ABSTRACT

SHAIKH, SOHAIL AHMED. Detecting Ribosomal Pausing with Deep Learning. (Under the direction of Dr. Steffen Heber.)

Translation is the process by which a cell decodes mRNA into protein. Translation and its regulation are essential steps during gene expression, and several important diseases have been linked to defects in genes involved in this process. During translation, ribosomes scan over mRNA molecules, read codons, and synthesize a corresponding polypeptide chain that then folds into a protein. Under certain circumstances, translating ribosomes can pause and accumulate at ribosome pause sites on the mRNA. This pausing can trigger mRNA degradation, changes in protein expression, and pathological states.

This thesis investigates various deep learning approaches to identify ribosome pause sites using ribosome profiling data. Ribosome profiling, also known as ribo-seq, is a novel method based on deep sequencing of ribosome-protected mRNA fragments. The technique allows users to identify individual translated transcripts as well as the precise positions of translating ribosomes on these transcripts, and it provides a genome-wide quantification of gene translation. We use peaks in ribo-seq data to experimentally define ribosome pause sites as mRNA positions which show a significant increase of normalized ribosome footprint counts. This yields the label set for our training and test data. Subsequently, we train and evaluate artificial neural networks using mRNA fragments that represent potential pause sites together with the corresponding labels. Our ultimate goal is to find a neural network that is able to predict pause sites reliably based purely on mRNA sequences without ribo-seq information.

In our study we evaluated different neural network architectures including convolutional neural networks, recurrent neural networks and attention networks as well as different approaches to encode mRNA sequences as inputs. Our best model, a Gated Recurrent Unit based model with an attention layer, obtained a prediction accuracy of up to 89%. We also compared the performance of our models with ROSE (Ribosome Stalling Estimator) and observed an accuracy improvement of up to 7%.

Detecting Ribosomal Pausing with Deep Learning

by
Sohail Ahmed Shaikh

A thesis submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Computer Science

Raleigh, North Carolina

2020

APPROVED BY:

Dr. Khaled Harfoush	Dr. Arnav Jhala

Dr. Steffen Heber
Chair of Advisory Committee

DEDICATION

This thesis is dedicated to my mother.

BIOGRAPHY

The author was born in the town of Miraj in India. He obtained his Bachelor of Engineering degree from University of Pune in 2015. He worked as a Software Engineer for 3 years before starting his Master of Science degree in Computer Science in Fall 2018 at North Carolina State University, Raleigh, NC. ...

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Steffen Heber for his help, advice and encouragement. His passion and motivation were a tremendous source of encouragement for me throughout my thesis. I also appreciate the help of my colleagues, Patrick Perkins, Jessica Parks and Lisa Herzog, for their help and support. Their suggestions and expertise helped me gain invaluable insights in the project. Lastly, I would like to thank my family and friends for their unconditional support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER

1

INTRODUCTION

1.1 Motivation

Ribosome profiling, or ribo-seq, is a method for investigating cellular translation by producing a global snapshot of the positions of all active ribosomes along messenger RNA (mRNA) [17]. During ribo-seq, ribosomes are arrested along mRNA during translation, and the mRNA segments around the ribosomes are digested. The resulting ribosome-protected fragments can be isolated, sequenced, and mapped back to a reference genome to identify the exact position of actively translating ribosomes. It has been shown that this technique is capable of achieving codon-specific resolution of ribosome positions within the translatome, and can be used to assess the translational rate of ribosomes as they move along transcripts [17]. It has been shown that the distribution of ribosomes along strands of mRNA is not uniform due to sequence-specific variations in the rate of translation [41].

The presence of transcript locations which exhibit extremely high ribosome occupancy suggests pausing during translation. Potential translation pause sites are commonly identified by searching for significant peaks in ribosome occupancy. Though the biological relevance of translational pause sites is not yet fully understood, it has been shown to have a role in the regulation of multiple biological processes, including gene expression, mRNA decay, and protein folding. Additionally, ribosome pause sites have been shown to be associated with specific codon usage, sequence motifs, tRNA availability, mRNA secondary structures, and interactions between nascent peptide chains [11, 30, 42] . The enhanced resolution provided by ribosome profiling allows researchers to quantify the presence of ribosome pause sites throughout the translatome in attempts to assess their biological causes and functions.

In recent years deep neural networks have become increasingly popular machine learning tools due to the availability of computational resources, more data and better algorithms. Deep learning has vast ap-

plications and has been incredibly successful in image recognition, machine translation and biological applications. The earliest learning algorithms were intended to be computational models of biological learning known as artificial neural networks. These systems are inspired by the brain however they are generally not designed to be realistic models of biological function.

Deep learning networks, such as Convolutional Neural Networks (CNNs), and Recurrent Neural Networks (RNNs), have become important tools for many classification problems. The traditional neural network architecture consists of an input layer, a hidden layer and a final output layer. Each neuron in the input layer is connected to each neuron in the hidden layer, and each neuron of the hidden layer is connected to the output layer. An activation function, such as *tanh* or *sigmoid*, is applied to each neuron to produce the output [19]. For classification problems, the number of neurons in the output layer is equal to the number of classes for prediction.

Deep neural networks have outperformed traditional methods mainly due to Convolutional Neural Networks (CNNs) [20]. CNNs are a part of most of the state-of-the-art deep learning algorithms and are used for image classification [24], object detection, and image retrieval. CNNs are composed of multiple layers performing convolutional operations and adding non-linearity with each layer. A convolutional neural network tries to perform a series of transformations on the input by learning a set of kernels or weight matrices at each layer. These kernels operate by extracting locally correlated features at each layer and jointly learn weights by dividing the inputs into smaller grids. Each convolution operation is usually followed by a non-linearity, like Rectified Linear Unit (ReLU), which makes them powerful to model complex functions. CNNs have been used to model biological sequence data [42]. In their work, Zhang, et al. use the convolution operators of CNNs as local motif detectors.

Recurrent Neural Networks (RNNs) are neural network models for sequential data. In a biological sequence context, RNNs can process sequence one residue at a time. They also have the ability to store information from the past inputs which make them extremely popular for processing sequential data [19]. RNNs are extremely flexible for different types of sequences as they can handle input sequences of varying length. They can process an entire amino acid sequence at a time to predict if it has certain biological properties. Long Short Term Memory (LSTM) [16] and the recently proposed Gated Recurrent Units (GRUs) [9] are some of the most popular RNN models used for processing sequential data. In addition to these approaches, performance of RNNs can also be improved by using bidirectional layers which can process input sequences and recognize structures in both the forward and the reverse direction. Attention mechanism [2] can also be used to further improve the performance of RNN models by recognizing important structures within a sequence which could be motifs in a biological context.

1.2 Problem Statement

Deep learning has been widely used in several biological and medical applications. The goal of this dissertation is to compare and contrast the methods that can be used to perform classification of pause sites in ribo-seq data.

1.3 Biological Background

The following section gives an overview of the biological background and definitions essential to understanding the problem statement.

1.3.1 Central Dogma of Molecular Biology

The central dogma of molecular biology explains the flow of information from DNA to protein. It describes the process of transcription of the DNA to mRNA and translation of the mRNA to protein [28]. Proteins are complex molecules composed of amino acids that perform many of the important functions in living organisms. A typical protein in the human body consists of 300-400 amino acids [1].

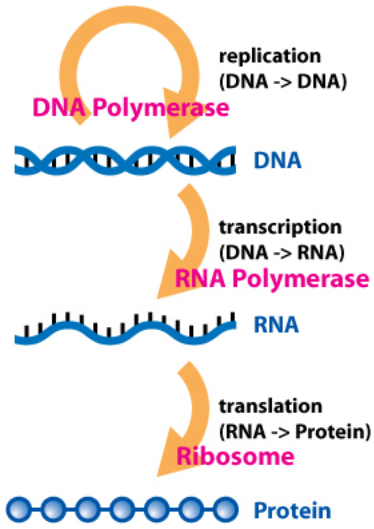

Figure 1.1 Central Dogma of Molecular Biology [10]

1.3.2 Transcription

Transcription is the process by which sections of DNA are copied into RNA. The enzyme RNA polymerase reads the DNA sequence and produces a complementary RNA strand (see figure 1.1).

1.3.3 Translation

In translation, mRNA is decoded by ribosomes to produce an amino acid chain. The amino acid chain is then folded into complex structures resulting in mature proteins. Translation has three steps:

- **Initiation:**
  The ribosome assembles around target mRNA and the first tRNA is attached to the start codon.
- **Elongation:**
  The mRNA is read, one codon at a time, the amino acid matching the codon is added to the peptide chain.
- **Termination:**
  The polypeptide chain is released when the stop codon — UAG, UAA or UGA — enters the ribosome.

After translation the polypeptide chain folds the final mature protein. Figure 1.2 shows the process of translation and the growing polypeptide chain.

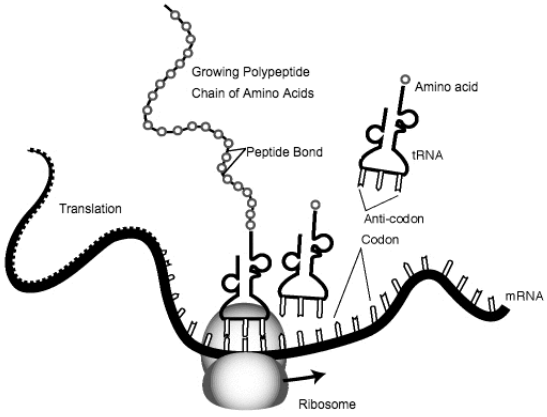

Figure 1.2 Illustration of translation and the growing polypeptide chain. The tRNA transfers the amino acids to the ribosome, which attaches to the growing polypeptide chain during the process of Elongation.

1.3.4 Ribo-Seq

Ribo-seq, also called ribosome profiling or ribosome footprinting, was introduced by Ingolia et al. in 2009 [17]. The technique uses high throughput sequencing [35] to characterize actively translating mRNA, producing a global snapshot of all the ribosomes active in a cell. It is similar to RNA-seq [21] but involves sequencing only the mRNA fragments that are protected by the ribosome during the process of translation.

In ribosome footprinting, translating ribosomes are arrested and the attached mRNAs are treated with nuclease. This process destroys the regions that are not protected by ribosomes, leaving only the protected regions. The protected regions, or "footprints", typically 28-30 nucleotides in length, are then amplified, sequenced and mapped back to the original mRNA to determine the exact locations of translating ribosomes[4]. Figure 1.3 provides an illustration of the process of ribo-seq.

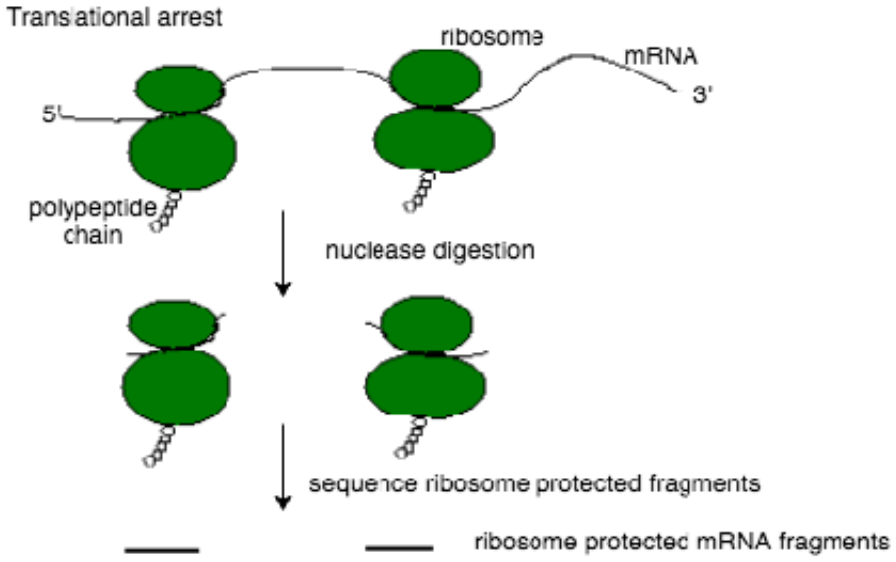

Figure 1.3 The ribo-seq process: actively translating ribosomes are arrested, nuclease digestion destroys the unprotected mRNA fragments, and the protected mRNA fragments are sequenced.

1.4 Machine Learning Background

The following section gives an overview of the topics pertaining to Machine Learning, specifically Deep Learning. Deep Learning algorithms are central to our approach to detecting pause sites in ribo-seq data.

1.4.1 Supervised Learning

Supervised learning algorithms learn to associate an input X with some output Y, inferring a function f given the input X and output Y, such that $Y = f(X)$. These algorithms can be classified into two types:

- **Classification algorithms:**
  The output can be one or several classes and the goal of the learning algorithm is to classify the input into one of the classes.
- **Regression algorithms:**
  The output is a real value, such as temperature values.

1.4.2 Deep Learning

Traditional machine learning algorithms require considerable domain expertise, as they process raw data to extract relevant features for internal representation of the data [25]. In representation learning, the features required for internal representation of the raw data are learned by the algorithms themselves. Deep Learning algorithms leverage representation learning and are composed of multiple simple layers that are able to learn more complex and abstract representations. This makes Deep Learning highly suitable for complex tasks, such as speech recognition, object detection, genomic applications, etc.

Deep Learning refers to deep neural networks, which are inspired by the function of the human brain. Although they're inspired by the human brain, these are probabilistic models and are not meant to be realistic models of biological function. A standard neural network (NN) consists of multiple connected computation units, or neurons, which each produce a real-valued output or activation [34]. An objective function is computed, measuring the error of the output. This error is the distance between the output scores and the desired values. Learning is the process of finding the optimal weights that produce the desired values and minimize this error. A function that is often used for optimizing this procedure is stochastic gradient descent (SGD) [3]. The number of adjustable parameters (optimal weights) of the neural network can range from hundreds to millions in deep networks [25]).

1.4.3 Softmax and Loss Functions

Softmax is commonly used as the output final layer of a neural network based classifier and represents the probability distribution over n different classes. Formally, this is given by:

$$\text{softmax}(z)_i = \frac{exp(z_i)}{\sum_j exp(z_j)}$$

In the above equation, the exponentiation function is applied to each element z_i of the input vector z and this value is normalized by dividing by the sum of all these exponentials. In other words, after applying softmax the output values are transformed such that they have a value between (0, 1) and they sum to 1.

The Cross-entropy loss, or log loss, is commonly used with the softmax function for training the neural network. It measures the performance of a classification model whose output is a probability value between 0 and 1. It is given by the formula:

$$\text{Cross Entropy loss} = -\sum_{c=1}^{m} y_c \, log(\hat{y_c})$$

where m is the number of classes, y_c is the binary indicator if class label c is the correct classification and $\hat{y_c}$ is the predicted probability of the sample belonging to the class label c.

1.4.4 Dropout Regularization Technique

A machine learning algorithm should perform well not only on the training data but also on new inputs. There are two challenges that machine learning algorithms face [12]:

- **Underfitting:** This occurs when the model cannot perform well with respect to minimizing the error on the training set.
- **Overfitting:** This occurs when the model cannot perform well with respect to minimizing the gap between the training and testing error.

Regularization is the process of modifying the learning algorithm to minimize underfitting and overfitting.

Dropout [38] is a practical and widely used technique for regularization of neural networks. It randomly removes units from a network by multiplying its output value by zero during the process of training. This prevents the network from overfitting by making the network resilient to learning noise and adapting too much during training.

1.4.5 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) have become extremely popular and ubiquitous in applications related to computer vision [24]. A key difference between CNNs and traditional Deep Neural Networks is that CNNs employ the convolution operation in the convolution layers, which are the hidden layers in the network. The four key ideas that make up a CNN are local connections, shared weights, pooling, and use of multiple hidden layers [25].

CNNs are composed of multiple layers, performing convolutional operations and adding non-linearity with each layer. A CNN performs a series of transformations to the input by performing a convolution operation between the input and a set of kernels/filters organized into feature maps. This results in the extraction of locally correlated features and in case of sequence data, this could be motifs. The subsequent layers may aggregate the motifs into more complex arrangements [25]. A non-linearity operation, like Rectified Linear Unit (ReLU), is performed after each convolution operation.

The neurons in a CNN compute an output value by applying the convolution operation to the input values coming from the previous layer. The weights of the kernels and the biases are learned by minimizing the error at each iteration. Many neurons in a CNN can share the same filter in a convolving layer, thus making the layer more memory efficient.

A convolutional layer may be followed by a pooling layer in CNNs. The pooling layers could be of different types, such as Max Pooling or Average Pooling. A typical Max Pooling unit takes the maximum value of a local feature map. This reduces the dimensions of the input and the resulting representation is invariant to small shifts and distortions.

Figure 1.4 shows the AlexNet [20] architecture which was implemented in 2012 by Krizhevsky et al. It was one of the most influential papers in Computer Vision, employing CNNs. The first five layers in AlexNet are convolutional layers, followed by max pooling layers, and the last three layers are fully connected layers. It uses the Rectified Linear Unit (ReLU) instead of tanh and sigmoid, which is critical to its performance improvement over previous neural network architectures.

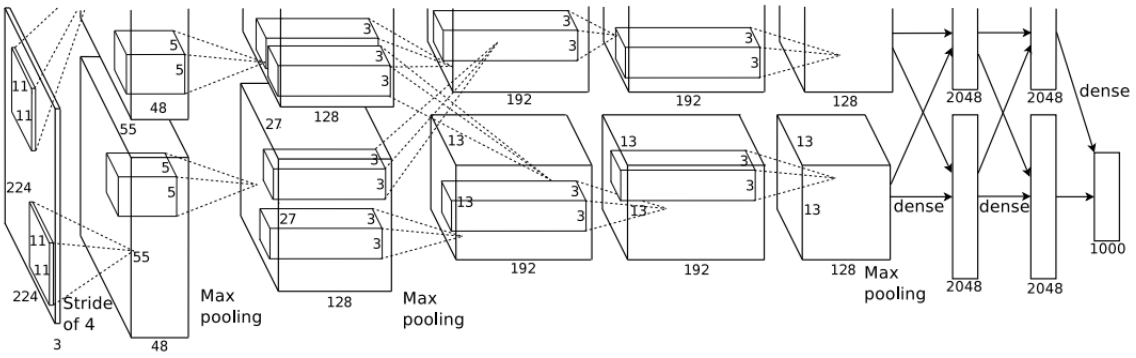

Figure 1.4 Illustration of the AlexNet CNN architecture [20]. The input layer is a 224 x 224 x 3 image. The first convolutional layer has 96 kernels with size 11 x 11 x 3 and stride of 4. The second convolutional layer takes as an input, the output of the previous layer and has 256 kernels of size 5 x 5 x 48. The first and the second convolutional layers, have max pooling layers. The third convolutional layer has 384 kernels of size 3 x 3 x 256, followed by the fourth layer having 384 kernels of size 3 x 3 x 192. The final convolutional layer has 256 kernels of size 3 x 3 x 192, followed by fully connected layers having 4096 neurons each. The last layer is the softmax output layer, predicting 1000 class labels.

1.4.6 Recurrent Neural Networks

Traditional Neural networks do not have a memory or an internal state, making it difficult to use them for input sequences of variable length. Recurrent Neural Networks, or RNNs [32], are most widely used with sequential inputs such as speech recognition, sentiment classification, DNA sequence analysis, machine translation, etc.

In RNNs, the network processes an input sequence one element at a time. The values of the hidden units are maintained in a "state vector" that contains the information of all the previous elements of the sequence. This hidden state is also passed as an input to the next unit of an RNN layer. This enables parameter sharing across different parts of the model and allows generalization across different types of inputs, like sequences with different lengths. Such parameter sharing is particularly important when a specific piece of information can occur at multiple positions within the sequence; for example, in DNA sequences this could be nucleotides or codons that appear in different parts of the sequence. A 1-D convolutional neural network also allows parameter sharing across time, but it is shallow [12]. In RNNs, however, each output value is a function of previous output values. This results in sharing of parameters through very deep computational graphs.

Basic RNN structures suffer from the vanishing gradient problem, which occurs when the weights learned during the procedure of gradient descent becomes infinitesimally small [15]. This prevents the weights from changing at each iteration, and as a consequence, it becomes hard for very deep networks to learn long term dependencies. Gated Recurrent Units (GRUs) [7] and Long Short-Term Memory (LSTM) networks [16] use multiple memory states or gates, which help combat the vanishing gradient problem. The GRUs and LSTMs are the most popular RNNs used in several state-of-the-art machine learning systems, like machine translation.

Figure 1.5 shows an example of an unfolded RNN [14]. Each node represents a layer of the RNN. 'w1' refers to weighted connections from input layer to hidden layer, 'w2' is the weight from the hidden layer to itself and 'w3' refers to the weights from the hidden layers to the output.

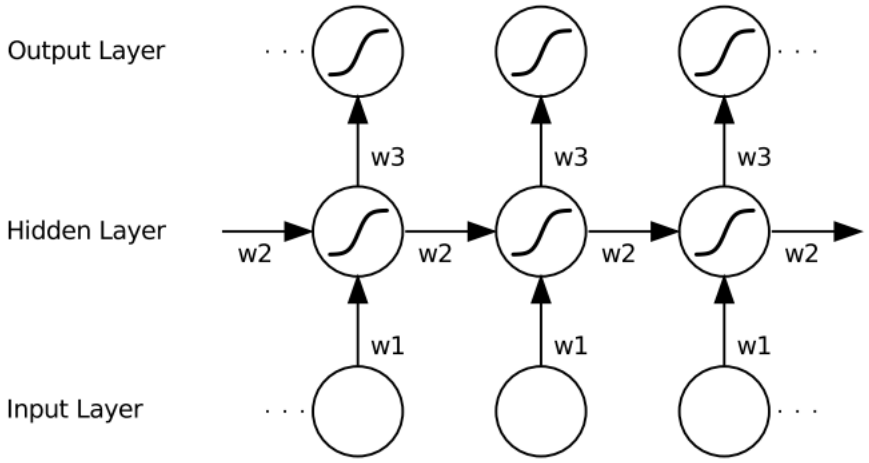

Figure 1.5 Example of an unfolded RNN [14].

1.4.7 Attention Mechanism

Although RNNs work well for short sequences, their performance degrades with an increase in length of the input sequence [8]. The attention models have been one of the most influential mechanisms introduced [2] for solving this problem. They are now used in the most cutting-edge machine translation systems.

The attention mechanism is built around an architecture that consumes the outputs of all the hidden states of the network instead of only the final hidden state. In addition, all hidden states are not just combined using a dot product, but for every output step a distinct vector with different weights applied to the hidden state is computed. These different weights help capture the context of a part of the input and enable allocation of more attention or context to parts of the input that would lead to an improvement in the performance of the neural network.

1.5 Related Work

Several studies have used ribo-seq data to investigate ribosome stalling during translation [5, 6, 22, 29, 31, 33, 40, 42]. Typically, these studies use statistical approaches or traditional machine learning to model the distribution of ribosomes during translation and to infer ribosome pause sites. So far, only few researchers have used deep learning in combination with ribo-seq data. The Ribosome Stalling Estimator (ROSE) uses classification to infer ribosome stalling from ribo-seq data [42]. ROSE uses a prediction model based on a parallel CNN architecture where filters act as local motif detectors. The authors train their models on human and yeast datasets, and they show that their predictions correlate with factors that are known to affect ribosome stalling such as codon usage bias, tRNA adaptation index, codon co-occurrence bias, proline codons, N6-methyladenosine (m6A) modification, mRNA secondary structure and protein-nucleotide binding. The accuracy of the ROSE model was evaluated using cross-validation and the model obtained an AUROC score of 83%.

Ixnos uses a neural network model to predict local mRNA to protein translation rates, and to design the fastest or slowest predicted coding sequence for a gene [40]. The authors use a regression framework to model the ribosomal density along transcripts as a function of sequence features. Ribosomal density is calculated as scaled footprint counts by dividing raw footprint counts at each position by the average footprint count on the transcript. This normalization controls for variable mRNA abundances and translation. The authors determine the sequence neighborhood that best predicts ribosome density by comparing a series of models ranging from an A-site only model to a model spanning codon positions -7 to +5.

RiboSmoothR is a tool for identification of pause sites in ribo-seq data [31]. The algorithm evaluates three windows around each potential ribosome position (upstream, downstream, and centered on the position). It then calculates z-scores for the position based on the mean and standard deviations of ribo-seq read counts in these windows. Based on these three scores a threshold-based algorithm determines peaks in the ribo-seq footprint data which are used as a proxy for ribosomal pausing. RiboSmoothR does not take into account sequence features that are known to affect ribosomal pausing, such as presence of mRNA secondary structures or presence of motifs.

CHAPTER

2

METHODS

This chapter includes the various methods used to apply different learning algorithms to classify ribosomal peaks as well as the methods used to generate labels and generate the ribo-seq dataset used for predicting pause sites. This chapter also includes a set of experiments performed on a simulated dataset to perform motif detection so as to provide an upper bound on prediction accuracy when neural networks are used to extract information from DNA sequences.

2.1 Data Processing

Eight ribosome profiling samples, along with corresponding RNA-seq samples, from *Arabidopsis thaliana* were downloaded from the NCBI SRA [27]. The data was generated by a study of the effect of certain plant hormones on growth rates. The dataset contained four pairs of replicate sets, with two different treatment types and strains. Trimming of the adapters was performed with the FastX tool-kit [13], and the resulting reads were mapped to the TAIR 10 [23] genome sequence of *Arabidopsis thaliana* using Tophat with default parameters [39]. The footprints were assigned to the first position of their P-sites by estimating read-length specific P-site offsets. The P-site of the ribosome is the site at which the tRNA is linked to the growing polypeptide chain, and chemical reactions at this site play a vital role in translation initiation, elongation, and termination. The RNA-seq reads were assigned to their middle position. Some genes can be poorly sequenced due to issues with sequencing depth, which can introduce unexpected biases in identifying ribosomal stalling. To account for this, genes with less than 100 total reads in both RNA-seq and ribo-seq were removed from the dataset. Similar thresholds have been used in other works [33]. Only genes without isoforms were considered, in order to limit the effects that differential isoform-specific translation might have on ribosome abundance when taking a gene-wise perspective. The final set consisted of 5,384 genes.

Data was normalized to remove biases. More specifically, after mapping the ribosome profiling and RNA-seq

reads to the reference genome, their codon-level reads were first scaled by the mean coverage level within each gene, which removed the coverage differences among genes. Next, the scaled ribosome profiling reads were divided by the scaled RNA-seq reads in the corresponding location to eliminate the shared biases between these two fractions. After that, a logarithm operation was applied to yield the final normalized ribosome footprint density. These scaled footprint densities were used to generate labels for the training dataset. A threshold of two standard deviations above the mean of the normalized ribosome footprint counts was set as the definition for peaks, similar to the work done by Zhang et al. [42]. Thus, all the sequences above the threshold were labeled as peaks and the ones below the threshold as non-peaks. To keep the data set balanced for each label, an equal number of samples with peaks and non-peaks were retained, resulting in 14k samples with peak as the label and 14k samples with non-peak as the label.

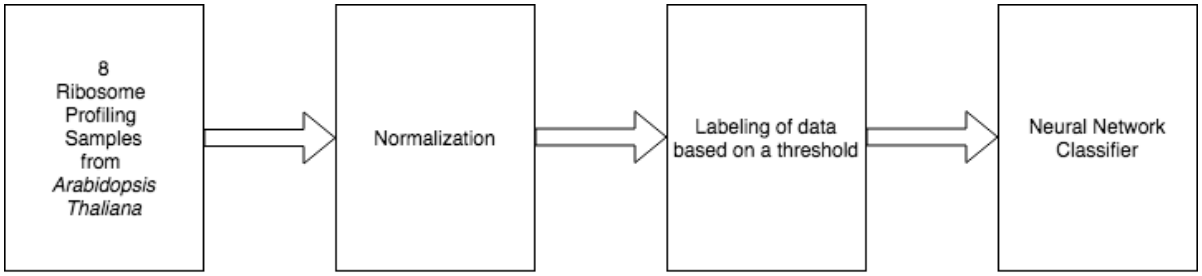

Figure 2.1 Data processing workflow.

Figure 2.2 shows the normalized ribosome footprint counts for the gene "AT3G13750" from *Arabidopsis thaliana*. The dashed red line with a value of 2.7 indicates the threshold above which we label all the sites as having peaks.

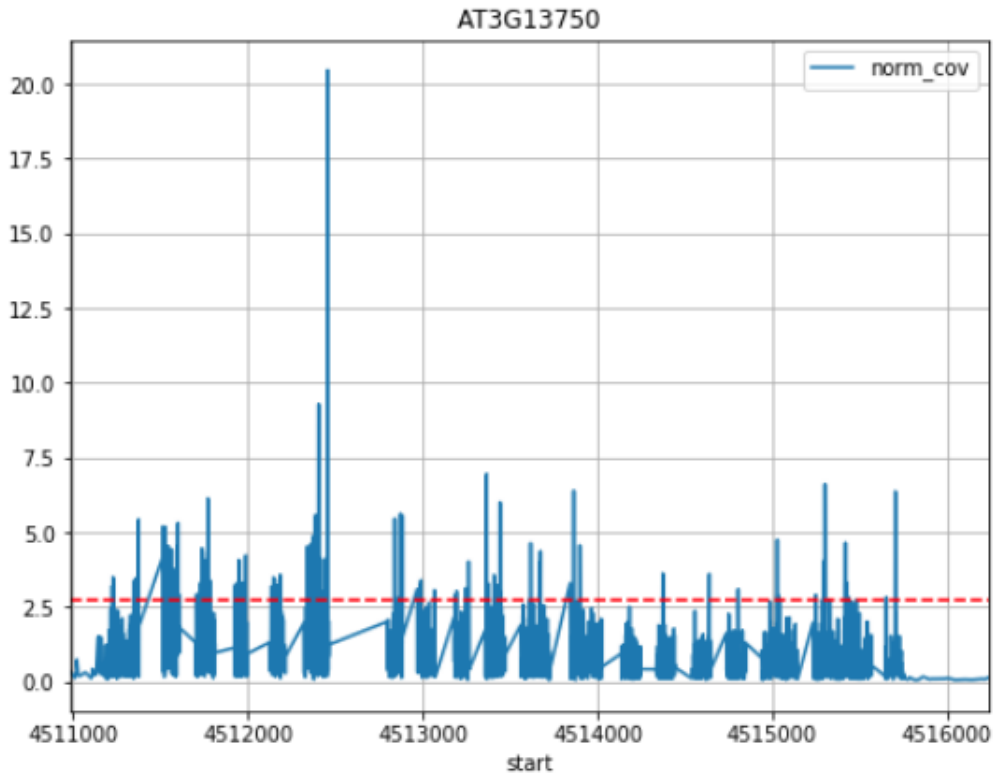

Figure 2.2 Normalized footprint scores for the gene "AT3G13750". The red line indicates the threshold for labeling peaks.

2.2 Data Representation

The inputs to the neural network consist of nucleotide sequences around the P-site. We take 45 nucleotides in the upstream and downstream region of the P-site to get an input nucleotide sequence of 90 nucleotides. The label for each nucleotide sequence is generated as described in the above section. We then encode each sequence of 90 nucleotides so that it can be fed to the input layer of the neural network. For neural networks, input representations affect performance. Input representations should be chosen to minimize information loss.

When working with biological sequence data the numeric representation of the data influences model performance. An obvious solution is to use sparse or one-hot encoding, where an amino acid is represented by a vector of the length 20, containing a single one and 19 zeros. The parameters connecting an amino acid input to the first hidden layer are often called the embedding for that amino acid. For our experiments, we decided to use a one hot encoding representation of DNA sequences. We used two different input representations for our experiments. In the first representation, we encode each of the 64 codons by one of the bits in a binary vector of length 64. Figure 2.3 shows one-hot codon encoding of an input nucleotide sequence of length 90. The benefit of this method is that it reduces the length of the input sequence and thus makes the network faster to train. In the second representation, we encode each nucleotide by one-hot encoding. Figure 2.4 shows one-hot nucleotide encoding of an input nucleotide sequence of length 90. This representation takes longer to train, however we found an improvement in accuracy with this method.

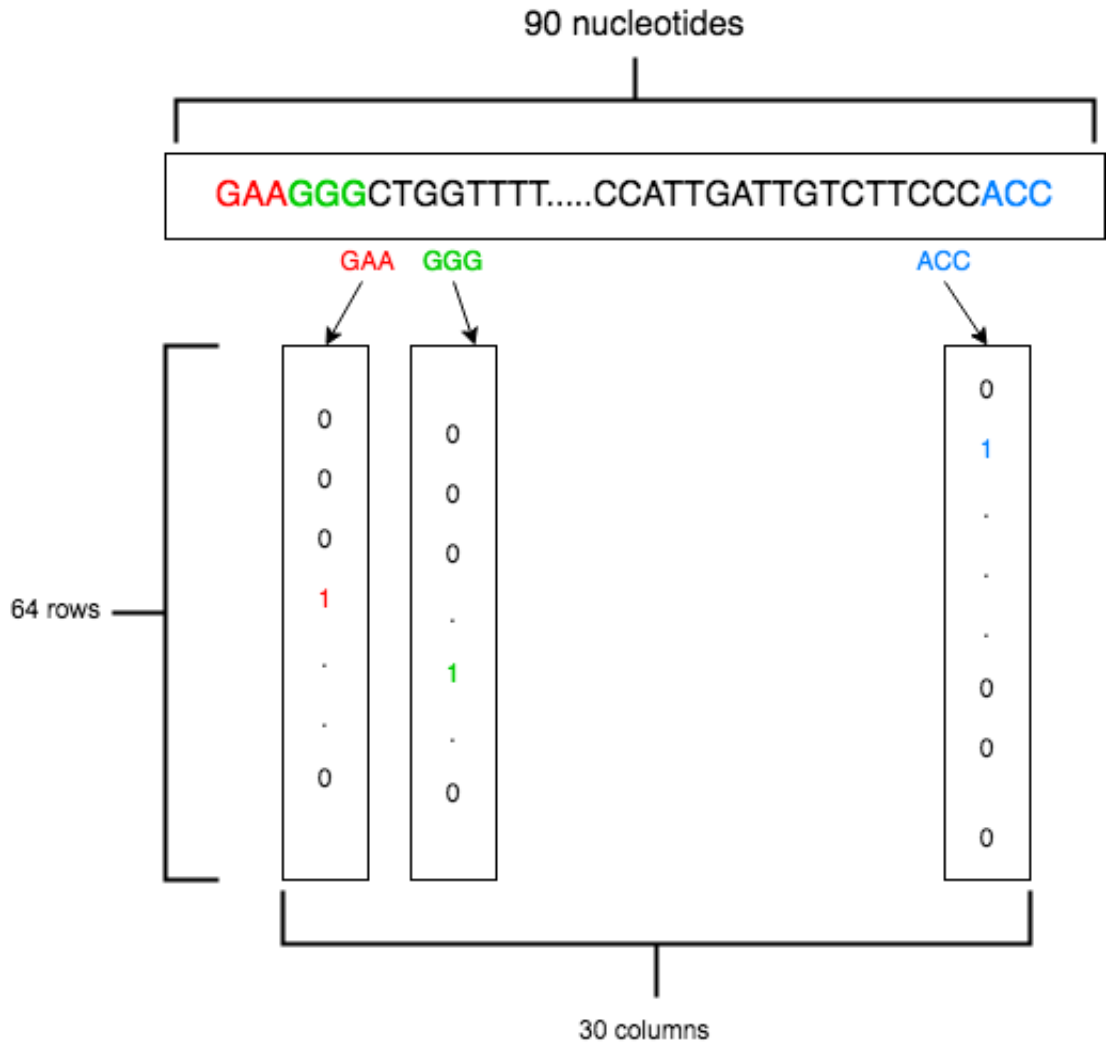

Figure 2.3 One-hot codon encoding of an input sequence of length 90.

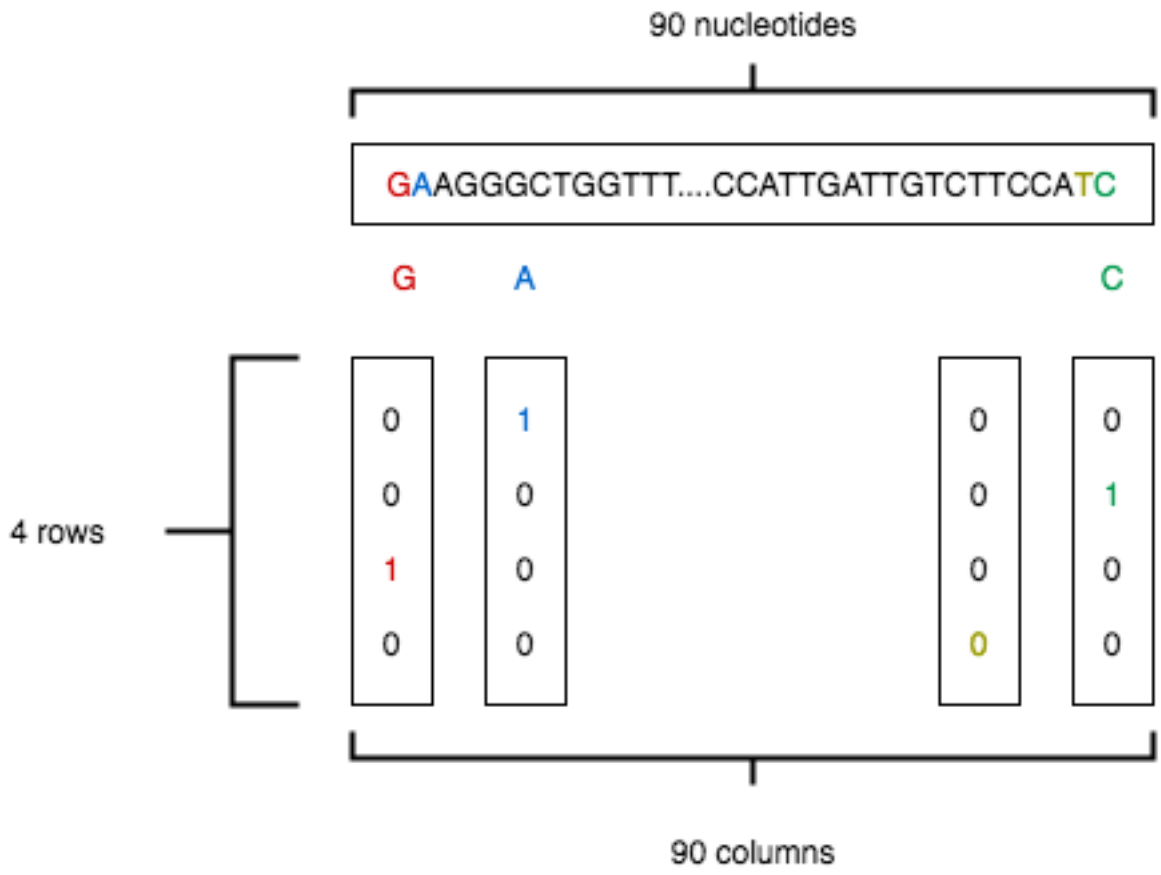

Figure 2.4 One-hot nucleotide encoding of an input sequence of length 90.

2.3 Experiments with Simulated Data

To investigate the upper bounds of the expected accuracy of neural network models for predicting pause sites, the minimum number of samples required to train a neural network which would provide good accuracy and the effect of errors in the labels on the performance of neural network models, a labeled data set was created. This dataset was generated from *Arabidopsis thaliana* which was downloaded from NCBI SRA. The dataset was labeled based on the presence of a known motif [26]. Each row in the dataset consisted of 183 nucleotides with a concomitant label indicating the presence of the motif.

We used one hot-codon encoding for the input representation of the neural network. With the one-hot codon encoding, each codon is encoded as a binary vector of length 64, as each codon can have 64 possible values. Since each codon has 3 nucleotides, this essentially gives us a matrix of 61 x 64. Experiments were carried out with neural network accuracy for sample sizes varying between 9k samples to 114k samples. We used a 80-20 split as the training-validation/test split. We also performed a 5-fold cross validation and took the average accuracy. This was plotted to show the variation of accuracy of the neural network with the sample size.

We performed another experiment with the same set of samples, i.e. between 9k and 114k samples, but with errors in the form of misclassified labels to simulate technical as well experimental biases. We misclassified somewhere between 10% to 45% samples in each dataset and plotted the accuracy of the neural network to show the variation of accuracy of the neural network with error rate.

We also compare the accuracy of our model with the accuracy of Ribosome Stalling Estimator (ROSE) [42]. The ROSE model consists of Convolutional Neural Networks as the backbone of the network. It's input layer, which is a one-hot codon representation, is connected to 3 CNN layers stacked in parallel, which is finally connected to a fully connected dense layer as an output.

2.3.1 Neural Network Architecture for Experiments on Simulated Data

Our neural network architecture was based on recurrent neural networks (RNN). We used a Long Short Term Memory layer, which is a type of RNN layer. The input vector encoded representation, which is a 61 x 64 matrix, is fed into an LSTM layer. We add a dropout layer after this to prevent overfitting of the network. After this we add a bidirectional layer to capture additional context and enable learning information in both the directions of the sequence. Table 2.1 describes the different parameters of the layers of this model. Figure 2.5 shows a visualization of this model.

Table 2.1 Neural network architecture for simulated data.

	Layer	**Description**
1	Input	64 x 30 one-hot codon representation
2	LSTM	100 units
3	Dropout	0.5 probability
4	Dense	1 class

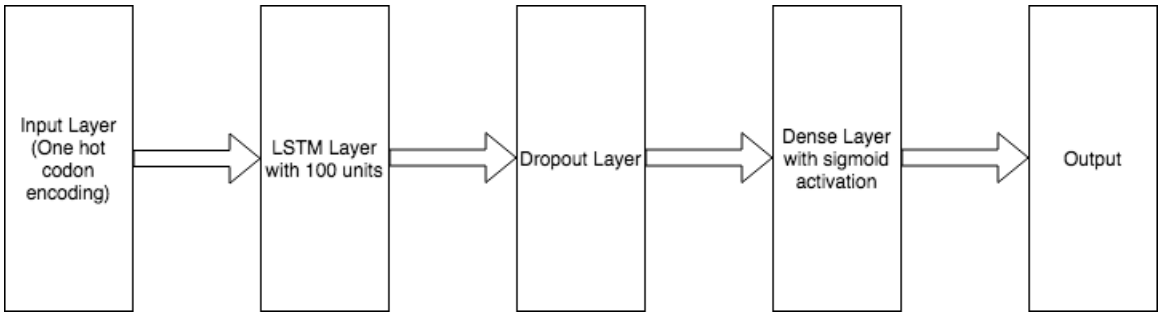

Figure 2.5 Neural network architecture for experiments on simulated data.

2.4 Stacked Convolutional Neural Network

We used a stacked CNN architecture for predicting pause sites in ribo-Seq data (See section 1.4.5 for details on CNN). Filters/kernels of different sizes in CNN can act as motif detectors [42]. Also, because of the property of convolution operation, we can have parameter sharing which enables sharing of features in different parts of the input sequence. This is useful for detecting low level features like sequence motifs or even high level features which could be more complex.

The input layer is connected to three convolutional layers with filters of sizes 32, 64 and 128 stacked on top of one another. The kernel size of the convolutional layers is 3. ReLU is the activation function for all of the layers. This is followed by a fully connected dense layer. We use "binary cross-entropy" as the loss function. We also introduce dropout with a value of 0.3 after the convolutional layers. The model was tuned with the Bayesian optimization method for hyperparameter tuning [37]. Figure 2.6 shows a visualization of different layers of this model.

2.4.1 Codon Encoding

The input layer is a one-hot encoding representation of codons. The rest of the architecture is as described above. Table 2.2 gives the different layers of the stacked CNN network and their parameters.

Table 2.2 Description of stacked CNN architecture with codon encoding.

	Layer	**Kernel Size**	**Description**
1	Input representation (One-hot codon encoding)		
2	Conv1D	3	32 kernels
3	Dropout		probability 0.3
4	MaxPool1D	2	
5	Conv1D	3	64 kernels
6	Dropout		probability 0.3
7	MaxPool1D	2	
8	Conv1D	3	128 kernels
9	Dropout		probability 0.3
10	MaxPool1D	2	
11	Softmax		1 output

2.4.2 Nucleotide Encoding

Table 2.3 shows the architecture of the stacked CNN network with a different input layer, which uses nucleotide encoding. The overall architecture is similar to the one with one-hot codon encoding as the input layer. However, we had to re-tune some hyperparameters based on the new input layer. Specifically, the dropout layer probability is 0.4 in this network. Also, the kernel size is 11 in the first layer, 5 in the second layer and 11 in the third layer.

Table 2.3 Description of stacked CNN architecture with nucleotide encoding.

	Layer	**Kernel Size**	**Description**
1	Input representation (One-hot nucleotide encoding)		Input shape of 90 x 4
2	Conv1D	11	32 kernels
4	MaxPool1D	2	
5	Conv1D	5	64 kernels
6	Dropout		probability 0.4
7	MaxPool1D	2	
8	Conv1D	11	128 kernels
9	Dropout		probability 0.4
10	MaxPool1D	2	
11	Softmax		1 output

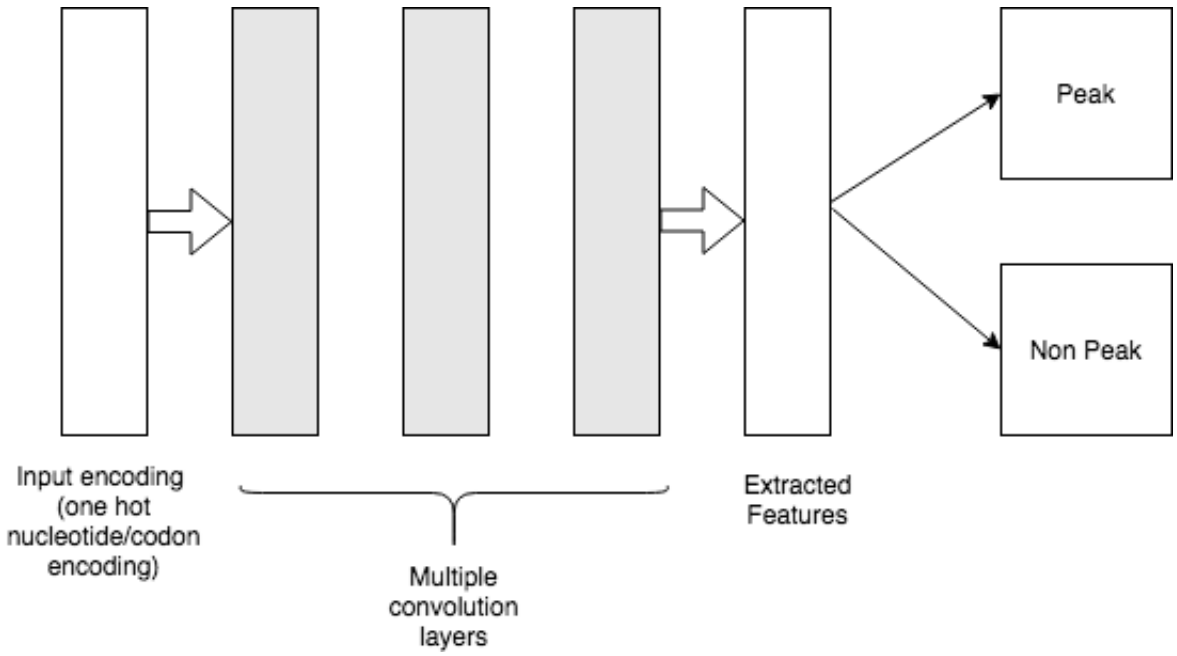

Figure 2.6 Stacked CNN architecture.

2.5 GRUs with Bidirectional Layer

Recurrent Neural Networks (RNNs) are extremely popular for processing sequential data (see section 1.4.6 for details on RNNs). We use Gated Recurrent Units, a type of RNN. The one-hot encoded representation of the data forms the input layer of the model. The input layer is connected to an RNN layer of 100 GRU cells. After this, we add a dropout layer for regularization. The value of dropout is 0.4, meaning that after each epoch we randomly choose 40% of the total neurons and set their value to 0. The values of the remaining neurons are boosted to account for the loss. The next layer added to the model is a fully connected dense layer. We use the ADAM optimization function which combines both "momentum" for improving stochastic gradient descent as well as RMSProp optimization function. We analyzed the performance of the model with two different input representations — one hot nucleotide encoding and one hot codon encoding. Table 2.4 describes the layers of this model and their hyperparameters. Figure 2.7 shows a visualization of the layers of this model.

Table 2.4 Description of the GRU-based neural network.

	Layer	**Description**
1	Input representation (One-hot codon\One-hot nucleotide encoding)	Codon encoding: 64 x 30 Nucleotide encoding: 90 x 4
2	GRU layer	100 hidden units
3	Dropout	probability 0.4
4	Bidirectional GRU layer	100 hidden units
5	Dropout	probability 0.4
6	Sigmoid	2 outputs

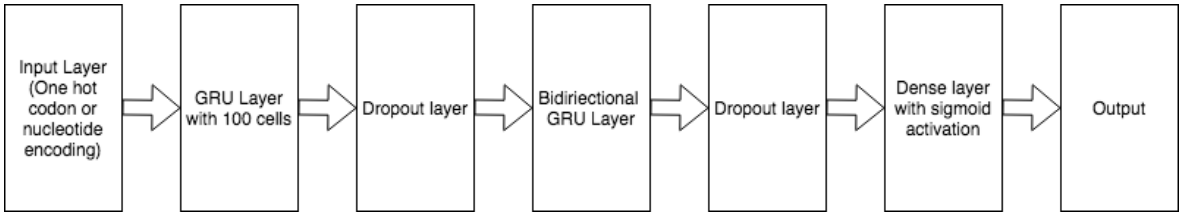

Figure 2.7 Architecture of the GRU-based neural network.

2.6 Attention Networks

The input sequence is converted into a one-hot encoding representation for each nucleotide or codon. The input layer is connected to a GRU layer with 100 hidden units. This is connected to a bidirectional layer which is capable of processing input in the forward and the reverse direction. This layer is then connected to an attention layer. Finally, the attention layer is connected to a fully connected dense layer which produces the output. We use dropout with a value of 0.4 for regularization. Table 2.5 describes the layers and hyperparameters of this network.

Table 2.5 Description of GRU based neural network with Attention mechanism

	Layer	**Description**
1	Input representation (One-hot codon\One-hot nucleotide encoding)	Codon encoding: 64 x 30 Nucleotide encoding: 90 x 4
2	GRU layer	100 hidden units
4	Dropout	probability 0.4
5	Bidirectional GRU layer	100 hidden units
6	Dropout	probability 0.4
7	Attention with context	
8	Sigmoid	2 outputs

CHAPTER

3

RESULTS

The following chapter discusses the results of our experiments. We first discuss the results of the simulated data experiments, which were essential to gauge the performance of neural networks to detect pause sites and to prioritize different architectures. We then describe the results of our experiments using different neural network architectures to detect pause sites in ribo-seq data.

3.1 Experiments with Simulated Data

To understand the upper bounds of the expected accuracy of neural network models for prediction of pause sites we performed experiments on simulated data. We assumed that ribosome pausing is due to the presence of sequence motifs. Such motifs have been identified in C.elegans [36]. We generated artifical data by using mRNA fragments of 183 nucleotides from *Arabidopsis thaliana*. We implanted a motif: "TTTGA" in 50 % sequences and labeled those sequences as "Class 1: Pause Sites" and the remaining sequences as "Class 2: No Pause Sites". We then trained and tested our neural network model on a balanced data set. We divide the samples into a 80-20 split for training and testing.

We first investigated the relationship between number of samples used for training and the accuracy of the model by checking the performance of our model with training datasets of different sizes. This was done to get a lower bound on the minimum number of samples that would be required for training the model to get a high accuracy. Our model was based on LSTM, which is a type of RNN. The details of this model can be found in chapter 2 in section 2.3. We do the same analysis with the ROSE (Ribosome Stalling Estimator) model to compare the performance of a CNN-based model with an RNN-based model and to see the merits in investigating RNN-based models on nucleotide sequences.

Figure 3.1 shows the accuracy of our model and that of the ROSE model in predicting sequences with

known motifs in them, as the number of samples in the dataset increases. For data sets with less than 20k samples, we see a higher degree of variance for our models. With more than 20k samples, the model quickly converges to close to 100% accuracy without much variance in the prediction accuracy. The ROSE model shows a higher variance in prediction accuracy, though the accuracy increases linearly with increase in the number of samples in the training dataset.

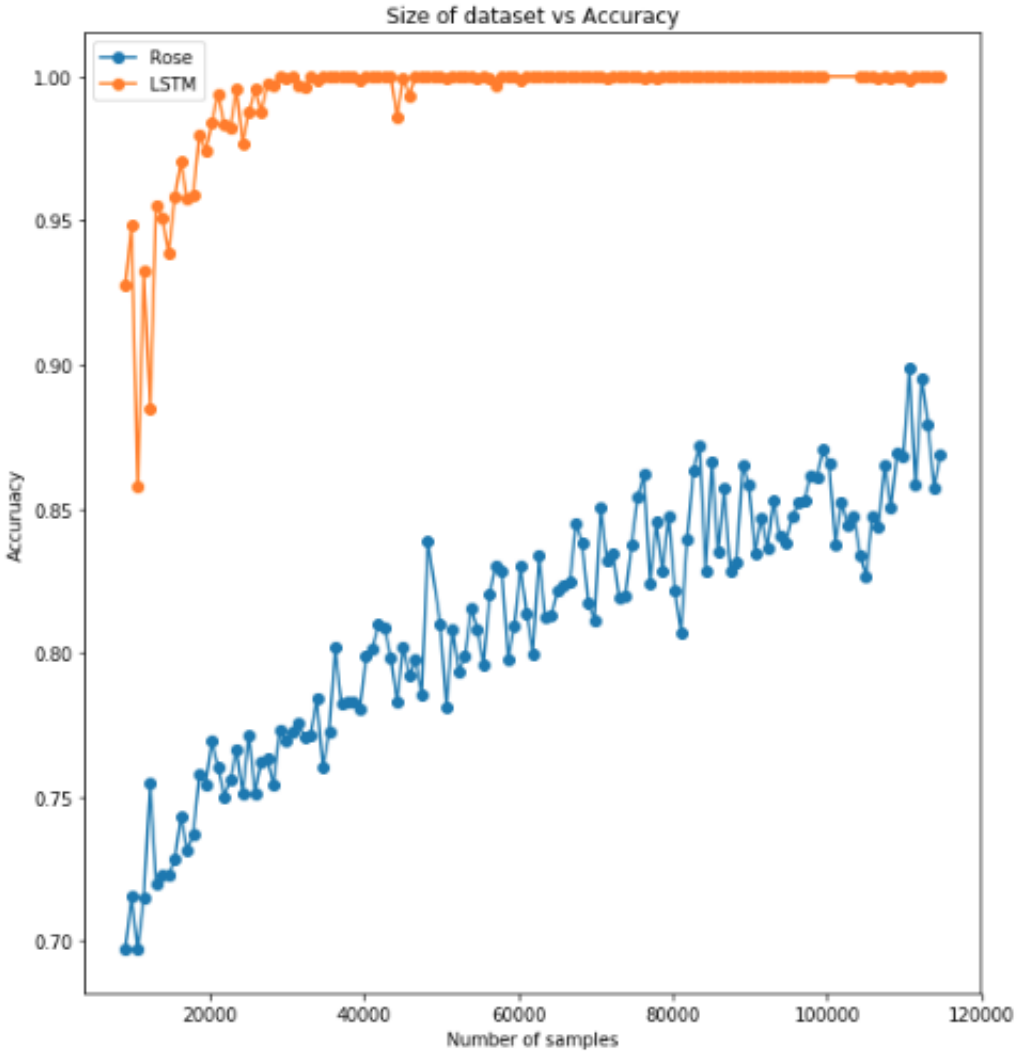

Figure 3.1 Change in accuracy of LSTM-based model and ROSE model with increase in number of samples in the input dataset.

We also investigated the accuracy of our model in the presence of errors in training labels. We introduced errors — from 5% to 45% with a step size of 5% — in the training dataset in the form of misclassifications of labels. This was done to simulate technical as well as experimental biases in the dataset and to gauge how resilient neural network models would be in the presence of labeling errors.

Figure 3.2 shows the accuracy of our model in detection of nucleotide sequences with motifs, and how

this accuracy changes with increase in labeling error rate in in the training dataset. The accuracy of the model increases with increase in the number of samples available in the training data set. With as few as 10k samples our simple model was able to predict sequences with known motifs with 90% accuracy. The accuracy of the model decreases for samples of all sizes as the percent error in the labels in training dataset increases. We found that increase in number of samples in the training dataset makes the neural network models fairly resilient to errors in labeling in the training dataset. For datasets with around 100k samples the accuracy drops only 15% with as many as 40% errors in the labels of the training dataset. This means that our network was able to accurately detect sequences containing motifs in the testing data set even though there were errors in the labels of the training dataset. With 15-20% errors in labels our model was able to correctly classify sequences containing motifs with an accuracy of 90% for training set sizes of more than 50k samples. Sample sizes less than 70k show a linear decline in the accuracy with increased percent error in the form of misclassified labels.

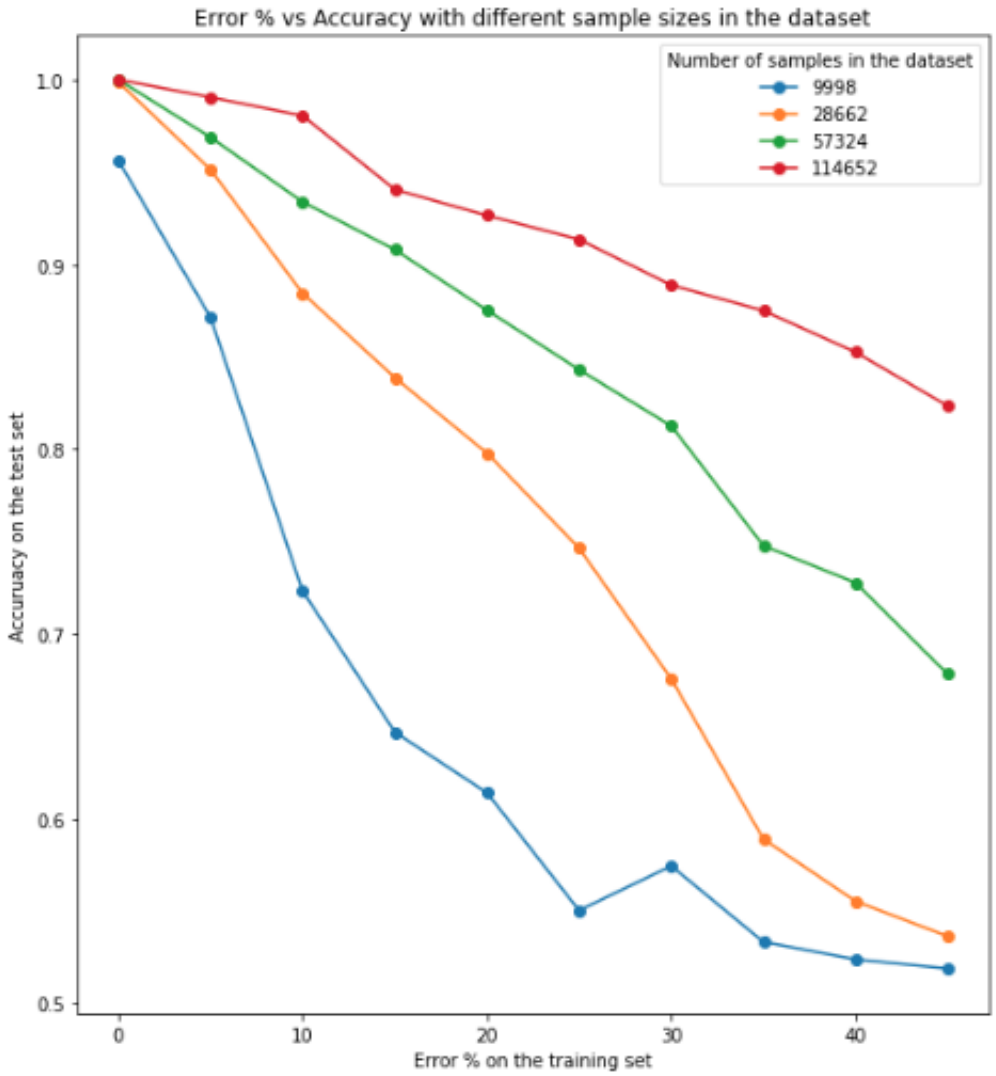

Figure 3.2 Change in accuracy of the LSTM model with increasing error rate in the training data in the form of misclassified labels.

3.2 Experiments with Ribo-Seq data

To predict pause sites in ribo-seq data, we use a set of classifiers using different neural network approaches. The ribo-seq data for *Arabidopsis thaliana* was processed to remove technical and experimental biases and labeled to indicate "peaks" and "non-peaks". The details of the data processing and labeling can be referred to in section 2.1. We used a balanced dataset for training, with 14k samples having the labeled "peak" and the remaining 14k samples label of "non-peak". We also performed experiments to investigate the effect of different encoding mechanisms — one-hot nucleotide encoding and one-hot codon encoding — on the performance of the neural network (see section 2.2). Each input sequence to our network consisted of 90 nucleotides with the P-site at its center. We chose 90 nucleotides because it produced the highest accuracy in our models.

3.2.1 Stacked CNN Network

To investigate the performance of CNN-based networks in predicting ribosomal peaks, we used a stacked CNN network with one-hot codon encoding representation in the input layer. The details of the neural network architecture used for this experiment can be found in Chapter 2 in section 2.4. The ribosome profiling data along with corresponding RNA-seq samples from *Arabidopsis thaliana* were obtained from NCBI SRA. The details of this data set can be found in section 2.1 in chapter 2. We used a 80-10-10 split of the data set for training, testing and validation of our model.

The graph below shows the performance of the network on the training and validation set. We see a small amount of overfitting, but we couldn't eliminate it by using regularization. We chose to limit the number of epochs to train the network at 60, as we see an increase in overfitting on the training data. We trained on 22,548 samples and validated on 2,818 samples. The size of the test data set was 2818 samples. Table 3.1 shows the performance of the network on the test dataset. We found that the accuracy of this network was 76.97% with a precision score of 0.77 and a recall score of 0.77.

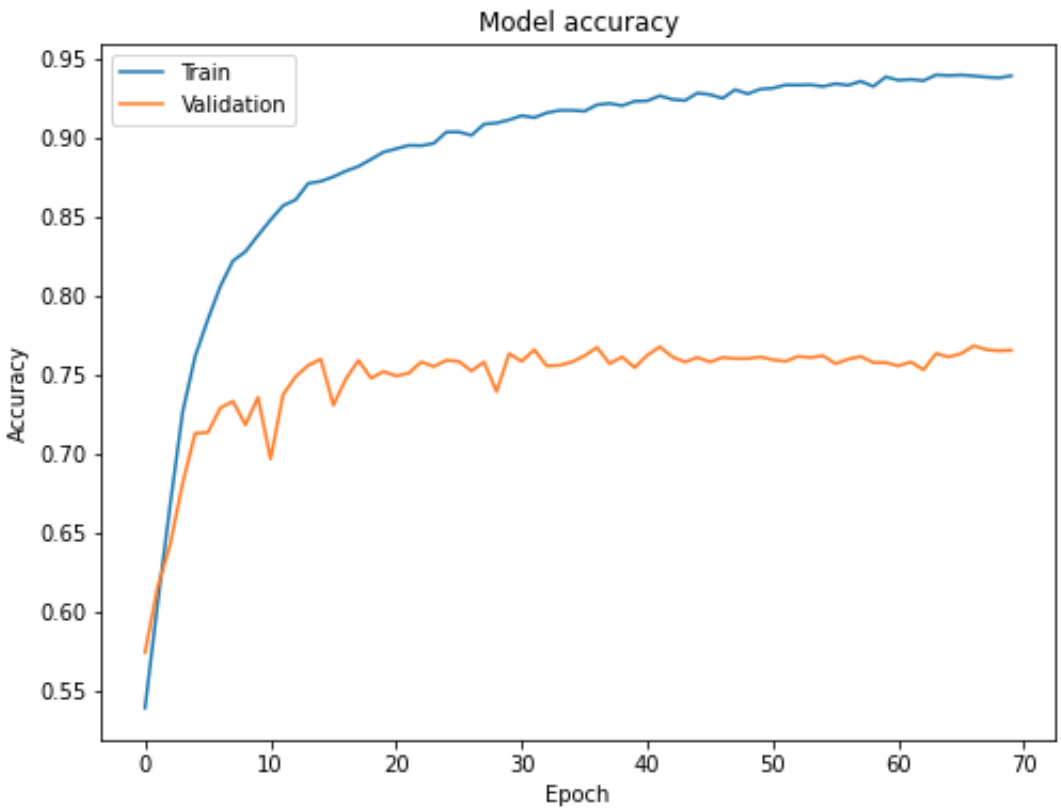

Figure 3.3 Change in accuracy with increase in number of epochs on Train and Validation dataset for stacked CNN network with codon encoding.

Table 3.1 Results for stacked CNN with codon encoding.

Accuracy	Precision	Recall
76.97	0.77	0.77

To study the effect of input encoding on the stacked CNN architecture, we performed an experiment to predict peaks in ribo-seq data with a different input representation, i.e. one hot-nucleotide encoding instead of one-hot codon encoding. The graph below shows the performance of the network on the training and validation set. We don't see substantial overfitting in this network as compared to the network with codon encoding. The maximum validation accuracy for this network is 86%. We trained on 22,548 samples and validated on 2,818 samples. The size of the test dataset was 2.8k samples. Table 3.2 shows the performance of the network on the test dataset. We found that the accuracy of this network was 85.45% with a precision score of 0.85 and a recall score of 0.86.

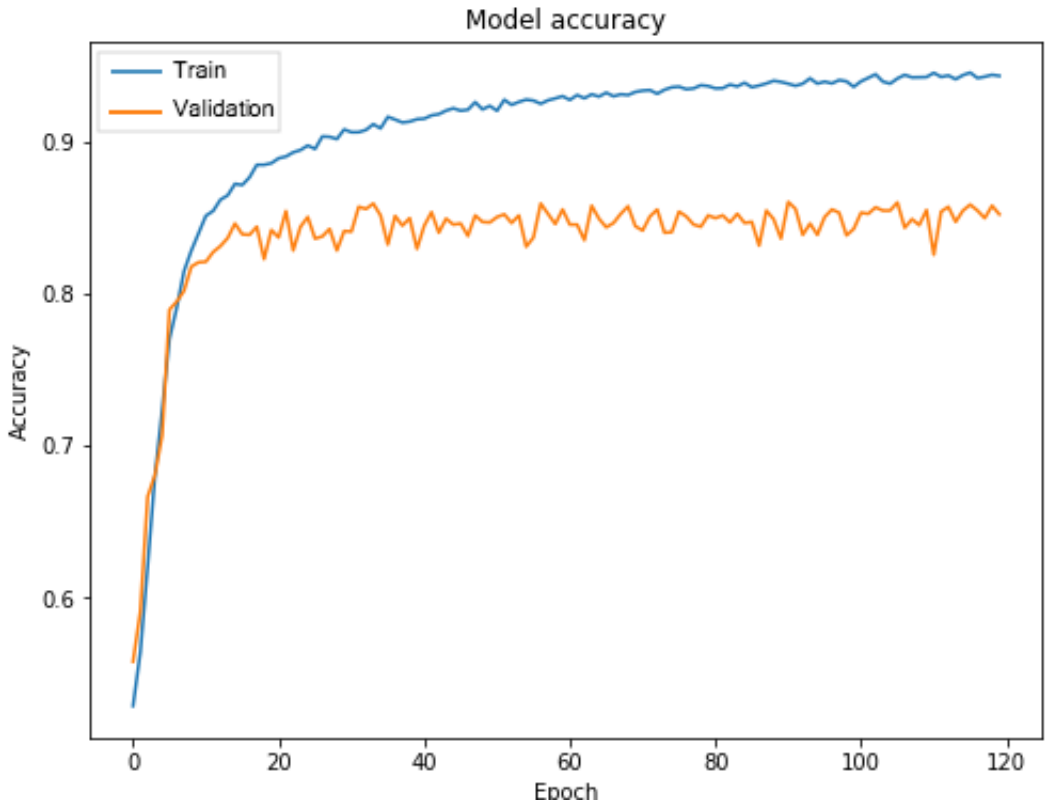

Figure 3.4 Change in accuracy with increase in number of epochs on Train and Validation dataset for stacked CNN network with nucleotide encoding.

Table 3.2 Results for stacked CNN with nucleotide encoding.

Accuracy	Precision	Recall
85.45	0.85	0.86

3.2.2 GRU-based Network

Based on our experiments with simulated data, we performed experiments with an RNN-based network to predict peaks in ribo-seq data. We used a GRU-based network with one-hot codon encoding as the input representation in this experiment to predict peaks in ribosome profiling data. The network consists of the input layer followed by a GRU layer with 100 cells connected to a bidirectional GRU layer and followed by a dense softmax layer with one output predicting if the input nucleotide sequence belongs to the peak class or the non-peak class. A detailed architecture of this model can be referred to in Chapter 2. The labeled dataset used for the experiment is the same as section 3.1 with an 80-10-10 split of the dataset for training, testing and validation of the model.

The graph below shows the performance of the network on the training and validation set. We also see some overfitting here. The network was trained with 22,548 samples and validated on 2,818 samples. The size of the test data set was 2.8k samples. Table 3.3 shows the performance of the network on the test dataset. We found that the accuracy of this network was 81.32% with a precision score of 0.81 and a recall score of 0.81.

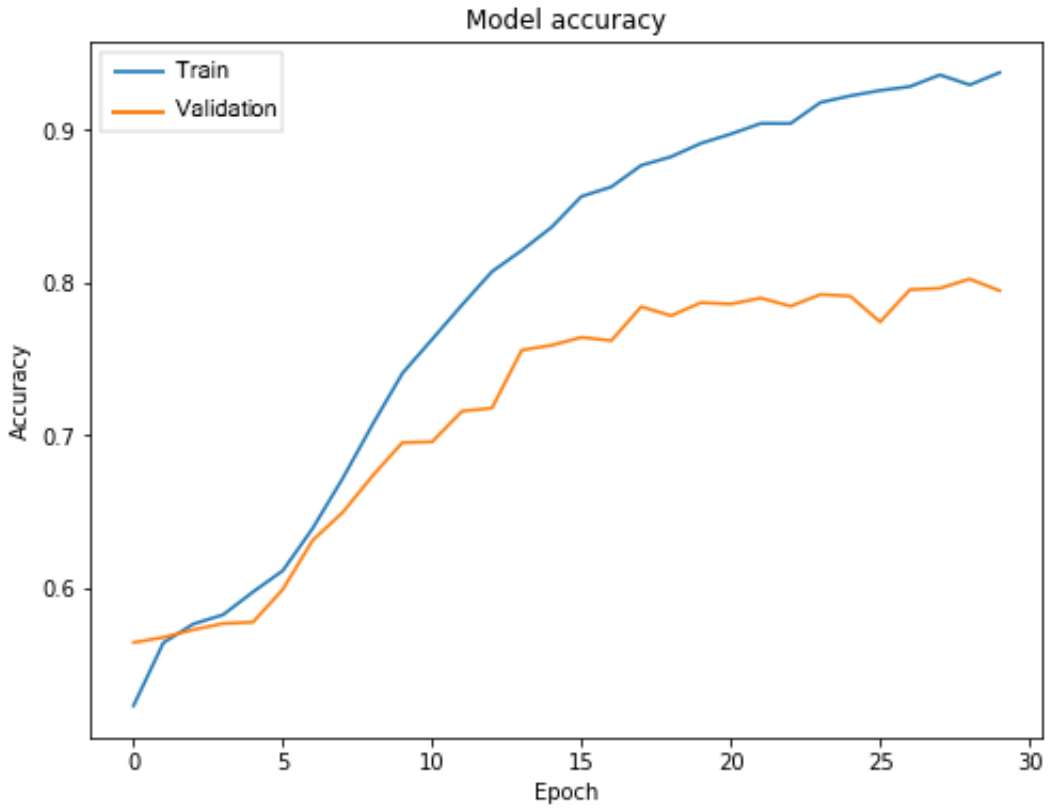

Figure 3.5 Change in accuracy with increase in number of epochs on Train and Validation dataset for GRU-based network with codon encoding.

Table 3.3 Results for GRU-based network with codon encoding.

Accuracy	Precision	Recall
81.32	0.81	0.81

To study the effect of input representation on prediction of peaks using the GRU-based architecture, we performed an experiment with one-hot nucleotide encoding as the input layer instead of one-hot codon encoding.

The graph below shows the performance of the network on the training and validation set. Similar to previous nucleotide encoding in stacked CNNs, we don't see significant overfitting here. Table 3.4 shows the performance of the network on the test data set. The size of the test data set was 2.8k samples. We found that the accuracy of this network was 88.18% with a precision score of 0.88 and a recall score of 0.88.

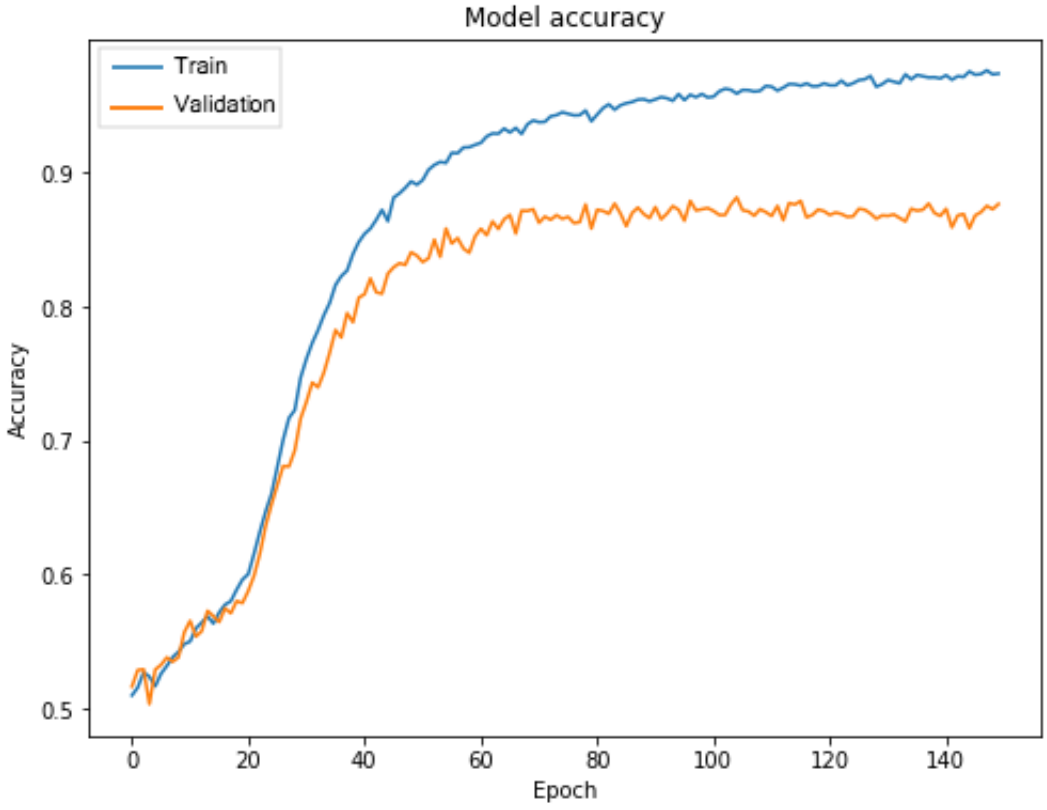

Figure 3.6 Change in accuracy with increase in number of epochs on Train and Validation data set for GRU-based network with nucleotide encoding.

Table 3.4 Results for GRU-based network with nucleotide encoding.

Accuracy	Precision	Recall
88.18	0.88	0.88

3.2.3 GRU-based Network with Attention Layer

Attention networks [2] have been used in state-of-the-art machine translation systems. Attention networks enable assigning higher weights to relevant parts of the input vector which can improve performance of the neural network. To test if an attention layer would improve the accuracy of our network, we performed an experiment by adding an attention layer to the GRU-based network described in the section above.

We used a GRU-based network with one-hot codon encoding as the input representation in this experiment with an attention layer. The network consists of the input layer, followed by a GRU layer with 100 cells, connected to a bidirectional GRU layer, which in turn is connected to an attention layer, and followed by a dense softmax layer, with one output predicting if the input nucleotide sequence belongs to the peak class or the non-peak class. A detailed architecture of this model can be referred to in section 2.6 of Chapter 2. The labeled dataset used for the experiment is the same as section 3.1 with an 80-10-10 split of the data set for training, testing and validation of the model.

The graph below shows the performance of the network on the training and validation set. Again, we see some overfitting here which is consistent with other models when codon encoding is used in the input layer. The network was trained on 22,548 samples and validated on 2,818 samples. The size of the test dataset was 2.8k samples. Table 3.5 shows the performance of the network on the test dataset. We found that the accuracy of this network was 78.98% with a precision score of 0.79 and a recall score of 0.79.

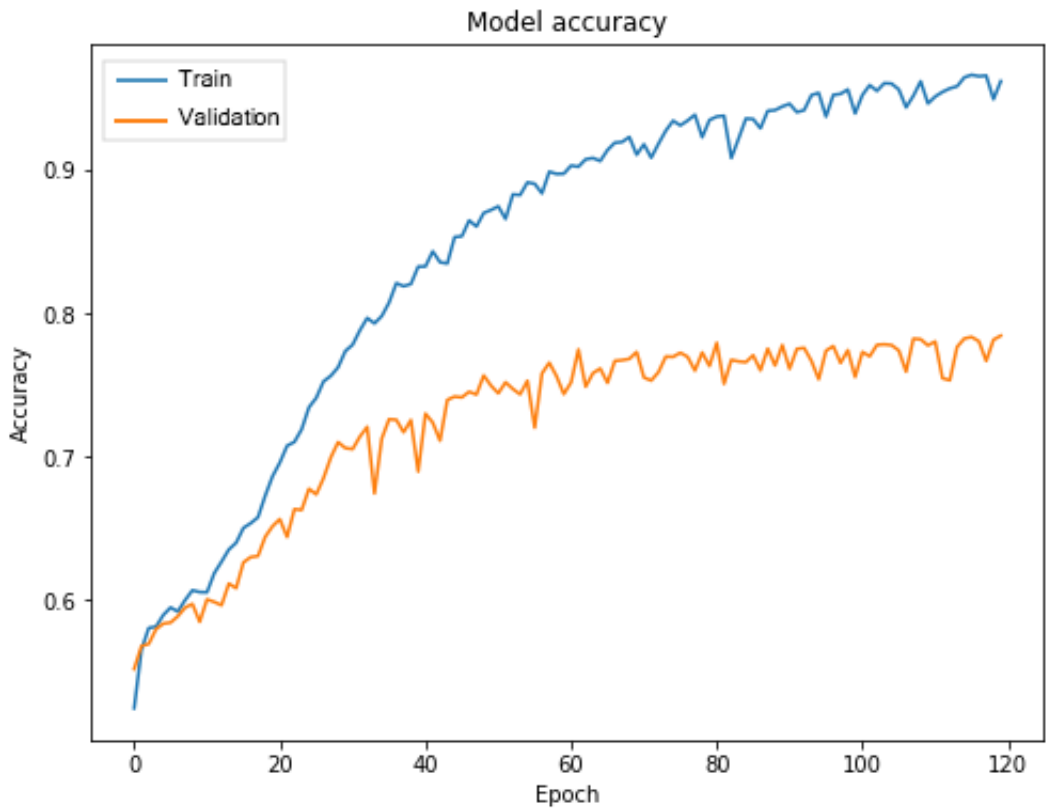

Figure 3.7 Change in accuracy with increase in number of epochs on Train and Validation data set for GRU-based network with attention layer and codon encoding.

Table 3.5 Results for attention-based GRU network with codon encoding.

Accuracy	**Precision**	**Recall**
78.98	0.79	0.79

To test the effect of one-hot nucleotide representation on the accuracy of the GRU-based model with attention mechanism, we changed the input layer to nucleotide encoding and predicted peaks in ribo-seq data.

The graph below shows the performance of the network on the training and validation set. We trained on 22,548 samples and validated on 2,818 samples. The maximum validation accuracy during training was 89.65%. The size of the test dataset was 2.8k samples. Table 3.6 shows the accuracy, precision and recall on the test dataset.

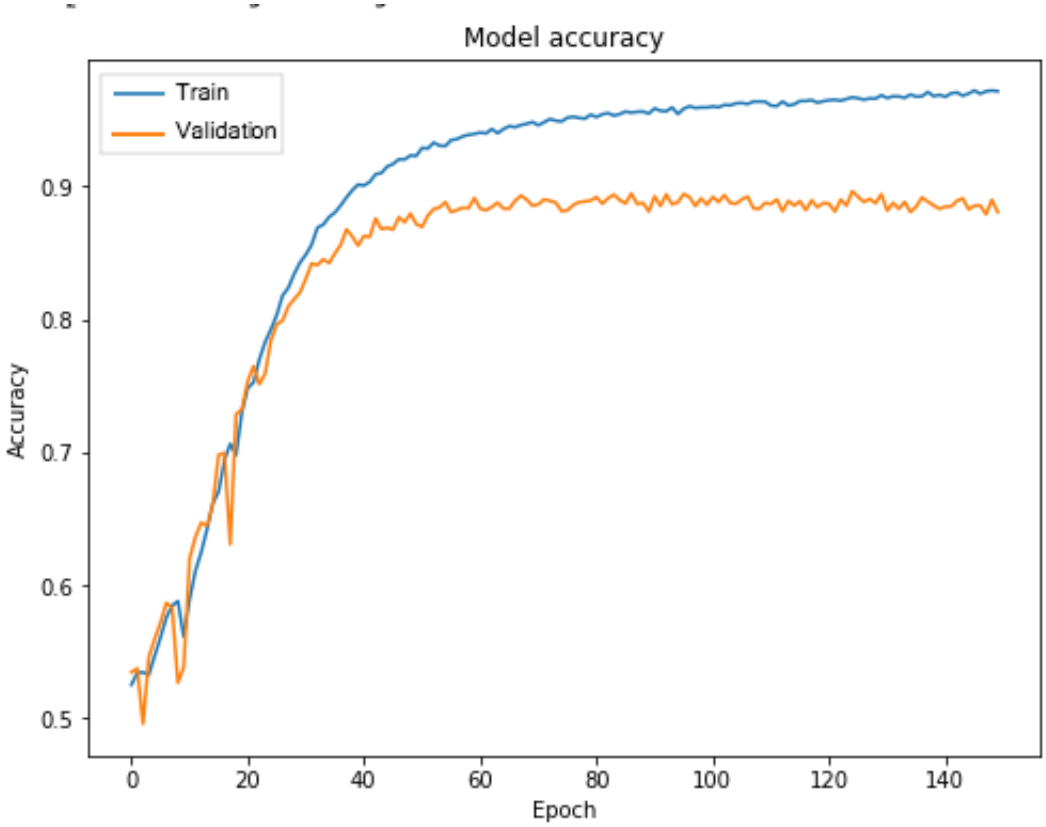

Figure 3.8 Change in accuracy with increase in number of epochs on Train and Validation dataset for GRU-based network with attention layer and nucleotide encoding.

Table 3.6 Results for attention-based GRU network with nucleotide encoding.

Accuracy	Precision	Recall
88.97	0.89	0.89

3.2.4 ROSE Model

To compare the performance of the ROSE model (Ribosome Stalling Estimator) by Zhang et al. with the models that we developed, we use the ROSE model to predict peaks in ribosome profiling data. The ROSE model consists of a parallel CNN architecture with 3 different kernel sizes corresponding to short (5-7), medium (8-9) and long (10-13) ranges, respectively. These three kernels act as local motif detectors. Following the CNN layers the dimension of the output is reduced by a max pooling layer of size 2. They use one-hot codon encoding in the input layer of their network. The dataset used for this experiment is the same as described in section 3.1.

The graph below shows the performance of the ROSE model (Ribosome Stalling Estimator) network on the training and validation set. The network was trained on 22,548 samples and validated on 2,818 samples. The size of the test dataset was 2.8k samples. Table 3.7 shows the performance of the network on the test dataset. We found that the accuracy of this network was 82.65% with a precision score of 0.83 and a recall

score of 0.83.

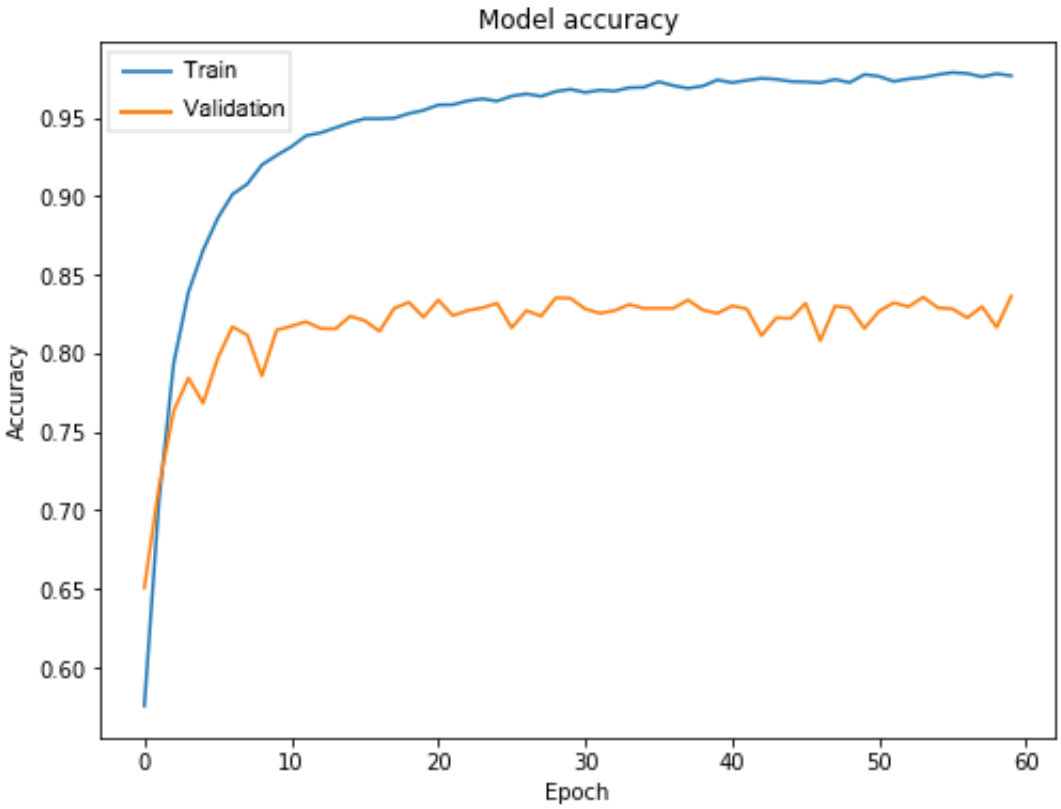

Figure 3.9 Change in accuracy of ROSE model with increase in number of epochs for Train and Validation dataset.

Table 3.7 Results for ROSE model.

Accuracy	Precision	Recall
82.65	0.83	0.83

3.3 Summary

3.3.1 Summary of Experiments on Motif Detection with Simulated Data

Table 3.8 compares the ROSE model and an LSTM-based model described in section 3.1. The LSTM-based model yields 100% accuracy for the problem of motif detection with only 28k samples. The ROSE model required a large number of samples (114k) to reach an accuracy of 90%. We also see a high amount of variance in the prediction of ROSE.

Table 3.8 Results for comparison of LSTM-based network and ROSE for motif detection in simulated data.

	Model	**Number of samples in model with best accuracy**	**Best Accuracy**
1	LSTM	All datasets with more than 28k samples	100%
2	ROSE	114k samples	90%

Table 3.9 describes the accuracy of the LSTM-based model with different numbers of samples in the dataset and the effect of introducing errors in the training labels as described in section 3.1. We found that a high number of samples in the training dataset makes the neural network models fairly resilient to errors in the labels in the training dataset, meaning that the neural network model was able to capture the features from the nucleotide sequences accurately to detect motifs.

Table 3.9 Summary of experiments on simulated data for measuring accuracy of the network with increasing percent error in labeling.

	Model	**Number of samples**	**Error in labeling**	**Accuracy**
1	LSTM	9998	0-45%	Accuracy is 95% when there are no errors. Accuracy gradually decreases with decrease in error. For error more than 25% accuracy is ∼50%
2	LSTM	28662	0-45%	Accuracy is 100% when there are no errors. Accuracy gradually decreases with decrease in error. For error more than 35% accuracy is ∼55%
3	LSTM	57324	0-45%	Accuracy is 100% when there are no errors. Accuracy gradually decreases with decrease in error. The accuracy does not drop as sharply and is ∼65% with 45% errors in the training labels.
4	LSTM	114652	0-45%	Accuracy is 100% when there are no errors. Accuracy gradually decreases with decrease in error. The accuracy does not drop sharply and is still 85% with 45% errors in the training labels.

3.3.2 Summary of Experiments with Ribo-Seq Data for Pause Site Prediction

Table 3.10 shows the performance of different neural network models with respect to accuracy, precision and recall. We found that our GRU model with attention layer and one-hot nucleotide encoding for the input layer gave the best accuracy, at ~89%. This was followed by a GRU-based model with ~88% accuracy, followed by stacked CNN-based model with ~85% accuracy. The ROSE model (Ribosome Stalling Estimator) [42] had an accuracy of 82.65%. Also, we see that models with one-hot nucleotide encoding as the input layer perform better than models with codon encoding as the input layer with an improvement of 5-10% in the prediction accuracy.

Table 3.10 Summary of results of different models on ribo-seq data

	Model	**Accuracy**	**Precision**	**Recall**
1	ROSE (Ribosome Stalling Estimator)	82.65%	0.82	0.82
2	GRU with bidirectional layer and codon encoding	81.34%	0.81	0.81
3	GRU with bidirectional layer and nucleotide encoding	88.20%	0.88	0.88
4	Stacked CNN with codon encoding	78.95%	0.79	0.79
5	Stacked CNN with nucleotide encoding	85.45%	0.85	0.86
6	GRUs with attention layer and codon encoding	78.98%	0.79	0.79
7	GRU with attention layer and nucleotide encoding	88.97%	0.89	0.89

CHAPTER

4

DISCUSSION, FUTURE WORK AND CONCLUSION

4.1 Discussion

This thesis investigates various neural network architectures to detect ribosome pause sites in *Arabidopsis thaliana*. Since there are few experimentally confirmed ribosome pause sites, we used peaks in ribo-seq data as experimental evidence for ribosome pause sites.

We started with a simulation study in which we assume that pause sites occur at the locations of a simple sequence motif. This is an extreme simplification. Likely, true ribosome pause sites are characterized by more complex motifs and/or 3-dimensional mRNA folding structures. Nevertheless, this experiment provided us with a simple tool to prioritize neural network architectures, it yielded an important baseline for the number of data points that might be required for training, and it gave us evidence that even with imperfect data (see [18] for a review of technical or experimental biases, errors in sequencing, ambiguous mapping, etc. in ribosome footprinting) deep learning approaches might produce correct predictions. We found that with sufficiently large training sets our network was resilient to up to 40% misclassified labels.

After switching to 'real' ribo-seq data we found that different sequence encoding mechanisms, such as nucleotide encoding and codon encoding, as the input layer of the neural networks has a strong effect on the performance of the neural network with respect to the accuracy of the models, the variance in accuracy and also the time required to train such models. One-hot codon encoding in the first layer of the neural network shows some overfitting and high variance. In contrast, the one-hot nucleotide encoding shows higher prediction accuracy by approximately 8%-10%. The increased accuracy of nucleotide encoding suggests that

the neural network models are capable of capturing mRNA sequence information that is not present in the codon encoding or amino acid sequence of encoded proteins. One of the reasons for this could be that the sparse representation of one-hot codon encoding makes it harder for the neural network model to learn relevant features from the sequences as compared to nucleotide encoding, which is denser.

Among the different neural network architectures that we compared, Gated Recurrent Units (GRUs) with attention layer, a variation on recurrent neural networks, gave us the best accuracy, at 89%. The GRUs are suited for sequential data as they can retain information from the previous hidden states which prevents the vanishing gradient problem [15]. The attention mechanism further improves the performance of the network by re-weighting the parts of the input structure that are more important. We hypothesize that these are the reasons why this architecture yielded the best performance. This architecture had an accuracy improvement of about 7% over the ROSE tool.

4.2 Future Work

There are several avenues for future research. Our research relies on the assumption that ribosome stalling can be detected by peaks in ribosome profiling data. However such peaks can also arise due to several other factors, for example mRNA-binding proteins, mRNA folding, PCR amplification artifacts etc. (see [21] for a more thorough review). An important extension of our research is to integrate these factors in our classification models. Another direction is to design models that predict a numeric ribosome pause score instead of a binary class label. This would require modeling the peak prediction problem as a regression problem to predict the numeric scores at each position of the input sequence instead of classifying the input into labels.

Finding biological significance from the predicted peaks is an important topic with which we are currently engaged. Ribosome stalling can result from several factors, including presence of motifs, occurrence of certain codons, presence of mRNA secondary structures, etc. We are trying to look at conserved motifs in the sequences that are predicted to have peaks. We are also trying to relate presence of peaks with codon frequency to see if there is a correlation between frequency of some codons upstream or downstream of the P-site. Ribosome stalling can also occur due to the presence of mRNA secondary structures because the mRNA has to be unfolded as it is translated. A possible future avenue could be labeling of peaks due to presence of mRNA secondary structures in our training data set which would enable our models to detect pausing due to mRNA secondary structures.

4.3 Conclusion

In this research, we worked on detecting ribosomal pausing from mRNA sequences. We focused on training deep learning models based on data obtained from ribosomal footprinting experiments and explored different neural network architectures that can be used to achieve this. Through our experiments using simulated data for motif detection we established that neural network-based models are considerably resilient to errors given enough samples in the training data.

We found that the encoding mechanism, such as nucleotide encoding and codon encoding as the input layer of the neural networks, has an effect on the performance of the neural network with respect to the accuracy of the model. Our experiments indicate that nucleotide encoding of DNA sequences, in general, leads to an improvement in accuracy of around 5-8% as compared to codon encoding. We propose a few neural network models based on Convolutional Neural Networks and Recurrent Neural Networks that gave us the best accuracy for our problem and can be used to capture information from nucleotide sequences. A Gated Recurrent Unit (GRU) based model with attention layer gave us the best accuracy of ~89%.

Furthermore, we list potential future work, such as classification of pausing due to mRNA secondary structures or detection and classification of artifacts such as PCR over-amplification which can introduce experimental bias in ribo-seq experiments.

BIBLIOGRAPHY

[1] Alberts, B. et al. “The shape and structure of proteins”. *Molecular Biology of the Cell. 4th edition*. Garland Science, 2002.

[2] Bahdanau, D. et al. “Neural machine translation by jointly learning to align and translate”. *arXiv preprint arXiv:1409.0473* (2014).

[3] Bottou, L. “Large-scale machine learning with stochastic gradient descent”. *Proceedings of COMPSTAT'2010*. Springer, 2010, pp. 177–186.

[4] Brar, G. A. & Weissman, J. S. “Ribosome profiling reveals the what, when, where and how of protein synthesis”. *Nature reviews Molecular cell biology* **16**.11 (2015), pp. 651–664.

[5] Calviello, L. & Ohler, U. “Beyond read-counts: Ribo-seq data analysis to understand the functions of the transcriptome”. *Trends in Genetics* **33**.10 (2017), pp. 728–744.

[6] Calviello, L. et al. “Detecting actively translated open reading frames in ribosome profiling data”. *Nature methods* **13**.2 (2016), p. 165.

[7] Cho, K. et al. “Learning phrase representations using RNN encoder-decoder for statistical machine translation”. *arXiv preprint arXiv:1406.1078* (2014).

[8] Cho, K. et al. “On the properties of neural machine translation: Encoder-decoder approaches”. *arXiv preprint arXiv:1409.1259* (2014).

[9] Chung, J. et al. “Empirical evaluation of gated recurrent neural networks on sequence modeling”. *arXiv preprint arXiv:1412.3555* (2014).

[10] Commons, W. *File:Central Dogma of Molecular Biochemistry with Enzymes.jpg — Wikimedia Commons, the free media repository*. [Online; accessed 24-March-2020]. 2020.

[11] Fluman, N. et al. “mRNA-programmed translation pauses in the targeting of E. coli membrane proteins”. *Elife* **3** (2014), e03440.

[12] Goodfellow, I. et al. *Deep learning*. MIT press, 2016.

[13] Gordon, A., Hannon, G., et al. “Fastx-toolkit”. *FASTQ/A short-reads preprocessing tools (unpublished) http://hannonlab. cshl. edu/fastx_toolkit* **5** (2010).

[14] Graves, A. “Supervised sequence labelling”. *Supervised sequence labelling with recurrent neural networks*. Springer, 2012, pp. 5–13.

[15] Hochreiter, S. “The vanishing gradient problem during learning recurrent neural nets and problem solutions”. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* **6**.02 (1998), pp. 107–116.

[16] Hochreiter, S. & Schmidhuber, J. “Long short-term memory”. *Neural computation* **9**.8 (1997), pp. 1735–1780.

[17] Ingolia, N. T. et al. “Genome-wide analysis in vivo of translation with nucleotide resolution using ribosome profiling”. *science* **324**.5924 (2009), pp. 218–223.

[18] Ingolia, N. T. et al. “The ribosome profiling strategy for monitoring translation in vivo by deep sequencing of ribosome-protected mRNA fragments”. *Nature protocols* **7**.8 (2012), pp. 1534–1550.

[19] Jurtz, V. I. et al. “An introduction to deep learning on biological sequence data: examples and solutions”. *Bioinformatics* **33**.22 (2017), pp. 3685–3690.

[20] Krizhevsky, A. et al. “Imagenet classification with deep convolutional neural networks”. *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[21] Kukurba, K. R. & Montgomery, S. B. “RNA sequencing and analysis”. *Cold Spring Harbor Protocols* **2015**.11 (2015), pdb–top084970.

[22] Kumari, R. et al. “PausePred and Rfeet: webtools for inferring ribosome pauses and visualizing footprint density from ribosome profiling data”. *Rna* **24**.10 (2018), pp. 1297–1304.

[23] Lamesch, P. et al. “The Arabidopsis Information Resource (TAIR): improved gene annotation and new tools”. *Nucleic acids research* **40**.D1 (2012), pp. D1202–D1210.

[24] LeCun, Y. et al. “Gradient-based learning applied to document recognition”. *Proceedings of the IEEE* **86**.11 (1998), pp. 2278–2324.

[25] LeCun, Y. et al. “Deep learning”. *nature* **521**.7553 (2015), pp. 436–444.

[26] Ma, S. et al. “Discovery of stress responsive DNA regulatory motifs in Arabidopsis”. *PLoS One* **7**.8 (2012).

[27] Merchante, C. et al. “Gene-specific translation regulation mediated by the hormone-signaling molecule EIN2”. *Cell* **163**.3 (2015), pp. 684–697.

[28] Morange, M. “The Central Dogma of molecular biology”. *Resonance* **14**.3 (2009), pp. 236–247.

[29] OâĂŹConnor, P. B. et al. “Comparative survey of the relative impact of mRNA features on local ribosome profiling read density”. *Nature Communications* **7**.1 (2016), pp. 1–12.

[30] Pelechano, V. et al. “Widespread co-translational RNA decay reveals ribosome dynamics”. *Cell* **161**.6 (2015), pp. 1400–1412.

[31] Perkins, P. & Heber, S. “Identification of ribosome pause sites using a Z-Score based peak detection algorithm”. *2018 IEEE 8th International Conference on Computational Advances in Bio and Medical Sciences (ICCABS)*. IEEE. 2018, pp. 1–6.

[32] Rumelhart, D. E. et al. “Sequential thought processes in PDP models”. *Parallel distributed processing: explorations in the microstructures of cognition* **2** (1986), pp. 3–57.

[33] Sabi, R. & Tuller, T. “A comparative genomics study on the effect of individual amino acids on ribosome stalling”. *BMC genomics* **16**.S10 (2015), S5.

[34] Schmidhuber, J. “Deep learning in neural networks: An overview”. *Neural networks* **61** (2015), pp. 85–117.

[35] Schuster, S. C. “Next-generation sequencing transforms today’s biology”. *Nature methods* **5**.1 (2008), pp. 16–18.

[36] Sleumer, M. C. et al. “Regulatory elements of Caenorhabditis elegans ribosomal protein genes”. *BMC genomics* **13**.1 (2012), p. 433.

[37] Snoek, J. et al. “Practical bayesian optimization of machine learning algorithms”. *Advances in neural information processing systems*. 2012, pp. 2951–2959.

[38] Srivastava, N. et al. “Dropout: a simple way to prevent neural networks from overfitting”. *The journal of machine learning research* **15**.1 (2014), pp. 1929–1958.

[39] Trapnell, C. et al. “TopHat: discovering splice junctions with RNA-Seq”. *Bioinformatics* **25**.9 (2009), pp. 1105–1111.

[40] Tunney, R. et al. “Accurate design of translational output by a neural network model of ribosome distribution”. *Nature structural & molecular biology* **25**.7 (2018), pp. 577–582.

[41] Varenne, S. et al. “Translation is a non-uniform process: effect of tRNA availability on the rate of elongation of nascent polypeptide chains”. *Journal of molecular biology* **180**.3 (1984), pp. 549–576.

[42] Zhang, S. et al. “ROSE: a deep learning based framework for predicting ribosome stalling”. *Available at SSRN 3155721* (2018).