Secondary structure prediction of RNA using convolutional neural networks

Bisera Chauleva*

Department of Computer Science and Engineering,
University for Information Science and Technology
'St. Paul the Apostole' Ohrid, North Macedonia
Email: bisera.cauleva@cse.uist.edu.mk
*Corresponding author

Atanas Hristov

Department of Computer Systems, Complex and Networks,
University for Information Science and Technology
'St. Paul the Apostole' Ohrid, North Macedonia
Email: atanas.hristov@uist.edu.mk

Ustijana Rechkoska Shikoska

Department of Electrical Engineering and Computer Science,
University for Information Science and Technology
'St. Paul the Apostole' Ohrid, North Macedonia
Email: ustijana.r.shikoska@uist.edu.mk

Ljubinka Gjergjeska Sandjakoska

Department of Machine Learning,
University for Information Science and Technology
'St. Paul the Apostole' Ohrid, North Macedonia
Email: ljubinka.gjergjeska@uist.edu.mk

Abstract: One of the most popular bioinformatics and genetics topics is the secondary structure prediction of RNA since it is the first step in developing new therapeutic and pharmacological methods. The crucial point in these branches is the development time and accuracy. The work presented in this paper will look upon the state-of-the-art techniques that are currently used for prediction. The deep learning method is evaluated to speed up and to increase accuracy level. Furthermore, modern convolutional neural networks known as residual network, as a secondary structure prediction of RNA, are considered as a more accurate and efficient approach. Finally, a comparison of evaluating results with existing methods is performed, which will prove the successfulness of the proposed algorithm, taken into consideration the benchmark evaluation.

Keywords: ribonucleic acid; RNA; secondary structure prediction; bioinformatics; artificial intelligence; deep learning; convolutional neural networks; PyTorch; classification; residual networks; ResNet; artificial neural networks.

Reference to this paper should be made as follows: Chauleva, B., Hristov, A., Shikoska, U.R. and Sandjakoska, L.G. (2022) 'Secondary structure prediction of RNA using convolutional neural networks', *Int. J. Student Project Reporting*, Vol. 1, No. 1, pp.43–67.

Biographical notes: Bisera Chauleva obtained her Bachelors degree from the Faculty for Computer Science and Engineering, University of Information Science and Technology 'St. Paul the Apostole' Ohrid, N. Macedonia. She is currently an MSc student. Her research interest is centred on the usage of the machine learning algorithms for the purpose of solving bioinformatic's problems. She has one research paper considering parallelisation for speed up of RNA secondary structure prediction methods.

Atanas Hristov obtained his Doctors degree from Computer Systems and Networks, Technical University of Sofia in 2013 and worked over Post-Doctoral Fellowship in the University of Antwerpen, Belgium September 2015–June 2016. He has coauthored over 25 publications interested into supercomputing, parallelisation, big data, algorithms for AI and ML, networks etc. He works as an Associate Professor for subjects of: programming, operating systems, cryptography, network architectures, communication protocols, parallel programming, big data analytics, high-performance computing at University of Information Science and Technology 'St. Paul the Apostole' Ohrid, N. Macedonia.

Ustijana Rechkoska Shikoska obtained her Doctoral degree from the Department of Electrical Engineering and Computer Science from University of 'Kiril and Metodius' Skopje, N. Macedonia. She has coauthored more than 35 publications with consideration of areas such as: databases, algorithms, computer communication networks, internet security etc. She also works as an Associate Professor as well as the Dean of Faculty for Computer Science and Engineering at Univeristy for Information Science and Technology 'St. Paul the Apostole' Ohird, N. Macedonia.

Ljubinka Gjergjeska Sandjakoska obtained her MSc degree from University Ss Cyril and Methodius, Faculty of Mechanical Engineering – Skopje where she is working on Doctors degree considering Artificial Networks and Machine Learning. She has coauthored more than 15 research papers considering different topics among which: big data analytics, deep neural networks, cognitive computing, drug development etc. She works as an assistant on the University of Information Science and Technology 'St. Paul the Apostole' Ohrid, N. Macedonia, for the subjects: discrete mathematics, pattern recognition, artificial intelligence, operation research, and knowledge based systems.

1 Introduction

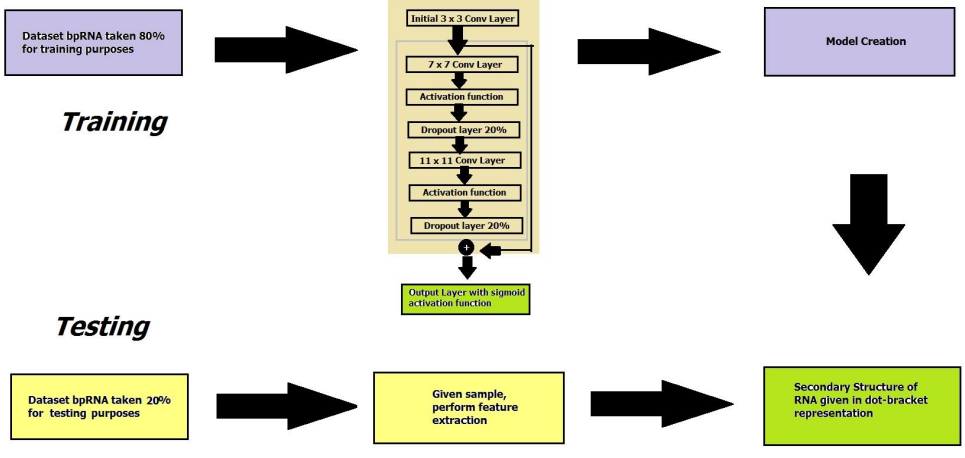
Ribonucleic acid (RNA) as a complex molecule, takes function in cellular protein synthesis, additionally acting as deoxyribonucleic acid (DNA) in some viruses. RNA is known for the importance as element after DNA, based on the usage in encoding, and decoding genes, as well as regulation of their expression in the living organisms. Since RNA receptors and its final structure are detected with tertiary structure formation, secondary structure of the RNA must be used.

RNA is constructed of ribose nucleotides connected with phosphor-di-ester bonds forming strands of varying lengths. The nitrogenous bases in RNA are adenine, guanine, cytosine, and uracil, where the uracil replaces thymine in DNA. The presence of self-complementary sequences in the RNA strand leads to internal base-pairing and folding onto itself. The chain forms complex structures that consist of different kinds of loops, which is the aim of the prediction algorithm.

Starting from the beginning, X-ray diffraction and nuclear magnetic resonance (NMR) were first methods used for structure prediction of RNA. Later, due to inefficiency and impracticality, new methods based on dynamical programming were developed. Some of the mostly used methods in practice are: Nussinov-Jacobson algorithm (Zhao and Sahni, 2016), Zuker's (Palkowski and Bielecki, 2016) MFE algorithm, maximum expected accuracy (MEA) algorithm (Clote et al., 2017), pseudoknotted algorithms (Mustoe and Buasan, 2018), etc.

However, most of these methods have enough accuracy, and speed in the prediction process, but they are falling down once working with higher amount of sequences, as well as longer nucleotides are taken in consideration.

Figure 1 Flowchart of network created for secondary structure prediction of RNA, upper blocks are showing the training process and creation of model, lower part blocks for testing purposes (see online version for colours)



Since, for solving this problem, following paper proposes CNN approach based on a more modern type known as residual networks (ResNet). This approach will be tested over 10.000 sequences with lengths of around 1,000 nucleotides, extracted from bpRNA database (Danaae et al., 2018). Data being used considers dot bracket representation of secondary structures. In order to perform faster and more accurate prediction, the data will be prepared within embedded layer to suit a dictionary approach. The main model considers three convolutional layers, yet inspired by method proposed in Singh et al. (2019), the proposed model uses (ResNet) approach, for the purpose of proper classification as well as for avoiding vanishing gradient levels. The main reason behind ResNet as type of CNN used is the immersive capability to solve very complex problems that require very large amount of layers in a very efficient way, not affecting time and

accuracy. The concept of constructing CNN for the problem of finding secondary structure of RNA is given with the following flowchart.

Residual learning or ResNet are constructions that mainly prefer skipping some connections with the possibility to skip some layers. These ResNet models are mainly designed for the purpose of double or triple layer skips instead of using consecutive layers which is the most often type of deep neural networks (DNN). Skipping over layers provides us with the possibility to avoid the vanishing gradient problem.

In this study, we are using residual learning type of CNN for optimisation of network parameters. The study uses concept proposed by He et al. (2015), where initial convolution layer is with size of 3 by 3 followed by ResNet block that consists of two convolutional layers considering kernel sizes of 11 by 11 and 7 by 7 respectively. The function being used is the rectified linear unit (ReLU) nonlinear activation function whereas for the dropout regularisation the specified value is 20% for purpose of avoiding the overfitting case.

2 Related work

In the following section a review of the methods that are already used by the research community has been provided, starting from the oldest to the newest and most advanced approaches. All the advancements followed by details that are going to find usage in building the CNN architecture represented in this paper, are going to be discussed (Chauleva, et al., 2020).

- Nussinov-Jacobson algorithm (Zhao and Sahni, 2016) proposed by Nussinov and Jacobson in 1978, has been defined as algorithm for secondary RNA structure prediction based on the folding principle or also known as single structure prediction approach, where RNA strand is folding onto itself, without taking in consideration complex formations like pseudoknots (stem-loops present in structure).

  The main version of the algorithm uses the maximum amount of base pairs for optimising of the score. Standard approach of implementation is interested in the usage of 2D array.

- The score considers *Xi* and *Xj* values, which are stored in a form of a matrix $M[i][j]$ (Palkowski and Bielecki, 2016).

- Zuker's minimum free energy (MFE) algorithm (Kong et al., 2018), however, is based on calculation of the MFE scheme, where experimentally predetermined values are added for each base pair that is found within dynamic programming matrix. The free energy depends on the sequence part of actual segment and the most adjacent base pairs. The total free energy would be the sum of all increments. There are certain limitations that must be considered for MFE method. Moreover, as the most important to be specified are energies of bulge loops and single non-canonical pairs that are not being predicted.

  RNA folding process lacks the balancing, yet the kinetics of the process may be pointed out. Due to that, the structure obtained by MFE might not be the same as the actual native fold. As another drawback case of MFE method that must be

considered is lack of prediction of pseudoknots in the structure, similarly as in the Nussinov's algorithm.

- The concept of MEA algorithm (Clote et al., 2017) deploys the technique of partition function calculation known as McCaskill's partitioning function (Palkowski and Bielecki, 2019), where prediction of base-pair probabilities is mainly considered. Moreover, these probabilities are integrated into dynamic programming algorithm. This method has been tested in different kinds of RNA, showing a high level of accuracy measure, even higher than the one provided by the MFE method.

  MFE method finds one specific best guess for secondary structure of RNA. Eventually, with this method, high probability base pairs are chosen that leads to higher accuracy level. Other relevant difference is within characteristics used. MEA method utilises stable base pair probability, over the thermodynamic measures established by MFE algorithm.

  More advance models for pseudoknot detection and secondary prediction are Pseudoknot algorithms (Liu et al., 2015). There are two steps that are building up this kind of dynamic programming algorithm. First and foremost, the possibility of finding helices that could form pseudoknot formations is achieved followed by the calculation and folding of the whole structure formation.

  Once obtained the energy dot plot that gives the candidate pseudoknot helix list, H, along with the corresponding helix energies, some characteristics must be considered, in particular: sequences longer than 100 nucleotides or more are taken in account, due to what tested sequences are ranging from 200, 500 until 1,000 nucleotides, secondary, the energy level of $\Delta G°$ must be 25% of the free energy of the MFE. The $\Delta G°$ of Hi would be obtained from the nearest-neighbour AU/GU pairs. Eventually, filtration of helices in the H would consider some predefined steps. Moreover, a helix Hi would be accepted into H if it has more than three base pairs (Mustoe and Buasan, 2018).

- The most advanced algorithms used for the justification of secondary structure prediction of RNA are considering, appraise different kind of neural networks (NNs). Following are the most relevant as well as analogous to the presented paper:

  a *RNA secondary structure prediction using an ensemble of two-dimensional DNN and transfer learning* (Singh, 2019), is a method considering pseudoknots as the most advanced and delicate structures to be predicted. This method is however more sophisticated than the one proposed in this paper, but the idea of using CNN is the same. Due to its advance features being taken into consideration, this can be supported as the most accurate method as well as the starting point for our research. Moreover, its immersive capabilities are going to be discussed within results comparison section. Yet, it must be discussed that the concepts for the usage of ResNet has been inspired by this paper, with difference of the missing steps and concepts that are not considered. Main difference must be established in the usage of RNN's and two different types of databases for data entry and pre-processing as well as for the transfer learning.

b *Predicting RNA secondary structure via adaptive deep recurrent neural networks (RNN) with energy-based filter* (Lu et al., 2019), as the title suggests is a method used for secondary structure prediction of RNA mainly based on the usage of RNN. Considering that this research paper follows fairly different concept of prediction, it is not considered into development of current paper. However the results obtained may be discussed for the purpose of proper differentiation and power of NNs for same task.

c *Prediction of RNA secondary structure with pseudoknots using coupled DNN* (Mao et al., 2020), is method that uses homologous type of sequences for the intention of finding the secondary structure of RNA. Besides, this method uses multiple sequences approach and mainly falls within the approaches of comparative analysis for the purpose of structure discovering. Since within this paper, single RNA approach is considered, this might not be appropriate method to be compared with.

d *RNA secondary structure prediction using deep learning with thermodynamic integrations* (Sato et al., 2020), however, observes one of the oldest approaches and mostly used one, the Zuker MFE method that is interested into thermodynamic parameters when applied as additional features to the overall folding process of the RNA. Fundamentally, this concept is far different by the one proposed in this paper, due to the lack of usage of these parameters in the case. These parameters are highly impacting the level of accuracy, yet, when pseudo-knots are taken into consideration, this method may lacks large amount accuracy, especially if considered sequences are very long.

e *A new method of RNA secondary structure prediction based on convolutional neural network and dynamic programming* (Zhang et al., 2019b), is method which is the most fundamentally similar to the CNN that are going to be discussed furthermore. Crucial difference might be encountered in the application of dynamical programming algorithm present in the beginning of this method that uses matrix representation and the sliding window approach to classify the pairing of the bases based on such values obtained. Essentially, the idea behind this algorithm is proper evaluation and pre-processing of data, since data being used lacks that, which is not the case with the bpRNA database used in this study. As final point established, the way CNN are being implemented in the discussed paper may be seen as very different than the one presented in the current one, since it uses classical type of convolutional neural networks. This gives good support for comparison discussed in the results section.

f *Predicting RNA SHAPE scores with deep learning* (Noah Bliss, 2020), may be defined as the most irrelevant to the one proposed, yet very important for the purpose of secondary structure prediction. This is one of the newest methods in the branch considering NNs. Moreover the structure approaches finding of Pearson correlation coefficient with experimental SHAPE scores, due to these advance concepts being incorporated in the study, it is usually categorised as one of the most sophisticated.

3 Research methodology

CNN as advanced architectures of standard artificial NNs, have the ability of replacing the process of manual feature engineering by representation learning, known as feature learning. Feature learning allows detection of some latent data characteristic, without human intervention. Extraction of general purpose features that work well for unknown classes, obtained by employing shift-invariant filters, which supports concise and reliable analysis results. The importance of feature identification and learning is in promotion of faster learning without explicit direction for that. CNNs are applied in variety of domains such as: recognition of breast cancer from image, semantic segmentation or recognition of semantics of every pixel in a given image, end to end robotic control (Aarshay, 2016).

In bioinformatics field (Zhang et al., 2019b), CNNs are used for: gene expression, regulation and ChIP-seq data, analysis of gene expression levels, etc. CNN has been applied on both microarray and sequencing data of RNA binding proteins, for a purpose of learning sequence binding specificities. CNNs are powerful methods in solving tasks where a spatial information has been provided, as well as for the purpose of natural language processes (NLP) tasks.

In this section general overview of components and building blocks of the NNs, followed by some main characteristics are provided.

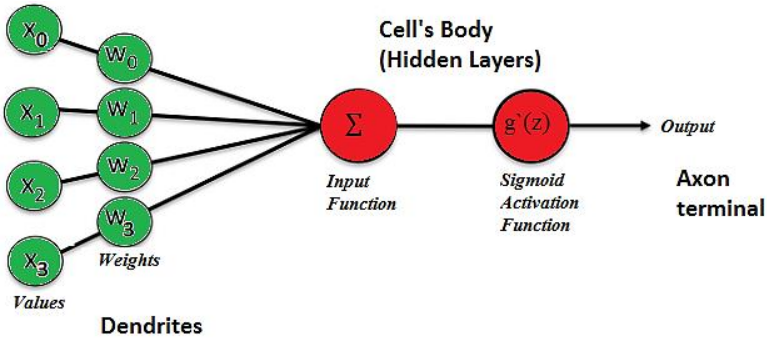
Main processing unit of each layer in artificial NN is defined to be the neuron (Figure 2). If a comparison to the human neurons is performed, drawn conclusion is that, not any difference in the function exists except for the physical difference.

Some of the basic parts that build up the neuron as a structure are (Aarshay, 2016):

- Dendrites – part of the neuron that takes impulse from other neuron, or input data.
- Neuron's body – takes impulses, analyse them and makes decision what step to take next.
- Axon terminal – exit part of neuron that gives the output in form of electrical signal.

Dendrites of each neuron takes electrical impulses as input data in the cell's body, makes some processing or combines the data provided in order to obtain useful information and as final product outputs electrical impulses to other set of neurons or so called terminals.

Figure 2 Comparison between human neuron and artificial neuron (see online version for colours)



Source: Library (2020)

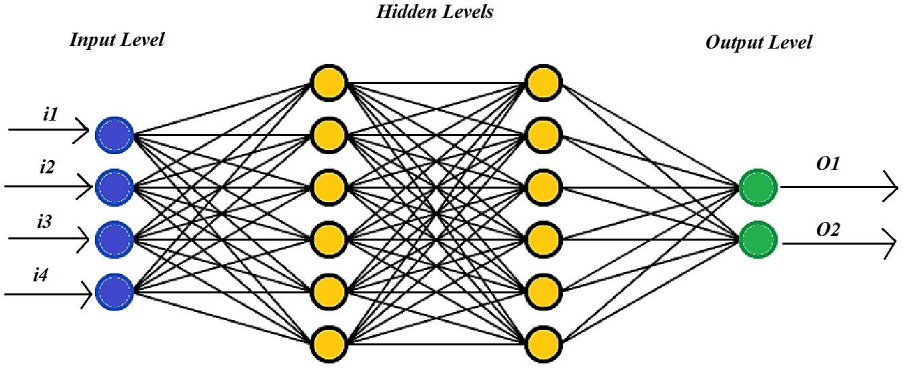
On Figure 2, is given an input layer which works similarly to how dendrites work. The major goal of the neuron is to collect all of the given inputs, analyse them and makes conclusion based on the analysis. The processing is done in the hidden layers. Finally, after the hidden layer follows the output layer that transmits the final output finding all other neurons in the same way as in the human NN.

A basic structure of NN considers the following different layers (Thomas, 2019) (Figure 3):

- Input layer: what is given to the neurons as an input raw data.
- Hidden layers: what would be analysed, organised and formed into complex conclusion based on the input data provided to network.
- Output layer: the final step which will provide results or conclusions from the previous layers.

Usually the secondary structure of RNA prediction considers RNN. Moreover, the main difference between the method of convolution and recurrent one is in the representation of the sequence. RNN works with simple sequence, whereas CNN works with matrices or vectors. Finally, prediction of the structure would consider the dot bracket representation, which is based on the probability of matching the bases or establishing the G-C, A-U and G-U relationships between bases.

Figure 3 General structure and levels of artificial NN (see online version for colours)

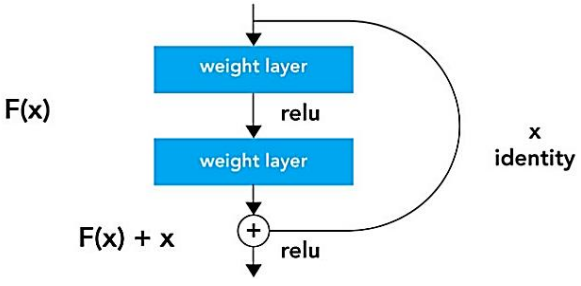


Source: Sorokina (2017)

Discussing about CNNs, these DNN are composed by several different layers of convolutions that consider nonlinear activation functions such as ReLU or tanh affecting the obtained results. Additionally, in feed-forward NN concept, each input neuron relates to output neuron that is followed by in the next layer, also called as fully connected layer, compared to that in the convolutions this concept lacks. Besides, results are formed on local connections, where each region of the input is connected to a neuron in the output. Each layer concern different filters, usually of a high amount and as final obtains the results. In addition, these networks use subsampling layers. Furthermore, during the training phase, a CNN automatically learns the values of its filters based on the task being performed. Finally, the last layer is a classifier which idea is to use high-level features.

Substantially, in order to solve a complex problem, additional layers are added to the DNN that affect the levels of accuracy and performance. Main benefit behind adding more layers relies on the progressive learning on more complex features. Nevertheless, it was detected that there is a maximum threshold for depth with the traditional convolutional neural network model (Mujtaba, 2020). In order to solve the problem of training very deep networks, introduction of ResNet was done. The main features behind ResNet are the so known residual blocks (Figure 4). Predominantly, different is that there exist direct connections that skip some layers which are in between. Due to this skip connection, the output of the layer might not perform the same. Without using skip connections, the input is multiplied by the weights of the layer followed by adding a bias term.

Figure 4 ResNet blocks, residual network (see online version for colours)



Source: Mujtaba (2020)

Main benefit behind the usage of skip connections or ResNet overall, is seen in the vanishing gradient problem solving in DNN. Additionally, these connections aid by allowing the model to learn the identity functions that assures higher layer will have the same performance as lower layer (Zhang et al., 2019a).

3.1 Structure of the algorithm

For the formal definition of the algorithm, the following declaration should be considered:

$$A = \{(x_i, y_i)\} \text{ where } i = 1, 2, 3, .., n$$

- A is the set of pairs of sequences.
- n is the amount of available bases that would form pairs.
- x defined as amount of primary structures given.
- y stands for the amount of secondary structures obtained.

The network is considering the following definitions. Let consider that f is a function, where F is the set of functions building up the network, therefore $\boldsymbol{f} \in \boldsymbol{F}$, dataset of features mapping

As $a = x \rightarrow y$, labels y we get to:

$$f = \arg_{f} L(a(f, x), y) \text{ where } \boldsymbol{f} \in \boldsymbol{F}$$

For the purpose of vector representation, it must be considered usage of two dimensions V and W, where V is for primary and W is for secondary structure annotation.

Basic Python packages that are used for building the algorithm are:

- Torch and Torch.nn, used for a purpose of DNN, creation of convolutional neural networks and working with them.
- Torchtext (Pytorch, 2019), as a package used for the data processing utilities and formation of datasets for natural language.
- Matplotlib (Hunter, 2012), used as library for creating static, animated, and interactive visualisations within Python.
- Numpy (The NumPy community, 2008) used for working with arrays, multidimensional arrays mostly.
- Os (Python, 2001), library for reading from file in Python.
- Pandas (2008), the library used for manipulation and working with CSV and other text files, mostly for reading and writing data, alignment, reshaping, slicing, indexing, grouping merging, etc.
- Forgi (Beckmann et al., 2015), library for RNA analysis and representation of RNA structure, based on Vienna RNA package.

In order to obtain higher efficiency, code has been divided into different files. Every file has a different purpose:

utilities.py Has defined functions for work with the sequence, sequences predictions comparison, making the balance of the sequences and making the visualisation at the end with matplotlib and forgi libraries.

dataset.py Has been used for defining the dataset, uploading the data from files downloaded from the official database for bpRNA, reading and writing them into sequences, loading them as key value pairs to be easily processed and worked with.

model.py Has been used for declaration of model (defined convolutional layers and other layers of the ResNet).

train.py Has been used for optimisation, validation and evaluation of the model.

3.2 Input layer

Input layer consist of a files that include RNA sequences. The files have been retrieved from the official database bpRNA (Danaae et al., 2018). The files would consider dot-bracket representation type; therefore the sequence would be of type symbols/characters. The amounts of sequences used in this work are around 10.000 sequences. Since these files are being pre-processed and checked, additional cleaning and processing of the data in the files is not acquired and not applied, therefore, this must be considered as additional step when considering for different data as source.

When working with deep learning where natural language is present, reading from file acquires steps such as: to tokenise characters, to perform mapping from a character to unique identifier, to make conversation of character into integer form, also to perform

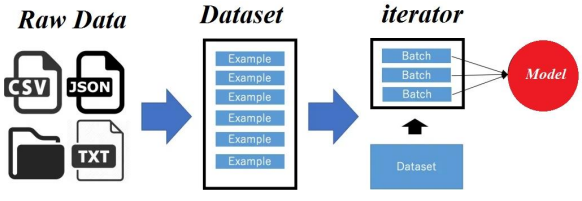
loading data in a format that NN accepts it, and finally must be aware of the equality in sequences length.

In order to deal with all of these characteristics, the usage of Torchtext (Pytorch, 2019) is essential. Torchtext's role is to take any file in a raw format, such as is csv, json, etc. and make it into a dataset.

Dataset could be defined as a block of data, which has been pre-processed. Form of representation is canonical, or easy to be used by other data structures. After formation of the dataset, follows the step where the iterator, must be used for a purpose of batching, numeration, packaging and moving up of the data. This is the step that gives possibility to a data to come up to the deep neural network, hidden layers (Figure 5).

To introduce these variables to the NN vector representation has been used. Following is that, each of the characters is represented with some form of zeros and ones in the vector.

Figure 5 TorchText parts and workflow (see online version for colours)



Moreover, ending up with hyper-parametrised vectors, there is some need of feeding these characters to the NN. That is where the embedded layer, previously discussed, comes into usage. Usually this is a layer of a size given $vocabulary_{size} * dense_{vector_{size}}$, where for each word in the vocabulary (character); there exist a corresponding index in embedded matrix.

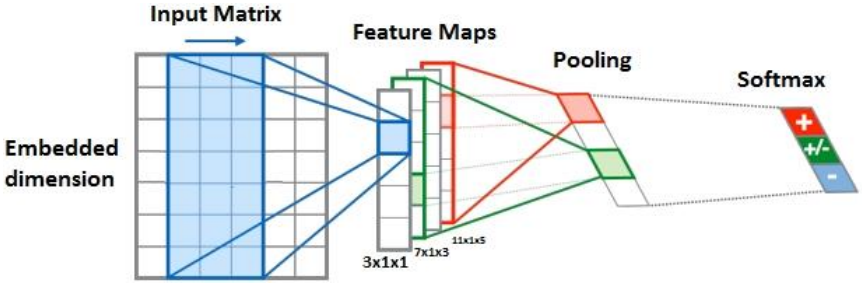
3.3 Building up the model

Working with sequential data, usage of CNN must consider one dimension, so convolution would be defined as convolution of one dimension (Figure 3). This CNN filter working on the principle of sliding down the sequence.

Since working with embedded CNN, the input and output size must be the same size, therefore must not be any change in the amount of channels specified. Difference is present in the kernel size, since it starts with amount of 11 degrades to 7 and the final convolutional layer has 3 as a kernel size.

Finally, the formation of a new matrix known as 'convolved feature' or 'activation map' as well as 'feature map' (Thomas, 2019), is provided. It is important to note that filters act as feature detectors from the original input image. The specification of a stride gives the amount of move from one to another element when applying the filter (kernel). Usually the smaller the stride is, the higher the overlapping will be (Figure 6), for this purpose stride considered in this network is considered to be fixed and 1.

Figure 6 Formation of feature map (see online version for colours)



Source: Library (2020)

3.4 ReLU function and dropout layers

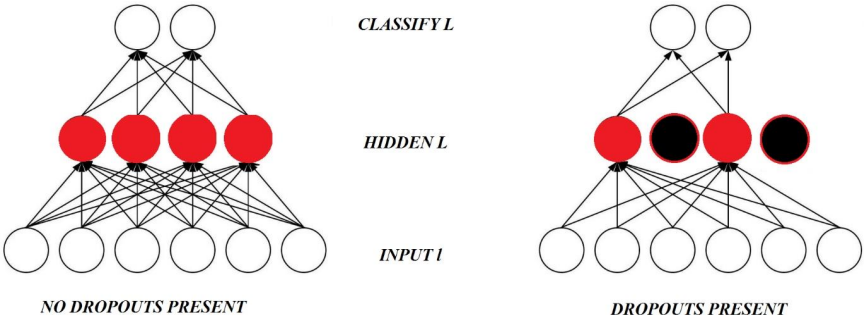
The CNN architecture is consisted of three hidden layers, one which is initialisation convolutional layer and other two layers within the ResNet block, with the ReLU preceding them (Paszke et al., 2019) as activation function as well as dropout regularisation method.

Most used activation function is ReLU because of its possibility to work with nonlinear data ReLU among other activation functions (ELU, SELU, LeakyReLU, Swish…) is very easy to be calculated and implemented in general, since it uses comparison with input and the value of zero (Paszke, et al., 2019). ReLU activation function allows CNN to scale up the size. The most important thing about usage of this function is the avoiding the false zero approach, when high value of input is present.

ReLU function in the proposed algorithm has been implemented in two places, both of which are before the dropout layer, which is not an exact layer by the theory, but is defined as that in practise.

Moreover, the dropout layer as layer mostly applied on top layers consisted of large number of parameters for the purpose of prevention of feature co-adaptation and overfitting. Besides, it may also be considered as regulariser. Therefore, as widening of residual blocks results in an increase of parameters, we tend to implement dropout layers after ReLU activation functions. Previously, dropout as regulariser in ResNet was used as identity part of the block that has shown big amount of negative effects of that. Followed by that, the dropout is therefore added between convolutional layers, acting as selector within neurons for the role deciding which one will continue on the next level.

Figure 7 Dropout layer role within convolutional neural network (see online version for colours)



However, this is practical when prevention of confusions in the model or over-fitting of the training data is present. Simple representation of these layers in our network would be to filter out which neurons are important and drop out which are not, also seen on (Figure 7) (Brownlee, 2018).

As a final statement, addition of the dropout layer into ResNet block was considered to be between convolutions after ReLU for a purpose of disconcert batch normalisation in the following block and avoidance of overfitting, additionally as the best specified amount of it was decided to be value of 0.2% or 20%. Nevertheless, in deep ResNet this approach aids with diminishing feature.

3.5 Optimisation and loss function

PyTorch (Anaconda, 2012) has inbuilt loss and optimisation functions. In this paper is used cross entropy (CE) function, which works with a batch of size N corresponding to C labels that give a variable of dimension N by C. This value of a variable has been obtained when the variable of that dimension passes throughout the model.

Optimisation package in PyTorch, allows easy interfaced optimisation of the algorithms. It works in a way that passing of the parameters in the model that will be updated at each iteration is done. Another available possibility is to specify the learning rate to be per layer or per variable. The learning rate specified in this case is considered to be 1e-3 (Paszke et al., 2019).

For the learning rate the specification of the scheduler must be done. The learning rate of each parameter group is specified to be by gamma in every step_size of the epoch (whole set). This decay may happen simultaneously with other changes presented in the learning rate, which come outside of this scheduler.

3.6 Training and testing of the network

Foremost, the training starts with one convolution layer followed by block of ResNet. The following (Figure 8) gives an example of the training with amount of epochs (traversals over whole set) set to 10 over the whole set and also over the deeper layers by the ResNet blocks specified as batches (sections divided in the embedded layer) that are specified with additional batch_size. This is followed by the conversion of data and target into PyTorch variables.

Once the training has been finished, the result for ten epoches will give the following ratio between the training loss and validation loss showed in Figure 8.

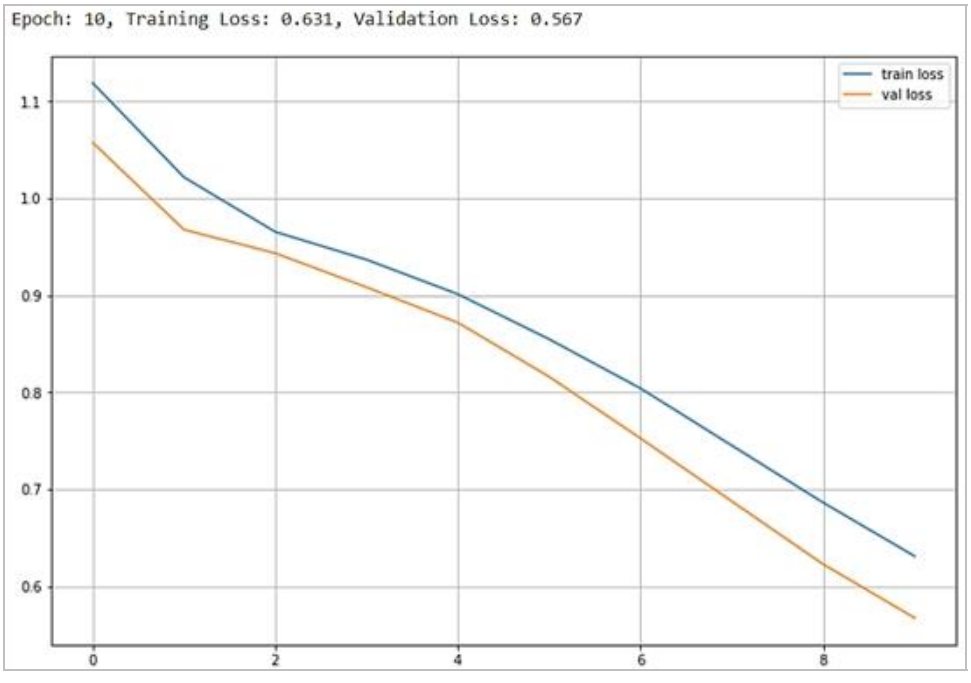
Classification problems base prediction on discrete class as output. Moreover, the interest is put over the separation of dataset into mismatched and possibly distinctive classes that are mainly based on dissimilar parameters, therefore novel and obscure record will proceed into the classes.

Knowing that entropy is the quantifier of randomness when the information has been processed, whereas CE is a quantifier of the difference between the randomness when two arbitrary variables are being examined, statement must be made that:

- Cross-entropy (Paszke et al., 2019) is a loss function used with intention to learning how to use the probability of memorised data. Once faulty prediction occurs it will provide greater penalty, for the predictions, with higher confidence levels. However, if the divergence of the predicted probability escalates, the cross-entropy loss will

also escalate. Moreover, if predicted probability is around 0.020 when genuine examination ranges up to 1, triggered point will be high loss value.

Figure 8 Validation and train loss of the network for epochs of value 10 (see online version for colours)



Finally, if considering an ideal model yet to be faultless that will acquire a value that has logarithmic loss roughly 0.

- CE loss, where calculation of the gradient when output neurons of the CNN are accounted for back-propagation, as well as when optimisation of defined loss function uses tuning of the parameters within network. Therefore, the computation of the gradient of CE Loss is done with the respect of each CNN class score which is defined in s.
- Terms that are defined as negative are the zero one. However, scenario with the loss gradient taken with respect to the negative classes must also be on account of Softmax to the positive class that will depend on the scores that are obtained from negative classes.
- However the gradient expression will be the same for values of the C with lack of consideration of the ground truth class C_p, since score of $C_p(s_p)$ are defined in the nominator of the following formulae:

$$\boldsymbol{CE} = -\log \frac{e_p^s}{\sum_j^c e_j^s}$$

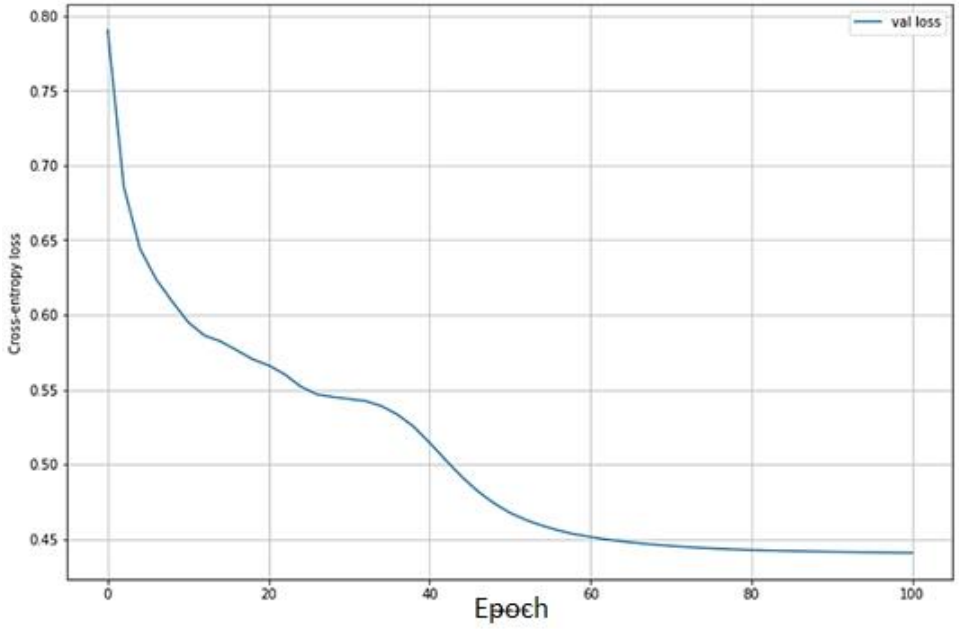
The usage of this function is due to the classification tasks, since it works on the basis of confident model it only predicts accurately, with a very high probability. Once implemented the loss in the network will have different value (Figure 9).

However, the classic Softmax and cross-entropy loss combination is often used as norm for training NNs that must be calculated from the output probability of the ground-truth class. Moreover, measurement of the cross-entropy is done between the ground truth label y and the output of the NN ý. Succeeding is the adjustment of network's parameters to reduce the cross-entropy with the usage of back-propagation.

For the final classification task, a softmax function is being used. Defined as nonlinear function with a special usage within the network, since it is only used at the end, when taken a vector of real numbers. Softmax function (Paszke et al., 2019) is used mainly where trafnsformation from numeric output in the last linear layer must be made, in order to obtain probabilities from the final layer and perform proper classification of stated values (in this case the within two classes of characters).

Therefore, when exponents of each output are taken and being normalised, each number by the sum of those exponents has been summed up and forms a total amount of one. The importance of it might be found when collecting all possible probabilities in the networks, for a purpose of obtaining probability of one.

Figure 9 CE loss function (validated loss) in the network (see online version for colours)



CE loss that has been previously discussed is one loss function that has a purpose of multi-class classification problem.

Finally, the loss for optimising NN models usually uses the cross-entropy approach with known one-hot representation (label of target class) and a categorical distribution calculated by the output of the network through softmax. However, with this common setting of the cost function, the network is trained to make the categorical distribution as close as possible to target elements that are all zero except the one for the true class

which are stated as one. It has been widely accepted that minimising the cross-entropy loss is an objective which reasonably conforms achieving high classification accuracy (Li and Maki, 2018).

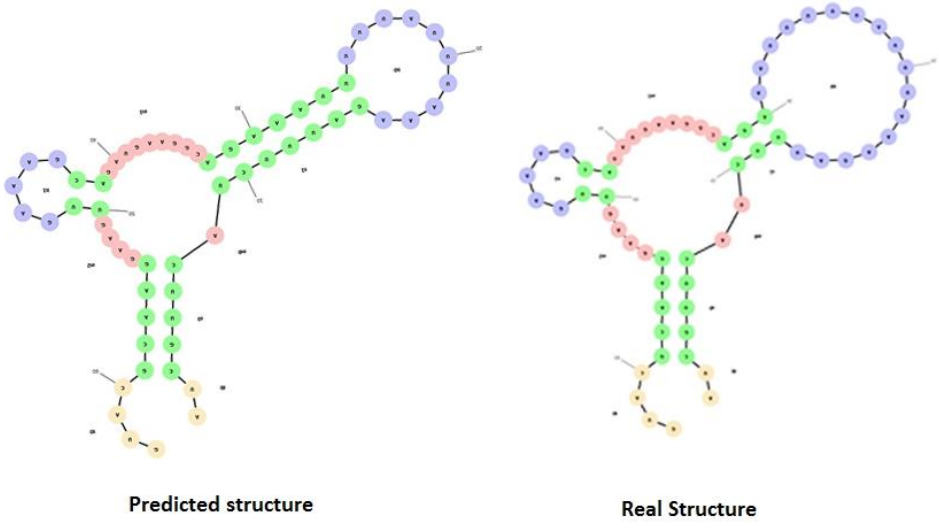
3.7 Output layer of the network

Once taken the result from the classification step, the obtained result would consider three main character representation :{ (,) and }. These characters are mainly used to describe the state of pairing between bases such as: paired, half-paired or unpaired, which leads to the formation of secondary structures in the RNA, also called as loops. Matching parenthesis and dots denotes paired and free bases, respectively. The algorithm could execute them in the raw format of dot-bracket representation, which is a way to get the first output, yet the dot-bracket notation usually happens to be cumbersome, considering that it leads to large expressions that usually consider information that is partially obscured for human readers.

Because of it counting of the identical characters is acquired and formation of pairs too, which must be done manually. For more convenience, if certain features in the sequence are modelled in readable way, such as graphical shaping of the different loops, it solves the major problem. For solving previously stated problem, special visualisation package has been used, ViennaRNA package (Hofacker, 2017) in specific, through the forgi and forgi.nn libraries.

First and foremost, the representation of the actual and predicted model is outputted as dot-bracket representation that additionally has used it for building up the representation of the secondary structure of the RNA within visual format. The final representation is seen on Figure 9.

Figure 10 Usage of forgi library for visual comparison of real structure and predicted structure (see online version for colours)



4 Methods and results

In order to be able to replicate these methods and obtain similar or same results, following steps must be taken into consideration:

- With help of the main Python (2001) library Os, reading of characters in the bpRNA (Danaae et al., 2018) files is performed and characters are pushed into sequences. Mainly for the purpose of working with sequences we need to consider library as NumPy (The NumPy community, 2008).
- Once sequences of dot – brackets are formed they are also given values to be able to perform operation over key-value pairs formed, therefore each character has value or weight to be operated with.
- This is followed by the creation of a model, with the help of Torch library or PyTorch which has torch.nn (Anaconda, 2012) library that is used for the creation and work with NNs, in this case the convolutional neural network or more specific ResNet type of it.
- Consequently, CNN must use three convolutional layers, two dropout layers and two ReLU layers. The algorithm would follow the structure of (Figure 1), where input layers starts with key-value pairs, dot brackets and their weight. This enters the initial convolutional layer that has kernel size of 11(filters), which would continue to the ResNet block considering convolution layer of 7 by 7 and ReLU function as well as first dropout layer (with 0.2 value implied), that ends up into next convolutional layer considering five kernels (filters). Next it proceeds into ReLU function with additional dropout layer which gives input to the final output layer considering cross-entropy optimisation and max-pooling as final mediators into the training process, at the end the calling of the visualisation library forgi (Hofacker, 2017) is performed for visual representation of the results obtained.

The last part of the algorithm to be discussed after the model is created is the training and validation loss (Paszke et al., 2019). For this purpose we need to create functions that will approach the average loss of the sequences one for the training and one for the validating. For the purpose of comparing how big is their difference that is preferred to be as low as possible, backward propagation in the CNN is used, with updating of the weights after each running until reaching the final optimal solution. This is followed by the plotting with Matplotlib library (Hunter, 2012) that shows how good the learning process goes.

4.1 Methods evaluation

After overall evaluation of the model and obtained results, an accuracy level comparison of the algorithms must be made for the purpose of verifying its accuracy and sensitivity. Therefore, the classical benchmark specification known as positive-predictive value (PPV) is used, which is founded on the base-pair prediction accuracy and another one known as sensitivity specificity for overall sensitivity prediction.

Secondary structure prediction of RNA usually uses two benchmarked estimators for purpose of predicting accuracy and this has been taken as general rule within bioinformatics area.

Sensitivity is defined as percentage of known pairs which are correctly predicted, and PPV is the percentage of predicted pairs that has been known in the structure. These two statistical benchmarks are calculated with the following formulae (Zhi, 2009):

$$sensitivity = \frac{number\ of\ correctly\ predicted\ base\ pairs}{total\ number\ of\ known\ base\ pairs}$$

$$PPV = \frac{number\ of\ correctly\ predicted\ base\ pairs}{total\ number\ of\ predicted\ base\ pairs}$$

Moreover, these benchmark specifications are widely used as estimators of the state of the algorithm, starting from the very beginning of algorithms for the purpose of Secondary Structure prediction, as well as needed in order to obtain valid values which are used for the calculation of the F-score, furthermore. The values obtained for sensitivity are ranging around 0.72 and the values for PPV or specificity are also around 0.80.

However, F-score which can be also referred as F1-score is a measure of how accurate a model is with a given dataset. Often it is accepted to be used within binary classification systems. Besides, the F-score uses a combination of the sensitivity and specificity of the model, which also can be stated as the harmonic mean of the model's precision and recall.

The F-score finds its usage within wide variety of machine learning models, in particular and most often within natural language processing. Adjustment of the F-score is possible and depends on the parameters such as sensitivity and specificity and their adjustment overall.

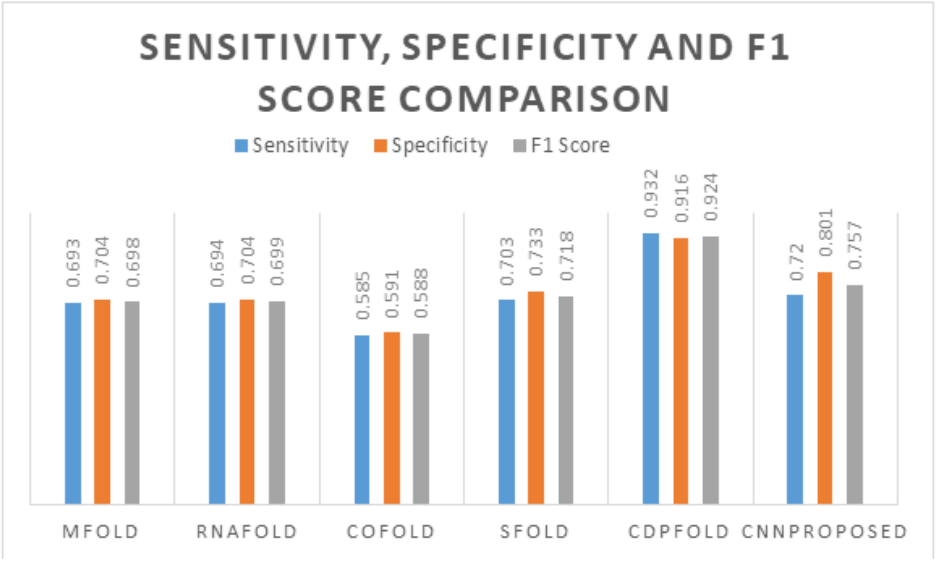
Obtained values of the previous two statistical benchmarks which are easily obtained with Python functions, are after that input in the benchmark F-score, which was already discussed, and is given within the following formulae:

$$F\ score = \frac{2 * sensitivity\ AND\ specificity}{sensitivity\ OR\ specificity}$$

For the purpose of comparison with similar methods that employ NNs, we use the comparison metric F1 previously calculated, given on Figure 11. From the figure we can conclude that CNN approach (labelled as CNNProposed) or last one in the bar chart), has the value of F1 around 0.76 or 76%, which compared to most models is between highest. However, better and more sophisticated can be spotted to be CDPFOLD represented by Zhang et al. (2019b).

The Nussinov Jacobson algorithm as the oldest one would have running time of $O(n^2)$ (Venkatachalam et al., 2014), compared to that the algorithm of Zuker's MFE works with the time complexity of $O(n^4)$ (Zhao and Sahni, 2016), whereas when considering a little bit more advance algorithm such as MEA it works in $O(n^4)$ (Clote et al., 2017). Consecutively the amount of time needed to execute the Pesudoknotted algorithm would be $O(n^5)$ (Mustoe and Buasan, 2018).

Figure 11 Sensitivity, specificity and F1 scores comparison, other popular methods considering NNs and our method, labelled as CNNproposed) (see online version for colours)



For the purpose of complexity evaluation within NN usually we analyse model's complexity by taking the count of total amount learnable parameters, moreover, collecting the size of the parameter file in terms of MB for our model. However, this kind of information is very useful for understanding the minimum amount of GPU memory required for execution of the model. For the purpose of calculating time complexity of our method we have taken statement from this study (Bienstock, 2018), which states that CNN ResNet can calculate time complexity with the following formulae based on the cross-entropy combined with soft-max:

$$Cross\text{-}Entropy + Soft - Max = O\left(\left(m\log(m)\Delta^{O(k2)}/\epsilon\right)^{n+m+N} D\right)$$

From where n and m are the input and output dimensions and N is the total number of parameters. We Δ to express the maximum vertex in-degree in G which is directed graph defining the NN.

In all results the node computations are linear with bias term and normalised coefficients, and activation functions with Lipschitz constant at most 1 and with 0 as a fixed point which also includes ReLU as well, additionally k stands for the depth, and finally, D stands for the size of the dataset.

From here we can conclude that the time complexity of this algorithm stays way much forward older methods, and can compete with other methods considering NN's.

5 Discussion and future work

5.1 Technical implementation

For the purpose of algorithm testing, from dynamical algorithms, until the newly discussed algorithm, platform being used is of performances: CPU Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz. For the algorithms: Nussinov-Jacobson, Zuker's MFE and Four Russian implementation of Nussinov, MEA and pseudoknotted had been used the source code based in C/C++ language and tested in the Visual Studio 2017 Package (Microsoft, 2012) as a running environment with additional support of OpenMP (Board, 2018) for the purpose of parallelisation.

For the second part, where design of the CNN architecture had been considered, the following components were mainly considered:

Python 3.7 (64 bit) (Pytorch, 2016), with the addition of the Anaconda Navigator), Jupyter Notebook 6.0.3 version (Perez and Granger, 2014), environment for interactive testing and computing, ViennaRNA Forgi – library for RNA for visualisation of the secondary structure (Hofacker, 2017), Pytorch 1.6.0 version (Anaconda, 2012) – building the convolutional neural network support whereas the dataset has being extracted from bpRNA – 1 m version 1.0 (Danaae et al., 2018).

However, model has been constructed of three convolutional layers, two dropout layers and two ReLU function layers (Lab, 2016), where one initialization convolution layer was taken out of the ResNet block which concluded all of the previously mentioned steps. Moreover, with the help of the forgi library taken from Vienna RNA package final comparison between the real and the predicted structure of RNA is proved with visual representation of otherwise burdensome representation with dot-bracket.

Some changes in the algorithms could result in better accuracy and better time complexity which may be seen as a new chance for a purpose of obtaining better results, and may be discussed as a future work in this field.

5.2 Research implications

Constructing NN for the purpose of predicting secondary structure of RNA has been discussed throughout different studies (Singh, 2019; Bliss et al., 2020; Lu et al., 2019), the proposed model with convolutional neural network considering modern type of implementation known as ResNet approach is done for the purpose of proper classification of the results into adequate groups as well as for avoidance of vanishing gradient that often occurs within very deep networks consisted of large amount of layers. Additionally, reason behind ResNet as type of CNN being used, is the immersive capability to solve very complex problems that require very large amount of layers in a very efficient way, not affecting time and accuracy.

However, considering the large amount of studies implementing different techniques and models for the same purpose, withdrawing only the most efficient and competitive conclusions must be made. For this purpose, constructing model which will run the best possible way was only considered.

ResNet are novel type of networks when searching throughout different types of CNN, yet they have proved their ability to precisely and efficiently approach large amount of problems, which was also proved within this paper. Moreover, this study

intentionally avoids prediction of the pseudoknots for the purpose of avoiding low accuracy rates that usually occur within such a type of models.

Additionally, for advancement of this study a transfer learning approach may be considered that will furthermore add up some missing parts, which came as complementary intention from recent studies (Lu et al., 2019), showing up great results with this approach too.

Finally, this may be defined as a good starting point for more efficient and accurate network with additional work over it.

5.3 Future work

First and foremost, as advancement of the proposed method, different type of NN can be considered for the same purpose as well as addition to the already existing, CNN model, which is discussed in similar studies with the RNN and combination of RNN (Lu et al., 2019) and CNN models (Wang et al., 2019).

Secondary, a hybrid version of NNs may also find purpose in the implementation for increasing the speed up and accuracy, when considering different NNs in combination. Since this thesis considered only one type of activation and regulation function, whereas there are a large amount of them present, this could be seen as another possible implementation.

Finally, the addition of more features that would shift the sensitivity and accuracy levels to even higher value may be evaluated and implemented.

Moreover, this paper suggests usage of modern type of CNN known as ResNet for the purpose of avoiding diminishing gradient as well as for avoiding very DNN that add to the complexity but they lack the amount of accuracy needed. This can be a crucial point for development of additional models for even better performance, yet it may be qualified as a very competitive approach considering the already established time complexity and accuracy level as one of the highest compared to the already existing method which consider the approach of NNs.

Competitiveness of the suggested method is high, yet any method being suggested may acquire additional changes which may better the results. Moreover, this method lacks the ability to predict some complex structures such as pseudoknots, which is intentionally avoided within the network for the purpose of avoiding high levels of inaccuracy that may occur with such structures. Therefore, ability for their prediction may be considered with addition of transitional learning model or with some complementary NN that will add more value to the already exiting model.

Future work may also consider advancement in multiple levels of RNA structure detection, such as the possibility to add over the secondary structure detection the detection of tertiary structure of the RNA.

6 Conclusions

After the testing and obtaining the F-score (Zhang et al., 2019b), with the benchmark formula applied over true and predicted pairs in the RNA models, above 75% of F1 values has been proved. However, running time also proved that time complexity taken from training and testing of the model with 10,000 sequences, was very good (Danaae et al., 2018), compared to older and already existing competing algorithms.

Working with secondary structure of RNA, as a complex process still lacks the possibility to detect pseudoknot structures. RNAs taken in consideration for development of model as well as for testing were free from pseudoknots, mainly due to the possibility to predict faulty stem areas. This can be seen as a point for advancement since pseudoknots play important role in overall function of RNA, and their existence cannot be ignored for extreme accuracy obtaining.

Moreover, the length of sequences differs which also makes possibility to experiment with different kind of RNA and their length; this is very important when considering the possible different kinds of RNA available in nature. The main representation being used is dot bracket representation, which has not been reflected once some faulty of knots was present.

When working with deep learning methods such as in this case, with consideration of the CNN modern approach ResNet, the amount of data to be used plays a crucial role in the prediction process, for a purpose of avoiding overgeneralised models the data was mostly preserved to be around 1,000 nucleotides long.

Since RNA sequences could be differentiated among different types, to specify, when functional RNA is present, we may obtain a fixed structure, which is not the case with some non-functional RNA sequences. And as a matter of fact, around 2%–3% of RNAs are functional only.

However, RNAs are instable sequences; any factor from aside may have some impact, which usually can lead to changes in the structure prediction too, so addition of large amount of prediction characteristics can enlarge the amount of accuracy in the prediction of secondary structure of RNA.

Nonetheless, introduced algorithm that considers working with CNN approach modern version defined as ResNet for a purpose of secondary structure prediction of RNA, obtained the best possible results when considering algorithms in its class, that does not consider pseudoknots, and has results nearby similar to the one that considers them. This research gives some possible approaches of incorporating artificial intelligence (AI) as a new way of solving tedious and redundantly large work that lacks accuracy and time efficiency of other older approaches.

As a final conclusion to be established, DNN such as CNN (Aarshay, 2016; Thomas, 2019), are an extremely competitive method to be implemented, when accuracy and time are crucial points to be established.

In the area of bioinformatics where development of a new drug is sensitive work, time and accuracy is of high importance. Additionally to consider is that implementation of the CNN for the purpose of secondary structure prediction of the RNA could be done in a very practical way, with usage of less resources and with a possibility to predict a structure which would have above 80% accuracy and sensitivity, according to the benchmark testing (Zhang et al., 2019b).

7 Author contributions

H.A, R.Sh.U, and Gj.S.Lj conceived and directed the project. Ch.B had to obtain data from bpRNA database, research and contribute to the development of CNN. Also the conduction of the data, analysis as well as the interpretation of results was task performed by Ch.B. All of the authors were included in the designing of the study, whereas the reviewing of the data was performed by Gj.S.Lj. For the writing and editing of the

manuscript, all of the authors were employed. Afterwards following with the reviewing process of the manuscript, before final approval to be send for publication.

8 Lessons learned

Working with secondary structure of RNA, as a complex process still lacks the possibility to detect pseudoknot structures. RNAs taken in consideration for development of model as well for testing were free from pseudoknots, mainly due to the possibility to predict faulty stem areas. This can be seen as a point for advancement since pseudoknots play important role in overall function of RNA, and their existence cannot be ignored for extreme accuracy obtaining.

Working with sequences taken from bpRNA database, model was built over 10,318 RNA secondary structures from seven different sources. The length of sequences differs which also makes possibility to experiment with different kind of RNA and their length; this is very important when considering the possible different kinds of RNA available in nature. The main representation being used is dot bracket representation, which has not been reflected once some faulty of knots was present.

When working with deep learning methods such as in this case, with consideration of the CNN approach, the amount of data to be used plays a large role in the prediction process, for a purpose of avoiding overgeneralised models the data was mostly preserved to be around 1000 nucleotides long, notice must be made, since this can lead to false predictions.

As a final notice, RNAs are instable sequences; any factor from aside may have some impact, which usually can lead to changes in the structure prediction too, so addition of large amount of prediction characteristics can enlarge the amount of accuracy in the prediction of secondary structure of RNA.

To conclude, introduced algorithm that considers working with CNN approach for a purpose of Secondary Structure prediction of RNA, obtained the best possible results when considering algorithms in its class, that does not consider pseudoknots, and has results nearby similar to the one that considers them.

Acknowledgements

The work in this paper was partially supported by the University of Information Science and Technology ‘St.Paul the Apostole’, Ohrid, Macedonia and BioInformatics and Genomic Privacy Laboratory (BIGL).

References

Aarshay, J. (2016) *Fundamentals of Deep Learning – Artificial Neural Networks* [online] https://www.analyticsvidhya.com/blog/2016/03/introduction-deep-learning-fundamentals-neural-networks/ (accessed 2 August 2020).

Anaconda, I.O.A. (2012) *Python for Machine Learning Anaconda* [online] https://www.anaconda.com (accessed 20 July 2020).

Beckmann, T., Kerpedjiev and Hofacker (2015) Vienna RNA Package [online] https://viennarna.github.io/forgi (accessed 1 August 2020).

Bienstock, D.G.M. (2018) *Principled Deep Neural Network Training*, arXiv, p.26.

Bliss, N., Bindweald, E. and Shapio, B.A. (2020) 'Predicting RNA SHAPE scores with deep learning', *RNA Biology*, Vol. 17, No. 9, pp.1324–1330.

Board, O.A.R. (2018) *OpenMP* [online] https://www.openmp.org (accessed 3 January 2020]).

Brownie, J. (2020) *Softmax Activation function in Python* [online] (accessed 2020).

Brownlee, J. (2018) *Dropout Regulization of Neural Networks* [online] https://machinelearning mastery.com/dropout-for-regularizing-deep-neural-networks/ (accessed 4 August 2020).

Chauleva, B., Hristov, A. and Sandjakoska, L. (2020) 'Single RNA secondary structure prediction based dynamical programming algorithms: to parallelize or not?', *CIIT 2020*, Skopje.

Clote, P., Lou, F. and Lorenz, W.A. (2017) 'Maximum expected accuracy structural neighbours of an RNA secondary structure', *BMC Bioinformatics*, Vol. S6, No. 13, p.13.

Danaae, P., Rouches, M. and Deng, D. (2018) *bpRNA – Single Molecule RNA* [online] http://bprna.cgrb.oregonstate.edu/ (accessed 1 August 2020).

He, K., Zhang, X., Ren, S. and Sun, J. (2015) *Deep Residual Learning for Image Recognition*, CVPR, p.12.

Hofacker, I. (2017) *VienaRNA Package – TBI Uni.* [online] https://www.tbi.univie.ac.at/RNA (accessed 23 June 2020).

Hunter, J.D. (2012) *Matplotlib* [online] https://matplotlib.org/ (accessed 5 August 2020).

Kong, Q., Liu, Z. and Xiaobing, T. (2018) *The Computation of the Barrier Tree for BHG of RNA Folding Structure*, Hangzhou, IEEE.

Lab, F.A.R. (2016) *Neural Networks Library PyTorch* [online] http://www.pytorch.org (accessed 6 March 2020).

Li, V. and Maki, A. (2018) 'Feature contraction: new ConvNet', *The British Machine Vision Conference (BMVC)*, Newcastle, p.11.

Library, M.O.L. (2020) *MIT Open Learning*, 2021, Massachusetts Institute of Technology, Massachusetts.

Liu, Z., Zhu, D. and Dai, Q. (2015) *Predicting Algorithm of RNA Folding Structure with Pseudoknots*, RNA, Shenzhen.

Lu, W. et al. (2019) 'Predicting RNA secondary structure via adaptive deep recurrent neural networks with energy-based filter', *BMC*, Vol. 20, No. 25, p.20.

Mao, K., Wang, J. and Xiao, Y. (2020) 'Prediction of RNA secondary structure with pseudoknots using coupled deep neural networks', *Biophys. Rep.*, Vol. II, No. 6, pp.146–154.

Microsoft (2012) *Visual Studio 2017 C/C++ Dev Environment* [online] https://visualstudio.microsoft.com/ (accessed 21 February 2020).

Mujtaba, H. (2020) mygreatlearning [online] https://www.mygreatlearning.com/blog/resnet/ (accessed 24 June 2020).

Mustoe, A.M. and Buasan, S. (2018) *Pervasive Regulatory Functions of mRNA Structure Revealed by High-Resolution SHAPE Probing*, s.l., Cell.

Noah Bliss, E.B.B.A. (2020) 'Graph neural representational learning of RNA secondary structures for predicting RNA-protein interactions', *RNA Biology*, Vol. 17, No. 9, pp.1324–1330.

Palkowski, M. and Bielecki, W. (2016) 'Parallel tiled Nussinov RNA folding loop nest generated using both dependance graph transitive closure and loop skewing', *BMC Bioinformatics*, Vol. 18, No. 3, p.290.

Palkowski, M. and Bielecki, W. (2019) 'Parallel cache efficient code for computing the McCaskill partition function', *FedCSIS*, Vol. 210, p.207.

Pandas (2008) Pandas [online] https://pandas.pydata.org/about/ (accessed 05 August 2020_.

Paszke, A., Gross, S. and Chintala, S. (2019) *Pytorch* [online] https://pytorch.org/docs/stable /generated/torch.nn.ReLU.html?highlight=relu#torch.nn.ReLU (accessed 5 August 2020).

Perez, F. and Granger, B. (2014) *Jupyter Environment* [online] https://www.jupyter.org (accessed 25 August 2020).

Python Software Foundation (2001) *Python Standard Library* [online] https://docs.python.org/3/library/os.html [Accessed 1 August 2020].

Pytorch (2016) *Neural Networks Library PyTorch* [online] http://www.pytorch.org (accessed 03 2020).

Pytorch (2019) *Pytorch – Torchtext* [online] https://pytorch.org/tutorials/beginner/text_sentiment_ngrams_tutorial.html#sphx-glr-beginner-text-sentiment-ngrams-tutorial-py (accessed 5 August 2020).

Sato, K., Akiyama, M. and Sakakibara, Y. (2020) 'RNA secondary structure prediction using deep learning with thermodynamic integrations', *Nature Communications*, Vol. 10, No. 4, p.8.

Singh, J., Hanson, J., Kuldip, P. and Zhou, Y. (2019) 'RNA secondary structure prediction using an ensemble of two-dimensional deep neural networks and transfer learning', *Nature Communications*, Vol. 5407, No. 9, p.10.

Sorokina, K. (2017) *Medium* [online] https://medium.com/@ksusorokina/image-classification-with-convolutional-neural-networks-496815db12a8 (accessed 23 June 2020).

The NumPy community (2008) NumPY [online] https://numpy.org/devdocs/user/quickstart.html (accessed 05 August 2020).

Thomas, A. (2019) *Coding the Deep Learning Revolution*, 1st ed., ebook: ebook.

Venkatachalam, B., Gusfield, D. and Frid, Y. (2014) 'Faster algorithm for RNA folding using the four Russian method', *Mol. Biol*., Vol. 5, No. 2, p.9.

Wang, L. et al. (2019) 'DMfold: a novel method to predict RNA secondary structure with pseudoknots based on deep learning and improved base pair maximization principle', *Frontiers Genetics*, Vol. 4, No. 3, p.10, pp.1664–8021, p.143.

Zhang, A., Lipton, Z.C., Li, M. and Smola, A.J. (2019a) *Dive Into Deep Learning* [online] https://d2l.ai/index.html (accessed 8 August 2020).

Zhang, H., Zhang, C. and Li, Z. (2019b) 'A new method of RNA secondary structure prediction based on convolutional neural network dynamic programming', *Frontiers in Genetics*, Vol. 10, No. 3, p.467.

Zhao, C. and Sahni, S. (2016) 'Cache and energy efficient algorithms for Nussinov RNA folding', *BMC Bioinformatics*, Vol. 18, No. 15, p.6.

Zhi, J.L.J.W.G. (2009) 'Improved RNA secondary structure prediction by maximizing expected pair accuracy', *RNA*, Vol. 1805, No. 13, p.10.