Genegram: RNA Secondary Structure Prediction Using Formal Grammars and Residual Neural Networks*

Polina Lunina[1,2][0000−0002−7172−2647] ✉, Semyon Grigorev[1,2][0000−0002−7966−0698], and Vadim Abzalov[1,2][0000−0002−0805−0315]

[1] Saint Petersburg State University, 7/9 Universitetskaya nab., St. Petersburg, 199034, Russia
[2] JetBrains Research, Primorskiy prospekt 68-70, Building 1, St. Petersburg 197374, Russia
lunina_polina@mail.ru ✉, semyon.grigorev@jetbrains.com, vadim.i.abzalov@gmail.com

Abstract. Improvement in RNA secondary structure prediction accuracy is one of the key focuses in computational genomics due to its crucial role in the functional analysis of RNA molecules. We propose a new approach to solve this problem. Simple context-free grammar describes the most common types of stems and the parsing matrix for some sequence represents all the theoretically possible folds for each subsequence. This information is excessive because only one combination of stems will be presented in a real secondary structure. Moreover, the parsing matrix can be incomplete due to inevitable limitations in grammar. So, we process parsing matrices with residual neural networks in order to generate a valid secondary structure. This approach allows to process pseudoknots, wobble base pairs, and multiple contacts due to the flexible nature of the neural network training process. Our solution was implemented as a Python console tool Genegram.

Keywords: RNA · Secondary Structure · Genomic Sequences · CNN · ResNet · Machine Learning · Formal Grammars · Parsing.

1 Introduction

Secondary structure of the RNA molecule is actively involved in different genome regulation mechanisms, moreover, it contains a lot of important information for sequences phylogenetic and taxonomic analysis. Therefore, RNA secondary structure prediction problem is quite critical in computational genomics, unfortunately, it still remains open due to the complexity and variability of secondary structure formation laws along with a high degree of noisiness in biological data. Another challenging part here is the processing of pseudoknotted structures that

* Supported by the Russian Science Foundation grant 18-11-00100

are known to be widely represented in organisms genomes, including functionally important RNA regions, nevertheless, the overlapping nature of pseudoknots makes handling them an NP-complete task [1].

The best results in the secondary structure prediction field belong to the laboratory methods, such as X-ray structural analysis [2] and nuclear magnetic resonance [3], however, the high cost and complexity of such experiments lead to the rapid development of computational methods all the diversity of which can be divided into two groups. The first one contains comparative methods that analyze several homologous sequences employing evolutionary approaches [4,5]. These methods are known to be quite reliable but they demand a lot of data and complicated manual analysis. The second group aggregates different single sequence methods that process one sequence at a time applying some mathematical or physical model along with its optimization scheme. The most popular approach here is the minimum free energy (MFE) principle based on secondary structure thermodynamic stability, and the task of searching the MFE folding can be solved by dynamic programming [6,7], heuristic algorithms [8,9] and other optimization techniques [10,11]. Moreover, there are methods that do not demand any physically measurable parameters and model secondary structure by maximizing some function of expected accuracy (MEA) [12,13] or building stochastic context-free grammar [14,15,7]. RNA folding mechanisms are known to be nontrivial for understanding and formalization, therefore, machine learning techniques are widely used for secondary structure prediction either integrated into some strict algorithm pipeline [16,17] or as the main method itself [18,19].

In this work, we introduce a new approach for RNA secondary structure prediction employing the combination of ordinary formal grammars and artificial neural networks. The main ideas were outlined in [20,21] and this research is conducted to the further development of this approach in the context of secondary structure prediction problem. The classical way of formal languages application here is to use complicated stochastic grammars for modeling the whole structure [14,15,22] and this approach is known to be quite successful, however, building such grammar requires a lot of theoretical and practical difficulties. To overcome these issues we propose to split the secondary structure into elementary parts (stems), describe them by very simple grammar, and use neural networks to synthesize the final result. So, on the one hand, using neural networks allows to skip full formalization of RNA secondary structure and on the other hand, the grammar provides some sort of basis for neural network training.

2 Proposed Approach Overview

In this section, we focus more on the theoretical aspects of the proposed approach that underlie the experimental research presented in section 3.

Secondary structure can be viewed as a recursive composition of stems having different lengths and loop sizes [23], so, we propose to use formal grammar to encode the most common types of stems, considering RNA sequence as a string in the $\{A, C, G, U\}$ alphabet. Thereby, the parsing matrix for some sequence

will contain information about whether each subsequence of this sequence can fold to stem or not. This matrix is not yet a representation of a valid secondary structure, because all these stems cannot exist at once and, besides, there can be more complex elements that are not expressible in terms of given grammar (for example, pseudoknots and wobble base pairs). Therefore, we propose to process parsing matrices by neural networks that should filter extra stems and add the missing elements in order to generate a maximal approximation of real secondary structure in some compatible with another studies format. In figure 1 the global solution architecture is presented and in the next sections, we will speak in detail about each of the specified components.

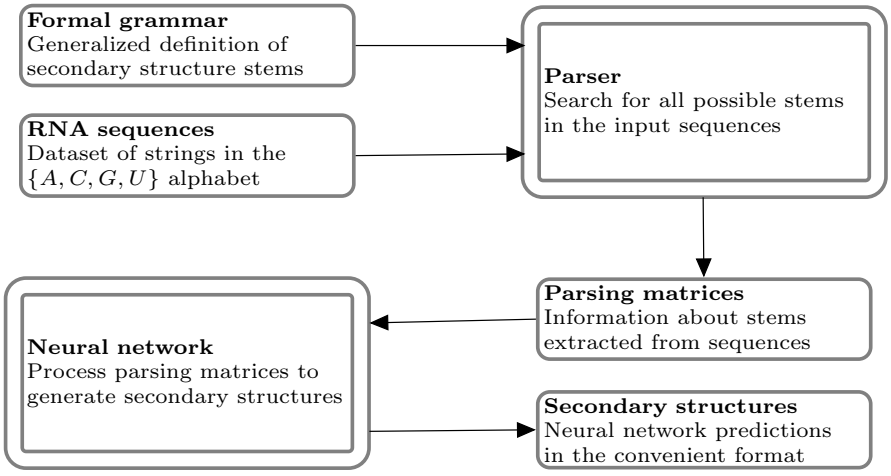


Fig. 1: Global solution architecture

2.1 Formal Grammar

As it was mentioned before, our approach employs formal grammar not for modeling the whole structure, but for encoding its simple constructional elements (stems), therefore this grammar does not require probabilistic rules — probability estimation is to be done by neural networks. We do not strictly fix the grammar and it can be modified in accordance with research purposes or data characteristics, so, the grammar presented below is just a concrete example that showed quite successful results during the experiments.

In figure 2 context-free grammar G_0 that we use in this work as well as in the previous ones is presented. This grammar describes the recursive compositions of stems having height at least 3 (lines **7-12**) and loop size from 1 up to 20 (line **3**). Note that these constants are not mandatory and might be defined

experimentally for each task. Also, G_0 allows only four classical nucleotides (line **4**) and conventional base pairs (line **5**), because adding the rules for more pairs and nucleotides complicates the grammar unacceptably in the context of performance, moreover, context-free grammars cannot explicitly express pseudoknots, therefore, we expect the neural network to handle all these features instead. Also, we consider only stems of height three or more (lines **1-2**), because including shorter stems would overload the parsing matrix with too much extra information. So, by this rules, a sequence folds to stem if and only if it is derivable from start nonterminal *start* of G_0 (line **1**).

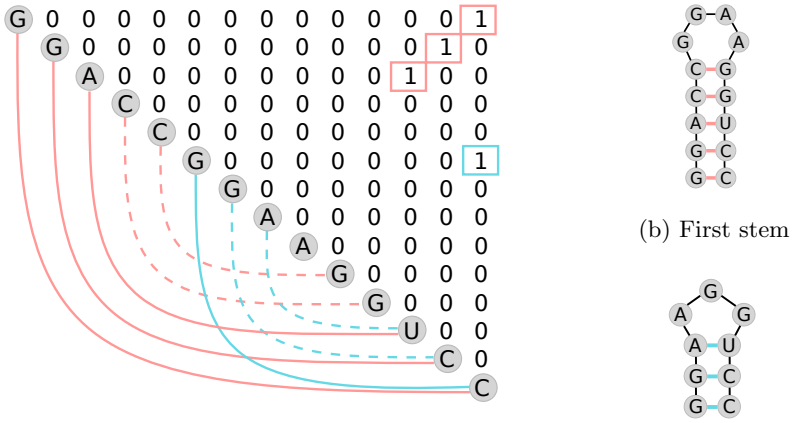
```
1  start: stem3<s0>
2  s0: loop | loop stem3<s0> s0
3  loop: nucl*[1..20]
4  nucl: A | U | C | G
5  stem1<s>: A s U | G s C | U s A | C s G
6  stem2<s>: stem1<stem1<s>>
7  stem3<s>:
8        stem1<stem2<s>>
9      | A stem3<s> U
10     | U stem3<s> A
11     | C stem3<s> G
12     | G stem3<s> C
```

Fig. 2: Context-free grammar G_0 for RNA secondary structure stems description

Having a grammar, we want to find all the subsequences of some given sequence that may fold to stems and this is to be done by means of parsing. Formally, the result of a matrix-based parsing algorithm for an input string w is an upper-triangular boolean matrix M_P, where $M_P[i, j] = 1$ if and only if the substring $w[i, j]$ is derivable from grammar start nonterminal. From the practical point of view, this means that parsing matrix contains one in a cell if and only if a correspondent substring folds to stem according to the rules of a given grammar, so, each stem results in a diagonal chain of ones in the matrix, because if sequence $w_1...w_n$ is a stem than it is clear that $w_2...w_{n-1}$ is a stem, $w_3...w_{n-2}$ is a stem and so on while the stem height is at least 3.

In figure 3 we provide the parsing result for a short RNA sequence and show how parsing matrix maps with secondary structure stems. Each one cell describes the stem of height at least 3, so, this sequence contains two subsequences that may fold to stems of the first nesting level. These stems expected hydrogen bonds along with corresponding matrix cells are painted in two different colors. All nucleotide bonds forming a stem of height three or more are represented by solid lines, moreover, it is obvious that each stem of height three encapsulates stems of heights two and one which are highlighted by dotted lines. Note that these stems interfere with each other, thereby, secondary structure cannot contain both of them at the same time. So, the parsing matrix for a sequence describes all the

theoretically possible folds there, but at the current step, we cannot know which one of them would be presented in the real secondary structure. Moreover, G_0 has certain limitations, such as stem height, loop size, and possible base pairs, so, some of the required stems may be missing in the parsing matrix. While creating a grammar we were guided by two competing ideas: to cover as many types of stems as possible and to stay with an adequate amount of extra information in the parsing matrix along with acceptable time costs for parsing.



(a) Parsing matrix

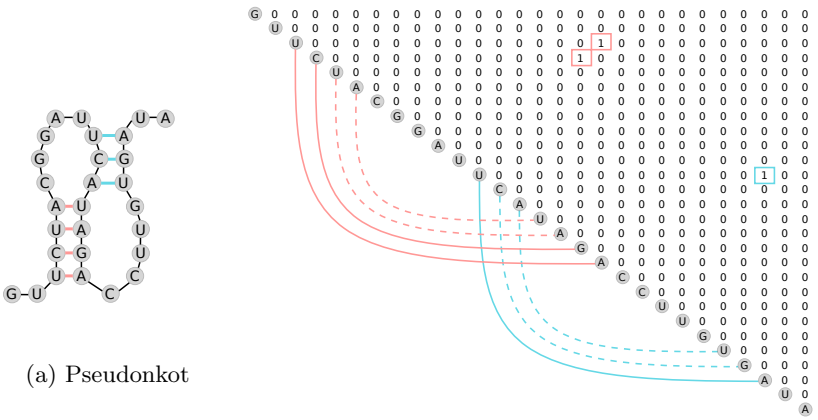
(b) First stem

(c) Second stem

Fig. 3: Stems extracted from RNA sequence by parser

Let us consider pseudoknotted structures processing. Even though it is clear that pseudoknot is not explicitly expressed in $G-0$, as well as in any context-free grammar, it consists of two stem-loop structures having half of one stem located between two halves of another, so, each of these stems separately can be derived by the rules of G_0. Therefore, pseudoknots will be reflected in the parsing matrix, and handling them becomes an additional task for a neural network. In figure 4 an example of simple pseudoknot along with corresponding parsing result is provided. Two stems of this pseudoknot are highlighted with two different colors and it can be seen that all the related nucleotide bonds are presented in the parsing matrix, although at this point it is not determined whether this sequence contains a pseudoknot or it just has two possible folds in terms of grammar.

To sum up, our grammar explicitly or implicitly describes a wide range of secondary structure elements and even the missing ones do not become a problem for our approach — neural network is still capable to handle them. We think of such flexibility as one of the key advantages of our solution.



(a) Pseudonkot

(b) Parsing matrix

Fig. 4: Pseudoknots processing in terms of G_0

2.2 Neural Network

The matrix that our parser produced for some sequence and fixed grammar should be processed by a neural network in order to achieve a maximal similarity with the expected secondary structure for this sequence. Therefore, we need to specify the data preparing pipeline and describe the required architecture for the problem at hand. It should be noted that these are global and permanent aspects in the context of current work, however, both of them can and probably will be changed while conducting some different research.

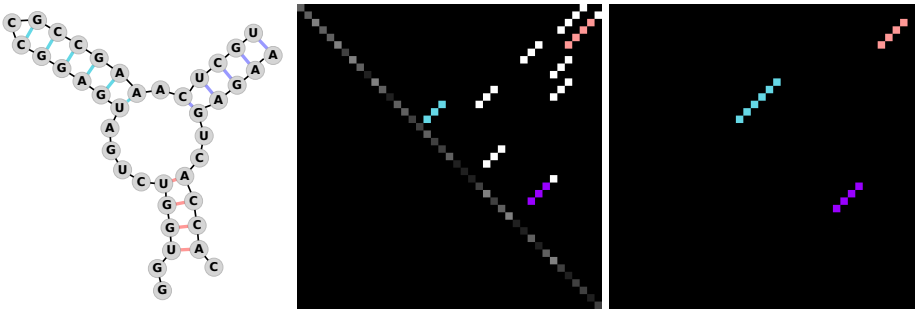
Data The input data for neural network (parsing matrices) was described in the previous section and now let us define the reference data source and format. There are specialized biological databases containing RNA chains of various organisms along with their secondary structures obtained by reliable methods, and such data is known to be the best for algorithms training and validation. These databases may store data using different representations (dot-bracket, connectivity table, and others), so, we need to choose a format that is convenient for our experiments and compatible with others.

One of the ways of RNA secondary structure formal representation is so-called contact map, which for an input string w is a boolean matrix M_C, where $M_C[i, j] = 1$ if and only if nucleotides in positions i and j form a hydrogen bond (or, to put it simply, a contact) in secondary structure. Even though it is not the most popular format, contact matrix can be easily obtained from the corresponding connectivity table, moreover, there are certain tools that transform connectivity tables to dot-brackets [6], which provides the required compatibility. Let us consider for the above sequence w the discussed earlier parsing matrix M_P that has $M_P[i, j] = 1$ if and only ifsubsequence $w[i, j]$ folds to a stem. It is clear

that the first and the last nucleotides of every stem form a contact, therefore, we can easily transfer between parsing matrix and contact map definitions and view the parsing matrix as a sort of a contact map, so, this format is convenient for our experiments. Note that if parser finds a stem of height three than we will see only one cell with 1 in matrix, but such stem always wraps a stem of height two which wraps a stem of height one, so, we are always missing two contacts, therefore, after parsing we should set $M_P[i-1, j+1] := 1$, $M_P[i-2, j+2] := 1$ if $M_P[i][j] = 1$, $i = 0..size(M_P), j = i + 1..size(M_P)$ for complete semantic equality of parsing and contact matrices.

So, a neural network should take parsing matrices as inputs and contact maps as desired outputs for the same set of RNA sequences. For convenience, we transform both matrices to black-and-white images by replacing zero cells with black pixels and one cells with white pixels. Also, we code RNA sequence at the input image main diagonal by four types of gray pixels corresponding to the four possible nucleotides in case the chain itself contains any important information about secondary structure formation patterns.

In figure 5 we provide two-dimensional secondary structure visualization for RNA sequence along with neural network input and reference images that were made from parsing and contact matrices respectively. Contacts belonging to the three stems presented in this sequence are highlighted with three different colors in all pictures (so, input and reference images are actually grayscale, colored pixels are only here for clarification). It can be seen that not all of the stems found by the parser are presented in the real secondary structure. Moreover, the parsing result is missing several contacts since they were formed by non-canonical nucleotide pairs $A - G$ that are not expressed by grammar G_0. Now the purpose of a neural network becomes clear — it should take image 5b and transform it to the image 5c as accurately as possible. And the ideas behind this transformation are outlined in the next section.



(a) Secondary structure visualization

(b) Input image for neural network

(c) Reference image for neural network

Fig. 5: Secondary structure representations

Parallel ResNet One of the popular architectures for complicated images processing tasks is a residual neural network based on adding skip connections between blocks of layers [24]. ResNets solve the problem of vanishing gradient and allow to effectively use deep convolutional networks.

In this work, we developed a new architecture that showed its applicability for secondary structure prediction task during experimental research. The idea is to combine n identical residual networks that take the same input, go through independent sequences of layers and connect their n outputs with weighted sum hanging it over to the final residual unit that directly generates output. This parallel residual architecture along with the scheme of a typical residual unit is presented in figure 6, where k and n can be selected based on empirical evidence. We believe that the advantage of this parallel technique is that these separate networks are able to find different types of features in data and some sort of voting system allows the whole model to decide for the particular pixel whether each network behaves correctly or not.

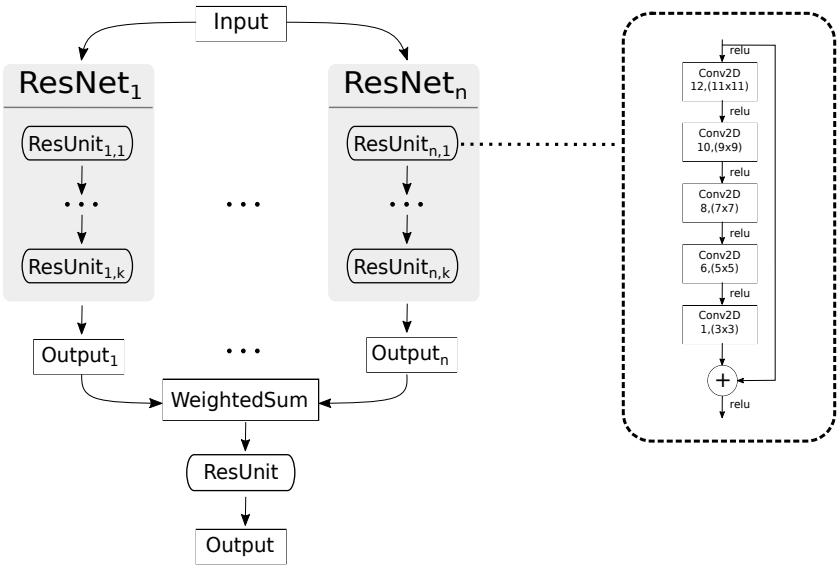


Fig. 6: Parallel ResNet architecture

3 Experiments

This section demonstrates the practical results of the proposed approach application. We introduce an easy-to-use implementation, specify some experimental details and describe currently available models. We compare our results with several popular tools and show the competitive power of our models.

3.1 Tool

The proposed approach was implemented as a Python console tool Genegram. Code, installation instructions, and documentation are available by link https://github.com/JetBrains-Research/Genegram. This tool accepts a file with RNA sequences in fasta format and returns the connectivity tables (ct) for the corresponding secondary structures. We use an effective Python implementation of the parsing algorithm [25] that shows high performance due to the GPGPU utilization and Keras [26] library with Tensorflow-GPU [27] framework for running the predictive model. Genegram tool allows to select different models and process sequences of lengths 1-200 containing only four canonical nucleotides.

3.2 Setup

The approach itself is quite flexible and Genegram tool provides a convenient environment for different experiments allowing to change both grammar and network. However, in the present work, we freeze grammar G_0 from figure 2 and parallel ResNet architecture from figure 6 with $k := 10$, $n := 4$. We consider only small RNAs (lengths from 1 to 200) due to the insufficient amount of longer sequences in biological databases and GPU storage limitations. Each sequence of l nucleotides results in the image of size $l \times l$, therefore, images are grouped into batches by image size and cyclically duplicated until all batches have the same length. We use two data augmentation techniques on the training dataset: firstly, we mirror each image along the main diagonal (so that mirrored image represents the same sequence turned backwards), and secondly, we copy both regular and mirrored images twice. On this data, we run 10-fold cross-validation in order to choose the best train and validation split for each model.

The quality of the results is estimated by classical machine learning metrics calculated on the pixel-by-pixel difference between predicted and reference images. Further, we consider TP (true positives, i.e. true white pixels), FP (false positives, i.e. false white pixels), and FN (false negatives, i.e. false black pixels) as numbers of right and wrong decisions among all contacts and calculate three following metrics for each image of some dataset.

- $Precision = \dfrac{TP}{TP + FP}$ (proportion of correct contacts among all detected).
- $Recall = \dfrac{TP}{TP + FN}$ (proportion of detected contacts among all expected).
- $F1 = 2 * \dfrac{Precision * Recall}{Precision + Recall}$ (harmonic mean — aggregation metrics).

The loss function is based on the idea of maximizing $F1$ score and that is to be achieved by minimizing the dataset mean $1 - F1$ through gradient descent. However, $F1$ is not differentiable as a function, so, we replace the sums of discrete integer values with a continuous sum of probabilities. Also, we charge our loss with two penalty coefficients responsible for huge dispersion between *Precision* and *Recall* for each image ($k1$) and the whole dataset ($k2$). Figure 7 demonstrates the Python code for the currently used $F1\_loss$ function that accepts two arguments: predicted image y_p and true image y_t.

```
from keras import backend as K

def f1_loss(y_t, y_p):
    #normalize pixels values to [0, 1]
    y_t, y_p = K.minimum(y_t / 255, 1), K.minimum(y_p / 255, 1)
    #calculate differentiable versions of tw, fw and fb
    tw = K.sum(K.cast(y_t * y_p, 'float32'), axis=[1, 2, 3])
    fw = K.sum(K.cast((1 - y_t) * y_p, 'float32'), axis=[1, 2, 3])
    fb = K.sum(K.cast(y_t * (1 - y_p), 'float32'), axis=[1, 2, 3])
    #calculate precision and recall secure from zero division error
    prec = tw / (tw + fw + K.epsilon())
    rec = tw / (tw + fb + K.epsilon())
    #penalties for huge difference between precision and recall
    #calculated for each image and whole dataset respectively
    k1 = 1 -  K.abs(prec- rec)
    k2 = 1 -  K.abs(K.mean(prec) - K.mean(rec))
    #calculate upgraded f1 score and return its mean value
    f1 = k1 * k2 * 2 * prec * rec / (prec + rec + K.epsilon())
    return 1 - K.mean(f1)
```

Fig. 7: $F1\_loss$ function

For hyperparameters, we use Dropout after each residual unit to deal with overfitting, L2-regularization that also prevents overfitting and allows to search for complex data patterns, and Adagrad optimizer that automatically sets the learning rate and is known to be a powerful solution for sparse data processing.

For a comparative analysis of the results we selected six tools based on various concepts and algorithms. All of them demonstrate adequate speed and high accuracy, can handle pseudoknots and are easy to launch and use.

- SPOT-RNA [18] — deep neural networks + transfer learning.
- Ipknot [12] — MEA + integer programming.
- Knotty [11] — MFE + sparse dynamic programming.
- RNAstructure [6] — MFE + dynamic programming.
- PknotsRG [10] — MFE + Turner energy rules.
- HotKnots [8] — MFE + heuristic algorithm.

3.3 Models

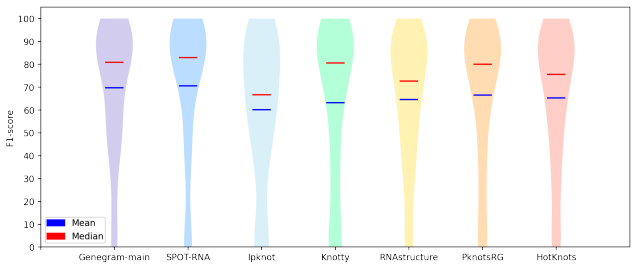
In these conditions, we trained three identical networks on three different datasets and compared them with each other and with other tools by several criteria. All

three models are available to use in the Genegram tool and the default model here is Genegram-main. Let us describe the specifics of each dataset, present the results of the best 10-fold cross-validation splits, and draw some conclusions.

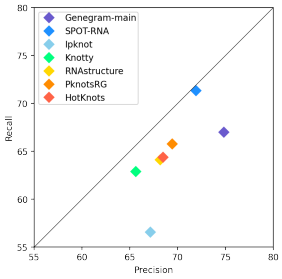
Genegram-main The first model was trained on data obtained from RNA STRAND database [28] that is quite popular in different RNA analysis research due to its quality and usability. This database is an assembly of carefully curated and validated RNA sequences with secondary structures collected from different sources, and it contains only trustful structures obtained by reliable methods (laboratory or comparative). This database is supposed to provide the most representative sample of RNA secondary structures along with a guaranteed high quality of data which makes Genegram-main model trained on RNA STRAND the most general one for now, therefore, we set it as a default choice for our tool.

We selected 1091 sequences with lengths up to 200 and removed samples with gaps or inaccuracies in primary and secondary structures. The distribution of sequences lengths in the prepared dataset is demonstrated in figure 8c. Neural network output images are grayscale and may contain multiple contacts, therefore, we applied binarization by threshold 0.6 with multiplets removal as postprocessing for this model and calculated the following metrics afterwards. Figure 8a shows the estimation of Genegram-main and other six tools on validation set of size 109 by metrics $F1$. The shape of colored plots shows the density of $F1$ values, the red line represents its median result and the blue line corresponds to the mean one. It can be seen that Genegram-main has median $F1$ equal to 81 (as well as Knotty which is the second result after SPOT-RNA), and mean $F1$ equal to 70 that significantly exceeds the results of all competing tools except SPOT-RNA. The estimation on the same set by mean *Precision* and *Recall* metrics is presented in figure 8b. Note that the binarization threshold allows us to slightly manipulate the balance between these two metrics and we choose it so that $F1$ mean value is the highest, even though it leads to a small excess of *Precision* over *Recall*.

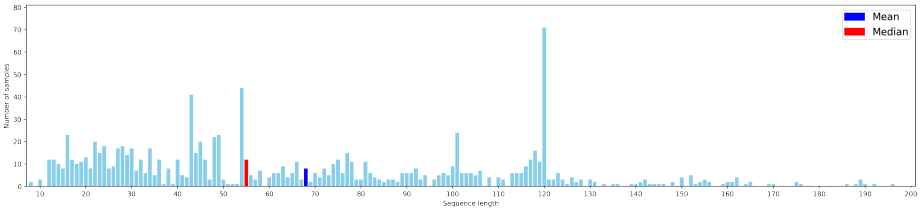
All models were also estimated by several more specific functional criteria, such as pseudoknots prediction accuracy and sensitivity to different types of base pairs, so, table 1 demonstrates the results of these experiments. Considered validation set contained 11 pseudoknots and we calculated the number of correctly predicted contacts for each pseudoknot in each model output. The second column of table 1 shows the number of perfectly detected pseudoknots and the third column — the number of ones that had up to 25% extra or missing contacts. Clearly, pseudoknotted structures prediction is a non-trivial task for almost all models including Genegram-main. The other three columns of table 1 demonstrate how these tools handle different types of base pairs. Watson-Crick pairs are the most common and stable ones and it can be seen that all models have high enough canonical pairs prediction accuracy. Besides, it is known that wobble base pairs also have a biological role and regularly appear in real-world data, especially GU pair that has stability close to the Watson-Crick bonds. Our model slightly loses to other tools in GU pairs prediction, however, the



(a) $F1$ distribution, mean, and median on valid set



(b) *Precision* and *Recall* mean on valid set



(c) Sequences lengths distribution on total set

Fig. 8: Analysis of Genegram-main model compared with other tools

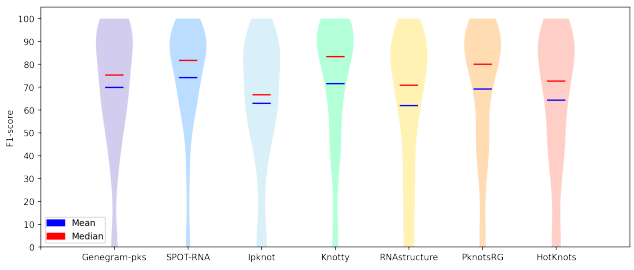
great advantage of our approach is that it does not limit any types of pairs to be presented in neural network output (as well as SPOT-RNA that also uses neural networks), so, Genegram-main and SPOT-RNA are able to handle seven more wobble base pairs (AA, AC, AG, CC, CU, GG, UU), unlike other considered tools.

Table 1: Genegram-main and other models quality criteria measured on valid set

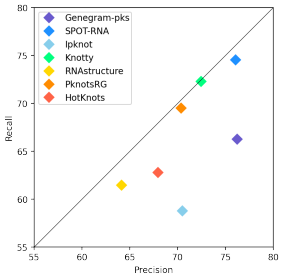
	Pseudoknots		Base pairs		
	No errors	25% errors	Watson-Crick	GU	Others
Genegram-main	0	3	1067	68	53
SPOT-RNA	1	4	1190	134	39
Ipknot	0	1	939	92	0
Knotty	1	3	1093	126	0
RNAstructure	1	1	1113	119	0
PknotsRG	1	5	1157	123	0
HotKnots	1	2	1113	112	0
Expected	11	11	1650	185	135

Genegram-pks Although the main model shows high results in general, it has poor pseudoknots prediction accuracy because RNA STRAND database contains only 86 sequences with pseudoknots and the neural network has not enough data to learn them, moreover, the amount of pseudoknots in the considered validation set is not enough to draw any conclusions. To improve this, we extended the previous dataset with sequences from Pseudobase [29] database that is known to be a carefully built collection of pseudoknotted structures. Even though this database mostly contains not the whole sequences but only fragments with pseudoknots, we believe that the presence of such data in the training set may allow us to reach higher results in pseudoknots prediction.

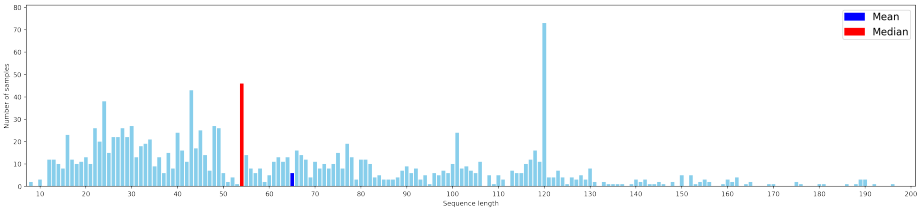
We selected 1447 sequences having lengths up to 200 with no gaps or inaccuracies in primary and secondary structures and applied the same binarization by threshold 0.6 with multiplets removal as postprocessing. Note that both times we used 10-fold cross-validation on randomly shuffled data, so, Genegram-pks has separate from Genegram-main validation set, even though samples may intersect. Figure 9c shows the distribution of sequences lengths in total Genegram-pks dataset, figure 9a estimates all seven models on 145 validation sequences by $F1$ and figure 9b — by *Precision* and *Recall* metrics. Genegram-pks is behind two tools by mean $F1$ value (70) and behind three tools by median (75) one, so, these results are quite competitive, however, Genegram-main was able to achieve the higher numbers.



(a) $F1$ distribution, mean, and median on valid set



(b) *Precision* and *Recall* mean on valid set



(c) Sequences lengths distribution on total set

Fig. 9: Analysis of Genegram-pks model compared with other tools

Similarly, we present the results of functional tests for this model and other tools in table 2. It can be seen that Watson-Crick pairs are predicted similarly to the main model, but wobble base pairs prediction accuracy decreased. However, the pseudoknotted structures processing was of particular interest in the context of Genegram-pks model and estimation by this criteria shows significant growth compared to the Genegram-main — almost a quarter of all pseudoknots was detected perfectly, and about a half had small prediction errors. To sum up, this model generally losses to the main one, however, it shows very promising results in the aspect of pseudoknots detection being one of the best models for this problem.

Table 2: Genegram-pks and other models quality criteria measured on valid set

	Pseudoknots		Base pairs		
	No errors	25% errors	Watson-Crick	GU	Others
Genegram-main	9	18	1628	61	47
SPOT-RNA	4	10	1893	173	71
Ipknot	2	5	1497	120	0
Knotty	11	20	1833	156	0
RNAstructure	3	7	1550	140	0
PknotsRG	11	20	1742	150	0
HotKnots	6	6	1621	143	0
Expected	43	43	2407	258	156

Genegram-mps (NOT READY!)

Intresting object that our approach allows to consider without its explicit definition is a multiplet that appears when base pair starts to form bonds with other bases or base pairs. Multiplets are known to be functionally important, although, they are classified as tertiary structure features, so, secondary structure prediction tools are not usually able to can handle them.

PDB provides more structural information, because it contains detailed crystallography result. The data is straight from nature: more variability in data — more possibilities but harder to learn.

Experimental model that has possibilities in predicting more features of secondary and even tertiary structures.

For now it's more a discussion about the possibilities of our approach than really presentable result.

Multiplets have different typology, hard to estimate
(https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6467009/)

Global tests In the previous sections we estimated three Genegram models separately on their validation sets and in this section, we provide some global tests for all the tools.

Due to the similar composition of training and validation sets for each model, the high results on validation make no reference to the general model quality, therefore, we downloaded several comparative and laboratory RNA secondary structure databases and evaluated Genegram models along with other tools on such data. These databases do not contain only purely selected structures, moreover, huge comparative databases usually aggregate a lot of homologous sequences with repetitive structural patterns, however, this experiment should check all models behavior on row data from various sources as opposed to the careful estimation on clean structures presented in the previous sections. We selected the following databases: CRW [30], Sprinzl [31], SRP [32], Rfam [33], PDB [34] (only single-chain samples without unmodelled bases) and Pseudobase [29]. Note that most of these samples are completely new, however, some sequences or even databases could be previously used for some models training (e.g. Pseudobase was included in Genegram-pks training set). Table 3 represents the estimation of three Genegram models along with other tools by mean $F1$ score on all databases and the last column shows the mean $F1$ value among total samples. It can be seen that in most cases our models show adequate numbers regarding other tools, so, we can state that all neural networks generalize well on new datasets.

Table 3: All models estimation by $F1$ metrics on different databases

	CRW	Sprinzl	SRP	Rfam	PDB	Pseudobase	All data
Genegram-main	69	73	59	43	74	42	69
Genegram-pks	66	72	56	48	74	81	68
Genegram-mps							
SPOT-RNA	89	88	65	72	81	70	87
Ipknot	74	79	49	64	76	59	75
Knotty	74	75	63	63	79	76	74
RNAstructure	66	69	56	64	77	62	67
PknotsRG	62	61	65	66	79	77	62
HotKnots	61	60	66	66	78	61	61
Total samples	11606	8777	360	1043	290	357	22433

Finally, we measured the time required for each tool to process fasta file with 100 RNA sequences and return secondary structure prediction for each sequence in its default output format. These tests were carried out on the workstation

with the following characteristics: Ubuntu 20.04.2 LTS, Intel Core i5-10210U CPU 1.60GHz, NVIDIA GeForce MX250 GPU, and 7.5 GB RAM. We prepared two datasets by randomly selecting sequences from RNA STRAND database: the first set contained 100 sequences having lengths up to 100 and the second one — 100 sequences with lengths up to 200. Table 4 shows the results of a performance experiment for such two fasta files. Genegram and SPOT-RNA were launched on GPU and the other tools have only CPU versions. It can be seen that the elapsed time spread among tools is quite big, moreover, several tools significantly lose in performance while processing longer sequences since various secondary structure prediction techniques may have completely different algorithmic complexity. However, Genegram shows an adequate performance due to the effective use of GPGPU in the parsing algorithm as well as in the neural network.

Table 4: Tools time performance on 100 sequences

	Elapsed time, seconds	
	Lengths 1-100	Lengths 1-200
Genegram	28	39
SPOT-RNA	68	235
Ipknot	1	1
Knotty	283	3050
RNAstructure	10	15
PknotsRG	15	95
HotKnots	37	366

4 Conclusion

We describe a new approach for RNA secondary structure prediction that employs the combination of formal grammars and neural networks. We provide an implementation, console tool Genegram, that allows to use several models designed for slightly different purposes and one can choose which model is more suitable for specific research. Genegram shows high quality and performance even compared with leading tools which makes it applicable for RNA secondary structure prediction. Moreover, our approach is quite flexible, because the final step of its pipeline is a neural network that has no limits in features presented in data, so, we can process such elements as pseudoknots, wobble base pairs, and multiplets if a proper, representable, and clean dataset is provided. And manipulating with grammar, network architecture and hyperparameters may reveal more possibilities of our approach in the secondary structure prediction filed.

We can provide several directions for future research. Firstly, all parameters tuning may lead to even better results for our models and we are planning to continue supporting and improving our tool, besides, it is possible that we will soon introduce the web application. Secondly, the current Genegram version can process only small RNA sequences having lengths up to 200 nucleotides, so, we need a solution for longer sequences processing. That requires searching for a big dataset of fully modeled high-quality long sequences and, perhaps, some architecture changes because bigger images processing is a non-trivial task demanding a lot of GPU memory. Moreover, we plan to run more experiments on RNA crystallography data from PDB database, especially concerning multiplets processing, in order to check the possibilities of our models in complicated secondary and tertiary features processing. And finally, in a long-term perspective, we want to adopt this approach for protein secondary structure prediction by similarly encoding main structural features using grammar and developing the corresponding neural network architecture.

References

1. R. B. Lyngsø and C. N. Pedersen, "Rna pseudoknot prediction in energy-based models," *Journal of computational biology*, vol. 7, no. 3-4, pp. 409–427, 2000.
2. E. Westhof, "Twenty years of rna crystallography," *Rna*, vol. 21, no. 4, pp. 486–487, 2015.
3. B. Fürtig, C. Richter, J. Wöhnert, and H. Schwalbe, "Nmr spectroscopy of rna," *ChemBioChem*, vol. 4, no. 10, pp. 936–962, 2003.
4. I. L. Hofacker and P. F. Stadler, "Automatic detection of conserved base pairing patterns in rna virus genomes," *Computers & chemistry*, vol. 23, no. 3-4, pp. 401–414, 1999.
5. F. Tahi, M. Gouy, and M. Régnier, "Automatic rna secondary structure prediction with a comparative approach," *Computers & chemistry*, vol. 26, no. 5, pp. 521–530, 2002.
6. S. Bellaousov, J. S. Reuter, M. G. Seetin, and D. H. Mathews, "Rnastructure: web servers for rna secondary structure prediction and analysis," *Nucleic acids research*, vol. 41, no. W1, pp. W471–W474, 2013.
7. E. Rivas and S. R. Eddy, "A dynamic programming algorithm for rna structure prediction including pseudoknots," *Journal of molecular biology*, vol. 285, no. 5, pp. 2053–2068, 1999.
8. J. Ren, B. Rastegari, A. Condon, and H. H. Hoos, "Hotknots: heuristic prediction of rna secondary structures including pseudoknots," *Rna*, vol. 11, no. 10, pp. 1494–1504, 2005.
9. J. Ruan, G. D. Stormo, and W. Zhang, "Ilm: a web server for predicting rna secondary structures with pseudoknots," *Nucleic acids research*, vol. 32, no. suppl_2, pp. W146–W149, 2004.
10. J. Reeder, P. Steffen, and R. Giegerich, "pknotsrg: Rna pseudoknot folding including near-optimal structures and sliding windows," *Nucleic acids research*, vol. 35, no. suppl_2, pp. W320–W324, 2007.
11. H. Jabbari, I. Wark, C. Montemagno, and S. Will, "Knotty: efficient and accurate prediction of complex rna pseudoknot structures," *Bioinformatics*, vol. 34, no. 22, pp. 3849–3856, 2018.

12. K. Sato, Y. Kato, M. Hamada, T. Akutsu, and K. Asai, "Ipknot: fast and accurate prediction of rna secondary structures with pseudoknots using integer programming," *Bioinformatics*, vol. 27, no. 13, pp. i85–i93, 2011.
13. K. Sato, M. Hamada, K. Asai, and T. Mituyama, "Centroidfold: a web server for rna secondary structure prediction," *Nucleic acids research*, vol. 37, no. suppl_2, pp. W277–W280, 2009.
14. B. Knudsen and J. Hein, "Rna secondary structure prediction using stochastic context-free grammars and evolutionary history.," *Bioinformatics (Oxford, England)*, vol. 15, no. 6, pp. 446–454, 1999.
15. R. D. Dowell and S. R. Eddy, "Evaluation of several lightweight stochastic context-free grammars for rna secondary structure prediction," *BMC bioinformatics*, vol. 5, no. 1, pp. 1–14, 2004.
16. M. Akiyama, K. Sato, and Y. Sakakibara, "A max-margin training of rna secondary structure prediction integrated with the thermodynamic model," *Journal of bioinformatics and computational biology*, vol. 16, no. 06, p. 1840025, 2018.
17. C. B. Do, D. A. Woods, and S. Batzoglou, "Contrafold: Rna secondary structure prediction without physics-based models," *Bioinformatics*, vol. 22, no. 14, pp. e90–e98, 2006.
18. 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*, vol. 10, no. 1, pp. 1–13, 2019.
19. B. Apolloni, L. Lotorto, A. Morpurgo, and A. Zanaboni, "Rna secondary structure prediction by mft neural networks," 2003.
20. S. Grigorev. and P. Lunina., "The composition of dense neural networks and formal grammars for secondary structure analysis," in *Proceedings of the 12th International Joint Conference on Biomedical Engineering Systems and Technologies - Volume 3 BIOINFORMATICS: BIOINFORMATICS,*, pp. 234–241, INSTICC, SciTePress, 2019.
21. P. Lunina and S. Grigorev, "On secondary structure analysis by using formal grammars and artificial neural networks," in *International Meeting on Computational Intelligence Methods for Bioinformatics and Biostatistics*, pp. 193–203, Springer, 2019.
22. E. Rivas and S. R. Eddy, "The language of rna: a formal grammar that includes pseudoknots," *Bioinformatics*, vol. 16, no. 4, pp. 334–340, 2000.
23. M. Quadrini., E. Merelli., and R. Piergallini., "Loop grammars to identify rna structural patterns," in *Proceedings of the 12th International Joint Conference on Biomedical Engineering Systems and Technologies - Volume 3 BIOINFORMATICS: BIOINFORMATICS,*, pp. 302–309, INSTICC, SciTePress, 2019.
24. K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.
25. R. Azimov and S. Grigorev, "Context-free path querying by matrix multiplication," in *Proceedings of the 1st ACM SIGMOD Joint International Workshop on Graph Data Management Experiences & Systems (GRADES) and Network Data Analytics (NDA)*, GRADES-NDA '18, (New York, NY, USA), Association for Computing Machinery, 2018.
26. F. Chollet *et al.*, "Keras." https://keras.io, 2015.
27. M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané,

R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

28. M. Andronescu, V. Bereg, H. H. Hoos, and A. Condon, "Rna strand: the rna secondary structure and statistical analysis database," *BMC bioinformatics*, vol. 9, no. 1, pp. 1–10, 2008.
29. F. Van Batenburg, A. P. Gultyaev, C. Pleij, J. Ng, and J. Oliehoek, "Pseudobase: a database with rna pseudoknots," *Nucleic Acids Research*, vol. 28, no. 1, pp. 201–204, 2000.
30. J. J. Cannone, S. Subramanian, M. N. Schnare, J. R. Collett, L. M. D'Souza, Y. Du, B. Feng, N. Lin, L. V. Madabusi, K. M. Müller, *et al.*, "The comparative rna web (crw) site: an online database of comparative sequence and structure information for ribosomal, intron, and other rnas," *BMC bioinformatics*, vol. 3, no. 1, pp. 1–31, 2002.
31. M. Sprinzl, C. Horn, M. Brown, A. Ioudovitch, and S. Steinberg, "Compilation of trna sequences and sequences of trna genes," *Nucleic acids research*, vol. 26, no. 1, pp. 148–153, 1998.
32. C. Zwieb and N. Larsen, "The signal recognition particle (srp) database," *Nucleic acids research*, vol. 20, no. Suppl, p. 2207, 1992.
33. S. Griffiths-Jones, A. Bateman, M. Marshall, A. Khanna, and S. R. Eddy, "Rfam: an rna family database," *Nucleic acids research*, vol. 31, no. 1, pp. 439–441, 2003.
34. H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne, "The protein data bank," *Nucleic acids research*, vol. 28, no. 1, pp. 235–242, 2000.