Image and Video Understanding

2VO 710.095 WS

Christoph Feichtenhofer, Axel Pinz

Slide credits:
Many thanks to all the great computer vision researchers on which this presentation relies on.
Most material is taken from tutorials at NIPS, CVPR and BMVC conferences.

Outline

- Convolutional Networs (ConvNets) for Image Classification
  - Operations in each layer
  - Architecture
  - Visualizations
  - Results

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

M. Zeiler & R. Fergus, *Visualizing and Understanding Convolutional Networks*, ECCV, 2014

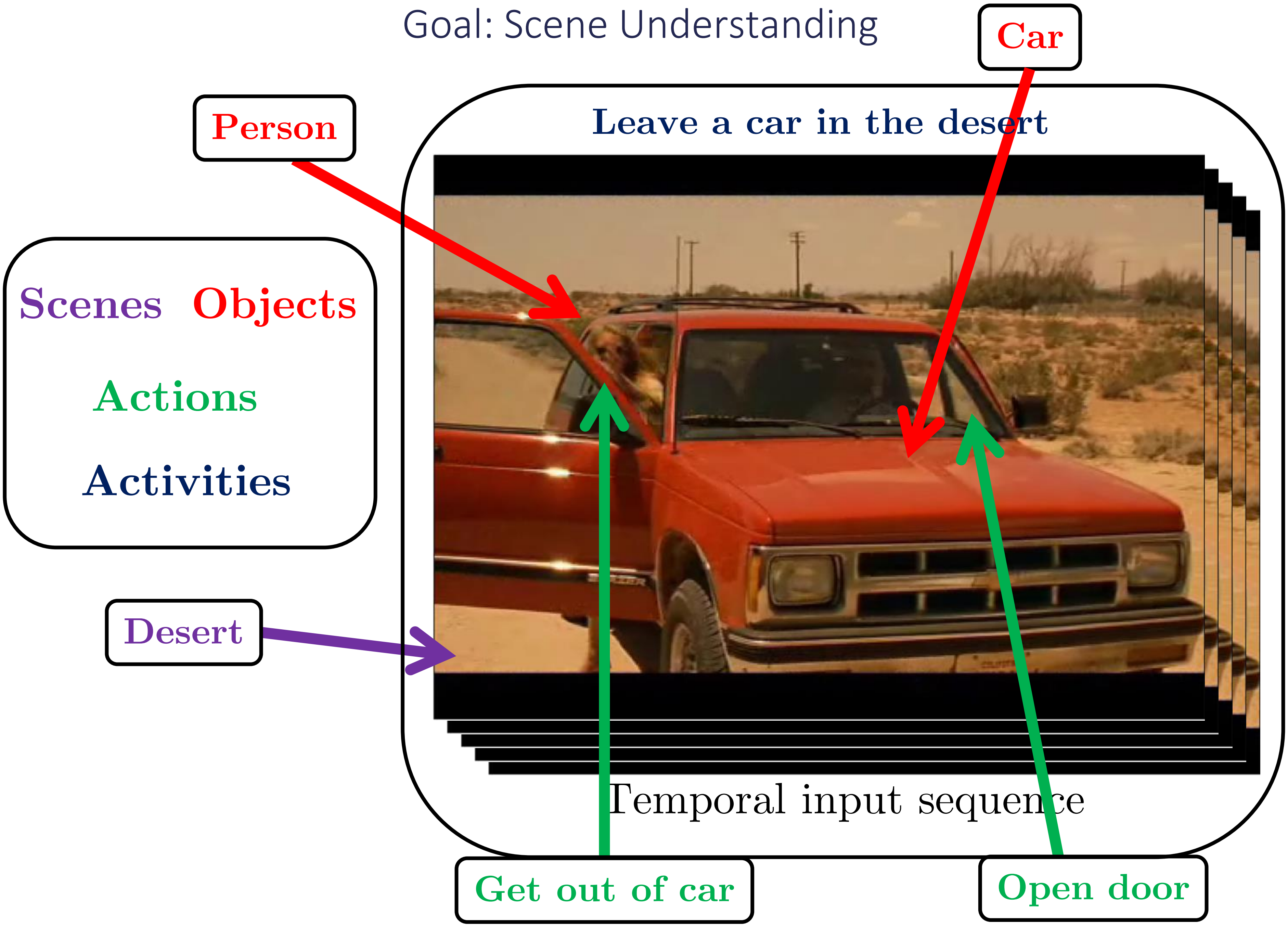
- Representations for Video Classification
  - Hand-designed features
  - Spatiotemporal ConvNets
  - Two-stream ConvNets

Wang et al., *Action Recognition by Dense Trajectories,* CVPR 2011.

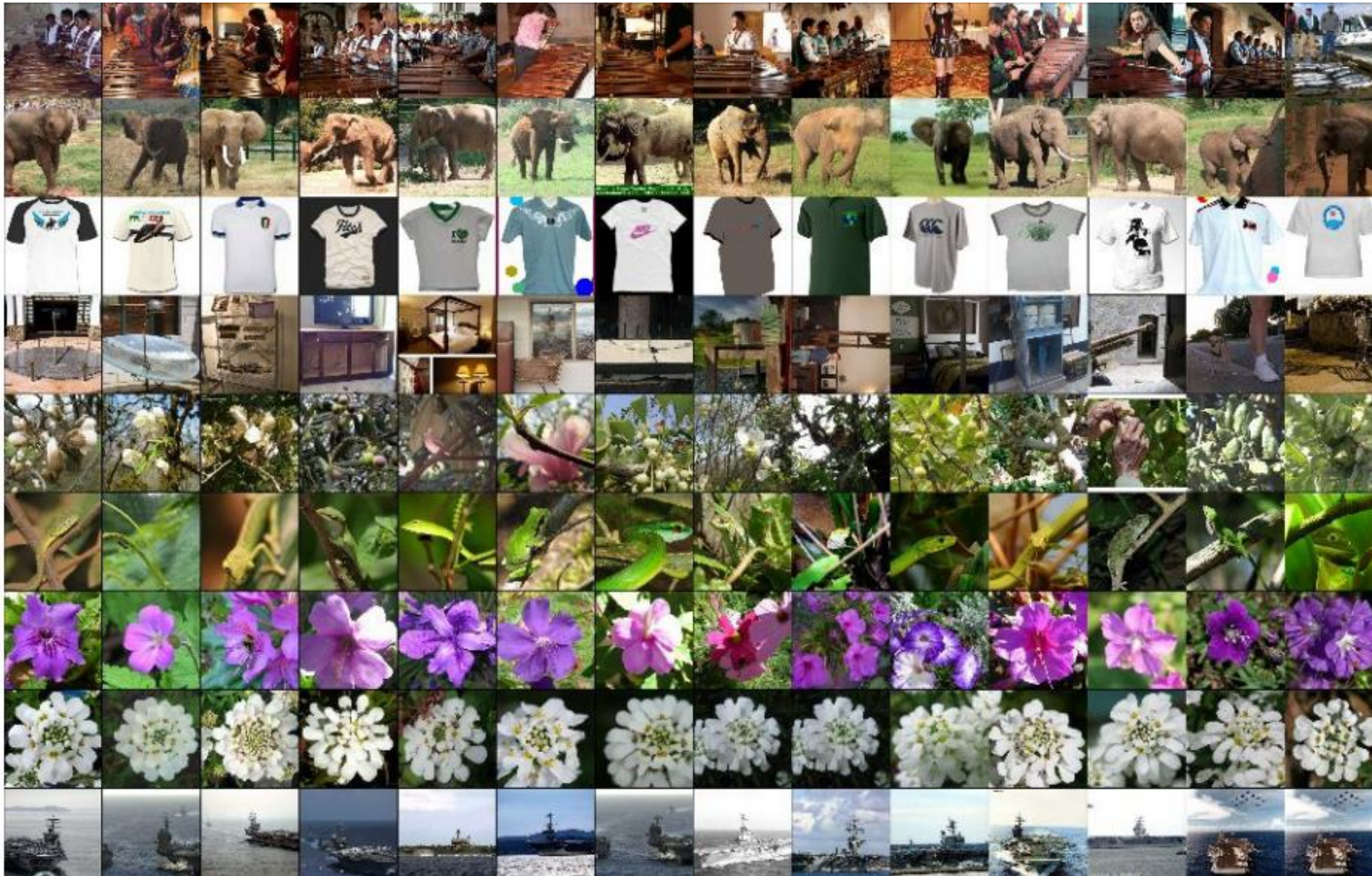
Karpathy et al., *Large-scale Video Classification with Convolutional* Neural Networks, CVPR 2014

K. Simonyan & A. Zisserman, *Two-Stream Convolutional Networks for Action Recognition in Videos,* NIPS 2014

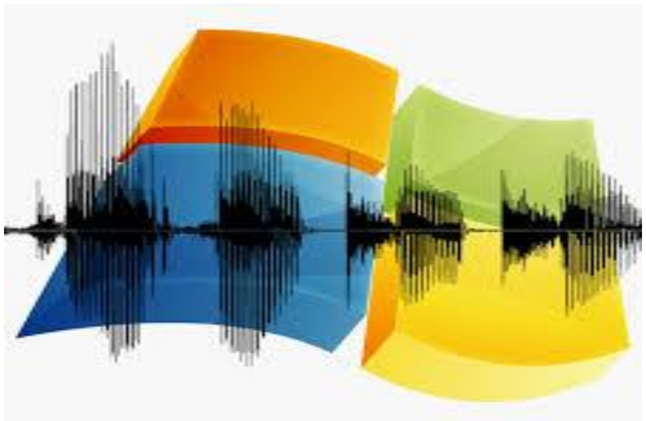
Goal: Scene Understanding

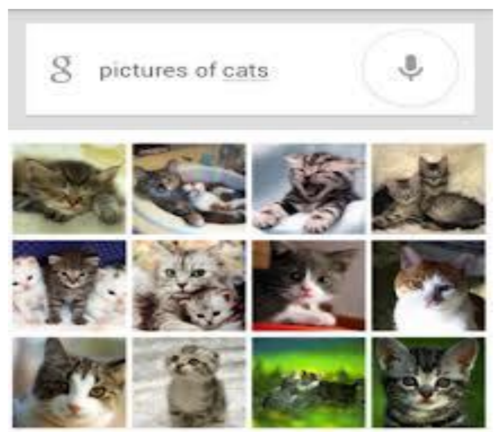


One application: Image retrieval



Deep Learning - breakthrough in visual and speech recognition





Credit: B. Ginzburg

A lot of buzz about Deep Learning

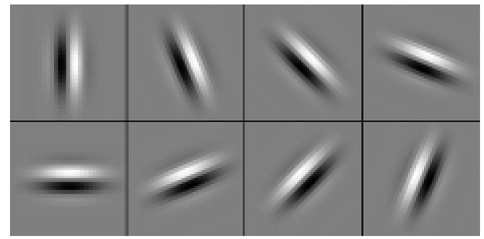
[Microsoft On Deep Learning for Speech]goto 3:00-5:10

Credit: B. Ginzburg

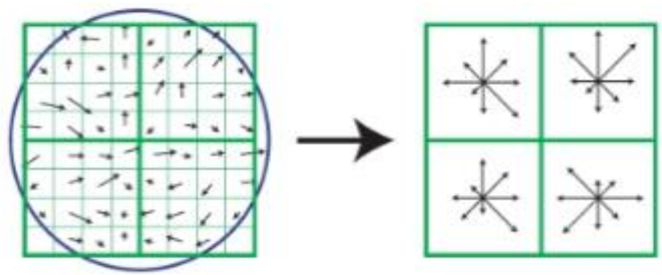
Summary: Compare: SIFT Descriptor

Image Pixels

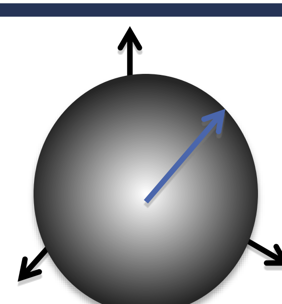
Apply
Gabor filters



Spatial pool
(Sum)



Normalize to unit length

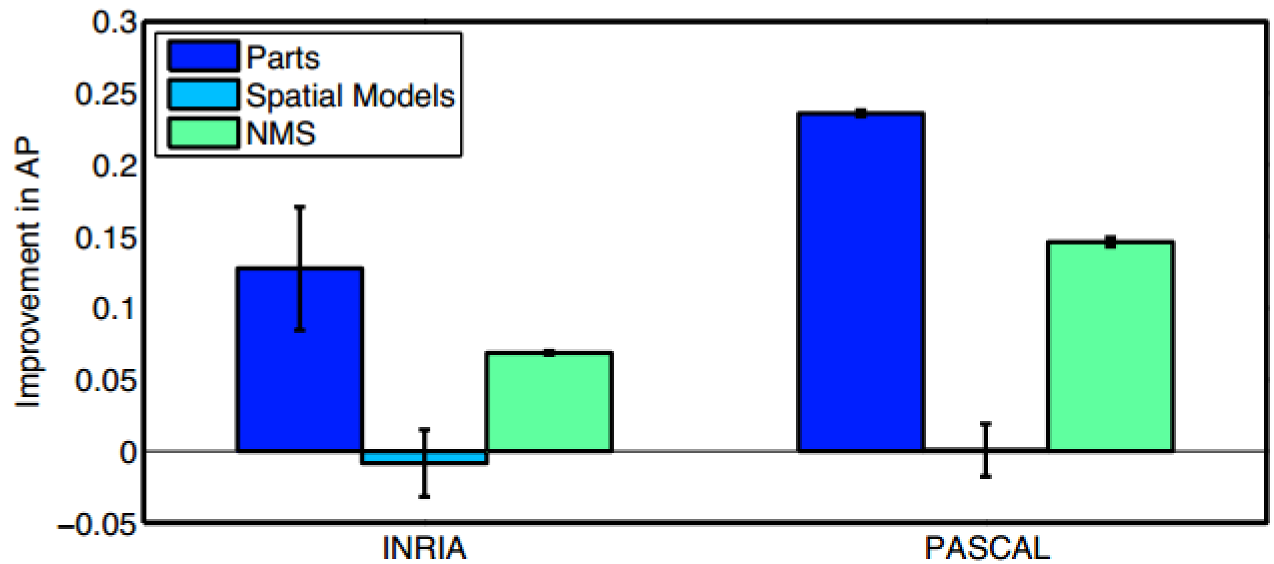


Feature Vector

Credit: R. Fergus

What are the weakest links limiting performance?

- Replace each component of the deformable part model detector with humans
- Good Features (part detection) and accurate Localization (NMS) are most important

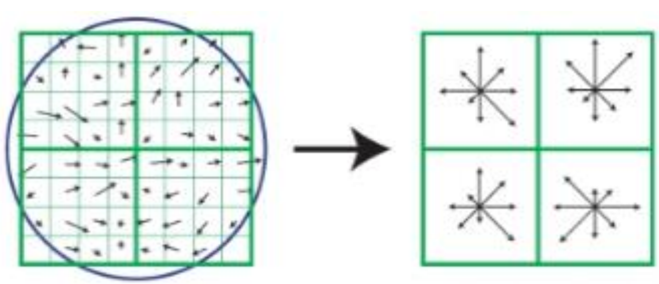


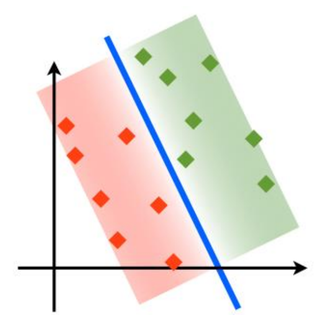
[Parikh & Zitnick CVPR’10]

Typical visual recognition pipeline

- Select / develop features
- Add on top of this Machine Learning for multi-class recognition and train classifier

Image/Video Pixels → Feature Extractor e.g. SIFT, HoG... → Trainable Classifier e.g. SVM → Object Class

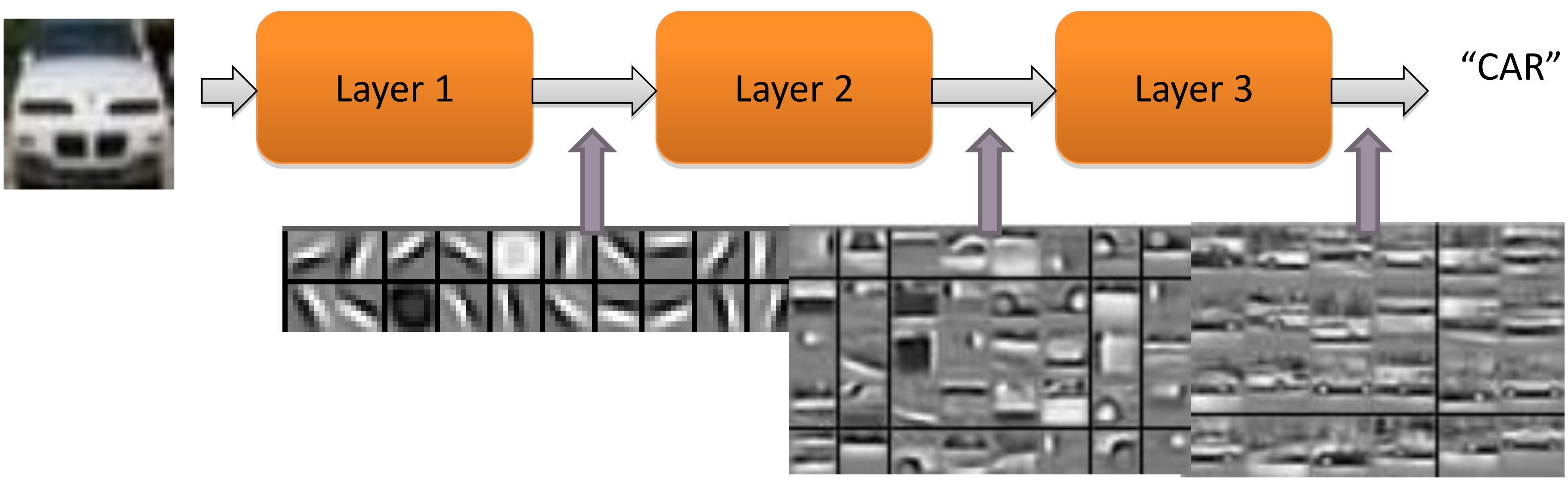




Intuition Behind Deep Neural Nets

- Build features automatically based on training data
- Combine feature extraction and classification
- All the way from pixels → classifier
- One layer extracts features from output of previous layer

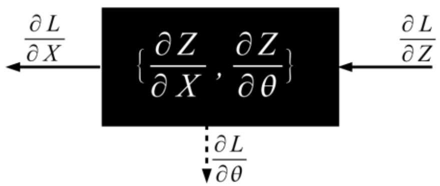
→ **Each box is a feature detector**



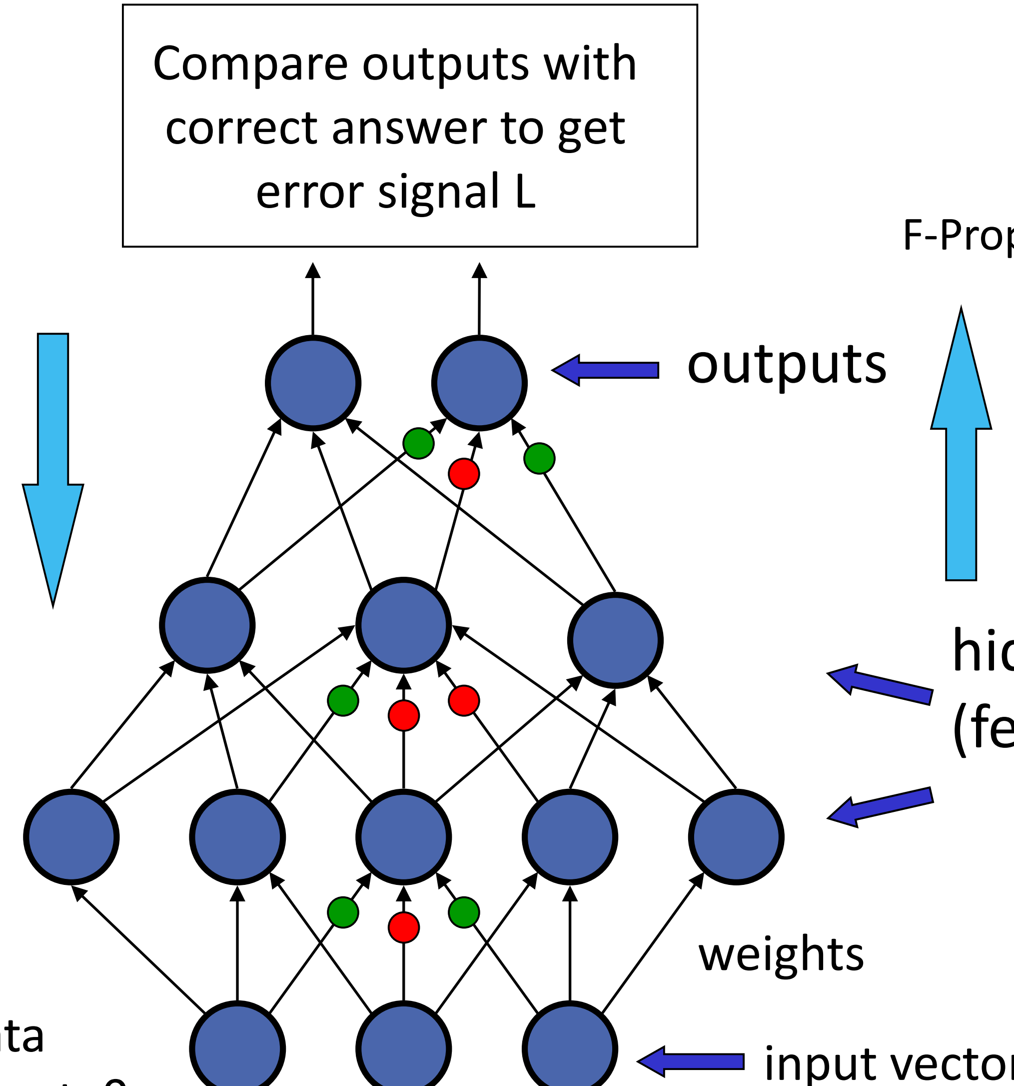
Some Key Ingredients for Convolutional Neural Networks

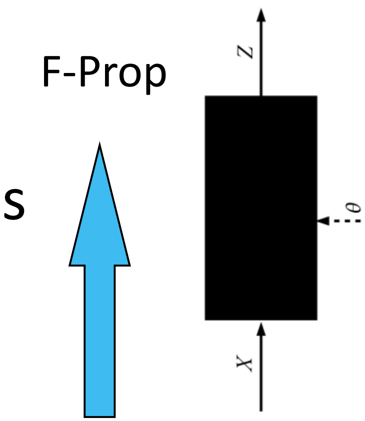
Neural networks trained via backpropagation

B-Prop

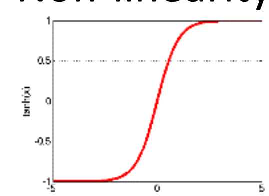


Back-propagate error signal to get derivatives for learning parameters θ





Non-linearity



Training

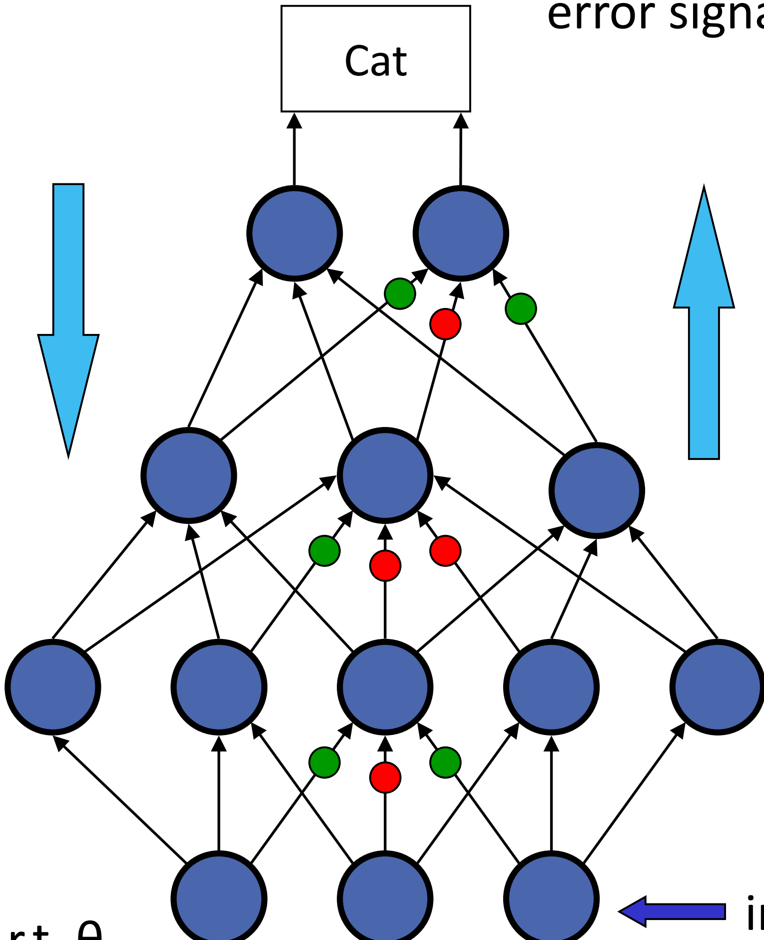
- F-Prop / B-Prop
- Learning by stochastic gradient descent (SGD):
- A) Compute loss L on small mini-batch of data
- B) Compute gradient w.r.t. θ
- C) Use gradient to update θ (make a step in the opposite direction)

Credit: G. Hinton

Neural networks trained via backpropagation

Compare outputs with correct answer to get error signal

Back-propagate error signal to get derivatives for learning parameters θ



input vector

Training

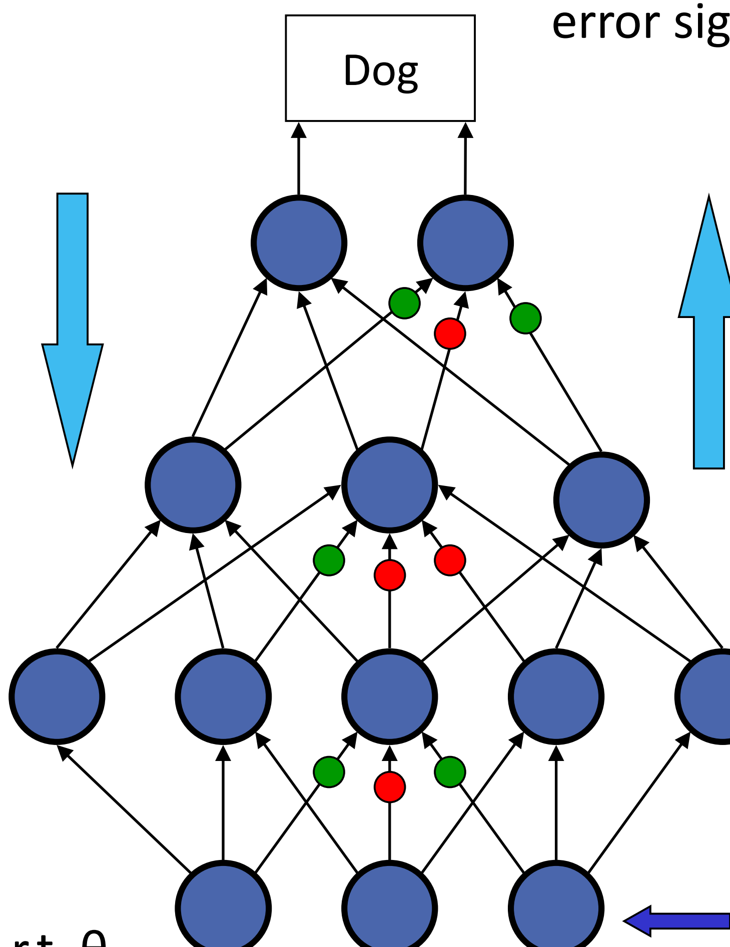
- F-Prop / B-Prop
- Learning by SGD:
- A) Compute loss on small mini-batch
- B) Compute gradient w.r.t. θ
- C) Use gradient to update θ

Credit: G. Hinton

Neural networks trained via backpropagation

Compare outputs with correct answer to get error signal

Back-propagate error signal to get derivatives for learning parameters θ



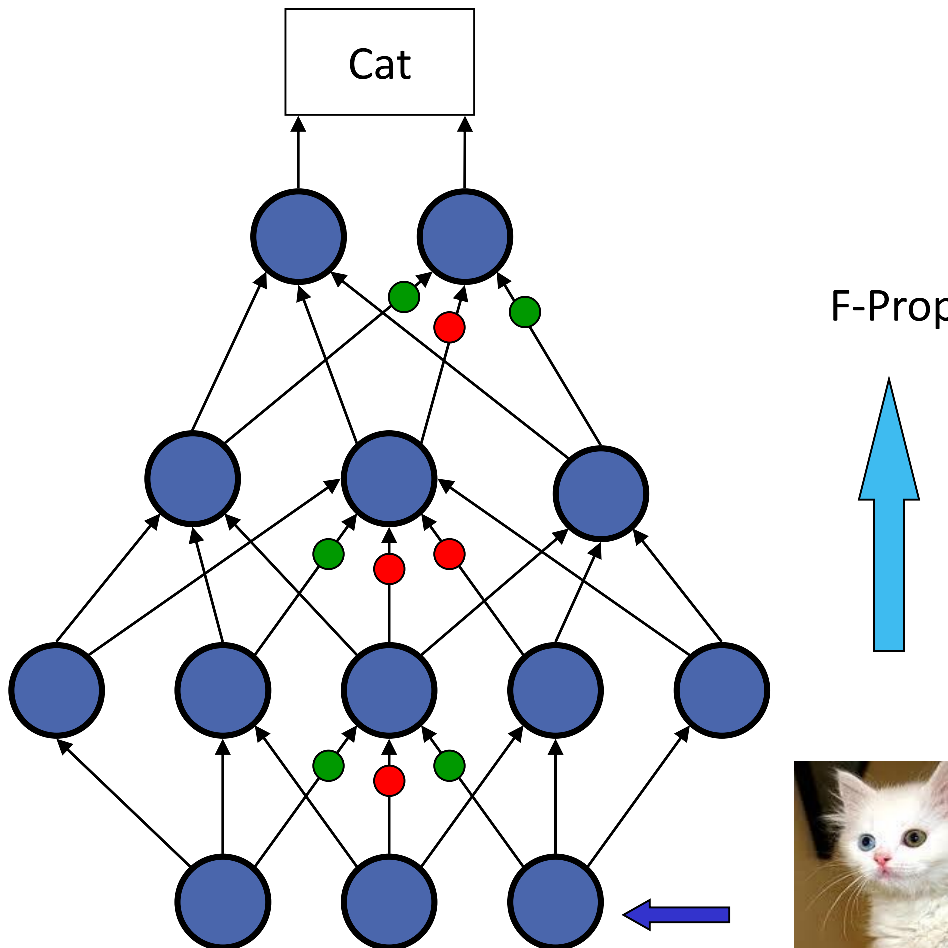
input vector

Training

- F-Prop / B-Prop
- Learning by SGD:
- A) Compute loss on small mini-batch
- B) Compute gradient w.r.t. θ
- C) Use gradient to update θ

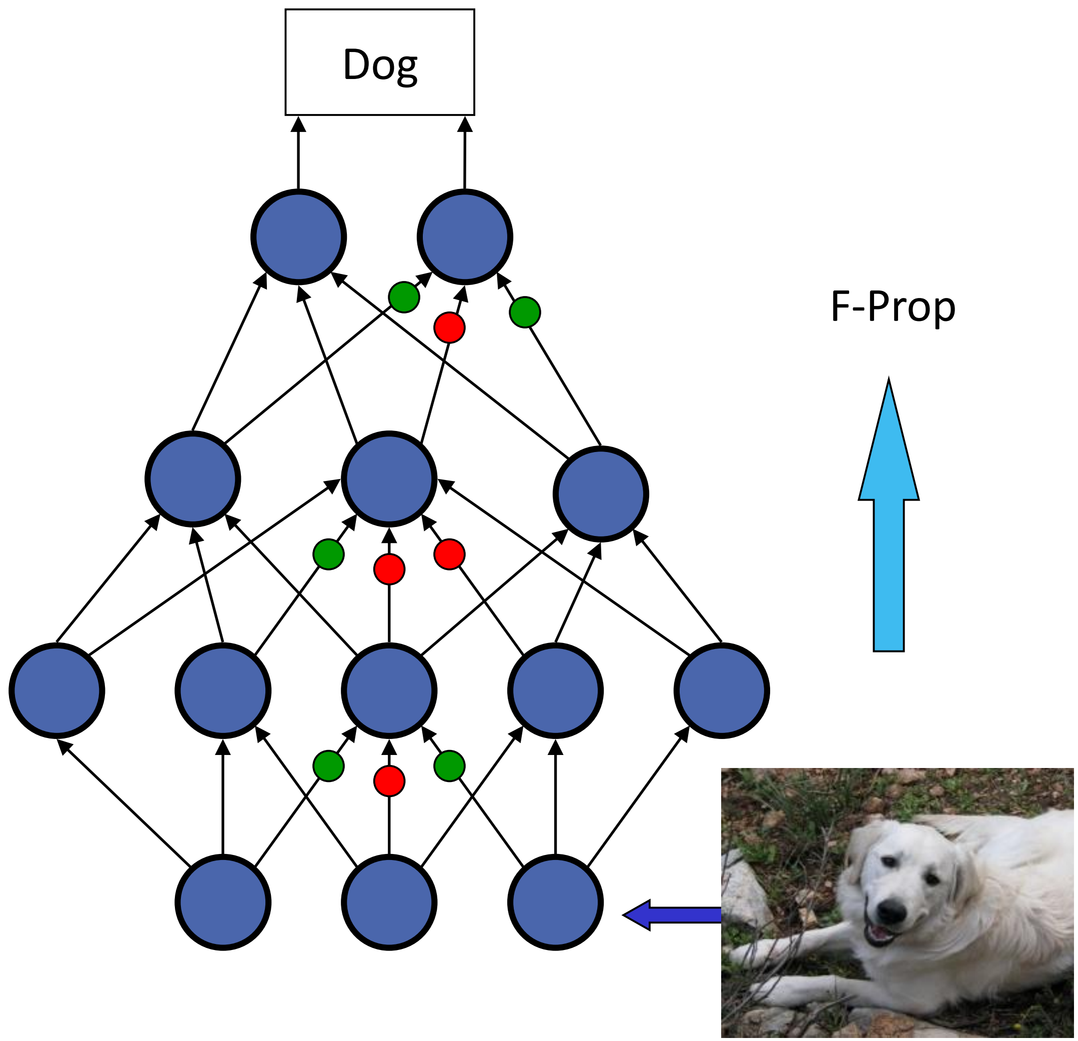
Credit: G. Hinton

Neural networks testing



Credit: G. Hinton

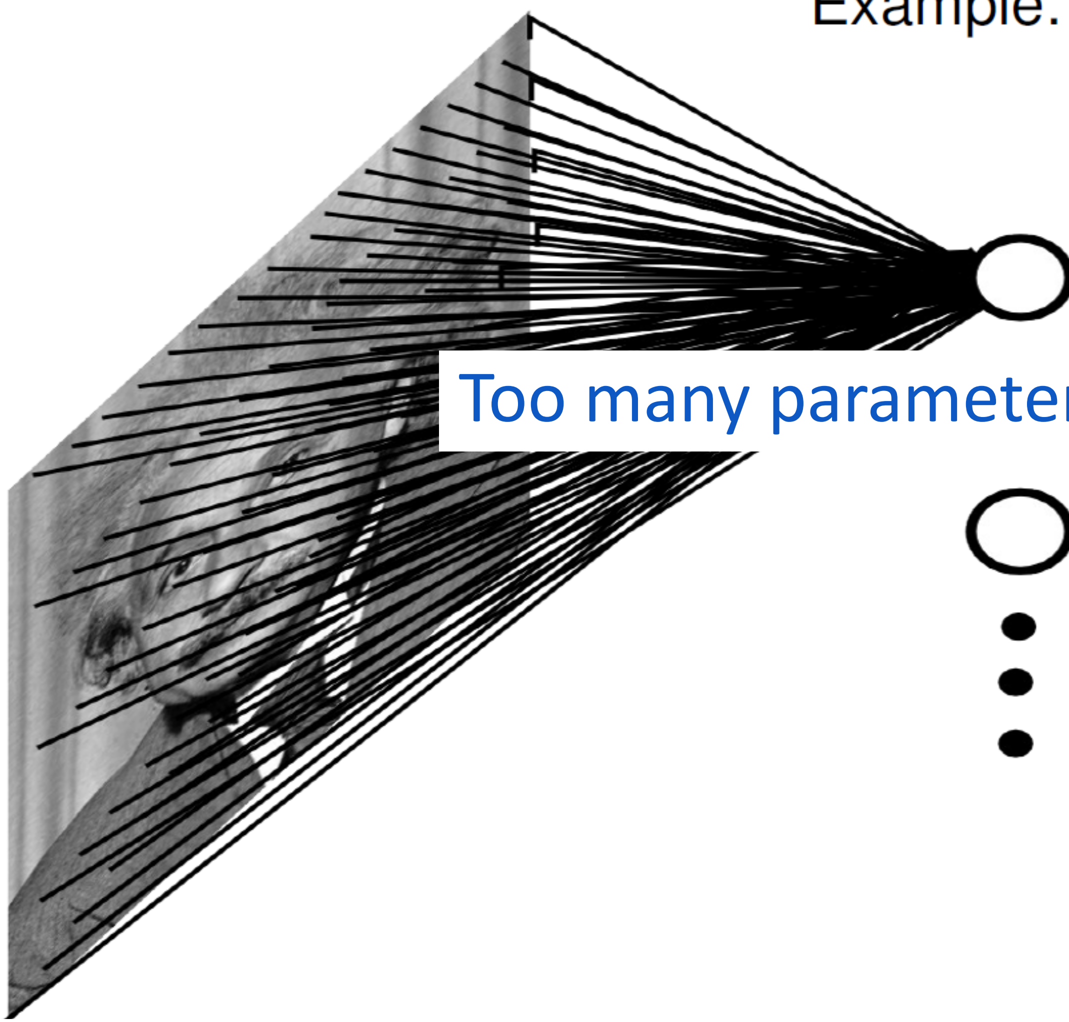
Neural networks testing



Credit: G. Hinton

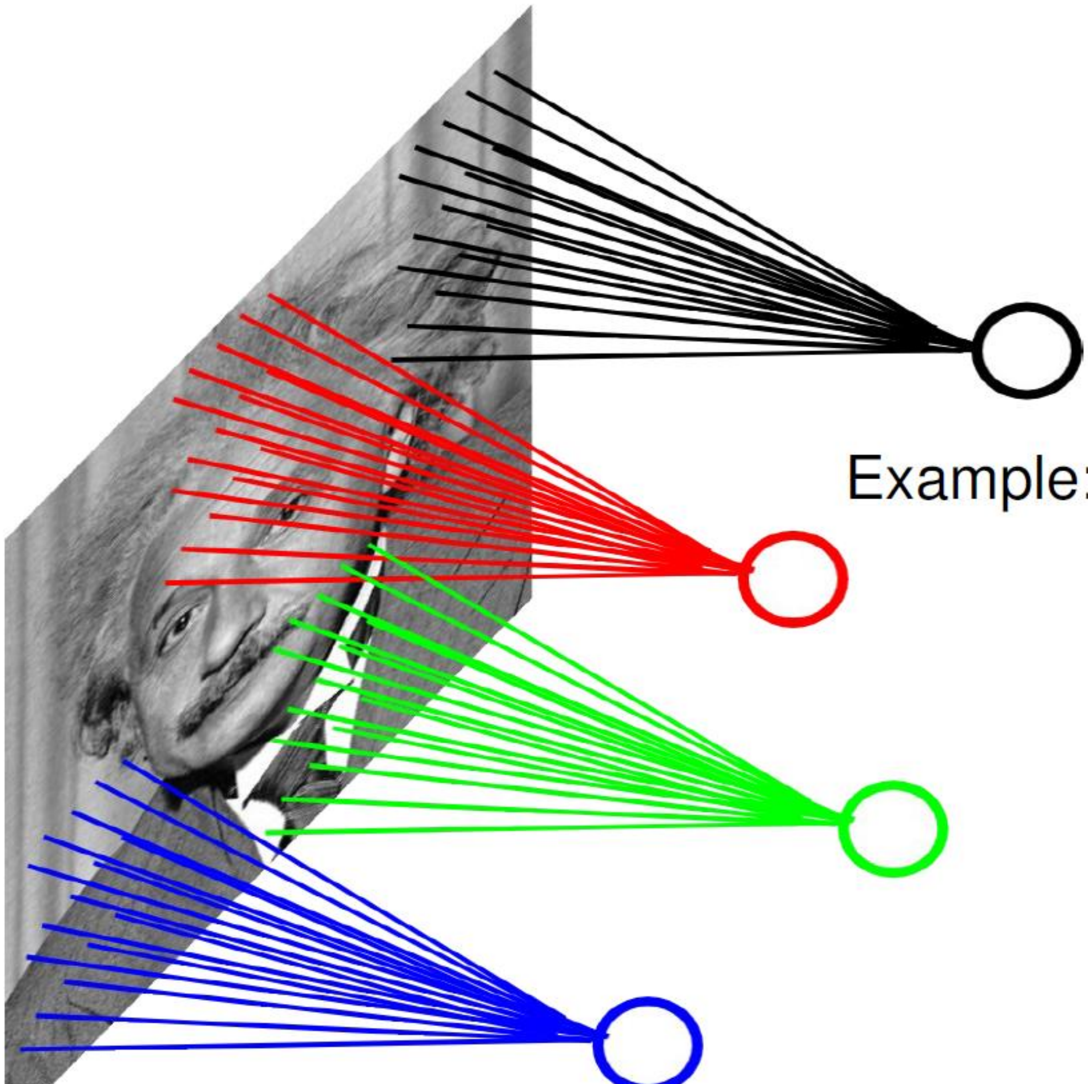
Motivation: Images as a composition of local parts
"Pixel-based" representation

Example: 1000x1000 image



Credit: M. A. Ranzato

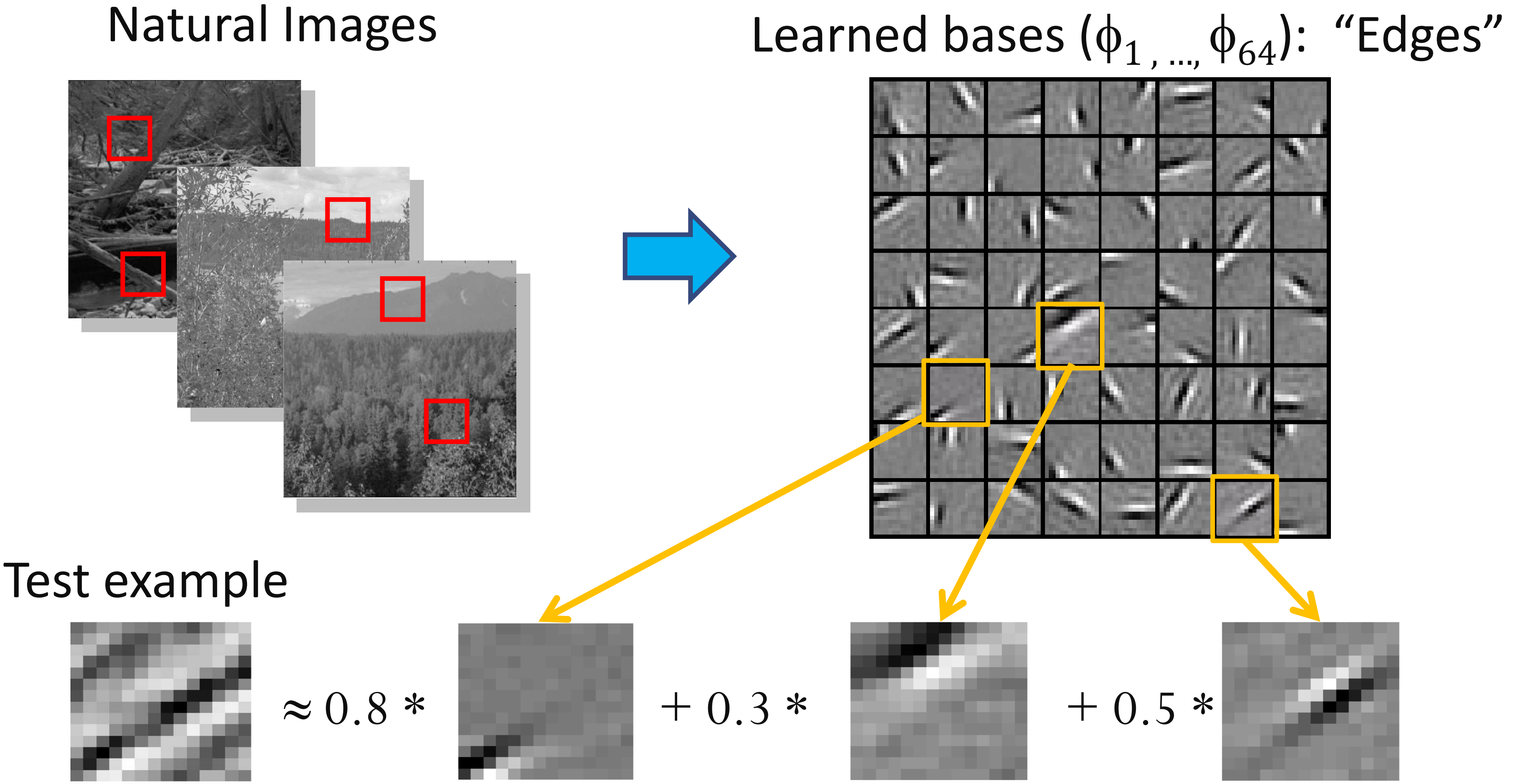
Motivation: Images as a composition of local parts
“Patch-based” representation



Example: 1000x1000 image

Credit: M. A. Ranzato

Motivation: Images as a composition of local parts
Sparse coding example



$$x \approx 0.8 * \phi_{36} + 0.3 * \phi_{42} + 0.5 * \phi_{63}$$

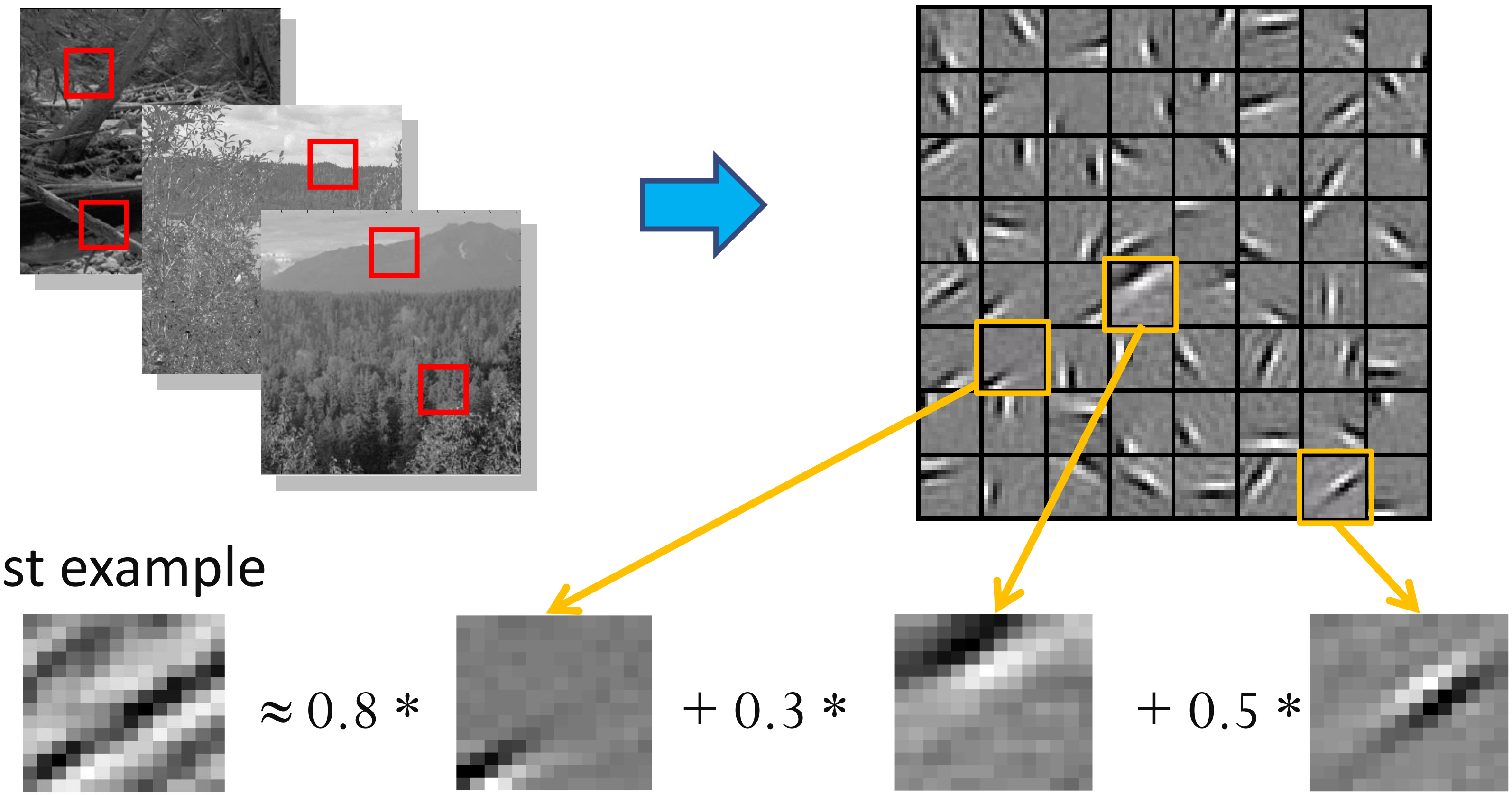
[a_1, …, a_{64}] = [0, 0, …, 0, **0.8**, 0, …, 0, **0.3**, 0, …, 0, **0.5**, 0]
(feature representation)

Credit: A. Coates

Motivation: Images as a composition of local parts

Sparse coding example

Natural Images — **Learned bases ($\phi_1, \ldots, \phi_{64}$): "Edges"**

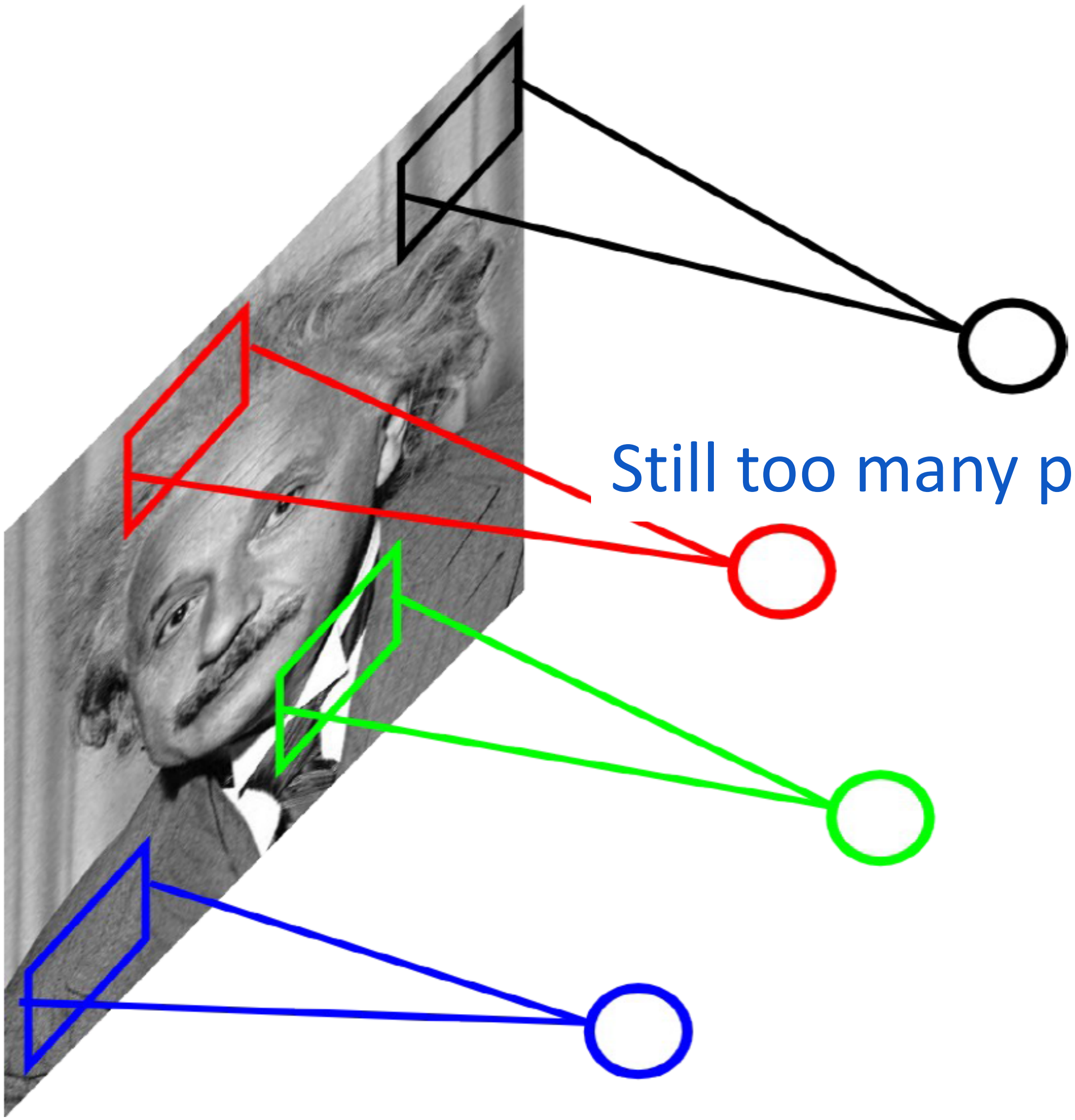


- Method "invents" edge detection
- Automatically learns to represent an image in terms of the edges that appear in it
- Gives a more succinct, higher-level representation than the raw pixels
- Quantitatively similar to primary visual cortex (area V1) in brain

Credit: A. Coates

Motivation: Images as a composition of local parts

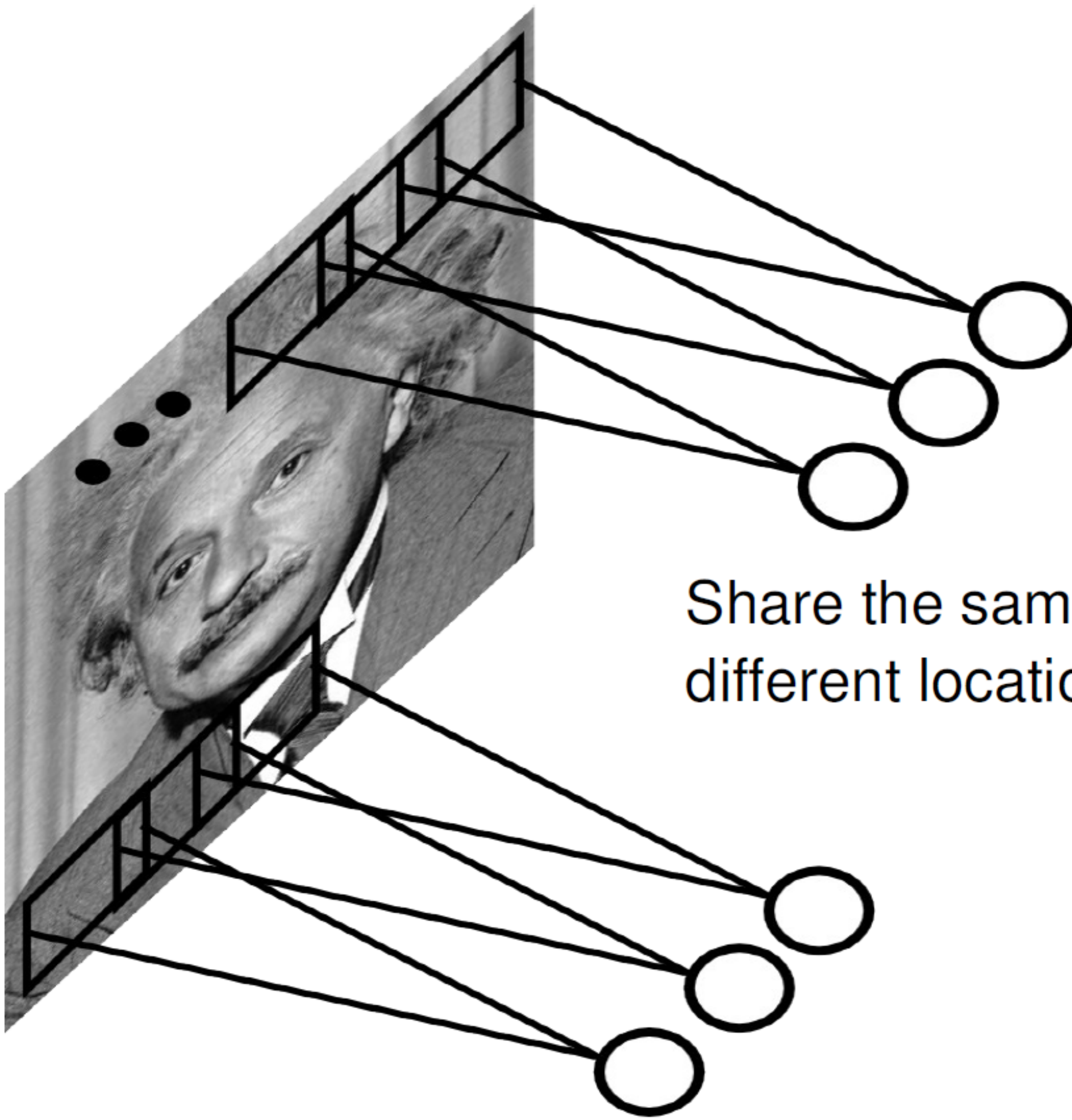
"Patch-based" representation



Still too many parameters!

Credit: M. A. Ranzato

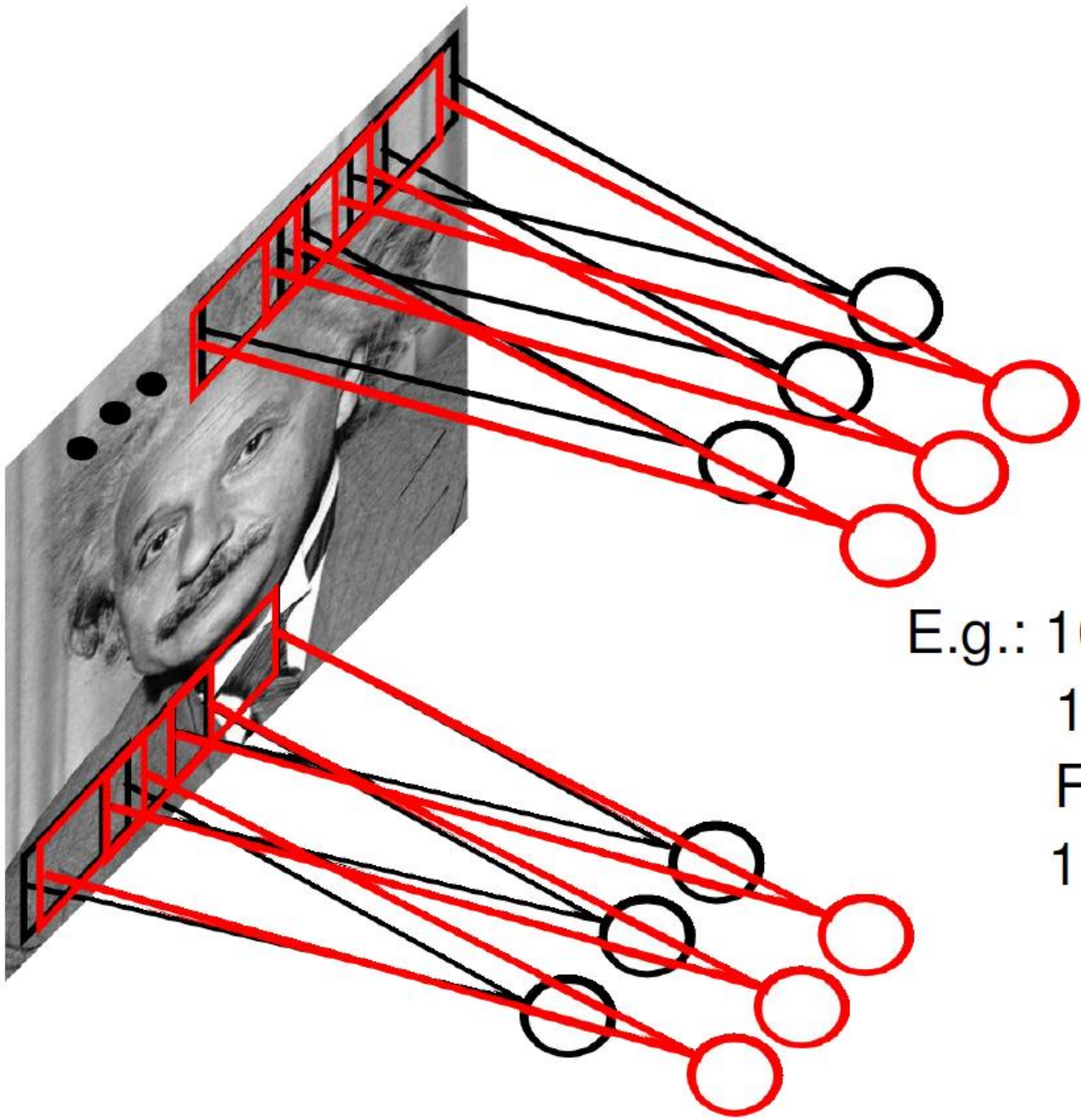
Motivation: Images as a composition of local parts
Convolution example



Share the same parameters across different locations:

Credit: M. A. Ranzato

Motivation: Images as a composition of local parts
Convolution example

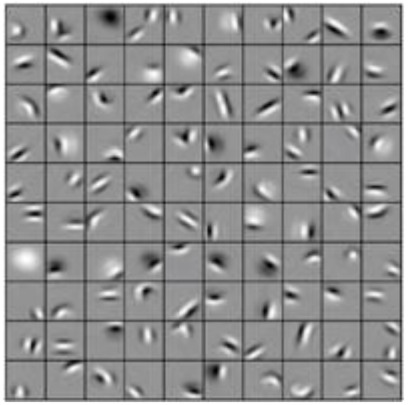


E.g.: 1000x1000 image
100 Filters
Filter size: 10x10
10K parameters

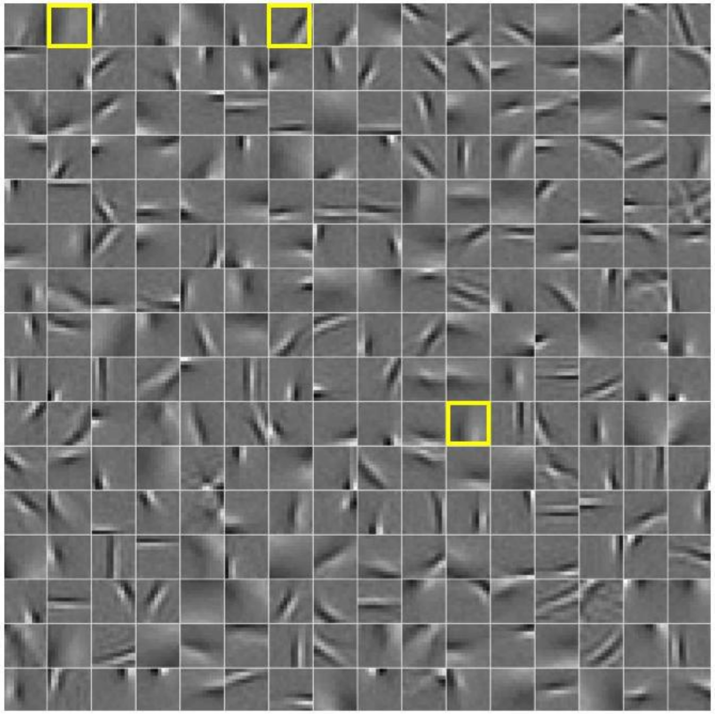
Credit: M. A. Ranzato

Motivation: Images as a composition of local parts
Filtering example

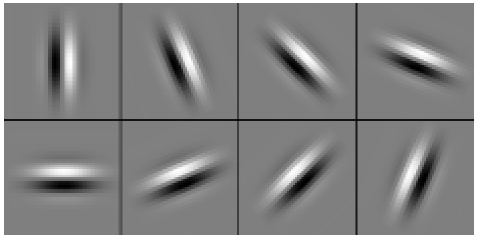
- Why translation *equivariance*?
  - Input translation leads to a translation of features
    - Fewer filters needed: no translated replications
    - But still need to cover orientation/frequency



Patch-based



Patch-based



Convolutional

Credit: R. Fergus

3x3 box average filter to blur an image (e.g., to remove noise)

$$h = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Numerical calculation: Smoothing via local averaging

$$g(x, y) = \sum_{k,l} f(x-k, y-l)h(k,l)$$

$f =$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$g =$

	0								

Credit: S. Seitz

Convolution: The 2D case

Numerical calculation: Smoothing via local averaging

$$h = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$g(x,y) = \sum_{k,l} f(x-k, y-l)h(k,l)$$

$f =$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$g =$

	0								

Credit: S. Seitz

Convolution: The 2D case

Numerical calculation: Smoothing via local averaging

$$h = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$g(x,y) = \sum_{k,l} f(x-k, y-l)h(k,l)$$

$f =$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$g =$

	0	10							

Credit: S. Seitz

Convolution: The 2D case

Numerical calculation: Smoothing via local averaging

$$h = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$g(x,y) = \sum_{k,l} f(x-k, y-l)h(k,l)$$

$f =$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$g =$

	0	10	20						

Credit: S. Seitz

Convolution: The 2D case

Numerical calculation: Smoothing via local averaging

$$h = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$g(x,y) = \sum_{k,l} f(x-k, y-l)h(k,l)$$

$f =$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$g =$

	0	10	20	30					

Credit: S. Seitz

Convolution: The 2D case

Numerical calculation: Smoothing via local averaging

$$h = \frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

$$g(x,y) = \sum_{k,l} f(x-k, y-l)h(k,l)$$

$f =$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$g =$

	0	10	20	30	30				

Credit: S. Seitz

Convolution: The 2D case

Numerical calculation: Smoothing via local averaging

$$h = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$g(x,y) = \sum_{k,l} f(x-k, y-l)h(k,l)$$

$f =$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$g =$

	0	10	20	30	30	30			

Credit: S. Seitz

Convolution: The 2D case

Numerical calculation: Smoothing via local averaging

$$h = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$g(x,y) = \sum_{k,l} f(x-k, y-l)h(k,l)$$

$f =$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$g =$

	0	10	20	30	30	30			
				?					

Credit: S. Seitz

Convolution: The 2D case

Numerical calculation: Smoothing via local averaging

$$h = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$g(x,y) = \sum_{k,l} f(x-k, y-l)\,h(k,l)$$

$f =$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$g =$

	0	10	20	30	30	30			
				50					

Credit: S. Seitz

Think of the filter as a feature detector now (e.g., how smooth is a region?)

$$h = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Numerical calculation: Smoothing via local averaging

$$g(x,y) = \sum_{k,l} f(x-k, y-l)h(k,l)$$

$f =$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$g =$

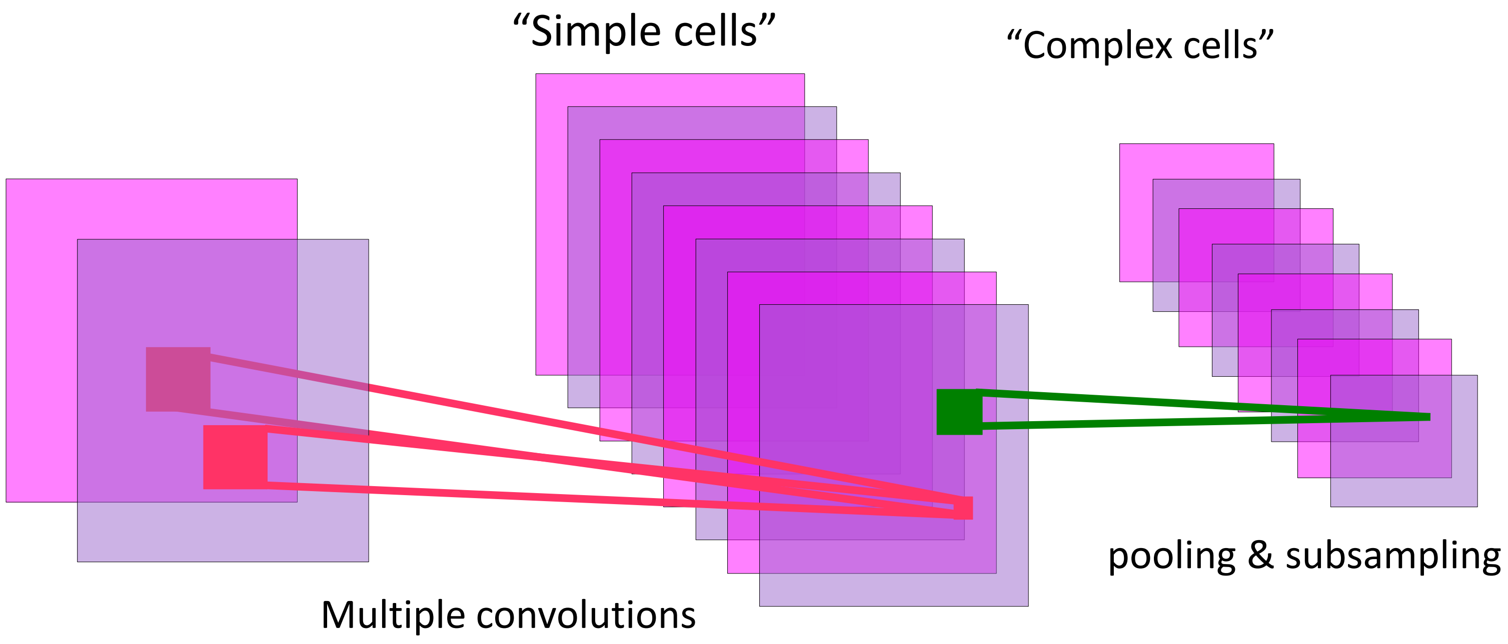
	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

Credit: S. Seitz

Convolutional Neural Networks

Multistage HubelWiesel Architecture:
An Old Idea for Local Shift Invariance

- [Hubel & Wiesel 1962]
  - Simple cells detect local features
  - Complex cells "pool" the outputs of simple cells within a retinotopic neighborhood.



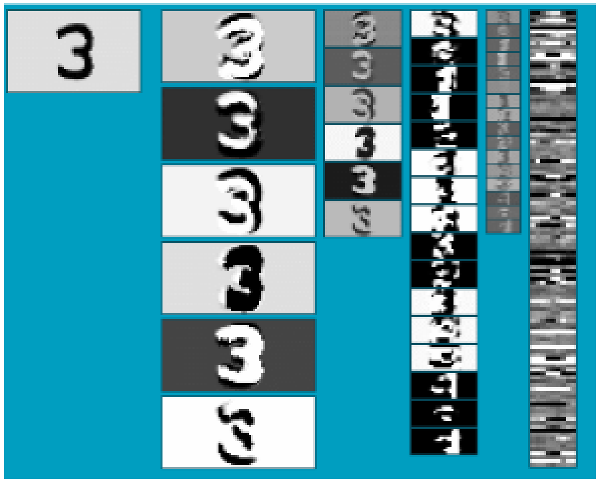
Convolutional Networks
[LeCun 1988-present]

Retinotopic Feature Maps

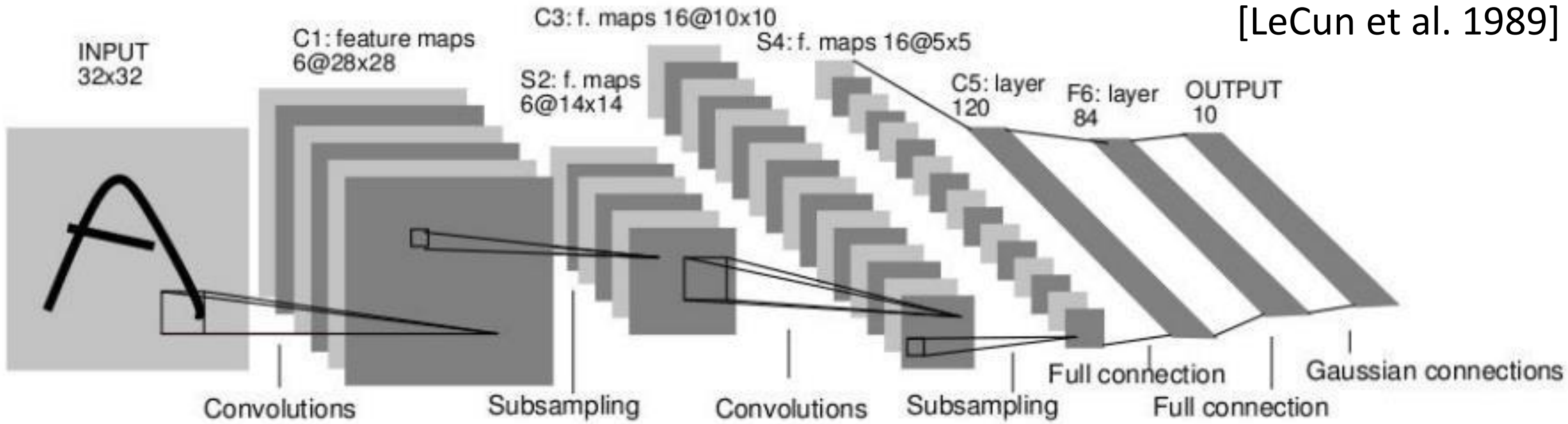
Credit: Y. LeCun

Convolutional Neural Networks

- Neural network with specialized connectivity structure
- After a few convolution and subsampling stages, spatial resolution is very small
- Use fully connected layers up to classification at output layer



[LeCun et al. 1989]

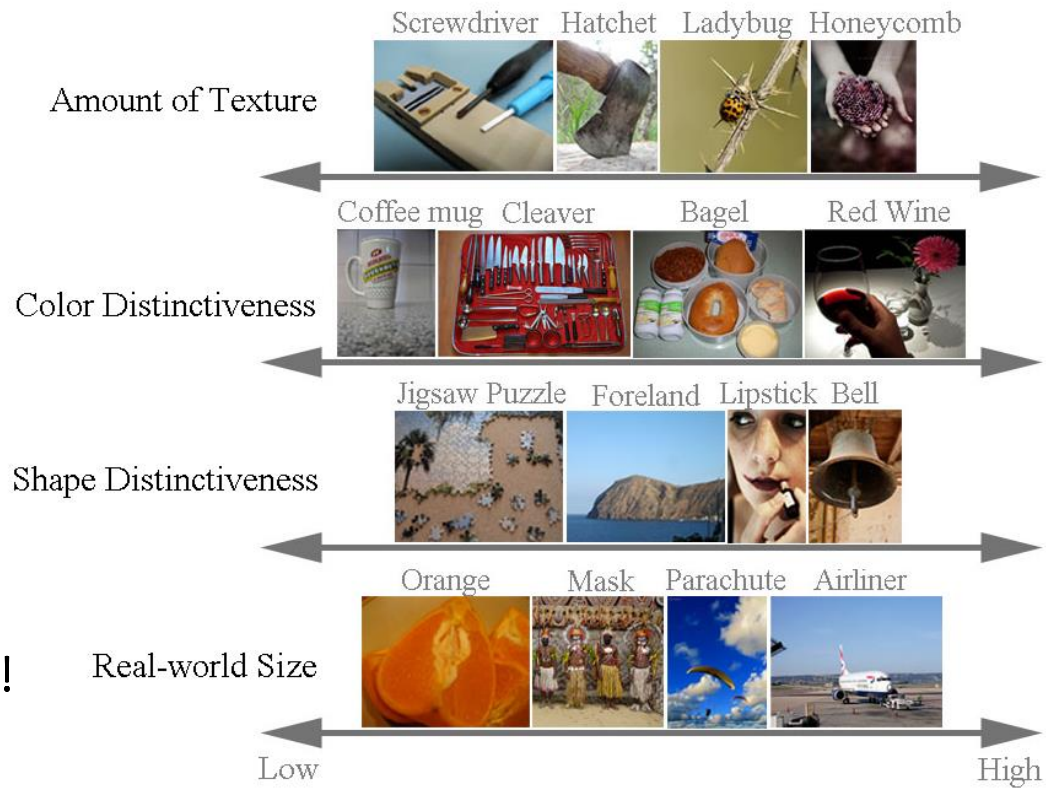


[LeCun et al. 1989]

Training large ConvNets on IMAGENET

Key ingredients for CNNs

- **Large annotated dataset**
- Strong regularization (dropout)
- GPU(s) for fast processing
  - ~ 150 images/sec
  - days—weeks of training



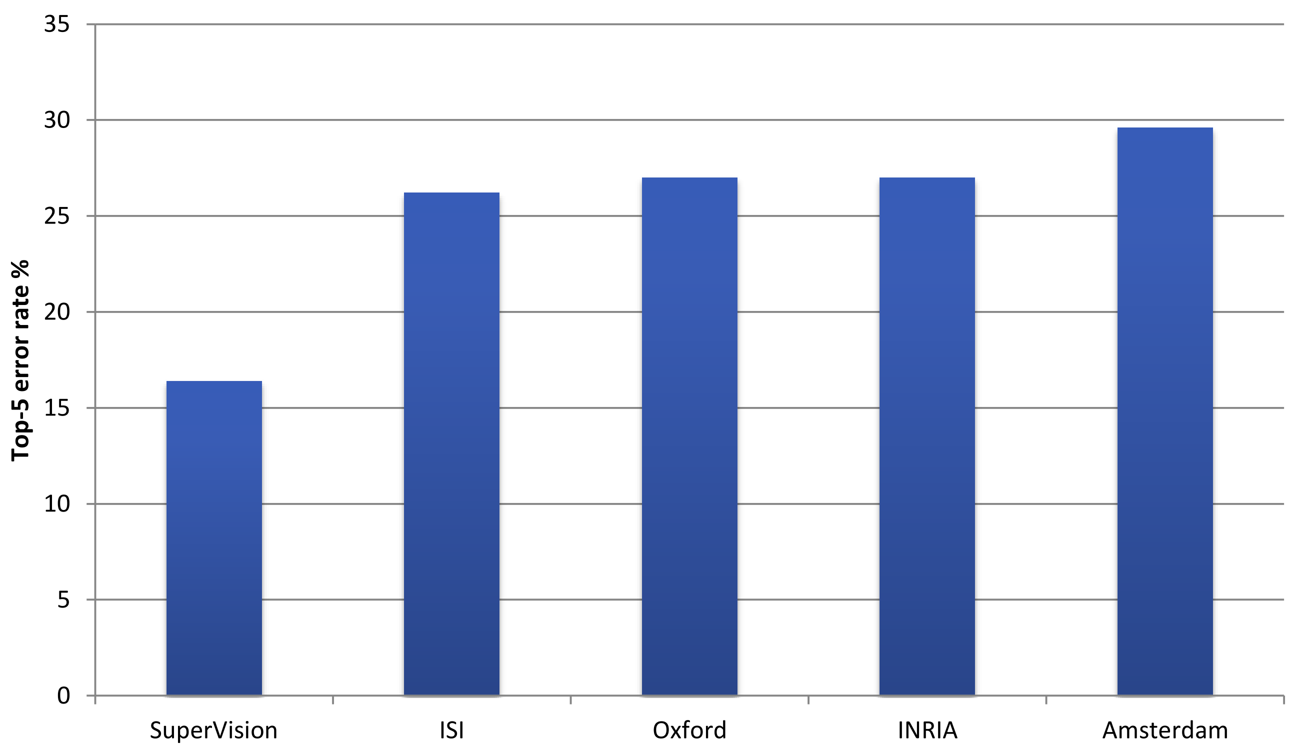
[Russakovsky et al. ICCV'13]

Imagenet database:

- 1K classes
- ~ 1K training images per class!
- ~ 1M training images

ImageNet Classification 2012

- Krizhevsky et al. -- 16.4% error (top-5)
- Next best (non-convnet) – 26.2% error



Credit: R. Fergus

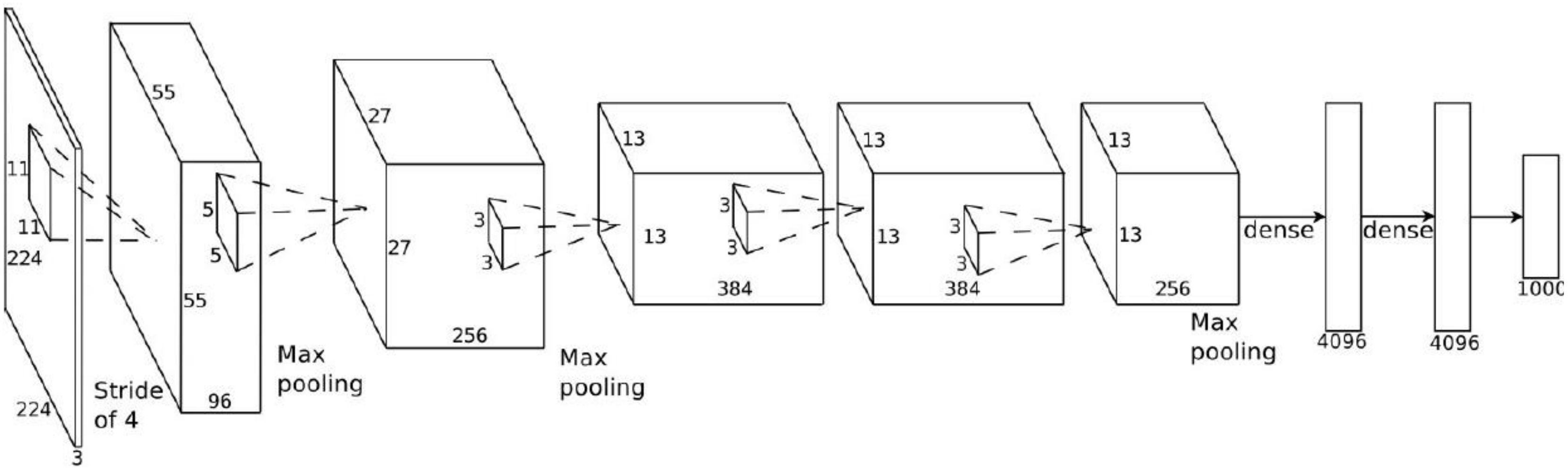
Architecture of [Krizhevsky et al. NIPS'12]

ImageNet Classification 2012

- 16.4% error (top-5)
- next best (variant of SIFT + Fisher Vectors) – 26.2% error

Same idea as in [LeCun'98] but on a larger scale:

- more training images (10^6 vs 10^3)
- more hidden layers

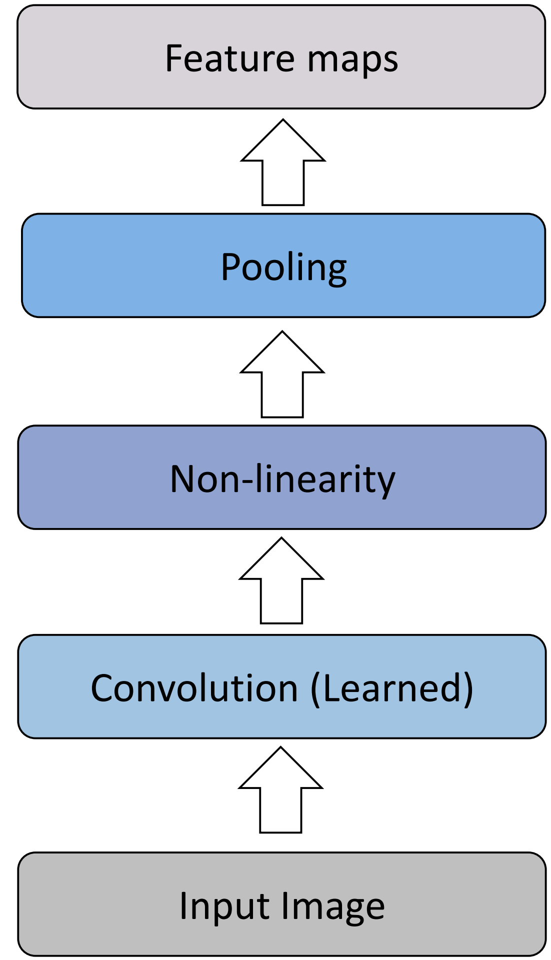


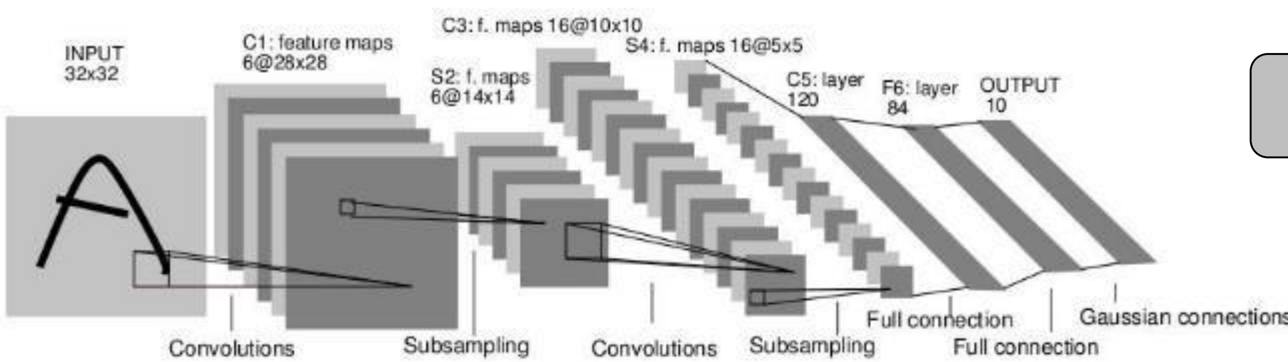
Overall: 60,000,000 parameters which are trained on 2 GPUs for a week with several tricks to reduce overfitting

- Data augmentation
- DropOut (new regularization)

ConvNet Architecture

- Feed-forward:
  - Convolve input
  - Non-linearity (rectified linear)
  - Pooling (local max)
- Supervised
- Train convolutional filters by back-propagating classification error





Credit: R. Fergus

Components of Each Layer

Pixels /

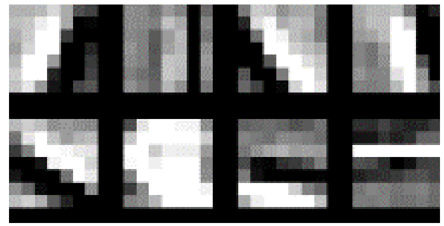
Features

Filter with

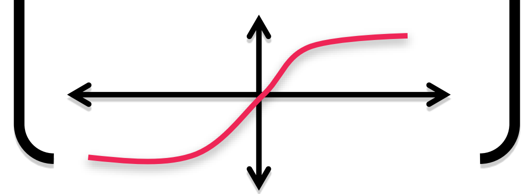
Dictionary

(convolutional

or tiled)

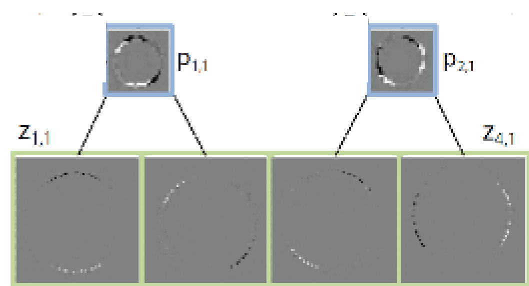


+ Non-linearity



Spatial/Feature

(Sum or Max)



[Optional]

Normalization

between

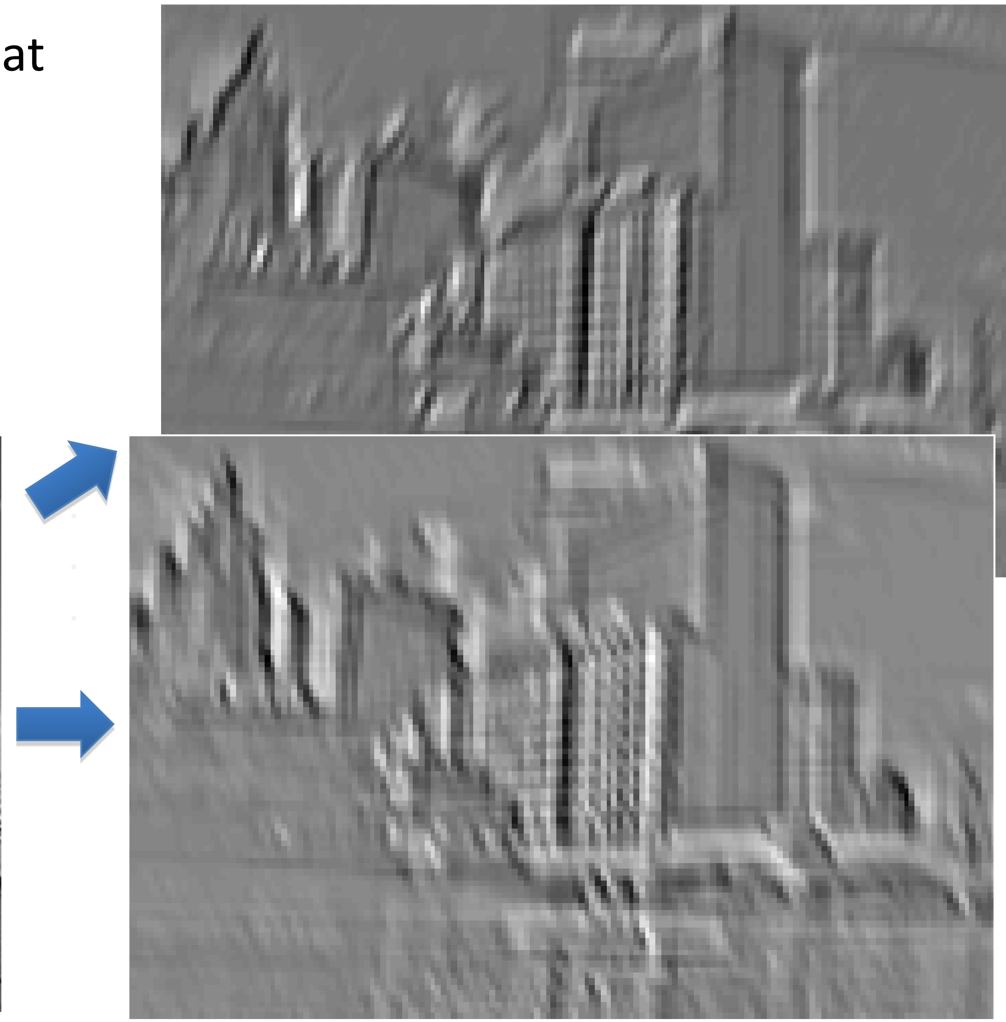
feature responses

Output Features

Credit: R. Fergus

Convolutional filtering

- Why convolution?
  - Statistics of images look similar at different locations
  - Dependencies are very local
  - Filtering is an opteration with translation *equivariance*

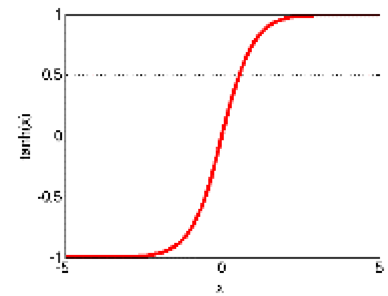


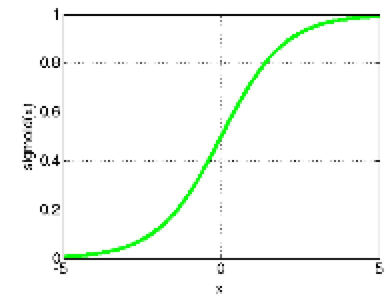
Credit: R. Fergus

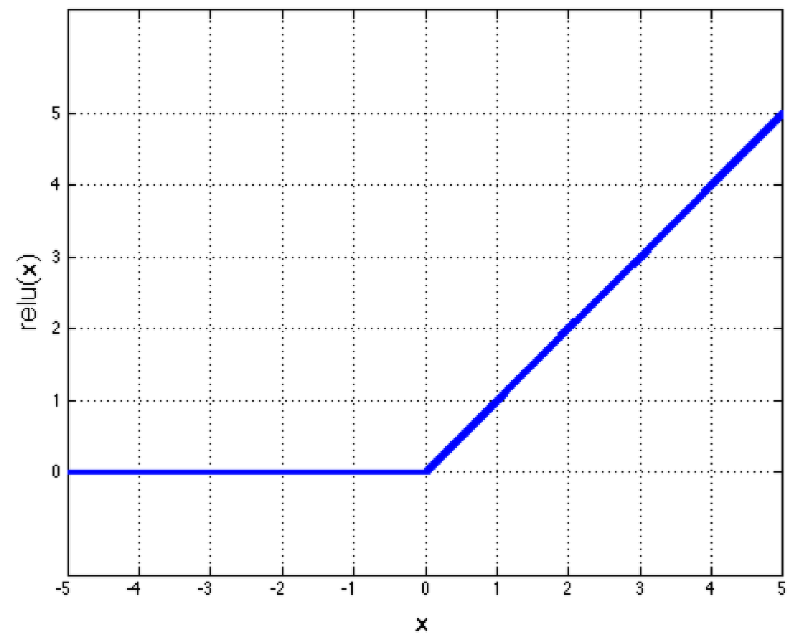
Non-Linearity

- Non-linearity applied to each response
  - Per-feature independent
  - Tanh
  - Sigmoid: 1/(1+exp(-x))
  - Rectified linear : max(0,x)
    - Simplifies backprop
    - Makes learning faster
    - Avoids saturation issues (much higher dynamic range)

  → Preferred option



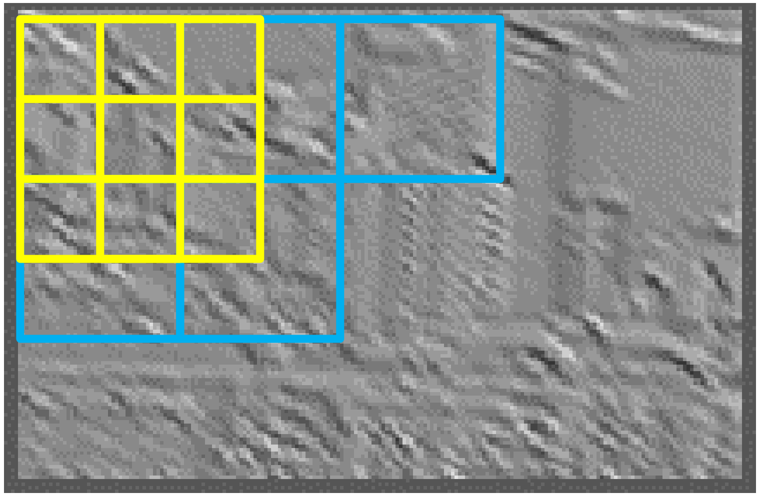




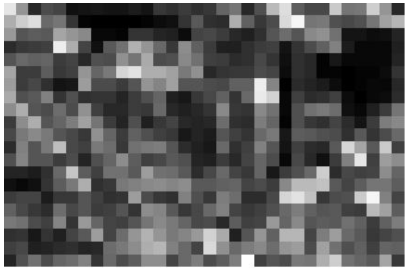
Credit: R. Fergus

Pooling

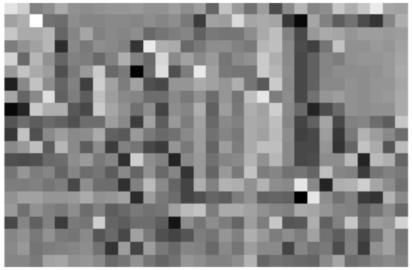
- Spatial Pooling
  - Non-overlapping / overlapping regions (-0.3% in error)
  - Sum or max
  - Provides invariance to local transformations



Max



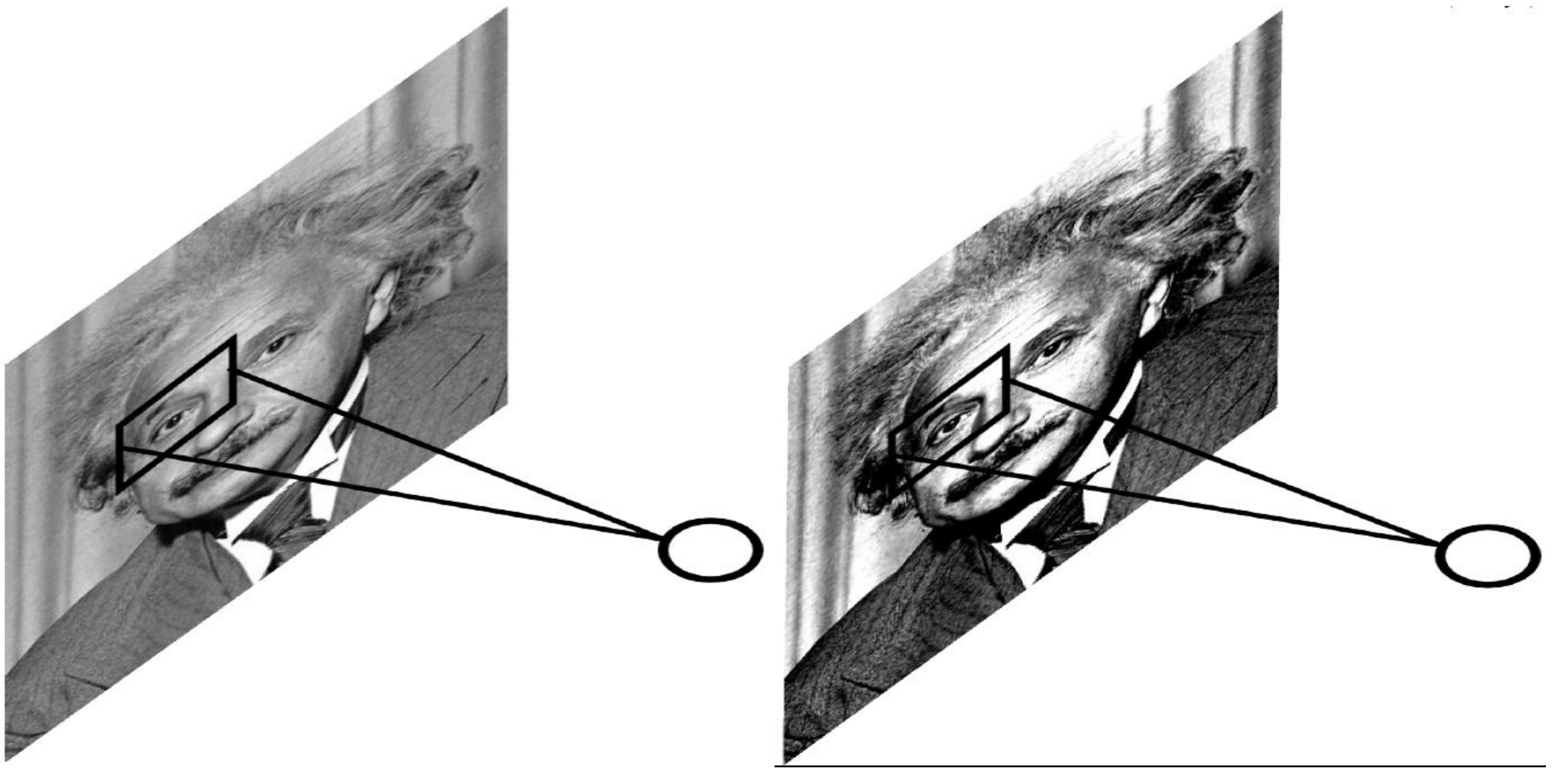
Sum



Credit: R. Fergus

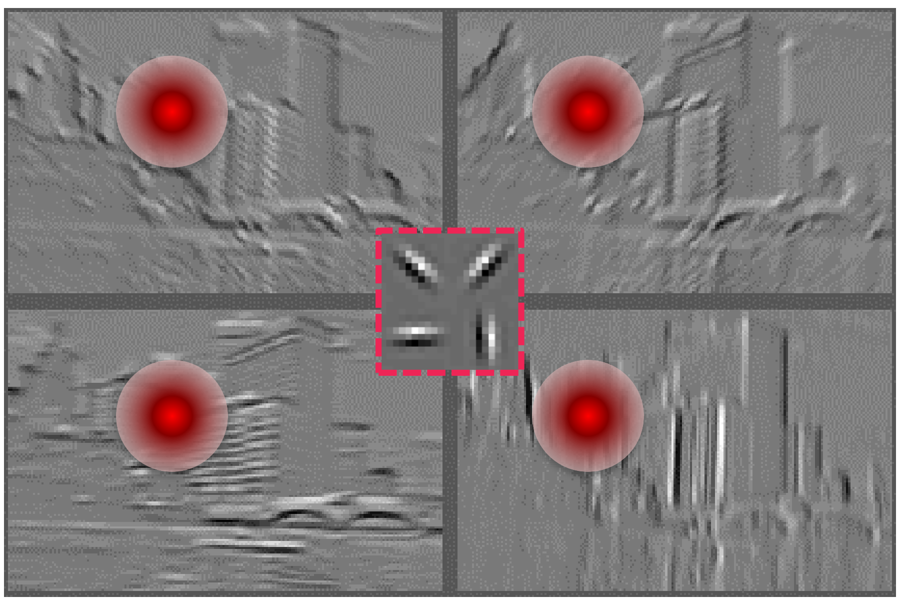
Normalization

- Contrast normalization

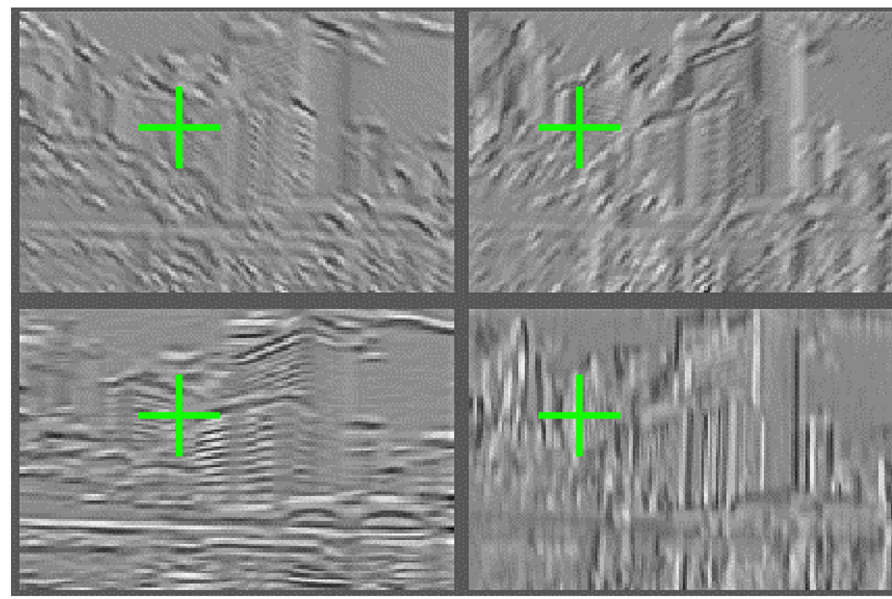


The same response for different contrasts is desired

- Contrast nor eature maps)
  - Local mean Gaussian
  - Equalizes th



Feature Maps



Feature Maps
After Contrast Normalization

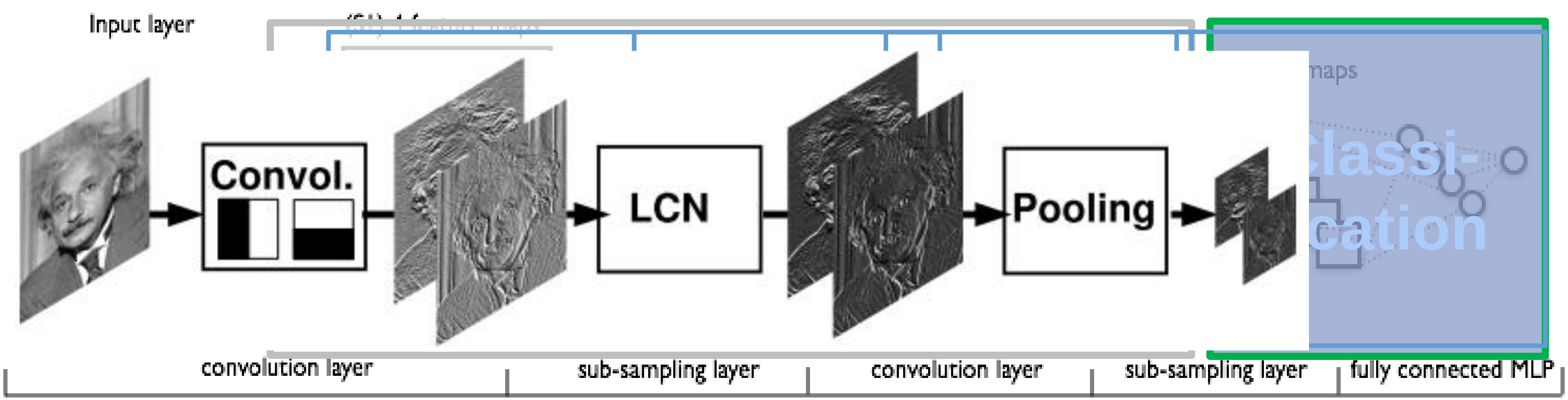
Credit: R. Fergus

Recap: Components of a CNN

CNN - multi-layer NN architecture

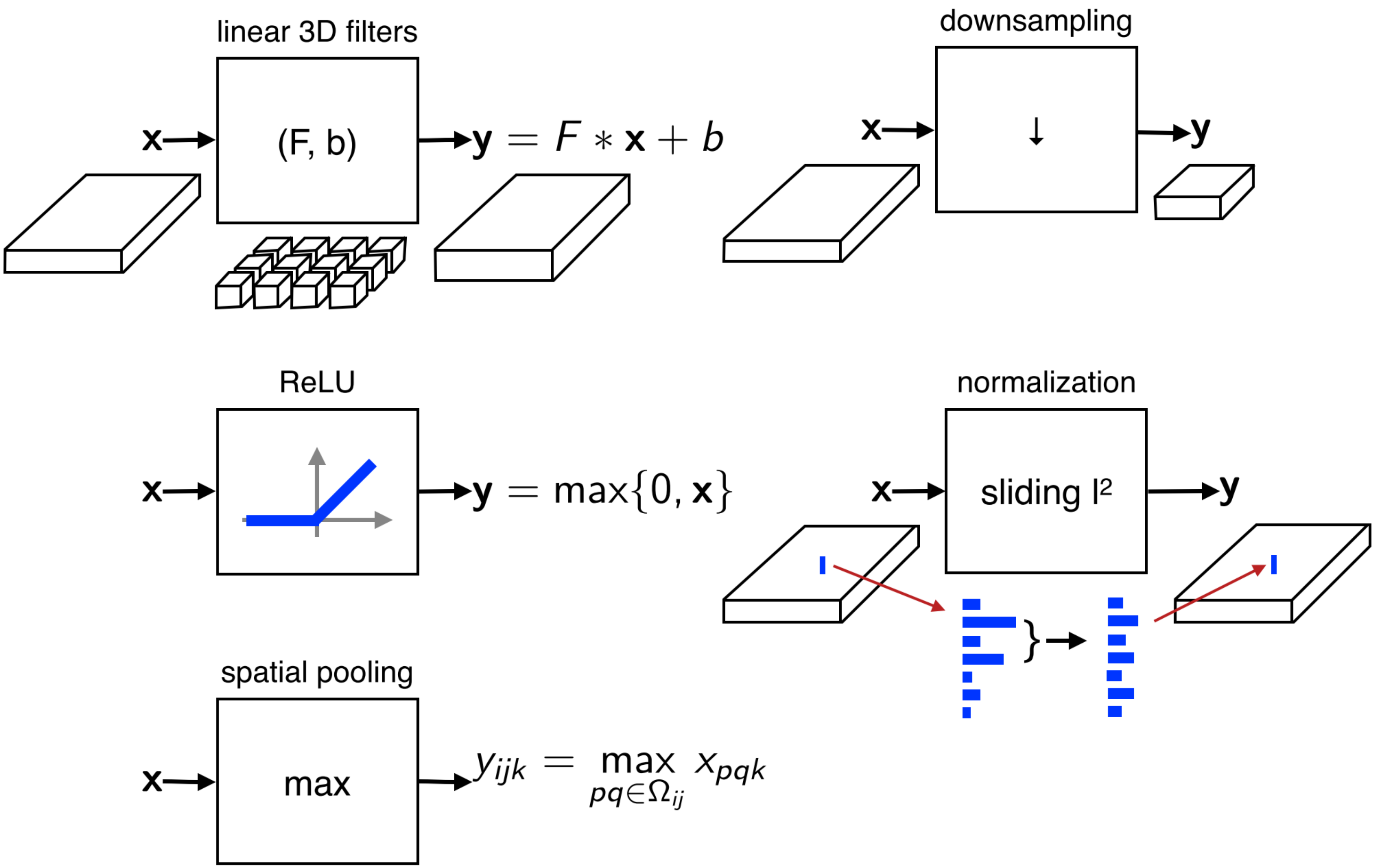
- Convolutional + Non-Linear Layer
- Sub-sampling Layer
- Convolutional +Non-Linear Layer
- Fully connected layers

- Supervised



Credit: B. Ginzburg

Summary: Components of Each Layer



Credit: A. Vedaldi

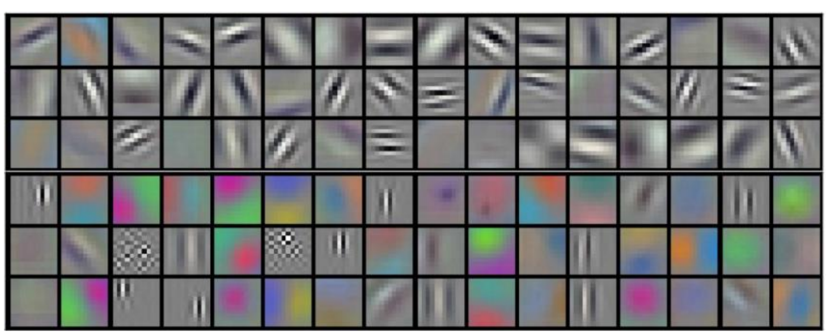
Architecture of Krizhevsky et al.

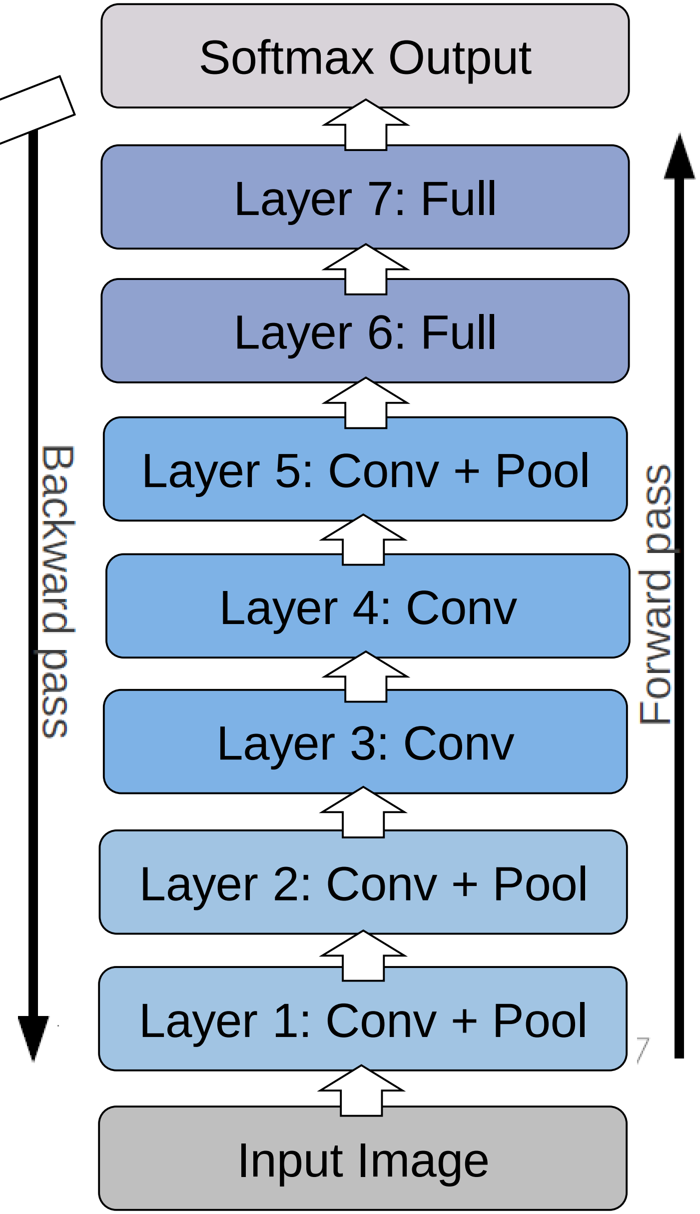
One output unit per class

x_i = total input to output unit i

$$f(x_i) = \frac{\exp(x_i)}{\sum_{j=1}^{1000} \exp(x_j)}$$

- Trained via backprop by maximizing the log-prob. of the correct class-label
- 8 layers total
- Trained on ImageNet dataset





Credit: R. Fergus

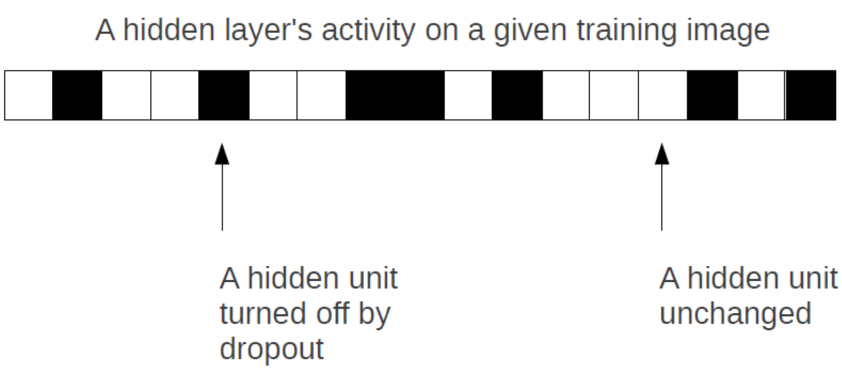
Improving Generalization: DropOut

[Hinton et al. NIPS'12]

Motivation:

- Random Forests generalize well due to averaging of many models
- Decision Trees are fast - ConvNets are slow – many models are not feasible

- Similar to random forest bagging [Breiman'94], but differs in that parameters are shared
- For fully connected layers only:
  - In training: Independently set each hidden unit activity to zero with 0.5 probability
  - In testing multiply neuron output by 0.5

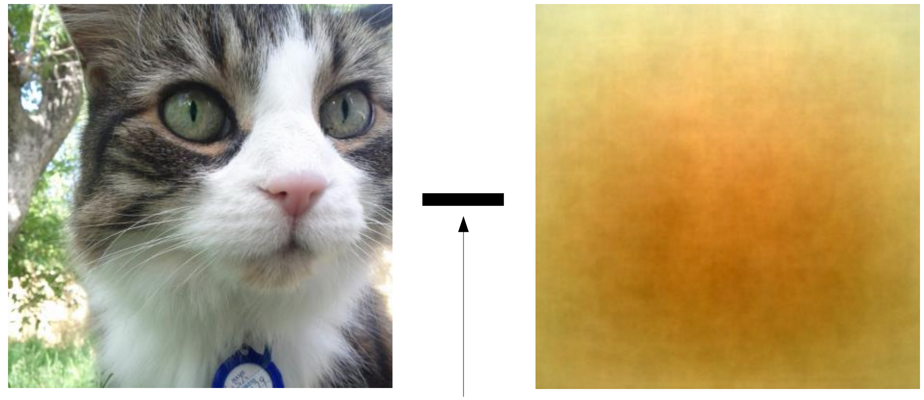


- Corresponds to averaging over exponentially many samples from different model architectures

Further pre-processing tricks

- Mean removal

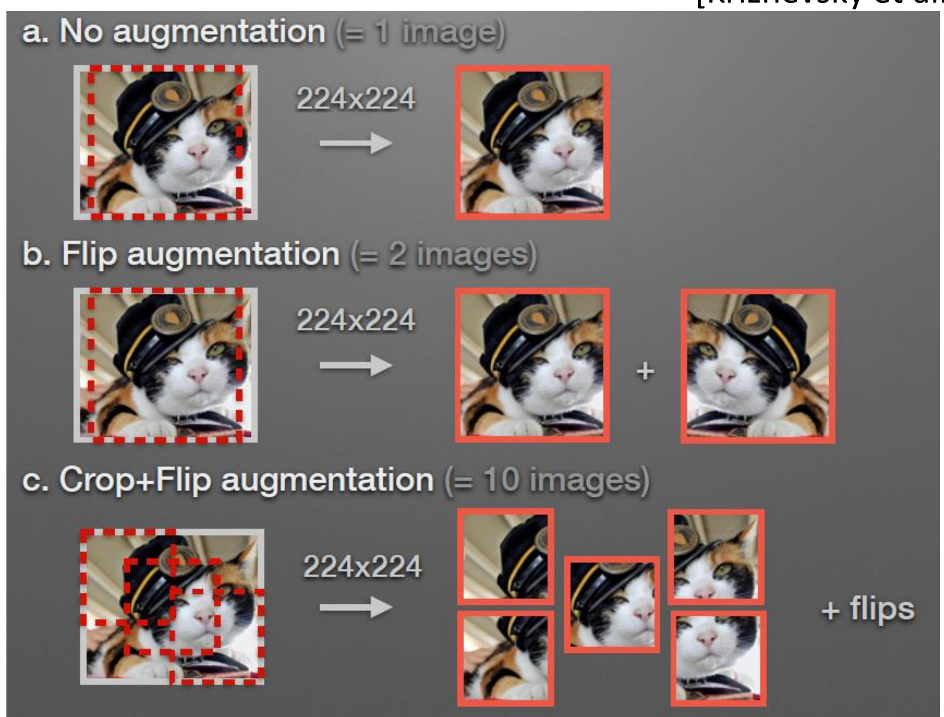
Centered (0-mean) RGB values.



[Krizhevsky et al. NIPS'12]

- Data augmentation

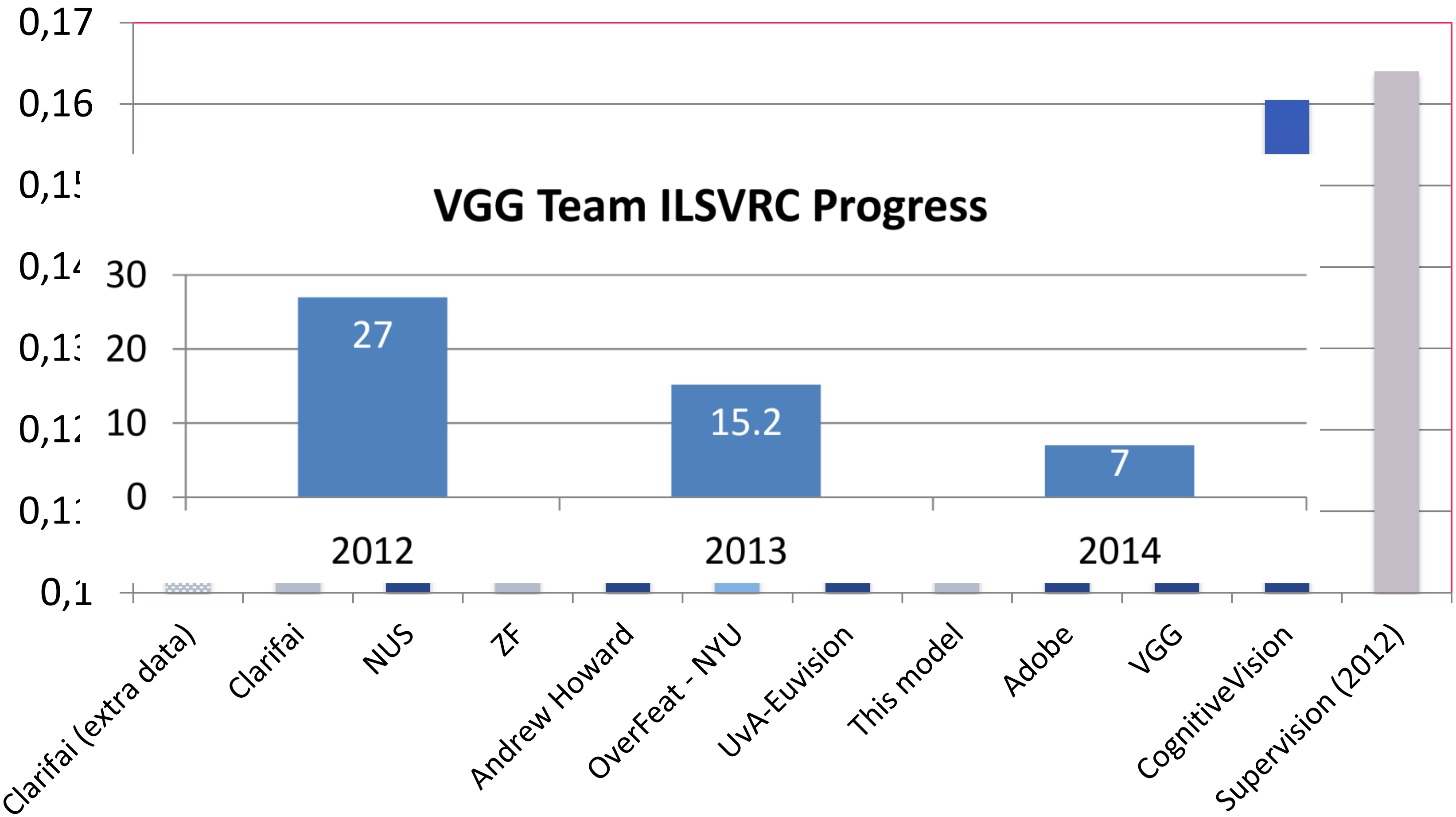
Train on 224x224 patches extracted randomly from images, and also their horizontal reflections



[Chatfield et al. BMVC'14]

ImageNet Classification 2013/2014 Results

- http://www.image-net.org/challenges/LSVRC/2013/results.php

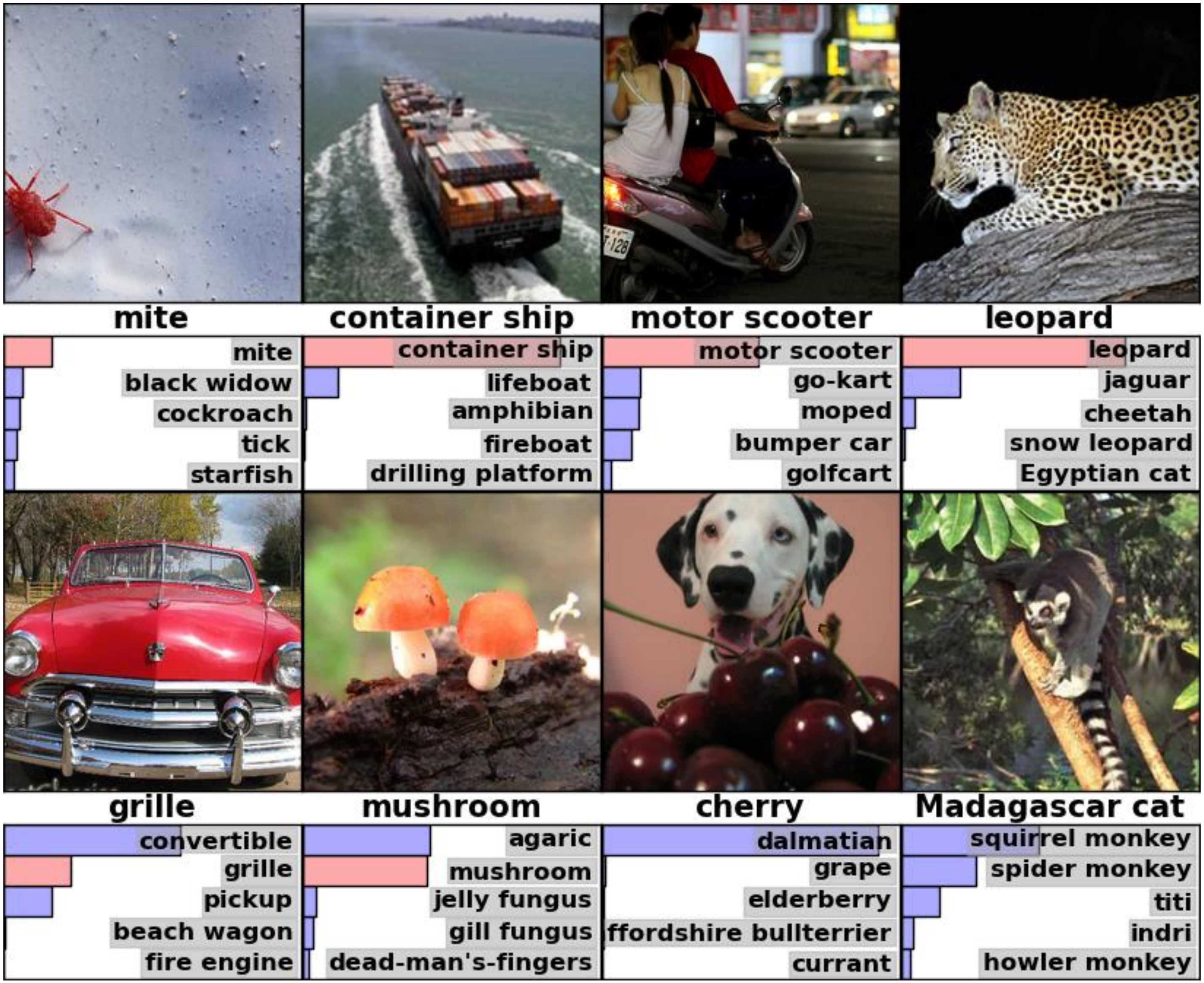


- Pre-2012: 26.2% error → 2012: 16.5% error → 2013: 11.2% error

Credit: R. Fergus

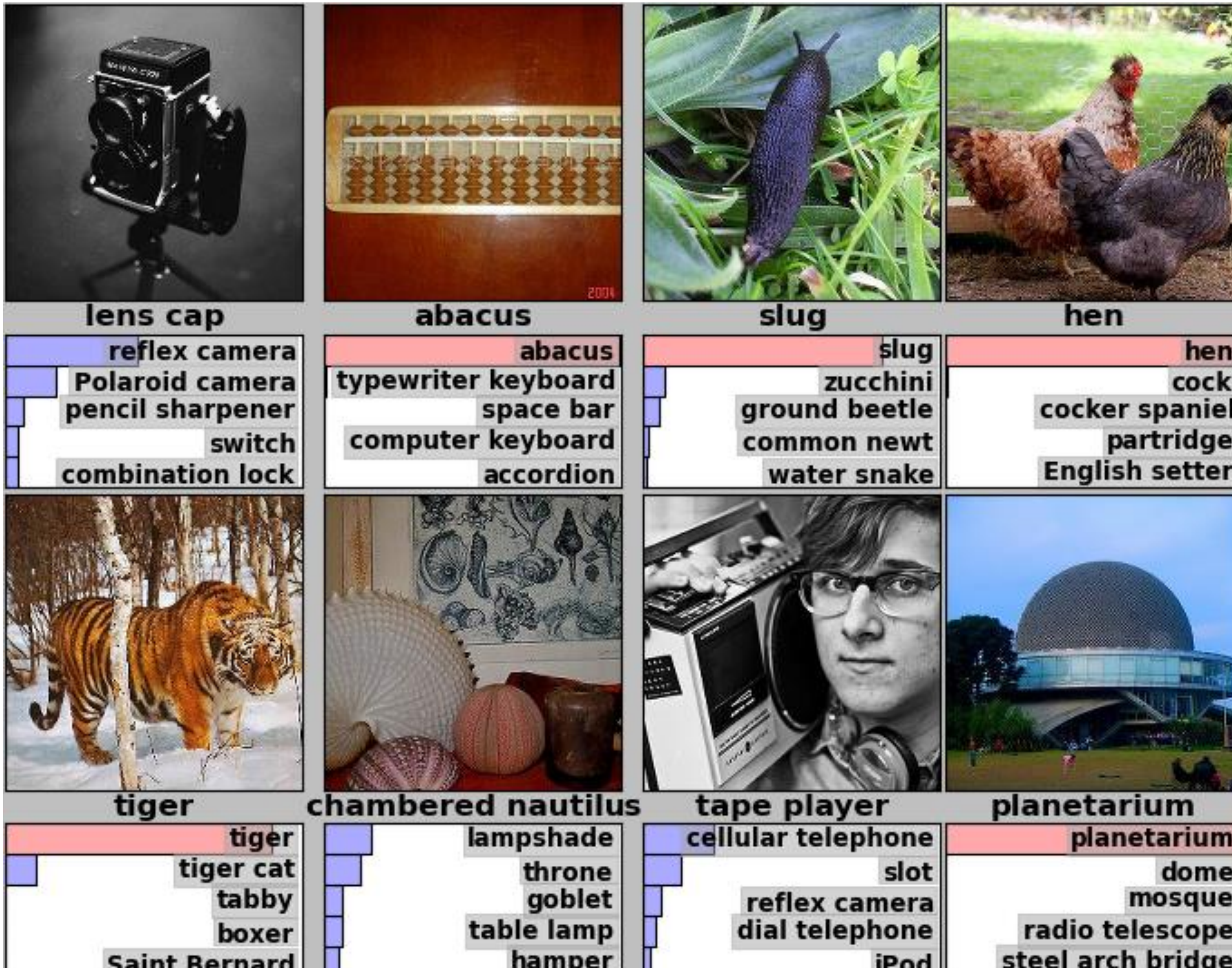
ImageNet Sample classifications

[Krizhevsky et al. NIPS'12]



ImageNet Sample classifications

[Krizhevsky et al. NIPS'12]



ImageNet Classification Progress from '12-'14

2012 Teams	%error
Supervision (Toronto)	15.3
ISI (Tokyo)	26.1
VGG (Oxford)	26.9
XRCE/INRIA	27.0
UvA (Amsterdam)	29.6
INRIA/LEAR	33.4

2013 Teams	%error
Clarifai (NYU spinoff)	11.7
NUS (singapore)	12.9
Zeiler-Fergus (NYU)	13.5
A. Howard	13.5
OverFeat (NYU)	14.1
UvA (Amsterdam)	14.2
Adobe	15.2
VGG (Oxford)	15.2
VGG (Oxford)	23.0

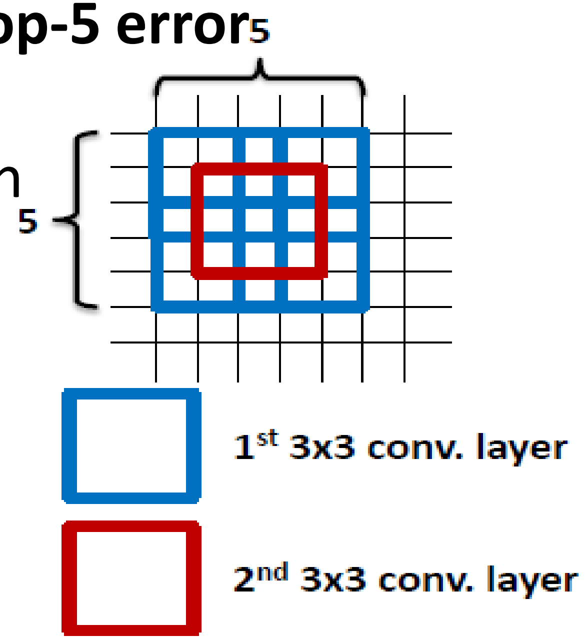
2014 Teams	%error
GoogLeNet	6.6
VGG (Oxford)	7.3
MSRA	8.0
A. Howard	8.1
DeeperVision	9.5
NUS-BST	9.7
TTIC-ECP	10.2
XYZ	11.2
UvA	12.1

Credit: Y. LeCun

Better (≈deeper) architectures exist now

ILSVRC14 Winners: **~7.3% Top-5 error**

– VGG: 16 layers of stacked 3x3 convolution with stride 1



Other details:

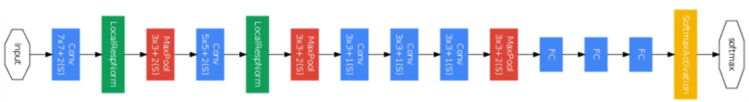
- Rectification (ReLU) non-linearity
- 5 max-pool layers (x2 reduction)
- no normalisation
- 3 fully-connected (FC) layers

Credit: K. Simonyan

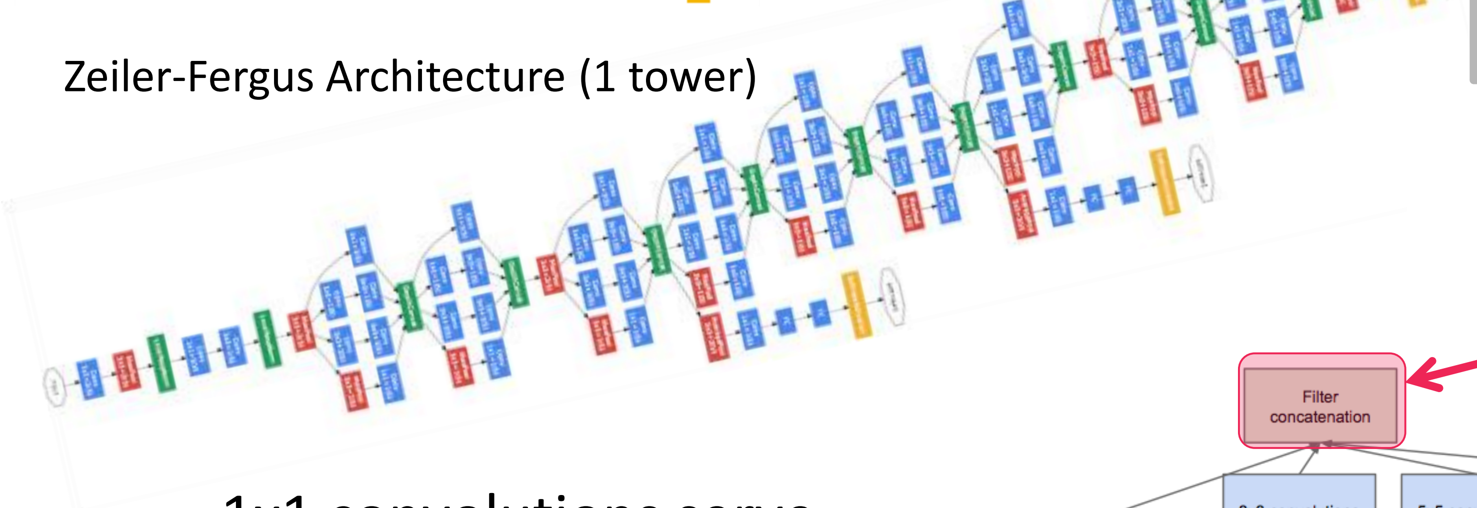
Better (≈deeper) architectures exist now

ILSVRC14 Winners: **~6.6% Top-5 error**

- GoogLeNet: composition of multi-scale dimension-reduced modules:

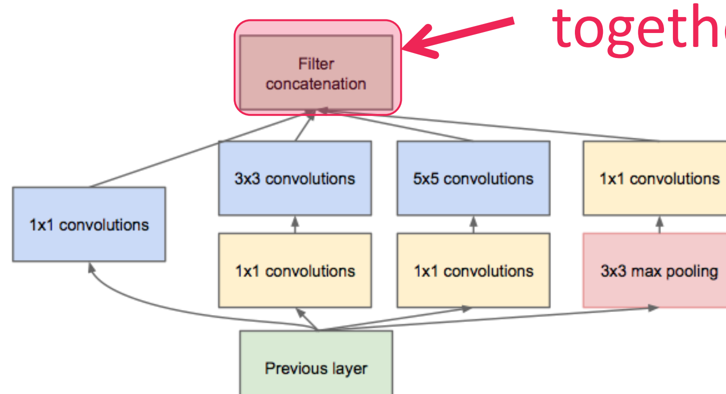


Zeiler-Fergus Architecture (1 tower)



Convolution
Pooling
Softmax
Other

"wire together what fires together"



- 1x1 convolutions serve as dimensionality reduction

Credit: C. Szegedy

Classification failure cases

Groundtruth: **coffee mug**

GoogLeNet:

- **table lamp**
- **lamp shade**
- **printer**
- **projector**
- **desktop computer**

Credit: C. Szegedy

Classification failure cases

Groundtruth: **hay**

GoogLeNet:

- **sorrel (horse)**
- **hartebeest**
- **Arabian camel**
- **warthog**
- **gaselle**

Credit: C. Szegedy

Classification failure cases

Groundtruth: **Police car**

GoogLeNet:

- **laptop**
- **hair drier**
- **binocular**
- **ATM machine**
- **seat belt**

Credit: C. Szegedy

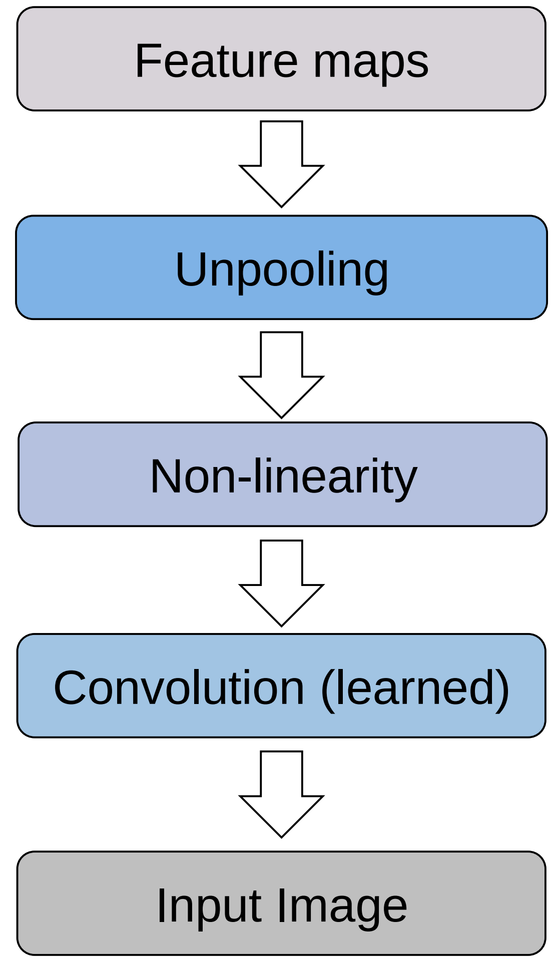
What is learned?
Visualizing CNNs

M. Zeiler & R. Fergus, *Visualizing and Understanding Convolutional Networks*, ECCV, 2014

Visualization using Deconvolutional Networks

[Zeiler et al. CVPR'10, ICCV'11, ECCV'14]

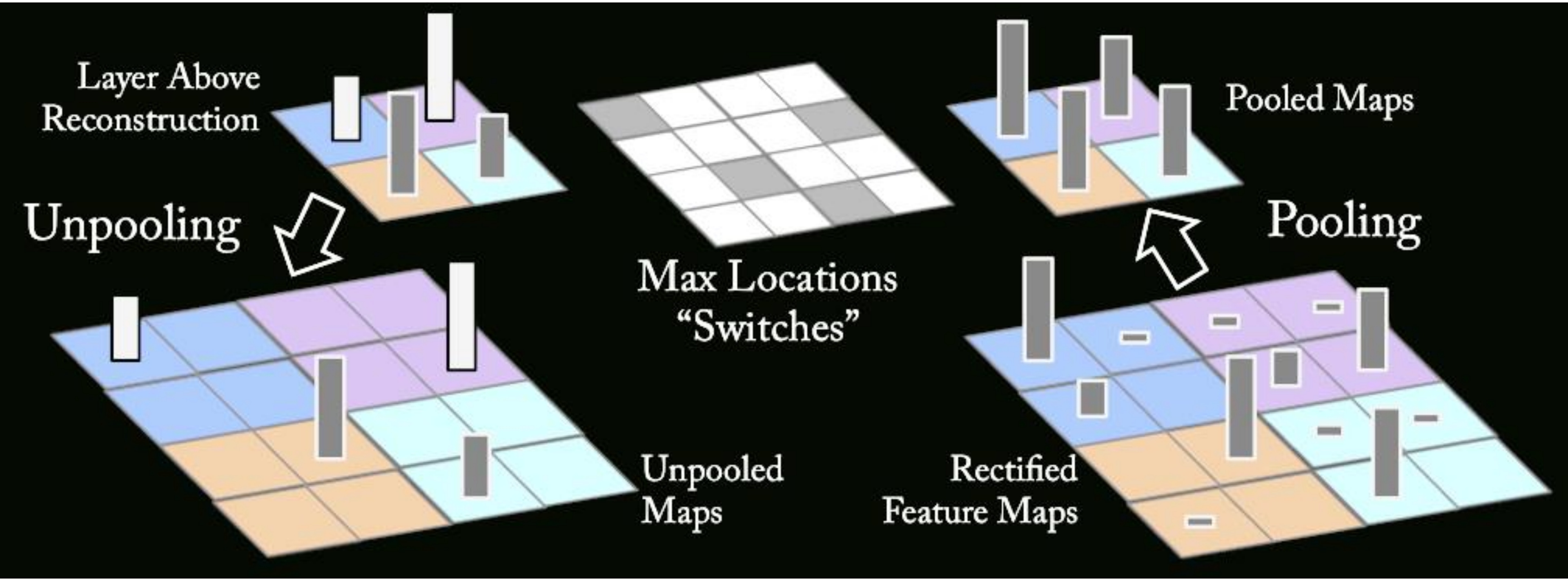
- Provides way to map activations at high layers back to the input
- Same operations as Convnet, but in reverse:
  - Unpool feature maps
  - Convolve unpooled maps
    - Filters copied from Convnet
- Used here purely as a probe
  - Originally proposed as unsupervised learning method
  - No inference, no learning



Credit: R. Fergus

Unpooling Operation

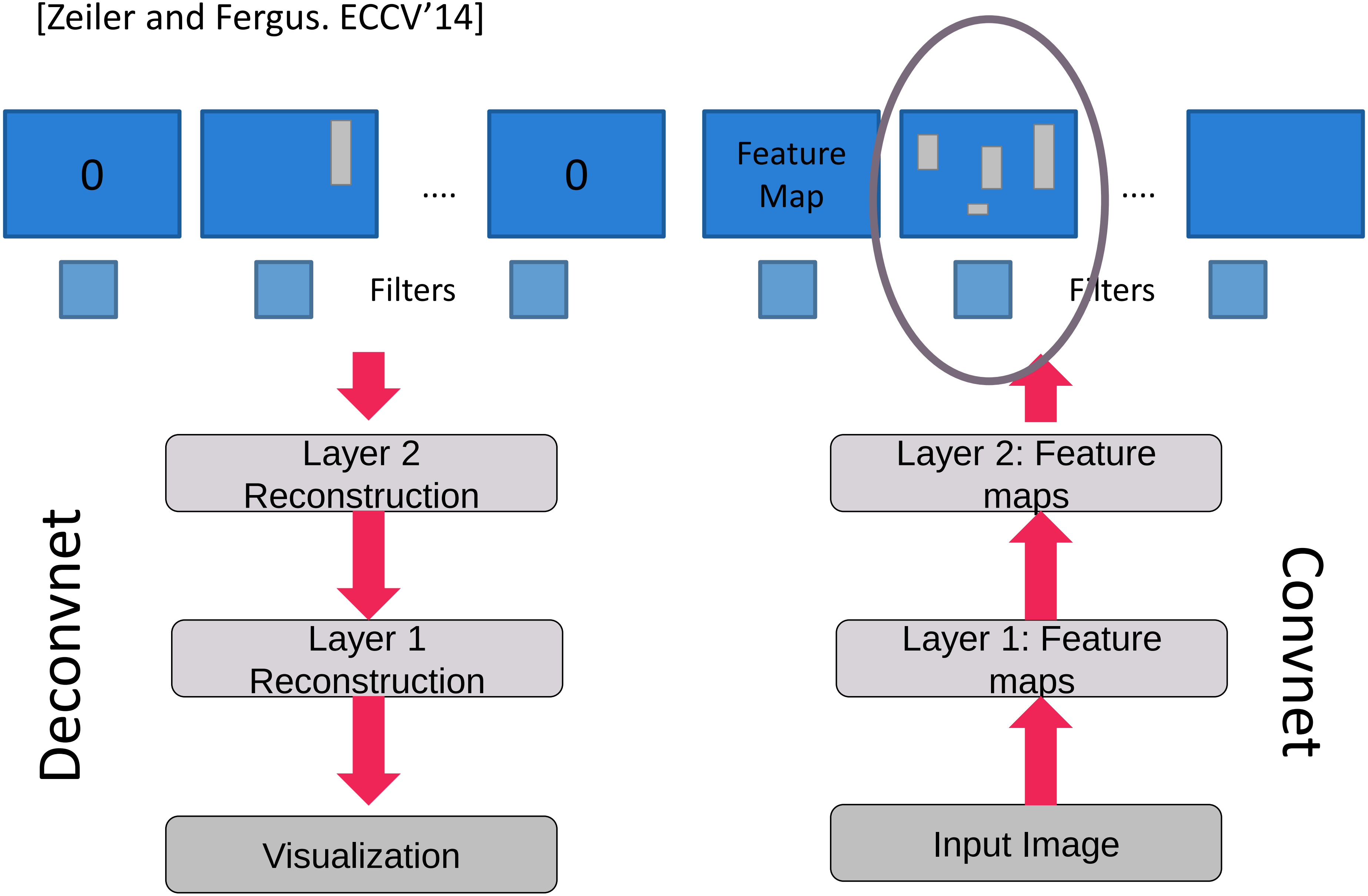
- Switches record where the pooled activations came from to "unpool" the reconstructed layer above



Credit: R. Fergus

Deconvnet Projection from Higher Layers

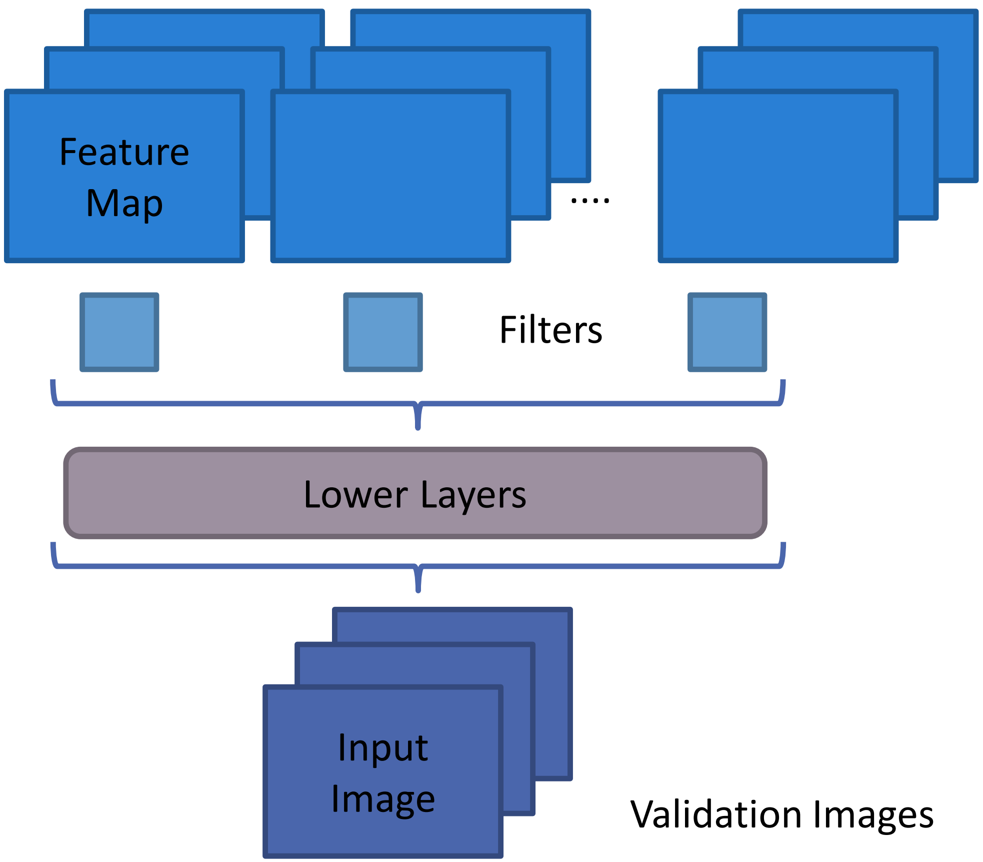
[Zeiler and Fergus. ECCV'14]



Credit: R. Fergus

Visualizations of Higher Layers

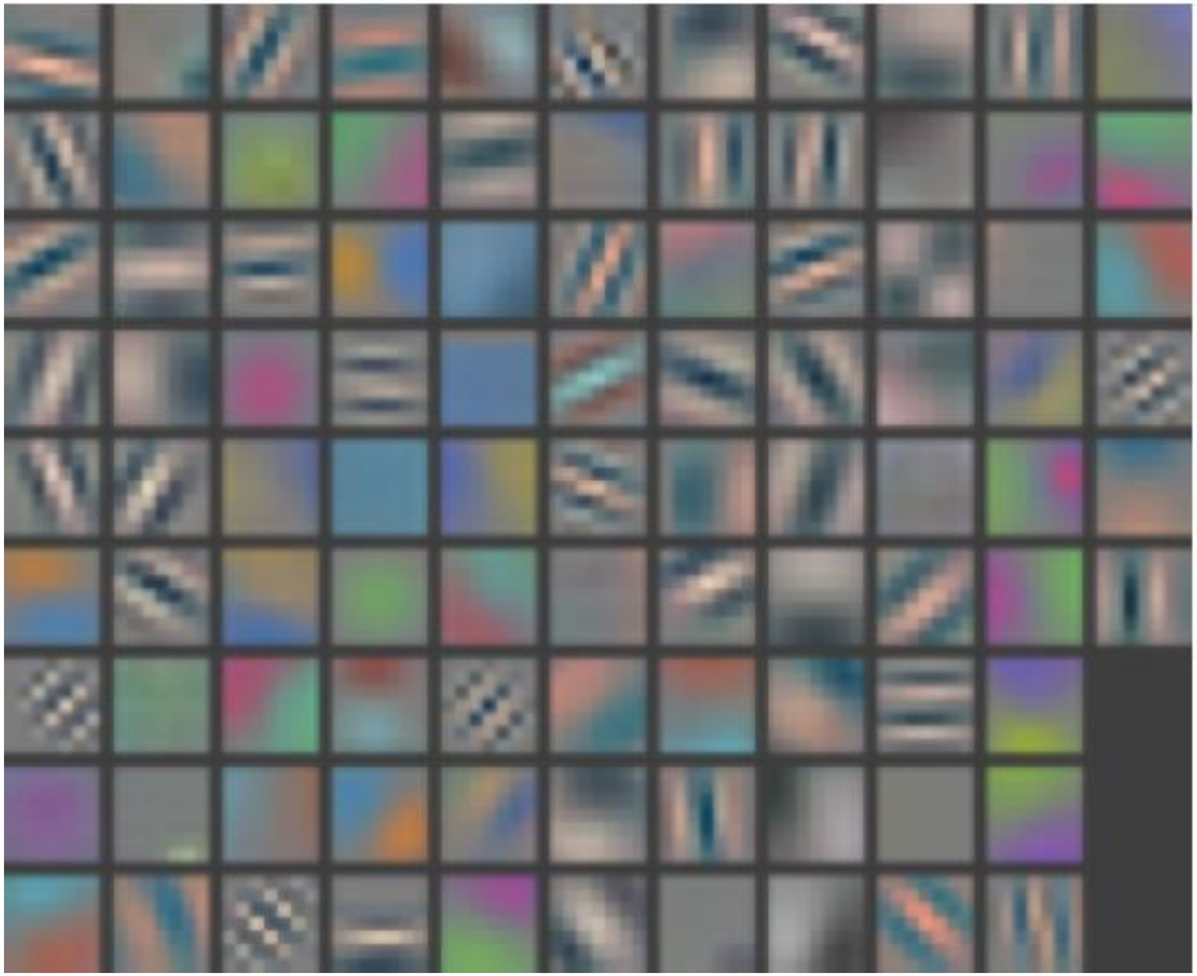
- Use ImageNet 2012 validation set (stack of images)
- Push each image through network and look for image with the strongest activation for each feature map



- Take max activation from feature map associated with each filter
- Use Deconvnet to project back to pixel space
- Use pooling “switches” distinctive to that activation

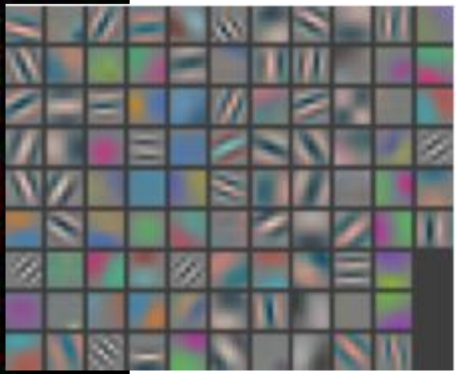
Credit: R. Fergus

Layer 1 Filters



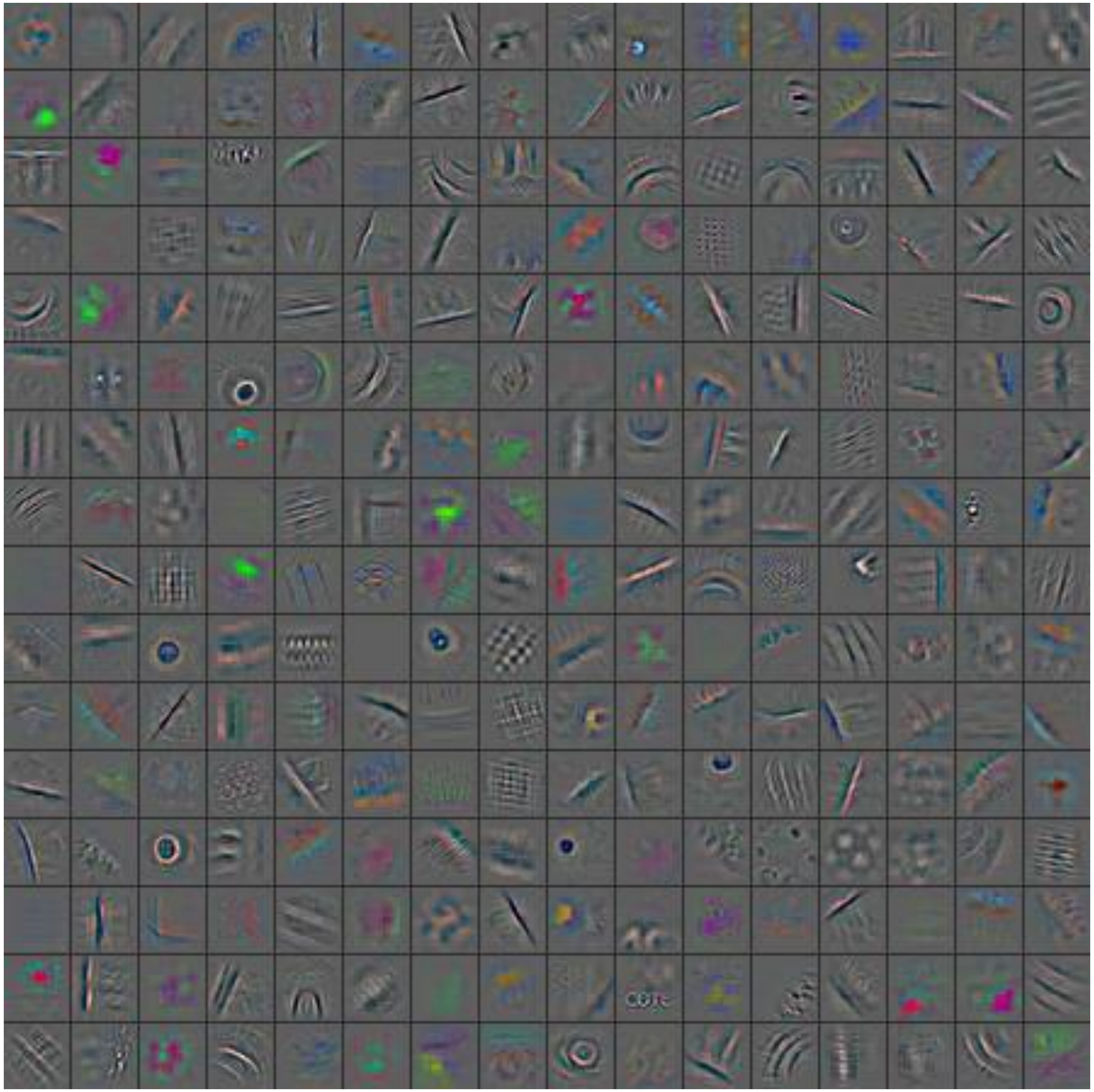
Credit: R. Fergus

Layer 1: Top-9 Patches

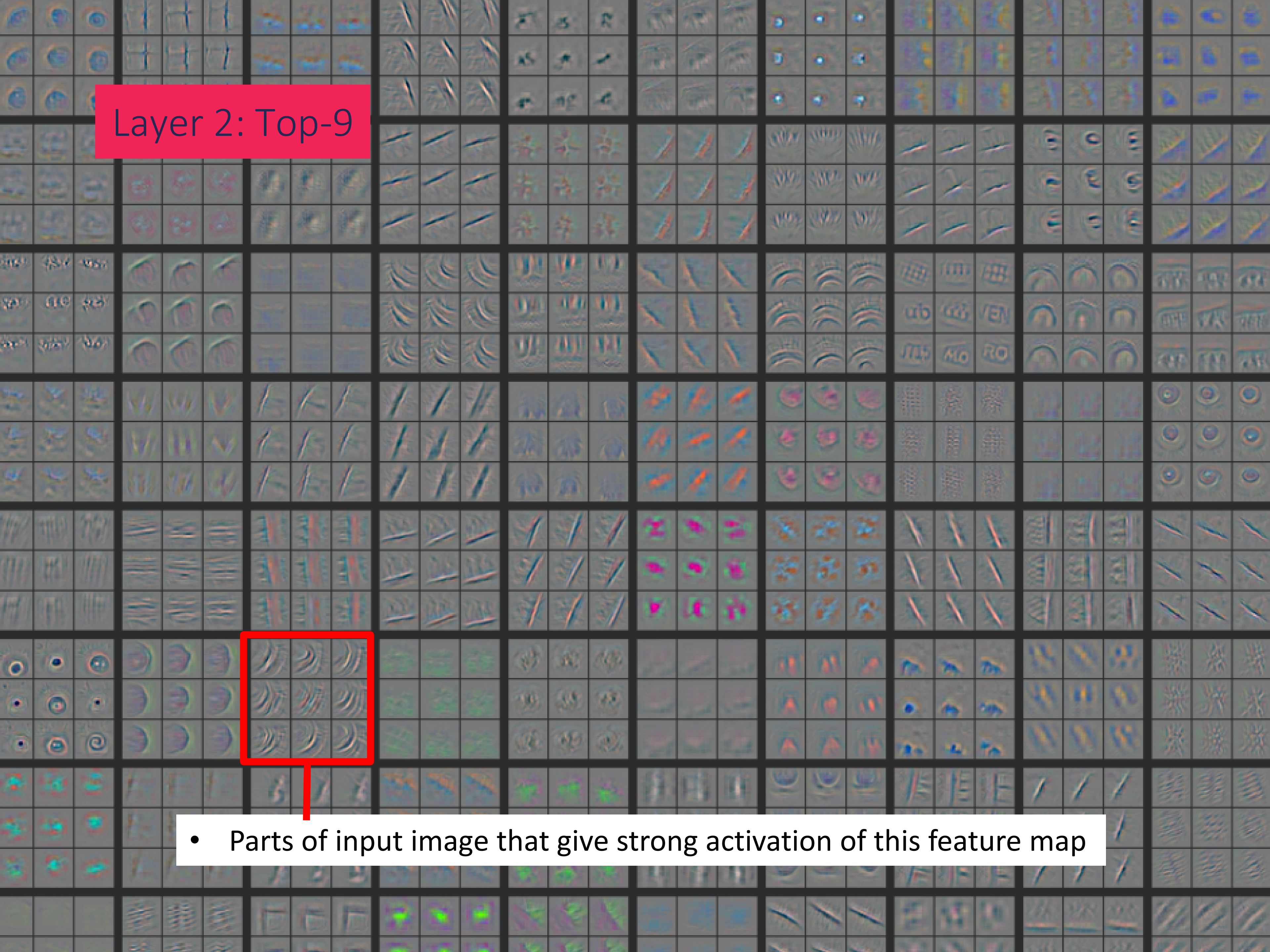


Credit: R. Fergus

Layer 2: Top-1



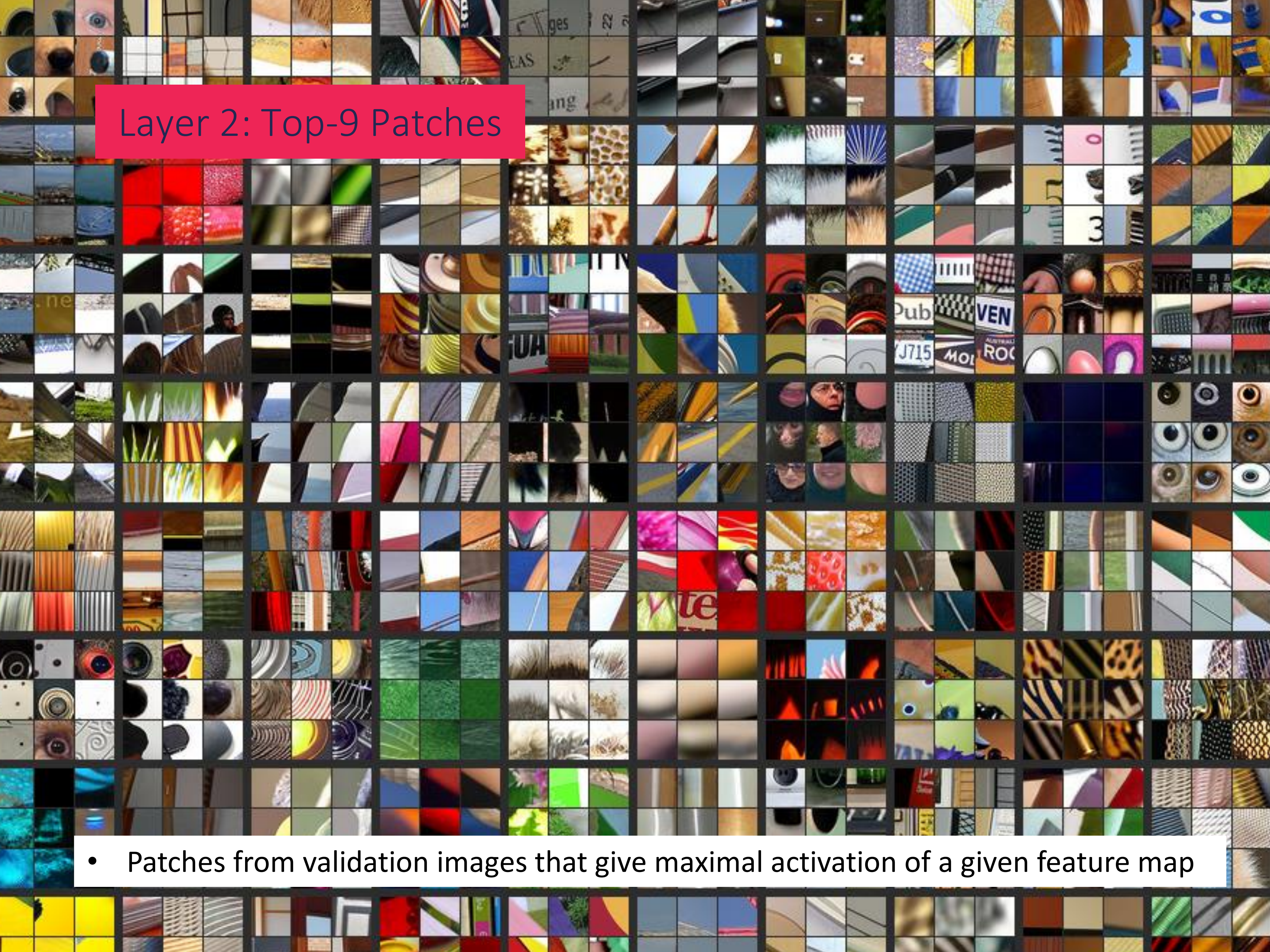
Credit: R. Fergus



Layer 2: Top-9

- Parts of input image that give strong activation of this feature map

Layer 2: Top-9 Patches



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


Layer 3: Top-1

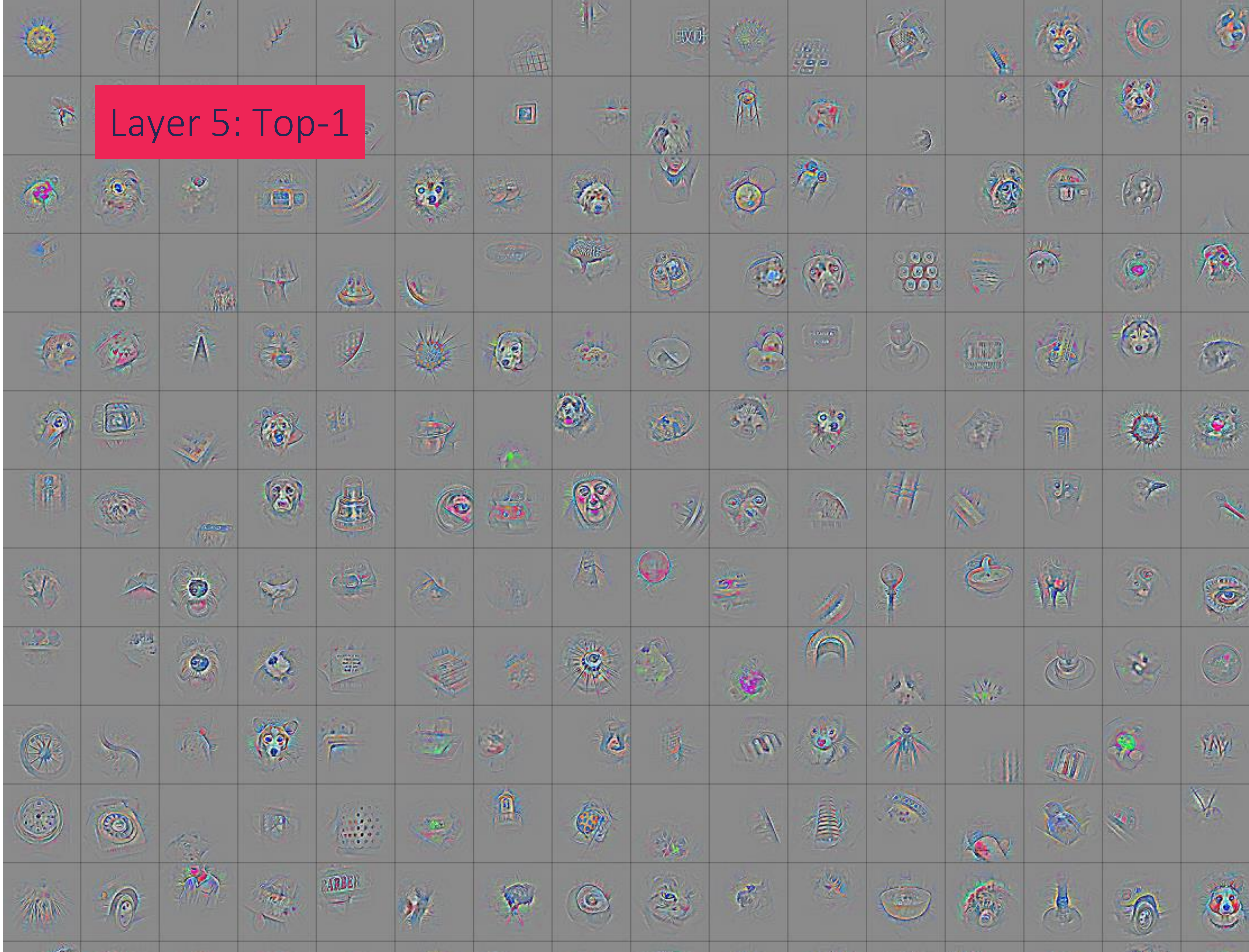

Layer 3: Top-9

Layer 3: Top-9


Layer 4: Top-1

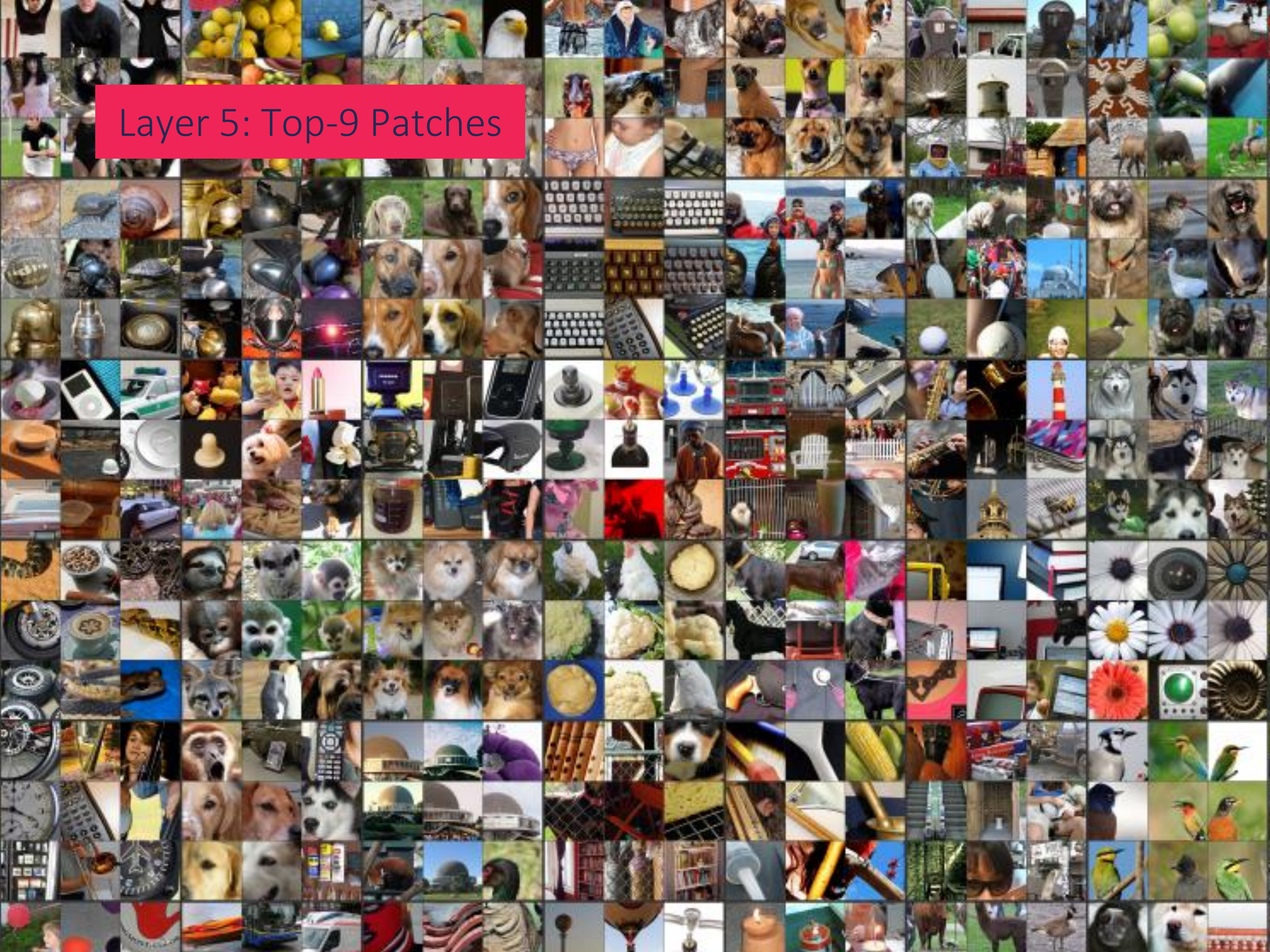

Layer 4: Top-9

Layer 4: Top-9 patches

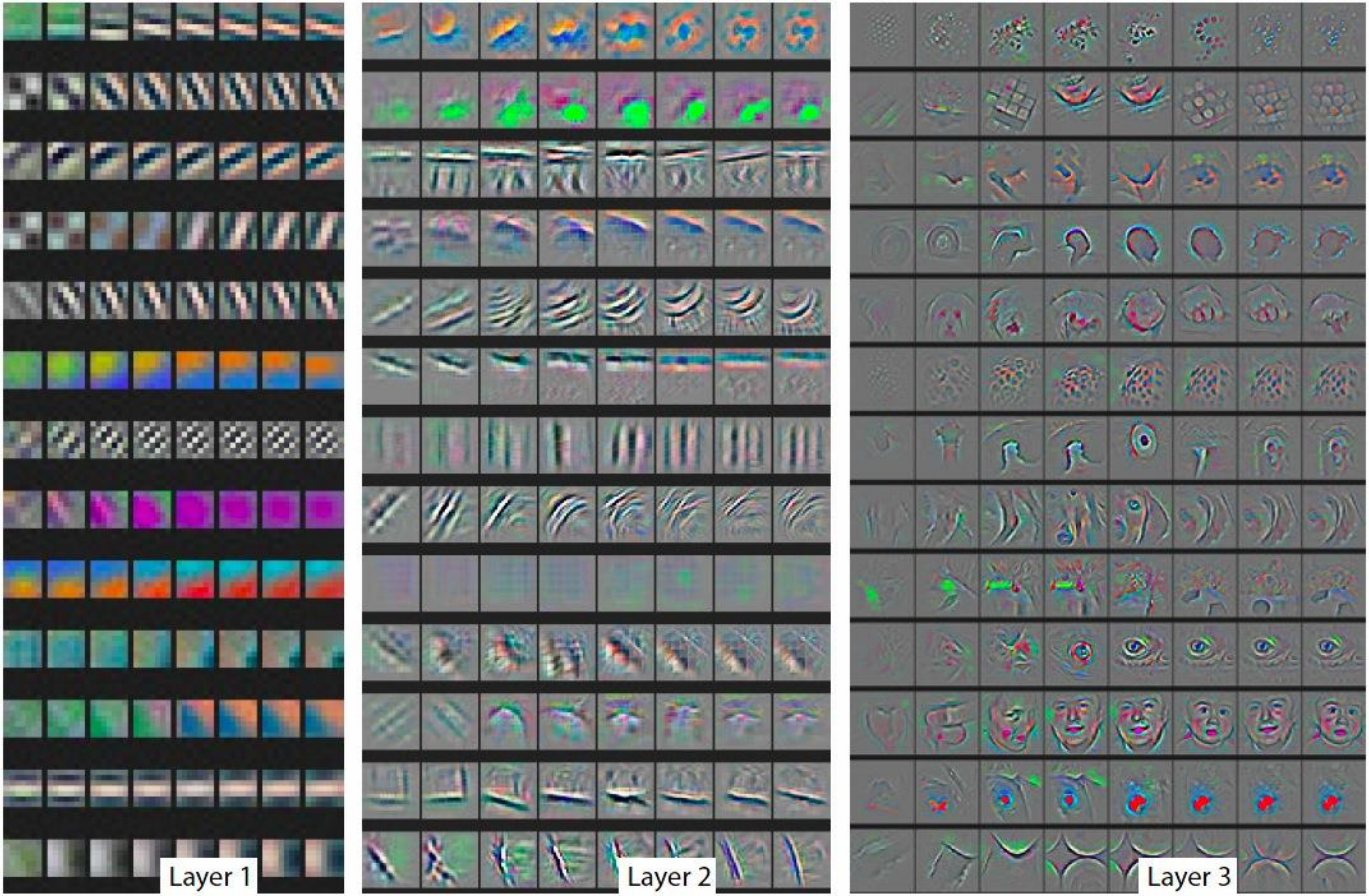

Layer 5: Top-1


Layer 5: Top-9

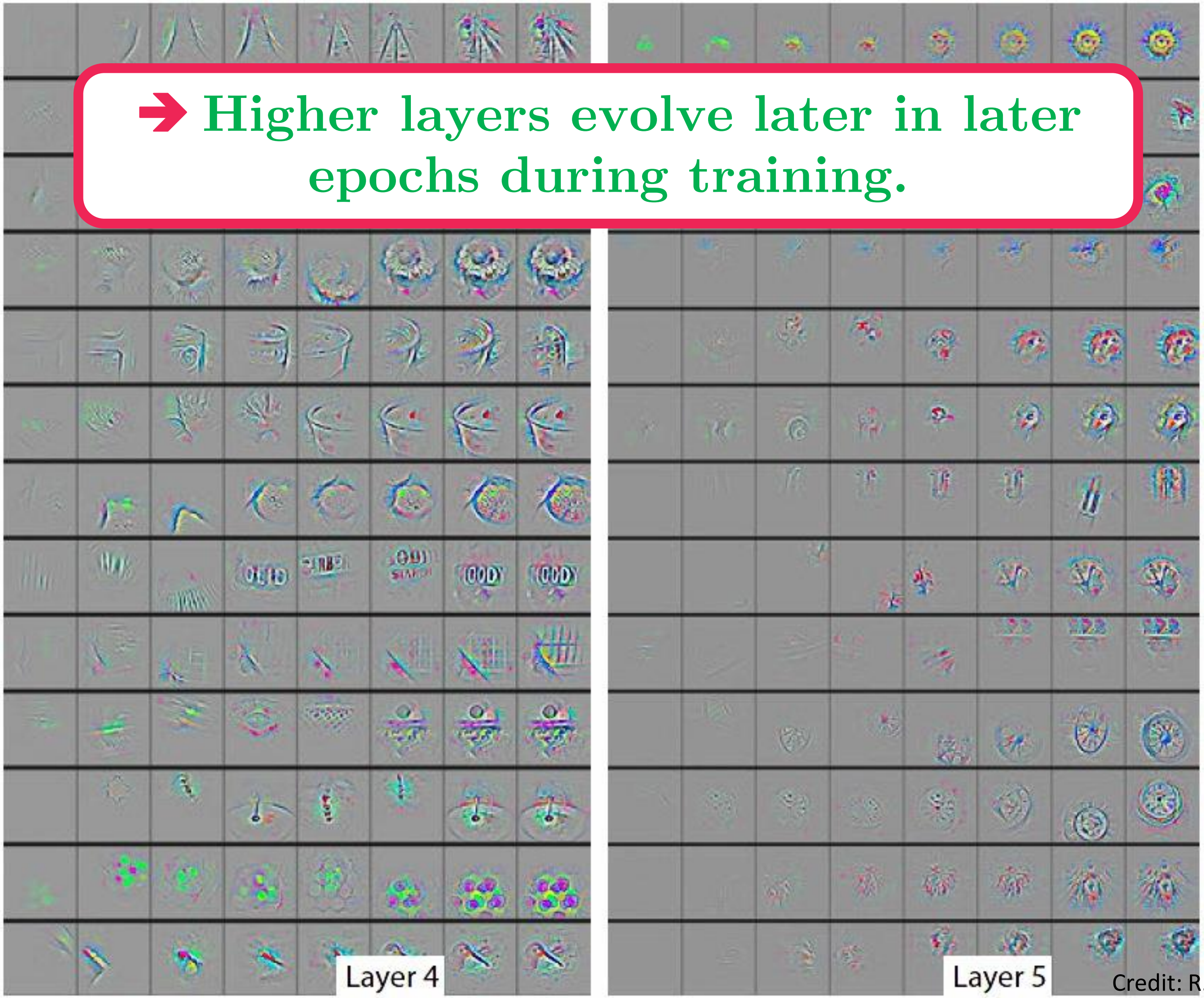
Layer 5: Top-9 Patches



Evolution of Features During Training



Evolution of Features During Training



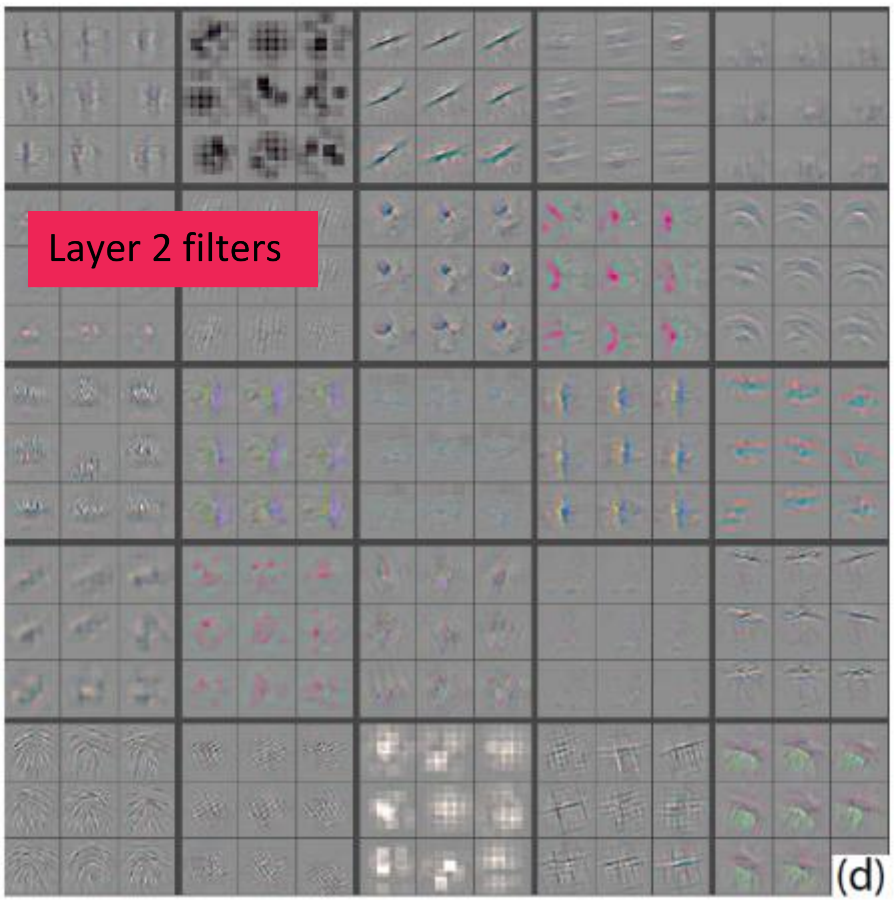
➔ **Higher layers evolve later in later epochs during training.**

Credit: R. Fergus

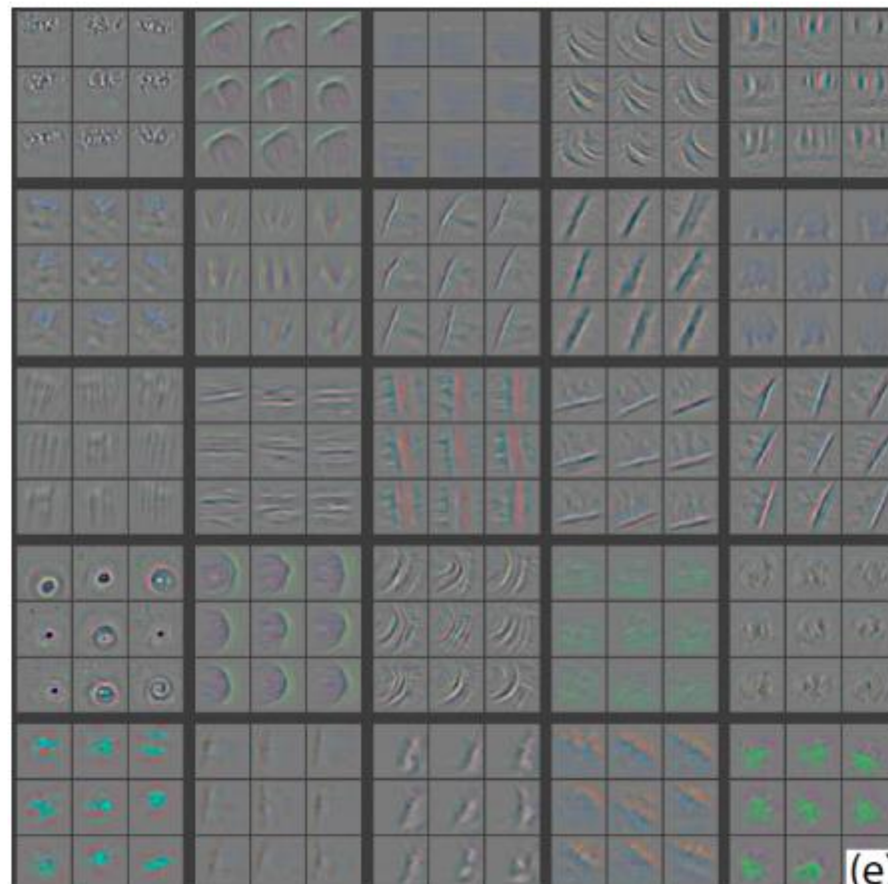
Visualizations can be used to improve model

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

Blocking artifacts



Smaller stride for convolution

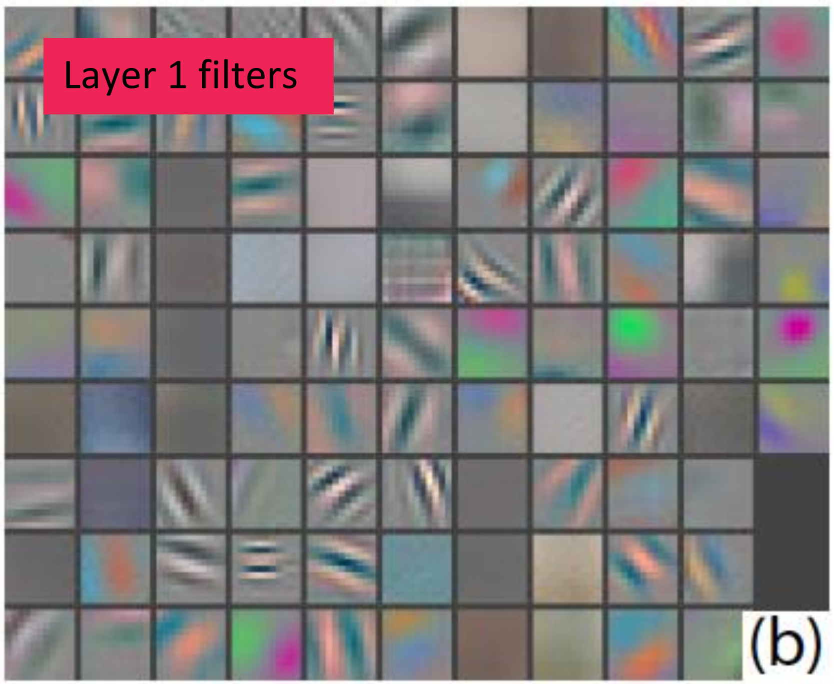


Credit: R. Fergus

Visualizations can be used to improve model

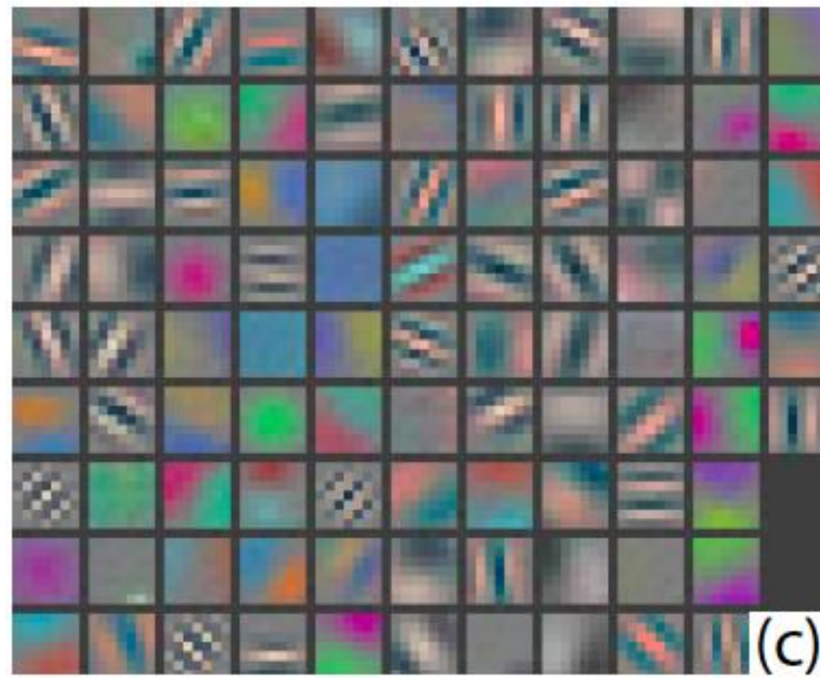
- Visualization of Krizhevsky et al.'s architecture showed some problems with layers 1 and 2
- Alter architecture: smaller stride & filter size
  - Visualizations look better and Performance improves

Too specific for low-level
+ dead filters



11x11 filters, stride 4

Restrict size (smaller)



7x7 filters, stride 2

Credit: R. Fergus

Occlusion Experiment

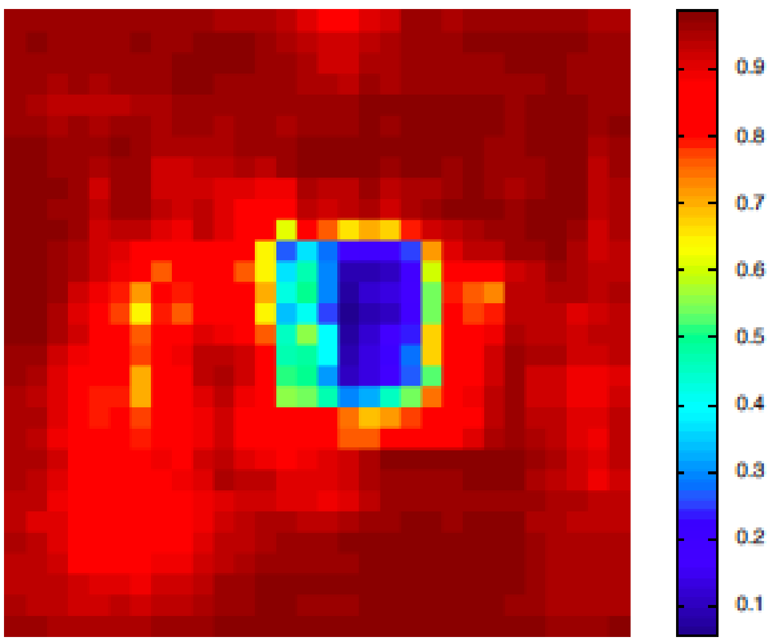
- Mask parts of input with occluding square
- Monitor output of classification network

- Perhaps network using scene context?

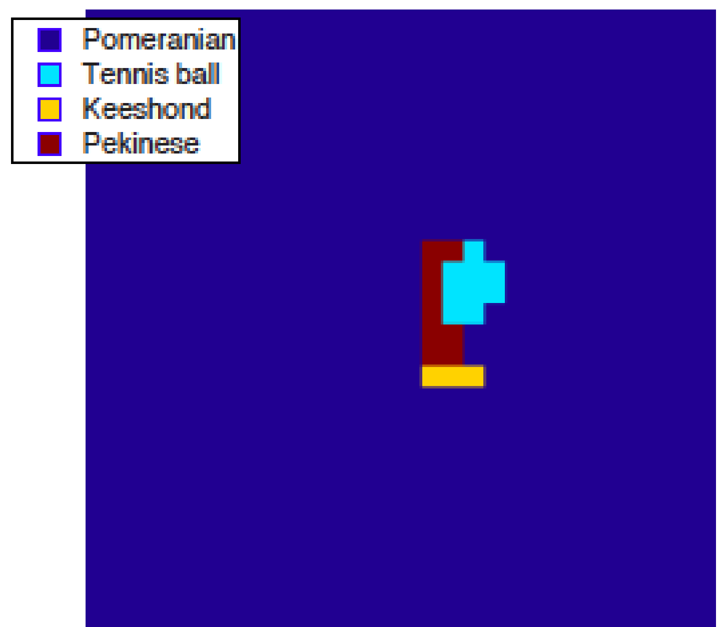
Credit: R. Fergus

Input image

p(True class)



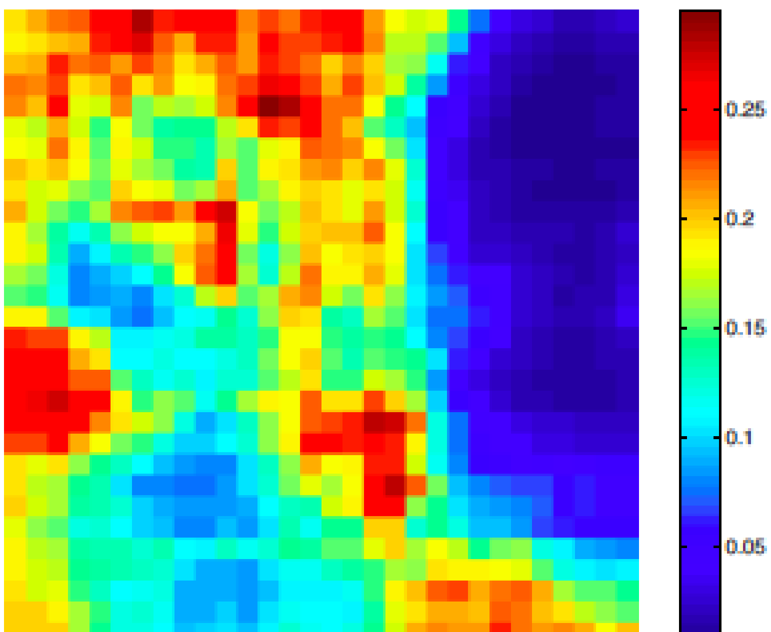
Most probable class



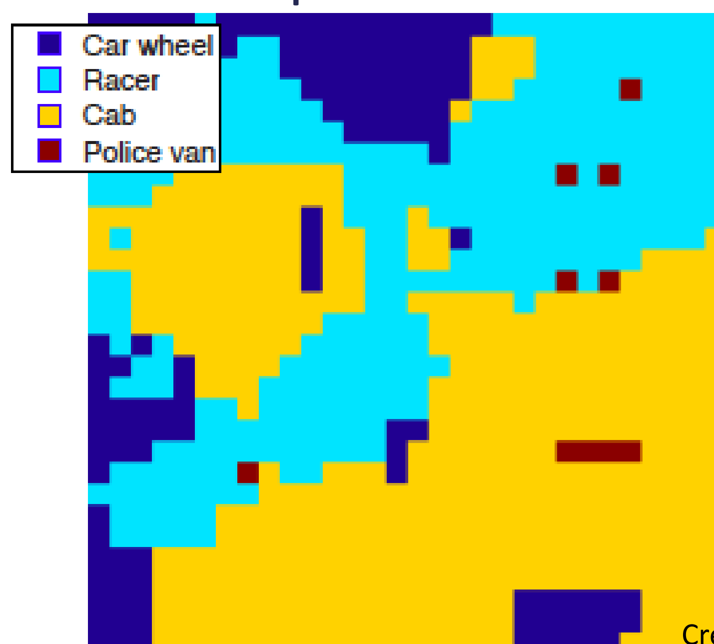
Credit: R. Fergus

Input image

p(True class)



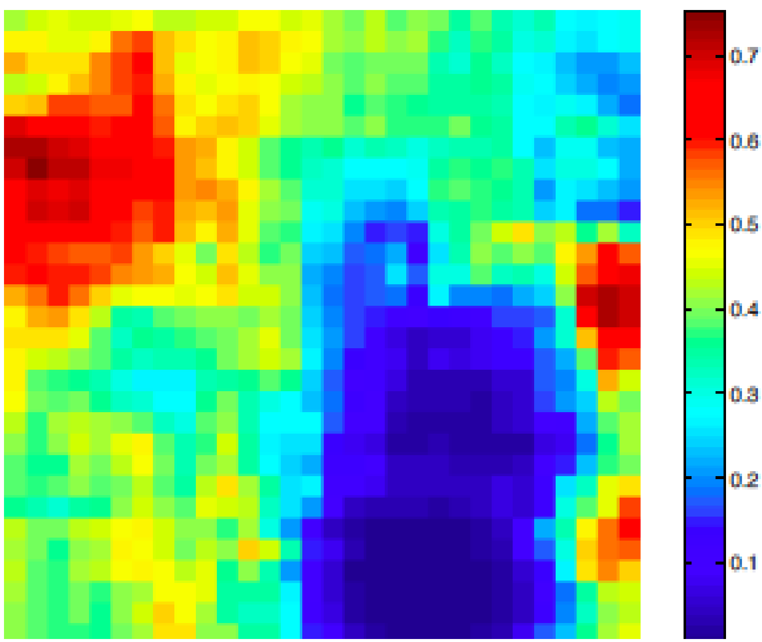
Most probable class



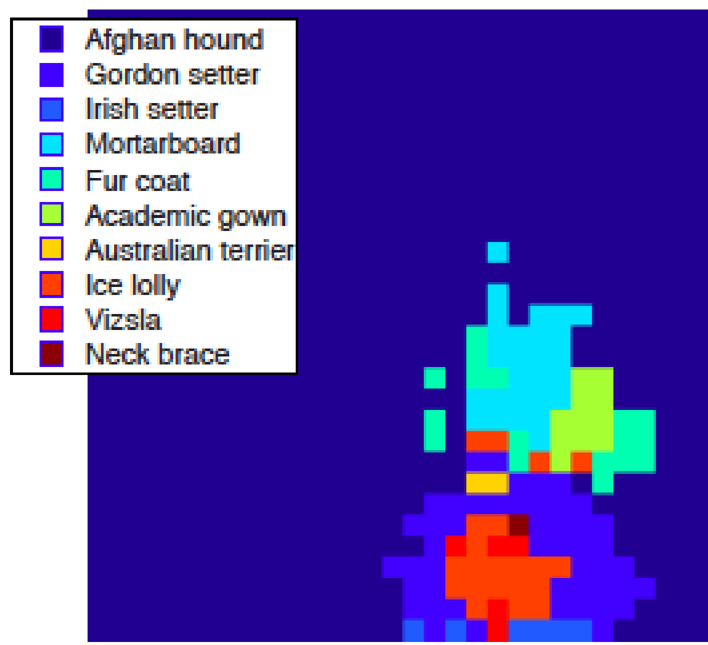
Credit: R. Fergus

Input image

p(True class)



Most probable class

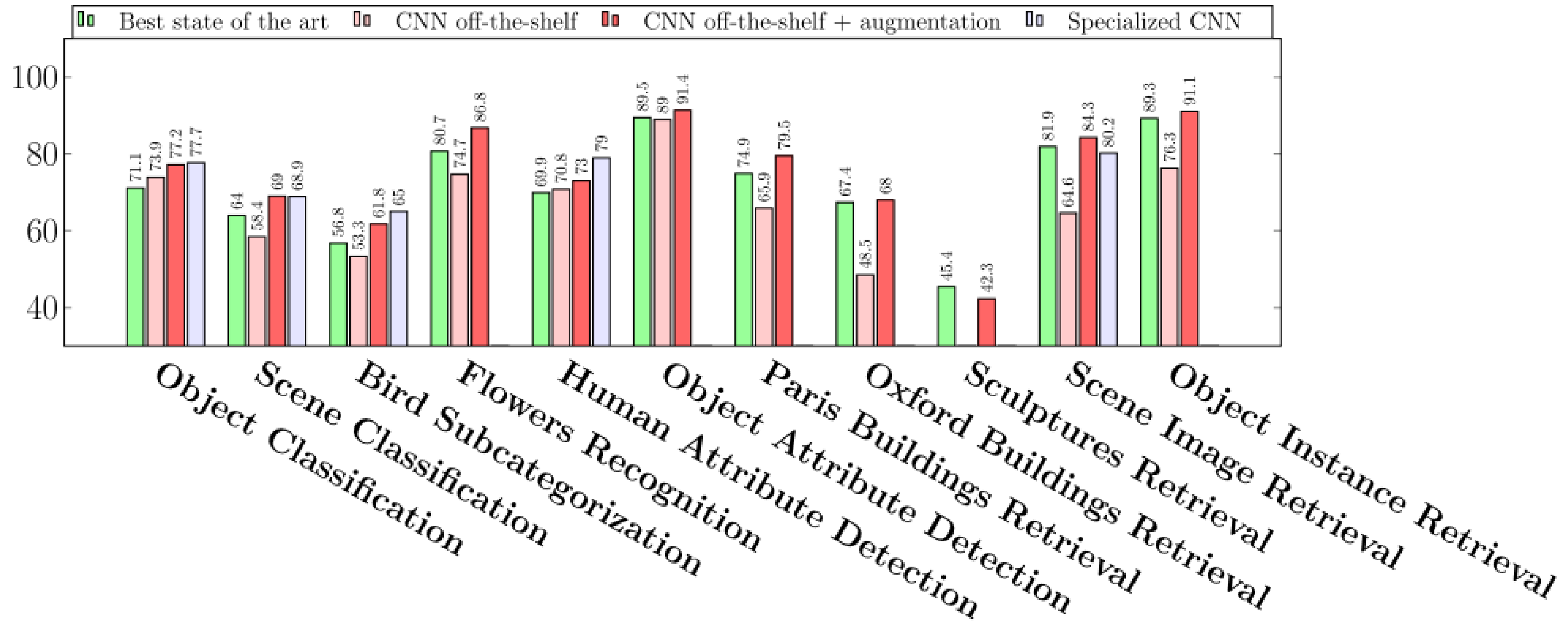


Credit: R. Fergus

Feature Generalization

ImageNet pre-training

- **Labeled data is rare for detection:** leverage large classification labeled datasets for pre-training.
- **ImageNet Classification pretraining + fine-tuning on a different task** has been shown to work very well by many people.
  - [Razavian'14] took the off-the-shelf convnet **OverFeat + SVM classifier** on top and obtained many state-of-the-art or competitive results on 10+ datasets and visual tasks

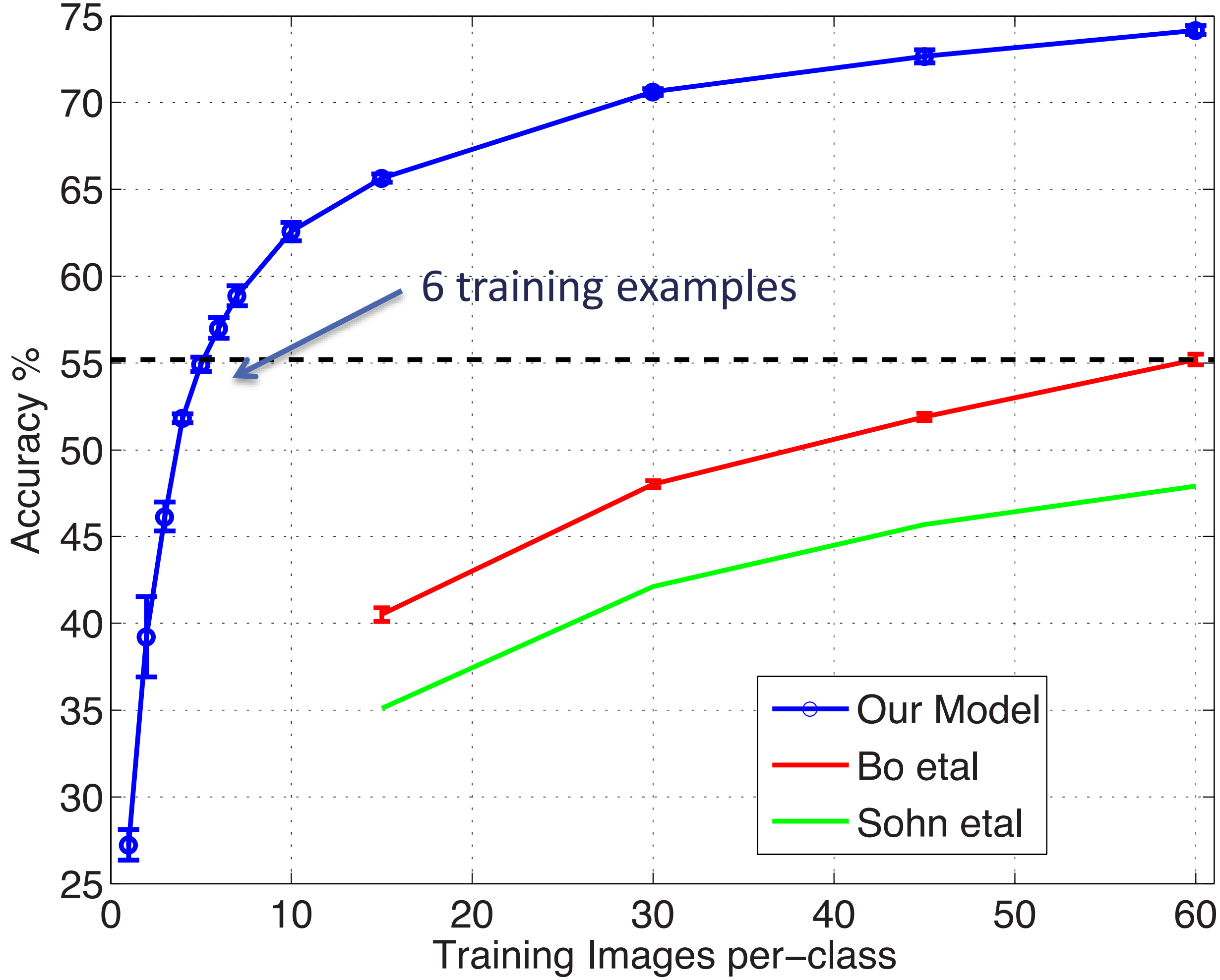


CNN Features off-the-shelf: an Astounding Baseline for Recognition. Razavian, Ali Sharif, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. *arXiv preprint arXiv:1403.6382* (2014).

Credit: P. Sermanet

Classifier re-training on Caltech 256

State of the art accuracy with only 6 training samples/class

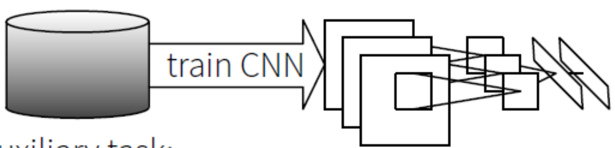


Zeiler & Fergus, *Visualizing and Understanding Convolutional Networks*, ECCV'14

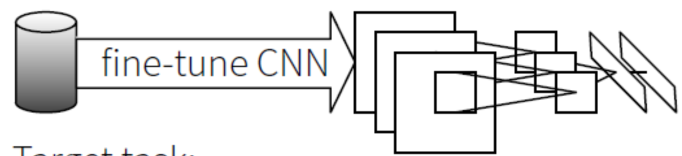
Credit: R. Fergus

Feature sharing via transfer learning

- Pre-training allows to use big models for small datasets
  - For example: Pre-Train model on large ImageNet 2012 training set



Auxiliary task:
ILSVRC 2012 *classification*
(1.2 million images)



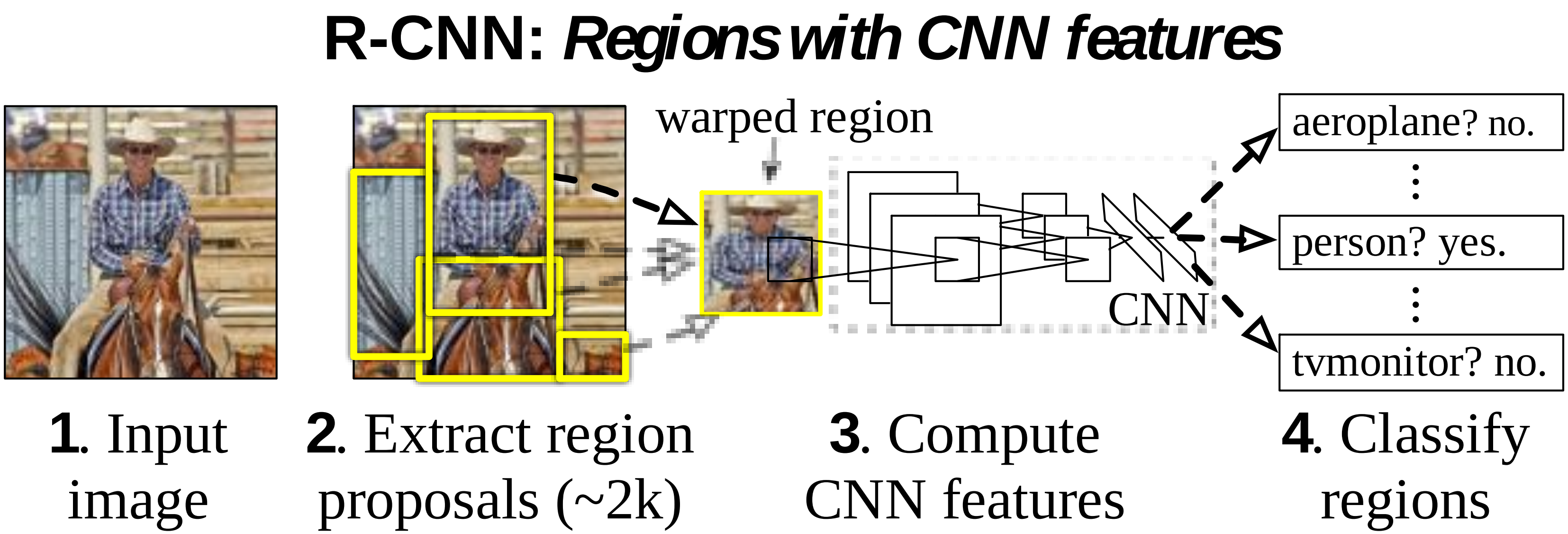
Target task:
PASCAL VOC detection
(~25k object labels)

[Girshick et al. CVPR'14]

- Re-train on new dataset (fine tuning or transfer learning)
  - Either: Just the classifier-layer or the whole network
    - For fine-tuning pre-trained layers, the learning rate has to be lowered to avoid unlearning the pre-trained weights
    - For fine tuning new layers (e.g. the new classifier layer) the learning rate has to be higher
  - Better: Two stage fine-tuning
    - Stage 1: First only learn new layers with the learning rate of pre-trained layers set to zero
    - Stage 2: Use default learning rate to fine-tune everything (optimize all parameters jointly)
- Classify test set of new dataset

(Fine tuned) CNNs for detection on the Pascal dataset

- Combines bottom-up region proposals with rich features computed by a CNN



[Girshick et al. CVPR'14]

- Previous state-of-the art: 35.1% mean average precision
- scratch: Training on Pascal train+val data
- pre-train: Pre-training on ImageNet and just the classifier is trained on Pascal
- fine-tune: Two stage fine-tuning on Pascal

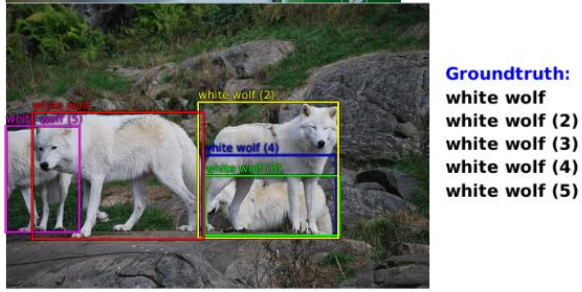
PASCAL-DET

scratch	pre-train	fine-tune
40.7	45.5	54.1

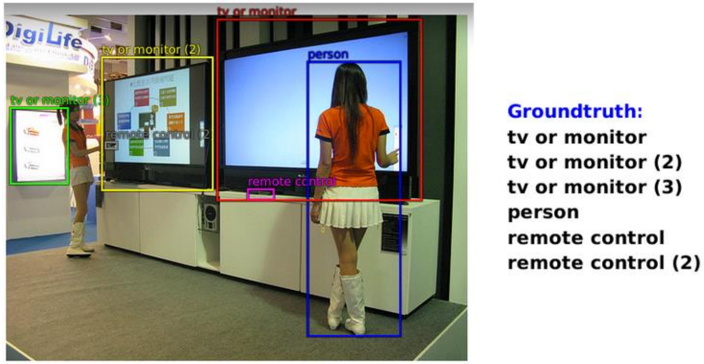
CNNs have set a new state-of-the art for many tasks

- classification

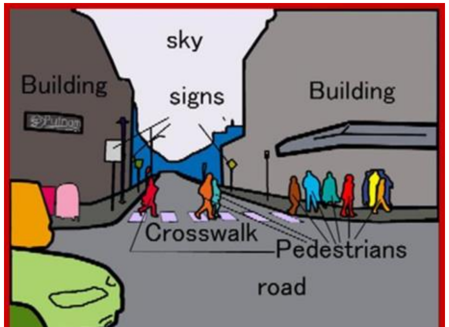
- localization



- detection



- segmentation



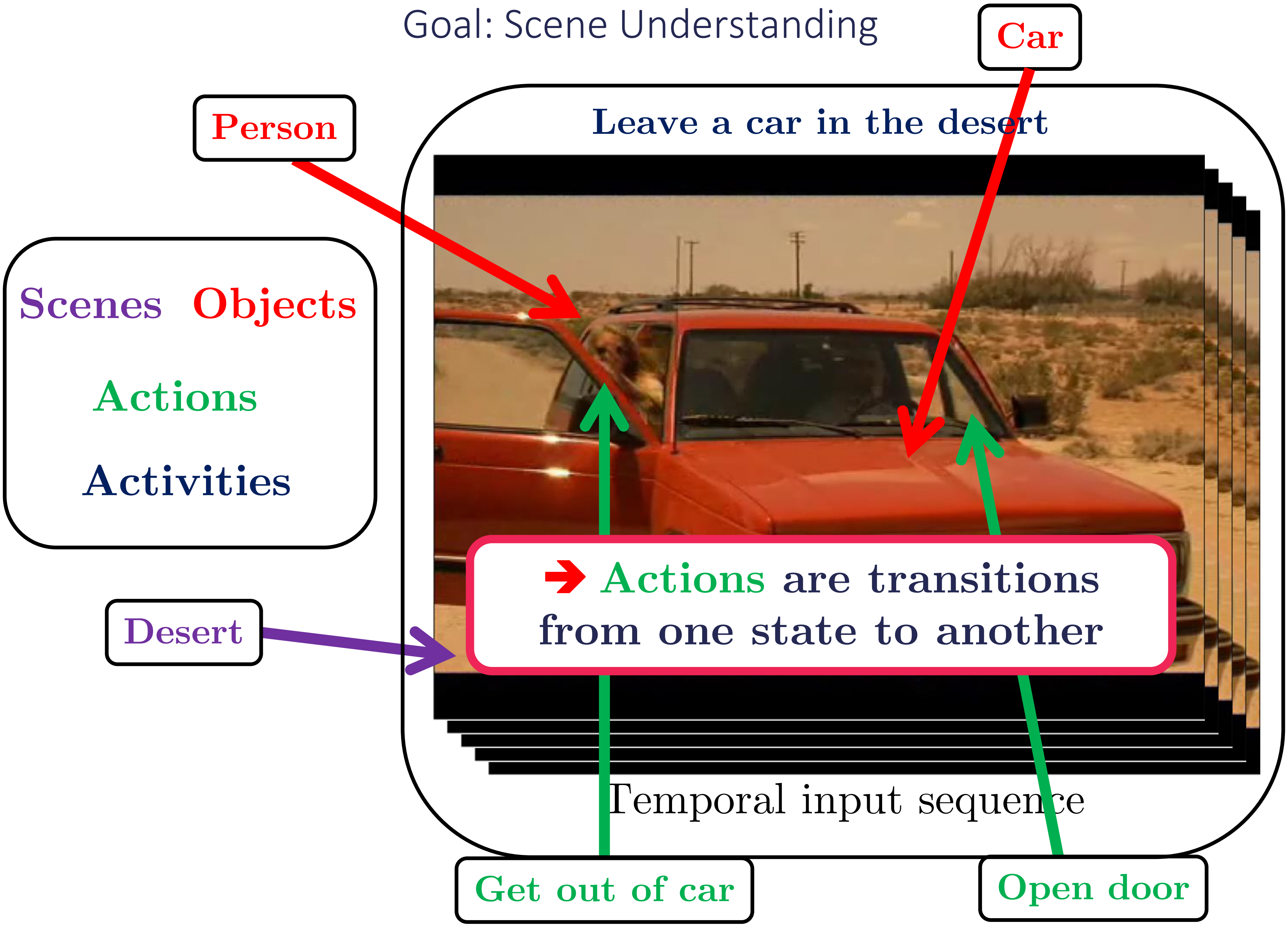
difficulty

Credit: P. Sermanet

… except for

Video Recognition

Goal: Scene Understanding



Action classification

I. Laptev and P. Pérez, Retrieving actions in movies, ICCV'07

Large-scale Video Classification

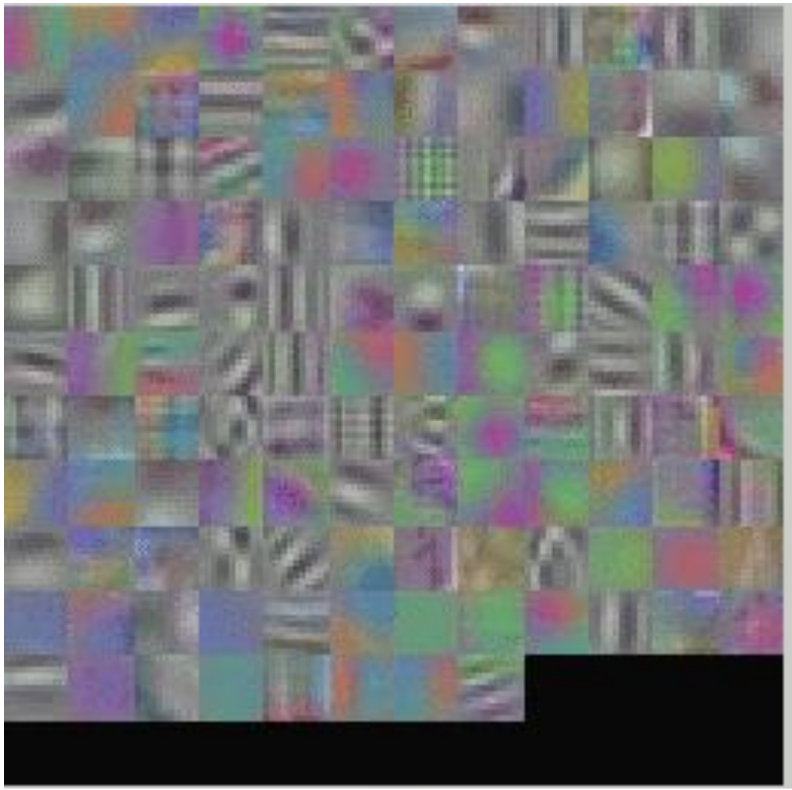
Sports-1M dataset

- 1 million YouTube videos in 487 classes of sports

Karpathy et al., Large-scale Video Classification with Convolutional Neural Networks, CVPR 2014

Large-scale Video Classification

Learned features of the first Layer

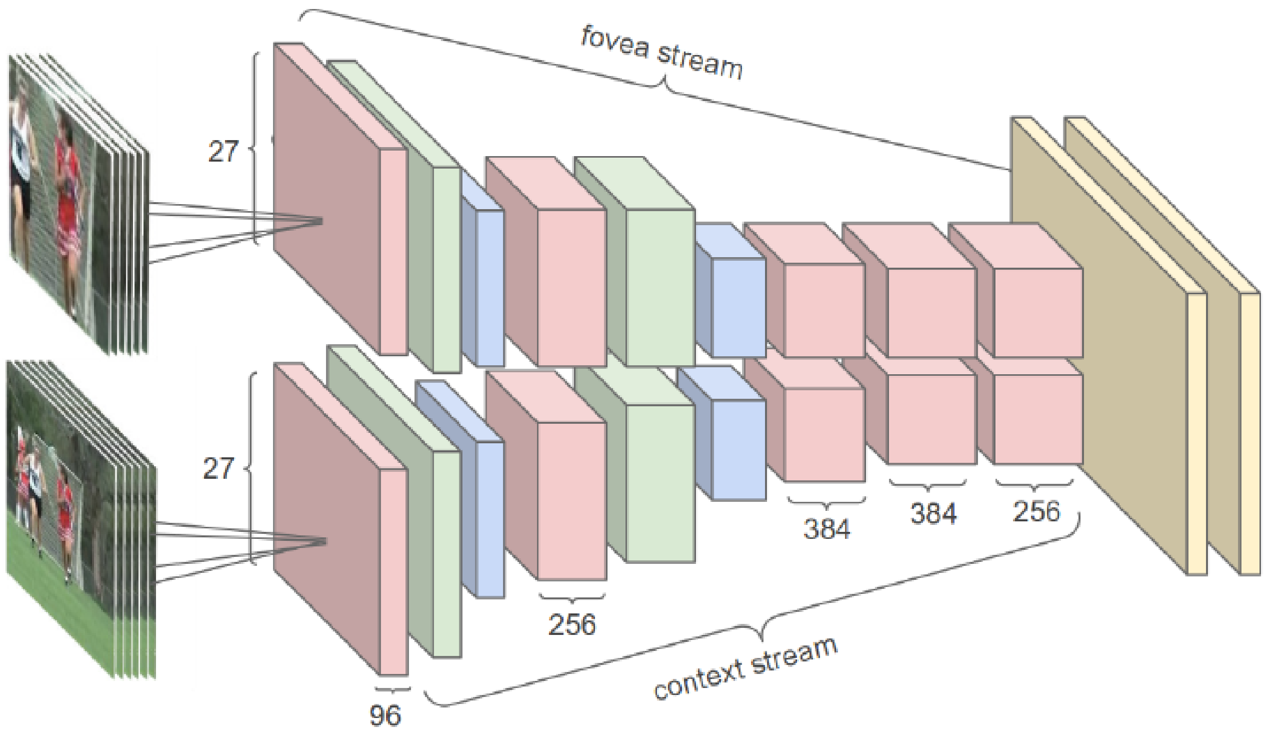


Karpathy et al., Large-scale Video Classification with Convolutional Neural Networks, CVPR 2014

Large-scale Video Classification

Multiresolution architecture

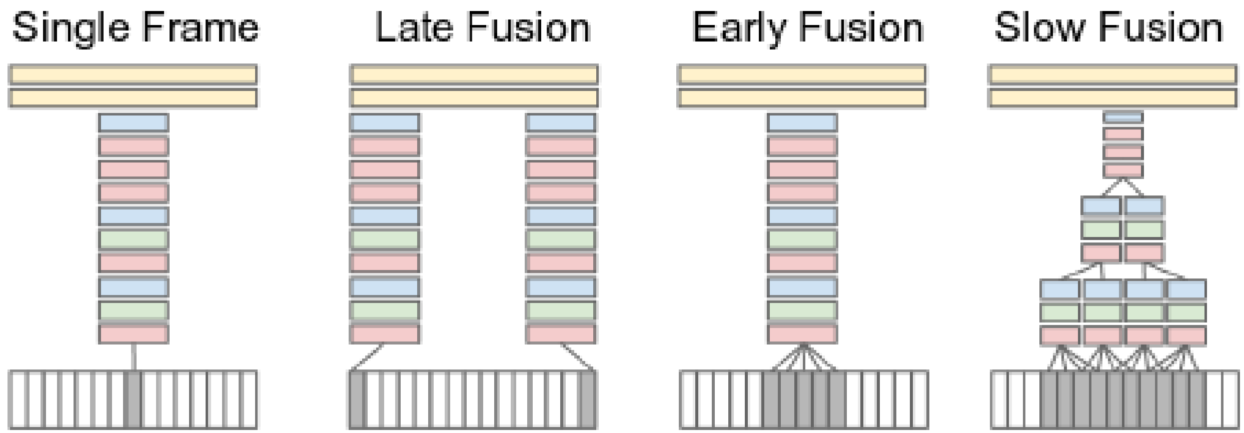
- Context + Fovea Stream



Karpathy et al., Large-scale Video Classification with Convolutional Neural Networks, CVPR 2014

Large-scale Video Classification

- Fusing information over temporal dimension through



Model	Clip Hit@1	Video Hit@1	Video Hit@5
Feature Histograms + Neural Net	-	55.3	-
Single-Frame	41.1	59.3	77.7
Single-Frame + Multires	**42.4**	**60.0**	**78.5**
Single-Frame Fovea Only	30.0	49.9	72.8
Single-Frame Context Only	38.1	56.0	77.2
Early Fusion	38.9	57.7	76.8
Late Fusion	40.7	59.3	78.7
Slow Fusion	**41.9**	**60.9**	**80.2**
CNN Average (Single+Early+Late+Slow)	41.4	63.9	82.4

Large-scale Video Classification

Transfer learning on UCF-101

- 13320 videos in 101 classes

Model	3-fold Accuracy
Soomro et al [22]	43.9%
Feature Histograms + Neural Net	59.0%
Train from scratch	41.3%
Fine-tune top layer	64.1%
Fine-tune top 3 layers	**65.4%**
Fine-tune all layers	62.2%

Table 3: Results on UCF-101 for various Transfer Learning approaches using the Slow Fusion network.

85.9% using Improved Dense Trajectories + Fisher Vectors [Wang et al. '13]

87.6% using Two-stream CNN [Simonyan and Zisserman '14]

Karpathy et al., Large-scale Video Classification with Convolutional Neural Networks, CVPR 2014

Large-scale Video Classification

Transfer learning on UCF-101

- 13320 videos in 101 classes

Group	mAP from scratch	mAP fine-tune top 3	mAP fine-tune top
Human-Object Interaction	0.26	0.55	0.52
Body-Motion Only	0.32	0.57	0.52
Human-Human Interaction	0.40	0.68	0.65
Playing Musical Instruments	0.42	0.65	0.46
Sports	**0.57**	**0.79**	**0.80**
All groups	0.44	0.68	0.66

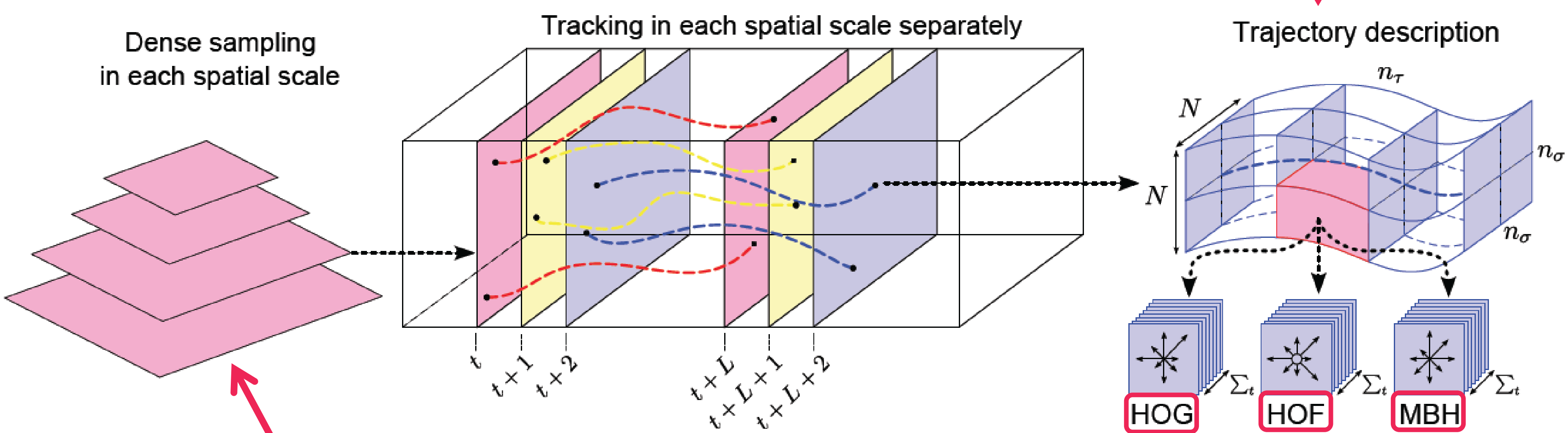
Table 4: Mean Average Precision of the Slow Fusion network on UCF-101 classes broken down by category groups.

Karpathy et al., Large-scale Video Classification with Convolutional Neural Networks, CVPR 2014

State of the art in Action Recognition: Dense Trajectories + Fisher Vectors [Wang et al. '13]

- Low level primitive features are extracted densely at the first layer by tracking trajectories in a dense optical flow field

motion trajectory aligned pooling of primitives



multiscale interest points

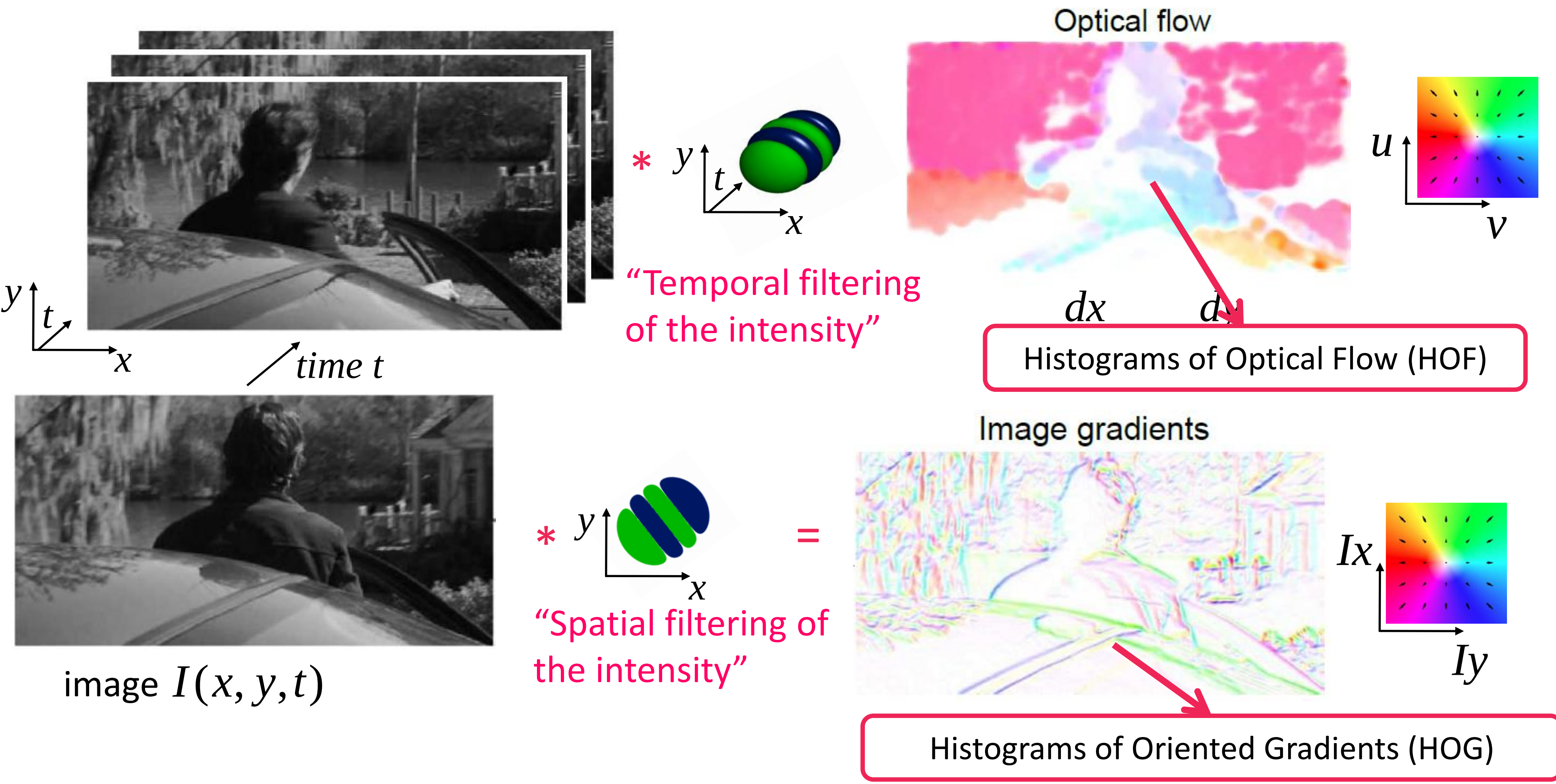
...creates feature descriptors

Wang et al. "Action Recognition by Dense Trajectories." CVPR 2011.

➔ Optical flow also captures camera motion and parallax

within a convolutional framework

- Optical flow is the apparent motion of the brightness pattern between images
- Image gradients are the directional change of the intensity or colour in the image

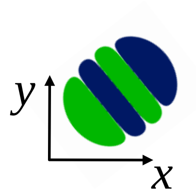


State of the art hand-crafted features within a convolutional framework

- Motion boundaries are the image gradients of the horizontal and vertical Optical flow components (i.e. u and v)

Optical flow

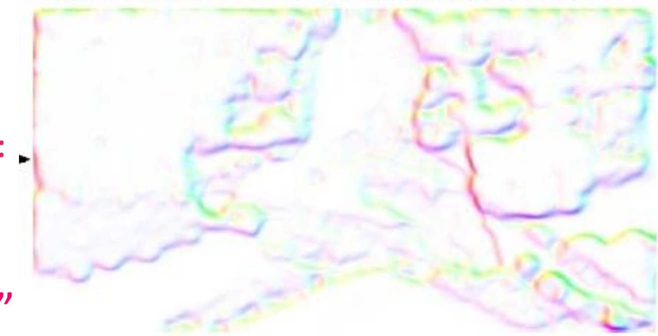
$$\boxed{u = \frac{dx}{dt}} \; *$$



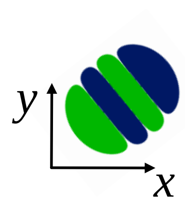
=

"Spatial filtering of the horizontal flow"

Horizontal motion boundaries



$$\boxed{v = \frac{dy}{dt}} \; *$$

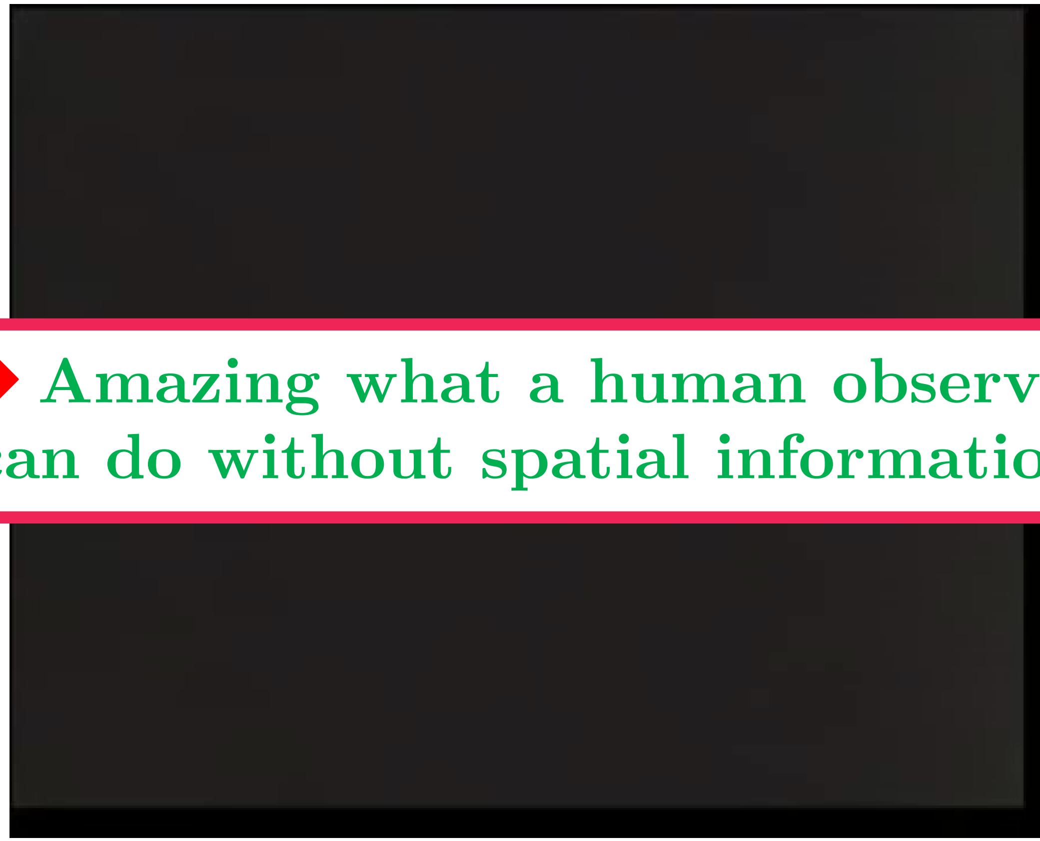


=

"Spatial filtering of the vertical flow"

Vertical motion boundaries

Johansson: Perception of Biological Motion



➔ **Amazing what a human observer can do without spatial information**

Sources: Johansson, G. “Visual perception of biological motion and a model for its analysis.” Perception & Psychophysics. 14(2):201-211. 1973.
Videos were made by JB Maas in 1971 (released via Houghton-Mifflin and now available on Youtube).

Credit: J. Corso

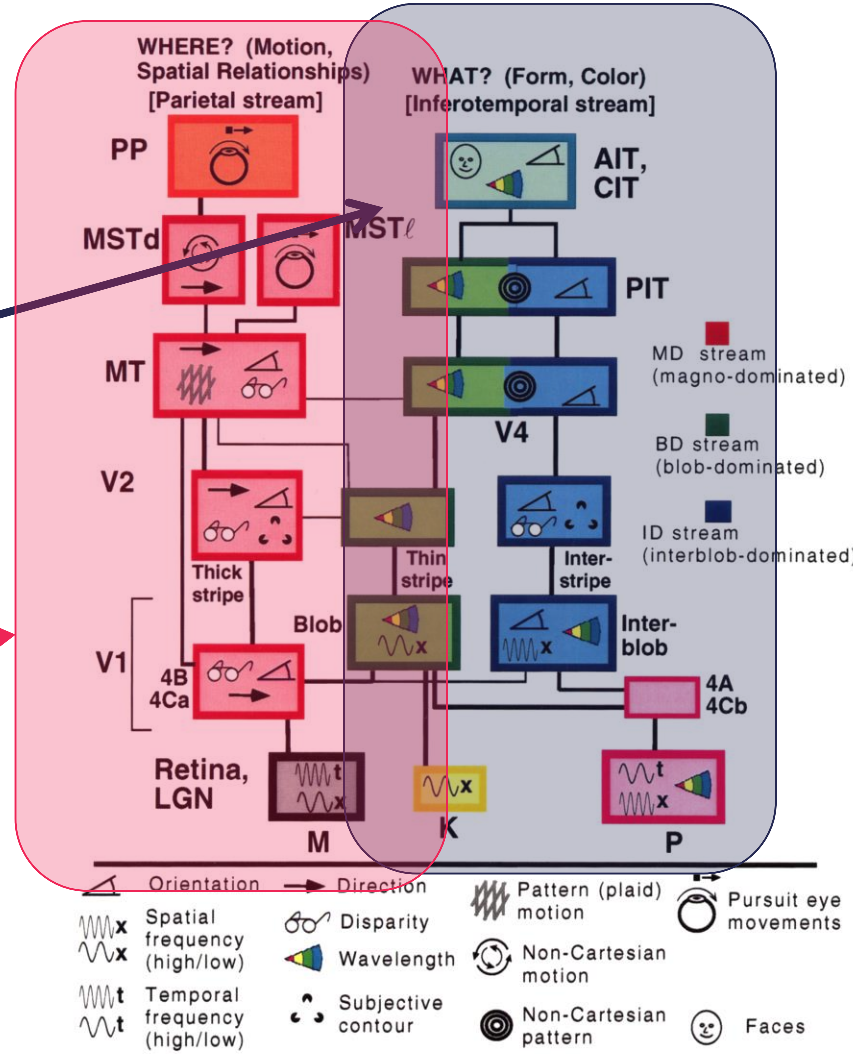
Motivation: Separate visual pathways for perception and action

The Human Visual Cortex has two **hierarchical** pathways

- Ventral stream performs object recognition
- Dorsal stream recognizes motion and locates objects

Spatial ConvNet?

Temporal ConvNet?



David C. Van Essen, Jack L. Gallant, Neural mechanisms of form and motion processing in the primate visual system,
Neuron, Volume 13, Issue 1, July 1994, Pages 1-10, ISSN 0896-6273

Two-Stream Convolutional Networks for Action Recognition in Videos

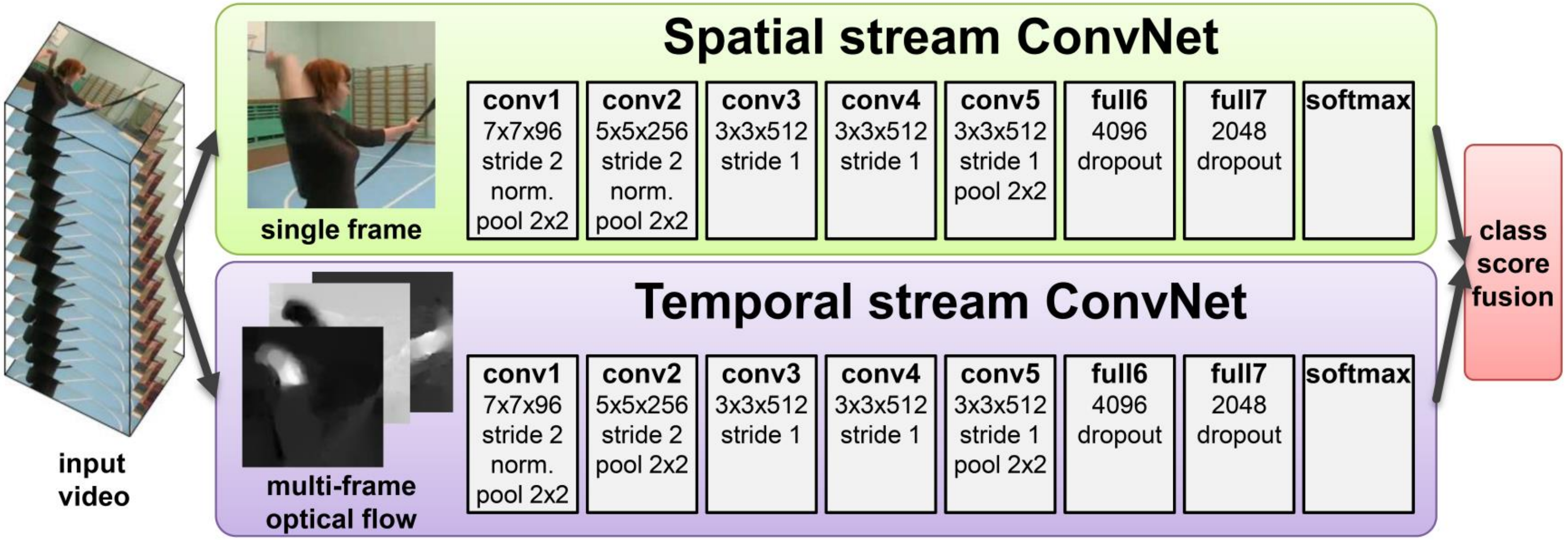


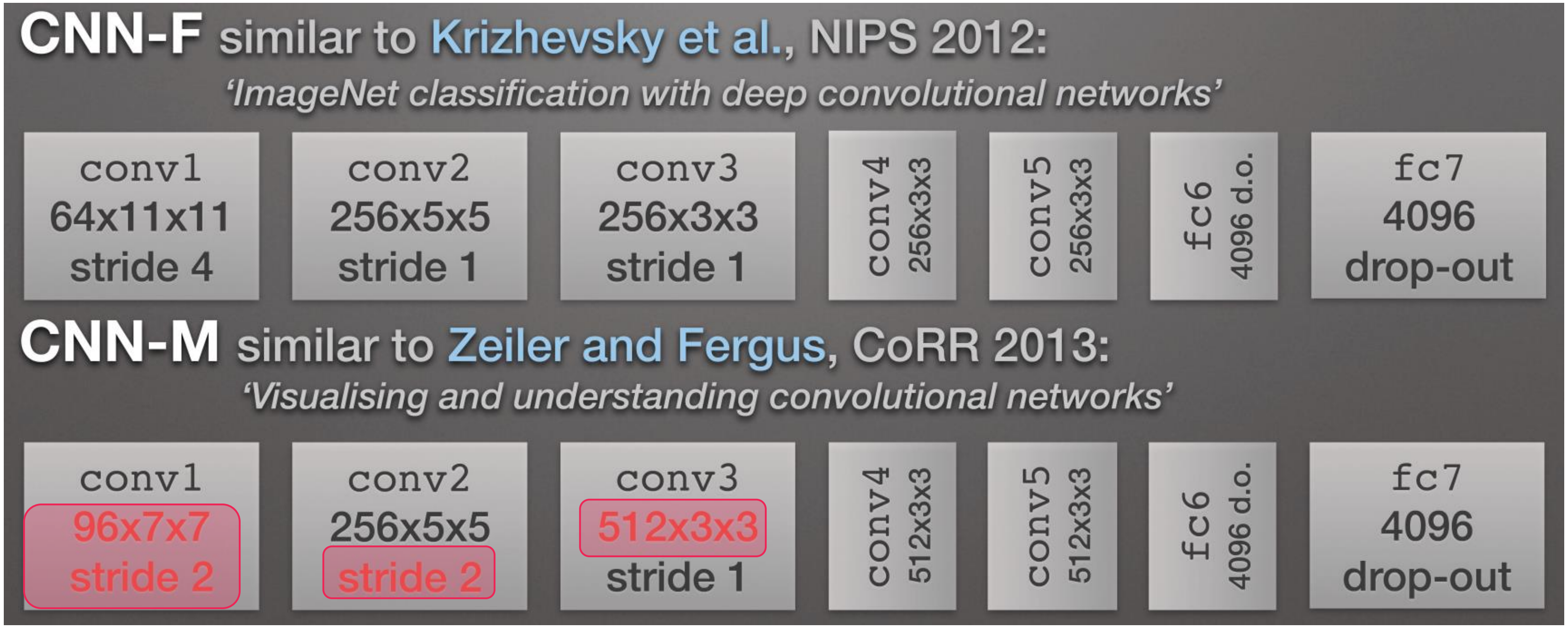
Figure 1: **Two-stream architecture for video classification.**

[Simonyan and Zisserman NIPS'14]

Individual processing of spatial and temporal information

- Using a separate ConvNet recognition stream for each
- Late fusion via softmax score averaging

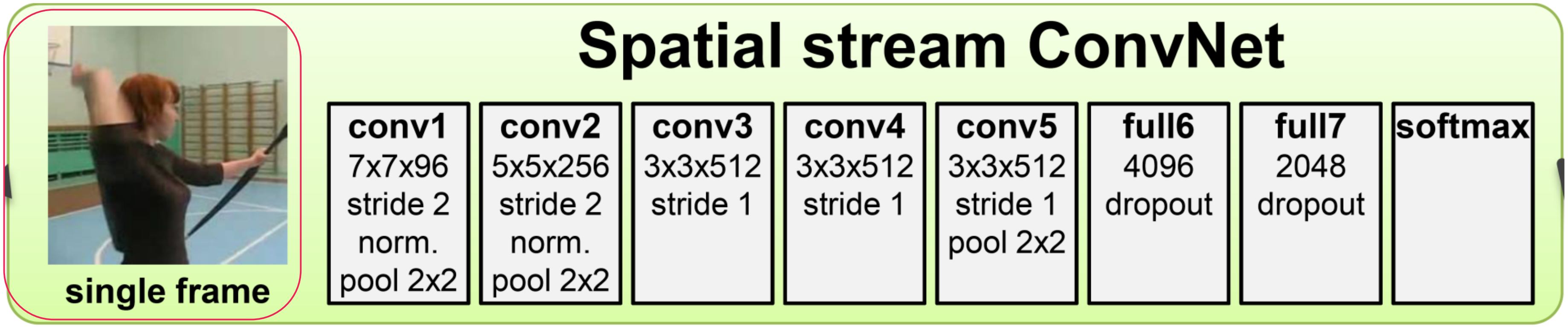
Spatial stream ConvNet



[Chatfield et al. BMVC'14]

Same network (CNN-M) used for both streams

- Based on [Krizhevsky et al. NIPS'12]
- Better (≈deeper) architectures exist now (see last lecture)
  - GoogLeNet
  - VGG Very Deep

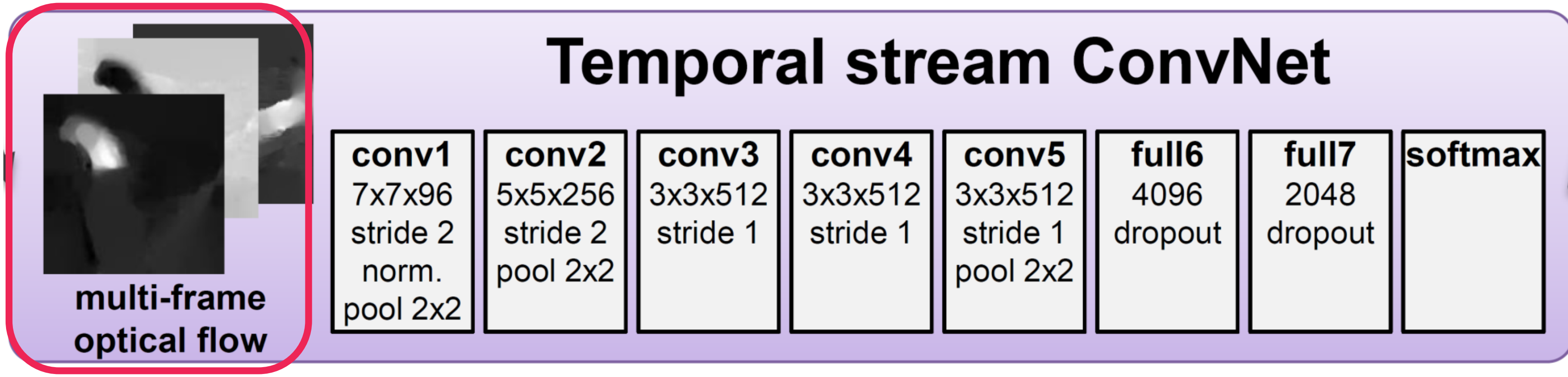


- Performs image classification on single RGB frames

Training:
- Supervised pre-training on ILSVRC (1.2M images in 1000 classes)
- Fine tuning of the softmax layer using the video frames

Testing
- ConvNet processes overy 25th frame of a video
- Data augmentation: 10 ConvNet inputs for each frame (crops & flips)
- Results for all ConvNets are averaged

[Simonyan and Zisserman NIPS'14]



Same model as in the spatial net exept

- Optical flow over several frames acts as input

Displacement vector field between (a) and (b)

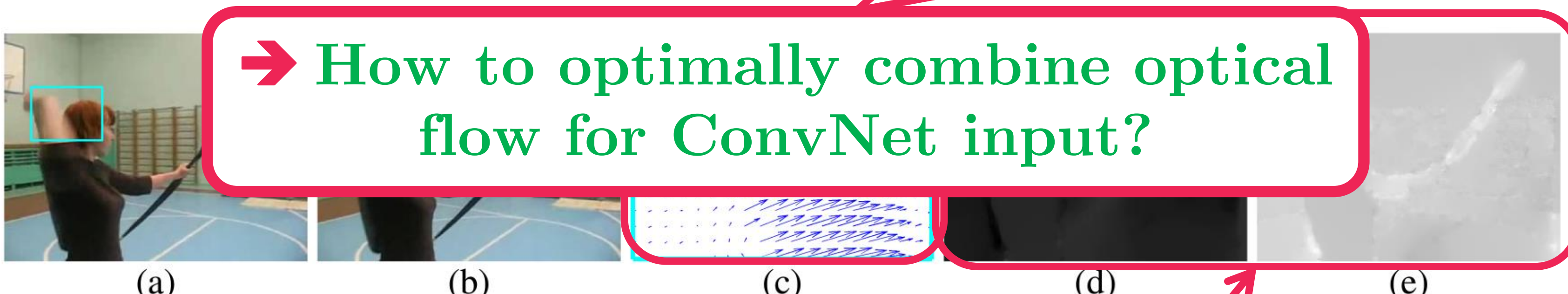


Figure 2: **Optical flow.** (a),(b): a pair of consecutive video frames with the area around a moving hand outlined with a cyan rectangle. (c): a close-up of dense optical flow in the outlined area; (d): horizontal component d^x of the displacement vector field (higher intensity corresponds to positive values, lower intensity to negative values). (e): vertical component d^y. Note how (d) and (e) highlight the moving hand and bow. The input to a ConvNet contains multiple flows (Sect. 3.1).

Horizontal and vertical flow is rescaled to [0, 255] for ConvNet input

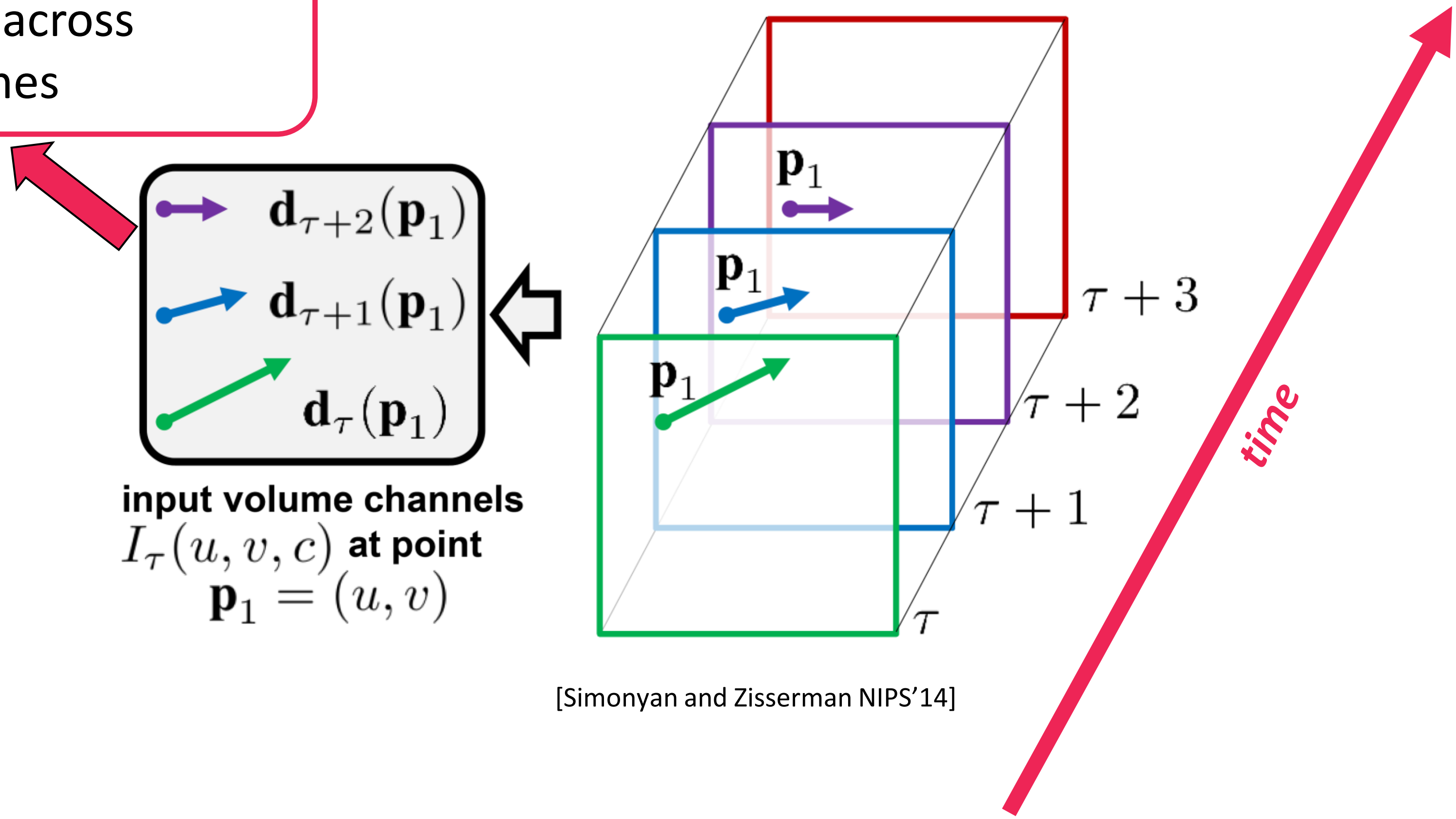
[Simonyan and Zisserman NIPS'14]

Optical flow stacking

Stack horizontal and vertical displacement fields $\mathbf{d}$

- Optical flow, $\mathbf{d}$, over several frames, τ, acts as input I_τ to the network

Input I_τ at $\mathbf{p}_1$ represents motion at $\mathbf{p}_1$ across multiple frames

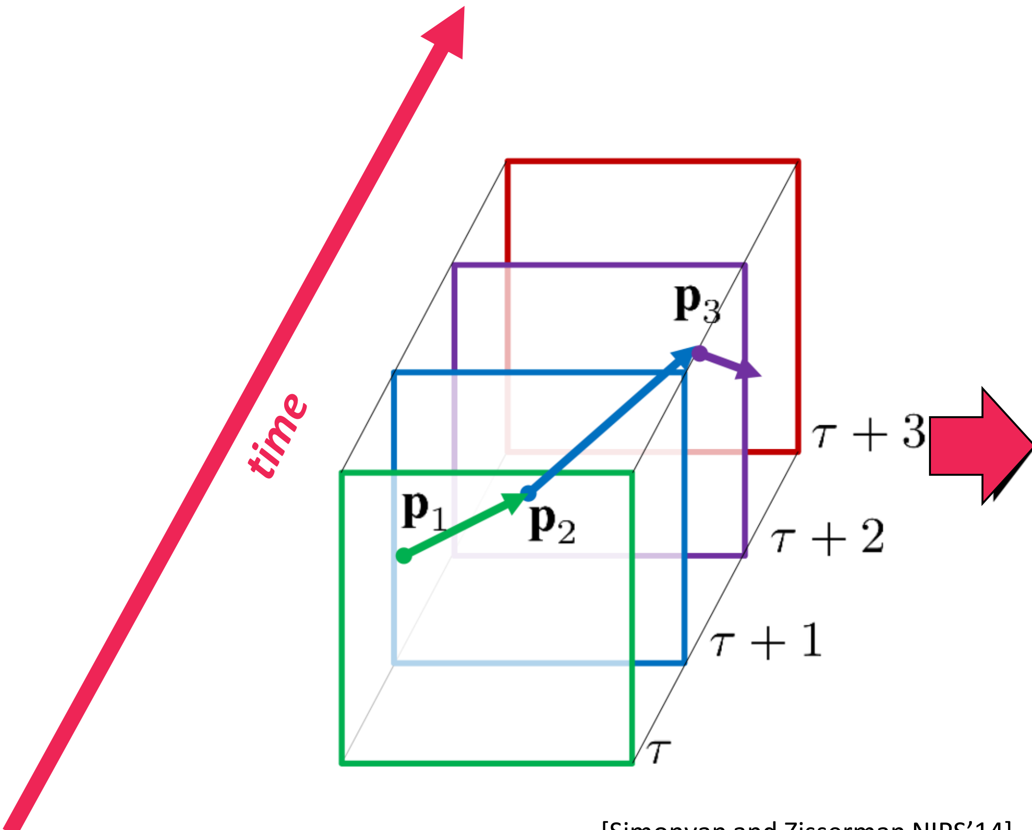


[Simonyan and Zisserman NIPS'14]

Trajectory stacking

Stack horizontal and vertical displacement fields $\mathbf{d}$ along the tractory

- Trajectory over several frames, τ, acts as input I_τ to the network



Input I_τ at $\mathbf{p}_1$ represents motion across a trajectory that starts at $\mathbf{p}_1$

[Simonyan and Zisserman NIPS'14]

Empirical evaluation

Datasets

- UCF101: 101 classes, 13K videos, ~180 frames in a vid
- HMDB51: 51 classes, 6.8K videos
- Evaluation protocol: average classification accuracy over 3 train and test splits

[Kuehne et al. ICCV'11]

Empirical evaluation: Overfit prevention

Action recognition datasets are rather small (≈10K videos)

- Many images (=frames) but they are very similar

Spatial net overfit prevention:

- Supervised pre-training on large dataset (ILSVRC 1.2M images in 1000 classes)
- Fine tuning of the softmax layer using the video frames

Temporal net:

- Multitask Learning (train a model based on several loss functions for different tasks)
  - Each task has its own (softmax) loss
  - Total loss ≈ sum over task losses
  - One Task = UCF101 classification, other task = HMDB51 classification
  - Datasets are not merged, however backprop operates on the sum of both losses

[Simonyan and Zisserman NIPS'14]

Empirical evaluation

Spatial net:

- Pre trained network is better than scratch
- Fine tuning the whole net is similar to re-train just the last layer
- Training from scratch is unpractical even with high dropout

Table 1: **Individual ConvNets accuracy on UCF-101 (split 1).**

(a) **Spatial ConvNet.**

Training setting	Dropout ratio	
	0.5	0.9
From scratch	42.5%	52.3%
Pre-trained + fine-tuning	70.8%	**72.8%**
Pre-trained + last layer	**72.7%**	59.9%

(b) **Temporal ConvNet.**

Input configuration	Mean subtraction	
	off	on
Single-frame optical flow ($L = 1$)	-	73.9%
Optical flow stacking (1) ($L = 5$)	-	80.4%
Optical flow stacking (1) ($L = 10$)	79.9%	**81.0%**
Trajectory stacking (2)($L = 10$)	79.6%	80.2%
Optical flow stacking (1)($L = 10$), bi-dir.	-	**81.2%**

[Simonyan and Zisserman NIPS'14]

Empirical evaluation

Temporal net:

- Flow or trajectory stacking improves significantly ($\approx 7\%$ improvement in accuracy)
- Mean subtraction brings only minor improvements

Table 1: **Individual ConvNets accuracy on UCF-101 (split 1).**

(a) **Spatial ConvNet.**

Training setting	Dropout ratio	
	0.5	0.9
From scratch	42.5%	52.3%
Pre-trained + fine-tuning	70.8%	**72.8%**
Pre-trained + last layer	**72.7%**	59.9%

(b) **Temporal ConvNet.**

Input configuration	Mean subtraction	
	off	on
Single-frame optical flow ($L = 1$)	-	73.9%
Optical flow stacking (1) ($L = 5$)	-	80.4%
Optical flow stacking (1) ($L = 10$)	79.9%	**81.0%**
Trajectory stacking (2)($L = 10$)	79.6%	80.2%
Optical flow stacking (1)($L = 10$), bi-dir.	-	**81.2%**

[Simonyan and Zisserman NIPS'14]

Empirical evaluation

Table 2: **Temporal ConvNet accuracy on HMDB-51 (split 1 with additional training data).**

Training setting	Accuracy
Training on HMDB-51 without additional data	46.6%
Fine-tuning a ConvNet, pre-trained on UCF-101	49.0%
Training on HMDB-51 with classes added from UCF-101	52.8%
Multi-task learning on HMDB-51 and UCF-101	**55.4%**

Temporal net, multi task learning:

- Additional data improves recognition
- Fine tuning a model that is trained only on a small dataset is challenging
  - Small learning rate ➔ Net stays spezialized on the original data
  - Large learning rate ➔ Net overfits the new dataset
- Combining both datasets works better than fine tuning approach
- Multi-task learning works best
  - (But backpropagation operates on both datasets simultaneously)

[Simonyan and Zisserman NIPS'14]

Empirical evaluation: Comparison with the state of the art

Table 4: **Mean accuracy (over three splits) on UCF-101 and HMDB-51.**

Method	UCF-101	HMDB-51
Improved dense trajectories (IDT) [26, 27]	85.9%	57.2%
IDT with higher-dimensional encodings [20]	**87.9%**	61.1%
IDT with stacked Fisher encoding [21] (based on Deep Fisher Net [23])	-	**66.8%**
Spatio-temporal HMAX network [11, 16]	-	22.8%
"Slow fusion" spatio-temporal ConvNet [14]	65.4%	-
Spatial stream ConvNet	73.0%	40.5%
Temporal stream ConvNet	83.7%	54.6%
Two-stream model (fusion by averaging)	86.9%	58.0%
Two-stream model (fusion by SVM)	**88.0%**	**59.4%**

- Spatial / temporal stream network is better than spatiotemporal processing
- Hand-crafted features are still better
- Datasets too small?

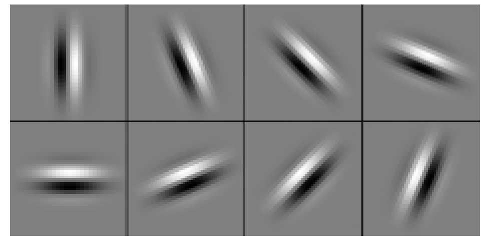
[Simonyan and Zisserman NIPS'14]

Compare: Hand-Crafted Descriptors

Image Pixels (HOG /SIFT…

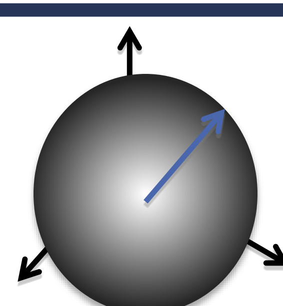
Image Motion (HOF / MBH …)

Apply Gabor filters



→ **Is the network able to generalize hand-crafted representations?**

Normalize to unit length

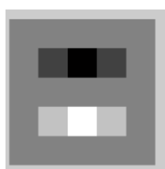


Feature Vector

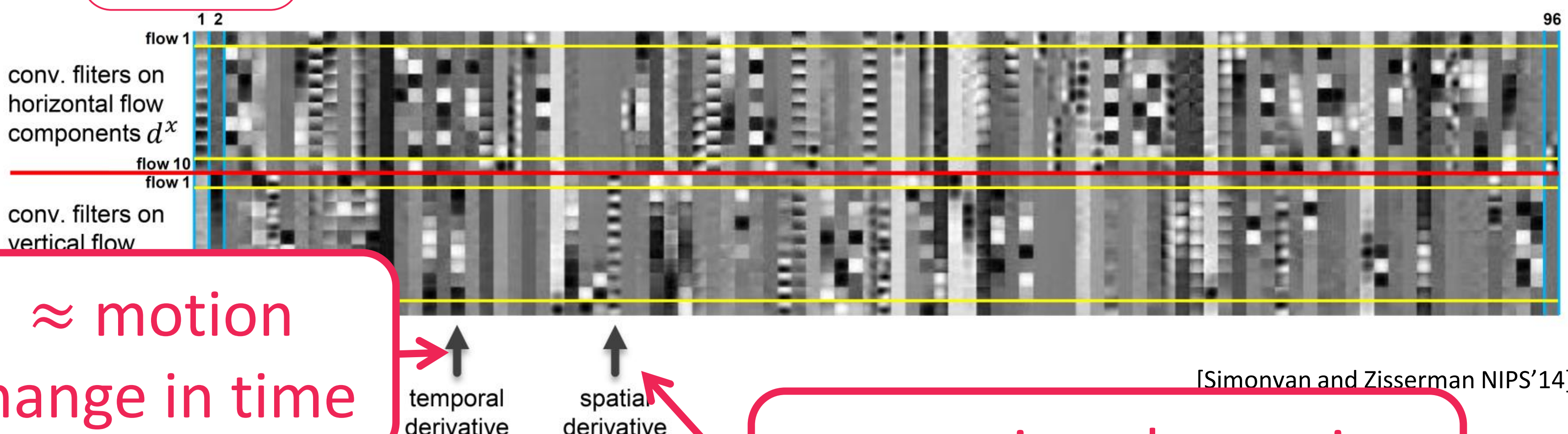
Adapted from R. Fergus

Relation to convolutional networks

- Trajectory over several frames acts as input to the network
- HOG (Histograms of Oriented Gradients) ≈ single layer in the spatial network
- HOF (Histograms of Oriented Gradients) ≈ single layer in the temporal network
- MBH (Histograms of Oriented Gradients) ≈ single layer in the temporalnetwork



First order derivative filter



≈ motion change in time

≈ motion change in space ≈ MBH

[Simonyan and Zisserman NIPS'14]

Figure 4: **First-layer convolutional filters lea** … sation is split into 96 columns and 20 rows: each col … input channel.

Summary

- Convolutional Networs (ConvNets) for Image Classification
  - Overall architecture defines operations in each layer
  - Visualizations improve results on ImageNet
  - Fine-tuning on other datasets helps
- Representations for Video Classification
  - Hand-designed features are still competitive
  - Straightforward application of spatiotemporal ConvNets performs worse
  - Two-stream ConvNets are able to generalize hand-crafted representations

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

M. Zeiler & R. Fergus, *Visualizing and Understanding Convolutional Networks*, ECCV, 2014

Wang et al., *Action Recognition by Dense Trajectories,* CVPR 2011.

Karpathy et al., *Large-scale Video Classification with Convolutional* Neural Networks, CVPR 2014

K. Simonyan & A. Zisserman, *Two-Stream Convolutional Networks for Action Recognition in Videos,* NIPS 2014