OXFORD

Briefings in Bioinformatics, 2025, **26(1)**, bbae634

https://doi.org/10.1093/bib/bbae634

Review

Benchmarking recent computational tools for DNA-binding protein identification

Xizi Luo, Amadeus Song Yi Chi, Andre Huikai Lin, Tze Jet Ong, Limsoon Wong, Chowdhury Rafeed Rahman*

School of Computing, National University of Singapore, Singapore 119077, Singapore

*Corresponding author. School of Computing, National University of Singapore, Singapore 119077, Singapore. E-mail: e0823054@u.nus.edu

Abstract

Identification of DNA-binding proteins (DBPs) is a crucial task in genome annotation, as it aids in understanding gene regulation, DNA replication, transcriptional control, and various cellular processes. In this paper, we conduct an unbiased benchmarking of 11 state-of-the-art computational tools as well as traditional tools such as ScanProsite, BLAST, and HMMER for identifying DBPs. We highlight the data leakage issue in conventional datasets leading to inflated performance. We introduce new evaluation datasets to support further development. Through a comprehensive evaluation pipeline, we identify potential limitations in models, feature extraction techniques, and training methods, and recommend solutions regarding these issues. We show that combining the predictions of the two best computational tools with BLAST-based prediction significantly enhances DBP identification capability. We provide this consensus method as user-friendly software. The datasets and software are available at https://github.com/Rafeed-bot/DNA_BP_Benchmarking.

Keywords: DNA-binding protein; machine learning; BLAST; CD-HIT; deep learning; motif

Introduction

Identification of DNA-binding proteins (DBP) is a fundamental task in molecular biology. DBPs have a wide range of significant applications including the development of drugs, antibiotics, and steroids for various biological effects. They are also extensively used in biophysical, biochemical and biological studies of DNA [1]. DBPs bind to DNA through a variety of mechanisms, often mediated by specific structural motifs that recognize and interact with particular DNA sequences or structures. These interactions are typically governed by a combination of hydrogen bonds, van der Waals forces and ionic interactions between the protein's amino acid residues and the DNA's nucleotide bases or phosphate backbone. One of the most well-known motifs is the helix-turn-helix (HTH) motif commonly found in transcription factors. The HTH motif consists of two α-helices separated by a short turn. One helix fits into the major groove of the DNA allowing specific base pair recognition while the other stabilizes the interaction [2, 3]. Another common motif is the zinc finger where a zinc ion stabilizes a loop of amino acids that interacts with the DNA [4]. Additionally, proteins can bind to DNA in a sequence-specific manner or in a sequence-independent manner. Sequence-specific interactions involve proteins recognizing particular nucleotide sequences such as the binding of transcription factors to promoter regions to regulate gene expression [5]. Conversely, sequence-independent interactions involve proteins binding to DNA based on its structure or overall shape such as the binding of histones to form nucleosomes [6].

Given the importance of DBPs, various experimental methods have been employed to identify them such as filter binding assays [7], genetic analysis [8], and chromatin immunoprecipitation on microarrays [9]. However, these experimental methods are time-consuming and expensive [10]. Computational tools for DBP identification are essential due to the vast amount of genomic data generated by high-throughput sequencing technologies [11]. Our motivation for this benchmarking study stems from the need to systematically evaluate and compare the performance of existing computational tools for DBP identification. With many tools available, each employing different models, datasets, and features, there is a lack of systematic study regarding their effectiveness. Another goal of this study is to evaluate the performance gain of these tools over simple BLAST search to understand the necessity of these tools for DBP identification.

Machine learning (ML)-based computational methods have gained prominence in recent years for DBP identification leveraging advances in statistical techniques and feature engineering. These methods can be broadly categorized based on the models they employ and the features they utilize for prediction. Traditional ML models such as Support Vector Machines (SVM) [12–16], Random Forests (RF) [17–19], and nearest neighbors algorithm [20] have been used for DBP prediction. These models typically rely on manually curated features extracted from protein sequences and structures. More recent approaches employ deep learning architectures such as Convolutional Neural Networks (CNN) [21, 22], Recurrent Neural Networks (RNN) [22–24], and pretrained protein large language models (as feature extractors) [25, 26]. CNNs are adept at capturing spatial hierarchies in protein sequences, while RNNs effectively model sequential dependencies. PLMs on the other hand relies on self attention mechanism [27] (similar to Transformer architecture)

Received: September 5, 2024. **Revised:** October 29, 2024. **Accepted:** November 20, 2024

in order to have a deeper understanding on relative importance among amino acids of a protein sequence. Although these models often require large datasets for training and fine-tuning, they have reported superior performance in capturing the intricate characteristics of DNA-binding sites.

Features used for prediction can be derived from various sources. Sequence-based features are obtained directly from the amino acid sequences of proteins. Common sequence-based features include amino acid composition [28, 29], dipeptide composition [30, 31], physicochemical properties such as hydrophobicity and charge [12, 32, 33], and PLM embedding [34, 35]. Given the importance of the three-dimensional conformation of proteins in DNA binding, structural features provide valuable insights. These include secondary structure elements (e.g. alpha helices, beta sheets), solvent accessibility [36], and binding site geometries. Position-specific scoring matrices (PSSMs) and profiles derived from multiple sequence alignments capture evolutionary conservation signifying functional importance which can aid in identifying potential DNA-binding regions [14, 34, 37, 38].

One major issue with these methods is the quality and representativeness of the datasets used for training and evaluation. Existing datasets suffer from bias and data leakage leading to inflated performance. Another issue is that some tools primarily focus on various feature extraction techniques from the protein sequences which may include irrelevant or inappropriate feature sets not targeted towards DBP prediction resulting in overfitting. Additionally, the models employed have bias towards specific characteristics of the protein sequences which causes mispredictions.

In this paper, we introduce novel benchmarking datasets which are designed to provide a more realistic and unbiased evaluation of DBP prediction tools. We benchmark 11 state-of-the-art tools analyzing their models, datasets, and feature types and identify the top-performing tools through rigorous evaluation. Furthermore, we explore the potential of traditional tools like BLAST [39], ScanProsite [40], and HMMER [41] for DBP prediction providing insights into their capabilities and limitations. We show that the two best computational tools (according to our evaluation) ensembled with BLAST significantly improves performance. Finally, we analyze the common mistakes made by the top-performing tools to summarize potential causes and possible solutions. We provide our curated train-test split and the ensemble method as user-friendly software via publicly accessible GitHub repository.

Benchmarked tools

The benchmarked tools can be categorized based on the models they proposed, the feature representation methods they employed and the datasets they utilized (Table 1 and Supplementary Data 8). We discuss each categorization in this section. We conclude this section by discussing the issues we faced in running these 11 recent computational tools.

Feature representation-based categorization

The feature representation methods can be categorized into three types: sequence, evolutionary, and structure-based feature representation (summarized in Table 1). Sequence-based features are derived from the original amino acid sequences. Common techniques include one-hot encoding, physicochemical properties, amino acid composition and others. Starting with the most basic approach, PDBP-Fusion and LSTM-CNN_Fusion used one-hot encoding on the sequence residues, allowing the classifiers to learn features automatically. More Recent tools such as PB_DBP and PreDBP-PLMs encoded DNA sequences through pretrained protein language models (PLM). In contrast, PseAAC and DeepDBP examined the frequency of single amino acids, dipeptides, and tripeptides, as well as the distribution of single amino acids. They also focused on the frequency of non-consecutive amino acids and their distributions. DNABP and iDNAProt-ES both utilized physicochemical property features as part of their final feature vectors. These features categorize each amino acid residue based on properties such as hydrophobicity, polarity, and polarizability; and then calculate the composition, transition, and distribution of these groupings [37, 43]. Additionally, DNABP included a sequence-based feature generated by DNABR [48], a tool that predicts DNA-binding residues and their corresponding reliability indices given a sequence. Sequence-derived features have the advantage of being straightforward to compute and do not require evolutionary information, making them faster and more suitable for large-scale analyses. However, since protein sequence is often not a good representative of its 3D secondary structure [49, 50], relying solely on sequence-based features may not capture the complex patterns associated with DBPs.

Evolutionary features are derived from PSSMs generated by PSI-BLAST [51]. For instance, Local-DPP and iDNAProt-ES utilized the average probability of each residue position mutating to one of the 20 residue types during the evolutionary process. Unique to Local-DPP is its consideration of local features by segmenting the PSSM into several sub-PSSMs of the same size before applying any feature extraction technique. Another commonly used feature set involved selecting rows in the PSSM corresponding to the same amino acid type and summing the values in each column (used by DNABP, StackDPPred, KK-DBP, and PreDBP-PLMs). In DNABP, these 20 values were combined with physicochemical property values through some arithmetic operations. Additionally, one feature set involved the occurrence probabilities for pairs of the same amino acids separated by a certain distance along the sequence, while another feature set involved pairs of different amino acids; both calculated from PSSM matrix [34, 37, 44, 47]. It is noteworthy that KK-DBP not only utilized conventional PSSMs with 20 columns but also included features generated by reduced PSSMs (RPSSMs) calculated as the average square of the sum of PSSM values from different columns grouped at specific distances. KK-DBP and PreDBP-PLMs also proposed column-wise variances derived from the RPSSM as one of its final feature sets. Unique to iDNAProt-ES, this tool proposed a feature set that considered the column-wise distribution of values in the PSSM retrieved by calculating partial sums column-wise. Finally, as a deep learning-based tool, LSTM-CNN_Fusion used PSSMs as direct input to the CNN model for automatic feature extraction without employing any PSSM transformation. Evolutionary features derived from PSSMs have proven to be highly informative for DBP prediction, as demonstrated by the superior performance of tools like Local-DPP and LSTM-CNN_Fusion (discussed later). These features capture evolutionary conservation patterns that are often associated with functional regions in proteins. However, processing PSSMs can be challenging due to their variable length (corresponding to protein sequence length). Methods like truncating the PSSM matrix to a fixed size or deriving fixed size summary statistics feature vector may lead to significant loss of information, especially for longer sequences. Additionally, the reliance on PSI-BLAST for PSSM generation can be computationally intensive.

Table 1. Benchmarked tool summary

Tool (publication date)	Feature type	Computation model	Dataset (Train + Test)	Web server (YES / NO)	Code availability
Local-DPP [42] (23/06/2016)	Evolutionary	Classic	PDB1075 + PDB186	NO	Not available
DNABP [43] (01/12/2016)	Sequence + Evolutionary	Classic	PDB14K (97%) + PDB14K (3%)	NO	Not available
iDNAProt-ES [37] (02/11/2017)	Sequence + Evolutionary + Structure	Classic	PDB1075 + PDB186	YES	Not available
StackDPPred [44] (19/07/2018)	Evolutionary + Structure	Classic (stage)	PDB1075 + PDB186	YES	Available but not runnable
PseAAC [45] (11/10/2018)	Sequence	Classic	PDB1075 + PDB186	NO	Available
DeepDBP [46] (19/03/2020)	Sequence	Deep Learning	PDB1075 + PDB186	NO	Partially available (no feature extraction)
PDBP-Fusion [24] (03/05/2021)	Sequence	Deep Learning	PDB14K + PDB2272	NO	Not available
KK-DBP [47] (29/11/2021)	Evolutionary	Classic	PDB1075 + PDB186	NO	Not available
LSTM-CNN_Fusion [23] (02/06/2022)	Sequence + Evolutionary	Deep Learning	PDB14K + PDB2272	NO	Available
PB_DBP [35] (04/12/2023)	Sequence	Deep Learning	A custom dataset (80% + 20%)	NO	Not available
PreDBP-PLMs [34] (08/07/2024)	Sequence + Evolutionary	Deep Learning	PDB1075 + PDB186 PDB14K + PDB2272	NO	Available

For structure-based feature representation, the three-dimensional geometry of proteins is leveraged to identify potential DBPs. iDNAProt-ES proposed a set of structural features extracted from information provided by SPIDER2 [52] as an SPD file. These features included the occurrence and composition of secondary structures, accessible surface area, torsional angles, and structural probabilities. Secondary structure occurrence calculated the frequency of three types of structural motifs in proteins: α-helix (H), β-sheet (E), and random coil (C). Structural probabilities represented the likelihood of each amino acid being in H, E, or C conformation. Beyond these basic features, the matrix in the SPD file was processed similarly to PSSM-related features. For instance, the average products of values from both the same and different columns grouped at specific distances were calculated for torsional angles and structural probabilities. StackDPPred proposed integrating the Residue-wise Contact Energy Matrix (RCEM) [53] into its feature set. The RCEM feature (20 × 20 matrix) approximates the structural stability of proteins by using predicted residue contact energies derived from known 3D structures in order to account for amino acid interactions and intrinsically disordered regions in proteins. Structure-based features can provide insights into the spatial arrangement and folding patterns of proteins, which are crucial for DNA-binding activity. However, obtaining accurate structural information can be resource-intensive and time-consuming. Additionally, the inclusion of numerous structural features derived from predicted models may introduce errors and increase the feature dimensionality significantly, leading to overfitting [54]. Incorporating structural features requires careful consideration to balance the benefit of additional information against the risk of model complexity and data quality.

Computational model-based categorization

We discuss two distinct groups of modeling—one group utilized classic ML models that relied on complex feature extraction techniques, while the other group employed deep learning models capable of automatic feature extraction and end-to-end training. The most commonly used classic ML model was the RF Classifier [55] used by Local-DPP, DNABP, StackDPPred, PseAAC, and KK-DBP. iDNAProt-ES exclusively used SVM [56], while PseAAC used Extra Tree Classifier. StackDPPred proposed a stacking framework which involved two stages of learning. The base classifiers included SVM, Logistic Regression [57], KNN [58], and RF, while the meta-classifier was another SVM with a radial basis function kernel [59]. Classic ML models like Random Forest and SVM offer advantages such as interpretability, faster training, and robustness with small datasets. However, they require fixed-size feature vectors, which can lead to information loss for variable-length sequences. Handling heterogeneous features can also introduce noise if not managed carefully. While simpler than deep learning models, they may struggle with complex sequence patterns and dependencies.

In the deep learning group, a combination of CNN [60] and Long Short-Term Memory networks (LSTM) [61] was utilized by both PDBP-Fusion and LSTM-CNN_Fusion. Unlike PDBP-Fusion, which stacked the LSTM layer on top of the CNN layers, LSTM-CNN_Fusion used these two models in parallel, with CNN extracting features from the PSSMs and LSTM learning information from the original protein sequences. DeepDBP experimented with both CNN and Artificial Neural Network (ANN) model. DeepDBP-ANN utilized sequence composition-based summary statistics while DeepDBP-CNN automatically learned features from the raw sequence with the assistance of a trainable embedding layer and a convolution layer. Regarding the two PLM-related tools, PB_DBP employed a BiLSTM to process the output of a pretrained PLM model, while PreDBP-PLMs utilized a CNN for this purpose. Deep learning models offer several advantages, such as the ability to automatically learn complex patterns from raw data, making them well-suited for tasks involving sequence dependencies and large datasets. Architectures like CNNs excel at capturing local patterns, while LSTMs are effective at modeling sequential dependencies. However, these models come with drawbacks including high computational costs and the need for large datasets to prevent overfitting. Additionally, deep learning models are often less interpretable than traditional ML models, making it challenging to understand the underlying decision-making process. Finally, pretrained PLM embeddings can be biased based on the category of protein sequences they were mostly pretrained on.

Table 2. Commonly used dataset summary

Dataset	Source	+ve & -ve Sequence Number	Max, Min, median sequence length	Data curation process
PDB1075	Liu *et al.* [62]	525 & 550	1323, 50, 189	Remove seqs >=25% similarity and irregular chars
PDB186	Lou *et al.* [36]	93 & 93	1323, 64, 208	Remove seqs >=25% similarity and irregular chars
PDB14K	DNABP [43]	7131 & 7131	4911, 47, 327	Remove seqs >= 40% similarity and irregular chars
PDB2272	PDBP-Fusion [24]	1153 & 1119	5184, 51, 325	Remove seqs >= 25% similarity and irregular chars

Dataset-based categorization

The tools can be divided into two groups based on the training dataset they used (column 4 of Table 1). DNABP, PDBP-Fusion, LSTM-CNN_Fusion, and PreDBP-PLMs all used PDB14K [43] as their training dataset. Among these tools, PDBP-Fusion, LSTM-CNN_Fusion and PreDBP-PLMs utilized PDB2272 [24] as their test dataset. DNABP took out 203 positive and 203 negative sequences from PDB14K and used them for independent testing. The remaining seven tools except PB_DBP used PDB1075 as the training dataset and PDB186 [36] as the test dataset. PB_DBP created a large dataset consisting of around 42K DBPs and non-DBPs, and split the dataset into 80% train and 20% test set. But this dataset has not been made publicly available. Details of the datasets used will be discussed in Section 3.

Tool benchmarking issues

None of the computational tools that we benchmarked had any usable software or web server available except for iDNAProt-ES and StackDPPred. Although iDNAProt-ES has its own web server, it is not effective as it requires PSSM evolutionary feature and SPD structural feature files and cannot work with raw sequences as input. The web server provided by StackDPPred is not accessible from Singapore. On top of that, none of the tools provided the trained models that could directly be used for prediction. As a result, we looked into their provided codebase to train and test these tools ourselves. Six out of the eleven tools did not have any code available and we could not retrieve the source codes even after corresponding with their authors (see the last column of Table 1). StackDPPred and DeepDBP had end-to-end running issues which required our attention. The codes provided by PseAAC, LSTM-CNN_Fusion, and PreDBP-PLMs were runnable. As mentioned in Supplementary Data 1, some of the tools had some inconsistencies between the provided code and the paper. In such cases, we simply followed the code. In some other cases, the selected features were not clearly indicated and as a result, we had to rerun the feature selection process. Furthermore, two feature values of DNABP required running of the DNABR classifier (predicts DNA-binding residue confidence scores) on the protein sequence of interest. Since the server of this classifier is currently not available, we had to run DNABP without using these two feature values. Finally, we attempted to achieve similar results as reported in the corresponding tool papers after training on the mentioned training set and testing on the mentioned test set, which would ensure correct replicability of the tools. Unfortunately, our achieved performance was much worse than the reported performance for 2 out of the 11 tools— iDNAProt-ES (around 40% drop in MCC score) and DeepDBP (over 80% drop in MCC score) marked in yellow in Supplementary Data 4. Furthermore, since the training and test datasets of PB_DBP were not available, we could not make sure that it's reported performance was reproducible.

Current dataset details and limitations

Among the 11 tools we benchmarked, 2 combinations of training and test datasets were commonly found—(1) PDB1075 as training & PDB186 as test, (2) PDB14K as training & PDB2272 as test. Table 2 contains the basic information of the four datasets. PDB1075 and PDB186 were originally curated by Liu *et al.* [62] and Lou *et al.* [36]. DBPs were initially acquired from the Protein Data Bank (PDB) [63] by mmCIF keyword of 'DNA binding protein' through the advanced search interface. To mitigate fragmentary sequences, proteins with fewer than 50 and 60 amino acids were excluded from PDB1075 and PDB186, respectively. Additionally, proteins containing the residue 'X' were omitted to avoid unknown residues. Subsequently, PISCES [64] and NCBI's BLASTCLUST [51] were employed to eliminate proteins exhibiting more than 25% identity with any protein within PDB1075 and PDB186, respectively. Similarly, the non-DBPs were randomly selected from other proteins in PDB and filtered using the same criteria.

In case of PDB14K, 'DNA binding' was used as a keyword to search the UniProt database to obtain DBPs. Only sequences with lengths between 50 and 6000 amino acids were retained, and redundant data were removed using a 40% similarity threshold. To obtain the non-DBPs, all proteins from the UniProt database that lacked any implied RNA/DNA-binding functionality were obtained following a procedure proposed by Cai and Lin [12]. PDB2272 was curated in a similar fashion. Random selection was performed on the non-DBPs in order to make all these four datasets have almost equal number of DBPs and non-DBPs.

The primary limitations of these datasets can be categorized into three main points. Firstly, non-DBPs (negative) vastly outnumber DBPs (positive) in the real world. Consequently, datasets with approximately equal number of positive and negative samples do not accurately represent real-life scenario. The random undersampling of non-DBPs during training for class balancing eliminates important data points (these data maybe important for a comprehensive understanding of the full feature space of negative samples) potentially leading to a less robust model. The most alarming aspect is that this random undersampling was also performed in the test sets of these earlier works for class balancing; this would confound the extrapolation to real-life performance [65, 66].

Secondly, data leakage is prevalent between the training and test datasets described above. When CD-HIT [67] is applied at 40% similarity threshold, 85 out of the 93 DBPs in PDB186 test set exhibit >40% similarity to one or more DBPs in the corresponding training dataset PDB1075. Similarly, 1007 out of 1153 DBPs in PDB2272 test set exhibit >40% similarity to one or more DBPs in the corresponding training dataset PDB14K. Wang *et al.* [68] described the occurrence of highly similar data across training and test datasets as 'data doppelgangers'. These high numbers of data doppelgangers result in models being repeatedly tested on data that are highly similar to what they encountered during training, leading to inflated performance. The fact that ML

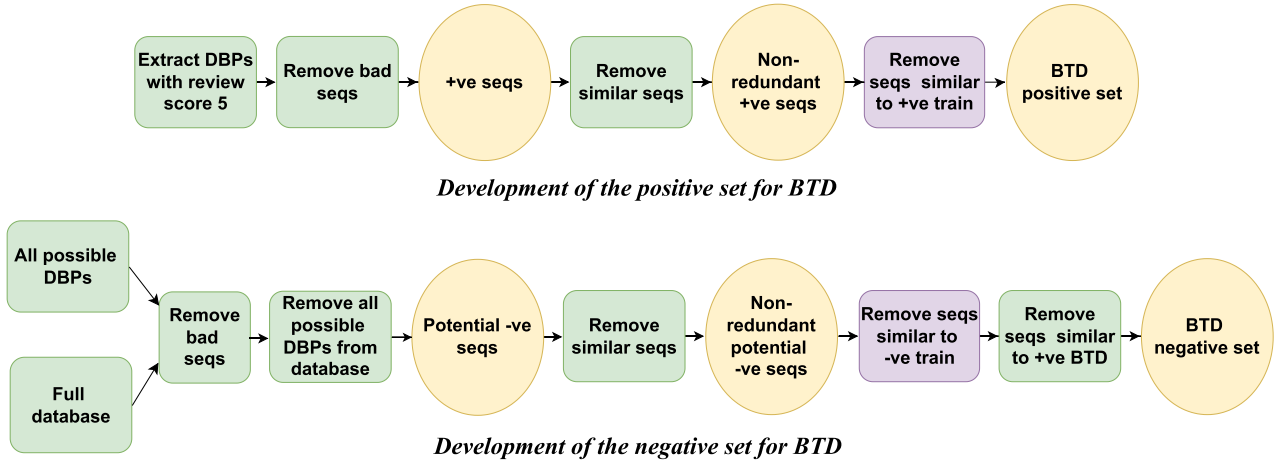

Figure 1. Benchmarking dataset BTD development pipeline.

models have a tendency of overfitting to sequence type training data makes matters even worse [69]. The primary and secondary structure of proteins belonging to the same species can be quite different depending on their biological function [70] further emphasizing the need for non-redundant test data with respect to the training set.

Finally, sequences appearing in both the positive and negative sets cause ambiguity during training. PDB14K contains 74 sequences labeled as both positive and negative, while one sequence labeled as positive in PDB1075 appears as negative in PDB14K.

These limitations emphasizes the necessity for a new benchmarking dataset—one with minimized similarity between training and test datasets to more accurately evaluate the performance of various models under conditions where negative samples outnumber positive ones with no ambiguity.

Benchmarking dataset development

To address the shortcomings of current datasets, we constructed a new dataset named BTD. To construct the positive set, we first extracted DBPs from UniProtKB [71] selecting only those manually annotated with a review score of 5 (top subfigure of Fig. 1). Sequences containing non-amino acid characters including ambiguous ones and those with lengths outside the range of 50–3000 amino acids were excluded. CD-HIT was applied with a similarity threshold of 40% to reduce redundancy within the extracted DBPs. To further prevent redundancy between training and test DBPs, we filtered out sequences similar to those found in commonly used training dataset (PDB14K and PDB1075) DBPs at 40% similarity threshold. This filtering step ensures the elimination of potential data leakage between the training datasets and BTD. Without this precaution, the test data in BTD would maintain internal non-redundancy but might still overlap with the training datasets, compromising the integrity of the evaluation process. The output refined dataset constitutes the positive component of BTD containing 16 384 sequences.

To compile a negative set (bottom subfigure of Fig. 1), we first identified all possible DBPs from UniProtKB by filtering with keywords indicative of RNA/DNA-binding functionality as described in [12]. Concurrently, the full Swiss-Prot database was also downloaded. The set of non-DBPs was obtained by excluding the possible DBPs from the Swiss-Prot database. Note that sequences containing non-amino acid characters and outside the length range of 50–3000 amino acids were not included. CD-HIT was then applied to the potential non-DBPs to remove redundancy with 40% similarity threshold. To avoid data doppelgangers between training and test sets, we further filtered out sequences similar to those in commonly used training dataset non-DBPs at 40% similarity threshold. Finally, non-DBPs with over 40% similarity to at least one DBP of BTD positive set were removed. This step addressed the ambiguity in protein databases, where non-DBPs were not explicitly labeled. By filtering out potential non-DBP sequences with high similarity to known DBPs, we ensure our non-DBP set accurately represents negative examples, minimizing mislabeling. Though it may limit predictions for functionally different yet similar sequences, this trade-off is necessary to ensure reliable benchmarking. This yielded the negative component of BTD comprising 40 734 sequences.

In addition to BTD, we developed two new datasets, EBTD and HBTD to investigate the impact of high and low similarity between training and test datasets on model performance, respectively. While BTD positive and negative sequences have less than 40% similarity compared to the corresponding training sequences, EBTD is the complete opposite in this regard. EBTD creation steps are the same as BTD creation steps except for the boxes in purple as shown in Fig. 1. In case of EBTD, we used CD-HIT to retain only those sequences that have over 40% similarity to at least one sequence of the same class from the training set (we made sure that we do not have the same sequence in training and in EBTD). The resulting EBTD contains 2,305 positive samples and 1963 negative samples. HBTD, on the other hand, is a subset of BTD, with a modification applied during the step marked in purple in Fig. 1. Specifically, CD-HIT was used with a 30% similarity threshold instead of the 40% (used for BTD). This adjustment aims to increase the difficulty of predictions, thereby providing a more rigorous evaluation of model robustness. The HBTD dataset consists of 16,003 positive samples and 38 618 negative samples.

Due to emerging needs in the evaluation process, we expanded BTD into two new datasets: BTD-Combo (used in Subsection 5.3) and a proposed train-test dataset (used in Subsection 5.4). BTD-Combo is designed for five-fold cross-validation and comprises non-redundant PDB1075, PDB14K, and BTD. The non-redundancy of PDB1075 and PDB14K was achieved by merging these two datasets and applying CD-HIT with a 40% similarity threshold separately for positive and negative samples (Supplementary Fig. S1). This approach ensures minimum data leakage during cross validation using BTD-Combo. BTD-Combo contains 22 332 DBPs

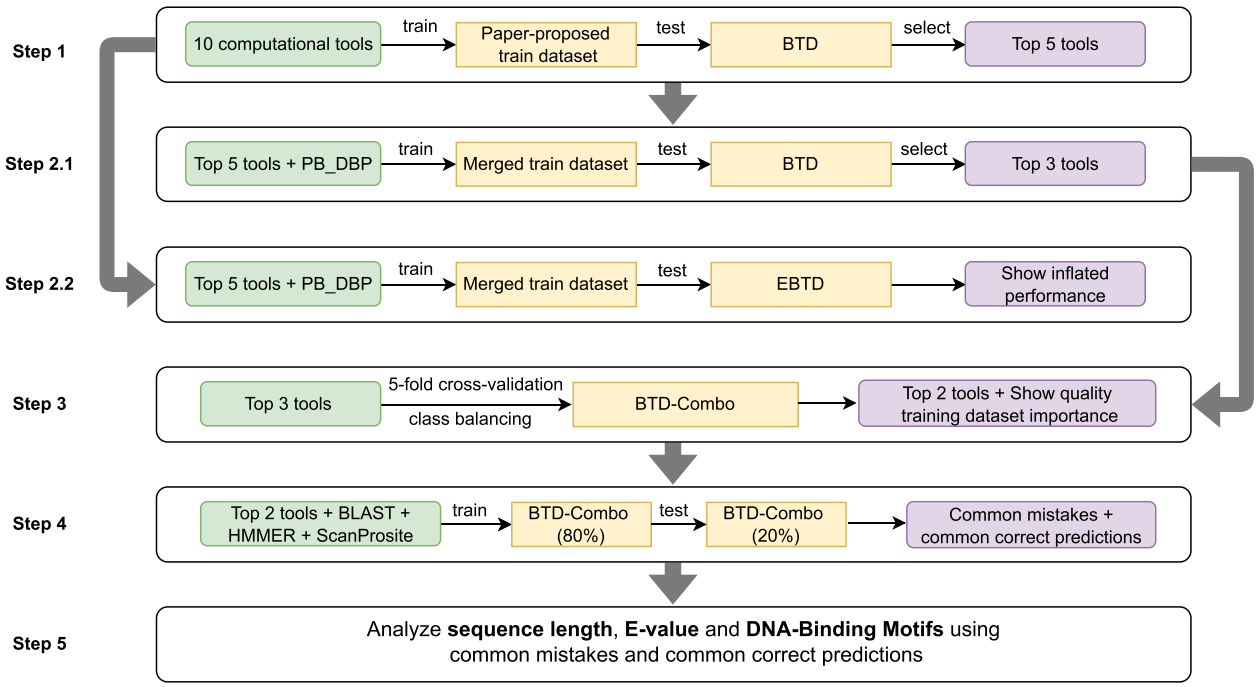

Figure 2. Proposed evaluation pipeline for benchmarking analysis.

and 44 405 non-DBPs. The train-test dataset was constructed from BTD-Combo. We split BTD-Combo into 80% train and 20% test ensuring stratification by class labels and by length percentile groups (Supplementary Fig. S1). The training set has 17 857 positive and 35 428 negative sequences while the test set has 4475 positive and 8977 negative sequences. This train-test dataset ensures that model performance is evaluated more reliably by minimizing variability due to differences in sequence lengths and class distribution.

Tool evaluation

Figure 2 illustrates our tool evaluation pipeline. We start by training the tool models on their respective training datasets from the paper and testing on BTD. The five best tools selected from the this step are then trained on a merged, unbiased training set and tested on BTD, EBTD and HBTD. EBTD testing is performed to show the inflated performance when the training and test datasets are similar, whereas HBTD testing is designed to evaluate the robustness of tools on an even less familiar test dataset. The three best tools selected based on the BTD performance are then five-fold cross-validated on the BTD-Combo dataset (created by merging existing training sets with BTD) using two different class balancing techniques. The goal of this step is to check the influence of training set quality on test performance. The two tools with superior cross-validation performance are then trained and tested based on a length-balanced train-test split created from BTD-Combo. We also benchmark popular bioinformatics tools such as BLAST, ScanProsite, and HMMER on the same train-test split after some simple modifications. Finally, we perform error analysis of the two selected tools based on their test predictions. As evaluation metric, we use sensitivity (true positive rate), specificity (true negative rate), and MCC (Matthews Correlation Coefficient) score (see Supplementary Data 2 for more details of these metrics). We consider DBPs as positive and non-DBPs as negative samples for all cases. The best tools have been chosen based on MCC score in each step of evaluation [72] unless a tool is highly biased towards a particular class (denoted by a large difference between sensitivity and specificity).

Table 3. Performance of ten tools on BTD after training on their respective original training data

Tool	Sensitivity	Specificity	MCC
Local-DPP	**0.4237**	**0.6365**	**0.0561**
DNABP	**0.5454**	**0.6217**	**0.1528**
iDNAProt-ES	0.4043	0.5994	0.0036
StackDPPred	**0.5306**	**0.5286**	**0.058**
PseAAC	0.3333	0.4	−0.2469
DeepDBP	0.5623	0.1654	−0.2869
PDBP-Fusion	**0.4612**	**0.6682**	**0.1213**
KK-DBP	0.4287	0.62	0.0451
LSTM-CNN_Fusion	**0.4841**	**0.6213**	**0.0970**
PreDBP-PLMs	0.1195	0.9593	0.1463

Training on proposed dataset

We first trained each of the ten tools using the dataset specified in its respective paper. PB_DBP was excluded because its training dataset was not available. This approach ensured that we could replicate the original trained models as closely as possible thereby reproducing the capabilities of each tool. We removed sequences labeled as both positive and negative from these datasets to remove ambiguity. DNABP, Local-DPP, StackDPPred, PDBP-Fusion, LSTM-CNN_Fusion, and PreDBP-PLMs performed the best on the benchmark test set BTD based on MCC score (Table 3). However, the predictions of PreDBP-PLMs were highly biased, exhibiting high specificity but low sensitivity. Therefore, we excluded this tool from further experiments. Conversely, iDNAProt-ES, PseAAC, and DeepDBP showed the worst performance. The possible reasons of their performance issue and potential solutions are discussed in detail in Section 7.

Table 4. Performance of six tools trained on the merged unbiased training dataset and tested on BTD, HBTD, and EBTD separately

Tool	Sensitivity			Specificity			MCC		
	BTD	HBTD	EBTD	BTD	HBTD	EBTD	BTD	HBTD	EBTD
Local-DPP	**0.5168**	0.5103	0.9701	**0.6642**	0.6512	0.9277	**0.168**	0.1503	0.9007
DNABP	**0.5422**	0.537	0.9492	**0.6125**	0.6044	0.8492	**0.1412**	0.1297	0.8068
StackDPPred	0.4273	0.4187	0.9653	0.6152	0.5972	0.9358	0.0393	0.0148	0.9029
PDBP-Fusion	0.4989	0.4932	0.9445	0.6409	0.6316	0.8293	0.1291	0.1158	0.7837
LSTM-CNN_Fusion	**0.5023**	0.4962	0.9575	**0.6491**	0.6365	0.893	**0.14**	0.1231	0.8553
PB_DBP	0.4092	0.4016	0.9562	0.7093	0.6931	0.9032	0.1089	0.0914	0.8631

Training on merged unbiased dataset

The PDB14K training dataset contains approximately 14 times more samples than PDB1075, which may influence the performance of the corresponding tools due to differences in data distribution and scale. To address this variation in training conditions, we merged PDB14K & PDB1075 and re-evaluated the performance of the top five tools and PB_DBP after training on this new, unified dataset. This approach minimizes discrepancies arising from differences in the size and composition of the original training sets. As shown in Table 4, the average MCC score of the six tools on EBTD is around 0.85 while the score is only around 0.1 on BTD, demonstrating that high similarity between the training and test sets significantly inflates model performance. Such similarity issue is present in the respective original training and test positive samples (DBPs) described in detail in Section 3. We show comparison of the original paper-reported true positive rate (sensitivity) with BTD and EBTD sensitivity in Supplementary Data 5. BTD sensitivity degrades more than 40% on average compared to reported sensitivity, while EBTD sensitivity is within 13% of the reported sensitivity on average. This indicates that the reported performances (Supplementary Data 4) in the original papers are largely inflated and unreliable. We also evaluated the tools on a more challenging test dataset HBTD featuring even less similarity between train and test sequences. The results show that the performance metrics of all six tools dropped slightly. Based on the models' performance on BTD, we selected the top three tools—Local-DPP, DNABP, and LSTM-CNN_Fusion for the next stage of evaluation.

Evaluation after training dataset improvement

Here we explore whether using a more representative training dataset can improve the performance of the top three tools selected in the previous subsection. We implemented five-fold cross-validation on BTD-Combo dataset (described in Section 4). The results in Table 5 indicate a significant improvement in performance for Local-DPP and LSTM-CNN_Fusion, with the new average MCC score exceeding 0.4 suggesting the importance of a more representative training dataset. We also observe the poor performance of DNABP (negative MCC score). DNABP was originally developed based on PDB14K, and certain model parameters can be highly sensitive. With the drastic change in training data, re-tuning of model parameters is often necessary. Furthermore, the features used during this cross-validation were pre-selected based on PDB14K training data used in the corresponding paper, which may cause the performance degradation.

In order to assess the impact of class balancing on model performance, we designed two different class balancing scenarios. We shall discuss their effect on the performance of Local-DPP and LSTM-CNN_Fusion. When training on the original training splits without any balancing, specificity is much higher than sensitivity as the number of non-DBPs is 2X compared to DBPs biasing the model towards negative class prediction. When the majority negative class is randomly undersampled to balance both classes, the sensitivity for both tools increases significantly while decreasing specificity marking a more balanced performance in terms of the two classes. Since a large majority of the protein sequences in the real world are non-DBP (negative class), high true negative rate (specificity) maybe preferred by the scientific community. When the minority positive class is oversampled by repeating each positive sample more than once for class balancing, sensitivity increases in Local-DPP, while there is a slight decrease in specificity. This effect seems to be more desirable. Such is not the case for LSTM-CNN_Fusion where there is a large increase in sensitivity and a significant decrease in specificity. Repetitive oversampling makes duplicates of the same samples of the minority class, causing over parameterized deep learning tools such as LSTM-CNN_Fusion to overfit on the minority class, hence such bias is seen towards the minority positive class. Note that throughout the analysis above, the over- and under-sampling were performed only on the training set and not on the test set.

Comparing against traditional tools on the proposed train-test dataset

We now compare Local-DPP and LSTM-CNN_Fusion against traditional tools such as BLAST, ScanProsite, and HMMER using our proposed train-test dataset described in Section 4. BLAST and HMMER are both homology detection tools, with BLAST using local alignment-based sequence similarity searching and HMMER employing HMM-based statistical modeling. ScanProsite on the other hand detects homologous regions by finding specific motifs or conserved patterns associated with protein functions. We employed simple repetitive oversampling on the minority positive samples for class balancing during Local-DPP and LSTM-CNN_Fusion model training. Note that we did not perform oversampling on the test set.

To use BLAST for DBP identification, we used the positive subset of the new training dataset as the protein database. Each test sample was queried against this database using BLAST search. If the output list of training DBPs contained a sequence with over 35% identity to the query sequence and an E-value score smaller than a pre-defined threshold (smaller E-value denotes higher similarity) was found for the sequence, the query test sequence was considered positive. Otherwise, it was predicted as negative. Supplementary Data 6 shows BLAST performance in identifying DBPs for different E-value thresholds. We see a decrease in sensitivity and an increase in specificity as we decrease the E-value threshold from 0.01 to 0.000001. We obtained the highest MCC

Table 5. Five-fold cross-validation performance on original and on balanced BTD-Combo dataset

Tool	Imbalanced			Balanced via Majority Class Undersampling			Balanced via Minority Class Oversampling		
	Sensitivity	Specificity	MCC	Sensitivity	Specificity	MCC	Sensitivity	Specificity	MCC
Local-DPP	0.4541	0.9119	0.4234	0.7148	0.7364	0.4319	0.5283	0.8756	0.4345
DNABP	0.1034	0.7637	−0.1591	0.3136	0.5017	−0.1757	0.1716	0.6631	−0.1733
LSTM-CNN_Fusion	0.4809	0.8766	0.3935	0.6626	0.7659	0.4209	0.7409	0.6956	0.415

Table 6. Comparing the best computational tools with traditional tools on the proposed train-test dataset

Tool	Sensitivity	Specificity	MCC
Local-DPP	0.5846	0.8501	0.4493
LSTM-CNN_Fusion	0.6974	0.7456	0.4257
BLAST	0.5824	0.8147	0.4020
HMMER	0.8346	0.6038	0.4146
ScanProsite	0.5806	0.7918	0.3723

score of 0.402 at a 0.01 threshold (although the MCC scores for the different thresholds are similar). Table 6 shows BLAST performance for the E-value threshold of 0.01. Although the sensitivity of Local-DPP is similar to BLAST, its specificity is better than BLAST. The MCC score of BLAST is slightly smaller compared to both Local-DPP and LSTM-CNN_Fusion. In order to further investigate the perfomance of BLAST, we performed five-fold cross validation on BTD-Combo dataset proposed in Subsection 5.3. We saw similar performance for each of the five folds (Supplementary Data 7), which ensures the consistency of BLAST performance.

Our use of HMMER for DBP identification is highly similar to that of BLAST. Specifically, we employed jackhmmer, an iterative search tool that identifies sequence similarities against a protein database. The output of jackhmmer provides hits for each query sequence based on homologous matches in the database. In our approach, we used the positive subset of the train set as the database, and all test sequences were queried against it. If a query sequence had no matches, it was marked as non-DBP. For matched sequences, the lowest E-value of the matching pair was obtained. If the E-value was below the predefined threshold, the sequence was classified as positive (DBP); otherwise, it was considered negative. Supplementary Data 9 shows jackhmmer performance in identifying DBPs for different E-value thresholds. We obtained the highest MCC score of 0.4146 at 0.0001 threshold shown in Table 6. Although HMMER's MCC score is slightly higher than that of BLAST, its specificity is much lower. Since there are far more negative samples compared to positive ones in real world, we prioritize a tool's ability to accurately identify non-DBPs. Therefore, we chose to proceed with BLAST for further experiments.

To use ScanProsite for DBP prediction, we simply provided the test sequences to its web server using default settings. ScanProsite compares the submitted sequences against the PROSITE database which utilizes motifs from protein sequences found in UniProtKB, PDB and some other protein databases. For each test sequence, we obtained the motif significance scores for possible DNA-binding motifs identified by ScanProsite. A higher significance score suggests a greater likelihood that the match represents a DNA-binding motif. If more than one such motif was detected, we would assign the highest motif score to the test sequence. In cases where there was no detectable motifs, we simply assigned 0. This assigned score was used for classification based on a predefined threshold. ScanProsite achieved the highest MCC score of 0.3723 on test data when we classified a protein as DBP whenever we obtained a positive motif score (last row of Table 6), meaning we were using a classification threshold of 0. ScanProsite's motif database includes sequences from the Swiss-Prot database, which is also the source of our test data. Hence, there is obvious data leakage from the training to the test dataset. Nevertheless, the two computational tools outperform ScanProsite in terms of MCC score. As we increase the classification threshold to higher values for ScanProsite, the sensitivity keeps going down, while the specificity keeps going up significantly (see Supplementary Data 3).

Combining BLAST with the best tools

The top three rows of Table 6 show the best methods according to our evaluation. Let us now consider the prediction of these three methods on our proposed test set described in the previous subsection. Although Local-DPP and LSTM-CNN_Fusion have a considerable amount of overlap in their true positive predictions (%), this overlap is small between these two tools and BLAST (see Venn diagram of Fig. 3 and Supplementary Fig. S2). Motivated by this discovery, we combined the predictions made by BLAST, Local-DPP, and LSTM-CNN_Fusion through majority voting, achieving *sensitivity, specificity, and MCC of 0.6545, 0.8479, and 0.5079*, respectively, on the proposed test set. This MCC score is significantly higher compared to the best MCC score of 0.4493 achieved by Local-DPP. In fact, this voting system achieves similar high specificity as Local-DPP while achieving a much higher sensitivity. The specificity did not increase, because the overlap of true negative predictions (%) among these three methods is considerably large (Supplementary Fig. S3).

Error analysis

Having identified the best two tools, we proceed to analyze the common mistakes and common correct predictions made by them on the test set described in Supplementary Fig. S1. Our aim is to uncover general patterns and challenges inherent in DBP prediction. We discuss the relevant analyses in detail in this section.

Sequence length analysis

Figure 4 shows error distributions across different length ranges segmented by every 10^{th} percentile of both positive and negative sequences. In case of positive test sequences, shorter sequences exhibit the highest error rates. This error rate consistently decreased with increasing sequence length. Conversely, for negative test sequences, the error rates increased with sequence length. These findings suggest that the selected tools tend to

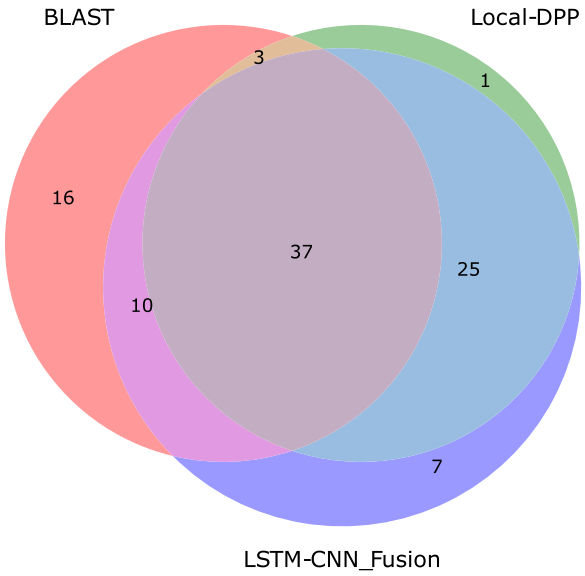

Figure 3. Venn diagram of true positive sequences (%) predicted by BLAST and the two best performing tools.

overfit based on the length of training sequences, leading to biased predictions that may obscure the true biological features relevant for classification. The possible reasons of such bias is discussed in the second paragraph of Section 7.

E-ratio analysis

Before analyzing the E-ratio plots in Fig. 5, let us discuss how the E-ratio was generated. The first step was to create two databases named TrainDB+ and TrainDB- containing positive and negative training sequences, respectively. In order to get the E-ratio of a single positive test sample, we performed BLAST search on TrainDB- using the positive test sample as the query and obtained the mean of the lowest 5% E-values (lower E-value indicates higher alignment) denoted as *E-opposite*. We then performed BLAST search on TrainDB+ using the same positive test sample as query sequence and obtained the mean of the lowest 5% E-values denoted as *E-same*. The *E-ratio* of this positive test sample is the ratio of *E-opposite* and *E-same*. This process was repeated for all common correct and common mistake positive test samples to generate the left sub-figure box plots in Fig. 5. A higher E-ratio indicates that the positive test sample is more similar to the positive training samples compared to the negative training samples. Similarly, for each negative test sample, we calculated the E-ratio by performing BLAST searches on TrainDB+ to obtain *E-opposite* and on TrainDB- to obtain *E-same*. A higher E-ratio indicates that the negative test sample is more similar to the negative training samples compared to the positive training samples. All E-ratios have been log-scaled for better visualization.

We performed a one-sided Wilcoxon rank-sum test [73], revealing that the E-ratios of correctly classified samples were significantly higher than those of misclassified samples for both the positive and negative classes (*P*-value of 1.77×10^{-85} and 0.0005 for the positive and negative classes, respectively). This significance analysis indicates that misclassified samples are relatively more similar to training samples of the opposite class compared to the correctly classified samples. This phenomenon is significantly more pronounced in DBPs (positive) compared to non-DBPs (negative).

Motif score analysis

Ideally, ScanProsite should not provide any DNA-binding motif as output for non-DBPs, while for DBPs, it should provide one or more motifs with high motif significance scores. Our hypothesis was that mistaken non-DBPs would be assigned relatively high motif scores, while mistaken DBPs would be assigned relatively low motif scores by ScanProsite. Fig. 6 shows a summary of the log-scaled maximum motif score for each test sequence (for sequences with no predicted motifs, a log-scaled score of 0 was assigned). Positive (DBP) common correct samples have significantly higher motif scores compared to positive common mistakes (one-sided Wilcoxon rank-sum test *P*-value of 4.01×10^{-140}), while negative common correct samples had much lower motif scores compared to negative common mistakes (one-sided Wilcoxon rank-sum test *P*-value of 1.54×10^{-106}), proving our hypothesis. This means that mistaken non-DBPs have some local patterns which have close resemblance to DNA-binding motifs stored in ScanProsite database, making them harder to predict as non-DBPs.

BLAST-only true positive analysis

We examined why 16% of DBPs (652 sequences) were identified only by BLAST and not by the other two tools, as shown in Fig. 3. First, we explored why Local-DPP and LSTM-CNN_Fusion failed to correctly classify these sequences by conducting the three error analyses mentioned in the previous three subsections. Plots from the top row of Supplementary Fig. S4 indicates that the error pattern of these DBPs for all three analyses are very similar to the commonly mistaken DBPs (positive sequences) by the two tools and hence, they failed to correctly identify these 16% sequences.

Next, to understand why BLAST could identify these DBPs (652 sequences), we investigated how BLAST matches sequences through local alignment. We hypothesized that these sequences were correctly identified due to significant local similarity with known positive sequences from the database. To test this hypothesis, we performed BLAST searches for these 652 sequences (query) against positive training sequences (database), retaining the match with the lowest E-value for each sequence. We then used the Smith-Waterman algorithm [74] to compute the highest local alignment scores for these matches. For comparison, we repeated the same process with two additional sets: the full set of positive sequences correctly and incorrectly predicted by BLAST. Plot from the bottom row of Supplementary Fig. S4 shows that the alignment scores of the 652 sequences closely match those of the correctly predicted positives (high alignemnt score) and are distinct from the misclassified ones (low alignemnt score), indicating that local alignment plays a crucial role in BLAST's predictive success.

Discussion and recommendations

Local-DPP and LSTM-CNN_Fusion are the two best tools according to our benchmarking. Both of these tools used evolutionary features generated from PSSM transformation, highlighting the importance of such features. LSTM-CNN_Fusion truncates the PSSM matrix to a fixed size, while Local-DPP calculates summary statistics for a Random Forest model. Such processing may lead to significant loss of information. Instead, these variable-size feature matrices might be more smartly processed utilizing recent variable-size graph learning algorithms [75, 76]. Additionally, instead of using BLAST for PSSM generation, software such as

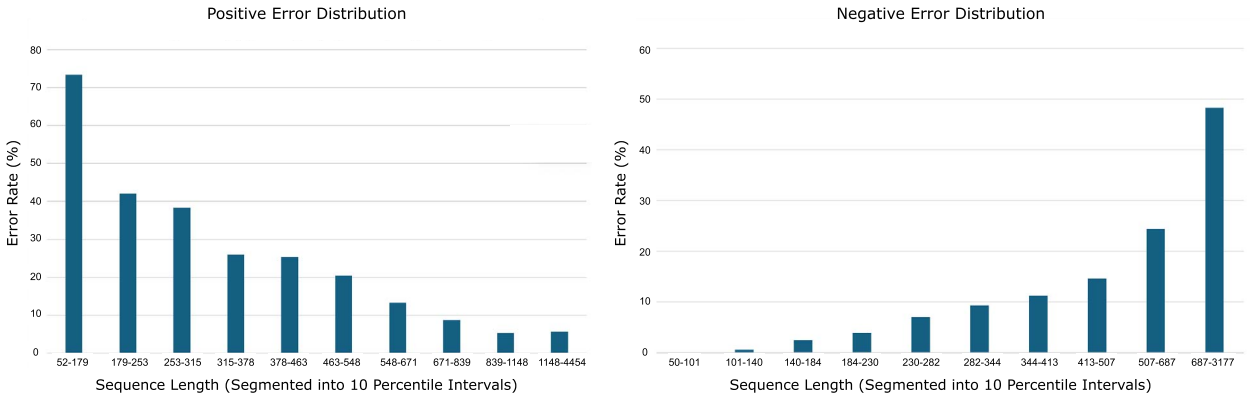

Figure 4. Error rate for positive (left) and negative (right) sequences of different length ranges.

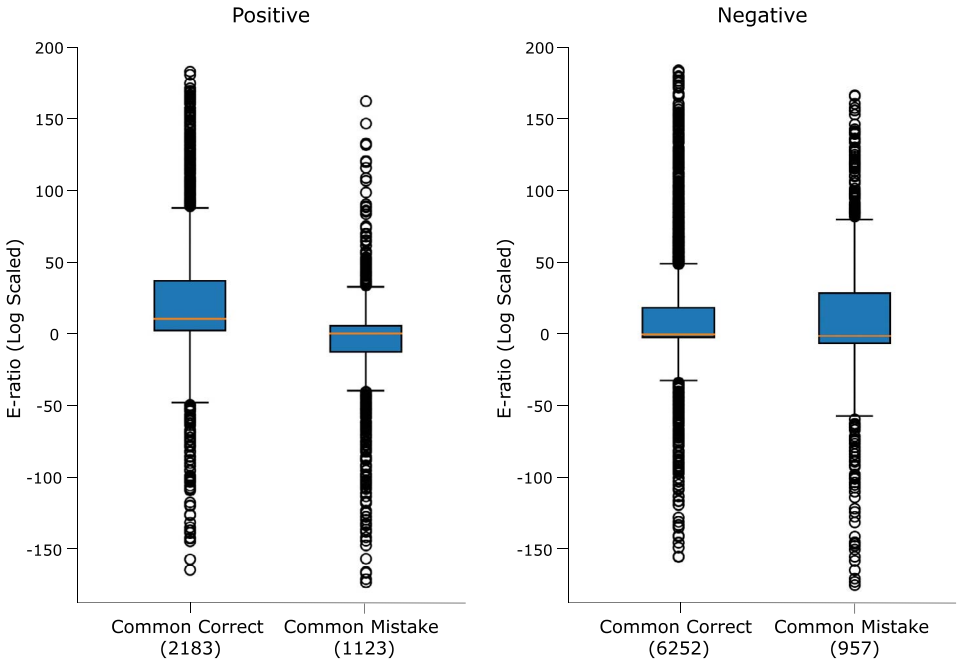

Figure 5. E-ratio plot for positive (left) and negative (right) test samples.

HHblits might be used for obtaining profile HMMs (more generalized form of PSSMs) from sequences [77]. Such measures might help reduce the E-ratio bias described in Subsection 6.2.

For the two recent deep learning-based tools named PDBP-Fusion and LSTM-CNN_Fusion, a notable issue is the need to select a fixed maximum length (700) for input sequences to fit within the neural network's architecture. This constraint might result in the loss of crucial information, as sequences exceeding the predetermined length are truncated. Furthermore, these tools take a residue-based encoding scheme when using the LSTM model, meaning that the LSTM has to go through 700 time steps before making a prediction. The models were originally trained on only around 14K training samples while needing to learn sequential patterns over a path of length 700. Such a model might cause sub-optimal learning and overfitting [78, 79]. In such cases, smart window-based encoding schemes [80] and attention mechanisms [81, 82] might help in reducing the length bias described in Subsection 6.1. This length bias also exists in Random Forest-based Local-DPP, probably because the size of the sub-PSSMs it uses as its features depend on the protein sequence length.

The latest tools, PB_DBP and PreDBP-PLMs, highlight the trend of performing bioinformatics prediction tasks with the help of PLMs [83–86]. While promising, they tend to have high specificity and very low sensitivity, leading to the misclassification of many actual DBPs. Vast majority of the sequences used for PLM pre-training consists of non-DBPs. Since these models are not fine-tuned on DBP identification specific training data in PB_DBP and PreDBP-PLMs, they may generate embeddings that fail to carry DBP-specific features. Future work should focus on refining these pretrained PLMs to reduce biases and improve generalization.

DeepDBP, PseAAC, and iDNAProt-ES are the three tools showing the worst performance during our benchmarking. Both DeepDBP and PseAAC relied solely on sequence-derived summary statistics, which shows the necessity of more informative novel sequence-derived features and perhaps also inclusion of evolutionary and structural features. While exploring ways to incorporate structural insights, we considered using AlphaFold [87], which tackles a key challenge in molecular biology: predicting protein structures from sequence data, a task traditionally reliant on costly and time-intensive methods like X-ray crystallography or NMR spectroscopy [88]. By leveraging ML, AlphaFold offers rapid

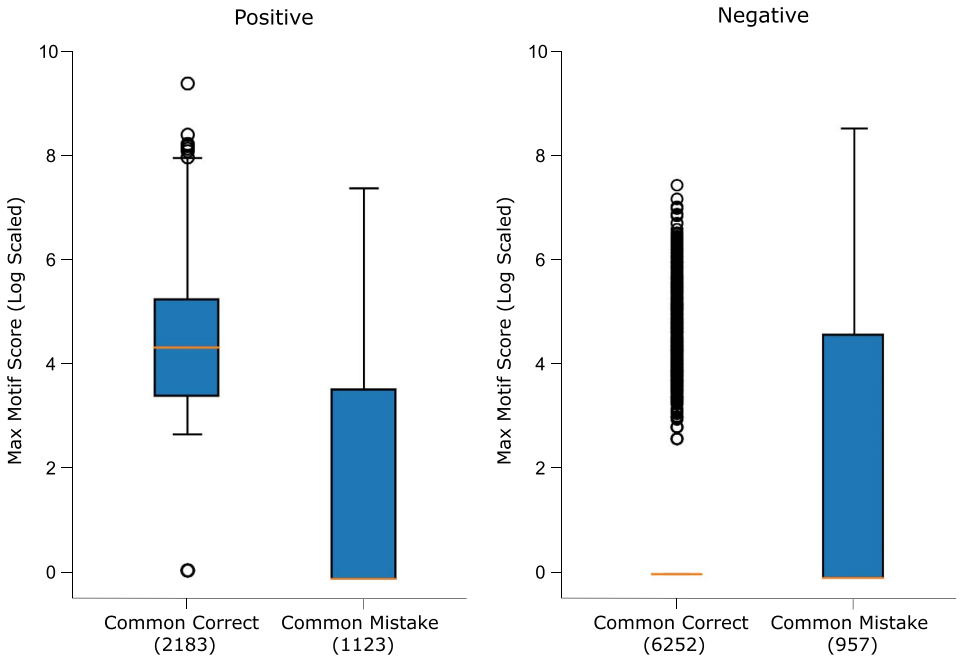

Figure 6. ScanProsite assigned motif score plot for positive (left) and negative (right) test samples.

and accurate predictions, greatly advancing structural biology. Structural insights from AlphaFold can enhance DBP prediction by capturing subtle structural motifs and spatial interactions that are difficult to infer from sequence data alone. However, despite its potential, AlphaFold's large runtime and computational cost (exceeding one hour per sequence on T4 GPU) may make it challenging to use for large-scale studies and to deploy for real world use. On the other hand, iDNAProt-ES used all three types of features (sequence, evolutionary and structural), but still demonstrated poor performance. This particular method derived many different features from the PSSMs and from the torsion angles of the structural features, and might have suffered from errors in the determination (by third-party tools) of these features and from a severely increased curse of dimensionality. Using irrelevant and/or error-containing features might lead to overfitting [54, 89].

All tools utilizing classic ML models require one single feature vector per protein sequence. As a result, heterogeneous features of different scales and different inter-sample differences are concatenated together leading to inaccurate model behavior [90, 91]. Instead, each heterogeneous feature type might be encoded in different branches of a learnable network as suggested in [92]. Tools such as StackDPPred stack various ML models in multiple stages, which may introduce additional complexity without substantial benefit. Proper ablation studies should be conducted before implementing stacking strategies to ensure they are justified [93, 94]. There are tools such as HMMER and ScanProsite that provide potential DNA-binding motif locations along with significance scores in the protein sequences. These locations might be given special importance during DBP prediction. Furthermore, tools like DNABR [48], DRNApred [95], and HybridDBRpred [96] might detect whether a residue of a protein is DNA-binding or not. Such residue-level predictions might assist in modeling the overall behavior of the entire protein sequence, thereby reducing the motif significance bias described in Subsection 6.3.

Our evaluation showed moderate performance gain of the two best computational tools over traditional BLAST in terms of MCC. However, many DBPs predicted by them were not predicted by BLAST. The ability to predict DBPs missed by BLAST adds value to these new tools. Indeed, combining these three tools through simple means such as majority voting achieved significantly better performance (Subsection 5.5). This simple ensemble approach achieved 65% sensitivity and 85% specificity on our proposed test set. From the literature [22], it can be estimated that there are 9X more non-DBPs (negative class) compared to DBPs (positive class) in real-life. Thus, every DBP correctly identified by this ensemble approach would be accompanied by about two false positives. This might be good enough for practical use, though there is scope for further improvement.

Researchers should avoid data leakage between training and test set through proper similarity threshold-based filtering. Test set should be representative of the real world in terms of sample number, heterogeneity and class imbalance. We have provided our developed training and test fasta files via GitHub. Researchers can perform cross-validation on the training set to develop their models, while the test set can be used for final validation. Note that both the training and test data we have provided contain significantly more non-DBPs compared to DBPs. All the recent tools we benchmarked have recommended using random undersampling for class balancing the training set. However, as shown in Table 5, this balancing approach negatively impacts the true negative rate (specificity). Random undersampling of the majority class can potentially introduce gaps in the feature space making the trained model less robust in such areas [97, 98]. Instead, clustering-based undersampling might be performed to ensure that sampling covers the entire majority class feature space [99]. Alternatively, oversampling of the minority class might be preferable.

One prominent issue is the lack of usability of these developed tools. Although some of the tools provide their own web server, these servers are rarely maintained after publication; as they are not commercial software. The best practice would be to provide the codes along with the trained model via free public repositories such as GitHub, GitLab, or Zenodo such that it is

possible to run these tools with minimal effort on raw sequences. We have provided Local-DPP, LSTM-CNN_Fusion, BLAST, and majority voting-based ensembling scheme for DBP identification via GitHub, which can be followed as an example by future researchers.

In our benchmarking approach, we show that many existing works on DBP identification have not been thoughtfully tested. A more thoughtfully designed evaluation is then presented and these existing works are then tested accordingly, revealing that their previously reported performance is exaggerated due to the effects of data doppelgangers in their test sets and using insufficiently representative training and test datasets. By retraining on a more representative training dataset, two of these previously reported methods are 'rescued' in terms of performance, though they did not perform significantly better than BLAST. The true value of the retrained methods is then demonstrated by showing that their predicted sets of DBPs are distinct from those identified by BLAST; and thus, a simple majority vote among these two retrained methods and BLAST yields superior performance. Poor evaluation design can be observed in many other protein class prediction problems, and similarly, many methods proposed for these problems may have reported seemingly exaggerated performance (likely for similar reasons of data doppelgangers and non-representative training/test datasets). The same benchmarking strategy should be readily applicable to studying these problems. Examples of such problems include (but are not restricted to)—(a) protein function prediction [100], (b) subcellular localization prediction [101], (c) protein–protein interaction prediction [102], and (d) post translational modification site prediction [103].

Conclusion

We benchmarked 11 state-of-the-art computational DBP identification tools in this paper. We began by categorizing and analyzing these tools based on their models, datasets and types of features used. By scrutinizing the conventional datasets commonly used by these tools, we identified significant limitations, particularly issues related to data leakage leading to inflated performance. To address these issues, we developed two new benchmarking datasets: BTD and EBTD. We demonstrated the inflated performance using EBTD. BTD, designed to mitigate the adverse effects of data leakage, provides a more realistic and unbiased evaluation of different DBP prediction tools, serving as a valuable reference for future tool development and evaluation. Using BTD, we re-evaluated the 11 tools, selected the best 2 tools, and assessed their effectiveness against traditional methods in predicting DBPs through simple adjustments. We showed the significant performance gain of the two tools combined with traditional BLAST search. Additionally, we provided a high-quality train-test dataset for future development based on BTD and available popular training datasets. This dataset along with the top-performing methods (Local-DPP, LSTM-CNN_Fusion, BLAST) and their ensemble classifier are publicly available at https://github.com/Rafeed-bot/DNA_BP_Benchmarking. These methods are directly applicable on raw protein sequences for DBP identification. Beyond tool evaluation, we analyzed the mistakes made by top-performing tools, providing insights for improvement of DBP prediction tools and explored the reasons why BLAST outperformed these tools on certain positive samples. Finally, we discussed possible limitations of current models, feature extraction methods, and data balancing techniques, and offered potential solutions for future research efforts in this field.

Key Points

- We designed a comprehensive evaluation pipeline that systematically evaluates 11 recent machine learning (ML)-based DBP identification tools.
- We analyzed the test prediction mistakes made by top-performing tools identifying their potential limitations in terms of model architecture, feature extraction, and class balancing.
- We showed that although the best of these tools do not convincingly outperform BLAST, they still provide substantial value when integrated together with BLAST into a simple majority-voting ensemble.
- We provide recommendations on more robust development and evaluation and better usability of future tools.
- We provide the two best-performing ML-based tools, BLAST and the ensemble method as user-friendly software, as well as our proposed datasets, publicly available via GitHub.

Supplementary data

Supplementary data is available at *Briefings in Bioinformatics* online.

References

1. Zimmer C, Wähnert U. Nonintercalating DNA-binding ligands: specificity of the interaction and their use as tools in biophysical, biochemical and biological investigations of the genetic material. *Prog Biophys Mol Biol* 1986;**47**:31–112. https://doi.org/10.1016/0079-6107(86)90005-2.
2. Brennan RG, Matthews BW. The helix-turn-helix DNA binding motif. *J Biol Chem* 1989;**264**:1903–6. https://doi.org/10.1016/S0021-9258(18)94115-3.
3. Moxley RA, Jarrett HW, Mitra S. Methods for transcription factor separation. *J Chromatogr B* 2003 Interactions in Biological Systems; **797**:269–88. https://doi.org/10.1016/S1570-0232(03)00609-3.
4. Klug A. The discovery of zinc fingers and their applications in gene regulation and genome manipulation. *Annu Rev Biochem* 2010;**79**:213–31. https://doi.org/10.1146/annurev-biochem-010909-095056.
5. Latchman DS. Transcription factors: an overview. *Int J Biochem Cell Biol* 1997;**29**:1305–12. https://doi.org/10.1016/S1357-2725(97)00085-X.
6. Luger K, Mäder AW, Richmond RK. *et al*. Crystal structure of the nucleosome core particle at 2.8 å resolution. *Nature* 1997;**389**:251–60. https://doi.org/10.1038/38444.
7. Oehler S, Alex R, Barker A. Is nitrocellulose filter binding really a universal assay for protein–dna interactions? *Anal Biochem* 1999;**268**:330–6. https://doi.org/10.1006/abio.1998.3056.
8. Freeman K, Gwadz M, Shore D. Molecular and genetic analysis of the toxic effect of rap1 overexpression in yeast. *Genetics* 1995;**141**:1253–62. https://doi.org/10.1093/genetics/141.4.1253.
9. Buck MJ, Lieb JD. Chip-chip: considerations for the design, analysis, and application of genome-wide chromatin immunoprecipitation experiments. *Genomics* 2004;**83**:349–60. https://doi.org/10.1016/j.ygeno.2003.11.004.

10. Langlois RE, Hui L. Boosting the prediction and understanding of DNA-binding domains from sequence. *Nucleic Acids Res* 2010;**38**:3149–58. https://doi.org/10.1093/nar/gkq061.
11. Shendure J, Ji H. Next-generation DNA sequencing. *Nat Biotechnol* 2008;**26**:1135–45. https://doi.org/10.1038/nbt1486.
12. Cai Y-D, Lin SL. Support vector machines for predicting rRNA-, RNA-, and DNA-binding proteins from amino acid sequence. *Biochim Biophys Acta (BBA)-Proteins Proteom* 2003;**1648**:127–33. https://doi.org/10.1016/S1570-9639(03)00112-2.
13. Bhardwaj N, Langlois RE, Zhao G. *et al.* Kernel-based machine learning protocol for predicting DNA-binding proteins. *Nucleic Acids Res* 2005;**33**:6486–93. https://doi.org/10.1093/nar/gki949.
14. Raghava GPS, Gromiha MM, Kumar M. Identification of DNA-binding proteins using support vector machines and evolutionary profiles. *BMC Bioinform* 2007;**8**:463. https://doi.org/10.1186/1471-2105-8-463.
15. Huang H-L, I-C, Y-F. *et al.* Predicting and analyzing DNA-binding domains using a systematic approach to identifying a set of informative physicochemical and biochemical properties. *Bmc Bioinformatics* 2011;**12**:1–13. https://doi.org/10.1186/1471-2105-12-1.
16. Zhang Y, Jun X, Zheng W. *et al.* NewDNA-Prot: prediction of DNA-binding proteins by employing support vector machine and a comprehensive sequence representation. *Comput Biol Chem* 2014;**52**:51–9. https://doi.org/10.1016/j.compbiolchem.2014.09.002.
17. Jiansheng W, Liu H, Duan X. *et al.* Prediction of DNA-binding residues in proteins from amino acid sequences using a Random Forest model with a hybrid feature. *Bioinformatics* 2009;**25**:30–5. https://doi.org/10.1093/bioinformatics/btn583.
18. Nimrod G, Schushan M, Szilágyi A. *et al.* iDBPs: a web server for the identification of DNA binding proteins. *Bioinformatics* 2010;**26**:692–3. https://doi.org/10.1093/bioinformatics/btq019.
19. Saifur Rahman M, Shatabda S, Saha S. *et al.* DPP-PseAAC: a dna-binding protein prediction model using Chou's general pseaac. *J Theor Biol* 2018;**452**:22–34. https://doi.org/10.1016/j.jtbi.2018.05.006.
20. Qian Z, Cai Y-D, Li Y. A novel computational method to predict transcription factor DNA binding preference. *Biochem Biophys Res Commun* 2006;**348**:1034–7. https://doi.org/10.1016/j.bbrc.2006.07.149.
21. 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. https://doi.org/10.1038/nbt.3300.
22. Yu-Hui Q, Hua Y, Gong X-J. *et al.* On the prediction of DNA-binding proteins only from primary sequences: a deep learning approach. *PloS One* 2017;**12**:e0188129.
23. Weizhong L, Xiaoyi Chen Y, Zhang HW. *et al.* Research on DNA-binding protein identification method based on LSTM-CNN feature fusion. *Comput Math Methods Med* 2022;**2022**:1–10. https://doi.org/10.1155/2022/9705275.
24. Li G, Xiuquan D, Li X. *et al.* Prediction of DNA binding proteins using local features and long-term dependencies with primary sequences based on deep learning. *PeerJ* 2021;**9**:e11262. https://doi.org/10.7717/peerj.11262.
25. Elnaggar A, Heinzinger M, Dallago C. *et al.* ProtTrans: toward understanding the language of life through self-supervised learning. *IEEE Trans Pattern Anal Mach Intell* 2021;**44**:7112–27.
26. Rao R, Bhattacharya N, Thomas N. *et al.* Evaluating protein transfer learning with tape. *Adv Neural Inf Process Syst* 2019;**32**:9689–701.
27. Vaswani A. Attention is all you need. *Adv Neural Inf Process Syst* 2017;**30**:5998–6008.
28. Szilágyi A, Skolnick J. Efficient prediction of nucleic acid binding function from low-resolution protein structures. *J Mol Biol* 2006;**358**:922–33. https://doi.org/10.1016/j.jmb.2006.02.053.
29. Krishna Kumar K, Pugalenthi G, Suganthan PN. DNA-Prot: identification of DNA binding proteins from protein sequence information using Random Forest. *J Biomol Struct Dyn* 2009;**26**:679–86. https://doi.org/10.1080/07391102.2009.10507281.
30. Fang Y, Guo Y, Feng Y. *et al.* Predicting DNA-binding proteins: approached from Chou's pseudo amino acid composition and other specific sequence features. *Amino Acids* 2008;**34**:103–9. https://doi.org/10.1007/s00726-007-0568-2.
31. Nanni L, Lumini A. Combing ontologies and dipeptide composition for predicting DNA-binding proteins. *Amino Acids* 2008;**34**:635–41. https://doi.org/10.1007/s00726-007-0016-3.
32. Song L, Li D, Zeng X. *et al.* nDNA-Prot: identification of DNA-binding proteins based on unbalanced classification. *BMC bioinformatics* 2014;**15**:1–10.
33. Liu B, Jinghao X, Fan S. *et al.* PseDNA-Pro: DNA-binding protein identification by combining Chou's pseaac and physicochemical distance transformation. *Mol Inform* 2015;**34**:8–17. https://doi.org/10.1002/minf.201400025.
34. Qi D, Song C, Liu T. PreDBP-PLMs: prediction of DNA-binding proteins based on pre-trained protein language models and convolutional neural networks. *Anal Biochem* 2024;**694**:115603. https://doi.org/10.1016/j.ab.2024.115603.
35. Li J, Zhang S, Fang C. PB_DBP: identifying DNA-binding proteins using ProBert_BiLSTM model. In: *Proceedings of the 2023 6th International Conference on Big Data Technologies*, pp. 242–6. New York, NY, USA: Association for Computing Machinery, 2023.
36. Lou W, Wang X, Chen F. *et al.* Sequence based prediction of DNA-binding proteins based on hybrid feature selection using Random Forest and Gaussian naive Bayes. *PloS One* 2014;**9**:e86703. https://doi.org/10.1371/journal.pone.0086703.
37. Chowdhury SY, Shatabda S, Dehzangi A. iDNAProt-ES: identification of DNA-binding proteins using evolutionary and structural features. *Sci Rep* 2017;**7**:14938. https://doi.org/10.1038/s41598-017-14945-1.
38. Ali F, Ahmed S, Swati ZNK. *et al.* DP-BINDER: machine learning model for prediction of DNA-binding proteins by fusing evolutionary and physicochemical information. *J Comput Aided Mol Des* 2019;**33**:645–58. https://doi.org/10.1007/s10822-019-00207-x.
39. Altschul SF, Gish W, Miller W. *et al.* Basic local alignment search tool. *J Mol Biol* 1990;**215**:403–10. https://doi.org/10.1016/S0022-2836(05)80360-2.
40. De Castro E, Sigrist CJA, Gattiker A. *et al.* ScanProsite: detection of prosite signature matches and prorule-associated functional and structural residues in proteins. *Nucleic Acids Res* 2006;**34**:W362–5. https://doi.org/10.1093/nar/gkl124.
41. Eddy SR. Accelerated profile hmm searches. *PLoS Comput Biol* 2011;**7**:e1002195. https://doi.org/10.1371/journal.pcbi.1002195.
42. Wei L, Tang J, Zou Q. Local-DPP: an improved DNA-binding protein prediction method by exploring local evolutionary information. *Inform Sci* 2017;**384**:135–44. https://doi.org/10.1016/j.ins.2016.06.026.
43. Ma X, Guo J, Sun X. DNABP: identification of DNA-binding proteins based on feature selection using a Random Forest and predicting binding residues. *PloS One* 2016;**11**:e0167345. https://doi.org/10.1371/journal.pone.0167345.

44. Mishra A, Pokhrel P, Hoque MT. StackDBPred: a stacking based prediction of DNA-binding protein from sequence. *Bioinformatics* 2019;**35**:433–41. https://doi.org/10.1093/bioinformatics/bty653.
45. Adilina S, Farid DM, Shatabda S. Effective DNA binding protein prediction by using key features via Chou's general pseaac. *J Theor Biol* 2019;**460**:64–78. https://doi.org/10.1016/j.jtbi.2018.10.027.
46. Shadman Shadab M, Khan TA, Neezi NA. *et al*. DeepDBP: deep neural networks for identification of DNA-binding proteins. *Inform Med Unlocked* 2020;**19**:100318. https://doi.org/10.1016/j.imu.2020.100318.
47. Jia Y, Huang S, Zhang T. KK-DBP: a multi-feature fusion method for DNA-binding protein identification based on Random Forest. *Front Genet* 2021;**12**:811158. https://doi.org/10.3389/fgene.2021.811158.
48. Ma X, Guo J, Liu H-D. *et al*. Sequence-based prediction of DNA-binding residues in proteins with conservation and correlation information. *IEEE/ACM Trans Comput Biol Bioinform* 2012;**9**:1766–75. https://doi.org/10.1109/TCBB.2012.106.
49. Anfinsen CB. Principles that govern the folding of protein chains. *Science* 1973;**181**:223–30. https://doi.org/10.1126/science.181.4096.223.
50. Dill KA, MacCallum JL. The protein-folding problem, 50 years on. *Science* 2012;**338**:1042–6. https://doi.org/10.1126/science.1219021.
51. Altschul SF, Madden TL, Schäffer AA. *et al*. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res* 1997;**25**:3389–402. https://doi.org/10.1093/nar/25.17.3389.
52. Yang Y, Heffernan R, Paliwal K. *et al*. SPIDER2: a package to predict secondary structure, accessible surface area, and main-chain torsional angles by deep neural networks. *Prediction of Protein Secondary Structure* 2017;**1484**:55–63. https://doi.org/10.1007/978-1-4939-6406-2_6.
53. Dosztanyi Z, Csizmok V, Tompa P. *et al*. The pairwise energy content estimated from amino acid composition discriminates between folded and intrinsically unstructured proteins. *J Mol Biol* 2005;**347**:827–39. https://doi.org/10.1016/j.jmb.2005.01.071.
54. Guyon I, Elisseeff A. An introduction to variable and feature selection. *J Mach Learn Res* 2003;**3**:1157–82.
55. Breiman L. Random forests. *Mach Learn* 2001;**45**:5–32. https://doi.org/10.1023/A:1010933404324.
56. Cortes C, Vapnik V. Support-vector networks. *Mach Learn* 1995;**20**:273–97. https://doi.org/10.1007/BF00994018.
57. Hastie T, Tibshirani R, Friedman JH. *et al*. *The Elements of Statistical Learning: data Mining, Inference, and Prediction*, Vol. 2. New York, NY: Springer, 2009.
58. Altman NS. An introduction to kernel and nearest-neighbor nonparametric regression. *Am Statist* 1992;**46**:175–85. https://doi.org/10.1080/00031305.1992.10475879.
59. Schölkopf B, Smola AJ. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and beyond*. Cambridge, MA: MIT press, 2002. https://doi.org/10.7551/mitpress/4175.001.0001.
60. LeCun Y, Bottou L, Bengio Y. *et al*. Gradient-based learning applied to document recognition. *Proc IEEE* 1998;**86**:2278–324. https://doi.org/10.1109/5.726791.
61. Hochreiter S, Schmidhuber J. Long short-term memory. *Neural Comput* 1997;**9**:1735–80. https://doi.org/10.1162/neco.1997.9.8.1735.
62. Liu B, Jinghao X, Lan X. *et al*. iDNA-Prot—Dis: identifying DNA-binding proteins by incorporating amino acid distance-pairs and reduced alphabet profile into the general pseudo amino acid composition. *PloS One* 2014;**9**:e106691. https://doi.org/10.1371/journal.pone.0106691.
63. Berman HM, Westbrook J, Feng Z. *et al*. The Protein Data Bank. *Nucleic Acids Res* 2000;**28**:235–42. https://doi.org/10.1093/nar/28.1.235.
64. Wang G, Dunbrack RL. Pisces: recent improvements to a PDB sequence culling server. *Nucleic Acids Res* 2005;**33**:W94–8. https://doi.org/10.1093/nar/gki402.
65. Japkowicz N. The class imbalance problem: significance and strategies. In: *Proc. of the Int'l Conf. on artificial intelligence*, Vol. 56, pp. 111–7, 2000.
66. He H, Garcia EA. Learning from imbalanced data. *IEEE Trans Knowl Data Eng* 2009;**21**:1263–84. https://doi.org/10.1109/TKDE.2008.239.
67. Li W, Godzik A. CD-HIT: a fast program for clustering and comparing large sets of protein or nucleotide sequences. *Bioinformatics* 2006;**22**:1658–9. https://doi.org/10.1093/bioinformatics/btl158.
68. Wang LR, Wong L, Goh WWB. How doppelgänger effects in biomedical data confound machine learning. *Drug Discov Today* 2022;**27**:678–85. https://doi.org/10.1016/j.drudis.2021.10.017.
69. Yang M, Huifen L, Liu J. *et al*. lncRNAfunc: a knowledgebase of lncRNA function in human cancer. *Nucleic Acids Res* 2022;**50**:D1295–306. https://doi.org/10.1093/nar/gkab1035.
70. Zhang J. Evolution by gene duplication: an update. *Trends Ecol Evol* 2003;**18**:292–8. https://doi.org/10.1016/S0169-5347(03)00033-8.
71. Uniprot: The universal protein knowledgebase in 2023. *Nucleic Acids Res* 2023;**51**:D523–31. https://doi.org/10.1093/nar/gkac1052.
72. Chicco D, Jurman G. The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genomics* 2020;**21**:1–13. https://doi.org/10.1186/s12864-019-6413-7.
73. Wilcoxon F. Individual comparisons by ranking methods. In: Kotz S, Johnson NL (eds), *Breakthroughs in Statistics: Methodology and Distribution*. New York, NY: Springer, 1992, 196–202. https://doi.org/10.1007/978-1-4612-4380-9_16.
74. Smith TF, Waterman MS. *et al*. Identification of common molecular subsequences. *J Mol Biol* 1981;**147**:195–7. https://doi.org/10.1016/0022-2836(81)90087-5.
75. Kipf TN, Welling M. Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907. 2017. https://arxiv.org/abs/1609.02907.
76. Velickovic P, Cucurull G, Casanova A. *et al*. Graph attention networks. *Stat* 2017;**1050**:10–48550.
77. Remmert M, Biegert A, Hauser A. *et al*. HHblits: lightning-fast iterative protein sequence searching by HMM-HMM alignment. *Nat Methods* 2012;**9**:173–5. https://doi.org/10.1038/nmeth.1818.
78. Pascanu R, Mikolov T, Bengio Y. On the difficulty of training recurrent neural networks. In: *International Conference on Machine Learning*, pp. 1310–8. Atlanta, Georgia, USA: PMLR, 2013.
79. Greff K, Srivastava RK, Koutník J. *et al*. LSTM: a search space odyssey. *IEEE Trans Neural Netw Learn Syst* 2016;**28**:2222–32. https://doi.org/10.1109/TNNLS.2016.2582924.
80. Lin J, Keogh E, Lonardi S. *et al*. A symbolic representation of time series, with implications for streaming algorithms. In: *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, pp. 2–11. New York, NY, USA: Association for Computing Machinery, 2003.
81. Bahdanau D, Cho K, Bengio Y. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473. 2014. https://doi.org/10.48550/arXiv.1409.0473.

82. Luong M-T, Pham H, Manning CD. Effective approaches to attention-based neural machine translation. arXiv preprint arXiv:1508.04025. 2015.
83. Villegas-Morcillo A, Gomez AM, Sanchez V. An analysis of protein language model embeddings for fold prediction. *Brief Bioinform* 2022;**23**:bbac142. https://doi.org/10.1093/bib/bbac142.
84. Weissenow K, Heinzinger M, Rost B. Protein language-model embeddings for fast, accurate, and alignment-free protein structure prediction. *Structure* 2022;**30**:1169–1177.e4. https://doi.org/10.1016/j.str.2022.05.001.
85. Zhang Y, Lin J, Zhao L. *et al*. A novel antibacterial peptide recognition algorithm based on BERT. *Brief Bioinform* 2021;**22**:bbab200. https://doi.org/10.1093/bib/bbab200.
86. Yuan Q, Sheng Chen Y, Wang HZ. *et al*. Alignment-free metal ion-binding site prediction from protein sequence through pretrained language model and multi-task learning. *Brief Bioinform* 2022;**23**:bbac444. https://doi.org/10.1093/bib/bbac444.
87. Jumper J, Evans R, Pritzel A. *et al*. Highly accurate protein structure prediction with AlphaFold. *Nature* 2021;**596**:583–9. https://doi.org/10.1038/s41586-021-03819-2.
88. Bertoline LMF, Lima AN, Krieger JE. *et al*. Before and after AlphaFold2: an overview of protein structure prediction. *Front Bioinform* 2023;**3**:1120370. https://doi.org/10.3389/fbinf.2023.1120370.
89. Hastie T. The elements of statistical learning: data mining, inference, and prediction. 2009.
90. Cervantes J, Garcia-Lamont F, Rodríguez-Mazahua L. *et al*. A comprehensive survey on support vector machine classification: applications, challenges and trends. *Neurocomputing* 2020;**408**:189–215. https://doi.org/10.1016/j.neucom.2019.10.118.
91. Louppe G, Wehenkel L, Sutera A. *et al*. Understanding variable importances in forests of randomized trees. *Advances in neural information processing systems* 2013;**26**.
92. Rahman CR, Amin R, Shatabda S. *et al*. A convolution based computational approach towards DNA n6-methyladenine site identification and motif extraction in rice genome. *Sci Rep* 2021;**11**:10357. https://doi.org/10.1038/s41598-021-89850-9.
93. Van der Laan MJ, Polley EC, Hubbard AE. Super learner. *Stat Appl Genet Mol Biol* 2007;**6**. https://doi.org/10.2202/1544-6115.1309.
94. Sill J, Takács G, Mackey L. *et al*. Feature-weighted linear stacking. arXiv preprint arXiv:0911.0460. 2009. https://doi.org/10.48550/arXiv.0911.0460.
95. Yan J, Kurgan L. DRNApred, fast sequence-based method that accurately predicts and discriminates DNA-and RNA-binding residues. *Nucleic Acids Res* 2017;**45**:gkx059–e84. https://doi.org/10.1093/nar/gkx059.
96. Zhang J, Basu S, Kurgan L. HybridDBRpred: improved sequence-based prediction of DNA-binding amino acids using annotations from structured complexes and disordered proteins. *Nucleic Acids Res* 2024;**52**:e10–0. https://doi.org/10.1093/nar/gkad1131.
97. Batista GEAPA, Prati RC, Monard MC. A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD Explor Newslett* 2004;**6**:20–9. https://doi.org/10.1145/1007730.1007735.
98. He H, Garcia EA. Learning from imbalanced data. *IEEE Trans Knowl Data Eng* 2009;**21**:1263–84. https://doi.org/10.1109/TKDE.2008.239.
99. Yen S-J, Lee Y-S. Cluster-based under-sampling approaches for imbalanced data distributions. *Exp Syst Appl* 2009;**36**:5718–27. https://doi.org/10.1016/j.eswa.2008.06.108.
100. Kulmanov M, Hoehndorf R. DeepGoPlus: improved protein function prediction from sequence. *Bioinformatics* 2020;**36**:422–9. https://doi.org/10.1093/bioinformatics/btz595.
101. Armenteros JJA, Sønderby CK, Sønderby SK. *et al*. DeepLoc: prediction of protein subcellular localization using deep learning. *Bioinformatics* 3395, 2017;**33**:3387.
102. Du X, Sun S, Hu C. *et al*. DeepPPI: boosting prediction of protein–protein interactions with deep neural networks. *J Chem Inf Model* 2017;**57**:1499–510. https://doi.org/10.1021/acs.jcim.7b00028.
103. Luo F, Wang M, Liu Y. *et al*. DeepPhos: prediction of protein phosphorylation sites with deep learning. *Bioinformatics* 2019;**35**:2766–73. https://doi.org/10.1093/bioinformatics/bty1051.

Briefings in Bioinformatics, 2025, **26(1)**, bbae634
https://doi.org/10.1093/bib/bbae634
Review