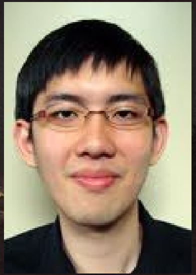
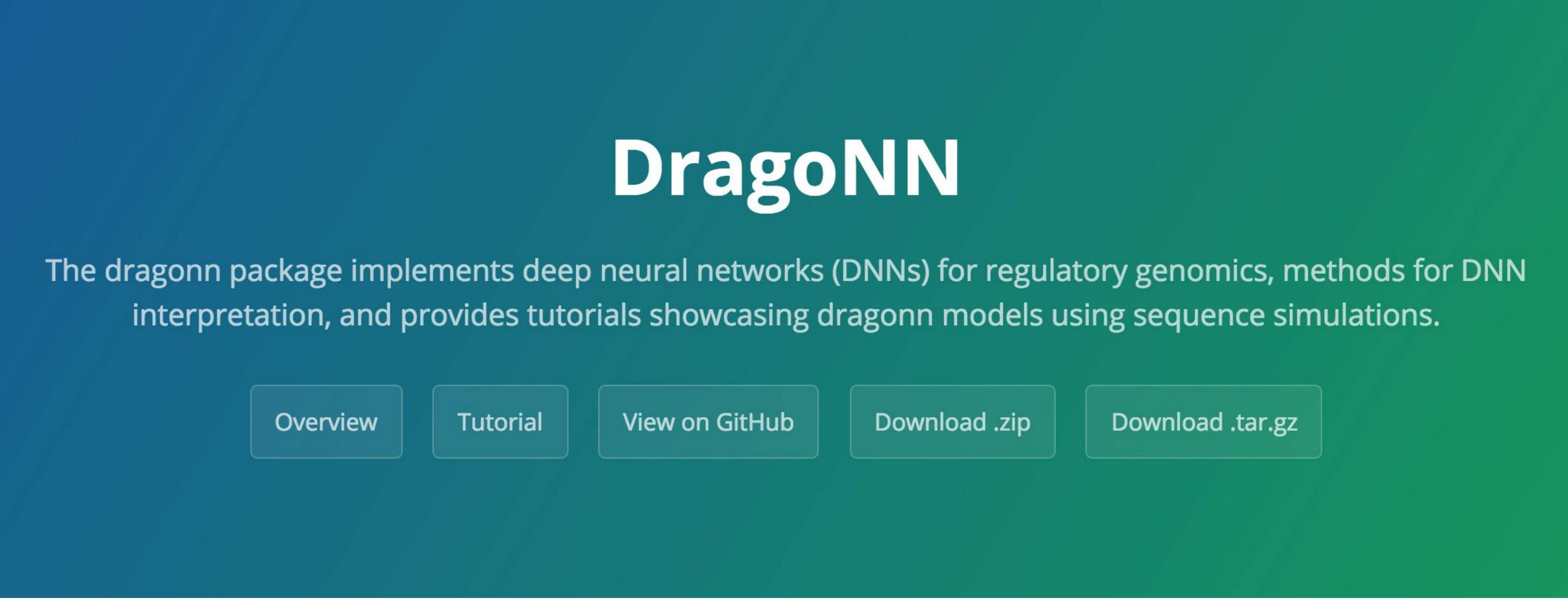
How to train your DragoNN

(Deep RegulAtory GenOmic Neural Network)



A workshop on Deep Learning for Regulatory Genomics

June 9th 2016
ENCODE Users Meeting
Stanford University



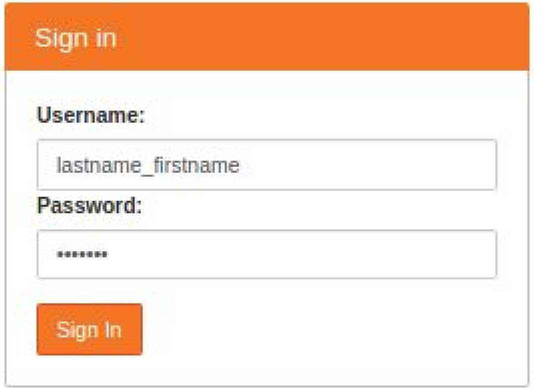
DragoNN

The dragonn package implements deep neural networks (DNNs) for regulatory genomics, methods for DNN interpretation, and provides tutorials showcasing dragonn models using sequence simulations.

For code, tutorial, and upcoming workshops: http://kundajelab.github.io/dragonn/

Primer with guidelines and in-vivo models coming soon!

Before we begin ..


Sign in

Username:

lastname_firstname

Password:

•••••••

Sign In

Logging in and starting the tutorial

1. Point your browser to

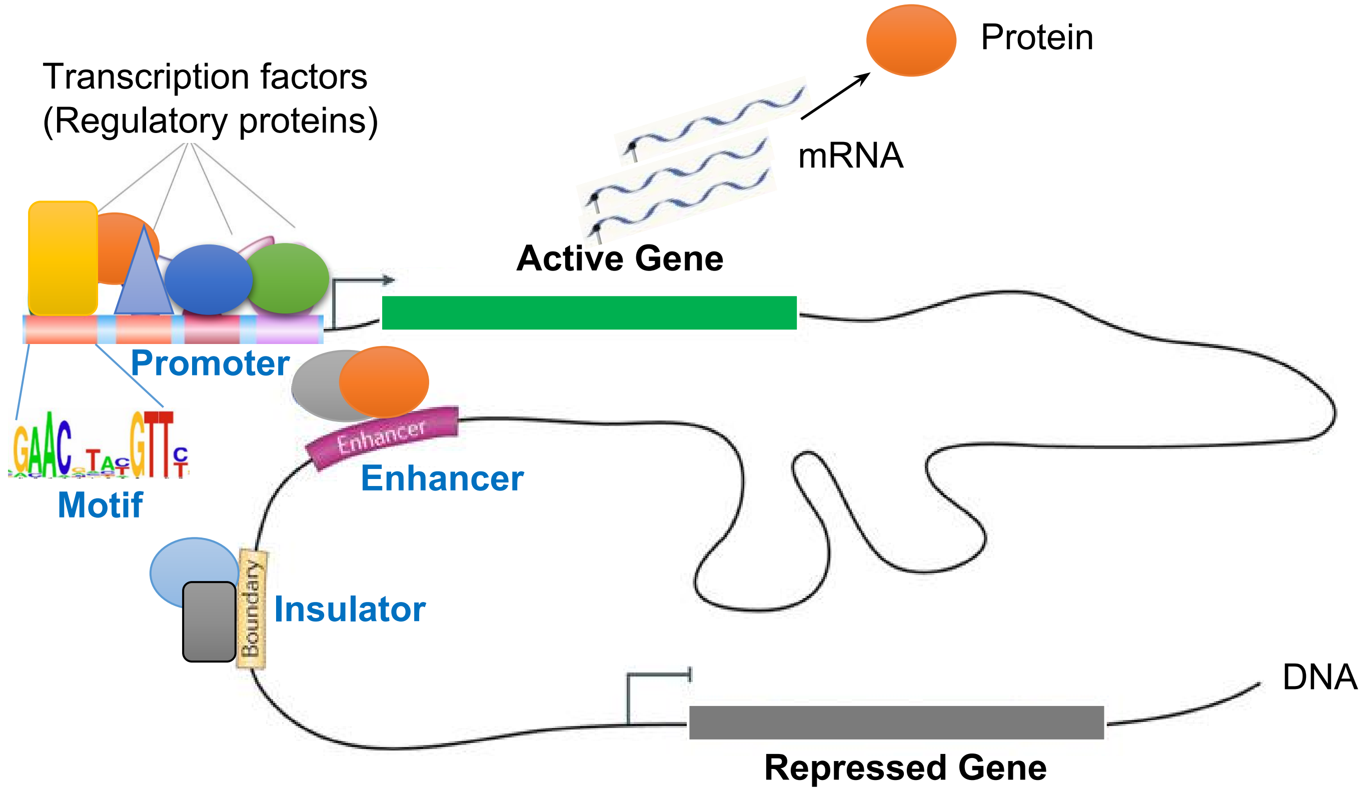
http://mitra.stanford.edu/dragonn.html

This should bring up a login page to the dragonn client:

2. Your username is of the format **lastname_firstname** based on the information you used to register for this tutorial.

The password is **dragonn**

Gene Regulation



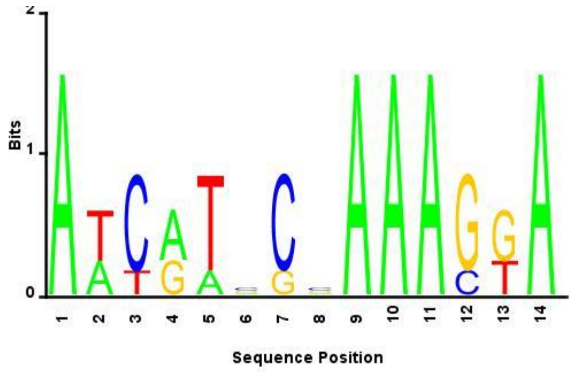
Sequence motifs

ATTATAGCAAACTA
AACATGCCAAAGTA
ATCATCCAAAAGGA
ATCGTCCGAAAGGA
AACGAGCGAAAGGA

Set of aligned sequences

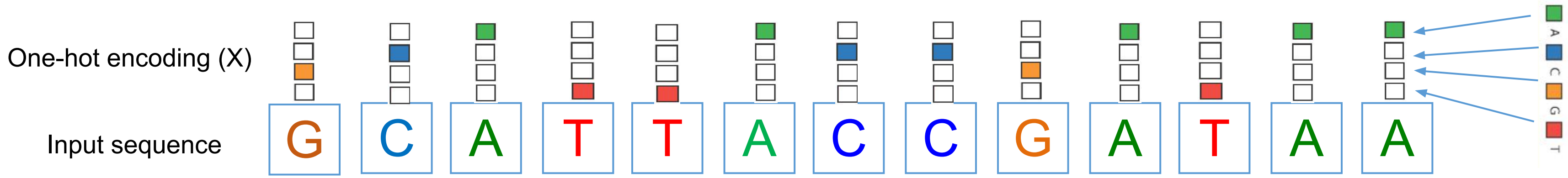
A	1	0.4	0	0.6	0.2	0.2	0	0.2	1	1	1	0	0	1
C	0	0	0.8	0	0	0.4	0.8	0.4	0	0	0	0.2	0	0
G	0	0	0	0.4	0	0.4	0.2	0.4	0	0	0	0.8	0.6	0
T	0	0.6	0.2	0	0.8	0	0	0	0	0	0	0	0.4	0

Position-specific scoring matrix
$p_i(x_i = a_i)$



PSSM logo

For a subsequence $S = a_1, a_2, \dots, a_k$ where $a_i \in \{A, C, G, T\}$ log-odds score$(S) = \sum_{i=1\dots k} \log_2 \left(\frac{p_i(x_i = a_i)}{p_{bg}(x_i = a_i)} \right)$ $\geq$ threshold $\Longrightarrow$ true hit

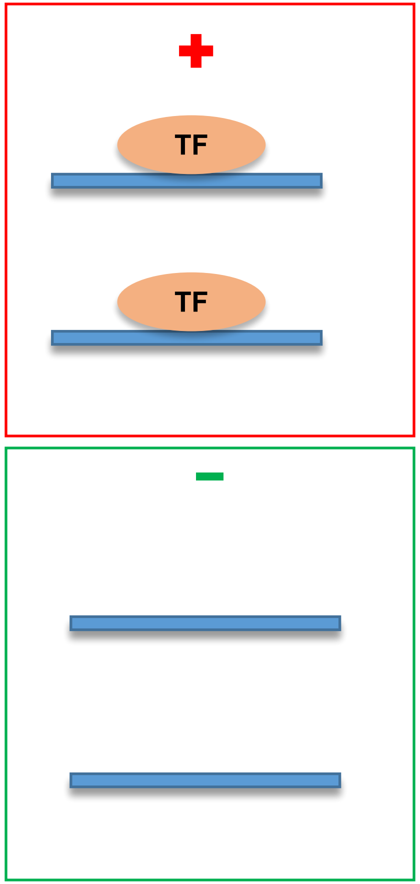


Learning regulatory sequence patterns

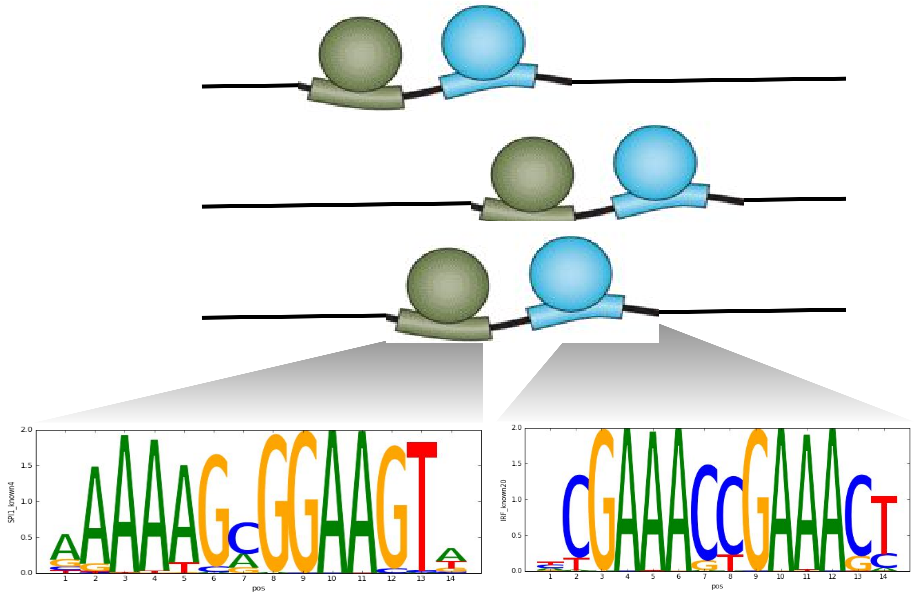
- Positive class of genomic sequences bound a TF of interest

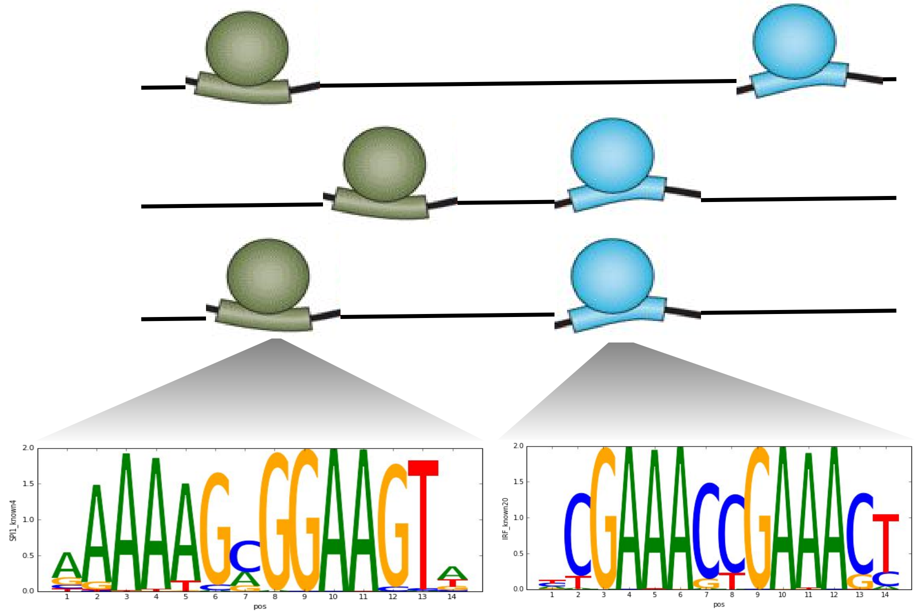
Can we learn patterns (motifs) in the DNA sequence that distinguish these 2 classes of genomic sequences?

- Negative class of genomic sequences not bound by TF of interest



Simulation

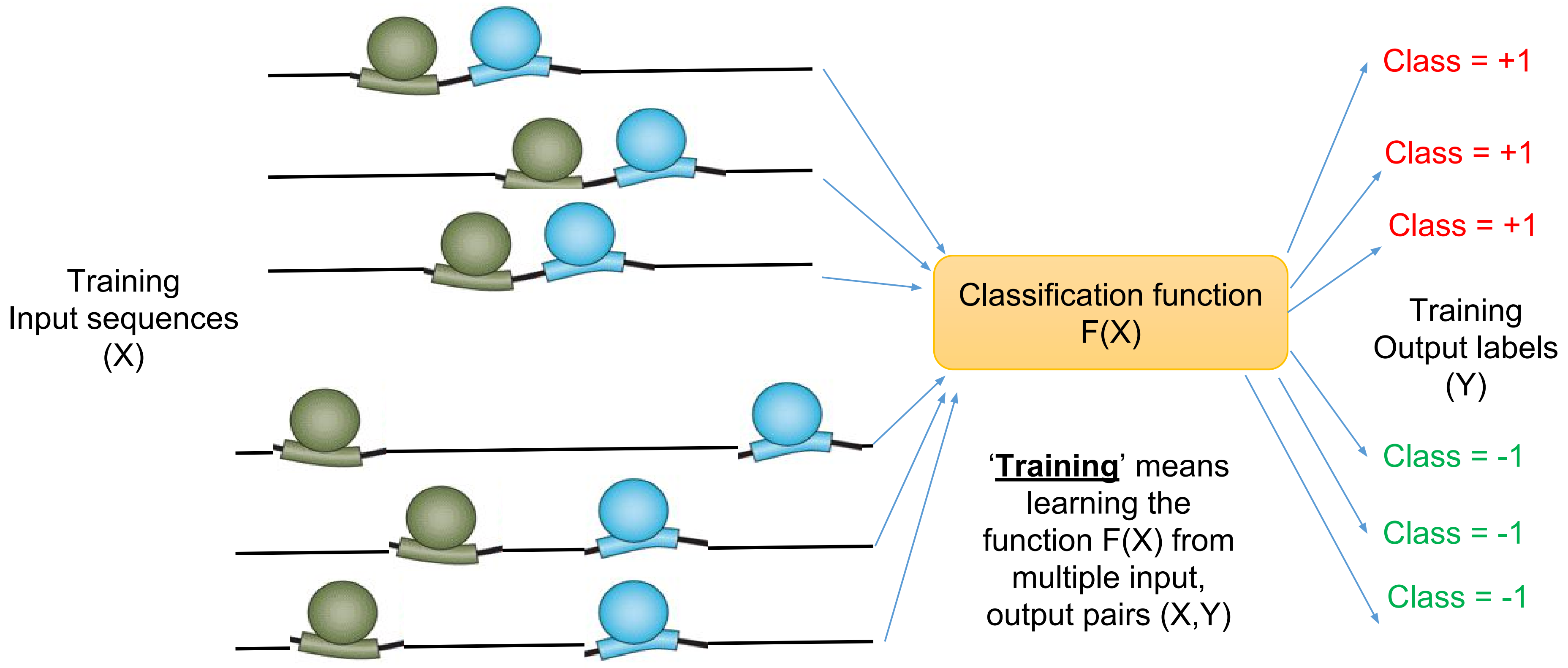




Positive class of genomic sequences containing two motifs with relatively **fixed** spacing

Negative class of genomic sequences containing two motifs with **random** spacing

Supervised machine learning



‘**<u>Training</u>**’ means learning the function F(X) from multiple input, output pairs (X,Y)

Supervised machine learning

Test
Input sequences
(X)

Classification function
F(X)

Is its class Y = + 1 or – 1?
Probability (Y = +1 given X)

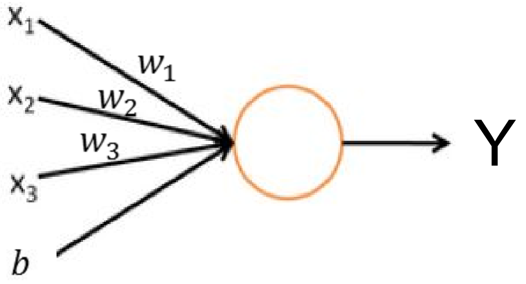
‘**<u>Testing</u>**’ means predicting the output labels (class) Y for a new sequence X not used in training

A simple classifier (An artificial neuron)

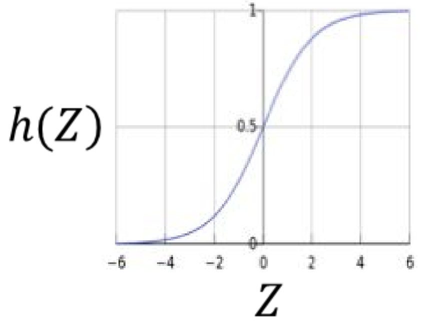
$$Y = F(x_1, x_2, x_3)$$

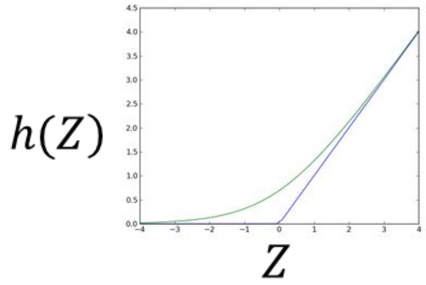
$$Z = w_1.x_1 + w_2.x_2 + w_3.x_3 + b$$

$$Y = h(Z)$$



Logistic / Sigmoid

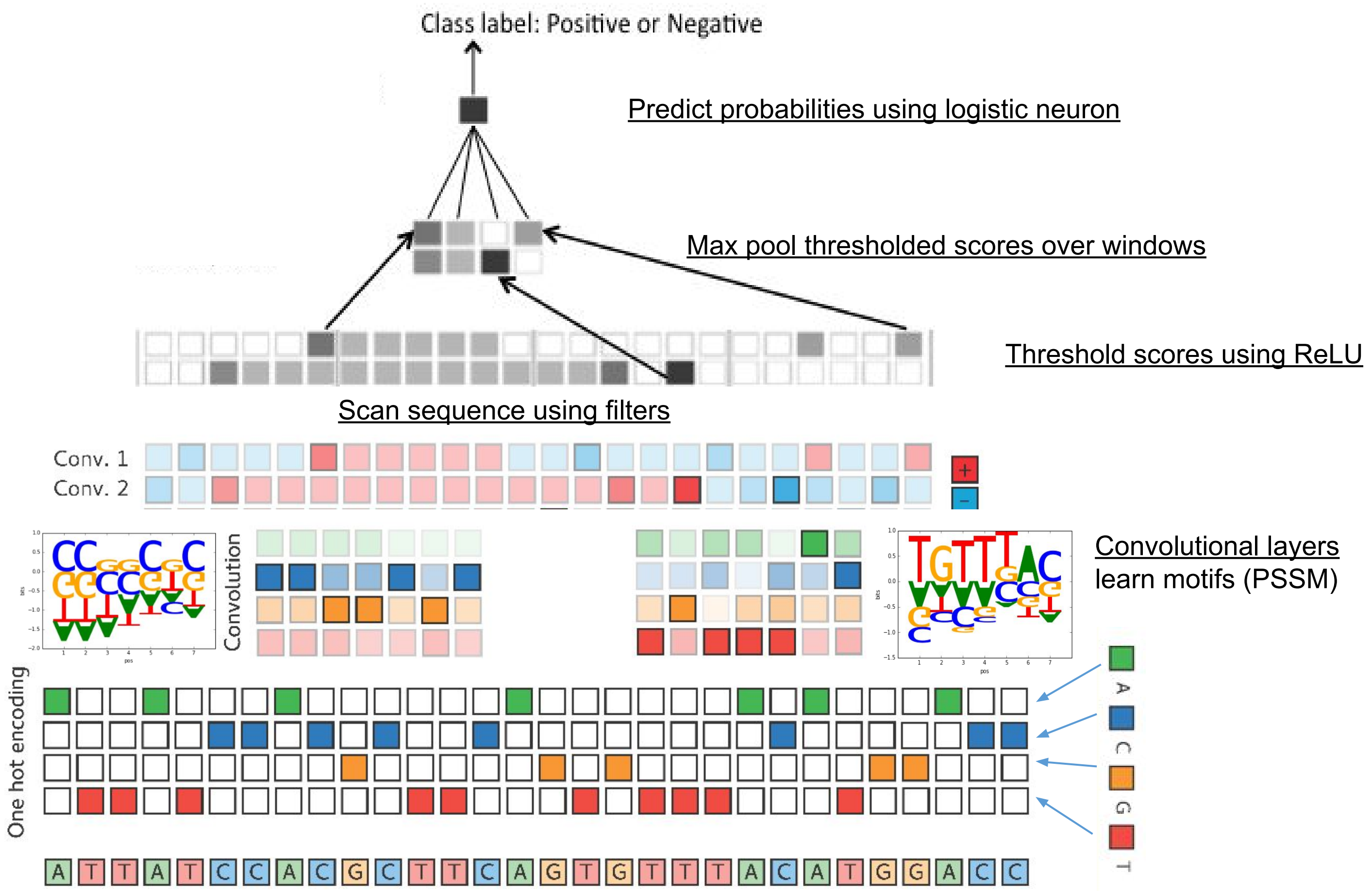




ReLu (Rectified Linear Unit)

w, *b* are the parameters of this neuron Training means learning the optimal w's and b

Biological motivation of DCNN


Class label: Positive or Negative
Predict probabilities using logistic neuron
Max pool thresholded scores over windows
Threshold scores using ReLU
Scan sequence using filters
Conv. 1
Conv. 2
+
−
Convolution
Convolutional layers learn motifs (PSSM)
One hot encoding
A
C
G
T
A T T A T C C A C G C T T C A G T G T T T A C A T G G A C C

Deep convolutional neural network

Logistic output

Typically followed by one or more fully connected layers

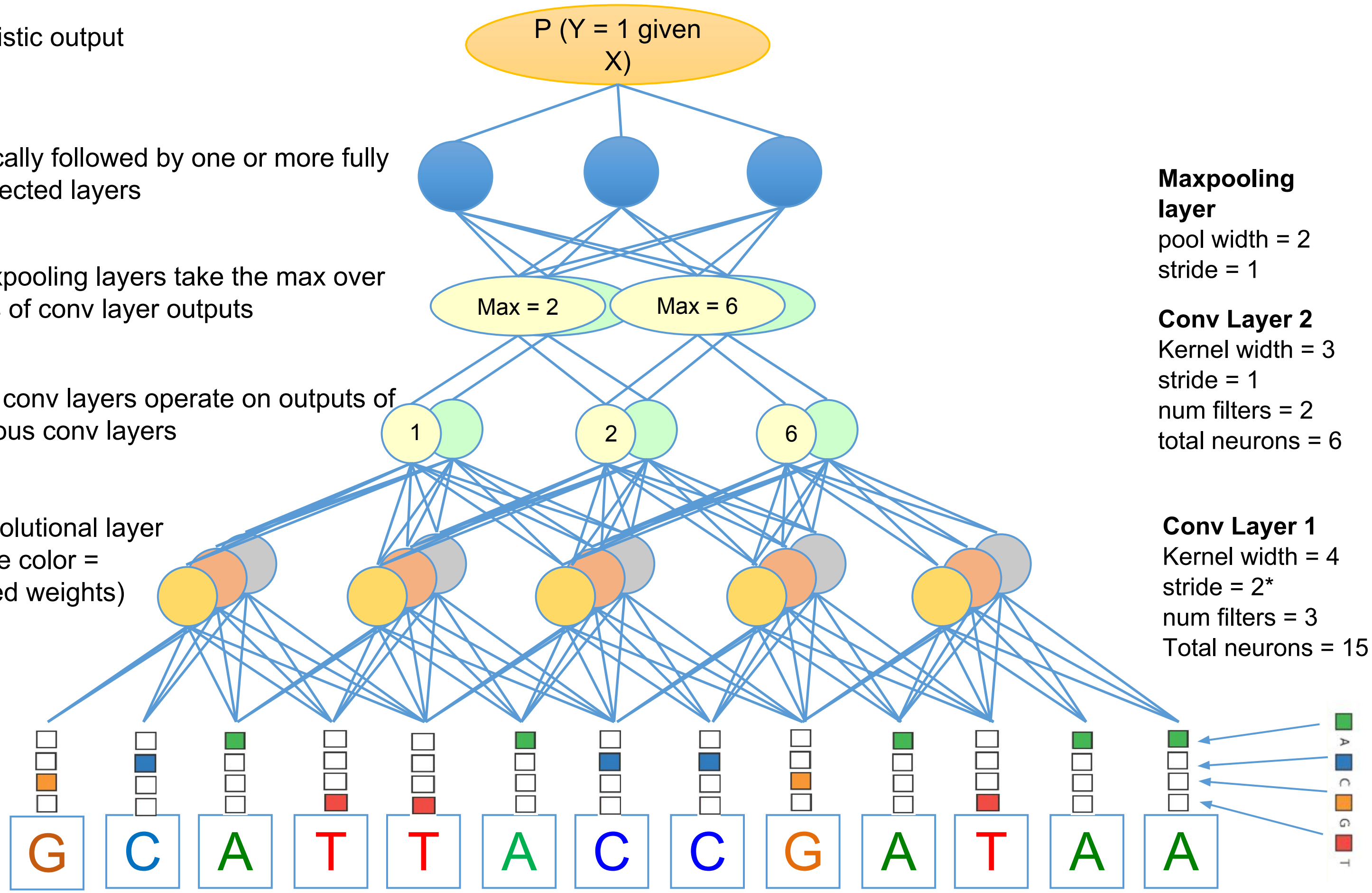
Maxpooling layers take the max over sets of conv layer outputs

Later conv layers operate on outputs of previous conv layers

Convolutional layer (same color = shared weights)

One-hot encoding (X)

Input sequence



Maxpooling layer
pool width = 2
stride = 1

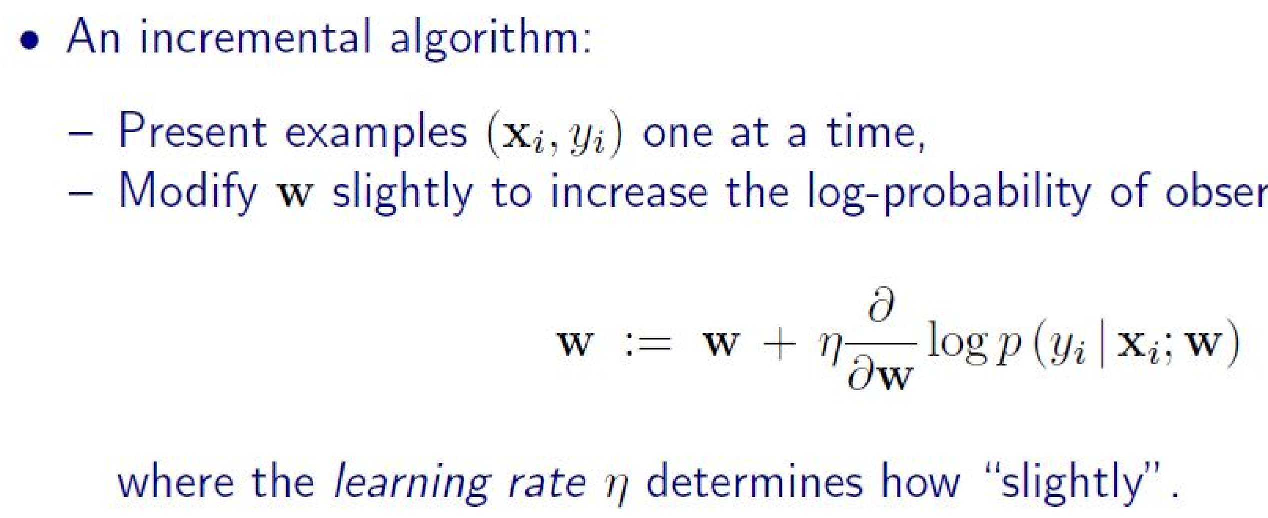
Conv Layer 2
Kernel width = 3
stride = 1
num filters = 2
total neurons = 6

Conv Layer 1
Kernel width = 4
stride = 2*
num filters = 3
Total neurons = 15

Training a neural network

Learning weights via optimization algorithm called **stochastic gradient descent**

Optimization Objective: Minimize error (**loss**) on the training datasets
i.e. difference between true Y and predicted output F(X)



- An incremental algorithm:
  - Present examples $(\mathbf{x}_i, y_i)$ one at a time,
  - Modify $\mathbf{w}$ slightly to increase the log-probability of observed y_i:

$$\mathbf{w} \;:=\; \mathbf{w} + \eta \frac{\partial}{\partial \mathbf{w}} \log p\,(y_i \,|\, \mathbf{x}_i; \mathbf{w})$$

where the *learning rate* η determines how "slightly".

Measures of performance

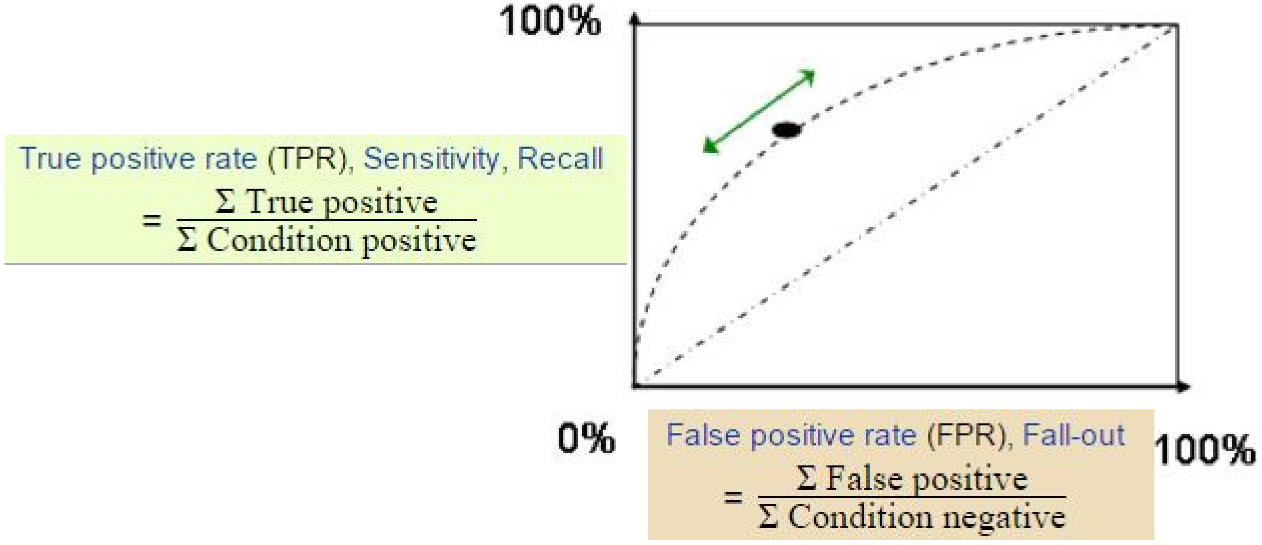
TP, FP, FN, TN are absolute counts of true positives, false positives, false negatives and true negatives

- N - sample size
- $N^+ = FN + TP$ number of positive examples
- $N^- = FP + TN$ number of negative examples
- $O^+ = TP + FP$ number of positive predictions
- $O^- = FN + TN$ number of negative predictions

outputs\ labeling	$y = +1$	$y = -1$	Σ
$f(x) = +1$	TP	FP	O^+
$f(x) = -1$	FN	TN	O^-
Σ	N^+	N^-	N

Measures of performance

Area under the Receiver operating curve (auROC)
Compares sensitivity (recall) to false positive rate (1-specificity) at various thresholds
auROC = 1 (Perfect classifier)
auROC = 0.5 (Random classifier)

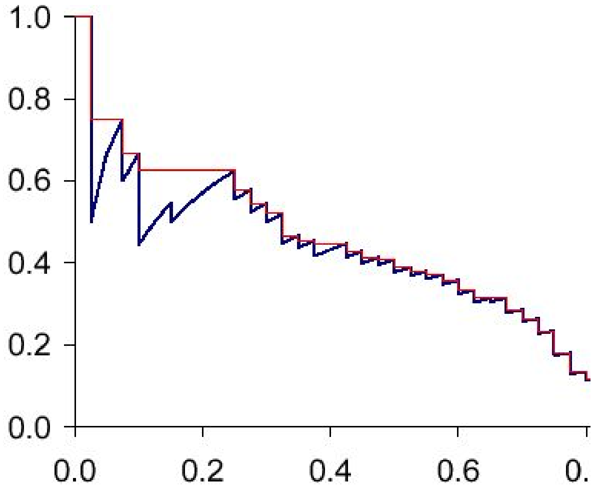


Measures of performance

Area under Precision-Recall Curve (auPRC)
Compares precision (1- false discovery rate) to recall (sensitivity) at various thresholds
auPRC = 1 (Perfect classifier)

Positive predictive value (PPV), Precision

$$= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Test outcome positive}}$$



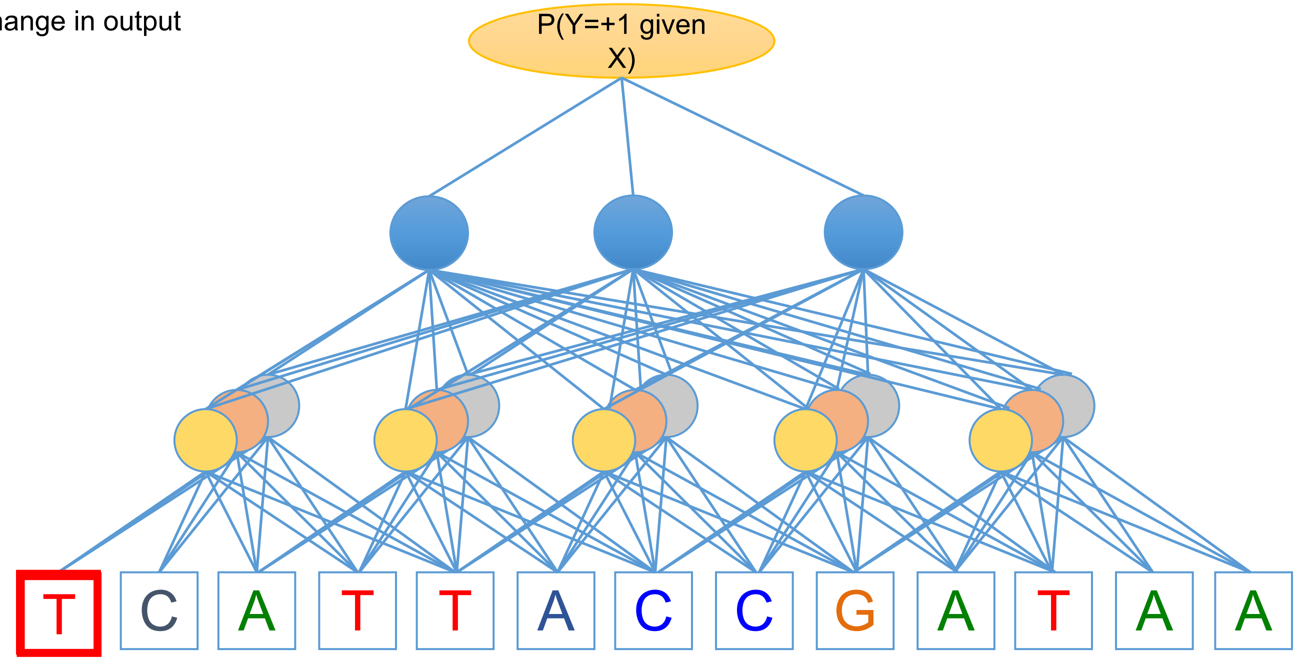
True positive rate (TPR), Sensitivity, Recall

$$= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$$

Interpretation: In-silico mutagenesis

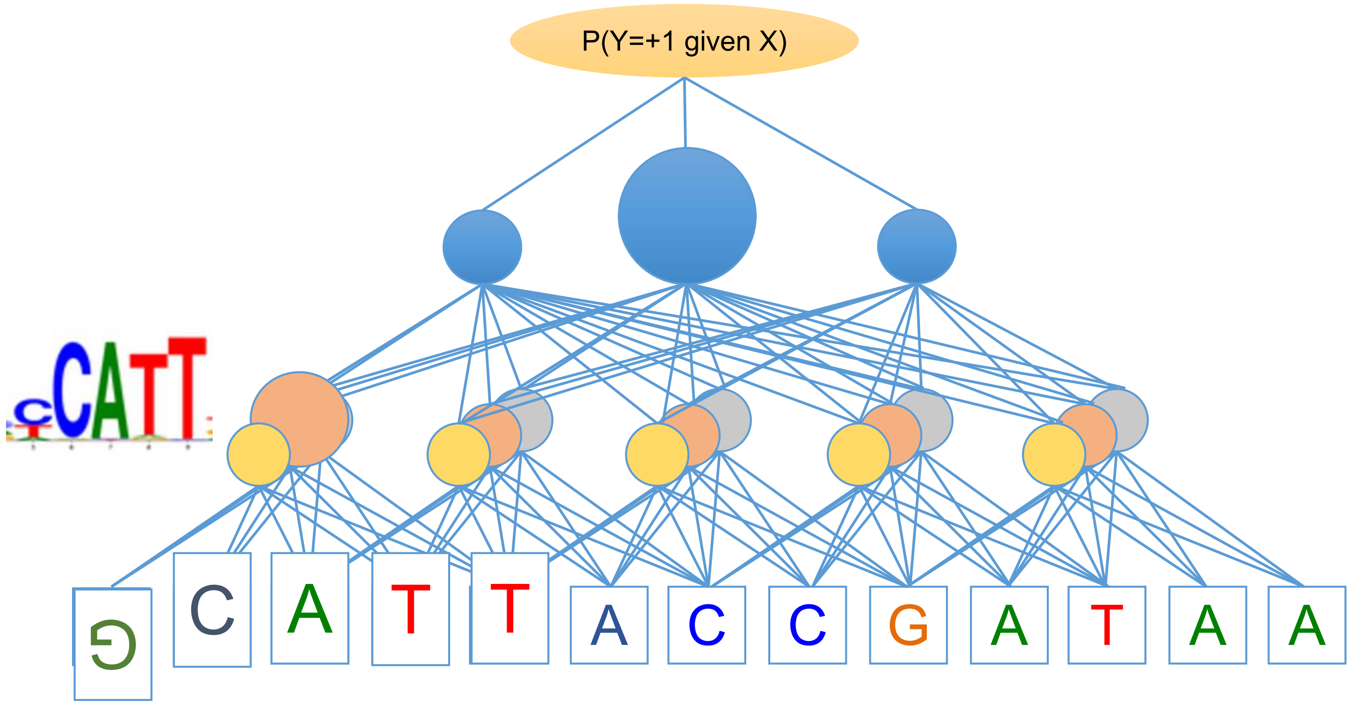
Output: Bound (+1) vs. not bound (0)

Assess change in output



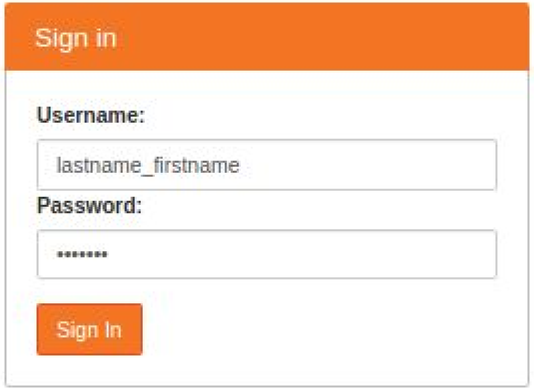
Input: One-hot encoded DNA sequence

Interpretation: DeepLIFT (Deep learning feature importance)



https://arxiv.org/abs/1605.01713

Starting the tutorial


Sign in

Username:

lastname_firstname

Password:

•••••••

Logging in and starting the tutorial

1. Point your browser to

http://mitra.stanford.edu/dragonn.html

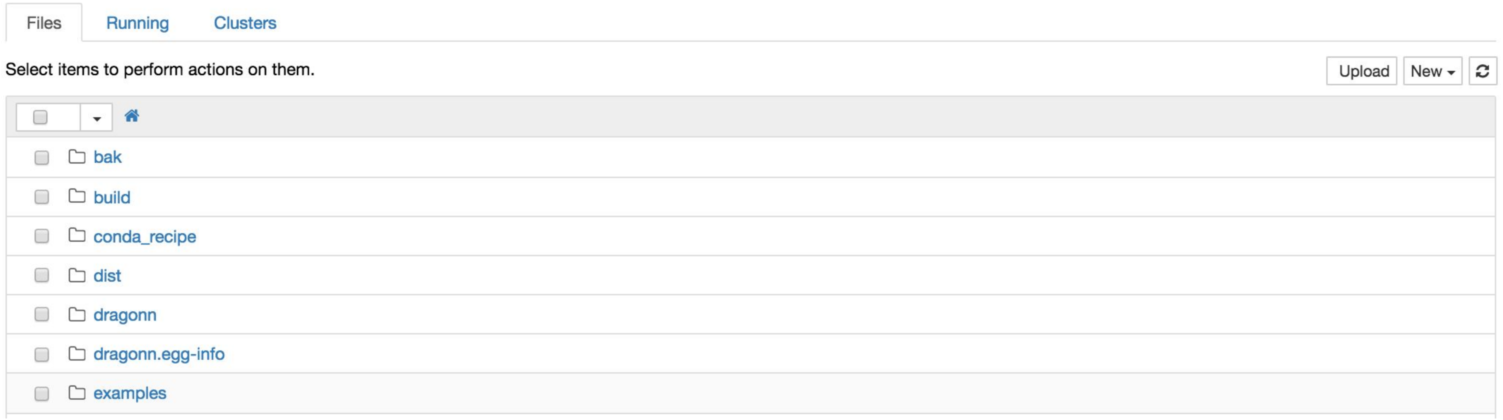
This should bring up a login page to the dragonn client:

2. Your username is of the format **lastname_firstname** based on the information you used to register for this tutorial.

The password is **dragonn**

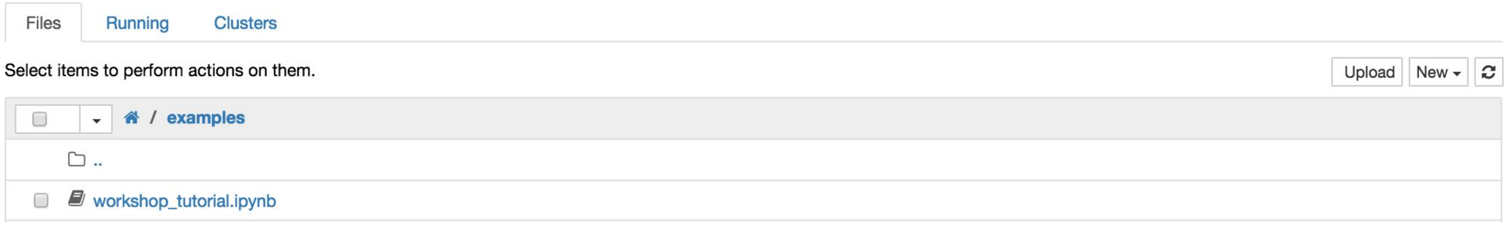
Starting the tutorial

3. Click on the **workshop_tutorial.ipynb** link inside the examples folder to open up the jupyter notebook for the tutorial.



Select items to perform actions on them.

bak
build
conda_recipe
dist
dragonn
dragonn.egg-info
examples



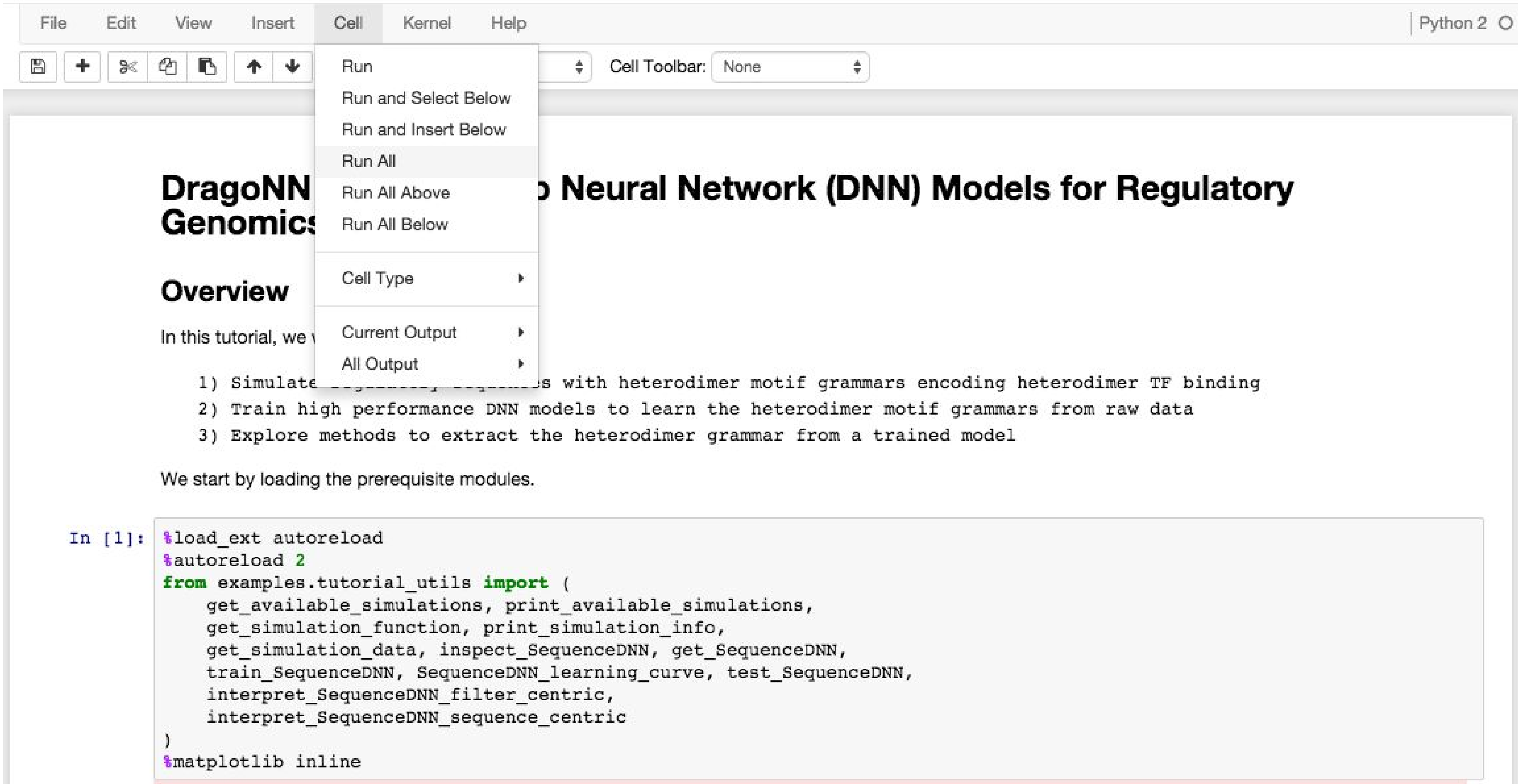
Select items to perform actions on them.

/ examples

..
workshop_tutorial.ipynb

Starting the tutorial

4. Click the "Run All" in the "Cell" dropdown menu



DragoNN [...] Neural Network (DNN) Models for Regulatory Genomics

Overview

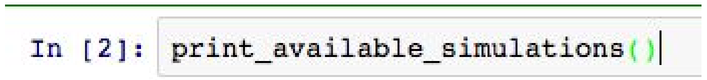
In this tutorial, we

```
1) Simulate                      s with heterodimer motif grammars encoding heterodimer TF binding
2) Train high performance DNN models to learn the heterodimer motif grammars from raw data
3) Explore methods to extract the heterodimer grammar from a trained model
```

We start by loading the prerequisite modules.

```
In [1]: %load_ext autoreload
        %autoreload 2
        from examples.tutorial_utils import (
            get_available_simulations, print_available_simulations,
            get_simulation_function, print_simulation_info,
            get_simulation_data, inspect_SequenceDNN, get_SequenceDNN,
            train_SequenceDNN, SequenceDNN_learning_curve, test_SequenceDNN,
            interpret_SequenceDNN_filter_centric,
            interpret_SequenceDNN_sequence_centric
        )
        %matplotlib inline
```

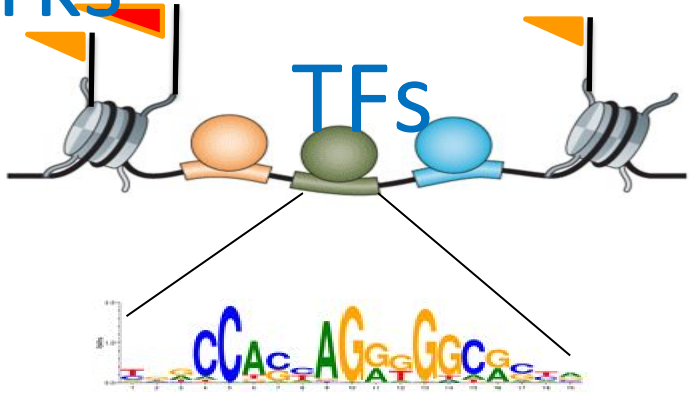
Sequence Simulations



```
In [2]: print_available_simulations()
```

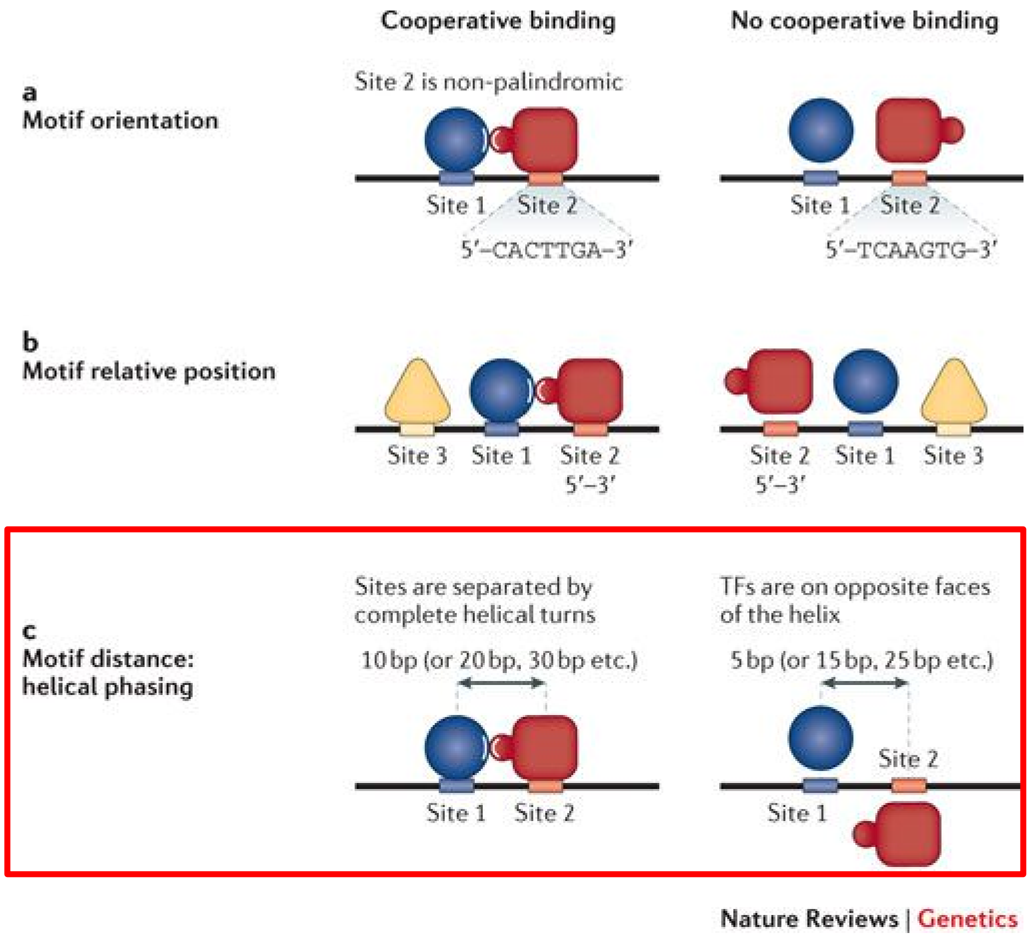
Simulation Name	**“Positive” class sequence**	**“Negative” class sequence**
simulate_single_motif_detection	Contains a single motif	Random sequence
simulate_motif_counting	Contains many instances of a motif	Contains few instances of a motif
simulate_motif_density_localization	Contains multiple instances of a motif in center	Contains multiple instances of a motif throughout
simulate_multi_motif_embedding	Contains multiple motifs, one instance of each	Random sequence
simulate_differential_accessibility	Contains a group of motifs	Contains a different group of motifs
simulate_heterodimer_grammar	Contains two motifs positioned closely	Contains two motifs positioned independently

Transcription factor (TF) binding in regulatory elements



Adapted from Shlyueva et al. (2014) Nature Reviews Genetics.

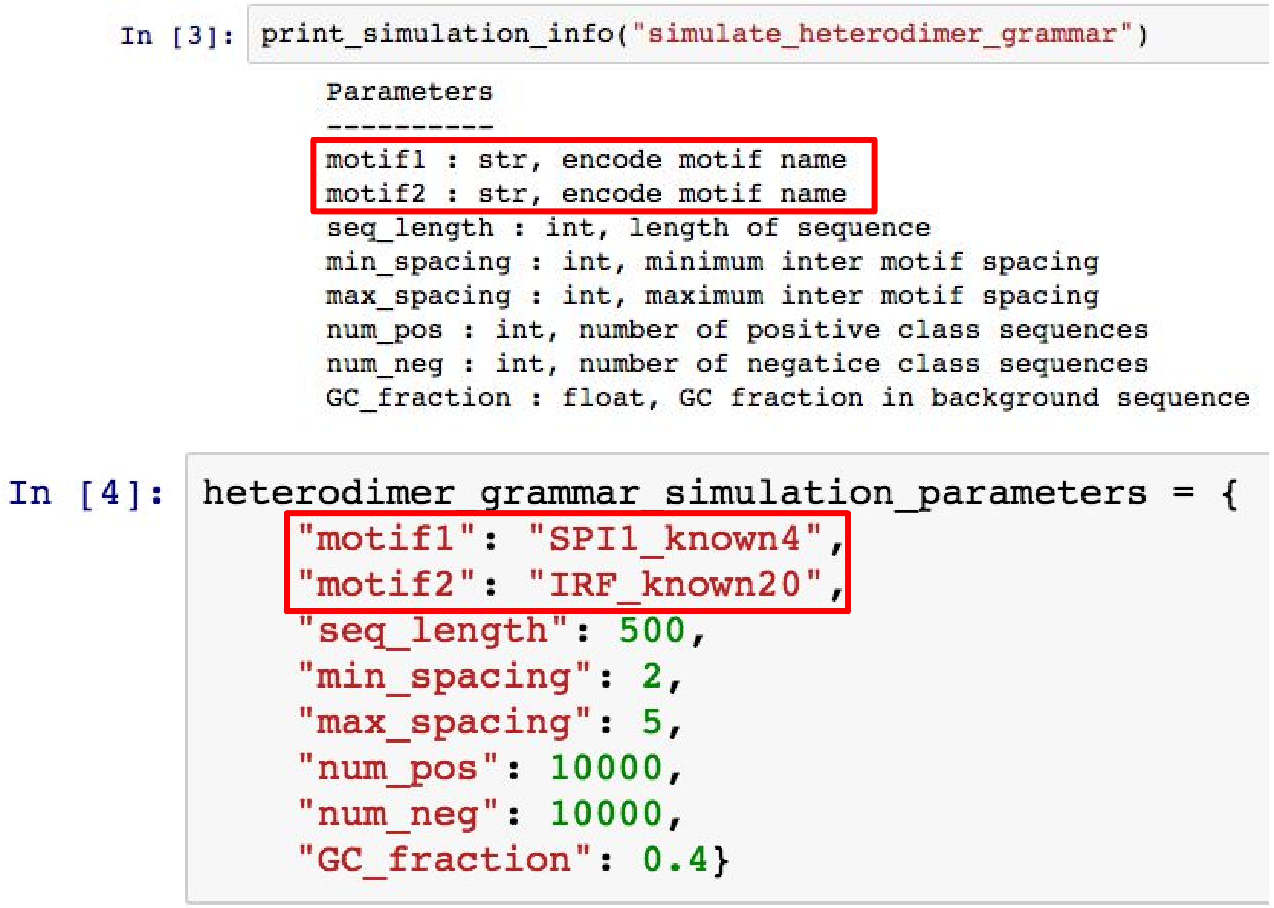
Cooperative vs non-cooperative binding



Cooperative binding
No cooperative binding
a
Motif orientation
Site 2 is non-palindromic
Site 1
Site 2
5′–CACTTGA–3′
Site 1
Site 2
5′–TCAAGTG–3′
b
Motif relative position
Site 3
Site 1
Site 2
5′–3′
Site 2
Site 1
Site 3
5′–3′
c
Motif distance: helical phasing
Sites are separated by complete helical turns
10 bp (or 20 bp, 30 bp etc.)
Site 1
Site 2
TFs are on opposite faces of the helix
5 bp (or 15 bp, 25 bp etc.)
Site 2
Site 1
Nature Reviews | Genetics

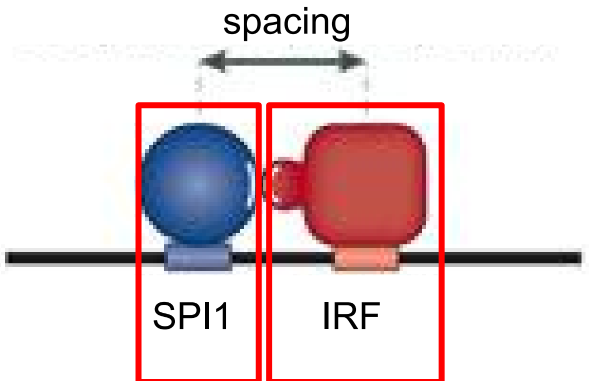
taken from François Spitz & Eileen E. M. Furlong. Nature Review Genetics 13, 613-626 (2012).

Defining Simulation Parameters

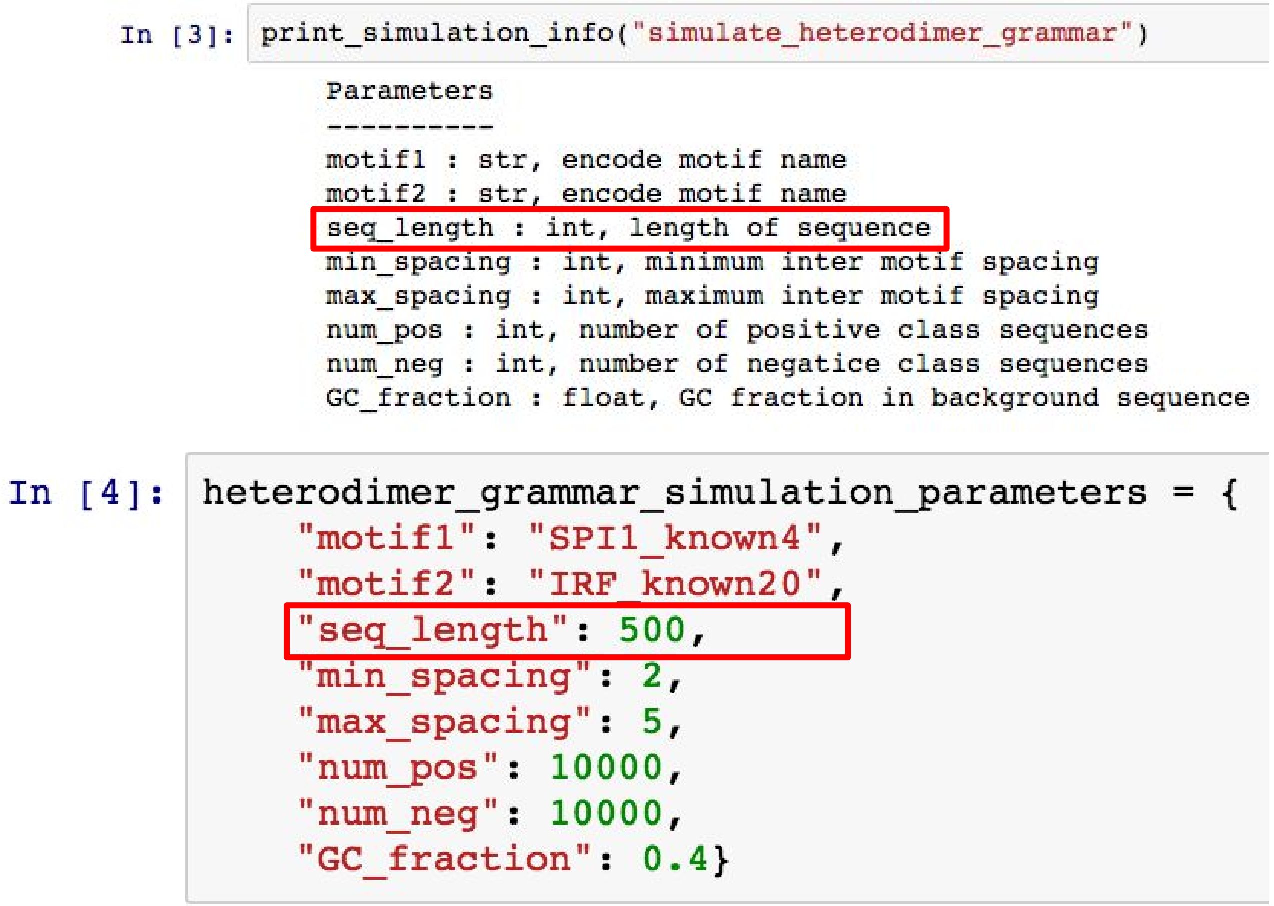


```
In [3]: print_simulation_info("simulate_heterodimer_grammar")

        Parameters
        ----------
        motif1 : str, encode motif name
        motif2 : str, encode motif name
        seq_length : int, length of sequence
        min_spacing : int, minimum inter motif spacing
        max_spacing : int, maximum inter motif spacing
        num_pos : int, number of positive class sequences
        num_neg : int, number of negatice class sequences
        GC_fraction : float, GC fraction in background sequence
```

```
In [4]: heterodimer_grammar_simulation_parameters = {
            "motif1": "SPI1_known4",
            "motif2": "IRF_known20",
            "seq_length": 500,
            "min_spacing": 2,
            "max_spacing": 5,
            "num_pos": 10000,
            "num_neg": 10000,
            "GC_fraction": 0.4}
```

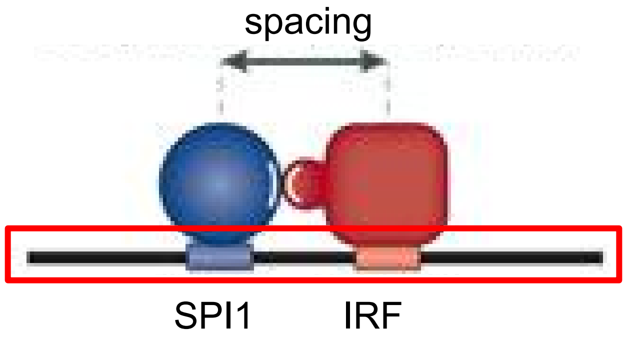


Defining Simulation Parameters

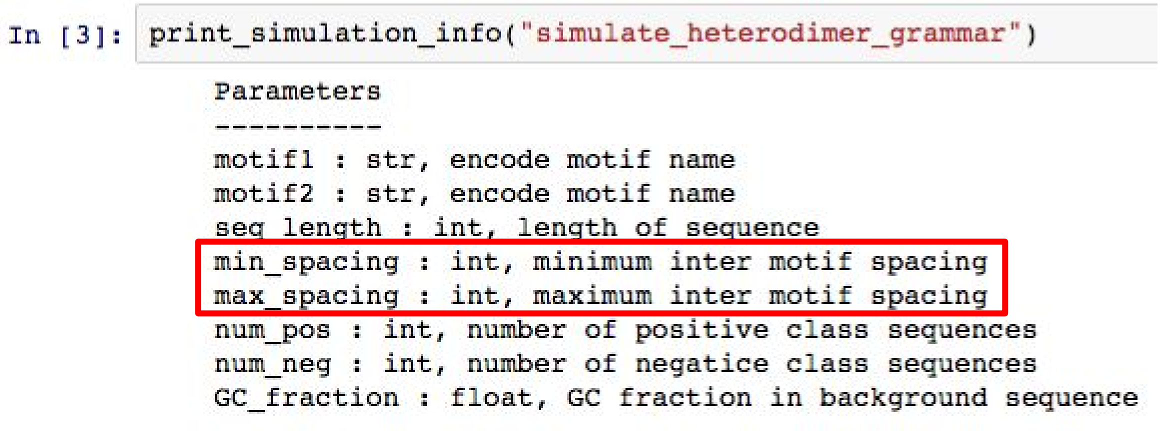


```
In [3]: print_simulation_info("simulate_heterodimer_grammar")

            Parameters
            ----------
            motif1 : str, encode motif name
            motif2 : str, encode motif name
            seq_length : int, length of sequence
            min_spacing : int, minimum inter motif spacing
            max_spacing : int, maximum inter motif spacing
            num_pos : int, number of positive class sequences
            num_neg : int, number of negatice class sequences
            GC_fraction : float, GC fraction in background sequence
```

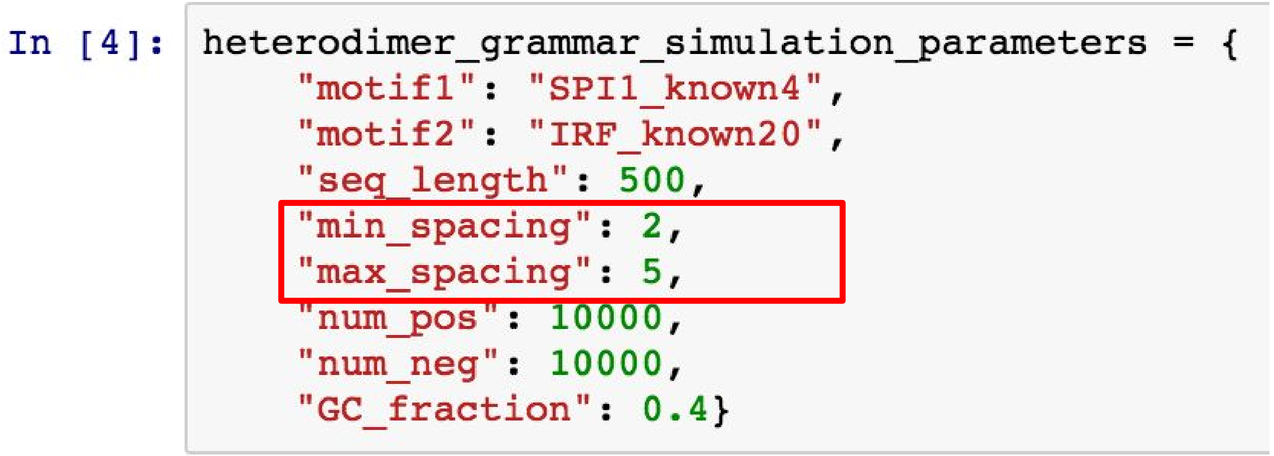
```
In [4]: heterodimer_grammar_simulation_parameters = {
            "motif1": "SPI1_known4",
            "motif2": "IRF_known20",
            "seq_length": 500,
            "min_spacing": 2,
            "max_spacing": 5,
            "num_pos": 10000,
            "num_neg": 10000,
            "GC_fraction": 0.4}
```



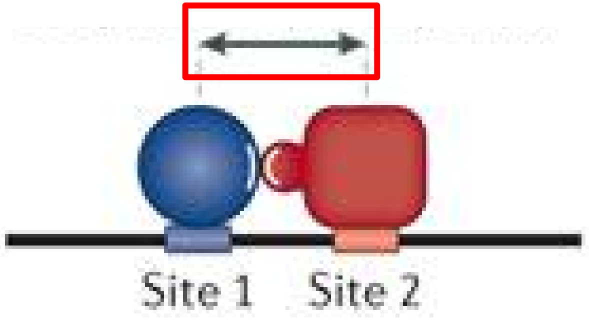
Defining Simulation Parameters



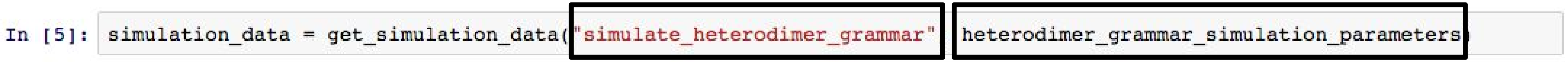
```
In [3]: print_simulation_info("simulate_heterodimer_grammar")

        Parameters
        ----------
        motif1 : str, encode motif name
        motif2 : str, encode motif name
        seq_length : int, length of sequence
        min_spacing : int, minimum inter motif spacing
        max_spacing : int, maximum inter motif spacing
        num_pos : int, number of positive class sequences
        num_neg : int, number of negatice class sequences
        GC_fraction : float, GC fraction in background sequence
```



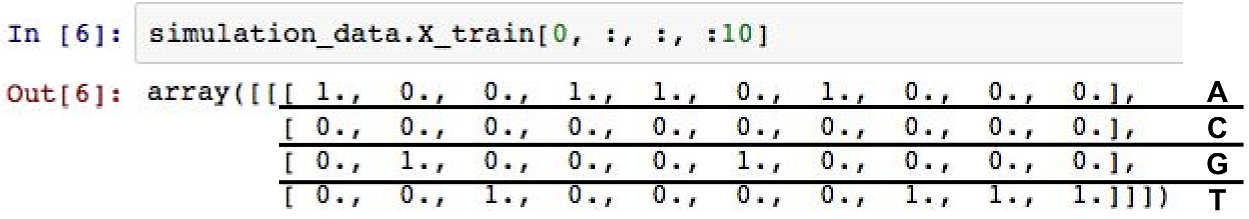
```
In [4]: heterodimer_grammar_simulation_parameters = {
            "motif1": "SPI1_known4",
            "motif2": "IRF_known20",
            "seq_length": 500,
            "min_spacing": 2,
            "max_spacing": 5,
            "num_pos": 10000,
            "num_neg": 10000,
            "GC_fraction": 0.4}
```



Getting Simulation Data



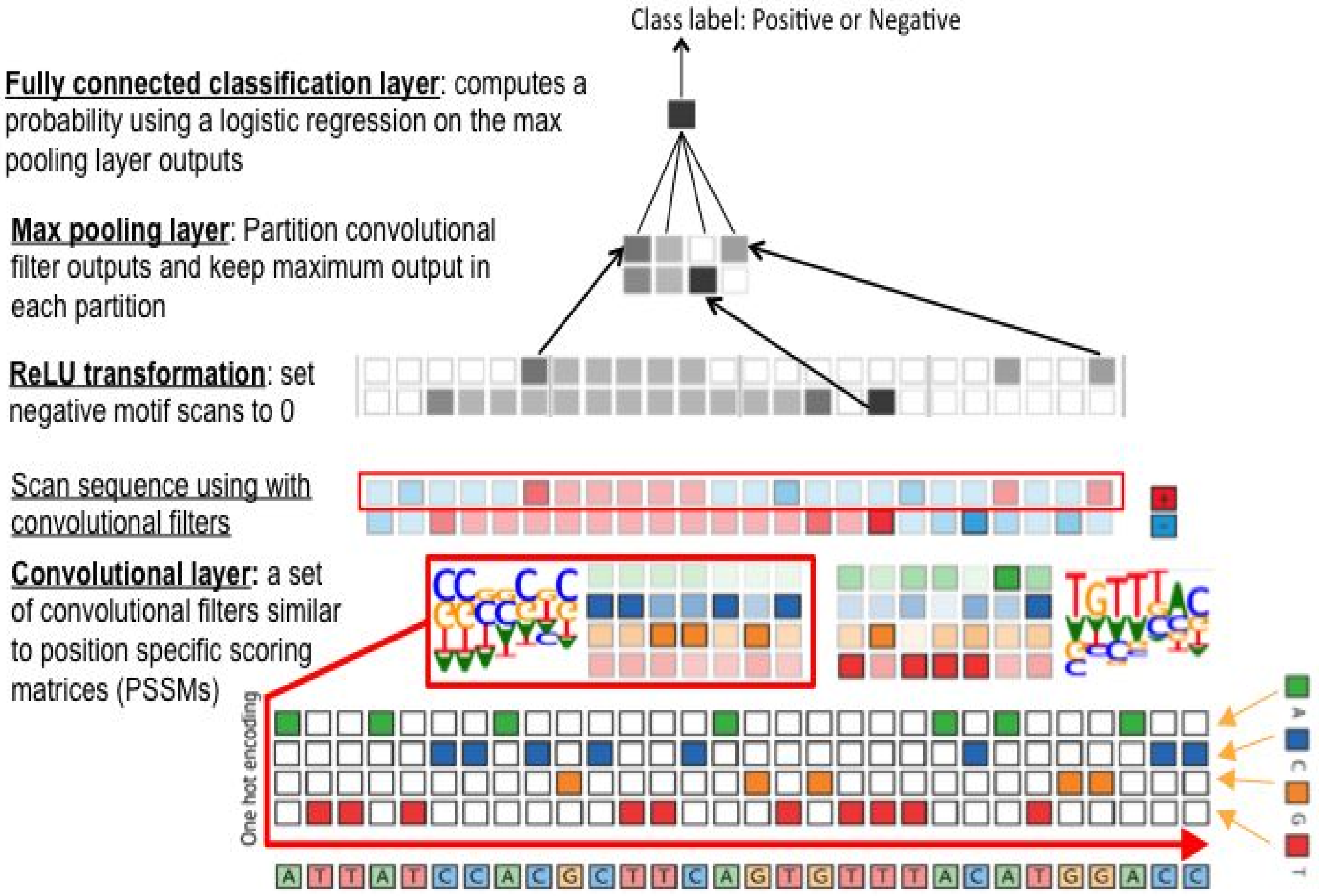
```
In [5]: simulation_data = get_simulation_data("simulate_heterodimer_grammar", heterodimer_grammar_simulation_parameters)
```



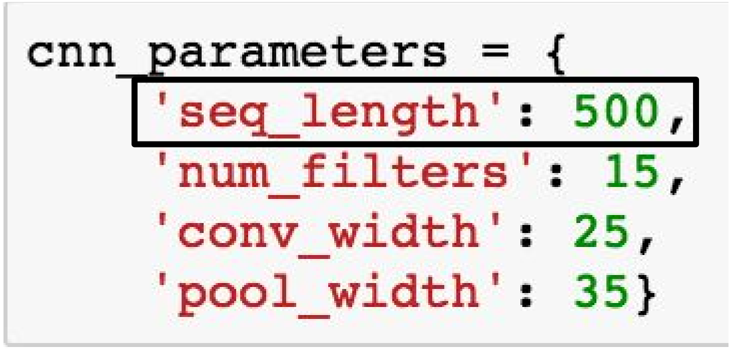
```
In [6]: simulation_data.X_train[0, :, :, :10]

Out[6]: array([[[ 1.,  0.,  0.,  1.,  1.,  0.,  1.,  0.,  0.,  0.],
          [ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
          [ 0.,  1.,  0.,  0.,  0.,  1.,  0.,  0.,  0.,  0.],
          [ 0.,  0.,  1.,  0.,  0.,  0.,  0.,  1.,  1.,  1.]]])
```

Underlying Sequence: “A G T A A G A T T T”

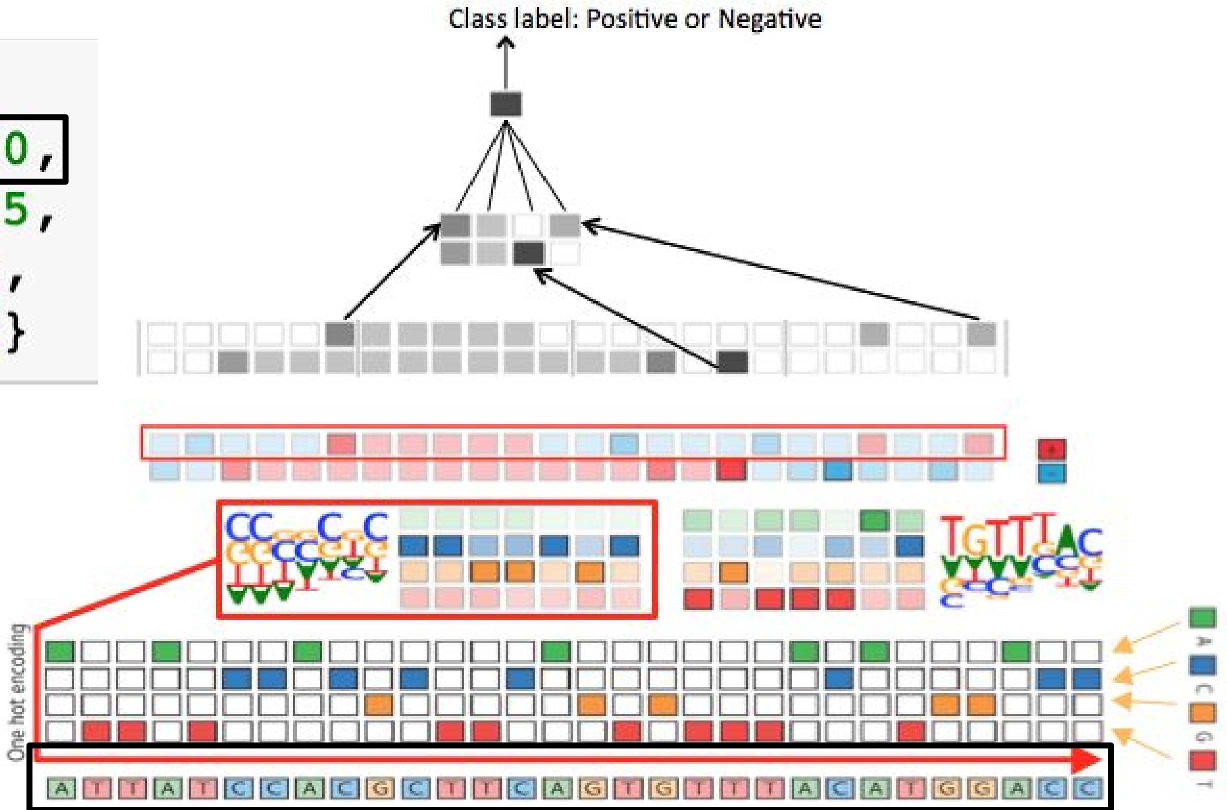
Convolutional Neural Networks


Class label: Positive or Negative
Fully connected classification layer: computes a probability using a logistic regression on the max pooling layer outputs
Max pooling layer: Partition convolutional filter outputs and keep maximum output in each partition
ReLU transformation: set negative motif scans to 0
Scan sequence using with convolutional filters
Convolutional layer: a set of convolutional filters similar to position specific scoring matrices (PSSMs)
One hot encoding
+
-
A
C
G
T
A T T A T C C A C G C T T C A G T G T T T A C A T G G A C C

Defining SequenceDNN models



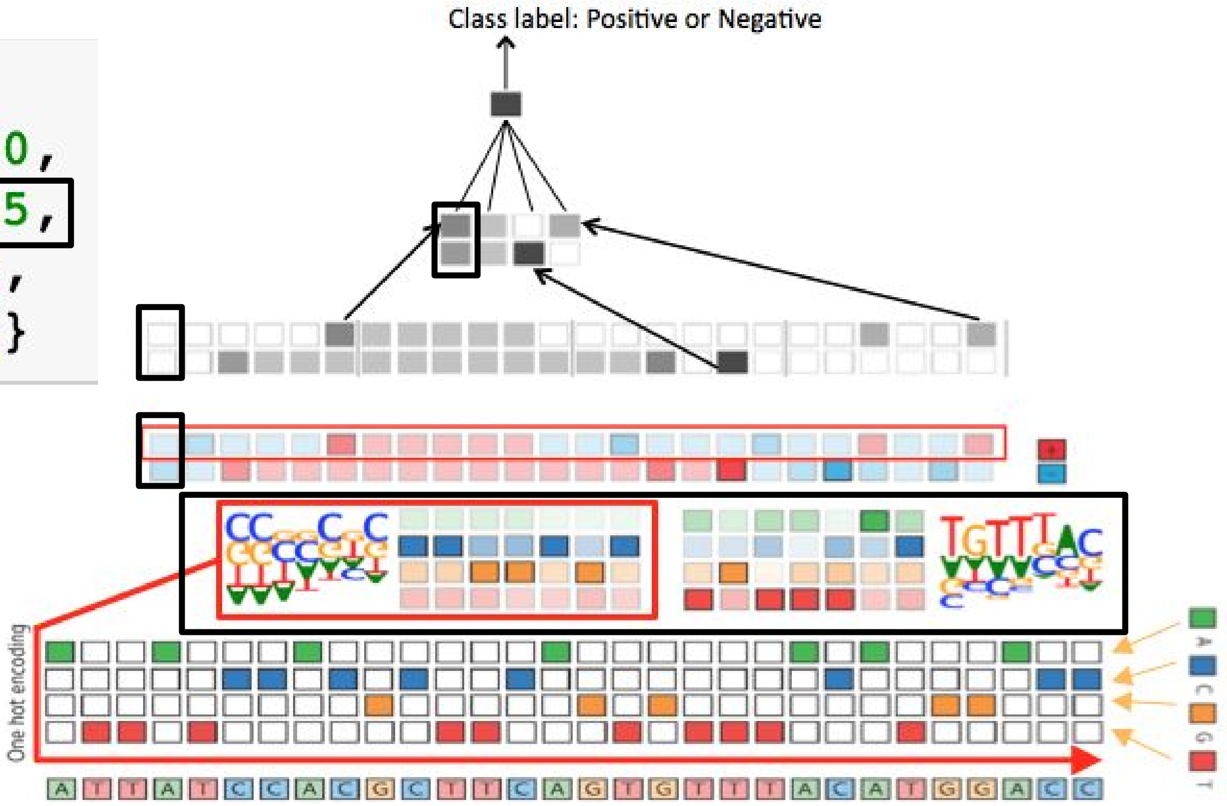
```
cnn_parameters = {
    'seq_length': 500,
    'num_filters': 15,
    'conv_width': 25,
    'pool_width': 35}
```



Class label: Positive or Negative
One hot encoding
A
C
G
T
A T T A T C C A C G C T T C A G T G T T T A C A T G G A C C

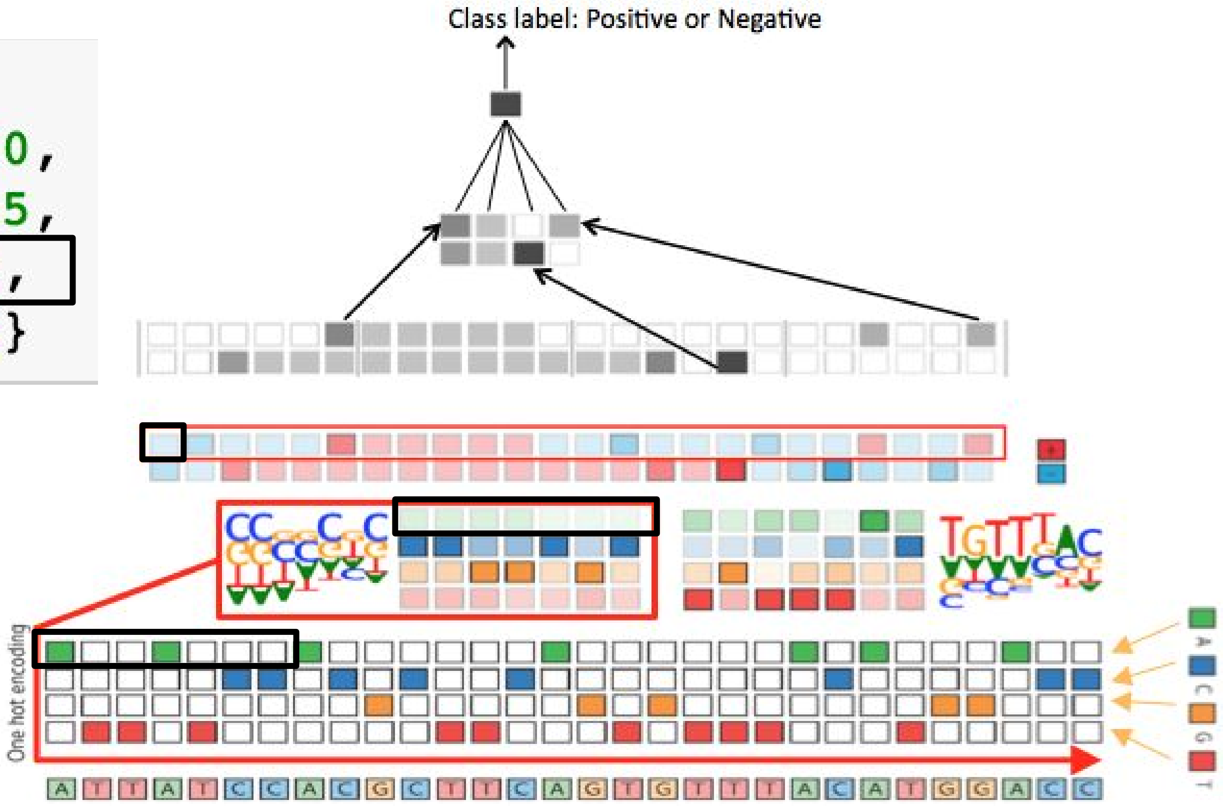
Defining SequenceDNN models

```
cnn_parameters = {
        'seq_length': 500,
        'num_filters': 15,
        'conv_width': 25,
        'pool_width': 35}
```


Class label: Positive or Negative
One hot encoding
A
C
G
T
A T T A T C C A C G C T T C A G T G T T T A C A T G G A C C

Defining SequenceDNN models

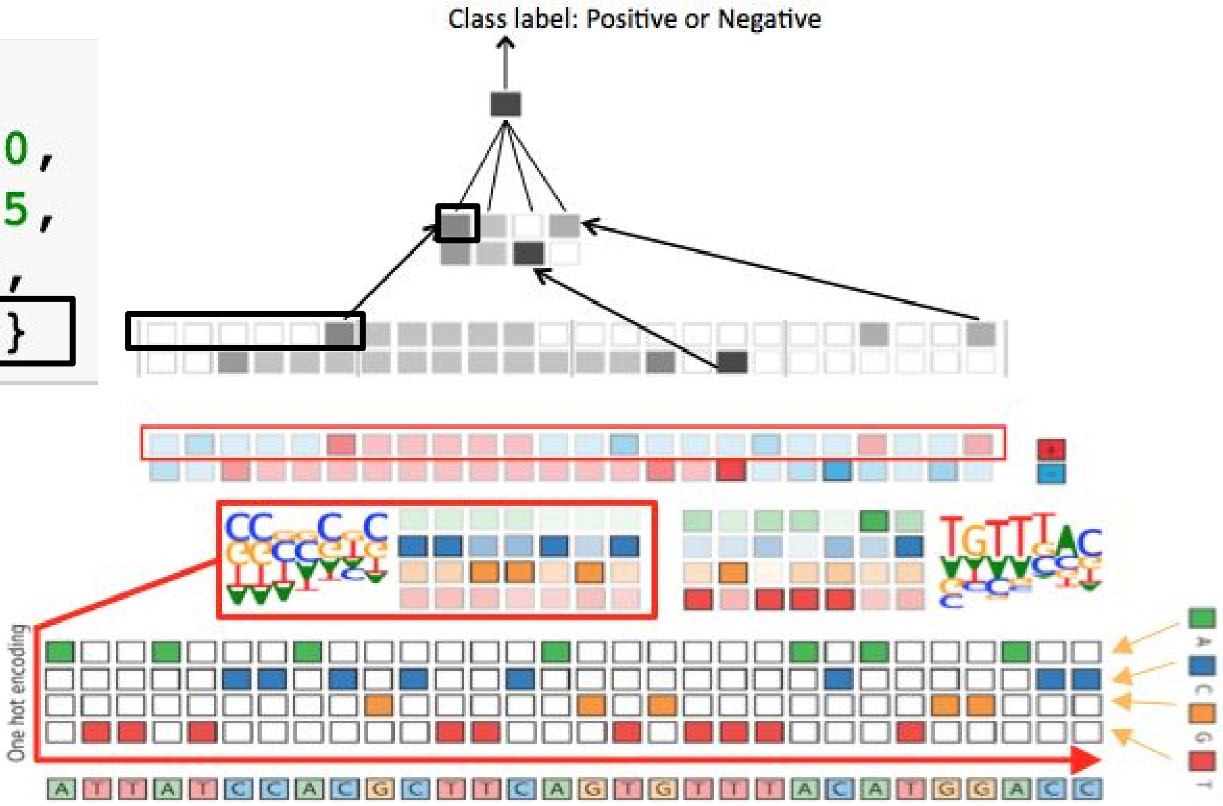
```
cnn_parameters = {
        'seq_length': 500,
        'num_filters': 15,
        'conv_width': 25,
        'pool_width': 35}
```



Class label: Positive or Negative
CC C C
TGTTTAC
One hot encoding
A
C
G
T
A T T A T C C A C G C T T C A G T G T T T A C A T G G A C C

Defining SequenceDNN models

```
cnn_parameters = {
        'seq_length': 500,
        'num_filters': 15,
        'conv_width': 25,
        'pool_width': 35}
```

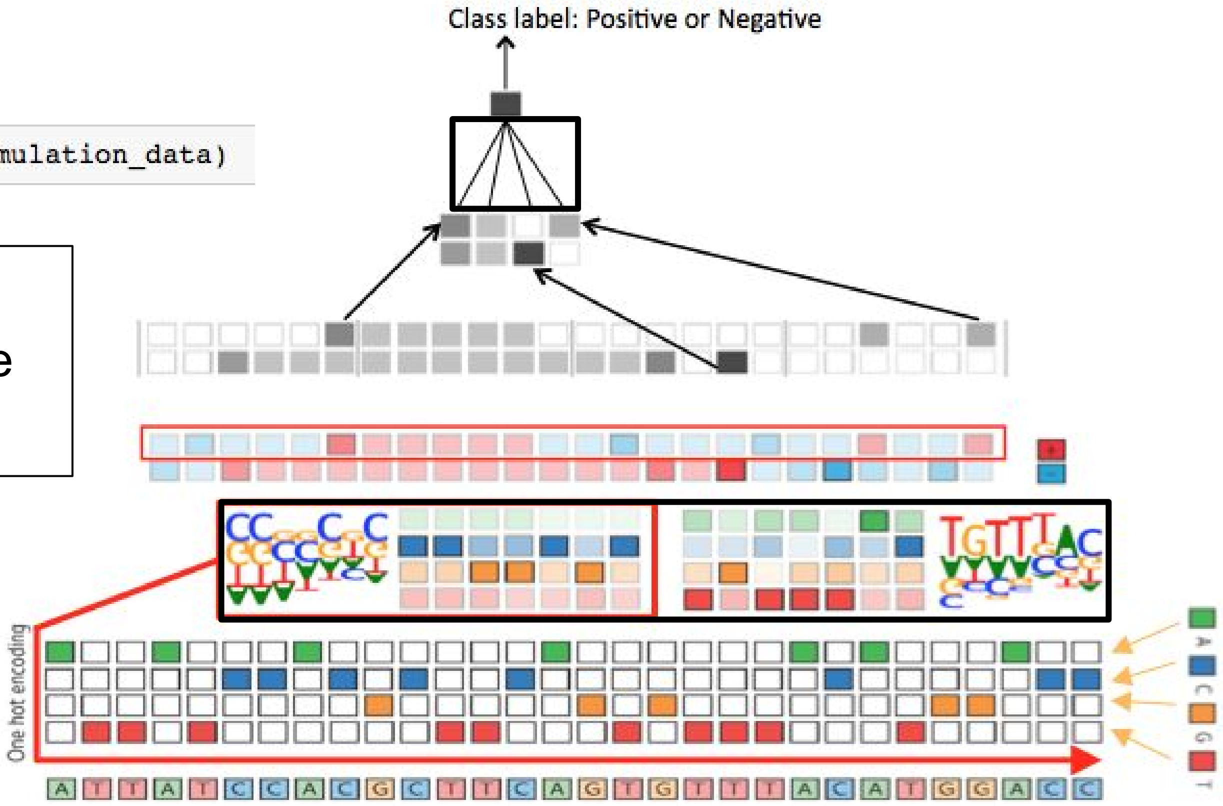

Class label: Positive or Negative
One hot encoding
A
C
G
T
A T T A T C C A C G C T T C A G T G T T T A C A T G G A C C

Training a DNN model

```
In [9]: cnn_model = get_SequenceDNN(cnn_
```

```
In [10]: train_SequenceDNN(cnn_model, simulation_data)
```

Training: Iterative updates to model parameters to minimize the “loss function”

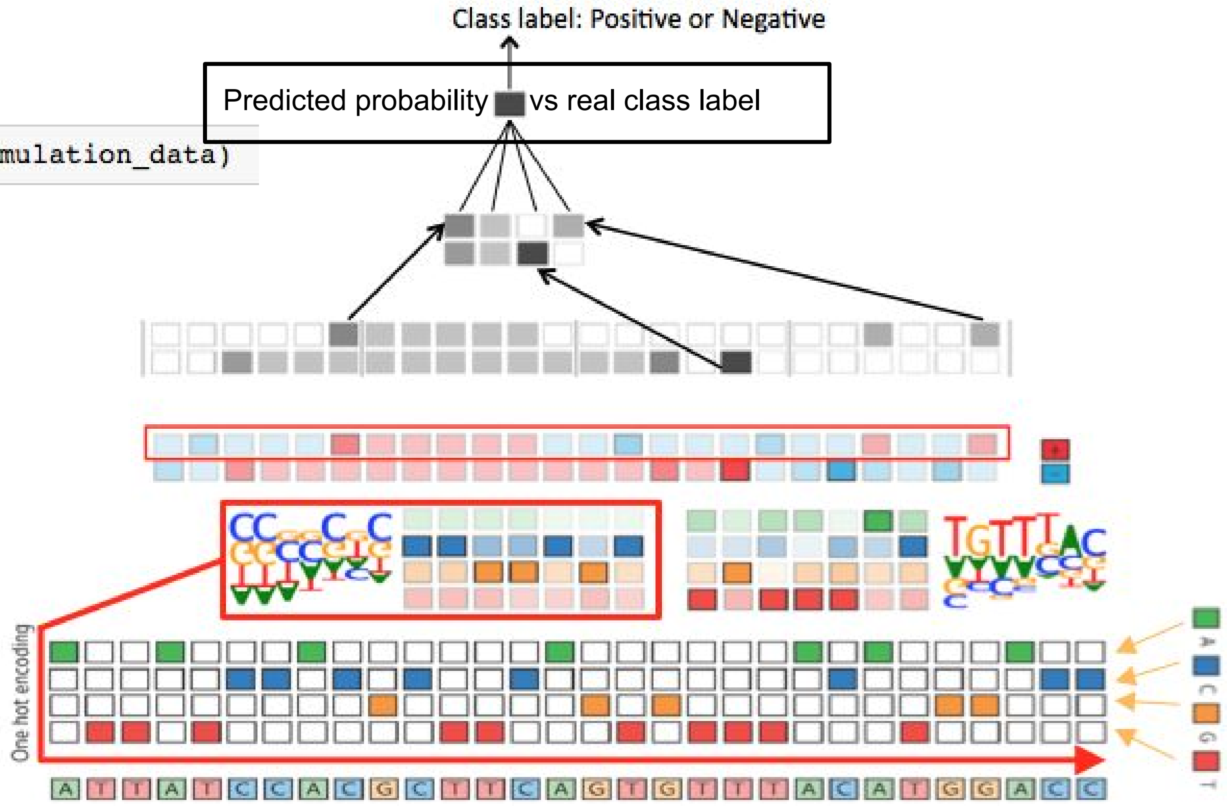

Class label: Positive or Negative
One hot encoding
A
C
G
T
A T T A T C C A C G C T T C A G T G T T T A C A T G G A C C

Training a DNN model

The loss: quantifies error in model predictions

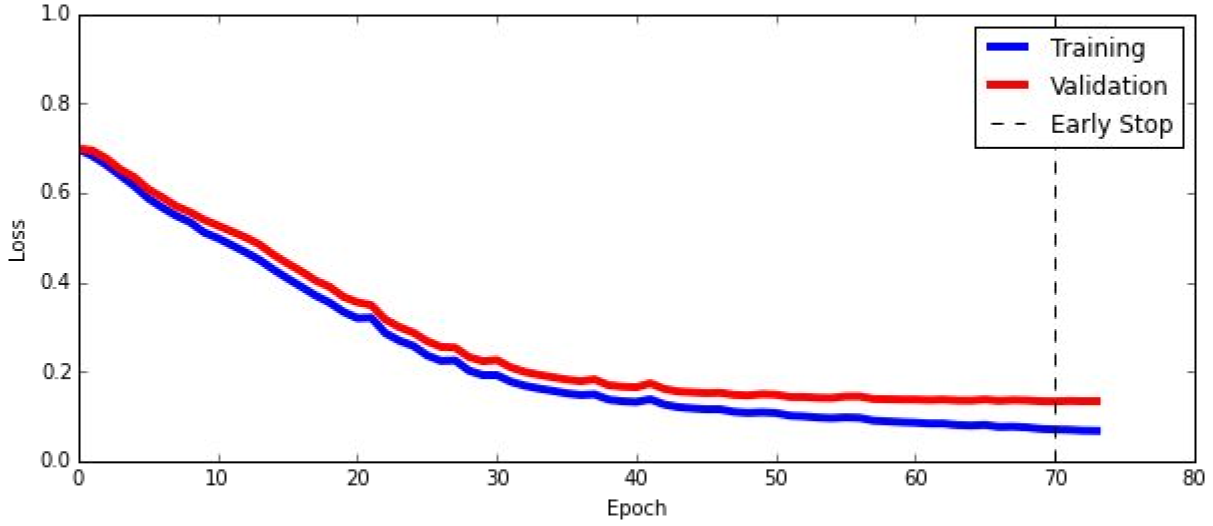
```
In [9]: cnn_model = get_SequenceDNN(cnn_
```

```
In [10]: train_SequenceDNN(cnn_model, simulation_data)
```


Class label: Positive or Negative
Predicted probability vs real class label
One hot encoding
A
C
G
T
A T T A T C C A C G C T T C A G T G T T T A C A T G G A C C

When to stop training?

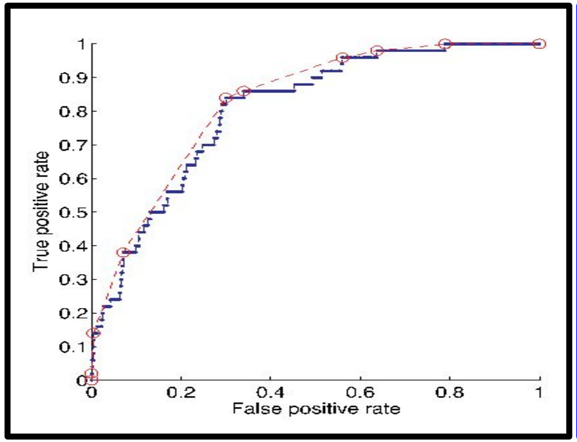
```
In [11]: SequenceDNN_learning_curve(cnn_model)
```

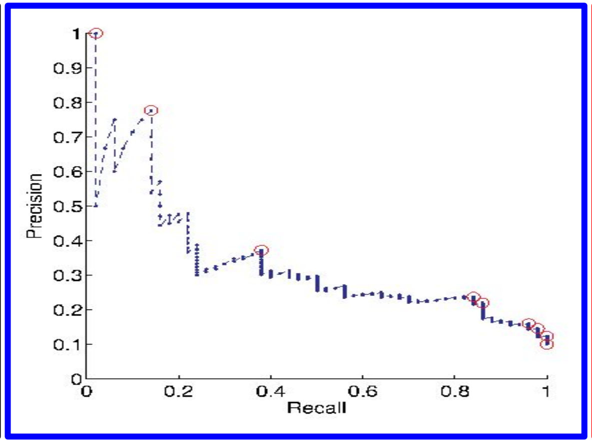


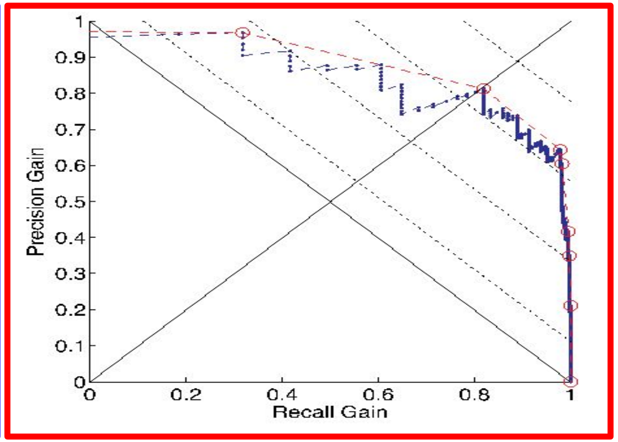
Performance Metrics

```
Epoch 0: validation loss: 0.699
Balanced Accuracy: 52.92%          auROC: 0.544     auPRC: 0.557     auPRG: 0.074
Recall at 5%|10%|20% FDR: 0.3%|0.3%|0.5%           Num Positives: 1645    Num Negatives: 1555
```

```
Epoch 73: validation loss: 0.134
Balanced Accuracy: 95.86%          auROC: 0.986     auPRC: 0.988     auPRG: 0.983
Recall at 5%|10%|20% FDR: 95.7%|97.0%|98.4%        Num Positives: 1645    Num Negatives: 1555
```







```
In [12]: test_SequenceDNN(cnn_model, simulation_data)

Test performance:
Balanced Accuracy: 95.88%      auROC: 0.987    auPRC: 0.988    auPRG: 0.984
Recall at 5%|10%|20% FDR: 95.8%|97.3%|98.5%    Num Positives: 2017    Num Negatives: 1983
```

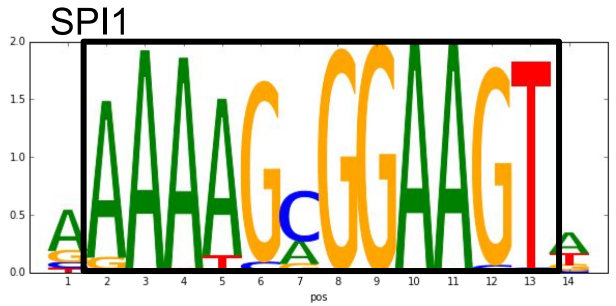
Taken from Flach, Peter and Kull, Meeli. NIPS (2015).

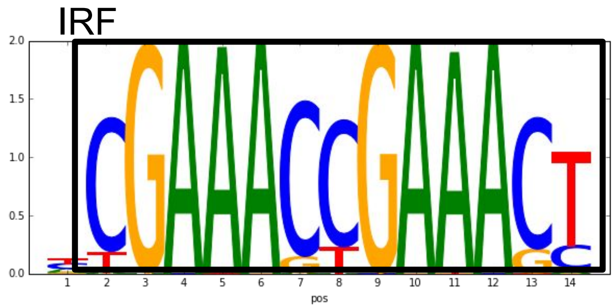
Interpreting DNN models: two broad approaches

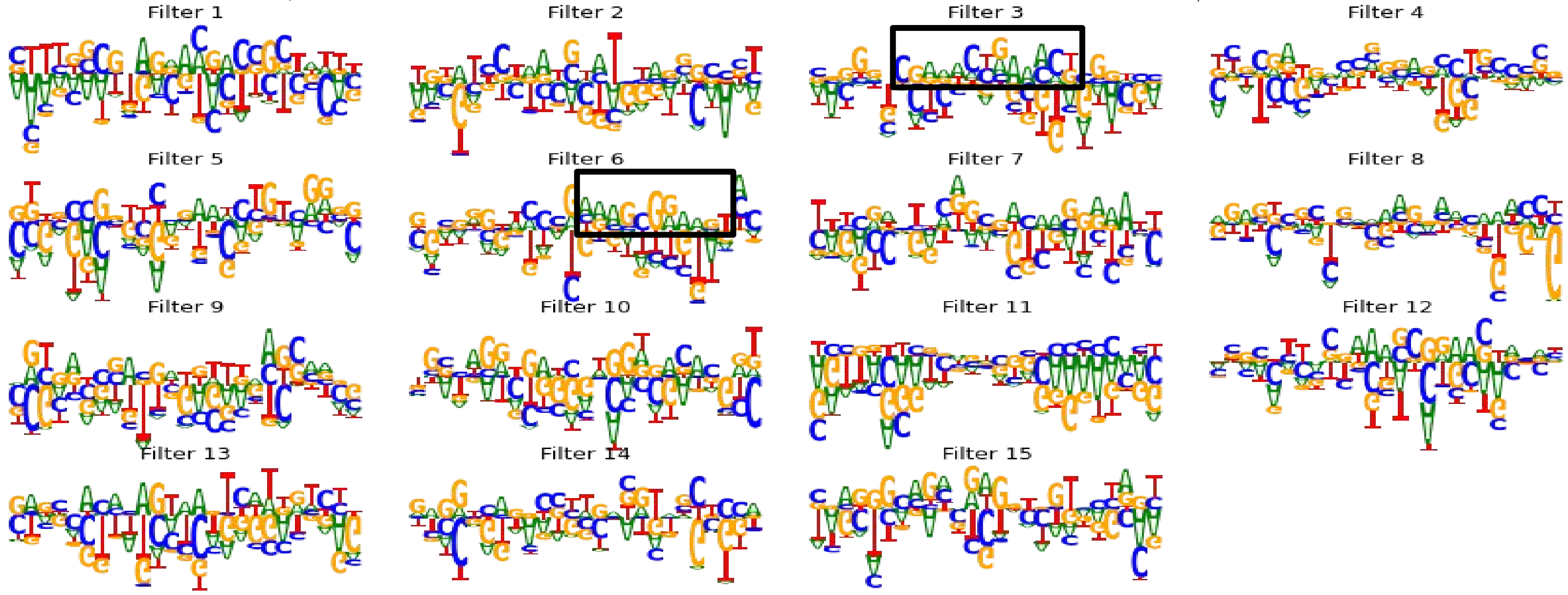
1. Model-centered approach: interpret model parameters directly
    - Example: inspect learned convolutional filters and try to infer sequence motifs from them

2. Input sequence-centered approach: sequence-specific model activity
    - Example: propagate input sequence through the model, inspect outputs in convolutional and max pooling layer, try to infer sequence properties from those output

Model-centered Interpretation





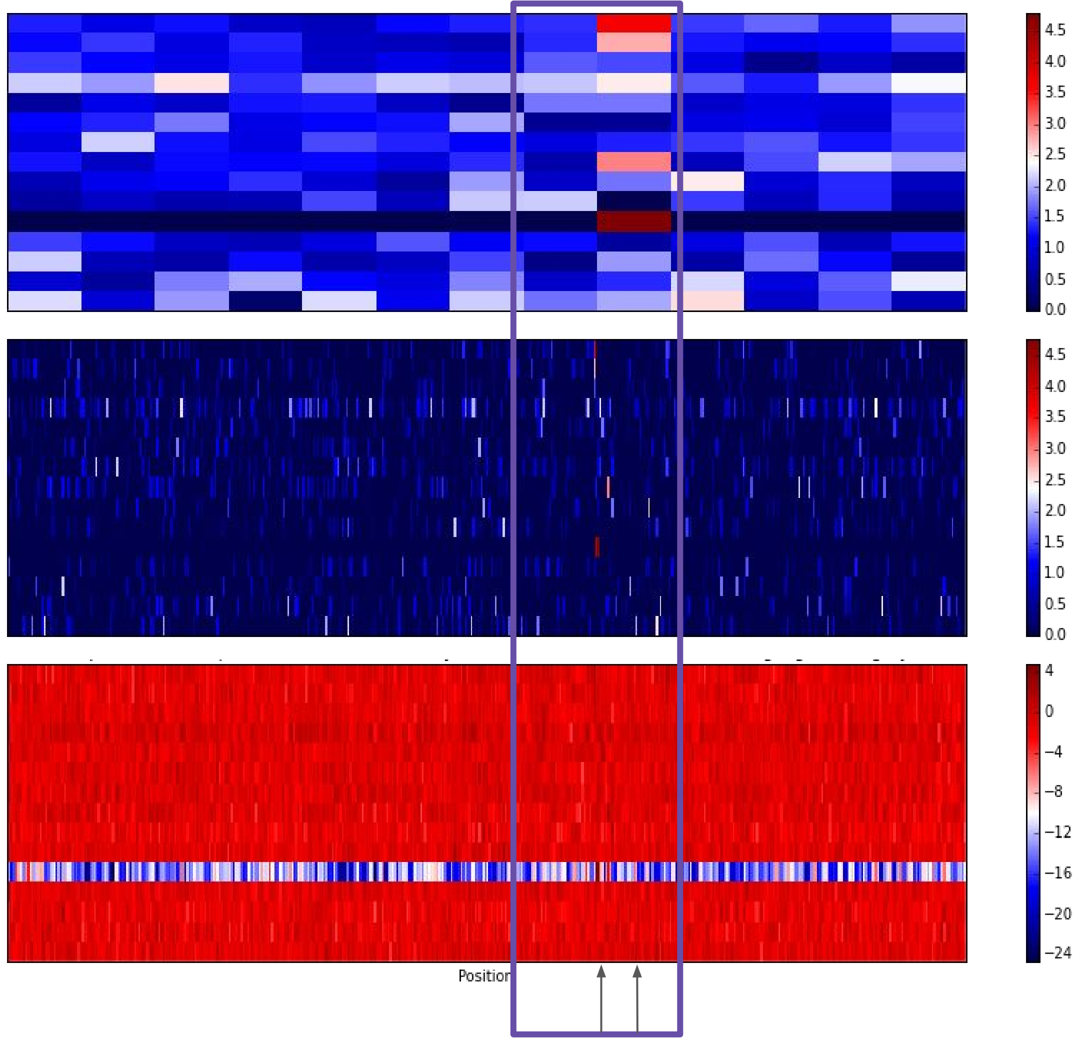


Input sequence centered Interpretation

15 Convolutional Filters, post max pooling

15 Convolutional Filters, post ReLU

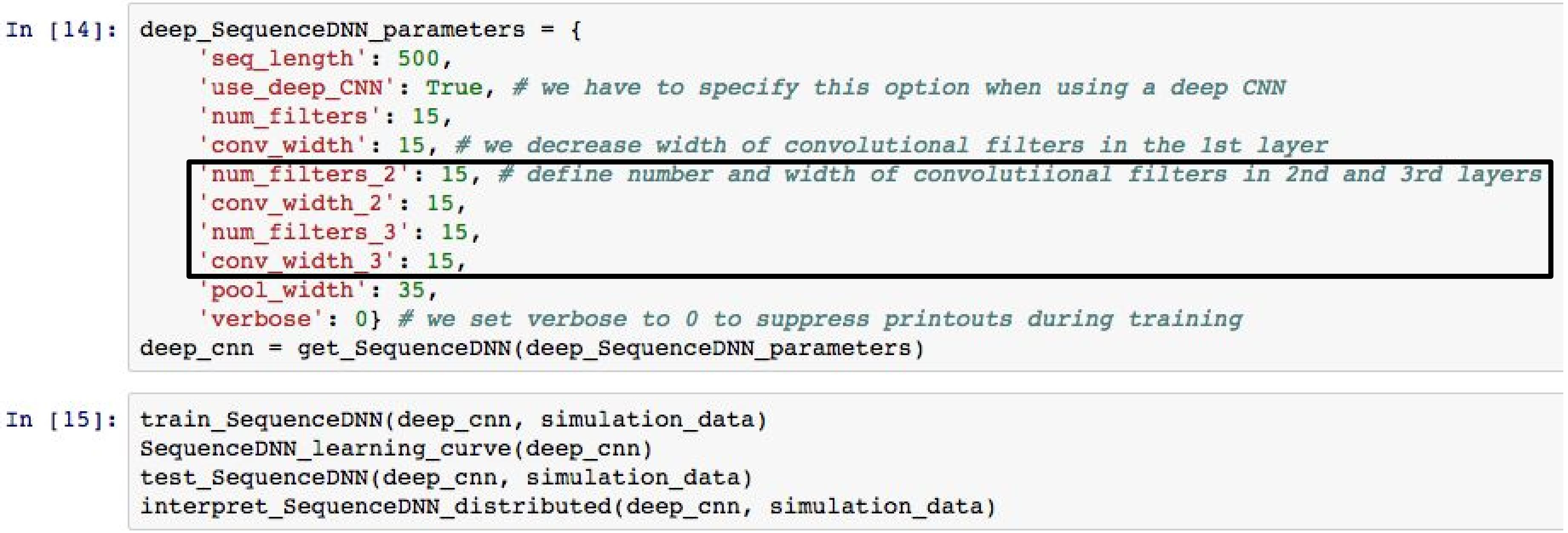
15 Convolutional Filters, pre ReLU



Motif sites

WE HAVE TO GO
DEEPER!
DIYLOL.COM

Repeat with a deeper 3-layered CNN

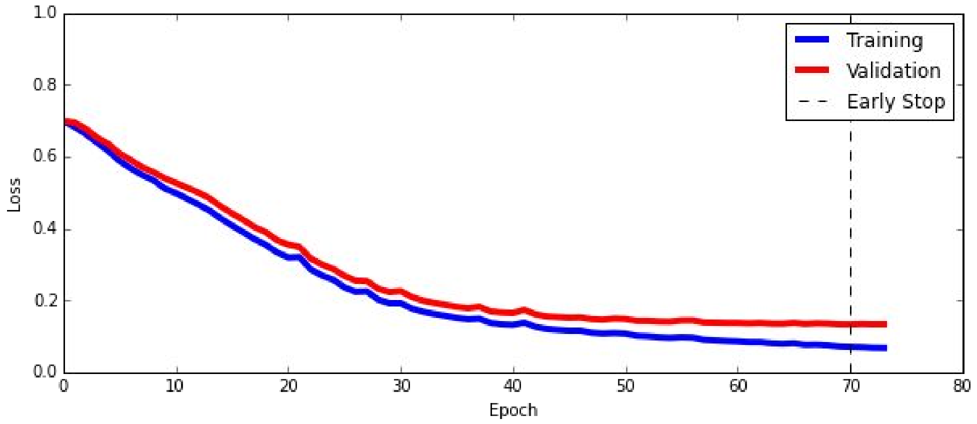


```
In [14]: deep_SequenceDNN_parameters = {
             'seq_length': 500,
             'use_deep_CNN': True, # we have to specify this option when using a deep CNN
             'num_filters': 15,
             'conv_width': 15, # we decrease width of convolutional filters in the 1st layer
             'num_filters_2': 15, # define number and width of convolutiional filters in 2nd and 3rd layers
             'conv_width_2': 15,
             'num_filters_3': 15,
             'conv_width_3': 15,
             'pool_width': 35,
             'verbose': 0} # we set verbose to 0 to suppress printouts during training
         deep_cnn = get_SequenceDNN(deep_SequenceDNN_parameters)
```

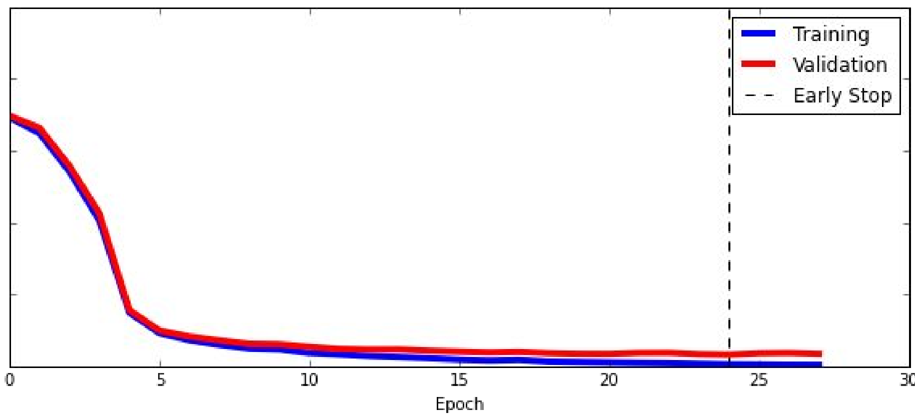
```
In [15]: train_SequenceDNN(deep_cnn, simulation_data)
         SequenceDNN_learning_curve(deep_cnn)
         test_SequenceDNN(deep_cnn, simulation_data)
         interpret_SequenceDNN_distributed(deep_cnn, simulation_data)
```

Faster and Better Learning

Shallow CNN

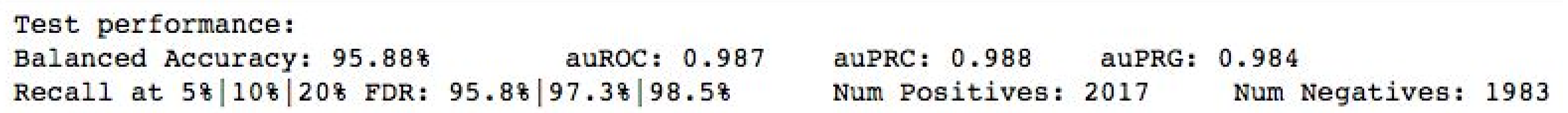


Deep CNN



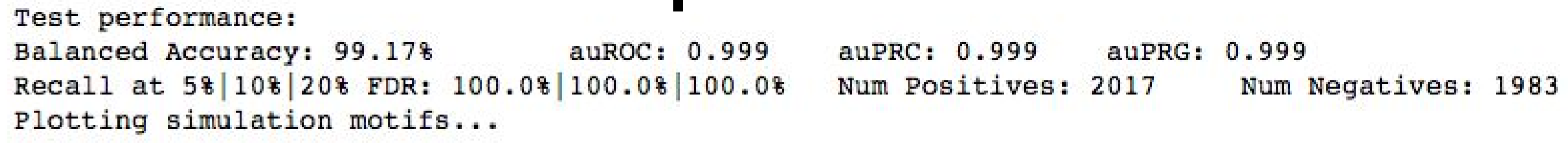
Better Test Performance Metrics

Shallow CNN



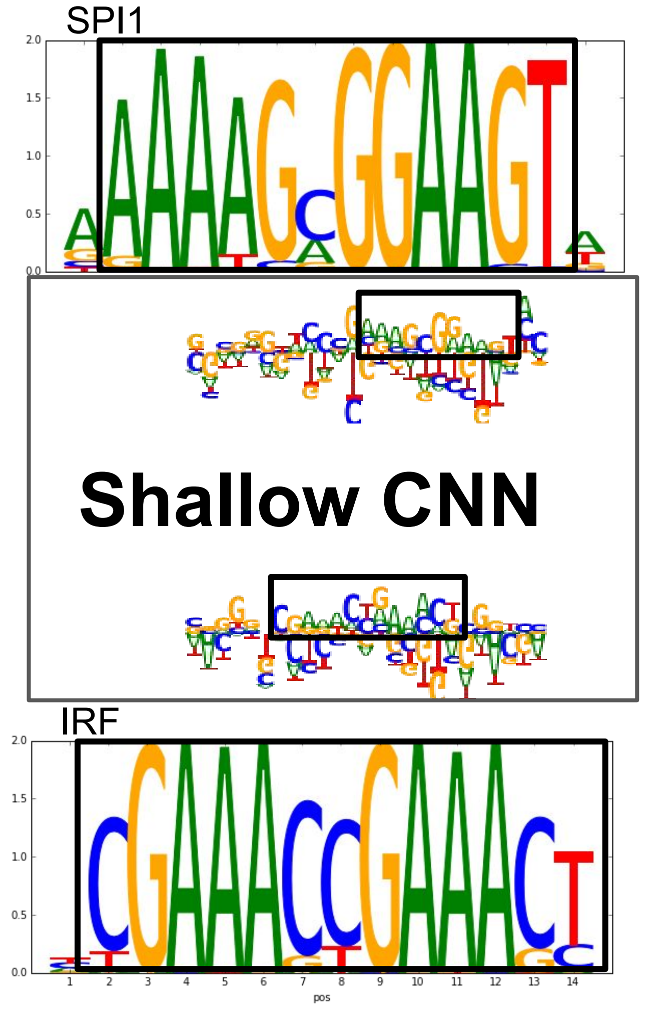
```
Test performance:
Balanced Accuracy: 95.88%        auROC: 0.987    auPRC: 0.988    auPRG: 0.984
Recall at 5%|10%|20% FDR: 95.8%|97.3%|98.5%      Num Positives: 2017     Num Negatives: 1983
```

Deep CNN

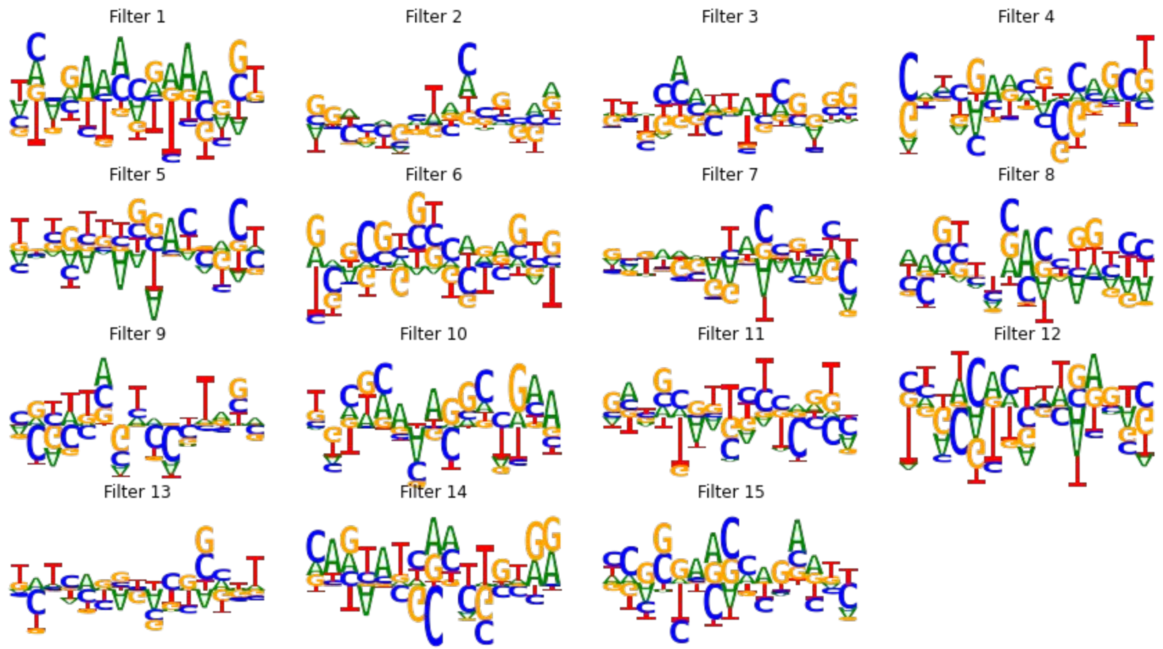


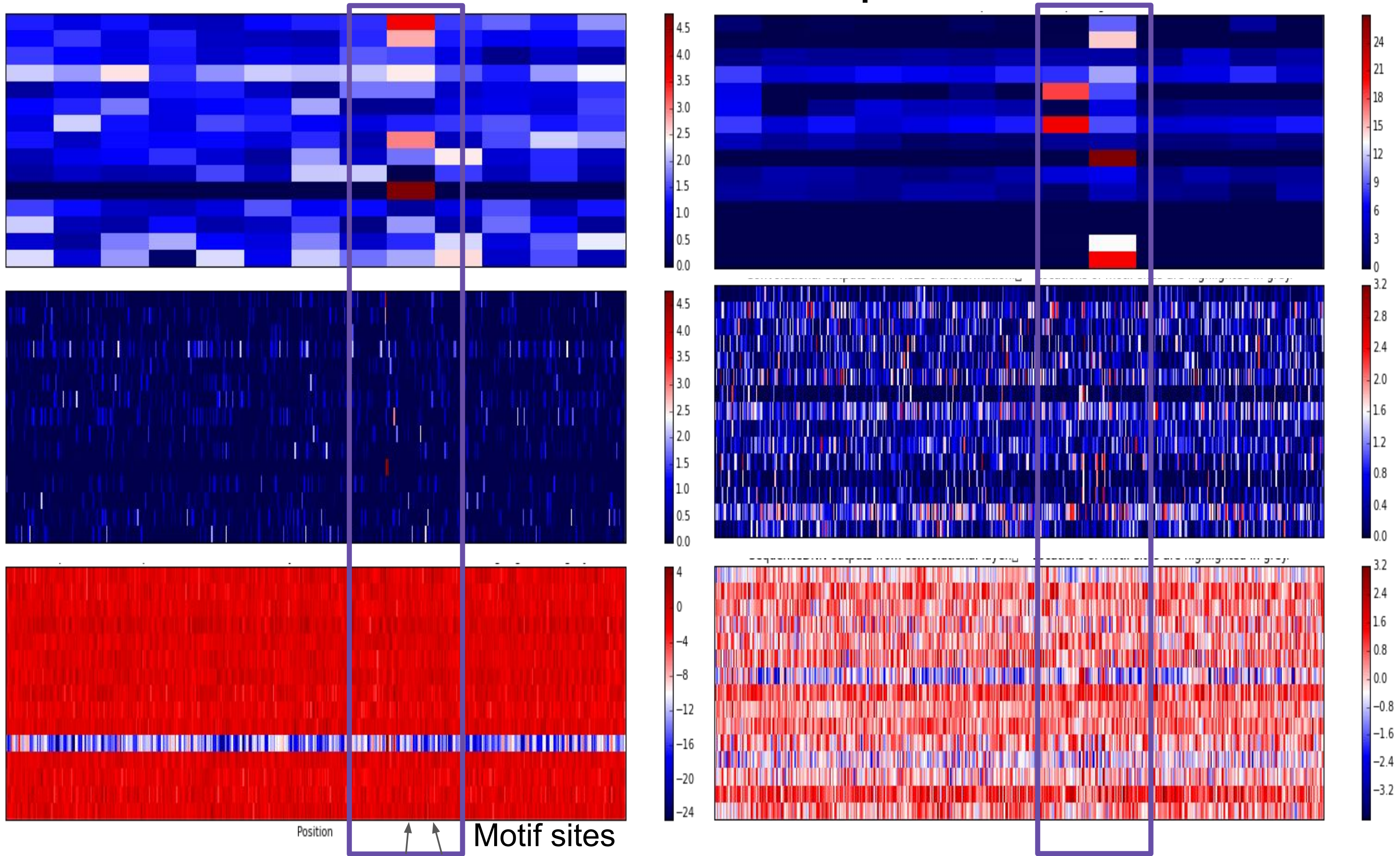
```
Test performance:
Balanced Accuracy: 99.17%        auROC: 0.999    auPRC: 0.999    auPRG: 0.999
Recall at 5%|10%|20% FDR: 100.0%|100.0%|100.0%   Num Positives: 2017     Num Negatives: 1983
Plotting simulation motifs...
```

Model-centered Interpretation



Deep CNN




Shallow CNN
Deep CNN
15 Convolutional Filters, post max pooling
15 Convolutional Filters, post ReLU
15 Convolutional Filters, pre ReLU
Position
Motif sites

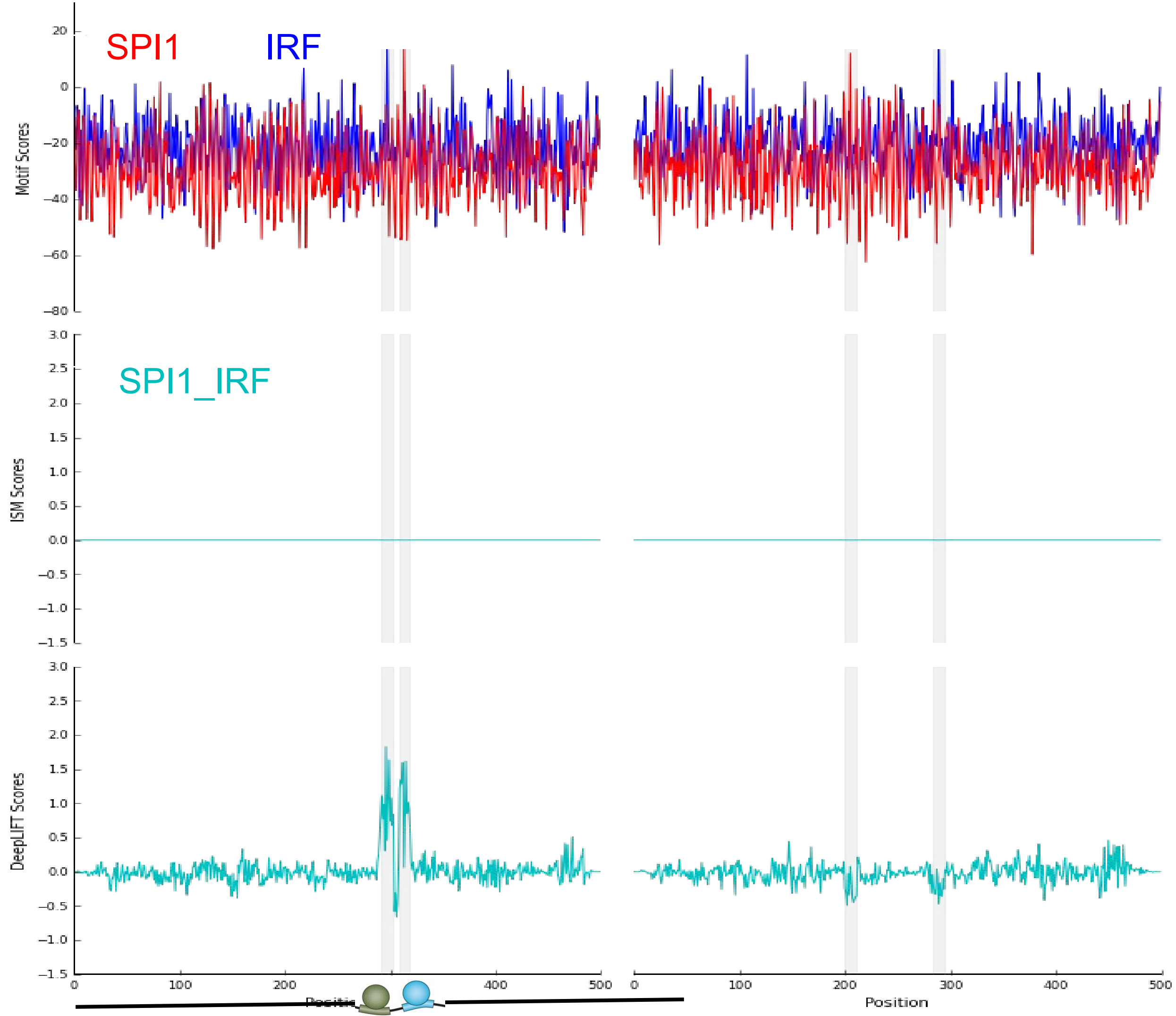
THAT'S NOT ENOUGH
WE HAVE TO GO DEEPER
quickmeme.com

Integrative Interpretation of DNN Models

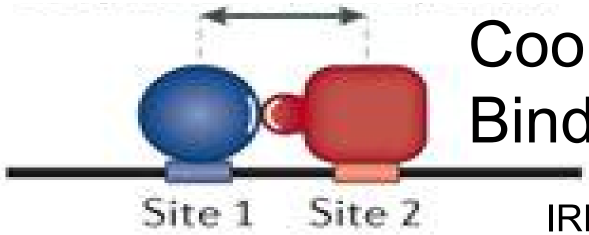
- Interpretation through internal layers DNN layers, both model-centric and input sequence-centric, suffers from the distributed nature of DNNs

- Solution: “integrate” using DeepLIFT and in-silico mutagenesis (ISM)
  - DeepLIFT: score each nucleotide based on its net contribution to the final fully connected layer, integrating across all filters and layers in between
  - ISM: mutate one nucleotide at a time, compute difference in prediction, score based on average difference in prediction

Integrative Input sequence scores

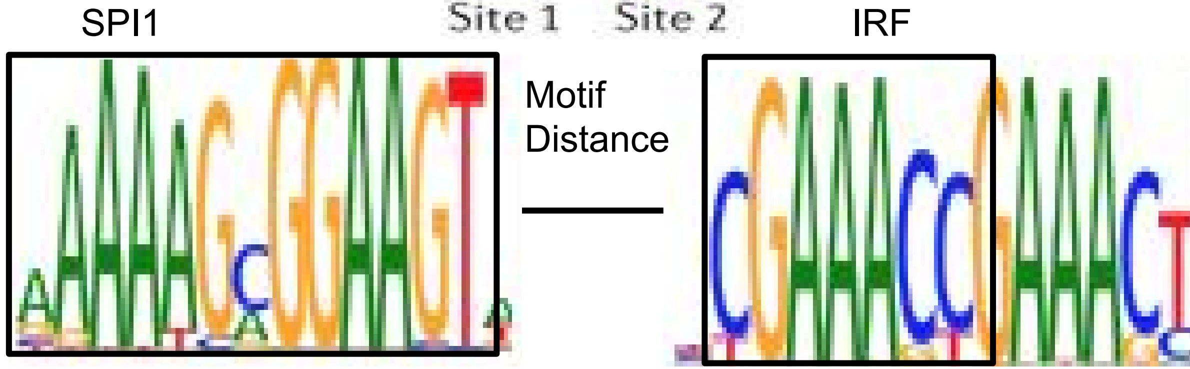


DeepLIFT can recover complex properties from DNN models!

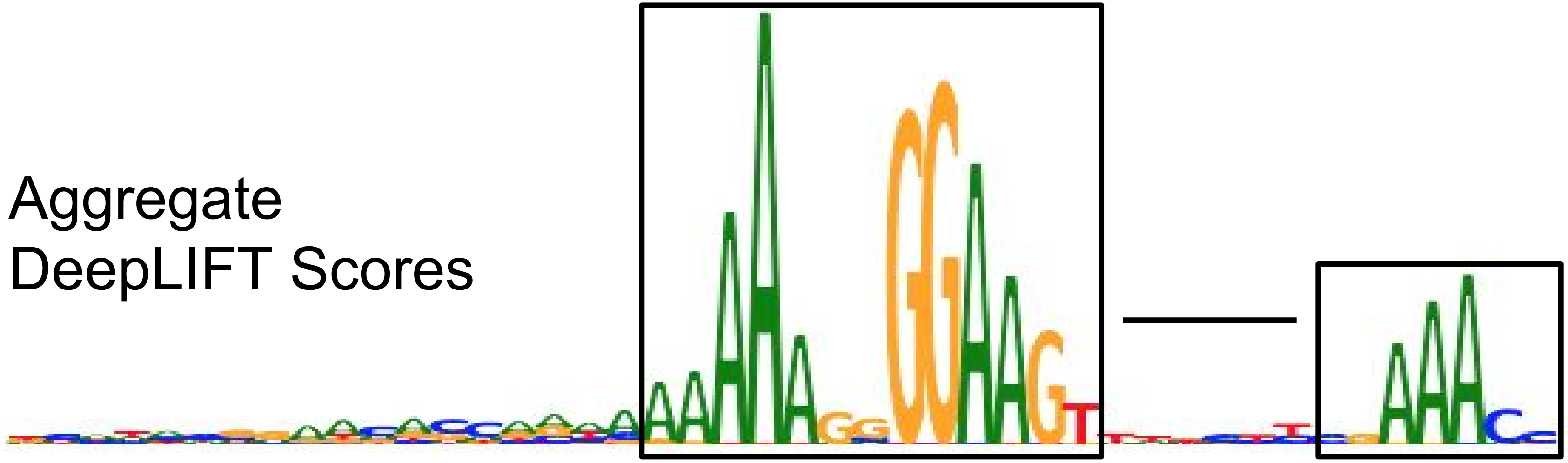


Cooperative Binding

Simulated Motifs



Aggregate DeepLIFT Scores



Extras

Investigate the one layered CNN model used here for the following simulations:

1. single motif detection simulation of TAL1 in 1000bp sequence with 40% GC content
2. motif density localization simulation of 2-4 TAL1 motif instances in the central of 150bp of a total 1000bp sequence with 40% GC

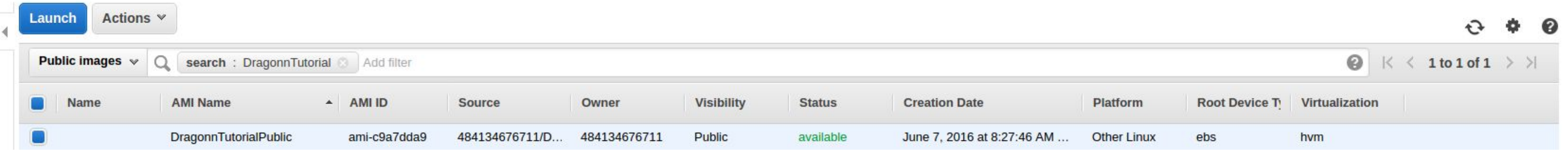
Key questions:

1) What could explain the difference in ISM's sensitivity to the TAL1 motif sequence between the simulations?
2) What does that tell us about the the scope of ISM for feature discovery? Under what conditions is it likely to show sensitivity to sequence features?

Starter code is provided in the tutorial notebook.

To access this tutorial on Amazon AWS

1. Create an account on Amazon Web Services: www.aws.amazon.com/signin
2. You will need to launch an EC2 instance using the public AMI "**DragonnTutorialPublic**"
3. Go to Services > EC2 > AMIs
4. Select "Public Images"
5. In the search bar, enter "DragonnTutorialPublic"
6. Click "Launch" and follow the instructions. Note: you must select instance type "g2.2xlarge" or "g2.8xlarge" to create an instance with GPU's

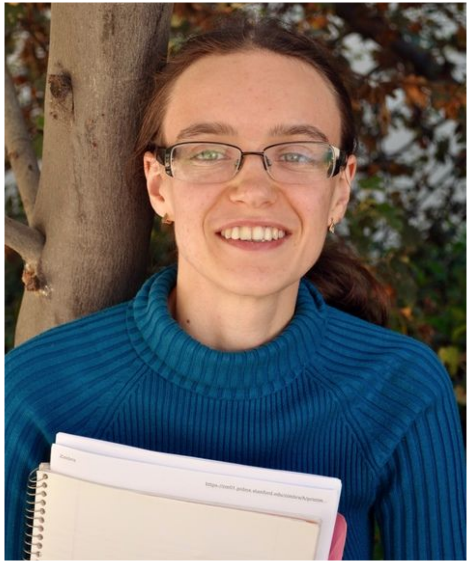


Name	AMI Name	AMI ID	Source	Owner	Visibility	Status	Creation Date	Platform	Root Device Ty	Virtualization
	DragonnTutorialPublic	ami-c9a7dda9	484134676711/D...	484134676711	Public	available	June 7, 2016 at 8:27:46 AM ...	Other Linux	ebs	hvm

Acknowledgements

Deep Learners at Kundajelab:

Avanti Shrikumar
Nathan Boley
Peyton Greenside
Chris Probert
Nasa Armstrong
Irene Kaplow
Michael Wainberg
Oana Ursu
Rahul Mohan


Anna Shcherbina

Chuan Sheng Foo

Anshul Kundaje