STANFORD
COMPUTER SCIENCE

STANFORD SCHOOL of MEDICINE | Genetics

How to train your DragoNN

A primer on deep learning for regulatory genomics

Surag Nair
(CS PhD student)

Anshul Kundaje

Genetics, Computer Science

Stanford University

What will we cover today

- Supervised machine learning for decoding regulatory DNA
- Why are convolutional neural networks a good model family for regulatory DNA
- How do we train neural networks
- How do we evaluate neural networks
- Commonly used neural network architectures

TGCCAAGCAGCAAAGTTTTGCTGCTGTTTATTTTTGTAGCTCTTACTATATTCT
ACTTTTACCATTGAAAATATTGAGGAAGTTATTTATATTTCTATTTTTTATATAT
TATATATTTTATGTATTTTAATATTACTATTACACATAATTATTTTTTATATATATGA
AGTACCAATGACTTCCTTTTCCAAAGCAATAATGAAATTTCACAGTATGAAA
ATGGAAGAAATCAATAAAATTATACGTGACCTGTGGCGAAGTACCTATCGTG
GACAAGGTGAGTACCATGGTGTATCACAAATGCTCTTTCCAAAGCCCTCTCC
GCAGCTCTTCCCCTTATGACCTCTCATCATGCCAGCATTACCTCCCTGGACCC
CTTTCTAAGCATGTCTTTGAGATTTTCTAAGAATTCTTATCTTGGCAACATCTT
GTAGCAAGAAAATGTAAAGTTTTCTGTTCCAGAGCCTAACAGGACTTACATA
TTTGACTGCAGTAGGCATTATATTTAGCTGATGACATAATAGGTTCTGTCATA
GTGTAGATAGGGATAAGCCAAAATGCAATAAGAAAAACCATCCAGAGGAA
ACTCTTTTTTTTTTCTTTTTCTTTTTTTTTTTTTCCAGATGGAGTCTCGCACTTC
TCTGTCACCCGGGCTGGAGCGCAGTGGTGCAATCTTGGCTCACTGCAACCT
CCACCTCCTGGGTTCAGGTGATTCTCCCACCTCAGCCTCCCGAGTAGTAGCT
GGAATTACAGGTGCGCGCTCCCACACCTGGCTAATTTTTTGTATTCTTAGTA
GAGATGGGGTTTCACCATGTTGGCCAGGCTGGTCTCAAACTCCTGCCCTCA
GGTGATCTGCCCACCTTGGCCTCCCAGTGTTGGGTTTACAGGCGTGAGCCA
CCGCGCCTGGCCTGGAGGAAACTCTTAACAGGGAAACTAAGAAAGAGTTG
AGGCTGAGGAACTGGGGCATCTGGGTTGCTTCTGGCCAGACCACCAGGCT
CTTGAATCCTCCCAGCCAGAGAAAGAGTTTCCACACCAGCCATTGTTTTCCT
CTGGTAATGTCAGCCTCATCTGTTGTTCCTAGGCTTACTTGATATGTTTGTAA
ATGACAAAAGGCTACAGAGCATAGGTTCCTCTAAAATATTCTTCTTCCTGTGT
CAGATATTGAATACATAGAAATACGGTCTGATGCCGATGAAAATGTATCAGCT
TCTGATAAAAGGCGGAATTATAACTACCGAGTGGTGATGCTGAAGGGAGAC
ACAGCCTTGGATATGCGAGGACGATGCAGTGCTGGACAAAAGGCAGGTAT
CTCAAAAGCCTGGGGAGCCAACTCACCCAAGTAACTGAAAGAGAGAAACA
AACATCAGTGCAGTGGAAGCACCCAAGGCTACACCTGAATGGTGGGAAGC
TCTTTGCTGCTATATAAAATGAATCAGGCTCAGCTACTATTATT .....

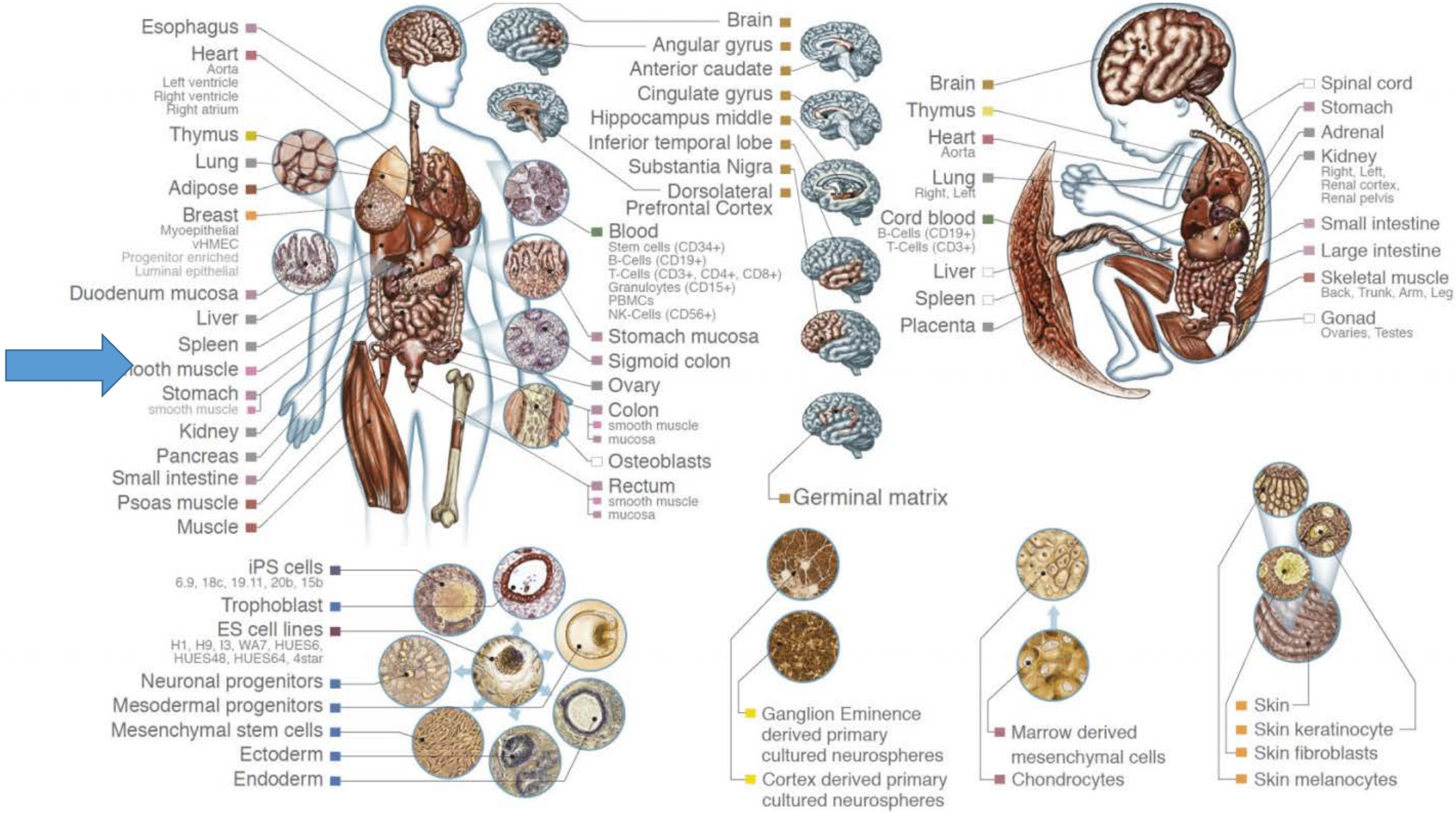
Decoding genome function

Function?

Human genome = DNA sequence with ~ 3 billion letters

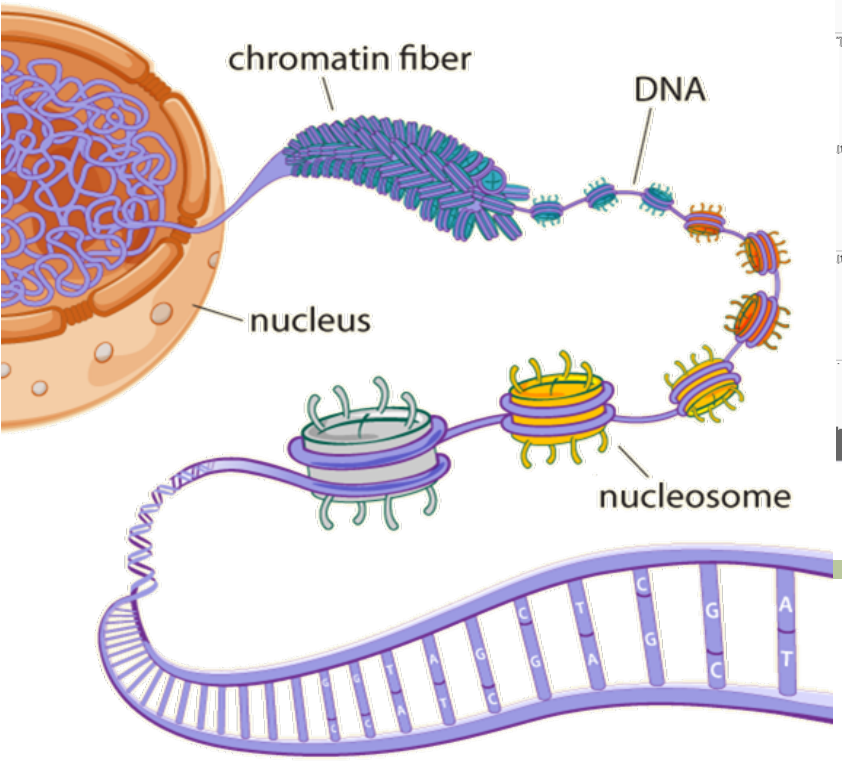
One genome ⇔ many cell types

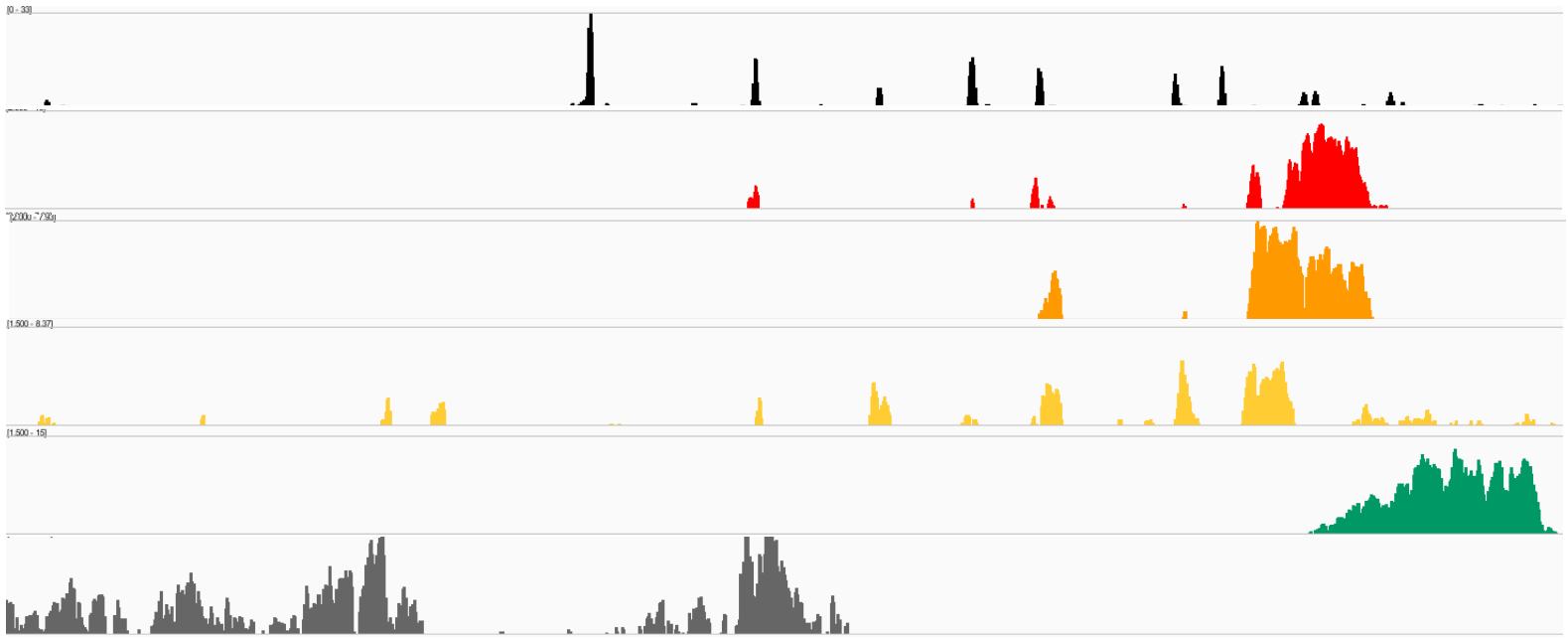
ACCAGTTACGACGG
TCAGGGTACTGATA
CCCCAAACCGTTGA
CCGCATTTACAGAC
GGGGTTTGGGTTTT
GCCCCACACAGGTA
CGTTAGCTACTGGT
TTAGCAATTTACCG
TTACAACGTTTACA
GGGTTACGGTTGGG
ATTTGAAAAAAAGT
TTGAGTTGGTTTTT
TCACGGTAGAACGT
ACCTTACAAA……

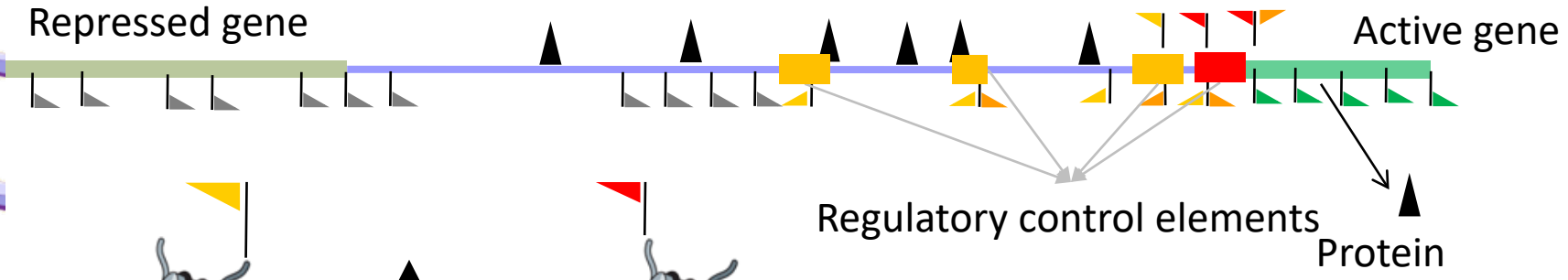


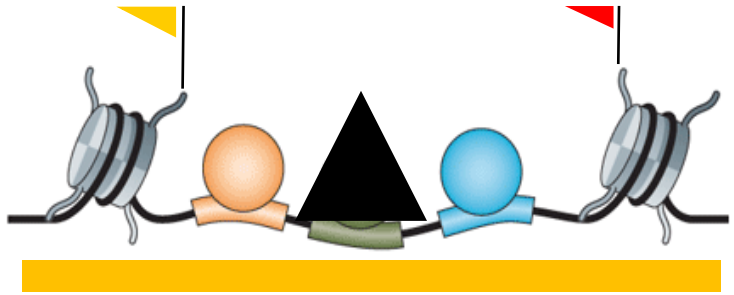
http://www.roadmapepigenomics.org/

Molecular profiling of functional elements in the genome



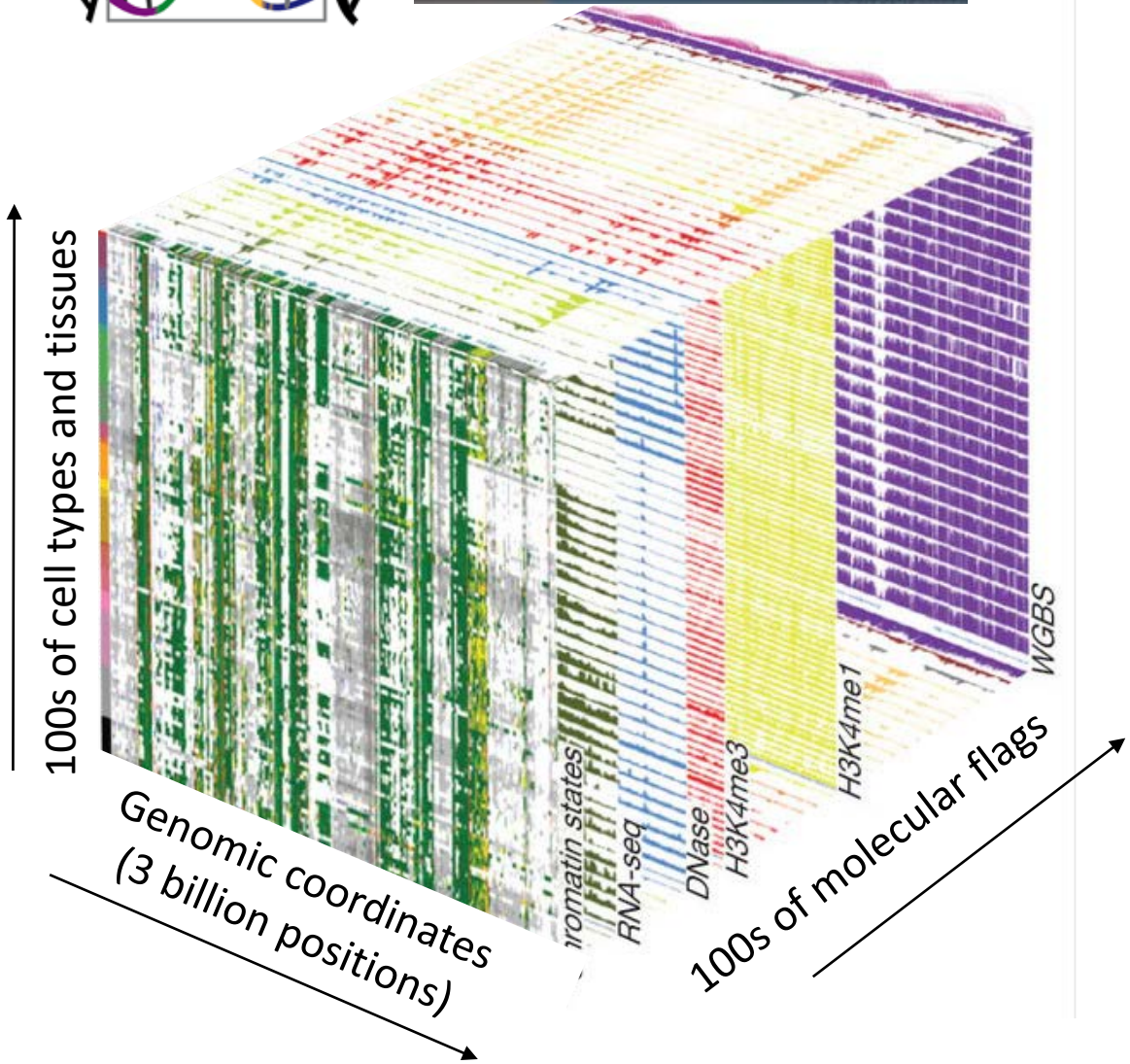




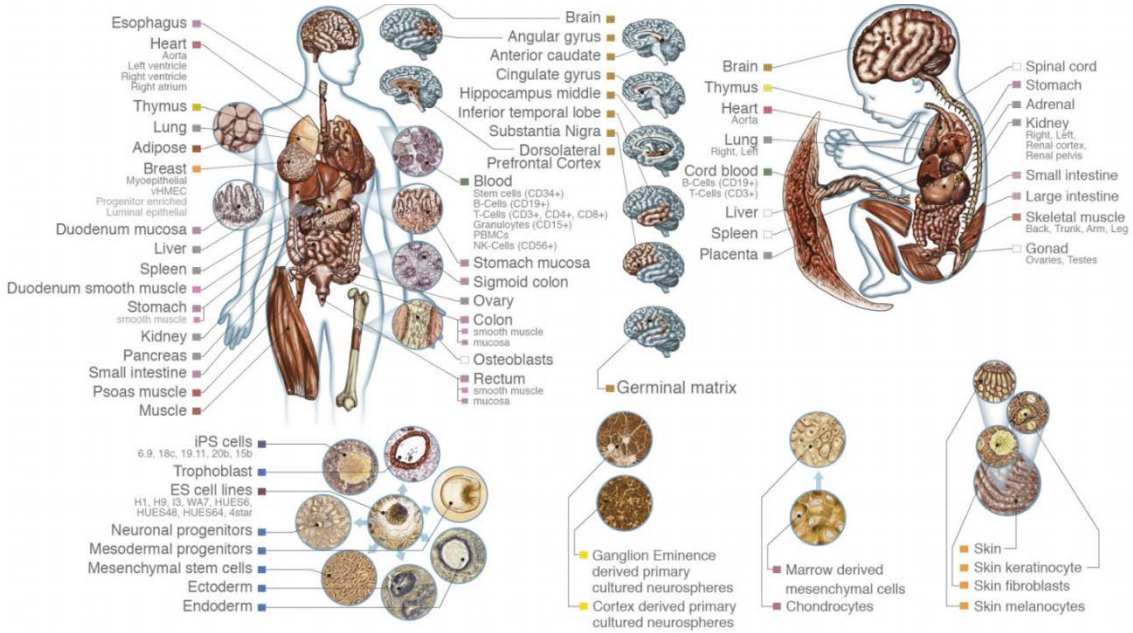


https://www.broadinstitute.org/news/1504

Massive genomic data resources



Dunham, Kundaje et al. 2012 Nature
Kundaje et al. 2015 Nature



100s of Cell-Types/Tissues

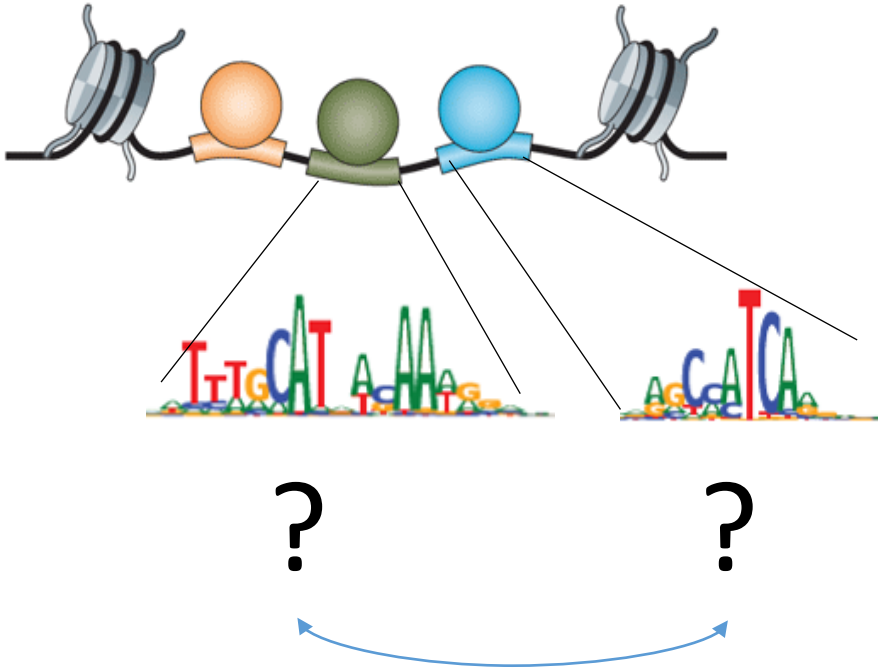
Machine learning,
Probabilistic models,
Deep learning

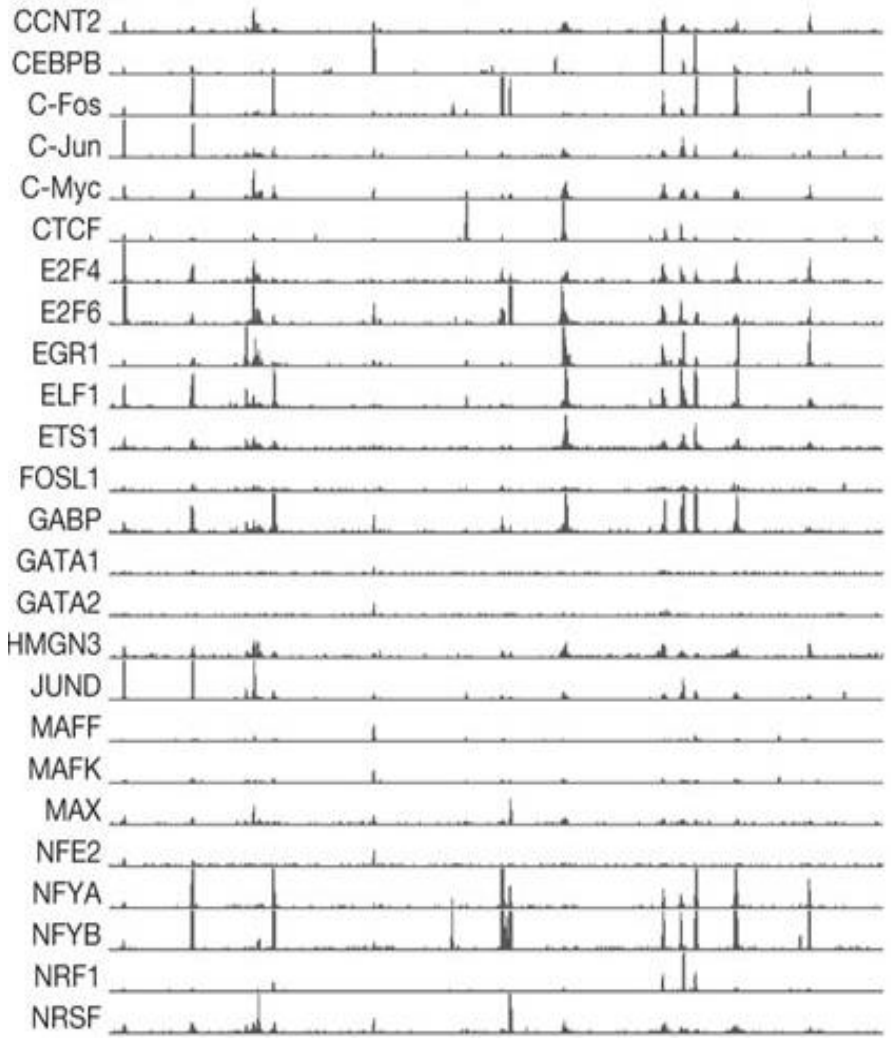
Functional annotation of the human genome

Predict molecular effects of mutations and genetic variants

Deciphering functional DNA words and their syntax in regulatory DNA

Syntax: Rules of arrangement, preferred spacing, orientation, interactions between works



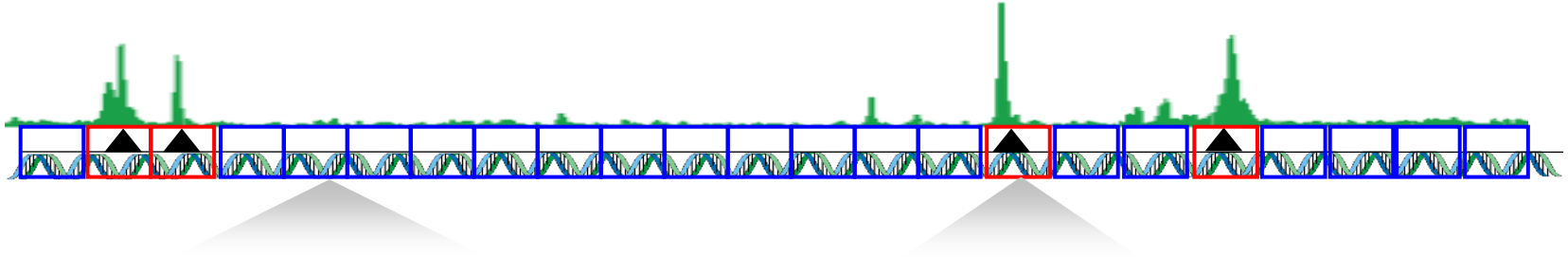


Protein-DNA binding maps

Adapted from Thurman et al 2012

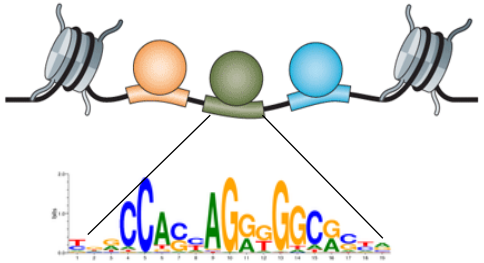
Predictive model of regulatory DNA

Transcription factor ChIP-seq data OR chromatin accessibility (DNase-seq / ATAC-seq data)



…GACTTGAAACGGCATTG…
Inactive (0) (0.3)

…GACAGATAATGCATTGA…
Active (+1) (20.2)



…GACAGATAATGCATTGA…

…ACTGTCATGGATATTCT…

…GATATTCTACTGTAAG…

DNA sequences (S_i)

…CAACCTTGAACGGCATTG…

…GACTTGAAACGGCATTG…

…CAGTATGCATACGTGAA…

Classification or Regression model $F(S_i)$

Arvey et al. 2012
Ghandi et al. 2014
Setty et al. 2015

Class = +1 (20.2)

Class = +1 (10.6)

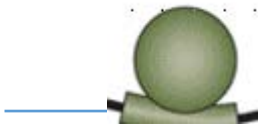
Class = +1 (15.8)

Measured Labels (Y_i)

Class = 0 (0.3)

Class = 0 (1.2)

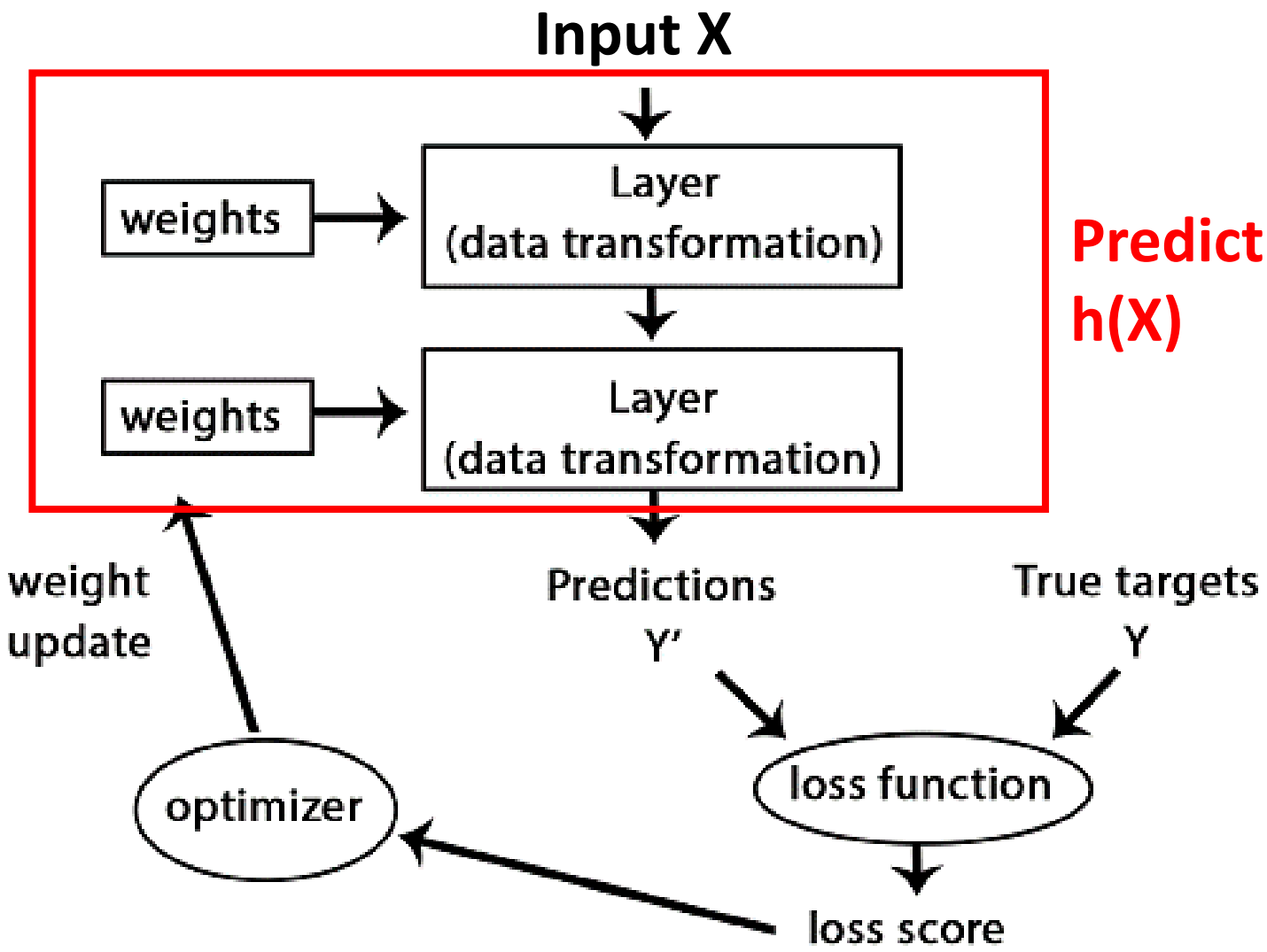
Class = 0 (3.5)



Bound

Unbound

Supervised machine learning



https://www.manning.com/books/deep-learning-with-python

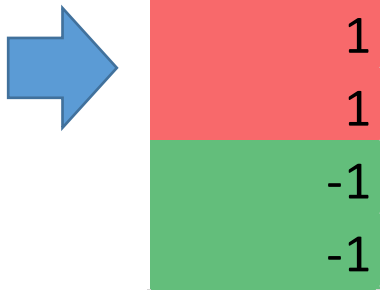
How to represent DNA sequence?

Bag of *k*-mers (all possible subsequences of length *k*)

K-mer features ($x_1, x_2, x_3 \ \ldots x_n$)

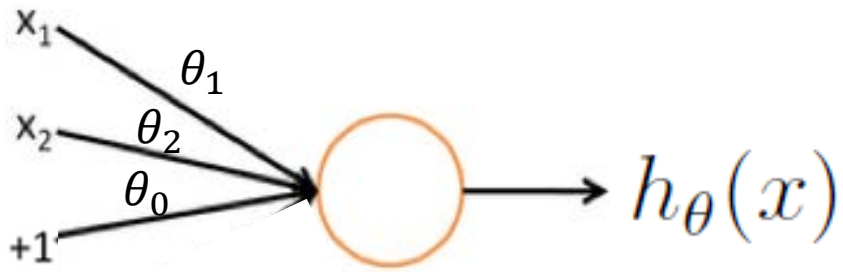
Input Sequences ($S^{(1)} \ldots S^{(m)}$)

	AGA	GAT	ATC	TCG	CGA	GAG	AGT	GTG
Sequence1	1	4	5	2	0	0	0	0
Sequence2	4	1	1	0	0	2	8	8
Sequence3	6	3	3	2	2	4	10	10
Sequence4	5	5	6	2	0	2	8	8
Sequence5	10	4	4	2	2	6	18	18



Output labels ($Y_1 \ldots Y_m$)

Linear artificial neuron (linear regression)

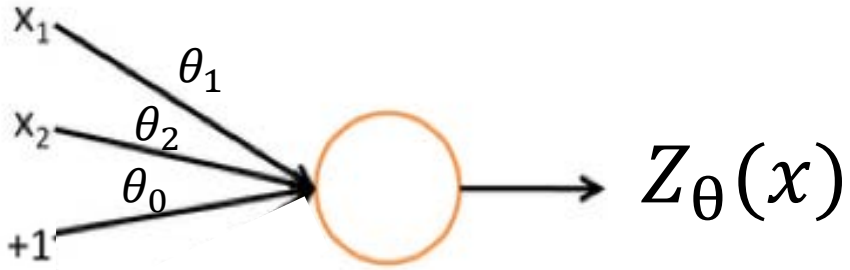


parameters

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 = \sum_{i=0}^{n} \theta_i x_i = \theta^T x,$$

Training the neuron means learning the parameters to minimize some loss

Non-linear sigmoid neuron (logistic regression)



$$Z_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

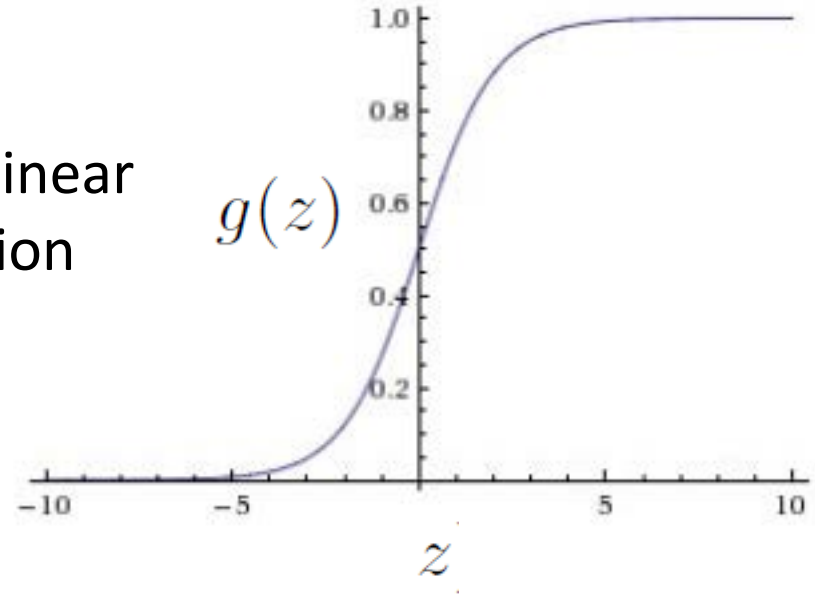
$$g(z) = \frac{1}{1 + e^{-z}}$$

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}},$$

Logistic / Sigmoid

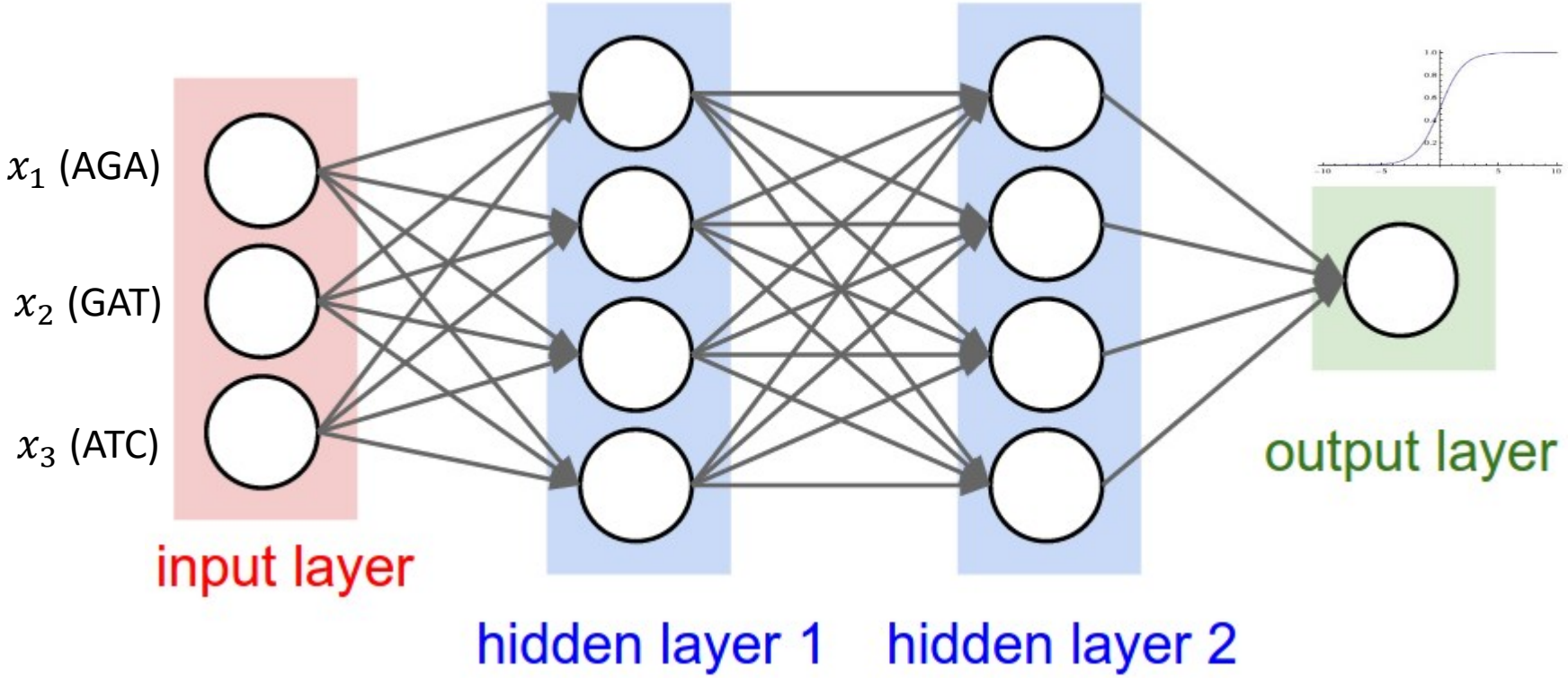
Useful for predicting probabilities

Non-linear function



Training the model means learning the parameters to minimize some logistic loss

Dense (fully-connected) deep neural network



How many parameters does this DNN have?

3x4 + 4x4 + 4x1 + 9 = 12 + 16 + 4 + 9 = 41

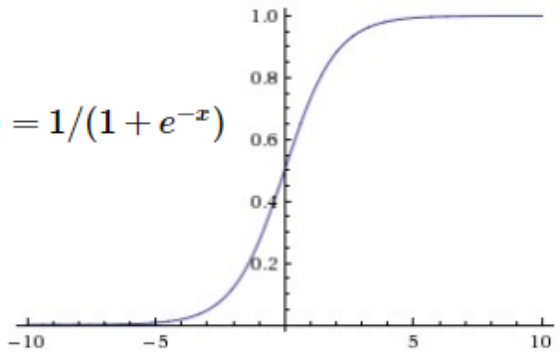
Architecture and hyperparameters of DNN

- How many layers?
- How many neurons per layer?
- What types of activations to use?

Search over architectures and identify optimal architecture by evaluating performance on validation set

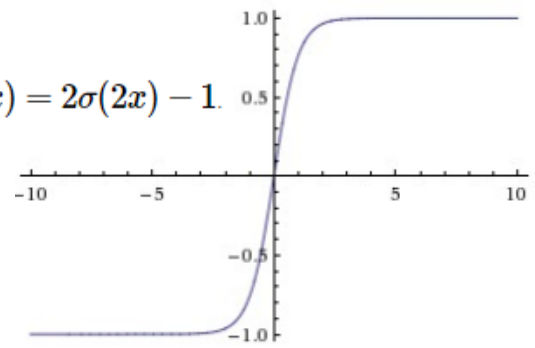
Types of non-linear neurons (activations)

$\sigma(x) = 1/(1+e^{-x})$



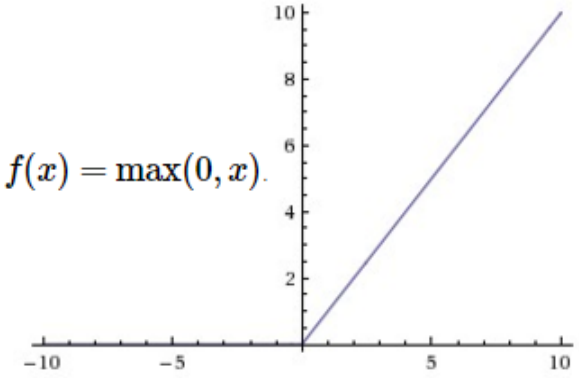
Sigmoid/Logistic

$\tanh(x) = 2\sigma(2x) - 1$



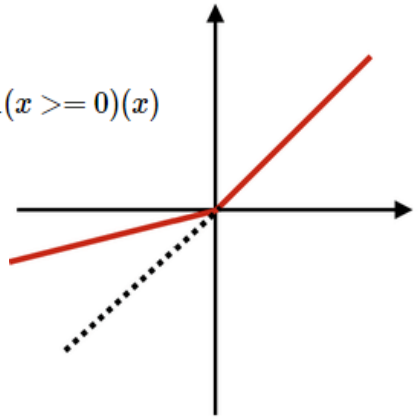
tanh

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



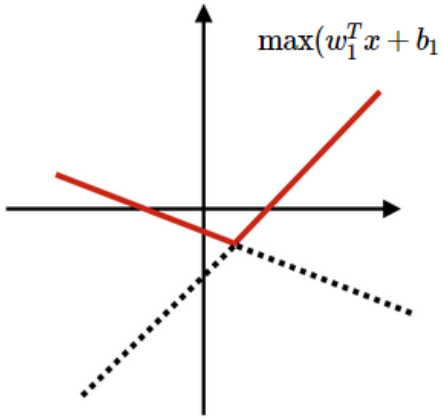
ReLU (rectified linear unit)

$f(x) = 1(x < 0)(\alpha x) + 1(x >= 0)(x)$



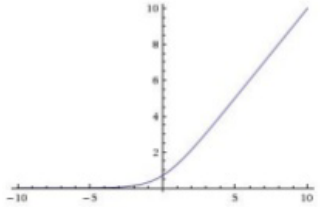
PReLU

$\max(w_1^T x + b_1, w_2^T x + b_2)$

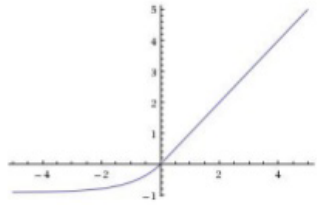


Maxout (n=2)

Softplus and Exponential Linear Unit (ELU)



$softplus(x) = log(1 + e^x)$



$$f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha\,(\exp(x) - 1) & \text{if } x \leq 0 \end{cases}$$

How do we train these models i.e. learn the weights of the neurons?

Loss (cost) function to minimize

For a specific configuration of parameters (weights) how far different is your prediction from true label

Square loss for regression

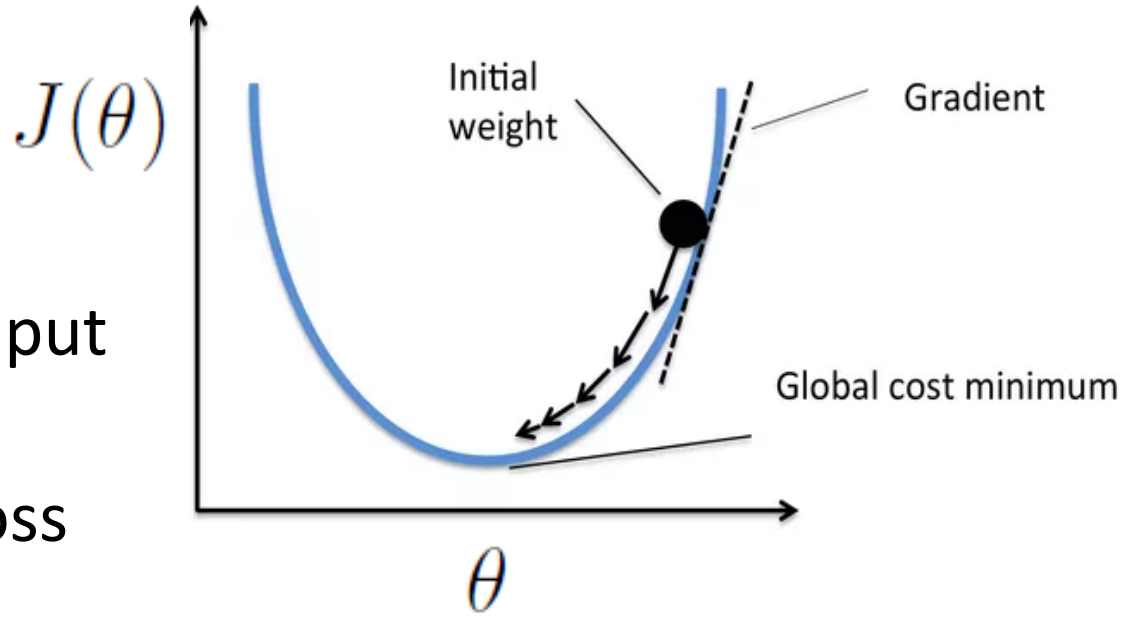
$$J(\theta) = \frac{1}{2}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2.$$

Logistic loss / binary cross entropy for classification

$$J(\theta) = -\left(\sum_{i=1}^{m} y^{(i)} \log h(x^{(i)}) + (1 - y^{(i)}) \log(1 - h(x^{(i)}))\right)$$

Iterative algorithm to minimize loss (stochastic gradient descent)

1. Define loss $J(\theta)$
2. Randomly initialize all weights θ_j
3. Iterate multiple times (epochs) over all input sequences with labels $(x^{(i)}, y^{(i)})$
   - Update each θ_j slightly to decrease loss



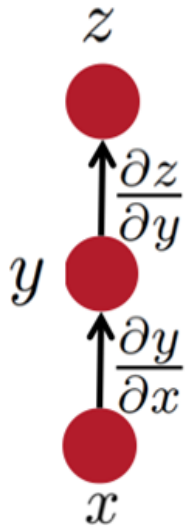
$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J_{(x^i, y^i)}(\theta_j)$$

$\frac{\partial}{\partial \theta_j} J(\theta)$. is called the gradient

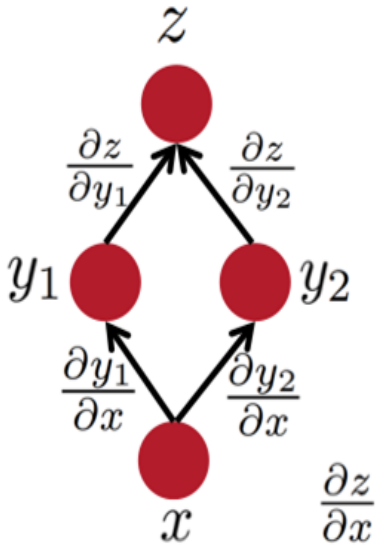
α is called the **learning rate**

Gradients can be computed efficiently using a message passing algorithm called backpropagation

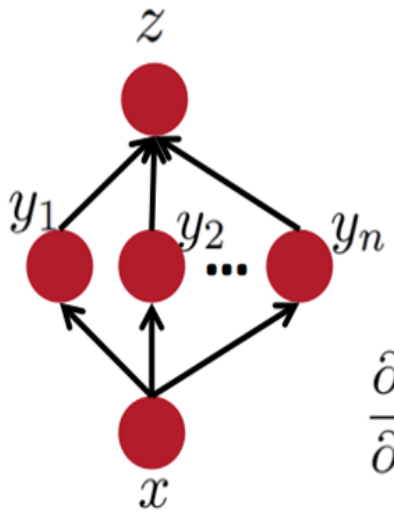
Computing gradients efficiently using backpropagation chain rule



$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}$$



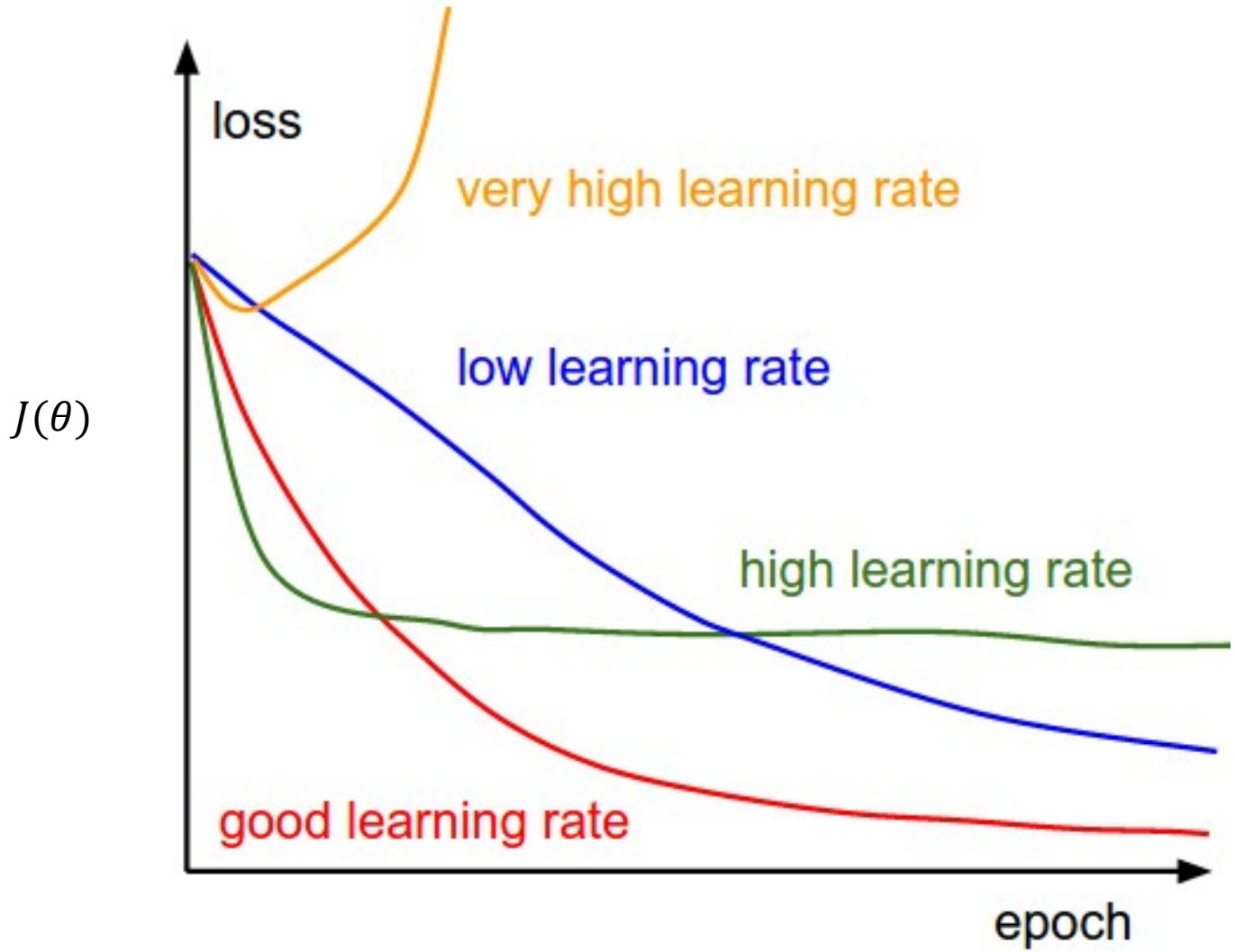
$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y_1}\frac{\partial y_1}{\partial x} + \frac{\partial z}{\partial y_2}\frac{\partial y_2}{\partial x}$$



$$\frac{\partial z}{\partial x} = \sum_{i=1}^{n} \frac{\partial z}{\partial y_i}\frac{\partial y_i}{\partial x}$$

http://neuralnetworksanddeeplearning.com/chap2.html

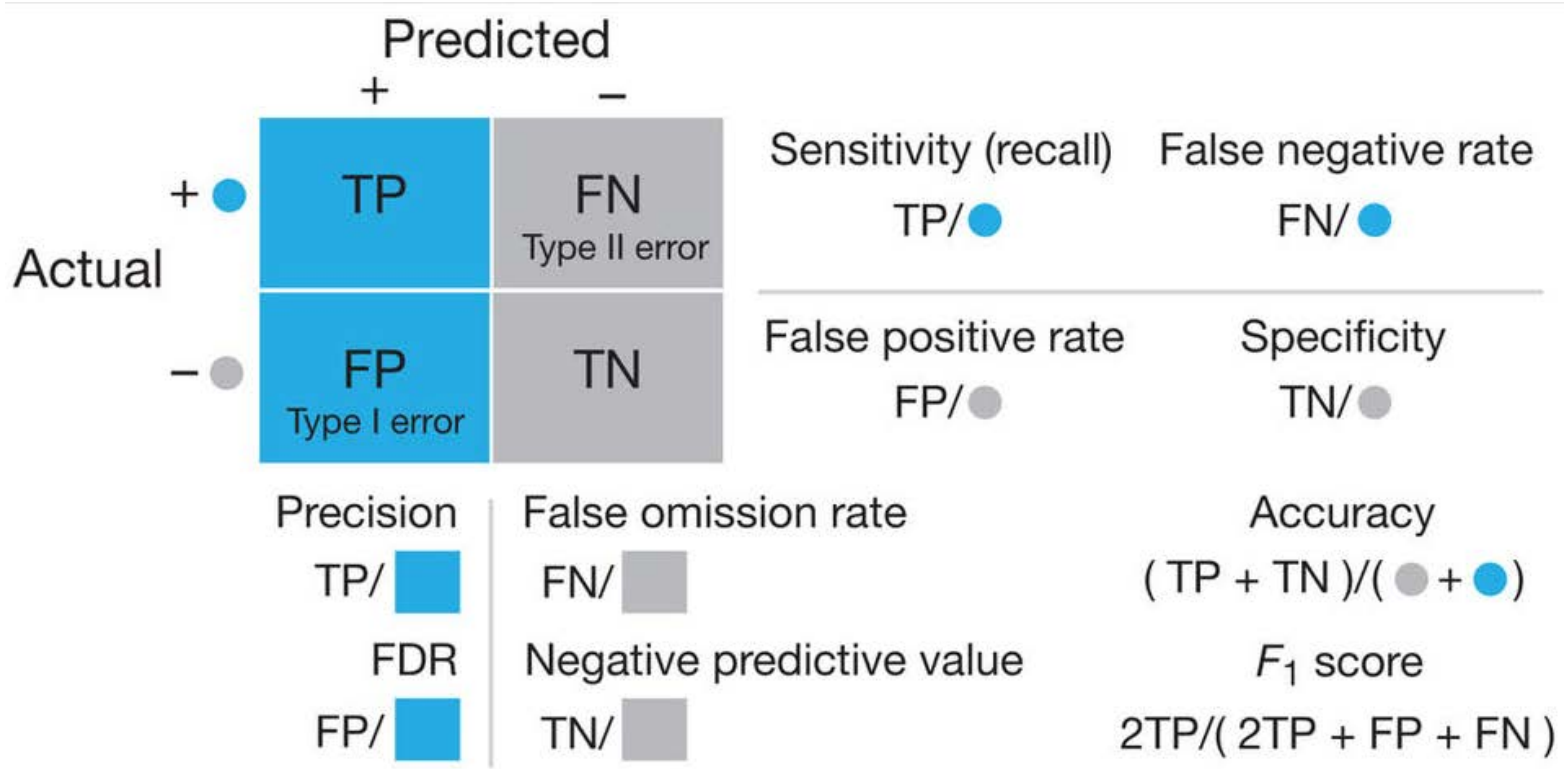
Visualizing loss learning curve is very useful to understand learning behavior of neural network



How do measure predictive performance?

Measures of classification performance

Select a threshold on the output probability (e.g. 0.5) to predict positive or negative

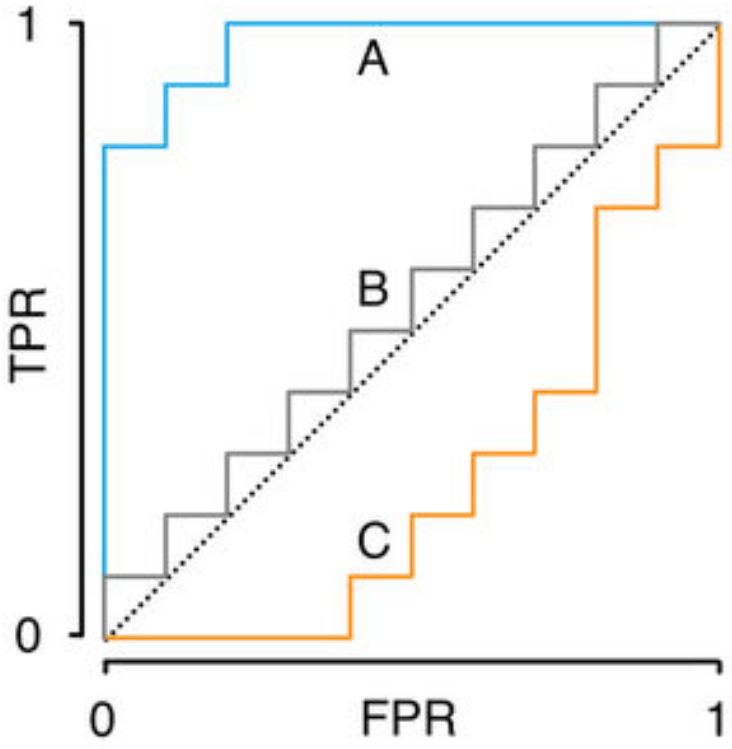


ROC curve and precision-recall curve

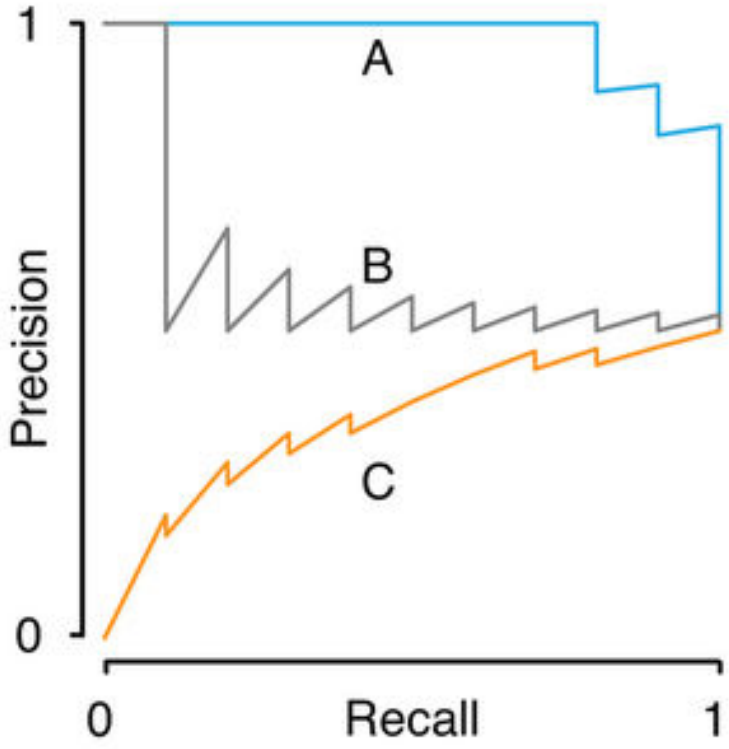
You can tune the output probability threshold to get different trade offs between sensitivity/specificity or precision/recall i.e. a curve

	Predicted +	Predicted −		
Actual +	TP	FN (Type II error)	Sensitivity (recall) TP/●	False negative rate FN/●
Actual −	FP (Type I error)	TN	False positive rate FP/●	Specificity TN/●
Precision TP/■	False omission rate FN/■	Accuracy (TP + TN)/(● + ●)		
FDR FP/■	Negative predictive value TN/■	F_1 score 2TP/(2TP + FP + FN)		

ROC curve

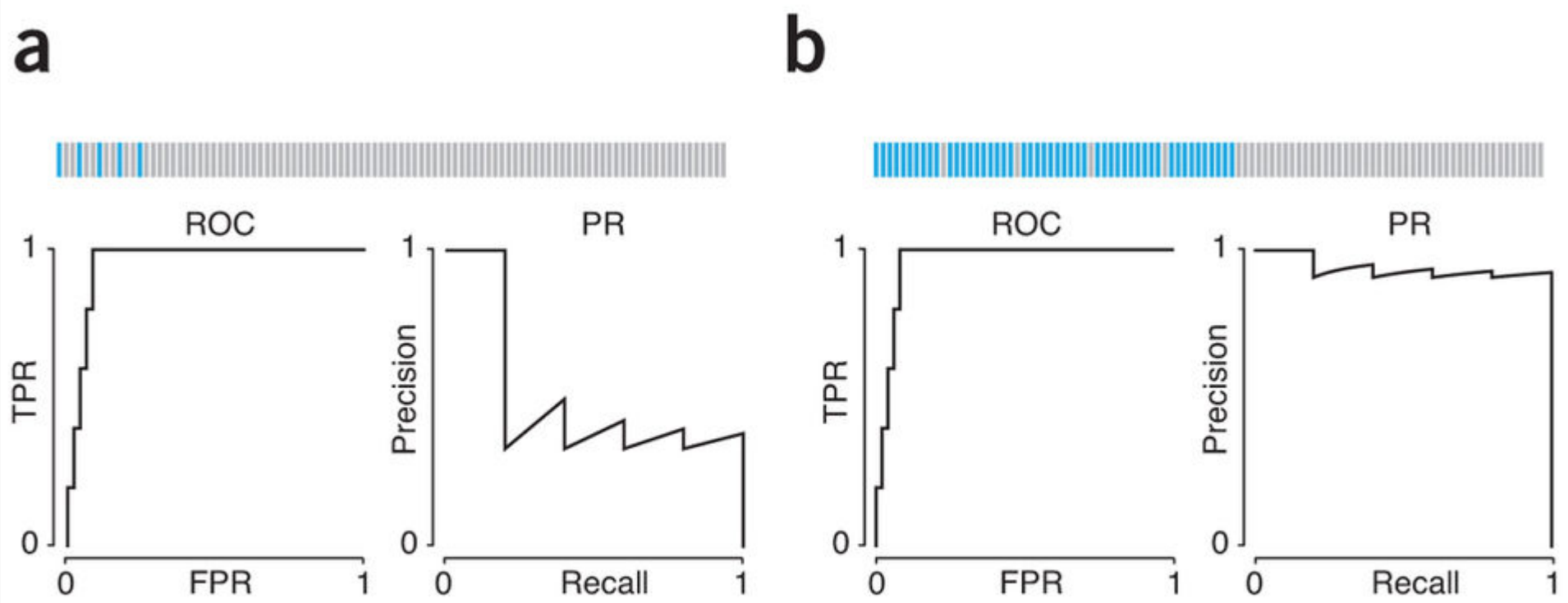


PR curve



Area under the curve (AUC) is a useful measure of predictive performance across all thresholds

auROC can be misleading for imbalanced classes (many more negatives than positives)



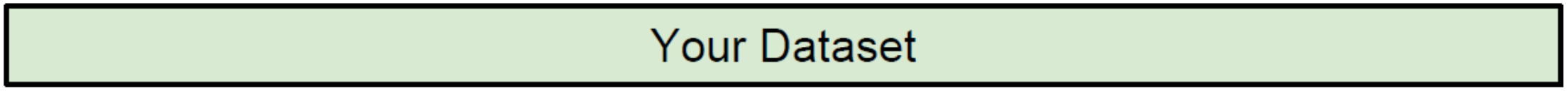
(**a**,**b**) ROC and PR curves for two data sets with very different class balances: (**a**) 5% positive and (**b**) 50% positive observations. For each panel, observations are shown as vertical lines (top), of which 5% or 50% are positive (blue).

Very common evaluation flaw in genomics papers

Evaluate performance of model on held-out test data

Idea #1: Choose hyperparameters that work best on the data

BAD: K = 1 always works perfectly on training data



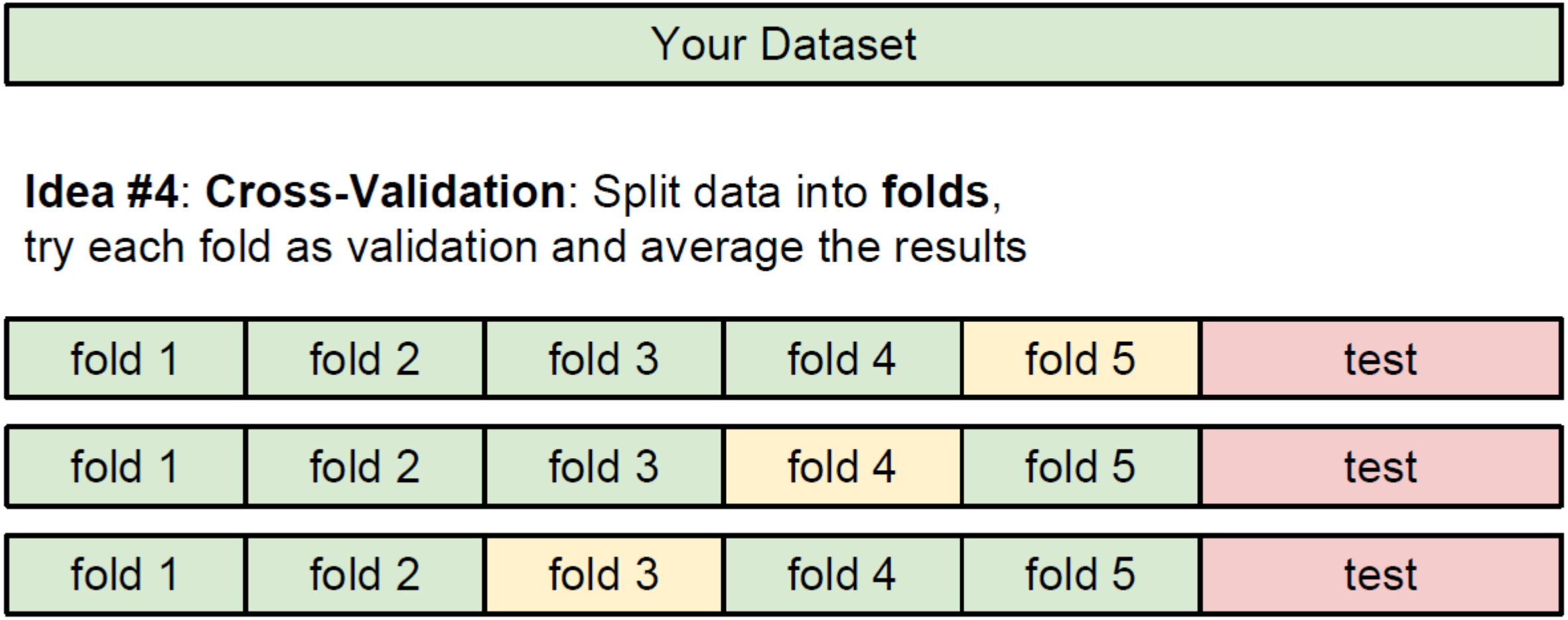
Idea #2: Split data into **train** and **test**, choose hyperparameters that work best on test data

BAD: No idea how algorithm will perform on new data

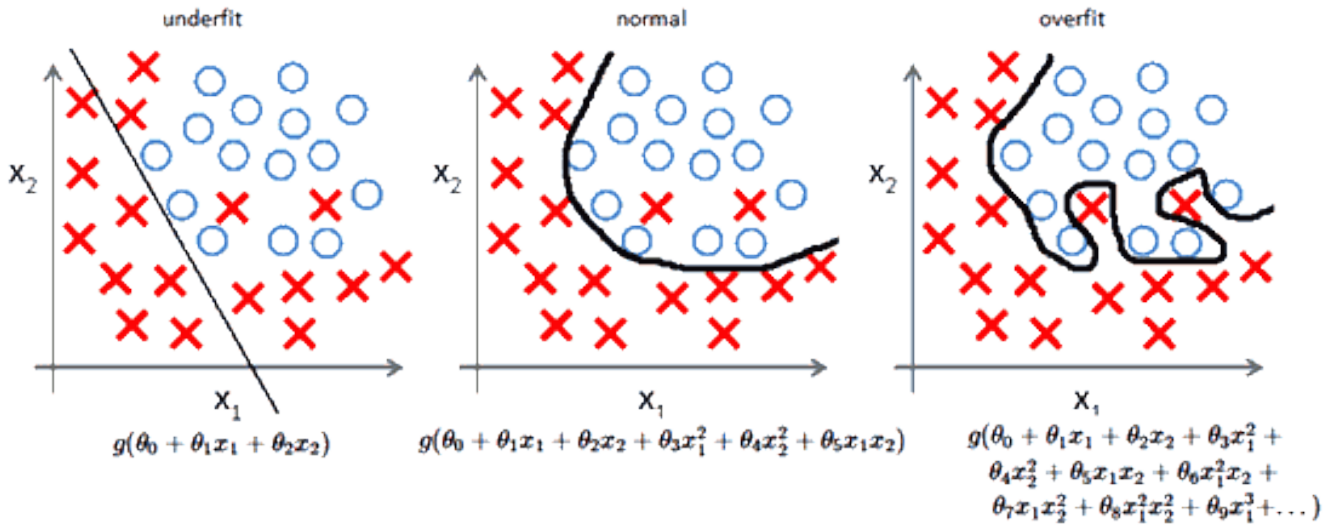
Idea #3: Split data into **train**, **val**, and **test**; choose hyperparameters on val and evaluate on test

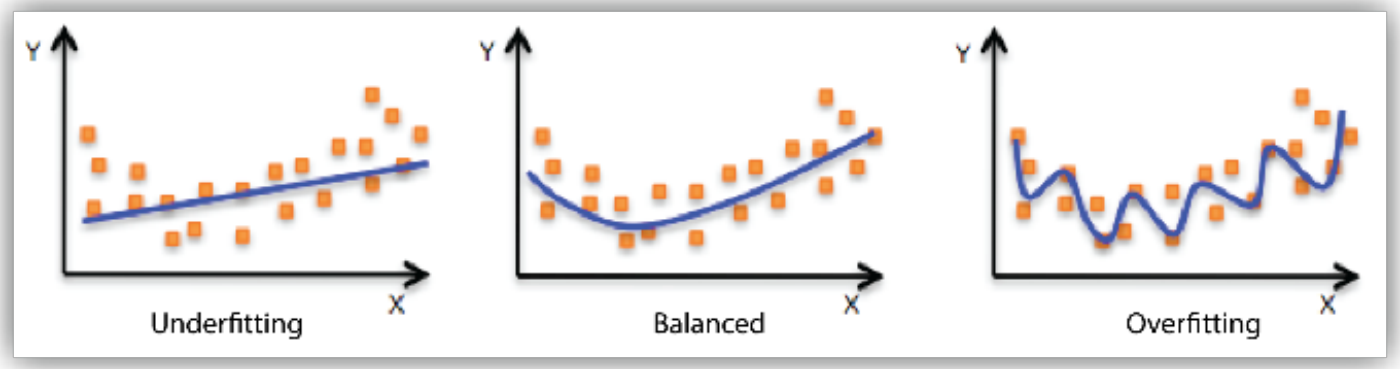
Better!

Cross-validation



Underfitting and overfitting





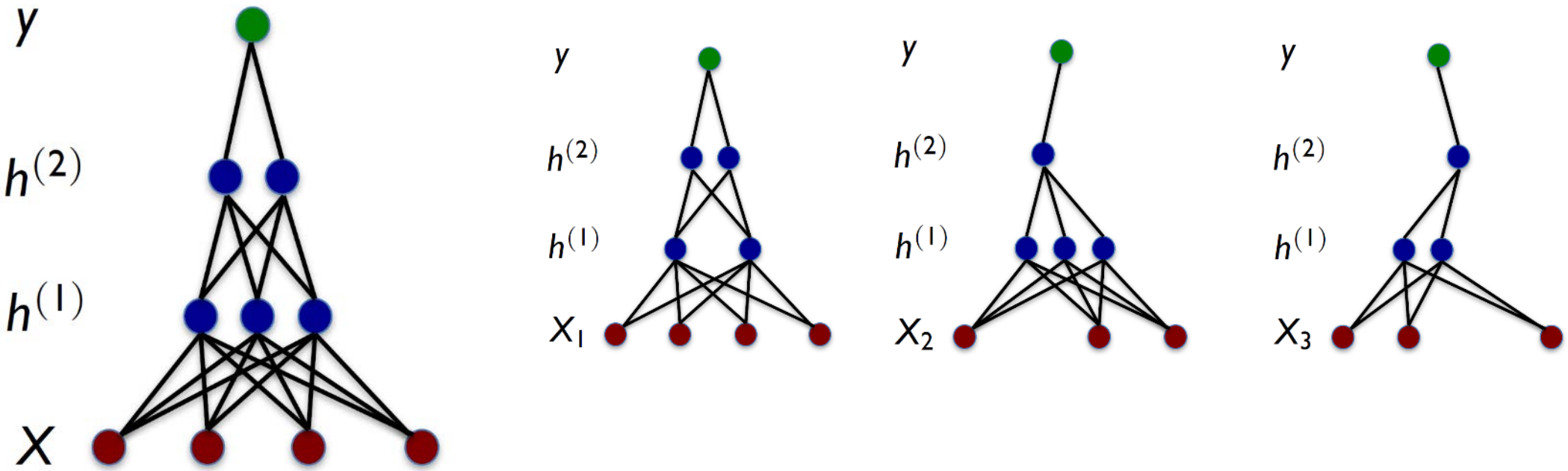
More complex models (with many more parameters than training examples) can always fit the 'training data' better but may do poorly on unseen data i.e. not generalize

Why?

Because data (inputs and outputs) is noisy. Complex models will start fitting noise.

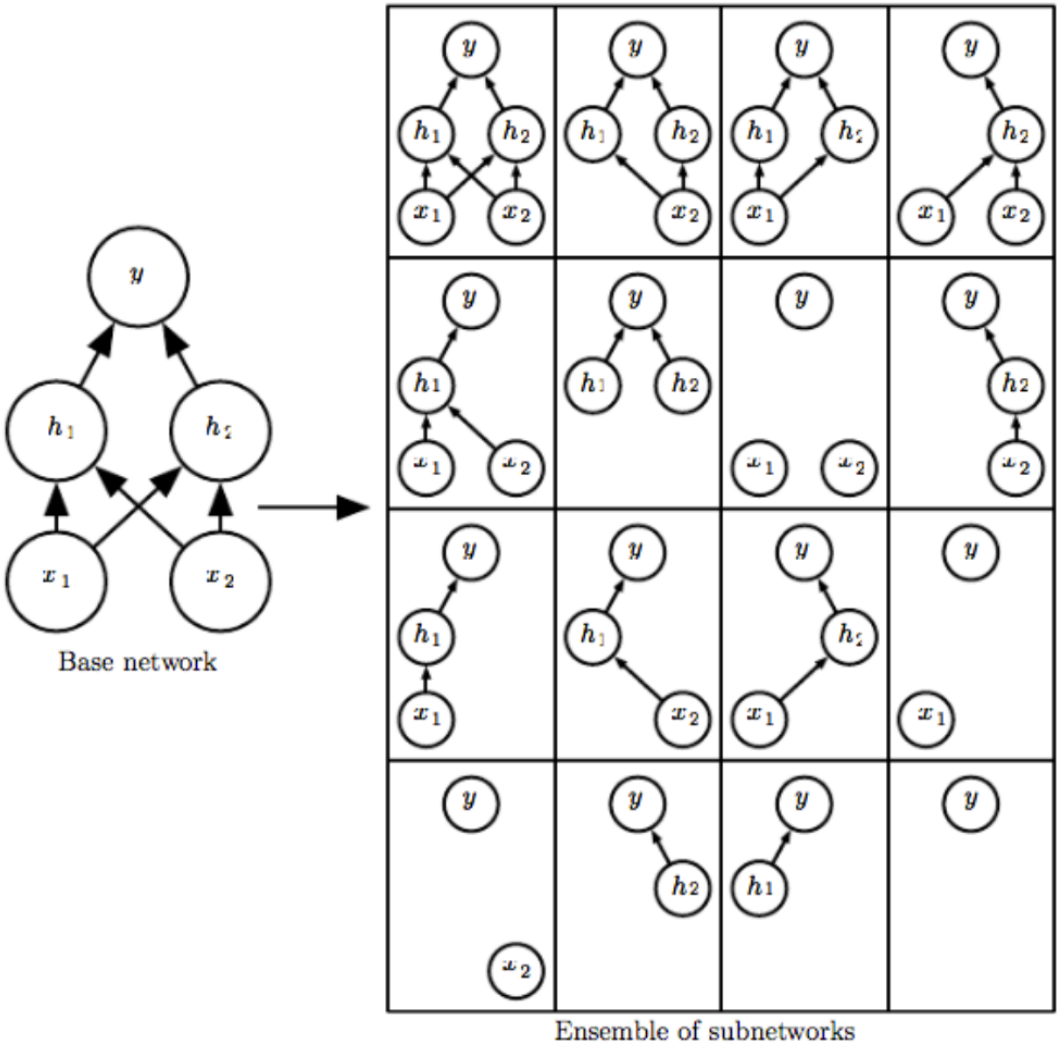
Dropout: A simple training strategy to avoid overfitting

While training using stochastic gradient descent, set each hidden/input unit to 0 with some dropout probability e.g. 0.5 for each training example



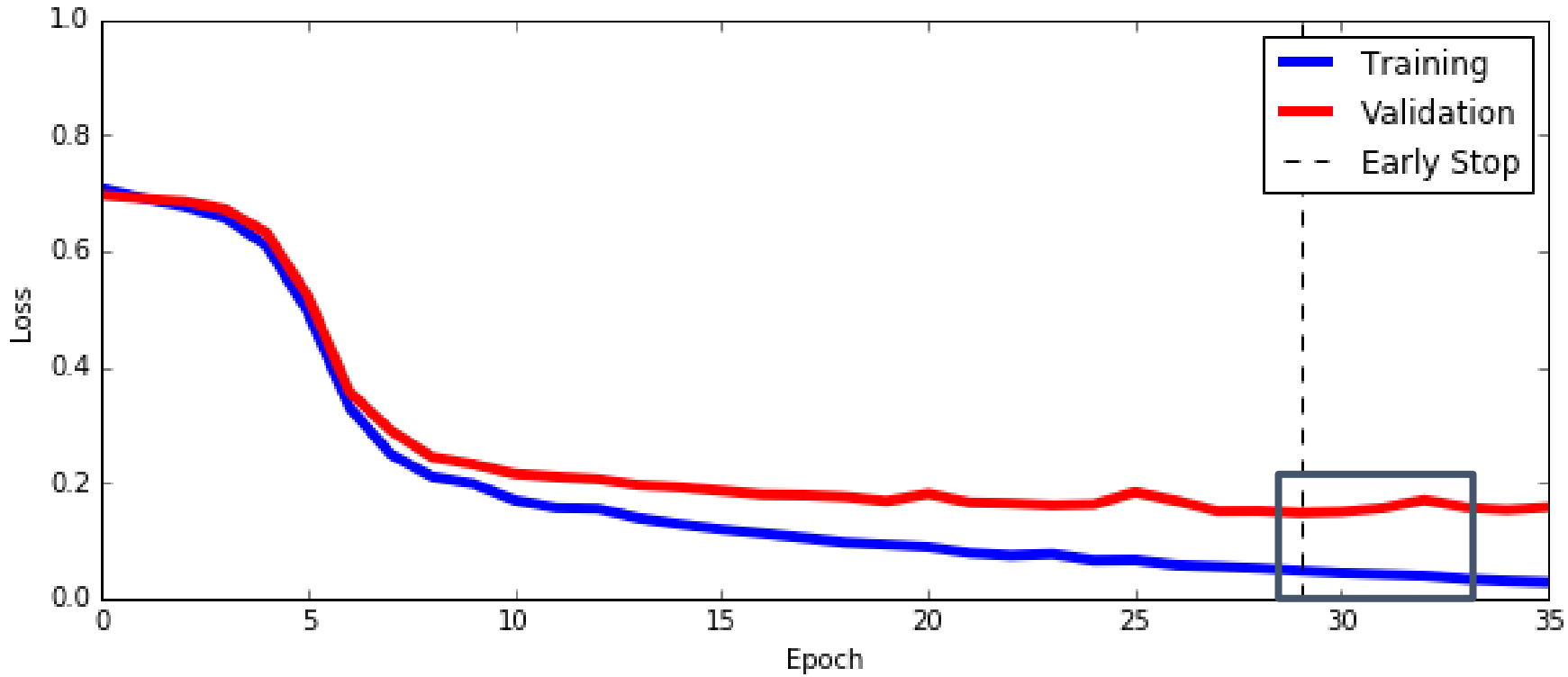
At test time, multiply the output of each unit by its dropout probability.

Why does dropout work?



Dropout is approximately training and averaging an exponentially large ensemble of networks.

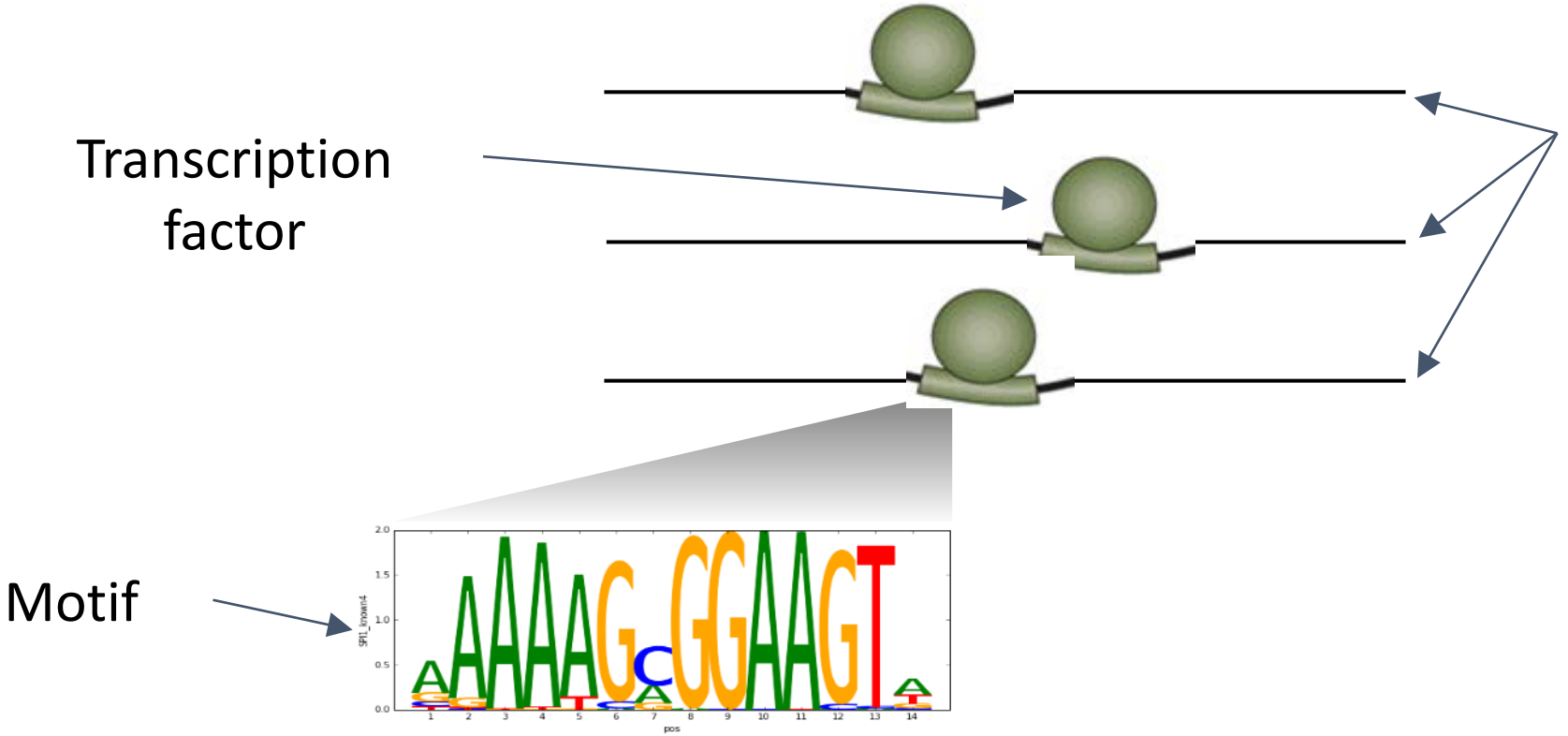
Early stopping using loss learning curve to avoid overfitting



It’s overfitting

Can we automatically learn predictive representations of the input sequences from the dataset instead of predefining arbitrary features?

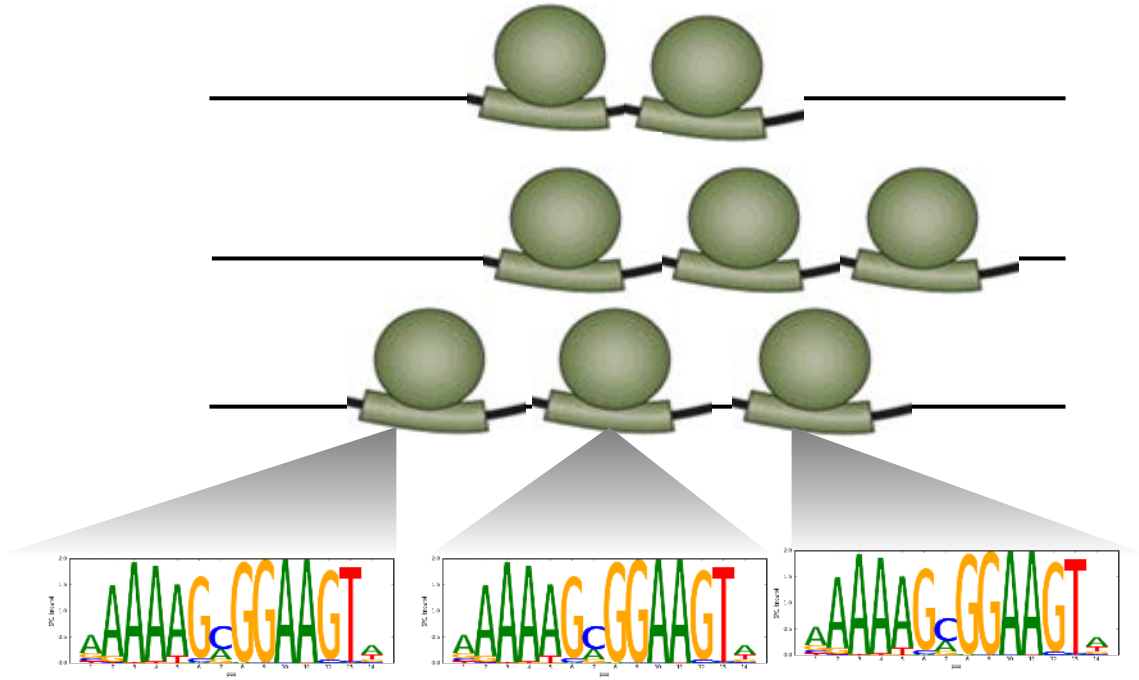
Key properties of regulatory sequence



TRANSCRIPTION FACTOR BINDING

Regulatory proteins called **transcription factors (TFs)** bind to high affinity sequence patterns (**motifs**) in regulatory DNA

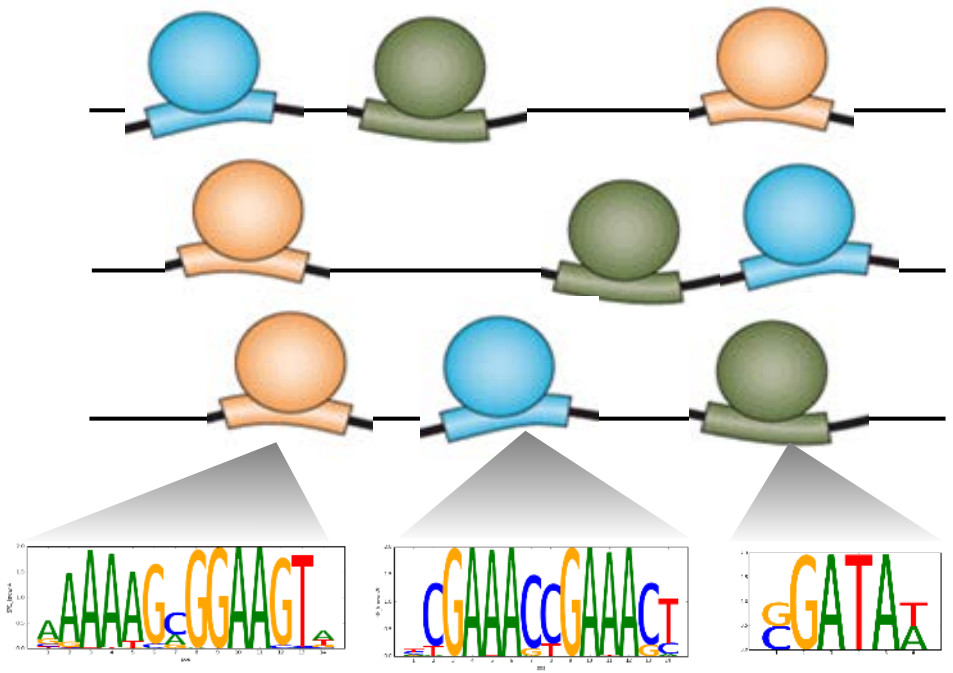
Key properties of regulatory sequence



HOMOTYPIC MOTIF DENSITY

Regulatory sequences often contain **more than one binding instance** of a TF resulting in **homotypic clusters of motifs of the same TF**

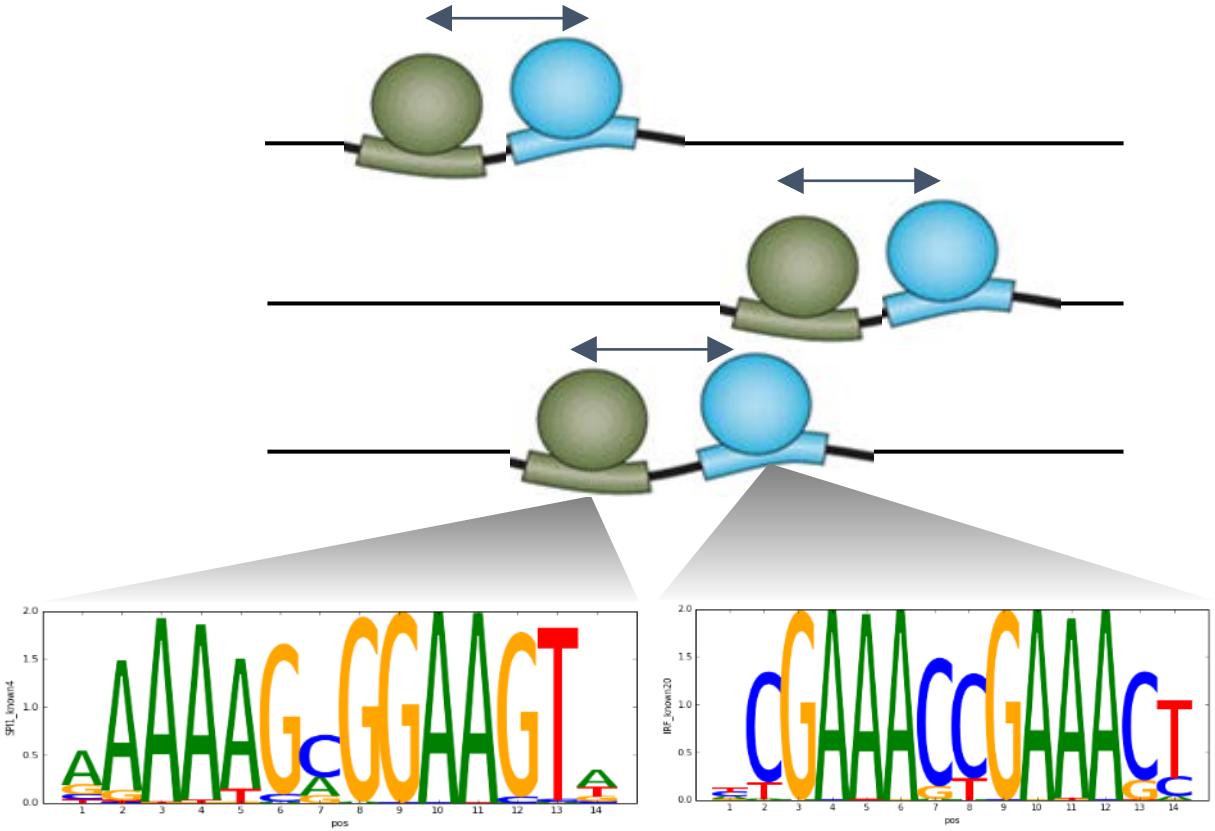
Key properties of regulatory sequence



HETEROTYPIC MOTIF COMBINATIONS

Regulatory sequences often bound by **combinations of TFs** resulting in **heterotypic clusters of motifs of different TFs**

Key properties of regulatory sequence



SPATIAL GRAMMARS OF HETEROTYPIC MOTIF COMBINATIONS

Regulatory sequences are often bound by **combinations of TFs** with specific **spatial and positional constraints** resulting in distinct **motif grammars**

Sequence motifs

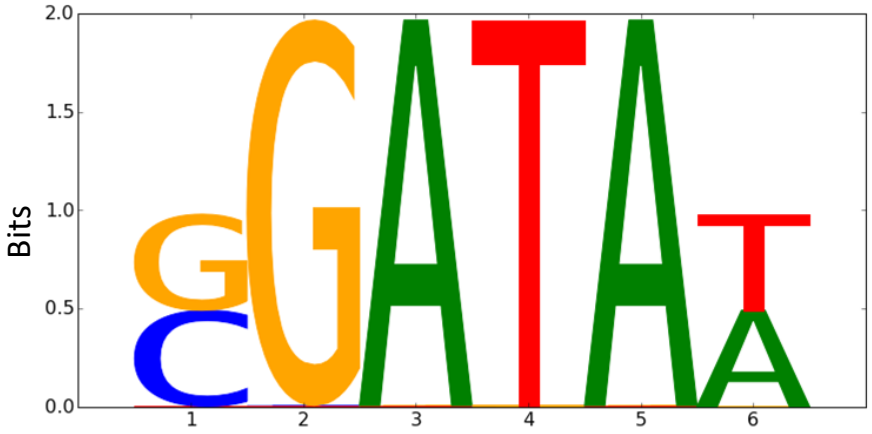
GGATAA
CGATAA
CGATAT
GGATAT

Set of aligned sequences
Bound by TF

$$p_i(x_i = a_i)$$

A	0	0	1	0	1	0.5
C	0.5	0	0	0	0	0
G	0.5	1	0	0	0	0
T	0	0	0	1	0	0.5

Position weight matrix
(PWM)



PWM
logo

https://en.wikipedia.org/wiki/Sequence_logo

The information content (y-axis) of position i is given by:[2]

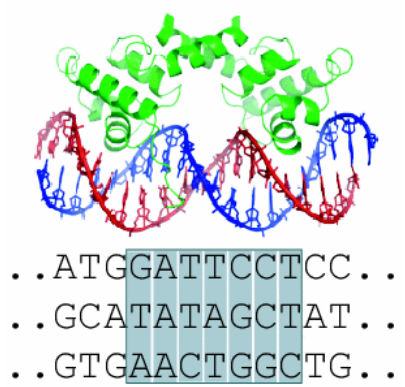
$$R_i = \log_2(4) - (H_i + e_n)$$

where H_i is the uncertainty (sometimes called the Shannon entropy) of position i

$$H_i = -\sum f_{a,i} \times \log_2 f_{a,i}$$

. The height of letter a in column i is given by

$$\text{height} = f_{a,i} \times R_i$$



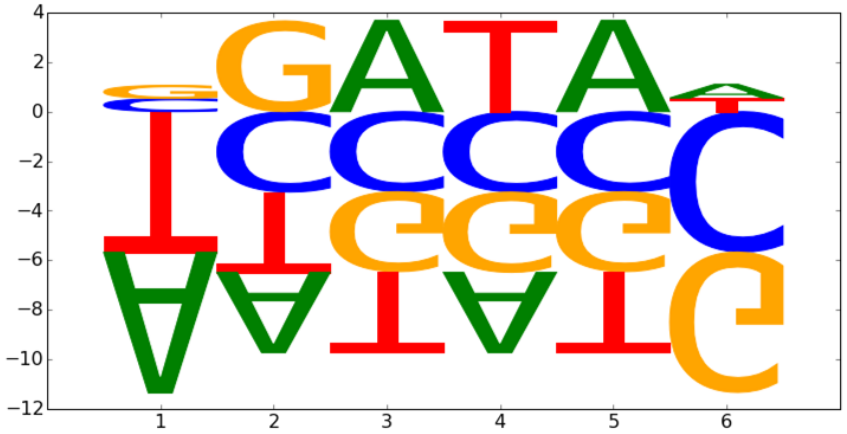
Sequence motifs

Accounting for genomic background nucleotide distribution

Position-specific scoring matrix (PSSM)

$$\log_2\left(\frac{p_i(x_i = a_i)}{p_{bg}(x_i = a_i)}\right)$$

A	-5.7	-3.2	3.7	-3.2	3.7	0.6
C	0.5	-3.2	-3.2	-3.2	-3.2	-5.7
G	0.5	3.7	-3.2	-3.2	-3.2	-5.7
T	-5.7	-3.2	-3.2	3.7	-3.2	0.5



PSSM logo

Scoring a sequence with a motif PSSM

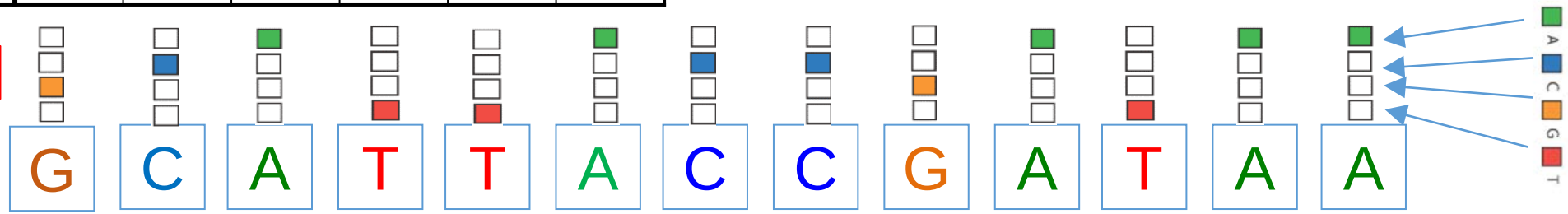
PSSM parameters

Scoring weights w

A	-5.7	-3.2	3.7	-3.2	3.7	0.6
C	0.5	-3.2	-3.2	-3.2	-3.2	-5.7
G	0.5	3.7	-3.2	-3.2	-3.2	-5.7
T	-5.7	-3.2	-3.2	3.7	-3.2	0.5

One-hot encoding $x[b, j]$

Input sequence S



Convolution

Scoring a sequence with a PSSM

Motif match Scores

$\sum(\boldsymbol{w} .* x[:, 1:6])$

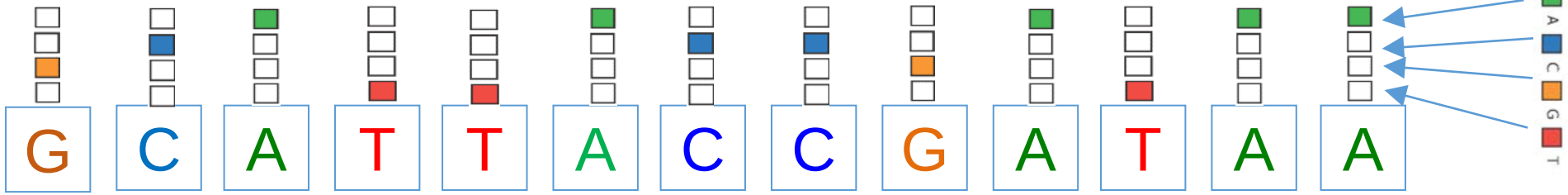
2.0												

Scoring weights $\boldsymbol{w}$

A	-5.7	-3.2	3.7	-3.2	3.7	0.6
C	0.5	-3.2	-3.2	-3.2	-3.2	-5.7
G	0.5	3.7	-3.2	-3.2	-3.2	-5.7
T	-5.7	-3.2	-3.2	3.7	-3.2	0.5

One-hot encoding ($x_{b,j}$)

Input sequence



Convolution

Motif match Scores
$\sum(\boldsymbol{w} .* x[:, 2:7])$

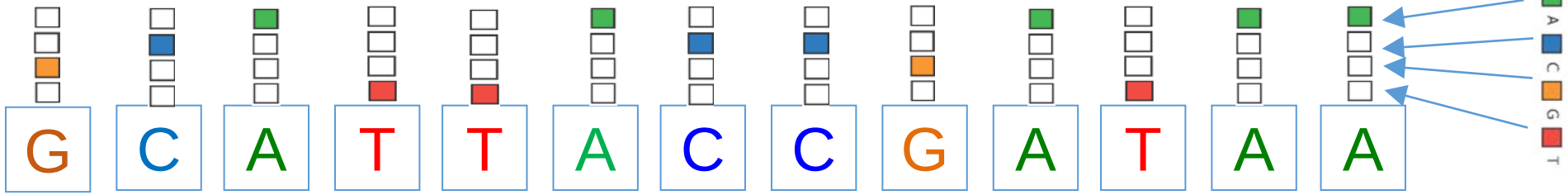
2.0	-4.3											

Scoring weights $\boldsymbol{w}$

A	-5.7	-3.2	3.7	-3.2	3.7	0.6
C	0.5	-3.2	-3.2	-3.2	-3.2	-5.7
G	0.5	3.7	-3.2	-3.2	-3.2	-5.7
T	-5.7	-3.2	-3.2	3.7	-3.2	0.5

One-hot encoding ($x_{b,j}$)

Input sequence



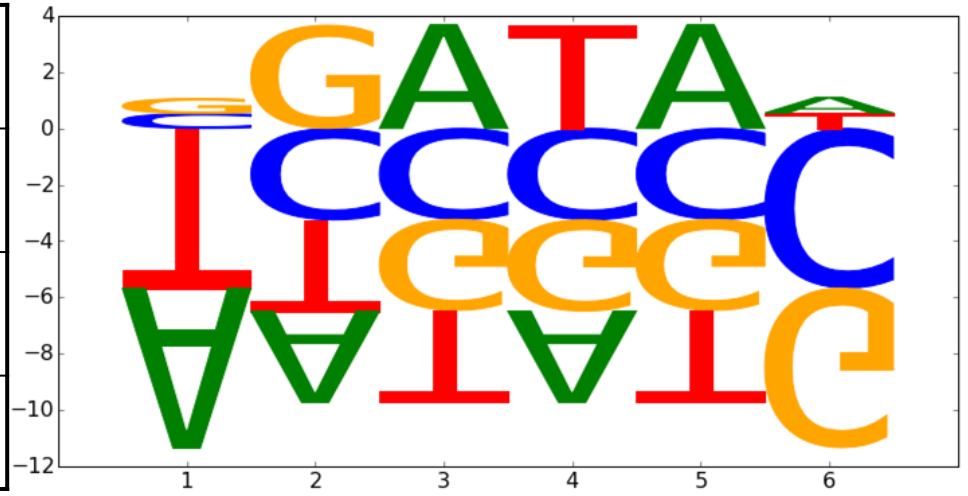
Convolution

Motif match Scores
$\sum(\boldsymbol{w} .* x[:, i: i+5])$

2.0	-4.3	-24	-17	-18	-11	-12	16	-5.5	-8.5	-5.2		

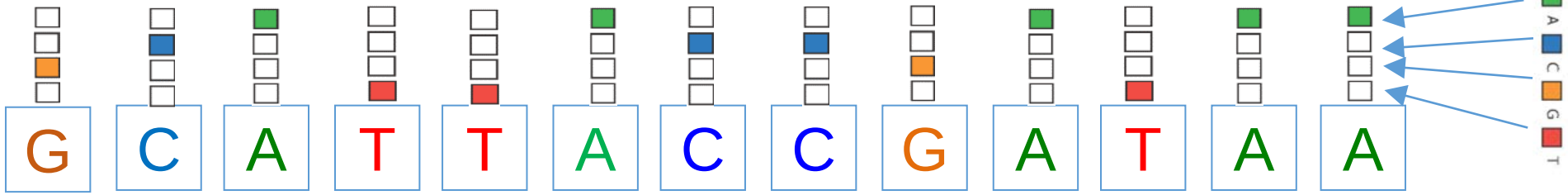
Scoring weights $\boldsymbol{w}$

A	**-5.7**	**-3.2**	**3.7**	**-3.2**	**3.7**	**0.6**
C	**0.5**	**-3.2**	**-3.2**	**-3.2**	**-3.2**	**-5.7**
G	**0.5**	**3.7**	**-3.2**	**-3.2**	**-3.2**	**-5.7**
T	**-5.7**	**-3.2**	**-3.2**	**3.7**	**-3.2**	**0.5**



One-hot encoding ($x_{b,j}$)

Input sequence



Thresholding scores

Thresholded Scores
$\max(0, \sum(\boldsymbol{w} .* x[:, i: i+5])$

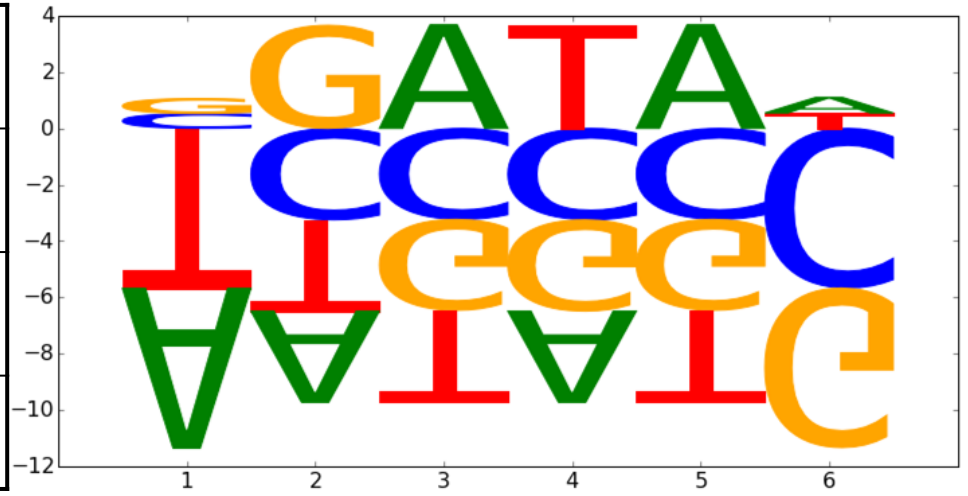
2.0	0	0	0	0	0	0	16	0	0	0		

Motif match Scores
$\sum(\boldsymbol{w} .* x[:, i: i+5])$

2.0	-4.3	-24	-17	-18	-11	-12	16	-5.5	-8.5	-5.2		

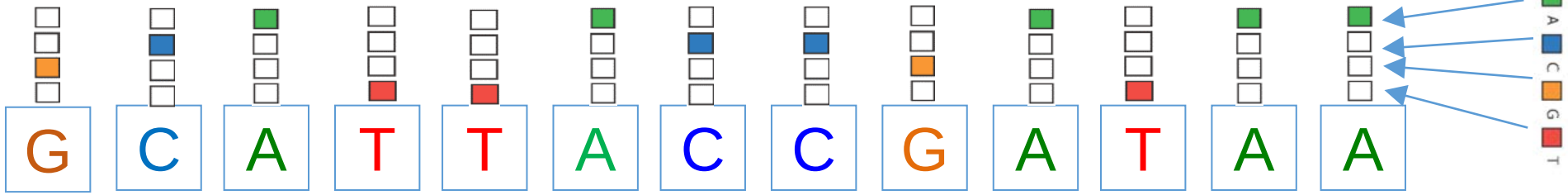
Scoring weights $\boldsymbol{w}$

A	**-5.7**	**-3.2**	**3.7**	**-3.2**	**3.7**	**0.6**
C	**0.5**	**-3.2**	**-3.2**	**-3.2**	**-3.2**	**-5.7**
G	**0.5**	**3.7**	**-3.2**	**-3.2**	**-3.2**	**-5.7**
T	**-5.7**	**-3.2**	**-3.2**	**3.7**	**-3.2**	**0.5**



One-hot encoding ($x_{b,j}$)

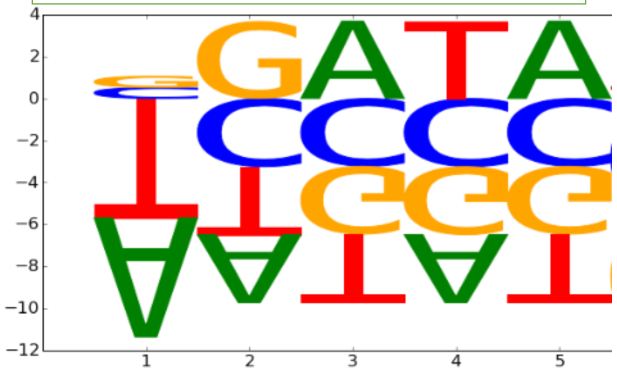
Input sequence



An artificial neuron on DNA sequence is a motif pattern detector

Neuron = Motif: (position weight matrix)!

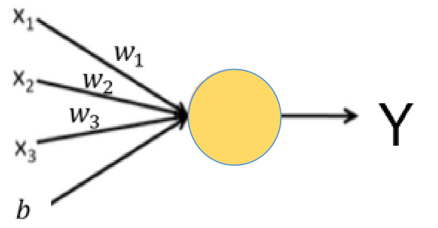
Binary Output: Yes (1) vs No (0)



Is TF bound? **Y** = 11.6

Z =W*X = 0.5+3.7+3.7+3.7 = 11.6

A	-5.7	-3.2	3.7	-3.2	3.7
C	0.5	-3.2	-3.2	-3.2	-3.2
G	0.5	3.7	-3.2	-3.2	-3.2
T	-5.7	-3.2	-3.2	3.7	-3.2



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

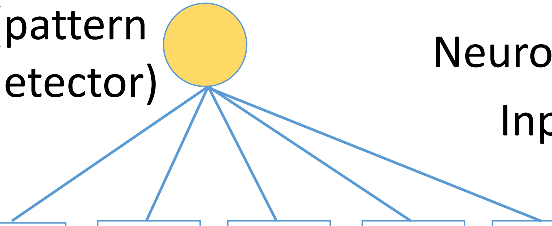
weight parameters

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

$$Y = h(Z)$$

Non-linear function

Artificial neuron (pattern detector)

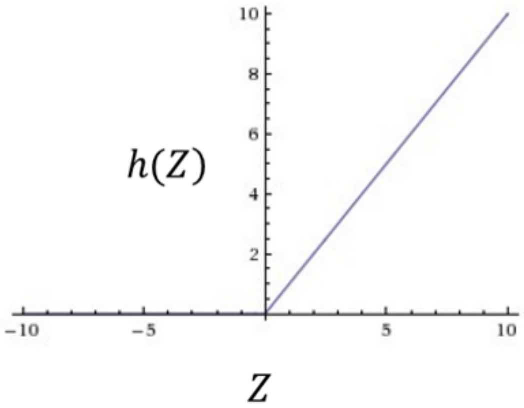


Neuron weights $\mathbf{W}_{1..20}$ (20 values = 5 positions x 4 nucleotides)

Input $\mathbf{X}_{1..20}$ (20 binary values = 5 positions x 4 nucleotides)

ReLu (Rectified Linear Unit)

Useful for thresholding



	C	G	A	T	A	A	C	C	G	A	T	A	T
A	0	0	1	0	1	1	0	0	0	1	0	1	0
C	1	0	0	0	0	0	1	1	0	0	0	0	0
G	0	1	0	0	0	0	0	0	1	0	0	0	0
T	0	0	0	1	0	0	0	0	0	0	1	0	1

One-hot encoded input: DNA sequence represented as ones and zeros

Deep convolutional neural network: Scanning pattern detectors

Binary Output:
Yes (1) vs No (0)

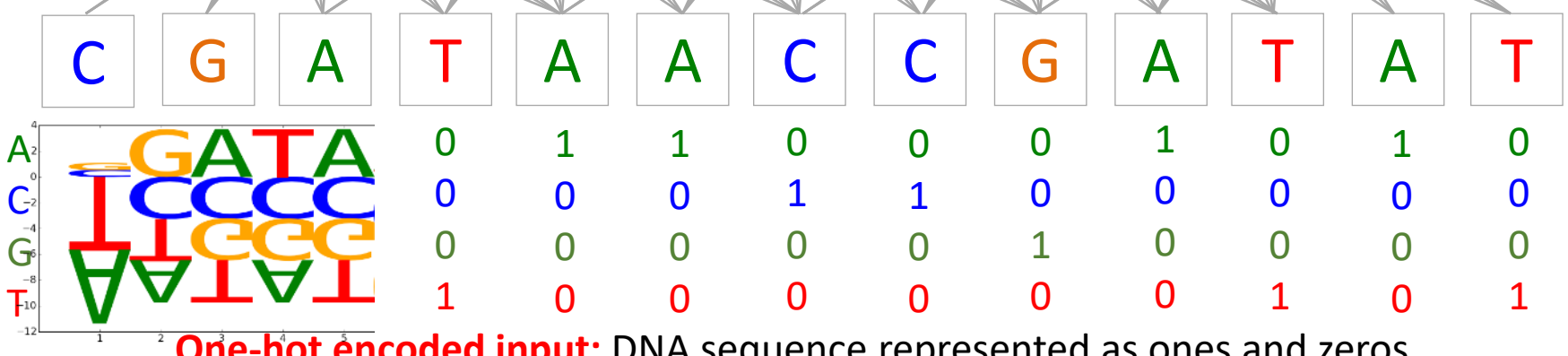
Is TF bound?

Combinations of motifs
(motif grammars)

3 artificial neurons
(motif detectors)

Advantages of convnets

- Good at capturing local patterns
- Translational invariance (with pooling layers .. Next slide). Patterns can occur at different positions in the input.
- Significant reduction in parameters compared to denseNets



One-hot encoded input: DNA sequence represented as ones and zeros

Pooling operations allow positional invariance

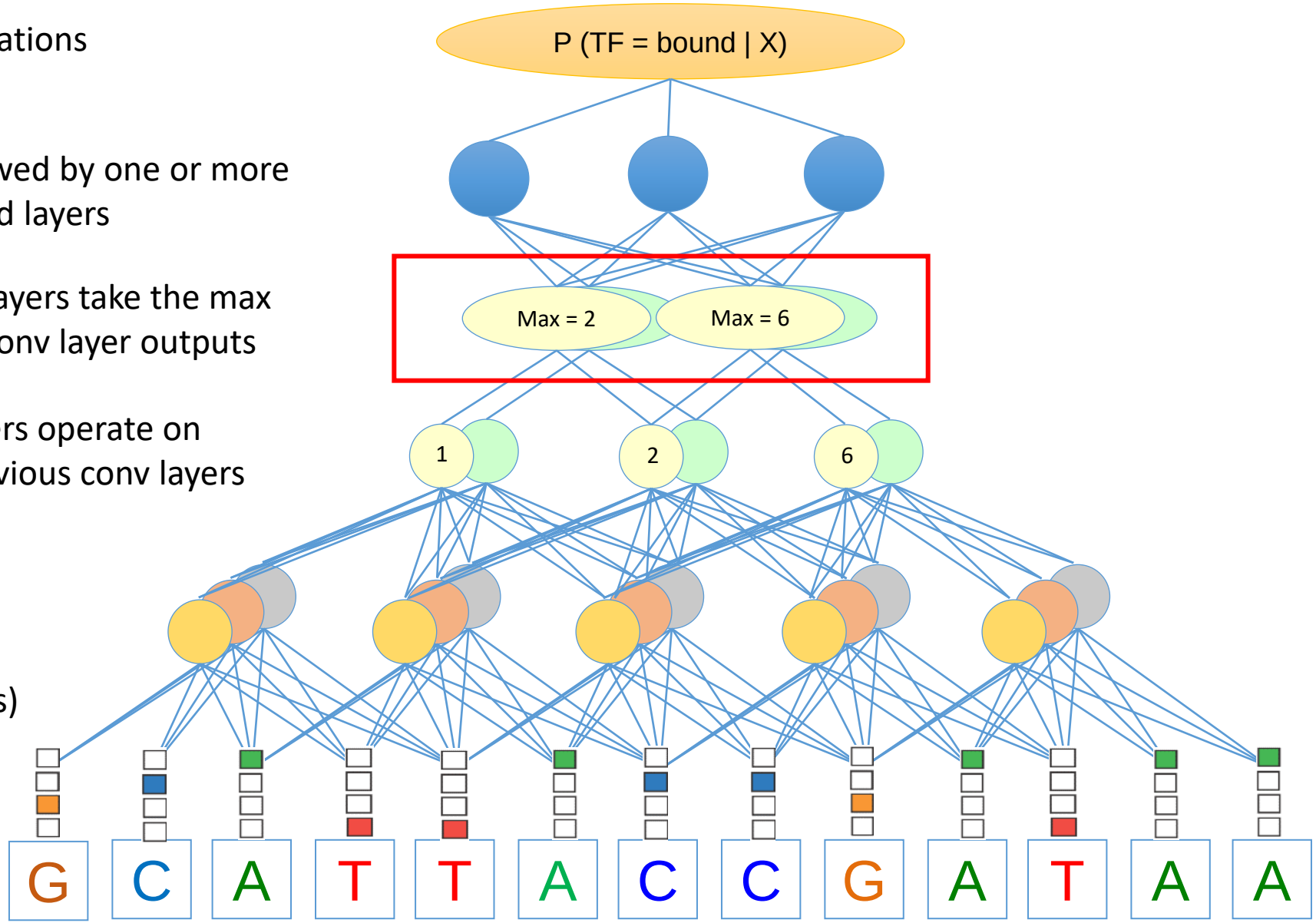
Sigmoid activations

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)



Maxpooling layer
pool width = 2
stride = 1

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

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

***for genomics, a stride of 1 for conv layers is recommended**

Multi-task or multi-output model

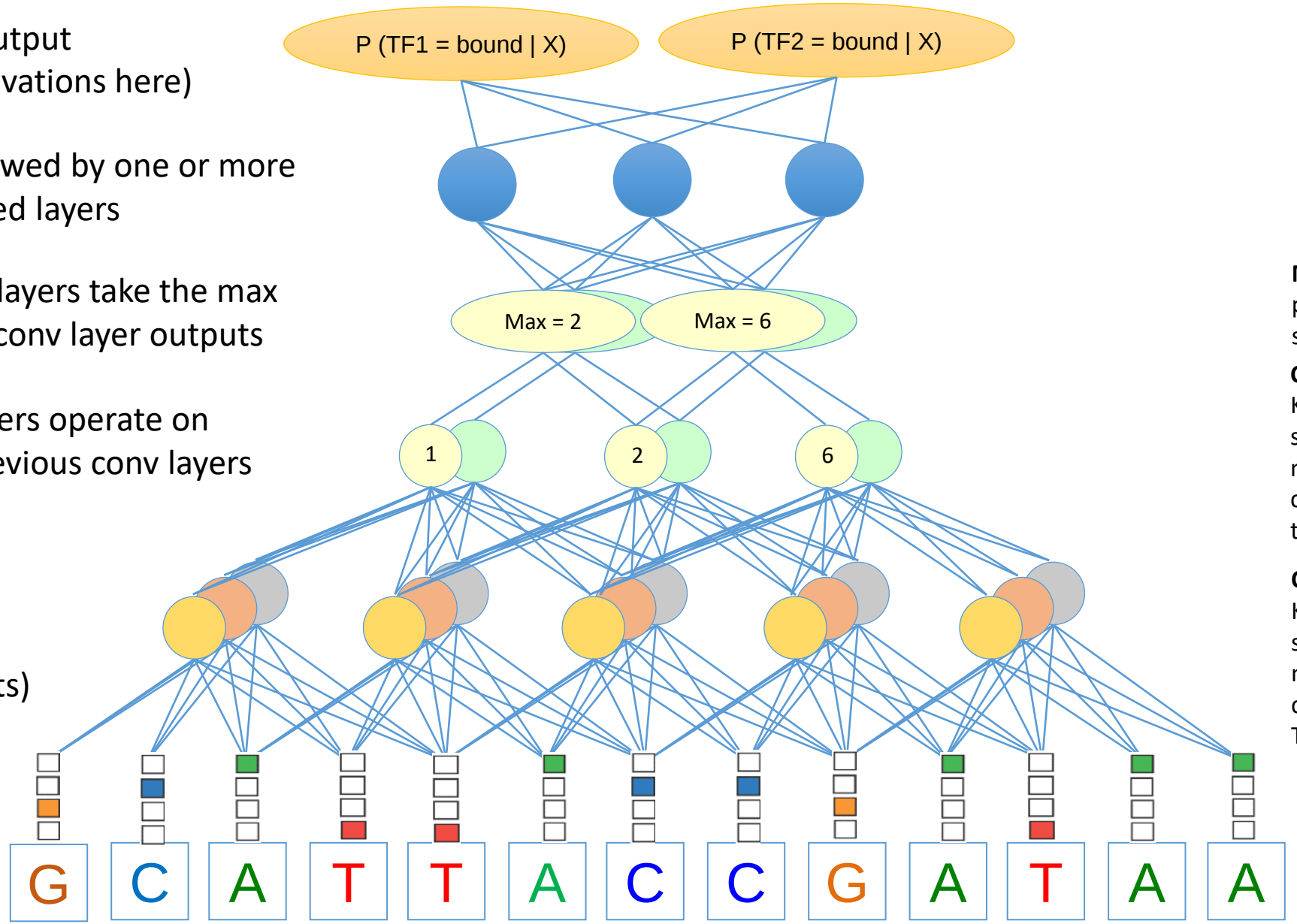
Multi-task output
(sigmoid activations here)

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)

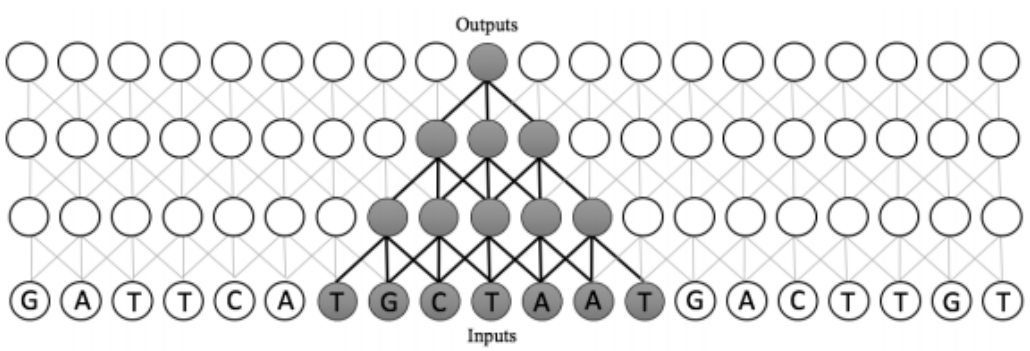


Maxpooling layer
pool width = 2
stride = 1

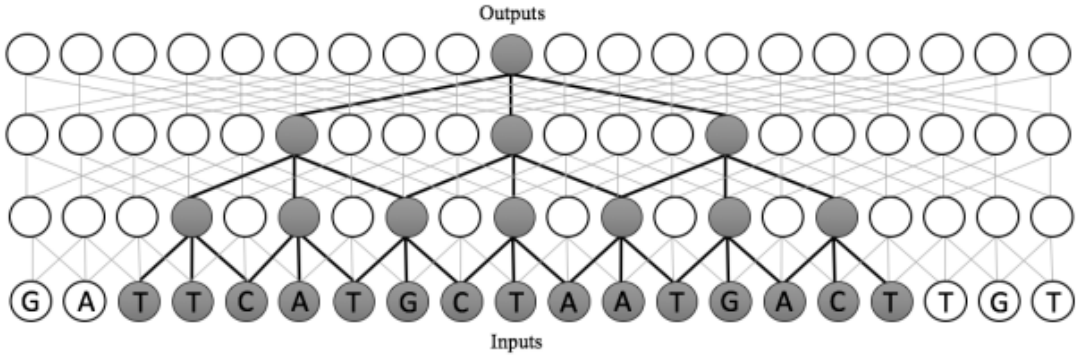
Conv Layer 2
Kernel width = 3
stride = 1
num filters / num channels = 2
total neurons = 6

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

Dilated convolutions instead of convolutions



(a) Convolution



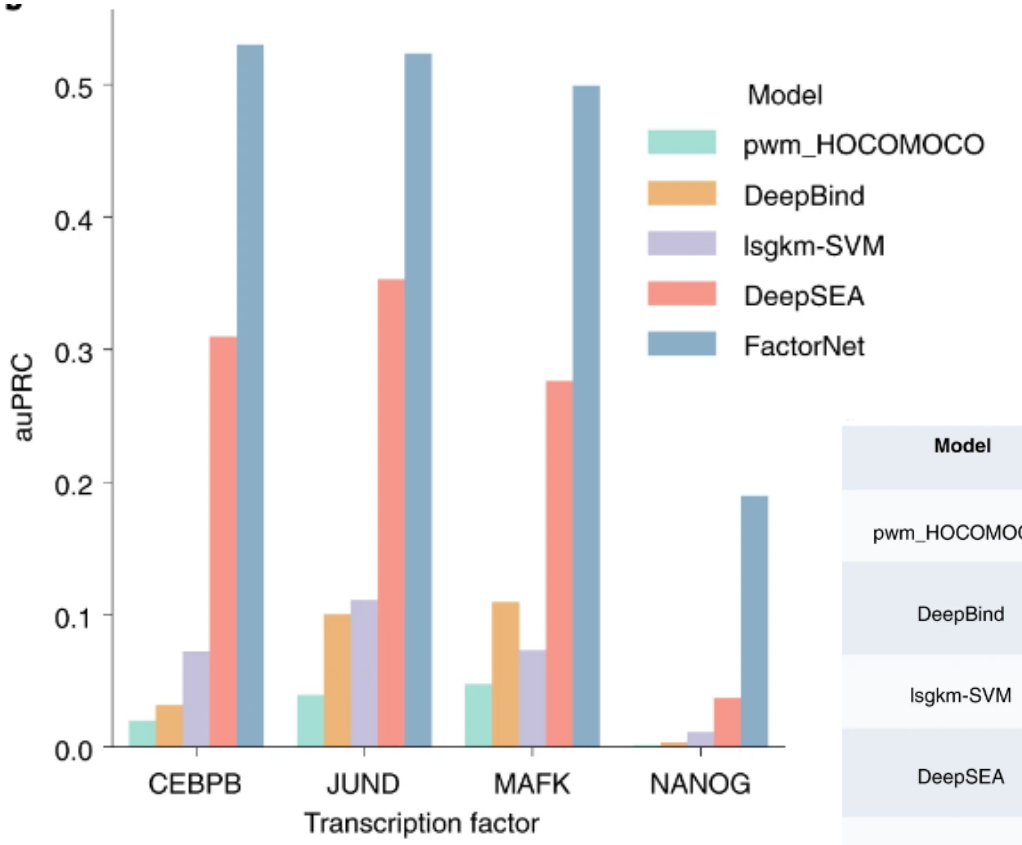
(c) Dilated Convolution

- For CNNs, the size of the receptive field is linear in the number of layers and the kernel width. Thus, scaling the receptive field to incorporate a large input introduces more layers, making training more difficult.
- Dilated CNNs: Larger receptive field but few parameters and short distance gradient propagation

https://arxiv.org/pdf/1710.01278.pdf

https://www.inference.vc/dilated-convolutions-and-kronecker-factorisation/

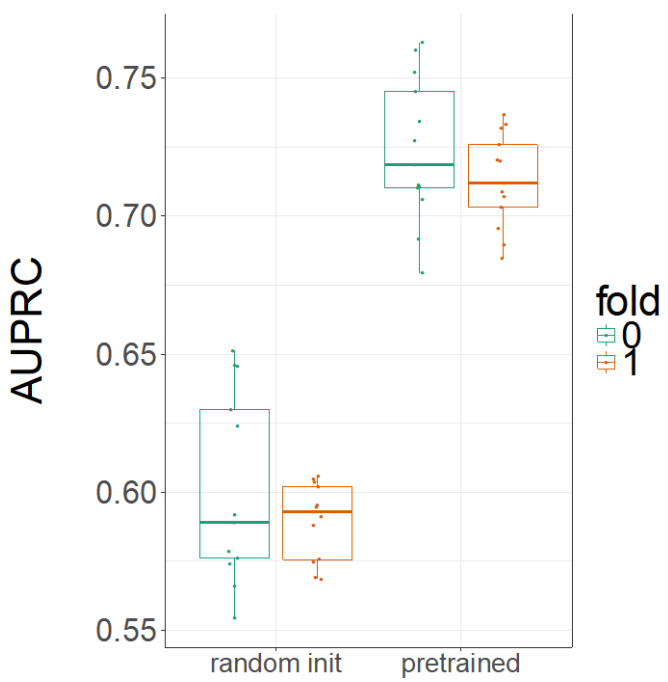
Different deep learning models can perform very differently depending on problem formulation and training methods

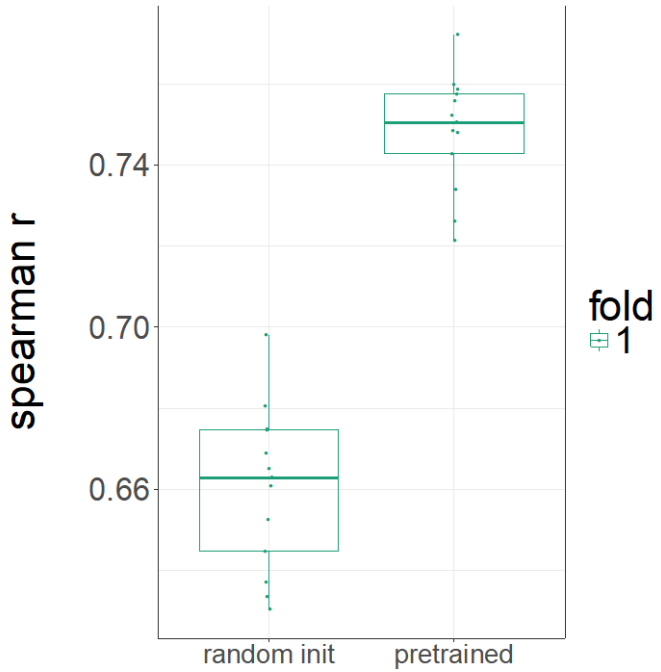


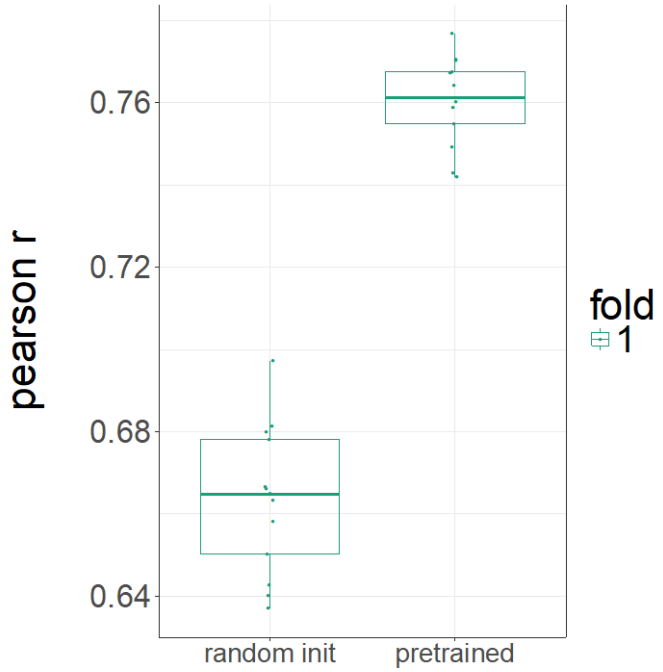
Model	Publication	Type	Input modalities	Framework	Size	Prediction time ($n = 256$)	Input sequence length
pwm_HOCOMOCO	Kulakovskiy et al. 2015	Position weight matrix scan	DNA sequence	Keras	16 KB	1.2 ms	101 bp
DeepBind	Alipanahi et al. 2015	Convolutional neural network	DNA sequence	Keras	36 KB	2.4 ms	101 bp
lsgkm-SVM	Ghandi et al. 2014	Support vector machine	DNA sequence	LS-GKM	4 MB	10 s	101 bp
DeepSEA	Zhou et al. 2015	Convolutional neural network	DNA sequence	PyTorch	211 MB	94 ms	1,000 bp
FactorNet	Quang et al. 2017	Convolutional and recurrent neural network	DNA sequence + DNase + annotation	Keras	3–13 MB	118 ms	1,002 bp

https://www.nature.com/articles/s41587-019-0140-0

Transfer learning: Improving performance by borrowing from pre-trained reference models







Parameters initialized using multi-task reference model trained on 1000s of ENCODE/Roadmap datasets across 100s of cell types/tissues

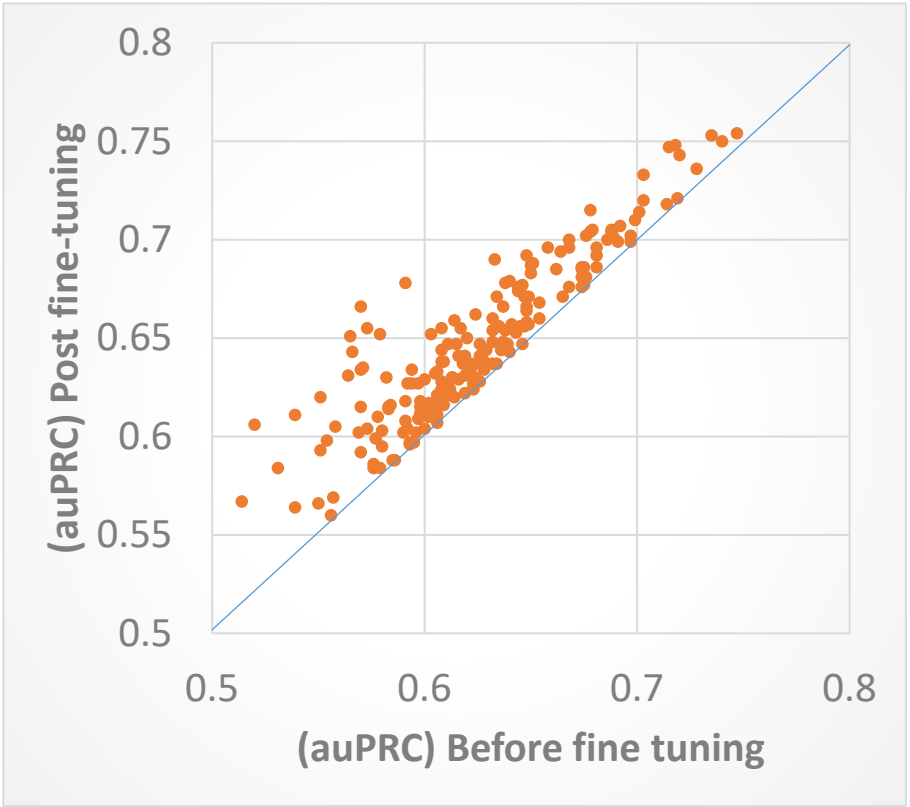
‘Naïve’ multi-tasking does not always help

- This is NOT an architecture problem. It’s a learning problem.
- Simply optimizing average loss across tasks biases towards specific tasks and task similarity
- Especially problematic when tasks have highly varying class imbalance and also differences in relatedness
- How you weight tasks matters

Hacky but easy fix

- Fine tune the multi-task model on each task after the multi-task training

Chromatin accessibility models across 200 mouse cell types/tissues



Reviews

Review Article | Published: 10 April 2019

Deep learning: new computational modelling techniques for genomics

Gökcen Eraslan, Žiga Avsec, Julien Gagneur & Fabian J. Theis

Nature Reviews Genetics **20**, 389–403 (2019) | Cite this article

66k Accesses | **367** Citations | **488** Altmetric | Metrics

https://www.nature.com/articles/s41576-019-0122-6

Review | 29 July 2016 | OPEN ACCESS

Deep learning for computational biology

Christof Angermueller, Tanel Pärnamaa, Leopold Parts, Oliver Stegle

Author Information

Molecular Systems Biology (2016) 12: 878 | https://doi.org/10.15252/msb.20156651

https://www.embopress.org/doi/full/10.15252/msb.20156651

Opportunities and obstacles for deep learning in biology and medicine

Travers Ching†, Daniel S. Himmelstein, Brett K. Beaulieu-Jones, Alexandr A. Kalinin, Brian T. Do, Gregory P. Way, Enrico Ferrero, Paul-Michael Agapow, Michael Zietz, Michael M. Hoffman, Wei Xie, Gail L. Rosen, Benjamin J. Lengerich, Johnny Israeli, Jack Lanchantin, Stephen Woloszynek, Anne E. Carpenter, Avanti Shrikumar, Jinbo Xu, Evan M. Cofer, Christopher A. Lavender, Srinivas C. Turaga, Amr M. Alexandari, Zhiyong Lu, David J. Harris, Dave DeCaprio, Yanjun Qi, Anshul Kundaje, Yifan Peng, Laura K. Wiley, Marwin H. S. Segler, Simina M. Boca, S. Joshua Swamidass, Austin Huang , Anthony Gitter and Casey S. Greene See fewer authors

Published: 04 April 2018 | https://doi.org/10.1098/rsif.2017.0387

https://royalsocietypublishing.org/doi/10.1098/rsif.2017.0387