BMC Genomic Data

DATABASE

Open Access

Genomic benchmarks: a collection of datasets for genomic sequence classification

Katarína Grešová[1,2], Vlastimil Martinek[1,2], David Čechák[1,2], Petr Šimeček[1*] and Panagiotis Alexiou[1]

Abstract

Background Recently, deep neural networks have been successfully applied in many biological fields. In 2020, a deep learning model AlphaFold won the protein folding competition with predicted structures within the error tolerance of experimental methods. However, this solution to the most prominent bioinformatic challenge of the past 50 years has been possible only thanks to a carefully curated benchmark of experimentally predicted protein structures. In Genomics, we have similar challenges (annotation of genomes and identification of functional elements) but currently, we lack benchmarks similar to protein folding competition.

Results Here we present a collection of curated and easily accessible sequence classification datasets in the field of genomics. The proposed collection is based on a combination of novel datasets constructed from the mining of publicly available databases and existing datasets obtained from published articles. The collection currently contains nine datasets that focus on regulatory elements (promoters, enhancers, open chromatin region) from three model organisms: human, mouse, and roundworm. A simple convolution neural network is also included in a repository and can be used as a baseline model. Benchmarks and the baseline model are distributed as the Python package 'genomic-benchmarks', and the code is available at https://github.com/ML-Bioinfo-CEITEC/genomic_benchmarks.

Conclusions Deep learning techniques revolutionized many biological fields but mainly thanks to the carefully curated benchmarks. For the field of Genomics, we propose a collection of benchmark datasets for the classification of genomic sequences with an interface for the most commonly used deep learning libraries, implementation of the simple neural network and a training framework that can be used as a starting point for future research. The main aim of this effort is to create a repository for shared datasets that will make machine learning for genomics more comparable and reproducible while reducing the overhead of researchers who want to enter the field, leading to healthy competition and new discoveries.

Keywords Genomics, Dataset, Benchmark, Deep learning, Convolutional neural network

Background

Recently, deep neural networks have been successfully applied to identify functional elements in the genomes of humans and other organisms, such as promoters [1], enhancers [2], transcription factor binding sites [3], and others. Neural network models have been shown to be capable of predicting histone accessibility [4], RNA-protein binding [5], and accurately identify short non-coding RNA loci within the genomic background [6].

However, deep neural network models are highly dependent on large amounts of high-quality training data

*Correspondence:
Petr Šimeček
petr.simecek@ceitec.muni.cz
[1] Centre for Molecular Medicine, Central European Institute of Technology (CEITEC), Masaryk University, Brno, Czechia
[2] National Centre for Biomolecular Research, Faculty of Science, Masaryk University, Brno, Czechia

[7]. Comparing the quality of various deep learning models can be challenging, as the authors often use different datasets for evaluation, and quality metrics can be heavily influenced by data preprocessing techniques and other technical differences [8].

Many computational fields have developed established benchmarks, for example, SQuAD for question answering [9], IMDB Sentiment for text classification [10], and ImageNet for image recognition [11]. Benchmarks are crucial in driving innovation. The annual competition for object identification [12] catalyzed the boom in AI, leading in just seven years to models that exceed human capabilities.

In biology, a great challenge over the past 50 years has been *the protein folding problem*. To compare different protein folding algorithms, the community introduced the Critical Assessment of protein Structure Prediction (CASP) [13] challenge benchmark that provides research groups with the opportunity to objectively test their methods. In 2021, AlphaFold [14] won this competition producing predicted structures within the error tolerance of experimental methods. This carefully curated benchmark led to the solution of the most prominent bioinformatic challenge of the past 50 years.

In Genomics, we have similar challenges in annotation of genomes and identification and classification of functional elements, but currently we lack benchmarks similar to CASP. Practically, machine learning tasks in Genomics commonly involve the classification of genomic sequences into several categories and/or contrasting them to a genomic background (a negative set). For example, a well-studied question in Genomics is the prediction of enhancer loci on a genome. For this question, the benchmark situation is highly fragmented. As an example, [15] proposed a benchmark dataset based on the chromatin state from multiple cell lines. Both enhancer and non-enhancer sequences were retrieved from experimental chromatin information. The CD-HIT software [16] was used to filter similar sequences, and the benchmark dataset was made available as a pdf file. However, information stored in a pdf file is suitable for human communication, but computers cannot easily extract data from these files. Despite not being easily machine readable, it was used by many subsequent publications ([2, 17–26] or [27]) as a gold standard for enhancer prediction, highlighting the need for benchmark datasets in this field. Other common sources of enhancer data are the VISTA Enhancer Browser [28], the FANTOM5 [29], the ENCODE project [30], and the Roadmap Epigenomics Project [31] which provide a wealth of positive samples but no negatives. A researcher would need to implement their own method of negative selection, thus introducing individual selection biases to the samples.

Another highly studied question in Genomics is the prediction of promoters. Benchmark situation in this field has its own problems. For example, [32] extracted positive samples from EPD [33] and the non-promoter sequences were randomly extracted from coding regions and non-coding regions, and used as two negative sets. This method for creating a negative set is not an established one. Other authors used only coding sequences or only non-coding sequences as a negative set [34] or combined coding and non-coding sequences as a one negative set [35–37]. Even [32] are already pointing to the problem of missing benchmarks and reproducibility, saying that it is difficult to compare their results with other published results due to differences in data and experimental protocol. Several years later, [38] created their own dataset and reported similar problems. They were unable to compare the results with other published tools because the datasets were derived from different sources, used different proprocessing procedures, or were not made available at all.

In this paper, we propose a collection of benchmark datasets for the classification of genomic sequences, focusing on ease of use for machine learning purposes. The datasets are distributed as a Python package 'genomic-benchmarks' that is available on GitHub[1] and distributed through The Python Package Index (PyPI)[2]. The package provides an interface that allows the user to easily work with the benchmarks using Python. Included are utilities for data processing, cleaning procedures, and summary reporting. Additionally, it contains functions that make training a neural network classifier easier, such as PyTorch [39] and TensorFlow [40] data loaders and notebooks containing basic deep learning architectures that can be used as templates for prototyping new methods. Importantly, every dataset presented here comes with an associated notebook that fully reproduces the dataset generation process, to ensure transparency and reproducibility of benchmark generation in the future.

Construction and content

Overview of Datasets

The currently selected datasets are divided into three categories. There is a group of datasets focused on human regulatory functional elements, either produced from mining the Ensembl database, or from published datasets used in multiple articles. For promoters, we have imported human non-TATA promoters [41]. For enhancers, we used human enhancers from [42] paper, Ensembl human enhancers from the FANTOM5 Project [29] and

[1] https://github.com/ML-Bioinfo-CEITEC/genomic_benchmarks

[2] https://pypi.org/project/genomic-benchmarks/

Table 1 Description of datasets in genomic benchmark package. Several pieces of information are provided about each dataset: a) *Name* is unique identification of dataset in genomic benchmark package b) *# of sequences* is combined count of all sequences from all classes c) *# of classes* is count of all classes in a dataset d) *Class ratio* is a ratio between number of sequences in a biggest class and number of sequences in a smallest class e) *Median length* is computed for all sequences from all classes in a dataset f) *Standard deviation* is also computed for all sequences from all classes in a dataset

Name	# of sequences	# of classes	Class ratio	Median length	Standard deviation
dummy_mouse_enhancers_ensembl	1210	2	1.0	2381	984.4
demo_coding_vs_intergenomic_seqs	100000	2	1.0	200	0.0
demo_human_or_worm	100000	2	1.0	200	0.0
drosophila_enhancers_stark	6914	2	1.0	2142	285.5
human_enhancers_cohn	27791	2	1.0	500	0.0
human_enhancers_ensembl	154842	2	1.0	269	122.6
human_ensembl_regulatory	289061	3	1.2	401	184.3
human_nontata_promoters	36131	2	1.2	251	0.0
human_ocr_ensembl	174756	2	1.0	315	108.1

```
>>> from genomic_benchmarks.data_check import list_datasets
>>>
>>> list_datasets()
['dummy_mouse_enhancers_ensembl', 'demo_coding_vs_intergenomic_seqs',
'demo_human_or_worm', 'drosophila_enhancers_stark', 'human_enhancers_cohn',
'human_enhancers_ensembl', 'human_ensembl_regulatory', 'human_nontata_promoters',
'human_ocr_ensembl']
```

Fig. 1 Python code for listing all available dataset in the Genomic benchmarks package

drosophila enhancer [43]. We have also included open chromatin regions and multiclass datasets composed of three regulatory elements (enhancers, promoters, and open chromatin regions), both constructed from the Ensembl regulatory build [44]. The second category consists of 'demo' datasets that were computationally generated for this project, and focus on classification of genomic sequences between different species or types of transcripts (protein coding vs non-coding). Finally, the third category 'dummy' has a single small dataset which can be used for quick prototyping of methods due to its small size. From the point of view of the model organism, our datasets include primarily human data, but also mouse (*Mus musculus*), and roundworm (*Caenorhabditis elegans*) and fruit fly (*Drosophila melanogaster*). An overview of available datasets is given in Table 1 and simple code for listing all currently available datasets in Fig. 1. Additional examples of usage can be found in the project's README (dataset info, downloading the dataset, getting dataset loader), TensorFlow/PyTorch workflows in 'notebooks' folder and finally 'experiments' folder contains papermill runs for each combination of a dataset and a framework.

The *Human enhancers Cohn* dataset was adapted from [42]. Enhancers are genomic regulatory functional elements that can be bound by specific DNA binding proteins so as to regulate the transcription of a particular gene. Unlike promoters, enhancers do not need to be in a close proximity to the affected gene, and may be up to several million bases away, making their detection a difficult task.

The *Drosophila enhancers Stark* dataset was adapted from [43]. These enhancers were experimentally validated and we excluded the weak ones. Original coordinates referred to the dm3 [45] assembly of the D. melanogaster genome. We used pyliftover[3] tool to map coordinates to the dm6 assembly [46]. Negative sequences are randomly generated from drosophila genome dm6 to match lengths of positive sequences and to not overlap them.

The *Human enhancers Ensembl* dataset was constructed from Human enhancers from The FANTOM5 project [29] accessed through the Ensembl database [47]. Negative sequences have been randomly generated from the Human genome GRCh38 to match the lengths of positive sequences and not overlap them.

The *Human non-TATA promoters* dataset was adapted from [41]. These sequences are of length 251bp: from

[3] https://github.com/konstantint/pyliftover

```
>>> from genomic_benchmarks.dataset_getters.pytorch_datasets import get_dataset
>>>
>>> dset = get_dataset('human_nontata_promoters', split='train', version=0)
>>> dset[0]
('CAATCTCACAGGCTCCTGGTTGTCTACCCATGGACCCAGAGGTTCTTTGACAGCTTTGGCA...TCCAGGAGATGT', 0)
```

Fig. 2 Python code for loading dataset as a PyTorch Dataset object using `get_dataset()` function. This function takes three arguments: name of dataset, train or test split, and version of the dataset

-200 to +50bp around transcription start site (TSS). To create non-promoters sequences of length 251bp, the authors of the original paper used random fragments of human genes located after first exons.

The *Human ocr Ensembl* dataset was constructed from the Ensembl database [47]. Positive sequences are Human Open Chromatin Regions (OCRs) from The Ensembl Regulatory Build [44]. Open chromatin regions are regions of the genome that can be preferentially accessed by DNA regulatory elements because of their open chromatin structure. In the Ensembl Regulatory Build, this label is assigned to open chromatin regions, which were experimentally observed through DNase-seq, but covered by none of the other annotations (enhancer, promoter, gene, TSS, CTCF, etc.). Negative sequences were generated from the Human genome GRCh38 to match the lengths of positive sequences and not overlap them.

The *Human regulatory Ensembl* dataset was constructed from Ensembl database [47]. This dataset has three classes: enhancer, promoter and open chromatin region from The Ensembl Regulatory Build [44]. Open chromatin region sequences are the same as the positive sequences in the Human ocr Ensembl dataset.

Reproducibility

The pre-processing and data cleaning process we followed is fully reproducible. We provide a Jupyter notebook that can be used to recreate each given dataset, and can be found in the `docs` folder of the GitHub repository[4]. All dependencies are provided, and a fixed random seed is set so that the notebook will always produce the same data splits.

Each dataset is divided into training and testing subsets. For some datasets, which contain only positive samples, we had to generate appropriate negative samples (dummy mouse enhancers Ensembl, drosophila enhancers stark, human enhancers Ensembl and human open chromatin region Ensembl dataset). Negative samples were selected from the same genome as the positive samples. For each positive sample, we generated a random interval in the genome with the same length as a given sample. We picked only those intervals not overlapping with any of the positive samples.

Data format

All samples were stored as genomic coordinates, and datasets originally provided as sequences (human enhancers Cohn, human nonTATA promoters) were mapped to the reference using the 'seq2loc' tool included in the package. Data were stored as compressed (gzipped) CSV tables of genomic coordinates, containing all information typically found in a BED format table. Column names are *id*, *region*, *start*, *end*, and *strand*. Each dataset has *train* and *test* subfolders and a separate table for each class. Furthermore, each dataset contains a YAML information file with metadata such as its version, the names of included classes, and links to sequence files of the reference genome. The stored coordinates and linked sequence files were used to produce the final datasets, ensuring the reproducibility of our method. For more information, visit the `datasets` folder of the GitHub repository[5]. To speed up this conversion from a list of genomic coordinates to a locally stored folder of nucleotide sequences, we provide a cloud based cache of the full sequence datasets which can be used simply by setting the `use_cloud_cache=True` option.

Utility and discussion

Easy data access tools

Python package with the data is installed using one command line command: `pip install genomic-benchmarks`. The installed package contains ready-to-use data loaders for the two most commonly used deep learning frameworks, TensorFlow and PyTorch. This feature is important for reproducibility and for the adoption of the package, particularly by people with limited knowledge of genomics. Data loaders allow the user to load any of the provided datasets using single line of code. Full examples including imports and accessing one sample of the data are shown in Figs. 2 and 3 for PyTorch and TensorFlow respectively. However, our data are not

[4] https://github.com/ML-Bioinfo-CEITEC/genomic_benchmarks/tree/main/docs

[5] https://github.com/ML-Bioinfo-CEITEC/genomic_benchmarks/tree/main/datasets

```
>>> import tensorflow as tf
>>> from genomic_benchmarks.loc2seq import download_dataset
>>>
>>> seq_path = download_dataset("human_nontata_promoters", version=0)
Downloading 1VdUg0Zu into /home/user/.genomic_benchmarks/human_nontata_promoters.zip
... Done.
Unzipping...Done.
>>>
>>> BATCH_SIZE = 64
>>> CLASSES = ['negative', 'positive']
>>>
>>> train_dset = tf.keras.preprocessing.text_dataset_from_directory(
...     directory=seq_path/'train'
...     batch_size=BATCH_SIZE,
...     class_names=CLASSES)
Found 27097 files belonging to 2 classes.
>>>
>>> list(train_dset)[0][0][0]
<tf.Tensor: shape=(), dtype=string, numpy=b'TCCTGCCTTTCCACTTGCACCAGT...TGCTGCGGGCGG'>
```

Fig. 3 Python code for loading the dataset as TensorFlow Dataset object. First, we download dataset to our local machine and then we use TensorFlow function `text_dataset_from_directory()` to create a Dataset object

```
>>> from genomic_benchmarks.loc2seq import download_dataset
>>> from pathlib import Path
>>>
>>> seq_path = download_dataset("human_nontata_promoters", version=0)
Downloading 1VdUg0Zu into /home/user/.genomic_benchmarks/human_nontata_promoters.zip
... Done.
Unzipping...Done.
>>>
>>> tmp_dict = {}
>>> for dset in ['train', 'test']:
...   for c in ['negative', 'positive']:
...     for f in Path(seq_path/f'{dset}/{c}/').glob('*.txt'):
...       txt = f.read_text()
...       tmp_dict[f.stem] = (dset, int(c == "positive"), txt)
>>>
>>> tmp_dict
{'7227': ('train', 0, 'TGGTCGTTAAAAAGATGCAGGCAGAAGGCA...'),
 '11037': ('train', 0, 'GCAGTGGTGAGGACCTGCATCCTGCATGTC...'),
 ...}
```

Fig. 4 Python code for downloading and acessing the dataset as a raw text files. First, we download dataset to our local machine and then we sequentialy read all files and store the samples in a dictionary. A full example can be found at https://github.com/ML-Bioinfo-CEITEC/genomic_benchmarks/blob/main/notebooks/How_To_Train_BERT_Classifier_With_HF.ipynb

bound to any particular library or a tool. We provide an interface to the two most commonly used deep learning frameworks, but data are easily accessible using even plain Python, as shown in Fig. 4. Furthermore, we made Genomic benchmarks available as Hugging Face datasets[6], expanding their acessibility.

Baseline model

On top of ready-to-use data loaders, we provide tools for training neural networks and simple convolutional neural network (CNN) architecture (adapted from [48]). Demonstrative Jupyter notebook is provided in the `notebooks` folder of the GitHub repository[7], PyTorch version is also shown in Fig. 5, and it can be used as a starting point for further research and experimentation

[6] https://huggingface.co/katarinagresova

[7] https://github.com/ML-Bioinfo-CEITEC/genomic_benchmarks/tree/main/notebooks

```
# Imports
import torch
from torch.utils.data import DataLoader
from torchtext.data.utils import get_tokenizer
from genomic_benchmarks.dataset_getters.pytorch_datasets import HumanEnhancersCohn
from genomic_benchmarks.models.torch import CNN
from genomic_benchmarks.dataset_getters.utils import coll_factory, LetterTokenizer,
                                                     build_vocab

# Data preparation
train_dset =  HumanEnhancersCohn('train', version=0)
tokenizer = get_tokenizer(LetterTokenizer())
vocabulary = build_vocab(train_dset, tokenizer, use_padding=False)
device = 'cuda' if torch.cuda.is_available() else 'cpu'
collate = coll_factory(vocabulary, tokenizer, device, pad_to_length = None)
train_loader = DataLoader(train_dset, batch_size=32, shuffle=True, collate_fn=collate)

# Model preparation
model = CNN(
    number_of_classes=2,
    vocab_size=vocabulary.__len__(),
    embedding_dim=100,
    input_len=500
).to(device)

# Model training
model.train(train_loader, epochs=5)
```

Fig. 5 Python code showing the whole process of getting the dataset, tools, model and training the CNN model on the dataset. Thanks to out package, necessary code has only few lines and is easily understandable and expandable

with genomic benchmark data. CNN is an architecture that is able to find input features without feature engineering and has a relatively small number of parameters due to weights sharing (see [49] for more). Our implementation consists of three convolutional layers with 16, 8, and 4 filters, with a kernel size of 8. The output of each convolutional layer goes through the batch normalization layer and the max-pooling layer. The output of the last set of layers is flattened and goes through two dense layers. The last layer is designed to predict probabilities that the input sample belongs to any of the given classes. The architecture of the model is shown in Fig. 6. To get a baseline estimate for researchers using these benchmarks, we fit the CNN model described above to each dataset included in our collection. Training notebooks are provided in an `experiments` folder of the GitHub repository[8]. The models were trained for 10 epochs with batch size 64. The accuracy and F1 score for PyTorch and Tensorflow CNN models on all genomic benchmark datasets are shown in Table 2. In addition, we provide an example notebook how to train a DNABERT model [50] using Genomic Benchmarks[9].

Future development

We are aware of the limitations of the current repository. While we strive to include diverse data, still most of our benchmark datasets are balanced, or close to balanced, having similar length of sequences and a limited number of classes. Our main datasets all come from the human genome, and all deal with regulatory features. In the future, we would like to increase the diversity of our datasets to be able to diagnose the model's sensitivity to those factors. Many machine learning tasks in Genomics consist of binary classification of a class of Genomic functional elements against a background. However, it can be beneficial to start expanding the field into multi-class classification problems, especially for functional elements that have similar characteristics to each other against the background. We will expand our benchmark collection to include more imbalanced datasets, and more multi-class datasets.

[8] https://github.com/ML-Bioinfo-CEITEC/genomic_benchmarks/tree/main/experiments

[9] https://github.com/ML-Bioinfo-CEITEC/genomic_benchmarks/blob/main/notebooks/How_To_Train_BERT_Classifier_With_HF.ipynb

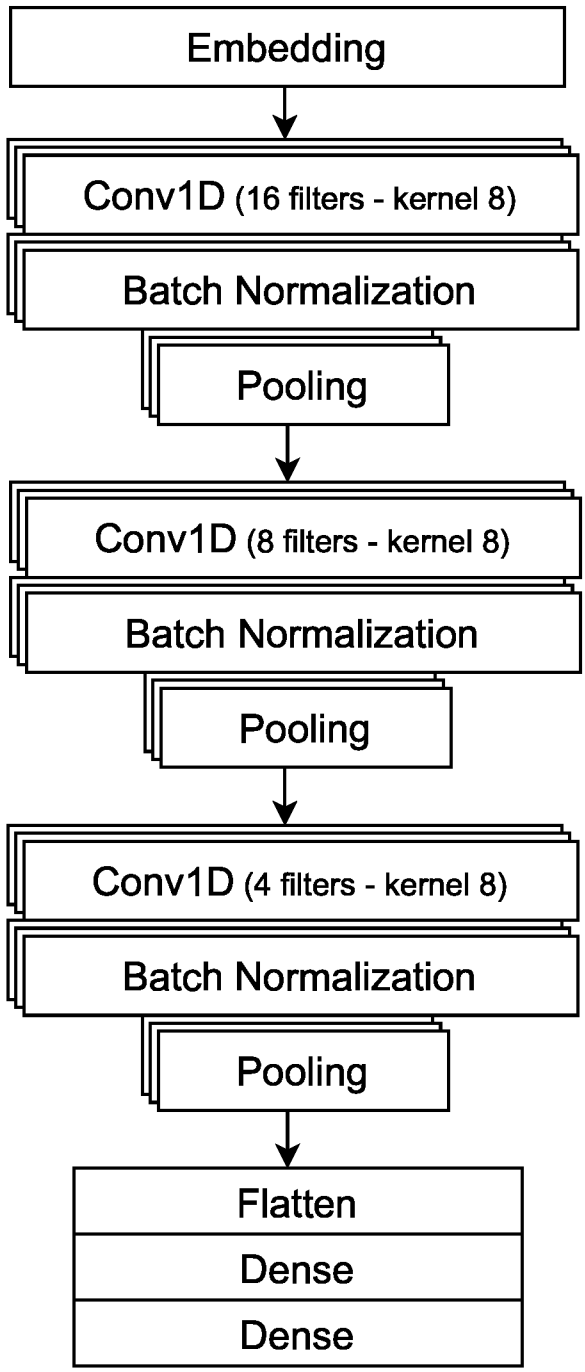

Fig. 6 CNN architecture. The neural network consists of three convolutional layers with 16, 8, and 4 filters, with a kernel size of 8. The output of each convolutional layer goes through the batch normalization layer and the max-pooling layer. The output is then flattened and passes through two dense layers. The last layer is designed to predict the probabilities that the input sample belongs to any of the given classes

Table 2 Performance of baseline models on benchmark datasets

	Pytorch		Tensorflow	
Dataset	**Accuracy**	**F1 score**	**Accuracy**	**F1 score**
dummy_mouse_enhancers_ensembl	69.0	70.4	50.0	66.9
demo_coding_vs_intergenomic_seqs	87.6	86.8	89.6	89.4
demo_human_or_worm	93.0	92.8	94.2	93.2
drosophila_enhancers_stark	58.6	44.5	52.4	69.1
human_enhancers_cohn	69.5	67.1	68.9	71.3
human_enhancers_ensembl	68.9	56.5	81.1	74.6
human_ensembl_regulatory	93.3	93.3	79.3	79.3
human_nontata_promoters	84.6	83.7	86.5	84.4
human_ocr_ensembl	68.0	66.1	68.8	72.0

Conclusions

Machine learning, especially deep learning, have recently started revolutionizing the field of genomics. Deep learning methods are highly dependent on large amounts of high-quality data to train and benchmark data are needed to accurately compare performance of different models. Here, we propose a collection of Genomic Benchmarks, produced with the aim of being easily accessible and reproducible. Our intention is to lower the difficulty of entry into the machine learning for Genomics field for researchers that may not have extensive knowledge of Genomics but want to apply their knowledge of machine learning in this field. Such an approach worked well for the field of protein folding, where benchmark-based competitions helped revolutionize the field.

The nine genomics datasets that have been currently added are a first step towards the direction of a large repository of Genomic Benchmarks. Beyond making access to these datasets easy for users, we have ensured that adding more datasets in a reproducible way is an easy task for further development of the repository. We encourage users to propose datasets or subfields of interest that would be useful in future releases. We have provided guidelines and tools to unify access to any genomic data and we will happily host submitted genomic datasets of sufficient quality and interest.

In this manuscript, we have implemented a simple convolutional neural network as a baseline model trained and evaluated on all of our datasets.

Improvement on this baseline will be certainly achieved by using different architectures and training schemes. We have an open call for users that outperform the baseline to submit their solution via our Github repository, and be added to a 'Leaderboard' of methods for each dataset. We hope that this will create a healthy competition on this set of reproducible datasets, and promote machine learning research in Genomics.

Abbreviations

CNN	Convolutional neural network
OCR	Open chromatin region
TSS	Transcription start site

Acknowledgements
We are thankful to Google Cloud for providing P. Simecek and V. Martinek free research credits. Additional computational resources were provided by the e-INFRA CZ project (ID:90140), supported by the Ministry of Education, Youth and Sports of the Czech Republic.

Author' contributions
KG did current state of the field research. KG and PS created and collected datasets. VM implemented data loaders. DC, PS and KG implemented baseline models. KG, PS and PA prepared the manuscript. All authors read and approved the final manuscript.

Funding
The work of P. Simecek was supported by the H2020 MSCA IF LanguageOfDNA (nb. 896172) and funding from Czech Science Foundation, project no. 23-04260L. The work of P. Alexiou was supported by grant H2020-WF-01-2018: 867414. The work of K. Gresova, V. Martinek, and D. Cechak was supported by EMBO Installation Grant 4431 "Deep Learning for Genomic and Transcriptomic Pattern Identification" to P. Alexiou. The funding bodies played no role in the design of the study and collection, analysis, and interpretation of data and in writing the manuscript.

Availability of data and materials
The datasets generated and/or analysed during the current study are available in the GitHub repository, https://github.com/ML-Bioinfo-CEITEC/genomic_benchmarks.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

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

Received: 18 August 2022 Accepted: 31 March 2023
Published online: 01 May 2023

References

1. Oubounyt M, Louadi Z, Tayara H, Chong KT. DeePromoter: robust promoter predictor using deep learning. Front Genet. 2019;10:286.
2. Le NQK, Ho QT, Nguyen TTD, Ou YY. A transformer architecture based on BERT and 2D convolutional neural network to identify DNA enhancers from sequence information. Brief Bioinform. 2021;22(5).
3. Quang D, Xie X. FactorNet: a deep learning framework for predicting cell type specific transcription factor binding from nucleotide-resolution sequential data. Methods. 2019;166:40–7.
4. Yin Q, Wu M, Liu Q, Lv H, Jiang R. DeepHistone: a deep learning approach to predicting histone modifications. BMC Genomics. 2019;20(2):11–23.
5. Shen Z, Zhang Q, Han K, Huang Ds. A deep learning model for RNA-protein binding preference prediction based on hierarchical LSTM and attention network. IEEE/ACM Trans Comput Biol Bioinforma. 2020;19(2):753–62.
6. Georgakilas GK, Grioni A, Liakos KG, Chalupova E, Plessas FC, Alexiou P. Multi-branch convolutional neural network for identification of small non-coding RNA genomic loci. Sci Rep. 2020;10(1):1–10.
7. Sun C, Shrivastava A, Singh S, Gupta A. Revisiting unreasonable effectiveness of data in deep learning era. In: Proceedings of the IEEE international conference on computer vision. Institute of Electrical and Electronics Engineers Inc., United States. 2017. p. 843–852.
8. Nawi NM, Atomi WH, Rehman MZ. The effect of data pre-processing on optimized training of artificial neural networks. Procedia Technol. 2013;11:32–9.
9. Rajpurkar P, Zhang J, Lopyrev K, Liang P. Squad: 100,000+ questions for machine comprehension of text. 2016. arXiv preprint arXiv:1606.05250.
10. Maas A, Daly RE, Pham PT, Huang D, Ng AY, Potts C. Learning word vectors for sentiment analysis. In: Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies. Association for Computational Linguistics, Portland, Oregon, USA. 2011. p. 142–150.
11. Deng J, Dong W, Socher R, Li LJ, Li K, Fei-Fei L, Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. IEEE. 2009. p. 248–255.
12. Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, et al. ImageNet Large Scale Visual Recognition Challenge. Int J Comput Vis. 2015;115(3):211–52. https://doi.org/10.1007/s11263-015-0816-y.
13. Moult J, Pedersen JT, Judson R, Fidelis K. A large-scale experiment to assess protein structure prediction methods. Wiley Online Library; 1995.
14. Jumper J, Evans R, Pritzel A, Green T, Figurnov M, Ronneberger O, et al. Highly accurate protein structure prediction with AlphaFold. Nature. 2021;596(7873):583–9.
15. Liu B, Fang L, Long R, Lan X, Chou KC. iEnhancer-2L: a two-layer predictor for identifying enhancers and their strength by pseudo k-tuple nucleotide composition. Bioinformatics. 2016;32(3):362–9.
16. Li W, Godzik A. Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. Bioinformatics. 2006;22(13):1658–9.
17. Liu B, Li K, Huang DS, Chou KC. iEnhancer-EL: identifying enhancers and their strength with ensemble learning approach. Bioinformatics. 2018;34(22):3835–42.
18. Le NQK, Yapp EKY, Ho QT, Nagasundaram N, Ou YY, Yeh HY. iEnhancer-5Step: identifying enhancers using hidden information of DNA sequences via Chou's 5-step rule and word embedding. Anal Biochem. 2019;571:53–61.
19. Tahir M, Hayat M, Kabir M. Sequence based predictor for discrimination of enhancer and their types by applying general form of Chou's trinucleotide composition. Comput Methods Prog Biomed. 2017;146:69–75.
20. Jia C, He W. EnhancerPred: a predictor for discovering enhancers based on the combination and selection of multiple features. Sci Rep. 2016;6(1):1–7.
21. He W, Jia C. EnhancerPred2. 0: predicting enhancers and their strength based on position-specific trinucleotide propensity and electron–ion interaction potential feature selection. Mol BioSyst. 2017;13(4):767–74.
22. Nguyen QH, Nguyen-Vo TH, Le NQK, Do TT, Rahardja S, Nguyen BP. iEnhancer-ECNN: identifying enhancers and their strength using ensembles of convolutional neural networks. BMC Genomics. 2019;20(9):1–10.
23. Khanal J, Tayara H, Chong KT. Identifying enhancers and their strength by the integration of word embedding and convolution neural network. IEEE Access. 2020;8:58369–76.
24. Zhang TH, Flores M, Huang Y. ES-ARCNN: Predicting enhancer strength by using data augmentation and residual convolutional neural network. Anal Biochem. 2021;618:114120.

25. Inayat N, Khan M, Iqbal N, Khan S, Raza M, Khan DM, et al. iEnhancer-DHF: Identification of Enhancers and Their Strengths Using Optimize Deep Neural Network With Multiple Features Extraction Methods. IEEE Access. 2021;9:40783–96.
26. Mu X, Wang Y, Duan M, Liu S, Li F, Wang X, et al. A Novel Position-Specific Encoding Algorithm (SeqPose) of Nucleotide Sequences and Its Application for Detecting Enhancers. Int J Mol Sci. 2021;22(6):3079.
27. Yang R, Wu F, Zhang C, Zhang L. iEnhancer-GAN: A Deep Learning Framework in Combination with Word Embedding and Sequence Generative Adversarial Net to Identify Enhancers and Their Strength. Int J Mol Sci. 2021;22(7):3589.
28. Visel A, Minovitsky S, Dubchak I, Pennacchio LA. VISTA Enhancer browser—a database of tissue-specific human enhancers. Nucleic Acids Res. 2007;35(suppl_1):88–92.
29. Andersson R, Gebhard C, Miguel-Escalada I, Hoof I, Bornholdt J, Boyd M, et al. An atlas of active enhancers across human cell types and tissues. Nature. 2014;507(7493):455–61.
30. ENCODE Project Consortium, et al. An integrated encyclopedia of DNA elements in the human genome. Nature. 2012;489(7414):57.
31. Kundaje A, Meuleman W, Ernst J, Bilenky M, Yen A, Heravi-Moussavi A, et al. Integrative analysis of 111 reference human epigenomes. Nature. 2015;518(7539):317–30.
32. Lin H, Li QZ. Eukaryotic and prokaryotic promoter prediction using hybrid approach. Theory Biosci. 2011;130(2):91–100.
33. Schmid CD, Perier R, Praz V, Bucher P. EPD in its twentieth year: towards complete promoter coverage of selected model organisms. Nucleic Acids Res. 2006;34(suppl_1):82–5.
34. Gordon L, Chervonenkis AY, Gammerman AJ, Shahmuradov IA, Solovyev VV. Sequence alignment kernel for recognition of promoter regions. Bioinformatics. 2003;19(15):1964–71.
35. Ohler U. Identification of core promoter modules in Drosophila and their application in accurate transcription start site prediction. Nucleic Acids Res. 2006;34(20):5943–50.
36. Yang JY, Zhou Y, Yu ZG, Anh V, Zhou LQ. Human Pol II promoter recognition based on primary sequences and free energy of dinucleotides. BMC Bioinformatics. 2008;9(1):1–13.
37. Rani TS, Bhavani SD, Bapi RS. Analysis of E. coli promoter recognition problem in dinucleotide feature space. Bioinformatics. 2007;23(5):582–8.
38. Lai HY, Zhang ZY, Su ZD, Su W, Ding H, Chen W, et al. iProEP: a computational predictor for predicting promoter. Mol Ther Nucleic Acids. 2019;17:337–46.
39. Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, et al. Pytorch: An imperative style, high-performance deep learning library. Adv Neural Inf Process Syst. 2019;32:8026–37.
40. Abadi M, Barham P, Chen J, Chen Z, Davis A, Dean J, et al. {TensorFlow}: A System for {Large-Scale} Machine Learning. In: 12th USENIX symposium on operating systems design and implementation (OSDI 16). USENIX Association, Savannah, GA, USA. 2016. p. 265–283.
41. Umarov RK, Solovyev VV. Recognition of prokaryotic and eukaryotic promoters using convolutional deep learning neural networks. PLoS ONE. 2017;12(2):0171410.
42. Cohn D, Zuk O, Kaplan T. Enhancer identification using transfer and adversarial deep learning of DNA sequences. BioRxiv. 2018:264200.
43. Kvon EZ, Kazmar T, Stampfel G, Yáñez-Cuna JO, Pagani M, Schernhuber K, et al. Genome-scale functional characterization of Drosophila developmental enhancers in vivo. Nature. 2014;512(7512):91–5.
44. Zerbino DR, Wilder SP, Johnson N, Juettemann T, Flicek PR. The ensembl regulatory build. Genome Biol. 2015;16(1):1–8.
45. Hoskins RA, Carlson JW, Kennedy C, Acevedo D, Evans-Holm M, Frise E, et al. Sequence finishing and mapping of Drosophila melanogaster heterochromatin. Science. 2007;316(5831):1625–8.
46. dos Santos G, Schroeder AJ, Goodman JL, Strelets VB, Crosby MA, Thurmond J, et al. FlyBase: introduction of the Drosophila melanogaster Release 6 reference genome assembly and large-scale migration of genome annotations. Nucleic Acids Res. 2015;43(D1):690–7.
47. Howe KL, Achuthan P, Allen J, Allen J, Alvarez-Jarreta J, Amode MR, et al. Ensembl 2021. Nucleic Acids Res. 2021;49(D1):884–91.
48. Klimentova E, Polacek J, Simecek P, Alexiou P. PENGUINN: Precise exploration of nuclear G-quadruplexes using interpretable neural networks. Front Genet. 2020;11:1287.
49. Albawi S, Mohammed TA, Al-Zawi S, Understanding of a convolutional neural network. In: 2017 international conference on engineering and technology (ICET). IEEE. 2017. p. 1–6.
50. Ji Y, Zhou Z, Liu H, Davuluri RV. DNABERT: pre-trained Bidirectional Encoder Representations from Transformers model for DNA-language in genome. Bioinformatics. 2021;37(15):2112–20.

Publisher's Note