CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Predicting Antisense Oligonucleotide Thermodynamics using Deep Learning

Master's thesis in Computer science and engineering

Emil Wingårdh

Max Karlsson

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2023

Master's thesis 2023

Predicting Antisense Oligonucleotide Thermodynamics using Deep Learning

Emil Wingårdh, Max Karlsson

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2023

Predicting Antisense Oligonucleotide Thermodynamics using Deep Learning
Emil Wingårdh, Max Karlsson

Supervisor: Alexander Schliep, Department of Computer Science and Engineering
Supervisor: Shirin Tavara, Department of Computer Science and Engineering
Examiner: Devdatt Dubhashi, Department of Computer Science and Engineering

Master's Thesis 2023
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2023

Predicting Antisense Oligonucleotide Thermodynamics using Deep Learning
Emil Wingårdh, Max Karlsson
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Antisense oligonucleotides (ASOs) have emerged as a promising approach in medicine for the treatment of diseases associated with disrupted protein production. However, the lengthy drug discovery process poses a significant challenge. This project aimed to address this challenge by developing a baseline model to accurately predict the binding affinity of ASOs to mRNA, thereby expediting the drug development timeline.

Drawing upon the successful deep learning approach by Buterez in DNA hybridization [1], the baseline model was tailored to handle short ASO sequences. Various model architectures and features were explored, and recurrent neural networks (RNNs) based on Buterez's approach were employed as a benchmark for performance evaluation. The accuracy of the models was assessed based on their ability to predict the $\Delta\Delta G^{\circ}$, representing the difference in Gibbs free energy between the ASO sequence and a perfect match.

To optimize model performance, different input embeddings were tested, and architecture modifications were implemented. As a result, the final model achieved a high accuracy of approximately 96% with an error margin of$\pm$1.0.

By enabling accurate predictions of ASO binding affinity, this research contributes to streamlining the drug development process and holds promise for the advancement of precision medicine.

Keywords: ASO, Deep learning, RNN, LSTM.

Acknowledgements

We want to thank our examiner Devdatt Dubhashi and our supervisors Alexander Schliep and Shirin Tavara for their guidance and help during this masters thesis.

Emil Wingårdh, Max Karlsson, Gothenburg, 2023-06-16

Contents

List of Figures

List of Tables

1

Introduction

Numerous diseases, both common and rare, affect protein production within cells. Conditions like ALS, a neurodegenerative disease characterized by progressive motor neuron loss, and Alzheimer's disease are examples of diseases without a cure, where Antisense oligonucleotides (ASOs), hold promise for potential therapeutic interventions. Although several antisense oligonucleotide drugs, including Nusinersen, have received approval from the US Food and Drug Administration (FDA), the journey from discovery to an approved drug remains a lengthy process.

ASOs represent a novel class of drugs composed of short, chemically modified, single-stranded DNA molecules. In contrast to traditional small-molecule drugs that interact with proteins, ASOs function by preventing or decreasing the production of disease-causing proteins. This is achieved through a molecular mechanism of action involving hybridization between the patient's mRNA encoding the target protein and an ASO, leading to mRNA degradation. mRNA serves as an intermediary between the DNA code and the cellular machinery responsible for protein synthesis from amino acids.

The efficacy and safety of ASOs are primarily determined by the binding energy between the ASO and either the intended target mRNA or off-target mRNA. This binding energy is influenced by the nucleotide sequence of both the ASO and mRNA and is governed by the thermodynamics of the reaction.

Machine learning techniques, including random forest models, neural networks, and deep learning, have been extensively employed in drug discovery and other stages of drug development, leading to accelerated research and reduced risks in clinical trials [2]. However, certain areas of drug discovery remain relatively understudied, such as the field of ASOs.

Neural networks have garnered considerable attention and popularity due to their ability to learn from data, make predictions, and perform classifications without explicit programming. Through the process of training or learning, neural networks adapt their connections' strengths, enabling them to capture complex relationships within the data and improve their performance over time. By harnessing the power of neural networks, the drug discovery process could potentially be significantly shortened, provided the network's accuracy regarding drug safety and efficacy is sufficiently high. This can be done by predicting the binding energy of the ASOs to target mRNA sequences.

1.1 Aim

The aim of this project was to make significant contributions to ASO research focused on discovering potential cures for diseases that disrupt protein production within cells, particularly those currently lacking effective treatments. The primary objectives were to streamline the drug development process and enhance safety measures. This was achieved by developing a regression model capable of accurately predicting the efficacy of ASOs in targeting specific mRNA sequences. Multiple Recurrent Neural Network models were extensively investigated to identify the most suitable model for the task at hand. By accomplishing these goals, the project aimed to advance the field of ASO research, facilitating the discovery of novel therapeutic interventions for diseases with unmet medical needs.

1.1.1 Limitations

The limitation of this project is that there will not be any simulated antisense oligonucleotide strands present in the data. This project is a first step to see if deep learning is a viable approach for predicting binding energy between DNA and RNA sequences. Therefore we will only use simulated DNA strands in the dataset.

2

Background

2.1 Protein synthesis

Deoxyribonucleic acid, DNA, is stored inside the nucleus of the cell. Two strands of DNA are held together by nitrogenous bases. These bases are cytosine, guanine, adenine, and thymine. One of the strands contains the original code, which if followed correctly can assemble proteins outside of the nucleus. The other strand is the exact opposite of the first. This is the case since the bases only attach to complementary bases. Adenine only binds with thymine, and cytosine only binds with guanine [3]. This is referred to as the canonical Watson-Crick base pairing.

Messenger RNA, mRNA, stores a complementary template strand, which creates an exact copy of the original strand. The only difference in the new strand is that in the mRNA molecule, thymine is replaced with a new base called uracil. Inside the nucleus, the DNA code is not understood, only transcribed. The translation is done later on. The mRNA can be read by transfer RNA, tRNA, outside of the nucleus.

Ribosomes, a complex molecular machine found within cells that plays a crucial role in protein synthesis, are found in the cell cytoplasm or attached to the rough endoplasmic reticulum [4]. The ribosome matches the mRNA with the correct tRNA molecules that match the three-base code of mRNA. The tRNA carries an amino acid, which is dropped off if it's matched with mRNA. The messenger RNA goes through the ribosome as if on a conveyor belt. Each three-base code is matched with a new tRNA molecule, which drops off a new amino acid. A chain of amino acids begins to form, a polypeptide chain. When completed, the polypeptide chain is the final product, manufactured according to the original DNA code in the nucleus.

These polypeptide chains have the potential to fold and form proteins. When a polypeptide chain exceeds 40 amino acids, it undergoes tighter packing and is referred to as a protein. Proteins and polypeptides serve as the fundamental building blocks of various structures within the human body, including keratin in hair and fingernails, as well as hormones circulating in the bloodstream [5]. Figure 2.1 illustrates the different steps of the translation process.

The original code strand of the DNA duplex is sometimes referred to as the sense strand since it provides the genetic code for protein synthesis when read correctly. In contrast, the complementary strand is called the antisense strand, which serves as the template during transcription. Antisense oligonucleotides (ASOs) aim to

intercept the messenger RNA (mRNA) before it undergoes translation. ASOs can bind to the mRNA if it exhibits similarity to the original code sequence [6].

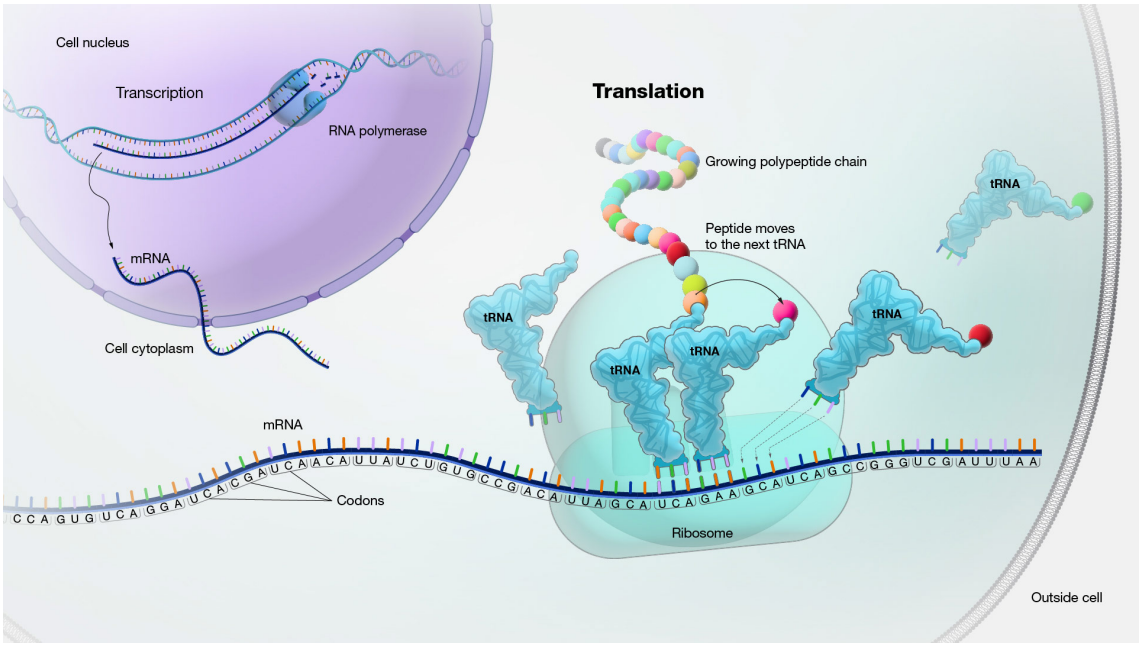


Figure 2.1: The translation process. This figure depicts the essential steps of translation, where messenger RNA (mRNA) is used to synthesize proteins. It includes initiation, elongation, and termination. The mRNA is read by ribosomes, and transfer RNAs (tRNAs) bring amino acids to build the polypeptide chain. The process concludes with the release of the protein. The image is taken from National Human Genome Research Institute.

2.2 Antisense oligonucleotides

Antisense oligonucleotides (ASOs) are synthetic strands of nucleic acids that are designed to target specific mRNA sequences. They are composed of short strands of nucleic acids, typically 18-25 bases in length, that are complementary to specific mRNA sequences. They bind to the mRNA through complementary base pairing, forming a double-stranded DNA duplex. This duplex can inhibit the translation of the mRNA into protein, either by triggering its degradation or by preventing its interaction with ribosomes during translation. ASOs can be designed to target any mRNA sequence, and can therefore be used to inhibit the expression of virtually any protein [7]. Figure 2.2 illustrates how the ASOs can bind to mRNA sequences and inhibit protein expression. ASOs can be used to target a wide range of diseases caused by aberrant protein production, including genetic disorders, viral infections, and cancer. In recent years, there has been significant progress in the development of ASOs as therapeutic agents, and several ASO-based drugs have been approved by regulatory agencies for clinical use.

ASOs have several advantages over other approaches to modulating gene expres-

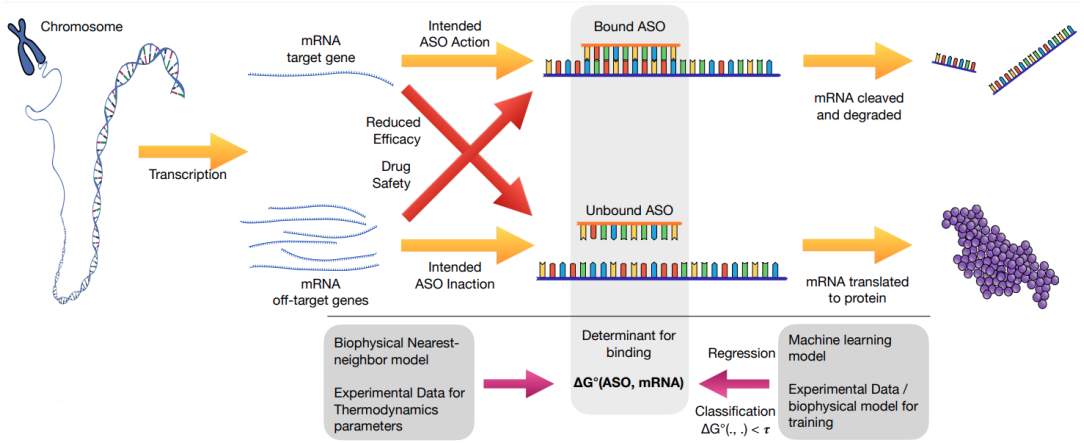


Figure 2.2: The transcription from the chromosome consists of mRNA, some of these mRNA sequences are the target genes that we want to cleave and degrade, while the vast majority of genes are off-target genes that the ASO should not bind to. Drug safety and efficacy are dependent on how well the ASO binds to target and off-target genes. The optimal case is that the ASO only binds to the target gene and no other genes, to guarantee safety and efficacy. Whether or not this happens depends on thermodynamics. The image is taken from Tavara et al. (Manuscript, 2022).

sion. For example, they are highly specific, meaning that they can selectively target individual genes without affecting other genes or cellular processes. Additionally, they can be delivered to specific tissues or cells in the body, making them useful for treating diseases that are caused by a specific genetic defect or that affect specific tissues. Finally, they can be used to modulate gene expression in a variety of ways, including silencing, splicing, and editing.

Despite these advantages, ASOs also have several limitations. For example, they can be degraded by nucleases in the bloodstream or within cells, limiting their effectiveness. Additionally, they can trigger off-target effects, meaning that they can inhibit the expression of unintended genes or induce unwanted immune responses. Finally, they can be difficult and expensive to produce and deliver.

2.3 Deep learning in DNA hybridization

Antisense technology has advanced, and ASOs in various chemical classes are beginning to provide important benefits to patients with a wide range of common and rare diseases. A large number of ASOs are in advanced clinical trials that will provide further results in the near future [8], but there is still some way to go before they are given out to the public.

There has been progress using deep learning to predict DNA hybridization [1], which is quite similar in nature to the problem at hand. DNA hybridization is the process in which two single-stranded DNA molecules bond together and form a double-stranded molecule. This process requires a DNA sequence to somewhat match a complement

in order to bond. The requirement is the same for ASOs and mRNA. For an ASO to bind to an mRNA they have to match somewhat so that the bond is strong enough. However, the ASO does not need to be a sub-string in order for the bond to be stable. With sequences matching, we mean that the strands are complementary and connected via hydrogen bonds in base pairs. In Watson-Crick base pairing for RNA, adenine (A) forms a pair with uracil (U), forming two hydrogen bonds, while guanine (G) forms a pair with cytosine (C), forming three hydrogen bonds [9].

Buterez [1] used two different machine learning approaches for DNA hybridization. The first approach was an image-based approach using a convolution neural network(CNN), where they used a 2D representation of the DNA as input for the network. The second approach was a sequence-based model, a recurrent neural network (RNN). The RNN used the sequences of the four amino acids A, C, G, and T as input and applied existing knowledge in natural language processing to process the data. The authors argue that the string representation of DNA sequences was a natural fit for the RNNs. They got satisfactory results for both approaches, but the RNNs were slightly better.

There has been some other research in the field of deep learning and how to use it to model genomics [10], but not anything with regression on ASOs. This makes deep learning with ASOs a fascinating topic.

2.4 Thermodynamics

In DNA and RNA, the thermodynamics of base pairing can be influenced by the identity of the nearest neighbour, which refers to the adjacent nucleotide in a DNA or RNA strand. The nearest neighbour effect refers to the impact that the identity of the nearest neighbour has on the stability of a base pair, and thus the overall stability of the double helix.

The nearest neighbour effect is due to the fact that base pairing involves the formation of hydrogen bonds between complementary bases. The number and strength of these hydrogen bonds depend on the identity of the bases involved and the surrounding chemical environment, which includes the nearest neighbour. Therefore, the identity of the nearest neighbour can affect the thermodynamics of base pairing by altering the strength and stability of the hydrogen bonds formed between the bases.

For example, the thermodynamic stability of a base pair between cytosine (C) and guanine (G) is stronger than that between adenine (A) and thymine (T) because the former involves three hydrogen bonds, while the latter involves only two. However, the presence of a strong nearest neighbour, such as a G or C, can further stabilize the A-T base pair by providing additional interactions and reducing the destabilizing effects of water molecules.

The nearest neighbour effect has been extensively studied and is incorporated into many thermodynamic models of DNA and RNA folding. One widely used model is the nearest-neighbour model developed by SantaLucia [11]

2.4.1 Gibbs Free Energy

Gibbs free energy (ΔG°) is a measure of the energy available for useful work in a system at a constant temperature and pressure. It combines the effects of heat energy (enthalpy) and the degree of disorder (entropy) to assess the spontaneity and stability of a chemical or physical process [12].

In a chemical reaction, ΔG° determines whether the reaction is favorable (exergonic, $\Delta G^\circ < 0$) or unfavorable (endergonic,$\Delta G^\circ > 0$). A negative ΔG° indicates a spontaneous reaction that releases energy, while a positive ΔG° requires an input of energy to occur. If ΔG° is zero, the reaction is at equilibrium.

Understanding Gibbs free energy helps determine the feasibility and direction of chemical reactions and phase transitions. It is important in chemistry, biochemistry, and thermodynamics to evaluate the stability and energy profiles of systems [12].

2.4.2 SantaLucia's Nearest neighbour model

The nearest neighbour model is a widely used approach for predicting the stability of DNA and RNA duplexes based on their sequence. It relies on experimentally determined thermodynamic parameters to calculate the Gibbs free energy change (ΔG°) associated with the formation of a given base pair, considering the adjacent nucleotides. This model assumes that the overall free energy change of duplex formation is the sum of the individual free energy changes of each base pair, including additional contributions from stacking interactions between adjacent base pairs.

The thermodynamic parameters used in the nearest neighbour model describe the free energy changes associated with the formation of all possible base pair combinations, as well as the influence of neighbouring base pairs on these energy changes. These parameters are derived from experimental measurements of free energy changes in specific sequences, and they are often reported as enthalpy change (ΔH), entropy change (ΔS), and free energy change at $37°C$ (ΔG°_{37}) for each base pair and nearest neighbour combination [11]. Table 2.1 displays all the different relations in energy between different neighbour pairs.

The nearest neighbour model is typically employed in conjunction with thermodynamic software programs, to predict the stability of DNA and RNA duplexes. This predictive capability is valuable for designing various nucleic acid-based tools, including oligonucleotide probes, primers, and other applications in molecular biology. For instance, the model's predictions are used in PCR (polymerase chain reaction) to determine optimal annealing temperatures, in hybridization-based assays to design specific probes for target detection, and in gene expression analysis to study RNA folding and stability.

Base pair	ΔH° (kcal/mol)	ΔS° (cal/(mol K))	ΔG°_{37} (kcal/mol)
AA/TT	-7.9	-22.2	-1.00
AC/TG	-8.4	-22.4	-1.30
AG/TC	-7.8	-21.0	-0.90
AT/TA	-7.2	-21.3	-0.88
CA/GT	-8.5	-22.7	-1.45
CC/GG	-8.0	-19.9	-1.84
CG/GC	-10.6	-27.2	-2.17
CT/GA	-7.8	-21.0	-0.90
GA/CT	-8.2	-22.2	-1.30
GC/CG	-8.5	-24.4	-1.44
GG/CC	-8.0	-19.9	-1.84
GT/CA	-8.4	-22.7	-1.45
TA/AT	-7.2	-21.3	-0.88
TC/AG	-8.2	-22.2	-1.30
TG/AC	-8.4	-22.4	-1.30
TT/AA	-7.9	-22.2	-1.00

Table 2.1: SantaLucia nearest-neighbour thermodynamic parameters for DNA Watson-Crick and homogeneous base pairs. The first column represents a neighbour pair and it's complementary bindings. For example: AA/TT, AA are the neighbours and TT the binding complement. The table is a recreation of a table from SantaLucia [11].

While the nearest neighbour model has proven to be a valuable tool, it is important to consider its limitations. One limitation arises from the model's inability to fully account for sequence context and long-range interactions that can influence duplex stability. Deviations between predicted and experimental values can occur due to unaccounted sequence motifs or structural features that significantly impact duplex stability. Additionally, the experimental conditions under which the thermodynamic parameters were measured may not always align with the specific experimental conditions of interest, requiring adjustments or corrections to improve predictions.

2.4.3 Deviations in duplex stability

The deviations observed in the nearest neighbour model for RNA duplex stability prediction have been the subject of extensive research. Various studies have highlighted the limitations of the model and the significant deviations between predicted and experimental values. The deviations occur due to several factors that are not adequately captured by the nearest neighbour model, leading to inaccurate predictions.

Mathews et al. [13] incorporated chemical modification constraints into a dynamic programming algorithm for RNA secondary structure prediction. They found that the nearest neighbour model often failed to predict the correct folding stability, especially in cases where non-canonical base pairs and tertiary interactions were

involved. These deviations were attributed to the inability of the nearest neighbour model to account for the influence of structural motifs and non-local interactions on duplex stability.

Another study by Huang et al. [14] focused on parameter optimization and sensitivity analysis for thermodynamics-based RNA secondary structure prediction. Their investigation revealed that the nearest neighbour model suffered from systematic errors and biases in predicting the thermodynamic properties of RNA duplexes. The deviations were attributed to oversimplifications in the model, such as the assumption of independent contributions from individual base pairs and the neglect of higher-order effects.

To address these deviations and improve the accuracy of duplex stability predictions, researchers have proposed several enhancements to the nearest neighbour model. Zhao et al. [15] developed a prediction method that considers both structural and compositional properties of RNA duplexes. Their approach incorporated additional features to better capture the contributions of non-local interactions and structural motifs. This improved model demonstrated reduced deviations compared to the traditional nearest neighbour model.

2.5 Artificial Neural Networks

Artificial neural networks, also simply known as neural networks, or NNs for short, are inspired by the biological brain. The brain has a lot of neurons which are nerve cells that allow the body to do everything the body does by sending messages all over it. Similarly, an NN is an interconnected group of nodes called artificial neurons that receives, processes, and sends signals to connected neurons. Generally, neurons are in different layers within the network, firstly an input layer and lastly an output layer, and between those, there might be a lot of other layers or possibly none, depending on the NN architecture. Figure 2.3 shows an example of a neural network. There are a lot of different types of NNs and the one used in this project is the recurrent neural network.

2.6 Recurrent Neural Networks

Recurrent neural networks, or RNNs for short, are neural networks that can deal with sequential data and hold knowledge about the past, which is something a traditional feed-forward neural network cannot do. The data points in the input of ordinary neural networks are supposed to be independent. Therefore if the data consists of sequences where one data point is dependent on other data points, there is a need to account for these dependencies in the neural network. RNNs can store states or information from previous input by using their "memory".

At each time step t, an RNN takes as input the current element of the sequence, $x(t)$, and the internal state of the network at the previous time step, $h(t-1)$. It then calculates a new internal state, $h(t)$, and an output, $y(t)$, based on these inputs

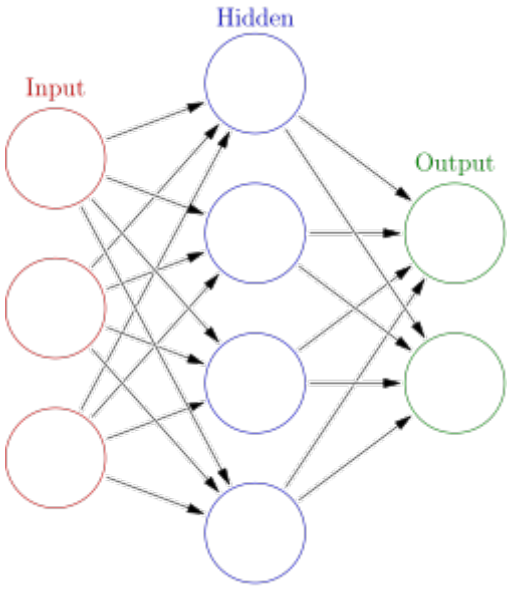


Figure 2.3: An example of an artificial neural network with 3 input neurons a hidden layer with 4 neurons and 2 output neurons. The image is taken from Wikipedia.

and the weights of the network. The new internal state and output are then passed on to the next time step as input [16]. Figure 2.4 shows a compact version of an RNN where one can see the loop principle.

Mathematically, an RNN can be represented as a sequence of computations that are applied at each time step. Let's say we have a sequence of length T, where each element $x(t)$ is a vector of size D. The computation at each time step can be written as:

$$h(t) = f(W_h[h(t-1), x(t)] + b_h)$$
$$y(t) = g(W_y[h(t)] + b_y)$$

where W_h and W_y are weight matrices, b_h and b_y are bias vectors, and f and g are activation functions. The square brackets denote the concatenation of vectors.

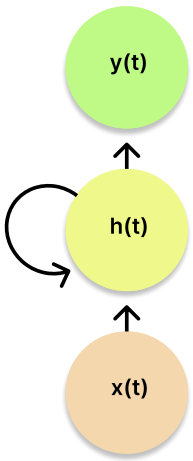


Figure 2.4: A compact RNN model.

The weight matrices and bias vectors are learned during training using backpropagation through time (BPTT), which is a variant of the backpropagation algorithm that takes into account the fact that the network has loops. However, an RNN cell is merely a concept. In practice, the cell is almost always an LSTM or GRU cell.

The main reason that RNNs are not used in practice is because of something called the vanishing gradient problem. The vanishing gradient problem prevents RNNs from bridging more than 5-10 time steps. The reason this happens is that the back-propagated error signals either increase or decrease every time step because of matrix multiplications that are involved in the calculation of the gradients [16].

Longer sequences cause small values in the matrix multiplication to decrease for every layer and vanish. Numbers above one causes the gradient to explode during the matrix multiplication, increasing the values of each weight by too much.

2.7 Long Short-Term Memory Network

The Long Short-Term Memory network, or LSTM network, is an RNN that is designed to handle the vanishing gradient problem. LSTM uses something called constant error carousels, or CECs, which is the central feature of LSTMs and is accessed by the gate units [17]. Through training, these gate units learn when to grant access to the CECs. The CECs enforce a constant error flow and are where we get our short-term memory storage. This constant error flow is how it overcomes the vanishing gradient problem in RNNs. Because of that, it can be used to create larger recurrent networks that can be used for difficult sequence problems. An LSTM network consists of memory blocks instead of traditional neurons. Each block has special components that make it smarter than both the classical neuron and memory for RNNs. The blocks contain gates that manage the block's state and output. There are three types of gates in a block:

- **Forget Gate:** decides what information should be thrown away
- **Input Gate:** decides which values from the input should be put in the memory state
- **Output Gate:** decides what to output based on input and memory state

LSTM networks have been shown to be effective for a wide range of tasks, including natural language processing, speech recognition, and time-series prediction. They have been used in numerous applications, from machine translation to stock market forecasting.

2.8 ASHA

ASHA (Asynchronous Successive Halving Algorithm) is a state-of-the-art algorithm used for efficient hyperparameter optimization and model selection. It is designed to minimize the computational resources required while maximizing the performance of the selected model. ASHA follows a progressive elimination strategy, where models with poor performance are discarded early in the optimization process.

The algorithm operates asynchronously, allowing multiple evaluations to occur simultaneously. It starts by training a set of candidate models for a fixed resource budget, such as training time or epochs. Then, the best-performing models are selected based on their intermediate validation scores. The selected models are then allocated additional resources, and the process is repeated iteratively, progressively narrowing down the pool of models until only the best model remains.

ASHA has shown significant improvements in terms of efficiency and effectiveness compared to traditional hyperparameter optimization algorithms. It balances the

exploration and exploitation of the search space, allowing for faster convergence and better utilization of computational resources**asha**.

2.9 Measurements in Genomics

Throughout the project, different measurements were used and explored. Different measurements have different characteristics that provide different overviews of the similarities and dissimilarities in the sequences. This section explains these different metrics.

2.9.1 Hamming Distance

The Hamming distance is a measure of the difference between two sequences of equal length. It quantifies the number of positions at which the corresponding elements in the sequences are different. In other words, it calculates the minimum number of substitutions needed to change one sequence into the other.

Hamming distance is widely used in various fields, including computer science, information theory, and genetics. In genetics, it is often employed to compare DNA or RNA sequences and assess their similarity or evolutionary relationships [18].

This can be done easily with mRNA sequences, where the strings contain the letters A,C,G,U, which represent the nitrogenous bases adenine, cytosine, guanine, and uracil respectively.

2.9.2 Q-grams

The q-gram distance is a measure of similarity between two strings based on the number of shared subsequences of length q, known as q-grams. It counts the number of q-grams that appear in both strings and calculates the difference between them.

Q-gram distance is commonly used in various applications such as plagiarism detection, spell checking, and DNA sequence alignment [19]. For small ASO sequences, a q of 2 and 3 should suffice.

2.9.3 GC-Content

The GC content is a widely used measurement in genomics and molecular biology. It refers to the proportion of nitrogenous bases in a DNA or RNA sequence that are either guanine (G) or cytosine (C). GC content is often expressed as a percentage and provides valuable information about the sequence characteristics and stability [20].

The GC content of a sequence can influence its structural properties, such as melting temperature and stability. Sequences with higher GC content tend to have higher melting temperatures due to the stronger bonding between G-C base pairs compared to adenine (A) and thymine (T) base pairs. The GC content can also affect the secondary structure formation, protein binding, and gene expression regulation.

Measuring the GC content of a sequence is relatively straightforward. It involves calculating the percentage of G and C bases out of the total bases in the sequence. This information can be useful for various applications, including primer design, gene expression analysis, and evolutionary studies [20].

2.9.4 Longest Common Factor

The longest common factor (LCF) is a concept used in genomics to measure the similarity or commonality between two DNA or RNA sequences. It refers to the length of the longest identical substring that can be found in both sequences. In genomic settings, the LCF has various applications and can provide insights into evolutionary relationships, sequence conservation, and functional regions.

By identifying the LCF between different sequences, researchers can uncover conserved regions that are critical for maintaining the structure or function of the genetic material. These conserved regions often indicate important regulatory elements, binding sites, or functional domains [21].

3

Methods

The goal of the project was to adapt an initial deep learning approach to the setting of the short ASO sequences, explore different model architectures and features, and provide a benchmark, using RNNs as a baseline model as they had shown optimistic results on similar problems.

3.1 Data collection

Thermodynamics data for ASOs is sparse. In contrast, the thermodynamics of DNA-DNA hybridization for naturally occurring DNA is quite well understood. Reasonable amounts of experimental data are available, but not in high enough quantity for deep learning. More importantly, accurate biophysical models such as the ViennaRNA package for Python can provide arbitrary amounts of simulated data. This made it possible to use scripts in order to collect data sets that could be used in order to train and test a regression model.

The data was generated in the form {(binding site sequence, ASO sequence, $\Delta\Delta G^{\circ}$)}. $\Delta\Delta G^{\circ}$ is the difference in Gibbs free energy between the Watson-Crick complement and a binding site with mismatches. The two sequences can be summarized by considering the Hamming distance, q-gram distance, and some more distance functions. The focus here is on Hamming distance and q-gram distance. This makes it so that the data generated and to be analyzed changes into the form {(D(ASO sequence, binding site sequence), $\Delta\Delta G^{\circ}$)} for the chosen distance function D. This transformation is needed for the model to work with linear layers. Also, a requirement that has to be met when generating the data is that the sequences need to be disjoint in order not to get inaccurate results.

Dataset	Total lines	Binding sites	ASO sequences per binding site
Small dataset	50 000	1 000	50
Medium dataset	500 000	10 000	50
Large dataset	5 000 000	100 000	50

Table 3.1: The different datasets created and used.

Multiple different data files of different sizes were created as seen in table 3.1. Every dataset was created similarly inside of the 50 different ASO sequences per binding

site, and that is, there were 10 randomized oligonucleotides of every Hamming distance from 1 to 5 compared with the binding sites. However, something not included in the calculations of the number of lines and ASO sequences per binding site is the exact matches.

The exact matches are the ones with 0 Hamming distance, and there can only be one of those per binding site, so technically there are 51 ASO sequences per binding site and thereby 51 000, 510 000, and 5 100 000 total lines in the datasets respectively. The primarily used dataset for training was the medium dataset file. The small dataset was used for fast debugging and comparing concepts, while the large dataset was used to see if there was a difference in performance on a larger dataset. The medium dataset consists of 52% A and T content, and 48% G and C content which is considered balanced.

When the data sets had been created, they were shuffled around and split into training, validation, and test datasets with a 60%, 20%, 20% split. This was seen as a reasonable split since we had quite a lot of data.

3.2 Encoding

To ensure the proper functioning of the RNN models, it was necessary to encode the RNA sequences present in the data file. This encoding process was essential for the effective utilization of the embeddings layer in a PyTorch LSTM network. The embeddings layer plays a critical role in mapping discrete tokens (e.g., words or nucleotides) to a continuous vector space known as the embedding space. Within this space, tokens are assigned unique vectors, and similar tokens are positioned closer together.

During training, the embedding layer learns the optimal vector representation for each token by optimizing a task-specific loss function. This enables the network to capture meaningful semantic relationships between tokens, leading to improved performance on various downstream tasks, such as text classification or language generation.

The encoding procedure varied among different models, necessitating distinct embedding spaces. Nevertheless, they all adhered to the same underlying principle. In the case of a standard RNA sequence, such as in the baseline model, each nitrogen base was encoded as a discrete token.

To handle more complex encodings, particularly those incorporating the SantaLucia neighbour model, see Section 2.4.2, tokens were assigned to represent all possible combinations of neighbouring nucleotides. In the Neighbour embeddings model, which considers the nearest neighbour, a total of 4^2 distinct tokens were used to represent the various combinations of neighbouring nucleotides. This is because there are four nitrogenous bases and each base has four possible neighbour pairings, resulting in 4^2 possible combinations.

3.3 Models

The primary objective of the project was to develop a regression model capable of capturing the relationship between binding variables and accurately predicting the binding affinity of ASOs to mRNA. To achieve this, various models were constructed and trained using previously generated datasets. The ultimate goal was to enable the identification and evaluation of specific oligonucleotides against numerous potential binding sites.

The intention was to identify oligonucleotides that exhibit appropriate binding characteristics, whereby they bind optimally to their intended target sites without excessively binding to other unintended sites. Therefore, the models aimed to compute the binding energies between a given oligonucleotide and multiple sites, allowing for efficient assessment and comparison.

In designing the initial baseline model, inspiration was drawn from Buterez's paper [1], which employed distinct machine learning approaches. The baseline model adopted a similar architecture to their RNN model, as it demonstrated slightly better performance. Given the sequence-like nature of the data, representing the RNA sequences as strings and leveraging natural language processing techniques appeared to be the most suitable approach for establishing a robust baseline model for the project's objectives.

3.3.1 Baseline

The initial step in the project involved establishing a baseline model as soon as possible. This baseline model took the form of a relatively simple RNN architecture with an input format comprising the binding site sequence, ASO sequence, and $\Delta\Delta G^{\circ}$ (a measure of binding energy). The two RNA sequences were embedded together to enable the RNN to discern differences between various sequences. This embedding process involved translating the amino acids into a numerical representation and incorporating tokens to indicate the start, midpoint, and end of the sequence. The baseline model served as a starting point for subsequent improvements.

The architecture of the model drew inspiration from a previous RNN model [1], with necessary adaptations to suit the dataset. The modified model consisted of a 2-layered, bidirectional LSTM with a hidden size of 100 and a dropout rate of 20%. Additionally, a fully connected layer with an input size of 200 and an output size of 1 served as the final layer of the model. Figure 3.1 depicts the architecture of the baseline model.

To determine the appropriate learning rate, PyTorch Lightning's Learning Rate Finder was utilized. This mechanism rapidly ran the model with incrementally increasing learning rates, suggesting a suitable starting rate based on observed loss values. Early stopping was implemented to address overfitting, with a stopping criterion of no significant improvement after 3 epochs.

Furthermore, a learning rate scheduler was incorporated into the model. The selected scheduler was ReduceLROnPlateau, which reduces the learning rate by a

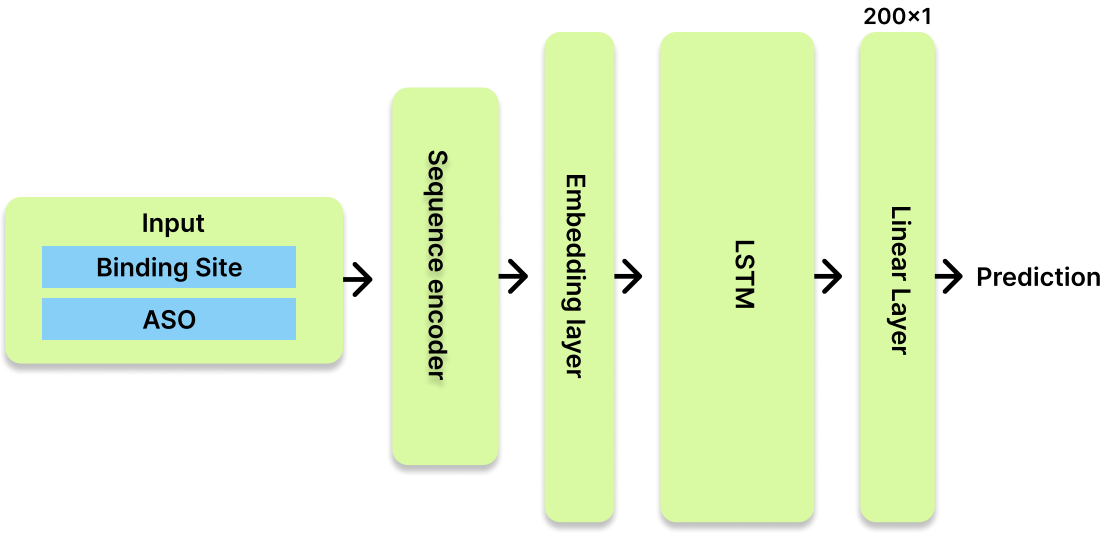


Figure 3.1: The architecture of the baseline model.

user-defined factor after a certain number of epochs without improvements in the training loss. A patience of 5 epochs and a gamma value of 0.2 were chosen, indicating that if the training loss did not improve for 5 consecutive epochs, the learning rate would be divided by 5. These parameter values were carefully tested during training iterations and found to be suitable, allowing for multiple learning rate reductions throughout a complete training cycle.

3.3.2 Neighbour Embedding

The next model was based on the baseline and was built upon from there. As established in Section 2.4, the free energy depends on the structure of the RNA sequence. Depending on the nearest neighbour of a base, the energy required to bind to it changes. Therefore it would make sense to not use a linear encoding for the RNA sequences, and use a nonlinear encoding instead. In the Neighbour Embedding model, the SantaLucia nearest neighbour coefficients were used to try to make a more realistic embedding to improve performance.

Other than the encoding of the sequences, there were no changes to this particular model compared to the Baseline. This was decided in order to compare the two different encodings to each other.

3.3.3 Improved models

Upon evaluating the performance of a more realistic embedding, notable differences in performance persisted between the baseline and neighbour models. Recognizing the potential for improvement across all models, several architectural modifications were explored and tested.

Initially, various architectural changes were made to the LSTM part of the network. However, as the experimentation progressed, greater emphasis was placed on

the linear part of the network, which follows the processing of data by the LSTM network.

To enhance the models, a novel approach involved incorporating multiple linear layers instead of a single layer. A range of three, four, and five additional layers were implemented and thoroughly experimented with. Moreover, different layer sizes were tested to compare their impact on performance. Figure 3.2 illustrates the architecture for the model with three linear layers.

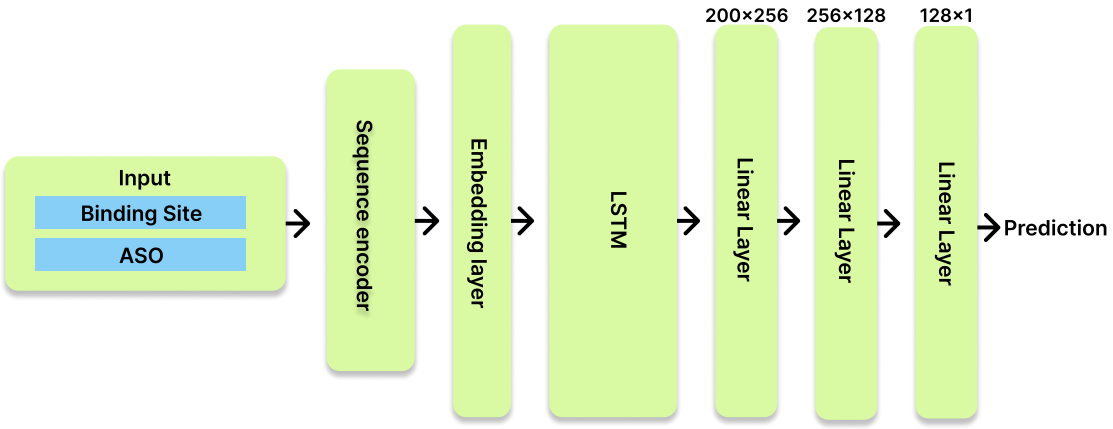


Figure 3.2: The architecture of the improved model with additional linear layers. The numbers above the linear layers are the dimensions of the layer.

3.4 Analysis

In order to measure performance, one must define the features that are used in the models to describe similarities and dissimilarities in the sequences. Part of the process was to find out which possible features work best for the models. This was explored by testing a few different features that showed promise and looking at the results. The first such feature was to simply compare an encoded combination of the strings of the binding sites and the oligonucleotides with the $\Delta\Delta G^{\circ}$. Another feature that was used is the Hamming distance, which is a commonly used measure for RNA and DNA, as explained in Section 2.9.1.

To measure sequence dissimilarity, a number of features were used. The Hamming distance is usually good, but the q-gram distance is also a potentially useful feature and was also examined. In these small ASO sequences, a q of 2 and 3 were tested. To measure the q-gram distance we first measure the occurrences of q-grams in the mRNA sequence and the ASO sequence. As explained in Section 2.9.2, the q-gram distance is given by the difference in occurrences of a certain q which in this case is 2 or 3. Here, adding weights to different 2- and 3-gram strings was also needed, since different mRNA sequences have different binding energies.

Another initial feature used is the difference in Gibbs free energy between the Watson-Crick complement and a binding site with mismatches, $\Delta\Delta G^{\circ}$. Gibbs free energy (ΔG°) is a thermodynamic potential and is explained further in Section 2.4.1.

The performance of the different models was measured by the accuracy of the models with regard to the prediction of off-target bindings. Since the models predict the $\Delta\Delta G^{\circ}$, a float value, an interval for the accuracy was introduced. We defined three different accuracy intervals; high, medium, and broad. The high accuracy interval was ± 0.5. The medium accuracy interval was larger, namely ± 1.0. The broad accuracy had the largest interval of ± 2.0. These margins of error were deemed reasonable intervals since there is some deviation between the nearest neighbour model that is used when calculating the energies, and reality.

There were some concerns that the model wouldn't be able to as accurately estimate energies in a certain critical range, which was the range of 4.0 to 10.0. This range was seen as critical since this was the range in which sequences could bind or not bind depending on various factors in sequence alignments. Thus, in addition to these aforementioned accuracies, a "critical accuracy" was added. This special accuracy used the same interval as the broad accuracy but only accounted for predictions within the critical range. These accuracies were compared to the baselines to further analyze the performance.

Accuracy is very important since otherwise there might arise problems when using the program for real-life applications. A reason for this being very important is that we need to see if an ASO binds off-target since it then could cause harm to the subject if used. An ASO may bind off-target due to other RNAs sequence similarity or completely RNA-independent, and it may degrade the mRNA and induce toxicity [22]. However, shorter RNA transcripts are less likely to have mismatches.

The accuracy of $\Delta\Delta G^{\circ}$ can be measured relative to many different features. It will mainly be measured against the time it takes to train, however, it is still important that it performs well relative to other features. Those being ΔG°, Difference in GC, Longest common factor, and Hamming distance. This is done because GC pairs are stronger than other pairings and can thereby influence sequence characteristics, as mentioned in Section 2.9.3. The Longest common factor, or LCF, is used since similar sequences should be able to bind easier, and could thus help with predicting the thermodynamics.

3.4.1 Hypertuning

We utilized the Python package Ray Tune for hypertuning our parameters. It worked well with our setup and was easy to set up. We focused on tuning the different linear layers outside of the RNN, as we observed various results in early training depending on the sizes of these layers. Additionally, the batch size, dropout in the RNN and linear layers, and the learning rate were tuned. These parameters were considered crucial for performance improvement. Ray Tune took a config of different parameter choices as input and used an asynchronous successive halving algorithm, ASHA, to prevent bad parameter combinations from running for too long (see Section 2.8). This allowed us to explore numerous combinations without significantly increasing the runtime. Each time the program ran with Ray Tune enabled, we could specify the desired number of combinations, and it would randomly select and compare them. Finally, it would identify the best parameter combination for the model.

Parameter				
L1 size	128	256	512	1024
L2 size	256	512	1024	2048
L3 size	128	256	512	1024
L4 size	64	128	256	512
lr	0.001	0.002	0.0025	0.003
Batch size	64	128	256	512
dropout	0.1	0.2	0.3	

Table 3.2: Table for the parameters used while hypertuning which has $4^6 * 3 = 12288$ combinations.Every row depicts the different parameter choices for that parameter that the algorithm may choose.

4

Results

Here are the results of the different models in terms of accuracy when calculating the $\Delta\Delta G^{\circ}$ as well as some end results of different features shown in scatterplots which will be explained in section 4.1.4. The results shown here are all from the shuffled dataset named Medium dataset which has 510 000 total lines, i.e. 10 000 binding sites and 50 oligo sequences per binding site plus 1 extra oligo sequence per binding site with the exact match. All the models, except for the ones in section 4.3, were run with the same parameters, with a batch size of 128, a learning rate of 0.0025, a dropout value of 0.2, and a hidden size of 100. The runs with the same amount of layers also had the same amount and structure of neurons. 1 linear layer had neurons of 2*Hidden size down to 1. 3 linear layers had neurons of 2*Hidden size to 256 to 128 to 1. 5 layers had neurons of 2*Hidden size to 1024 to 512 to 256 to 128 to 1.

Multiple runs were made for every different model and version, all of which were run 5 times, except for the final hypertuned version which was run 10 times. Multiple runs are made in order to see an average and be able to compare them to see how stable of a result the model can achieve. In the graphs, the dotted lines show specific runs of the model while the clear line is the average of those dotted lines. The maximum value of the average line in every graph is highlighted. However, in the scatterplots, there is no average dot. All of the different colors of the dots in the scatterplots are for the different runs.

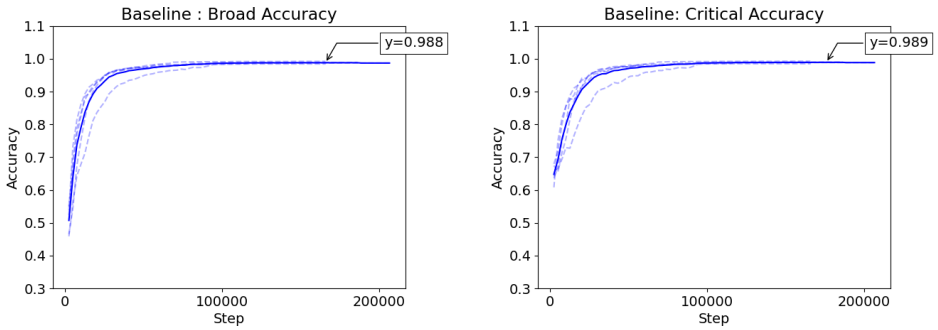
In the graphs with lines, we have the accuracy relative to the time steps. This is so that it can be seen how fast the model trains and in order to compare the different runs throughout the training. These are made from the validation data, while the feature analysis with scatter plots is from the test data.

4.1 Initial models

Here are the initial models, the baseline model, the neighbour embedding model, a comparison result of them, and a feature analysis. These models used 1 layer.

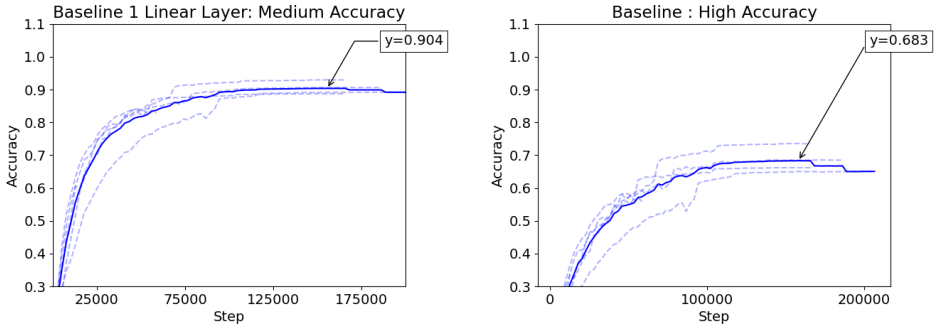
4.1.1 Baseline Model

As seen in Figure 4.1, the broad accuracy — which was deemed a reasonable accuracy for the float values of the $\Delta\Delta G^{\circ}$ — quickly gains a steady accuracy of around 98.8%.



(a) ±2.0 accuracy

(b) ±2.0 accuracy for the (binding site, oligonucleotide) matches with 4-10 $\Delta\Delta G^{\circ}$



(c) ±1.0 accuracy

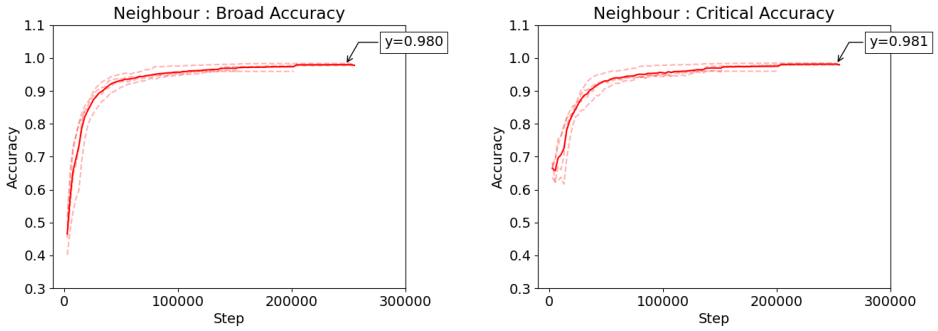
(d) ±0.5 accuracy

Figure 4.1: The different accuracies plotted for the baseline model.

The critical accuracy looks about the same as the broad accuracy, very hard to distinguish them in the plots but from the values, it can be seen that it ends at around 0.1% higher than the broad accuracy. When it gets that close to 100% a seemingly small increase of 0.1% is actually quite big and this tells us that in the end, the model performs better on the values that are the most important for our purpose.

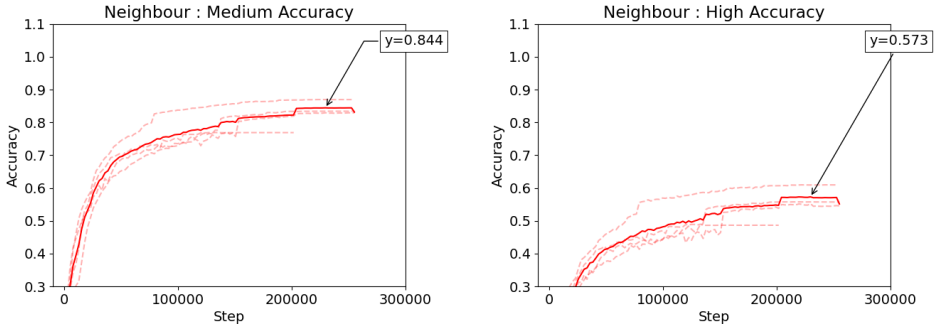
The medium accuracy gains an accuracy of around 90% which is still quite good but still has room for improvement. The High accuracy gains an accuracy of around 68% which makes it right about every 2 out of 3 times which is not a very high chance. However, having to go down all the way to a ±0.5 accuracy is of course better and more desirable if a large value is achieved, but not necessary for the purposes of any of these models. Because the energy calculation already has a margin of error, it can also be conflicting with that. As can also be seen in Figure 4.1, the medium and the high accuracies are a little less steady than the broad as the different runs look a bit different from each other but they still tend to reach around the same result in the end.

4.1.2 Neighbour Embedding



(a) ±2.0 accuracy

(b) ±2.0 accuracy for the (binding site, oligonucleotide) matches with 4-10 $\Delta\Delta G^{\circ}$



(c) ±1.0 accuracy

(d) ±0.5 accuracy

Figure 4.2: The different accuracies plotted for the nearest neighbour embedding model. The maximum value of the average line is highlighted.

As seen in Figure 4.2, it follows roughly the same patterns as in the baseline but at a slightly lower value. However, in these figures, we see that the model tends to be more coherent in the beginning and then receives a bit different end results with every run, as opposed to the baseline which had a coherent end but a more varied beginning. The broad accuracy gets about 98% with about the same result for critical accuracy, being 0.1% higher. The medium and high accuracy ends up being a bit varied every run with an accuracy of around 84% and 57% respectively.

4.1.3 Comparison

Figure 4.3 shows more clearly that the Baseline model is better than the Neighbour model on every metric. It's not a huge difference, but it's a few % more on every accuracy metric.

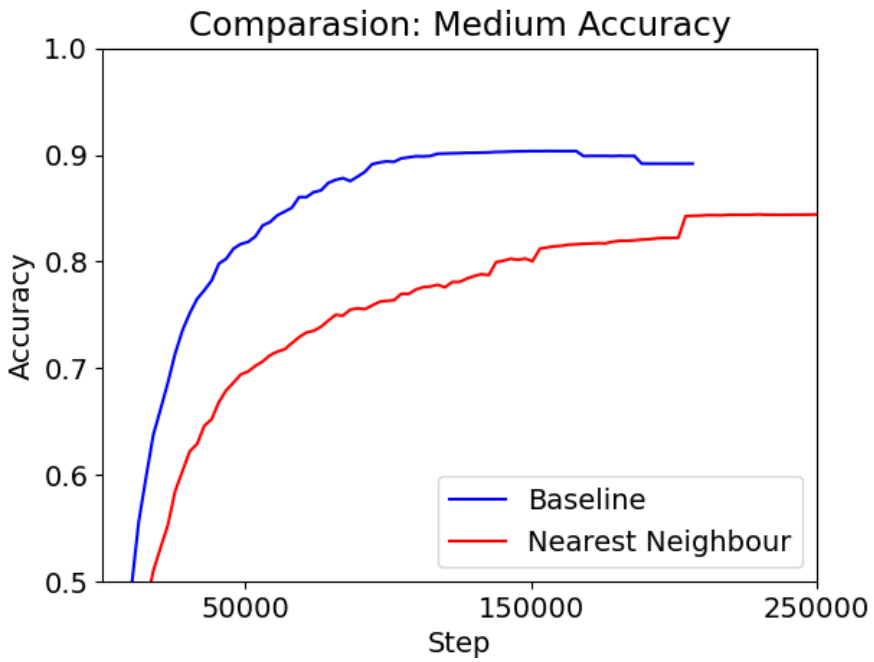


Figure 4.3: The ±1.0 accuracy plotted for both models for comparison.

4.1.4 Feature analysis

The feature graphs shown in Figure 4.4 - 4.5 illustrate the accuracy values achieved at the end of training. Each graph represents 5 runs of the model, with different colored dots indicating individual runs.

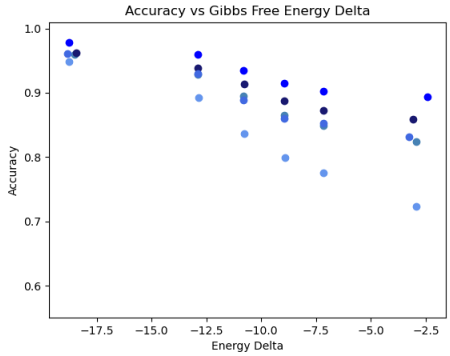
Each sub-graph has a distinct x-axis. In cases where the axis values are continuous, they have been processed and grouped together to provide more discernible values. This applies to sub-graphs (a) and (b).

Sub-figure (a) represents the accuracy for different ΔG° (Gibbs free energy) values. A smaller negative value of ΔG° indicates less similarity between sequences.

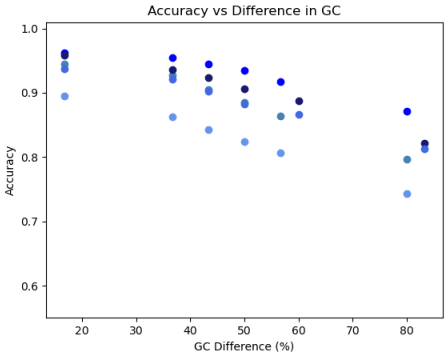
Sub-figure (b) illustrates the difference in GC content, measured as the variation in the percentage of GC pairs within the RNA sequence. GC pairs are stronger than other pairings and can influence sequence characteristics. A full 15-bit sequence of G or C would give the value 100.0, and a full sequence of A or T would give the value 0.0 in the graphs' x-axis.

In sub-figure (c), the Longest Common Factor (LCF) is depicted with an inverted relationship. A value of 0 on the graph represents the highest possible LCF, as the data was calculated using the formula $15 - \text{LCF}$.

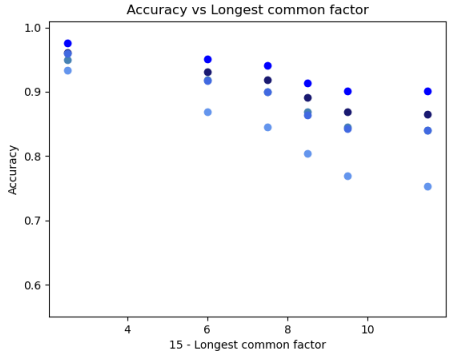
Sub-figure (d) presents the hamming distance, which measures the dissimilarity between sequences. A higher hamming distance indicates a greater number of changes required to make the sequences identical. The measured distances range from 0 to 5, as the sequences are 15 bits in length and it is already quite clear if it would bind any good or not with higher distances so those are not needed as much.



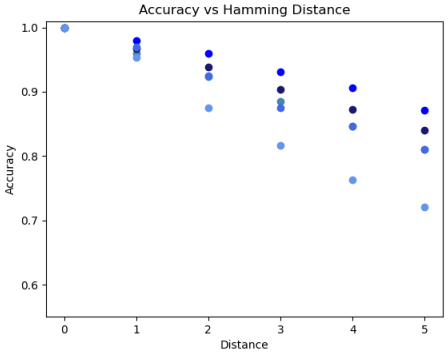
(a) ΔG° accuracy



(b) Accuracy for difference in GC

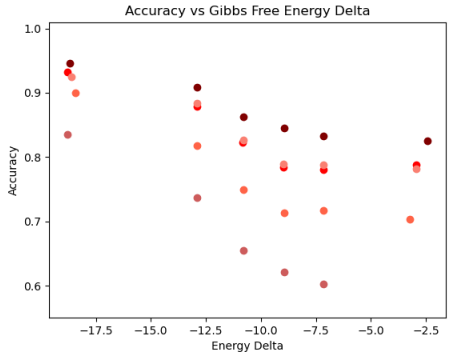


(c) Accuracy for the longest common factor

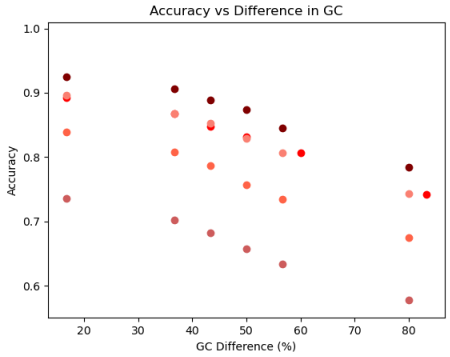


(d) Accuracy for Hamming Distance

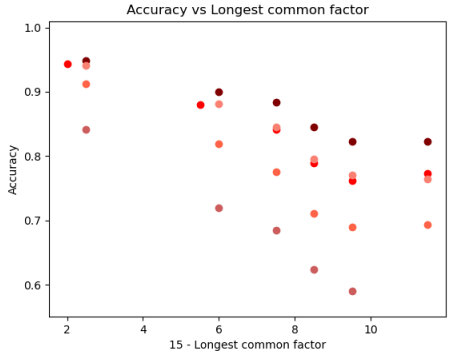
Figure 4.4: The different accuracies for the different features for the medium accuracy, that is ± 1.0 accuracy. All runs were run on the Baseline model with only 1 layer like in Figure 4.1.



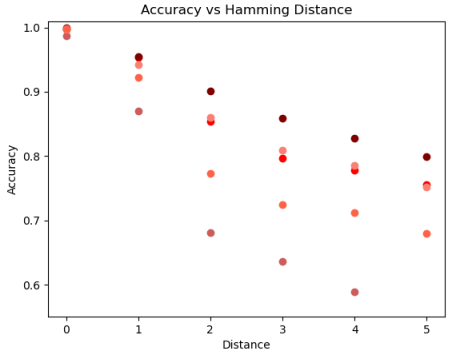
(a) ΔG° accuracy



(b) Accuracy for difference in GC



(c) Accuracy for the longest common factor



(d) Accuracy for Hamming Distance

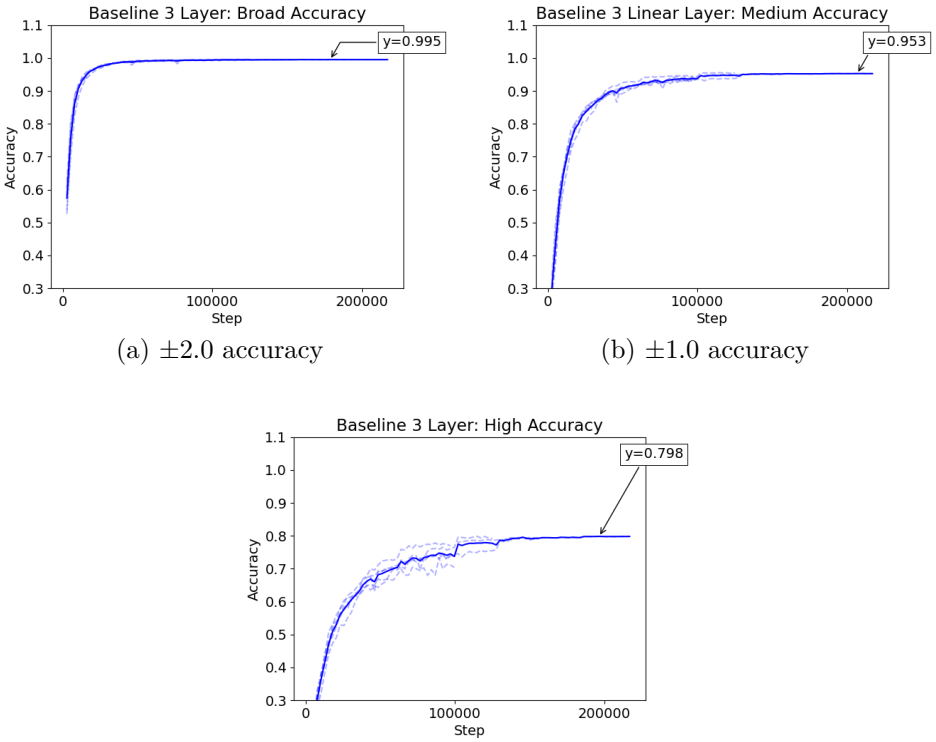
Figure 4.5: The different accuracies for the different features with the medium accuracy, that is ±1.0 accuracy. All runs were run on the neighbour embedding model with only 1 layer like in Figure 4.2.

As seen in the base model Figure 4.4 and the neighbour embedding model Figure 4.5, there is almost always a better accuracy with a lower hamming distance. The less GC content, the better the accuracy. The higher the negative value of Gibbs free energy, the better the accuracy. The longer the LCF, the better the accuracy for the most part. The accuracy is almost always better for the base model compared to the neighbour embedding. Unfortunately as seen, there can be outlier runs that get quite a bit different accuracy than the others. This means that the neighbour embedding sometimes gets better runs than the base model, however, looking at the average it is clear that the base model gets better results.

4.2 Model improvements

This section presents the results of the different model improvements that were made to the two main models: the baseline and nearest neighbour model. The biggest change was that the complexity of the architecture was changed. Improving the complexity of the linear part of the model was a logical approach in the search for more accuracy. From here, the critical accuracy will no longer be present because as could be seen in Figure 4.1 and 4.2, the accuracy was about the same as the broad which is the case for all of the model versions.

4.2.1 Baseline



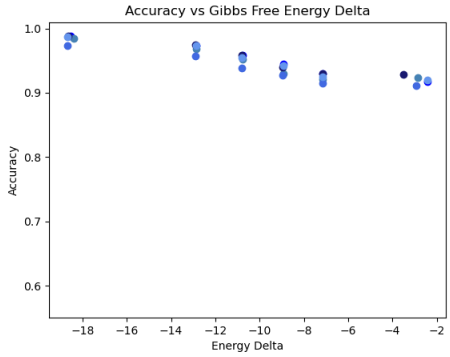
(a) ±2.0 accuracy

(b) ±1.0 accuracy

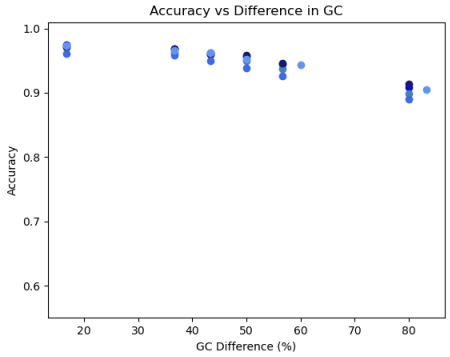
(c) ±0.5 accuracy

Figure 4.6: The different accuracies for the baseline model with 3 layers.

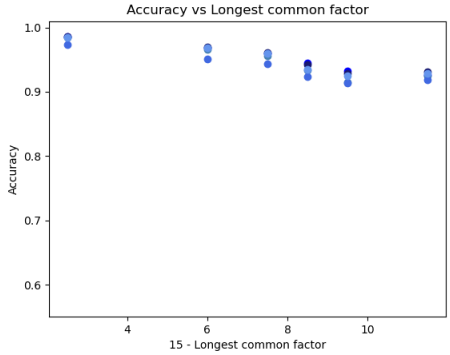
Figure 4.6 shows the different accuracy plots when using three linear layers after the LSTM network. There are accuracy increases compared to the normal singular layer model in all ranges, but the most significant one is the 5% accuracy increase in the medium range. This means that the medium accuracy is up to 95%.



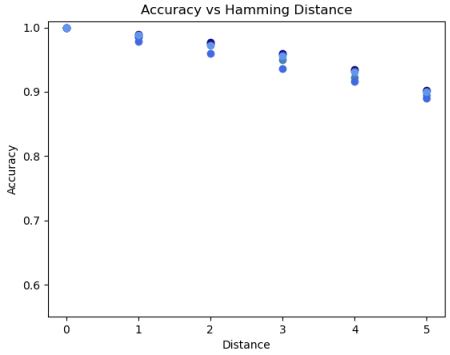
(a) ΔG° accuracy



(b) Accuracy for difference in GC



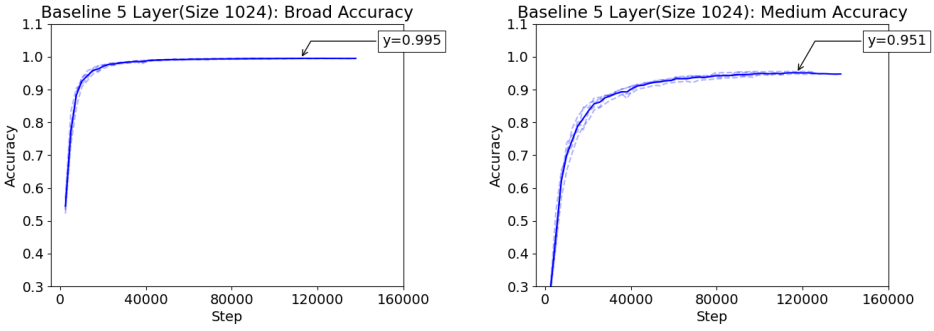
(c) Accuracy for the longest common factor



(d) Accuracy for Hamming Distance

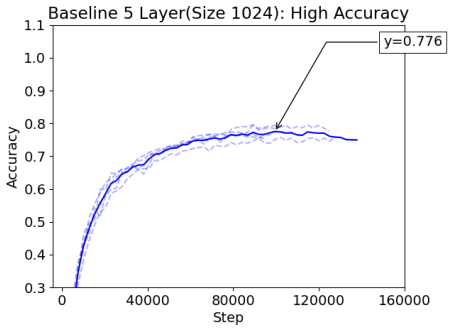
Figure 4.7: The different accuracies for the different features for the medium accuracy, that is ± 1.0 accuracy. All runs were run on the Baseline model with 3 layers.

Figure 4.7 shows the different features of the new improved 3-layer model. If one looks at the former feature plot, namely Figure 4.4 it's clear that the variance between runs has decreased, as the difference between the highest accuracy for any specific value and the lowest is a lot less. Overall the accuracy is slightly increased for every feature as well.



(a) ±2.0 accuracy

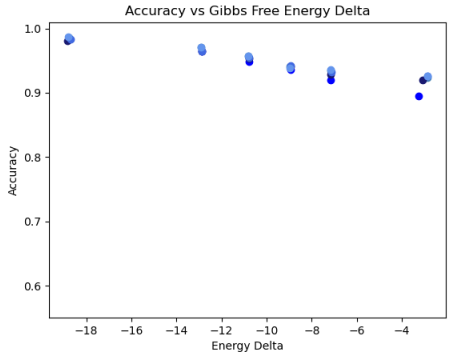
(b) ±1.0 accuracy



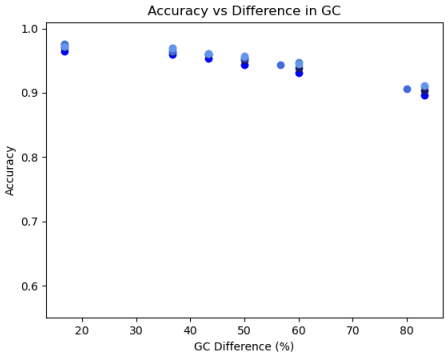
(c) ±0.5 accuracy

Figure 4.8: The different accuracies for the baseline model with 5 layers. The highest value of the average plot is highlighted.

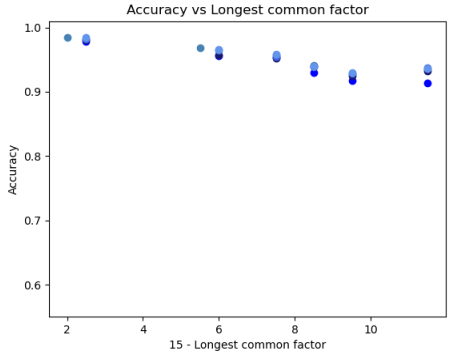
When the number of layers is increased to five, there is not a significant increase in performance, as can be seen in Figure 4.8. It is very similar in regard to accuracy as Figure 4.6.



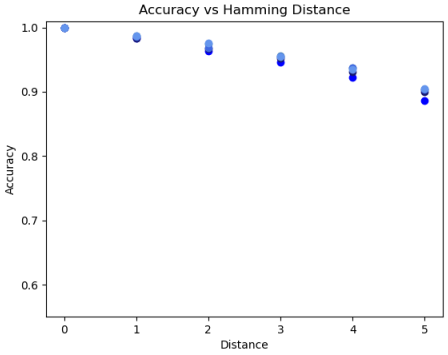
(a) ΔG° accuracy



(b) Accuracy for difference in GC



(c) Accuracy for the longest common factor

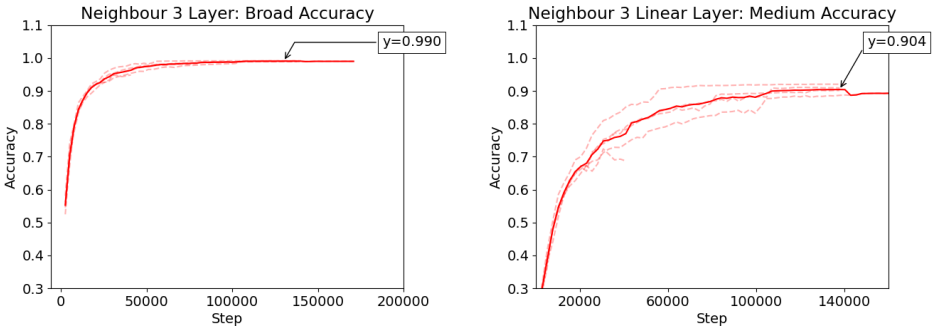


(d) Accuracy for Hamming Distance

Figure 4.9: The different accuracies for the different features for the medium accuracy, that is ± 1.0 accuracy. All runs were run on the Baseline model with 5 layers.

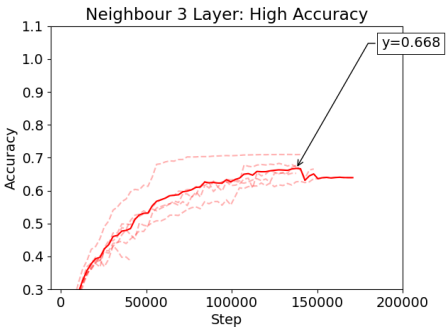
The feature scatter plots of the five-layer baseline model is also very similar to the one of the tree layer model, as can be seen by comparing Figure 4.7 and Figure 4.9.

4.2.2 Nearest Neighbour



(a) ±2.0 accuracy

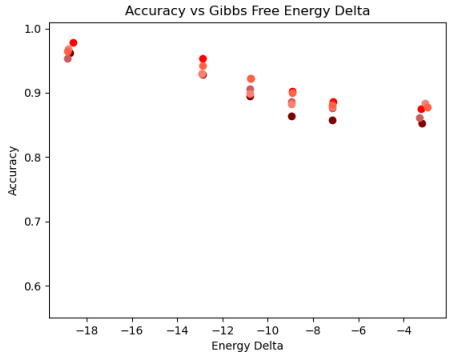
(b) ±1.0 accuracy



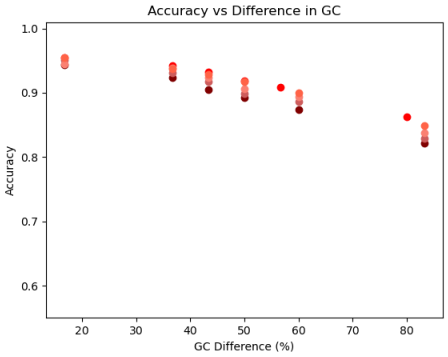
(c) ±0.5 accuracy

Figure 4.10: The different accuracies for the neighbour embedding model with 3 layers.

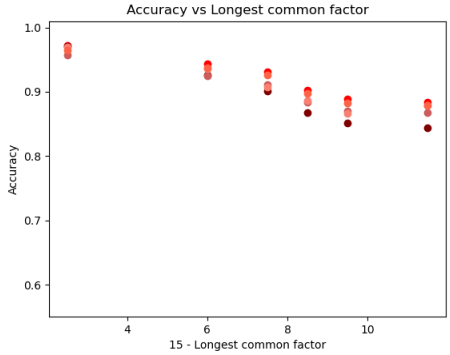
Figure 4.10 shows that the Nearest Neighbour model gains a big advantage in accuracy when increasing the number of layers in the linear part of the model, as can be seen, if compared to Figure 4.2. It reaches the highest accuracy of 90% in the medium range.



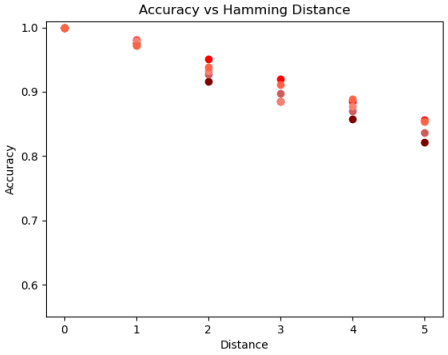
(a) ΔG° accuracy



(b) Accuracy for difference in GC



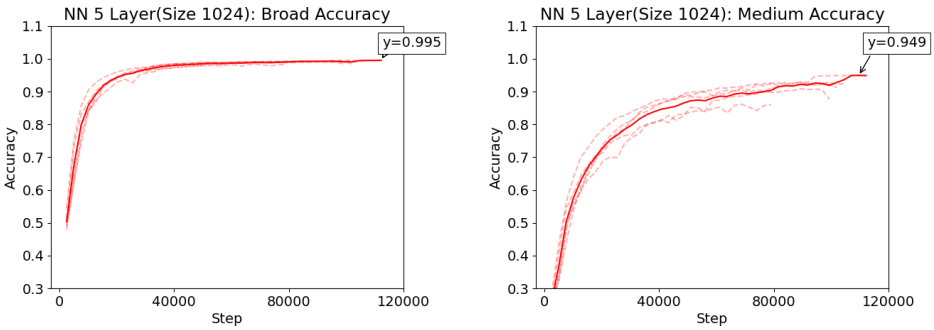
(c) Accuracy for the longest common factor



(d) Accuracy for Hamming Distance

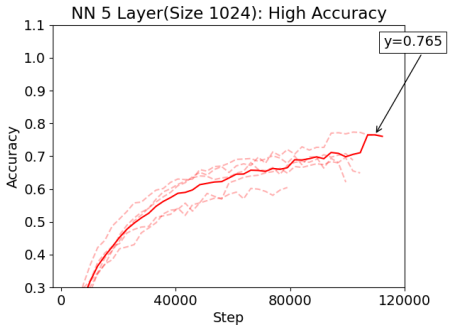
Figure 4.11: The different accuracies for the different features for the medium accuracy, which is ±1.0 accuracy. All runs were run on the Nearest Neighbour model with 3 layers.

As can be seen in Figure 4.11, the variance is quite low in all the different feature plots. The runs seem more consistent than the ones in Figure 4.5.



(a) ±2.0 accuracy

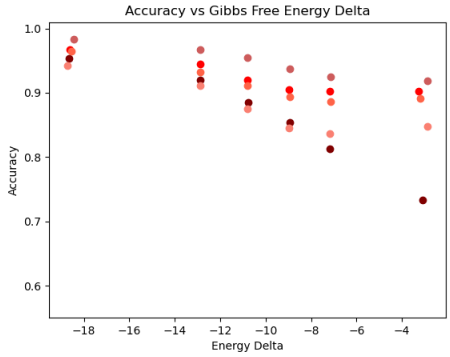
(b) ±1.0 accuracy



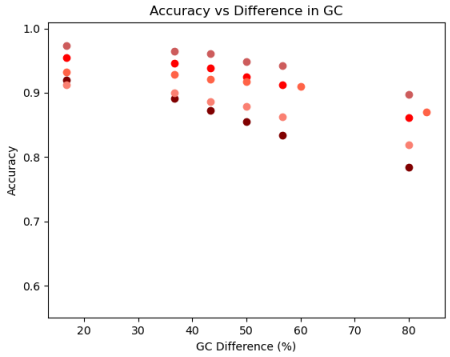
(c) ±0.5 accuracy

Figure 4.12: The different accuracies for the neighbour embedding model with 5 layers, where the first layer has size 1024, and the rest have increasingly smaller. The dotted lines are different runs, and the clear line is the average of the different runs. The highest value of the average plot is highlighted.

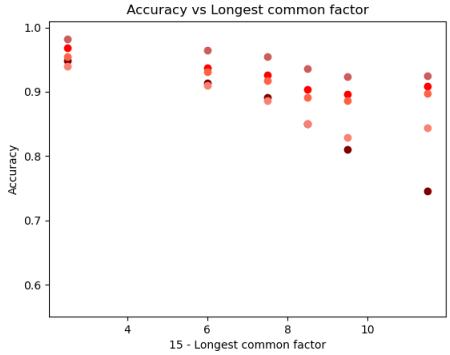
In Figure 4.12 one can see that the accuracy is quite a bit higher for every metric, compared to Figure 4.10, but the dotted lines, which are the different runs, show a larger variance. An accuracy of close to 95% is achieved in the medium range. The variance of the result is further highlighted in Figure 4.13, where for every different feature, there is a clearly visible high variance between the different runs.



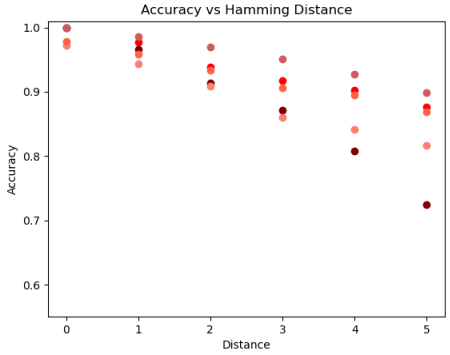
(a) ΔG° accuracy



(b) Accuracy for difference in GC



(c) Accuracy for the longest common factor



(d) Accuracy for Hamming Distance

Figure 4.13: The different accuracies for the different features for the medium accuracy, that is ± 1.0 accuracy. All runs were run on the neighbour embedding model with 5 layers.

4.3 Hypertuning

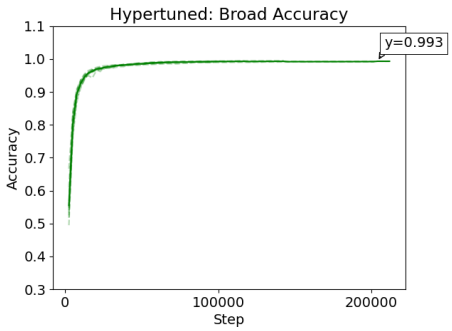
The Hypertuning was done with medium accuracy ±1.0. Since the hypertuning of the parameters has a lot of potential combinations, it takes quite a time to go through them all. Therefore, there had to be multiple runs with only 30 different combinations for each run. The best combinations of every run then had to be compared to each other.

Run	L1 size	L2 size	L3 size	L4 size	Lr	Batch size	Dropout	Accuracy
1	256	256	256	64	0.001	256	0.2	0.933572
2	512	512	512	256	0.002	128	0.2	0.952813
3	1024	1024	256	128	0.002	64	0.2	0.93357
4	1024	256	256	64	0.003	128	0.1	0.892021
5	**128**	**1024**	**512**	**256**	**0.003**	**128**	**0.2**	**0.959404**
6	128	512	1024	128	0.003	128	0.3	0.947577
7	128	1024	128	64	0.0025	128	0.3	0.953714
8	128	2048	1024	128	0.002	64	0.1	0.951053
9	512	512	512	512	0.0025	128	0.1	0.950634

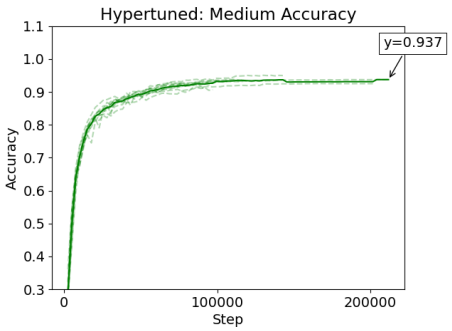
Table 4.1: The best combinations for the different runs

Table 4.1 shows the best combinations for every run and their end accuracy. Run 5 had the best combination of all the runs, which ended with an accuracy of 95.9%. However, since there are a total of $4^6 * 3 = 12288$ possible combinations and only 9 runs with 30 combinations each, that is 270 combinations were run, there could still be better combinations not tested yet.

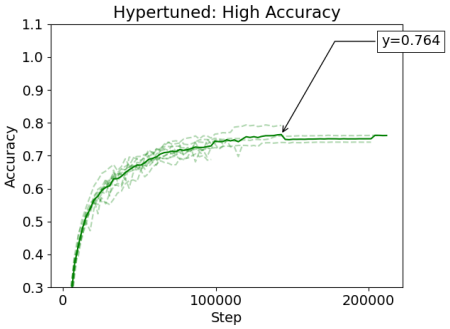
Figures 4.14-4.15 are 5-layer runs with the parameter combination that got the best result in the hypertuning. These runs were run 10 times as opposed to all the previous graphs that only have 5.



(a) ±2.0 accuracy



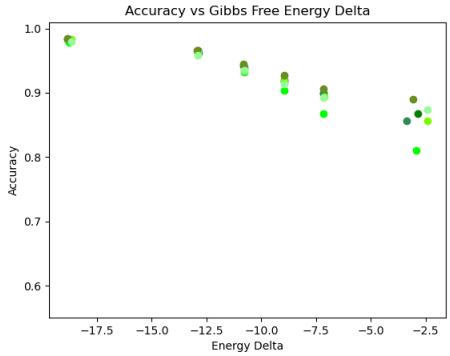
(b) ±1.0 accuracy



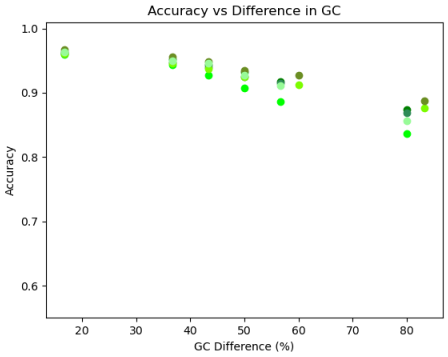
(c) ±0.5 accuracy

Figure 4.14: The different accuracies for the baseline model with 5 layers after hypertuning. The dotted lines are different runs, and the clear line is the average of the different runs. The highest value of the average plot is highlighted.

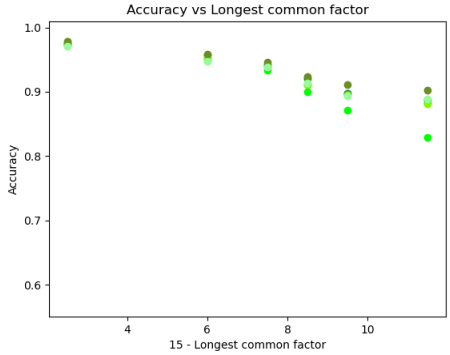
Figure 4.14 shows the results of the final hypertuned model. The highest medium accuracy achieved is approximately 94% in the medium range, which is similar to other results. The model is quite stable, as there is not much variance other than for the high accuracy, which is usually the most unstable accuracy.



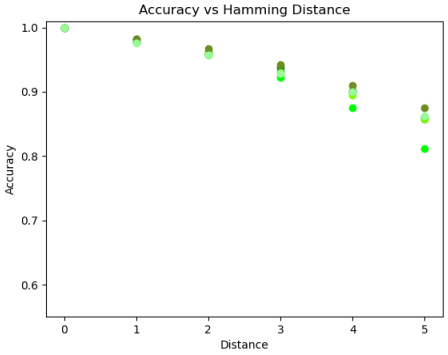
(a) ΔG° accuracy



(b) Accuracy for difference in GC



(c) Accuracy for the longest common factor



(d) Accuracy for Hamming Distance

Figure 4.15: The different accuracies for the different features for the medium accuracy, that is ± 1.0 accuracy. All runs were run on the baseline model with 5 layers.

The scatter plots of Figure 4.15 depict similar-looking scatter plots to other results. However, there is somewhat more variance here than shown in scatter plots of the baseline model with 3 layers, Figure 4.7, and 5 layers, Figure 4.9.

4.4 Final Comparison

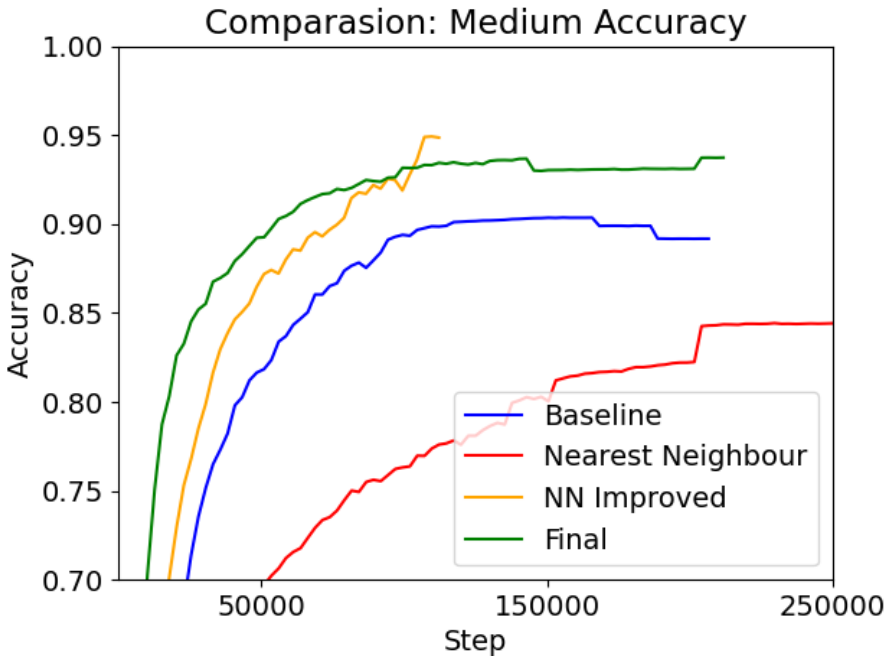


Figure 4.16: The ±1.0 accuracy plotted for both models with 1 layer and the final 5-layer versions of both models for comparison.

In Figure 4.16 it can be seen that the nearest neighbour model gets a higher value in the end, but looking at the graphs from Figure 4.12 it can be seen that the spike of accuracy in the neighbour model comes from the fact that all but one run has died out, so it is no longer the average, but the absolute max of the runs. This means that the final hypertuned is better than the neighbour model overall. However, looking at Figure 4.8 with intuitively chosen parameters, it achieves greater accuracy than the hypertuned and the nearest neighbour models' spike.

5

Discussion & Conclusion

This section presents a discussion of the results in the previous section and the conclusions that were made.

5.1 Discussion

The results from the testing gave a clear insight into what should be the model going forward. High accuracy in many models was achieved, whereas a few models stood out in overall performance. Below are some insights into different models and thoughts about the performance.

5.1.1 Sequence embedding

The results clearly indicated that the impact of different embeddings was not as substantial as initially anticipated. Surprisingly, modifying the embedding to introduce non-linearity and enhance realism did not guarantee improved accuracy. In fact, it appeared that a more complex embedding led to inferior accuracy levels, as evidenced by the outcomes of the nearest-neighbour model. Additionally, including a third neighbour in the embedding further exacerbated the decrease in accuracy.

This decline in accuracy could be attributed to the process of embedding, which involved multiplying tokens by values based on the nearest neighbour. This approach resulted in a larger number of potential sequences, making it more challenging for the model to learn the appropriate energy associated with each sequence. Notably, when the model architecture was made more complex to account for the input complexity, including in the neighbour models, there was a notable improvement in accuracy. However, even with this improvement, the accuracy of the normal embedding models surpassed that of the neighbour embedding models. The normal embedding models were also affected by an increase in the accuracy of the input complexity change so even if the neighbour embedding came close to being better than the original embedding with this, it was not really that close after changing the original input complexity as well.

Based on these findings, it was concluded that further pursuit of a different embedding approach was not warranted. One potential explanation for the ineffectiveness of the more realistic neighbour embedding could be attributed to the deviations in duplex stability prediction, as mentioned in Section 2.4.3. The combination of the

duplex stability error and the increased number of combinations mentioned above may have led to a cumulative error, making it exceedingly difficult to achieve higher accuracy for these models.

Given the inherent error associated with the other embeddings, the baseline model emerged as the most favourable option. Consequently, efforts were directed toward refining and enhancing the performance of the baseline model in subsequent iterations.

5.1.2 Feature Analysis and model improvement

During the analysis of the results, it became evident that the models exhibited varying levels of proficiency when dealing with different aspects of the data. The figures presented in the feature comparison section of the results unequivocally highlighted discernible disparities in the accuracy levels across various features. However, a common observation across all features was that as the RNA sequences deviated further from the ideal match, the accuracy of the models noticeably declined. Furthermore, inconsistencies between model runs were also observed.

To address these discrepancies and enhance accuracy, extensive efforts were undertaken through multiple trials. Various approaches were explored, including the adjustment of different parameters and the manipulation of the number of layers within the linear component of the models. Notably, increasing the number of linear layers yielded promising outcomes by reducing the disparity in accuracy between RNA sequences with minimal dissimilarities and those with more pronounced dissimilarities. Furthermore, an increase in the number of layers resulted in a decrease in the variability observed between different runs of the models.

When exploring these approaches, once again it was evident that the nearest neighbour model was not quite good enough. While increasing the number of linear layers did reduce the disparity in accuracy for every model, the reduction was larger for the models with simple embedding. When increasing from a singular linear layer to three linear layers, the biggest decrease in disparity was achieved for both types of embedding. However, as can be seen when comparing Figure 4.7 and Figure 4.11, the baseline model performs better across the board. These differences in performance could be observed in any test.

Since adding additional layers worked so well, a total of five linear layers were tested as well. This seemed to work quite well, but only when changing the sizes of the layers involved. In retrospect, it might have been better to further investigate using only three layers and see if that was the best combination, but the thought at the time was that the baseline embedding with five linear layers would be the best-performing model.

5.1.3 Hypertuning

After getting a sense of how to achieve minimal dissimilarities, hypertuning started, where different hyperparameters were tested in combinations, see Table 3.2. As the

hypertuning results show, see Table 4.1, it became evident that some combinations of parameters were a lot better than others. The highest-performing combination got a medium accuracy of 95.9% which was the highest accuracy observed. However, it is possible that three layers would have performed on a similar level if hypertuned. Figure 4.6 has an accuracy of 95.3%, but it's an average of multiple runs, while the result from the hypertuning is a singular result.

As mentioned in the results section, only 270 out of 12288 different combinations were tested. This fact makes the hypertuning approach that was used quite questionable. It's very plausible that the best combination hasn't been tested, especially since the average result of the 5-layer model with intuitively chosen hyperparameters got a better result in Figure 4.8 than the average of the hypertuned Figure 4.14. Since the hypertuning achieved an accuracy of 95.9% however, it is not only the fact that only 270 out of 12288 were tested that attributes to this problem, but also the fact that only one run was run per combination which gave a big outlier from the average.

5.1.4 Future work

Moving forward it would be interesting to see if an even higher accuracy could be achieved by using an improved version of the nearest neighbour model, using some of the improvements presented in Section 2.4.3. A more accurate duplex stability prediction might make it easier to predict the $\Delta\Delta G^{\circ}$ for complex RNA sequences, which was the main issue for the nonlinear embedding models, where the drop-off in performance was larger and more varied than that of the linear models.

Additionally, further parameter tuning could be made, in a more organized manner. It could be possible to get a few percentages higher accuracy by tweaking the numbers some more.

One could also try to use different deep learning models and see if they would give a better result. A transformer decoder model might be of interest since they tend to work well on sequences as well.

Lastly, this model can't be used for medical purposes quite yet. This is a first step where deep learning approaches were tested. No actual synthetic ASO strands were used in our datasets. However, when this proved quite successful, the research may continue.

5.2 Conclusion

Overall, the experiments were a success. The goal of the project was to determine if a deep learning regression model could be applied to accurately predict if an ASO sequence would bind to a target sequence, and that was largely a success. A few different models were examined and tested, all of which were RNN models with different modifications to input, output, and internal architecture. Different embeddings were applied to the data input, but in the end, the simpler of the embeddings was the best-performing one.

Despite the project being an overall success, there were some aspects that could have been done better. The hypertuning took a lot of time but did not give the greatest result when taking the investment into account. It is not even sure that the right combinations of hyperparameters were tested, which isn't optimal.

However, an accuracy of over 95% with a margin of error of ± 1.0 is very good considering that the estimation of the energy is not exact and that the most important aspect is deciding if the ASO sequence would bind or not, which can be decided with this margin of error.

Bibliography

[1] D. Buterez, "Scaling up dna digital data storage by efficiently predicting dna hybridisation using deep learning," *Scientific reports*, vol. 11, no. 1, pp. 1–12, 2021.

[2] S. Dara, S. Dhamercherla, S. S. Jadav, C. M. Babu, and M. J. Ahsan, "Machine learning in drug discovery: A review," *Artificial Intelligence Review*, vol. 55, no. 3, pp. 1947–1999, 2021. DOI: 10.1007/s10462-021-10058-4.

[3] B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter, *Molecular Biology of the Cell*, 6th ed. Garland Science, 2015.

[4] M. Nomura, R. Gourse, and G. Baughman, "Regulation of the synthesis of ribosomes and components of the translational apparatus," *Cold Spring Harbor Perspectives in Biology*, vol. 5, no. 1, a013933, 2013. DOI: 10.1101/cshperspect.a013933. [Online]. Available: https://doi.org/10.1101/cshperspect.a013933.

[5] B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter, *Molecular Biology of the Cell*, 4th. Garland Science, 2002.

[6] S. T. Crooke, "Molecular mechanisms of antisense oligonucleotides," in *Antisense Drug Technology: Principles, Strategies, and Applications*, 3rd, CRC Press, 2017.

[7] T. M. Wheeler and T. J. Crook, "Antisense oligonucleotides in drug discovery and development," *Expert opinion on drug discovery*, vol. 11, no. 3, pp. 245–252, 2016.

[8] S. T. Crooke, B. F. Baker, R. M. Crooke, and X.-h. Liang, "Antisense technology: An overview and prospectus," *Nature Reviews Drug Discovery*, vol. 20, no. 6, pp. 427–453, 2021.

[9] H. J. Cleaves, "Watson–crick pairing," in *Encyclopedia of Astrobiology*, M. Gargaud, R. Amils, J. C. Quintanilla, *et al.*, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 1775–1776, ISBN: 978-3-642-11274-4. DOI: 10.1007/978-3-642-11274-4_1683. [Online]. Available: https://doi.org/10.1007/978-3-642-11274-4_1683.

[10] G. Eraslan,. Avsec, J. Gagneur, and F. J. Theis, "Deep learning: New computational modelling techniques for genomics," *Nature Reviews Genetics*, vol. 20, no. 7, pp. 389–403, 2019.

[11] J. SantaLucia, "A unified view of polymer, dumbbell, and oligonucleotide dna nearest-neighbor thermodynamics," *Proceedings of the National Academy of Sciences*, vol. 95, no. 4, pp. 1460–1465, 1998.

[12] P. Atkins and J. de Paula, *Atkins' Physical Chemistry*, 8th. Oxford University Press, 2006.

[13] D. H. Mathews, M. D. Disney, J. L. Childs, S. J. Schroeder, and M. Zuker, "Incorporating chemical modification constraints into a dynamic programming algorithm for prediction of rna secondary structure," *Proceedings of the National Academy of Sciences*, vol. 101, no. 19, pp. 7287–7292, 2004.

[14] H. Huang, X. R. Yang, Y. Y. He, and Y. Gu, "Parameter optimization and sensitivity analysis for thermodynamics-based rna secondary structure prediction," *Journal of theoretical biology*, vol. 360, pp. 45–54, 2014.

[15] P. Zhao, Y. Zhang, X. Li, and Z. Zhao, "Prediction of rna secondary structure using neural network models," *Methods*, vol. 126, pp. 3–10, 2017.

[16] R. M. Schmidt, "Recurrent neural networks (rnns): A gentle introduction and overview," *arXiv preprint arXiv:1912.05911*, 2019.

[17] R. C. Staudemeyer and E. R. Morris, "Understanding lstm–a tutorial into long short-term memory recurrent neural networks," *arXiv preprint arXiv:1909.09586*, 2019.

[18] R. W. Hamming, "Error detecting and error correcting codes," *The Bell System Technical Journal*, vol. 29, no. 2, pp. 147–160, 1950.

[19] E. Ukkonen, "Approximate string-matching with q-grams and maximal matches," *Theoretical Computer Science*, vol. 92, no. 1, pp. 191–211, 1992.

[20] S. Rouskin and M. Zubradt, "Measuring mrna abundance with gc-content normalized rt-qpcr," *Methods in Molecular Biology*, vol. 1543, pp. 73–86, 2017. DOI: 10.1007/978-1-4939-6724-6_6.

[21] D. Gusfield, "Algorithms on strings, trees, and sequences: Computer science and computational biology," 1997.

[22] E. V. Minikel, "Antisense part v: Safety," *Antisense*, 2019.