Contents lists available at ScienceDirect

Computer Methods and Programs in Biomedicine

journal homepage: www.elsevier.com/locate/cmpb

Predicting gene expression levels from DNA sequences and post-transcriptional information with transformers

Vittorio Pipoli[b], Mattia Cappelli[a], Alessandro Palladini[a], Carlo Peluso[a], Marta Lovino[b,*], Elisa Ficarra[b]

[a] *Department of Control and Computer Engineering, Corso Duca degli Abruzzi, 24, Turin, Piedmont 10129 Italy*
[b] *Enzo Ferrari Engineering Department, University of Modena and Reggio Emilia, Via P. Vivarelli, 10, Modena, Emilia Romagna 41125, Italy*

A R T I C L E I N F O

Article history:
Received 28 April 2022
Revised 4 July 2022
Accepted 20 July 2022

Keywords:
Attention
DNA
Gene-expression
Prediction
Transcription-factors
Transformers

A B S T R A C T

Background and objectives: In the latest years, the prediction of gene expression levels has been crucial due to its potential applications in the clinics. In this context, Xpresso and others methods based on Convolutional Neural Networks and Transformers were firstly proposed to this aim. However, all these methods embed data with a standard one-hot encoding algorithm, resulting in impressively sparse matrices. In addition, post-transcriptional regulation processes, which are of uttermost importance in the gene expression process, are not considered in the model.

Methods: This paper presents Transformer DeepLncLoc, a novel method to predict the abundance of the mRNA (i.e., gene expression levels) by processing gene promoter sequences, managing the problem as a regression task. The model exploits a transformer-based architecture, introducing the DeepLncLoc method to perform the data embedding. Since DeepLncloc is based on word2vec algorithm, it avoids the sparse matrices problem.

Results: Post-transcriptional information related to mRNA stability and transcription factors is included in the model, leading to significantly improved performances compared to the state-of-the-art works. Transformer DeepLncLoc reached 0.76 of R^2 evaluation metric compared to 0.74 of Xpresso.

Conclusion: The Multi-Headed Attention mechanisms which characterizes the transformer methodology is suitable for modeling the interactions between DNA's locations, overcoming the recurrent models. Finally, the integration of the transcription factors data in the pipeline leads to impressive gains in predictive power.

1. Introduction

Gene expression is the process of producing a functional product from the instructions stored in the DNA. Predicting the abundance levels of these products - so, predicting the gene expression levels - is crucial for several applications, from drug discovery to pathway enrichment analysis.

Several studies proposed Machine Learning approaches to predict gene expression. This challenge can be addressed by exploiting sophisticated Deep Learning architectures on DNA reference sequences [1–4,10].

In detail, Convolutional Neural Networks (CNNs) were extensively adopted to address specific tasks, ranging from predicting tissue-specific expression from long promoter-proximal sequences (ExPecto, Zouh et al. [1]) to predicting the gene expression raw counts from CAGE and ChIP-seq experiments (Basenji, Kelley et al. [2]).

ExPecto [1] predicts tissue-specific expression from a wide regulatory region of 40-kbp promoter-proximal sequences and which genes are mutated. On the other hand, Basenji [2] is based on dilated convolutional filters that spot longer relationships in the inputs concerning standard convolutions. However, the limited receptive field of the Convolutional Networks (CNN) can not compete with the MultiHeadedAttention layer of a Transformer architecture [5], even using dilated filters. In addition, its main limitation consists in the use of cell line data that are more homogeneous than data from human tissues.

Regarding the solutions for predicting gene expression levels directly from the DNA sequence, Xpresso (Agarwal and Shendure [10]) is the most complete, accessible, and reproducible project. Xpresso's [10] architecture is based on CNNs and, the hyperpa-

* Corresponding author
E-mail address: marta.lovino@unimore.it (M. Lovino).

https://doi.org/10.1016/j.cmpb.2022.107035
0169-2607/© 2022 Elsevier B.V. All rights reserved.

rameters were optimized using a metaheuristic approach. Indeed, a small change in one of the hyperparameters can substantially decrease the performance and lower the stability and robustness of the architecture.

Cutting-edge deep architectures were then proposed bringing further improvements. The Enformer network (Avsec et al. [4]) takes steps forward compared to Basenji by introducing a Transformer architecture [5] to integrate long-range interactions in the genome.

A common limitation of all the cited models is the embedding of the input sequences. Every model uses a one-hot encoding, which leads to sparse matrices, which are not very informative.

Here, we propose some alternatives based on Word2Vec embeddings [8] and a domain-specific embedding method called DeepLncLoc [7].

In addition, we present an innovative pipeline called **Transformer DeepLncLoc**. Such a method relies on the transformer's [5] capability of modeling long-range dependencies and a task-aware embedding, overcoming the classical CNN-based solutions. As its name suggests, it is built on top of the DeepLncLoc embedding [7] and a vanilla Transformer Encoder Block [5].

Moreover, we propose two additional reference models, used as baselines for further evaluation.

The baseline architectures are:

- **LSTM DeepLncLoc** is an LSTM-based network fed with DeepLncLoc embedded data, used as a baseline for evaluating the DeepLncLoc embedding method [7].
- **DivideEtImpera** is an experimental model whose aim is to find a more stable Convolutional-based solution, which will be compared directly with Xpresso [10].

Furthermore, in this paper, transcription factors data are integrated with the model, leading to a significant improvement in gene expression prediction.

Transcription factors are proteins that regulate the transcription rate of genetic information from DNA to messenger RNA. Eukaryotic transcription factors work by binding to their target DNA site, located near their target genes, to recruit or block the transcription machinery onto the promoter region of the gene of interest. Their function relies on the ability to find their target site quickly and selectively [13,14].

The rest of the paper is structured as follows. Data refers to the data used for the training phase of the models and their main characteristics. Afterwards we present the Methods and their Results. At the end, a Discussion and Conclusion part is provided.

2. Data

The dataset is obtained from the Xpresso paper [10], and contains about 18,000 gene sequences with their expression values already processed and easily usable. Though gene expression is cell-type specific, Xpresso reformulates the gene expression prediction task to a cell-type agnostic approach. In particular, Xpresso averages the expression profiles of 56 different tissues to create its labels. This averaging operation is performed for two main reasons. Firstly, Xpresso's authors checked the pairwise Spearman correlations between all the pairs of expression profiles, concluding that they were highly correlated with an average correlation of 0.78. Secondly, cell-specific gene expression is regulated by factors located tens of thousands of basepairs far. As a consequence, the relatively short sequences used by Xpresso are not long enough to catch such dependencies. Consequently, the relatively short sequences used by Xpresso are not long enough to catch such dependencies. So, given that the labels are obtained using an averaging operation, they will often be referred to as *median expression levels*.

Xpresso's authors refers to these gene's sequences with the name of promoters, that are sequences of DNA located upstream the Transcription Start Site (TSS) [12], usually 100–1000 base pairs long, containing specific DNA sequences that provide a secure initial binding site for RNA polymerase and proteins called transcription factors recruiting RNA polymerase. Nevertheless, the actual dataset's sequences contains other DNA regions with respect to the promoters such as the neighboorod of the TSS and the codifying part.

Indeed, in Xpresso, gene sequences contain 20,000 bp for each gene (10,000 bp upstream and 10,000 downstream the TSS) and not the promoter part only. Furthermore, Xpresso performs a fine-tuning of the promoter region, identifying 7000 bp upstream and 3500 bp downstream the TSS as the best interval to predict gene expression.

Xpresso model, in addition to the gene sequences, exploits for each gene some extra information, named mRNA half-life features, to predict the gene expression levels.

The half-life of mRNAs is "the time required for degrading 50% of the existing mRNA molecules" [11]. Knowledge of the half-life of mRNA could potentially provide information about the stability of different types of mRNA.

The half-life of mRNA is challenging to be determined experimentally because an mRNA molecule is short-lived (between 3 and 8 min). However, equations describing the decay of mRNA and the growth of cells can be used to estimate the mRNA half-life. Indeed, the information collected in the Xpresso paper refer to 8 values that could explain the variability of mRNA half-lives [10], such as: *coding exon density*, $5'$ *UTR G/C content*, $3'$ *UTR G/C content, ORF G/C content*, $5'$ *UTR length*, $3'$ *UTR length, ORF length, intron length*. In molecular biology and genetics, GC-content (or guanine-cytosine content) is the percentage of nitrogenous bases in a DNA or RNA molecule that are either guanine (G) or cytosine (C) [16]. Within two years of their discovery in 1977, introns were found to affect gene expression positively. Indeed, distributions of the length and matching rate of optimally matched intron segments are consistent with sequence features of miRNA and siRNA [17]. These results indicate that the interaction between intron sequences and mRNA sequences is a kind of functional RNA-RNA interaction [17]. In molecular genetics, an open reading frame (ORF) is the part of a reading frame that can be translated. An ORF is a continuous stretch of codons that may [18] begin with a start codon (usually AUG) and ends at a stop codon (usually UAA, UAG, or UGA) [19]. In addition, an ATG codon (AUG in terms of RNA) within the ORF (not necessarily the first) may indicate where translation starts. One common use of open reading frames (ORFs) is evidence to assist in gene prediction. Long ORFs are often used, along with other evidence, to initially identify candidate protein-coding regions or functional RNA-coding regions in a DNA sequence [20]. However, the presence of an ORF does not necessarily mean that the region is always translated. For example, in a randomly generated DNA sequence with an equal percentage of each nucleotide, a stop-codon would be expected once every 21 codons [20].

2.1. Trascription factors

As previously mentioned, a Transcription Factor (TF) is a protein that controls the rate of transcription of genetic information from DNA to messenger RNA by binding to a specific DNA sequence [13,14]. Hence, such information is related to the gene expression [15].

Therefore, TFs are exploited in the proposed method and integrated into the DL architecture.

Transcription factors information was retrieved from the ENCODE Transcription Factor Targets [13] dataset offered by the Harmonizome project [29], which provides information from original

Table 1
TFs table integrated in the proposed model. The rows correspond to the 22,449 genes and the columns to the 181 transcription factors. The value in position x_{ij} is equal to 1 if gene i is targeted by transcription factor j, 0 otherwise.

	TF-1	TF-2	TF-3	....	TF-M
GENE-1	0	1	1	....	1
GENE-2	1	1	0	....	1
....	....	....	....	....	.....
GENE-N	0	0	0	....	0

datasets distilled into attribute tables that define significant associations between genes and attributes, where attributes could be genes, proteins, cell lines, transcription factors, tissues, experimental perturbations, diseases, phenotypes, or drugs, depending on the dataset. In this specific case, the transcription factors dataset used for this work provides the associations between 22,449 distinct genes and their related transcription factors target. The association is performed through many pipelines [28] described in the ENCODE project (Encyclopedia of DNA Elements) [27]. It is a genome mapping project that seeks to annotate the human genome with information about genes and elements that regulate gene transcription, such as transcription factor binding sites. So far, the ENCODE project has mapped transcription factor binding sites using ChIP-Seq [30], which then have been used to infer the target genes of transcription factors by computing the proximity of transcription factor binding sites to transcription start sites of genes. At the end of the full chain of data preprocessing steps, the number of transcription factors associated with the 22,449 genes resulted to be 181. Hence, the final result of this analysis is delivered as a dataset that associates each gene with a list of transcription factors.

In this work, a presence-absence matrix is created leveraging the ENCODE dataset, containing the transcription factor information. An example of the TF matrix is reported in Table 1, where the rows correspond to the 22,449 genes and the columns to the 181 transcription factors. The value in position x_{ij} is equal to 1 if gene i is targeted by transcription factor j, 0 otherwise.

2.2. Experimental conditions

In this work, three experimental conditions were considered to predict gene expression levels:

- using gene sequence **promoter** information only;
- using gene sequence **promoters** and **half-life features** information;
- using gene sequence **promoters, half-life features** and **transcription factors** information.

Hence, the Results section refers to the performances of Transformer DeepLncLoc, LSTM DeepLncLoc, and DivideEtImpera architectures in these three conditions.

3. Methods

In this section, the base encoding DeepLncLoc is presented. Then, the proposed method Transformer DeepLncLoc and the two baseline architectures are described.

3.1. DeepLncLoc

DeepLncLoc [7] is a domain-specific embedding used to synthesize information of long sequences of nucleotides in a compact fashion. It was initially proposed in the paper “DeepLncLoc: a deep learning framework for long non-coding RNA subcellular localization prediction based on subsequence embedding” [7].

The embedding process is defined as follows.

1. The sequences are divided in k-mer[9] of 3 to create a vocabulary. The total number of distinct 3-mer for ATCG nucleotides is 64.
   This vocabulary is then processed by the word2vec algorithm [8] which associates each different 3-mer to a vector of length embedding size. The result is an embedding matrix of shape (3-mer, embedding size), which in our specific case is 64×64. The best value of k is found via hyperparameter tuning (see Appendix F).
2. The data is cleverly reshaped to preserve the order of the sequences. Initially, the data are divided into consecutive slices of an arbitrary length L, which is a hyperparameter: the values $[50, 100, 105, 140, 150, 210]$ were evaluated on the validation set, and the best value resulted in being 210. Hence, the initial sequence of dimension $10,500$ will be embedded into a matrix of 210 slices, and each slice will be 50 elements long. Afterward, each subsequence is converted in a k-mer form, and for each triplet, the respective embedding vector is associated. We can represent each subsequence with a matrix, concatenating the vectors calculated by word2vec [8] for each triplet.
3. As proposed in the original work, the mean of these vectors is taken to represent each subsequence with a vector. Then, the vectors related to the subsequences are concatenated to obtain the embedding of the whole sequence. The final dimension of the embedding is a matrix 210×64, where 210 is the number of slices and 64 are the features for each slice.

The main advantages of the DeepLncLoc [7] embedding are to avoid the sparse matrix representation (typical of the one-hot encoding) thanks to word2vec [8] and to compress the data exploiting a domain-aware approach making use of k-mer. In addition, it allows the usage of sequence processing models that would not be possible to exploit on the raw sequences. This dimensionality reduction is crucial in order to train complex many-to-one sequence models. For instance, LSTM-based networks can employ many resources and can be time-consuming: instead, feeding the networks with a reduced feature matrix makes the training phase lighter and faster (Figs. 1 and 2).

3.2. Transformer DeepLncLoc

The Transformer [5] is one of the newest Deep Learning models, state of the art in the field of Natural Language Processing. The main pillars of this architecture are: Embedding of the tokens (word2vec), Mikolov [8], Positional encoding (sinusoidal functions) [5], MultiHeadedAttention [5]. This paper presents Transformer DeepLncLoc, a transformer-based architecture combining the DeepLncLoc embedding advantages [7] with the transformers’ [5] capability in finding complex and long-range dependencies. The transformers [5] build themselves the embedding given the sequences with a word2vec [8] approach, and then they add a positional encoding for keeping track of the position of the words. On the other hand, DeepLncLoc [7] is based on word2vec [8] too, but it is an offline procedure. Therefore, the integration of the DeepLncLoc embedding with the transformer architecture is performed using a BatchNormalization[24] layer just after the input layer to solve numerical issues. Then, the classical transformer’s Positional Encoder is applied. Subsequently, there is a Transformer Encoding Block, the One-Dimensional Global Average Pooling, the concatenation with the Half-life features and the last two dense layers to accomplish the regression task. After the hyperparameter tuning, the best optimizer is Adam, and the best loss is mean square error (MSE) [22].

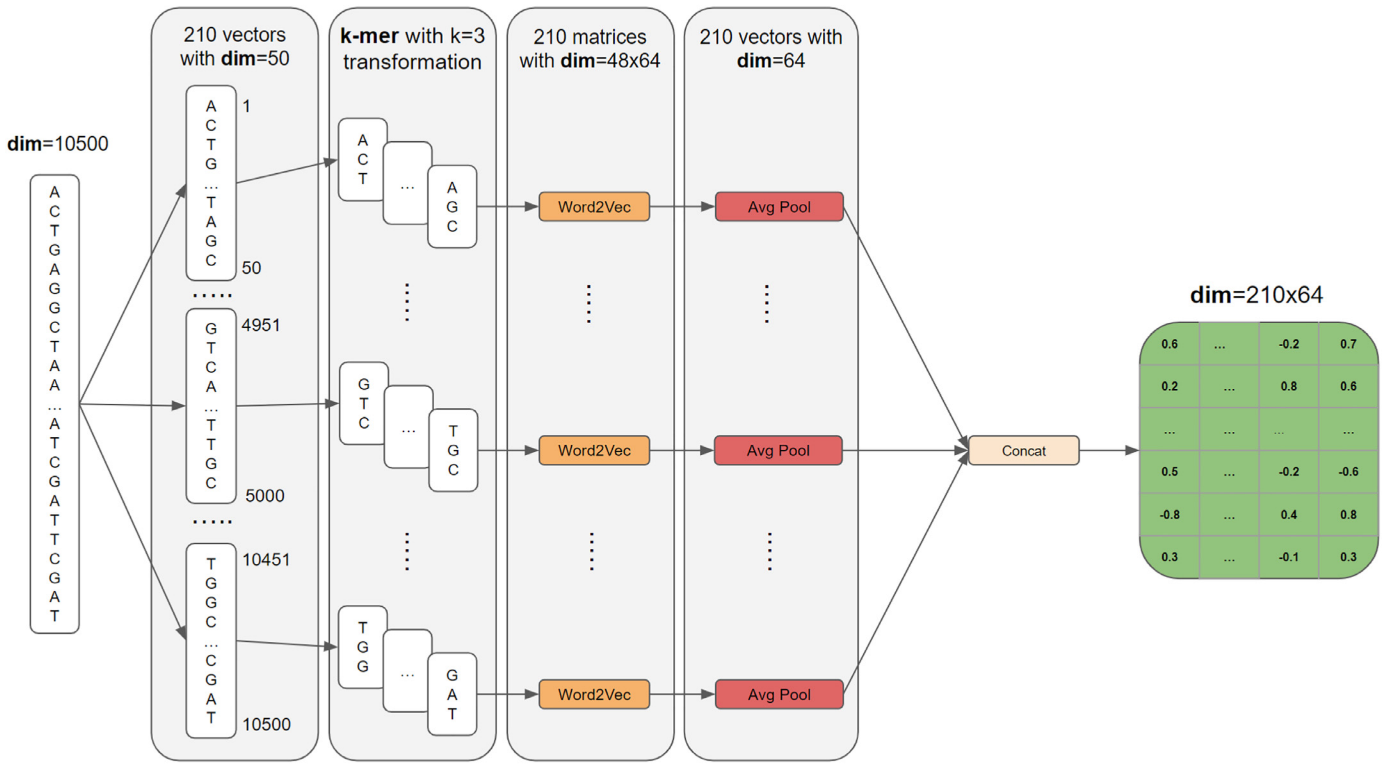


Fig. 1. DeepLncLoc methodology. In the picture it is possible to see how the sequences are processed from the sequence split into chunks of 50 base pairs each to the dense embedding representation. It is important to notice that the training phase of the Word2Vec embedder for the *k*-mer is not reported in this illustration.

3.3. LSTM DeepLncLoc

LSTM DeepLncLoc provides a baseline for the Transformer DeepLncLoc [7] architecture. It consists of a Long Short term Memory (LSTM) [6] based model fed with DeepLncLoc embedded data. Indeed, it is the simplest type of processing that can be applied to a sequence. In detail, the model is composed of a 100 units LSTM [6] feature extractor and fully connected layer in order to allow the regression task. After the hyperparameter tuning, the best optimizer is Adam, and the best loss is MSE [22].

3.4. DivideEtImpera

DivideEtImpera is based on classical Conv1D layers, devised to find a more stable convolutional solution. It receives as input the one-hot encoding version of the sequences, splits them into different chunks (inspired by DeepLncLoc embedding), and employs a deeper convolutional structure (128 filters per layer, 3 convolutional layers in total for each chunk and 4 after the concatenation) which exploits only kernels of size 3, because on average this allows extracting more complex features rather than a shallow Convolutional Network with a big filter such Xpresso [10].

The main idea behind this model is to reduce the main problem in subparts, solve them and finally recombine everything and find the solution. We divide our sequence into ten chunks (found by validation), apply five conv/pool layers to each chunk separately, concatenate all the results, and then apply five conv/pool layers again, and the final result is processed by the dense layers Section 3.4. We tried different combinations of depth in every stage of the network, ending up with 128 filters for each convolutional layer and a pool size of 5 for each MaxPooling layer. Finally, we found out that the best optimizer/loss combination is SGD [23] with MSE [22].

Concerning the additional data, half-life features, and transcription factors, they are integrated in the same way for all the models. Namely, they are concatenated after the features extraction process of the promoters. In particular, when transcription factors are involved in the pipeline, they are fed to the model by means of a boolean vector, that can be easily retrieved from the presence-absence matrix that we created.

All the methods in this work are available on the github page.

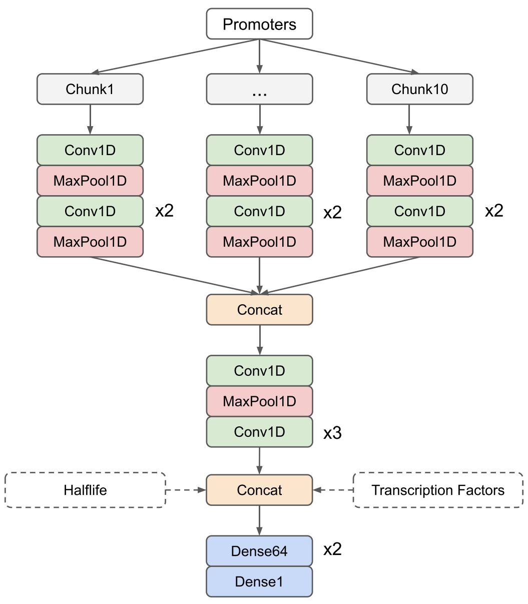
4. Results

In this section, the evaluation of the models is done in three different Experimental conditions, replicating the same evaluation setting of the *Xpresso* paper for the sake of comparison. The latter consists of using the R^2 metric for evaluation and keeping the best ten runs for each model to build confidence intervals (CIs). We clarify that for the best ten runs we consider the results of the first non-degenerate ten runs. The latter may happen, even if the odds of such events are low, as Xpresso's authors stated in their paper. Hence, very few instances have been discarded. In regression, the R^2 coefficient of determination is a statistical measure of how well the regression predictions approximate the real data points, computed as the ratio of the explained variance to the total variance [26]. By doing so, the stability of each model and the performance can be clearly stated. The R^2 evaluation metric can be defined as follows:

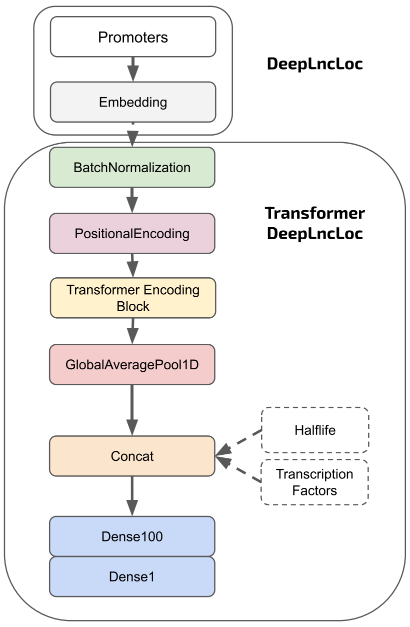
$$R^2 = 1 - \frac{\sum_{i=1}^{N}(Y_i - \hat{Y}_i)^2}{\sum_{i=1}^{N}(Y_i - \bar{Y}_i)^2}$$

where N is the number of data points, Y_i the observed values, $\hat{Y}_i$ the predicted values and $\bar{Y} = \frac{1}{n}\sum_i Y_i$.

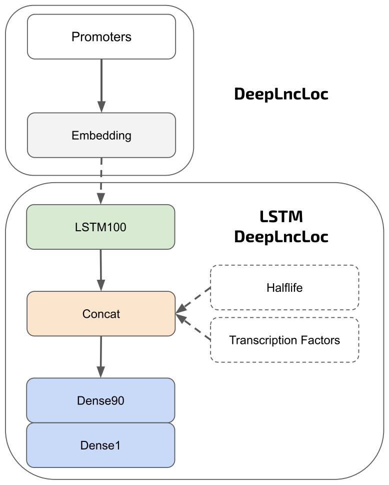
We clarify the fact that we run the *Xpresso*'s Google Colab notebook to obtain the CIs of the dataset composed by promoter and halflife features, while for the other conditions, we used an adapted version created for the sake of the project.



(a) DivideEtImpera architecture.



(b) Transformer DeepLncLoc architecture.



(c) LSTM DeepLncLoc architecture.

Fig. 2. Chart representation of the proposed methodologies. All the methods shares the same configuration of the concatenation layer with half-life and TF data. Moreover, the arrows are not continuous to highlight the fact that the integration of these data sources are optional. Additionally, the arrow between DeepLncLoc and the LSTM and Transformer methodologies is dashed because DeepLncLoc runs offline.

Table 2
95% Confidence Intervals based on R^2 scores produced from the best 10 independent trials using only promoters sequences.

Model	LB	Mean	UB
Xpresso	0.526	0.531	0.536
LSTM DeepLncLoc	0.574	0.580	0.585
DivideEtImpera	0.529	0.534	0.539
Transformer DeepLncLoc	**0.588**	**0.596**	**0.603**

Table 3
95% Confidence Intervals based on R^2 scores produced from the best 10 independent trials using promoter sequences and halflife features.

Model	LB	Mean	UB
Xpresso	0.559	0.567	0.574
LSTM DeepLncLoc	0.603	0.606	0.609
DivideEtImpera	0.580	0.582	0.583
Transformer DeepLncLoc	**0.608**	**0.610**	**0.612**

The experiments are grouped and evaluated considering the same input data.

First of all, the models are evaluated only on the promoter gene sequences, and in Table 2 can be seen the CIs.

Afterwards, the experiments shown in Table 3 are evaluated considering promoter gene sequences and the halflife features. In particular, this table represents a direct comparison between Xpresso's performances under its preferred experimental conditions against the models proposed in this paper. It is possible to notice that the CI of Xpresso's performances and the ones of the proposed methodologies never overlap, granting statistical evidence of the improvements.

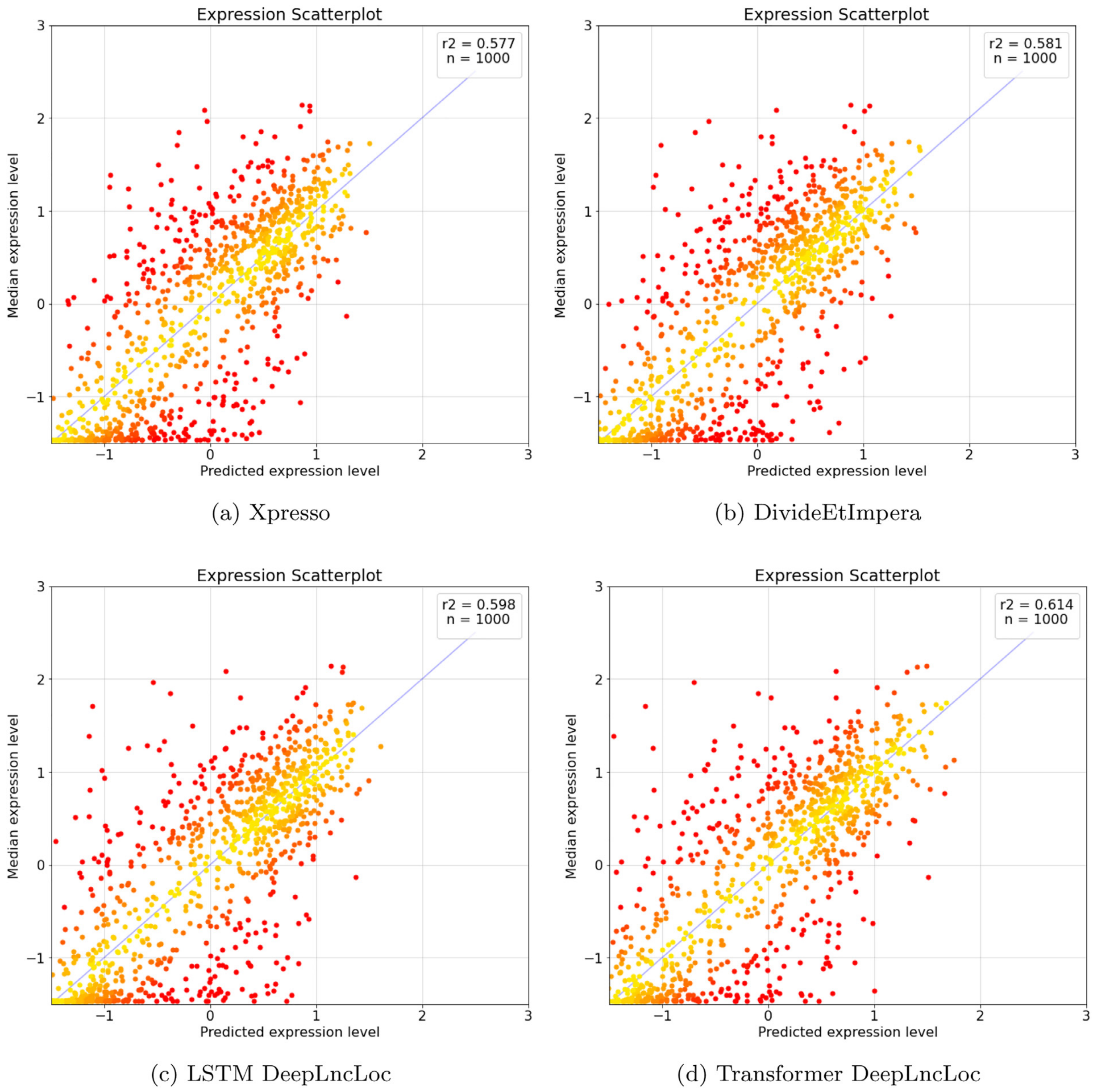


Fig. 3. Expression Scatter Plots: these graphs are devised to highlight the differences between the labels (axis-y), which are referred as *Median Expression Levels*, and the predictions (axis-x), which are referred as *Predicted Expression Levels*.

Table 4
95% Confidence Intervals based on R^2 scores produced from the best 10 independent trials using promoter sequences, halflife features and transcription factors data.

Model	LB	Mean	UB
Xpresso	0.742	0.745	0.747
LSTM DeepLncLoc	0.753	0.755	0.758
DivideEtImpera	0.757	0.759	0.760
Transformer DeepLncLoc	**0.756**	**0.760**	**0.764**

Finally, the models are evaluated considering all the data available: sequences, halflife data and transcription factors (Table 4).

Given the evident boost in performances due to the integration of transcription factors, a vanilla Multi-Layer Perceptron has been trained on transcription factors data only to assess their informative power. The evaluation's confidence interval obtained by means of ten runs with 95% level of confidence is the following: (0.662, 0.670, 0.677). See Appendix C for more details.

In Fig. 3 it's possible to have a visual evaluation of the models through the scatter plot. Given the predicted expression level on the x axes and the median expression level on the y axes, ideally, the closer the points are to the bisector, the better the model's predictive power.

5. Discussion

In this section, we discuss the results, making comparisons between the models' performances and trying to highlight their strengths and weaknesses.

Firstly, by giving a glance at the results tables, we can state that **Transformer DeepLncLoc** outperforms every model considered in every experimental condition (for more, please refer to section Experimental conditions).

This result is because Transformer DeepLncLoc architecture mixes the methods that try to solve the main limitations of the classical approaches. In fact, on the one hand, the **DeepLncLoc** [7] embedding method is capable of modeling and creating a compressed, dense, and domain-aware embedding. On the other hand, the Transformer Encoder block [5] has a great predisposition in long-range modeling dependencies. In addition, we can study and compare the results of **LSTM DeepLncLoc** and **DivideEtImpera** to understand better the effectiveness of the **DeepLncLoc** embedding. First of all, it is essential to remark that the **DeepLncLoc** embedding works offline so that you can compute the embedded version of the data one time, and then you can train the models feeding them directly with the embedded data. Consequently, if the embedding was computed in the right way, you need only an algorithm capable of finding patterns in sequences, like a Recurrent model or an Attention-based one. Indeed, implementing just a classic LSTM-based solution on top of such embedding entails reaching leading performance [6]. On the contrary, *DivideEtImpera* computes the embedding online every time that its training routine is invoked. It has to accomplish two tasks instead of one (computing the embedding and analyzing the resulting sequences), and for this reason, the likelihood of failure increases, resulting in lower performance on average. Similarly, this is the reason why **Transformer DeepLncLoc** shifts the odds of failure just on the Transformer Encoder.

From these considerations, we can conclude that the embedding of the raw sequences is a very crucial part of the process and that **DeepLncLoc** [7] offers an excellent solution to the problem. Nevertheless, a Transformer based solution [5] seems the perfect choice in order to analyze the embedded sequences, overtaking the performances of recurrent-based solutions thanks to the Multi-Headed Attention's capability to model longer relationships.

At this point, some useful considerations can be done about **DivideEtImpera**. As already stated, the logic behind the design of this model is not intended to create a competitive architecture like **Transformer DeepLncLoc**, but to understand how to create stable embedding and feature extraction exploiting solely Convolutional Layers. Indeed, its peak performances are not relevant like our LSTM or Transformer's ones. Nevertheless, it is the second performing model on the dataset integrated with Transcription Factors. Moreover, this framework is more stable and less dependent on the tuning of the hyperparameters with respect to **Xpresso** [10]. This point is achieved thanks to its peculiar deep chunking architecture, and in the end, it can be seen like the Convolutional counterpart of **DeepLncLoc** [7].

The final considerations are related to the integration of the Transcription Factors data. Thanks to the results shown in Table 4, we can state that TF additional data gives a massive boost to all the models. Moreover, it is essential to say that they achieve remarkable results also in a stand-alone evaluation, as stated in the Results section.

The main reason of the results achieved by the TF are due to their main role in the transcription regulation process. Indeed TF can either stimulate or repress transcription of the related gene affecting the gene expression levels [25].

6. Conclusion

The aim of this paper is to predict the abundance of the mRNA by processing gene promoter sequences, handling the problem as a regression task.

A deep study of the existent models like Xpresso[10] was performed in order to spot their main weak points and to devise new models capable of overcoming their performances.

The main drawback of the presented Convolutional-based solution is the embedding type, usually a one-hot encoding, and the limited receptive field typical of the Convolutional Neural Network. In this paper we used more dense and task-aware embeddings like DeepLncLoc [7] and architectures capable of modeling complex long-range dependencies like Transformers [5]. By analyzing the results, it is possible to understand that the Transformers can generalize better concerning LSTM [6], hence they can reach better results, and this is probably due to the multi-head attention layer that finds more complex patterns for LSTM [6].

From the dataset perspective, the relevant finding is related to the transcription factors capable of giving a massive boost in performance.

A possible future improvement is exhaustive hyperparameter research for models and better integration between the additional data.

Funding

This study was funded by the European Union's Horizon 2020 research and innovation programme DECIDER under Grant Agreement 965193.

Declaration of Competing Interest

Authors declare that they have no conflict of interest.

CRediT authorship contribution statement

Vittorio Pipoli: Conceptualization, Methodology, Software, Investigation, Data curation, Writing – original draft, Visualization, Validation, Writing – review & editing. **Mattia Cappelli:** Conceptualization, Methodology, Software, Investigation, Data curation, Writing – original draft, Visualization, Validation, Writing – review & editing. **Alessandro Palladini:** Conceptualization, Methodology, Software, Investigation, Data curation, Writing – original draft, Visualization, Validation, Writing – review & editing. **Carlo Peluso:** Conceptualization, Methodology, Software, Investigation, Data curation, Writing – original draft, Visualization, Validation, Writing – review & editing. **Marta Lovino:** Conceptualization, Validation, Writing – review & editing, Supervision, Project administration, Validation, Investigation. **Elisa Ficarra:** Conceptualization, Funding acquisition, Supervision, Project administration, Validation, Investigation.

Appendix A. Implementation

All this work and the data are available through the GitHub repository. The implementation has been possible thanks to Google Colaboratory. Furthermore, the project is fully coded in Python and the deep learning framework adopted for the realization of the models is TensorFlow.

Appendix B. Methods complexity

The illustrated methods in this work have a different complexity. As specified in Appendix A, the experimentations were carried out thanks to Google Colaboratory. This platform changes the available hardware, without possibility of controlling it. Therefore, it is not fair to compare runtimes among methods. However, it is useful to report the number of parameters of the used methods employing the same dataset, namely the one that includes promoters and halflife data (Table B1).

Concerning the runtime, it's impossible to define a fixed time for the experiments due to the random assignment of resources on Google Colab. However, it's possible to give an estimate of the

Table B1
Methods along with number of parameters and GPU usage, using the dataset that includes promoters and halflife.

Method	# of parameters	GPU usage [GB]
Xpresso	112,485	4.77
DivideEtImpera	1,892,289	8.79
DeepLncLoc	4096	–
LSTM DeepLncLoc	76,157	3.78
Transformer DeepLncLoc	123,881	2.42

maximum time required for each experiment which is less than half an hour for each model.

Appendix C. Statistical relevance

In statistics, the Confidence Intervals (CIs) are employed to assess the possible range around the estimate of a statistical measure of a population (e.g. mean), highlighting also how stable the estimate is. For the purposes of this work, the CI have been computed employing the t-student distribution. For each experiment, composed by ten R^2 results, the standard deviation has been computed with the unbiased estimator:

$$\sigma = \frac{\sum_{i=1}^{N}(\bar{X} - X_i)^2}{N-1}$$

where N is the number of data points, X_i the observed values and $\bar{X} = \frac{1}{n}\sum_i X_i$ the average of the observed values. Then, the lower bound (LB), upper bound (UB) and mean (Mean) values of the CI are computed as follows:

$$\text{Mean} = \bar{X} = \frac{1}{N}\sum_{i=1}^{N} X_i, \quad LB = \bar{X} - t_{\alpha/2}\frac{\sigma}{\sqrt{N}}, \quad UB = \bar{X} + t_{\alpha/2}\frac{\sigma}{\sqrt{N}}$$

where N is the number of data points, X_i the observed values, α is the significance level (in our case equal to 0.05, so that the confidence level is 0.95) and $t_{\alpha/2}$ is the t-student distribution evaluated at $\alpha/2$.

To further prove the enhancement of this work with respect to the literature, we set two different statistical tests. In both tests, the null hypothesis is that the mean of the first and second populations are equal, and then we test if the mean of the second population is significantly greater with respect to the first. The significance level of the tests is the same as the Confidence Interval, hence it is 0.05. The results are shown in Tables C1 and C2.

The first test performed is the one-sided Wilcoxon-test [21] and compared the results of Xpresso with the methods shown in this work.

The second test performed is the t-test which is applied with the same settings as the first.

Table C1
Wilcoxon test to compare the performances of the methods.

First measure	Second measure	p-value
Xpresso	Transformer DeepLncLoc	0.97×10^{-3}
Xpresso	LSTM DeepLncLoc	0.97×10^{-3}
LSTM DeepLncLoc	Transformer DeepLncLoc	0.02

Table C2
T-test to compare the performances of the methods.

First measure	Second measure	p-value
Xpresso	Transformer DeepLncLoc	1.05×10^{-10}
Xpresso	LSTM DeepLncLoc	9.68×10^{-10}
LSTM DeepLncLoc	Transformer DeepLncLoc	0.007

The p-values are all smaller than the chosen threshold which is 0.05, hence proving the evidence against the null hypothesis.

Appendix D. Prediction of mouse species' expression levels

The focus of this Manuscript is the human species but to further enrich our work the methods developed along with Xpresso are tested on another mammalian species, the mouse. The dataset used for these experiments is the same used in the Xpresso paper. It is composed like the main dataset (e.g. the humans) with promoters and halflife data. The main difference is the number of genes, that in the case of the mouse is 21,856. All the methods presented in this work are applied to this dataset, and they all use the same hyperparameters. In Table D1 all the results for the different models are summarised. The experiments confirm what was discussed in Section 6. Indeed, the relative performances of the models mirror what happened with human data, showing again that Xpresso's performances are overtaken by all the proposed models and the best performing model is still TransformerDeepLncLoc. The only difference is that all the models performed a substantial increase in the evaluation metric in absolute values.

Table D1
95% Confidence Intervals based on R^2 scores produced from the best 10 independent trials using promoter sequences and halflife features of the mouse dataset.

Model	LB	Mean	UB
Xpresso	0.692	0.700	0.708
LSTM DeepLncLoc	0.738	0.740	0.743
DivideEtImpera	0.719	0.723	0.727
Transformer DeepLncLoc	**0.754**	**0.757**	**0.761**

The experiments confirm what was discussed in Section 6. The main difference in the results of the mouse is the magnitude. Indeed the models generally perform better with this new dataset. The other important evidence is the coherence with the Xpresso results using our implementation.

Appendix E. Attention plot analysis

The analysis of the attention mechanism of the transformer could give interesting hints about the model interpretation and the patterns spotted.

The analysis of the attention head of the Transformer DeepLncLoc method has been evaluated on the first batch of the validation set in the epoch of lowest validation loss. In particular, the graph shows the average weights for the four attention heads of the transformer encoder block of the first batch. As it is possible to see in Fig. E1, the most frequent patterns are vertical patterns which mean that in the sequences there are some important DNA's locations where all the other locations pay attention. Moreover, it is important to remark that the Transcription Start Site is located around position 140. By observing the plot, it appears that the first head models have strong attention to the gene sequence (and in particular the first exon, which is mostly coding sequence), a weaker one on the promoter part of the sequences, and very low attention to the TSS. The second head, instead, is paying particular attention to the TSS. Finally, the third head is paying attention to the regions which immediately surround the TSS while the fourth pays attention to the TSS and its neighborhood. In the end, we can say that the attention mechanism highlighted the most important biological regions of the sequences.

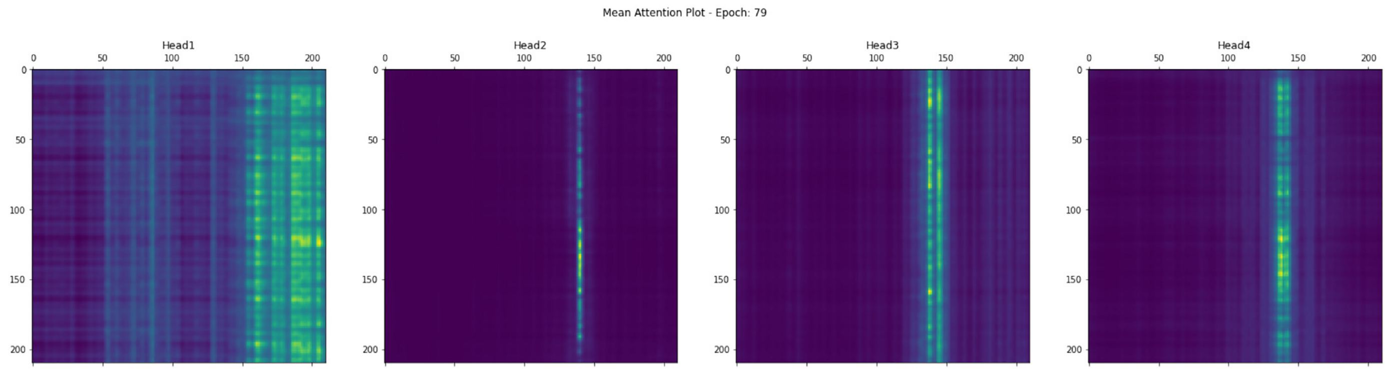


Fig. E1. Attention Plot of TransformerDeepLncLoc at the epoch of lowest validation loss, which gave an R^2 of 0.613 on the test set.

Appendix F. DeepLncLoc *k*-mer trials

The k represents the size of a window that slides on a sequence, with a stride equal to one. This window extracts sequentially substrings composed of k nucleotides. Actually, the k can be seen as a hyperparameter concerning the DeepLncLoc model. It is selected by performing DeepLncLoc in combination with the BioLSTM or the Transformer method. Finally, It is important also to underline that the experiments to choose the *k* are performed using the human dataset composed of sequences and half-life data (Table F1).

Table F1
95% Confidence Intervals based on R^2 scores for different *k* values, produced using promoter sequences and halflife features of the human dataset.

Model	LB	Mean	UB
Transformer DeepLncLoc (3-mer)	**0.608**	**0.610**	**0.612**
Transformer DeepLncLoc (6-mer)	0.499	0.502	0.505
Transformer DeepLncLoc (9-mer)	0.329	0.334	0.339
LSTM DeepLncLoc (3-mer)	**0.603**	**0.606**	**0.609**
LSTM DeepLncLoc (6-mer)	0.439	0.448	0.457
LSTM DeepLncLoc (9-mer)	0.306	0.312	0.319

References

[1] J. Zhou, C.L. Theesfeld, K. Yao, K.M. Chen, A.K. Wong, O.G. Troyanskaya, Deep learning sequence-based ab initio prediction of variant effects on expression and disease risk, Nat. Genet. 50 (8) (2018) 1171–1179, doi:10.1038/s41588-018-0160-6. Epub 2018 Jul 16. PMID: 30013180; PMCID: PMC6094955
[2] D.R. Kelley, Y.A. Reshef, M. Bileschi, D. Belanger, C.Y. McLean, J. Snoek, Sequential regulatory activity prediction across chromosomes with convolutional neural networks, Genome Res. 28 (2018) 739–750, doi:10.1101/gr.227819.117.
[3] Y. Zhang, X. Zhou, X. Cai, Predicting gene expression from DNA sequence using residual neural network,. 10.1101/2020.06.21.163956.
[4] Ž. Avsec, V. Agarwal, D. Visentin, J.R. Ledsam, A. Grabska-Barwinska, K.R. Taylor, Y. Assael, J. Jumper, P. Kohli, D.R. Kelley, Effective gene expression prediction from sequence by integrating long-range interactions, bioRxiv (2021). 10.1101/2021.04.07.438649
[5] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, N. Aidan, Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, 2017. arXiv:1706.03762
[6] S. Hochreiter, J. Schmidhuber, Long short-term memory, 1997. PubMed10.1162/neco.1997.9.8.1735.
[7] M. Zeng, Y. Wu, C. Lu, F. Zhang, F.-X. Wu, M. Li, Deeplncloc: a deep learning framework for long non-coding RNA subcellular localization prediction based on subsequence embedding, 2021. BioRxiv10.1101/2021.03.13.435245.
[8] T. Mikolov, et al., Efficient estimation of word representations in vector space, 2013.
[9] B. Chor, D. Horn, N. Goldman, Y. Levy, T. Massingham, Genomic DNA k-mer spectra: models and modalities, Genome biology 10 (10) (2009) 1–10.
[10] V. Agarwal, J. Shendure, Predicting mRNA abundance directly from genomic sequence using deep convolutional neural networks, 2018. BioRxiv 10.1101/416685v1, https://github.com/vagarwal87/Xpresso.
[11] C.Y. Chen, N. Ezzeddine, A.B. Shyu, Messenger RNA half-life measurements in mammalian cells, 2008. 10.1016/S0076-6879(08)02617-7.
[12] I. Abugessaisa, S. Noguchi, A. Hasegawa, A. Kondo, H. Kawaji, P. Carninci, T. Kasukawa, refTSS: a reference data set for human and mouse transcription start sites, J. Mol. Biol. 431 (13) (2019) 2407–2422.
[13] D.S. Latchman, Transcription factors: an overview, Int. J. Biochem. Cell Biol. 29 (12) (1997) 1305–1312. ScienceDirect- PMC- PMID
[14] M. Karin, Too many transcription factors: positive and negative interactions, New Biol. 2 (2) (1990) 126–131. PMID
[15] R. Magnusson, et al., White-box deep neural network prediction of genome-wide transcriptome signatures, TFcorrelation.
[16] Definition of GC - content on CancerWeb of Newcastle University, UK.
[17] Q. Zhang, H. Li, X.-q. Zhao, H. Xue, Y. Zheng, H. Meng, Y. Jia, S.-l. Bo, The evolution mechanism of intron length, Genomics 108 (2) (2016) 47–55, doi:10.1016/j.ygeno.2016.07.004. ISSN 0888-7543, (ScienceDirect)
[18] P. Sieber, M. Platzer, S. Schuster, The definition of open reading frame revisited", Trends Genet. 34 (3) (2018) 167–170. PMID.
[19] L.C. Brody, Stop Codon, National Human Genome Research Institute. National Institutes of Health. Retrieved 2021-08-25.
[20] J. Slonczewski, J.W. Foster, Microbiology: An Evolving Science, W.W. Norton & Co., New York, 2009.
[21] F. Wilcoxon, Individual comparisons by ranking methods, Biom. Bull. 1 (1945) 80–83, doi:10.2307/3001968.
[22] C. Sammut, G.I. Webb, Mean Squared Error - Encyclopedia of Machine Learning, Springer US, Boston (MA), 2010, doi:10.1007/978-0-387-30164-8_528. ISBN 978-0-387-30164-8
[23] H.E. Robbins, A stochastic approximation method, Ann. Math. Stat. 22 (2007) 400–407.
[24] S. Ioffe, C. Szegedy, Batch normalization: accelerating deep network training by reducing internal covariate shift, 2015.
[25] Definition of Transcription Factors - Scitable, Nature Education.
[26] R.G.D. Steel, J.H. Torrie, Principles and Procedures of Statistics with Special Reference to the Biological Sciences, McGraw Hill, 1960.
[27] The ENCyclopedia Of DNA Elements (ENCODE) Project, ENCODE Project Consortium. 10.1126/science.1105136.
[28] Pipeline for Transcription Factor ChIP-seq.
[29] The harmonizome: a collection of processed datasets gathered to serve and mine knowledge about genes and proteins, Andrew D. Rouillard, Gregory W. Gundersen, Nicolas F. Fernandez, Zichen Wang, Caroline D. Monteiro, Michael G. McDermott, Avi Ma'ayan. 10.1093/database/baw100.
[30] RNA-seq and ChIP-seq as Complementary Approaches for Comprehension of Plant Transcriptional Regulatory Mechanism, Isiaka Ibrahim Muhammad, Sze Ling Kong, Siti Nor Akmar Abdullah and Umaiyal Munusamy, 10.3390%2Fijms21010167.