

Protein-to-mRNA Ratio Prediction from mRNA Sequence using a Deep Neural Network approach

Submitted in partial fulfillment of the requirements for the degree of

Bachelor of Science

in

Artificial Intelligence and Machine Learning

by

Richard Neuboeck

Student Number 210125630



Department of Computing
Goldsmiths, University of London
New Cross
London
SE14 6NW
United Kingdom

March 10, 2024

Version 1.04

Dedication

I would be lying to say that I'm not tempted to follow [Snoop Dogg](#)'s lead but the truth is that there are many people who I owe gratitude. Foremost I dedicate this work to my late mother Helene who always encouraged me. Thank you Sam for the time and effort to polish up my writing, Irene for reading between the lines and your recommendations, Lukas for your insights, Brian for our constant technical and inspirational exchange and Ivo for feedback and for allowing me access to specialized hardware. My deepest thanks go to my spouse Viktoria and kids Riva and Ray for their never ending support without which this project would never have been possible. I can't imagine that there are many four and two year olds that patiently listen to neural network design and then ask interesting questions and also "*What's for dinner?*".

Declaration

I, Richard Neuboeck hereby declare that this thesis represents my own work which has been done after registration for the degree of Bachelor of Science (BSc) at the University of London, Goldsmith, and has not been previously included in a thesis or dissertation submitted to this or any other institution for a degree, diploma or other qualifications. Where information has been derived from other sources, I confirm that this has been indicated in the thesis.

Contents

1	Introduction	4
1.1	Template	4
1.2	Background	4
1.3	Concept and Motivation	5
2	Literature Review	6
2.1	Prior Research	6
2.2	Future Research	8
3	Design	9
3.1	Structure	9
3.2	Raw Data	10
3.3	Pre-Processing	11
3.4	Baseline	12
3.5	Neural Networks	12
4	Implementation	15
4.1	Software and Technologies	15
4.2	Hardware	16
4.3	Application	16
5	Evaluation	18
5.1	Pre-Processing	18
5.2	Data Structure	18
5.3	Neural Networks	18
6	Conclusion	21
	Bibliography	23

Introduction

1.1 Template

CM3015 Machine Learning and Neural Networks (#2)

1.2 Background

Biological organisms can comprise of a single cell or a multitude of cells such as in the case of humans who are made up of around 37 trillion cells. Each cell of every living organism and also most viruses stores all the necessary information for building and maintaining one's functionality (aka life) in DNA [1] (apart from RNA viruses, that only use RNA). The machinery inside the cell performing (almost) all the work is called a protein. The pathway, shown in Figure 1.1, from DNA to RNA to proteins [2] is fundamental. When there is need for a particular protein the specific gene, which is a certain part of the DNA, is read and transcribed into RNA. There can be many RNA molecule copies of one gene and each of these copies can translate into multiple copies of the same protein. It is important to note that the ratios between DNA, RNA and protein is not static and can vary in different tissue types. The whole transcription and translation process is tightly regulated as even small irregularities can have dramatic effects. In the worst case, such irregularities can result in cancer [3] [4].

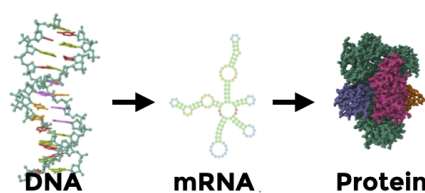


Figure 1.1: Fundamental Biological Pathway

Technological advances have made it possible to synthesise custom RNA molecules at large scale. These find their use among other things in vaccines [5] that once injected use the organism's cellular machinery to produce the encoded protein to perform specific tasks. While the currently most widely known RNA based vaccine is for COVID earlier applications had used this methodology to fight cancer [6]. As described, one RNA molecule can translate into many proteins. The ratio between those molecules, known as the Protein-to-mRNA ratio (PTR), depends on a complex

regulatory system [7]. For vaccines it is an obvious necessity to have a high PTR as one would like to inject a low dosage of RNA and have a significant effect from as many proteins generated as possible.

1.3 Concept and Motivation

There have been various attempts across different scientific disciplines, that can be broadly summed up as bioinformatics, at predicting the PTR (described in more detail in Chapter 2 "Literature Review"). Some with low accuracy inferring the ratio from the presence of other molecules, others using models trained on many measurements that have been more successful. However all these methods relied on numerous costly laboratory experiments to quantify the required parameters. Sequencing available genetic materials is relatively cheap nowadays but every other measurement is still an expensive and time-consuming endeavour. Keeping that in mind, **I propose a deep neural network that can predict the PTR from RNA sequence alone.** This would minimize the need for laboratory experiments and provide the user of the model with an easy and quick analysis opportunity.

A real-life use case is the aforementioned RNA vaccine design done by researchers at pharmaceutical companies (in affiliation with research done at universities). This bioinformatics based research is already done in part on (powerful) computers before the theory is tested in a laboratory. Using the proposed model, it would be possible for the researcher to predict the translational efficiency of the designed molecule within minutes on a computer instead of days in laboratories. Furthermore, the model could be used to find commonalities in RNA molecules with high PTRs which then in turn could be used to enhance a vaccine for example by attaching appropriate untranslated regions 5' and 3' of the coding region.

As previously mentioned, the cellular regulatory mechanism is highly complex and not yet fully understood. Universities around the world invest a lot in pure research into this area. A Google Scholar search for "translation regulation" shows 17,300 matching papers in 2023 alone and much more research is needed to unravel these mysteries. A vast amount of this research is done in silico instead of in vitro as it is much more cost and time effective. By analysing RNA sequences with high PTR value this model could help researchers in identifying regulatory sequence motifs which would add valuable information about regulation of translation.

The output of this research is not intended for end-user consumption but as the example use cases describe for scientific purpose where users base their work on this analysis or expand it. The users have two options:

1. use the pre-trained keras model and incorporate that in their workflow, or
2. use the neural network layer architecture and their own data to train a customized model.

Literature Review

2.1 Prior Research

With the availability of the sequenced human genome [8] in electronic form, researchers began to analyse and annotate regions and identify what sequences are protein coding. Gene sequencing data became abundant and relatively cheap to acquire quickly but protein level measurements are still rare, complex and involve costly experiments. Over the years there have been very different approaches to predict the Protein-to-mRNA ratio (PTR):

Wilhelm et al. [9] used the predicted protein coding gene sequences and correlated this information with in house mass-spectrometry data to estimate which genes do transcribe into proteins. They also attempted to analyse the abundance ratio between mRNA and protein (PTR) and concluded that the ratio between those two is seemingly conserved between different tissues. They corroborated that the transcription rate from DNA to mRNA is a viable proxy for protein abundance prediction. The conclusions from this analysis temporarily shifted the focus to predict how much protein would be produced entirely to the DNA sequence, the regulatory machines attached there and made no difference in what tissues the genes were expressed. Fortelny et al. [10] subsequently highlighted some flaws in Wilhelm's interpretation due to the large scale of the analysis leading to the false assumption that mRNA would not have regulatory control in protein production and the ratio between mRNA and protein would be constant. From their analysis it was clear that detailed measurements of observed protein levels per gene and per tissue would be necessary to underpin predictions. Wilhelm also concurred that a more detailed analysis is necessary in response to this work.

Even though it has been widely accepted that translation rates of expressed genes are preserved across tissues there have been attempts to create models to predict the protein abundance in dependence to mRNA and other factors. Tuller et al. [11] built a linear predictor and showed that basing their predictor only on the mRNA abundance level is not sufficient for an accurate prediction. To improve the accuracy they analysed 32 additional features but found that the inclusion of these additional factors yielded only a negligible improvement. However, in their analysis they found that the inclusion of the tRNA adaptation rate and evolutionary rate significantly improved the prediction accuracy over the base line. In a different approach Mehdi et al. [12] used a Bayesian network in their approach. Aside from the fact that

their model is more complex than the previously used linear predictor, they also used mRNA abundance level, mRNA-protein interaction, mRNA folding energy and half-life and tRNA adaptation levels as inputs. This attempt resulted in slightly higher accuracy than that achieved by Tuller et al. [11]. Despite these disparities in their approach, both attempts share a common reliance on input features that are not cost efficient nor easily obtained. There remains a dependence on expensive experimental measurements.

Advancements toward a less cost-intensive method have been achieved by Eraslan et al. [13]. Like Tuller et al. [11], they apply a multivariant linear predictor model. In their case though they base the training on the mRNA sequence, synthesis and degradation rates and 45 known mRNA and 3 protein sequence motifs. As there usually exist multiple mRNA transcripts per gene, they made sure only the canonical sequence goes into the training as other transcripts are mostly non-coding. The effort they put into adequate feature selection pays off as the output is a much better PTR value than previous models have achieved. In their study, they also found potential new control regions in the transcripts. Even though this method yields promising results, the effort in feature engineering is extensive. Nevertheless they concluded with a very important point: **their results show that a protein-to-mRNA prediction is possible based only on the transcript sequence.**

In a related study about translation by Zrimec et al. [14] they predicted mRNA abundance from DNA sequence data using deep learning. Their convolutional neural network trained on sequence data alone reached a 0.822 level of accuracy. They concluded that not a single motif or region features but the whole regulatory construct of the untranslated and coding region define the level of translation. Since mRNA follows the same regulatory schemes, as explained in detail by Mignone et al. [15], a related approach with refined input could be used to predict the protein abundance per tissue.

The selection and combination of layers for a deep neural network has significant impact on its performance. Many approaches have already been applied with varying success. Another important factor that influences the model performance is the encoding of the biological data. Traditionally one hot encoding of each sequence position has been done. But there are also models that use k-mer encoding and embedding like more recent language models. Trabelsi et al. [16] compare different models and encodings in their paper. They conclude that there is no "one-fits-all" solution and the architecture and encoding depends on the available data and target. However, they showed that **a combination of convolutional and recurrent layers produced more reliable results** than either of them alone. As for the encoding, k-mer would be their recommendation but only if the data set is large enough. Otherwise, there was no significant difference in the performance of the model using one hot encoding or k-mers.

Modelling neural networks is as Chollet (Chapter 9.3) [17] describes less a scientific than an engineering task that relies heavily on experience. For their comparison of neural network architectures for DNA/RNA sequence processing Trabelsi et al. [16] therefore built a program called *deepRAM* that allows for different layer combinations within certain limits of how the layers are arranged and the unit sizes. As

previously mentioned, a model with seemingly similar aim to that proposed herein has been built by Zrimec et al. [14] using a convolutional layer followed by pooling, dropout and dense layers. Another promising neural network model featuring the characteristics of convolutional and recurrent layers and that has been successfully used to predict the function of DNA sequences has been built by Quang, et al. [18]. The commonality between these models is that they are only working with sequence lengths in the hundreds whereas the proposed model for the PTR prediction based on mRNA sequences will have to deal with sequences with lengths of many thousand base pairs.

Further research into neural network layer functionality [19], [20], [21], [22], [23] concludes that the convolutional layer is capable of finding motifs in the input sequence but since these motifs never show up in the exact same position, one and the same motif might be learned multiple times. To avoid this, the addition of a max-pool layer identifies such motifs and scales down the feature space to make it more robust. What is still missing is the connection between several motifs over the whole sequence length which can be achieved using an Long Short-Term Memory (LSTM) layer. Since going over the sequence once might miss some motifs, it is helpful to cover the sequence again in the opposite direction with another LSTM layer. Both are combined into a bidirectional LSTM layer. Dropout is preferably applied in static positions directly to the LSTM layers instead of a separate dropout layer according to Chollet (Chapter 10.4.1) [17]. To prevent premature over-fitting, an additional dropout layer is introduced after the LSTM layers. Two fully connected dense layers subsequently sum the results up. With the lack of an activation function and unit size of 1 the last dense layer will output the expected PTR value regression.

2.2 Future Research

Outside the scope of this work but still a necessary point of research for subsequent work on this topic is the next level of model and encoding using a transformer based architecture. Choi et al. [24] presents many leading research articles that base their analysis with great success on this type of deep neural network. From the evolution of neural network model usage and output shown in this article it is entirely possible that the usage of a transformer model in combination with k-mer encoding would yield a better result.

As most of the neural network toolkit find its basis in Natural Language Processing it can be argued that these tools, though adequate, might need additional adjustments for biological data. *Meta* [25] for example used an NLP approach in their protein structure prediction which shows good performance. Jumper et al. [26] developing *AlphaFold* however came up with their own neural network building block called *Evoformer* as part of their model which dominates protein prediction accuracy currently. In that regard it is likely that biological sequence processing would need refined neural network building blocks as well.

Design

3.1 Structure

The project can be divided into seven major blocks shown in figure 3.1 which in turn can be broken down into many smaller parts described in more detail below.

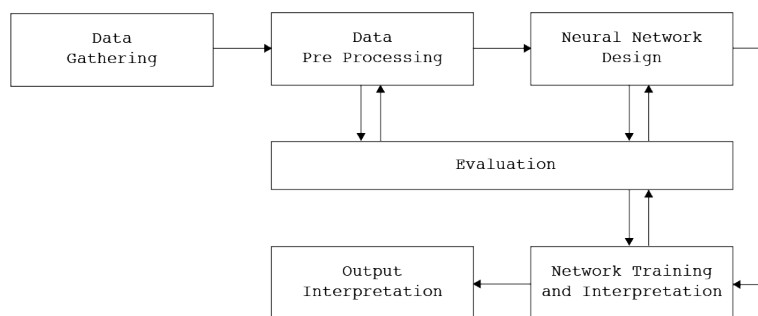


Figure 3.1: Project Flow.

Before delving into the data and pre-processing steps it is necessary to understand the goal of the project in more detail. The general target is a deep neural network capable of predicting the PTR from a given RNA sequence present in a particular human tissue. To approach this target, a simple (common sense) baseline for PTR prediction will be established. For the neural network, however, two strategies; one-hot and multi-hot encoding; and two different network models out of four tested are compared for their respective performances. Figure 3.2 illustrates the analysis pathway. For the one-hot encoded dataset approach, PTRs for one tissue only will be predicted (therefore no separate tissue information will be provided to the neural network). The chosen tissue is "lung" because it has the highest expressed protein count within the given dataset. Preliminary analysis on the small dataset is also a viable way to try different models in faster succession than with the whole dataset. The multi-hot encoded dataset will be concatenated with the categorical one-hot encoded tissue information to provide the neural network with more information to learn from, see Figure 3.3c.

The common way to encode biological sequences for input into neural networks is one-hot encoding each sequence position as shown in Figure 3.3a. This is the first approach tested but this schema neglects annotation information (UTRs, CDS and codon positions) that is readily available and might be useful for the neural network. So, the second approach is to multi-hot encode the sequence and store in addition to

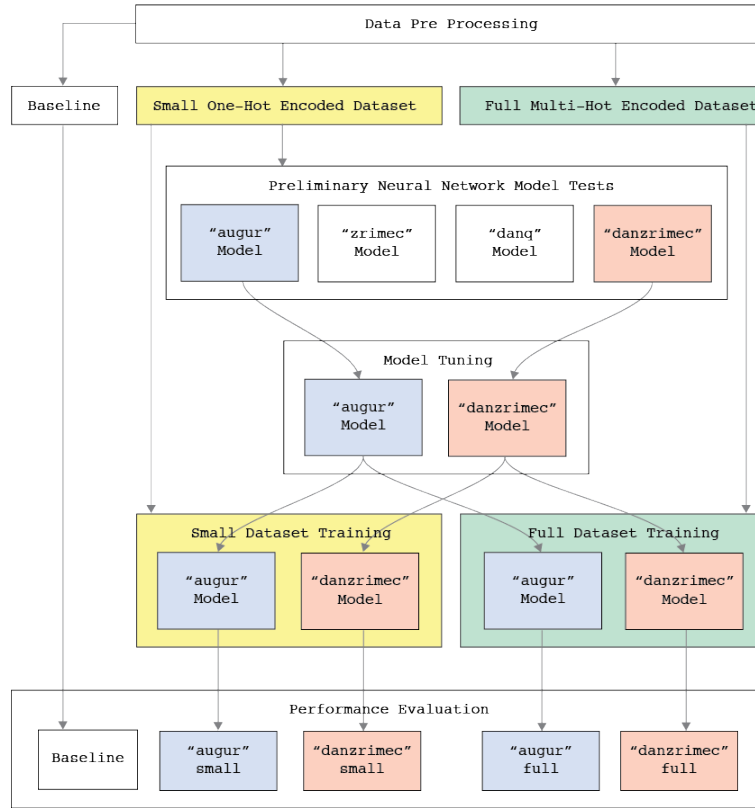


Figure 3.2: Project Pathway : after pre-processing the data the baseline PTR is calculated. A subset is one-hot encoded and used to evaluate different neural network layouts of which two with promising results are chosen. The two models are then trained on the small, one-hot encoded dataset and the full, multi-hot encoded dataset. In the last stage the outcomes of each model is evaluated and compared between models, input and the baseline.

the base pairs (A,T,G,C) [27] if the current position is in the untranslated regions (UTR) [27] or the coding sequence (CDS) [27] and when in the coding sequence which codon position it is (1, 2, 3). This expands the previous four position vector to a ten position vector per sequence letter. The first four are the base, the following three the region (5'UTR, CDS, 3'UTR) and the last three are the codon position (1, 2, 3). Figure 3.3b illustrates this in detail.

3.2 Raw Data

The unprocessed data, that was part of Eraslan's research [13], is stored in a TSV file containing 11,575 rows and 91 columns representing a compilation of many features describing the analysed gene name, gene ID, transcript ID, protein ID, measured mRNA abundance levels for 29 tissues, measured protein abundance levels for 29 tissues and calculated PTR values for 29 tissues. Figure 3.4 shows a small subset of the data. Extracted from this will be the transcript ID and related PTR values per tissue.

The second data source represents the assembled information from Ensembl and GenCode for 89,411 human protein coding RNA sequences stored as FASTA files and annotation data stored as BED files that together occupy 1.3GB. The repository can be found on GitHub under <https://github.com/bleuchien/GENCODE43>.

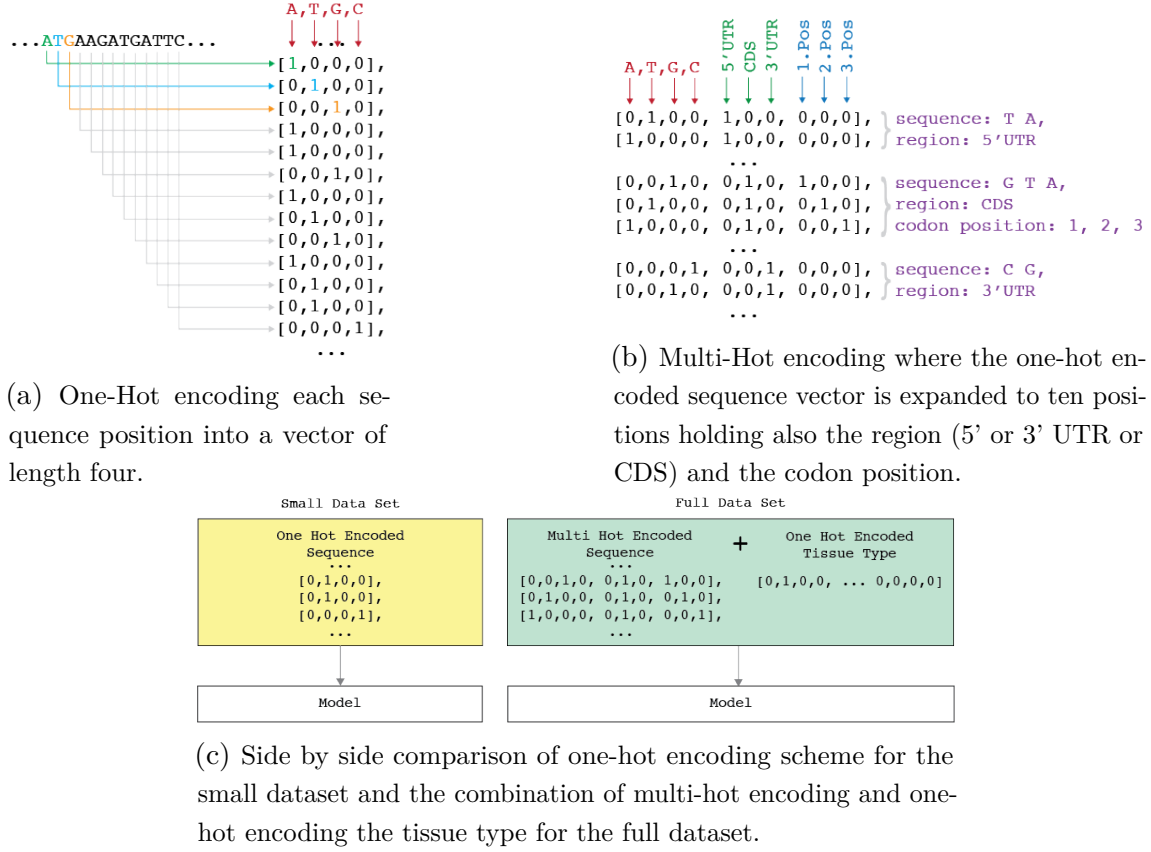


Figure 3.3: Visualisations of the two encoding schemes.

With the above-mentioned transcript ID, the corresponding biological sequence can be extracted in addition to the annotation data containing details about the start and stop base pair for the 5' and 3' untranslated regions as well as the coding sequence.

3.3 Pre-Processing

The first step is the extraction of the transcript IDs and PTR values per tissue from the TSV file. Since the original data was compiled in 2019 and accessed in 2023 for this research, there have been significant updates to the human genome knowledge base. When cross-referencing the transcript IDs with the sequence and annotation files, it is necessary to check the Ensembl online database for the current canonical version of this transcript and update accordingly.

With the assembled information of transcript ID, RNA sequence and annotation data sanity checks for biological appropriate start and stop codons, Clark (Chapter 3) [27], of the coding sequence as well as the length of the coding sequence that must be triplets are performed. Sequences that do not conform to these standards will be dumped from the analysis.

The remaining sequences sport some extreme outliers in sequence lengths that reach up to 109,000 base pairs. To prevent a degraded learning experience for the

GeneName	EnsemblGeneID	EnsemblTranscriptID	EnsemblProteinID	Adrenal_mRNA	Appendices_mRNA	Brain_mRNA	Colon_mRNA	Duodenum_mRNA	Endometrium_mRNA	Esophagus_mRNA	FallopianTube_mRNA	Fat_mRNA	Gallbladder_mRNA	Heart_mRNA
AT1G	ENSG00000121410	ENST00000263100	ENSP00000263100	NA	1.073	NA	NA	NA	NA	NA	1.054	NA	NA	NA
ATCF	ENSG00000148584	ENST00000373993	ENSP00000383105	NA	NA	NA	1.971	2.324	NA	NA	NA	NA	1.569	NA
AZM	ENSG00000175899	ENST00000318602	ENSP00000323929	3.154	3.021	2.824	3.321	3.006	3.344	3.412	2.659	3.653	3.569	3.297
AZML1	ENSG00000166535	ENST00000299698	ENSP00000299698	NA	NA	1.355	NA	NA	NA	3.248	NA	NA	NA	NA
AGALT	ENSG00000128274	ENST00000401850	ENSP00000384794	1.625	1.567	NA	NA	NA	NA	1.691	1.564	NA	NA	NA
AGMT	ENSG00000118017	ENST00000236709	ENSP00000236709	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
AAAS	ENSG00000094914	ENST00000209673	ENSP00000209673	2.314	2.195	2.312	2.336	2.278	2.164	2.144	2.488	2.192	2.301	2.323
AACS	ENSG000000981760	ENST00000316519	ENSP00000324842	1.737	1.692	1.738	1.998	1.958	1.822	2.215	1.891	2.379	1.820	1.708
AADAC	ENSG00000114771	ENST00000232892	ENSP00000232892	2.980	NA	NA	NA	3.154	NA	NA	NA	1.007	2.089	NA
AADAT	ENSG00000109576	ENST00000515480	ENSP00000423341	1.278	NA	1.942	1.550	NA	1.761	1.411	1.539	1.240	1.478	1.408
AAGAB	ENSG00000103591	ENST00000261880	ENSP00000261880	2.153	2.112	2.012	2.325	2.055	2.087	2.110	2.121	2.006	2.008	2.036
AAK1	ENSG00000115977	ENST00000409085	ENSP00000386456	1.321	1.072	1.740	NA	1.002	1.021	NA	NA	1.123	NA	1.806
AAMDC	ENSG00000008784	ENST00000526415	ENSP00000431808	2.829	1.766	2.114	2.192	2.284	2.100	2.047	2.400	2.366	2.078	2.740
AAMP	ENSG00000127837	ENST00000420660	ENSP00000416394	2.613	2.520	2.655	2.732	2.735	2.542	2.562	2.727	2.783	2.711	NA
AAR2	ENSG00000131043	ENST00000373932	ENSP00000363043	2.055	1.968	2.044	2.161	2.054	2.184	2.144	2.212	2.316	2.171	2.073
AARD	ENSG00000205002	ENST00000378279	ENSP00000367528	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
AARS	ENSG00000090861	ENST00000261772	ENSP00000261772	2.543	2.522	2.877	2.455	2.388	2.499	2.382	2.531	2.401	2.367	2.613
AASDH	ENSG00000157426	ENST00000602986	ENSP00000473564	1.626	1.655	NA	1.674	1.546	1.741	NA	1.793	NA	NA	NA
AASDHPT	ENSG00000149313	ENST00000278618	ENSP00000278618	2.229	2.105	2.519	1.979	1.973	2.216	2.196	2.275	2.111	1.932	2.338
AASS	ENSG000000008311	ENST00000417368	ENSP00000403768	1.821	1.489	2.164	1.537	1.348	1.954	1.833	1.790	2.308	1.751	2.079
AAIK	ENSG00000181409	ENST00000417379	ENSP00000398796	1.378	NA	2.248	NA	2.240	NA	NA	NA	NA	NA	NA
AAAT	ENSG00000183044	ENST00000425191	ENSP00000411916	2.214	1.318	3.017	1.603	2.417	1.300	1.377	1.368	1.391	1.277	1.908
ABCA1	ENSG00000165029	ENST00000374736	ENSP00000363968	2.385	1.811	2.056	1.806	1.674	1.367	1.795	1.468	2.504	2.216	1.475
ABCA12	ENSG00000144452	ENST00000389661	ENSP00000374312	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
ABCA2	ENSG00000107331	ENST00000371605	ENSP00000360666	1.875	NA	2.716	1.370	1.592	1.440	1.531	1.586	1.768	1.674	1.807
ABCA3	ENSG00000167972	ENST00000382381	ENSP00000371818	1.772	1.384	2.390	NA	1.231	1.373	1.435	1.258	1.617	1.565	1.631
ABCA5	ENSG00000154265	ENST00000588877	ENSP00000467882	1.896	NA	1.352	1.780	NA	1.418	NA	1.869	NA	NA	1.772
ABCA6	ENSG00000154262	ENST00000284425	ENSP00000284425	1.691	1.440	NA	1.472	1.186	1.362	1.802	1.044	2.471	2.097	1.669
ABCA7	ENSG00000064687	ENST00000435683	ENSP00000465322	1.305	NA	NA	NA	1.462	NA	NA	NA	NA	NA	NA
ABCA8	ENSG00000141338	ENST00000586539	ENSP00000467271	2.524	1.244	2.100	2.194	1.775	1.631	1.964	1.791	2.527	2.336	2.024
ABCA9	ENSG00000154258	ENST00000453985	ENSP00000394264	1.509	NA	NA	NA	1.558	1.446	1.274	2.299	NA	NA	1.665
ABCB1	ENSG00000085563	ENST00000621132	ENSP00000478255	3.045	1.449	1.729	2.204	2.399	1.699	1.374	1.250	NA	1.999	1.256
ABCB10	ENSG00000135776	ENST00000344517	ENSP00000355637	1.670	1.837	1.626	2.022	2.134	1.795	1.872	1.905	1.887	1.728	1.787

Figure 3.4: Raw data in tabular form. Showing abbreviated gene name, Ensembl IDs for the DNA, mRNA and protein and abundance measurements. The full table shows measurements for mRNA and protein abundance for 29 human tissues and the calculated PTR.

neural network due to a poor padding to signal ratio, an appropriate upper threshold needs to be set in place which will be the mean sequence length + standard deviation = 8,000 base pairs.

In the last pre-processing step, the sequence data will be encoded for the two neural networks. One set is one-hot encoded, the other is multi-hot encoded. Both data structures are stored as compressed serialized Python object on disk for further processing.

At this point the smaller, one-hot encoded dataset consists of about 8,000 entries and the multi-hot encoded, full dataset of 214,000 entries.

3.4 Baseline

Predicting the PTR as "good guess" is impossible, but one can analyse the PTR value range in the available data. From that it is easy to get the mean and simply "predict" this value for each sequence which allows the calculation of the mean absolute error. This metric will also be used to assess the neural network performance.

The mean PTR value is 4.980 and the resulting mean absolute error is 0.711.

3.5 Neural Networks

Four deep neural network layouts are preliminary tested with the small dataset for their performance. Design decisions for neural networks are usually based on previous experience in the field and work on this particular data. To overcome the experience bottleneck, two of the layouts are roughly based on models from previous

research. One of the new models is a combination of features of the previous two models and the third is a new model built from scratch and evolved. Central parts of the models are the following layer types (not all models use all layers):

- 1D convolutional layer (with either input dimension 4 [baseline] or 10 [full analysis]) for feature detection [18],
- Bidirectional LSTM layer to find context between the detected features [18], [14],
- 1D max-pool layer to down sample the feature map [19],
- Batch normalization to better generalize after a convolutional layer [20],
- Dense layer to regress the PTR.

The full analysis model has mixed input. One part is the multi-hot encoded annotated sequence and the second part is the one hot encoded categorical tissue type. Both inputs will be combined using a concatenation layer. See Figure 3.5.

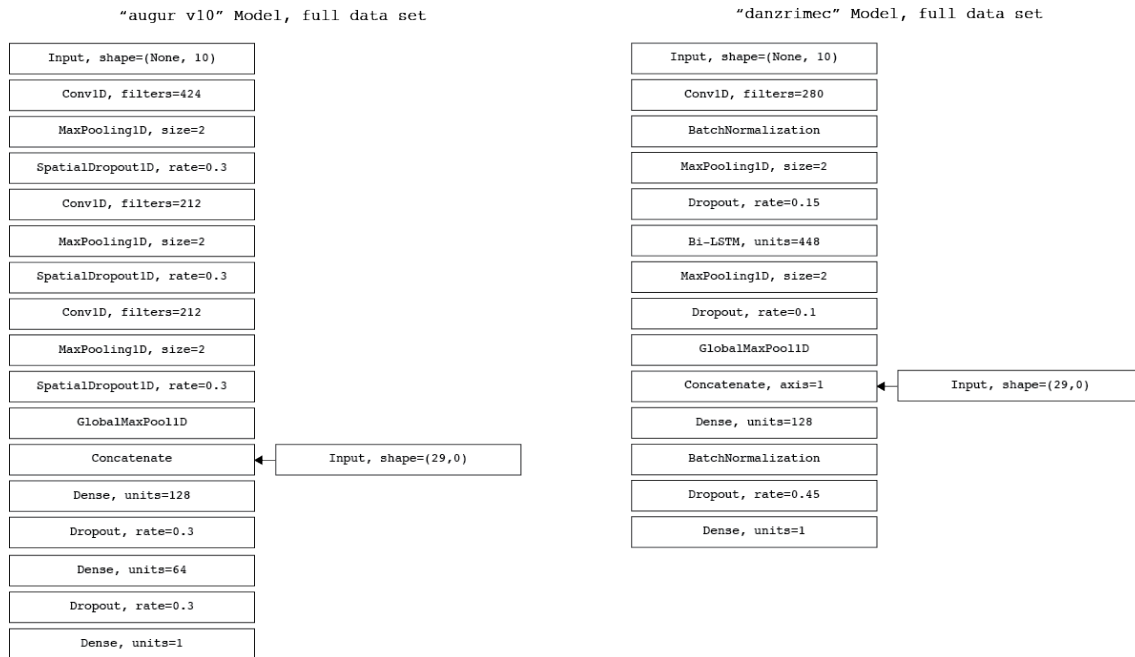


Figure 3.5: Neural Network Models for the full dataset showing each layer type and key information. The primary (top) input is for the full dataset multi-hot encoded and in lower levels concatenated with the one-hot encoded tissue type.

To increase prediction accuracy, k-fold cross validation is applied for the small dataset models since the overall sample count is in the range where both approaches work but using k-fold could gives a slight advantage.

The following models are tested:

- "augur" model : a hand crafted model from scratch : based on information from Chollet [17], Wang et al. [28], and the other models [14], [18], [16],

-
- "zrimec" model : a version of the model Zrimec [14] used and which seems to be reduced version of a *Fully Convolutional Network* (FCN) studied by Wang et al. [28], updated to work with Keras release 2,
 - "danq" model : a version of the DanQ [18] model that needed similar updates,
 - "danzrimec" model : a (hand crafted) combination of features of the Zrimec and DanQ model layout.

Two networks, namely "augur" and "danzrimec", with promising mean absolute error trend are chosen for further development. Those two models undergo a hyperparameter search (keras-tuner) also using a one-hot encoded subset of the whole dataset to find optimal values. The "augur" model undergoes further improvements by adding several new layers to the initial model.

The choice to develop and tune the networks only on a one-hot encoded subset and not the full dataset and both encoding schemes is due to time and resource constraints. The reduced dataset can be processed on a current workstation (based on a 13th generation Intel CPU and 64GB RAM). The full dataset needs more processing power in form of CPU cores or GPU cores and about 12 times as much memory.

Chapter 4

Implementation

As with all projects there are unexpected setbacks, big issues that are sometimes resolved within hours and some seemingly small problems that take days to find a satisfactory solution. The path to the final application is all but straightforward and more like the dynamic model of information retrieval by *Marcia Bates* a meandering path additionally with many branches through that have dead ends.

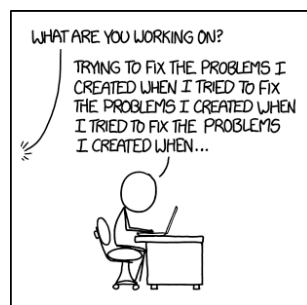


Figure 4.1: Fixing Problems <https://xkcd.com/1739>.

The application was developed as many Jupyter notebooks. The current version including all branches can be found on GitHub under <https://github.com/bleuchien/PTR-augur> where also the raw input can be found.

4.1 Software and Technologies

git and **GitHub** are used for version control and storage of the pipeline and data.


The data for the project was stored initially in TSV, CSV, BED and Fasta files which are all UTF-8 encoded text file based formats. At the end of the pre-processing stage the data structures are stored in binary format as compressed (LZMA) serialized Python objects (pickle).

Most of the processing pipeline is **Python** based and run in **Jupyter Lab** (4.0) with some exceptions where only terminal access was possible so the Jupyter code cells have been combined into one Python script. The **Python** (3.11) libraries used are *pandas* (2.1), *numpy* (1.26), *matplotlib* (3.8), *pathlib*, *re*, *requests* (2.31), *beautifulsoup* (4.12), *pickle*, *lzma*, *keras* (2.15), *tensorflow* (2.15), *scikit-learn* (1.3), *keras-tuner* (1.4), *math*, *time*, *google.colab*, *gc*.

4.2 Hardware

Initial development happened on a 13th generation **24 core Intel CPU** based workstation with **64GB RAM** and on-board graphics. Disk drive limitations are not an issue for this analysis as the dataset is held in memory. Memory constraints on the system made it necessary to move to a bigger machine which is an **AMD Epyc based 128 core system with 512GB RAM** and on-board graphics. In parallel to this attempt some notebooks have been run on a GPU machine equipped with **4 NVIDIA 3090 GPUs with 24GB memory each**. However, the GPU memory was not sufficient to work with the full dataset which made another switch to a **Google Colab Pro+ A100** instance with **40GB GPU memory** necessary. Even there, the for GPU machines optimized, full training of the "danzrimec" model did not run entirely through due to out of memory errors. The large input dataset and deep network with huge layers made the usage of GPU machines infeasible for this model. However, since the "danzrimec" neural network model uses an extensive bi-directional LSTM layer which is running faster on CPU than on GPU (see layer performance evaluation [29]), also considering that the recurrent dropout feature used prohibits cuRNN optimization, this model was trained on an AMD CPU based cluster machine.

Training and evaluation of the "danzrimec" network on the multi-hot and categorical dataset took 15 days using around 100 cores and almost 300GB of RAM. Determining the ideal training length had similar requirements and longer runtime.



```
top - 21:56:22 up 355 days, 3:35, 7 users, load average: 106.05, 100.73, 95.
Tasks: 1176 total, 1 running, 1175 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.0 us, 0.1 sy, 58.6 ni, 31.3 id, 0.0 wa, 1.0 hi, 0.1 si, 0.0 st
MiB Mem : 515618.0 total, 76366.5 free, 437724.0 used, 1527.5 buff/cache
MiB Swap: 16192.0 total, 11.9 free, 16180.1 used, 74863.4 avail Mem

  PID USER      PR  NI  VIRT  RES  SHR  S  %CPU  %MEM    TIME+  COMMAND
2353965 hawk    30  10 343.8g 285.8g 66112 S   8342  56.6 929784:27 python
2446573 root     20   0 1940+ 4916 3832 S    0.7   0.9 41:15.52 top
```

Figure 4.2: Resource Usage Screenshot.

NB: Access to appropriate hardware for the scope of this project is difficult to manage and/or very cost intensive.

4.3 Application

The whole pipeline consists of multiple Jupyter notebooks.

Pre-processing of the raw data 3.4 is split into two notebooks and ends with the storage of one and multi-hot encoded data stored compressed on disk.

- *data_preprocessing.ipynb* : import of the raw data, extraction of the relevant information, cross referencing transcription IDs with Gencode dataset IDs and updating the transcription IDs if necessary by querying the Ensembl database servers.
- *data_preprocessing2.ipynb* : import of the stage 1 CSV data (previous pre-processing output), importing the RNA sequence, UTR and CDS location data, sanity checking the coding region, sequence length analysis and threshold selection. One-hot and multi-hot encoding, compression and serialization of the data (pickle files).

The RNA sequences in this analysis are of different lengths from which two issues arise: the efficient dealing with in-homogeneous data structures and handling of neural network input padding for equal length. Usually, the necessary padding is applied in one of the first steps. This would alleviate the in-homogeneity but it would also grow the size of the input data tremendously and increase the noise to signal ratio to a degree that would have negative learning impact for the neural network. To solve this problem, Tensorflow allows for application of padding per batch when the input data is stored as Tensorflow dataset. To further decrease the padding overhead, the (test) input is sorted according to sequence length prior to padding application.

The following series of notebooks contains the analysis pathway:

- *neuralnet.ipynb* : common sense base line and preliminary analysis of four different neural network models,
- *neuralnet2.ipynb* : initial refinement of the hand crafted "augur" model,
- *neuralnet2-ds.ipynb* : hyper parameter optimization using keras-tuner for the "augur" model,
- *neuralnet2-ds-dzmodel2.ipynb* : hyper parameter optimization using keras-tuner for the "danzrimec" model,
- *neuralnet_baseline.ipynb* : full training and evaluation of one-hot encoded (small) dataset using k-fold cross validation for models "augur" and "danzrimec",
- *neuralnet_baseline-opt2.ipynb* : exploration, training and evaluation of further evolved "augur" models,
- *neuralnet_full.ipynb* : full training and evaluation of multi-hot encoded and categorical dataset for models "augur" and "danzrimec",
- *neuralnet_full-opt2.ipynb* : full training of the best "augur" model (v10) from the previous optimization.

Additional notebooks have been created to answer questions that arose during development:

- *neuralnet-unsorted.ipynb* : tests to make sure that there is no negative impact of sorting the training input,
- *neuralnet-ragged.ipynb* : initial tests to create a ragged tensor,
- *neuralnet-batch*.ipynb* : notebooks to test the impact of batch size on the learning curve. Tested batch sizes: 16, 32, 64, 128. For the dataset at hand 64 has turned out to be optimal but running the "danzrimec" model on the A100 GPU made a batch size reduction necessary.

Evaluation

5.1 Pre-Processing

The original data source (from 2019) was out of date and updates were necessary. The pre-processing pipeline worked flawlessly. Data integrity throughout the conversion and modifications was ensured by continuous monitoring of the record count, manual inspection of out of the ordinary entries and random probing of the dataset. The proper function of the one-hot and multi-hot encoding method has been verified manually with hand crafted test data and on random samples from the real dataset. Data storage and retrieval during and after the pre-processing stages on and from the file system has been reviewed by comparing the in and output variables for identity.

5.2 Data Structure

Interestingly in-homogeneous arrays turned out to be a challenge. One would imagine that this kind of input is not out of the ordinary. Numpy allows for the automatic creation of variable length arrays when forcing the data type "object" but it uses 64-bit integer variables for every other entry making the structure unnecessarily memory greedy. By manual array creation and forcing the usage of 8-bit integers, it was possible to significantly reduce the memory footprint. The Tensorflow dataset, which is necessary to apply batch wise padding, can handle in-homogeneous arrays but only as ragged tensors. The default creation of this data structure is fairly straightforward but debilitatingly slow for large datasets. The applied solution temporarily reduces the dimensionality of the nested arrays while retaining the information of which array entry belongs to which previous sub array. The speedup using this seemingly cumbersome workaround for the large dataset was from more than 24 hours of processing (aborted before it could finish) down to 35 seconds. Overall a significant amount of time was necessary to determine the ideal combination of Python lists, numpy arrays and Tensorflow dataset data structures as well as the best optimization method. However, the performance gain and reduction in memory usage increased the productivity considerably.

5.3 Neural Networks

The final model output allows for the comparison of the performance of different neural network models and input schemes with varying complexity.

Several different network layouts have been preliminary tested on the smaller one-hot encoded dataset. The chosen metrics are **mean square error** (MSE) and **mean absolute error** (MAE). MSE was selected for the smoothness around zero and therefore better performance during gradient descent, Chollet (Chapter 10.2.3) [17], while MAE gave a better sense for the accuracy of the model. Neither the initially tested DanQ [18] nor Zrimec [14] model on their own yielded useful results with the given mRNA sequences because they were developed for much shorter input lengths and less spread. The combination though was a deeper neural network that showed promising results. The two layouts with MAE closest to the baseline - "augur" and "danzrimec" - were selected for further analysis.

The basic "augur" model combined with the smaller dataset underwent further refinements by introducing step by step additional layers of type Dense, Convolution, MaxPool1D, SpatialDropout1D and a bi-LSTM to significantly improve the MAE as Table 5.1 shows. The changes were based on the idea to add blocks of layers comprised of a convolutional layer for subsequent motif detection followed by a pooling layer for generalization as motifs usually do not occur in the exact same location and a dropout layer to aid the learning process by generalization. The final model iteration 3.5 also uses a bi-directional LSTM layer like the "danzrimec" model. An additional dropout and dense layer pair was introduced at the end of the model to help improve generalization capabilities.

Model	v1	v6	v7	v8	v9	v9b	v10
"augur"	1.001	1.061	0.932	0.802	0.726	>1.2	0.707

Table 5.1: MAE values visualizing the "augur" model evolution.

The hyperparameters of both models combined with the small input dataset have been initially tuned with a 100 epoch Bayesian optimization keras-tuner run to find optimal parameters. Following this each model has been trained on training subsets of each dataset until there was either a continuous increase in validation MAE or no noticeable improvement within the evaluation time. The final models have then been trained on the union of training and validation subsets until the previously established validation MAE low point has been reached and subsequently evaluated on yet unseen testing data.

The performance of the neural networks is measured in two steps. First by validating that the network in combination with a particular input encoding was able to beat the baseline and second by comparing the MAE values of the different network architectures between each other. As a lower MAE value is closer to the real value, lower is better. The testing dataset elements have been picked at random from the initial dataset at the beginning of the analysis. Testing the network on data that is not in the original dataset would, if the consensus mRNA sequence is known, mean at least in-vitro mRNA and protein abundance measurements, to have values to compare the network output with, which is outside the scope of this analysis.

Table 5.2 summarizes the performance of both network architectures and respective encoding schema used. With one-hot encoded input the model "augur v10"

was a slight improvement over the baseline and the "danzrimec" model was not even close to the baseline. Using multi-hot encoding the "danzrimec" model significantly improved over the baseline while the "augur v10" results were marginally worse.

	Small Dataset one-hot encoded		Full Dataset multi-hot encoded	
Baseline	"augur v10"	"danzrimec"	"augur v10"	"danzrimec"
0.712	0.707	0.900	0.728	0.588

Table 5.2: Summary of the MAE evaluation results of the selected models trained with the respective dataset in comparison to the baseline.

There are several noteworthy observations:

- The purpose of the one-hot encoded dataset was it to keep the input as simple as possible to observe potential benefits of providing the models with simple or more complex multi-hot encoded input. To accomplish this it was necessary to select one particular tissue instead of all available for the one-hot encoded dataset. This resulted in a substantially smaller dataset, in comparison to the multi-hot encoded dataset, which in turn limited the models capability to learn the complex structures. In an attempt to compensate for the smaller dataset size k-fold cross validation was applied for training.
- With either input encoding the results of the "augur v10" model stayed close to the baseline. There was obviously no benefit, but even a penalty of providing the model with additional details to process the mRNA sequences. This leads to the conclusion that even though the model is seemingly a good starting point, it is in general not yet capable of handling the complexity of the data. As the results in table 5.1 show the FCN approach works. But the model lacks in generalization capabilities when applied to the test dataset judging from an MAE of 0.704 during training and 0.728 during testing. In comparison to that did the training and validation MAE of the "danzrimec" model stay almost the same.
- The "danzrimec" model showed a much clearer, positive trend from one-hot to multi-hot encoded input. The provided additional input features significantly increase the model performance. Simple one-hot encoded sequence input is obviously not a reasonable choice of input as the testing MAE result shows. The architecture of this model is better adapted to handle the complexities of the biological data in combination with the multi-hot encoded approach. The ability for improved learning speed and generalization comes from the combination of BatchNormalization and Dropout layers [30].
- Previous PTR prediction methods relied on costly experimental measurements for model training and also usage while the analysed models only need measurements of mRNA and protein abundance for training. For usage solely the mRNA sequence and annotation data, which is readily online available, is necessary.

Conclusion

Prior research in this field with short DNA sequences already proved that neural networks are capable of adequate prediction results. In this work, it was shown that a novel neural network design for dealing with long sequences and a broad range of different lengths with additional annotations can perform equally. Compared to other neural networks the final models are deep and complex but this comes as a result from huge input variability and the need for further generalization. For more accurate prediction results the analysed neural network architectures need additional refinement and the models further tuning.

With that in mind, the next step to optimize the neural network would build a new pipeline to rapidly test different architectures and run tests on these in parallel on many computers. Powerful hardware with at least 512GB RAM and 128 CPU cores and/or Nvidia A100 GPUs per machine is a necessity for this research. The usage of A100s needs additional enhancements due to the GPU memory limitations and decreased learning efficiency with reduced batch size. Jupyter is a fabulous tool for the initial development phase and visualisations in-between but for rapidly testing different network architectures a pure Python implementation would be easier and faster to deploy with less overhead.

The two models evaluated in this analysis might serve as a sensible starting point for subsequent research. As the evaluation shows a deep neural network is necessary to capture the complexity of the biological data. An option for the expansion of both models would be to stack LSTM layers to increase the learning capacity and capture different levels of regulatory motif abstraction. Additional normalization and dropout layers with different rates might help to generalize further and it would also be worth exploring the effects of supplemental blocks of convolution, pooling and spatial dropout layers. As even the full dataset with 214,000 entries can, in machine learning terms, be seen as small a switch from LSTM to gated recurrent units (GRUs) could have a positive effect on accuracy and processing speed as evaluated by Yang et al. [31]. Especially due to the depth of the models the autotuning of hyperparameters has its own set of problems and will need in-depth testing of different tuners [32]. As tuners do not necessarily result in the optimal parameter selection a manual approach should not be neglected. It would also be prudent to explore different learning rates and other optimizers and their parameters as well.

Even though one of the models yielded considerable better performance using

multi-hot encoding, forgoing the additional annotation and using k-mer encoding is an important next level test. The input would carry less information to the network but the input size would decrease manifold thereby increasing processing performance. Using k-mer encoding would also open up the possibility to try out a transformer architecture that could theoretically bring further improvements [24].

Proposed Pipeline Overview

- Build models and save as *.keras* files,
- prepare multi-hot encoded dataset (and for further studies k-mer encoded dataset),
- send model run and evaluation script in parallel to many cluster nodes,
 - run hyper parameter optimizer,
 - run training with optimized parameters,
 - evaluate and return result,
- choose the most promising model for refinement.

The hardware requirements to run tuning, training and evaluation as described above in parallel are substantial and would still take days to complete a preliminary evaluation of different models, but the outcome would be a highly optimized model that would most likely bring the MAE value down much further.

While the initial data wrangling had its challenges, it was still fairly simple compared to the following optimizations: trying different data structures, combinations of those for efficient usage and many neural network designs. But the limiting factor was resources in form of compute hardware. In retrospect and with the newly acquired wisdom working through these tasks it was ambitious to try to tackle a problem of this magnitude in the time frame of a Bachelor's thesis with the expectation to have an optimal prediction result. Even though particular model and encoding combinations were able to beat the baseline, the MAE results are not yet at production level.

Due to my background in biology and IT and a fresh understanding of neural networks, I can conclude that I would immensely enjoy working on further improvements on this project.

Bibliography

- [1] P. J. Russel, *Genetics*. Benjamin/Cummings, an imprint of Addison-Wesely Longman, Inc., 1998, p. 48.
- [2] A. B., J. A., and L. J. et al., *Molecular Biology of the Cell. 4th edition*. New York: Garland Science, 2002, From DNA to RNA. [Online]. Available: <https://www.ncbi.nlm.nih.gov/books/NBK26887/>.
- [3] N. Robichaud, N. Sonenberg, D. Ruggero, and R. J. Schneider, “Translational control in cancer,” *Cold Spring Harbor perspectives in biology*, vol. 11, no. 7, a032896, Jul. 2019, ISSN: 1943-0264. DOI: [10.1101/cshperspect.a032896](https://doi.org/10.1101/cshperspect.a032896). [Online]. Available: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6601465/>.
- [4] P. Song, F. Yang, H. Jin, and X. Wang, “The regulation of protein translation and its implications for cancer,” *Signal transduction and targeted therapy*, vol. 6, no. 1, p. 68, Feb. 2021, ISSN: 2095-9907. DOI: [10.1038/s41392-020-00444-9](https://doi.org/10.1038/s41392-020-00444-9). [Online]. Available: <https://www.nature.com/articles/s41392-020-00444-9>.
- [5] N. Pardi, M. J. Hogan, F. W. Porter, and D. Weissman, “Mrna vaccines - a new era in vaccinology,” *Nature reviews. Drug discovery*, vol. 17, no. 4, pp. 261–279, Apr. 2018, ISSN: 1474-1776. DOI: [10.1038/nrd.2017.243](https://doi.org/10.1038/nrd.2017.243). [Online]. Available: <https://www.nature.com/articles/nrd.2017.243>.
- [6] C. L. Lorentzen, J. B. Haanen, Ö. Met, and I. M. Svane, “Clinical advances and ongoing trials on mrna vaccines for cancer treatment,” *The Lancet. Oncology*, vol. 23, no. 10, e450–e458, Nov. 2022, ISSN: 1470-2045. DOI: [10.1016/S1470-2045\(22\)00372-2](https://doi.org/10.1016/S1470-2045(22)00372-2). [Online]. Available: [https://www.thelancet.com/journals/lanonc/article/PIIS1470-2045\(22\)00372-2/fulltext](https://www.thelancet.com/journals/lanonc/article/PIIS1470-2045(22)00372-2/fulltext).
- [7] J. W. B. Hershey, N. Sonenberg, and M. B. Mathews, “Principles of translational control: An overview,” *Cold Spring Harbor perspectives in biology*, vol. 4, no. 12, a011528, Dec. 2012, ISSN: 1943-0264. DOI: [10.1101/cshperspect.a011528](https://doi.org/10.1101/cshperspect.a011528). [Online]. Available: <https://cshperspectives.cshlp.org/content/4/12/a011528>.
- [8] “Human genome project timeline.” (), [Online]. Available: <https://www.genome.gov/human-genome-project/timeline> (visited on 01/07/2024).
- [9] M. Wilhelm, J. Schlegl, H. Hahne, et al., “Mass-spectrometry-based draft of the human proteome,” *Nature*, vol. 509, no. 7502, pp. 582–587, May 2014, ISSN: 0028-0836. DOI: [10.1038/nature13319](https://doi.org/10.1038/nature13319). [Online]. Available: <https://www.nature.com/articles/nature13319>.

-
- [10] N. Fortelny, C. M. Overall, P. Pavlidis, and G. V. C. Freue, “Can we predict protein from mrna levels?” *Nature*, vol. 547, no. 7664, E19–E20, Jul. 2017, ISSN: 0028-0836. DOI: [10.1038/nature22293](https://doi.org/10.1038/nature22293). [Online]. Available: <https://www.nature.com/articles/nature23293>.
 - [11] T. Tuller, M. Kupiec, and E. Ruppin, “Determinants of protein abundance and translation efficiency in *s. cerevisiae*,” *PLoS computational biology*, vol. 3, no. 12, e248, Dec. 2007, ISSN: 1553-734X. DOI: [10.1371/journal.pcbi.0030248](https://doi.org/10.1371/journal.pcbi.0030248). [Online]. Available: <https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.0030248>.
 - [12] A. M. Mehdi, R. Patrick, T. L. Bailey, and M. Bodén, “Predicting the dynamics of protein abundance,” *Molecular and Cellular Proteomics : MCP*, vol. 13, no. 5, pp. 1330–1340, May 2014, ISSN: 1535-9476. DOI: [10.1074/mcp.m113.033076](https://doi.org/10.1074/mcp.m113.033076). [Online]. Available: [https://www.mcponline.org/article/S1535-9476\(20\)33106-6/fulltext](https://www.mcponline.org/article/S1535-9476(20)33106-6/fulltext).
 - [13] B. Eraslan, D. Wang, M. Gusic, *et al.*, “Quantification and discovery of sequence determinants of protein-per-mrna amount in 29 human tissues,” *Molecular systems biology*, vol. 15, no. 2, e8513, Feb. 2019, ISSN: 1744-4292. DOI: [10.15252/msb.20188513](https://doi.org/10.15252/msb.20188513). [Online]. Available: <https://www.embopress.org/doi/full/10.15252/msb.20188513>.
 - [14] J. Zrimec, C. S. Börlin, F. Buric, *et al.*, “Deep learning suggests that gene expression is encoded in all parts of a co-evolving interacting gene regulatory structure,” *Nature communications*, vol. 11, no. 1, p. 6141, Dec. 2020, ISSN: 2041-1723. DOI: [10.1038/s41467-020-19921-4](https://doi.org/10.1038/s41467-020-19921-4). [Online]. Available: <https://www.nature.com/articles/s41467-020-19921-4>.
 - [15] F. Mignone, C. Gissi, S. Liuni, and G. Pesole, “Untranslated regions of mrnas,” *Genome biology*, vol. 3, no. 3, REVIEWS0004, 2002, ISSN: 1474-7596. DOI: [10.1186/gb-2002-3-3-reviews0004](https://doi.org/10.1186/gb-2002-3-3-reviews0004). [Online]. Available: <https://genomebiology.biomedcentral.com/articles/10.1186/gb-2002-3-3-reviews0004>.
 - [16] A. Trabelsi, M. Chaabane, and A. Ben-Hur, “Comprehensive evaluation of deep learning architectures for prediction of dna/rna sequence binding specificities,” *Bioinformatics (Oxford, England)*, vol. 35, no. 14, pp. i269–i277, Jul. 2019, ISSN: 1367-4803. DOI: [10.1093/bioinformatics/btz339](https://doi.org/10.1093/bioinformatics/btz339). [Online]. Available: <https://academic.oup.com/bioinformatics/article/35/14/i269/5529112>.
 - [17] C. F., *Deep Learning with Python*. Manning, 2021. [Online]. Available: <https://www.manning.com/books/deep-learning-with-python-second-edition>.
 - [18] D. Quang and X. Xie, “Danq: A hybrid convolutional and recurrent deep neural network for quantifying the function of dna sequences,” *Nucleic acids research*, vol. 44, no. 11, e107, Jun. 2016, ISSN: 0305-1048. DOI: [10.1093/nar/gkw226](https://doi.org/10.1093/nar/gkw226). [Online]. Available: <https://academic.oup.com/nar/article/44/11/e107/2468300>.
 - [19] A. Zafar, M. Aamir, N. Mohd Naw, *et al.*, “A comparison of pooling methods for convolutional neural networks,” *Applied Sciences*, vol. 12, no. 17, 2022, ISSN: 2076-3417. DOI: [10.3390/app12178643](https://doi.org/10.3390/app12178643). [Online]. Available: <https://www.mdpi.com/2076-3417/12/17/8643>.

-
- [20] I. Al-Shourbaji, P. H. Kachare, L. Abualigah, *et al.*, “A deep batch normalized convolution approach for improving covid-19 detection from chest x-ray images,” *Pathogens (Basel, Switzerland)*, vol. 12, no. 1, p. 17, Dec. 2022, ISSN: 2076-0817. DOI: [10.3390/pathogens12010017](https://doi.org/10.3390/pathogens12010017). [Online]. Available: <https://www.mdpi.com/2076-0817/12/1/17>.
 - [21] H. Gunasekaran, K. Ramalakshmi, A. Rex Macedo Arokiaraj, S. Deepa Kanmani, C. Venkatesan, and C. Suresh Gnana Dhas, “Analysis of dna sequence classification using cnn and hybrid models,” *Computational and mathematical methods in medicine*, vol. 2021, p. 1835056, 2021, ISSN: 1748-670X. DOI: [10.1155/2021/1835056](https://doi.org/10.1155/2021/1835056). [Online]. Available: <https://www.hindawi.com/journals/cmmm/2021/1835056/>.
 - [22] W. S. Alharbi and M. Rashid, “A review of deep learning applications in human genomics using next-generation sequencing data,” *Human genomics*, vol. 16, no. 1, p. 26, Jul. 2022, ISSN: 1473-9542. DOI: [10.1186/s40246-022-00396-x](https://doi.org/10.1186/s40246-022-00396-x). [Online]. Available: <https://humgenomics.biomedcentral.com/articles/10.1186/s40246-022-00396-x>.
 - [23] “Differnt ways to combine cnn and lstm networks for time series classification tasks.” (), [Online]. Available: <https://medium.com/@mijanr/different-ways-to-combine-cnn-and-lstm-networks-for-time-series-classification-tasks-b03fc37e91b6> (visited on 02/07/2024).
 - [24] S. R. Choi and M. Lee, “Transformer architecture and attention mechanisms in genome data analysis: A comprehensive review,” *Biology*, vol. 12, no. 7, p. 1033, Jul. 2023, ISSN: 2079-7737. DOI: [10.3390/biology12071033](https://doi.org/10.3390/biology12071033). [Online]. Available: <https://www.mdpi.com/2079-7737/12/7/1033>.
 - [25] Z. Lin, H. Akin, R. Rao, *et al.*, “Evolutionary-scale prediction of atomic level protein structure with a language model,” *bioRxiv*, 2022. DOI: [10.1101/2022.07.20.500902](https://doi.org/10.1101/2022.07.20.500902). eprint: <https://www.biorxiv.org/content/early/2022/12/21/2022.07.20.500902.full.pdf>. [Online]. Available: <https://www.biorxiv.org/content/early/2022/12/21/2022.07.20.500902>.
 - [26] J. Jumper, R. Evans, A. Pritzel, *et al.*, “Highly accurate protein structure prediction with alphafold,” *Nature*, vol. 596, no. 7873, pp. 583–589, Aug. 2021, ISSN: 0028-0836. DOI: [10.1038/s41586-021-03819-2](https://doi.org/10.1038/s41586-021-03819-2). [Online]. Available: <https://www.nature.com/articles/s41586-021-03819-2>.
 - [27] D. P. Clark, *Molecular Biology*. Academic Cell, 2005, pp. 54, 136.
 - [28] Z. Wang, W. Yan, and T. Oates, *Time series classification from scratch with deep neural networks: A strong baseline*, 2016. arXiv: [1611.06455](https://arxiv.org/abs/1611.06455) [cs.LG].
 - [29] “Compare tensorflow performance on cpus and gpus.” (), [Online]. Available: <https://fabrice-daniel.medium.com/compare-tensorflow-performances-on-cpus-and-gpus-af6fdcf29395> (visited on 02/07/2024).
 - [30] C. Garbin, X. Zhu, and O. Marques, “Dropout vs. batch normalization: An empirical study of their impact to deep learning,” *Multimedia Tools Appl.*, vol. 79, no. 19–20, pp. 12777–12815, May 2020, ISSN: 1380-7501. DOI: [10.1007/s11042-019-08453-9](https://doi.org/10.1007/s11042-019-08453-9). [Online]. Available: <https://link.springer.com/article/10.1007/s11042-019-08453-9>.

-
- [31] S. Yang, X. Yu, and Y. Zhou, “Lstm and gru neural network performance comparison study: Taking yelp review dataset as an example,” in *2020 International Workshop on Electronic Communication and Artificial Intelligence (IWECAI)*, 2020, pp. 98–101. DOI: [10.1109/IWECAI50956.2020.00027](https://doi.org/10.1109/IWECAI50956.2020.00027).
- [32] N. Shawki, R. R. Nunez, I. Obeid, and J. Picone, “On automating hyperparameter optimization for deep learning applications,” in *2021 IEEE Signal Processing in Medicine and Biology Symposium (SPMB)*, 2021, pp. 1–7. DOI: [10.1109/SPMB52430.2021.9672266](https://doi.org/10.1109/SPMB52430.2021.9672266).