Convolutional Neural Networks

CMSC 733 Fall 2015

Angjoo Kanazawa

Overview

Goal: Understand what Convolutional Neural Networks (ConvNets) are & intuition behind it.

1. Brief Motivation for Deep Learning
2. What are ConvNets?
3. ConvNets for Object Detection

First of all what is Deep Learning?

- Composition of non-linear transformation of the data.
- Goal: **Learn useful representations, aka features, directly from data.**

- Many varieties, can be unsupervised or supervised.
- Today is about ConvNets, which is a **supervised** deep learning method.

Recap: Supervised Learning

$\left\{(\boldsymbol{x}^i, y^i), i=1 \ldots P\right\}$ training dataset

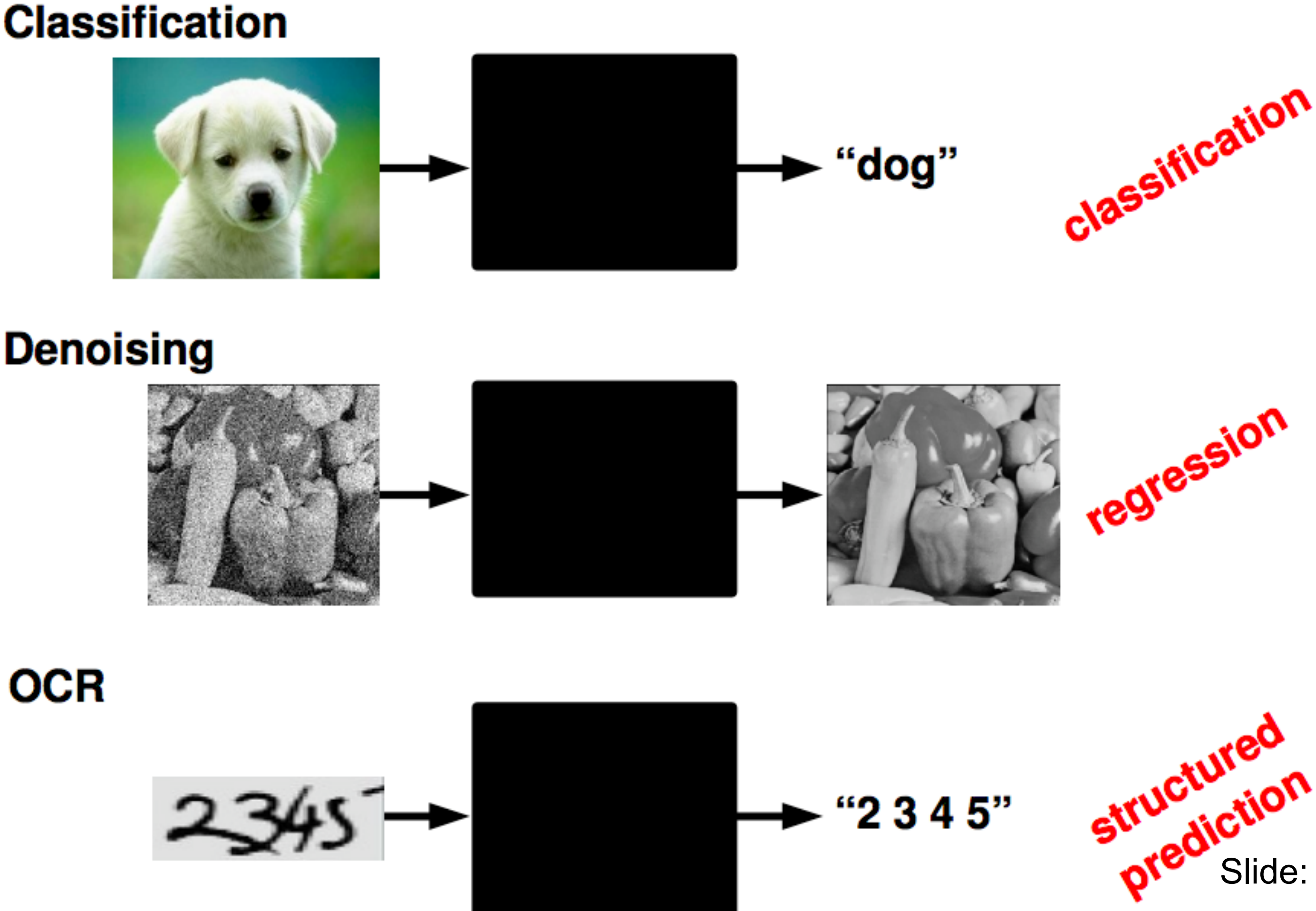
$\boldsymbol{x}^i$ i-th input training example

y^i i-th target label

P number of training examples

Slide: M. Ranzato

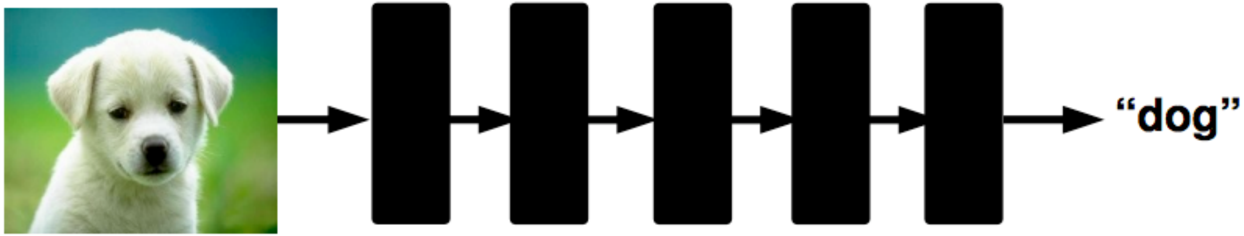
Supervised Learning: Examples



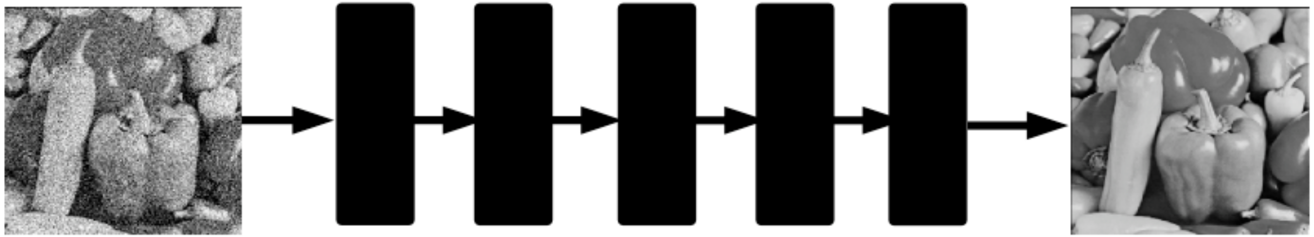
Slide: M. Ranzato

Supervised Deep Learning

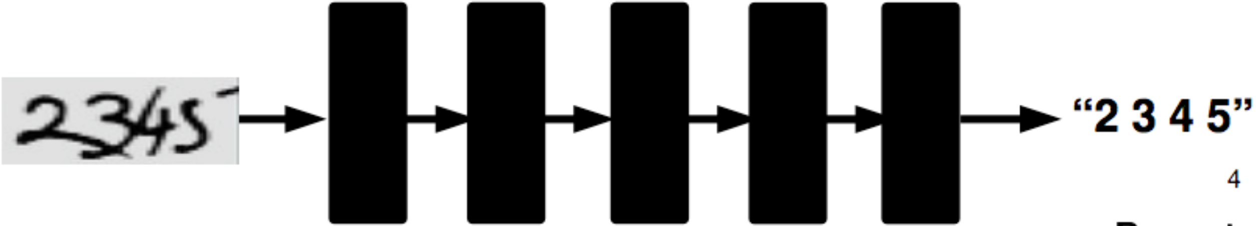
Classification



Denoising



OCR



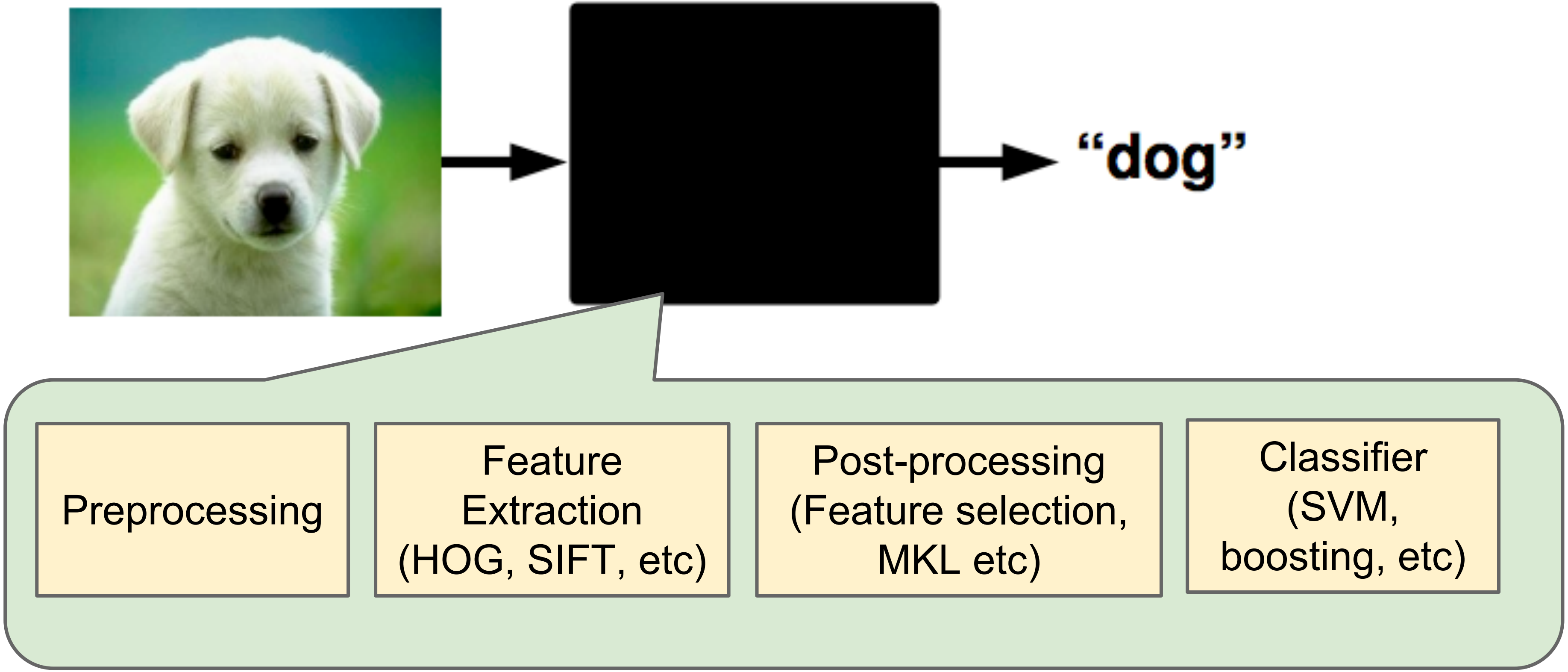
Ranzato

So deep learning is about learning feature representation in a compositional manner.

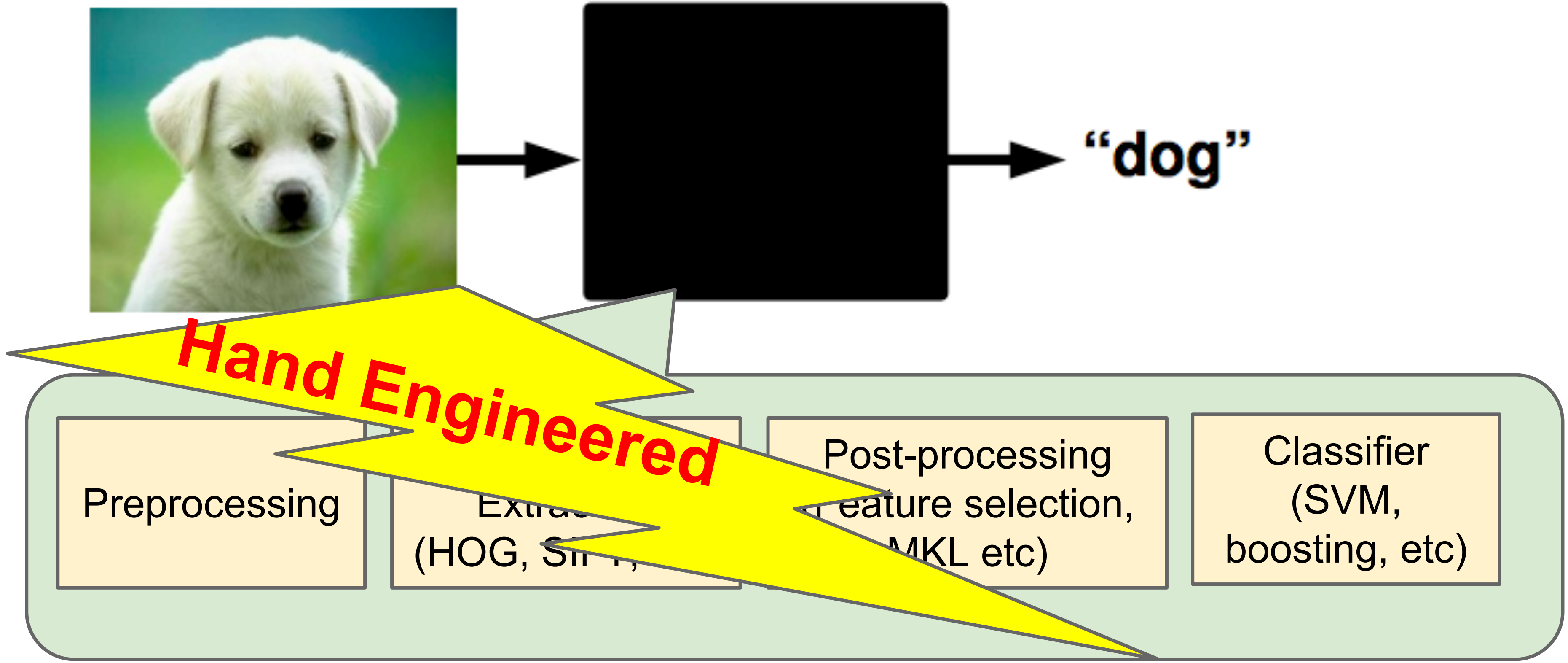
But wait,

why learn features?

The Black Box in a Traditional Recognition Approach



The Black Box in a Traditional Recognition Approach



Preprocessing | Feature Extraction (HOG, SIFT, etc) | Post-processing (Feature selection, MKL etc)

- Most critical for accuracy
- Most time-consuming in development
- What is the best feature???
- What is next?? Keep on crafting better features?

⇒ Let’s learn feature representation directly from data.

Learn features and classifier *together*

⇒ Learn an end-to-end recognition system.

A non-linear map that takes raw pixels directly to labels.

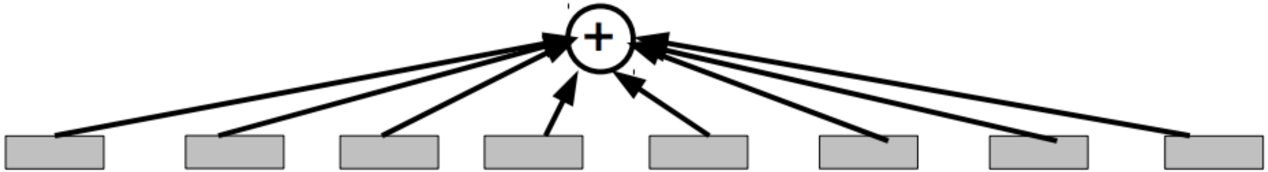
Q: How can we build such a highly non-linear system?

A: By combining simple building blocks we can make more and more complex systems.

Slide: M. Ranzato

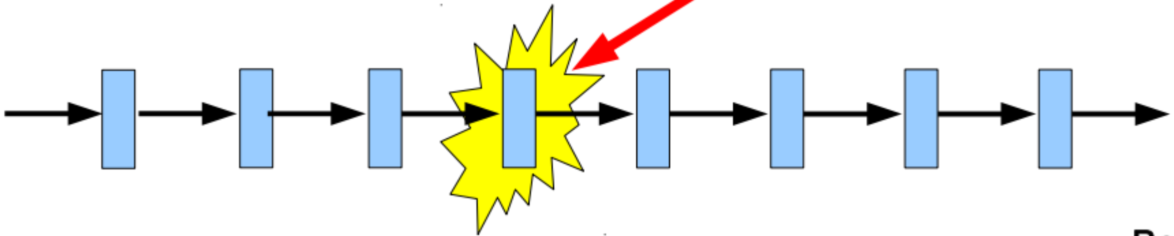
Building a complicated function

Proposal #1:



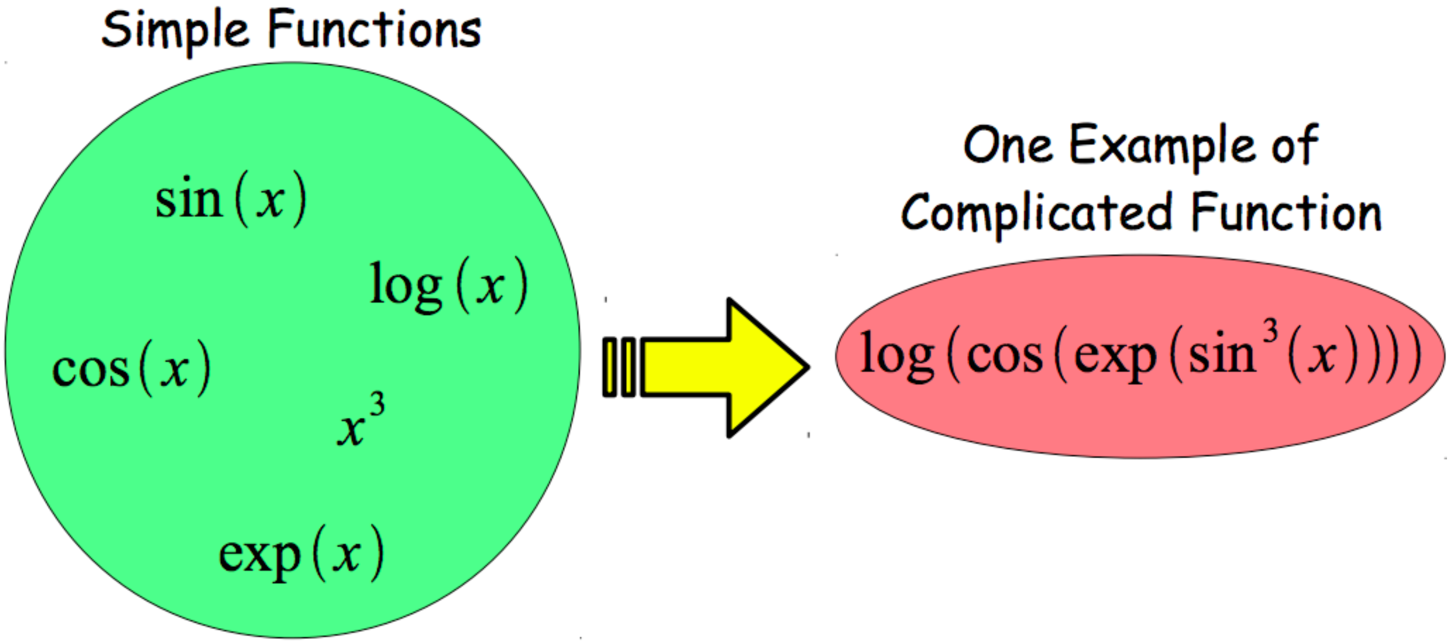
Each box is a simple nonlinear function

Proposal #2:



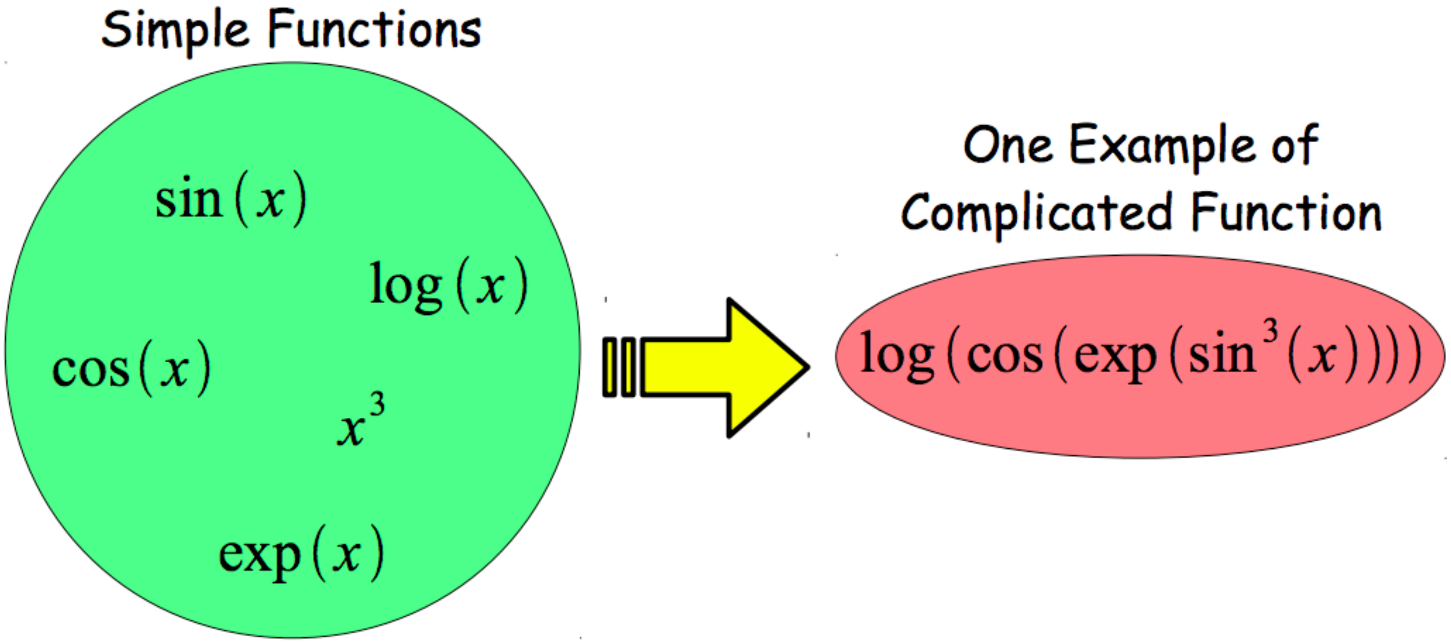
Ranzato

Building a complicated function



Slide: M. Ranzato

Building a complicated function



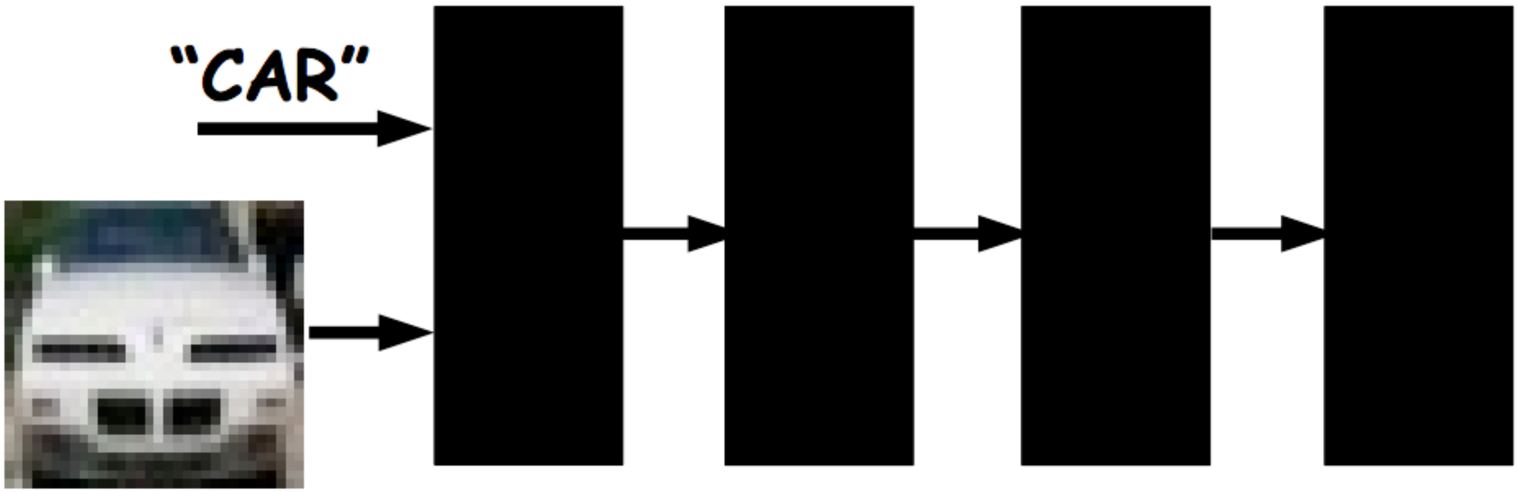
- Composition is at the core of deep learning methods
- Each "simple function" will have parameters subject to learning

Slide: M. Ranzato

Intuition behind Deep Neural Nets

Slide: M. Ranzato

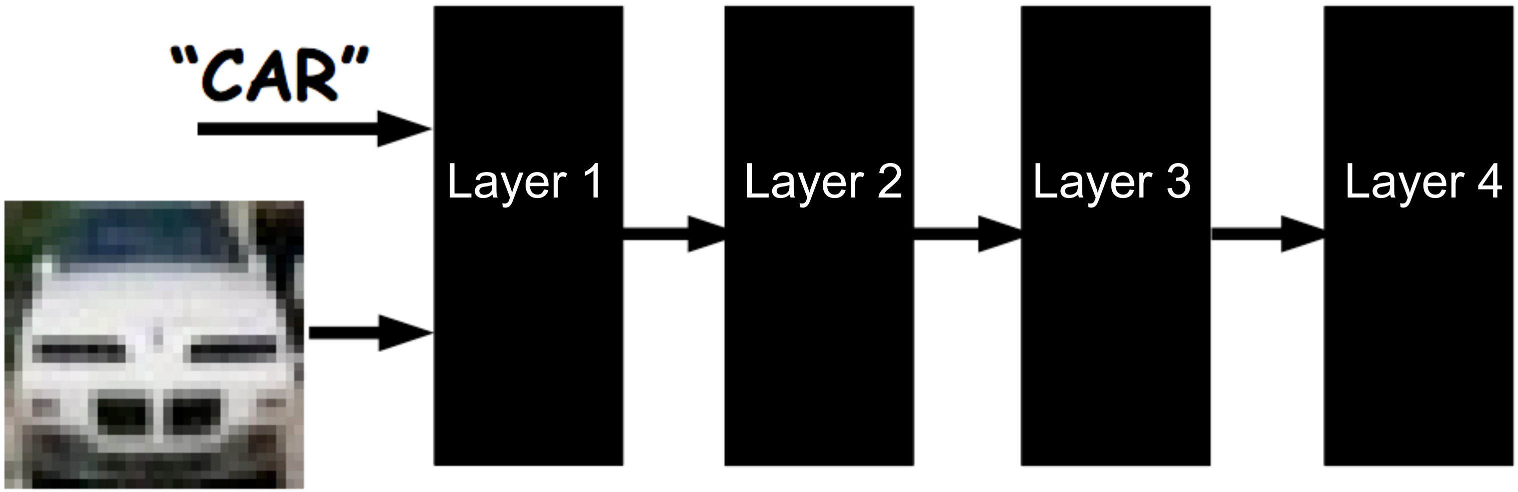
Intuition behind Deep Neural Nets



NOTE: Each black box can have trainable parameters.
Their composition makes a highly non-linear system.

Slide: M. Ranzato

Intuition behind Deep Neural Nets



NOTE: Each black box can have trainable parameters.
Their composition makes a highly non-linear system.

The final layer outputs a probability distribution of categories.

Slide: M. Ranzato

A simple single layer Neural Network

Consists of a linear combination of input through a nonlinear function:

$$\begin{aligned} z &= Wx + b \\ a &= f(z) \end{aligned}$$

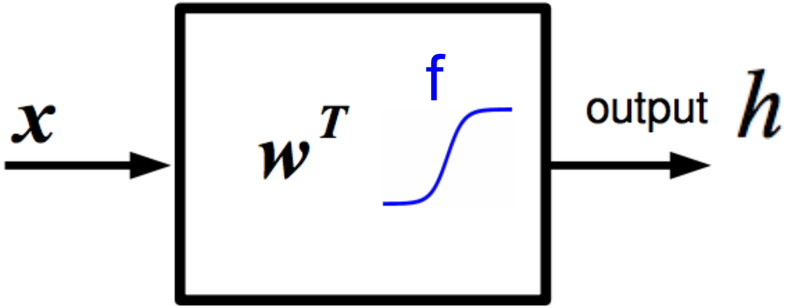
W is the weight parameter to be learned.

x is the output of the previous layer

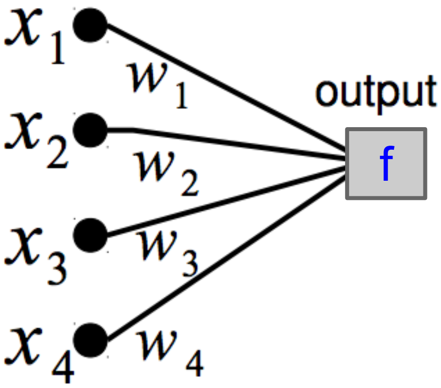
f is a simple nonlinear function. Popular choice is max(x,0), called ReLu (Rectified Linear Unit)

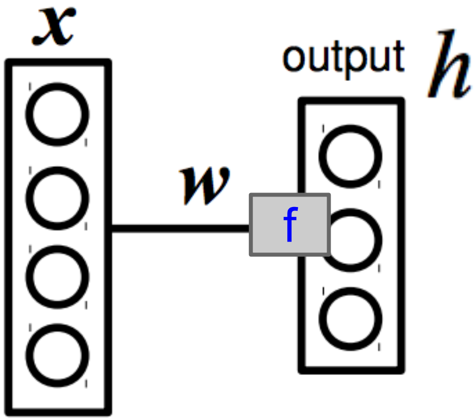
1 layer: Graphical Representation

$f(X;W) =$

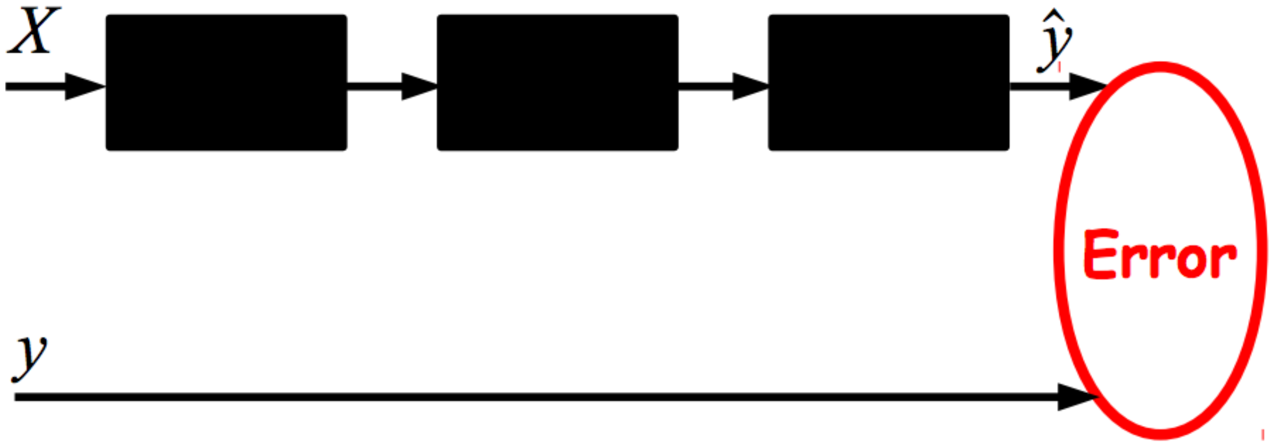


h is called a neuron, hidden unit or feature.





Joint training architecture overview



NOTE: Multi-layer neural nets with more than two layers are nowadays called **deep nets**!!

NOTE: User must specify number of layers, number of hidden units, type of layers and loss function.

Ranzato

Neural Net Training

A) Compute loss on small mini-batch

Slide: M. Ranzato

Neural Net Training

A) Compute loss on small mini-batch

Slide: M. Ranzato

Neural Net Training

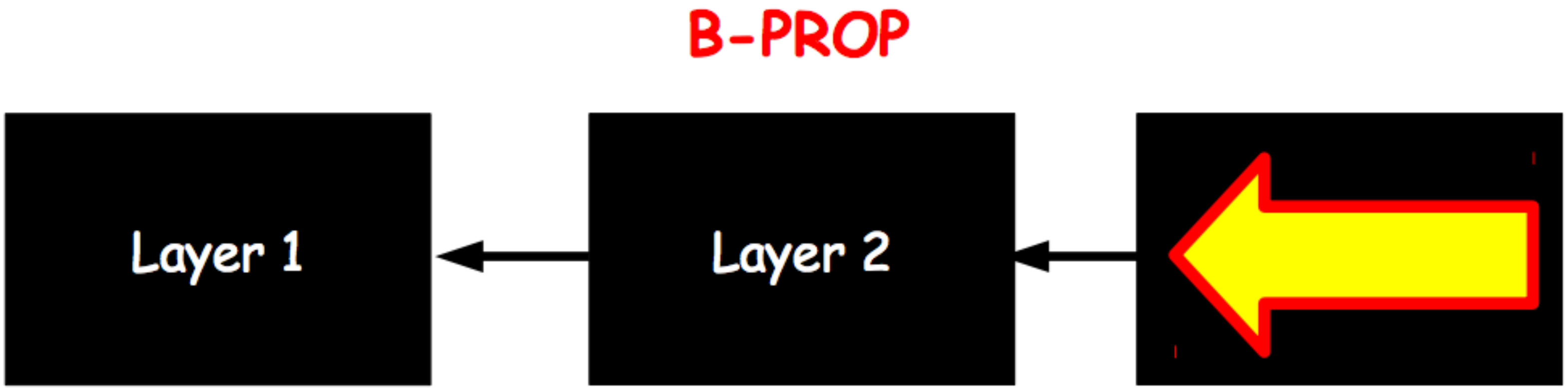
A) Compute loss on small mini-batch

Slide: M. Ranzato

Neural Net Training

A) Compute loss on small mini-batch

B) Compute gradient w.r.t. parameters

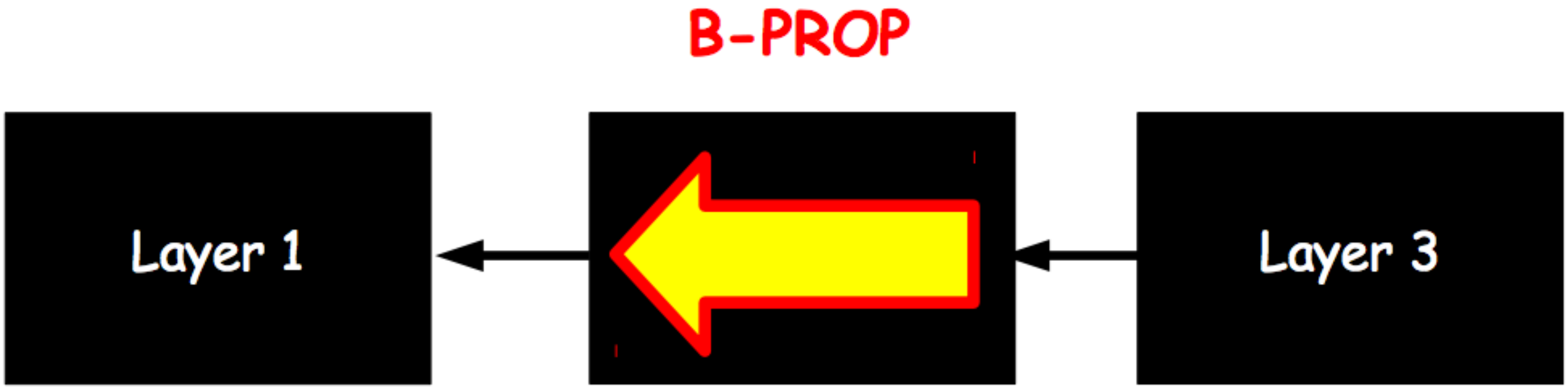


Slide: M. Ranzato

Neural Net Training

A) Compute loss on small mini-batch

B) Compute gradient w.r.t. parameters

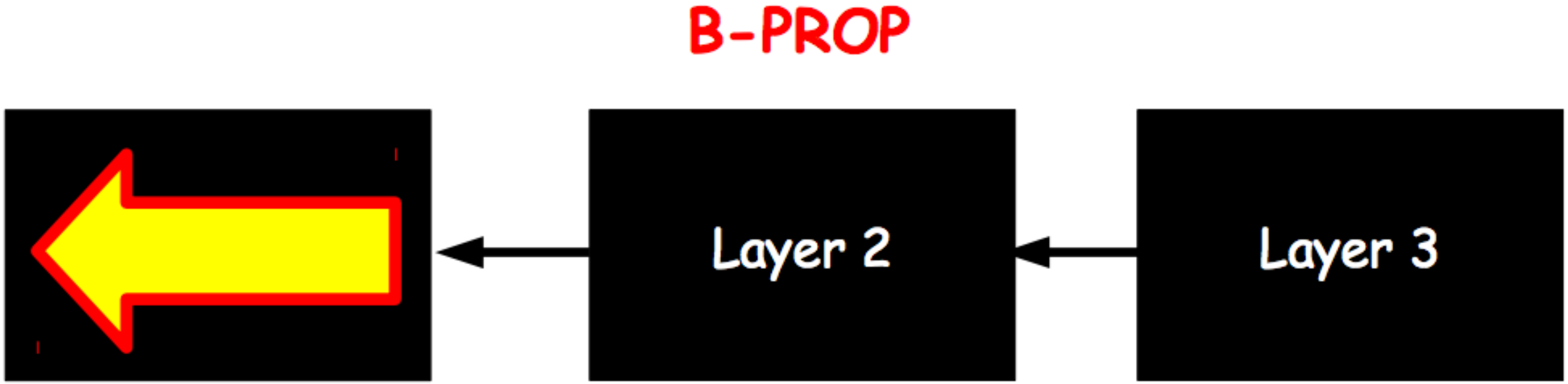


Slide: M. Ranzato

Neural Net Training

A) Compute loss on small mini-batch

B) Compute gradient w.r.t. parameters



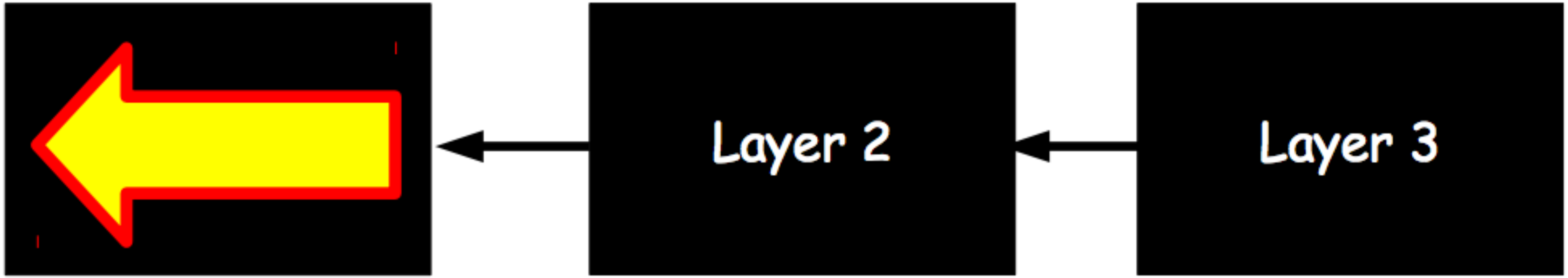
Slide: M. Ranzato

Neural Net Training

A) Compute loss on small mini-batch

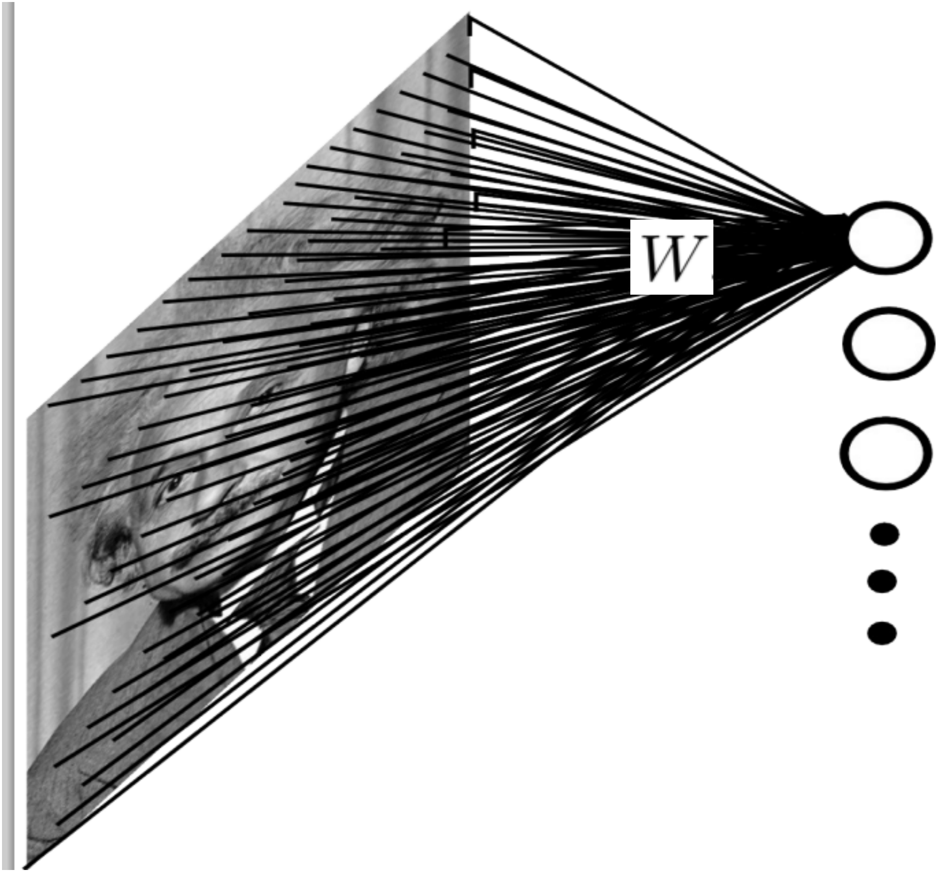
B) Compute gradient w.r.t. parameters

C) Use gradient to update parameters $W \leftarrow W - \eta \frac{dL}{dW}$



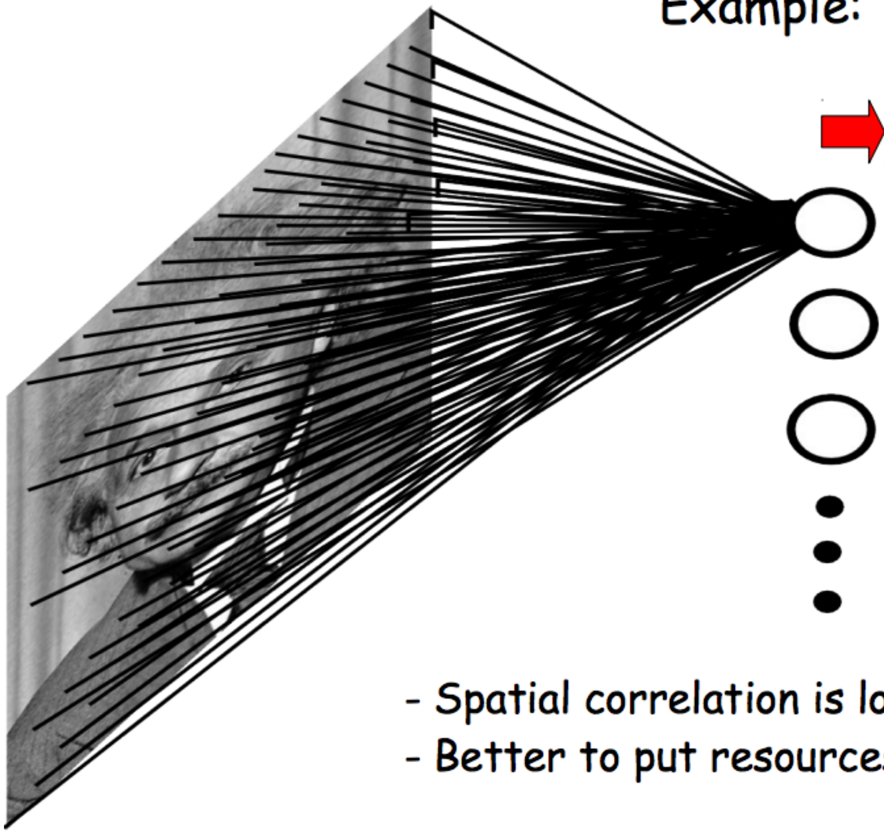
Slide: M. Ranzato

When the input data is an image..



Slide: M. Ranzato

When the input data is an image..



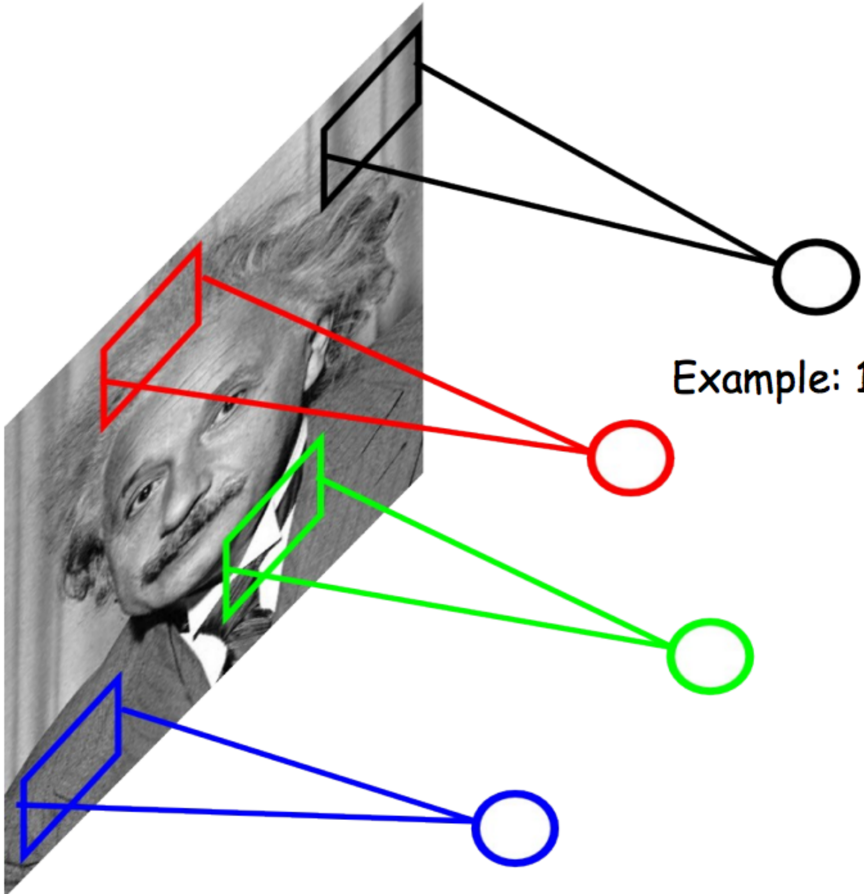
Example: 1000x1000 image

1M hidden units

→ 1B parameters!!!

- Spatial correlation is local
- Better to put resources elsewhere!

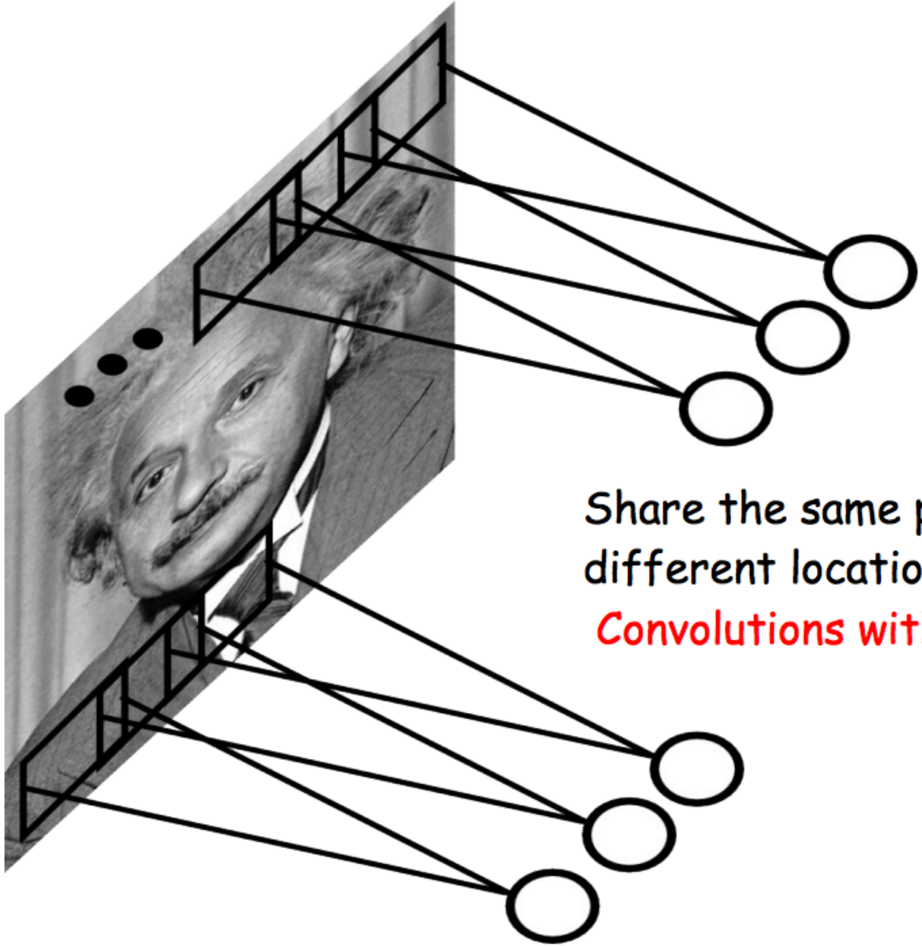
Reduce connection to local regions



Example: 1000x1000 image
1M hidden units
Filter size: 10x10
10M parameters

Ranzato

Reuse the same kernel everywhere

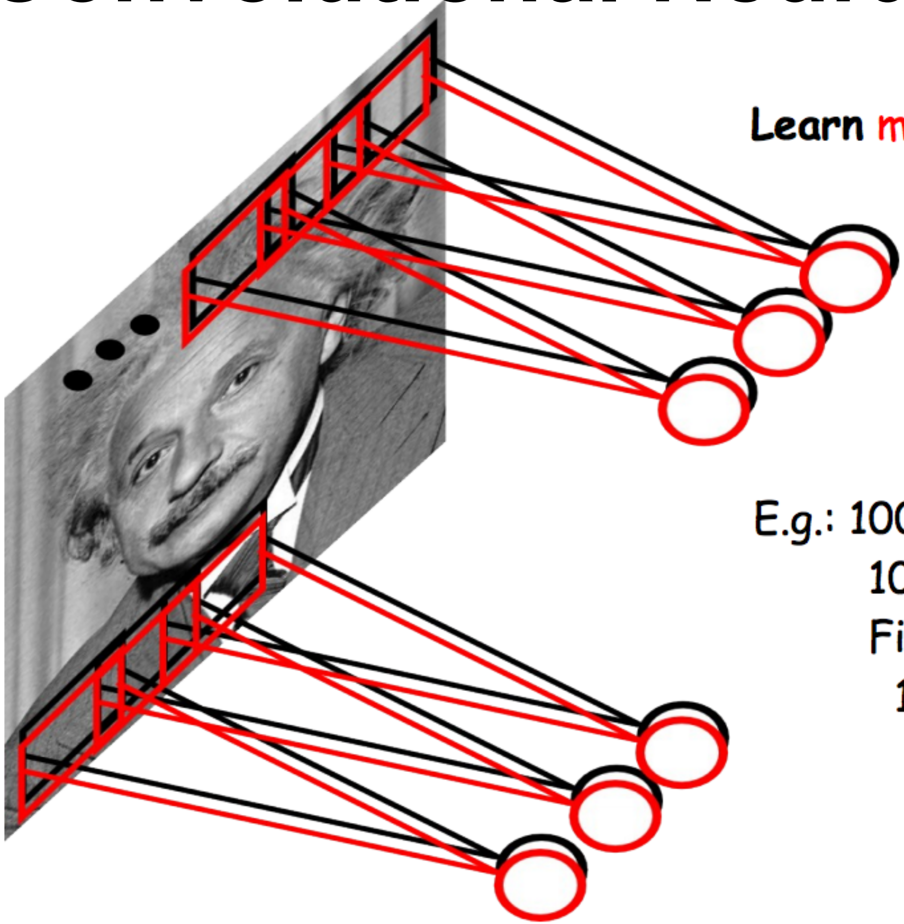


Because interesting features (edges) can happen at anywhere in the image.

Share the same parameters across different locations:
Convolutions with learned kernels

Ranzato

Convolutional Neural Nets



Learn multiple filters.

E.g.: 1000x1000 image
100 Filters
Filter size: 10x10
10K parameters

Ranzato

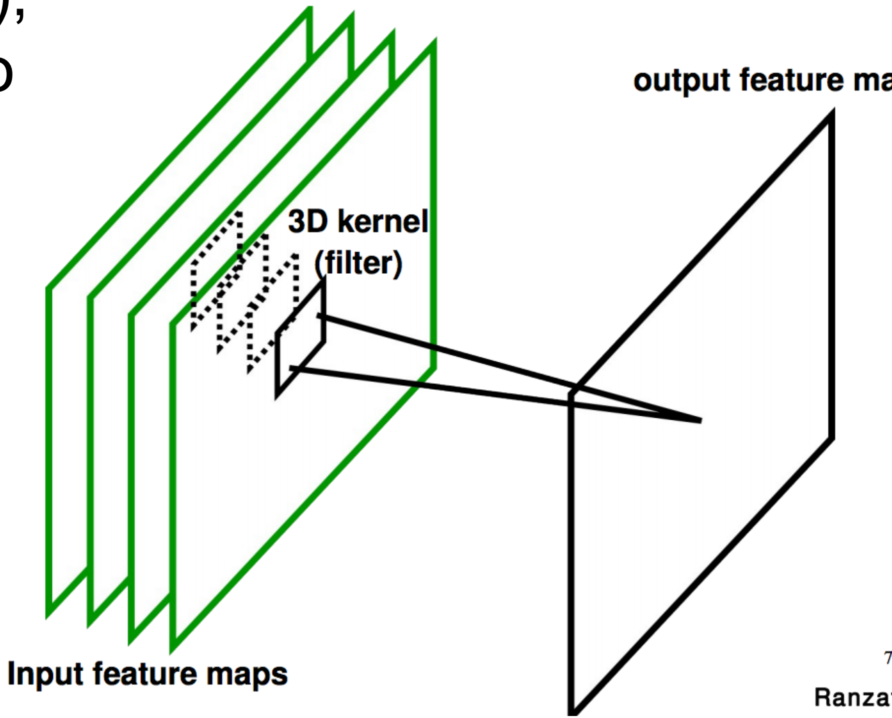
LeCun et al. “Gradient-based learning applied to document recognition” IEEE 1998

Detail

If the input has 3 channels (R,G,B), 3 separate k by k filter is applied to each channel.

Output of convolving 1 feature is called a *feature map*.

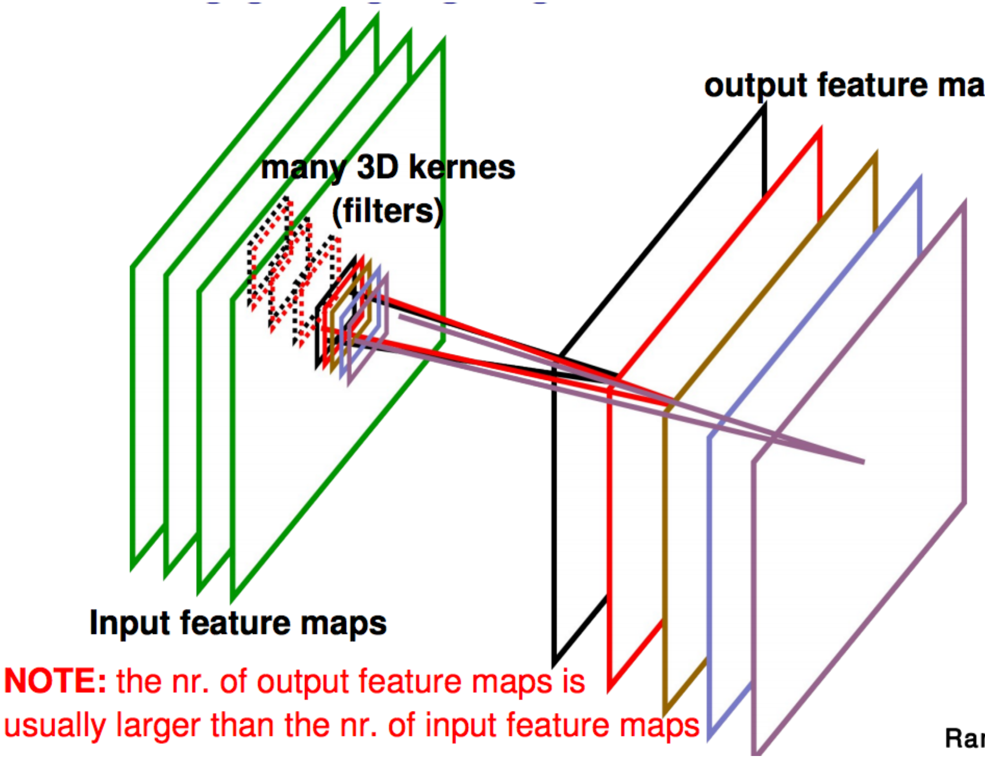
This is just sliding window, ex. the output of one part filter of DPM is a feature map



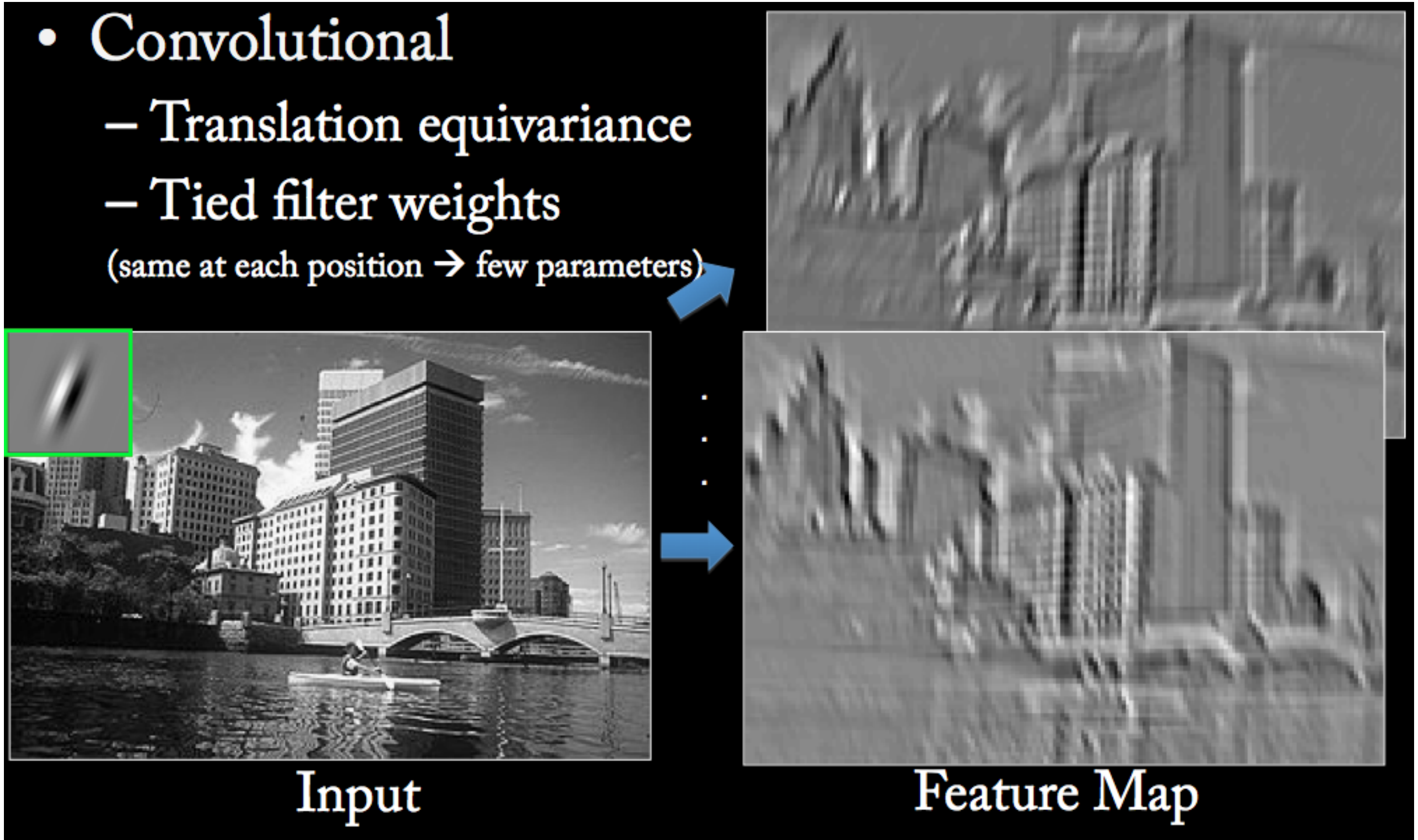
Using multiple filters

Each filter detects features in the output of previous layer.

So to capture different features, learn multiple filters.

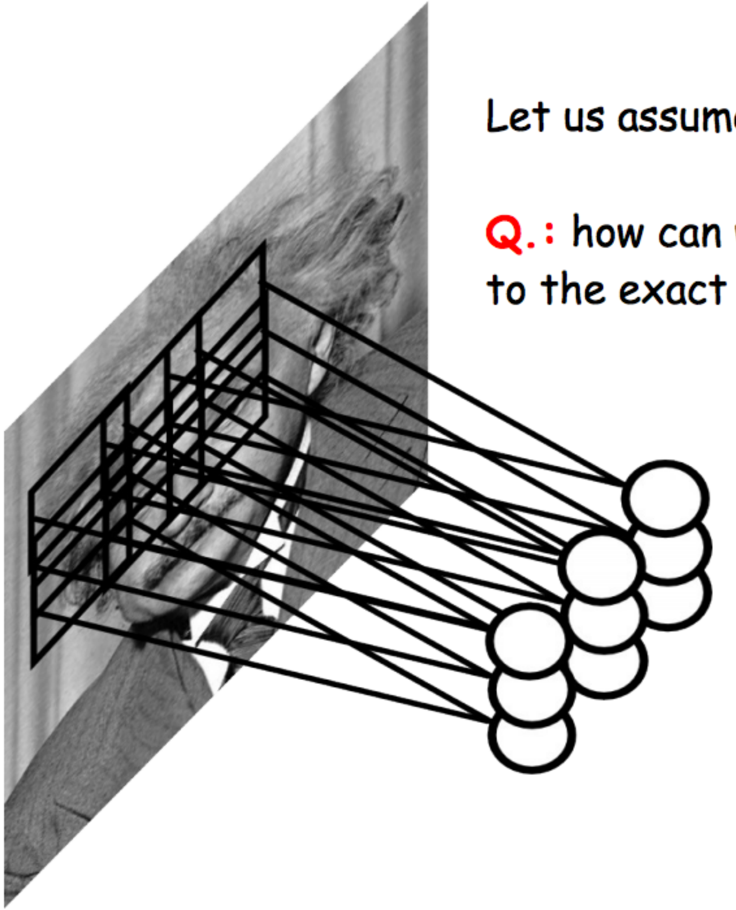


Example of filtering



Slide: R. Fergus

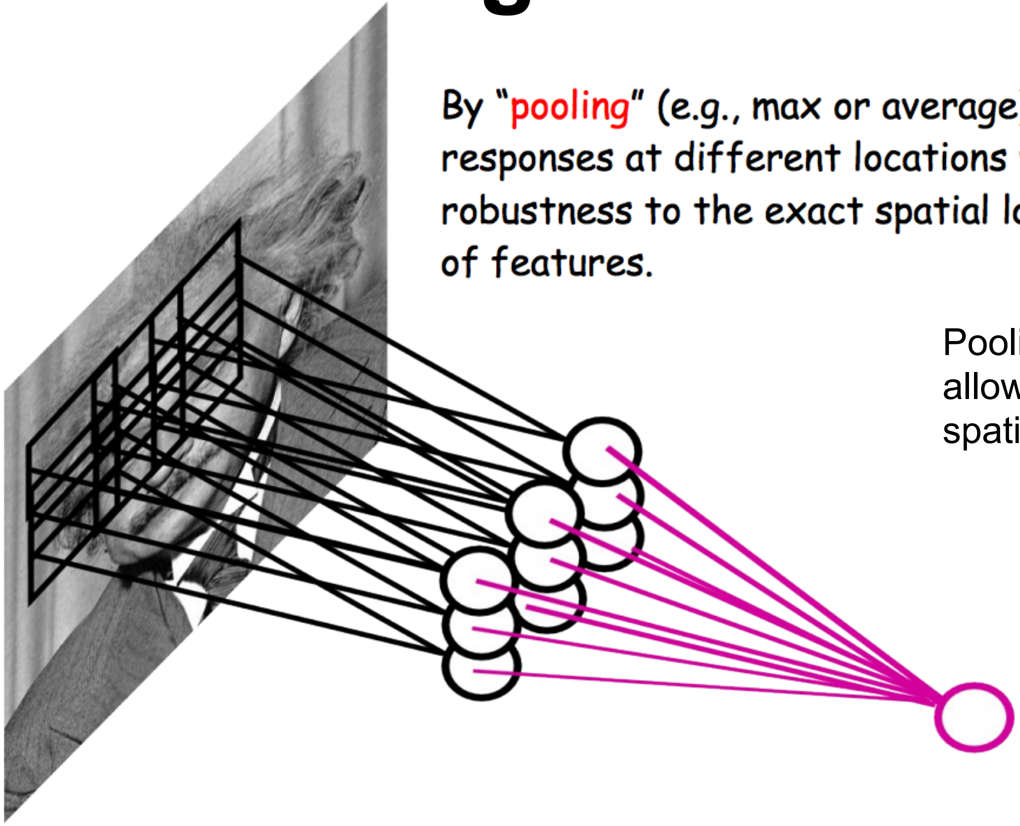
Building Translation Invariance



Let us assume filter is an "eye" detector.

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

Building Translation Invariance via Spatial Pooling



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

Pooling also subsamples the image, allowing the next layer to look at larger spatial regions.

Ranzato

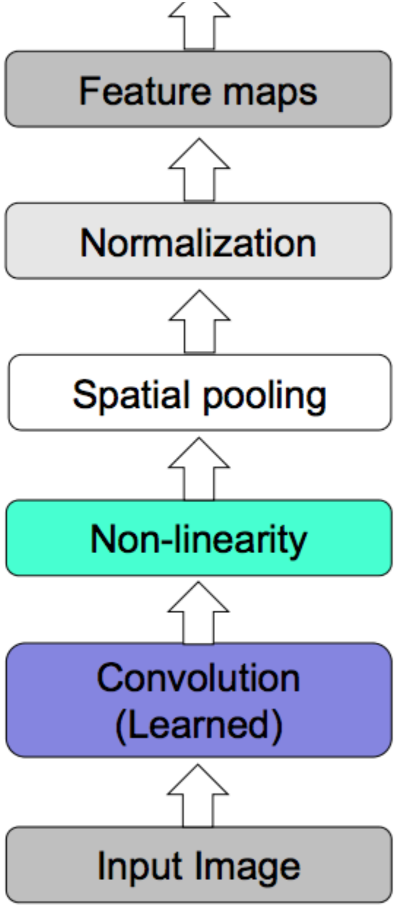
Summary of a typical convolutional layer

Doing all of this consists one layer.

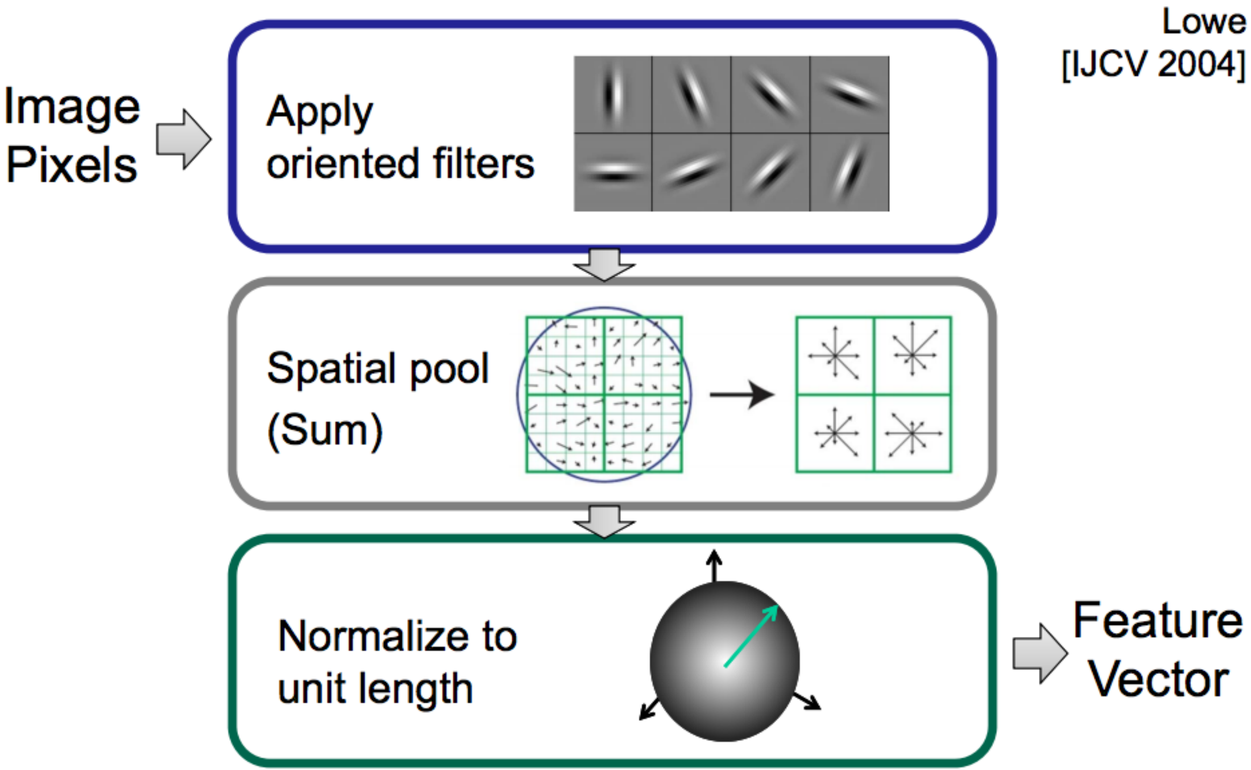
- Pooling and normalization is optional.

Stack them up and train just like multi-layer neural nets.

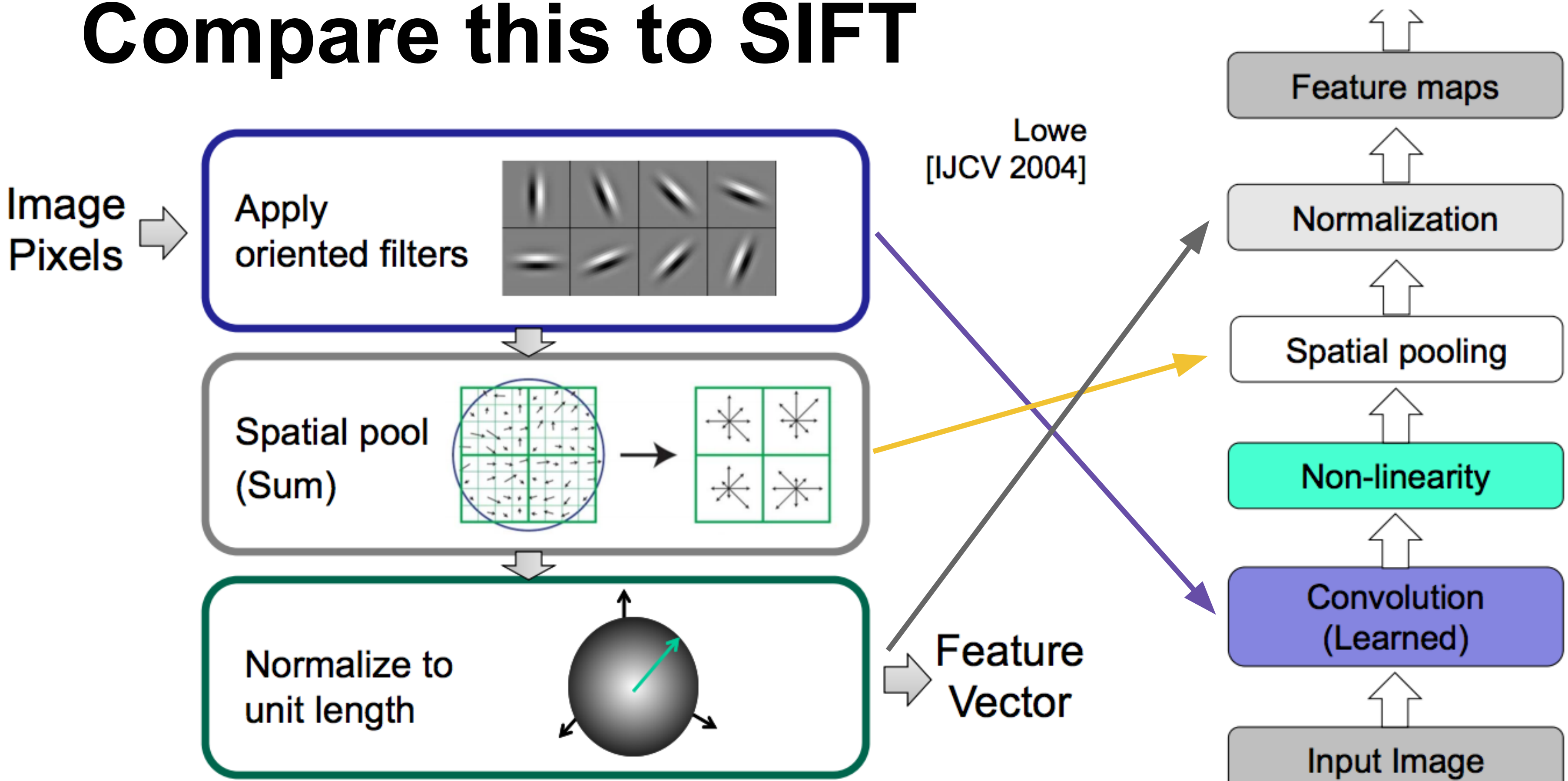
Final layer is usually fully connected neural net with output size == number of classes



Compare this to SIFT



Compare this to SIFT



Revisiting the composition idea

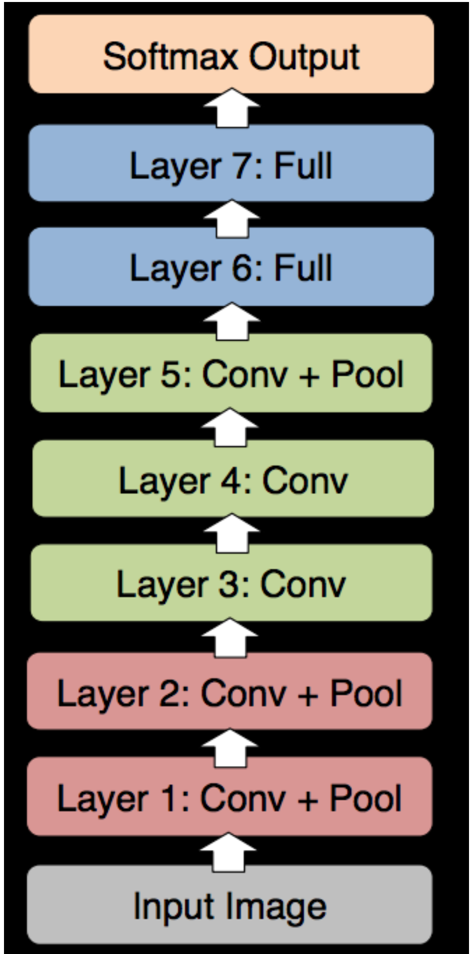
Every layer learns a feature detector by combining the output of the layer before.

⇒ More and more abstract features are learned as we stack layers.

Keep this in mind and let’s look at what kind of things ConvNets learn.

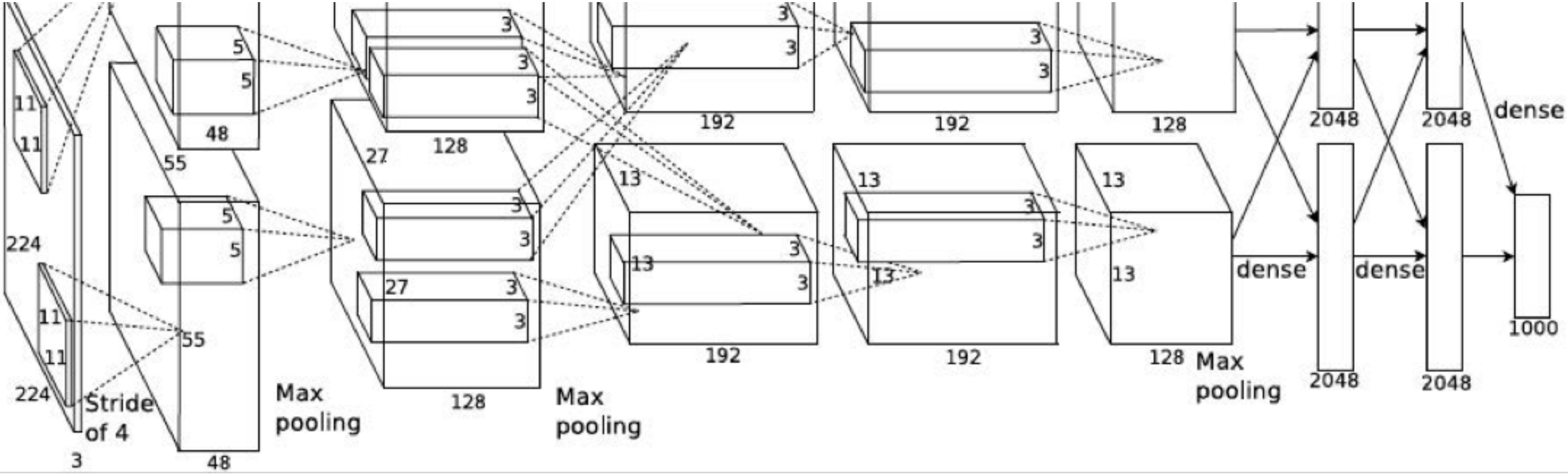
Architecture of Alex Krizhevsky et al.

- 8 layers total.
- Trained on Imagenet Dataset (1000 categories, 1.2M training images, 150k test images)
- 18.2% top-5 error
  - Winner of the ILSVRC-2012 challenge.



Slide: R. Fergus

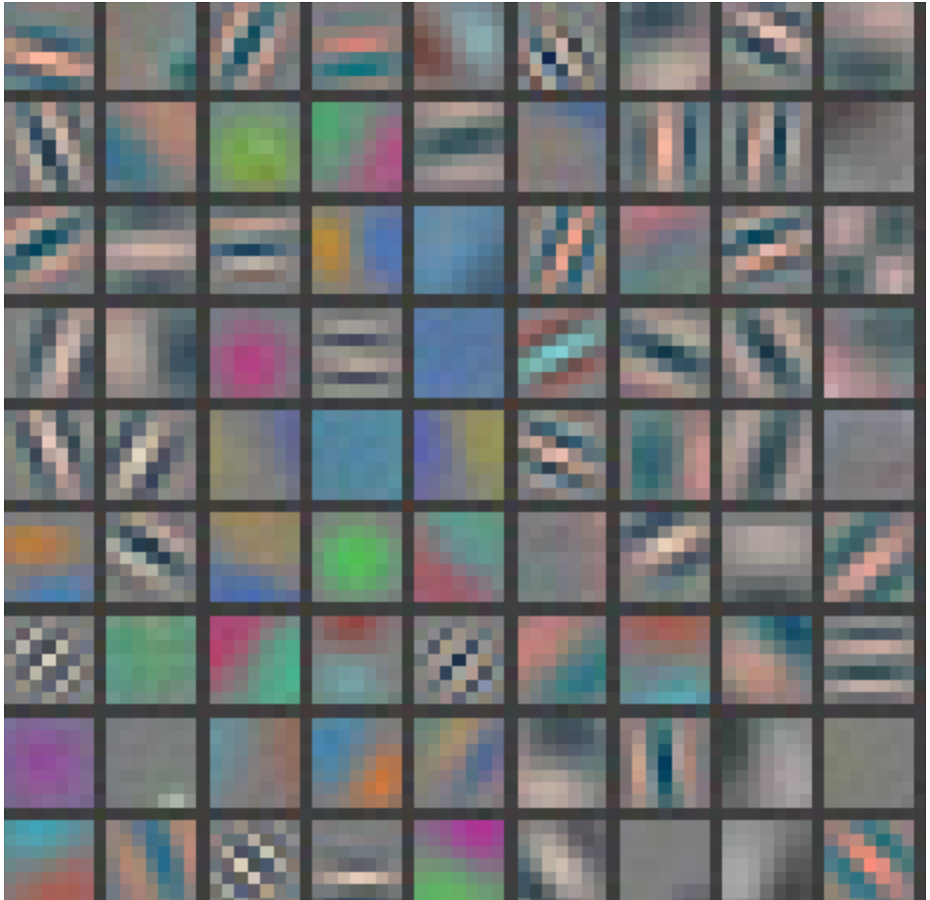
Architecture of Alex Krizhevsky et al.



First layer filters

Showing 81 filters of 11x11x3.

Capture low-level features like oriented edges, blobs.



Note these oriented edges are analogous to what SIFT uses to compute the gradients.

Top 9 patches that activate each filter in layer 1

Each 3x3 block shows the top 9 patches for one filter.

Layer 2: Top-9 Patches

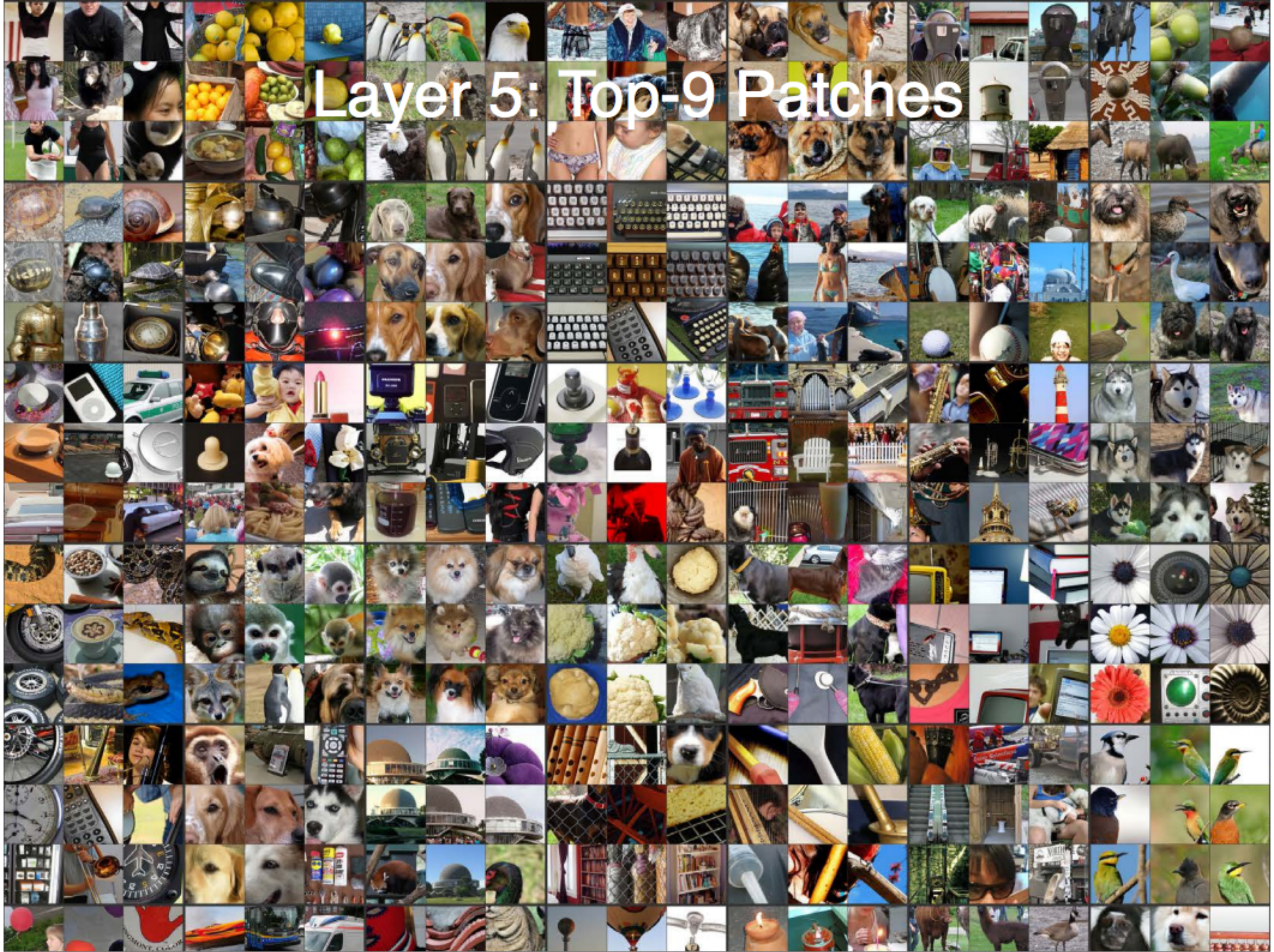
Layer 2: Top-9 Patches

Note how the previous low-level features are combined to detect a little more abstract features like textures.


Layer 3: Top-9 Patches

Layer 3: Top-9 Patches

Layer 4: Top-9 Patches


Layer 5: Top-9 Patches

ConvNets as generic feature extractor

- A well-trained ConvNets is an excellent feature extractor.
- Chop the network at desired layer and use the output as a feature representation to train a SVM on some other vision dataset.

	Cal-101 (30/class)	Cal-256 (60/class)
SVM (1)	44.8 ± 0.7	24.6 ± 0.4
SVM (2)	66.2 ± 0.5	39.6 ± 0.3
SVM (3)	72.3 ± 0.4	46.0 ± 0.3
SVM (4)	76.6 ± 0.4	51.3 ± 0.1
SVM (5)	$\mathbf{86.2 \pm 0.8}$	65.6 ± 0.3
SVM (7)	$\mathbf{85.5 \pm 0.4}$	$\mathbf{71.7 \pm 0.2}$
Softmax (5)	82.9 ± 0.4	65.7 ± 0.5
Softmax (7)	$\mathbf{85.4 \pm 0.4}$	$\mathbf{72.6 \pm 0.1}$

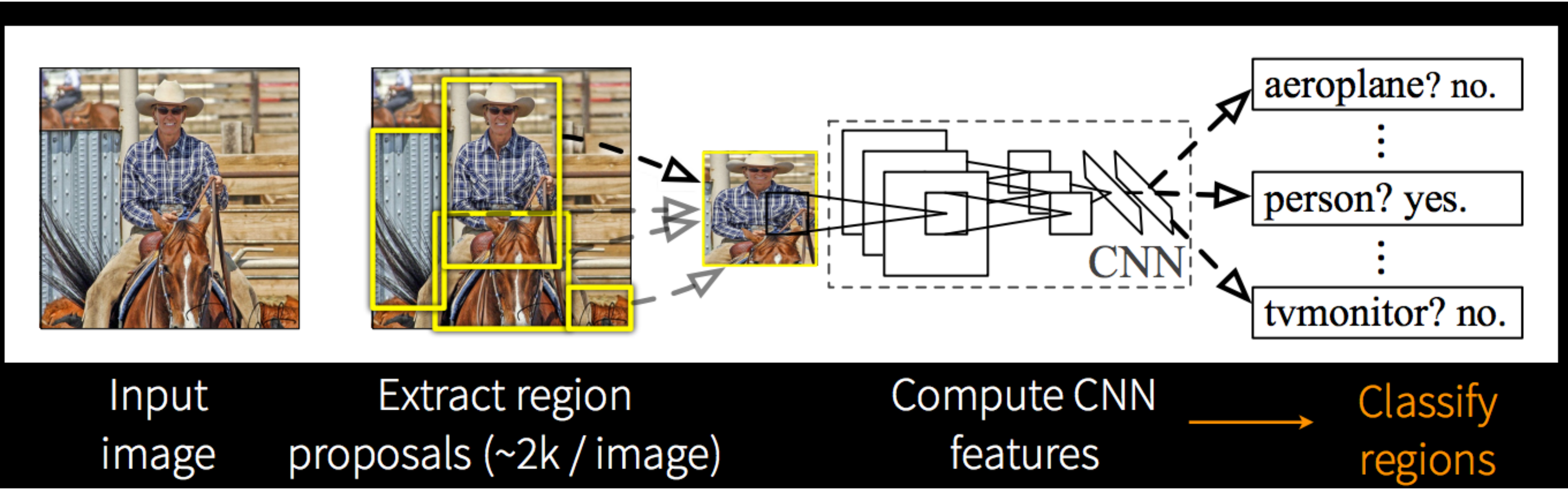
- Improve further by taking a pre-trained ConvNet and re-training it on a different dataset. Called *fine-tuning*

One way to do detection with ConvNets

Since ConvNets extract discriminative features, one can crop images at the object bounding box and train a good SVM on each category.

⇒ Extract regions that are likely to have objects, then apply ConvNet + SVM on each and use the confidence to do maximum suppression.

R-CNN: Regions with CNN features



Best performing method on PASCAL 2012 Detection improving previous methods by 30%

Slide: [R. Girshick]

ConvNet Libraries

- Cuda-convnet (Alex Krizhevsky, Google)
- Caffe (Y. Jia, Berkeley, now Google)
    - Pre-trained Krizhevsky model available.
- Torch7 (Idiap, NYU, NEC)

more around.