pysster: Learning Sequence and Structure Motifs in DNA and RNA Sequences using Convolutional Neural Networks

Stefan Budach [1,*] and Annalisa Marsico [1,2,*]

[1] RNA Bioinformatics, Max Planck Institute for Molecular Genetics, Berlin, 14195, Germany and
[2] Department of Mathematics and Computer Science, Free University Berlin, Berlin, 14195, Germany.

* To whom correspondence should be addressed.

Abstract

Summary: Convolutional neural networks have been shown to perform exceptionally well in a variety of tasks, including biological sequence classification. Available implementations, however, are usually optimized for a particular task and difficult to reuse. To enable researchers to utilize these networks more easily we implemented pysster, a Python package for training and interpretation of convolutional neural networks. The package can be applied to both DNA and RNA to classify sets of sequences by learning sequence and secondary structure motifs. It offers an automated hyper-parameter optimization and options to visualize learned motifs along with information about their positional and class enrichment. The package runs seamlessly on CPU and GPU and provides a simple interface to train and evaluate a network with a handful lines of code.
Availability: pysster is freely available at https://github.com/budach/pysster.
Contact: budach@molgen.mpg.de, marsico@molgen.mpg.de

1 Introduction

In recent years, deep convolutional neural networks (CNNs) have been shown to be an accurate method for biological sequence classification and sequence motif detection (Alipanahi *et al.*, 2015) (Kelley *et al.*, 2016) (Angermueller *et al.*, 2017). The increasing amount of sequence data and the rise of general-purpose computing on Graphics Processing Units (GPUs) have enabled CNNs to outperform other machine learning methods, such as random forests and support vector machines, in terms of both classification performance and runtime performance (Kelley *et al.*, 2016) (Angermueller *et al.*, 2017). While a number of publications have made use of CNNs on biological data and also published their source code and the resulting models (e.g. Alipanahi *et al.* (2015), Angermueller *et al.* (2017)), these implementations are usually tailored to a specific problem and therefore hard to reuse. Basset (Kelley *et al.*, 2016) provides a more general framework for training of CNNs on DNA sequences, but is not applicable to RNA sequence/secondary structure data and it also does not provide detailed motif interpretations, such as motif locations and class enrichment of motifs.

To address these issues and to enable researchers to utilize CNNs more easily, we implemented pysster, a python package for training and interpretation of convolutional neural network classifiers. Our package extends previous implementations and in addition to DNA sequences it can be applied to RNA sequence/secondary structure input to classify the input by learning sequence and secondary structure motifs. The package also provides information about the positional and class enrichment of learned motifs. Being easy to install and providing a simple programming interface we hope that our package enables more researchers to make effective use of CNNs in classifying large sets of biological sequences.

2 Implementation And Features

We implemented an established network architecture and multiple interpretation options as an easy-to-use python package. The basic architecture of the network consists of a variable number of convolutional and max-pooling layers followed by a variable number of dense layers (Figure 1A). These layers are interspersed by dropout layers after the input layer and after every max-pooling and dense layer. Using an automated grid search, the network can be tuned via a number of hyper-parameters, such as number of convolutional layers, number of kernels, length of kernels and dropout ratios.

The main features of the package are:

- applicable to DNA sequence and RNA sequence/structure input
- multi-class and single-label or multi-label classifications
- automated hyper-parameter tuning (grid search)
- learning of sequence/structure motifs for RNA input
- motif interpretation in terms of positional and class enrichment (Figure S1) and motif co-occurrence (Figure S2)
- visualization of all network layers using visualization by optimization (Olah *et al.*, 2017)
- seamless CPU and GPU computation by building on top of TensorFlow (Abadi *et al.*, 2016) and Keras (Chollet *et al.*, 2015)

The visualization of convolutional kernels from the first convolutional layer as sequence motifs is implemented analogous to previous methods (Alipanahi *et al.*, 2015). In brief, kernels are visualized by extracting subsequences of the length of the kernel from the input sequences if a subsequence leads to a kernel activation higher than a certain threshold (a maximum of one subsequence per input sequence). As all subsequences are

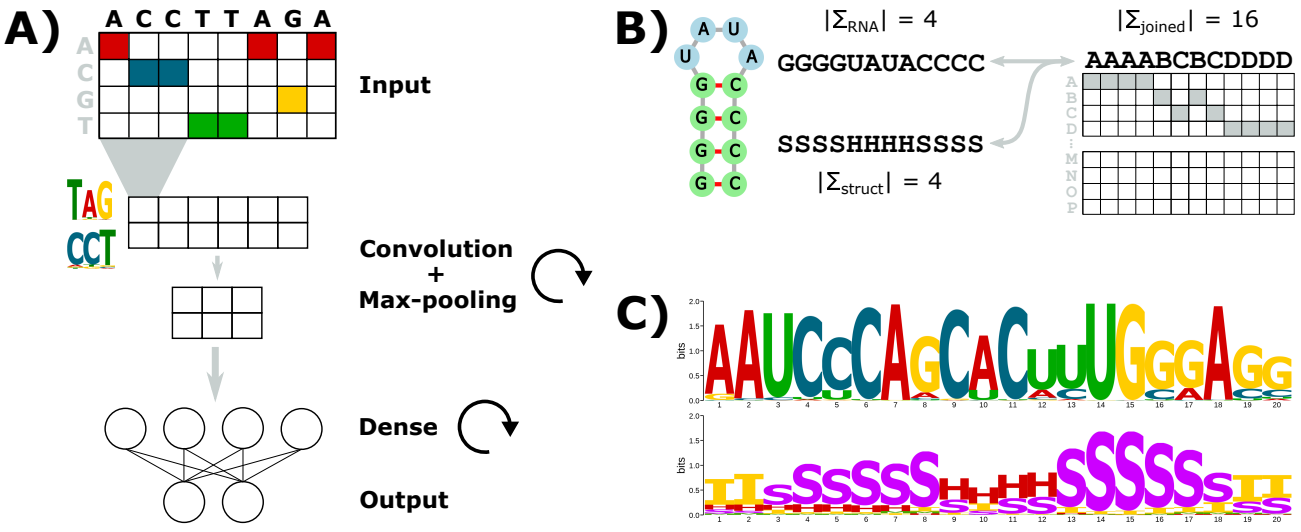

Fig. 1. A) The basic network architecture consists of a variable number of convolutional/max-pooling stacks followed by a variable number of dense layers interspersed by dropout layers. The network can be tuned via an extensive list of hyper-parameters (see documentation on github). The network input are one-hot encoded sequences and the network outputs predicted probabilities, indicating class membership. B) RNA sequence and structure input strings are encoded into a single string by combining the sequence alphabet and the secondary structure alphabet into an extended alphabet consisting of arbitrary characters. Subsequently, this string is one-hot encoded and used as the network input. C) For the motif interpretation the string over the arbitrary alphabet can be decoded into the two original strings to construct sequence logos for the original alphabets. The shown example motifs correspond to a kernel in the classification task of RNA A-to-I editing sites (see the tutorial workflow on github for detailed information).

of the same length, it is now possible to compute and visualize a position-weight matrix (PWM) in the usual way. To extend and facilitate motif interpretation, we also plot the positions of where these subsequences were extracted from, indicating positional enrichment. Doing this separately for every input class further shows class enrichment of motifs (Figure S1).

To be able to handle RNA sequence/secondary structure input while maintaining the mentioned visualization options, we encode the sequence string and the corresponding secondary structure string into a single new string using an extended alphabet. More precisely, all possible combinations of characters from the RNA alphabet of length 4 (A, C, G, U) and the annotated secondary structure alphabet of length 4 (H = hairpin, I = internal loop/bulge, M = multi loop, S = stem) can be uniquely represented by an alphabet of length 16 comprised of arbitrary characters (Figure 1B). The network is then trained on the new strings and after the subsequences for the motif visualizations have been extracted, as described earlier, the subsequences are decoded into the two original strings. This makes it possible to compute and visualize two PWMs, one for the RNA alphabet and one for the annotated secondary structure alphabet (Figure 1C). All other mentioned interpretation options are not affected by this procedure.

3 Case Study & Conclusion

Pysster is freely available at https://github.com/budach/pysster and includes an extensive documentation with detailed descriptions of its features. Finally, the documentation also includes two tutorials: 1) a tutorial showcasing the visualization of all network layers using visualization by optimization on an artificial data set and 2) a general workflow tutorial showcasing the main functionality of pysster on an RNA A-to-I editing data set (Picardi *et al.*, 2017) that was used to generate Figure 1C, Figure S1 and S2 and where we demonstrate that our CNN implementation is able to learn meaningful and biologically interpretable sequence/structure motifs.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., *et al.* (2016). Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*.

Alipanahi, B., Delong, A., Weirauch, M. T., and Frey, B. J. (2015). Predicting the sequence specificities of dna-and rna-binding proteins by deep learning. *Nature biotechnology*, **33**(8), 831–838.

Angermueller, C., Lee, H. J., Reik, W., and Stegle, O. (2017). Deepcpg: accurate prediction of single-cell dna methylation states using deep learning. *Genome Biology*, **18**(1), 67.

Chollet, F. *et al.* (2015). Keras. https://github.com/fchollet/keras.

Kelley, D. R., Snoek, J., and Rinn, J. L. (2016). Basset: learning the regulatory code of the accessible genome with deep convolutional neural networks. *Genome research*, **26**(7), 990–999.

Olah, C., Mordvintsev, A., and Schubert, L. (2017). Feature visualization. *Distill*. https://distill.pub/2017/feature-visualization.

Picardi, E., D'Erchia, A. M., Lo Giudice, C., and Pesole, G. (2017). Rediportal: a comprehensive database of a-to-i rna editing events in humans. *Nucleic acids research*, **45**(D1), D750–D757.

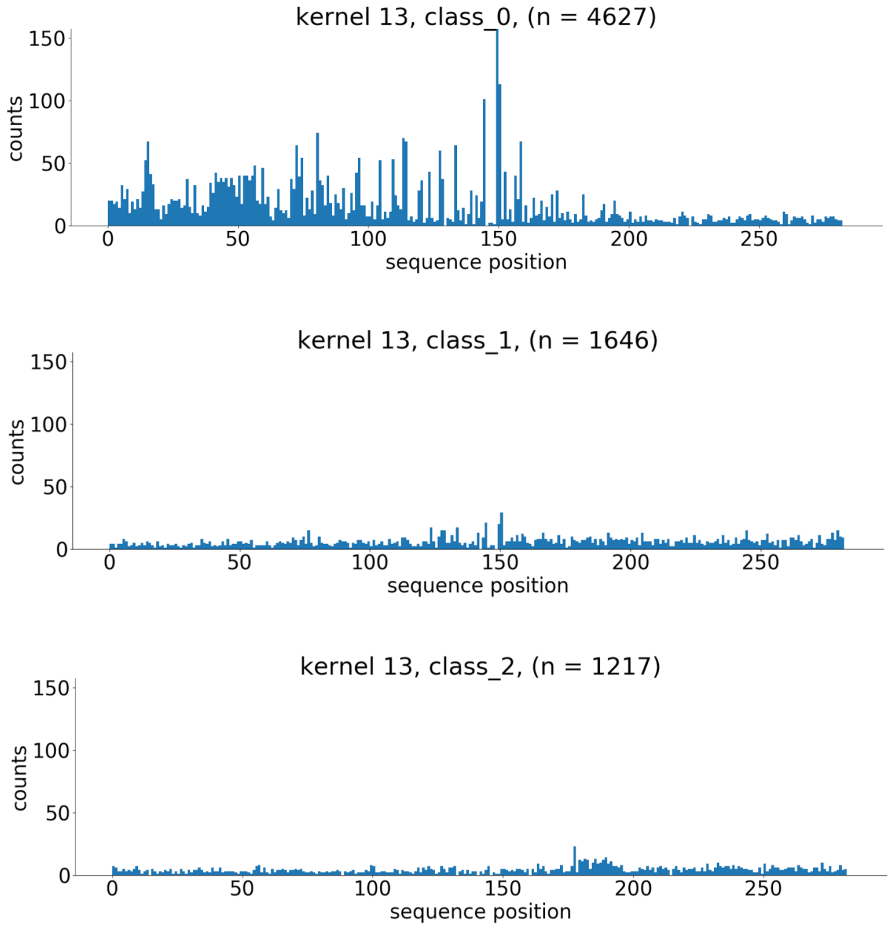

Figure S1. The histograms show the positional enrichment of a kernel (i.e. sequence positions at which subsequences leading to kernel activations higher than a threshold have been extracted from) for an RNA A-to-I editing classification data set (Picardi *et al*., 2017) consisting of 3 balanced classes and input sequences of length 300. The motifs corresponding to the kernel are shown in Figure 1C. The motifs are mainly found in the first class and preferentially start close to sequence position 150. More details about the data set and the biological interpretation for this particular kernel can be found in the example workflow tutorial at github.

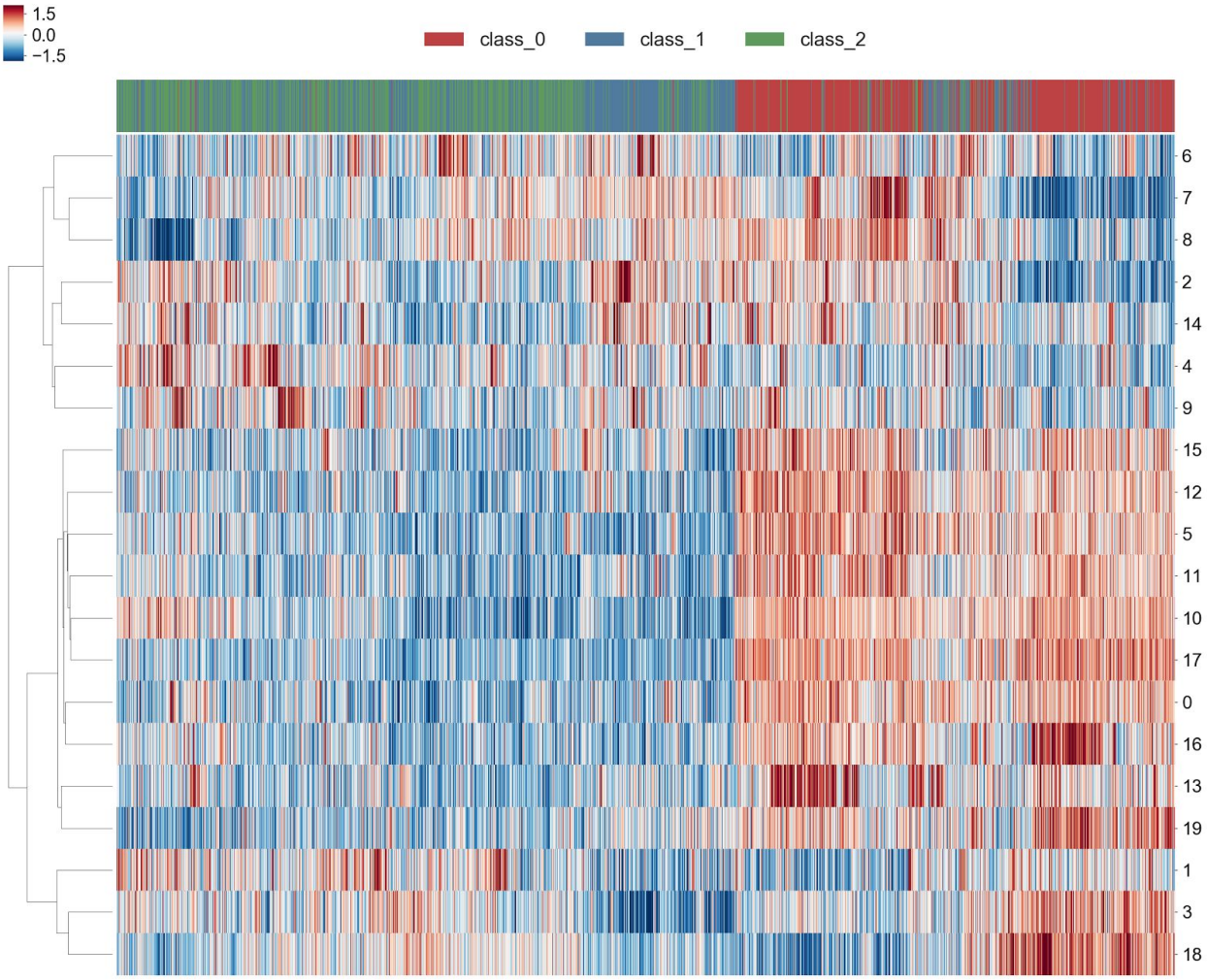

Figure S2. The heatmap shown here depicts a hierarchical clustering of both kernels (rows) and input sequences (columns) of a standardized matrix M where each cell M(i,j) represents the maximum activation value of kernel i for input sequence j. The resulting clustering is indicative of both class enrichment (i.e. kernels enriched or depleted in a certain class compared to the others), as well as motif co-occurrences. The heatmap shown here refers to the classification results on the RNA A-to-I editing data set showcased in the github tutorial. It is important to bear in mind that, besides putative motif co-occurrences, convolutional neural network kernels tend to learn strong motifs multiple times and hence, these tend to be clustered together. Nonetheless this visualization represents a valuable attempt to start looking for independent co-occurring motifs.