pysster: Classification Of Biological Sequences By Learning Sequence And Structure Motifs With 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 (CNNs) 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 CNNs on biological sequence data. Sequences are classified by learning sequence and structure motifs and the package offers an automated hyper-parameter optimization procedure 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. Using an RNA A-to-I editing data set and CLIP-seq binding site sequences we demonstrate that pysster classifies sequences with higher accuracy than other methods and is able to recover known sequence and structure motifs.
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, these implementations are usually hard to reuse (Alipanahi *et al.*, 2015) or tailored to a specific problem, such as prediction of DNA CpG methylation from single-cell data (Angermueller *et al.*, 2017). Basset (Kelley *et al.*, 2016) and iDeep (Pan and Shen, 2017) represent more general frameworks for training of CNNs on DNA and RNA data, respectively. However, they don't provide detailed motif interpretations, such as motif locations and class enrichment of motifs, and they are not able to learn structure motifs in the RNA case.

To address these issues and to enable researchers to easily utilize CNNs, we implemented pysster, a python package for training CNN classifiers on biological sequences. Supervised classification is enabled by the automatic detection of sequence motifs. Our package focuses on interpretability and extends previous implementations by providing information about the positional and class enrichment of learned motifs. Moreover, by incorporating structure information, e.g. in the form of secondary structure predictions for RNA sequences, it is possible to learn structure motifs corresponding to the sequence motifs. We demonstrate that our tool is able to learn well-known motifs for an RNA A-to-I editing data set and multiple CLIP-seq data sets and that it outperforms GraphProt (Maticzka *et al.*, 2014), a state of the art classifier for RNA sequences and structures, both in terms of classification and runtime performance. Providing a simple programming interface we hope that our package enables more researchers to make effective use of CNNs in classifying and interpreting 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:

- multi-class and single-label or multi-label classifications
- sensible default parameters and an optional hyper-parameter tuning
- learning of motifs + interpretation in terms of positional and class enrichment (Figure S1) and motif co-occurrence (Figure S2)
- support of input strings over user-defined alphabets (i.e. applicable to DNA, RNA and protein data)
- optional use of structure information, handcrafted features and recurrent layers
- visualization of all network layers using visualization by optimization (Olah *et al.*, 2017)

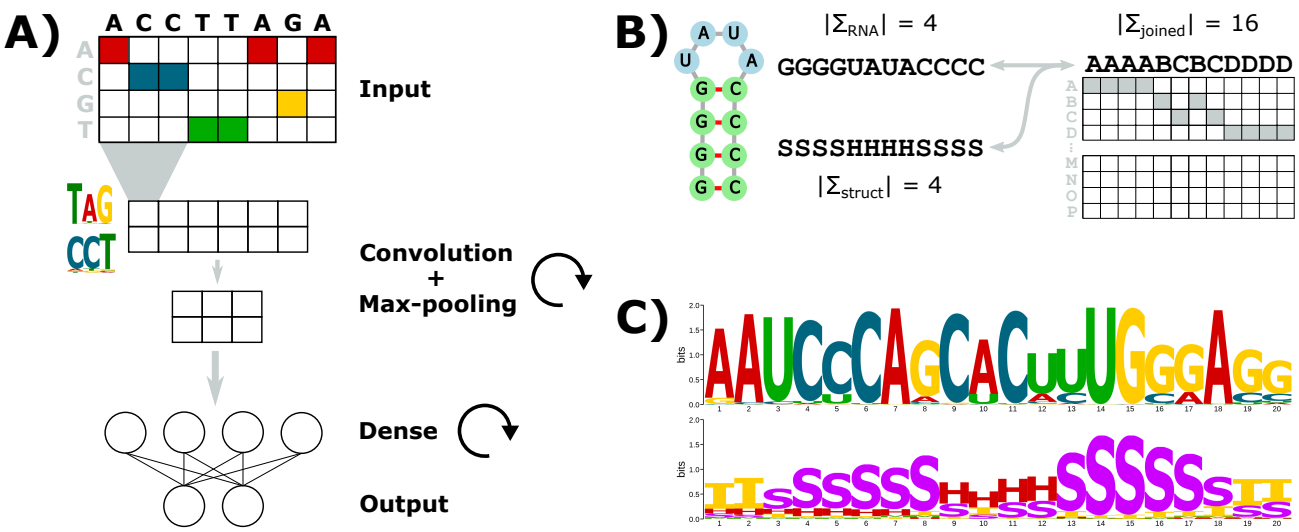


Fig. 1. A) The basic network architecture consists of a variable number of convolutional/max-pooling stacks followed by a variable number of dense layers interspersed by dropout layers. The network can be tuned via an extensive list of hyper-parameters. The network input are one-hot encoded sequences 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 motif corresponds to an ALU repeat motif found in the classification task of RNA A-to-I editing sites (see the tutorial workflow on github for detailed information).

- seamless CPU or GPU computation by building on top of TensorFlow (Abadi *et al.*, 2016) and Keras (Chollet *et al.*, 2015).

Structure information (e.g. for RNA in the form of dot-bracket strings or annotated dot-bracket strings) is incorporated into the network by encoding the given sequence string over an alphabet of size N and the corresponding structure string over an alphabet of size M into a single new string using an extended alphabet of size N*M (Figure 1B). The network is then trained on these new strings, which can be decoded back into the original strings after the training to enable the visualization of two motifs as position-weight matrices (Figure 1C). Detailed descriptions of the model and visualization options can be found in the tutorials and documentation.

3 Case Studies

Pysster is freely available at https://github.com/budach/pysster. Its documentation includes a workflow tutorial that showcases the main functionality on an RNA A-to-I editing data set (Picardi *et al.*, 2017). Editing sites are known to be enriched in repetitive ALU sequences and we show that we are able to classify the editing location with high accuracy and that we learn known ALU motifs (Figure 1C). Finally, we have trained our tool on sequences derived from CLIP-seq data for a number of RNA binding proteins with known binding site motifs (Table S1). Similar to GraphProt and ssHMM (Heller *et al.*, 2017), an unsupervised hidden Markov model-based approach for learning sequence/structure motifs, pysster can recover the known motifs, but outperforms GraphProt both in terms of classification and runtime performance.

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.

Heller, D., Krestel, R., Ohler, U., Vingron, M., and Marsico, A. (2017). sshmm: extracting intuitive sequence-structure motifs from high-throughput rna-binding protein data. *Nucleic acids research*, **45**(19), 11004–11018.

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.

Maticzka, D., Lange, S. J., Costa, F., and Backofen, R. (2014). Graphprot: modeling binding preferences of rna-binding proteins. *Genome biology*, **15**(1), R17.

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

Pan, X. and Shen, H.-B. (2017). Rna-protein binding motifs mining with a new hybrid deep learning based cross-domain knowledge integration approach. *BMC bioinformatics*, **18**(1), 136.

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.

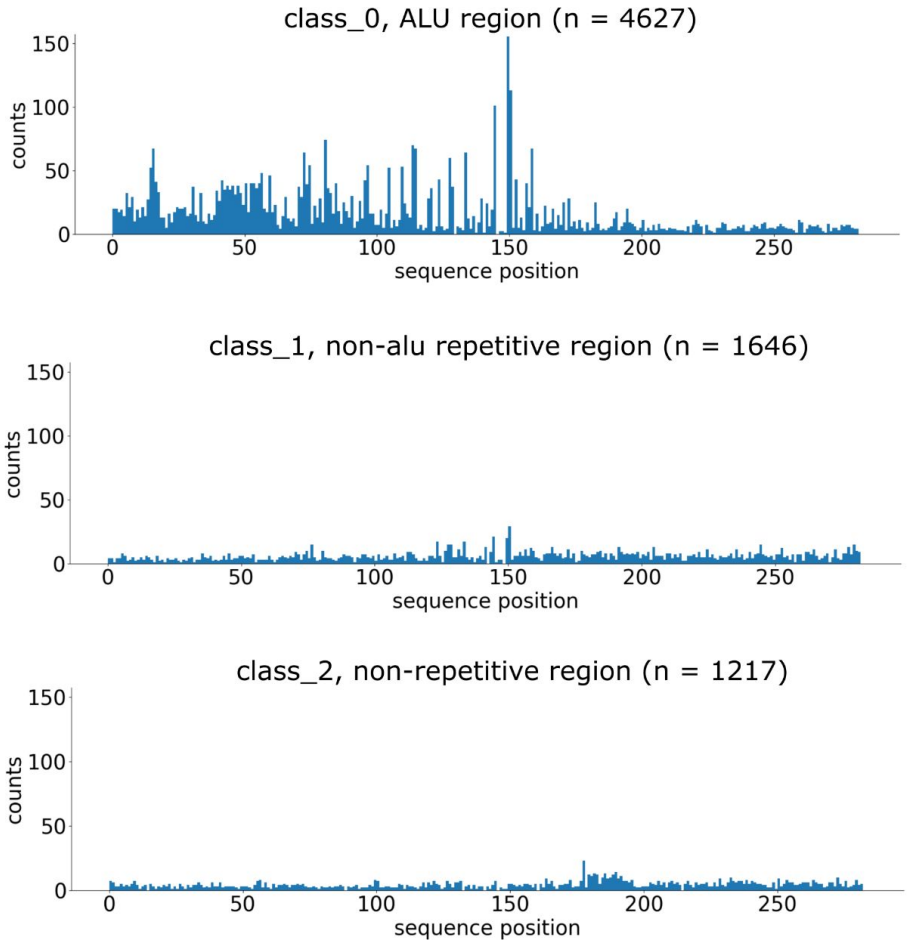


Figure S1. The histograms show the positional enrichment of a kernel (i.e. sequence positions at which subsequences leading to kernel activations higher than a threshold have been extracted from) for an RNA A-to-I editing classification data set consisting of three classes and input sequences of length 300. The motif corresponding to the kernel is shown in Figure 1C and is a known ALU repeat motif. The motif is therefore mainly found in the first class and preferentially starts 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.

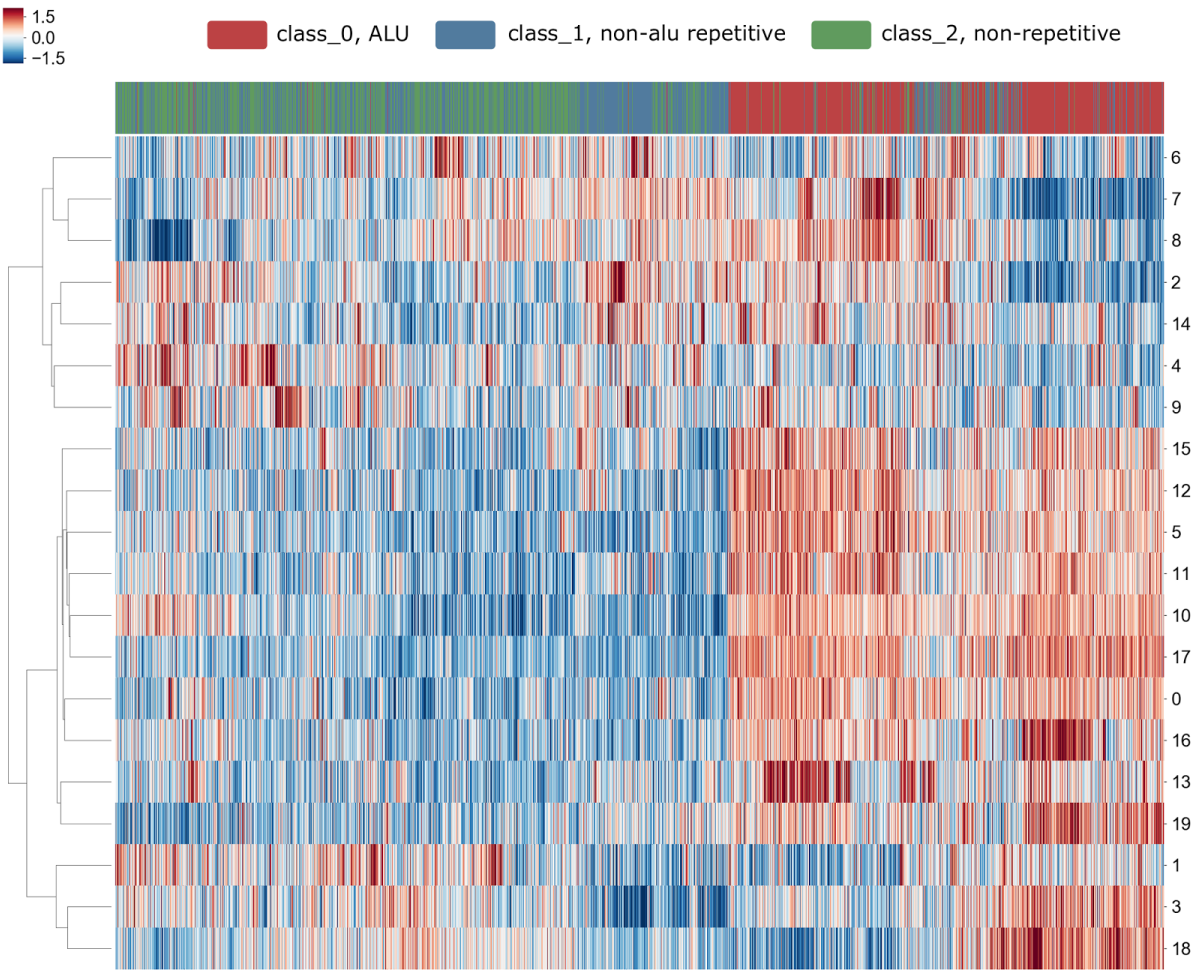
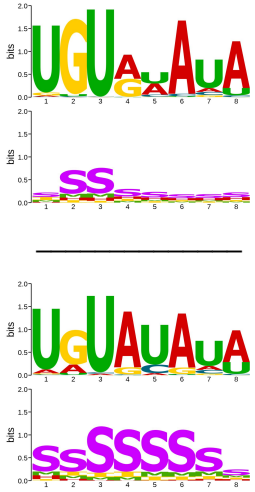


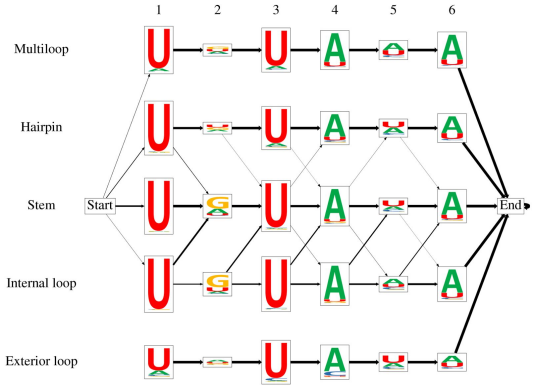
Figure S2. The heatmap shown here depicts a hierarchical clustering of both kernels (rows) and input sequences (columns) of a standardized matrix M where each cell M(i,j) represents the maximum activation value of kernel i for input sequence j. The resulting clustering is indicative of both class enrichment (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 co-occurring motifs.

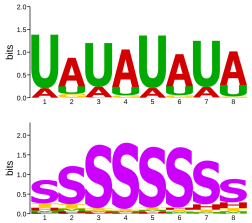
Protein	Literature	pysster (top 2)	ssHMM (from [2])	GraphProt	Performance

PUM2

[1]





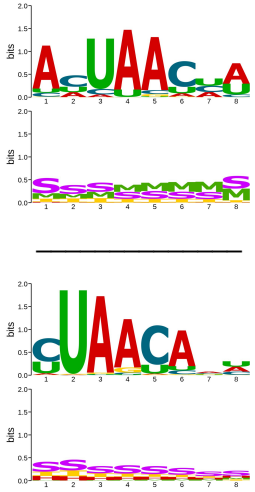


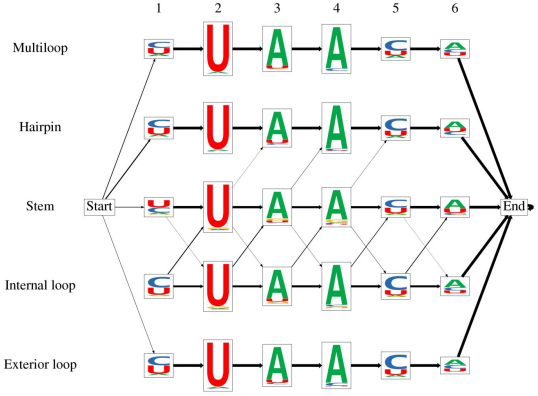
	ROC-AUC	PR-AUC	Runtime (seconds)
pysster (CPU)	0.948	0.929	89
	0.949	0.932	105
	0.945	0.925	76
pysster (GPU)	0.948	0.930	53
	0.944	0.944	49
	0.945	0.924	52
GraphProt	0.853	0.853	2503

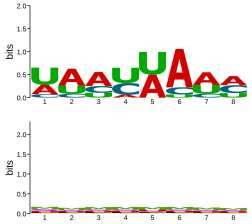
(7776 training + 2592 test sequences)

QKI

[1]





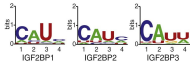


	ROC-AUC	PR-AUC	Runtime (seconds)
pysster (CPU)	0.963	0.949	101
	0.965	0.953	81
	0.965	0.955	84
pysster (GPU)	0.966	0.959	55
	0.966	0.955	44
	0.962	0.943	43
GraphProt	0.888	0.891	2014

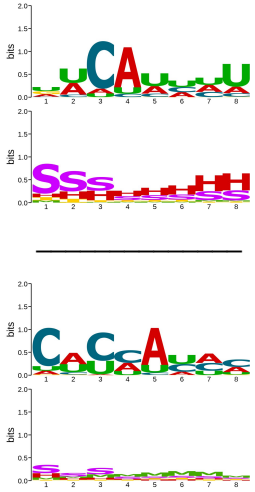
(6524 training + 2176 test sequences)

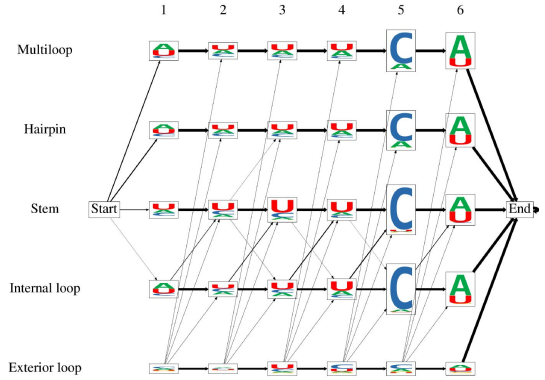
Protein	Literature	pysster (top 2)	ssHMM (from [2])	GraphProt	Performance

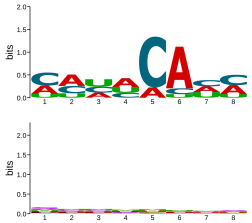
IGF2BP 1/2/3



[1]





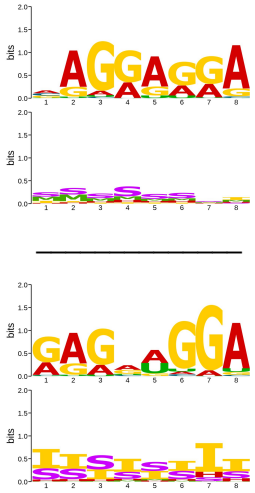


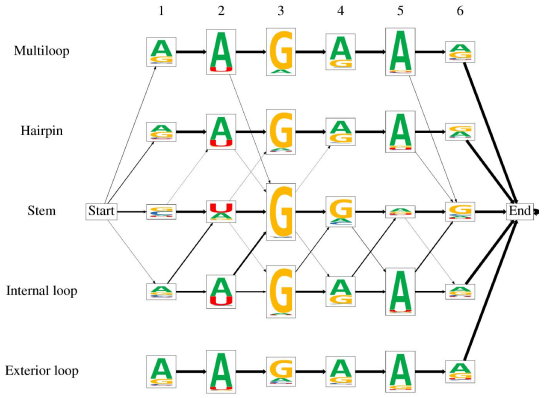
	ROC-AUC	PR-AUC	Runtime (seconds)
pysster (CPU)	0.837	0.844	85
	0.835	0.840	68
	0.837	0.843	81
pysster (GPU)	0.837	0.840	55
	0.837	0.843	51
	0.836	0.843	50
GraphProt	0.701	0.663	2400

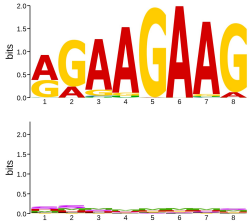
(7694 training + 2566 test sequences)

SFRS1/2

[3]

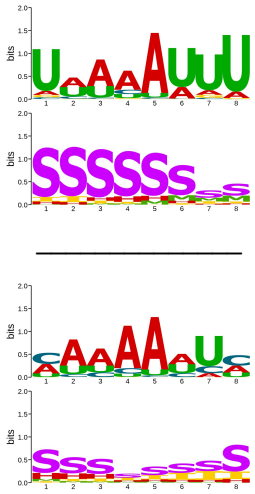
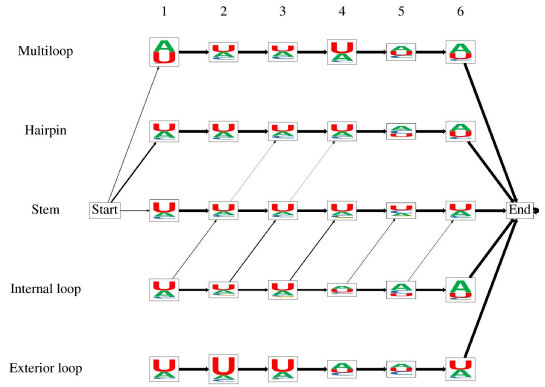
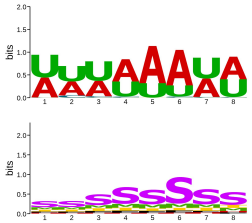
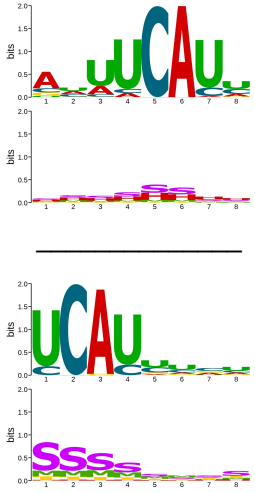
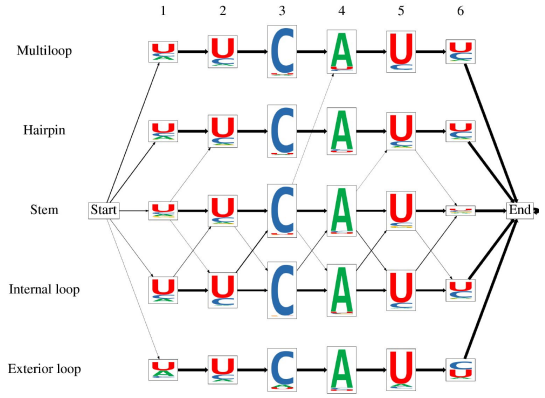
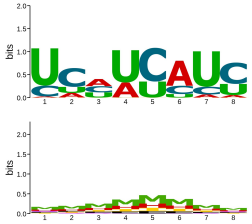






	ROC-AUC	PR-AUC	Runtime (seconds)
pysster (CPU)	0.934	0.943	475
	0.934	0.943	426
	0.933	0.943	389
pysster (GPU)	0.933	0.943	208
	0.934	0.943	218
	0.933	0.943	239
GraphProt	0.870	0.852	10730

(32604 training + 10868 test sequences)

Protein	Literature	pysster (top 2)	ssHMM (from [2])	GraphProt	Performance
TAF2N	AU-rich stem-loops [4]				
NOVA	UCAUY [5]				

	ROC-AUC	PR-AUC	Runtime (seconds)
pysster (CPU)	0.905	0.893	142
	0.910	0.902	144
	0.903	0.893	131
pysster (GPU)	0.908	0.902	73
	0.906	0.896	73
	0.909	0.900	82
GraphProt	0.700	0.658	3960

(12976 training + 4326 test sequences)

	ROC-AUC	PR-AUC	Runtime (seconds)
pysster (CPU)	0.952	0.944	100
	0.953	0.947	101
	0.953	0.946	96
pysster (GPU)	0.954	0.949	56
	0.955	0.949	58
	0.956	0.948	53
GraphProt	0.881	0.886	2531

(8348 training + 2784 test sequences)

Table S1. Performance comparison of pysster and GraphProt (v1.1.7) on CLIP-seq data for proteins with well-known RNA binding site motifs. For these binary classification tasks positive sets contained bindings sites from the protein of interest and negative sets contained randomly selected binding sites from 24 other proteins. All used sequences are of length 200 and centered at a binding site. Data are based on the ssHMM paper and the measurement source code and full output is available at https://github.com/budach/pysster/tree/master/tutorials/rbp. Aside from

setting the motif length for GraphProt and the kernel length for pysster to 8 both tools were run with default parameters. Runtime includes RNA secondary structure prediction, model training and prediction of held-out test sequences and was measured on a machine with two E5-2697Av4 CPUs (32 physical cores) and an Nvidia Titan X GPU. Due to the non-deterministic nature of neural networks pysster was run three times to show that the predictive performance is stable. In addition, pysster performance is also shown for the CPU-only version. By default, pysster produces 30 motifs and the two with the highest importance score showing enrichment in the positive set are featured in the table. For ssHMM only the learned motifs are shown, as it is an unsupervised model only trained on the positive set.

References

[1] Hafner, M., Landthaler, M., Burger, L., Khorshid, M., Hausser, J., Berninger, P., Rothballer, A., Ascano, M., Jungkamp, A.-C., Munschauer, M., *et al.* (2010). Transcriptome-wide identification of rna-binding protein and microrna target sites by par-clip. *Cell*, **141**(1), 129–141.

[2] Heller, D., Krestel, R., Ohler, U., Vingron, M., and Marsico, A. (2017). sshmm: extracting intuitive sequence-structure motifs from high-throughput rna-binding protein data. *Nucleic acids research*, **45**(19), 11004–11018.

[3] Sanford, J. R., Wang, X., Mort, M., VanDuyn, N., Cooper, D. N., Mooney, S. D., Edenberg, H. J., and Liu, Y. (2009). Splicing factor Sfrs1 recognizes a functionally diverse landscape of rna transcripts. *Genome research*, **19**(3), 381–394.

[4] Hoell, J. I., Larsson, E., Runge, S., Nusbaum, J. D., Duggimpudi, S., Farazi, T. A., Hafner, M., Borkhardt, A., Sander, C., and Tuschl, T. (2011). Rna targets of wild-type and mutant fet family proteins. *Nature Structural and Molecular Biology*, **18**(12), 1428.

[5] Jensen, K. B., Dredge, B. K., Stefani, G., Zhong, R., Buckanovich, R. J., Okano, H. J., Yang, Y. Y., and Darnell, R. B. (2000). Nova-1 regulates neuron-specific alternative splicing and is essential for neuronal viability. *Neuron*, **25**(2), 359–371.