BMC Bioinformatics

RESEARCH Open Access

Supervised promoter recognition: a benchmark framework

Raul I. Perez Martell*, Alison Ziesel, Hosna Jabbari and Ulrike Stege

*Correspondence:
ivanpmartell@uvic.ca
Department of Computer Science, University of Victoria, Victoria, BC, Canada

Abstract

Motivation: Deep learning has become a prevalent method in identifying genomic regulatory sequences such as promoters. In a number of recent papers, the performance of deep learning models has continually been reported as an improvement over alternatives for sequence-based promoter recognition. However, the performance improvements in these models do not account for the different datasets that models are evaluated on. The lack of a consensus dataset and procedure for benchmarking purposes has made the comparison of each model's true performance difficult to assess.

Results: We present a framework called Supervised Promoter Recognition Framework ('SUPR REF') capable of streamlining the complete process of training, validating, testing, and comparing promoter recognition models in a systematic manner. *SUPR REF* includes the creation of biologically relevant benchmark datasets to be used in the evaluation process of deep learning promoter recognition models. We showcase this framework by comparing the models' performances on alternative datasets, and properly evaluate previously published models on new benchmark datasets. Our results show that the reliability of deep learning ab initio promoter recognition models on eukaryotic genomic sequences is still not at a sufficient level, as overall performance is still low. These results originate from a subset of promoters, the well-known RNA Polymerase II core promoters. Furthermore, given the observational nature of these data, cross-validation results from small promoter datasets need to be interpreted with caution.

Keywords: Machine learning, Deep learning, Bioinformatics, Promoter recognition

Introduction

Promoters are non-coding genomic elements necessary for the expression of their associated gene(s). Most promoters include transcription factor binding sites (TFBS): short, often palindromic DNA sequences bound by transcription factor proteins to provide greater control over a promoter's activity. TFBS can be summarised as 'motifs' [1] that represent the set of related short sequences preferred by a given transcription factor (TF). A frequent example of a motif found in eukaryotic promoters is the TATA box, a cis-regulatory element characterised by its consensus sequence of repeating T and A base pairs. Other relevant promoter motifs can be found in the JASPAR database [2]. A minimal promoter must be able to recruit RNA polymerase (RNAP) to allow for transcription to occur. Promoters can be separated into three main regions: core, proximal, and distal. Core promoters are located closest to their gene, containing the transcription start site (TSS) of the gene and usually include general TFBS to create an RNAP binding site. Proximal promoters are located approximately 200 base pairs upstream from the TSS, and are the DNA regions where more gene-specific TF bind. Distal promoters are generally found thousands of base pairs upstream from the core promoter and include several other TFBS that can recruit proteins to enhance or silence the RNAP's transcription process. While a promoter sequence itself is not expressed, mutations to the promoter sequence can have a prominent impact on gene expression. E.g., recombination events may occur, which produce a novel promoter-gene combination, resulting in a significantly different expression pattern for the controlled gene; these are observed in certain types of cancer [3]. As the central regulatory feature for gene expression, promoters can provide potentially significant information used for predicting downstream gene expression patterns. Rudge et al. [4] have shown that understanding gene expression patterns can lead to therapies for disease control or prevention. Furthermore, biological experimentation for developing insights into the mechanisms of therapies and cell machinery are done in organisms where promoters might not be as well defined as for humans and other model organisms. Therefore, understanding and reliably identifying promoters within genomes are essential to this goal.

Identification of promoters through their genomic sequence is highly complex because of their sequence and structure diversity, which excludes any universal promoter elements. Considering the central dogma of molecular biology, all of the non-genetic effects involved in promoter activity could, in theory, be ultimately mapped to the DNA. This allows the genetic sequence of promoters, transcription factors, and other interacting proteins available in the genome to have the ability to fully characterise a promoter. Therefore, we focus on the promoter identification problem known as *ab initio promoter recognition*, which entails the use of DNA sequences solely to identify promoters. The work presented here can aid in any type of ab initio promoter recognition, but initially aims to help in recognition models for eukaryotic RNAP II core promoters since they take part in the transcription of RNAs that will become messenger RNAs (mRNAs) and also small regulatory RNAs—the former being the products used by ribosomes to synthesise proteins, while the latter play a role in regulatory processes such as activation and inhibition of gene transcription. Core promoters are chosen for their relatively small size to other promoter sequences and the abundance of work on them. Recognition models for core promoter sequences are also constantly regarded in the literature as highly precise or accurate in a multitude of metrics. Therefore, we aim to validate these findings that have become prevalent within the promoter recognition field.

A review by Li et al. [5] revealed that current promoter identification efforts mainly lie in supervised machine learning (ML) techniques, with deep learning (DL) as the latest promising approach by utilizing data from different High-Throughput Sequencing (HTS) methods. Early ML models require the use of biological signals to be 'handcrafted' into features. Crafting these features can become especially problematic when domain-knowledge is incomplete, as is the case in promoter recognition. DL can account for incomplete domain-knowledge by 'learning' these features, while additional 'handcrafted' features can be supplied when domain-knowledge is present. Once ab initio methods can accurately recognise promoters, the methods and their outputs can be analysed to understand the sequences affecting specific promoters and their relation to genetic activity, as well as provide further insights into the genomic characteristics of promoters.

Other promoter recognition tools also exist [6–8]. It is interesting to note that even the state-of-the-art for these non-supervised ML tools are not able to achieve the performance of the DL models examined here, giving us a reason to further explore why DL models are obtaining such increased performance.

Additional DL models exist for TF binding prediction [9–12]. These models can be used as a mean for predicting the functionality of DNA sequences after fine-tuning. This suggests the ability to also predict promoters when specifically re-training the models for this task. Although possible to use them as promoter recognition tools, no promoter recognition results could be obtained easily from them; as such, we do not consider them for this work.

Performance of all ML-based methods is heavily impacted by the underlying training data used. Raeder et al. [13] demonstrated that this problem can be exacerbated when using imbalanced datasets, such as promoter recognition. This imbalance is evident, as promoter sequences generally account for less than 1% of a genome. Since we are interested in the minority class (promoters) which contains significantly fewer instances, models will tend to focus on learning the characteristics of the majority class (non-promoters), therefore neglecting to learn promoter features. Scientists must ensure that the training data used in the ML process is a statistically representative sample of the many genomic sequences found in nature. It is also crucial that a held-out dataset be used for testing and comparing different models' performances. This means that a trained model must not have previously seen any data within the testing dataset to ensure the validity of its performance metrics. Results from common evaluation methods for an ML model, such as cross-validation, can be especially deceptive when the overall dataset used for training and testing is limited. Therefore, results obtained in this manner can create the impression that a model's performance is adequate when in reality this may not be the case. Early ML models for ab initio promoter recognition were benchmarked by Bajic et al. [7]. The benchmarking process was done using a limited subset of available genomic data, and found the models to be insufficiently sensitive for promoter recognition. Deep learning promoter recognition (DLPR) models frequently exhibit improved recognition performance over previous models through cross-validation assessments. The caveat is that these models are still trained and tested on similarly limited datasets, generally comprising less than 60,000 DNA sequences, which cannot possibly cover a representative sample of the many genomic sequences found in nature.

We present a framework called *Supervised Promoter Recognition Framework* ('SUPR REF') capable of streamlining the complete process of training, validating, testing, and comparing promoter recognition models in a systematic manner. Our framework includes two different but compatible methods for the creation of benchmark datasets, to be used in the evaluation process of DLPR models. The first method utilises sequence alignment to find the promoter sequences in a genome, while the second uses promoter annotations that specify the TSS to extrapolate a promoter's location within the genome.

In addition, we implemented previously published models where no source code was provided to help ease reproducible research within the field. We benchmark published DL promoter recognition models using *SUPR REF* and assess their performance on realistic genome datasets.

We demonstrate the benchmarking procedure of *SUPR REF* on three previously published studies: CNNProm by Umarov and Solovyev [14], ICNNP by Qian et al. [15], and DProm by Oubounyt et al. [16]. We also include more thorough experimentation on the most advanced architecture implemented (DProm)—a convolutional recurrent neural network. Benchmarking of previous work includes a comparison between their training and testing datasets in addition to comparisons of strategies to create synthetic datasets. Experiments include multiple training and testing datasets showcasing a comparison of different promoter annotations, sampling methods to overcome the imbalanced data, and the effect of output functions in neural networks. The previously mentioned benchmarks demonstrate the capability to implement literature models while testing them on a genomic scale. Further benchmarking is also performed on the state-of-the-art model known as DNABERT by Ji et al. [17] to understand the current capabilities of promoter recognition tools. In this work, we focus on the most recent and accurate DLPR models, whose superior performance was previously established [14, 16, 17] against other models [7] and were found to have better performance.

We show that the reliability of deep learning ab initio promoter recognition models on eukaryotic genomic sequences is still not at an acceptable level, as overall performance (measured by MCC) is low. We achieve this by utilising *SUPR REF* to create larger biologically relevant datasets to test models more thoroughly. The remainder of the paper is organised as follows. The "Materials and methods" section describes *SUPR REF* and the benchmarks performed to obtain our results. In the "Results and discussion" section, we examine the results from our implementations and their comparisons, along with the benchmarking results. Furthermore, we provide results of experiments on interesting functionality within DLPR models and analysis of datasets that provide these models with biased performance metrics. Finally, "Conclusions and future work" recapitulates our research aims and its significance, as well as suggesting approaches for future work.

Materials and methods

This section introduces our framework *SUPR REF* and the implemented models and datasets from previously published studies to showcase its use.

We then introduce larger benchmarking datasets that reflect the statistics of promoter sequences found in nature, and suggest a method for a fair benchmarking of the DLPR models. Finally, we describe the metrics within the evaluation procedure for model comparison and benchmarking.

SUPR REF

Different frameworks exist for generalised use of DL in bioinformatics [18, 19]. However, their use in more specific tasks would require extensive re-engineering. While some specialised frameworks exist in other domains (see [20, 21]), to the best of our knowledge none exists for promoter recognition. A promoter recognition framework should aid in rapid advancement of novel recognition techniques, but must also offer benchmarking

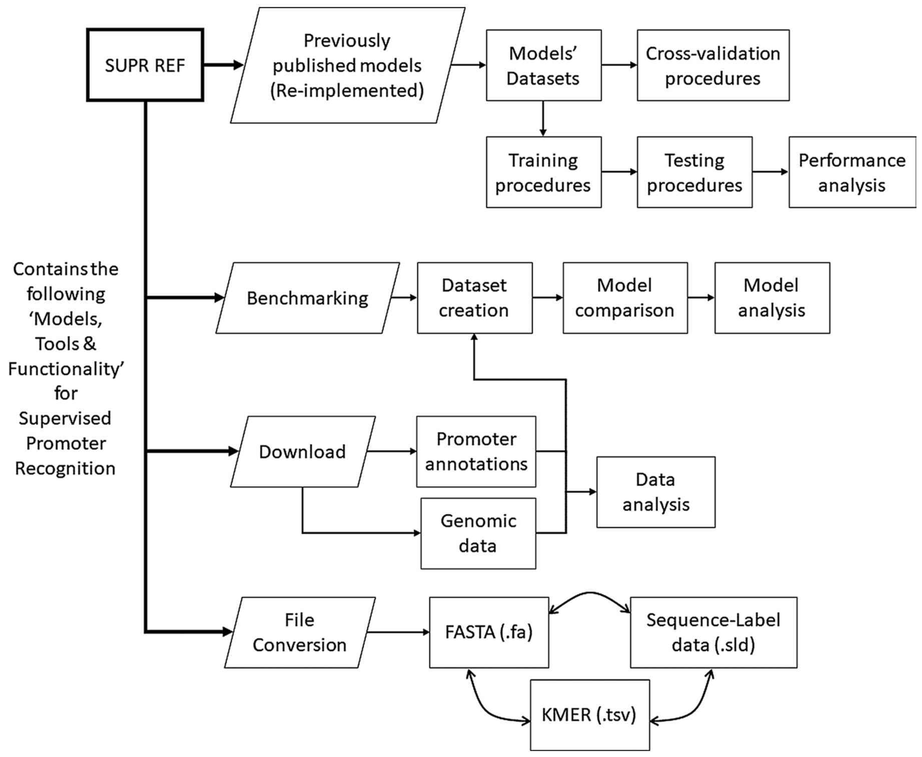

Fig. 1 Schematic displaying main functionalities from SUPR REF. The framework is described through 4 main functionalities: Re-implementation of previous models, Benchmarking capabilities for ML models, Downloading capabilities for annotation and genomic data, and File conversion capabilities that different models require

capabilities that can clearly assess the significance of these advances over previous models. It is important for our benchmarking process to be simple and systematic to limit the amount of time spent during this process and to help researchers focus on the creation of better models. These capabilities can be achieved using *SUPR REF*.

SUPR REF offers an advantage of domain-knowledge datasets and models that can account for promoter-specific problems that would otherwise go unnoticed in a generalised setting. Additionally, *SUPR REF* allows researchers without prior DL knowledge to compare models in a systematic way and to make informed decisions of which model to use within their specific context.

SUPR REF is a Python 3 framework designed to ease the training and testing procedures of machine learning models for promoter recognition. We made use of pyfaidx [22] to enable the efficient processing of DNA sequences within python. The framework can be installed as a command-line tool for automation and productivity purposes. It contains four main functionalities covering (1) implementations of published models, (2) benchmarking, (3) data acquisition, and (4) data conversion. See Fig. 1 for an overview of *SUPR REF*, including the different utilities within each main functionality of *SUPR REF*.

SUPR REF contains third party implemented architectures of recently published models, and the creation of their datasets. The datasets can be split into non-overlapping training and testing datasets between all models enabling proper model comparison. *SUPR REF* also includes training, testing, and cross-validation procedures for each model to analyse and visualise the models' outputs.

The benchmarking system offers two main utilities, as well as analysis notebooks to help visualise and interpret how the trained models function. The `create` utility allows the creation of datasets from two approaches. The annotation approach utilises promoter annotation data and its corresponding reference genome to create a dataset. The sequence alignment approach creates a dataset by mapping the promoters in a genome using sequence alignment tools. Next, the `test` utility tests a trained DLPR model with previously published datasets or *SUPR REF*-created datasets. Three additional aiding utilities are offered.

As a framework, SUPR REF can also aid in more general promoter recognition endeavours, including data acquisition, data conversion, and ML training with two additional utilities. The `download` utility is a data acquisition tool which aids in downloading promoter annotation data from biologically tested sources such as EPFL's Eukaryotic Promoter Database [23], UCSC's upstream sequences [24] and Riken's FANTOM 5 project [25]. The `conversion` utility helps convert data from our dataset creation tool to and from common datasets used in bioinformatics tools, such as fasta and kmers (from DNABERT). Finally, a `train` utility is available to help train promoter recognition models utilising DL approaches (user-made PyTorch modules) and other scikit-learn machine learning algorithms. These models can be trained on *SUPR REF*-created datasets which can be tailored to your specific needs within promoter recognition, such as promoter length and amount of upstream and downstream bases within the promoter region.

All utilities within *SUPR REF* include multiple parameters that can be adjusted based on the user's needs, such as different dataset creation properties, and training and testing parameters. More information on each utility and its parameters can be found on the *SUPR REF* documentation.

DL models

As a recent field of study, DLPR models have not yet been systematically reviewed, although a recent overview can be found in [26]. In this work, we showcase SUPR REF by focusing on the comparison of four recent DLPR models created for the human (hg38) genome, including the state-of-the-art DLPR model known as DNABERT. Human promoters are some of the most studied and annotated in the literature within eukaryotes. Therefore, human promoter annotations should contain adequately comprehensive data to successfully train and test an effective model.

The first method, **CNNProm**, was proposed by Umarov and Solovyev [14] with a convolutional neural network (CNN) architecture. Promoter sequences were obtained from the Eukaryotic Promoter Database (EPD) [23], while the non-promoter sequences were random subsequences within genes located after their first exons. Since the dataset is ambiguously described and only partially attainable, we obtained a dataset that to the best of our abilities resembled its description. The created dataset consists of promoter sequences from EPD that span 251 bases. Non-promoter sequences span the same length as promoters and are obtained from random locations after the first exon of each human gene's sequence. To obtain these non-promoter sequences, we query the UCSC genome database [24]. Promoters were further separated into two classes and classified by different CNNProm models: promoters with a TATA box motif (TATA), and promoters without a TATA box motif (non-TATA). The CNNProm TATA model had a slightly different architecture from CNNProm non-TATA model. The main difference being the number of filters in the CNN and the pooling size, where TATA had a lower value in both cases. To reproduce their results, we used the previously created dataset with sequences of length 251. To properly compare them to other models, we used the same method of obtaining the sequences with an added 49 more bases upstream, totalling 300 bases for the sequences length.

Subsequently, **ICNNP** is a DLPR method by Qian et al. [15], which was described as an improvement over CNNProm. Similar to CNNProm, ICNNP consists of a convolutional neural network architecture. Unlike CNNProm, which classified human sequences of length 251, ICNNP classifies sequences of length 300. This increase in length includes 49 additional bases in the upstream region of the promoter sequence. The ICNNP dataset includes a mix of TATA and non-TATA promoters from EPD, while the non-promoter sequences consist of human introns and coding sequences (CDS) collected from the 1998 GENIE dataset [27]. The non-promoter dataset of ICNNP was originally compiled by the Berkeley Drosophila Genome Project (BDGP) [28] for use as a representative benchmark dataset of human DNA sequences. To create ICNNP, the authors first focused on assessing the importance of various motif locations, referred to as 'element sequences', using *Support Vector Machines*. Paraphrasing Qian et al., element sequences are sequence motifs located near the TSS and their location is found relative to the TSS. Non-element sequences are the rest of the DNA (sequences where no common motifs are found). For the element and non-element sequences to be utilised properly in the ICNNP model, they must be located at specific locations within the input DNA sequence. This means that the element sequences' locations or the TSS must be known beforehand. The predetermined locations were then extracted from the sequence to be used along a compressed (max-pooled) version of the complete sequence for classification by the neural network. Element sequences were found to account for most of the differentiative signal within promoters, while the compression of non-elements was found to help remove noise from the model. Therefore, the main difference between CNNProm and ICNNP was not from the neural network architecture, but from the preprocessing of the sequences.

Next, **DeePromoter (DProm)** [16] consists of a convolutional long short-term memory (CLSTM) architecture. Similar to the work by Umarov and Solovyev [14], Oubounyt et al. created different models for promoters with and without a TATA box motif, along with model separation by species. Promoter sequences were obtained from EPD; for non-promoter sequences, DProm used synthetically created sequences by changing the promoter sequences. DProm models classify human sequences with a length of 300 bases.

Finally, **DNABERT** [17] was chosen as the state-of-the-art for promoter recognition for its favorable comparison against the previously mentioned models and an improved version of them known as PromID [29]. DNABERT consists of the 'Transformer' deep learning architecture that has been popular in the natural language processing field. Bidirectional Encoder Representations from Transformers (BERT), was first introduced by Google [30] where it delivered state-of-the-art results in many NLP benchmark tests. BERT has been made accessible to other fields by using fine-tuning. In the case of DNABERT, BERT was fine-tuned for use on DNA sequences. Promoter sequences were obtained from EPD for both TATA and non-TATA promoters. TATA and non-TATA non-promoter sequences were obtained in different manners. TATA non-promoter sequences were randomly picked from genomic regions outside promoter regions which also contained a TATA motif. non-TATA non-promoter sequences were created in the same manner as DProm.

The first three models encode DNA sequences using one-hot encoding, i.e., a matrix where each of the rows consists of unique bases (A, C, G, T) and the columns are the locations in the sequence. Sequences in ICNNP are further separated into 'element' and 'non-element' subsequences before one-hot encoding. In contrast, sequences in DNABERT are pre-processed into k-mer tokens and are represented as a matrix containing numerical vectors for each token. The dataset for DNABERT contained 59,196 sequences comprising of 6130 sequences (50% promoters) for 'Human TATA' model and 53,066 sequences (50% promoters) for 'Human non-TATA' model. These two datasets were mixed into a 'complete model' for general (TATA and non-TATA) promoter recognition.

Each of the previous DLPR methods was assessed on a different dataset. The dataset for CNNProm contained 57,224 DNA sequences, comprising of 9682 sequences (14.72% promoter) for 'Human TATA' model and 47,542 sequences (41.67% promoter) for 'Human non-TATA' model. The dataset for ICNNP contained the least amount of sequences with 12,391 sequences (57.75% promoter). Finally, the dataset for DProm contained 59,194 sequences comprising of 6130 sequences (50% promoters) for 'Human TATA' model and 53,064 sequences (50% promoters) for 'Human non-TATA' model. The specific number of promoters and non-promoters for the three implemented methods are available in Table 1.

Benchmark datasets

SUPR REF contains two dataset creation approaches. The sequence alignment approach receives as input promoter sequences and requires a local alignment tool (e.g. BLAST) to align them to a genome. When exact promoter sequences (without mutations or indels) are found in a genome, this approach might not be as accurate or efficient as exact string matching algorithms. Therefore, we recommend using this approach when annotations are unavailable, or when there are slight variations in the genome or promoter sequences as in the case of non-reference genomes (e.g. single-cell and whole-genome sequencing). The annotation approach requires annotations containing the TSS locations of genes. The annotations for this study were obtained from MGA database [31]. MGA data are collected from multiple sources, including EPD. The exact genome annotation used is called `Hs_EPDnew_006_hg38`.[1] With the TSS locations denoted as $+1$, a promoter region is extrapolated by obtaining a user-defined number of bases upstream and bases downstream of the TSS. To match with the previously described DL

[1] https://ccg.epfl.ch/mga/hg38/epd/Hs_EPDnew_006_hg38.sga.gz.

Table 1 Performance results from implemented human DLPR models

		CNNProm dataset												**ICNNP dataset**			
		1426 promoters				**19,811 promoters**				**21,237 promoters**				**7156 promoters**			
		8256 non-promoters				**27,731 non-promoters**				**35,987 non-promoters**				**5235 non-promoters**			
		TATA				**Non-TATA**				**Complete dataset**				**Complete dataset**			
		MCC	**PPV**	**Sn**	**Sp**	**MCC**	**PPV**	**Sn**	**Sp**	**MCC**	**PPV**	**Sn**	**Sp**	**MCC**	**PPV**	**Sn**	**Sp**
OS*	1	90	–	95	98	89	–	90	98	–	–	–	–	–	–	82	79
	2	–	–	–	–	–	–	–	–	–	–	–	–	–	–	90	87
	3	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CT*	1	92	95	91	99	72	85	83	89	73	85	82	91	–	–	–	–
	CI	90–93	94–96	89–93	99–99	71–73	81–88	80–86	86–92	72–74	82–87	79–84	89–93	–	–	–	–
	2	–	–	–	–	–	–	–	–	–	–	–	–	64	83	88	74
	CI	–	–	–	–	–	–	–	–	–	–	–	–	61–67	80–87	83–93	66–82
	3	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
	CI	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CT	1	86	82	94	97	73	91	76	95	76	88	82	94	69	76	87	84
	2	–	–	–	–	–	–	–	–	69	76	87	84	70	88	87	83
	3	16	18	91	28	− 5	41	93	05	− 4	37	95	03	07	59	96	07
CT^{+}	1	94	93	97	99	76	89	83	93	80	90	84	95	74	93	84	91
	2	–	–	–	–	–	–	–	–	72	79	86	87	82	95	89	93
	3	19	18	99	22	− 3	41	96	03	0	37	97	03	07	59	96	07

Table 1 (continued)

		DProm dataset											
		3065 promoters				**26,532 promoters**				**29,597 promoters**			
		3065 non-promoters				**26,532 non-promoters**				**29,597 non-promoters**			
		TATA				**Non-TATA**				**Complete dataset**			
		MCC	**PPV**	**Sn**	**Sp**	**MCC**	**PPV**	**Sn**	**Sp**	**MCC**	**PPV**	**Sn**	**Sp**
OS*	1	62	75	91	–	26	58	83	–	–	–	–	–
	2	–	–	–	–	–	–	–	–	–	–	–	–
	3	88	93	95	–	92	97	95	–	–	–	–	–
CT*	1	–	–	–	–	–	–	–	–	–	–	–	–
	CI	–	–	–	–	–	–	–	–	–	–	–	–
	2	–	–	–	–	–	–	–	–	–	–	–	–
	CI	–	–	–	–	–	–	–	–	–	–	–	–
	3	90	95	95	95	91	96	95	96	91	97	94	97
	CI	88–92	92–98	93–97	91–98	90–92	94–97	94–97	94–97	91–92	95–98	92–96	95–98
CT⁻	1	16	18	91	28	− 5	41	93	05	− 4	37	95	03
	2	–	–	–	–	–	–	–	–	27	58	87	37
	3	86	95	91	95	91	98	93	98	91	96	95	96
CT⁺	1	65	74	95	67	14	54	82	29	12	53	84	26
	2	–	–	–	–	–	–	–	–	27	58	86	38
	3	99	1	99	1	94	98	96	98	94	98	96	98

OS indicates model results from original studies. CT shows cross-testing results from our implemented models. (1) CNNProm, (2) ICNNP, (3) DProm model. CT^{+}: results from training and testing datasets with some overlapping sequences. CT : results from properly split training and testing datasets having no overlapping sequences. CT^{*} and OS^{*}: results obtained from 10-fold cross-validation. All values expressed as percentages. 99% confidence interval values for our 10-fold cross validation results shown within CI rows models, we obtained promoter regions spanning from − 249 to + 50 for each TSS annotation. The annotation approach was chosen to create the benchmark datasets in this study, as the data was readily available for the hg38 genome.

With *SUPR REF*, we created multiple benchmark datasets using the annotation approach and a sliding window over the hg38 genome. The creation parameters of these benchmark datasets follow a stride of 50 and a window size of 300 to resemble the datasets for previous models. Any window with an ambiguous base 'N' was removed to avoid model noise and bias. Sequences were categorised as promoter if at least 250 out of 300 bases (83%) were contiguously overlapping with true promoter sequences. All other sequences were categorised as non-promoter.

These benchmark datasets follow the IO tagging scheme, where "I" means inside (promoter sequence) and "O" means outside (non-promoter sequence). Tagging schemes stem from the Natural Language Processing (NLP) community [32]. They translate well into promoter recognition where DNA is regarded as a text corpus, and promoters as entities within that corpus.

To elicit a more realistic performance of the DLPR models, we created a larger testing dataset comprising of hg38 chromosome 1 and 2 (hg38chr1&2). This larger dataset contains 422,912 sequences, as opposed to less than 60,000 sequences from each study's cross-validation test. The dataset contains overlapping sequences designed to test sequences where a few bases on the sequences' edges are different. This could alert us if a model has strict positional learned behaviour or if it is overfitting on specific sequence patterns. The hg38chr1&2 dataset was split into its respective chromosomes. The hg38 chromosome 1 (hg38chr1) benchmark dataset includes 196,224 DNA sequences with 3748 (1.91%) promoter sequences. These sequences consist of multiple known promoters from EPD for the approximately 2000 known coding genes in the human chromosome 1, and also the overlapping sequences that occur from the sliding window approach. The hg38 chromosome 2 (hg38chr2) benchmark dataset includes 226,688 DNA sequences with 2632 (1.16%) promoter sequences. Notably, it is larger than hg38chr1 even though chromosome 1 contains more base pairs. The reason is the higher predominance of 'N' bases in chromosome 1 annotation. The sequences for hg38chr2 also consist of multiple known promoters from EPD for the approximately 1200 known coding genes in the human chromosome 2, and also the overlapping sequences that occur from the sliding window approach.

A fair testing procedure is incomplete without a held-out testing dataset that is completely unseen by the trained model. Since some models are trained on the complete set of human promoters, it is important to test them on a dataset where human promoters are not available to ensure no overlapping occurs within the training and testing datasets. Therefore we also include a dataset of mm10 chromosomes. As with the human benchmark dataset, the mm10 dataset contains sequences from chromosome 1 and 2 of mice. The mm10 chromosome 1 (mm10chr1) benchmark dataset includes 191,093 DNA sequences with 3275 (1.71%) promoter sequences, while the mm10 chromosome 2 (mm10chr2) benchmark dataset includes 193,183 DNA sequences with 4129 (2.13%) promoter sequences.

Model implementations

Our implementation of CNNProm follows the work by Umarov and Solovyev [14]. To reproduce the published results, we trained our implemented CNNProm model utilising sequences with a length of 251 bases. Using 251 bases, the architecture produces the same outcome as the original work from Umarov and Solovyev. This is regardless of unintentional dissimilarities within the dataset sequences due to unavailability of the original dataset. Therefore based upon the original dataset description, we composed a dataset which produced very similar results. Afterwards, to properly compare our implemented CNNProm model to other DLPR models, we used sequences with 300 bases to match the other three models which were originally designed for use with 300 bases. To create a model that encapsulates both TATA and non-TATA promoters, we decided to use the non-TATA architecture, which contains the highest amount of CNN filters between the two models. In the context of promoter recognition, CNN layers are used to find motifs within the sequence regardless of their locations within the sequence. With this architecture, we implemented the CNNProm model for this study using the same configuration as described by Umarov and Solovyev [14] and shown in Fig. 2a. In *SUPR REF*, a CNN layer for motif discovery comprises of a user-specified number of filters of length 15 and 21 for finding differently sized motifs, followed by a max-pooling operation and a rectified linear unit (ReLU) activation function. This common CNN architecture can be expanded by increasing the number of CNN layers on top of each other. Additionally, each CNN layer can make use of normalization and dropout.

Our implementation of ICNNP follows the work by Qian et al. [15]. Since the original work by Qian et al. used a limited subset of promoter sequences from EPD and did not disclose the exact sequences utilised, we randomly chose EPD promoters to match this amount. This unfortunately adds variability to our model's outcome, making them differ from the results by the original work. The input sequences contain 300 nucleotides, and the architecture uses a similar CNN layer to CNNProm containing 200 filters and a max-pool layer, with the addition of a parallel layer containing location-specific 'element' subsequences that are concatenated to the CNN layer's output to aid in the classification process. This architecture is shown in Fig. 2b. This architecture is clearly dependent on nucleotide positions within the 300 length sequence, which at a genome level will require a sliding window approach to use a very short stride. This means that the model will have to be run once for almost every base in a genome, which in the human genome would amount to more than 6 billion times, making it infeasible for use in a genome wide promoter recognition setting.

Our implementation of DProm follows the work by Oubounyt et al. [16]. The input sequences contain 300 bases, and this architecture utilises a combination of CNN and recurrent neural network (RNN) layers. This architecture is shown in Fig. 2c. The CNN layer acts as an embedding layer to locate the motifs that get passed to a bidirectional long short-term memory (LSTM) RNN layer that interprets the sequential signal of the data, namely the order in which motifs are located within the sequence.

DNABERT was not re-implemented like the previous models as its source code was readily available. We utilised DNABERT-Prom-300 model by Ji et al. [17], which was included as a fine-tuned model designed for promoter recognition from DNABERT's

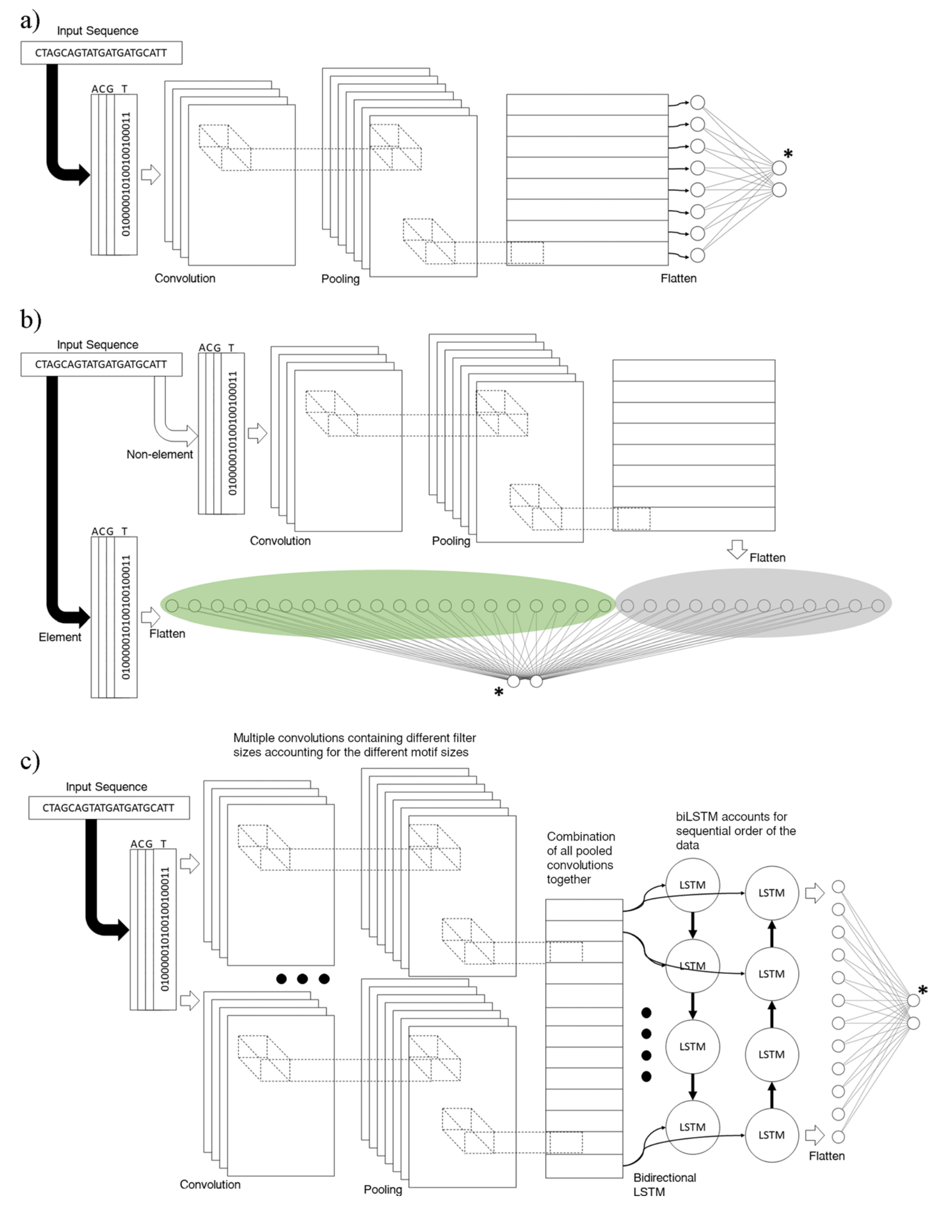

Fig. 2 Model architectures from **a** CNNProm [14], **b** ICNNP [15], **c** DProm [16]. *Two neurons depicted as output layer for softmax activation, but a single output neuron is used in case of sigmoid activation

pre-trained modelling. Further information on this model was given in DL models subsection of "Materials and methods".

Data cleaning

While implementing the models and recreating their datasets, we noticed that some promoters from EPD appeared multiple times within the same dataset. This could lead to biases in the training and testing datasets. To avoid this, we removed any duplicate sequence after the first occurrence. For example, the CNNProm non-TATA dataset contains 687 duplicate sequences, which can translate to $\sim 2-5\%$ difference in MCC values.

We further created a script to completely separate all testing and training datasets for all three implemented models for our cross-testing evaluation. To this end, we first generated separate training and testing datasets for DProm since it contains all promoter sequences. Afterwards, the training and testing datasets are further reduced to match the number of promoters needed by CNNProm and ICNNP. The non-promoter sequences are then appended to the training and testing datasets to match the complete amount of data needed to train each model. Finally, we make sure that no sequences are overlapping between training and testing datasets for all three models by comparing the rows from each dataset.

Model benchmarking

Our training and testing procedures were implemented using Skorch [33], a library adding functionality to PyTorch. The most useful feature is its ability to use training, testing, cross-validation, and evaluation metrics supported by Scikit-learn [34, 35]. This makes *SUPR REF*'s implementation easy to maintain, and training and testing procedures consistent.

The training process involves importing a *SUPR REF* created dataset and using stratified sampling to create the training batches. The models can be trained with two different output functions: sigmoid and softmax. Sigmoid outputs a single value that can separate promoters and non-promoters by a threshold. Softmax outputs two values: the probability that the input sequence is or is not a promoter. The loss function follows the output function: sigmoid utilises 'binary cross entropy loss'. Softmax utilises 'cross entropy loss'. Model training is completed once one of the following two conditions is met: 50 epochs have occurred or the performance of the model has not increased in 5 epochs. Each epoch produces a checkpoint file with the best performing model at that time. The checkpoint file contains all the model's parameters, or each neuron's weight values in the neural network architecture. The learning rate for all models was set to 0.001. Batch size and optimizer differed depending on the original model's description.

The testing process imports a dataset and loads the best performing checkpoint of a trained model. Next, the trained model is used to predict whether the sequences from the imported dataset are promoters or non-promoters. The predictions are then compared to their sequences' true classification, and evaluation metrics are calculated.

Evaluation metrics

Choosing the proper metrics to evaluate supervised machine learning models is a crucial step for testing the reliability to perform the task that the model has been trained for. Metrics also help to compare different trained models when tested on the same dataset. In the case of promoter recognition, the problem can be represented analogously to the classification of DNA sequences. Therefore, promoter recognition can be designed as a binary classification task that can be evaluated—such as in related work [14–16]—with the following metrics: Sensitivity (Sn), Specificity (Sp), Precision (PPV), and Matthews correlation coefficient (MCC).

$$Sn = \frac{TP}{TP + FN} \tag{1}$$

$$Sp = \frac{TN}{TN + FP} \tag{2}$$

$$PPV = \frac{TP}{TP + FP} \tag{3}$$

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{4}$$

Here: *TP* is the number of true positives (i.e., correctly identified promoter sequences); *TN*, true negative (correctly rejected promoter sequences); *FP*, false positive (incorrectly identified promoter sequences); *FN*, false negative (incorrectly rejected promoter sequences).

Results and discussion

From the datasets of the original studies for each DLPR model discussed earlier it is clear that apart from the architectures, non-promoter sequences are the main differentiating factors affecting their training and subsequent performance metrics. We investigate the extent that the different negative sequences affects these datasets by testing each of the previous DLPR models on human datasets from other studies. We refer to this process as *cross-testing* the models. First, we mixed the complete human dataset—including TATA and non-TATA—of each study to create a complete training dataset for each corresponding model. In essence, we implemented and trained each model on its study's set of available datasets containing sequences of length 300, including a complete dataset merging TATA and non-TATA promoters. Note that the original CNNProm models used sequences of length 250, and thus differ from our reimplemented CNNProm model which utilises sequences of length 300 for a fair comparison with the other models. Then, each model was tested on all the datasets for a proper comparison of the models.

Results of cross-testing are provided in Table 1, showing performance metrics of each model when tested on their own dataset, alongside our implemented cross validation results. Each model's results from a published study is contained within the `original study` (OS) row. Our implementation results are found under `cross-testing` (**CT** =[CT^*, CT^+, CT^-]) rows, where the models are tested on data that might not be exactly the same as the original study, thus creating the difference in results between OS rows and **CT** rows. The data difference is explained in the model implementation subsection of "Materials and methods". **CT** rows include models trained on sequences that overlap (CT^+) with sequences on the testing dataset and models trained on proper train-test split (CT^-) datasets; this shows the effect that improper splitting can have when evaluating models. Additionally, CT^* rows contain 10-fold cross-validation results from our implemented models, along a 99% confidence confidence interval. For easier visibility, CNNProm, ICNNP, and DProm have been renamed to 1, 2, and 3 respectively. From Table 1, we see that OS (1) CNNProm has been tested on all three datasets, while OS (2) ICNNP and OS (3) DProm were tested only on their own dataset. This is especially problematic as OS (2) ICNNP and OS (3) DProm compare their cross-validation results with OS (1) CNNProm's results, but it is unclear how OS (1) CNNProm was trained or tested on the other datasets. This unknown procedure makes it difficult to assess the fairness and validity of the comparison between the models.

Results from our cross validation (CT^*) did not exactly match the OS results. This is evident in the CNNProm non-TATA and complete models, as CNNProm was originally designed for sequences of length 250 and the architecture is highly tuned for that length. We observe that some OS results resemble CT^+ more closely than those from properly split datasets i.e., ICNNP model being superior to CNNProm model on the ICNNP dataset. Notably, the CNNProm non-TATA model did not obtain the same performance as in the original publication, even with overlapping and duplicate sequences when trained on sequences of length 300, which prompted us to examine the original CNNProm non-TATA dataset further. We found that the original dataset includes 686 duplicate sequences out of 47,541 sequences that can skew cross-validation, while our implemented dataset creation for CNNProm did not include duplicate sequences.

When comparing results from OS (1) CNNProm tested on CNNProm dataset and ICNNP dataset, we observe a discrepancy for the CNNProm model: CNNProm performs better on the CNNProm complete dataset, while ICNNP outperforms CNNProm on their ICNNP complete dataset. The inferior performance of CNNProm seems to be occurring because of the improperly split training and testing datasets, as the properly split dataset has very similar performance to ICNNP on MCC. This discrepancy in performance could also be due to the difference in non-promoter sequences since the promoter sequences for ICNNP dataset are a subset of the CNNProm dataset promoter sequences. Similarly, results from OS (1) CNNProm on the DProm dataset differ significantly from its performance on the CNNProm dataset. As previously mentioned with the CNNProm and ICNNP datasets, promoter sequences are very similar in both CNNProm and DProm datasets; in this case, CNNProm promoter sequences being a subset of DProm promoter sequences. Therefore, the difference in CNNProm's performance on the DProm dataset can be mostly attributed to non-promoter sequences.

The conflicting performance results from all three models when tested on datasets from different studies show that cross-validation results on small datasets are insufficient to capture a model's true performance. We infer that discrepancies in test results might stem from a limited number of training samples and the inability of the models to properly capture the many variations of promoters within the datasets in a single model. The discrepancies could also occur due to insufficient patterns to adequately represent the distribution of promoter and non-promoter sequences within real DNA sequences found in nature. As such, we used a more representative `benchmark` dataset to validate and compare the true performance of these models.

Benchmark comparison

In previous results, all three implemented models were trained and tested on a common promoter database (EPD), meaning that some promoter sequences appear in both the training and testing datasets of each other, with the exception of CT^-. This bias can skew the sensitivity and precision metrics towards more positive values. In a similar manner, some non-promoter sequences in certain training sets (CNNProm, ICNNP) are also found in the testing datasets, which will skew specificity and MCC in the same manner as previously stated. Therefore, all trained models were also tested on hg38 benchmark dataset. ICNNP and DProm models were also tested on the smaller hg38chr1&2 and mm10chr1&2 benchmark datasets. These chromosomes were selected because of their greater length compared to other chromosomes, while representing the naturally occurring ratio of ∼ 1% promoter and ∼ 99% non-promoter sequences.

The results of the models being tested on the larger, more representative human genome dataset and the smaller human and mouse chromosome datasets are shown in Table 2. Note that the precision of all three models decrease to nearly zero in the hg38 dataset. This implies that many non-promoter sequences are identified as promoters by these models, which highlights the limited capability of the models when dealing with larger sets of data that resemble the promoter to non-promoter ratio in nature. It is also important to notice that models are able to perform similarly within related species, as the models perform comparably in both human and mouse chromosomes. Finally, note how precision generally gets lower as the amount of non-promoter sequences increases, which occurs from the increasing number of false positives.

Benchmark on state-of-the-art

The latest DLPR tool, known as DNABERT [17], is currently the best performing tool for promoter recognition by utilising BERT [30], a highly specialised deep learning architecture for language processing. DNABERT requires the DNA sequences to be transformed into k-mer form, which is not easily achievable through their software. Although the algorithmic function for transforming a sequence in its k-mer equivalent is provided in their source code, it must be programmed further for a generalised setting for converting dataset files. Therefore, our software suite included this functionality as a simple command. Subsequently, we created a chromosome 1 dataset to test their tool in the specific format required by DNABERT. Details of the exact experiment sequences and parameters utilised for the testing dataset is available within the Additional file 1.

For our testing purposes, we utilised their fine-tuned pre-trained model for promoter recognition (DNABERT-Prom-300) and obtained the following results shown in Fig. 3. From these results, we can see how promoter recognition is still a coin toss where ∼ 50% of the time the result could be incorrect. Note that DNABERT was trained on human promoter data, which makes it biased towards better results in this testing dataset. Therefore, it seems that current promoter recognition tools are still not reliable enough to properly solve this problem.

Synthetic data

We further explored promoter modification into non-promoter sequences by observing the effects of the modified base distribution on the training of models. The models had DProm architecture and were trained on multiple different synthetic non-promoter datasets. In the DProm study, substitutions occurred in a uniform manner, i.e., any base had the same probability of being randomly substituted. Here, we calculated the original promoters' base distributions and made the substitutions match the original promoter distributions, i.e., the bases were shuffled instead of replaced by independent DNA subsequences. Therefore, the two types of synthetic data tested are the 'uniform distribution' (UD) and 'original promoter distribution' (OPD), respectively. When training only on TATA promoters, the models had difficulty differentiating the original promoter

Table 2 Results from implemented models when tested on multiple benchmark datasets

Model	hg38 dataset			
	4,602,408 non-promoter sequences			
	Sn	Sp	PPV	MCC
CNNProm	0.584	0.943	0.014	0.0009
ICNNP	0.474	0.944	0.012	0.0007
DProm	0.931	0.025	0.001	− 0.0001

Model	hg38chr1 dataset				hg38chr2 dataset			
	192,476 non-promoter sequences				224,056 non-promoter sequences			
	Sn	Sp	PPV	MCC	Sn	Sp	PPV	MCC
ICNNP	0.470	0.890	0.077	0.154	0.487	0.908	0.058	0.143
DProm	0.891	0.024	0.017	− 0.073	0.887	0.022	0.011	− 0.066
Model	**mm10chr1 dataset**				**mm10chr2 dataset**			
	191,094 non-promoter sequences				**193,183 non-promoter sequences**			
	Sn	**Sp**	**PPV**	**MCC**	**Sn**	**Sp**	**PPV**	**MCC**
ICNNP	0.342	0.904	0.058	0.106	0.369	0.895	0.070	0.120
DProm	0.867	0.046	0.15	− 0.053	0.856	0.050	0.019	− 0.060

Models are clearly lacking in precision (PPV)

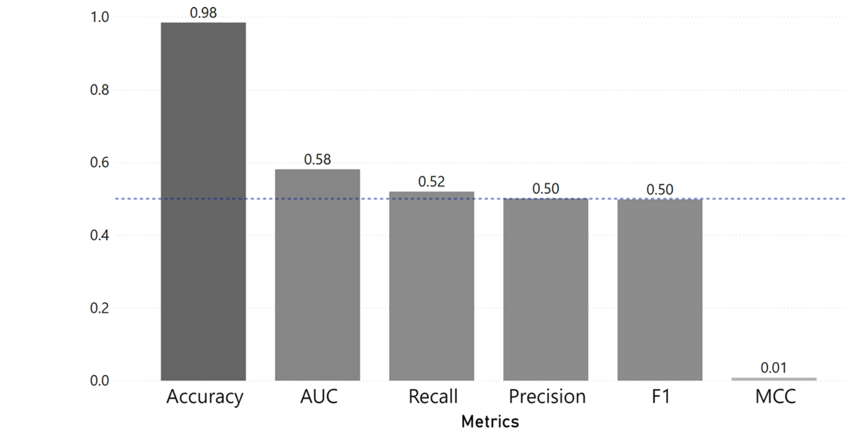

Fig. 3 Results from DNABERT using their 'Prediction' functionality for promoter recognition with a human chromosome 1 dataset created by `SUPR REF`

distribution sequences from non-promoters, leading to a lower sensitivity compared to the uniform distribution sequences. The sensitivity increased for non-TATA promoters, although uniform distribution models still outperformed the original promoter distribution. The lower sensitivity was traded for slightly higher specificity, although MCC remained generally slightly lower for OPD than UD trained models. These results can be seen in Fig. 4a. These experiments validate the use of uniformly random distribution substitutions rather than original promoter distribution substitutions as done by Oubounyt et al. [16].

While non-promoter sequences created from OPD helped models obtain slightly higher specificity than UD sequences when tested on hg38chr1, using non-promoter synthetic data to train models had disadvantages over non-synthetic data: all models trained on synthetic data had lower specificity than models trained on non-synthetic data.

Sampling methods

Common methods to compensate for highly imbalanced datasets when training models include oversampling and undersampling. Oversampling is generally done by duplicating the minority class within the training dataset, while undersampling removes samples from the majority class. Both of these methods try bringing the number of promoter and non-promoter sequences to a similar value to aid in model training. We trained a model, referred to as 'normal sampling', with the benchmark hg38chr1 which contains around 100 non-promoters to each promoter sequence in the dataset. We then trained a second model referred to as 'undersampling'. This model contains at most 10 non-promoter sequences for each promoter in the dataset by limiting the number of non-promoter sequences. Finally, we trained a third model referred to as 'oversampling', trained with a ratio of one promoter per 10 non-promoter sequences—in this case the promoter sequences were duplicated to achieve that ratio.

Comparing the results from these three sampling methods, shown in Fig. 4b, we observe that the normal sampling model could perform slightly better than the other two methods in MCC, precision and specificity. The oversampling method generally had

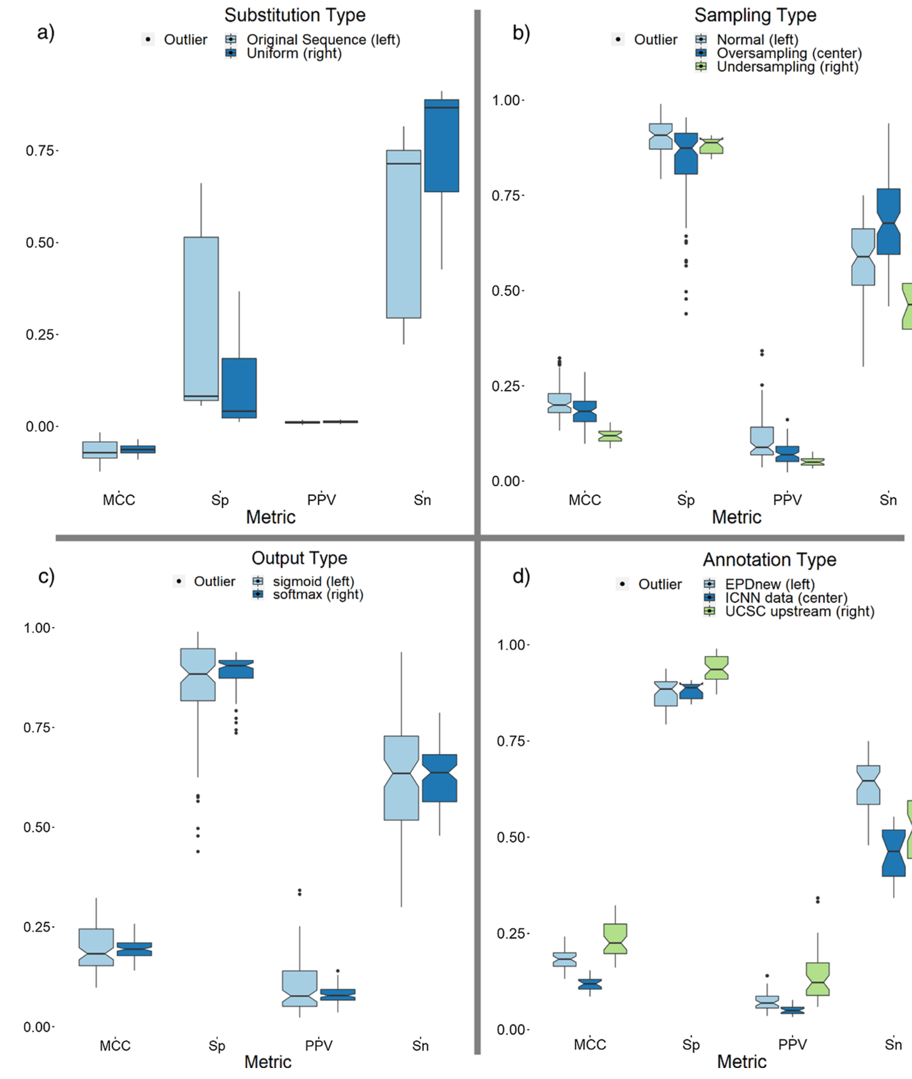

Fig. 4 Results from experiments in **a** synthetic data, **b** sampling methods, **c** output functions, **d** annotations

higher sensitivity than other sampling methods at the cost of slightly lower precision and specificity. The undersampling method seems to be the worst performing method out of all three sampling methods tested, which is the method that could be attributed to the training of most DLPR models in the literature when utilising limited data.

Output functions

We explored how the activation function in the output layer of a neural network can affect the performance of models. The most commonly used output activation functions are sigmoid and softmax. When using sigmoid, a numerical threshold is required to separate the classification between promoters and non-promoters. The threshold in our experiments for models utilising a sigmoid activation function is 0.5. This threshold is unnecessary for a softmax function since it outputs the probability of a sequence being a promoter or non-promoter. Results for this experiment is shown in Fig. 4c. Notably, the sigmoid function in most of our models tends to provide a small performance increase in precision while the softmax function tends to provide increased sensitivity, and less fluctuation within all metrics.

Annotations

Promoter annotation datasets contain promoter sequences that have been biologically verified. The EPDnew database has been utilised in most promoter recognition studies. We compare these annotations to UCSC's upstream datasets, which contain upstream sequences for every biologically verified gene in multiple organisms. These upstream sequences can be used to obtain promoter sequences since they are adjacent to the TSS of their respective genes. It is important to note that the upstream dataset cannot account for alternative TSS of genes that occur frequently in nature. Results for this experiment are shown in Fig. 4d. When tested on benchmark datasets, models trained on UCSC upstream data had higher precision than models trained on EPDnew promoters. The trade-off to the higher precision in models trained on UCSC upstream data was of lower sensitivity.

Analysis

To understand how the models could differentiate between promoter and non-promoter classes, *SUPR REF* allows us to analyse the sequences of the datasets. This helps us choose the relevant positions of the sequence to extract. These subsequences can then be used to train and test models.

Looking at the TATA dataset in Fig. 5a, we find that the TATA box motif to be a clear differentiating factor ranging from −35 to −20. We can also find minor differentiating bases between −200 and −40. Take careful consideration that motifs are located relative to the TSS (denoted as + 1), but the TSS is actually located at position 201 within the DNA sequence in both Fig. 5a, b. In the non-TATA dataset used for training CNNProm models as seen in Fig. 5b, promoter sequences tend to contain minor differentiating motifs in the regions from − 22 to − 10, and + 17 to + 31. A more pronounced differentiating region can be clearly located from − 4 to + 2. We explored whether a smaller stretch of DNA would be enough to create a model that performs similarly to the results of the CNNProm study by training a model with only the −35 to +2 subsequence. Note that this includes a significant differentiating region from −2 to +1 as well as the TATA box motif. We found that this limited model performs similarly to models utilising the complete sequence. To validate the impact that training and testing datasets can have in the evaluation of a model's performance, we further explored using only the 10 bases spanning the TATA box motif to train a model (10-bases model). The TATA box is found in the regions between − 33 to − 23, and using only these 10 bases in a cross-validation experiment, we achieved the results shown in Table 3, which are comparable to the results from OS CNNProm in Table 1. This shows us that minor differentiating regions such as − 200 to − 40 and very short but significant differentiating regions such as − 2 to + 1 did not affect the performance of this model because the much longer and highly significant TATA box region was still present within the training dataset.

Table 3 CNNProm model trained and tested on 10 bases of the human TATA dataset compared to original CNNProm TATA dataset results

Model	Sn	Sp	PPV	MCC
CNNProm 10-bases	0.95	0.95	0.91	0.89
Original CNNProm	0.95	0.98	–	0.90

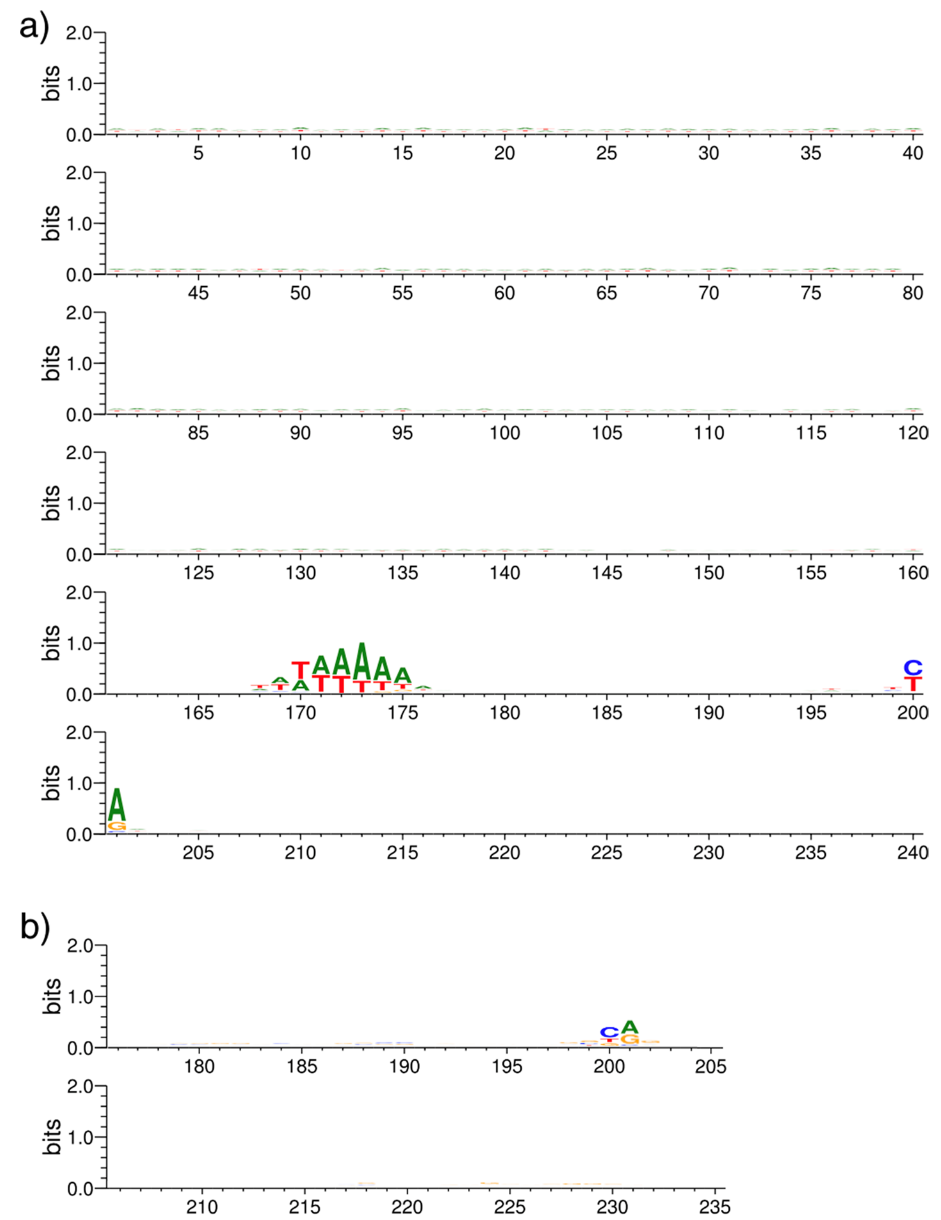

Fig. 5 Differentiating bases from promoters in **a** CNNProm TATA dataset, **b** CNNProm non-TATA dataset. Notice: tiny letters are included within the seqlogos, e.g., position 1–40 in (**a**) and 179–184 in (**b**). Clear TATA box motif located near position 170–175 in (**a**)

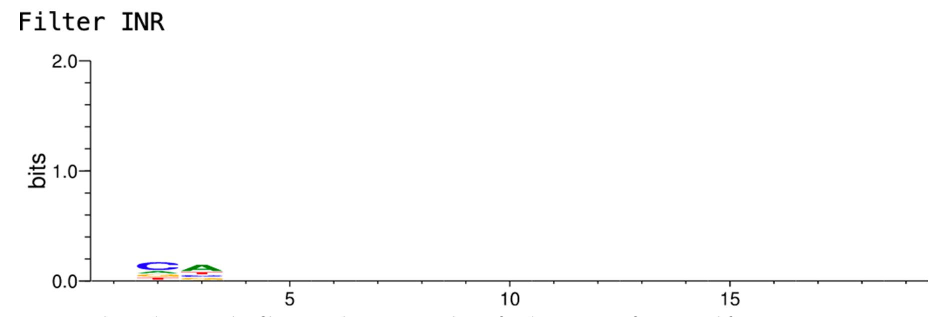

Fig. 6 Seqlogo showing the filter weights in a CNN layer for the INR motif imported from JASPAR

Analysis toolkit for trained models

Models from the studies examined here categorise sequences into being promoters or non-promoters. The output from these models do not provide a way to further explore what features the models might be using to make this decision. Therefore, we created an analysis toolkit for CNN and RNN layers within trained models that can highlight the bases and locations that carry the most weight into the final output. When visualising the CNN and RNN layers to obtain the recognition elements, we can sometimes find that the weights correlate to known motifs from databases such as JASPAR. One problem is that a single element might be distributed into multiple CNN filters or RNN hidden weights, thus making it a painful task to correlate them with known motifs. Another problem is that multiple elements might be included in a single CNN filter or RNN hidden weight, making it difficult to decipher how they were combined or even which elements were combined. Therefore, we also created a model that can import JASPAR motifs as CNN filters to analyse where they appear in promoters and non-promoters. This ensures that each element correlate exactly to only one CNN filter, avoiding the previous problems. This model was trained with the CNNProm non-TATA dataset. The results from the analysis give multiple seqlogo visuals for each filter (JASPAR PolII motif), as well as visuals for the complete sequence; a combination of all the filters in the sequence provides a global view of the bases having an impact in the final output. Figure 6 shows an example of an imported JASPAR motif, specifically the human *initator element* (INR). Notice that the filter weights seem very small because of normalisation within the neural network, which causes some bases to not appear in the seqlogo in comparison to the JASPAR seqlogo.[2] Figure 7 depicts the location where the model found INR within promoter sequences in CNNProm human non-TATA dataset. Lastly, Fig. 8 shows which bases the model is looking at using all human JASPAR motif filters combined.

Conclusions and future work

We demonstrated that the proper assessment of promoter recognition models can be heavily impacted by both the training and the testing datasets that are assumed to be a sufficient sample size of the DNA sequences observed in nature. Although the three

[2] https://jaspar.genereg.net/static/logos/all/versions/POL002.1.png.

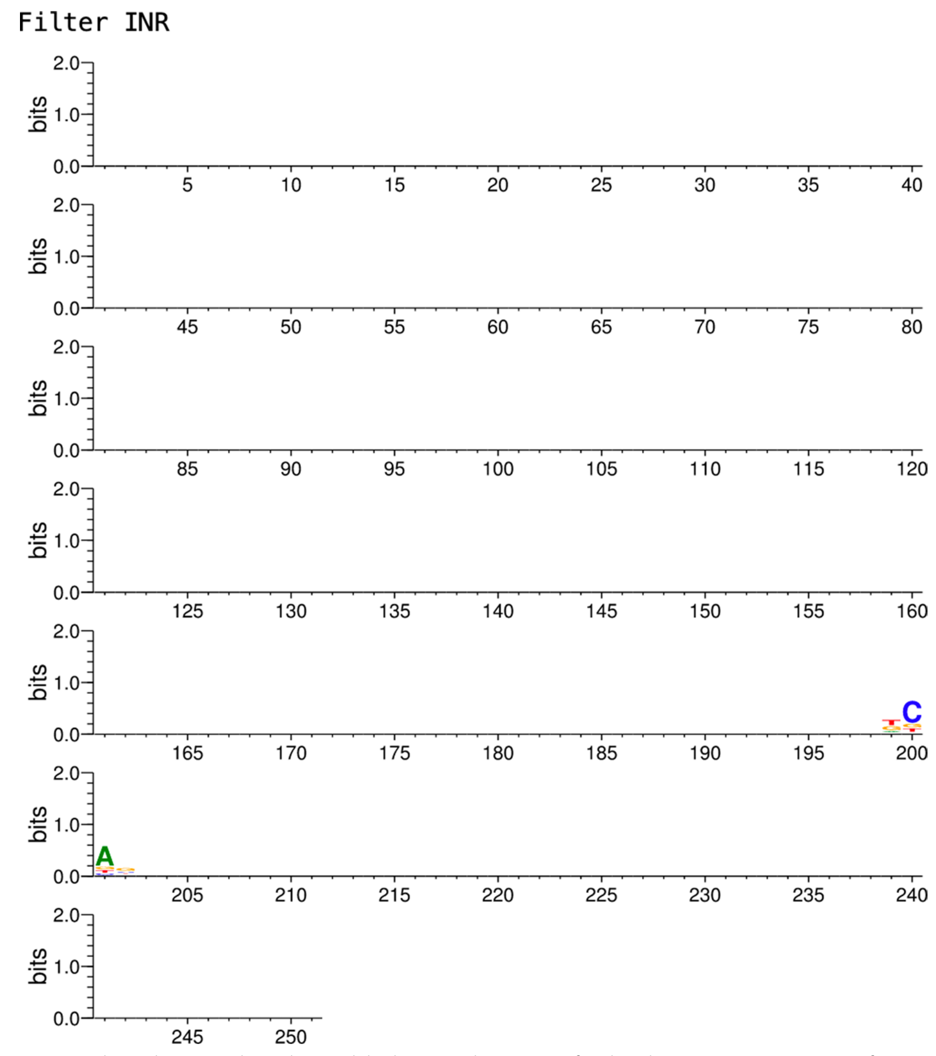

Fig. 7 Seqlogo showing where the model is locating the INR motif within the promoter sequences of CNNProm human non-TATA dataset

studies we examined utilised a similar set of promoter sequences from EPDnew, they did not use the same dataset for a proper comparison of their models. Selecting the best promoter recognition model can therefore be difficult to assess using the results from the published studies or, worse still, can lead to the wrong conclusion if choosing an under-performing model. To tackle this problem and bring clearer comparisons to assess DLPR models, we created *SUPR REF* and included the three literature model architectures, training procedures, as well as tools for benchmark dataset creation and their subsequent analysis. The dataset creation capability was also used to benchmark a state-of-the-art DLPR model to assess the capabilities of current models on promoter recognition.

We showcase *SUPR REF* by comparing previously published DLPR models in a highly imbalanced dataset (resembling ratios found at the genomic scale), alongside several experiments to explore different training methods that can increase

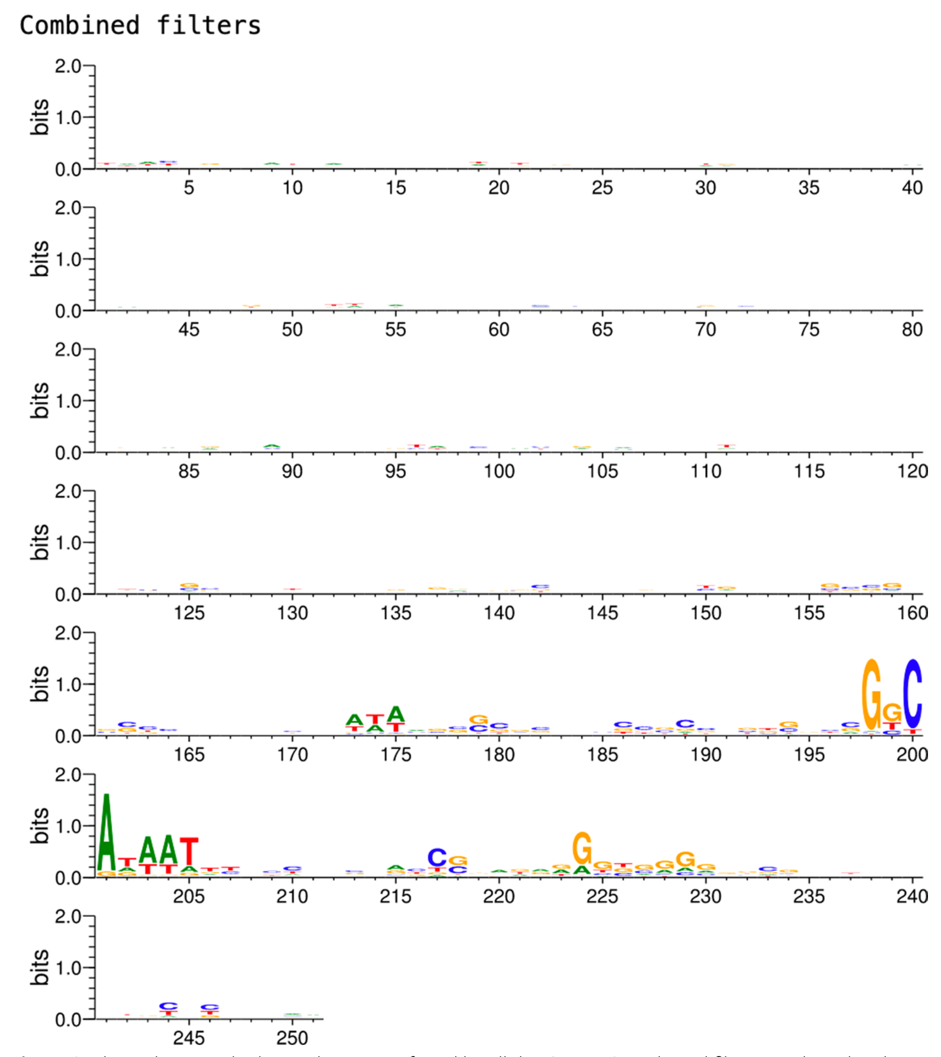

Fig. 8 Seqlogo showing the bases that are preferred by all the CNN JASPAR-based filters combined within promoter sequences of CNNProm human non-TATA dataset

performance. We provide a more specific promoter recognition analysis and data that help carry and accelerate DLPR research. Our results indicate that the true performance of ab initio promoter recognition models is still not at an acceptable level for highly imbalanced datasets, as overall performance is still low. Nevertheless, our framework can clarify the underlying models that recognise promoters within highly imbalanced data and between related species.

SUPR REF is a framework that aims to be at the forefront of reproducible research within deep learning for bioinformatics. Within the last decade, an ongoing debate within the scientific community regarding reproducibility problems has come to light [36]. This debate has also permeated computational fields, as shown by Hutson [37], where data should be more accessible and distributed. The bioinformatics field is no exception to this problem, and current technology can help mitigate this [38].

Our framework makes use of Scikit-learn, which simplifies the inclusion of state-of-the-art algorithms for training and testing models. Further improvements for benchmarking DLPR models can be achieved from other recently developed testing techniques, such as nested cross-validation by Bates et al. [39], made available by Scikit-learn and other similar libraries which *SUPR REF* used for its development.

Following our results, we hypothesise that training models with a mix of promoters from multiple related species could potentially be a mitigation strategy for the imbalance in training datasets of promoter recognition within that branch of life. This works in accordance to how models trained on mouse promoters can perform similarly on human promoters without the need of retraining the model. The inclusion of more promoter sequences can increase performance as found with the UCSC upstream annotations containing 60,555 sequences as opposed to approximately 30,000 from EPD. These ideas can be further combined by training a model on different annotations and multiple related species. In addition, increasing the dataset size of promoter sequences could be possible using non-reference genomes. Although MCC deteriorated with more data when looking only at the hg38 dataset Table 2, it is highly unlikely that an expansion to non-reference genomes would help increase performance for a single model. Perhaps these deep learning models meet a limit when trying to predict multiple promoter sequences because there is no single but multiple contradictory functions underlying.

Therefore, once the myriad of biologically functioning variants of promoter sequences are identified, they can be clustered together by similarity. Then, the sequences that are clustered together can be used to create a model that can specialise in that type of sequence or promoter. This can help the training process of DLPR models differentiate between the distribution of base pairs that are involved in promoters sequences and non-promoter sequences, thus avoiding noise that might occur in the case of IO tagging scheme. As such, promoters would not only be categorised as a single class, but as multiple classes of promoters depending on the clustering. Techniques to further improve performance of DLPR models are incrementally developed from best performing ideas and designs. It is imperative that frameworks like *SUPR REF* simplify the process of improving models by removing the need to implement previously known techniques and being hindered by implementation details. This work is a step toward this goal while we increase our understanding of promoter sequences necessary for the transcription process that occurs within cells.

Supplementary Information

The online version contains supplementary material available at https://doi.org/10.1186/s12859-022-04647-5.

Additional file 1. Additional file includes a pdf file with a step-by-step example of our testing framework utilised for DNABERT.

Acknowledgements
This work has been supported by Google Cloud research and Azure for Health. The computational results have been achieved in part using Compute Canada, Google Cloud, and Azure Clusters.

Authors' contributions
Conceptualisation: RIPM, Methodology and formal analysis: RIPM, Data acquisition and curation: RIPM, Software implementation: RIPM, Visualisation: RIPM, Writing–original draft: RIPM, Writing–review and editing: RIPM, AZ, HJ and US, Funding acquisition: HJ and US. All authors read and approved the final manuscript.

Funding
Google Cloud academic research grant, Azure Sponsorship through the Microsoft AI for Health Azure grant, University of Victoria graduate fellowship (RIPM), and NSERC Discovery Grants (HJ and US).

Availability of data and materials
Source code and documentation for the use of SUPR REF can be found at https://doi.org/10.5281/zenodo.5823112 [40] or GitHub for the latest version. Data and functionality for obtaining third party data is also available at the same location.

Declarations

Competing interests
The authors declare that they have no competing interests.

Received: 10 January 2022 Accepted: 16 March 2022
Published online: 02 April 2022

References
1. Lambert S, et al. The human transcription factors. Cell. 2018;172(4):650–65. https://doi.org/10.1016/j.cell.2018.01.029.
2. Fornes O, et al. JASPAR 2020: update of the open-access database of transcription factor binding profiles. Nucleic Acids Res. 2019;48(D1):7–92. https://doi.org/10.1093/nar/gkz1001.
3. Krzyzanowski P, et al. Regional perturbation of gene transcription is associated with intrachromosomal rearrangements and gene fusion transcripts in high grade ovarian cancer. Sci Rep. 2019. https://doi.org/10.1038/s41598-019-39878-9.
4. Rudge T, et al. Characterization of intrinsic properties of promoters. ACS Synth Biol. 2016;5(1):89–98. https://doi.org/10.1021/acssynbio.5b00116.
5. Li Y, et al. The identification of cis-regulatory elements: a review from a machine learning perspective. BioSystems. 2015;138:6–17. https://doi.org/10.1016/j.biosystems.2015.10.002.
6. Narang V, et al. Computational modeling of oligonucleotide positional densities for human promoter prediction. Artif Intell Med. 2005;35(1–2):107–19. https://doi.org/10.1016/j.artmed.2005.02.005.
7. Bajic V, et al. Performance assessment of promoter predictions on ENCODE regions in the EGASP experiment. Genome Biol. 2006;7(1):3–113. https://doi.org/10.1186/gb-2006-7-s1-s3.
8. de Medeiros OM, et al. TSSFinder–fast and accurate ab initio prediction of the core promoter in eukaryotic genomes. Brief Bioinform. 2021. https://doi.org/10.1093/bib/bbab198.
9. Zhou J, Troyanskaya OG. Predicting effects of noncoding variants with deep learning-based sequence model. Nat Methods. 2015;12(10):931–4. https://doi.org/10.1038/nmeth.3547.
10. Alipanahi B, et al. Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning. Nat Biotechnol. 2015;33(8):831–8. https://doi.org/10.1038/nbt.3300.
11. Quang D, Xie X. DanQ: a hybrid convolutional and recurrent deep neural network for quantifying the function of DNA sequences. Nucleic Acids Res. 2016;44(11):107. https://doi.org/10.1093/nar/gkw226.
12. Li J, et al. DeepATT: a hybrid category attention neural network for identifying functional effects of DNA sequences. Brief Bioinform. 2021;22(3):159. https://doi.org/10.1093/bib/bbaa159.
13. Raeder T, et al. Learning from imbalanced data: evaluation matters. Intell Syst Ref Libr. 2012. https://doi.org/10.1007/978-3-642-23166-7_12.
14. Umarov R, Solovyev V. Recognition of prokaryotic and eukaryotic promoters using convolutional deep learning neural networks. PLoS ONE. 2017;12(2):0171410. https://doi.org/10.1371/journal.pone.0171410.
15. Qian Y et al. An improved promoter recognition model using convolutional neural network. In: 2018 IEEE 42nd annual COMPSAC, 2018. IEEE, Tokyo, Japan. p. 471–476. https://doi.org/10.1109/COMPSAC.2018.00072.
16. Oubounyt M, et al. DeePromoter: robust promoter predictor using deep learning. Front Genet. 2019;10:286–286. https://doi.org/10.3389/fgene.2019.00286.
17. Ji Y, et al. DNABERT: pre-trained bidirectional encoder representations from transformers model for DNA-language in genome. Bioinformatics. 2021;37(15):2112–20. https://doi.org/10.1093/bioinformatics/btab083.
18. Chen K, et al. Selene: a PyTorch-based deep learning library for sequence data. Nat Methods. 2019;16(4):315–8. https://doi.org/10.1038/s41592-019-0360-8.
19. Kopp W, et al. Deep learning for genomics using Janggu. Nat Commun. 2020;11(1):3488. https://doi.org/10.1038/s41467-020-17155-y.
20. Budach S, Marsico A. pysster: classification of biological sequences by learning sequence and structure motifs with convolutional neural networks. Bioinformatics. 2018;34(17):3035–7. https://doi.org/10.1093/bioinformatics/bty222.
21. Avsec Z, et al. The Kipoi repository accelerates community exchange and reuse of predictive models for genomics. Nat Biotechnol. 2019;37(6):592–600. https://doi.org/10.1038/s41587-019-0140-0.
22. Shirley MD et al. Efficient "pythonic" access to FASTA files using pyfaidx. Technical Report e1196, PeerJ Inc. 2015. https://doi.org/10.7287/peerj.preprints.970v1.
23. Dréos R, et al. The eukaryotic promoter database in its 30th year: focus on non-vertebrate organisms. Nucleic Acids Res. 2017;45(D1):51–5. https://doi.org/10.1093/nar/gkw1069.
24. Haeussler M, et al. The UCSC genome browser database: 2019 update. Nucleic Acids Res. 2019;47:853–8. https://doi.org/10.1093/nar/gky1095.
25. The FANTOM Consortium et al. A promoter-level mammalian expression atlas. Nature. 2014;507(7493):462–70. https://doi.org/10.1038/nature13182.

26. Perez Martell R. Deep learning for promoter recognition: a robust testing methodology. M.Sc. Thesis, University of Victoria; 2020.
27. Reese M, et al. Genie–gene finding in *Drosophila melanogaster*. Genome Res. 2000;10(4):529–38. https://doi.org/10.1101/gr.10.4.529.
28. The FlyBase Consortium. The FlyBase database of the *Drosophila* genome projects and community literature. Nucleic Acids Res. 1999;27(1):85–8. https://doi.org/10.1093/nar/27.1.85.
29. Umarov R, et al. Promoter analysis and prediction in the human genome using sequence-based deep learning models. Bioinformatics. 2019;35(16):2730–7. https://doi.org/10.1093/bioinformatics/bty1068.
30. Devlin J, et al. BERT: pre-training of deep bidirectional transformers for language understanding; 2019. arXiv:1810.04805 [cs].
31. Dréos R, et al. MGA repository: a curated data resource for ChIP-seq and other genome annotated data. Nucleic Acids Res. 2018;46(D1):175–80. https://doi.org/10.1093/nar/gkx995.
32. Ju Y, et al. CircSLNN: identifying RBP-binding sites on circRNAs via sequence labeling neural networks. Front Genet. 2019. https://doi.org/10.3389/fgene.2019.01184.
33. Tietz M et al. Skorch: a scikit-learn compatible neural network library that wraps PyTorch. Online: skorch.readthedocs.io/en/stable/; 2017. https://skorch.readthedocs.io/en/stable/.
34. Pedregosa F, et al. Scikit-learn: machine learning in python. J Mach Learn Res. 2011;12:2825–30.
35. Buitinck L et al. API design for machine learning software: experiences from the scikit-learn project. In: ECML PKDD: languages for data mining and machine learning; 2013. p. 108–122.
36. Fidler F, Wilcox J. Reproducibility of scientific results; 2021.
37. Hutson M. Artificial intelligence faces reproducibility crisis. Science. 2018;359(6377):725–6. https://doi.org/10.1126/science.359.6377.725.
38. Kim Y-M, et al. Experimenting with reproducibility: a case study of robustness in bioinformatics. GigaScience. 2018. https://doi.org/10.1093/gigascience/giy077.
39. Bates S et al. Cross-validation: what does it estimate and how well does it do it?. 2021. arXiv:2104.00673.
40. Perez I. ivanpmartell/suprref: SUPRREF. Zenodo. 2022. https://doi.org/10.5281/zenodo.5823112; https://zenodo.org/record/5823112.

Publisher's Note