[Tinoco, 1971] Ignacio Tinoco, Olke C. Uhlenbeck, and Mark D. Levine. “Estimation of Secondary Structure in Ribonucleic Acids”. *Nature* 230.5293 (1971), pp. 362–367 (cit. on p. 43).

[Turner, 1988] D. H. Turner, N. Sugimoto, and S. M. Freier. “RNA Structure Prediction”. *Annual Review of Biophysics and Biophysical Chemistry* 17 (1988), pp. 167–192 (cit. on p. 34).

[Turner, 2010] Douglas H. Turner and David H. Mathews. “NNDB: The Nearest Neighbor Parameter Database for Predicting Stability of Nucleic Acid Secondary Structure”. *Nucleic Acids Research* 38.suppl_1 (2010), pp. D280–D282 (cit. on p. 34).

[Underwood, 2010] Jason G. Underwood, Andrew V. Uzilov, Sol Katzman, et al. “FragSeq: Transcriptome-Wide RNA Structure Probing Using High-Throughput Sequencing”. *Nature Methods* 7.12 (2010), pp. 995–1001 (cit. on p. 14).

[Varani, 2000] G. Varani and W. H. McClain. “The G x U Wobble Base Pair. A Fundamental Building Block of RNA Structure Crucial to RNA Function in Diverse Biological Systems”. *EMBO reports* 1.1 (2000), pp. 18–23 (cit. on p. 8).

[Vaswani, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, et al. “Attention Is All You Need”. *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS’17. Red Hook, NY, USA: Curran Associates Inc., 2017, pp. 6000–6010 (cit. on pp. 28, 30).

[Virtanen, 2020] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, et al. “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python”. *Nature Methods* 17.3 (2020), pp. 261–272 (cit. on p. 64).

[Waldispühl, 2007] J. Waldispühl and P. Clote. “Computing the Partition Function and Sampling for Saturated Secondary Structures of RNA, with Respect to the Turner Energy Model”. *Journal of Computational Biology: A Journal of Computational Molecular Cell Biology* 14.2 (2007), pp. 190–215 (cit. on p. 37).

[Walter, 1994] A E Walter, D H Turner, J Kim, et al. “Coaxial Stacking of Helixes Enhances Binding of Oligoribonucleotides and Improves Predictions of RNA Folding.” *Proceedings of the National Academy of Sciences of the United States of America* 91.20 (1994), pp. 9218–9222 (cit. on p. 44).

[Wan, 2014] Yue Wan, Kun Qu, Qiangfeng Cliff Zhang, et al. “Landscape and Variation of RNA Secondary Structure across the Human Transcriptome”. *Nature* 505.7485 (2014), pp. 706–709 (cit. on pp. 1, 2, 9).

[Wang, 2019a] Linyu Wang, Yuanning Liu, Xiaodan Zhong, et al. “DMfold: A Novel Method to Predict RNA Secondary Structure With Pseudoknots Based on Deep Learning and Improved Base Pair Maximization Principle”. *Frontiers in Genetics* 10 (2019), p. 143 (cit. on p. 49).

[Wang, 2022] Wen-Hung Wang, Arunee Thitithanyanont, and Sheng-Fan Wang. “Synthesis, Assembly and Processing of Viral Proteins”. *Viruses* 14.8 (2022), p. 1650 (cit. on p. 8).

[Wang, 2019b] Xinlu Wang, Yifang Xuan, Yuling Han, et al. “Regulation of HIV-1 Gag-Pol Expression by Shiftless, an Inhibitor of Programmed -1 Ribosomal Frameshifting”. *Cell* 176.3 (2019), 625–635.e14 (cit. on p. 11).

[Wang, 2020] Yili Wang, Yuanning Liu, Shuo Wang, et al. “ATTfold: RNA Secondary Structure Prediction With Pseudoknots Based on Attention Mechanism”. *Frontiers in Genetics* 11 (2020), p. 612086 (cit. on p. 49).

[Wang, 2025] Zhen Wang, Yizhen Feng, Qingwen Tian, et al. “RNADiffFold: Generative RNA Secondary Structure Prediction Using Discrete Diffusion Models”. *Briefings in Bioinformatics* 26.1 (2025), bbae618 (cit. on pp. 46, 106, 107).

[Waterman, 1978] M. S. Waterman and T. F. Smith. “RNA Secondary Structure: A Complete Mathematical Analysis”. *Mathematical Biosciences* 42.3 (1978), pp. 257–266 (cit. on p. 43).

[Watson, 1963] J. D. Watson. “Involvement of RNA in the Synthesis of Proteins”. *Science (New York, N.Y.)* 140.3562 (1963), pp. 17–26 (cit. on p. 6).

[Weeks, 2021] Kevin Weeks. *Dr. Kevin Weeks: A MaP of RNA Structural Landscapes*. 2021 (cit. on p. 9).

[Weiss, 1991] B G Weiss and S Schlesinger. “Recombination between Sindbis Virus RNAs.” *Journal of Virology* 65.8 (1991), pp. 4017–4025 (cit. on p. 7).

[Will, 2011] Cindy L. Will and Reinhard Lührmann. “Spliceosome Structure and Function”. *Cold Spring Harbor Perspectives in Biology* 3.7 (2011), a003707 (cit. on p. 6).

[Williamson, 2005] James R. Williamson. “Assembly of the 30S Ribosomal Subunit”. *Quarterly Reviews of Biophysics* 38.4 (2005), pp. 397–403 (cit. on pp. 84, 98).

[Willmott, 2020] Devin Willmott, David Murrugarra, and Qiang Ye. “Improving RNA Secondary Structure Prediction via State Inference with Deep Recurrent Neural Networks”. *Computational and Mathematical Biophysics* 8.1 (2020), pp. 36–50 (cit. on p. 39).

[Wilusz, 2009] Jeremy E. Wilusz, Hongjae Sunwoo, and David L. Spector. "Long Noncoding RNAs: Functional Surprises from the RNA World". *Genes & Development* 23.13 (2009), pp. 1494–1504 (cit. on p. 7).

[Winkler, 2005] Wade C. Winkler and Ronald R. Breaker. "Regulation of Bacterial Gene Expression by Riboswitches". *Annual Review of Microbiology* 59 (2005), pp. 487–517 (cit. on p. 7).

[Wirecki, 2020] Tomasz K Wirecki, Katarzyna Merdas, Agata Bernat, et al. "RNAProbe: A Web Server for Normalization and Analysis of RNA Structure Probing Data". *Nucleic Acids Research* 48.W1 (2020), W292–W299 (cit. on p. 15).

[Wuchty, 1999] S. Wuchty, W. Fontana, I. L. Hofacker, et al. "Complete Suboptimal Folding of RNA and the Stability of Secondary Structures". *Biopolymers* 49.2 (1999), pp. 145–165 (cit. on pp. 34, 44).

[Xia, 1998] Tianbing Xia, John SantaLucia, Mark E. Burkard, et al. "Thermodynamic Parameters for an Expanded Nearest-Neighbor Model for Formation of RNA Duplexes with Watson-Crick Base Pairs". *Biochemistry* 37.42 (1998), pp. 14719–14735 (cit. on p. 34).

[Yang, 2024] Tzu-Hsien Yang. "DEBFold: Computational Identification of RNA Secondary Structures for Sequences across Structural Families Using Deep Learning". *Journal of Chemical Information and Modeling* 64.9 (2024), pp. 3756–3766 (cit. on pp. 50, 53, 69).

[Yankey, 2021] Allison Yankey, Sean C. Clark, Michael C. Owens, et al. "Purification and Structural Characterization of the Long Noncoding RNAs". *Methods in Molecular Biology (Clifton, N.J.)* 2372 (2021), pp. 93–110 (cit. on p. 2).

[Ying, 2019] Xue Ying. "An Overview of Overfitting and Its Solutions". *Journal of Physics: Conference Series* 1168.2 (2019), p. 022022 (cit. on p. 17).

[Yu, 2023] Daseuli Yu, Hee-Jeong Han, Jeonghye Yu, et al. "Pseudoknot-Targeting Cas13b Combats SARS-CoV-2 Infection by Suppressing Viral Replication". *Molecular Therapy* 31.6 (2023), pp. 1675–1687 (cit. on p. 12).

[Zadeh, 2011] Joseph N. Zadeh, Conrad D. Steenberg, Justin S. Bois, et al. "NUPACK: Analysis and Design of Nucleic Acid Systems". *Journal of Computational Chemistry* 32.1 (2011), pp. 170–173 (cit. on p. 47).

[Zakov, 2011] Shay Zakov, Yoav Goldberg, Michael Elhadad, et al. "Rich Parameterization Improves RNA Structure Prediction". *Journal of Computational Biology: A Journal of Computational Molecular Cell Biology* 18.11 (2011), pp. 1525–1542 (cit. on p. 45).

[Zhang, 2020] He Zhang, Liang Zhang, David H Mathews, et al. "LinearPartition: Linear-Time Approximation of RNA Folding Partition Function and Base-Pairing Probabilities". *Bioinformatics* 36.Supplement_1 (2020), pp. i258–i267 (cit. on p. 48).

[Zhao, 2023] Qi Zhao, Qian Mao, Zheng Zhao, et al. "RNA Independent Fragment Partition Method Based on Deep Learning for RNA Secondary Structure Prediction". *Scientific Reports* 13.1 (2023), p. 2861 (cit. on pp. 55, 80, 94).

[Zhao, 2024] Yichong Zhao, Kenta Oono, Hiroki Takizawa, et al. "GenerRNA: A Generative Pre-Trained Language Model for de Novo RNA Design". *PLOS ONE* 19.10 (2024), e0310814 (cit. on p. 106).

[Zuker, 1989a] M. Zuker. "Computer Prediction of RNA Structure". *Methods in Enzymology* 180 (1989), pp. 262–288 (cit. on pp. 34, 43).

[Zuker, 1981] M. Zuker and P. Stiegler. "Optimal Computer Folding of Large RNA Sequences Using Thermodynamics and Auxiliary Information". *Nucleic Acids Research* 9.1 (1981), pp. 133–148 (cit. on pp. 34, 35, 43).

[Zuker, 1989b] Michael Zuker. "On Finding All Suboptimal Foldings of an RNA Molecule". *Science* 244.4900 (1989), pp. 48–52 (cit. on pp. 35, 43).

[Zuker, 2003] Michael Zuker. "Mfold Web Server for Nucleic Acid Folding and Hybridization Prediction". *Nucleic Acids Research* 31.13 (2003), pp. 3406–3415 (cit. on p. 43).

[Zwieb, 1998] C. Zwieb, N. Larsen, and J. Wower. “The tmRNA Database (tmRDB)”. *Nucleic Acids Research* 26.1 (1998), pp. 166–167 (cit. on p. 52).

[Zwieb, 2003] Christian Zwieb, Jan Gorodkin, Bjarne Knudsen, et al. “tmRDB (tmRNA Database)”. *Nucleic Acids Research* 31.1 (2003), pp. 446–447 (cit. on p. 52).