The Interdisciplinary Center, Herzlia
Efi Arazi School of Computer Science
M.Sc. program - Research Track

Deep Learning and Sequence Determinants of Gene Co-Expression

by
Sharon Mayer Sultan

M.Sc. dissertation, submitted in partial fulfillment of the requirements
for the M.Sc. degree, research track, School of Computer Science
The Interdisciplinary Center, Herzliya

February 2021

This work was carried out under the supervision of Prof. **Zohar Yakhini** from the Efi Arazi School of Computer Science, The Interdisciplinary Center, Herzliya.

Deep Learning and Sequence Determinants of Gene Co-Expression

Sharon Sultan, Zohar Yakhini

February 2021

Abstract

Gene co-expression can be used as a second order indicator to associate genes with a biological process, to better understand the role of genes and to explore regulatory roles they may play in living organisms and systems. Tracing back and deciphering such connections, and the processes that form them, can be a complicated inference process. Along with that, deep neural networks are increasingly taking part in the field of computational biology. They are extensively used these days to learn, predict and unlock complex biological processes and hidden connections. In this paper we present the use of deep learning tools to predict co-expression connections and decipher their hidden genomic sequence determinants if such exist. We first explore the capacity of a deep neural network, in the context of predicting and discovering hidden motifs in synthetically generated data, built on top of actual genome parts. We then further explore and apply similar techniques on real gene co-expression data from yeast. We show that the deep learning based learning process we established here produces meaningful predictions and uncovers candidate co-expression associated motifs (CoAMs). We also show that some of these CoAMs are known to play regulatory roles in yeast.

Contents

1 Introduction

In the field of computational biology, many resources have been invested in the last few decades to acquire the knowledge of how genes encode to gene products. This knowledge can then be applied to classify and understand the role that genes and other elements play in biological process. Part of this effort materialized by taking samples from living organisms and measuring expression of genes and of gene products and regulators, including proteins and non-coding RNA molecules. With gene expression data widely available, gene co-expression analysis was found to be a useful tool for discovering and understanding biological regulatory processes; the processes that eventually control the transcription and translation of genetic sequences into the gene products. Expression data can be used to construct co-expression networks, such networks have been used to cluster and classify genes and to associate them with functional groups [7, 24, 27]. Nonetheless, this doesn't shed much light on the underlying cause that eventually results with a pair of genes having a correlated expression pattern. Over time, accumulated knowledge began to unravel that the regulation of gene expression involves a set of highly complex mechanisms. Those mechanisms are influenced by many factors, both internal and external[12], and can be active during any step of regulatory pathway. Not surprisingly, some of the those mechanisms are influenced by the genomic sequence of genes and their surrounding regions. [3, 21].

For almost a decade, the field of deep learning revolutionized the area of data processing. Combinations of increasingly growing computation power along continuously and rapidly evolving techniques, made it a go-to tool whenever there is a sufficiently large dataset involved. It earned its place and proved to be a very powerful tool for pattern recognition and detection tasks, and lately it has been widely applied in the field of genetic and genomic research for numerous tasks; it have been successfully used for predicting sequence specificities of binding proteins[1], predicting off-target activity in CRISPR-Cas9 [20], predicting gene expression [4], inferring spatial transcriptomics from (H&E) images[17, 18, 19], and more.

In recent work, reported by Tasaki et al. (2020)[26], addresses differential co-expression prediction using deep learning. In this paper they develop a model that utilizes deep learning to predict differential co-expression based on genomic binding sites on RNAs and promoters. They also report predicting negative/positive co-expression relationships between gene pairs, both in-tissue and inter-tissue, and investigate the key factors driving the co-expression.

Having a deep neural network performing well on the task it was trained to master is highly important, but not always a straight forward goal to achieve; being able to learn or deduce the hidden logic that guides a trained network in making decisions, is sometimes just as important and not less challenging. Up until recent years, most of those who were using deep neural networks, to say nothing of those who were not, were still considering the deep neural networks to be black boxes. One way to shed light into such black boxes, is to try to determine the importance of input features in producing a certain prediction, this is referred to as prediction attribution; a relatively recent emerging such technique is integrated-gradients[25]. Integrated-gradients uses sensitivity and implementation-invariance axioms to attribute deep neural network prediction to its input; those inputs could be pixels in a picture, individual words in text and in our case, nucleotide base positions in a genomic sequence. Examples for integrated gradients applications, and comparison to simple gradients prediction attribution is shown in Fig. 1

In this paper, we investigate the capabilities of deep neural networks in the context of learning and predicting sequence determinants of genes co-expression. This is motivated by the assumption that much of the process of regulation is driven by the genes' adjacent sequences (e.g. genes promoter region and 3'UTR regions). We show that for synthetic data the network can easily handle noise, which we would expect to be inherently present, and uncover those signals even when they are obscure. We also demonstrate how, with a well trained network, we can trace back and unveil the underlying cause for co-expression using integrated-gradients prediction attribution technique and minimum-hypergeometric (mHG) based search[8, 16]. We then challenge the fundamental motivating assumption by applying our method to actual yeast expression data. We first show that using adjacent input sequences (i.e. promoter regions and 3'UTRs) along with TF regulation vectors, which we assembled from existing regulation databases, the network is capable of learning to correctly predict much of the co-expression. We take this further and leave out the TF regulation vectors as the final challenge; we show that the network is still capable of predicting co-expression in a significant manner. Finally, we use this network, again along with integrated gradients and mHG statistical analysis, to propose some insights on the input adjacent regions, including the discovery of putative CoAM.

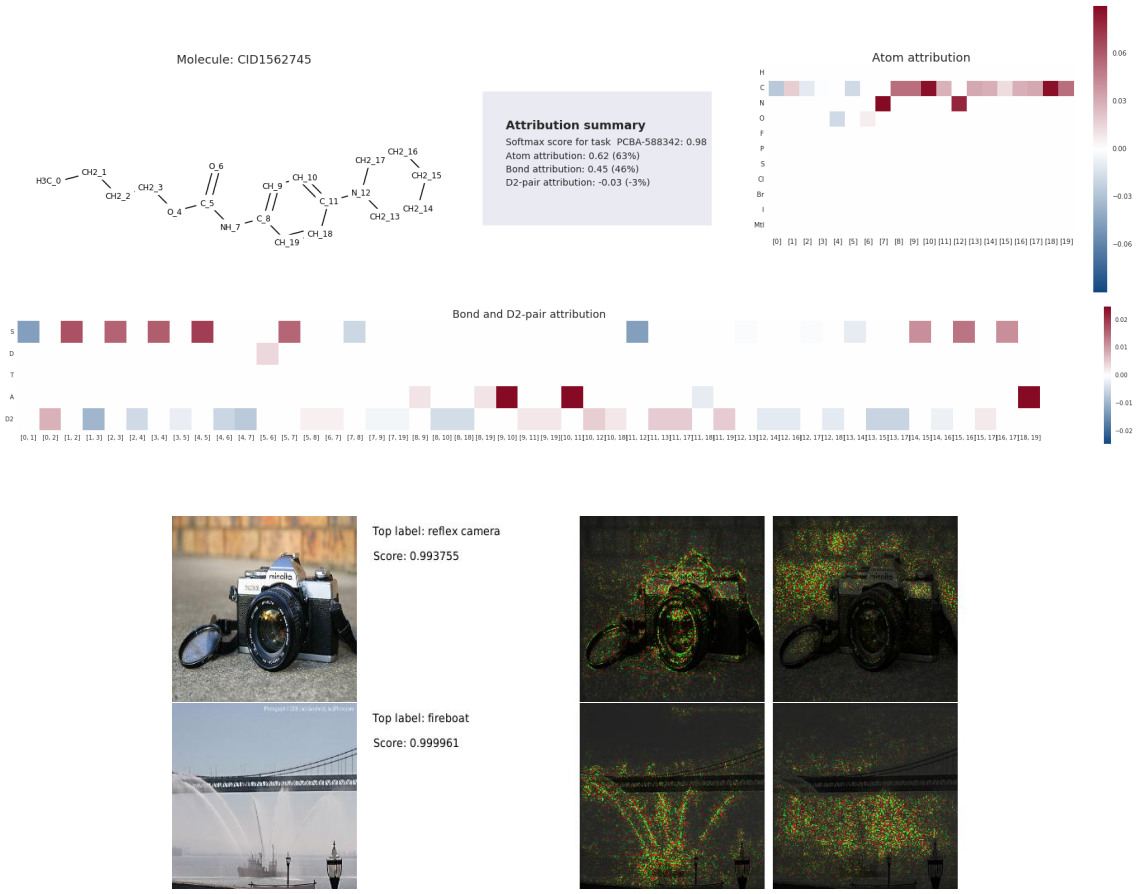


Fig. 1 **(left)** Attribution for a molecule under the W2N2 network [13]. By applying integrated gradients, Sundararajan et al. (2020)[25] were able find anomaly that revealed that parts of the input were left unintentionally not fully convolved. **(right)** Integrated gradients vs. gradients at the image. Left-to-right: original input image, label and softmax score for the highest scoring class, visualization of integrated gradients, visualization of gradients*image.

2 Methods

Here we discuss the structure of the neural-network used, as well as the process we constructed to generate both synthetic and real inputs for the learning and test phases. The network training was first tested on synthetic data, which we describe below. We also cover the process we used to identify motifs associated with co-expression using integrated-gradients. We call these co-expression associated motifs, or CoAM.

Data processing

Synthetic co-expression data

For the purpose of creating the synthetic data we used a real expression data taken from humans; the dataset we used is GSE66360[23] (Whole blood gene expression data). Our first step was to filter out genes with low coefficient of variation: CV < 0.2 out of the assumption that genes that almost don't vary will have little or even negative contribution to the co-expression analysis process. The next step was to calculate gene co-expression matrix. Namely, if E is the expression matrix with row entries $G = g_1, g_2, \dot{;} g_n$ representing the measured gene products, and columns $S = s_1, s_2, \ldots, s_m$ expression sample representing instances (individuals), we define r_{g_1,g_2} to be Spearman rank correlation of g_1 and g_2 over the samples S:

$$r_{g_1,g_2} = \rho_{rg_1,rg_2} = \frac{\text{cov}(rg_1, rg_2)}{\sigma_{rg_1}\sigma_{r_{g2}}}$$

where ρ denotes the Pearson correlation, rg_i is the ranked g_i expression vector, σ_x is standard deviation of a vector x, and $\text{cov}(x, y)$ is the covariance of the vectors x, y.

We divided the pairs (g_i, g_j) into two classes, based on a cutoff threshold applied to their measured co-expression. This gives rise to labeling every pair of genes g_1 and g_2 as co-expressed or not. For each gene we extracted its promoter region sequence based on UCSC hg38, the full human genome assembly [14, 11], and its annotations [10]. Using the annotations we considered promoter regions to be 200 bps upstream the transcription start site. We then observed the set of co-expressed pairs, and for all these pairs we modified their promoter region by planting in each promoter an occurrence of an 8-mer. These were randomly picked from a pool of 20 randomly pre-generated 8-mer motifs. The outcome of this process is a dataset of manipulated pairs

of promoters, labeled as co-expressed according to the presence of our synthetic co-expression associated motifs. See Fig. 2.

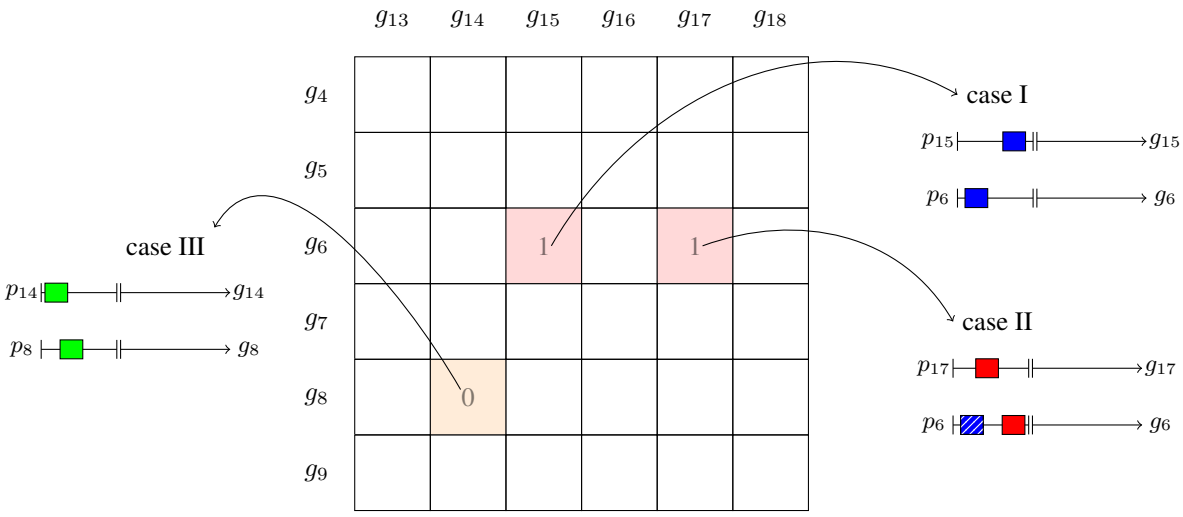


Fig. 2 Synthetic data generation - modifying promoter sequences according to co-expression matrix. **(case I)** randomly plant CoAM (blue box) in promoter regions of a pair of co-expressed genes. **(case II)** randomly plant a different CoAM (red box) in promoter regions of pair of co-expressed genes where on of the promoters has a previously planted CoAM (blue box with lines). **(case III)** A CoAM (green box) in pair of gene that are not co-expressed. This happens in one of two cases: either each promoter region share a previously planted CoAM or by chance both share a sequence identical to a CoAM.

Real dataset

The dataset we used was taken from GSE18121[6], which contains expression measurements of yeast subjected to heat stress through time course of 30 minutes. The total number of samples is 42: for 3 strains of yeast, at 7 time points (at 0, 5, 10, 15, 20, 25 and 30 minutes) with 2 replicates per each sample. The expression was measured using GPL90 Affymetrix microarray.

The yeast complete genomic sequence and its annotations were both obtained from Saccharomyces Genome Database (SGD) [9, 5]. Using this full chromosomal sequences and the annotations we extracted two sequences per gene: the first is the promoter region sequence, that is consisted of the 1000 nucleotides that precede the transcription start start. The other sequence is the most downstream untranslated region of the gene (the most downstream 3'UTR), which we considered to be the sequence that follows the end position of the last mRNA coding region, to its termination

position.

Besides these two sequences, we wanted to be able to provide the network with a hint vector; for this we used known regulating transcription factors (TFs). We constructed Gene to TF regulation matrix, where each gene is represented as a row entry in the matrix, and each TF as a columns. For each gene, known regulating TF, either up or down regulating, columns we assigned the value 1, while all other columns are 0. This regulation matrix was assembled using data retrieved from Yeastract[22]

Learning co-expression

Train test split

Co-expression relations between genes are not independent, meaning that certain genes tend to be co-expressed with many other genes. Therefore, a strict train-test-validation split is necessary in order to avoid having such prior as much as possible. To achieve this, the split has to be done over the complete set of genes, prior to the co-expression calculation. This way the input sample sequences are disjoint through all phases. See Fig. 3.

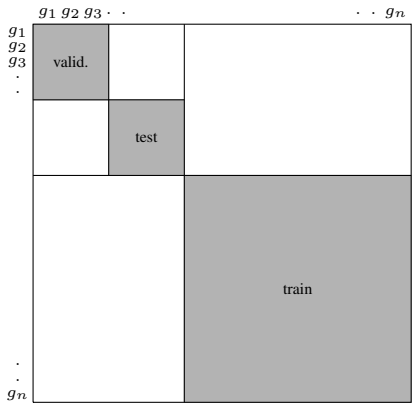


Fig. 3 Train-test-validation split. The set of genes is split into train, validation and test sets. This results in the above sparse block diagonal matrix as a mask for the co-expression matrix.

Convolutional neural network

In the field of machine learning and artificial neural networks (ANN), convolutional neural networks (CNN) are usually used to solve complex problems of pattern recognition and detection, primarily in images. CNNs are doing an outstanding job in extracting features from input images.

Roughly, this is done by passing the input through three types of layers. The first type is convolutional layers which act as high dimensional locally connected layers from the input, to higher dimensional output. The second type of layers is the pooling layers, which down sample the input along the image spacial dimension. The last type of layers are fully connected layers which act as classification layers on top of the extracted features from the usually interleaving convolutional and pooling layers.

For our classification task we consider our input sequences, encoded as one-hot vectors, to be similar to images; thus we decided to use a CNN for the task. A simplified schematic view of the architectural structure of the network used is shown is Fig. 4. The input to the network are one-hot matrices of shape: sequence length $\times$ nucleotides vocab size, that represent the input DNA sequences. DNA sequences we used are taken from promoter region of genes and from the most downstream 3'UTR (only in yeast data). Each input matrix then goes, separately, through several convolutional/pooling layers. Their output is then flattened into a single high dimensional vector, that goes through several fully connected layer. The output layers is a two dimensional vector for classification. The structure in details of the each CNN we used, is brought in Results section.

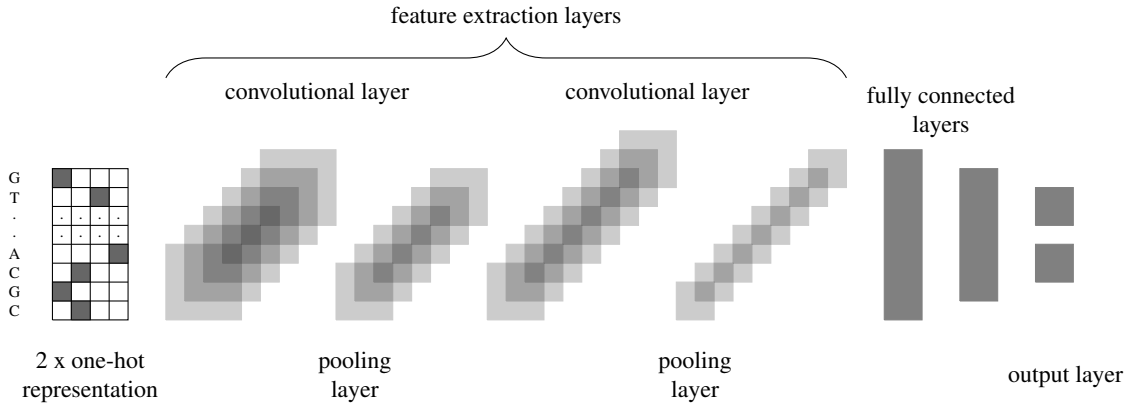


Fig. 4 A schematic sketch of a convolutional neural network that was we used. This basic structure was modified and optimized for training over our different datasets.

Robustness analysis

As mentioned above, real world data is not expected to be noise free. On the contrary, to begin with not all gene interactions is coded into the regions with consider in our training. Moreover, there are many other unconsidered factors that play a role in the process of regulating expression

of genes. On top of that, our method of using a cut-off threshold for classification of co-expressed genes adds an inherent noise. Therefore, we also explore and assess the robustness of the model to noise. This is done with two differently noisified datasets; each constructed using a different type of noise. The first type is a symmetric noise, where we symmetrically swap labels on both directions, with some probability p_{noise}. The other type of noise is false labeling of co-expressed pairs, where we falsely label negatively labeled samples as if they were co-expressed. The latter type probably better represents the type of noise we would expect to encounter in real world datasets.

Uncovering co-expression associated motifs

In case our assumption is correct, and there are indeed co-expression associated motifs in the regions of interest we looked at, and in case we eventually have a neural network that can predict them successfully, the next obvious step is to understand what parts in those regions contributed the most to the predictions, or, uncover the hidden CoAM.

Integrated gradients

For the purpose of identifying the hidden motifs, we use attribution techniques. Those prove to be helpful when there is a need to debug and better understand network prediction, to analyze its robustness, and assess the confidence of predictions. There are many approaches to compute feature contribution, many of them suffer from problems like high computational cost, relying on generated non-realistic input, and failing to properly handle interactions within the feature space. Eventually most produce, in many cases, noisy and non-meaningful output. Integrated-gradients attribution methods introduced by [25](Sundararajan et al. 2017) overcomes some of those problems.

The principle in the base of integrated gradients is using a baseline input x', that could be black image when dealing with images or zero embedding vector for texts. In our case that would be zeros matrix in the dimensions of $(sequencelength \times vocabsize)$. The next step is to construct a straight line path between baseline input x' and input x. Integrated gradients are defined as the path integral of gradients along this constructed straight line path. That is, for the input of the i^{th}

dimension x_i define:

$$\text{IntegratedGradients}_i(x) := (x_i - x_i') \times \int_{\alpha=0}^{1} \frac{\partial F(x' + \alpha \times (x - x'))}{\partial x_i} d\alpha$$

Where F is the function computed by the network. Eventually we'd expect the output of the attribution process to assign for each nucleotide in our input x, a value that indicates, in both magnitude and sign, its negative or positive contribution to the predicted label.

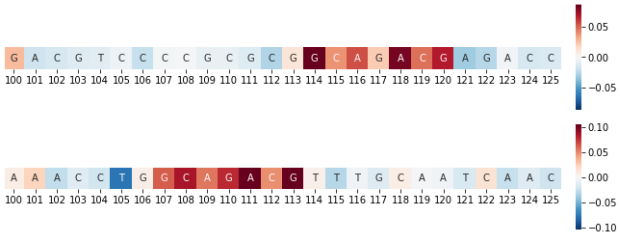


Fig. 5 Section of integrated-gradients output (26 bp, from 100 to 125) on two promoter region sequences input. In this case input was predicted as co-expressed (label=1) by the network, and attribution was for label=1. The suspected CoAM (GGCAGACG) in the input sequence is "hot" and it is clearly within the region of interest detected by the IG.

Extract CoAM from regions of interest

The prediction attribution process is designed to support the detection of parts of the input that most contributed to the actual prediction. This is by no means an accurate function from predictions to the underlying motifs. Rather than that, it can help us in isolating regions of interest, possibly multiple regions per instance; those instances can then be statistically analyzed to extract potential motif candidates.

We defined those regions of interest R_i to be the $L_i^{jk}, k < K, j \in \{1, 2\}$ neighborhood of the top significant K inputs from attribution process applied for positive co-expression class decision:

$$R_i = \left\{ \text{MergeIntersecting}\left(L_i^{jk}\right) : k < K, j \in \{1, 2\} \right\}$$

$$L_i^{jk} = \left\{ x_i^j[t_k - l : t_k + l] : t_k \in \mathsf{topK}, j \in \{1, 2\} \right\}$$

We define the L to be $L = \{L_1, L_2, \ldots, L_S\}$ where $n_1 < n_2 \iff L_{n_1} = L_i^{jk}, L_{n_2} = L_l^{uv}$

and $P_{CoE}(x_i^j, x_i^{1-j}) > P_{CoE}(x_l^u, x_l^{1-u})$. Which is in fact a list of L_i^{jk} sequences sorted in a descending order, by how confident was the network in predicting co-expression for the input instance s_i with the source genomic sequence x_i^j. We feed L into DRIMust[16] which takes as input a ranked list of sequences and returns motifs that are over-represented at the top of the list, where the determination of the threshold that defines "top" is data driven. The search is based on minimum-hypergeometric (mHG). From our perspective, its output is the list of CoAM candidates.

Complete process pipeline overview

To wrap it all together, the complete pipeline for CoAMs extraction is shown and described in Fig. 6. The input to this pipeline is the produced dataset we constructed using expression data (for labeling), complete genomic sequence of the relevant organism including annotations (for input sequences extraction, i.e. promoters, 3'UTRs) and TF regulations vectors constructed from a library of known TFs per gene.

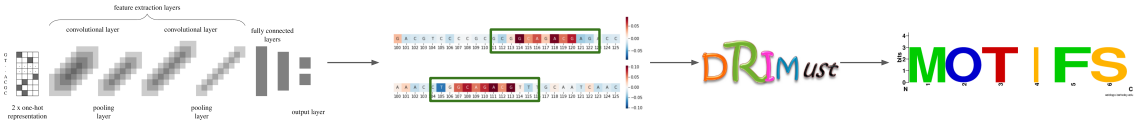


Fig. 6 Overview of the proposed pipeline (left to right). Training a CNN to successfully classify co-expressed pairs of genes; given a successfully trained CNN apply integrated gradient to attribute positive co-expression to parts of the input sequences; rank attributed sequences by the confidence of the predictions; feed the ranked list to DRIMust; DRIMust's output is a set of possible CoAMs, which are enriched motifs and a list of ranked k-mers by enrichment.

3 Results

Co-expression prediction in synthetic data

Our first stop was to verify that our network architecture is capable of learning to correctly classify co-expression as well as handling noise, while we use synthetic data modified promoters as input. For that purpose, from GSE66360 human expression dataset, we generated a synthetic co-expression data set, based on the calculated co-expression network (see **Methods**). We set a co-expression threshold for labeling the data such that:

$$\text{Label}(\text{CoE}(g_i, g_j)) = \begin{cases} 1, & \text{if CoE}(g_i, g_j) > 0.6 \\ 0, & \text{otherwise} \end{cases}$$

At first, as expected, our data was extremely imbalanced, with training size of 6.5M pairs of genes, out of which 4% are labeled as 1 (i.e. CoE > 0.6), our test and validation sets are of size of 300K with similar skew in class balance.

To overcome class balance skew, we downsampled the data; this was done while keeping all examples labeled as 1, and uniformly sampling 0-labeled examples to match 1-labeled set. Eventually we ended up with 500K pairs in a balanced train set, and 25K in each of the validation/test sets.

Once we had the balanced and labeled dataset, we applied the synthetic co-expression data generation process as illustrated in Fig. 2; during this process it turned out that the network of co-expression is highly connected. In this highly connected network, when picking a CoAM to use, an attempt to re-use an already existing motif in one promoter would eventually end up with a single CoAM, that dominates the entire dataset and appears in almost all pairs as the only motif. This is undesirable. On the other hand, random uniform selection of a of motif for each pair would result with most promoter sequences being totally contaminated with all

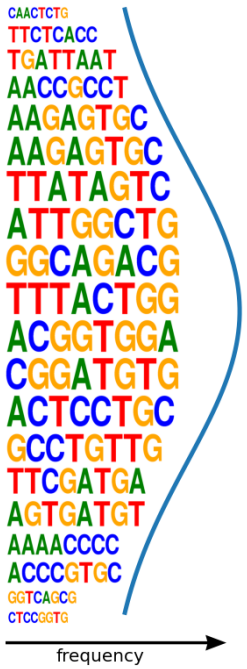


Fig. 7 Motifs occurrence frequency in the synthetic dataset.

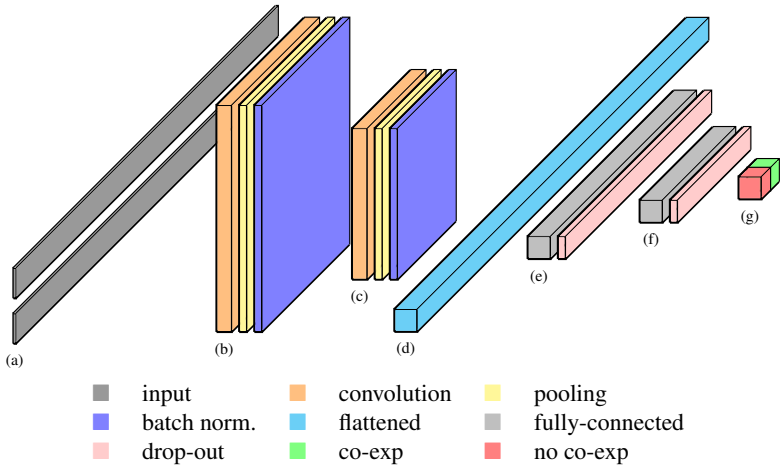


Fig. 8 1D CNN network architecture used for training with synthetic dataset. **(a)** input layer, two one-hot encoded genetic sequences of shape 200 x 4. **(b)** first 1D convolutional layer with 128 filters of size=5 each, followed by a max-pooling layer with filter size=5 and stride=2, followed by a batch-normalization layer. **(c)** second 1D convolutional layer with 256 filters of size=3 each, followed by a max-pooling layer with filter size=3 and stride=2, followed by a batch-normalization layer. **(d)** falttened convolutional output 1D vector (size=23552). **(e)** first fully connected layer with 128 nodes with relu activation, followed by a dropout layer (rate=0.2). **(f)** second fully connected layer with 64 nodes with relu activation, followed by a dropout layer (rate=0.2). **(g)** output layer, 2 nodes for specifying the probability assigned to co-expression. The training loss function used in the training is softmax cross entropy.

the 20 promoters. Therefore, we decided to go for a middle ground, where we normally sample the 20 motifs in such a way that the CoAM frequency is normally distributed as demonstrated in Fig. 7. More precisely, we define a truncated normal distribution over the integer indices values of $idx \in [0, 19]$, centered at the $\mu = 10$ with standard-deviation of $\sigma = 3$ which we use to sample a motif index when selecting a motif to plant. The down side of this approach, is that the such random selection and plating of CoAMs might still yield domination of some connection over others. Namely this might result with rarely occurring CoAMs hidden by more dominant ones.

For training, we used a model with a vanilla 1-D CNN architecture. The architecture in detail is described in Fig. 8. The loss function we used is softmax cross entropy, with mini-batch with size of 32 and learning rate of $2\mathrm{e}{-4}$. The network learning optimizer algorithm used is adaptive moment estimation (aka Adam optimizer)[15]

DL model successfully predicts co-expression and uncovers co-expression associated motifs. The CNN model successfully learns and classifies co-expressed in pairs of genes in the test

set. It pretty quickly converges to performance of accuracy=0.94 with high precision and recall. The performance details are shown in Fig. 10.

Immediately visible is the fact that the synthetics dataset is imbalanced, there are 14k samples labeled as 1 vs. 8k samples labeled as 0; this is a result of labeling examples according to **case III** rule in Fig. 2. Yet, this doesn't seem to have any noticeable effect on the performance of the model.

Given the well trained model, we set to uncover the planted motifs. We applied integrated-gradients to all instances in the evaluation set and computed $\text{IG}_{CoE}(x_i)$, we followed process described in **Methods**, and obtained the regions of interest. After feeding the ranked list of regions of interest into DRIMust, we were able to uncover the 5 most frequent CoAM; this is demonstrated in Fig. 9. Nonetheless, there are still 15 uncovered CoAM. This is probably related to the way that planted the motifs, eventually resulting in multiple occurrences of motifs per promoter, leading the network to prefer the most common motifs over learning much less common motifs. To further validate and assess the results, we performed two additional tasks. The first task was to use the fact that we know which motifs we expect to uncover and validate the output of DRIMust; here we explicitly calculated the enrichment of out synthetic motifs in the ranked list we provided to DRIMust. We discovered that there are 3 significantly enriched motifs that DRIMust omitted despite the fact that they have p-values of: 10^{-12}, 10^{-16}, 10^{-94}. This could be a result of a limitation or an existing issue in DRIMust's algorithm. The second validation was to assess how much of the co-expressed pairs we can attribute to the uncovered 8 motifs (5 from DRIMust + 3 we manually uncovered). We found out that over 95% of the co-expressed pairs can be attributed to the 8 uncovered motifs. This could explain the preference of the CNN to generalize and learn these motifs over the other 12.

Fig. 9 Synthetic motifs uncovered after training by the integrated gradients and DRIMust. Vividly colored motifs are those that were in the output of DRIMust; in green rectangles are motifs that were unexpectedly omitted from DRIMust output despite the fact that they were actually significantly enriched in the top of the ranked list provided to DRIMust. The 8 uncovered motifs can be accounted to more than 95% of the co-expressed pairs of genes in the dataset.

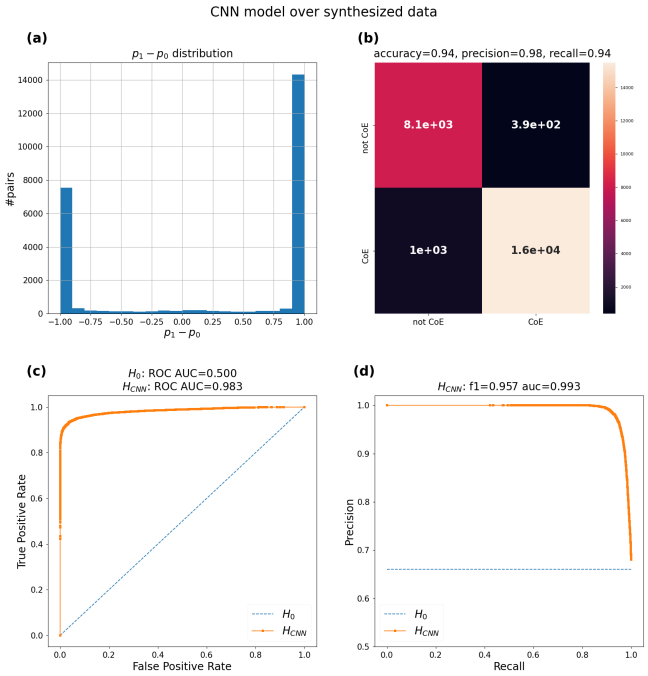


Fig. 10 CNN model synthetic evaluation-set performance, of co-expression inference from promoters with planted motifs. **(a)** distribution of the distance between the confidence in each class. Clearly shows that the network was trained to successfully and confidently predict co-expression. **(b)** confusion matrix of co-expression class prediction. Both here and in predictions distributions it is visible that the synthetic dataset resulted in an imbalanced classes; this is due to labels applied according to case III in Fig. 2 **(c)** ROC curve and AUC. **(d)** PR curve and AUC. Both ROC and PR curves show almost perfect performance over the synthetic dataset.

DL shows robustness to noisy data. At this point we introduced the two noisified datasets with both types noise as described in **Methods**. In each dataset we used an increasing rate p_{noise} of noise, starting from $p_{noise} = 0.05$ up to $p_{noise} = 0.5$. The performance of the model over these datasets is shown in Fig. 11.

In both noisified datasets with both types of noise, it seems that the model was capable of handling a great amount of inconsistency. Looking at the performance of the model trained with the dataset that was symmetrically noisified (inconsistency by symmetric label swapping), it seems that the performance is consistently high until p_{noise} hits 0.3, then it is gradually decreasing until the model is no longer able to predict co-expression at noise rate of 0.5 (complete inconsistency), and performs as good as random choice null model.

Looking at the model that was trained with the datasets noisified with false labeling, the model is

far more robust to the misleading labeling; and is capable of achieving equivalently good results to the model trained with clean dataset, over the eval-set even when almost half of the dataset is noise. At the point of $p_{noise} = 0.5$ the model does not perform as well but it still outperforms the null model.

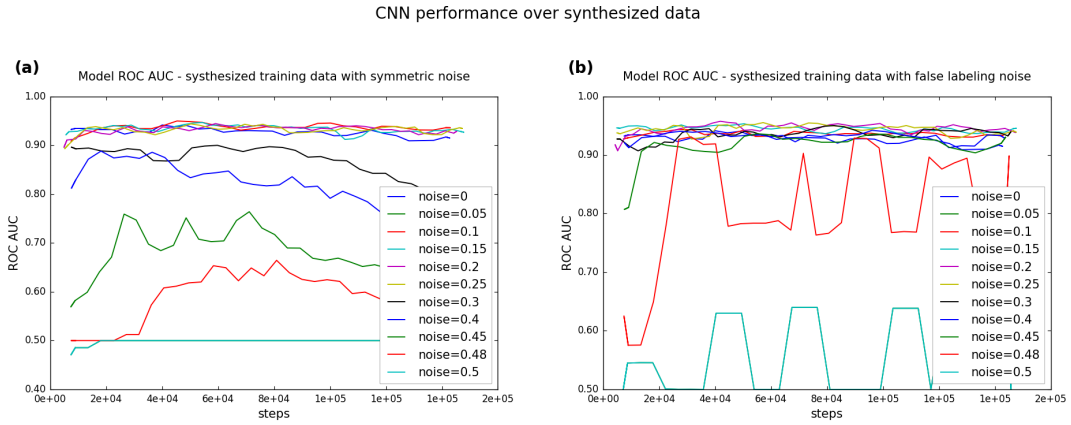


Fig. 11 CNN model performance of co-expression inference after training over noisified synthetic datasets. **(a)** ROC AUC of datasets noisified with inconsistency noise achieved by symmetric labels swapping. **(b)** ROC AUC of datasets noisified with misleading noise achieved by unidirectional false labeling, where negative samples were false labeled as true labels. This was done while keeping the class balancing.

Co-expression predictions in yeast

Here we applied the same principles for predicting gene co-expression using yeast gene expression data from GSE18121.

We used a CNN model with a similar but augmented architecture to the one used with synthetic dataset. In this model, we've doubled the input sequences we feed the network by also providing the network with 3'UTR. Therefore, we augmented the architecture by introducing a new separate convolutional path for the 3'UTRs. Both paths, for promoters and 3'UTRs, share the same structure, but each is trained with a separate set of weights. The structure of the paths itself was updated by adding a third set of convolutional layers, along other changes in layers parameters (see Fig. 13).

As mentioned in **Methods**, we trained two variations of this model, the first variation also takes as input the TF regulation, while the other trains only seeing the promoter and 3'UTR sequences. Our motivation for training a model that trains using genes TFs regulation vector is an attempt to

have a rough headroom assessment of co-expression prediction capability; this makes sense since we know that TFs play a significant role in regulation process. There is a minor structural change in the model that gets the TFs regulation vectors as input; there, we serially concatenated both the regulation to the single high-dimensional flattened vector that follows the convolutional paths, and precedes the fully connected layers. The entire network structure details are provided in Fig. 13

As we could expect the model that got as input the TF regulation vectors significantly outperformed the one that did not. Yet, both were able to learn to some extent to predict when a pair of genes from the test set are co-expressed. The model trained with TF regulation vector was able to achieve almost 70% accuracy; the one trained only on promoters and 3'UTR was able to achieve almost 60% accuracy. The performance details of both models are presented in the FIg. 14.

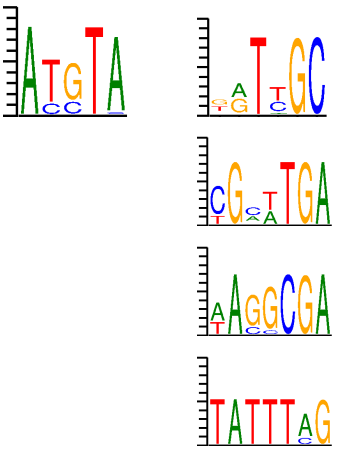


Fig. 12 CoAM uncovered after training on yeast expression data without TF regulation vectors, using integrated gradients and DRIMust. **(left)** 3'UTR regions. **(right)** promoter regions.

For the next step, of searching for CoAM candidates, we used the network trained solely with the promoters and 3'UTRs sequences. The reason for us to prefer using this less performing model is the fact that we wanted to avoid having the model focusing on TFs regulation vector and overlooking the data encoded in the sequences; this is since the predictions attribution to the sequences is the main input to DRIMust. The suggested motifs output from DRIMust is shown in 12.

The full output from DRIMust, for both synthetic and yeast datasets, including motifs occurrences, statistics and the full list of enriched k-mers is provided in the **Supplementary Material** section.

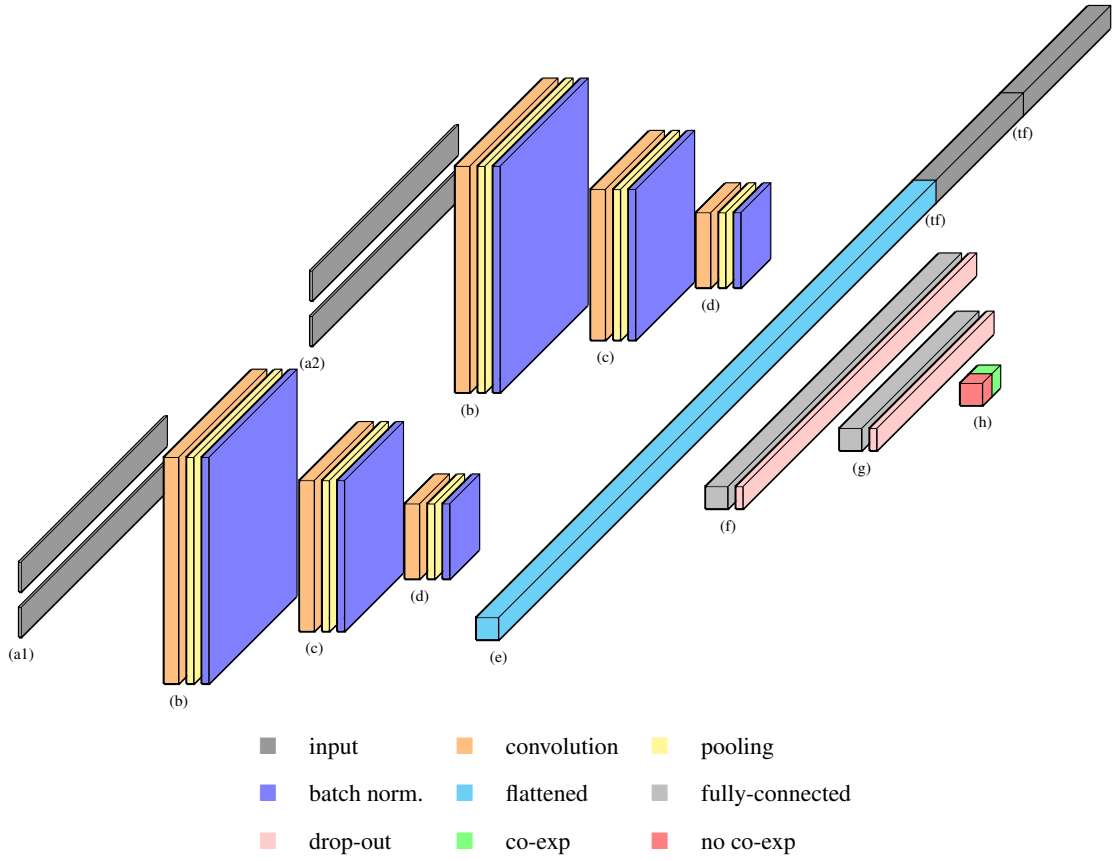
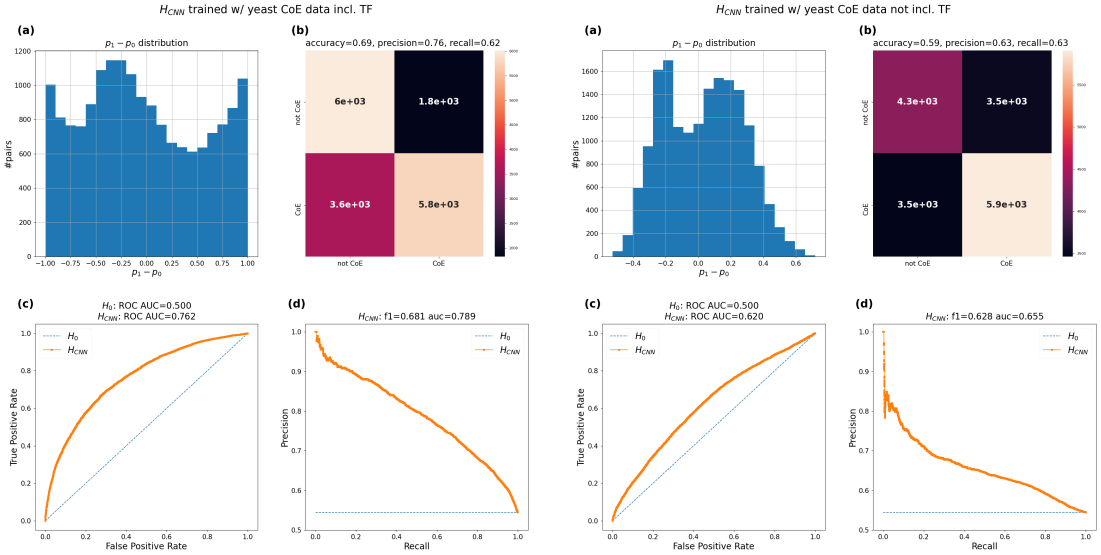


Fig. 13 1D Network network architecture used for training with yeast expression dataset. **(a1)** promoter input layer, two one-hot encoded genetic sequences of shape 1000 x 4. **(a2)** 3'UTR input layer, two one-hot encoded genetic sequences padded when necessary to of shape 1000 x 4. **(b)** first 1D convolutional layers (with distinct set of weights), with 96 filters of size=11 and stride=4 each, followed by max-pooling layer with filter size=6 and stride=4, followed by batch-normalization layer. **(c)** second 1D convolutional layers (with distinct set of weights) with 128 filters of size=3 each, followed by max-pooling layer with filter size=6 and stride=4, followed by batch-normalization layer. **(d)** third 1D convolutional layers (with distinct set of weights) with 256 filters of size=3 each, followed by max-pooling layer with filter size=2 and stride=2, followed by batch-normalization layer. **(e)** falttened convolutional output 1D vector (size=529472). **(tf1,tf2)** transcription factor activation binary vector appended to the falttened convolutional output. **(f)** first fully connected layer with 64 nodes with relu activation, followed by a drop out layer with dropout (rate=0.3). **(g)** second fully connected layer with 12 nodes with linear activation, followed by a drop out layer with dropout (rate=0.3). **(h)** output layer, 2 nodes for specifying the probability assigned to co-expression. The training loss function used in the training is softmax cross entropy.



Results using transcriptioin factors

Results without using transcription factors

Fig. 14 CNN model performance of yeast co-expression predictione for. **(left)** network inputs are promoters, 3'UTR and TF regulation vector. **(right)** network inputs are promomers and 3'UTR only. **(a)** distribution of the distance between the confidence in each class. The difference in the confidence of the predictions is visible between the models; the model that trained with the TF regulation vectors tend to have more confident predictions. **(b)** confusion matrix of co-expression class prediction. **(c)** ROC curve and AUC. Both models show significant capability in predicting co-expression, whit the model trained seeing the TF regulation vectors outperforming the one that didn't, as expected. **(d)** PR curve and AUC. Here we notice that for the two models we can successfully trade-off recall for precision, up to the point that the model trained with TFs regulation vector is able to predict 20% of the pairs with 90% precision; the model trained on the sequences only can predict 10% of the pairs with 80% precision.

4 Discussion

It is interesting to look in to the performance differences between the two models trained on yeast data. As we would expect, the model that was trained with the TF regulation vector performs significantly better; the model was capable of learning the TFs influence on co-expression. One thing that demonstrated this effect is the histograms of $p_1 - p_0$ in Fig. 14 (**a**); those histograms actually reflect certainty of the model in the prediction. The closest the values to the edges $\{-1, 1\}$, the more certain the model is in the prediction. The TFs regulation vector are associated with predictions that are more decisive and accurate.

As we mentioned in the **Introduction**, regulation of gene expression and co-expression is a set of pretty complicated mechanisms. Therefore, it wouldn't surprise us if we wouldn't be able to achieve perfect accuracy. Yet, it is very interesting to change the prediction threshold and examine the precision/recall trade-off. The model trained with the TFs regulation matrix is able to predict 20% of the co-expressed pairs with precision $> 90\%$. The model that was only to train using promoter regions and 3'UTR is not as good, yet, it is capable of predicting 10% of the co-expressed pairs with precision of 80%.

Looking at the artifacts of the next phase of our process, the CoAM candidates yields very interesting results. DRIMust emitted a single CoAM candidate in the 3'UTR region: ATGTA with $p_{value} = 1.5E - 7$. This motif was observed by Aranda Proudfoot (1999)[2], as transcriptional pause element in yeast.

For the k-mers that DRIMust found enriched in the ranked integrated-gradient output, we looked for matching TFs. We first extracted from yeast database the full list of transcription factors and their binding site per-gene; with this list extracted we matched all the k-mers that had a no-miss hit with any TF binding site. The list of those motifs is shown in Tab. 1

The entire lists of k-mers emitted by DRIMust for both 3'UTRs and promoter regions are available in **Supplementary material** section.

There are still unanswered questions and directions of improving of our statements in this paper, which leave room for future related work. A further exploration of the performance of deep learning models in this context can be conducted with more variations of synthetic data; for instance, the dataset can be constructed with the number of planted motifs being increased by one or more order of magnitude. Up to the point that we can assess if the model is capable of detecting general common motifs (single occurrence motifs in the dataset). Maybe the most interesting new path to take would be to apply this pipeline on other organisms. Humans would be the most challenging, since the regulation mechanisms get more complex and include new entities such as miRNA molecules.

Finally, in this paper we perform binary classification of gene co-expression, we could generalize our prediction task to include directionality, and even try training a regression model.

p-value	motif	reverse complement	TF / gene	sites
6.6E-5	GTATTTAG	CTAAATAC	ABF1 / SPR3	CGTATTTAGTGAT
5.8E-4	CGCGTG	CACGCG	PDR1;PDR3 / YOR1	CGCGTGGTTCCGTGGAAAT
7.3E-4	TATTTAG	CTAAATA	ABF1 / SPR3	CGTATTTAGTGAT
2.0E-3	GACGGAT	ATCCGTC	CSRE / FBP1	CGGACGGATGGA
2.0E-3	GACGGAT	ATCCGTC	CSRE / FBP1	TCCGGACGGATGG
2.0E-3	GGTGAC	GTCACC	REB1 / ACT1	CTGTCACCCGGCC
2.0E-3	GGTGAC	GTCACC	UASH / UME6	TCGTCTGAGGTGACA
3.0E-3	GCTTG	CAAGC	RAP1 / TEF1	AACACCCAAGCACAG
3.0E-3	GGTTGC	GCAACC	ARC / ARG1	TCGCAACCTATTTCCATTAACGG
3.0E-3	GTTGCA	TGCAAC	BAS1 / ADE2	GACAAATGACTCTTGTTGCATG
3.0E-3	GTTGC	GCAAC	ARC / ARG1	TCGCAACCTATTTCCATTAACGG
3.0E-3	GTTGC	GCAAC	BAS1 / ADE2	GACAAATGACTCTTGTTGCATG
3.0E-3	GTTGC	GCAAC	DAL82 / DAL7	GCTGAAAGTTGCGGTGCGATAGA ATACCGCGGATTTTGGAA
3.0E-3	GTTGC	GCAAC	UIS / DAL7	GAAAGTTGCGGTG
3.0E-3	TAGGTTG	CAACCTA	ARC / ARG1TTTA	TCGCAACCTATTTCCATTAACGG
4.0E-3	GAGTCC	GGACTC	URSSGA / SGA1	GGGACTCAGGCACAGAAGCAAGGG TCCTTTTTTGGTTCCCTGTTTCCTC
4.0E-3	GTCTGA	TCAGAC	UASH / UME6	TCGTCTGAGGTGACA
4.0E-3	TAAGGA	TCCTTA	MCM1 / FAR1	TTTCCAAGTAAGGAAA
4.0E-3	TAAGGA	TCCTTA	MCM1 / CDC47	TTTCCTTATAAGGAAA
4.0E-3	TAAGGA	TCCTTA	MCM1 / CDC47	TTTCCTTATAAGGAAA

Tab. 1 List of k-mers (CoAM candidates) produced by of DRIMust for yeast promoters regions, with no-miss matching TF/gene binding sites.

5 Supplementary Material

k-mers output of DRIMust for yeast promoters

kmer	pvalue	N	B	n	b	enrichment
TGCTTGA	1.8E-5	4526	6	25	3	90.52
TGTATTTAG	6.6E-5	4526	8	25	3	67.89
GTATTTAG	6.6E-5	4526	8	25	3	67.89
AAGGCGA	2.0E-4	4526	14	197	6	9.85
GCTTGA	5.4E-4	4526	24	33	4	22.86
CGCGTG	5.8E-4	4526	6	12	2	125.72
TTAGGTTGCTTG	7.0E-4	4526	4	25	2	90.52
TGTATTTAGGTT	7.0E-4	4526	4	25	2	90.52
GTATTTAGGTTG	7.0E-4	4526	4	25	2	90.52
AATGTATTTAGG	7.0E-4	4526	4	25	2	90.52
ATTTAGGTTGCT	7.0E-4	4526	4	25	2	90.52
TTTAGGTTGCTT	7.0E-4	4526	4	25	2	90.52
ATGTATTTAGGT	7.0E-4	4526	4	25	2	90.52
TAGGTTGCTTGA	7.0E-4	4526	4	25	2	90.52
TATTTAGGTTGC	7.0E-4	4526	4	25	2	90.52
TATTTAG	7.3E-4	4526	14	25	3	38.79
TCGCTTG	7.7E-4	4526	12	183	5	10.31
GTTGCTT	8.8E-4	4526	16	63	4	17.96
TATGGTCTGATA	9.4E-4	4526	4	29	2	78.03
AATATATGGTCT	9.4E-4	4526	4	29	2	78.03
ATGGTCTGATAT	9.4E-4	4526	4	29	2	78.03
TGGTCTGATATA	9.4E-4	4526	4	29	2	78.03
TATATGGTCTGA	9.4E-4	4526	4	29	2	78.03
GTCTGATATAAA	9.4E-4	4526	4	29	2	78.03
ATATATGGTCTG	9.4E-4	4526	4	29	2	78.03
GGTCTGATATAA	9.4E-4	4526	4	29	2	78.03

kmer	pvalue	N	B	n	b	enrichment
ATATGGTCTGAT	9.4E-4	4526	4	29	2	78.03
TCTGATATAAA	9.4E-4	4526	4	29	2	78.03
CTGATATAAA	9.4E-4	4526	4	29	2	78.03
TGATATAAA	9.4E-4	4526	4	29	2	78.03
GATCGCTTG	0.001	4526	8	183	4	12.37
ATCGCTTG	0.001	4526	8	183	4	12.37
GCTTGAA	0.001	4526	8	183	4	12.37
AGGCGAA	0.001	4526	8	183	4	12.37
GCGAAGA	0.001	4526	8	183	4	12.37
AGGCGA	0.002	4526	30	214	8	5.64
CCAAGGAACGCG	0.002	4526	2	2	1	1131.50
CTCCAAGGAACG	0.002	4526	2	2	1	1131.50
TCCAAGGAACGC	0.002	4526	2	2	1	1131.50
CAAGGAACGCGT	0.002	4526	2	2	1	1131.50
AAAACTCCAAGG	0.002	4526	2	2	1	1131.50
AAACTCCAAGGA	0.002	4526	2	2	1	1131.50
AACTCCAAGGAA	0.002	4526	2	2	1	1131.50
ACTCCAAGGAAC	0.002	4526	2	2	1	1131.50
AAGGAACGCGTG	0.002	4526	2	2	1	1131.50
TGTAGC	0.002	4526	8	14	2	80.82
GACGGAT	0.002	4526	4	45	2	50.29
GGTGAC	0.002	4526	6	24	2	62.86
GTTGC	0.003	4526	70	229	13	3.67
TATTTAGGT	0.003	4526	6	25	2	60.35
TTTAGGTT	0.003	4526	6	25	2	60.35
ATTTAGGT	0.003	4526	6	25	2	60.35
TAGGTTG	0.003	4526	6	25	2	60.35
GGTTGC	0.003	4526	6	25	2	60.35
TCTGTTGCAGCT	0.003	4526	2	3	1	754.33
TGAATTCTGTTG	0.003	4526	2	3	1	754.33
GTTGCAGCTGAC	0.003	4526	2	3	1	754.33
ATTCTGTTGCAG	0.003	4526	2	3	1	754.33
TTCTGTTGCAGC	0.003	4526	2	3	1	754.33
TGTTGCAGCTGA	0.003	4526	2	3	1	754.33
CTGTTGCAGCTG	0.003	4526	2	3	1	754.33
AATTCTGTTGCA	0.003	4526	2	3	1	754.33
GAATTCTGTTGC	0.003	4526	2	3	1	754.33
GTTGCA	0.003	4526	26	204	7	5.97
GCTTG	0.003	4526	88	33	6	9.35
TTAGGTT	0.003	4526	10	164	4	11.04
CGTGTGT	0.003	4526	6	139	3	16.28
TGTTTTCCTGAA	0.003	4526	4	53	2	42.70
GAGTGTTTTCCT	0.003	4526	4	53	2	42.70
AAGAGTGTTTTC	0.003	4526	4	53	2	42.70
AGAGTGTTTTCC	0.003	4526	4	53	2	42.70
GTGTTTTCCTGA	0.003	4526	4	53	2	42.70
AGTGTTTTCCTG	0.003	4526	4	53	2	42.70
AAAGAGTGTTTT	0.003	4526	4	53	2	42.70
AATATATGG	0.004	4526	6	29	2	52.02
GATATAAA	0.004	4526	6	29	2	52.02
ATATATGG	0.004	4526	6	29	2	52.02
ATGGTCT	0.004	4526	6	29	2	52.02
TATATGG	0.004	4526	6	29	2	52.02
TGATATA	0.004	4526	6	29	2	52.02
TCTGATA	0.004	4526	6	29	2	52.02
GTCTGA	0.004	4526	6	29	2	52.02
AGTTTTTGGGTT	0.004	4526	2	4	1	565.75
TTGGGTTTGTAT	0.004	4526	2	4	1	565.75
TGGGTTTGTATA	0.004	4526	2	4	1	565.75
GGGTTTGTATAA	0.004	4526	2	4	1	565.75

kmer	pvalue	N	B	n	b	enrichment
TTTTGGGTTTGT	0.004	4526	2	4	1	565.75
TTTGGGTTTGTA	0.004	4526	2	4	1	565.75
GTTTTTGGGTTT	0.004	4526	2	4	1	565.75
TTTTTGGGTTTG	0.004	4526	2	4	1	565.75
GGTTTGTATAAT	0.004	4526	2	4	1	565.75
GAGTCC	0.004	4526	12	408	6	5.55
GCGAAG	0.004	4526	10	68	3	19.97
TAAGGA	0.004	4526	28	198	7	5.71
ATTGCATTCCAA	0.004	4526	12	56	3	20.21
GCATTCCAAAAA	0.004	4526	12	56	3	20.21
TGCATTCCAAAA	0.004	4526	12	56	3	20.21
CATTCCAAAAAT	0.004	4526	12	56	3	20.21
TCCAAAAATAA	0.004	4526	12	56	3	20.21
CCAAAAATAA	0.004	4526	12	56	3	20.21

k-mers output of DRIMust for yeast 3'UTRs

kmer	pvalue	N	B	n	b	enrichment
AACGTAAGAAAC	3.8E-4	1632	4	7	2	116.57
GTAAGAAACTAA	3.8E-4	1632	4	7	2	116.57
AAAACGTAAGAA	3.8E-4	1632	4	7	2	116.57
AAACGTAAGAAA	3.8E-4	1632	4	7	2	116.57
AGCTAAAACGTA	3.8E-4	1632	4	7	2	116.57
ACGTAAGAAACT	3.8E-4	1632	4	7	2	116.57
AAGAAACTAAGG	3.8E-4	1632	4	7	2	116.57
CGTAAGAAACTA	3.8E-4	1632	4	7	2	116.57
GCTAAAACGTAA	3.8E-4	1632	4	7	2	116.57
TAAGAAACTAAG	3.8E-4	1632	4	7	2	116.57
CTAAAACGTAAG	3.8E-4	1632	4	7	2	116.57
TAAAACGTAAGA	3.8E-4	1632	4	7	2	116.57
GAAAATTAT	4.7E-4	1632	6	27	3	30.22
AAAATTAT	4.7E-4	1632	6	27	3	30.22
TATGTAC	4.7E-4	1632	6	27	3	30.22
ATGTA	7.7E-4	1632	94	48	12	4.34
AGTCAT	10.0E-4	1632	20	60	6	8.16
GAGCTAAAACGT	0.001	1632	6	7	2	77.71
TGATTAGTGTTA	0.001	1632	6	7	2	77.71
ATTAGTGTTAGA	0.001	1632	6	7	2	77.71
TTTTGATTAGTG	0.001	1632	6	7	2	77.71
AGTGTTAGAGCT	0.001	1632	6	7	2	77.71
GTTAGAGCTAAA	0.001	1632	6	7	2	77.71
GTGTTAGAGCTA	0.001	1632	6	7	2	77.71
TTTGATTAGTGT	0.001	1632	6	7	2	77.71
GATTAGTGTTAG	0.001	1632	6	7	2	77.71
TTGATTAGTGTT	0.001	1632	6	7	2	77.71
TAGTGTTAGAGC	0.001	1632	6	7	2	77.71
AGAGCTAAAACG	0.001	1632	6	7	2	77.71
TGTTAGAGCTAA	0.001	1632	6	7	2	77.71
TTAGAGCTAAAA	0.001	1632	6	7	2	77.71
AATTTTGATTAG	0.001	1632	6	7	2	77.71
TAGAGCTAAAAC	0.001	1632	6	7	2	77.71
ATTTTGATTAGT	0.001	1632	6	7	2	77.71
TTAGTGTTAGAG	0.001	1632	6	7	2	77.71
AGCTAAAACGT	0.001	1632	6	7	2	77.71
GCTAAAACGT	0.001	1632	6	7	2	77.71
CTAAAACGT	0.001	1632	6	7	2	77.71
GTAAGAAA	0.001	1632	6	7	2	77.71
TAAAACGT	0.001	1632	6	7	2	77.71
GAAACTA	0.001	1632	6	7	2	77.71

kmer	pvalue	N	B	n	b	enrichment
AAACGTA	0.001	1632	6	7	2	77.71
AAACTAA	0.001	1632	6	7	2	77.71
AACGTA	0.001	1632	6	7	2	77.71
GAGAAA	0.002	1632	16	27	4	15.11
GAGAAAAT	0.002	1632	8	27	3	22.67
AAATTAT	0.002	1632	8	27	3	22.67
AGAGAAA	0.002	1632	8	27	3	22.67
AGAGAA	0.002	1632	12	41	4	13.27
GAAAAGA	0.002	1632	26	6	3	31.38
TCACTA	0.002	1632	10	22	3	22.25
AGGATGC	0.002	1632	4	17	2	48.00
GGATGC	0.002	1632	4	17	2	48.00
AAAAGAGTGGAT	0.002	1632	8	6	2	68.00
AAGAAAAGAGTG	0.002	1632	8	6	2	68.00
AGAGTGGATGTA	0.002	1632	8	6	2	68.00
AGAAAAGAGTGG	0.002	1632	8	6	2	68.00
AACAAGAAAAGA	0.002	1632	8	6	2	68.00
GATGTAGCAACT	0.002	1632	8	6	2	68.00
AAGAGTGGATGT	0.002	1632	8	6	2	68.00
ATGTAGCAACTG	0.002	1632	8	6	2	68.00
AGTGGATGTAGC	0.002	1632	8	6	2	68.00
ACAAGAAAAGAG	0.002	1632	8	6	2	68.00
GAAAAGAGTGGA	0.002	1632	8	6	2	68.00
GAGTGGATGTAG	0.002	1632	8	6	2	68.00
TGGATGTAGCAA	0.002	1632	8	6	2	68.00
AAAGAGTGGATG	0.002	1632	8	6	2	68.00
GTGGATGTAGCA	0.002	1632	8	6	2	68.00
GGATGTAGCAAC	0.002	1632	8	6	2	68.00
TAACAAGAAAAG	0.002	1632	8	6	2	68.00
CAAGAAAAGAGT	0.002	1632	8	6	2	68.00
TGTAGCAACTG	0.002	1632	8	6	2	68.00
GTAGCAACTG	0.002	1632	8	6	2	68.00
TAGCAACTG	0.002	1632	8	6	2	68.00
AGCAACTG	0.002	1632	8	6	2	68.00
AAGAG	0.003	1632	66	44	9	5.06
GTAAAAAAA	0.003	1632	6	50	3	16.32
AAGAAA	0.003	1632	86	9	5	10.54
TAAGAAACT	0.003	1632	8	7	2	58.29
CGTAAGAA	0.003	1632	8	7	2	58.29
AAGAAACT	0.003	1632	8	7	2	58.29
ACGTAA	0.003	1632	8	7	2	58.29
ATTAGT	0.003	1632	8	7	2	58.29
TTAGTG	0.003	1632	8	7	2	58.29
TGGATCAT	0.004	1632	6	54	3	15.11
GGATCAT	0.004	1632	6	54	3	15.11
ACTCGGGAAATA	0.004	1632	4	22	2	37.09
TTAACTCGGGAA	0.004	1632	4	22	2	37.09
TAACTCGGGAAA	0.004	1632	4	22	2	37.09
CTCGGGAAATAT	0.004	1632	4	22	2	37.09
TCGGGAAATATG	0.004	1632	4	22	2	37.09
TATGTATCACTA	0.004	1632	4	22	2	37.09
GAAATATGTATC	0.004	1632	4	22	2	37.09
TTTTAACTCGGG	0.004	1632	4	22	2	37.09
CGGGAAATATGT	0.004	1632	4	22	2	37.09
AACTCGGGAAAT	0.004	1632	4	22	2	37.09
ATGTATCACTA	0.004	1632	4	22	2	37.09
TGTATCACTA	0.004	1632	4	22	2	37.09
GTATCACTA	0.004	1632	4	22	2	37.09
TATCACTA	0.004	1632	4	22	2	37.09

References

[1] Babak Alipanahi et al. “Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning”. In: *Nature Biotechnology* 33.8 (July 2015), pp. 831–838. DOI: 10.1038/nbt.3300. URL: https://doi.org/10.1038/nbt.3300.

[2] Agusti
n Aranda and Nick J. Proudfoot. “Definition of Transcriptional Pause Elements in Fission Yeast”. In: *Molecular and Cellular Biology* 19.2 (Feb. 1999), pp. 1251–1261. DOI: 10.1128/mcb.19.2.1251. URL: https://doi.org/10.1128/mcb.19.2.1251.

[3] Adrian P. Bird. “CpG-rich islands and the function of DNA methylation”. In: *Nature* 321.6067 (May 1986), pp. 209–213. DOI: 10.1038/321209a0. URL: https://doi.org/10.1038/321209a0.

[4] Yifei Chen et al. “Gene expression inference with deep learning”. In: *Bioinformatics* 32.12 (Feb. 2016), pp. 1832–1839. DOI: 10.1093/bioinformatics/btw074. URL: https://doi.org/10.1093/bioinformatics/btw074.

[5] J. M. Cherry et al. “Saccharomyces Genome Database: the genomics resource of budding yeast”. In: *Nucleic Acids Research* 40.D1 (Nov. 2011), pp. D700–D705. DOI: 10.1093/nar/gkr1029. URL: https://doi.org/10.1093/nar/gkr1029.

[6] L Ashley Cowart et al. “Revealing a signaling role of phytosphingosine-1-phosphate in yeast”. In: *Molecular Systems Biology* 6.1 (Jan. 2010), p. 349. DOI: 10.1038/msb.2010.3. URL: https://doi.org/10.1038/msb.2010.3.

[7] Sipko van Dam et al. “Gene co-expression analysis for functional classification and gene–disease predictions”. In: *Briefings in Bioinformatics* 19.4 (Jan. 2017), pp. 575–592. ISSN: 1477-4054. DOI: 10.1093/bib/bbw139. eprint: https://academic.oup.com/bib/article-pdf/19/4/575/25193126/bbw139.pdf. URL: https://doi.org/10.1093/bib/bbw139.

[8] Eran Eden et al. “Discovering Motifs in Ranked Lists of DNA Sequences”. In: *PLoS Computational Biology* 3.3 (Mar. 2007). Ed. by Ernest Fraenkel, e39. DOI: 10.1371/journal.pcbi.0030039. URL: https://doi.org/10.1371/journal.pcbi.0030039.

[9] Stacia R. Engel et al. “The Reference Genome Sequence ofSaccharomyces cerevisiae: Then and Now”. In: *G3: Genes — Genomes — Genetics* 4.3 (Dec. 2013), pp. 389–398. DOI: 10.1534/g3.113.008995. URL: https://doi.org/10.1534/g3.113.008995.

[10] Maximilian Haeussler et al. “The UCSC Genome Browser database: 2019 update”. In: *Nucleic Acids Research* 47.D1 (Nov. 2018), pp. D853–D858. DOI: 10.1093/nar/gky1095. URL: https://doi.org/10.1093/nar/gky1095.

[11] “Initial sequencing and analysis of the human genome”. In: *Nature* 409.6822 (Feb. 2001), pp. 860–921. DOI: 10.1038/35057062. URL: https://doi.org/10.1038/35057062.

[12] Rudolf Jaenisch and Adrian Bird. “Epigenetic regulation of gene expression: how the genome integrates intrinsic and environmental signals”. In: *Nature Genetics* 33.S3 (Mar. 2003), pp. 245–254. DOI: 10.1038/ng1089. URL: https://doi.org/10.1038/ng1089.

[13] Steven Kearnes et al. “Molecular graph convolutions: moving beyond fingerprints”. In: *Journal of Computer-Aided Molecular Design* 30.8 (Aug. 2016), pp. 595–608. DOI: 10.1007/s10822-016-9938-8. URL: https://doi.org/10.1007/s10822-016-9938-8.

[14] W. J. Kent et al. “The Human Genome Browser at UCSC”. In: *Genome Research* 12.6 (May 2002), pp. 996–1006. DOI: 10.1101/gr.229102. URL: https://doi.org/10.1101/gr.229102.

[15] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2014. eprint: arXiv:1412.6980.

[16] Limor Leibovich et al. “DRIMust: a web server for discovering rank imbalanced motifs using suffix trees”. In: *Nucleic Acids Research* 41.W1 (May 2013), W174–W179. DOI: 10.1093/nar/gkt407. URL: https://doi.org/10.1093/nar/gkt407.

[17] Alona Levy-Jurgenson et al. “Predicting Methylation from Sequence and Gene Expression Using Deep Learning with Attention”. In: (Dec. 2018). DOI: 10.1101/491357. URL: https://doi.org/10.1101/491357.

[18] Alona Levy-Jurgenson et al. “Predicting Methylation from Sequence and Gene Expression Using Deep Learning with Attention”. In: *Algorithms for Computational Biology*. Springer International Publishing, 2019, pp. 179–190. DOI: 10.1007/978-3-030-18174-1_13. URL: https://doi.org/10.1007/978-3-030-18174-1_13.

[19] Alona Levy-Jurgenson et al. “Spatial transcriptomics inferred from pathology whole-slide images links tumor heterogeneity to survival in breast and lung cancer”. In: *Scientific Reports* 10.1 (Nov. 2020). DOI: 10.1038/s41598-020-75708-z. URL: https://doi.org/10.1038/s41598-020-75708-z.

[20] Jiecong Lin and Ka-Chun Wong. “Off-target predictions in CRISPR-Cas9 gene editing using deep learning”. In: *Bioinformatics* 34.17 (Sept. 2018), pp. i656–i663. DOI: 10.1093/bioinformatics/bty554. URL: https://doi.org/10.1093/bioinformatics/bty554.

[21] Barsanjit Mazumder, Vasudevan Seshadri, and Paul L Fox. “Translational control by the 3′-UTR: the ends specify the means”. In: *Trends in Biochemical Sciences* 28.2 (Feb. 2003), pp. 91–98. DOI: 10.1016/s0968-0004(03)00002-1. URL: https://doi.org/10.1016/s0968-0004(03)00002-1.

[22] Pedro T Monteiro et al. “YEASTRACT: a portal for cross-species comparative genomics of transcription regulation in yeasts”. In: *Nucleic Acids Research* 48.D1 (Oct. 2019), pp. D642–D649. DOI: 10.1093/nar/gkz859. URL: https://doi.org/10.1093/nar/gkz859.

[23] Evan D. Muse et al. “A Whole Blood Molecular Signature for Acute Myocardial Infarction”. In: *Scientific Reports* 7.1 (Sept. 2017). DOI: 10.1038/s41598-017-12166-0. URL: https://doi.org/10.1038/s41598-017-12166-0.

[24] Joshua M. Stuart et al. “A Gene-Coexpression Network for Global Discovery of Conserved Genetic Modules”. In: *Science* 302.5643 (2003), pp. 249–255. ISSN: 0036-8075. DOI: 10.1126/science.1087447. eprint: https://science.sciencemag.org/content/302/5643/249.full.pdf. URL: https://science.sciencemag.org/content/302/5643/249.

[25] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. “Axiomatic Attribution for Deep Networks”. In: *CoRR* abs/1703.01365 (2017). arXiv: 1703.01365. URL: http://arxiv.org/abs/1703.01365.

[26] Shinya Tasaki et al. “Deep learning decodes the principles of differential gene expression”. In: *Nature Machine Intelligence* 2.7 (2020), pp. 376–386. DOI: 10.1038/s42256-020-0201-6. URL: https://doi.org/10.1038%2Fs42256-020-0201-6.

[27] Bin Zhang and Steve Horvath. “A General Framework for Weighted Gene Co-Expression Network Analysis”. In: *Statistical Applications in Genetics and Molecular Biology* 4.1 (12 Aug. 2005). DOI: https://doi.org/10.2202/1544-6115.1128. URL: https://www.degruyter.com/view/journals/sagmb/4/1/article-sagmb.2005.4.1.1128.xml.xml.

המרכז הבינתחומי בהרצליה
בית-ספר אפי ארזי למדעי המחשב
התכנית לתואר שני (M.Sc.) - מסלול מחקרי

למידה עמוקה ורצפים קובעי מִתְאָם של ביטוי גנטי

מאת
שרון מאיר סולטן

עבודת תזה המוגשת כחלק מהדרישות לשם קבלת תואר מוסמך M.Sc.
במסלול המחקרי בבית ספר אפי ארזי למדעי המחשב, המרכז הבינתחומי הרצליה

פברואר 2021

תקציר

מִתְאָם של ביטוי גנטי יכול לשמש כמַחְוָן על-מנת למצוא קשר בין גנים לבין תהליכים ביולוגיים. זאת על-מנת להבין טוב יותר את תפקידם של גנים וכדי לחקור תהליכי בקרה ביצורים חיים. הִתְחַקּוּת אחר הקשרים הללו, והתהליכים שיוצרים אותם, והבנתם, עלולים להיות מורכבים וקשים להֶשֵּׂק. לצד זה, רשתות נוירונים עמוקות הולכות ותופסות תפקיד חשוב בתחום הביולוגיה החישובית. נעשה ברשתות אלו שימוש נרחב בימינו על-מנת ללמוד לגלות, לחזות ולפתור בעיות ביולוגיות מורכבות וקשרים סמויים. במאמר זה אנו מציגים שימוש בכלים של למידה עמוקה כדי לחזות קשרי מתאם של ביטוי גנטי ולפענח את מיקומם של רצפים גנטיים חבויים אשר קובעים קשרים אלו. תחילה אנו חוקרים את יכולות הרשת העמוקה בהקשר של חיזוי וגילוי של מוטיבים חבויים בנתונים סינתטיים, שנבנו מעל רצפים גנומים ממשיים. לאחר מכן, אנו מעמיקים ומיישמים את אותן השיטות על נתונים אמיתיים של מתאם של ביטוי גנטי שנאספו ממדידות בשְׂמָרִים. עולה בידינו להראות כי התהליך המבוסס למידה עמוקה שבנינו, מסוגל להפיק חיזויים בעלי משמעות ולחשוף מוטיבים. מוטיבים אלה יכולים להיות המפתח להבנת שאפשר כי הם שייכים למתאם של ביטוי גנטי (CoAMs = Co-expression Associated Motifs). כמו כן אנו מראים כי חלק מן המוטיבים הללו מוכר והם ממלאים תפקיד ידוע בתהליכי בקרה בשמרים.

עבודה זו בוצעה בהדרכתו של פרופ' **זהר יכיני** מבי"ס אפי ארזי למדעי המחשב, המרכז הבינתחומי, הרצליה.