Comenius University in Bratislava

Faculty of Mathematics, Physics and Informatics

COMPARING BIOLOGICAL SEQUENCES WITH NEURAL NETWORKS

Master's thesis

2021 Bc. Filip Kerák

Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

COMPARING BIOLOGICAL SEQUENCES WITH NEURAL NETWORKS

Master's thesis

Study Programme:	Applied Informatics
Field of Study:	2511 Computer Science
Department:	Department of Applied Informatics
Supervisor:	doc. Mgr. Tomáš Vinař, PhD.

Bratislava, 2021 Bc. Filip Kerák

49642564

Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

THESIS ASSIGNMENT

Name and Surname:	Bc. Filip Kerák
Study programme:	Applied Computer Science (Single degree study, master II. deg., full time form)
Field of Study:	Computer Science
Type of Thesis:	Diploma Thesis
Language of Thesis:	English
Secondary language:	Slovak

Title: Comparing Biological Sequences with Neural Networks

Annotation: Comparison of biological sequences is often performed by a simple dynamic programming algorithm called Needleman-Wunsch algorithm. Due to its nature, the algorithm only admits simple scoring schemes which do not fully capture the properties of biological sequences. The goal of this thesis is to design an analogous algorithm, where cells of the dynamic programming matrix will be replaced with a neural network, thus admiting much more complex non-linear scoring schemes. More general scoring scheme that can be automatically trained will allow us to apply sequence alignment to other sequence representations, including electrical signals from nanopore sequencing or to various representations of biological sequences that include uncertainty.

Supervisor:	doc. Mgr. Tomáš Vinař, PhD.
Department:	FMFI.KAI - Department of Applied Informatics
Head of department:	prof. Ing. Igor Farkaš, Dr.
Assigned:	26.11.2020
Approved:	26.11.2020

prof. RNDr. Roman Ďurikovič, PhD.
Guarantor of Study Programme

....

Student Supervisor

Poďakovanie

...

Abstrakt

...

Kľúčové slová: ...

Abstract

...

Keywords: ...

Contents

Chapter 1

Preliminaries

In our work we will use various terms and methods, therefore in this chapter we introduce all the necessary preliminaries and current state of the problematic. Most of the bioinformatics terms and definitions we use are from book Understanding Bioinformatics [6].

1.1 Bioinformatics preliminaries

Define sequences as words over an alphabet Σ. Based on the chosen alphabet the sequence represents different structure. Probably the most important alphabet is $\Sigma = \{A, C, T, G\}$, sequences built from those letters describe DNA-Deoxyribonucleic acid . If we use U instead of T then $\Sigma = \{A, C, U, G\}$ and sequences represent RNA-Ribonucleic acid. In some problems we work with protein sequences, to represent those we need a larger alphabet consisting of 20 elements.

1.1.1 *DNA* - Deoxyribonucleic acid

DNA stores all the genetic information, that is required to create and maintain an organism. DNA is build from building blocks called nucleotides also called bases. There are only 4 different nucleotides: A - adenine, C - cytosine, T - thymine, G - guanine. Those building blocks link together and create a strand, our word over Σ. DNA have a three dimensional structure of double helix, where two strands are linked. Nucleotides from one strand form a pair with specific nucleotides from other strand. A creates pair only with T and C with G, therefore a sequence of a strand is complementary to the base sequence of its partner strand. All the genetic information is encoded by order of those base pairs.

For example human genome contains about 3 billions of those base pairs. Since those sequences determine biological features of all organisms we are quite interested in understanding which parts encode which feature, how changes in those sequences influence organism, how it can be used, etc.

1.1.2 *RNA* - Ribonucleic acid

For the purposes of this thesis we can look at RNA quite similarly. In RNA T-thymine is replaced by U- uracil so $\Sigma = \{A, C, U, G\}$. RNA molecules are also quite shorter than DNA.

1.1.3 Sequencing

Sequencing is a process for determining order of bases of DNA or RNA molecule. Currently technology is not capable to sequence whole genome. Current generation sequencing tools have reads of length up to hundred kilo base pairs. However those reads can have up to 10% error rate. Whole genome then needs to be reconstructed from those short reads.

1.1.4 Alignment

Usually we want to compare sequences with other sequences, that are already known and stored in databases or samples we have collected before. To do this we have to align those sequences. First possible task is global alignment, in this case we want to align sequences from beginning to end. This approach is used in case we want to know to which organism belongs our sample or we want to see how the sample is changed compared to original sequence. There are three types of changes that can occur in sequences. First is insertion in this case a new base is inserted into original sequence. Contrary deletion is when base from original sequence is missing in sample. Last is mutation, it means that a base from original sequence has changed to a different base.

Little bit different task is local alignment, by usage of local alignment we want to find one or more similar contiguous regions within sequences. Often similar sequences have similar function, so this can help us predict function of the gene from sample.

In order to tell which alignments are good and which are bad we need a scoring scheme. Simplest scheme would be +1 for match, -1 for mismatch and -1 for space. In reality more complex scoring schemes are used to better capture biological aspects of the problem.

1.1.5 Global alignment - Needleman-Wunsch algorithm

Problem of global alignment can be solved by using dynamic programming and Needleman-Wunsch algorithm.

We have two sequences X and Y of lengths m and n. We create a matrix A of size m*n. Cell A[i,j] represents a sub-problem of global alignment of sequences $x_1, x_2...x_i$ and $y_1, y_2...y_j$. First row and first column represents that one of the sequences has length zero and is aligned with spaces. So it look like this $A[i, 0] = -i$ and $A[0, j] = -j$.

Now general case for $i > 0$ and $j > 0$ will be:

$$A[i,j] = max \begin{cases} A[i-1, j-1] + s(x_i, y_i) \\ A[i-1, j] - 1 \\ A[i, j-1] - 1 \end{cases}$$

Where $s(x_i, y_i)$ is score of match or mismatch and is defined as:

$$s(x_i, y_i) = \begin{cases} 1, \ if \ x_i = y_i \\ -1, \ if \ x_i \neq y_i \end{cases}$$

Diagonal case $A[i-1, j-1] + s(x_i, y_i)$ means that sequences are aligned with match or mismatch, up case $A[i-1, j] - 1$ means that x_i is aligned with space and left case $A[i, j-j] - 1$ means that y_j is aligned with space.

When whole table is filled the score of global alignment is in right bottom corner. If we want the whole alignment we take the last cell and go back up to first cell A[0,0]. Often exist more than one path with same score.

This was the simplest version, some improvements as affine gap scoring or scoring matrix are often introduced to better capture biological nature of the problem. In affine gap scoring we have greater penalty for opening a gap and then lower penalty for extending an existing gap.Some base mutations are more probable than others, to accommodate this we can use substitution matrix and give some mutations lower penalty.

	A	C	G	T
A	2	-2	-1	-2
C	-2	1	-2	-1
G	-1	-2	1	-2
T	-2	-1	-2	2

Table 1.1: Example of substitution matrix

		G	A	A	G	G	C	C	T	A	C
	0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-10
A	-1	-1	0	-1	-2	-3	-4	-5	-6	-7	-8
A	-2	-2	0	1	0	-1	-2	-3	-4	-5	-6
G	-3	-1	-1	0	2	1	0	-1	-2	-3	-4
G	-4	-2	-2	-1	1	3	2	1	0	-1	-2
C	-5	-3	-3	-2	0	2	4	3	2	1	0
C	-6	-4	-4	-3	-1	1	3	5	4	3	2
A	-7	-5	-3	-3	-2	0	2	4	4	5	4
T	-8	-6	-4	-4	-3	-1	1	3	5	4	4
A	-9	-7	-5	-3	-4	-2	0	2	4	6	5
A	-10	-8	-6	-4	-4	-3	-1	1	3	5	5

Table 1.2: Example of dynamic programming table for global alignment.

1.1.6 Local alignment - Smith-Waterman algorithm

In local alignment we want to find contiguous similar regions between two sequences not necessary from beginning to end. For this we use Smith-Waterman algorithm, it is quite similar to Needleman-Wunsch algorithm. Most important change is that we accept empty alignment with score 0, therefore the lowest value that can be in a cell is 0.

Algorithm then looks like this. Again we start with matrix A of size m*n, baut first row and column are filled with zeros $A[i, 0] = 0$ and $A[0, j] = 0$. Cell A[i,j] represents a sub-problem of highest score of local alignment between sequences $x_1, x_2 ... x_i$ and $y_1, y_2 ... y_j$.

General case have a little change we choose maximum from four options. Three are the same as before and fourth option is zero.

$$A[i,j] = max \begin{cases} 0 \\ A[i-1, j-1] + s(x_i, y_i) \\ A[i-1, j] - 1 \\ A[i, j-1] - 1 \end{cases}$$

Where $s(x_i, y_i)$ is score of match or mismatch and is defined as:

$$s(x_i, y_i) = \begin{cases} 1, \ if \ x_i = y_i \\ -1, \ if \ x_i \neq y_i \end{cases}$$

Score of best local alignment is then maximum from A. There may be more then one equally good alignments. If we want to construct the alignment we start in the cell with highest value and we go backward until we reach a cell with 0 in it. Again this is the simplest version and same improvements

		G	A	A	G	G	C	C	T	A	C
	0	0	0	0	0	0	0	0	0	0	0
A	0	0	1	1	0	0	0	0	0	1	0
A	0	0	1	2	1	0	0	0	0	1	0
G	0	1	0	1	3	2	1	0	0	0	0
G	0	1	0	0	2	4	3	2	1	0	0
C	0	0	0	0	1	3	5	4	3	2	1
C	0	0	0	0	0	2	4	6	5	4	3
A	0	0	1	1	0	1	3	5	5	6	5
T	0	0	0	0	0	0	2	4	6	5	5
A	0	0	1	1	0	0	1	3	5	7	6
A	0	0	1	2	1	0	0	2	4	6	6

Table 1.3: Example of dynamic programming table for local alignment.

can be used to better express biological nature of the problem.

1.2 Neural networks

1.2.1 Overview

Neural networks are computational models that have been inspired by human brain. In biology neuron is a cell, that receive electro-chemical signals from other neurons. If the received signal is strong enough to surpass a certain threshold neuron spikes and sends the signal further. After a spike neuron recovers and for short amount of time does not spike again.

Simplest neural network model is a perceptron, it is a model that imitates biological neuron. It has inputs $x_1, x_2, ...x_n$. For each input x_i it contains a weight w_i. Inputs are multiplied by their weights and summed together, the summation is then used as input for an activation function, if an output from the activation function exceeds set threshold output of the perceptron is 1 otherwise it is 0. This model is used for binary classification and can only separate classes that are linearly separable.

Activation functions are used to bring non-linearity to neural network models. There are activation functions that are quite common, because of their properties. First property is differentiability, since gradient is used to train neural networks it is important to have activation functions that are easily differentiable. One of the first activation function was sigmoid.

$$S(x) = \frac{1}{1 - e^x}$$

Nowadays many different are more common:

- Hyperbolic tangent
- Rectified linear - $relu(x) = \begin{cases} x, \ if\ x > 0 \\ 0, \ otherwise \end{cases}$
- Leaky ReLU - $LeakyReLU(x) = \begin{cases} x, \ if\ x > 0 \\ 0.01x, \ otherwise \end{cases}$
- Softmax

1.2.2 Fully Connected Networks

Those networks consist of one input layer, one or more hidden layers and one output layer. Each layer contains one or more artificial neurons like the perceptron, however each layer can have different activation function. Each neuron has it own set of weights for inputs. Neuron takes all the inputs, multiply it with weights and apply activation function. After all neurons from one layer compute their outputs, those outputs serve as inputs for next layer. Outputs from the last layer are considered to be outputs from whole model.

Those type of networks have the most general usage and produce state of the art results in mathematical and statistical problems, however in task as image processing or natural language recognition they have been outperformed by different types of architectures.

1.2.3 Convolutional Networks

This type of architecture became quite popular since 2012 when neural network called AlexNet from author Alex Krizhevsky, achieved state of the art results in image recognition competition on ImageNet dataset. One of the important steps in his approach was using GPU to speed up training process. Since then computational capacity of computers have quite increased and it is quite simple to build own models that can be trained on large image datasets in short time.

Convolutional networks perform great on 2-dimensional inputs as images and 3-dimensional inputs such as medical scans. Typical convolutional network architecture consist of input layer, few convolutional layers followed by pooling layers and it ends with one or more fully connected layers.

Most interesting part are convolutional and pooling layers. Convolutional layer contains kernels often called filters. Those are windows of fixed size,for example size 3*3 is quite common, those kernels are moved through the input and at each position it is applied. Applications usually means matrix multiplication followed by activation function. By applying those filters we receive output with little bit smaller width and height and possibly more channels, depending on number of applied filters. After convolutional layer pooling layer is often used. Pooling layers reduce dimensions of the input. Pooling layer takes window usually of size 2*2 and moves through the input and reduces each window to one value. Max-pooling is most frequent, however average pooling can be also used. When input is reduced by convolutional and pooling layers it is flattened and standard fully connected layers are used to produce output.

1.2.4 Recurrent Networks

Reccurent models are used in cases when data are time or context dependent, for example time series prediction, speech recognition, music composition or language translation.

Reccurent networks have similar architecture to standard fully connected feed forward networks, however they use output from all previous inputs to compute actual output. The most simple recurrent network use simple vector to represent state. They perform the computation in same way as fully connected network, however they concat this state vector with input vector and from this new vector output is computed. After output is computed state vector is updated from output values and it is prepared to be used in next step.

Most used recurrent network is Long Term Short Memory network - LSTM. Problem of simple recurrent network is that it can not retain the information for many steps, because it is continuously overrode. LSTM overcomes this by introducing memory cell. Memory cell contains input gate and forget gate. Those gates decides when value of the cell is updated, therefore the cell can retain information for many steps without overriding it. Those cells also overcome vanishing gradient problem and help to train network faster.

1.2.5 Usage of Neural Networks for comparing biological sequences

Shortcoming of dynamic programming is that it can not fully represent non-linearity, that come from biological nature of the problem. Modern neural networks perform very well in fields like text processing and language translation. Those tasks are somewhat similar to sequence alignment in a manner that both problems are heavily context dependent and both are special kind of mapping off original sequence to an output sequence. There have already been some approaches to use neural networks in alignment tasks, now we will mention some of them.

NEPAL In paper "Optimizing scoring function of dynamic programming of pairwise profile alignment using derivative free neural network" [5] authors proposed a way to improve alignment of profile sequences, for our purposes those are words over alphabet of size 20.

They used classic dynamic programming, however instead of using standard scoring function they used small fully connected neural network to compute similarity score that was used as score for match or mismatch in each cell of dynamic programming table. Input of the network consisted of 10 elements from one profile and 10 from the other. Architecture of the network can be seen in figure 1.1.

Most interesting part of this paper is the approach that authors used to train the network. They used derivative free approach particularly they used evolutionary strategies to train the network.

In paper "Deep learning-based tool for alignment and single nucleotide variant identification" [2] three different types of networks were used to per-

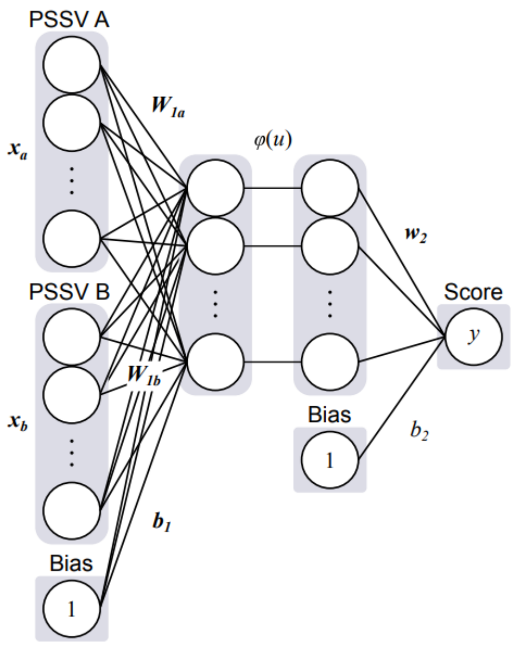


Figure 1.1: Neural network from [5]

form alignment. First model was using convolution. First both reads and reference were broken into smaller parts called k-mers. Then the best match for each k-mer was found with convolutional neural network. Another two models are vanilla recurrent network and LSTM network. Those models take query string and reference string and classify them into two classes matched or unmatched. Then from all matched cases based on best matching score, a reference string is picked for alignment. Architecture of RNN/LSTM is in the figure 1.2. From figure 1.3 it can be seen, that vanilla RNN has higher accuracy up to 40 base-pairs but LSTM was able to perform with relatively good accuracy up to 80 base-pairs, at about 90 base-pairs both models stopped learning. Those models required many different parameters to be set for fine tuning. Authors used genetic approach to find out the best combination of those parameters. They created an initial population with different parameters and let them evolve, in 10 generations they found appropriate set of parameters. Evolution of accuracy of those population can be seen in figure 1.4. Authors concluded that one of limitations of those models is that maximum length of a read is 40 base pairs.

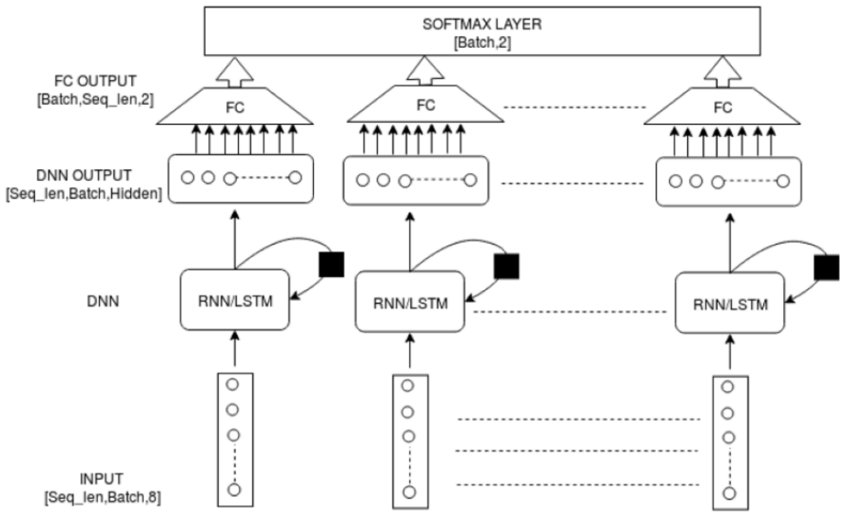


Figure 1.2: Deep recurrent network from [2]

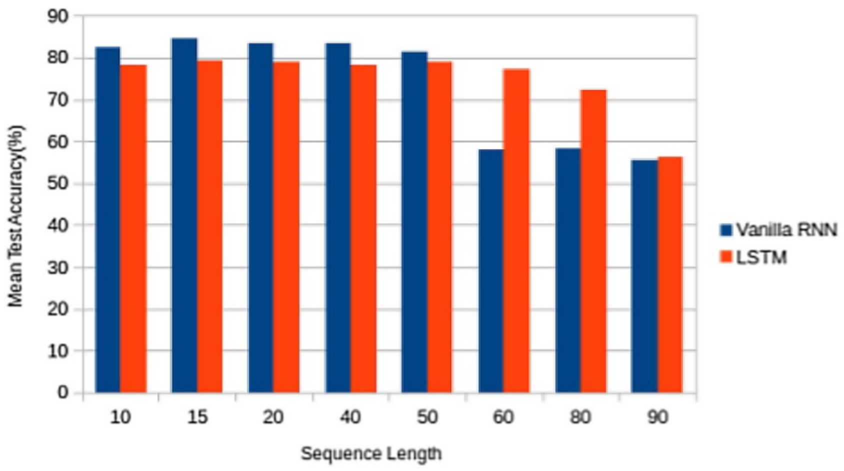


Figure 1.3: Accuracy compare of RNN and LSTM from [2]

Last work that we would like to mention is from paper "Attention Is All You Need" [4]. Authors introduced new model called transformer. This

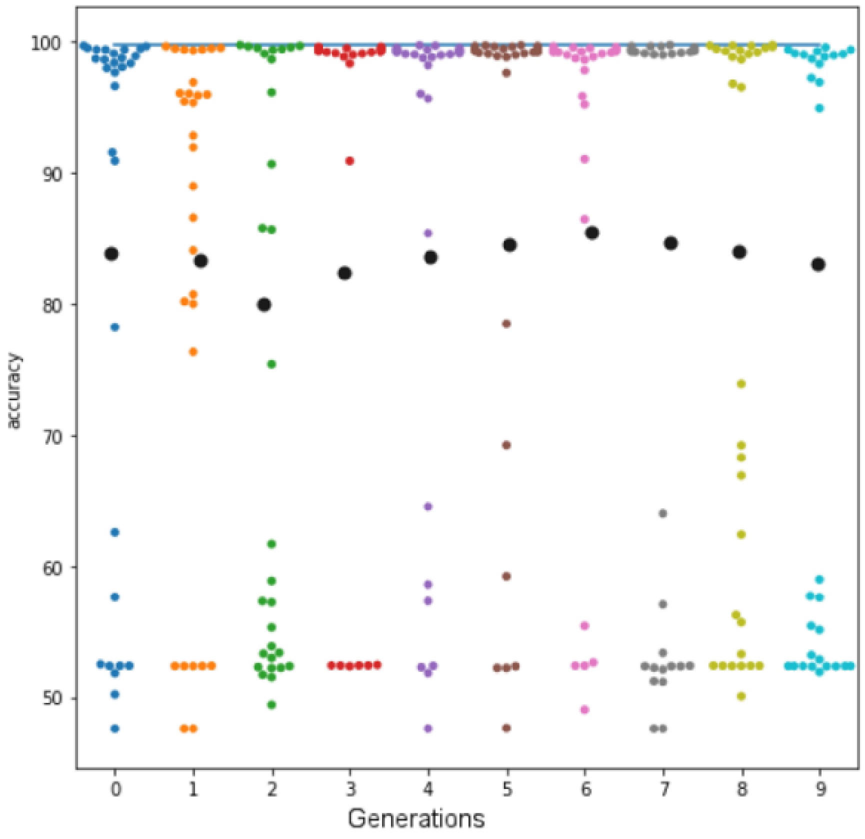


Figure 1.4: Accuracy of Population after each generation from [2]

model uses only attention layers to perform sequence modelling tasks. Authors demonstrated it on the language translation however it looks like promising approach for more sequence related tasks that are context dependent. Attention layer helps model to choose only parts of the input that are important for prediction. In standard RNN or convolutional networks the problem is that the important part of the input can have variable length. It leads to that some parts can be missing when they are needed and some unnecessary parts of the input are used in computation. This is where attention comes very useful, since it choose the important inputs. Whole architecture of the transformer model is in figure 1.5.

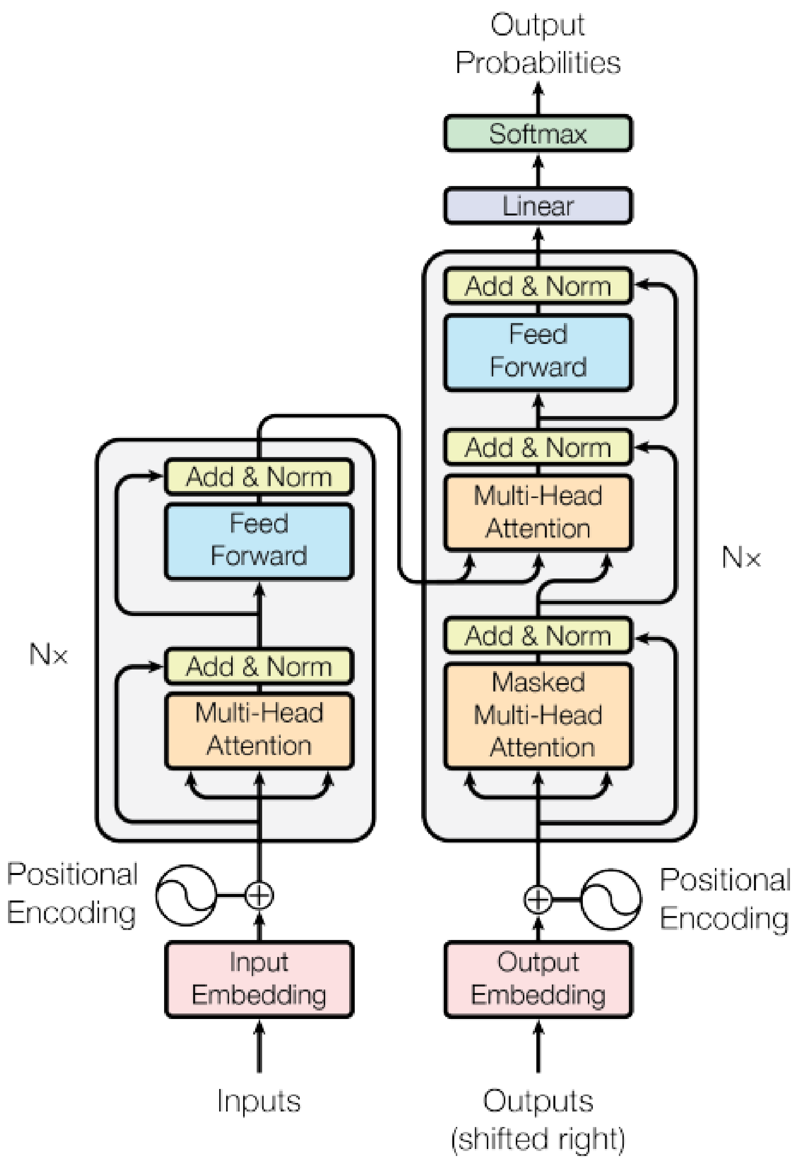


Figure 1.5: The transformer model from [4]

Chapter 2

Possible improvements of alignment task with NN

2.1 Global allignment scoring

2.2 Global sequence allignment

2.3 Maximal local allignment score

2.4 Raw data allignment

2.5 Raw signal allignment

Chapter 3

Data and tools

3.1 Sequences

3.2 Reads

3.3 Evaluation of models

Chapter 4

Design

4.1 Used technologies

4.1.1 Frameworks

4.2 Architecture

4.2.1 Convolutional model architecture

4.2.2 LSTM model architecture

4.2.3 Combined model architecture

Chapter 5

Implementation

5.1 Convolutional model

5.1.1 Training

5.1.2 Evaluation

5.2 Genetic algorithm

5.2.1 Training

5.2.2 Evaluation

5.3 LSTM model

5.3.1 Training

5.3.2 Evaluation

5.4 Combined model

5.4.1 Training

5.4.2 Evaluation

Chapter 6

Research results

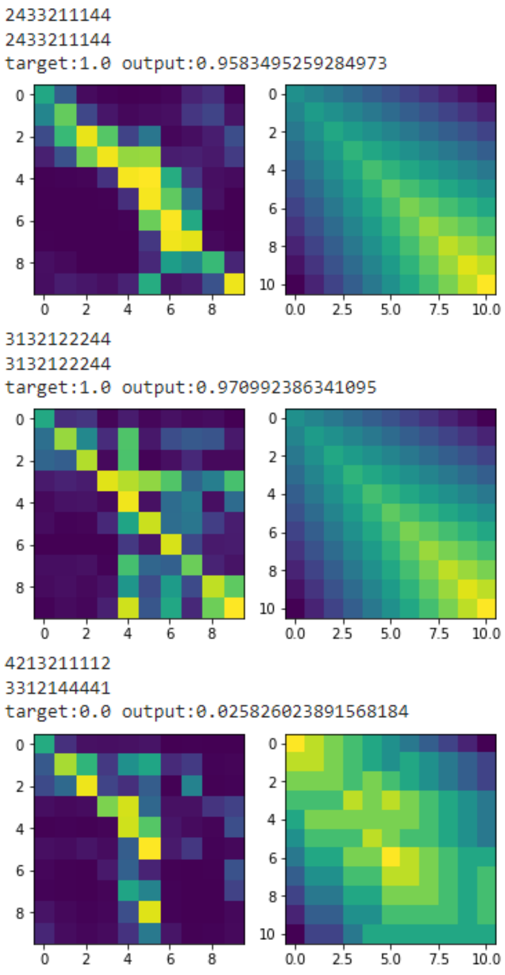
In this chapter we present results from implemented models and discuss about their possibilities.

6.1 Results from models

Fully convolutional model First model we implemented was conolutional network, that used 1-dimensional convolution. We take two sequences, embed those sequences as numbers and then join them. This is the input for this model. Desired output of this model was global alignment score of those two sequences. However this model have not performed very well and accuracy of deciding if sequences are close according to the output score was about 54%. Just slightly better than random guess.

Recurrent modem LSTM network-

Recurent model with convolution. This model achieved best results. It is a reccurent network that uses 2-dimensional matrix to represent state. This matrix is recurrently filled in same manner as dynamic programming matrix. At each step we take a look at neighbourhood of the cell that is going to be filled. Neighbourhood for convolution is window from matrix up and left of fixed size and in sequences we also take into consideration window of fixed size. Output of this network is whole table which in bottom right corner contains value between 0 and 1 that represents overall quality of potential alignment.



```
2433211144
2433211144
target:1.0 output:0.9583495259284973
```

```
3132122244
3132122244
target:1.0 output:0.970992386341095
```

```
4213211112
3312144441
target:0.0 output:0.025826023891568184
```

Figure 6.1: Output table from combined model compared to needleman-wunsch.

6.2 Comparison of models

6.3 Comparison on different data

Chapter 7

Discussion

Bibliography

[1] Pelin Dogan, Boyang Li, Leonid Sigal, and Markus Gross. A neural multi-sequence alignment technique (neumatch). In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[2] G Gupta and S Saini. DAVI: Deep learning-based tool for alignment and single nucleotide variant identification. *Machine Learning: Science and Technology*, 1(2):025013, jun 2020.

[3] Satoshi Koide, Keisuke Kawano, and Takuro Kutsuna. Neural edit operations for biological sequences. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[4] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, page 5998–6008. Curran Associates, Inc., 2017.

[5] Kazunori D. Yamada. Derivative-free neural network for optimizing the scoring functions associated with dynamic programming of pairwise-profile alignment. *Algorithms Mol. Biol.*, 13(1):5:1–5:8, 2018.

[6] Marketa Zvelebil and Jeremy O. Baum. *Understanding Bioinformatics.* Garland Science, 2008.

List of Figures