FACULTY OF INFORMATION TECHNOLOGY CTU IN PRAGUE

Assignment of bachelor's thesis

Title:	Predicting Aptamer Binding Strength in In Vitro Sequence Selection Using Deep Neural Networks
Student:	Linda Beková
Supervisor:	Ing. Daniel Vašata, Ph.D.
Study program:	Informatics
Branch / specialization:	Knowledge Engineering
Department:	Department of Applied Mathematics
Validity:	until the end of summer semester 2024/2025

Instructions

Selection protocols such as SELEX, where molecules are selected over multiple rounds for their ability to bind to a target of interest, are popular methods for obtaining binders for diagnostic and therapeutic purposes.
The aim of the thesis is to analyze the possibilities of supervised deep neural networks trained on sequence ensembles from single rounds of SELEX experiments for thrombin aptamers.
The model trained from sequence data at a given round should be used to predict the effects of a selection (i.e., the selection strength) at later rounds. Moreover, the model's performance should be compared with different learning approaches, including random forests and recently used unsupervised Restricted Boltzmann Machines (RBM).

Detailed assignment points:
1) Get familiar with the SELEX experiment and with the provided experimental data from its several rounds.
2) Preprocess the provided data so they can be used for training of neural networks in a supervised manner.
3) Research and select two suitable architectures of deep neural networks that could be used to predict the binding strength between rounds of the experiment.
4) Train and evaluate the selected architectures on the provided data. Compare the results with other possible learning approaches and discuss their consistency with recent RBM results.

Electronically approved by Ing. Magda Friedjungová, Ph.D. on 14 February 2024 in Prague.

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Bachelor's thesis

Predicting Aptamer Binding Strength in In Vitro Sequence Selection Using Deep Neural Networks

Linda Beková

Department of Applied Mathematics
Supervisor: Ing. Vašata Daniel Ph.D.

May 15, 2024

Acknowledgements

First and foremost, I would like to express my gratitude to my supervisor, Ing. Daniel Vašata, Ph.D., for his guidance, support, and great insights during the past year of working on this thesis. I would like to extend this gratitude to Mgr. Jana Zdarsová and Mgr. Barbora Vavřichová from ELSA CTU for their support and advice during our consultations. Further, I thank my partner Antonín for his support throughout the past 3 years. I am grateful to my friends Eliška, Patrik and David for being there for me during my ups and downs in life and studies. Finally, I would like to express my gratitude to all those who supported me throughout the years of studying at FIT CTU.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as a school work under the provisions of Article 60 (1) of the Act.

In Prague on May 15, 2024 . . . . . .

Czech Technical University in Prague
Faculty of Information Technology

Citation of this thesis

Beková, Linda. *Predicting Aptamer Binding Strength in In Vitro Sequence Selection Using Deep Neural Networks.* Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2024.

Abstrakt

Práce se zabývá problémem zpracování SELEX experimentů pomocí hlubokého učení. Součástí práce je užití dopředné neuronové sítě, konvoluční neuronové sítě, obousměrné dlouhé krátkodobé paměti a metody náhodného lesu pomocí programovacího jazyku Python a porovnání jejich schopnosti predikovat výsledky SELEX experimentů. Práce rozšiřuje předchozí výzkum schopnosti vázání aptamérů na protein trombin pomocí Restricted Boltzmann Machines a nabízí více přístupů ke zpracování tohoto problému. Odhady vybraných modelů dosáhly vysoké přesnosti na souboru dat prezentovaném v předchozím výzkumu. Při testování na dodatečně vytvořených datech měly modely potíže s predikcí schopnosti aptamerů se vázat, a proto byly považovány za nedostatečné pro využití v medicíně. Výsledky jednotlivých modelů a přístupů jsou porovnány. Ze všech algoritmů ukázaly nejlepší úspěšnost alogitmy Restricted Boltzmann Machines a následně Random Forests.

Klíčová slova SELEX, odhad pevnosti vázání aptamerů, neuronové sítě, analýza dat, hluboké učení, Restricted Boltzmann Machine, náhodné lesy, Python

Abstract

This thesis addresses the problem of processing SELEX experiments using deep learning. The work includes employing a Feed-Forward Neural Network, a Convolutional Neural Network, a Bidirectional Long Short-Term Memory, and a Random Forest using the Python programming language and comparing their ability to predict the results of SELEX experiments. The thesis expands on previous research on aptamers' binding ability using Restricted Boltzmann Machines and offers multiple approaches to handling this problem. The selected models' predictions achieved a high accuracy on a dataset presented in previous research. When tested on additionally generated data, the models had difficulty differentiating between binders and non-binders and, therefore, were concluded as insufficient for use in the medical field. The results of individual models and approaches are compared. Of all the algorithms, the best performance showed the Restricted Boltzmann Machines followed by Random Forests.

Keywords SELEX, binding strength prediction of aptamers, deep neural networks, data analysis, deep learning, Restricted Boltzmann Machine, Random forests, Python

Contents

List of Figures

List of Tables

Introduction

As Zhou and Rossi in [1] note, the "magic bullet" method was developed for cancer therapy more than a hundred years ago, in which the ideal therapeutic agent targets a specific tumor cell and kills it. Since then, the main goal has been to find specific molecular defects causing a patient's health issues to implement "targeted therapy". Even though modern medicine can treat cancer to a certain extent, the disease remains, to this day, one of many challenges requiring the development of new therapies. Unfortunately, most therapeutic practices typically lack precise targeting of disease sites. Therefore, many side effects occur as even healthy tissue is exposed to the treatment. With targeted therapy, the drugs could be localized to disease sites, achieving more effective treatment and the patient's faster recovery.

The problem of target therapy, according to Gioacchino et al. [2], requires a method for selecting the target binders called the Systematic Evolution of Ligands by Exponential Enrichment (SELEX). This method uses short oligonucleotides, known as aptamers, to test their binding abilities to the given target molecule.

As the knowledge of aptamers binding ability is not only useful when it comes to cancer, but also as [3] states, the prediction of binding affinity is also a key issue in drug discovery. Drug development – from development through approval to the drug's launch on the market – takes, on average, about 12 years. As the length of the process increases, so does the price once the drug enters the market. Hence, predictions of certain accuracy could be crucial in the development stage, as they could reduce the number of wet-lab[1] experiments, thus leading to more time and cost-efficient drug production.

A research paper [2] from 2022 called "Generative and interpretable machine learning for aptamer design and analysis of in vitro sequence selection" attempted to solve this issue by using machine learning algorithms on a dataset from [5, 6]. The Restricted Boltzmann Machines (RBM) were used on data

[1]The term wet-lab represents the analysis and testing of physical samples of chemicals, liquids, or biological samples such as drugs and fluids [4].

collected from SELEX experiments (the same dataset is used in this thesis) to find the variables that make an aptamer a binder to the selected protein target. In addition, two models were presented. One took into account sequence occurrence to identify the best binders. The second only considered unique molecule sequences.

The authors of [2] also state that certain supervised learning models were employed but did not achieve sufficient accuracy when run on the test set. It also claims that the low results were produced as the dataset contained only positive examples of aptamers that survived each cycle of SELEX. It thus concludes that this problem is too challenging for commonly supervised learning approaches.

We intended to test this hypothesis and attempt to solve this issue by applying multiple supervised learning approaches and comparing their abilities to deal with the problem.

Objectives

The aims of this thesis are as follows:

- get familiar with the SELEX experiment and with the provided experimental data from its several rounds;
- preprocess the provided data so they can be used for training of neural networks in a supervised manner;
- research and select two suitable architectures of deep neural networks that could be used to predict the binding strength between rounds of the experiment;
- train and evaluate the selected architectures on the provided data;
- compare the results with other possible learning approaches and discuss their consistency with recent RBM results.

Our main focus will be binary classification of aptamers binding ability using Feed-Forward, Convolution, and Bidirectional Long Short-Term Memory neural networks. In addition, we will employ Random Forests to offer an alternative to Artificial Neural Network models. The results of all models will be compared to RBM results created in paper [2].

Overview

This thesis consists of two parts, one dedicated to the theoretical background and the second to the practical part of our research. The first chapter will detail the SELEX experiment and describe the steps taken to collect our dataset. As well as that, in this chapter, we will provide an overview of the algorithms selected for this research.

The second chapter will be dedicated to the practical section of our research. Here, we will look in detail at the dataset used. As well as that, we will describe the necessary changes we have made to the original dataset. We will introduce several possible approaches to handling this problem, test them using the selected algorithms, and describe which approach was the most effective. The results of all algorithms, including RBM, will be discussed and compared to determine which is the best at predicting aptamers' binding ability.

Chapter 1

Theoretical Background

In this section, we will look at the theoretical background of the SELEX procedure and explain the retrieval of the dataset used in this thesis. We will introduce the algorithm called Restricted Boltzmann Machines and present our selected algorithms for handling the task. The models we have chosen for this problem include the Feed-Forward Neural Network, Convolution Neural Network, and Bidirectional Long Short-Term Memory Neural Network. To give an alternative to deep learning algorithms, we will also introduce the Random Forest.

1.1 Systematic Evolution of Ligands by Exponential Enrichment

Systematic Evolution of Ligands by Exponential Enrichment is a procedure introduced by Tuerk and Gold, see [7] for more detain, in which multiple oligonucleotides[2], known as aptamers. These aptamers compete with one another to bind with a target protein (cells, tissues, and viruses) over numerous rounds. The following text is a summary selected from [2].

The SELEX process, in this case, used to obtain aptamers that bind to thrombin[3], as can be seen in Fig. 1.1 consists of the following steps:

1. Initial library: The initial library consists of a pool containing DNA aptamers;

2. Incubation with target: The pool is combined with the target, and aptamers compete to bind to thrombin;

[2]Oligonucleotides are short polymers of building blocks made of DNA or RNA and consist of a nucleotide or base [8]. Polymer is any substance of natural or synthetic class comprising large molecules [9].

[3]Thrombin is a molecule that stimulates the growth of tumor cells [10].

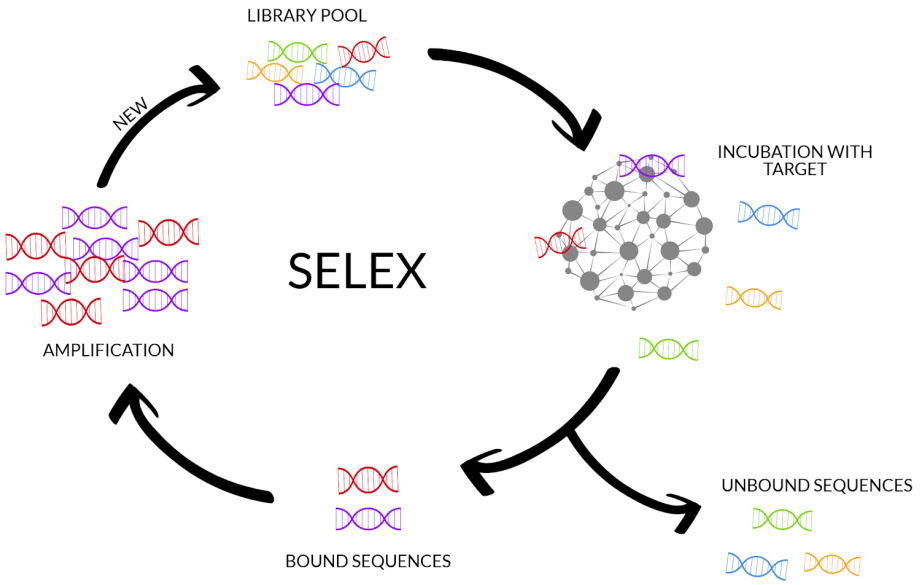

Figure 1.1: Systematic Evolution of Ligands by Exponential Enrichment process diagram [11].

3. Unbound sequences: All the sequences that have not bonded or have bonded too weakly are washed away, so only a pool of strong binders remains;

4. Bound sequences: Created bonds between aptamers and the target are dissociated using heat, and left aptamers are sequenced;

5. Amplification: The polymerase chain reaction (PCR) creates multiple copies of the remaining sequences, serving as a new library for the next SELEX cycle.

This way, only the strongest binders are obtained after multiple rounds of binding and washing away weak binders. Typical SELEX protocols state that the number of counts in the final round of a sequence is closely related to their fitness, and the ones with the highest counts are considered the best binders.

The advantage of using aptamers is that they can be generated through fairly cheap chemical synthesis and be easily modified.

1.2 Restricted Boltzmann Machines

As noted by Goodfellow et al. in [12], RBMs are probabilistic neural network models serving as building blocks for other deep-learning models. These

graphs contain a layer of visible variables and a layer of latent variables. These two layers are connected through their units. These connections are undirected, and there may not be any intra-layer connections between the units, as shown in Fig. 1.2. A general Boltzmann Machine can have connections between two units of the same layer. Thus, the Boltzmann Machine model, which does not allow arbitrary connections between units, is called "restricted". The following overview summarises [12] and [13].

The RBMs have been widely used in fields such as dimensionality reduction, classification, pattern recognition, etc. One of the advantages of using these models is that they can be stacked together to create more complex models.

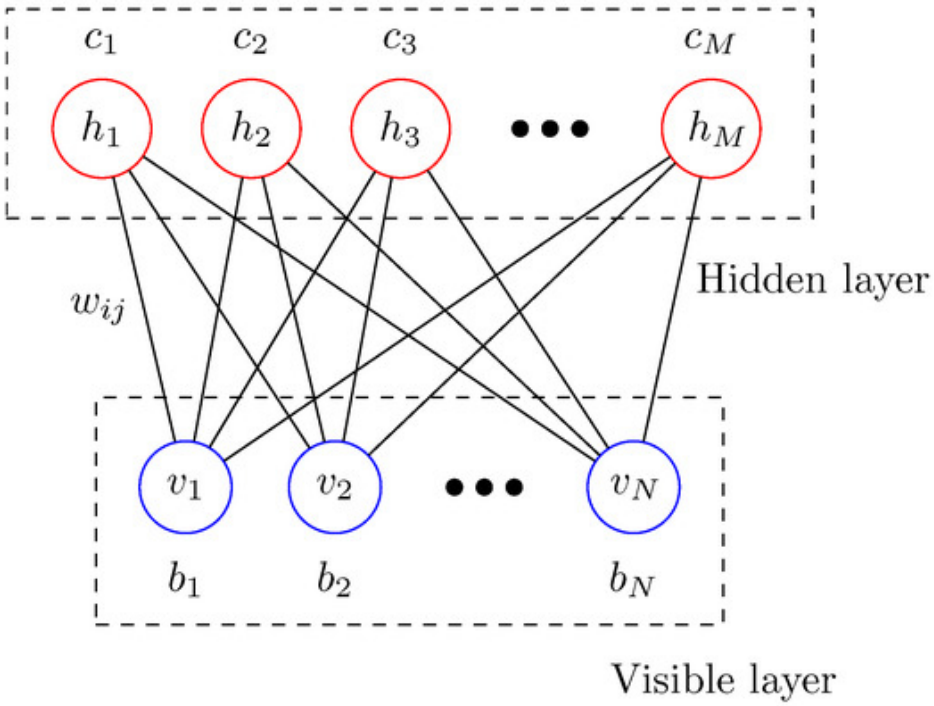

Figure 1.2: Restricted Boltzmann Machines model diagram [13].

Let $\boldsymbol{v} = (v_1, ..., v_n)$ be a vector of n random binary values of the visible observed layer and $\boldsymbol{h} = (h_1, ..., h_m)$ be the vector of m random binary values of the hidden layer. As the Restricted Boltzmann Machine algorithm is an energy-based model [4], its joint probability distribution is shown below:

[4]Many undirected models depend on the assumption that $\forall x, p(x) > 0$. To invoke this attribute, energy-based models can be used where $p(x) = \exp(-E(x))$. As the result of an exponential function is always positive, the developers have the freedom to choose the energy function. [12]

$$P(\boldsymbol{v} = \boldsymbol{v}, \boldsymbol{h} = \boldsymbol{h}) = \frac{1}{Z} \exp(-E(\boldsymbol{v}, \boldsymbol{h})) \tag{1.1}$$

As we can see from Eq. (1.1), the probability distribution is calculated through an energy function E and the normalizing constant of Z, also known as the partition function, described as follows:

$$Z = \sum_{v} \sum_{h} \exp\{-E(v, h)\}. \tag{1.2}$$

The connections between the visible and the hidden layer units are described by the weight matrix $\mathbf{W} \in \mathbb{R}^{n,m}$ where connection strength between $v_i, i \in \{1, .., n\}$ and $h_j, j \in \{1, .., m\}$ units is represented by a point w_{ij} in the weight matrix. As well as that, bias vectors $\boldsymbol{b}$ and $\boldsymbol{c}$ are applied to visible and hidden layer units. The energy is calculated through the following equation:

$$E(\boldsymbol{v}, \boldsymbol{h}) = -\boldsymbol{b}^{\intercal}\boldsymbol{v} - \boldsymbol{c}^{\intercal}\boldsymbol{h} - \boldsymbol{v}^{\intercal}\mathbf{W}\boldsymbol{h}. \tag{1.3}$$

1.2.1 Training

Gioacchino et al. [2] utilized the RBM to learn the probability distributions over aptamers collected from multiple SELEX cycles. Former research regarding the analysis of aptamers assigned a probability of $p(\boldsymbol{s}, \boldsymbol{h})$ to a system state, where $\boldsymbol{s}$ represented a vector of visible units given by single nucleotides along an aptamer sequence and a vector $\boldsymbol{h}$ represented the hidden units' configuration. From these visible configurations, hidden units were made to extract hidden factors of variation.

The training process of RBM involved finding parameters that would ensure maximization of the log-likelihood of the observed data. The likelihood of a single sequence $\boldsymbol{s}$ given by marginalization of all possible hidden unit configurations as indicated below:

$$p(\boldsymbol{s}) = \int p(\boldsymbol{s}, \boldsymbol{h}) d\boldsymbol{h}. \tag{1.4}$$

Likelihood of all sequences from a single cycle is then added to give the final log-likelihood as shown below:

$$L = \sum_{\boldsymbol{s} \in \text{round } r} \log p(\boldsymbol{s}). \tag{1.5}$$

1.3 Our Approach

We chose to utilize multiple machine learning (ML) models and compare their results to determine the best model for this problem. The models we tried were Feed-Forward Neural Networks (FFNN), Convolution Neural Networks

(CNN), and Random Forests (RF). As we further wanted to experiment with more complex neural networks, we additionally implemented Bidirectional Long Short-Term Memory Neural Networks (LSTM).

All chosen Artificial Neural Network models (ANN) were utilized via PyTorch[5] and trained over 100 epochs with early stopping after an increase in the average validation loss from the overall reached minimum for 10 consecutive epochs to prevent overfitting. After stopping, the ANN models reverted to the weights of the stage with the lowest average validation loss. All ANN models used the Cross-Entropy loss function for learning.

Even though the training loss continuously decreases, at some point, the validation loss begins to slowly rise. That is where overfitting occurs [12]. As we assume that the lower the validation loss, the lower the test loss, we need to minimise the validation loss. We achieve this by reverting the model's parameters back to the state of the lowest reached validation loss. If the model continued to learn even after the validation loss increases over multiple epochs, and we would not revert the model back to its optimum state, we'd reach a model that would not perform on the test data as well as it could.

1.3.1 Feed-Forward Neural Network

In this section, we briefly introduce the Feed-Forward Neural Network from [14] as the basic knowledge of ANNs is assumed. The FFNN comprises multiple layers, where an output of one layer serves as an input for the following layer, and all output units interact with all input units. There are weighted connections between the neurons of different layers. The neural network does not allow intra-layer connections. All layers, apart from the input and the output layers, are called hidden layers, as found in Fig. 1.3. An increase in the number of layers can improve the model's flexibility but also cause overfitting and slow down the learning process [12].

1.3.2 Convolution Neural Network

The CNNs use convolution to find higher-order features in the data. They are most valuable when the input values are related spatially, with an N-dimensional grid-like topology, and contain a specific set of repeating patterns. This gives them an advantage over FFNNs, as FFNNs can only take in one-dimensional data and are not well scalable. With CNNs, we have the option of arranging the neurons in a multi-dimensional structure to work with all features of the data. [12, 16]

CNN is a special neural network that uses convolution in at least one of its layers, as seen in Fig. 1.4.

The authors of [12] note that if a CNN has a kernel size smaller than its input data, the CNN is said to have sparse interactions. This differs from the

[5] `https://pytorch.org/docs/stable/optim.html`

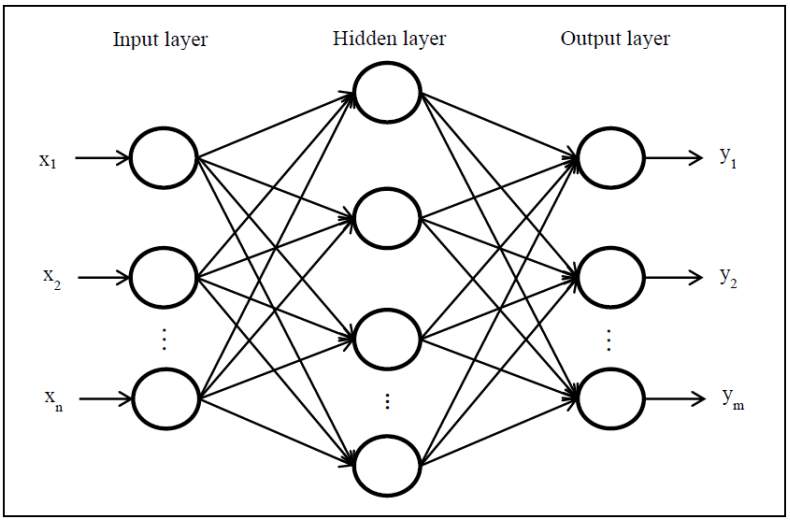

Figure 1.3: Feed-Forward Neural Network structure [15].

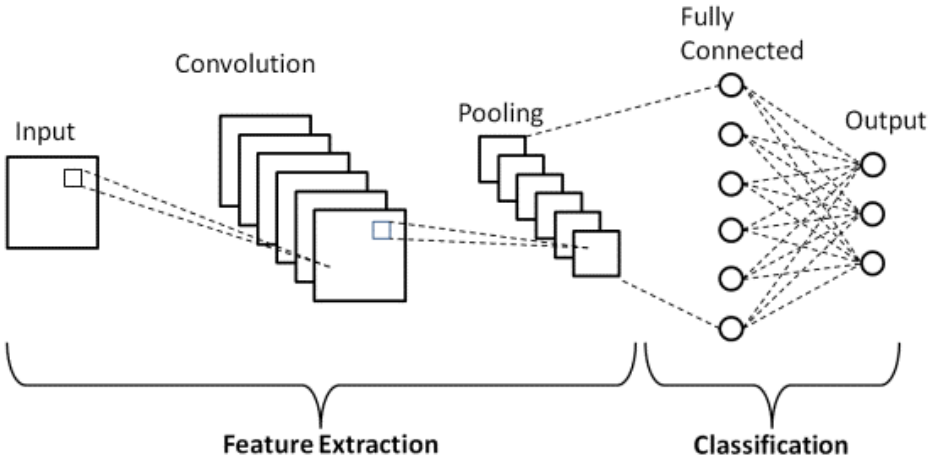

Figure 1.4: Convolution Neural Network structure [17].

FFNNs, as in a FFNN model, all input units interact with all output units. In a CNN model, the output unit is connected only to the number of input units of the kernel size. Therefore, the CNN can detect even small features in the data. As a result, the model improves its statistical efficiency.

A CNN runs in the following stages. Firstly, a layer completes a set of convolutions run in parallel. This action results in a set of linear activations. Secondly, these linear activations are run through a nonlinear activation function. Lastly, a pooling function is applied to the output and to modify it to

its summary statistic. [12]

1.3.3 Bidirectional Long Short-Term Memory

Hochreiter and Schmidhuber created the Long Short-Term Memory in [18]. Its creation was motivated by the RNN's backpropagation error, which can blow up or vanish exponentially. An LSTM layer consists of memory blocks that are recurrently connected. Each of these memory blocks contains one or more recurrently connected memory cells, as well as multiplicative units called input, output, and forget gates.

The LSTM neural networks, as described by Goodfellow et al. in [12], are a type of gated RNNs, which, compared to general RNNs, can alter the connection weights that can change at each time step instead of using manually chosen constants. As well as that, gated RNNs can accumulate information over a long duration. Instead of manually choosing when the neural network should forget the old state of said information, the gated RNNs learn and manage this themselves. The diagram of the LSTM structure can be seen in Fig. 1.5.

The following summary is based on [19] explanation of the LSTM. Each connecting line carries an entire vector from the output of a node to the input of the next. The boxes coloured in yellow represent learned neural network layers, and pink circles show mathematical operations such as addition or multiplication. Moreover, the image shows lines merging and forking. When two lines are merged, the vectors they carry get concatenated, and when forked, copies of the carried vector are made and passed to all directions of the line. The cell state is the key component of the LSTM. In this diagram, it is represented by the horizontal line passing through the top of the cell. Each LSTM cell can alter the cell state by adding or removing information from it. Gates control these processes.

For the LSTM to calculate which information to alter, input, or forget, the model uses $\mathbf{W}_f, \mathbf{W}_i, \mathbf{W}_C$ weight matrices. Furthermore, it uses the bias vectors of $\boldsymbol{b}_f, \boldsymbol{b}_i, \boldsymbol{b}_C$ to add to the calculation.[18]

The LSTM executes the following steps. Firstly, the LSTM has to decide how much of the previously learned information it wants to keep. This is done through the forget gate – a sigmoid layer – which makes the decision depending on the previous hidden state $\boldsymbol{h}_{t-1}$ vector and the new input of the cell $\boldsymbol{x}_t$. The operations made to create an output of the forget gate can be seen in the following equation:

$$\boldsymbol{f}_t = \sigma(\mathbf{W}_f \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_f). \tag{1.6}$$

Secondly, the cell has to decide what information it wants to add to the cell state, represented as $\boldsymbol{C}_{t-1}$ vector from the previous cell and the newly created cell state vector of $\boldsymbol{C}_t$. As shown in Fig. 1.5, adding new information

to the cell state is calculated through two operations. The initial calculation describes which values of the previous cell state should be added to and is as follows:

$$\boldsymbol{i}_t = \sigma(\mathbf{W}_i \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_i). \tag{1.7}$$

As well as that, the cell creates a new vector of values, which could be potentially added to the cell state. This is created through an equation shown below:

$$\tilde{\boldsymbol{C}}_t = \tanh(\mathbf{W}_C \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_C). \tag{1.8}$$

The outputs of the two calculations are combined to be added to the cell state. The following operations describe the combining of the outputs of the explored gates:

$$\boldsymbol{C}_t = \boldsymbol{f}_t \cdot \boldsymbol{C}_{t-1} + \boldsymbol{i}_t \cdot \tilde{\boldsymbol{C}}_t. \tag{1.9}$$

The final edit the cell makes is done to the hidden state, where a sigmoid layer created a $\boldsymbol{o}_t$ vector using the following equation, $\boldsymbol{o}_t = \sigma(\mathbf{W}_o[\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_o)$ and combines it with the *tanh* of the cell state. The result is then outputted and is shown below:

$$\boldsymbol{h}_t = \boldsymbol{o}_t \cdot \tanh(\boldsymbol{C}_t). \tag{1.10}$$

The three gates use a sigmoid activation function, as shown in Fig. 1.5. As its outputs lie between 0 and 1, the function output can be interpreted as turning off the information flow when the value is 0 and allowing the full flow of information when the value is 1 [20].

The bidirectional LSTM (BiLSTM) neural network structure can be found in Fig. 1.6. As mentioned in [22], the BiLSTM processes the data in both directions, the forward and the backwards, allowing the model to process both past and future context. BiLSTM can process both directions simultaneously. Both directions have a separate hidden layer, which is fed to the same output layer [23].

As the input data is passed to LSTM layers, the hidden states and the cell states get updated in the same manner as for a single LSTM cell, as stated by [22]. The process of the backward pass works similarly to the forward pass, with the difference in the input sequence being fed to the model in reverse order. At each time step, the hidden states from both layers get combined.

1.3.4 Random Forest

The Random Forest algorithm is used as an alternative to the ANN approaches. The RF model [25] comprises many decision trees independent

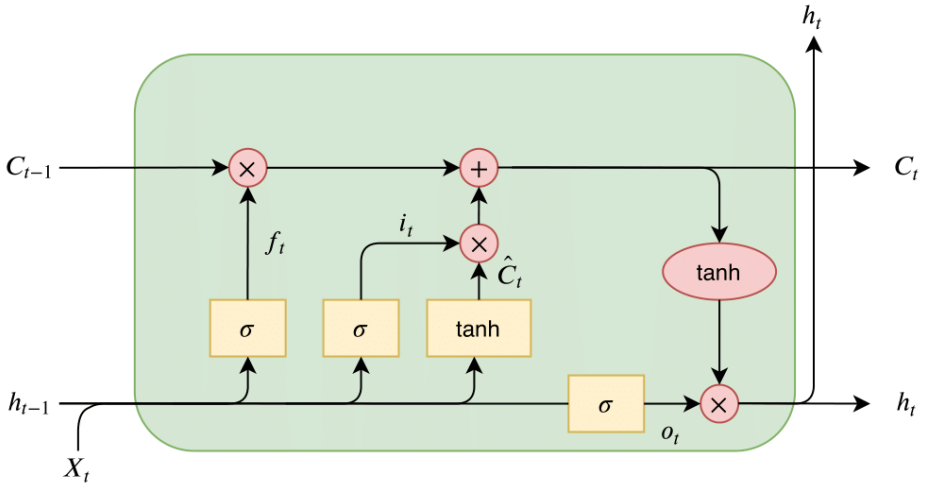

Figure 1.5: Long Short-Term Memory structure [21].

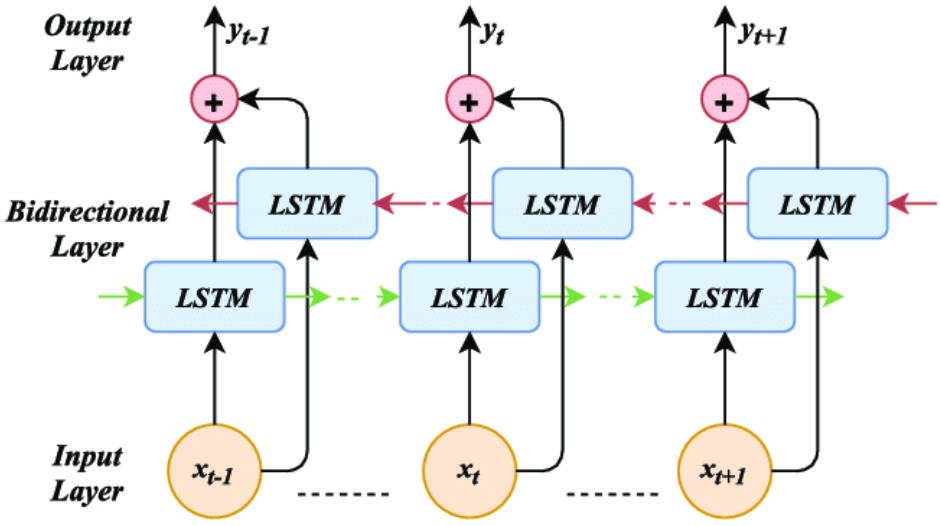

Figure 1.6: Bidirectional Long Short-Term Memory [24].

of one another. In the case of a classification problem, the majority vote of all decision trees is the final output. In contrast, in the case of regression, the individual outputs of the decision trees are averaged to create the final output.

As explained in [26], from a training dataset D, the model creates n datasets $D_1, ..., D_n$ of the same size using Bootstrap (selection with repetition). A tree is created and trained on each newly created dataset. The RF model consists of created trees $T_1, ..., T_n$. Each data point is then run through each tree in the forest for testing. The result is created by a majority vote of outputs from all trees. This process can be seen in Fig. 1.7.

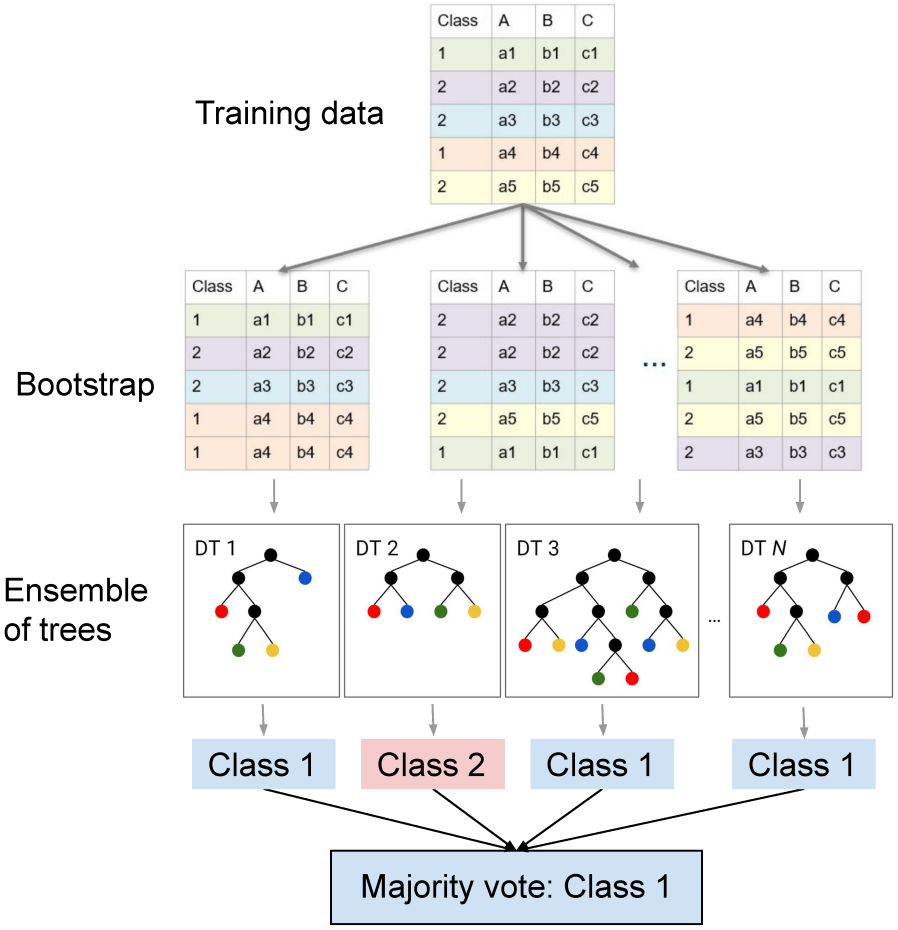

Figure 1.7: Random Forest model [27].

Chapter 2

Experiments

In this section, we will look in detail at the dataset presented by the previous study [2] and discuss the changes we have made to better work with the data. In addition, we will analyze the results we've achieved and explain the process of achieving them.

2.1 Dataset

Previously in studies [2, 5], a pool of 10^{15} unique DNA nanostructures of the same length was exposed to the target thrombin protein. Sequences that have not bonded or bonded weakly were washed away, so only the binders remained in the pool. The washing intensified as the rounds progressed, so only the strongest binders were left. The process of SELEX was repeated eight times, and only in the 5th cycle did the sequencing begin.

The dataset used for our research comprised four rounds (rounds 5 to 8) of SELEX. Each data file contained molecules that survived the washing procedure of SELEX from a specific cycle and the number of its occurrences.

A sample taken from each cycle captured a different number of sequences. In round 5, 891 959 sequences bonded, of which 891 914 were unique. After round 6, 735 974 unique out of 736 436 sequences were recorded. In round 7, the number of sequences increased to 750 926, of which 744 597 were unique, and in round 8, the number of bonded molecules decreased to 725 431 with 719 413 unique sequences.

Each recorded molecule was described as a sequence of 40 nucleotides with four base types (A, C, G, and T). An example of data from a single cycle is shown in Table 2.1. First, a unique sequence name was recorded, followed by the sequence's occurrence and the aptamer on the following line.

```
>seq0-2
AGGGTTGGGGAGGGTGGAGACGTTCGGTTGGGGGGGCGGA
>seq1-2
GGGGTTGGTGTGGGTGGATATAGCCCCGCTGTAAGCAAAC
>seq2-2
AGCAGCCAGCACGGTGAGGTCAGTGGGTGGTGAGGTTGGG
>seq3-2
GATGGTTGGGAGGATTGGTTCCACGGTAGTTTACGTGTAC
```

Table 2.1: Initial data example

2.1.1 Statistics

We inspected all datasets for duplicates to determine how to work with our data to achieve the highest accuracy. None of the provided datasets contained any duplicates. Across all the datasets, we worked with 3 081 855 unique sequences.

We then experimented with molecule occurrences across different rounds, firstly checking the consecutive rounds. Rounds 5 and 6 record 137 common sequences, 891 777 did not survive the latter round, and 735 837 sequences appeared in round 6 without being present in round 5.

As shown in Table 2.2, precisely 1822 sequences appeared in both rounds 6 and 7. 734 152 molecules were present in round 6 but were not recorded in round 7, and 742 775 molecules appeared in round 7 without being a part of the round 6 data.

From rounds 7 to 8, precisely 712 536 new sequences appeared, and 737 720 sequences disappeared to the next round. 6 877 sequences were present in both cycles.

105 sequences appeared only in rounds 5 and 7, all with occurrence equal to 1. 1016 sequences showed up only in rounds 6 and 8, of which only 8 had a higher occurrence than 1. Rounds 5 and 8 had 68 common sequences not detected in other rounds. Of these sequences, none had an occurrence higher than 1. No sequence was present, only in a single cycle. Thus, we can conclude that all sequences that appear newly created during a cycle were always a part of the dataset but were not recorded by mistake or by not being present in the sample.

rounds	disappeared	stayed	appeared
round 5 to 6	891 777	137	735 837
round 6 to 7	734 152	1822	742 775
round 7 to 8	737 720	6877	712 536

Table 2.2: Representation of sequences across consecutive rounds

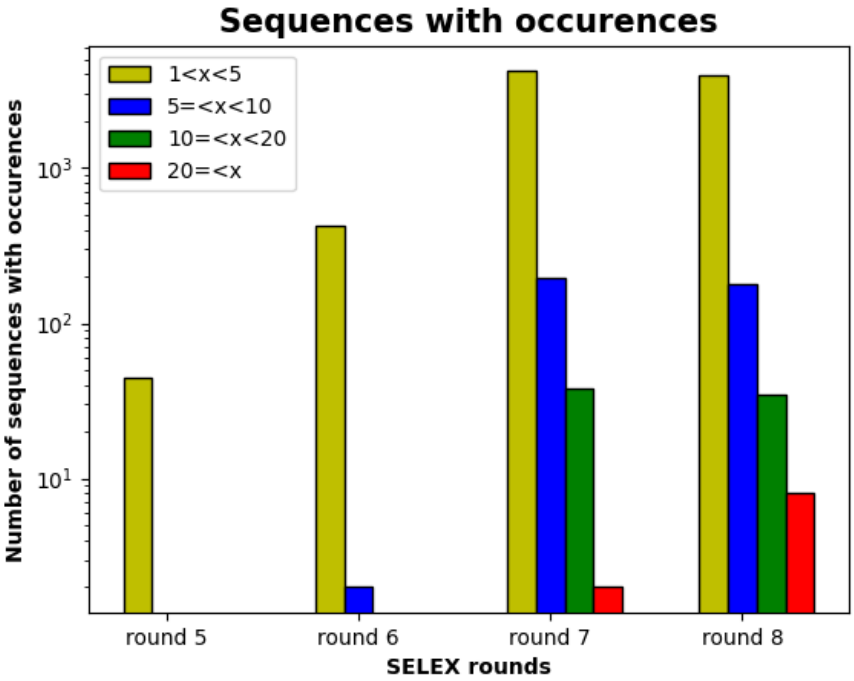

Figure 2.1: Sequences with occurrences higher than 1

As we can see in Fig. 2.1, the sequences of higher occurrences increased across the rounds. Round 5 only contained sequences of counts 1 and 2. The number of sequences of occurrence between 1 and 5 increased in the following cycle, and a few sequence occurrences rose between 5 and 10. In the 6th round were only two sequences of counts higher or equal to 5: one of count 5 and the other of count 6. In round 7, two sequences of occurrence 22 appeared, being the only two thus far to reach occurrence above 20. The highest occurrence of 29 during the SELEX process was reached in the 8th cycle.

Even though the number of sequences of higher occurrences increased in the first rounds, there was a decrease from round 7 to round 8, as seen in Fig. 2.2. Even though the reason for this would seem to be the increase in counts of sequences, thus causing a lower number of unique sequences of higher counts, as can be seen in Fig.2.3, both the number of unique and duplicated sequences decreased from round 7. Hence, we assumed this to be the cause of a stronger washing.

As shown in Fig. 2.3, the number of unique sequences and all sequences drastically decreased from round 5 to round 6. The trend of disappearing sequences continued across all rounds due to the intensified washing and the domination of stronger binders of the pool. The only exception was the 7th

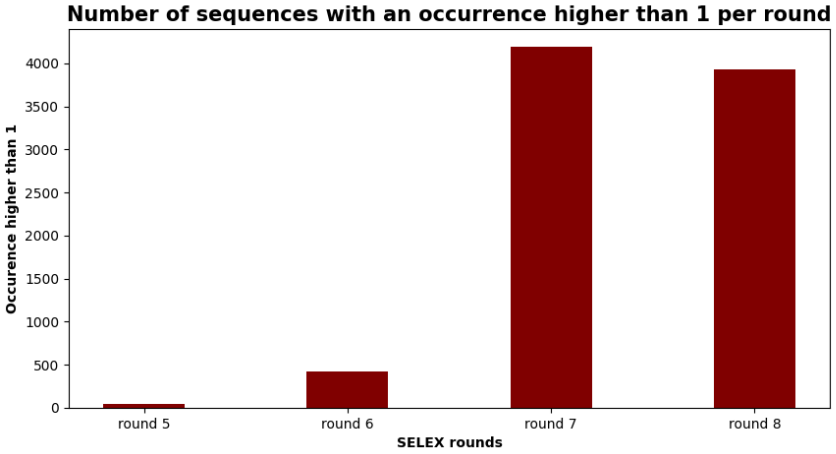

Figure 2.2: Sequences with an occurrence higher than 1 per round of the SELEX process

round, where the sequence counts increased, as seen in Fig.2.1, and so did the number of all sequences. This may result from a sequencing error, as well as caused by the limited sampling of the whole pool [2]. As the sequence sequencing is done only on a sample, round 6 may have contained more sequences which did not show up in the reading. Further, both values increased in the following cycle and decreased again in the 8th pool, which contained the least unique sequences of all recorded rounds.

2.1.2 Preprocessing

Because we decided to use binary classification to find whether an aptamer is a binder, we had to determine what would classify each sequence as 1 or 0. We experimented with using single rounds or joining multiple rounds.

As the aptamers were sequenced only after the washing-off process, we do not have an account of the aptamers that did not survive the cycle. Because we do not have records of the initial pool, we can not know which aptamers were washed away without inspecting the previous SELEX rounds. Therefore, if we wanted to work with only single cycles, we would have to classify sequences with occurrence of 1 as 0 and sequences with higher counts as 1. As the number of sequences classified as 0 far surpassed those classified as 1, this method proved insufficient for our research as it meant the data was unbalanced. Moreover, it would mean we would have to consider some successful bonds poor. Considering the bonds of sequences with lower counts as poor would cause other issues as the sequences may have been sufficient binders

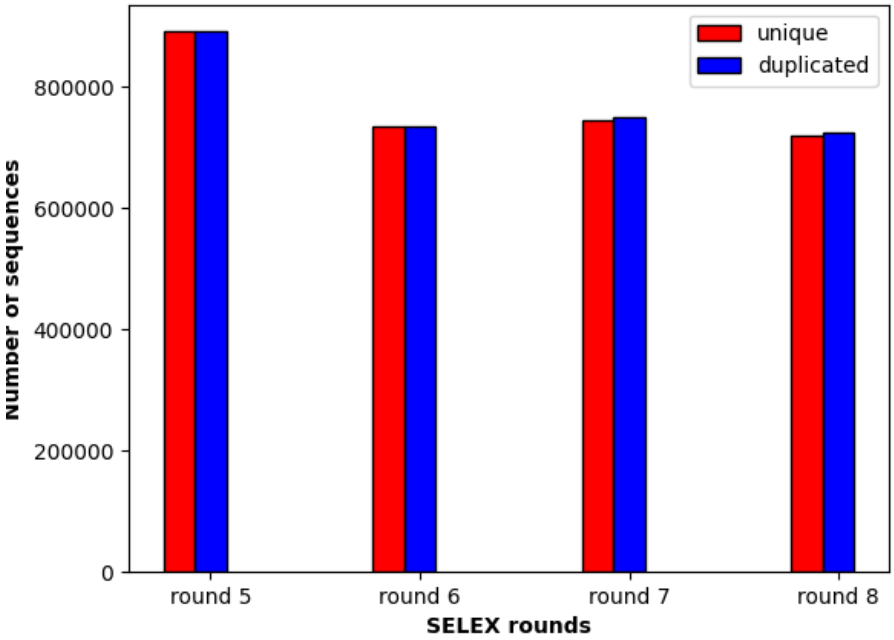

Figure 2.3: Unique and duplicate sequences per round

but were not all captured in the sample. Therefore, we rejected this method and concentrated our research on multiple-cycle training.

To do that, we first decided to convert the sequences into dataframes where each letter of the molecule would make up a single value. Thus, we created dataframes with 40 columns containing nucleotides and 1 column in which we stored their occurrences. Consecutively, we merged two consecutive rounds and decided to concentrate on the occurrences in the latter round as a concluding factor for each sequence's classification.

If the molecule showed up in the later round, we considered it a successful bond (ignoring that some of these sequences were recorded multiple times). If it wasn't, we thought it to be washed away, so it was classified as 0.

Even though we considered this method to be the best, as it meant that our data would be balanced, there are downsides to this type of labelling. As the reading process of the molecules in SELEX is done from a sample, there's a possibility that some molecules that have bonded and remained in the pool for the next cycle were not recorded. In addition, there may be an error when recording the molecules, so a change in the base type of a single nucleotide creates an entirely different sequence. This mistake is undetectable afterwards. Thence, it may happen that molecules with an occurrence of 1 were never a

part of the dataset and resulted from a reading error. On the other hand, as stated in Section 2.1.1, no molecules appeared in a single cycle. Therefore, this type of mistake could have either been made in multiple rounds, not been made, or was made, but its result increased the count of a sequence already present in the dataset.

Further, we changed the data structure with one-hot encoding, found in Fig.2.4. This way, each record would comprise an array containing 4 nested arrays, each of the size of a single aptamer. Each nucleotide would then be saved as a 1, depending on its base type and position in the sequence. Nucleotides of type A were saved in the first array, type C in the second, type G in the third, and type T in the last array.

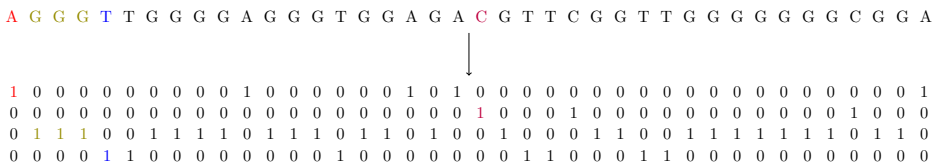

Figure 2.4: An example of the preprocessing process

We considered two possible approaches for model training: to use the unique counts of sequences or to duplicate the sequences depending on their occurrence in the latter cycle. The first approach would ensure that few good binders would not dominate the dataset. Because it would not consider the aptamer's occurrence in the later round, the model could not give a more significant advantage to the features of the better binders. With the second approach, the model could learn the features of better binders but might not be able to recognise other features from different aptamers, as the dataset could be polluted with many values of few sequences.

The second approach required the data to be further preprocessed, so sequences of higher counts were duplicated in the dataset depending on their occurrence. For all duplicates, the classification stayed at 1.

At this stage, the data followed a spatial arrangement and had the ideal structure for a CNN. The data was randomly separated into training, validation, and test data. 80% of the data was assigned to the training process, of which 75% went to the training data. The remaining 20% were allocated to the test data. In the case of ANNs, the data needed to be further converted into DataLoaders. We set the batch size equal to 128 for all ANN models.

2.2 Used Technologies

The employment of all ANN models involved the use of the Jupyter Notebook[6] environment and the use of Pytorch[7] – a Python-based library for building neural networks. As we used the Python programming language, the model implementation also involved using pandas[8] and NumPy[9] libraries. We additionally used the scikit-learn[10] library to implement the Random Forests. Furthermore, we used the scikit-learn built-in techniques to evaluate the models' predictions.

2.3 Results

As mentioned in Section 2.1.2, we experimented with ML models and applied them on two joined cycles. To test the models thoroughly, we trained and tested them on data from different rounds, including sequential and nonsequential rounds. In addition, we tried two approaches: using unique sequences and duplicated sequences.

The main reason for using the consecutive rounds was that if we'd be able to train the models to predict which aptamers would get to the next round in the SELEX process, we'd be able then to use the trained model on the initial pool and run it as many times as needed to receive the best aptamers from a particular cycle. On the other hand, as we necessarily do not need to find out which aptamers survived a single cycle of the SELEX procedure and instead can explore only those that got to the final cycle and proved themselves to be the best binders, we also experimented with training the model on nonsequential rounds such as 5 and 8.

We preprocessed our data to receive training, validation, and test datasets. Later, we discovered that the best way to compare our models with RBM was to use the sequences generated by RBM in former research [2]. As we decided that the prediction of these aptamers' binding ability was the key issue, we proposed a model which would use the whole previously used data of two joined rounds in only two parts: training and validation subsets. This way, we would let the model learn on the whole data and hopefully achieve better results. The model testing would then be performed on newly generated data from the former research [2] shown in Tab. 2.3. Thus creating the following approaches to model training:

- $data \in \{\mathbf{U} - unique, \mathbf{D} - duplicate\}$;

[6] `https://jupyter.org/`
[7] `https://pytorch.org/docs/stable/optim.html`
[8] `https://pypi.org/project/pandas/`
[9] `https://numpy.org/`
[10] `https://scikit-learn.org/stable/`

- *approach* $\in \{\mathbf{M1} - (train, validation, test), \mathbf{M2} - (train, validation)\}$.

When combining the approaches and data techniques, we create four possible approaches: M1-U, M1-D, M2-U, and M2-D. To find out which is the best for our problem, we ran them all through our chosen algorithms.

As we can see, the Tab. 2.3 contains only sequences of length 20, whereas until now, we were strictly using aptamers of length 40. Our aptamers consist of 2 joined single-helix aptamers of length 20 [2]. To test the models on this new data, we had to convert them to the same structure as the initial dataset. When two binding single-helix aptamers are combined, they create a binder, and vice versa with non-binders (Petr Šulc, personal communication, April 19, 2024). Thus, we concatenated the sequences from Tab. 2.3 to create a dataset with the right dimensions.

As we generated all possible combinations which gave a certain SELEX result (two binders or two non-binders), 793 new sequences were created – 64 non-binders and 729 binders. Since the dataset was not balanced, we could not use accuracy to measure success efficiently. For this reason, we also used the F1 score. Predictions of the best-performing models of all employed algorithms were recorded in Tab. 2.3 to be compared to RBM.

2.3.1 Feed-Forward networks

To find the FFNN structure best suitable for aptamer binding prediction, numerous FFNNs were tested with various hidden layers of a varying number of neurons. Only the input size of the first layer and the output of the last layer stayed unchanged. The input size of the first layer consisted of 160 elements, as the input contained 4 arrays of 40 elements each. Whereas the output layer's output size stayed consistent at 2.

All implemented FFNNs contained a Rectified Linear Unit (ReLU) activation function in their hidden layers and a linear function in their output layers. The model tuning consisted of experimenting with the following hyperparameters:

- *optimizers* $\in \{\texttt{Adam}, \texttt{AdamW}, \texttt{RMSprop}, \texttt{SGD}\}$
- *learning rate* $\in \{0.01, 0.05, 0.1\}$;
- *number of hidden layers* $\in \{1, 2, 3\}$;
- *layer input and output sizes* $\in \{16, 32, 64, 128, 256\}$;
- *dropout* $\in \{0, 0.1, 0.2\}$;
- *activation* $\in \{ReLU, sigmoid\}$.

sequence	label	FFNN	CNN	RF	BiLSTM	RBM
AGTGATGATGTGTGGTAGGC	**0**	1	1	1	1	0
AGTGTAGGTGTGGATGATGC	**0**	1	1	1	1	0
TAGGTTTTGGGTAGCGTGGT	**0**	1	1	1	0	0
AGGGATGATGTGTGGCAGGA	**0**	1	1	0	1	0
CTAGGACGGGTAGGGCGGTG	**0**	1	0	0	0	0
AGGGATGTGTGTGGTAGGCT	**0**	1	1	1	1	0
AGGGATGCTGCGTGGTAGGC	**1**	1	1	1	1	1
GAGGGTTGGTGTGGTTGGCA	**1**	1	1	1	1	1
AGGGTTGGTGTGTGGTTGGC	**1**	1	1	1	1	1
ATGGTTGGTTTATGGTTGGC	**1**	1	0	1	1	1
GAAGGGTGGTCAGGGTGGGA	**1**	0	0	1	1	1
GGAGGGTGGGTCGGGTGGGA	**1**	0	0	1	1	1
GGGGTTGGTACAGGGTTGGC	**1**	1	1	1	1	1
AGATGGGCAGGTTGGTGCGG	**1**	1	0	1	1	1
AGATGGGTGGGTAGGGTGGG	**1**	0	1	1	1	1
ATAGGGTGGGTGGGTGGGTA	**1**	1	1	1	0	1
TGGTGGTTGGGTTGGGTTGG	**1**	1	0	1	1	1
TGGGATGGGATTGGTAGGCG	**0**	1	1	1	1	1
AGGGTTGGTTATGTGGTTGG	**1**	1	1	1	1	1
ATTGGTTGGGTAGGGTGGTT	**1**	1	1	1	1	1
AAACGGTTGGTGAGGTTGGT	**1**	1	1	1	1	1
CGGGGTGGTGTGGGTGGGAG	**1**	0	1	1	0	1
TATTGGTTGGATAGGTTGGT	**1**	1	1	1	1	1
AGGGTTGGGTGGTTGGATGA	**1**	1	1	1	1	1
CGGGTTGGGGGGTTGGATTC	**1**	1	1	0	0	1
CGGTTGGGGGGGTTGGATAC	**1**	0	0	0	0	1
TGTGGGTTGGTGAGGTAGGT	**0**	1	0	1	1	1
AGGGATGATGTGTGGTAGGC	**1**	1	1	1	1	1
GTAGGATGGGTAGGGTGGTC	**1**	1	1	1	1	1
AGGGATGATGTGTGGTTGGC	**1**	1	1	1	1	1
AGGGATGGTGTGTGGTAGGC	**1**	1	1	1	1	1
AGGGTTGATGTGTGGTAGGC	**1**	1	1	1	1	1
AGGGATGGTGTGTGGTTGGC	**1**	1	1	1	1	1
AGGGTTGATGTGTGGTTGGC	**1**	1	1	1	1	1
AGGGTTGGTGTGTGGTAGGC	**1**	1	1	1	1	1

Table 2.3: Sequences created by RBM in paper [2] and tested by SELEX

The best accuracy showed a model of 2 hidden layers with a sigmoid activation function at the first hidden layer. The layers comprised $64 \times 128 \times 16 \times 2$ neurons. The model used the `SGD` optimizer with a learning rate of 0.01. It scored 88.2% using the M1-D approach. We applied this same model to the M1-U approach and reached an accuracy of 88.1%.

When we tested the additional data from Tab. 2.3, the M1-U approach reached 0.027 on the F1 score. The M1-D, M2-U, and M2-D approaches performed even worse, achieving an F1 score of 0.0. As the results were too poor, we experimented further with adding regularization techniques. The technique we have used was the dropout technique. We tried setting the dropout rate to 0.1 and 0.2 to test its effect on the predictions.

For both of the M1 approaches, the accuracy of the initial testing stayed unchanged. Both approaches' accuracies stayed high on the final testing. However, the F1 score reached 0.0 for all approaches apart from the M1-U approach, with a dropout rate 0.2. With the dropout rate increase, the M1-U approach's F1 score hit 0.0 at the dropout rate of 0.1 but rose again to 0.031 at the 0.2 dropout rate, being the best result across all attempts. The rest of the approaches (M1-D, M2-U, M2-D) showed no improvement when the dropout rate was increased.

The results of all models can be seen in the comparison Table. 2.4. As mentioned previously in 2.3, the M2 approaches were not tested on initial data and instead used all initial data to learn on. For this reason, the column of initial testing is not filled for these rows.

dropout	approach	initial testing (%)	final testing (%)	final testing (F1 score)
0	M1-U	88.1	82.2	0.027
0	M1-D	88.2	84.2	0.0
0	M2-U	–	80.7	0.0
0	M2-D	–	81.5	0.0
0.1	M1-U	88.1	84.4	0.0
0.1	M1-D	88.2	84.1	0.0
0.1	M2-U	–	84.4	0.0
0.1	M2-D	–	78.0	0.0
0.2	M1-U	88.1	84.4	0.031
0.2	M1-D	88.2	84.6	0.0
0.2	M2-U	–	82.4	0.0
0.2	M2-D	–	80.5	0.0

Table 2.4: FFNN approaches comparison

In the process of hyperparameter tuning, we discovered the following observations:

- `RMSprop` optimizer performed the worst as it showed practically no signs of learning.

- `SGD` optimizer performed the best out of all optimizers.
- The optimal number of hidden layers was 2.
- The model's prediction accuracy stayed unchanged for both M1 approaches across all tried dropout rates.
- The models did not achieve higher accuracy or F1 scores using higher learning rates. The used model had an issue learning when a learning rate 0.1 was used.
- The models using the M1-U approach showed the best results on the final testing, whereas the M1-D approaches showed the best accuracy on the initial data.

The approaches with an F1 score equal to 0.0 had an issue predicting non-binders, as they predicted them all as binders. Their predictions of binders were better, and so the model performed well on the final testing accuracy using all approaches.

The predictions of the model of 0.2 dropout rate using the M1-U approach can be found in Fig. 2.5. As we can see, the model predicted most sequences as binders (value of 1). The model could predict only 2 non-binders of 64 and 668 of 729 binders correctly. Even though the model using the M1-U approach scored the best at the final testing F1 score, the model still performed insufficiently. As the best-performing model cannot fully differentiate binders from non-binders of the final test data, the FFNN is inadequate for aptamers' binding ability prediction.

The best-scoring FFNN model's predictions were recorded in Tab. 2.3. The model was unable to predict any of the non-binders correctly.

2.3.2 Convolution networks

Our CNN model consisted of multiple hidden 1D convolution layers, followed by max pooling. The input layer, for all CNNs tested, consisted of the input size 4 as the input consisted of 4 arrays. The last layer – the only linear layer in the structure – contained an output size of 2. The activation function utilized in our CNN models was the ReLU function in all hidden layers.

Our CNN model tuning consisted of the same processes as the FFNN. Hyper-parameters such as optimizers, learning rates, number of layers, and the number of neurons were tuned to find the best model. Furthermore, we experimented with different kernel sizes of the input layer. All tried hyperparameter values can be found below:

- *optimizers* $\in \{$`Adam`, `AdamW`, `RMSprop`, `SGD`$\}$;
- *learning rate* $\in \{0.01, 0.05, 0.1\}$;

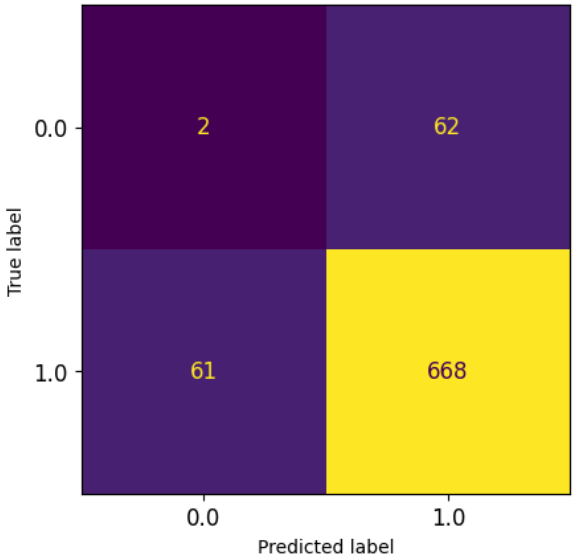

Figure 2.5: FFNN best-performing architecture predictions

- *number of hidden layers* $\in \{1, 2, 3\}$;
- *layer input and output sizes* $\in \{16, 32, 64, 128, 256\}$;
- *kernel size* $\in \{3, 5\}$;
- *dropout* $\in \{0, 0.1, 0.2\}$.

The CNN model structure, which performed the best, consisted of 2 hidden convolution layers, an input convolution layer, and a linear output layer. The model consisted of $64 \times 256 \times 32 \times 2$ neurons. A kernel size of 5 was used in the first layer and was followed by layers with the kernel size of 3. The model used the `SGD` optimizer with a learning rate of 0.01. This model achieved an initial accuracy of 88.2% using the M1-D approach and 87.8% accuracy using the M1-U approach. Using the M1-D approach, the model reached a high accuracy on the final data, but the F1 score reached 0.0, as it could not predict a single non-binder correctly. The model's performance using the M1-U approach was worse in accuracy of both the initial and the final data, but the M1-U approach managed to score better on the F1 score as it reached the score of 0.027. The scores are shown in Tab. 2.5.

When the model used the M2-U approach and learned the whole initial data, the model's performance of the final predictions improved not only in accuracy but in F1 score as well. The model was now able to predict the sequences with an accuracy of 83.4% and the F1 score of 0.143, being the best result thus far. The increase in data for training, unfortunately, did not help

the model with the M2-D approach accomplish better results, as the accuracy and the F1 score stayed the same as for the M1-D approach.

To hopefully improve the model's prediction ability, we added the dropout regularization technique. The model achieved the best results using the M1-D approach. The F1 score using the M1-D approach rose to 0.183 with a dropout rate of 0.1 and 0.177 with a dropout of 0.2. Even though the F1 score using dropout increased, the accuracy of the predictions of the final data testing significantly decreased with the increase in the dropout rate. Thus, the model did not achieve better results using this regularization technique with the M1-U approach.

dropout	approach	initial testing (%)	final testing (%)	final testing (F1 score)
0	M1-U	87.8	82.0	0.027
0	M1-D	88.2	85.4	0.0
0	M2-U	–	83.4	0.143
0	M2-D	–	85.2	0.0
0.1	M1-U	83.1	77.4	0.028
0.1	M1-D	82.1	82.0	0.183
0.1	M2-U	–	84.7	0.062
0.1	M2-D	–	80.5	0.094
0.2	M1-U	80.4	65.5	0.080
0.2	M1-D	80.3	81.3	0.177
0.2	M2-U	–	66.2	0.112
0.2	M2-D	–	67.9	0.130

Table 2.5: CNN approaches comparison

From the exploration of different CNN models, we have found the following observations:

- The change in learning rate showed no significant change in the model's ability to predict binders on both the initial and the final dataset, apart from the M2-U approach model with the dropout rate set to 0.2. This model showed no signs of learning at a 0.01 learning rate.
- The best performing optimizer was `SGD`.
- The dropout rate increase overall helped the model accomplish better results on the F1 score but, for most approaches, significantly decreased the model's accuracy on the final data.

As both the accuracy and the F1 score of the M1-D approach with a dropout of 0.1 was high, we concluded this CNN model structure using the M1-D approach to be the best for this problem. Hence, we filled the model's predictions to Tab. 2.3 to compare its results to other algorithms. Additionally, the model's predictions are shown in a confusion matrix in Fig. 2.6. As

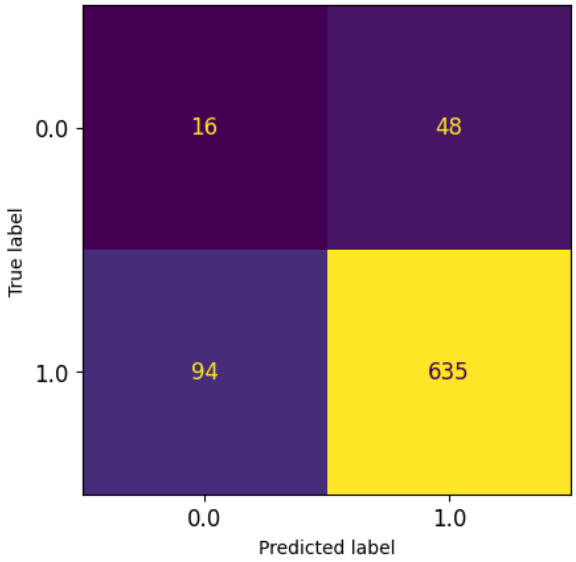

Figure 2.6: CNN best-performing architecture predictions

we can see, the model using the M1-D approach with a dropout rate of 0.1 could predict 16 of 64 non-binders and 635 of 729 binders correctly.

2.3.3 Bidirectional Long Short-Term Memory

To employ the best BiLSTM neural network model, we experimented with the hidden size values, which we kept relatively low, and the number of layers. As in the other ANN models, we kept the number of training epochs equal 100 with early stopping. All our BiLSTM models consisted of an input bidirectional LSTM layer and a linear output layer of 2 neurons. Nonetheless, we experimented with the number of linear layers inside the model following the LSTM layers. During experimentation, the following hyperparameters were explored:

- *optimizers* $\in \{\texttt{Adam}, \texttt{AdamW}, \texttt{RMSprop}, \texttt{SGD}\}$;
- *learning rate* $\in \{0.01, 0.05, 0.1\}$;
- *hidden size* $\in \{32, 64, 128\}$;
- *number of layers* $\in \{2, 3\}$;
- *dropout* $\in \{0, 0.1, 0.2\}$.

The following BiLSTM model showed the best performance on the initial data. The model contained the hidden size of 32 and 3 LSTM layers followed

by a linear output layer of 2 neurons, with no additional hidden linear layers. The `AdamW` with the learning rate of 0.01 helped the model achieve the score of 88.4% accuracy using the M1-D approach and 88.2% using the M1-U approach. Even though the model using the M1-D approach reached better accuracy on the initial data, its F1 score on the final data came out much worse than for the M1-U approach. The model could still reach a high accuracy score on the final data, but the F1 score came to 0.0 using the M1-D approach and 0.210 using the M1-U approach. The results of different approaches applied to the same model are shown in Tab. 2.6.

The model's learning of the entire data (M2 approaches) did not help the model accomplish better results. The model's accuracy reached 0.037 using the M2-U approach and 0.0 using the M2-D approach. Even though the model could train on more data, the results worsened as even the accuracies of both approaches were worse than via the M1 approaches. Adding the dropout regularization technique did not help the model achieve better results, as the F1 scores were close to 0.

dropout	approach	initial testing (%)	final testing (%)	final testing (F1 score)
0	M1-U	88.2	81.0	0.210
0	M1-D	88.4	81.2	0.0
0	M2-U	–	80.5	0.037
0	M2-D	–	79.0	0.0
0.1	M1-U	88.3	79.1	0.011
0.1	M1-D	88.4	82.8	0.014
0.1	M2-U	–	84.1	0.0
0.1	M2-D	–	78.9	0.0
0.2	M1-U	88.1	80.7	0.012
0.2	M1-D	88.2	82.9	0.0
0.2	M2-U	–	85.6	0.0
0.2	M2-D	–	81.2	0.062

Table 2.6: BiLSTM approaches comparison

As we can see, even though the model results on the initial data were the best across employed ML algorithms, the model's ability to predict the final data drastically decreased. The accuracy of the final data may have stayed relatively high, but the F1 score was below substantial. Many of the approaches predicted all non-binders to be binders and so received an F1 score of 0.0. The best-performing approach on the F1 score was the M1-U approach without the dropout technique, which scored 0.210 on the final data.

The observations we made during the experiments are stated below:

- The addition of a linear layer did not help the model with performance on the initial test data. The best performance showed a model with a linear layer containing 128 neurons, scoring 88.2%. When the number

approach	hidden size	LSTM layers	optimizer	additional neurons	final accuracy (%)	F1 score
M1-D	32	3	`SGD`	256	85.3	0.159
M1-D	32	3	`SGD`	32	83.4	0.176
M1-D	32	3	`SGD`	64	87.2	0.240
M1-D	32	3	`SGD`	–	81.2	0.118
M1-U	32	3	`SGD`	–	81.9	0.122
M1-U	64	3	`SGD`	–	86.7	0.146

Table 2.7: Other BiLSTM model structures' performances

of neurons in the layer changed, the accuracy decreased to the lowest value of 88.0%.

- The additional linear layer helped the model perform better on the final test dataset.
- The best performance had the `AdamW` optimizer.
- The increased learning rate did not improve the model's learning ability. The optimum learning rate was shown to be 0.01.
- The model using the M1-U approach performed slightly worse than the M1-D approach on the initial data, but its overall performance on the test data was much better.

The identical BiLSTM structure using `SGD` optimizer achieved a slightly lower accuracy of 88.3%. As the model using `AdamW` optimizer did not perform well on the F1 score of the final data, we tested other model structures using `SGD` optimizer. Other models scored similarly on the initial data, all with an accuracy of about 88.2%. Some of these models are shown in Tab. 2.7. The BiLSTM architectures which did not contain an additional layer have the column of additional neurons unfilled. As we can see, even though the results of these models were lower on the initial data, these models far surpassed the previously described BiLSTM model on the final testing.

The model structure of hidden size of 32, 3 LSTM layers, an additional linear layer with 64 neurons using the `SGD` optimizer applied on the M1-D approach reached the highest F1 score of all tried models. We recorded the results of this model's prediction to Tab. 2.3 to compare them to the performances of other models. The model's predictions are shown in Fig. 2.7. The best-performing model could predict 16 of 64 non-binders and 676 of 729 binders correctly.

2.3.4 Random Forests

The hyperparameter of the number of trees was tested from the value of 1 up to 70. After training the Random Forests of varying numbers of trees, the RF model's ability to predict scores for validation data was tested and

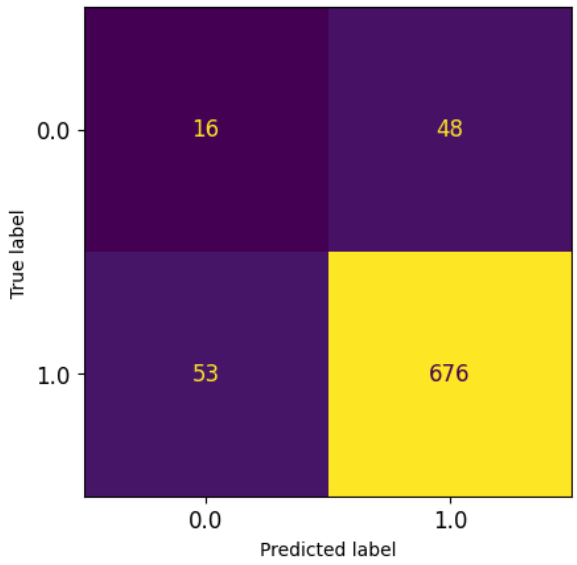

Figure 2.7: BiLSTM best-performing architecture predictions

saved. Lastly, the model with the highest validation accuracy was tested on the remaining data to give the final accuracy of the aptamers' binding ability.

Both approaches with unique values (M1-U and M2-U) performed on the validation data best at 69 trees, whereas both duplicate value approaches (M1-D and M2-D) scored the best at 70 trees. The approach M1-D performed slightly better at 87.9% on the initial testing than the M1-U approach, which scored 87.8%.

Afterwards, we tested all approaches' prediction ability on newly generated data from Table 2.3. At the final testing, the M2-U performed the best, scoring 0.392 on the F1 score, followed by M1-U at 0.337, M2-D at 0.297, and finally, M1-D at 0.266. The models scored well on both the final testing accuracy and the F1 score. The results of our experiments can be seen in Tab. 2.8.

approach	trees	initial data testing (%)	final testing (%)	final test (F1 score)
M1-U	69	87.8	91.5	0.337
M1-D	70	87.9	90.2	0.266
M2-U	69	–	92.1	0.392
M2-D	70	–	91.0	0.297

Table 2.8: RF approaches comparison

As the M2-U approach model performed the best on the final data on both accuracy and F1 score, we decided to compare the results of this model to other algorithms' results. The results of its predictions can be seen in Tab. 2.3.

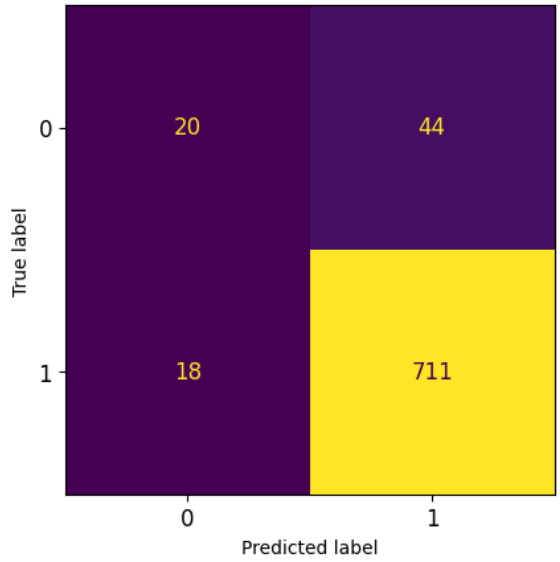

Figure 2.8: RF best-performing architecture predictions

The results of the M2-U approach model's final testing predictions can be seen in Fig. 2.8. The model could predict 20 of 64 non-binders and 711 of 729 binders correctly. Other models predicted fewer binders and fewer non-binders, proving the M2-U approach to be the best at predicting the data.

2.3.5 Discussion

Using the M1 approaches, we were able to train the models to predict the initial data with an accuracy of around 88% using all ML algorithms. These percentages were achieved through the merging of rounds 5 and 8. The merging of other cycles did not accomplish such results. Rounds 5 and 6 scored 78%, rounds 6 and 7 around 64% and rounds 7 and 8 barely 54%. Our reasoning for the decrease in models' ability to predict the aptamers in the consecutive rounds is that as the rounds progressed and the washing intensified, the aptamer structure became more similar as fewer parasite sequences were present in the data. As well as that, the use of consecutive cycles appeared to not be as effective, as stated in Section 2.1.1, as the sequences tended to disappear and reappear in nonsequential rounds. Therefore, we tried rounds 5 and 8, where the models performed the best. Presumably, the parasite sequences in cycle 5 disappeared through the rounds and were not present in round 8, creating a larger difference between the two pools.

As previously stated in Section 2.3, the results when using unique and duplicated data achieved similar results. We assume the results are overly similar as the duplication of sequences by occurrence does not make a significant dif-

ference between the two datasets. The unique joined rounds 5 and 8 contained 1 611 232 sequences, and the duplicate approach has 1 617 250 sequences, an increase of 6018 values. As this number is relatively small compared to the original value, the models do not show much difference when trained on the unique or the duplicate data.

Even though the models could predict the aptamer's binding ability on the initial dataset with high accuracy, their performance on the final data was insufficient. All algorithms had difficulty predicting non-binders correctly. The best-performing algorithms we have experimented with are the Random Forests and the BiLSTM. The Random Forests could predict the most binders and non-binders correctly of all ML algorithms. The BiLSTM predicted the aptamers the best of all tried ANN models, as its predictions were better using accuracy and F1 scores than other ANNs. We can not conclude a single approach is the best for the handling of this problem as each algorithm performed the best using a different one.

When we look back at Tab. 2.3, we can see that the best predictions overall were made by RBM presented in previous research [2], followed by Random Forest and the Bidirectional LSTM. As the data we compare the models on consists of only 35 sequences, our model's comparison with the RBM might not be as accurate, as the RBM's prediction accuracy could drastically differ with the increase of sequences. As well as that, the RBM generated these sequences. Thus, the aptamers in Tab. 2.3 could be biased towards this model. As we do not know whether the increase or change in sequences the models are compared on would make a significant difference to the results, we have to conclude that the RBM shows the best ability of aptamers' binding ability prediction.

As shown, these models are not substantial for solving this problem as even the best-performing models did not accomplish results substantial for use in the medical field. Therefore, if the aptamers' binding ability prediction should be automatised, there is much work to be done in the future. There are other possible approaches to consider, such as separating the initial sequences into two sequences of length 20. The models could then train on these shorter sequences. As well as that, experimentation with classifying sequences of occurrence 1 as 0 ensures that the models are less affected by the sequencing error. Furthermore, the weights of the ANN models could be initialized to the occurrence value of the sequences.

Conclusion

This thesis aimed to analyze the possibilities of supervised deep neural networks trained on sequence ensembles from rounds of SELEX experiments for thrombin aptamers. Our goals included:

- getting familiar with the SELEX experiment and with the provided experimental data from its several rounds;
- preprocess the provided data so they can be used for training of neural networks in a supervised manner;
- research and select two suitable architectures of deep neural networks that could be used to predict the binding strength between rounds of the experiment;
- train and evaluate the selected architectures on the provided data;
- compare the results with other possible learning approaches and discuss their consistency with recent RBM results.

The practical part of this thesis validated and expanded on the work of authors of paper [2]. Our work presented multiple approaches to handling this issue and suggested improvements for future work. The selected models' predictions achieved a high accuracy on a dataset presented in previous research. When tested on additionally generated data, the models had difficulty differentiating between binders and non-binders and, therefore, were concluded as insufficient for use in the medical field. The results of individual models and approaches were compared. Of all the algorithms, the best performance showed the Restricted Boltzmann Machines followed by Random Forests.

Bibliography

1. ZHOU, Jiehua; ROSSI, John J. Cell-type-specific, aptamer-functionalized agents for targeted disease therapy. *Molecular Therapy-Nucleic Acids*. 2014, vol. 3. Available from DOI: 10.1038/mtna.2014.21.
2. GIOACCHINO, Andrea Di; PROCYK, Jonah; MOLARI, Marco; SCHRECK, John S.; ZHOU, Yu; LIU, Yan; MONASSON, Rémi; COCCO, Simona; ŠULC, Petr. Generative and interpretable machine learning for aptamer design and analysis of in vitro sequence selection. *PLoS computational biology*. 2022, vol. 18, no. 9. Available from DOI: 10.1371/journal.pcbi.1010561.
3. D'SOUZA, Sofia; PREMA, K.V.; BALAJI, Seetharaman. Machine learning models for drug–target interactions: current knowledge and future directions. *Drug Discovery Today*. 2020, vol. 25, no. 4, pp. 748–756. Available from DOI: 10.1016/j.drudis.2020.03.003.
4. ADAMS, Christie. *Wet Lab vs Dry Lab: Challenges, Benefits and Skills Required — BioSpace — biospace.com* [online]. BioSpace, 2023. Available also from: https://www.biospace.com/article/wet-lab-vs-dry-lab-which-is-best-for-you-/. [visited on 17-04-2024].
5. ZHOU, Yu; QI, Xiaodong; LIU, Yan; ZHANG, Fei; YAN, Hao. DNA-Nanoscaffold-Assisted Selection of Femtomolar Bivalent Human α-Thrombin Aptamers with Potent Anticoagulant Activity. *ChemBioChem*. 2019, vol. 20, no. 19, pp. 2494–2503. Available from DOI: 10.1002/cbic.201900265.
6. GIOACCHINO, Andrea Di; PROCYK, Jonah; MOLARI, Marco; SCHRECK, John S.; ZHOU, Yu; LIU, Yan; MONASSON, Rémi; COCCO, Simona; SULC, Petr. *Data for "Generative and interpretable machine learning for aptamer design and analysis of in vitro sequence selection" — zenodo.org* [online]. 2022. Available also from: https://zenodo.org/records/6341687. [visited on 17-04-2024].

7. TUERK, Craig; GOLD, Larry. Systematic Evolution of Ligands by Exponential Enrichment: RNA Ligands to Bacteriophage T4 DNA Polymerase. *Science*. 1990, vol. 249, no. 4968, pp. 505–510. Available from DOI: 10.1126/science.2200121.

8. NANOSTRING. *Common questions in molecular biology: What is an example of an oligonucleotide? — nanostring.com* [online]. 2023. Available also from: https://nanostring.com/blog/what-is-an-example-of-an-oligonucleotide/. [visited on 21-04-2024].

9. *Polymer — Description, Examples, Types, Material, Uses, & Facts — britannica.com* [online]. Encyclopædia Britannica, inc., 2024. Available also from: https://www.britannica.com/science/polymer. [visited on 21-04-2024].

10. NARAYANAN, Sheshadri. Multifunctional roles of thrombin. *Annals of Clinical & Laboratory Science*. 1999, vol. 29, no. 4, pp. 275–280.

11. *Team:Madrid-OLM/AptDiscovery - 2018.igem.org — 2018.igem.org* [online]. [N.d.]. Available also from: https://2018.igem.org/Team:Madrid-OLM/AptDiscovery. [visited on 21-04-2024].

12. GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. *Deep learning*. MIT press, 2016.

13. OH, Sangchul; BAGGAG, Abdelkader; NHA, Hyunchul. Entropy, free energy, and work of restricted boltzmann machines. *Entropy*. 2020, vol. 22, no. 5, p. 538. Available from DOI: 10.3390/e22050538.

14. YAGAWA, Genki; OISHI, Atsuya. Feedforward Neural Networks. In: *Computational Mechanics with Neural Networks*. Cham: Springer International Publishing, 2021, pp. 11–23. ISBN 978-3-030-66111-3. Available from DOI: 10.1007/978-3-030-66111-3_2.

15. AHMADIAN, Sajad; KHANTEYMOORI, Ali Reza. Training back propagation neural networks using asexual reproduction optimization. In: *2015 7th conference on information and knowledge technology (IKT)*. IEEE, 2015, pp. 1–6.

16. PATTERSON, Josh; GIBSON, Adam. *Deep learning: A practitioner's approach*. " O'Reilly Media, Inc.", 2017.

17. BALAJI, Sai. *Binary Image classifier CNN using TensorFlow — medium.com* [online]. 2023. Available also from: https://medium.com/techiepedia/binary-image-classifier-cnn-using-tensorflow-a3f5d6746697. [visited on 21-04-2024].

18. HOCHREITER, Sepp; SCHMIDHUBER, Jürgen. Long Short-Term Memory. *Neural Computation*. 1997, vol. 9, no. 8, pp. 1735–1780. ISSN 0899-7667. Available from DOI: 10.1162/neco.1997.9.8.1735.

19. OLAH, Christopher. *Understanding LSTM Networks – colah's blog — colah.github.io* [online]. [N.d.]. Available also from: https://colah.github.io/posts/2015-08-Understanding-LSTMs/. [visited on 10-05-2024].

20. DIPIETRO, Robert; HAGER, Gregory D. Deep learning: RNNs and LSTM. In: *Handbook of medical image computing and computer assisted intervention*. Elsevier, 2020, pp. 503–519.

21. INGOLFSSON, Thorir Mar. *Insights into LSTM architecture — Thorir Mar Ingolfsson — thorirmar.com* [online]. 2021. Available also from: https://thorirmar.com/post/insight_into_lstm/. [visited on 04-05-2024].

22. ANISHNAMA. *Understanding Bidirectional LSTM for Sequential Data Processing* [online]. 2023. Available also from: https://medium.com/@anishnama20/understanding-bidirectional-lstm-for-sequential-data-processing-b83d6283befc. [visited on 12-05-2024].

23. GRAVES, Alex; JAITLY, Navdeep; MOHAMED, Abdel-rahman. Hybrid speech recognition with deep bidirectional LSTM. In: *2013 IEEE workshop on automatic speech recognition and understanding*. IEEE, 2013, pp. 273–278.

24. AUGUSTINE O. NWAJANA. *A Deep Learning Approach for Human Activities Recognition From Multimodal Sensing Devices - Scientific Figure* [online]. ResearchGate, [n.d.]. Available also from: https://www.researchgate.net/figure/Bidirectional-LSTM-model-showing-the-input-and-output-layers-The-red-arrows-represent_fig3_344554659. [visited on 04-05-2024].

25. RIGATTI, Steven J. Random forest. *Journal of Insurance Medicine*. 2017, vol. 47, no. 1, pp. 31–39. Available from DOI: 10.17849/insm-47-01-31-39.1.

26. KLOUDA, Karel; VAŠATA, Daniel. *Vytěžování znalostí z dat: Ensamble metody* [online]. Czech Technical University in Prague, Faculty of Information Technology, 2022-02. Available also from: https://courses.fit.cvut.cz/BI-VZD/lectures/files/BI-VZD-02-cs-handout.pdf. [visited on 17-04-2024].

27. *8. Image classification - Random Forests — pages.cms.hu-berlin.de* [online]. Humboldt-Universität zu Berlin. Department of Geography., 2023. Available also from: https://pages.cms.hu-berlin.de/EOL/geo_rs/S08_Image_classification2.html. [visited on 21-04-2024].

Appendix A

Acronyms

ANN Artificial Neural Network

BiLSTM Bidirectional Long Short-Term Memory

CNN Convolution Neural Network

FFNN Feed-Forward Neural Network

LSTM Long Short Term Memory

ML Machine Learning

PCR Polymerase Chain Reaction

RBM Restricted Boltzmann Machines

ReLU Rectified Linear Unit

RF Random Forest

RNN Recurrent Neural Network

SELEX Systematic Evolution of Ligands by Exponential Enrichment

Appendix B

Contents of Attached Media

- `convolution` ..... the directory of Convolution Neural Network structures
  - `round_5_6` ..... the directory of notebooks using rounds 5 and 6
  - `round_5_8` ..... the directory of notebooks using rounds 5 and 8
  - `dupes-r78.ipynb` ..... Jupyter Notebook using rounds 7 and 8 with duplicate values
  - `unique-r67.ipynb` ..... Jupyter Notebook using rounds 6 and 7 with unique values
- `data` ..... the directory containing the data
  - `initial_data` ..... the directory containing unpreprocessed data
- `ffnn` ..... the directory of Feed-Forward Neural Networks
  - `round_5_6` ..... the directory of notebooks using rounds 5 and 6
  - `round_5_8` ..... the directory of notebooks using rounds 5 and 8
  - `round_5_7` ..... the directory of notebooks using rounds 5 and 7
  - `round_6_7` ..... the directory of notebooks using rounds 6 and 7
- `lstm` ..... the directory of BiLSTM
- `lstm` . the directory containing Jupyter Notebooks used for data statistics
- `randomforest` ..... the directory of Random Forest Jupyter Notebooks
- `single round` .. the directory containing the experimentation with single SELEX rounds
- `stats` ..... the directory of generated images
- `tests` ..... the directory of tests of CNN, FFNN and BiLSTM
- `text` ..... the directory containing this thesis
  - `BP.pdf` ..... the thesis text in PDF format
- `main.py` ..... python file
- `README.md` ..... description file