Deep Learning Architectures for DNA Sequence Classification

Giosué Lo Bosco[1,2(✉)] and Mattia Antonino Di Gangi[3,4]

[1] Dipartimento di Matematica e Informatica, Universitá degli studi di Palermo, Palermo, Italy
giosue.lobosco@unipa.it

[2] Dipartimento di Scienze per l'Innovazione e le Tecnologie Abilitanti, Istituto Euro Mediterraneo di Scienza e Tecnologia, Palermo, Italy

[3] Fondazione Bruno Kessler, Trento, Italy

[4] ICT International Doctoral School, University of Trento, Trento, Italy

Abstract. DNA sequence classification is a key task in a generic computational framework for biomedical data analysis, and in recent years several machine learning technique have been adopted to successful accomplish with this task. Anyway, the main difficulty behind the problem remains the feature selection process. Sequences do not have explicit features, and the commonly used representations introduce the main drawback of the high dimensionality. For sure, machine learning method devoted to supervised classification tasks are strongly dependent on the feature extraction step, and in order to build a good representation it is necessary to recognize and measure meaningful details of the items to classify. Recently, neural deep learning architectures or deep learning models, were proved to be able to extract automatically useful features from input patterns. In this work we present two different deep learning architectures for the purpose of DNA sequence classification. Their comparison is carried out on a public data-set of DNA sequences, for five different classification tasks.

Keywords: DNA sequence classification · Convolutional Neural Networks · Recurrent Neural Networks · Deep learning networks

1 Introduction

One of the primary goal in biology is the understanding of the relationships between protein structure and function. To understand the structure-function paradigm, particularly useful structural information comes from the primary amino acid sequences. DNA sequence classification is extremely useful for this task, following the principle that sequences having similar structures have also similar functions. Sequence similarity is traditionally established by using

G. Lo Bosco and M.A. Di Gangi—Both authors have the same contribution to this paper.

A. Petrosino et al. (Eds.): WILF 2016, LNAI 10147, pp. 162–171, 2017.
DOI: 10.1007/978-3-319-52962-2_14

sequence alignment methods, such as BLAST [1] and FASTA [2]. This choice is motivated by two main assumptions: (1) the functional elements share common sequence features and (2) the relative order of the functional elements is conserved between different sequences. Although these assumptions are valid in a broad range of cases, they are not general. For example, in the case of cis-regulatory elements related sequences, there is little evidence suggesting that the order between different elements would have any significant effect in regulating gene expression. Anyway, despite recent efforts, the key issue that seriously limits the application of the alignment methods still remains their time computational complexity. As such, the recently developed alignment-free methods [3,4] have emerged as a promising approach to investigate the regulatory genome. For sure, these methodologies involve a feature extraction phase such as spectral representation of DNA sequences [5–7]. The performance of conventional machine learning algorithms depends on feature representations, which are typically designed by experts. As a subsequent phase, it is necessary to identify which features are more appropriate to face with the given task, and nowadays this remains a very crucial and difficult step.

Deep learning has recently emerged as a successful paradigm for big data, also because of the technological advances in terms of the low level cost of parallel computing architectures, so that deep learning has given significant contributions in several basic but arduous artificial intelligence tasks. Very important advancements have been made in computer vision [8,9], but also in natural language processing (NLP) [10,11] and machine translation [12–14], where deep learning techniques represent now the state of the art.

The main contribution of deep learning methods to bioinformatics have been in genomic medicine and medical imaging research field. To the best of our knowledge, very few contribution have been provided for the sequence classification problem. For a deep review about deep learning in bioinformatics, see the review by Seonwoo et al. [15].

The taxonomy of deep neural models mainly includes *Convolutional Neural Networks (CNN)*, *Recurrent Neural Networks (RNN)* and *Stacked Auto-encoders (SAE)*. CNNs are characterized by an initial layer of convolutional filters, followed by a non Linearity, a sub-sampling, and a fully connected layer which realized the final classification. LeNet [16] was one of the very first convolutional neural networks which helped propel the field of Deep Learning. Collobert et al. [17] firstly shown that CNNs can be used effectively also for sequence analysis, in the case of a generic text. RNNs are designed to exploit sequential information of input data with cyclic connections among building blocks like perceptrons or long short-term memory units (LSTMs). Differently from other architectures, RNNs are capable of handling sequence information over time, and this is an outstanding property in the case of sequence classification. They are able to incorporate contextual information from past inputs, with the advantage to be robust to localised distortions of the input sequence along the time.

In this study we compare CNNs and RNNs deep learning architectures for the special case of bacterial classification using a publicly available data-set. We will use a variant of the LeNet network as choice for the architecture of the CNN, and a long short-term memory (LSTM) network as choice for RNN. The latter, is a type of recurrent neural networks with a more complex computational unit that leads to better performance. We compare these networks with their variants trained for multi-task learning, which should exploit the relations between the classes at the different levels of the taxonomy. In the next two sections the main components of CNNs and RNNs deep learning architectures are introduced and described, in Sect. 4 data-sets, experiments and results are reported. Conclusions and remarks are discussed in Sect. 5.

2 Deep Learning Architectures

2.1 Convolutional Neural Networks

The convolutional layers are the core components of a CNN. They calculate L one-dimensional convolutions between the kernel vectors w^l, of size $2n+1$, and the input signal x:

$$q^l(i) = \sum_{u=-n}^{n} w^l(u)x(i-u) \tag{1}$$

In Eq. 1 $q^l(i)$ is the component i of the l-th output vector and $w^l(u)$ is the component u of the l-th kernel vector. After a bias term b^l is added and the nonlinear function g is applied:

$$h^l(i) = g(q^l(i) + b^l). \tag{2}$$

Another important component of a CNN is the max-pooling layer, that usually follows the convolutional layer in the computation flow. It is a non-linear down-sampling layer that partitions the input vector into a set of non-overlapping regions and, for each sub-region, the maximum value is considered as output. This processing layer reduces the complexity for the following layers and operates a sort of translational invariance. Convolution and max-pooling are usually considered together as two highly connected blocks.

2.2 Recurrent Neural Networks

Recurrent Neural Networks are generally used for processing sequences of data which evolves along the time axis. The simpler version of a RNN owns an internal state h_t which is a summary of the sequence seen until the previous time step $(t-1)$ and it is used together with the new input x_t:

$$h_t = \sigma(W_h x_t + U_h h_{t-1} + b_h)$$

$$y_t = \sigma(W_y h_t + b_y)$$

where W_h and U_h are respectively the weight matrices for the input and the internal state, W_y is the weight matrix for producing the output from the internal state, and the two b are bias vectors.

One problem that limits the usage of RNNs that respect this formulation, is that all the time steps have the same weight and, consequently, the contribution of an input in the hidden state is subjected to exponential decay. A variant of RNN has been introduced in [18] with the name of *Long Short-Term Memory (LSTM)*. LSTM is a recurrent neural unit consisting of four internal gates, each computing a different function of the input, that allow the layer to exploit long range relations. The four gates provide the unit with the capability of choosing the weight to give to every single input, and to forget the current hidden state for the computation of the next output.

2.3 Softmax Layer

Both the two previous architectures needs a further layer in order to compute a classification task. Softmax layer [19] is composed by K units, where K is the number of different classes. Each unit is densely connected with the previous layer and computes the probability that an element is of class k by means of the formula:

$$softmax_k(x) = \frac{e^{W_k x + b_k}}{\sum_{l=1}^{k} e^{W_l x + b_l}}$$

where W_l is the weight matrix connecting the l-th unit to the previous layer, x is the output of the previous layer and b_l is the bias for the l-th unit. Softmax is widely used by deep learning practitioners as a classification layer because of the normalized probability distribution it outputs, which proves particularly useful during back-propagation.

2.4 Character Embedding

In this work we evaluate models for classifying genomic sequences without providing a-priori information by means of feature engineering. One method for achieving this is represented by the use of *character-level one-hot encoding*. This representation takes into account each character i of the alphabet 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, which is tackled in the NLP literature by means of an embedding layer.

Word Embedding [20,21] is used to give a continuous vector representation to each word of a sequence, and in the vector space obtained with this process words with a similar meaning are close in distance.

The same idea has been also applied at character-level [22], and while in this case there is no semantic similarity to find, it represents a first step for fully-automated feature learning.

The embedding representation can be implemented by a single feed-forward layer with a non-linear activation, where each row of the weight matrix W gives the representation for the corresponding symbol and it can be jointly trained with the rest of the network.

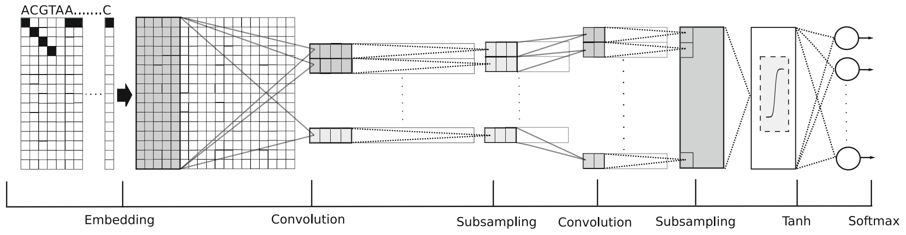

Fig. 1. The architecture of the proposed CNN

3 Deep Learning Networks for DNA Sequence Classification

As the used data-set encodes the sequences according to *IUPAC* nucleotide codes, the vectors that we consider are of size 16. Note that the IUPAC alphabet is composed by 15 characters, so we have used a padding character in order to have only equally sized sequences.

3.1 Convolutional Neural Network

The Convolutional Neural Network (CNN) used in this study is inspired by Lecun's LeNet-5 [16]. The first layer is an embedding layer, which takes as input 16-dimensional one-hot encoding of sequence characters, and produces a 10 dimensional continuous vector. On top of the input we have two convolutional layers, each followed by a max pooling layer. The two convolutional layers use filters of size 5 and have a growing number of filters, respectively 10 and 20. The width and the stride of the pooling layers are both of 5 time steps, considering a one-hot vector as a single unit.

The convolutional layers are stacked with two fully connected layers. The first one, consisting of 500 units, uses a *tanh* activation function, which is the same activation used by the lower levels. The second one is the classification layer and uses the $softmax$ activation.

This network is similar to that proposed in [23], except for the addition of an embedding layer and the absence of preprocessing of the sequences that is achieved by means of the one-hot encoding. Figure 1 shows the architecture of the proposed CNN.

3.2 Recurrent Neural Network

The recurrent network is a 6-layered network whose input is in the form of one-hot encoding vectors, analogously to the previous section's model. The first layer is an embedding layer and it is followed by a max pooling layer of size 2. The max pooling reduces the computation for the following layer, and at the same time gives some capability of translational invariance to the network. After the

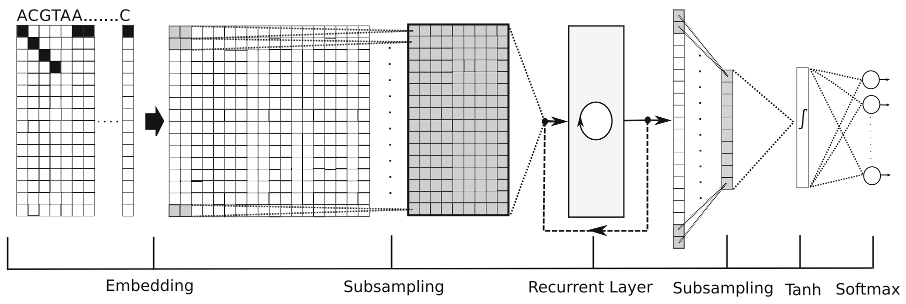

Fig. 2. The architecture of the proposed LSTM

max pooling there is a recurrent layer, implemented in this work as an LSTM, which processes its input from left to right and produces an output vector of size 20 at each time step. The LSTM layer is followed by another max pooling layer and in the top we have two fully-connected layers like those explained in the previous section (Fig. 2).

3.3 Multi-task Learning

Multi-task learning, in its original formulation, is the joint training of a machine learning system on multiple related tasks [24] in order to share the internal representation and achieve a better regularization. The used data-set is organized into five nested taxonomic ranks, i.e. each of the labels at a given rank of the taxonomy exists within one of the categories of the more coarse rank. For this reason it appears natural to jointly train a deep learning model for the five ranks at the same time.

We propose a multi-task learning variant for both networks by simply replicating the dense layers. The first dense layer is replicated five times, once for each taxon, each replica having 500 units, exactly like the layer in the original architecture. The softmax layers are also five, and each of them is stacked to only one of the fully-connected layers. We expect these networks to be able to perform better than their single task version, while requiring less time to train. In fact, even though the resulting network is bigger, the train is performed just once instead of five times, causing a sensitive gain in terms of time.

4 Experimental Results

4.1 Dataset Description

We have tested the effectiveness of the two proposed deep learning models in the case of the so called 16S data-set.

Studies about bacteria species are based on the analysis of their 16S ribosomal RNA (rRNA for short) housekeeping gene, as biologically validated by Drancourt et al. [25]. Ribosomes are the complex structures that acts the protein synthesis in each living organism. They consists of two subunits, each of which is composed of protein and a type of RNA, known as ribosomal RNA (rRNA). Prokaryotic ribosomes consist of 30S subunit and 50S subunit. The first subunit is also composed of 16S ribosomal RNA. The presence of hyper variable regions in the 16S rRNA gene provides a species specific signature sequence which is useful for bacterial identification process. Because the 16S rRNA gene is very short, just around 1500 nucleotide bases, it can be easily copied and sequenced. The use of 16S rRNA gene sequences to study bacterial phylogeny and taxonomy has been by far the most common housekeeping genetic marker used for several reasons that include its presence in almost all bacteria, its unchanged role function over evolution, and its versatility, in terms of length, for computational purposes.

The 16S rRNA data-set used in this study was downloaded from the RDP Ribosomal Database Project II (RDP-II) [26] and the selected sequences were chosen according to the type strain, uncultured and isolates source, average length of about 1200–1400 bps, good RDP system quality and *class* taxonomy by NCBI. By a subsequent filtering phase, a total amount of 3000 sequences have been selected, and can be grouped into 5 ordered taxonomic ranks, named *Phylum* (most coarse grained), *Class*, *Order*, *Family* and *Genus* (more fine grained). The resulting number of classes is increasing from a minimum of 3 classes (Phylum) to a maximum of 393 (Genus). Between them there are *Class*, *Order* and *Family* with 6, 22, 65 classes.

4.2 Experiments

For each taxonomic rank a 10-Fold cross-validation has been performed. We have chosen 15 epochs for each fold, without using early stopping validation. This choice has been motivated by our experimental observations.

In Table 1 we report the results of mean (μ) and standard deviation (σ) of the accuracy of the two networks computed on 10 test folds, for both single-task (CNN and LSTM) and multi-task (CNN-MT and LSTM-MT) learning.

Table 1. Average accuracies of the proposed models over 10-fold.

	Phylum		Class		Order		Family		Genus	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
CNN	0,995	0,003	0,993	0,006	0,937	0,012	0,893	0,019	0,676	0,065
CNN-MT	0,981	0,007	0,978	0,008	0,908	0,021	0,851	0,024	0,692	0,024
LSTM	0,982	0,028	0,977	0,022	0,902	0,028	0,857	0,034	0,728	0,030
LSTM-MT	0,992	0,007	0,990	0,008	0,941	0,029	0,897	0,023	0,733	0,030

As far as we know, the deep learning model state of the art for classifying this data-set is represented by a CNN that uses *k*-mers representation, that has been recently introduced by Rizzo et al. [23]. The comparison between the

Table 2. Approximate time in minutes for training one step of 10-fold cross-validation for the single-task (first five columns) and multi-task architecture (last column).

	Phylum	Class	Order	Family	Genus	Total	Mtask
CNN	6.5	6.5	6.7	6.7	6.8	33.2	21.5
LSTM	13.0	14.7	14.7	15.0	17.5	74.9	44.0

single-task models is absolutely favorable for the CNN, also in terms of training time, indeed the CNN can be trained in half the time than the LSTM for the same number of epochs and minibatch size. multi-task learning affects the two models in different ways, for instance it makes the CNN worse in the first four taxonomy ranks, while slightly improving the last, but improves the performance of LSTM, making it comparable to that of the single-task CNN on the first two categories and slightly better for the rest. Also in this case the time for the CNN is a bit less than the half respect to the time of the LSTM (see Table 2). Our results are anyway far from the state of the art, with the exception of the first two taxonomy ranks, which doesn't use positional information. We expect that this will not be the general case, and we plan to investigate on this by applying our networks to other sequence data sets related to other biological properties.

5 Conclusion

In this study, two deep learning architectures were compared for an automatic classification of bacteria species with no steps of sequence preprocessing. In particular, we have proposed a long short-time memory network (LSTM) that exploits the nucleotides positions of a sequence, in comparison with a classical convolutional neural network (CNN). Results show a superiority of the CNNs in the four simplest classification tasks, while its performance poorly degrades for the last where the LSTM works better. Moreover, we have studied how the multi-task learning affects the models, both in terms of performance and training time. Our results showed that it helps the LSTM for all the tasks, while it harms CNN performance. Further analysis is needed in order to explain this difference. Taking into consideration the results of the state of the art, we are going to expand the two models with positional information, coming from both convolutional and recurrent layers, trying to further exploit contextual information, for example with bidirectional RNNs.

References

1. Altschul, S., Gish, W., Miller, W., et al.: Basic local alignment search tool. J. Mol. Biol. **25**(3), 403–410 (1990)
2. Lipman, D., Pearson, W.: Rapid and sensitive protein similarity searches. Science **227**(4693), 1435–1441 (1985)

3. Vinga, S., Almeida, J.: Alignment-free sequence comparison a review. Bioinformatics **19**(4), 513–523 (2003)
4. Pinello, L., Lo Bosco, G., Yuan, G.-C.: Applications of alignment-free methods in epigenomics. Brief. Bioinform. **15**(3), 419–430 (2014)
5. Pinello, L., Lo Bosco, G., Hanlon, B., Yuan, G.-C.: A motif-independent metric for DNA sequence specificity. BMC Bioinform. **12**, 1–9 (2011)
6. Lo Bosco, G., Pinello, L.: A new feature selection methodology for K-mers representation of DNA sequences. In: Serio, C., Liò, P., Nonis, A., Tagliaferri, R. (eds.) CIBB 2014. LNCS, vol. 8623, pp. 99–108. Springer, Heidelberg (2015). doi:10.1007/978-3-319-24462-4_9
7. Lo Bosco, G.: Alignment free dissimilarities for nucleosome classification. In: Angelini, C., Rancoita, P.M.V., Rovetta, S. (eds.) CIBB 2015. LNCS, vol. 9874, pp. 114–128. Springer, Heidelberg (2016). doi:10.1007/978-3-319-44332-4_9
8. Farabet, C., Couprie, C., Najman, L., et al.: Learning hierarchical features for scene labeling. IEEE Trans. Pattern Anal. Mach. Intell. **35**(8), 1915–1929 (2013)
9. Tompson, J.J., Jain, A., LeCun, Y., et al.: Joint training of a convolutional network and a graphical model for human pose estimation. In: Advances in Neural Information Processing Systems, pp. 1799–1807 (2014)
10. Kiros, R., Zhu, Y., Salakhutdinov, R.R., et al.: Skip-thought vectors. In: Advances in Neural Information Processing Systems, pp. 3276–3284 (2015)
11. Li, J., Luong, M.-T., Jurafsky, D.: A hierarchical neural autoencoder for paragraphs and documents. In: Proceedings of 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, pp. 1106–1115 (2015)
12. Luong, M.-T., Pham, H., Manning, C.D.: Effective approaches attention-based neural machine translation. In: Proceedings of Conference on Empirical Methods in Natural Language Processing, pp. 1412–1421 (2015)
13. Cho, K., Van Merrienboer, B., Gulcehre, C., et al.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. In: Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1724–1734 (2014)
14. Chatterjee, R., Farajian, M.A., Conforti, C., Jalalvand, S., Balaraman, V., Di Gangi, M.A., Ataman, D., Turchi, M., Negri, M., Federico, M.: FBK's neural machine translation systems for IWSLT 2016. In: Proceedings of 13th International Workshop on Spoken Language Translation (IWSLT 2016) (2016)
15. Seonwoo, M., Byunghan, L., Sungroh, Y.: Deep learning in bioinformatics. In: Briefings in Bioinformatics (2016)
16. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998)
17. Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., Kuksa, P.: Natural language processing (almost) from scratch. J. Mach. Learn. Res. **12**, 2493–2537 (2011)
18. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
19. Bridle, J.S.: Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In: Soulié, F.F., Hérault, J. (eds.) Neurocomputing, pp. 227–236. Springer, Heidelberg (1990)
20. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. CoRR, abs/1301.3781 (2013)

21. Pennington, J., Socher, R., Manning, C.D.: Glove: global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1532–1543, October 2014
22. Dos Santos, C.N., Zadrozny, B.: Learning character-level representations for part-of-speech tagging. In: Proceedings of the 31st International Conference on Machine Learning (ICML), pp. 1818–1826 (2014)
23. Rizzo, R., Fiannaca, A., La Rosa, M., Urso, A.: A deep learning approach to dna sequence classification. In: Angelini, C., Rancoita, P.M.V., Rovetta, S. (eds.) CIBB 2015. LNCS, vol. 9874, pp. 129–140. Springer, Heidelberg (2016). doi:10.1007/978-3-319-44332-4_10
24. Caruana, R.: Multi-task learning: a knowledge-based source of inductive bias. Mach. Learn. **28**, 41–75 (1997)
25. Drancourt, M., Berger, P., Raoult, D.: Systematic 16S rRNA gene sequencing of atypical clinical isolates identified 27 new bacterial species associated with humans. J. Clin. Microbiol. **42**(5), 2197–2202 (2004)
26. https://rdp.cme.msu.edu/