6-1-2023

Finding motifs using DNA images derived from sparse representations

Shane K Chu
Washington University School of Medicine in St. Louis
Gary D Stormo
Washington University School of Medicine in St. Louis

Bioinformatics, 2023, **39(6)**, btad378
https://doi.org/10.1093/bioinformatics/btad378
Advance access publication 9 June 2023
Original Paper

OXFORD

Sequence analysis

Finding motifs using DNA images derived from sparse representations

Shane K. Chu [1,*] **and Gary D. Stormo** [2]

[1]Department of Computer Science and Engineering, Washington University in St. Louis, St. Louis, MO 63130, United States
[2]Department of Genetics, Washington University School of Medicine, St. Louis, MO 63110, United States

*Corresponding author. Department of Computer Science and Engineering, Washington University in St. Louis, McKelvey School of Engineering, 1 Brookings Drive, St. Louis, MO 63130-4899, USA. E-mail: skchu@wustl.edu (S.K.C.)

Associate Editor: Pier Luigi Martelli

Abstract

Motivation: Motifs play a crucial role in computational biology, as they provide valuable information about the binding specificity of proteins. However, conventional motif discovery methods typically rely on simple combinatoric or probabilistic approaches, which can be biased by heuristics such as substring-masking for multiple motif discovery. In recent years, deep neural networks have become increasingly popular for motif discovery, as they are capable of capturing complex patterns in data. Nonetheless, inferring motifs from neural networks remains a challenging problem, both from a modeling and computational standpoint, despite the success of these networks in supervised learning tasks.

Results: We present a principled representation learning approach based on a hierarchical sparse representation for motif discovery. Our method effectively discovers gapped, long, or overlapping motifs that we show to commonly exist in next-generation sequencing datasets, in addition to the short and enriched primary binding sites. Our model is fully interpretable, fast, and capable of capturing motifs in a large number of DNA strings. A key concept emerged from our approach—enumerating at the image level—effectively overcomes the k-mers paradigm, enabling modest computational resources for capturing the long and varied but conserved patterns, in addition to capturing the primary binding sites.

Availability and implementation: Our method is available as a Julia package under the MIT license at https://github.com/kchu25/MOTIFs.jl, and the results on experimental data can be found at https://zenodo.org/record/7783033.

1 Introduction

Identifying conserved substrings in a set of unaligned DNA strings is a fundamental challenge in computational biology. These conserved substrings, known as motifs, emerge due to evolutionary forces, and are known to play a crucial role in regulatory mechanisms. Elucidating the regulatory motifs present in specific genomic regions, such as the promoters and enhancers, can shed light on gene regulation mechanisms and contribute to our understanding of biological processes. As such, accurately identifying motifs is an essential step toward understanding the complex interplay between genes and their regulation.

Motifs are often inferred from the representations of computational models applied to DNA strings. Traditionally, motifs are inferred directly, as the conventional methods typically depict the motifs as consensus strings (Li *et al.* 1999), product multinomials (Bailey and Elkan 1995, Liu *et al.* 1995), or position weight matrices (Hertz and Stormo 1999, Heinz *et al.* 2010, Bailey 2021). While the traditional methods, e.g. STREME and HOMER, are efficient in identifying the primary motif, they often employ heuristics such as substring masking that turns the methodology into a sequential procedure. This methodological approach can make it challenging to discover secondary motifs in the dataset in a principled manner, as secondary motifs usually share identical patterns with the primary motifs. Moreover, these conventional methods frequently rely on k-mer enumeration to generate initial seeds for the optimization subroutine, constraining them to identify only ungapped motifs and limiting the maximum length of motifs that can be discovered. For example, STREME version 5.5.1 by default can only detect motifs that are up to 30 base pairs in length.

More recently, motifs are inferred from deep learning methodologies, epitomized by the use of convolutional neural network (CNN). Compared with traditional methods, inferring the motifs is less straightforward: some approaches identify the motifs as the filters in the first convolutional layers, while others use model agnostic explanation methods such as DeepLift or SHapley Additive exPlanations (SHAP) (Lundberg and Lee 2017, Shrikumar *et al.* 2017). Furthermore, the filters in CNNs are convolved with DNA strings to create embeddings (Fig. 1), which by construction are primarily intended for building regressors, e.g. predicting the bigWig coverage (Kelley *et al.* 2016, Avsec *et al.* 2021a,b, Yuan and Kelley 2022). Therefore, identifying all conserved patterns in datasets is typically not the main objective of CNNs. Consequently, the motifs predicted by CNNs are not significantly different from those predicted by traditional methods, and a principled, unifying model that effectively captures the long, gapped, and flexible motifs present in the dataset is still lacking.

In this work, we introduce a hierarchical sparse representation as a principled framework for motif discovery. Our method is capable of capturing statistically significant long,

Received: November 7, 2022. **Revised:** May 10, 2023. **Editorial Decision:** June 7, 2023. **Accepted:** June 8, 2023

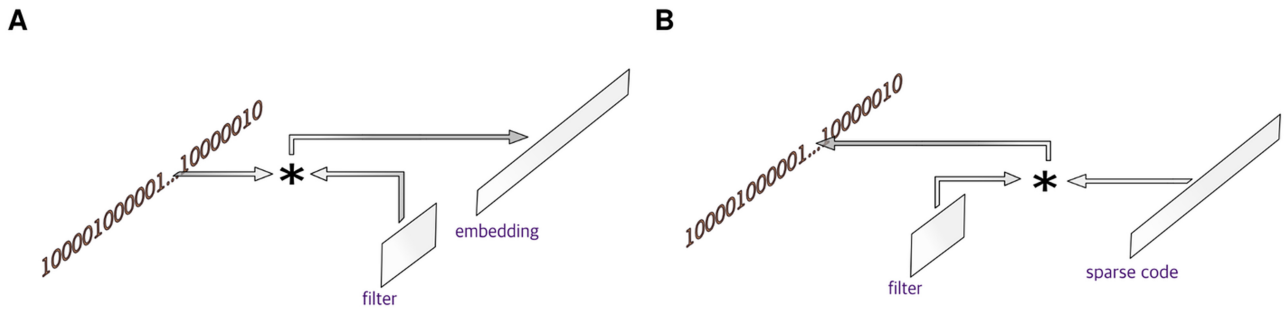

Figure 1. (A) Convolutional neural network (CNN). (B) Sparse representation. The symbol $*$ denotes the convolution. In CNN, the filters are convolved with the one-hot encoded DNA string to generate an embedding for downstream purposes, e.g. predicting the bigWig coverage. In sparse representations, the filters are convolved with the sparse code, where the sparse code plays the role of an indicator, indicating where the filters should represent the DNA string.

gapped, and flexible motifs in addition to the primary motifs in the dataset. These motifs are often longer than 30 base pairs in length, beyond the computable range for methods based on k-mer enumerations such as STREME and HOMER. In addition, as sparse representations are interpretable by design, we alleviate the need for explanation methods like SHAP. Our method can efficiently scale to large datasets containing hundreds of thousands of DNA strings, requiring only a moderate amount of computational resources.

2 Materials and methods

2.1 Notation

We refer to DNA strings as strings defined on $\Sigma = \{\mathsf{A}, \mathsf{C}, \mathsf{G}, \mathsf{T}\}$. The i-th element of a vector $\boldsymbol{v}$ is denoted by $\boldsymbol{v}[i]$. Vectors and matrices are denoted by boldface letters, while scalars are non-bold. We use the notation $\boldsymbol{x} \geq 0$ to indicate that all components of the vector or matrix $\boldsymbol{x}$ are non-negative. The function $|\cdot|$ returns the cardinality of a finite set. The norms $||\cdot||_F$, $||\cdot||_2$, $||\cdot||_1$, and $||\cdot||_0$ denote the Frobenius norm, ℓ_2 norm, ℓ_1 norm, and ℓ_0 norm, respectively.

2.2 Problem formulation

Our method begins with a simple idea: we induce a sparse representation of a DNA string by assuming that each one-hot encoded DNA string $\boldsymbol{s}_n$ can be represented as a finite sum of linear convolutions, i.e.

$$\boldsymbol{s}_n \approx \sum_m \boldsymbol{d}_m * \boldsymbol{x}_{mn} \tag{1}$$

where $\boldsymbol{d}_m$ represents the features, often called filters, and $\boldsymbol{x}_{mn}$ is the corresponding sparse vector of $\boldsymbol{s}_n$, known as the sparse code (Bristow *et al.* 2013, Heide *et al.* 2015, Wohlberg 2016, Dumitrescu and Irofti 2018, Garcia-Cardona and Wohlberg 2018). We consider the filters can be reshaped as ℓ-column position frequency matrices (PFMs) that capture the frequency of nucleotides at each position of a length-ℓ DNA string, i.e.

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

which resolves the scaling ambiguity between the filters and the sparse code. A key insight we present is that the sparse code can be likened to images. We refer to such images as *code images*, and we generate one for each string $\boldsymbol{s}_n$ by horizontally concatenating the sparse code $\boldsymbol{x}_{mn}$ for all m (Fig. 2A). Using the set of DNA strings and their corresponding code images, we proceed to construct a sparse representation specifically tailored to represent these images (Fig. 2B). By doing so, we are able to identify a more extensive and diverse set of patterns in the DNA strings by enumerating the combinations within the sparse code. This technique, which we refer to as *enumerating at the image level*, yields a broader range of significant patterns than by enumerating the k-mers in the DNA strings.

To start the motif discovery of DNA strings $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_N$, we first tackle the following problem as a precursor to enumerating at the image level:

$$\begin{aligned} \underset{\{\boldsymbol{F}_k, \boldsymbol{d}_m, \boldsymbol{x}_{mn}, \boldsymbol{y}_{mn}, \boldsymbol{z}_{kn}\}}{\operatorname{argmin}} & \frac{1}{2}\sum_n \| \sum_m \boldsymbol{d}_m * \boldsymbol{x}_{mn} + \tilde{\boldsymbol{d}}_m * \boldsymbol{y}_{mn} - \boldsymbol{s}_n \|_2^2 \\ & + \mu \textstyle\sum_k \|\boldsymbol{F}_k\|_1 + \lambda\left(\sum_{n,m} \|\boldsymbol{x}_{mn}\|_1 + \|\boldsymbol{y}_{mn}\|_1\right) \\ \text{subject to} \quad & \textstyle\sum_k \boldsymbol{F}_k * \boldsymbol{z}_{kn} = \mathcal{T}(\boldsymbol{X}_{\cdot n}, \boldsymbol{Y}_{\cdot n}) \\ & \boldsymbol{F}_k \geq 0, \|\boldsymbol{F}_k\|_F = 1 \; \forall k = 1, \ldots, K \\ & \boldsymbol{Z}_{\cdot n} \in \{\boldsymbol{Z}_{\cdot n} : ||\boldsymbol{Z}_{\cdot n}||_0 \leq \alpha\} \; \forall n = 1, \ldots, N \\ & \mathsf{reshape}(\boldsymbol{d}_m) \in \mathcal{P}_\ell \; \forall m = 1, \ldots, M \\ & \boldsymbol{x}_{mn}, \boldsymbol{y}_{mn} \geq 0 \; \forall m = 1, \ldots, M, n = 1, \ldots, N \end{aligned} \tag{2}$$

In problem 2, our first goal is to approximate DNA strings by a sum of convolutions while promoting the sparsity in the code vectors $\boldsymbol{x}_{mn}$, $\boldsymbol{y}_{mn}$. The filter $\boldsymbol{d}_m$ is reversed to obtain $\tilde{\boldsymbol{d}}_m$, enabling the consideration of patterns in both the forward and complementary strands of DNA strings. Our second goal in problem 2 is to extract patterns at the image level (Fig. 2B). To be precise, the matrices $\boldsymbol{X}_{\cdot n}$, $\boldsymbol{Y}_{\cdot n}$, and $\boldsymbol{Z}_{\cdot n}$ are constructed by horizontally concatenating the corresponding code vectors $\boldsymbol{x}_{1n}, \ldots, \boldsymbol{x}_{Mn}$, $\boldsymbol{y}_{1n}, \ldots, \boldsymbol{y}_{Mn}$, and $\boldsymbol{z}_{1n}, \ldots, \boldsymbol{z}_{Kn}$, respectively. Similarly, the matrix $(\boldsymbol{X}_{\cdot n}, , \boldsymbol{Y}_{\cdot n})$ is formed by concatenating $\boldsymbol{X}_{\cdot n}$ and $\boldsymbol{Y}_{\cdot n}$ horizontally. The transformed version of $(\boldsymbol{X}_{\cdot n}, \boldsymbol{Y}_{\cdot n})$ through a linear transform $\mathcal{T}$ is denoted as $\mathcal{T}(\boldsymbol{X}_{\cdot n}, \boldsymbol{Y}_{\cdot n})$. To ensure a parsimonious representation of the transformed image $\mathcal{T}(\boldsymbol{X}_{\cdot n}, \boldsymbol{Y}_{\cdot n})$, we constrain the matrix $\boldsymbol{Z}_{\cdot n}$ to be at most α-sparse and require the sum of convolutions

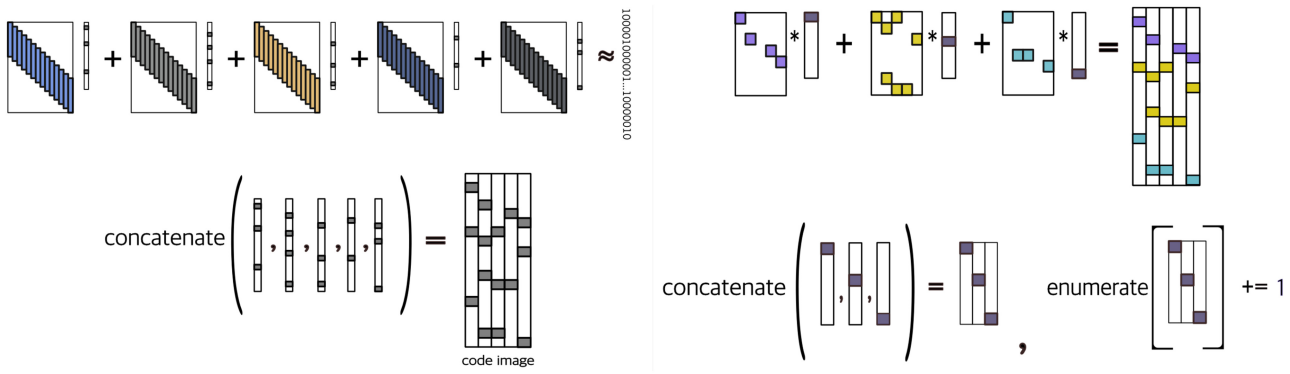

Figure 2. We show how we enumerate at the image level, with a single one-hot encoded DNA string as the input: (A) we first obtain a sparse representation of the one-hot encoded DNA string as a sum of linear convolutions. Next, we concatenate the collection of the sparse code in this sparse representation as an image. We refer to this image as a *code image*. (B) We obtain a sparse representation of the code image. Using the sparse representation of the code image, we enumerate the configurations within the sparse code.

$\sum_k \boldsymbol{F}_k * \boldsymbol{z}_{kn}$ to match $\mathcal{T}(\boldsymbol{X}_{\cdot n}, \boldsymbol{Y}_{\cdot n})$. Last, we impose a sparsity penalty on the filters $\boldsymbol{F}_k$ so that they capture the most salient features in the images $\mathcal{T}(\boldsymbol{X}_{\cdot n}, \boldsymbol{Y}_{\cdot n})$.

2.3 Obtaining the sparse representation: the classical way

We can design iterative algorithms to solve problem 2 by alternatively minimizing the sparse code $\{\boldsymbol{x}_{mn}, \boldsymbol{y}_{mn}, \boldsymbol{z}_{kn}\}$ and the filters $\{\boldsymbol{F}_k, \boldsymbol{d}_m\}$. To do so, we define $\boldsymbol{L}_{\cdot n}$ and $\boldsymbol{R}_{\cdot n}$ as the left and right halves of the image $\sum_k \boldsymbol{F}_k * \boldsymbol{z}_{kn}$, respectively, and for simplicity, let $\mathcal{T}$ be the identity transform. With the filters $\{\boldsymbol{F}_k, \boldsymbol{d}_m\}$ held fixed, we apply Alternating Direction Method of Multipliers (ADMM) (Boyd 2010) to problem 2 to solve the sparse code $\{\boldsymbol{x}_{mn}, \boldsymbol{y}_{mn}, \boldsymbol{z}_{kn}\}$:

$$\begin{aligned}
&\boldsymbol{x}_{mn}^{t+1}, \boldsymbol{y}_{mn}^{t+1} = \\
&\operatorname{argmin}_{\{\boldsymbol{x}_{mn}\geq 0, \boldsymbol{y}_{mn}\geq 0\}} \frac{1}{2}\left\| \sum_h \boldsymbol{d}_h * \boldsymbol{x}_{mn} + \tilde{\boldsymbol{d}}_h * \boldsymbol{y}_{mn} - \boldsymbol{s}_n \right\|_2^2 \\
&\qquad +\lambda(\|\boldsymbol{x}_{mn}\|_1 + \|\boldsymbol{y}_{mn}\|_1) \\
&\qquad +\frac{\rho}{2}\left(\left\| \boldsymbol{L}_{mn}^t - \boldsymbol{x}_{mn} + \boldsymbol{\Gamma}_{mn}^t \right\|_2^2 \right. \\
&\qquad \left. +\left\| \boldsymbol{R}_{mn}^t - \boldsymbol{y}_{mn} + \Upsilon_{mn}^t \right\|_2^2 \right), \\
&\qquad \forall m = 1, \ldots, M, n = 1, \ldots, N
\end{aligned} \tag{3}$$

$$\begin{aligned}
\boldsymbol{Z}_{\cdot n}^{t+1} &= \operatorname{argmin}_{\{\boldsymbol{Z}_{\cdot n}\}} 1_\alpha(\boldsymbol{Z}_{\cdot n}) \\
&\quad +\frac{\rho}{2}\left\| \sum_k \boldsymbol{F}_k * \boldsymbol{z}_{kn} - (\boldsymbol{X}_{\cdot n}^{t+1}, \boldsymbol{Y}_{\cdot n}^{t+1}) + (\boldsymbol{\Gamma}_{\cdot n}^t, \Upsilon_{\cdot n}^t) \right\|_F^2, \\
&\quad \forall n = 1, \ldots, N
\end{aligned} \tag{4}$$

$$\begin{aligned}
&(\boldsymbol{\Gamma}_{\cdot n}^{t+1}, \Upsilon_{\cdot n}^{t+1}) = (\boldsymbol{\Gamma}_{\cdot n}^t, \Upsilon_{\cdot n}^t) + \textstyle\sum_k \boldsymbol{F}_k * \boldsymbol{z}_{kn}^{t+1} - (\boldsymbol{X}_{\cdot n}^{t+1}, \boldsymbol{Y}_{\cdot n}^{t+1}), \\
&\qquad \forall n = 1, \ldots, N
\end{aligned} \tag{5}$$

where $\boldsymbol{L}_{mn}, \boldsymbol{R}_{mn}$ are the m-th column of $\boldsymbol{L}_{\cdot n}$ and $\boldsymbol{R}_{\cdot n}$, respectively. The scaled dual variables are $\boldsymbol{\Gamma}_{\cdot n}$ and $\Upsilon_{\cdot n}$, and the penalty parameter is ρ. The function $1_\alpha(\cdot)$ is an indicator function that projects the input matrix into the space of matrices with at most α non-zero elements.

The solution to equation (3) can be obtained by alternatively solving $\boldsymbol{x}_{mn}$ and $\boldsymbol{y}_{mn}$ using iterative shrinkage thresholding algorithm (ISTA), i.e. for all m, n,

$$\begin{aligned}
\boldsymbol{x}_{mn}^{t+1} &= \mathcal{S}_{\lambda\eta_t}^+(\boldsymbol{x}_{mn}^t - \eta^t \boldsymbol{d}_m \circledast \triangle - \rho(\boldsymbol{x}_{mn}^t - \boldsymbol{L}_{mn}^t - \boldsymbol{\Gamma}_{mn}^t)) \\
\boldsymbol{y}_{mn}^{t+1} &= \mathcal{S}_{\lambda\eta_t}^+(\boldsymbol{y}_{mn}^t - \eta^t \tilde{\boldsymbol{d}}_m \circledast \triangle - \rho(\boldsymbol{y}_{mn}^t - \boldsymbol{R}_{mn}^t - \Upsilon_{mn}^t))
\end{aligned} \tag{6}$$

with $\quad \triangle = \sum_h \boldsymbol{d}_h * \boldsymbol{x}_{hn}^t + \tilde{\boldsymbol{d}}_h * \boldsymbol{y}_{hn}^t - \boldsymbol{s}_n$

where $\circledast$ is the cross correlation, $\mathcal{S}_{\lambda\eta^t}^+(\cdot)$ is the non-negative soft-threshold operator, and η^t the step size at t. We consider projected gradient descent to solve equation 4. Here, an update step for all n is

$$\boldsymbol{Z}_{\cdot n}^{t+1} = \operatorname{Project}_\alpha(\boldsymbol{Z}_{\cdot n}^t - \gamma^t \mathcal{G}_Z) \tag{7}$$

where $\operatorname{Project}_\alpha(\cdot)$ keeps at most α largest magnitude components and zeros out the rest, $\mathcal{G}_Z$ the gradient of the penalty term of equation (4), and γ^t is the step size at t. On the other hand, by applying block coordinate descent with method of multipliers to problem 2 with the sparse code $\{\boldsymbol{x}_{mn}, \boldsymbol{y}_{mn}, \boldsymbol{z}_{kn}\}$ held fixed, we obtain the following iterative algorithm to solve the filters $\{\boldsymbol{F}_k, \boldsymbol{d}_m\}$:

$$\begin{aligned}
\boldsymbol{d}_m^{t+1} &= \operatorname*{argmin}_{\operatorname{reshape}(\boldsymbol{d}_m)\in\mathcal{P}_\ell} \frac{1}{2}\sum_n \left\| \sum_m \boldsymbol{d}_m * \boldsymbol{x}_{mn} + \tilde{\boldsymbol{d}}_m * \boldsymbol{y}_{mn} - \boldsymbol{s}_n \right\|_2^2, \\
&\quad \forall m = 1, \ldots, M
\end{aligned} \tag{8}$$

$$\begin{aligned}
\boldsymbol{F}_k^{t+1} &= \operatorname*{argmin}_{\boldsymbol{F}_k\geq 0} \mu \sum_k \|\boldsymbol{F}_k\|_1 \\
&\quad +\frac{\tau}{2}\sum_n \left\| \sum_k \boldsymbol{F}_k * \boldsymbol{z}_{kn} - (\boldsymbol{X}_{\cdot n}, \boldsymbol{Y}_{\cdot n}) + \boldsymbol{\Theta}_{\cdot n}^t \right\|_F^2, \\
&\quad \forall k = 1, \ldots, K
\end{aligned} \tag{9}$$

$$\begin{aligned}\boldsymbol{\Theta}_{\cdot n}^{t+1} &= \boldsymbol{\Theta}_{\cdot n}^{t} + \textstyle\sum_k \boldsymbol{F}_k * \boldsymbol{z}_{kn} - (\boldsymbol{X}_{\cdot n}, \boldsymbol{Y}_{\cdot n}),\\ &\forall n = 1, \ldots, N\end{aligned} \tag{10}$$

where τ is the penalty term and $\boldsymbol{\Theta}_{\cdot n}$ are the scaled dual variables. We can solve equation (8) via mirror descent (Beck and Teboulle 2003). Because we can reshape each $\boldsymbol{d}_m$ into a ℓ-column PFM by assumption, a way to express each component of $\boldsymbol{d}_m$ is $\boldsymbol{d}_m[4(j_1-1)+j_2]$ for $j_1 = 1, \ldots, \ell, j_2 = 1, \ldots, 4$. We define the distance function ψ associated with the mirror descent as the sum of the negative entropy of each column of the reshaped filter $\boldsymbol{d}_m$. This function is expressed as:

$$\psi(\boldsymbol{d}_1, \ldots, \boldsymbol{d}_M) = \sum_{m=1}^{M}\sum_{j_1=1}^{\ell}\sum_{j_2=1}^{4} \boldsymbol{d}_m[4(j_1-1)+j_2] \log\left(\boldsymbol{d}_m[4(j_1-1)+j_2]\right).$$

Therefore, to update each filter $\boldsymbol{d}_m$ with all components indexed by j_1 and j_2 in the mirror descent, we have the following expression:

$$\boldsymbol{d}_m^{t+1}[4(j_1-1)+j_2] = \frac{\boldsymbol{d}_m^{t}[4(j_1-1)+j_2] \exp\left(-\pi^t \boldsymbol{g}_m^{t}[4(j_1-1)+j_2]\right)}{\sum_{j_3=1}^{4} \boldsymbol{d}_m^{t}[4(j_1-1)+j_3] \exp(-\pi^t \boldsymbol{g}_m^{t}[4(j_1-1)+j_3])} \tag{11}$$

with π^t the step size at t, and $\boldsymbol{g}_m^t$ the gradient of $\boldsymbol{d}_m$ in equation (8). We solve equation (9) by ISTA for each filter $\boldsymbol{F}_k$:

$$\begin{aligned}\boldsymbol{F}_k^{t+1} &= \mathcal{S}_{\mu\omega^t}^{+}\left(\boldsymbol{F}_k^{t} - \omega^t \tau \sum_n \boldsymbol{z}_{kn} \circledast \triangle'\right)\\ \text{with} \quad \triangle' &= \textstyle\sum_{k'} \boldsymbol{F}_{k'}^{t} * \boldsymbol{z}_{k'n} - (\boldsymbol{X}_{\cdot n}, \boldsymbol{Y}_{\cdot n}) + \boldsymbol{\Theta}_{\cdot n}^{t}\end{aligned} \tag{12}$$

and then normalize so that each $\|\boldsymbol{F}_k\|_F = 1$. Here, ω^t is the step size at t and $\mathcal{S}_{\mu\omega^t}^{+}(\cdot)$ is the non-negative soft-threshold operator.

2.4 Obtaining the sparse representation by deep unfolding

Rather than relying on the algorithm from Section 2.3 to obtain the sparse representation, we adopt a different approach called *deep unfolding* (Gregor and LeCun 2010, Monga *et al.* 2021). The deep unfolding approach employs a neural network that parameterizes the iterates of an iterative algorithm as its forward pass, which obtains an approximate representation of the problem and has been shown to achieve significantly faster convergence in practice (Gregor and LeCun 2010).

We use deep unfolding to obtain an approximate sparse representation in problem 2. To construct our network, we use the iterates from Section 2.3, with the sparse code $\{\boldsymbol{x}_{mn}, \boldsymbol{y}_{mn}, \boldsymbol{z}_{kn}\}$ and the scaled dual variables $\{\boldsymbol{\Gamma}_{\cdot n}, \Upsilon_{\cdot n}, \boldsymbol{\Theta}_{\cdot n}\}$ in the network initialized to be zero. Next, the equations (6), (7), (11), and (12) can be implemented as non-linearities of the layers in the network. We construct the first $2 \times \mathcal{K}_1$ layers by interleaving the iterates of equations (6) and (7), and the subsequent $2 \times \mathcal{K}_2$ layers by executing the iterates of equations (11) and (12). The parameters of the our network are the filters $\{\boldsymbol{d}_m, \boldsymbol{F}_k\}$, the sparsity inducing parameters $\{\lambda, \mu\}$, the penalty parameters $\{\rho, \tau\}$, and the stepsizes $\{\eta^{t_1}, \gamma^{t_1}, \pi^{t_2}, \omega^{t_2} : t_1 = 1, \ldots, \mathcal{K}_1, t_2 = 1, \ldots, \mathcal{K}_2\}$, learned by training the network with backpropagation.

Once the network completes its forward pass, we define the loss of the network as

$$\frac{1}{N}\sum_n \left[\left\| \sum_m \boldsymbol{d}_m^{\mathcal{K}_2} * \boldsymbol{x}_{mn}^{\mathcal{K}_1} + \tilde{\boldsymbol{d}}_m^{\mathcal{K}_2} * \boldsymbol{y}_{mn}^{\mathcal{K}_1} - \boldsymbol{s}_n \right\|_2^2 + \left\| \textstyle\sum_k \boldsymbol{F}_k^{\mathcal{K}_2} * \boldsymbol{z}_{kn}^{\mathcal{K}_1} - \mathcal{T}\left(\boldsymbol{X}_{\cdot n}^{\mathcal{K}_1}, \boldsymbol{Y}_{\cdot n}^{\mathcal{K}_1}\right) \right\|_F^2 \right]$$

where the sparse code $\{\boldsymbol{x}_{mn}^{\mathcal{K}_1}, \boldsymbol{y}_{mn}^{\mathcal{K}_1}, \boldsymbol{z}_{kn}^{\mathcal{K}_1}\}$ are from the final output of the first $2 \times \mathcal{K}_1$ layers, and the filters $\{\boldsymbol{d}_m^{\mathcal{K}_2}, \boldsymbol{F}_k^{\mathcal{K}_2}\}$ are from the final output of the subsequent $2 \times \mathcal{K}_2$ layers of the network. An illustration of the unfolded network is shown in Fig. 3.

2.5 Enumerating at the image level

After the network is trained, we can retrieve the sparse code of the code images, i.e. the set $\{\boldsymbol{Z}_{\cdot n}\}$ by a network forward pass. Similar to the seeding phase in methods such as STREME and HOMER, our motif discovery also performs enumerations. Yet, instead of enumerating the oligonucleotides, we enumerate the combinations of the sparse code components in each $\boldsymbol{Z}_{\cdot n}$, for $n = 1, \ldots, N$. Specifically, we count how $q \in \mathbb{Z}_+$ components are arranged in each $\boldsymbol{Z}_{\cdot n}$. Given that $\boldsymbol{Z}_{\cdot n}$ has at most α components, the number of operations required to enumerate all possible combinations of q components in each $\boldsymbol{Z}_{\cdot n}$ is at most $\binom{\alpha}{q}$. This is a relatively small number, especially considering that we restrict both α and q to be small integers.

We refer to each combination of q components and its spatial information a *configuration*. The configuration is defined as a tuple:

$$\left(f_{(1)}, d_{(1)}, f_{(2)}, d_{(2)}, \ldots, d_{(q-1)}, f_{(q)}\right)$$

where the numbers $f_{(1)}, f_{(2)}, \ldots, f_{(q)}$ are the q filter-indices of filters $\{\boldsymbol{F}_k\}$ directly inferred from $\boldsymbol{Z}_{\cdot n}$ in the combination, sorted by their spatial occurrences in the DNA string $\boldsymbol{s}_n$. The numbers $d_{(1)}, d_{(2)}, \ldots, d_{(q-1)}$ are the distances (number of nucleotides) in between their neighboring components in the configuration. We store the configurations as keys in a hash table $\mathcal{H}$. The value associated with each key in $\mathcal{H}$ is a collection of DNA strings of the same length, each obtained by taking the DNA string in the region covered by the particular configuration, as we perform the enumerations through DNA strings $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n$. Thus, each set of the DNA strings associated with a key defines a multiple sequence alignment (MSA), and corresponds to a position weight matrix (PWM).

We select up to J most enriched PWMs in the hash table $\mathcal{H}$ as the motifs. As multiple keys in $\mathcal{H}$ may correspond to similar motifs, we reduce such redundancy by merging the similar PWMs using the average log-likelihood ratio (Wang and Stormo 2003). We report the resulting PWMs $\boldsymbol{P}_1, \ldots, \boldsymbol{P}_{J'}$ as the discovered motif in the dataset.

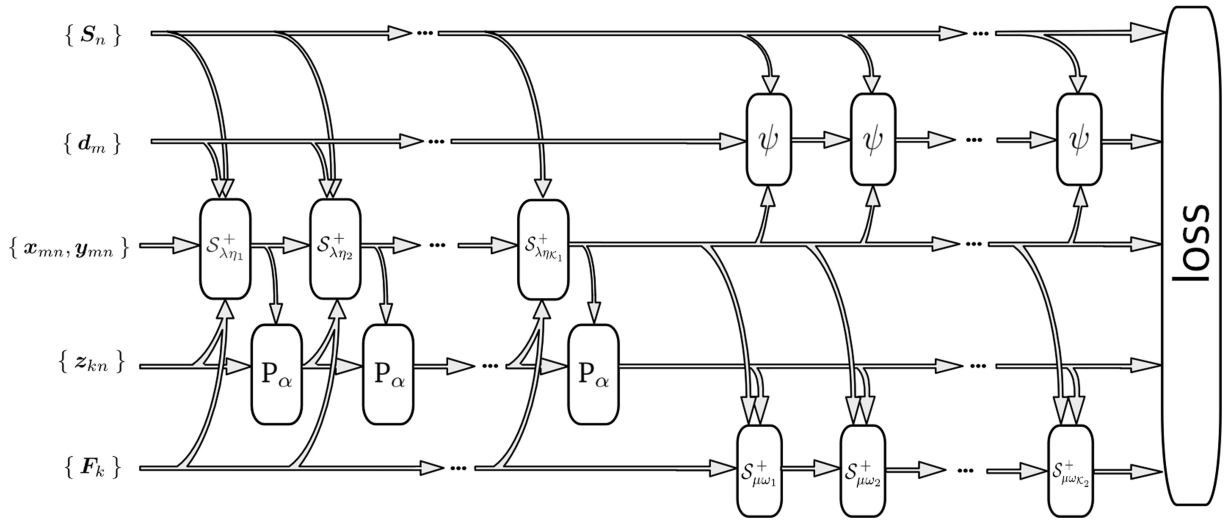

Figure 3. The unfolded network obtained by parameterizing the iterates derived in Section 2.3.

2.6 Soft versus hard clustering representation of the motifs

Since a PWM represents an average and proteins can have distinct binding preferences, the motif discovery problem is, in a sense, similar to the clustering problem. Soft-clustering scenarios, such as mixtures of Gaussians, allow a point to belong to multiple clusters, exhibiting characteristics that coexist in each of them. This situation frequently arises in motif discovery, where, for instance, the primary motif may frequently occur by itself but occasionally appears twice in certain DNA strings in the dataset.

In this work, we present our result in the soft-clustering representation. Unlike most traditional motif discovery methods that use hard clustering representations, which use PWMs composed of mutually exclusive DNA substrings, our method employs soft clustering representations that allow PWMs derived by multiple sequence alignment to share DNA substrings with other PWMs. We provide a detailed comparison of both approaches and their trade-offs in Supplementary File S1 Section C.

2.7 Implementation

2.7.1 Hyperparameters

We implement our model with $M = |\{\boldsymbol{d}_m\}| = 50$ filters for the sparse representation of DNA strings $\{\boldsymbol{s}_N\}$, where each filter (a PFM) $\boldsymbol{d}_m$ covers $\ell = 8$ nucleotides. Additionally, we use $K = |\{\boldsymbol{F}_k\}| = 24$ filters for code images, where each filter $\boldsymbol{F}_k$ can cover $12 \times (2 \times M)$ pixels. To limit the number of non-zero elements in each code image $\boldsymbol{Z}_{\cdot n}$, we set α to 32. We set the operator $\mathcal{T}$ as a scaling operator, i.e. $\mathcal{T}(\boldsymbol{M}) = \beta \boldsymbol{M}$, where $\beta = 100$ to ensure numerical stability. Our unfolded network is trained using AdaBelief (Zhuang *et al.* 2020), with a batch size of 6 DNA strings. We set $\mathcal{K}_1$ to 6 interleaved iterations on the sparse code and $\mathcal{K}_2$ to 3 iterations on the filters. This parameterization results in a network, i.e. Fig. 3, consisting of 30, 421 parameters, which is much smaller in size compared with current mainstream deep learning models. We set $q = 3$ for the configurations, which results in at most $\binom{\alpha}{q} = 4,960$ counting operations for each retrieved code image $\boldsymbol{Z}_{\cdot n}$. We select up to $J = 1000$ most enriched motifs from the stored enumerations in the hash table $\mathcal{H}$ to display in the results.

2.7.2 Motif significance

We randomly select DNA strings and set aside 15% of them as a test set that is not used during training. For each motif discovered and indexed by j, we follow this procedure: Let N_T denote the total number of available positions in the test set, where N_t is the number of DNA strings in the test set, and L is the length of each string. We define $N_T = N_t L$. The number of hits of the j-th motif in the test set is denoted by τ_h, and the number of misses is denoted by $\tau_m = N_T - \tau_h$. A hit at a position is defined as the positions occupied by the PWM that scored above its threshold. The score threshold for each PWM is determined using the approximation algorithm pvalue2score with a P-value of 1e−3 (Touzet and Varré 2007). Hits and misses for the control set are similarly defined as c_h and c_m, respectively. The control set is constructed by shuffling the DNA strings in the test set such that the 2-mer frequency is preserved. We then perform Fisher's exact test to test the null hypothesis that the odds ratio $(\tau_h/\tau_m)/(c_h/c_m)$ is one, against the alternative hypothesis that they are not equal.

2.7.3 Motif occurrences

We define the number of instances of a motif as the number of position that a PWM scores above its score threshold, i.e. the number of hits (Section 2.7.2) in the dataset.

3 Results

We take experimental datasets from JASPAR, FactorBook, ReMap, and Avsec *et al.* (Weirauch *et al.* 2014, Avsec *et al.* 2021b, Castro-Mondragon *et al.* 2022, Hammal *et al.* 2022, Pratt *et al.* 2022) to conduct motif analysis, for which we detailed our data processing steps in Supplementary File S1 Section A. In our analysis, we characterize all the motifs as position weight matrices (PWMs). All motifs presented in this section (motifs shown in Figs 4–6) are statistically significant with a P-value $< 1e-6$, as determined by the Fisher exact test (Section 2.7.2). Additionally, we make the following characterizations:

- *Primary motif*: the short (often 6–12bp), ungapped, PWM that correspond to the core binding sites of a transcription factor (TF).

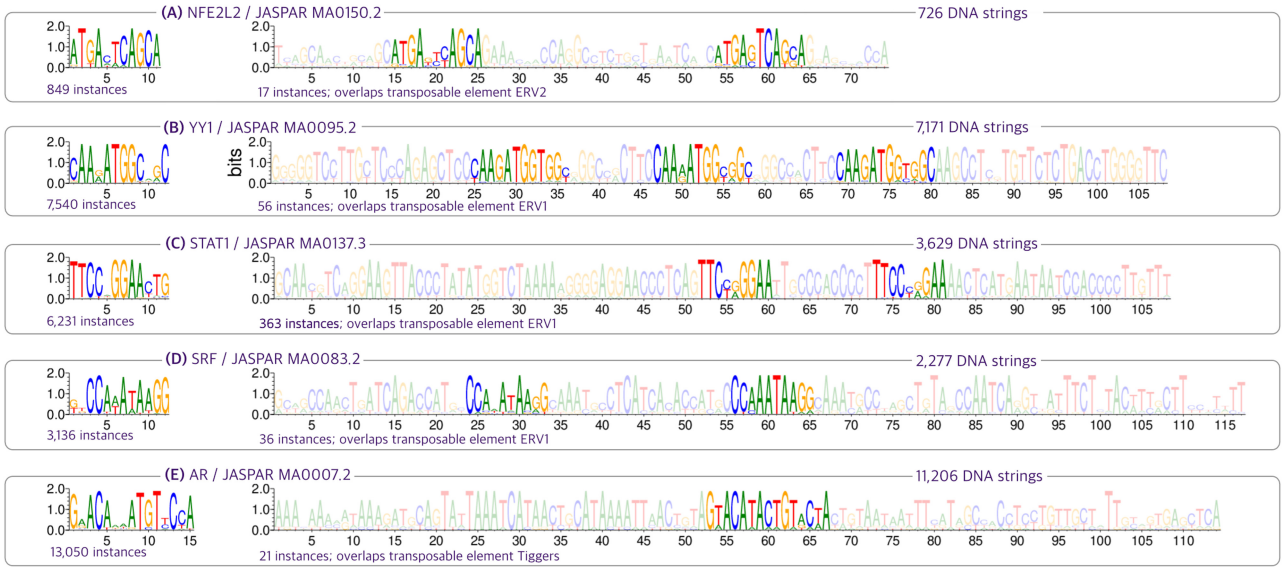

Figure 4. For each dataset, we list the TF/JASPAR-ID and the number of DNA strings in the dataset. We show how the primary motifs are embedded in the repetitive elements, with the primary motifs on the left and their overlap on the right. Below each sequence logo, we show the number of instances that occurred throughout the dataset. In short, (A) NFE2L2 overlaps the ERV2. (B, C, and D) YY1, STAT1, and SRF overlaps the ERV1. (E) AR overlaps Tiggers. Each transposable element is validated by taking the consensus string of each PWM and then search to confirm via Dfam (Hubley *et al.* 2016). For more, see Supplementary File S1 Section E.

- *Long motif*: the long (often longer than 30bp) PWM with uniformly high information content columns, e.g. Fig. 4; these motifs, like gapped motifs, can also characterize multiple binding sites of the same or different TFs.
- *Gapped motif*: the PWM that contain groups of consecutive high information content columns separated by low information content columns, e.g. PWMs in Fig. 6.

Our analysis of the datasets reveals a large presence of long motifs and gapped motifs in these experimental datasets. In particular, among the 91 ChIP-Seq datasets we selected from JASPAR, 50 of them contain long motifs that are transposable elements, as we verified in the Dfam database (Hubley *et al.* 2016) (Supplementary File S1 Section E). Moreover, gapped motifs were identified in many datasets from diverse sources, including 89 out of 198 datasets in JASPAR across various experiment types including ChIP-seq, DAP-seq, SELEX, PBM, and ChIP-Chip. This large presence of long motifs and gapped motifs in our analysis is noteworthy, as they are often overlooked in motif discovery and may have important implications in transcriptional regulation. In the following sections, we will highlight the discovery of gapped and long motifs. We skip the results on primary motif discovery as our method almost always find them (Supplementary File S1 Section F), and it is straight-forward problem tackled by many well-established methods.

3.1 Long motifs

A central theme that frequently occurs in long motif discovery is the overlap between the long motifs and the primary motifs. These overlappings present a significant challenge for traditional motif discovery methods, discussed in Section 4.1. In qualitative terms, the primary motifs can either be "embedded" within the long motifs or "compounded" in proximity to other motifs in the dataset.

3.1.1 "Embedded" motifs

Our method simultaneously discovers the primary motifs and the long motifs that embed these primary motifs, as shown in Fig. 4. We observe many such cases, especially in ChIP-Seq datasets, where TF binding sites in vivo can often overlap with repetitive elements. Our result reveals a relationship between primary motifs and repetitive elements, and suggests that the use of *repeat masking* is not strictly necessary for motif discovery.

3.1.2 "Compound" motifs

Our analysis shows that compound motifs are frequently seen in ChIP-Seq datasets. These findings suggest that our method effectively identifies binding sites including those that work in conjunction with the TF of interest, as TFs often work in concert with other TFs to regulate gene expression in vivo. We show in Fig. 5 several compound motifs that we identified in a ChIP-Nexus experiment exploring the localization of Oct4 pluripotency factor, using experimental data from (Avsec *et al.* 2021b).

3.2 Gapped motifs

Our method detects several gapped motifs that have been previously reported in the literature, shown in Fig. 6. For instance Zuo *et al.* (2023) report that traditional motif discovery methods have been shown to underestimate the binding sites of CTCF, a zinc finger protein containing 11 zinc finger domains, resulting in a paradox known as *long fingers but short motifs*. This paradox highlights that the full binding sites of evolutionarily conserved zinc finger domains may follow an irregular structure that is not easily detected by traditional methods. Our method successfully identifies the upstream motif TGCAG of the core binding sites of CTCF, as reported in (Zuo *et al.* 2023), shown in Fig. 6A. In addition, we confirm that ESR2, a Nuclear Receptor factors binds with its half sites AGGTCA (Khorasanizadeh and Rastinejad 2001,

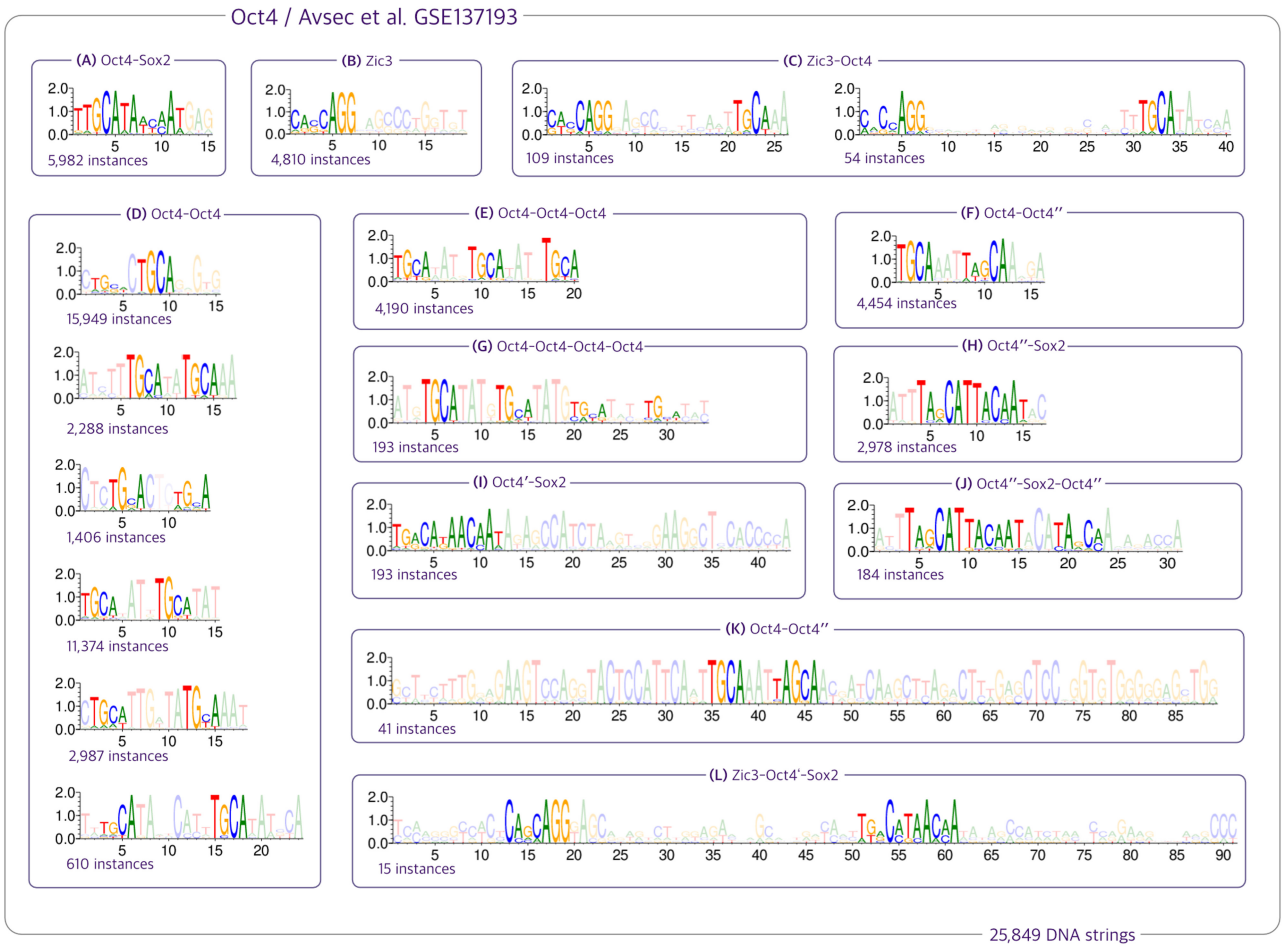

Figure 5. (A, B, and D) We validated the presence of motifs reported in the ChIP-Nexus experiment (GEO: GSE137193) conducted by (Avsec *et al.* 2021a), such as Oct4-Sox2, Oct4-Oct4, and Zic3. (D) Our analysis revealed new insights into the Oct4-Oct4 motif, including the potential for variable spacing between the half-sites TGCA, ranging from 1 to 8 nucleotides. (E, G, C) We also observed that Oct4 can co-occur up to 4 times with equal-length spacers of TATG, and it is frequently associated with Zic3 factors. (F, H, I, J, K, L) Furthermore, we identified two minor variants of Oct4, which we designate as Oct4′ and Oct4″, due to the insertion of a nucleotide A either upstream or downstream of the central nucleotide G in Oct4. Our additional findings for Sox2, Klf4, and Nanog pluripotency factors from (Avsec *et al.* 2021a) are presented in Supplementary File S1 Section F.

Siggers and Gordân 2014), which we found to be separated by a spacer up to 36 nucleotides, shown in Fig. 6B.

Notably, our method exhibits high sensitivity in quantifying the spacers within gapped motifs, providing detailed insights into gapped motifs' composition. For example, we identified a gapped motif in the MAFF factor from JASPAR that exhibits a gap between the primary motif and its partial complement, with a total of 26 spacers, shown in Supplementary File S1 Section D.

4 Discussion

4.1 Distributed representation of DNA strings using sparse representation

A key distinction between traditional and more recent approaches to motif discovery is in how motifs are represented during optimization. Traditional methods, such as STREME and HOMER, typically use local representations, such as PWMs, for motif characterization during optimization. In contrast, recent approaches often use deep learning that leverage distributive representations to learn and represent motifs (Hinton *et al.* 1986). The choice between the two types of representation involves a trade-off between interpretability and expressiveness, with local representations being easier to interpret. However, problems that characterize motifs with local representations has generally been shown to be NP-hard (Bafna *et al.* 1997, Li *et al.* 1999, Akutsu *et al.* 2000). Due to this, common heuristics, such as substring masking, are often used during optimization to find motifs, resulting a sequential procedure for motif discovery (Bailey and Elkan 1995, Heinz *et al.* 2010, Bailey 2021). This sequential procedure may pose challenges as primary motifs can overlap with secondary motifs, including gapped motifs and transposable elements present in the dataset. Our result demonstrates the existence of these secondary motifs, and there are simulated experiments indicate that methods such as STREME and HOMER are less effective when multiple motifs overlaps (Chu and Stormo 2022).

We select sparse representations for DNA strings as they provide a distributive representation with a clear interpretation. By approximating DNA strings as a sum of linear convolutions (equation (1)), the non-zero components in the sparse code essentially act as indicators, indicating where filters should be used to represent DNA substrings (Fig. 2A). The combined sparse code, which we refer to as code images, provides a high-level view on the spatial arrangements of the filters, which we build another sparse representation upon (Fig. 2B). This sparse representation on the code images

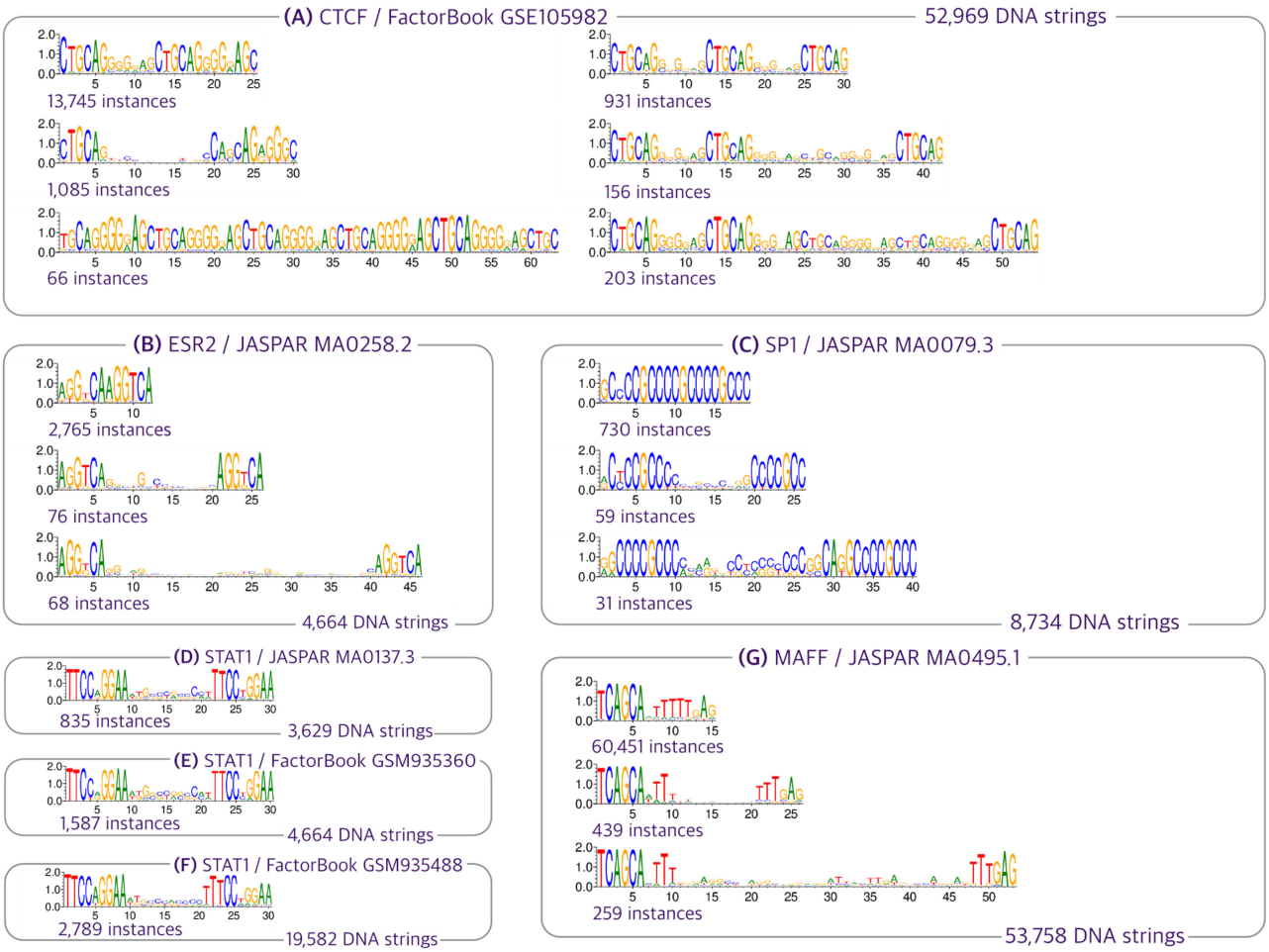

Figure 6. We present a selection of gapped motifs, chosen from the many we discovered in each dataset (labeled with TF/GEO accession number or TF/JASPAR-ID). (A) Notably, we discovered the presence of the upstream motif **CTGCAG** of CTCF (Zuo *et al.* 2023); on the right, we note that the upstream motif of the core motif occur independently of the primary motif of CTCF. (C) Our analysis reveals that the primary motif of SP1 and ESR2 exhibits repetition with spacers of varying lengths. We note that ESR2 variable spacing correspond to the findings reported in (Siggers and Gordân, 2014). (D, E, F) In the case of STAT1, we observe a fixed spacing across all three datasets. (G) We present a gapped motif discovered in MAFF, which demonstrates our method's capability to capture gapped motifs with spacers longer than 40 bp. To avoid clutter, we display only motifs with the smallest, middle, and largest spacers, as determined by their pairwise distance (number of nucleotides) for ESR2, SP1, and MAFF. A full result of our motif discovery is in Supplementary File S1 Section F.

enables us to identify the conserved patterns in the dataset, akin to enumerating k-mers at the nucleotide level. Yet, by enumerating at the image level, the spatial relationship in between the regulatory elements is preserved, in contrast to k-mers enumerations, which do not account for the spatial information.

4.2 Comparison to recent deep learning methodologies in regulatory genomics

As with recent work on deep learning for regulatory genomics (Alipanahi *et al.* 2015, Avsec *et al.* 2021b, Yuan and Kelley 2022), our approach use distributive representations to characterize patterns in DNA strings. Additionally, our method incorporates the design of a neural network (Fig. 3). However, our network design process is distinct from standard deep learning practices. Rather than using traditional design tools, we create the network architecture by unfolding the iterative algorithm detailed in Section 2.3. The unfolding technique incorporates hyperparameters, such as sparsity-inducing parameters, step-sizes, and penalty parameters from problem 2, as part of the network architecture, which are automatically tuned with backpropagation (Gregor and LeCun 2010, Monga *et al.* 2021). This results in a more expressive model compared with the original. We note that our network is fully interpretable as the forward pass can be seen as optimizing the objective posed by problem 2 via gradient descent. Inferring the motifs relies on enumerating at the image level (Section 2.5 and Fig. 2), without relying on explanation methods such as SHAP (Lundberg and Lee 2017). Our network is light in parameters (Section 2.7.1), allowing us to quickly train and infer motifs in a matter of minutes using a single GPU, and only requires DNA strings (FASTA files) as inputs.

4.3 Extensions to other regulatory genomics problems

Our method produces a computational graph (Fig. 3), which permits us to easily extend it to other regulatory genomics problems, such as DNA classifications or regressions. To accomplish this, one can treat the sparse code as an embedding, analogous to the embeddings constructed in the CNNs (Fig. 1). Furthermore, with the filters in problem 2 held fixed, the sparse code provides an alternative way of representing the binding sites. This alternative approach could be valuable for estimating the *recognition code*, e.g. designing C2H2 zinc-

fingers with novel specificity (Gupta *et al.* 2014, Najafabadi *et al.* 2015, Aizenshtein-Gazit and Orenstein 2022, Ichikawa *et al.* 2023).

4.4 Future work

Our method effectively captures the gapped motifs (Figs 5 and 6), but highly varying spacers are common in these motifs. For instance, a gapped motif in BZIP MA0495.1 from JASPAR appears with 26 different spacer lengths (Supplementary File S1 Section D), resulting in numerous PWMs in our results. We intend to improve motif summarization and visualization in the future.

5 Conclusion

We present a motif discovery method that includes the discovery of long, gapped, or overlapping motifs. Our key concept, enumerating at the image level, enables a more extensive and diverse pattern identification in DNA strings compared with enumerating the k-mers. Our study demonstrates that the sparse representation is a highly interpretable modeling technique for DNA strings. This approach enables us to reveal that transposable elements and gapped motifs are common in ChIP-Seq datasets from JASPAR, Factorbook, and Remap (Castro-Mondragon *et al.* 2022, Hammal *et al.* 2022, Pratt *et al.* 2022). Our methodology is compatible with the deep learning paradigm through deep unfolding, enabling us to extend it to various regulatory genomics problems.

Acknowledgements

We thank Petko Petkov for the initial motivation for this project, Ting Wang for help identifying the L1 repetitive element, and the anonymous reviewers for their valuable comments that helped to improve the paper.

Supplementary data

Supplementary data are available at *Bioinformatics* online.

Conflict of interest

None declared.

Funding

This research was funded by NIH [grant number 1R01GM125736]; and internal funds from Washington University School of Medicine.

References

Aizenshtein-Gazit S, Orenstein Y. Deepzf: improved DNA-binding prediction of c2h2-zinc-finger proteins by deep transfer learning. *Bioinformatics* 2022;**38**:ii62–7.

Akutsu T, Arimura H, Shimozono S. On approximation algorithms for local multiple alignment. In: *Proceedings of the Fourth Annual International Conference on Computational Molecular Biology, Tokyo, Japan*, 2000. pp. 1–7.

Alipanahi B, Delong A, Weirauch MT *et al.* Predicting the sequence specificities of DNA-and RNA-binding proteins by deep learning. *Nat Biotechnol* 2015;**33**:831–8.

Avsec Ž, Agarwal V, Visentin D *et al.* Effective gene expression prediction from sequence by integrating long-range interactions. *Nat Methods* 2021a;**18**:1196–203.

Avsec Ž, Weilert M, Shrikumar A *et al.* Base-resolution models of transcription-factor binding reveal soft motif syntax. *Nat Genet* 2021b;**53**:354–66.

Bafna V, L. Lawler E, Pevzner PA *et al.* Approximation algorithms for multiple sequence alignment. *Theor Comput Sci* 1997;**182**:233–44.

Bailey TL. Streme: accurate and versatile sequence motif discovery. *Bioinformatics* 2021;**37**:2834–40.

Bailey TL, Elkan C. Unsupervised learning of multiple motifs in biopolymers using expectation maximization. *Mach Learn* 1995;**21**:51–80.

Beck A, Teboulle M. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operat Res Lett* 2003;**31**:167–75.

Boyd S. Distributed optimization and statistical learning via the alternating direction method of multipliers. *FNT Mach Learn* 2010;**3**:1–122.

Bristow H *et al.* Fast convolutional sparse coding. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR*, 2013. pp. 391–8.

Castro-Mondragon JA, Riudavets-Puig R, Rauluseviciute I *et al.* JASPAR 2022: the 9th release of the open-access database of transcription factor binding profiles. *Nucleic Acids Res* 2022;**50**:D165–73.

Chu S, Stormo G. Deep unfolded convolutional dictionary learning for motif discovery. *bioRxiv*, 2022–11. https://doi.org/10.1101/2022.11.06.515322.

Dumitrescu B, Irofti P. *Dictionary Learning Algorithms and Applications*. Berlin, Germany:Springer, 2018.

Garcia-Cardona C, Wohlberg B. Convolutional dictionary learning: a comparative review and new algorithms. *IEEE Trans Comput Imaging* 2018;**4**:366–81.

Gregor K, LeCun Y. Learning fast approximations of sparse coding. In: *Proceedings of the 27th International Conference on International Conference on Machine Learning*, 2010, 399–406.

Gupta A, Christensen RG, Bell HA *et al.* An improved predictive recognition model for cys2-his2 zinc finger proteins. *Nucleic Acids Res* 2014;**42**:4800–12.

Hammal F, de Langen P, Bergon A *et al.* Remap 2022: a database of human, mouse, drosophila and arabidopsis regulatory regions from an integrative analysis of DNA-binding sequencing experiments. *Nucleic Acids Res* 2022;**50**:D316–25.

Heide F *et al.* Fast and flexible convolutional sparse coding. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA*, 2015. pp. 5135–5143.

Heinz S, Benner C, Spann N *et al.* Simple combinations of lineage-determining transcription factors prime cis-regulatory elements required for macrophage and b cell identities. *Mol Cell* 2010;**38**:576–89.

Hertz GZ, Stormo GD. Identifying DNA and protein patterns with statistically significant alignments of multiple sequences. *Bioinformatics (Oxford, England)* 1999;**15**:563–77.

Hinton GE *et al.* Learning distributed representations of concepts. In: *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*, Amherst, MA, Vol. 1, 1986, p. 12.

Hubley R, Finn RD, Clements J *et al.* The dfam database of repetitive DNA families. *Nucleic Acids Res* 2016;**44**:D81–9.

Ichikawa DM, Abdin O, Alerasool N *et al.* A universal deep-learning model for zinc finger design enables transcription factor reprogramming. *Nat Biotechnol* 2023. pp. 1–13.

Kelley DR, Snoek J, Rinn JL *et al.* Basset: learning the regulatory code of the accessible genome with deep convolutional neural networks. *Genome Res* 2016;**26**:990–9.

Khorasanizadeh S, Rastinejad F. Nuclear-receptor interactions on DNA-response elements. *Trends Biochem Sci* 2001;**26**:384–90.

Li M *et al.* Finding similar regions in many strings. In: *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing, Atlanta, GA*, 1999. pp. 473–82.

Liu JS, Neuwald AF, Lawrence CE *et al.* Bayesian models for multiple local sequence alignment and gibbs sampling strategies. *J Am Stat Assoc* 1995;**90**:1156–70.

Lundberg SM, Lee S-I. A unified approach to interpreting model predictions. *Adv Neural Inform Process Syst* 2017;**30**. p. 4766.

Monga V, Li Y, Eldar YC *et al.* Algorithm unrolling: interpretable, efficient deep learning for signal and image processing. *IEEE Signal Process Mag* 2021;**38**:18–44.

Najafabadi HS, Albu M, Hughes TR *et al.* Identification of c2h2-zf binding preferences from chip-seq data using rcade. *Bioinformatics* 2015; **31**:2879–81.

Pratt HE, Andrews GR, Phalke N *et al.* Factorbook: an updated catalog of transcription factor motifs and candidate regulatory motif sites. *Nucleic Acids Res* 2022;**50**:D141–9.

Shrikumar A *et al.* Learning important features through propagating activation differences. In: *International Conference on Machine Learning, Sydney, Australia*, PMLR, 2017. pp. 3145–53.

Siggers T, Gordân R. Protein–DNA binding: complexities and multiprotein codes. *Nucleic Acids Res* 2014;**42**:2099–111.

Touzet H, Varré J-S. Efficient and accurate p-value computation for position weight matrices. *Algorithms Mol Biol* 2007;**2**:1–12.

Wang T, Stormo GD. Combining phylogenetic data with co-regulated genes to identify regulatory motifs. *Bioinformatics* 2003;**19**: 2369–80.

Weirauch MT, Yang A, Albu M *et al.* Determination and inference of eukaryotic transcription factor sequence specificity. *Cell* 2014;**158**: 1431–43.

Wohlberg B. Efficient algorithms for convolutional sparse representations. *IEEE Trans Image Process* 2016;**25**:301–15.

Yuan H, Kelley DR. Scbasset: sequence-based modeling of single-cell atac-seq using convolutional neural networks. *Nat Methods* 2022; **19**:1088–96.

Zhuang J *et al.* Adabelief optimizer: adapting stepsizes by the belief in observed gradients. *Adv Neural Inform Process Syst* 2020;**33**: 18795–806.

Zuo Z, Billings T, Walker M *et al.* On the dependent recognition of some long zinc finger proteins. *Nucleic Acids Res* 2023 (in press).