MACHINE LEARNING
Science and Technology

PURPOSE-LED PUBLISHING™

PAPER • OPEN ACCESS

An extended de Bruijn graph for feature engineering over biological sequential data

To cite this article: Mert Onur Cakiroglu *et al* 2024 *Mach. Learn.: Sci. Technol.* **5** 035020

MACHINE LEARNING
Science and Technology

OPEN ACCESS

RECEIVED
17 March 2024

REVISED
15 June 2024

ACCEPTED FOR PUBLICATION
5 July 2024

PUBLISHED
19 July 2024

PAPER

An extended de Bruijn graph for feature engineering over biological sequential data

Mert Onur Cakiroglu[1], **Hasan Kurban**[1,2,3,6,*], **Parichit Sharma**[1], **M Oguzhan Kulekci**[1], **Elham Khorasani Buxton**[4], **Maryam Raeeszadeh-Sarmazdeh**[5] and **Mehmet M Dalkilic**[1,2]

[1] Computer Science Department, Indiana University, Bloomington, IN, United States of America
[2] Data Science Program, Indiana University, Bloomington, IN, United States of America
[3] Electrical and Computer Engineering, Texas A&M University at Qatar, Doha, Qatar
[4] Computer Science Department, University of Illinois, Springfield, IL, United States of America
[5] Chemical and Materials Engineering, University of Nevada, Reno, NV, United States of America
[6] College of Science and Engineering, Hamad Bin Khalifa University, Doha, Qatar
[*] Author to whom any correspondence should be addressed.

E-mail: **hasan.kurban@tamu.edu**, **meocakir@iu.edu**, **parishar@iu.edu**, **okulekci@iu.edu**, **esahe2@uis.edu**, **maryamr@unr.edu** and **dalkilic@iu.edu**

Keywords: de Bruijn graph, machine learning, bioinformatics

Supplementary material for this article is available online

Abstract

In this study, we introduce a novel de Bruijn graph (dBG) based framework for feature engineering in biological sequential data such as proteins. This framework simplifies feature extraction by dynamically generating high-quality, interpretable features for traditional AI (TAI) algorithms. Our framework accounts for amino acid substitutions by efficiently adjusting the edge weights in the dBG using a secondary trie structure. We extract motifs from the dBG by traversing the heavy edges, and then incorporate alignment algorithms like BLAST and Smith–Waterman to generate features for TAI algorithms. Empirical validation on TIMP (tissue inhibitors of matrix metalloproteinase) data demonstrates significant accuracy improvements over a robust baseline, state-of-the-art PLM models, and those from the popular GLAM2 tool. Furthermore, our framework successfully identified Glycine and Arginine-rich motifs with high coverage, highlighting it is potential in general pattern discovery.

1. Introduction

A recent paper estimates that there are as many as 10^{12} microbial species alone on our planet [1]. This profoundly large number remains, certainly, a lower bound on all organisms that are *universally* related in some general way through information biomolecules: DNA, RNA, and proteins (figure 1 Left top). Interestingly, these biomolecules can be described by their primary (linear) form as a string or, in the parlance of biology, a sequence over small alphabets. A portion of DNA that is transcribed into RNA and perhaps forms either the final or intermediary template of a protein is called a gene [3]. By identifying and understanding genes, humankind can understand life. From an artificial intelligence (AI) perspective, genes can be considered as a kind of feature. Used in an unsupervised approach, clustered genes can be used to build a taxa that might be used to better understand evolutionary processes (figure 1 Left bottom). From a supervised approach, genes might elucidate function through pathways [4].

In section 2, we elaborate on a complex pipeline for engineering these features. Figure 1 (Right) shows two of possibly many different engineered features obtained from sequence alignment. Here, features are built as a kind of consensus string. Unlike features that are used to train machine learning models, these features are *operationally* related through a threshold on an extended edit distance (in its original formulation NP-complete), and used to build disparate (possibly gapped) substrings called motifs. Gene data, in fact, is the one of the most prevalent biological data currently available. In humans, genes exist in

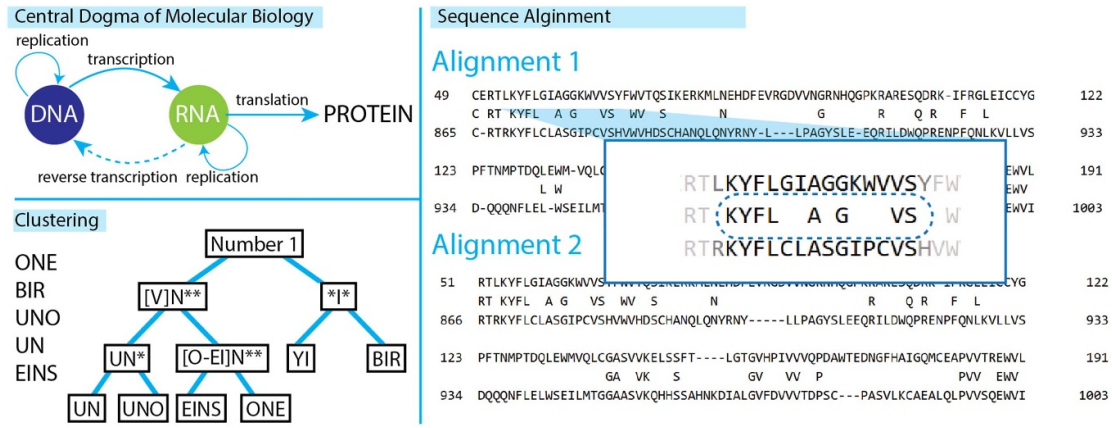

Figure 1. (Left top) Central Dogma of Molecular Biology shows the flow of information (dotted line shows exceptional circumstances). (Left bottom) Example from linguistic for the word 'one' using Spanish, French, German, English, Chinese, and English. 'V' stands for a family of similar vowel and * for any symbol. The taxa that results is consistent with current linguistic families. (Right) An example [2] for feature discovery as a motif. Sequence alignment is a pipeline whose parameters can vastly change the outcome. Dashes denote insertion of blank space. The blank spaces actually denote *gaps* which can represent either no significant overlap, spacing requirements, or even impossible matching e.g. match a space with a character.

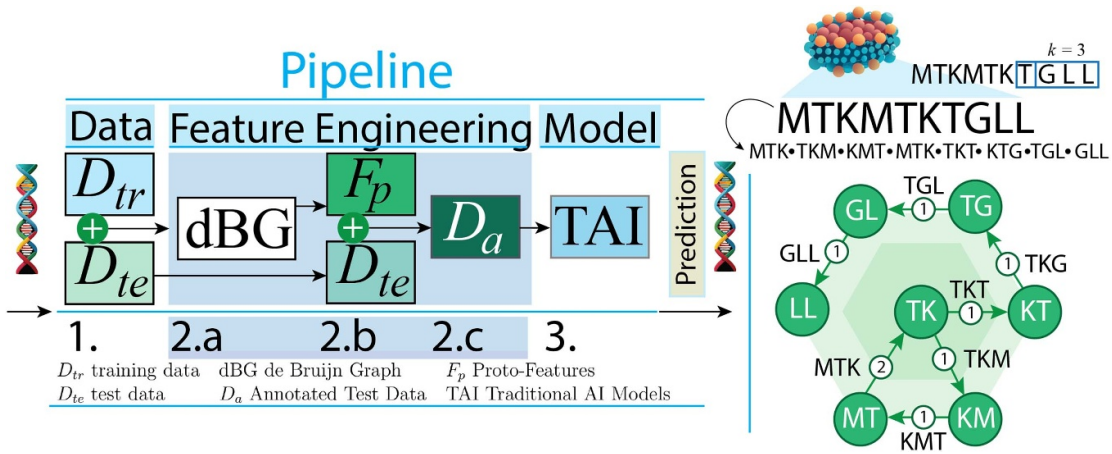

Figure 2. (Left) Schematic Overview of the Feature Engineering Pipeline for Sequence Data. Initially, both training and test datasets are inputted into the feature engineering block (Step 1). Subsequently, a de Bruijn Graph (dBG) is constructed from the training data, facilitating the generation of proto-features (F_p) (Step 2.a). These proto-features, along with the test data, are utilized to endow the test data with numerical features (Step 2.b). The is followed by the application of a Traditional Artificial Intelligence (TAI) model(s) (Step 3), culminating in classification. The process demonstrates the efficacy of translating sequence data into actionable insights for TAI applications. (Right) Illustration of dBG construction steps over small protein. We iterate over a sequence with a sliding window of width $k = 3$ generating a series of unique tuples. These tuples are transformed into a weighted directed graph where the edge weights represents the occurrence count of the corresponding tuple, as illustrated on the bottom.

sizes ranging from hundreds to millions of characters and the average length is in the range of 2^{14} characters. When treated as conventional features, motifs present the same problems, as encountered with traditional features: apparently arbitrary lengths, characters, non-transitive distance and so on.

This work focuses on how traditional AI (TAI) can be partnered with unstructured data, specifically, functional or non-functional biomolecules, for binary classification. The principle challenge is that no apparent features exist. To that end, the problem studied here is: given a set of unstructured data representing biomolecules, efficiently construct a feature space that, when added to the data, can be used to train downstream TAI algorithms.

More specifically, we employ an extension of the de Bruijn graph (dBG) engineering annotated features providing a simpler, more efficient and effect pipeline (figure 2 Left).

The key contributions of this work are:

- A novel, automated and efficient pipeline that dynamically discover motifs and builds a feature space, in which TAI algorithms can be used.
- A novel implementation of the extended de Bruijn (dBG) that allows efficient path discovery.
- Experimental demonstration that feature space is accurate over several disparate TAI algorithms, and
- Empirical analysis to showcase robustness, scalability, and modularity of this approach.

Rest of the paper is organized as follows: section 2 provides a comprehensive background, setting the stage for our investigation by reviewing relevant literature to situate our work within the current research landscape. Section 3 describes the methods employed in our study, detailing the algorithmic approaches and computational techniques foundational to this work. In section 4, we report the experimental results obtained, offering a thorough analysis to elucidate the efficacy and implications of our findings. The paper concludes in section 5, where we synthesize the main insights, discuss the broader impacts of our contributions, and propose directions for future research.

2. Background and related work

Motif discovery (MD), the process of discovering significant patterns in unaligned sequence data, is complex and context dependent yielding a divers array of techniques that currently yields no universally successful approach (see [5] for a review). MD can be treated as a kind of optimization problem over the space of an augmented alphabet. There must be included a scoring scheme that quantifies matches, mismatches, and penalties for gaps. Generally this is done with a matrix that describes biological relationships using probabilities and various post-processing to capture changes due to time. Two general approaches exist: probabilistic where distributions of n-ary substrings are used–called position-specific scoring matrix (PSSM) or combinatorial where various enumerations of substrings are used. Most variants of motif finding are NP-hard and, therefore, solutions are approximations. While probabilistic approaches are usually associated with subtle discovery, combinatoric approaches are generally more efficient algorithmically. Several approaches are well-known throughout the community [6–12].

We now discuss existing work that is directly relevant to the literature of MD. Multiple EM for Motif Elicitation (MEME) [13] is based on the Expectation Maximization [14] and now a collection of tools [15] that includes GLAM2 [16], DREME [17], MEME-ChIP [18] along with other useful tools. GLAM2 is a computational tool designed for the discovery of gapped motifs in a set of DNA or protein sequences. It considered as extension of the original MEME algorithm. Unlike MEME, GLAM2 does not pursue the simultaneous detection of multiple motifs. Instead, it employs a replication procedure, whereby it conducts repeated attempts to identify the most optimal motif. DREME is a discriminative tool that efficiently discovers short, ungapped DNA motifs in large collections of sequences. It is particularly useful for identifying transcription factor binding sites in ChIP-seq data. While DREME uses a computationally efficient approach, it only identifies motifs that can be expressed in the IUPAC alphabet, which includes the standard DNA alphabet and eleven wildcard characters. The Gibbs Sampling [19] is an old but powerful Markov Chain Monte Carlo algorithm used for pattern discovery. Iteratively sampling from the conditional probability distribution of each variable yields patterns, though it can be computationally intensive to run. PRATT [20] uses a flexible pattern representation and a greedy algorithm to identify patterns. PRATT performs a branch-and-bound search and other heuristics to speed up the search.

While we were unable to find any comparable work to our own, the bulk of feature engineering relies on information extrinsic to the simple sequence itself. The taxonomy of feature engineering methods for biological sequences can be broadly classified into five categories. (I) Sequence-based features encompass methods such as n-gram analysis, which breaks sequences into continuous overlapping subsequences of size n [21], or skip-grams, which are non-contiguous sequences that skip k items [22]. Another approach includes composition, measuring chemical composition, e.g. GC content in DNA [23]. Complexity measures like entropy or LZ complexity are used to identify significant regions that likely are biologically important [24]. Combinatoric k-mer analysis counts the occurrences of all possible subsequences of length k in the sequence, examining overrepresented totals [25]. (II) Structure-based methods consider conformation (2D, 3D, 4D) topologies determined by the original sequence [26]. For example, the 3D tertiary structure representation of proteins relies on the protein backbone, which consists of a sequence of atoms, including the alpha carbon, nitrogen, and carbonyl carbon atoms. This backbone forms the core framework of the protein and can be used to describe the protein structure [27, 28]. There have also been studies that extract features using the wavelet representation of proteins, which are visual representations that capture spatial

and frequency information of the protein structure [29]. (III) Evolutionary features incorporate methods that look at the evolutionary aspects of the sequences [30]. For example, evolutionary info has been used to locate biologically relevant interfaces of the proteins [31]. PSI-BLAST utilizes a position-specific scoring matrix [32] to identify distant evolutionary relationships. (IV) Functional features take into account the function of the sequences. These include gene ontology features providing information about the biological processes [33], cellular components, and molecular functions associated with a sequence; pathway analysis features looking at metabolic or signaling pathways; protein-protein interaction; or gene regulatory networks. (V) AI/ML includes autoencoders [34] and deep learning [35], recurrent neural networks (RNNs) [36] where order matters, and transformer models, which are advanced models that leverage self-attention mechanisms to better understand the context of a sequence.

3. Methods

Here, we describe the general notation used in this work. D_{tr}, D_{te} are the training sequences, test sequences data, respectively. Both D_{tr} and D_{te} includes a feature $\{1,0\}$ indicating functioning or non-functioning. The annotated data (D_a) is constructed by aligning proto-features (F_p) with the input sequences. Let $\mathcal{A} = \{a_1, a_2, \ldots, a_n\}$ be an alphabet of size n. $\mathcal{A}^k$ for positive integer $k \in \mathbb{Z}_+$ is the set of words of length $|s| = k$. A word is a string of characters defined in $A : w = \{c_1, c_2, \cdots, c_k\}$, where $c_i \in \mathcal{A}$. The concatenation of two words $s, t \in \mathcal{A}$ is $st = s[0]\cdots s[|s|-1]t[0]\cdots t[|t|-1]$. A prefix of s is $s[0]\cdots s[j]$ and a suffix is $s[j]\cdots s[k-1]$ for some $0 \leqslant j < k$. The order-k dBG of a collection of strings D (data) over $\mathcal{A}$ is a directed multigraph where $V = \mathcal{A}^{k-1}$ and $E = \{(s,t)|s[0]t = st[k-2]\}$. Below, we explain the feature engineering pipeline (figure 2 Left), and dBG construction is illustrated in figure 2 (Right).

Step 1. Both D_{tr}, D_{te} are fed to the FEB. D_{tr} is used to build the dBG.

Step 2.a. The dBG, built from D_{tr}, is extended in two ways: adding weights and additional nodes. A weight maps E to a non-negative integer $\nu \in \mathbf{Z}_{\geqslant 0}$. In this study we explore extending the search space by adding weights. We define the term 'k-tuple' as a subsequence of length k extracted from a larger set sequences (D_{tr} in our case). As described before, the dBG consists of overlapping k-tuples represented by weighted edges, where the nodes are constructed from the (k-1)-tuples. The k-tuples contain (k-1)-tuple of the source node as the prefix and (k-1)-tuple of the target node as suffix in an overlapping way. When a k-tuple is encountered for the first time, add an edge to the dBG with weight 1, any subsequent occurrences of that tuple will increment the weight by 1.

The parameter k significantly influences the structure of the resulting graph, as different values can lead to markedly distinct graphs. Specifically, smaller k values tend to produce denser graphs with a reduced number of nodes because given the finite alphabet, this limits number of possible k-tuples. This approach may limit the identification of longer, potentially meaningful patterns. On the other hand, larger k values yield sparser graphs with an increased node count, due to a greater variety of possible k-tuples and subsequently, less overlapping edges. This condition allows the detection of complex patterns. Yet, if k is set too high, it might prevent edge overlap entirely, diminishing the graph's connectivity and pattern recognition capability. The optimal value cannot be determined *a priori* and is a function of $|\mathcal{A}|$ as well as the complexity of $D_{tr} \cup D_{te}$.

3.1. Substitution approximation

Substitution approximation is an optional step that enhances the representational power of the dBG by adjusting the edge weights [37]. Edge weights are adjusted using the substitution matrix (SB) that refine edit distance by changing equality {0,1} to $\mathbb{R}_{\geqslant 0}$ where increasingly better matches have greater values. Let $\mathbf{M}$ be a SB matrix (BLOSUM62 [38] in this work), and let $e, f \in E$; e, f are similar if, for some $\theta \geqslant \mathbb{R}_{\geqslant 0}$, $\mathbf{M}[e[i], f[i]] > \theta$ for $1 \leqslant i \leqslant |e|$. The value of θ is 0 by default, but is adjusted according to properties of the data and need. To measure similarity between edges, we can use relative similarity score. A relative similarity score between e, f can be computed as:

$$sim(e,f) = \sum_{i=1}^{k} \frac{M[e[i], f[i]]}{M[e[i], e[i]]}. \tag{1}$$

To take into account similarities, edge weights are incremented proportionally to other similar edges. Let $e \in E$, w_e be the edge weight, $\mathsf{E}_{e,\tau} = \{d \in E | sim(e,d) \geqslant \tau, \tau \in \mathbb{R}_{\geqslant 0}\}$ be the set of similar edges to e for some threshold τ. The new edge weight of e is

$$w_e \leftarrow w_e + \kappa \sum_{f \in \mathsf{E}_{e,\tau}} w_f \cdot sim(e,f) \tag{2}$$

Algorithm 1: FindPatterns

input : *Graph g, Similarity_Constant K, Threshold_Coefficient C*
output : *HashMap o*

```
1  Function FindPatterns (g, K, C):
2      T ← g.maxEdge.weight × C
3      o ← HashMap(string, int)
4      while g.maxEdge.weight ⩾ T do
5          s ← Traverse(g.maxEdge, K, T)
6          if s in o then
7              sup ← o.get(s)
8              o.add(s, sup + 1)
9          else
10             o.add(s, 1)
11     return o
```

where κ allows changing the overall edge similarity effects. If $\kappa = 0$, the extended dGB and classical dGB are identical. The construction is complete once all weight updates have been computed.

Step 2.b. In this step, the dBG generates proto-features F_p. Weights reflect pattern frequency, i.e. relevance. The *findPatterns* algorithm (algorithm 1), shows the general steps. The *Traverse* function discovers frequently occurring patterns with support. Next, we provide the general flow of steps followed by the algorithm:

(i) Identify the heaviest weight in the dBG and set it as the starting point for traversal (Line 2 on algorithm 1).
(ii) Begin traversal of the dBG. At each branch, choose the heaviest available edge that has not been traversed and exceeds a specified weight threshold. The traversal concludes when no eligible edges remain (*Traverse*).
(iii) Go back to the starting edge and reverse the traversal direction to incorporate incoming edges (*Traverse*).
(iv) Reduce the weights of all traversed edges by 1. Additionally, adjust the weights of edges similar to the traversed edges, decreasing them proportionally to their relative similarity score ($\kappa \times sim(t, s)$), where t is a traversed edge and s is a similar edge to t (*Traverse*).
(v) Record the tuple of traversed paths into F_p. (Line 6–10 on algorithm 1).

Search begins with greatest weight and $C \in [0, 1]$ that determines the significance of weight *w.r.t* the threshold T. Smaller C values yield more and longer patterns, but contain increasing false positives; conversely, greater C values have the opposite effect. The *Traverse* algorithm reduces the weights of traversed edges, continuing until all $w < T$. Algorithm 1 starts from the greatest weight and traverses both forward and backward. While traversing it chooses heaviest adjacent edge. The forward traversal process extends the pattern by moving along outgoing edges, expanding it from the end, while the backward traversal is progresses by following incoming edges, thereby growing the pattern from the start. After threshold T is met on both ends, the weights of all visited edges are decremented producing a sequence from the path. With F_p the next steps create the individual proto-features or motifs.

For each motif, we calculate the alignment score using Smith–Waterman (SW) [39] and BLAST [40, 41] $\mathcal{O}(|s| \cdot |t|)$ time for both. SW proves highly effective in our experiments when working with sequences of equal lengths; however, the results from the SW algorithm are not normalized. When considering sequences with a diverse range of sizes (which is the real-word scenario most of the time), this approach introduces size bias, wherein shorter sequences exhibit smaller features and vice versa. To mitigate this, we applied our own normalization that aims to normalize the selected columns based on the relative importance of the values within each row, while preserving the scale dictated by the median of the row. Let $\mathbf{F}$ be a $\mathbb{R}^{|D_{te}| \times |F_p|}$ matrix where $\mathbf{F}[i,]$ represents the features of $s_i' \in D_a$ and $\mathbf{F}[,j]$ represents a column for scores for a particular $t_j \in F_p$. Then each value in $\mathbf{F}$ is normalized by:

$$x_i = \sum_{j=1}^{|F_p|} \mathbf{F}[i,j] \tag{3}$$

$$\mathbf{X} = \left[x_{i_1} x_{i_2} \cdots x_{i_{F_p}} \right]^t \tag{4}$$

$$\mathsf{m}(\mathbf{X}) = \begin{cases} x_{(n+1)/2} & n \text{ is odd} \\ \frac{1}{2}\left(x_{n/2} + x_{n/2+1}\right) & o.w. \end{cases} \tag{5}$$

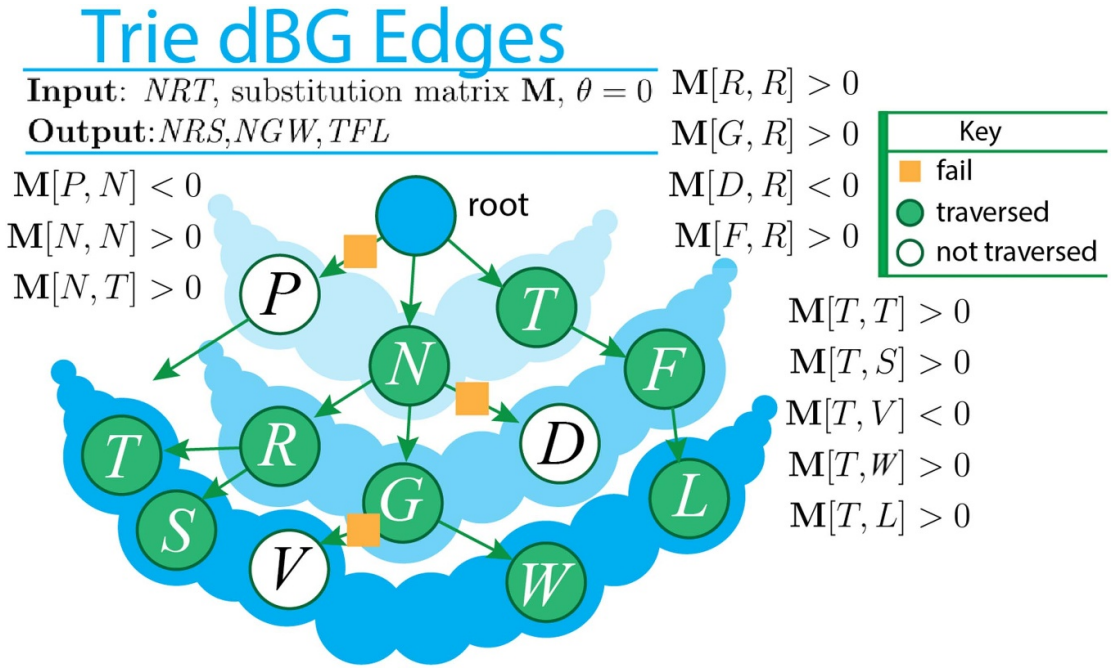

Figure 3. Illustration of trie-based search for identifying edges analogous to *NRT* in a de Bruijn Graph (dBG) using a mock substitution matrix **M**. Beginning at the root, the search progresses through trie branches that meet the substitution score threshold ($\theta = 0$) at each index. The search culminates at leaf nodes representing sequences that, collectively, correspond to dBG edges similar to the input edge *NRT*, excluding the input edge itself from the resultant set. This process exemplifies the efficient filtering of edges based on substitution scores.

$$\mathbf{F}[i,j] \leftarrow \mathsf{m}(\mathbf{X}^t)\,\frac{\mathbf{F}[i,j]}{x_i}. \tag{6}$$

BLAST is among the most popular tools for alignment, leveraging a database of sequences. Using *blastp* the resulting normalized bitscores (overall alignment quality) are treated as features. The highest E-value (number of expected hits of similar quality) is used. For multiple matches per sequence-motif pair, the best alignment (lowest E-value) is selected. Although, this method proved to be sufficient in smaller data sets, larger data sets, on the other hand, were too sparse to be effective.

Step 2.c. Here for each $s \in D_{te}$, a datum is constructed: $s' \in D_a$ includes both the sequence ID and label $\{0,1\}$ from s with a vector of scores $\sigma_1, \ldots, \sigma_{|F_p|}$ for $\sigma \in \mathbb{R}_{\geqslant 0}$ from aligning s to each $t \in F_p$.

3.2. Complexity details

Construction of the dBG is $\mathcal{O}(n\ell)$ where $|D_{tr}| = n$ and $\ell = \sum_i |s_i|$ for $s \in D_{tr}$. The space complexity is $\mathcal{O}(E+V)$. The most computationally heavy part of the approximation algorithm is finding similar edges. A trie is built from edges that allow for the early elimination of non-similar edges during construction. The structure also allows multi-trie search based on substitution scores (figure 3). Algorithm 2 defines a search algorithm that identifies tuples that are similar to a specified (key) tuple using this trie structure. The search algorithm starts at the root of the trie (defined in line 3) and compares the similarity scores between every possible trie path and the corresponding key index at lines 8–9. The algorithm only traverses the edges that fit the θ similarity constraint, reducing the number of edges that needed to be checked at every index. Building the trie takes $\mathcal{O}(Ek)$ time where k is the tuple length and $\mathcal{O}(m)$ space. Searching through similar edges for every edge in the graph takes $\mathcal{O}(Ekm)$ time, where m ($m \leqslant \ell$) is the edge count in the trie. The next step traverses the graph to extract proto-features. We can find the heaviest edge in $\mathcal{O}(\log E)$ time by maintaining a heap. In total, a single traversal of the graph takes $\mathcal{O}(E \log E)$ time. The number of iterations adds only a small constant and can be ignored. In total, the entire process takes $\mathcal{O}(n\ell + E(km + E\log E))$ time and $\mathcal{O}(E+V+m)$ space, which is quite efficient.

Algorithm 2: FindSimilar

input : *tupleKey*, *sub*, θ, *current* = `NULL`, *index* = 0
output : Set of similar tuples to *sub*

```
Function FindSimilar(tupleKey, sub, θ, current, index):
    if current = NULL then
        current ← root
    if current.leaf then
        return {(tupleKey, current.weight)}
    similar ← ∅
    for each child in current.children do
        score ← sub.getScore(tupleKey[index], child.char)
        if score ⩾ θ then
            newKey ← tupleKey[: index] + child.char + tupleKey[index + 1 :]
            childTuples ← FindSimilar(newKey, sub, θ, child, index + 1)
            similar ← similar ∪ childTuples
    return similar
```

4. Data and overview of experiments

4.1. Dataset details and background

We evaluate the **dBG** feature extraction technique by training TAI models on features extracted from TIMP (tissue inhibitors of matrix metalloproteinase (MMPs)) data [42]. TIMP data is widely used for finding variants (sequence with mutations) that bind to MMPs, thus potentially regulating MMP functionality [43]. However, the data is relatively small and consists of only 307 protein sequences, of which the majority (289) are functional. The skewed class distribution induces class imbalance, and makes it difficult to train models with traditional handcrafted features. To overcome this, researchers are starting to use protein language models (PLMs) as a first step to extract protein embedding (vector representation of protein sequences on continuous scale) [44]. Downstream tasks further perform dimension reduction on protein embedding to extract low-dimensional features used for training TAI models.

Although, the approach has shown promise in building better models, it requires a dimension reduction on protein embeddings that runs in the 1000s. In contrast, our dBG workflow can extract a small set of high-quality motifs in a short amount of time. More specifically, our classification experiments on TIMP data are run on only 34 motifs. Moreover, in comparison to the traditional, but widely used tool like GLAM2, dBG offers some unique advantages. For instance, GLAM2 requires the exact number of consensus motifs for the program to work. In contrast, dBG does not mandate the user to specify number of motifs. Instead, a default weight threshold is used to extract all motifs above the threshold. Also, unlike GLAM2, dBG extracts non-gapped motifs. This is important because alignment algorithms like BLAST cannot work with gaps, forcing the use of the SW algorithm, which can process gaps but is slower. Moreover, experimental results (section 4) indicate that, when class distribution is balanced, both dBG and GLAM2 have good performance. However, on data with disproportionate class distribution, models trained with dBG motifs performs reasonably better than GLAM2. Taken together, dBG, GLAM2 and PLMs significantly improves the performance of novel MD algorithms, with dBG showing early promise for adapting to data with skewed class distribution.

We train several TAI classifiers on the extracted motifs and show that dBG extracted motifs outperforms the classification models shared in a recent study on TIMP data [44]. Furthermore, while PLM extracts features that are black box to biologists, motifs generated by our feature engineering pipeline are interpretable and allows alignment with biologist domain knowledge and intuition. The extracted features can be presented to biologists, enabling them to identify, validate and fine-tune biologically relevant features, thereby completing the feedback loop and improving the overall accuracy and relevance of the results

4.2. Overview of experiments

I. Comprehensive study of TAI models We train 166 [45] different classification models, including logistic regression, tree and ensemble-based models, random forest, and deep learning models, to showcase the suitability of dBG motifs for classifying TIMP sequences. Two alignment algorithms, SW and BLAST, are used to check whether motifs are biased *w.r.t* the algorithm. For robust evaluation, we perform two different experiments, and compare dBG motifs with a random baseline, and also the state of the art in sequence based motif extraction i.e. embeddings based on PLMs [44] and GLAM2 motifs. Since, PLM embeddings are

shown to outperform traditional handcrafted and structure/biophysical features (k-mer, n-gram) [46], so we only consider GLAM2 motifs.

In the first experiment, we mitigate the impact of highly skewed class distribution by up sampling the low-frequency class, followed by model training on train set, and prediction on the test set. For comparison, we construct a baseline by extracting random motifs of the same length as the dBG motifs from each sequence, and generating the count matrix for the random motifs. The process is repeated 100 times, and the final count matrix is taken to be the average of 100 trials. The second experiment aims to replicate the difficult yet more frequently encountered case, i.e. prediction under skewed class distribution. In this case, we evaluate the models trained on dBG motifs with PLM based embedding and GLAM2 motifs. Additional results are found in the supplementary materials.

II. Applying dBG beyond classification As an extension of dGB, we use dBG to find Glycine-Arginine (GA) rich motifs, and verify our findings with results published in a recent study [47]. We find that dBG motifs have reasonable overlap with the motif reported in [47]. This use case provides an early validation of dBG approach as a more general framework for feature extraction.

Implementation note In experiment I and II, we use k-fold cross validation ($k = 5$) to partition the data and train models on each fold. Additionally, 5 trials are done, and model hyperparameters are selected based on accuracy on the validation fold across the trials. To address the skewed class distribution, minority class is up sampled [48] and distribution of functional and non functional sequences is balanced. In experiment II, up sampling was done only on train set, and test set was kept as usual. This helps to evaluate performance under a more practical scenario of encountering skewed class distribution. BLAST alignments are not possible for GLAM2 because it produce gapped motifs, so only SW alignment is done for GLAM2 motifs.

5. Results

5.1. Training TAI models on dBG generated features

Figures 4 and 5 demonstrate the results of the top 10 models (out of 166) that performed best in classifying the sequences.

5.1.1. Prediction under balanced class distribution

The results show that the models trained on dBG are significantly better than models trained on randomly sampled motifs. For the dGB motifs, the top models have balanced accuracy and an F1 score of 1, indicating perfect classification on test data. For random motifs, though models show good performance on validation data, quality degrades on test data due to over fitting and lack of generalization. Tree-based models emerge as the best classifiers for dBG motifs; for random motifs, deep neural networks performed the best. Observe that performance is unaffected across SW and BLAST, possibly indicating the robustness of dBG motifs to different alignment algorithms. Notably, as shown in table 1, models trained on dBG and GLAM2 motifs have significantly better classification accuracy than current state of the art that reported a mean AUC-ROC of $0.80 - 0.95$ from the best model trained via a low-dimensional representation of protein embedding [44]. The result show that, when proportion of classes is balanced, dBG and GLAM2 motifs do better in identifying specific motifs characteristic of functional versus non-functional TIMP sequences.

5.1.2. Testing in the wild with skewed class distribution

Table 2 compares the classification performance of top models on test set in the wild. Result show that dBG and GLAM2 motifs outperform PLM based feature embeddings. Specifically, models trained on dBG motifs did notably better than those trained on GLAM2 motifs. The performance of dBG motifs is evidently better across different types of models for example, random forest, logistic regression etc. Moreover, performance improvement is conserved in BLAST and SW based alignments, whereas GLAM2 can only be aligned via SW. The analysis emphasize that dBG can provide a reliable feature extraction framework with minimal supervision, and performs well under best and worst case scenarios.

5.2. Extending the dBG to extract GAR motifs

In addition to using dBG features for training TAI models, we wanted to study the application of dBG as a broader feature mining framework. To observe it's usefulness, we employ dBG for finding Glycine and Arginine-rich (GAR) motifs; the objective is to study the overlap between the set of motifs discovered by dBG and the existing baselines for finding GAR motifs [49]. We refer to a recent work in this area [47] called glycine motif finder (GMF), which is designed to find GAR-rich motifs. Note that GMF is fine-tuned for a

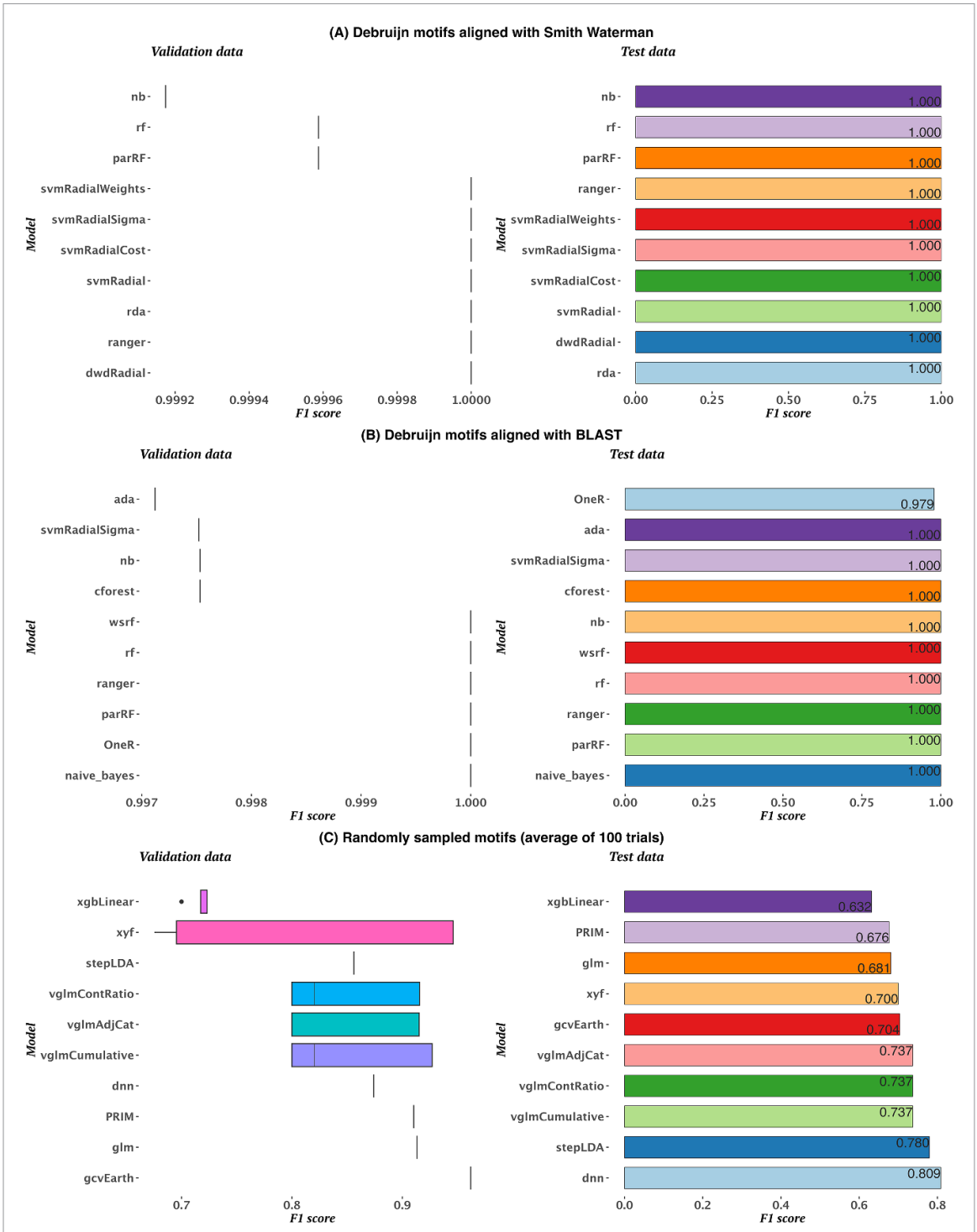

Figure 4. Comparing the F1 score of models built with dBG versus randomly sampled motifs from TIMP data. Models trained on dBG motifs have significantly better classification accuracy.

specific purpose, but dBG provides a more general approach. The same sequences as used in [47] are used as input to dBG for extracting the motifs, and results are given in table 3.

Since, dBG depends on a SB matrix for expanding the graph search space by adjusting the weights, we used different SB matrices (BLOSUM45, 50, 62, 80, 90 and PAM 30, 70, 250) to evaluate the sensitivity of different matrices for finding GAR motifs. Additionally, a baseline was constructed by setting $\kappa = 0$ (no approximation) for comparison with the results obtained by using approximate dBG search via different SB matrices. We observed that matrices BLOSUM80, BLOSUM90, and PAM30 are noticeably effective in detecting a higher number of GAR motifs, as reported by [47]. Interestingly, the baseline dGB ($\kappa = 0$) is able

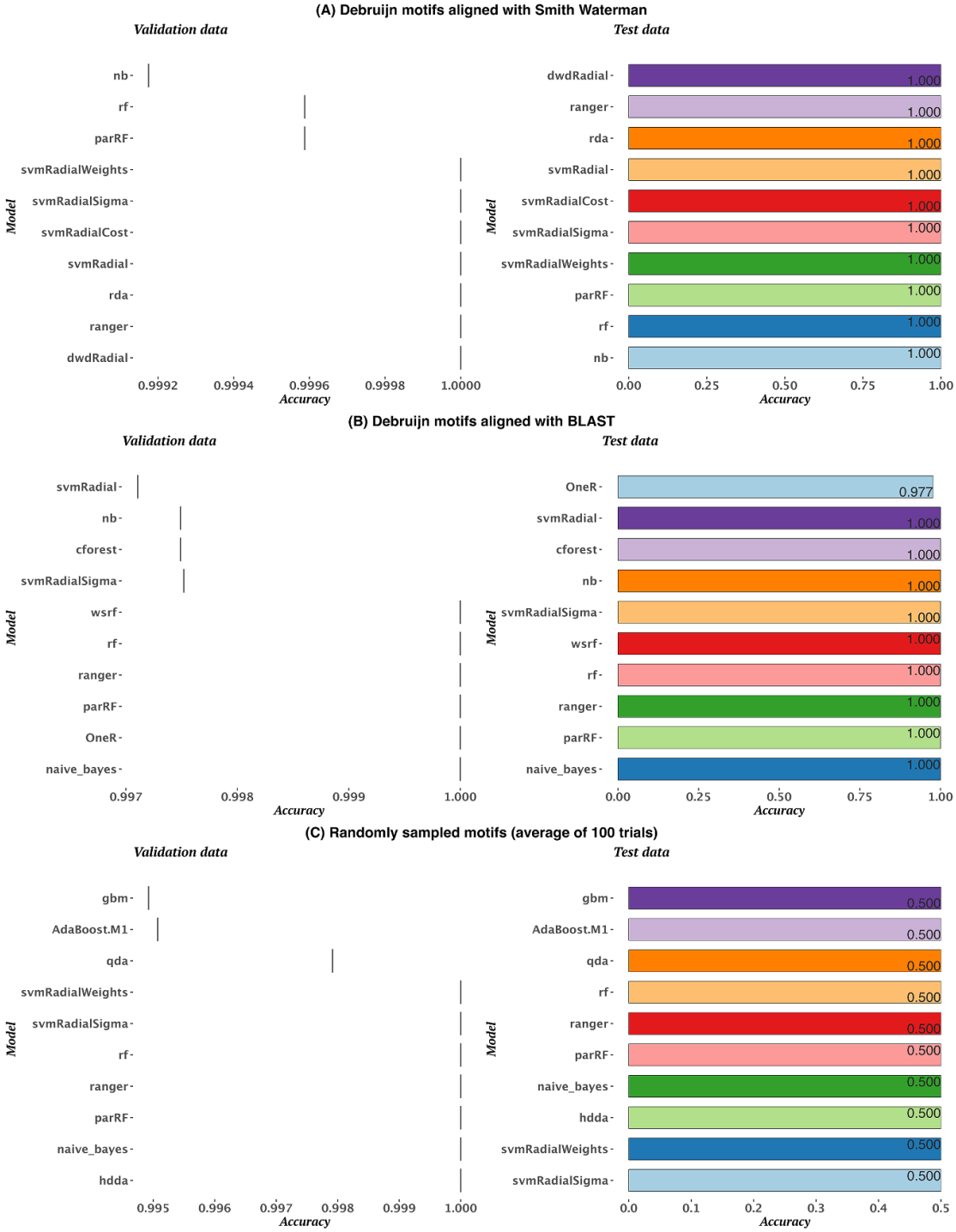

Figure 5. Comparing the balanced accuracy of models trained on dBG motifs versus randomly sampled motifs (of same length) from TIMP data. dBG motifs surpass prediction accuracy of models trained on randomly sampled motifs.

to detect GAR motifs, but to a lesser degree, which shows that the dBG approach assisted by approximate search is better for finding GAR motifs. In table 3, '#Total Motifs' denote total number of motifs found by dBG, '#GMF Motifs' denotes the number of GAR motifs (found by GMF) [47]. Among the different matrices, BLOSUM90 performed the best by finding the most GAR motifs while minimizing the false positive rate. Both PAM30 and BLOSUM80 discovered the same number of GAR motifs. However, PAM30 did not yield any non-GAR patterns.

Table 1. (Experiment 1) Comparison of best models (trained on dBG motifs) with SOTA PLM and GLAM2.

(A) SW aligned dBG motifs

Validation Data			Test Data	
Model Name	Model Type	AUC-ROC	Model	AUC-ROC
svmRadialCost	SVM with Radial Basis Function	1	svmRadialCost	1
svmRadial	SVM with Radial Basis Function	1	svmRadial	1
ada	Regularized Discriminant Analysis	1	ada	1
ranger	Random Forest	1	ranger	1
dwdRadial	Discriminant with Radial Basis Function	1	dwdRadial	1
LogitBoost	Logistic Regression	1	LogitBoost	1

(B) Blast aligned dBG motifs

Validation Data			Test Data	
Model Name	Model Type	AUC-ROC	Model	AUC-ROC
wsrf	Weighted Subspace Random Forest	1	wsrf	1
rf	Random Forest	1	rf	1
ranger	Random Forest	1	ranger	1
parRF	Parallel Random Forest	1	parRF	1
naive_bayes	Naive Bayes Classifier	1	naive_bayes	1
LogitBoost	Logistic Regression	1	LogitBoost	1

(C) SW aligned GLAM2 motifs

Validation Data			Test Data	
Model Name	Model Type	AUC-ROC	Model	AUC-ROC
svmRadialCost	SVM with Radial Basis Function	1	svmRadialCost	1
svmRadial	SVM with Radial Basis Function	1	svmRadial	1
ada	Regularized Discriminant Analysis	1	ada	1
ranger	Random Forest	1	ranger	1
dwdRadial	Discriminant with Radial Basis Function	1	dwdRadial	1
LogitBoost	Logistic Regression	1	LogitBoost	1

(D)[a] Motifs extracted from SOTA PLMs

Protein Language Model	Best Classifier	AUC-ROC (Test Data)
Unirep	RandomForest	0.80
ESM-1b	RandomForest	0.86
ProtT5	Logistic Regression	0.95

[a] Results reported in [22]. For dBG, AUC-ROC is the average score of 5-fold cross validation.

6. Summary and future work

In this work, we introduced a novel feature engineering pipeline leveraging an extended dBG to process unstructured biological sequential data for TAI models. Our approach dynamically constructs a traditional feature space, enabling the effective use of TAI algorithms across diverse datasets. The experimental results validate that dBG-generated motifs are significant for MD and classification tasks. Furthermore, our dBG-based MD algorithm outperforms conventional methods, including PLMs and GLAM2 motifs, particularly in scenarios with skewed class distributions. The robustness of dBG features across different alignment algorithms (SW and BLAST) and various classification models underscores the versatility and efficacy of our approach.

Additionally, our extension of dBG to identify GAR motifs further validates the broader applicability of this method in feature extraction tasks beyond classification. The findings indicate that dBG, especially when assisted by substitution matrices like BLOSUM and PAM, can effectively detect biologically relevant motifs.

The scalability of our pipeline make it a powerful tool for feature engineering. Future work will focus on integrating domain-specific information to enhance the biological significance of the features and exploring hybrid systems that combine dBG with advanced machine learning techniques like deep learning models. We

Table 2. (Experiment 2) Comparison of best models (trained on dBG motifs) with PLM and GLAM2 motifs.

(A) SW aligned dBG motifs

Validation Data			Test Data	
Model Name	Model Type	AUC-ROC	Model	AUC-ROC
svmRadialCost	SVM with Radial Basis Function	1	svmRadialCost	0.97
svmRadial	SVM with Radial Basis Function	1	svmRadial	0.97
ada	Regularized Discriminant Analysis	1	ada	0.97
ranger	Random Forest	1	ranger	0.97
dwdRadial	Discriminant with Radial Basis Function	1	dwdRadial	0.97
LogitBoost	Logistic Regression	1	LogitBoost	0.97

(B) Blast aligned dBG motifs

Validation Data			Test Data	
Model Name	Model Type	AUC-ROC	Model	AUC-ROC
wsrf	Weighted Subspace Random Forest	1	wsrf	0.97
rf	Random Forest	1	rf	0.97
ranger	Random Forest	1	ranger	0.96
parRF	Parallel Random Forest	1	parRF	0.96
naive_bayes	Naive Bayes Classifier	1	naive_bayes	0.97
LogitBoost	Logistic Regression	1	LogitBoost	0.96

(C) SW aligned GLAM2 motifs

Validation Data			Test Data	
Model Name	Model Type	AUC-ROC	Model	AUC-ROC
svmRadialCost	SVM with Radial Basis Function	1	svmRadialCost	0.91
svmRadial	SVM with Radial Basis Function	1	svmRadial	0.89
ada	Regularized Discriminant Analysis	1	ada	0.92
ranger	Random Forest	1	ranger	0.92
dwdRadial	Discriminant with Radial Basis Function	1	dwdRadial	0.91
LogitBoost	Logistic Regression	1	LogitBoost	0.94

(D) [a]Motifs extracted from SOTA PLMs

Protein Language Model	Best Classifier	AUC-ROC (Test Data)
Unirep	RandomForest	0.80
ESM-1b	RandomForest	0.86
ProtT5	Logistic Regression	0.95

[a] Results reported in [22]. For dBG, AUC-ROC is the average score of 5-fold cross validation.

Table 3. Motifs in Approximate de Bruijn Graph by Substitution Matrices.

Substitution Matrix	# GMF Motifs	# Total Motifs
BLOSUM80	11	20
BLOSUM90	13	17
PAM30	11	11
No Approx. ($\kappa = 0$)	8	8

Note: GMF refers to Graph Motif Finding. This table compares the effectiveness of different matrices in motif identification, with κ indicating the approximation parameter.

also aim to expand the application of dBG to other unstructured data domains, providing a more general-purpose framework for feature extraction and data analysis.

Data availability statement

The data that support the findings of this study are openly available at the following URL/DOI: https://github.com/parichit/Debruijn_Graph_Classification.

ORCID iDs

Mert Onur Cakiroglu https://orcid.org/0009-0001-0798-1361
Parichit Sharma https://orcid.org/0000-0003-0822-1089

References

[1] Locey K J and Lennon J T 2016 Scaling laws predict global microbial diversity *Proc. Natl Acad. Sci. USA* **113** 5970–5
[2] Altschul S F and Pop M 2017 *Handbook of Discrete and Combinatorial Mathematics* 2nd edn (CRC Press/Taylor & Francis)
[3] Lewin B 2006 *Essential Genes* (Pearson Prentice Hall)
[4] Costello J C, Dalkilic M M, Beason S M, Patwardhan R, Middha S, Eads B D and Andrews J R 2009 Gene networks in drosophila melanogaster: integrating experimental data to predict gene function *Genome Biol.* **10** R97
[5] Hashim F A, Mabrouk M S and Al-Atabany W 2019 Review of different sequence motif finding algorithms *Avicenna J. Med. Biotechnol.* **11** 130–48
[6] Hon L S and Jain A N 2006 A deterministic motif finding algorithm with application to the human genome *Bioinformatics* **22** 1047–54
[7] Lawrence C E and Reilly A A 1990 An expectation maximization (EM) algorithm for the identification and characterization of common sites in unaligned biopolymer sequences *Proteins Struct. Funct. Bioinform.* **7** 41–51
[8] Lawrence C E, Altschul S F, Boguski M S, Liu J S, Neuwald A F and Wootton J C 1993 Detecting subtle sequence signals: a gibbs sampling strategy for multiple alignment *Science* **262** 208–14
[9] Pevzner P A and Sze S H 2000 Combinatorial approaches to finding subtle signals in DNA sequences *Proc. Int. Conf. Intelligent Systems For Molecular Biology* **vol 8** 269–78
[10] Sinha S and Tompa M 2003 YMF: a program for discovery of novel transcription factor binding sites by statistical overrepresentation *Nucleic Acids Res.* **31** 3586–8
[11] Buhler J and Tompa M 2002 Finding motifs using random projections *J. Comput. Biol.* **9** 225–42
[12] Zaslavsky E and Singh M 2006 A combinatorial optimization approach for diverse motif finding applications *Algorithms Mol. Biol.* **1** 13
[13] Bailey T L and Elkan C 1994 Fitting a mixture model by expectation maximization to discover motifs in bipolymers *Int. Conf. on Intelligent Systems for Molecular Biology* (available at: https://www.cs.toronto.edu/~brudno/csc2417_15/10.1.1.121.7056.pdf)
[14] Aitkin M and Rubin D B 1985 Estimation and hypothesis testing in finite mixture models *J. R. Stat. Soc.* B **47** 67–75
[15] Bailey T L, Johnson J, Grant C E and Noble W S 2015 The meme suite *Nucleic Acids Res.* **43** W39–W49
[16] Frith M C, Saunders N F W, Kobe B and Bailey T L 2008 Discovering sequence motifs with arbitrary insertions and deletions *PLoS Comput. Biol.* **4** e1000071
[17] Bailey T L 2011 DREME: motif discovery in transcription factor ChIP-seq data *Bioinformatics* **27** 1653–9
[18] Machanick P and Bailey T L 2011 MEME-ChiP: motif analysis of large DNA datasets *Bioinformatics* **27** 1696–7
[19] Geman S and Geman D 1984 Stochastic relaxation, gibbs distributions and the bayesian restoration of images *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-6** 721–41
[20] Jonassen I 1997 Efficient discovery of conserved patterns using a pattern graph *Bioinformatics* **13** 509–22
[21] Burdukiewicz M, Sobczyk P, Rödiger S, Duda-Madej A, Mackiewicz P and Kotulska M 2017 Amyloidogenic motifs revealed by n-gram analysis *Sci. Rep.* **7** 12961
[22] Ashiqul Islam S M, Heil B J, Kearney C M and Baker E J 2017 Protein classification using modified n-grams and skip-grams *Bioinformatics* **34** 1481–7
[23] Al-Ajlan A and El Allali A 2019 CNN-MGP: Convolutional neural networks for metagenomics gene prediction *Interdiscip. Sci. Comput. Life Sci.* **11** 628–35
[24] Yu C, He R L and Yau S S-T 2014 Viral genome phylogeny based on Lempel-Ziv complexity and hausdorff distance *J. Theor. Biol.* **348** 12–20
[25] Pickett B D, Miller J B and Ridge P G 2017 Kmer-SSR: a fast and exhaustive SSR search algorithm *Bioinformatics* **33** 3922–8
[26] Gligorijević V *et al* 2021 Structure-based protein function prediction using graph convolutional networks *Nat. Commun.* **12** 3168
[27] Shi J-Y and Zhang Y-N 2009 Using texture descriptor and radon transform to characterize protein structure and build fast fold recognition *Int. Association of Computer Science and Information Technology - Spring Conf.* pp 466–70
[28] Brahnam S, Nanni L, Shi J-Y and Lumini A 2010 Local phase quantization texture descriptor for protein classification *Biocomp* pp 159–65
[29] Nanni L, Brahnam S and Lumini A 2012 Wavelet images and chou's pseudo amino acid composition for protein classification *Amino Acids* **43** 657–65
[30] Akbar S, Khan S, Ali F, Hayat M, Qasim M and Gul S 2020 iHBp-deePPSSM: identifying hormone binding proteins using psepssm based evolutionary features and deep learning approach *Chemometr. Intell. Lab. Syst.* **204** 104103
[31] Duarte J M, Srebniak A, Schärer M A and Capitani G 2012 Protein interface classification by evolutionary analysis *BMC Bioinform.* **13** 334
[32] Gribskov M, McLachlan A D and Eisenberg D 1987 Profile analysis: detection of distantly related proteins *Proc. Natl Acad. Sci.* **84** 4355–8
[33] Mazandu G K, Chimusa E R and Mulder N J 2016 Gene Ontology semantic similarity tools: survey on features and challenges for biological knowledge discovery *Briefings Bioinform.* **18** 886–901
[34] Yang Q, Jia C and Li T 2019 Prediction of aptamer-protein interacting pairs based on sparse autoencoder feature extraction and an ensemble classifier *Math. Biosci.* **311** 103–8
[35] Cui Y, Dong Q, Hong D and Wang X 2019 Predicting protein-ligand binding residues with deep convolutional neural networks *BMC Bioinform.* **20** 1–12
[36] Zhao L, Wang J, Hu Y and Cheng L 2020 Conjoint feature representation of GO and protein sequence for PPI prediction based on an inception RNN attention network *Mol. Ther. Nucleic Acids* **22** 198–208
[37] Patwardhan R, Tang H, Sun K and Dalkilic M 2006 An approximate de Bruijn graph approach to multiple local alignment and Motif discovery in protein sequences *VLDB Workshop on Data Mining and Bioinformatics* (*Seoul, Korea, 11 September*) 158–69
[38] Henikoff S and Henikoff J G 1992 Amino acid substitution matrices from protein blocks *Proc. Natl Acad. Sci.* **89** 10915–9
[39] Smith T F and Waterman M S 1981 Identification of common molecular subsequences *J. Mol. Biol.* **147** 195–7

[40] Altschul S F, Gish W, Miller W, Myers E W and Lipman D J 1990 Basic local alignment search tool *J. Mol. Biol.* **215** 403–10
[41] Camacho C, Coulouris G, Avagyan V, Ma N, Papadopoulos J, Bealer K and Madden T L 2009 Blast+: architecture and applications *BMC Bioinform.* **10** 1–9
[42] Raeeszadeh-Sarmazdeh M *et al* 2019 Directed evolution of the metalloproteinase inhibitor TIMP-1 reveals that its N-and C-terminal domains cooperate in matrix metalloproteinase recognition *J. Biol. Chem.* **294** 9476–88
[43] Raeeszadeh-Sarmazdeh M, Toumaian M, Do L and Khorasani-Buxton E 2021 Machine-learning guided directed evolution of metalloproteinase inhibitors *FASEB J.* **35**
[44] Khorasani Buxton E, Patel R, Toumaian M R and Raeeszadeh-Sarmazdeh M 2021 Application of protein language models to low-N engineering of metalloproteinase inhibitors *Int. Conf. on Computational Science and Computational Intelligence (CSCI)* pp 361–5
[45] Kuhn M 2008 Building predictive models in R using the caret package *J. Stat. Softw.* **28** 1–26
[46] Villegas-Morcillo A, Makrodimitris S, van Ham R C H J, Gomez A M, Sanchez V and Reinders M J T 2021 Unsupervised protein embeddings outperform hand-crafted sequence and structure features at predicting molecular function *Bioinformatics* **37** 162–70
[47] Wang Y-C, Huang S-H, Chang C-P and Li C 2023 Identification and characterization of glycine- and arginine-rich motifs in proteins by a novel gar motif finder program *Genes* **14** 330
[48] Menardi G and Torelli N 2014 Training and assessing classification rules with imbalanced data *Data Min. Knowl. Discovery* **28** 92–122
[49] Shubina M Y *et al* 2020 The gar domain integrates functions that are necessary for the proper localization of fibrillarin (FBL) inside eukaryotic cells *PeerJ* **8** e9029