Deep learning models for RNA-protein binding

by

Akira Ishida

A thesis submitted in conformity with the requirements
for the degree of Master of Applied Science
Graduate Department of Edward S. Rogers Sr. Department of Electrical and Computer Engineering
University of Toronto

Abstract

Deep learning models for RNA-protein binding

Akira Ishida
Master of Applied Science
Graduate Department of Edward S. Rogers Sr. Department of Electrical and Computer Engineering
University of Toronto
2017

RNA binding proteins(RBPs) are crucial bio-molecules that fine tune gene expression in cells. Each RBP prefers to bind to a specific RNA sub-sequence, like a key fitting a lock. Understanding the specific binding preferences of RBPs is an important step to understanding the various steps of gene expression in cells and in solving several genetic disorders. There are thousands of RBPs in humans and only a small fraction of them are well understood. In this work, we develop deep neural network models that allow us to learn binding preferences for a large number of RBPs from high throughput data, without requiring any specific domain knowledge or feature engineering. Deep learning has improved state of the art in several fields such as image classification, speech recognition and even genomics. Deep learning approaches obviate the need for careful feature engineering by learning useful representations directly from the data. We propose two deep architectures and use them to predict RNA-protein binding. Based on recent findings that show the importance of RNA secondary structure in RBP binding, we incorporate computationally predicted secondary structure features as input to our models and show its effectiveness in boosting prediction performance. We demonstrate that our models achieve significantly higher correlations on held out in-vitro testing data compared to previous approaches. We show that our model can generalize well to in-vivo CLIP-SEQ data and achieve higher median AUCs than other approaches. We demonstrate that our models discover known preferences for proteins such as CPO and VTS1 as well as report other proteins for which we find secondary structure playing an important role in binding. We demonstrate the strengths of our model compared to other approaches such as the ability to combine information from long distances along the sequence input.

Acknowledgements

I would like to sincerely thank everyone who supported me in the pursuit of my Masters degree. I am very grateful to my supervisors Prof. Brendan Frey and Prof. David Duvenaud for their invaluable guidance and support during the course of my thesis. I would also like to express my gratitude to Leo J. Lee and Andrew Delong for mentoring me and providing direction to my work. I am also grateful to Nathan Killoran and Alireza Makhzani for helpful discussion on my work.

Contents

Chapter 1

Background

1.1 Molecular Biology

1.1.1 Central dogma of molecular biology

In almost all cells the expression of genetic information occurs by DNA specifying the synthesis of RNA, followed by RNA specifying the synthesis of polypeptides that subsequently form proteins. Due to its universality, the flow of genetic information as DNA→ RNA → polypeptide(protein) is known as the central dogma of molecular biology. The two successive steps involved are described as follows:

- transcription: A stretch of DNA containing at least one gene is copied into RNA. If the gene encodes a protein then the copied RNA is known as messenger RNA(mRNA). This occurs in the nucleus for eukaryotes

- translation: mRNA is decoded into a chain of amino acids or a polypeptide that later folds into a protein and performs functions in the cell. This occurs in the ribosomes found in the cytoplasm.

1.1.2 Role of RNA binding proteins

In eukaryotes, transcription and translation processes occur in separate compartments of the cell, the nucleus and cytoplasm respectively. This allows them to have extensive post transcriptional regulation mediated by RNA binding proteins(RBPs). RBPs fine tune gene expression by regulating various stages of processing pre-mRNA and generating a large diversity of processed mRNA from the genome. This is done by regulating maturation, stability, transport and degradation of cellular RNAs. Several pre-mRNA processing reactions are regulated by RBPs, including splicing, editing and polyadenylation. For instance, HuR binds to target mRNA to enhance its stability and translation[10] whereas, TIA-1 and TIAR suppress mRNA translation[23]. There are a large number of RBPs in eukaryotes, with vertebrates having thousands, each having unique RNA-binding preferences[15]. One explanation for the large number of RBPs in eukaryotes is that they have evolved highly specific post-transcriptional processes to fine-tune gene expression and hence require a large number of RBPs to control the process of gene expression[3]. Due to their critical role in regulating post-transcriptional expression, mutations in RBPs or their binding sites can lead to several diseases such as muscular atrophies and neurological disorders

[35, 38]. Since RBPs are involved in several stages of post-transcriptional regulation, understanding their binding preferences is crucial in understanding RNA processing, localization and regulation.

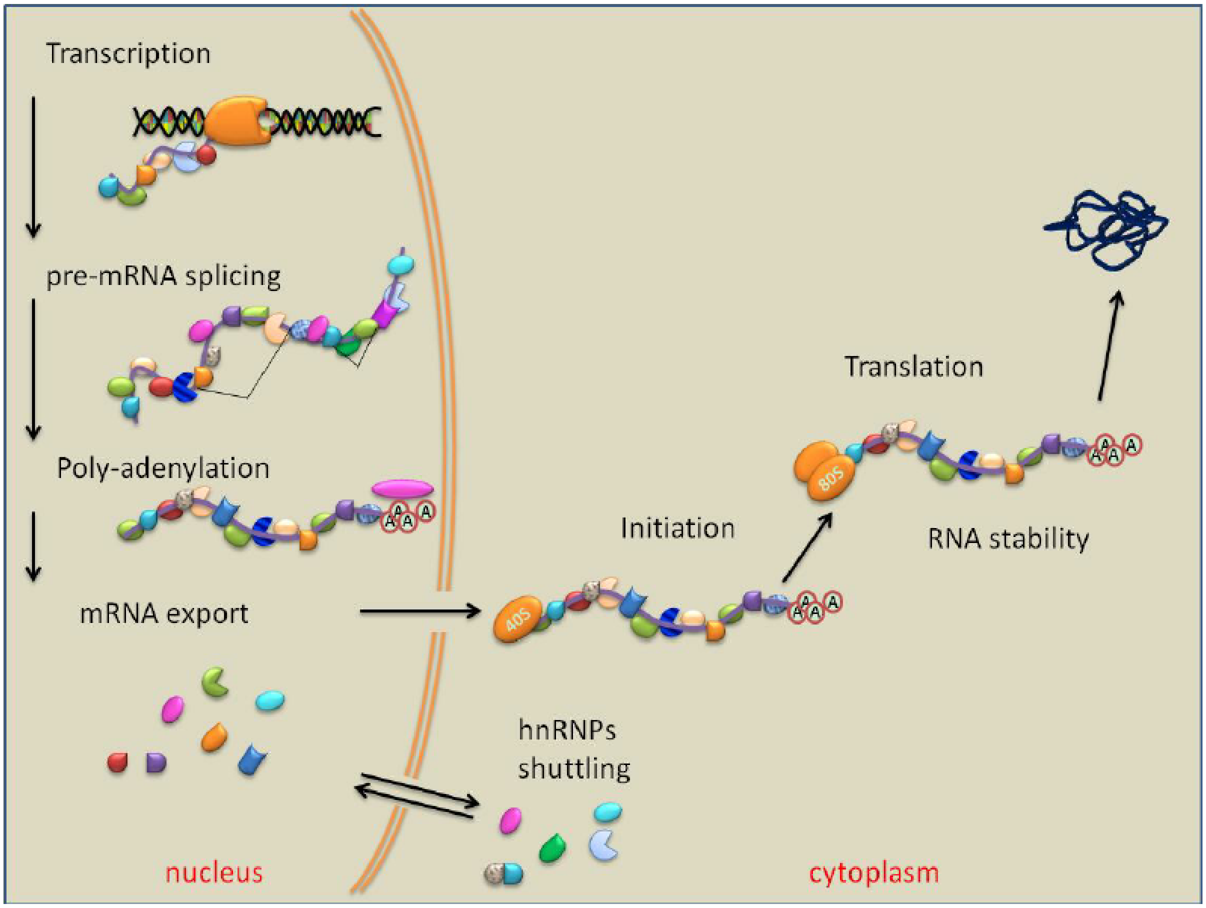


Figure 1.1: An illustration of the role of RBPs in the various stages of pre-mRNA processing.[44]

1.1.3 RBP binding preferences and role of structure

RNA is a single stranded molecule and folds onto itself, forming structures stabilized by hydrogen bonding between its bases. Thus an RNA molecule has both a specific order of bases and a secondary structure that can dictate what proteins bind to it. RBPs are known to have a preference for both a specific sequence order and secondary structure of a portion of RNA[17]. The shape of RNA which results from its folding is represented as base pairing between its nucleotides as shown in figure (1.2). There are several computational methods that predict the secondary structure of an RNA sequence from the order of its bases based on thermodynamic stability constraints[47, 34]. The structures can be represented as graphs[21] or average probability vectors over the ensemble of all probable structures[34]. In addition, there are experimental methods that can probe the RNA structure[12] but such data is scarce. It is commonly assumed that most proteins bind to accessible sites and few prefer a specific structural context[31, 42]. For instance, it is known that Vts1p is a yeast RBP that preferentially binds the sequence motif CNGG within RNA hairpins[4]. Even though specific sequence preferences can be learnt from in-vitro experiments, a very small fraction(15%-40%) of specific RBP sequence motifs are occupied in-vivo[49]. It has been shown that local secondary structure restricts access to a large subset of sequence motifs that would otherwise be bound [49].

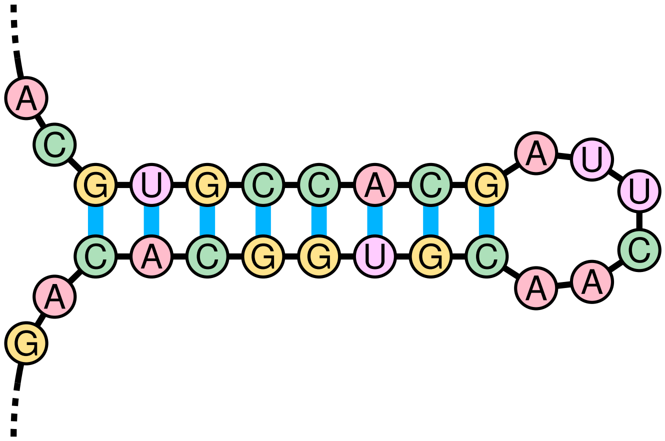


Figure 1.2: An example of base pairing in RNA resulting in a hairpin loop secondary structure[52]

1.1.4 Experimental methods for investigating RBP binding

Experimental methods for investigating RNA-protein binding can be categorized into *in-vivo* and *in-vitro* based on the whether they are performed in a living cell or in a controlled environment outside the cell, respectively. Before the development of high throughput methods, methods such as *in-vitro* Electrophoretic mobility shift assay(EMSA) [9] and *in-vivo* fluorescence[8] methods were used to investigate RNA-protein interaction in small-scale experiments. High-throughput sequencing based experimental techniques such as CLIP-seq [33, 26, 18], SELEX [11, 48], RNAcompete [42, 43], RNA Bind-n-Seq [28] allow for the measurement of RBP binding affinities in a transcriptome-wide manner. However, high-throughput methods do not yield quantitative biophysical measurements to quantify the binding energy such as the dissociation constant (K_d). Among the high-throughput techniques, the *in-vivo* methods HITS-CLIP, CLIP-seq and RIP-seq provide high quality test data for bench-marking binding models but owing to the high amount of noise and low resolution[13, 25], training models on them directly is challenging. In-vivo experiments are complicated by the presence of other RBPs and binding measurements could be the result of competition or complex formation between them. In-vitro experiments offer higher resolution, less noise and more accurate computational secondary structure prediction[45]. RNAcompete[42, 43] is a high-throughput in-vitro method that measures the binding intensities of over 200 RBPs to more than 240 000 probe sequences designed to cover each 9-mer at least 16 times. The large amount of data available for the experiments, allows for training data-driven models such as deep neural networks and is the basis of the data used for this work. A new approach RNAcompete-S[7] allows for querying a large diversity of RNA primary sequences and secondary structures. The RNAcompete-S method improves upon the RNAcompete[43] method which only allowed limited secondary structure representation and had probe sequences which were designed to be unstructured. The RNAcompete-S dataset is better suited to infer sequence-structure models since the probe sequences have stronger secondary structures. However, the data is available only for seven RBPs and performance results using other computational models are not available for this dataset at the time of this writing.

1.1.5 Models for RBP binding

Several methods have been developed to model RBP binding preferences including methods that rely on both sequence and secondary structure. MEMERIS [20] was the first approach to incorporate unpaired nucleotide probabilities and explored binding in unpaired regions. The RNAcompete assay provided a comprehensive analysis of binding preferences of RBPs covering 205 genes and 24 different eukaryotes[43]. Several methods have been trained on data from RNAcompete including Deepbind[2], RNAcontext[22], and RCK[40]. Deepbind was the first approach to use deep neural networks to learn RBP binding preferences from RNAcompete data. It uses a convolutional neural network to model the mapping from sequence to binding intensity. However, Deepbind did not incorporate secondary structure as a feature. RNAcontext is a model that incorporates structure in the form of structural context probability vectors that represent the ensemble of all possible structures. It learns a position weight matrix(PWM) to represent the sequence motif and global structure preferences to represent the preference of an RBP to bind to a given structural context. Since it uses a PWM, RNAcompete is position independent and cannot model relationships between different individual positions and how they affect binding. RCK extends RNAcontext to a k-mer model that uses local structure preferences. For each possible k-mer the RCK model learns a sequence score and a score for each structural context. Due to its design that learns a weight for each possible k-mer, the RCK method is limited to small values of k ($k \leq 6$) and hence cannot model longer positional dependencies. Also, it loses resolution by averaging the structural context across a k-mer and cannot model single position specific preferences. The RNAcompete-S approach [7] also uses a k-mer model that combines sequence and structure components and learns predictive k-mer features using logistic regression.

1.2 Machine Learning

Machine learning is the study of algorithms to understand and make predictions based on data. Machine learning algorithms are especially useful in solving problems where hard-coded rules or heuristics are difficult to define. Even for simple problems such as identifying handwritten digits, an approach based on predefined rules requires an enormous amount of rules and exceptions to those rules. Machine learning algorithms approach such problems by defining an adaptive mapping from input x to the output y, as $f(x)$, and learning the rules of the mapping based on the data. Parametric models used a fixed size of parameters to define the model and use the data to learn the values for those parameters. In addition to learnable parameters, these models require additional calibration parameters that are set before the training, such as the number of learnable parameters, the architecture of the model etc. These additional parameters are known as *hyperparameters*. Machine learning approaches often split the data into a *training set*, *validation set* and a *test set*. The training set is used to learn the parameters of the model, the validation set is used to compare between models trained from the data and choose the right setting for hyperparameters and finally, the test set is used to evaluate the model ability to *generalize* to data it has never seen. Machine learning has been applied in recent years to several areas such as identifying objects in images[27], transcribing speech to text[16], and understanding the genetic code[32, 2].

1.2.1 Deep learning and distributed representations

Conventional machine learning methods rely heavily on the choice of data representation, also known as features. The choice of such features determines the success of conventional machine learning approaches and requires considerable domain expertise and fine tuning. Representation learning is a set of methods for learning representations from raw data that can be useful for any classification or prediction task[6]. Deep learning techniques are representation learning techniques that learn multiple levels of representation by the composition of several non-linear transformations. The rationale behind stacking several layers of transformations is the assumption that the data is generated by several underlying factors correlated in a hierarchical fashion. The features in the lower layers encode low-level statistics of the data such as edges in an image, and higher layers build up on lower layers to represent more abstract concepts such as contours and faces. Deep learning approaches use distributed representations to represent features. To distinguish between $O(2^N)$ input spaces, a dense distributed representation requires only $O(N)$ parameters[6]. The combination of different hidden features allows the model to generalize to new combinations of learned features beyond those seen during training [29]. Composing layers of distributed representations affords another exponential increase in the expressive power of the representations[37].

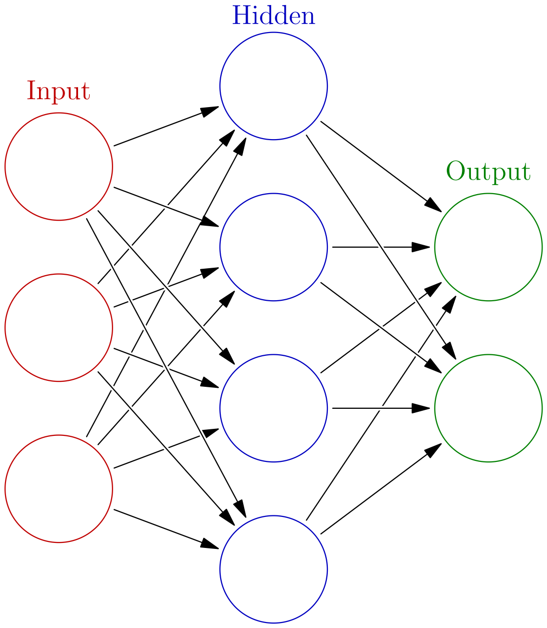
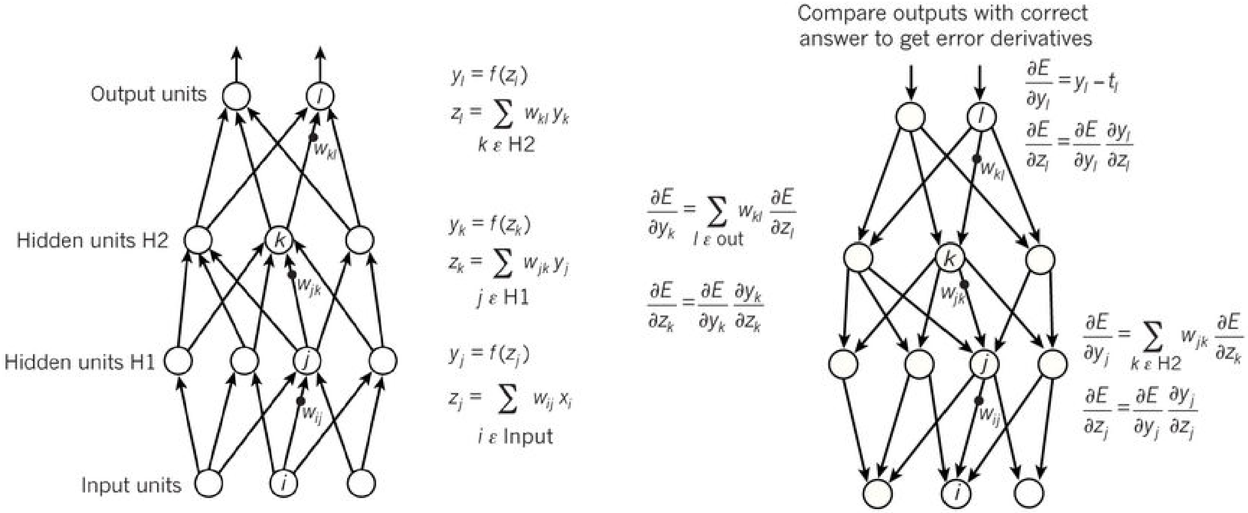


Figure 1.3: An illustration of a feed-forward neural network made up of an interconnection of neurons[51]

1.2.2 Feed Forward Neural Networks

An artificial neural network is a computational graph that relates an input vector x to an output y through a series of sub-units known as *neurons*. As shown in figure (1.3) each neuron receives as input a weighted linear combination of inputs or outputs from other neurons and applies a non-linearity to produce an output that gets forwarded to other computational units. These weights are like adjustable 'knobs' that define the input-output mapping. The neurons can model hidden variables within the data and are hence also known as hidden units. A feed forward network is an artificial neural network where connections between neurons do not form cycles and the values of a layer of hidden units are completely determined given the previous layer. This sequential computation of values based on the topological order of units from input to output is known as *forward propagation*.

In order to learn the weights of the connections in the neural network the standard approach is to use the *backpropagation algorithm*[46]. The backpropagation algorithm computes the gradient of the cost function with respect to the weights, which for each weight indicates how the cost function would change if the weight was perturbed by a small amount. Figure (1.4) explains the procedure for learning the weights.


Output units
Hidden units H2
Hidden units H1
Input units
w_{kl}
w_{jk}
w_{ij}
$y_l = f(z_l)$
$z_l = \sum_{k \in H2} w_{kl} y_k$
$y_k = f(z_k)$
$z_k = \sum_{j \in H1} w_{jk} y_j$
$y_j = f(z_j)$
$z_j = \sum_{i \in Input} w_{ij} x_i$
Compare outputs with correct answer to get error derivatives
$\frac{\partial E}{\partial y_l} = y_l - t_l$
$\frac{\partial E}{\partial z_l} = \frac{\partial E}{\partial y_l} \frac{\partial y_l}{\partial z_l}$
$\frac{\partial E}{\partial y_k} = \sum_{l \in out} w_{kl} \frac{\partial E}{\partial z_l}$
$\frac{\partial E}{\partial z_k} = \frac{\partial E}{\partial y_k} \frac{\partial y_k}{\partial z_k}$
$\frac{\partial E}{\partial y_j} = \sum_{k \in H2} w_{jk} \frac{\partial E}{\partial z_k}$
$\frac{\partial E}{\partial z_j} = \frac{\partial E}{\partial y_j} \frac{\partial y_j}{\partial z_j}$
i
j
k
l

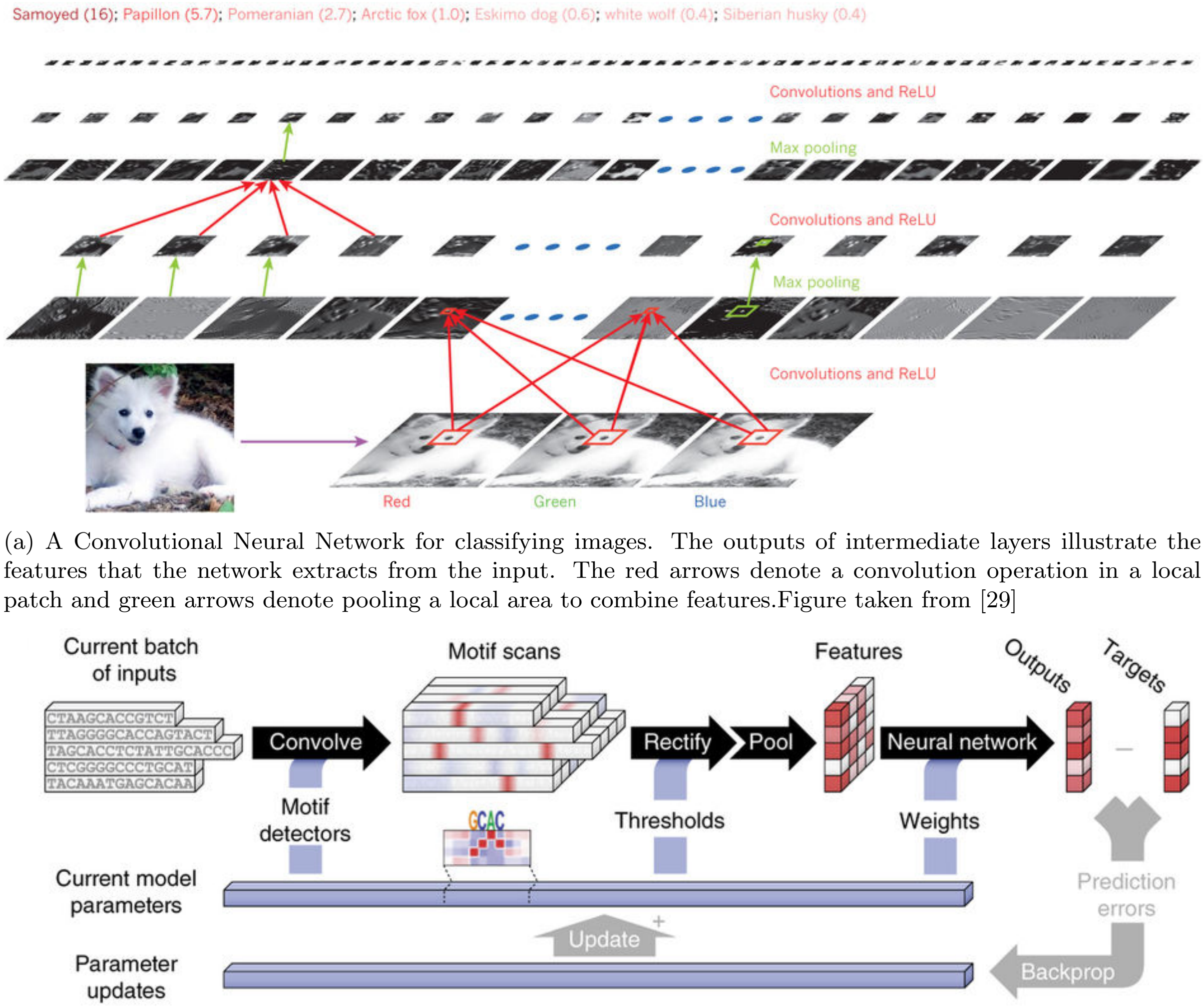
(a) Forward propagation

(b) Backward propagation

Figure 1.4: (a) Computing the forward pass in a neural network involves computing the weighted combination of inputs to a neuron, denoted here as z and passing it through a non-linearity to produce the activation of the unit y as $f(z)$. (b) Computing the gradient with respect to each unit involves computing a weighted sum of the gradients from the layer above it and multiplying with respect to the gradient of its activation $f(z)$. This gradient is then passed on to lower layers. Figure taken from [29]

1.2.3 Convolutional Neural Networks

Convolutional neural networks(CNNs) or ConvNets are neural networks that compute the correlation of filters with local patches and pass it through a non-linearity such as a rectified linear unit(ReLU) to produce the activation for the patch. A ReLU activation is defined as $\text{ReLU}(x) = \max(0, x)$ and allows smooth propagation of gradients through layers in deep architectures. The convolution filter is moved along the input to compute the local correlations and a feature map is produced. ConvNets are useful for data where local groups of data are highly correlated and form distinct patterns or motifs. The same set of filters is used for each position in the input, which gives makes ConvNets invariant to shifting the local motifs along the input. The convolutions are usually coupled with pooling layers that take the average or maximum of a feature map in a local patch or along an entire axis. Pooling allows the network to merge similar features into one and allows each unit to look at a larger portion of the input, known as the *receptive field* of the unit. Though CNNs have been used extensively for image classification tasks[27, 19] they have also recently been shown to be very effective in handling genomic sequences [2, 30]. Since DNA is a discrete sequence it can be encoded as one-hot-encoding, which encodes a sequence of length k into a matrix of dimension $k \times 4$ with each column representing one of the four bases (A,G,C,T). An entry in the encoding vector is set to 1 if that position contains the base corresponding to the column and 0 otherwise. As shown in figure (1.5(b)) feature maps produced from the encoded input show the location of motifs relevant to the task.



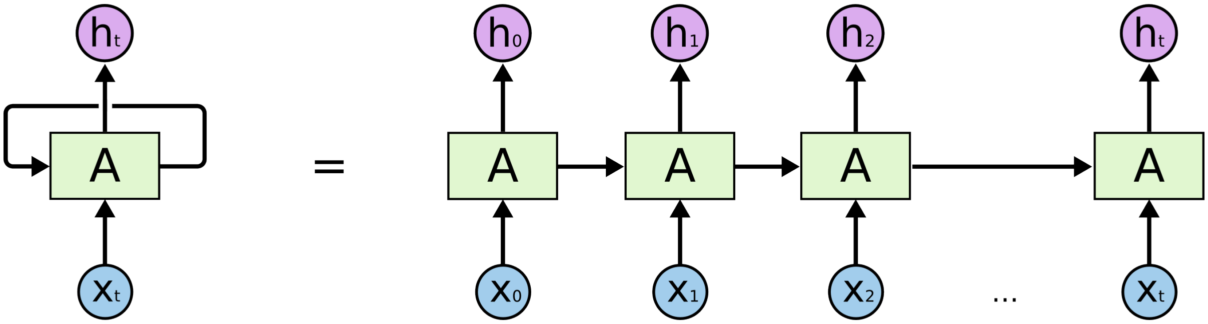
(a) A Convolutional Neural Network for classifying images. The outputs of intermediate layers illustrate the features that the network extracts from the input. The red arrows denote a convolution operation in a local patch and green arrows denote pooling a local area to combine features.Figure taken from [29]

(b) A Convolutional Neural Network for scoring DNA sequences. The feature maps act as local motif detectors. Pooling along the sequence allows the model to combine feature scans along the sequence length and be invariant to the location of the motif. Figure taken from [2]

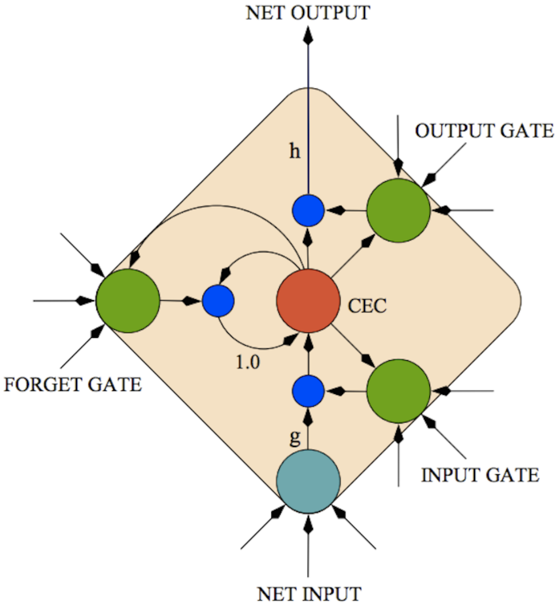
Figure 1.5: Examples of Convolutional Neural Nets used in literature

1.2.4 Recurrent Neural Networks and Long Short Term Memory

Recurrent neural networks(RNNs) are networks for processing sequential data. They do this by sharing parameters across time steps and thus are able to share useful representations between them. RNNs process the input sequence one element at a time, maintaining a state vector that contains a summary of all observations seen so far. The input at each time step is used to update the hidden state. The output of the time step is produced by a non-linear combination of the input at the given time step and the state from the previous time step. To train an RNN, it is unfolded in time and the weights across the time steps are restricted to be the same(Figure(1.6(a)). Once unfolded RNNs are similar to very deep feed-forward networks with shared weights across layers. A practical problem in training RNNs on long sequences is that the gradient diminishes or explodes in value due to the accumulation of gradients from non-linear activations in each time step[41]. To remedy the vanishing gradient problem recurrent architectures with gating mechanisms such as Long Short term Memory(LSTM)[14] were introduced. As shown in Figure(1.6(b)), LSTM recurrent units have an explicit memory cell, the natural behavior of which, is to remember inputs for a long time. The LSTM cell accumulates the external signal while copying its own internal memory and has a multiplicatively gated forget gate that controls when to clear the memory. This allows the gradient to pass smoothly and for the model to learn long term dependencies.



(a) (Left)Recurrent networks allow information in a sequence to persist by using the past input to update its state. (Right) A recurrent network unrolled in time to illustrate the flow of data. Figure taken from [39]



(b) An LSTM cell with input, output and forget gates. The forget gate determines what parts of the previous cell state need to be forgotten. The input gate determines what values of the input affect the cell state and finally the output gate determines what values of the cell state affect the output of the cell. Figure taken from [5]

Figure 1.6: Recurrent networks and LSTM cells

Chapter 2

Methods

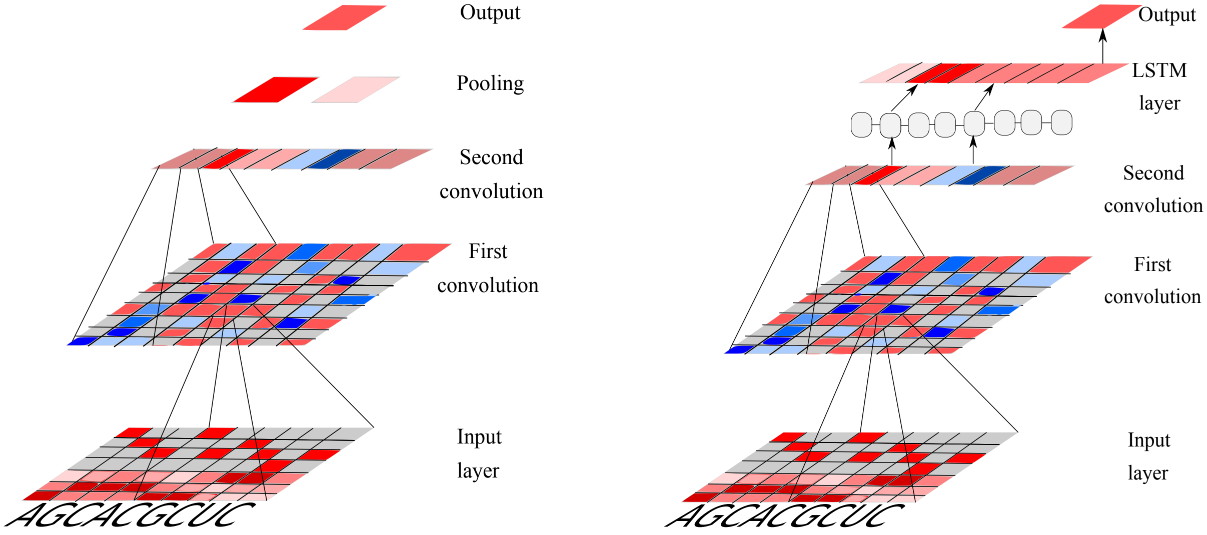
2.1 The cDeepbind model

2.1.1 Overview and model comparison

We have developed two deep learning models for jointly modelling RBP sequence and secondary structure preferences. We have named the models developed in this work as Context-Deepbind or cDeepbind. The cDeepbind-CNN model has a fully convolutional architecture with two convolution layers followed by global max and average pooling in the final layer. The cDeepbind-RNN also has two convolution layers but an extra LSTM layer that processes the feature map produced by the convolution to produce the score from its final time-step. The CNN model is faster since computation for the entire sequence can be done in parallel, whereas the RNN model requires processing the feature map sequentially. Also since the CNN model does not have any sub sampling or fully connected layers, the value of a position in the final feature map is only influenced by inputs in positions that are in the receptive field of that value. Thus, from the model it is possible to produce a binding track that shows the binding scores for each position in the sequence. However, the LSTM model is more powerful and can model more complex input-output relationships. The convolution layer can take advantage of local correlations in the input such as sequence motifs and structural contexts and produce a feature map that highlights relevant parts of the input. The LSTM layer can then model complex interactions between different parts of the input and produce a summary vector that can then be used to predict the binding score for the entire sequence.

2.1.2 Model architecture

The model architectures for cDeepbind models are described in figure(2.1). We use a filter width of 16 for the convolution filters since most RBP motifs are known to be short. The number of filters for the convolution is selected randomly from $\{8, 16, 24\}$. The convolution layers use a ReLU activation. The cDeepbind-RNN model uses a recurrent layer of LSTM cells. The number of hidden units in the cells is a hyperparameter and selected randomly from $\{10, 20, 30\}$. We use random normal initialization for the weights and initialize the biases to a small positive value. The model is trained with mini-batches of 100 inputs using the ADAM optimizer[24] owing to its effectiveness in training recurrent architectures that have sparse gradients.



(a) cDeepbind-CNN architecture

(b) cDeepbind-RNN architecture

Figure 2.1: (a) The cDeepbind-CNN architecture uses two convolution layers followed by global max and average pooling. The values of the feature map in the second convolution layer **q** is a summary of local information within its receptive field. Thus, the value in q_k depends on $(s_k, \ldots, s_{k+w})$, where w is the receptive field of units in **q**. The pooling then combines these features along the sequence length to produce the final prediction (b) The cDeepbind-RNN architecture uses an LSTM layer to process the feature map of the convolution layers to produce a summary vector **o**. Since each sequential output in the LSTM is influenced by past inputs in the sequence o_k depends on $(s_0, \ldots, s_{k+w})$, where w is the receptive field of units in **q**

2.1.3 Input and pre-processing

The models take as input a sequence $s = (s_1, \ldots, s_n)$ from the alphabet $A = \{A, G, C, T\}$ and a structural annotation vector $\mathbf{r} = (r_1, \ldots, r_n)$ where $r_i \in R^5$ and denotes the probability of the position being in the contexts (paired(P), hairpin loop(H), inner loop(I), multi-loop(M) or external region(E)). The structural context is computed using a variant of RNAplfold [34] provided by the RNAcontext authors[22], that annotates the sequences into the aforementioned five structural profiles as opposed to the usual paired and unpaired profiles from RNAplfold. The sum of each structural context probability for a given q_i sums to 1. The sequence input is encoded as a four-dimensional one-hot-encoding and is appended to the structural context vector to give a 9-dimensional input vector. The input is passed through two ReLU activated convolution layers that have multiple filters to produce a one-dimensional track of the same length as the input. The recurrent layer uses each sequence position as the time-step for recurrence. The final output vector is then reduced to a scalar score (f) using a linear layer and is the predicted score for the input. The true scores are taken to be the RNAcompete probe intensities pre-processed to have the scores greater than the 99.95^{th} percentile to be clamped at the value of the 99.95^{th} percentile as done in Deepbind and RCK. Furthermore, the scores are normalized to have a mean of zero and a variance of one to ensure that consistent scales for learning rates and initial values of weights can be used without regard to individual intensity ranges for each protein. The loss function used is the mean squared error(MSE)

$$loss(f, t) = \frac{1}{2}(f - t)^2 + \lambda||\mathbf{W}||_2 \tag{2.1}$$

Where f is the predicted binding intensity and t is the true binding intensity from RNAcompete. We also add an l_2 penalty on the weights $\mathbf{W}$ as a regularizer.

2.1.4 Training pipeline and hyperparameter tuning

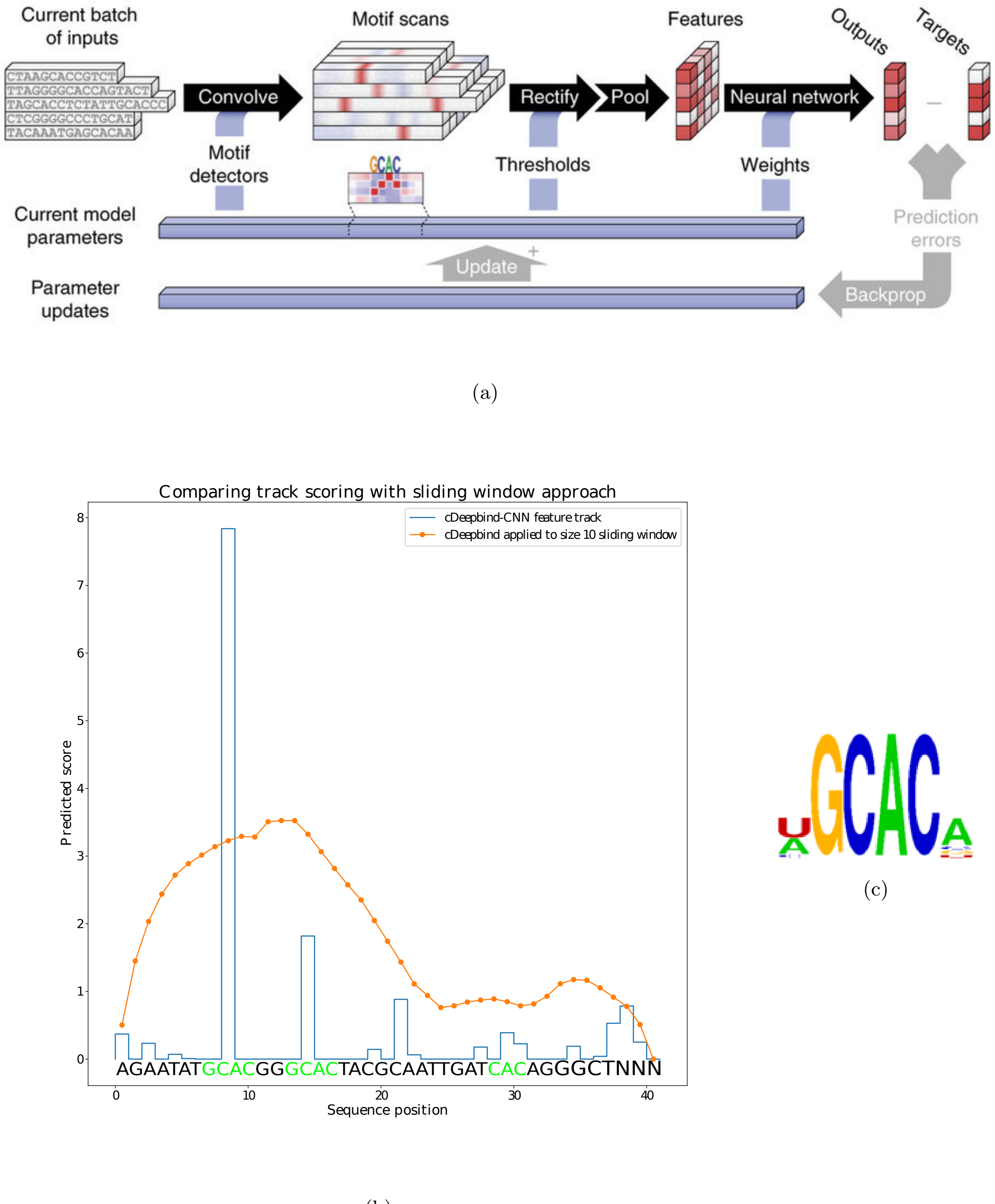
We use random hyperparameter search using 3-fold cross validation to find good hyperparameter settings for our models. The training pipeline for training on the RNAcompete 2013 dataset is described in figure(2.2). We sample hyperparameter settings for each protein and choose the setting with least average validation cost on 3-fold cross validation. We then use this choice of parameters to train 3 models in parallel to ensure we do not get a poor model due to poor random initialization. We find that the most important hyperparameters for the model were the learning rate, rate of weight decay, number of LSTM units and variance of the random normal initializer for the weights. The code for the model was written in Tensorflow[1] and the models were run on NVIDIA Tesla K80 GPUs. The training and evaluation time for a single protein on a single GPU while training 3 candidate models in parallel is about 20 minutes for the cDeepbind-CNN model and 60 minutes for the cDeepbind-RNN model. We split the list of proteins on 15 GPUs and were able to train models in parallel for the complete list of 244 proteins in 6 hours for the CNN model and 18 hours for the RNN model.

Algorithm 1: Training procedure for RNAcompete 2013 dataset

```
for $protein \in protein\ list$ do
    $\theta_1, \ldots, \theta_5 \leftarrow$ Select random hyperparameters ;
    $model_1(\theta_1), \ldots, model_5(\theta_5) \leftarrow$ Initialize;
    for $\theta_i \in \{\theta_1 \ldots \theta_5\}$ in parallel do
        set $cost_i$=0;
        for $k \in \{1, 2, 3\}$ do
            $cost_{ik}$ = cost of model($\theta_i$,$fold_k$) on validation fold $k$
        end
        $cost_i$= $\sum_{k=1}^{3} cost_{ik}$;
    end
    $\theta_{best} = \theta_{argmin(cost_1,\ldots,cost_5)}$ $model_1(\theta_{best}), \ldots, model_3(\theta_{best}) \leftarrow$ Initialize;
    set $cost^{test} = 0$;
    for $i \in \{1, 2, 3\}$ in parallel do
        $cost_i^{test}$ = cost of $model_i(\theta_{best})$ on test set
    end
    $model_{best} = model_{argmin(cost_1^{test},\ldots,cost_3^{test})}$
end
```

Figure 2.2: Pseudo-code for the training pipeline

2.1.5 Comparison of cDeepbind-CNN architecture with Deepbind



(a)

(b)

(c)

Figure 2.3: (a)The original Deepbind model used a convolution layer followed by a fully connected layer[2](b)Comparing binding track obtained from cDeepbind with the binding score from a sliding window. The sequence is a high affinity sequence for CPO which is known to bind to a GCAC motif. The sequence is shown below the plot with the motif highlighted(c) Sequence logo for CPO [43]

For cDeepbind-CNN we use an architecture that does not have any transformation that destroys the order of inputs in the sequence. The feature map from the second convolution $\mathbf{q}$ retains the positional order of the input. In the Deepbind architecture, a fully connected neural network was used on the pooled outputs of the convolution filters (Figure(2.3(a))) to produce the final score. Since the network learns weights to combine the features to get the final score, it would be invariant to shuffling of the features, and it loses information about the order of elements in the input sequence. Hence, Deepbind can only produce a single binding score for its entire input window. To illustrate this limitation we applied cDeepbind-CNN to a sequence with high affinity for CPO taken from the RNAcompete test set(Figure2.3). The CPO protein is known to have a high affinity for GCAC motifs (Figure2.3(c)). We observe that the binding track $\mathbf{q}$ obtained from the cDeepbind model highlights the location of the motif in the sequence with high resolution. To compare with a Deepbind like architecture we applied the cDeepbind model to overlapping sliding windows of size 10 over the sequence. The score for each position was assigned as the average score from all windows that overlapped with that position. We observe that the binding track provides much more fine grained information than the sliding window approach and saves greatly on computation time. Thus, cDeepbind-CNN can be feasibly applied to transcriptome wide tracks without computing averages for sliding windows, which is not possible with a Deepbind like architecture.

Chapter 3

Results

We compared the cDeepbind model with other methods such as RCK, Deepbind and RNAcontext. We observed that our model outperforms all other approaches and can learn relevant sequence and structural preferences for the proteins under study.

3.1 cDeepbind is more accurate at in vitro binding prediction than state of the art

3.1.1 Comparison on RNAcompete 2013 dataset

We compared the performance of our model on predicting probe intensities on the RNAcompete dataset[43] across 244 experiments. Similar to the RCK method[40], we observe that even though the probes in this experiment were designed to be unstructured we can still learn some structural preferences from it. The data consists of a training set of sequences (set A) and a held out test set (set B). The models are trained on set A and the performance measure is taken as the Pearson correlation of predicted intensities with the probe intensities on the test set. We used the results published in the RCK study[40] for comparison against other methods. As illustrated in figure(3.1), our model significantly outperforms all other methods on in-vitro binding prediction. We improved upon the previous deep learning approach Deepbind that did not incorporate structure features and did not use a recurrent layer in its architecture. cDeepbind-RNN achieved an average Pearson correlation of 0.556 versus 0.435 for the original Deepbind model(P-value $= 4.46 \times 10^{-37}$) as shown in figure(**??**). Our models also outperformed the state of the art method RCK, that used both sequence and structure features as input which achieved an average Pearson correlation of 0.461 (P-value $= 3.33 \times 10^{-33}$) illustrated by the dot plot in figure(**??**). We found that the average relative improvement in Pearson correlation over RCK was 22.35% over the 244 experiments. We believe the improvement in performance results from the model being able to recognize longer patterns in the input and to use structural information to better discriminate between sequences in the input.

3.1.2 Comparison on the original RNAcompete 2009 dataset

To test our models ability to pick up binding preferences from data containing highly structured sequences we trained it on data from the original RNAcompete study[42]. The study contained experiments for 9

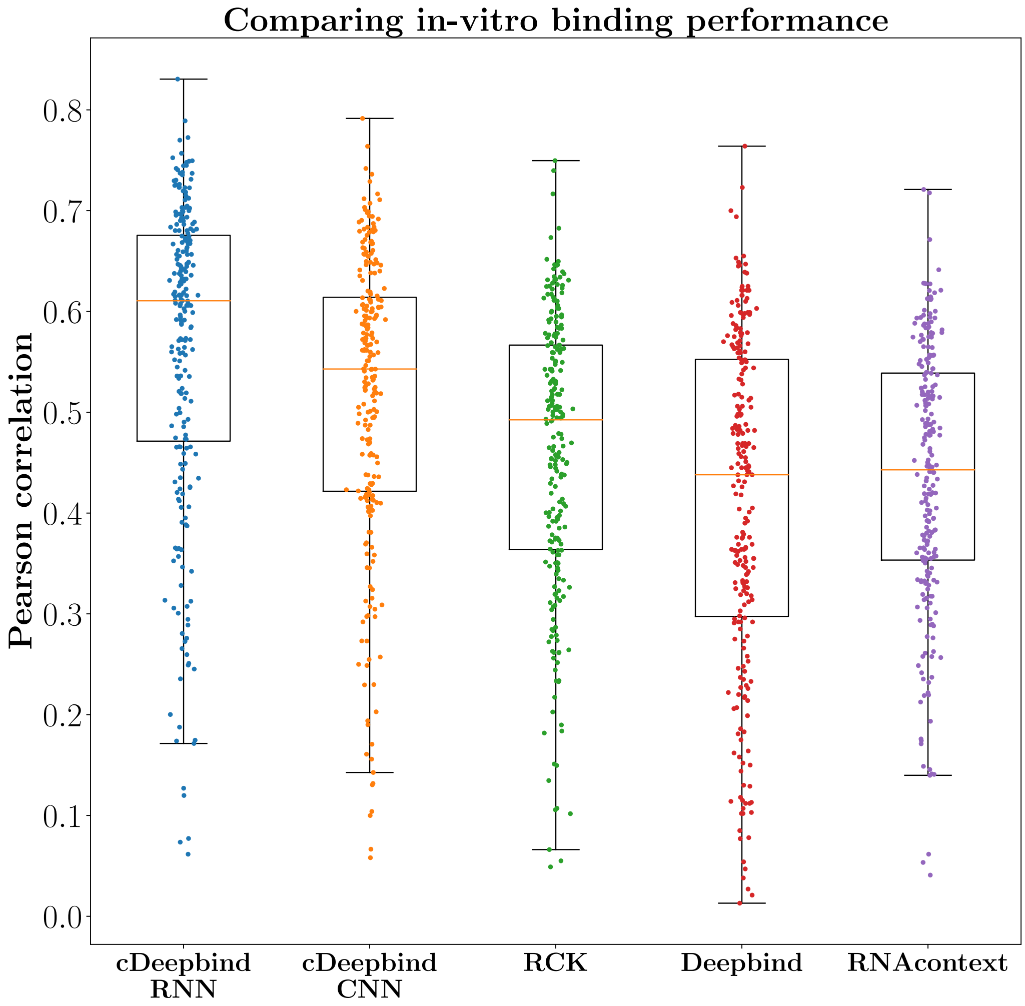
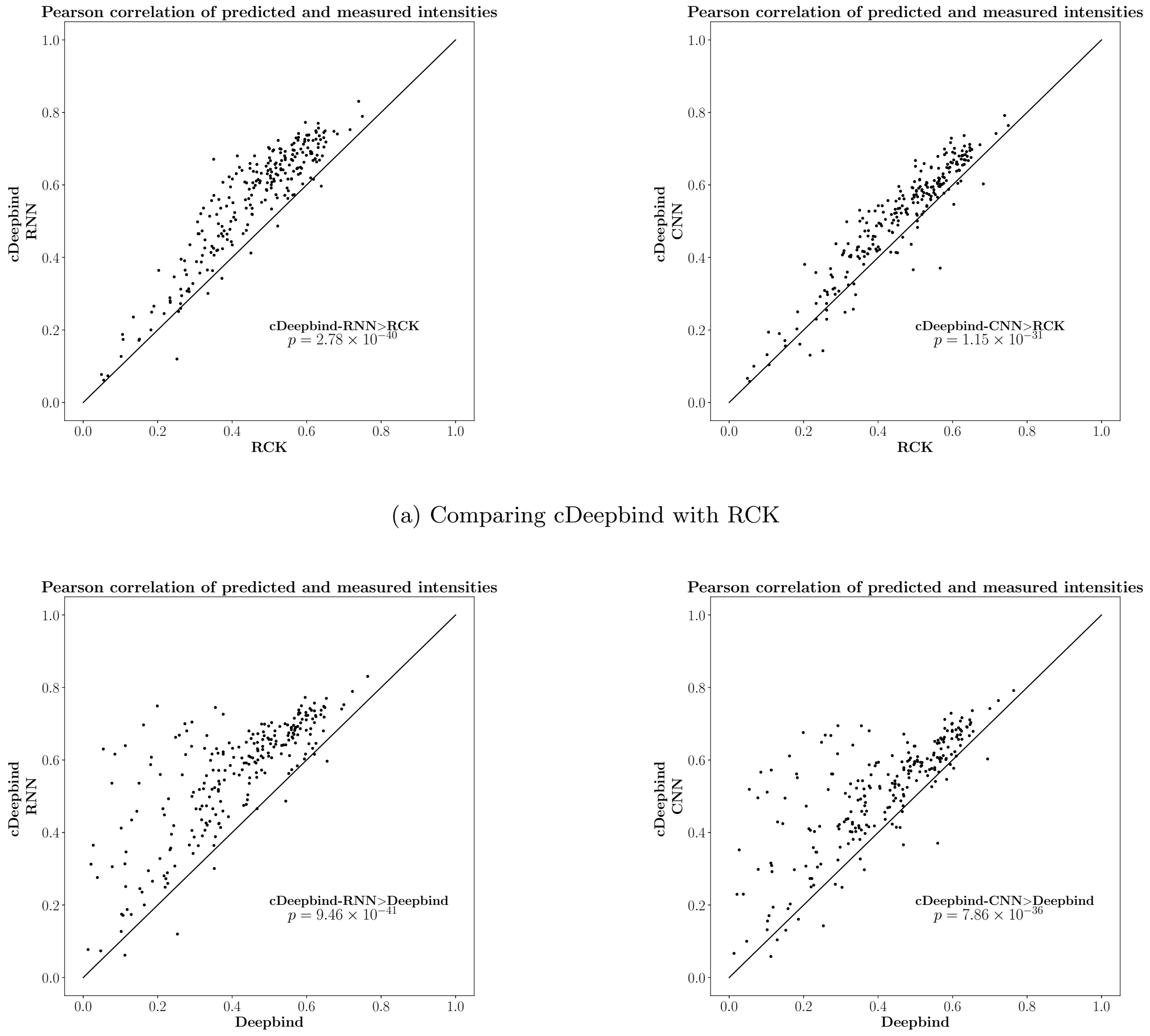


Figure 3.1: Box plot comparing the performance of extant methods on *in-vitro* binding prediction on the RNAcompete dataset. All models were trained on Set A sequences and tested on Set B. Dots represent Pearson correlation on the test set for 244 experiments.



(a) Comparing cDeepbind with RCK

(b) Comparing cDeepbind with the original Deepbind model

Figure 3.2: The scatter plots illustrate the effectiveness of cDeepbind in predicting *in-vitro* binding. The models were trained on Set A sequences in the RNAcompete experiment and tested on Set B sequences. The points in the plot represent Pearson correlation of predicted and measured intensities on the 244 experiments. *P*-values were calculated using Wilcoxon signed-rank test

Method Protein	RNAcontext	RCK	cDeepbind-RNN
Fusip	0.457	0.632	0.640
HuR	0.719	0.764	0.799
PTB	0.399	0.488	0.499
RBM4	0.707	0.747	0.750
SF2	0.602	0.689	0.696
SLM2	0.535	0.613	0.688
U1A	0.458	0.583	0.669
VTS1	0.516	0.526	0.537
YB1	0.229	0.376	0.370
Average	**0.514**	**0.602**	**0.628**

Table 3.1: Comparing cDeepbind with other methods on the original RNAcompete 2009 study.

proteins each having two sets of sequences. Each set was designed to cover all possible 10 base sequences in an unstructured context and all possible 7 and 8 base sequences within a hairpin loop context. We trained our model on set A for each protein and evaluated it on set B. The performance comparison for the different proteins is given in table (3.1). Our model outperformed RCK by achieving an average Pearson correlation of 0.628 compared to 0.602 for RCK (P-value = 7.5×10^{-3}). We obtained big improvements (> 0.05 in Pearson correlation) for the proteins U1A and SLM2.

3.2 cDeepbind model is sensitive to secondary structure

To evaluate the impact of secondary structure in the improvement of predictions we trained the cDeepbind-RNN model without secondary structure as input. Both sets of models were trained with an LSTM output layer and the same hyperparameter search space for the sizes of the hidden units. We found that the model that has access to secondary structure features performs better with an average Pearson correlation of 0.561 versus 0.528 for the model that does not use secondary structure features(P-value=2.20×10^{-27}). We find that the relative improvement is greatest in some proteins that have a known secondary structure preference such as CPO, RBFOX1. We also discovered that several other proteins had a big improvement in predictions upon including secondary structure, such as SF1, PUM, MSI, CG14718 and others. We observe that our CNN model performs better than other methods RCK and RNAcontext that also use secondary structure features, but does not perform as well as our RNN model(Figure(3.1).

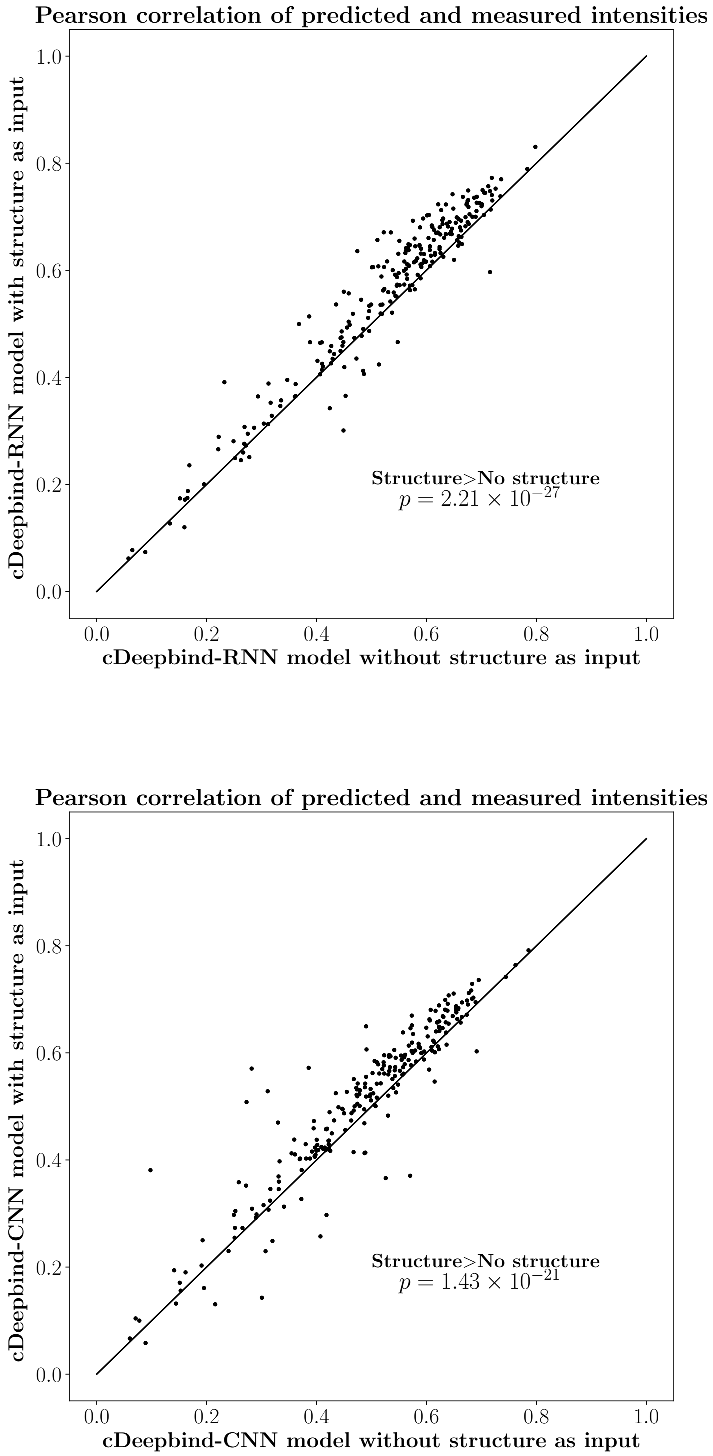
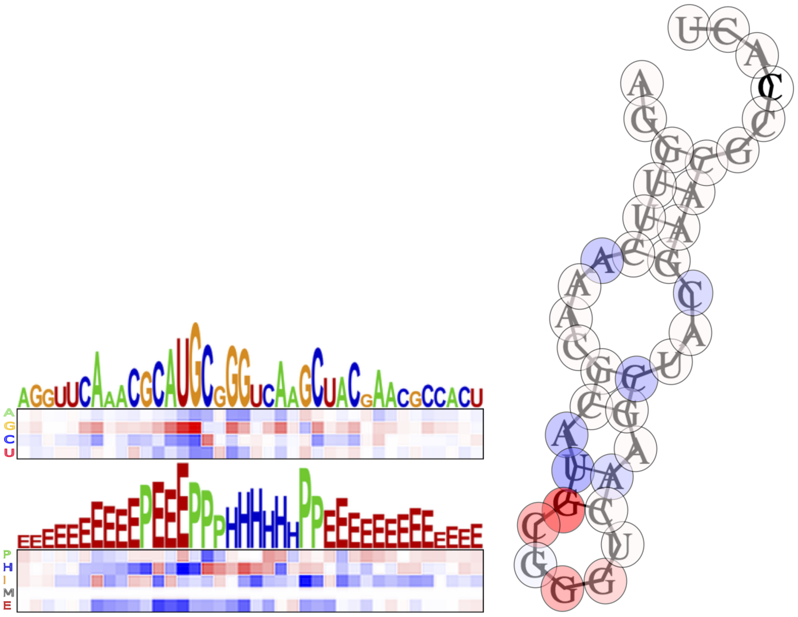


Figure 3.3: Comparing cDeepbind models with and without structure as input

3.2.1 cDeepbind discovers known structural preferences for VTS1p

To visualize the binding preferences learned by the cDeepbind-RNN model we looked at some sequences that had the highest experimentally observed binding intensities for the protein and visualized the gradient of the loss function with respect to the input. It should be noted here that the input to the model are discrete sequences encoded as one-hot encoded vectors. The gradient with respect to the one-hot-vector highlights the parts of the sequence that the model is sensitive to. Even though the model treats the input space as continuous, it is useful to think of the direction of the gradient at a given position being an indicator of different inputs at that position resulting in either an increase or decrease in the overall score. We illustrate the sensitivity of the model to different parts of the input using a mutation map. The mutation map is a color map with four rows corresponding to the bases A,G,C,T in the input from top to bottom. The cells are colored on a spectrum from blue to red with white meaning zero gradient. The red cells correspond to a predicted increase in score while the blue cells correspond to a predicted decrease in score. Our model confirms the known binding preference of VTS1p to bind to CNGG motifs within hairpin loops. The gradient shows that the model is sensitive to the hairpin region flanked by paired nucleotides . The mutation map shows that when the model encounters the CNGG motif, it is sensitive to the C and two Gs, with all other nucleotides in their positions predicting a decrease in score and the model does not care about the nucleotide sandwiched between the C and G since it shows almost a zero gradient in that cell.

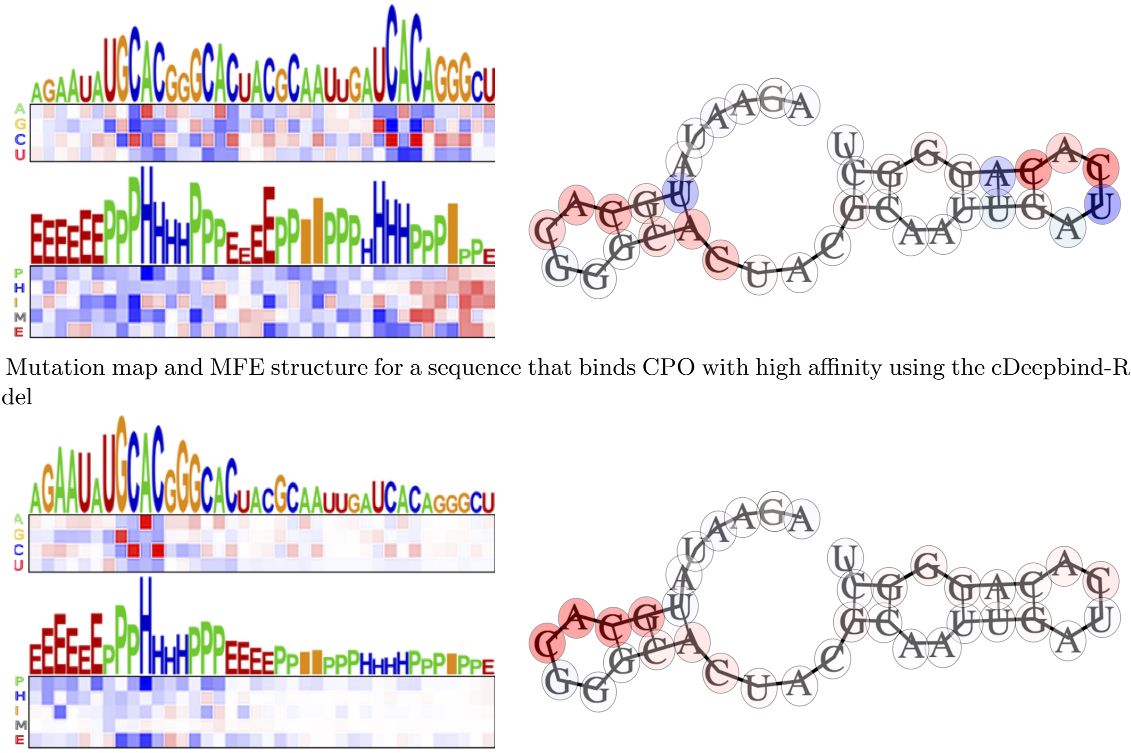

AGGUUCAAACGCAUGCGGGUCAAGCUACGAACGCCACU
A
G
C
U
EEEEEEEEEEEPEEEPPPHHHHHHPPEEEEEEEEEEEE
P
H
I
M
E

(a) Mutation map(left) and Minimum Free Energy(MFE) structure(right) for a sequence that binds VTS1 with high affinity using the cDeepbind-RNN model

Figure 3.4: An illustration of the gradients for a sequence having high binding target score for VTS1p in the RNAcompete 2009 dataset. The mutation is a color map along the length of the sequence with blue cells denoting a decrease in the score and red cells denoting an increase in score predicted by the gradient at the position. The structure diagram is the minimum free energy(MFE) structure predicted by RNAfold. The gradient is overlaid onto the structure to illustrate how each position affects the score.

3.2.2 Comparing CNN and RNN models for CPO

We investigated the trained models for the protein CPO to compare and contrast between the RNN and CNN models and to verify binding patterns known in literature. CPO is a protein that is known to bind to the GCAC motif[43]. In a recent work [50] the binding specificity of the protein RBPMS was investigated. The RRM domain of RBPMS was reported to bind to a pair of tandem CAC motifs spaced by a variable nucleotide window. It was also reported that the majority of the residues of the RNA recognition motif(RRM) domain of RBPMS were strictly conserved in the corresponding RRM of CPO. Thus it is reasonable to assume similar binding patterns for RBPMS and CPO.



(a) Mutation map and MFE structure for a sequence that binds CPO with high affinity using the cDeepbind-RNN model

(b) Mutation map and MFE structure for a sequence that binds CPO with high affinity using the cDeepbind-CNN model

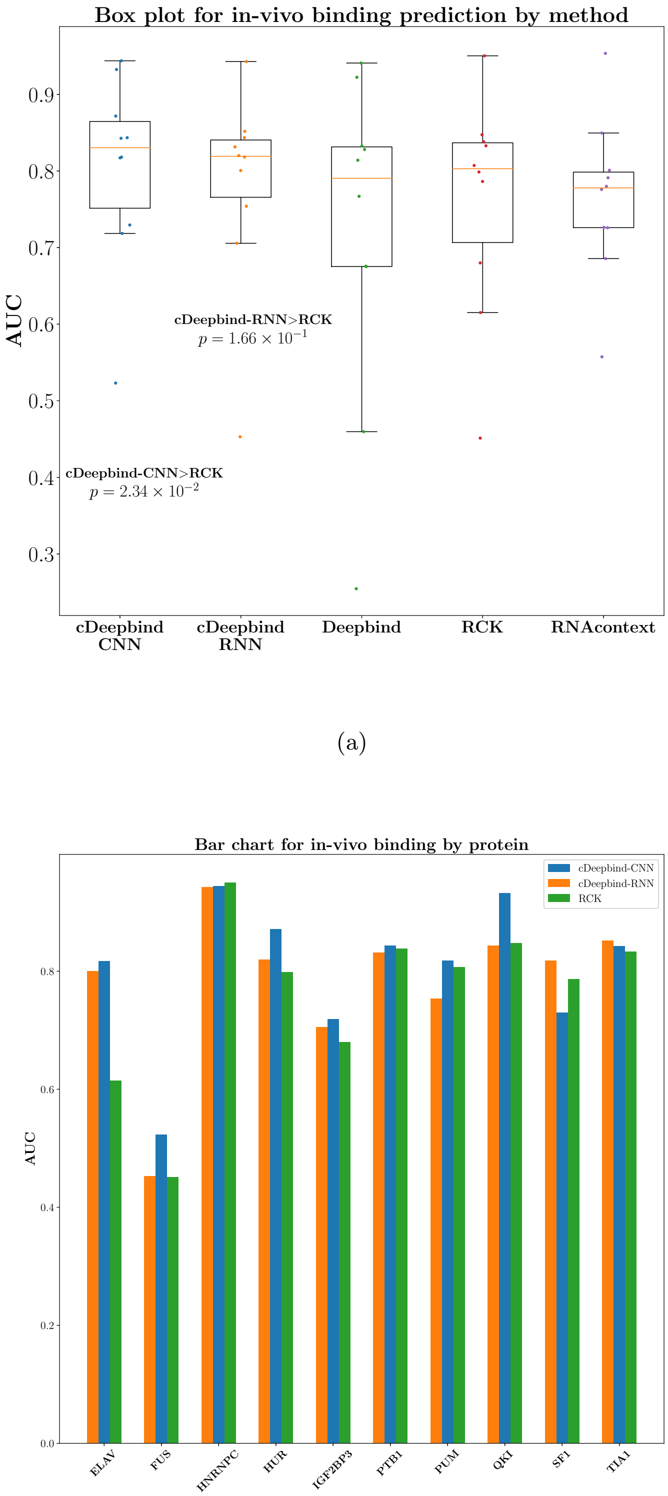
Figure 3.5: An illustration of gradients for the CNN and RNN models on a sequence that has a high target score for binding with CPO. The mutation map illustrates the difference between modelling methods for the two models. The RNN model accumulates the contribution from multiple motifs, whereas the CNN model pools the binding track to be strongly biased by what it considers the strongest site.

We take the same sequence that was analyzed in section 2.1.5, which is a sequence from the test set of the CPO model that has normalized target score of 26.90. The cDeepbind-RNN model assigns a prediction of 23.91, whereas the cDeepbind-CNN model assigns a score of 4.1. As illustrated in figure(3.5), both the models are sensitive to the *CAC* motifs within the sequence, with the gradient suggesting that any alterations in those positions in the sequence would lead to a lower score. The gradient also suggests that the models would have preferred a *G* instead of a *U* preceding the *CAC* at the far right of the sequence which agrees with the literature motif for CPO. We also see here that the CNN model focuses on just one motif and does not take into account the combined effect of the multiple motifs in the sequence. The RNN however can handle non-additive effects of multiple motifs in a better way due to its architecture and is able to make a prediction closer to the target score. It is also worth noting that the action of multiple CAC motifs predicted by the RNN model agrees with the observations regarding the RRM domain of RBMPS[50]. In summary, this example illustrates the ability of the RNN model to combine information along the sequence in a non-linear way and its advantage over the CNN in handling such dependencies.

3.3 Comparison on in vivo binding data

We evaluated our model on 23 pairs of RNAcompete and CLIP experiments that covered 10 proteins. The CLIP experiments were taken from the GraphProt study[36]. The CLIP datasets contained in vivo binding sites collected from CLIP experiments. Control sequences were extracted from unbound regions of the same bound transcripts. The dataset contained flanking sequences of length 150 nucleotides on both sides of both bound and control sequences. We used the sequences including flanks for secondary structure prediction with RNAplfold while only the sequence and predicted probabilities were used for testing. Since the RCK paper bench marked models trained on the entire set of sequences including the test set, we retrained models for this comparison on the larger set of sequences. The performance values for RCK, RNAcontext and Deepbind were taken from the RCK study.

We compared both our CNN and RNN models that use structure features against other methods(Figure(3.6)). The AUC for a protein was taken as the average AUC for all pairs for RNAcompete-CLIP experiments that covered the proteins. To compare against other methods we consider the median of the AUCs for the 10 proteins. We find that our both our CNN and RNN models beat other methods achieving median AUCs of 0.8304 and 0.8191 respectively against 0.803 for RCK and 0.790 for Deepbind. The p-values between cDeepbind-CNN and the next best performing method, RCK, was 0.023, whereas the p-value between cDeepbind-RNN and RCK was 0.166(Wilcoxon signed-rank test). We believe we do not see as big of a boost in performance in-vivo as compared to in-vitro because of the noisy nature of the data and inaccuracy of secondary structure prediction in-vivo. In addition there are very few proteins in the in-vivo evaluation set to quantify improvements accurately.



(a)

(b)

Figure 3.6: Comparison of cDeepbind models with previous methods. We obtain a significant improvement in performance in-vivo with our CNN model.

Chapter 4

Conclusions

We have presented a deep learning approach to model RBP binding preferences that incorporates both sequence and secondary structure information. Our approach provides an improvement over k-mer models by using distributed representations in the form of convolutional filters to model the binding preferences. Our models can learn preferences for long motifs in the input. We show that our CNN model can produce binding tracks from the input and our RNN model can store past context in its internal state to make better predictions. Our models significantly outperform previous methods on in-vitro evaluation based on Pearson correlation on held out test data.On in-vivo evaluation we achieve improvement in median AUC on both of our models, with the improvement being significant for the CNN model. We believe in-vivo data for more proteins that overlap with the set of proteins for which we had training data would demonstrate the improvement in performance more significantly.

Our models show a large gap in performance with and without secondary structure information for some proteins that are known to be sensitive to secondary structure such as VTS1p and RBFOX1. In addition we discover such difference for other proteins such as SF1, PUM and MSI which do not have known secondary structure preferences.

Even though we applied our model to the RNAcompete in-vitro data, our approach is general and the model can learn relevant features from any data source. It would be useful in the future to train and validate on RNA-bind-n-seq[28] and RNAcompete-S[7]. Future directions for this work could include training on in-vivo data, training a multitask prediction model for similar proteins, and generative modelling of binding sequences. It would also be useful to explore additional architectures and pooling schemes such as noisy-OR pooling in the final layer which would yield a binding probability and make the scores more interpretable.

Bibliography

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[2] Babak Alipanahi, Andrew Delong, Matthew T Weirauch, and Brendan J Frey. Predicting the sequence specificities of dna-and rna-binding proteins by deep learning. *Nature biotechnology*, 33(8):831–838, 2015.

[3] Vivek Anantharaman, Eugene V Koonin, and L Aravind. Comparative genomics and evolution of proteins involved in rna metabolism. *Nucleic acids research*, 30(7):1427–1464, 2002.

[4] Tzvi Aviv, Zhen Lin, Giora Ben-Ari, Craig A Smibert, and Frank Sicheri. Sequence-specific recognition of rna hairpins by the sam domain of vts1p. *Nature structural & molecular biology*, 13(2):168–176, 2006.

[5] Franoise Beaufays. The neural networks behind google voice transcription, Aug 2015.

[6] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[7] Kate B Cook, Shankar Vembu, Kevin CH Ha, Hong Zheng, Kaitlin U Laverty, Timothy R Hughes, Debashish Ray, and Quaid D Morris. Rnacompete-s: Combined rna sequence/structure preferences for rna binding proteins derived from a single-step in vitro selection. *Methods*, 2017.

[8] John Czworkowski, OW Odom, and Boyd Hardesty. Fluorescence study of the topology of messenger rna bound to the 30s ribosomal subunit of escherichia coli. *Biochemistry*, 30(19):4821–4830, 1991.

[9] Albert E Dahlberg, C Wesley Dingman, and Andrew C Peacock. Electrophoretic characterization of bacterial polyribosomes in agarose-acrylamide composite gels. *Journal of molecular biology*, 41(1):139IN21147–146IN26, 1969.

[10] Isabel López de Silanes, Ming Zhan, Ashish Lal, Xiaoling Yang, and Myriam Gorospe. Identification of a target rna motif for rna-binding protein hur. *Proceedings of the National Academy of Sciences of the United States of America*, 101(9):2987–2992, 2004.

[11] Andrew D Ellington and Jack W Szostak. In vitro selection of rna molecules that bind specific ligands. *nature*, 346(6287):818, 1990.

[12] Ryan A Flynn, Qiangfeng Cliff Zhang, Robert C Spitale, Byron Lee, Maxwell R Mumbach, and Howard Y Chang. Transcriptome-wide interrogation of rna secondary structure in living cells with icshape. *Nature protocols*, 11(2):273–290, 2016.

[13] Xiang-Dong Fu and Manuel Ares Jr. Context-dependent control of alternative splicing by rna-binding proteins. *Nature Reviews. Genetics*, 15(10):689, 2014.

[14] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. 1999.

[15] Tina Glisovic, Jennifer L Bachorik, Jeongsik Yong, and Gideon Dreyfuss. Rna-binding proteins and post-transcriptional gene regulation. *FEBS letters*, 582(14):1977–1986, 2008.

[16] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pages 6645–6649. IEEE, 2013.

[17] Jörg Hackermüller, Nicole-Claudia Meisner, Manfred Auer, Markus Jaritz, and Peter F Stadler. The effect of rna secondary structures on rna-ligand binding and the modifier rna mechanism: a quantitative model. *Gene*, 345(1):3–12, 2005.

[18] Markus Hafner, Markus Landthaler, Lukas Burger, Mohsen Khorshid, Jean Hausser, Philipp Berninger, Andrea Rothballer, Manuel Ascano, Anna-Carina Jungkamp, Mathias Munschauer, et al. Transcriptome-wide identification of rna-binding protein and microrna target sites by par-clip. *Cell*, 141(1):129–141, 2010.

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[20] Michael Hiller, Rainer Pudimat, Anke Busch, and Rolf Backofen. Using rna secondary structures to guide sequence motif finding towards single-stranded regions. *Nucleic acids research*, 34(17):e117–e117, 2006.

[21] Stefan Janssen and Robert Giegerich. The rna shapes studio. *Bioinformatics*, 31(3):423–425, 2014.

[22] Hilal Kazan, Debashish Ray, Esther T Chan, Timothy R Hughes, and Quaid Morris. Rnacontext: a new method for learning the sequence and structure binding preferences of rna-binding proteins. *PLoS computational biology*, 6(7):e1000832, 2010.

[23] Henry S Kim, Matthew CJ Wilce, Yano MK Yoga, Nicole R Pendini, Menachem J Gunzburg, Nathan P Cowieson, Gerald M Wilson, Bryan RG Williams, Myriam Gorospe, and Jacqueline A Wilce. Different modes of interaction by tiar and hur with target rna and dna. *Nucleic acids research*, 39(3):1117–1130, 2011.

[24] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[25] Shivendra Kishore, Lukasz Jaskiewicz, Lukas Burger, Jean Hausser, Mohsen Khorshid, and Mihaela Zavolan. A quantitative analysis of clip methods for identifying binding sites of rna-binding proteins. *Nature methods*, 8(7):559–564, 2011.

[26] Julian König, Kathi Zarnack, Gregor Rot, Tomaž Curk, Melis Kayikci, Blaž Zupan, Daniel J Turner, Nicholas M Luscombe, and Jernej Ule. iclip reveals the function of hnrnp particles in splicing at individual nucleotide resolution. *Nature structural & molecular biology*, 17(7):909–915, 2010.

[27] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[28] Nicole Lambert, Alex Robertson, Mohini Jangi, Sean McGeary, Phillip A Sharp, and Christopher B Burge. Rna bind-n-seq: quantitative assessment of the sequence and structural binding specificity of rna binding proteins. *Molecular cell*, 54(5):887–900, 2014.

[29] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[30] Michael Ka Kit Leung, Andrew Delong, and Brendan J Frey. Inference of the human polyadenylation code. *bioRxiv*, page 130591, 2017.

[31] Xiao Li, Hilal Kazan, Howard D Lipshitz, and Quaid D Morris. Finding the target sites of rna-binding proteins. *Wiley Interdisciplinary Reviews: RNA*, 5(1):111–130, 2014.

[32] Maxwell W Libbrecht and William Stafford Noble. Machine learning in genetics and genomics. *Nature Reviews. Genetics*, 16(6):321, 2015.

[33] Donny D Licatalosi, Aldo Mele, John J Fak, Jernej Ule, Melis Kayikci, Sung Wook Chi, Tyson A Clark, Anthony C Schweitzer, John E Blume, Xuning Wang, et al. Hits-clip yields genome-wide insights into brain alternative rna processing. *Nature*, 456(7221):464–469, 2008.

[34] Ronny Lorenz, Stephan H Bernhart, Christian Hoener Zu Siederdissen, Hakim Tafer, Christoph Flamm, Peter F Stadler, and Ivo L Hofacker. Viennarna package 2.0. *Algorithms for Molecular Biology*, 6(1):26, 2011.

[35] Kiven E Lukong, Kai-wei Chang, Edouard W Khandjian, and Stéphane Richard. Rna-binding proteins in human genetic disease. *Trends in Genetics*, 24(8):416–425, 2008.

[36] Daniel Maticzka, Sita J Lange, Fabrizio Costa, and Rolf Backofen. Graphprot: modeling binding preferences of rna-binding proteins. *Genome biology*, 15(1):R17, 2014.

[37] Guido F Montufar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In *Advances in neural information processing systems*, pages 2924–2932, 2014.

[38] Kiran Musunuru. Cell-specific rna-binding proteins in human disease. *Trends in cardiovascular medicine*, 13(5):188–195, 2003.

[39] Chris Olah. Understanding lstm networks, Aug 2015.

[40] Yaron Orenstein, Yuhao Wang, and Bonnie Berger. Rck: accurate and efficient inference of sequence-and structure-based protein–rna binding models from rnacompete data. *Bioinformatics*, 32(12):i351–i359, 2016.

[41] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pages 1310–1318, 2013.

[42] Debashish Ray, Hilal Kazan, Esther T Chan, Lourdes Pena Castillo, Sidharth Chaudhry, Shaheynoor Talukder, Benjamin J Blencowe, Quaid Morris, and Timothy R Hughes. Rapid and systematic analysis of the rna recognition specificities of rna-binding proteins. *Nature biotechnology*, 27(7):667–670, 2009.

[43] Debashish Ray, Hilal Kazan, Kate B Cook, Matthew T Weirauch, Hamed S Najafabadi, Xiao Li, Serge Gueroussov, Mihai Albu, Hong Zheng, Ally Yang, et al. A compendium of rna-binding motifs for decoding gene regulation. *Nature*, 499(7457):172–177, 2013.

[44] Maria Grazia Romanelli, Erica Diani, and Patricia Marie-Jeanne Lievens. New insights into functional roles of the polypyrimidine tract-binding protein. *International journal of molecular sciences*, 14(11):22906–22932, 2013.

[45] Silvi Rouskin, Meghan Zubradt, Stefan Washietl, Manolis Kellis, and Jonathan S Weissman. Genome-wide probing of rna structure reveals active unfolding of mrna structures in vivo. *Nature*, 505(7485):701, 2014.

[46] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

[47] Peter Steffen, Björn Voß, Marc Rehmsmeier, Jens Reeder, and Robert Giegerich. Rnashapes: an integrated rna analysis package based on abstract shapes. *Bioinformatics*, 22(4):500–503, 2005.

[48] Regina Stoltenburg, Christine Reinemann, and Beate Strehlitz. Selexa (r) evolutionary method to generate high-affinity nucleic acid ligands. *Biomolecular engineering*, 24(4):381–403, 2007.

[49] J Matthew Taliaferro, Nicole J Lambert, Peter H Sudmant, Daniel Dominguez, Jason J Merkin, Maria S Alexis, Cassandra A Bazile, and Christopher B Burge. Rna sequence context effects measured in vitro predict in vivo protein binding and regulation. *Molecular cell*, 64(2):294–306, 2016.

[50] Marianna Teplova, Thalia A Farazi, Thomas Tuschl, and Dinshaw J Patel. Structural basis underlying cac rna recognition by the rrm domain of dimeric rna-binding protein rbpms. *Quarterly reviews of biophysics*, 49, 2016.

[51] Wikipedia. Artificial neural network — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Artificial%20neural%20network&oldid=794673218`, 2017. [Online; accessed 14-August-2017].

[52] Wikipedia. Stem-loop — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Stem-loop&oldid=779234593`, 2017. [Online; accessed 13-August-2017].