Bioinformatics, 37(23), 2021, 4567–4568
doi: 10.1093/bioinformatics/btab687
Advance Access Publication Date: 2 October 2021
Applications Note

Sequence analysis

CNN-PepPred: an open-source tool to create convolutional NN models for the discovery of patterns in peptide sets—application to peptide–MHC class II binding prediction

Valentin Junet [1,2] **and Xavier Daura** [2,3,4,*]

[1]Anaxomics Biotech SL, Barcelona 08008, Spain, [2]Institute of Biotechnology and Biomedicine, Universitat Autònoma de Barcelona, Barcelona 08193, Spain, [3]Catalan Institution for Research and Advanced Studies (ICREA), Barcelona 08010, Spain and [4]Biomedical Research Networking Center in Bioengineering, Biomaterials and Nanomedicine (CIBER-BBN), Madrid 28029, Spain

*To whom correspondence should be addressed.
Associate Editor: Tobias Marschall

Received on May 13, 2021; revised on August 2, 2021; editorial decision on September 24, 2021; accepted on September 28, 2021

Abstract

Summary: The ability to unveil binding patterns in peptide sets has important applications in several biomedical areas, including the development of vaccines. We present an open-source tool, CNN-PepPred, that uses convolutional neural networks to discover such patterns, along with its application to peptide–HLA class II binding prediction. The tool can be used locally on different operating systems, with CPUs or GPUs, to train, evaluate, apply and visualize models.

Availability and implementation: CNN-PepPred is freely available as a Python tool with a detailed User's Guide at https://github.com/ComputBioI-IBB/CNN-PepPred. The site includes the peptide sets used in this study, extracted from IEDB (www.iedb.org).

Contact: xavier.daura@uab.cat

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

The prediction of antigenic determinants—epitopes—of specific proteins or full proteomes is a key aspect of immunoinformatics, with a wide-range of applications from the study of autoimmunity to the development of cancer immunotherapies and therapeutic as well as prophylactic vaccines. For MHC epitopes in particular, experiments coupling natural-ligand elution to mass spectrometry identification (Caron *et al.*, 2015) have led in recent years to an explosion in the amount of data available on MHC peptide recognition, enabling data-driven approaches to *in silico* epitope prediction (Wang *et al.*, 2010).

The state-of-the-art in the prediction of peptide–MHC class II binding is arguably represented by the NetMHCII family of algorithms (Andreatta *et al.*, 2015). Their core is a neural-network-based algorithm called NNAlign (Nielsen and Lund, 2009). NetMHCII features an allele-specific algorithm, while NetMHCIIpan deploys a universal network for the prediction of binding to any MHC class II allele with known protein sequence (Jensen *et al.*, 2018). The functionality of both tools is constrained to the application of pre-trained models. NNAlign (Nielsen and Andreatta, 2017), on the other hand, offers the possibility of training new models. These tools, however, are only available as on-line platforms or executables and there exists, to our knowledge, no reference providing information on the implementation details of the neural network behind them, in a manner that would make it reproducible. Recently, these algorithms have taken a new research direction to include multiple-allele datasets (MA) (Alvarez *et al.*, 2019; Reynisson *et al.*, 2020). Other groups have also developed their own methods to include MA data (Chen *et al.*, 2019; Racle *et al.*, 2019), but are also only available for application purposes in the form of a platform or an executable. While all these tools are being successfully used in multiple applications and are amenable to non-expert use, we believe that the lack of clear implementation details and open code for the core predictive algorithm hinder their use by bioinformatics researchers who need a tool that can be fully integrated in larger codes and possibly modified to fit particular needs in different areas of peptide recognition.

It is indeed to satisfy our own needs in this field that we developed CNN-PepPred, a new tool for the discovery of patterns in peptide sets. The tool was developed to conveniently fit general purpose needs such as the training of models, their application to new data, their evaluation by cross-validation and gaining visual insight on the training process. The software is accompanied by a detailed User's

Guide and has been tested on Linux and Windows, using CPUs and GPUs. In the following section, we provide a concise description of CNN-PepPred. We refer the reader to the User's Guide for a detailed description.

2 Description

CNN-PepPred is based on Convolutional Neural Networks (CNN) and written in Python. Supplementary Figure S1B (Supplementary Information) illustrates the architecture of the CNN, implemented with the open-source libraries Keras (github.com/fchollet/keras) and TensorFlow (tensorflow.org). The peptides are first encoded using the blosum62 similarity matrix (Henikoff and Henikoff, 1992). In the schema, this corresponds to filling the input table with the blosum62 similarity between the residue (row) and the amino acid type (column). To deal with the different lengths in the peptide sets, the symbol '-' representing the absence of further residues was introduced and added at the end of all peptides in the training set so that their lengths match. The convolutional layer applies different filters (only one is illustrated in the schema) to all overlapping l-mers in the peptide, where l is a user defined parameter, typically $l = 9$ in applications related to MHC class II. The ReLu activation layer sets all negative outputs to zero. The MaxPool layer will select the highest outputs and the dense layer will connect the results to generate the output, i.e. the predicted binding score. The final model is an equally weighted ensemble of CNNs, resulting from different numbers of filters in the convolutional layer times the number of initial configurations.

Both NNAlign and CNN-PepPred use a blosum encoding. The main difference between the two methods is that while NNAlign uses a two-step procedure that identifies a core l-mer within a peptide and then applies a network weight configuration to the core (and flanking regions) to generate the output, CNN-PepPred uses convolution to slide through all possible overlapping l-mers and subsequent layers to activate them and generate the output. In other words, CNN-PepPred does not select a core binder but makes full use of all overlapping l-mers within the peptide, and what may be considered the core binder is the overlapping l-mer that is most activated during the feed-forward pass.

The tool contains many functionalities that can be conveniently called by filling a simple template. Users can train their own models, evaluate them through cross-validation, save them and apply them to new data (Supplementary Fig. S1A). It is also possible to visualize the binding motif of a trained model with *Logomaker* (Tareen and Kinney, 2020). For more insight on the feed-forward pass on a selected peptide set, the user can also print all layers and corresponding outputs of the trained model. In addition, advanced users may incorporate the CNNPepPred class, providing all the functionality, to other Python developments.

When applied to HLA class II binding, using 51 alleles with data curated by the IEDB (www.iedb.org), our tool showed significant improvement over NNAlign-2.1 (Nielsen and Andreatta, 2017) in predictive performance (see Supplementary Information). We used the T-cell epitope benchmark dataset from Jensen *et al.* (2018) as an independent evaluation set for the models trained with the data retrieved from IEDB. CNN-PepPred was found to perform better on average than NetMHCIIpan-3.2 (Jensen *et al.*, 2018), although the results were overall similar as shown in Appendix D of the User's guide. Appendix C contains also a cross-validation comparison with the data from NetMHCII-2.3 and NetMHCIIpan-3.2, giving overall similar results. Note that the latter two tools cannot be used themselves to train new models.

In Supplementary Appendices SB.2 and SC.2, we compared computation times for CNN-PepPred, NNAlign and NetMHCIIpan, showing that, for the specific CPU and GPU used, CNN-PepPred compares well to NNAlign for training and clearly outperforms NetMHCIIpan for prediction, NNAlign being the faster prediction tool.

The github repository contains two main Python scripts, one with the CNNPepPred class and one that uses this class following a template filled by the user. The class was written so that it could be also used independently from the template. The User's Guide contains a full description of the class, its parameters, the template, the model, the visualization, the IEDB data and the benchmarking.

The core training algorithm of CNN-PepPred consists only of a few Keras lines of code. It can, therefore, be also used independently from the tool, as a starting point for further analysis and/or generalizations that could fit other types of peptide sets. Thanks to the ease of use of libraries like Keras and TensorFlow, the implementation does not require the developer to fully implement specific feed-forward/back-propagation passes that would fit one type of data. The CNN-PepPred code is provided under an open-source license.

Acknowledgement

The authors thank Judith Farrés and José M Mas from Anaxomics Biotech SL for their support and helpful discussions.

Funding

This project received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie [765158] (COSMIC; www.cosmic-h2020.eu) and from the Spanish Ministry for Science, Innovation and Universities [PID2019-111364RB-I00].

Conflict of Interest: none declared.

References

Alvarez,B. *et al.* (2019) NNAlign_MA; MHC peptidome deconvolution for accurate MHC binding motif characterization and improved T-cell epitope predictions. *Mol. Cell. Proteom.*, **18**, 2459–2477.

Andreatta,M. *et al.* (2015) Accurate pan-specific prediction of peptide–MHC class II binding affinity with improved binding core identification. *Immunogenetics*, **67**, 641–650.

Caron,E. *et al.* (2015) Analysis of major histocompatibility complex (MHC) immunopeptidomes using mass spectrometry. *Mol. Cell. Proteom.*, **14**, 3105–3117.

Chen,B. *et al.* (2019) Predicting HLA class II antigen presentation through integrated deep learning. *Nat. Biotechnol*, **37**, 1332–1343.

Henikoff,S. and Henikoff,J.G. (1992) Amino acid substitution matrices from protein blocks. *Proc. Natl. Acad. Sci. USA*, **89**, 10915–10919.

Jensen,K.K. *et al.* (2018) Improved methods for predicting peptide binding affinity to MHC class II molecules. *Immunology*, **154**, 394–406.

Nielsen,M. and Andreatta,M. (2017) NNAlign: a platform to construct and evaluate artificial neural network models of receptor-ligand interactions. *Nucleic Acids Res.*, **45**, W344–W349.

Nielsen,M. and Lund,O. (2009) NN-align. An artificial neural network-based alignment algorithm for MHC class II peptide binding prediction. *BMC Bioinformatics*, **10**, 296.

Racle,J. *et al.* (2019) Robust prediction of HLA class II epitopes by deep motif deconvolution of immunopeptidomes. *Nat. Biotechnol.*, **37**, 1283–1286.

Reynisson,B. *et al.* (2020) NetMHCpan-4.1 and NetMHCIIpan-4.0: improved predictions of MHC antigen presentation by concurrent motif deconvolution and integration of MS MHC eluted ligand data. *Nucleic Acids Res.*, **48**, W449–W454.

Tareen,A. and Kinney,J.B. (2020) Logomaker: beautiful sequence logos in Python. *Bioinformatics*, **36**, 2272–2274.

Wang,P. *et al.* (2010) Peptide binding predictions for HLA DR, DP and DQ molecules. *BMC Bioinformatics*, **11**, 568.