DeepBind

Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning

Nature Biotechnology 33, 831–838 (2015) doi:10.1038/nbt.3300

Method Details

Hayan Lee

Research Fellow @ Simons Institute for the Theory of Computing, UC Berkeley
Simons Postdoctral Fellow @ DOE JGI, Lawrence Berkeley National Laboratory

Outline

- Overview
- Forward propagation
- Backward propagation

Forward-propagation

Convolution Net

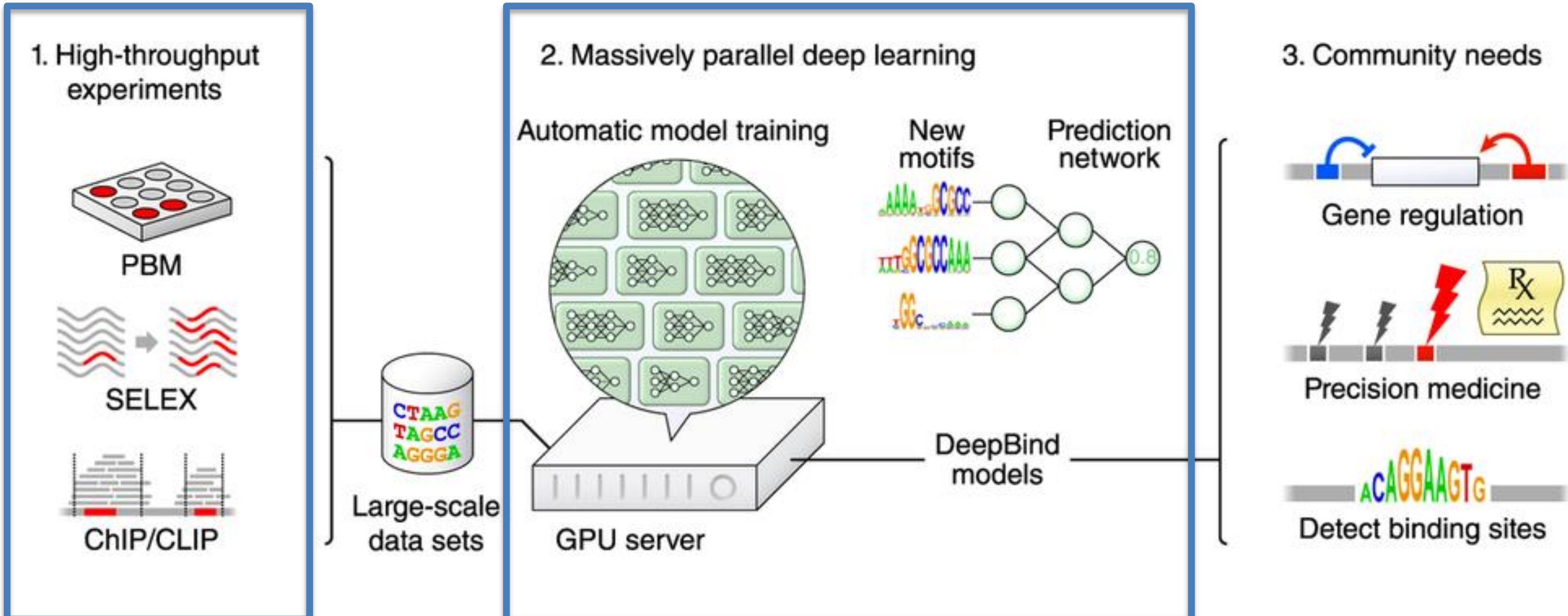
Rectification

Pooling

Neural Network

Back-propagation

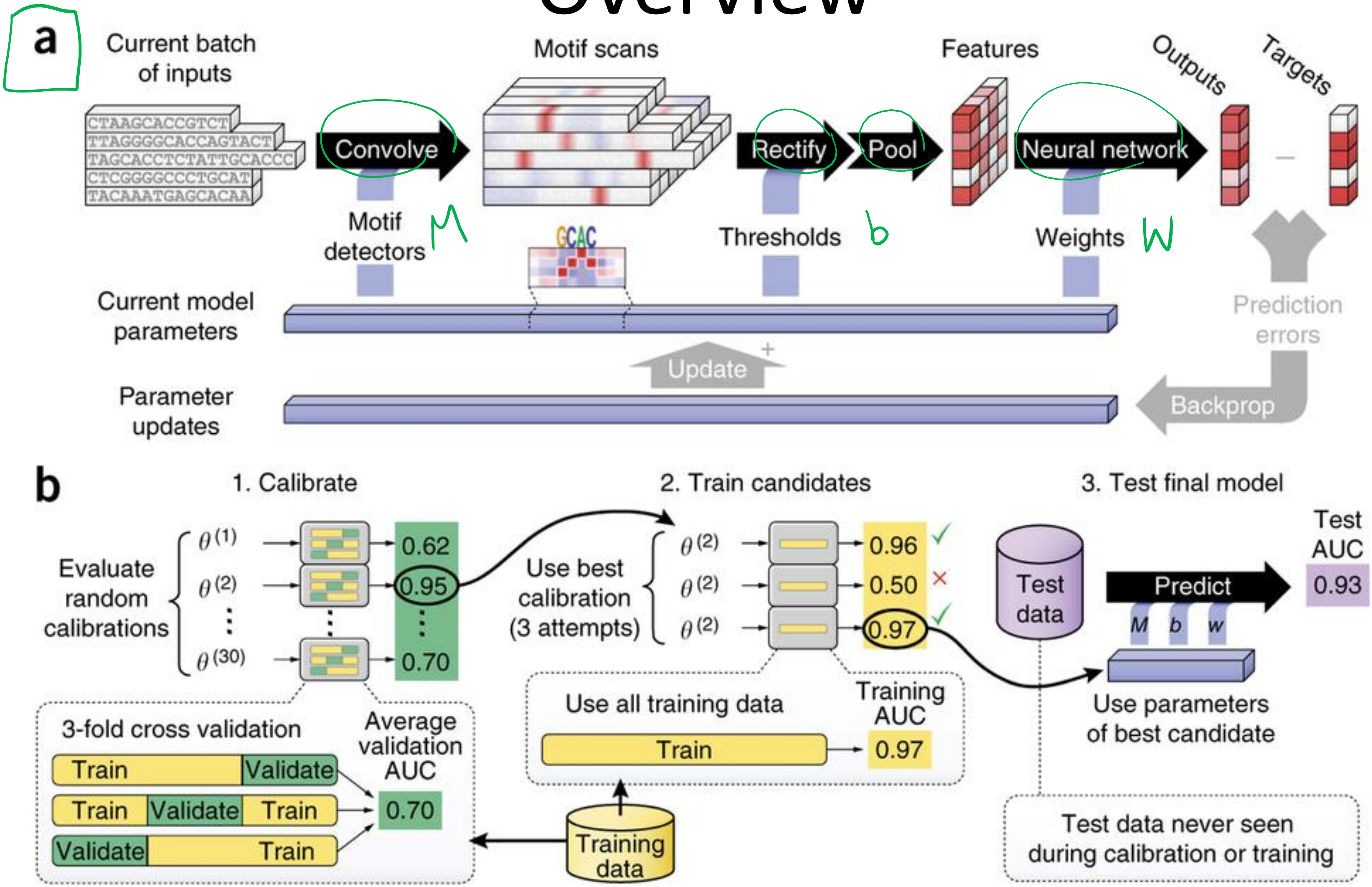
Overview



14–101 nt

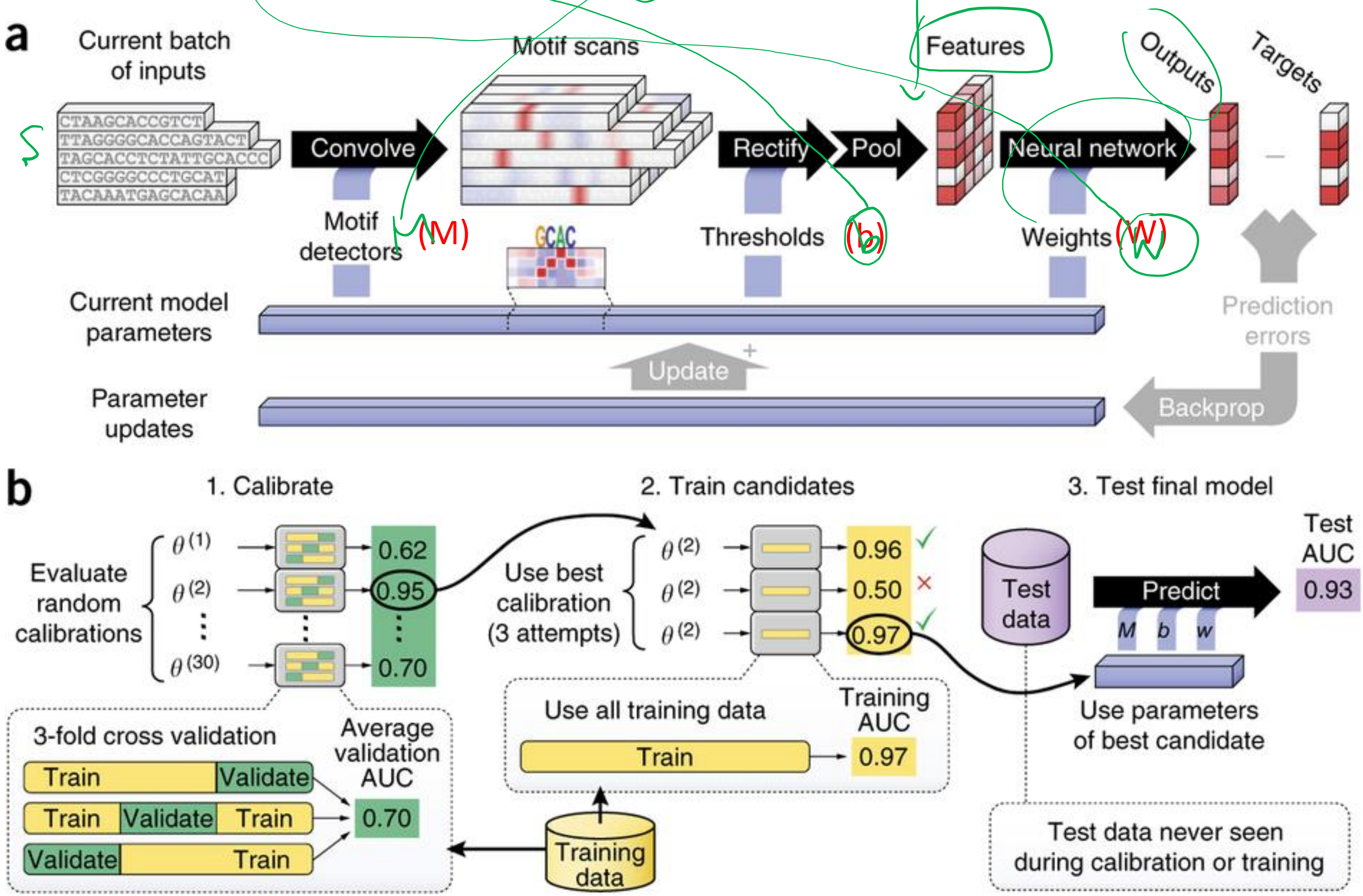
PBM : Protein binding microarrays

Overview


a
Current batch of inputs
CTAAGCACCGTCT
TTAGGGGCACCAGTACT
TAGCACCTCTATTGCACCC
CTCGGGGCCCTGCAT
TACAAATGAGCACAA
Convolve
Motif scans
Rectify
Pool
Features
Neural network
Outputs
Targets
Motif detectors
M
GCAC
Thresholds
b
Weights
W
Current model parameters
Prediction errors
Update
Parameter updates
Backprop
b
1. Calibrate
2. Train candidates
3. Test final model
Evaluate random calibrations
θ (1)
θ (2)
θ (30)
0.62
0.95
0.70
Use best calibration (3 attempts)
0.96
0.50
0.97
Test data
Test AUC
Predict
0.93
M
b
w
3-fold cross validation
Train
Validate
Average validation AUC
0.70
Use all training data
Training AUC
0.97
Training data
Use parameters of best candidate
Test data never seen during calibration or training

$$f(s) = net_w(pool(rect_b(conv_M(s))))$$

(M, b and W are trainable parameters)



Convolution

$$f(s) = net_w(pool(rect_b(conv_M(s))))$$

$s = (s_1, \ldots, s_n)$ Sequences can have varying lengths (14–101 nt in our experiments),

$conv_M(s)$ The convolution stage scans a set of motif detectors with parameters M across the sequence. Motif detector $\boldsymbol{M_k}$ is a $\boldsymbol{m \times 4}$ matrix, much like a PWM(Position Weight Matrices) of length $\boldsymbol{m}$ but without requiring coefficients to be probabilities or log odds ratios.

Convolution : Encoding & Padding

Input Sequence: ATGG

$$S_{i,j} = \begin{cases} .25 & \text{if } s_{i-m+1} = \text{N} \text{ or } i < m \text{ or } i > n-m \\ 1 & \text{if } s_{i-m+1} = j^{\text{th}} \text{ base in } (\text{A, C, G, T}) \\ 0 & \text{otherwise} \end{cases}$$

n is sequence length, 4

m is motif length = 3

$$\begin{array}{cc} & \begin{array}{cccc} A & C & G & T \end{array} \\ S = & \begin{bmatrix} .25 & .25 & .25 & .25 \\ .25 & .25 & .25 & .25 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ .25 & .25 & .25 & .25 \\ .25 & .25 & .25 & .25 \end{bmatrix} \end{array}$$

m-1=2

n=4

Σ=1

m-1

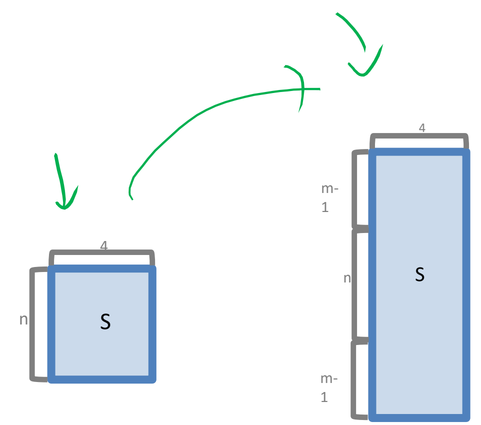
A T G G

m-1

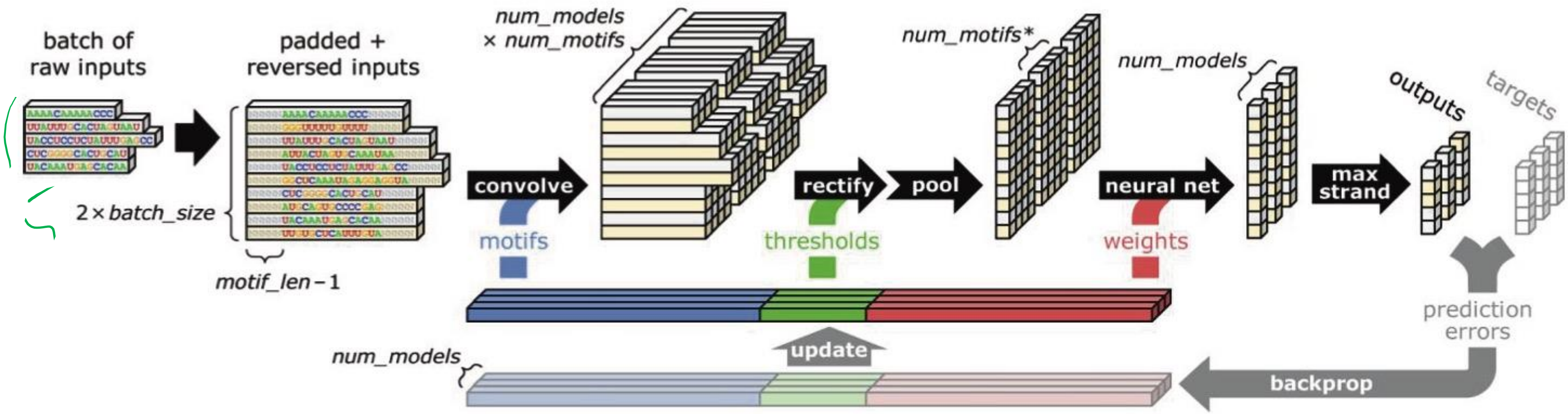
size(S) = n + 2(m-1)

Training and scoring sequences

Convolution



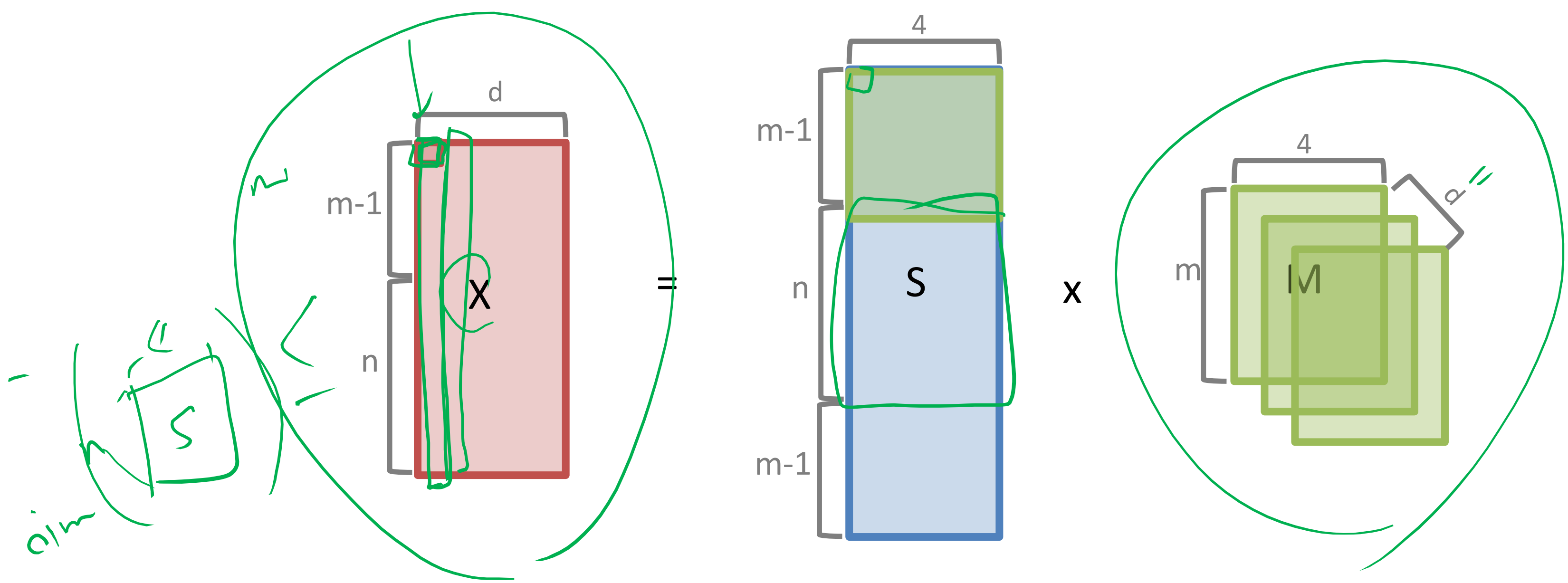
features



hyper parameters

batch_size=5, motif_len=6, num_motifs=4, num_models=3

Convolution



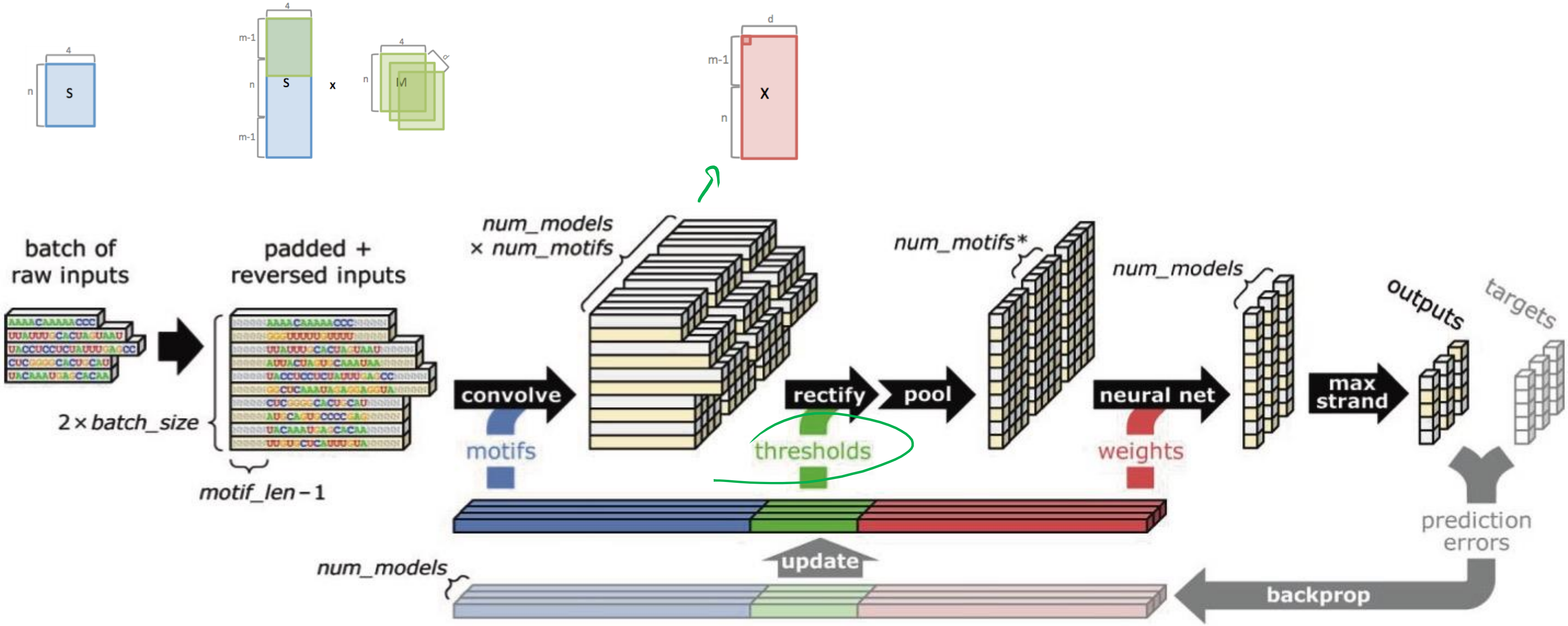
$$X_{i,k} = \sum_{j=1}^{m} \sum_{l=1}^{4} S_{i+j,l} M_{k,j,l}$$

$\sum S_{ij} M_{ij}$

- For all $1 \le i \le n+m-1$ and $1 \le k \le d$, where d is the number of motif detectors
- S (input sequences) : (n+2m-2) x 4
- M (motif detectors) : m x 4 x d
- X (output) : (n+m-1) x d

Training and scoring sequences

Rectification



batch_size=5, motif_len=6, num_motifs=4, num_models=3

Rectification

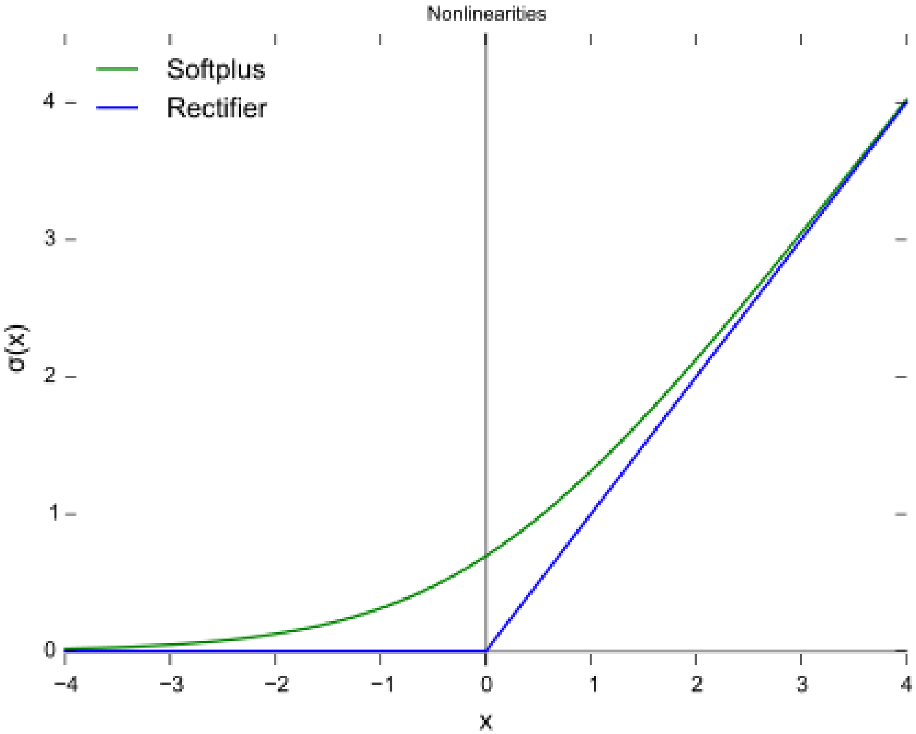
$$f(s) = net_w(pool(rect_b(conv_M(s))))$$

$rect_b$

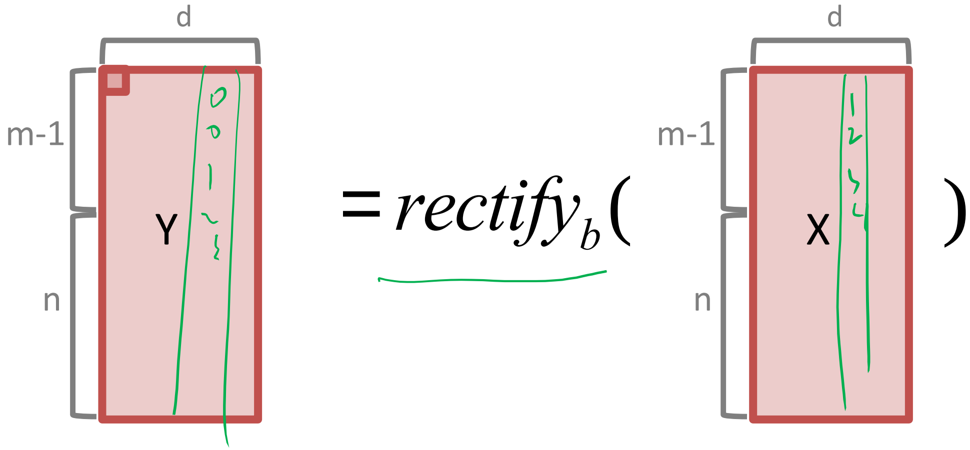
The rectification stage isolates positions with a good pattern match by shifting the response of detector $\boldsymbol{M_k}$ by $\boldsymbol{b_k}$ and clamping all negative values to zero.

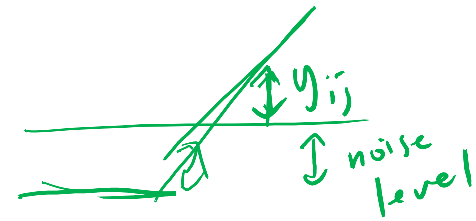
$$f(x) = \ln(1+e^x)$$

$$f(x) = \max(0, x)$$



Rectification





$$Y = rectify_b(X) \qquad \boxed{b = (b_1, \ldots, b_d)}$$

$$Y_{i,k} = \max(0, X_{i,k} - b_k)$$

- For all $1 \le i \le n+m-1$ and $1 \le k \le d$, where d is the number of motif detectors
- X (input) : (n+m-1) x 4
- Y (output) : (n+m-1) x 4

Pooling

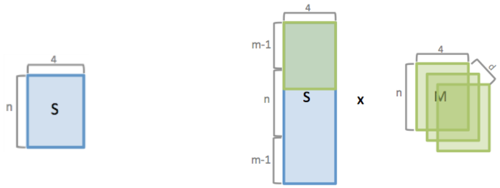
Feature Selection

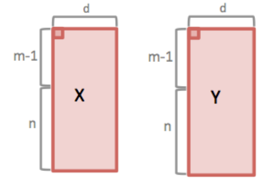
$$f(s) = net_w(pool(rect_b(conv_M(s))))$$

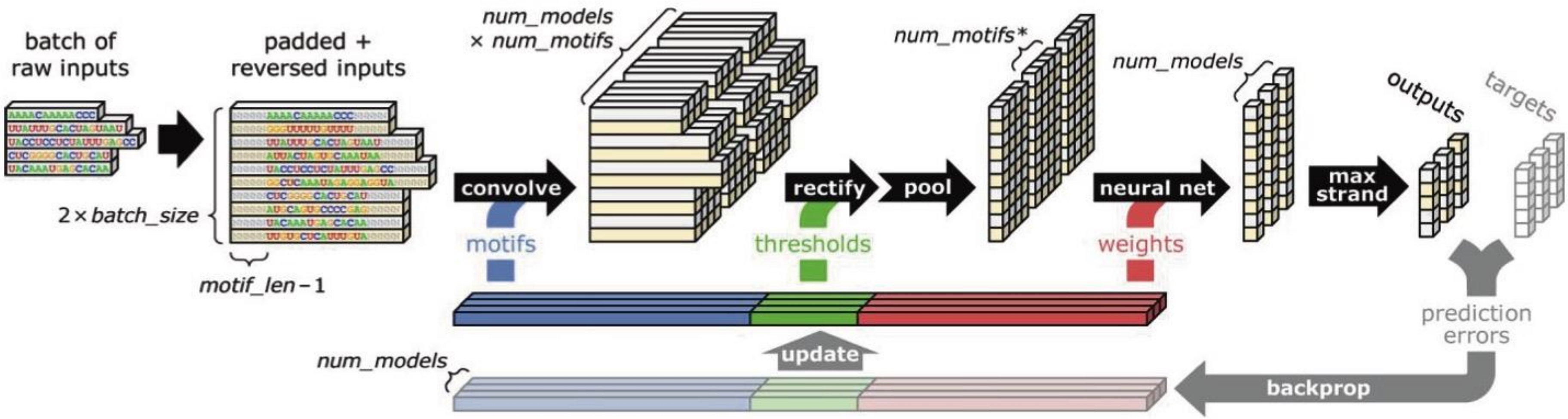
pool

The pooling stage computes the maximum and average of each motif detector's rectified response across the sequence; maximizing helps to identify the presence of longer motifs, whereas averaging helps to identify cumulative effects of short motifs, and the contribution of each is determined automatically by learning.

Pooling

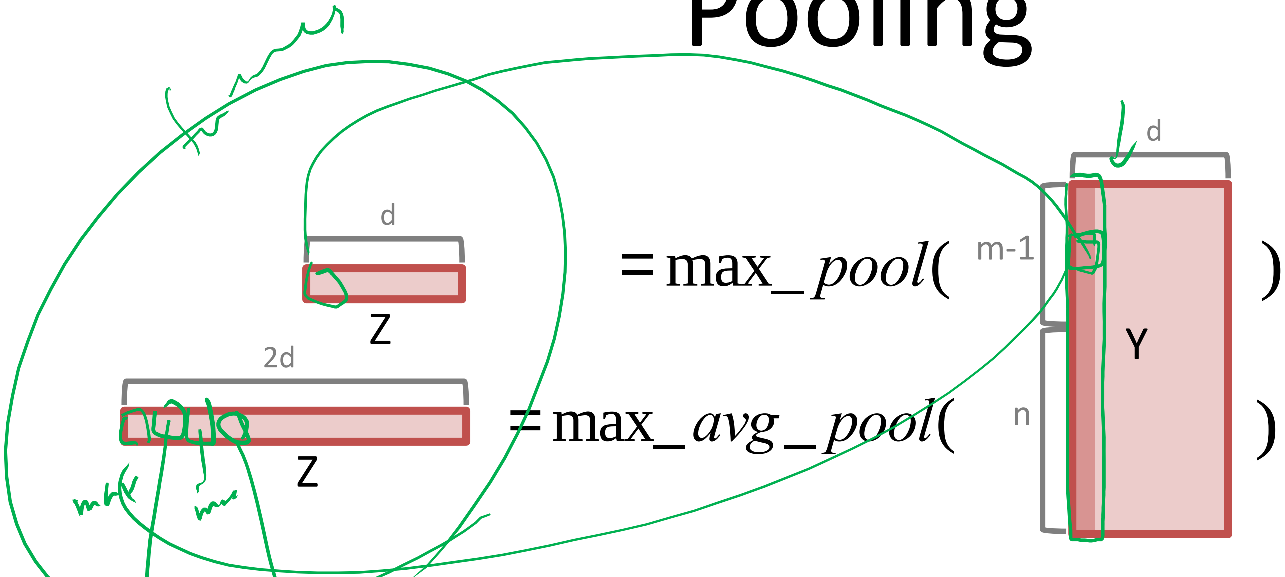






batch_size=5, motif_len=6, num_motifs=4, num_models=3

Pooling



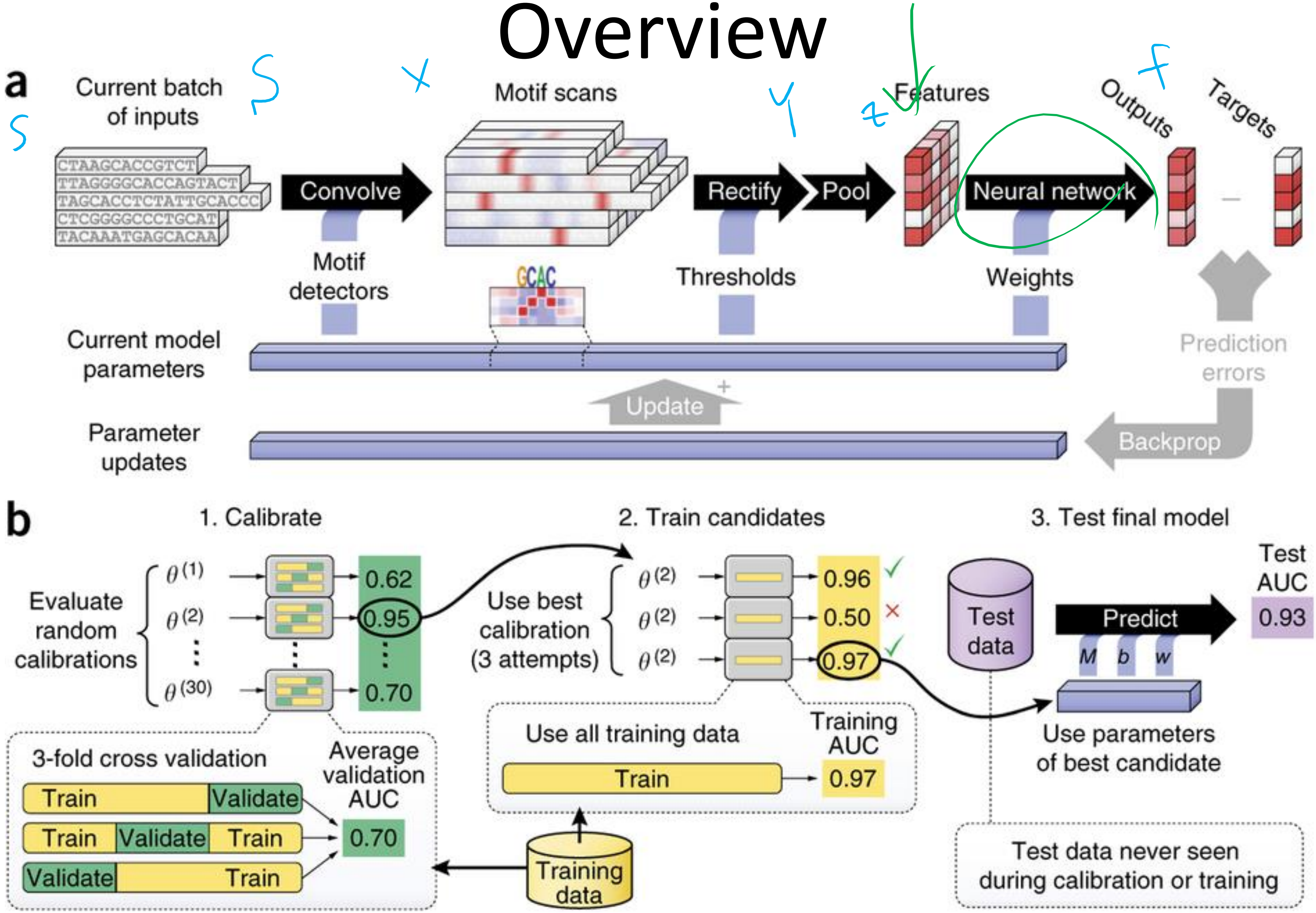
$$z_k = \max(Y_{1,k}, Y_{2,k}, \ldots, Y_{n,k}) \quad \text{(for DNA binding)}$$

$$z_{2k+0} = \max(Y_{1,k}, Y_{2,k}, \ldots, Y_{n,k}) \quad \text{(for RNA binding)}$$

$$z_{2k+1} = avg(Y_{1,k}, Y_{2,k}, \ldots, Y_{n,k})$$

- $1 \le k \le d$, where d is the number of motif detectors
- Y (input) : (n+m-1) x 4
- Z (output) : 1 x d (for DNA binding)
- Z (output) : 1 x 2d (for RNA binding, RBP model)

Overview



a
Current batch of inputs
S
S
X
Y
Z
f
CTAAGCACCGTCT
TTAGGGGCACCAGTACT
TAGCACCTCTATTGCACCC
CTCGGGGCCCTGCAT
TACAAATGAGCACAA
Convolve
Motif scans
Rectify
Pool
Features
Neural network
Outputs
Targets
Motif detectors
GCAC
Thresholds
Weights
Current model parameters
Prediction errors
Update
Parameter updates
Backprop
b
1. Calibrate
2. Train candidates
3. Test final model
Evaluate random calibrations
θ(1)
θ(2)
θ(30)
0.62
0.95
0.70
Use best calibration (3 attempts)
0.96
0.50
0.97
Test data
Predict
M
b
w
Test AUC
0.93
3-fold cross validation
Average validation AUC
Train
Validate
0.70
Use all training data
Training AUC
0.97
Use parameters of best candidate
Training data
Test data never seen during calibration or training

Neural network

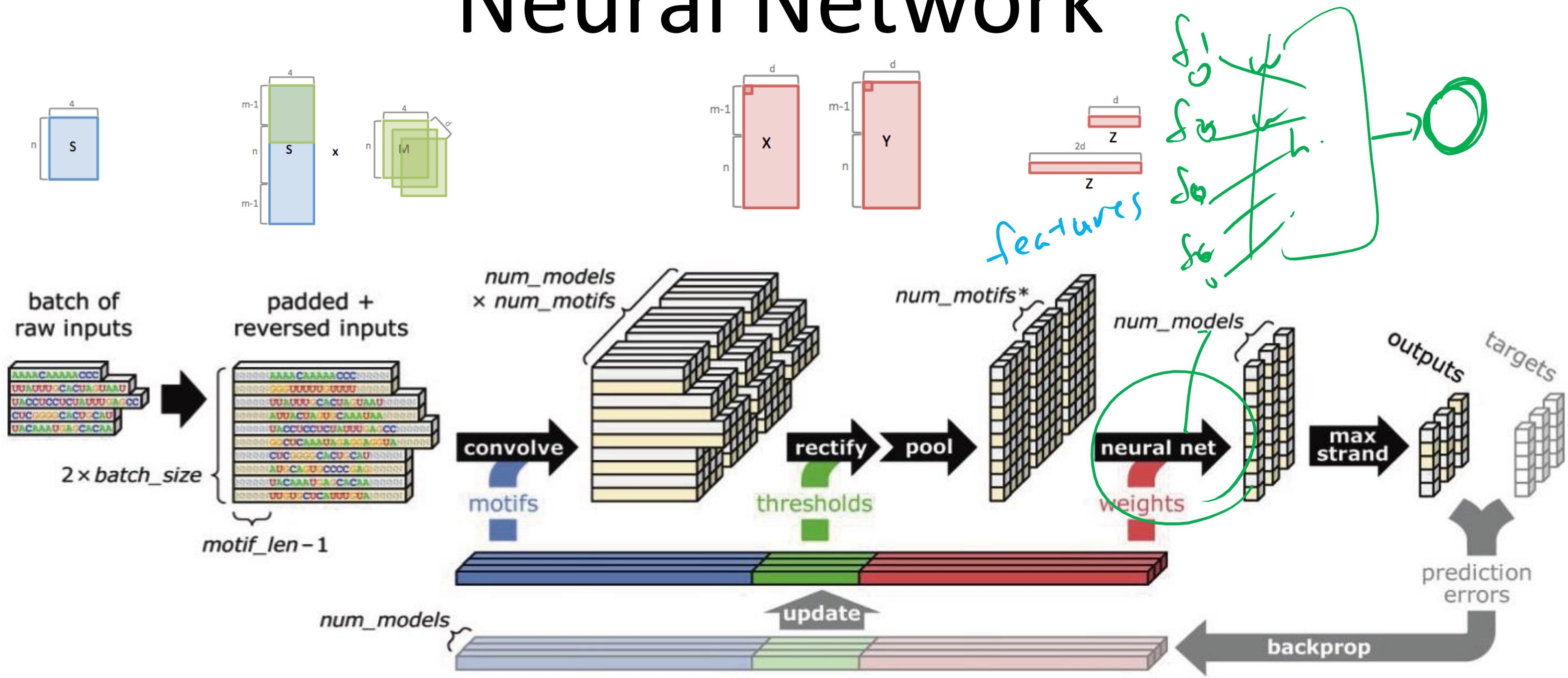
$$f(s) = net_w(pool(rect_b(conv_M(s))))$$

net_w

Output vector z from pooling plays a role of feature vector.
These values are fed into a nonlinear neural network with weights ***W***, which combines the responses to produce a score

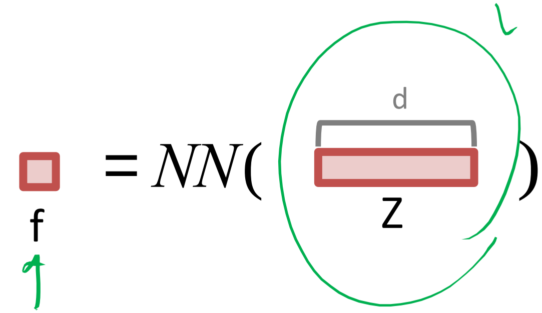
Training and scoring sequences

Neural Network



batch_size=5, motif_len=6, num_motifs=4, num_models=3

Neural Network



Without hidden layer

$$p = w_{d+1} + \sum_{k=1}^{d} w_k z_k$$

One hidden layer with 32 rectified-linear units(ReLU)

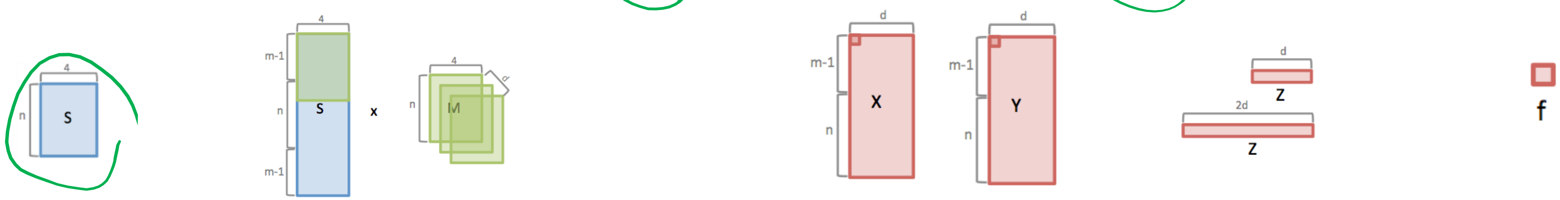
$$h_j = \max(0, w_{j,d+1} + \sum_{k=1}^{d} w_{j,k} z_k)$$

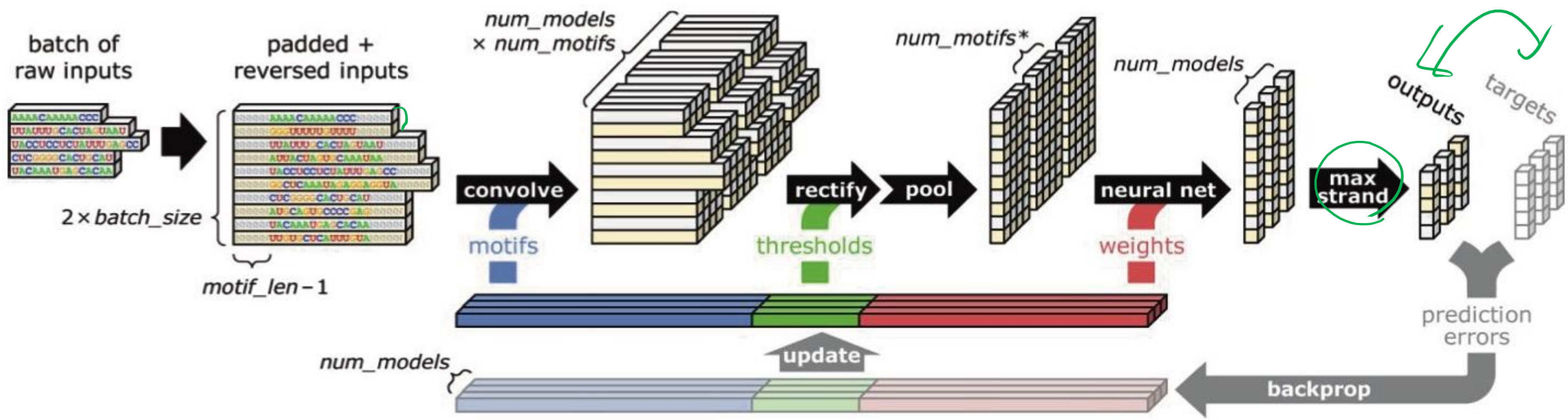
$$f = w_{33} + \sum_{j=1}^{32} w_j z_j$$

- $1 \le k \le d$, where d is the number of motif detectors
- Z (input) : 1 x d (for DNA binding)
- Z (input) : 1 x 2d (for RNA binding, RBP model)
- P or f : a scalar output score

Forward propagation

$$s \xrightarrow[M]{} X \xrightarrow[b]{} Y \longrightarrow z \xrightarrow[W]{} h \xrightarrow[w]{} \boxed{f}$$



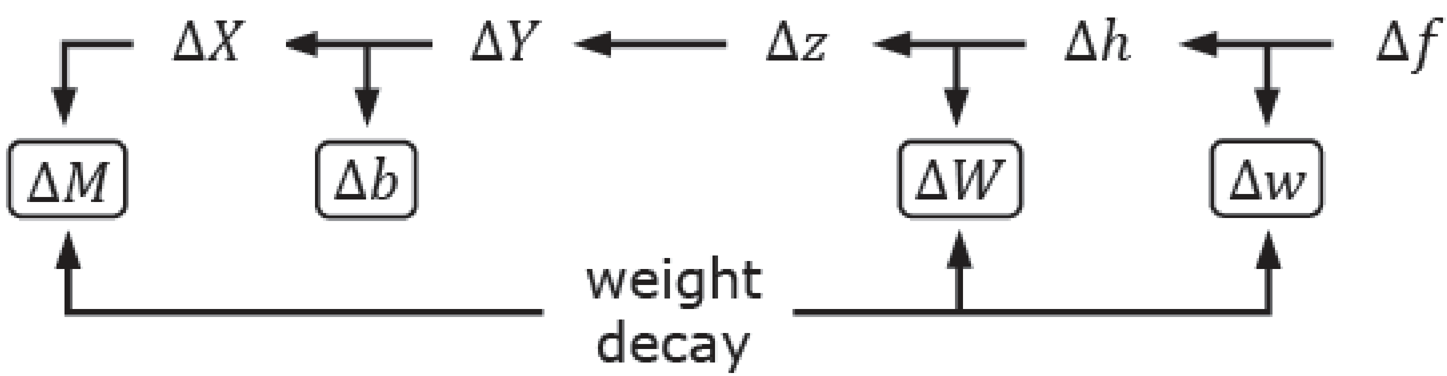


batch_size=5, motif_len=6, num_motifs=4, num_models=3

Back-propagation

Forward Propagation

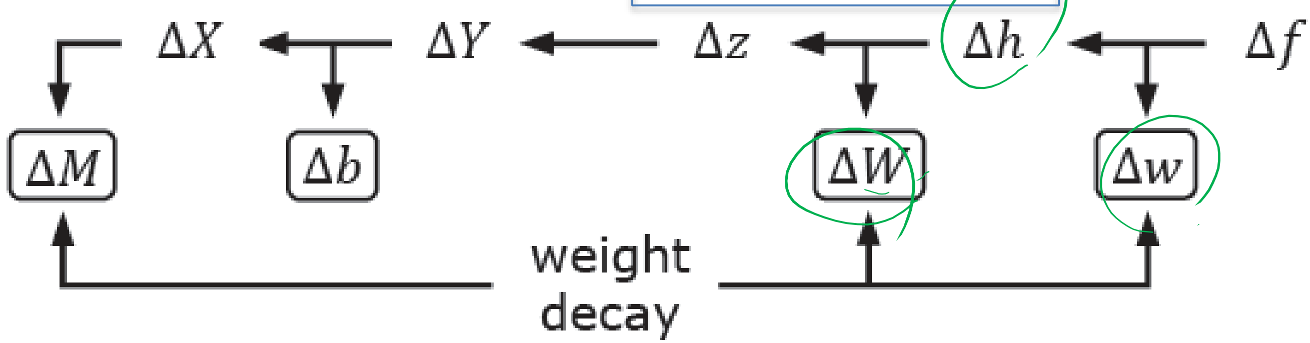
Back-propagation



Contributes to the gradients but independent of Df

Back-propagation : NN

$$\Delta z_k = \sum_{j=1}^{32} W_{j,k} \Delta h_j$$



$$\Delta W_{j,k} = z_k \Delta h_j \quad \text{for } k = 1 \dots d$$
$$\Delta W_{j,d+1} = \Delta h_j$$

Neural network with drop out

$$\Delta w_j = m_j h_j \Delta f \quad \text{for } j = 1 \dots 32$$
$$\Delta w_{33} = \Delta f$$

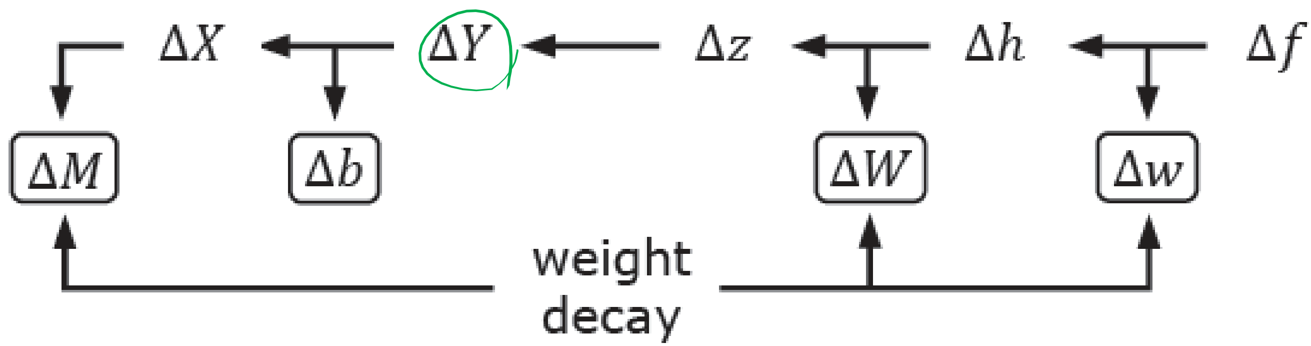
$m_j = \{0,1\}$, mask value for dropout

$$\Delta h_j = \begin{cases} m_j w_j \Delta f & \text{if } h_j > 0 \\ 0 & \text{otherwise} \end{cases}$$

Back-propagation : Pooling

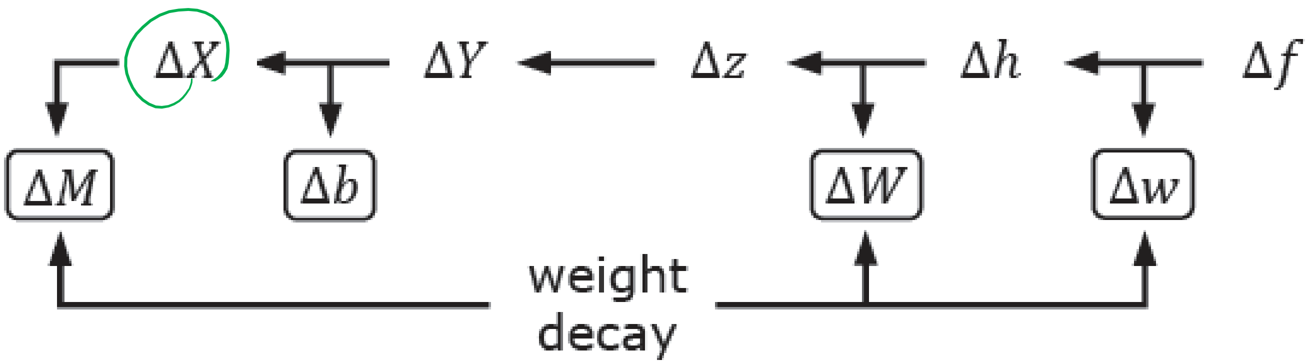
$$\Delta Y_{i,k} = \begin{cases} \Delta z_k & \text{if } i = \operatorname{argmax}(Y_{1,k}, \dots, Y_{n,k}) \\ 0 & \text{otherwise.} \end{cases}$$

$$\Delta Y_{i,k} = \frac{\Delta z_{2k+1}}{n+m-1} + \begin{cases} \Delta z_{2k+0} & \text{if } i = \operatorname{argmax}(Y_{1,k}, \dots, Y_{n,k}) \\ 0 & \text{otherwise.} \end{cases}$$



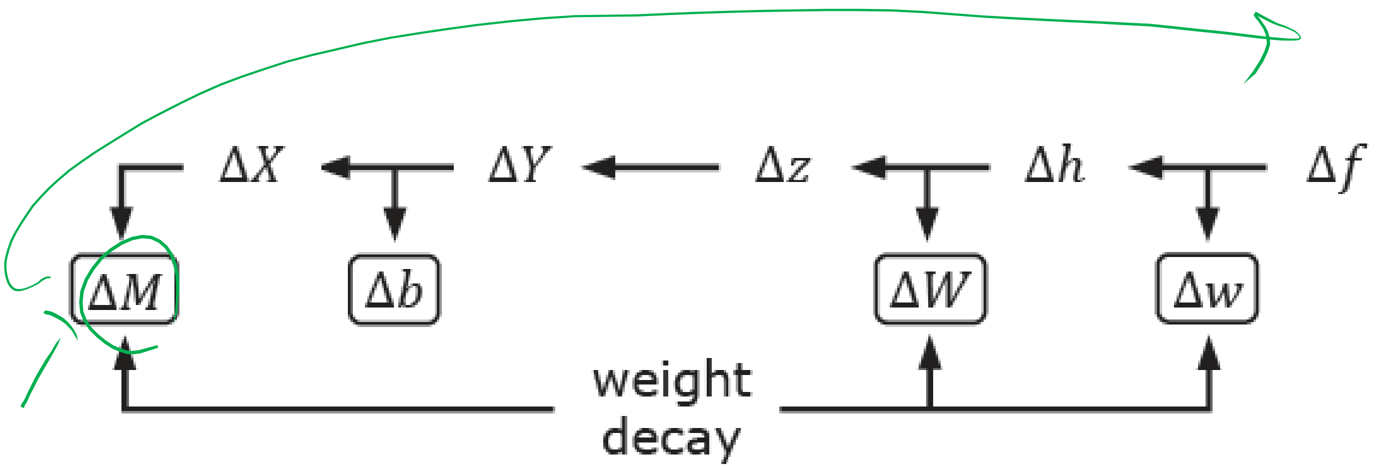
Back-propagation : Rectification

$$\Delta X_{i,k} = \begin{cases} \Delta Y_{i,k} & \text{if } Y_{i,k} > 0 \\ 0 & \text{otherwise} \end{cases}$$



$$\Delta b_k = \sum_{i=1}^{n+m-1} \Delta Y_{i,k}$$

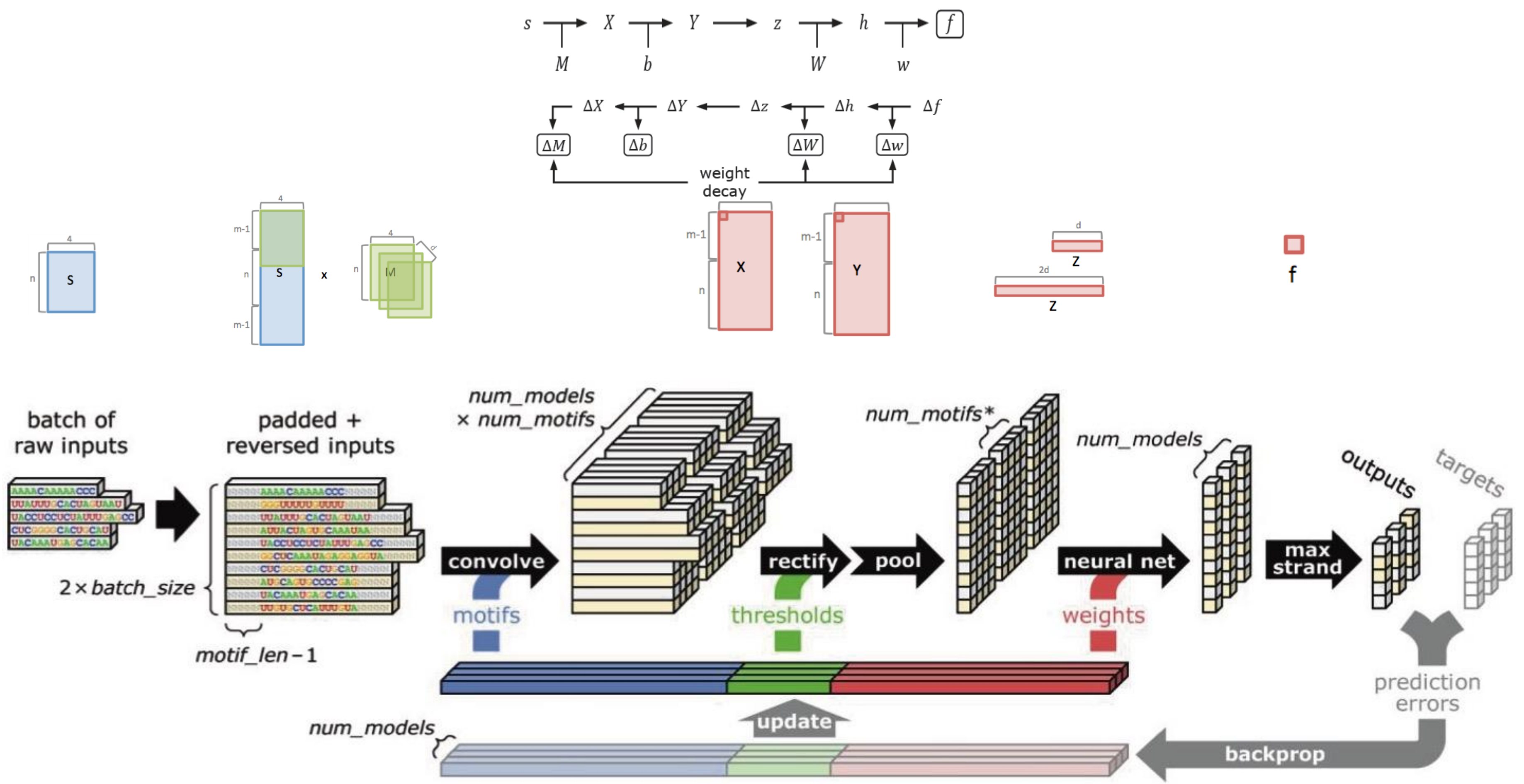
Back-propagation : Convolution



$$\Delta M_{k,j,l} = \sum_{i=1}^{n+m-1} S_{i+j,l}\,\Delta X_{i,k}$$

Updates

Forward/Backward Propagation



batch_size=5, motif_len=6, num_motifs=4, num_models=3

Summary of DeepBind

- Advantages
  - Can be applied to both PBM and ChIP-seq
  - Can learn from millions of sequences
  - Generalizes well across technologies, without need for correcting for technology-specific biases
  - Tolerates noise
  - Trains predictive models fully automatically
- Disadvantages
  - Model can overfit data (regularizers such as dropout, weight decay, and early stopping are used)
  - Parameters do not have any physical meaning

Q&A