BMC Bioinformatics

RESEARCH Open Access

Deep learning architectures for prediction of nucleosome positioning from sequences data

Mattia Di Gangi[1,2], Giosuè Lo Bosco[3,4*] and Riccardo Rizzo[5]

From 5th International Work-Conference on Bioinformatics and Biomedical Engineering
Granada, Spain. 26-28 April 2017

Abstract

Background: Nucleosomes are DNA-histone complex, each wrapping about 150 pairs of double-stranded DNA. Their function is fundamental for one of the primary functions of Chromatin i.e. packing the DNA into the nucleus of the Eukaryote cells. Several biological studies have shown that the nucleosome positioning influences the regulation of cell type-specific gene activities. Moreover, computational studies have shown evidence of sequence specificity concerning the DNA fragment wrapped into nucleosomes, clearly underlined by the organization of particular DNA substrings. As the main consequence, the identification of nucleosomes on a genomic scale has been successfully performed by computational methods using a sequence features representation.

Results: In this work, we propose a deep learning model for nucleosome identification. Our model stacks convolutional layers and Long Short-term Memories to automatically extract features from short- and long-range dependencies in a sequence. Using this model we are able to avoid the feature extraction and selection steps while improving the classification performances.

Conclusions: Results computed on eleven data sets of five different organisms, from Yeast to Human, show the superiority of the proposed method with respect to the state of the art recently presented in the literature.

Keywords: Nucleosome classification, Epigenetic, Deep learning networks, Recurrent neural networks

Background

The genome of eukaryote species is embedded into the nuclei of their cells. This is due to the presence of *nucleosomes*, which are histone octamers where 150 bp of DNA are wrapped in about 1.7 turns, separated by stretches of DNA referred as *linker* sequences. Starting from this low-level organization, the nucleosomes arrange themselves through successively higher-order structures to finally form the chromosomes. Apart from this packing property, the genome-wide location of the nucleosomes is fundamental for many biological processes. Gene regulation is maybe the most relevant of these processes, established by the nucleosome property of influencing DNA protein binding. For example, several studies have shown evidence that the transcription activity in promoter regions is inversely proportional to the nucleosome lacking [1]. Other important biological problems directly related to nucleosome positioning are co-transcriptional splicing, DNA replication and DNA repair [2–5].

A more detailed characterization of the eukaryotic DNA machinery is that nucleosome positioning and high-order chromatin structures together constitute a sort of control logic of this machinery. Moreover there is special evidence that distinct DNA sequence features are associated to nucleosome presence [6, 7]. Unfortunately, this is not the unique cause that determines the phenomenon,

*Correspondence: giosue.lobosco@unipa.it
[3]Dipartimento di Matematica e Informatica, Università degli studi di Palermo, Via Archirafi, 34, 90123 Palermo, Italy
[4]Dipartimento di Scienze per l'Innovazione tecnologica, Istituto Euro-Mediterraneo di Scienza e Tecnologia, Via Michele Miraglia, 20, 90139 Palermo, Italy
Full list of author information is available at the end of the article

as demonstrated by the influence that special proteins can operate to nucleosomes [8]. Nonetheless, there is a special interest in understanding to which extent the DNA sequence specificity is responsible for nucleosome positioning. This is demonstrated by the number of computational models that have been proposed so far [9]. Some of them are based on statistical models that use as a priori information the estimated distributions of dinucleotides from in vitro and in vivo sequences [10–13]. Also biophysical inspired models have been proposed [14, 15]. Others use machine learning methodologies such as logistic regression [16], Hidden Markov Models [17] and support vector machines [18]. The main characteristic shared by all the above-mentioned methodologies is the use of manually-extracted features to represent the input sequences. Establishing such details is crucial for the effectiveness of the computational solution, and this task is usually performed by supervision of the experts and a preliminary step of feature representation.

For example, in the case of the machine learning methodology for nucleosome identification, an effective way of representing the sequences is by the so called *k-mers* representation as demonstrated by several methodologies [19–25]. This representation maps a DNA sequence into a numerical space by means of a fixed-length vectors whose components are the count of each of the substrings belonging to a finite set of words.

Among the possible machine learning methods, the recent adoption of the so called *deep learning neural networks* (DLNN) [26] has produced as result very important advancements in several arduous artificial intelligence tasks [27]. We can affirm that nowadays, DLNN represents the state of the art for most of the machine learning problems. Moreover, another special property of the DLNN models that contributes to further improve their potentiality is the so called *representation learning property* i.e. the ability of such methods to automatically extract and consider useful features from the input patterns without using any a priori knowledge about the problem domain.

The two most effective kinds of DLNN successfully adopted so far for sequential input data belong to the classes of *Convolutional Neural Networks* (CNNs) and *Recurrent Neural Networks* (RNNs). CNNs have a feed-forward scheme, and are mainly composed of an initial layer of convolutional filters whose output is processed by nonlinear activation functions followed by a sub-sampling layer. The final classification is decided by a fully connected layer [28]. Collobert et al. [29] firstly shown that CNNs can be used effectively also for sequence analysis, in the general case of text classification.

RNNs have been proposed for processing sequential data. They are characterized by an hidden state that acts like an internal memory. The memory about the past is implemented by feedback connections directed through the hidden state so that it can be updated taking into consideration what happened in the previous time steps. Due to this memory property, RNNs are able to handle sequence information over time, and this could represent a prominent property for sequence classification.

Some DLNN for sequence classification have been proposed so far, both for generic sequences [30–32] and for the case of nucleosome related ones [33, 34]. In particular, a CNN architecture that adopts the k-mers as feature representation and extraction step has been proposed for nucleosome classification[33]. Moreover, a more effective version of a DLNN which uses a convolutional layer for automatically extract the sequence features, and a recurrent layer to deal with the longer-range positional dependency between part of the sequence has been recently proposed [34].

In this work, we extend this latter contribution by slightly modifying the network architecture and enlarging the experimental part with new datasets and more evaluation setups. The extended evaluation allows a comparison of the proposed methodology for nucleosome identification against many previously proposed methodologies. The main advantage of the proposed method is that it does not need any feature extraction or selection process, a step that seems to be fundamental for all the most successful computational methods for nucleosome identification.

Methods

DLNNs represent now the state of the art approach for many machine learning applications: they are characterized by the presence of several stacked layers, each producing a higher-level representation of the data coming from the previous layer. For a classification task, the last layer is able to predict the probabilities for each class relying on a high-level representation of the data that has been learnt automatically by the network. One interesting aspect of deep learning is its capability to learn to extract features starting from input in a raw form, i.e. one-hot encoding for genome sequences, even though it's possible to add external knowledge from which the network can benefit. In this work, we do not take in consideration the latter option. In the deep learning literature, we can find various kinds of layers that can be applicable to different types of data and can be chosen according to the type of features we expect can be useful for the task. For instance, convolutional layers can learn local relations in a sequence or an image, recurrent layers can learn features useful for sequences of data and the more classical feed-forward layers can learn global features from the samples.

A network architecture is given by the number, the kind and the sequence of the layers, together with the parameter values.

In the following subsections we will introduce the basic layers used to build our neural model and will discuss the architecture used for the network, finally, we will describe the datasets used for the experiments.

Convolutional layer

In general, a 1-D convolutional layer processes an input x in the form of numerical matrix of size $m \times R$, giving as output a matrix h of size $L \times R$, where L is the number of output feature maps. A feature map h is the result of the convolution of the data and a kernel w^l of length k, to which a nonlinear function g is applied:

$$q_i^l = \sum_{u=0}^{k} \mathbf{w}^l(u)\mathbf{x}(i-u) \qquad (1)$$

$$h_i^l = g\left(q_i^l + b^l\right) \qquad (2)$$

In the formulas above, k is a parameter of the convolution and represents the number of adjacent string positions taken into account during the convolution. The choice of k determines the kind of features that can be learned by the network, as well as the computational speed. A higher value of k means that a larger area is covered each time, but requires more computations to be performed. Common values of k range from 1 to 7. We choose to use filters of size not greater than 5 because the output of the convolutional layer can be seen as continuous vectors for k-mers, and from the previous study seems that k-mers longer than 5 do not help in this classification task.

In many cases, a convolutional layer is followed by a pooling layer that is used to subsample and regularize the input, in order to speed up the computation process and to prevent overfitting. The *max-pooling* acts as a nonlinear down-sampling by partitioning the input vector into a set of non-overlapping regions and, for each sub-region, their maximum value is chosen as output. This processing layer reduces the complexity of the following layers and operates a sort of translational invariance. This characteristic can be useful in the case of DNA sequences, in particular when the presence of certain DNA substring is more relevant than their positioning.

Long short-term memory layer

A recurrent neural network (RNN) is a processing unit capable to process sequences of data. It stores a so-called *hidden state* that is updated after the computation of every time step. The new hidden state is a function of the old hidden state and the input at the current time step:

$$\mathbf{h_t} = g\left(\mathbf{W_h x_t} + \mathbf{U_h h_{t-1}} + \mathbf{b_h}\right)$$

where $\mathbf{W_h}$ and $\mathbf{U_h}$ are respectively the weight matrices for the input and the hidden state, b_h is a bias vector, and g is a nonlinear activation.

For each time step t, the hidden state h_t contains a summary of the sequence seen until time step t, thus the last hidden state h_T should contain a summary of the sentence. Unfortunately, as $\mathbf{W_h}$ and $\mathbf{U_h}$ are fixed during the processing of a sequence, the importance of each input decays exponentially with the distance from the end. Another problem that limits the learning capability of this kind of layer is the *vanishing gradient problem* [35], which greatly reduces the number of time steps that are actually involved in the learning. A *Long Short-Term Memory* (LSTM) layer [36] is a variant of a recurrent layer explicitly designed to alleviate the vanishing gradient problem , and selecting the inputs that are relevant for updating the hidden state. The LSTM can be described by the following equations

$$\begin{aligned}
\mathbf{f_t} &= \sigma\left(\mathbf{W_f x_t} + \mathbf{U_f h_{t-1}} + \mathbf{b_f}\right)\\
\mathbf{i_t} &= \sigma\left(\mathbf{W_i x_t} + \mathbf{U_i h_{t-1}} + \mathbf{b_i}\right)\\
\mathbf{o_t} &= \sigma\left(\mathbf{W_o x_t} + \mathbf{U_o h_{t-1}} + \mathbf{b_o}\right)\\
\mathbf{c_t} &= \tanh\left(\mathbf{W_c x_t} + \mathbf{U_c h_{t-1}} + \mathbf{b_c}\right)\\
\mathbf{s_t} &= \mathbf{f_t} \odot \mathbf{s_{t-1}} + \mathbf{i_t} \odot \mathbf{c_t}\\
\mathbf{h_t} &= \tanh\left(\mathbf{s_t}\right) \odot \mathbf{o_t}
\end{aligned}$$

where $\mathbf{i_t}, \mathbf{f_t}, \mathbf{o_t}$ are respectively, the input, forget and output gates, and $\odot$ represents the element-wise multiplication. The function sigma and tanh indicate sigmoid and hyperbolic tangent functions respectively. The gates are vectors that assume values in the range $[0, 1]$ and decide, component by component, which part of the input, of the previous hidden state and of the candidate output should flow through the network.

The LSTMs are used nowadays for several applications ranging from automatic speech recognition [37], machine translation [38], image captioning [39], and all the tasks requiring sequence processing or generation.

Fully connected layer

The fully connected layer is constituted by arrays of nonlinear units where each unit calculates a weighted sum of all the outputs from the preceding layer and generates an output signal by using a nonlinear function. The nonlinearity can be a smooth curve as the sigmoid or a piecewise linear function as the so-called *rectified linear units* (ReLU).

Dropout

The dropout [40, 41] is a layer that acts as a regularization to prevent the overfitting. It can be added after any layer of the network and it randomly sets to zero units from the preceding layer with a fixed probability p during the training. Without the dropout, each computing unit is pushed to learn to detect one single feature from the preceding layer, which is a clear sign of overfitting.

By shutting down the units randomly, the network cannot rely on the outcome of specific units and the learning is spread among the features. The regularization obtained with this method can be considered as an efficient model averaging. In fact, as during each round of training the network architecture is different, it is equivalent to train a high number of networks with shared parameters. When $p = 0.5$, it is equivalent to training $2^{|W|}$ networks with shared parameters, where $|W|$ is the number of neurons subject to dropout.

The proposed deep architecture

Finding the most suitable architecture for a DLNN is an open problem. Zeng et al. [42] analyze the design of the convolutional neural networks for sequence classification. The authors studied the structure of the convolutional neural network by varying the number of kernels and the number of layers. Among the conclusions of the work, the authors find that increasing the number of convolutional layers gives only a small improvement in the network performances, and a more complex architecture needs more training time and more training data to obtain a little gain in the classification results. Starting from these observations, the authors conclude that CNN performances do not scale easily with the complexity of the network. Following these results, in this work we experiment a different approach by adding an LSTM layer after the convolution and the sub-sampling. The idea is to let the convolutional layer to extract the most relevant local features, and in the following step use the LSTM to find the relations of these features along the sequence.

The proposed architecture is composed of different layers, as it is shown in Fig. 1: a convolutional layer, a max pooling layer, a dropout layer, a LSTM layer, and two fully connected layers. The first layer from left to right is a convolutional layer whose main role is the feature extraction from the input data x of $4 \times R$ binary values, where R is the length of the sequence. This layer is composed by a bank of $n = 50$ $1D$ convolutions [28] between the kernel vectors w^l $l = 1, 2, \ldots n$ and the input sequence x. The subsequent Max Pooling layer with width and stride 2 helps to capture the most salient features extracted by the convolution and reduces the output size of the input vectors. Then, the Dropout operation with probability $p = 0.5$ is used to prevent overfitting during the training phase. The LSTM layer scans the sequential features output of the previous layer and outputs its hidden state at each time step, it has 50 hidden memory units. The purpose is to find long-range relations between the time steps along all the sequence. The convolutional and LSTM layers have a regularizer L_2 with $\lambda = 0.001$. The outputs from all the LSTM time steps are then concatenated in a single vector. This vector is then fed to 2 subsequent fully-connected layers that reduce its length to 150 and then to 1. The first fully connected layer has a *ReLU* nonlinearity, and the output of the network is a real value in the interval $[0, 1]$ calculated by using a sigmoid function. We recall that the model we propose does not need any prior information about feature representation by means of any feature engineering process. For this reason, the input of our model is in the form *character-level one-hot encoding*. This representation takes into account each character i of the alphabet $\{A, C, G, T\}$ by a vector of length equal to the alphabet size, having all zero entries except for a single one in position i. This method leads to a sparse representation of the input.

The network has been implemented using the Keras [43] environment on different hardwares running both Tensorflow [44] and Theano [45] libraries deep learning framework, and it has been trained by stochastic gradient descent using the *Adam* optimizer [46]. The loss function is the binary cross entropy, which makes the training of the last layer equivalent to a logistic regression, but with the back-propagating gradient that affects also the feature learning.

Dataset descriptions

In this work we have used two groups of datasets, which are related to two recent papers about nucleosome positioning identification methods [18, 47]. We have chosen these datasets because they have been used to evaluate at least one state-of-the-art methodology for nucleosome positioning identification. As a consequence, we can easily put our results in comparison with several recently proposed methods.

The first group of datasets is composed of three sets of DNA sequences underlying nucleosomes from *Homo sapiens*(HM) , *Caenorhabditis elegans* (CE) and *Drosophila melanogaster*(DM). The details about all the steps that have been adopted in order to extract and filter such data can be found in the paper by Guo et al. [18] and in the references therein. Each dataset is composed of

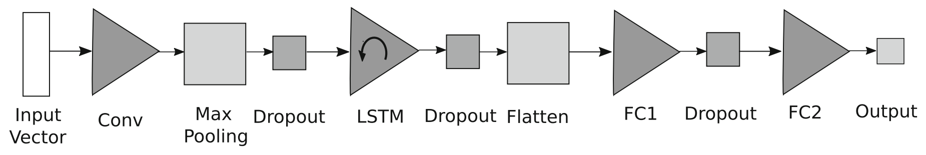

Fig. 1 The deep neural network architecture of the classifier

Table 1 The distribution of samples in the first group of dataset

	First group		
	HM	DM	CE
N	2273	2900	2567
L	2300	2850	2608
T	4573	5750	5175

HM indicates the group of Human sequences; DM indicates the group of Drosophyla sequences; CE indicates the group of C. Elegans sequences. The row label N indicates the nucleosome sequences; the row label L indicates the linker sequences and T inidicates the total number of sequences

data sequences of length 147 base pairs (bp) grouped in two classes of samples: the nucleosome-forming sequence samples (positive data) and the linkers, or nucleosome-inhibiting sequence samples (negative data). The distribution of the classes for this group of datasets is shown in Table 1. The second group of datasets is related to the paper by Liu et al. [47] and is composed of eight sets of DNA sequences related to three species: *Homo sapiens* (HM), *Drosophila melanogaster* (DM) and *Yeast* (YS). Unlike the first group of datasets, in this second case, different categories of sequences are present from each of the species. In particular, we find whole genome (WG) and promoter (PM) sequences of YS, and the largest chromosome (LC), promoter (PM) and 5'UTR exon region (5U) sequences from DM and HM. The authors of the related paper have provided only the bed files of the datasets, thus we had to retrieve the sequences of each set with the following procedure. We have first extracted the coordinates of the midpoints of each nucleosomal and linker sequence from the bed files and got the sequence range coordinates by extending 75 bp left and 75 right of the midpoint. We have then fetched the 151 bp sized nucleosomal and linker sequences using the genome files downloaded from UCSC Table Browser. The distribution of the classes for this group of datasets is shown in Table 2.

Results an discussion

We have performed experiments in two different settings, one for each group of datasets, in order to be comparable with the state of the art.

In the first setting, a 10-fold schema is used to evaluate the first dataset group. For each iteration, 1 fold is used for testing and the other 9 for training. A 10% of the dataset is selected among the training set as a validation step for early stopping. The predicted labels are obtained by thresholding the output value of the DLNN, that ranges in the interval $[0, 1]$, with the value 0.5. Output values below 0.5 are classified as linkers otherwise as nucleosomes. In the second experimental setting, we have computed the receiver operating characteristic (ROC) curves and the Area under the ROC curves (AUC) for all the datasets of the second group. Liu et al. [47] have proposed an evaluation protocol for these datasets that consists in sampling 100 test samples (with replacement) of 100 sequences each from the whole dataset, computing the ROC curve for every sample and finally the average of the sensitivity and specificity over the 100 samples. Unlike the methods used in that work, our method requires a training set; hence we have split each dataset a priori in a training, validation and test set. The split between training and test set is made in a way that there are enough data for training a strong model while having a large pool from which to select test data, and not to bias the comparison. After the training/test split, we select a number of samples equivalent to the 10% of the dataset size from the training set as a validation set used for early stopping. If the validation size would exceed 1000, we clip it to this number. The numbers used for the splits are listed in Table 3. The training phase is constituted by 100 epochs, with a learning rate of $3 * 10^{-4}$. The inputs are divided in min batches with a batch size of 64. For each dataset we train one only model using the selected training set and then compute the ROC curve and the AUC, as well as the specificity and sensitivity over the 100 tests. Specificity and sensitivity are computed by setting a score threshold, i.e. a sequence is classified as a nucleosomal sequence if the corresponding output of the network is greater than the predefined threshold or otherwise it is classified as a linker sequence. For all the experiments we have used two different networks, named *DLNN-3* and *DLNN-5*. The two networks differ in the size of the convolution kernels that compose the first layer of the network, which is 3 for the first network and 5 for the second.

Table 2 The distribution of samples in the second group of dataset

	Second group							
	HM			DM			YS	
	LC	PM	5U	LC	PM	5U	WG	PM
N	97209	56404	11769	46054	48251	4669	39661	27373
L	65563	44639	4880	30458	28763	2704	4824	4463
T	162772	101043	16649	76512	77014	7373	44485	31836

The meaning of the row labels is the same of the Table 1; The label YS indicates the Yeast group of sequences; WC indicates the whole genome, LC indicates the largest chromosome, PM indicates the Propoter sequences; 5U indicates the sequences from the 5UTR exon region

Table 3 Training/Validation/Test split for the second group of datasets

Dataset	Total samples	#Training	#Validation	#Test
HM-LC	162772	81298	1000	80474
HM-PM	101043	51321	1000	48722
HM-5U	16649	10654	1000	4995
DM-LC	76512	55512	1000	20000
DM-PM	77014	68312	1000	7702
DM-5U	7373	3687	737	2949
YS-WG	44485	39485	1000	4000
YS-PM	31836	27836	1000	3000

Results on first group of dataset

The results obtained by the DLNN we propose are shown in Table 4. We have reported mean values of accuracy, sensitivity and specificity of the three datasets computed by the two versions of the DLNN. We also report the results of the *iNuc-PseKNC* method computed on the same dataset as shown in the results of the related paper [18]. In the table, the best values are shown in bold. It is clearly observable that the models we are proposing outperform the iNuc-PseKNC on each dataset in terms of Sensitivity. This means that our models are more able to predict nucleosome class than iNuc-PseKNC. The improvements in the sensitivity of our methods are always considerable and can be up to +9% for the case of DM. The method iNuc-PseKNC continues to be better than our model in accuracy and specificity only on the HM dataset.

Results on second group of dataset

On these group of datasets, we know the results of eight state of the art methodologies for nucleosome positioning identification [10–17] as reported in the reference paper [47]. Table 5 summarize the AUCs computed by other methods in the first eight column of the table. Note that in most of the cases the paper by Liu [47] does not report precise values of the AUC, thus we have reported approximate values in terms of open interval (minimum and maximum values of the interval in brackets) or very close to (symbol '∼'). Such approximate values have been extracted looking at the plot bars shown in the paper. The last two column summarizes the experiments of the two version of the networks (DLNN-3 and DLNN-5). We point out that the networks share the same architecture, but use different training set opportunely extracted from the species, and complementary to the test set used for computing the ROC. Also in the case of this kind of experiments, the bold values indicate the best AUC performances. In the case of DLNN-3, its AUC values are better than any other method on HM-LC, YS-WG, HM-PM datasets, and equal to the best on DM-LC, DM-PM, DM-5U. Only in the case of YS-PM and HM-5U the AUC values are worst but very close to the best ones. The AUC values are further improved by the DLNN-5, showing the best values on HM-LC, YS-WG, HM-PM, YS-PM, and performing equally to the best in DM-PM, DM-LC, letting HM-5U the only dataset where it performs worst.

Conclusions

In this study, a deep neural network model for the automatic classification of nucleosome forming sequences has been proposed. The proposed network is able to exploit the information about nucleotides position in a sequence, by using a mix of convolutional and recurrent layers. The effectiveness of the proposed method is proved in terms of Sensitivity, specificity, accuracy and AUC values, for which it pushes forward the state of the art in almost all the considered datasets. Anyway, we think that the major advantage of what we are proposing, is that it does not make any use of any a priori information, or feature extraction or selection step, about the features that are

Table 4 10-fold cross validation performances on the first group of dataset

Method(Species)	Accuracy		Sensitivity		Specificity	
	μ	σ	μ	σ	μ	σ
iNuc-PseKNC(CE)	86.90	x	90.30	x	83.55	x
iNuc-PseKNC(DM)	79.97	x	78,31	x	81.65	x
iNuc-PseKNC(HM)	**86,27**	x	87,86	x	**84,70**	x
DLNN-3(CE)	**89.60**	0.8	**93.36**	1.27	**85.93**	2,13
DLNN-3(DM)	**85.54**	1.13	**87.60**	2.55	**83.42**	2.65
DLNN-3(HM)	84.65	2.16	**89.67**	2.83	79.64	4.29
DLNN-5(CE)	**89.62**	2.45	**93.04**	3.68	**86.34**	5.54
DLNN-5(DM)	**85.60**	0.75	**87.81**	2.79	**83.33**	2.74
LNN-5(HM)	85.37	1.91	**88.34**	1,82	82.29	4.86

iNuc-PseKNC refers to the method introduced in [18]; CE, DM, HM refers to the datasets descried in Table 1; DLNN refers to the DLNN proposed in this paper and -3 or -5 refers to the kernel dimension in the first convolutional layer of the net. Best values are in bold

Table 5 Area under the ROC curve performances on the first group of dataset

	N_score [16]	NuPop [17]	NucEnergEN [14]	Segal [10]	Field [11]	Kaplan [12]	Heijden [13]	Finestr [15]	DLNN-3	DLNN-5
HM-LC	~0.65	(0.6,0.65)	(0.6,0.65)	(0.35,0.4)	~0.65	~0.65	~0.6	(0.6, 0.65)	**0.79**	**0.81**
DM-LC	0.59	(0.65,0.70)	~0.7	0.33	(0.70,0.75)	**(0.70,0.75)**	(0.65,0.70)	0.57	**0.71**	**0.71**
YS-WG	0.77	0.74	(0.65,0.70)	0.49	0.77	~0.7	~0.65	~0.7	**0.79**	**0.83**
HM-PM	(0.6,0.65)	0.67	(0.6,0.65)	(0.4,0.45)	(0.6,0.65)	~0.6	~0.55	~0.55	**0.77**	**0.77**
DM-PM	0.62	~0.7	(0.70,0.75)	0.32	**(0.70,0.75)**	**(0.70,0.75)**	(0.55,0.6)	(0.5,0.55)	**0.72**	**0.73**
YS-PM	0.70	0.74	(0.7,0.75)	0.52	0.79	(0.7,0.75)	~0.65	~0.7	0.73	**0.83**
HM-5U	(0.55,0.6)	~0.65	**~0.7**	0.37	~0.65	~0.65	~0.6	(0.55,0.6)	0.67	0.68
DM-5U	0.54	(0.6,0.65)	(0.65,0.70)	0.38	**(0.65,0.70)**	**(0.65,0.70)**	(0.55,0.6)	~0.5	**0.66**	**0.67**

Each column refers to a computational method for nucleosome positioning. The rst eight column show the values reported in the paper by Liu et al. [47], sometimes with approximate values (interval range or close to symbol"). Last two columns regard our proposed method. Best values are in bold relevant for the classification of nucleosomes. It automatically learns the task by using the training set of labelled sequences, using the *character-level one-hot encoding* as input representation. Actually, at least 300 of experimental datasets of genome-wide nucleosome occupancy profiles determined in the different cell type of about ten species are currently available [48]. Taking into consideration this actual scenario, we think that the method we propose, due to this precision, could be very useful in the study of the difference in nucleosome organization between different species.

Abbreviations

AUC: Area under the ROC curve; CE: Caenorhabditis elegans; CNNs: Convolutional neural networks; DLNN: Deep learning neural networks; DM: Drosophila melanogaster; HM: Homo sapiens; LC: Largest chromosome; LSTM: Long short-term memory; PM: Promoter; ReLU: rectified linear units; RNNs: Recurrent neural networks; ROC: Receiver operating characteristic; WG: Whole genome; YS: Yeast; 5U: 5'UTR exon region

Funding

Publication of this manuscript was sponsored by annual funding for basic research (FFABR) of the Italian Ministry for Education.

Availability of data and materials

The Keras source code of the networks used in this work are freely available at https://github.com/mattiadg/Nucleosome-predict. The data used for the experiments are downloadable as supplementary data of the two related papers [18, 47].

About this supplement

This article has been published as part of *BMC Bioinformatics Volume 19 Supplement 14, 2018: Selected articles from the 5th International Work-Conference on Bioinformatics and Biomedical Engineering: bioinformatics*. The full contents of the supplement are available online at https://bmcbioinformatics.biomedcentral.com/articles/supplements/volume-19-supplement-14.

Authors' contributions

GLB and RR conceived the study. GLB, RR, MDG designed the methodology. MDG implemented the methodology. GLB prepared the data. RR and MDG ran the experiments. GLB and RR analyzed and interpreted the data. The authors read and approved the final manuscript.

Authors' information

Mattia A. Di Gangi is currently a PhD Student at the University of Trento, Italy. He received his "cum laude" M.Sc. in Computer Science from the University of Palermo, Italy. His research interests focus on deep learning and machine translation, and his research is undertaken in collaboration with Fondazione Bruno Kessler (FBK) of Trento, Italy. **Giosué Lo Bosco** received his degree in Mathematics and the Ph.D. degree in computer science from the University of Palermo, Italy, in 2000 and 2004, respectively. He is currently Associate Professor of Computer Science at the Department of Mathematics and Computer Science, University of Palermo. His research is focused on the design of Pattern Recognition and Machine learning methodologies, with applications to biomedical data. **Riccardo Rizzo** received his degree in Electronic Engineering on 1990. He is currently staff researcher at Institute for high performance computing and networking, Italian National Research Council. His research is focused mainly on machine learning methods for sequence analysis, biomedical data and on neural networks applications.

Ethics approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

Competing interests

The authors declare that they have no competing interests.

Author details

[1]Fondazione Bruno Kessler, Via Sommarive, 18, 38123 Trento, Italy. [2]ICT International Doctoral School, Via Sommarive, 9, 38123 Trento, Italy. [3]Dipartimento di Matematica e Informatica, Università degli studi di Palermo, Via Archirafi, 34, 90123 Palermo, Italy. [4]Dipartimento di Scienze per l'Innovazione tecnologica, Istituto Euro-Mediterraneo di Scienza e Tecnologia, Via Michele Miraglia, 20, 90139 Palermo, Italy. [5]CNR-ICAR, National Research Council of Italy, Via Ugo La Malfa, 153, 90146 Palermo, Italy.

Published: 20 November 2018

References

1. Svaren J, Horz W. Transcription factors vs. nucleosomes: Regulation of the pho5 promoter in yeast. Trends Biochem Sci. 1997;22:93–7.
2. Kornberg R, Lorch Y. Twenty-five years of the nucleosome, fundamental particle of the eukaryote chromosome. Cell. 1999;98(3):285–94.
3. Hodges C, Bintu L, Lubkowska L, Kashlev M, Bustamante C. Nucleosomal fluctuations govern the transcription dynamics of rna polymerase ii. Science. 2009;325(5940):626–8.
4. Tilgner H, Nikolaou C, Althammer S, Sammeth M, Beato M, Valcárcel J, Guigó R. Nucleosome positioning as a determinant of exon recognition. Nat Struct Mol Biol. 2009;16(9):996–1002.
5. Choi JK, Kim YJ. Intrinsic variability of gene expression encoded in nucleosome positioning sequences. Nat Genet. 2009;41(4):498–503.
6. Struhl K, Segal E. Determinants of nucleosome positioning. Nat StructMol Biol. 2013;20(3):267–73.
7. Yuan G-C. Linking genome to epigenome. Wiley Interdiscip Rev Syst Biol Med. 2012;4(3):297–309.
8. Sala A, Toto M, Pinello L, Gabriele A, Di Benedetto V, Ingrassia AMR, Lo Bosco G, Di Gesù V, Giancarlo R, Corona DFV. Genome-wide characterization of chromatin binding and nucleosome spacing activity of the nucleosome remodelling atpase iswi. EMBO J. 2011;30(9):1766–77.
9. Pinello L, Lo Bosco G, Yuan G-C. Applications of alignment-free methods in epigenomics. Brief Bioinform. 2014;15(3):419–30.
10. Segal E, Fondufe-Mittendorf Y, Chen L, Thåström A, Field Y, Moore I, Wang J, Widom J. A genomic code for nucleosome positioning. Nature. 2006;442(5):772–8.
11. Field Y, Kaplan N, Fondufe-Mittendorf Y, Moore IK, Sharon E, Lubling Y, Widom J, Segal E. Distinct modes of regulation by chromatin encoded through nucleosome positioning signals. PLoS Comput Biol. 2008;4(11).
12. Kaplan N, Moore IK, Fondufe-Mittendorf Y, Gossett AJ, Tillo D, Field Y, LeProust EM, Hughes TR, Lieb JD, Widom J, Segal E. The dna-encoded nucleosome organization of a eukaryotic genome. Nature. 2009;458(7236):362–6.
13. van der Heijden T, van Vugt JJ, Logie C, van Noort J. Sequence-based prediction of single nucleosome positioning and genome-wide nucleosome occupancy. Proc Natl Acad Sci USA. 2010;109(38):2514–22.
14. Locke G, Tolkunov D, Moqtaderi Z, Struhl K, Morozov AV. High-throughput sequencing reveals a simple model of nucleosome energetics. Proc Natl Acad Sci USA. 2010;107(49):20998–1003.
15. Gabdank I, Barash D, Trifonov EN. Finestr: a web server for single-base-resolution nucleosome positioning. Bioinformatics. 2010;26(6):845–6.
16. Yuan G-CC, Liu JS. Genomic sequence is highly predictive of local nucleosome depletion. PLoS Comput Biol. 2008;4(1):13.
17. Xi L, Fondufe-Mittendorf Y, Xia L, Flatow J, Widom J, Wang J-P. Predicting nucleosome positioning using a duration hidden markov model. BMC Bioinformatics. 2010;11(1):346.
18. Guo S-H, Deng E-Z, Xu L-Q, Ding H, Lin H, Chen W, Chou K-C. inuc-pseknc: a sequence-based predictor for predicting nucleosome positioning in genomes with pseudo k-tuple nucleotide composition. Bioinformatics. 2014;30(11):1522–9.
19. Kuksa P, Pavlovic V. Efficient alignment-free dna barcode analytics. BMC Bioinformatics. 2009;10(14):9.
20. Pinello L, Lo Bosco G, Hanlon B, Yuan G-C. A motif-independent metric for dna sequence specificity. BMC Bioinformatics. 2012;12:408.

21. Pinello L, Lo Bosco G. A new feature selection methodology for k-mers representation of dna sequences. In: Computational Intelligence Methods for Bioinformatics and Biostatistics. Lecture Notes in Computer Science, vol. 8623. 2015. p. 99–108.
22. Rizzo R, Fiannaca A, La Rosa M, Urso A. The general regression neural network to classify barcode and mini-barcode dna. In: Computational Intelligence Methods for Bioinformatics and Biostatistics, Lecture Notes in Computer Science, vol. 8623. 2015. p. 142–55.
23. Lo Bosco G. Alignment free dissimilarities for nucleosome classification. In: Computational Intelligence Methods for Bioinformatics and Biostatistics, Lecture Notes in Computer Science, vol. 9874. 2016. p. 114–28.
24. Fiannaca A, La Rosa M, Rizzo R, Urso A. Analysis of dna barcode sequences using neural gas and spectral representation. In: Iliadis L, Papadopoulos H, Jayne C, editors. Engineering Applications of Neural Networks, Communications in Computer and Information Science, vol 384. Berlin: Springer; 2013. p. 212–21.
25. Fiannaca A, La Rosa M, Rizzo R, Urso A. A k-mer-based barcode dna classification methodology based on spectral representation and a neural gas network. Artif Intell Med. 2015;64(3):173–84. https://doi.org/10.1016/j.artmed.2015.06.002.
26. Bengio Y. Learning deep architectures for ai. Found Trends Mach Learn. 2009;2(1):1–127.
27. LeCun Y, Bengio Y, Hinton G. Deep learning. Nature. 2015;521(7553): 436–44.
28. LeCun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. Proc IEEE. 1998;86(11):2278–324.
29. Collobert R, Weston J, Bottou L, Karlen M, Kavukcuoglu K, Kuksa P. Natural language processing (almost) from scratch. J Mach Learn Res. 2011;12:2493–537.
30. Rizzo R, Fiannaca A, La Rosa M, Urso A. A deep learning approach to dna sequence classification. In: Computational Intelligence Methods for Bioinformatics and Biostatistics, Lecture Notes in Computer Science vol. 9874. 2016. p. 129–140.
31. Lo Bosco G, Di Gangi MA. Fuzzy Logic and Soft Computing Applications: 11th International Workshop, WILF 2016, Naples, Italy, December 19–21, 2016. In: Petrosino A, Loia V, Pedrycz W, editors. Revised Selected Papers. Springer; 2017. p. 162–71.
32. Fiannaca A, La Paglia L, La Rosa M, Lo Bosco G, Renda G, Rizzo R, Gaglio S, Urso A. Deep learning models for bacteria taxonomic classication of metagenomic data. BMC Bioinformatics. 2018;19(S7):198.
33. Lo Bosco G, Rizzo R, Fiannaca A, La Rosa M, Urso A. A deep learning model for epigenomic studies. In: 12th International Conference on Signal-Image Technology Internet-Based Systems (SITIS). Naples; 2016. p. 688–92.
34. Di Gangi MA, Gaglio S, La Bua C, Lo Bosco G, Rizzo R. A deep learning network for exploiting positional information in nucleosome related sequences. In: Rojas I, Ortuño F, editors. Bioinformatics and Biomedical Engineering: 5th International Work-Conference, IWBBIO 2017, Granada, Spain, April 26–28, 2017, Proceedings, Part II. Springer; 2017. p. 524–33.
35. Hochreiter S, Bengio Y, Frasconi P, Schmidhuber J. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In: Kremer SC, Kolen JF, editors. A Field Guide to Dynamical Recurrent Neural Networks. IEEE Press; 2001.
36. Hochreiter S, Schmidhuber J. Long short-term memory. Neural Comput. 1997;9(8):1735–80.
37. Graves A, Mohamed A-r, Hinton G. Speech recognition with deep recurrent neural networks. In: Acoustics, Speech and Signal Processing (icassp), 2013 Ieee International Conference On. IEEE; 2013. p. 6645–9.
38. Di Gangi MA, Bertoldi N, Federico M. Fbk's participation to the english-to-german news translation task of wmt 2017. In: 2nd Conference on Machine Translation (WMT17), vol 2. Copenhagen; 2017. p. 271–5.
39. Wang C, Yang H, Bartz C, Meinel C. Image captioning with deep bidirectional lstms. In: Proceedings of the 2016 ACM on Multimedia Conference, MM '16. New York: ACM; 2016. p. 988–97. https://doi.org/10.1145/2964284.2964299. http://doi.acm.org/10.1145/2964284.2964299.
40. Hinton GE, Srivastava N, Krizhevsky A, Sutskever I, Salakhutdinov RR. Improving neural networks by preventing co-adaptation of feature detectors. arXiv e-prints. 2012;abs/1207.0580:1–18.
41. Srivastava N, Hinton GE, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res. 2014;15(1):1929–58.
42. Zeng H, Edwards MD, Liu G, Gifford DK. Convolutional neural network architectures for predicting dna–protein binding. Bioinformatics. 2016;32(12):121–7.
43. Chollet F, et al. Keras. GitHub. 2015. https://github.com/fchollet/keras.
44. Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, Corrado GS, Davis A, Dean J, Devin M, Ghemawat S, Goodfellow I, Harp A, Irving G, Isard M, Jia Y, Jozefowicz R, Kaiser L, Kudlur M, Levenberg J, Mané D, Monga R, Moore S, Murray D, Olah C, Schuster M, Shlens J, Steiner B, Sutskever I, Talwar K, Tucker P, Vanhoucke V, Vasudevan V, Viégas F, Vinyals O, Warden P, Wattenberg M, Wicke M, Yu Y, Zheng X. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. Software available from tensorflow.org. 2015. https://www.tensorflow.org/.
45. Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. arXiv e-prints. 2016;abs/1605.02688:1–19.
46. Kingma D, Ba J. Adam: A method for stochastic optimization. Proc 3rd Int Conf Learn Represent (ICLR). 2015.
47. Liu H, Zhang R, Xiong W, Guan J, Zhuang Z, Zhou S. A comparative evaluation on prediction methods of nucleosome positioning. Brief Bioinform. 2014;15(6):1014–27. https://doi.org/10.1093/bib/bbt062.
48. Teif VB. Nucleosome positioning: resources and tools online. Brief Bioinform. 2016;17(5):745–57.