UNIVERSITY OF ZAGREB
FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

MASTER THESIS No. 2917

COMBINING PROTEIN AND RNA STRUCTURES INFORMATION IN DEVELOPING NEW SCORING FUNCTION

Ivona Martinović

Zagreb, June 2022

UNIVERSITY OF ZAGREB
FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

MASTER THESIS No. 2917

COMBINING PROTEIN AND RNA STRUCTURES INFORMATION IN DEVELOPING NEW SCORING FUNCTION

Ivona Martinović

Zagreb, June 2022

UNIVERSITY OF ZAGREB
FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

Zagreb, 11 March 2022

MASTER THESIS ASSIGNMENT No. 2917

Student:	**Ivona Martinović (0036506038)**
Study:	Computing
Profile:	Computer Science
Mentor:	prof. Mile Šikić

Title: **Combining Protein and RNA Structures Information in Developing New Scoring Function**

Description:

Following the recent breakthrough in protein structure prediction achieved by Deepmind's AlphaFold2 neural network model, there is an interest in predicting RNA structures using machine learning techniques. Unfortunately, compared to proteins, there are far fewer structural measurements of RNA molecules currently available, which adds to the difficulty of the task. One way of approaching this problem is to train a neural network that can score how far a given configuration of atoms is from being a valid RNA molecule. Combining this with some of the existing algorithms for stochastic molecular generation gives a solid RNA structure prediction model. Such a model, named ARES (Atomic Rotationally Equivariant Scorer), has recently been implemented. The task of this work is to investigate to which extent information from the vast pool of protein structure data can be used to improve the scoring of RNA molecules. On a fundamental level, both proteins and RNA molecules are long chains of organic compounds and fold under similar conditions inside a cell. Concretely, the task is to adjust and retrain the ARES neural network model for atom configuration scoring and obtain results on protein and RNA structure data. The results should then be compared to the original model. The solution should be implemented in Python with Pytorch or a similar deep learning library. The source code should be documented using comments and follow the Google Python Style Guide. The complete application should be hosted on GitHub under an OSI approved license.

Submission date: 27 June 2022

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 11. ožujka 2022.

DIPLOMSKI ZADATAK br. 2917

Pristupnica:	**Ivona Martinović (0036506038)**
Studij:	Računarstvo
Profil:	Računarska znanost
Mentor:	prof. dr. sc. Mile Šikić

Zadatak: **Kombiniranje informacija o strukturi proteina i RNA u razvoju nove funkcije bodovanje temeljene na dubokom učenju**

Opis zadatka:

Nakon neočekivanih dostignuća koja je Deepmind-ov model Alphafold2, temeljen na neuronskim mrežama, nedavno ostvario na području predviđanja strukture proteina, postoji interes za predviđanje strukture RNA molekula pomoću tehnika strojnog učenja. Za razliku od proteina, trenutno je dostupno značajeno manje strukturnih mjerenja RNA molekula, što dodatno otežava problem. Jedan od pristupa je treniranje neuronske mreže koja bi ocjenjivala koliko je neka konfiguracija atoma daleko od toga da bude ispravna RNA molekula. U kombinaciji s nekim od postojećih algoritama za stohastičku generaciju približne strukture molekula, tako istrenirana mreža daje razuman model za predikciju struktura RNA molekula. Jedan takav model, nazvan ARES (Atomic Rotationally Equivariant Scorer), nedavno je implementiran. Zadatak ovoga rada je istražiti u kolikoj se mjeri velika količina podataka o proteinskim strukturama može iskoristiti da bi se popravilo ocjenjivanje strukture RNA molekula. I proteini i RNA molekule su u osnovi dugi lanci organskih spojeva koji se svijaju u sličnom okruženju unutar stanice. Konkretno, zadatak je podesiti i ponovno istrenirati neuronsku mrežu za ocjenjivanje konfiguracija atoma iz sistema ARES te usporedit njezine rezultate na strukturnim podatcima za proteine i RNA molekule. Rezultate usporediti s originalnim ARES modelom. Rješenje treba biti implementirano u Pythonu koristeći Pytorch ili sličnu biblioteku za duboko učenje. Kod treba biti dokumentirati koristeći komentare i razvijati prema Google Python Style Guide ako je moguće. Cijeli programski proizvod potrebno je postaviti na GitHub pod jednu od OSI odabranih licenci.

Rok za predaju rada: 27. lipnja 2022.

Big thanks to
Professor Mile Šikić,
Filip, Rafael, Josipa,
Frigi, Sara, Dominik, Lovro,
Iva, Angus, Daniel,
Filip, Hana, Helena, Laura,
and to my family
for their support and patience.

TABLE OF CONTENTS

1. Introduction

Understanding both protein and RNA structures greatly facilitates understanding of their functionality. Knowing protein structure is crucial for our knowledge of how they serve as enzymes, bind to other proteins in signal cascades, are misregulated in diseases and are targeted by antibodies and small molecules in therapies. RNAs also fold into simple and complex structures to regulate every step of their lifecycle, including transcription, translation, decay and RNA transport.

The 'protein folding problem' (Dill et al., 2008) has been an open research problem for more than 50 years (Anfinsen, 1973). As a result of enormous experimental effort, there are currently more than 150000 high-resolution protein structures in the Protein Data Bank (PDB) (Bank, 1971) - a database for the three-dimensional structural data of large biological molecules, such as proteins and nucleic acids. Even if the number of protein structures may seem big, it is nothing in comparison to the billions of known protein sequences. For RNA there exist only about 5000 structures in PDB, which makes learning RNA structures using AI significantly more challenging.

One of the most significant scientific discoveries in 2021 has been AlphaFold2 (Jumper et al., 2021) method for the determination of protein 3D structures developed by DeepMind. This model takes protein sequence as input and outputs the 3D structure of this protein. The 3D structure refers to the three-dimensional coordinates of atoms in an amino acid-chain molecule.

While several AI-based RNA prediction programs including MXFold2 (Sato et al., 2021), SPOT-RNA (Singh et al., 2019) have been developed to varying success, they mostly predict RNA secondary and not tertiary structures (explained in Section 2.1). Recently, the first 3D RNA structure prediction with promising results was developed, called DeepFoldRNA (Pearce et al., 2022).

The main problem with predicting structures for sequences for which experimentally determined structure doesn't exist is how to validate the obtained result. There are worldwide experiments, such as Critical Assessment of protein Structure Prediction (CASP) (Moult et al., 1995) and RNA-Puzzles blind RNA structure prediction

challenges (Miao and Westhof, 2017), which give an opportunity to objectively test structure prediction methods in a double-blinded fashion (neither predictors nor organisers and assessors know the structures before the evaluation). The structures that competitors try to solve are usually soon-to-be experimentally solved or just solved and are kept on hold by PDB. Even though this could be used as a type of evaluation, there are not many structures in these experiments, and CASP takes place every two years since 1994, while RNA-Puzzles have had only four rounds since 2012.

Since determining RNA structures is more challenging, a recent discovery that it is possible to predict the root mean square deviation (RMSD) from a predicted structure to a natural with a method called ARES (Atomic Rotationally Equivariant Scorer) (Townshend et al., 2021), greatly facilitates the evaluation of prediction models. It could also enable selecting the best structures according to ARES and couple them with the starting set of structures to train a new model which could lead to even better prediction models.

ARES was trained using only 18 RNA molecules for which experimentally determined structures were published between 1994 and 2006. As ARES learns entirely from atomic structure, using no other information such as related RNA or protein sequences, and it makes no assumptions about what structural features might be important, the aim of this thesis is to investigate to which extent information from the vast pool of protein structure data can be used to improve the scoring of RNA molecules.

The remainder of this thesis is structured as follows: Chapter 2 introduces the datasets used in this thesis, then in Chapter 3 there is a description of the model, after which it moves on with results in Chapter 4 and discussion in Chapter 5, and concludes with Chapter 6.

2. Data Summary

2.1. Structure types

Term primary structure refers to the sequence of amino acids or nucleotides. Secondary structure for proteins is a term for local sub-structures on the polypeptide backbone chain and the secondary structures are defined by hydrogen bonds between the main-chain peptide groups. There are two main types: the α-helix and the β-sheets. RNA secondary structure consists of nucleotides which are paired or unpaired - although RNA is a single-stranded molecule it can fold over and form hairpin loops. Tertiary protein structure is a three-dimensional structure created by a single polypeptide chain, while quaternary structure consists of two or more individual polypeptide chains that function together as a unit called a multimer. RNA or nucleic acid tertiary structure is a three-dimensional shape of a nucleic acid polymer, and quaternary structure refers to the interactions between separate nucleic acid molecules, or between nucleic acid molecules and proteins. In this thesis, the term structure will be used for the tertiary structure if it is not stated otherwise, both for proteins and for RNAs. Figure 2.1.A shows protein primary, secondary, tertiary and quarterly structures and figure 2.1.B shows RNA primary, secondary and tertiary structures.

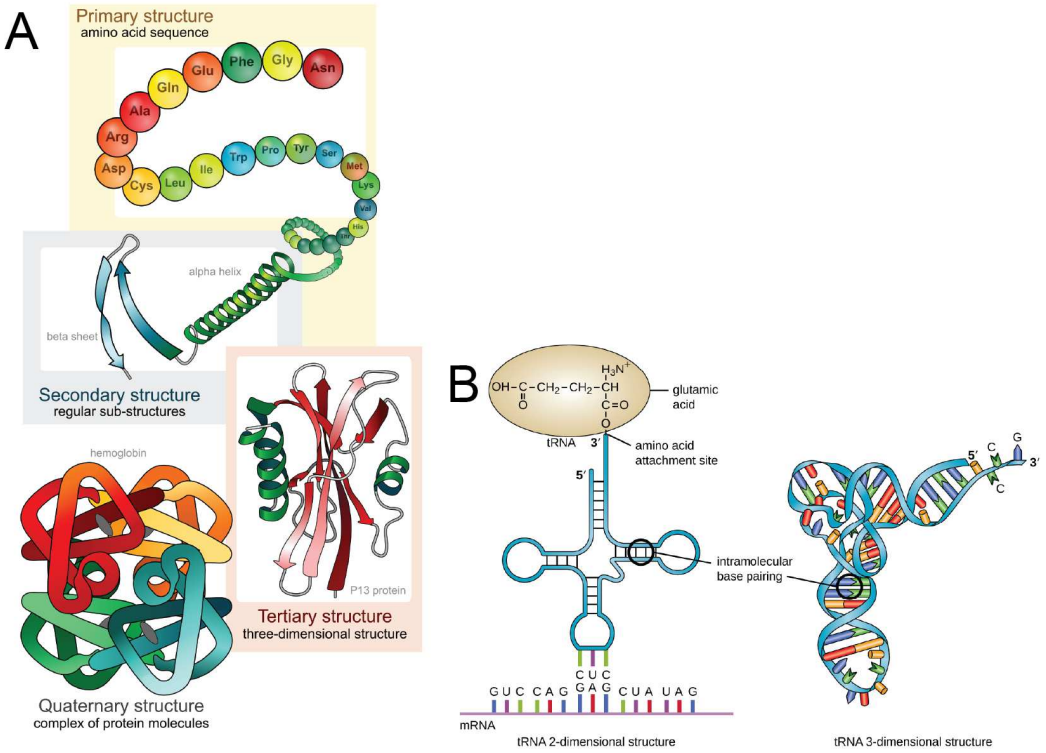

Figure 2.1: Structure types. **A**, protein structure types. **B**, RNA structure types (CNX).

2.2. Data used for training ARES

For training ARES, they used 18 RNA molecules with known structures. These structures are shown in figure 2.2. For those 18 sequences, they generated 1000 structures of each RNA using Rosetta FARFAR2 (Watkins et al., 2020) sampling method. They used 14000 generated structures for training and 4000 for validation. PDB IDs of sequences whose structures were used for validation are bolded in the table 2.1. This table also shows the lengths of these RNAs. These RNAs are short, the minimal length is 17 nucleic acids and the maximal is 46. The median of lengths is 26, while the mean is 26.944 nucleic acids and the standard deviation is 7.268. Every generated structure has its RMSD score, which is explained more thoroughly in the next section and in equation 2.1, and the distribution of RMSD scores per RNA is shown in figure 2.3.

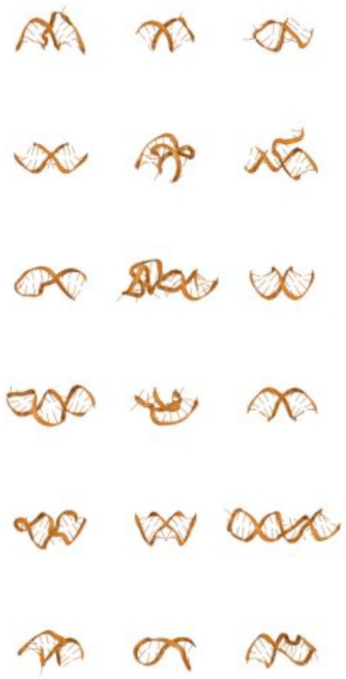

Figure 2.2: RNAs used for training from (Townshend et al., 2021)

RNA PDB ID	length	RNA PDB ID	length	RNA PDB ID	length
157D	24	1KD5	22	1XJR	46
1A4D	41	**1KKA**	17	255D	24
1CSL	28	1L2X	27	283D	24
1DQF	19	1MHK	32	28SP	28
1ESY	19	**1Q9A**	27	2A43	26
1I9X	26	1QWA	21	2F88	34

Table 2.1: Lengths of RNAs used for training

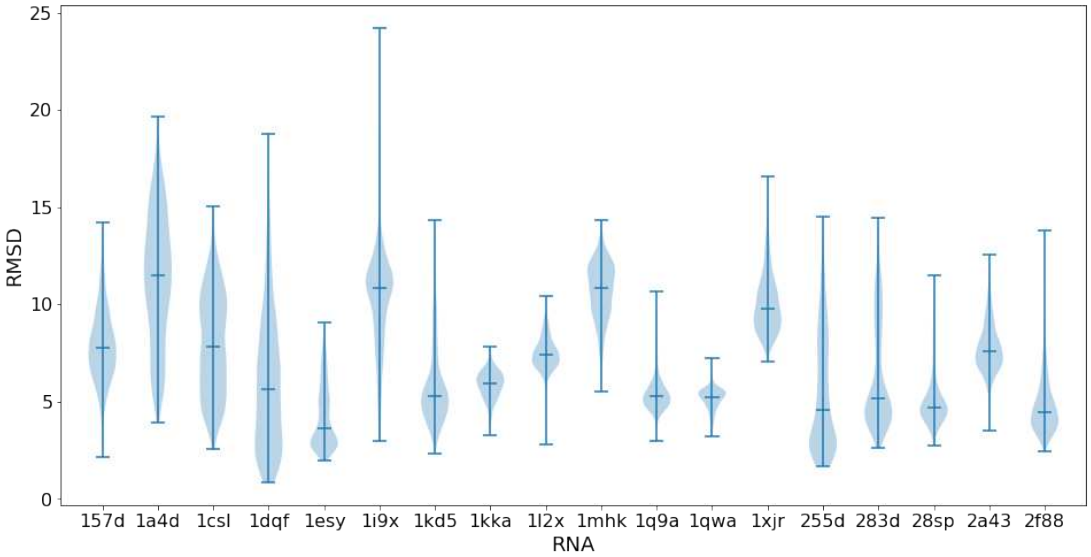

Figure 2.3: RMSD distribution per RNA used for training

2.3. Training and validation data

In the PDB every molecular model has a unique identification code (ID) consisting of four alphanumeric characters (the first one is always a number in the range 1-9). Protein chains that have as different structures as possible were chosen for the training. The results of the weekly clustering of protein sequences in the PDB by MMseqs2 at 70% sequence identity were used, downloaded from here. These files use polymer entity identifiers, instead of chain identifiers. A polymer entity can contain more than one chain, but all of the chains have the same structure and are only positioned somewhere else in space. The entity identifier consists of six characters, the first four are molecular ID, followed by an underscore and a number which is the entity identifier.

In total 50 protein chains were chosen and used, every chain from a different cluster. After choosing protein chains, the dataset was created. As the structures from the PDB are correct and their RMSD is 0, different structures for the same protein chains had to be generated. For generating protein structures Rosetta AbinitioRelax application (Raman et al., 2009; Simons et al., 1999; Bonneau et al., 2001) was used. This application predicts the 3-dimensional structure of a protein using its amino acid sequence. It creates structures which are both close and distant to the native structure. Both amino acid sequence and nucleotide sequence are written in a FASTA format. Besides amino acid sequence, the application takes as input fragment files - 3mers and 9mers.

Structural fragments are short segments of the peptide backbone, typically from 5 to 15 residues long, and do not include the side chains. If modelled directly, every protein can create an exponential number of states. In order to reduce the conformational space, fragments are used. Fragments were created, or rather picked, using the Rosetta fragment picker application (Gront et al., 2011). There are three stages in the fragment picking process: first is preparation (reading input files - FASTA file, additional files describing secondary structure which are generated with helper modules, etc), then actual fragment picking when the fragment candidates are pushed into a collector and final step is a selection when the final fragment set is prepared based on the candidates. Rosetta AbinitioRelax application needs 3mers and 9mers, therefore these fragments needed to be created. Fragment types 3mers and 9mers means that there will be blocks of 3, or 9, consecutive amino acids in every fragment. Fragment data line consists of PDB ID of the fragment origin, chain ID for the origin PDB, PDB residue number for the origin PDB, amino acid identity in the origin PDB, secondary structure for the origin PDB, 3D coordinates of the Cα atom, and ψ, ϕ and ω angles.

Rosetta AbinitioRelax application then generates a stated number of structures, which in our case was 1000 for every protein chain. An optional input file is a PDB format file that the application uses for calculating the RMSD score.

$$
\begin{aligned}
RMSD(\mathbf{v}, \mathbf{w}) &= \sqrt{\frac{1}{n}\sum_{i=1}^{n} ||v_i - w_i||^2} \\
&= \sqrt{\frac{1}{n}\sum_{i=1}^{n}((v_{ix} - w_{ix})^2 + (v_{iy} - w_{iy})^2 + (v_{iz} - w_{iz})^2)}
\end{aligned}
\tag{2.1}
$$

Root mean square deviation of atomic coordinates is calculated using the formula 2.1 where *n* is the number of atoms and vectors ***v*** and ***w*** are sets of n 3D coordinates, one for the predicted structure, one for the true structure. The score can be calculated for the backbone heavy atoms C, N, O, and Cα or sometimes just the Cα atom. Rosetta AbinitioRelax calculates Cα RMSD. The score itself is expressed in length units, which in molecular biology is the Ångström (Å) which is equal to 10^{-10} m. To compare atom coordinates, the structures first have to be aligned. A structural alignment is a form of sequence alignment based on a comparison of shapes and it attempts to establish equivalences in three-dimensional shapes of structures.

Programs for visualising structures can also be used for aligning the structures and calculating the RMSD score. One such program is PyMOL. To get Cα RMSD score,

alignment with only Cα atoms must be done and the score from cycle 0 is the desired one.

In figure 2.4 there is protein chain 1N36H, the original structure in green and one of the generated structures in purple. They were aligned and their RMSD score is 9.926.

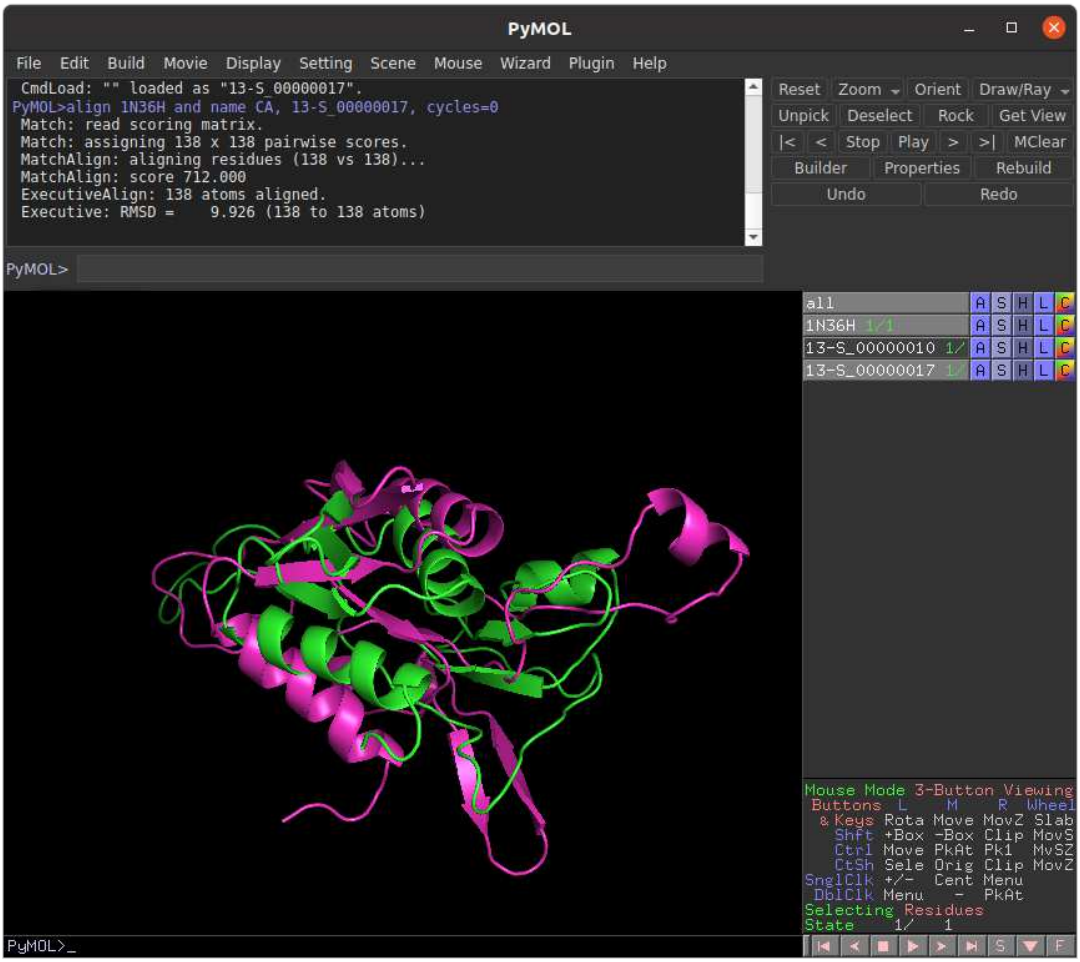

Figure 2.4: Protein chain 1N36H original and generated aligned in PyMOL

Chosen protein chains can be found in table 2.2. It lists all of the chosen protein chains with their PDB ID, entity ID, chain label, length and ID of the cluster from which they are. The selection was limited to proteins with lengths between 70 and 350 amino acids because of the complexity of longer proteins and the time and resources needed for generating structures. For the chosen proteins, the minimal length found is 76 amino acids and the maximal is 324, the median of lengths is 140 amino acids, the mean is 163.3 and the standard deviation is 66.959. PDB IDs of sequences whose structures were used for validation are in bold.

Cluster	PDB ID	Entity	Chain	Length	Cluster	PDB ID	Entity	Chain	Length
0	7S0E	2	H	223	33	1N36	6	F	101
1	4KQ4	1	L	214	34	3OTO	15	O	88
2	**3TLR**	**1**	**A**	**100**	**35**	**6SPB**	**4**	**D**	**207**
3	7KKL	2	B	118	36	1N36	8	H	138
4	6M2K	3	A	274	37	3OTO	19	S	80
5	3P44	1	A	257	38	3OTO	7	G	155
6	1GHL	1	A	130	39	3OTO	4	D	208
7	**4QCI**	**1**	**A**	**206**	40	1N36	2	B	234
8	4M8Y	1	A	100	41	1N36	13	M	118
9	1N36	12	L	124	42	3OTO	5	E	150
10	7OJX	3	C	76	43	1N36	16	P	83
11	3A7X	1	A	223	**44**	**1N36**	**9**	**I**	**127**
12	7CBT	1	A	298	45	1N36	11	K	119
13	209L	1	A	165	46	3OTO	10	J	98
14	**7LX5**	**2**	**B**	**196**	**47**	**1N36**	**20**	**T**	**99**
16	6C61	2	B	243	48	7S1K	18	R	142
17	6U42	1	1	138	49	3OTO	17	Q	104
18	6XI7	1	A	169	50	7QGH	16	P	271
20	**6MUP**	**2**	**B**	**90**	51	3WKJ	4	D	93
21	5DX4	1	A	126	52	6YHS	6	C	199
22	103M	1	A	154	54	7S1K	46	q	98
25	**2V22**	**1**	**A**	**296**	55	7JIL	45	r	113
27	2NZD	3	C	106	56	6TQO	3	C	205
30	7K4M	2	B	146	57	7S1K	7	G	150
31	**3IA3**	**2**	**B**	**135**	**58**	**7S1K**	**54**	**y**	**80**

Table 2.2: Chosen protein chains

To create the pipeline the Snakemake (Mölder et al., 2021) workflow management system was used. In this tool, the rules are defined using Python-based description language. Every rule defines how to get output files from input files which can be defined through a shell command or a Python script. Snakemake automatically determines dependencies between the rules using input and output files and builds a directed acyclic graph. When executing the workflow, it will only execute those rules for those samples for which the files don't already exist. This is especially useful when the pipeline crashes or some more samples are added. Another advantage of Snakemake is that

it scales simply from single to multicore or even to compute clusters or to the cloud, without modifications to the code. The code for the pipeline can be found on the link: https://github.com/im50603/master-thesis/tree/master.

It was important to divide files into directories so that they don't overwrite each other because some of Rosetta's modules create files with default names. Our pipeline incorporates moving the files, creating fragments files, removing unnecessary files, creating options files where parameters and paths to other input files are listed and defined, creating structures (using more seeds for every sample to increase the speed of creation as Rosetta AbinitioRelax module doesn't support multithreading), combining the structures in same files, cleaning the created structures of irrelevant atoms such as hydrogen or OXT atom which stands as a placeholder for other possible residues, and extracting the RMSD score in a separate file.

After obtaining the results with these proteins, another dataset for training containing only shorter proteins was created. Chosen proteins are listed in table 2.3. The structural models used for training were obtained in the same manner as described before, the only difference was choosing protein IDs. The PDB was searched with length and entity type filters and then only one protein from the structural group was selected as a representative. For these proteins, the minimal length is 12 amino acids and the maximal is 99, the median of lengths is 71 amino acids, the mean is 69.4 and the standard deviation is 21.835. Once again, bolded PDB IDs are the ones of the sequences whose structures were used for validation.

Cluster	PDB ID	Entity	Chain	Length	Cluster	PDB ID	Entity	Chain	Length
2	1BMG	1	A	98	**1806**	**5C02**	**1**	**A**	**25**
8	4YOB	1	A	99	1840	1B5B	1	A	94
10	1C3T	1	A	76	1864	1CNR	1	A	46
34	1AB3	1	A	88	1887	1A43	1	A	72
81	1B75	1	A	94	1915	1BT0	1	A	73
324	**7PTI**	**1**	**A**	**58**	1959	1CMG	1	A	73
449	5BMI	1	A	56	1964	6SVC	1	A	35
460	4ZYF	1	A	93	**2010**	**1AP4**	**1**	**A**	**89**
662	5AI3	1	A	54	2024	1CM3	1	A	85
817	**1A70**	**1**	**A**	**97**	2042	1A91	1	A	79
899	5DKQ	1	A	91	2127	5D8V	1	A	83
971	1BDD	1	A	60	2323	1BTB	1	A	89
982	1AEY	1	A	58	**2325**	**6SHR**	**1**	**A**	**64**
1002	**1A32**	**1**	**A**	**85**	2329	1BH0	1	A	29
1131	**1AME**	**1**	**A**	**66**	2361	1AE2	1	A	86
1296	6QJK	1	A	97	2396	5EBX	1	A	62
1299	1AHO	1	A	64	2442	5DSU	1	A	75
1346	**7O2K**	**1**	**A**	**60**	2455	1CNL	1	A	12
1371	1CIS	1	A	66	2647	1B6Q	1	A	56
1384	1AG6	1	A	99	2679	5CN0	1	A	70
1536	1BQT	1	A	70	**2707**	**4XDX**	**1**	**A**	**70**
1596	**1C54**	**1**	**A**	**96**	2794	4YDX	1	A	67
1602	1COI	1	A	29	2799	1BO0	1	A	76
1729	1B5M	1	A	84	2983	1AZH	1	A	36
1788	1ABQ	1	A	56	2992	7RN3	1	A	30

Table 2.3: Chosen short protein chains

2.4. Evaluation data

To assess ARES's performances, Benchmark 1 from (Townshend et al., 2021) was used. It includes all RNAs from the RNA-Puzzles blind RNA structure prediction challenge for which experimentally determined structures were published between 2010 and 2017, meaning the first 21 puzzles, except for Puzzle 16, but Puzzle 14 is in two forms - bounded and free form, which is labelled 14b and 14f. For every puzzle, there are three types of structural models: decoys, near-native models and a native structural

model. A native structure is the real structure, obtained experimentally. Figure 2.5 shows the native structures. Decoys are structural models taken from the FARFAR2-Puzzles dataset (Watkins et al., 2020). Near-native structures were created in the same manner as decoys, using the FARFAR2 tool, with energetic restraint. For each puzzle, the number of near-native structures created is 1% of the number of decoys. Table 2.4 shows how many structures are being evaluated for each puzzle, and figure 2.6 shows the distribution of RMSD values of structures for each puzzle.

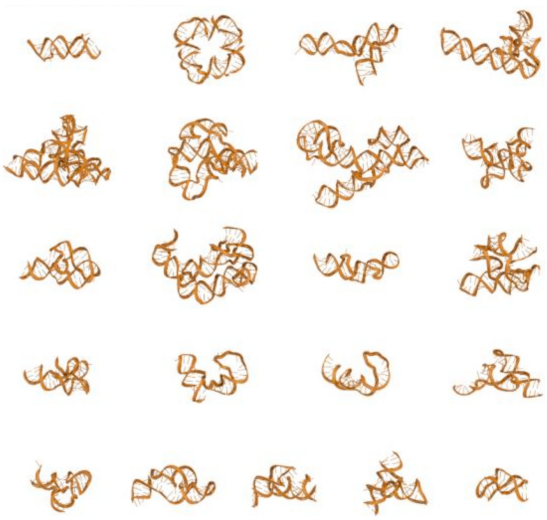

Figure 2.5: RNAs from Benchmark 1 from (Townshend et al., 2021)

RNA Puzzle	decoys	near-natives	native	total
1	14052	141	1	14194
2	20000	200	1	20201
3	33442	334	1	33777
4	5766	58	1	5825
5	24901	249	1	25151
6	28859	289	1	29149
7	7963	80	1	8044
8	33086	331	1	33418
9	18660	187	1	18848
10	5873	59	1	5933
11	41952	420	1	42373
12	35506	355	1	35862
13	20296	203	1	20500
14b	24531	0	1	24532
14f	15112	0	1	15113
15	6123	61	1	6185
17	16529	165	1	16695
18	17091	171	1	17263
19	4499	45	1	4545
20	1547	15	1	1563
21	48146	481	1	48628

Table 2.4: Number of structural models in Benchmark 1

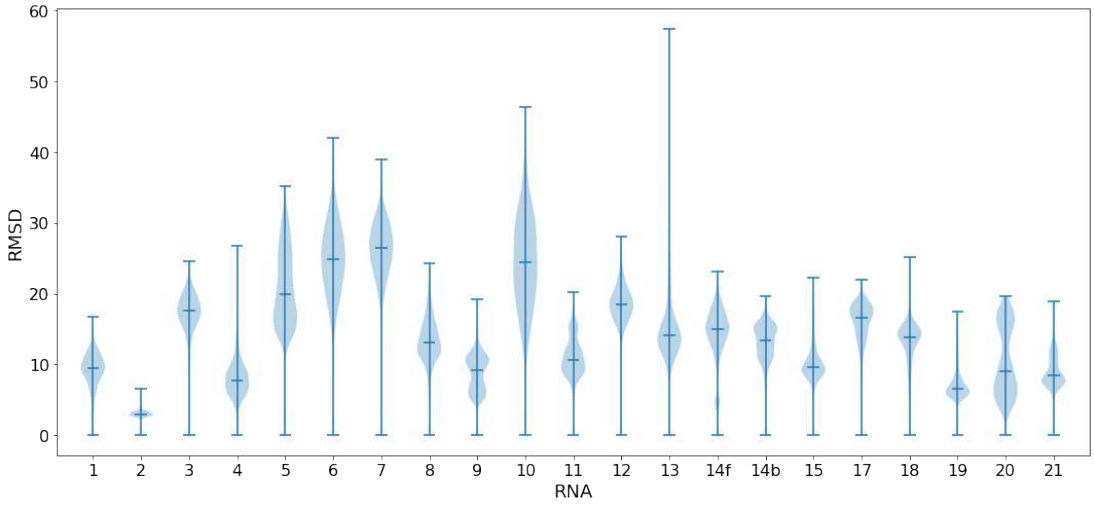

Figure 2.6: RMSD distribution per RNA Puzzle from evaluation dataset

3. Network Architecture

The neural network used in this thesis is the one from (Townshend et al., 2021). Figure 3.1 shows the network architecture, and the idea of each layer is explained later in the text. The number after the name of the layer is the number of channels that this layer has. The goal of this network design is to recognize specific atom arrangements in three-dimensional space or geometric patterns called structural motifs. Since the same motif can be rotated or translated in space, the model's key property is equivariance. A network layer is equivariant to a transformation (in this case rotation or translation) if the transformation of its input leads to a corresponding transformation of its output. The initial layers recognize instances of the structural motifs and accumulate local information while preserving information about the orientation and position of inputs. The later layers collect the information from previous layers and get the global picture. Furthermore, this also means that the network recognizes that finer-scale motifs, like base pairs, assemble into coarser-scale motifs, such as helices. This is enabled by another important property, invariance. Invariance means that rotation and translation of input don't lead to output transformation.

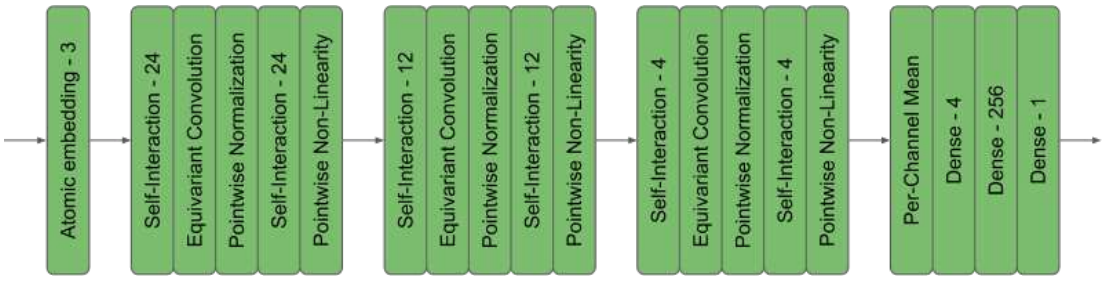

Figure 3.1: Network architecture

3.1. Atomic Embedding

The first layer in the network is atomic embedding. As equivariant convolution takes 3D coordinates of atoms with associated features for each atom as input, this layer

a, b - certain atom $r_{ab} = \|\vec{r_{ab}}\|$ - length of the vector between atoms a and b
l - angular order, $l \in \{0, 1, ..., L\}$ L - maximum order used, $L = 2$ m - angular index, $m \in \{-l, -l+1, ..., l\}$ E - dimension of the equivariant convolution c - radial index, $c \in \{0, 1, ..., E\}$
V_{acm}^{l} - feature associated with atom a, for certain lcm group
in 3.5: σ = 1 n = 11 $\mu_j = \frac{12}{11} j$
C - Clebsch-Gordan coefficients $C \neq 0$ for $\|l_i - l_f\| \leq l_o \leq l_i + l_f$
b^l - learnable scalar bias term (one per order)
d - new radial index
b_d - learnable bias term
W_{cd} - learnable weight matric

Table 3.1: Labels and indices used in equations

generates the initial features, which are the encodings of the element types. The layer uses one hot encoding for carbon, nitrogen and oxygen, while other element types are not used. Features are stored in variable V_{acm}^{l}, the indices are explained in table 3.1 and later in the text and the encoding is shown in 3.1. As the encoding shows, there are 3 dimensions and channels used, currently $c \in 0, 1, 2$.

3.2. Equivariant Convolution

The most important layers in the model are equivariant convolution layers. Features which are commonly used when predicting protein structure are element type, surface shape, and the spatial distribution of charge and potential. In these layers, features are

updated repeatedly by sharing the information locally, between neighbouring atoms. The equivariant convolution is based on filter functions being applied one at a time to each atom in the source atom's local neighbourhood. The filter functions take only the 3D vector between the two atoms as input and their output is combined with the current features of this source atom to produce updated features. The crucial task for ensuing equivariance and orientation preservation is choosing the filter functions and how the outputs are combined with features. There are two types of functions used: radial and angular functions, in this case real spherical harmonics. The final output of the filter function is equation 3.2, where R_c is the radial function and Y_m^l is the angular function. This choice of the filter function is valid because any function can be written in spherical coordinates as a linear combination of spherical harmonics, as shown in 3.3. In this equation, $\sum_{nlm}^{N}$ is a short form for triple sum, $\sum_{nlm}^{N} = \sum_{n=1}^{N} \sum_{l=0}^{n-1} \sum_{m=-l}^{l}$, a_{nlm} are the coefficients which have to be determined, and R_{nl} and Y_m^l are radial functions and real spherical harmonics. N is the number of harmonics used and if it approaches infinity, the result is the exact function. In most cases, it is enough to have the approximation and use $N \geq 25$. The indices l and m indicate the degree and order of the function. The label l is called a degree because spherical harmonics take their simplest form in the Cartesian coordinate system as homogenous polynomials of degree l in (x, y, z). The order m is associated with the Legendre polynomial P_l^m, which is the result of solving Laplace's equation in spherical coordinates.

Spherical harmonics, more precisely Laplace's spherical harmonics, are often used to describe the aforementioned features because of their properties. The first property is that Laplace's spherical harmonics are a special set of spherical harmonics which form a complete set of orthogonal functions. That means that any function defined on a surface of a sphere can be written as a linear combination of these spherical harmonics. Another property is that the space of spherical harmonics is closed with respect to rotations and they can be transformed between themselves in a predictable manner by rotation. That means that, in this case, proteins can be rotated just by transforming coefficients in the series.

Spherical harmonics are functions which assign values from the unit sphere to the set of real numbers, hence there are multiple ways to visualize them. Figure 3.2.A shows spherical representations where the colour of the unit sphere corresponds to the value of Y_l^m. In Figure 3.2.B you can see the orbital representation, where for every point of the sphere the radial component is added - the distance of the surface from the origin is the absolute value of the harmonic. The colours green and yellow represent regions where the function is positive or negative, respectively. In both figures, the

first row shows a spherical harmonic with a degree $l = 0$ and order $m = 0$. For each next row, the degree increments by one. The middle column has $m = 0$, the right columns are positive values, also incremented by one, and the left ones are negative order values, decremented by one.

In 3.2 degree l is called angular order and m is called angular index and it determines which harmonic in the group is referred to. The maximum order used is $L = 2$, and since there are $2l + 1$ harmonics per group (the harmonics are grouped by l), there are $M = \sum_{l=0}^{L}(2l + 1)$ angular functions in total.

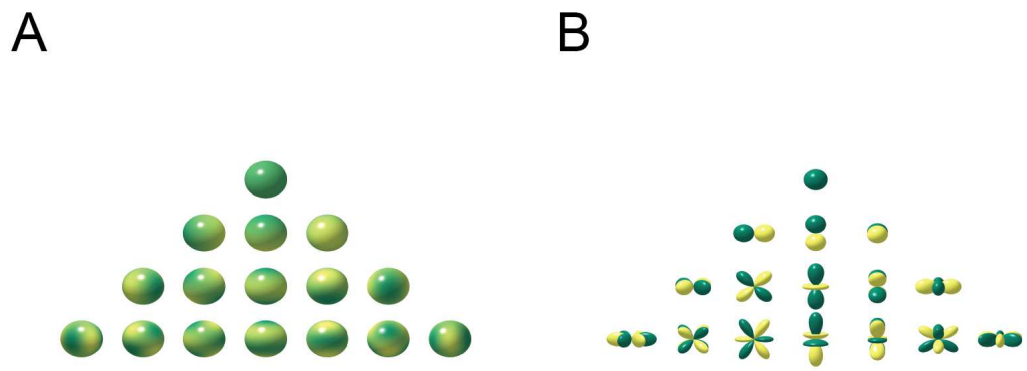

Figure 3.2: Spherical harmonics. **A**, spherical representation. **B**, orbital representation.

Since spherical harmonics can be used to parametrize only the functions on the surface of a sphere, they are often combined with radial functions which can be used for any 3D function, as shown in filter function 3.3. There are many radial functions that can be used, but in (Townshend et al., 2021) the radial functions are implemented with a dense neural network. For every neighbouring pair, a filter bank of Gaussians is applied to the magnitude of the vector between them $r_{ab}^{\rightarrow}$. The neural network has three layers, the input layer where these Gaussians are used, after which there is one hidden layer with a dimension of 12 and then the output layer with E scalar radial filter outputs, as shown in equations 3.5 and 3.4. Parameter E is called the dimension of the equivariant convolution and for three convolutions in the network, its value is 24, 12 and 4. In equation 3.2, c denotes which output is referred, $c \in 0, 1, \ldots, E$ and initial features for each atom has a total of $E = 3$ radial features for three element types. These functions don't take the orientation of relationships between atoms into consideration, they consider only the distances which makes them invariant to translations and rotations. The orientations are covered with angular functions, which means each

$$V^0_{a00} = 1 \text{ if atom } a \text{ is C}$$ $$V^0_{a10} = 1 \text{ if atom } a \text{ is O}$$ $$V^0_{a20} = 1 \text{ if atom } a \text{ is N} \quad (3.1)$$
$$F^l_{cm}(\vec{r_{ab}}) = R_c(r_{ab})Y^l_m(\hat{r_{ab}}) \quad (3.2)$$
$$A(\vec{r}) = \sum_{nlm}^{N} a_{nlm} R_{nl}(r) Y^l_m(\theta, \phi); N \geq n > l \geq |m| \geq 0 \quad (3.3)$$
$$[R_0(r_{ab}), R_1(r_{ab}), ..., R_E(r_{ab})] = Dense([G_0(r_{ab}), G_1(r_{ab}), ..., G_n(r_{ab})]) \quad (3.4)$$
$$G_j(r_{ab}) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(r_{ab}-\mu_j)^2}{2\sigma^2}} \quad (3.5)$$
$$L^{l_o}_{acm_o}(V^{l_i}_{acm_i}) = \sum_{m_i, m_f} C^{(l_o,m_o)}_{(l_f,m_f),(l_i,m_i)} \sum_{b \in neighbors(a)} F^{l_f}_{cm_f}(\vec{r_{ab}}) V^{l_i}_{bcm_i} \quad (3.6)$$

Table 3.2: Formulas used in the first few layers

filter function is equivariant to rotations within an angular order l.

Equation 3.6 shows how filter functions are combined with features V. All indices and labels are specified in table 3.1, only subscripts f, i and o are added, and they stand for the filter, input and output features, respectively. In cases when l_f and l_i yield outputs of the same angular order l_o, new values of V are calculated for each combination and outputs are concatenated across the c dimension. The neighbouring function, and thus the point convolution, is restricted to the $K = 50$ nearest neighbours (K includes the source atom itself) because laws of physics regarding intermolecular and intramolecular interactions are local.

3.3. Pointwise Normalization

The equation for pointwise normalization is given in equation 3.7, where indices mean the same as before, and are given in table 3.1. The normalization is calculated sep-

arately on each atom's features, as one can see in the equation. Practice shows that neural networks learn more easily with normalized data so the choice of such layers is valid.

3.4. Pointwise Non-linearity

As the name itself says, pointwise non-linearity is also calculated separately on each atom's features. The equation 3.8 shows how it is calculated, with an explanation for η function in 3.9. Without non-linearity in a neural network, it would be impossible to model most real-world data, due to the fact that most of the relationships are non-linear. The chosen non-linearity η is called shifted softplus, and the shifting ensures that $\eta(0) = 0$ and improves the convergence of the network. It shows similarity to ELUs, while having infinite order of continuity.

3.5. Pointwise Self-Interaction

Pointwise self-interaction is used to mix information across radial channels between equivariant convolution layers in order to update features, and the equation is shown in 3.10. In addition to the old indices, a new radial index d is defined. The number of these output radial indices is called the dimension of the pointwise self-interaction and it is the same for each angular order l of spherical harmonics within the self-interaction layer. The dimensions in the model are 24, 24, 12, 12, 4, and 4, respectively. The bias term is used only for angular order 0.

3.6. Per-Channel Mean

Equation 3.11 states that this layer averages features corresponding to the zeroth-order harmonic across all atoms. Per-channel mean comes after all equivariant convolutions, so positions of atoms and all of the features that are not related to zeroth-order harmonic are not important anymore and are dropped in this layer. The zeroth-order harmonic is invariant to rotations, therefore all further layers will be invariant to rotations. New features E are indexed only by the radial index c and it represents a molecule-wide embedding that is insensitive to the original RNA's size.

$$N(V^l_{acm}) = \frac{V^l_{acm}}{\sqrt{\sum_{c,m}(V^l_{acm})^2}}$$	(3.7)
$$P(V^l_{acm}) = \begin{cases} \eta(V^l_{acm}), & \text{if } l = 0 \\ V^l_{acm} \cdot \eta(\sqrt{(\sum_m (V^l_{acm})^2)} + b^l), & \text{otherwise} \end{cases}$$	(3.8)
$$\eta(x) = ln(0.5e^x + 0.5)$$	(3.9)
$$S^l_{adm}(V^l_{acm}) = b_d + \sum_c V^l_{acm} W_{cd}$$	(3.10)
$$E_c = \sum_a V^0_{ac0}$$	(3.11)
$$E_d = b_d + \sum_c W_{cd} E_c$$	(3.12)

Table 3.3: Formulas used in other network layers

3.7. Dense Layers

Dense layers are regular fully connected neural network layers, with standard equation 3.12.

3.8. Parametrization and initialization

From figure 3.1 one can see that this model is not a big model, only 20 layers, where 5 layers have fixed parameters - pointwise normalizations, per-channel mean and atomic embedding. There are 18.1k parameters in total. All biases were initialized to 0, and other parameters were initialized using Xavier uniform initialization. Only the first fully connected layer uses an ELU non-linearity, while the other two layers don't use non-linearity.

3.9. Training

The loss used for training was Huber loss, as a function of the difference between the predicted and true root mean square deviation. Equation 3.13 shows how the loss is calculated. In this equation, a is the difference between the predicted and true value and δ is a function parameter.

$$L_\delta(a) = \begin{cases} \frac{1}{2}a^2, & \text{for } |a| \leq \delta \\ \delta \cdot (|a| - \frac{1}{2}\delta), & \text{otherwise} \end{cases} \tag{3.13}$$

Figure 3.3 shows the Huber loss function ($\delta = 1$) and squared loss in green and blue, respectively. The Huber loss function is quadratic for smaller a values and linear for larger ones. It was introduced to solve squared loss' biggest disadvantage - it tends to be dominated by the outliers, since the Huber loss is less sensitive to them.

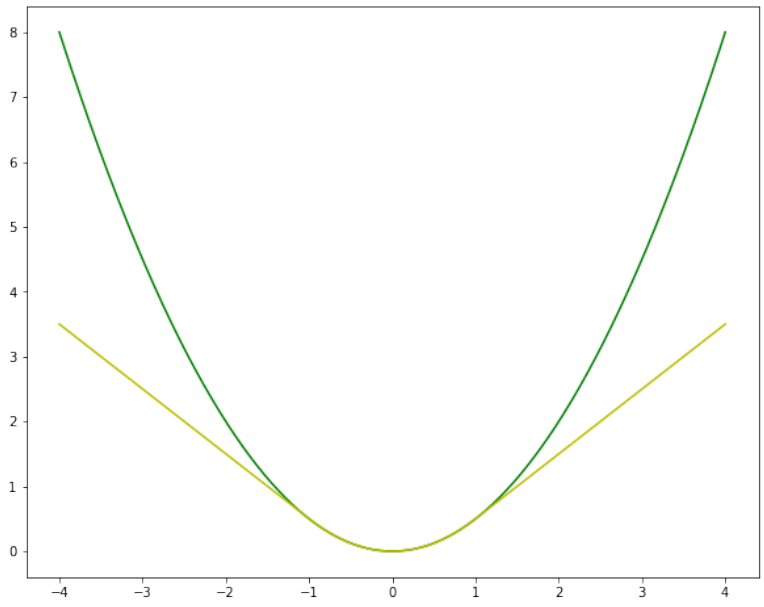

Figure 3.3: Huber loss

3.10. Evaluation

Even though when training the model predicted and true RMSD are compared, the goal of this model is actually to get a new scoring function for RNA structures so evaluation of the models was not done by comparing the exact scores. Instead, in order to see how good a model performs, the top 1, 10 and 100 best structural models according to the given score were studied. In chosen groups, the structural model with

the best true RMSD score was found and this RMSD was marked in the table. It is not important whether the model gives the exact RMSD or not, it is more important that it can differentiate good structural models from bad ones.

4. Results

4.1. Choosing hyperparameters

Authors from (Townshend et al., 2021) tried out 100 combinations of hyperparameters, so there was no need for a thorough analysis of hyperparameters, but since the loss function was unstable, a few additional tests were done. The first test was done to see if the change in batch size would influence the number and amplitude of spikes. Figure 4.1 shows the loss when the model is trained with batch sizes 8 and 16, while all the other parameters remain unchanged. The opaque lines show a smoothened loss, while the more transparent colours show the real values. Because the batch size is different, the number of steps on the x-axis is not optimal, therefore figure 4.2 shows the same losses with a number of epochs on the x-axis.

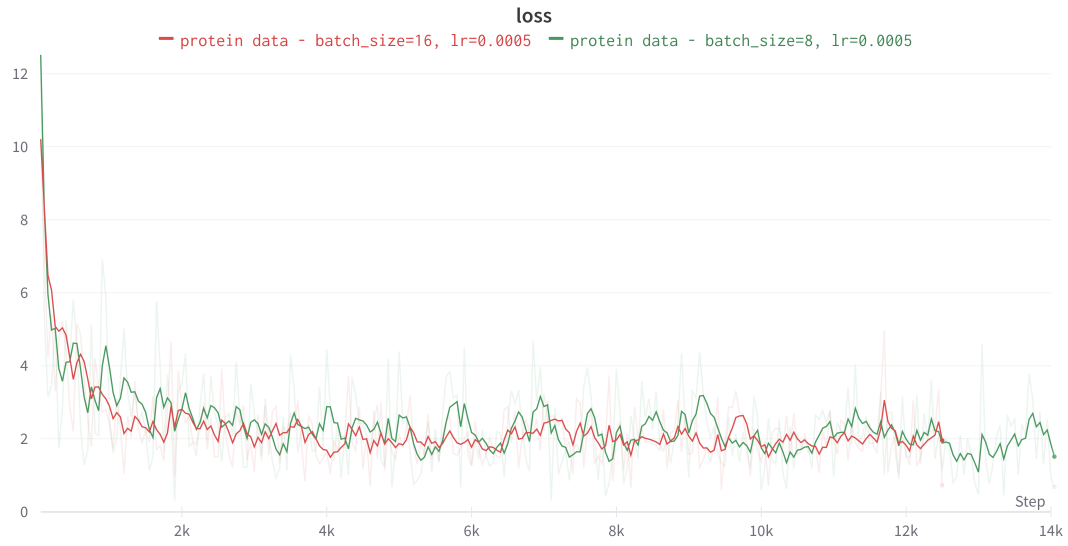

Figure 4.1: Hyperparameters test - batch size

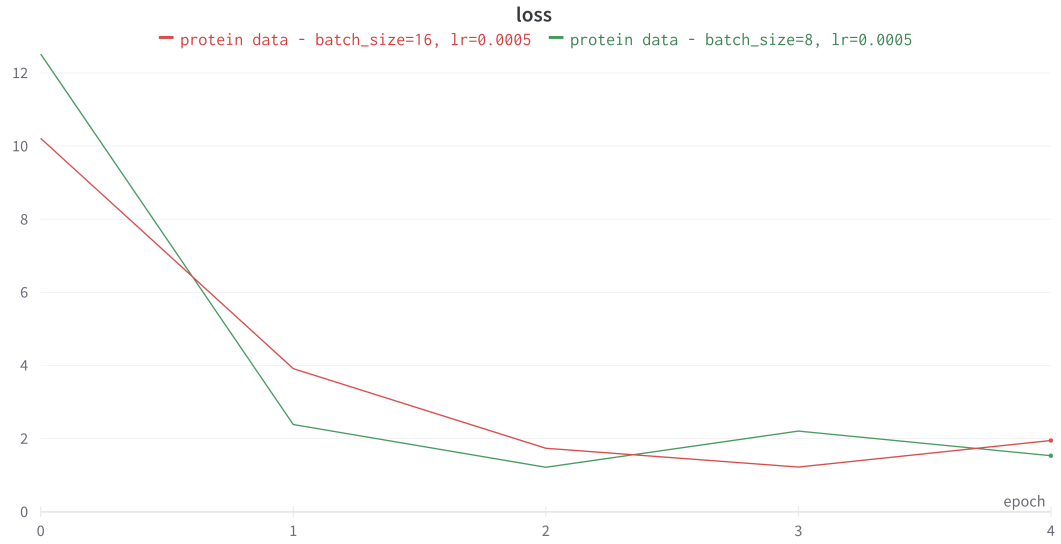

Figure 4.2: Hyperparameters test - batch size (over epochs)

Another test was to see how the learning rate affects the loss function. Figure 4.3 shows the loss functions when the model is trained with learning rates 0.005 and 0.00005, while the other hyperparameters remain the same, with batch size set to 8. The graph shows that the smaller learning rate causes the model to learn slower.

Figure 4.3: Hyperparameters test - learning rate

The final comparison, shown in Figure 4.4, was between the hyperparameters used in (Townshend et al., 2021) - learning rate of 0.01 and batch size 16 and values used in previous comparisons - learning rate 0.005 and batch size 16. The loss had the smallest values when using hyperparameters from (Townshend et al., 2021). Consequently, all

of the models mentioned further in the text are trained using hyperparameters from (Townshend et al., 2021).

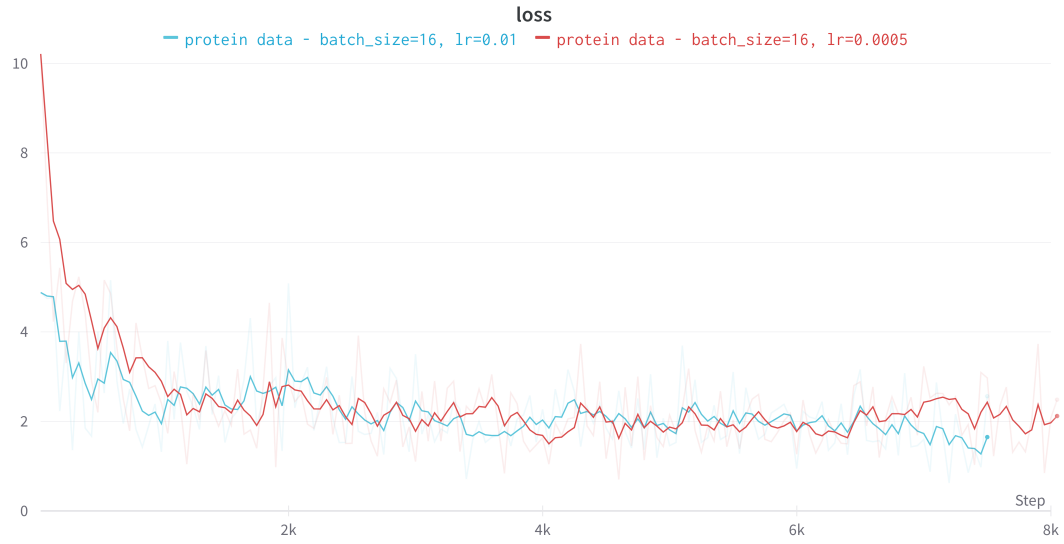

Figure 4.4: Hyperparameters test - final

4.2. Training results

Figures 4.5.A and 4.5.B show the loss function. The purple line represents the loss when the model was trained with the original RNA data - the same structural models for the same 18 RNAs as in (Townshend et al., 2021). The turquoise line refers to the loss of the model trained with protein data and we can see that the values are higher than in the case of training with RNA. Because of this, the model was trained with both protein and RNA data - shown using the red line in the figures. This did not reduce the loss a lot so we tried to fine-tune the model trained on protein data with RNA data. The loss for this is shown with the green line and it seems to be even better than when the model is trained with RNA data only. The difference between the figures is in the fact that 4.5.A has the number of steps on the x-axis, which is not the best option because there is a different number of samples, and therefore a different number of steps. In contrast,4.5.B has the number of epochs on the x-axis, but does not provide the same amount of information as 4.5.A. Figures 4.5.C and 4.5.D show the validation loss for the same training runs, once again with steps and epochs on the x-axis. The validation intervals were not the same for each training which can be seen on graph 4.5.C.

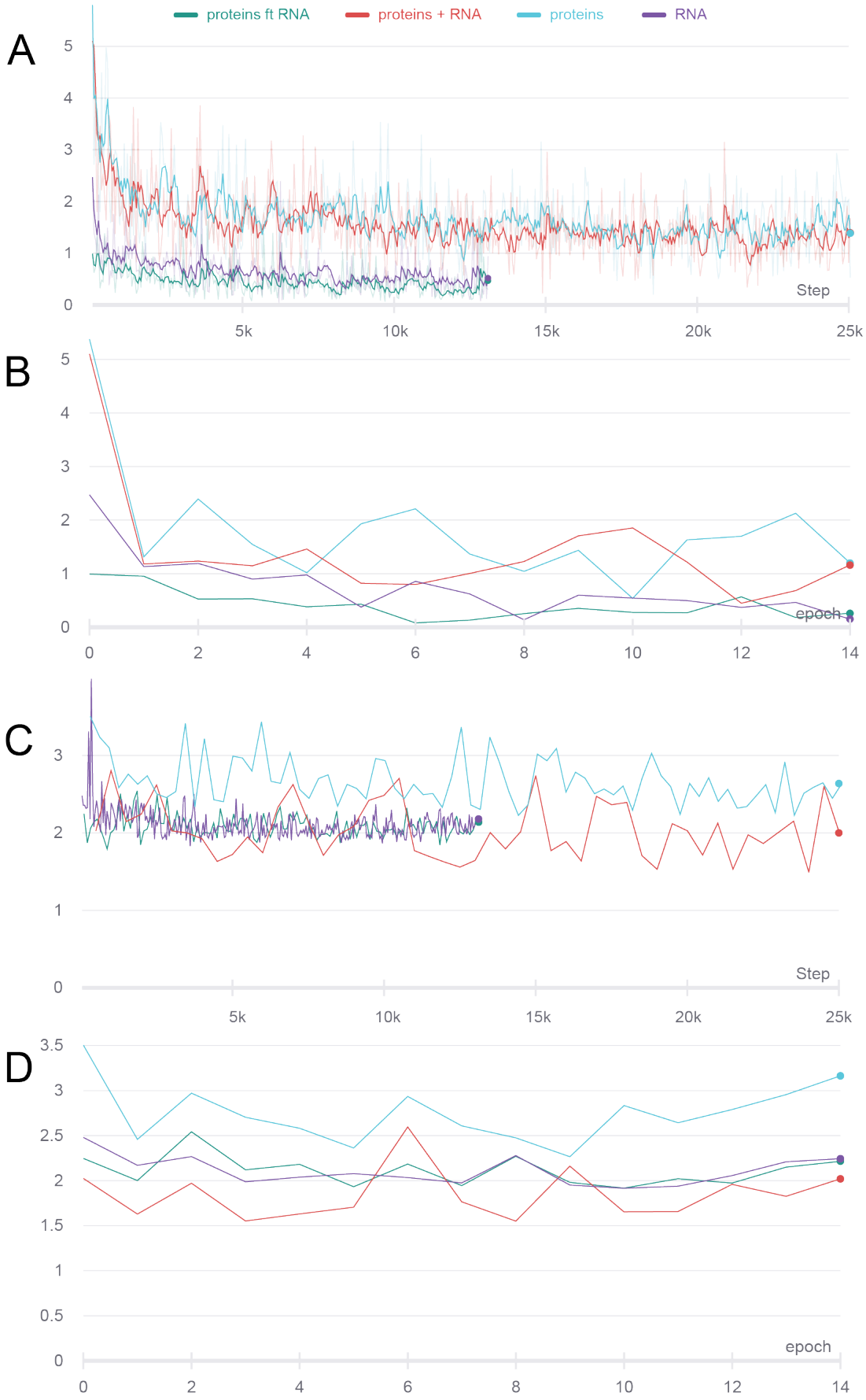

Figure 4.5: Loss functions - training on proteins. **A** and **B**, training loss. **C** and **D** validation loss.

It is clear that the loss takes on higher values when the model is trained on proteins than when trained on original RNAs. A possible explanation could be the fact that the proteins are much longer than the original RNAs. Because of this, we decided to try to train the model with shorter proteins, on the other generated dataset.

Figure 4.6 shows the loss functions when the model was trained with longer proteins, with short proteins and with original RNAs. It is clear that when training with short proteins the loss is significantly lower than when training with longer proteins.

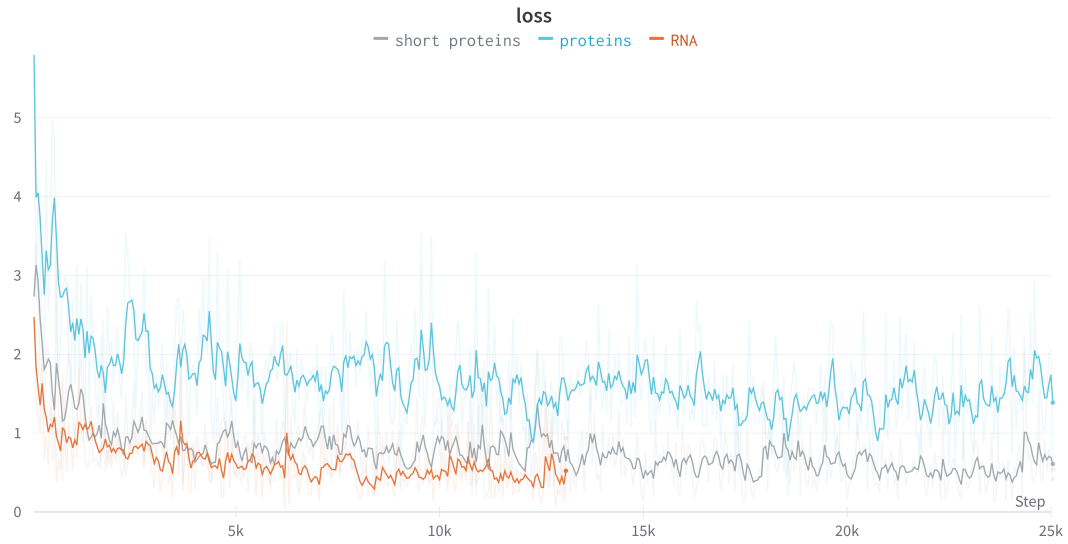

Figure 4.6: Loss functions - short proteins vs proteins

Since the loss had lower values, we tried to train using the combination of short proteins and RNAs and to fine-tune the model trained on short proteins with RNAs. The loss functions can be seen in Figures 4.7.A and 4.7.B, and validation loss functions in Figures 4.7.C and 4.7.D. As in the previous figures, there is the loss function with steps and epochs on the x-axis because of the difference in the number of samples, where the one with steps on the x-axis is trimmed on the right side. Also, the loss function in 4.7.A is smoothened.

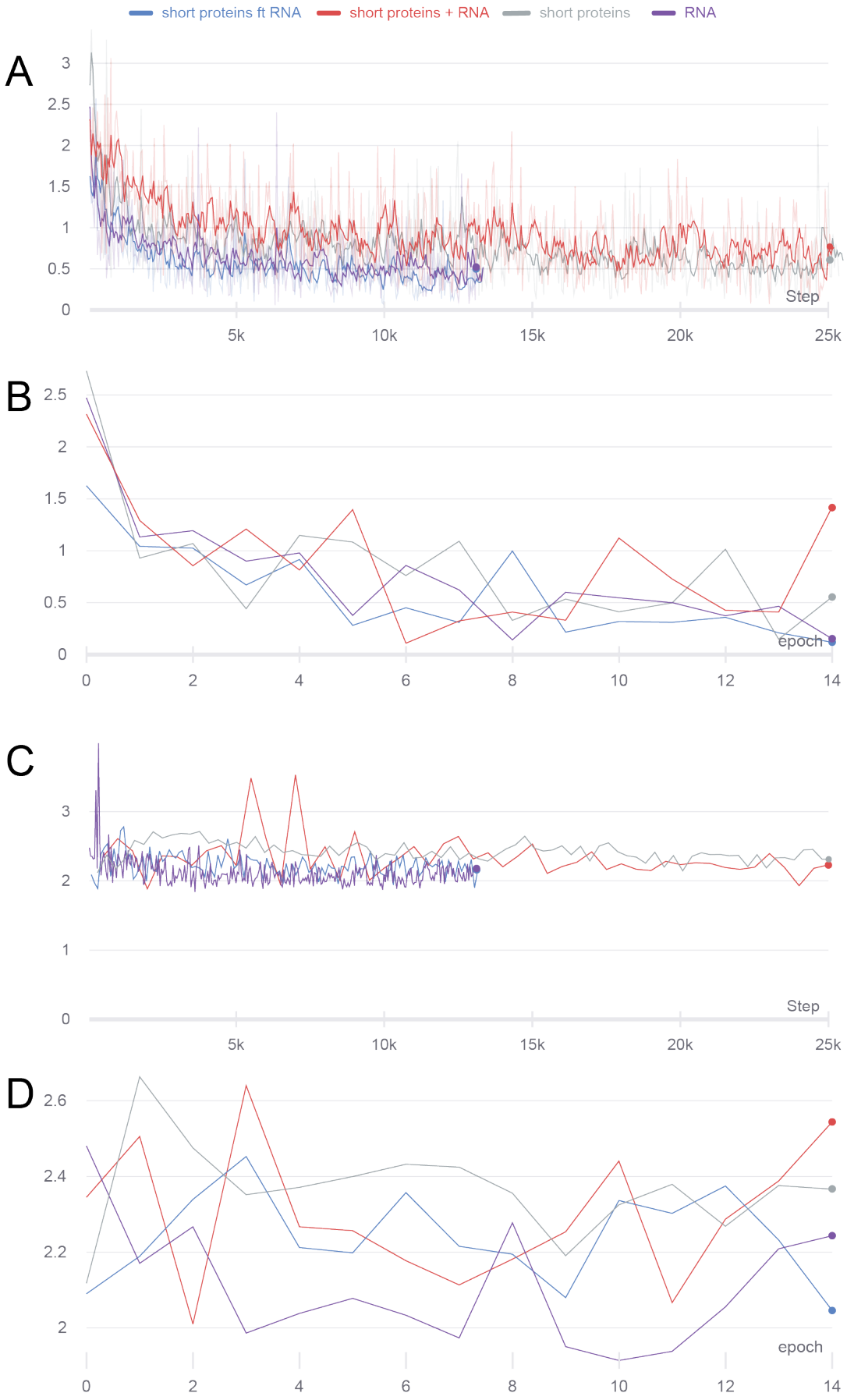

Figure 4.7: Loss functions - training on short proteins. **A** and **B**, training loss. **C** and **D** validation loss.

Lastly, we decided to combine shorter and longer proteins. Figure 4.8.A shows the loss function for the model trained with all of the proteins, the model trained with all proteins and RNAs combined and the model trained with all of the proteins and then fine-tuned with RNA data. This loss is smoothened and trimmed, as there is a larger number of steps for models trained with all of the proteins because of the larger dataset. Figure 4.8.B is the same loss, but on the x-axis, there is a number of epochs instead of a number of steps. Figures 4.8.C and 4.8.D show the validation loss, with number of steps and number of epochs on the x-axis, respectively. Once again, the validation interval on figure 4.8.C was not the same for every model.

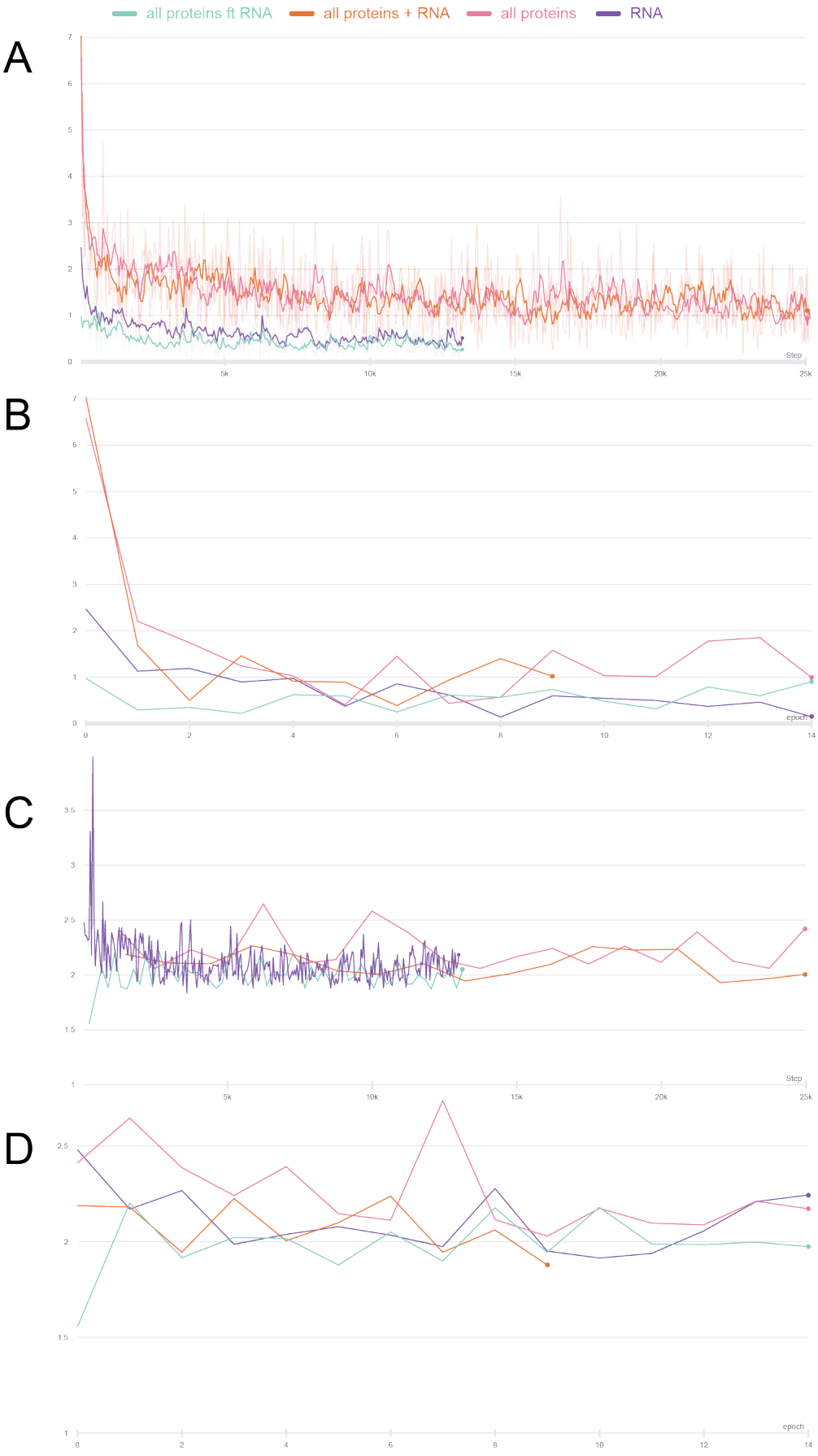

Figure 4.8: Loss functions - training on all proteins. **A** and **B**, training loss. **C** and **D** validation loss.

4.3. Evaluation results

Figure 4.9 shows evaluation results. In (Townshend et al., 2021), authors have uploaded their evaluation results and these are labelled as ARES. One can observe that this table is not the same as the one in the original paper. The tables differ for Puzzles 14b and 14f, and this is because the authors haven't published the near-native structural models for these Puzzles, so there are no good structural models to be scored well and found in the top 1, 10 and 100. Even though the same model was trained with the same data and the same hyperparameters, the model didn't learn as much. The model that we have trained is in the right part of the figure, labelled as RNA, and this model will be used as the benchmark further in the text.

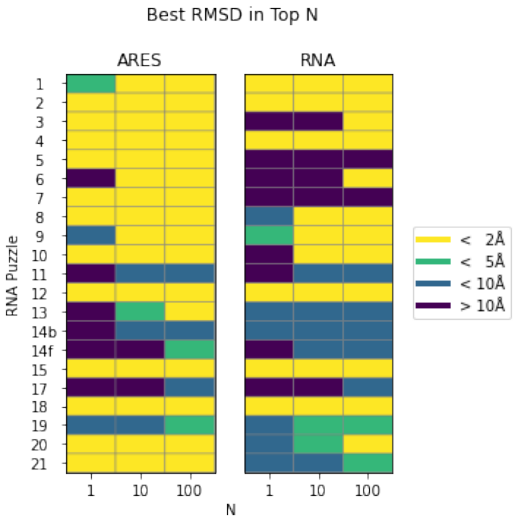

Figure 4.9: Evaluation results - comparison with (Townshend et al., 2021)

If we observe each cell in the tables, we can see that out of 63 cells in total, the model that we have trained performed better for 6 cells, while ARES was better for 19 cells. This reflects that the model that we have trained on RNA data is better for Puzzles 1, 9, 14b and 19, but worse for Puzzles 3, 5, 6, 7, 8, 10, 13, 20 and 21, meaning better for 4 puzzles and worse for 9.

Figure 4.10 shows evaluation results for models trained on longer proteins and protein-RNA combinations.

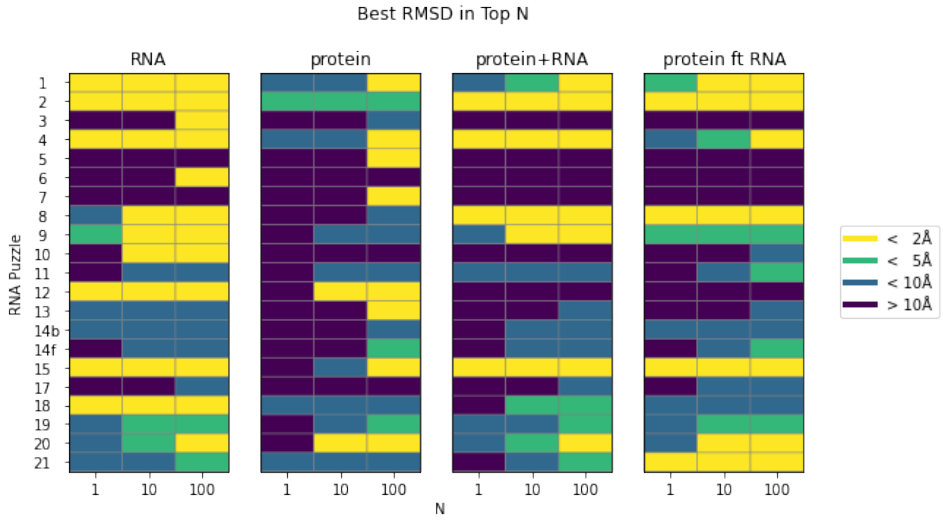

Figure 4.10: Evaluation results - models trained on proteins

The model trained with only protein data is much worse than the one trained with RNA. It is worse in 33 out of 63 cells, and better in 5 cells. This manifests in a way that it is better for Puzzles 5 and 7, the same for Puzzle 11 and maybe Puzzle 20, and it is worse for all the other puzzles. The model trained with both proteins and RNA is better than the model trained only on proteins, but worse than the one trained only on RNA data. In comparison to the model trained on RNAs, it is worse in 18 cells and better in 2 cells. It improved the results for Puzzles 8 and 11, it is the same for Puzzles 2, 4, 5, 7, 14f, 15, 17 and 20 and worse for the rest. In total, better for two puzzles, the same for 8 and worse for 11 puzzles. The model trained on proteins and then fine-tuned with RNA is even better than the one trained on proteins and RNA combined. If compared to the model trained on RNA data, it performs worse in 17 cells and better in 8 cells. It is better for Puzzles 8, 11, 14f, 17, 20 and 21, and the same for Puzzles 2, 5, 7, 14b, 15 and 19. This means it is better for 6, the same for 6 and worse for 9 puzzles.

Figure 4.11 shows evaluation results for models trained on short proteins and short protein-RNA combinations.

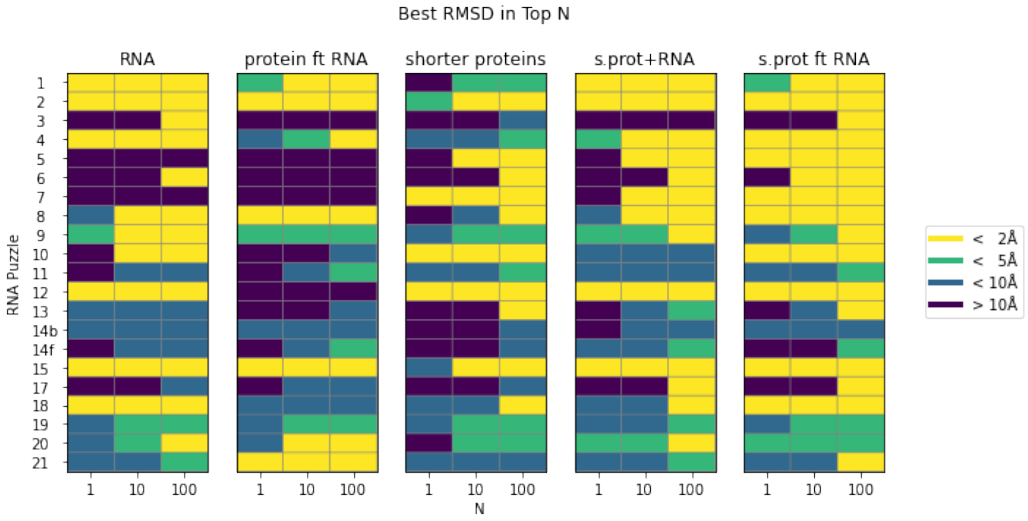

Figure 4.11: Evaluation results - models trained on short proteins

The model trained only on short proteins is better than the one trained on longer proteins, but worse than one trained on proteins and fine-tuned on RNA. If we compare it to the model trained on the original RNAs it is worse in 24 cells and better in 9 cells. This means it is better for Puzzles 5, 7, 10 and 11, and the same for Puzzles 6, 12, 17 and 19, in total it improved for 4 puzzles, stayed the same for 4 puzzles and it deteriorated for 13 puzzles.

The model trained on both short proteins and RNAs is the first model that is better than the one trained only on RNAs. It is better in 11 cells and worse in 10 cells. It is better for Puzzles 5, 7, 11, 14f, 17 and 20, for Puzzle 13 it got one cell better and one worse, and it is worse for Puzzles 3, 4, 9, 10, 14b, 18 and 19. In total it is better for 7 puzzles, worse for 7 puzzles and the same for 7 puzzles.

The model trained on short proteins and fine-tuned on RNA gives the best results. It performs better in 16 cells and worse in 6 cells. It works better for Puzzles 5, 6, 7, 8, 10, 11, 17 and 21, worse for Puzzles 1 and 9, and Puzzles 13, 14f and 20 have the case where it has one cell better and one worse. If we consider the aforementioned case staying the same, then this model performs better for 8 puzzles, worse for 2 puzzles and stays the same for 11 puzzles in total.

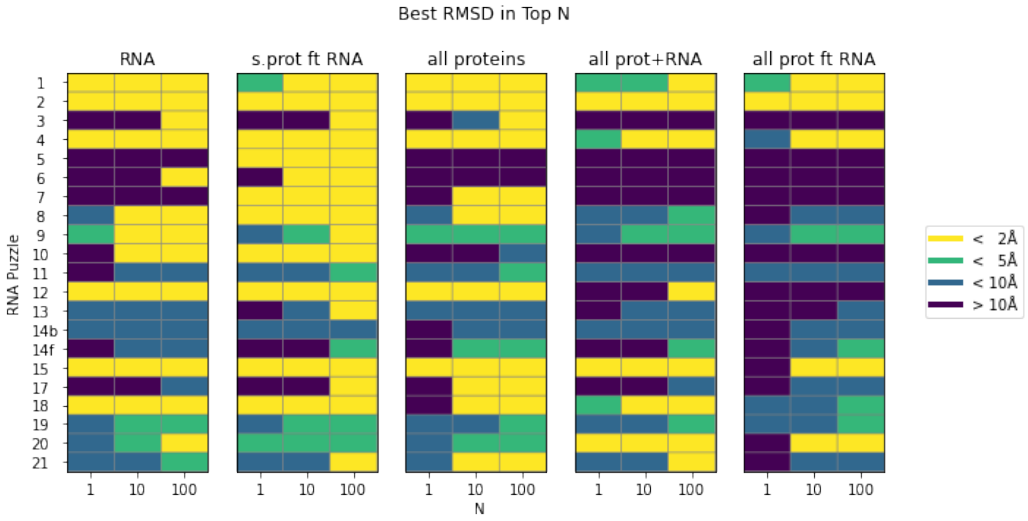

Figure 4.12: Evaluation results - models trained on all proteins

When we combined longer and shorter proteins and trained the model on the combined protein data, the evaluation results were better than when training only on RNA data. It is also better than the best one with longer proteins (trained on proteins and fine-tuned on RNA), but worse than the best one with shorter proteins (trained on short proteins and fine-tuned on RNA). Compared to the model trained only on RNA data, it is worse in 9 cells and better in 11 cells. In terms of puzzles, it is better for Puzzles 3, 7, 11, 14f, 17, 21 and worse for Puzzles 6, 9, 10, 14b, 18, 19, and 20.

The model trained on combined protein and RNA data had worse results than one trained only on proteins combined. In comparison with the model trained on RNA data only, it is worse in 18 cells and better in 5. For Puzzles 2, 14b, 15 and 17 it has the same results, it has one better and one worse cell for Puzzle 14f and it is better for Puzzles 11, 20 and 21.

The model trained on combined protein data and then fine-tuned on RNAs had even worse results. It is worse in 27 cells and better in 3. It has the same results for Puzzles 2, 5 and 7, it is better for Puzzles 11, 14f and 17, and it has one better and one worse cell for Puzzle 20.

Even though the results published with (Townshend et al., 2021) are not achievable, if we want to compare our best model with theirs, the evaluation results can be found in Figure 4.13.

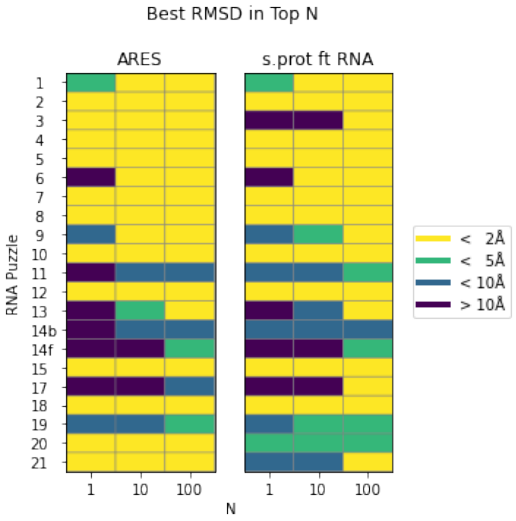

Figure 4.13: Evaluation results - comparison best with (Townshend et al., 2021)

ARES is better in 9 cells and worse in 5 cells. Our model improved for Puzzles 11, 14b, 17 and 19, and got worse for Puzzles 3, 9, 13, 20 and 21. The best-scoring structural model is near-native in 57% of Puzzles for ARES and in 43% for our model. The top 10 best-scoring models include at least one near-native model for 71% of the benchmark RNAs when using ARES, compared with 52% for the model trained on short proteins and fine-tuned with RNA. If we look at the top 100 best-scoring models, in both cases in 76% of the RNAs the best models include at least one near-native model.

5. Discussion

According to the results, it seems that we can improve the scoring function using both protein and RNA data. Although training on longer proteins did not produce better results, this could be due to the complex structures that longer proteins often fold into. As the model architecture is shallow, it is probably not capable of distinguishing structural patterns as well as in the case of shorter biomolecules. In (Townshend et al., 2021) the model was trained only with 18 short RNAs so it can make sense that it learns better with shorter proteins. Besides the complexity of the model, proteins' complexity could also be the reason for worse results. Even if the model learns these structures, during evaluation we test its performances on RNA Puzzles, which do not take on such complex structures as proteins, so this could also be a reason why the results aren't as good as when training with longer proteins.

On the other hand, when we train the model on shorter proteins, if we use only the proteins, we still don't get better results than when training with RNAs, but we get much better results than when training with longer proteins. In both cases, adding RNA to the training data improves performance. It seems that fine-tuning the model that has already been trained on protein data with RNAs works even better than when we train the model on combined protein and RNA data. The reason for such results could be due to the model being able to learn more about RNA when it is fine-tuned than when it is training on both protein and RNA data, and can not distinguish the difference.

The fact that we got better results when training the model with longer and shorter proteins than when training with only RNA data is very interesting, since this would mean that we can utilise protein data, whose quantity is much higher than RNA data, for predicting and scoring RNA structures. Worse results when adding RNA to the training set were not expected and it could be because the model is overfitted so adding more data is actually corrupting the weights. It could also mean that protein and RNA data are quite different, and when the model has learnt properly about protein data, it can not be switched to RNA data. Next experiments should include searching for

better hyperparameters for fine-tuning, testing if the model is overfitting and repeating the experiment with smaller and larger amounts of combined longer and shorter protein data.

Another thing that could have impacted the results is the evaluation dataset itself. It consists of only 21 RNAs, therefore expanding the dataset could give a more objective evaluation. This dataset was chosen so the comparison between (Townshend et al., 2021) and our models would be possible, but even if 21 RNAs is enough, the dataset isn't optimal. There is not an equal number of structural models per puzzle, and this number differs highly - for some puzzles, there are only about 1500 structures, while for others there are more than 48000 structures. Ignoring the number of structures, if we look at the distribution of RMSD values for each puzzle, the distribution varies a lot. This impacts the simplicity of getting a good structure, since if all of the structures are close to the native one (as is the case for Puzzle 2), the task of giving a good score to at least one near-native structure is trivial.

Except for changing or expanding the evaluation dataset, another thing that could be tried out is a better selection of the training dataset. All of the protein datasets used for training were chosen randomly, the only thing that was cared for is that every protein should be from a different structural cluster and if their length is in the desired range. The reason for better results with shorter proteins could have come from the selection of proteins as short and longer proteins were not from the same clusters. It would be better to include some biological insights and see what structural patterns are included in the training and in the evaluation to see if some important pattern was accidentally skipped or if it prevails. There are also some structural patterns, like beta sheets, which proteins have and RNAs don't, so it could be better to avoid such structures when choosing the training dataset.

Additionally, the choice of Huber loss as a loss function maybe isn't optimal. It may not be optimal due to the fact that how good the loss function was doesn't tell much about how good evaluation results would be for that same model. This may be improved by trying to predict the RMSD group, not the RMSD value itself. For example, using bins 0-2Å, 2-4Å, 4-6Å etc and then using cross-entropy loss. These bins are more similar to the way of evaluating the results and should be easier to obtain.

In future work, one thing we should try is to expand the model, and see if the problem with longer protein structures was that the model was too shallow. Another thing to try out with the model is changing the loss function. As for the evaluation, the dataset should be equalized, we should try to generate more structures and get the distribution of RMSD values which are more alike. We could also add more RNAs to

the evaluation dataset, preferably RNAs which are different to the ones currently in the training and evaluation datasets.

6. Conclusion

The aim of this thesis was to investigate to which extent information from the protein structure data can be used to improve the scoring of RNA molecules. This was done by retraining the ARES neural network model (Townshend et al., 2021) for atom configuration scoring and obtaining results on the RNA structure data.

The results have shown that combining protein and RNA data can improve the scoring function. It is best to train the model on shorter proteins and fine-tune it on RNA data. It is also possible to get better results by training the model on a larger amount of protein data than when training only on RNA data. This confirms the premise that the ARES model learns entirely from atomic structure, using no other information such as related RNA or protein sequences, and it makes no assumptions about what structural features might be important.

There is much more that can be done to confirm the obtained results. Expanding the model by adding more equivariant convolution layers, changing the loss function of the model, expanding and equalizing the evaluation dataset are just some of the examples. It would also be useful to have biologically more detailed insight into the training, validation and evaluation datasets so that more conclusions could be made about structure complexity, existence or non-existence of certain structural motifs and the general relationship between RNA and protein structures.

The fact is that RNA structure prediction and structure scoring tasks are difficult and there are far fewer structural measurements of RNA molecules currently available, which adds to the difficulty of the task. The evidence that protein data can help with improving the results for RNA scoring, and consequently for RNA prediction, is important because the amount of protein data is much higher, therefore there are more samples which can be utilized for training the models and in the long run do the same revolution with RNAs as AlphaFold2 (Jumper et al., 2021) did with proteins.

REFERENCES

C. B. Anfinsen. Principles that govern the folding of protein chains. *Science*, 181 (4096):223–230, July 1973.

Protein Data Bank. Protein data bank. *Nature New Biol*, 233:223, 1971.

R. Bonneau, J. Tsai, I. Ruczinski, D. Chivian, C. Rohl, C. E. Strauss, and D. Baker. Rosetta in CASP4: progress in ab initio protein structure prediction. *Proteins*, Suppl 5:119–126, 2001.

OpenStax CNX. Structure and function of rna. `https://courses.lumenlearning.com/suny-mcc-microbiology/chapter/structure-and-function-of-rna/`. Accessed: 2022-06-01.

K. A. Dill, S. B. Ozkan, M. S. Shell, and T. R. Weikl. The protein folding problem. *Annu Rev Biophys*, 37:289–316, 2008. doi: 10.1146/annurev.biophys.37.092707.153558.

Dominik Gront, Daniel W. Kulp, Robert M. Vernon, Charlie E. M. Strauss, and David Baker. Generalized fragment picking in rosetta: design, protocols and applications. *PLoS One*, 6(8):e23294, August 2011.

J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, Aug 2021. ISSN 1476-4687. doi: 10.1038/s41586-021-03819-2. URL `https://doi.org/10.1038/s41586-021-03819-2`.

Z. Miao and E. Westhof. RNA structure: Advances and assessment of 3D structure prediction. *Annu Rev Biophys*, 46:483–503, March 2017.

J. Moult, J. T. Pedersen, R. Judson, and K. Fidelis. A large-scale experiment to assess protein structure prediction methods. *Proteins*, 23(3):ii–v, November 1995.

F. Mölder, K. P. Jablonski, B. Letcherand, M. B. Hall, C. H. Tomkins-Tinch, et al. Sustainable data analysis with snakemake [version 2; peer review: 2 approved]. *F1000Research*, 2021. URL https://doi.org/10.12688/f1000research.29032.2.

R. Pearce, G. S. Omenn, and Y. Zhang. De novo rna tertiary structure prediction at atomic resolution using geometric potentials from deep learning. *bioRxiv*, 2022. doi: 10.1101/2022.05.15.491755. URL https://www.biorxiv.org/content/early/2022/05/15/2022.05.15.491755.

S. Raman, R. Vernon, J. Thompson, M. Tyka, R. Sadreyev, et al. Structure prediction for casp8 with all-atom refinement using rosetta. *Proteins*, 77 Suppl 9(0 9):89–99, 2009. ISSN 1097-0134. doi: 10.1002/prot.22540. URL https://pubmed.ncbi.nlm.nih.gov/19701941. 19701941[pmid].

K. Sato, M. Akiyama, and Y. Sakakibara. Rna secondary structure prediction using deep learning with thermodynamic integration. *Nature Communications*, 12(1):941, Feb 2021. ISSN 2041-1723. doi: 10.1038/s41467-021-21194-4. URL https://doi.org/10.1038/s41467-021-21194-4.

K. T. Simons, R. Bonneau, I. Ruczinski, and D. Baker. Ab initio protein structure prediction of CASP III targets using ROSETTA. *Proteins*, Suppl 3:171–176, 1999.

J. Singh, J. Hanson, K. Paliwal, and Y. Zhou. Rna secondary structure prediction using an ensemble of two-dimensional deep neural networks and transfer learning. *Nature Communications*, 10(1):5407, Nov 2019. ISSN 2041-1723. doi: 10.1038/s41467-019-13395-9. URL https://doi.org/10.1038/s41467-019-13395-9.

R. J. L. Townshend, S. Eismann, A. M. Watkins, R. Rangan, M. Karelina, R. Das, and R. O. Dror. Geometric deep learning of rna structure. *Science*, 373(6558):1047–1051, 2021. doi: 10.1126/science.abe5650. URL https://www.science.org/doi/abs/10.1126/science.abe5650.

A. M. Watkins, R. Rangan, and R. Das. Farfar2: Improved de novo rosetta prediction of complex global rna folds. *Structure*, 28(8):963–976.e6, 2020. ISSN 0969-2126. doi: https://doi.org/10.1016/j.str.2020.05.011. URL https://www.sciencedirect.com/science/article/pii/S0969212620301805.

Combining Protein and RNA Structures Information in Developing New Scoring Function

Abstract

Compared to protein structure prediction, the RNA structure prediction task is even more difficult due to the fact that there are far fewer structural measurements of RNA molecules currently available. One way of approaching the structure prediction problem is to train a neural network that can score how far a given configuration of atoms is from being a valid RNA molecule. Such a model, named ARES (Atomic Rotationally Equivariant Scorer), has recently been implemented. This thesis has investigated to which extent information from the vast pool of protein structure data can be used to improve the scoring of RNA molecules. It has been shown that the combination of protein and RNA data can improve the results. Training the model on a larger amount of protein data has confirmed the fact that the ARES neural network can utilize the fact that both proteins and RNA molecules are long chains of organic compounds and fold under similar conditions inside a cell and consequently this model has better results than the one trained only on RNA data.

Keywords: RNA structures, protein folding, Equivariant Convolution, spherical harmonics, root mean square deviation

Kombiniranje informacija o strukturi proteina i RNA u razvoju nove funkcije bodovanje temeljene na dubokom učenju

Sažetak

Predikcija strukture RNA molekula još je teži zadatak od predikcije stukture proteine zbog toga što je trenutno dostupno značajno manje strukturnih mjerenja RNA molekula. Jedan od pristupa predviđanju strukture RNA molekula je treniranje neuronske mreže koja bi ocjenjivala koliko je neka konfiguracija atoma daleko od toga da bude ispravna RNA molekula. Jedan takav model, nazvan ARES (Atomic Rotationally Equivariant Scorer), nedavno je implementiran. Ovaj rad istražio je u kolikoj se mjeri velika količina podataka o proteinskim strukturama može iskoristiti da bi se popravilo ocjenjivanje strukture RNA molekula. Pokazano je da kombinacija podataka o strukturama proteina i RNA molekula može poboljšati rezultate. Treniranje modela na većoj količini podataka o strukturama proteina pokazuje da ARES neuronska mreža može iskoristit činjenicu da su i proteini i RNA molekule u osnovi dugi lanci organskih spojeva koji se svijaju u sličnom okruženju unutar stanice i posljedično takav model ima bolje rezultate od onoga treniranog samo na podacima o strukturama RNA molekula.

Ključne riječi: RNA strukture, savijanje u proteinskoj strukturi, ekvivarijantna konvolucija, kugline funkcije, korijen srednje kvadratne devijacije