ECE 5984: Introduction to Machine Learning

Topics:

- Decision/Classification Trees

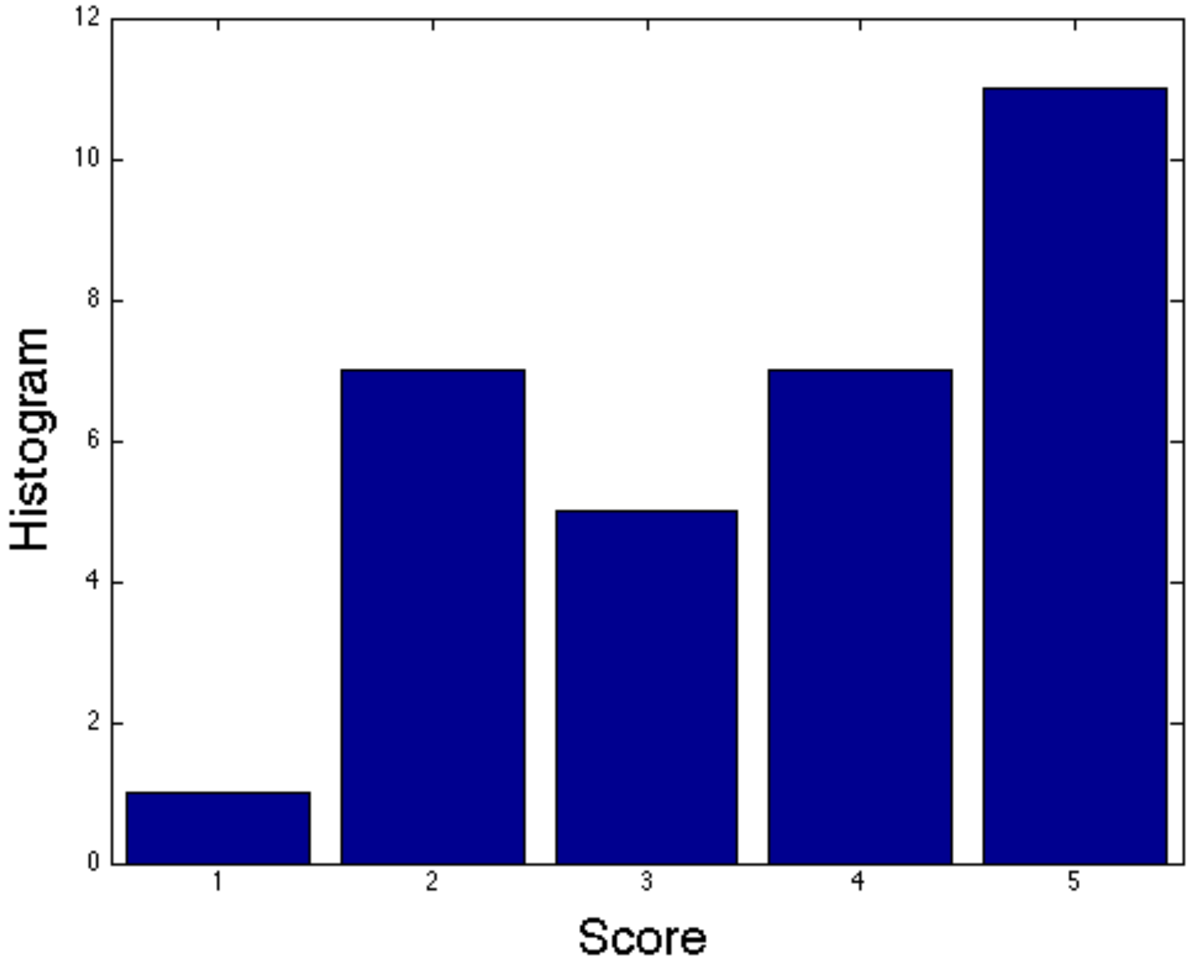
Readings: Murphy 16.1-16.2; Hastie 9.2

Dhruv Batra

Virginia Tech

Project Proposals Graded

- Mean 3.6/5 = 72%



Administrativia

- Project Mid-Sem Spotlight Presentations
  - Friday: ~~5-7pm~~, 3-5pm Whittemore ~~654~~ 457A
  - 5 slides (recommended)
  - 4 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

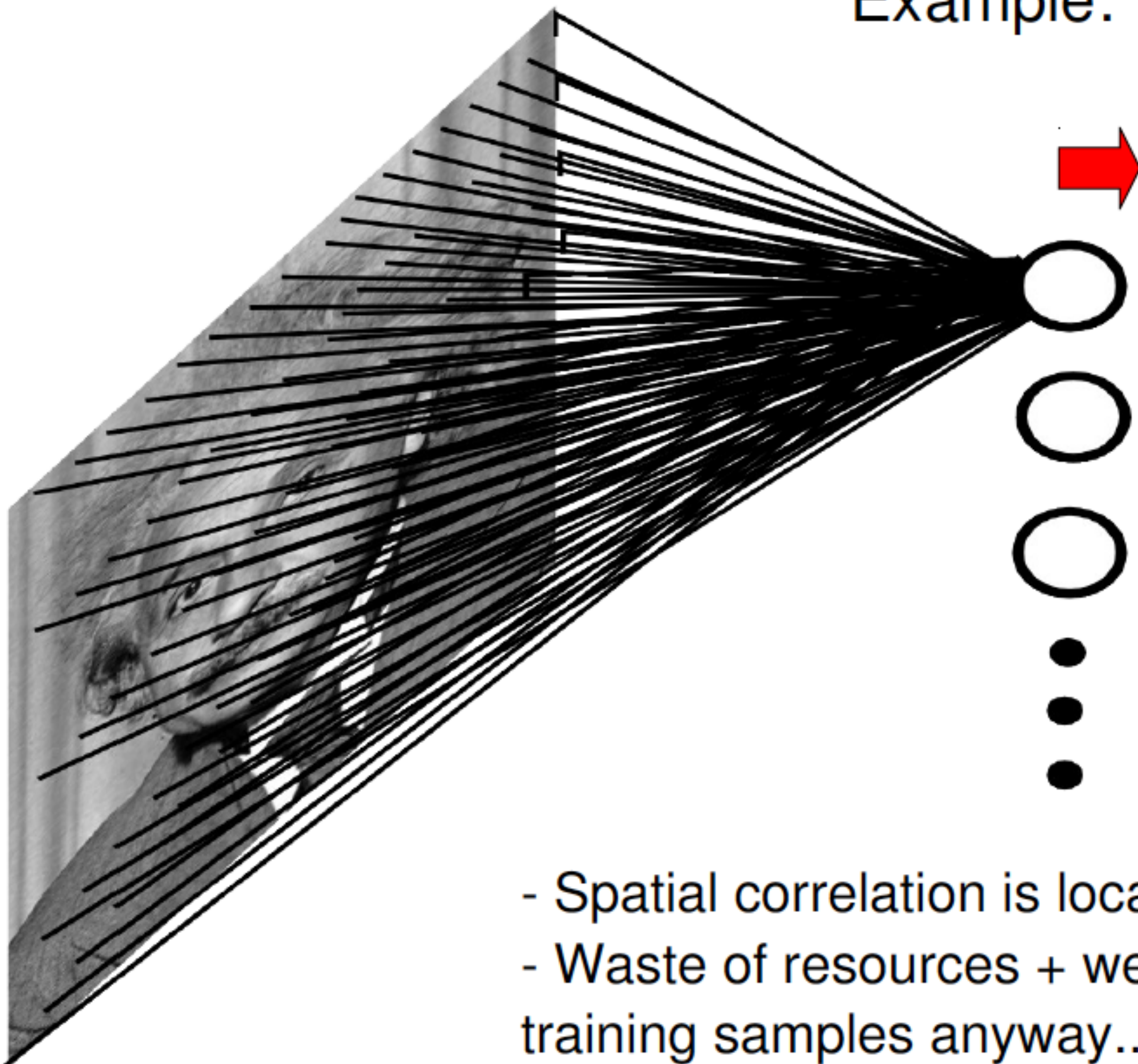
Recap of Last Time

Convolution Explained

- http://setosa.io/ev/image-kernels/

- https://github.com/bruckner/deepViz

Fully Connected Layer



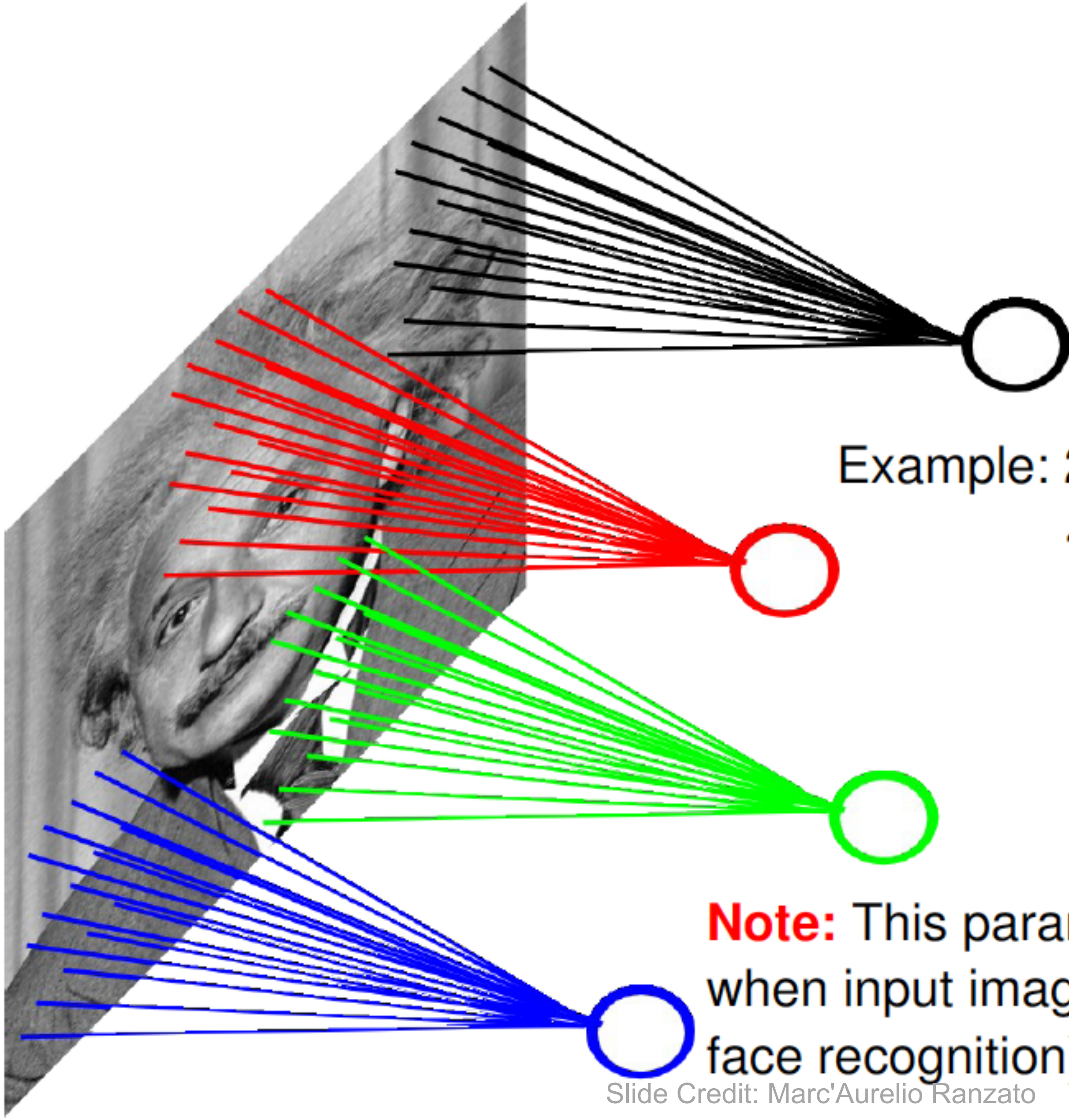
Example: 200x200 image
40K hidden units
~2B parameters!!!

- Spatial correlation is local
- Waste of resources + we have not enough training samples anyway..

Ranzato

Slide Credit: Marc'Aurelio Ranzato

Locally Connected Layer



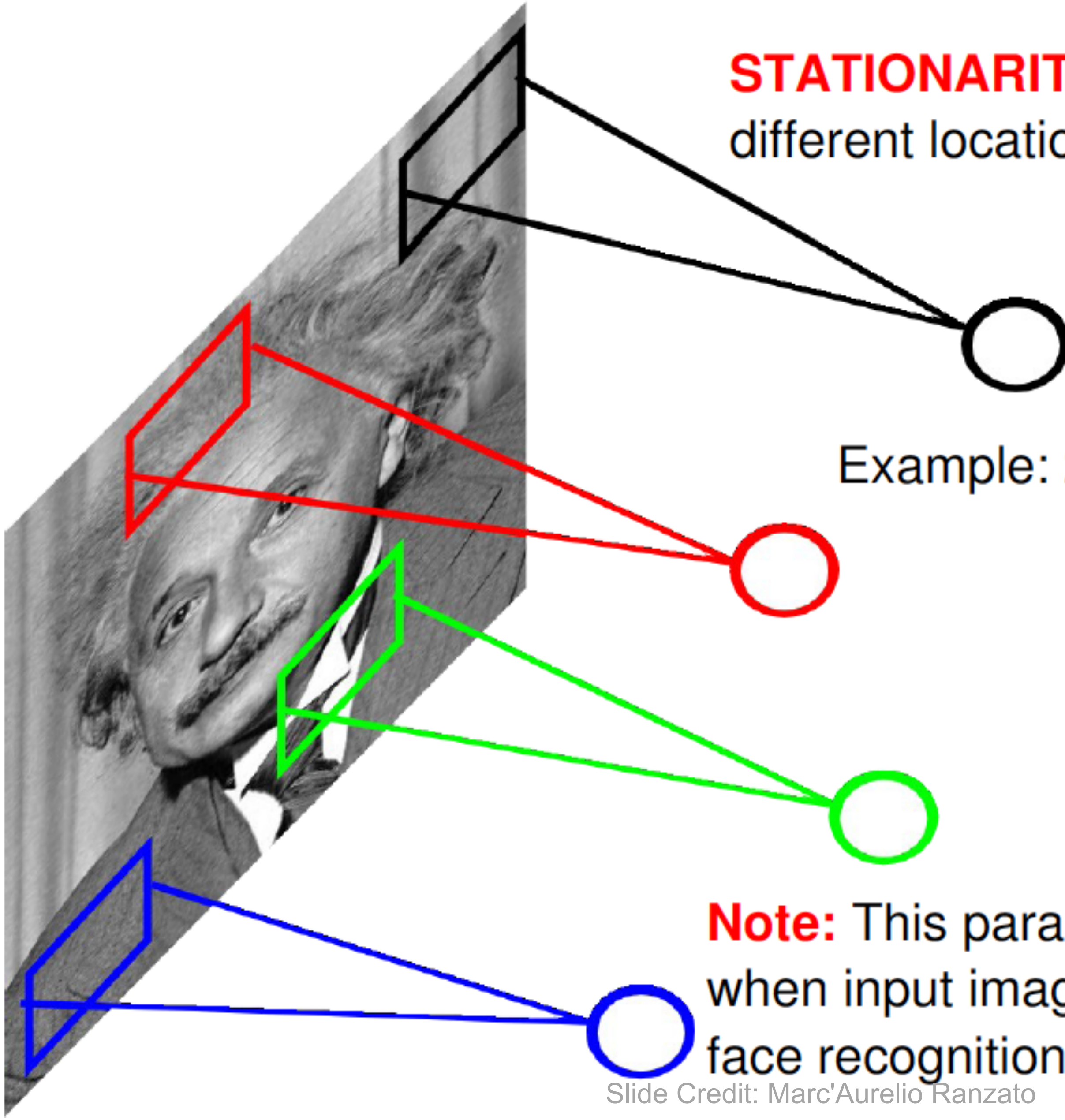
Example: 200x200 image
40K hidden units
Filter size: 10x10
4M parameters

Note: This parameterization is good when input image is registered (e.g., face recognition).

Ranzato

Slide Credit: Marc'Aurelio Ranzato

Locally Connected Layer



STATIONARITY? Statistics is similar at different locations

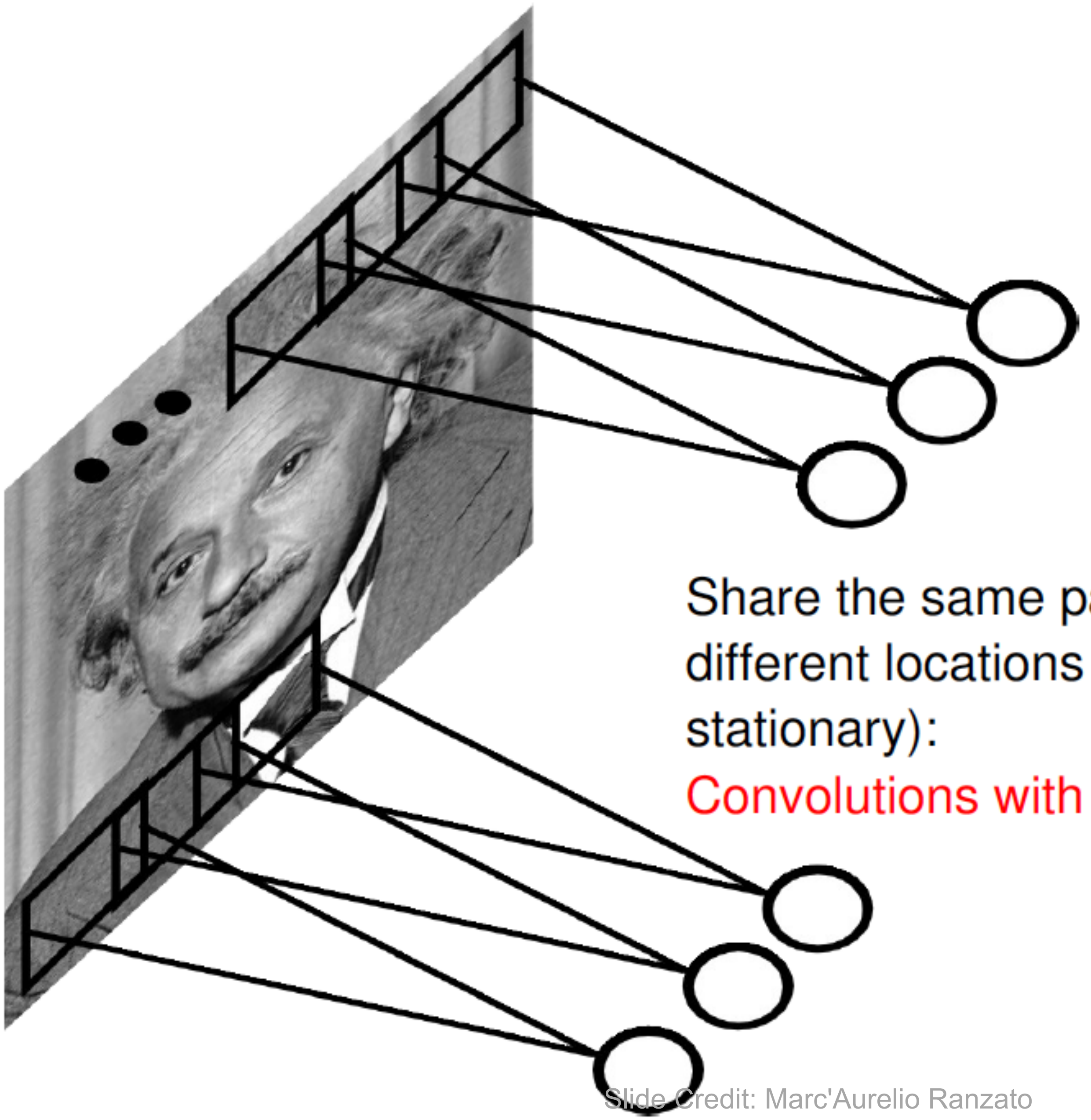
Example: 200x200 image
40K hidden units
Filter size: 10x10
4M parameters

Note: This parameterization is good when input image is registered (e.g., face recognition).

Slide Credit: Marc'Aurelio Ranzato

Ranzato

Convolutional Layer

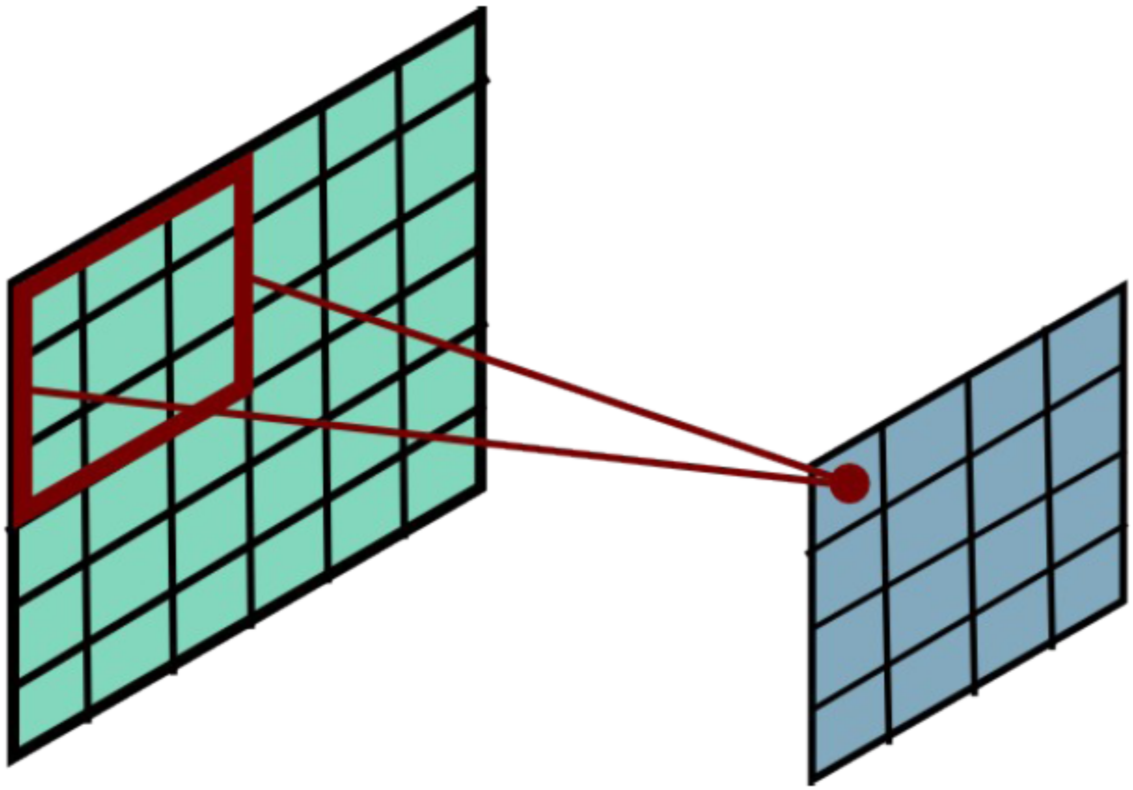


Share the same parameters across different locations (assuming input is stationary):

Convolutions with learned kernels

Slide Credit: Marc'Aurelio Ranzato

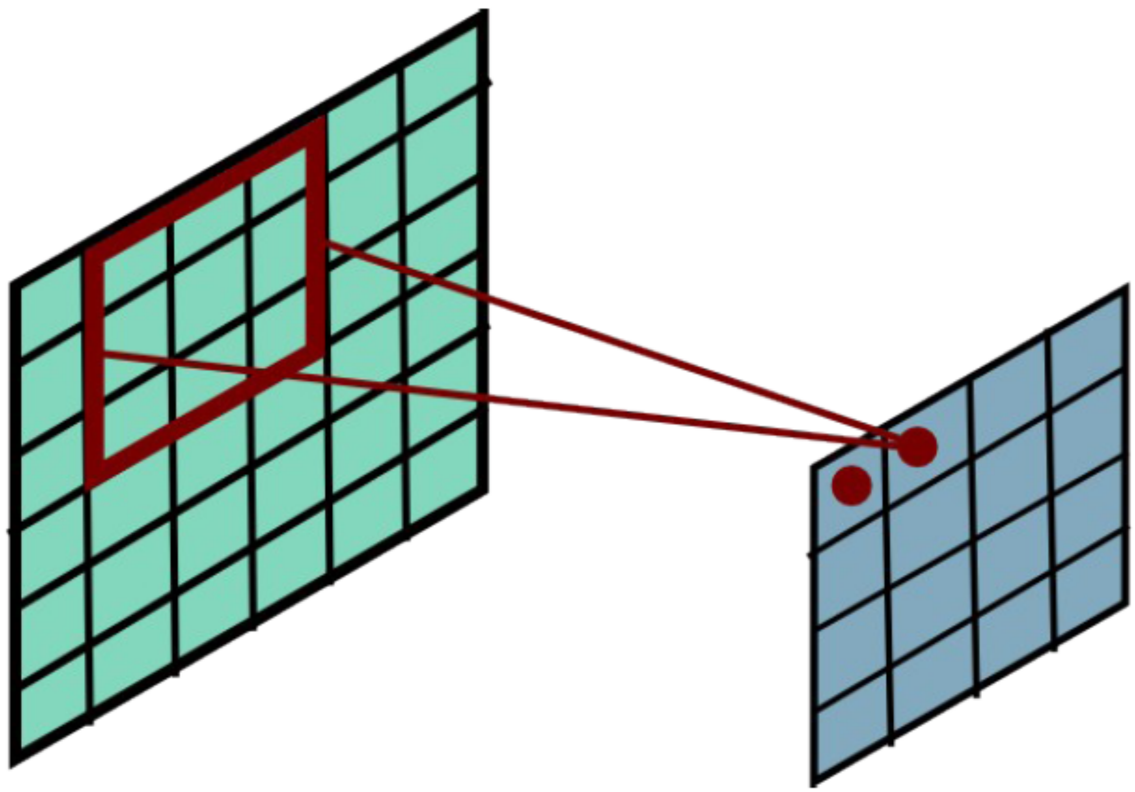
Convolutional Layer



Slide Credit: Marc'Aurelio Ranzato

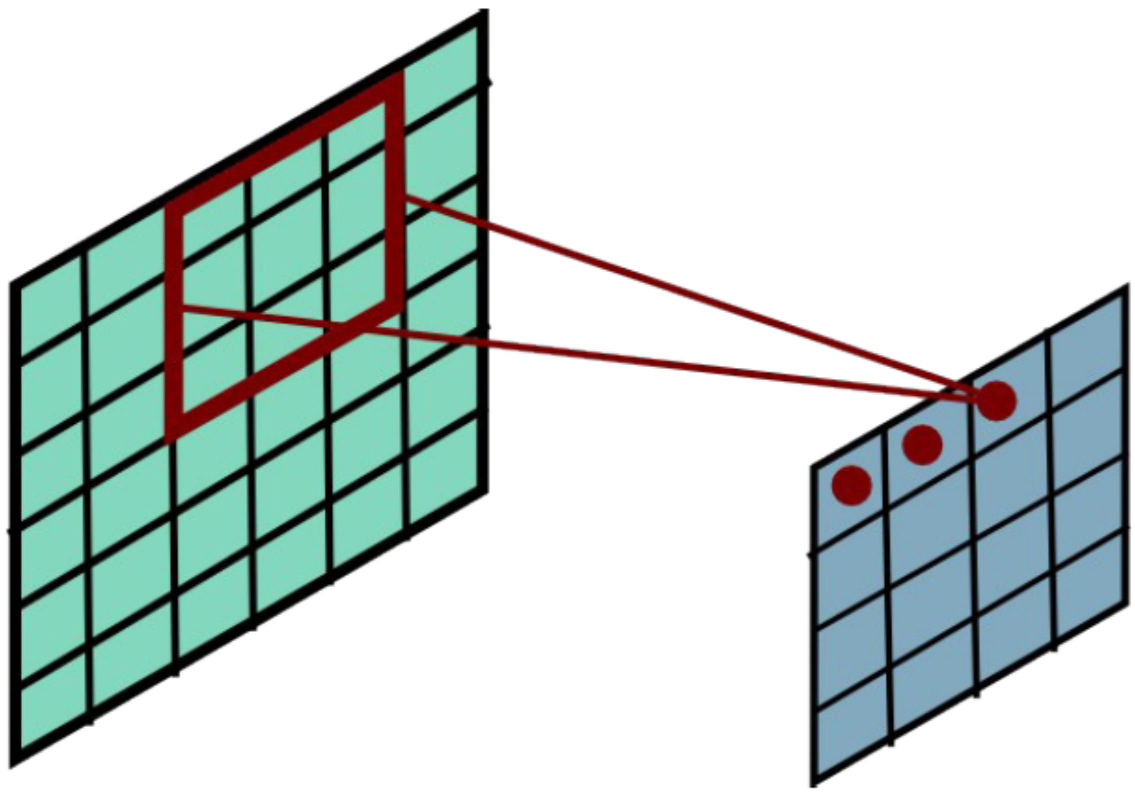
Ranzato

Convolutional Layer



Slide Credit: Marc'Aurelio Ranzato

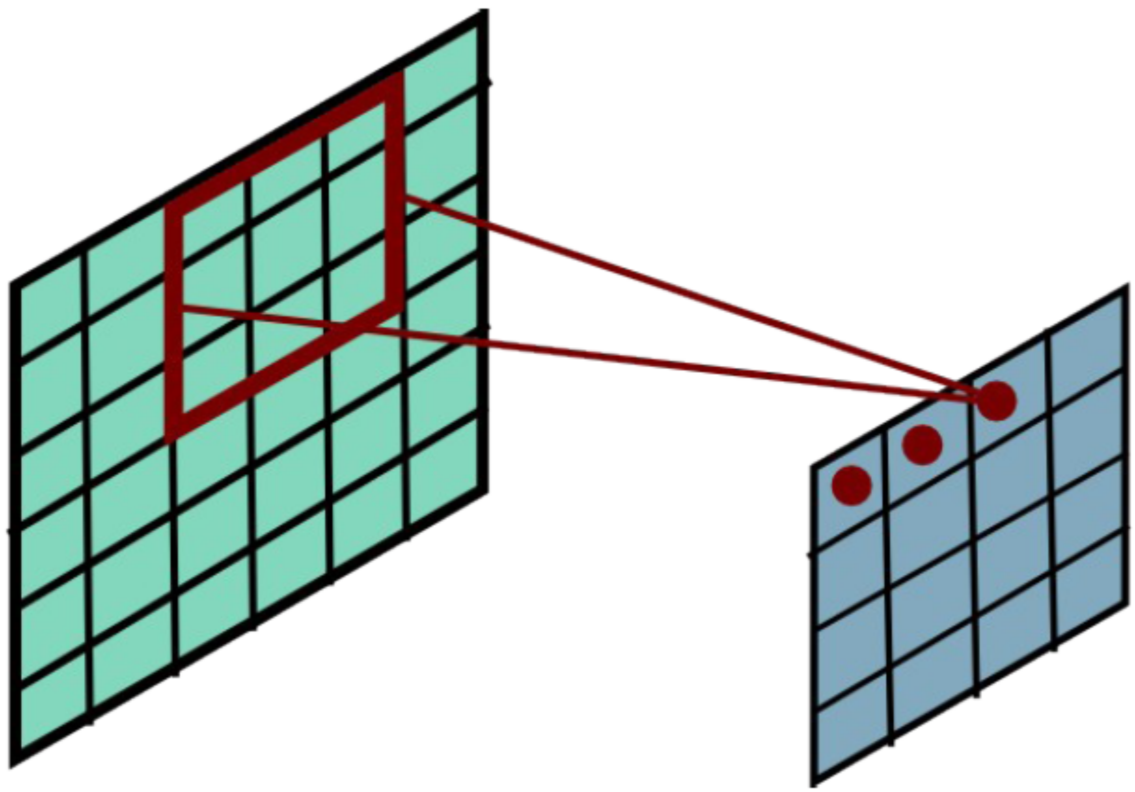
Convolutional Layer



Slide Credit: Marc'Aurelio Ranzato

Ranzato

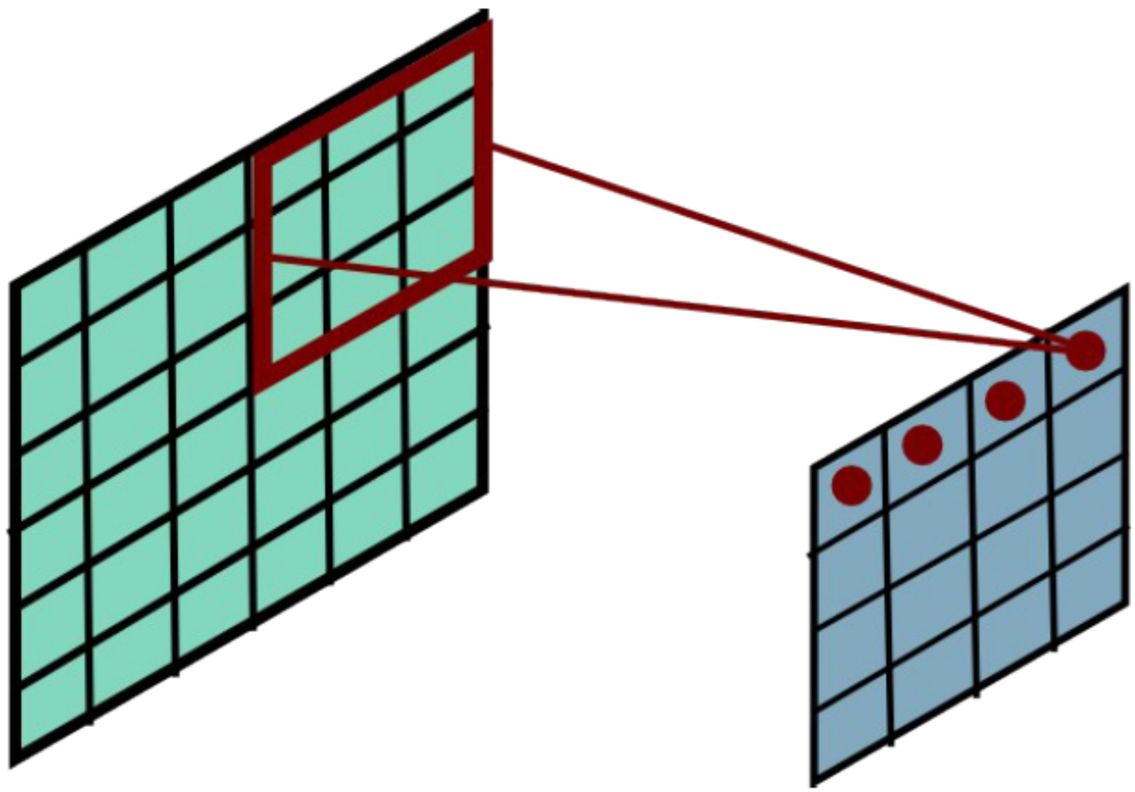
Convolutional Layer



Slide Credit: Marc'Aurelio Ranzato

Ranzato

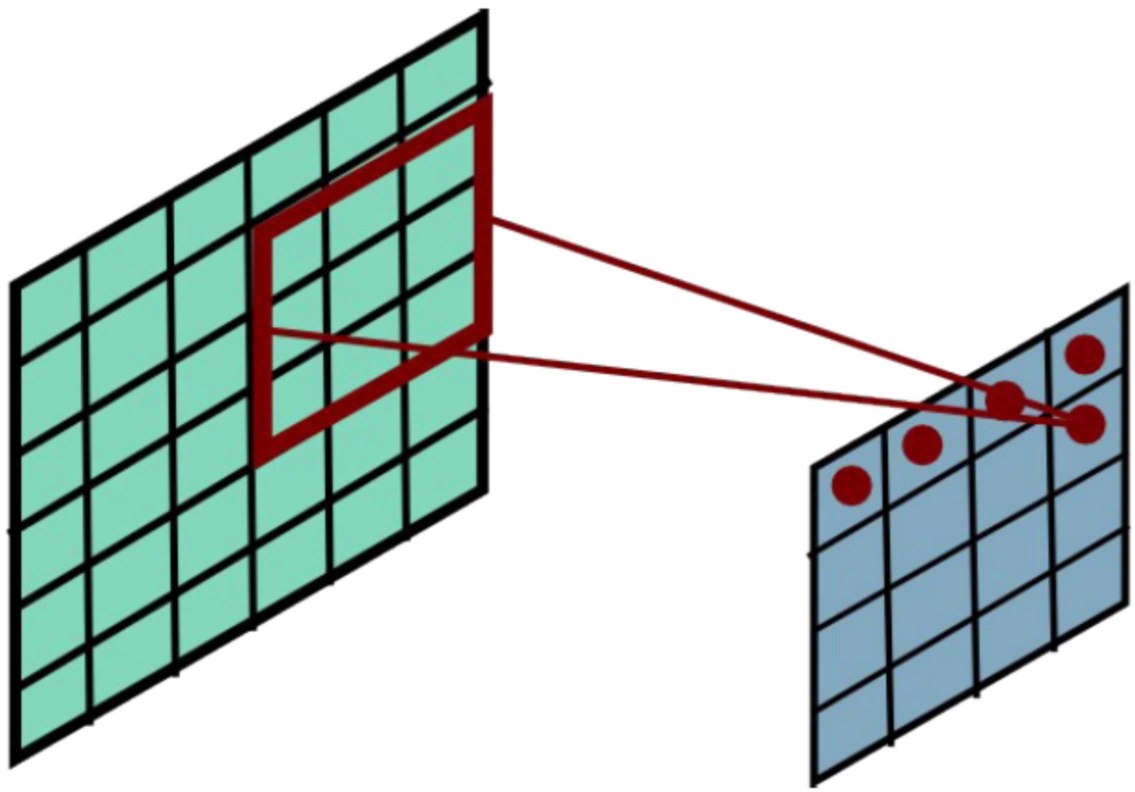
Convolutional Layer



Slide Credit: Marc'Aurelio Ranzato

Ranzato

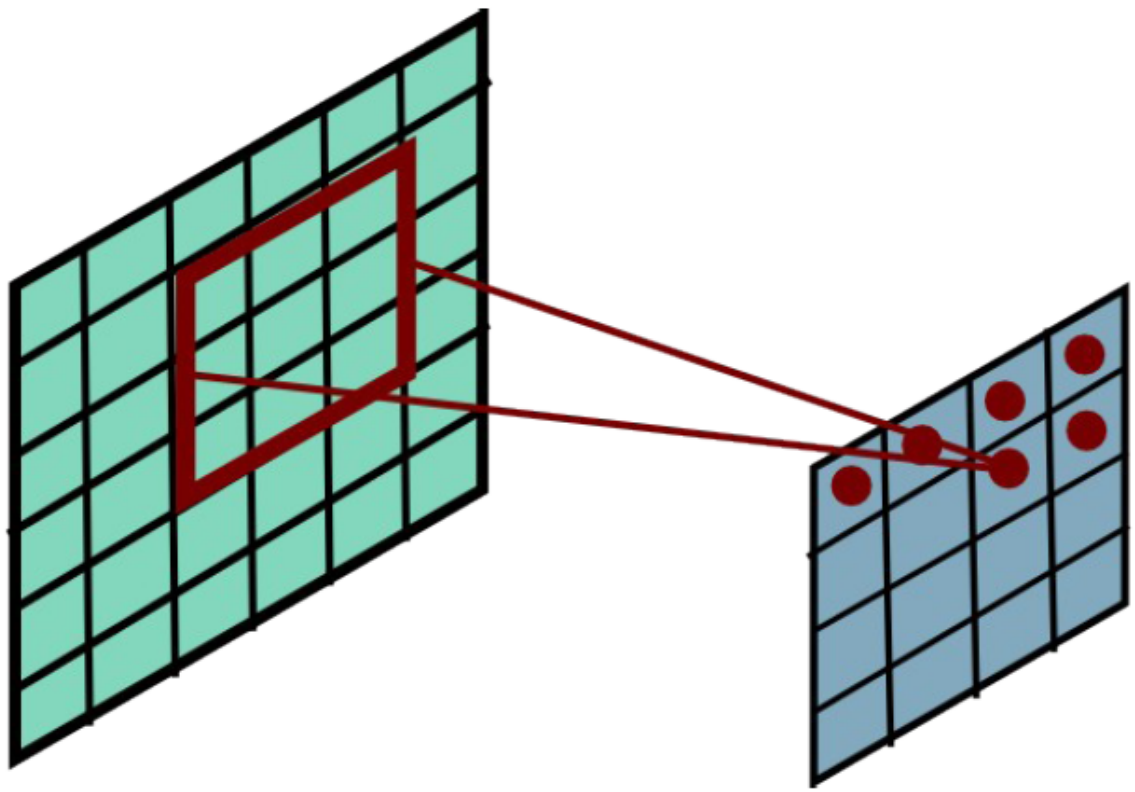
Convolutional Layer



Slide Credit: Marc'Aurelio Ranzato

Ranzato

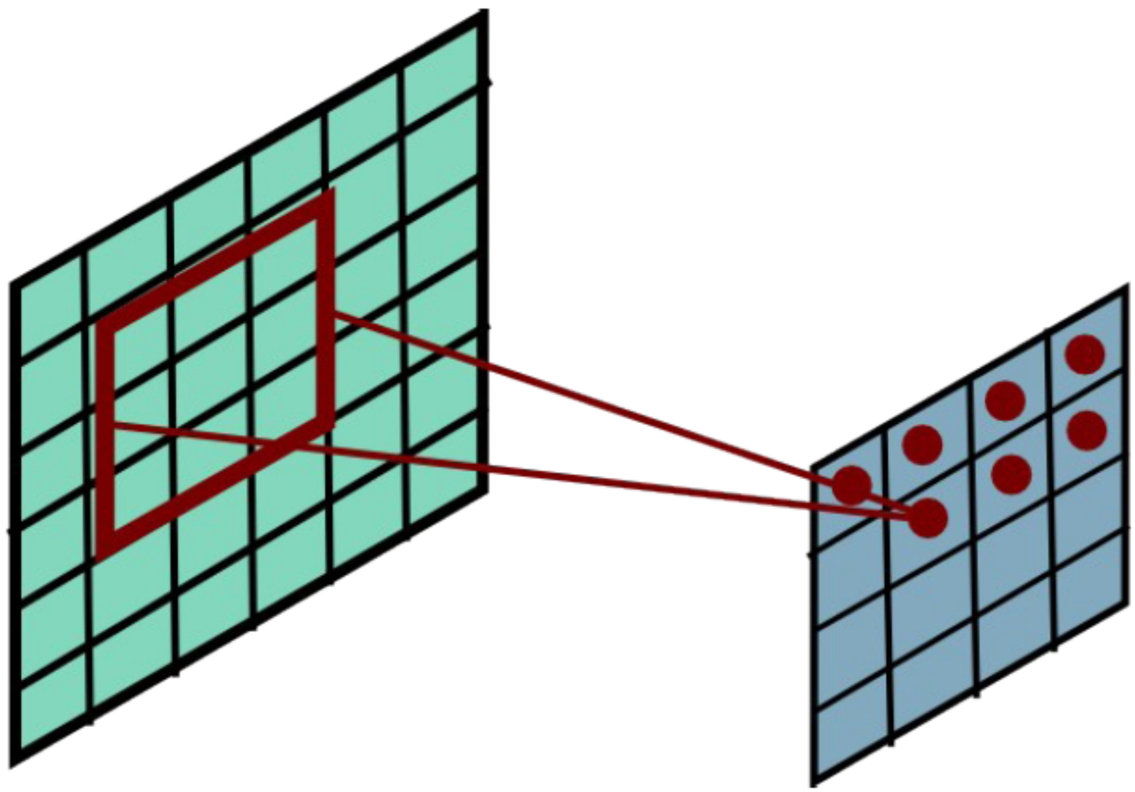
Convolutional Layer



Slide Credit: Marc'Aurelio Ranzato

Ranzato

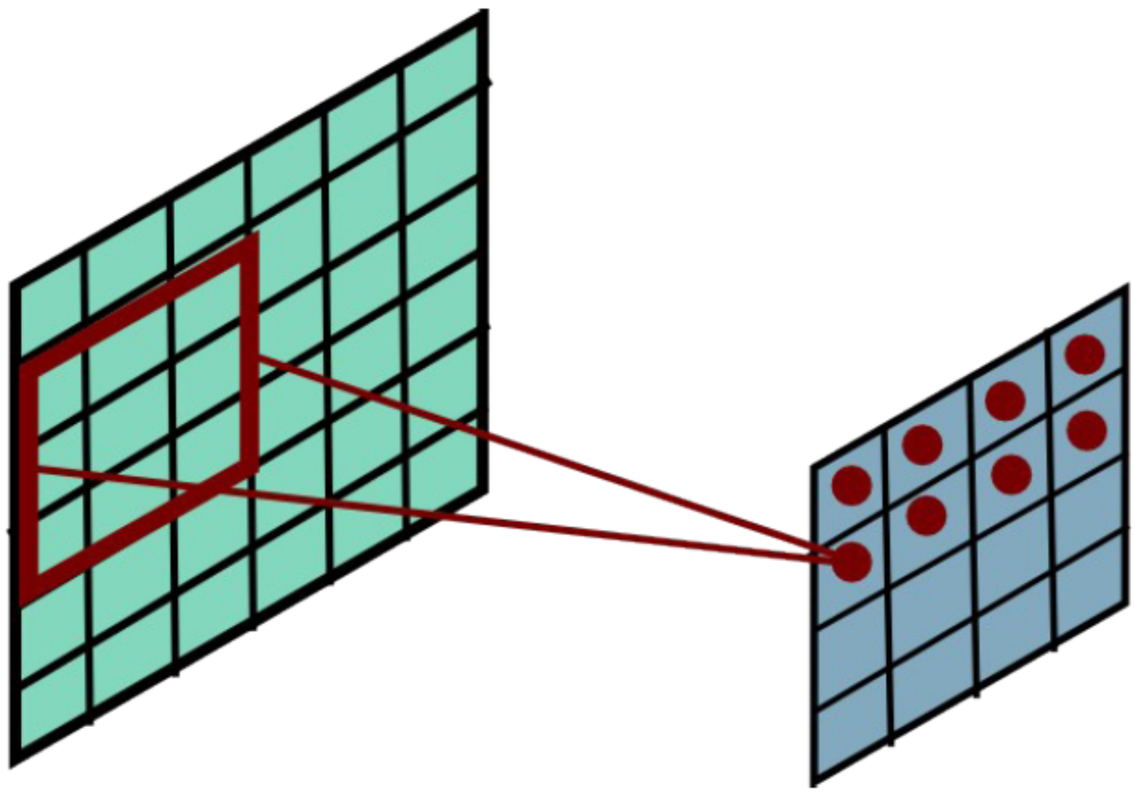
Convolutional Layer



Slide Credit: Marc'Aurelio Ranzato

Ranzato

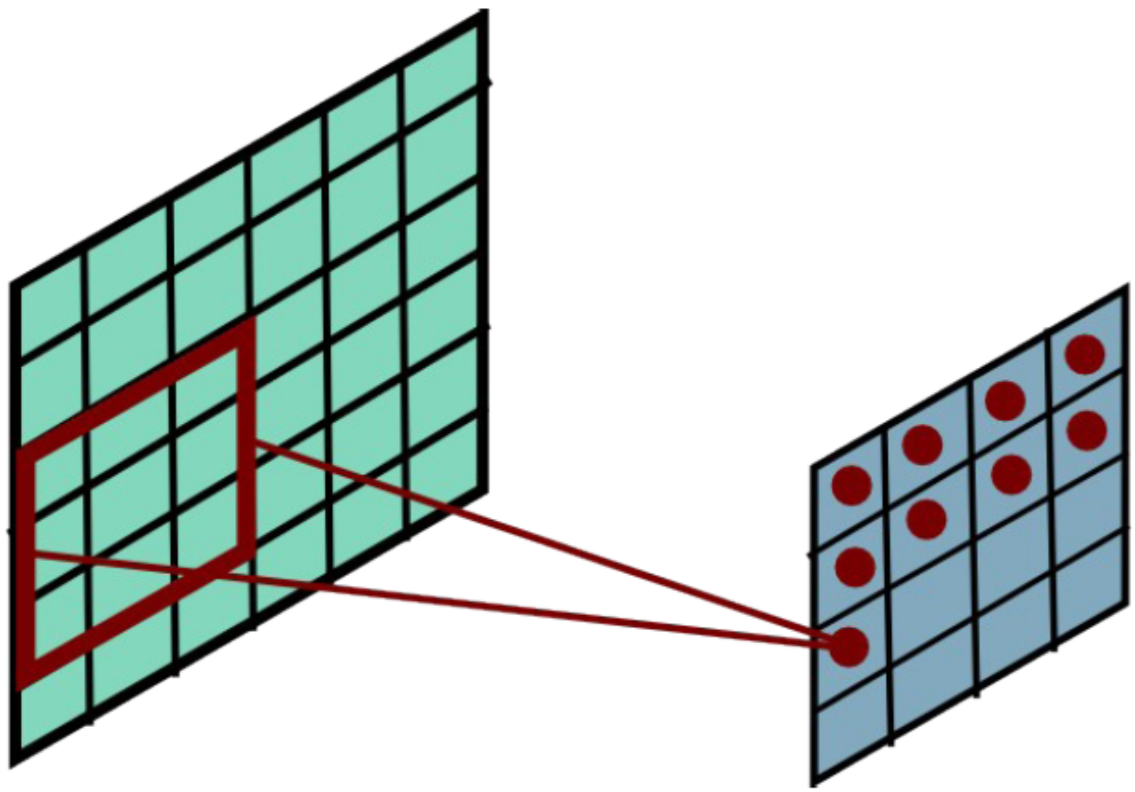
Convolutional Layer



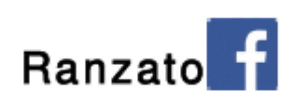
Slide Credit: Marc'Aurelio Ranzato

Ranzato

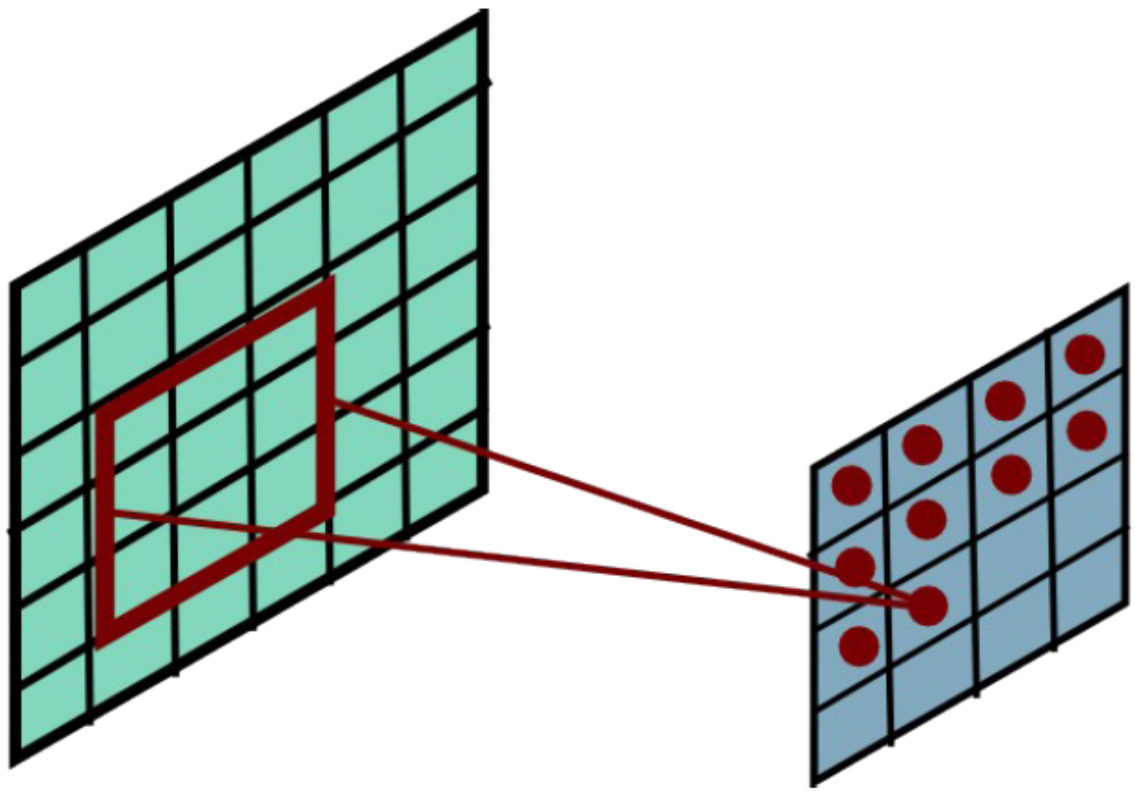
Convolutional Layer



Slide Credit: Marc'Aurelio Ranzato



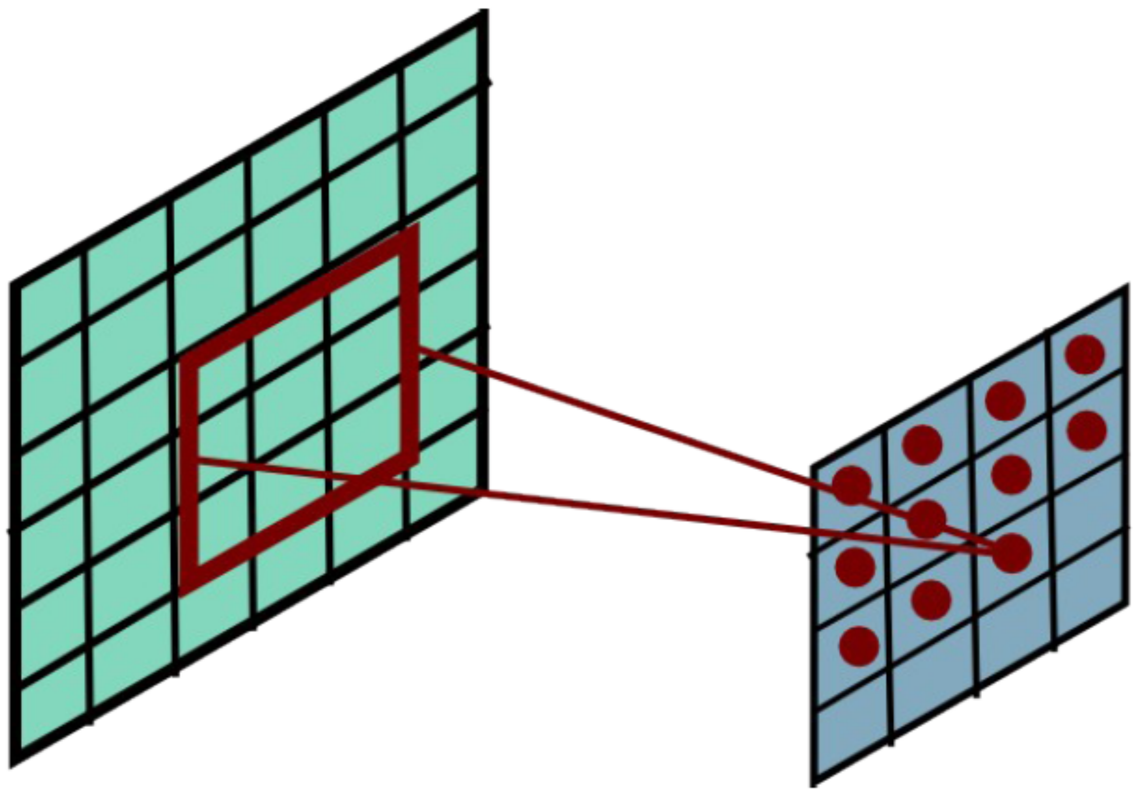
Convolutional Layer



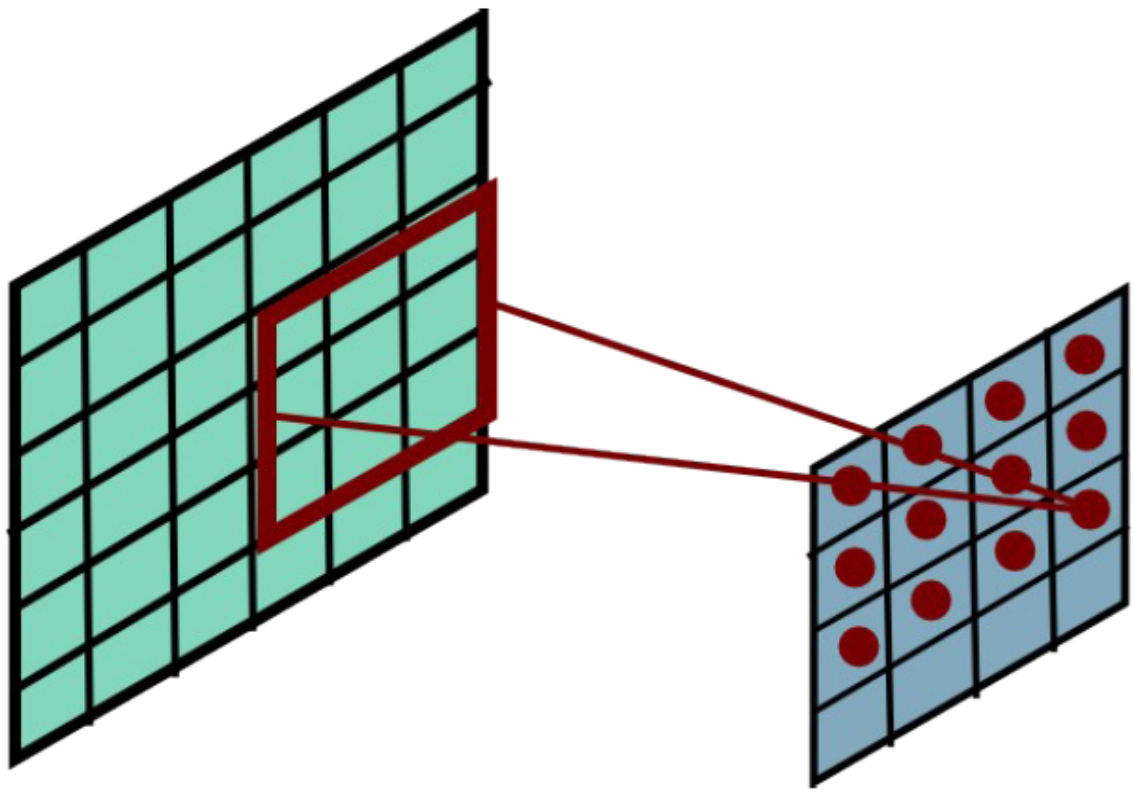
Slide Credit: Marc'Aurelio Ranzato

Ranzato

Convolutional Layer



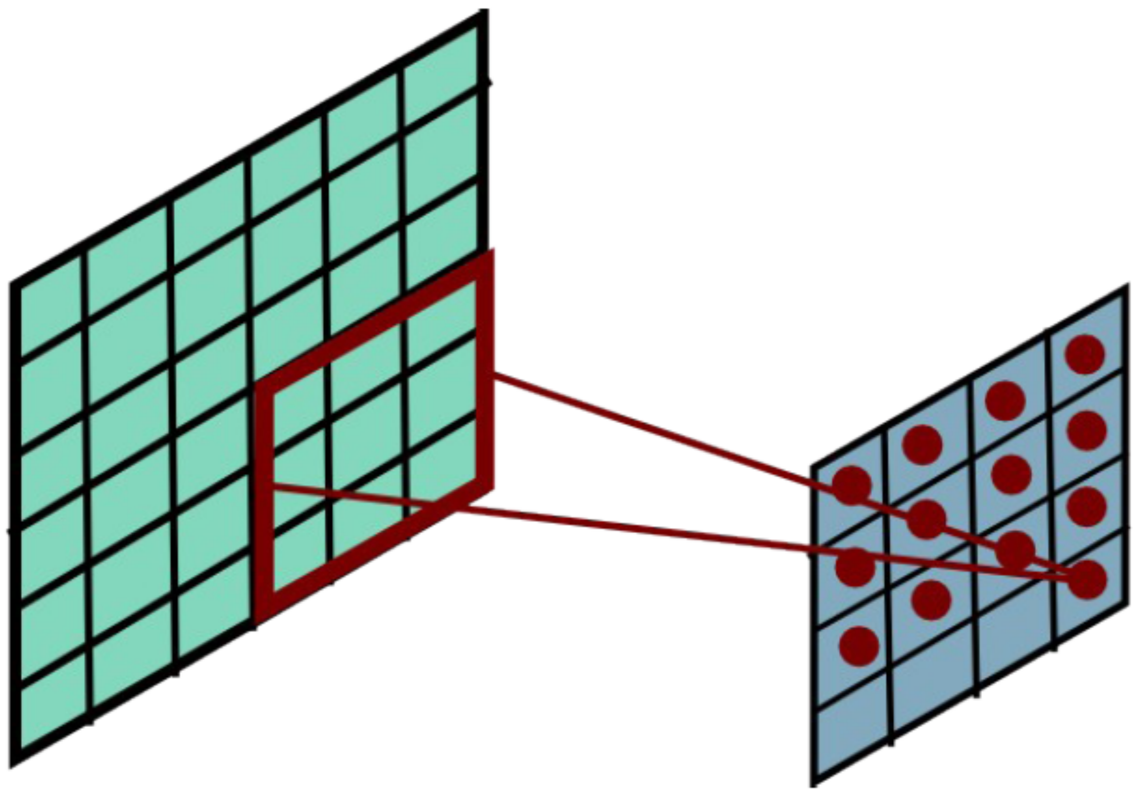
Convolutional Layer



Slide Credit: Marc'Aurelio Ranzato

Ranzato

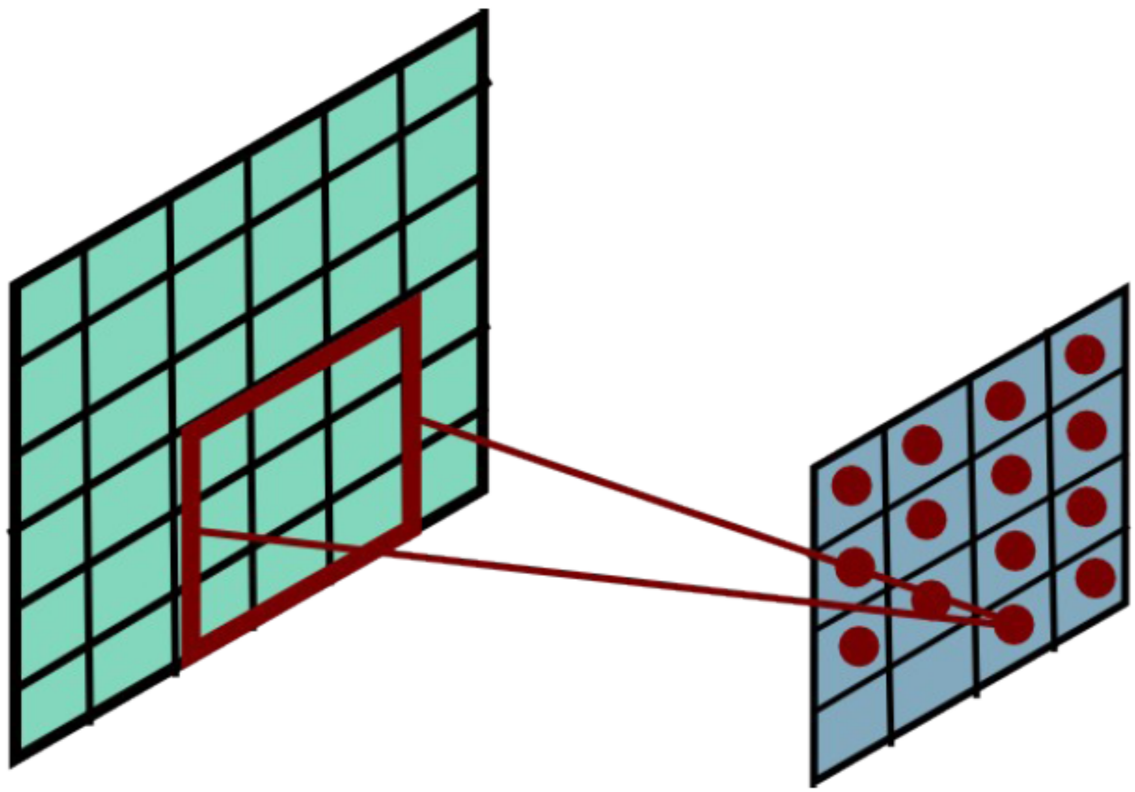
Convolutional Layer



Slide Credit: Marc'Aurelio Ranzato

Ranzato f

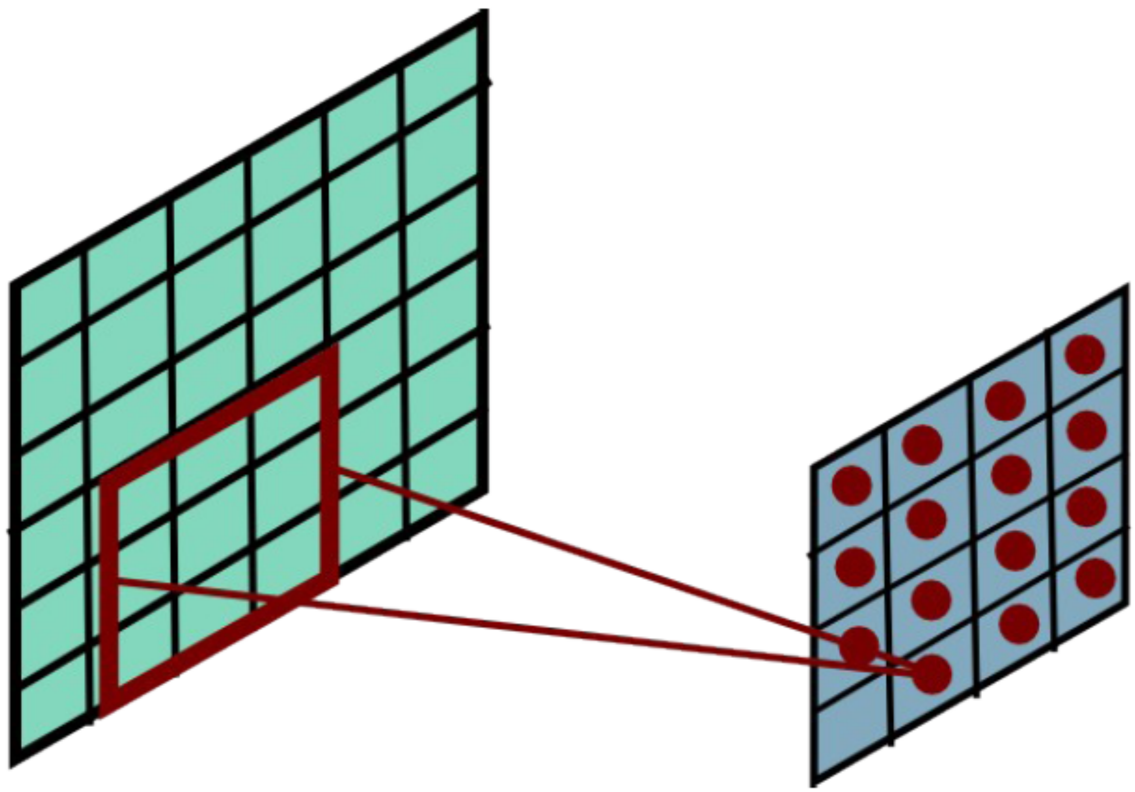
Convolutional Layer



Slide Credit: Marc'Aurelio Ranzato

Ranzato

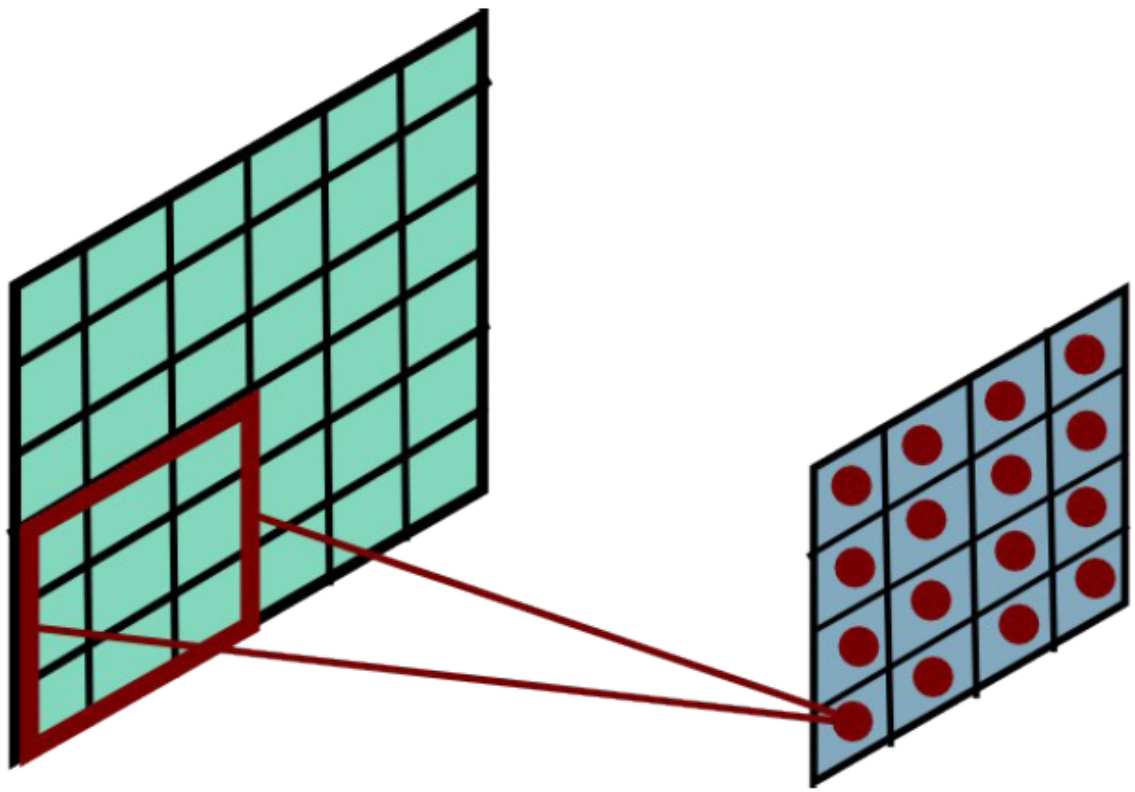
Convolutional Layer



Slide Credit: Marc'Aurelio Ranzato

Ranzato

Convolutional Layer



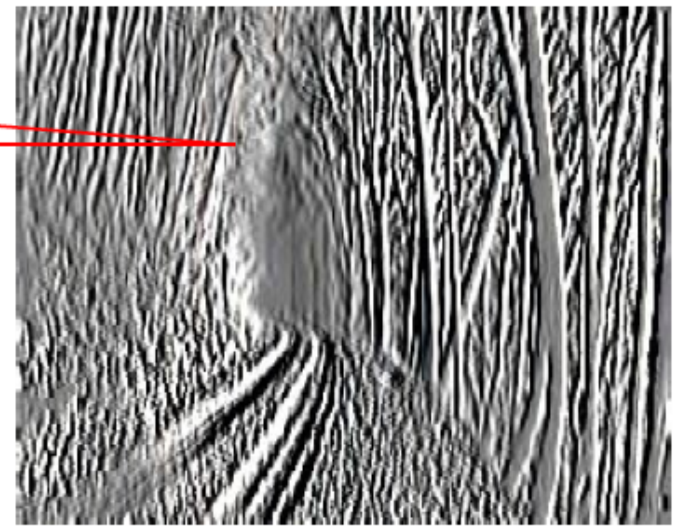
Mathieu et al. "Fast training of CNNs through FFTs" ICLR 2014

Slide Credit: Marc'Aurelio Ranzato

Ranzato

Convolutional Layer

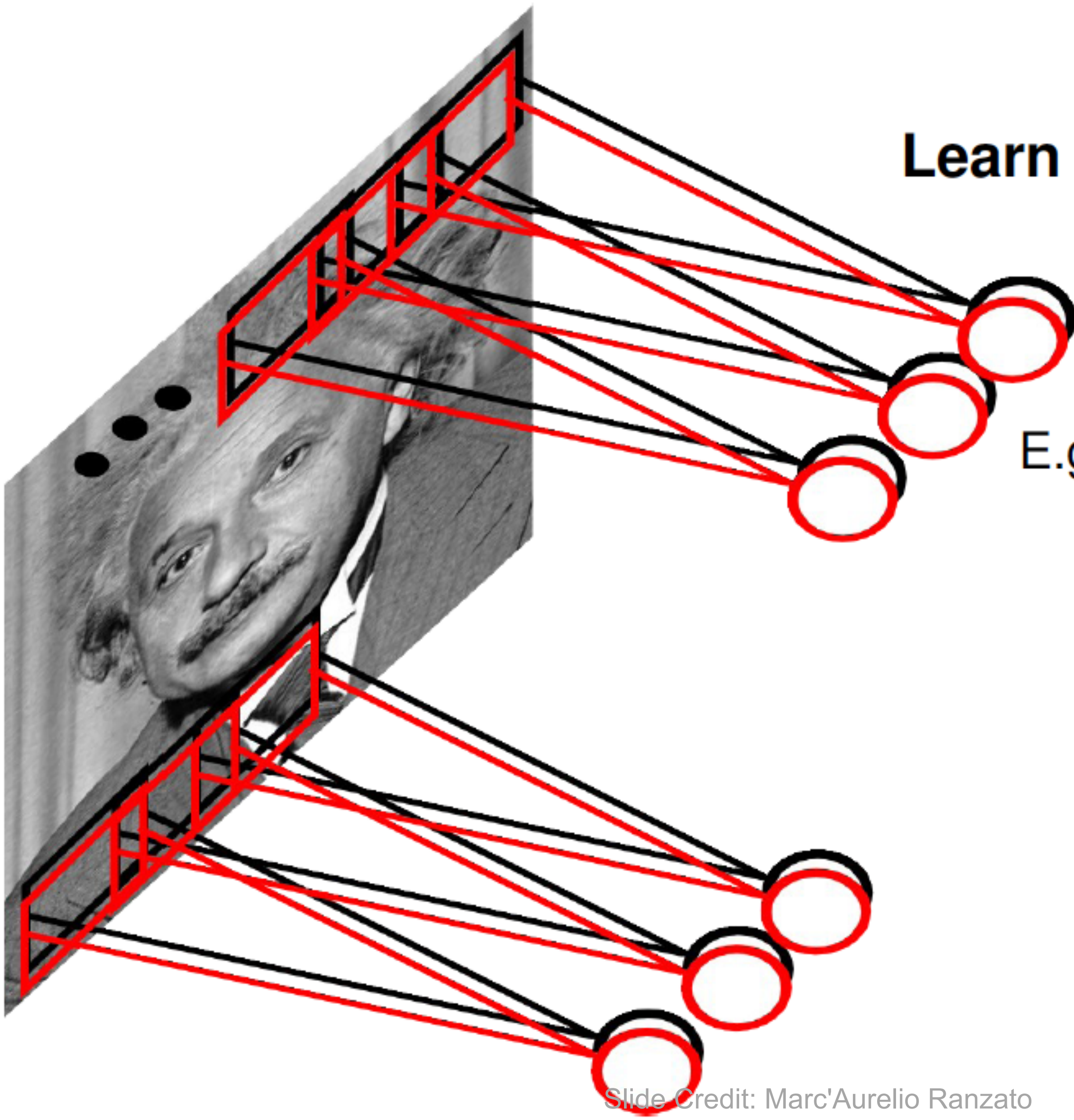
$$* \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} =$$



Slide Credit: Marc'Aurelio Ranzato

Ranzato

Convolutional Layer

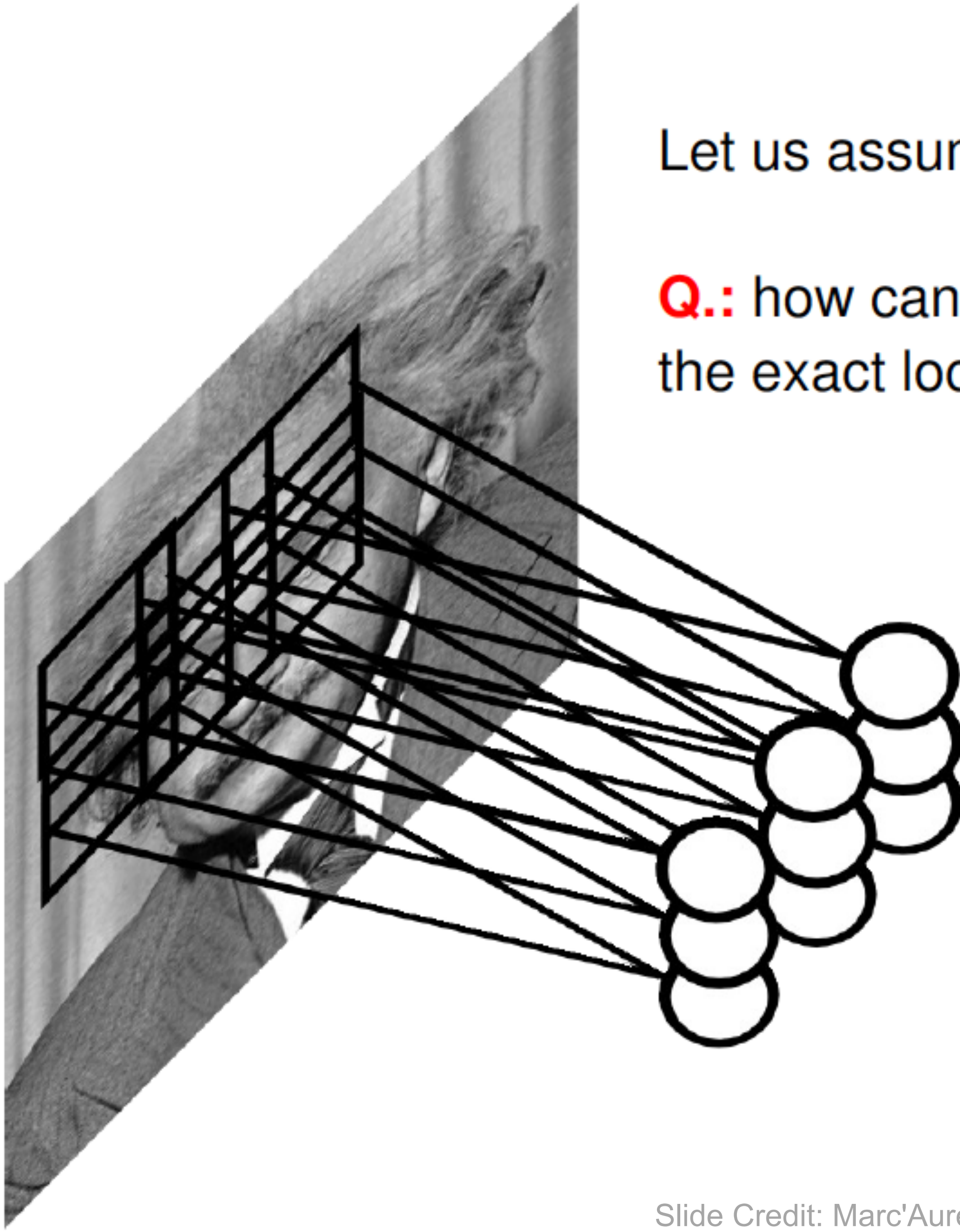


Learn multiple filters.

E.g.: 200x200 image
100 Filters
Filter size: 10x10
10K parameters

Ranzato

Pooling Layer



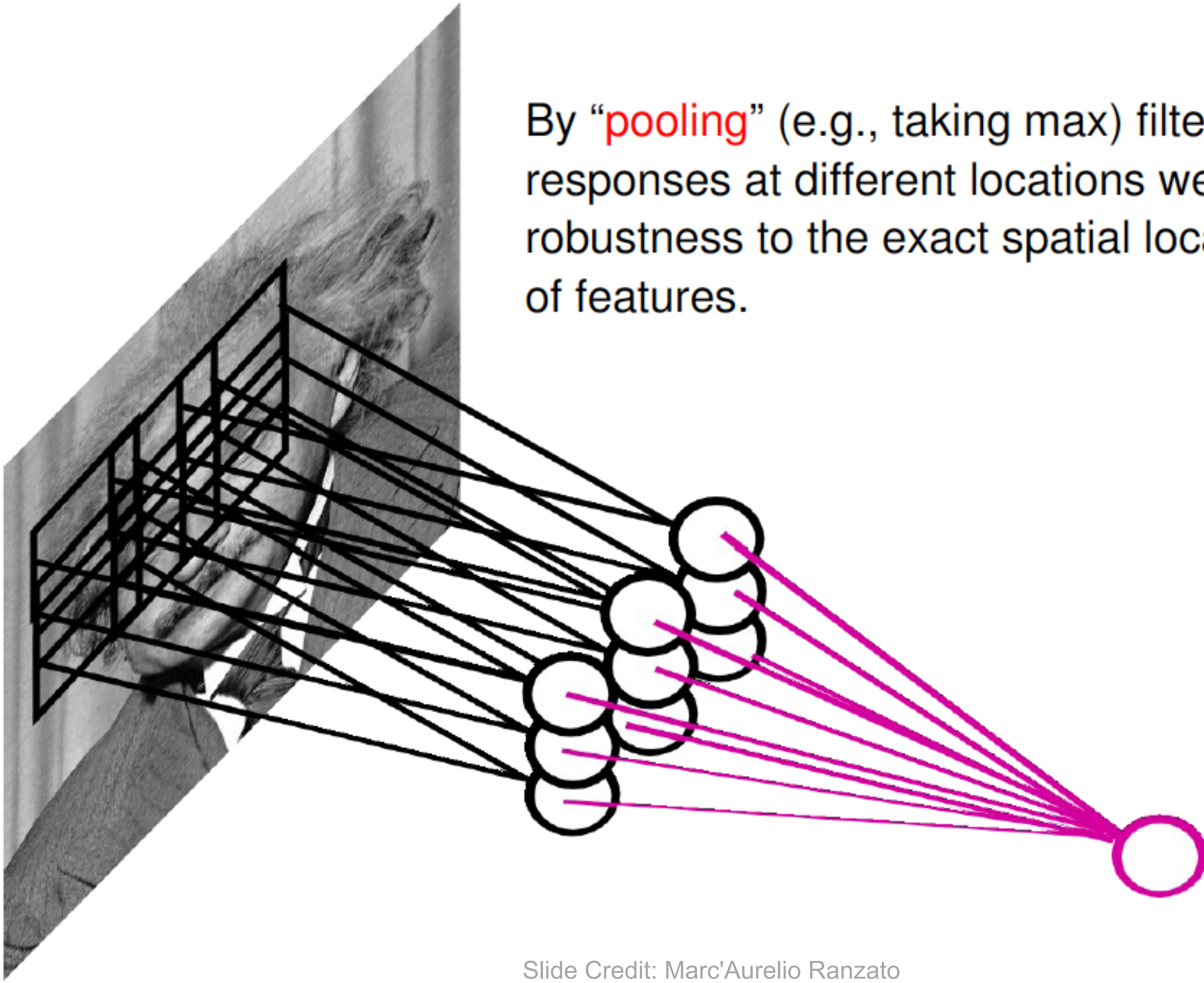
Let us assume filter is an “eye” detector.

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

Slide Credit: Marc'Aurelio Ranzato

Ranzato

Pooling Layer



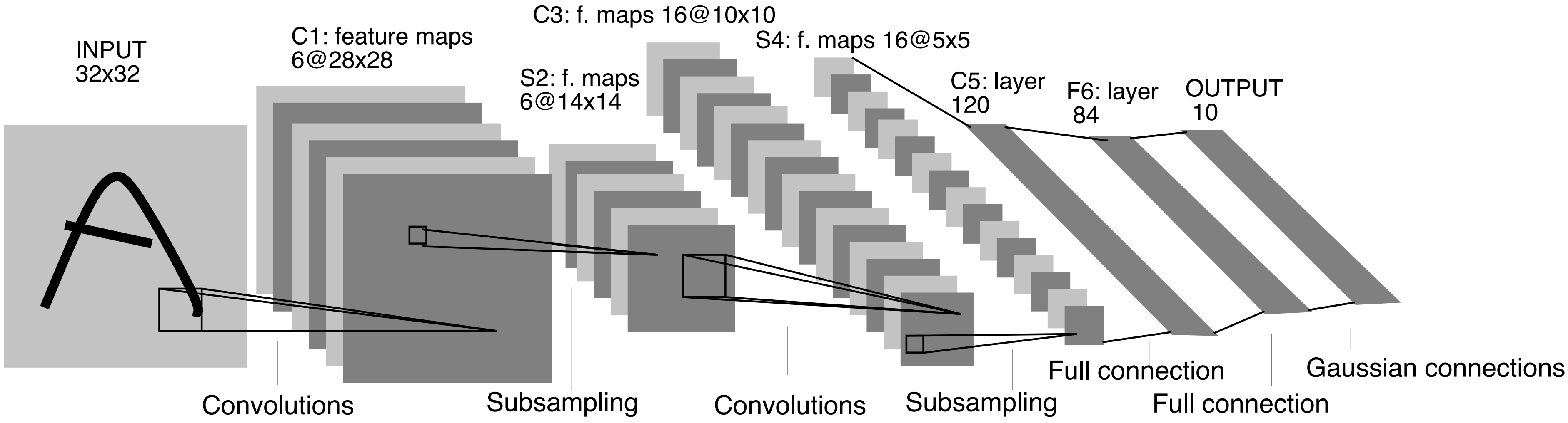
By "pooling" (e.g., taking max) filter responses at different locations we gain robustness to the exact spatial location of features.

Slide Credit: Marc'Aurelio Ranzato

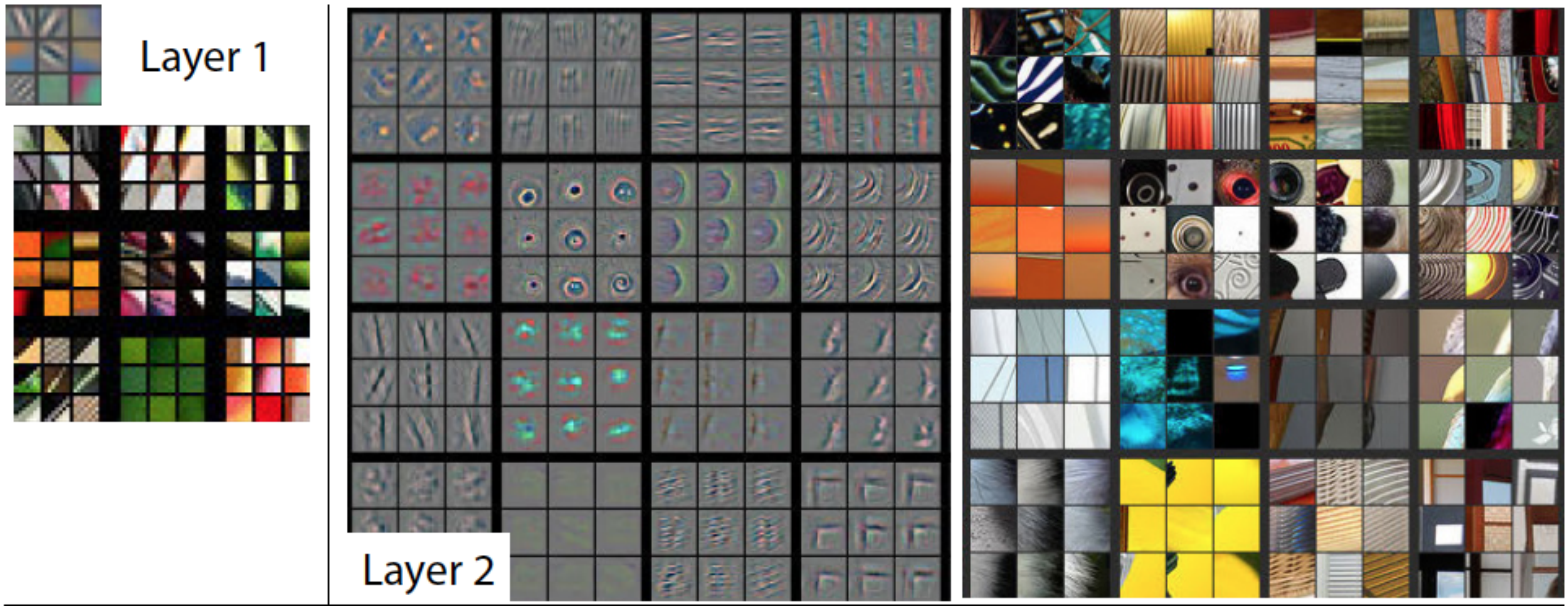
Ranzato

Convolutional Nets

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



Visualizing Learned Filters



Visualizing Learned Filters

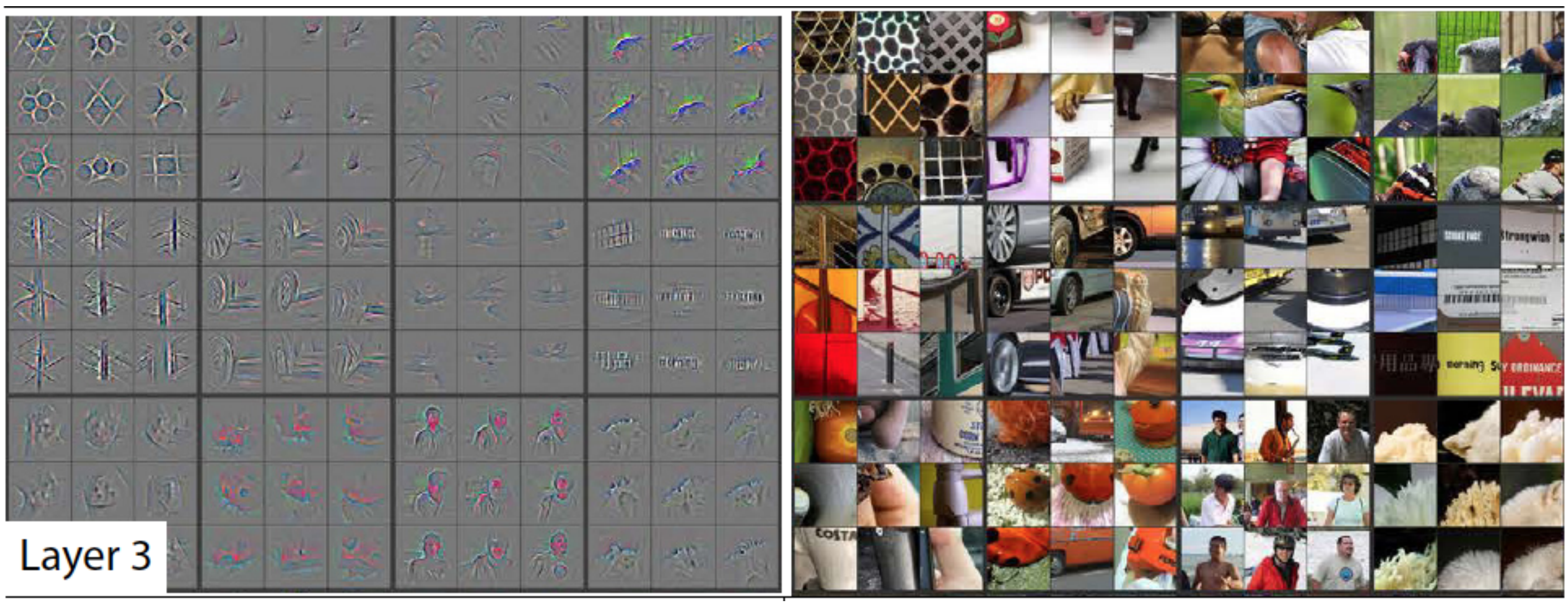


Figure Credit: [Zeiler & Fergus ECCV14]

Visualizing Learned Filters

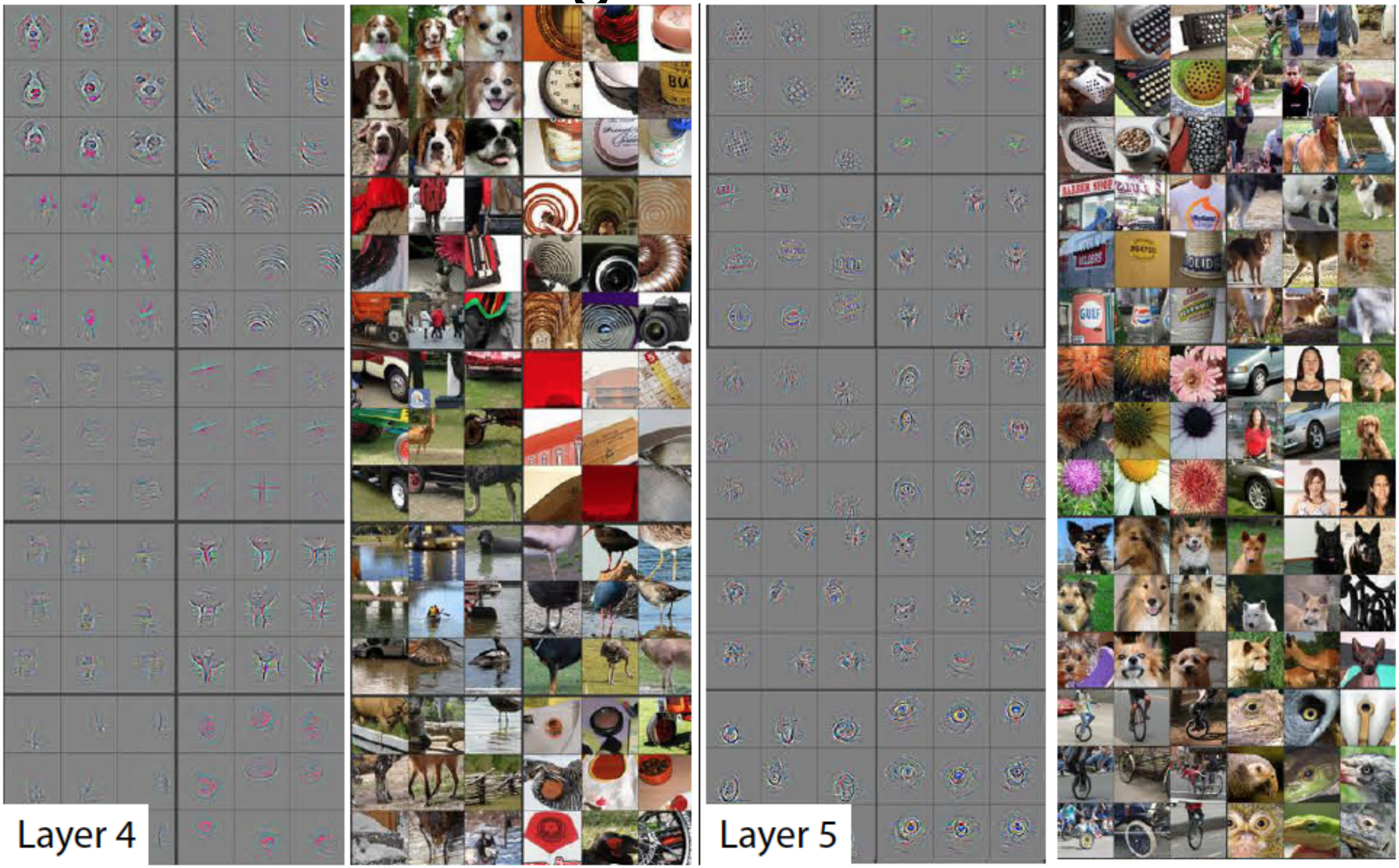


Figure Credit: [Zeiler & Fergus ECCV14]

Addressing non-linearly separable data – Option 1, non-linear features

- Choose non-linear features, e.g.,
  - Typical linear features: $w_0 + \sum_i w_i x_i$
  - Example of non-linear features:
    - Degree 2 polynomials, $w_0 + \sum_i w_i x_i + \sum_{ij} w_{ij} x_i x_j$

- Classifier $h_{\mathbf{w}}(\mathbf{x})$ still linear in parameters **w**
  - As easy to learn
  - Data is linearly separable in higher dimensional spaces
  - Express via kernels

Addressing non-linearly separable data – Option 2, non-linear classifier

- Choose a classifier $h_{\mathbf{w}}(\mathbf{x})$ that is non-linear in parameters **w**, e.g.,
  - Decision trees, neural networks, …

- More general than linear classifiers
- But, can often be harder to learn (non-convex/ concave optimization required)
- Often very useful (outperforms linear classifiers)
- In a way, both ideas are related

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

Learning Decision Trees

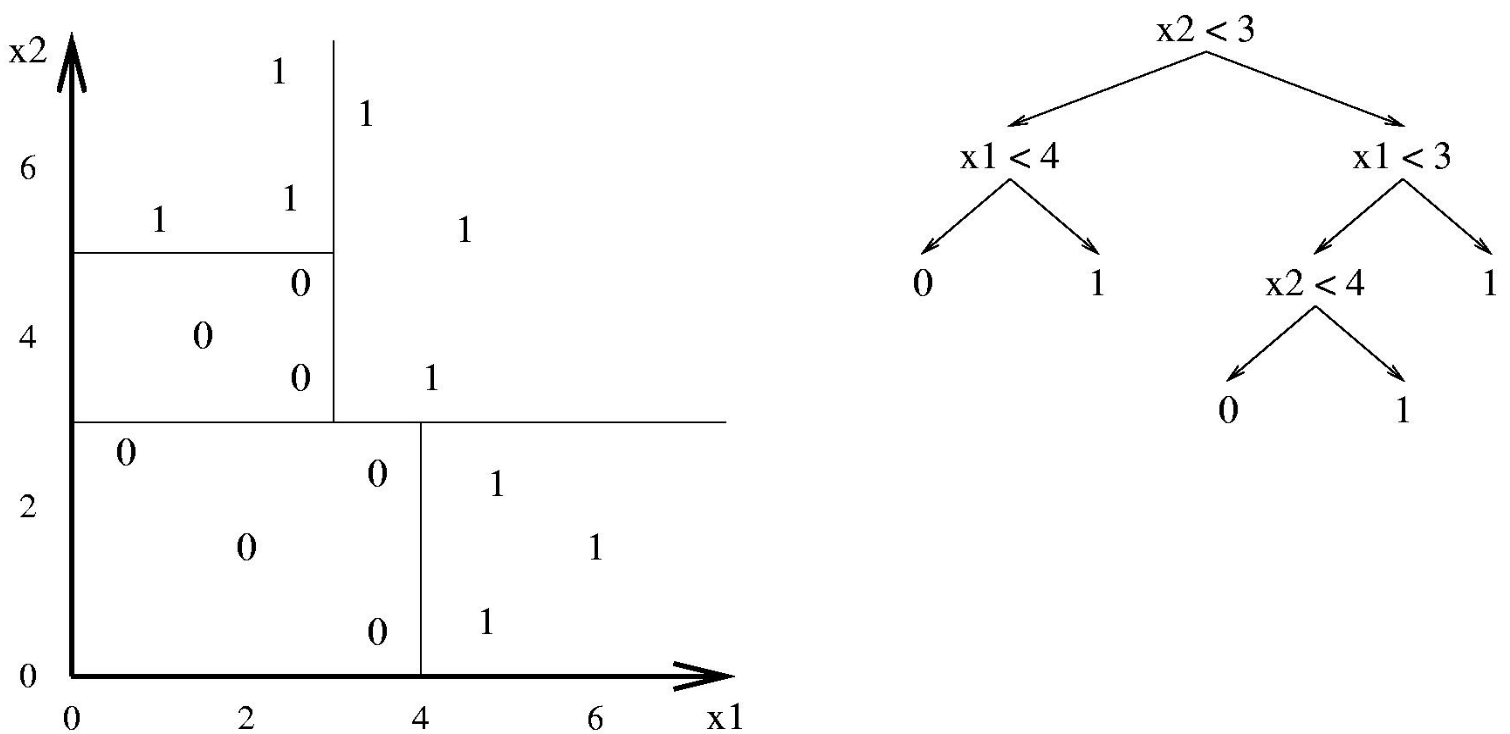
Decision trees provide a very popular and efficient hypothesis space.

- **Variable Size**. Any boolean function can be represented.
- **Deterministic**.
- **Discrete and Continuous Parameters**.

Slide Credit: Pedro Domingos, Tom Mitchel, Tom Dietterich

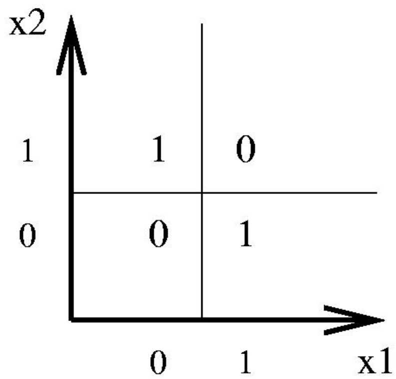
Decision Tree Decision Boundaries

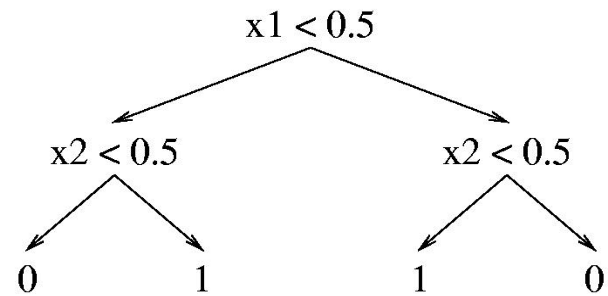
Decision trees divide the feature space into axis-parallel rectangles, and label each rectangle with one of the K classes.



Slide Credit: Pedro Domingos, Tom Mitchel, Tom Dietterich

Decision Trees Can Represent Any Boolean Function





The tree will in the worst case require exponentially many nodes, however.

Decision Trees Provide Variable-Size Hypothesis Space

As the number of nodes (or depth) of tree increases, the hypothesis space grows

- **depth 1** ("decision stump") can represent any boolean function of one feature.
- **depth 2** Any boolean function of two features; some boolean functions involving three features (e.g., $(x_1 \land x_2) \lor (\neg x_1 \land \neg x_3)$)
- **etc.**

A small dataset: Miles Per Gallon

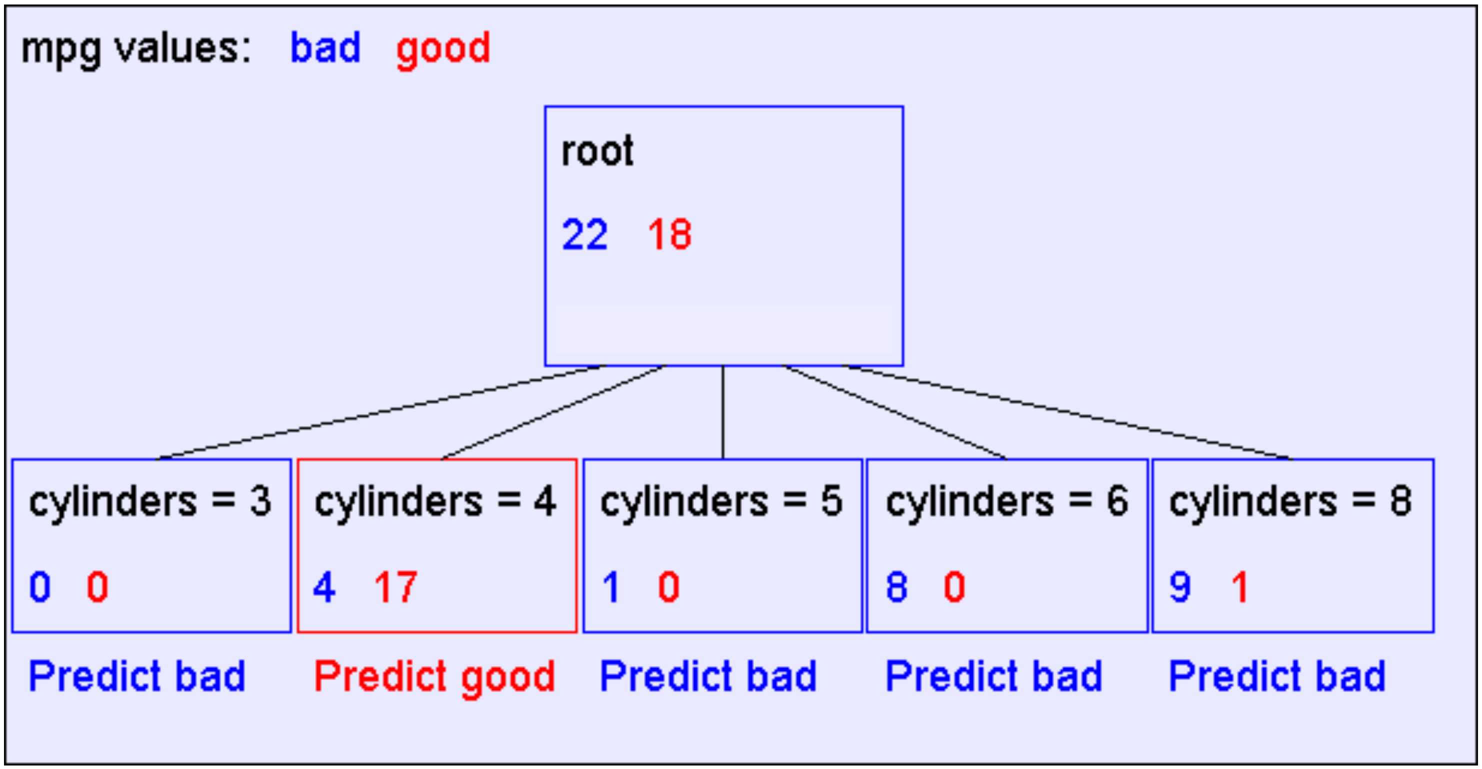
Suppose we want to predict MPG

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	europe
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	europe
bad	5	medium	medium	medium	medium	75to78	europe

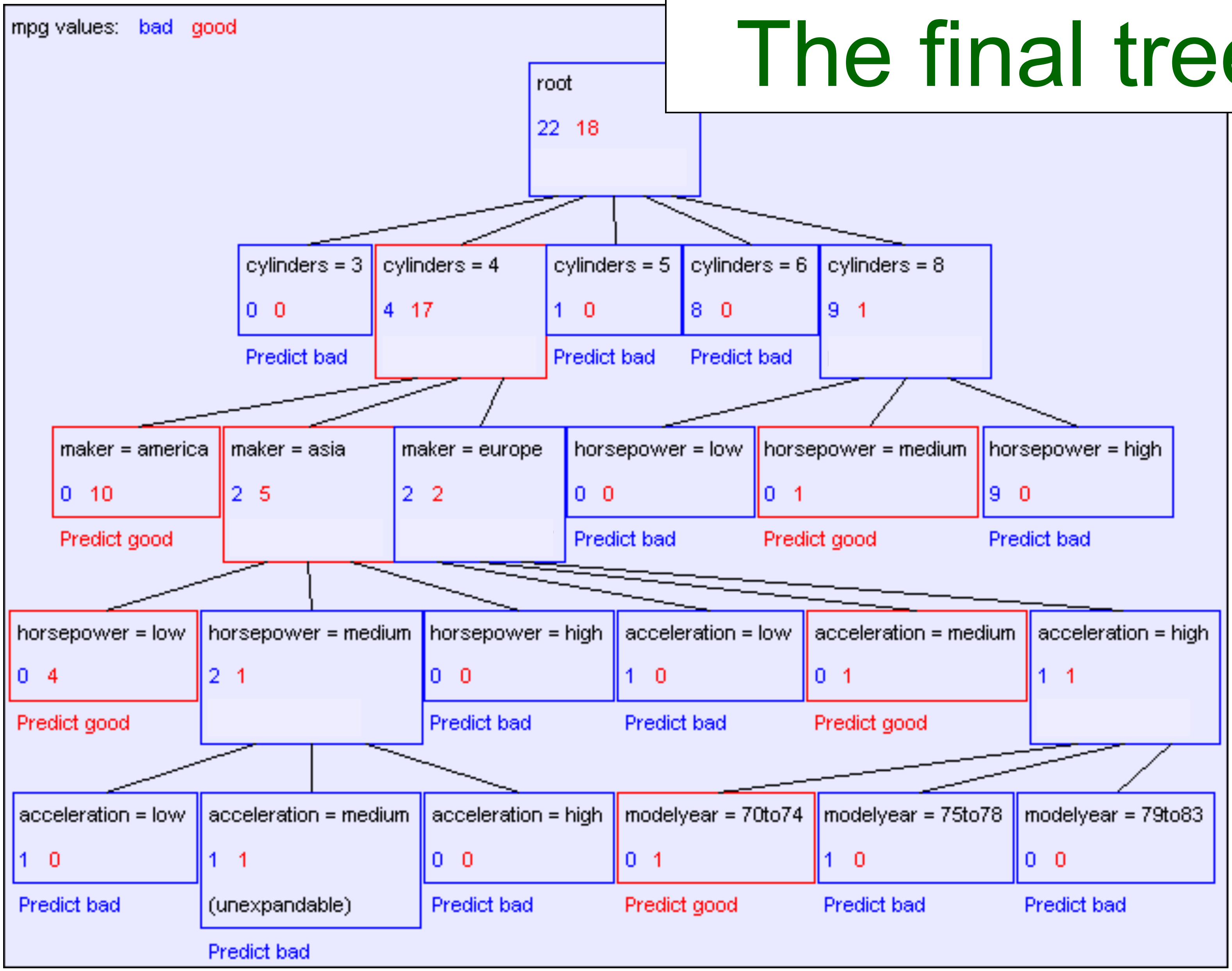
40 Records

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.

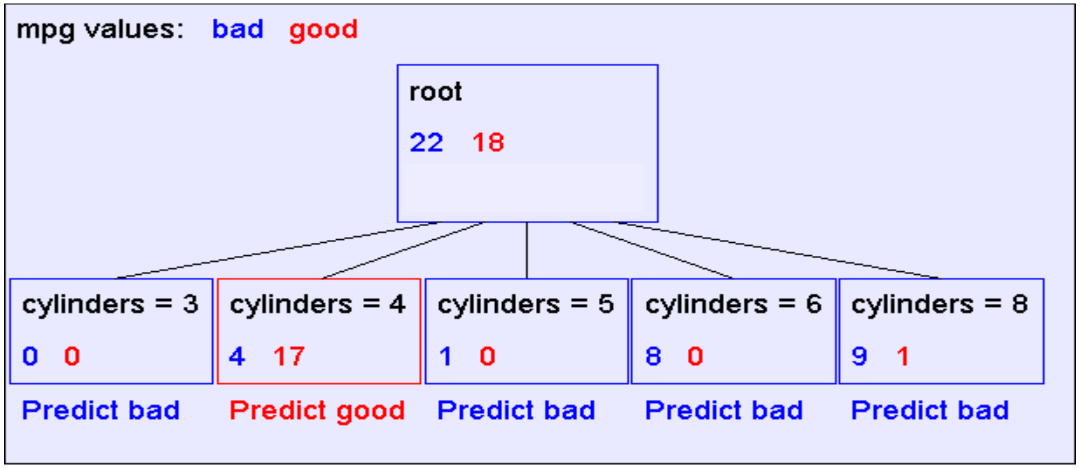
- Many trees can represent the same concept
- But, not all trees will have the same size!
  - e.g., $Y = (A \wedge B) \vee (\neg A \wedge C)$ (A and B) or (not A and C)

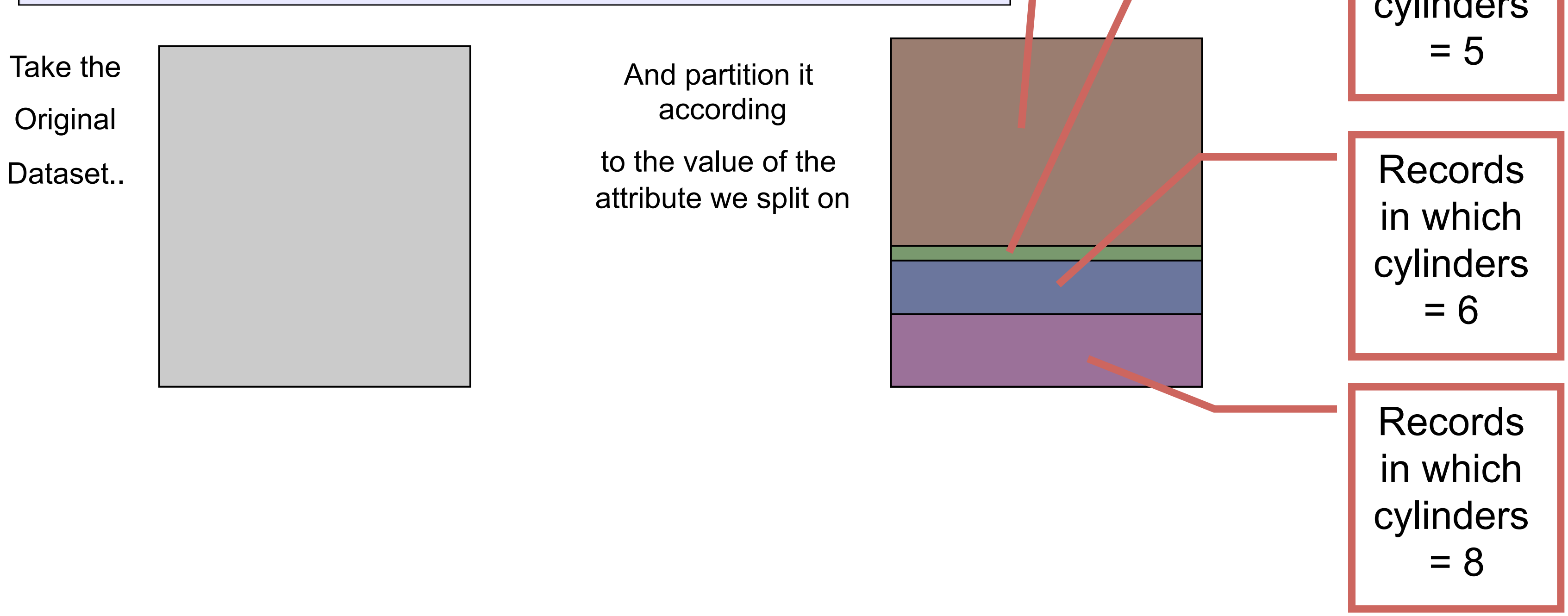
Learning decision trees is hard!!!

- Learning the simplest (smallest) decision tree is an NP-complete problem [Hyafil & Rivest ’76]

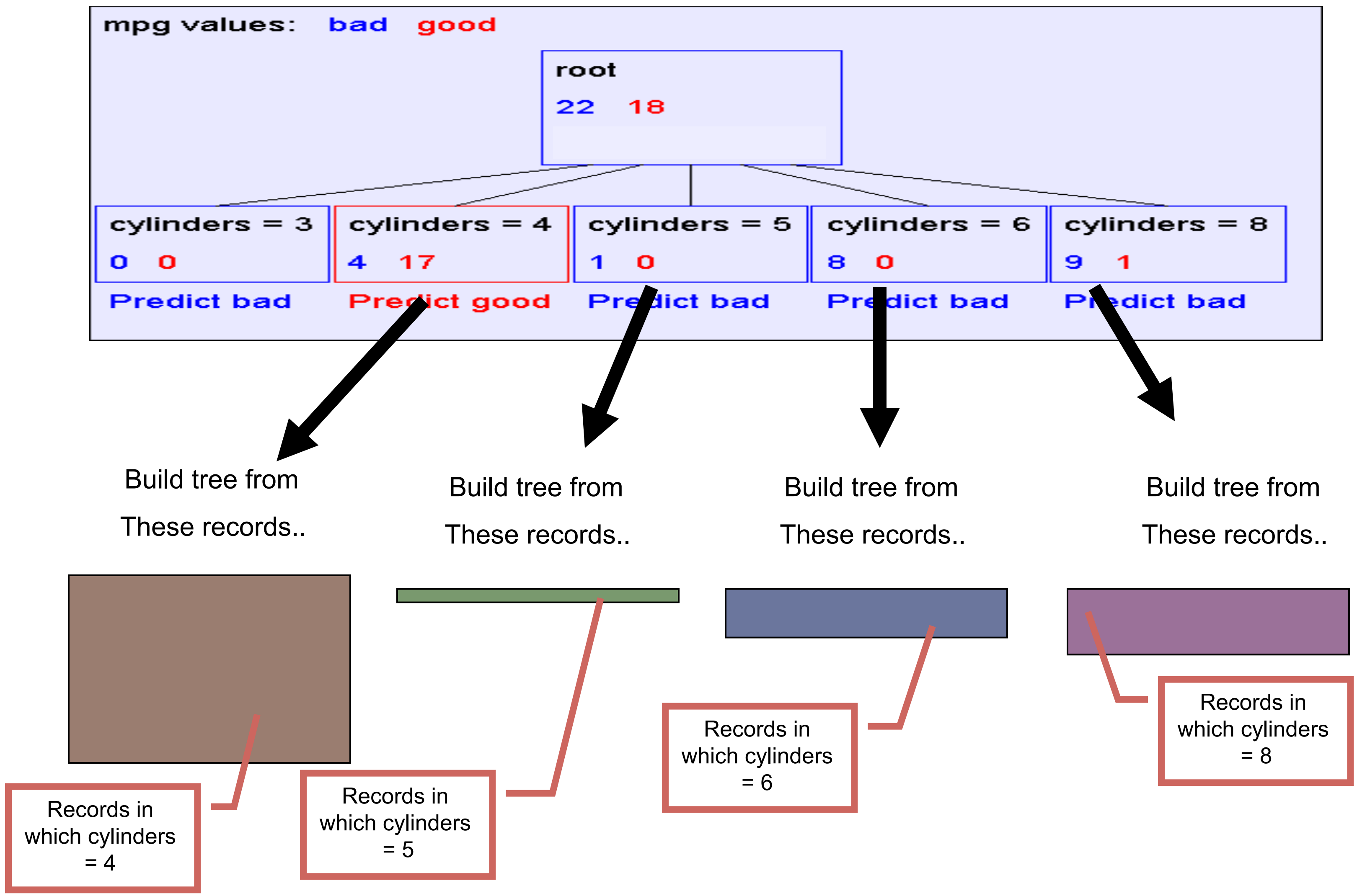
- Resort to a greedy heuristic:
  - Start from empty decision tree
  - Split on **next best attribute (feature)**
  - Recurse
    - “Iterative Dichotomizer” (ID3)
    - C4.5 (ID3+improvements)

Recursion Step

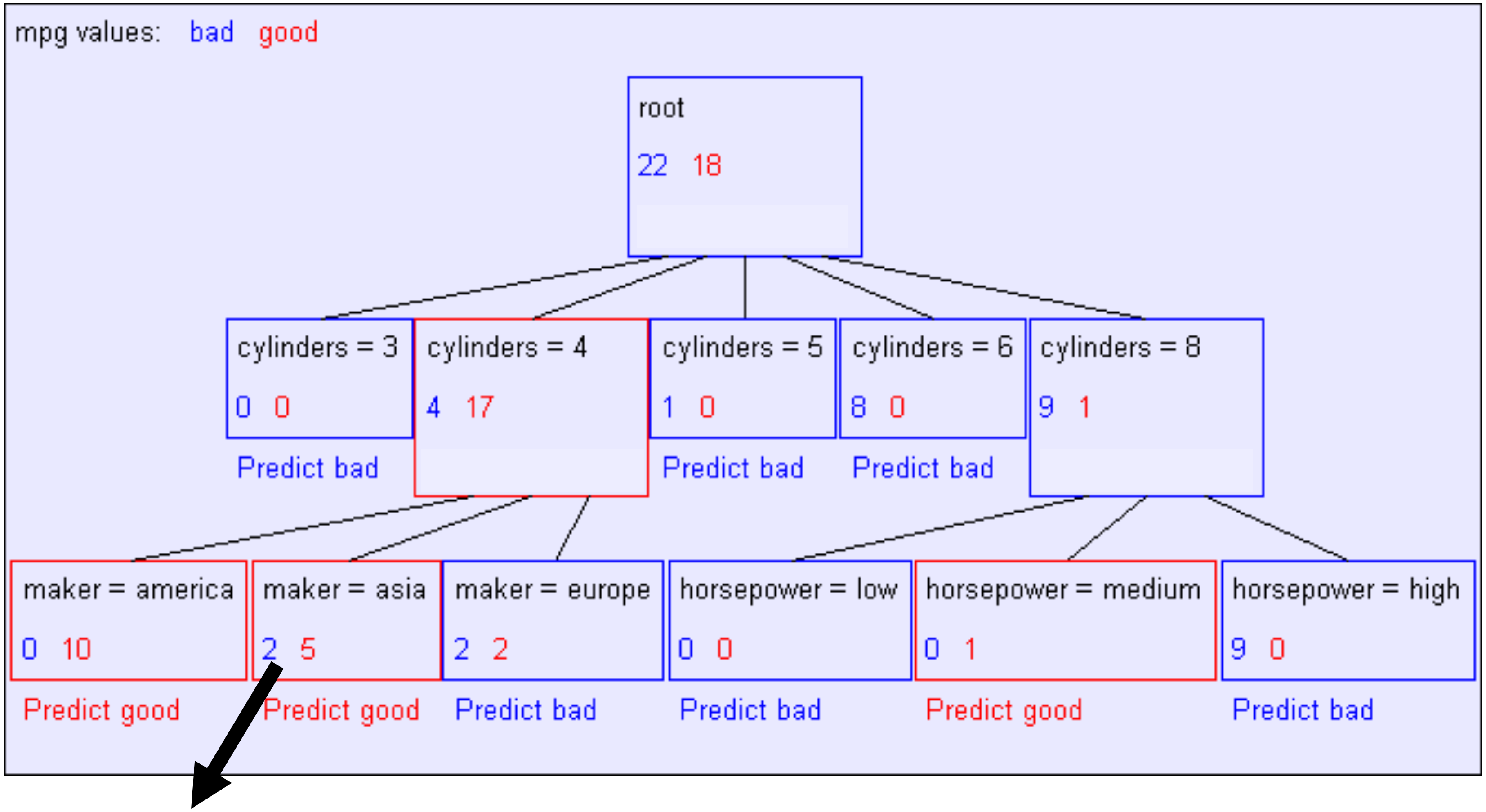




Recursion Step



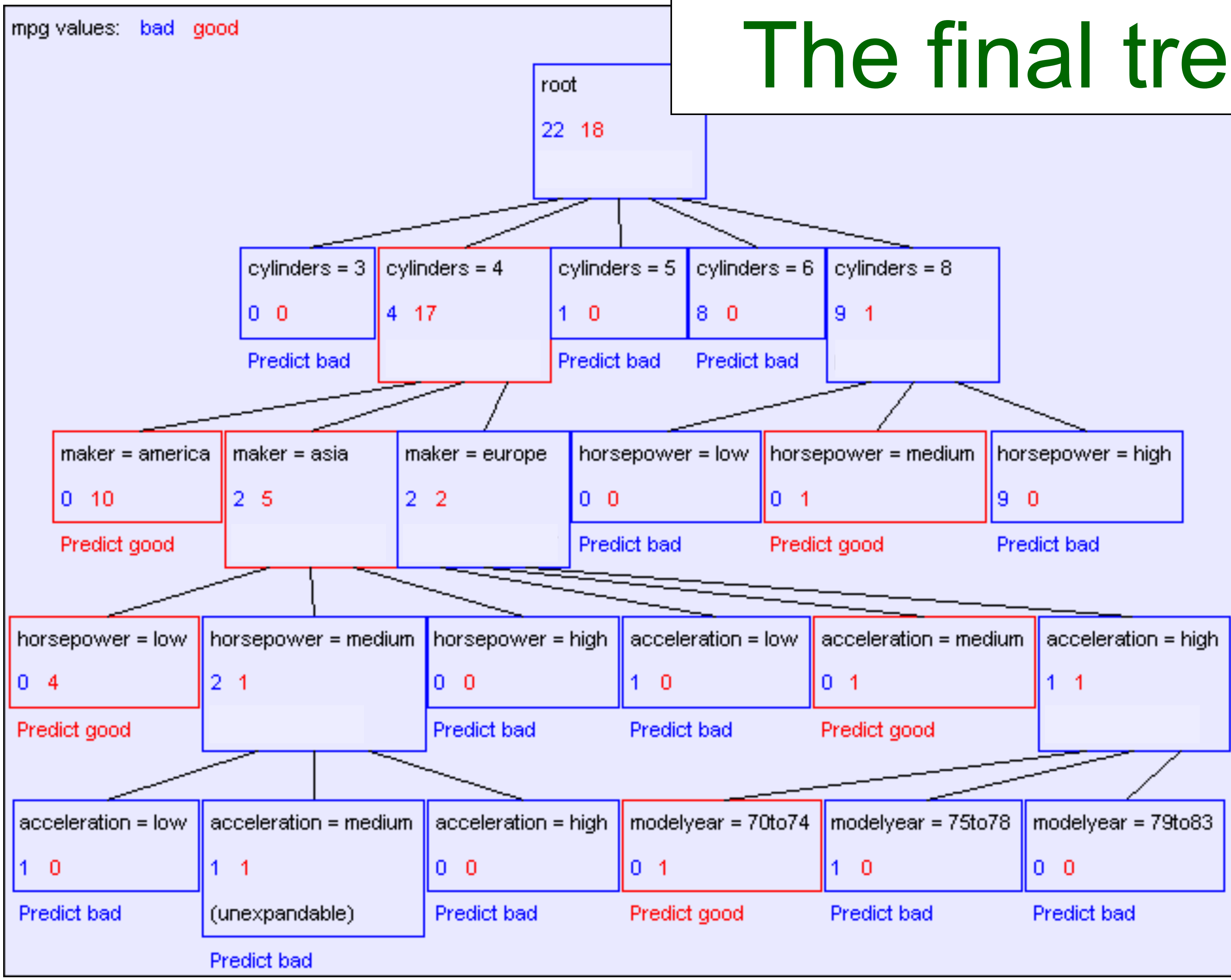
Second level of tree



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



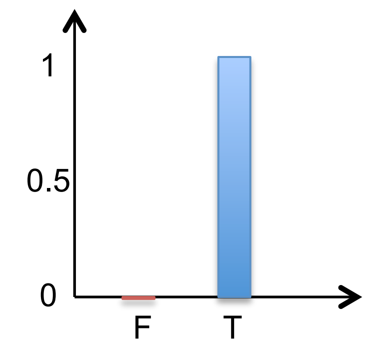
Choosing a good attribute

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

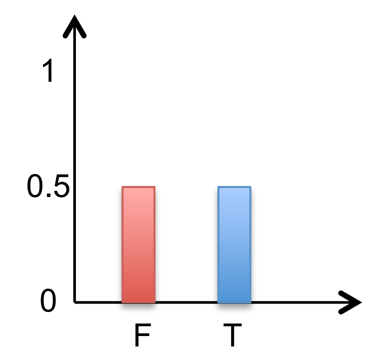
Measuring uncertainty

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

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



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



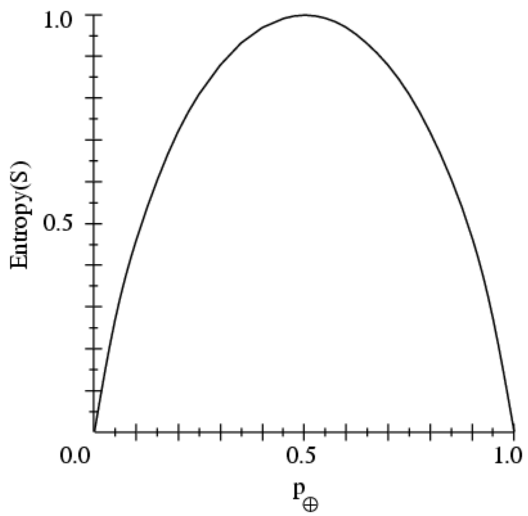
Entropy

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

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

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

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

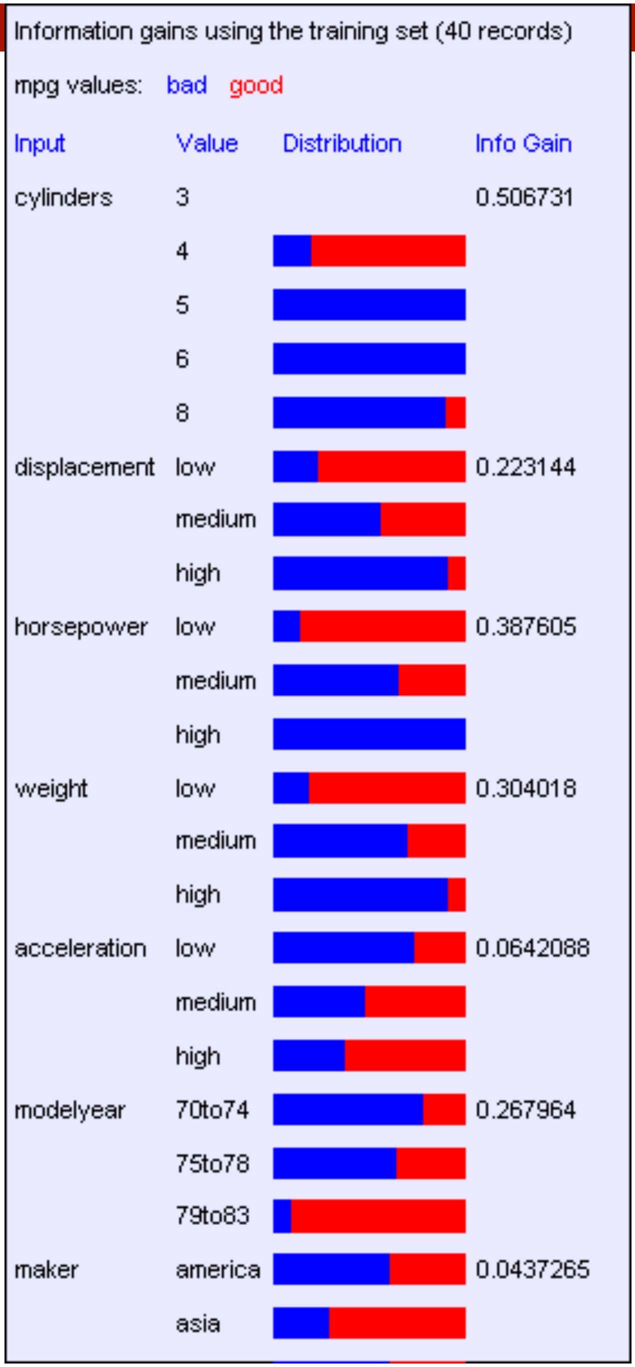
- Information gain is difference $IG(X) = H(Y) - H(Y \mid X)$
  - (Technically it's mutual information; but in this context also referred to as information gain)

Learning decision trees

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

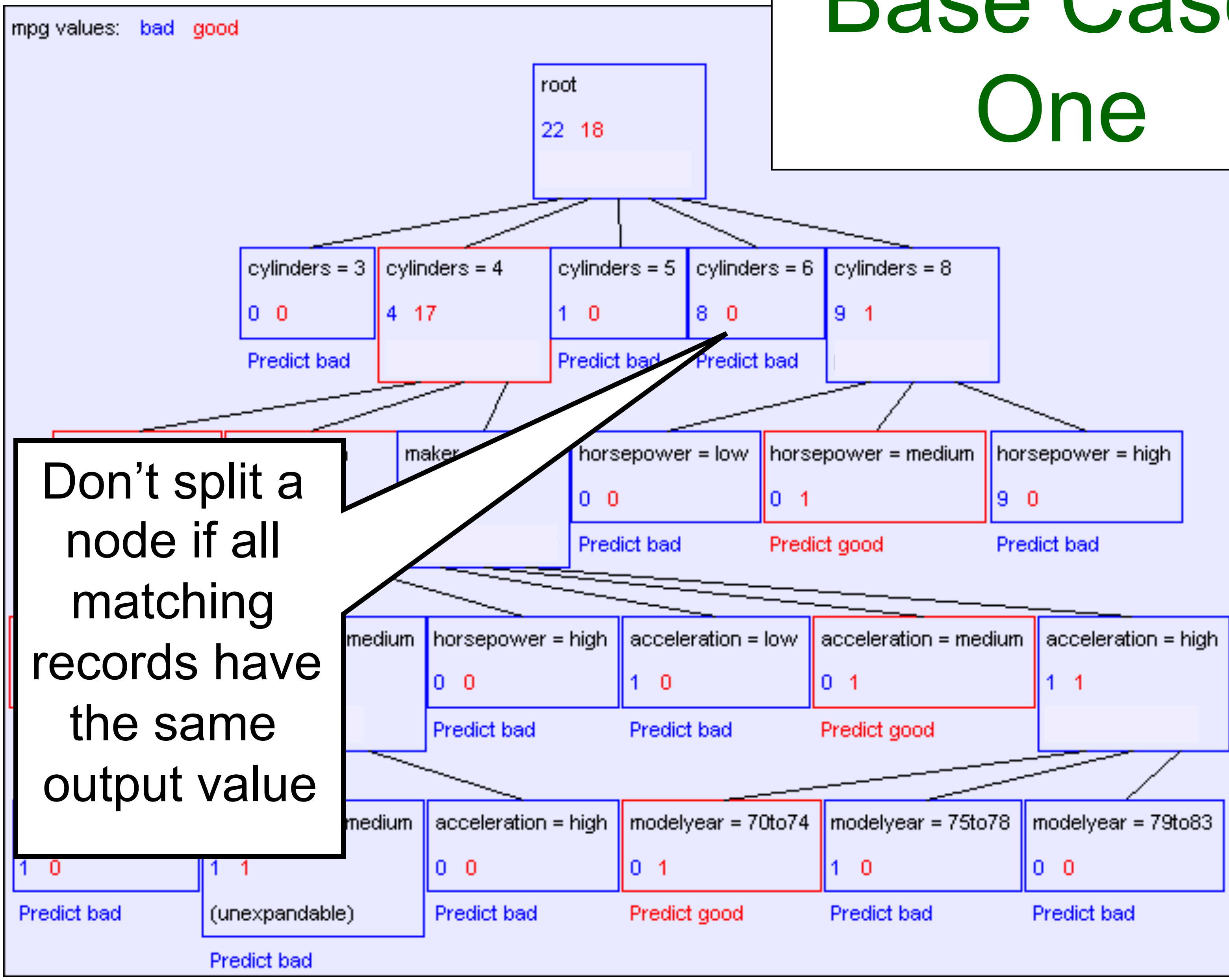
Suppose we want to predict MPG

Look at all the information gains…



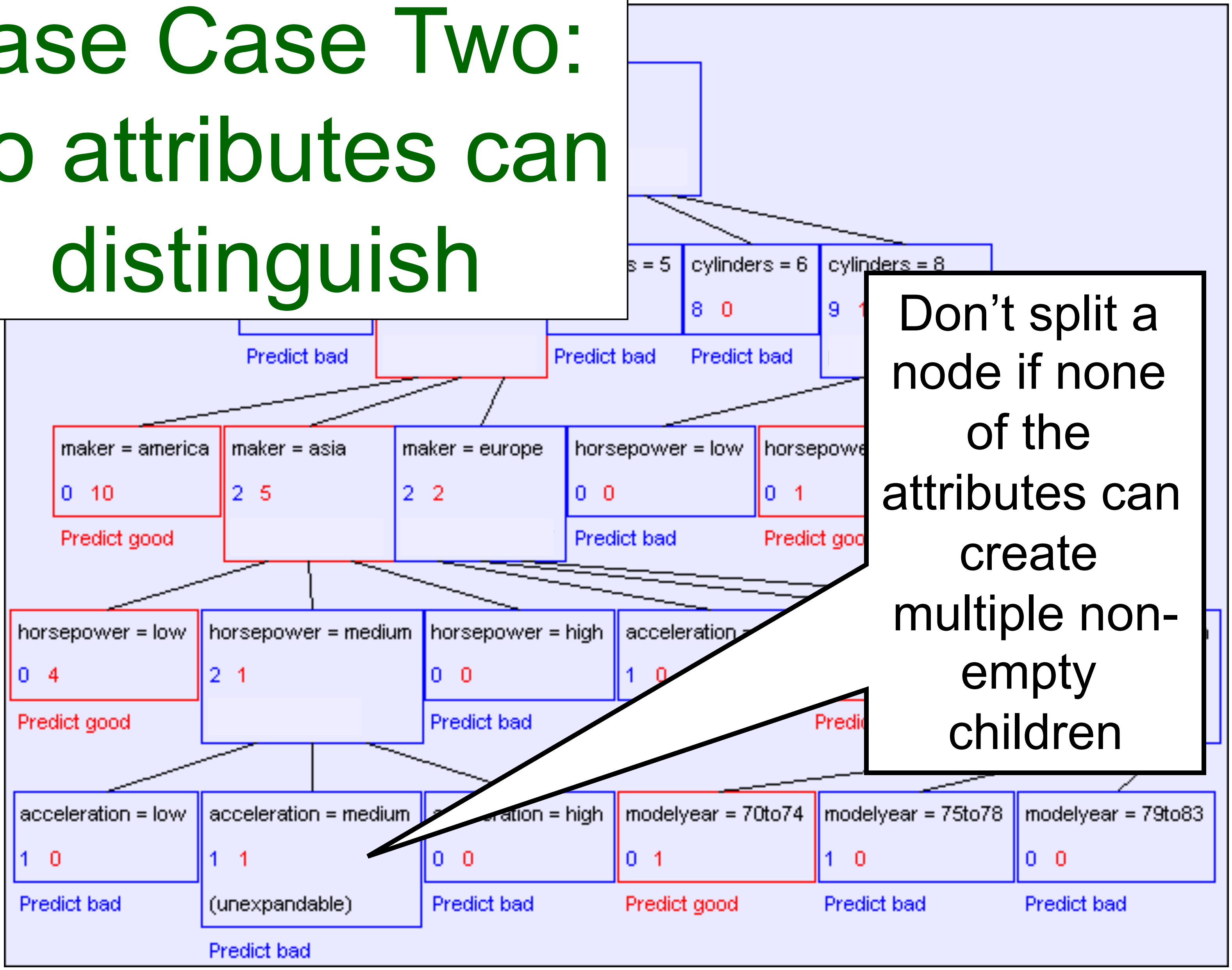
When do we stop?

Base Case One



Don't split a node if all matching records have the same output value

Base Case Two: No attributes can distinguish



Don’t split a node if none of the attributes can create multiple non-empty children

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

Proposed Base Case 3:

If all attributes have zero information gain then don't recurse

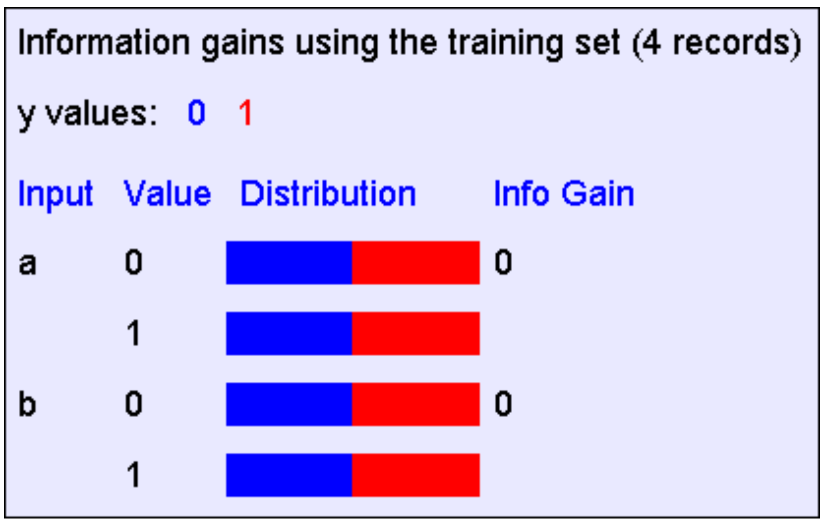
• *Is this a good idea?*

The problem with Base Case 3

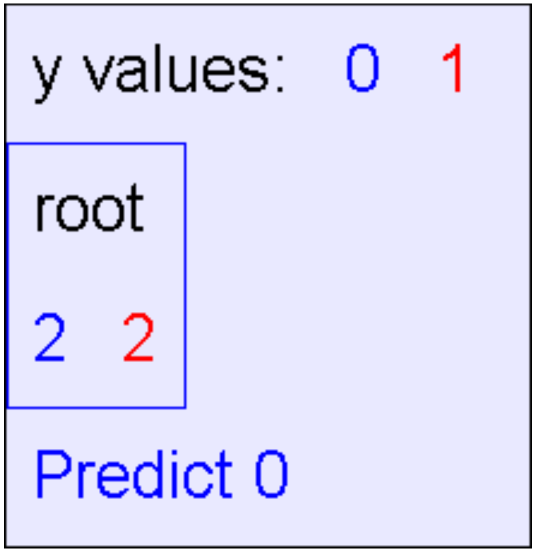
a	b	y
0	0	0
0	1	1
1	0	1
1	1	0

y = a XOR b

The information gains:



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 XOR b

The resulting decision tree: