The Composition of Dense Neural Networks and Formal Grammars for Secondary Structure Analysis

Semyon Grigorev[1,2], Polina Lunina[1,2]
[1]*St. Petersburg State University, 7/9 Universitetskaya nab., St.Petersburg, Russia*
[2]*JetBrains Research, Universitetskaya emb., 7-9-11/5A, St.Petersburg, Russia*
s.v.grigoriev@spbu.ru, Semen.Grigorev@jetbrains.com, lunina_polina@mail.ru

Keywords: Dense Neural Network, DNN, Machine Learning, Secondary Structure, Genomic Sequences, Protenomic Sequences, Formal Grammar, Parsing

Abstract: Abstract is very abstract.

1 INTRODUCTION

Accurate, fast, precise sequences classification and subsequences detection are an open problems in different areas of bioinformatics, such as genomics and protenomics. Challenge here is a high variability of sequences which are the same class (or one want to mark as a same class). It is the fact that for some type of sequences its secondary structure is a principal (**?**) and even conservative (**?**). These facts may be used for sequences processing.

For example, algorithms that can efficiently and accurately identify and classify bacterial taxonomic hierarchy have become a focus in computational genomics. The idea that secondary structure of genomic sequences is sufficient for solving the detection and classification problems lies at the heart of many tools (Rivas and Eddy, 2000; Knudsen and Hein, 1999; Yuan et al., 2015; Dowell and Eddy, 2004). !!!! One of the way to specify the secondary is to use formal grammars. The problem here is taht the sequences obtained from the real bacteria usually contain a huge number of mutations and noise which renders precise methods impractical. Probabilistic grammars (**?**) and covariance models (CMs) are a way to take the noise into account (Durbin et al., 1998). For example, CMs are successfully used in the Infernal tool (Nawrocki and Eddy, 2013). Neural networks is another way to deal with noisy data. The works (Sherman, 2017; Higashi et al., 2009) utilize neural networks for 16s rRNA processing and demonstrate promising results.

Long-distance-contacts!!! (long sequences, chemerics, etc) HMMs fails. Probabilistic grammars trainig is hard.

In this work we propose the way to combine formal grammars and neural networks for secondary structure features processing. The key idea is does not try to model full (sub)sequence of interest by grammar, but create grammar which describes features of secondary structure and use neural network for these features processing. We provide results of our evaluation which demonstrates !!!

2 PROPOSED SOLUTION

We propose to combine neural networks and ordinary context-free grammars (not probabilistic which are usually used in this area) in order to handle information of sequences' secondary structure. Namely, we propose to extract secondary structure features by using the ordinary context-free grammar and use the dense neural network for features processing. Features can be extracted by any parsing algorithm and then presented as a boolean matrix but we choose parsing algorithm based on matrix multiplication.

In this section we describe all components of our recipe and provide some examples end explanations on it.

2.1 Context-Free Grammars

The first component is a context-free grammar. It is a well-known fact that secondary structure of sequence may be approximated by using formal grammars. There is number of works that utilize this fact for different purposes (**?**).

Probabilistic context-free grammars are usually used for secondary structure modeling because it allows to dealt with variations (mutations or some kinds of noise). In the opposite of it, we use ordinary (not probabilistic) grammars. Our goal is not to model secondary structure of whole sequence (which required probabilistic grammars), but describe features of secondary structure, such as stems, loops, pseudoknots and it's composition. Of course, the set of feature types is limited by class of the grammar which we use. For example, pseudoknots can not be expressed by context-free grammars, but can be expresed by using cinjunctive (Devi and Arumugam, 2017; Zier-Vogel and Domaratzki, 2013; Okhotin, 2001) or multiple context-free (Seki et al., 1991; Riechert et al., 2016).

The context-free grammar G_0 which we use in our experiments is presented in figure 1. It is a context-free grammar over four-letters in the alphabet $\Sigma = \{A, C, G, T\}$ with start nonterminal `s1`. This grammar which describes composition of stems with bounded minimal height.

First of all, we provide a brief description of grammar specification language. Left hand side and right hand side of rule are separated by the `:` sign. In the right hand side one can use extended regular expressions over union alphabet of terminals and nonterminals. Such constructions as bounded repetition and alternative are available. For example, `any*[2..10]` is a bounded repetition and it is stands that nonterminal `any` may be repeated any number of times from 2 up to 10. Example of rule which uses alternatives is `any: A | T | C | G` which stands that `any` is one of four terminals.

Another important feature of the language is parametric rules or metarules which allow one to create a reusable grammar templates. More details on metarules one can find in (**?**). The example of metarule in our grammar is `stem1<s>: A s T | G s C | T s A | C s G`. This rule has one parameter `s` which stands for something that should be embedded into stem. Application of this rule to `any_str` allow one to define stem with loop of length from 2 up to 10. In our grammar we use metarules in order to describe stems with bounded minimal height: `stem1<s>` is a stem with height exactly 1, `stem2<s>` is a stem with height exactly 2, and `stem<s>` is a stem with height not lower than 3.

```
s1: stem<s0> any

any_str : any*[2..10]

s0: any_str | any_str stem<s0> s0

any: A | T | C | G

stem1<s>: A s T | G s C | T s A | C s G

stem2<s>: stem1< stem1<s> >

stem<s>:
      A stem<s> T
    | T stem<s> A
    | C stem<s> G
    | G stem<s> C
    | stem1< stem2<s> >
 }
```

Figure 1: Context-free grammar G_0 for RNA secondary structure features extraction

Now we explain what does this grammar meens. This grammar describe a recursive composition of stems. To see it one can look at the rule for `s0` which is recursive and shows that composition of stems may be embedded into stem (`|stem<s0>|` in the right side of this rule). Every stem should has height not lower then 3 and builds only from classical base pairs. Stems may be connected by arbitrary sequence of length from 2 up to 10, and loops have the same length. Graphical explanation of this description one can find in figure 2.

Note that grammar is a variable parameter and may be tuned for specific cases. The grammar presented above is a result of set of experiments, so there are no reasons to stend that it is the best grammar for scondary structure features extraction. For example, one can vary length of unfoldable sequence by changing rule for `any_str`: `any_str : any*[0..10]`, `any_str : any*[1..8]`, or something else. Also one can increase (or decrease for some reason) the minimal height of stem, or add some new features, such as pseudoknots, in the grammar (in case of usage of conjunctive grammars instead of context-free one).

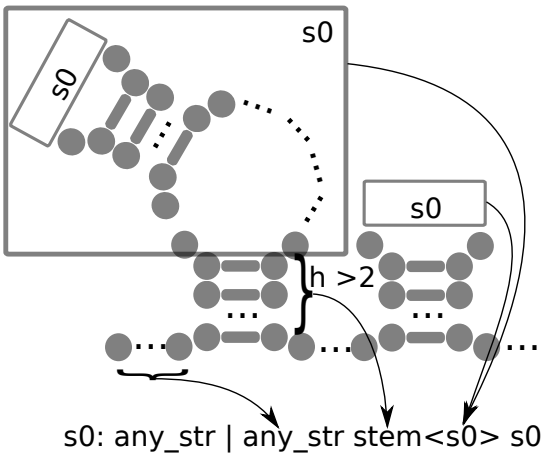

Figure 2: Graphical explanation of pattern which is described by grammar in the figure 1

2.2 Parsing Algorithm

In the classcal scenario parsing is used for answering the question wheter or not given sequence derivable in the given grammar. Additionally, in case if sequence is derivable, derivation tree may be provided as a result of parsing. It is a classical way: there is a huge number of works on modelling secondary stricture of full sequence of interest by using probabilistic grammars and respective parsing techniques (**?**). We propose to use parsing as a feature extraction: we want to find all derivable substrings of given string for all nonterminals, not to check derivability of given string or find the most probably derivation. So, we use undirected parsing.

CYK is a classical well-known algorithm for undirected parsing. This algortithm and is modofocation is used in big number of works (**?**), but it demonstrates poor performance on realistic input.

An alternative approach are algortihms which are based on matrix multiplication, such as Valiant's algorithm (Valiant, 1975) which provides subcubic parsing.

In our work we use another version of matrixbased algorithm (Azimov and Grigorev, 2018). Theoretical time complexity of this algorithm is worthe than complexity of the Valiant's algorithm, but in practice these algorithm avoid machinery on submatrices manipulation and demonstrate better performance with simple implementation.

From the practical point of view, matrix-based algorithms allows to easely utilise advanced tecniques, such as algorithms for sparse matrices, and for boolean matrices, GPGPU-based libraries, etc.

Moreover, matrix-based approach can be generalized to conjunctive and even boolean grammars (Okhotin, 2014), as far as to multiple context-free grammars (Cohen and Gildea, 2016), which can provide a base for more expressive features descriptions handling without significant changes in other parts of our solution.

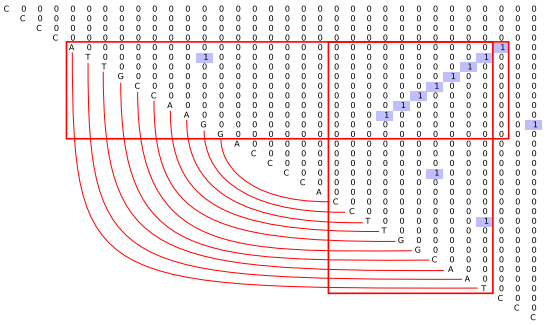

Figure 3: Parsing result for sequence which should folds to stem

2.3 Matrices

The result of parsing is a set of square boolean matrices. Each matrix M_N contains information of all substrings which can be derived from nontermnal N. In the other word, $M_N[i,j] = 1$ iff $N \Rightarrow^*_G w[i,j-1]$ where w is the input sequence and G is context-free grammar, and N is a nonterminal. Thus, result of parsing is a set of matrices: one matrix for each nonterminal form grammar. For futher processing we can select nonterminals of interest. For our case, for grammar G_0 we select matrix for nonterminal `s1`.

The example of such matrix is provided in figure 3. This matrix is a result of parsing of the sequence

w_1 =CCCCATTGCCAAGGACCCCACCTTGGCAATCCC

w.r.t the grammar G_0. In the figure one can see upper right triangle of parsing matrix (bottom left is always empty, so omitted) with input string on te diagonal. Note that string is added only for example readability and real matrix does not contains input string, only results of its parsing. Each filled cell $[i,j]$ which contains 1 is denote that subsequence $w_1[i,j-1]$ is derivable form `s1` in G_0 (so, this subsequence folds to stem with heigh 3 or more). In order to find stems with heght more than 3 one should detect diagonal chains of 1-s: in our example stem has height equals 10 and one can find chain of 1-s of the length $8 = 10-2$ (first 1 is a root of the stem of heght 3 and each next 1 is a new base pair upon this stem — root of the stem with height increased by one). Red boxes and contact map are added for navigation sinplification.

Our goal is to extract all features of secondary

structure, so our parser finds all substrings which can be derived from `s1`. As a result there are some 1-s out of the chain. These are correct results: corresponded subsequences can be derived from `s1`. In the current exmaple these 1-s may be treated as noise in some sense, but, as we show late, such behaviour may be useful in some cases. Moreover, for long sequences with complex structure it may be not evident, which feachures of secondary structure are principal.

We use these matrices as an input for artifactal newural network which should detect sufficient features (long chain in our example) and utilize these for applied problem solution (sequence detection or classification, for example). We drop out bottom left triangle and vectorize matrices row-by-row in order to get bit vector which then converts to byte vector and uses as an input. Transition from bit vector to byte vector is done in order to decrease size of the input which is critical for long sequences. On the other hand, such operation may significantly complicate network architecture and training, and it is a reason to try to use bitwise networks (Kim and Smaragdis, 2016) in the future.

2.4 Artifactal Neural Networks

Artifactal neural networks is one of possible choices for different classification problems in case when data has hard-to-formalize principal for problem features and contains some kindes of noise. Different types of networks are successfully utilized for images, speach, natural languages processing.

Classical scenario for classification problems is to provide features vectors and try to classify them which meens that network can select important features for each requied class. In our case the fact that $w[i, j-1]$ is derivable from nonterminal N which is encoded in the matrix is exactly a feature. So, vectorized matrix is a vector of features which is a typical input for newral network.

We use dense neural network because data locality is broken during vectorization and any convolutions is inapplicable. Moreover, convolutions are used mostly for features extraction, but in out case features are already extracted by parsing. Thus we need only to detect proncipal features and reletions between them. And it is a typical area for dense networks.

One of the problem with arbitrary data processig by using neural networks is input size normalization. Inpiut layer of network has fixed size, but input sequence lengtah and hance length of vectorized parsing result may be varied even for fixed task. For example, length of tRNA may be approximately from 59 upto 250. We propose two possible ways to solve this

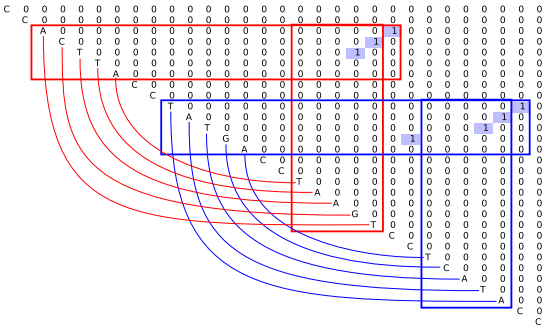

Figure 4: Parsing result for sequence which should folds to pseudoknot

problem. The first one is subsequences processing: for some tasks it may be anough to process only subsequence, not full sequence. Here we can fix length of subsequence lower than shortest sequence which we want to handle. The second one is to fix an upper bound and filling gap with special symblos. For example, if we want to handle tRNAs we can fix input lenght eqals 250 and when we want to process sequence of length 60, then we should fill rest 190 symbols by selected special symbol.

The example of neural network which we use is presented in figure 8. We actively use dropout and batch normalization because network should perform a number of nontrivial transformations: decompress data from bytes and prepare normalized input which requre additional power. Despite the fact that initially batch normalization is an alternative for dropout (Ioffe and Szegedy, 2015), we use both of themtogether because separately using has now effects.

3 Examples

Here we provide more examples of matrices and point out some observations about it in order to provide better intuitoin on our idea.

First is observation about pseudoknots. Let consider the next sequence which can folds to pseudoknot as an example:

$w_2 =$`CCACTTACCTATGACCTAAGTCCTCATACC.`

Note, that loops are very short for example minimization. As mentioned above, pseudoknots can not be expressed in terms of context-free grammars. But one can meen pseudoknot as a two crossing stems in some sense, and parser can extraxt both of them, as presened in the figure 4. So, if newural network is powerful enough, then it can detect that if these two fetures occurce simulatenously, then sequence contains pseudocnot. As a result, we can detect features

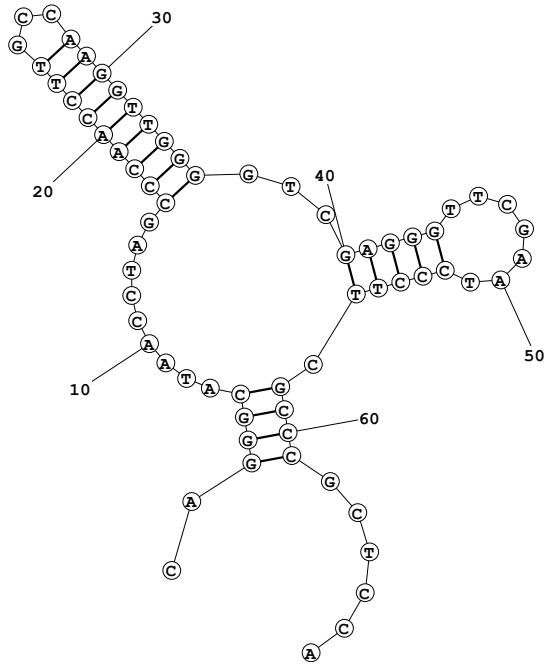

Figure 5: Predicted secondary structure for w_3

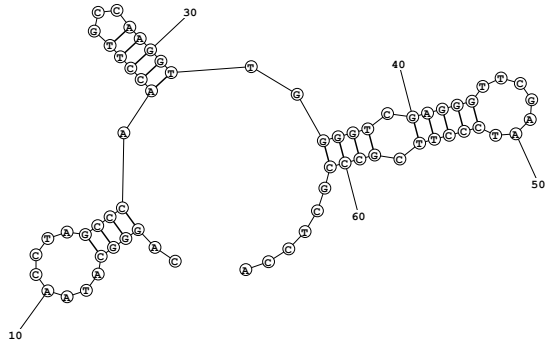

Figure 6: Predicted secondary structure for w_3

which are not expressible in context-free grammars by usin proposed way.

The second is an example of matrix for real tRNA. Parsing result of the tRNA[1] sequence

w_3 =CAGGGCATAACCTAGCCCAACCTTGCCAAGG
TTGGGGTCGAGGGTTCGAATCCCTTCGCCCGCTCCA

is presented in the figure 7. Also, one can see predicted secondary structures[2] (top two) in the figures 5 and 6.

Colored boxes in te figure 7 marks features which correspond to these two predicted foldings: blue marks for 5 and red for 6. Note, that our grammar G_0 handles only classical base pairs, so the pair `G - T` which exists in predicted foldings, is not presented in parsing result. Anyway, we can see, that all expected information on secondary structure is presented in the matrix, of couce, with some additional features. And it is a field for neural networks — to select appreprate features.

Thus we can conclude, that very nontrivial compositions of secondary structure features may be detected by using a powerful enough neral networks. It is an interestin question for future reserach: what kinds of applications may be built by using such results?

4 EVALUATION

We evaluate the proposed approach on two cases: 16s rRNA detection and tRNA classification. Note that goal of the evaluation is to demonstrate applicability of approach which describd above. So, we are not provide comparison with existing tools and we do not try to solve real problems. All of tese are tasks for future work.

4.1 16s rRNA Sequences

The first problem is 16s rRNA detection. We specify context-free grammars which detect stems with the hight of more than two pairs and their arbitrary compositions (namely, G_0). For network training we use dataset consisting of two parts: random subsequences of 16s rRNA sequences from the Green Genes database (DeSantis et al., 2006) form positive examples, while the negative examples are random subsequences of full genes from the NCBI database (Geer et al., 2010). All sequences have the length of 512 symbols, totally up to 310000 sequences. After training, current accuracy is 90% for validation set (up to 81000 sequences), thus we conclude that our approach is applicable.

4.2 tRNA Sequences

The second problem is tRNA classification: we train neural network to separate tRNAs into two clsses: prokaryote and eukaryote. We prepare 50000 sequences from GtRNADB (Chan and Lowe, 2009) for training: 35000 for training and 15000 for test. In this case we use the next trick for data size normalization. We set the upper bound of sequence length equals 220

[1]Novosphingobium_aromaticivorans_DSM_12444_chr.trna57-GlyGCC (268150-268084) Gly (GCC) 67 bp Sc: 22.97. From GtRNAdb: `http://gtrnadb2009.ucsc.edu/download.html`. Access date: 02.11.2018.

[2]Predicted secondary structures are given by using the Fold Web Server with default settings: `http://rna.urmc.rochester.edu/RNAstructureWeb/Servers/Fold/Fold.html` Access date: 02.11.2018.

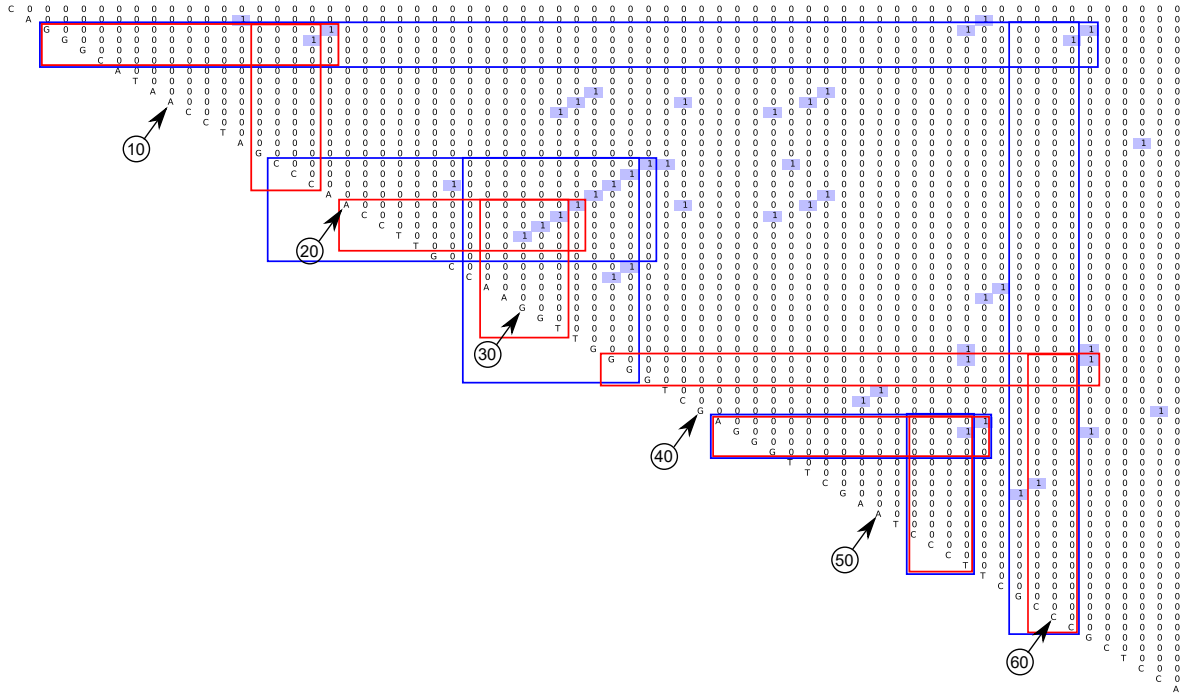

Figure 7: Parsing result for real tRNA (w_3)

and after that we aligne real tRNA sequence w by te next way: first k symbols of the input is w ($|w| = k$) and the rest $220 - k$ symbols are filled by `D` — a special symbol which is not in input alphabet.

Also we prepare validation set which contains 217984 sequences for pro and 62656 sequences fro euk. All data for validation given from tRNADB-CE[3] (Abe et al., 2010).

The architecture of network which we use in this experiment is presented in the figure 8. Note that it is a training configuretion: it contains dropout and batch normalization layers which will be removed after training. This network contains six dense layers and use `relu` and `sigmoid` activation functions.

After training our network demonstarte accuracy is 97%. For validation set we get the next results: 3276 of eukaryote (5.23% of all eukaryote) are classified as prokaruote and 4373 of prokaryote (2.01% of all prokaryote) are classified as eukaryote.

As a result, we can conclude that input normalizatoin by filling sequence to upper bound of length by special symbol works. Also we can conclude thet secondary structure contains sufficient information for classification.

[3]tRNADB-CE: tRNA gene database curated manually by experts. URL: `http://trna.ie.niigata-u.ac.jp/cgi-bin/trnadb/index.cgi`. Access date: 31.10.2018

5 DISCUSSION AND FUTURE WORK

The presented is a work in progress. The ongoing experiment is finding all instances of 16s rRNA in full genomes. Also we plan to use the proposed approach for the filtration of chimeric sequences and the classification. Composition of our approach with other methods and tools as well as grammar tuning and detailed performance evaluation may improve the applicability for the real data processing.

One of prolem of proposed approach is ithat parsing is a bottlenack of performance. Possible solution is to construct network which can handle sequences, not parsing data. It may be done by the next way.

1. Create a trainig set of matrices by using parsing.
2. Build and train the network NN_1 which handle vectorized matrices.
3. Create new network NN_2 by extending of NN_1 with head (set of layers) which should convert sequence to input for NN_1
4. Train NN_2. Weights of layers from NN_1 should be fixed.
5. For concrete problem we can tune weights of NN_2 after second trained to appropriate quality.

This way we can use parsing only for training which is less performance critical stage than using.

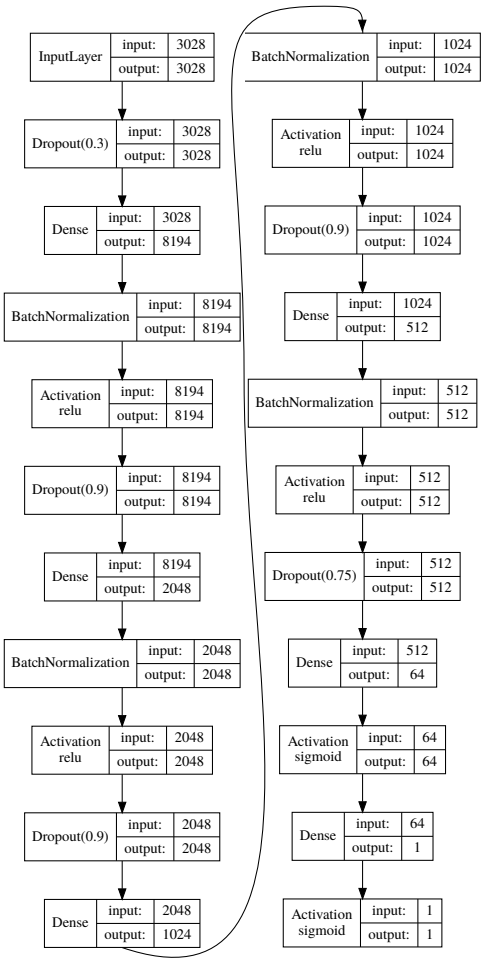

Figure 8: architecture of the neural network for tRNA classification

Anoter task is an understending features whish network extracts in order to get inspiration in grammar tuning, for example. It may be done by trained network visualization. There is a set of tools for user-friendly convolutional networks visualisation, but not for dense networks. It may be useful to create such tool and customize it for our bomain.

We do some experints in genomic sequence analysis, but what about protenomics? There are works on grammar-based approaches to protenomics seqences analysis (Witold Dyrka) (Dyrka et al., 2018) This area provides new challenges, such as more complex grammar, more symbols in alphabet, more complex rules of interactions, more complex featres. As a result, more powerful languages may be required in this area. So, it may be intersting to applay proposed approach for protenomics sequences analysis. One of the possible crutial problem is to detect functionally equivalence sequences with sufficiently different length.

Also it may be interesting to use other types of neural networks. Bitwise networks (Kim and Smaragdis, 2016) may de reasonable because the result of parsing is a bitwize matrix, so it looka like natural way to use these networks to process such result. Another direction is convolutional networks utilization. One can treat parsing matices as a bitmaps: one can set a specific color for each nonterminal and get multicolor picture as a sum of matrix. The problem here is a picture size: typical matrix size is $n \times n$ where n is a length of the input sequence.

Important part of work is a training data preparation. One of the dificult problems is a balanced data set creation. Biological datasets (like GreenGenes) contains huge nomder of samples for some well-studied organisms and very small number of samples for other. Moreover, datasets often contains unclassified and candidatus sequences. It is not evident how we should prepare datasets in order to get high-quality trained network.

To conclude, our work in the early beginning stage and current results is promissing. There is a huge number of experiments in different directions which may be potentially interesting. In order to choose a right direction we hope to discuss future work with the community.

ACKNOWLEDGEMENTS

The research was supported by the Russian Science Foundation grant 18-11-00100 and a grant from JetBrains Research.

REFERENCES

Abe, T., Ikemura, T., Sugahara, J., Kanai, A., Ohara, Y., Uehara, H., Kinouchi, M., Kanaya, S., Yamada, Y., Muto, A., and Inokuchi, H. (2010). tRNADB-CE 2011: tRNA gene database curated manually by experts. *Nucleic Acids Research*, 39(Database):D210–D213.

Azimov, R. and Grigorev, S. (2018). 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, pages 5:1–5:10, New York, NY, USA. ACM.

Chan, P. P. and Lowe, T. M. (2009). GtRNAdb: a database of transfer RNA genes detected in genomic sequence. *Nucleic Acids Research*, 37(Database):D93–D97.

Cohen, S. B. and Gildea, D. (2016). Parsing linear context-free rewriting systems with fast matrix multiplication. *Computational Linguistics*, 42(3):421–455.

DeSantis, T. Z., Hugenholtz, P., Larsen, N., Rojas, M., Brodie, E. L., Keller, K., Huber, T., Dalevi, D., Hu, P., and Andersen, G. L. (2006). Greengenes, a chimera-checked 16S rRNA gene database and workbench compatible with ARB. *Appl. Environ. Microbiol.*, 72(7):5069–5072.

Devi, K. K. and Arumugam, S. (2017). Probabilistic conjunctive grammar. In *Theoretical Computer Science and Discrete Mathematics*, pages 119–127. Springer International Publishing.

Dowell, R. D. and Eddy, S. R. (2004). Evaluation of several lightweight stochastic context-free grammars for rna secondary structure prediction. *BMC bioinformatics*, 5(1):71.

Durbin, R., Eddy, S. R., Krogh, A., and Mitchison, G. (1998). *Biological sequence analysis: probabilistic models of proteins and nucleic acids*. Cambridge university press.

Dyrka, W., Coste, F., and Talibart, J. (2018). Estimating probabilistic context-free grammars for proteins using contact map constraints. *CoRR*, abs/1805.08630.

Geer, L. Y., Marchler-Bauer, A., Geer, R. C., Han, L., He, J., He, S., Liu, C., Shi, W., and Bryant, S. H. (2010). The NCBI BioSystems database. *Nucleic Acids Res.*, 38(Database issue):D492–496.

Higashi, S., Hungria, M., and Brunetto, M. (2009). Bacteria classification based on 16s ribosomal gene using artificial neural networks. In *Proceedings of the 8th WSEAS International Conference on Computational intelligence, man-machine systems and cybernetics*, pages 86–91.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167.

Kim, M. and Smaragdis, P. (2016). Bitwise neural networks. *CoRR*, abs/1601.06071.

Knudsen, B. and Hein, J. (1999). Rna secondary structure prediction using stochastic context-free grammars and evolutionary history. *Bioinformatics (Oxford, England)*, 15(6):446–454.

Nawrocki, E. P. and Eddy, S. R. (2013). Infernal 1.1: 100-fold faster RNA homology searches. *Bioinformatics*, 29(22):2933–2935.

Okhotin, A. (2001). Conjunctive grammars. *J. Autom. Lang. Comb.*, 6(4):519–535.

Okhotin, A. (2014). Parsing by matrix multiplication generalized to boolean grammars. *Theoretical Computer Science*, 516:101 – 120.

Riechert, M., Höner zu Siederdissen, C., and Stadler, P. F. (2016). Algebraic dynamic programming for multiple context-free grammars. *Theor. Comput. Sci.*, 639(C):91–109.

Rivas, E. and Eddy, S. R. (2000). The language of rna: a formal grammar that includes pseudoknots. *Bioinformatics*, 16(4):334–340.

Seki, H., Matsumura, T., Fujii, M., and Kasami, T. (1991). On multiple context-free grammars. *Theoretical Computer Science*, 88(2):191 – 229.

Sherman, D. (2017). Humidor: Microbial community classification of the 16s gene by training cigar strings with convolutional neural networks.

Valiant, L. G. (1975). General context-free recognition in less than cubic time. *J. Comput. Syst. Sci.*, 10(2):308–315.

Yuan, C., Lei, J., Cole, J., and Sun, Y. (2015). Reconstructing 16s rrna genes in metagenomic data. *Bioinformatics*, 31(12):i35–i43.

Zier-Vogel, R. and Domaratzki, M. (2013). Rna pseudoknot prediction through stochastic conjunctive grammars. *Computability in Europe 2013. Informal Proceedings*, pages 80–89.