

ECE 5424: Introduction to Machine Learning

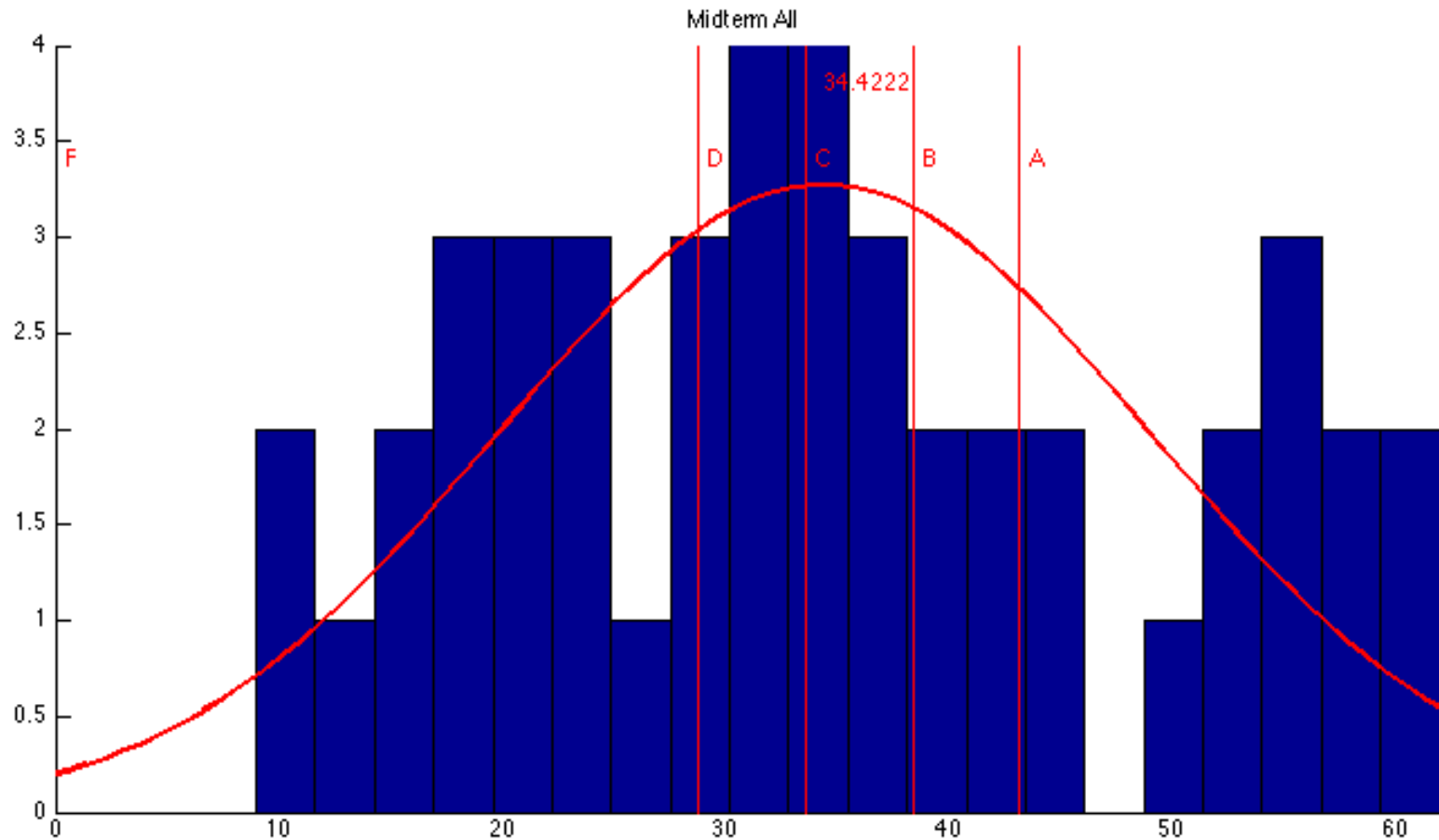
Topics:

- Neural Networks (again)
- Decision/Classification Trees

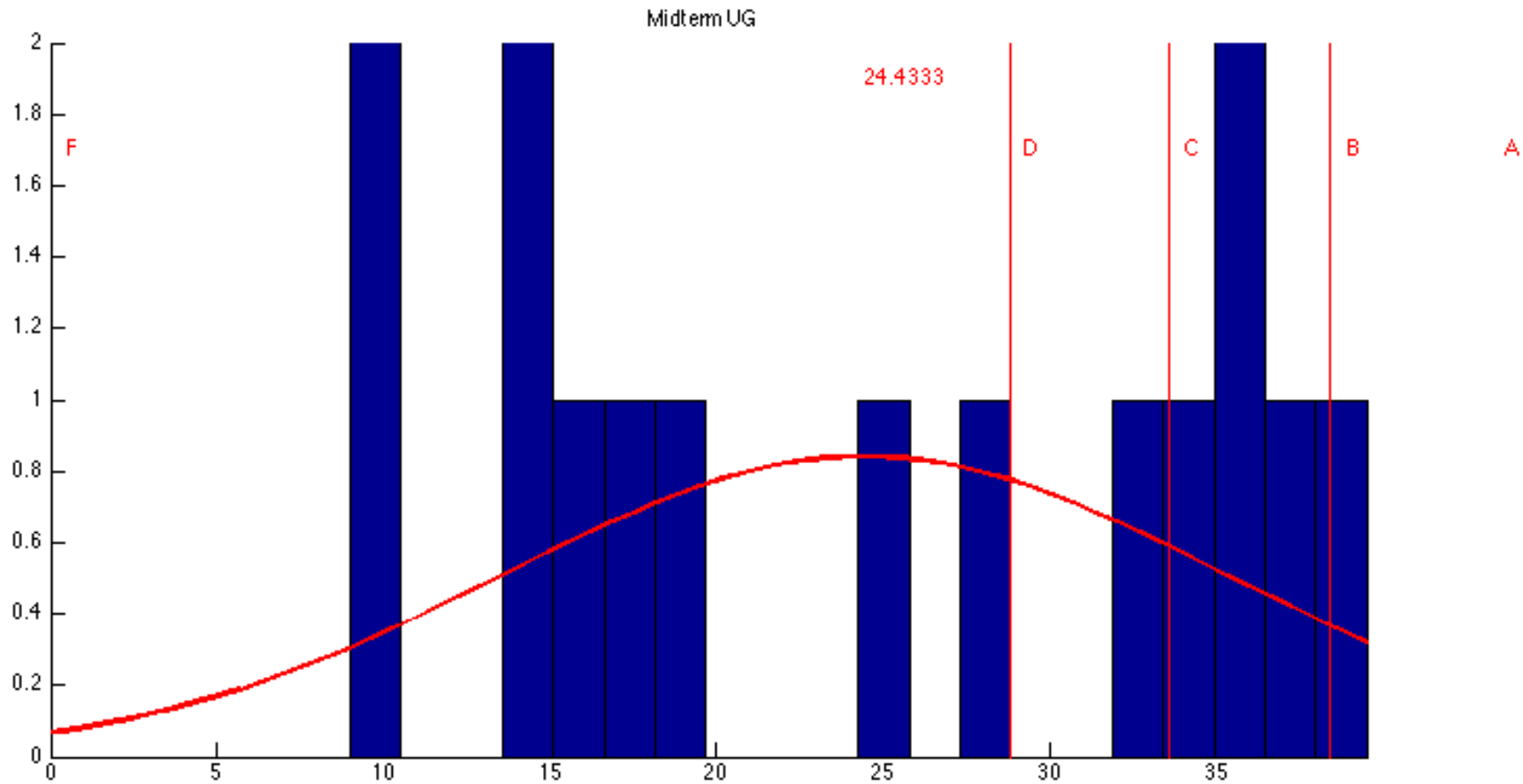
Readings: Murphy 16.1-16.2; Hastie 9.2

Stefan Lee
Virginia Tech

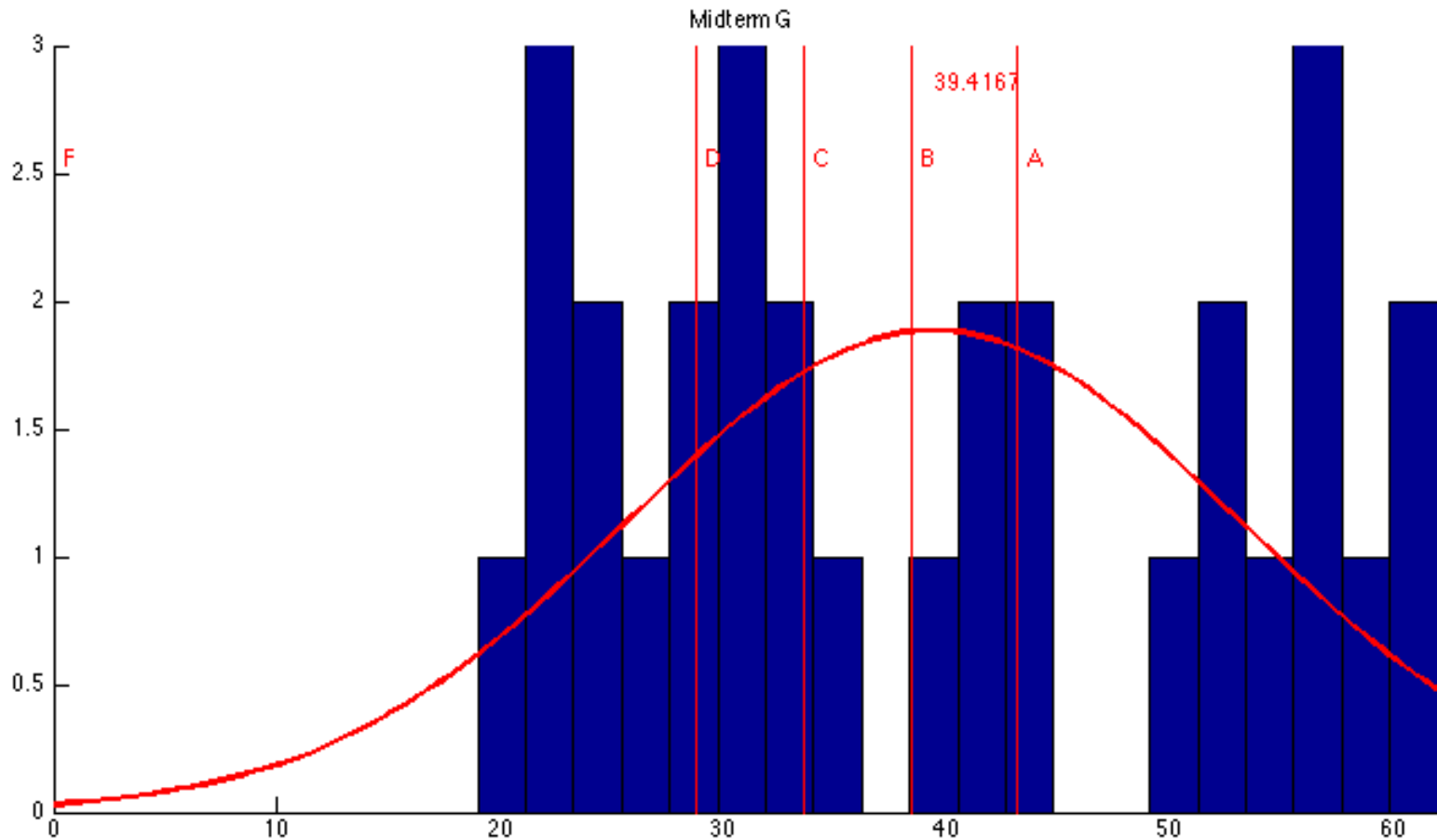
Midterms Graded (All)



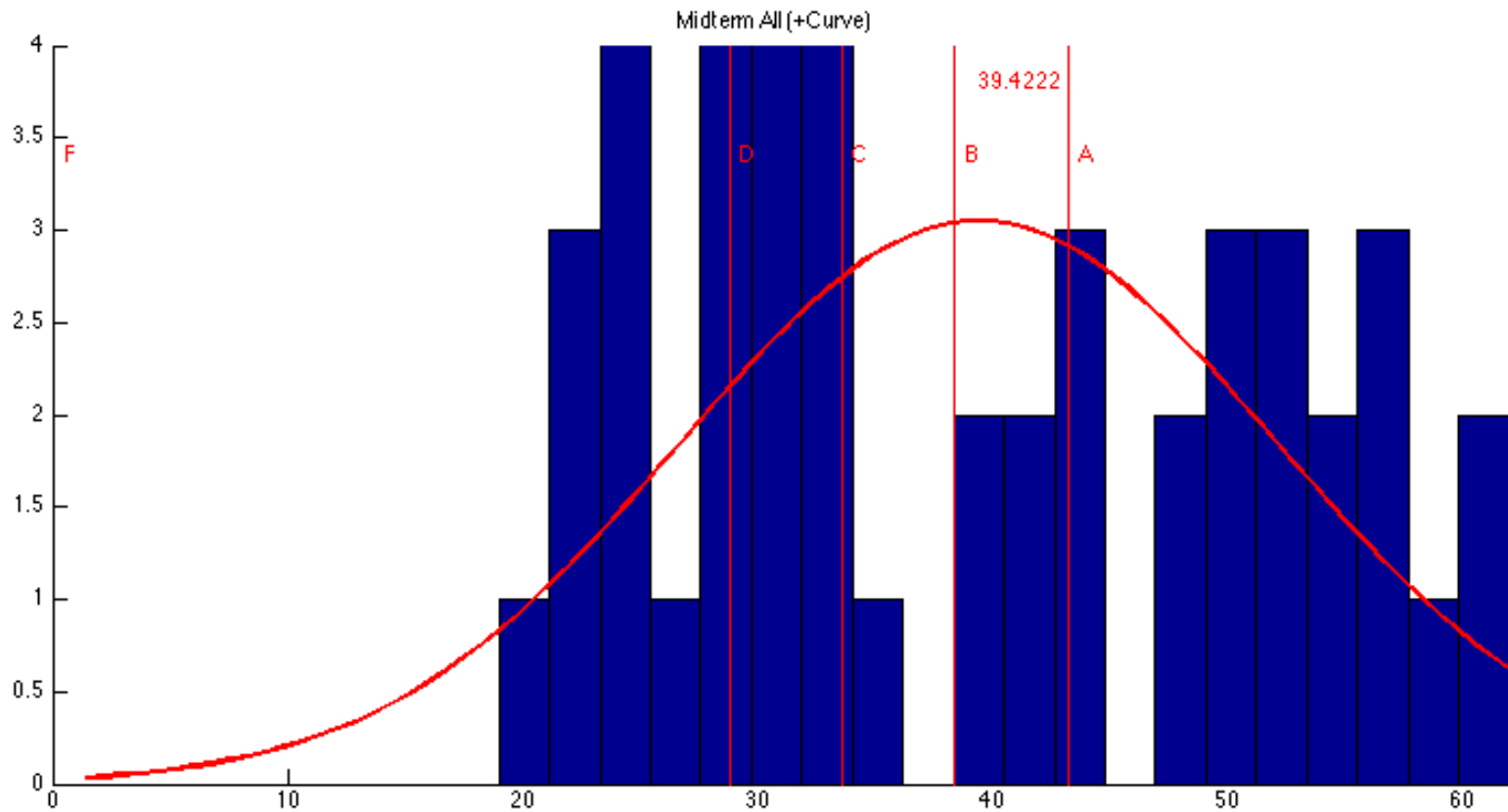
Midterms Graded (UGrad)



Midterms Graded (Grad)



Midterms Graded (All + Curve)



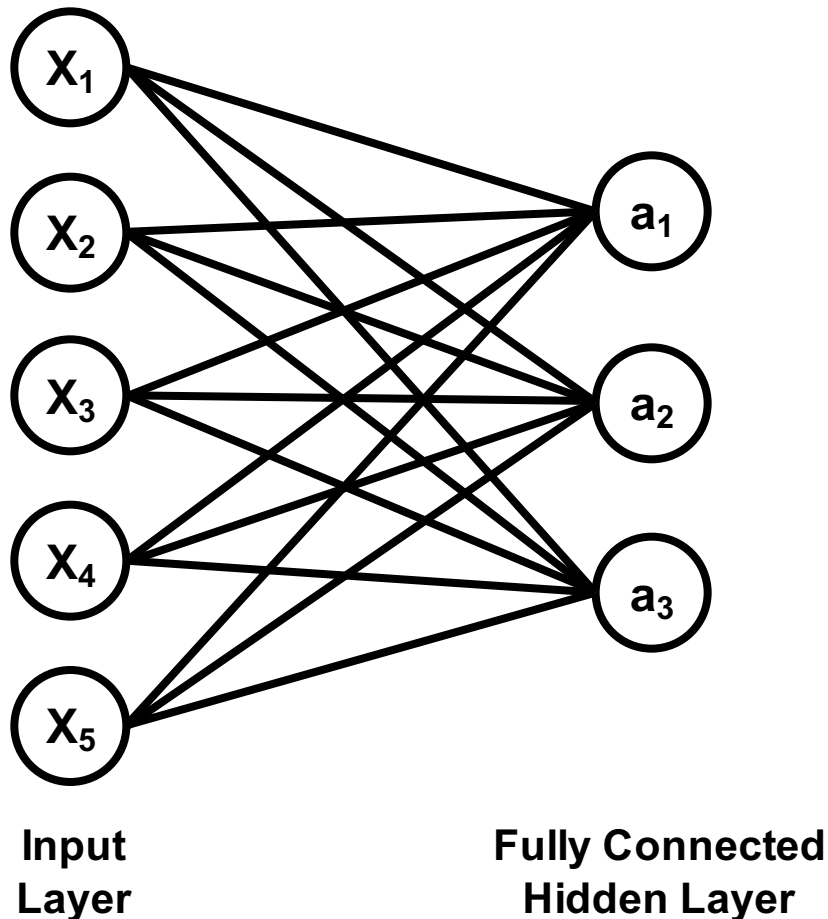
Administrativa

- Project Mid-Sem Spotlight Presentations
 - T/R Nov 8/10th: In-class
 - 5 slides (recommended)
 - 7 minute time (STRICT) + 1-2 min Q&A
 - Tell the class what you're working on
 - Any results yet?
 - Problems faced?
 - Upload slides on Scholar
- Also remember HW3 is due Nov 7th! Start Early.
 - Small typo on assignment, will send email from Scholar today.

Does everyone understand
Convolutional Neural Networks?

Lets Try Convolutional Neural Networks Again

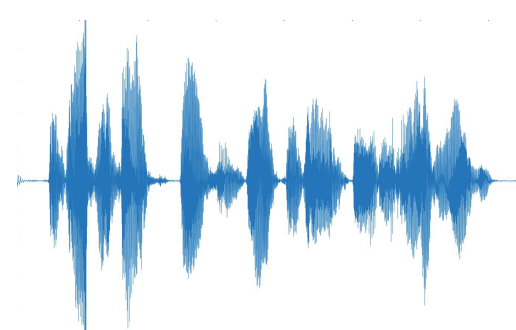
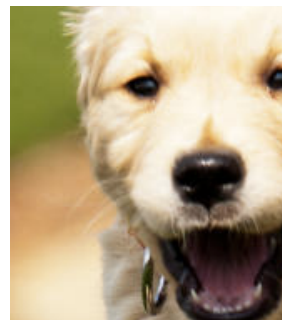
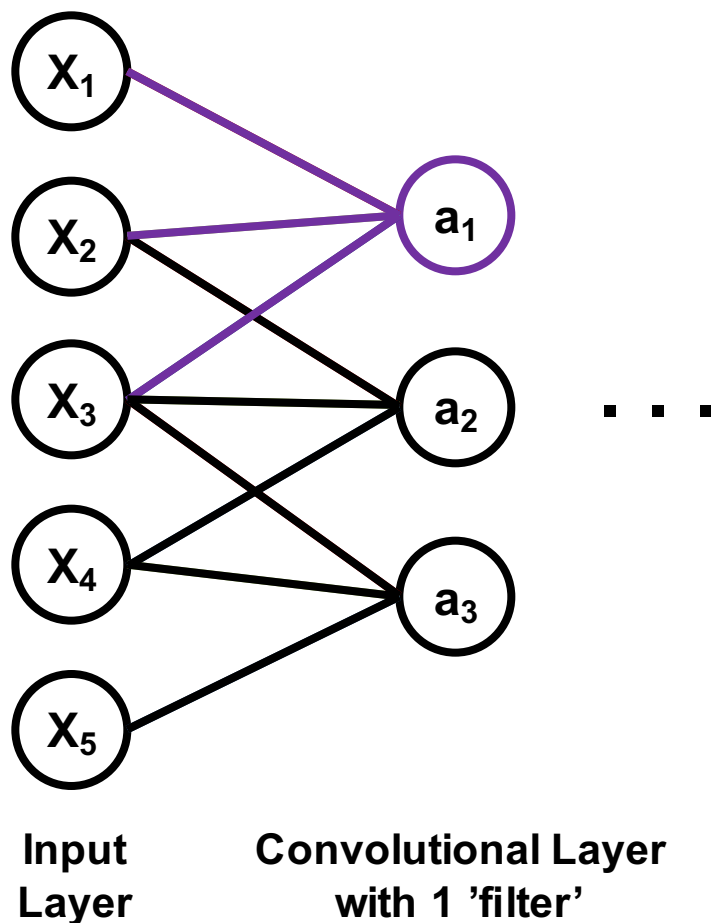
- Lets consider a neural network with a vector input:



Fully connected layers have a ton of parameters if input dimension is large!

Lets Try Convolutional Neural Networks Again

- The assumption: locality

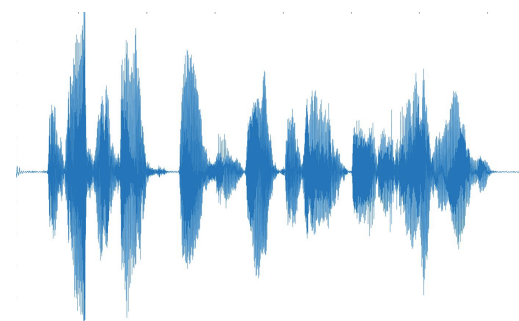
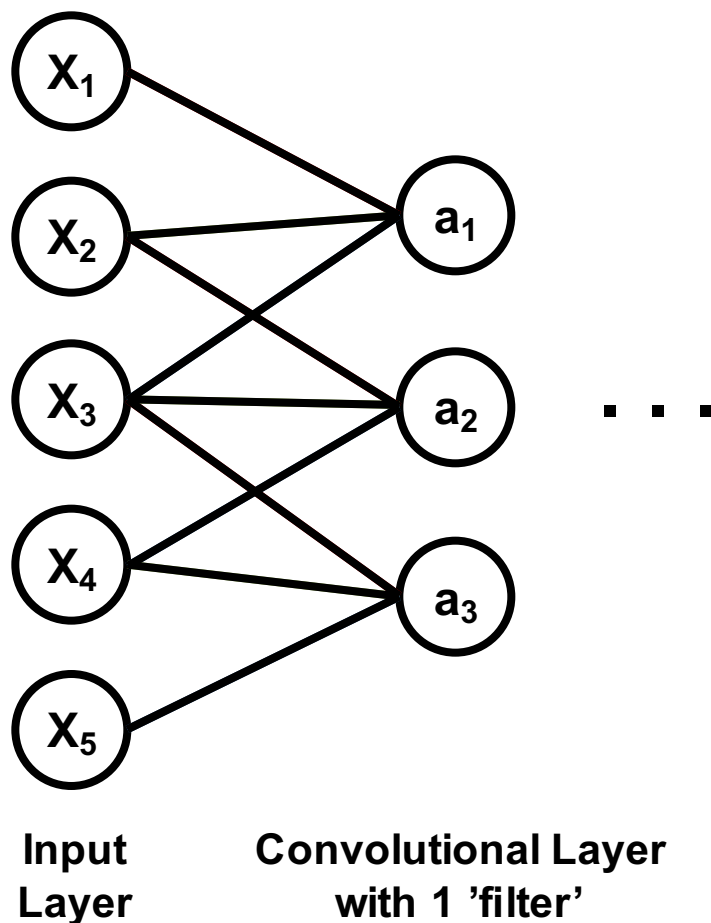


Assume input dependencies are local in the signal.

Connect each neuron to only a local subset of the input signal.

Lets Try Convolutional Neural Networks Again

- Fixing stationarity

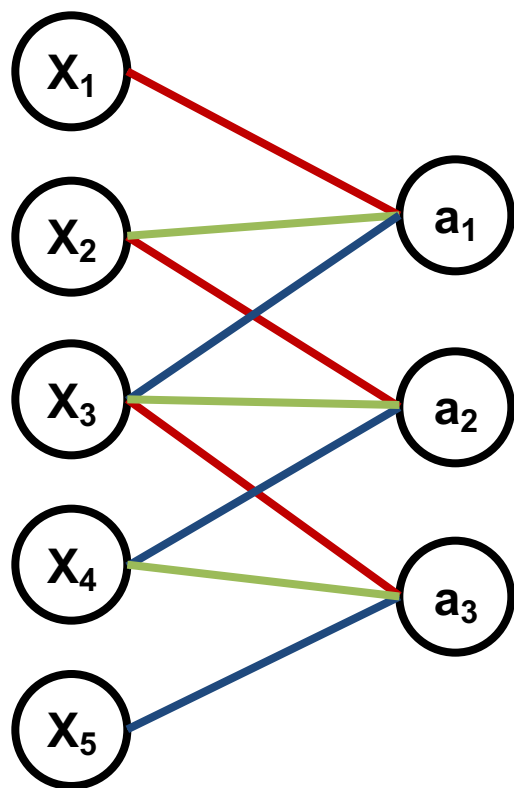


Problem! This local dependency would be stationary in the signal.

Example: Speech detection would need to learn a 'hard k' sound detector at each location!

Lets Try Convolutional Neural Networks Again

- Fixing stationarity



Input
Layer

Convolutional Layer
with 1 'filter'

$$a_1 = x_1 * w_1 + x_2 * w_2 + x_3 * w_3$$

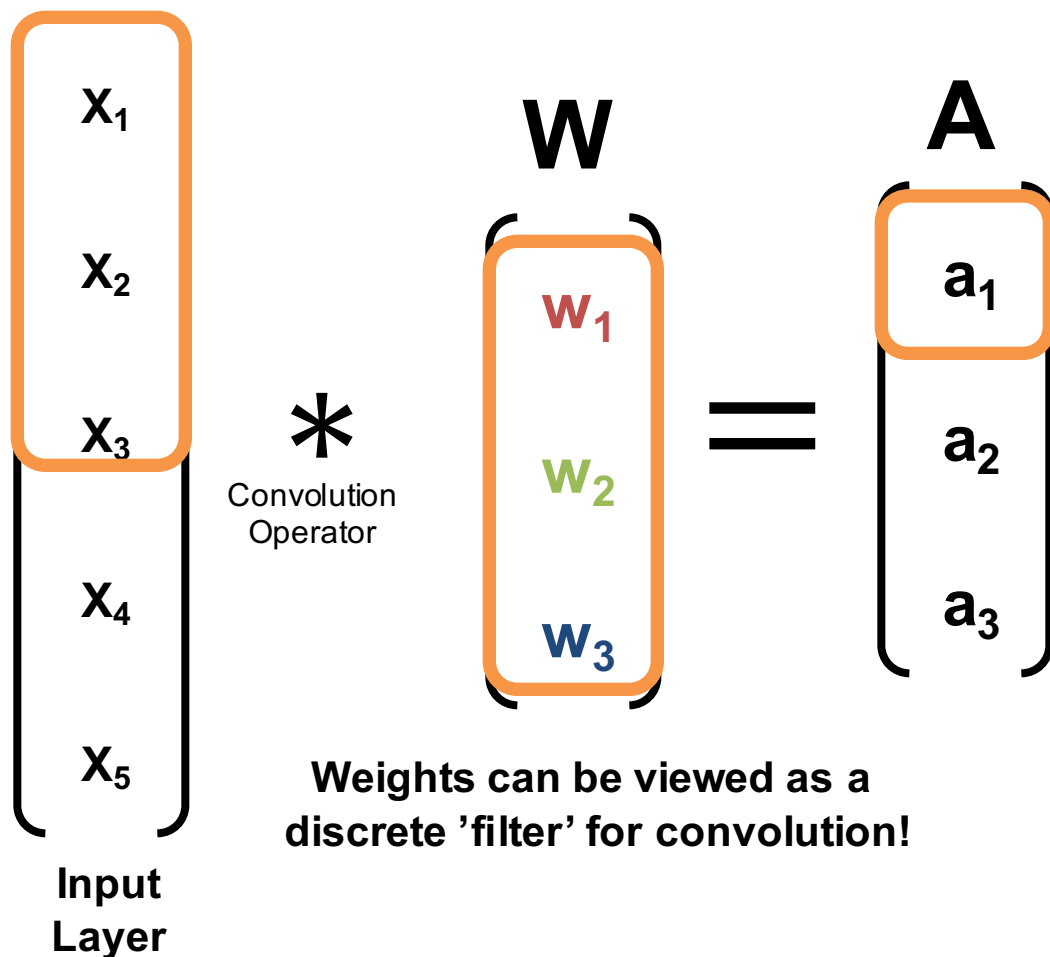
$$a_2 = x_2 * w_1 + x_3 * w_2 + x_4 * w_3$$

$$a_3 = x_3 * w_1 + x_4 * w_2 + x_5 * w_3$$

Solution: Share weights at each
local group!

Lets Try Convolutional Neural Networks Again

- Looks like a discrete convolution!



We can use powerful convolutional approaches (FFT for example) do forward-pass efficiently!

Lets Try Convolutional Neural Networks Again

- For representative power, we might want to learn multiple filters over the input.

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} \begin{matrix} * \\ * \\ \vdots \\ * \end{matrix} \begin{matrix} W_1 \\ W_2 \\ \vdots \\ W_m \end{matrix} = \begin{matrix} A_1 \\ A_2 \\ \vdots \\ A_m \end{matrix} \begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ \vdots & \vdots & \vdots \\ A_{m1} & A_{m2} & A_{m3} \end{pmatrix}$$

Input Layer

We can learn multiple filters per layer.

Each filter outputs a vector. We can stack these vectors as the output of our layer.

Lets Try Convolutional Neural Networks Again

- Now we have a 2D signal for the next layer? What now?

$$\begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} & \mathbf{A}_{13} \\ \mathbf{A}_{21} & \mathbf{A}_{22} & \mathbf{A}_{23} \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \mathbf{A}_{m1} & \mathbf{A}_{m2} & \mathbf{A}_{m3} \end{pmatrix} * \begin{pmatrix} \mathbf{w}_1 & \mathbf{w}_2 \\ \mathbf{w}_3 & \mathbf{w}_4 \\ \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \\ \mathbf{w}_{2m-1} & \mathbf{w}_{2m} \end{pmatrix}$$

One filter!

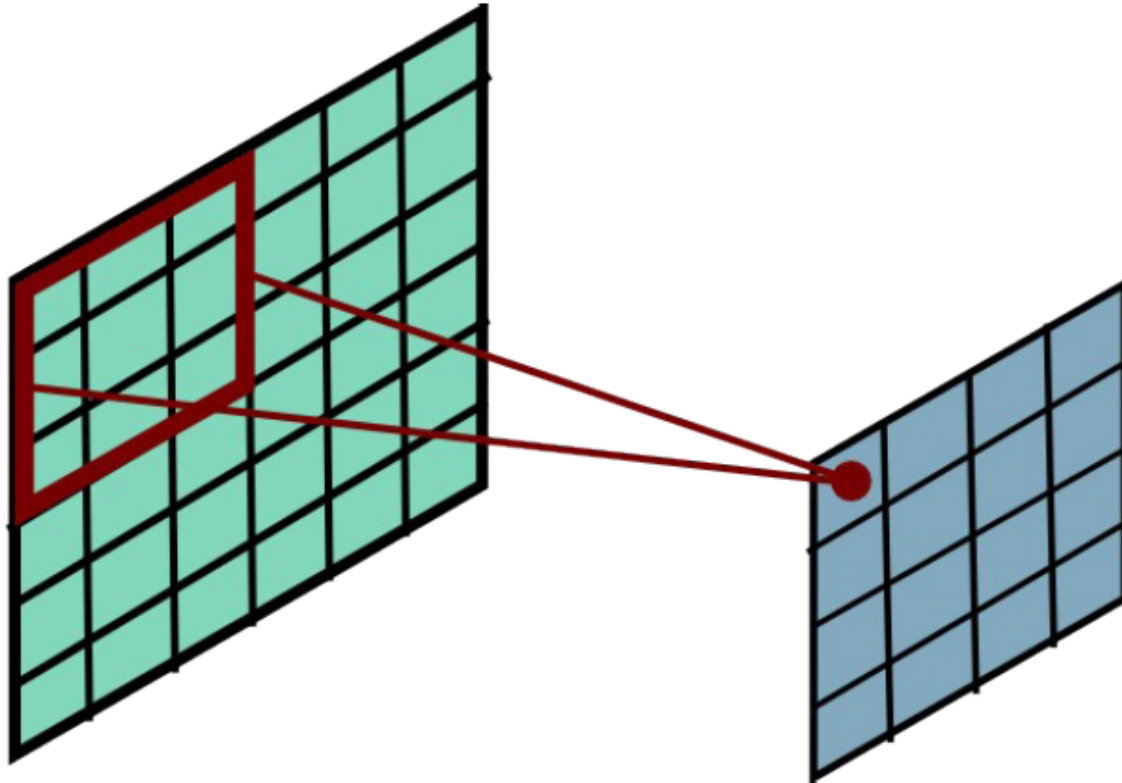
What size?

One good idea is to make one dim the same as the number of filters in the last layer.

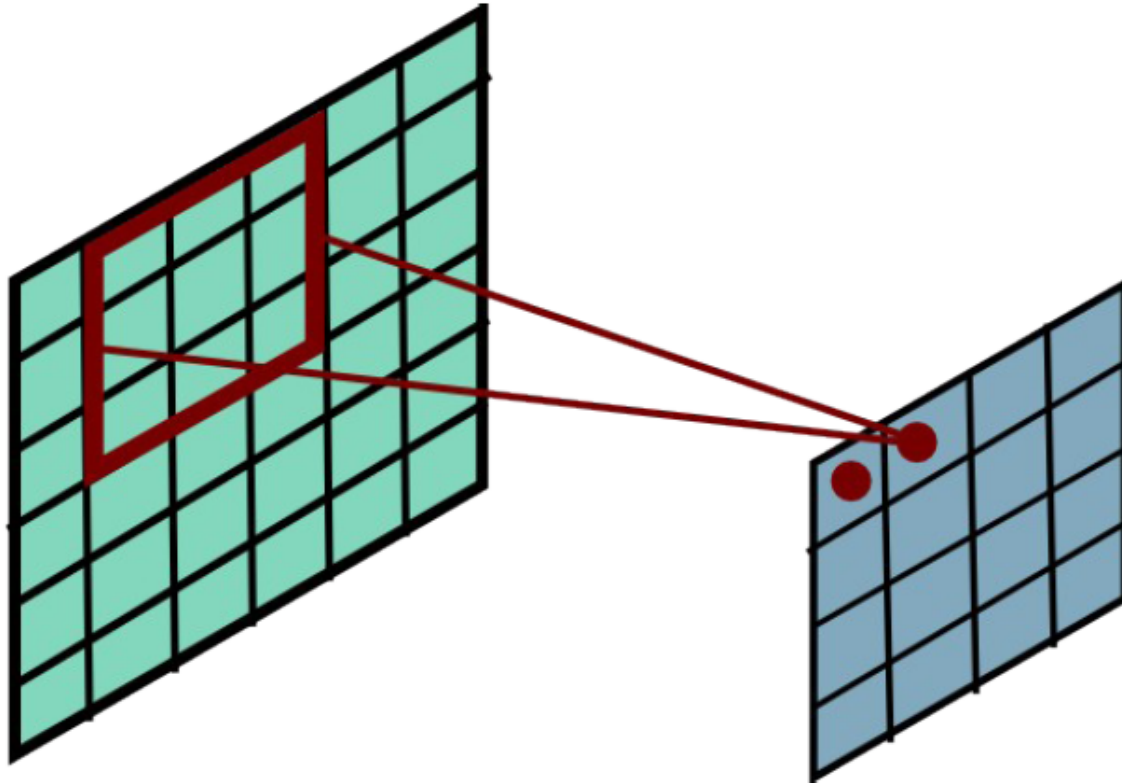
Why? Mix evidence collected from all filters at a certain spatial location.

Apply same logic and learn 2D filters!

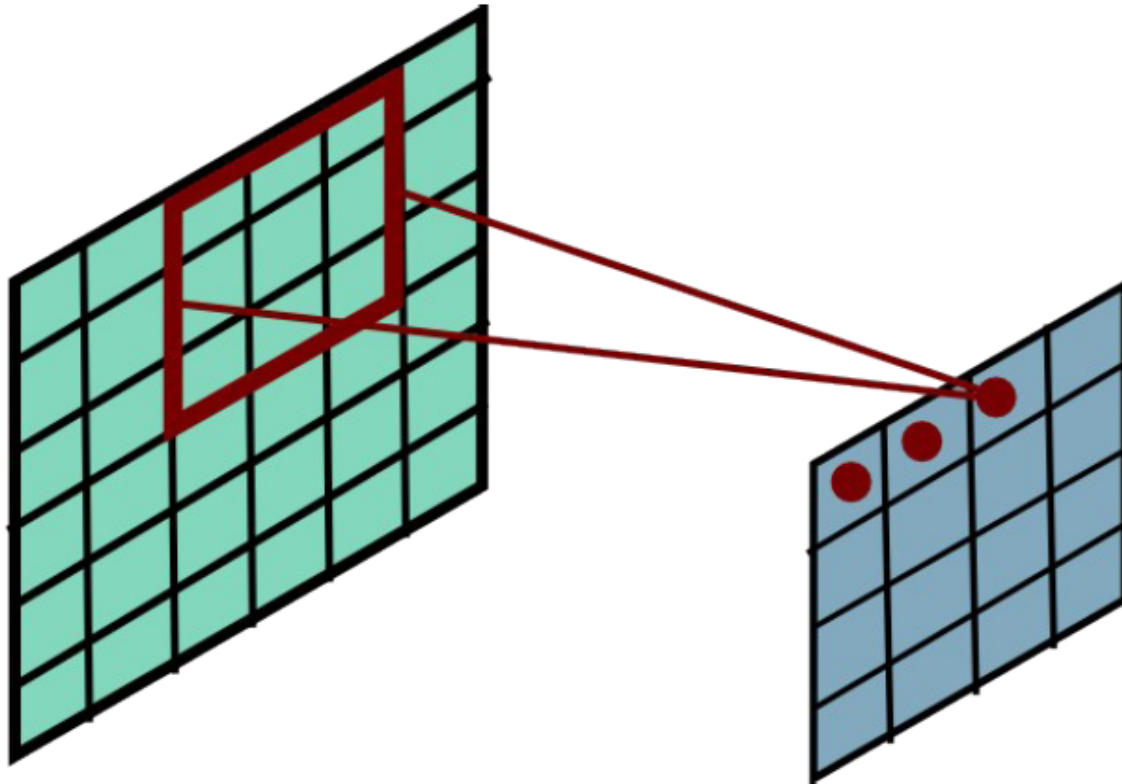
Convolutional Layer



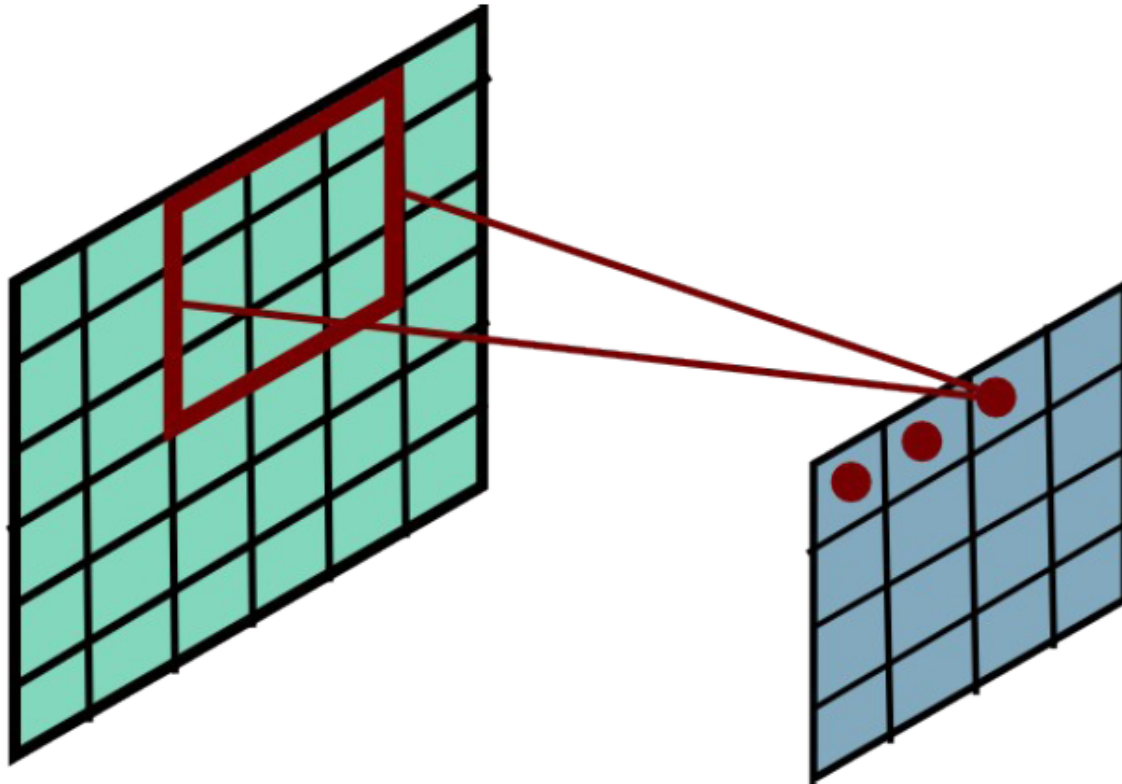
Convolutional Layer



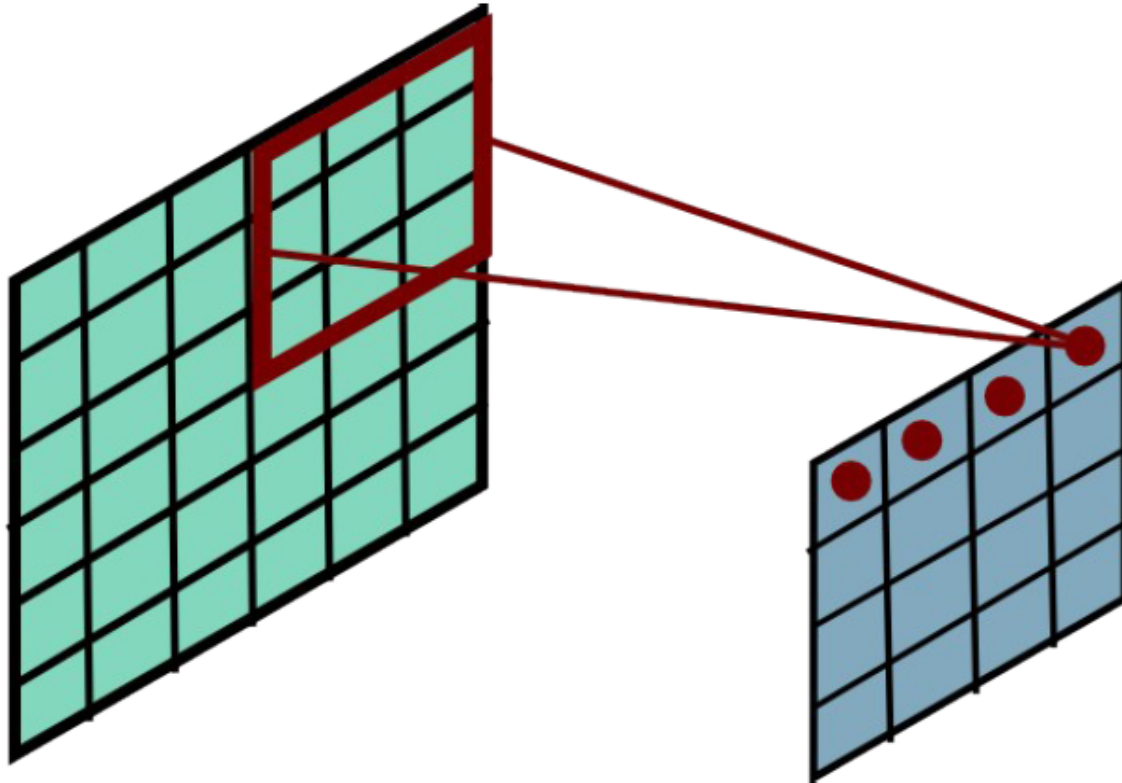
Convolutional Layer



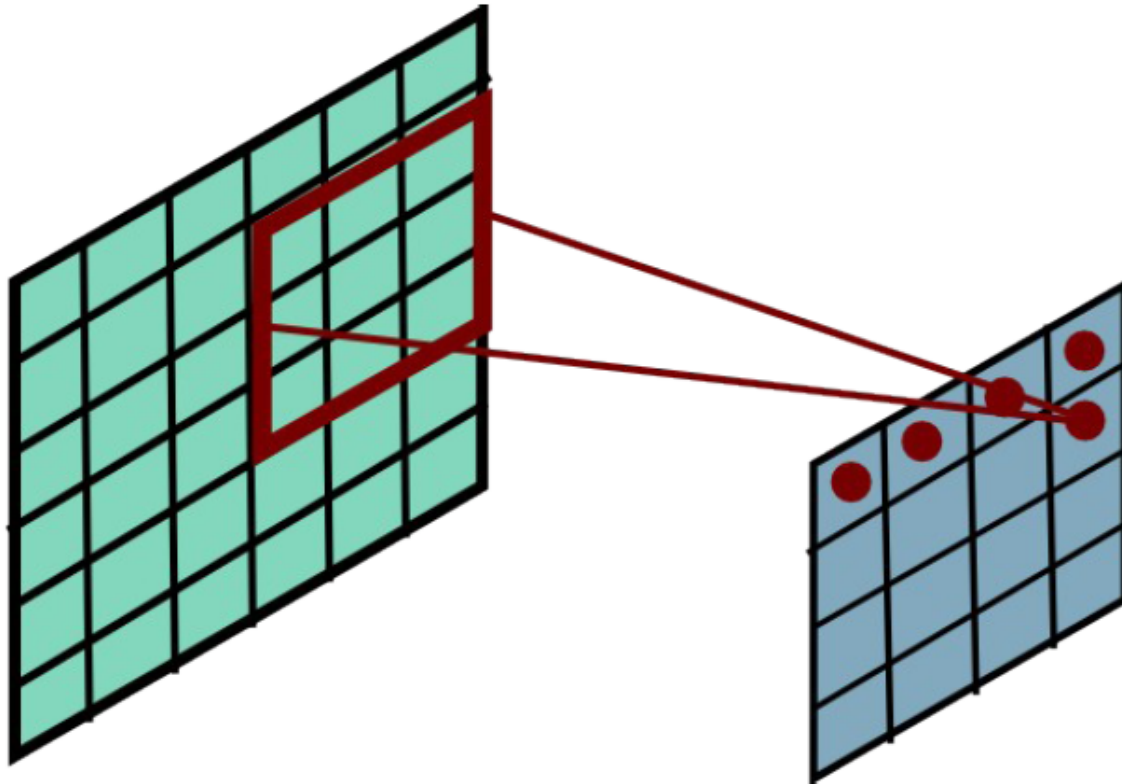
Convolutional Layer



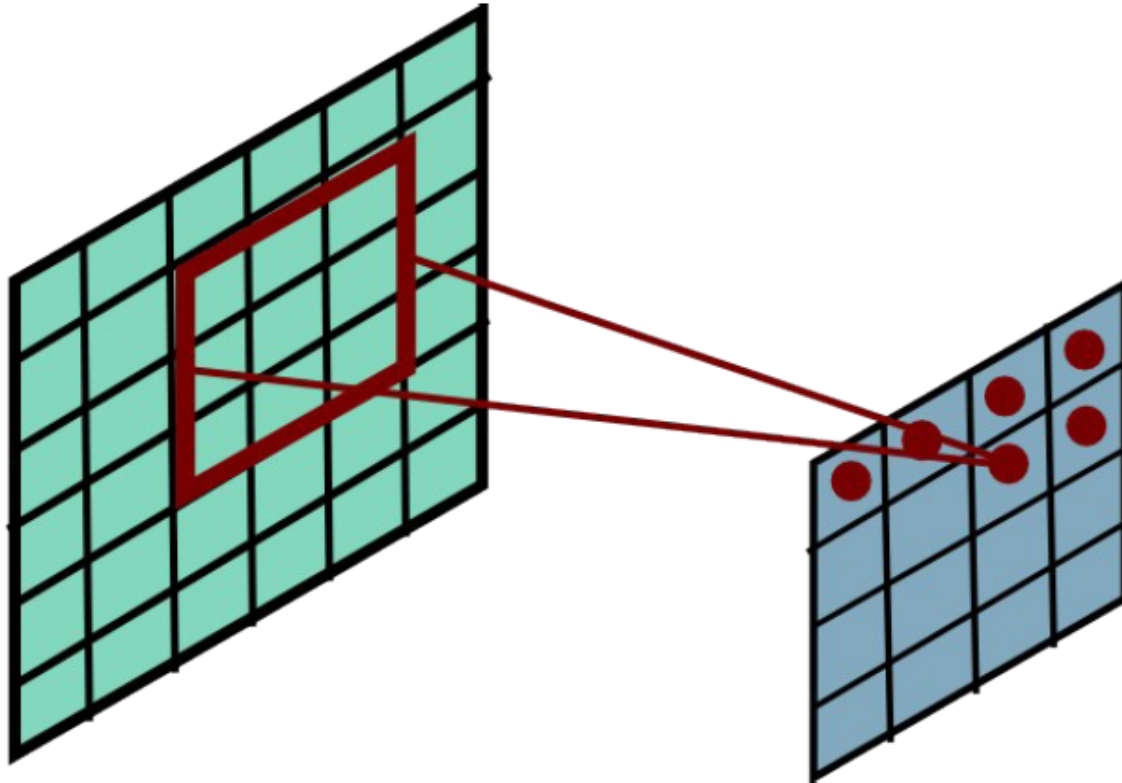
Convolutional Layer



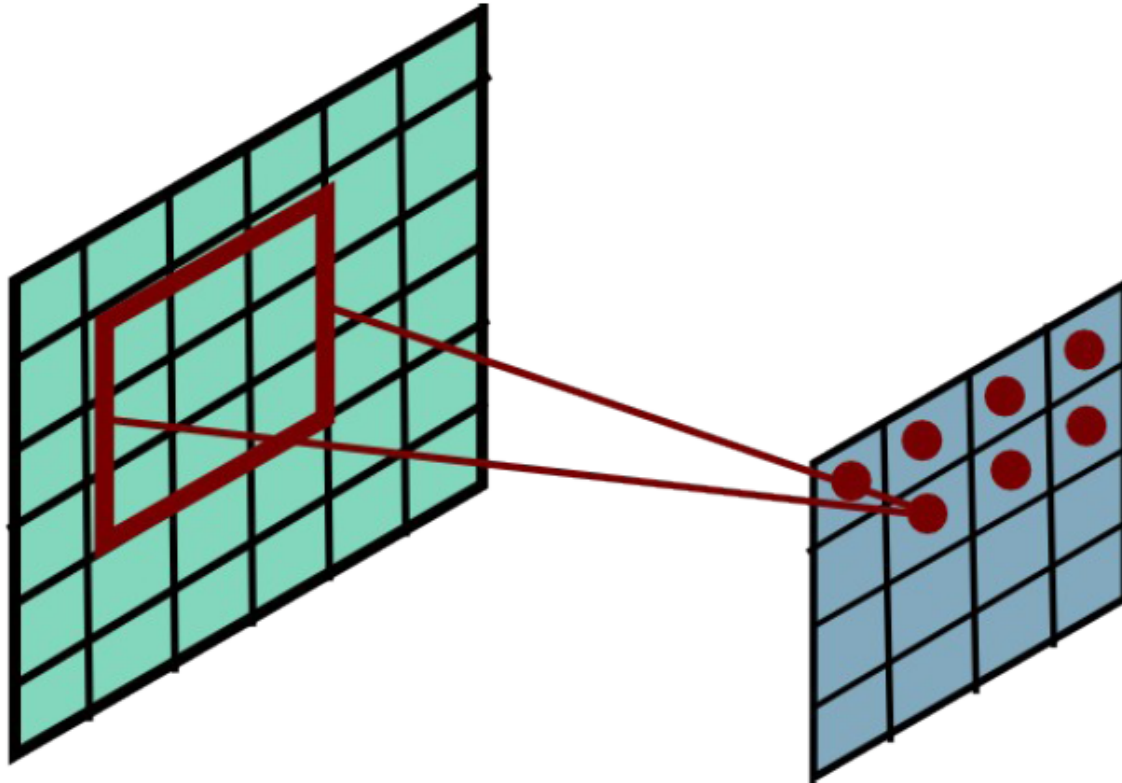
Convolutional Layer



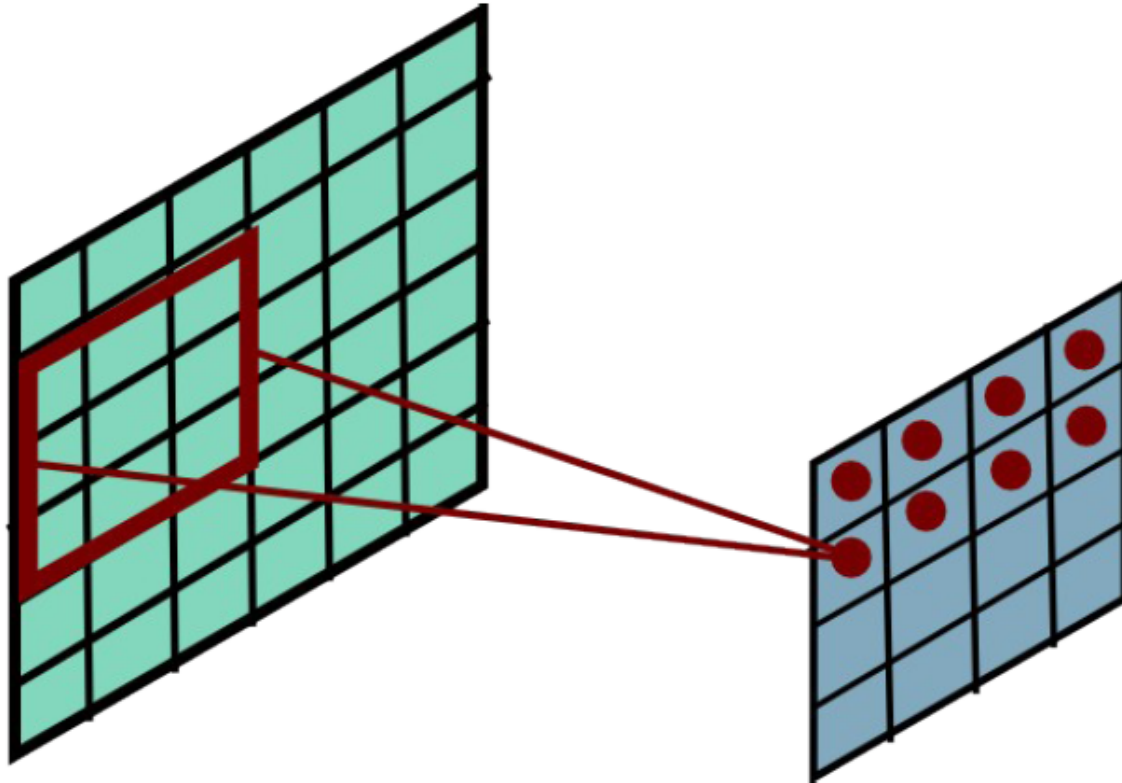
Convolutional Layer



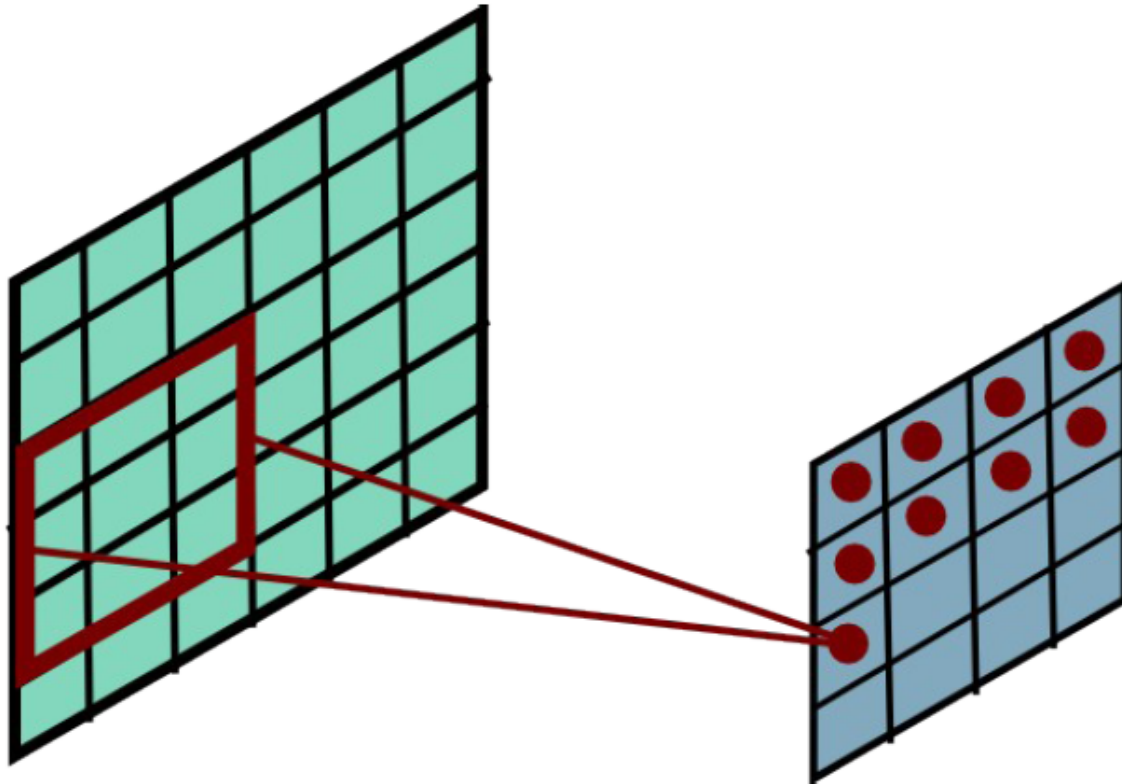
Convolutional Layer



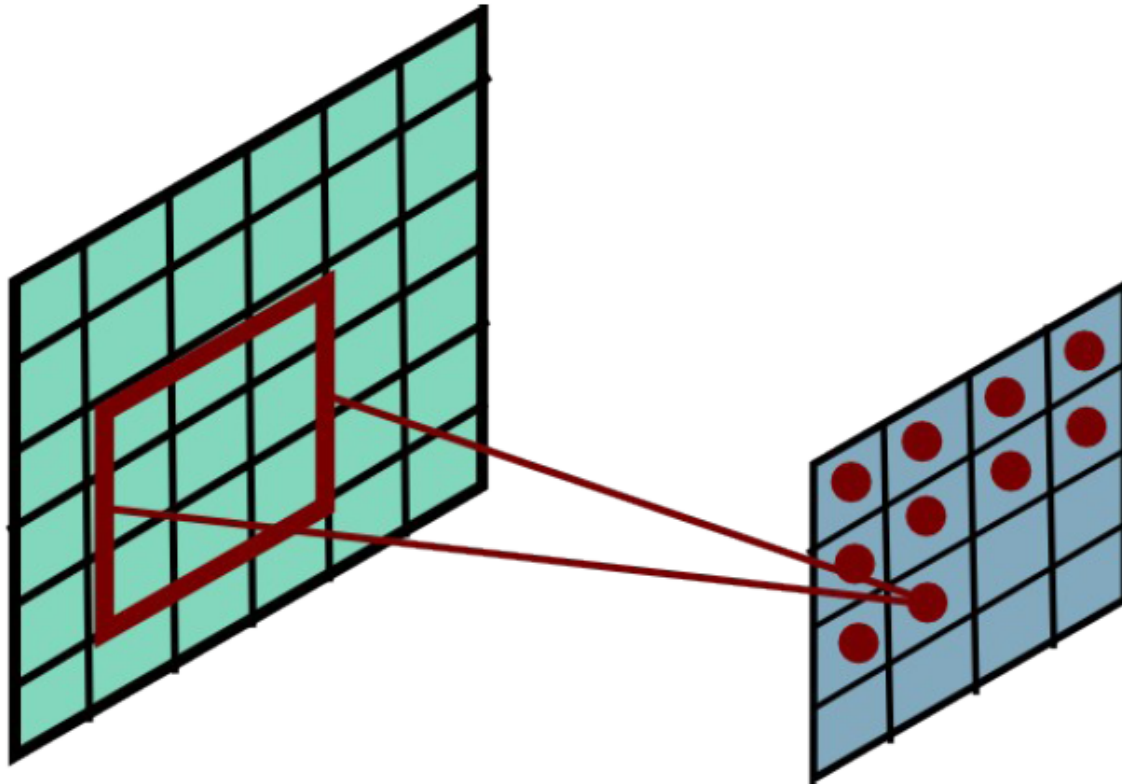
Convolutional Layer



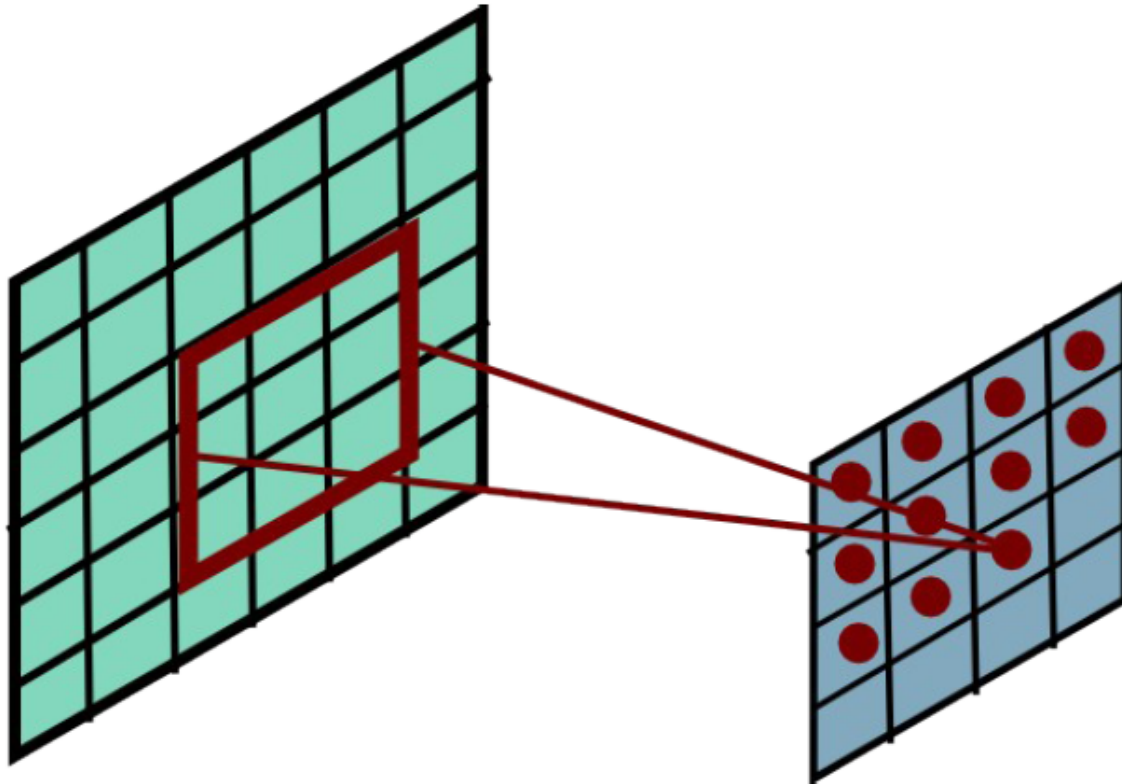
Convolutional Layer



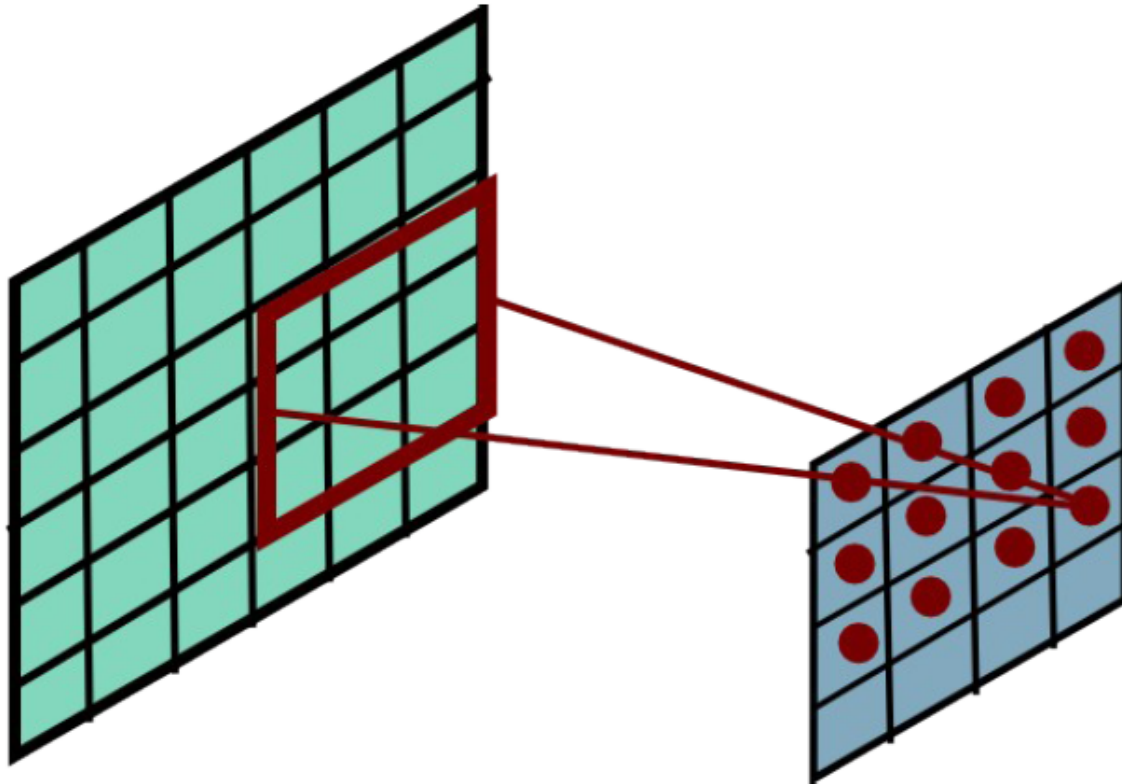
Convolutional Layer



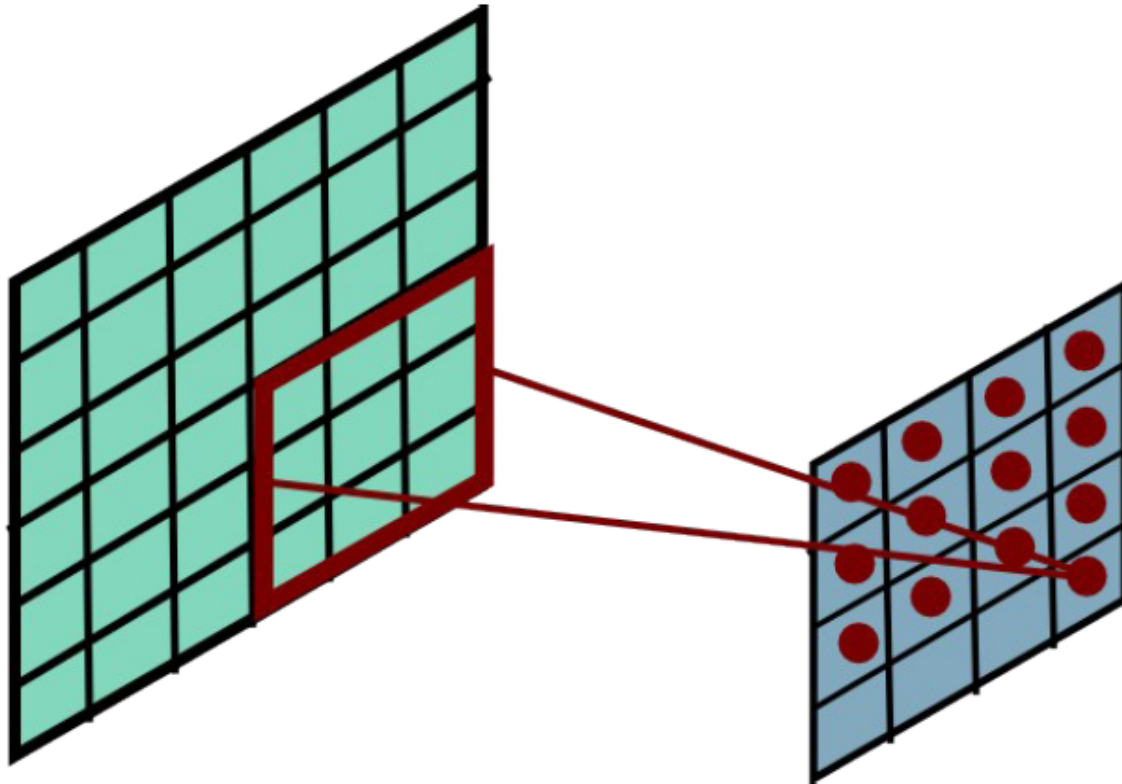
Convolutional Layer



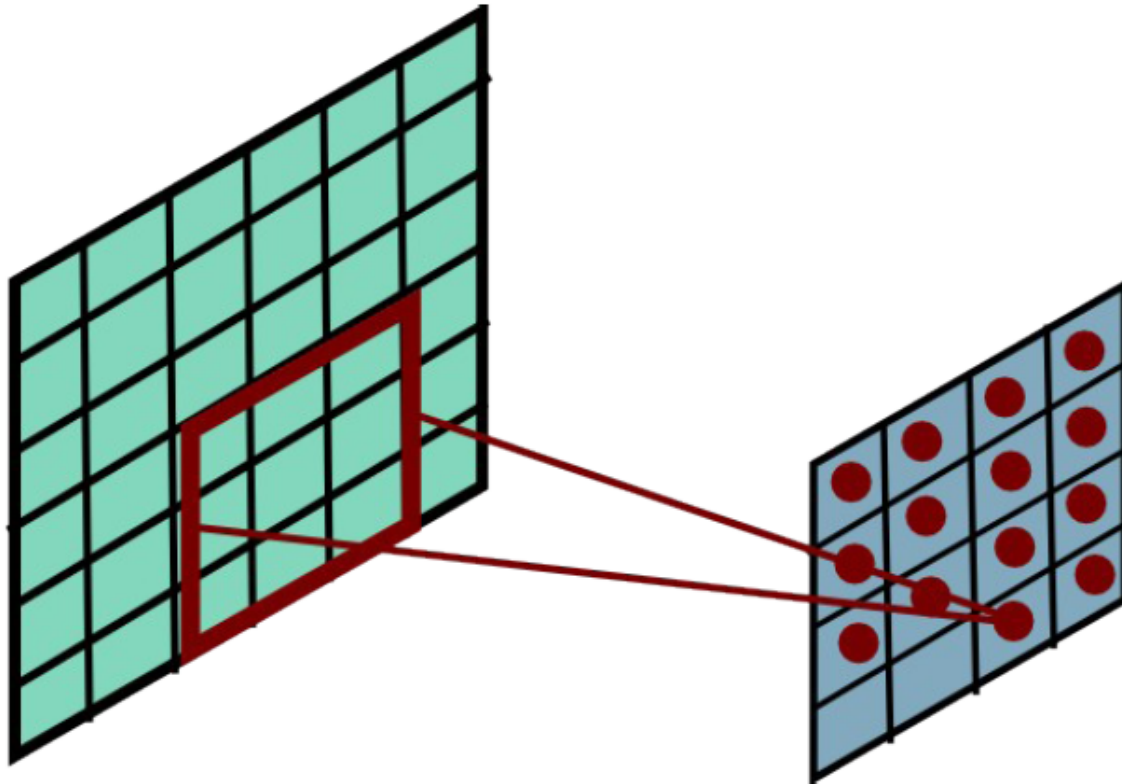
Convolutional Layer



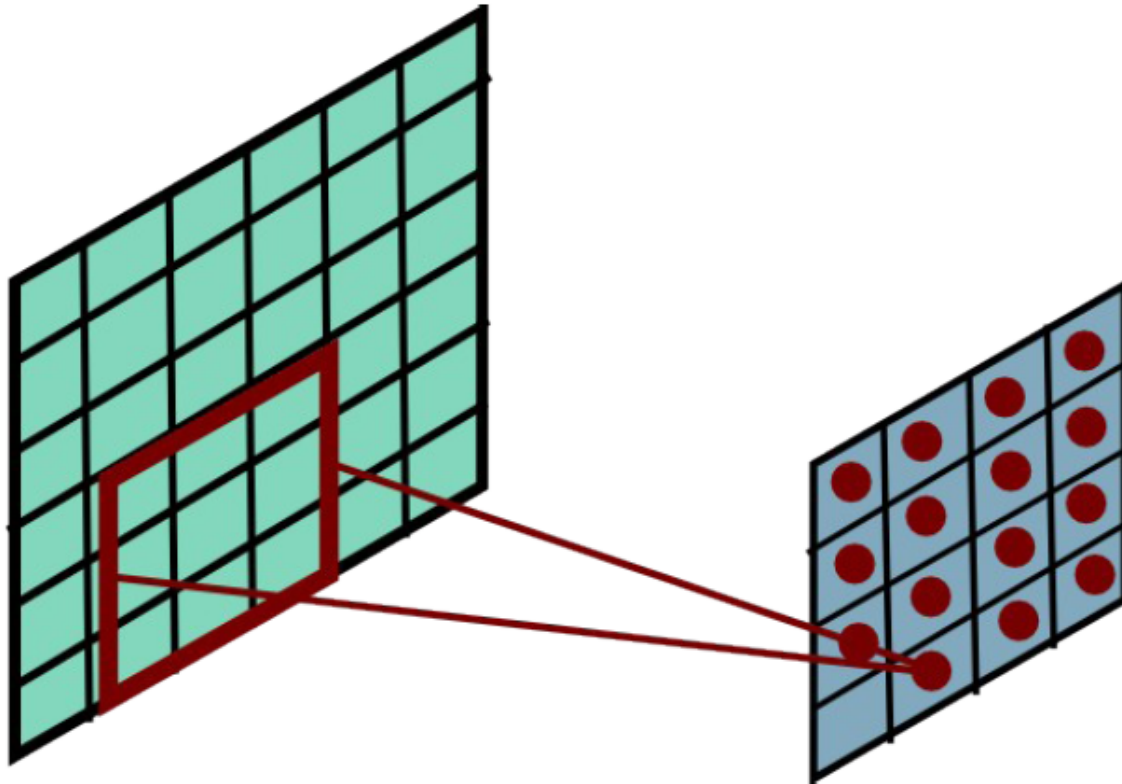
Convolutional Layer



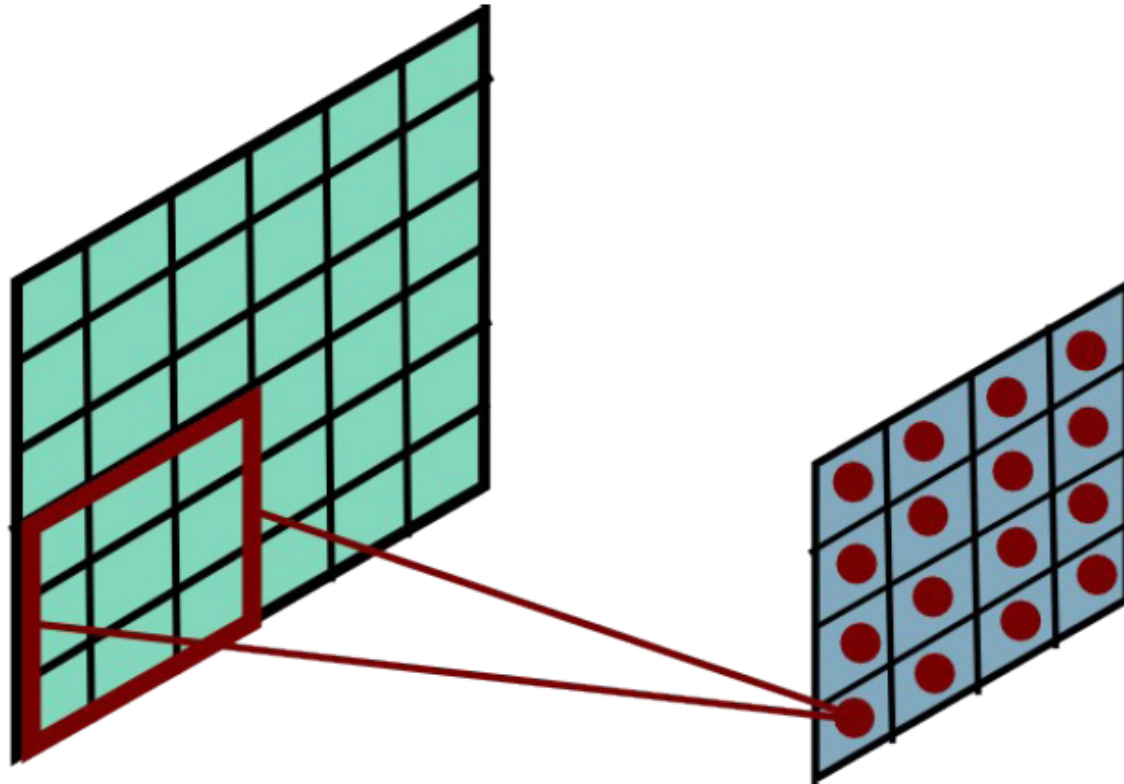
Convolutional Layer



Convolutional Layer



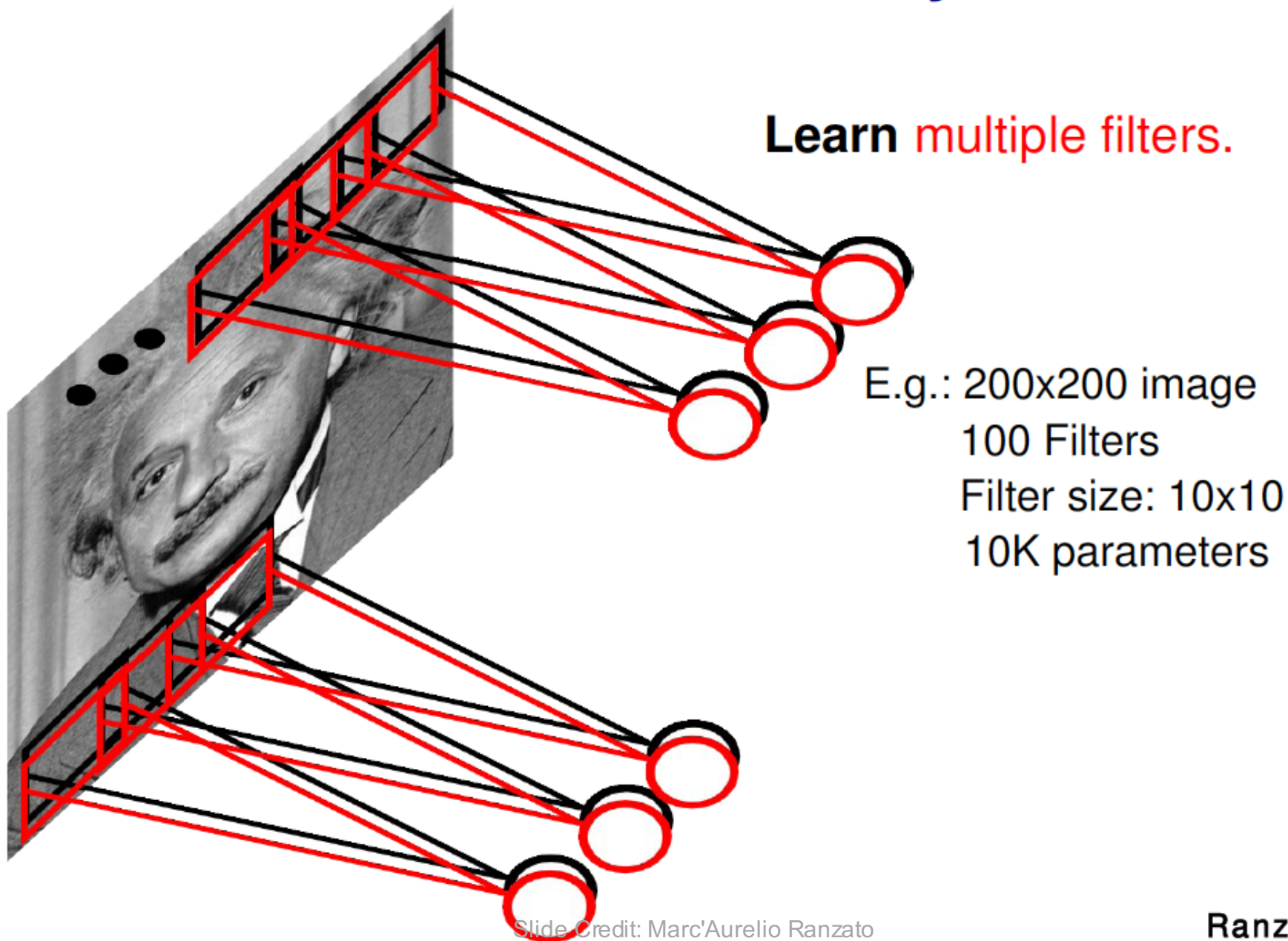
Convolutional Layer



Lets Try Convolutional Neural Networks Again

- What about n-d inputs?
 - Just learn higher dimensional filters!
 - For images:
 - <http://setosa.io/ev/image-kernels/>

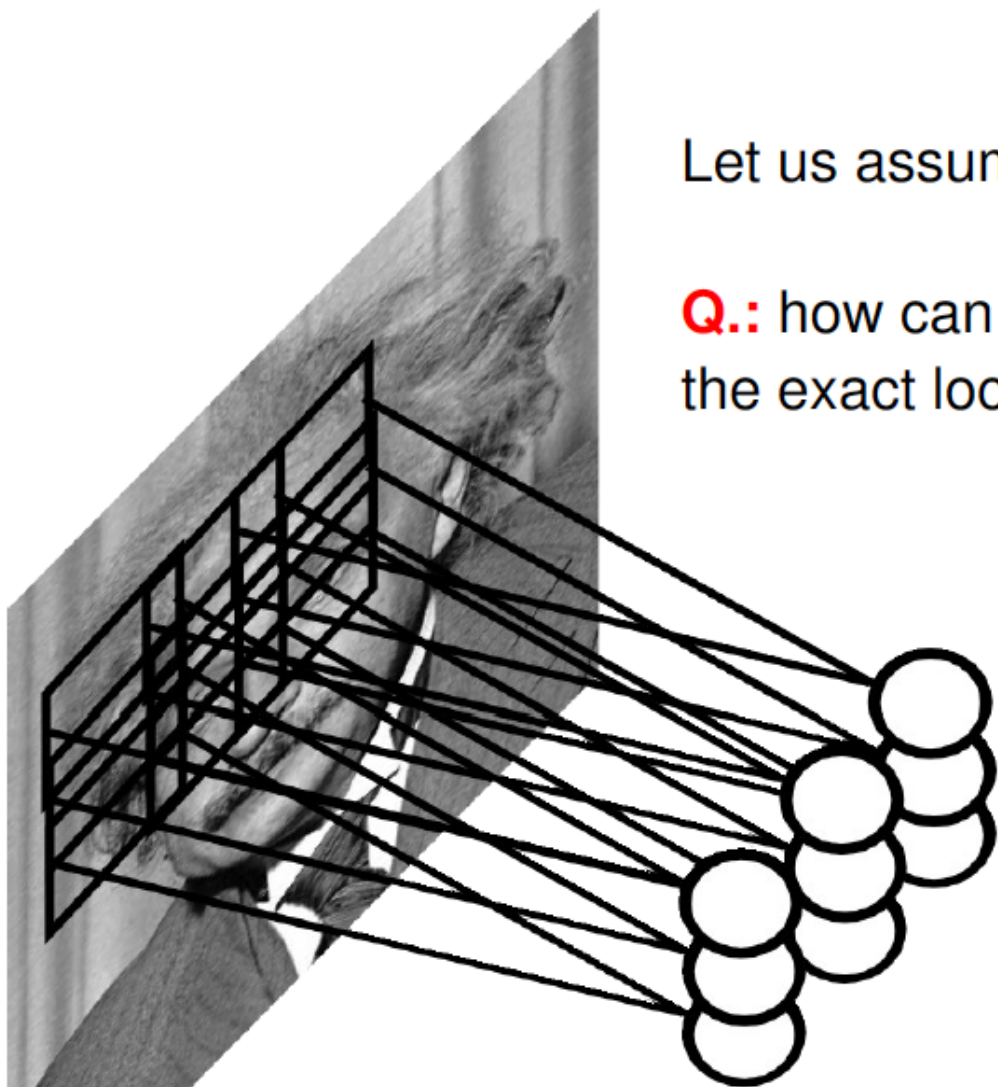
Convolutional Layer



Pooling Layer

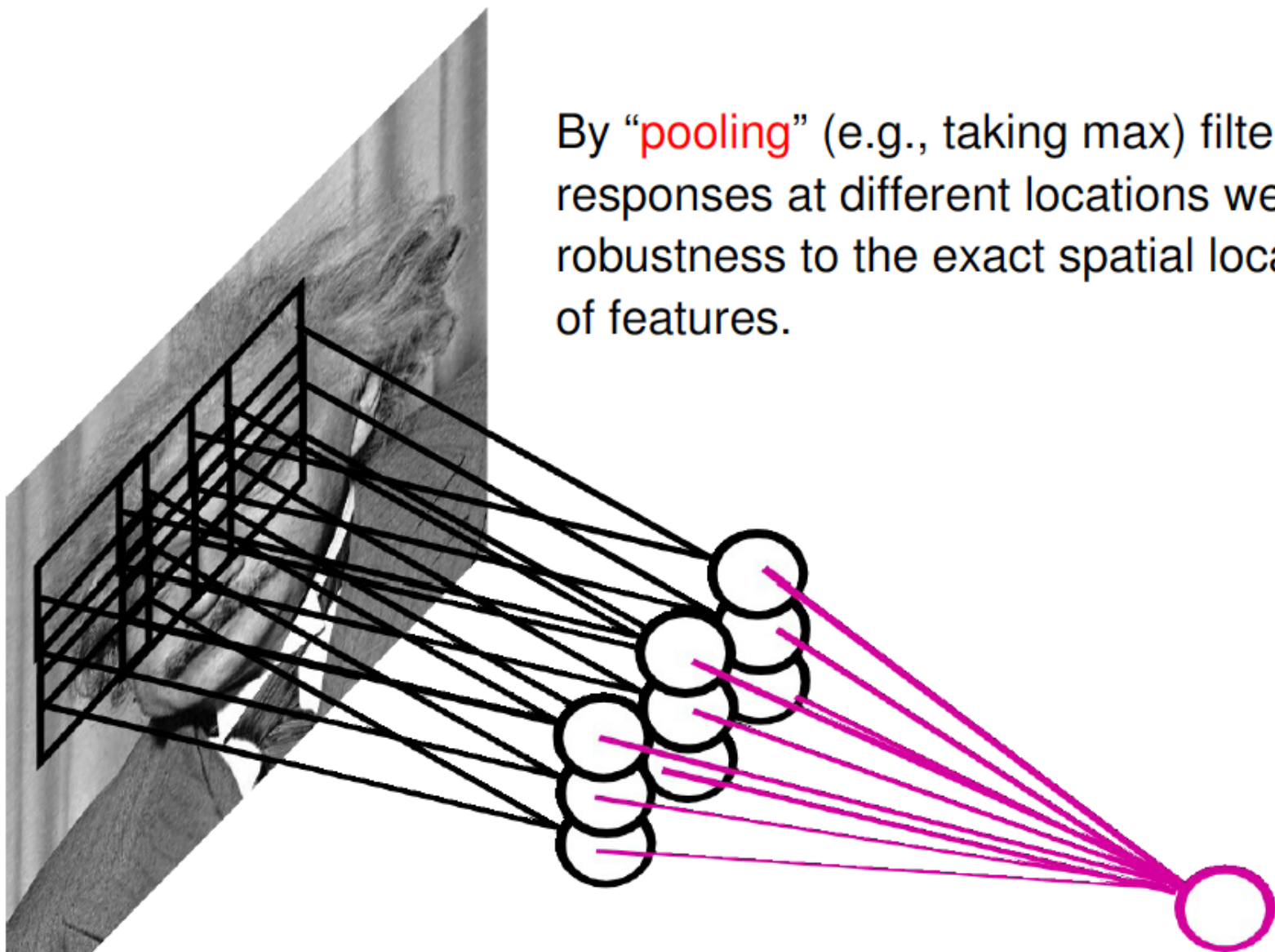
Let us assume filter is an “eye” detector.

Q.: how can we make the detection robust to the exact location of the eye?



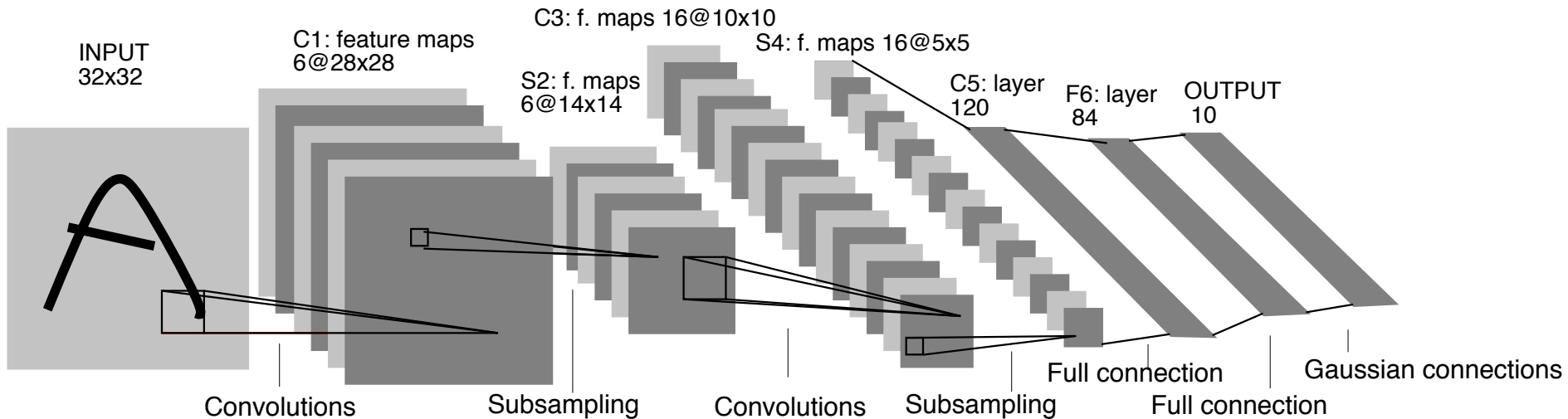
Pooling Layer

By “pooling” (e.g., taking max) filter responses at different locations we gain robustness to the exact spatial location of features.

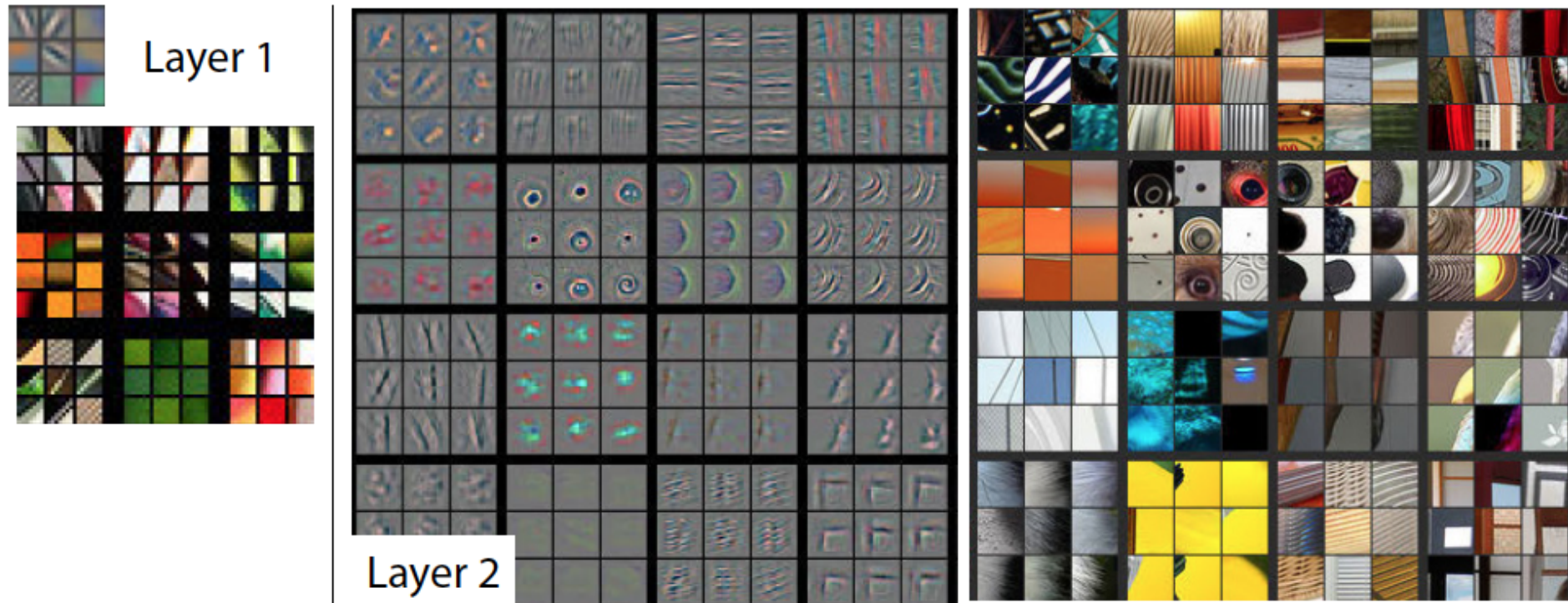


Convolutional Nets

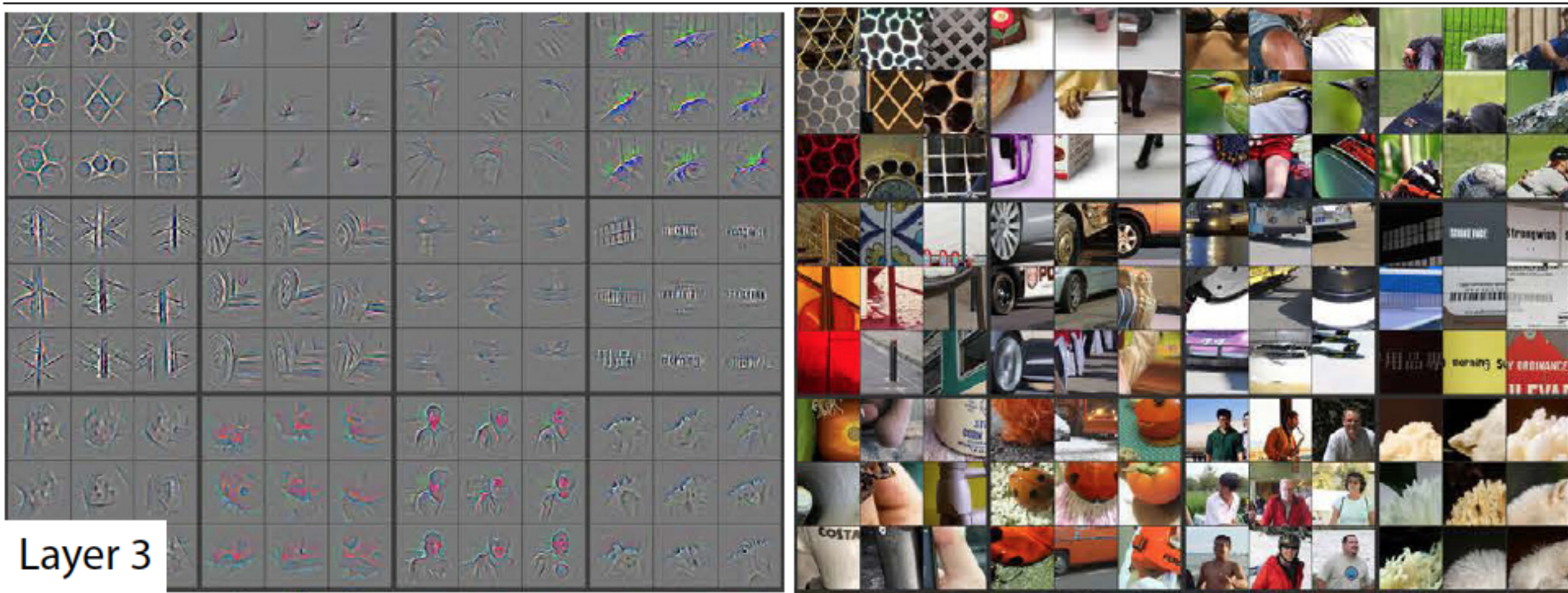
- Example:
 - <http://yann.lecun.com/exdb/lenet/index.html>



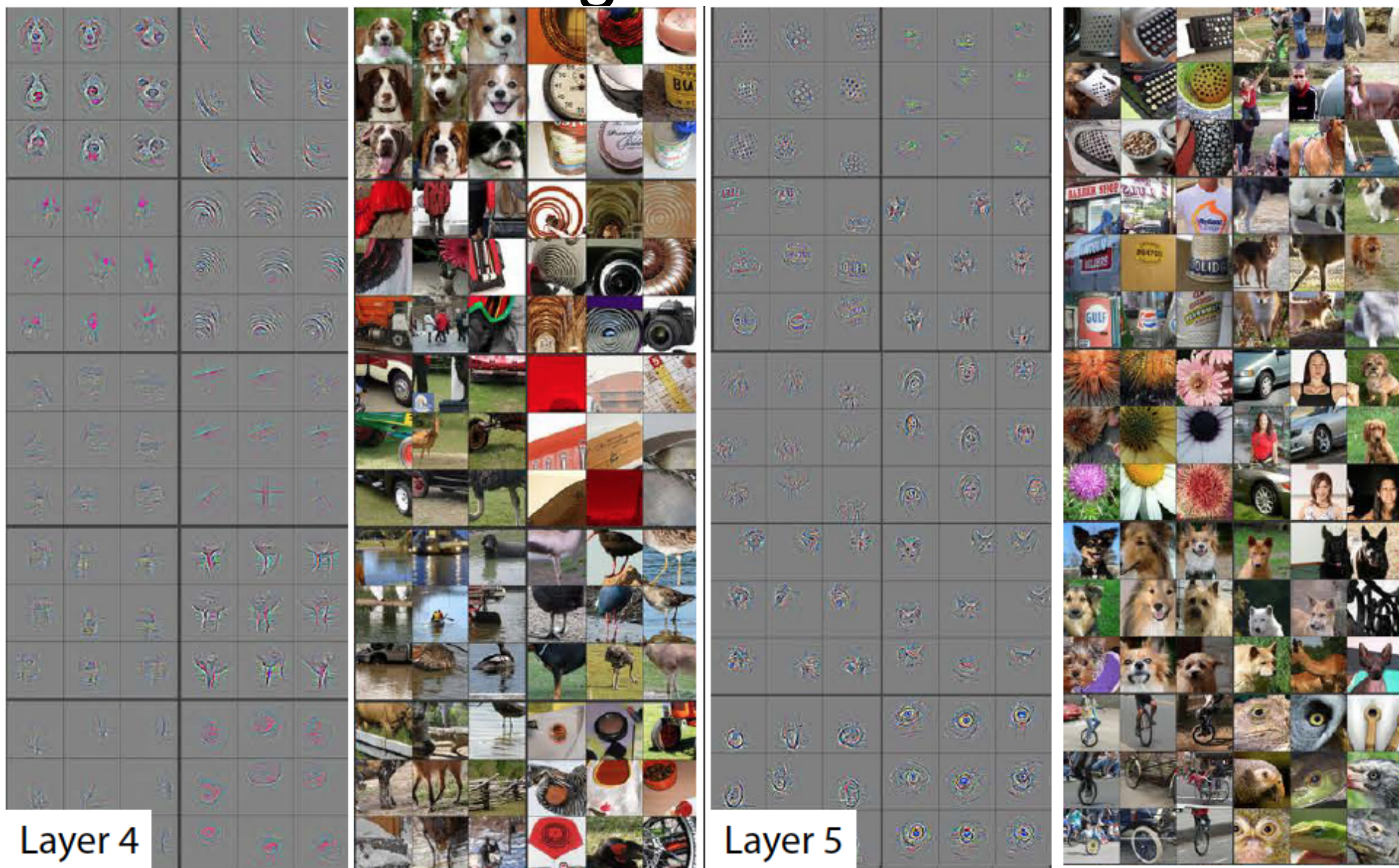
Visualizing Learned Filters



Visualizing Learned Filters

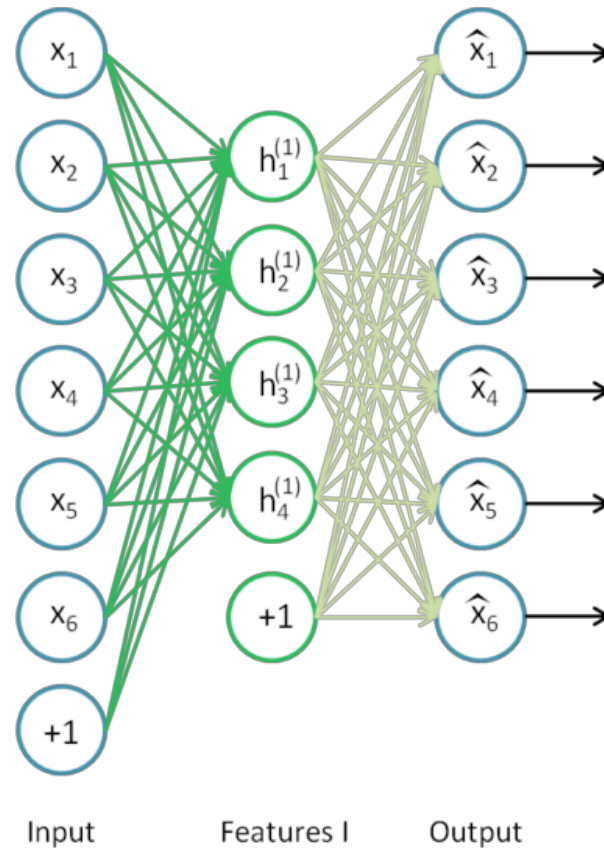


Visualizing Learned Filters

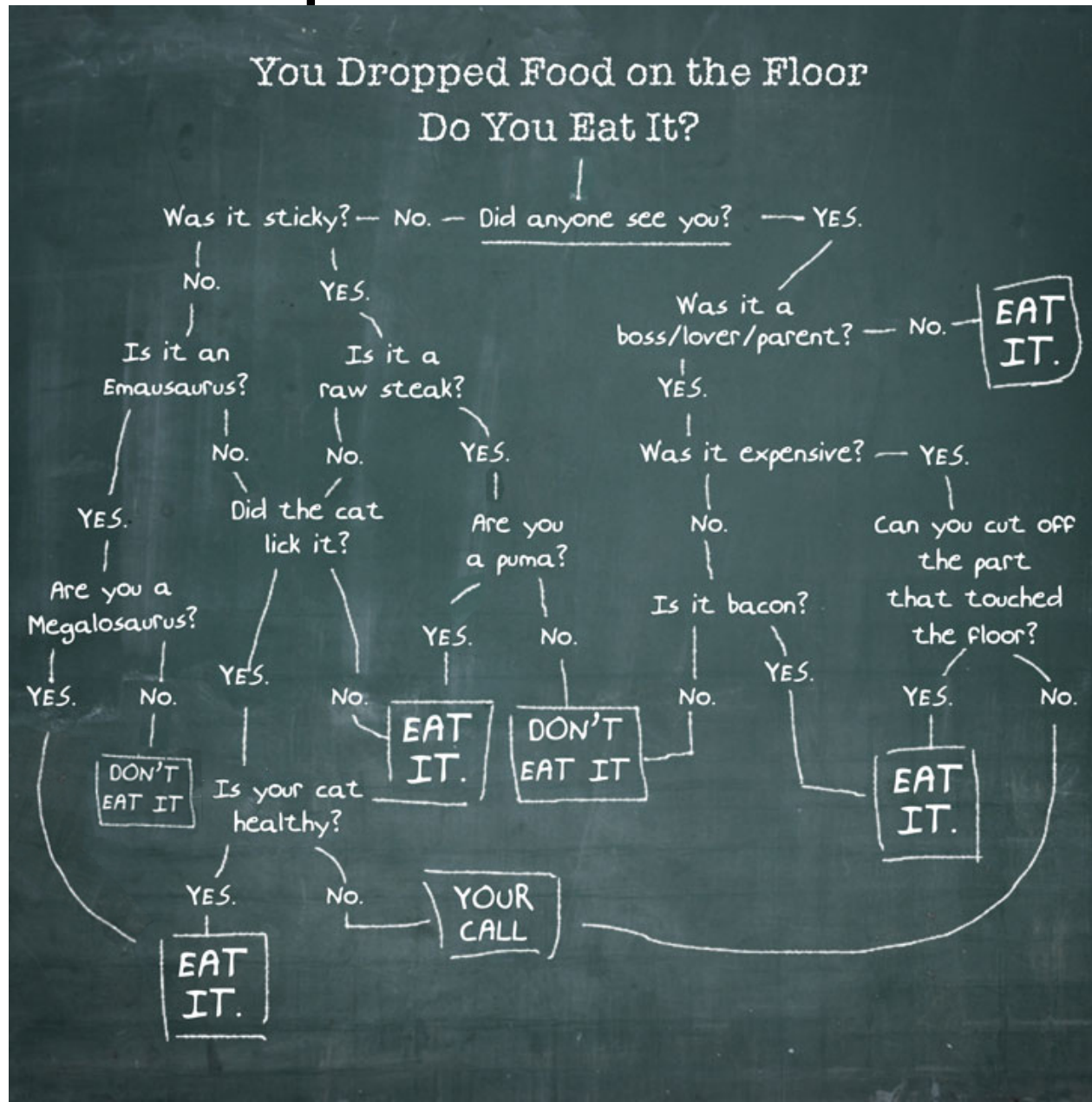


Autoencoders

- Goal
 - Compression: Output tries to predict input



New Topic: Decision Trees



Synonyms

- Decision Trees
- Classification and Regression Trees (CART)
- Algorithms for learning decision trees:
 - ID3
 - C4.5
- Random Forests
 - Multiple decision trees

Decision Trees

- Demo
 - <http://www.cs.technion.ac.il/~rani/LocBoost/>

Pose Estimation

- Random Forests!
 - Multiple decision trees
 - <http://youtu.be/HNkbG3KsY84>



 This image cannot currently be displayed.

 This image cannot currently be displayed.

 This image cannot currently be displayed.

 This image cannot currently be displayed.

A small dataset: Miles Per Gallon

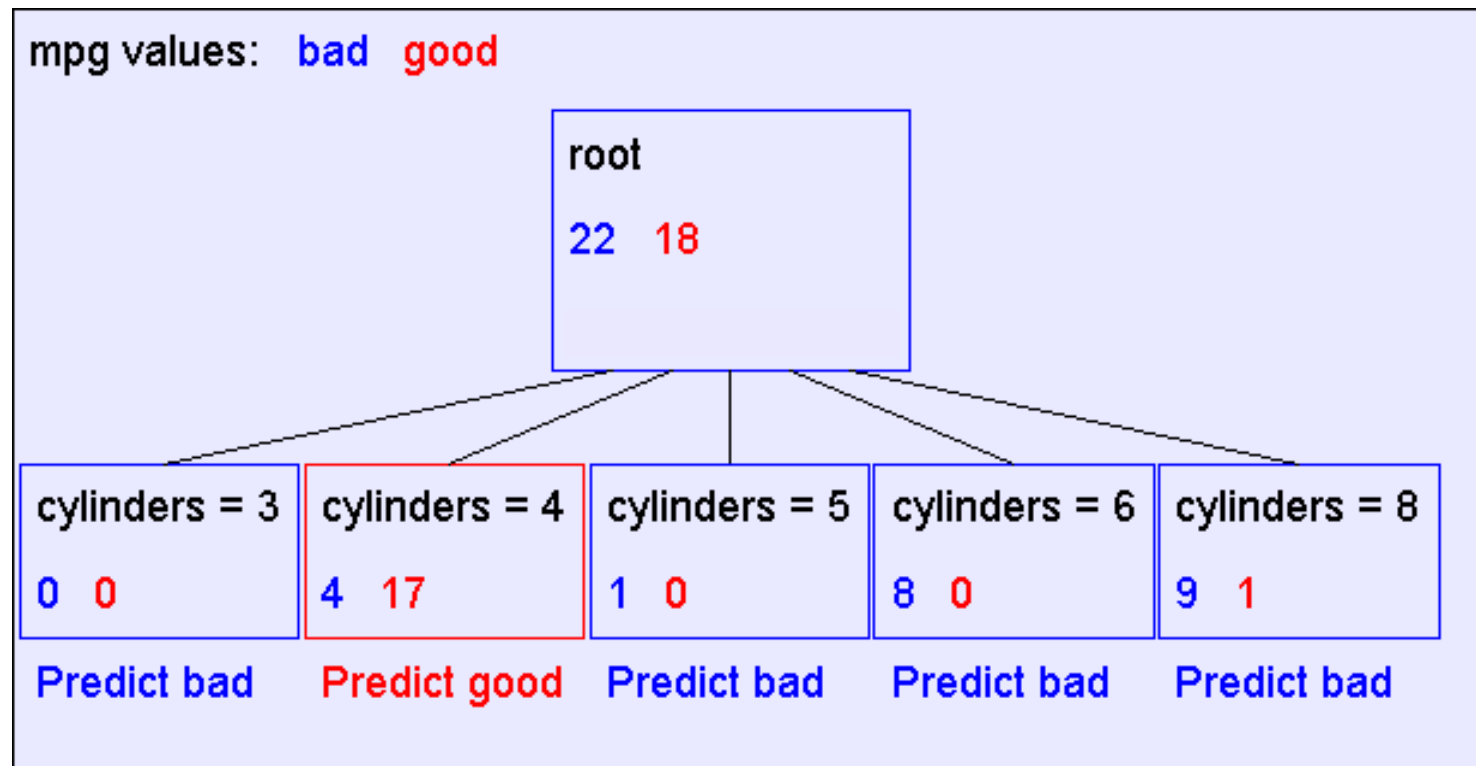
Suppose we want
to predict MPG

mpg	cylinders	displacement	horsepower	weight	acceleration	modelyear	maker
good	4	low	low	low	high	75to78	asia
bad	6	medium	medium	medium	medium	70to74	america
bad	4	medium	medium	medium	low	75to78	europa
bad	8	high	high	high	low	70to74	america
bad	6	medium	medium	medium	medium	70to74	america
bad	4	low	medium	low	medium	70to74	asia
bad	4	low	medium	low	low	70to74	asia
bad	8	high	high	high	low	75to78	america
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
bad	8	high	high	high	low	70to74	america
good	8	high	medium	high	high	79to83	america
bad	8	high	high	high	low	75to78	america
good	4	low	low	low	low	79to83	america
bad	6	medium	medium	medium	high	75to78	america
good	4	medium	low	low	low	79to83	america
good	4	low	low	medium	high	79to83	america
bad	8	high	high	high	low	70to74	america
good	4	low	medium	low	medium	75to78	europa
bad	5	medium	medium	medium	medium	75to78	europa

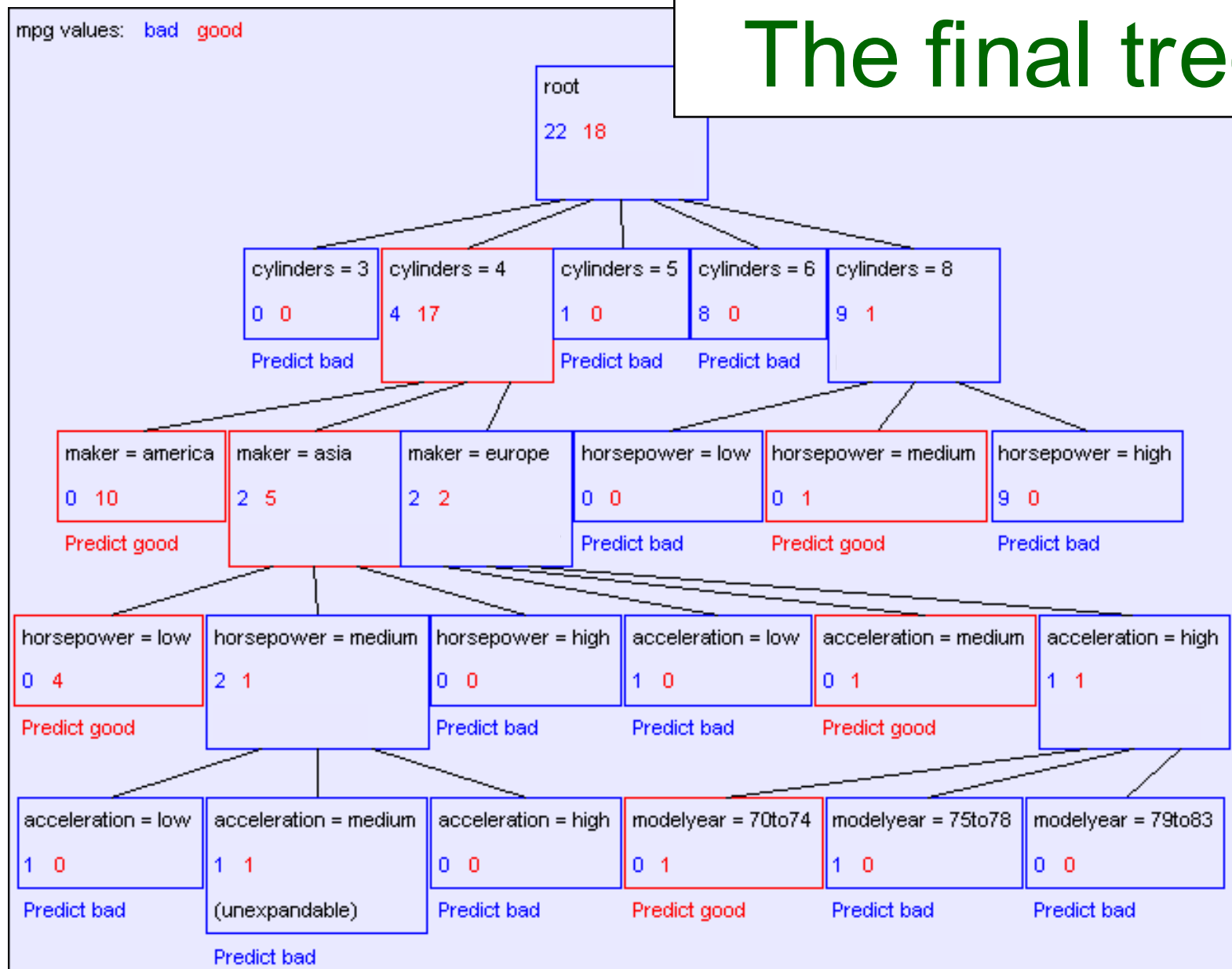
40 Records

From the UCI repository (thanks to Ross Quinlan)

A Decision Stump



The final tree



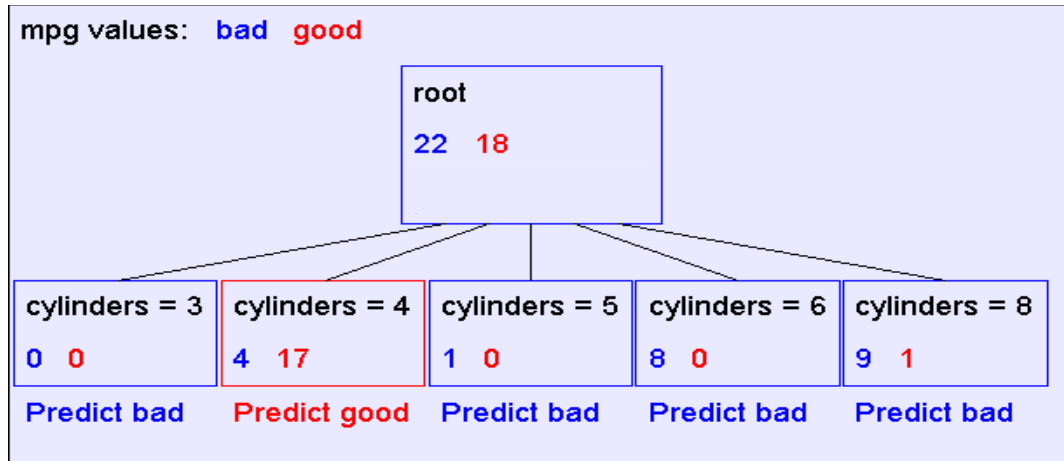
Comments

- Not all features/attributes need to appear in the tree.
- A features/attribute X_i may appear in multiple branches.
- On a path, no feature may appear more than once.
 - Not true for continuous features. We'll see later.
- Many trees can represent the same concept
- But, not all trees will have the same size!
 - e.g., $Y = (A \wedge B) \quad (\neg A \wedge C) \quad (A \text{ and } B) \text{ or } (\text{not } A \text{ and } C)$

Learning decision trees is hard!!!

- Learning the simplest (smallest) decision tree is an NP-complete problem [Hyafil & Rivest '76]
- Resort to a greedy heuristic:
 - Start from empty decision tree
 - Split on **next best attribute (feature)**
 - Recurse
 - “Iterative Dichotomizer” (ID3)
 - C4.5 (ID3+improvements)

Recursion Step



Records
in which
cylinders
= 4

Records
in which
cylinders
= 5

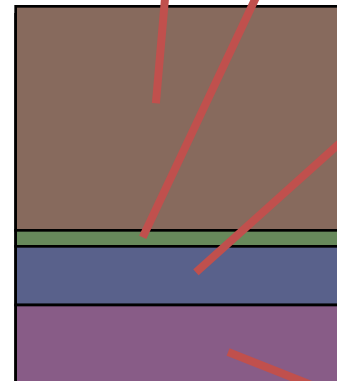
Records
in which
cylinders
= 6

Records
in which
cylinders
= 8

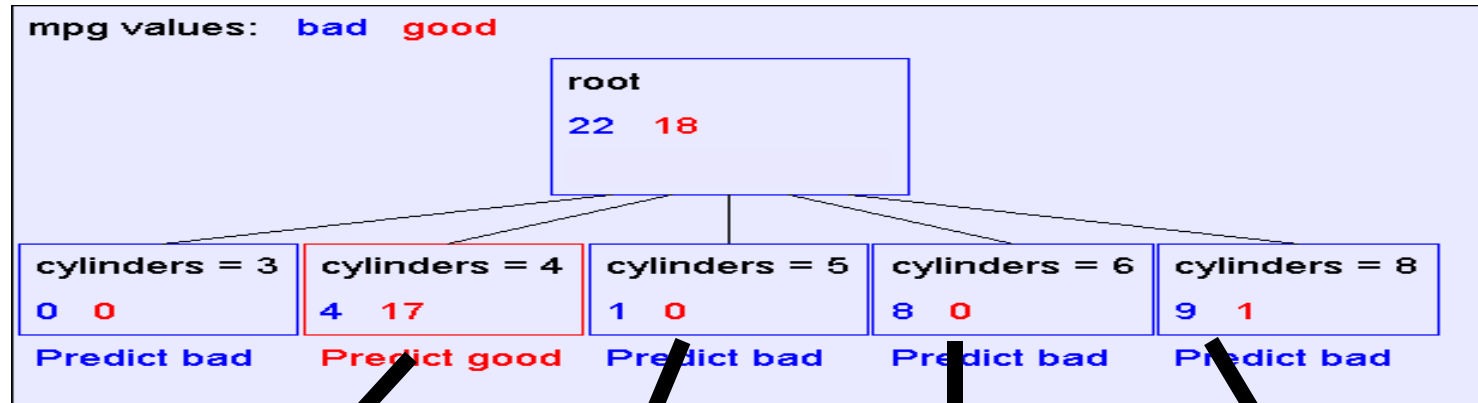
Take the
Original
Dataset..



And partition it
according
to the value of the
attribute we split on



Recursion Step

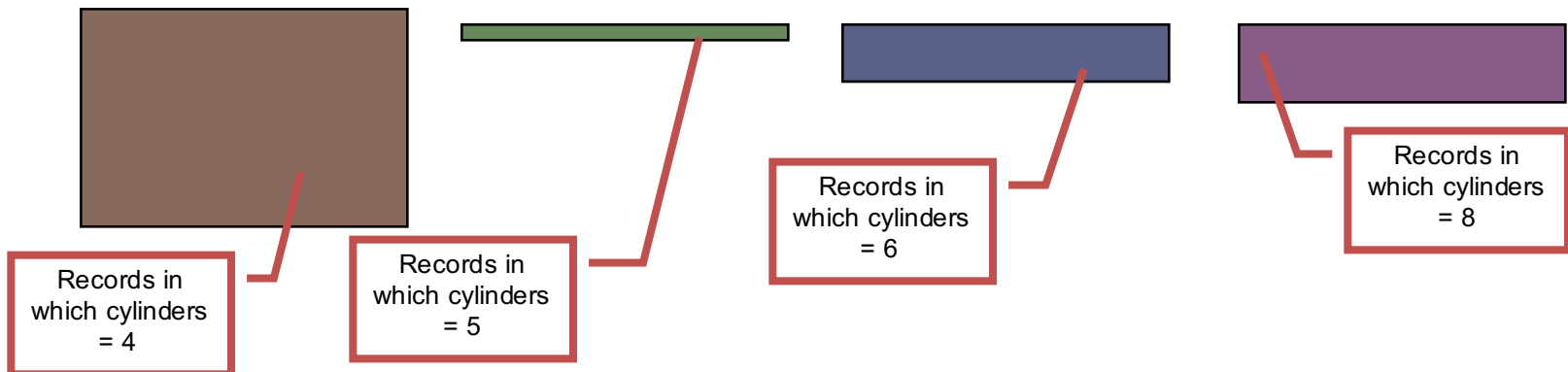


Build tree from
These records..

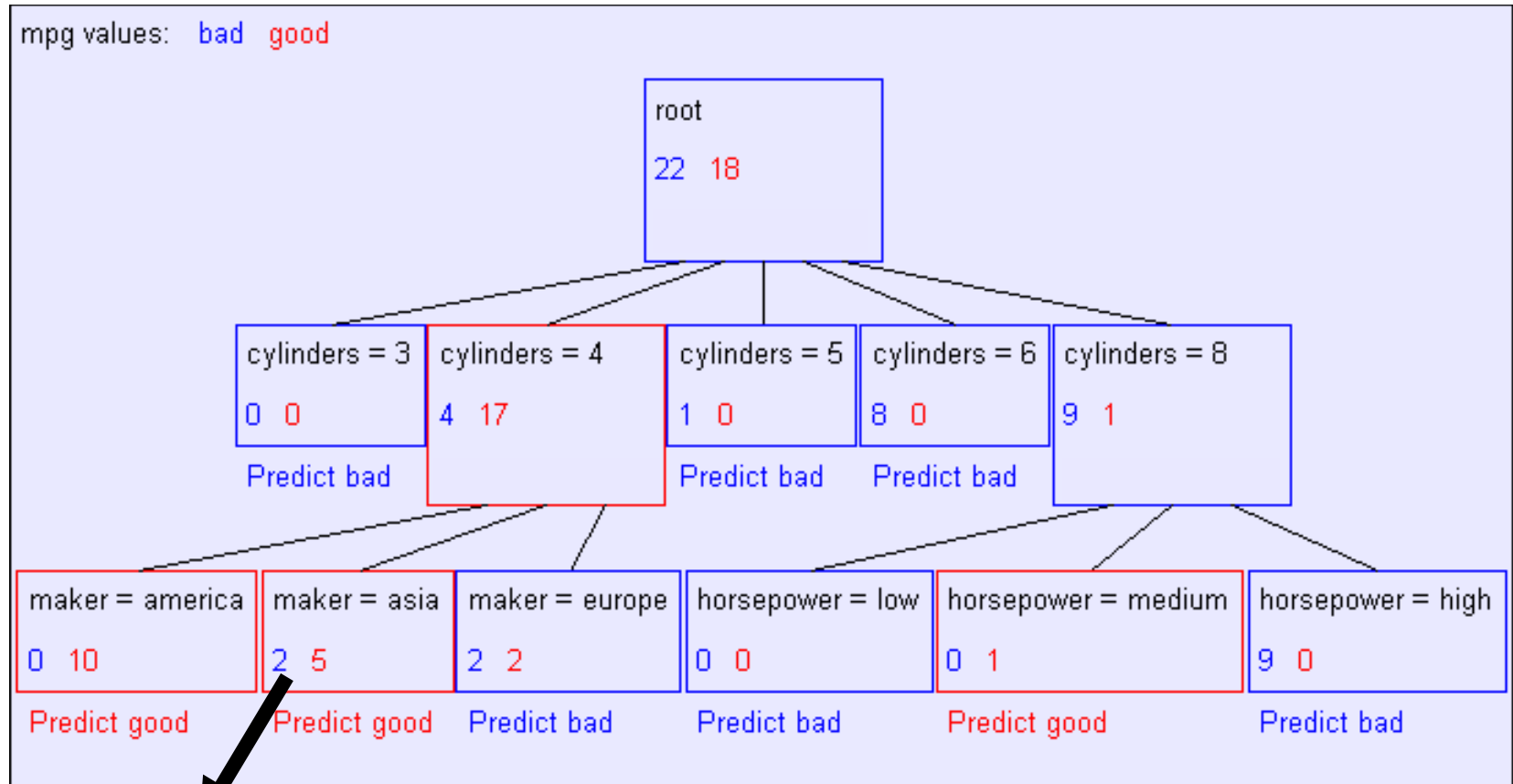
Build tree from
These records..

Build tree from
These records..

Build tree from
These records..



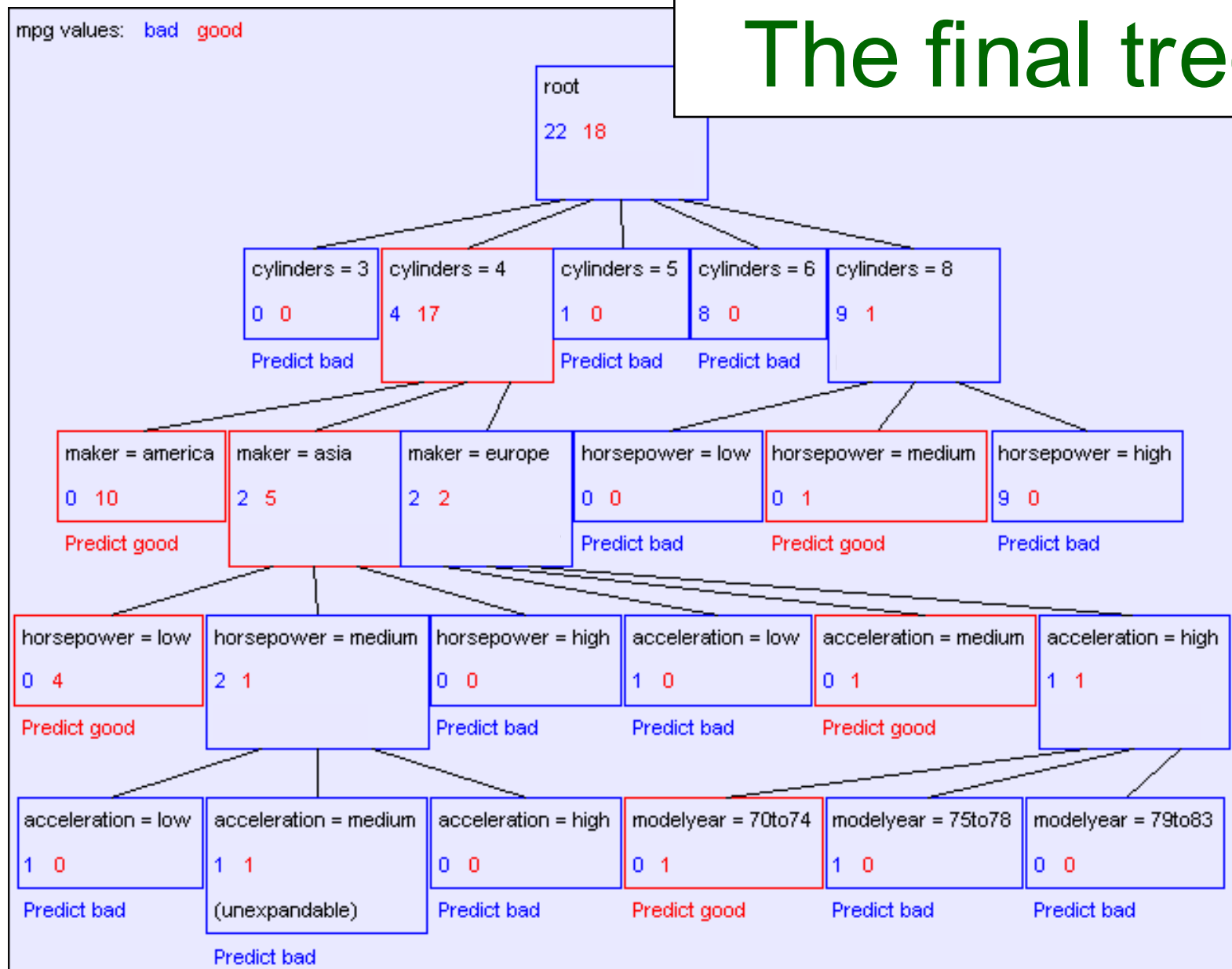
Second level of tree



Recursively build a tree from the seven records in which there are four cylinders and the maker was based in Asia

(Similar recursion in the other cases)

The final tree



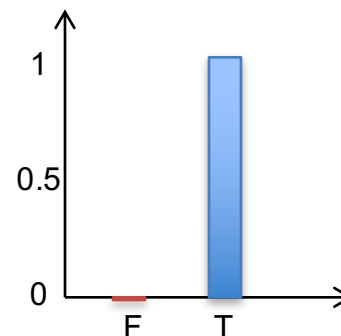
Choosing a good attribute

X_1	X_2	Y
T	T	T
T	F	T
T	T	T
T	F	T
F	T	T
F	F	F
F	T	F
F	F	F

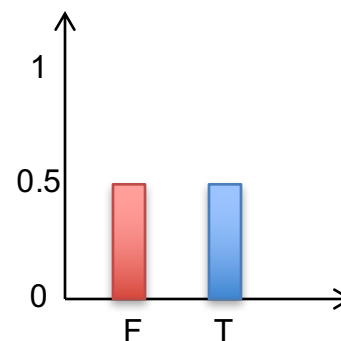
Measuring uncertainty

- Good split if we are more certain about classification after split
 - Deterministic good (all true or all false)
 - Uniform distribution bad

$P(Y=F \mid X_1=T) =$ 0	$P(Y=T \mid X_1=T) =$ 1
----------------------------	----------------------------



$P(Y=F \mid X_2=F) =$ 1/2	$P(Y=T \mid X_2=F) =$ 1/2
------------------------------	------------------------------



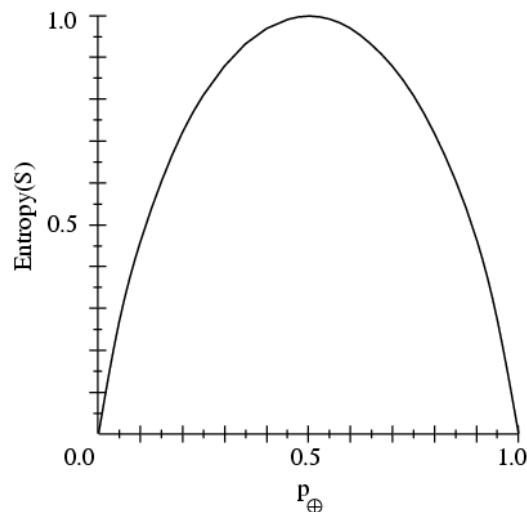
Entropy

Entropy $H(X)$ of a random variable Y

$$H(Y) = - \sum_{i=1}^k P(Y = y_i) \log_2 P(Y = y_i)$$

More uncertainty, more entropy!

Information Theory interpretation: $H(Y)$ is the expected number of bits needed to encode a randomly drawn value of Y (under most efficient code)



Information gain

- Advantage of attribute – decrease in uncertainty
 - Entropy of Y before you split
 - Entropy after split
 - Weight by probability of following each branch, i.e., normalized number of records

$$H(Y | X) = - \sum_{j=1}^v P(X = x_j) \sum_{i=1}^k P(Y = y_i | X = x_j) \log_2 P(Y = y_i | X = x_j)$$

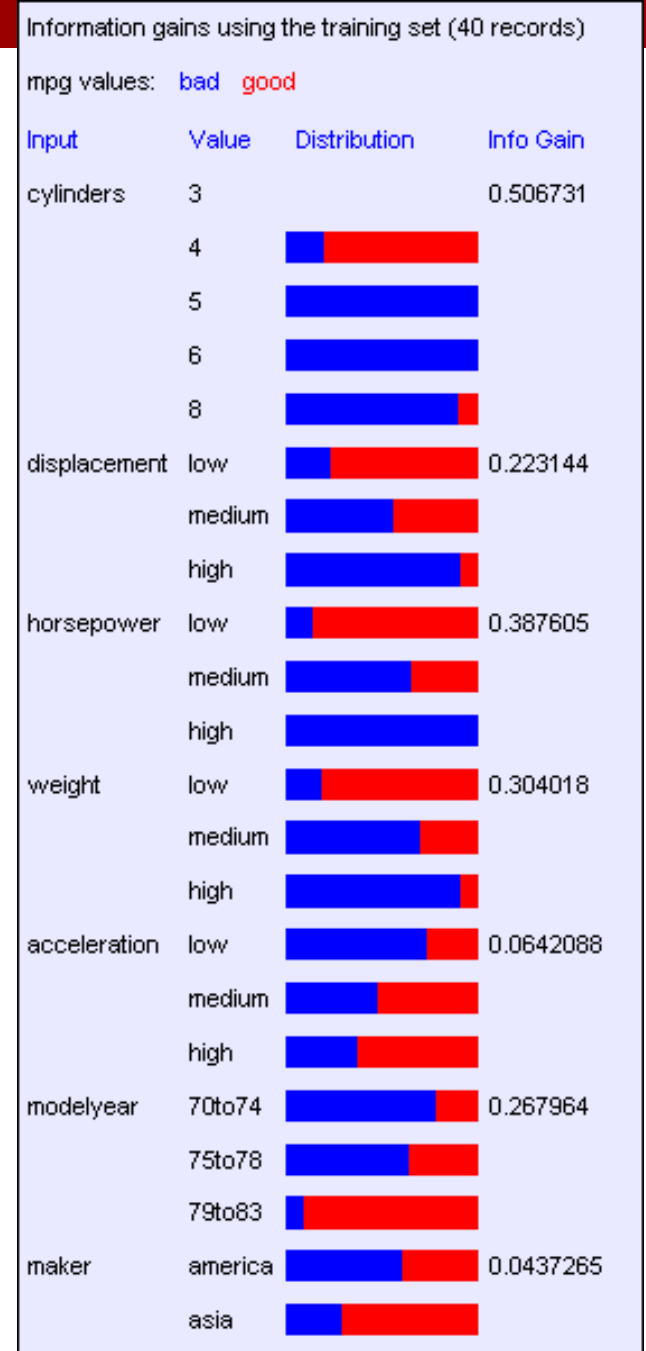
- Information gain is difference $IG(X) = H(Y) - H(Y | X)$
 - (Technically it's mutual information; but in this context also referred to as information gain)

Learning decision trees

- Start from empty decision tree
- Split on **next best attribute (feature)**
 - Use, for example, information gain to select attribute
 - Split on $\arg \max_i IG(X_i) = \arg \max_i H(Y) - H(Y | X_i)$
- Recurse

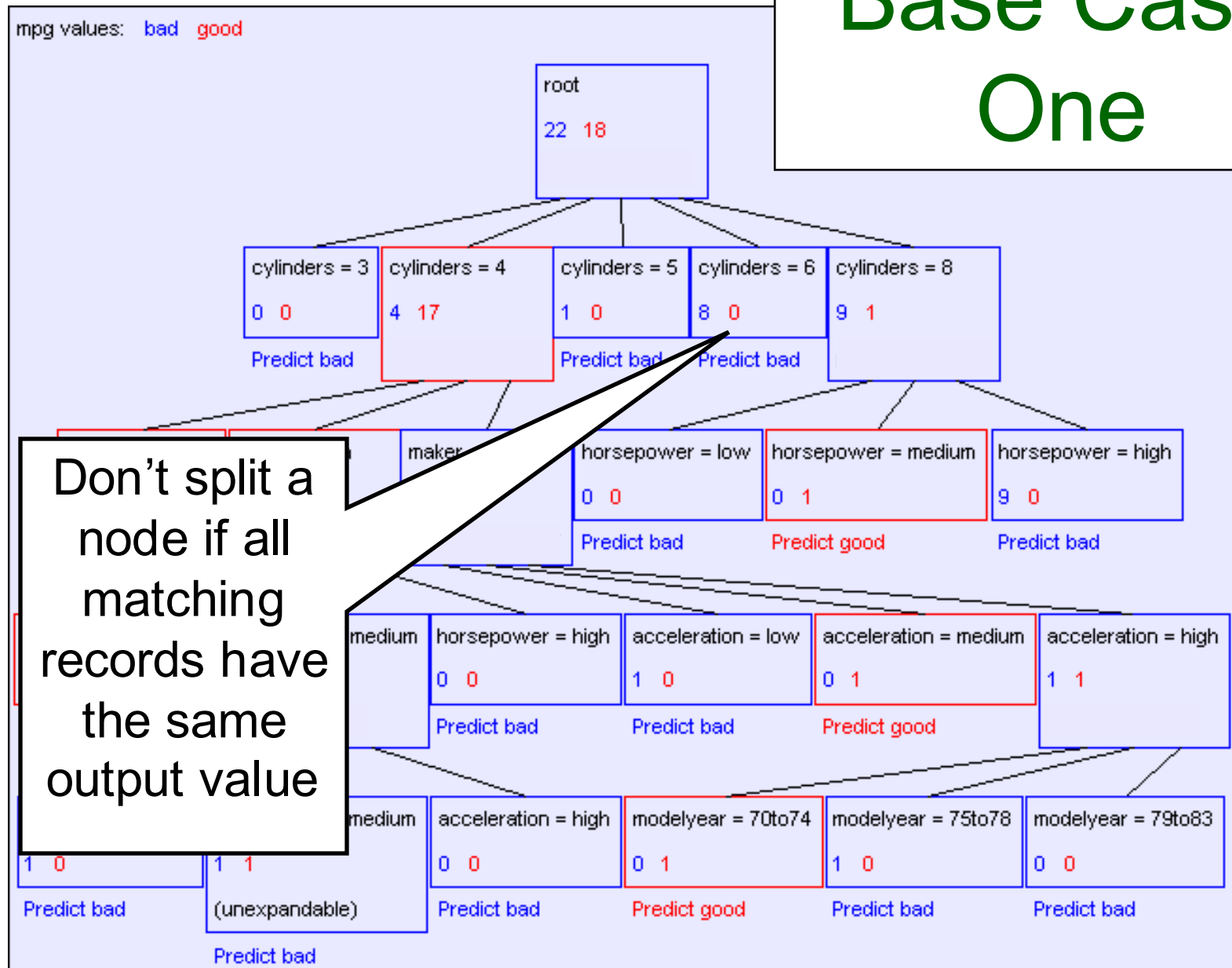
Suppose we want
to predict MPG

Look at all the
information
gains...

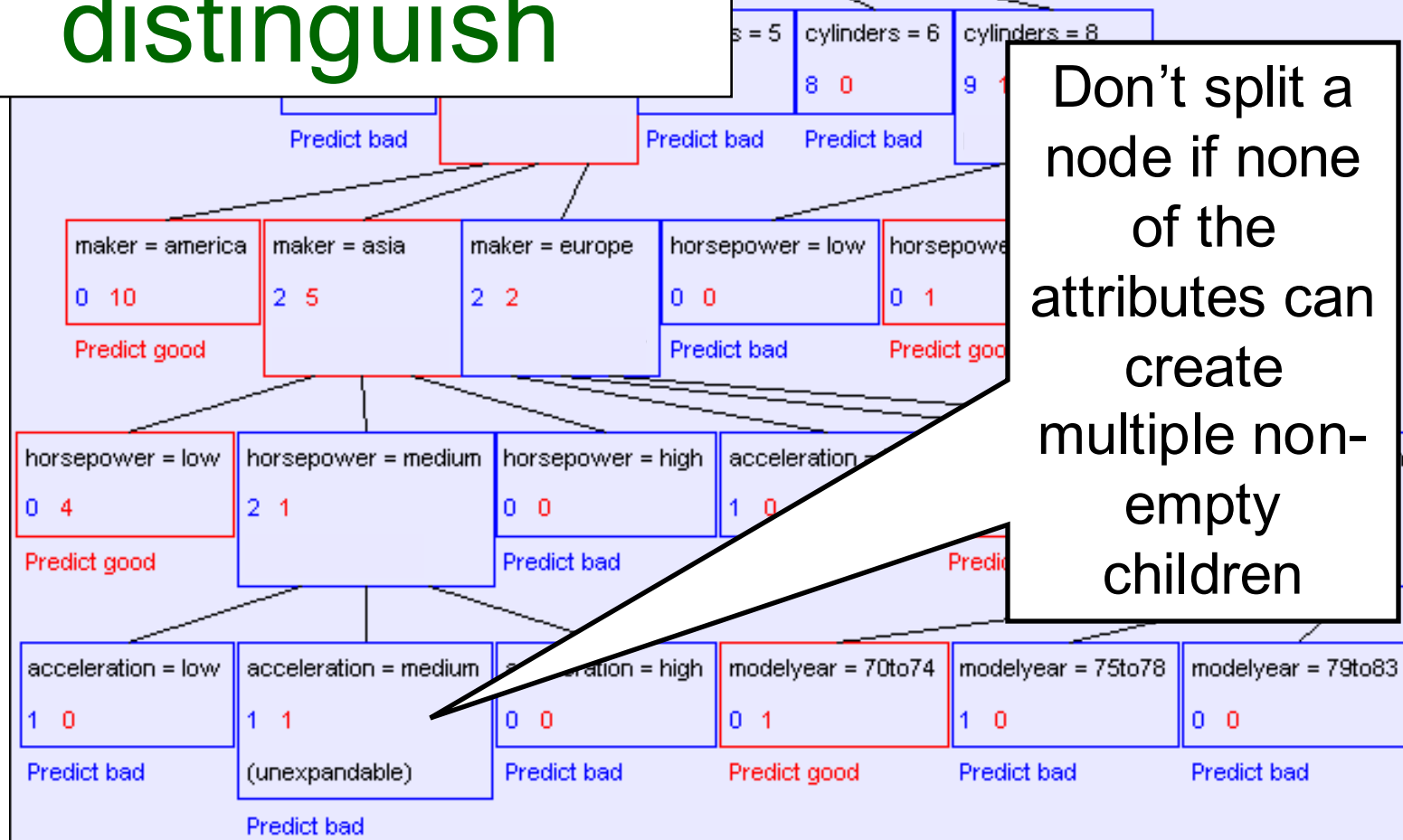


When do we stop?

Base Case One



Base Case Two: No attributes can distinguish

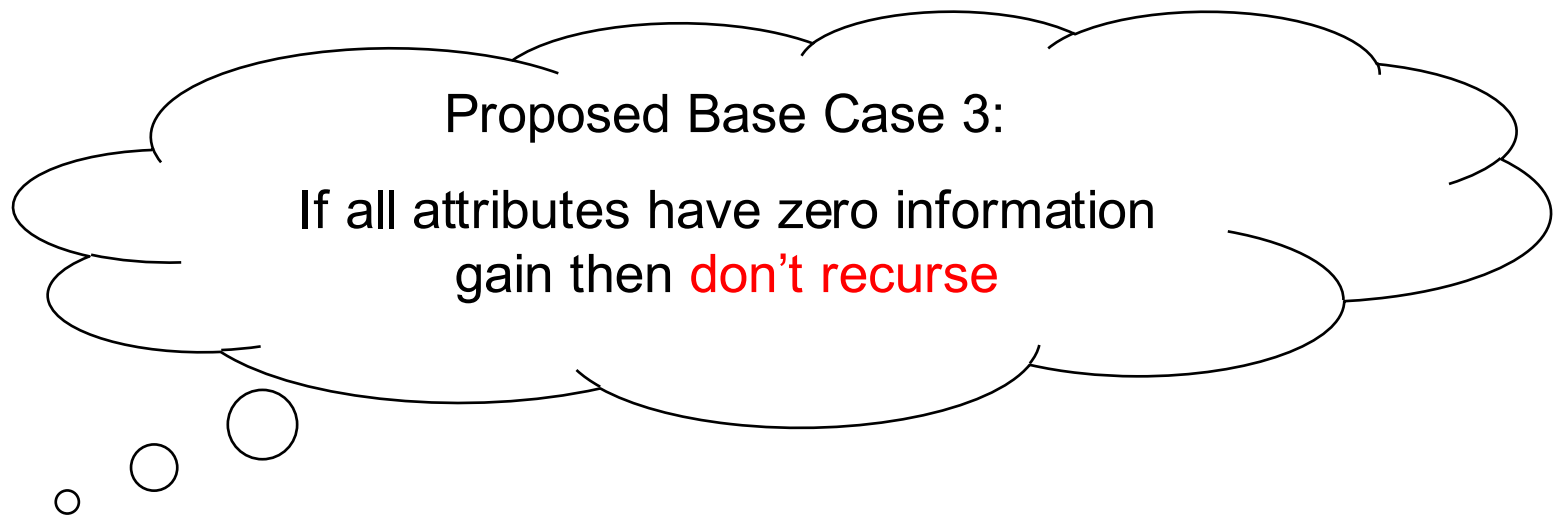


Base Cases

- Base Case One: If all records in current data subset have the same output then **don't recurse**
- Base Case Two: If all records have exactly the same set of input attributes then **don't recurse**

Base Cases: An idea

- Base Case One: If all records in current data subset have the same output then **don't recurse**
- Base Case Two: If all records have exactly the same set of input attributes then **don't recurse**



• *Is this a good idea?*

The problem with Base Case 3





a	b	y
0	0	0
0	1	1
1	0	1
1	1	0

$$y = a \text{ XOR } b$$

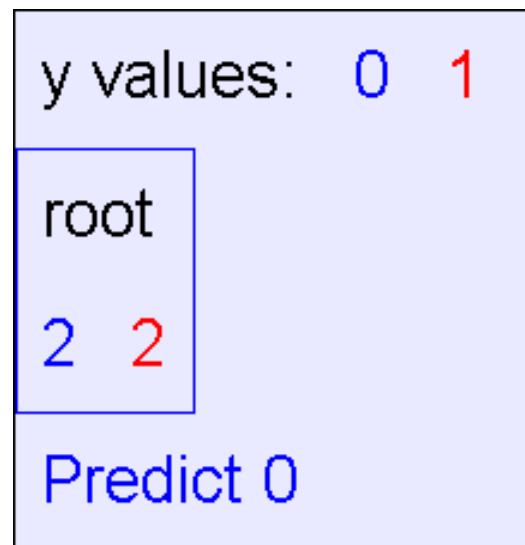
The information gains:

Information gains using the training set (4 records)

y values: 0 1

Input	Value	Distribution	Info Gain
a	0		0
	1		
b	0		0
	1		

The resulting decision tree:



If we omit Base Case 3:

a	b	y
0	0	0
0	1	1
1	0	1
1	1	0

$$y = a \text{ XOR } b$$

The resulting decision tree:

