Deep unfolded convolutional dictionary learning for motif discovery

Shane Chu [*1] and Gary Stormo [†2]

[1]*Department of Computer Science and Engineering, Washing University in St.Louis*
[2]*Department of Genetics, Washington University School of Medicine, St. Louis*

Abstract

We present a principled representation learning approach based on convolutional dictionary learning (CDL) for motif discovery. We unroll an iterative algorithm that optimizes CDL as a forward pass in a neural network, resulting in a network that is fully interpretable, fast, and capable of finding motifs in large datasets. Simulated data show that our network is more sensitive and specific for discovering binding sites that exhibit complex binding patterns than popular motif discovery methods such as STREME and HOMER. Our network reveals statistically significant motifs and their diverse binding modes from the JASPAR database that are currently not reported.

1 Background and motivation

Finding conserved substrings in a set of unaligned DNA strings is a fundamental problem in computational biology. These conserved substrings, called *motifs*, arise from the convergence of evolutionary forces and are marked as molecular switches for regulatory mechanisms. Accurate characterization of motifs reveals binding specificity of proteins, which in turn provides features for downstream regulatory genomics tasks and helps understand how cells carry out gene expressions.

All models devised for motif discovery represent motifs in either a localist representation or a distributed representation [Hin84]. Most combinatorial and probabilistic formulations for motif discovery represent the motif in a localist representation – either as a consensus string, a product-multinomial distribution, or a position weight matrix (PWM).

While a localist representation is easy to interpret, the models that employ it to represent motifs tend to be overly simplistic, e.g., MEME [BE95] formulates motif discovery as a two-component mixture model, for which one component represents the motif, and the other component represents the genomic background. To overcome this over-simplification, e.g., so that MEME can find multiple motifs in the input strings, MEME performs sequential motif discovery: it "masks out" the substrings used by the discovered motifs before finding other motifs. Such an attempt is problematic for motif discovery as there may be more than one motif in the dataset, and motifs can exhibit binding modes such as *multi-meric binding* or *alternative structural conformations* [SG14]. Both STREME and HOMER still employ similar heuristics for motif discovery at the time of writing. The results section 3.3 shows how this heuristic can lead to incomplete motif discovery results.

The most popular model that employs distributed representation for motif discovery at the time of writing is the convolutional neural network (CNN). Although the model is expressive, it is challenging to interpret what represents the motif as the model itself is in a hierarchical distributed representation. Some attempts seek to interpret the first convolution layer of the CNN as motifs, e.g., [KP21; Nov+22], but provide no quantitative justifications for the choice of CNN architectures. Other attempts seek to find interpretations through model

[*]skchu@wustl.edu
[†]stormo@wustl.edu

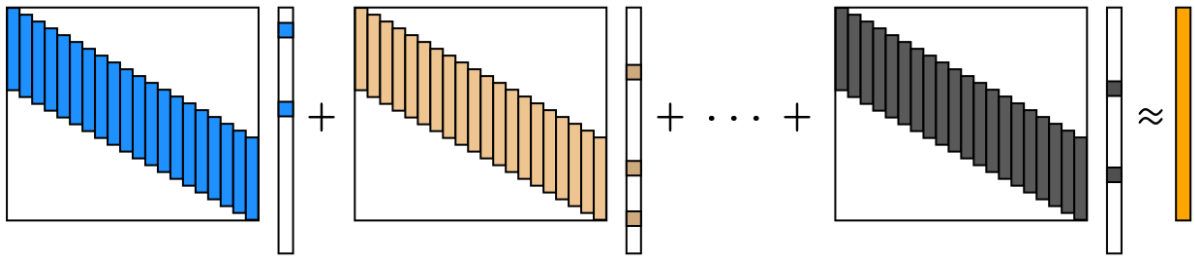

Figure 1: Approximate the signal (in orange) using a sum of convolutions. A convolution $\boldsymbol{f} * \boldsymbol{x}$ is equivalent to $\boldsymbol{F}\boldsymbol{x}$, a Toeplitz matrix $\boldsymbol{F}$ multiply a (sparse) vector $\boldsymbol{x}$.

agnostic explanation methods such as SHAP [LL17]; however, SHAP is computationally expensive when the model contains many features.

To find an expressive, interpretable model that avoids the need for heuristics such as "masks out", we ask the following question: what is the simplest distributed representation on the input strings that we can use to infer the motifs? We find a solution to such a question to be convolutional dictionary learning (CDL) [GW18; Woh17; DI18; HHW15; BEL13]. The basic idea is we approximate each DNA string $\boldsymbol{s}$ as a sum of the convolution of feature vectors and sparse vectors, i.e.,

$$\sum_m \boldsymbol{f}_m * \boldsymbol{x}_m \approx \boldsymbol{s}$$

Conventionally, the feature vectors $\boldsymbol{f}_m$'s are called the filters, and the sparse vectors $\boldsymbol{x}$'s are called the sparse code.

In this work, we present a motif discovery method based on CDL. The interpretation is straightforward: the CDL approximates each input sequence with a sparse linear combination of shift-invariant filters. This built-in interpretation can be efficiently used to infer the enriched patterns in the dataset, i.e., the motifs. Further, CDL allows us to *simultaneously* discover motifs in the input strings as opposed to sequential motif discovery. To scale CDL for large-scale motif discovery, we unfold the iterative procedure that optimizes CDL as a forward pass of a neural network [GL10; MLE21]. This results in a fully interpretable network that learns an approximate sparse representation in a timely manner. Our method discovers zero or more motifs for each sequence in the dataset. Simulated data suggests that our method is more capable of capturing multiple motifs in the dataset, especially when they share similar patterns. Additionally, our method reveals alternative modes of binding that are not reported in the JASPAR database [Cas+22].

2 Methodology

2.1 Notation

The term DNA strings referred to strings defined on the alphabet $\{$ A,C,G,T $\}$. The map $\mathsf{r}(\cdot)$ takes a vector and reshapes it into a matrix following the column-major order. The i-th component of a vector $\boldsymbol{v}$ is denoted as $\boldsymbol{v}[i]$, and the i-th to the j-th consecutive components of $\boldsymbol{v}$ is denoted by $\boldsymbol{v}[i{:}j]$. Similarly, $\boldsymbol{W}[:,i]$ is the i-th column of the matrix $\boldsymbol{W}$. We write $\boldsymbol{x} \geq \boldsymbol{0}$ to indicate that all the components of the vector $\boldsymbol{x}$ are non-negative. Brackets around an integer K, i.e., $[K]$, is the set of integers $\{ 1, \ldots, K \}$. The function $|\cdot|$ takes a finite set and returns its number of elements. The function $\lceil \cdot \rceil$ rounds up a real number. The map $\mathsf{diag}(\cdot)$ returns a diagonal matrix with identical diagonal components from the input matrix. The norms $\|\cdot\|_F, \|\cdot\|_2$, and $\|\cdot\|_1$ are the Frobenius norm, ℓ_2 norm, and ℓ_1 norm, respectively.

For the motif discovery problem, assume we have a set of one-hot encoded DNA strings $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_N$ each consisting of L nucleotides. Each string $\boldsymbol{s}_n$ may contain zero or multiple motifs. We will first formulate

convolutional dictionary learning (CDL) to obtain a sparse representation of the dataset. The sparse representation refers to a dictionary $\boldsymbol{D}$ which consists of filters $\boldsymbol{d}_1, \ldots, \boldsymbol{d}_M$ and a sparse code $\boldsymbol{X}$ of (sparse) code vectors $\boldsymbol{x}_{11}, \ldots, \boldsymbol{x}_{MN}$ (Figure 1 correspond to one DNA string, i.e., $N = 1$). Our goal with CDL is to obtain the sparse representation for which we could infer what and where the motifs are in the strings $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_N$.

2.2 Convolutional dictionary learning

We approximate each DNA sequence $\boldsymbol{s}$ as a sum of convolutions:

$$\boldsymbol{s} \approx \sum_m \boldsymbol{d}_m * \boldsymbol{x}_m$$

where $\boldsymbol{d}_m$'s are the filters, and $\boldsymbol{x}_m$'s are the code vectors. We consider the reshaped filters as position frequency matrices (PFM), i.e., $\mathrm{r}(\boldsymbol{d}_m) \in \mathcal{P}_\ell$, where

$$\mathcal{P}_\ell = \left\{ \boldsymbol{p} \in \mathbb{R}_+^{4\times\ell} : \sum_\alpha \boldsymbol{p}[\alpha, \kappa] = 1,\ \kappa \in [\ell] \right\}.$$

is the set of PFMs with ℓ columns, which avoids the scaling ambiguity between the filters and the sparse code. Now, with DNA strings $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_N$, we formulate the sparse representation discovery for DNA sequences as a convolutional dictionary learning (CDL) problem

$$\underset{\{\,\mathrm{r}(\boldsymbol{d}_m) \in \mathcal{P}_\ell\,\},\{\,\boldsymbol{x}_{mn} \geq \boldsymbol{0}\,\}}{\operatorname{argmin}} \ \frac{1}{2} \sum_n \left\| \sum_m \boldsymbol{d}_m * \boldsymbol{x}_{mn} - \boldsymbol{s}_n \right\|_2^2 + \lambda \sum_{n,m} \|\boldsymbol{x}_{mn}\|_1 \tag{1}$$

for which we can compactly write as

$$\underset{\boldsymbol{D}\in\mathcal{C}_{\boldsymbol{D}},\, \boldsymbol{X}\in\mathcal{C}_{\boldsymbol{X}}}{\operatorname{argmin}} \ \frac{1}{2} \|\boldsymbol{D}\boldsymbol{X} - \boldsymbol{S}\|_F^2 + \lambda \|\boldsymbol{X}\|_1 \tag{2}$$

where matrices $\boldsymbol{D}, \boldsymbol{X}, \boldsymbol{S}$ are the concatenation of all the filters as Toeplitz matrices, all the code vectors, and all the input DNA sequences, respectively. The sets $\mathcal{C}_{\boldsymbol{D}}$ and $\mathcal{C}_{\boldsymbol{X}}$ encode the same constraints as in 1.[1]. The parameter λ controls the trade-off between the sparsity of $\boldsymbol{X}$ and the reconstruction loss.

2.3 Alternating minimization for convolutional dictionary learning

A common strategy to obtain the solution to problem 2 is via alternating minimization [GW18]. The alternating minimization proceeds by iteratively solving the two following convex sub-problems of 2:

$$\underset{\boldsymbol{X}\in\mathcal{C}_{\boldsymbol{X}}}{\operatorname{argmin}} \frac{1}{2} \|\boldsymbol{D}\boldsymbol{X} - \boldsymbol{S}\|_F^2 + \lambda \|\boldsymbol{X}\|_1 \qquad \underset{\boldsymbol{D}\in\mathcal{C}_{\boldsymbol{D}}}{\operatorname{argmin}} \frac{1}{2} \sum_n \left\| \sum_m \boldsymbol{X}_{mn} \boldsymbol{d}_m - \boldsymbol{s}_n \right\|_2^2$$

Here, $\boldsymbol{X}_{mn}$ is the Toeplitz matrix that corresponds to $\boldsymbol{x}_{mn}$. The problem on the left is convolution sparse coding (CSC), and the problem on the right is dictionary update (DU). It is well-known that CSC can be solved by the iterative shrinkage-thresholding algorithm (ISTA). Specifically, the sparse code $\boldsymbol{X}$ at the t-th iteration of ISTA is

$$\boldsymbol{X}^t = \mathcal{T}_{\lambda\eta_t}\left(\boldsymbol{X}^{t-1} + \eta_t \boldsymbol{D}^\top \left(\boldsymbol{S} - \boldsymbol{D}\boldsymbol{X}^{t-1}\right)\right) \tag{3}$$

where η_t is the step-size at the t-th iteration, and $\mathcal{T}_{\lambda\eta_t}$ is the non-negative soft-thresholding operator parameterized by $\lambda\eta_t$. For DU, we consider mirror descent [NY83; BT03]. Let $\boldsymbol{d}_{m\kappa} = \boldsymbol{d}_m[4(\kappa-1)+1{:}4(\kappa-1)+4]$ be

[1] In practice, since we are doing 1D convolutions, we have that $\mathcal{C}_{\boldsymbol{X}} \triangleq \{\boldsymbol{x}_n : \boldsymbol{x}_{mn} \geq \boldsymbol{0},\ \boldsymbol{P}\boldsymbol{x}_{mn} = \boldsymbol{0}\}$, where $\boldsymbol{P}$ is a projector that maps each equally spaced three-consecutive components of the sparse code to zero.

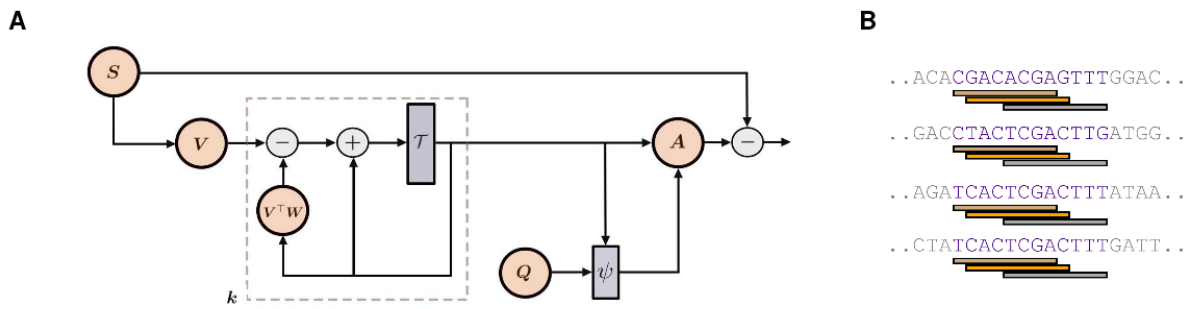

Figure 2: A. The computation graph of the unfolded CDL. Its forward pass corresponds to the alternating minimization of the CDL problem in 2. The dashed part corresponds to k ISTA iterations in CSC. The network then executes a mirror descent step with the distance function ψ for the dictionary update. **B.** Illustration of a triplet (equation 10): three filters, m_1 in brown, m_2 in orange, and m_3 in grey that jointly represents a set of substrings (in purple) in a particular configuration.

the κ-th non-overlapping four-consecutive components of $\boldsymbol{d}_m$ and $\phi(\boldsymbol{v}) = \sum_i \boldsymbol{v}_i \log \boldsymbol{v}_i$ the negative entropy. We define the distance function associated with the Bregman divergence as

$$\psi(\boldsymbol{D}) = \psi(\boldsymbol{d}_1, \ldots, \boldsymbol{d}_M) = \sum_{m,\kappa} \phi(\boldsymbol{d}_{m\kappa})$$

so an update step in mirror descent can be expressed as

$$\boldsymbol{d}^t_{m\kappa j} = \frac{\boldsymbol{d}^{t-1}_{m\kappa j} \exp\big(-\mu \boldsymbol{g}_{m\kappa j}\big)}{\sum_i \boldsymbol{d}^{t-1}_{m\kappa i} \exp\Big(-\mu \boldsymbol{g}_{m\kappa i}\Big)} \qquad \text{with} \qquad \boldsymbol{g}_m = \sum_{m,n} \boldsymbol{X}^\top_{mn} \boldsymbol{X}_{mn} \boldsymbol{d}^{t-1}_m - \sum_n \boldsymbol{X}^\top_{mn} \boldsymbol{s}_n \tag{4}$$

where $\boldsymbol{g}_m$ is the gradient of $\boldsymbol{d}_m$, μ is the step-size, and $\boldsymbol{d}_{m\kappa j} = \boldsymbol{d}_{m\kappa}[j]$ is the j-th component of $\boldsymbol{d}_{m\kappa}$.

Informed by equations 3 and 4, we unfold this iterative procedure of alternating minimization into a neural network. This technique, called *deep unfolding*, obtains an approximate sparse representation and greatly speeds up the optimization [MLE21; GL10].

2.4 Approximate the sparse representation with deep unfolding

Our first K layers of the neural network consist of nonlinearities that correspond to the ISTA iterations in CSC:

$$\boldsymbol{X}^k = \mathcal{T}_{\lambda_k \eta_k}\left(\boldsymbol{X}^{k-1} + \eta_k \boldsymbol{V}^\top \left(\boldsymbol{S} - \boldsymbol{W}\boldsymbol{X}^{k-1}\right)\right), \quad k \in [K] \tag{5}$$

where the dictionary $\boldsymbol{D}$ in 3 is now "untied" and replaced with network weights $\boldsymbol{V}$ and $\boldsymbol{W}$. The untying of weights induces competition among the filters and their corresponding sparse codes, a phenomenon termed "mutual inhibition and explaining away" in [GL10]. The parameters λ_k and η_k now become learnable weights in the network. We take the sparse code from the K-th layer, $\boldsymbol{X}^K$, and introduce the network weight $\boldsymbol{Q}$ that corresponds to filters $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_M$ to form a single update step to form a dictionary $\boldsymbol{A}$

$$\boldsymbol{a}_{m\kappa j} = \frac{\boldsymbol{q}_{m\kappa j} \exp\left(-\mu \boldsymbol{g}_{m\kappa j}\right)}{\sum_i \boldsymbol{q}_{m\kappa i} \exp\Big(-\mu \boldsymbol{g}_{m\kappa i}\Big)} \tag{6}$$

for which we keep the same notation convention as in 4, and the step-size μ is learnable. Altogether, the learnable parameter in the network is $\boldsymbol{\Theta} = \{\boldsymbol{V}, \boldsymbol{W}, \boldsymbol{Q}, \boldsymbol{\eta}, \boldsymbol{\lambda}, \mu\}$ where $\boldsymbol{\eta} = \{\eta_1, \ldots, \eta_K\}$ and $\boldsymbol{\lambda} =$

$\{ \lambda_1, \ldots, \lambda_K \}$. We formulate the loss of this network as

$$\mathcal{L}_{\boldsymbol{\Theta}}(\boldsymbol{S}) = \frac{1}{N} \left\| \boldsymbol{A}\boldsymbol{X}^K - \boldsymbol{S} \right\|_F^2 + \gamma \| \boldsymbol{A}^\top \boldsymbol{A} - \mathsf{diag}(\boldsymbol{A}^\top \boldsymbol{A}) \|_F^2 \tag{7}$$

The term $\gamma \| \boldsymbol{A}^\top \boldsymbol{A} - \mathsf{diag}(\boldsymbol{A}^\top \boldsymbol{A}) \|_F^2$ is the coherence regularizer that ensures the filters in $\boldsymbol{A}$ are diverse, and γ is a trade-off parameter of this regularizer and the reconstruction loss.

2.5 Motif discovery via sparse representations

Once we obtained the learned sparse representation from 7, we consider the set of sparse code components with high magnitude:

$$\boldsymbol{V}_\alpha = \{ \, \boldsymbol{x}_{mn}[i] : \boldsymbol{x}_{mn}[i] > \alpha, \, i \in [L], \, m \in [M], \, n \in [N] \, \} \tag{8}$$

where α is a positive real number, e.g., selected by fixing a percentile of the magnitudes of all sparse-code components. Next, given three filters indexed by m_1, m_2, m_3, integers $\triangle_1, \triangle_2, \triangle_3$, and a neighborhood parameter δ, we look for integers $1 \leq i_1 \leq i_2 \leq i_3 \leq L$ and $n \in [N]$ satisfying the following condition

$$\begin{aligned} &\boldsymbol{x}_{m_1 n}[i_1] \in \boldsymbol{V}_\alpha, \qquad i_2 - i_1 = \triangle_1 \leq \delta, \qquad i_1 = i_2 \Rightarrow m_1 \neq m_2 \\ &\boldsymbol{x}_{m_2 n}[i_2] \in \boldsymbol{V}_\alpha, \qquad i_3 - i_1 = \triangle_2 \leq \delta, \qquad i_1 = i_3 \Rightarrow m_1 \neq m_3 \\ &\boldsymbol{x}_{m_3 n}[i_3] \in \boldsymbol{V}_\alpha, \qquad i_3 - i_2 = \triangle_3 \leq \delta, \qquad i_2 = i_3 \Rightarrow m_2 \neq m_3. \end{aligned} \tag{9}$$

The set

$$\begin{aligned} &\boldsymbol{E}_k \left(m_1, m_2, m_3, \triangle_1, \triangle_2, \triangle_3 \right) = \\ &\quad \{ \, (i_1, i_2, i_3, n) : \text{condition 9 is satisfied}, \, n \in [N] \, \} \end{aligned} \tag{10}$$

characterizes how three filters m_1, m_2, m_3 jointly represent a set of DNA strings in a particular configuration. We will simply call such a set a *triplet* and its cardinality $|\boldsymbol{E}_k \left(m_1, m_2, m_3, \triangle_1, \triangle_2, \triangle_3 \right)|$ as *triplet count*.

The distribution of the triplet count is positively skewed. The highly enriched triplets on the right tend to represent the frequently occurring DNA sequence pattern, shown in figure 3. Each of these enriched triplets defines a multiple sequence alignment (MSA) and hence a PWM. Note that the enumeration of the triplet (equation 10), if done naively, results in time complexity $\mathcal{O}(NL^3)$. However, when the valid neighborhood parameter δ is small, e.g., $\delta \leq 20$, we can first locate the valid neighborhood of all the triplets with a binary search and then perform the enumerations – this reduces the time complexity to $\mathcal{O}(NL \log L)$.

We perform a generalized ESD test [Ros83] to identify the outliers in the triplet count distribution, which are the enriched triplets. With a suitable significance level and an upper bound to cap the number of outliers, we identify the enriched triplets and transform them into PWMs. Following the approach in [WS03], we merge similar PWMs using the average log-likelihood ratio. We expand/trim the width of PWMs by calculating the information content of adjacent positions/columns [SS90]. We then perform greedy alignment [SH89; HHS90; LMW99] with the merged PWMs $\boldsymbol{P}_1, \ldots, \boldsymbol{P}_J$ to capture the binding sites the enriched triplets may not cover.

In the greedy alignment, each PWM $\boldsymbol{P}_j$ is re-estimated with the aligned positions that each score above a threshold τ_j. This step allows zero or more occurrences per sequence for each PWM. We obtain the score-threshold τ_j from an approximation algorithm, pvalue2Score, with a well-chosen p-value [TV07]. If some positions are selected by more than one PWM, we let the PWM that achieves the highest score re-estimate with such positions. The greedy alignment ensures all the estimated PWMs are from sets of mutually disjoint positions. We use Fisher's exact test on the held-out sequences to evaluate the significance of discovered motifs.

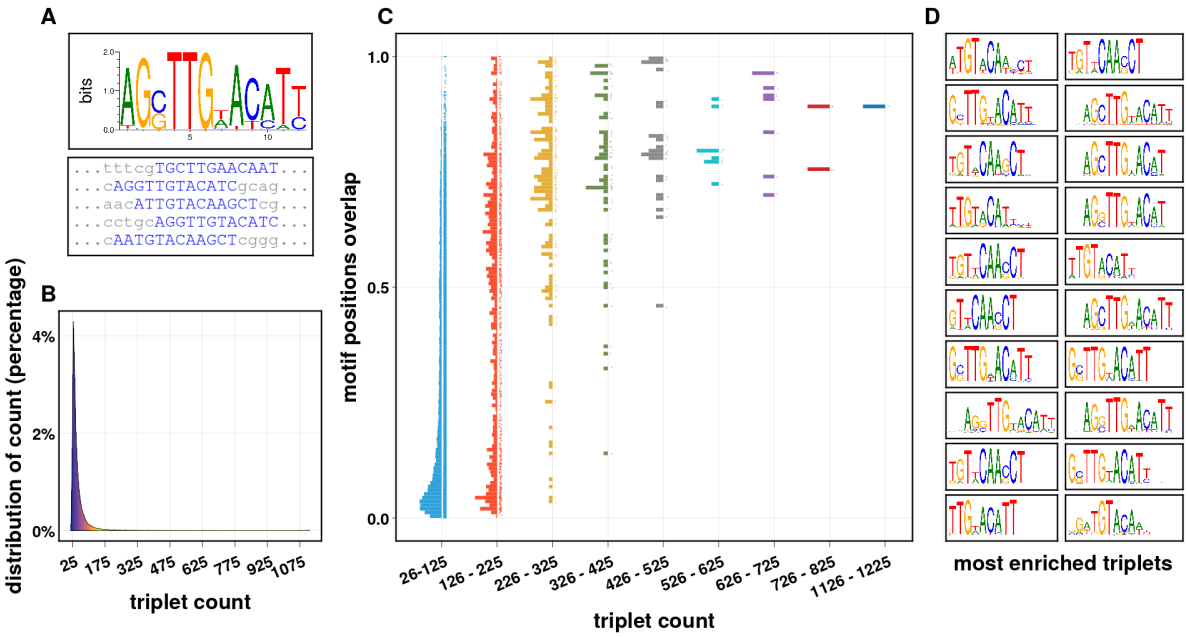

Figure 3: Illustration of the effectiveness of triplets on simple simulated data, using all sparse code components above the percentile $\alpha = 99.5\%$ (equation 8). **A.** A ground truth motif and its realizations, in blue, are planted in 2,500 i.i.d. background strings (single motif, one occurrence per sequence, and equiprobable to occur in both forward and reverse complement direction). **B.** The empirical distribution of the triplet count, $|\boldsymbol{E}_k(\cdot)|$, for triplets with counts greater than 25. **C.** Distribution of motif position overlap for different ranges of the triplet counts. The motif position overlap is $|G \cap T|/|T|$, where G is the positions (blue strings in A) of the planted motif and T is the positions covered by the triplet. This plot shows an upward trend of the overlap: the higher the triplet count, the more likely the set of positions covered by the triplet intersects the set of planted motif positions. **D.** The top 20 triplets with the highest counts, with the substrings they covered each turned into an MSA, and here visualized as PWMs.

3 Results

We name our method as *unfolded convolutional dictionary learning* (uCDL). The following sections show results in both simulated and real biological data. For all experiments, we compare with STREME version 5.5.0 and HOMER version 4.11. The command line arguments used are `streme --maxw 30 -p <input> -o <output>` and `findMotifs.pl <input> fasta <output>`.

3.1 Simulated data: motifs with multiple binding patterns

To capture motifs of multimeric binding and multiple binding domains [SG14], we construct each of the simulated datasets in the following manner. The ground truth motifs are PFMs. Specifically, we draw a point from the Dirichlet distribution with a concentration parameter of 0.1 as each column in the ground truth PFM. We then

1. Determine the number of *binding patterns*, i.e., the number of motifs in the dataset, by drawing a number from *DiscreteUniform*(2,3,4,5).
2. Determine the width of each motif by drawing instances from *DiscreteUniform*$(8, 9, \ldots, 22)$.
3. Once the first motif $\boldsymbol{G}_1$ is constructed with width w_1, the second motif is constructed by drawing an overlap ratio o from $\boldsymbol{o} \sim$ *Uniform*$([0.1, 0.9])$. We force the first columns of the second motif $\boldsymbol{G}_2$ to be identical to columns $\boldsymbol{G}_1[:, w_1 - \lceil o \times w_1 \rceil + 1{:}w_1]$. The upper-bound 0.9 and the lower-bound 0.1 for the range of the random variable $\boldsymbol{o}$ guarantees that there is at least some overlap, but the two motifs are never identical.
4. If there are more than two binding patterns, as indicated by 1), the 3rd motif uniformly and randomly picks either the first or the second motif to overlap. Then we construct it following the procedure in 3). We generate the 4th and the 5th motif similarly if needed.
5. We generate 3,000 100bp sequences $\{\boldsymbol{s}_n\}$ with i.i.d. background from *DiscreteUniform*$\{$ A,C,G,T $\}$. For each sequence $\boldsymbol{s}_n$, we uniformly and randomly chose a location to plant a motif instance, where we draw the instance from one of the ground truth motifs uniformly at random. The planted instance is equally likely to occur in its forward or reverse complement direction.

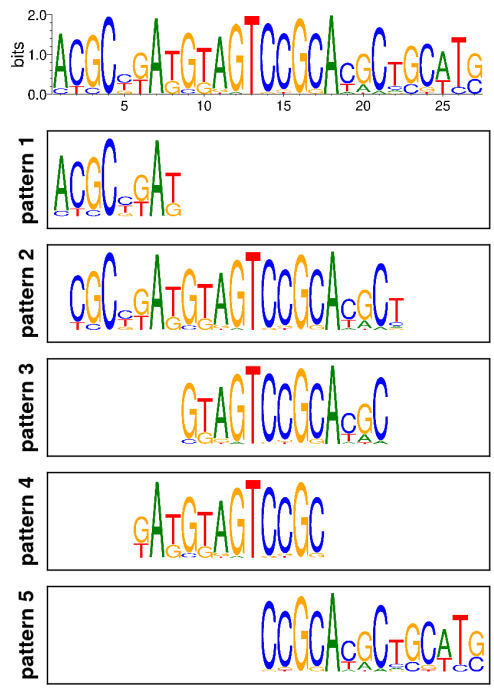

Figure 4: (example) Simulated ground truth motifs with five binding patterns, viewed as PWMs. The top PWM, not directly used in the simulated data, characterizes the intrinsic binding specificity of an imaginary protein, e.g., a Zinc finger protein. Each column is a point drawn from the Dirichlet distribution with a concentration of 0.1. We plant the instances from the five distinct binding patterns into the background strings.

3.2 Simulated data: motifs with variable spacing

We construct each simulated dataset that contains motifs with variable spacing as follows:

1. Determine the binding sites as "blocks" of motifs by uniformly randomly selecting the number of blocks from *Uniform(2,3)*. Simulate the motif for each "block" in the same manner as step 2 in 3.1.

2. In between every two i-th and $(i{+}1)$-th blocks of motifs, we set a random variable $v_{i,i+1} \sim$ *DiscreteUniform*$(1, \nu)$ where ν is a number drawn from *DiscreteUniform*(1,10). The random variable $v_{i,i+1}$ characterizes the variable spacing in between each adjacent block of motifs, and ν is the maximal number of nucleotides that separate the i-th and $(i{+}1)$-th motifs.

3. We generate 3,000 100bp sequences $\{\boldsymbol{s}_n\}$ with i.i.d. background from *DiscreteUniform*{ A,C,G,T }. For each sequence $\boldsymbol{s}_n$, we uniformly and randomly chose a location to plant an instance of the ground truth motifs; we draw a motif instance from each block, and each i-th and $(i{+}1)$-th adjacent block is separated by the number of background nucleotides determined by drawing a number from each random variable $v_{i,i+1}$'s. The planted instance is equally likely to occur in its forward or reverse complement direction

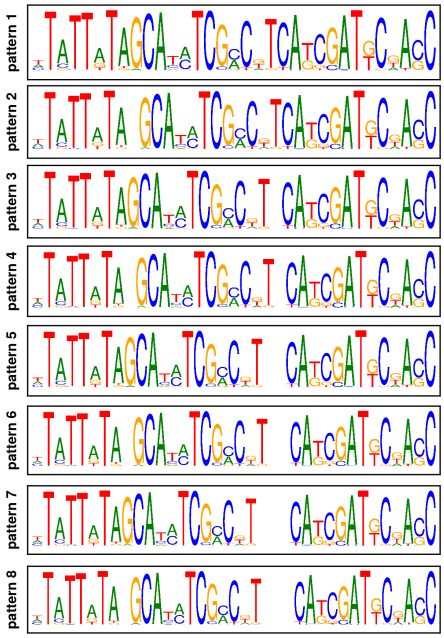

Figure 5: (example) A simulated ground truth motif of three blocks with variable spacing. Two spacers are random variables $v_{12}\sim$*DiscreteUniform*$(0,1)$ and $v_{23}\sim$ *DiscreteUniform*$(0,1,2,3)$that characterize the number of background nucleotide in between the adjacent blocks.

We generate 2,000 simulated datasets following the above procedure for each simulated data scenario: motifs with multiple binding patterns and motifs with variable spacing. For each simulated dataset, denote G_i and D_j, the set of positions of the i-th ground truth motif and the j-th discovered motif, respectively. Let $G = \cup_i G_i$ be the union of the planted positions of all the ground truth motifs, and $D = \cup_j D_j$ be the union of positions of all the occurrences of the discovered motifs. The positions of occurrences of the discovered motifs are the positions that motifs (PWMs) scored above a threshold; we set each discovered PWM's threshold using pvalue2score with p-value set to 2.5e-4 ([TV07]). Following [PS+00], we define the performance coefficient as $|G \cap D|/|G \cup D|$, sensitivity $|G \cap D|/|D|$, and specificity $|G \cap D|/|G|$ as the performance measure. We calculated the performance measures on the test sets: for each simulated dataset, the test set is another newly generated 3,000 100bp strings using its respective ground truth.

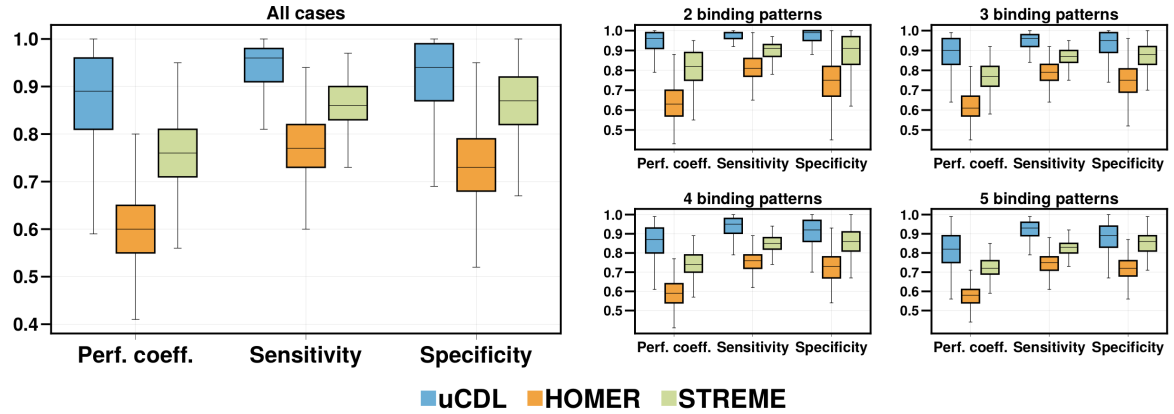

Figure 6: Performance on the simulated motifs with multiple binding patterns. The overall performance on all 2,000 simulated datasets is on the left, and separate cases in the 2,000 datasets with a fixed number of binding patterns are on the right. The midline in the interquartile range is the median. Our result shows that, for relatively conserved motifs in i.i.d. genomic background, uCDL on average is more sensitive and specific in all cases where multiple binding patterns exist, each is equiprobable to occur, and may share similar patterns.

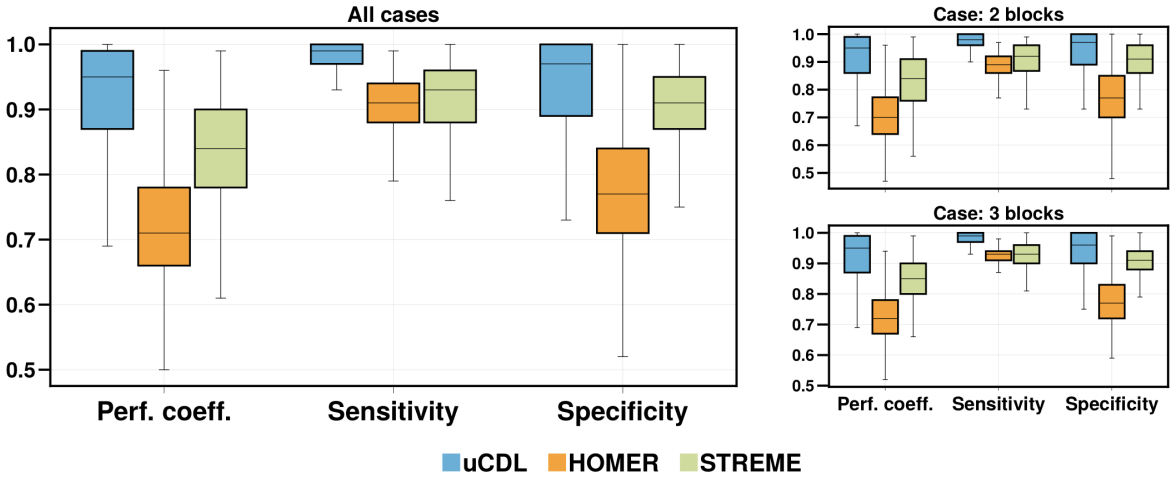

Figure 7: Performance on the simulated motifs with variable spacings. The overall performance on all 2,000 simulated datasets is on the left, and separate cases in the 2,000 cases with two blocks or three blocks of motifs are on the right. The midline in the interquartile range is the median. Our result shows that, for relatively conserved motifs in i.i.d. genomic background, uCDL is on average more sensitive and specific for two to three blocks of motifs with variable spacing.

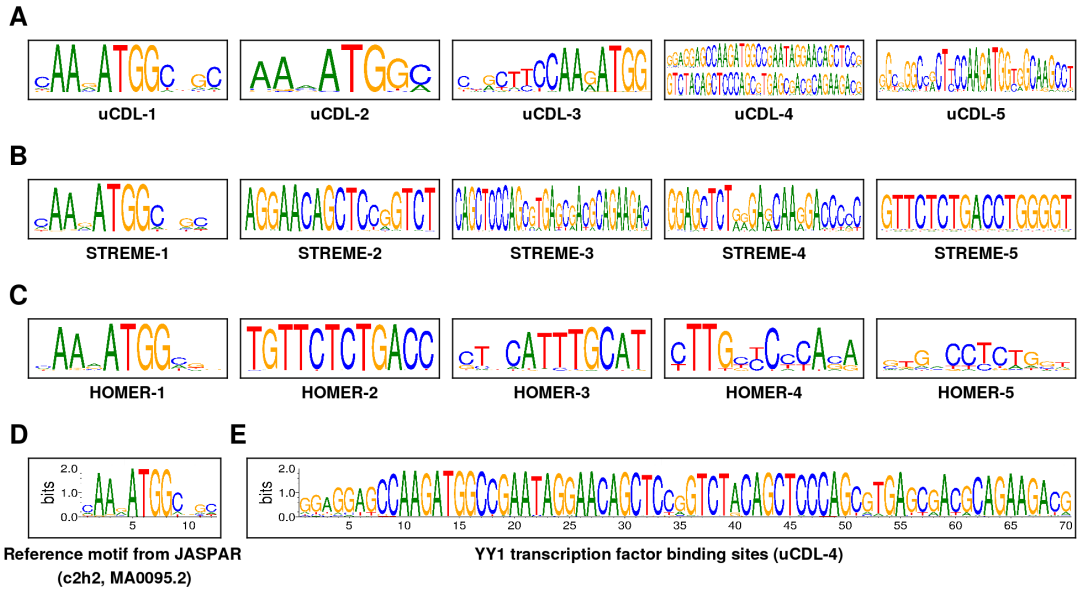

Figure 8: Motifs discovery results on MA0095.2 (C2H2) from JASPAR [Cas+22]. We list the significant reported motifs from each software in the order of statistical significance. We have the top 5 motifs **A.** discovered by uCDL. **B.** discovered by STREME. **C.** discovered by HOMER. **D.** Reference motif from JASPAR. **E.** The L1PA5 transposable element that is known to bind the YY1 transcription factor [ABM04; Sch+13], rediscovered by uCDL.

3.3 Biological data

An example that highlights the difference between motifs discovered by uCDL compared to STREME and HOMER is in figure 8. In figure 8, all methods find the reference motif from JASPAR. Additionally, uCDL finds an alternative binding pattern (uCDL-3) absent from STREME and HOMER's results — a pattern CTTCC appended to the left of the reference motif, and a long motif (uCDL-4) that corresponds to the L1PA5 transposable element that is known to bind the YY1 transcription factor [ABM04; Sch+13]. The positions 10 to 19 of uCDL-4 overlap the reference motif. The long motif uCDL-4 is only partially discovered by STREME (STREME-2 and STREME-3, which corresponds to positions 23 and 41 of uCDL-4 in figure 8 part **(E)**. We suspect heuristic such as "masks out" leads to patterns such as uCDL-3 not discovered by STREME and HOMER.

Many datasets contain motifs that co-occur in the same sequence, especially when the two motifs are identical. We list some of such instances from the JASPAR datasets in figure 9. Our observations correspond to several modes of DNA binding pointed out in previous studies [SG14]. For example, the datasets from the BZIP family (MA0462.1, MA0478.1) contain adjacent TGAC half-sites that are concentrated at around 12 or 40 nucleotides apart from each other. The motifs in the nuclear receptors datasets (MA0114.2, MA0505.1, MA0512.1, not shown in figure) also show that different length spacers separate the AGGTCA half-sites.

Some datasets contain motifs that exhibit multimeric binding [SG14], i.e., a core motif and other motifs that are the expanded version of the core motif may exist. We observe such binding mode in the BZIP, C2H2, p53 proteins, and others (See supplementary file 2). Some datasets contain motifs with multiple DNA binding domains, e.g., we find the dataset MA0162.2 from JASPAR has another significant motif consisting of consecutive 2mers of C. Some datasets from JASPAR contain significant motifs that show alternative structural conformation, e.g., MA0102.3. We show these results in figure 10. Note that the binding mode of a protein is not mutually exclusive. For example, the significant motifs in the dataset MA0095.2 from JASPAR suggest that a protein can have binding modes exhibiting variable spacing and multimeric binding.

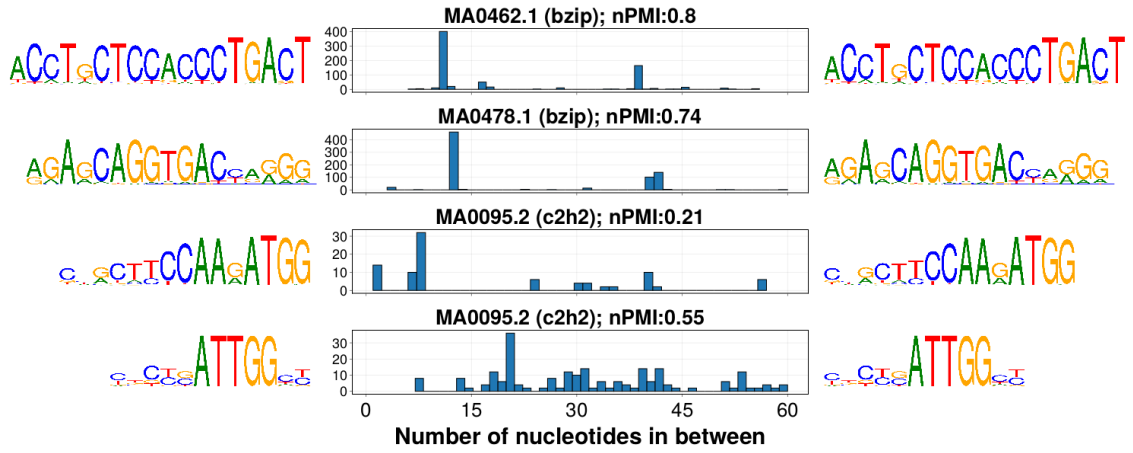

Figure 9: Motif co-occurrence on the same DNA string from some JASPAR datasets [Cas+22], not reported on JASPAR. All the motifs are statistically significant (Fisher's exact test, $p < 1\text{e}{-6}$). The normalized pairwise mutual information (nPMI) measures the degree of co-occurrence, where a value of 0 indicates no co-occurrence and a value of 1 indicates complete co-occurrence (Supplementary file 1, section C.2). The histogram considers all the co-occurring pairs on the same string, including the motifs that may co-occur more than once on the same string.

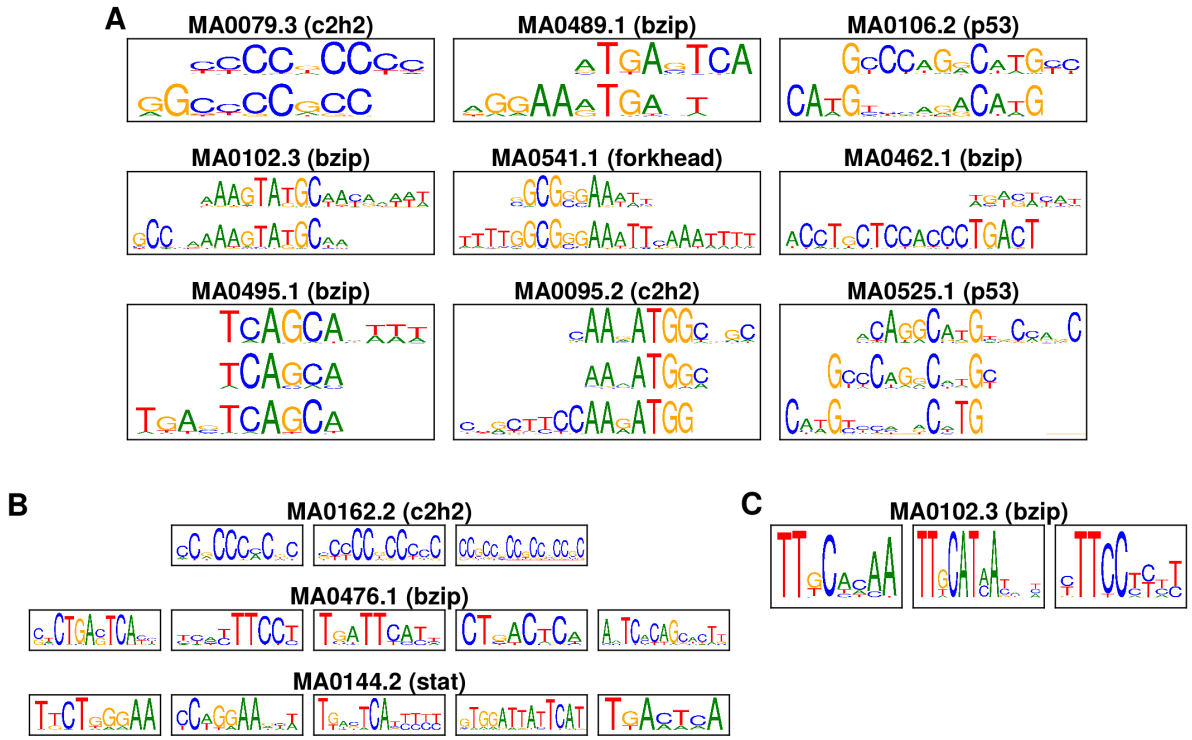

Figure 10: Motifs with **A.** multimeric binding **B.** multiple DNA binding domains **C.** alternate structural conformation [SG14] from some JASPAR datasets [Cas+22], not reported on JASPAR. All the motifs are statistically significant (Fisher's exact test, $p < 1\text{e}{-6}$).

We also found that long motifs exist in some datasets from JASPAR. These long motifs may be transposable elements and often contain the core motif as its sites. We show some of the long motifs we discovered in

figure 11.

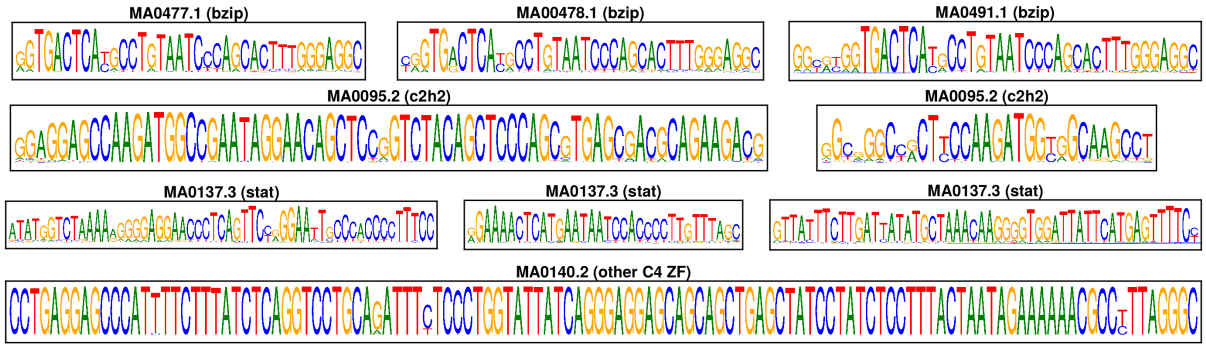

Figure 11: Some long motifs discovered from the JASPAR datasets [Kha+18], not reported on JASPAR. All the motifs are statistically significant (Fisher's exact test, $p < 1\text{e}{-6}$)

3.4 Speed

uCDL finds motifs for large datasets in a reasonable time. As shown in figure 12, using simulated data in section 3.1, it takes on average about 13 minutes to find motifs for a dataset that contains 60,000 100bp input DNA strings. Figure 12 shows that the runtime of uCDL is linear to the input size. We performed the tests on a computer with an NVidia Geforce RTX 3090 GPU, AMD Ryzen 5 5600X 6-Core Processor, and 64 GB of RAM. The CDL model consists of 160 length-8 filters, and we trained the network for 25 epochs. Further improvement is possible, for example, if we could design a learning-based technique that learns a map to replace the approximation algorithm pvalue2score [TV07] for the PWM score threshold.

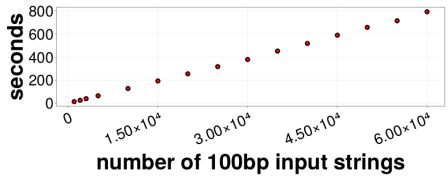

Figure 12: Average run of 10 simulated datasets (section 3.1) for motif discovery with uCDL.

4 Discussion

Some may ask, as we set the filters as position frequency matrix in section 2.2: why do we not interpret the filters as the motifs? The answer is that since our model is in a distributed representation, it rarely represents a substring using a single filter. The filters are almost always jointly, i.e., linearly combined to represent the substrings. Further, the batch size used to train the unfolded CDL (uCDL) network can also affect how each filter is learned; we show this in supplementary file 1, section B.3. Our results show that the triplet (section 2.5) is a more reliable way to look at frequently occurring patterns. We constrain the filters as position frequency matrices simply because this is more interpretable than just an ordinary feature vector.

The uCDL (section 2.4) is a different model than the CDL model (section 2.2) but enjoys some remarkable advantages over the CDL without sacrificing model interpretability. An advantage of uCDL is that the training is much faster than CDL – we no longer compute the gradient using all data points as in typical optimization subroutines – we now train the model using backpropagation with stochastic gradient descent. As a model, uCDL is a neural network with a structural bias to learn a sparse representation. Every layer in the uCDL has a clear interpretation, as it mimics the alternating minimization of CDL, outlined in section **??**. Another advantage of uCDL is that it is more expressive than the CDL model, e.g., the sparsity-inducing parameter λ and the step-sizes η_t and μ in CDL are now parameters of the network, no longer require hyperparameter tuning.

A limitation of uCDL is that we do not obtain a unique sparse representation for each run of the motif discovery. Motif discovery methods do not generally provide identifiability guarantees, especially for over-parameterized models such as neural networks. However, our implementation of uCDL will only report statistically significant motifs, which should be mostly identical in each run. We encourage users who use our program to train uCDL multiple times when in doubt.

We can use the computational graph in figure 2 for various downstream regulatory genomics problems such as functional DNA classifications ([Gha+14]) and promoter strength estimations ([Jor+22]). We can view the computational graph as an encoder-decoder architecture: the encoder takes the DNA strings as input and outputs a sparse code. We can then design a decoder that takes the sparse code for other applications. Attacking regulatory genomics problems from such an approach provides advantages for interpretability as the sparse code carries syntactic and semantic structures in the DNA strings. The filters can be viewd as "soft" kmers and that sparse coding is a trick for finding kmers enrichment without explicit enumerations.

5 Usage

Our method written in Julia [Bez+17], and is available under the MIT license as open source software at https://github.com/kchu25/uCDL.jl. The user can install our software from the Julia registry via the command `add uCDL`. Once installed, the user can run the program for motif discovery with `using uCDL` followed by `find motif(<fasta path>, <output path>)`.

6 Conclusion

We present our unfolded convolutional dictionary learning (uCDL) method for motif discovery. Both simulated and experimental data suggest that uCDL is more effective at capturing motifs that exhibit complex binding modes than methods such as STREME and HOMER. The complex binding modes include variable spacing, multimeric binding, multiple DNA binding domains, and alternative structural conformation [SG14]. Our method is the first that efficiently infers the motifs via a sparse representation, a model with distributed representation that contains intuitive built-in interpretations that allow for efficient inference.Our approach is scalable to large datasets. We can extend the resulting computational graph from deep unfolding for downstream regulatory genomics problems to extract biological strings' syntactic and semantic structures.

7 Acknowledgement

We thank Ting Wang for help identifying the L1 repetitive element and Ulugbek Kamilov for the suggestion of using deep unfolding to scale up convolutional dictionary learning. This research is funded by NIH grant 1R01GM125736 and internal funds from Washington University School of Medicine.

References

[NY83] Arkadij Semenovič Nemirovskij and David Borisovich Yudin. "Problem complexity and method efficiency in optimization". In: (1983).

[Ros83] Bernard Rosner. "Percentage points for a generalized ESD many-outlier procedure". In: *Technometrics* 25.2 (1983), pp. 165–172.

[Hin84] Geoffrey E Hinton. "Distributed representations". In: (1984).

[SH89] Gary D Stormo and George W Hartzell. "Identifying protein-binding sites from unaligned DNA fragments". In: *Proceedings of the National Academy of Sciences* 86.4 (1989), pp. 1183–1187.

[HHS90] Gerald Z Hertz, George W Hartzell III, and Gary D Stormo. "Identification of consensus patterns in unaligned DNA sequences known to be functionally related". In: *Bioinformatics* 6.2 (1990), pp. 81–92.

[SS90] Thomas D Schneider and R Michael Stephens. "Sequence logos: a new way to display consensus sequences". In: *Nucleic acids research* 18.20 (1990), pp. 6097–6100.

[BE95] Timothy L Bailey and Charles Elkan. "Unsupervised learning of multiple motifs in biopolymers using expectation maximization". In: *Machine learning* 21.1 (1995), pp. 51–80.

[LMW99] Ming Li, Bin Ma, and Lusheng Wang. "Finding similar regions in many strings". In: *Proceedings of the thirty-first annual ACM symposium on Theory of computing*. 1999, pp. 473–482.

[PS+00] Pavel A Pevzner, Sing-Hoi Sze, et al. "Combinatorial approaches to finding subtle signals in DNA sequences." In: *ISMB*. Vol. 8. 2000, pp. 269–278.

[BT03] Amir Beck and Marc Teboulle. "Mirror descent and nonlinear projected subgradient methods for convex optimization". In: *Operations Research Letters* 31.3 (2003), pp. 167–175.

[WS03] Ting Wang and Gary D Stormo. "Combining phylogenetic data with co-regulated genes to identify regulatory motifs". In: *Bioinformatics* 19.18 (2003), pp. 2369–2380.

[ABM04] Jyoti N Athanikar, Richard M Badge, and John V Moran. "A YY1-binding site is required for accurate human LINE-1 transcription initiation". In: *Nucleic acids research* 32.13 (2004), pp. 3846–3855.

[TV07] Hélène Touzet and Jean-Stéphane Varré. "Efficient and accurate P-value computation for Position Weight Matrices". In: *Algorithms for Molecular Biology* 2.1 (2007), pp. 1–12.

[GL10] Karol Gregor and Yann LeCun. "Learning fast approximations of sparse coding". In: *Proceedings of the 27th international conference on international conference on machine learning*. 2010, pp. 399–406.

[BEL13] Hilton Bristow, Anders Eriksson, and Simon Lucey. "Fast convolutional sparse coding". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2013, pp. 391–398.

[Sch+13] Petra C Schwalie et al. "Co-binding by YY1 identifies the transcriptionally active, highly conserved set of CTCF-bound regions in primate genomes". In: *Genome biology* 14.12 (2013), pp. 1–15.

[Gha+14] Mahmoud Ghandi et al. "Enhanced regulatory sequence prediction using gapped k-mer features". In: *PLoS computational biology* 10.7 (2014), e1003711.

[SG14] Trevor Siggers and Raluca Gordân. "Protein–DNA binding: complexities and multi-protein codes". In: *Nucleic acids research* 42.4 (2014), pp. 2099–2111.

[HHW15] Felix Heide, Wolfgang Heidrich, and Gordon Wetzstein. "Fast and Flexible Convolutional Sparse Coding". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2015.

[Bez+17] Jeff Bezanson et al. "Julia: A fresh approach to numerical computing". In: *SIAM review* 59.1 (2017), pp. 65–98.

[LL17] Scott M Lundberg and Su-In Lee. "A unified approach to interpreting model predictions". In: *Advances in neural information processing systems* 30 (2017).

[Woh17] Brendt Wohlberg. "Convolutional sparse coding with overlapping group norms". In: *arXiv preprint arXiv:1708.09038* (2017).

[DI18] Bogdan Dumitrescu and Paul Irofti. *Dictionary learning algorithms and applications*. Springer, 2018.

[GW18] Cristina Garcia-Cardona and Brendt Wohlberg. "Convolutional dictionary learning: A comparative review and new algorithms". In: *IEEE Transactions on Computational Imaging* 4.3 (2018), pp. 366–381.

[Kha+18] Aziz Khan et al. "JASPAR 2018: update of the open-access database of transcription factor binding profiles and its web framework". In: *Nucleic acids research* 46.D1 (2018), pp. D260–D266.

[KP21] Peter K Koo and Matt Ploenzke. "Improving representations of genomic sequence motifs in convolutional networks with exponential activations". In: *Nature machine intelligence* 3.3 (2021), pp. 258–266.

[MLE21] Vishal Monga, Yuelong Li, and Yonina C Eldar. "Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing". In: *IEEE Signal Processing Magazine* 38.2 (2021), pp. 18–44.

[Cas+22] Jaime A Castro-Mondragon et al. "JASPAR 2022: the 9th release of the open-access database of transcription factor binding profiles". In: *Nucleic acids research* 50.D1 (2022), pp. D165–D173.

[Jor+22] Tobias Jores et al. “Learning the Grammar of Regulatory DNA in Plants”. In: *Plant and Animal Genome XXIX Conference (January 8-12, 2022)*. PAG. 2022.

[Nov+22] Gherman Novakovsky et al. “Obtaining genetics insights from deep learning via explainable artificial intelligence”. In: *Nature Reviews Genetics* (2022), pp. 1–13.