Object recognition and computer vision

Convolutional neural networks for visual recognition

Ivan Laptev

http://www.di.ens.fr/~laptev

INRIA, WILLOW, ENS/INRIA/CNRS UMR 8548

Departement d'Informatique, Ecole Normale Supérieure, Paris

With slides from: R. Fergus, A. Vedaldi, S. Lazebnik, A. Karpathy and L. Fei Fei

Announcements

Assignment 2 was due last week.
http://www.di.ens.fr/willow/teaching/recvis17/assignment2/

Assignment 3 is due next week:
http://www.di.ens.fr/willow/teaching/recvis17/assignment3/

Topics for final projects will be released on Friday 17/11 this week. Please check the class website.

Final project

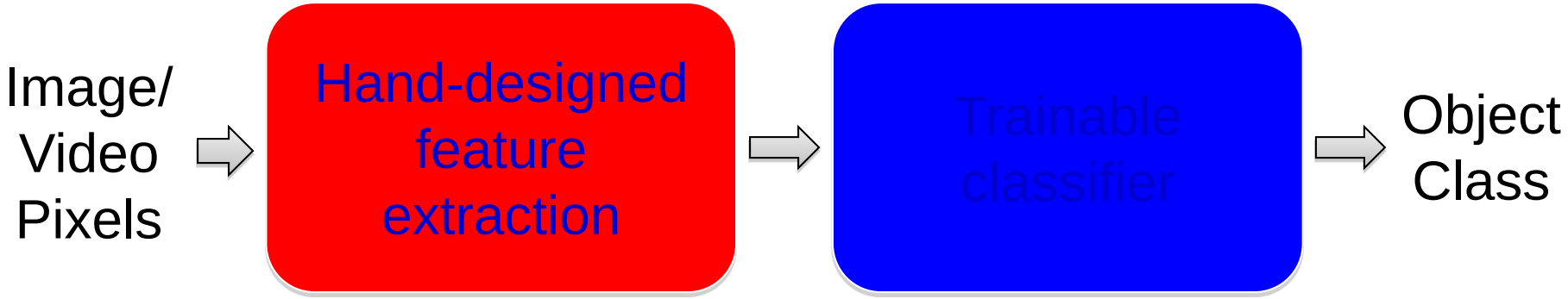
- Select the topic + write project proposal
- Get proposal OK:ed by us and do the project
- Present the work in the class
- Write project report

- Can be done individually or as a group **of max 2 people**
- The proposed project topics are from the recent top-conferences publications in computer vision, see example topics from 2016 here: http://www.di.ens.fr/willow/teaching/recvis16/
- Student-defined projects are welcome
- Final project can be joint with another MVA course

Outline

1. Convolutional neural networks (CNNs)
2. Understanding and visualizing CNN representations
3. Transferring learnt representation to other tasks
4. Typical CNN architectures for image classification
5. Beyond classification

Traditional Recognition Approach

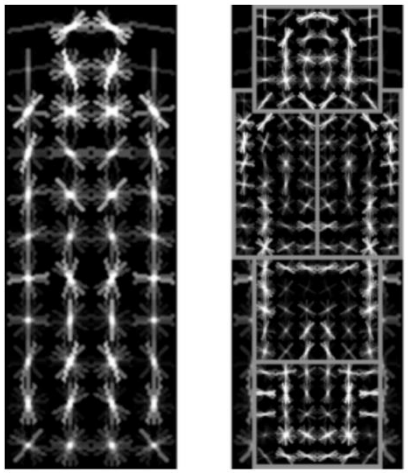


- Features are not learned
- Trainable classifier is often generic (e.g. SVM)

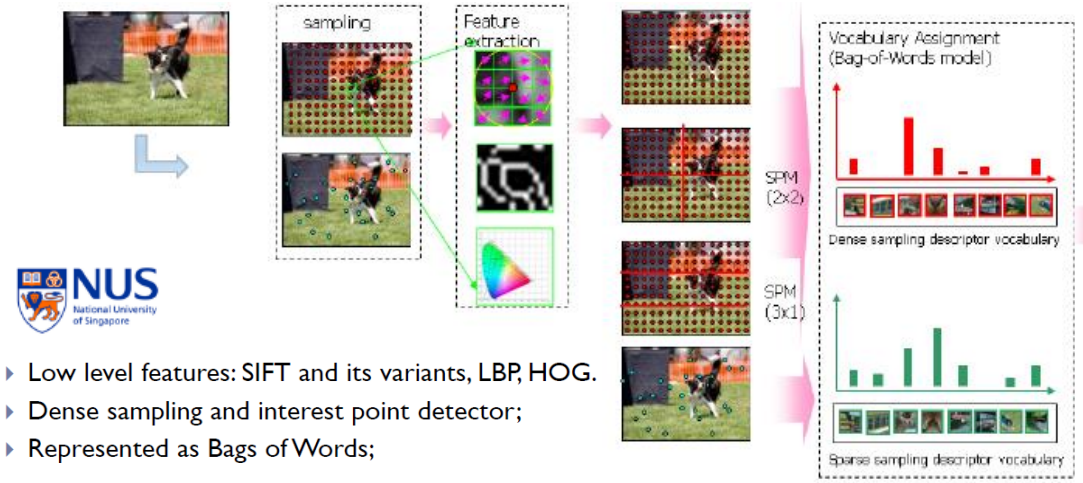
Slide: R. Fergus / S. Lazebnik

Traditional Recognition Approach

- Features are key to recent progress in recognition
- Multitude of hand-designed features currently in use
  - SIFT, HOG, ……
- Where next? Better classifiers? Or keep building more features?



Felzenszwalb, Girshick, McAllester and Ramanan, PAMI 2007



Yan & Huang
(Winner of PASCAL 2010 classification competition)

Slide: R. Fergus / S. Lazebnik

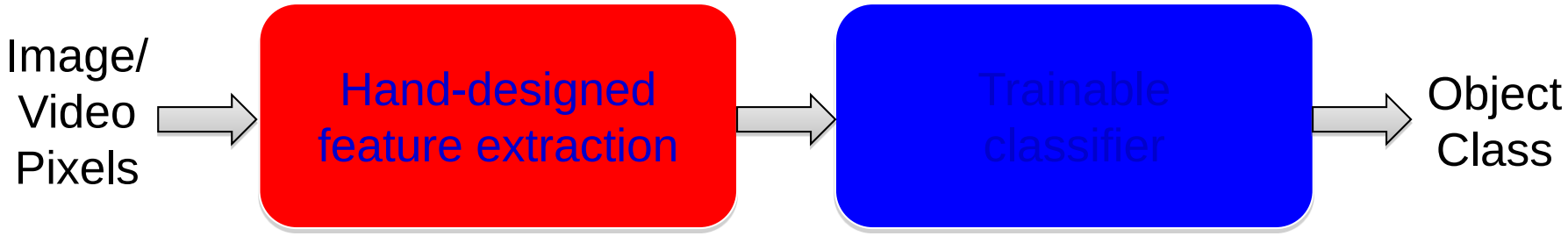
What about learning the features?

- Learn a *feature hierarchy* all the way from pixels to classifier
- Each layer extracts features from the output of previous layer
- Train all layers jointly

Slide: R. Fergus / S. Lazebnik

“Shallow” vs. “deep” architectures

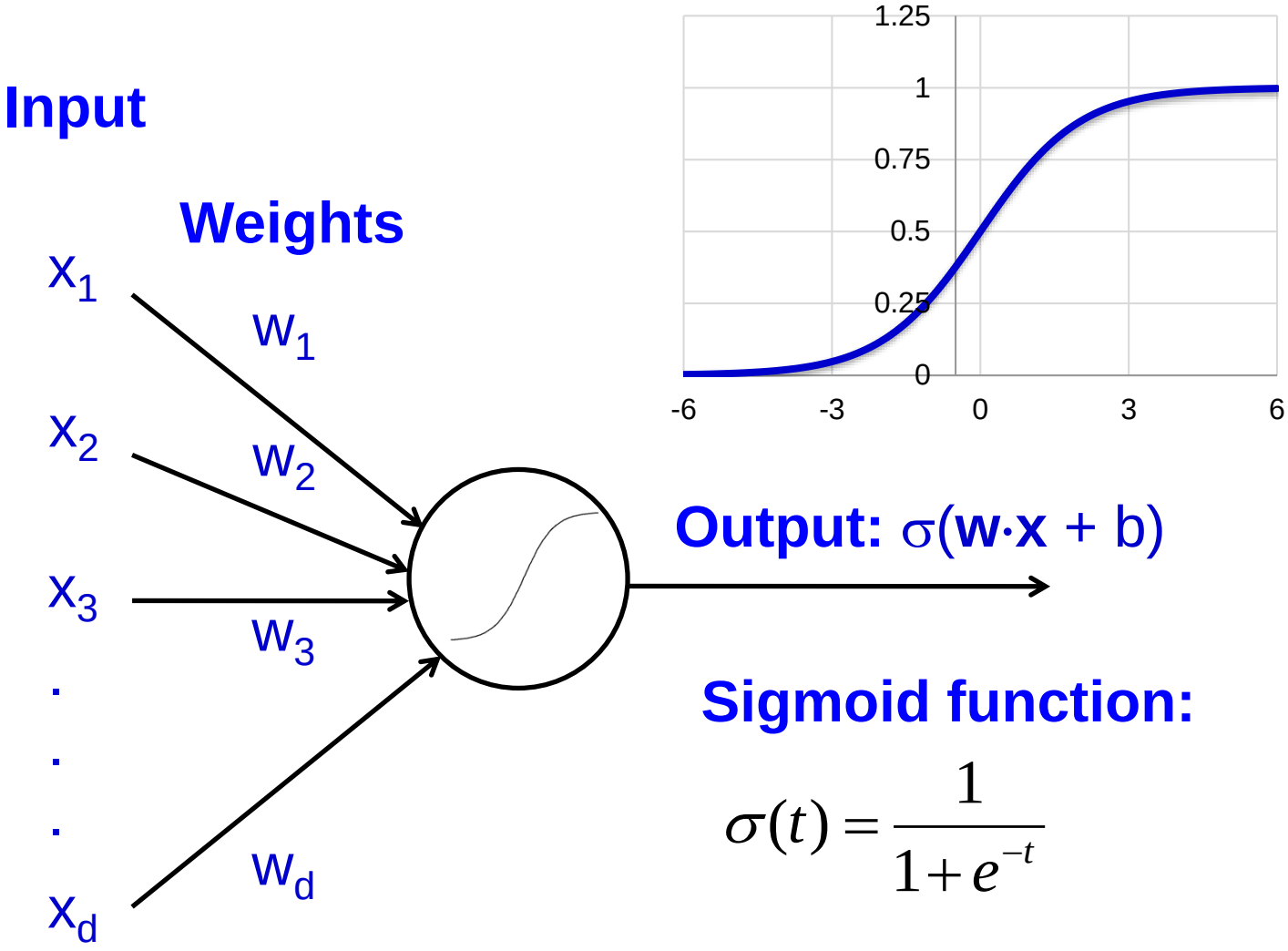
Traditional recognition: “Shallow” architecture



Deep learning: “Deep” architecture

Slide: R. Fergus / S. Lazebnik

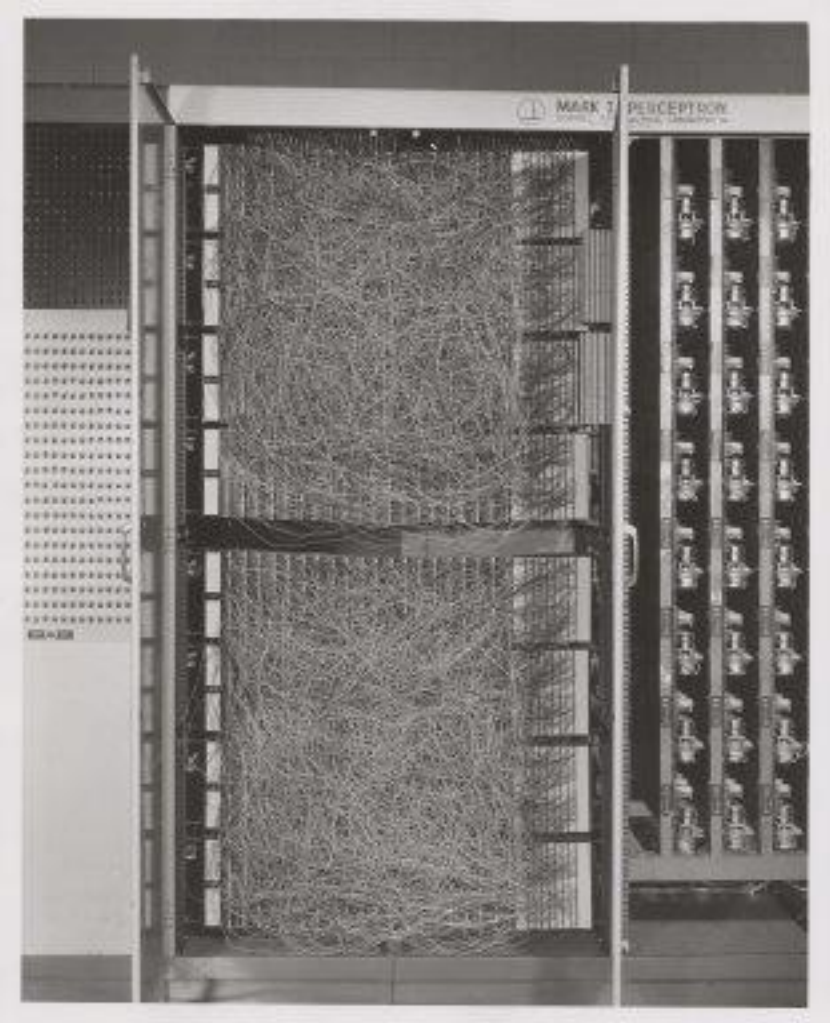
Background: Perceptrons



[Rosenblatt, 1957]

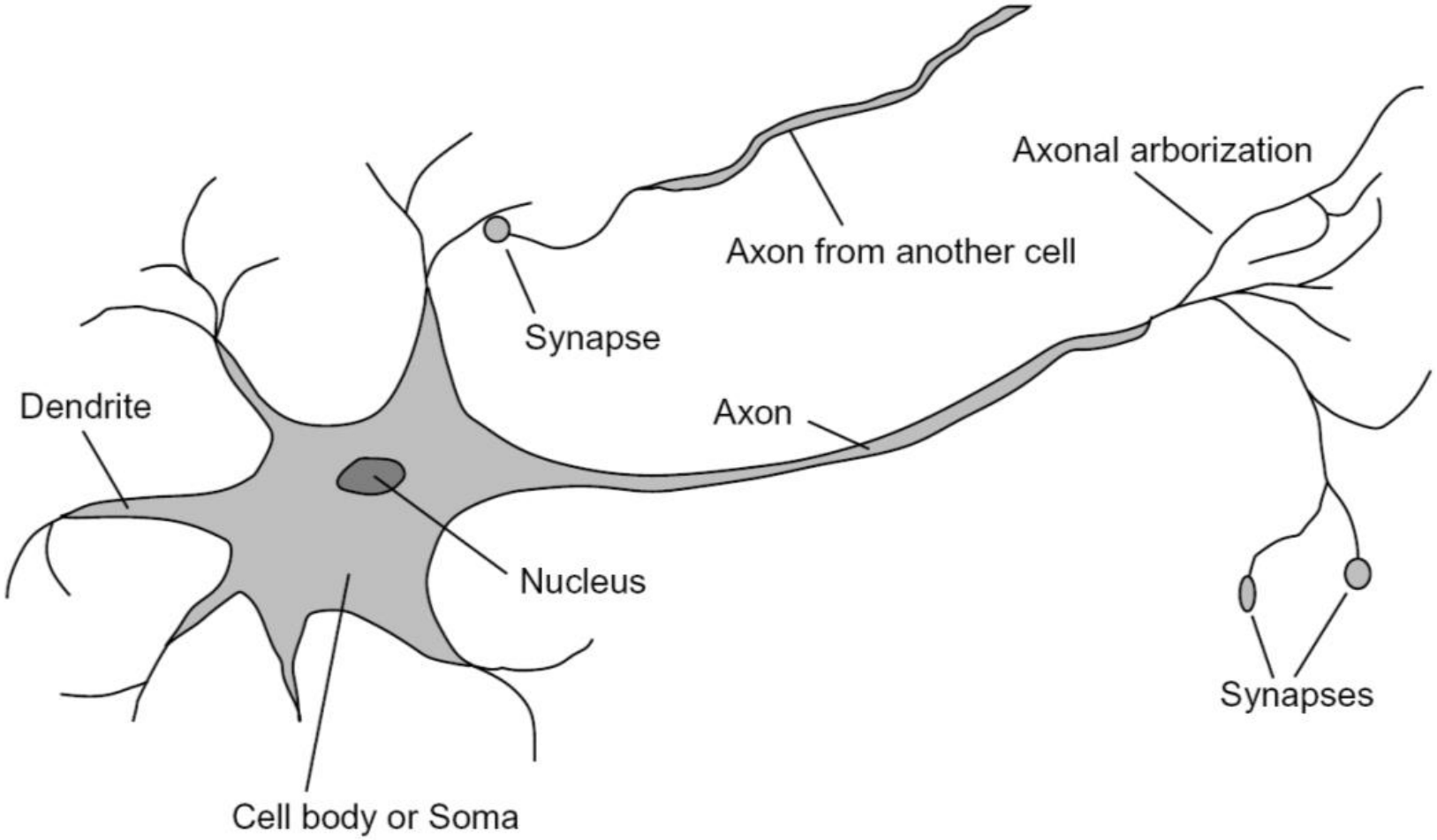
Slide: R. Fergus / S. Lazebnik

Background: Perceptrons



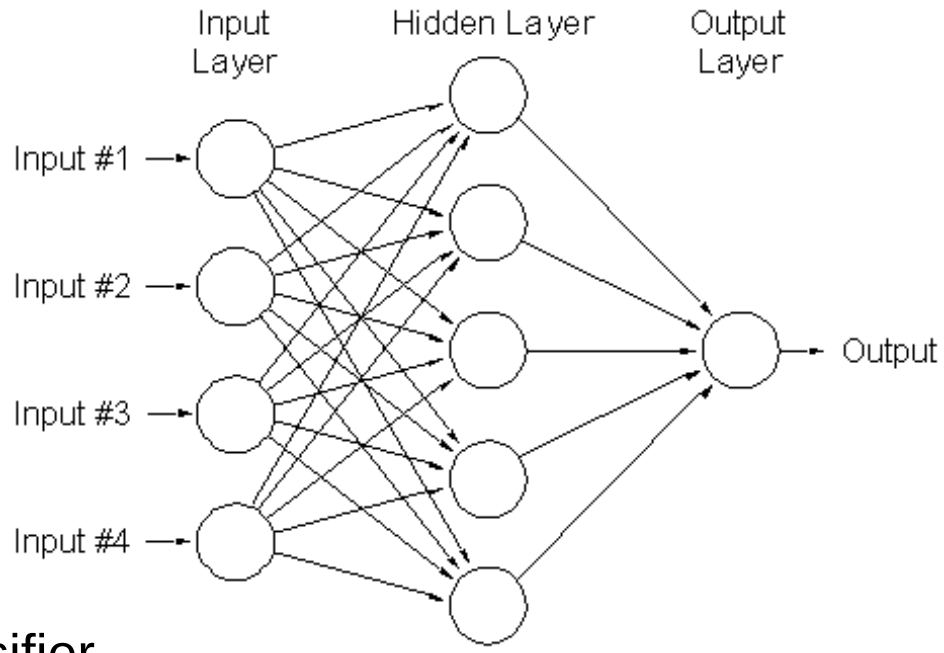
[Mark I Percepetron]:
Connected to a 20x20 pixel array of photocells

Inspiration: Neuron cells



Slide: R. Fergus / S. Lazebnik

Background: Multi-Layer Neural Networks



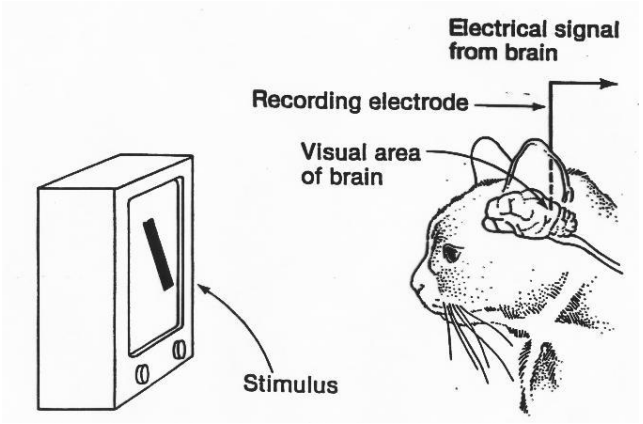
- Nonlinear classifier
- **Training:** find network weights **w** to minimize the error between true training labels y_i and estimated labels $f_{\mathbf{w}}(\mathbf{x}_i)$:

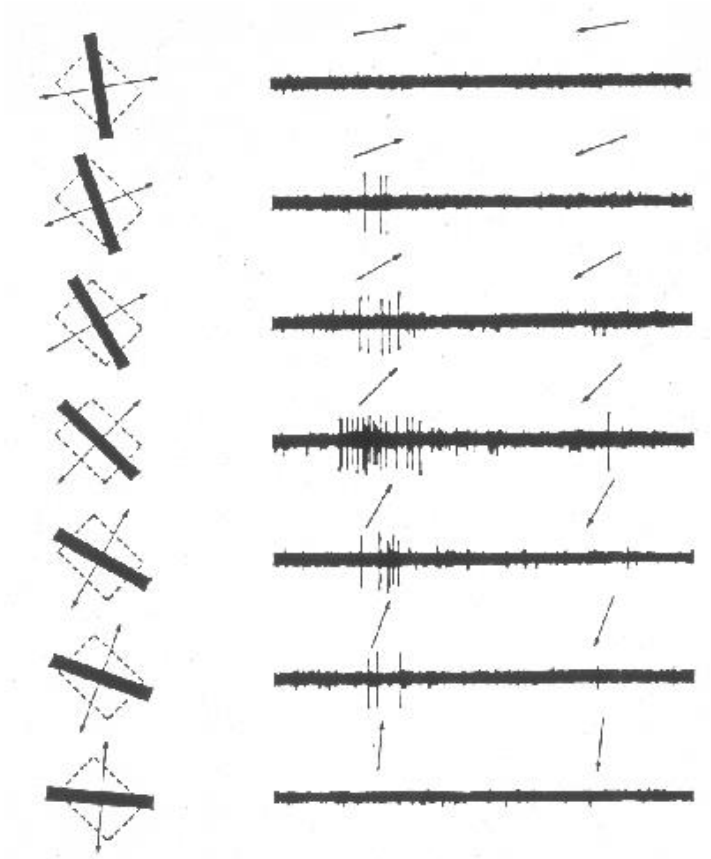
$$E(\mathbf{w}) = \sum_{i=1}^{N} \left(y_i - f_{\mathbf{w}}(\mathbf{x}_i) \right)^2$$

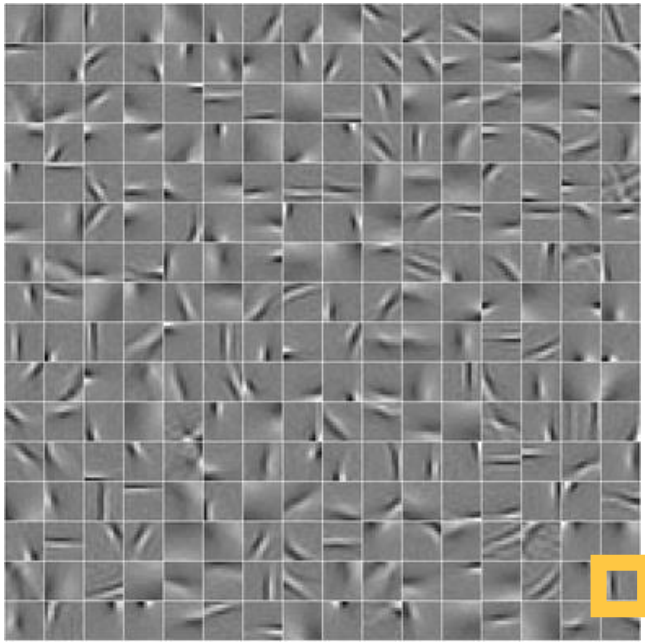
- Minimization can be done by gradient descent provided *f* is differentiable
  - This training method is called **back-propagation**

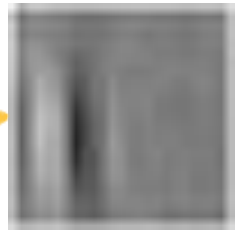
Slide: R. Fergus / S. Lazebnik

Discovery of oriented cells in the visual cortex
[Hubel and Wiesel 59]





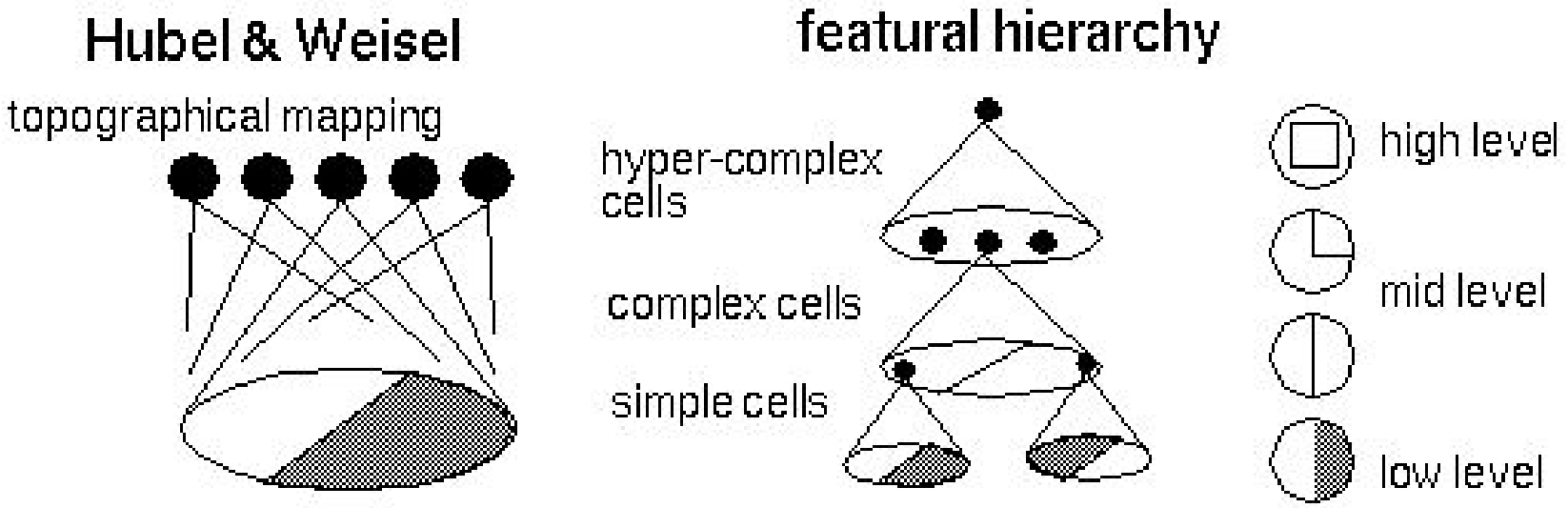




oriented filter

Hubel/Wiesel Architecture

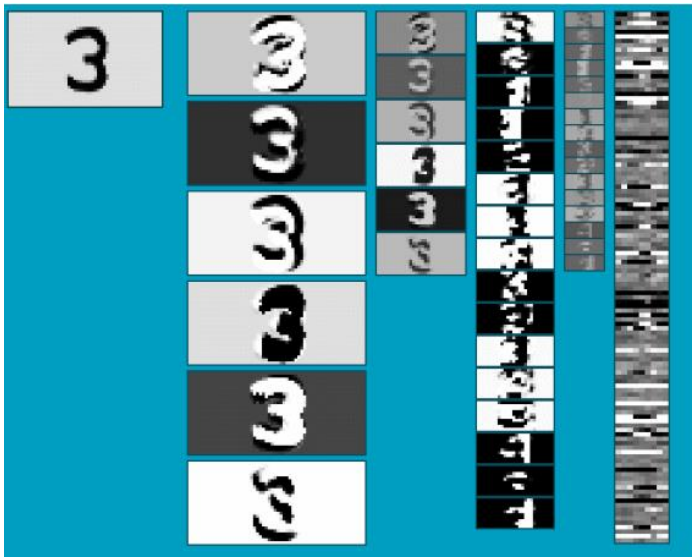
- D. Hubel and T. Wiesel (1959, 1962, Nobel Prize 1981)
  - Visual cortex consists of a hierarchy of *simple*, *complex*, and *hyper-complex* cells

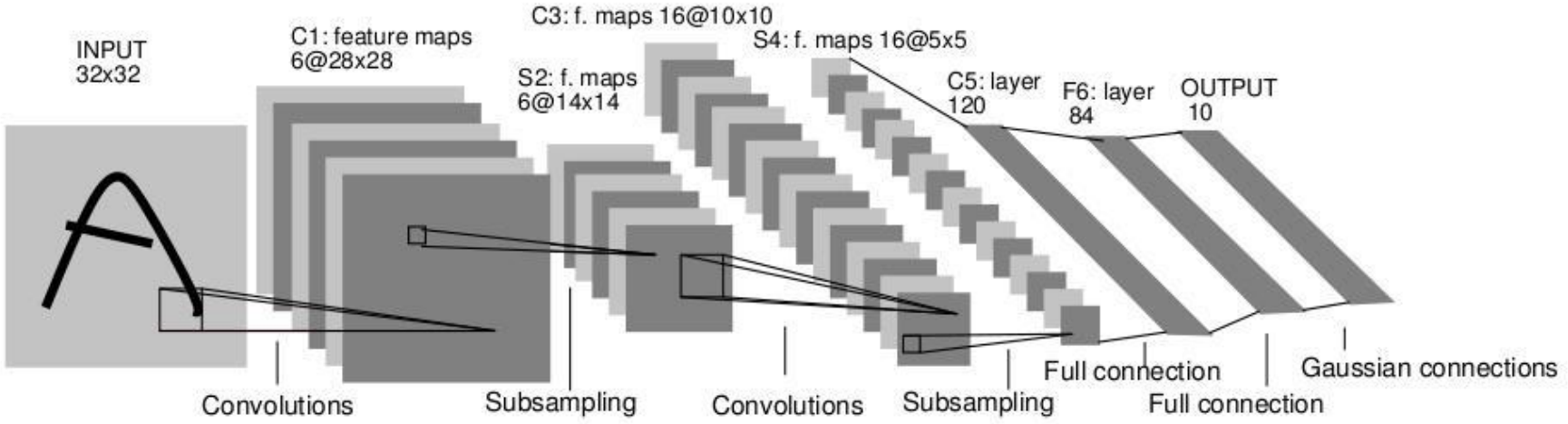


Source

Review: Convolutional Neural Networks (CNN, Convnet)

- Neural network with specialized connectivity structure
- Stack multiple stages of feature extractors
- Higher stages compute more global, more invariant features
- Classification layer at the end



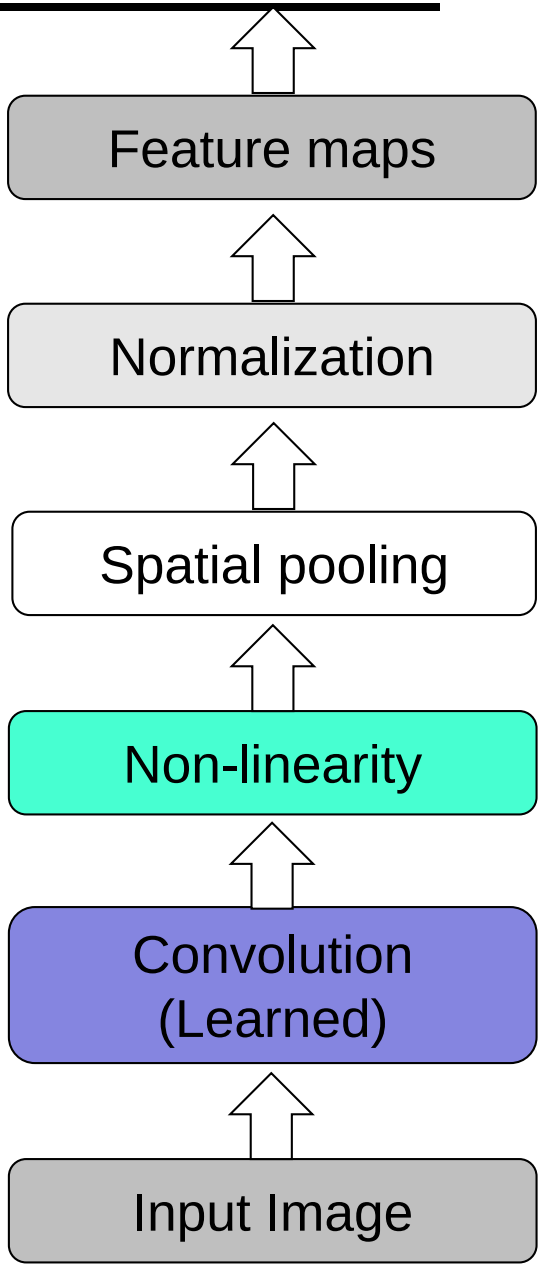


Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86(11): 2278–2324, 1998.

Slide: R. Fergus / S. Lazebnik

Review: Convolutional Neural Networks (CNN, Convnet)

- Feed-forward feature extraction:
  1. Convolve input with learned filters
  2. Non-linearity
  3. Spatial pooling
  4. Normalization
- Supervised training of convolutional filters by back-propagating classification error



Slide: R. Fergus / S. Lazebnik

1. Convolution

- Layer with a special connectivity structure
- Dependencies are local
- Translation invariance

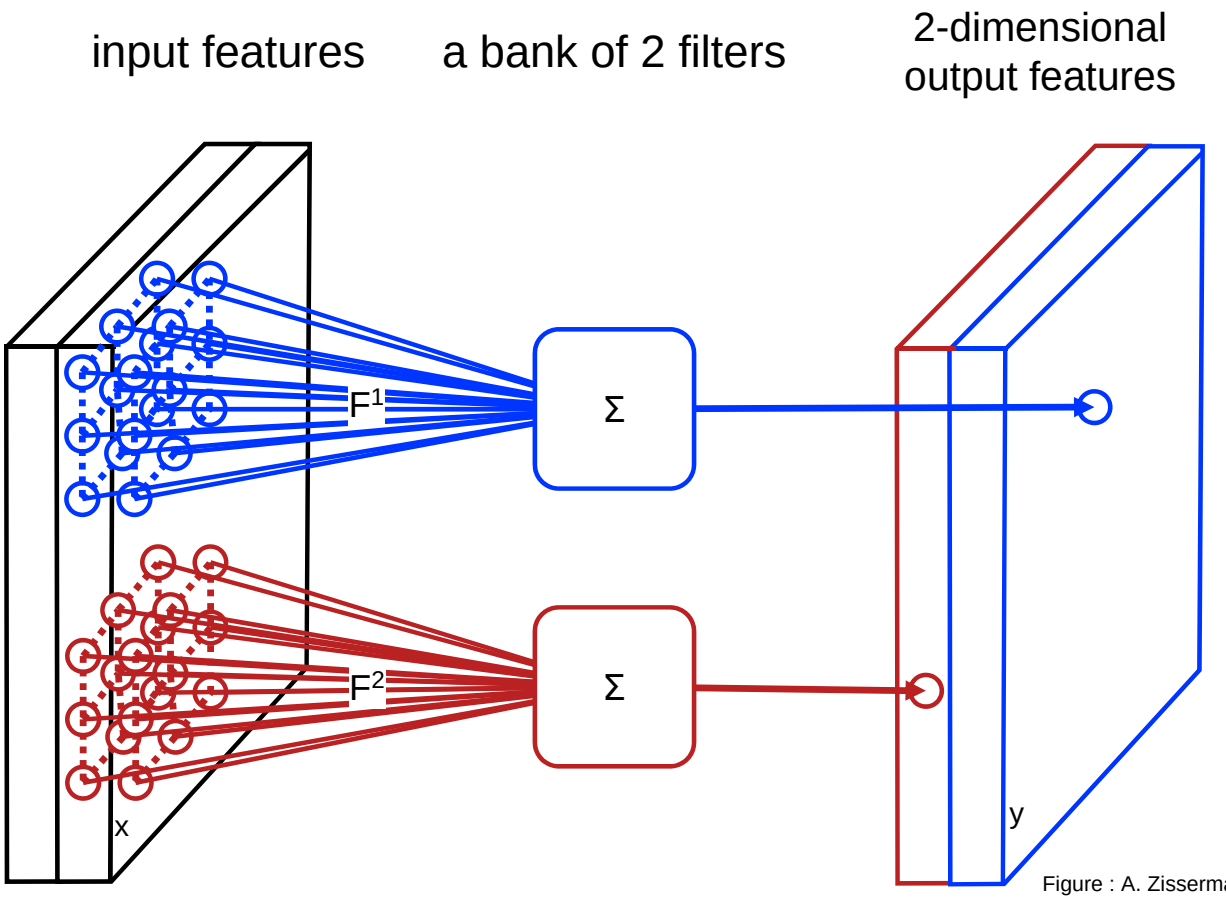
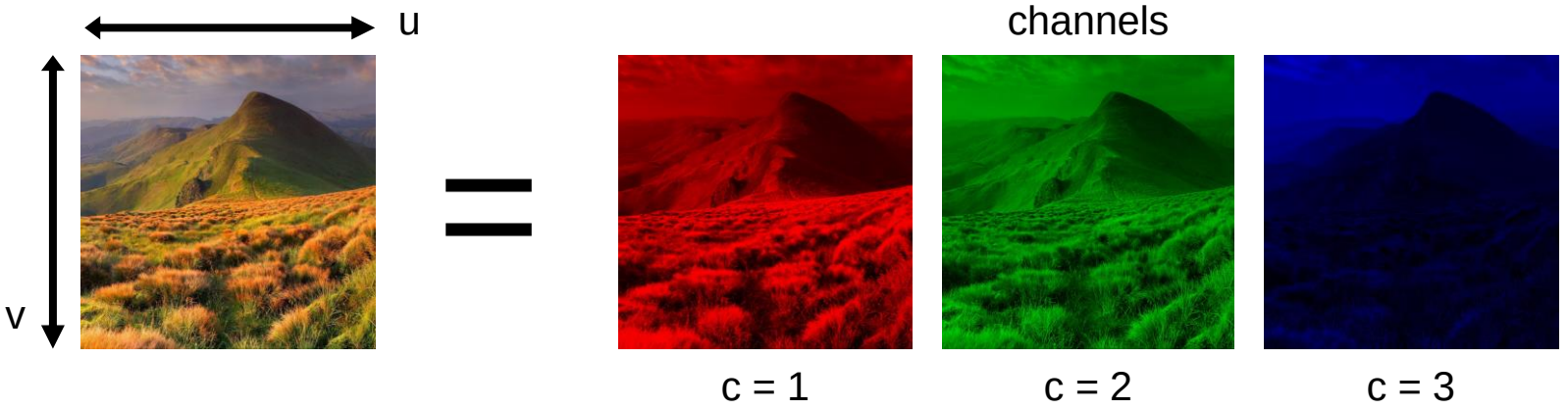


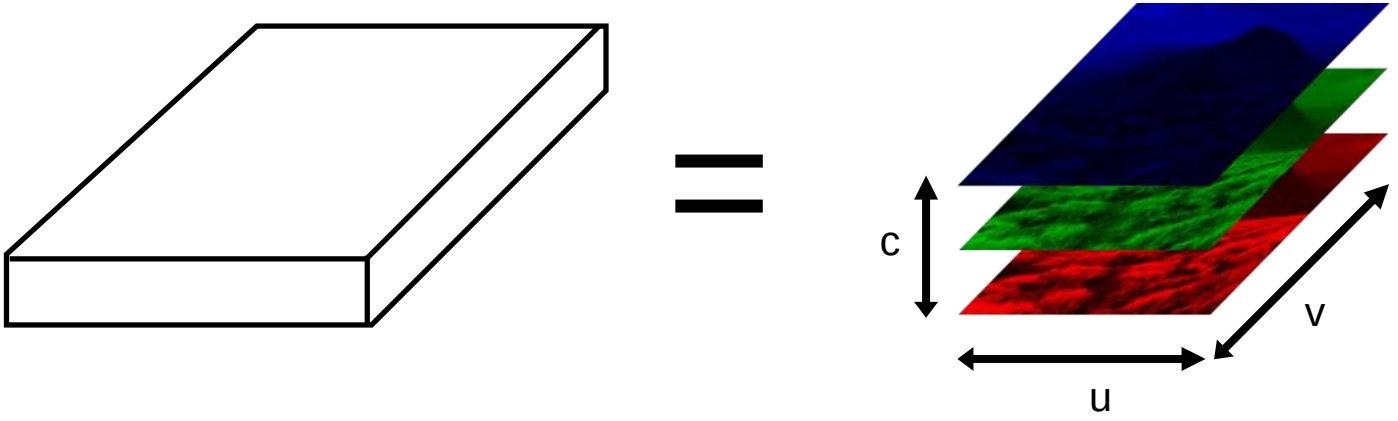
Figure : A. Zisserman

A signal processing notation

The data manipulated by a CNN has the form of 3D tensors. These are interpreted as **discrete vector fields** **x**, assigning a feature vector (x_{uv1}, …, x_{uvC}) at each spatial location (v,u).

A colour image is a simple example of a vector field with 3D features (RGB):

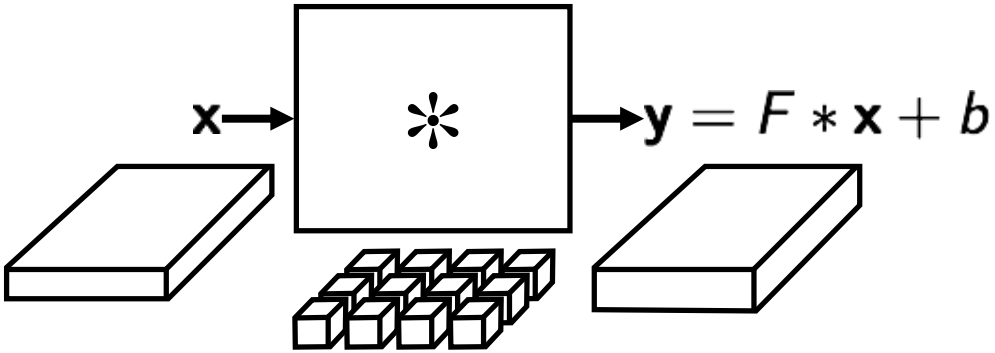




Slide: A. Vedaldi

Linear convolution

With a bank of 3D filters



$$y_{v'u'c'} = b_{c'} + \sum_{\bar{v}=1}^{\bar{H}} \sum_{\bar{u}=1}^{\bar{W}} \sum_{c=1}^{C} x_{v'+\bar{v}-1,u'+\bar{u}-1,c} f_{\bar{v}\bar{u}cc'}$$

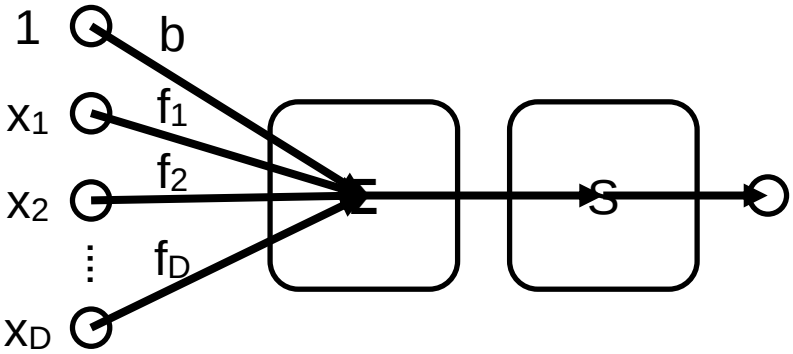
Linear convolution applies a bank of linear filters *F* to the input tensor **x**.

- Input tensor **x** = H × W × K array
- Filter bank *F* = $\bar{H}$ × $\bar{W}$ × K × Q array
- Output tensor **y** = (H - $\bar{H}$ + 1) × (W - $\bar{W}$ + 1) × Q array

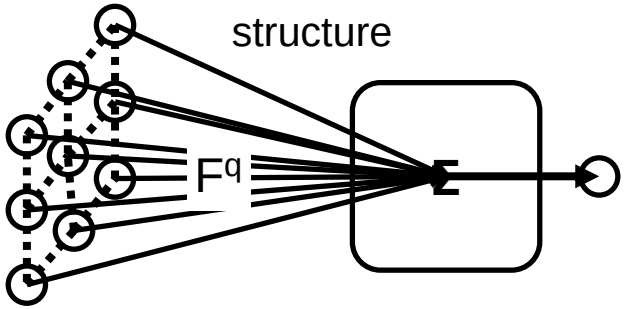
Slide: A. Vedaldi

Linear convolution

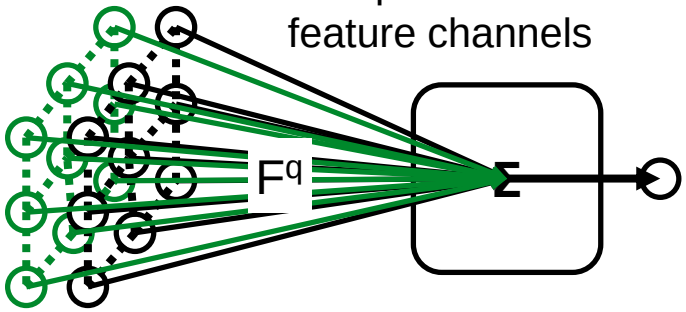
As a neural network



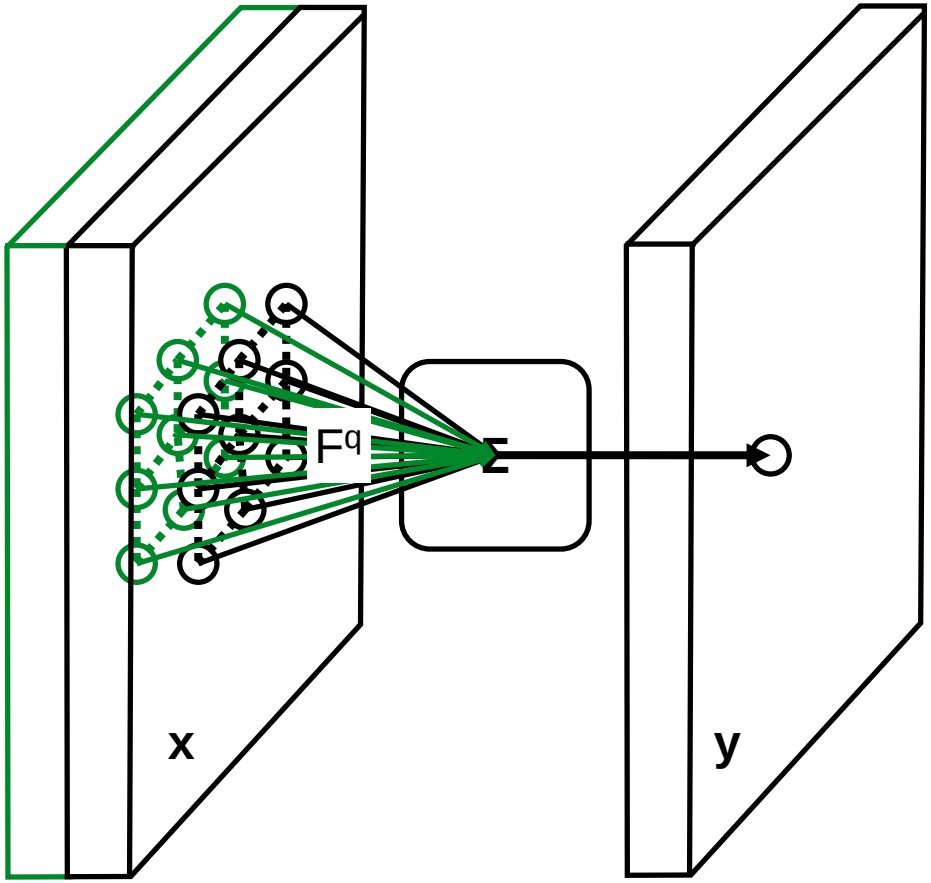
lattice structure



multiple feature channels



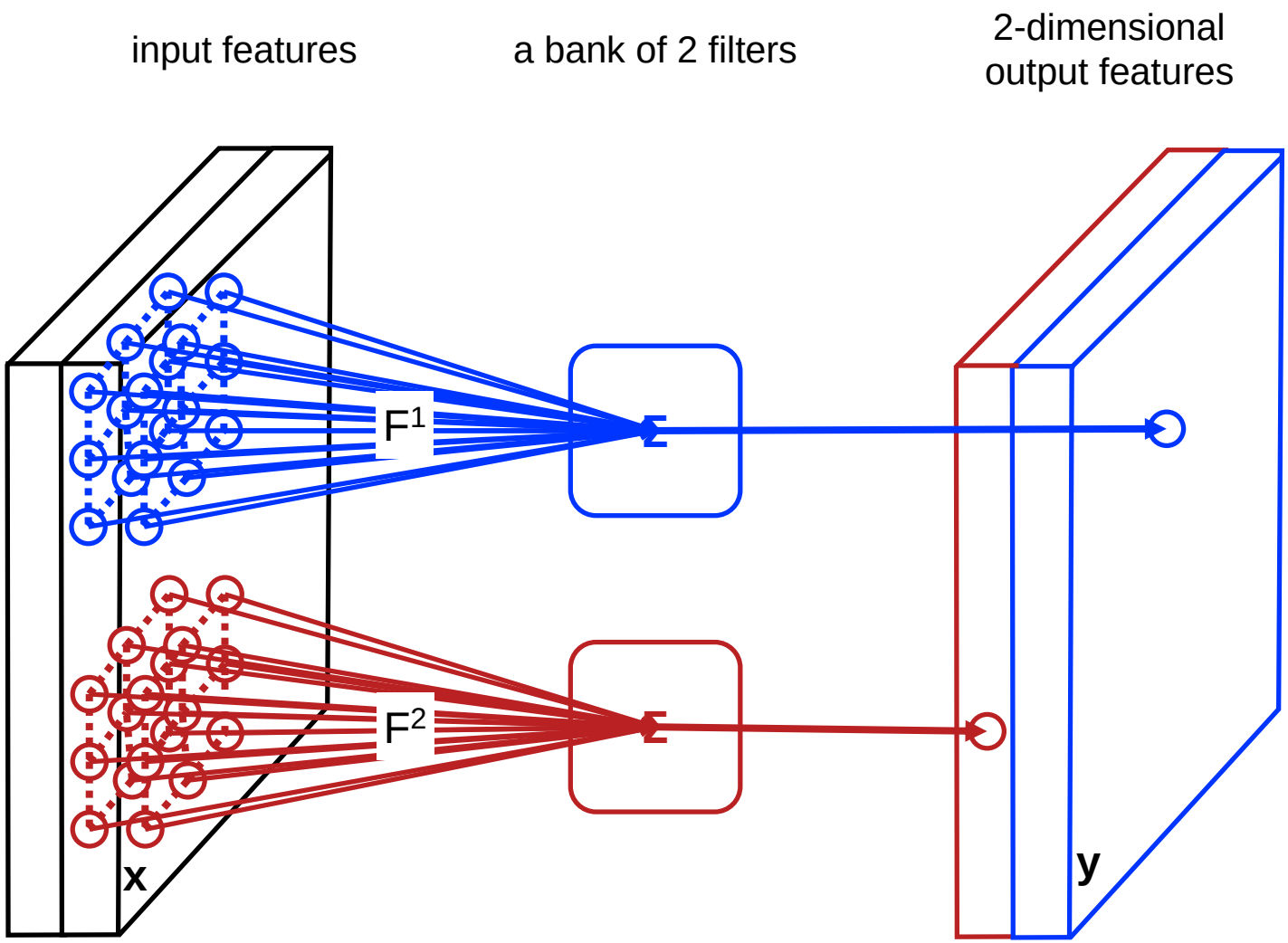
local and translation invariant action



Slide: A. Vedaldi

Linear convolution

As a neural network

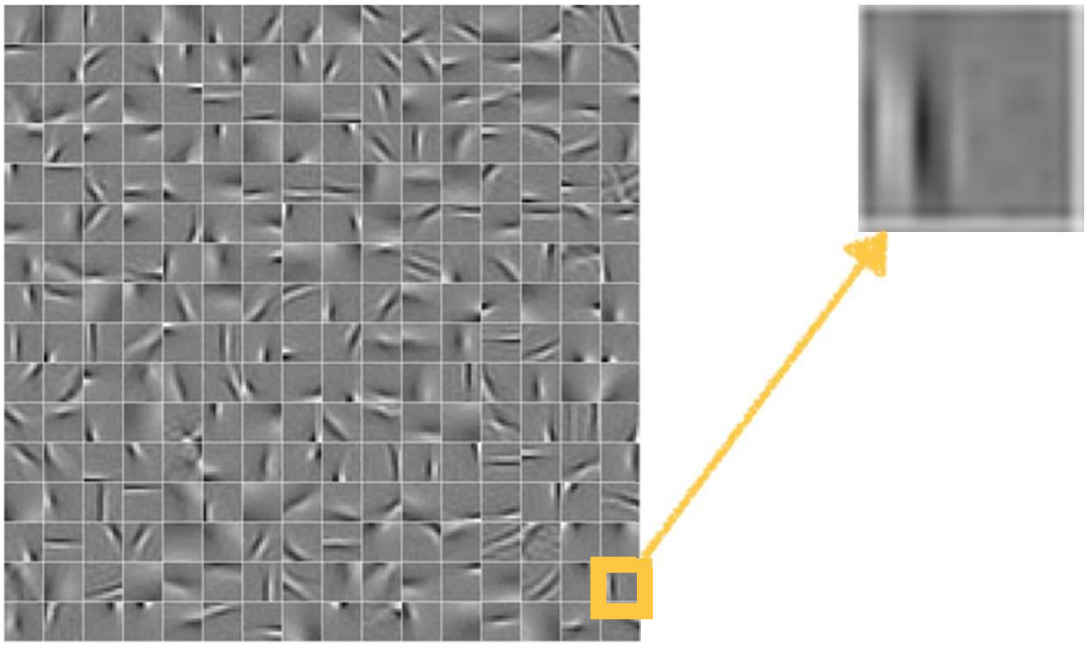


Filter bank example

A bank of 256 filters (learned from data)

Each filter is 1D (it applies to a grayscale image)

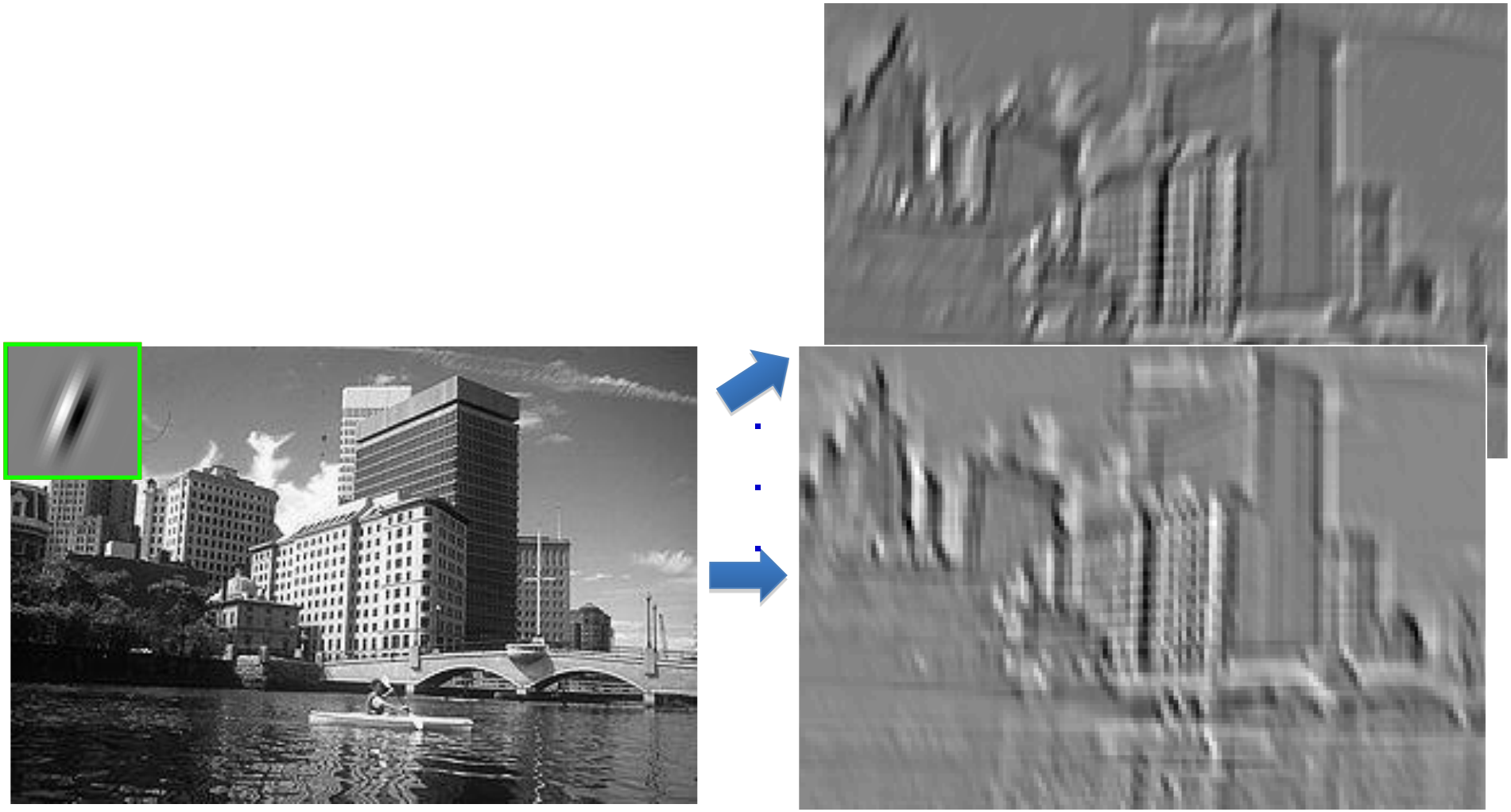
Each filter is 16x16 pixels



Slide: A. Zisserman

Filtering

Each filter generates a feature map

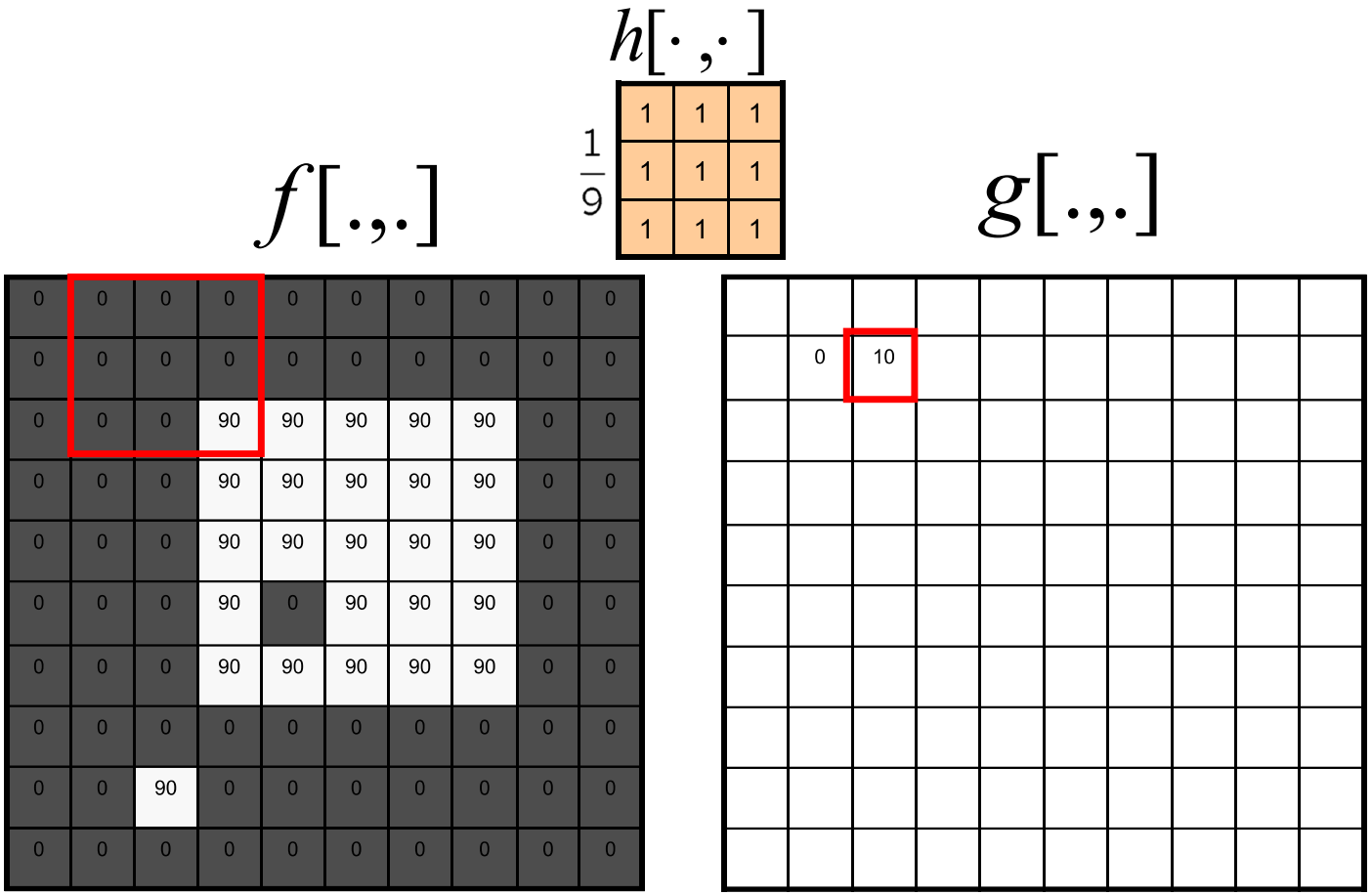


Input

Feature Map

Slide: R. Fergus / S. Lazebnik

How to determine the action of a filter?



$$g[m,n] = \sum_{k,l} h[k,l]\, f[m+k, n+l]$$

Note, similarity to the dot product between h and f

Slide: A. Zisserman

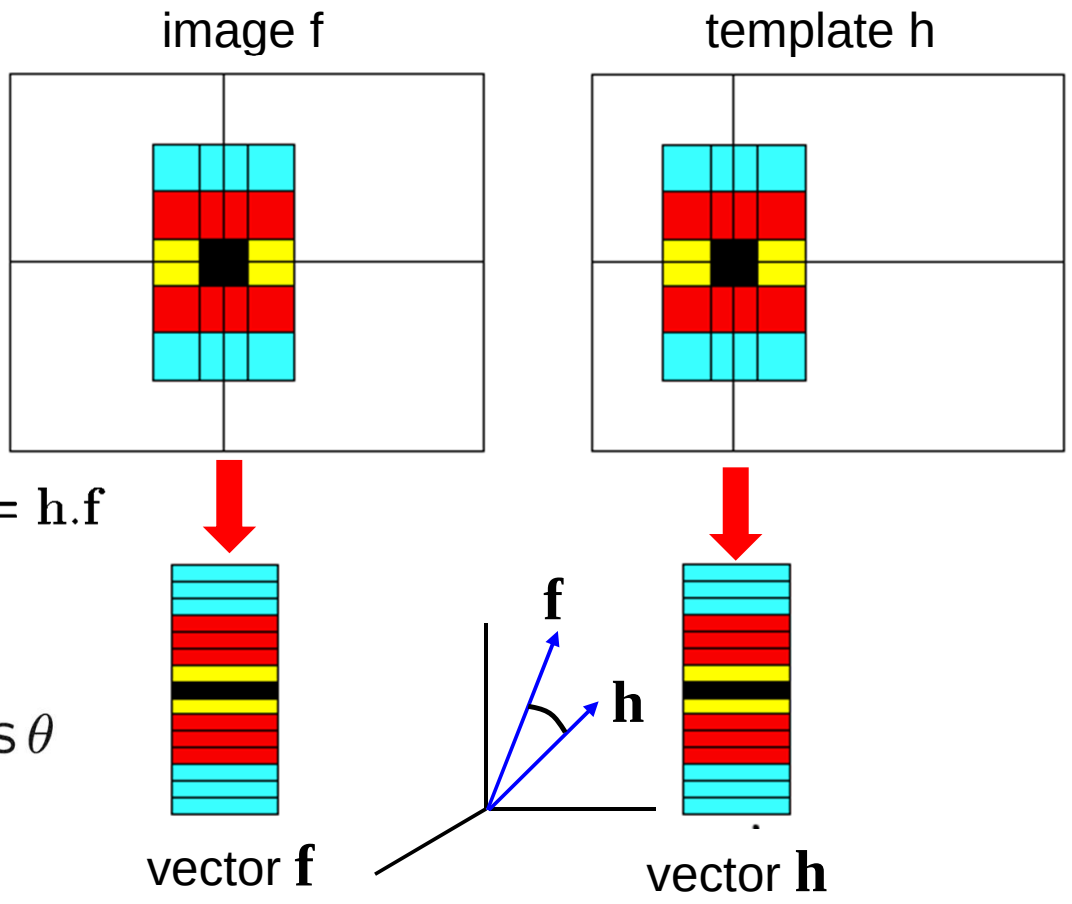
Re-arrange regions as vectors

Write regions as vectors

$$f(x,y) \rightarrow \mathbf{f},$$
$$h(x,y) \rightarrow \mathbf{h}$$

$$g(x,y) = h(x,y) * f(x,y) = \mathbf{h}.\mathbf{f}$$

usually $|\mathbf{h}| = 1$,
so that $\mathbf{g} = \mathbf{h}.\mathbf{f} = |\mathbf{f}| \cos\theta$

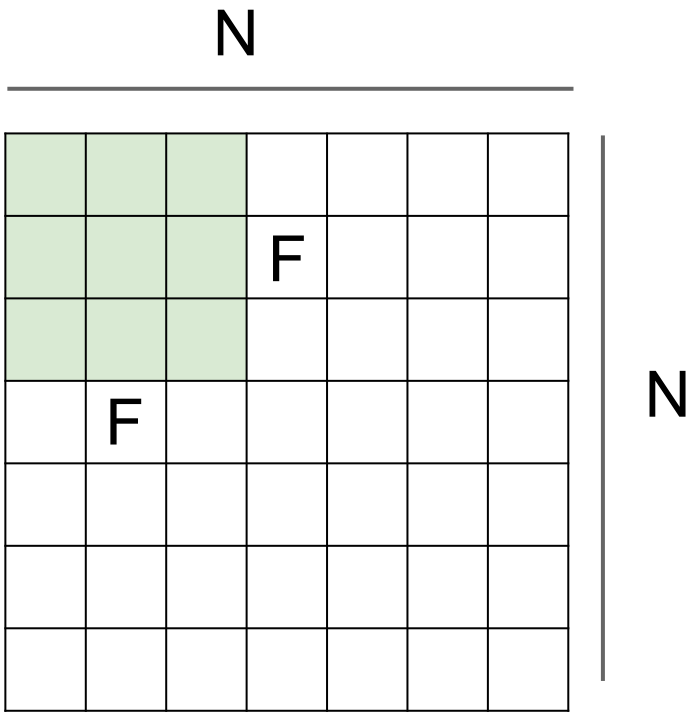


- g is max when f and h are parallel
- maximum response when filter matches signal

Slide: A. Zisserman

Convolution details

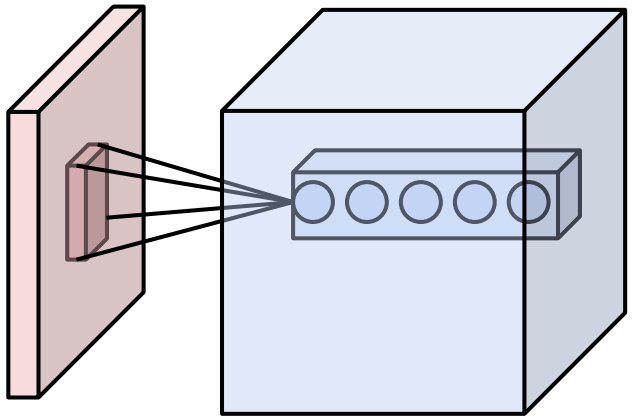
What is the output size?



Output size:
(N - F) / stride + 1

e.g. N = 7, F = 3:
stride 1 => (7 - 3)/1 + 1 = 5
stride 2 => (7 - 3)/2 + 1 = 3
stride 3 => (7 - 3)/3 + 1 = … :\

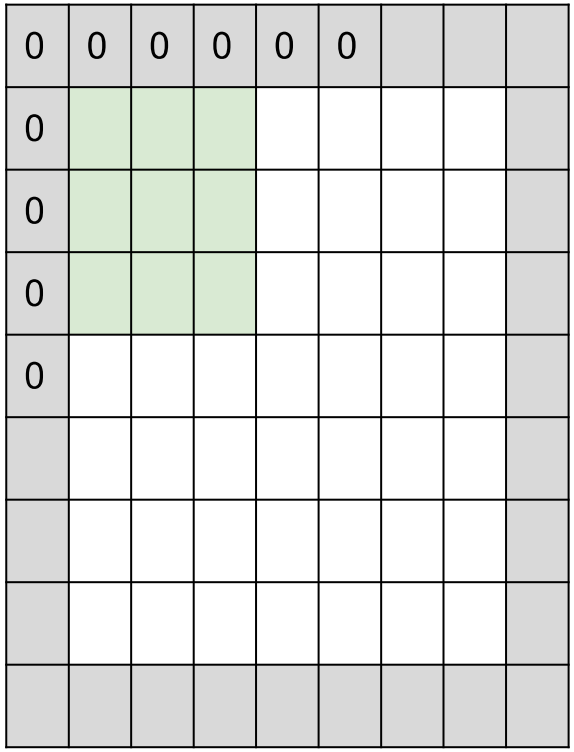
Example: What is the output volume?



Input volume: 32x32x3
Receptive fields: 5x5, stride 3
Number of neurons: 5

Output volume: (32 - 5) / 3 + 1 = 10, so: 10x10x5

Zero padding (in each channel)



e.g. input 7x7
neuron with receptive field 3x3, stride 1
pad with 1 pixel border => what is the output?

7x7 => preserved size!

in general, common to see stride 1, size F, and zero-padding with (F-1)/2.
(Will preserve input size spatially)

What is the number of parameters?

- Consider an input gray-scale image of 1000x1000 pixels.
- What is the number of parameters of a filter bank of 100 7x7 filters?
- How does it compare to a fully connected layer that considers the entire input image?

Convolution:
100x 7x7
= 4900 parameters

vs.

Fully connected layer:
1000x1000
x
1000x1000
= 1T parameters.

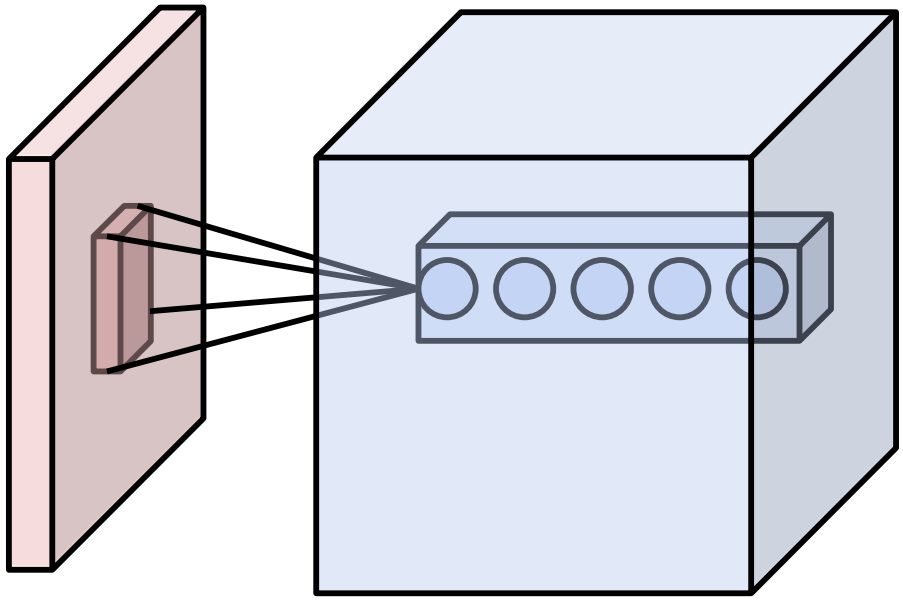
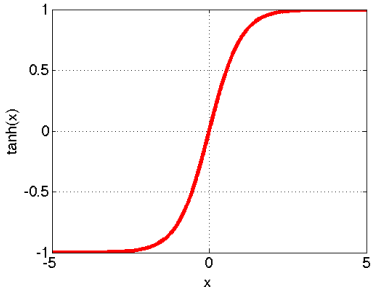
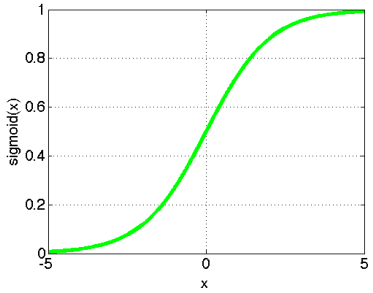


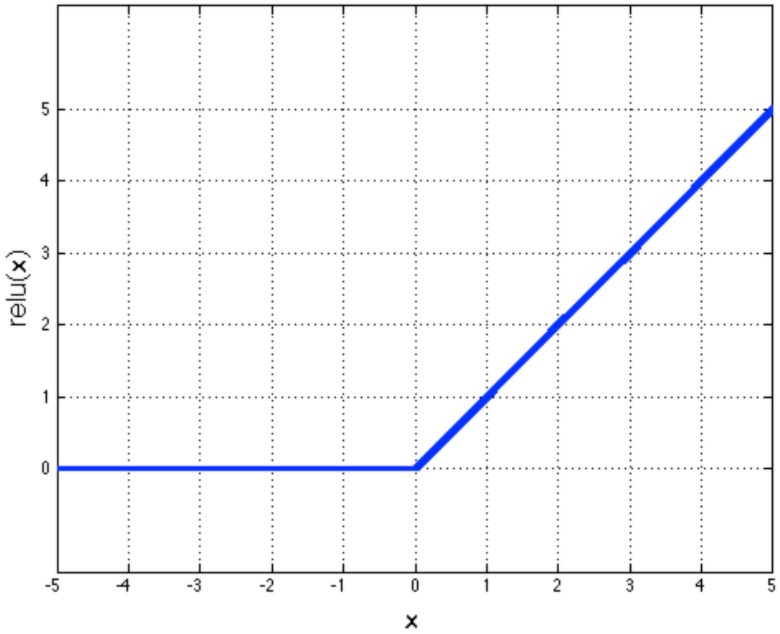
Figure: A. Zisserman

2. Non-Linearity

- Per-element (independent)
- Options:
  - Tanh
  - Sigmoid: 1/(1+exp(-x))
  - Rectified linear unit (ReLU)
    - Simplifies backpropagation
    - Makes learning faster
    - Avoids saturation issues
      → Preferred option



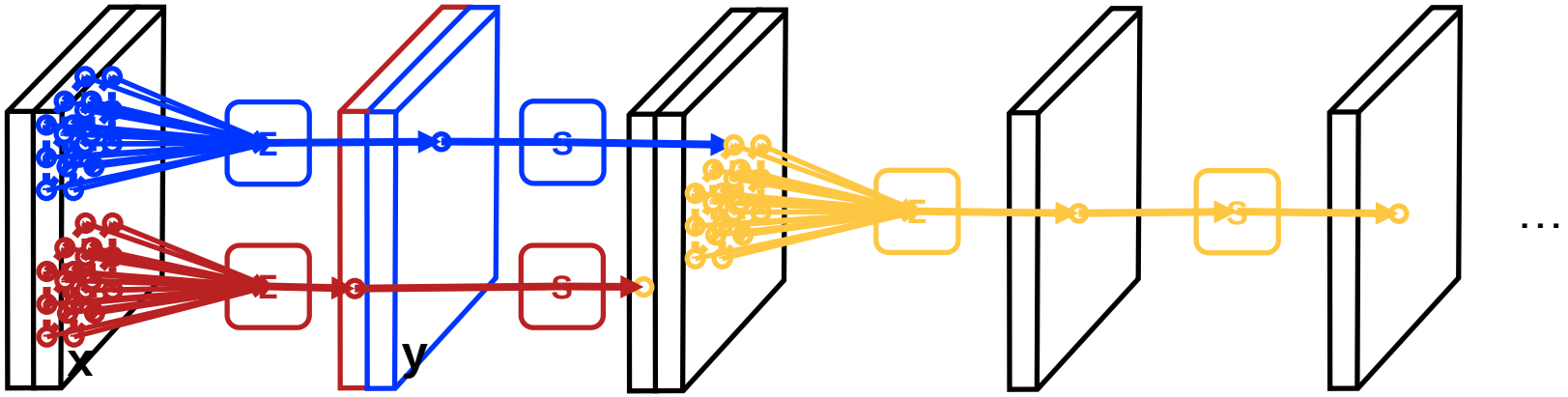




Slide: R. Fergus / S. Lazebnik

Multiple layers

Convolution, gating, convolution, …



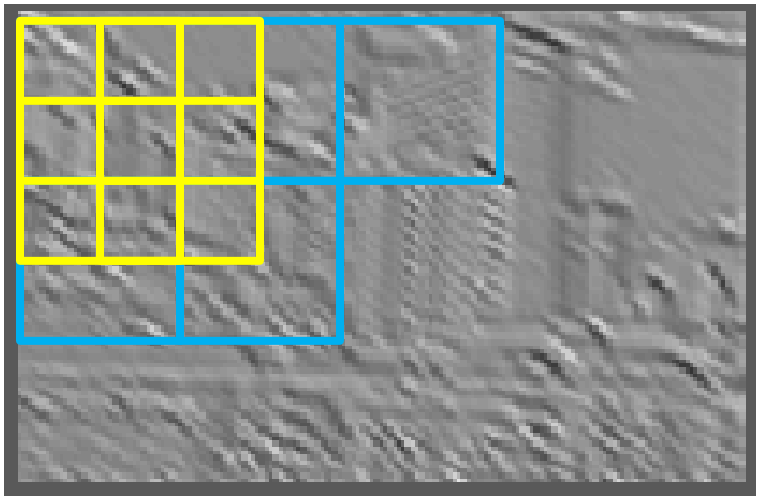
A deep convolutional neural networks chains several filtering + non-linear activation function sequences.

The non-linear activation functions are essential. Why?

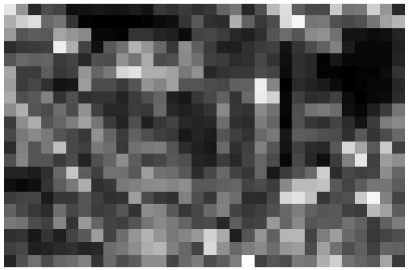
Slide: A. Vedaldi

3. Spatial Pooling

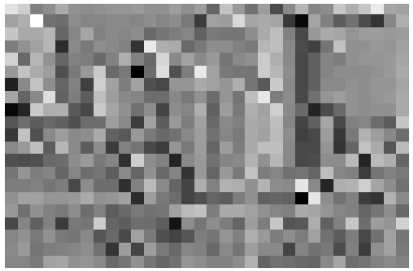
- Sum or max
- Non-overlapping / overlapping regions
- Role of pooling:
  - Invariance to small transformations
  - Larger receptive fields (see more of input)



Max

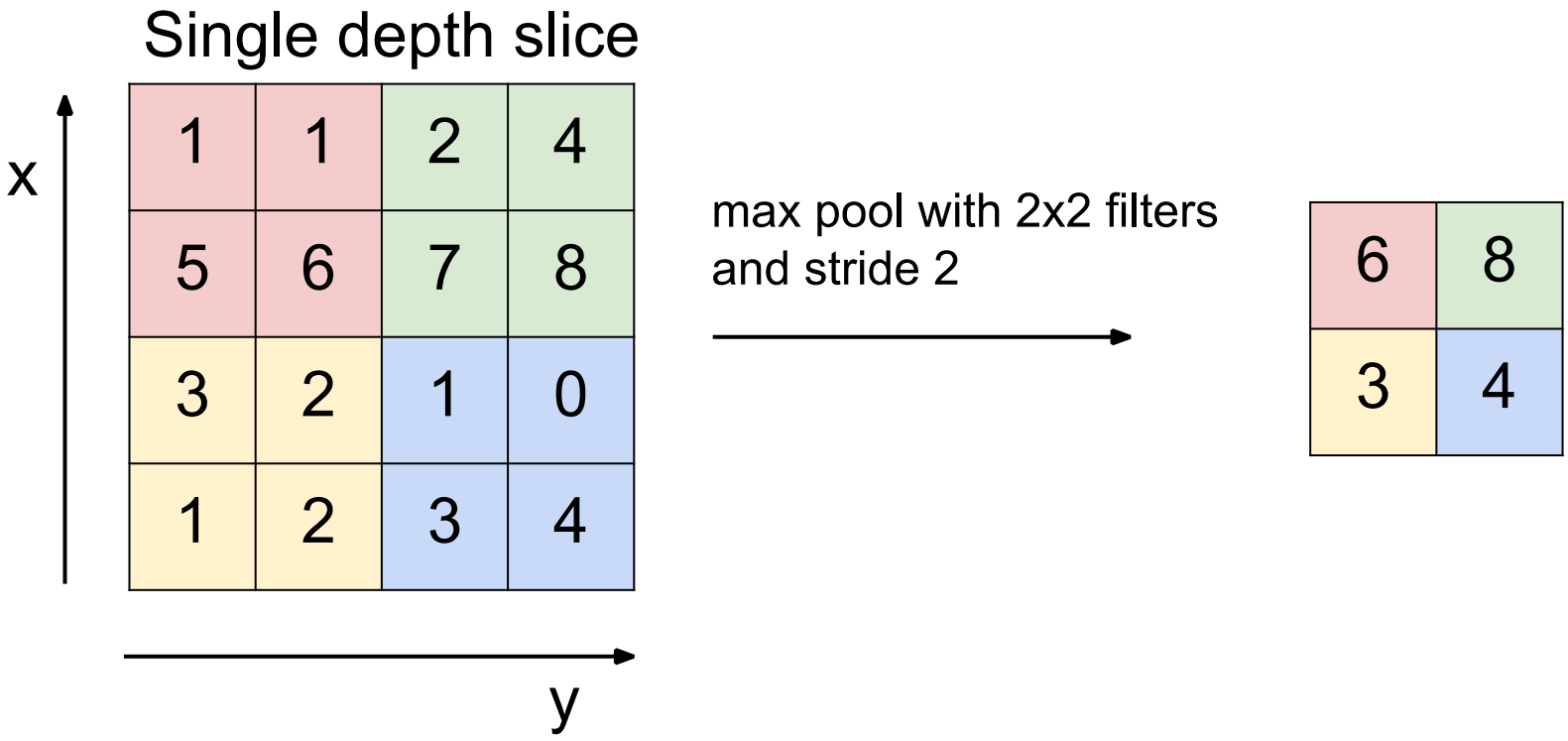


Sum



Slide: R. Fergus / S. Lazebnik

Spatial pooling example



Dimensions of pooling outputs

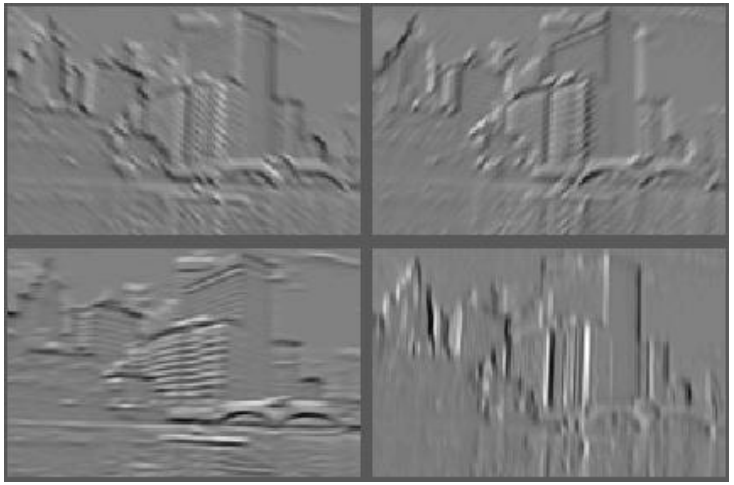
Input volume of size [W1 x H1 x D1]
Pooling unit receptive fields F x F and applying them at strides of S gives

Output volume: [W2, H2, D1]
W2 = (W1-F)/S+1, H2 = (H1-F)/S+1

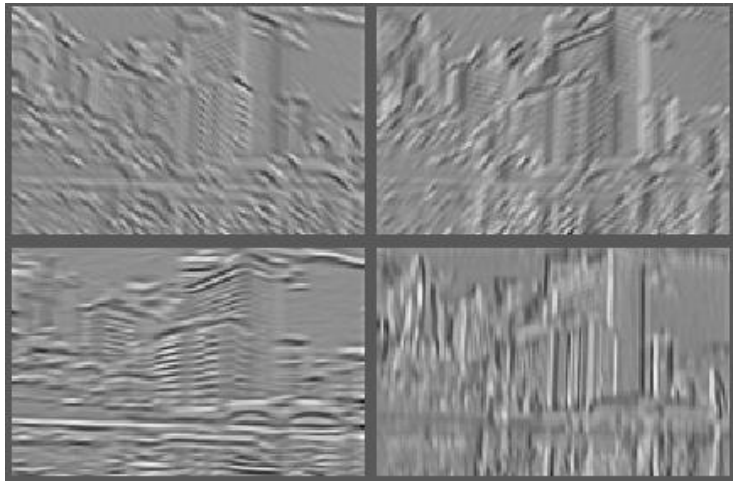
Note: pooling happens independently in each slice

4. Normalization

- Within or across feature maps
- Before or after spatial pooling



Feature Maps

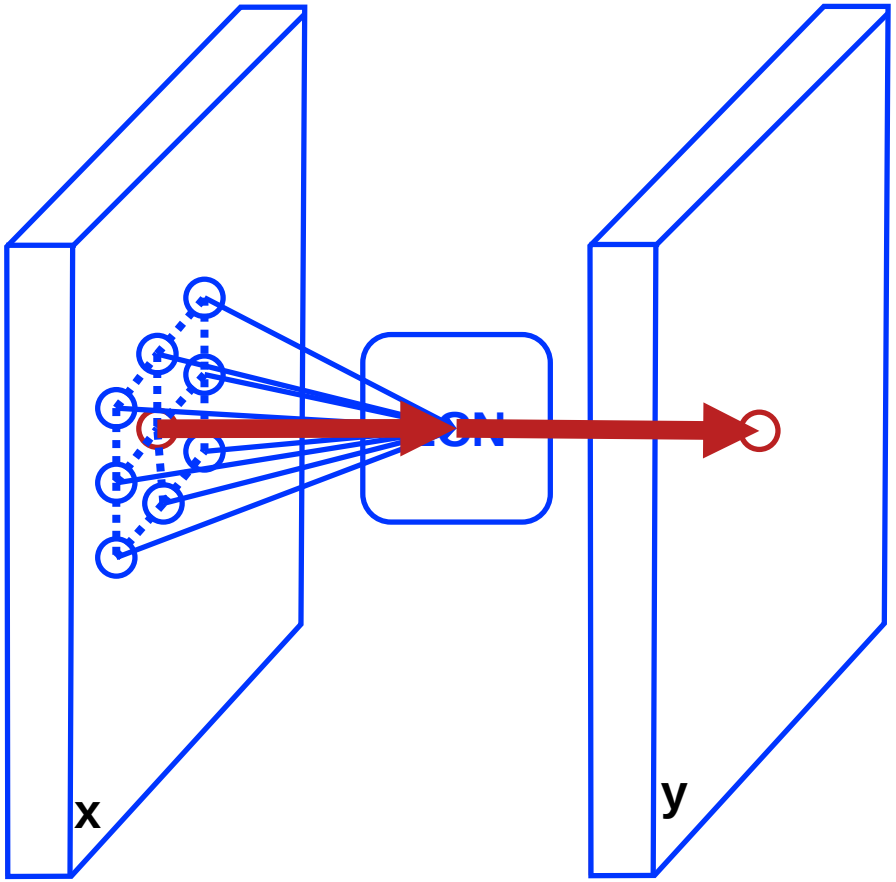


Feature Maps
After Contrast Normalization

Slide: R. Fergus / S. Lazebnik

Local contrast normalisation

Normalise image/feature patches



$$y_{ijq} = \frac{x_{ijq} - \mu_{ijq}}{\sigma_{ijq}}$$

$$\mu_{ijq} = \frac{1}{|\mathcal{N}(i,j)|} \sum_{(u,v)\in\mathcal{N}(i,j)} y_{uvq}$$

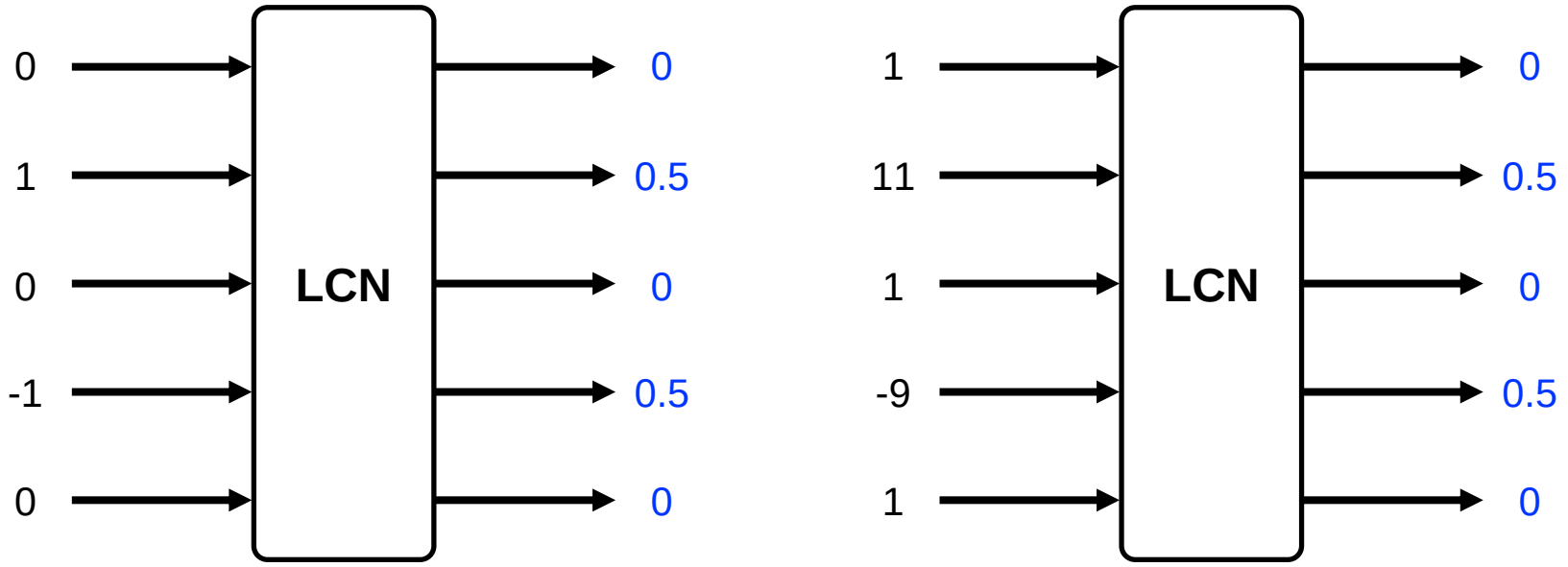
$$\sigma^2_{ijq} = \frac{1}{|\mathcal{N}(i,j)|} \sum_{(u,v)\in\mathcal{N}(i,j)} (y_{uvq} - \mu_{ijq})^2$$

Slide: A. Vedaldi

Local contrast normalisation

Example

It has a local equalising effect:



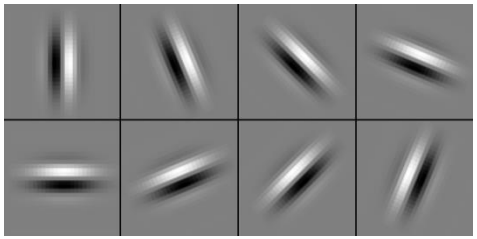
Slide: A. Vedaldi

Compare: SIFT Descriptor

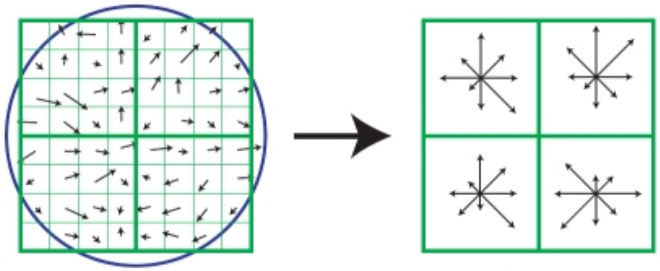
Lowe
[IJCV 2004]

Image Pixels

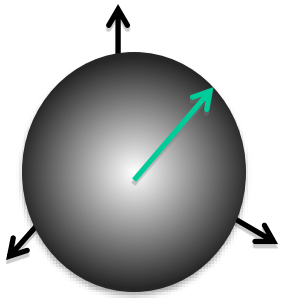
Apply oriented filters



Spatial pool (Sum)



Normalize to unit length



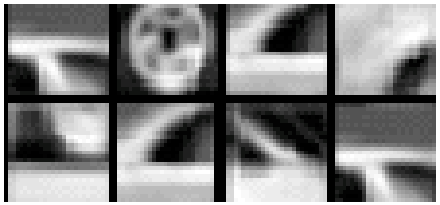
Feature Vector

Slide: R. Fergus / S. Lazebnik

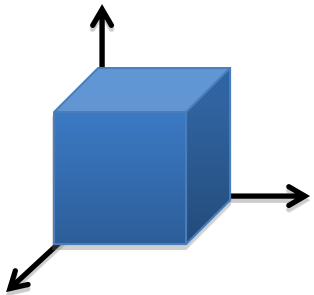
Compare: Spatial Pyramid Matching

SIFT features

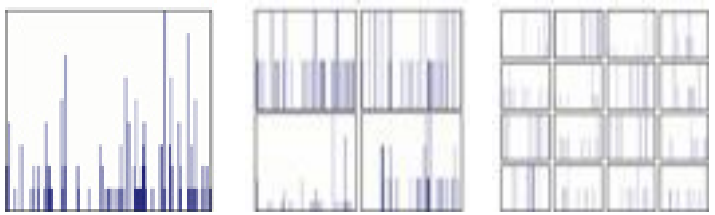
Filter with Visual Words



Take max VW response (L-inf normalization)



Multi-scale spatial pool (Sum)



Global image descriptor

Lazebnik, Schmid, Ponce [CVPR 2006]

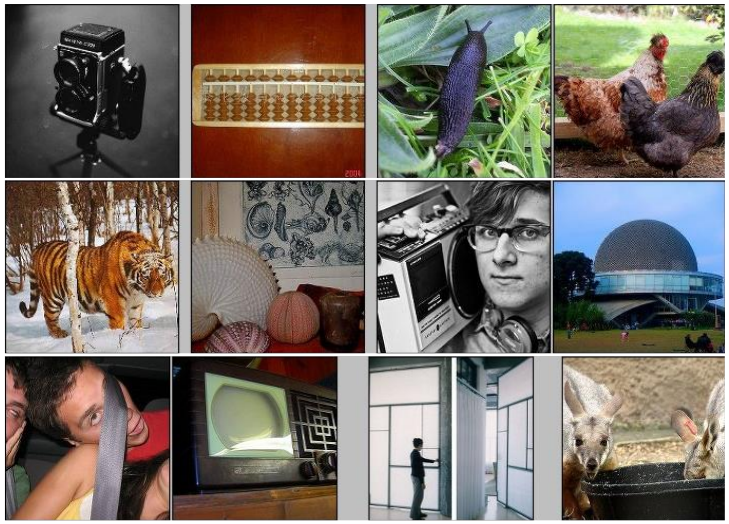
Slide: R. Fergus / S. Lazebnik

Convnet Successes

- Handwritten text/digits
  - MNIST (0.17% error [Ciresan et al. 2011])
  - Arabic & Chinese [Ciresan et al. 2012]
- Simpler recognition benchmarks
  - CIFAR-10 (9.3% error [Wan et al. 2013])
  - Traffic sign recognition
    - 0.56% error vs 1.16% for humans [Ciresan et al. 2011]
- But until recently, less good at more complex datasets
  - Caltech-101/256 (few training examples)

Slide: R. Fergus / S. Lazebnik

ImageNet Challenge 2012



[Deng et al. CVPR 2009]

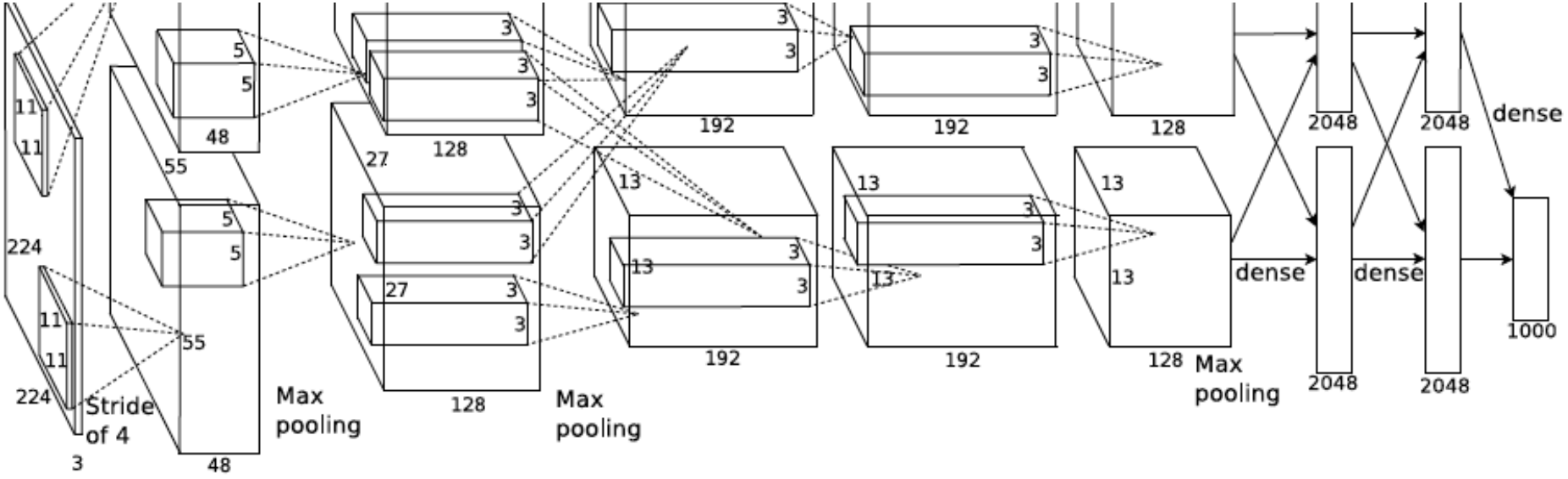
- ~14 million labeled images, 20k classes
- Images gathered from Internet
- Human labels via Amazon Turk
- Challenge: 1.2 million training images, 1000 classes

A. Krizhevsky, I. Sutskever, and G. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, NIPS 2012

Slide: R. Fergus / S. Lazebnik

ImageNet Challenge 2012

- Similar framework to LeCun'98 but:
  - Bigger model (7 hidden layers, 650,000 units, 60,000,000 params)
  - More data (10^6 vs. 10^3 images)
  - GPU implementation (50x speedup over CPU)
    - Trained on two GPUs for a week
  - Better regularization for training (DropOut)



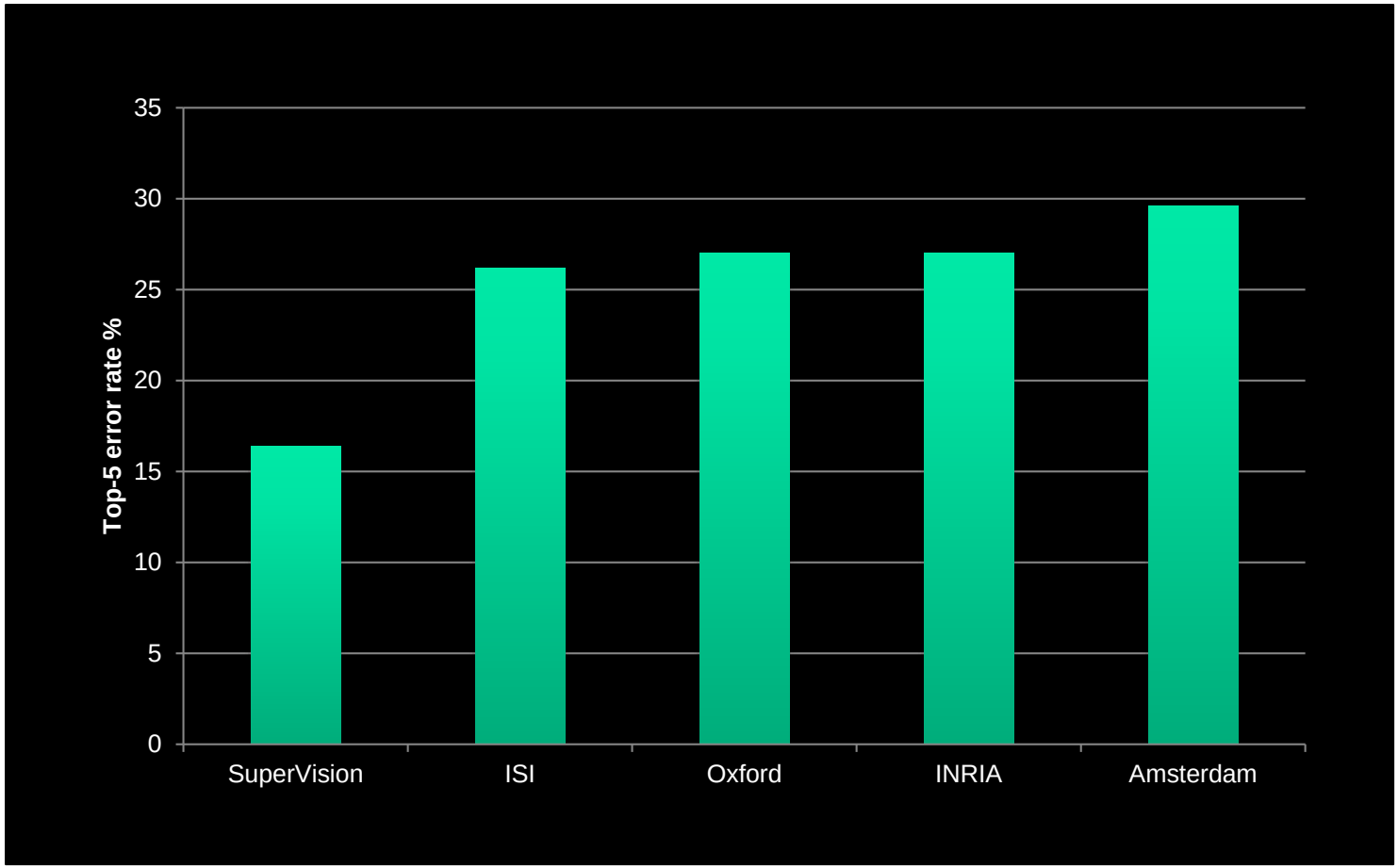
A. Krizhevsky, I. Sutskever, and G. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, NIPS 2012

Slide: R. Fergus / S. Lazebnik

ImageNet Challenge 2012

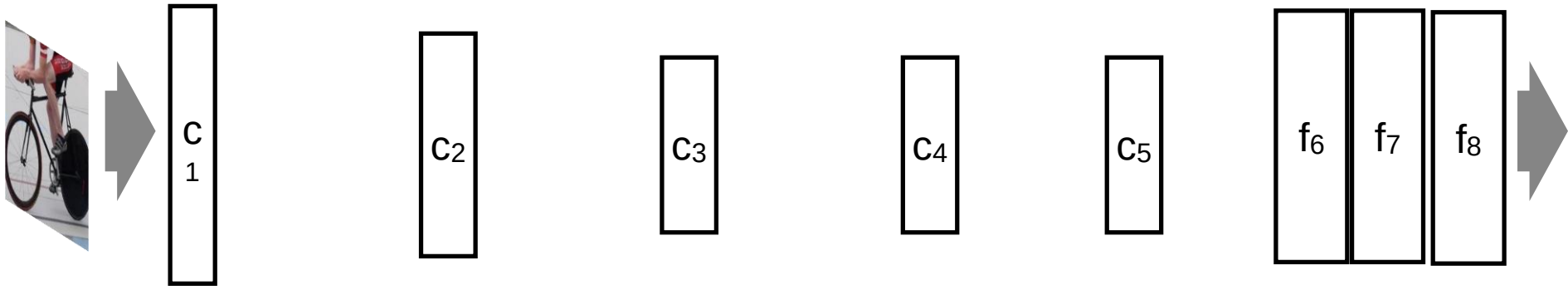
Krizhevsky et al. -- **16.4% error** (top-5)

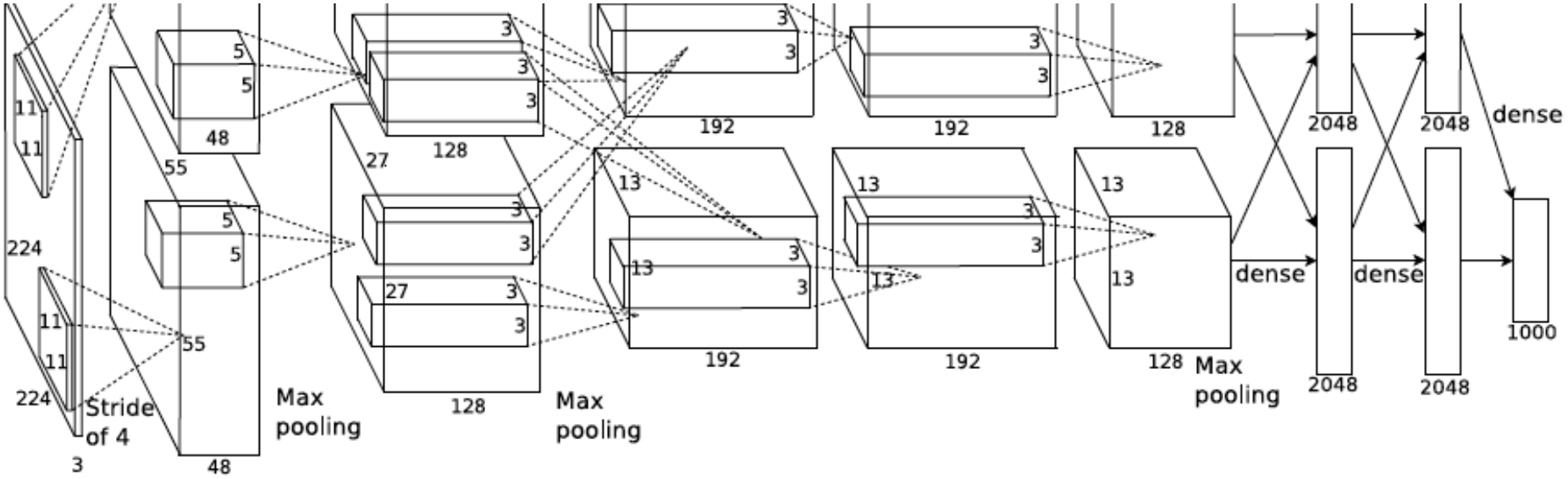
Next best (non-convnet) **– 26.2% error**



Slide: R. Fergus / S. Lazebnik

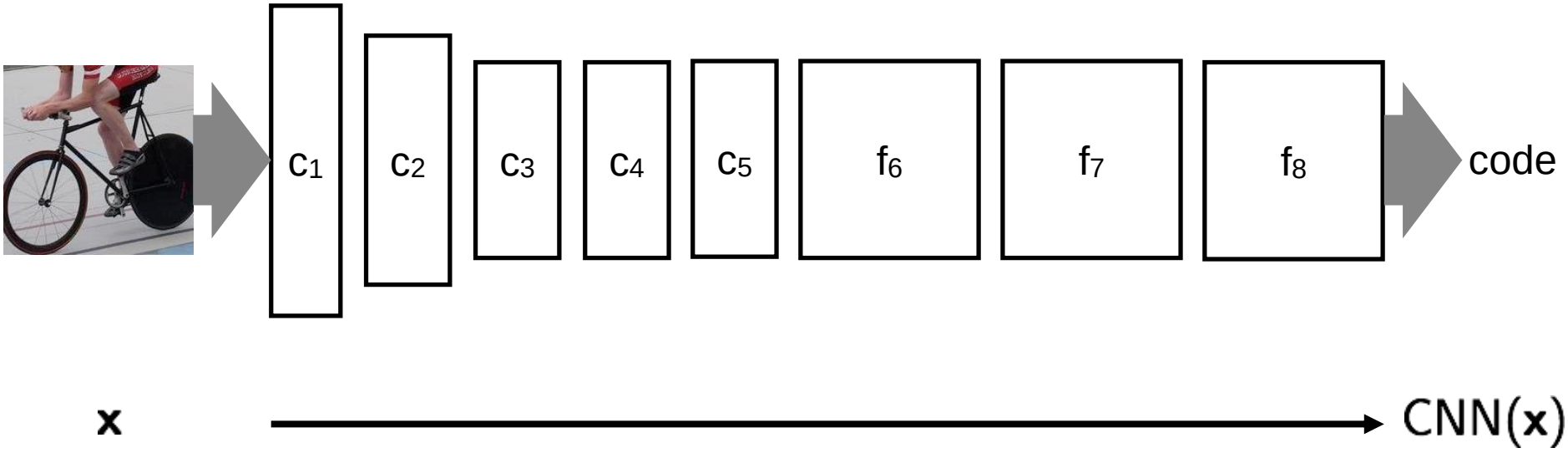
Typical CNN architecture: AlexNet





Figures: A. Vedaldi

A typical CNN design



From left to right

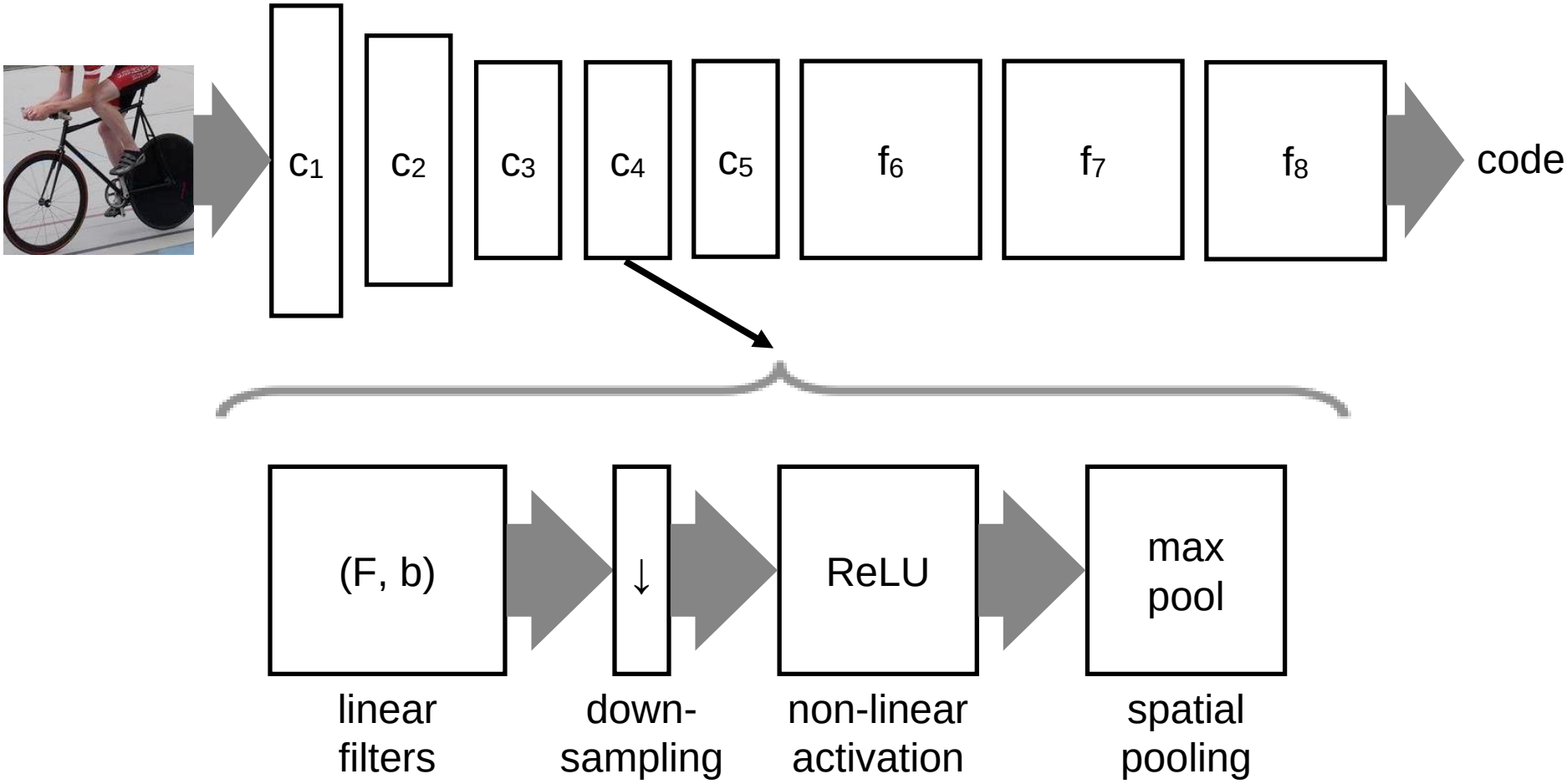
- decreasing spatial resolution
- increasing feature dimensionality

“Fully-connected” layers

- same as convolutional, but with 1×1 spatial resolution
- contain most of the parameters

Slide: A. Vedaldi

Convolutional layers

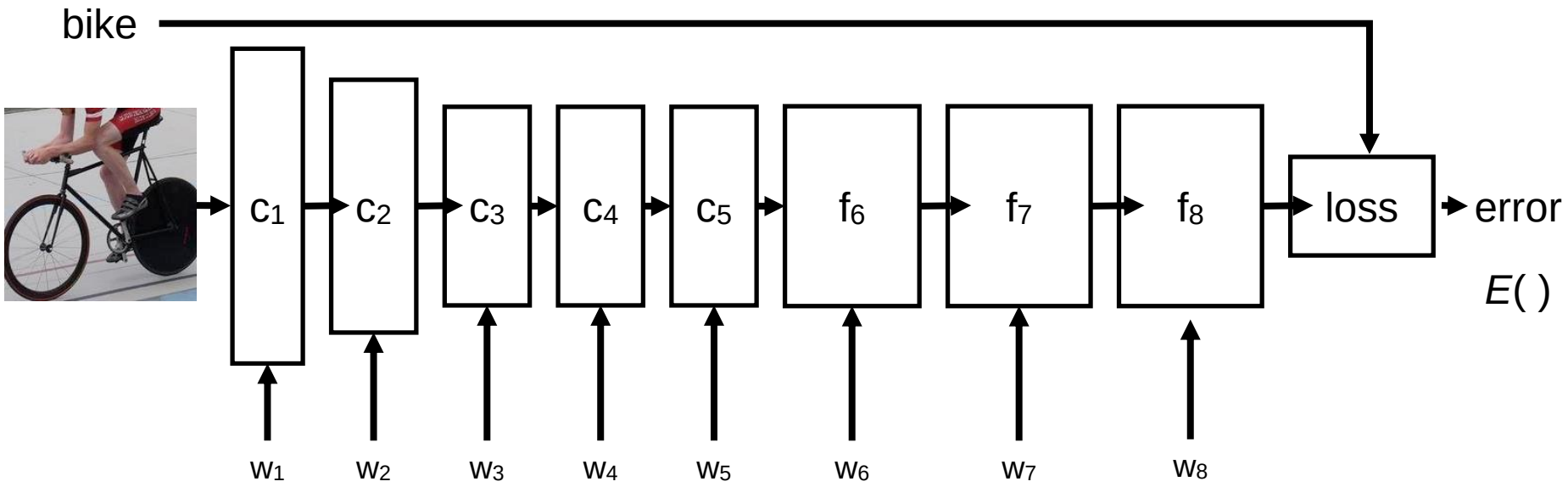


Each block c_1, c_2, …, f_8 is in turn a composition of convolution, downsampling, ReLU and max pooling.

Downsampling and max pooling are optional.

Slide: A. Vedaldi

Learning a CNN



$$\text{argmin } E(\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_8)$$

Stochastic gradient descent
(with momentum, dropout, …)

Slide: A. Vedaldi

Stochastic gradient descent

The objective function is an average over many data points:

$$E(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N} E_i(\mathbf{w})$$

Key idea: approximate the gradient sampling a point at a time:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \nabla E_i(\mathbf{w}_t), \qquad i \sim U(\{1, 2, \ldots, N\})$$

uniform distribution

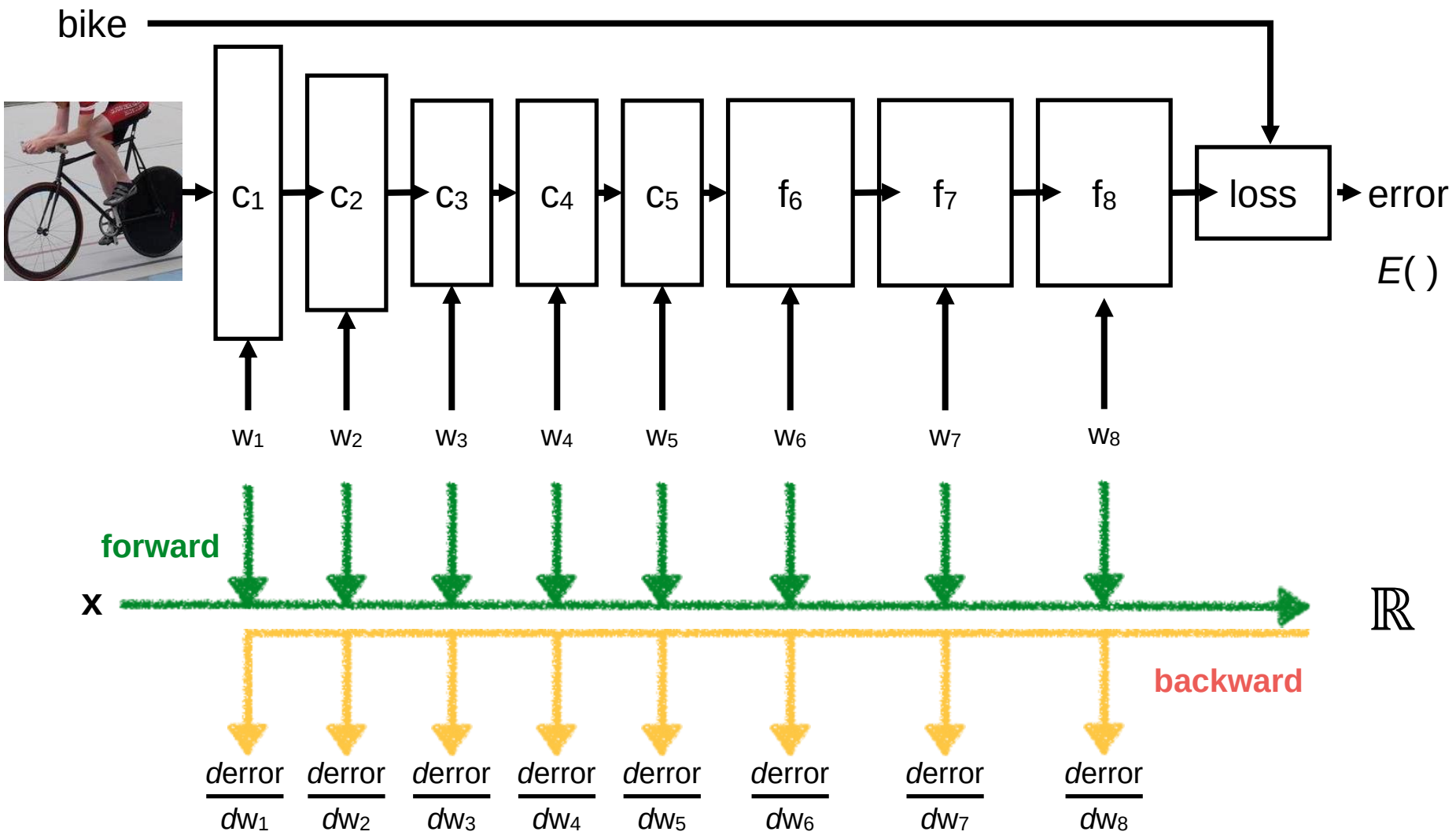
Details:

- **Epochs**: all points are visited sequentially, but in random order
- **Validation**: evaluate $E(\mathbf{w}_t)$ on an held-out validation set to diagnose objective decrease
- **Learning rate** η_t: is decreased tenfold once the objective $E(\mathbf{w}_t)$ stops decreasing.
- **Momentum**: the gradient estimate is smoothed by using a moving average:

$$\mathbf{m}_{t+1} = 0.9\,\mathbf{m}_t + \eta_t \nabla E_i(\mathbf{w}_t), \qquad \mathbf{w}_{t+1} = \mathbf{w}_t - \mathbf{m}_{t+1}$$

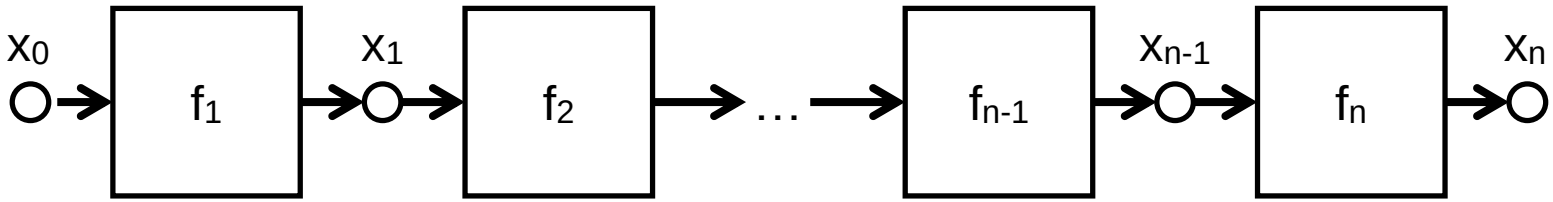
Backpropagation

Computing derivatives using the chain rule



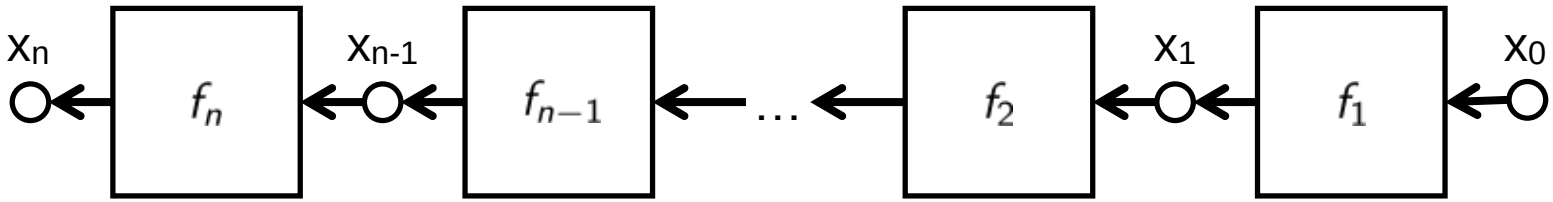
Slide: A. Vedaldi

Chain rule: scalar version



Slide: A. Vedaldi

Chain rule: scalar version



A composition of *n* functions

$$\mathbf{x}_n = (\ f_n \circ f_{n-1} \circ \dots \circ f_2 \circ f_1\ )\ (\mathbf{x}_0)$$

$$\frac{dx_n}{dx_0} = \frac{df_n}{dx_{n-1}} \times \frac{df_{n-1}}{dx_{n-2}} \times \dots \times \frac{df_2}{dx_1} \times \frac{df_1}{dx_0}$$

Derivative obtained using the chain rule

Outline

1. Convolutional neural networks (CNNs)

2. Understanding and visualizing CNN representations

3. Transferring learnt representation to other tasks

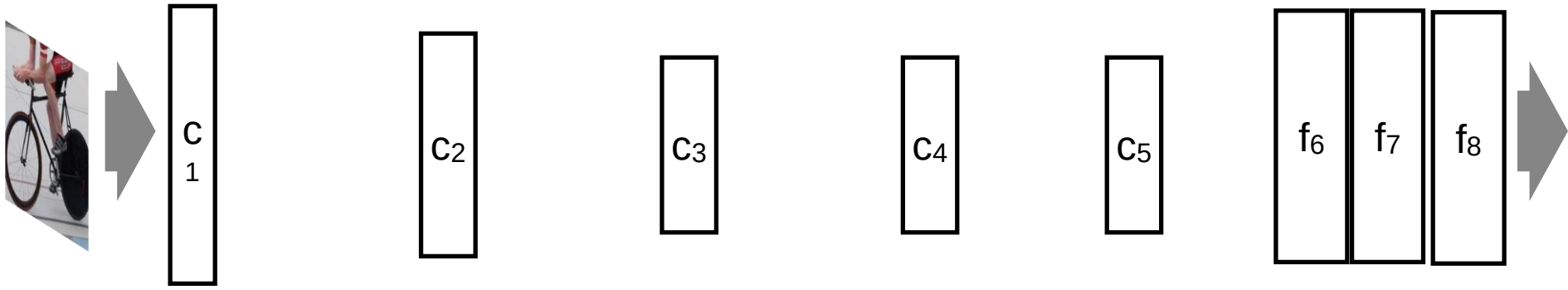
4. Typical CNN architectures for image classification

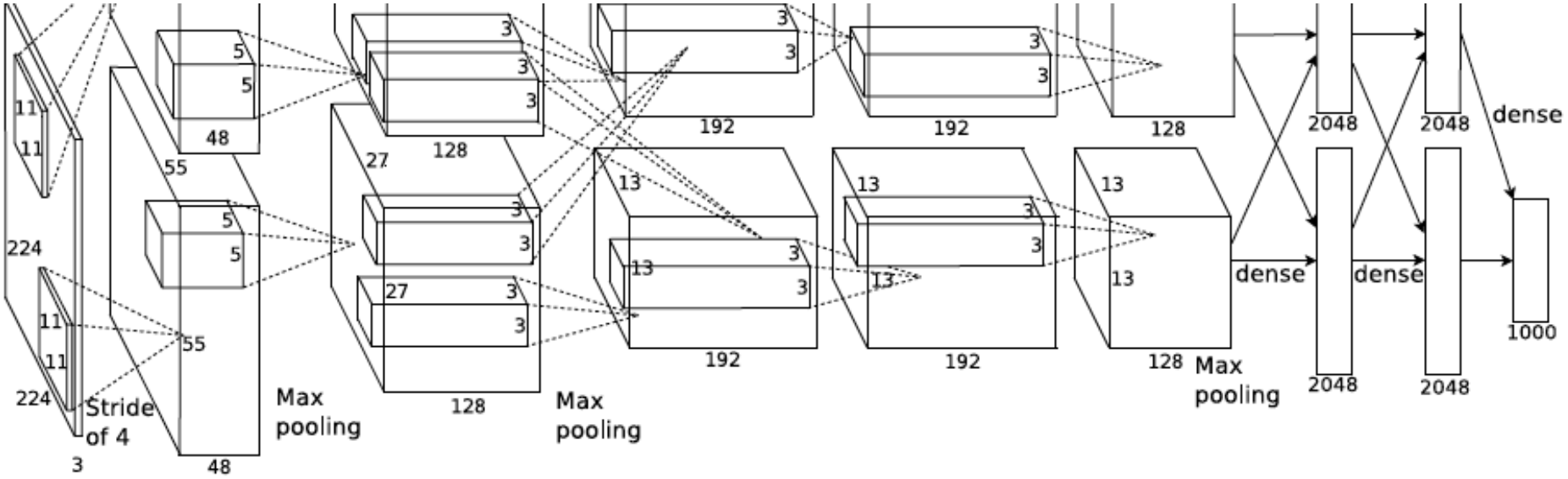
5. Beyond classification

Part 2: What is CNN learning?

Understanding and visualizing CNNs

Consider AlexNet architecture





Figures: A. Vedaldi

Visualizing Convnets

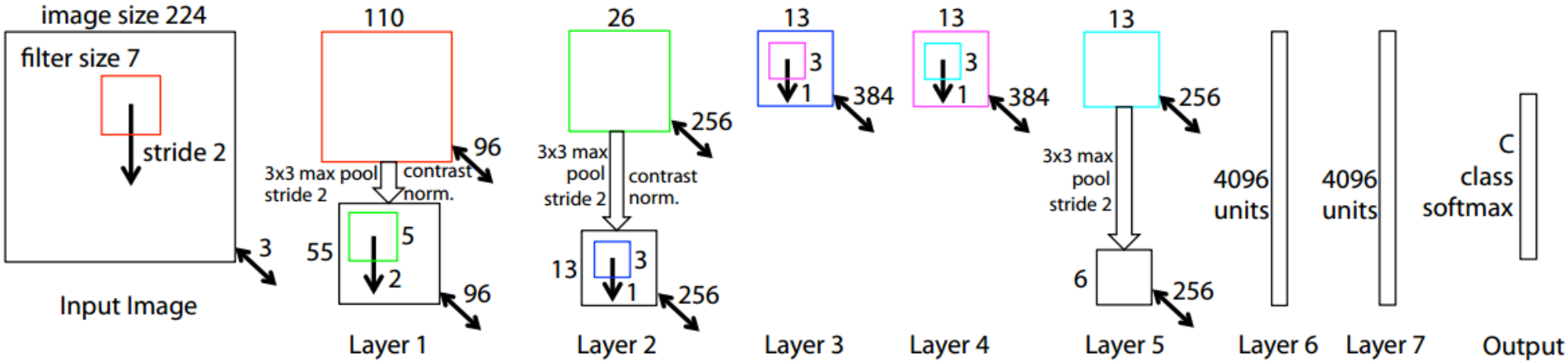
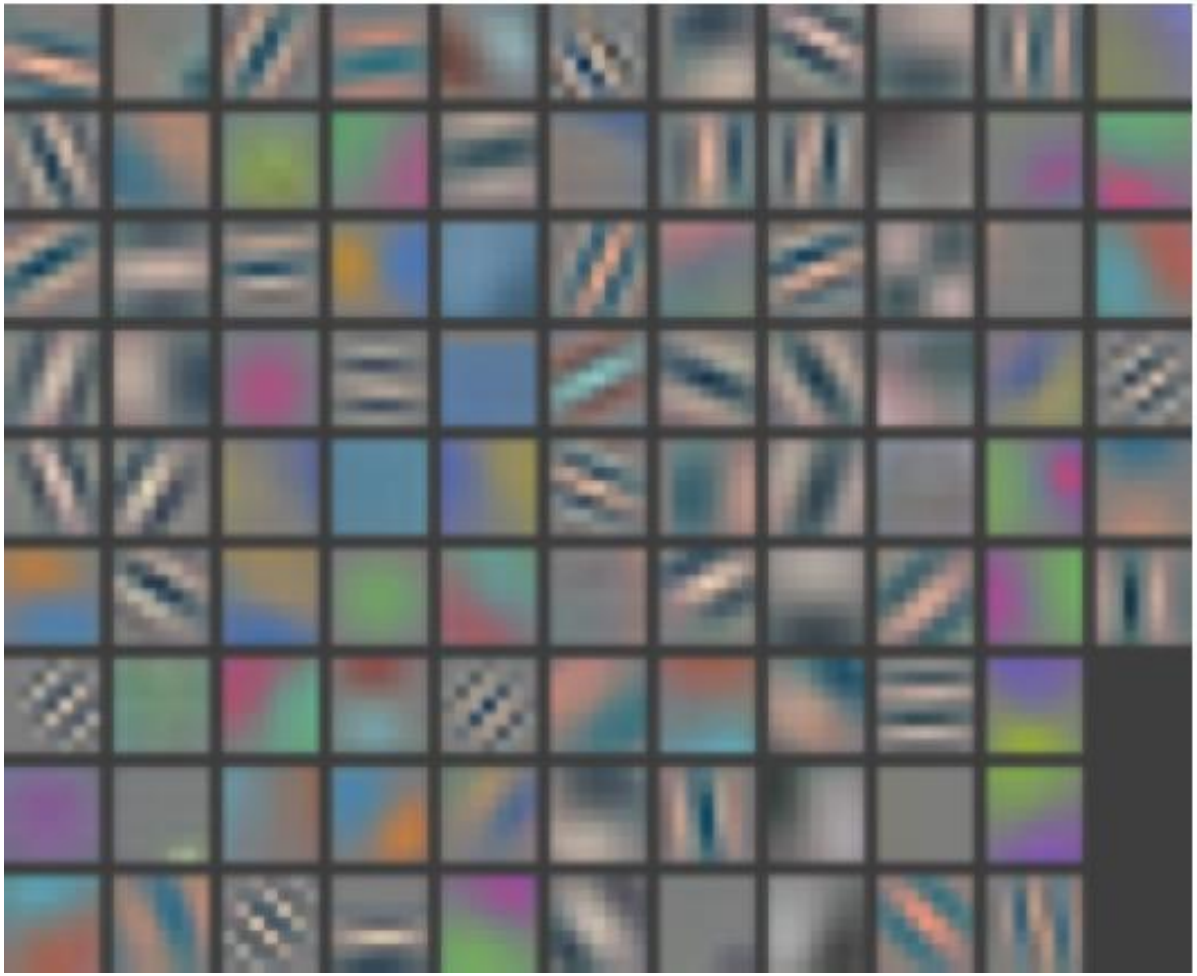


Figure 3. Architecture of our 8 layer convnet model. A 224 by 224 crop of an image (with 3 color planes) is presented as the input. This is convolved with 96 different 1st layer filters (red), each of size 7 by 7, using a stride of 2 in both x and y. The resulting feature maps are then: (i) passed through a rectified linear function (not shown), (ii) pooled (max within 3x3 regions, using stride 2) and (iii) contrast normalized across feature maps to give 96 different 55 by 55 element feature maps. Similar operations are repeated in layers 2,3,4,5. The last two layers are fully connected, taking features from the top convolutional layer as input in vector form ($6 \cdot 6 \cdot 256 = 9216$ dimensions). The final layer is a C-way softmax function, C being the number of classes. All filters and feature maps are square in shape.

M. Zeiler and R. Fergus, Visualizing and Understanding Convolutional Networks, arXiv preprint, 2013

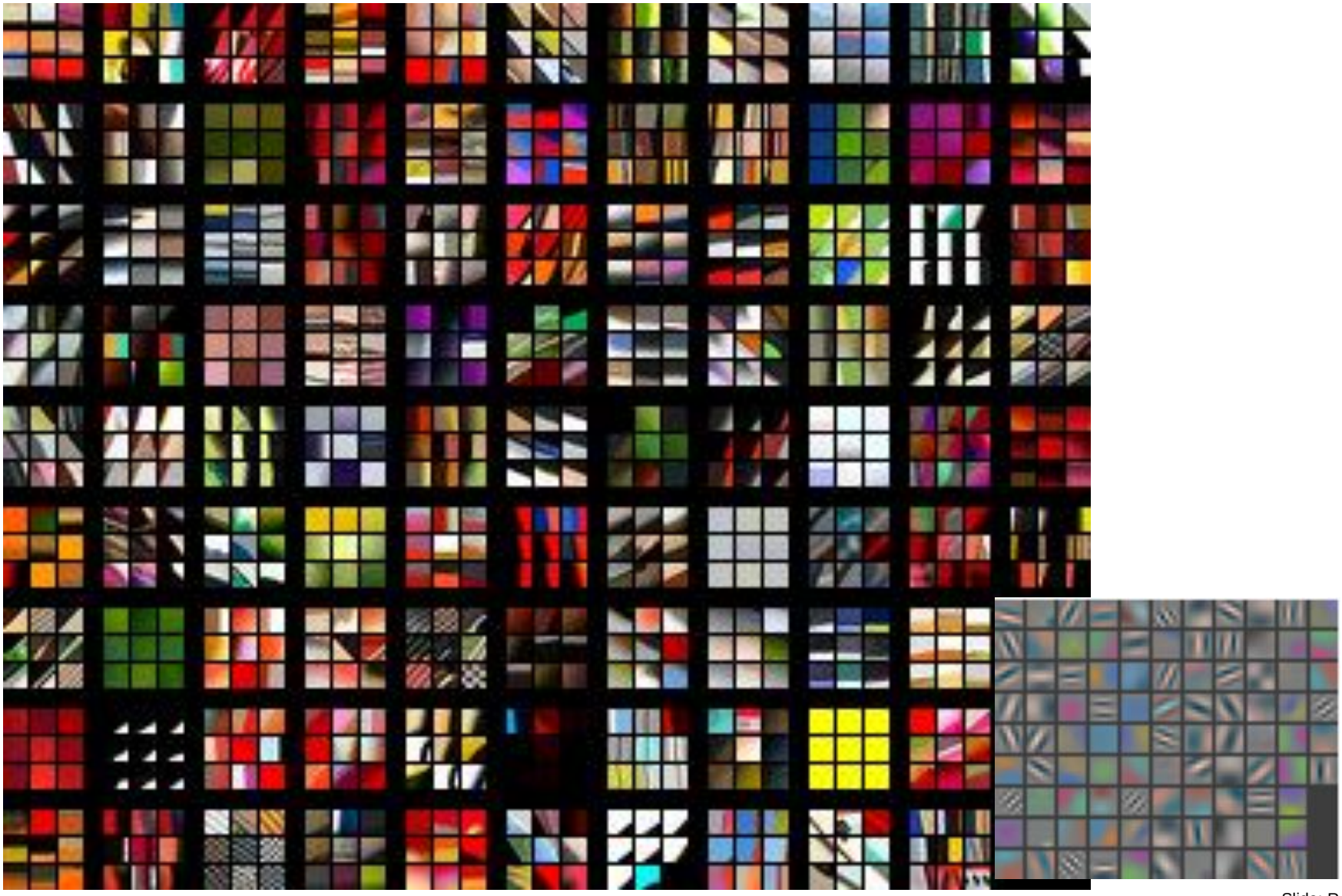
Slide: R. Fergus / S. Lazebnik

Layer 1 Filters



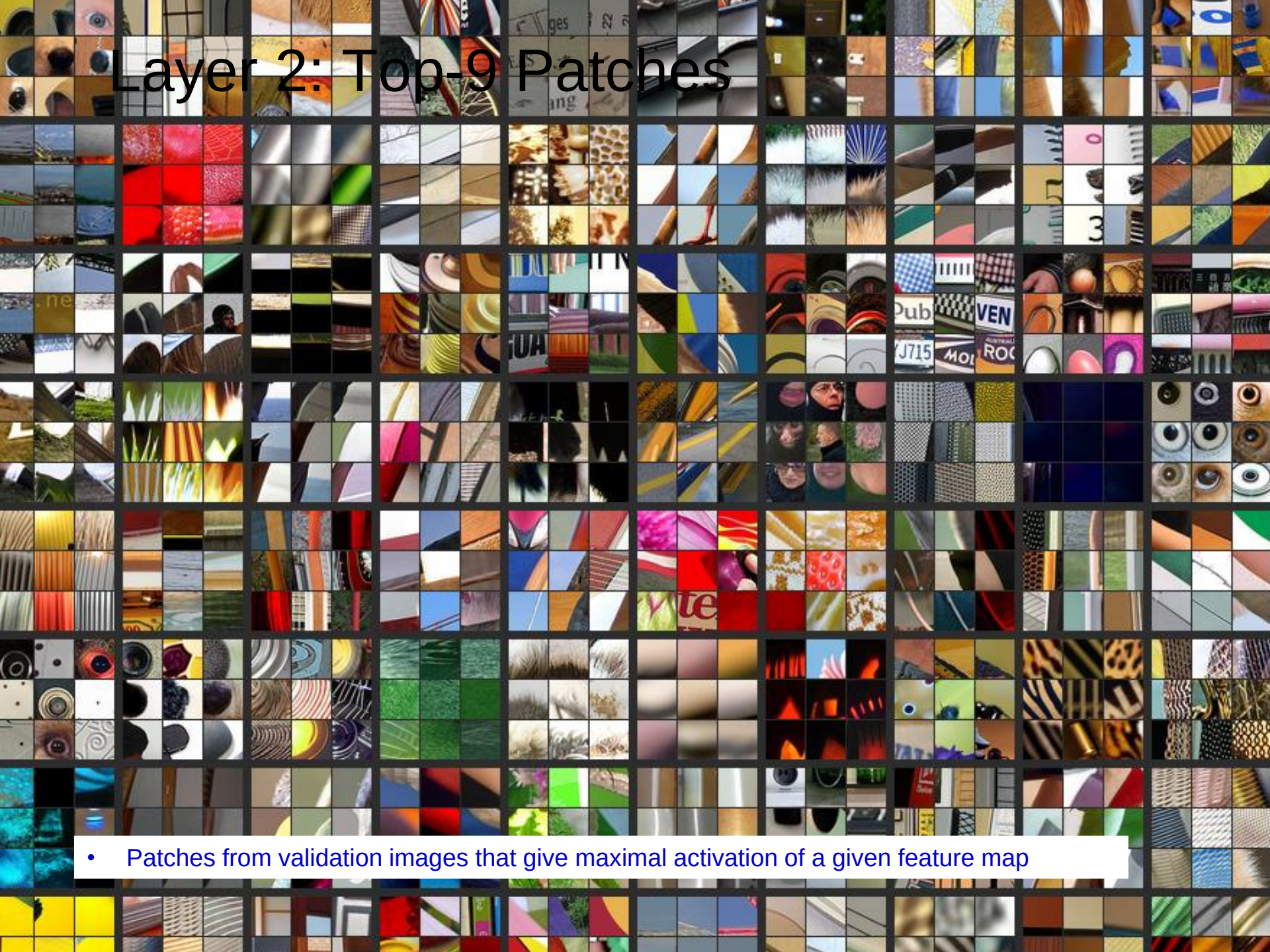
Slide: R. Fergus

Layer 1: Top-9 Patches



Slide: R. Fergus

Layer 2: Top-9 Patches

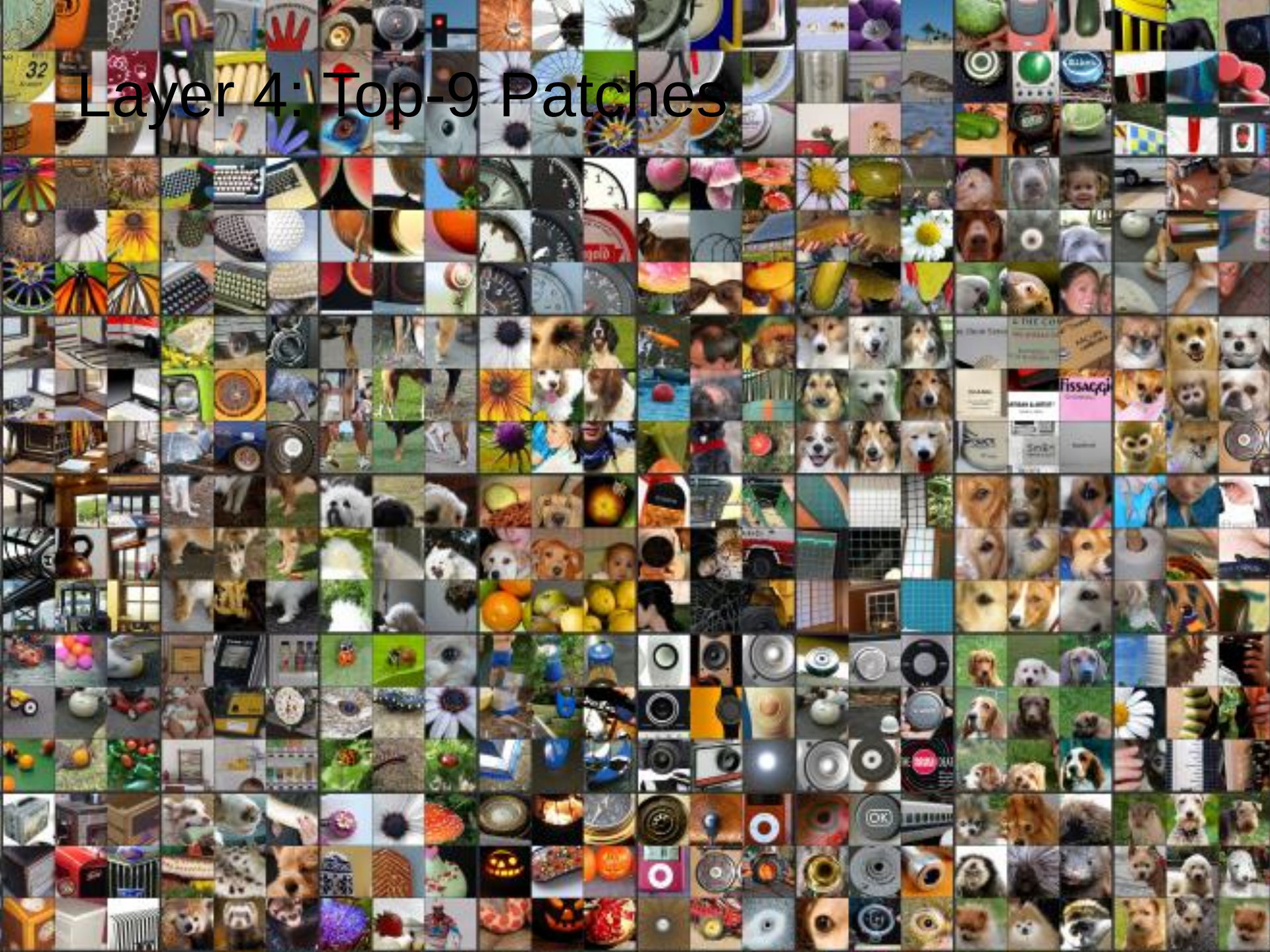


- Patches from validation images that give maximal activation of a given feature map

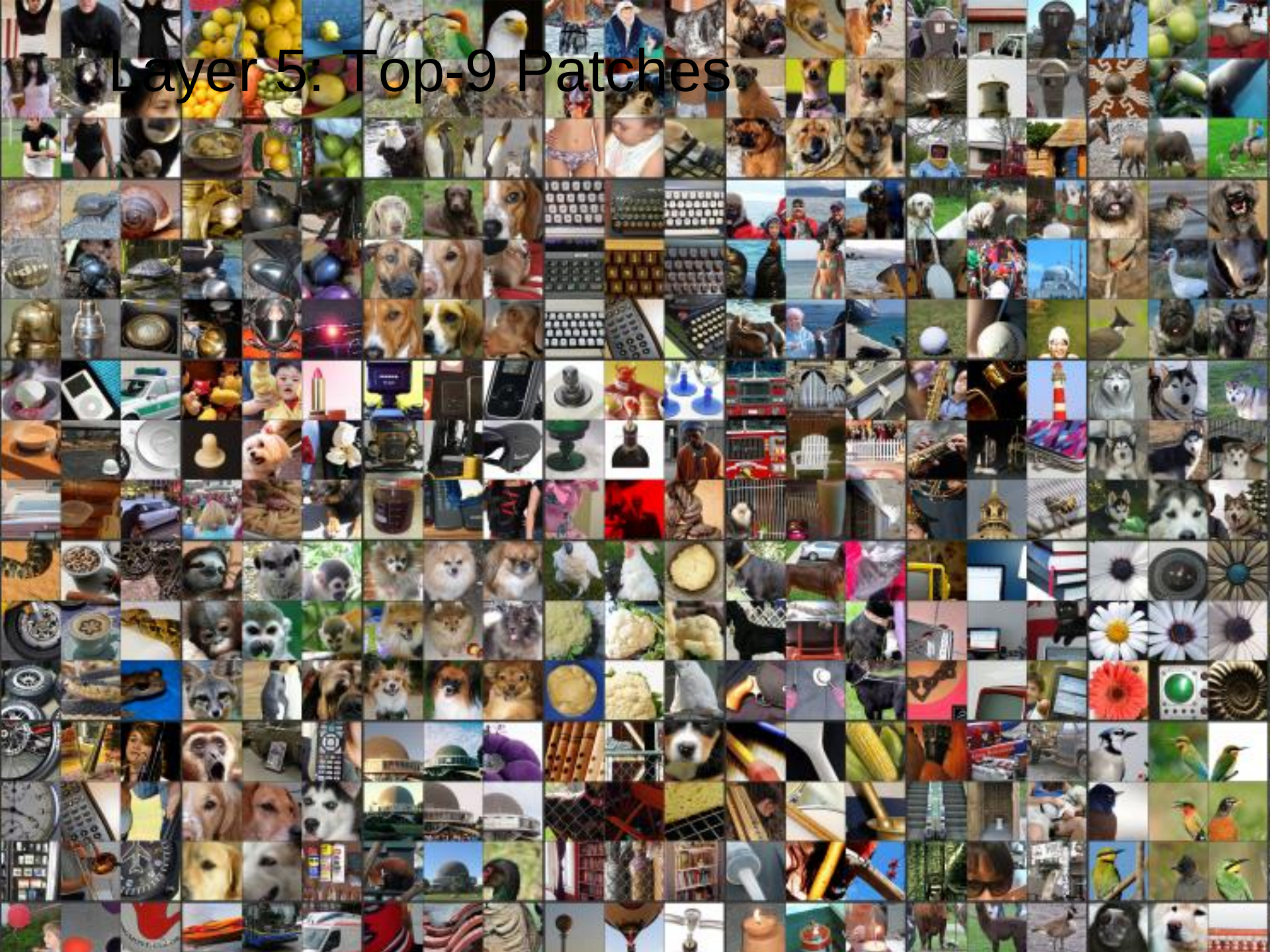
Layer 2: Top-9 Patches


Layer 3: Top-9 Patches

Layer 3: Top-9 Patches

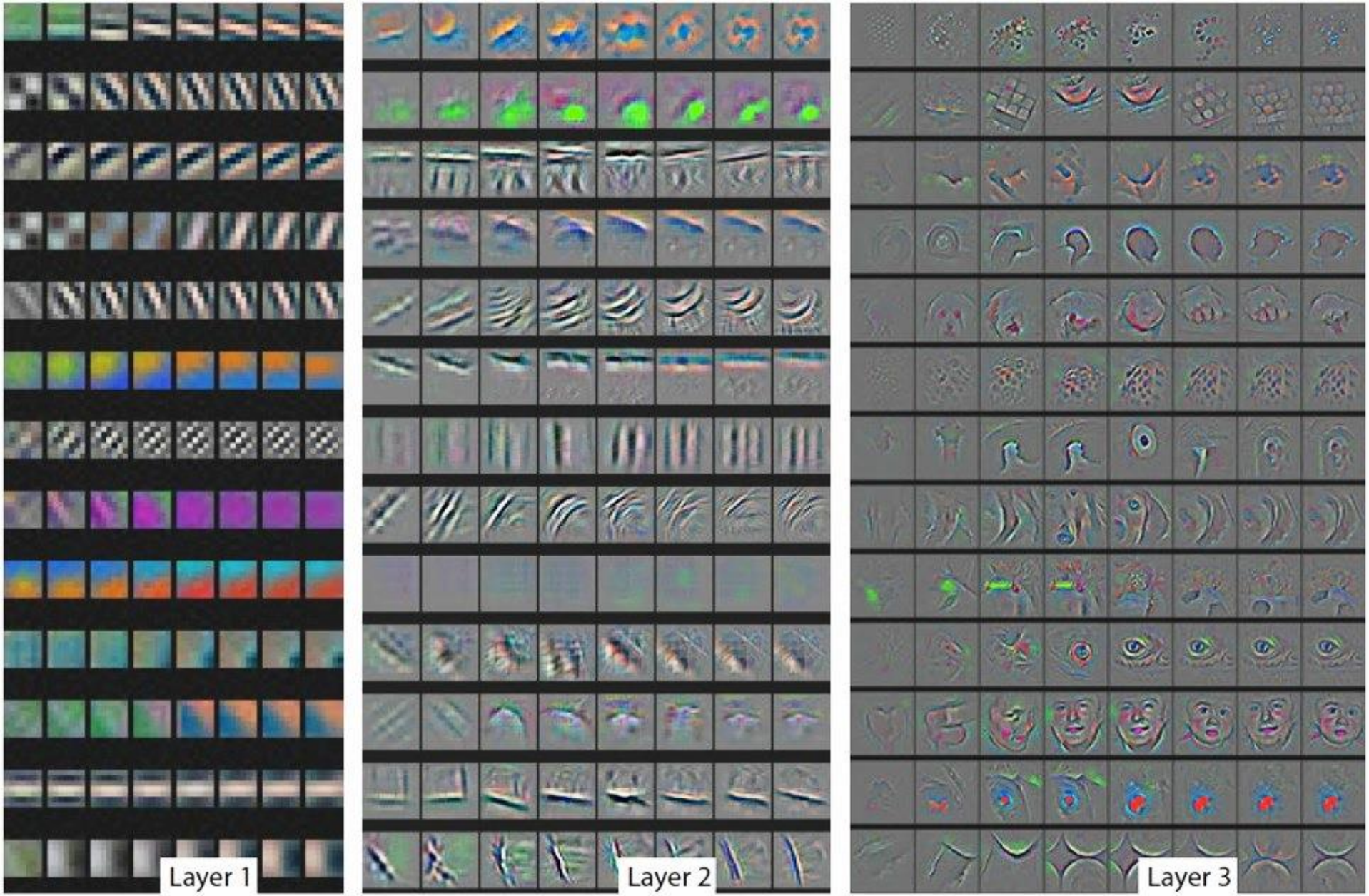

Layer 4: Top-9 Patches

Layer 4: Top-9 Patches

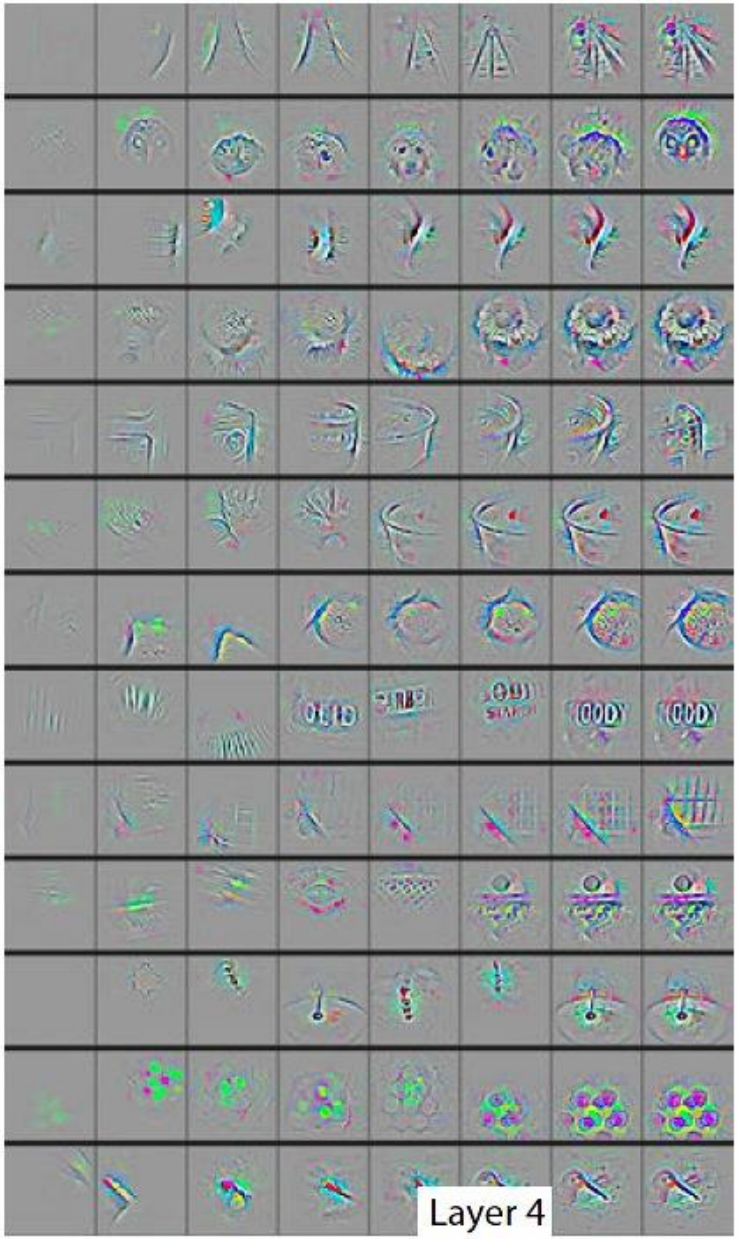

Layer 5: Top-9 Patches

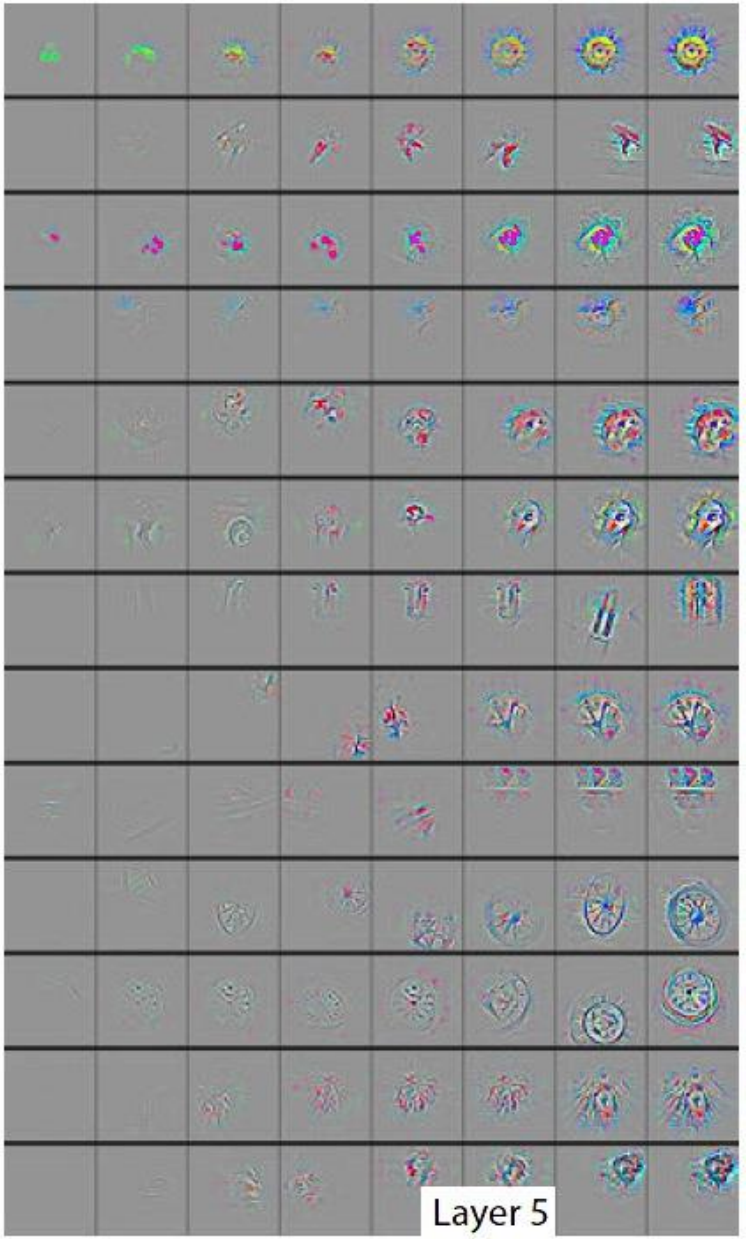
Layer 5: Top-9 Patches

Evolution of Features During Training



Evolution of Features During Training





Slide: R. Fergus

Diagnosing Problems

- Visualization of Krizhevsky et al.'s architecture showed some problems with layers 1 and 2
  - Large stride of 4 used
- Alter architecture: smaller stride & filter size
  - Visualizations look better
  - Performance improves

Slide: R. Fergus

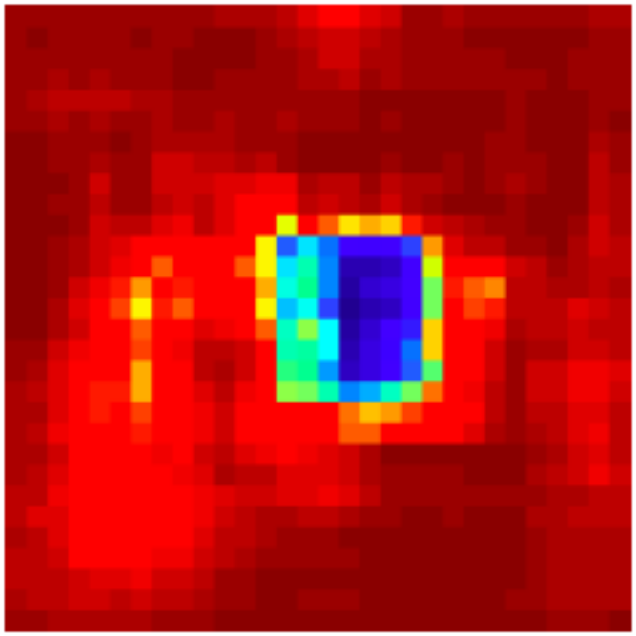
Occlusion Experiment

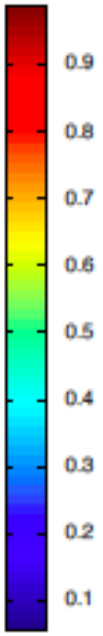
- Mask parts of input with occluding square
- Monitor output

Slide: R. Fergus

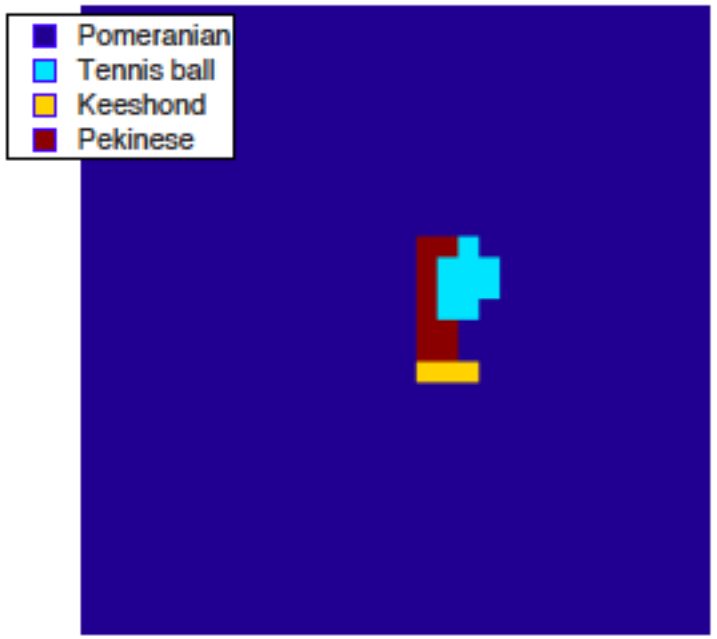
Input image

p(True class)





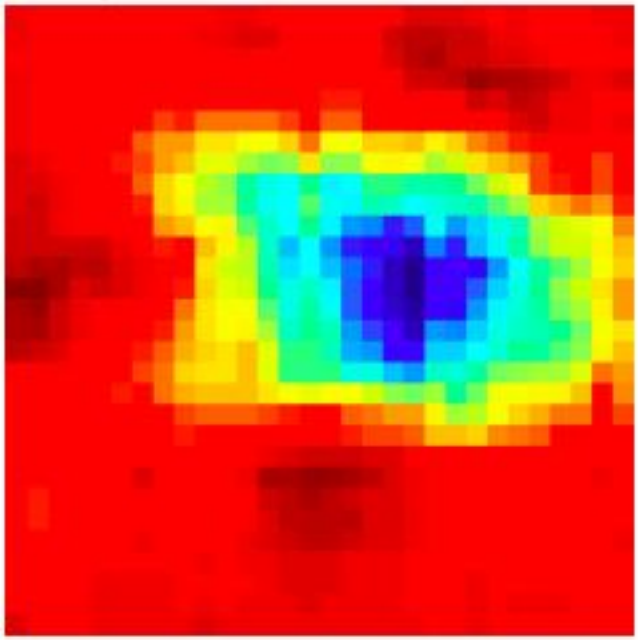
Most probable class



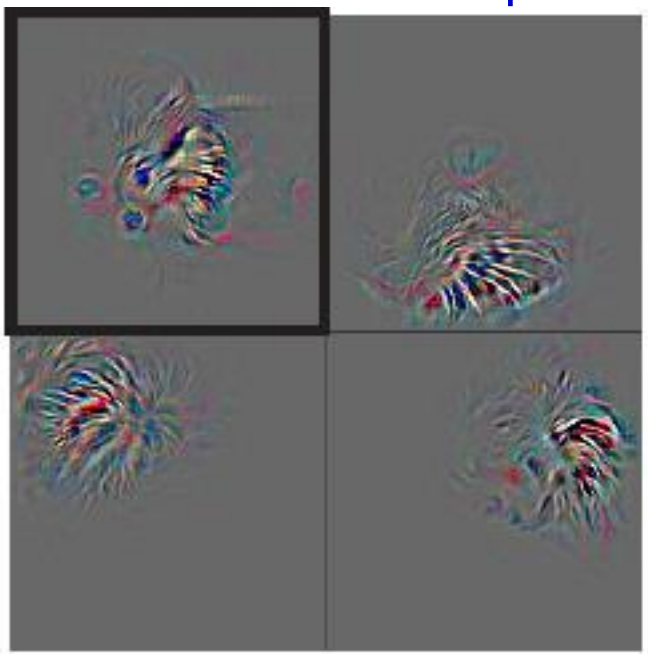
Slide: R. Fergus

Input image

Total activation in most active 5th layer feature map



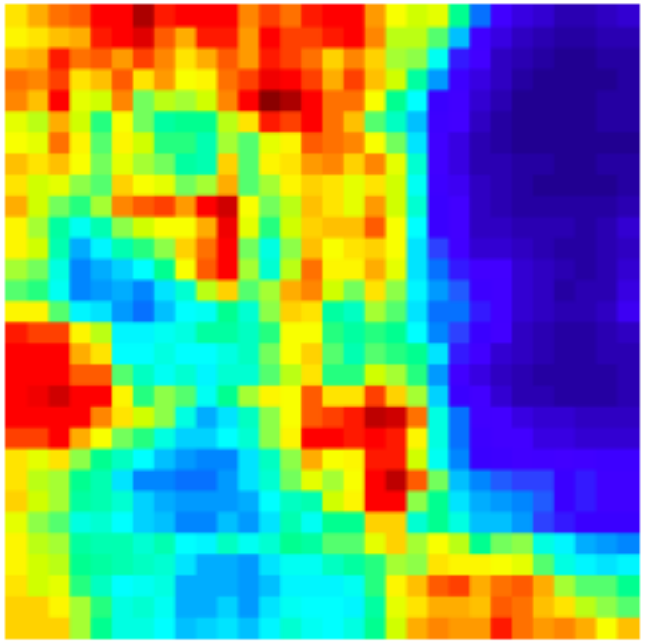
Other activations from same feature map

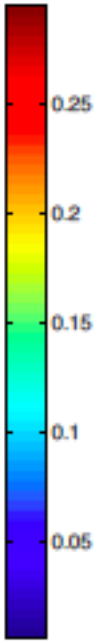


Slide: R. Fergus

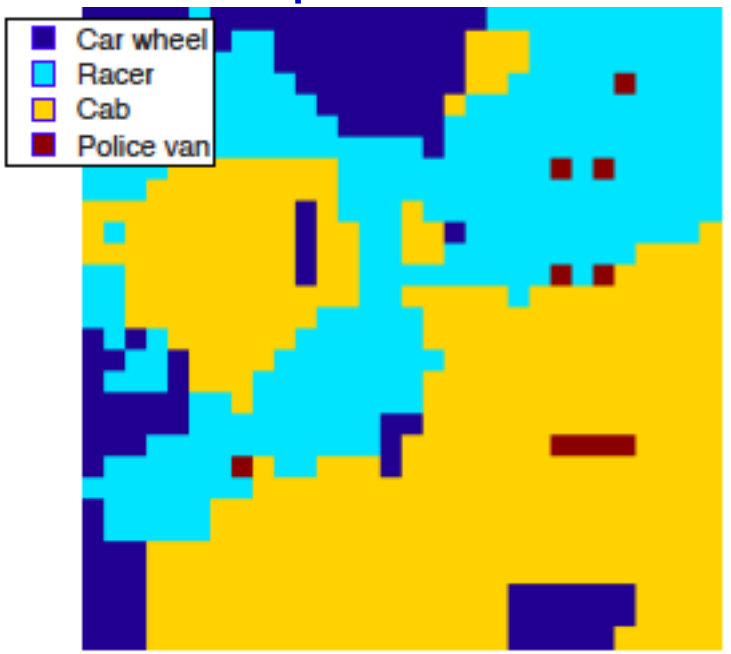
Input image

p(True class)





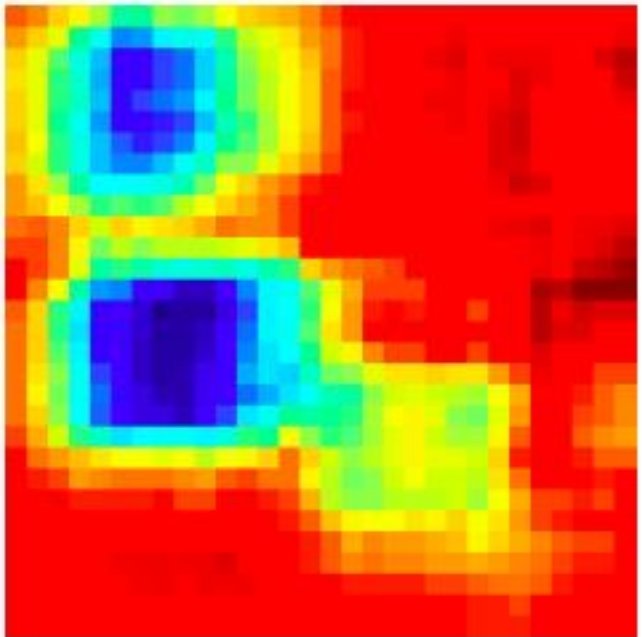
Most probable class



Slide: R. Fergus

Input image

Total activation in most active 5th layer feature map

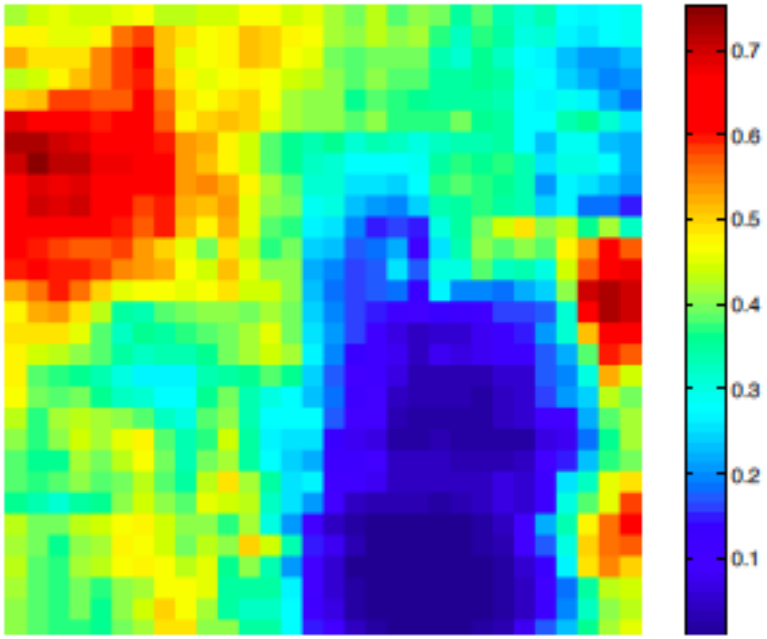


Other activations from same feature map

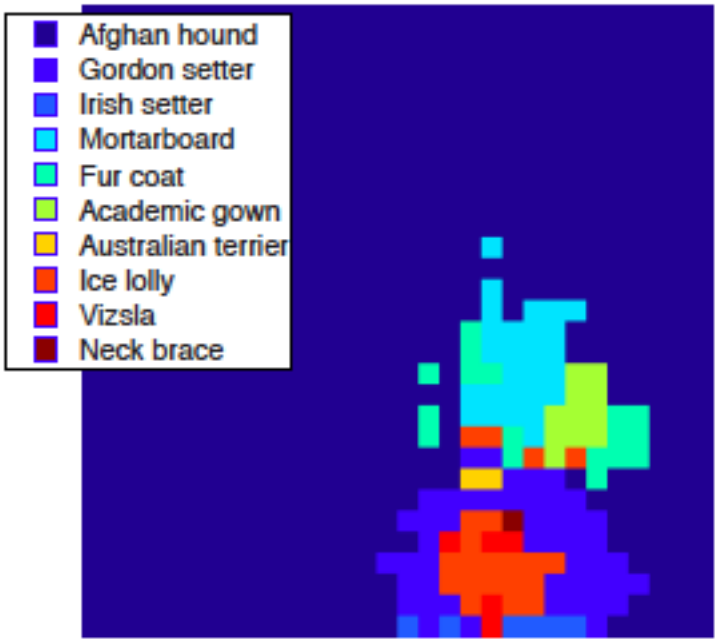
Slide: R. Fergus

Input image

p(True class)



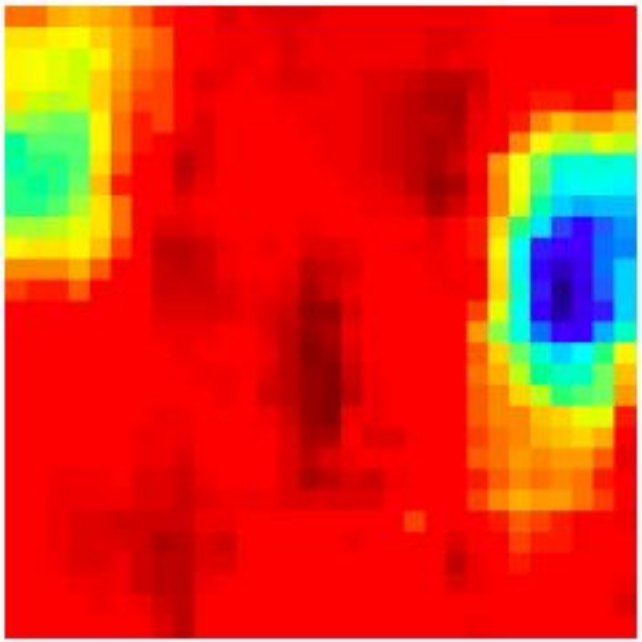
Most probable class



Slide: R. Fergus

Input image

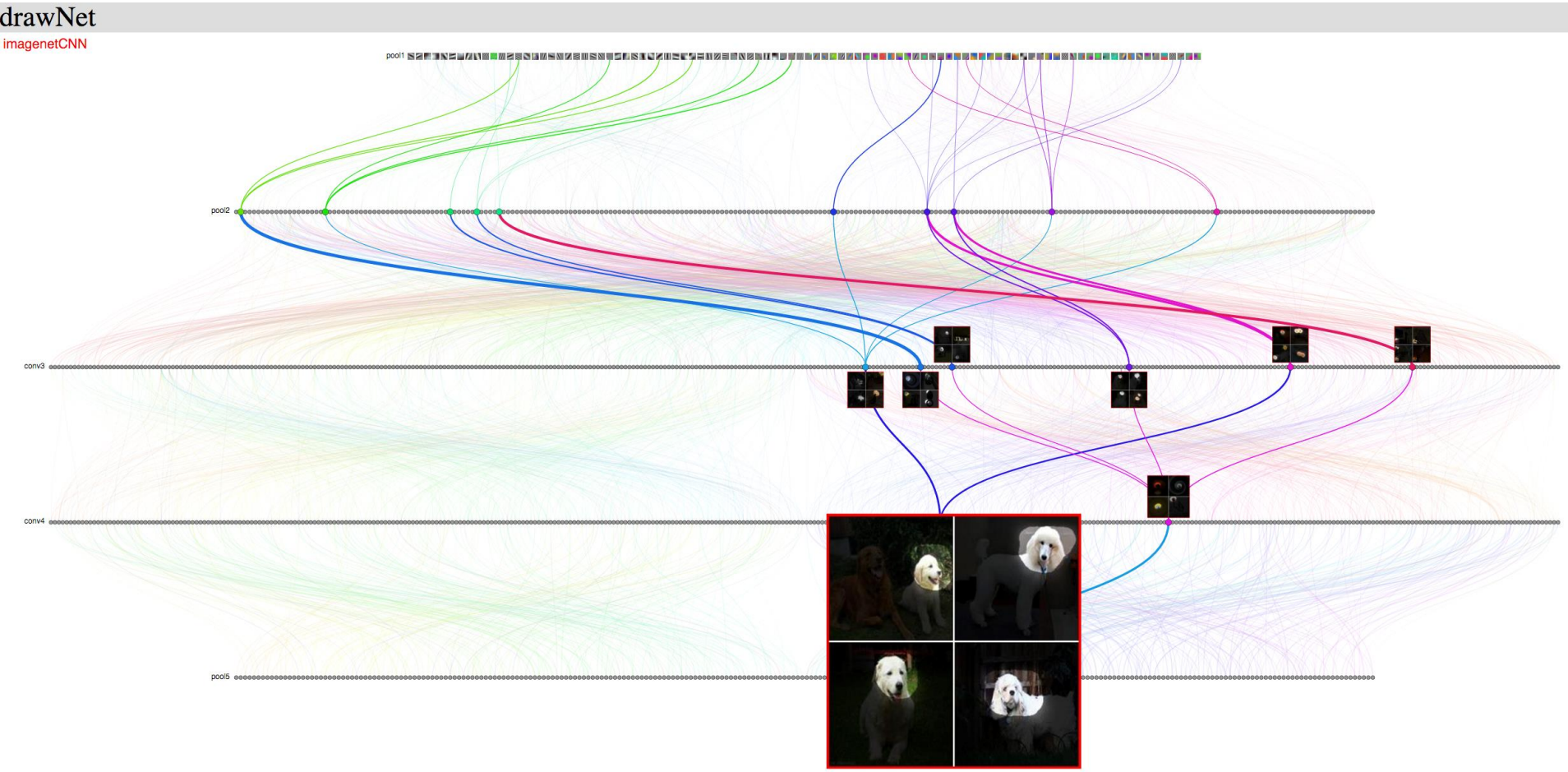
Total activation in most active 5th layer feature map



Other activations from same feature map

Slide: R. Fergus

Interactive visualization



http://people.csail.mit.edu/torralba/research/drawCNN/drawNet.html?path=imagenetCNN

Visualizing specific neurons

Deep Visualization Toolbox

yosinski.com/deepvis

#deepvis


Jason Yosinski


Jeff Clune


Anh Nguyen


Thomas Fuchs


Hod Lipson

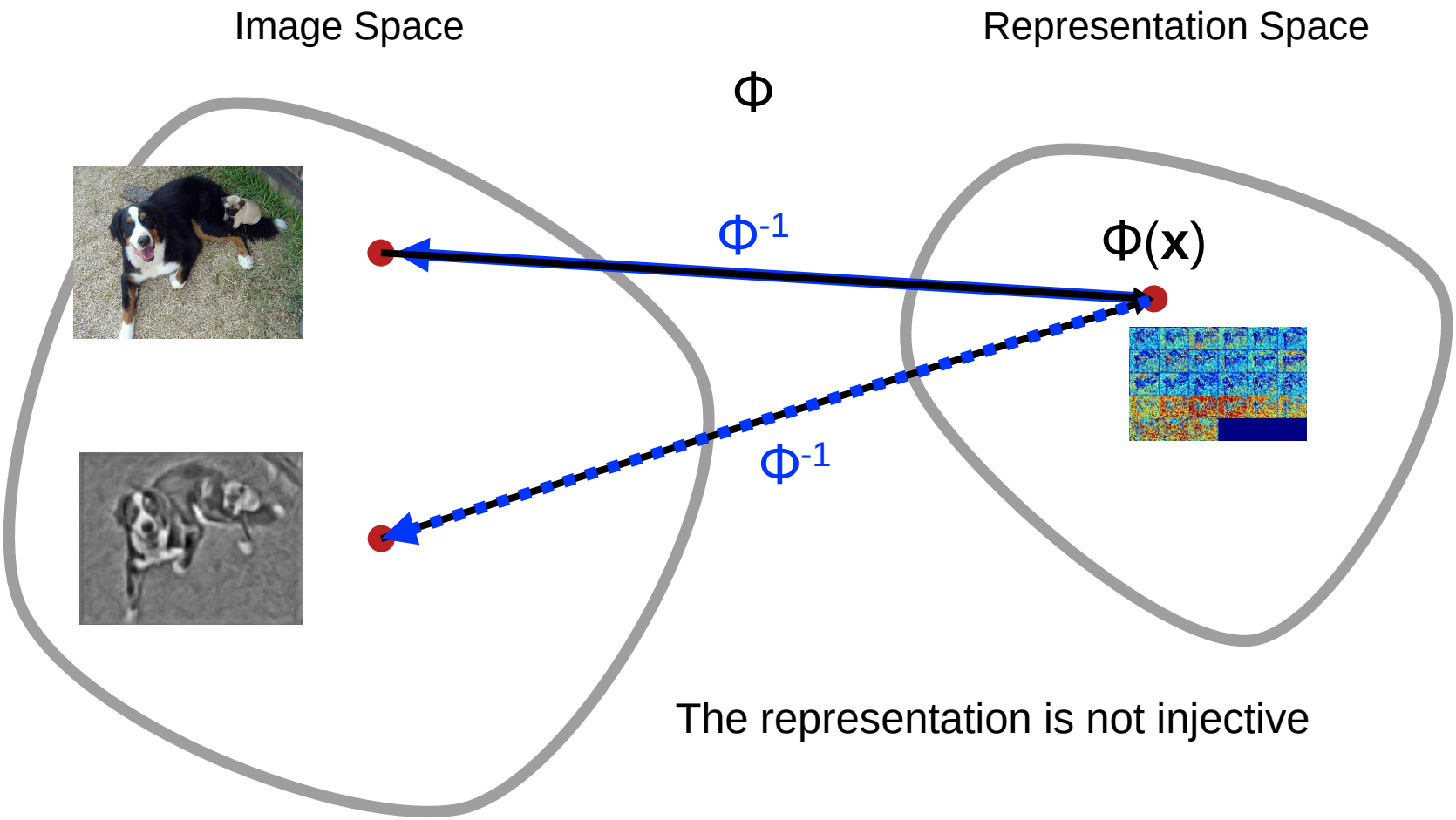
Cornell University

University of Wyoming

Jet Propulsion Laboratory
California Institute of Technology

[Yosinski et al., 2015], http://yosinski.com/deepvis#toolbox

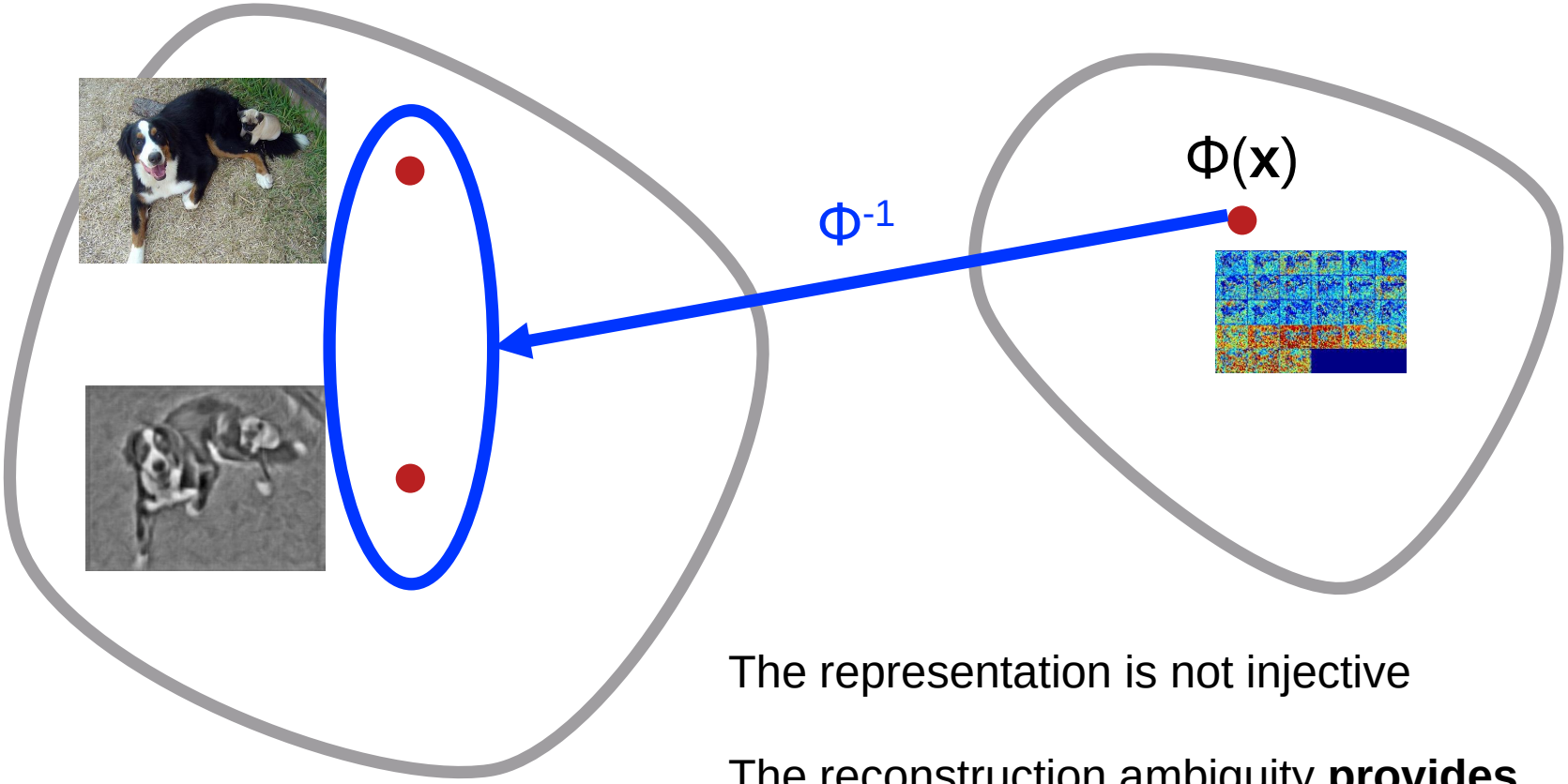
Visualization: Pre-Image



The representation is not injective

Slide: A. Vedaldi

Visualization: Pre-Image



The representation is not injective

The reconstruction ambiguity **provides useful information about the representation**

Slide: A. Vedaldi

Finding a Pre-Image

A simple yet general and effective method

$$\min_{\mathbf{x}} \|\Phi(\mathbf{x}) - \Phi_0\|_2^2$$

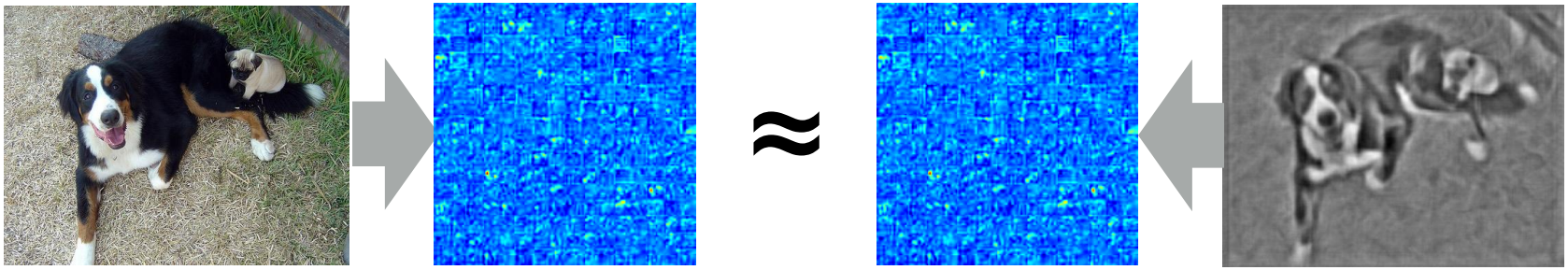


Image | Representation | Reconstruction | Pre-Image

Start from **random noise**

Optimize using stochastic **gradient descent**

Slide: A. Vedaldi

Finding a Pre-Image

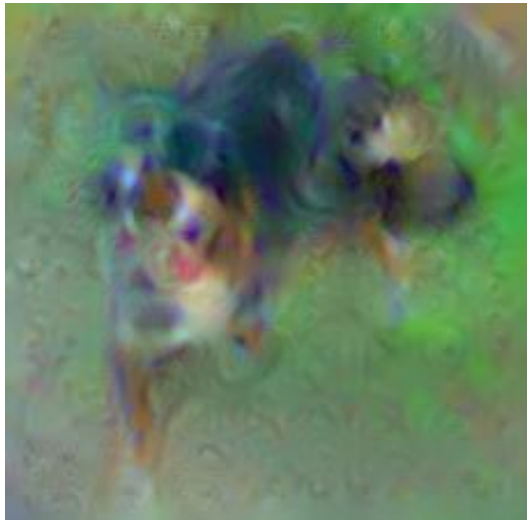
A simple yet general and effective method

$$\min_{\mathbf{x}} \|\Phi(\mathbf{x}) - \Phi_0\|_2^2$$

No prior

TV-norm β = 1

TV-norm β = 2



Slide: A. Vedaldi

Related Work

Analysis tools

Visualizing higher-layer features of a deep network
Ethan et al. 2009
[intermediate features]

Deep inside convolutional networks
Simonyan et al. 2014
[deepest features, aka “deep dreams”]

DeConvNets
Zeiler et al. In ECCV, 2014
[intermediate features]

Understanding neural networks through deep visualisation
Yosinksi et al. 2015
[intermediate features]

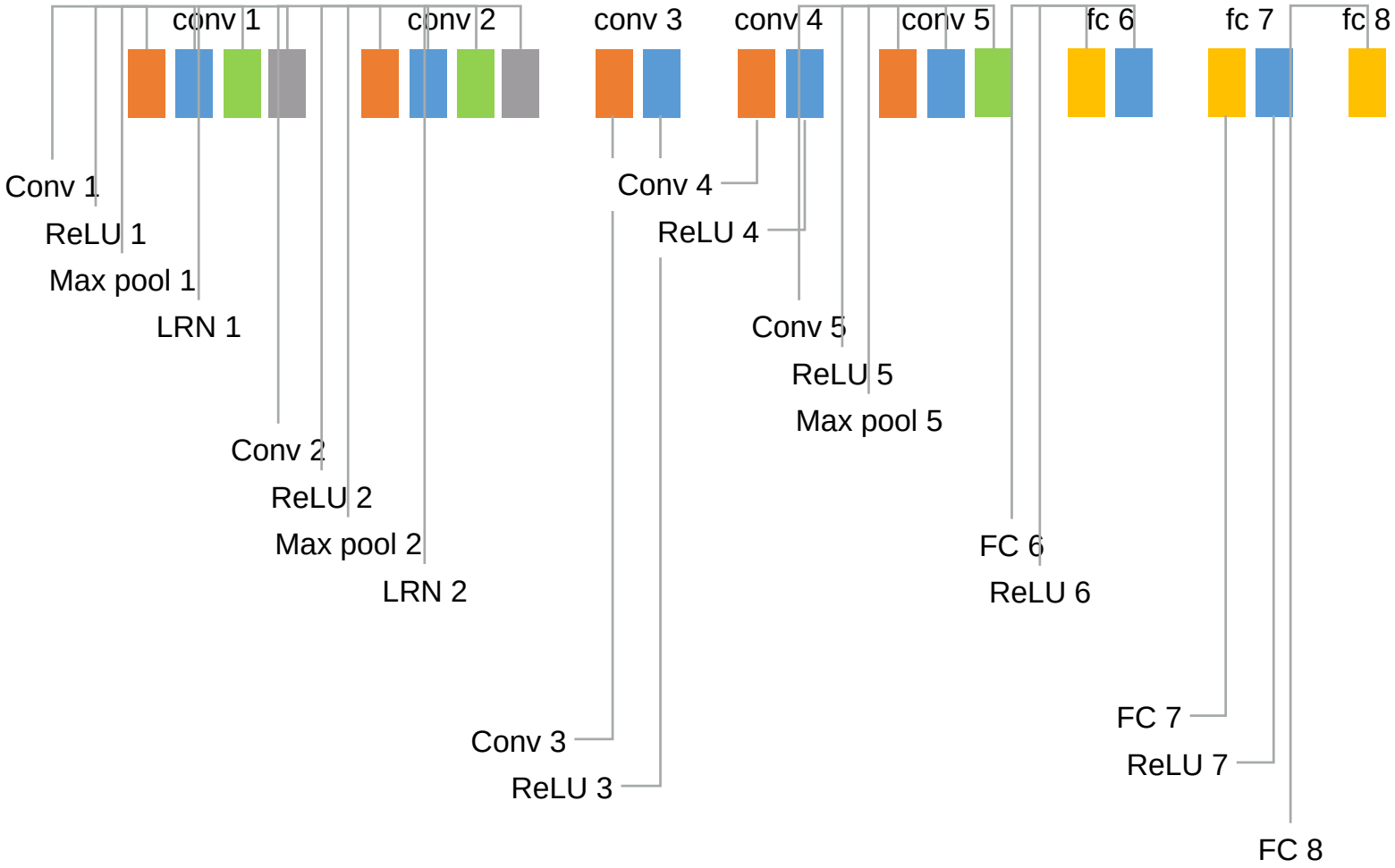
Artistic tools

Google’s “inceptionsm”
Mordvintsev et al. 2015

Style synthesis and transfer
Gatys et al. 2015

Slide: A. Vedaldi

Inverting a Deep CNN



AlexNet
[Krizhevsky et al. 2012]

Inverting a Deep CNN

Original Image

Slide: A. Vedaldi

Inverting a Deep CNN

Original Image

Slide: A. Vedaldi

Inverting a Deep CNN

Original Image

Slide: A. Vedaldi

Inverting a Deep CNN

Original Image

Slide: A. Vedaldi

Inverting a Deep CNN

Original Image

Slide: A. Vedaldi

Inverting a Deep CNN

Original Image

Slide: A. Vedaldi

Inverting a Deep CNN

Original Image

Slide: A. Vedaldi

Inverting a Deep CNN

Original Image

Slide: A. Vedaldi

Inverting a Deep CNN

Original Image

Slide: A. Vedaldi

Inverting a Deep CNN

Original Image

Slide: A. Vedaldi

Inverting a Deep CNN

Original Image

Slide: A. Vedaldi

Inverting a Deep CNN

Original Image

Slide: A. Vedaldi

Inverting a Deep CNN

Original Image

Slide: A. Vedaldi

Inverting a Deep CNN

Original Image

Slide: A. Vedaldi

Inverting a Deep CNN

Original Image

Slide: A. Vedaldi

Inverting a Deep CNN

Original Image

Slide: A. Vedaldi

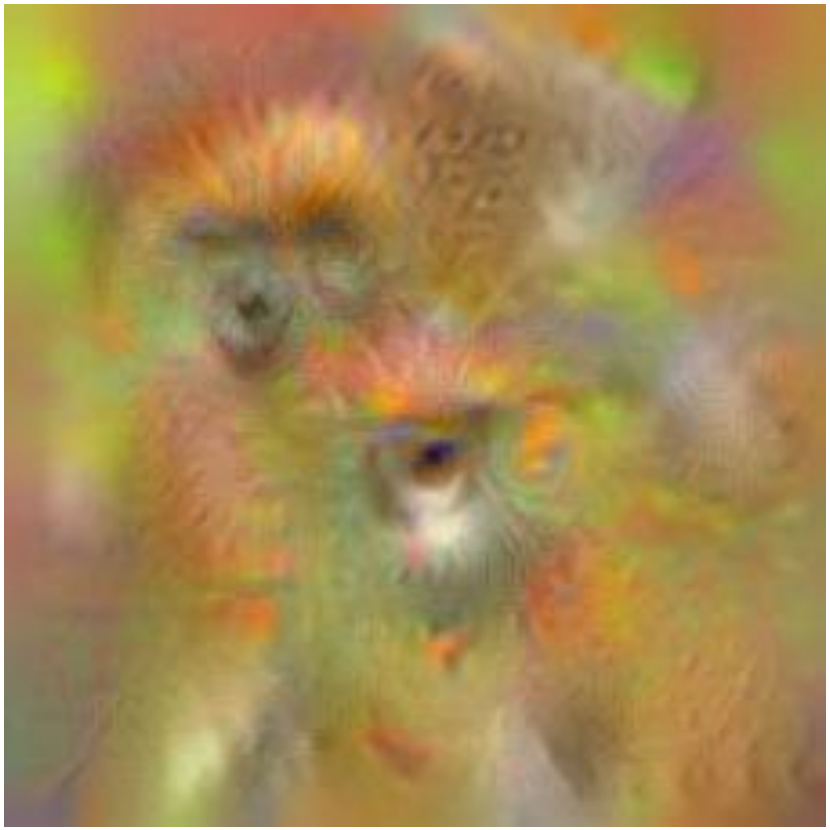
Inverting a Deep CNN

Original Image

Slide: A. Vedaldi

Inverting a Deep CNN

Original Image



Slide: A. Vedaldi

Inverting a Deep CNN

Original Image

Slide: A. Vedaldi

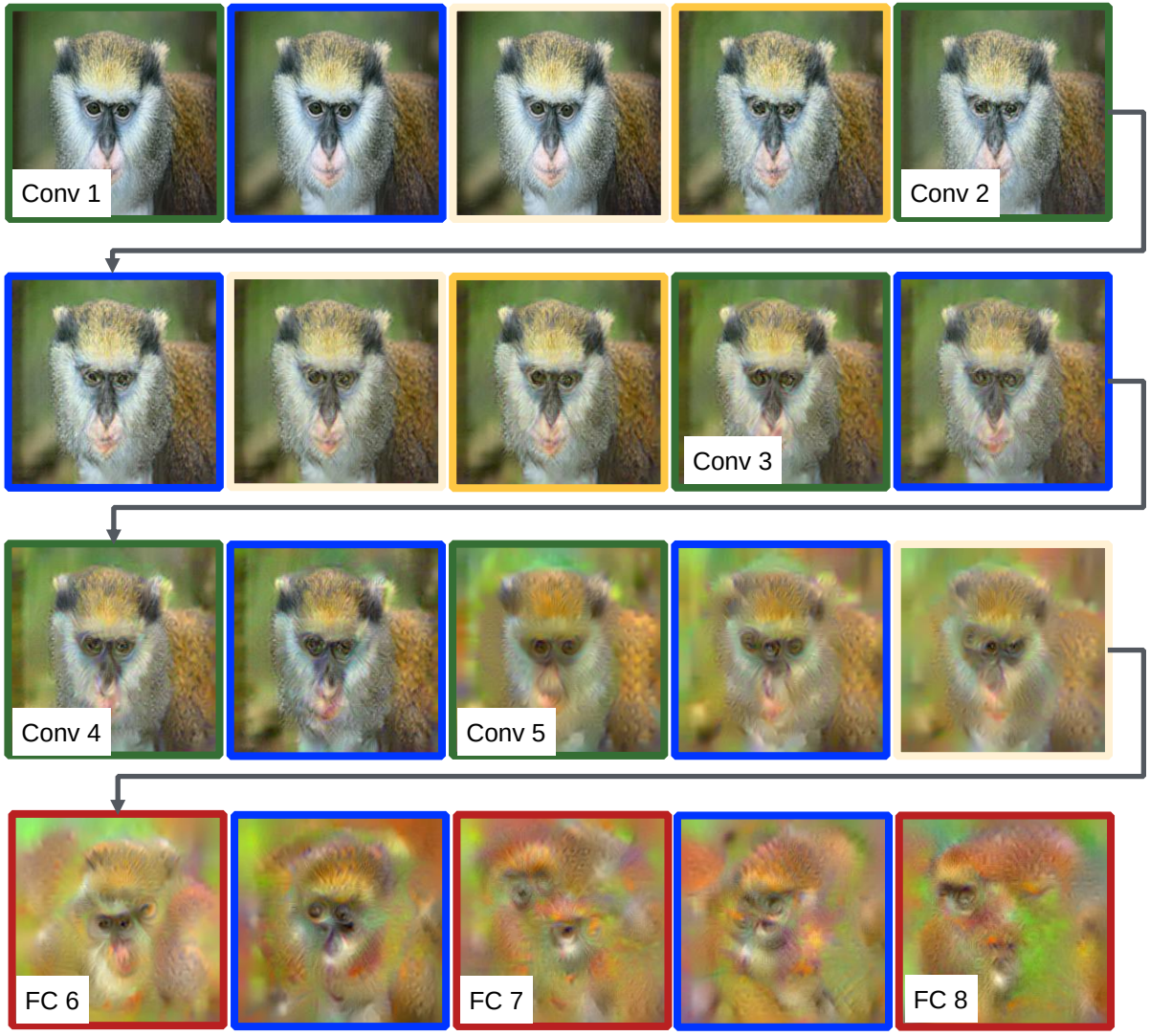
Inverting a Deep CNN

Original Image

Slide: A. Vedaldi

Inverting a Deep CNN

Original Image



Slide: A. Vedaldi

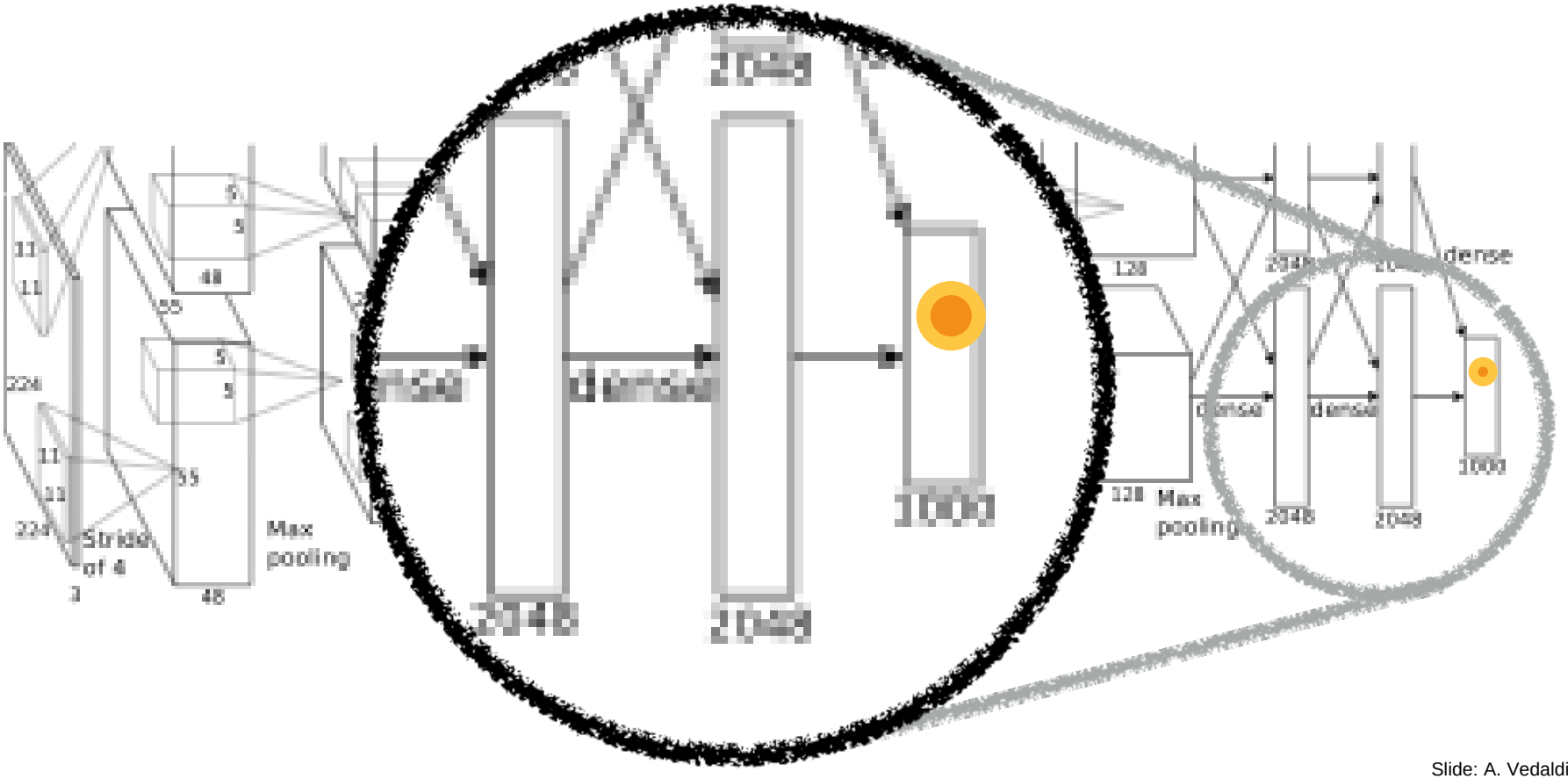
CNNs = visual codes?

Slide: A. Vedaldi

Activation Maximization

Look for an image that maximally activates a **specific feature component**

$$\min_{\mathbf{x}} -\langle \mathbf{e}_k, \Phi(\mathbf{x}) \rangle + R_{TV}(\mathbf{x}) + R_{\alpha}(\mathbf{x})$$



Slide: A. Vedaldi

Related Work

Visualizing higher-layer features of a deep network
Ethan et al. 2009
[intermediate features]

Deep inside convolutional networks
Simonyan et al. 2014
[deepest features, aka “deep dreams”]

Google “inceptionsm”
Mordvintsev et al. 2015

Understanding neural networks through deep visualisation
Yosinksi et al. 2015
[intermediate features]

Slide: A. Vedaldi

Outline

1. Convolutional neural networks (CNNs)

2. Understanding and visualizing CNN representations

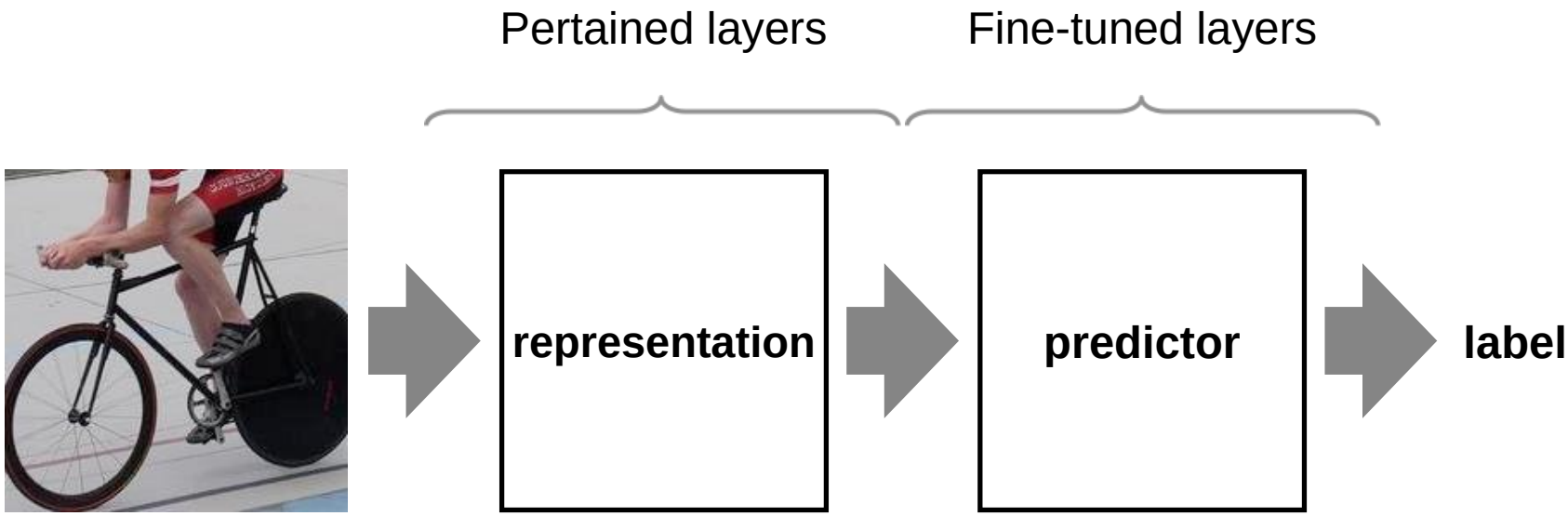
3. Transferring learnt representation to other tasks

4. Typical CNN architectures for image classification

5. Beyond classification

Pre-training and transfer learning

[Evaluations in A. S. Razavian, 2014, Chatfield et al., 2014]



CNN as universal representations

- First several layers in most CNNs are generic
- They can be reused when training data is comparatively scarce

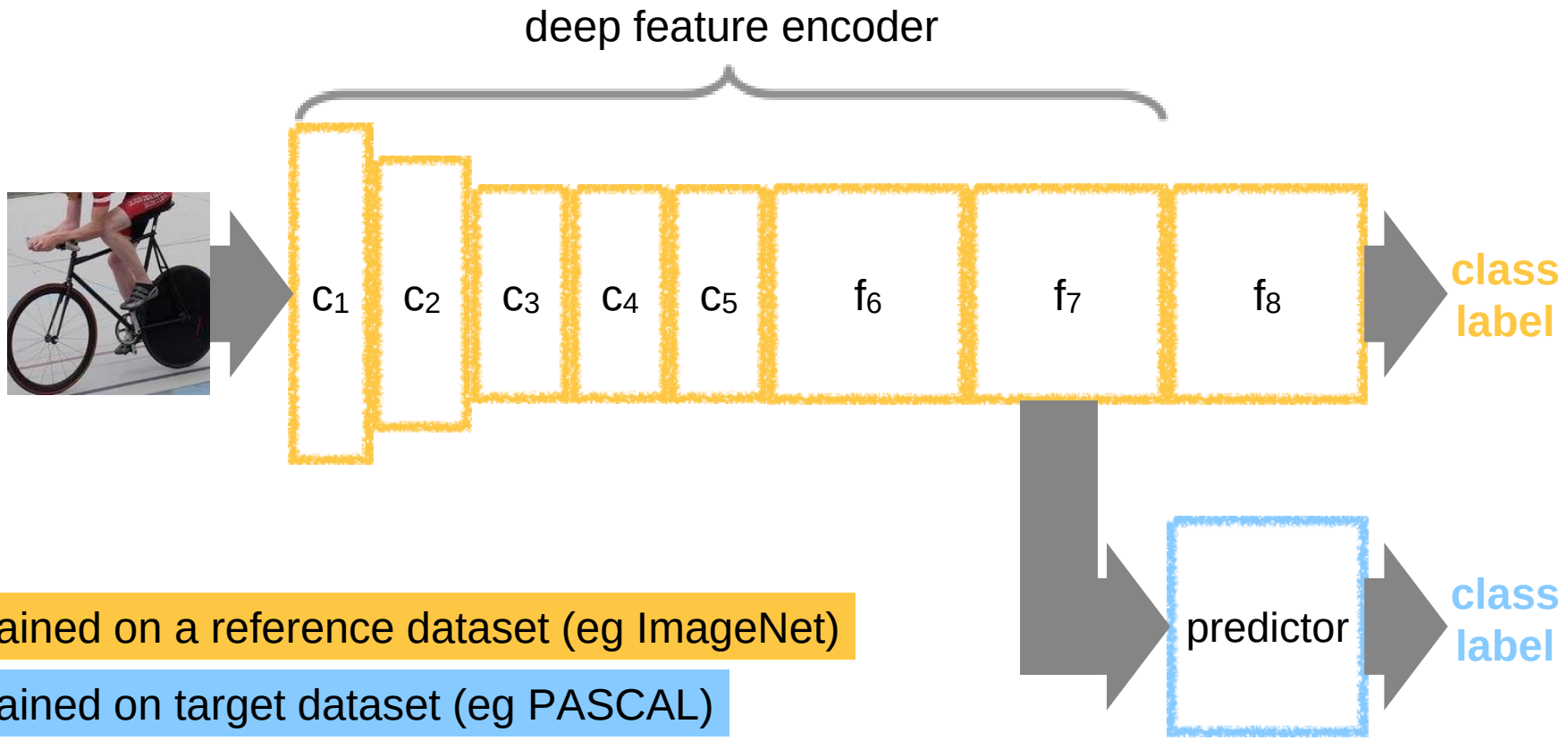
Application

- Pre-train on ImageNet classification
  1M images
- Cut at some deep conv or FC layer
  to get features

Slide: A. Vedaldi

Transfer learning

Deep representations are generic

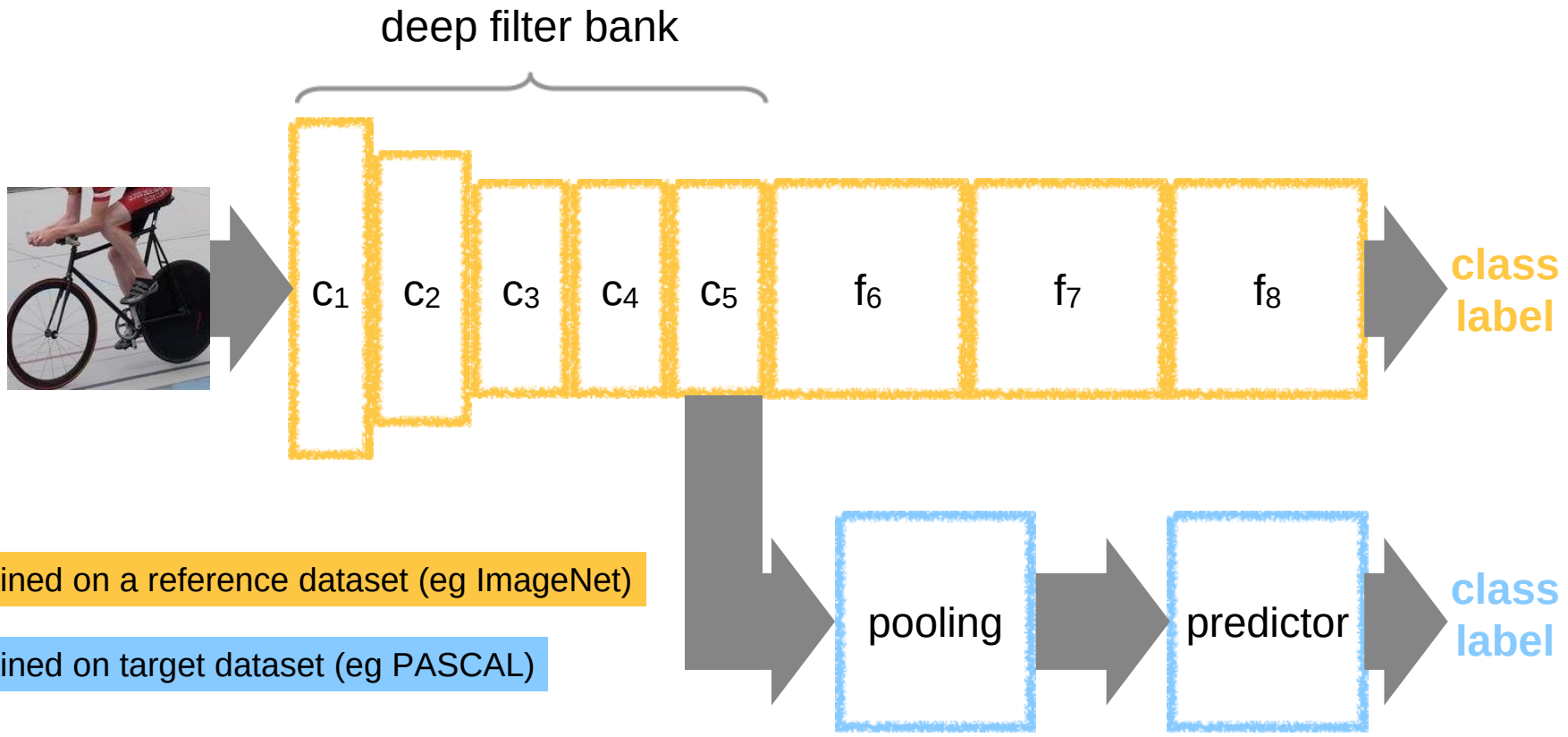


A general purpose deep encoder is obtained by chopping off the last layers of a CNN trained on a large dataset.

Slide: A. Vedaldi

CNNs as filter banks

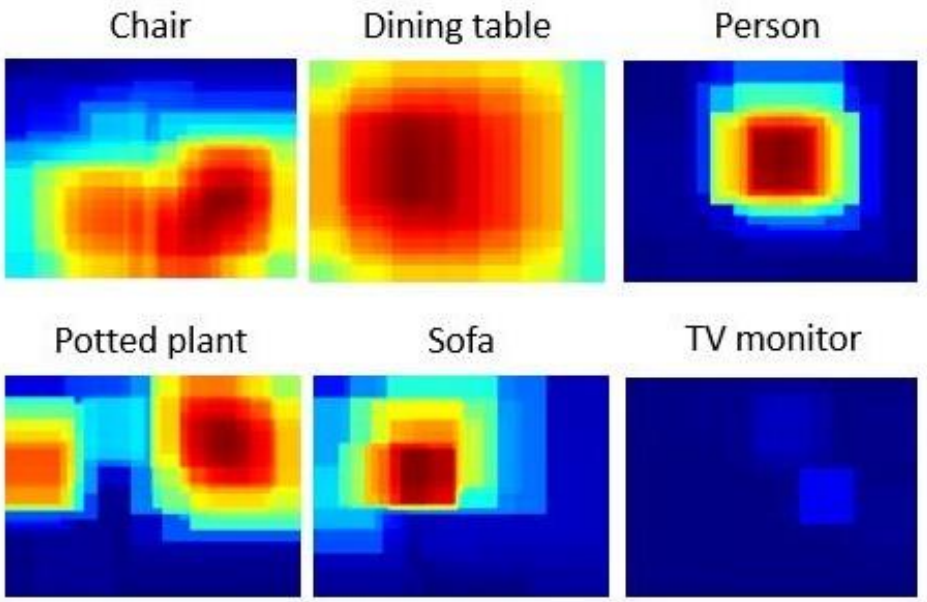
Deep representations used as local features



In R-CNN and similar models, the most important shared component are the convolutional features.

Slide: A. Vedaldi

Example



Learning and Transferring Mid-Level Image Representations using Convolutional Neural Networks

M. Oquab, L. Bottou, I. Laptev, J. Sivic

In CVPR 2014

http://www.di.ens.fr/willow/research/cnn/

Slide: A. Vedaldi

ImageNet classification challenge

ImageNet classification challenge

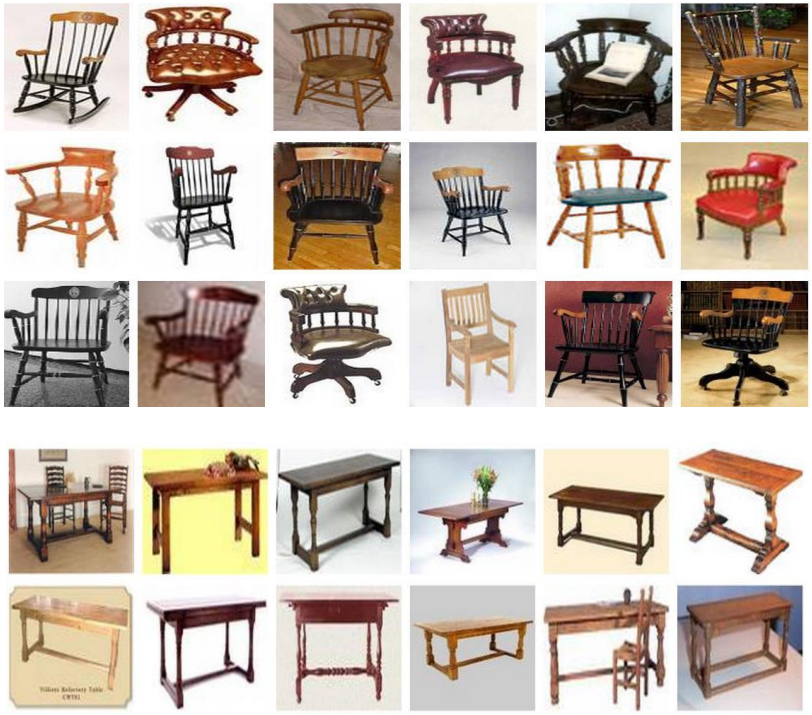
Object centric

1000 classes

1.2M images

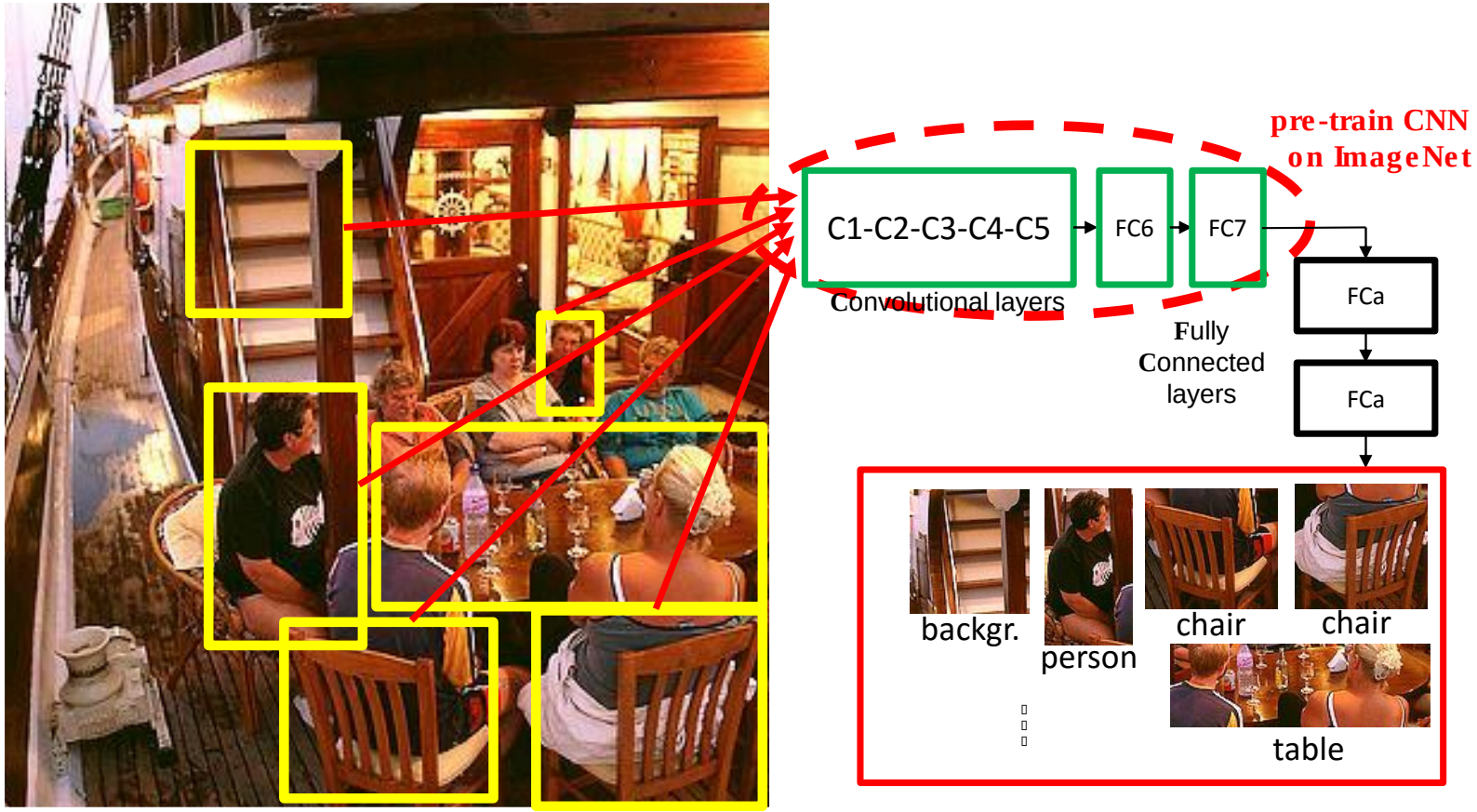
Let’s look at the data

A typical image with chairs and tables
on Flickr.com



Images of chairs and tables
in ImageNet

How to use CNNs in cluttered scenes?

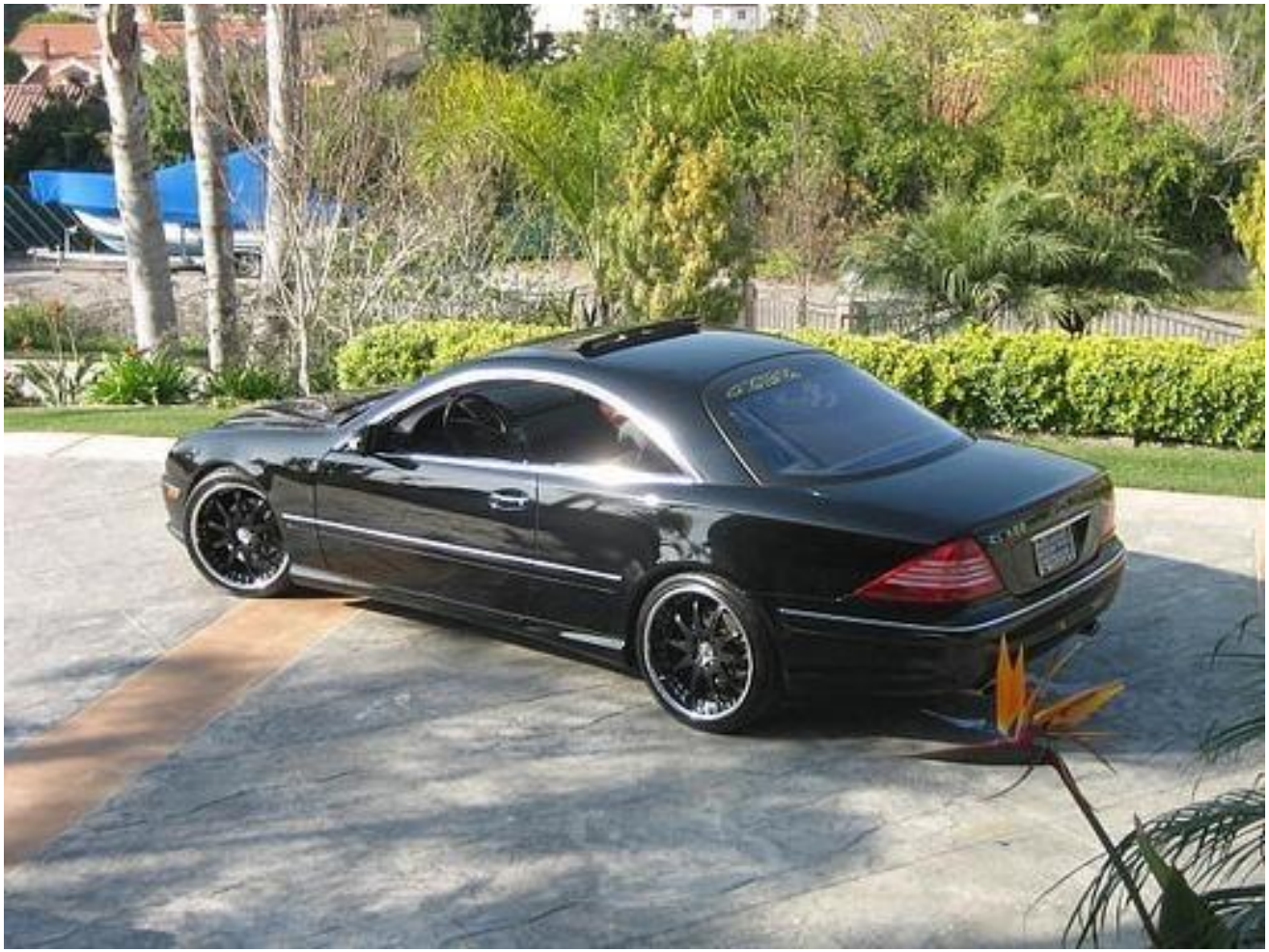


Use ImageNet pre-trained CNN as a representation.

Post-train on new task.

[Girshick et al.'14], [Oquab et al.'14], [Sermanet et al.'13], [Donahue et al. '13], [Zeiler & Fergus '13] ...

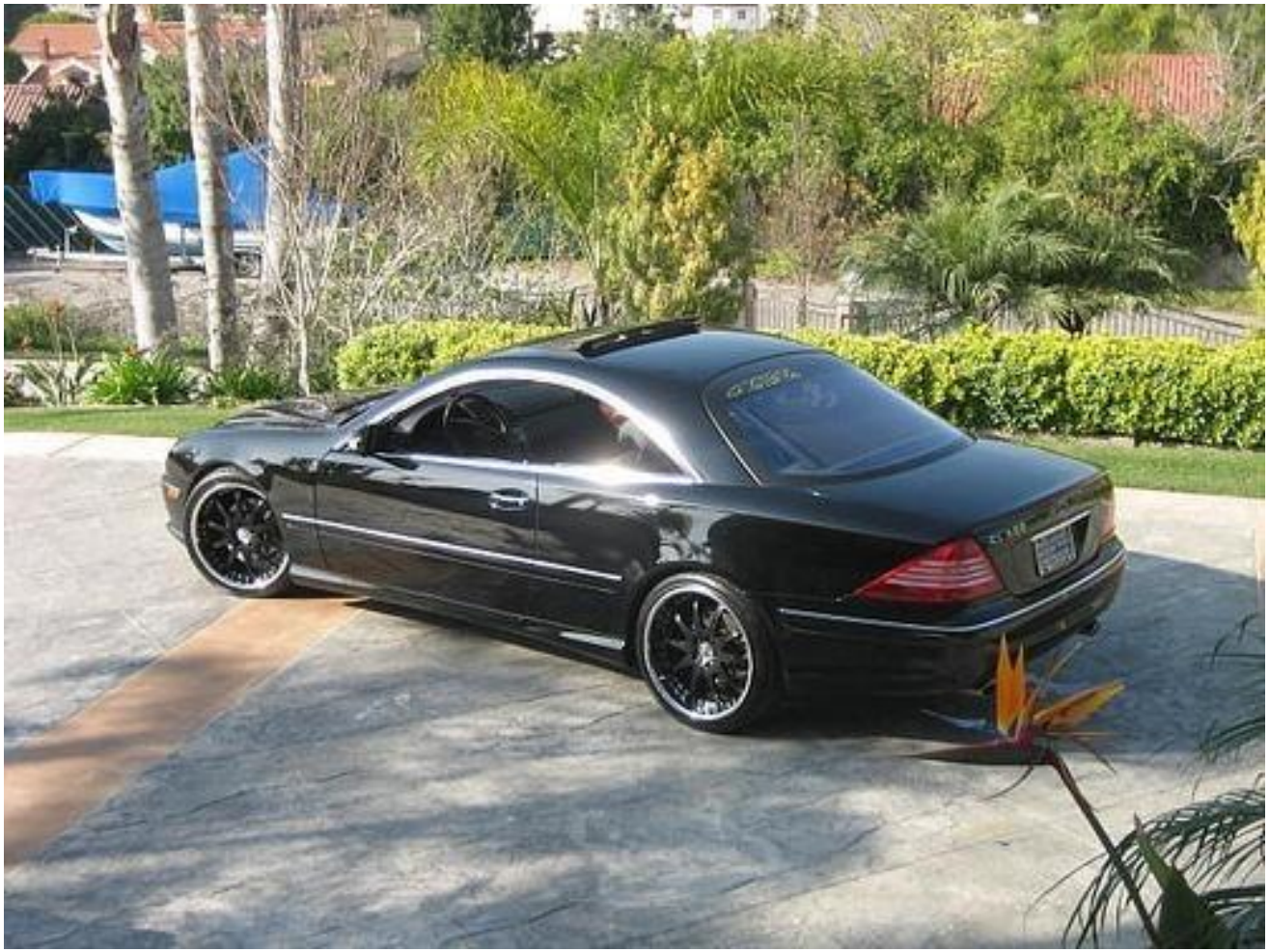
Object classification (easy)



Is there a car?

Source: Pascal VOC

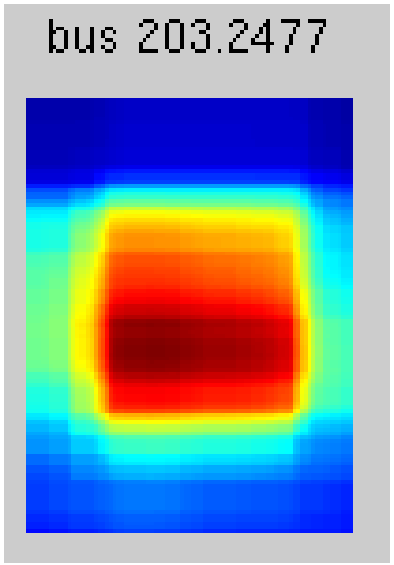
Object classification (harder)

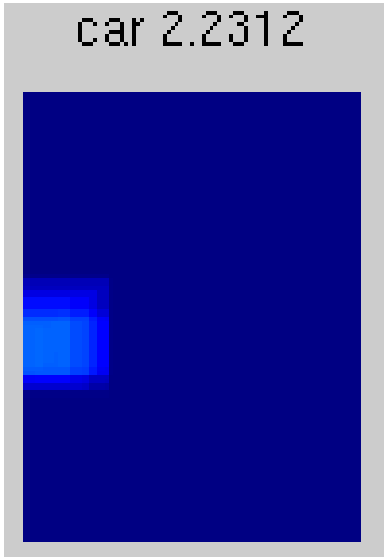


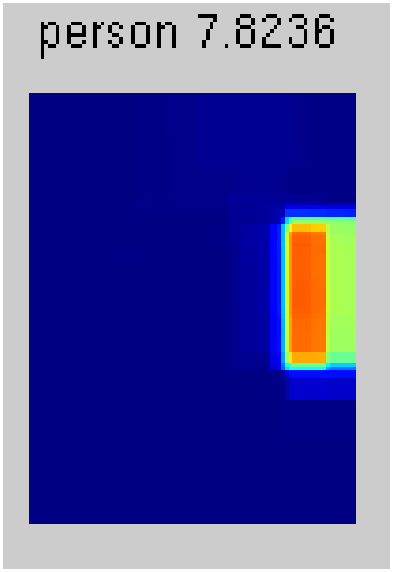
Is there a boat?

Source: Pascal VOC

Object classification





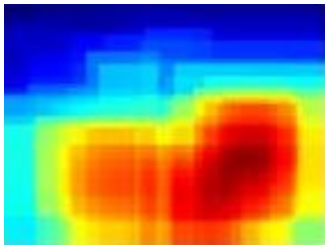


[Oquab, Bottou, Laptev and Sivic, CVPR 2014]

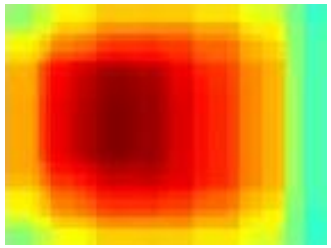
Goal

Recognize (name) objects present in an image depicting a **complex cluttered scene**

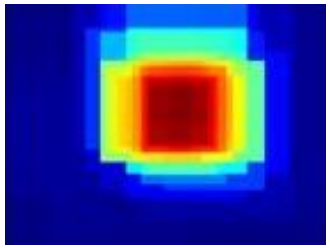
chai r



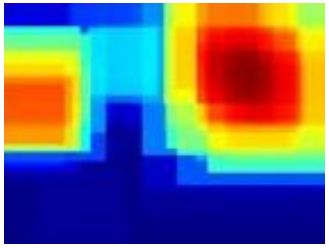
di ni ngt abl e



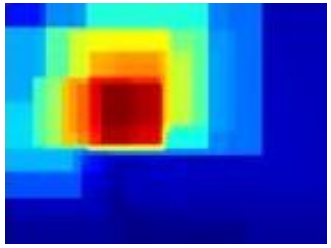
per son



pot t edpl ant



sof a



t vmoni t or

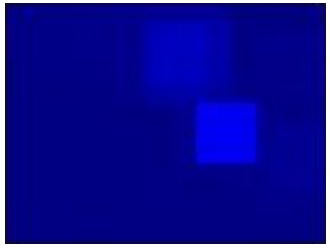
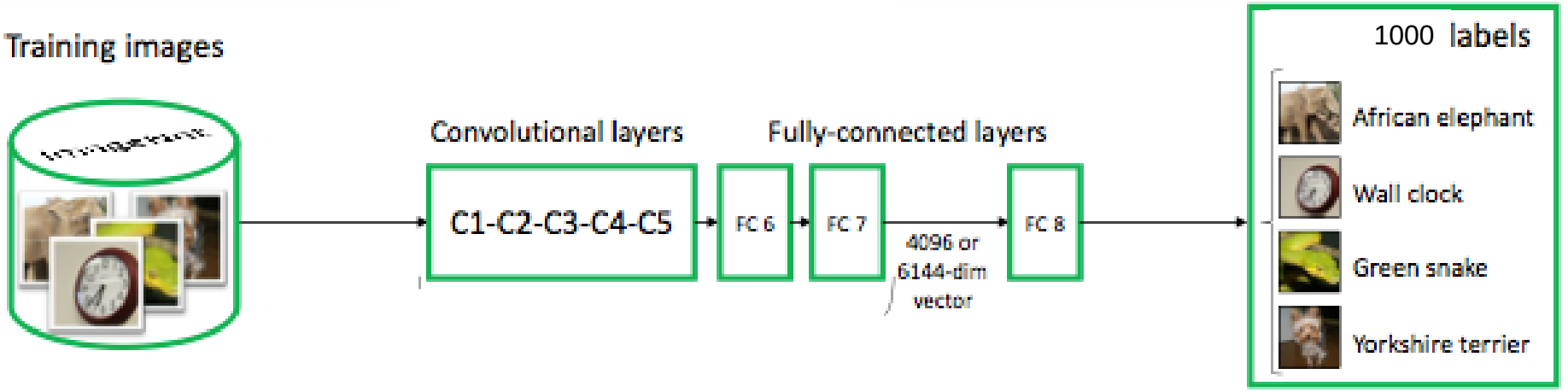


Image source:
Pascal VOC challenge

Background – Convolutional neural network of [Krizhevsky et al. 2012]



Input: ~1M **labelled** images (1000 images / 1000 classes)

Number of parameters: ~60 million --- image representation

Training time: ~1 week on one GPU

Learn parameters using stochastic gradient descent on cross-entropy error function.

Can we transfer learnt parameters to other tasks with limited training data?

Challenge

The dataset statistics between the source task (ImageNet) and the target task (Pascal VOC) can be very different.

- Type of objects and labels
- Object size, object location, scene clutter
- Object viewpoints, imaging conditions

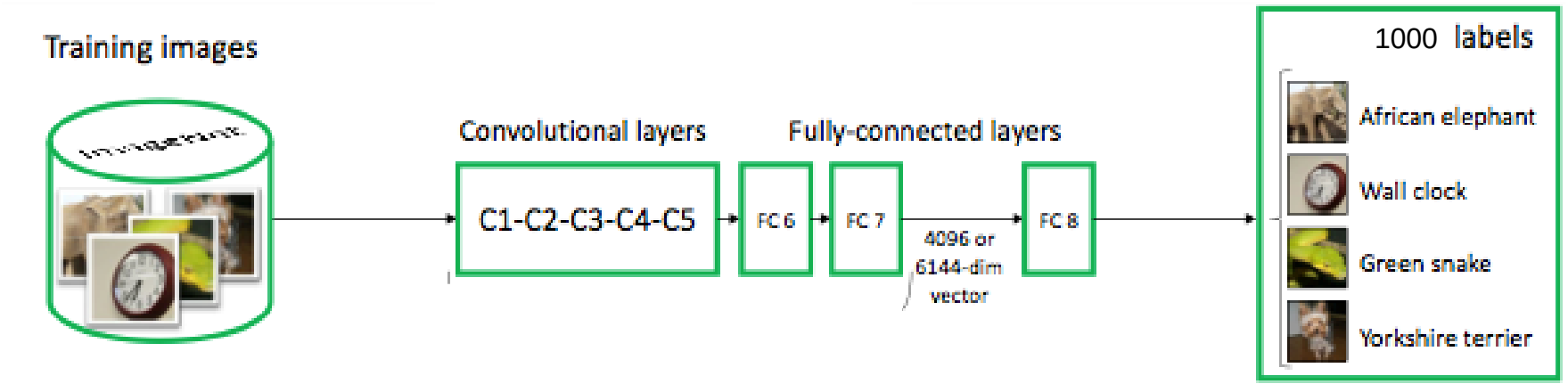
ImageNet

Maltese terrier

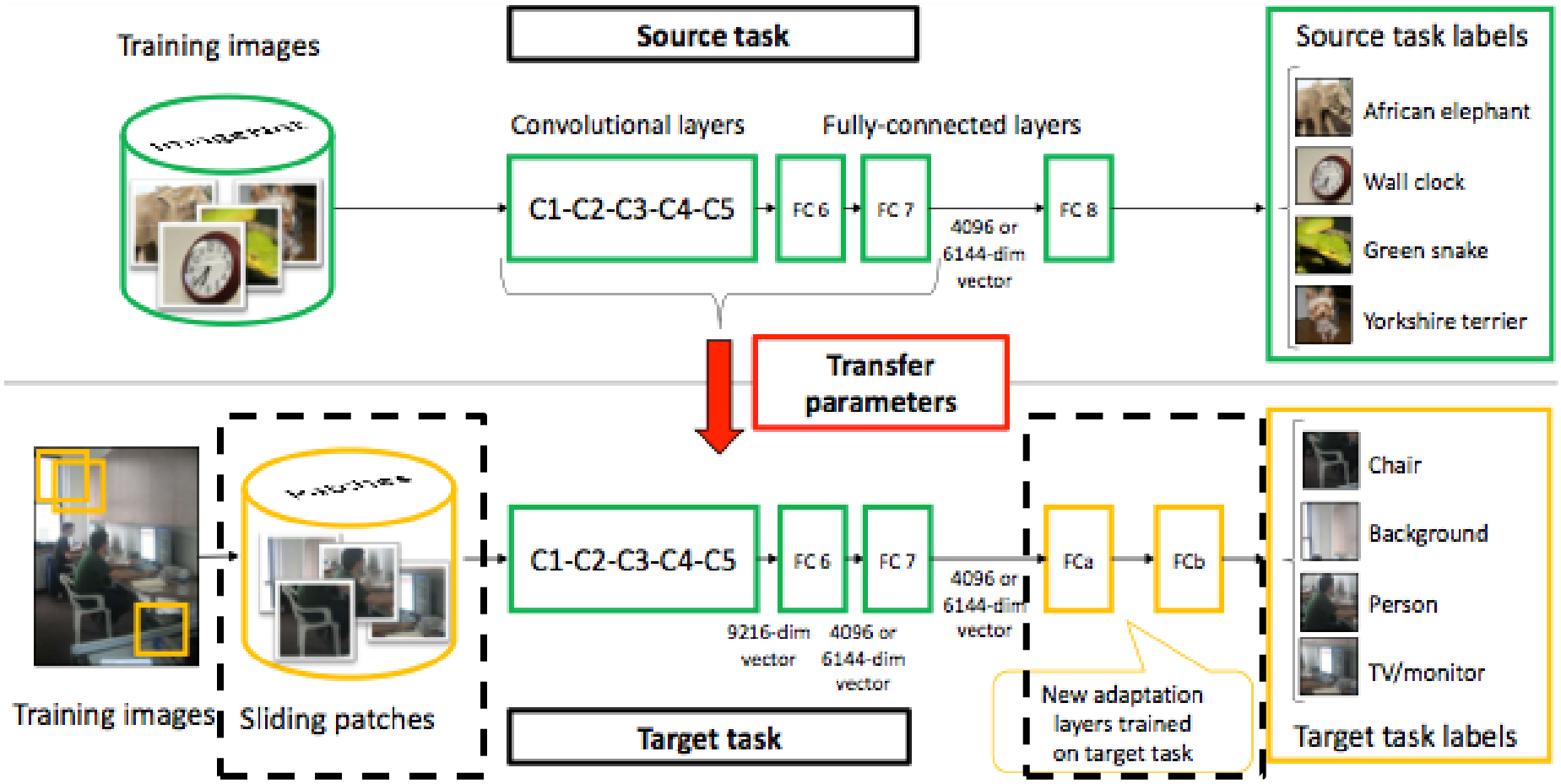
Pascal VOC

Dog

Approach


Training images
Convolutional layers
Fully-connected layers
C1-C2-C3-C4-C5
FC 6
FC 7
4096 or
6144-dim
vector
FC 8
1000 labels
African elephant
Wall clock
Green snake
Yorkshire terrier

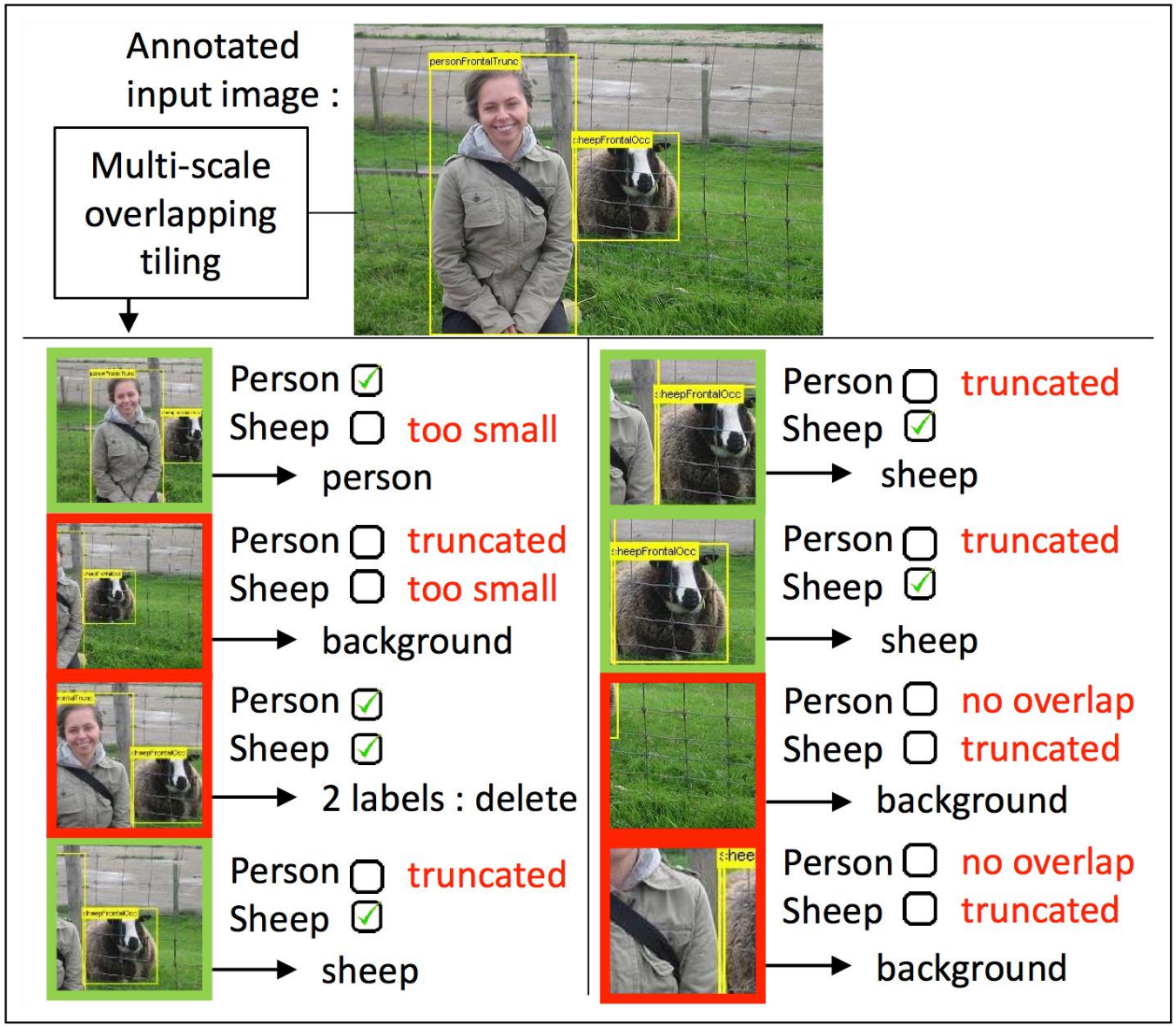
Approach [Oquab, Bottou, Laptev, Sivic, CVPR'14]



1. Design training/test procedure using sliding windows
2. Train adaptation layers to map labels

See also [Girshick et al.'13], [Donahue et al.'13], [Sermanet et al. '14], [Zeiler and Fergus '13]
Transfer learning workshop at ICCV'13, ImageNet workshop at ICCV'13

Approach – sliding window training / testing



Results – Pascal VOC 2007 object classification

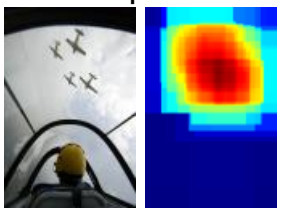
	plane	bike	bird	boat	btl	bus	car	cat	chair	cow
INRIA [32]	77.5	63.6	56.1	71.9	33.1	60.6	78.0	58.8	53.5	42.6
NUS-PSL [44]	82.5	79.6	64.8	73.4	54.2	75.0	77.5	79.2	46.2	62.7
PRE-1000C	88.5	81.5	87.9	82.0	47.5	75.5	90.1	87.2	61.6	75.7

table	dog	horse	moto	pers	plant	sheep	sofa	train	tv	mAP
54.9	45.8	77.5	64.0	85.9	36.3	44.7	50.6	79.2	53.2	59.4
41.4	74.6	85.0	76.8	91.1	53.9	61.0	67.5	83.6	70.6	70.5
67.3	85.5	83.5	80.0	95.6	60.8	76.8	58.0	90.4	77.9	77.7

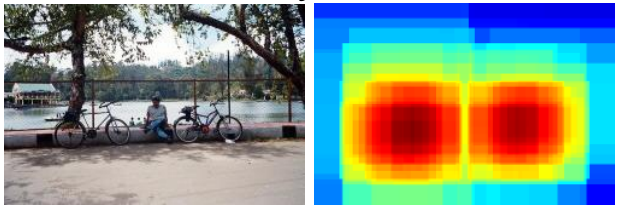
(a) Representative true positives

aer opl ane

aer opl ane

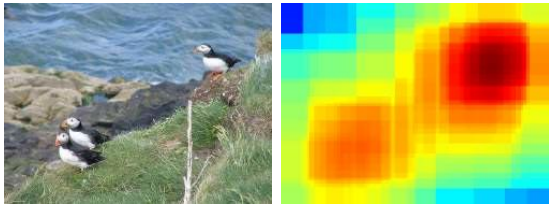


bi cycl e

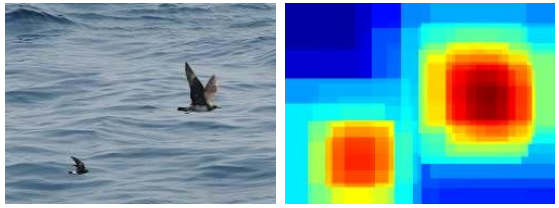


bi cycl e

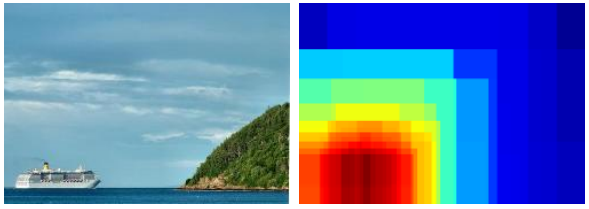
bi r d



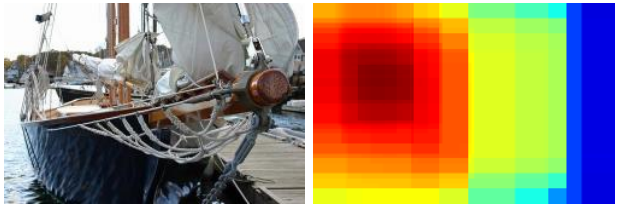
bi r d



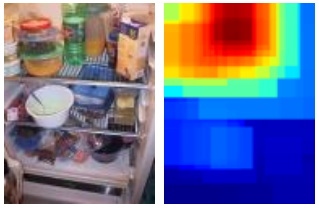
boat



boat



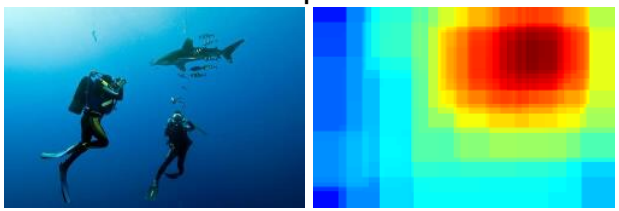
bot t l e



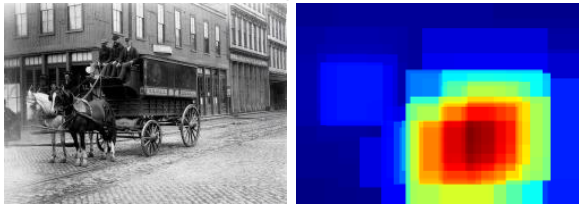
bot t l e

(b) Top ranking false positives

aer opl ane

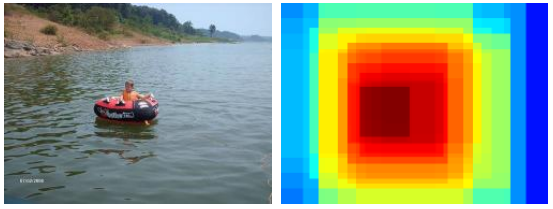


bi cycl e

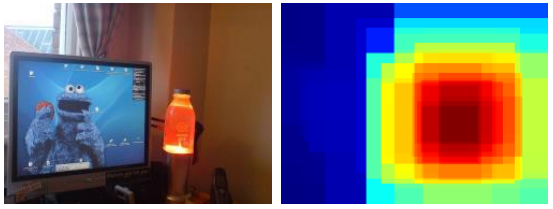


bi r d

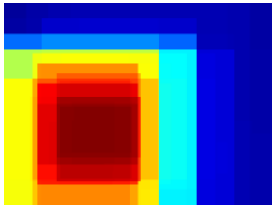
boat



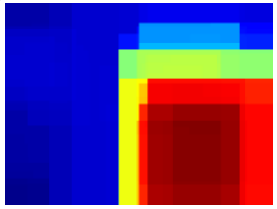
bot t l e



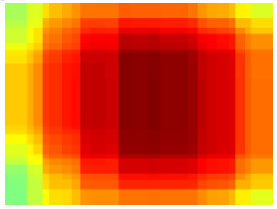
di ni ngt abl e

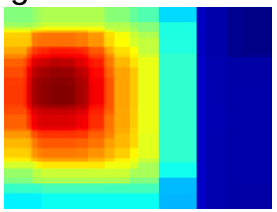
di ni ngt abl e

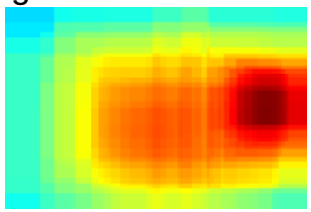
di ni ngt abl e

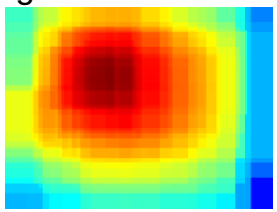
dog



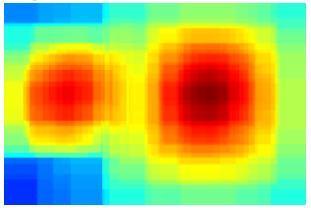
dog

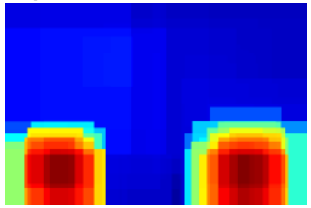
dog

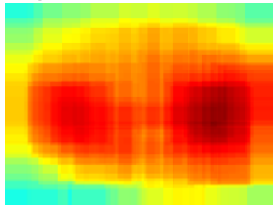
sheep

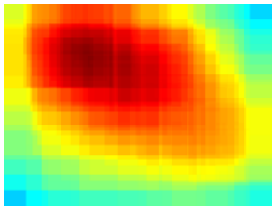
sheep

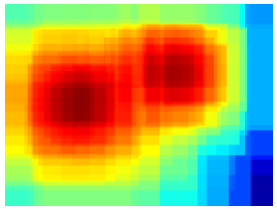
sheep

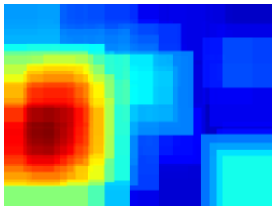
t vmoni t or

t vmoni t or

t vmoni t or

Results – Pascal VOC 2012 object classification

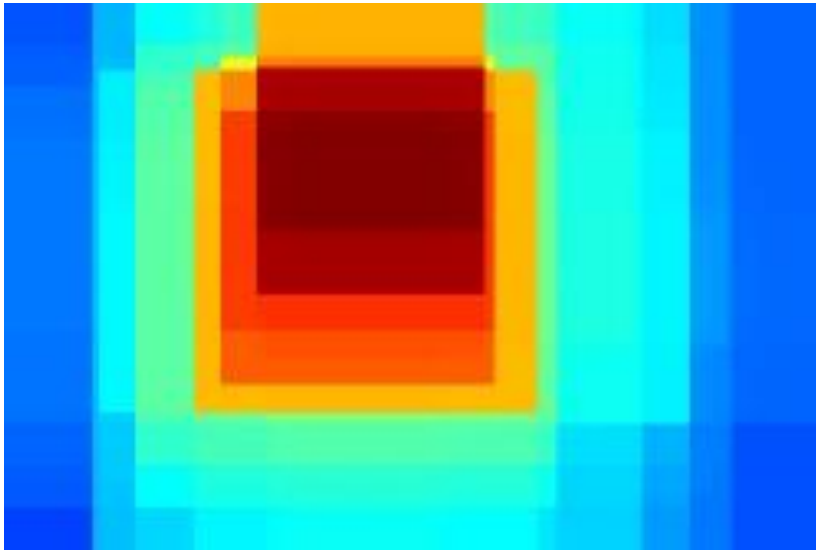
	plane	bike	bird	boat	btl	bus	car	cat	chair	cow	table	dog	horse	moto	pers	plant	sheep	sofa	train	tv	mAP
NUS-PSL [49]	97.3	84.2	80.8	85.3	60.8	89.9	86.8	89.3	75.4	77.8	75.1	83.0	87.5	90.1	95.0	57.8	79.2	73.4	94.5	80.7	82.2
NO PRETRAIN	85.2	75.0	69.4	66.2	48.8	82.1	79.5	79.8	62.4	61.9	49.8	75.9	71.4	82.7	93.1	59.1	69.7	49.3	80.0	76.7	70.9
PRE-1000C	93.5	78.4	*87.7*	80.9	57.3	85.0	81.6	*89.4*	66.9	73.8	62.0	*89.5*	83.2	87.6	*95.8*	*61.4*	79.0	54.3	88.0	78.3	78.7
PRE-1000R	93.2	77.9	83.8	80.0	55.8	82.7	79.0	84.3	66.2	71.7	59.5	83.4	81.4	84.8	95.2	59.8	74.9	52.9	83.8	75.7	76.3
PRE-1512	94.6	82.9	88.2	84.1	60.3	89.0	84.4	90.7	72.1	86.8	69.0	92.1	93.4	88.6	96.1	64.3	86.6	62.3	91.1	79.8	82.8

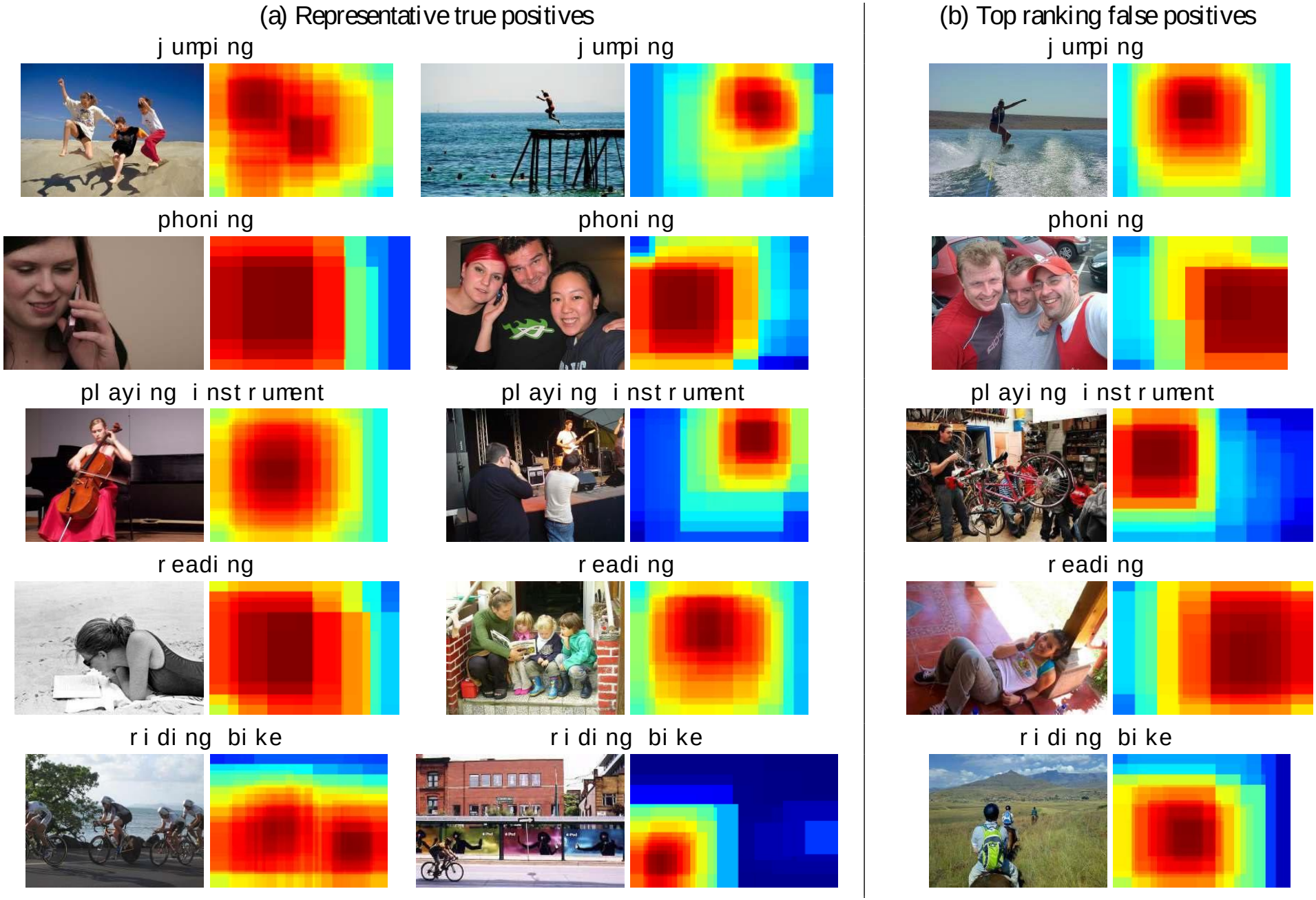
- Pre-training helps
- Random pre-training classes reduce performance (but not so much)
- Pre-training on classes related to the target task helps

Results – Pascal VOC 2012 action classification

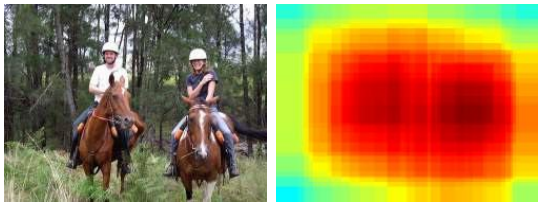
Action	jump	phon	instr	read	bike	horse	run	phot	comp	walk	mAP
STANFORD [1]	75.7	44.8	66.6	44.4	93.2	94.2	87.6	38.4	70.6	75.6	69.1
OXFORD [1]	77.0	50.4	65.3	39.5	94.1	95.9	87.7	42.7	68.6	74.5	69.6
NO PRETRAIN	43.2	30.6	50.2	25.0	76.8	80.7	75.2	22.2	37.9	55.6	49.7
PRE-1512	73.4	44.8	74.8	43.2	92.1	94.3	83.4	45.7	65.5	66.8	68.4
PRE-1512U	74.8	46.0	75.6	45.3	93.5	95.0	86.5	49.3	66.7	69.5	70.2

t aki ng phot o

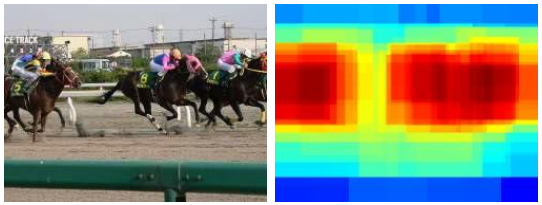



(a) Representative true positives
(b) Top ranking false positives
jumping
jumping
jumping
phoning
phoning
phoning
playing instrument
playing instrument
playing instrument
reading
reading
reading
riding bike
riding bike
riding bike

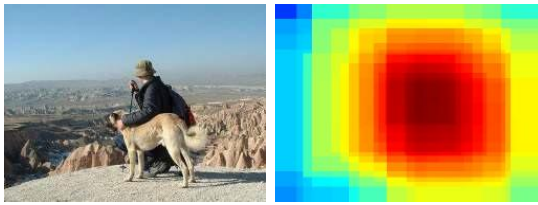
r i di ng hor se



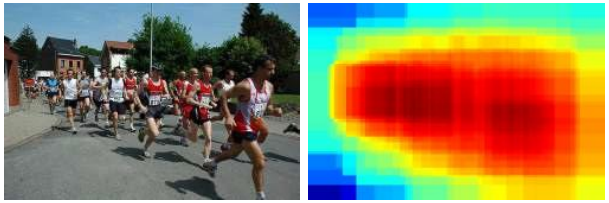
r i di ng hor se



r i di ng hor se

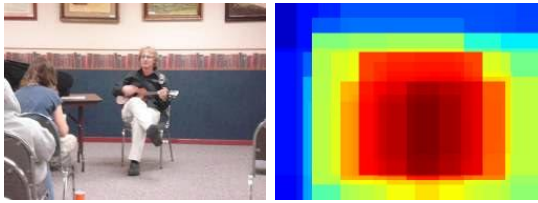


r unni ng



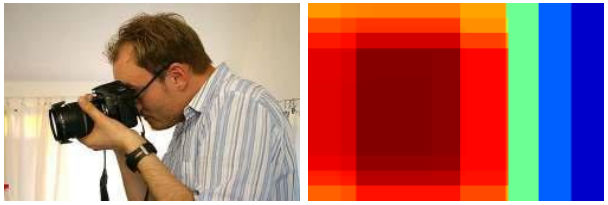
r unni ng

r unni ng

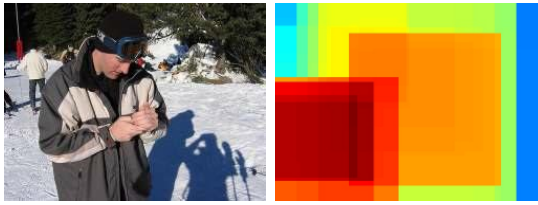


t aki ng phot o

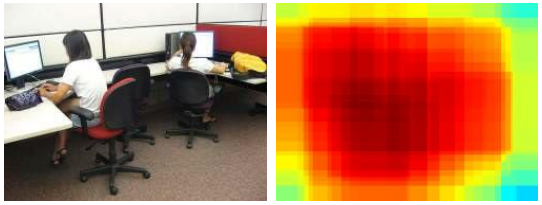
t aki ng phot o



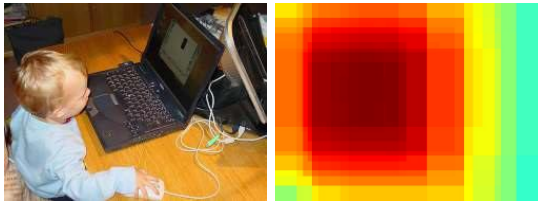
t aki ng phot o



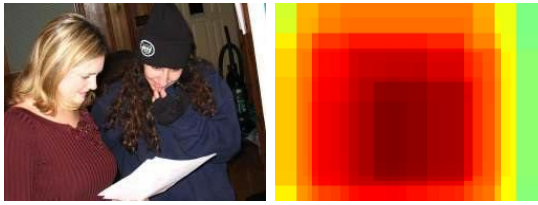
usi ng comput er



usi ng comput er



usi ng comput er



More results at:

http://www.di.ens.fr/willow/research/cnn/

Using CNN Features on Other Datasets

- Take model trained on, e.g., ImageNet 2012 training set
- Take outputs of 6^{th} or 7^{th} layer before or after nonlinearity
- Classify test set of new dataset
- Optional: fine-tune features and/or classifier on new dataset

See A3, part 2.

CNN features for other recognition tasks

[1] Caltech-101 (30 samples per class)

	$DeCAF_5$	$DeCAF_6$	$DeCAF_7$
LogReg	63.29 ± 6.6	84.30 ± 1.6	84.87 ± 0.6
LogReg with Dropout	-	86.08 ± 0.8	85.68 ± 0.6
SVM	77.12 ± 1.1	84.77 ± 1.2	83.24 ± 1.2
SVM with Dropout	-	$\mathbf{86.91 \pm 0.7}$	85.51 ± 0.9
Yang et al. (2009)		84.3	
Jarrett et al. (2009)		65.5	

[1] SUN 397 dataset (DeCAF)

	$DeCAF_6$	$DeCAF_7$
LogReg	$\mathbf{40.94 \pm 0.3}$	40.84 ± 0.3
SVM	39.36 ± 0.3	40.66 ± 0.3
Xiao et al. (2010)	38.0	

[1] Caltech-UCSD Birds (DeCAF)

Method	Accuracy
$DeCAF_6$	58.75
DPD + $DeCAF_6$	**64.96**
DPD (Zhang et al., 2013)	50.98
POOF (Berg & Belhumeur, 2013)	56.78

[2] MIT-67 Indoor Scenes dataset (OverFeat)

Method	mean Accuracy
ROI + Gist[36]	26.05
DPM[30]	30.40
Object Bank[25]	37.60
RBow[31]	37.93
BoP[22]	46.10
miSVM[26]	46.40
D-Parts[40]	51.40
IFV[22]	60.77
MLrep[11]	**64.03**
CNN-SVM	58.44

[1] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition, arXiv preprint, 2014

[2] A. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, CNN Features off-the-shelf: an Astounding Baseline for Recognition, arXiv preprint, 2014

Slide: R. Fergus / S. Lazebnik

Caltech 101 & 256

http://www.vision.caltech.edu/Image_Datasets/Caltech101/
http://www.vision.caltech.edu/Image_Datasets/Caltech256/

Fei-Fei, Fergus, Perona, 2004

Griffin, Holub, Perona, 2007

CNN features for other recognition tasks

[1] Caltech-101 (30 samples per class)

	$DeCAF_5$	$DeCAF_6$	$DeCAF_7$
LogReg	63.29 ± 6.6	84.30 ± 1.6	84.87 ± 0.6
LogReg with Dropout	-	86.08 ± 0.8	85.68 ± 0.6
SVM	77.12 ± 1.1	84.77 ± 1.2	83.24 ± 1.2
SVM with Dropout	-	$\mathbf{86.91 \pm 0.7}$	85.51 ± 0.9
Yang et al. (2009)		84.3	
Jarrett et al. (2009)		65.5	

[1] SUN 397 dataset (DeCAF)

	$DeCAF_6$	$DeCAF_7$
LogReg	$\mathbf{40.94 \pm 0.3}$	40.84 ± 0.3
SVM	39.36 ± 0.3	40.66 ± 0.3
Xiao et al. (2010)	38.0	

[1] Caltech-UCSD Birds (DeCAF)

Method	Accuracy
$DeCAF_6$	58.75
DPD + $DeCAF_6$	**64.96**
DPD (Zhang et al., 2013)	50.98
POOF (Berg & Belhumeur, 2013)	56.78

[2] MIT-67 Indoor Scenes dataset (OverFeat)

Method	mean Accuracy
ROI + Gist[36]	26.05
DPM[30]	30.40
Object Bank[25]	37.60
RBow[31]	37.93
BoP[22]	46.10
miSVM[26]	46.40
D-Parts[40]	51.40
IFV[22]	60.77
MLrep[11]	**64.03**
CNN-SVM	58.44

[1] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition, arXiv preprint, 2014

[2] A. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, CNN Features off-the-shelf: an Astounding Baseline for Recognition, arXiv preprint, 2014

Slide: R. Fergus / S. Lazebnik

SUN dataset

~900 scene categories (~400 well-sampled), 130K images

J. Xiao, J. Hays, K. Ehinger, A. Oliva, and A. Torralba, "SUN Database: Large-scale Scene Recognition from Abbey to Zoo," CVPR 2010

http://groups.csail.mit.edu/vision/SUN/

CNN features for other recognition tasks

[1] Caltech-101 (30 samples per class)

	$DeCAF_5$	$DeCAF_6$	$DeCAF_7$
LogReg	63.29 ± 6.6	84.30 ± 1.6	84.87 ± 0.6
LogReg with Dropout	-	86.08 ± 0.8	85.68 ± 0.6
SVM	77.12 ± 1.1	84.77 ± 1.2	83.24 ± 1.2
SVM with Dropout	-	$\mathbf{86.91 \pm 0.7}$	85.51 ± 0.9
Yang et al. (2009)		84.3	
Jarrett et al. (2009)		65.5	

[1] SUN 397 dataset (DeCAF)

	$DeCAF_6$	$DeCAF_7$
LogReg	$\mathbf{40.94 \pm 0.3}$	40.84 ± 0.3
SVM	39.36 ± 0.3	40.66 ± 0.3
Xiao et al. (2010)	38.0	

[1] Caltech-UCSD Birds (DeCAF)

Method	Accuracy
$DeCAF_6$	58.75
DPD + $DeCAF_6$	**64.96**
DPD (Zhang et al., 2013)	50.98
POOF (Berg & Belhumeur, 2013)	56.78

[2] MIT-67 Indoor Scenes dataset (OverFeat)

Method	mean Accuracy
ROI + Gist[36]	26.05
DPM[30]	30.40
Object Bank[25]	37.60
RBow[31]	37.93
BoP[22]	46.10
miSVM[26]	46.40
D-Parts[40]	51.40
IFV[22]	60.77
MLrep[11]	**64.03**
CNN-SVM	58.44

[1] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition, arXiv preprint, 2014

[2] A. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, CNN Features off-the-shelf: an Astounding Baseline for Recognition, arXiv preprint, 2014

Slide: R. Fergus / S. Lazebnik

Caltech-UCSD Birds

more images and annotations, see Caltech-UCSD Birds-200-2011

CNN features for other recognition tasks

[1] Caltech-101 (30 samples per class)

	$DeCAF_5$	$DeCAF_6$	$DeCAF_7$
LogReg	63.29 ± 6.6	84.30 ± 1.6	84.87 ± 0.6
LogReg with Dropout	-	86.08 ± 0.8	85.68 ± 0.6
SVM	77.12 ± 1.1	84.77 ± 1.2	83.24 ± 1.2
SVM with Dropout	-	$\mathbf{86.91 \pm 0.7}$	85.51 ± 0.9
Yang et al. (2009)		84.3	
Jarrett et al. (2009)		65.5	

[1] SUN 397 dataset (DeCAF)

	$DeCAF_6$	$DeCAF_7$
LogReg	$\mathbf{40.94 \pm 0.3}$	40.84 ± 0.3
SVM	39.36 ± 0.3	40.66 ± 0.3
Xiao et al. (2010)	38.0	

[1] Caltech-UCSD Birds (DeCAF)

Method	Accuracy
$DeCAF_6$	58.75
DPD + $DeCAF_6$	**64.96**
DPD (Zhang et al., 2013)	50.98
POOF (Berg & Belhumeur, 2013)	56.78

[2] MIT-67 Indoor Scenes dataset (OverFeat)

Method	mean Accuracy
ROI + Gist[36]	26.05
DPM[30]	30.40
Object Bank[25]	37.60
RBow[31]	37.93
BoP[22]	46.10
miSVM[26]	46.40
D-Parts[40]	51.40
IFV[22]	60.77
MLrep[11]	**64.03**
CNN-SVM	58.44

[1] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition, arXiv preprint, 2014

[2] A. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, CNN Features off-the-shelf: an Astounding Baseline for Recognition, arXiv preprint, 2014

Slide: R. Fergus / S. Lazebnik

MIT 67 indoor scene dataset

Indoor Scene Recognition

Indoor scene recognition is a challenging open problem in high level vision. Most scene recognition models that work well for outdoor scenes perform poorly in the indoor domain. The main difficulty is that while some indoor scenes (e.g. corridors) can be well characterized by global spatial properties, others (e.g., bookstores) are better characterized by the objects they contain. More generally, to address the indoor scenes recognition problem we need a model that can exploit local and global discriminative information.

Database

The database contains 67 Indoor categories, and a total of 15620 images. The number of images varies across categories, bu there are at least 100 images per category. All images are in jpg format. The images provided here are for research purposes only.

Download (tar file, 2.4 Gbytes)

CNN features for other recognition tasks

[1] Caltech-101 (30 samples per class)

	$DeCAF_5$	$DeCAF_6$	$DeCAF_7$
LogReg	63.29 ± 6.6	84.30 ± 1.6	84.87 ± 0.6
LogReg with Dropout	-	86.08 ± 0.8	85.68 ± 0.6
SVM	77.12 ± 1.1	84.77 ± 1.2	83.24 ± 1.2
SVM with Dropout	-	$\mathbf{86.91 \pm 0.7}$	85.51 ± 0.9
Yang et al. (2009)		84.3	
Jarrett et al. (2009)		65.5	

[1] SUN 397 dataset (DeCAF)

	$DeCAF_6$	$DeCAF_7$
LogReg	$\mathbf{40.94 \pm 0.3}$	40.84 ± 0.3
SVM	39.36 ± 0.3	40.66 ± 0.3
Xiao et al. (2010)	38.0	

[1] Caltech-UCSD Birds (DeCAF)

Method	Accuracy
$DeCAF_6$	58.75
DPD + $DeCAF_6$	**64.96**
DPD (Zhang et al., 2013)	50.98
POOF (Berg & Belhumeur, 2013)	56.78

[2] MIT-67 Indoor Scenes dataset (OverFeat)

Method	mean Accuracy
ROI + Gist[36]	26.05
DPM[30]	30.40
Object Bank[25]	37.60
RBow[31]	37.93
BoP[22]	46.10
miSVM[26]	46.40
D-Parts[40]	51.40
IFV[22]	60.77
MLrep[11]	**64.03**
CNN-SVM	58.44

[1] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition, arXiv preprint, 2014

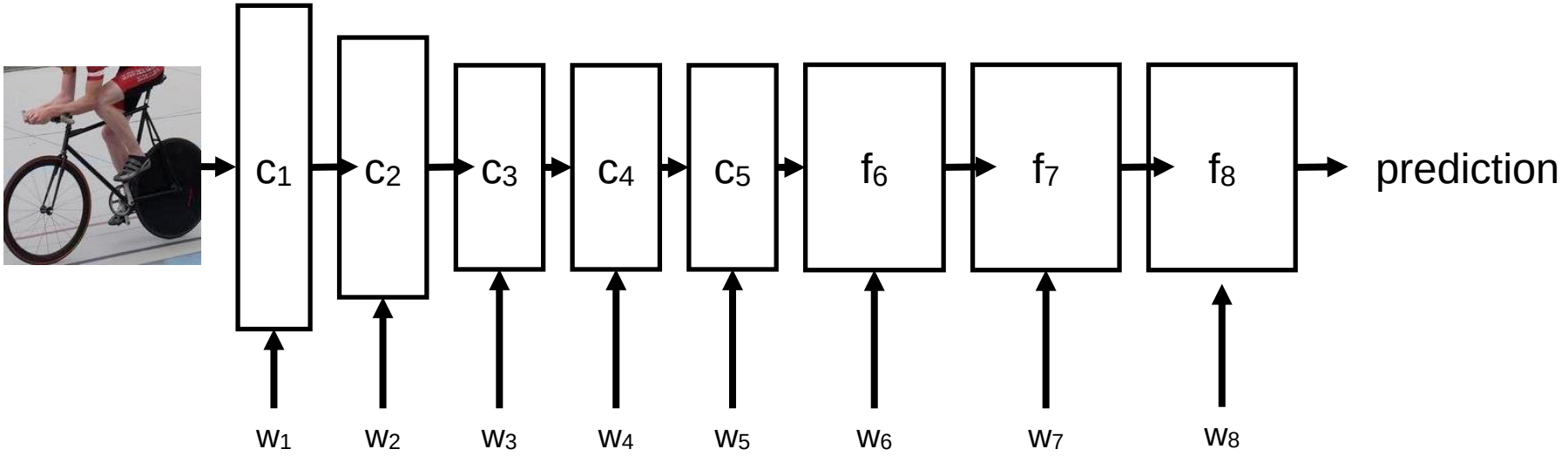
[2] A. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, CNN Features off-the-shelf: an Astounding Baseline for Recognition, arXiv preprint, 2014

Slide: R. Fergus / S. Lazebnik

Outline

1. Convolutional neural networks (CNNs)
2. Understanding and visualizing CNN representations
3. Transferring learnt representation to other tasks
4. Typical CNN architectures for image classification
5. Beyond classification

A CNN for image classification

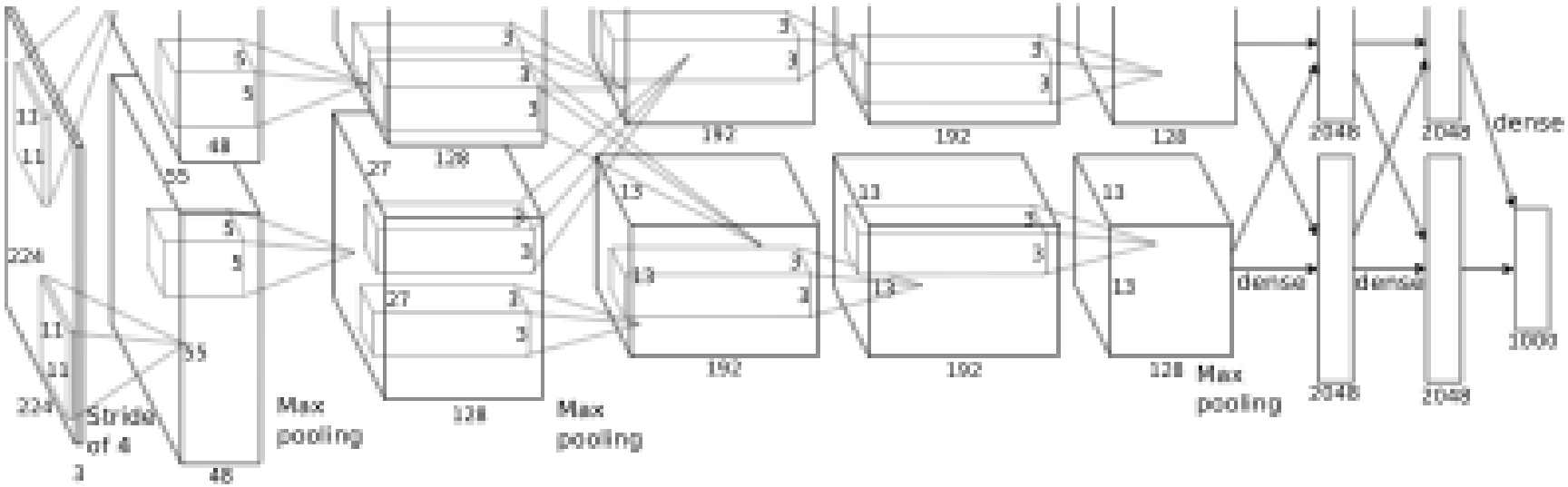


Recall: the goal of this model is to map an input image to a class prediction.

Slide: A. Vedaldi

Recall: The AlexNet model

A breakthrough in image understanding



[AlexNet by Krizhevsky et al. 2012]

Each large block represents a data tensor

Each smaller block represents a filter

The filter size and stride are shown

The number of filters can be deduced from the number of feature channels

There are two parallel streams in this network (for efficiency reasons)

Slide: A. Vedaldi

How deep is deep enough?

AlexNet (2012)

5 convolutional layers

3 fully-connected layers

Slide: A. Vedaldi

How deep is deep enough?

AlexNet (2012)

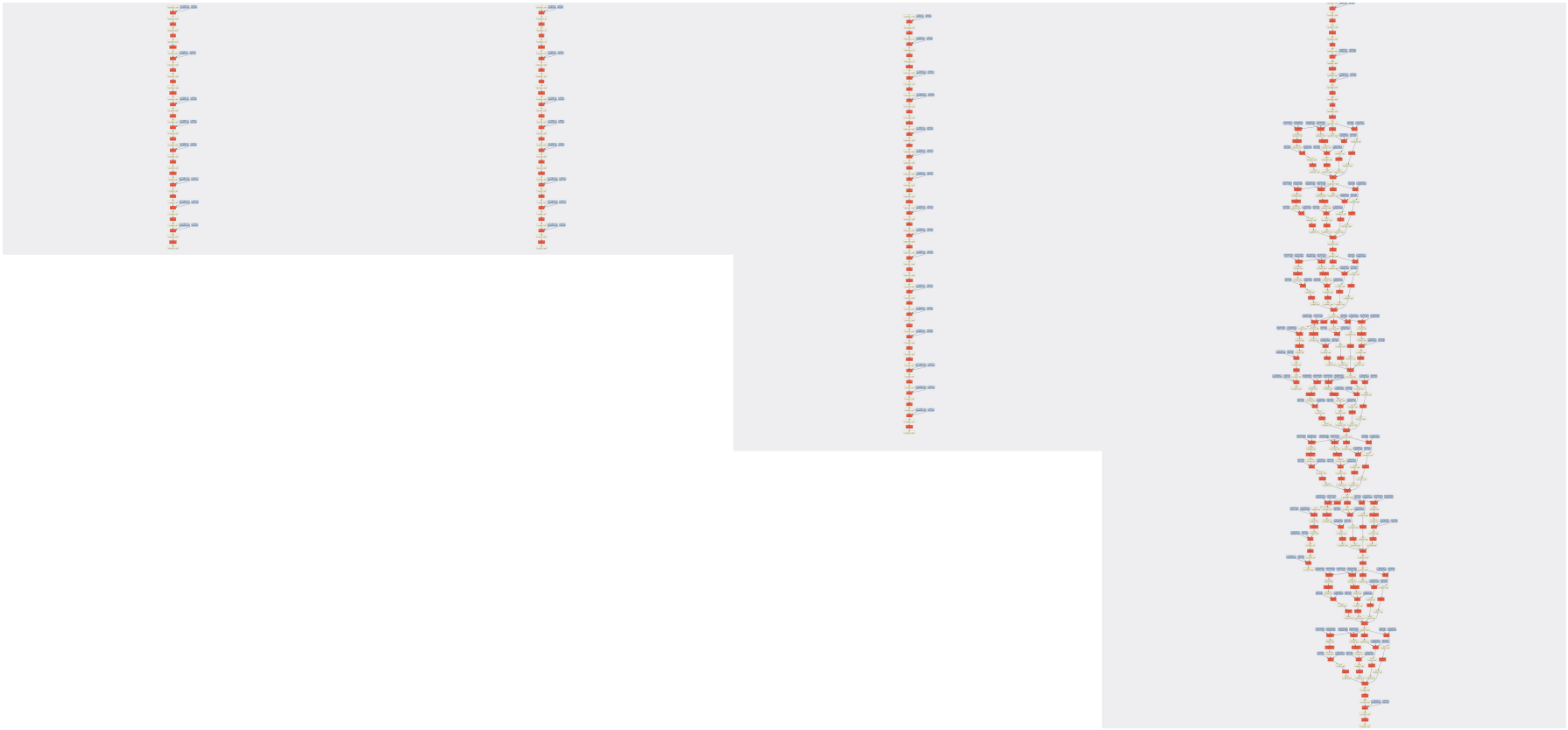
VGG-M (2013)

VGG-VD-16 (2014)

Slide: A. Vedaldi

How deep is deep enough?

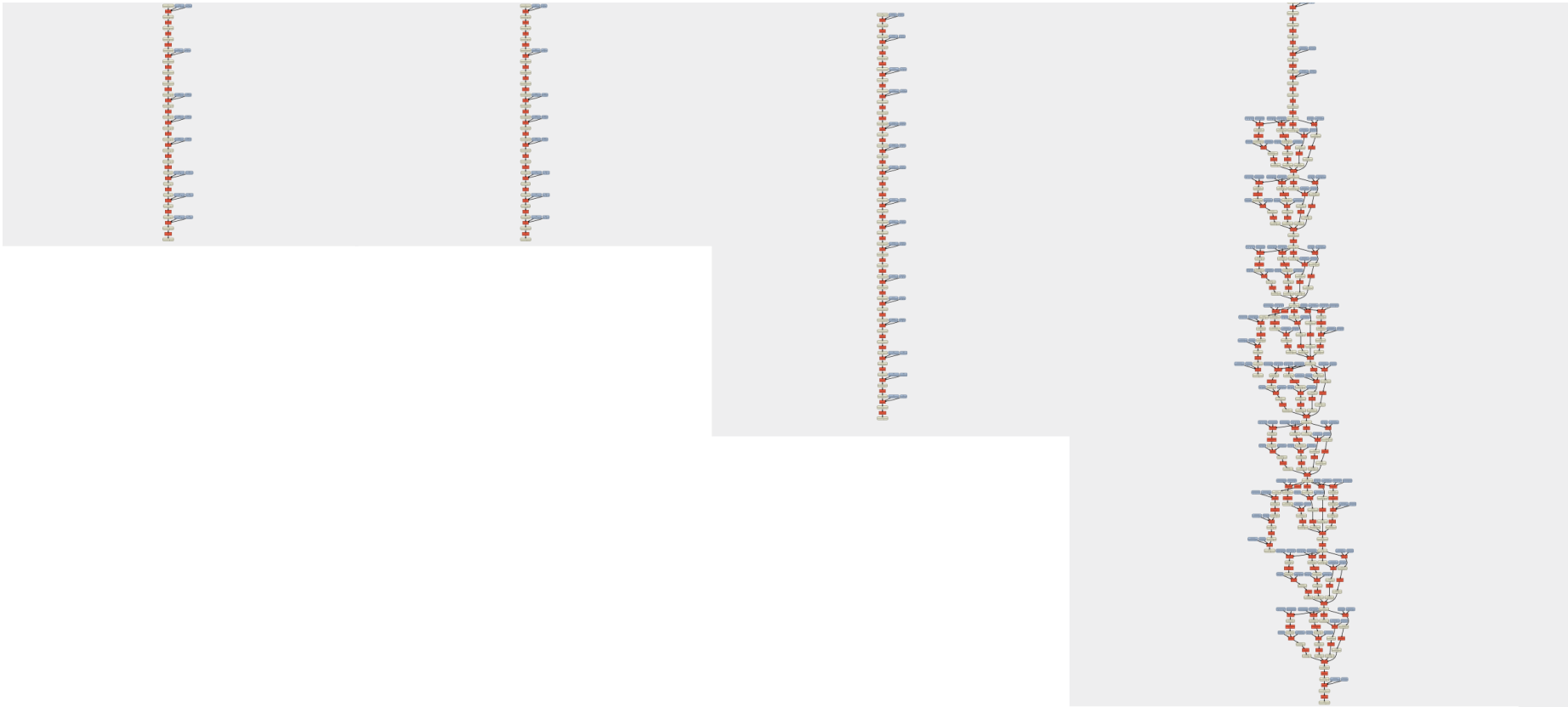
AlexNet (2012) VGG-M (2013) VGG-VD-16 (2014) GoogLeNet (2014)



Slide: A. Vedaldi

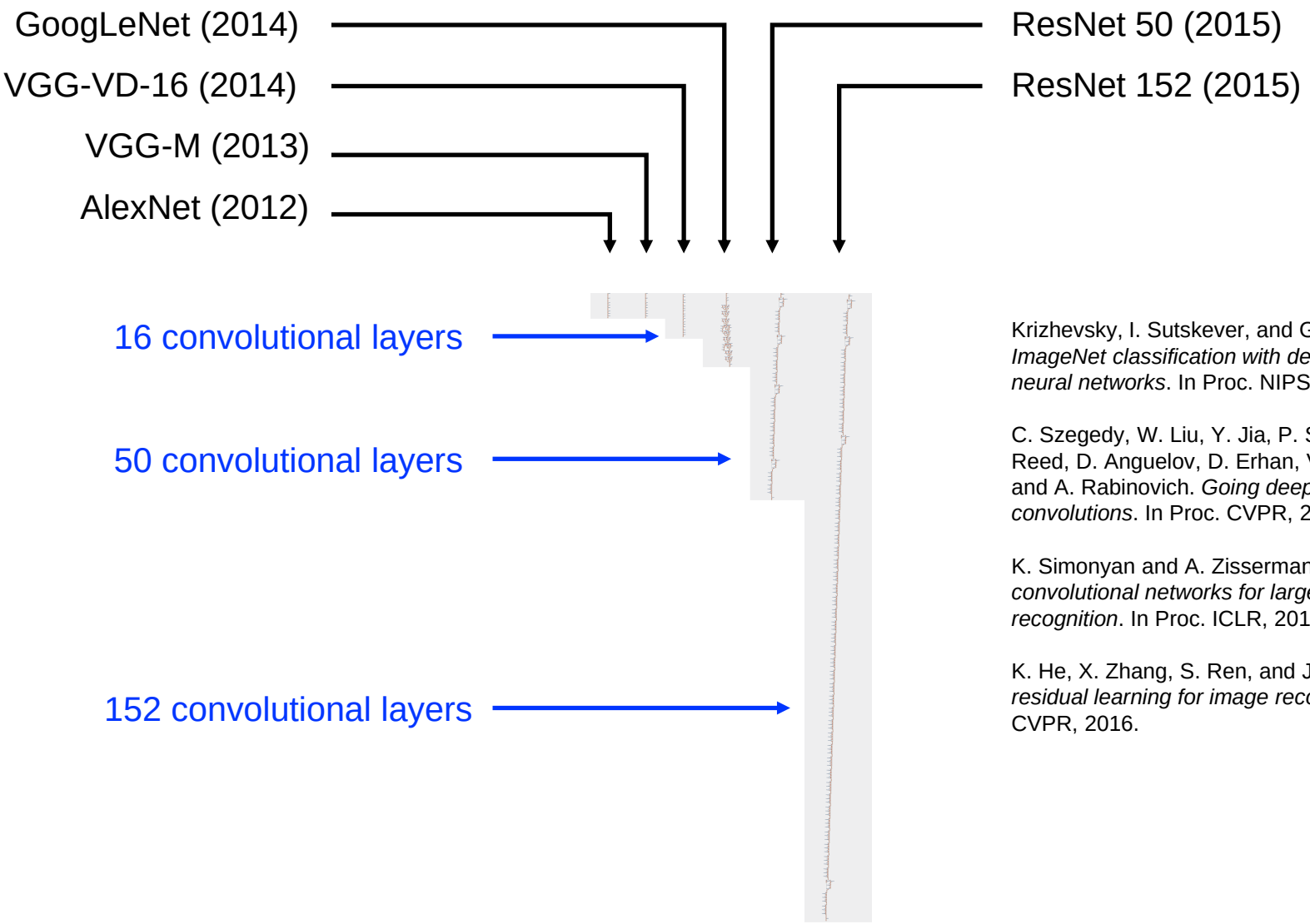
How deep is deep enough?

AlexNet (2012) VGG-M (2013) VGG-VD-16 (2014) GoogLeNet (2014)



Slide: A. Vedaldi

How deep is deep enough?



Krizhevsky, I. Sutskever, and G. E. Hinton. *ImageNet classification with deep convolutional neural networks*. In Proc. NIPS, 2012.

C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. *Going deeper with convolutions*. In Proc. CVPR, 2015.

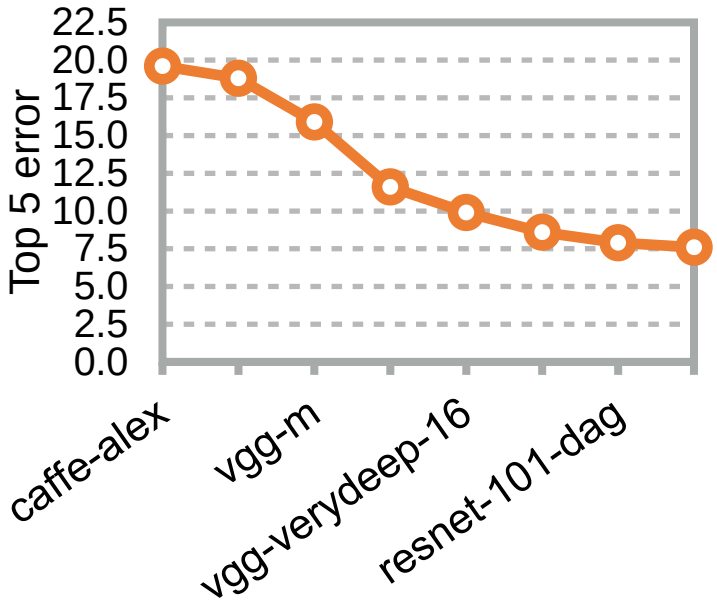
K. Simonyan and A. Zisserman. *Very deep convolutional networks for large-scale image recognition*. In Proc. ICLR, 2015.

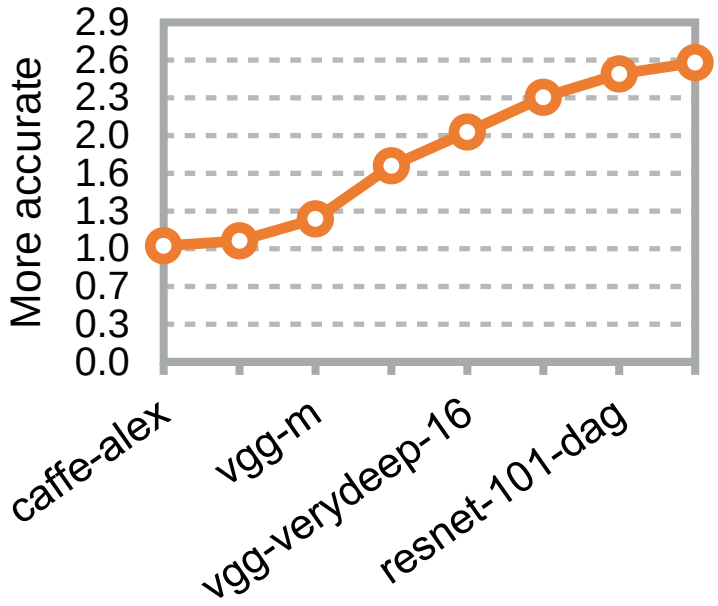
K. He, X. Zhang, S. Ren, and J. Sun. *Deep residual learning for image recognition*. In Proc. CVPR, 2016.

Slide: A. Vedaldi

Accuracy

3 × more accurate in 3 years

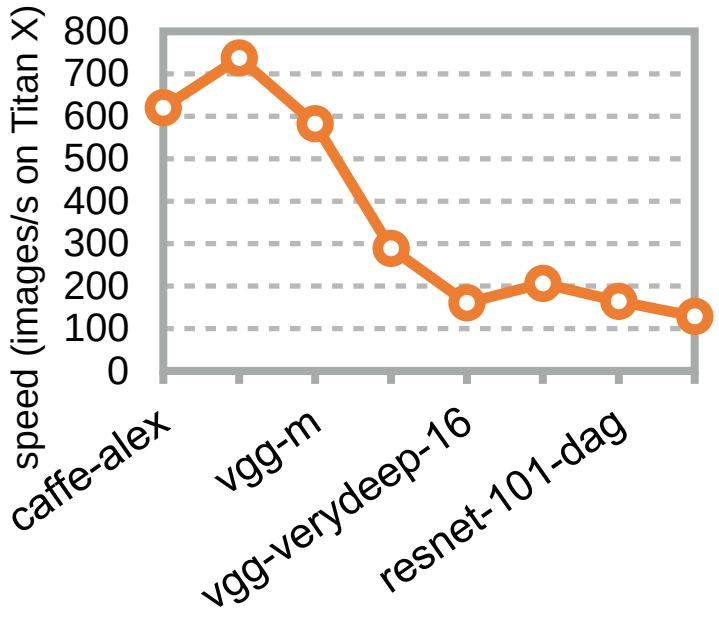


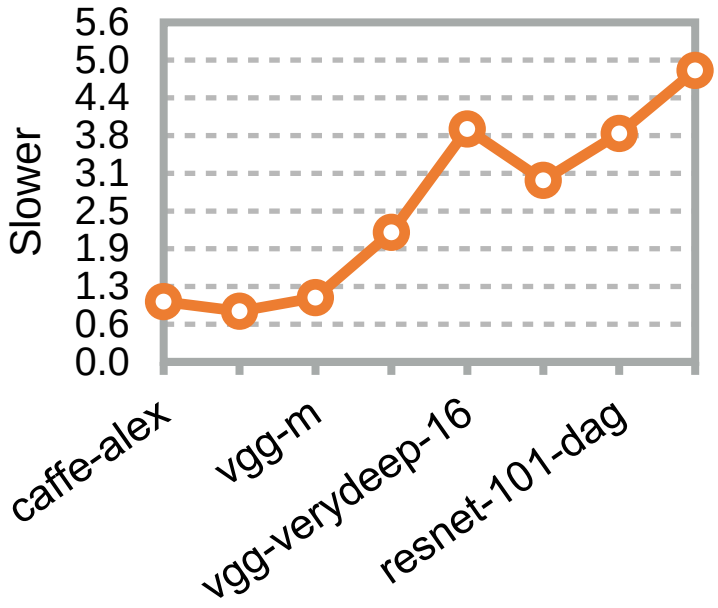


Slide: A. Vedaldi

Speed

5 × slower





Remark: *101 ResNet* layers same size/speed as *16 VGG-VD* layers

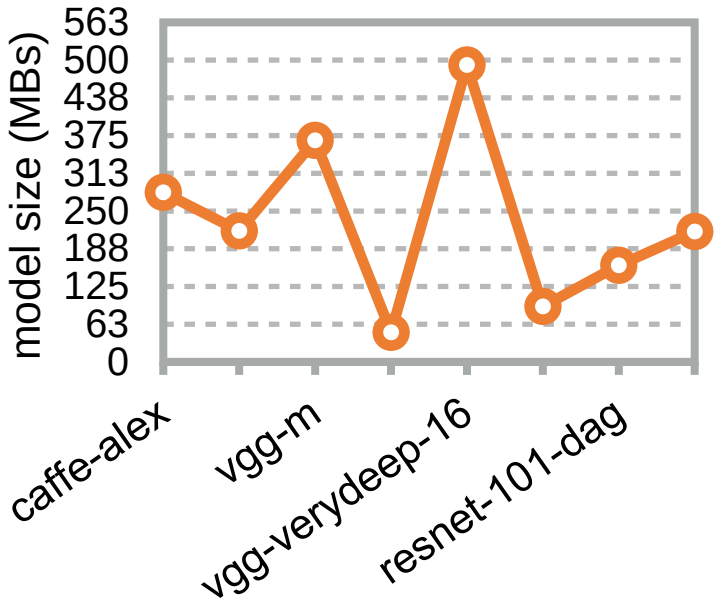
Reason: far fewer feature channels (quadratic speed/space gain)

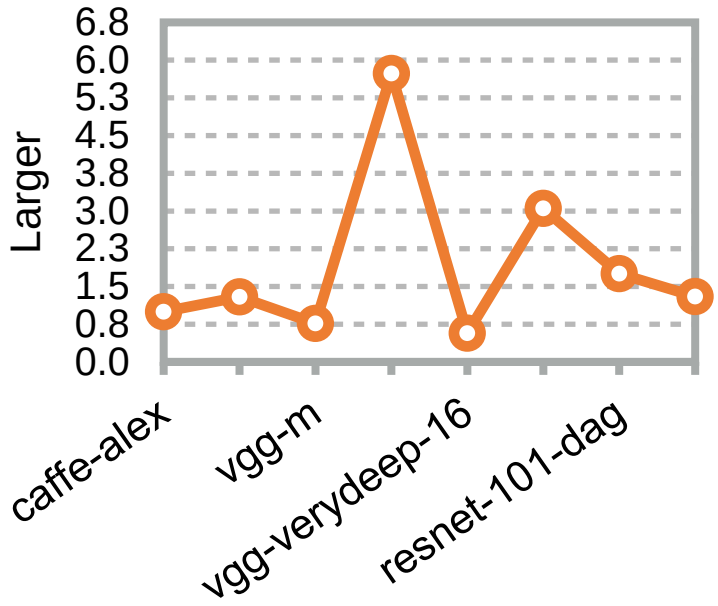
Moral: optimize your architecture

Slide: A. Vedaldi

Model size

Num. of parameters is about the same





Remark: *101 ResNet* layers same size/speed as *16 VGG-VD* layers

Reason: far fewer feature channels (quadratic speed/space gain)

Moral: optimize your architecture

CNN architectures – notes and details

Increased depth of recent architectures

Number of parameters matter

(How to count parameters?)

Power of small filters

ResNet architecture

Very deep networks

VGG Net – excellent performance on ILSVRC'14

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

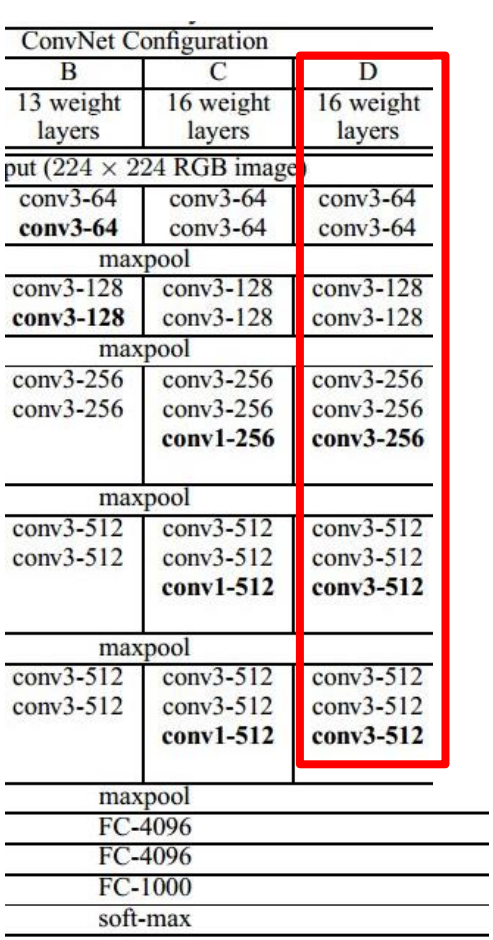
[Simonyan&Zisserman 2014]

Recent trends – very deep networks

VGG Net – excellent performance on ILSVRC'14

(not counting biases)

INPUT: [224x224x3] memory: 224*224*3=150K params: 0
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*3)*64 = 1,728
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*64)*64 = 36,864
POOL2: [112x112x64] memory: 112*112*64=800K params: 0
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*64)*128 = 73,728
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*128)*128 = 147,456
POOL2: [56x56x128] memory: 56*56*128=400K params: 0
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*128)*256 = 294,912
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
POOL2: [28x28x256] memory: 28*28*256=200K params: 0
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*256)*512 = 1,179,648
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
POOL2: [14x14x512] memory: 14*14*512=100K params: 0
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
POOL2: [7x7x512] memory: 7*7*512=25K params: 0
FC: [1x1x4096] memory: 4096 params: 7*7*512*4096 = 102,760,448
FC: [1x1x4096] memory: 4096 params: 4096*4096 = 16,777,216
FC: [1x1x1000] memory: 1000 params: 4096*1000 = 4,096,000

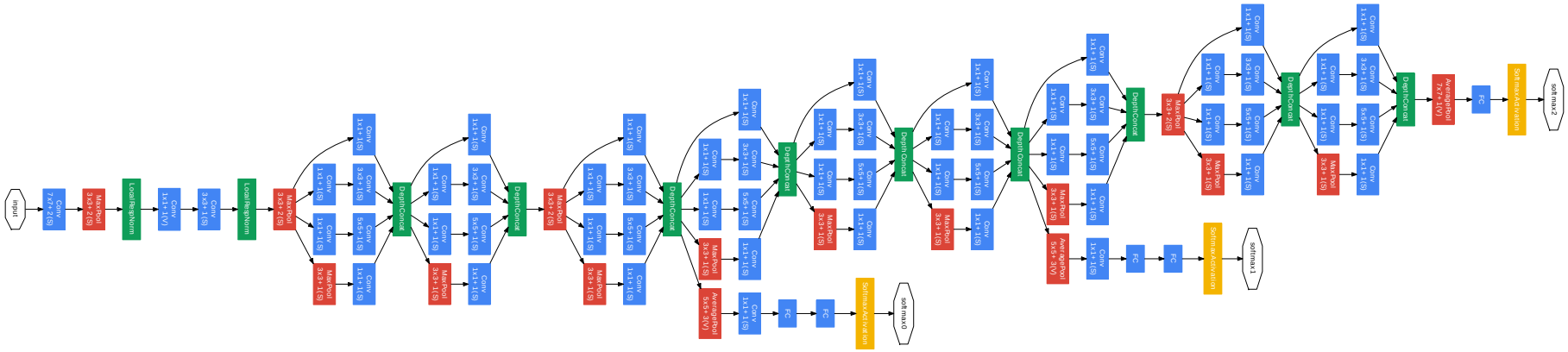


ConvNet Configuration		
B	C	D
13 weight layers	16 weight layers	16 weight layers
put (224 × 224 RGB image)		
conv3-64	conv3-64	conv3-64
conv3-64	conv3-64	conv3-64
maxpool		
conv3-128	conv3-128	conv3-128
conv3-128	conv3-128	conv3-128
maxpool		
conv3-256	conv3-256	conv3-256
conv3-256	conv3-256	conv3-256
	conv1-256	**conv3-256**
maxpool		
conv3-512	conv3-512	conv3-512
conv3-512	conv3-512	conv3-512
	conv1-512	**conv3-512**
maxpool		
conv3-512	conv3-512	conv3-512
conv3-512	conv3-512	conv3-512
	conv1-512	**conv3-512**
maxpool		
FC-4096		
FC-4096		
FC-1000		
soft-max		

TOTAL memory: 24M * 4 bytes ~= 93MB / image (only forward! ~*2 for bwd)
TOTAL params: 138M parameters

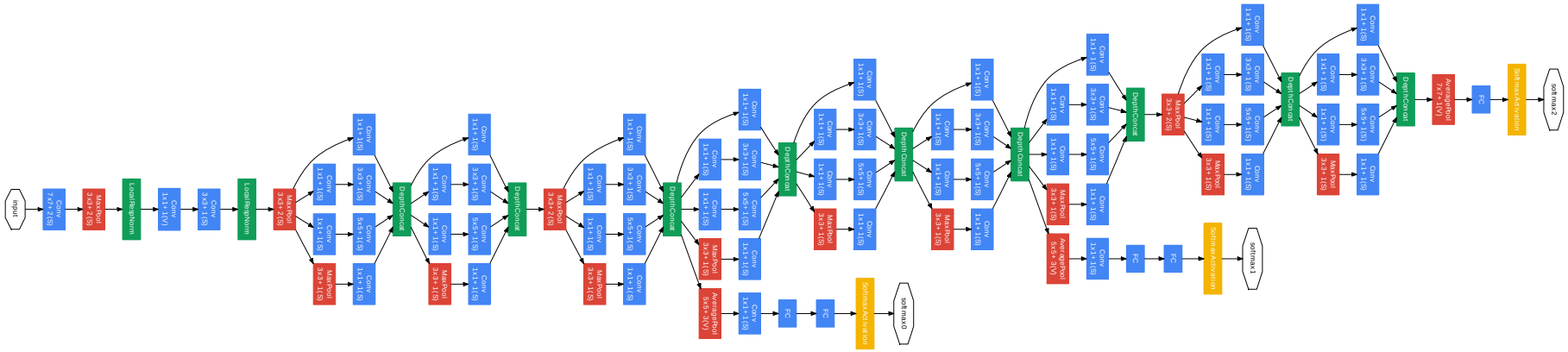
Slide: A. Karpathy / L. Fei Fei

GoogLeNet [Szegedy et al., CVPR'15]

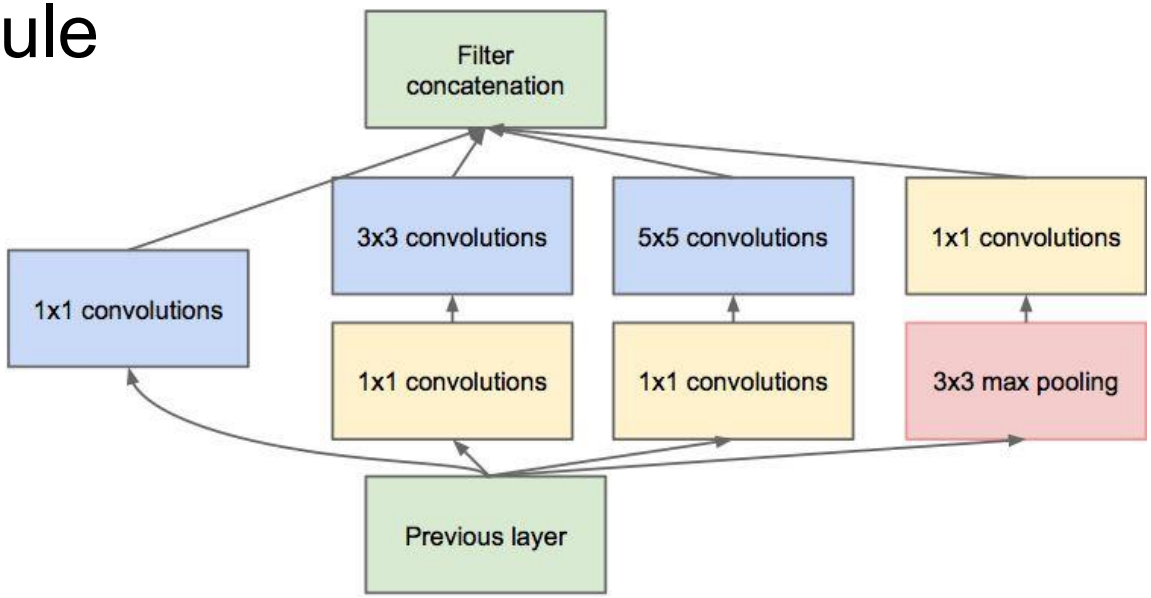


- Only ~5M params
  - After last pooling layer, volume is of size [7x7x1024]
  - Use average pooling into [1x1x1024] before the FC layer.
- Use additional losses during training

GoogLeNet [Szegedy et al., CVPR'15]



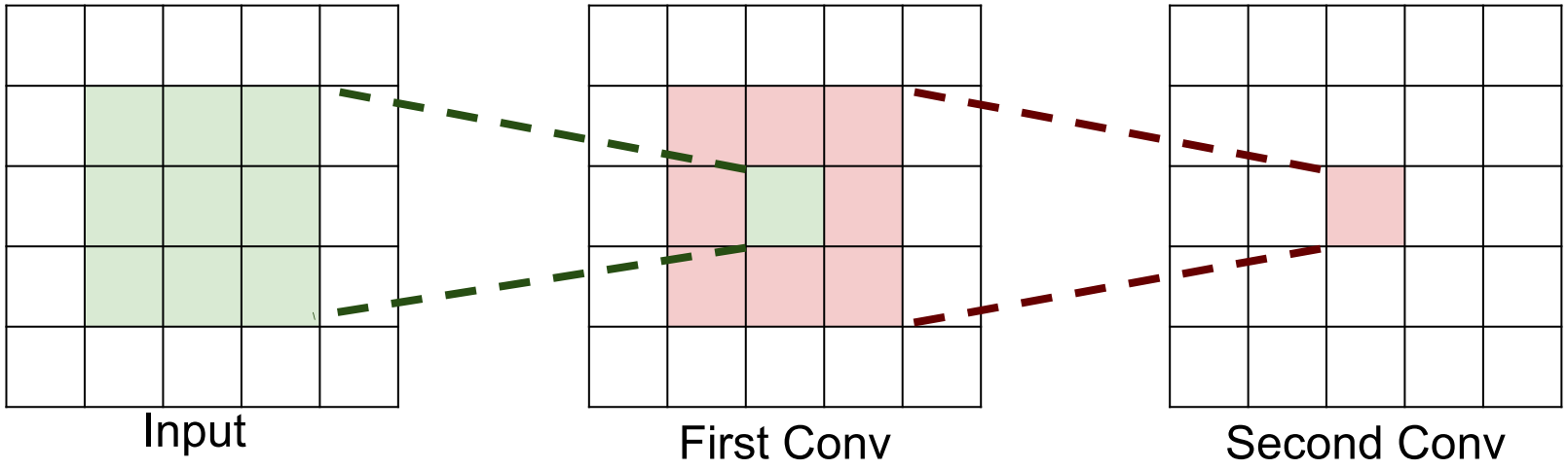
- "Inception" module



The power of small filters

Suppose we stack two CONV layers with receptive filed size 3x3

Q: What region of input does each neuron in 2nd CONV see?



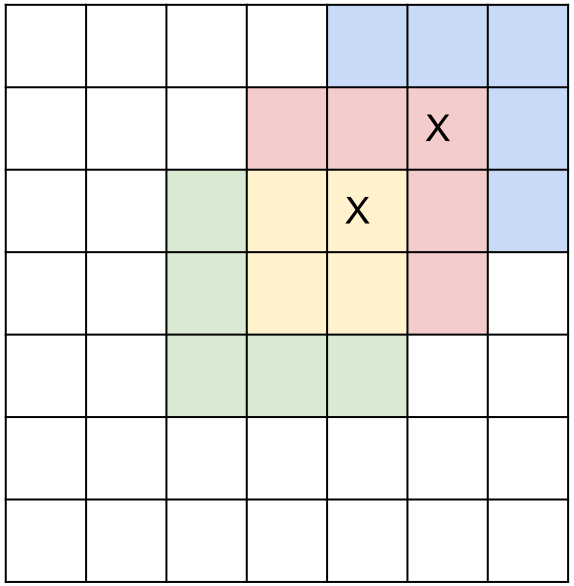
Answer: [5x5]

Slide: A. Karpathy / L. Fei Fei

The power of small filters

Suppose we stack three CONV layers with receptive filed size 3x3

Q: What region of input does each neuron in 3rd CONV see?



Answer: [7x7]

Slide: A. Karpathy / L. Fei Fei

The power of small filters

Suppose input has depth C & we want output depth C as well.

1x CONV with 7x7 filters

Number of weights:

C*(7*7*C)
= **49 C^2**

3x CONV with 3x3 filters

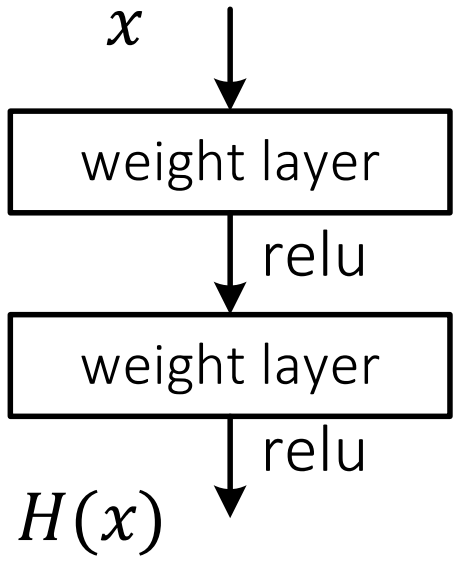
Number of weights:

C*(3*3*C) + C*(3*3*C) + C*(3*3*C)
= 3 * 9 * C^2
= **27 C^2**

Residual networks [ResNets]

Plain net

any two stacked layers



$H(x)$ is any desired mapping,

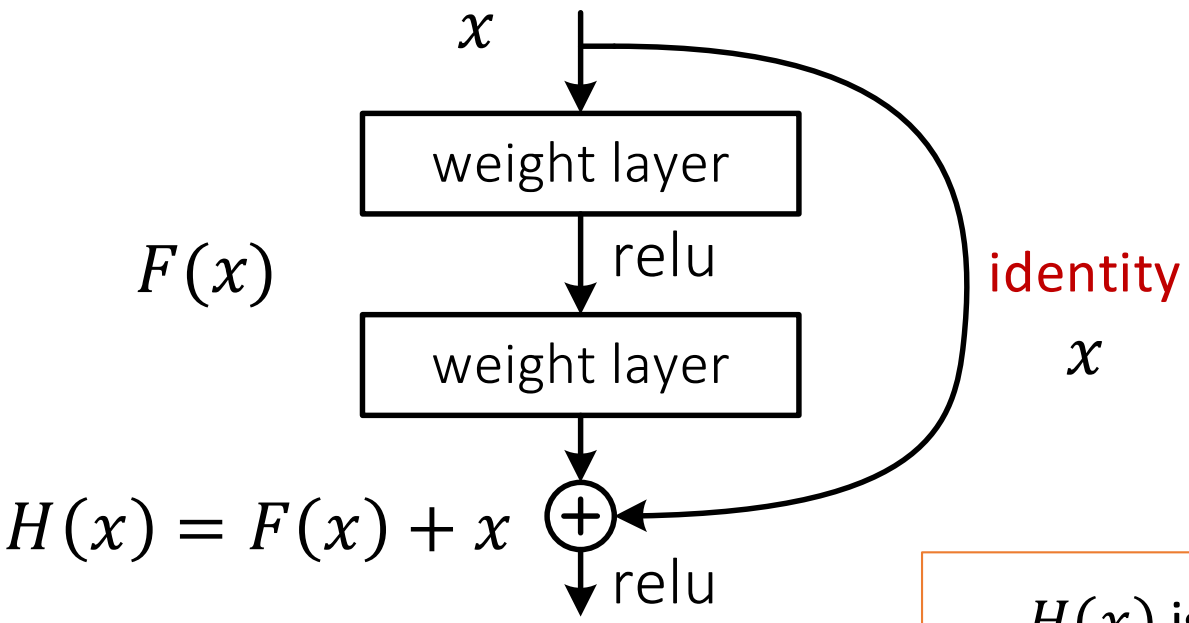
hope the 2 weight layers fit $H(x)$

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Deep Residual Learning for Image Recognition”. CVPR 2016.

Slide: K. He

Residual networks [ResNets]

Residual net



$H(x)$ is any desired mapping,

~~hope the 2 weight layers fit $H(x)$~~

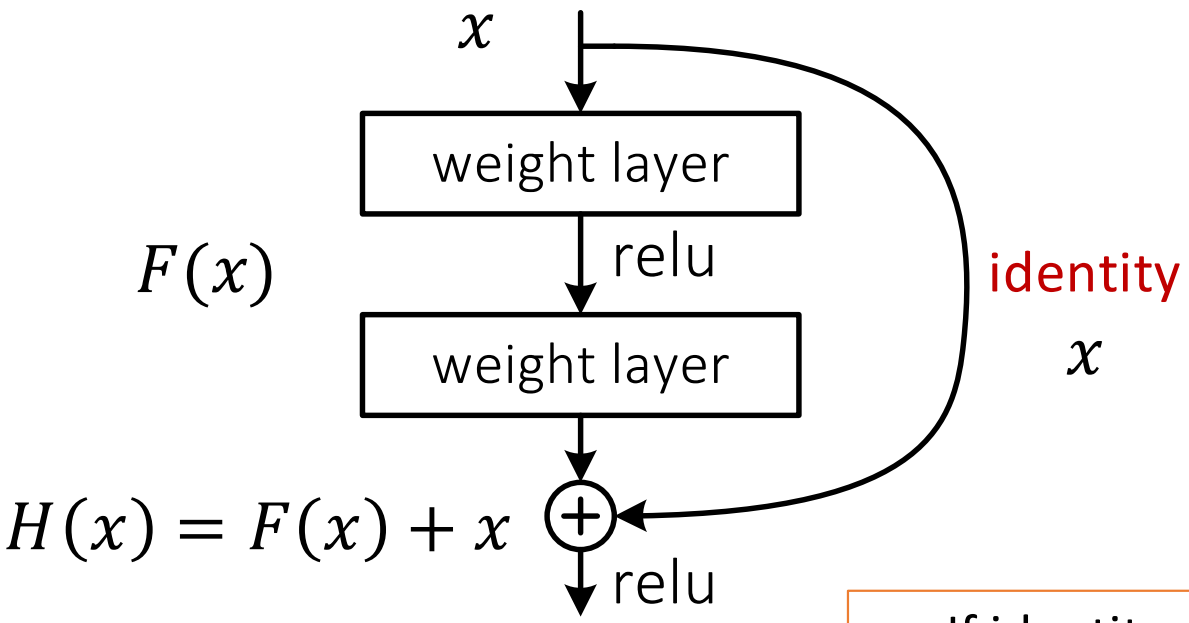
hope the 2 weight layers fit $F(x)$

let $H(x) = F(x) + x$

Slide: K. He

Residual networks [ResNets]

$F(x)$ is a residual mapping w.r.t. identity



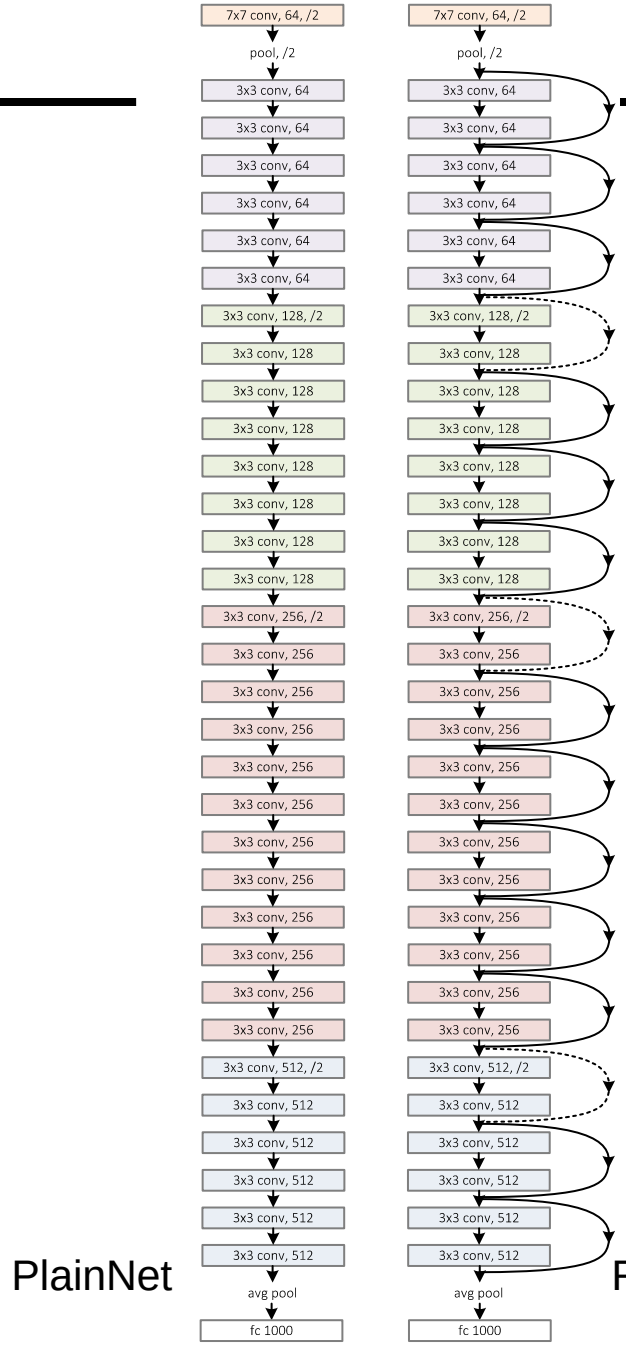
- If identity were optimal, easy to set weights as 0
- If optimal mapping is closer to identity, easier to find small fluctuations

Slide: K. He

Network design

Basic design (vgg-style)

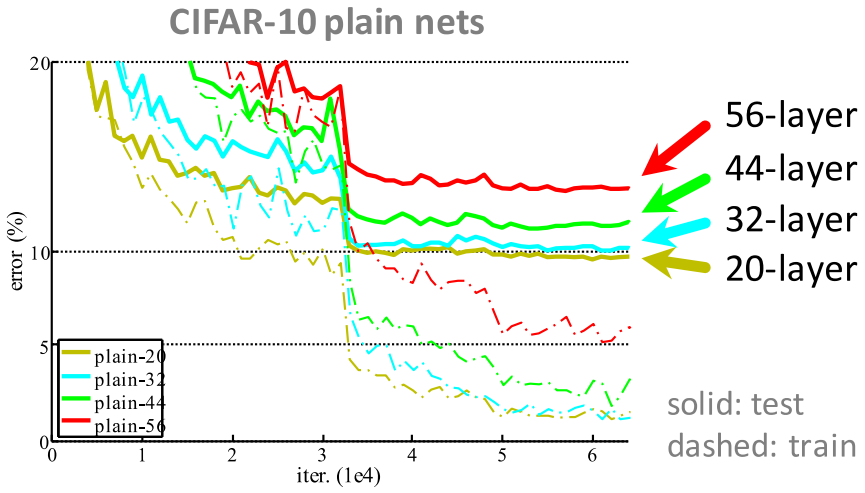
- almost all 3x3 conv
- Spatial size /2 => # filters x2
- Simple design, just deep
- No fully connected layers
- No dropout

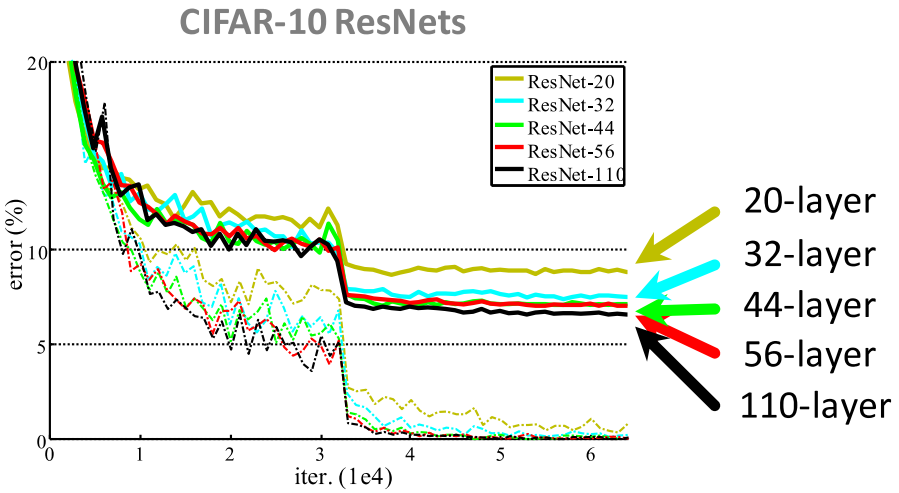


PlainNet ResNet

Slide: K. He

CIFAR-10 experiments





- Deep ResNets can be trained without difficulties
- Deeper ResNets have lower training error and lower test error

Slide: K. He

Outline

1. Convolutional neural networks (CNNs)
2. Understanding and visualizing CNN representations
3. Transferring learnt representation to other tasks
4. Typical CNN architectures for image classification
5. Beyond classification

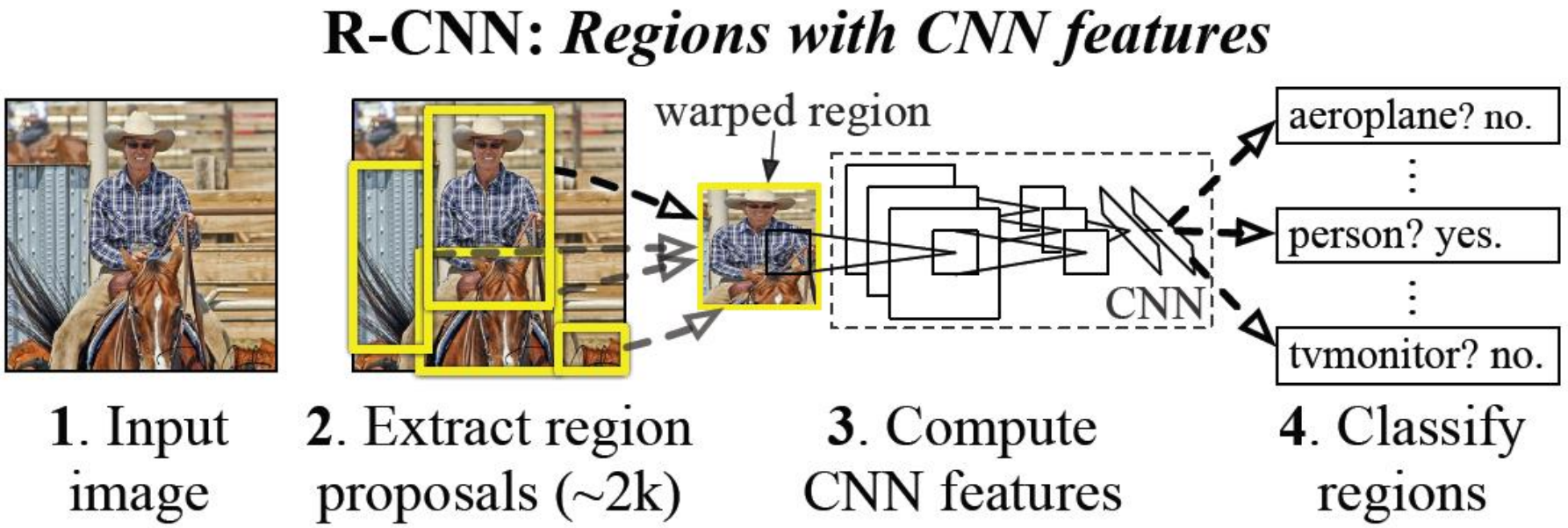
Beyond classification

- Detection
- Segmentation
- Regression
- Pose estimation
- Matching patches
- Synthesis

and many more…

Slide: Jia-Bin Huang

CNN features for detection

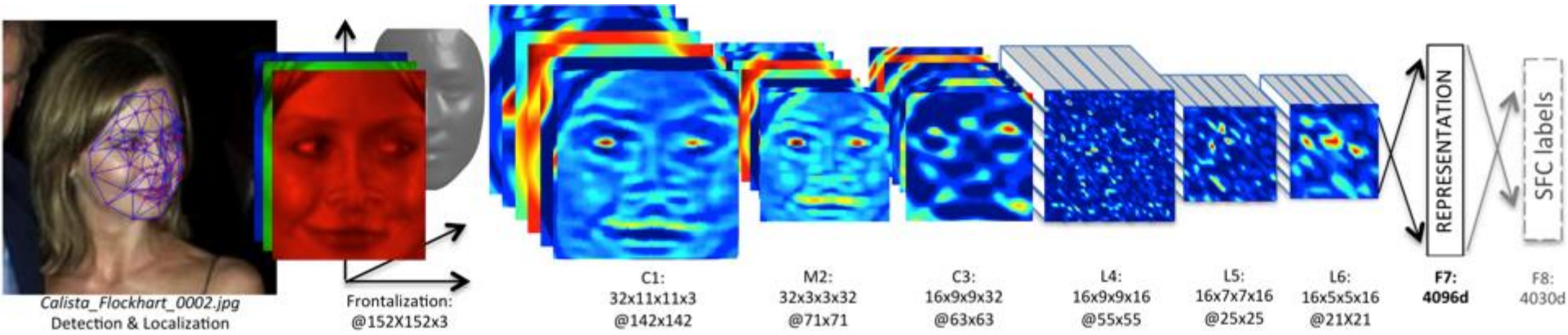


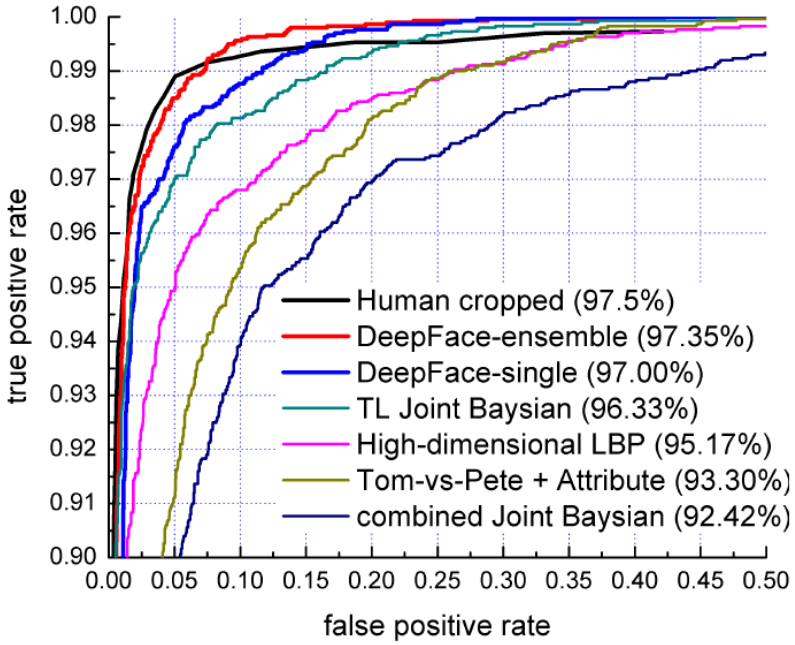
Object detection system overview. Our system (1) takes an input image, (2) extracts around 2000 bottom-up region proposals, (3) computes features for each proposal using a large convolutional neural network (CNN), and then (4) classifies each region using class-specific linear SVMs. **R-CNN achieves a mean average precision (mAP) of 53.7% on PASCAL VOC 2010**. For comparison, Uijlings et al. (2013) report 35.1% mAP using the same region proposals, but with a spatial pyramid and bag-of-visual-words approach. **The popular deformable part models perform at 33.4%.**

R. Girshick, J. Donahue, T. Darrell, and J. Malik, Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation, CVPR'14

Slide: Jia-Bin Huang

CNN features for face verification

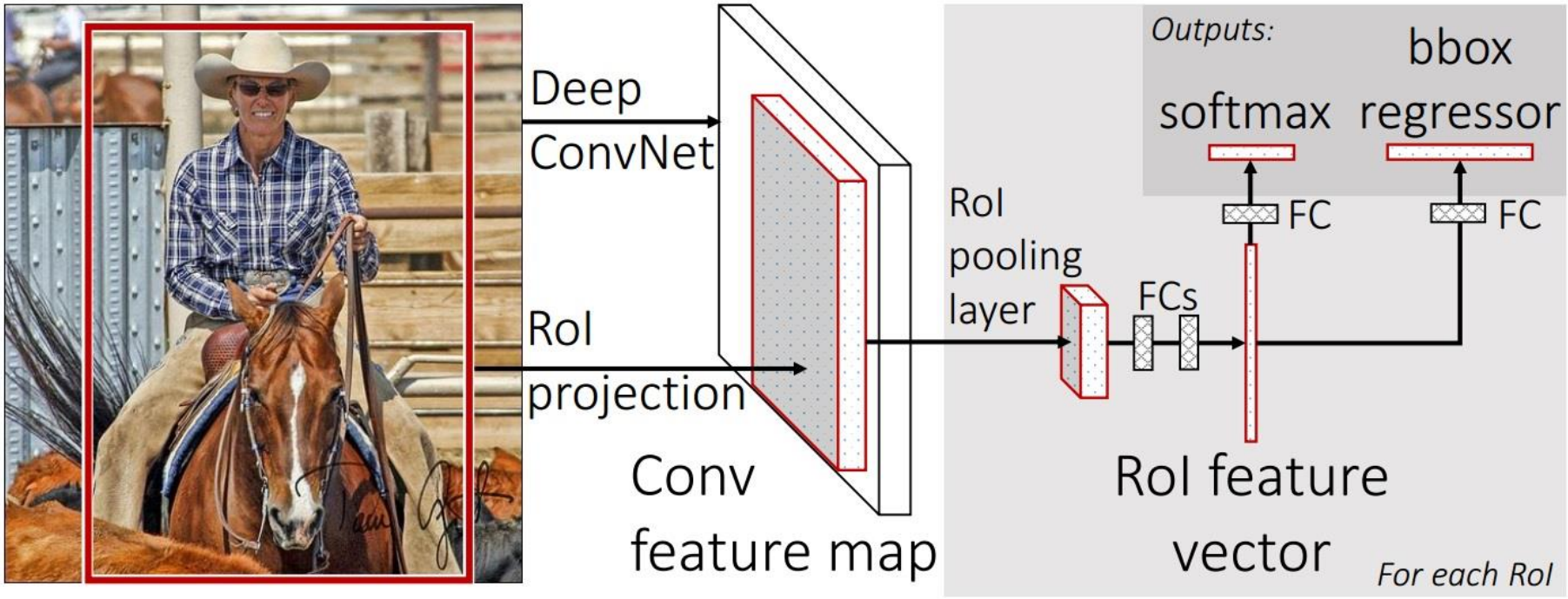




Y. Taigman, M. Yang, M. Ranzato, L. Wolf, DeepFace: Closing the Gap to Human-Level Performance in Face Verification, CVPR'14.

Slide: Jia-Bin Huang

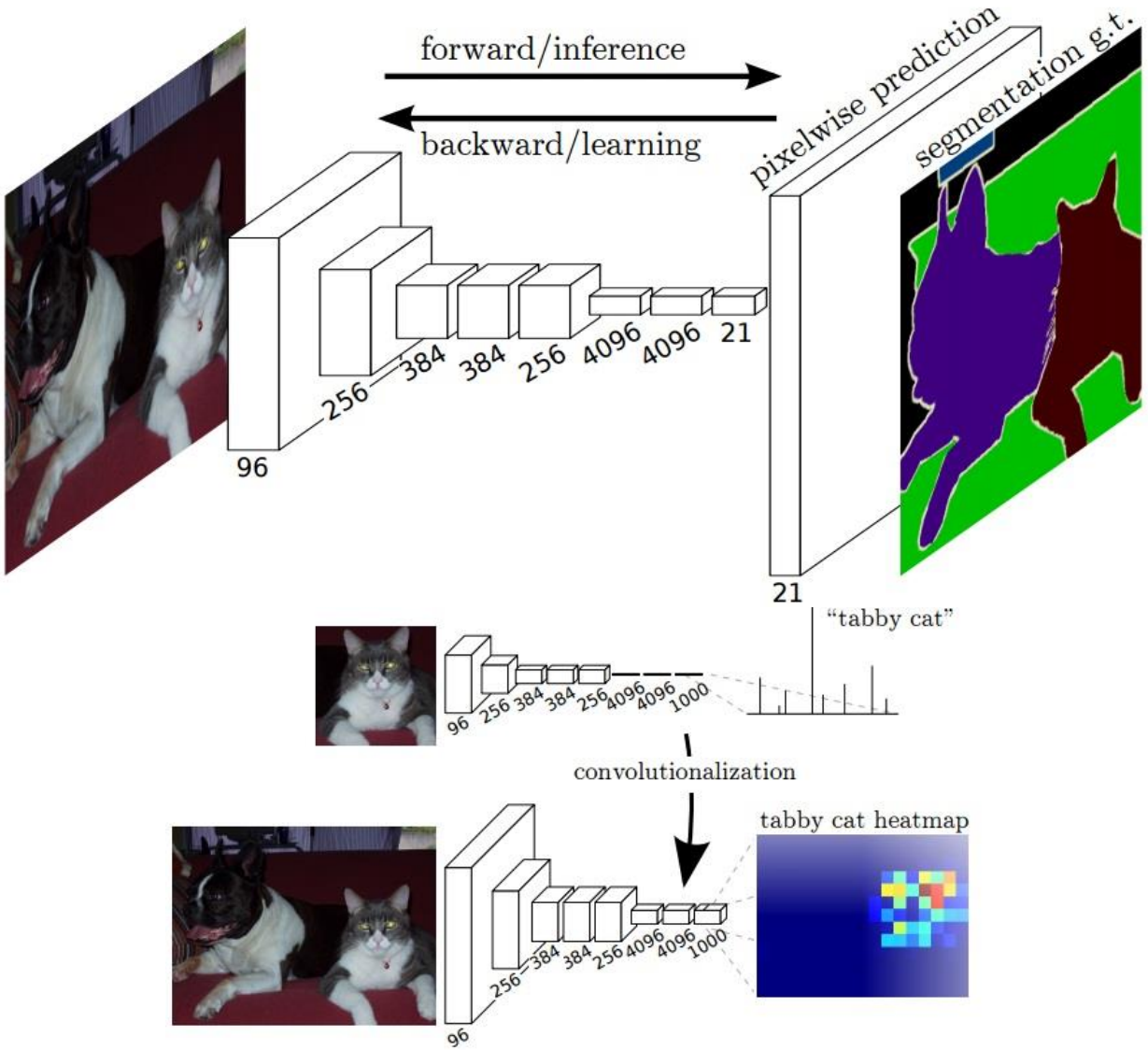
Fast R-CNN



Fast RCNN [Girshick, R 2015]
https://github.com/rbgirshick/fast-rcnn

Slide: Jia-Bin Huang

Labeling Pixels: Semantic Labels



Fully Convolutional Networks for Semantic Segmentation [Long et al. CVPR 2015]

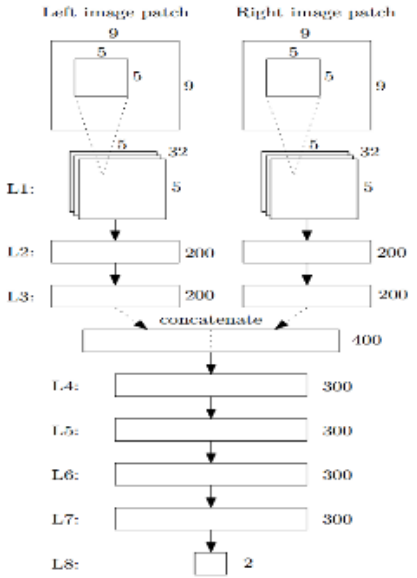
Slide: Jia-Bin Huang

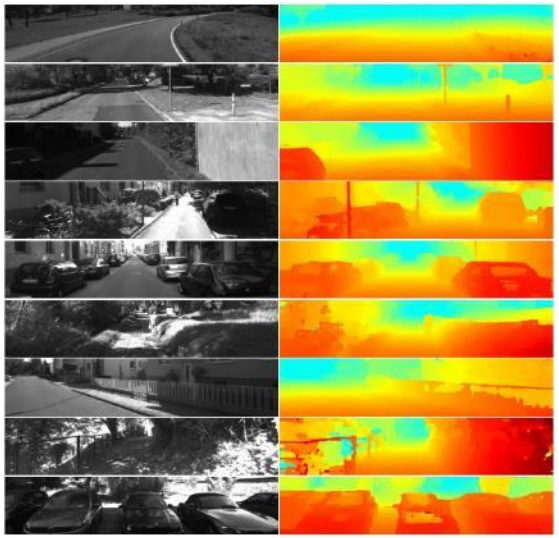
CNN for Regression

DeepPose [Toshev and Szegedy CVPR 2014]

Slide: Jia-Bin Huang

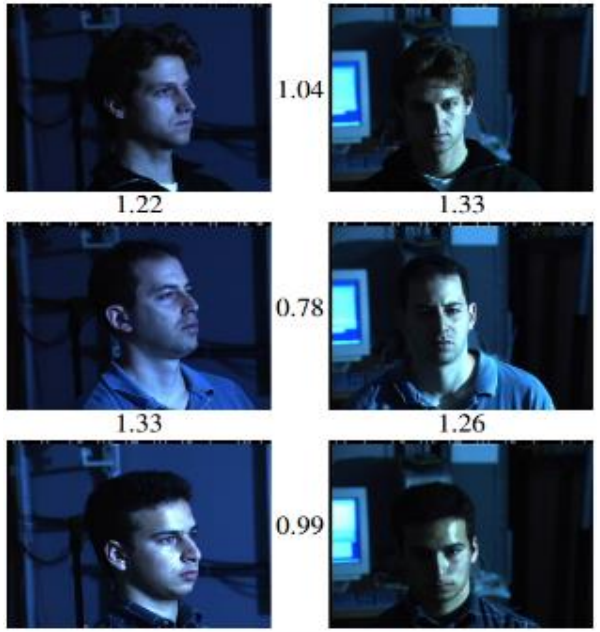
CNN as a Similarity Measure for Matching



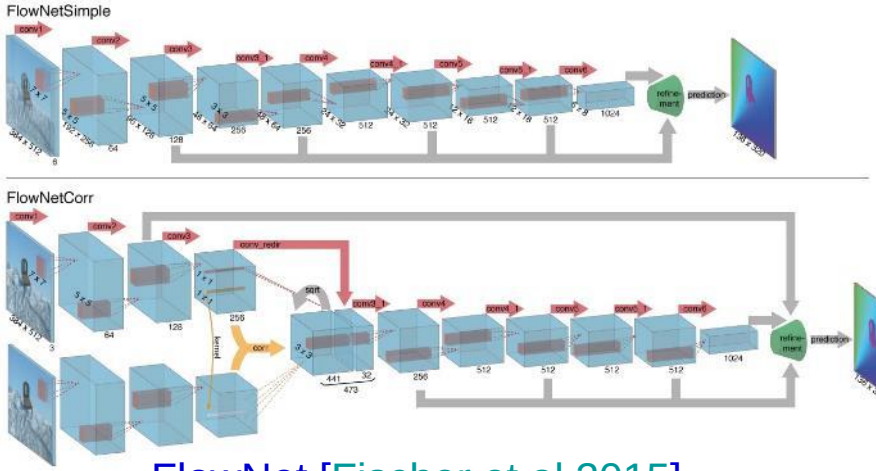


Stereo matching [Zbontar and LeCun CVPR 2015]

Compare patch [Zagoruyko and Komodakis 2015]



FaceNet [Schroff et al. 2015]

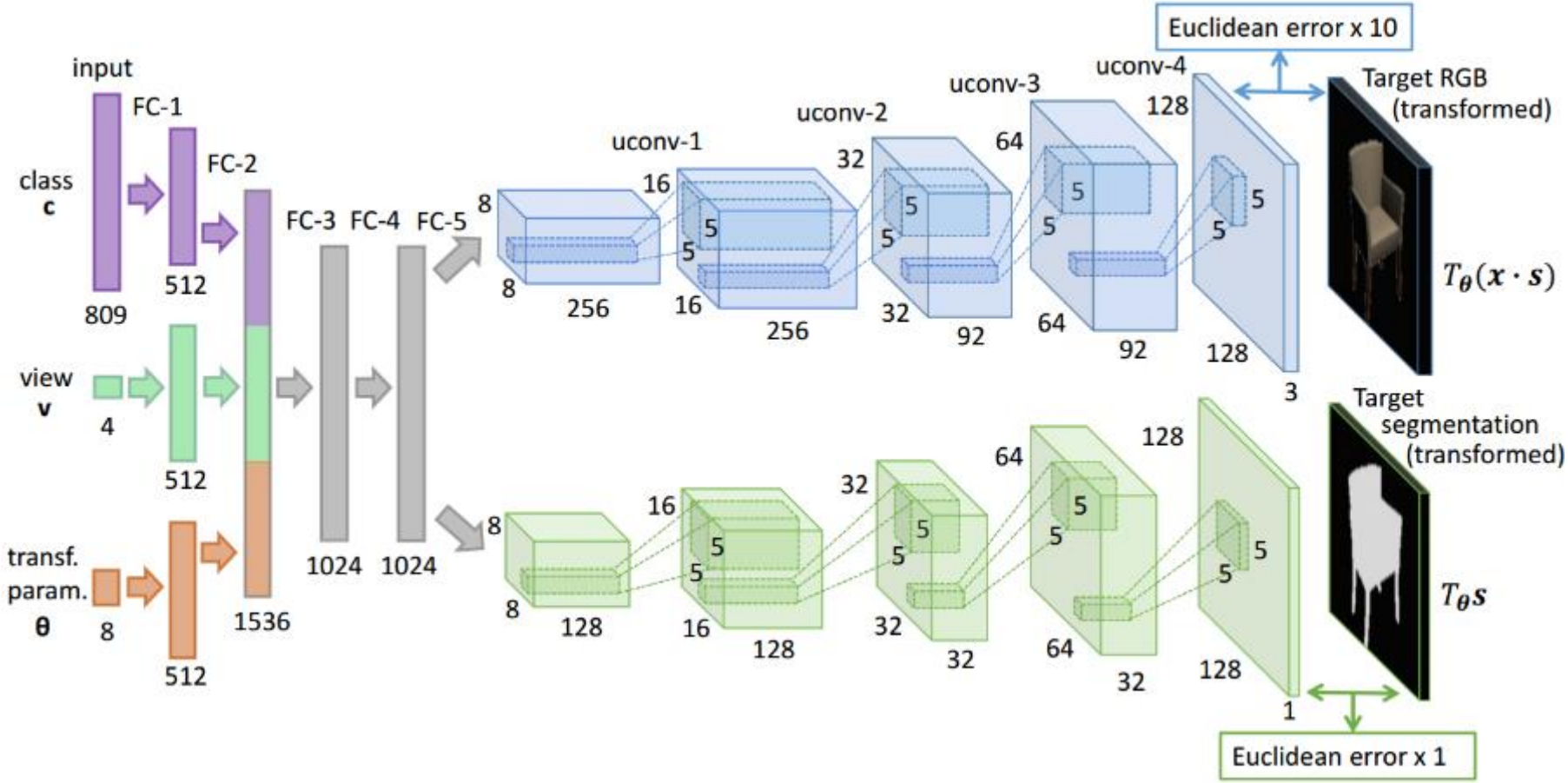


FlowNet [Fischer et al 2015]

Match ground and aerial images [Lin et al. CVPR 2015]

Slide: Jia-Bin Huang

CNN for Image Generation



Learning to Generate Chairs with Convolutional Neural Networks [Dosovitskiy et al. CVPR 2015]

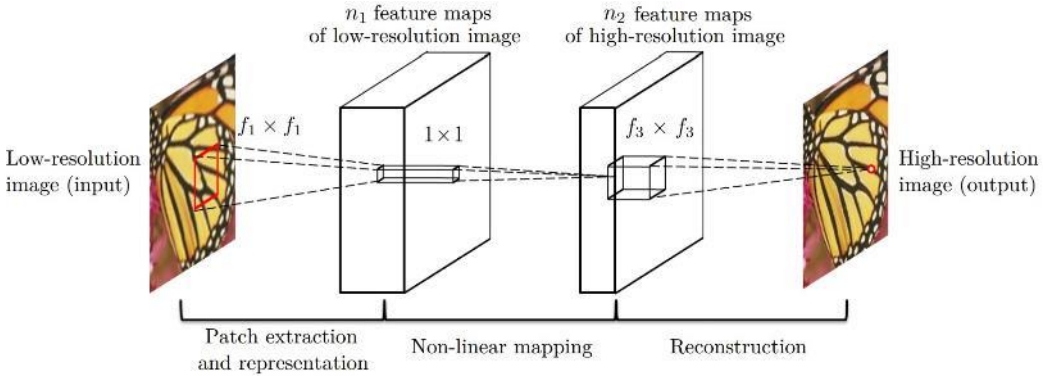
Slide: Jia-Bin Huang

Chair Morphing

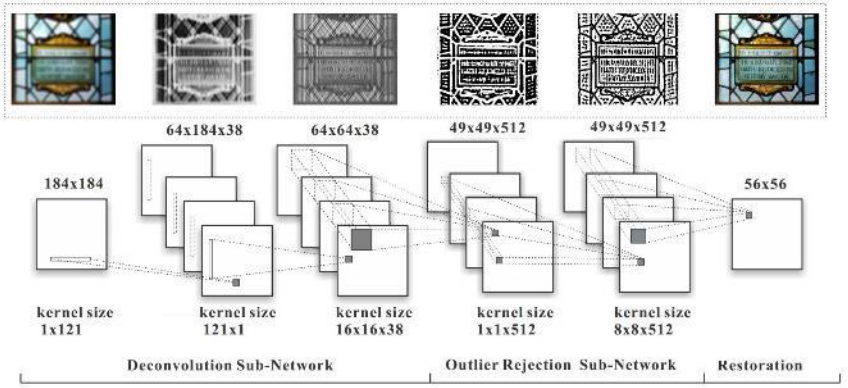
Learning to Generate Chairs with Convolutional Neural Networks [Dosovitskiy et al. CVPR 2015]

Slide: Jia-Bin Huang

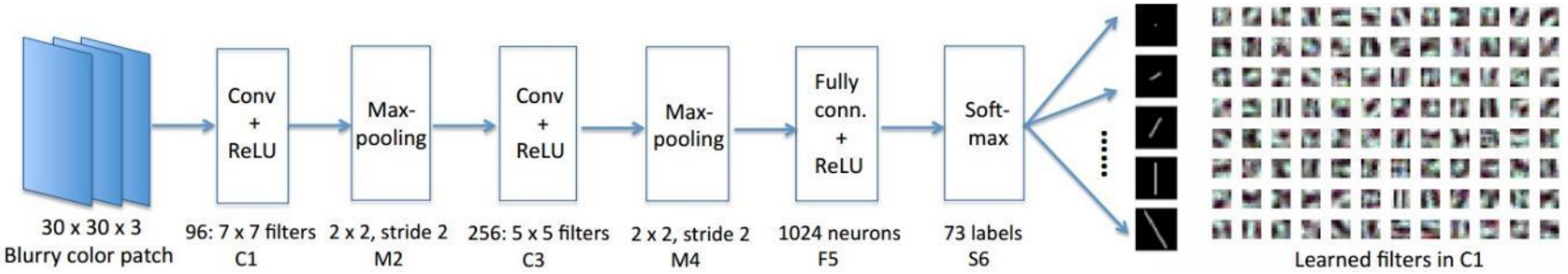
CNN for Image Restoration/Enhancement



Super-resolution
[Dong et al. ECCV 2014]



Non-blind deconvolution
[Xu et al. NIPS 2014]



Non-uniform blur estimation
[Sun et al. CVPR 2015]

Slide: Jia-Bin Huang

Summary

1. Convolutional neural networks (CNNs)
   - Building blocks – convolution, non-linearity, max-pooling

2. Understanding and visualizing CNN representations
   - Finding images and their parts that maximize activation
   - Inverting CNN representations

3. Transferring learnt representation to other tasks
   - CNN weights are transferrable with optional fine-tuning.

4. Typical CNN architectures for image classification
   - Increasing depth and performance,
   - Similar # of parameters, slower speed

CNN packages

- Cuda-convnet (Alex Krizhevsky, Google)
  - https://code.google.com/p/cuda-convnet/
- Caffe (Y. Jia, Berkeley)
  - http://caffe.berkeleyvision.org/
- Overfeat (NYU, based on Torch)
  - http://cilvr.nyu.edu/doku.php?id=software:overfeat:start
- MatConvNet (Oxford, for Matlab)
  - http://www.vlfeat.org/matconvnet/
- Tensor Flow (Google, Python)
  - https://www.tensorflow.org/
- PyTorch (FaceBook, Python)
  - http://pytorch.org/

References

Origins of convolutional neural networks

[Rosenblatt'57] F. Rosenblatt. The perceptron: A perceiving and recognizing automaton. Technical Report 85-460-1, Project PARA, [50] Cornell Aeronautical Lab, 1957

[Hubel&Wiesel'59] D. H. Hubel and T. N. Wiesel. Receptive fields of single neurones in the cat's striate cortex. *Journal of Physiology*,148:574–591, 1959

[Fukushima'80] K. Fukushima. Neocognitron: A self-organizing neu- ral network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202, 1980.

[Rumelhart'86] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, [51] 323(6088):533–536, 1986.

[LeCun et al.'89] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R.E. Howard, W. Hubbard, and L.D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Com- putation*, 1(4):541–551, Winter 1989.

Recent re-emergence of CNNs in computer vision

A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, 2012.

CNN representations are transferrable

[Donahue et al.'13] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolu- tional activation feature for generic visual recognition. *arXiv:1310.1531*, 2013.

[Girshick et al.'13] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

[Oquab et al.'13] M. Oquab, L. Bottou, I. Laptev, J. Sivic. Learning and transferring mid-level representations using convolutional neural networks, hal.inria.fr/hal-00911179, 2013

[Sermanet et al.'13] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus,Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv:1312.6229*, 2013

[Zeiler&Fergus'13] M. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. *arXiv:1311.2901*, 2013