Comprehensive evaluation of plant transcription factors binding sites discovery tools

Jyoti[1,2], Ritu[1,2], Sagar Gupta[1,2], Ravi Shankar*[1,2]

[1] Studio of Computational Biology & Bioinformatics,

The Himalayan Centre for High-throughput Computational Biology,

(HiCHiCoB, A BIC supported by DBT, India),

CSIR-Institute of Himalayan Bioresource Technology (CSIR-IHBT),

Palampur (HP), 176061, India.

[2]Academy of Scientific and Innovative Research (AcSIR),

Ghaziabad, Uttar Pradesh- 201002

Authors' email addresses:

Jyoti: dshwljyoti@gmail.com

Ritu: deshwalritu18@gmail.com

SG: sagargupta47321@gmail.com

RS: ravish@ihbt.res.in

*To whom correspondence should be addressed- Email-ravish@ihbt.res.in

Abstract

Transcription factors (TFs) are regulatory proteins that bind to a specific DNA sequence, known as the transcription factor binding site (TFBS), to regulate the rate of transcription process. For a better understanding of these regulatory proteins, it is crucial to identify these TFBSs across the genomic sequences. The identification of TFBS has been made possible by a number of experimental and computational techniques established during the past few decades. These existing approaches are skewed heavily around human TFBS discovery. The area of TFBS identification in plants is lagging behind due to a lack of experimental data for TFs and very scarce attempts to build plant-specific TFBS identification models. Therefore, to facilitate a better understanding of tools to identify TFBS in plants, here we provide a comprehensive survey of a large number of TFBS identification tools. We have also provided a repository along with corrected scripts, which also has datasets used in the present study. We expect this study to serve as a guide to better the understanding of plant TFBS tools and the care to be taken while applying them, especially during cross-species applications. The corrected scripts of these software are made available at https://github.com/SCBB-LAB/Comparative-analysis-of-plant-TFBS-software.

Keywords: Transcription factors, Transcription factor binding sites, Machine Learning, Deep learning, TF-DNA binding specificity

Introduction

The precise pattern of gene regulation is controlled by transcription factors (TFs) which bind to specific, short DNA sequences (approximately five to 31 bp), known as transcription factor binding sites [1, 2]. Transcription factors (TFs) recognize these cis-regulatory elements in the promoter regions of the target genes to either activate or inhibit their expression [2]. Hundreds of TFs are active throughout the complex process of transcriptional regulation, where they can operate alone or cooperatively to control the expression of a number of genes while binding upstream of the target genes [3, 4].

A TF can bind numerous TFBSs throughout the genome in the regulatory regions through its functional domain, which also determines the specificity of the binding [5]. The information from sequence similarity and their tertiary structure with a DNA-binding domain is frequently used to categorize the transcription factors [6]. The majority of plant TFs contain a DNA-binding region, an oligomerization site, a transcription regulatory domain, and a nuclear localization signal. Yet, several of the reported ones either lack a transcription regulatory domain [7] or possess a particular DNA-binding region that interacts with DNA indirectly via co-operative binding with other TFs [8]. The characteristics of one or more DNA-binding domains that are present in a TF are employed to categorize them into different families [9, 10].

In plant biology, *Arabidopsis* is a model plant that is widely studied. Around 2,296 TFs have been recognized and categorized into 58 families in accordance with the family assignment rules. Among these families, a large number of members belong to bHLH (Basic helix-loop-helix), MYB (Myeloblastosis), ERF (Ethylene-Responsive Element binding Factor), NAC (NAM, ATAF1/2, and CUC2), and bZIP (basic leucine zipper), which contain 225, 168, 139, 138, and 127 TFs, respectively. Whereas, the smallest number of TFs belong to LFY (LEAFY), NZZ/SPL

(Sporocyteless), and SAP (STERILE APETALA) which have only one member, respectively [11]. Compared to human transcription factors, experimental data on plant transcription factors, their binding locations, and interactions are much scarce. It is challenging to develop and test computational models for TFBS discovery due to the data shortage. It is only possible to develop species-specific techniques for TFBS discovery in plants because the binding regions for transcription factors differ greatly between plant species. In contrast, binding regions for animal TFs are usually more conserved and easier to recognize. Due to heterogeneity of TFBS, the limited genomic resources for most of the plant species, and the complexity and diversity of plant genomes, all contribute to the difficulties in identifying plant TFBSs.

Various studies have shown why different TFs bind to particular DNA regions with different binding affinity in particular ways [12]. Some studies have shown that there is no one-to-one correspondence between amino acid residues and nucleotide sequences in protein-DNA complexes [13]. This has also been proved by sequential substitution of bases in the binding sites using *in silico* tiling deletion [14] and mutagenesis techniques [15]. Additionally, the presence of co-factors, cooperative TF-DNA binding [16], chromatin accessibility [17], nucleosome occupancy [18], indirect co-operativity via competition with nucleosomes [19], pioneer TFs [20], and DNA methylation [21], are also known to affect TF-DNA binding specificity. TFs may also detect the structural features of their binding sites, such as sequence-dependent DNA bending and unwinding [22], commonly referred to as "shape readout" [23]. It can be challenging to comprehend how these components interact, but doing so is essential in order to understand how TFs regulate gene expression [24].

Despite the challenging aspects of the TFBS search problem, the TFBSs were mostly determined experimentally through methods like DNA Footprinting [25] and SELEX [26]. Since then, the support for TF-DNA binding preferences has been confirmed by a number of *in vivo* and *in vitro*

experimental techniques for different species [26]. The *in vitro* methods include Systematic Evolution of Ligands by EXponential enrichment (SELEX) and protein-binding DNA microarrays (PBM) [27], which use an organized and directed approach. These technological advancements allow genome-wide profiling of TFBS (summarized in **Supplementary Material S1 (Table 1))**. PBM uses a randomized DNA microarray on which the DNA-bound protein is assayed using fluorescent labeled antibodies. Some of the genomic methods are performed *in vivo,* such as chromatin immunoprecipitation of bound DNA, followed by either hybridization (ChIP-Chip) [28] or sequencing (ChIP-seq) [29]. These methods highlight protein-DNA interactions in their natural biological context.

With promising results from the standard ChIP-seq, its more effective variants also arrived. Chromatin immunoprecipitation followed by exonuclease digestion and sequencing (ChIP-exo) [30] optimizes the resolution of binding site mapping by combining the advantages of ChIP-seq with exonuclease digestion. This results in sharper peak calling and more precise identification of TFBS. ChIP-nexus [31] provides higher resolution than standard ChIP-seq, enabling more accurate positioning of protein-DNA interactions. In order to accomplish this, an additional stage of exonuclease-mediated DNA fragmentation is incorporated, which produces shorter fragments of DNA and optimizes the precision of binding site identification. Another method, ChIPmentation, implements the direct attachment of a sequencing-compatible adaptor onto bead-bound chromatin. This is a relatively novel approach that combines ChIP and tagmentation [32].

Methods like DNase-seq [33] facilitates reliable, genome-wide identification of open chromatin domains and enables excellent spatial resolution to recognize unique open chromatin regions. By highlighting potential regulatory components, such as promoters, enhancers, and other regulatory areas [34]. DAP-seq provides simple static images of TF binding sites [35]. However, it provides no insight into how TF binding influences or responds under diverse circumstances [36]. In order to

fully understand the dynamics of TF binding, other tests, such as time-course or condition-specific assays [34], such as DNase-seq and MNase-seq [37] on DAP-seq datasets are conducted. ChIP-seq and DAP-seq have significantly contributed to the generation of data for TF-DNA interactions, which can be used to build TF-DNA interaction models of high accuracy. A timeline of all these experimental techniques and the computational advancement in the identification of TFBS methods over the years are shown in **Figure 1.** As described and illustrated in **Figure 1**, TFBS analysis has evolved progressively as a consequence of advancements in experimental approaches, technological breakthroughs, and the accessibility of huge amounts of genomic data.

For the identification of regulatory regions and binding sites, experimental procedures and verification of such components are very tedious, costly, and time-consuming jobs. Significant progress has been made into the development of computational techniques. Effective computational techniques may be able to identify TFBSs with high accuracy, which shortens the time and cost needed by the experimental approaches. One of the first computational resources for TFBS identification was a program written in 1977 by Korn et al. [38] that was capable of finding sequence similarities in the regions that were directly upstream of transcription start sites (TSS). Only paired comparisons were used to account for both flexible gaps and mismatches. A number of approaches were further developed for TFBS discovery based on position weight matrices (PWM), such as the MEME suite [39], including DREME [40], EXTREME [41], and BioProspector [42]. The ability to communicate concepts accurately and computational efficiency are two key trade-offs in TFBS identification. However, when applied to solve real-world problems, probabilistic motif discovery methods do not ensure the delivery of the global optimum. Iterative refinement and stochastic optimization are often the foundations of these techniques. The iterative refining technique most frequently utilized is the expectation maximization (EM) [43]. Mostly, they generate PWM to scan the sequences, which is the most common method to characterize the DNA binding sequences due to its simplicity and understandability [40]. Gibbs sampling, paired with generic

Metropolis-Hastings, is a stochastic optimization method most frequently employed for motif identification [44]. They have some limitations when looking at inter-nucleotide dependency, and they also do not tackle the weak motif patterns of many TFs while exhibiting poor specificity. Moreover, these approaches completely ignore the importance of context and structural information, which have now been recognized as essential [45].

To counter the limitations of probabilistic motif-based approaches, machine learning-based (ML) methods have been invented, which build upon the feature representation of instances from the datasets. However, the traditional ML tools restrict their capacity to learn complex information about the base pair interactions that are requisite for TF-DNA binding and contextual information. Traditional ML approaches implement manually crafted features rather than self-extracting, engineered, and hidden features. In addition to it, when dealing with large-scale genome-wide datasets like ChIP-seq or DNase-seq data, several traditional TFBS discovery approaches can be computationally demanding. While the development of more sophisticated computational methods, including deep learning (DL) approaches, aims to overcome these challenges by performing parallel processing and simultaneously improving the identification performance [46]. Their performance increases sharply with the volume of training data from which they automatically learn strong and hidden features, which traditional machine learning approaches fail to achieve.

In recent years, deep learning algorithms have outperformed earlier methods in regulatory genomics. In this regard, DeepBind was the first deep learning approach to train a convolutional neural network (CNN) architecture to identify the specificity of DNA and RNA-binding proteins by using the available human TFBS dataset [47]. Moreover, another advance in a DL approach, KEGRU, integrated a Bidirectional Gated Recurrent Unit (GRU) network with *k*-mer embedding [48]. These tools were mainly developed for the human TFBSs identification. However, for plant-specific TFBS identification, very minimal development has happened. Only a few ML-based [49,

50] and DL-based [51–53] software have been developed to discover plant TFBS. Therefore, here we have tried to assess the existing software for animals and their suitability for plants, as well as evaluated the performance of the existing plants-based tools.

The main motivation of this study was to present a comprehensive study of these developed software covering their advantages and disadvantages. This analysis will help us to decide which tool to rely on for plant TFBS discovery and what issues the existing tools set have. In this study, we have concentrated on various ML and DL-based methods and evaluated a total of 15 different recent software across three different datasets. Many of these software were developed for animal systems. We have evaluated them for plant-specific datasets and assessed their scope for their use in plants as well. Unlike animal systems, which are mostly focused around humans or mice like model animals, plants pose a different problem as TFBSs have to be found across different species, where variability is also much higher [54]. In order to gain insights into how well these models properly function to capture TFBS across different species, which fathoms their actual applicability, we also evaluated the performance of the tools in a cross-species manner. In this evaluation, all the ML and DL-based tools we studied were initially trained using the TFBS datasets for *Arabidopsis thaliana* and subsequently tested on the TFBS dataset for the same TFs commonly found in *Zea mays* [55]. The overview of this entire study is illustrated in **Figure 2**.

Materials and Methods

Datasets sources

In the field of machine learning research, constructing a high-quality dataset is an essential and initial requirement. Datasets guide not only the quality of the raised model, but also their validity and generalization capabilities. Selection of suitable instances as positive and negative components, data cleansing, labeling, and resolving class imbalances are major parts of this crucial step. The

dataset is often separated into training, validation, and test subsets to ensure reliable model building and evaluation. In this direction, the primary step of this study was to develop benchmarking datasets. Therefore, we constructed three different types of datasets to perform verification, validation, and evaluation of all the existing tools for TFBS identification.

A significant quantity of TFBSs data has been produced, and many of them have been maintained in databases (**Supplementary Material S1 (Table 2)**). Multiple information on functional DNA sequences in the human genome that have been collected from various tissues and cell types, which are available through projects like ENCODE [56]. Similar to this, resources like Plant Cistrome Database [57], *Arabidopsis* RegNet [58], AGRIS [59], and PlantPAN [60] provide genomic data on TFs from several plant species. For a variety of TFs from various species, the databases HOCOMOCO [61] and JASPAR [62] include sizable collections of curated, computationally predicted, and empirically deduced TFBS motifs. For the purpose of this study, we gathered an extensive collection of data for plant and animal transcription factors from some of these data resources. Additionally, we used a large volume of DAP-seq data for *Arabidopsis thaliana* to fairly evaluate different software for the identification of TFBS in plants. This DAP-seq data was previously utilized by tools like TSPTFBS [51], PlantBind [53], and TSPTFBS 2.0 [63]. This dataset covered the peak data for 35 plant transcription factor families, as shown in **Figure 3** with (a) a line plot and (b) a pie chart.

For the first part of this study, the majority of the 690 ChIP-seq experimental datasets were retrieved from the ENCODE project, covering 161 TFs and 91 cell types for humans [56]. We collected datasets for plant species, including ChIP-seq [49, 52], MNase-seq [49], core promoter [49], DAP-seq data [57], and *Arabidopsis* RegNet [58]. This first part of the study was done primarily to verify the performance claims made by various tools. Since the tools developed for animals were also considered in this study for their possible use in plants, data from ENCODE for

animals was also considered. For the second part of the comparative study, *A. thaliana* DAP-seq TFBS experimental data were retrieved from the Plant Cistrome Database [57] and processed. This part of the study was mainly done for the model development and objective assessment of the plant-based tools as well as the animal-based tools on this plant dataset. For the third part of the comparative analyses, we used five common TFs between *Arabidopsis thaliana* and *Zea mays* [64] species. The reason to consider this third dataset was to study the performance of the pre-trained models on *Arabidopsis thaliana* in a cross-species manner and assess their usability for inter-species applications.

Datasets Creation

Various tools have been developed for different types and numbers of TFBSs data (**Supplementary Material S1 (Table 3**)). The first part of this study mainly focused on the dataset considered by all the 15 software evaluated here to verify their respective performance claims. Majority of these software were developed for mainly for human TFBS identification, such as DeepBind [47], gkmSVM [65], KEGRU and others. The TFBS ChIP-seq experimental data was downloaded from the ENCODE project and processed as per the documented methodologies of the respective software, as shown in **Figure 4(a)**. The raw peak files for plant TFBS-based tools were downloaded from the Plant Cistrome Database, plant ChIP-seq, MNase-seq (**Supplementary Table S1 (Sheet 1))**. This dataset contained sequences for *Homo sapiens*, *Arabidopsis thaliana,* and *Zea mays.* It formed the positive instances library for the dataset.

The preparation of negative instances dataset is a more critical step then that of the positive datasets. It is essential for model robustness, generalization, and real-world applications. The methodology to construct the negative data was different for each tool. The negative instances were created by DeepBind, deepRAM [66], and MAResNet [67] by randomizing the positive sequences while maintaining similar dinucleotide composition. While others, namely gkmSVM, LS-GKM

[68], KEGRU, WSCNNLSTM [69], DESSO [70], DNABERT [71], and *k*-mer grammar [49], used random genomic sequences with matching repeat and GC content to the positive dataset. All these tools except the *k*-mer grammar used the “genNullSeqs” function of gkmSVM. This “genNullSeqs” function identifies the nucleotide composition of a positive set of sequences. Following that, this function constructs a collection of genomic regions with nucleotide composition and sequence length that are comparable to those in the positive set. On the other hand, TSPTFBS, TSPTFBS 2.0, and PlantBind constructed their negative datasets by randomly extracting intergenic regions of the *Arabidopsis* genome (TAIR10). Tools like SeqConv [52] and Wimtrap [50] had selected random non-overlapping genomic sequences with the positive instances. In this entire study, this dataset has been named as Dataset “A”. The Dataset “A” was used to verify the original performance claimed by the software studied here.

To train and evaluate the models, data must be divided into training and testing sets. The models were first trained on their corresponding training sets and then carefully tested on the testing sets, where instances were independent and unseen. Wimtrap, an XGBoost-based approach, initially matches the motif provided with the given input ChIP-chip/seq data. On match, they are labeled as ‘positive’, else ‘negative’. For every TF-specific model, the train and test datasets components for each TF were divided into 80% and 20%, respectively. Wimtrap also employed 5-fold cross-validation, and each TF-specific dataset was split into five subsets where training and AUC computation were iterated five times, each time using a different subset for obtaining the ROC curve.

For DeepBind, the first deep learning-based approach for TFBS identification, from the overall positive and negative instances, only 500 instances for each were used as the test set. The rest were used to train the model. The strategy to split the peaks of the ChIP-seq dataset into training data and test data employed by DeepBind was also adopted by DESSO, deepRAM, and DNABERT.

Although the strategies to construct positive datasets were usually the same as those for DeepBind, the negative dataset creation for these tools were very different from that of DeepBind as described above. However, if we move to the next software based on an RNN-based approach, KEGRU, an equal number of negative samples were generated by matching the size, GC-content, and repeat fraction of the positive samples. This dataset was divided into three groups: training (85%), validation (5%), and testing sets (10%), which were created by randomly dividing the dataset into these three groups. Moving on to plant-based tools, such as TSPTFBS and TSPTFBS 2.0, both randomly divided their datasets in ratio of 80:20 for the training and testing datasets. For SeqConv, the proportion of the train:test datasets was 90:10.

For CNN-RNN-based TFBS identification software, WSCNNLSTM, the negative data was generated by matching the repetitive content, length, and GC content of the positive data. Additionally, during training, 1/8th of the training data was randomly selected as the validation data. Another tool, MAResNet, had the positive and negative datasets in an 80:20 ratio. The training dataset was further divided into training and validation sets in a ratio of 80:20. In the case of PlantBind, a positive dataset was constructed by combining all the peaks. The negative dataset was constructed by extracting random intergenic regions from the genome. These datasets were used as training, validation, and testing sets.

In the next part of the study, there was a need to evaluate these software on a common dataset, mainly plant-specific TFBS data, to objectively assess their performance and suitability for plants. Therefore, we retrieved DAP-seq data for 455 plant transcription factors from the Plant Cistrome Database for *Arabidopsis thaliana*. We first refined the data by implementing the method described by Liu et al. (2021) [51] to gain high-confidence positive peaks for the TFBSs (Figure 4(b)). They had overlapped these DAP-seq peaks with DNase-seq sites (DHS) to gain insights into relationship between chromatin accessibility [33]. It has to be noted that DHSs are the regions that emphasize

open chromatin regions that are more favorable for TF binding. Therefore, after filtering, TFBS datasets containing unique peaks and inaccessible *in vivo* regions were discarded. After this procedure, only DAP-seq data for 265 TFs were considered as the components for the positive dataset, which were used for training and evaluation purposes. The positive instances ranged from 1,024 to 11,990 peaks, covering a total of 9,11,342 binding regions.

In order to build a more natural and robust model, it is very essential to develop a refined negative dataset for all individual TFs [45, 72]. According to the studies, the formation of a strong negative dataset alone leads to a very significant impact on the performance and reliability of the model. A mere negligence to construct a proper strong negative dataset may cause biases and a misleading model. Therefore, to build a strong negative dataset, we used a big pool of 1,89,799 genomic background sequences, which consisted of random intergenic regions of the *Arabidopsis thaliana* genome of TAIR10 [51]. To refine this negative dataset, all the overlapping regions between negative and positive instances for each of the TFs were removed. These non-overlap regions were then used as the negative instances for each TF. Most of the existing published software have grossly neglected this and have built datasets where positive and negative instances overlap, compromising the reliability of the model and reported performance.

Another factor which influences model's performance is an imbalanced dataset where one class out-weights other class, resulting in a bias and class imbalance. Therefore, to solve this imbalance problem, a consistent sequence length and sample number for positive and negative instances were maintained for each of the 265 TFs. In order to ensure no memory and bias, not a single instance of the training and testing datasets overlapped. In this entire study, this dataset has been named as Dataset "B" (**Supplementary Table S1 (Sheet 2)).** However, to evaluate the performance of every software on this dataset, 70% of the dataset was used for training purposes, and 30% was kept aside

as a totally unseen test set to ensure unbiased performance testing with no scope for memory from data memorization.

At the time of model training of the tools on Dataset “A” and Dataset “B”, the models picked up knowledge from the training instances and optimized their parameters to reduce the classification errors on the validation and test datasets. During the training phase, the model's hyper-parameters were also adjusted to achieve optimal performance and generalization of the model. It acts as an objective assessment set to monitor the model's development and avoid overfitting. One can decide on hyper-parameter values after assessing the model's performance on the validation dataset. The hyper-parameters must be set prior to training an ML model because they determine the model's architecture. The ML algorithms’ capability to fit the model according to the data that is influenced by the hyper-parameters optimization function. This function is made up of a number of hyper-parameters that are set before the training process. A process of hyper-parameter settings that achieve the best performance on the data is called hyper-parameter optimization or tuning [73]. The two primary categories of hyper-parameter optimization techniques are manual search and automated search techniques. Manual search manually tests different hyper-parameter settings. In automatic search strategies, such as grid search [74], random search algorithms [75], and Bayesian hyper-parameter search [76], have been proposed to address the limitations of manual search. To improve ML models using hyper-parameters optimization, it is first necessary to identify the optimal hyper-parameters that must be adjusted in order to adapt the ML models to particular datasets. Along with hyper-parameter optimization, the validation dataset also helps in improving how well the model performs on unobserved data. The test dataset is a completely independent dataset that is not utilized in the development of the model or in the fine-tuning of the hyper-parameters. It operates as the model's last assessment set to determine how well it performs on hypothetical data and gauge how generic it is.

In addition to Datasets “A” and “B”, one more independent test dataset was raised and named as Dataset “C”. This dataset was constructed to examine the generalizability of the build models for cross-species TFBS identification. Pre-trained models may be extremely beneficial in the discovery of inter-species TFBS by leveraging features learned from one species to help in the identification of TFBS in another. These models capture important sequence patterns, motifs, and various levels of features from the input data within their layers. Particularly for highly conserved TFBSs, these features might be able to extend across species limitations. The detailed description of how to construct the Dataset “C” is shown in **Figure 4(c) and Supplementary Table S1 (Sheet 3)**. As shown in this figure, both positive and negative datasets were used as sources for testing purposes to attain an objective comparative evaluation. Due to data limitations for plant species, we found only five common TFs, namely LHY1 (MYB-related), MYB62 (MYB), MYB81 (MYB), MYB88 (MYB), and WRKY25 (WRKY), between *A. thaliana* and *Zea mays* (detailed in **Supplementary Table S1 (Sheet 3))**. In the processing of Dataset “C” (shown in **Figure 4(c))**, the ChIP-seq dataset for these common TFs for *Zea mays* were considered as the positive data. The negative dataset was constructed randomly from the intergenic regions while ensuring no overlap with the positive instances.

Survey and categorization of software to identify transcription factor binding sites

The recently developed computational approaches and algorithms to identify TFBS can be classified into two categories: Traditional machine learning approaches and deep-learning approaches. The traditional machine learning learns from manually extracted the features through structured inputs. The traditional ML approaches can be further classified into several algorithms. However, the most common types are support vector machines, decision trees, and Ensemble algorithms.

On the other hand, Deep-learning approaches are the most recent ones. They are founded on neural networks and differ from each other by means of their architecture and how they approach the input. Also, they automatically extract features and detect long range relationships and hidden features that are otherwise difficult to detect by manual approaches, such as traditional feature extraction methods. The deep-learning methods can be categorized into six major categories: (i) Plain feed-forward networks, (ii) Convolutional neural networks, (iii) Complex convolutional neural networks, (iv) Recurrent neural networks; (v) Transformers, and (vi) Hybrid architecture which involves various combinations of above mentioned major architectures. Fully connected feed-forward networks (FFNs) are integrated parts of all DL architectures. For this study, we attempted 15 ML and DL architectures for ChIP-seq and DAP-seq data, containing data from three species: humans, *Arabidopsis thaliana*, *Zea mays*. These considered tools were categorized and are briefly explained below according to their algorithms.

Traditional ML-based TFBS identification software

The term "traditional machine learning", a fundamental subfield of artificial intelligence, refers to a collection of statistical learning approaches for predicting and categorizing data based on well-defined features and established algorithms, some of which are mentioned above. Contrary to the recent deep learning techniques, traditional machine learning involves careful manual features extraction based on domain expertise and a variety of algorithms (**Figure 5 (a and b))**. Based on the kind of learning algorithms applied, the TFBS finding tools are discussed separately below.

i) gapped-kmer-SVM (gkmSVM): In order to account for variability in sequence and length of TFBS, gapped k-mers were proposed to represent longer motifs as *k*-mers with gaps at non-informative or degenerate TFBS positions. With the use of gapped k-mers as features, gkmSVM enhances kmer-SVM to train SVM kernels. It is a binary classifier implemented in C++ and R

programming languages that learns to split the input space with a hyper-plane boundary. Unlike traditional *k*-mer-based approaches that consider only contiguous *k*-mers, gkmSVM classifier incorporates gapped *k*-mers, which allows it to capture more complex patterns that may be separated by indels or mismatches within sequences and hence have the ability to capture biologically relevant information. A new tree algorithm, named iDL-bound, was implemented, which speeds up the computation. This tool implements three steps: (i) constructing the kernel matrix, (ii) training the SVM model, and finally (iii) sequence classification. For the first stage, kernel matrix scores are calculated by the pairwise similarity scores for all the sequences in the positive and negative sequences, done by the "gkmsvm_kernel" function. The second step is to train the SVM model using the "gkmsvm_train" function. Finally, the sequences are classified using the "gkmsvm_clasify" function. In the absence of a negative dataset or for larger negative sets, this tool also includes a new function, "genNullSeq", which samples the genome to generate a GC, length, and repeat fraction-matched negative sequence set. "gkmsvm_delta" function is used to calculate the score to check the impact of a variant.

ii) LS-GKM: It is an improved version of gkmSVM. It works by building an SVM classifier that can discriminate between regulatory and non-regulatory genomic sequences in the *k*-mer or gapped *k*-mer frequency feature vector space (**Figure 5 (a))**. A kernel matrix (also known as a Gram matrix) is evaluated during SVM training. This matrix is an n-by-n matrix that contains all potential inner products (also known as kernel functions) between a collection of vectors of "*n*" training samples. Direct calculation of the kernel matrix using gapped *k*-mers as features, however, soon becomes unfeasible as the number of samples rises. To fix this problem, gkm-SVM uses an effective technique that computes a complete kernel matrix with a run-time that grows linearly with "*n*" rather than "n^2". The pre-calculated kernel matrix and conventional SVM training techniques are then used to build an SVM classifier. On large-scale sequence datasets, LS-GKM optimizes the algorithm for scalable SVM training with gapped *k*-mers. Additional kernels for SVM training are

provided by LS-GKM. To address this issue, LS-GKM was developed by implementing gapped *k*-mer kernel functions within the LIBSVM framework [77]. A critical step in an SVM-based algorithm is the construction of the kernel matrix, and LS-GKM provides an effective approximation method for this purpose. Consequently, the computational complexity and memory requirements of LS-GKM are substantially lowered by this approximation strategy, making it computationally effective even for large-scale DNA sequence datasets. Additionally, it introduces expansions of the gkm-kernel, which is described as the radial basis function in the space of gapped *k*-mer frequency vectors, analogous to the fundamental RBF kernel. The most ChIP-seq and DNase-seq signals are concentrated in the middle areas within peaks, which is the inspiration for the second choice, known as center-weighted gkm-kernel or wgkm-kernel. The gapped k-mers in this new kernel are variably weighted according to how far from the peak's center they are. The final kernel, wgkmrbf-kernel, results from the combination of the first two. Multi-threading functionality is also implemented in LS-GKM for further speedup. Most notably, training LS-GKM on much larger datasets takes less time and memory than training gkmSVM.

iii) *k*-mer grammar: By using potential transcription factor binding sites within genomic regions, Mejía-Guerra and Buckler (2019) created a computational framework which is based on *k*-mer sequence features. By this tool, two different natural language processing approaches were applied: (1) “bag-of-words” which is commonly used for differentially weighting key words in tasks like sentiment analyses, and (2) a vector-space model using word2vec (vector-*k*-*mer*s), which captures semantic and linguistic relationships between words. On the basis of these approaches, they built “bag-of-*k*-mers” and “vector-*k*-mers” models, which help to categorize regulatory regions from non-regulatory regions. According to the authors of this tool, "bag-of-*k*-mers" models performed better overall, but the “vector-k-mers” models were more effective in identifying important groups of sequences inside regulatory regions. To construct the classifier, the “bag-of-*k*-*mer*s” model was implemented that used the *k*-mer individual frequencies to extract the information and fits a logistic

regression (**Figure 5 (b))** to a matrix filled with TF*IDF (the term frequency-inverse document frequency) transformation containing the raw counts per sequence [78]. In contrast, "vector-*k*-mers" gather data from *k*-mer co-occurrences by training a shallow neural network that figures out how likely each *k*-mer is given its context (window = 5). Each sequence in the "vector-*k*-mers" models is defined as a collection of "sentences" produced by walking "*k*" times and sliding by one base pair. Each phrase is divided into a set of sequential, distinct new words. In order to acquire regulatory and non-regulatory likelihoods for groups of *k*-mers, phrases are separated into neighborhoods for testing. In order to create "vector-*k*-mers" models, word2vec algorithms were implemented, which optimize an objective function by fitting sequence representations (*k*-mer vectors) via stochastic gradient descent (SGD), which implicitly corresponds to the likelihood for *k*-mer co-occurrences. The sequence representations (*k*-mers) can be transformed by inversion using the Bayes rule to estimate the likelihood that a new sequence is a component of a regulatory region based on its *k*-mer composition, as demonstrated for text categorization. This classification scheme reads each individual *k*-mer as a component of a composite likelihood approximation, allowing sequences to be classified without the need for further modeling or estimate stages. To understand the significance of important sequence elements and the interactions between sequence features, the pre-trained models were used on TF binding loci and core promoter-related species in different maize lines and two species (sorghum and rice) to gain insights into how the models correctly distinguished between random and regulatory regions.

iv) Wimtrap was constructed as a novel, simplified software that aims to predict cis-regulatory elements (CREs) and target genes of plant TFs in specific organs or conditions. This method uses an XGBoost-based approach (an ensemble machine learning) implemented in the R programming language, enabling the development of decision rule models using experimental TF ChIP-chip/seq data. Different layers of genomic features are integrated in the predictive models: the position on the gene, DNA sequence conservation, chromatin state, and other CRE imprints. Wimtrap takes

only positive data as an input in the form of the coordinates of the TF peaks. Chromatin characteristics were given high weight here to achieve better accuracy. A five-fold cross-validation was employed to report the model's performance. The established features form the basis for the pairwise correlations between them. The feature with the highest mean absolute correlation with the other features is eliminated if two features exhibit a correlation greater than 95%. This method is mainly focused on plants, especially on *A. thaliana*, the model species for plant molecular biology, also for which data are the most abundant.

Deep learning-based TFBS identification software

While the traditional machine learning approaches are still very relevant and extensively utilized, it is important to note that deep learning, a subset of machine learning, has attracted a lot of attention and gained popularity recently owing to its exceptional performance in a variety of applications. This has greatly motivated the research community to apply them to molecular biology. Deep learning frameworks may be categorized into five different categories: (i) CNN-based models (**Figure 5 (c)**), (ii) Complex CNN-based models (**Figure 5 (d, e)**), (iii) RNN-based models (**Figure 5 (f)**), (iv) Transformers-based models (**Figure 5 (h)**), and (v) Hybrid DL models (**Figure 5 (g)**). The tools based on DL models are listed in **Supplementary Material S2 (Table 1)** with their respective hyper-parameters used during the model training process.

1) CNN-based TFBS identification software

The convolutional neural network (CNN) is a type of feed-forward neural network with convolution estimation and depth architecture [79]. It has been utilized effectively in identifying images, video analysis, natural language processing, and other fields with good results [80, 81]. CNN typically works by inputting image information and passing it through a combination of convolutional layers, non-linear layers, pooling layers, and complete connection layers to get the final output result. The convolutional layer is largely responsible for feature extraction via the convolutional kernel scan,

whereas the pooling layer is primarily responsible for feature selection and information filtering. Because of its strong feature extraction capacity, CNN has been thought to have the capacity to identify the motif in the process of searching for TFBS. Below are a few tools which are based on CNN architecture:

i) DeepBind is based on the CNN framework and implemented in the python programming language. It was the first approach that used CNN (**Figure 5(c))** to identify DNA/RNA-binding sites across DNA/RNA sequences from high-throughput data, including PBM, SELEX, ChIP- and CLIP-seq data. DeepBind was developed for several advantages over traditional machine learning algorithms while banishing the requirement of manual feature engineering by automatically learning to extract key characteristics from the input sequences. This may be extremely helpful when working with large sequence datasets, since the algorithm is capable of finding relevant sub-sequences or patterns on its own. It also learns and derives hidden features and their associations in space. Spatial pattern detection is the forte of the CNN approach, which has given extremely superior results for motif discovery jobs. In the architecture of DeepBind, the data initially passes through the convolutional layer to extract fused features, followed by the pooling layer to optimize the features. Finally, the features pass through the activation function to produce the prediction result as an output, compare it to the target to determine the loss, and update the weight to improve prediction accuracy. The model provides a probability score that represents the likelihood of protein binding after accepting raw nucleotide sequences converted into one hot encoding (OHE) matrix as input. The Convolution layer scans a set of motif detectors to scan the sequence OHE matrix for feature extraction. The output array of convolution is fed into the rectification layer as an input array. Here, it computes a simple absolute value operation on the features extracted by the convolutional layer. After convolution and rectification, the pooling layer influences the robustness of extracted features and reduces the degree of overfitting. Finally, these inputs are fed into a fully connected (FC) layer and a score is calculated. The convolution layer has trainable motif detectors

that scan for the motif throughout the input sequence. A rectification layer, which is used to isolate positions with good pattern matches. The pooling layer calculates the maximum and average values of each motif detector's rectified response across the sequence. These values are fed into a nonlinear neural network with weights, which produces a score by combining the individual nodes' scores from the previous layer. The learning rate, momentum level, mini-batch size, parameter regularization strength, and dropout probability are just a few of the hyper-parameters to be optimized. DeepBind's training pipeline automatically adjusts, reducing the need for manual tweaking.

ii) DESSO (DEep Sequence and Shape mOtif): This framework was developed to accept both the OHE matrix of DNA sequences and DNA shape features as the input to identify sequence and shape motifs, respectively [82]. It is implemented in the python programming language. It enables the extraction of more complex motif patterns compared to existing motif prediction methods owing to its multi-layer network architecture. They designed a model in DESSO to identify all the significant TFBSs under the statistical hypothesis that the number of random sequence segments that contain the motif of interest in the human genome is binomially distributed. The first layer of the CNN model has numerous convolutional filters (denoted as motif detectors), which were utilized to detect low-level characteristics from given ChIP-seq peaks. Using the output from the convolutional layer, high-valued features were obtained by the following max pooling layer and a fully connected layer. In order to remove any possible biasness carried out by these features, a sizable set of background sequences from the human genome were selected while taking into consideration GC content, no overlaps with peaks in the positive dataset, CG frequency, promoter and repeat overlaps. The query sequence matrix and the background sequence matrix were obtained. The TFBS instance candidates in these metrics were used to compute a p-value using the binomial distribution. Finally, the optimal motif instances for a motif detector were identified as the motif instance candidates in the query

sequence matrix with the lowest p-value. The ReLU (rectified linear unit) activation function was used. ReLU performs better at convergence and can prevent the gradient vanishing problem.

iii) TSPTFBS (trans-species prediction of transcription factor binding sites), another CNN-based method **(Figure 5(c)),** which was designed to identify the TFBSs from the DAP-seq experimental dataset for *A. thaliana*. It has been implemented in python programming language. In this study, an advanced CNN approach [47] was used to model a large-scale DAP-seq TFBS dataset (265 TFs) in *Arabidopsis thaliana* [57]. The CNN model contains one convolution, a rectified linear unit (ReLU), max pooling, and three fully connected layers, which can identify the TF-DNA binding intensity for a given TF across any particular DNA sequence. It takes input in the form of an OHE matrix of the input sequence provided. In the initial phase of training, a CNN model was trained with the training set to learn the best parameters and hyper-parameters including different filter numbers. Finally, the model was evaluated on the independent test set, and its performance was reported. To illustrate the better prediction performance of the trained CNN model, the extracted learned convolution layer filters for all 265 TFs were mapped into the MEME database. After this mapping, 197 out of 265 TFs with high motif similarities (P-value ≤ 1.0e-5) between at least one of the learned filters and known motifs in the MEME database were extracted. This mapping result indicated that CNN learned valuable sequence features with biological implications for known motifs. This tool also works on transfer learning strategies to perform trans-species TFBS identifications.

iv) SeqConv is another CNN-based tool for TFBSs identification in plants. It was trained on a 104 TFs ChIP-seq dataset of Zea mays, and implemented in python programming language. The CNN model architecture is adopted from DeFine [83], a sequence-based deep learning model that optimally identifies the TF binding to given DNA sequences. The algorithm takes the input of raw sequences, which are then reverse-complemented to mimic a DNA double strand, as TFs recognize

both the strands of the DNA at a specific position. Later, both sequences are converted into OHE. The OHE stores each input sequence's reverse complementary sequence. As a result, the CNN simultaneously models the forward sequence and its reverse complementary sequence. The forward strand and its reverse-complement input matrices are fed into the convolution layers at the same time. The reverse-complementary sequence's convolution layers share the same set of filters as those used by the forward sequence's convolution layers. For the pooling layer, two pairs of one max pooling and one average pooling were applied. After that, the results of both reverse and forward strand sequences are combined into a vector. Finally, this combined vector is further passed on to the batch normalization layer before being input into the fully connected layer. The output from this layer is passed to the hidden layers, which have 64 nodes, which eventually leads to the final output layer having sigmoid activation.

2) Complex CNNs based TFBS identification software

In order to solve the problem of the vanishing/exploding gradient, complex CNN architectures are designed for deep learning models. These architectures are distinguished by their depth, significant use of convolutional layers, and use of additional features to enhance CNN’s performance. Some examples of complex CNN architectures are: ResNet (Residual Network) and DenseNet (Densely Connected Convolutional Network). In the ResNet network, a method known as skip connections is applied. The skip connection connects layer activations to subsequent levels by bypassing certain layers in between. This results in a residual block. ResNet is created by stacking these residual blocks together. DenseNet is a convolutional neural network with a tightly connected architecture. DenseNet has feature sharing and arbitrary interlayer connections. Below discussed are some tools which incorporate such architectures to find the TFBS:

i) MAResNet, a complex DL method, combined the bottom-up and top-down attention mechanisms with a feed-forward network named ResNet, implemented in python programming language

(**Figure 5(d))**. The bottom-up and top-down structures imitate the feed-forward and feedback attention processes. The multi-scale attention mechanism can extract rich and representative sequence features to capture relevant patterns and motifs. As the number of layers increases, distinct modules' attention perception characteristics adjust. The two different layers, (i) the soft mask branch and (ii) the trunk branch, are used to extract features from input DNA sequence. The soft mask branch includes a rapid feed-forward sweep and a top-down feedback step. The bottom-up (down-sample) level rapidly collects global information from the sequence features, whereas the top-down (up-sample) step integrates global information with the original feature maps. The designed soft mask branch is used as identity mapping by making reference to residual learning, because stacking attention modules may make the gradient vanish. In residual learning, each subsequent layer in a deep neural network is only responsible for fine-tuning the output from a previous layer by just adding a learned "residual" to the input. The gradient vanishing problem occurs during the training process when the gradients of the loss function with respect to the network's parameters become very small, effectively causing the network to stop learning or learn very slowly. The trunk branch is used to extract the features of the CNN, whereas the soft mask branch works as a feature selector to enhance the recognizable features and suppress the noise from the trunk branch. The network architecture of MAResNet works in the following ways: the input DNA sequence is represented by a binary OHE feature matrix of size '*l*'x4, where '*l*' is the length of the sequence and four different bases. The features are input successively to the convolutional layer, batch normalization layer and ReLU layer, successively. Next, three attention modules among the four residual blocks are referred to as 'stage 1', 'stage 2', and 'stage 3', respectively. In this architecture, attention modules generate attention-aware features that differ adaptively as layers move deeper into the network. In 'stage 1' (shallow layer), multi-scale attention is implemented to focus on the data of various scales of receptive fields by using three distinct sizes of convolution kernels, namely three, five, and nine. In the shallow layer, multi-scale attention modules are employed to extract additional sequence information, while the attention-aware features of different

modules vary adaptively as the layers go deeper. In the stages two and three attention modules, they utilized 64 convolution kernels with a kernel size of 13. The size of the feature map at each level is reduced in order to minimize the number of parameters in FC layers. The average pooling layer was added to further reduce the size of the feature map. The pooling layer enhances the robustness of the model by preventing network overfitting. Finally, the dropout and the softmax function were used at the end of the network model. In this tool, hyper-parameter optimization was done using grid search.

ii) TSPTFBS 2.0: The updated version of TSPTFBS has utilized DenseNet for model construction on a large-scale dataset of a total of 389 plant TFs. It additionally trained the models for 389 TFs including 265 TFs from *Arabidopsis thaliana* DAP-seq data, 104 ChIP-seq for *Zea mays* ChIP-seq and 20 for rice TFBS datasets. In order to strongly connect each layer with all the preceding layers, DenseNet uses the ResNet concept (**Figure 5(e))** [84]. The benefits of DenseNet include feature map reuse via dense connections, reducing interdependence between layers by reusing feature maps from different layers. It provides compact and differentiated input features via shortcut connections of varying lengths, and effectively reduces the vanishing gradient problem, which is difficult to optimize in deep networks. The DenseNet model used in TSPTFBS 2.0 has the following architectural components: an average pooling layer (stride = 2), four DenseBlocks, three TransitionLayers, and two convolution layers with the same number of convolution filters (filters = 64, kernel size = 3). The DenseBlock is composed of several layers with deep connections between them, usually convolutional layers. Every layer in a dense block is linked to every layer that came before it, including its own input. In the DenseBlocks, the size of the feature maps (two-dimensional grids of values that represent the presence of learned features within an input data volume) inside stays fixed, so they can be used for concatenation, but their volume changes. However, a transition layer joins two neighboring dense blocks. Mainly, it aims to preserve channel depth while reducing the spatial dimensions of feature maps, usually the width and height. In

Transition Layers, on the other hand, use 1x1 convolution and 2x2 pooling layers to carry out downsampling between dense blocks. Each DenseBlock consists of L_i DenseLayers (L_i = 6, 12, 24, 16), each of which is made up of two convolution layers. All of these DenseLayers are directly connected to one another (with equal feature map sizes).

3) Recurrent neural network (RNN) based TFBS identification software

RNN is a form of artificial neural network intended to analyze data sequences. Unlike classic feed-forward neural networks, which are well-suited for fixed-size input data, RNNs are specialized for sequences of various lengths. This above-described characteristic makes RNN particularly helpful for time-series data, natural language processing, speech recognition, and other applications. RNN is meant to use sequential information from input data by using cyclic connections between building blocks such as the perceptron, long short-term memory (LSTM), or GRU. Some of the tools developed based on this philosophy to detect the regulatory sites are:

i) KEGRU is a deep learning method based on the RNN algorithm implemented in python programming language (**Figure 5(f))**. This tool is mainly used to identify TFBS by using Bidirectional Gated Recurrent Unit (BiGRU) network with *k*-mer embedding. The word embedding representation of the input sequence is used in this model to represent a *k*-mer as a low-dimensional vector. This representation is used to solve a demerit of OHE, in which as the dimensionality of the dataset increases, training becomes slower and more difficult. Therefore, the *k*-mer representation accurately depicts the distributional nature of text data. This *k*-mer dataset is sent to the BiGRU which extracts feature information from DNA sequences of any length, and then identifies TFBS by using feature information. The proposed model follows certain criteria coupled with a *k*-mer embedding representation of the input sequence to develop a network's internal state. This enables it to recognize inter-relationships and achieve optimal performance. In the first criteria, a specific length and stride window of *k*-mers are attained from the DNA sequences. Following that,

word2vec converts the *k*-mer sequences into a *d*-dimensional vector space, where ”*d*” stands for {0...∞}. Finally, BiGRU learns and captures the inter-dependent features from *k*-mer sequences and identifies TF binding sites. The dense layer equipped with a sigmoid activation function produces the result, which is then compared with the actual target labels using a loss function. In this approach, there are two categories of hyper-parameters used: model-related and data-related. In terms of the model, it has 12 parameter settings that are produced by combining several optimizers with a GRU number. This tool is mainly based on two steps: training the model and the statistical analysis of the data. All test results after training the model are categorized by cell type, hyper-parameter setting, and test indicators in data statistical analysis of the data.

4) Hybrid DL-based TFBS identification software

Hybrid neural network model is a combination of various DL architectures. Most of the existing ones have used convolutional neural networks and recurrent neural networks. CNN has shown efficiency in processing image-like data, but RNN is more appropriate for modeling sequential data. In genomics, it's standard practice to combine CNNs with RNNs. CNNs can be used to extract features from DNA or protein sequences, while RNNs identify sequential relationships in the data. Applications such as protein interaction prediction, genomic sequence categorization and DNA-protein interaction identification benefit from this combination. Here, below we discuss some of the hybrid DL-based TFBS identification software:

i) WSCNNLSTM proposed a combinatorial model using multiple-instance learning (MIL) [85] with hybrid deep neural networks (**Figure 5(g))** for identifying *in-vivo* protein-DNA binding and making use of *k*-mer encoding to modify DNA sequences. The architecture is implemented in python programming language. The model aims to learn the weakly supervised information of genomic sequences, as bound DNA sequences may have multiple TFBSs. The proposed framework may be considered in two sections: in the first one, this tool divides sequences into several

overlapping instances using a sliding window, and then uses *k*-mer encoding to transform all instances into image-like inputs of high-order dependencies. In the second one, to capture the forward and backward local dependencies between motif features, they added a bi-directional recurrent layer after the convolutional layer in the weakly supervised framework. The computed scores were generated by a softmax layer used in this model. Two neurons make up this layer, each one computes a probability and is closely related to the previous layer. Finally, using a fusion method, the Noisy-and, the outcomes values for each occurrence were integrated to provide the final results. For hyper-parameter optimization, three main strategies were used: the (i) Glorot uniform initializer, (ii) grid-search algorithm, and (iii) optimized by AdaDelta algorithm.

ii) deepRAM: The purpose of building the tool **deepRAM** was to explore various combinations of deep learning algorithms, such as CNN/RNN and CNN+RNN based models (**Figure 5(g))**, for identifying DNA- and RNA-binding specificity and selecting an optimal model in a fully automatic way to avoid the need for hand-tuning. It is implemented in python programming language. It explains the differences between models acquired through CNN and RNN and, hence, assists the user in choosing an appropriate network design. The architecture of deepRAM has input sequences represented in two ways: either using OHE or *k*-mer word representation computed using word2vec. For OHE, the input sequence is represented by a 4 x '*l*' matrix, where '*l*' is the length of the input sequence. The nucleotide at that point is represented by a single non-zero element in a vector of length four which is connected to each position in the sequence. For the representation of the *k*-mer embedding, the input sequence is divided into overlapping *k*-mers of length '*k*' with a sliding window and stride '*s*'. This is followed by the word2vec technique to map each *k*-mer in the sequence into a *d*-dimensional vector space. In deepRAM, DL architecture constitutes CNN-only models, known for their tendency to detect spatial patterns and thus motifs. RNN-only models are good at capturing sequential dependencies within the sequences. Hybrid CNN-RNN models offer the advantage of capturing both sequence-specific motifs and the context in which these motifs

appear. They applied the applicability of CNNs to biological sequence data by integrating one dimensional convolution. In the second class of architecture, RNN-only models, three types of RNN units were used: long short term memory (LSTM) units, GRU units, and Bi-GRU. The third type, the hybrid CNN-RNN model, considered hybrid convolution and recurrent networks that considered the orientations and spatial relationships between the motifs. In all three types of models, the final module is made up of one or two fully connected layers that integrate information from the entire sequence, followed by a sigmoid layer at the output of neural networks that calculates the probability that the input sequence contains a DNA/RNA-binding site.

iv) PlantBind is an attention-based multi-label neural network method for TFBS identification. It is an integrated model made up of two separate CNN networks and an LSTM-Attention block (**Figure 5(g))** that extracts and learns relevant information. This integrated model supports simultaneous analyses for multiple TFs, which combines data processing, embedding, feature extraction and multi-label output modules implemented in python programming language. The first module uses OHE to transform DNA sequences to be handled by CNN, along with 14 DNA shape features for each input sequence. Data processing yields two heterogeneous data types: i) a sequence matrix, and ii) a shape matrix. Two CNN models are then used to embed the two matrices into the same shape so that they can be combined to form one embedding matrix. In the feature extraction module, the embedded representation is passed to the LSTM layer to extract the underlying features shared by all TFs. Thereafter, the attention mechanism enables the model to focus on the relevant regions of the embedding vector for each specific TF. Finally, the multi-label module contains two fully connected layers to generate multiple binding sites, simultaneously. The framework was trained using a cross-entropy loss enhanced by online hard example mining (OHEM) and focal loss (FL). In summary, given a set of TFBSs, PlantBind learned to associate sequence context with TF types. Once this mapping is learned, the attention mechanism interprets the model and extracts the sequence content which contributes the most towards identification. The integrated model, which

uses a multi-label framework, allows the model to learn the underlying associations between the different TFs.

5) Transformer-based TFBS identification software

Traditionally, natural language processing (NLP) tasks such as machine translation and language modeling depended on RNNs and CNNs to handle text sequences. However, these models have significant limitations, including difficulties with long-term dependencies and a high computational cost. The Transformers based model solves these restrictions by employing self-attention techniques which allow the model to attend to distinct parts of the input sequence, even if they are far apart. Though the Transformers may be considered as a kind of RNN architecture, its remarkable leap in performance due to self-attention with multiple learning heads make it as the most successful deep-learning architecture puts it in a totally different league. Bidirectional Encoder Representations from Transformers (BERT) is a pre-trained language model which employs the Transformer architecture to generate high-quality text representations. It is a method of learning language representations that use a transformer, especially the encoder component.

i) DNABERT, BERT-based tool, applies Transformer, an attention-based architecture that has achieved state-of-the-art performance in most natural language processing tasks (**Figure 5(h))**. DNABERT is a unique pre-trained bidirectional encoder representation and mainly employed to capture global and transferable knowledge of genomic DNA sequences based on up and downstream nucleotide contexts covering long ranges. It has been implemented in python programming language. It provides a pre-training and fine-tuning prototypes which use massive quantities of data. Furthermore, it first creates general-purpose understandings before using task-specific data to address a variety of applications with the least amount of architectural adjustment. The pre-training of this tool's model is computationally expensive. The training procedure for DNABERT is the same as the BERT training process. They tokenized a DNA sequence using the *k*-

mer representation, a technique that has been frequently utilized in DNA sequence analysis, as opposed to thinking of each base as a separate token. The *k*-mer representation integrates more contextual information for each base by concatenating it with the ones that come after it. A *k*-mer is the result of combining them. Each *k*-mer is seen as a unique word having different numerical values (tokens). These *k*-mer tokens are represented as a matrix by embedding each token into a numerical vector. They masked continuous *k*-length stretches of certain *k*-mers during the pre-training phase, in which randomly or strategically selected regions were selected within the DNA sequences using masked language modeling. Masked language modeling is a strategy that randomly replaces a specific percentage of the token in a sentence with a special [MASK] token. These masked regions were marked as "masked" regions or masked tokens that represent where TFBS might be located. The model must then predict the original token given the context of the sentence's other tokens. During pre-training, DNABERT uses this strategy to get an excellent understanding of how various tokens are connected to one another in context. After this, 12 Transformer blocks receive the input after it passes through an embedding layer. Finally, for sentence-level classification, the first output among the final hidden states is utilized. For the token-level classification, the outputs for each individual masked token are used. Also, DNABERT's applicability was tested on other mammalian species after pre-training it on the human genome. On 78 mouse ENCODE ChIP-seq datasets [86], the pre-trained DNABERT models with human genome were fine-tuned and compared with CNN, CNN+LSTM, CNN+GRU algorithms, and randomly initialized DNABERT. Pre-trained DNABERT outperformed all baseline models.

Performance evaluation strategies

In this study, to evaluate the performance of all the considered tools for TFBS identification in plant systems, we compared all 15 of these tools across three different datasets for the comparative performance assessment as described above (Datasets “A”, “B”, and “C”). We evaluated all the existing software for the three datasets in order to objectively evaluate their efficiency. The default

parameters were also considered, as all of these software had already optimized their hyper-parameters during the training and model building processes for the classification task. Here, the term classification means that the goal of the model is to attribute the correct label to each data instance for the TFBS's presence or absence. The assessment of a classifier can be evaluated on the basis of certain analytical reasoning. In order to distinguish between TFBS instances that should have been true instances but were missed (false negative (FN)) and sites that are identified to be binding sites in actuality but were not (false-positive (FP)), we decided to count the number of true binding sites (true-positive (TP)) and those classified as non-TFBS (true negative (TN)). The ratio between the number of correctly classified samples, i.e., TP and TN, and the overall number of samples is called accuracy. The accuracy for the positive class can be seen as the fraction of positive instances that are correctly recovered by the model, also called the true positive rate (TPR) or sensitivity (Sn). The specificity (Sp) metric is used for evaluating a model’s ability to identify the TN from total negatives. Besides these, the precision metric quantifies the accuracy of positive predictions made by a model. Precision is defined as the ratio of true positive predictions to the total positive predictions made by the model. The recall is determined as the ratio of the number of positive samples accurately classified as positive to the total number of positive samples. The F1-score combines the precision and recall of a classifier into a single metric by taking their harmonic mean. The performance of the models were also assessed by computing the area under the receiver operating characteristic (AUROC) curve (AUC). A ROC curve depicts the trade-off between the true positive rate (TPR) and false positive rate (FPR) at various decision thresholds. Finally, six different metrics for performance were considered (Sn, Sp, F1-Score, Acc, Matthew's correlation coefficient (MCC), and AUC). MCC is a reliable statistical measure which produces a high score only if the model obtains good results in all the four confusion matrix categories (TP, TN, FN, and FP). We calculated Acc, Sp, Sn, F1-score, AUC and MCC, as given in the following equations:

$$\mathrm{Acc}=\frac{\mathrm{TN+TP}}{(\mathrm{TN+TP+FN+FP})}$$

$$\text{Specificity}(\text{Sp}) = \frac{\text{TN}}{(\text{TN}+\text{FP})}$$

$$\text{Sensitivity}(\text{Sn}) = \frac{\text{TP}}{(\text{TP}+\text{FN})}$$

$$F1-\text{Score} = 2 \times \left(\frac{\text{Precision} \times \text{Recall}}{\text{Precision}+\text{Recall}} \right)$$

$$\text{AUC} = \int_0^1 \Pr[\text{TP}](v)\,\text{dv}$$

$$\text{MCC} = \frac{\text{TP} \times \text{TN} - \text{FP} \times \text{FN}}{\sqrt{(\text{TP}+\text{FP})(\text{TP}+\text{FN})(\text{TN}+\text{FP})(\text{TN}+\text{FN})}}$$

Where:

TP = True Positives, TN = True Negatives, FP = False Positives, FN = False Negatives, Acc = Accuracy, AUC = Area Under Curve.

Results and Discussion

Datasets collection, processing, and construction

In order to gain insights into how well the ML/DL models capture TFBS across different species, we evaluated them on ChIP-seq and DAP-seq experimental data for plants and animals, while exactly following their reported implementation details and datasets. This first stage of evaluation was done to verify the claims made by the existing software and their veracity. Though this study primarily focused towards plants transcription factors, software for animal TFs were also evaluated for their possible use in plants. Thus, in order to first evaluate their claim on their own dataset, we had also to consider animal datasets. This dataset has been named as Dataset “A”, mentioned in the methods in details and in **Figure 4(a)** and **Supplementary Table S1 (Sheet 1)**.

Most software had common dataset which were developed for human TFBS identification and used 690 ChIP-seq experimental datasets from the ENCODE project. DESSO, MAResNet, and DNABERT used all ChIP-seq experiment data sets, which represent only 137 unique TFs. DeepBind, gkmSVM, LS-GKM, and KEGRU used the data for 506, 467, 322, and 125 ChIP-seq experiments. These 690 ChIP-seq experiments covered over A549, MCF-7, H1-HESC and HUVEC cell lines. They formed the positive dataset instances. deepRAM and WSCNNLSTM used only 83 and 50 ChIP-seq experiments from the ENCODE project, respectively. *k*-mer grammar considered core promoter regions related to TSSs, binding loci from ChIP-seq peaks of two TFs (i.e., Homeobox KNOTTED 1-KN1, bZIP FASCIATED EAR4-FEA4), and open chromatin regions by MNA-seq derived from two tissues for *Zea mays*. While SeqConv used 104 maize TFBS datasets produced by ChIP-seq to determine the TFBSs. PlantBind and TSPTFBS used a large-scale DAP-seq TFBS dataset (265 TFs) for *Arabidopsis thaliana*. The updated version of TSPTFBS 2.0 included 20 rice (*Oryza sativa),* 104 maize and 265 *Arabidopsis* TFs, covering a total of 389 plant TFs data. The negative instances of this dataset were derived exactly the same way as the corresponding implementing tools had mentioned, also detailed in the methods section.

Another dataset, Dataset “B”, was constructed to raise the plant specific models for all the considered software and evaluated on one common dataset, specifically on the plant dataset. In this, we used 265 TFs from *Arabidopsis thaliana* DAP-seq data retrieved from Plant Cistrome Database to form the positive dataset. However, random intergenic sequences for each TF were used to generate negative datasets for the respective TFs while ensuring the removal of overlapping regions. This must be pointed out that most of the compared software neglected this and had many instances where negative dataset sequences were overlapping with the positive instances. For the third Dataset “C”, we considered only five common TFs, namely LHY1, MYB62, MYB81, MYB88, and WRKY25. These were the only TFs common between *A. thaliana* and *Zea mays,* which could help to evaluate cross-species models for any given TF.

Bugs infest most of the existing tools

It is to be noted that the standalone versions of all the available software have been developed in the C, C++, python and R programming languages. However, we encountered several dependency issues and faced many types of programming errors while executing almost all the software. Considering the fact that there may be errors that render these important tools non-usable to biologists, we fixed all such errors and made them work. These types of errors were reported and corrected while the programs were being executed. **Supplementary Material S3** provides a detailed error list faced for every single software taken into this study. The listed errors fall into the categories of “SyntaxError”, “LogicError”, “KeyError”, “IndentationError”, “MemoryError”, “ImportError”, “AttributeError”, and many more. The **Table 2** provides a categorical overview of different types of errors faced during the extensive and exhaustive debugging of these software done in this study.

The most common type of error observed was a “SyntaxError”. This error happened when the python version’s functions, classes and modules did not support another one. The main reason here was that some of the tools’ programming scripts were written in Python2.7. This type of error was reported for KEGRU and WSCNNLSTM. Although, it is to be noted that Python2.7 has reached its end-of-life, and some modules may not be available or maintained anymore. Therefore, to solve this issue, we migrated these Python2.7 dependent tools’ programs into Python3.6 because it includes new modules and enhancements, compatibility with libraries, and access to the updated language features as well. Another type of error was encountered for mostly all the software: “ImportError” and “ModuleNotFoundError”. These types of errors occur when the interpreter cannot locate or import any specific module. The tools grouped into these types of errors are DeepBind, KEGRU, *k*-mer grammar, WSCNNLSTM, SeqConv, DESSO and PlantBind. In that case, we installed the

specific requirements of the modules. In the majority of programs, we were not able to access a file or directories that were not there at the given location, and hence python generated a “FileNotFoundError”. This type of error occurred while executing programs for KEGRU, SeqConv, TSPTFBS, PlanBind, and TSPTFBS 2.0. We handled this situation by providing appropriate file locations or directories. Another class of error belonged to “TypeError” and “KeyError”, noted for common runtime errors. These errors occur when certain operations are performed on incompatible data types and when a desired key is not found in a dictionary. The tools that were classified in these categories were KEGRU, WSCNNLSTM, and SeqConv. To fix these errors, we converted them to the appropriate data types and edited the dictionary keys. Here, it is noted that we also faced the “logicError” while executing code for KEGRU. This type of error occurs when there is a flaw in the logical flow of the code, leading to unexpected or incorrect results.

We were also unable to get the output of all the performance metrics for the evaluation of the software. By calculating performance metrics, we gain a deeper understanding of the models’ performance. Accordingly, we can maintain informed decisions regarding their optimization, selection, and deployment. Thus, we implemented a dedicated function or a bunch of code lines to accurately compute the performance metrics for the tools. This function, or a bunch of code lines, encapsulates the necessary calculations for evaluating the model's performance. The corrected scripts for every considered software have been made available at https://github.com/SCBB-LAB/Comparative-analysis-of-plant-TFBS-software.

Hyper-parameter optimization and impact on performance

The optimum performance of deep neural networks is attained by parameter and hyper-parameters tuning. Parameters are optimized automatically during the learning process. However, hyper-parameters work like the perimeter fixed by the developer, under which learning of parameters has

to be done. The performance of the DL methods is directly influenced by several hyper-parameters such as the number of hidden units, batch size, optimizer selection, learning rate, and layer order. Similarly, selecting a kernel function and weight decay are necessary hyper-parameters for support vector machines. In this study, we analyzed all 15 software (developed for animal and plant species) on their default hyper-parameters on Dataset “A”. We got almost similar performance results when we analyzed on their own dataset. However, on Dataset “B” we observed their performance after hyper-parameter optimization (**Supplementary Material S2 (Table 1))**. For the search of the best hyper-parameters on Dataset “B”, the best model performance for DeepBind (human-based) was optimized at: {“Learning rate”: 0.001, “Batch size”: 128, “Learning steps”: 20000, “Dropout”: 0.50}. However, as it can be seen from the **Figure 6**, there were no significant changes in the model performance for both Dataset “A” and Dataset “B”. There were also not much changes in their optimized hyper-parameters on Dataset “B” as compared with default hyper-parameters. For the gkmSVM (human-based) and LS-GKM (human-based), the results were reproduced when parameter C =1, 2, and 3. However, in each case we observed, the default value (C = 1) was optimal. For KEGRU, the most suitable hyper-parameter settings were: {“Batch size”: 200, “kmer length”: 5, “RNN unit”: 70, “Dropout”: 0.5, “Activation function”: softmax, “Optimizer”: Adam, “Loss”: Binary cross entropy}. By optimizing the hyper-parameters on Dataset “B”, KEGRU (human-based) attained the second position for this dataset. For deepRAM (human-based), automatic calibration, which was based on selecting 40 hyper-parameter settings at random from every possible combination. For each of the 40 calibration sets, the best hyper-parameter settings were: {“Batch size”: 128, “Learning steps”: 5000, “Optimizer”: Adagrad, “Dropout”: 0.15, “Activation function”: ReLU}. As it can be seen from **Figure 6 (a and c),** these automatic hyper-parameter calibration upgraded the average accuracy on deepRAM of plant specific dataset (Dataset “B”) above 90%. For the next tool, DESSO (human-based), the optimized hyper-parameters were: {“dropout rate”: 0.5, “Activation function”: ReLU, “Activation function (output layer)”: Sigmoid, “Optimizer”: mini-batch gradient descent, “Epochs”: 30, “Batch size”: 128}. Thus, there was an

improvement of 9.13% in average accuracy value of DESSO on Dataset “B” with optimized hyper-parameters as compared with Dataset “A”. The best performance of this tool was achieved at: {“Activation function”: ReLU, “Hidden layers units”: 100, 50, “Dropout layer”: 0.4, “Activation function”: Sigmoid, “Optimizer”: SGD, “Batch size”: 64, “Epochs”: 50}. For the TSPTFBS (plant-based), we trained their model on Dataset “B” with optimized hyper-parameters. However, in this case, as shown in **figure 6 (b)**, we attained an improved accuracy value with a difference of 4.86%. For DNABERT, model fine-tuning, the best performance was achieved at: {“kmer length”: 6, “Learning rate”: 2e-4, “Epochs”: 5, “Batch size”: 1024, “Weight decay”: 0.001, “Optimizer”: AdamW, “Adam epsilon”: 1e-6, “beta1”: 0.9, “beta2”: 0.98}. We attained an improved average accuracy from the default hyper-parameter with the lead of 2.38% as compared with Dataset “A”. For MAResNet (human-based), we adjusted the hyper-parameters on Dataset “B” and achieved an average accuracy improvement of 2.6% as compared to Dataset “A” and the default hyper-parameters. The final hyper-parameter settings for MAResNet were: {“Optimizer”: SGD, “Weight decay”: 5e-4, “Momentum”: 0.9, dropout = 0.75, “Learning rate”: 0.004, “Batch size”: 256, “Loss”: Cross-entropy, “Activation function”: Softmax}. Thus from the **Figure 6 (a and c)**, it can be deduced that the models’ performance is highly impacted by the dataset refinement and its hyper-parameters adjustment. The established hyper-parameters in this study for each of these tools should be preferred while applying these tools.

Performance across different validated datasets

A highly comprehensive benchmarking study was performed, where the three different datasets, “A”, “B”, and “C” were used to evaluate the performance of 15 different software**.** These tools covered various machine learning algorithms covering traditional ML approaches to recently developed DL methods for TFBS discovery, viz. gkmSVM (SVM), LS-GKM (SVM), k-mer grammar (Linear regression), Wimtrap (XGBoost), DeepBind (CNN), DESSO (CNN), TSPTFBS

(CNN), TFPTFBS 2.0 (DenseNet), SeqConv (CNN), KEGRU (RNN), WSCNNLSTM (CNN+RNN), deepRAM (CNN, RNN and CNN+RNN), MAResNet (CNN-Attention), PlantBind (CNN+RNN-Attention), and DNABERT (Transformer). Datasets "A" and "B" were considered for building as well as testing the built models, whereas Dataset "C" was considered mainly for testing the capability of these software from cross-species identification of TFBS.

Most of the reported performances were found correct

The analysis of model performance on Dataset "A" demonstrated that *in-silico* deep learning models were able to properly identify TFBS as per their claims. The reported performance here was also calculated as an average of all six performance metrics, providing an overall performance for each software. Among these tool sets, some of them were originally trained on human based ENCODE ChIP-seq datasets, such as gkmSVM, LS-GKM, DeepBind, DESSO, WSCNNLSTM, deepRAM, MAResNet, and DNABERT. *k*-mer grammar, Wimtrap, TSPTFBS, TSPTFBS 2.0, and SeqConv were developed on plant TF dataset for species *A. thaliana*, *Z. mays*, and *Oryza sativa*. Each software was evaluated using a wide range of TFs data. Their performance is shown in **Figure 6(a and b)** and listed in **Supplementary Table S2 (Sheet 1)**. Human ENCODE data consists of 690 ChIP-seq experimental datasets for 161 unique TFs evaluated in 91 diverse cell types. However, for the plant dataset, TSPTFBS used 265 DAP-seq TFs for *A. thaliana*, while a total of 387 used by TSPTFBS 2.0, including 265 DAP-seq TFs for *A. thaliana*, 28 ChIP-seq TFs for *A. thaliana*, and 20 TFs for *Oryza sativa.* SeqConv utilized 104 ChIP-seq TFs, and *k*-mer grammar utilized two MNase-seq [87], two ChIP-seq [88, 89], and core promoter [90–92] sequences retrieved for *Z. mays*. To train and evaluate their respective models, all the available tools utilized a balanced positive and negative dataset for training and testing, except WSCNNLSTM, which worked on 1 to 3 times the number of negative instances for each positive sample.

On Dataset “A”, TSPTFBS 2.0, scored the highest scores for all the performance metrics, with an average accuracy of 96.53% and achieved an average MCC value of 0.9340. TSPTFBS 2.0 employed the DenseNet architecture, which substantially outperformed all the other approaches while displaying a gap of only 0.97% between sensitivity and specificity. The second-best performer was *k*-mer grammar, which attained an average accuracy of 93.73% with an average MCC value of 0.8832. This performance was observed on core promoter regions related to TSSs, binding loci from ChIP-seq peaks of two TFs (i.e., Homeobox KNOTTED1-KN1, bZIP FASCIATED EAR4-FEA4), and open chromatin regions using MNase-seq obtained from two tissues (root and shoot). This tool classified regulatory regions based on sequence features, such as *k*-mers by differentially weighting key words. Additionally, the gap between sensitivity and specificity was 3.31%. The third-best performer was deepRAM, evaluated for 83 human TFs. This model achieved an average accuracy of 90.66% and an average MCC value of 0.8149. But it did not show a good balance between specificity and sensitivity, as the gap of 4.4% was noted. The fourth best was DNABERT, which is based on transformers. DNABERT was trained and tested for 690 human ENCODE ChIP-seq experimental datasets. It achieved an average accuracy of 89.54% with an average MCC value of 0.7950. The observed difference between sensitivity and specificity was 3.58%. DNABERT classified many of its negative instances in the positive class. Next to it was TSPTFBS, which was trained and evaluated for 265 TFs for *A. thaliana*, achieving an average accuracy of 86.32%. LS-GKM, evaluated for 322 for humans TFs, attained an average accuracy of 86.24%. However, TSPTFBS edged out LS-GKM with an average MCC value of 0.7722 compared to LS-GKM’s average MCC value of 0.7262. Two other tools, DESSO and MAResNet, were also evaluated for 690 human ENCODE ChIP-seq experimental datasets. They achieved very similar average accuracy of 85.11% and 85.61%, respectively. The difference could be observed for sensitivity, where DESSO had a higher success rate to identify TFBS compared to MAResNet, which had a higher average specificity value of 87.23%. They exhibited similar ranges of average MCC values of 0.7545 and 0.7533, respectively. We further observed that two more different tools

with distinct architectures, namely KEGRU (RNN) evaluated for 125 human ENCODE ChIP-seq experimental datasets and gkmSVM (SVM) evaluated for 467 human ENCODE ChIP-seq experimental datasets showed a very thin line of difference in their average accuracy, 84.55% and 84.31%, respectively. While both, the software have a sensitivity and specificity difference of 1.77% and 2.48%, respectively.

For Dataset "A", we may say that the performance reported by these 15 tools on their own dataset matched the performance we calculated. For this dataset, TSPTFBS 2.0, the updated version of TSPTFBS, has shown an outstanding performance, and outperformed all compared tools. TSPTFBS 2.0 has a substantially higher average accuracy because it has adopted the powerful architecture of DenseNet. DenseNet architecture is multi-layered, which can extract hierarchical features from TFBS sequences. DenseNet provides a number of strong points, including the ability to reduce the vanishing gradient problem, improve feature propagation, and maximize feature reuse. These results emphasize the potential of their strategy in assisting TFBS identification and its significance in the larger context of regulatory genomics. *k*-mer grammar, developed on plant-based dataset software, performed as the second best one because it captures information from the *k*-mer individual frequencies. deepRAM and DNABERT, both were developed for human TFBS identification, and performed well. DNABERT is a successful language model that accurately learns the semantics, syntax, and hidden grammar found in DNA sequences. From the **Figure 6 (a)**, it is clear that the performance results for all the software were significantly similar, as they have reported in their results.

Animal specific tools rebuilt for plants perform well

In the next stage, we evaluated the performance of all these algorithms for plant specific data, on which they were commonly trained and tested. Both, animal based as well as plant based tools were assessed in an objective manner. We constructed and evaluated the performance of all 15 software

on Dataset “B” having 265 DAP-seq experimental TFBS data for *A. thaliana*. **Figure 6 (c and d)**, **Figure 7(b)** and **Supplementary Table S2 (Sheet 2)** provide the complete performance details of all the software for Dataset “B”. For this dataset, TSPTFBS 2.0 (plant-based) again outperformed all the software and attained an average accuracy of 96.79%. TSPTFBS 2.0 achieved a significant balance between sensitivity and specificity (with a gap of just 0.26%). Its average MCC value of 0.9386 was also the highest among all the software. KEGRU (human-based), an RNN-based algorithm, achieved the second best performance with an average accuracy of 95.66% and an average MCC value of 0.9131, while displaying a very good balance between sensitivity and specificity with a difference of only 0.6%. The next best performing tool was DESSO (human-based), developed on a CNN-based approach, which attained an average accuracy of 94.24%, and an average MCC value of 0.8935. Another software LS-GKM, an SVM based approach, scored an average accuracy of 94.29%, and an average MCC value of 0.8862. Next to this, transformer based DNABERT (human-based) attained 91.92% average accuracy with an average MCC value of 0.8394 while displaying a difference of 2.66% between sensitivity and specificity. In addition, gkmSVM, a SVM based approach for human TFBS identification, and TSPTFBS, a CNN based framework for plant TFBS identification, performed equally well, with average accuracy values of 91.55% and 91.50%, respectively. However, gkmSVM edged out TSPTFBS with a better balance between the sensitivity and specificity, having a gap of only 0.3% as compared to TSPTFBS that had a difference of 0.84%. On the contrary, gkmSVM achieved an average MCC value of 0.8314, whereas TSPTFBS achieved an average MCC value of 0.8542. SeqConv, *k*-mer grammar and Wimtrap, the plant specific tools, clocked average accuracy values of 87.7%, 87.67%, and 82.09%, respectively. They scored average MCC values of 0.7895, 0.7901, and 0.6428, respectively. Among them, *k*-mer grammar displayed a good balance between sensitivity and specificity, with a gap of only 0.14%.

Among the top eight best performing software (scoring >90% of the performance metrics considered above), six of them were originally developed for human ChIP-seq data. While two of them, TSPTFBS and its updated version, TSPTFBS 2.0, were originally constructed for plant TFBS identification. The results obtained on Dataset "B" show how well the animal-based software performed on the plant-specific dataset, with higher average accuracy values and average MCC scores and a minimal gap between sensitivity and specificity. Though, TSPTFBS 2.0 emerged as the best one, very close to its performance has been an animal based software KEGRU. This underlines that the tools originally developed for animals may work well for plants too, if retrained with plant specific data. Among these software, TSPTFBS, TSPTFBS 2.0, KEGRU, and DESSO are based on deep learning algorithms. LS-GKM and gkmSVM are based on the SVM. DNABERT is based on the Transformer BERT algorithm. Due to DenseNet architecture, TSPTFBS 2.0 has secured the highest advantage. However, we had expected many advanced algorithms like the BERT model, adapted by DNABERT (human-based), to perform well as transformers that capture long ranged relationships strongly and contexts. But its performance did not meet the expectations. The reason could be that the pre-training was only conducted on the human reference genome, excluding the sequence conservation and diversity across species. High-variability across the plant species exists, and models developed for animals through BERT may go on different trajectories despite tuning with plant data. Additionally, BERT like models are not realistic as their training requires an enormous volume of data and very costly computational resources with continuous training and tuning, making them of the least practical value despite their enthralling theory.

It has to be noted that between the Datasets “A” and “B”, the performance of most of the tools differed. The reason could be that the tools followed a very different approach to construct their negative datasets from each other. It was observed that the significant role of the negative dataset was eventually neglected by all the existing tools. Every single tool was considering a negative dataset raised from random genomic sequences, where many regions were similar to positive

instances. Very few tools tried to implement a more robust method to construct a better quality negative set which would be more realistic. That is why a huge difference in the performance was noticed, as most of the tools performed very well on Dataset “B”. On this dataset, eight tools achieved almost equal average accuracy at the level of 90% or above as compared to Dataset “A”. On Dataset “A” which mainly reflected the data used by the compared tools while developing their own models, only three tools accomplished this score level.

Most of the tools have low practical value due to poor cross-species performance

Consequently, the search for reliable algorithms for TFBS discovery in plants is very much important. In plants, there is an essential requirement for cross-species functionality. In the actual application of inter-species TFBS identification, there is a huge performance gap. Hardly a few tools have reported their practical applications to identify cross-species TFBS, such as DNABERT (human-based), *k*-mer grammar, TSPTFBS, TSPTFBS 2.0, SeqConv, Wimtrap, and PlantBind. However, the criteria they followed to identify common TFBS between related species were quite different. DNABERT fine-tuned the human pre-trained model genome model on 78 mouse ENCODE ChIP-seq experimental datasets [86]. They reported that these pre-trained models on human datasets significantly performed on the mouse datasets, demonstrating the robustness and application of DNABERT even across different genomes. In the above analysis, we say it fails on plants.

For the tool *k*-mer grammar, they developed models for maize and used them to study the regulatory environment of closely related species. They trained the models to preferentially capture conserved features (those shared throughout lines or related species), which are necessary for the transfer of functional genomic annotations between different maize lines. They examined models that were trained on TF binding loci (*Zea mays* KN1) and tested across the core promoters in two species (sorghum and rice). Using the concept of transfer learning, TSPTFBS demonstrated a practical

application of the CNN models, trained for 265 TFs from *A. thaliana*. They tested it across other species. They gathered ChIP-seq datasets for 10 TFs from three different plant species, including four TFs from *Oryza sativa* (NAC6: GSM2748278, BZIP23: GSM2152800, ERF48: GSE93081, MADS29: GSE42201), four TFs from *Zea mays* (ARF5: GSM1925000, O2: GSE63991, P1: GSE38587, KN1: GSE39161), and two TFs from *Glycine max* (GLYMA.06G314400: GSE101672, GLYMA.13G317000: GSE101672). These TFs were screened on the basis of two requirements: (a) Its protein sequence must be very similar to at least any of the *Arabidopsis* TFs (as determined by BLASTP with an E-value of $1.0e^{-5}$); (b) or its TF motif must be highly similar to at least one *Arabidopsis* TF (as determined by TomTom with a P-value of $1.0e^{-3}$). For TSPTFBS 2.0, the updated version of TSPTFBS, they retrieved ChIP-seq datasets for 15 TFs encompassing six more species from ChIP-Hub [93]. This covered one dataset for each of the following species: *Eucalyptus grandis*, *Hordeum vulgare*, *Physcomitrella patens*, *Arabis alpina*, *Chlamydomonas reinhardtii*, and *Solanum lycopersicum*. They evaluated the performance of all the pre-trained models on these 15 TFs for the six plant species. The fourth one, SeqConv, simply applied their applicability by transferring a *Z. mays* ChIP-seq pre-trained model to *A. thaliana* DAP-seq TF datasets and reported an accuracy over 80%. For PlantBind, in order to test the application of the pre-trained model (*A. thaliana*) on datasets of *Zea mays*, they gathered ChIP-seq datasets (Zm00001d031717, Zm00001d042907, Zm00001d024644, and Zm00001d029875) for four maize TFs. They chose the comparable counterparts from maize based on their strong similarity to at least one *Arabidopsis* TF motif, determined by TomTom with an E-value cutoff of $1.0e^{2}$. On the contrary to these software, Wimtrap looked at how well pre-trained models could be applied to different conditions or species. Only the pre-trained models on flowers of *A. thaliana*, ripening fruits of *S. lycopersicum*, seedlings of *O. sativa*, and seedlings of *Z. mays* were used to predict the binding sites in the *A. thaliana* seedlings, with the highest AUC of 86% on ripening fruits of *S. lycopersicum*.

In our assessment for cross-species performance and applicability of these tools, we listed common TFs among different plant species in the available datasets. Only limited TFs have been covered by ChIP-seq and DAP-seq for plant species/conditions. And this value becomes much lower when one looks for common transcription factors across the species. We succeeded in getting only five common TFs, namely LHY1, MYB62, MYB81, MYB88, and WRKY25 between *A. thaliana* and *Z. mays* **(Supplementary Table S1 Sheet 3)**. This dataset was considered as Dataset "C", which was used to assess the cross-species performance of these tools. We used the pre-trained models for these common TFs, trained on the *A. thaliana* DAP-seq dataset (Dataset "B"), and tested all 15 software on the five common TFs from *Z. mays*. However, here we wish to point out that gkmSVM, PlantBind and Wimtrap were found unfit for the inter-species TF model evaluation for these common TFs. The main reason is the fact that gkmSVM has an inbuilt function for their model evaluation, which does not provide the facility to load data for independent evaluation. For PlantBind, as we can also observe in **Figure 6 (c, d),** the performance on Dataset "B" was below an average accuracy of 50%. Therefore, we did not consider its pre-trained model on Dataset "B" for inter-species TFBS identification. On the other hand, for Wimtrap, the PWM (Position Weight Matrices) for the respective TFs were required to perform the task of sub-sequence scanning in the test file for data input along with the TF peak files. This requirement may not work for many species as species specific motif information is mostly lacking in plants, and in fact, this is one of the major quests of TFBS discovery in plants. A limited amount of information on *A. thaliana* TFs PWMs is available, but almost nothing is available for other plant species, including *Zea mays.* This is a big limitation, as actual TFBS annotations are done on newly assembled genomes, where motif information is rarely present. Therefore, we had to discard these tools for Dataset "C" based cross-species evaluation. As we further tested the rest of the 12 tools' applicability in cross-species identification on an independent test dataset, we observed that common TFs trained models showed a very poor performance on Dataset "C" as illustrated in **Figure 6(e)** and **Supplementary Table S2 (Sheet 3).** The highest performance average accuracy of 68.17% was noted for KEGRU (human-

based). However, it worked well for two TFs, MYB62 and MYB81, with accuracy values of 97.03% and 90.27%, respectively. A gap of 2.35% and 2.92% between sensitivity and specificity for MYB62 and MYB81 was observed, respectively. The MCC scores for both TFs were 0.9410 and 0.8058, respectively. Most of the tools displayed heavy over-fitting **(Figure 6(e))**. Nine out of the total of 12 software merely attained an average accuracy value between 50% and 60%. MAResNet, had evaluated MYB88 with 67.78% accuracy, scoring sensitivity and specificity values of 75.90% and 59.66%, respectively. In addition to it, LS-GKM (for MYB88), deepRAM (for MYB62), TSPTFBS (for MYB88), DESSO (for MYB62), DNABERT (for MYB88), and TSPTFBS 2.0 (for MYB88) scored accuracy values of 66.70%, 64.72%, 64.17%, 60.85%, 62.48%, and 61.83%, respectively. The observed performance was very poor of all the tools to identify TFBS in cross-species manner. Overall, the inter-species benchmarking results reflected that all studied tools were of not much value and applicability to identifying the TFBS across the plant species, something which is shocking enough.

Conclusion

Decoding transcriptional regulation is the foundation for decoding the response and behavior of a plant species to various environmental factors, its development, and its propagation. It has many layers, among which understanding transcription factors and DNA is one of the primes. Fortunately, high-throughput techniques are now generating enough data to learn from, but these resources have been still very limited in plants, resulting in very stunted developments and poor understanding of transcriptional regulations in plants, as well as poor practices to detect transcription factor interaction sites across the plant genomes. Though for humans, several software have been developed to detect TFBS, the same is not true for plants. This has become more grave with the fact that approaches used to detect human TFBS have been most grossly copied in plants, with total disregard of the fact that unlike humans. Unlike human, where the focus has been on a single species, in plants, there are numerous species, and most of them having much complex and variable

genomes. Therefore, this requires plant specific biological insights while developing TFBS discovery tools for plants. Some tools, like TSPTFBS 2.0, have performed well due to innovative deep learning architectures like DenseNet, they too leave enough scope for improvement as they have not considered the contextual information, shape and contextual information. Interestingly, those developed for human when trained on plants data also performed almost equally well as TSPTFBS 2.0 due to better consideration of biologically relevant information about shape and flanking region contexts. They could have surpassed TSPTFBS 2.0 because they considered a better learning algorithm. Of TSPTFBS 2.0 could have scored further higher if they had incorporated more informative representations. Thus, both may be used together. But above all, none of the existing software have been able to even attempt properly the most important factor of inter-species variability across the plant genomes and TFBS. This limitation alone renders most of them unusable for practical purposes of their cross-species applications and to use them to annotate newly sequenced genomes. The way ahead is to understand the power of novel deep-learning algorithms as well as more informative representations of learning data, where sequence variability could also be learned. With the kind of variability across the plant genomes and the number of plant species in the queue, even high-throughput sequencing experiments like ChIP-seq and variants do not look pragmatic. This all makes for efficient computational algorithms, which could accurately detect plant TFBS even with minimal data. The findings from this study may serve as an alert as well as a guideline for cautious use of the existing software for plant regulatory element discovery. Also, the corrected scripts may help the biologists to run these tools with lesser hassles until some tools emerge in the future, learning from the matters raised here.

Declarations

Availability of data

All the secondary data used in this study were publicly available and their due references and sources have been provided in **Supplementary information**. All data and information generated/used, methodology related details etc. have also been made available in the Supplementary data files provided. The all corrected running scripts for each tool have also been made available at Github at https://github.com/SCBB-LAB/Comparative-analysis-of-plant-TFBS-software.

Authors' contributions

Jyoti carried out the most of the study. Jyoti, and Ritu carried out the benchmarking study. SG assisted in the study with analysis and evaluation. RS conceptualized, designed, analyzed, and supervised the entire study. Jyoti, Ritu, SG, and RS wrote the MS.

Acknowledgments

The study was carried out under the aegis of The Himalayan Centre for High-throughput Computational Biology (HiCHiCoB), a BIC supported by the Dept. of Biotechnology, Govt. Of India. Jyoti is thankful to UGC for her fellowship. Ritu is thankful to CSIR for her fellowship. SG is thankful to DBT, India for financial support as project associateship. Jyoti, SG, and Ritu are thankful to Academy of Scientific and Innovative Research (AcSIR) for their Ph.D. enrollment. All authors are thankful to the Director, CSIR-IHBT, for his kind support for this study.

Funding

The work was carried out under the aegis of The Himalayan Centre for High-throughput Computational Biology (HiCHiCoB), a BIC supported by DBT, Govt. of India

[BT/PR40122/BTIS/137/30/2021] and its National Network Project, S2S [Sanction order number: 102/IFD/SAN/397/2022-23 dated 26.08.2022].

Declaration of competing interest

The authors declare that they have no competing interests.

Key points

1. We provide a comprehensive survey and assessment of tools for the identification of plant TFBS using three different datasets. Most of the software showed bugs rendering them hard to use by biologists. We have fixed them and released in the public domain.
2. Most of the existing software have made correct claims about their performance. However, the datasets raised by them, especially the negative instances counterparts are noisy.
3. Compared to human, plants have much lesser data and information on transcription factors where most of the information has been limited around species like *A. thaliana*. Following human-based approach in plants has led to wrong directions as unlike human, plants have huge variability. Also, there has been very limited efforts made to incorporate plant specific information for plant based tools which are not just few but also way behind in terms of considerations of biological properties for information like context, flanking information, composition and DNA shape.
4. A comprehensive analysis and evaluation of 15 different and recent tools was performed which included even those developed for human. Human-based software when trained on plant data delivered similar level of the performance as was noticed for the best performing software for plants, TSPTFBS 2.0, underlining that they can be trained and used for plants also.
5. All of the studied tools performed very poorly when tested for their real world applicability by running them in cross-species manner. This highlights that plant science urgently needs

to develop plant specific models of transcriptional regulation with larger biological information where variability must be taken into the account.

References

1. Stewart AJ, Hannenhalli S, Plotkin JB. Why transcription factor binding sites are ten nucleotides long. Genetics 2012; 192:973–985
2. Mitchell PJ, Tjian R. Transcriptional regulation in mammalian cells by sequence-specific DNA binding proteins. Science 1989; 245:371–378
3. Jones DM, Vandepoele K. Identification and evolution of gene regulatory networks: insights from comparative studies in plants. Curr Opin Plant Biol 2020; 54:42–48
4. Weirauch MT, Yang A, Albu M, et al. Determination and inference of eukaryotic transcription factor sequence specificity. Cell 2014; 158:1431–1443
5. Huang H, Tudor M, Su T, et al. DNA binding properties of two Arabidopsis MADS domain proteins: binding consensus and dimer formation. Plant Cell 1996; 8:81–94
6. Purugganan MD, Wessler SR. Molecular Evolution of the Plant R Regulatory Gene Family. Genetics 1994; 138:849–854
7. Goff SA, Cone KC, Chandler VL. Functional analysis of the transcriptional activator encoded by the maize B gene: evidence for a direct functional interaction between two classes of regulatory proteins. Genes Dev 1992; 6:864–875
8. Washburn KB, Davis EA, Ackerman S. Coactivators and TAFs of transcription activation in wheat. Plant Mol Biol 1997; 35:1037–1043
9. Guilfoyle TJ, Ulmasov T, Hagen G. The ARF family of transcription factors and their role in plant hormone-responsive transcription. Cell Mol Life Sci 1998; 54:619–627
10. Martin C, Paz-Ares J. MYB transcription factors in plants. Trends in Genetics 1997; 13:67–73

11. Jin J, Tian F, Yang D-C, et al. PlantTFDB 4.0: toward a central hub for transcription factors and regulatory interactions in plants. Nucleic Acids Research 2017; 45:D1040–D1045
12. Rohs R, West SM, Liu P, et al. Nuance in the double-helix and its role in protein-DNA recognition. Curr Opin Struct Biol 2009; 19:171–177
13. Sarai A, Kono H. Protein-DNA recognition patterns and predictions. Annu Rev Biophys Biomol Struct 2005; 34:379–398
14. Diao Y, Fang R, Li B, et al. A tiling1deletion based genetic screen for cis-regulatory element identification in mammalian cells. Nat Methods 2017; 14:629–635
15. Thukral SK, Morrison ML, Young ET. Alanine scanning site-directed mutagenesis of the zinc fingers of transcription factor ADR1: residues that contact DNA and that transactivate. Proc Natl Acad Sci U S A 1991; 88:9188–9192
16. Franco-Zorrilla JM, López-Vidriero I, Carrasco JL, et al. DNA-binding specificities of plant transcription factors and their potential to define target genes. Proc Natl Acad Sci U S A 2014; 111:2367–2372
17. Amoutzias GD, Robertson DL, Van de Peer Y, et al. Choose your partners: dimerization in eukaryotic transcription factors. Trends Biochem Sci 2008; 33:220–229
18. Kaplan T, Li X-Y, Sabo PJ, et al. Quantitative models of the mechanisms that control genome-wide patterns of transcription factor binding during early Drosophila development. PLoS Genet 2011; 7:e1001290
19. Miller JA, Widom J. Collaborative competition mechanism for gene activation in vivo. Mol Cell Biol 2003; 23:1623–1632
20. Soufi A, Garcia MF, Jaroszewicz A, et al. Pioneer transcription factors target partial DNA motifs on nucleosomes to initiate reprogramming. Cell 2015; 161:555–568
21. Lazarovici A, Zhou T, Shafer A, et al. Probing DNA shape and methylation state on a genomic scale with DNase I. Proc Natl Acad Sci U S A 2013; 110:6376–6381

22. Stella S, Cascio D, Johnson RC. The shape of the DNA minor groove directs binding by the DNA-bending protein Fis. Genes Dev 2010; 24:814–826
23. Rohs R, West SM, Sosinsky A, et al. The role of DNA shape in protein-DNA recognition. Nature 2009; 461:1248–1253
24. Hu W, Huang C, Deng X, et al. TaASR1, a transcription factor gene in wheat, confers drought stress tolerance in transgenic tobacco. Plant, Cell & Environment 2013; 36:1449–1464
25. Galas DJ, Schmitz A. DNAse footprinting: a simple method for the detection of protein-DNA binding specificity. Nucleic Acids Res 1978; 5:3157–3170
26. Stoltenburg R, Reinemann C, Strehlitz B. SELEX—A (r)evolutionary method to generate high-affinity nucleic acid ligands. Biomolecular Engineering 2007; 24:381–403
27. Mukherjee S, Berger MF, Jona G, et al. Rapid analysis of the DNA-binding specificities of transcription factors with DNA microarrays. Nat Genet 2004; 36:1331–1339
28. Lee TI, Johnstone SE, Young RA. Chromatin immunoprecipitation and microarray-based analysis of protein location. Nat Protoc 2006; 1:729–748
29. Johnson DS, Mortazavi A, Myers RM, et al. Genome-wide mapping of in vivo protein-DNA interactions. Science 2007; 316:1497–1502
30. Sharma V, Majumdar S. Comparative analysis of ChIP-exo peak-callers: impact of data quality, read duplication and binding subtypes. BMC Bioinformatics 2020; 21:65
31. He Q, Johnston J, Zeitlinger J. ChIP-nexus enables improved detection of in vivo transcription factor binding footprints. Nat Biotechnol 2015; 33:395–401
32. Schmidl C, Rendeiro AF, Sheffield NC, et al. ChIPmentation: fast, robust, low-input ChIP-seq for histones and transcription factors. Nat Methods 2015; 12:963–965
33. Song L, Crawford GE. DNase-seq: a high-resolution technique for mapping active gene regulatory elements across the genome from mammalian cells. Cold Spring Harb Protoc 2010; 2010:pdb.prot5384

34. Buenrostro JD, Giresi PG, Zaba LC, et al. Transposition of native chromatin for fast and sensitive epigenomic profiling of open chromatin, DNA-binding proteins and nucleosome position. Nat Methods 2013; 10:1213–1218
35. Bartlett A, O'Malley RC, Huang S-SC, et al. Mapping genome-wide transcription-factor binding sites using DAP-seq. Nat Protoc 2017; 12:1659–1672
36. Rhee HS, Pugh BF. Comprehensive genome-wide protein-DNA interactions detected at single-nucleotide resolution. Cell 2011; 147:1408–1419
37. Chereji RV, Bryson TD, Henikoff S. Quantitative MNase-seq accurately maps nucleosome occupancy levels. Genome Biology 2019; 20:198
38. Korn LJ, Queen CL, Wegman MN. Computer analysis of nucleic acid regulatory sequences. Proc Natl Acad Sci U S A 1977; 74:4401–4405
39. Bailey TL, Williams N, Misleh C, et al. MEME: discovering and analyzing DNA and protein sequence motifs. Nucleic Acids Research 2006; 34:W369–W373
40. Bailey TL. DREME: motif discovery in transcription factor ChIP-seq data. Bioinformatics 2011; 27:1653
41. Quang D, Xie X. EXTREME: an online EM algorithm for motif discovery. Bioinformatics 2014; 30:1667–1673
42. Liu X, Brutlag DL, Liu JS. BIOPROSPECTOR: DISCOVERING CONSERVED DNA MOTIFS IN UPSTREAM REGULATORY REGIONS OF CO-EXPRESSED GENES. Biocomputing 2001 2000; 127–138
43. Bailey TL, Elkan C. The value of prior knowledge in discovering motifs with MEME. Proc Int Conf Intell Syst Mol Biol 1995; 3:21–29
44. Thompson W, Rouchka EC, Lawrence CE. Gibbs Recursive Sampler: finding transcription factor binding sites. Nucleic Acids Res 2003; 31:3580–3585
45. Sharma NK, Gupta S, Kumar A, et al. RBPSpot: Learning on appropriate contextual information for RBP binding sites discovery. IScience 2021; 24:103381

46. Zhang Y, Bao W, Cao Y, et al. A survey on protein-DNA-binding sites in computational biology. Brief Funct Genomics 2022; 21:357–375
47. Alipanahi B, Delong A, Weirauch MT, et al. Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning. Nat Biotechnol 2015; 33:831–838
48. Shen Z, Bao W, Huang D-S. Recurrent Neural Network for Predicting Transcription Factor Binding Sites. Sci Rep 2018; 8:15270
49. Mejía-Guerra MK, Buckler ES. A *k-mer* grammar analysis to uncover maize regulatory architecture. BMC Plant Biol 2019; 19:103
50. Rivière Q, Corso M, Ciortan M, et al. Exploiting Genomic Features to Improve the Prediction of Transcription Factor-Binding Sites in Plants. Plant Cell Physiol 2022; 63:1457–1473
51. Liu L, Zhang G, He S, et al. TSPTFBS: a Docker image for trans-species prediction of transcription factor binding sites in plants. Bioinformatics 2021; 37:260–262
52. Shen W, Pan J, Wang G, et al. Deep learning-based prediction of TFBSs in plants. Trends Plant Sci 2021; 26:1301–1302
53. Yan W, Li Z, Pian C, et al. PlantBind: an attention-based multi-label neural network for predicting plant transcription factor binding sites. Briefings in Bioinformatics 2022; 23:bbac425
54. Lehti-Shiu MD, Panchy N, Wang P, et al. Diversity, expansion, and evolutionary novelty of plant DNA-binding transcription factor families. Biochim Biophys Acta Gene Regul Mech 2017; 1860:3–20
55. Tu X, Mejía-Guerra MK, Valdes Franco JA, et al. Reconstructing the maize leaf regulatory network using ChIP-seq data of 104 transcription factors. Nat Commun 2020; 11:5089
56. ENCODE Project Consortium. An integrated encyclopedia of DNA elements in the human genome. Nature 2012; 489:57–74

57. O'Malley RC, Huang S-SC, Song L, et al. Cistrome and Epicistrome Features Shape the Regulatory DNA Landscape. Cell 2016; 165:1280–1292
58. Heyndrickx KS, Van de Velde J, Wang C, et al. A functional and evolutionary perspective on transcription factor binding in Arabidopsis thaliana. Plant Cell 2014; 26:3894–3910
59. Celli F, Malapela T, Wegner K, et al. AGRIS: providing access to agricultural research data exploiting open data on the web. F1000Res 2015; 4:110
60. Chang W-C, Lee T-Y, Huang H-D, et al. PlantPAN: Plant promoter analysis navigator, for identifying combinatorial cis-regulatory elements with distance constraint in plant gene groups. BMC Genomics 2008; 9:561
61. Kulakovskiy IV, Vorontsov IE, Yevshin IS, et al. HOCOMOCO: towards a complete collection of transcription factor binding models for human and mouse via large-scale ChIP-Seq analysis. Nucleic Acids Res 2018; 46:D252–D259
62. Sandelin A, Alkema W, Engström P, et al. JASPAR: an open-access database for eukaryotic transcription factor binding profiles. Nucleic Acids Res 2004; 32:D91-94
63. Cheng H, Liu L, Zhou Y, et al. TSPTFBS 2.0: trans-species prediction of transcription factor binding sites and identification of their core motifs in plants. Frontiers in Plant Science 2023; 14
64. Tu X, Mejía-Guerra MK, Valdes Franco JA, et al. Reconstructing the maize leaf regulatory network using ChIP-seq data of 104 transcription factors. Nat Commun 2020; 11:5089
65. Ghandi M, Mohammad-Noori M, Ghareghani N, et al. gkmSVM: an R package for gapped-kmer SVM. Bioinformatics 2016; 32:2205–2207.
66. Trabelsi A, Chaabane M, Ben-Hur A. Comprehensive evaluation of deep learning architectures for prediction of DNA/RNA sequence binding specificities. Bioinformatics 2019; 35:i269–i277
67. Han K, Shen L-C, Zhu Y-H, et al. MAResNet: predicting transcription factor binding sites by combining multi-scale bottom-up and top-down attention and residual network. Brief Bioinform 2022; 23:bbab445s

68. Lee D. LS-GKM: a new gkm-SVM for large-scale datasets. Bioinformatics 2016; 32:2196–2198
69. Zhang Q, Shen Z, Huang D-S. Modeling in-vivo protein-DNA binding by combining multiple-instance learning with a hybrid deep neural network. Sci Rep 2019; 9:8484
70. Yang J, Ma A, Hoppe AD, et al. Prediction of regulatory motifs from human Chip-sequencing data using a deep learning framework. Nucleic Acids Res 2019; 47:7809–7824
71. Ji Y, Zhou Z, Liu H, et al. DNABERT: pre-trained Bidirectional Encoder Representations from Transformers model for DNA-language in genome. Bioinformatics 2021; 37:2112–2120
72. Ritu, Gupta S, Sharma NK, et al. DeepPlnc: Bi-modal deep learning for highly accurate plant lncRNA discovery. Genomics 2022; 114:110443
73. Diaz GI, Fokoue-Nkoutche A, Nannicini G, et al. An effective algorithm for hyperparameter optimization of neural networks. IBM J. Res. & Dev. 2017; 61:9:1-9:11
74. Bergstra J, Bardenet R, Bengio Y, et al. Algorithms for Hyper-Parameter Optimization. Advances in Neural Information Processing Systems 2011; 24:
75. Bergstra J, Bengio Y. Random search for hyper-parameter optimization. J. Mach. Learn. Res. 2012; 13:281–305
76. Klein A, Falkner S, Bartels S, et al. Fast Bayesian Optimization of Machine Learning Hyperparameters on Large Datasets. Proceedings of the 20th International Conference on Artificial Intelligence and Statistics 2017; 528–536
77. Chang C-C, Lin C-J. LIBSVM: A library for support vector machines. ACM Trans. Intell. Syst. Technol. 2011; 2:27:1-27:27
78. Weikum G. Foundations of statistical natural language processing. SIGMOD Rec. 2002; 31:37–38
79. Gupta A, Rush AM. Dilated Convolutions for Modeling Long-Distance Genomic Dependencies. 2017;

80. Sutskever I, Vinyals O, Le QV. Sequence to Sequence Learning with Neural Networks. Advances in Neural Information Processing Systems 2014; 27:

81. Sun Y, Wang X, Tang X. Deep Learning Face Representation from Predicting 10,000 Classes. 2014; 1891–1898

82. Chiu T-P, Comoglio F, Zhou T, et al. DNAshapeR: an R/Bioconductor package for DNA shape prediction and feature encoding. Bioinformatics 2016; 32:1211–1213

83. Wang M, Tai C, E W, et al. DeFine: deep convolutional neural networks accurately quantify intensities of transcription factor-DNA binding and facilitate evaluation of functional non-coding variants. Nucleic Acids Res 2018; 46:e69

84. Huang G, Liu Z, Van Der Maaten L, et al. Densely Connected Convolutional Networks. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2017; 2261–2269

85. Gao Z, Ruan J. Computational modeling of in vivo and in vitro protein-DNA interactions by multiple instance learning. Bioinformatics 2017; 33:2097–2105

86. Mouse ENCODE Consortium, Stamatoyannopoulos JA, Snyder M, et al. An encyclopedia of mouse DNA elements (Mouse ENCODE). Genome Biol 2012; 13:418

87. Rodgers-Melnick E, Vera DL, Bass HW, et al. Open chromatin reveals the functional maize genome. Proc Natl Acad Sci U S A 2016; 113:E3177-3184

88. Pautler M, Eveland AL, LaRue T, et al. FASCIATED EAR4 Encodes a bZIP Transcription Factor That Regulates Shoot Meristem Size in Maize. The Plant Cell 2015; 27:104–120

89. Bolduc N, Yilmaz A, Mejia-Guerra MK, et al. Unraveling the KNOTTED1 regulatory network in maize meristems. Genes Dev 2012; 26:1685–1690

90. Alexandrov NN, Brover VV, Freidin S, et al. Insights into corn genes derived from large-scale cDNA sequencing. Plant Mol Biol 2009; 69:179–194

91. Soderlund C, Descour A, Kudrna D, et al. Sequencing, Mapping, and Analysis of 27,455 Maize Full-Length cDNAs. PLoS Genet 2009; 5:e1000740

92. Mejía-Guerra MK, Li W, Galeano NF, et al. Core Promoter Plasticity Between Maize Tissues and Genotypes Contrasts with Predominance of Sharp Transcription Initiation Sites. Plant Cell 2015; 27:3309–3320

93. Fu L-Y, Zhu T, Zhou X, et al. ChIP-Hub provides an integrative platform for exploring plant regulome. Nat Commun 2022; 13:3413

Tables

Table 1: List of existing softwares for transcription factor binding site prediction used for comparative benchamrking.

S.No.	Software	Algorithm	Year	Standalone (S)	Species	Datasource
1	DeepBind	CNN	2015 [47]	S	Human	ENCODE project [56]
2	gkmSVM	SVM	2016 [65]	S		
3	LS-GKM	SVM	2016 [68]	S		
4	KEGRU	RNN	2018 [48]	S		
5	*k*-mer grammar	Logistic regression	2019 [49]	S	Plant	Two Mnase-seq TFs [87], two ChIP-seq [88, 89], core promoter [90-92]
6	WSCNNLSTM	CNN+RNN	2019 [69]	S	Human	ENCODE project
7	deepRAM	CNN/RNN	2019 [66]	S		
8	DESSO	CNN	2019 [70]	S		
9	TSPTFBS	CNN	2021 [51]	S	Plant	Plant Cistrome Database [57]
10	DNABERT	Transformer	2021 [71]	S	Human	ENCODE project
11	SeqConv	CNN	2021 [52]	S	Plant	GSE137972 [64]
12	MAResNet	CNN+RNN	2021 [67]	S	Human	ENCODE project

13	Wimtrap	XGBoost	2022 [50]	S	Plant	*Arabidopsis* RegNet [58]
14	PlantBind	Attention-based	2022 [53]	S	Plant	Plant Cistrome Database
15	TSPTFBS 2.0	DenseNet	2023 [63]	S/W	Plant	Plant Cistrome Database, GSE137972 and ChIP-hub [93]

Table 2: List of major errors encountered in benchmarking tools during program execution

S.No.	**Type of Errors**	**Benchmarking tools**	**Major Errors**
1	**SyntaxError**	KEGRU	Missing parentheses in call to 'print'. Did you mean print(...)?
		WSCNNLSTM	• Missing parentheses in call to 'print'. Did you mean print(int 'data shape: ',data.shape)?
		KEGRU	• cannot import name 'pad_sequences' from 'keras.preprocessing.sequence' (/home/csir/.local/lib/python3.8/site-packages/keras/preprocessing/ sequence.py) • No module named 'keras_preprocessing' • No module named 'keras.layers.embeddings'
		k-mer grammar	• cannot import name 'joblib' from 'sklearn.externals' (/home/csir/miniconda3/lib/python3.10/site-packages/sklearn/externals/__init__.py) • No module named 'sqlalchemy'
		WSCNNLSTM	• No module named 'ANDNoisy' • No module named 'keras.engine.topology'
		SeqConv	• cannot import name 'batch normalization' from 'keras.layers.normalization' (/home/scbb/.local/lib/python3.8/site-packages/keras/layers/normalization/ __init__.py) • No module named 'keras.layers.advanced_activations'
	ModuleNotFoundError	DESSO	• No module named 'constants'
		PlantBind	• No module named 'google.protobuf'
3	**TypeError**	KEGRU	• Word2Vec.__init__() got an unexpected keyword argument 'size',

			• int() argument must be a string, a bytes-like object or a real number, not 'NoneType' • Model.fit() got an unexpected keyword argument 'nb_epoch' • can only concatenate str (not "NoneType") to str
		SeqConv	• 'float' object cannot be interpreted as an integer • 'dict_keys' object does not support indexing
		PlantBind	• '>=' not supported between instances of 'str' and 'float'
4	**KeyError**	WSCNNLSTM	• 'acc', ‘val_acc’
5	**logicError**	KEGRU	• *k-mer* in lower case
6	**MemoryError**	WSCNNLSTM	• tensorflow.python.framework.errors_impl.InternalError: Failed copying input tensor from /job:localhost/replica:0/task:0/device:CPU:0 to /job:localhost/replica:0/task:0/device:GPU:0 in order to run _EagerConst: Dst tensor is not initialized.
7	**FileNotFoundError**	KEGRU	• No such file or directory: '/home/szhen/szhen/DBLSTM/test2/init_pos_5gram_2stride'
		SeqConv	• can't open file 'prepare.py': [Errno 2] No such file or directory • No such file or directory: 'fa/zm.fa' • No such file or directory: 'train/zm_tf_train.txt' • can't open file 'generator.py': [Errno 2] No such file or directory • No such file or directory: 'maize4.chrmsize' • can't open file 'SeqConv.py': [Errno 2] No such file or directory
		TSPTFBS	• No such file or directory: 'Example/ABF2.fasta' • No such file or directory: 'Example/neg.fasta'
		PlantBind	• can't open file 'PlantBind/main.py': [Errno 2] No such file or directory
		TSPTFBS 2.0	• No such file or directory: '/home/hlcheng/corn/sample/neg_sample.npy'
8	**RuntimeError**	DNABERT	• CUDA error: no kernel image is available

			for execution on the device
		PlantBind	• Default process group has not been initialized, please make sure to call init_process_group.
9	**OSError**	DNABERT	• 'PATH_TO_DNABERT_REPO/src/transformers/dnabert-config/bert-config-6/config.json' was not found in model name list.
10	**Error in install.packages**	gkmSVM	• Error in install.packages("ROCR", "kernlab", "seqinr")
11	**AttributeError**	*k*-mer_grammar	• 'TfidfVectorizer' object has no attribute 'get_feature_names'. Did you mean: 'get_feature_names_out'?
		WSCNNLSTM	• del_num = range(len(history_all): 'range' object has no attribute 'pop'

Figure legends

Figure 1: A timeline of the major milestones in the area of transcription factors and their binding sites research. As can be seen here, PWM based approches to detect TFBS have remained prime. High-throughput technique at the later stage revolutionized the way and volume of data generation.

Figure 2: Overall layout of the present study**.** The image illustrates the outline of the entire evaluation flowchart to assess various models to identify TFBS. **a)** Illustrates data collection and processing to construct training and independent test datasets. **b)** Shows the various encoding schemes used for input datasets used by the evaluated tools, **c)** Description of the tools and their respective algorithms, **d)** and **e)** The evaluation criteria and the performance analyses parts.

Figure 3: The family wise distribution of 265 transcription factors data for the Dataset “A” using **(a)** line plot and **(b)** pie chart. Some of the transcription factors exhibit clear bias. Such bias is also due to more experiments taken for a particular family, and does not reflect the actual abundance in cell system.

Figure 4: Essential steps used to construct three different datasets. The major steps followed in the construction of **a)** Dataset "A", **b)** Dataset "B", and **c)** Dataset "C". The Dataset "A" was created for human and plant transcription factors (TFs). This dataset includes sequences from *Homo sapiens*, *Arabidopsis thaliana*, and *Zea mays*. The positive and negative instances within this dataset were generated following the same procedures as outlined in the respective implementing tools. The Dataset "B" was specifically designed for 265 *A. thaliana* TFs. In this case, random intergenic sequences were employed to produce negative datasets for each TF while ensuring the removal of any overlapping regions. As for the third dataset, Dataset "C", it contained only five common TFs: LHY1, MYB62, MYB81, MYB88, and WRKY25. This dataset serves as an independent test dataset to evaluate the software performance inter-species.

Figure 5: A brief overview of the machine and deep learning architectures that have been considered in this work: **a)** Support vector machine (SVM), **b)** Logistic regression, **c)** CNN algorithm, **d, e)** Complex CNN algorithm (ResNet and DenseNet), **f)** RNN algorithm, **g)** hybrid CNN-RNN models, and **h)** Transformer-based models. The input sequence is encoded as one-hot encoding (CNN and CNN-RNN hybrid), *k*-mer spectrum (SVM, RNN), and as token embeddings (transformer based software).

Figure 6: Comparative evaluation with different combinations of test datasets. **(a)** Dataset "A", **(b)** Dataset "B" used to train and test, and for comparative evaluation. For every such combinations, the performance metrics (sensitivity, specificity, F1-score, MCC and accuracy) are given in the form of heatmap. The first two plots **(**plots **(a, b)** and **(c, d))** clearly indicate that TSPTFBS 2.0 consistently outperforms all the existing tools for all the performance metrics based on Dataset "A" and Dataset "B". For Dataset "C", as shown in **(e, f),** is an independent test dataset used to compare the performance of all the listed tools for inter-species TFBS identification. It clearly shows how poorly

all the existing tools perform when applied in cross-species manner. None of the existing tools were found suitable for practical application for finding TFBS across different species.

Supplementary information

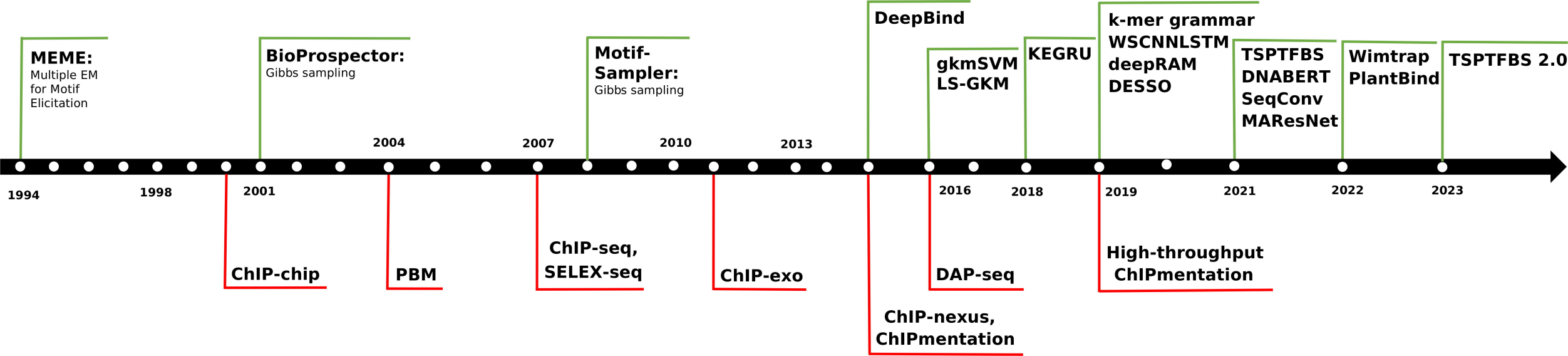
Supplementary Table S1: Detailed dataset description on Dataset “A”, Dataset “B”, and Dataset “C”.

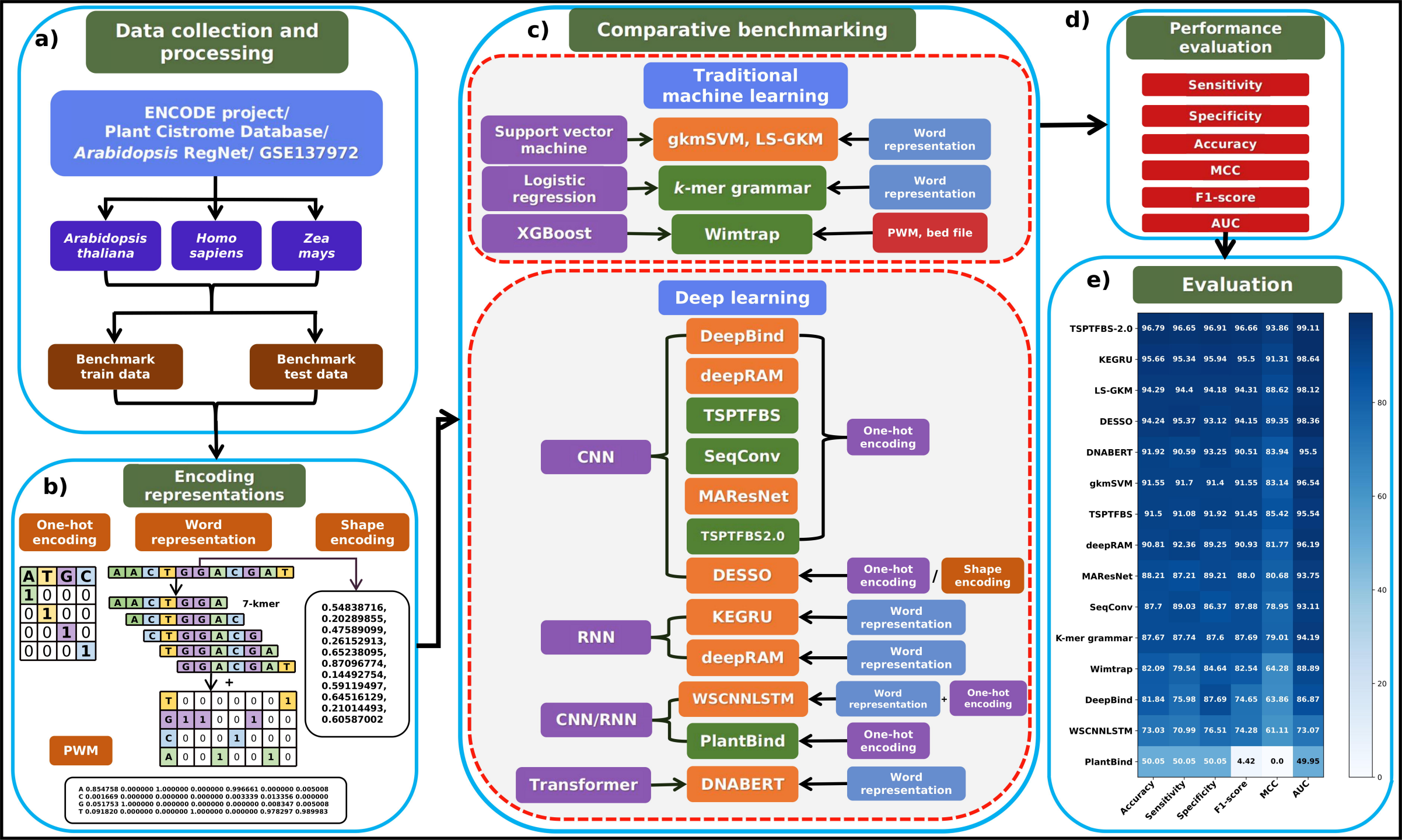
Supplementary Table S2: Evaluation details on Dataset “A”, Dataset “B”, and Dataset “C”, and runtime of all tools on Dataset “A”.

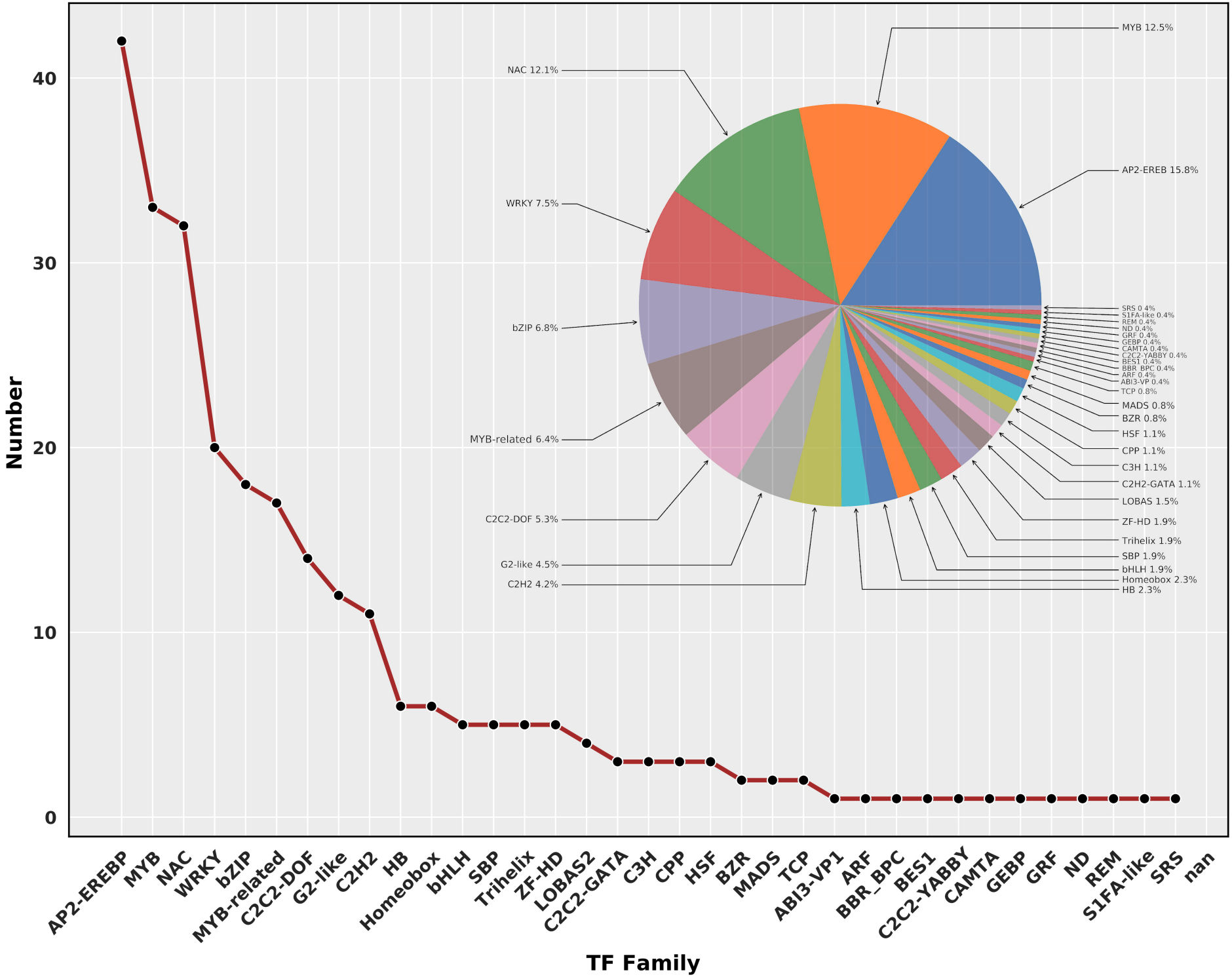
Supplementary Material S1: List of the existing experimental techniques, databases and some published tools for TFBS identification in human and plants.

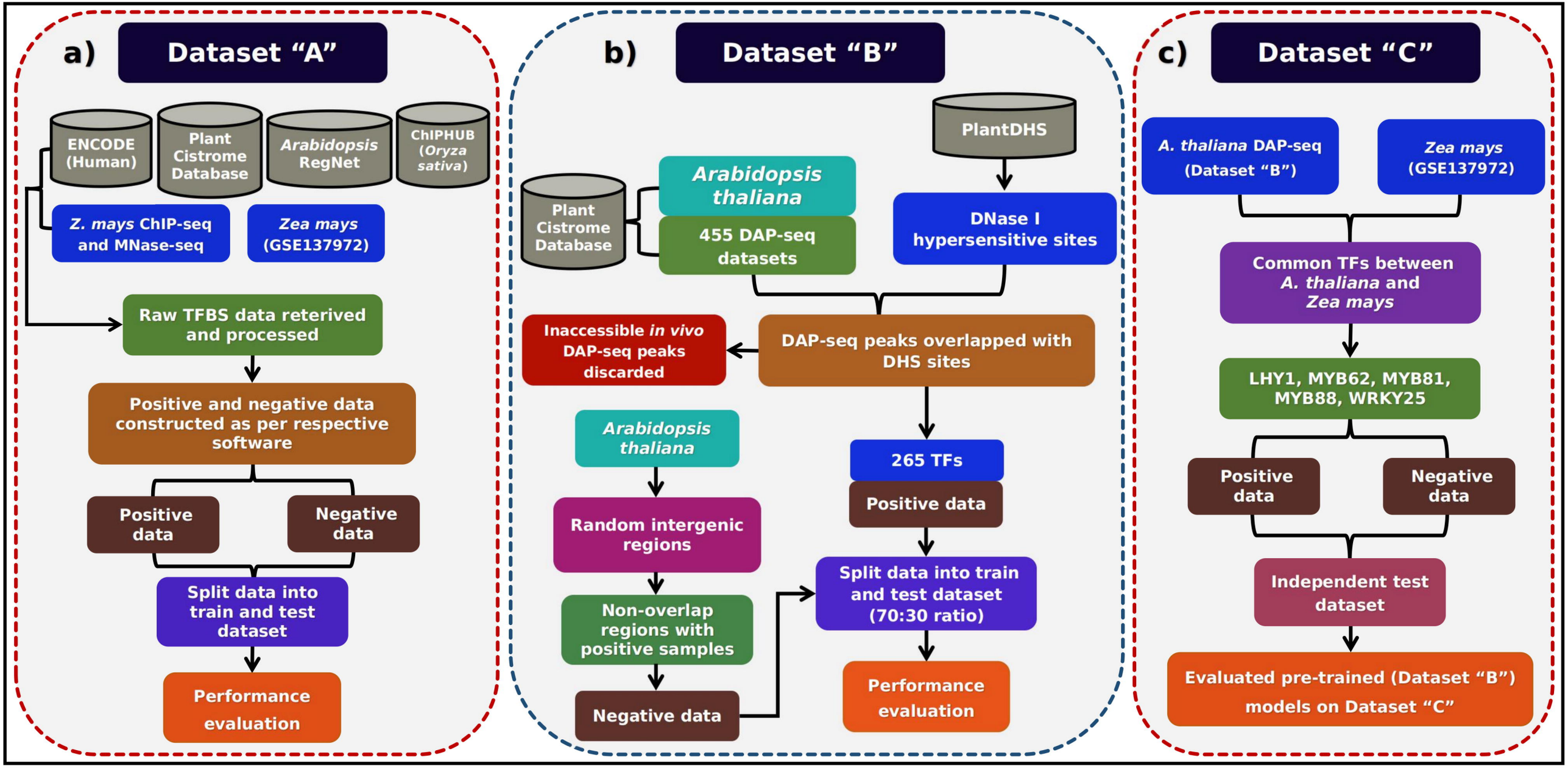
Supplementary Material S2: Hyper-parameters used by TFBS identification software.

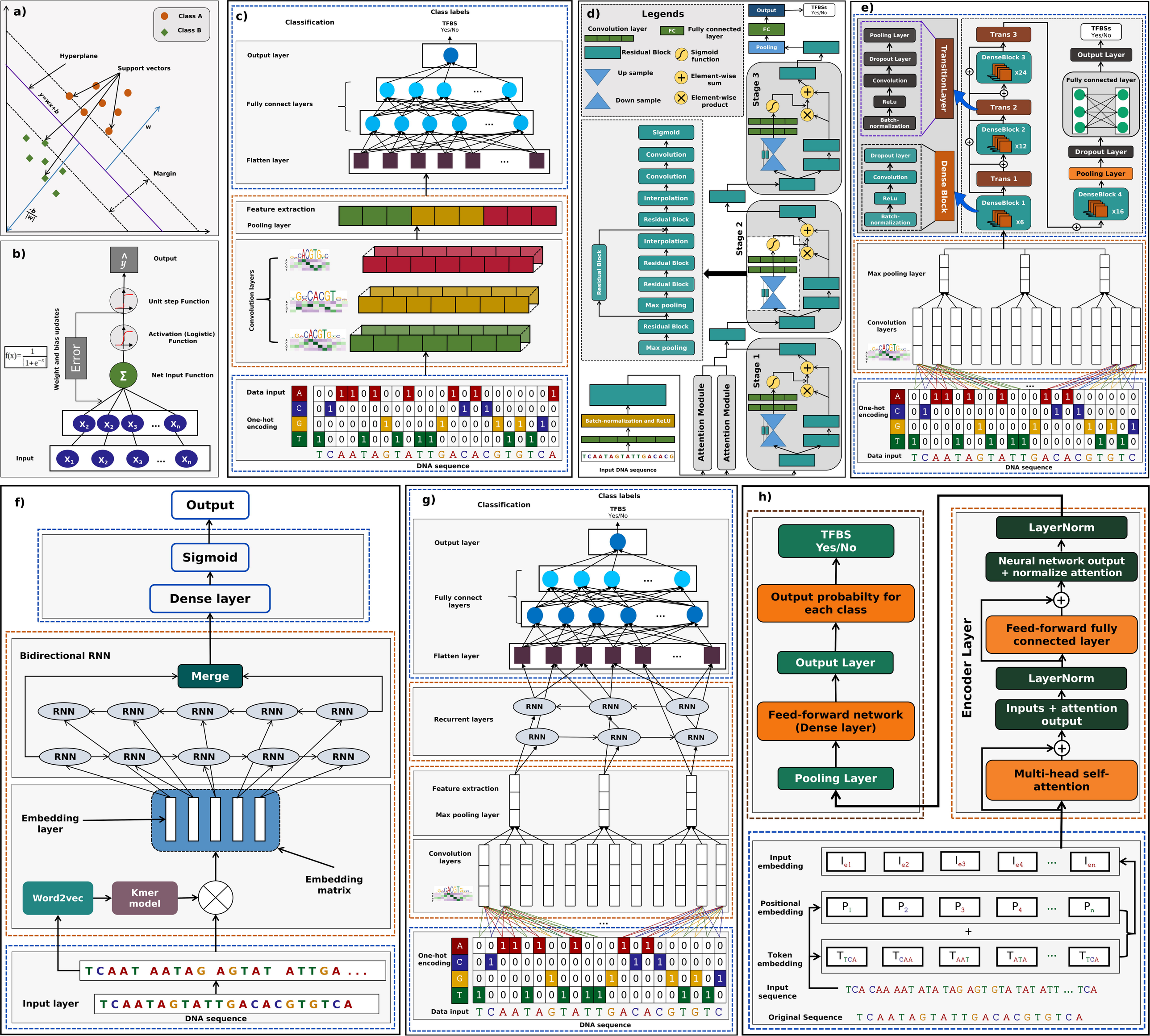
Supplementary Material S3: Bugs analyses, reporting and solutions of these errors of all the benchmarking tools.

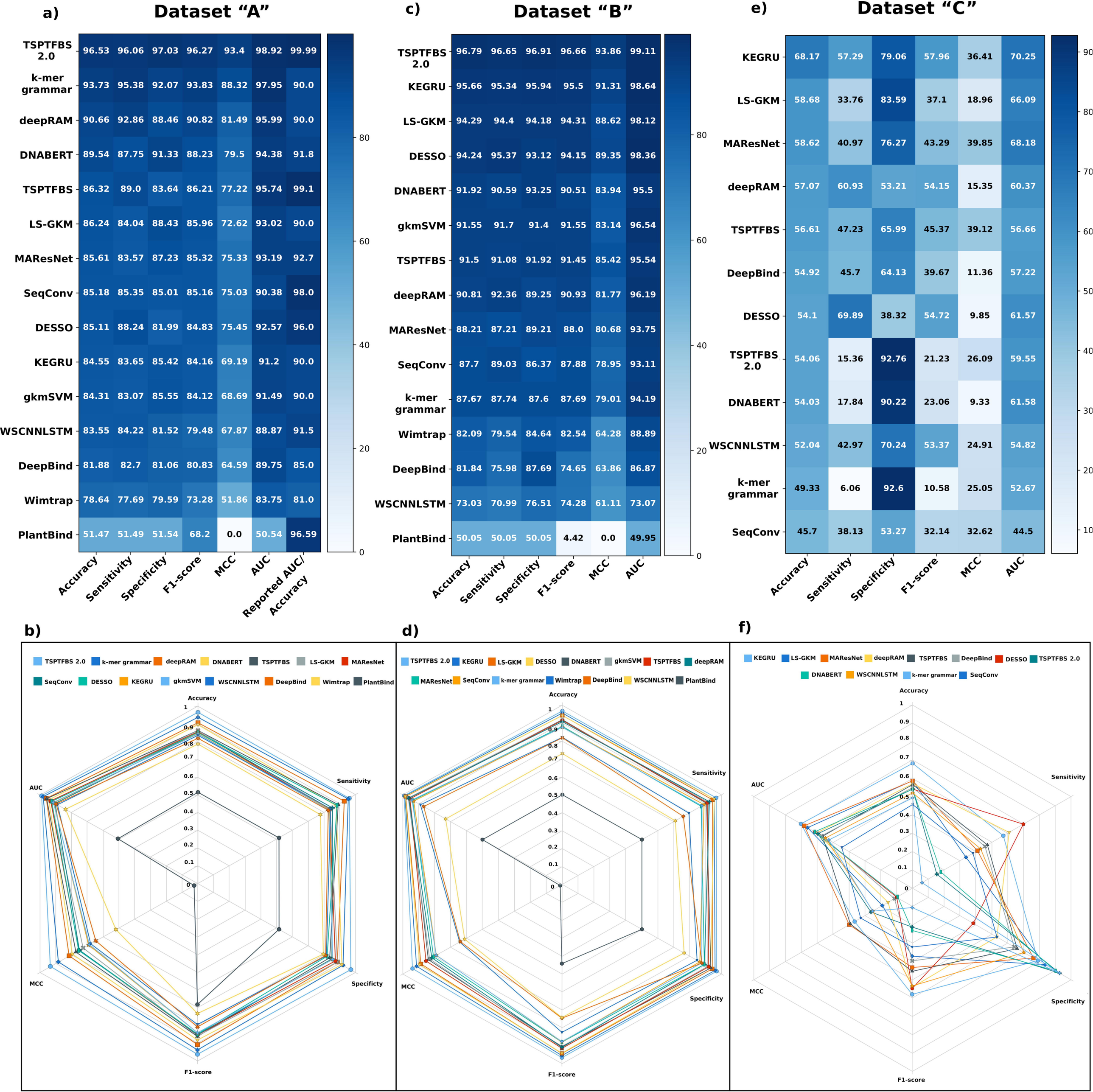

MEME:
Multiple EM for Motif Elicitation
BioProspector:
Gibbs sampling
Motif-Sampler:
Gibbs sampling
DeepBind
gkmSVM
LS-GKM
KEGRU
k-mer grammar
WSCNNLSTM
deepRAM
DESSO
TSPTFBS
DNABERT
SeqConv
MAResNet
Wimtrap
PlantBind
TSPTFBS 2.0
1994
1998
2001
2004
2007
2010
2013
2016
2018
2019
2021
2022
2023
ChIP-chip
PBM
ChIP-seq,
SELEX-seq
ChIP-exo
ChIP-nexus,
ChIPmentation
DAP-seq
High-throughput
ChIPmentation


a)
Data collection and processing
ENCODE project/
Plant Cistrome Database/
Arabidopsis RegNet/ GSE137972
Arabidopsis thaliana
Homo sapiens
Zea mays
Benchmark train data
Benchmark test data
b)
Encoding representations
One-hot encoding
Word representation
Shape encoding
A T G C
1 0 0 0
0 1 0 0
0 0 1 0
0 0 0 1
A A C T G G A C G A T
A A C T G G A
7-kmer
A C T G G A C
C T G G A C G
T G G A C G A
G G A C G A T
+
T 0 0 0 0 0 0 1
G 1 1 0 0 1 0 0
C 0 0 0 1 0 0 0
A 0 0 1 0 0 1 0
0.54838716,
0.20289855,
0.47589099,
0.26152913,
0.65238095,
0.87096774,
0.14492754,
0.59119497,
0.64516129,
0.21014493,
0.60587002
PWM
A 0.854758 0.000000 1.000000 0.000000 0.996661 0.000000 0.005008
C 0.001669 0.000000 0.000000 0.000000 0.003339 0.013356 0.000000
G 0.051753 1.000000 0.000000 0.000000 0.000000 0.008347 0.005008
T 0.091820 0.000000 0.000000 1.000000 0.000000 0.978297 0.989983
c)
Comparative benchmarking
Traditional machine learning
Support vector machine
gkmSVM, LS-GKM
Word representation
Logistic regression
k-mer grammar
Word representation
XGBoost
Wimtrap
PWM, bed file
Deep learning
CNN
DeepBind
deepRAM
TSPTFBS
SeqConv
MAResNet
TSPTFBS2.0
One-hot encoding
DESSO
One-hot encoding
/
Shape encoding
RNN
KEGRU
Word representation
deepRAM
Word representation
CNN/RNN
WSCNNLSTM
Word representation
+
One-hot encoding
PlantBind
One-hot encoding
Transformer
DNABERT
Word representation
d)
Performance evaluation
Sensitivity
Specificity
Accuracy
MCC
F1-score
AUC
e)
Evaluation
TSPTFBS-2.0 96.79 96.65 96.91 96.66 93.86 99.11
KEGRU 95.66 95.34 95.94 95.5 91.31 98.64
LS-GKM 94.29 94.4 94.18 94.31 88.62 98.12
DESSO 94.24 95.37 93.12 94.15 89.35 98.36
DNABERT 91.92 90.59 93.25 90.51 83.94 95.5
gkmSVM 91.55 91.7 91.4 91.55 83.14 96.54
TSPTFBS 91.5 91.08 91.92 91.45 85.42 95.54
deepRAM 90.81 92.36 89.25 90.93 81.77 96.19
MAResNet 88.21 87.21 89.21 88.0 80.68 93.75
SeqConv 87.7 89.03 86.37 87.88 78.95 93.11
K-mer grammar 87.67 87.74 87.6 87.69 79.01 94.19
Wimtrap 82.09 79.54 84.64 82.54 64.28 88.89
DeepBind 81.84 75.98 87.69 74.65 63.86 86.87
WSCNNLSTM 73.03 70.99 76.51 74.28 61.11 73.07
PlantBind 50.05 50.05 50.05 4.42 0.0 49.95
Accuracy
Sensitivity
Specificity
F1-score
MCC
AUC
80
60
40
20
0


Number
TF Family
0
10
20
30
40
AP2-EREBP
MYB
NAC
WRKY
bZIP
MYB-related
C2C2-DOF
G2-like
C2H2
HB
Homeobox
bHLH
SBP
Trihelix
ZF-HD
LOBAS2
C2C2-GATA
C3H
CPP
HSF
BZR
MADS
TCP
ABI3-VP1
ARF
BBR_BPC
BES1
C2C2-YABBY
CAMTA
GEBP
GRF
ND
REM
S1FA-like
SRS
nan
MYB 12.5%
NAC 12.1%
AP2-EREB 15.8%
WRKY 7.5%
bZIP 6.8%
MYB-related 6.4%
C2C2-DOF 5.3%
G2-like 4.5%
C2H2 4.2%
SRS 0.4%
S1FA-like 0.4%
REM 0.4%
ND 0.4%
GRF 0.4%
GEBP 0.4%
CAMTA 0.4%
C2C2-YABBY 0.4%
BES1 0.4%
BBR_BPC 0.4%
ARF 0.4%
ABI3-VP 0.4%
TCP 0.8%
MADS 0.8%
BZR 0.8%
HSF 1.1%
CPP 1.1%
C3H 1.1%
C2H2-GATA 1.1%
LOBAS 1.5%
ZF-HD 1.9%
Trihelix 1.9%
SBP 1.9%
bHLH 1.9%
Homeobox 2.3%
HB 2.3%


a)
Dataset "A"
ENCODE (Human)
Plant Cistrome Database
Arabidopsis RegNet
ChIPHUB (Oryza sativa)
Z. mays ChIP-seq and MNase-seq
Zea mays (GSE137972)
Raw TFBS data reterived and processed
Positive and negative data constructed as per respective software
Positive data
Negative data
Split data into train and test dataset
Performance evaluation
b)
Dataset "B"
PlantDHS
Arabidopsis thaliana
Plant Cistrome Database
455 DAP-seq datasets
DNase I hypersensitive sites
Inaccessible in vivo DAP-seq peaks discarded
DAP-seq peaks overlapped with DHS sites
Arabidopsis thaliana
265 TFs
Positive data
Random intergenic regions
Non-overlap regions with positive samples
Split data into train and test dataset (70:30 ratio)
Negative data
Performance evaluation
c)
Dataset "C"
A. thaliana DAP-seq (Dataset "B")
Zea mays (GSE137972)
Common TFs between A. thaliana and Zea mays
LHY1, MYB62, MYB81, MYB88, WRKY25
Positive data
Negative data
Independent test dataset
Evaluated pre-trained (Dataset "B") models on Dataset "C"


a)
Class A
Class B
Hyperplane
Support vectors
y=wx+b
w
Margin
b)
Output
Unit step Function
Activation (Logistic) Function
Net Input Function
Weight and bias updates
Error
Input
c)
Classification
Class labels
TFBS Yes/No
Output layer
Fully connect layers
Flatten layer
Feature extraction
Pooling layer
Convolution layers
Data input
One-hot encoding
DNA sequence
d)
Legends
Convolution layer
Fully connected layer
Residual Block
Sigmoid function
Up sample
Element-wise sum
Down sample
Element-wise product
Output
FC
Pooling
Stage 3
Stage 2
Stage 1
Sigmoid
Convolution
Interpolation
Residual Block
Max pooling
Batch-normalization and ReLU
Attention Module
Input DNA sequence
e)
Pooling Layer
Dropout Layer
Convolution
ReLu
Batch-normalization
TransitionLayer
Dropout layer
Dense Block
Trans 3
Trans 2
Trans 1
DenseBlock 3
DenseBlock 2
DenseBlock 1
DenseBlock 4
x24
x12
x6
x16
TFBSs Yes/No
Output Layer
Fully connected layer
Dropout Layer
Pooling Layer
Max pooling layer
Convolution layers
One-hot encoding
Data input
DNA sequence
f)
Output
Sigmoid
Dense layer
Bidirectional RNN
Merge
RNN
Embedding layer
Embedding matrix
Word2vec
Kmer model
Input layer
DNA sequence
g)
Classification
Class labels
TFBS Yes/No
Output layer
Fully connect layers
Flatten layer
Recurrent layers
Feature extraction
Max pooling layer
Convolution layers
One-hot encoding
Data input
DNA sequence
h)
TFBS Yes/No
Output probabilty for each class
Output Layer
Feed-forward network (Dense layer)
Pooling Layer
Encoder Layer
LayerNorm
Neural network output + normalize attention
Feed-forward fully connected layer
Inputs + attention output
Multi-head self-attention
Input embedding
Positional embedding
Token embedding
Input sequence
Original Sequence


a)
Dataset "A"
TSPTFBS 2.0 96.53 96.06 97.03 96.27 93.4 98.92 99.99
k-mer grammar 93.73 95.38 92.07 93.83 88.32 97.95 90.0
deepRAM 90.66 92.86 88.46 90.82 81.49 95.99 90.0
DNABERT 89.54 87.75 91.33 88.23 79.5 94.38 91.8
TSPTFBS 86.32 89.0 83.64 86.21 77.22 95.74 99.1
LS-GKM 86.24 84.04 88.43 85.96 72.62 93.02 90.0
MAResNet 85.61 83.57 87.23 85.32 75.33 93.19 92.7
SeqConv 85.18 85.35 85.01 85.16 75.03 90.38 98.0
DESSO 85.11 88.24 81.99 84.83 75.45 92.57 96.0
KEGRU 84.55 83.65 85.42 84.16 69.19 91.2 90.0
gkmSVM 84.31 83.07 85.55 84.12 68.69 91.49 90.0
WSCNNLSTM 83.55 84.22 81.52 79.48 67.87 88.87 91.5
DeepBind 81.88 82.7 81.06 80.83 64.59 89.75 85.0
Wimtrap 78.64 77.69 79.59 73.28 51.86 83.75 81.0
PlantBind 51.47 51.49 51.54 68.2 0.0 50.54 96.59
Accuracy
Sensitivity
Specificity
F1-score
MCC
AUC
Reported AUC/ Accuracy
c)
Dataset "B"
TSPTFBS 2.0 96.79 96.65 96.91 96.66 93.86 99.11
KEGRU 95.66 95.34 95.94 95.5 91.31 98.64
LS-GKM 94.29 94.4 94.18 94.31 88.62 98.12
DESSO 94.24 95.37 93.12 94.15 89.35 98.36
DNABERT 91.92 90.59 93.25 90.51 83.94 95.5
gkmSVM 91.55 91.7 91.4 91.55 83.14 96.54
TSPTFBS 91.5 91.08 91.92 91.45 85.42 95.54
deepRAM 90.81 92.36 89.25 90.93 81.77 96.19
MAResNet 88.21 87.21 89.21 88.0 80.68 93.75
SeqConv 87.7 89.03 86.37 87.88 78.95 93.11
k-mer grammar 87.67 87.74 87.6 87.69 79.01 94.19
Wimtrap 82.09 79.54 84.64 82.54 64.28 88.89
DeepBind 81.84 75.98 87.69 74.65 63.86 86.87
WSCNNLSTM 73.03 70.99 76.51 74.28 61.11 73.07
PlantBind 50.05 50.05 50.05 4.42 0.0 49.95
Accuracy
Sensitivity
Specificity
F1-score
MCC
AUC
e)
Dataset "C"
KEGRU 68.17 57.29 79.06 57.96 36.41 70.25
LS-GKM 58.68 33.76 83.59 37.1 18.96 66.09
MAResNet 58.62 40.97 76.27 43.29 39.85 68.18
deepRAM 57.07 60.93 53.21 54.15 15.35 60.37
TSPTFBS 56.61 47.23 65.99 45.37 39.12 56.66
DeepBind 54.92 45.7 64.13 39.67 11.36 57.22
DESSO 54.1 69.89 38.32 54.72 9.85 61.57
TSPTFBS 2.0 54.06 15.36 92.76 21.23 26.09 59.55
DNABERT 54.03 17.84 90.22 23.06 9.33 61.58
WSCNNLSTM 52.04 42.97 70.24 53.37 24.91 54.82
k-mer grammar 49.33 6.06 92.6 10.58 25.05 52.67
SeqConv 45.7 38.13 53.27 32.14 32.62 44.5
Accuracy
Sensitivity
Specificity
F1-score
MCC
AUC
b)
TSPTFBS 2.0 k-mer grammar deepRAM DNABERT TSPTFBS LS-GKM MAResNet SeqConv DESSO KEGRU gkmSVM WSCNNLSTM DeepBind Wimtrap PlantBind
Accuracy
Sensitivity
Specificity
F1-score
MCC
AUC
d)
TSPTFBS 2.0 KEGRU LS-GKM DESSO DNABERT gkmSVM TSPTFBS deepRAM MAResNet SeqConv k-mer grammar Wimtrap DeepBind WSCNNLSTM PlantBind
Accuracy
Sensitivity
Specificity
F1-score
MCC
AUC
f)
KEGRU LS-GKM MAResNet deepRAM TSPTFBS DeepBind DESSO TSPTFBS 2.0 DNABERT WSCNNLSTM k-mer grammar SeqConv
Accuracy
Sensitivity
Specificity
F1-score
MCC
AUC