Binary Classification of Proteins by a Machine Learning Approach

Damiano Perri[1(✉)], Marco Simonetti[1], Andrea Lombardi[2], Noelia Faginas-Lago[2], and Osvaldo Gervasi[3]

[1] Department of Mathematics and Computer Science, University of Florence, Florence, Italy
damiano.perri@gmail.com
[2] Department of Chemistry, Biology and Biotechnology, University of Perugia, Perugia, Italy
[3] Department of Mathematics and Computer Science, University of Perugia, Perugia, Italy

Abstract. In this work we present a system based on a Deep Learning approach, by using a Convolutional Neural Network, capable of classifying protein chains of amino acids based on the protein description contained in the Protein Data Bank. Each protein is fully described in its chemical-physical-geometric properties in a file in XML format. The aim of the work is to design a prototypical Deep Learning machinery for the collection and management of a huge amount of data and to validate it through its application to the classification of a sequences of amino acids. We envisage applying the described approach to more general classification problems in biomolecules, related to structural properties and similarities.

Keywords: Machine Learning · Computational chemistry · Protein Data Bank

1 Introduction

Proteins in Nature exhibit a very complicate relationship between their complex structures, with considerable differences from a chemical, physical and geometric point of view and their biological functions. The theoretical and computational study of proteins structure and function, as well as of nucleic acids, lipid membranes and other biosystems, is mainly carried on by Molecular Dynamics (MD) simulations, grounded upon classical mechanics and Force Fields. MD is used to sample the system phase space and to capture the relevant dynamical processes of proteins across different timescales. Simulations can be carried out at different levels of details. These can be atomistic simulations, where each atom is followed in detail, or can be based on coarse grained models, where group of atoms are replaced by pseudo-atoms and a reduced number of degrees of freedom allows to model biological phenomena accessing much longer time scales.

© Springer Nature Switzerland AG 2020
O. Gervasi et al. (Eds.): ICCSA 2020, LNCS 12255, pp. 549–558, 2020.
https://doi.org/10.1007/978-3-030-58820-5_41

In spite of continuous progress and increasing availability of High Performance Computing resources, many aspects in the dynamics and structure modelling still remain problematic, due to the inherent computational complexity of proteins and other biomolecules. These are related to (i) very high computational demand and sampling limitations and ii) limited force field accuracy, (iii) search for main stable structure and reactive or isomerization pathways, (iv) multiscale nature of dynamics.

In the last few years, the recourse to Machine Learning (ML) applications has become widespread in molecular dynamics. Particularly, approaches based on a variant of ML called deep neural networks [1,2] are becoming broadly popular. These can be applied to the many classification problems that, generally speaking, occur in the theoretical and computational modelling of proteins and other biomolecules. The growing amount of both experimental and theoretical data, available in databases and repositories, makes it increasingly feasible an efficient training of neural networks.

In this work, we have posed the basic problem of classifying as "real" a protein given its amino acid sequence, using a Deep Learning approach.

The choice of the network to process the sequences fell on Convolutional Neural Network (CNN). A choice in some ways less obvious than a Recurrent Neural Network (RNN) [3], given the nature and form of the data to be processed, but one that has revealed to be able to return results that are quite satisfactory.

In addition, CNNs can be easily implemented even on hardware that is not particularly performing, with the advantage of enabling to use the model developed on a plurality of platforms [4] and even on dedicated boards [5]. The cnn are mainly used for image analysis, they allow to extract features that are then used to correctly classify objects, people or things [6,7]. In this article we explain our approach to the problem using CNN 1D, i.e. one-dimensional.

The data set to train our network was taken from The Protein Data Bank (PDB) [8], that is a free access archive containing 3D structure data of proteins and nucleic acids.

2 The Architecture of the System

The system must be able to correctly classify a sequence of amino acids, telling whether it represents a "possible" real protein. To get to this, we decided to use a less conventional approach, but which seemed to us very promising, based on a CNN, a kind of networks particularly appreciated in the recognition of images and characteristics from two-dimensional and three-dimensional objects.

2.1 Data Extraction and Processing

Our work was developed entirely in Python working in the development environment provided by Google, called Colab. We divided the problem into a set of sub-problems. First of all we downloaded all the protein database provided by Protein Data Bank. The dataset is downloadable in two different formats:

PDB and XML. Between the two formats the information content is the same, the only difference being only the data format to store with the information. The PDB database is composed of hundreds of thousands of proteins all entirely described by single files in different formats: our choice has been placed on an open format like XML because it is much easier to be read and analyzed through the libraries made available by Python. The first problem we had to solve was related to the size of the dataset. In total there are 160,797 files in "xml.gz" compressed format for a total size of 57.6 GB. Once extracted the archive takes up more than 2 TB of disk space due to the huge amount of data it contains. First we created a script that solves the problem of rearranging the dataset in order to be conveniently handled by our tools. In practice, it outputs a single compressed file in which the proteins are arranged as a list. Each protein in the list is represented by a sequence of amino acids. In this way we were able to obtain a single file named "dataset.csv.gz" with a size of 7 GB, where just the relevant information was extracted, ready to be used for the next steps. Each single protein was entirely mapped and unequivocally encoded as a very precise sequence of numbers, consisting of a dictionary of type "amino acid's name: positive integer number".

A first analysis of the data showed that the amino acids present in the dataset were 52, but only a part of them (23), was really present in the whole set of proteins examined, containing 105,123 instances.

In addition, the sequences had variable lengths (defined as the number of amino acids in the chain) for the various molecules, with a distribution which is shown in the table below:

Length of molecules' sequences		
Chain length (interval)	Number of molecules	Number of molecules %
1 .. 9	96	0.09
10 .. 99	3,149	3.00
100 .. 999	83,526	79.46
1,000 .. 1,500	9,107	8.66
1,501 .. 9,999	9,117	8.67
10,000 .. 99,999	127	0.12
100,000 .. 1,000,000	1	0.00

Since the sequence to be fed to the network must have a predetermined fixed length, we have chosen to consider only those proteins with a length of less than 1,500, which make up about 91.21% of the entire set of sequences.

Since a protein in its constitutive unfolded sequence can be passed from left to right or vice versa, the dataset has been "augmented", by inserting the same sequences of proteins already present, but reversed. This allowed to obtain a dataset of real proteins with double size respect of the original one.

The second step for the preparation of the dataset was the generation of the fake samples (FALSE samples), in equal number to the real samples (TRUE samples).

Initially, trivial cases were inserted, as whole sequences of length 1,500 of the same amino acid repeated more times, obtaining 23 new false proteins. Subsequently, the generation level became more sophisticated: in each true protein, a fragment at a random position of the protein sequence was replaced by another one, giving place to a sequence mutation; each new fragment had a random length between 5 and 7% of the entire protein sequence length and contained amino acids randomly taken from the dictionary of all possible amino acids in our real cases (Fig. 1).

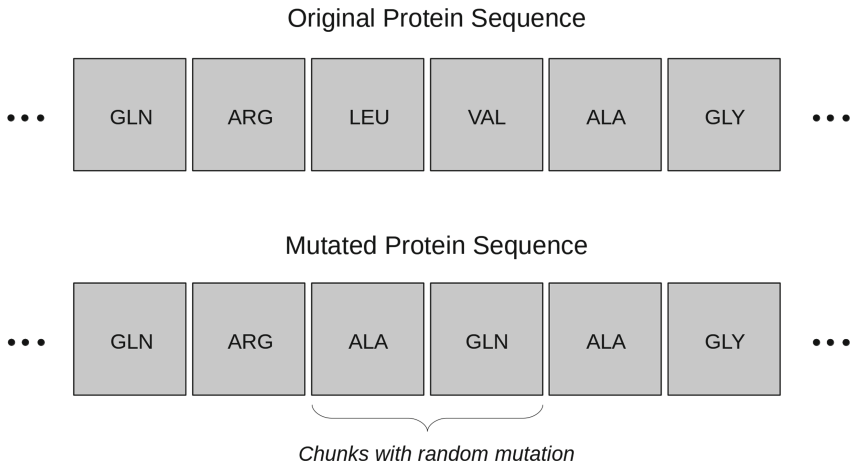

Fig. 1. False chunks in sequence

2.2 Model Construction and Validation

During the working phases we tested several networks built with different parameters and layers. A simple model and at the same time satisfactory for the quality of the results, was the one represented in Fig. 2 and Fig. 3, characterized as follows:

- Embedding Layer: to create the weight matrix and to index it, according to the inputs which should be lists of positive integers (encoded amino acids)
- Convolution Layer: these are our first two layers that define a feature detector with a kernel equal to 3.
- Pooling Layer: with a size of the max pooling windows of 5.
- Convolution Layer: another layer that define a feature detector with a kernel equal to 5 to refine outcomes.

- MaxPooling Layer: to reduce the complexity of the output and prevent overfitting of the data.
- The final dense layer: this is a fully connected layer with SIGMOID activation; this layer will reduce the vector dimensions to a binary vector since we have 2 classes that we want to predict.

The model is extremely light, using a total of only 13,988 parameters.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 1500, 3)           195
_________________________________________________________________
conv1d (Conv1D)              (None, 1498, 32)          320
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 1496, 32)          3104
_________________________________________________________________
max_pooling1d (MaxPooling1D) (None, 299, 32)           0
_________________________________________________________________
conv1d_2 (Conv1D)            (None, 295, 64)           10304
_________________________________________________________________
global_max_pooling1d (Global (None, 64)                0
_________________________________________________________________
dense (Dense)                (None, 1)                 65
=================================================================
Total params: 13,988
Trainable params: 13,988
Non-trainable params: 0
```

Fig. 2. Structure of the network model

In the section where the model is compiled we have chosen the "binary crossentropy" [9,10] as a feedback signal for learning the weight tensors to be minimized, as an optimizer to rule the gradient descent the "Adadelta optimizer" [11], with a fixed learning-rate equal to 1.0 and rho equal to 0.95, and as our metric the accuracy of the model.

The network was let to iterate on the training data (306,812 samples) in mini-batches of 50 samples for 50 epochs.

In addition, we have inserted checkpoints to the simulation, so that we can save the best weight configuration of the model.

2.3 Analysis of Results

The results obtained are very encouraging, having achieved a score of 95.6% on the accuracy of the tests.

In Fig. 4 and Fig. 5 it is shown the plot the loss and accuracy of the model, as a function of the various periods (epochs) of the simulation.

Despite the presence of sudden and rare spikes in the graphs, essentially due to a temporary displacement of the trajectory from the set of solutions by

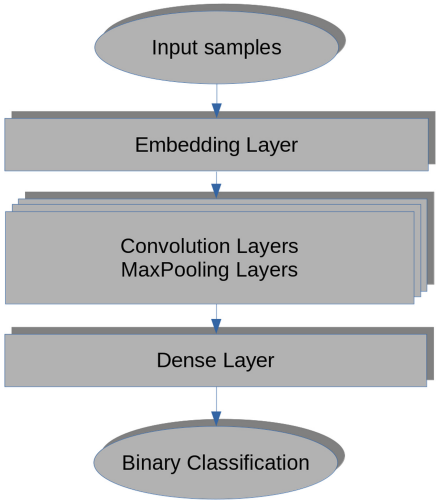

Fig. 3. Flow Diagram of the network model

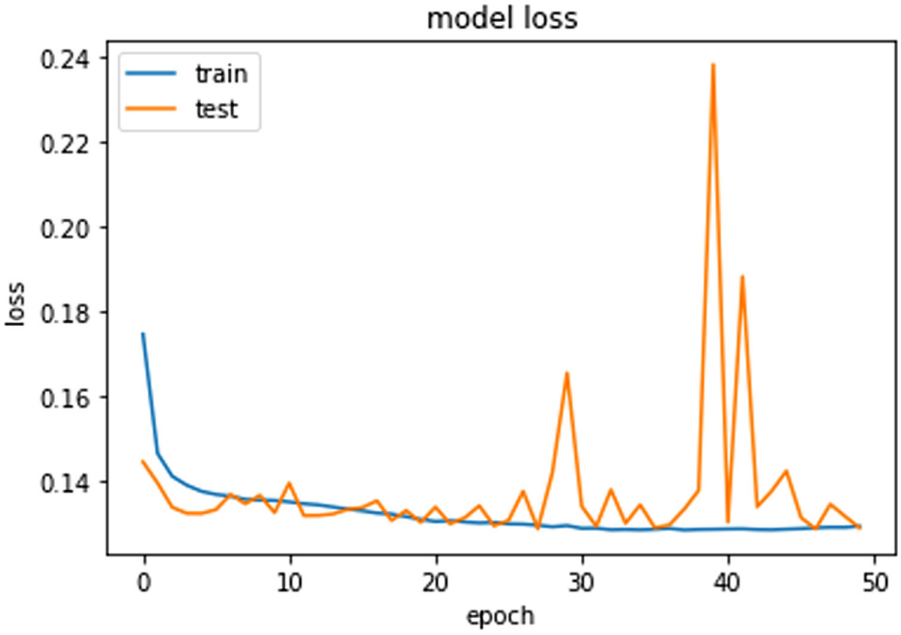

Fig. 4. Loss and validation loss for the model

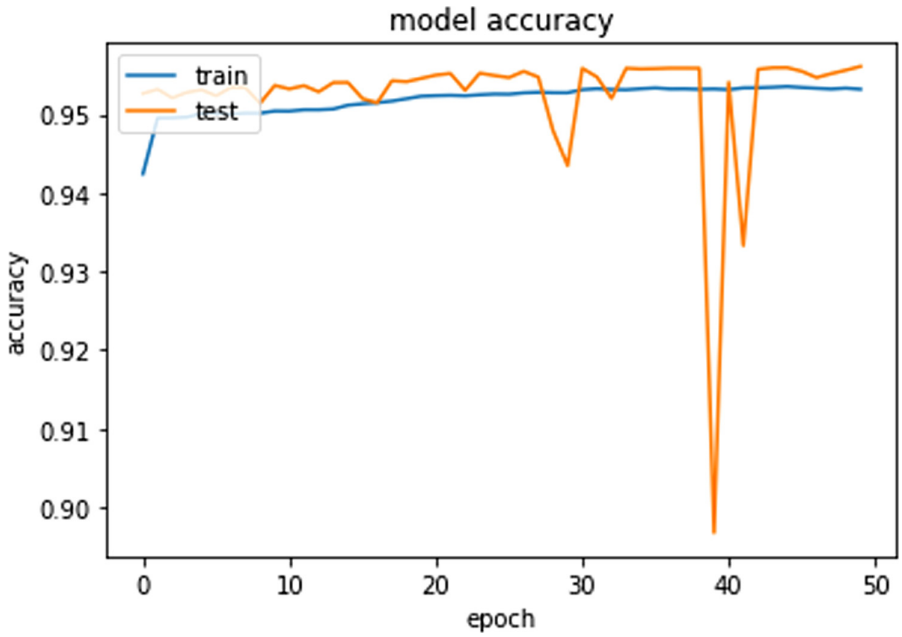

Fig. 5. Accuracy and validation accuracy for the model

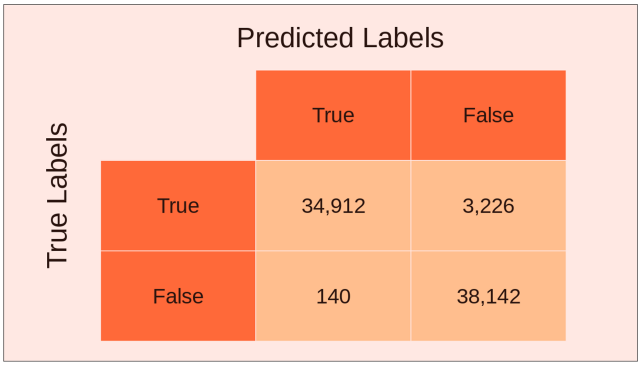

Fig. 6. Confusion matrix of results

the search algorithm, it can be seen that asymptotically there is a converging attractor around which our model is stationed.

The Confusion Matrix [12] indicates that the model is well performing, while highlighting rooms for improvement, especially in the phase of choosing the optimizer and modulating the hyper-parameters, see Fig. 6 and Fig. 7.

The apparent high number of "true negatives" depends on the phenomenon of training data padding, necessary for neural networks of this type, which require input sequences of a predetermined length.

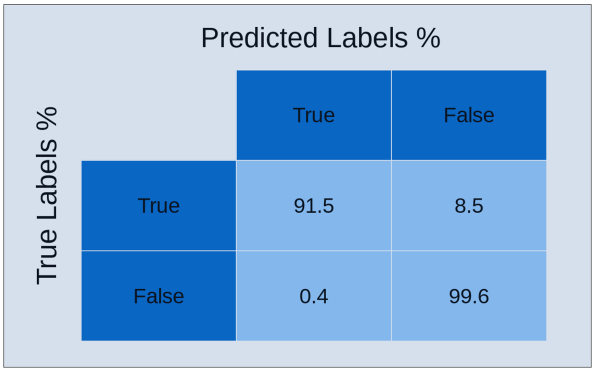

Fig. 7. Confusion matrix of results (%)

3 Further Development of the CNN Approach to Proteins as a Tool for Chemical Modelling

A further development of this work is to extend the Deep Learning model to the analysis of more subtle properties of the aminoacid sequences of the proteins and, above all, to classification of properties of the protein molecular structure and energy. General possible relationships between sequence, structure and function of proteins are of particular interest.

There is no obvious way how to proceed towards the objective of feeding the CNN with different kind of protein properties, but next work clearly would imply at least an adaptation or even a reshaping of the CNN input layer, while the preparatory phase of the organization of data can be retained.

We can now anticipate, without going into details, some of the possible ways to simply make next moves, according to some directions as follows:

- adding more view to the statistical analysis of sequences, recasting the learning as directed also by occurrence and recurrence of single aminoacids in the protein sequences and at which position
- adding to the sequences, here represented as a series of integer numbers, some physically meaningful parameter, such as deformation indexes [13] for the aminoacids in the chain or order parameters such as the number of hydrogen bonding interactions
- adding to the sequences parameters describing the global minimum energy structure corresponding to the protein native status, looking for patterns and regularities.

4 Conclusions and Future Works

Despite the very promising results shown by the simulations, we believe that there is considerable room for improvement both in the selection of the system's hyper-parameters and in the preparation and management of the data needed to learn the machine.

Our idea is to continue to explore the possibility of using CNN rather than RNN, as we consider it advantageous to use light models that can be easily implemented on machines with more modest hardware equipment.

Furthermore, it is important to limit the use of training data padding to decrease the number of "true negatives": a working hypothesis could be the segmentation of the sequences, hypotheses on which we are currently working.

Acknowledgments. A.L. and N.F.L. thank the Dipartimento di Chimica, Biologia e Biotecnologie dell'Università di Perugia (FRB, Fondo per la Ricerca di Base 2017) and the Italian MIUR and the University of Perugia for the financial support of the AMIS project through the program "Dipartimenti di Eccellenza". A. L. acknowledges financial support from MIUR PRIN 2015 (contract 2015F59J3R_002). A.L. thanks the OU Supercomputing Center for Education & Research (OSCER) at the University of Oklahoma, for allocated computing time.

References

1. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**, 436–444 (2015)
2. Schmidhuber, J.: Deep learning in neural networks: an overview. Neural Netw. **61**, 85–117 (2015)
3. Liu, X.: Deep recurrent neural network for protein function prediction from sequence (2017). arXiv: 1701.08318 [q-bio.QM]
4. Sze, V., et al.: Hardware for machine learning: challenges and opportunities. In: IEEE Custom Integrated Circuits Conference (CICC), pp. 1–8. IEEE (2017)
5. Zhang, C., et al.: Optimizing FPGA-based accelerator design for deep convolutional neural networks. In: Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, pp. 161–170 (2015)
6. Biondi, G., Franzoni, V., Gervasi, O., Perri, D.: An approach for improving automatic mouth emotion recognition. In: Misra, S., et al. (eds.) ICCSA 2019. LNCS, vol. 11619, pp. 649–664. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-24289-3_48
7. Perri, D., Sylos Labini, P., Gervasi, O., Tasso, S., Vella, F.: Towards a learning-based performance modeling for accelerating deep neural networks. In: Misra, S., et al. (eds.) ICCSA 2019. LNCS, vol. 11619, pp. 665–676. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-24289-3_49
8. Berman, H.M., et al.: The protein data bank. Nucleic Acids Res. **28**, 235–242 (2000). http://www.rcsb.org/
9. Rubinstein, R.Y., Kroese, D.P.: The Cross-entropy Method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation and Machine Learning. Springer, New York (2013)
10. De Boer, P.-T., et al.: A tutorial on the cross-entropy method. Ann. Oper. Res. **134**(1), 19–67 (2005). https://doi.org/10.1007/s10479-005-5724-z

11. Zeiler, M.D.: ADADELTA: an adaptive learning rate method. arXiv preprint arXiv:1212.5701 (2012)
12. Visa, S., et al.: Confusion matrix-based feature selection. MAICS **710**, 120–127 (2011)
13. Aquilanti, V., Lombardi, A., Yurtsever, E.: Global view of classical clusters: the hyperspherical approach to structure and dynamics. Phys. Chem. Chem. Phys. **4**, 5040–5051 (2002)