CS273B: Deep learning for Genomics and Biomedicine

Lecture 3: Convolutional neural networks for genomics and imaging data

10/03/2016

Anshul Kundaje, James Zou, Serafim Batzoglou

Administration stuff

- Start forming project teams! (5-6 students)
- Smaller teams are allowed if you have your own compute.
- If you need help finding team members, message on Piazza and message the TAs.
- Teams must be finalized by next Wednesday
- We will release suggested project topics and descriptions by weekend. You are free to pick your own projects.
- We will poll teams for preferences for paper presentations and try to accommodate requests as much as possible.
- If you are auditing the course and want to get added to Canvas, meet TAs after class.

Outline

- Multi-modal convolutional neural networks for predicting protein-DNA binding maps
- Convolutional neural networks on images

Convolutional neural networks for learning from DNA sequence

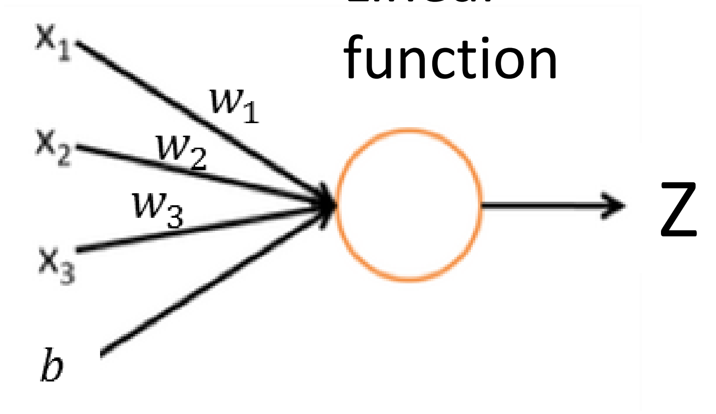
A simple classifier (An artificial neuron)

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

parameters

$$Z = \boxed{w_1}.x_1 + \boxed{w_2}.x_2 + \boxed{w_3}.x_3 + \boxed{b}$$

Linear function



Training the neuron means learning the optimal w's and b

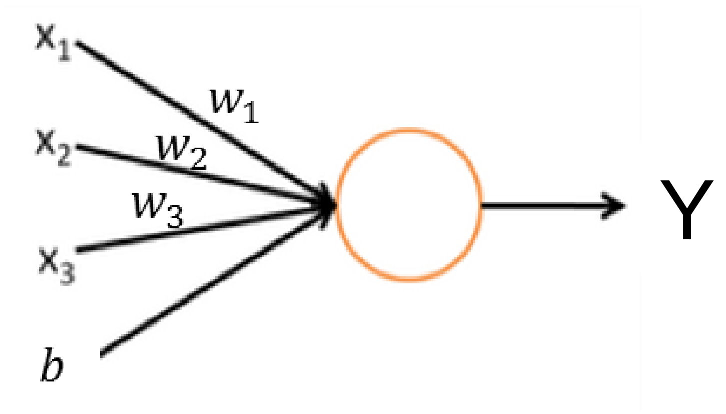
A simple classifier (An artificial neuron)

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

parameters

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

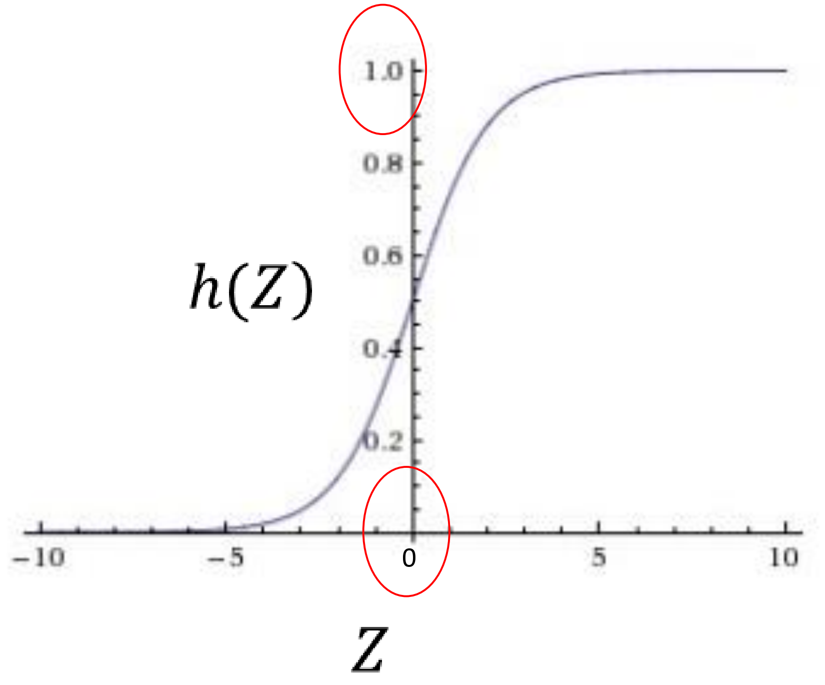
$$Y = h(Z)$$



Logistic / Sigmoid

Useful for predicting probabilities

Non-linear function



Training the neuron means learning the optimal w's and b

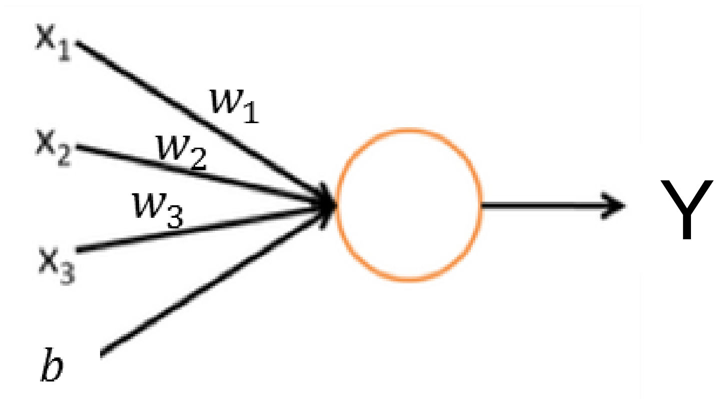
A simple classifier (An artificial neuron)

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

parameters

$$Z = \underbrace{w_1}.x_1 + \underbrace{w_2}.x_2 + \underbrace{w_3}.x_3 + \underbrace{b}$$

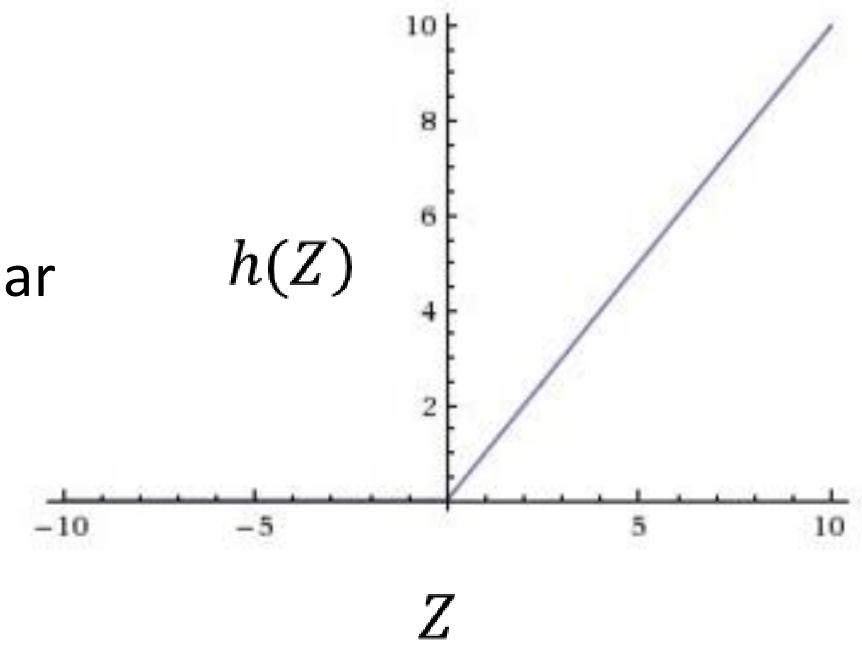
$$Y = h(Z)$$



ReLu (Rectified Linear Unit)

Useful for thresholding

Non-linear function



Training the neuron means learning the optimal w's and b

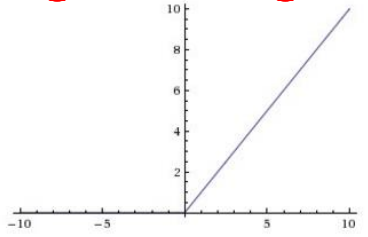
Artificial neuron can represent a motif

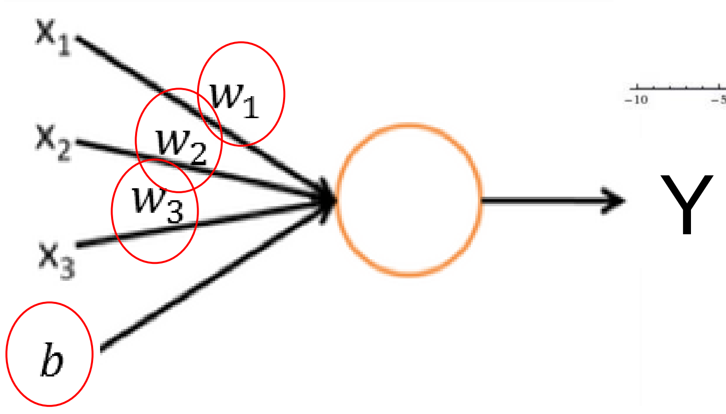
$$Y = F(x_1, x_2, x_3)$$

parameters

$$\boxed{Z} = w_1 . x_1 + w_2 . x_2 + w_3 . x_3 + b$$

$$Y = \boxed{h(Z)}$$





Thresholded Motif Scores max(0, W*x)	0	0	2.0	0	0	0

Motif match Scores sum(W * x)	-2.2	-5.4	2.0	-4.3	-24	-17

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)	☐ ☐ 🟧 ☐	☐ 🟦 ☐ ☐	🟩 ☐ ☐ ☐	☐ ☐ ☐ 🟥	☐ ☐ ☐ 🟥	🟩 ☐ ☐ ☐
Input sequence	G	C	A	T	T	A

Multi-task CNN

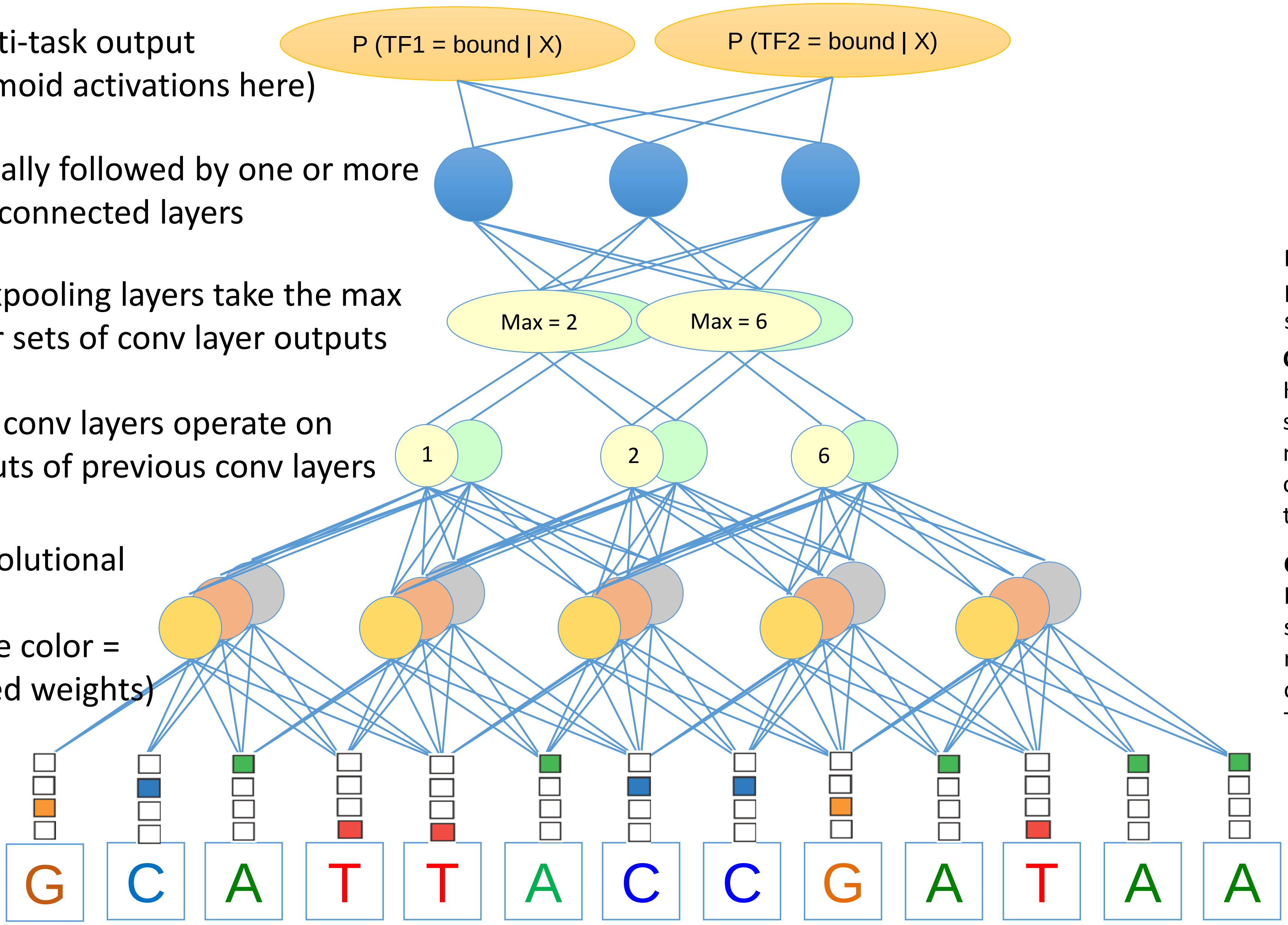
Multi-task output
(sigmoid activations here)

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

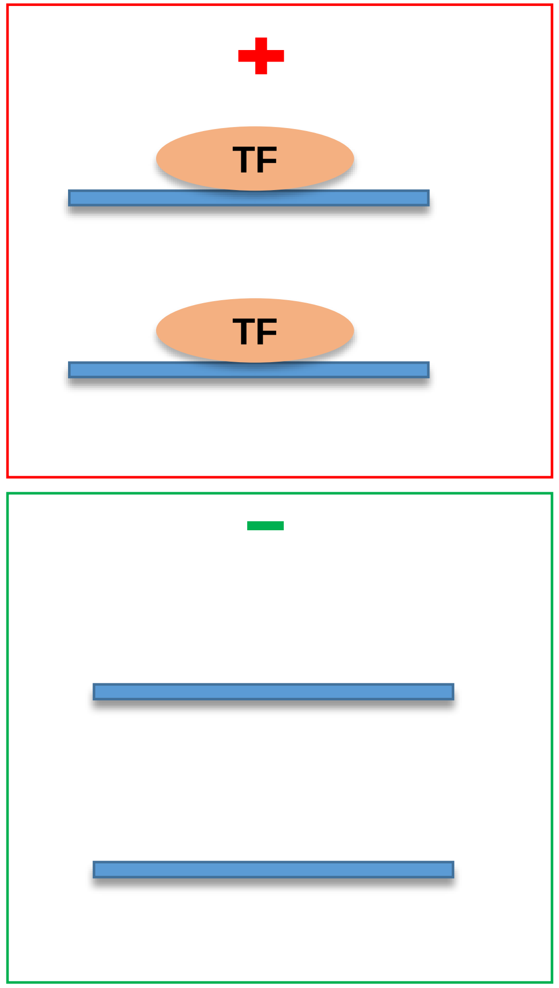
Multi-modal convolutional neural networks for predicting protein-DNA binding

Learning patterns in regulatory DNA sequence

- Positive class of genomic sequences bound a transcription factor of interest

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

- Negative class of genomic sequences not bound by a transcription factor of interest



Predicting binding in new cell types

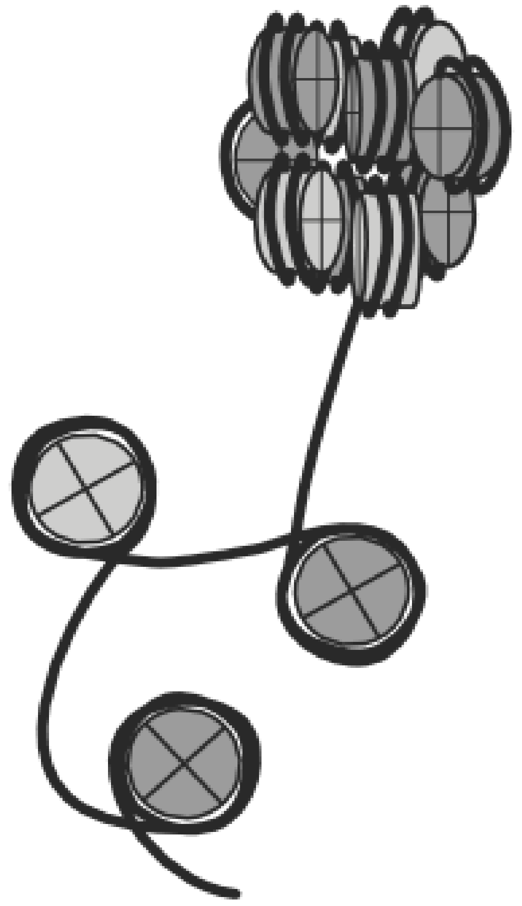
Sequence is static across cell types.

A sequence only model cannot generate cell-type specific binding predictions with no other input.

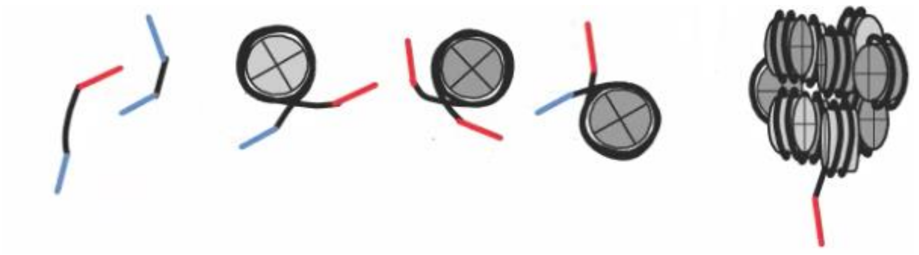
We need an some other type of input data type that provides some information about cell-type specific use of DNA sequence

Chromatin accessibility

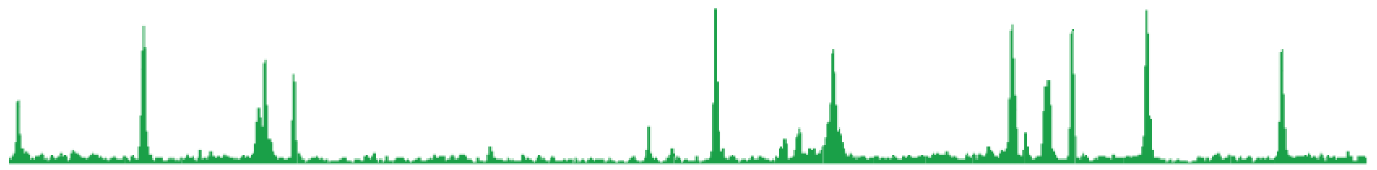
ATAC-Seq



Greenleaf lab



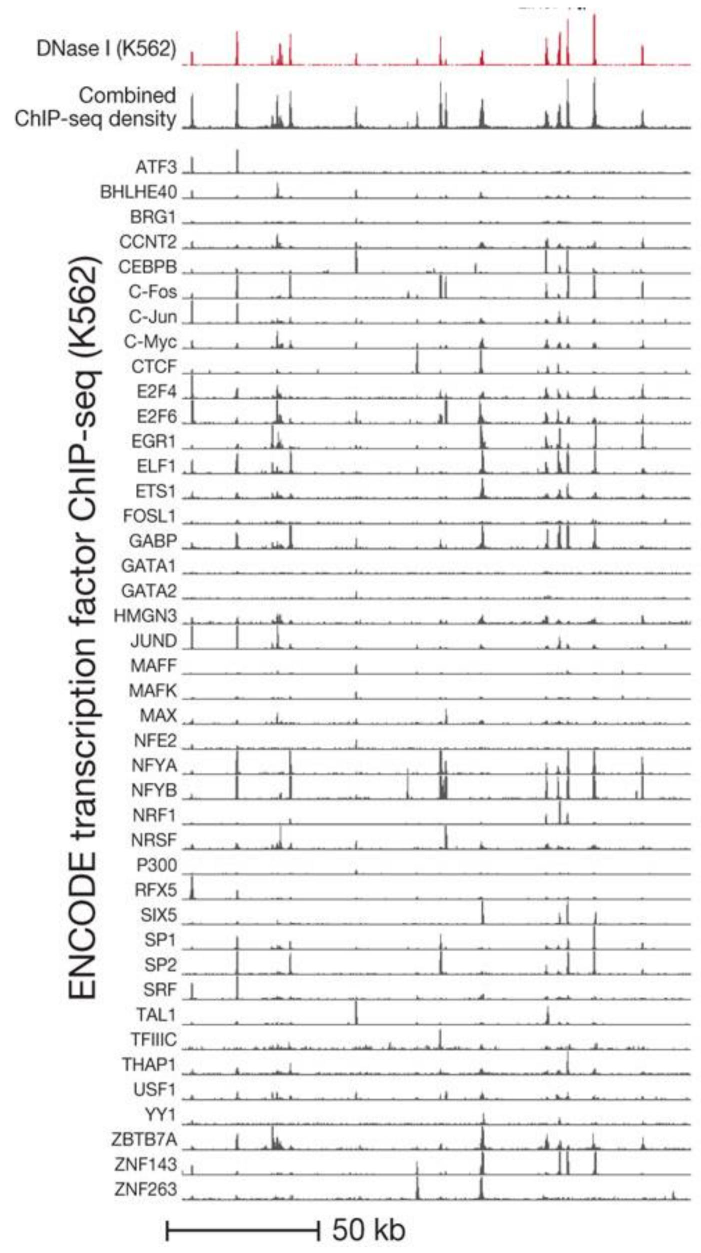
Read both ends and map to genome



ATAC-seq peaks identify accessible control elements

Buenrostro et al. (2013) Nature Methods.

Chromatin accessibility ~= sum (ChIP-seq for all DNA binding proteins)



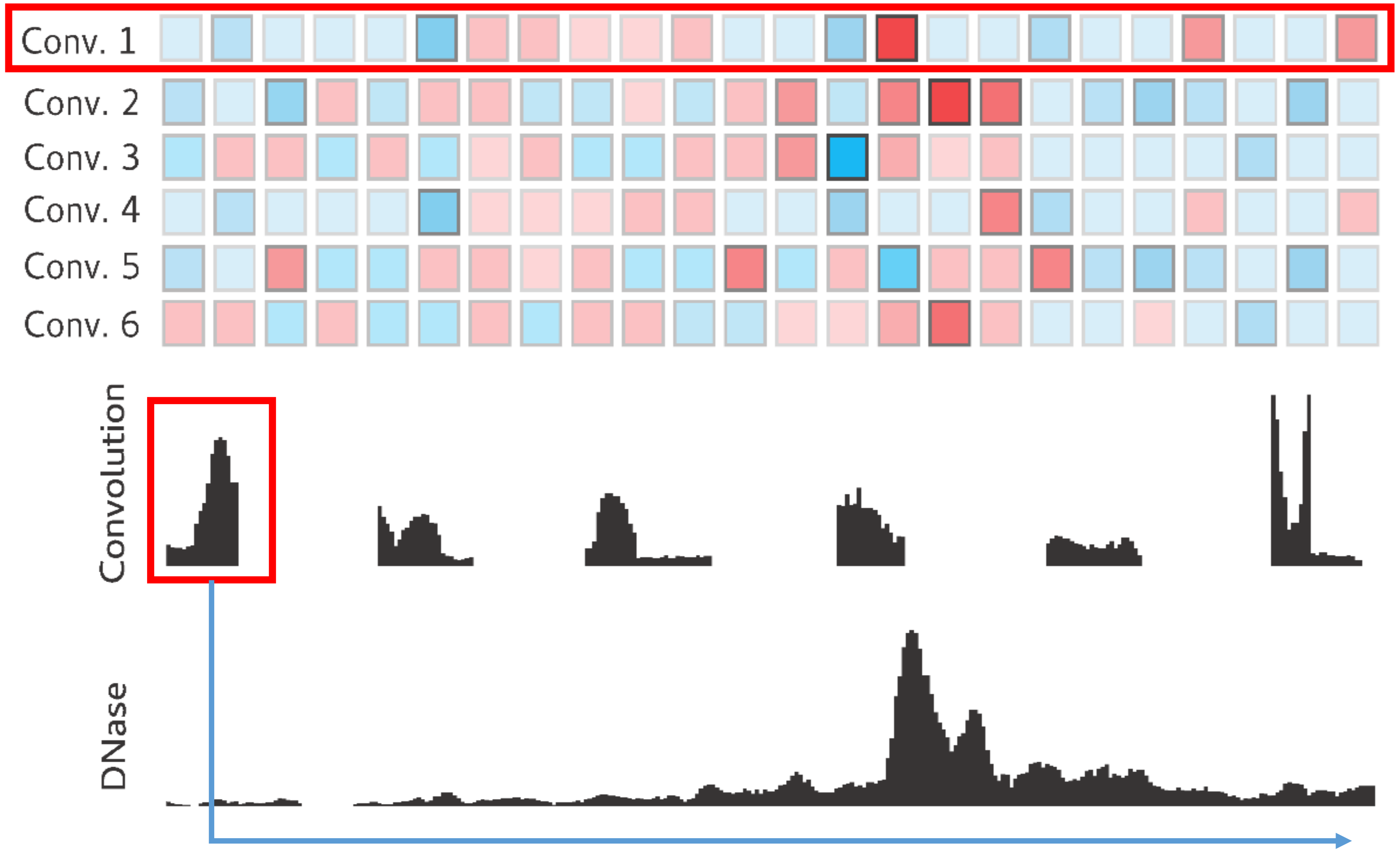
Peaks of chromatin accessibility signal at specific genomic locations tells us “something binds there”

BUT we don’t know who binds there.

Sequence patterns could tell us who binds there!

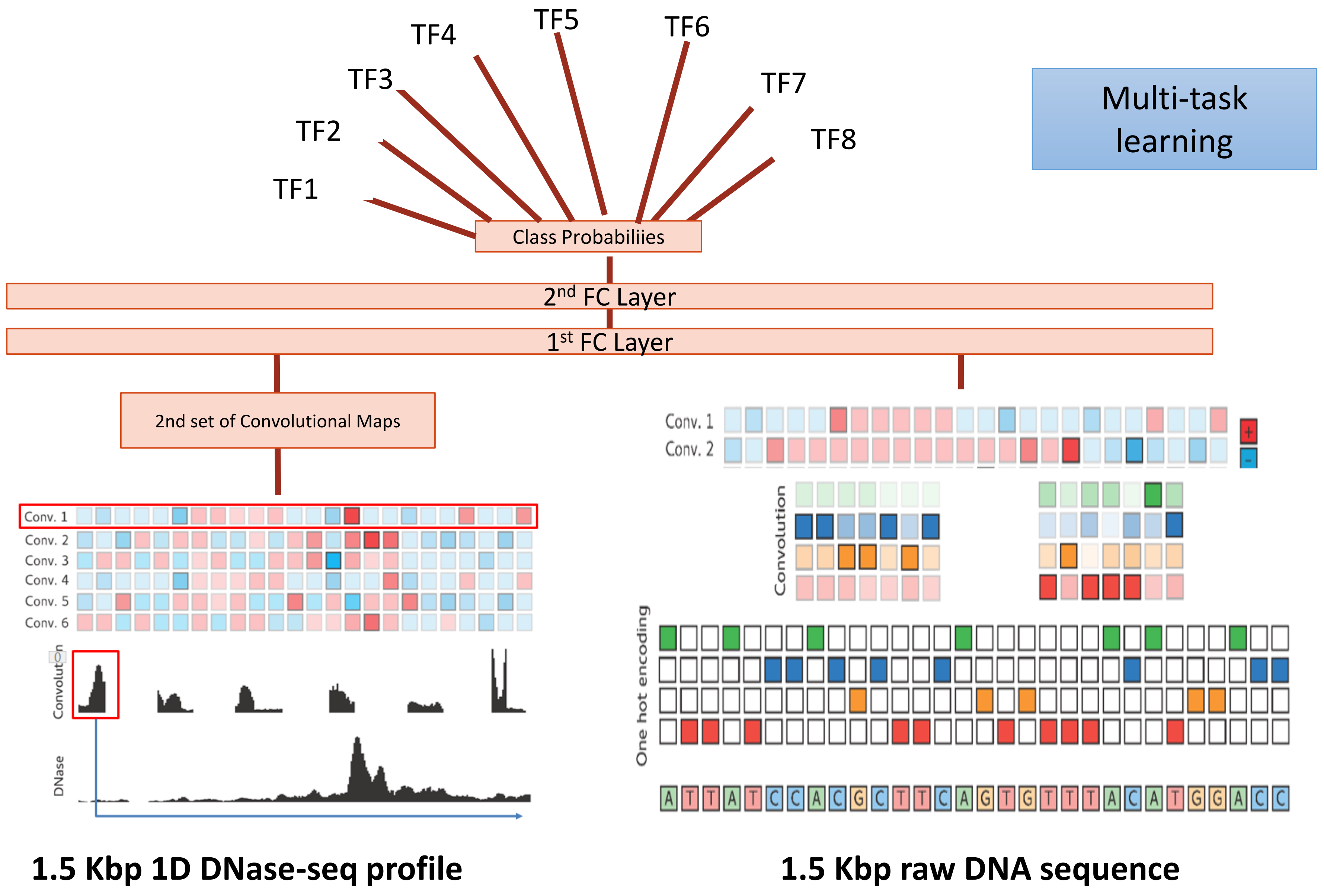
Integrate sequence + chromatin accessibility patterns to predict TF binding events (from ChIP-seq data)!

CNN filters for learning patterns from chromatin accessibility data



Scan chromatin accessibility profile using filter

Multi-modal integrative model



Multi-modal input

1.5 Kbp 1D DNase-seq profile

1.5 Kbp raw DNA sequence

日本語要約

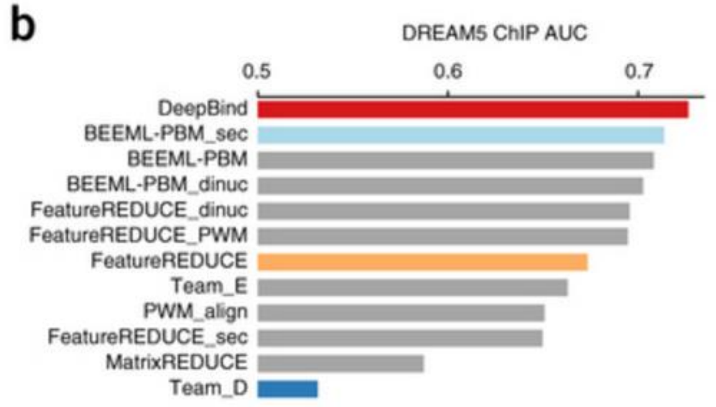
Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning

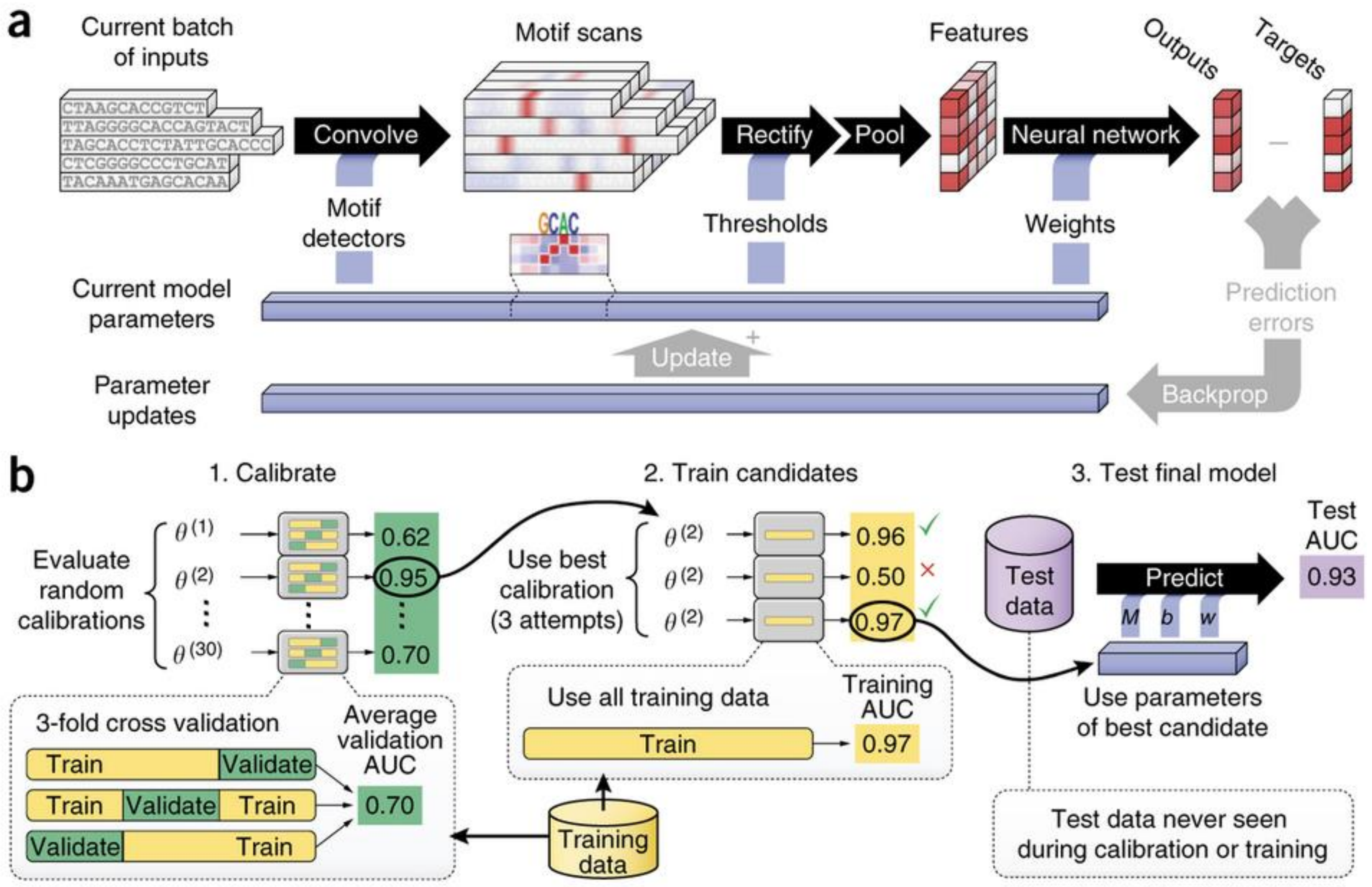
Babak Alipanahi, Andrew Delong, Matthew T Weirauch & Brendan J Frey

Affiliations | Contributions | Corresponding author

Nature Biotechnology **33**, 831–838 (2015) | doi:10.1038/nbt.3300
Received 28 November 2014 | Accepted 25 June 2015 | Published online 27 July 2015

b





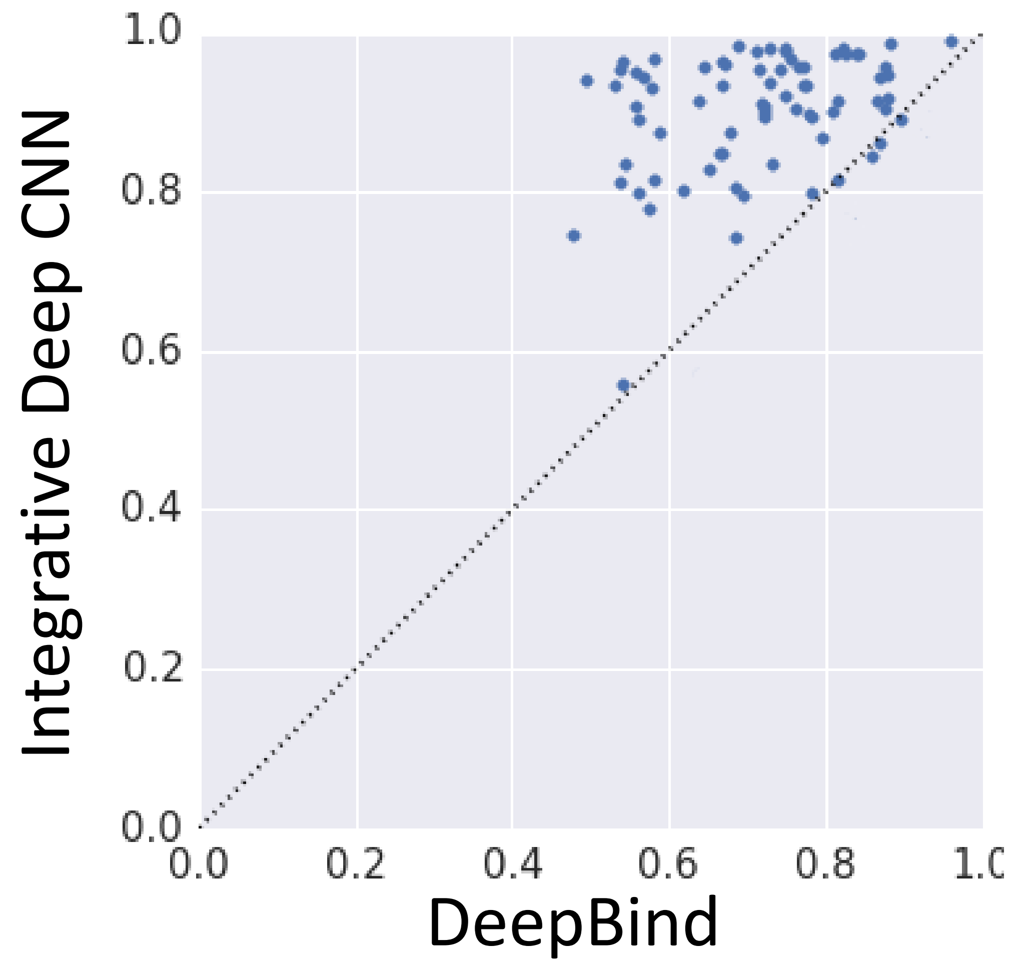
http://www.nature.com/nbt/journal/v33/n8/full/nbt.3300.html

Performance evaluation

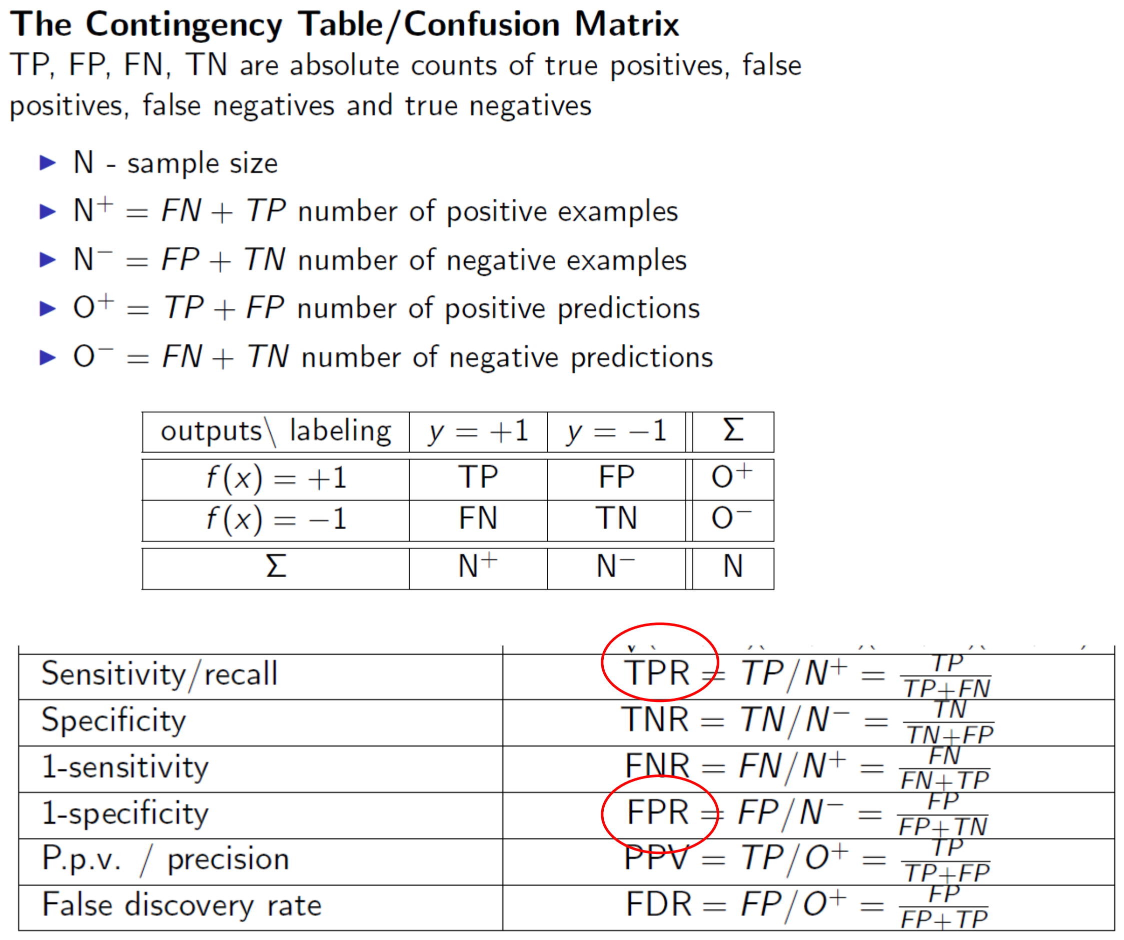
- <u>auROCs look great!</u>
- <u>Hurray! Looks like we don't need ChIP-seq data any more!</u>

Area under Receiver Operating Curve (auROC)

40 diverse TFs



(Alipanahi et al. 2015)



The Contingency Table/Confusion Matrix

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

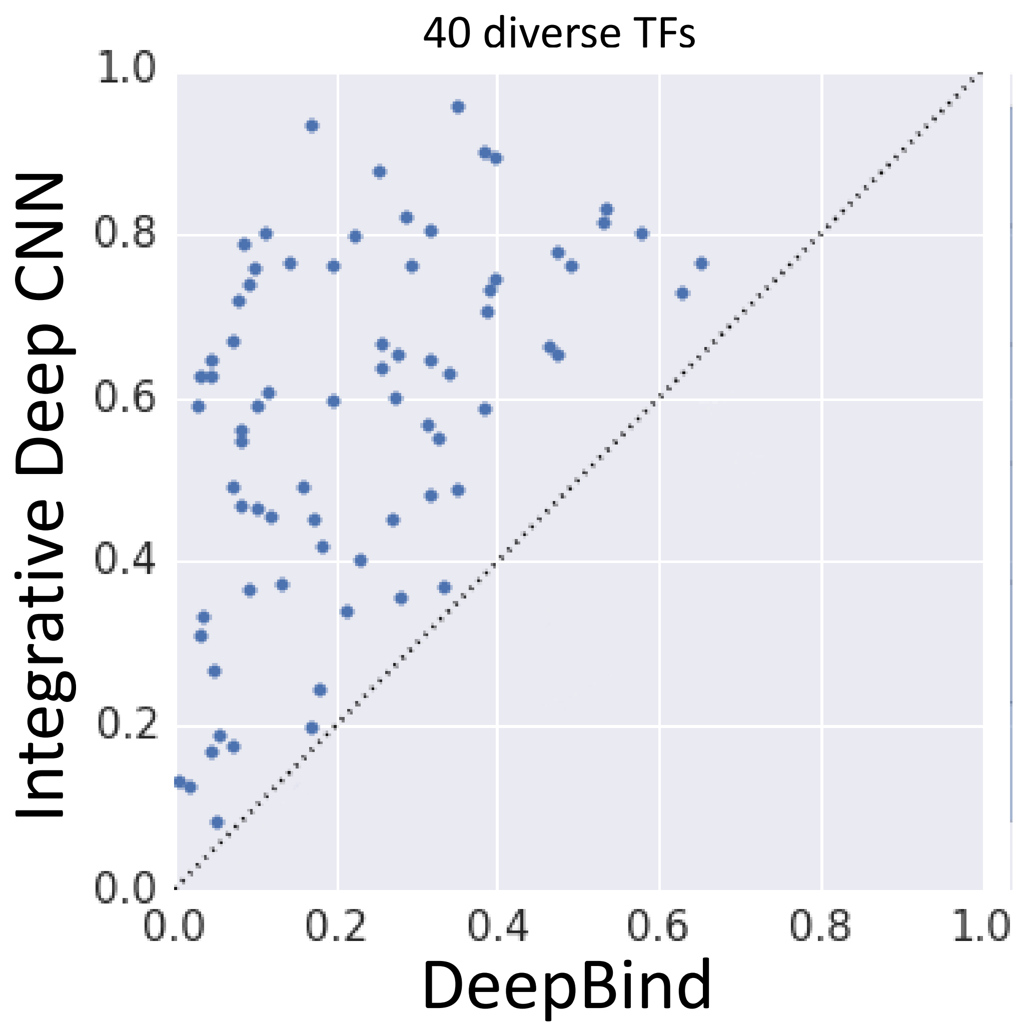
- N - sample size
- $N^+ = FN + TP$ number of positive examples
- $N^- = FP + TN$ number of negative examples
- $O^+ = TP + FP$ number of positive predictions
- $O^- = FN + TN$ number of negative predictions

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

Sensitivity/recall	$TPR = TP/N^+ = \frac{TP}{TP+FN}$
Specificity	$TNR = TN/N^- = \frac{TN}{TN+FP}$
1-sensitivity	$FNR = FN/N^+ = \frac{FN}{FN+TP}$
1-specificity	$FPR = FP/N^- = \frac{FP}{FP+TN}$
P.p.v. / precision	$PPV = TP/O^+ = \frac{TP}{TP+FP}$
False discovery rate	$FDR = FP/O^+ = \frac{FP}{FP+TP}$

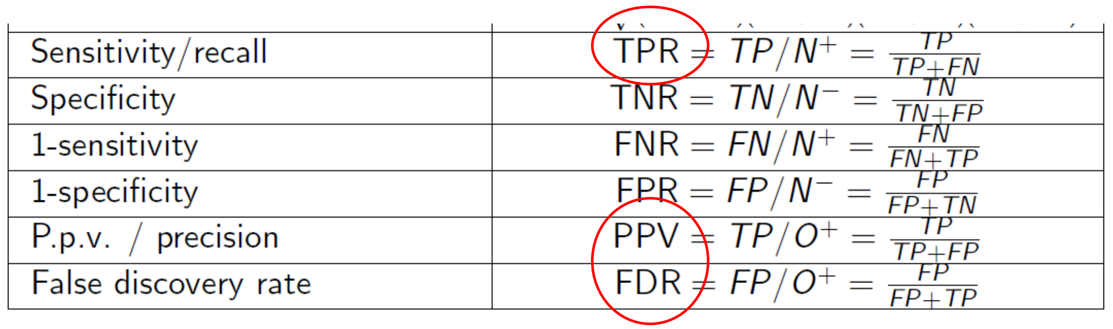
Performance evaluation metric matters!

Area under Precision-Recall Curve

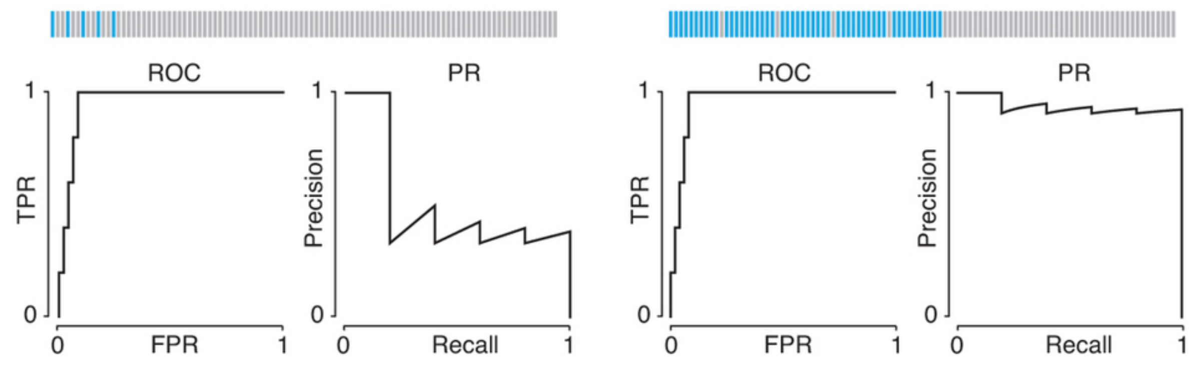


(Alipanahi et al. 2015)

- Prediction task is highly unbalanced (50-100x more negatives than positives)
- **auROC is highly misleading for unbalanced data!**



Sensitivity/recall	$\text{TPR} = TP/N^+ = \frac{TP}{TP+FN}$
Specificity	$\text{TNR} = TN/N^- = \frac{TN}{TN+FP}$
1-sensitivity	$\text{FNR} = FN/N^+ = \frac{FN}{FN+TP}$
1-specificity	$\text{FPR} = FP/N^- = \frac{FP}{FP+TN}$
P.p.v. / precision	$\text{PPV} = TP/O^+ = \frac{TP}{TP+FP}$
False discovery rate	$\text{FDR} = FP/O^+ = \frac{FP}{FP+TP}$

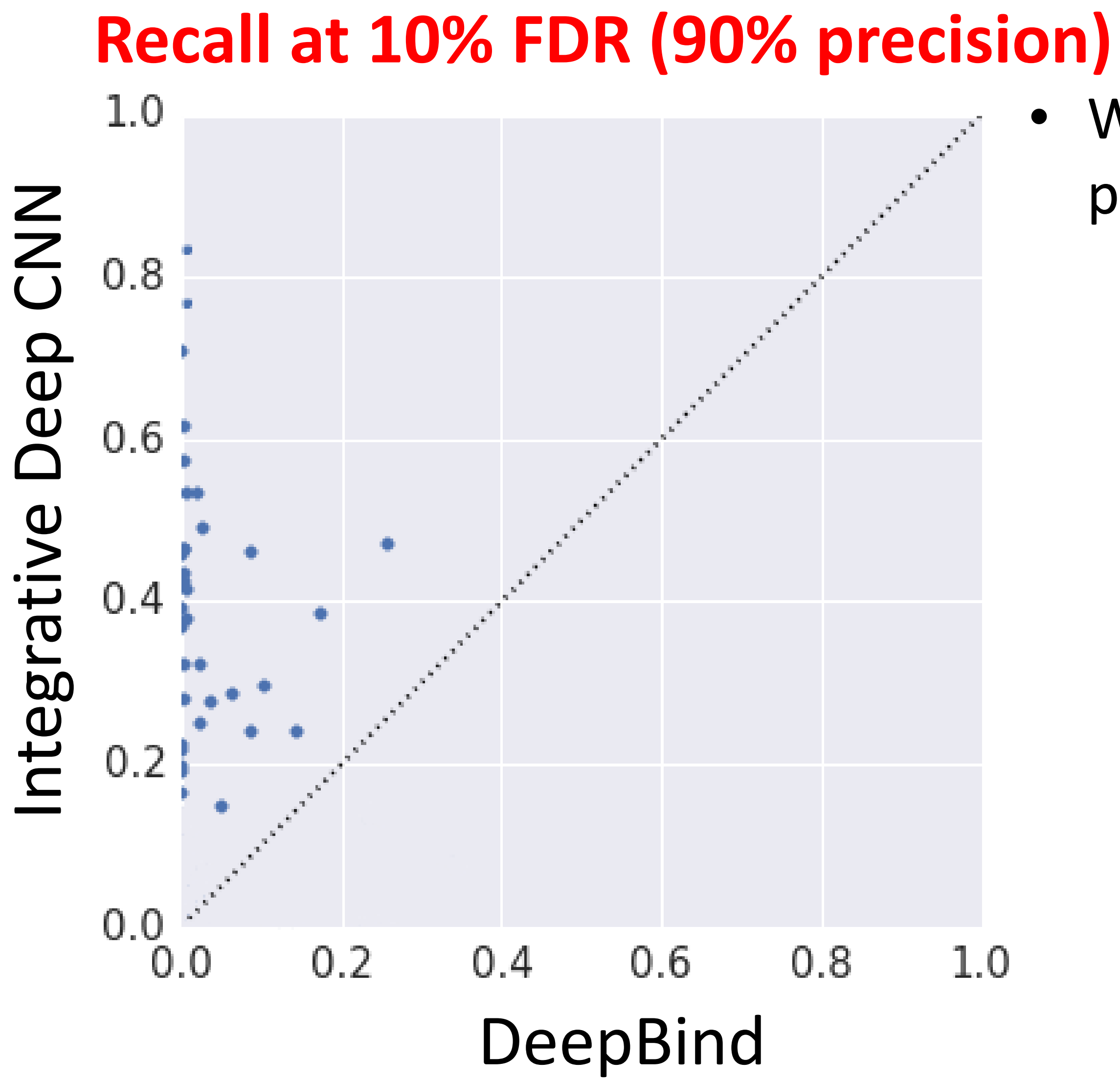


(a,b) ROC and PR curves for two data sets with very different class balances: (a) 5% positive and (b) 50% positive observations. For each panel, observations are shown as vertical lines (top), of which 5% or 50% are positive (blue).

http://www.nature.com/nmeth/journal/v13/n8/full/nmeth.3945.html

FYI: auPRC implementation in scikit-learn is wrong!

Negative set matters!



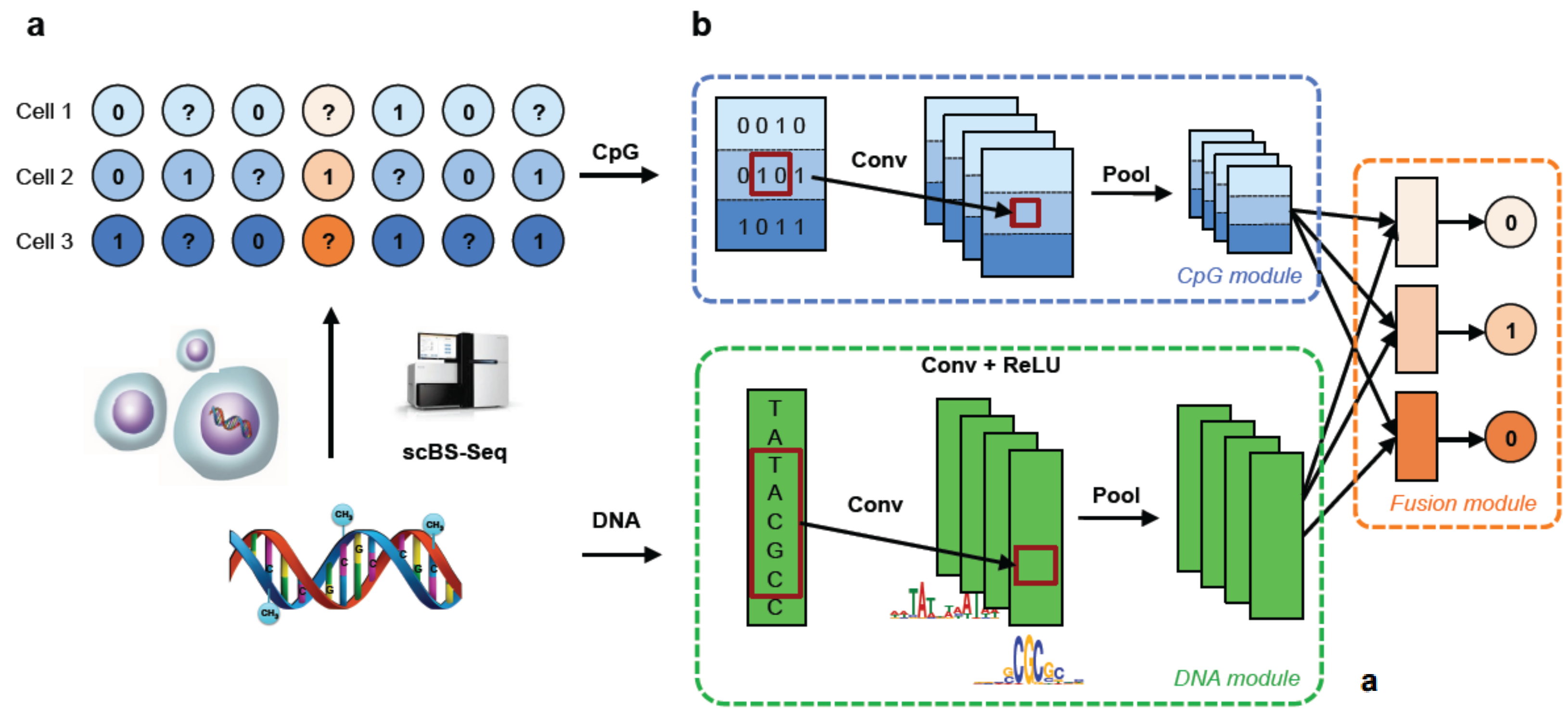
- Why does DeepBind do so poorly in this setting?
  - Trains on dinucleotide shuffled negatives (not representative of relevant genomic background)
  - **<u>Negative set matters!</u>**

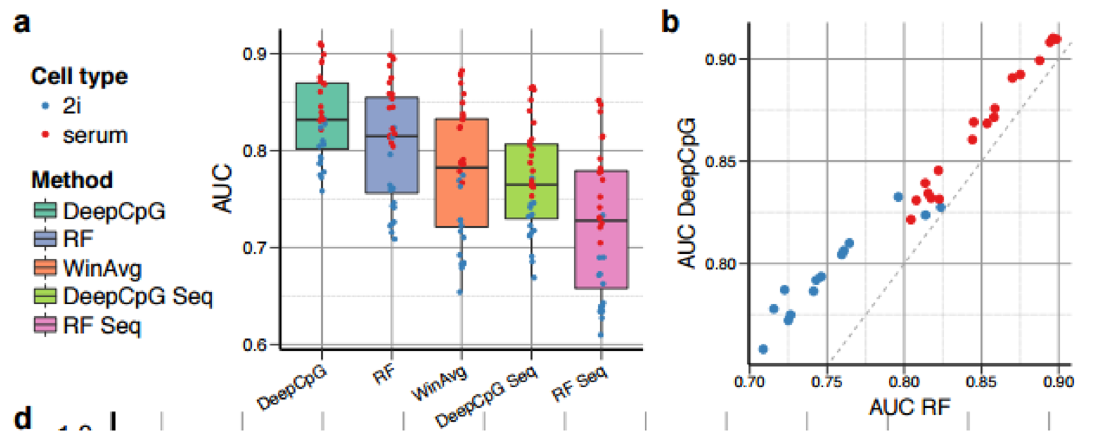
Accurate prediction of single-cell DNA methylation states using deep learning

Posted May 27, 2016.

Christof Angermueller, Heather Lee, Wolf Reik, Oliver Stegle

Download PDF



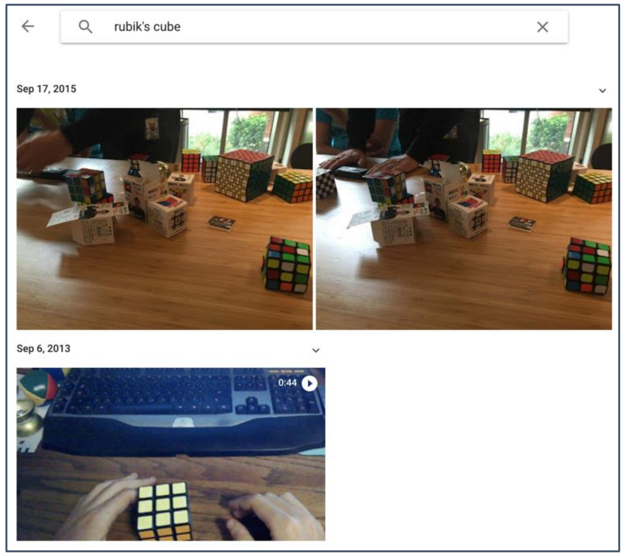


http://biorxiv.org/content/early/2016/05/27/055715

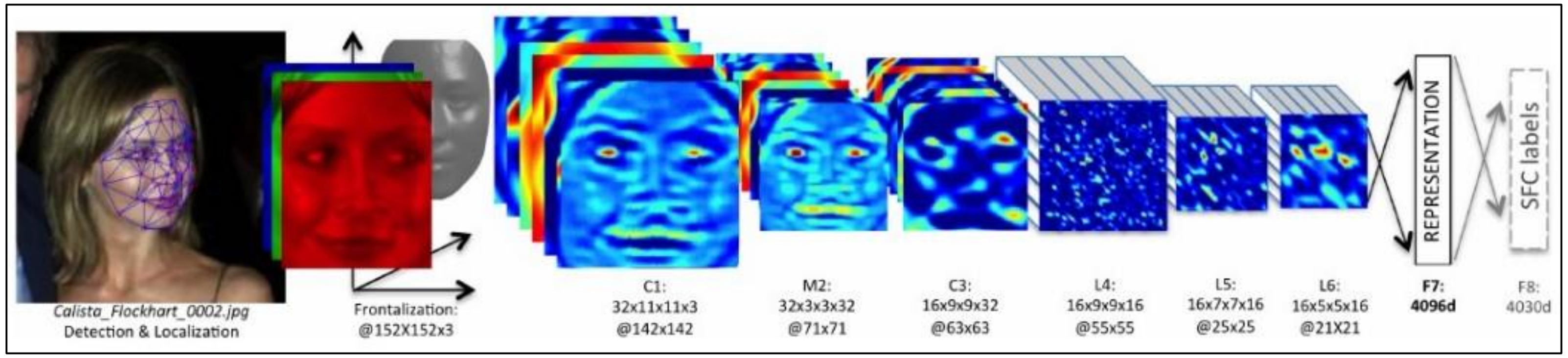
Convolutional neural networks for learning from imaging data

(Slides taken from Andrej Karpathy's CS231N lectures)

ConvNets are everywhere…



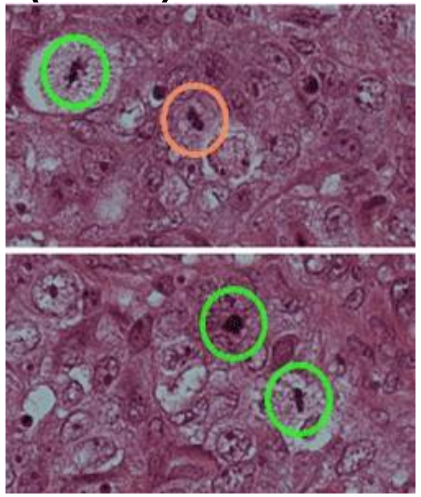
e.g. Google Photos search



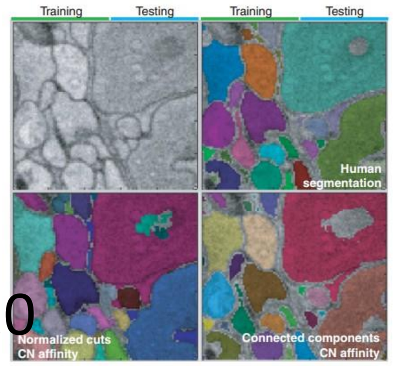
Face Verification, Taigman et al. 2014 (FAIR)

[Goodfellow et al. 2014]

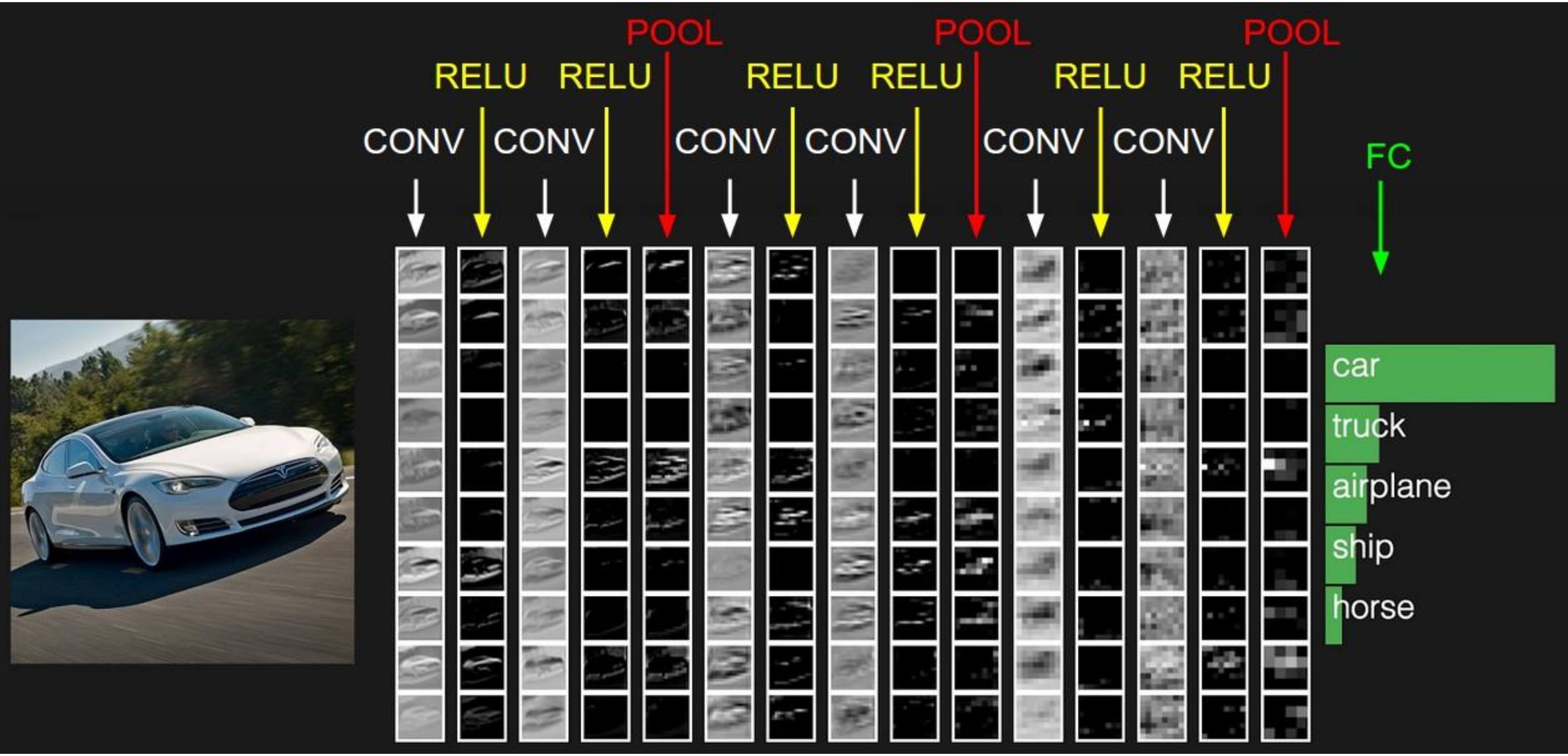
Self-driving cars

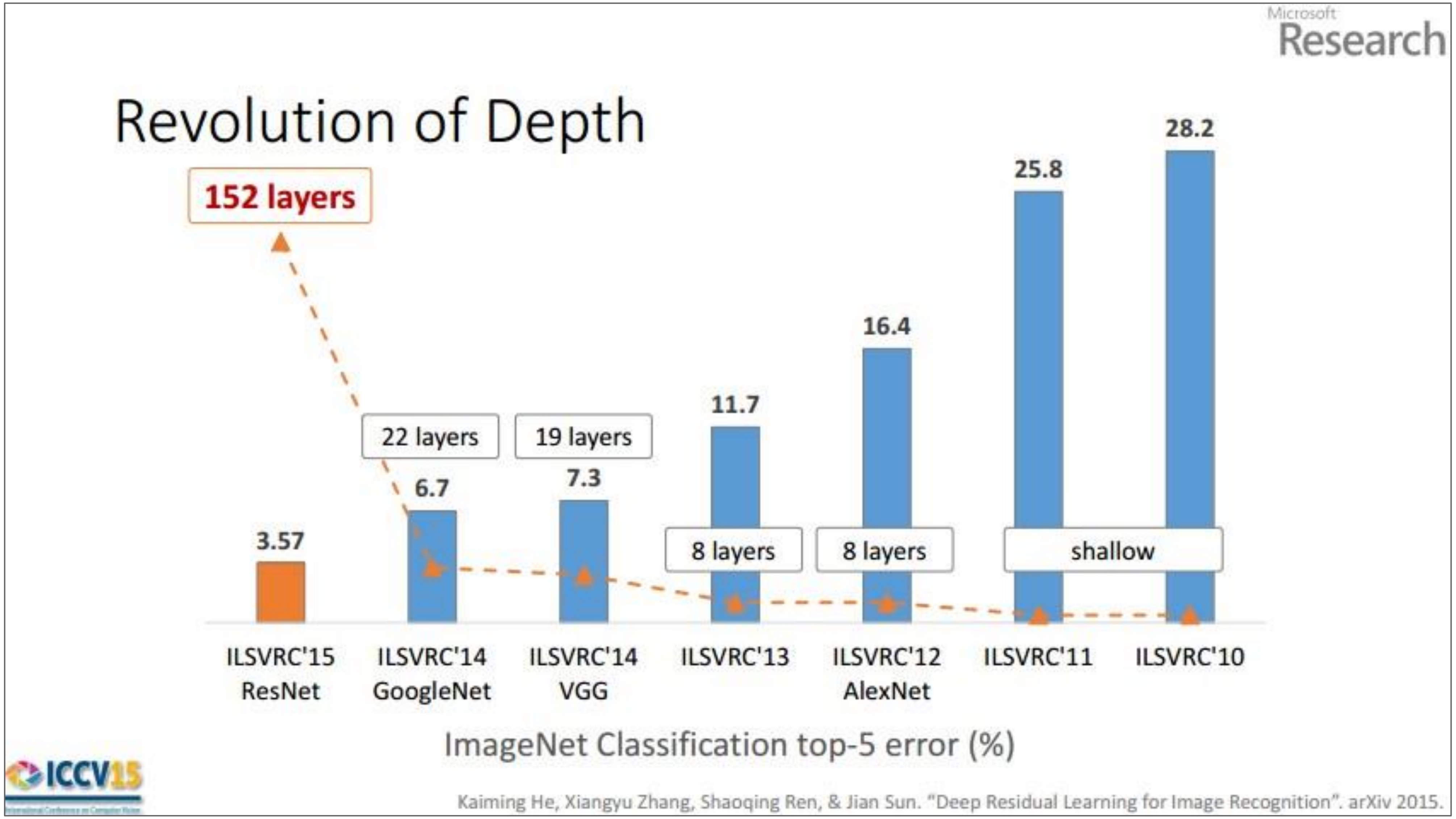


Ciresan et al. 201



Turaga et al 2010


POOL
POOL
POOL
RELU
RELU
RELU
RELU
RELU
RELU
CONV
CONV
CONV
CONV
CONV
CONV
FC
car
truck
airplane
ship
horse



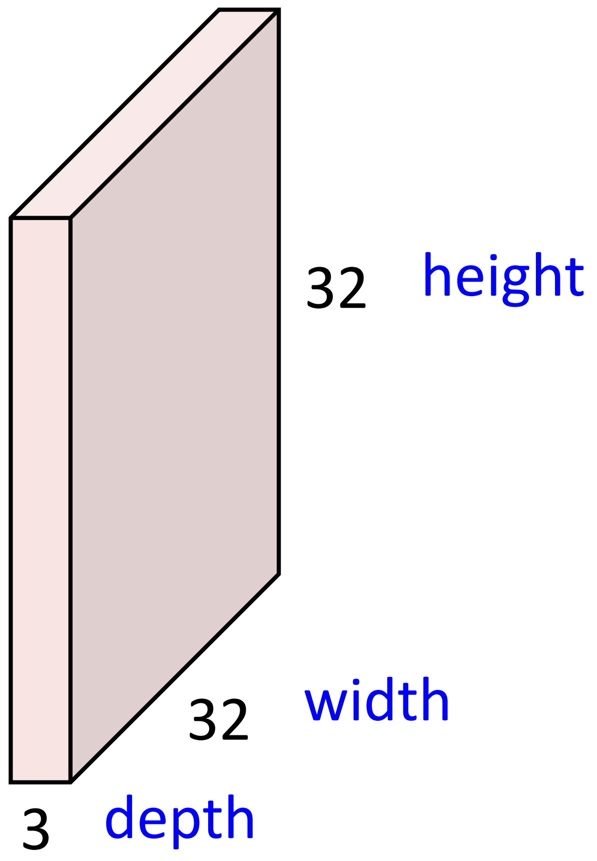
Revolution of Depth

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". arXiv 2015.

(slide from Kaiming He's recent presentation)

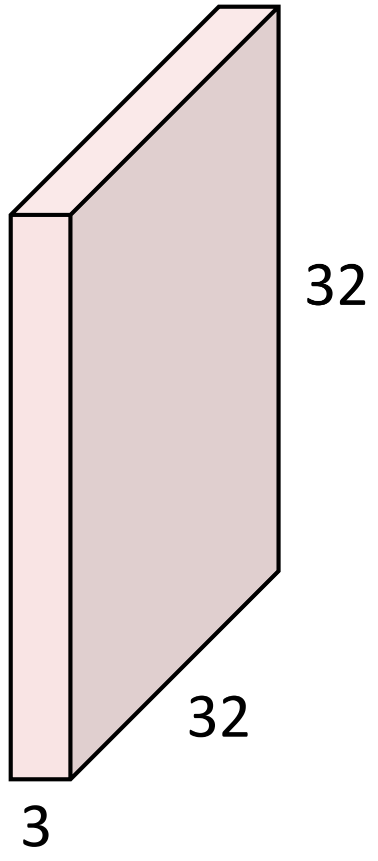
Convolution Layer

32x32x3 image

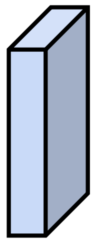


Convolution Layer

32x32x3 image



5x5x3 filter

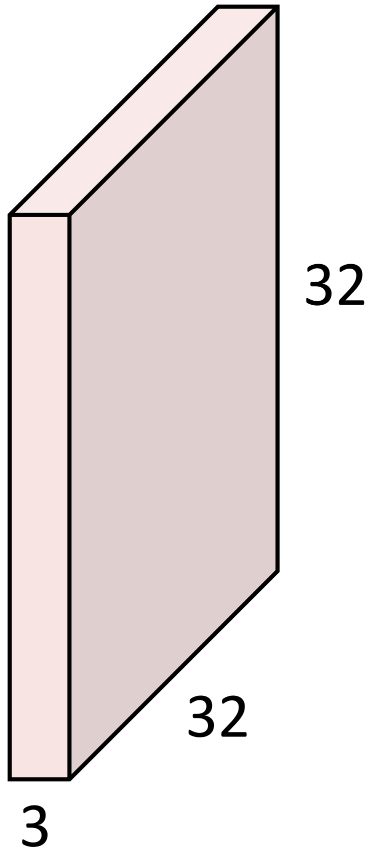


Convolve the filter with the image
i.e. “slide over the image spatially, computing dot products”

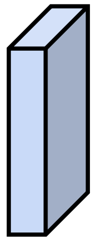
Convolution Layer

Filters always extend the full depth of the input volume

32x32x3 image

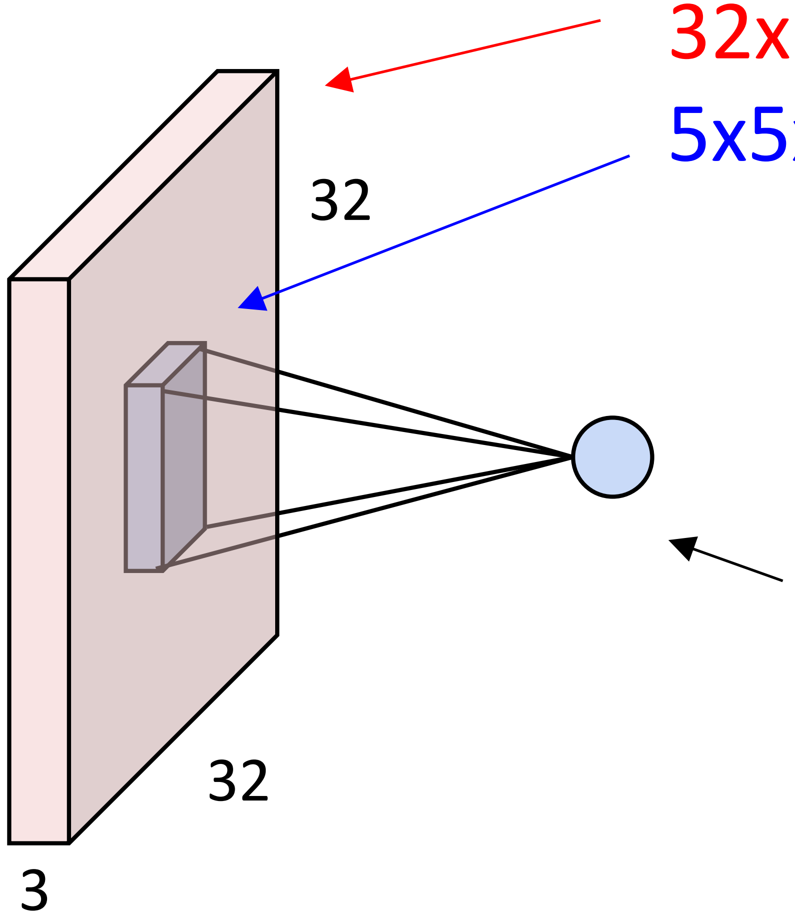


5x5x3 filter



Convolve the filter with the image i.e. "slide over the image spatially, computing dot products"

Convolution Layer



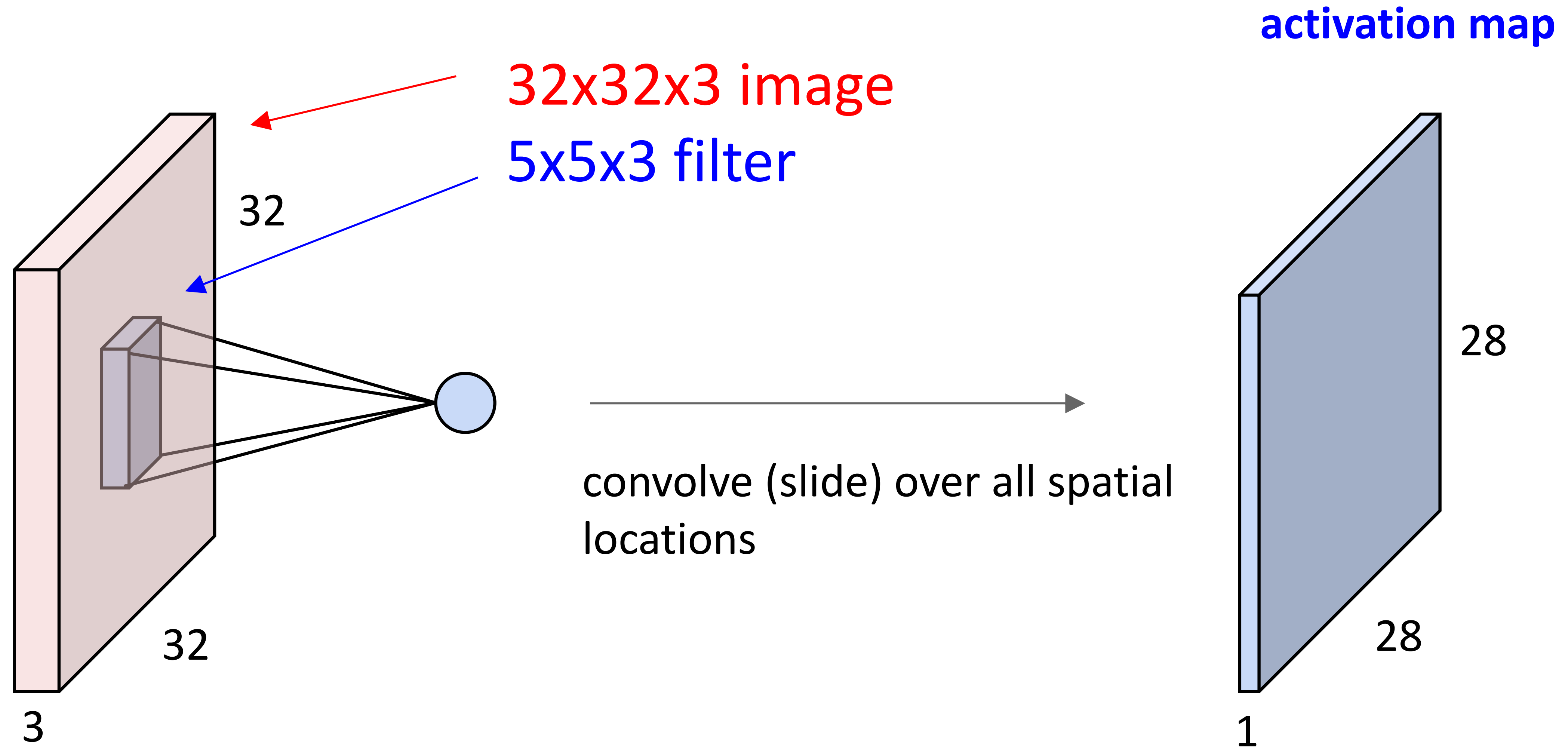
32x32x3 image

5x5x3 filter w

1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)

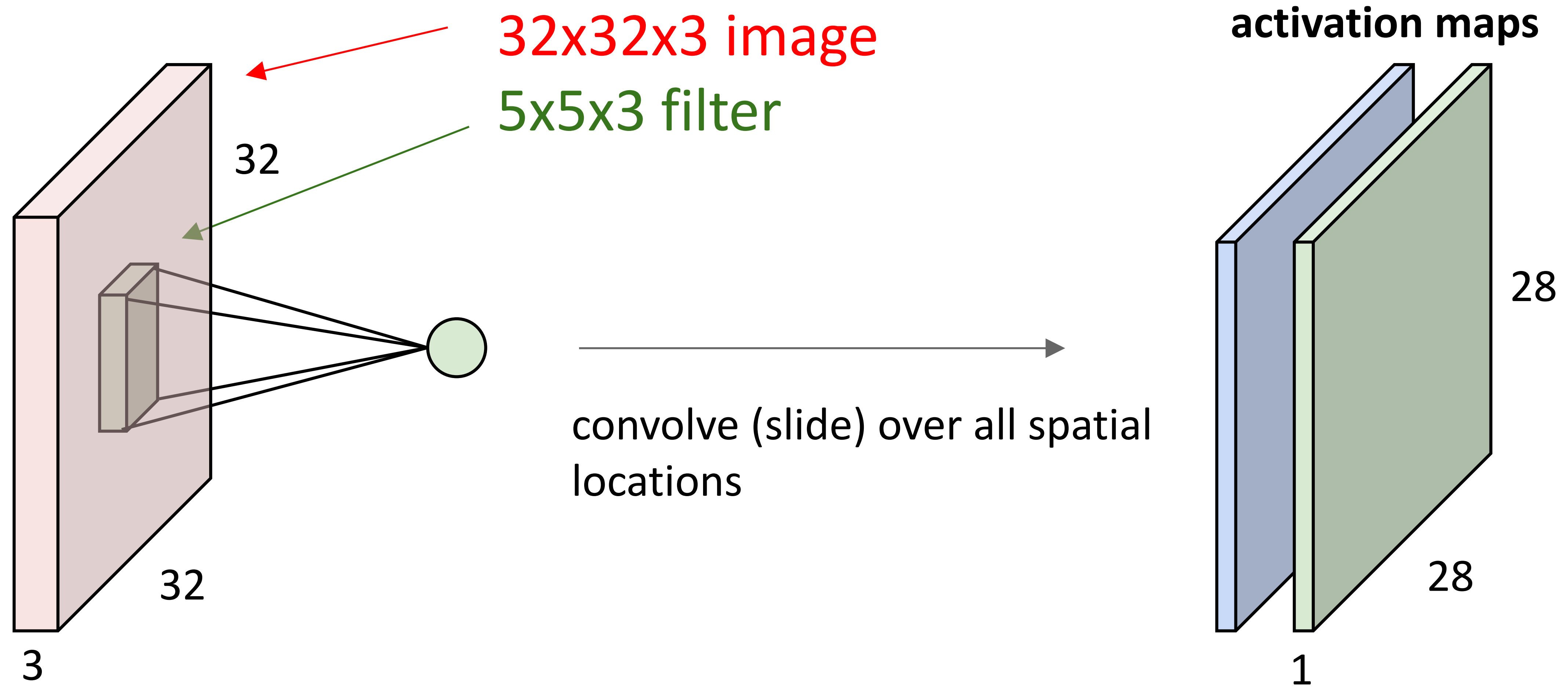
$$w^T x + b$$

Convolution Layer

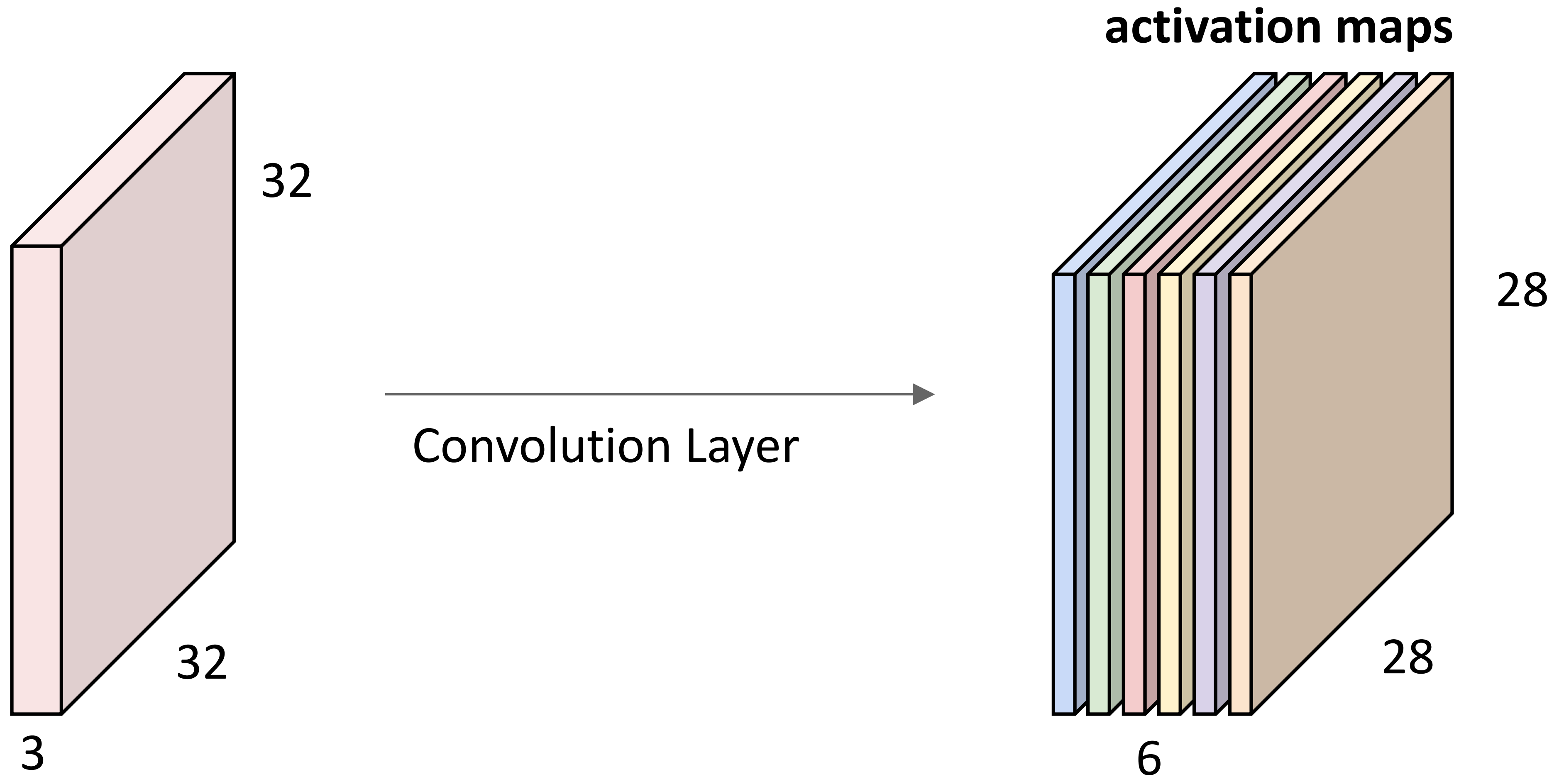


Convolution Layer

consider a second, green filter

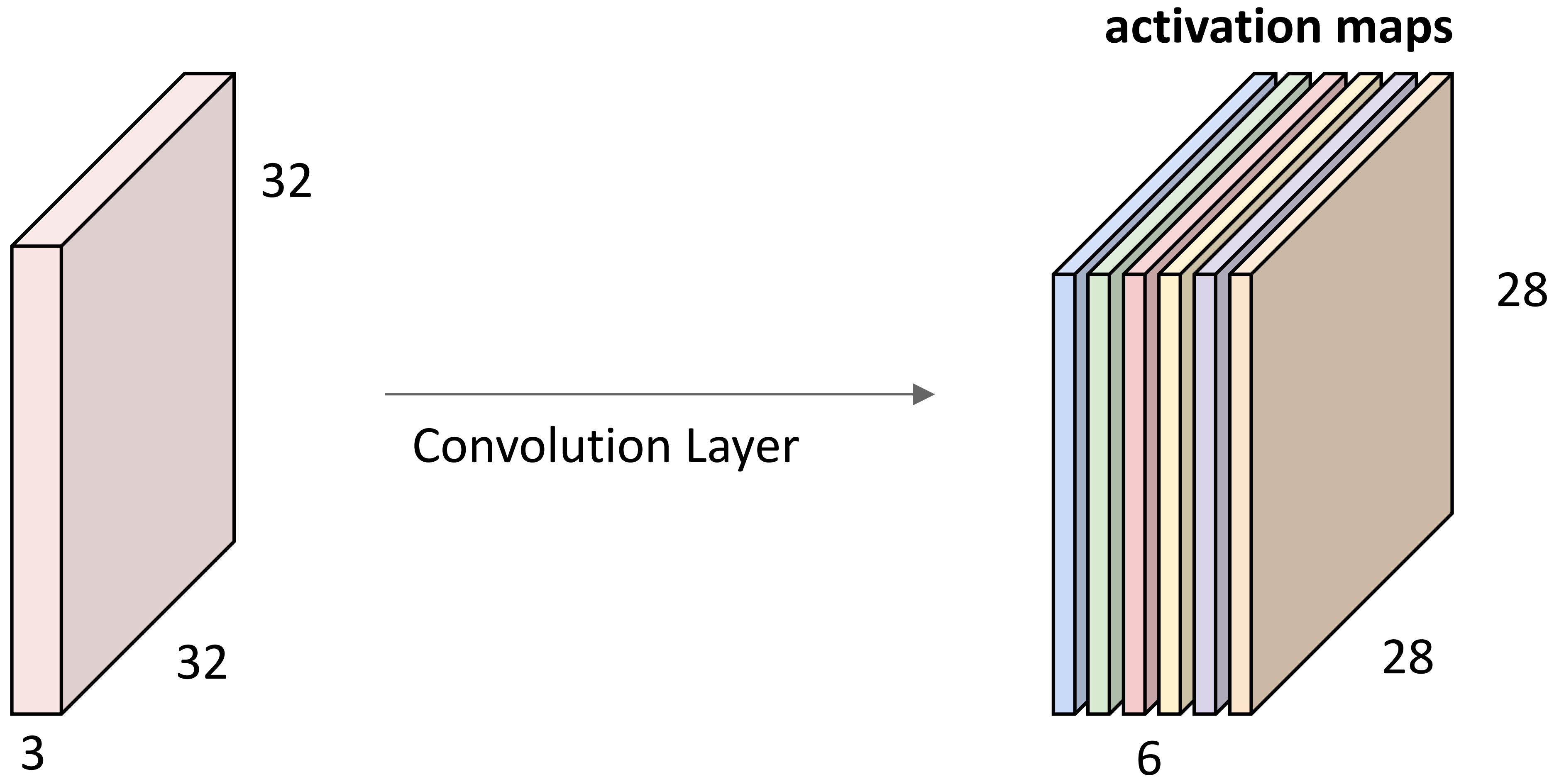


For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



We stack these up to get a "new image" of size 28x28x6!

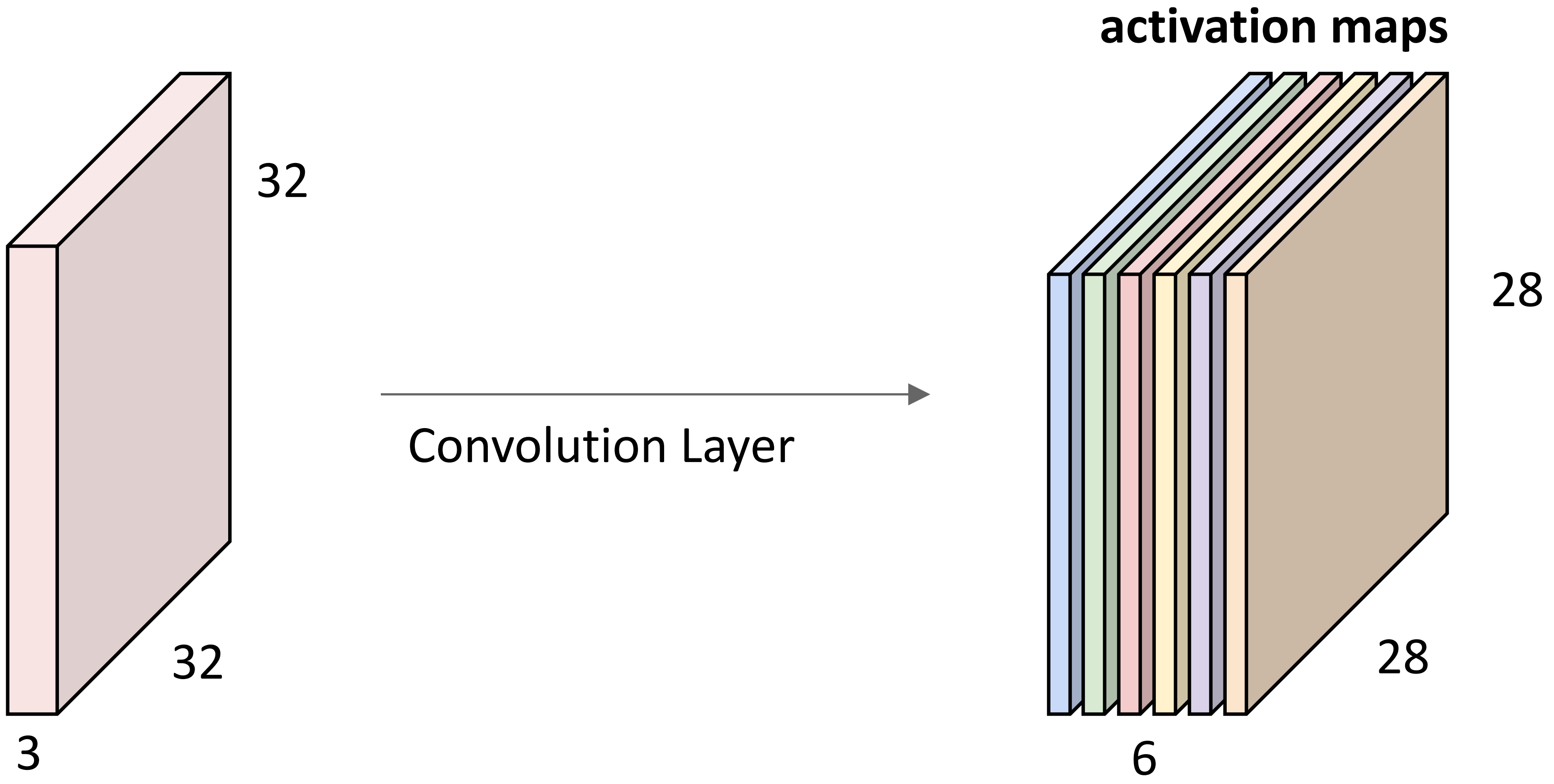
For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



We processed [32x32x3] volume into [28x28x6] volume.

Q: how many parameters would this be if we used a fully connected layer instead?

For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:

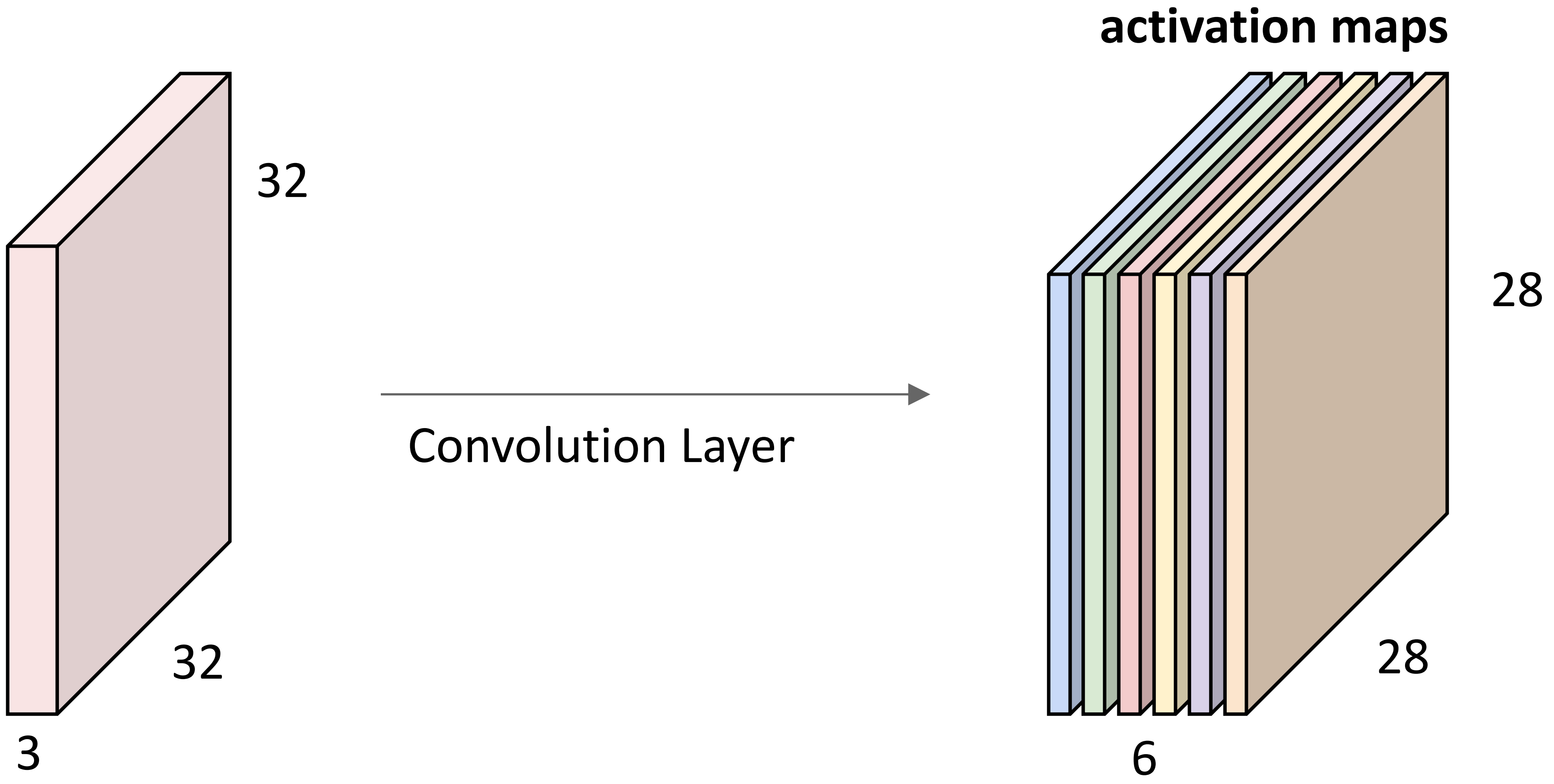


We processed [32x32x3] volume into [28x28x6] volume.

Q: how many parameters would this be if we used a fully connected layer instead?

A: (32*32*3)*(28*28*6) = **14.5M parameters, ~14.5M multiplies**

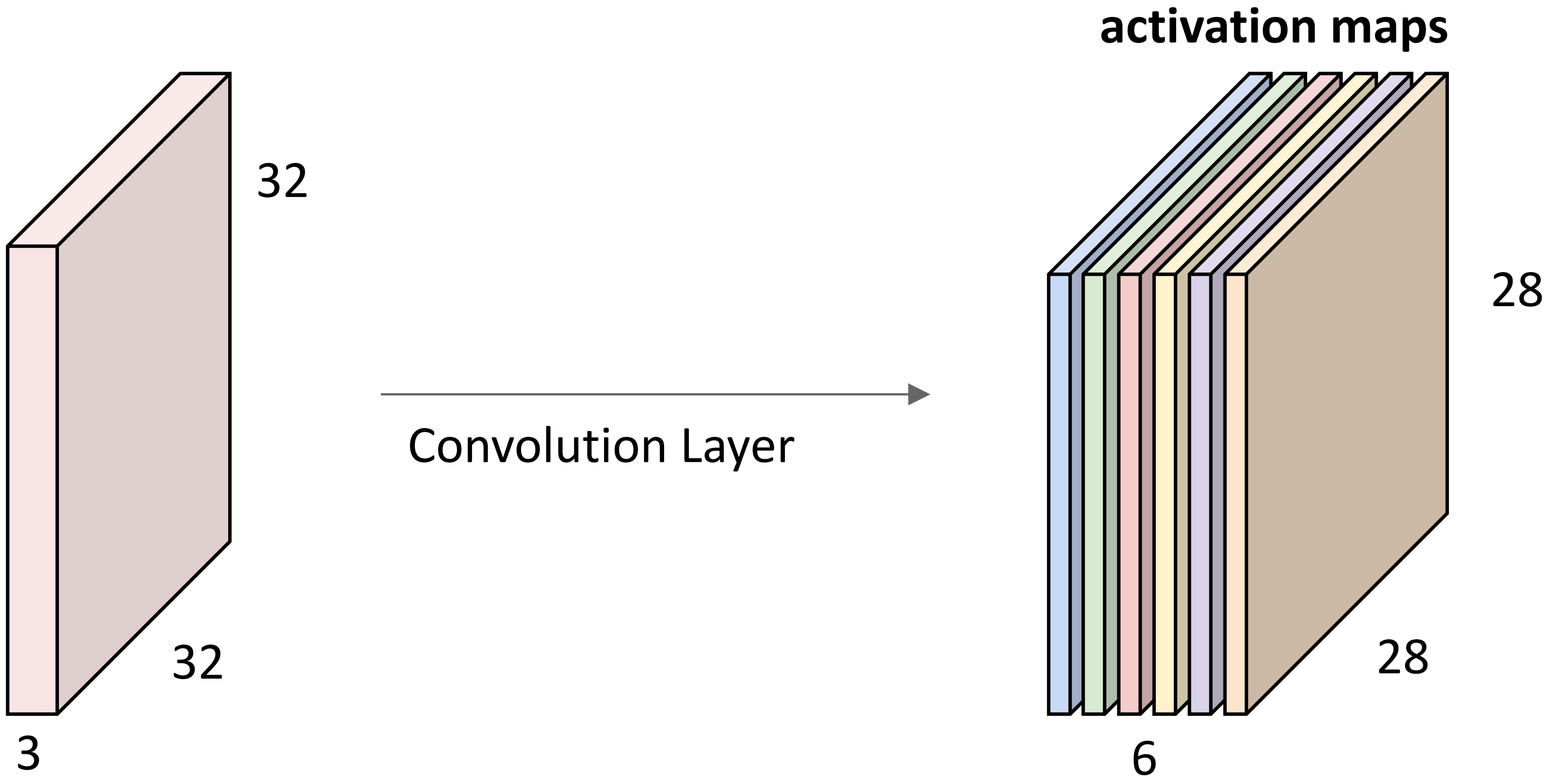
For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



We processed [32x32x3] volume into [28x28x6] volume.

Q: how many parameters are used instead?

For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:

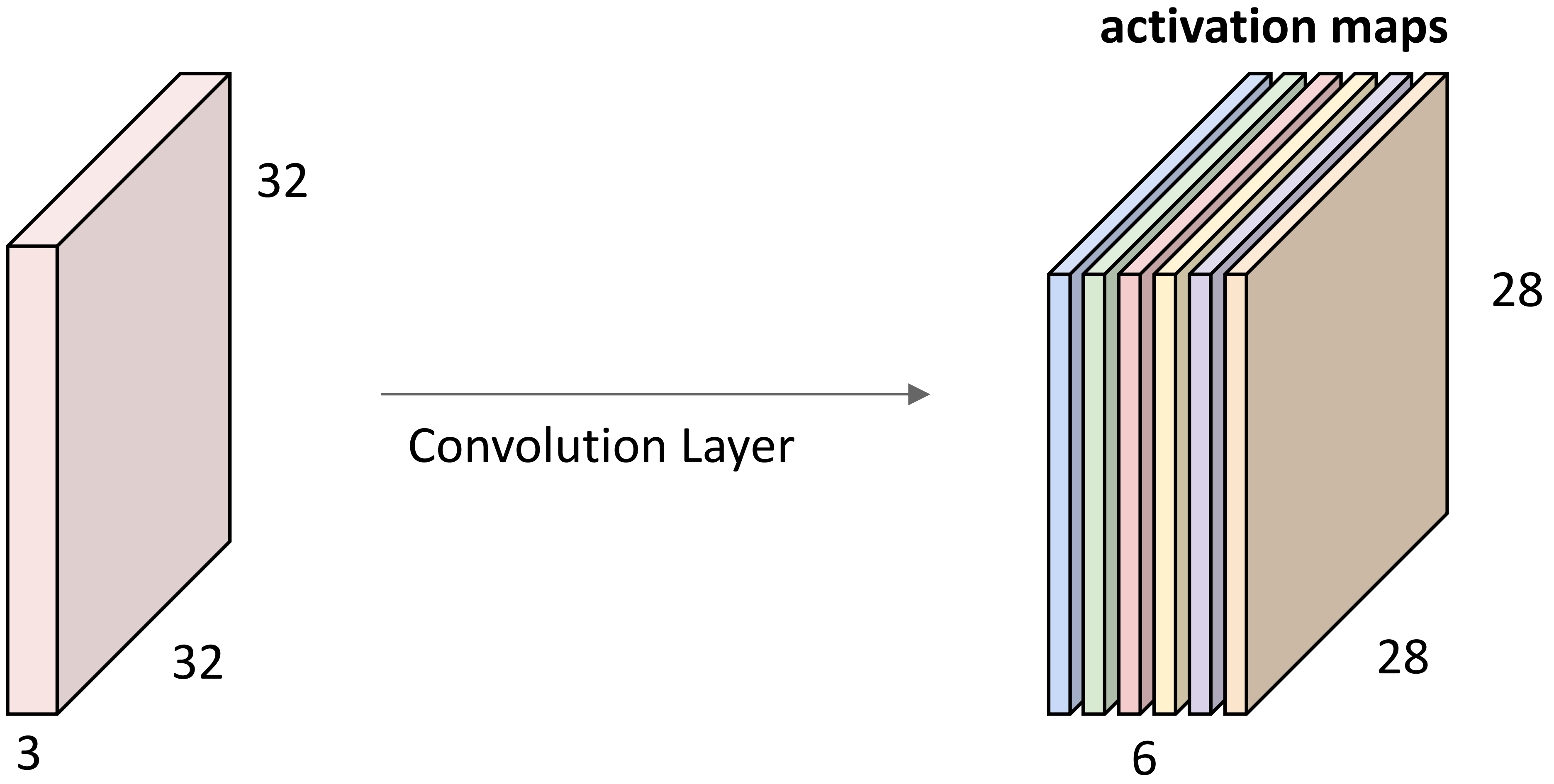


We processed [32x32x3] volume into [28x28x6] volume.

Q: how many parameters are used instead?   --- And how many multiplies?

A: (5*5*3)*6 = **450 parameters**

For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



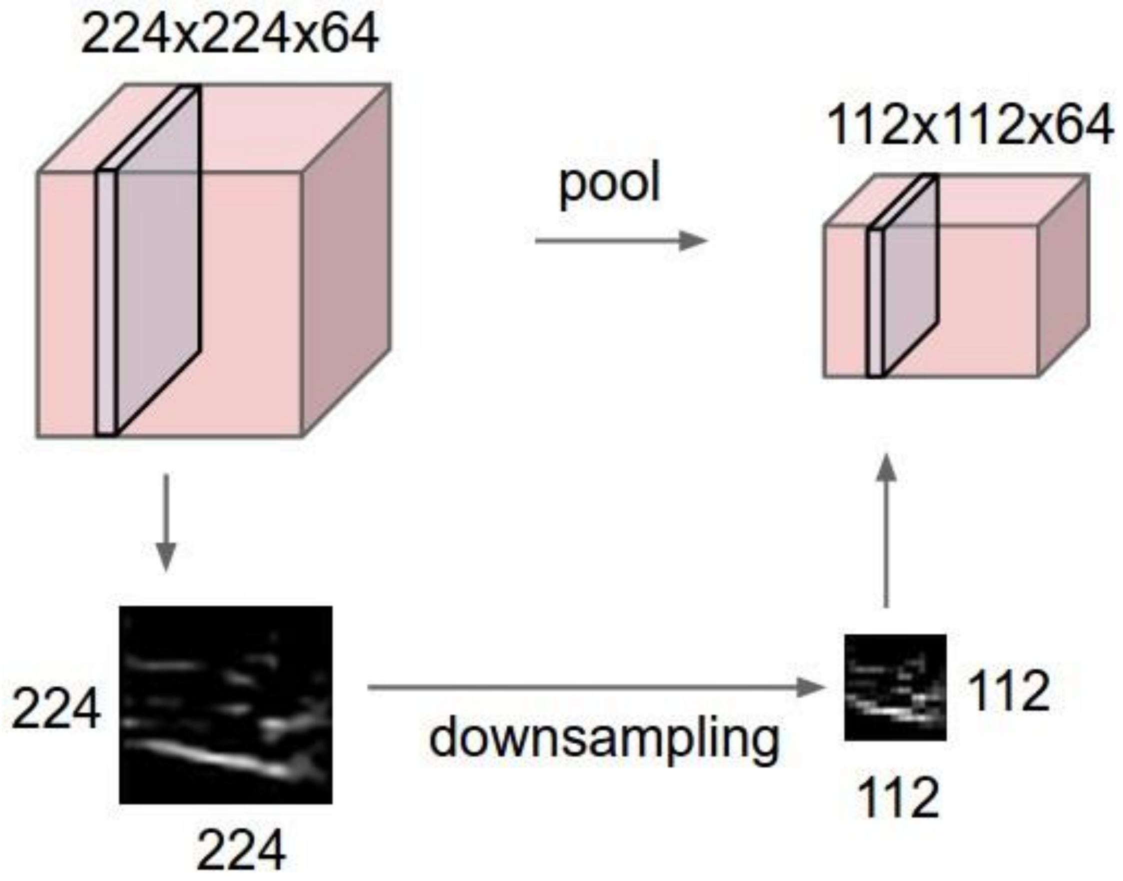
We processed [32x32x3] volume into [28x28x6] volume.

Q: how many parameters are used instead?

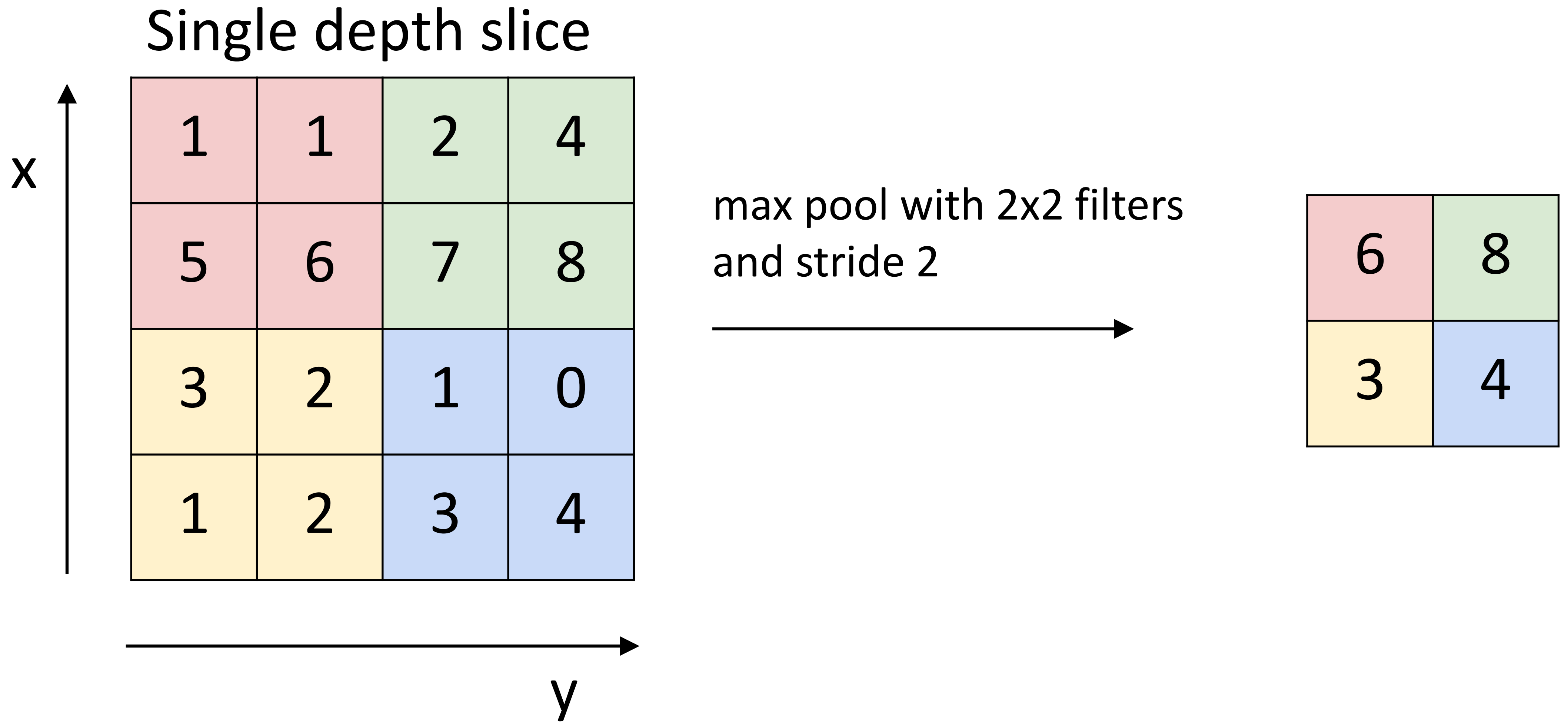
A: (5*5*3)*6 = **450 parameters**, (5*5*3)*(28*28*6) = **~350K multiplies**

Pooling layer

- makes the representations smaller and more manageable
- operates over each activation map independently:



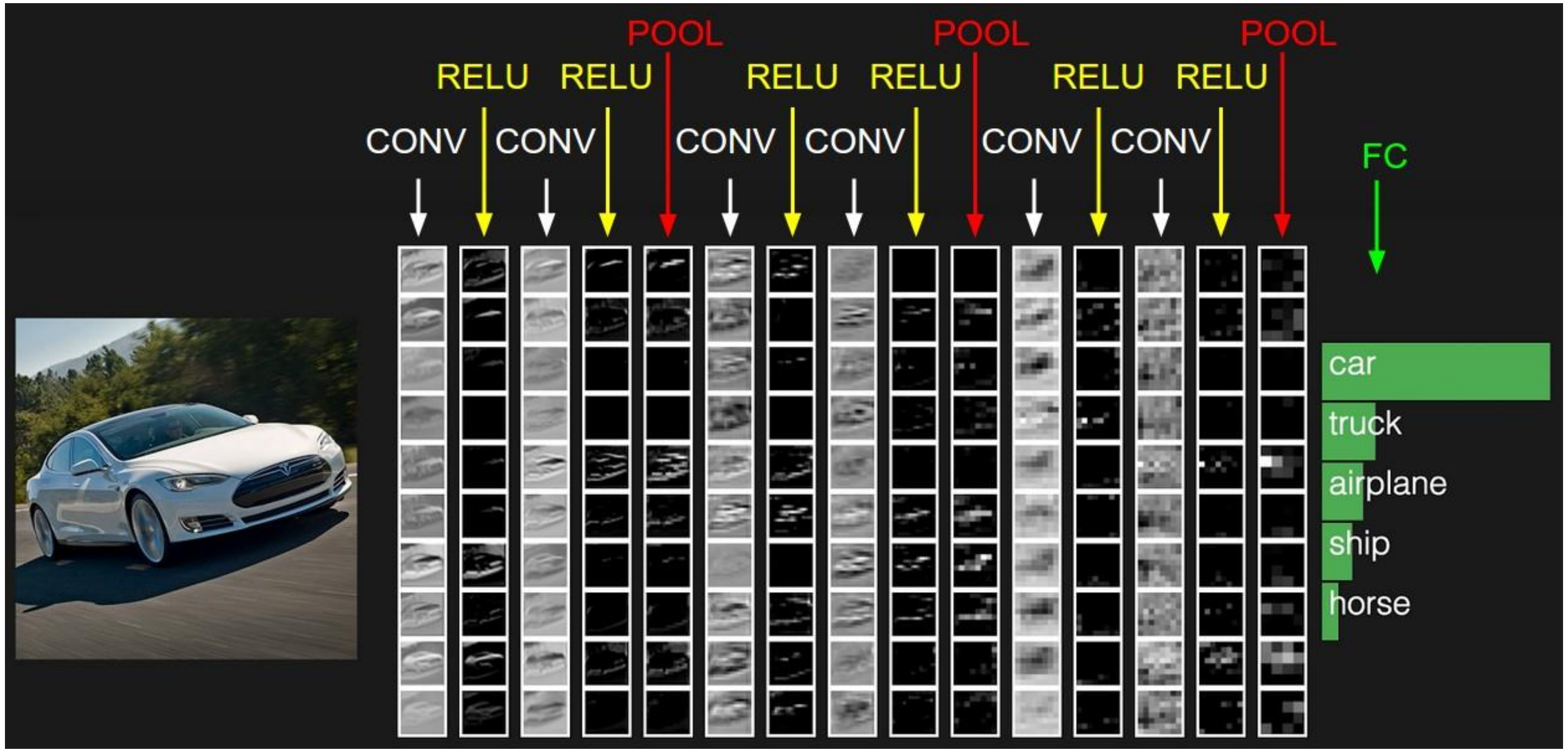
MAX POOLING



Single depth slice
x
y
1
1
2
4
5
6
7
8
3
2
1
0
1
2
3
4
max pool with 2x2 filters
and stride 2
6
8
3
4

Fully Connected Layer (FC layer)

- Contains neurons that connect to the entire input volume, as in ordinary Neural Networks



ConvNetJS demo: training on CIFAR-10

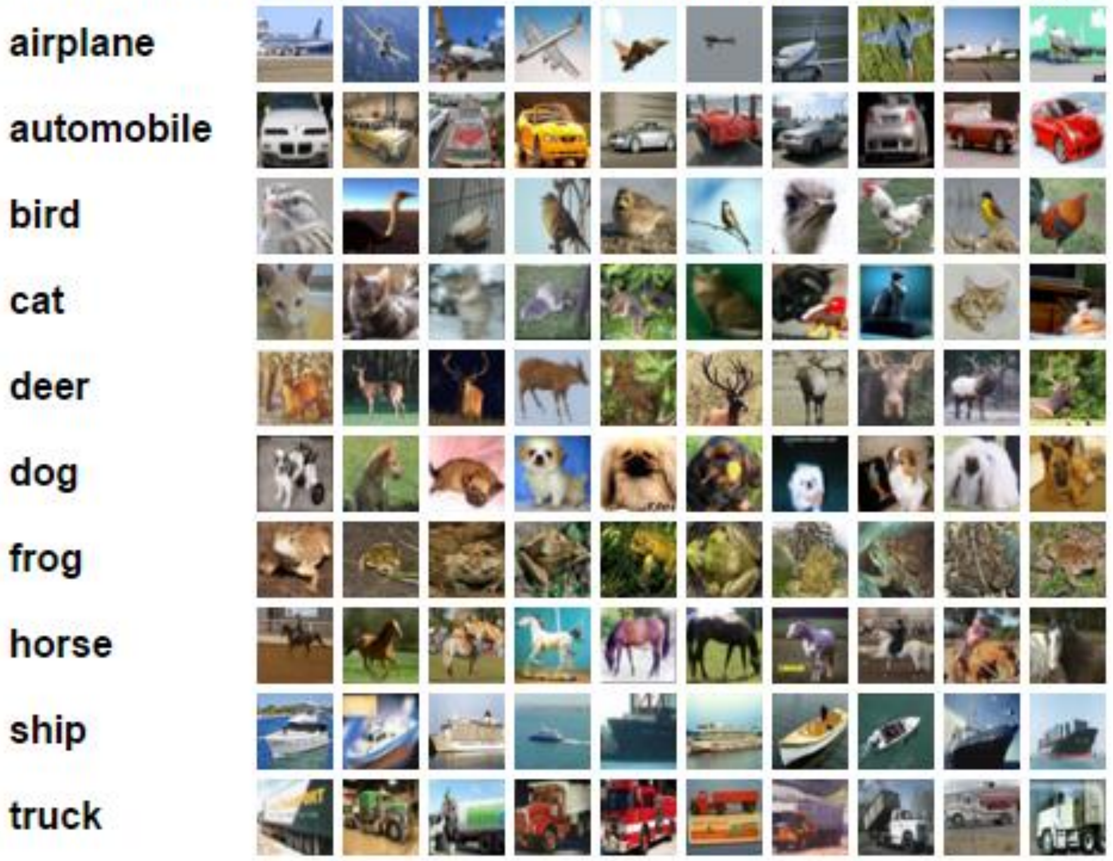
http://cs.stanford.edu/people/karpathy/convnetjs/demo/cifar10.html

The CIFAR-10 dataset

The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

The dataset is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class.

Here are the classes in the dataset, as well as 10 random images from each:



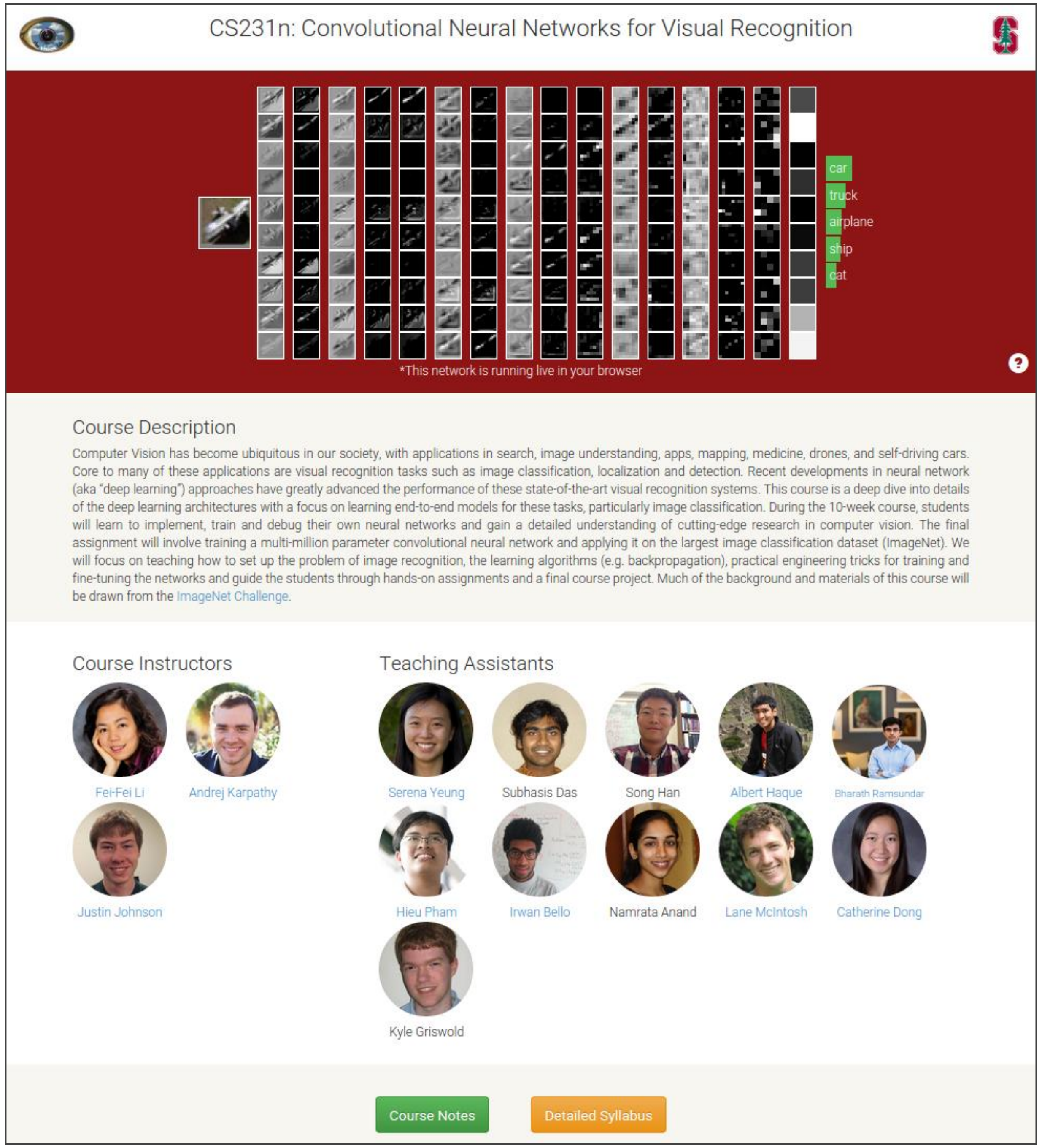
CNNs for vision/images

CS231n

cs231n.stanford.edu

- Specifically check out lecture 7
- https://www.youtube.com/watch?v=AQirPKrAyDg

Read http://cs231n.github.io/convolutional-networks/


CS231n: Convolutional Neural Networks for Visual Recognition

*This network is running live in your browser

Course Description

Computer Vision has become ubiquitous in our society, with applications in search, image understanding, apps, mapping, medicine, drones, and self-driving cars. Core to many of these applications are visual recognition tasks such as image classification, localization and detection. Recent developments in neural network (aka "deep learning") approaches have greatly advanced the performance of these state-of-the-art visual recognition systems. This course is a deep dive into details of the deep learning architectures with a focus on learning end-to-end models for these tasks, particularly image classification. During the 10-week course, students will learn to implement, train and debug their own neural networks and gain a detailed understanding of cutting-edge research in computer vision. The final assignment will involve training a multi-million parameter convolutional neural network and applying it on the largest image classification dataset (ImageNet). We will focus on teaching how to set up the problem of image recognition, the learning algorithms (e.g. backpropagation), practical engineering tricks for training and fine-tuning the networks and guide the students through hands-on assignments and a final course project. Much of the background and materials of this course will be drawn from the ImageNet Challenge.

Course Instructors

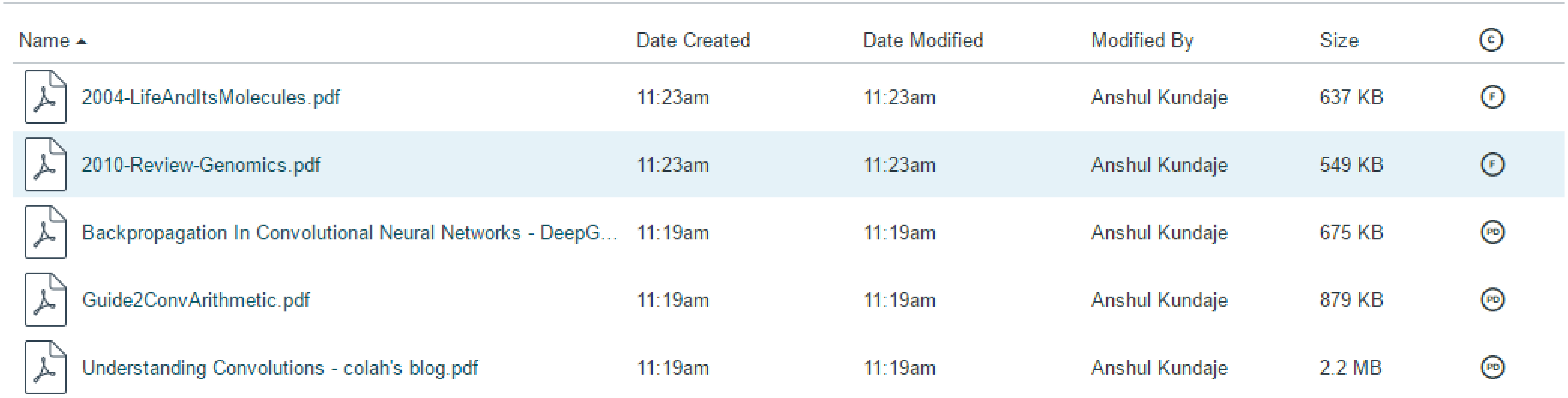
Fei-Fei Li
Andrej Karpathy
Justin Johnson

Teaching Assistants

Serena Yeung
Subhasis Das
Song Han
Albert Haque
Bharath Ramsundar
Hieu Pham
Irwan Bello
Namrata Anand
Lane McIntosh
Catherine Dong
Kyle Griswold

Additional optional readings

In Canvas



Name ▲	Date Created	Date Modified	Modified By	Size	Ⓒ
2004-LifeAndItsMolecules.pdf	11:23am	11:23am	Anshul Kundaje	637 KB	Ⓕ
2010-Review-Genomics.pdf	11:23am	11:23am	Anshul Kundaje	549 KB	Ⓕ
Backpropagation In Convolutional Neural Networks - DeepG…	11:19am	11:19am	Anshul Kundaje	675 KB	PD
Guide2ConvArithmetic.pdf	11:19am	11:19am	Anshul Kundaje	879 KB	PD
Understanding Convolutions - colah's blog.pdf	11:19am	11:19am	Anshul Kundaje	2.2 MB	PD

https://canvas.stanford.edu/courses/51037/files/folder/LectureMaterial/Lecture2