Title
Improved Algorithms for Predicting Polyadenylation Sites and Cell Membranes From Expression, Sequence, and Image Data

Permalink
https://escholarship.org/uc/item/87b0892h

Author
Arefeen, Ashraful

Publication Date
2019

Peer reviewed|Thesis/dissertation
UNIVERSITY OF CALIFORNIA
RIVERSIDE

Improved Algorithms for Predicting Polyadenylation Sites and Cell Membranes
From Expression, Sequence, and Image Data

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

by

Ashraful Arefeen

September 2019

Dissertation Committee:

Dr. Tao Jiang, Chairperson
Dr. Eamonn Keogh
Dr. Stefano Lonardi
Dr. Vassilis Tsotras
The Dissertation of Ashraful Arefeen is approved:

________________________________________________________________________

________________________________________________________________________

________________________________________________________________________

________________________________________________________________________

Committee Chairperson

University of California, Riverside
Acknowledgments

I like to express my gratitude to all who have influenced me in writing this dissertation.

I am extremely grateful to my advisor, Dr. Tao Jiang, for his guidance, patience and suggestions during my graduate studies. I will always remember the weekly meetings with him during these five years. In those meetings, his critical comments and constructive suggestions worked as motivational boost. Special thanks to my committee members, Dr. Eamonn Keogh, Dr. Stefano Lonardi and Dr. Vassilis Tsotras for their valuable feedback and comments during my proposal defense. Moreover, I am thankful to Dr. Xinshu (Grace) Xiao for her valuable comments and guidance in my research. I would also like to thank Dr. Gaudenz Danuser and Dr. Satwik Rajaram for giving me the opportunities to work on some cool projects during my summer internship at UT Southwestern Medical Center.

Finally, I like to thank my parents and my wife, Fouzia Hossain Oyshi, for their constant support during the most difficult period of my life. Without them, this dissertation would not have been possible.
To my parents for all the support.
ABSTRACT OF THE DISSERTATION

Improved Algorithms for Predicting Polyadenylation Sites and Cell Membranes From Expression, Sequence, and Image Data

by

Ashraful Arefeen

Doctor of Philosophy, Graduate Program in Computer Science
University of California, Riverside, September 2019
Dr. Tao Jiang, Chairperson

Alternative polyadenylation (polyA) sites near the 3′ end of a pre-mRNA creates multiple mRNA transcripts with different 3′ untranslated regions (3′ UTRs). The sequence elements of a 3′ UTR are essential for many biological activities such as mRNA stability, sub-cellular localization, protein translation, protein binding, and translation efficiency. Moreover, numerous studies in the literature have reported the correlation between diseases and the shortening (or lengthening) of 3′ UTRs. As alternative polyA sites are common in mammalian genes, we develop two algorithms, named TAPAS and DeepPASTA, for predicting polyA sites from different data: RNA-Seq expression and sequence data. TAPAS detects novel polyA sites of a gene from RNA-Seq reads by considering read coverage as a time series data. The method is then extended to identify polyA sites that are expressed differently between two biological samples and genes that contain 3′ UTRs with shortening/lengthening events. On the other hand, DeepPASTA predicts polyA sites from sequence and RNA secondary structure data using a deep learning framework. As polyadenylation is a tissue-specific event, the tool also predicts tissue-specific polyA sites. Moreover, the tool
can predict the most dominant (i.e., frequently used) polyA site of a gene in a specific tissue and relative dominance when two polyA sites of the same gene are given. Our extensive experiments demonstrate that both TAPAS and DeepPASTA significantly outperform the existing tools in polyA site analysis.

The cells and their internal organelles carry genetic information in all living organisms. An effective method of studying cells and their organelles at different timestamps is to analyze the fluorescent microscopic images of tissues. As a result, computer-automated analyses of such microscopic images are getting popular for their efficiency and minimal human interaction. One of the most important computer-automated analyses is the cell membrane prediction from cell nucleus data. We propose a new tool, named DeepCEP, to predict cell membranes from nuclei using the fluorescent microscopic image data. Our experiments demonstrate that DeepCEP can be a potentially useful tool for analyzing microscopic images in practice.
# Contents

**List of Figures**

**List of Tables**

1 **Introduction**

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1 Polyadenylation site analysis</td>
<td>1</td>
</tr>
<tr>
<td>1.1.1 Alternative polyadenylation site analysis from RNA-Seq expression data</td>
<td>2</td>
</tr>
<tr>
<td>1.1.2 PolyA site analysis from sequence data</td>
<td>3</td>
</tr>
<tr>
<td>1.2 Fluorescence microscopic image analysis for cell membrane prediction</td>
<td>5</td>
</tr>
<tr>
<td>1.3 Publications</td>
<td>6</td>
</tr>
</tbody>
</table>

2 **TAPAS: tool for alternative polyadenylation site analysis**

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 Introduction</td>
<td>7</td>
</tr>
<tr>
<td>2.2 Methods</td>
<td>13</td>
</tr>
<tr>
<td>2.2.1 Detecting alternative polyadenylation sites</td>
<td>15</td>
</tr>
<tr>
<td>2.2.2 Detecting differentially expressed APA sites</td>
<td>20</td>
</tr>
<tr>
<td>2.2.3 Detecting shortening/lengthening events of 3′ UTRs</td>
<td>22</td>
</tr>
<tr>
<td>2.3 Experimental results</td>
<td>22</td>
</tr>
<tr>
<td>2.3.1 Performance on detecting APA sites</td>
<td>22</td>
</tr>
<tr>
<td>2.3.2 Performance on APA site-based differential expression analysis</td>
<td>30</td>
</tr>
<tr>
<td>2.3.3 Performance on detecting shortening/lengthening events</td>
<td>36</td>
</tr>
<tr>
<td>2.4 Discussion and time/memory efficiency</td>
<td>39</td>
</tr>
</tbody>
</table>

3 **DeepPASTA: deep neural network based polyadenylation site analysis**

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1 Introduction</td>
<td>45</td>
</tr>
<tr>
<td>3.2 Materials and methods</td>
<td>50</td>
</tr>
<tr>
<td>3.2.1 Predicting polyA sites</td>
<td>54</td>
</tr>
<tr>
<td>3.2.2 Predicting tissue-specific polyA sites</td>
<td>57</td>
</tr>
<tr>
<td>3.2.3 Predicting tissue-specific relatively dominant polyA sites</td>
<td>59</td>
</tr>
<tr>
<td>3.2.4 Predicting tissue-specific absolutely dominant polyA sites</td>
<td>62</td>
</tr>
<tr>
<td>3.3 Experimental results</td>
<td>64</td>
</tr>
</tbody>
</table>

viii
3.3.1 Performance on predicting polyA sites ........................................ 65
3.3.2 Performance on predicting tissue-specific polyA sites ................ 74
3.3.3 Performance on predicting tissue-specific relatively dominant polyA sites .......................................................... 87
3.3.4 Performance on predicting tissue-specific absolutely dominant polyA sites .......................................................... 90
3.4 Discussion .................................................................................. 92

4 DeepCEP: deep learning based cell membrane prediction from nucleus 94
4.1 Introduction ............................................................................. 94
4.2 Methods .................................................................................. 99
  4.2.1 Predicting cell structures from three channels (filtration model) ... 101
  4.2.2 Predicting cell structures from the nucleus strain channel (cell structure model) ......................................................... 104
4.3 Experimental results ................................................................. 109
4.4 Discussion and future research ................................................... 115

5 Conclusions ............................................................................. 117

Bibliography ............................................................................... 119
List of Figures

2.1 A flowchart of the TAPAS pipeline. In the differential expression analysis, we assume that \( n \) RNA-Seq replicates are given for each condition. In the figure, mapped reads also include read coverage information. 14

2.2 Some examples of filtration. (a) The PELT algorithm might output \( cp_1 \) as a change point even though the true APA site is \( cp_2 \), which is removed by TAPAS. (b) If a 3′ UTR frame contains an intron (either annotated or novel), then a well might be created in the read coverage. (c) Three situations of the read coverage over the frame are illustrated. In case 1, the mean read coverages before and after the well are similar and TAPAS removes both change points \( cp_1 \) and \( cp_2 \) around the well. In case 2, the mean read coverage before the well is greater than the mean read coverage after the well and TAPAS keeps \( cp_1 \) as a potential APA site. In case 3, when the mean read coverage before the well is smaller than that after the well (which is not common), TAPAS would remove both change points as in the first case. 17

2.3 Length distribution of the 3′ UTR frames extract from the human RefSeq annotation GRCh37. The 3′ UTR frames have lengths ranging from 2 bps to 238,767 bps, with the average being 1,770.786 bps. 24

2.4 Performance of the tools in APA site detection on simulated data with different sequencing depths. Plot (a) shows the sensitivity and plot (b) shows the precision. 25

2.5 Number of correct APA sites detected by different tools on the real dataset when the flexible range for matching a predicted APA site to a true APA site of 3′-Seq is 50 bps (a) and 100 bps (b). 27

2.6 Performance of TAPAS, Cuffdiff, DESeq, and DEXSeq in differential expression analysis in terms of sensitivity (a) and precision (b). Cuffdiff_annote denotes running Cuffdiff with the transcriptome annotation and DEXSeq_gene denotes running DEXSeq to detect DE genes (instead of DE APA sites). 32

2.7 Performance of TAPAS, DaPars and ChangePoint on detecting genes with shortening/lengthening events in terms of sensitivity (a) and precision (b). 37
3.1 The input and output of the polyA site prediction and tissue-specific polyA site prediction models of DeepPASTA. a) The polyA site prediction model of DeepPASTA takes a genomic sequence of 200 nts and three energy efficient RNA secondary structures predicted by RNAshapes [108] from the sequence as the input and predicts whether the input sequence contains a polyA site at the middle or not. b) Similar to the previous model, the tissue-specific polyA site prediction model of DeepPASTA takes a sequence and three corresponding secondary structures generated by RNAshapes as the input and predicts whether the input sequence contains a polyA site at the middle or not for the nine tissues studied in [31].

3.2 The input and output of the tissue-specific relatively and absolutely dominant polyA site prediction models of DeepPASTA. a) The tissue-specific relatively dominant polyA site prediction model of DeepPASTA takes a couple of sequences and corresponding secondary structures containing polyA sites of some gene at the middle as the input and predicts which polyA site is relatively dominant. b) Unlike the relatively dominant model, the absolutely dominant model of DeepPASTA takes a sequence and corresponding secondary structure containing a polyA site of some gene at the middle as the input and predicts whether the polyA site is an absolutely dominant polyA site of the gene.

3.3 Architectures of the polyA site prediction model of DeepPASTA, M3 and M4. The polyA site prediction model has four sub-models: a sequence and three secondary structure sub-models. Each sub-model consists of a convolution layer, a maxpooling layer, a recurrent layer (i.e., a bi-directional LSTM), a flattening layer, and a fully connected layer. On the other hand, M3 (model represented by the red dotted line) consists of a sequence sub-model. M4 (model represented by the yellow dotted line) is similar to M3, but its sequence sub-model does not contain a recurrent layer.

3.4 The training phase of the polyA site prediction model of DeepPASTA. In each iteration of the training phase, the model predicts a likelihood value for the given input. This prediction is compared with the ground truth using a loss function. The loss value is then used to tune the parameters of the deep learning model.

3.5 Architecture of the tissue-specific polyA site prediction model of DeepPASTA. Similar to the polyA site prediction model of DeepPASTA, this model has a sequence and three secondary structure sub-models. Each of these sub-models consists of a convolution layer, a maxpooling layer, a recurrent layer, a flattening layer, and a fully connected layer. This model is a multi-label classification model that has nine neurons in the output layer for predicting polyA sites in the nine tissues studied in [31].
3.6 Architecture of the model of DeepPASTA for predicting relative dominance in a particular tissue. The model takes two sequences of 200 nts and corresponding secondary structures generated by RNAshapes containing polyA sites of some gene at the middle as the input. Each of these sequences and secondary structures is processed by a sub-unit, which consists of a sequence and a secondary structure sub-models. The output layer compares the outputs from the two sub-units to predict the relatively dominant polyA site.

3.7 Architecture of the model of DeepPASTA for predicting absolutely dominate polyA sites of each gene in a particular tissue. The model has a sequence and a secondary structure sub-models. The output layer predicts whether the input polyA site is an absolutely dominant polyA site or not.

3.8 The impact of negative examples on the performance of DeepPASTA. In order to test the performance of DeepPASTA in predicting polyA sites on different negative examples, three datasets are considered: datasets with shifted negative examples where positive examples are shifted left and right by 50 bases, with random negative examples that do not contain the hexamer signal and with random negative examples containing the hexamer signal. The positive examples of these datasets are the same. The number of examples for these three datasets are 286218, 190812 and 190744, respectively. Plots in a show the AUC and AUPRC performance of DeepPASTA on the dataset with the shifted negative examples. Plots in b show the AUC and AUPRC performance on the dataset with the random examples that do not contain the hexamer signal. Plots in c show the AUC and AUPRC performance on the dataset with the random examples containing the hexamer signal.

3.9 The RNA secondary structures of genes COX4I1 and ADORA2B helped DeepPASTA in predicting polyA sites. The figure shows the secondary structures generated by RNAshapes for the 100-nt upstream sequences of some polyA sites of the genes. Both polyA sites have AATAAA as the polyadenylation signal (PAS), but the locations of the signal in each input are far away from the polyA sites (the PASs and the polyA sites are colored red in the sequences). It is well known that the PAS often occurs 10-30 nts upstream of a polyA site ([46], [31] and [111]). Hence, one might conjecture that a PAS has to be near a polyA site in order for it to be functional. The folding of the RNA secondary structures reduces the distance between the PAS and polyA site in each gene. Similar phenomena are also described in [14].
3.10 Hexamer signals extracted from the true positive polyA sites predicted by DeepPASTA on dataset 1. In order to identify the most frequently used signals, we consider the top three high strength 6-mers in each input sequence based on saliency maps. The barplot on the left shows the overall 20 most frequently used hexamer signals in polyA site prediction. Most of these signals are annotated in the literature [46], [31] and [111]. In addition, DeepPASTA used some novel hexamer signals: UAAAAU, GAUAAA, UAAUAU, AAUAU, and UUAAAA. The four barplots on the right show the most frequently used hexamer signals in four equally divided regions (as illustrated at the bottom of the figure) of the input sequence. From the four barplots, it is seen that DeepPASTA used fewer signals from the fourth region (150-200 nts) in polyA site prediction. Similar to previous studies, DeepPASTA identified the U-rich signals as auxiliary upstream elements (AUEs) in the first region (1-49 nts), U/GU-rich signals as downstream elements in the third region (101-149 nts) and G-rich signals as auxiliary downstream elements (ADEs) in the fourth region (150-200 nts).

3.11 Hexamer signals extracted from the true positive polyA sites predicted by the tissue-specific model of DeepPASTA on dataset 1 for the brain tissue. In order to identify the most frequently used signals, we consider the top three high strength 6-mers in each input sequence based on saliency maps. The barplot on the left shows the overall 20 most frequently used hexamer signals in polyA site prediction for the brain tissue. The four barplots on the right show the most frequently used hexamer signals in four equally divided regions (as illustrated at the bottom of the figure) of the input sequence. From these four barplots, it is seen that DeepPASTA used the most hexamer signals from the second region (50-100 nts) in polyA site prediction for the brain tissue. The most frequently used signals in that region are AAUAAA, AAAAAA, AAAAA, UAAUAU, and CAAUAA. These signals are known as the polyadenylation signals (PASs) in the literature [46], [31] and [111].

3.12 Hexamer signals extracted from the true positive polyA sites predicted by the tissue-specific model of DeepPASTA on dataset 1 for the kidney tissue. In order to identify the most frequently used signals, we consider the top three high strength 6-mers in each input sequence based on saliency maps. The barplot on the left shows the overall 20 most frequently used hexamer signals in polyA site prediction for the kidney tissue. The four barplots on the right show the most frequently used hexamer signals in four equally divided regions (as illustrated at the bottom of the figure) of the input sequence. From these four barplots, it is seen that DeepPASTA used the most hexamer signals from the second region (50-100 nts) in polyA site prediction for the kidney tissue. The most frequently used signals in that region are AAUAAA, AUAAAA, AAUAAA, AUAAAG, and CAAUAA. Again, these signals are known as the polyadenylation signals (PASs) in the literature [46], [31] and [111].
3.13 Hexamer signals extracted from the true positive polyA sites predicted by the tissue-specific model of DeepPASTA on dataset 1 for the liver tissue. In order to identify the most frequently used signals, we consider the top three high strength 6-mers in each input sequence based on saliency maps. The barplot on the left shows the overall 20 most frequently used hexamer signals in polyA site prediction for the liver tissue. The four barplots on the right show the most frequently used hexamer signals in four equally divided regions (as illustrated at the bottom of the figure) of the input sequence. From these four barplots, it is seen that DeepPASTA used the most hexamer signals from the second region (50-100 nts) in polyA site prediction for the liver tissue. The most frequently used signals in that region are AAUAAA, AUAAAA, AAAUAA, AUAAAG, and AUUAAA. Again, these signals are known as the polyadenylation signals (PASs) in the literature [46], [31] and [111].

3.14 Hexamer signals extracted from the true positive polyA sites predicted by the tissue-specific model of DeepPASTA on dataset 1 for the MAQC_Brain1 tissue. In order to identify the most frequently used signals, we consider the top three high strength 6-mers in each input sequence based on saliency maps. The barplot on the left shows the overall 20 most frequently used hexamer signals in polyA site prediction for the MAQC_Brain1 tissue. The four barplots on the right show the most frequently used hexamer signals in four equally divided regions (as illustrated at the bottom of the figure) of the input sequence. From these four barplots, it is seen that DeepPASTA used the most hexamer signals from the second region (50-100 nts) in polyA site prediction for the MAQC_Brain1 tissue. The most frequently used signals in that region are AAUAAA, AAAUAA, AUAAAG, and AUUAAA. Again, these signals are known as the polyadenylation signals (PASs) in the literature [46], [31] and [111].

3.15 Hexamer signals extracted from the true positive polyA sites predicted by the tissue-specific model of DeepPASTA on dataset 1 for the MAQC_Brain2 tissue. In order to identify the most frequently used signals, we consider the top three high strength 6-mers in each input sequence based on saliency maps. The barplot on the left shows the overall 20 most frequently used hexamer signals in polyA site prediction for the MAQC_Brain2 tissue. The four barplots on the right show the most frequently used hexamer signals in four equally divided regions (as illustrated at the bottom of the figure) of the input sequence. From these four barplots, it is seen that DeepPASTA used the most hexamer signals from the second region (50-100 nts) in polyA site prediction for the MAQC_Brain2 tissue. The most frequently used signals in that region are AAUAAA, AAAUAA, AUAAAG, CAUAAA, and UAAUAA. Again, these signals are known as the polyadenylation signals (PASs) in the literature [46], [31] and [111].
3.16 Hexamer signals extracted from the true positive polyA sites predicted by the tissue-specific model of DeepPASTA on dataset 1 for the MAQC_UHR1 tissue. In order to identify the most frequently used signals, we consider the top three high strength 6-mers in each input sequence based on saliency maps. The barplot on the left shows the overall 20 most frequently used hexamer signals in polyA site prediction for the MAQC_UHR1 tissue. The four barplots on the right show the most frequently used hexamer signals in four equally divided regions (as illustrated at the bottom of the figure) of the input sequence. From these four barplots, it is seen that DeepPASTA used the most hexamer signals from the second region (50-100 nts) in polyA site prediction for the MAQC_UHR1 tissue. The most frequently used signals in that region are AAUAAA, AAAUAA, AUAAAA, AUUAAA, and UAAUAA. Again, these signals are known as the polyadenylation signals (PASs) in the literature [46], [31] and [111].

3.17 Hexamer signals extracted from the true positive polyA sites predicted by the tissue-specific model of DeepPASTA on dataset 1 for the MAQC_UHR2 tissue. In order to identify the most frequently used signals, we consider the top three high strength 6-mers in each input sequence based on saliency maps. The barplot on the left shows the overall 20 most frequently used hexamer signals in polyA site prediction for the MAQC_UHR2 tissue. The four barplots on the right show the most frequently used hexamer signals in four equally divided regions (as illustrated at the bottom of the figure) of the input sequence. From these four barplots, it is seen that DeepPASTA used the most hexamer signals from the second region (50-100 nts) in polyA site prediction for the MAQC_UHR2 tissue. The most frequently used signals in that region are AAUAAA, AAAUAA, AUAAAA, AUUAAA, and UAAUAA. Again, these signals are known as the polyadenylation signals (PASs) in the literature [46], [31] and [111].

3.18 Hexamer signals extracted from the true positive polyA sites predicted by the tissue-specific model of DeepPASTA on dataset 1 for the muscle tissue. In order to identify the most frequently used signals, we consider the top three high strength 6-mers in each input sequence based on saliency maps. The barplot on the left shows the overall 20 most frequently used hexamer signals in polyA site prediction for the muscle tissue. The four barplots on the right show the most frequently used hexamer signals in four equally divided regions (as illustrated at the bottom of the figure) of the input sequence. From these four barplots, it is seen that DeepPASTA used the most hexamer signals from the second region (50-100 nts) in polyA site prediction for the muscle tissue. The most frequently used signals in that region are AAUAAA, AUUAAA, AAAUAA, AUAAAA, and AUAAAG. Again, these signals are known as the polyadenylation signals (PASs) in the literature [46], [31] and [111].
3.19 Hexamer signals extracted from the true positive polyA sites predicted by the tissue-specific model of DeepPASTA on dataset 1 for the testis tissue. In order to identify the most frequently used signals, we consider the top three high strength 6-mers in each input sequence based on saliency maps. The barplot on the left shows the overall 20 most frequently used hexamer signals in polyA site prediction for the testis tissue. The four barplots on the right show the most frequently used hexamer signals in four equally divided regions (as illustrated at the bottom of the figure) of the input sequence. From these four barplots, it is seen that DeepPASTA used the most hexamer signals from the second region (50-100 nts) in polyA site prediction for the testis tissue. The most frequently used signals in that region are AAUAAA, AUAAAA, AAUAAA, AUAAAG, and AUUA AA. Again, these signals are known as the polyadenylation signals (PASs) in the literature [46], [31] and [111].

3.20 Number of examples in the training and validation data used in the experiments on predicting tissue-specific relatively dominant polyA sites. As shown in the left plot, the number of training examples in dataset 4 ranges from 59.4% to 64.3% of the total number of examples (used in training, validation and testing) across all tissues, and the number of validation examples ranges from 15.8% to 19.7%. As shown in the right plot, the numbers of training and validation examples range from 60.5% to 61.7% and from 22.3% to 22.8%, respectively.

3.21 Number of examples in the training and validation data used in the experiments on predicting tissue-specific absolutely dominant polyA sites of each gene. As the plot shows, the number of examples in the training data ranges from 54.1% to 55.9% and the number of examples in the validation data ranges from 22.6% to 23.4%.

4.1 The input and output of the filtering and main models of DeepCEP. a-b) The filtration model of DeepCEP takes a patch containing three channels (nucleus, stroma, and cell structure) as the input and predicts better cell structure containing patches. These patches work as the ground truth data of the main model. c-d) The main model of DeepCEP takes a patch containing two channels (nucleus and stroma) as the input and predicts cell membranes of the input patch. More specifically, this model predicts the cell membranes around the nuclei of the input patch.

4.2 Architecture of the filtration model. The model has three identical U-net sub-models. The input to the model is a patch consisting of three channels: nucleus, stroma, and unfiltered cell structure strain channels. The output from the model is a patch consisting of filtered cell structure strain channel. Each of these U-net sub-models consist of multiple convolution layers to extract features from the input patch.

4.3 Architectures of the Resnet and U-net sub-models of the cell structure model. These sub-models consist of multiple convolution layers to extract features from a patch of an input image.
4.4 Architecture of the cell structure model. The cell structure prediction model has four sub-models: a U-net sub-model and three identical Resnet sub-models. The input to the model is a patch consisting of nucleus and stroma strain channels. The output from the model is a patch consisting of cell membranes around the nuclei.

4.5 Some of the patches of the training data. The patches from the first row are the input patches of DeepCEP. These patches contain nucleus and stroma strain channels. The second row shows the unfiltered cell structure strain channel patches corresponding to the patches of first row. The third row shows the filtered cell structure strain channel patches corresponding to the patches of second row. The patches from the third row are the ground truth patches of DeepCEP. Note that the filtered patches are generated using the filtration model of section 4.2.1.

4.6 Performance comparison of DeepCEP and the baseline model for predicting cell membranes from nuclei. a) Some of the input patches from Dataset 1. Each of these patches consists of two channels: nucleus and stroma strain channels. b) Expert’s annotation of cell membranes for these input patches. c) When the input patches are given to DeepCEP, it predicts the cell membranes. d) Similar to DeepCEP, when the input patches are given to the baseline model, it predicts the cell membranes.
List of Tables

2.1 Performance comparison in APA site detection on simulated data. The number of true APA sites is 21731. ........................................ 26
2.2 Performance comparison in APA site detection on real data. Two flexible ranges (50 bps and 100 bps) are considered for matching a predicted APA site with a true one from 3'-Seq. ........................................ 28
2.3 Performance comparison in APA site detection on real data, when the prediction results of the tools compared are filtered by the 3' UTR frames defined by TAPAS. Two flexible ranges (50 bps and 100 bps) are considered for matching a predicted APA site with a true one from 3'-Seq. The number of predicted APA sites of TAPAS is lowered to be closer to those of Cufflinks' and IsoSCM's. For a further comparison, Cufflinks is run with the reference transcriptome in RefSeq (i.e., Cufflinks -g). Note that, given the number of APA sites predicted by Cufflinks -g, its performance should be directly compared with that of TAPAS provided in Table S2 rather than the numbers in this table. ........................................ 28
2.4 Performance comparison in detecting internal APA sites located inside the 3' UTR frames on real data. ........................................ 29
2.5 Performance comparison in APA site detection on real data. Two flexible ranges (50 bps and 100 bps) are considered for matching a predicted APA site with a true one from PAS-Seq. ........................................ 29
2.6 Performance comparison in the detection of genes with differentially expressed (DE) APA sites on simulated data. The number of genes with actual DE APA sites is 1254, and each such gene contains only one DE APA sites. Since DEXSeq is designed for differential splicing (DS) rather than DE analysis [76, 104], we consider DE genes with at least two transcripts (298 in total) as the benchmark when evaluating the performance of DEXSeq. Here, Cuffdiff_anno = Cuffdiff with annotation. ........................................ 34
2.7 Performance comparison in the detection of genes with shortening/lengthening events on simulated data. The actual number of genes with shortening/lengthening events is 674. ........................................ 38
2.8 Performance comparison in the detection of genes with shortening/lengthening events on real data. ........................................ 39
2.9 Performance comparison between TAPAS and 3P-Seq in APA site detection on mouse liver data. Paired-end RNA-Seq reads from standard polyA+ libraries for mouse liver (SRX196268) were downloaded from NCBI and mapped by TopHat2 to the mouse genome. For performance evaluation, a 3′-Seq dataset for mouse liver (GSM747483) was also downloaded from NCBI and used as benchmark. We ran TAPAS on the mapped reads and compared its predicted APA sites against the benchmark. As a comparison, we downloaded the 3P-Seq data for mouse liver (GSM1268948) from NCBI. Among the 29932 APA sites reported in the 3-Seq data, TAPAS and 3P-Seq identified 10900 and 19480 sites, respectively. In terms of sensitivity, 3P-Seq outperforms TAPAS; but TAPAS outperforms 3P-Seq in terms of precision. Note that TAPAS uses standard RNA-Seq data which is very popular and easy to perform while 3P-Seq requires complex biological steps and large amounts of RNA for its analysis [55].

2.10 Comparison of time (in minutes) and peak memory (in gigabytes) usage among the APA site detection tools on the simulated dataset with 50 million reads used in Section 2.3.1. Here, the running time of TAPAS includes the calculation of read coverage by SAMtools.

2.11 Comparison of time and peak memory usage among the tools for shortening/lengthening analysis on the simulated dataset with 50 millions reads used in Section 2.3.3. Again, the running time of TAPAS includes the calculation of read coverage by SAMtools.

3.1 Performance comparison between DeepPASTA, PolyAR, Dragon PolyA Spotter, DeeReCT-PolyA, Conv-Net, and DeepPolyA in polyA site prediction on the three datasets introduced in the beginning of section 3.3.1 in terms of AUC and AUPRC.

3.2 The effect of data leak on DeepPASTA in polyA site prediction on dataset 1 in terms of AUC and AUPRC.

3.3 Contributions of the RNN and RNA secondary structures in polyA site prediction on datasets 1 and 2 in terms of AUC and AUPRC.

3.4 Performance comparison between the tissue-specific model of DeepPASTA, DeepPolyA and basic (i.e., non-tissue-specific) polyA site prediction models of DeepPASTA on datasets 1 and 2. Table S1 of the Supplementary Materials shows the numbers of positive and negative examples in the test datasets. Datasets 1 and 2 are represented as D1 and D2, respectively, in the table.

3.5 Performance comparison between DeepPASTA and Conv-Net in relatively dominant polyA site prediction on dataset 4 in terms of AUC and AUPRC. The performance of Conv-Net is based on our implementation of the method described in [68].

3.6 Performance comparison between DeepPASTA and Conv-Net [68] in relatively dominant polyA site prediction on dataset 5 using AUC and AUPRC.
3.7 Performance comparison between DeepPASTA and Conv-Net in predicting absolutely dominant polyA sites on dataset 6 in terms of AUC and AUPRC. The 2\textsuperscript{nd} and 3\textsuperscript{rd} columns give the number of positive and negative examples in the test data.

3.8 Training time of the four DeepPASTA models in our experiments.

4.1 Performance comparison between DeepCEP and the baseline model in predicting cell from nuclei on Dataset 1 in term of sensitivity and precision. Dataset 1 contains 20 annotated patches, and there are 639 cells in these patches.

4.2 Performance comparison between DeepCEP and the baseline model in predicting cell from nuclei on Dataset 2 in term of sensitivity and precision. Dataset 2 contains 12000 patches, and there are 137058 cells in these patches.
Chapter 1

Introduction

1.1 Polyadenylation site analysis

According to the central dogma of molecular biology, a DNA sequence is converted to proteins using transcription, post-transcriptional, and translation processes. Initially, the transcription process synthesizes a pre-mRNA from a fragment of DNA [65]. This pre-mRNA is converted to a mature mRNA by the post-transcriptional process. Finally, the mature mRNA is translated into the corresponding protein. There are three important steps in the post-transcriptional process [58]: addition of a 5’ cap, addition of a polyadenylation (polyA) tail, and splicing. The polyA tail is added to the 3’ end of a pre-mRNA by the polyadenylation process. More precisely, the polyadenylation process consists of two steps [120]: cleavage near the 3’ end of a pre-mRNA and the addition of a polyA tail at the cleavage site.

A 3’ untranslated region (UTR) is a suffix of an mRNA that starts after the stop codon and ends before the cleavage site of the mRNA. Alternative cleavage sites near the
3' end of an pre-mRNA create more than one mRNA transcripts containing 3' UTRs of different lengths. The length of a 3' UTR and the sequence elements (such as those AU and GU rich elements) near a 3' end may have an impact on mRNA stability, mRNA localization, protein translation, protein binding, and translation efficiency [15]. Moreover, the secondary structure of a 3' UTR is also important for translation efficiency and disruption of expression [15]. Alternative polyadenylation is very common phenomenon in eukaryote genes [111] and most human genes have alternative polyadenylation in their post-transcription process [81]. Therefore, the analysis of polyA sites would be of great importance in the study of mammalian genes.

1.1.1 Alternative polyadenylation site analysis from RNA-Seq expression data

The study of transcription has been advanced by the recent improvement in sequencing technologies. The advent of next-generation sequencing (NGS) technologies [16] has drastically improved sequencing time and cost. RNA sequencing (RNA-Seq), also known as whole transcriptome shotgun sequencing [83], uses NGS to study the presence and expression of an RNA in a biological sample in a given moment [27, 124]. RNA-Seq is an efficient way to track the continuously changing transcriptome. More specifically, RNA-Seq facilitates the ability to look at alternative spliced transcripts, post-transcriptional modifications, gene fusion, mutations/SNPs, and changes in gene expression over time or differences in gene expression in different groups or treatments [78]. In addition to determining the exon/intron boundaries in genes, RNA-Seq can be used to profile the 5' and 3' ends of genes. As a result, RNA-Seq can be used in alternative polyadenylation (APA) site anal-
ysis. Recently, several tools have been published for detecting APA sites from RNA-Seq data or performing shortening/lengthening analysis. These tools consider either up to only two APA sites in a gene or only APA sites that occur in the last exon of a gene, although a gene may generally have more than two APA sites and an APA site may sometimes occur before the last exon. Furthermore, the tools are unable to integrate the analysis of shortening/lengthening events with APA site detection.

In chapter 2, we propose a new tool, called TAPAS, for detecting novel APA sites from RNA-Seq data. It can deal with more than two APA sites in a gene as well as APA sites that occur before the last exon. The tool is based on an existing method for finding change points in time series data, but some filtration techniques are also adopted to remove change points that are likely false APA sites. It is then extended to identify APA sites that are expressed differently between two biological samples and genes that contain 3′ UTRs with shortening/lengthening events. Our extensive experiments on simulated and real RNA-Seq data demonstrate that TAPAS outperforms the existing tools for APA site detection and shortening/lengthening analysis significantly.

1.1.2 PolyA site analysis from sequence data

DNA is a molecule that carries genetic instructions for the development, functioning, growth, and reproduction of all known organism and many viruses. The instruction/information in DNA is stored as a code made up of four chemical/nucleotide bases: adenine (A), guanine (G), cytosine (C), and thymine (T). Human DNA consists of about 3 billion bases, and the order of these bases is extremely important for carrying genetic instructions. DNA sequencing is the process of determining the order of these nucleotide
bases in DNA. Nowadays, DNA sequencing has become very crucial in numerous fields such as biotechnology, forensic biology, and medical diagnostics. The development of next-generation sequencing (NGS) technologies [16] has drastically improved sequencing time, cost, and efficiency. Moreover, these technologies have facilitated researchers in investigating insights into health, human origins, etc. Due to the advancement of sequencing technologies, computational tools are developed to analyze the sequence data. Selection of a polyadenylation site for the polyadenylation process also depends on the sequence motifs or cis-elements. As a result, several machine learning tools have been published for predicting polyA sites from sequence data or performing relative and absolute dominant analyses. These tools either consider limited sequence features or use relatively old algorithms for polyA site prediction. Moreover, none of the previous tools consider tissue specific polyA site analysis and RNA secondary structures as a feature to predict polyA sites.

In chapter 3, we propose a new deep learning model, called DeepPASTA, for predicting polyA sites from both sequence and RNA secondary structure data. The model is then extended to predict tissue-specific polyA sites. Moreover, the tool can predict the most dominant (i.e., frequently used) polyA site of a gene in a specific tissue and relative dominance when two polyA sites of the same gene are given. Our extensive experiments demonstrate that DeepPASTA significantly outperforms the existing tools for polyA site prediction and tissue-specific relative and absolute dominant polyA site prediction.
1.2 Fluorescence microscopic image analysis for cell membrane prediction

The cell is the basic building unit for all living organisms. The interior of a cell consists of different organelles, and one of the major organelles is the cell nucleus. The cell nucleus works as a repository of information for all living organisms. As cells and their organelles are very important, it is essential to various research areas to study cells and their sub-cellular components. One powerful way of studying cells and its organelles at different timestamps is to analyze the microscopic images of tissues. Fluorescence labeling on top of the microscopy imaging provides unprecedented opportunities to study the structural features and functional characteristics of a cell, e.g., cell nucleus, cell membrane, stroma, cell proliferation signal, etc. However, fluorescence labeling has some limitations. In order to overcome these limitations, computer-automated analyses on the microscopic images are getting popular. Recently, several machine learning tools have been developed for automatically analyzing the microscopic images. One of the most important automatic analyses is cell membrane prediction. Although there are several tools for cell membrane prediction, none of them predicts it only from the nucleus.

In chapter 4, we propose a new deep learning model, called DeepCEP, for predicting cell membranes from nuclei using the fluorescence microscopic image data. Our extensive experiments demonstrate that DeepCEP significantly outperforms a baseline model for cell membrane prediction.
1.3 Publications


Chapter 2

TAPAS: tool for alternative polyadenylation site analysis

2.1 Introduction

According to the central dogma of molecular biology, the transcription process in eukaryotes synthesizes a pre-mRNA from the genomic sequence of a gene [65]. The pre-mRNA is then converted to a mature mRNA by the post-transcriptional process. Finally, this mature mRNA is translated into the corresponding protein. The post-transcriptional process includes three major steps: the addition of a 5′ cap, addition of a polyadenylation (polyA) tail and splicing. In particular, a polyA tail is added at the 3′ end of a pre-mRNA with the help of the polyadenylation process. More precisely, the polyadenylation process consists of two steps [120]: cleavage near the 3′ end of a pre-mRNA and the addition of a polyA tail at the cleavage site. Certain cis-acting elements and trans-acting factors have
been found in the literature that influence the choice of a particular polyA cleavage site [15, 94]. In particular, the 3′ end sequence of a pre-mRNA usually contains a AAUAAA hexamer (or some close variant). This hexamer is called the polyadenylation signal (PAS) and it usually appears 10-30 bps upstream of the cleavage site [111]. The PAS serves as a binding site for the cleavage and polyadenylation specificity factor (CPSF). U-rich or U/G-rich elements located 20-40 bps downstream of the cleavage site are also involved in polyadenylation [111]. These U-rich or U/G-rich elements serve as the binding sites for the cleavage stimulation factor (CstF). In addition, some auxiliary elements upstream of the PAS and downstream of the cleavage site may enhance the polyadenylation process [111]. Due to the interactions between these cis elements and polyadenylation factors, alternative cleavage sites can be formed for a pre-mRNA, resulting in more than one mRNA transcript from a single pre-mRNA containing 3′ untranslated regions (3′ UTRs) of different lengths. Note that a 3′ UTR is a suffix of an mRNA sandwiched between the stop codon and polyadenylation cleavage site of the mRNA. The length of a 3′ UTR as well as some sequence elements in the 3′ UTR such as AU-rich elements and GU-rich elements may have impact on mRNA stability, mRNA localization, protein translation, protein binding and translation efficiency [15]. Moreover, the secondary structure of a 3′ UTR is also important for its translation efficiency and disruption of expression [15]. Alternative polyadenylation (cleavage) is very common in mammalian genes [111]. According to the study in [26], more than half of human genes have alternative polyadenylation in their post-transcriptional process. Therefore, the analysis of alternative (or all) polyadenylation sites (APA sites) would be of great importance for the study of mammalian genes.
The analysis of expressed sequence tags (ESTs) has provided genome-wide annotations of 3' UTRs. Not only does this analysis show that mammalian genes have multiple 3' UTRs [111], but also it reveals that neuronal cell mRNAs have longer 3' UTRs than liver cell mRNAs [106]. However, an EST based approach is not able to estimate the relative abundance of each 3' UTR in the resultant mRNAs [55]. Using 3P-Seq data, the 3' UTRs of genes in yeast, worm, fly, zebrafish, mouse and human genomes have been annotated in [48, 115, 85, 8, 79, 31, 45, 103]. Unlike EST based approaches, these methods based on 3P-Seq precisely detect the usage of different 3' UTRs in mRNAs. On the other hand, they require complex biochemical steps and large amounts of RNA for their analyses [55].

The advancement of RNA-Seq technology has provided new avenues for the study of transcription including the polyadenylation process. A typical RNA-Seq data analysis process begins with mapping RNA-Seq reads to some reference genome using tools like TopHat2 [53], HISAT [54]. Once the reads are mapped, mRNA transcripts (or isoforms) are assembled by using tools like Cufflinks [113], IsoLasso [70], StringTie [93], or TransComb [75], and their abundance levels are quantified by using tools like Cufflinks [113], RSEM [69], CEM [71], eXpress [96], Kallisto [19], etc. Moreover, differential expression between samples can be analyzed by using tools such as DESeq [7], Cuffdiff [114] or DEXSeq [6].

Recently, several methods for discovering 3' UTRs from RNA-Seq data have been introduced in the literature. The tool introduced in [77] studies the dynamic expression of 3' UTRs using a Poisson hidden Markov model. Due to the design of the model, the tool is only able to identify up to two alternative polyadenylation sites for a given gene. The web server 3USS [92] takes a pair of annotated genome and transcriptome and outputs alternative 3'
UTRs. It only reports the polyadenylation sites given in the transcriptome and thus would be unable to provide any novel APA sites. Roar [40] takes annotated APA sites from public databases to identify genes undergoing regulation of 3′ UTR length. Similar to 3USS, Roar is unable to discover novel APA sites. GETUTR [55] is another RNA-Seq based tool to estimate the 3′ UTR landscape. The method takes mapped reads and a reference genome as the input, and finds APA sites by using techniques to smooth read coverage including isotonic (or monotone) regression [63]. A drawback of the method is that these smoothing techniques may result in many false APA sites. On the other hand, although introns may occur in 3′ UTRs ([15], [17]), GETUTR does not consider intronic regions in its analysis and thus often misses 3′ UTRs that contain introns. IsoSCM [102] identifies alternative 3′ UTRs based on a multiple change-point inference model. It first uses the (statistical) model to infer change points in a gene that exhibit sharp increase or decrease in read coverage. Then it employs some additional mathematical constraint to filter change points that are likely to be false APA sites. Similar to GETUTR, the method does not consider introns inside a 3′ UTR. DaPars [127] and ChangePoint [122] are tools for comparing APA sites in two biological samples and detecting shortening/lengthening events. Both of these tools consider only two cleavage sites in their shortening/lengthening analysis, although a gene may have more than two APA sites.

In this chapter, we introduce a new tool, called TAPAS (i.e., Tool for Alternative Polyadenylation site Analysis), for detecting novel APA sites from RNA-Seq data. It can deal with more than two APA sites in a gene as well as 3′ UTRs that contain intronic regions. The tool is based on the Pruned Exact Linear Time (PELT) method for finding
change points in time series data [52], but some filtration techniques that take into account special properties of RNA-Seq data and the exonic structures of the 3′ UTRs of the same gene are also employed to remove change points that are likely false APA sites. The tool is then extended to identify APA sites that are expressed differently between two biological samples with multiple replicates by using an elaborate algorithm to align APA sites from each replicate and standard statistical approaches for differential expression analysis such as the one in [7]. The differential expression analysis is further extended to identify genes that have 3′ UTRs with shortening/lengthening events.

To assess the performance of TAPAS, we have conducted extensive experiments on both simulated and real data and compared TAPAS with the above mentioned tools IsoSCM, GETUTR, DaPars and ChangePoint for APA site or differential expression analysis. Moreover, since a complete transcriptome provides full information about APA sites, we also include the most popular tool for transcriptome assembly, Cufflinks, and its corresponding tool for transcript-based differential expression analysis, Cuffdiff, in the comparison. As none of these existing tools are able to perform all three types of APA site and differential expression analysis that TAPAS can do, we organize the comparison as three groups: (i) detection of APA sites (between TAPAS, IsoSCM, GETUTR, and Cufflinks), (ii) detection of genes with differentially expressed APA sites (between TAPAS, Cuffdiff, DESeq, and DEXSeq), and (iii) detection of genes with shortening/lengthening events (between TAPAS, DaPars and ChangePoint). We exclude 3USS, Roar and the tool in [77] from the comparison because they are either unable to discover novel APA sites or seriously restricted. In the simulation experiments, the tools are compared in terms of sensitivity and precision. Based
on these two performance measures, TAPAS outperforms IsoSCM, GETUTR and Cufflinks significantly in the detection of APA sites. When 3′-Seq (or polyA-Seq) and PAS-Seq data are considered as the ground truth in real data experiments, TAPAS is able to deliver more true APA sites than the other tools with a similar number of predicted APA sites. For the detection of genes with differentially expressed APA sites, TAPAS achieves a higher sensitivity than Cuffdiff and DEXSeq even though they are provided with an annotated transcriptome. Although its sensitivity is initially worse than that of DESeq, the gap decreases rapidly with the increase of sequencing depth. While its precision is also higher than that of Cuffdiff without the transcriptome annotation and DEXSeq, it is slightly lower than that of Cuffdiff with the transcriptome annotation and lower than that of DESeq (but the gaps shrink as well with the increase of sequencing depth). In the shortening/lengthening event analysis, TAPAS outperforms significantly DaPars and ChangePoint on simulated data. On a real dataset and once again using 3′-Seq data as the ground truth, TAPAS identifies more genes with real shortening/lengthening events than the other two, when all the tools are tuned to output similar number of events. We also analyze the time and memory efficiency of TAPAS and demonstrate that while TAPAS requires a significant amount of memory, its running time is comparable to that of the other tools.

The rest of the chapter is organized as follows. The method of TAPAS is discussed in section 2.2. The experimental results and comparison with the other tools are given in section 2.3. A brief evaluation of the running time and memory efficiency of the tools is given in section 2.4.
2.2 Methods

TAPAS takes a set of mapped RNA-Seq reads from standard polyA+ libraries along with the read coverage information and an annotated genome as the input to detect alternative polyadenylation sites (i.e., APA sites). It first extracts the 3′ UTRs of every gene in the genome annotation. The overlapping 3′ UTRs in a gene are merged into a 3′ UTR frame (if a gene has only one 3′ UTR, then that 3′ UTR is considered as the 3′ UTR frame of the gene). Then it estimate the the read coverage of the 3′ UTR frames. The read coverage of each of these frames is given as the input to the PELT algorithm to infer change points in a gene where the read coverage increases or decreases sharply. Since not all such change points are true APA sites, TAPAS filters them to produce a list of predicted APA sites. The abundance of an APA site (i.e., the total abundance of all transcripts that end at the APA site) can be estimated by using the quantification method in [113]. When two biological samples with multiple replicates are given, TAPAS can be applied to each replicate to obtain its set of APA sites and the associated abundance. The sets of APA sites from all replicates are then aligned using an elaborate algorithm and some standard statistical steps like those used in DESeq [7] are applied to identify APA sites that are differentially expressed in the two samples. This analysis can be easily extended to infer genes that have shortened/lengthened 3′ UTRs between the two samples. The flowchart shown in Figure 2.1 illustrates the main steps of TAPAS. Each of these steps is explained in detail below.
Figure 2.1: A flowchart of the TAPAS pipeline. In the differential expression analysis, we assume that $n$ RNA-Seq replicates are given for each condition. In the figure, mapped reads also include read coverage information.
2.2.1 Detecting alternative polyadenylation sites

As mentioned above, TAPAS starts its APA site analysis by extracting 3′ UTR frames of each gene from an annotated genome (or transcriptome, if it is available). Such an annotation typically provides some known 3′ UTRs of each gene. Some of the 3′ UTRs may overlap. In order to avoid the potential inference between overlapping 3′ UTRs in our subsequent change point analysis, we merge multiple overlapping 3′ UTRs of a gene into a frame. For convenience, if a gene has only one 3′ UTR, the 3′ UTR is also considered as the 3′ UTR frame of the gene. Next, it takes a set of standard RNA-Seq reads mapped to the reference genome by TopHat2 [53] along with read coverage information and extracts the read coverage for each base position of a 3′ UTR frame. The prune exact linear time (PELT) algorithm [52] based on dynamic programming is applied to infer APA sites in each 3′ UTR frame as follows.

Let the read coverage of a 3′ UTR frame be \( y_{1:n} = y_1, y_2, \ldots, y_n \) and \( t_{1:m} = t_1, t_2, \ldots, t_m \) the (potential) "change points" in the frame. These \( m \) change points split the sequence \( y_{1:n} \) into \( m + 1 \) segments, where the \( i^{th} \) segment is represented as \( y_{t_{i-1}+1:t_i} \), and can be determined by minimizing equation 2.1:

\[
\sum_{i=1}^{m+1} C(y_{t_{i-1}+1:t_i}) + m\gamma
\]  

(2.1)

where \( C(y_{t_{i-1}+1:t_i}) = -2 \times max_\lambda \sum_{j=t_{i-1}+1}^{t_i} \log f(y_j|\lambda) \)

The minimization involves a cost function \( C() \) and penalty \( m\gamma \), where \( \gamma \) is a parameter estimated from the read coverage \( y_{1:n} \). Similar to the method in [49], we assume that
the read coverage in a segment follows a Poisson distribution with density function \( f \) and mean \( \lambda \), and use twice the negative log-likelihood method to determine \( C \). More details of the PELT algorithm for inferring change points as well as determining the value of \( m \) are given in Algorithm 1.

**Algorithm 1** The PELT method for finding change points in a 3' UTR frame.

**procedure** PELTMethod\((y, C, \gamma)\)

**Input:**
- \( y \rightarrow \) read coverage of a 3' UTR frame, \((y_1, y_2, \ldots, y_n)\)
- \( C \rightarrow \) twice negative log-likelihood cost function on \( y \)
- \( \gamma \rightarrow \) penalty

**Initialize:**
- \( F(0) = -\gamma \)
- \( cp(0) = \text{NULL} \)
- \( R_1 = \{0\} \)

**for** \( t^* = 1, \ldots, n \) **do**

\[
\begin{align*}
F(t^*) &= \min_{t \in R_{t^*}} [F(t) + C(y_{t+1:t^*}) + \gamma] \\
t^1 &= \arg \{ \min_{t \in R_{t^*}} [F(t) + C(y_{t+1:t^*}) + \gamma] \} \\
cp(t^*) &= [cp(t^1), t^1] - \{0\} \\
R_{t^*+1} &= \{t^*, \{t \in R_{t^*} : F(t) + C(y_{t+1:t^*}) < F(t^*)\}\}
\end{align*}
\]

**Output:** change points, \( cp(n) \)

The change points found by the PELT algorithm indicate positions in a 3' UTR frame where the read coverage increases or decreases sharply. Not all of them are necessarily true APA sites. In particular, the read coverage typically decreases rather than increases at an APA site, although it may increase after an intron contained in a 3' UTR frame.
Figure 2.2: Some examples of filtration. (a) The PELT algorithm might output $cp_1$ as a change point even though the true APA site is $cp_2$, which is removed by TAPAS. (b) If a 3' UTR frame contains an intron (either annotated or novel), then a well might be created in the read coverage. (c) Three situations of the read coverage over the frame are illustrated. In case 1, the mean read coverages before and after the well are similar and TAPAS removes both change points $cp_1$ and $cp_2$ around the well. In case 2, the mean read coverage before the well is greater than the mean read coverage after the well and TAPAS keeps $cp_1$ as a potential APA site. In case 3, when the mean read coverage before the well is smaller than that after the well (which is not common), TAPAS would remove both change points as in the first case.
Therefore, we need filter the change points output by the PELT algorithm to reduce false positives.

It has been observed in our preliminary experiments that the PELT algorithm often outputs an extra change point before a true APA site when the read coverage increases or decreases gradually (please see Figure 2.2 (a) for more details). To remove the spurious change point, we scan the coverage between two consecutive change points from left to right. If it is generally decreasing, then TAPAS removes the first change point. If it is generally increasing, then TAPAS removes the second change point. The details of this filtration procedure are given in Algorithm 2.

**Algorithm 2** Filtration of change points found by PELT when the read coverage of a 3′ UTR frame increases (or decreases) gradually.

```markdown
procedure FilterRedundantChangePoints(cp, coverage, strand)

Input:

- cp → change points of a 3′ UTR frame
- coverage → read coverage of the 3′ UTR frame
- strand → strand of the 3′ UTR frame

if strand = positive then
  for each pair of consecutive change points, (cp_{i-1}, cp_i) do
    if most of the base positions between cp_{i-1} and cp_i have decreasing coverage then
      remove cp_{i-1} from the list of APA sites
  else
    for each pair of consecutive change points, (cp_i, cp_{i+1}) do
      if most of the base positions between cp_i and cp_{i+1} have increasing coverage then
        remove cp_{i+1} from the list of APA sites
```

If a 3′ UTR frame does not contain any intron, then the read coverage is generally expected to monotonically decrease across the frame. However, introns occur in 3′ UTRs [15] and they cannot be ignored [17] in APA site analysis. When introns (either annotated
or novel) exist in a frame, “wells” could be created in the read coverage, as illustrated in Fig. 2.2 (b). This might lead the PELT algorithm to output change points around the introns that are unlikely to be true APA sites. These spurious change points can be removed according to cases as illustrated in Figure 2.2 (c). More details of this filtration step are given in Algorithm 3. Note that various biases in RNA-Seq data such as positional bias, sequencing bias and mappability bias may also cause PELT to report false change points, but they are not dealt with explicitly here.

After filtering potentially spurious change points, TAPAS obtains a list of predicted APA sites for each 3′ UTR frame. Note that since the 3′ UTR frames are extracted from the input genome (or transcriptome) annotation and the end of each such frame is likely an (expressed) APA site, the real novelty of TAPAS is the detection of internal APA sites located inside the 3′ UTR frames.

**Estimation of the abundance of alternative 3′ UTRs**

In order to perform differential expression analysis based on APA sites, we need estimate the abundance of each APA site. Here, the abundance of an APA site is defined as the total abundance of all transcripts that end at the APA site. Instead of considering full transcripts (which are unknown), TAPAS considers all possible 3′ UTRs within a 3′ UTR frame, as a crude approximation. The introns (annotated or identified in the filtration step) located in a 3′ UTR are factored into the effective length of the 3′ UTR. Let \( R \) be the set of reads mapped to a 3′ UTR frame, \( T \) the set of all possible 3′ UTRs in the frame, and \( \rho_t \) and
the abundance and effective length of a specific 3′ UTR $t$, respectively. The abundance of $t$ can be estimated by equation 2.2, as done similarly in Cufflinks [113].

$$L(p|R) = \prod_{r \in R} \sum_{t \in T} a_{r,t} \frac{\rho_t}{\sum_{u \in T_r} \rho_u (l_t - l_r + 1)}$$  \hspace{1cm} (2.2)

Here, $a_{r,t} = 1$ when a 3′ UTR $t$ contains read $r$, or otherwise $a_{r,t} = 0$. $T_r$ denotes all 3′ UTRs containing read $r$. This likelihood function can be maximized by using an EM algorithm similar to the one introduced in the transcript quantification tool IsoEM [87]. The details of the EM algorithm are given in Algorithm 4. Note that here the abundance of a transcript is measured in read count rather than RPKM or FPKM.

### 2.2.2 Detecting differentially expressed APA sites

If two biological samples with multiple replicates are given, TAPAS first identifies potential APA sites for each replicate along with their abundance levels (measured in read count) by following the steps in section 2.2.1. It then “aligns” the APA sites from all replicates by merging them based on their genomic locations as follows. It puts all the APA sites of a gene across the replicates into a list and sorts them by their genomic locations. TAPAS then merges a pair of neighboring APA sites on the list into a cluster if their genomic distance is less than some threshold (which is set as 70 bps in our experiments based on several trials) and they are from different replicates. It repeats this step until no more neighboring APA sites can be merged. Finally, every singleton cluster (i.e., a cluster with only one APA site from some replicate) is merged with its nearest neighbor cluster. Each cluster will be considered as an APA site in the differential expression analysis, and
its genomic location is determined by the majority location in the cluster. If there is a tie, TAPAS takes the median genomic location of all APA sites in the cluster. If a cluster contains an APA site \( a \) from a replicate \( r \), then its abundance in \( r \) is defined as the abundance of \( a \). If the cluster does not contain any APA site from \( r \), then its abundance in \( r \) is zero.

Let \( A \) and \( B \) be two samples with \( m_A \) and \( m_B \) replicates, respectively, and \( m = m_A + m_B \). Suppose that the above alignment procedure results in \( n \) clusters for all genes. Denote the abundance (in read count) of these clusters in all replicates as an \( n \times m \) matrix \( k_{i,j} \), where \( i = 1, 2, \ldots n \) indexes the APA sites and \( j = 1, 2, \ldots m \) indexes the replicates. As in [7], we assume that the read counts of an APA site across all replicates from the same sample follow a negative binomial (NB) distribution:

\[
k_{i,x} \sim NB(\mu_{i,x}, \sigma_{i,x}^2),
\]

(2.3)

where \( \mu_{i,x} \) and \( \sigma_{i,x} \) are the mean and variance of the NB distribution, respectively, for APA site \( i \) in sample \( x \) (\( x = A \) or \( B \)). NB distributions can be used to model count data with over-dispersion [22] and are popular in RNA-Seq based differential expression analysis. The mean and variance can be estimated by fitting the data to a mathematical model, and the null hypothesis that an APA site is not differentially expressed between the two samples can be tested as in [7].

Finally, TAPAS reports an APA site as differentially expressed if the Benjamini & Hochberg adjusted \( p \)-value for the APA site is less or equal to 0.1.
2.2.3 Detecting shortening/lengthening events of 3’ UTRs

3’ UTRs (and their corresponding APA sites) are sometimes shortened or lengthened to cause significant changes in gene functions ([127], [12]). Hence, it would be interesting to accurately detect shortening/lengthening events between two biological conditions. We start with the above differential expression analysis for APA sites. Consider a pair of APA sites $i$ and $j$ where at least one APA site is differentially expressed and APA site $i$ precedes APA site $j$ on the genome. Denote the mean abundance of $i$ and $j$ in samples A and B as $e_{i,A}$, $e_{j,A}$, $e_{i,B}$, and $e_{j,B}$, respectively. We can use the following equation 2.4 to calculate the relative change value for the APA site pair:

$$rc_{i,j} = \log_2 \left( \frac{e_{j,B}}{e_{j,A}} \right) - \log_2 \left( \frac{e_{i,B}}{e_{i,A}} \right)$$  \hspace{1cm} (2.4)$$

Similar to [12], if $|rc_{ij}| \geq 1.0$, then the APA site pair $(i, j)$ is considered as giving rise to a shortening/lengthening event. TAPAS outputs all genes that contain APA site pairs with shortening/lengthening events.

2.3 Experimental results

In this section, we compare the performance of TAPAS with those of some state-of-the-art methods in term of detecting APA sites, differentially expressed APA sites and shortening/lengthening events on both simulated and real data.

2.3.1 Performance on detecting APA sites

In this experimental study, we compare TAPAS with two existing tools IsoSCM [102] and GETUTR [55] for detecting APA sites. As explained in Introduction, as APA
sites are uniquely determined by transcripts, we also include the most popular transcriptome assembly method Cufflinks [113] in the comparison. In order to simulate RNA-Seq data, we download the human RefSeq annotation GRCh37 (hg19) from the UCSC Genome Browser. The annotation contains 19150 genes with 44923 transcripts and 21731 APA sites. Among these genes, 17083, 1769 and 298 have one, two or more than two unique APA sites each, respectively. The distribution of the lengths of the 3′ UTR frames extracted by TAPAS from the annotation is plotted in Figure 2.3. Using this annotation and RNASeq Read Simulator\(^1\) (genexplvprofile.py with parameters -e -1,2) introduced in ([71]), an expression profile is generated with the log normal distribution. Based on this expression profile, single-end reads with lengths 76 bps are simulated to create 50, 100 and 150 million read datasets. We consider three datasets to evaluate how sequencing depth may impact the performance of the tools in APA site detection.

Since it is difficult to detect APA sites from RNA-Seq data at single nucleotide precision, some degree of flexibility is used to match predicted APA sites to the annotated ones as done similarly in [102]. For TAPAS, if a predicted APA site is within 50 bps of some annotated APA site then the prediction is considered as a true positive (TP), or otherwise a false positive (FP). We use 100 bps as the flexible range of matching for IsoSCM, GETUTR and Cufflinks because it was used in ([102], [55]). The numbers of TPs, FPs and true (i.e., annotated) APA sites (P) are used to calculate sensitivity ($\frac{TP}{P}$) and precision ($\frac{TP}{TP+FP}$). In the calculation of sensitivity, all TPs matching the same true APA site count as one TP.

\(^1\)http://alumni.cs.ucr.edu/~liw/rnaseqreadsimulator.html
Figure 2.3: Length distribution of the 3' UTR frames extract from the human RefSeq annotation GRCh37. The 3' UTR frames have lengths ranging from 2 bps to 238,767 bps, with the average being 1,770.786 bps.

Among the 21731 annotated APA sites, TAPAS identifies 16866, 18205 and 18871 true APA sites on the 50, 100 and 150 million read datasets, respectively. For the other tools, IsoSCM identifies 11790, 13583 and 14592 true APA sites, GETUTR identifies 15495, 16596, 17082 true APA sites and Cufflinks identifies 15117, 16303 and 16779 true APA sites, respectively. The sensitivity and precision of the methods are illustrated in Fig. 2.4. It can be seen from the figure that all tools perform better with the increase of sequencing depth. Table 2.1 provides a detailed account of the performance of the tools. Clearly, TAPAS outperforms all three other tools in both sensitivity and precision. Note that among the tools, IsoSCM and Cufflinks do not use the transcriptome annotation, but TAPAS and GETUTR use the annotation to define 3' UTR frames. However, once the reads are mapped to the frames, the annotation is no longer used in the latter two tools. In
Figure 2.4: Performance of the tools in APA site detection on simulated data with different sequencing depths. Plot (a) shows the sensitivity and plot (b) shows the precision.

In particular, these tools do not consult the annotated APA sites when deciding if a change point should be output as a predicted APA site. While the use of annotation might have helped the performance of TAPAS and GETUTR (especially its sensitivity), it does not benefit GETUTR’s precision because the tool does not perform rigorous filtration as TAPAS and IsoSCM do. Although Cufflinks achieves a decent sensitivity, its precision is low because it assembles many transcripts with incorrect APA sites. GETUTR and IsoSCM have the worst performance in the experiment (in term of precision). While the performance of GETUTR is consistent with the results in [55], it is reported in [102] that IsoSCM performs well when the sequencing depth is 500 reads/kb or more. Note that the sequencing depths for our 50, 100 and 150 million read datasets are in fact 326, 652 and 977 reads/kb, respectively.
Table 2.1: Performance comparison in APA site detection on simulated data. The number of true APA sites is 21731.

<table>
<thead>
<tr>
<th>Dataset (in million)</th>
<th>Tool name</th>
<th>Number of predicted APA sites</th>
<th>Correctly identified APA sites</th>
<th>Sensitivity (%)</th>
<th>Precision (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>TAPAS</td>
<td>19453</td>
<td>16866</td>
<td>77.61</td>
<td>86.70</td>
</tr>
<tr>
<td>100</td>
<td>TAPAS</td>
<td>20712</td>
<td>18205</td>
<td>83.77</td>
<td>87.90</td>
</tr>
<tr>
<td>150</td>
<td>TAPAS</td>
<td>21335</td>
<td>18871</td>
<td>86.84</td>
<td>88.45</td>
</tr>
<tr>
<td>50</td>
<td>Cufflinks</td>
<td>25952</td>
<td>15117</td>
<td>69.56</td>
<td>58.25</td>
</tr>
<tr>
<td>100</td>
<td>Cufflinks</td>
<td>26032</td>
<td>16303</td>
<td>75.02</td>
<td>62.63</td>
</tr>
<tr>
<td>150</td>
<td>Cufflinks</td>
<td>25499</td>
<td>16779</td>
<td>77.21</td>
<td>65.80</td>
</tr>
<tr>
<td>50</td>
<td>IsoSCM</td>
<td>28152</td>
<td>11790</td>
<td>54.25</td>
<td>41.88</td>
</tr>
<tr>
<td>100</td>
<td>IsoSCM</td>
<td>29201</td>
<td>13583</td>
<td>62.51</td>
<td>46.52</td>
</tr>
<tr>
<td>150</td>
<td>IsoSCM</td>
<td>29600</td>
<td>14592</td>
<td>67.15</td>
<td>49.3</td>
</tr>
<tr>
<td>50</td>
<td>GETUTR</td>
<td>50818</td>
<td>15492</td>
<td>67.15</td>
<td>30.49</td>
</tr>
<tr>
<td>100</td>
<td>GETUTR</td>
<td>53226</td>
<td>16596</td>
<td>76.37</td>
<td>31.18</td>
</tr>
<tr>
<td>150</td>
<td>GETUTR</td>
<td>54577</td>
<td>17082</td>
<td>78.61</td>
<td>31.3</td>
</tr>
</tbody>
</table>

However, the simulation study in [102] assumed the abundance is distributed uniformly among all transcripts while we use a log normal distribution. Moreover, a slightly different (and more relaxed) criterion was used in [102] to define correctly identified APA sites. To make sure that we have installed/run IsoSCM correctly, we created a small dataset based on Chromosome 18 with deep coverage (1000 reads/kb) and uniform abundance distribution. Using the evaluation criterion in [102], IsoSCM was able to achieve 86.98% precision and 96.71% sensitivity, matching the results reported in [102].

We also compare the performance of the four tools for detecting APA sites on real data. We download paired-end RNA-Seq reads from standard polyA+ libraries for mouse brain (GSE41637) from NCBI. TopHat2 is able to map 85.4% of these reads to the reference genome (76189196 out of 87264604 reads). The mouse RefSeq annotation NCBI37 (mm9) is downloaded from the UCSC Genome Browser. For performance evaluation, a 3’-Seq dataset...
Figure 2.5: Number of correct APA sites detected by different tools on the real dataset when the flexible range for matching a predicted APA site to a true APA site of 3′-Seq is 50 bps (a) and 100 bps (b).

(BED file of annotated APA sites, GSM747481) for mouse (GSE30198) is also downloaded from NCBI and used as the benchmark, as done similarly in ([102]) and ([127]). We run the tools with the mapped reads and compare their predicted APA sites against the benchmark using two flexible ranges of 50 bps and 100 bps for matching. Here, we consider two flexible ranges because the default flexible range for TAPAS is 50 bps but 100 bps was used as the default range in IsoSCM ([102]). Among the 33751 APA sites reported in the 3′-Seq data, TAPAS, Cufflinks, IsoSCM, and GETUTR identify 10429, 5711, 6354, and 3111 APA sites, respectively, using the flexible range of 50 bps. When the flexible range is increased to 100 bps, TAPAS, Cufflinks, IsoSCM, and GETUTR identify 12224, 7956, 7680, and 6977 APA sites in the benchmark, respectively. Clearly, all tools found more true APA sites with more
Table 2.2: Performance comparison in APA site detection on real data. Two flexible ranges (50 bps and 100 bps) are considered for matching a predicted APA site with a true one from 3’-Seq.

<table>
<thead>
<tr>
<th>Number of true APA sites based on 3’-Seq</th>
<th>Tool name</th>
<th>Number of predicted APA sites</th>
<th>Correctly identified APA sites (50 bps)</th>
<th>Precision (%)</th>
<th>Correctly identified APA sites (100 bps)</th>
<th>Precision (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>33751</td>
<td>TAPAS</td>
<td>33816</td>
<td>10429</td>
<td>30.84</td>
<td>12224</td>
<td>36.15</td>
</tr>
<tr>
<td></td>
<td>Cufflinks</td>
<td>71502</td>
<td>5711</td>
<td>7.99</td>
<td>7956</td>
<td>11.13</td>
</tr>
<tr>
<td></td>
<td>IsoSCM</td>
<td>36286</td>
<td>6354</td>
<td>17.51</td>
<td>7680</td>
<td>21.17</td>
</tr>
<tr>
<td></td>
<td>GETUTR</td>
<td>62858</td>
<td>3111</td>
<td>4.95</td>
<td>6977</td>
<td>11.10</td>
</tr>
</tbody>
</table>

flexibility in matching. The detailed performance of the tools is illustrated in Figure 2.5 and Table 2.2. Note that TAPAS and Cufflinks predicted similar numbers of APA sites while IsoSCM and GETUTR predicted many more. Clearly, TAPAS outperforms the three other tools on this real dataset.

Table 2.3: Performance comparison in APA site detection on real data, when the prediction results of the tools compared are filtered by the 3’ UTR frames defined by TAPAS. Two flexible ranges (50 bps and 100 bps) are considered for matching a predicted APA site with a true one from 3’-Seq. The number of predicted APA sites of TAPAS is lowered to be closer to those of Cufflinks’ and IsoSCM’s. For a further comparison, Cufflinks is run with the reference transcriptome in RefSeq (i.e., Cufflinks -g). Note that, given the number of APA sites predicted by Cufflinks -g, its performance should be directly compared with that of TAPAS provided in Table S2 rather than the numbers in this table.

<table>
<thead>
<tr>
<th>Tool name</th>
<th>Number of predicted APA sites within frames</th>
<th>Correctly identified APA sites (50 bps flexible range)</th>
<th>Precision (%)</th>
<th>Correctly identified APA sites (100 bps flexible range)</th>
<th>Precision (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAPAS</td>
<td>16313</td>
<td>8764</td>
<td>53.72</td>
<td>9764</td>
<td>59.85</td>
</tr>
<tr>
<td>Cufflinks</td>
<td>8719</td>
<td>3534</td>
<td>40.53</td>
<td>5034</td>
<td>57.74</td>
</tr>
<tr>
<td>Cufflinks -g</td>
<td>23594</td>
<td>9884</td>
<td>41.89</td>
<td>10838</td>
<td>45.94</td>
</tr>
<tr>
<td>IsoSCM</td>
<td>10016</td>
<td>4569</td>
<td>45.62</td>
<td>5606</td>
<td>55.97</td>
</tr>
<tr>
<td>GETUTR</td>
<td>23347</td>
<td>2289</td>
<td>9.80</td>
<td>5452</td>
<td>23.35</td>
</tr>
</tbody>
</table>
Table 2.4: Performance comparison in detecting internal APA sites located inside the 3′ UTR frames on real data.

<table>
<thead>
<tr>
<th>Tool name</th>
<th>Correctly predicted internal APA sites (50 bps flexible range)</th>
<th>Sensitivity (%)</th>
<th>Correctly predicted internal APA sites (100 bps flexible range)</th>
<th>Sensitivity (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAPAS</td>
<td>7598</td>
<td>46.69</td>
<td>8302</td>
<td>51.01</td>
</tr>
<tr>
<td>Cufflinks</td>
<td>3906</td>
<td>24.00</td>
<td>5520</td>
<td>33.92</td>
</tr>
<tr>
<td>IsoSCM</td>
<td>4640</td>
<td>28.51</td>
<td>5586</td>
<td>34.32</td>
</tr>
<tr>
<td>GETUTR</td>
<td>2512</td>
<td>15.43</td>
<td>5579</td>
<td>34.28</td>
</tr>
</tbody>
</table>

Tables 2.3, 2.4 (and 2.2) show that this advantage of TAPAS remains true when the prediction results of the other tools are filtered by the 3′ UTR frames or only internal APA sites located inside 3′ UTR frames are considered. In particular, it still outperforms Cufflinks even if the latter is provided with the reference transcriptome in RefSeq.

Table 2.5: Performance comparison in APA site detection on real data. Two flexible ranges (50 bps and 100 bps) are considered for matching a predicted APA site with a true one from PAS-Seq.

<table>
<thead>
<tr>
<th>Number of true APA sites based on PAS-Seq</th>
<th>Tool name</th>
<th>Number of predicted APA sites</th>
<th>Correctly identified APA sites (50 bps)</th>
<th>Precision (%)</th>
<th>Correctly identified APA sites (100 bps)</th>
<th>Precision (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>50148</td>
<td>TAPAS</td>
<td>33816</td>
<td>26336</td>
<td>77.88</td>
<td>29346</td>
<td>86.78</td>
</tr>
<tr>
<td></td>
<td>Cufflinks</td>
<td>71502</td>
<td>12338</td>
<td>17.26</td>
<td>17290</td>
<td>24.18</td>
</tr>
<tr>
<td></td>
<td>IsoSCM</td>
<td>36286</td>
<td>17606</td>
<td>47.38</td>
<td>19919</td>
<td>54.89</td>
</tr>
<tr>
<td></td>
<td>GETUTR</td>
<td>62858</td>
<td>6253</td>
<td>9.95</td>
<td>15442</td>
<td>24.57</td>
</tr>
</tbody>
</table>

Similar to the 3′-Seq data, we also use mouse PAS-Seq data (BED file of annotated APA sites) from NCBI (GSE25450) as a benchmark for the performance evaluation of the tools using two flexible ranges (50 and 100 bps). This PAS-Seq dataset contains APA sites from mouse ES (embryonic stem), NPS (Neuropeptide S) and neuron cells. We extract
APA sites from neuron for our evaluation. Moreover, we consider only APA sites that are supported by four or more reads of PAS-Seq. Among the 50148 APA sites reported in the PAS-Seq data, TAPAS, Cufflinks, IsoSCM, and GETUTR identify 26336, 12338, 17606, and 6253 APA sites, respectively, using flexible range 50 bps and 29346, 17290, 19919, and 15442 APA sites, respectively, using flexible range 100 bps. The detailed results of the tools are given in Table 2.5. Clearly, TAPAS outperforms other the tools again with respect to this new benchmark. Although all tools have better performance on the PAS-Seq benchmark (because it contains more sites), the trends are similar on both benchmarks.

2.3.2 Performance on APA site-based differential expression analysis

In this section, we compare the performance of TAPAS with Cuffdiff [114], DESeq [7] and DEXSeq [6] in detecting differentially expressed genes on simulated data. Note that TAPAS’s differential expression analysis is based on APA sites while Cuffdiff’s, DESeq’s and DEXSeq’s are based on transcripts, genes and exons, respectively. Moreover, DEXSeq is designed for differential splicing (DS) rather than DE analysis ([76] and [104]). The data is simulated for two conditions as follows. For condition 1, the expression profile created in section 2.3.1 is taken as its initial expression profile. Genes with at least one major transcript (i.e., a transcript that has RPKM value greater than or equal to 1) in the expression profile are kept, similar to [131]. For these genes, only transcripts with different APA sites are selected for the analysis. This results in 12683 genes with a total of 14716 transcripts (and 14716 APA sites). For condition 2, 1254 (around 10% of all genes) genes are randomly selected as differentially expressed (DE) genes and a major transcript of each selected gene is chosen as a DE transcript (indirectly, making the APA site of that
transcript a DE APA site). Among the 1254 DE genes, 630 are designated as up-regulated and 624 as down-regulated. For each up-regulated gene, the abundance of its DE transcript is increased by a factor of four and for each down-regulated gene, the abundance of its DE transcript is decreased by a factor of four, similarly to [21] and [130]. For the other (non-DE) transcripts, their abundance levels are kept the same as in condition 1. This gives us the initial expression profiles of both conditions.

Given the RPKM value $\rho_{t,c}$ of a transcript $t$ in the initial expression profile for condition $c$, a negative binomial distribution $NB(\mu_{t,c},\sigma_{t,c}^2)$ is used to generate a set of RNA-Seq reads $r_{t,j}$ for each replicate $j$ of condition $c$. We generate six replicates for each condition. The mean and variance of the negative binomial distribution are $\mu_{t,c} = \rho_{t,c} \cdot l_t \cdot \hat{s}$ and $\sigma_{t,c} = \mu_{t,c} + \phi \cdot \mu_{t,c}^2$, respectively, where $l_t$ is the effective length of the transcript $t$ in kilo bps, $\hat{s}$ the size of the RNA-Seq library in millions and $\phi$ the dispersion. We simulate four RNA-Seq datasets by setting $\hat{s} = 30, 50, 100$ and $150$ million with $\phi = 0.179$, as done in [56]. A similar simulation procedure was also adopted in [131].

To compare the performance of TAPAS, Cuffdiff, DESeq, and DEXSeq in differential expression analysis, we run all four tools on the simulated datasets to detect DE genes based on the abundance of either APA sites, transcripts, genes, or exons, respectively. In order to make a thorough comparison, Cuffdiff is run with and without the transcriptome annotation. DEXSeq divides exons into “counting bins” (or expressed segments [70]) according to the overlapping structure of annotated transcripts, and outputs DE counting bins between samples. It can be regarded as either a tool for detecting DE APA sites where we only consider counting bins in front of each APA site or a tool for detecting DE genes...
Figure 2.6: Performance of TAPAS, Cuffdiff, DESeq, and DEXSeq in differential expression analysis in terms of sensitivity (a) and precision (b). Cuffdiff_ann denotes running Cuffdiff with the transcriptome annotation and DEXSeq_gene denotes running DEXSeq to detect DE genes (instead of DE APA sites).

where we consider all counting bins in a gene. We adopt the latter option to be consistent with the other tools. Since it is designed for DS rather than DE analysis, we consider only true DE genes with at least two transcripts (298 in total) as the benchmark for evaluating DEXSeq. The FDR value of 0.1 is used in Cuffdiff to call a DE transcript to make it comparable with TAPAS (which uses the adjusted p-value of 0.1). Similar to TAPAS, the adjusted p-values for DESeq and DEXSeq are set to 0.1. The performance of TAPAS, Cuffdiff, DESeq, and DEXSeq are summarized in Fig. 2.6. From the figure, we can see that DESeq has the best overall performance and DEXSeq has the worst overall performance. Both TAPAS and Cuffdiff with annotation perform better with the increase of sequencing
depth. In terms of sensitivity, DESeq outperforms the rest of the tools when the number of reads is less than 100 million. But, the sensitivity of TAPAS catches up quickly when the number of reads gets close to 100 million (perhaps helped by its improved performance on lowly expressed DE APA sites). TAPAS outperforms both Cuffdiffs when the number of reads reaches 50 million or more, even if it is given the transcriptome annotation. It also achieves a better precision than Cuffdiff without annotation. Although its precision is worse than that of Cuffdiff with annotation and DESeq, the gap closes rapidly with increased sequencing depth (again, perhaps helped by its improved performance on lowly expressed DE APA sites). The detailed results of TAPAS, Cuffdiff, DESeq, and DEXSeq are given in Table 2.6.
Table 2.6: Performance comparison in the detection of genes with differentially expressed (DE) APA sites on simulated data. The number of genes with actual DE APA sites is 1254, and each such gene contains only one DE APA sites. Since DEXSeq is designed for differential splicing (DS) rather than DE analysis [76, 104], we consider DE genes with at least two transcripts (298 in total) as the benchmark when evaluating the performance of DEXSeq. Here, Cuffdiff_anno = Cuffdiff with annotation.

<table>
<thead>
<tr>
<th>Dataset (in million)</th>
<th>Tool name</th>
<th>Number of detected genes with DE APA sites</th>
<th>Correctly identified genes with DE APA sites</th>
<th>Sensitivity (%)</th>
<th>Precision (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>TAPAS</td>
<td>1282</td>
<td>955</td>
<td>76.16</td>
<td>74.49</td>
</tr>
<tr>
<td>50</td>
<td>TAPAS</td>
<td>1329</td>
<td>1048</td>
<td>83.57</td>
<td>78.86</td>
</tr>
<tr>
<td>100</td>
<td>TAPAS</td>
<td>1308</td>
<td>1119</td>
<td>89.23</td>
<td>85.55</td>
</tr>
<tr>
<td>150</td>
<td>TAPAS</td>
<td>1317</td>
<td>1139</td>
<td>90.83</td>
<td>86.48</td>
</tr>
<tr>
<td>30</td>
<td>Cuffdiff</td>
<td>1377</td>
<td>999</td>
<td>79.67</td>
<td>72.55</td>
</tr>
<tr>
<td>50</td>
<td>Cuffdiff</td>
<td>1388</td>
<td>1011</td>
<td>80.62</td>
<td>72.84</td>
</tr>
<tr>
<td>100</td>
<td>Cuffdiff</td>
<td>1429</td>
<td>1017</td>
<td>81.10</td>
<td>81.10</td>
</tr>
<tr>
<td>150</td>
<td>Cuffdiff</td>
<td>1446</td>
<td>1012</td>
<td>80.70</td>
<td>69.99</td>
</tr>
<tr>
<td>30</td>
<td>Cuffdiff_anno</td>
<td>1158</td>
<td>1022</td>
<td>81.50</td>
<td>88.26</td>
</tr>
<tr>
<td>50</td>
<td>Cuffdiff_anno</td>
<td>1180</td>
<td>1046</td>
<td>83.41</td>
<td>88.64</td>
</tr>
<tr>
<td>100</td>
<td>Cuffdiff_anno</td>
<td>1188</td>
<td>1057</td>
<td>84.29</td>
<td>88.97</td>
</tr>
<tr>
<td>150</td>
<td>Cuffdiff_anno</td>
<td>1200</td>
<td>1063</td>
<td>84.47</td>
<td>88.58</td>
</tr>
<tr>
<td>30</td>
<td>DEXSeq</td>
<td>1202</td>
<td>1129</td>
<td>90.03</td>
<td>93.93</td>
</tr>
<tr>
<td>50</td>
<td>DEXSeq</td>
<td>1210</td>
<td>1144</td>
<td>91.23</td>
<td>94.55</td>
</tr>
<tr>
<td>100</td>
<td>DEXSeq</td>
<td>1197</td>
<td>1124</td>
<td>89.63</td>
<td>93.90</td>
</tr>
<tr>
<td>150</td>
<td>DEXSeq</td>
<td>1235</td>
<td>1141</td>
<td>90.99</td>
<td>92.39</td>
</tr>
<tr>
<td>30</td>
<td>DEXSeq</td>
<td>281</td>
<td>198</td>
<td>66.44</td>
<td>70.46</td>
</tr>
<tr>
<td>50</td>
<td>DEXSeq</td>
<td>278</td>
<td>211</td>
<td>70.81</td>
<td>75.90</td>
</tr>
<tr>
<td>100</td>
<td>DEXSeq</td>
<td>268</td>
<td>215</td>
<td>72.15</td>
<td>80.22</td>
</tr>
<tr>
<td>150</td>
<td>DEXSeq</td>
<td>273</td>
<td>216</td>
<td>72.48</td>
<td>79.12</td>
</tr>
</tbody>
</table>

It is interesting to observe that TAPAS is able to achieve a better overall performance than Cuffdiff with annotation when the sequencing depth is high in the experiment. This is because in the simulated datasets, the average number of APA sites contained in a gene is $\frac{14716}{12683} = 1.16$. Thus, most genes (and hence 3'UTR frames) contain just a single APA site. This makes the estimation of the abundance of an APA site quite easy (actually
trivial) while Cuffdiff still has to face the challenging problem of quantification, since the average number of annotated transcripts for each gene is $\frac{40923}{19150} = 2.35$.

Although an RNA-Seq based differential expression analysis is generally expected to perform better with the increase of sequencing depth [138], it is interesting to observe that Cuffdiff does not exhibit this behavior when the transcriptome annotation is not given. In fact, its performance decreases slightly when the sequencing depth is increased. This could be caused by Cuffmerge, which is used by Cuffdiff without annotation to merge assembled transcripts from different replicates. In particular, Cuffmerge tends to merge a transcript that is contained in another into the latter transcript. It may also merge two similar transcripts into one transcript. Both cases may result in the loss of transcripts in a sample and thus false DE genes. When the sequencing depth increases, more transcripts are assembled for each replicate and hence more transcripts could be merged.

In this simulation experiment, DESeq is able to outperform the other tools mostly because each simulated DE gene contains only one DE transcript. Although the performance of DEXSeq is worse than the other tools, it is generally consistent with the performance results reported in ([76] and [104]). Again, DEXSeq is designed for DS analysis instead of DE analysis. Although we used a different benchmark for DEXSeq in the experiment to account for this difference, our specific simulation procedure above might still have put DS analysis methods at a disadvantageous position since each DE gene is only required to have one major transcript.
2.3.3 Performance on detecting shortening/lengthening events

In this section, we compare the performance of TAPAS with two methods DaPars [127] and ChangePoint [122] in the literature for detecting genes with 3′ UTRs that shortened or lengthened between conditions on both simulated and real data. For the simulation study, similar to the above differential expression analysis, we need generate data for two conditions. For condition 1, the log normal distribution used in section 2.3.1 is used again to obtain the initial expression profile, but we now consider only genes with at least one transcript whose RPKM value is greater than or equal to 2 and keep these genes for further analysis. The number of such gene is 7033. For each of these genes, select a transcript \( t \) with RPKM value at least 2 and introduce another transcript \( t′ \) (called an artificial transcript) that is the same as \( t \) but with a 3′ UTR half as long as that of \( t \). We then divide the initial expression value of \( t \) evenly between \( t \) and \( t′ \). To create data for condition 2, 674 genes are randomly selected as differentially expressed. Moreover, we make sure that the APA site of the artificial transcript in each such gene is at least 100 bps upstream of the corresponding original transcripts. Here, the distance of 100 bps is chosen because we found that the distance between two APA sites given in the benchmark data is more than 100 bps. Among the 674 artificial transcripts, 340 are chosen to be up-regulated and 344 are down-regulated by a factor of four. The abundance of the other (non-DE) transcripts is kept the same as in condition 1. This gives us the initial expression profiles of both conditions.

Similar to section 2.3.2, six replicates per condition are generated using negative binomial distributions. Three different datasets are created with sequencing depths of 50, 100, 150 million reads, respectively. TAPAS, DaPars and ChangePoint are run on these
datasets to compare their performance. We do not include Cuffdiff here because we have run Cufflinks on the first dataset (50 million read) and found that it output only one APA site for most genes and failed to identify most of the artificial APA sites. A similar observation about Cufflinks was also be made in [102]. Since ChangePoint does not support multiple replicates, it is run with only one replicate from each condition. The FDR cutoffs for both DaPars and ChangePoint are set to 0.1, since TAPAS uses 0.1 as adjusted p-value cutoff.

The performance of the tools is summarized in Fig. 2.7. Again, the performance of all tools improve with the increase of sequencing depth. TAPAS outperforms the other two methods significantly. The poor performance of ChangePoint can probably be attributed to the fact it allows only one replicate per condition. (We also tried running ChangePoint by
pooling all replicates but its performance got even worse.) The details results of all three tools can be found in Table 2.7.

**Table 2.7:** Performance comparison in the detection of genes with shortening/lengthening events on simulated data. The actual number of genes with shortening/lengthening events is 674.

<table>
<thead>
<tr>
<th>Dataset (in million)</th>
<th>Tool name</th>
<th>Number of predicted event genes</th>
<th>Correctly determined event genes</th>
<th>Sensitivity (%)</th>
<th>Precision (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>TAPAS</td>
<td>598</td>
<td>444</td>
<td>65.88</td>
<td>74.25</td>
</tr>
<tr>
<td>100</td>
<td>TAPAS</td>
<td>632</td>
<td>502</td>
<td>74.48</td>
<td>79.43</td>
</tr>
<tr>
<td>150</td>
<td>TAPAS</td>
<td>631</td>
<td>506</td>
<td>75.07</td>
<td>80.19</td>
</tr>
<tr>
<td>50</td>
<td>DaPars</td>
<td>727</td>
<td>422</td>
<td>62.61</td>
<td>58.05</td>
</tr>
<tr>
<td>100</td>
<td>DaPars</td>
<td>645</td>
<td>426</td>
<td>63.20</td>
<td>66.05</td>
</tr>
<tr>
<td>150</td>
<td>DaPars</td>
<td>618</td>
<td>443</td>
<td>65.73</td>
<td>71.68</td>
</tr>
<tr>
<td>50</td>
<td>ChangePoint</td>
<td>421</td>
<td>125</td>
<td>18.55</td>
<td>29.69</td>
</tr>
<tr>
<td>100</td>
<td>ChangePoint</td>
<td>525</td>
<td>125</td>
<td>18.55</td>
<td>23.81</td>
</tr>
<tr>
<td>150</td>
<td>ChangePoint</td>
<td>509</td>
<td>138</td>
<td>20.47</td>
<td>27.11</td>
</tr>
</tbody>
</table>

We also compare the tools on a real dataset (RNA-Seq reads from standard polyA+ libraries) used in [127]. Four replicates of MAQC human brain (SRX016368, SRX016367, SRX016366, SRX016365) and MAQC UHR (SRX016372, SRX016371, SRX016370, SRX016369) data are downloaded from NCBI. The reads are then mapped to the reference human genome by TopHat2 to be used by the tools for shortening/lengthening analysis. To evaluate the performance, 3′-Seq datasets are downloaded for MAQC human brain (GSM747473 and GSM747474) and UHR (GSM747475 and GSM747476) from NCBI. Similar to [127], the significance of each APA site in the 3′-Seq data is assessed by using Fisher’s exact test, and only statistically significant APA sites are kept to create the benchmark of shortening/lengthening events by estimating the abundance of the APA sites using 3′-Seq reads.
and relative change values for each pair of APA sites as in equation 2.4 and applying the cutoff $|rc_{ij}| \geq 1.0$.

**Table 2.8:** Performance comparison in the detection of genes with shortening/lengthening events on real data.

<table>
<thead>
<tr>
<th>Tool name</th>
<th>Shortening/lengthening event gene identified by tool</th>
<th>Precision (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAPAS</td>
<td>872</td>
<td>61.7</td>
</tr>
<tr>
<td>DaPars</td>
<td>808</td>
<td>39.85</td>
</tr>
<tr>
<td>ChangePoint</td>
<td>734</td>
<td>34.33</td>
</tr>
</tbody>
</table>

On this real dataset, TAPAS reports 872 genes having shortening/lengthening events with a precision of 61.7%. On the other hand, DaPars and ChangePoint output 808 and 734 genes having shortening/lengthening events with precision values of 39.85% and 34.33%, respectively. Clearly, TAPAS outperforms the other two tools significantly. The detailed results are given in Table 2.8.

### 2.4 Discussion and time/memory efficiency

In this work, we have introduced TAPAS, a bioinformatics tool for detecting novel APA sites from standard RNA-Seq data. It is also capable of finding differentially expressed APA sites and genes with shortening/lengthening events. Our extensive experiments on both simulated and real data show that TAPAS performs better than all existing RNA-Seq based tools for APA site analysis. Compared with methods based on more dedicated experimental protocols such as 3P-Seq, standard RNA-Seq data is more abundant and easier to obtain. Moreover, a preliminary analysis of a 3P-Seq data against the corresponding 3’-Seq data and TAPAS prediction results suggests that although 3P-Seq may have a higher
sensitivity than TAPAS, it actually achieves a lower precision than TAPAS (see Table 2.9).

Hence, we expect that TAPAS will serve as a useful APA site analysis tool in biological research.

**Table 2.9:** Performance comparison between TAPAS and 3P-Seq in APA site detection on mouse liver data. Paired-end RNA-Seq reads from standard polyA+ libraries for mouse liver (SRX196268) were downloaded from NCBI and mapped by TopHat2 to the mouse genome. For performance evaluation, a 3′-Seq dataset for mouse liver (GSM747483) was also downloaded from NCBI and used as benchmark. We ran TAPAS on the mapped reads and compared its predicted APA sites against the benchmark. As a comparison, we downloaded the 3P-Seq data for mouse liver (GSM1268948) from NCBI. Among the 29932 APA sites reported in the 3-Seq data, TAPAS and 3P-Seq identified 10900 and 19480 sites, respectively. In terms of sensitivity, 3P-Seq outperforms TAPAS; but TAPAS outperforms 3P-Seq in terms of precision. Note that TAPAS uses standard RNA-Seq data which is very popular and easy to perform while 3P-Seq requires complex biological steps and large amounts of RNA for its analysis [55].

<table>
<thead>
<tr>
<th>Number of APA sites in 3′-Seq data</th>
<th>Tool name</th>
<th>Number of output APA sites</th>
<th>Overlap with 3′-Seq (100 bps flexible range)</th>
<th>Sensitivity (%)</th>
<th>Precision (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>29932</td>
<td>TAPAS</td>
<td>25147</td>
<td>10900</td>
<td>36.42</td>
<td>43.35</td>
</tr>
<tr>
<td>3P-Seq</td>
<td>82551</td>
<td></td>
<td>19480</td>
<td>65.08</td>
<td>23.60</td>
</tr>
</tbody>
</table>

Although both TAPAS and Cufflinks are capable of finding novel APA sites, TAPAS relies on a transcriptomic or genomic annotation while Cufflinks can assemble transcripts from scratch. As a result, Cufflinks may potentially discover novel APA sites that are not found by TAPAS, especially because TAPAS only searches in 3′ UTR frames. Hence, one may consider combining the output of both TAPAS and Cufflinks to increase the coverage of novel APA sites.
Table 2.10: Comparison of time (in minutes) and peak memory (in gigabytes) usage among the APA site detection tools on the simulated dataset with 50 million reads used in Section 2.3.1. Here, the running time of TAPAS includes the calculation of read coverage by SAMtools.

<table>
<thead>
<tr>
<th>Tool name</th>
<th>Time (min)</th>
<th>Memory (GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAPAS</td>
<td>121</td>
<td>16.62</td>
</tr>
<tr>
<td>Cufflinks</td>
<td>97</td>
<td>1.00</td>
</tr>
<tr>
<td>IsoSCM</td>
<td>103</td>
<td>3.67</td>
</tr>
<tr>
<td>GETUTR</td>
<td>106</td>
<td>19.78</td>
</tr>
</tbody>
</table>

Since the efficiency of a bioinformatics tool is critical to its practical utility, we also compare TAPAS with the existing tools in term of time and memory efficiency. Similar to the study in the previous section, we divide the comparison into two groups: comparison between the APA site detection tools and comparison between the shortening/lengthening analysis tools. Our computation platform is a high-end computer cluster, where each node has 32 Intel Broadwell cores and 512 GB memory. We compare the four APA site detection tools (i.e., TAPAS, IsoSCM, GETUTR, and Cufflinks) on the simulated dataset with 50 million reads as considered in section 2.3.1 based on sequential running time (i.e., using a single core) and peak memory usage. As shown in Table 2.10, although TAPAS requires a significant amount of memory, its running time is comparable to that of the other three tools. We then compare the running time and memory efficiency of the three tools for shortening/lengthening event detection (i.e., TAPAS, DaPars and ChangePoint) on the dataset with 50 million reads as considered in section 2.3.3. It can be seen from Table 2.11 that TAPAS requires more time and memory than DaPars when it is run on a single core, but the gap in running time can be significantly reduced when more cores are used (one core per replicate) since DaPars is unable to take advantage of parallelism. TAPAS
is significantly more efficient than ChangePoint in both running time and peak memory usage. Also note that there is no option to parallelize ChangePoint and with the current setting we are unable to make its operation parallelize.

**Table 2.11:** Comparison of time and peak memory usage among the tools for shortening/lengthening analysis on the simulated dataset with 50 millions reads used in Section 2.3.3. Again, the running time of TAPAS includes the calculation of read coverage by SAMtools.

<table>
<thead>
<tr>
<th>Tool name</th>
<th>Time (min)</th>
<th>Memory (GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAPAS</td>
<td>803</td>
<td>7.70</td>
</tr>
<tr>
<td>TAPAS (parallel)</td>
<td>81</td>
<td>7.70</td>
</tr>
<tr>
<td>DaPars</td>
<td>49</td>
<td>3.99</td>
</tr>
<tr>
<td>ChangePoint</td>
<td>1876</td>
<td>19.55</td>
</tr>
</tbody>
</table>
Algorithm 3 Detection and removal of change points around a well.

procedure FilterChangePointsAroundWell(cp, coverage, strand)

Input:

- \( cp \rightarrow \) change points of a 3’ UTR frame. These change points divide the frame into segments
- \( coverage \rightarrow \) read coverage of the 3’ UTR frame
- \( strand \rightarrow \) strand of the 3’ UTR frame

\[ M \leftarrow \text{mean coverage of segments} \]

\[ \text{if } \text{strand} = \text{positive} \text{ then} \]

\[ \text{for each mean } m_i \text{ in } M \text{ do} \]

\[ \text{if } m_{i-1} > m_i < m_{i+1} \text{ then} \]

\[ \text{if } m_{i-1} = m_{i+1} \text{ then} \]

\[ \text{remove change points between } m_{i-1}, m_i \text{ and } m_i, m_{i+1} \]

\[ \text{else if } m_{i-1} > m_{i+1} \text{ then} \]

\[ \text{remove change points between } m_i \text{ and } m_{i+1} \]

\[ \text{else} \]

\[ \text{remove change point between } m_{i-1}, m_i \text{ and } m_i, m_{i+1} \]

\[ \text{else} \]

\[ \text{for each mean } m_i \text{ in } M \text{ do} \]

\[ \text{if } m_{i-1} > m_i < m_{i+1} \text{ then} \]

\[ \text{if } m_{i-1} = m_{i+1} \text{ then} \]

\[ \text{remove change points between } m_{i-1}, m_i \text{ and } m_i, m_{i+1} \]

\[ \text{else if } m_{i-1} < m_{i+1} \text{ then} \]

\[ \text{remove change points between } m_{i-1} \text{ and } m_i \]

\[ \text{else} \]

\[ \text{remove change point between } m_{i-1}, m_i \text{ and } m_i, m_{i+1} \]
Algorithm 4 EM algorithm for estimating the abundance of alternative 3’ UTRs.

procedure ABUNDANCECALCULATOR($T$, $l$, $R$)

Input:
- $T$ → set of all possible alternative 3’ UTRs in a frame
- $l$ → set of lengths of those alternative 3’ UTRs
- $R$ → set of reads mapped in the 3’ UTR frame

Assign random values to all $\rho_t$, where $t \in T$ and $\rho_t$ is the abundance of $t$

while not converged do

initialize all $\text{read}_t$ to 0, where $\text{read}_t$ is the read count for $t$

for each read $r$ in $R$ do

$T_r$ → set of alternative 3’ UTRs containing read $r$

for each alternative 3’ UTR $t$ ($t \in T_r$) do

$\text{read}_t = \text{read}_t + \frac{\rho_t}{\sum_{u \in T_r} \rho_u}$

$s = \sum_{t \in T} \frac{\text{read}_t}{(l_j - l_r + 1)}$

for each alternative 3’ UTR $t$ do

$\rho_t = \frac{\text{read}_t}{(l_j - l_r + 1) \times s}$

end for

end for

end while

$RC \leftarrow$ calculate the abundance (read counts) of all the 3’ UTRs (of the given frame) from $\rho$

Output: $RC$
Chapter 3

DeepPASTA: deep neural network based polyadenylation site analysis

3.1 Introduction

According to the central dogma of molecular biology, the genomic sequence of an eukaryotic gene is transformed into the corresponding protein by the transcription, post-transcriptional and translation processes. Initially, the transcription process converts a gene into a pre-mRNA, then this pre-mRNA is transformed into a mature mRNA by the post-transcriptional process and finally the mRNA is translated into the corresponding protein by the translation process. One of the important steps of the post-transcriptional process is the addition of a polyadenylation (polyA) tail at the 3’ end of a pre-mRNA. More specifically, the polyadenylation process consists of two steps [120]: cleavage near the 3’ end of a pre-mRNA and addition of a polyA tail at the cleavage site or polyA site.
Alternative cleavage sites near the 3′ end of a pre-mRNA create more than one mRNA transcript containing 3′ untranslated regions (3′ UTRs) of different lengths. A 3′ UTR is a suffix of an mRNA sandwiched between the stop codon and polyA site of the mRNA. The length of a 3′ UTR as well as some sequence elements (such as those AU and GU rich elements) may have impact on mRNA stability, mRNA localization, protein translation, protein binding and translation efficiency [15]. E.g. longer 3′ UTRs may have additional destabilization elements that alter the respective transcript’s stability [101] and in cancers, transcripts with shorter 3′ UTR can escape regulation from microRNAs ([72, 33]). Moreover, the secondary structure of a 3′ UTR is also important for translation efficiency and disruption of expression [15]. Alternative polyadenylation is very common in mammalian genes [111] and more than a half of human genes have alternative polyadenylation in their post-transcription process [81]. Moreover, errors in 3′-end processing may cause several inherited diseases [30]. Therefore, the prediction and analysis of polyA sites would be of great importance in the study of mammalian genes.

Several cis-elements and trans-factors influence the choice of a polyA or cleavage site ([15, 94]). The most important cis-element for a polyA site is the hexamer or polyadenylation signal (PAS), which usually occurs 10-30 nt upstream of the cleavage site [1, 13, 24, 31, 74, 97, 129, 110, 111]. The PAS serves as a binding site for the cleavage and polyadenylation specificity factor (CPSF) [29]. A polyA site also depends on the U or U/G-rich elements and these elements occur 20-40 nt downstream of that polyA site ([1, 24, 111, 31]). These U or U/G-rich elements serve as the binding sites for the cleavage stimulation factor (CstF) [29]. In addition, some auxiliary elements upstream of the PAS
and downstream of the cleavage site may enhance the polyadenylation process ([1, 46, 111]). Therefore, a polyA site typically depends on 4 different cis-elements: auxiliary upstream elements (AUEs), the upstream hexamer signal (i.e., PAS), downstream U/GU rich elements, and auxiliary downstream elements (ADEs). Moreover, the RNA secondary structures near the downstream region of mammalian polyadenylation signals impact the choice of polyA sites [126, 20].

Several tools have been introduced in the literature to predict polyA sites or PASs from human genomic sequences. DNAFSMiner [73, 74] predicts PASs from sequences using k-mer features in a support vector machine (SVM) model. Dragon PolyA Spotter [13] also predicts PASs from sequences using both an artificial neural network and a random forest. POLYAH [97] discriminates real PASs from other hexamer signals using a linear discriminant function. It focuses on only one PAS (AATAAA) in its analysis, although other PASs (variants of AATAAA) may influence polyA site selection. Polyadq [110] uses a quadratic linear discriminant function to predict real PAS regions. This tool considers only two signals (A(A/T)TAAA) in its analysis. However, a polyA site not only depends on the upstream PAS but also downstream U/GU rich elements, AUEs and DAEs. PolyA_svm [24] predicts polyA sites from sequences using a SVM model. PolyAR [1] also predicts polyA sites from sequences using a linear discriminant function. However, these tools use hand-picked sequence features. In order to overcome the limitation of hand-picked sequence features, deep learning models such as DeepPolyA [38], DeeReCT-PolyA [128] and Conv-Net [68] have been recently introduced to predict polyA sites, PASs and relatively dominant polyA sites (i.e., more frequently used polyA sites in a given gene). These models use all
convolution neural networks (CNNs) to extract features from the input genomic sequence. Although the secondary structure near a polyA site is essential for the polyA site to be selected for the polyadenylation process [126, 14, 20], none of these tools consider RNA secondary structures in their prediction procedures.

Polyadenylation occurs in a tissue specific manner [134, 112, 41, 125]. Different tissues show bias in selecting the locations of polyA sites within a gene, such as sites located in introns, internal exons and the last exon [134]. Different polyA sites in the last exon may result in mRNAs with different 3′ UTRs. On the other hand, the usage of polyA sites located in introns or internal exons may lead to the creation of premature stop codons or truncated proteins. Therefore, predicting the locations of tissue-specific polyA sites is important for understanding tissue-specific behaviors, variable 3′ UTRs and protein products [134].

One way to study tissue-specific choices of polyA sites is to consider the usages of different polyA sites. In a last exon, polyA sites closest to the 5′ and 3′ ends are called the proximal and distal polyA sites, respectively [134]. There are other polyA sites in between proximal and distal polyA sites, and these sites are called middle polyA sites [134]. Placenta, retina, blood, testis, and ovary tissues show preference for proximal polyA sites, i.e., high usage of proximal and low usage of distal polyA sites. On the other hand, bone marrow, uterus, ear, brain, the nervous system, and pancreatic islet show high usage of distal polyA sites [134]. Therefore, it would be interesting to predict relatively dominant polyA sites for a given gene to understand tissue specific behaviors. Conv-Net [68] is the first published tool to analyze relative dominance of polyA sites in human tissues. More specifically, the
tool takes a couple of polyA sites within a 3′ UTR and predicts the dominant polyA site using a deep learning algorithm.

In this chapter, we introduce a new tool, called DeepPASTA (i.e., Deep neural network based PolyA SiTe Analysis), to predict polyA sites from sequences and RNA secondary structures. As secondary structure near a polyA site is important for the polyA site selection [126, 20], DeepPASTA is the first tool to consider both sequence and RNA secondary structure in polyA site prediction. It employs both a CNN and a recurrent neural network (RNN). The CNN extracts features from sequences [3, 9, 51, 139] and secondary structures. On the other hand, the RNN is used to combine the effects of upstream and downstream signals [29, 119] in polyA site prediction. As the polyadenylation process is tissue-specific, we also formulate tissue-specific polyA site prediction as a multi-label classification problem where the usage of a polyA site is simultaneously analyzed for multiple tissues, and extend DeepPASTA to solve this problem. Similar to Conv-Net [68], DeepPASTA can also predict relatively dominant polyA sites of a gene in a specific tissue. We further generalize the relative dominance problem so DeepPASTA can also predict the most dominant polyA sites for each gene (i.e., the absolute dominance problem).

To assess the performance of DeepPASTA, we have conducted extensive experiments on human genomic sequence data and compared DeepPASTA with the above mentioned tools including PolyAR, Dragon PolyA Spotter, DeepPolyA, DeeReCT-PolyA, and Conv-Net for polyA site prediction or relative dominance. As none of the existing tools are able to perform all four types of polyA site analysis that DeepPASTA can do, we organize the comparisons as four groups: (i) prediction of human polyA sites (between DeepPASTA,
PolyAR, Dragon PolyA Spotter, DeepPolyA, DeeReCT-PolyA, and Conv-Net), (ii) prediction of human tissue-specific polyA sites (between DeepPolyA, the tissue-specific and non-tissue-specific DeepPASTA models), (iii) prediction of relatively dominant polyA sites (between DeepPASTA and Conv-Net), and (iv) prediction of absolute dominant polyA sites (between DeepPASTA and Conv-Net). The tools are compared in term of area under the curve (AUC) and area under the precision recall curve (AUPRC). Based on these two performance measures, DeepPASTA outperforms the other tools significantly in polyA site prediction. For tissue-specific relatively dominant polyA site prediction, DeepPASTA achieves better AUCs and AUPRCs than Conv-Net on most of the human tissues. In tissue-specific absolute dominant polyA site prediction, DeepPASTA again outperforms Conv-Net on all human tissues.

The rest of the chapter is organized as follows. The four different models of DeepPASTA are discussed in Section 3.2. The experimental results and comparisons with the other tools are discussed in Section 3.3. Section 3.3 also explains the sequence and secondary structure data generation procedure. At the end, conclusion is drawn in Section 3.4.

### 3.2 Materials and methods

In this section, we describe the four models of DeepPASTA for predicting polyA sites, tissue-specific polyA sites, tissue-specific relative dominance between polyA sites, and tissue-specific absolutely dominant polyA sites from human genomic sequence and RNA secondary structure data. The four prediction problems are formally defined as follows.
The first problem is a binary classification problem that takes a genomic sequence of 200 nts [1, 68] and some probable secondary structures predicted by RNAshapes [108] as the input and expects a score as the output indicating the likelihood for the middle position of the input sequence to be a polyA site. Note that RNAshapes is used here because it is one of the most popular tools for RNA secondary structure prediction [80, 135]. The second problem is a multi-label classification problem that takes a sequence and some corresponding RNA secondary structures as the input and asks which tissues may have polyA sites in the input sequence for a given set of tissues. The third problem is a multi-class classification problem that takes two polyA sites surrounding sequences (200 nts) as well as corresponding RNA secondary structures of a gene as the input and estimates the relatively dominant polyA site in a particular tissue. The final problem is a binary classification problem that takes a polyA site and its surrounding sequence (200 nts) as well as corresponding RNA secondary structure of a gene as the input and outputs a score indicating the likelihood for the input polyA site to be the absolutely dominant polyA site of the gene. The detailed input and output of the above four models are illustrated in Figure 3.1 and 3.2.
Figure 3.1: The input and output of the polyA site prediction and tissue-specific polyA site prediction models of DeepPASTA. a) The polyA site prediction model of DeepPASTA takes a genomic sequence of 200 nts and three energy efficient RNA secondary structures predicted by RNAshapes [108] from the sequence as the input and predicts whether the input sequence contains a polyA site at the middle or not. b) Similar to the previous model, the tissue-specific polyA site prediction model of DeepPASTA takes a sequence and three corresponding secondary structures generated by RNAshapes as the input and predicts whether the input sequence contains a polyA site at the middle or not for the nine tissues studied in [31].
Figure 3.2: The input and output of the tissue-specific relatively and absolutely dominant polyA site prediction models of DeepPASTA. a) The tissue-specific relatively dominant polyA site prediction model of DeepPASTA takes a couple of sequences and corresponding secondary structures containing polyA sites of some gene at the middle as the input and predicts which polyA site is relatively dominant. b) Unlike the relatively dominant model, the absolutely dominant model of DeepPASTA takes a sequence and corresponding secondary structure containing a polyA site of some gene at the middle as the input and predicts whether the polyA site is an absolutely dominant polyA site of the gene.

Recently, deep learning has been applied in bioinformatics with superior performance over conventional learning methods on many prediction/classification problems, such as protein-nucleotide binding prediction [3, 90, 135], functional genomic data prediction [37], translation initiation site [137], and ribosome stalling prediction [136]. Following these state-of-the-art methods, DeepPASTA also uses deep learning algorithms in its prediction models. Each of the four models of DeepPASTA employs both a CNN for extracting features and an RNN for combining the effects of these features. The four models are explained in detail in the following subsections.
3.2.1 Predicting polyA sites

The first model of DeepPASTA is the polyA site prediction model. The model takes a genomic sequence of 200 nts and some corresponding RNA secondary structures as the input to predict whether that sequence has a polyA site in the middle or not. Following the literature [135], three energy efficient RNA secondary structures are generated by RNAshapes [108] from the sequence and given as a part of the input. Figure 3.3 shows the overall architecture of the model. The model consists of four sub-models: a sequence sub-model and three identical secondary structure sub-models. The sequence sub-model starts with a convolution layer [66]. Following the work [3, 9, 51, 139], the convolution layer of DeepPASTA is used to extract features from the input sequence based on a sliding window. It uses a rectified linear unit (ReLU) [84] as the activation function to set negative values to zero. The next layer is a max pooling layer [28] that picks the maximum feature value within a window. After the max pooling layer, a bidirectional LSTM (long short term memory) recurrent layer [99, 43, 39] is used to consider both upstream and downstream signals for polyA site prediction. The last layer of the sequence sub-model is a fully connected layer. Each of the three input RNA secondary structures is fed to a secondary structure sub-model. Similar to the sequence sub-model, each secondary structure sub-model starts with a convolution layer to extract features from the input secondary structure. This convolution layer is followed by an average pooling layer. The average pooling layer calculates the average of all the feature values within a window. The next two layers of the sub-model are a bidirectional LSTM and a fully connected layer. The three secondary structure sub-models are combined using an addition layer and then concatenated with the sequence
sub-model. The concatenation layer is followed by multiple fully connected layers. The fully connected layers of the polyA site prediction model use ReLUs as the activation function. The model ends with a single neuron output layer with a sigmoid activation function. In order to prevent data overfitting, dropouts [107] are used in some of the layers.

Figure 3.3: Architectures of the polyA site prediction model of DeepPASTA, M3 and M4. The polyA site prediction model has four sub-models: a sequence and three secondary structure sub-models. Each sub-model consists of a convolution layer, a maxpooling layer, a recurrent layer (i.e., a bi-directional LSTM), a flattening layer, and a fully connected layer. On the other hand, M3 (model represented by the red dotted line) consists of a sequence sub-model. M4 (model represented by the yellow dotted line) is similar to M3, but its sequence sub-model does not contain a recurrent layer.
Separate training and validation data are used to train the model, while some test data is used to evaluate the performance of the trained model. Ground truth values of the training and validation data are taken from the PolyA-Seq data in [31]. The PolyA-Seq data provides tissue specific polyA sites in human. For the basic (i.e., non-tissue specific) polyA site prediction problem, we take the union of all the tissues to construct the ground truth data. Figure 3.4 shows the steps of the training phase. The model is trained using the Adam RMSprop with Nesterov momentum [35] optimizer. It uses a minibatch size of 1000 to minimize the average multi-task binary cross entropy loss on the training data. At the end of each training epoch, the validation loss is evaluated to monitor convergence. In order to expedite the learning process, a graphic processing unit (GPU) is used.
Figure 3.4: The training phase of the polyA site prediction model of DeepPASTA. In each iteration of the training phase, the model predicts a likelihood value for the given input. This prediction is compared with the ground truth using a loss function. The loss value is then used to tune the parameters of the deep learning model.

3.2.2 Predicting tissue-specific polyA sites

If a sequence and its corresponding RNA secondary structures are given as the input, the tissue-specific polyA site prediction model asks in which tissues the sequence contains a polyA site. The model is a multi-label classifier [5] that simultaneously considers nine different human tissues: brain, kidney, liver, maqc\_brain1, maqc\_brain2, maqc\_UHR1, maqc\_UHR2, muscle, and testis [31]. Following the literature [68], we consider the four samples maqc\_brain1, maqc\_brain2, maqc\_UHR1, and maqc\_UHR2 as four different tissues. The output from the multi-label classifier is a vector of nine values for the nine tissues. If
a tissue has a polyA site in the input sequence then the tissue’s corresponding value in the vector should be 1, otherwise 0. Note that the model also works for input sequence that does not contain any polyA site for the nine tissues. Here, our model considers nine tissues because the ground truth data, PolyA-Seq [31], involves nine tissues, but it can be applied to any set of tissues that has ground truth data.

**Figure 3.5:** Architecture of the tissue-specific polyA site prediction model of DeepPASTA. Similar to the polyA site prediction model of DeepPASTA, this model has a sequence and three secondary structure sub-models. Each of these sub-models consists of a convolution layer, a maxpooling layer, a recurrent layer, a flattening layer, and a fully connected layer. This model is a multi-label classification model that has nine neurons in the output layer for predicting polyA sites in the nine tissues studied in [31].
Figure 3.5 shows the overall architecture of the model. Similar to the basic (i.e., non-tissue-specific) polyA site prediction model described in Section 3.2.1, this model has one sequence and three identical secondary structure sub-models. The sequence and secondary structure sub-models are similar to the sequence and secondary structure sub-models of the basic polyA site prediction model, but they use parametric ReLUs (PReLU) [42] as the activation functions in its convolution and fully connected layers. The four sub-models are then combined using two concatenation layers. The latest concatenation layer is followed by multiple fully connected layers and these fully connected layers use ReLUs as the activation functions. The final layer of the model has nine output neurons for the nine tissues and it uses a sigmoid activation function. Dropouts are used in some of the layers to prevent overfitting.

The training process of this model is similar to the training process of the basic polyA site prediction model.

3.2.3 Predicting tissue-specific relatively dominant polyA sites

When a couple of sequences of 200 nts (and corresponding RNA secondary structures) containing polyA sites of some gene in a particular tissue are given as the input, this model predicts which polyA site is more dominant in the tissue (i.e., more frequently used; [68]). Note that we do not define this model as a multi-label classifier because the input sequence may not contain polyA sites in all tissues. Figure 3.6 shows the overall architecture of the model. The (relative dominance) strength value of each input sequence (and RNA secondary structure) is calculated using a sub-unit of the model. This sub-unit consists of two sub-models: sequence and secondary structure sub-models. The sequence
sub-model consists of one convolution layer with a PReLU activation function, one max pooling layer, one bidirectional LSTM recurrent layer, and one fully connected layer with a ReLU activation function. The architecture of the secondary structure sub-model is similar to the sequence sub-model, but uses an average pooling layer in place of the max pooling layer. The sequence and secondary structure sub-models are combined using a concatenation layer. The concatenation layer is followed by multiple fully connected layers. The sub-unit ends with an output layer with a single neuron that provides the strength value of the input sequence. At the end, the relatively more dominant polyA site is determined by comparing the output strength values of the two input sequences from the two sub-units using softmax [18].
Figure 3.6: Architecture of the model of DeepPASTA for predicting relative dominance in a particular tissue. The model takes two sequences of 200 nts and corresponding secondary structures generated by RNAshapes containing polyA sites of some gene at the middle as the input. Each of these sequences and secondary structures is processed by a sub-unit, which consists of a sequence and a secondary structure sub-models. The output layer compares the outputs from the two sub-units to predict the relatively dominant polyA site.

In order to train the model, the read counts of the two input sequences from the PolyA-Seq data [31] are used to construct the ground truth. More specifically, if the input sequences $S_1$ and $S_2$ have $R_1$ and $R_2$ read counts, then the ground truth strengths of these two sequences are $\frac{1+R_1}{2+R_1+R_2}$ and $\frac{1+R_2}{2+R_1+R_2}$, respectively. A similar procedure is also followed to calculate the ground truth in Conv-Net [68]. The training process of the relative
dominant polyA site prediction model is similar to the training process of the polyA site prediction model.

3.2.4 Predicting tissue-specific absolutely dominant polyA sites

When a sequence of 200 nts (and corresponding RNA secondary structure) containing a polyA site of some gene in a particular tissue is given as the input, this model predicts whether the polyA site is a most dominant site (i.e., the most frequently used) of the gene or not in the involved tissue. Usually, the most dominant polyA sites of a gene are more likely selected for the polyadenylation process. Again, we do not define this model as a multi-label classifier because the input sequence may not contain polyA sites in all tissues. Figure 3.7 shows the overall architecture of the model. The absolutely dominant prediction model has two sub-models: sequence and secondary structure sub-models. The sequence and secondary structure sub-models of this model are similar to the sequence and secondary structure sub-models of the relative dominance prediction model. These two sub-models are also combined using a concatenation layer. The concatenation layer is followed by multiple fully connected layers with ReLU activation functions. The final layer of the model has one output neuron and the activation function of this layer is sigmoid.
Figure 3.7: Architecture of the model of DeepPASTA for predicting absolutely dominate polyA sites of each gene in a particular tissue. The model has a sequence and a secondary structure sub-models. The output layer predicts whether the input polyA site is an absolutely dominant polyA site or not.

The read count values of polyA sites from the PolyA-Seq data are used to determine one or more absolutely dominant polyA sites of a gene. PolyA sites with the maximum read counts within a gene are considered as the (absolutely) dominant polyA sites and rest are considered as the non-dominant sites. The model is trained using similar steps as the other models described above.
3.3 Experimental results

In this section, we compare the performance of DeepPASTA with that of some state-of-the-art methods for predicting polyA sites, tissue-specific polyA sites as well as relatively/absolutely dominant polyA sites. We also compare the tissue-specific polyA site prediction model with the non-tissue-specific model.

In order to construct the sequence data for DeepPASTA’s models, polyA sites are collected from the PolyA-Seq experiments in [31]. As AUEs, the PAS, U/GU rich elements, and ADEs are typically within 100 nts upstream and downstream of a polyA site [1, 46], the genomic sequence of length 200 nts centered around a polyA site is taken from the human GRCh37 (hg19) reference genome (similar to [68]). These sequences are considered as the positive examples for the deep learning models. The models also need negative examples for training and testing. Therefore, four different sets of negative examples are constructed: two sets obtained by shifting each positive example left and right by 50 nts [68], random sequences containing upstream hexamer signals, and random sequences from coding and noncoding regions of genes [1]. The length of each of these negative examples is 200 nts. In the shifted negative examples, the polyA sites are not in the middle of the 200 nts sequences. In the negative examples with hexamer signals, the hexamer signals are in the upstream region of the sequences. Similar to the literature [68, 38, 137], these sequence examples are then fed to DeepPASTA by using the one-hot encoding representation. As there are four possible nucleotides (A, C, G, and T) in DNA sequences, the dimensionality of a sequence example is 4 x 200.
Since the selection of polyA sites in polyadenylation process also depends on the RNA secondary structures near the polyA sites [20, 126]. DeepPASTA considers both sequences and their corresponding RNA secondary structures for its prediction tasks as mentioned above. RNAsshapes [108] is used to predict the probable secondary structures of each sequence example [135, 80]. More specifically, the sequence is scanned using a sliding window (of size 100 nts) and a step size (of 100 nts) to predict first level abstract (i.e., the most detailed) representation/secondary structures [64]. As done in [135], the three most energy efficient secondary structures of the sequence are recorded for future analyses. Each position of the secondary structure is represented by one of seven symbols i.e., \( L, R, U, M, H, I, \) and \( E \), which stand for left hand base of a double strand, right hand base of a double strand, unpaired base, multiloop, hairpin loop, internal loop, and external region, respectively. Similar to the sequence input, the secondary structure input uses a one-hot encoding representation. Therefore, the dimensionality of a secondary structure input is \( 7 \times 200 \).

### 3.3.1 Performance on predicting polyA sites

In this experimental study, we compare DeepPASTA with five existing tools: PolyAR [1], Dragon PolyA Spotter [13], DeeReCT-PolyA [128], Conv-Net [68], and DeepPolyA [38] for predicting polyA sites on three datasets (to be introduced below). In order to train the model of DeepPASTA, we partition the human chromosomes into three groups: chromosomes 1 to 8 as group 1, chromosomes 9 to 14 as group 2, and chromosomes 15 to Y as group 3. Homologous genes from BioMart of Ensembl are considered to prevent potential data leak (i.e., training data containing information of test data). Using the genes from
chromosomes 1 to 8, homologous genes are extracted from chromosomes 9 to Y and are added into group 1. Similarly, using the genes from chromosomes 9 to 14, homologous genes are extracted from chromosomes 15 to Y and are added into group 2. The polyA sites in groups 1, 2 and 3 are collected from the PolyA-Seq data. Before homologous genes are moved, group 1, 2 and 3 have 251726, 125665 and 144208 polyA sites, respectively. After moving homologous genes, the number of polyA sites in group 1, 2 and 3 are 326695, 99498 and 95406, respectively. We then construct training, validation and test data from groups 1, 2 and 3, respectively. As mentioned above, four different types of negative sequences and their corresponding RNA secondary structures are collected from group 1 and down-sampled to make the ratio of positive and negative example as 1 : 1 in the training data. The down-sampling helps make the prediction model more robust [73, 13, 1]. A similar procedure is also followed to construct the validation data. The polyA site prediction model of DeepPASTA is then trained using the training and the hyperparameters of the model are tuned empirically using held-out validation data. The test data is constructed similarly, but we do not down-sample the negative examples (thus keeping the ratio of positive and negative examples as 1:4).

We use three test datasets to evaluate the performance of the tools. Datasets 1 and 2 are constructed from the test data and dataset 3 is taken from the literature [68]. Dataset 1 is constructed using the whole test data. Therefore, it contains 95406 positive and 381555 negative examples (ratio 1 : 4). Dataset 2 is a subset of dataset 1 and it consists of 95406 (i.e., all) positive examples and 95406 random sequences as the negative examples. We include both balanced data (dataset 2) and unbalanced data (dataset 1)
in the performance evaluation because PolyAR and Dragon PolyA Spotter use balanced datasets in their performance evaluations [1, 13] but in reality, the number of polyA sites is very small compared to the whole human genome. Note that datasets 1 and 2 do not contain any information about the training and validation data.

In order to compare with the most recent method Conv-Net [68] directly, we construct dataset 3 by considering only chromosomes 15 to Y. We do not consider chromosomes 1 to 14 because the polyA site prediction model of DeepPASTA is trained on those chromosomes. We introduce dataset 3 in the performance evaluation because we want to show the performance of DeepPASTA not only on the PolyA-Seq data but also on the dataset from Conv-Net literature. We collect the positive sequences of dataset 3 from [68]. As in [68], the negative sequences are constructed by shifting each positive example left and right by 50 nts. Therefore, the numbers of positive and negative sequences in dataset 3 are 6018 and 12036, respectively. For each sequence, three energy efficient RNA secondary structures are constructed using RNAshapes.

As the Conv-Net model is not publicly available, we construct the model using the description in [68] and train it using the sequences of the above training and validation data. We also train DeepPolyA using the same training and validation data because the tool was initially developed for plants. Since the tool Dragon PolyA Spotter needs sequences of length more than 200 nts as the input, we extend each sequence by 50 nts in both directions to make it 300 nts for Dragon PolyA Spotter. The performance of all the tools are compared using AUC and AUPRC.
<table>
<thead>
<tr>
<th>Tool name</th>
<th>Performance metric</th>
<th>Dataset 1</th>
<th>Dataset 2</th>
<th>Dataset 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeepPASTA</td>
<td>AUC</td>
<td>0.972</td>
<td>0.958</td>
<td>0.930</td>
</tr>
<tr>
<td></td>
<td>AUPRC</td>
<td>0.921</td>
<td>0.962</td>
<td>0.875</td>
</tr>
<tr>
<td>PolyAR</td>
<td>AUC</td>
<td>0.630</td>
<td>0.713</td>
<td>0.673</td>
</tr>
<tr>
<td></td>
<td>AUPRC</td>
<td>0.296</td>
<td>0.749</td>
<td>0.489</td>
</tr>
<tr>
<td>Dragon PolyA Spotter</td>
<td>AUC</td>
<td>0.609</td>
<td>0.711</td>
<td>0.639</td>
</tr>
<tr>
<td></td>
<td>AUPRC</td>
<td>0.261</td>
<td>0.693</td>
<td>0.421</td>
</tr>
<tr>
<td>DeeReCT-PolyA</td>
<td>AUC</td>
<td>0.637</td>
<td>0.711</td>
<td>0.659</td>
</tr>
<tr>
<td></td>
<td>AUPRC</td>
<td>0.261</td>
<td>0.695</td>
<td>0.421</td>
</tr>
<tr>
<td>Conv-Net</td>
<td>AUC</td>
<td>0.910</td>
<td>0.899</td>
<td>0.907*</td>
</tr>
<tr>
<td></td>
<td>AUPRC</td>
<td>0.782</td>
<td>0.913</td>
<td>0.853</td>
</tr>
<tr>
<td>DeepPolyA</td>
<td>AUC</td>
<td>0.925</td>
<td>0.913</td>
<td>0.906</td>
</tr>
<tr>
<td></td>
<td>AUPRC</td>
<td>0.804</td>
<td>0.922</td>
<td>0.854</td>
</tr>
</tbody>
</table>

* The AUC performance of Conv-Net on dataset 3 is taken from [68].

From Table 3.1, it can be seen that DeepPASTA clearly outperforms the other tools in polyA site prediction. DeepPolyA and Conv-Net perform better than PolyAR, Dragon PolyA Spotter, and DeeReCT-PolyA because they also use machine extracted features and deep learning algorithms. DeepPolyA preforms slightly better than Conv-Net perhaps because Conv-Net was originally designed to predict relatively dominant polyA sites. Dragon PolyA Spotter and DeeReCT-PolyA only predict hexamer signals in sequences, but a polyA site depends on other signals as well as the hexamer signal. As a result, Dragon PolyA Spotter and DeeReCT-PolyA perform the worst among the tools. Because PolyAR considers other signals along with the hexamer signal in its prediction process, it is able to perform better than Dragon PolyA Spotter and DeeReCT-PolyA. Moreover, all the tools generally perform better on dataset 2 than on dataset 1 as expected, but DeepPASTA drops slightly in AUC on dataset 2. In order to explain this, we conduct several experiments with different
sets of negative examples as described above. We find that DeepPASTA performs the best on the shifted negative examples and it performs better on negative examples with hexamer signals than random negative examples (Figure 3.8). As dataset 2 does not contain shifted negative examples and negative examples with hexamer signals, its AUC drops slightly. On the dataset from the Conv-Net work (dataset 3), DeepPASTA clearly performs better than Conv-Net in both AUC and AUPRC. Hence, although both DeepPASTA, Conv-Net, and DeepPolyA use deep learning algorithms, DeepPASTA performs significantly better than Conv-Net and DeepPolyA on all three datasets. The use of RNA secondary structures and recurrent neural networks in the model architecture may have helped improve the performance of DeepPASTA.

Two examples illustrating the contribution of RNA secondary structures are given in Figure 3.9. Moreover, the hexamer signals that contributed to the performance of DeepPASTA in polyA site prediction are analyzed in Figure 3.10.
Figure 3.8: The impact of negative examples on the performance of DeepPASTA. In order to test the performance of DeepPASTA in predicting polyA sites on different negative examples, three datasets are considered: datasets with shifted negative examples where positive examples are shifted left and right by 50 bases, with random negative examples that do not contain the hexamer signal and with random negative examples containing the hexamer signal. The positive examples of these datasets are the same. The number of examples for these three datasets are 286218, 190812 and 190744, respectively. Plots in a show the AUC and AUPRC performance of DeepPASTA on the dataset with the shifted negative examples. Plots in b show the AUC and AUPRC performance on the dataset with the random examples that do not contain the hexamer signal. Plots in c show the AUC and AUPRC performance on the dataset with the negative examples containing the hexamer signal.
**Figure 3.9:** The RNA secondary structures of genes COX4I1 and ADORA2B helped DeepPASTA in predicting polyA sites. The figure shows the secondary structures generated by RNAshapes for the 100-nt upstream sequences of some polyA sites of the genes. Both polyA sites have AATAAA as the polyadenylation signal (PAS), but the locations of the signal in each input are far away from the polyA sites (the PASs and the polyA sites are colored red in the sequences). It is well known that the PAS often occurs 10-30 nts upstream of a polyA site ([46], [31] and [111]). Hence, one might conjecture that a PAS has to be near a polyA site in order for it to be functional. The folding of the RNA secondary structures reduces the distance between the PAS and polyA site in each gene. Similar phenomena are also described in [14].
Figure 3.10: Hexamer signals extracted from the true positive polyA sites predicted by DeepPASTA on dataset 1. In order to identify the most frequently used signals, we consider the top three high strength 6-mers in each input sequence based on saliency maps. The barplot on the left shows the overall 20 most frequently used hexamer signals in polyA site prediction. Most of these signals are annotated in the literature [46], [31] and [111]. In addition, DeepPASTA used some novel hexamer signals: UAAAAU, GAAUAAA, UAAAUA, AAUUAA, and UUAAA. The four barplots on the right show the most frequently used hexamer signals in four equally divided regions (as illustrated at the bottom of the figure) of the input sequence. From the four barplots, it is seen that DeepPASTA used fewer signals from the fourth region (150-200 nts) in polyA site prediction. Similar to previous studies, DeepPASTA identified the U-rich signals as auxiliary upstream elements (AUEs) in the first region (1-49 nts), U/GU-rich signals as downstream elements in the third region (101-149 nts) and G-rich signals as auxiliary downstream elements (ADEs) in the fourth region (150-200 nts).
Table 3.2: The effect of data leak on DeepPASTA in polyA site prediction on dataset 1 in terms of AUC and AUPRC.

<table>
<thead>
<tr>
<th>PolyA site prediction model</th>
<th>AUC</th>
<th>AUPRC</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeepPASTA (homologs consolidated)</td>
<td>0.9724</td>
<td>0.9210</td>
</tr>
<tr>
<td>M2 (homologs not consolidated)</td>
<td>0.9725</td>
<td>0.9211</td>
</tr>
</tbody>
</table>

Effect of data leak on polyA site prediction

In order to test the effect of data leak on our polyA site prediction method, we compare the performance of our DeepPASTA model trained in the above (where all homologous genes were consolidated in the training and validation data to prevent potential data leak in testing) with another one where the homologous genes are not consolidated. More specifically, we construct the training and validation data based on chromosomes similarly as above but we do not move homologous genes from dataset 1 to the training and validation data. As a result, the numbers of positive examples in the training and validation data are 251726 and 125665, respectively. Again, the negative examples are down-sampled to make the ratio of positive and negative examples 1 : 1 in this training and validation data. For convenience, let us refer to the DeepPASTA model trained with the new data as M2. The performance of the two models are compared using AUC and AUPRC on dataset 1 in Table 3.2. As shown in the table, there is only slight performance difference between the models. This negligible difference demonstrates that homology does not cause serious data leak in our polyA site prediction method.
Table 3.3: Contributions of the RNN and RNA secondary structures in polyA site prediction on datasets 1 and 2 in terms of AUC and AUPRC.

<table>
<thead>
<tr>
<th>PolyA site prediction model</th>
<th>Dataset</th>
<th>AUC</th>
<th>AUPRC</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeepPASTA (sequences + RNA secondary structures)</td>
<td>Dataset 1</td>
<td>0.972</td>
<td>0.921</td>
</tr>
<tr>
<td></td>
<td>Dataset 2</td>
<td>0.958</td>
<td>0.962</td>
</tr>
<tr>
<td>M3 (sequences with CNN and RNN)</td>
<td>Dataset 1</td>
<td>0.960</td>
<td>0.893</td>
</tr>
<tr>
<td></td>
<td>Dataset 2</td>
<td>0.938</td>
<td>0.947</td>
</tr>
<tr>
<td>M4 (sequences with CNN)</td>
<td>Dataset 1</td>
<td>0.951</td>
<td>0.871</td>
</tr>
<tr>
<td></td>
<td>Dataset 2</td>
<td>0.931</td>
<td>0.940</td>
</tr>
</tbody>
</table>

Performance improvement using an RNN and RNA secondary structures

In order to test how the use of an RNN and RNA secondary structures may contribute to the performance of DeepPASTA, we consider three polyA site prediction models (see Figure 3.3): (i) the full polyA site prediction model of DeepPASTA, (ii) the model that uses only the sequence features (called M3) and (iii) the model that uses the sequence features and CNN (called M4). The models M3 and M4 are trained using the same training and validation data as in the training of full model of DeepPASTA. Note that these models have much less network complexity than DeepPASTA. The performance of the models are evaluated using AUC and AUPRC on datasets 1 and 2 in Table 3.3. The table shows that both RNN and RNA secondary structures make small (but non-negligible) contributions to the improved performance of DeepPASTA.

3.3.2 Performance on predicting tissue-specific polyA sites

Different human tissues can have different polyA sites [112] and the tissue specificity of polyA sites has been studied extensively in the literature [134, 112, 41, 125]. In this subsection, we analyze the performance of the DeepPASTA (multi-label) model for tissue-
specific polyA site prediction and compare it with that of the basic (i.e., non-tissue-specific) model and DeepPolyA [38]. Similar to the training of the basic model, the tissue-specific polyA site prediction model is also trained by consolidating homologous genes to prevent potential data leak. The training and validation data of this model are similar to the training and validation data of the basic model, but their ground truths are different. For each example in the training and validation data, the ground truth consists of nine labels (actually, score values) for nine tissues (brain, kidney, liver, maqc_brain1, maqc_brain2, maqc_UHR1, maqc_UHR2, muscle, and testis), indicating if a tissue is likely to have a polyA site or not. Similar to the basic model, the hyperparameters of the tissue-specific model are tuned empirically using held-out validation data. The performance of the tissue-specific model is compared with DeepPolyA and the basic model on datasets 1 and 2 using AUC and AUPRC in Table 3.4. While evaluating the tissue-specific performance, if an input sequence contains a polyA site in the middle for a given tissue then the sequence is a positive example for that tissue, otherwise its a negative example.
### Table 3.4: Performance comparison between the tissue-specific model of DeepPASTA, DeepPolyA and basic (i.e., non-tissue-specific) polyA site prediction models of DeepPASTA on datasets 1 and 2. Table S1 of the Supplementary Materials shows the numbers of positive and negative examples in the test datasets. Datasets 1 and 2 are represented as D1 and D2, respectively, in the table.

<table>
<thead>
<tr>
<th>Tissue</th>
<th>Data</th>
<th># of positive examples</th>
<th># of negative examples</th>
<th>DeepPolyA</th>
<th>Tissue-specific</th>
<th>Basic model</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>AUC</td>
<td>AUPRC</td>
<td>AUC</td>
</tr>
<tr>
<td>Brain</td>
<td>D1</td>
<td>18226</td>
<td>458735</td>
<td>0.883</td>
<td>0.202</td>
<td>0.921</td>
</tr>
<tr>
<td></td>
<td>D2</td>
<td>18226</td>
<td>172585</td>
<td>0.777</td>
<td>0.224</td>
<td>0.823</td>
</tr>
<tr>
<td>Kidney</td>
<td>D1</td>
<td>18557</td>
<td>458404</td>
<td>0.892</td>
<td>0.214</td>
<td>0.927</td>
</tr>
<tr>
<td></td>
<td>D2</td>
<td>18557</td>
<td>172254</td>
<td>0.789</td>
<td>0.235</td>
<td>0.834</td>
</tr>
<tr>
<td>Liver</td>
<td>D1</td>
<td>16231</td>
<td>460730</td>
<td>0.878</td>
<td>0.170</td>
<td>0.916</td>
</tr>
<tr>
<td></td>
<td>D2</td>
<td>16231</td>
<td>174580</td>
<td>0.767</td>
<td>0.189</td>
<td>0.813</td>
</tr>
<tr>
<td>MAQC</td>
<td>D1</td>
<td>18286</td>
<td>458675</td>
<td>0.899</td>
<td>0.220</td>
<td>0.938</td>
</tr>
<tr>
<td></td>
<td>D2</td>
<td>18286</td>
<td>172525</td>
<td>0.801</td>
<td>0.241</td>
<td>0.857</td>
</tr>
<tr>
<td>Brain1</td>
<td>D1</td>
<td>17194</td>
<td>459767</td>
<td>0.898</td>
<td>0.207</td>
<td>0.937</td>
</tr>
<tr>
<td></td>
<td>D2</td>
<td>17194</td>
<td>173617</td>
<td>0.799</td>
<td>0.227</td>
<td>0.855</td>
</tr>
<tr>
<td>MAQC</td>
<td>D1</td>
<td>20807</td>
<td>456154</td>
<td>0.892</td>
<td>0.237</td>
<td>0.928</td>
</tr>
<tr>
<td></td>
<td>D2</td>
<td>20807</td>
<td>170004</td>
<td>0.791</td>
<td>0.261</td>
<td>0.839</td>
</tr>
<tr>
<td>MAQC</td>
<td>D1</td>
<td>22166</td>
<td>454795</td>
<td>0.892</td>
<td>0.250</td>
<td>0.928</td>
</tr>
<tr>
<td></td>
<td>D2</td>
<td>22166</td>
<td>168645</td>
<td>0.792</td>
<td>0.276</td>
<td>0.840</td>
</tr>
<tr>
<td>Muscle</td>
<td>D1</td>
<td>20706</td>
<td>456255</td>
<td>0.877</td>
<td>0.207</td>
<td>0.910</td>
</tr>
<tr>
<td></td>
<td>D2</td>
<td>20706</td>
<td>170105</td>
<td>0.765</td>
<td>0.232</td>
<td>0.806</td>
</tr>
<tr>
<td>Testis</td>
<td>D1</td>
<td>21270</td>
<td>455691</td>
<td>0.876</td>
<td>0.211</td>
<td>0.908</td>
</tr>
<tr>
<td></td>
<td>D2</td>
<td>21270</td>
<td>169541</td>
<td>0.764</td>
<td>0.237</td>
<td>0.802</td>
</tr>
</tbody>
</table>

From the table, we can see that the tissue-specific model of DeepPASTA performs better than DeepPolyA and the non-tissue-specific model (of DeepPASTA) in predicting tissue-specific polyA sites. In fact, the two DeepPASTA models significantly outperform DeepPolyA in tissue-specific polyA site prediction. Although, the AUC differences between the DeepPASTA models are very small, the improvements in AUPRC are quite significant. Clearly, the use of all nine tissues simultaneously in training of the tissue-specific model has helped its performance. Although, the AUCs of both models decrease from dataset 1 to
dataset 2, their AUPRCs increase slightly. The reason of this performance variation between datasets 1 and 2 is due to the types of negative examples in the datasets (similar to the AUC variation observed in Section 3.3.1). Here, a sequence with the polyadenylation signals may not have polyA sites in every tissue. This makes the prediction task much harder for the tissue-specific model of DeepPASTA on both datasets. Moreover, the AUPRC values are much lower than the AUC values because the negative examples greatly outnumber the positive examples on both datasets. Similar to the basic model, the hexamer signals that contributed to the performance of DeepPASTA in tissue-specific polyA site prediction are analyzed in Figures 3.11–3.19.
Figure 3.11: Hexamer signals extracted from the true positive polyA sites predicted by the tissue-specific model of DeepPASTA on dataset 1 for the brain tissue. In order to identify the most frequently used signals, we consider the top three high strength 6-mers in each input sequence based on saliency maps. The barplot on the left shows the overall 20 most frequently used hexamer signals in polyA site prediction for the brain tissue. The four barplots on the right show the most frequently used hexamer signals in four equally divided regions (as illustrated at the bottom of the figure) of the input sequence. From these four barplots, it is seen that DeepPASTA used the most hexamer signals from the second region (50-100 nts) in polyA site prediction for the brain tissue. The most frequently used signals in that region are AAUAAA, AAAAAA, AAAAAG, AAUAAA, and CAAUAA. These signals are known as the polyadenylation signals (PASs) in the literature [46], [31] and [111].
Figure 3.12: Hexamer signals extracted from the true positive polyA sites predicted by the tissue-specific model of DeepPASTA on dataset 1 for the kidney tissue. In order to identify the most frequently used signals, we consider the top three high strength 6-mers in each input sequence based on saliency maps. The barplot on the left shows the overall 20 most frequently used hexamer signals in polyA site prediction for the kidney tissue. The four barplots on the right show the most frequently used hexamer signals in four equally divided regions (as illustrated at the bottom of the figure) of the input sequence. From these four barplots, it is seen that DeepPASTA used the most hexamer signals from the second region (50-100 nts) in polyA site prediction for the kidney tissue. The most frequently used signals in that region are AAUAAA, AUAAAA, AAAUAA, AUAAAG, and CAAUAA. Again, these signals are known as the polyadenylation signals (PASs) in the literature [46], [31] and [111].
Figure 3.13: Hexamer signals extracted from the true positive polyA sites predicted by the tissue-specific model of DeepPASTA on dataset 1 for the liver tissue. In order to identify the most frequently used signals, we consider the top three high strength 6-mers in each input sequence based on saliency maps. The barplot on the left shows the overall 20 most frequently used hexamer signals in polyA site prediction for the liver tissue. The four barplots on the right show the most frequently used hexamer signals in four equally divided regions (as illustrated at the bottom of the figure) of the input sequence. From these four barplots, it is seen that DeepPASTA used the most hexamer signals from the second region (50-100 nts) in polyA site prediction for the liver tissue. The most frequently used signals in that region are AAUAAA, AUAAAA, AAAUAA, AUAAAG, and AUUAAA. Again, these signals are known as the polyadenylation signals (PASs) in the literature [46], [31] and [111].
Figure 3.14: Hexamer signals extracted from the true positive polyA sites predicted by the tissue-specific model of DeepPASTA on dataset 1 for the MAQC_Brain1 tissue. In order to identify the most frequently used signals, we consider the top three high strength 6-mers in each input sequence based on saliency maps. The barplot on the left shows the overall 20 most frequently used hexamer signals in polyA site prediction for the MAQC_Brain1 tissue. The four barplots on the right show the most frequently used hexamer signals in four equally divided regions (as illustrated at the bottom of the figure) of the input sequence. From these four barplots, it is seen that DeepPASTA used the most hexamer signals from the second region (50-100 nts) in polyA site prediction for the MAQC_Brain1 tissue. The most frequently used signals in that region are AAUAAA, AAAUAA, AUAAAA, AUUAAA, and AUAAAG. Again, these signals are known as the polyadenylation signals (PASs) in the literature [46], [31] and [111].
Figure 3.15: Hexamer signals extracted from the true positive polyA sites predicted by the tissue-specific model of DeepPASTA on dataset 1 for the MAQC_Brain2 tissue. In order to identify the most frequently used signals, we consider the top three high strength 6-mers in each input sequence based on saliency maps. The barplot on the left shows the overall 20 most frequently used hexamer signals in polyA site prediction for the MAQC_Brain2 tissue. The four barplots on the right show the most frequently used hexamer signals in four equally divided regions (as illustrated at the bottom of the figure) of the input sequence. From these four barplots, it is seen that DeepPASTA used the most hexamer signals from the second region (50-100 nts) in polyA site prediction for the MAQC_Brain2 tissue. The most frequently used signals in that region are AAUAAA, AAAUAA, AUAAAG, CAAUAA, and UAAUAA. Again, these signals are known as the polyadenylation signals (PASs) in the literature [46], [31] and [111].
Figure 3.16: Hexamer signals extracted from the true positive polyA sites predicted by the tissue-specific model of DeepPASTA on dataset 1 for the MAQC_UHR1 tissue. In order to identify the most frequently used signals, we consider the top three high strength 6-mers in each input sequence based on saliency maps. The barplot on the left shows the overall 20 most frequently used hexamer signals in polyA site prediction for the MAQC_UHR1 tissue. The four barplots on the right show the most frequently used hexamer signals in four equally divided regions (as illustrated at the bottom of the figure) of the input sequence. From these four barplots, it is seen that DeepPASTA used the most hexamer signals from the second region (50-100 nts) in polyA site prediction for the MAQC_UHR1 tissue. The most frequently used signals in that region are AAUAAA, AAAUAA, AUAAAA, AUUAAA, and UAAUAA. Again, these signals are known as the polyadenylation signals (PASs) in the literature [46], [31] and [111].
Figure 3.17: Hexamer signals extracted from the true positive polyA sites predicted by the tissue-specific model of DeepPASTA on dataset 1 for the MAQC_UHR2 tissue. In order to identify the most frequently used signals, we consider the top three high strength 6-mers in each input sequence based on saliency maps. The barplot on the left shows the overall 20 most frequently used hexamer signals in polyA site prediction for the MAQC_UHR2 tissue. The four barplots on the right show the most frequently used hexamer signals in four equally divided regions (as illustrated at the bottom of the figure) of the input sequence. From these four barplots, it is seen that DeepPASTA used the most hexamer signals from the second region (50-100 nts) in polyA site prediction for the MAQC_UHR2 tissue. The most frequently used signals in that region are AAUAAA, AAAUAA, AUUAAA, AUAAAA, and UAAUAA. Again, these signals are known as the polyadenylation signals (PASs) in the literature [46], [31] and [111].
Figure 3.18: Hexamer signals extracted from the true positive polyA sites predicted by the tissue-specific model of DeepPASTA on dataset 1 for the muscle tissue. In order to identify the most frequently used signals, we consider the top three high strength 6-mers in each input sequence based on saliency maps. The barplot on the left shows the overall 20 most frequently used hexamer signals in polyA site prediction for the muscle tissue. The four barplots on the right show the most frequently used hexamer signals in four equally divided regions (as illustrated at the bottom of the figure) of the input sequence. From these four barplots, it is seen that DeepPASTA used the most hexamer signals from the second region (50-100 nts) in polyA site prediction for the muscle tissue. The most frequently used signals in that region are AAUAAA, AUUAA, AAAUAA, AUAAAA, and AUAAAG. Again, these signals are known as the polyadenylation signals (PASs) in the literature [46], [31] and [111].
Figure 3.19: Hexamer signals extracted from the true positive polyA sites predicted by the tissue-specific model of DeepPASTA on dataset 1 for the testis tissue. In order to identify the most frequently used signals, we consider the top three high strength 6-mers in each input sequence based on saliency maps. The barplot on the left shows the overall 20 most frequently used hexamer signals in polyA site prediction for the testis tissue. The four barplots on the right show the most frequently used hexamer signals in four equally divided regions (as illustrated at the bottom of the figure) of the input sequence. From these four barplots, it is seen that DeepPASTA used the most hexamer signals from the second region (50-100 nts) in polyA site prediction for the testis tissue. The most frequently used signals in that region are AAUAAA, AUAAAA, AAAUAA, AUAAAG, and AUUAAA. Again, these signals are known as the polyadenylation signals (PASs) in the literature [46], [31] and [111].
3.3.3 Performance on predicting tissue-specific relatively dominant polyA sites

In this subsection, we compare DeepPASTA with Conv-Net in predicting relatively dominant polyA sites in a particular tissue on two datasets: datasets 4 and 5. Dataset 4 contains nine tissues studied above [31] and for each tissue, the human chromosomes are partitioned into three groups: chromosomes 1 to 6 as group 1, chromosomes 7 to 12 as group 2 and chromosomes 13 to Y as group 3. Again, homologs are consolidated to prevent potential data leak. All pairs of polyA sites of a particular gene from the PolyA-Seq data in a specific tissue are considered as examples (ordered by their genomic locations). We construct training, validation and test data from the examples in groups 1, 2, and 3, respectively. The tissue-specific read counts (from the PolyA-Seq data) are used to define the true relative dominance. For simplicity, we do not consider examples consisting of polyA sites with equal read counts. For each tissue, a model is trained and tested using the data of that tissue. Dataset 5 is taken from [68]. The training, validation and test parts of dataset 5 are constructed following the same construction steps as for dataset 4. Note that homologs are not consolidated in this dataset. There are eight tissues in this dataset and for each tissue, a model is again trained and tested using the data of that tissue. Again, the hyperparameters of the models are tuned empirically using held-out validation data. For each tissue of datasets 4 and 5, a Conv-Net model is trained with the same training and validation data as DeepPASTA. Figure 3.20 shows the numbers of training and validation examples in datasets 4 and 5. The performance of DeepPASTA and Conv-Net in predicting tissue-specific relative dominance is compared using AUC and AUPRC in Tables 3.5 and
3.6. Clearly, DeepPASTA achieves a better overall performance and its improvement over Conv-Net is consistent across both datasets.

Figure 3.20: Number of examples in the training and validation data used in the experiments on predicting tissue-specific relatively dominant polyA sites. As shown in the left plot, the number of training examples in dataset 4 ranges from 59.4% to 64.3% of the total number of examples (used in training, validation and testing) across all tissues, and the number of validation examples ranges from 15.8% to 19.7%. As shown in the right plot, the numbers of training and validation examples range from 60.5% to 61.7% and from 22.3% to 22.8%, respectively.
Table 3.5: Performance comparison between DeepPASTA and Conv-Net in relatively dominant polyA site prediction on dataset 4 in terms of AUC and AUPRC. The performance of Conv-Net is based on our implementation of the method described in [68].

<table>
<thead>
<tr>
<th>Tissue</th>
<th># of Test Examples</th>
<th>DeepPASTA</th>
<th>Conv-Net</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>AUC</td>
<td>AUPRC</td>
</tr>
<tr>
<td>Brain</td>
<td>38726</td>
<td>0.748</td>
<td>0.729</td>
</tr>
<tr>
<td>Kidney</td>
<td>44363</td>
<td>0.708</td>
<td>0.694</td>
</tr>
<tr>
<td>Liver</td>
<td>39832</td>
<td>0.713</td>
<td>0.698</td>
</tr>
<tr>
<td>MAQC_Brian1</td>
<td>44242</td>
<td>0.723</td>
<td>0.707</td>
</tr>
<tr>
<td>MAQC_Brian2</td>
<td>40878</td>
<td>0.709</td>
<td>0.694</td>
</tr>
<tr>
<td>MAQC_UHR1</td>
<td>62064</td>
<td>0.704</td>
<td>0.704</td>
</tr>
<tr>
<td>MAQC_UHR2</td>
<td>62946</td>
<td>0.721</td>
<td>0.707</td>
</tr>
<tr>
<td>Muscle</td>
<td>49528</td>
<td>0.719</td>
<td>0.706</td>
</tr>
<tr>
<td>Testis</td>
<td>53820</td>
<td>0.733</td>
<td>0.714</td>
</tr>
</tbody>
</table>

Table 3.6: Performance comparison between DeepPASTA and Conv-Net [68] in relatively dominant polyA site prediction on dataset 5 using AUC and AUPRC.

<table>
<thead>
<tr>
<th>Tissue</th>
<th># of Test Examples</th>
<th>DeepPASTA</th>
<th>Conv-Net</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>AUC</td>
<td>AUPRC</td>
</tr>
<tr>
<td>Brain</td>
<td>10567</td>
<td>0.908</td>
<td>0.905</td>
</tr>
<tr>
<td>Breast</td>
<td>10905</td>
<td>0.900</td>
<td>0.899</td>
</tr>
<tr>
<td>ES cell</td>
<td>8351</td>
<td>0.910</td>
<td>0.912</td>
</tr>
<tr>
<td>Ovary</td>
<td>10146</td>
<td>0.903</td>
<td>0.900</td>
</tr>
<tr>
<td>SK muscle</td>
<td>8816</td>
<td>0.906</td>
<td>0.905</td>
</tr>
<tr>
<td>Testis</td>
<td>10456</td>
<td>0.893</td>
<td>0.890</td>
</tr>
<tr>
<td>BCells1</td>
<td>8674</td>
<td>0.905</td>
<td>0.906</td>
</tr>
<tr>
<td>BCells2</td>
<td>8118</td>
<td>0.901</td>
<td>0.896</td>
</tr>
</tbody>
</table>

* The AUC performance of Conv-Net is taken from [68].
3.3.4 Performance on predicting tissue-specific absolutely dominant polyA sites

In this subsection, we compare the performance of DeepPASTA and Conv-Net in predicting absolutely dominant polyA sites in a particular tissue. Similar to dataset 4 constructed in a previous subsection, the human chromosomes are partitioned and homologs are consolidated to construct a new dataset, called dataset 6. Among all the polyA sites of each gene, those that have the highest read counts in the PolyA-Seq data with respect to a particular tissue are considered as the absolutely dominant polyA sites of the gene (in the tissue). The rest of the polyA sites are considered as non-dominant polyA sites of the gene (i.e., negative examples) in the tissue. Figure 3.21 shows the numbers of training and validation examples in dataset 6. The performance of DeepPASTA in predicting absolutely dominant polyA sites is compared with Conv-Net using AUC and AUPRC on the test data of dataset 6, as shown in Table 3.7. DeepPASTA clearly outperforms Conv-Net in all tissues. This significant performance improvement of DeepPASTA can be partially attributed to its use of RNA secondary structures.
Figure 3.21: Number of examples in the training and validation data used in the experiments on predicting tissue-specific absolutely dominant polyA sites of each gene. As the plot shows, the number of examples in the training data ranges from 54.1% to 55.9% and the number of examples in the validation data ranges from 22.6% to 23.4%.

Table 3.7: Performance comparison between DeepPASTA and Conv-Net in predicting absolutely dominant polyA sites on dataset 6 in terms of AUC and AUPRC. The 2nd and 3rd columns give the number of positive and negative examples in the test data.

<table>
<thead>
<tr>
<th>Tissue</th>
<th>Pos #</th>
<th>Neg #</th>
<th>DeepPASTA</th>
<th>Conv-Net</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>AUC</td>
<td>AUPRC</td>
</tr>
<tr>
<td>Brain</td>
<td>5420</td>
<td>11859</td>
<td>0.703</td>
<td>0.533</td>
</tr>
<tr>
<td>Kidney</td>
<td>5362</td>
<td>12964</td>
<td>0.698</td>
<td>0.508</td>
</tr>
<tr>
<td>Liver</td>
<td>5006</td>
<td>11911</td>
<td>0.688</td>
<td>0.496</td>
</tr>
<tr>
<td>MAQC_Brain1</td>
<td>5365</td>
<td>13015</td>
<td>0.720</td>
<td>0.517</td>
</tr>
<tr>
<td>MAQC_Brain2</td>
<td>5256</td>
<td>12196</td>
<td>0.723</td>
<td>0.531</td>
</tr>
<tr>
<td>MAQC_UHR1</td>
<td>5633</td>
<td>16018</td>
<td>0.699</td>
<td>0.466</td>
</tr>
<tr>
<td>MAQC_UHR2</td>
<td>5762</td>
<td>16885</td>
<td>0.712</td>
<td>0.476</td>
</tr>
<tr>
<td>Muscle</td>
<td>5545</td>
<td>14240</td>
<td>0.693</td>
<td>0.488</td>
</tr>
<tr>
<td>Testis</td>
<td>5553</td>
<td>14669</td>
<td>0.693</td>
<td>0.485</td>
</tr>
</tbody>
</table>
3.4 Discussion

In this chapter, we introduced DeepPASTA, a deep learning based tool for predicting polyA sites from genomic sequence and RNA secondary structure data. The tool is also capable of predicting tissue-specific polyA sites as well as tissue-specific relatively and absolutely dominant polyA sites. Our extensive experiments show that DeepPASTA performs better than all existing tools in all four polyA site analyses. Table 3.8 illustrates that the four polyA site prediction models of DeepPASTA can be trained in a reasonable amount of time. Hence, we expect that DeepPASTA will serve as a useful polyA site analysis tool in biological research.
Table 3.8: Training time of the four DeepPASTA models in our experiments.

<table>
<thead>
<tr>
<th>Model</th>
<th>Data used</th>
<th>Tissue</th>
<th>Running time (hrs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PolyA site prediction</td>
<td></td>
<td></td>
<td>12.667</td>
</tr>
<tr>
<td>Tissue-specific polyA site prediction</td>
<td></td>
<td></td>
<td>9.767</td>
</tr>
<tr>
<td>Tissue-specific relative dominance</td>
<td>Dataset 4</td>
<td>Brain</td>
<td>1.20</td>
</tr>
<tr>
<td></td>
<td>Dataset 4</td>
<td>Kidney</td>
<td>1.15</td>
</tr>
<tr>
<td></td>
<td>Dataset 4</td>
<td>Liver</td>
<td>1.93</td>
</tr>
<tr>
<td></td>
<td>Dataset 4</td>
<td>MAQC_Brain1</td>
<td>2.17</td>
</tr>
<tr>
<td></td>
<td>Dataset 4</td>
<td>MAQC_Brain2</td>
<td>1.75</td>
</tr>
<tr>
<td></td>
<td>Dataset 4</td>
<td>MAQC_UHR1</td>
<td>1.87</td>
</tr>
<tr>
<td></td>
<td>Dataset 4</td>
<td>MAQC_UHR2</td>
<td>2.62</td>
</tr>
<tr>
<td></td>
<td>Dataset 4</td>
<td>Muscle</td>
<td>1.30</td>
</tr>
<tr>
<td></td>
<td>Dataset 4</td>
<td>Testis</td>
<td>1.65</td>
</tr>
<tr>
<td></td>
<td>Dataset 5</td>
<td>Brain</td>
<td>0.20</td>
</tr>
<tr>
<td></td>
<td>Dataset 5</td>
<td>Breast</td>
<td>0.18</td>
</tr>
<tr>
<td></td>
<td>Dataset 5</td>
<td>ES cell</td>
<td>0.17</td>
</tr>
<tr>
<td></td>
<td>Dataset 5</td>
<td>Ovary</td>
<td>0.17</td>
</tr>
<tr>
<td></td>
<td>Dataset 5</td>
<td>SK muscle</td>
<td>0.22</td>
</tr>
<tr>
<td></td>
<td>Dataset 5</td>
<td>Testis</td>
<td>0.18</td>
</tr>
<tr>
<td></td>
<td>Dataset 5</td>
<td>BCells1</td>
<td>0.15</td>
</tr>
<tr>
<td></td>
<td>Dataset 5</td>
<td>BCells2</td>
<td>0.17</td>
</tr>
<tr>
<td>Tissue-specific absolute dominance</td>
<td>Dataset 6</td>
<td>Brain</td>
<td>0.55</td>
</tr>
<tr>
<td></td>
<td>Dataset 6</td>
<td>Kidney</td>
<td>0.93</td>
</tr>
<tr>
<td></td>
<td>Dataset 6</td>
<td>Liver</td>
<td>0.87</td>
</tr>
<tr>
<td></td>
<td>Dataset 6</td>
<td>MAQC_Brain1</td>
<td>0.68</td>
</tr>
<tr>
<td></td>
<td>Dataset 6</td>
<td>MAQC_Brain2</td>
<td>0.52</td>
</tr>
<tr>
<td></td>
<td>Dataset 6</td>
<td>MAQC_UHR1</td>
<td>0.90</td>
</tr>
<tr>
<td></td>
<td>Dataset 6</td>
<td>MAQC_UHR2</td>
<td>0.72</td>
</tr>
<tr>
<td></td>
<td>Dataset 6</td>
<td>Muscle</td>
<td>0.90</td>
</tr>
<tr>
<td></td>
<td>Dataset 6</td>
<td>Testis</td>
<td>0.97</td>
</tr>
</tbody>
</table>
Chapter 4

DeepCEP: deep learning based cell membrane prediction from nucleus

4.1 Introduction

The cell is the basic membrane bounded unit that contains the fundamental molecules of life; all living organisms are composed of cells. The interior of a cell contains different organelles. One of the major organelles of a eukaryotic cell is the nucleus, and each cell usually contains only one nucleus. The cell nucleus is called the information center of the cell. The nucleus carries chromosomes, where most of the genetic information of the cell exists. This genetic information is necessary for cell growth and reproduction. Therefore, the cell is the basic structural, functional, and biological unit in all living organisms.

The ability to observe, extract, and study cells and their sub-cellular compartments is essential to various research areas, e.g., cellular dynamics characterization in normal and
pathological conditions [23], drug discovery [140]. Microscopy is a powerful way to observe these cells and organelles at different timestamps. However, it is challenging to visualize certain structural features or functional characteristics because biological samples are mostly water and are poorly refractile. Fluorescence labeling with dyes or dye-conjugated antibodies provides unprecedented opportunities to reveal these structural features or functional characteristics [25]. In fluorescence labeled microscopic images, different colors or channels represent different information: cell nuclei, cell membrane, stroma/nutrient, cell proliferation signal, cytoplasm, etc. However, fluorescence labeling has some limitations: special reagents must be selected, labeling is time consuming, specificity varies, labeling protocol can kill cells, and live cell protocols can be phototoxic [25]. Because of these limitations, the automatic generation of fluorescent images and the automatic analysis of the microscopy images are necessary for accurate and high throughput cell quantification. Such automated analyses are advantageous to researchers in two ways: the researchers do not have to wait for time consuming cellular image annotations from human experts and they can address different biological problems related to images [133]. One of the most important automatic analyses is cell membrane prediction from 2D microscopy images. The cell membrane protects the internal organelles of a cell from the outside environment. Apart from working as a gatekeeper, it also integrates the cytoplasm of a cell with the outside environment by initiating signaling cascades. The initiation of signaling cascades is highly correlated with the shape of the cell membrane [98]. As a result, researchers can infer the signaling information of the cell by accurately predicting the cell membrane/cell shape.
Over the past decades, several algorithms have been proposed for segmenting cell nuclei or cells in 2D images [32, 82]. Watershed-based models [122, 100] use only one channel to segment cell nuclei. Similar to the watershed-based models, levelset-based models [86, 36] and active contours models [123] segment only cell nuclei. A morphology-based model [34] assumes that the cell nucleus is a blob-like shape and with this assumption, it separates overlapping and touching nuclei. However, a cell contains other elements along with the nucleus, and it is essential to properly segment the whole cell. Segmenting a whole cell is more challenging than segmenting a nucleus because cell shape varies a lot. Moreover, adjacent cells can have a very weak boundary and this weak boundary may leads to inappropriate cell segmentation. In order to segment the whole cell, machine learning-based models are introduced [132, 109]. These models use pixel-based classification and segmentation of cells in bright contrast images. But, these models are evaluated on images with uniform cell appearance. The performances of these models with large variations in cell appearance are not demonstrated in the literature [133]. Moreover, these models cannot segment cells using only the nucleus strain channel. There are other machine learning-based methods [4, 2, 95] that use two channels for the segmentation task. These methods first segment the nuclei using the nucleus strain channel and then use the nuclei as seeds to segment the whole cell based on the cell body/cytoplasm strain channel. If the cell body strain channel is not given, these models cannot segment the cells.

Recently, deep learning [67] technique has been applied in image analysis with superior performance over conventional learning methods on many prediction/classification problems [62, 118]. One of the major advantages of using deep learning technique is that
it can extract and learn powerful image features by itself. Deep learning-based object recognition models are usually trained using images with one or a few objects of interest at the center of the images. Unlike these few objects containing images, microscopy images contain hundreds of cells, and all of these cells are considered objects of interest. At the same time, some of these cells are considered as outliers. As a result, training similar recognition models on microscopy images is challenging. Moreover, training a good deep learning model needs a sufficient amount of labeled data. But, labeling a microscopy image at the single cell level is extremely time consuming. In spite of all the limitations, deep learning based models are developed for segmenting cell nuclei, cytoplasm, and fibers [116, 117, 61, 91, 105] by using multiple channels. Some of these methods also identify cells of different classes [117, 60]. Unlike these models, [133] segments cells from single channel images (cytoplasm strain channel). Therefore, none of the existing deep learning-based methods utilize only the nucleus strain channel to segment cells.

The most common strain channel of any microscopy image is the nucleus strain channel. Therefore, if a model can predict the cell membranes by using only the nucleus strain channel, there is no need for using a fluorescence dye to label the cell structures. At the same time, the model can overcome all the overheads related to fluorescent labeling, e.g., selecting a proper dye for cell structure representation, using the dye properly to generate the structure clearly, etc. Moreover, a cell segmentation model is completely different from a cell membrane prediction model. Unlike the cell membrane prediction model, the cell segmentation model does not separate the membrane from the nucleus and cytoplasm. As
a result, the cell membrane prediction model can easily identify the cell membrane specific biological contributions.

In this chapter, we introduce a new tool, called DeepCEP (i.e., Deep learning based CELl membrane Prediction from nucleus), to predict cell membranes from nuclei. As the nucleus strain channel is the most common channel in any microscopy image, DeepCEP is the first tool to predict cell membranes from the nucleus channel. It employs both a residual neural network [50] (Resnet) and a U-net [88] network-like architectures in its model. Both Resnet and U-net use convolution neural networks (CNNs) in their model architectures to extract abstract features from images. In order to train the model, we use a pre-existing xenograft image data. The image data is collected from colon cancer tissues, and each of the images contains three channels: nucleus, cell structure, and stroma strain channels. In the data, if a nucleus is found in the stroma strain channel, there is no cell structure around that nucleus. In order to take advantage of this information, DeepCEP takes both the nucleus and stroma strain channels as input. The performance of a deep learning model largely depends on the quality of the ground truth data. To achieve better performance, DeepCEP is trained with filtered cell structure channel images as the ground truth data. These filtered cell structure channel images are generated by using another deep learning model.

As DeepCEP predicts cell membranes from nuclei, the best way to evaluate the performance of the model is to match the predicted cell structure with the cell structure strain channel. However, the cell structure strain channel is itself noisy and it is hard to properly evaluate the predicted structure at the image level. Therefore, we simplify the
performance evaluation of DeepCEP by considering only counting the number of cell centers. For both the predicted and annotated images, we calculate the position of each cell center. If the center location of a predicted cell is very close to the center location of an annotated cell, we consider that the cell is being predicted correctly. We then use the cell center count to calculate the sensitivity and precision of DeepCEP. To access the performance of DeepCEP, we have conducted experiments on fluorescence microscopic images and compared DeepCEP with a baseline model for predicting cell membranes. Based on sensitivity and precision, DeepCEP outperforms the baseline model significantly in cell membrane prediction.

The rest of the chapter is organized as follows. The model architecture of DeepCEP is discussed in Section 4.2. The experimental results of the model prediction are discussed in Section 4.3. At the end, conclusion is drawn in Section 4.4.

### 4.2 Methods

In order to predict the cell membranes from nuclei we proposed two deep learning models: filtering and cell structure models. The filtration model uses all three channels to predict the cell structure information. This cell structure information is used to denoise the cell structure strain channel. Later, the denoised cell structure strain channel is used as the ground truth of the second model (cell structure model). Unlike the filtration model, the input to the cell structure model is the nucleus channel. Therefore, the cell structure model is the final DeepCEP model that predicts cell membranes from the nucleus channel. Figure 4.1 illustrates the main steps of DeepCEP. Each of these steps is explained in detail below.
Filtering model

Three channels input patches. These channels are: nucleus, stroma and cell structure.

a) Training phase of the filtering model

Predicted patches containing cell structure.

Human annotated patches containing cell structure (ground truth).

Calculate loss

b) Testing phase of the filtering model

Predicted patches containing cell membranes.

The filtering model predicted patches (ground truth).

Calculate loss

c) Training phase of DeepCEP

Two channels input patches. These channels are: nucleus and stroma.

DeepCEP model

Predicted patches containing cell membranes.

d) Testing phase of DeepCEP

Two channels input patches. These channels are: nucleus and stroma.

DeepCEP model

Predicted patches containing cell membranes.

Figure 4.1: The input and output of the filtering and main models of DeepCEP. a-b) The filtration model of DeepCEP takes a patch containing three channels (nucleus, stroma, and cell structure) as the input and predicts better cell structure containing patches. These patches works as the ground truth data of the main model. c-d) The main model of DeepCEP takes a patch containing two channels (nucleus and stroma) as the input and predicts cell membranes of the input patch. More specifically, this model predicts the cell membranes around the nuclei of the input patch.
4.2.1 Predicting cell structures from three channels (filtration model)

It is natural for a model to predict the cell membrane information from the whole tissue’s fluorescent image. But, a fluorescent image contains hundreds of thousands of cells. In order to get the proper pixel level information, the whole fluorescent image is segmented into small patches. The filtration model takes a patch (128 x 128 x 3) of a fluorescent image as input and predicts a patch (128 x 128) with cell structure information as output. The input patch of this model contains three channels: nucleus, stroma and cell structure strain channels. On the other hand, the output image contains only the cell structure information. More specifically, this model predicts a noise-free cell structure strain channel. Later, this noise free cell structure strain channel is used as the ground truth data of the main model (model of Section 4.2.2). Figure 4.2 shows the overall architecture of the filtration model. The model consists of three sub-models: three identical U-net [88]-like structures. Each of these sub-models consists of a reducing path and an expansive path. The reducing path consists of the repeating application of two convolution layers [66]. Each of these two convolution layers has 3x3 filters. Unlike the traditional U-net’s convolution layers, these convolution layers are padded. When a convolution layer is padded, its output dimension is equal to the input dimension. Following the work [3, 9, 51, 139], the convolution layers of the model are used to extract features from the input patches. Each of the convolution layers uses a rectified linear unit (ReLU) [84] as the activation function to set negative values to zero. After each of two convolution layers, there is a 2x2 max pooling layer [28] to pick the maximum feature value within a window. As a result, a max pooling layer down-samples the number of features. After each max pooling layer, the number of filters is doubled
in the next pair of convolution layers. The expansive path also consists of the repeating application of two convolution layers. These convolution layers use 3x3 filters. Similar to the convolution layers of the reducing path, these convolution layers are padded and they use ReLU as the activation functions. Before each pair of convolution layers, an 2x2 upsampling layer \[141\] is used. An upsampling layer works as the opposite of a max pooling by increasing the number of features. The output from each of these upsampling layers is then concatenated with the correspondingly cropped feature map from the reducing path and is given as input to the following convolution layer of the expansive path. The last layer of a sub-model is a convolution layer with 1x1 filters. The three sub-models are combined using a concatenation layer. The concatenation layer is followed by multiple convolution layers with 3x3 filters. Similar to the previous convolution layers, these convolution layers use ReLUs as the activation functions. The final layer of the model is a convolution layer with a sigmoid activation function and 1x1 filters. In order to prevent data overfitting, dropouts \[107\] are used in some of the layers.
Figure 4.2: Architecture of the filtration model. The model has three identical U-net sub-models. The input to the model is a patch consisting of three channels: nucleus, stroma and unfiltered cell structure strain channels. The output from the model is a patch consisting of filtered cell structure strain channel. Each of these U-net sub-models consist of multiple convolution layers to extract features from the input patch.

Separate training and validation patches are used to train the model, while some test patches are used to evaluate the performance of the trained model. Expert’s annotated
patches are used as the ground truth patches of the training and validation data. The model is trained using the Adam [57] optimizer. It uses 2000 iterations in each training epoch to minimize the average multi-task binary cross entropy loss on the training data. At the end of each training epoch, the validation loss is evaluated to monitor convergence. In order to expedite the learning process, a graphic processing unit (GPU) is used.

4.2.2 Predicting cell structures from the nucleus strain channel (cell structure model)

The filtration model of Section 4.2.1 is used to reduce the noise of the cell structure strain channel. This denoised cell structure strain channel is then used as the ground truth data of the cell structure model. Similar to the filtration model, the cell structure model takes a patch (128 x 128 x 3) of a fluorescent image as input and predicts a patch (128 x 128) with cell structure information as output. From the fluorescent images, it is seen that the nuclei of stroma/nutrient channel do not have membranes around them. In order to use this information in the model’s prediction, the input patches contain two channels: nucleus and stroma strain channels. On the other hand, the output patches contain only the cell structure strain channel.

Figures 4.3 and 4.4 show the overall architecture of the model. Unlike the filtration model, this model consists of four sub-models: three identical Resnet [50]-like structures and one U-net [88]-like structure. Each of the three Resnet structures starts with two convolution layers with 3x3 filters. Each of these convolution layers is followed by a batch-normalization [47] and a parametric ReLU (PReLU) [42] layers. Batch-normalization layers
Figure 4.3: Architectures of the Resnet and U-net sub-models of the cell structure model. These sub-models consist of multiple convolution layers to extract features from a patch of an input image.
Figure 4.4: Architecture of the cell structure model. The cell structure prediction model has four sub-models: a U-net sub-model and three identical Resnet sub-models. The input to the model is a patch consisting of nucleus and stroma strain channels. The output from the model is a patch consisting of cell membranes around the nuclei.
are usually used to increase the speed, performance and stability of a model. After the first two convolution layers, three identical sub-units are applied. Each of these sub-units starts with a 2x2 upsampling layer [89]. The upsampling layer is followed by a convolution layer with a ReLU activation function and 1x1 filters. The output from the convolution layer passes through both a 2x2 maxpooling layer and a 2x2 average pooling layer at the same time. The outputs from the pooling layers are combined using a concatenation layer. The concatenation layer is followed by three convolution layers. Similar to the previous convolution layer of the sub-unit, these convolution layers use 1x1 filters and ReLUs as the activation functions. A copy of the sub-unit input is shortcutted with the output of the sub-unit using a concatenation layer. Three consecutive sub-units are followed by two convolution layers of 3x3 filters. In fact, these two convolution layers denote the end of a Resnet sub-model architecture. On the other hand, the U-net sub-model consists of a reducing path and an expansive path. The reducing path consists of the repeating application of a convolution, a batch-normalization and a ReLU activation function layers. After each ReLU activation layer, there is a convolution layer. The convolution layer is followed by a max pooling layer and an average pooling layer. Similar to the Resnet sub-model, the outputs from the pooling layers are combined using a concatenation layer. Starting from 8x8, the size of the pooling windows are doubled at each step. This double increment in the window size reduces the dimension of the features rapidly. On the other hand, the expansive path consists of the repeating application of two convolution layers. The convolution layers are followed by an upsampling layer. The upsampling layer works as the opposite of a pooling layer in the reducing path. Unlike the pooling layers of the reducing path,
the window size of the upsampling layers are decreased by half at each step. The output from each of these upsampling layers is then concatenated with the correspondingly cropped feature map from the reducing path. The last three layers of the U-net sub-model are the convolution layers: Among these three convolution layers the first two use 3x3 filters and the last one uses 1x1 filters. The four sub-models are then combined using a concatenation layer. The concatenation layer is followed by multiple convolution layers with 3x3 filters. All the convolution layers of the model are padded. That means the feature dimension does not change before and after a convolution layer. In order to prevent overfitting, dropouts are used in some of the layers. Moreover, L2 norm regularizations [44] are added at the output layers of sub-models and full model to prevent overfitting.

Separate training and validation patches are used to train the model, while some test patches are used to evaluate the performance of the trained model. Denoised patches are used as the ground truth patches of the training and validation data. The model is trained using the Adam RMSprop with Nesterov momentum [35] optimizer. The model uses a variant of mean aboslution error as the cost function. This modified cost function penalizes more if the high and low intensity pixels are not predicted correctly.

\[
\begin{align*}
    a &= \frac{1}{2} \left[ \text{mean}(y_{\text{true}}) + \frac{\text{max}(y_{\text{true}}) + \text{min}(y_{\text{true}})}{2} \right] \\
    \text{cost} &= \text{mean} \left( |y_{\text{pred}} - y_{\text{true}}| \times 10^{|y_{\text{true}} - a|} \right)
\end{align*}
\]  

(4.1)

In equation 4.1, \( y_{\text{true}} \) and \( y_{\text{pred}} \) denote the ground truth and predicted values, respectively. The model uses mini-batch size of 256 patches to minimize the cost function. At the end of each training epoch, the validation loss is evaluated to monitor convergence. In order to expedite the learning process, a graphic processing unit (GPU) is used.
4.3 Experimental results

In this section, we compare the performance of DeepCEP with a baseline tool for predicting cell membranes from nuclei using microscopic images. In order to train and test both DeepCEP and the baseline model, we use a selected pre-existing xenograft image data. Initially, human colon cancer tissues are implanted inside mouses to study the growth of tumors. From the mouse tumors, brightfield microscopy images are created from histological sections. Later, these microscopic images are strained using biomarkers to construct the Immunofluorescence (IF) microscopy images. We use these IF images to train and test DeepCEP and the baseline model. Each of these images has three channels: nucleus (blue), cell structure (green), and stroma (red) strain channels. In the data, if a nucleus is found in the stroma strain channel, there is no cell structure around that nucleus. In order to take advantage of this information, DeepCEP and the baseline model utilize both the nucleus and stroma strain channels to predict the cell membranes. Moreover, the cell structure channel of the image data is very noisy. In order to overcome the noise, we use a deep learning based filter model (4.2.1). This filter model takes a patch of three channels as input and outputs a noise free cell structure channel. The model is trained using twenty patches of size 128 x 128. We only use twenty patches as the training data because the model needs expert-annotated cell structure images as the ground truth data. Moreover, it is time consuming to annotate a 128 x 128 patch at the cell level. Later, the outputs from the filter model are used to denoise the cell structure channel. This denoised data significantly improves the performance of both DeepCEP and the baseline model.
Performance on predicting cell membranes from nuclei

In this experimental study, we compare DeepCEP with a baseline model for predicting cell membranes from nuclei. In order to train the model of DeepCEP, we collect three IF images and construct three groups. We construct three groups because we want to train the model using a threefold cross-validation technique [59]. The cross-validation technique makes the model more robust to unseen data. In each group, two images are used as the training data and one is used as the validation data. Each of these images contains approximately two hundred thousands of cells, and it is not possible to precisely predict the cell membranes of these cells by directly feeding the whole image to the model. As a result, each of these images is segmented into non-overlapping patches of size (128 x 128), and we obtain around 12500 patches from one image. DeepCEP uses a patch of nucleus and stroma strain channels as an input and outputs a patch containing the cell structure strain channel. In order to train the model, we use filtered cell structure strain channel as the ground truth data. Figure 4.5 shows some of the input and ground truth patches of the training data. At the same time, the figure shows the difference between filtered and unfiltered data. By training DeepCEP on three groups, we get three trained models. We use these three trained models on unseen test data and take the mean of these models as the output of DeepCEP. If the output from trained model \( i \) is \( O_i \), then the output of DeepCEP can be represented by equation 4.2.

\[
O_{DeepCEP} = \sum_{i=1}^{3} O_i
\]  

(4.2)
Figure 4.5: Some of the patches of the training data. The patches from the first row are the input patches of DeepCEP. These patches contain nucleus and stroma strain channels. The second row shows the unfiltered cell structure strain channel patches corresponding to the patches of first row. The third row shows the filtered cell structure strain channel patches corresponding to the patches of second row. The patches from the third row are the ground truth patches of DeepCEP. Note that the filtered patches are generated using the filtration model of section 4.2.1.
Similar to DeepCEP, the baseline model is a deep learning-based model. But, it contains one sub-model instead of four: a U-net-like structure. This U-net structure is similar to the U-net structure of DeepCEP’s sub-model. Moreover, we use the cost function (Equation 4.1) of DeepCEP as the cost function of the baseline model. Similar to DeepCEP, the baseline model is trained using a threefold cross-validation technique to make it robust to unseen test data. For consistency, we use the same training data to train the baseline model.

We evaluate the performance of the tools with respect to two datasets. Dataset 1 contains patches for which we have the expert’s annotated cell membranes. On the other hand, Dataset 2 does not contain any expert’s annotation. As a result, we use the prediction from the filter model to evaluate the performance of the tools in Dataset 2. Note that none of the patches of these two datasets exists in the training data.

Figure 4.6 shows the cell membranes predictions from DeepCEP and the baseline model on some of the input patches of Dataset 1. The figure also shows the expert’s annotation of those cell membranes. As DeepCEP predicts cell membranes from nuclei, the ideal way to evaluate the performance of DeepCEP is to match the predicted membranes with the expert’s annotations. But, there is no better way to quantify the performance of these predictions at the cell level. Therefore, we simplify the performance evaluation by considering counting the number of cell centers. More specifically, we calculate the position of each cell center from the prediction and the annotation. If the center location of a predicted cell is very close to the center location of an annotated cell, we consider that cell to be predicted correctly. Here, we consider less than or equal to 10 as the threshold for...
Figure 4.6: Performance comparison of DeepCEP and the baseline model for predicting cell membranes from nuclei. a) Some of the input patches from Dataset 1. Each of these patches consists of two channels: nucleus and stroma strain channels. b) Expert’s annotation of cell membranes for these input patches. c) When the input patches are given to DeepCEP, it predicts the cell membranes. d) Similar to DeepCEP, when the input patches are given to the baseline model, it predicts the cell membranes.
closest distance. We follow a sequence of steps to find the center of a cell from the predicted patches. In order to find the center of a cell, first, we need to specify the boundary of the cell. We use the filtration model of section 4.2.1 to get the boundary of the cell. More specifically, we give the predicted cell structure strain channel along with the nucleus and stroma strain channels as an input to the filtration model. The model then predicts the boundary of the cell. While predicting the boundary of the cells, we observe that some of the cell boundaries are not closed. In order to close those boundaries, we use dilation and erosion on those boundaries. From the boundaries of the cells, we calculate the center of gravity of the cells. In this way, we obtain the centers from the predicted cell membranes. On the other hand, we only calculate the center of gravity of the cells from the expert’s annotated patches. These centers from the predicted and annotated patches give us the cell counts. We then use the cell counts to calculate the sensitivity and precision of the cell membrane prediction of DeepCEP. We also follow the same procedure to calculate the sensitivity and precision of the baseline model. Tables 4.1 and 4.2 show the performance comparison of DeepCEP and the baseline model in predicting cells from nuclei on Dataset 1 and 2, respectively. From the tables, it is seen that DeepCEP clearly outperforms the baseline model in cell center prediction. At the same time, we can state that, DeepCEP outperforms the baseline model in predicting cell membranes from the nuclei. However, the performance of both models drops from Dataset 1 to Dataset 2. Note that Dataset 2 does not have annotations from an expert, and the annotations of the dataset is created using the filtration model. As a result, the declined performance in Dataset 2 is due to the prediction of the filtration model. Although both DeepCEP and the baseline model use
deep learning algorithms, DeepCEP performs significantly better than the baseline model on all datasets. The use of Resnet sub-models and complex model architecture may have helped improve the performance of DeepCEP.

**Table 4.1:** Performance comparison between DeepCEP and the baseline model in predicting cell from nuclei on Dataset 1 in term of sensitivity and precision. Dataset 1 contains 20 annotated patches, and there are 639 cells in these patches.

<table>
<thead>
<tr>
<th>Tool name</th>
<th>Number of predicted cells</th>
<th>Correctly predicted cells</th>
<th>Sensitivity (%)</th>
<th>Precision (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeepCEP</td>
<td>591</td>
<td>515</td>
<td>80.59</td>
<td>87.14</td>
</tr>
<tr>
<td>The baseline model</td>
<td>463</td>
<td>403</td>
<td>63.07</td>
<td>87.04</td>
</tr>
</tbody>
</table>

**Table 4.2:** Performance comparison between DeepCEP and the baseline model in predicting cell from nuclei on Dataset 2 in term of sensitivity and precision. Dataset 2 contains 12000 patches, and there are 137058 cells in these patches.

<table>
<thead>
<tr>
<th>Tool name</th>
<th>Number of predicted cells</th>
<th>Correctly predicted cells</th>
<th>Sensitivity (%)</th>
<th>Precision (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeepCEP</td>
<td>155000</td>
<td>102636</td>
<td>74.89</td>
<td>66.22</td>
</tr>
<tr>
<td>The baseline model</td>
<td>133316</td>
<td>87687</td>
<td>63.98</td>
<td>65.77</td>
</tr>
</tbody>
</table>

### 4.4 Discussion and future research

In this chapter, we introduced DeepCEP, a deep learning based tool for predicting cell membranes from nuclei using fluorescent microscopic image data. Our experiments on both annotated and unannotated data show that DeepCEP performs better than a baseline model for cell membrane prediction analysis. As a result, we expect that DeepCEP will serve as a useful tool in microscopic medical image analysis.

One of the possible applications of DeepCEP can be identifying heterogeneity with in a tissue. In the heterogenous locations, the cell membranes around the nuclei take
unusual shape. This shape change in cell membrane may trigger signals that are related to tumor/cancer. We can train the model of DeepCEP with only homogenous locations of a tissue and use the trained model on the whole tissue. If the model fails to predict the cell membranes of a particular location, we consider that location as a possible heterogenous location within the tissue. The identified heterogenous location can be studied further for tumor/cancer. We are unable to demonstrate such application of DeepCEP due to lack of ground truth data. Therefore, demonstrating some applications of DeepCEP can be an interesting future work in the field of tumor/cancer study.
Chapter 5

Conclusions

High throughput methods (NGS, microscopic imaging) are increasingly applied in various areas of biological and chemical research. The data generated by these technologies create new opportunities in the field of bioinformatics. There is a need for new and improved methods to efficiently deal with the challenges of this increasing amount of data.

In this dissertation, we addressed three problems that are related to polyadenylation process and microscopic imaging. First, we presented a tool for detecting alternative polyA (APA) sites from RNA-Seq expression data. We then extended the tool for detecting differentially expressed APA sites between two biological samples. Finally, we used the differentially expressed APA sites for identifying shortening/lengthening event genes between those two biological samples. Second, we proposed a novel deep learning-based classification method for predicting polyA sites from sequence and RNA secondary structure data. We then extended the method for predicting tissue-specific polyA sites. When a couple of sequences (and corresponding RNA secondary structures) containing polyA sites of some
gene in a particular tissue are given as the input, the method can also predict the relatively dominant polyA site. Moreover, the method can predict the absolutely dominant polyA sites of a gene from the sequence and RNA secondary structure data. Finally, we presented a deep leaning based regression method for predicting cell membranes from nuclei using fluorescent microscopic image data.
Bibliography


