Supplemental Materials for DeeReCT-PolyA: a robust and generic deep learning method for PAS identification

Zhihao Xia[1], Yu Li[2], Bin Zhang[3], Zhongxiao Li[2], Yuhui Hu[3], Wei Chen[3,*], and Xin Gao[2,*]

[1]*Department of Computer Science and Engineering (CSE), Washington University in St. Louis, St. Louis, MO 63130, USA*

[2]*King Abdullah University of Science and Technology (KAUST), Computational Bioscience Research Center (CBRC), Computer, Electrical and Mathematical Sciences and Engineering (CEMSE) Division, Thuwal, 23955-6900, Saudi Arabia*

[3]*Department of Biology, Southern University of Science and Technology (SUSTC), Shenzhen, 518055, China*

[*]All correspondence should be addressed to Wei Chen (chenw@sustc.edu.cn) and Xin Gao (xin.gao@kaust.edu.sa).

Table S1: Search range and sampling method for hyper-parameters

Parameter	Search Range	Sampling Method
Learning Rate	$[5 \times 10^{-4}, 5 \times 10^{-2}]$	log-uniform
Fine-tune Learning Rate	$[5 \times 10^{-5}, 5 \times 10^{-2}]$	log-uniform
Momentum	$[0.95, 0.99]$	sqrt-uniform
Initial Weight (conv)	$[1 \times 10^{-2}, 10]$	log-uniform
Initial Weight (FC)	$[1 \times 10^{-2}, 10]$	log-uniform
Weight Decay (conv)	$[1 \times 10^{-5}, 1 \times 10^{-3}]$	log-uniform
Weight Decay (FC)	$[1 \times 10^{-5}, 1 \times 10^{-3}]$	log-uniform
Keep Probability	$\{0.5, 0, 75, 1.0\}$	random-choice

S1: Deep Learning Architecture

In this work, we proposed a relatively shallow network including one convolutional layer and two fully-connected layers and several other layers. However, we indeed searched various kinds of deep learning architectures including increasing the network depth by adding more convoulutional layers while having smaller kernel size, having residual connections among layers [1], very deep networks with dense connections [2]. But in practice, we found a simple network with fewer layers and less parameters yields better results. Most importantly, a simpler network would have better interpretability which is crucial for us to understand the mechanism of polyadenylation. Thus, our proposed network begins with a convolutional layer with 16 kernels and each of them is of size 10. The output is then processed by a ReLU layer and downsampled with max-pooling of kernel size 10. The fully-connected network contains one hidden layer with 64 hidden neurons. During training, the batch size is set to 64.

S2: Hyper-parameter Search

Following [3], instead of manually searching the best hyper-parameters of our proposed model, we generated these parameters by random sampling during training and choose the best set of hyper-parameters based on its performance on the validation dataset. To sample a real number in the interval $[a, b]$ with sqrt-uniform or log-uniform, we first uniformly sample a real number $x \in [0, 1]$, then return $(b-a)\sqrt{x} + a$ or $10^{(\log_{10} b - \log_{10} a)x + \log_{10} a}$. The search range and sampling method of each hyper-parameter can is presented in Table S1. The parameter *Init Weight* is the scale of the normal distribution which is used to initialize the weight of a convolutional layer or a fully-connected (FC) layer. *Keep Probability* is the probability to **not** drop a value in the drop-out layer. Note that the final set of hyper-parameters to use during testing time is decided by the validation dataset, so each experiment would have different final actual set of hyper-parameters. The actual hyper-parameter sets we use in all experiments can be found at https://github.com/likesum/DeeReCT-PolyA.

S3: Filter-group Normalization

To stablize and accelerate the training procedure of a deep network, batch normalization (BN) is widely used, which normalizes the outputs of convolutional filters with the mean and variance computed in a training mini-batch. However, a particular drawback of BN is that it requires normalization along the batch dimension, while we do not necessarily have a batch of data, for example, during test time.

Motivated by a very recent work in computer vision [4] and the fact that many *cis*-elements are correlated in poly(A) regulation, in this work we propose a filter-group normalization (GN) method to normalize the outputs of convolutional layers without referring to the batch statistics. Instead of normalizing each channel of the output seperately with its mean and variance in a batch (batch-wise), we first divide filters into several groups. Then for a group of filters, we compute the statistics within the output feature channels that correspond to all filters in the group and perform a group-wise normalization. With a joint group normalization, the model can capture a set of hidden patterns which have similar distributions. Such patterns could be, for example, different U-rich or UG-rich elements. Note that the normalizing computation is independent along the batch axis which leads to a consistent training and testing normalizing step because it performs normalization for each sequence independently. The number of groups is set as a hyper-parameter that is automatically searched on the validation data. To demonstrate the effectiveness of the proposed filter-group normalization, we compare it with two other normalization methods, i.e., instance normalization (IN) and batch normalization, as well as the model without normalization. As shown in Table S2, the model with the filter-group normalization outperforms the ones with other normalization layers while not requiring any batch statistics. Although IN also performs independent normalization for each sequence, the sacrifice of the accuracy is high. Note that the model without any normalization layer usually converges much slower, and the training process is very unstable.

More importantly, GN is able to normalize the outputs of conv layers without referring to the batch statistics while BN fails when there is no batch of testing sequences, especially when transferred to a new dataset. We did an experiment to demonstrate this point. We test two pre-trained models of Omni-human, one of which is trained with BN and the other with GN, on BL and SP mouse data. When testing, only one sequence is fed to the model every time. No fine-tuning is performed. Results (Table S3) show that the model pre-trained with GN performs significantly better in this case.

S4: Transfer Learning

We did a comprehensive study to validate the idea of transfer learning by presenting many combinations of transfer learning, including intra-species (mouse to mouse or human to human) and cross-species (mouse to human or human to mouse) transfer learning, marrying similar sized datasets or a large sized dataset with a small one.

In the first scenario we incoporate a similar sized dataset to the new dataset. For example, when we transfer the model from BL mouse data to Omni human data, we first pre-train our DeeReCT-PolyA model with BL mouse sequences and then fine-tune the pre-trained model with Omni human data which is our target. Models before fine-tuning and after fine-tuning are both evaluated on Omni dataset. Note

Table S2: Comparison of different normalization methods on the Omni human dataset.

Variants	Size	Error Rate (%)			
		No-Norm	Inst-Norm	Batch-Norm	Group-Norm
AATAAA	24310	22.52	23.33	22.44	**21.99**
ATTAAA	7098	23.29	24.03	23.19	**23.01**
AAAAAG	1640	**26.62**	27.48	28.08	27.76
AAGAAA	1306	27.60	29.60	28.80	**26.80**
TATAAA	682	24.82	24.21	24.09	**23.60**
AATACA	634	22.84	22.83	22.80	**22.00**
AGTAAA	528	20.75	21.91	20.98	**20.21**
ACTAAA	368	26.43	26.09	**22.90**	25.79
GATAAA	342	22.35	20.27	**19.71**	22.15
CATAAA	314	25.83	25.38	**23.90**	25.54
AATATA	250	19.40	19.54	18.30	**17.82**
AATAGA	100	**20.00**	27.00	25.00	**20.00**
Average	–	23.08	23.85	23.04	**22.64**

Note: No-Norm indicates the model without any normalization layer.

Table S3: Comparison of BN and GN for transfer learning from Omni-human

	Average Error Rate (%)	
Normalization	BL	SP
BN	35.21	34.63
GN	**31.18**	**31.59**

that the model before fine-tuning didn't see any Omni sequences during training. As a baseline, one model is trained with only Omni data without any pre-training. The aforementioned experiment is repeated for all six combinations of Omni human data, BL and SP mouse data, with one as pre-training dataset and one as target dataset.

In the second scenario, we transfer a model pre-trained on a large dataset (Omni, BL or SP) to a smaller dataset (one fold of Dragon human data) to address the problem of insufficient data. Normally in 5-fold cross-validation, we use three of the folds for training, one of the folds for validation and one for testing. However, here we conserve only one of the folds for training, and use one for validation and the remaining three for testing, which results in a training fold cosisting of only $14740/5 = 2948$ sequences. In articular, the model is first pre-trained on one of the Omni, BL and SP dataset. The only one training fold of Dragon human is then used to fine-tune the pre-trained model to adapt it to Dragon human PAS recognition.

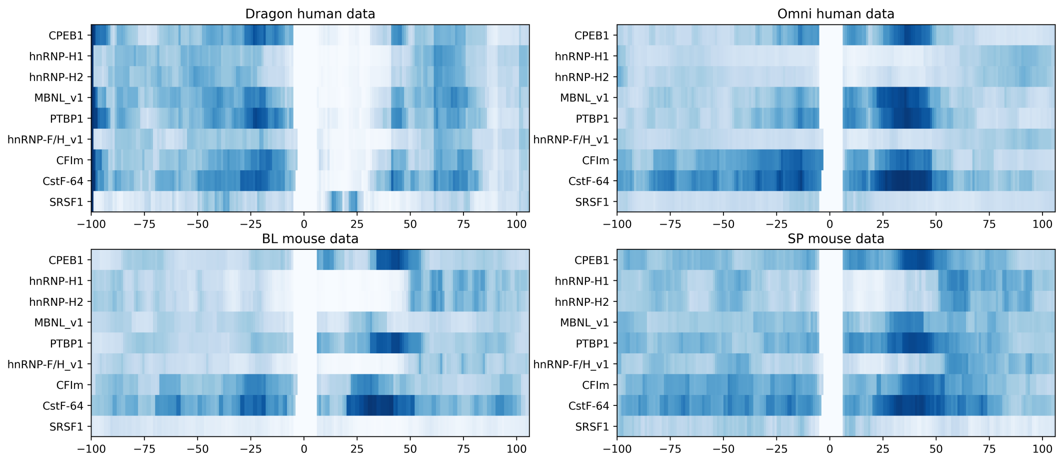


Figure S1: Visualization of the importance of different RBPs at different positions for models trained with four datasets. The colors denote the contribution of the RBP at that position to determining a true PAS motif. The darker blue, the more contribution the RBP at that position has to determining a true PAS motif. The more white, the less contribution. The x-axis shows the positions of the dimer in the sequence, where position 0 is the first base of the PAS motif. The y-axis lists all possible dimers. The y-axis lists RBPs we investigate.

S5: Methods to Visualize Trained Models

Importance of Dimers and RBPs

To visualize the importance of dimers, the first step is to construct a special sequence which only contains one dimer (out of 16 possible dimers) at position k (range from 0 to 205) and a PAS motif variant (one of the 12 human PAS motif variants) at the center of the sequence. The rest bits in this special sequence is set to 0. Such sequence is fed to the trained network and the network then returns a value indicates how likely this sequence is positive (contains true PAS motif). The returned value actually shows the importance of this dimer at position k. We put all dimers (y-axis) and their importance at all postions (x-axis) together in a heatmap to visualize their importance. The darker blue in the heatmap indicates the more contribution the dimer at that position has to determining a true PAS motif. The less blue, the less informative. Note that because the centered 6 bits in all sequences, no matter true or false, are always a PAS motif, so they have no information at all. Thus, in each line of the heatmap, the centered region is consistently white.

We have also investigated the importance of some known RBPs for models trained on human and mouse datasets (Figure S1). Our result indicates that RBPs like *CPEB1*, *PTBP1* and *CstF-64* are of great importance to determining a true PAS when it occurs in the downstream of a PAS motif. *MBNL_v1* and *CFIm* are informative, too.

Visualization of Convolutional Filters

Given that each convolutional filter in our model scans all subsequences of length 10 in the raw DNA sequence and output a value (known as activation) for each 10-mer, we can use these values, i.e., the output of the first layer, to visualize the learned convolutional filters. Specifically, for each filter, we find the 10-mer subsequence in the input sequence that the filter gives the largest activation to and regard the found subsequence as the *cis*-element captured by this filter. We repeat the aforementioned process for all sequences in the validation dataset and each sequence would have one subsequence "best" captured by this filter. All such subsequences (of length 10) are aligned to generate a position frequency matrix. The PFM is subsequently transformed into a sequence logo which indicates which kind of subsequence the filter looks for. All filters of the convolutional layer in our network are visualized in this way to tell us the important features for determining a PAS is true or pseudo. We can also compare the features of models trained on data of different species to understand the across-species connection of polyadenylation regulation.

Importance of Positions

To visualize the importance of postions for PAS recognition, we propose to see where the convolutional filters of the trained model are mostly looking at. Specifically, as mentioned above, each convolutional filter in our model scans through all 10-mer subsequences in the input sequence and assign a score to each of the subsequence at every position k. So we can get a score for each position assigned by each filter if a sequence is input to the model.

In fact, we input all positive sequences (containing a true PAS motif) and average the scores over all filters and all input sequences which then yield a single score vector, of which each element is a score for position k. The aforementioned process is repeated to get a score vector for negative sequences by feeding all negative sequences in the validation dataset to the model. These two vectors are plot in Figure S2 with x-axis as the position and y-axis as the score. Note that the regions where we observe the largest difference between the positive and the negative score vector are the most important regions to distinguish positive sequences from negative ones. As has been proved a lot by other studies in polyadenylation, Figure S2 shows that the 10-30 nt downstream of the PAS motif carries crucial information for PAS recognition.

Similarity of Leave-one-motif-out Models

In the leave-one-motif-out experiment, we train the model with data of all PAS motif variants except the target one and challenge it to predict on this unseen motif variant. Since there are twelve motif variants for human PAS, we have trained twelve different models. Here, by adopting the method we proposed in Section 2.8 and above, we transform the convolutional filters of these models into position frequency matrices (PFMs) and them measure the similarity between every two models. The similarity scores are plot as a heatmap in Figure S3. For the purpose of comparison, the similarity score of two randomly initialized CNN models is 0.00036 and the similarity of BL mouse model and SP mouse model is 0.0014 (Table 8).

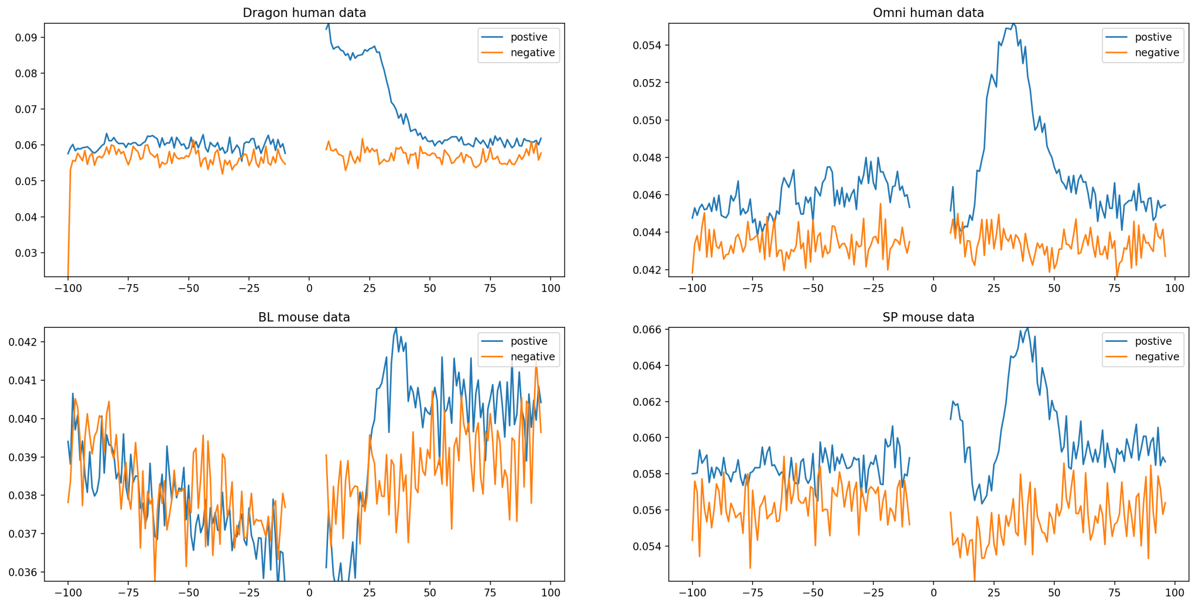


Figure S2: Visualization of the importance of different positions for models trained with four datasets. Scores for each position k in positive and negative sequences are plotted with x-axis as the position and y-axis as the score. Note that the difference between two lines shows how that position is important to distinguish true PAS from pseudo ones.

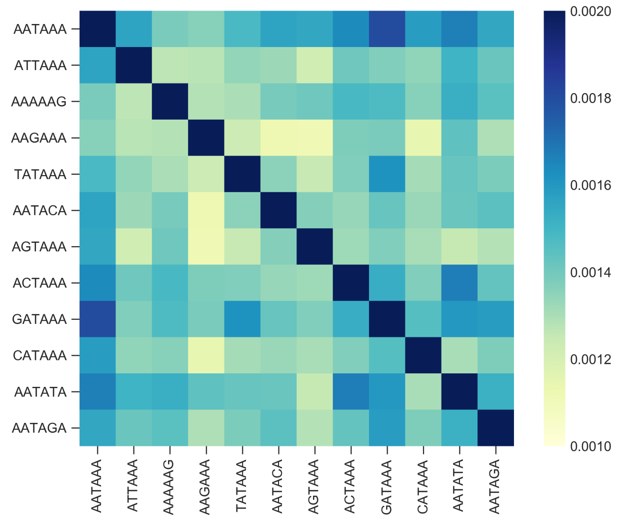


Figure S3: Similarity scores for every two of 12 leave-one-motif-out models. For example, the bottom-left corner indicates the similarity of a model trained with data excluding *AATAGA* and a model trained with data excluding *AATAAA*.

S6: Differences between different PAS motifs

While our model is able to deal with all PAS variants, we still tried to investigate the differences between different poly(A) signals. Given one model trained for all different poly(A) signals, we first visualized what does this model "look at" when processing different poly(A) signals, i.e., the *cis*-elements captured by the model in sequences that contain different poly(A) signals. Specifically, we input a set of sequences that only contain one type of poly(A) signals and derive the position frequency matrix in the same way as we visualized convolutional filters. Then we repeat this step for all poly(A) signals and finally compute the similarity of the *cis*-elements captured for every pair of poly(A) motifs. Results are shown as a heatmap (Figure S4), which shows that the *cis*-elements identified for *AATAAA*, *ATTAAA*, *TATAAA* and *AGTAAA* are very similar to each other while the signal *AATAGA* has very different *cis*-elements from other poly(A) motifs.

Next we try to answer the question if the model looks at different positions for different signals. We use the same method we used for visualizing the importance of positions (Figure S2). We plot the score assigned by the convolutional filters to different positions in positive sequences and sequences containing pseudo PAS motifs (Figure S5). We can see that while for all poly(A) signals the centered downstream region is very informative, the 50-80 nt downstream region is also informative for some poly(A) signals including *AAGAAA* and *GATAAA*. And different from other poly(A) signals, all positions in sequences containing *AATAAA* and *ATTAAA* are quite important for the identification of poly(A) sites.

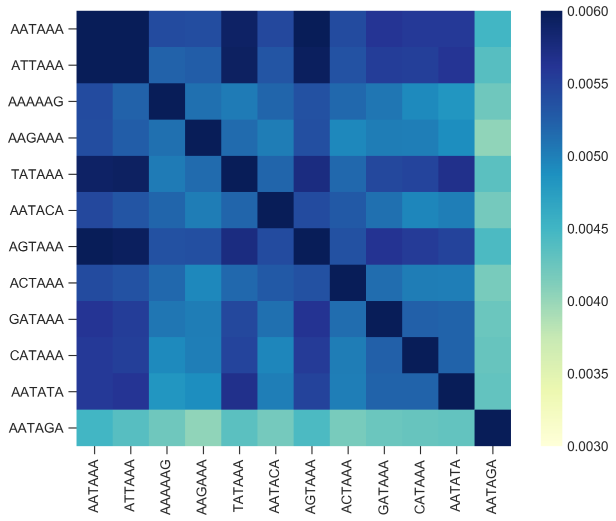


Figure S4: The similarity score of what DeeReCT-PolyA "look at" when processing sequences containing different poly(A) signals. For example, the bottom-left corner indicates the similarity of *cis*-elements captured by DeeReCT-PolyA in sequences containing *AATAGA* and sequences containing *AATAAA*.

S7: Results of DeeReCT-PolyA on SP and BL mouse data

For the purpose of comparison, we have trained a linear regression model as the baseline for BL and SP mouse data. Our model outperforms the baseline on all motif variants for both BL and SP data (Table S4, S5). To see if expert-designed features can help improve the performance, we extracted 1529 hand-crafted features [5] of the sequence, including *cis*-elements, RBP motifs, etc. and added them to our model. Specifically, these hand-crafted features are concatenated to the features extracted by our DeeReCT-PolyA model, i.e. the output of our pooling layer. After concatenation, the whole feature vector is input to the fully-connected layers. After training, the model gives a slightly lower error rate on BL and SP mouse data than DeeReCT-PolyA without hand-crafted features. We think the reason is that the features learned by our model already contains most of the information encoded in these known features.

S8: Evaluate the robustness of DeeReCT-PolyA against noise

To test if DeeReCT-PolyA is robust to noise, we did an experiment on the Dragon human dataset, where we randomly perturbed different numbers of nucleotides in the sequence and test the accuracy of DeeReCT-PolyA. Although there is no guarantee on the label of the sequence after perturbation, it is reasonable to assume that if the perturbation is small, the label will not change. More importantly, we investigated the case when mismatches are only introduced to certain regions (AUE, CUE, CDE, ADE)

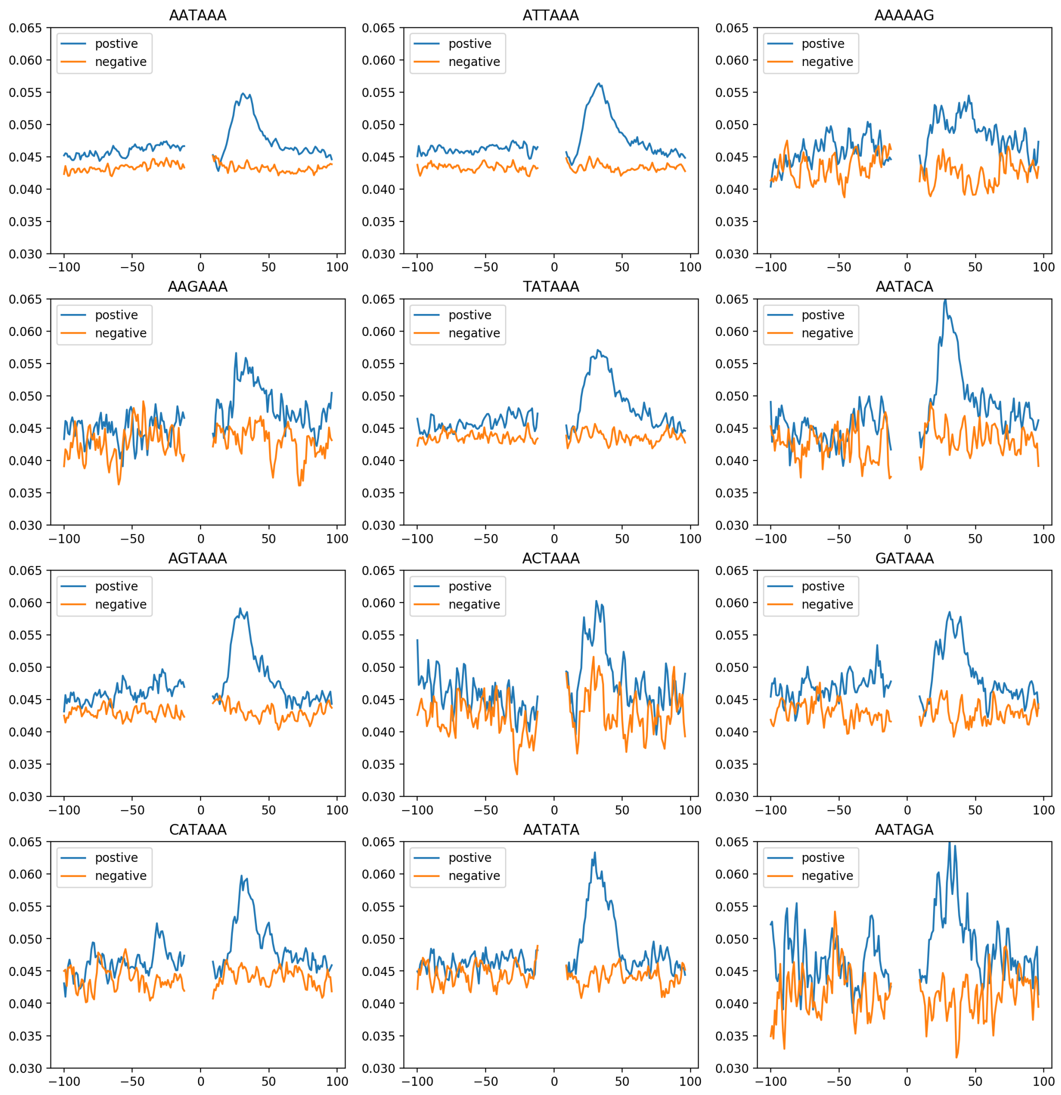


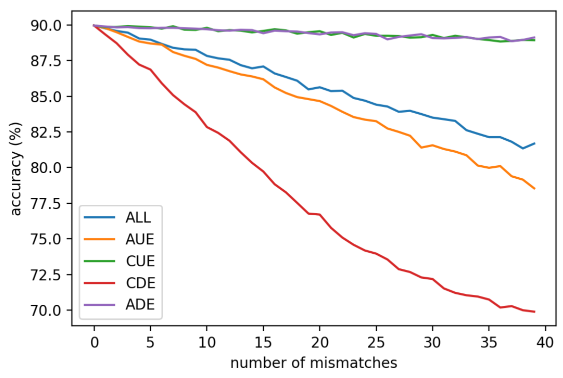
Figure S5: Visualization of the importance of different positions for DeeReCT-PolyA when processing sequences containing different poly(A) signals. Scores for each position k in positive and negative sequences are plotted with x-axis as the position and y-axis as the score. Note that the difference between two lines shows how that position is important to distinguish true PAS from pseudo ones.

Table S4: Evaluation of DeeReCT-PolyA on SP mouse poly(A) data.

Variants	Size	Error Rate (%)		
		Baseline	DeeReCT-PolyA	DeeReCT-PolyA + Features
AATAAA	17708	29.08	26.50	25.49
ATTAAA	7550	27.55	25.30	23.95
TTTAAA	2336	26.93	19.95	24.53
TATAAA	2178	25.76	22.91	22.77
AGTAAA	2224	24.42	22.88	20.77
CATAAA	1432	25.49	20.53	21.37
AATATA	1334	28.41	23.55	22.87
AATACA	1210	25.29	21.40	22.23
GATAAA	1032	20.45	17.84	18.02
AAGAAA	1022	20.84	15.07	15.85
AATGAA	982	24.04	18.84	17.01
ACTAAA	728	24.59	19.37	20.87
AATAGA	494	23.12	18.64	19.45
Average	–	27.26	24.11	23.60

Table S5: Evaluation of DeeReCT-PolyA on BL mouse poly(A) data.

Variants	Size	Error Rate (%)		
		Baseline	DeeReCT-PolyA	DeeReCT-PolyA + Features
AATAAA	20250	28.56	25.48	24.88
ATTAAA	9056	27.25	24.89	24.03
TTTAAA	2688	26.93	18.19	23.21
TATAAA	2518	26.17	22.44	22.92
AGTAAA	2376	23.91	21.63	20.92
CATAAA	1760	24.09	19.77	22.05
AATATA	1528	29.52	23.23	23.82
AATACA	1326	24.96	22.55	21.04
GATAAA	1176	21.93	18.54	20.40
AAGAAA	1126	22.38	15.81	18.48
AATGAA	1108	22.21	18.86	17.79
ACTAAA	776	22.93	20.24	22.15
AATAGA	536	24.45	21.24	19.60
Average	–	26.98	23.49	23.48



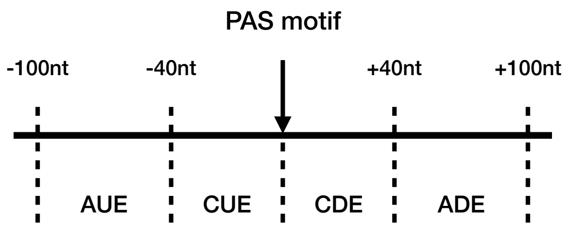


Figure S6: **Left**: The accuracy of the model on the Dragon human data when random mismatches are introduced to different regions of the sequence. **Right**: The sub-regions defined with respect to the centered PAS motif.

of the sequence and the purpose is to see which region is more important. Specifically, AUE, CUE, CDE and ADE are regions defined with respect to the centered PAS motif (Figure S6).

Results (Figure S6) show that DeeReCT-PolyA is robust to mismatches, as even if there are 40 mismatches in the whole sequence, DeeReCT-PolyA still has an around 82% accuracy on Dragon human data. Most importantly, One can observe that CDE, which is 0-40 nt downstream the PAS motif, is very informative for the identification of a poly(A) site. This observation coincides with our visualization of position importance (Figure S2) where the result also shows that this region is of great importance. As expected, DeeReCT-PolyA has a very stable performance with respect to mismathces in CUE and ADE.

References

[1] He, Kaiming, et al. Deep residual learning for image recognition. In *CVPR*, 2016.

[2] Huang, Gao, et al. Densely connected convolutional networks. In *CVPR*, 2017.

[3] Alipanahi, Babak, et al. Predicting the sequence specificities of DNA-and RNA-binding proteins by deep learning.In *Nature biotechnology 33.8: 831*, 2015

[4] Wu, Yuxin, et al. Group normalization. In *ECCV*, 2018.

[5] Leung, Michael Ka Kit, et al. Inference Of The Human Polyadenylation Code. In *bioRxiv*, 2017.