detail from the visualization of ResNet-18 // Graphcore

CMP784

DEEP LEARNING

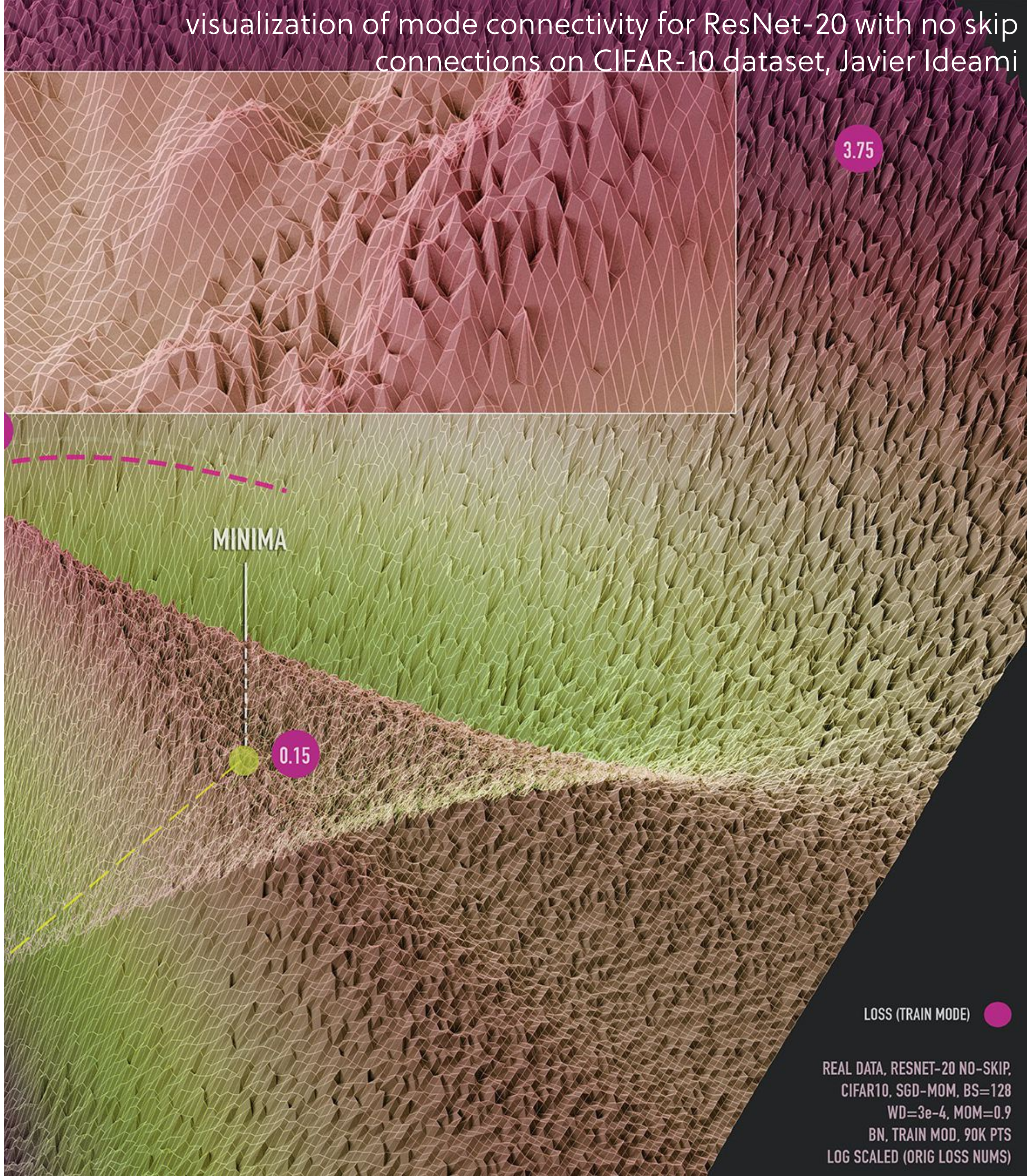
Lecture #05 – Convolutional Neural Networks

HACETTEPE UNIVERSITY COMPUTER VISION LAB

Erkut Erdem // Hacettepe University // Fall 2021

Previously on CMP784

- data preprocessing and normalization
- weight initializations
- ways to improve generalization
- babysitting the learning process
- hyperparameter selection
- optimization



visualization of mode connectivity for ResNet-20 with no skip connections on CIFAR-10 dataset, Javier Ideami

Breaking news!

- Practical 2 is out!
  - —Convolutional Neural Networks
  - —Due Wednesday, Nov. 17, 23:59:59

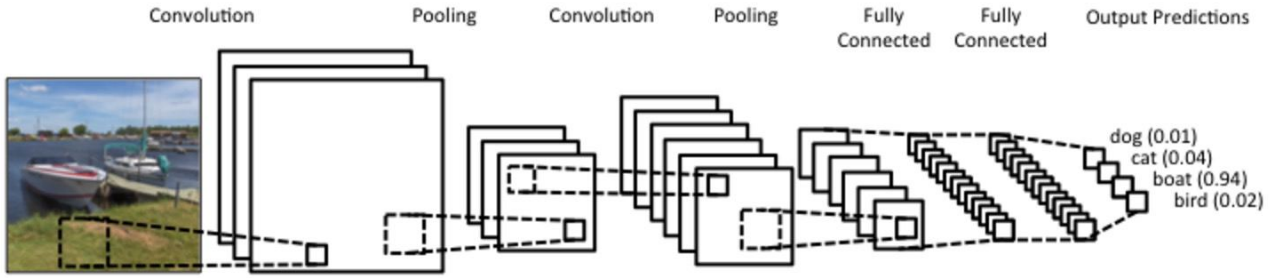


Image credit: wildml.com

- Project proposals is due Nov. 3!
  - about a half page
  - the research topic to be investigated,
  - what data you will use,
  - design overview,
  - a list of key readings.

  Note: The project should be done <u>in pairs</u>.


Deadlines in the syllabus are closer than they appear

Lecture Overview

- convolution layer
- pooling layer
- revolution of depth
- design guidelines
- residual connections
- semantic segmentation networks
- object detection networks

Disclaimer: Much of the material and slides for this lecture were borrowed from

— Andrea Vedaldi's tutorial on Convolutional Networks for Computer Vision Applications

— Kaiming He's ICML 2016 tutorial on Deep Residual Networks: Deep Learning Gets Way Deeper

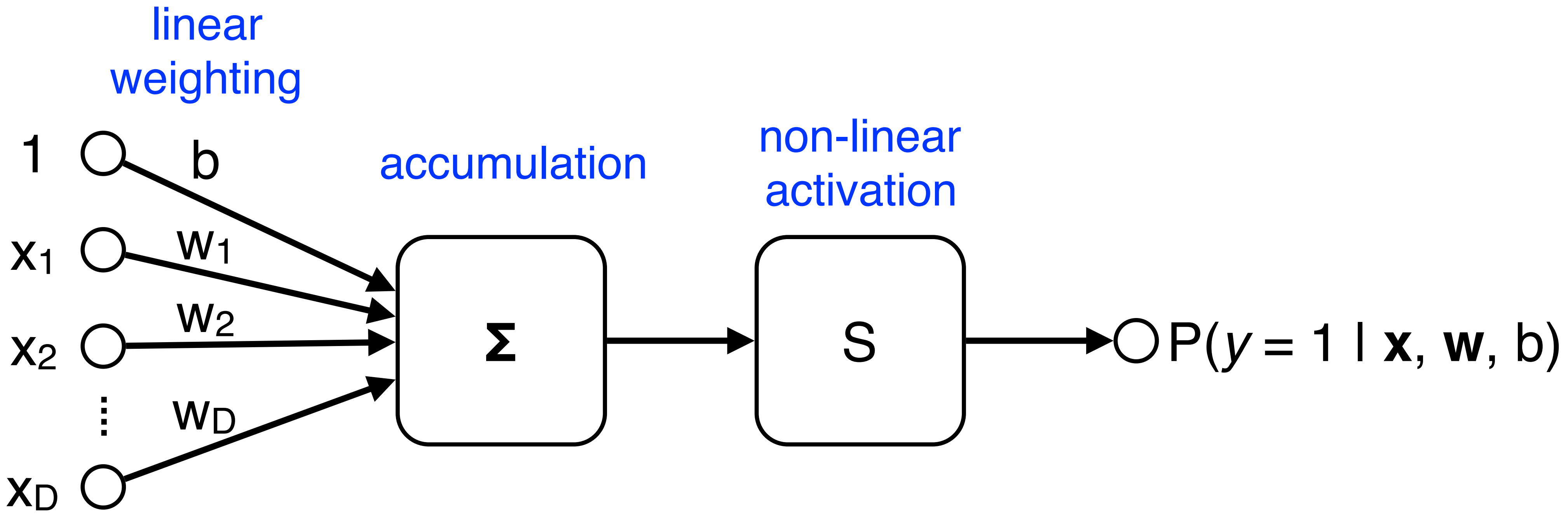
— Ross Girshick's talk on The Past, Present, and Future of Object Detection

— Fei-Fei Li, Andrej Karpathy and Justin Johnson's CS231n class

Perceptron

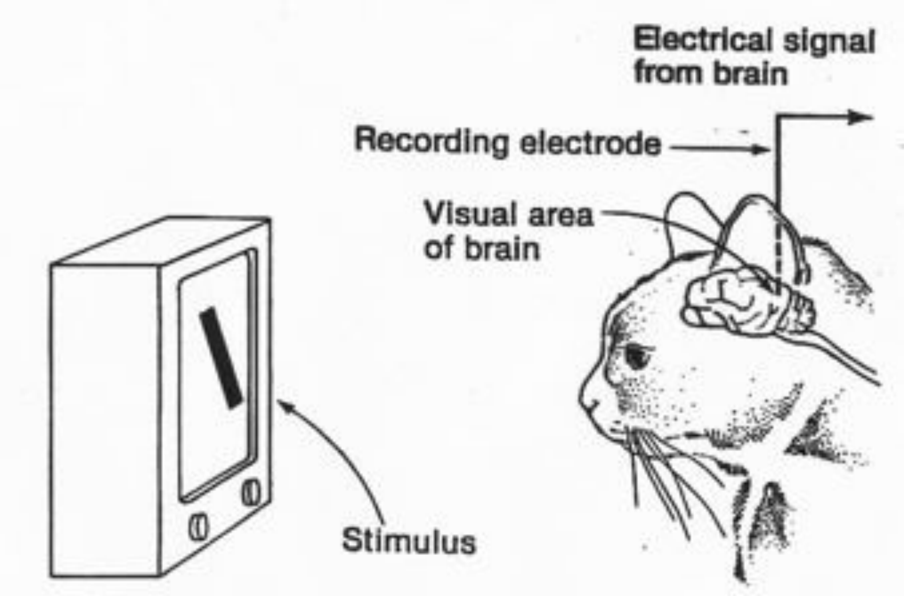
[Rosenblatt 57]

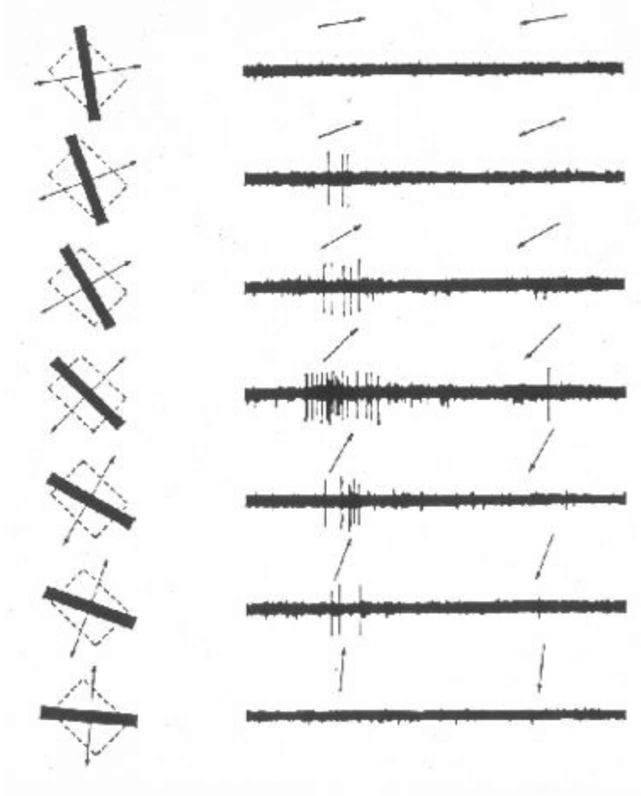
- The goal is estimating the posterior probability of the binary label y of a vector **x**:

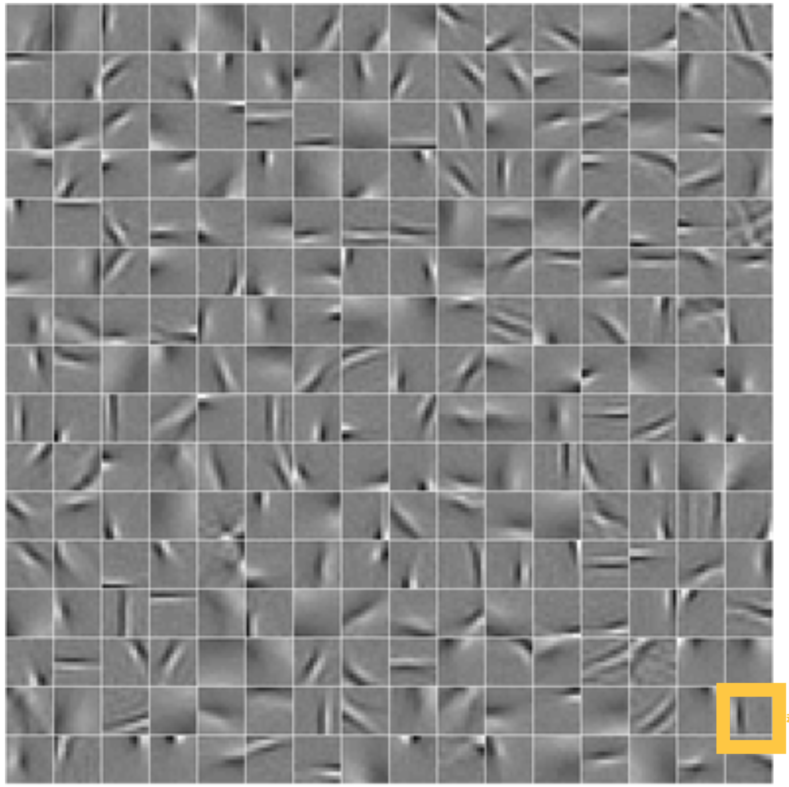


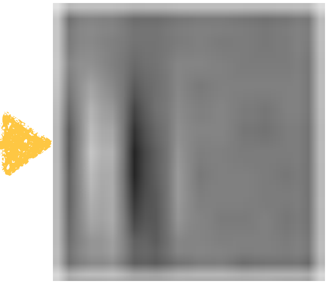
Discovery of oriented cells in the visual cortex

[Hubel and Wiesel 59]



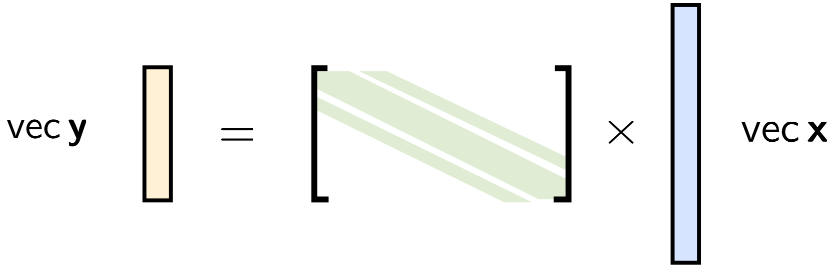






oriented filter

Convolution



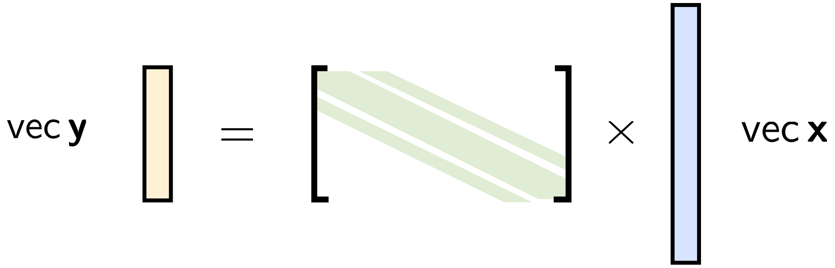
- Convolution = Spatial filtering

$$(a \star b)[i, j] = \sum_{i', j'} a[i', j'] b[i - i', j - j']$$

- Different filters (weights) reveal a different characteristics of the input.

$$* \, {}^{1/8} \begin{array}{|c|c|c|} \hline 0 & 1 & 0 \\ \hline 1 & 4 & 1 \\ \hline 0 & 1 & 0 \\ \hline \end{array}$$

Convolution



- Convolution = Spatial filtering

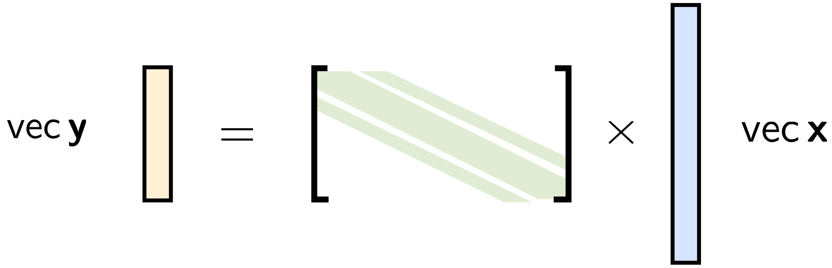
$$(a \star b)[i,j] = \sum_{i',j'} a[i',j']b[i-i',j-j']$$

- Different filters (weights) reveal a different characteristics of the input.

$*$

0	-1	0
-1	4	-1
0	-1	0

Convolution



- Convolution = Spatial filtering

$$(a \star b)[i, j] = \sum_{i', j'} a[i', j'] b[i - i', j - j']$$

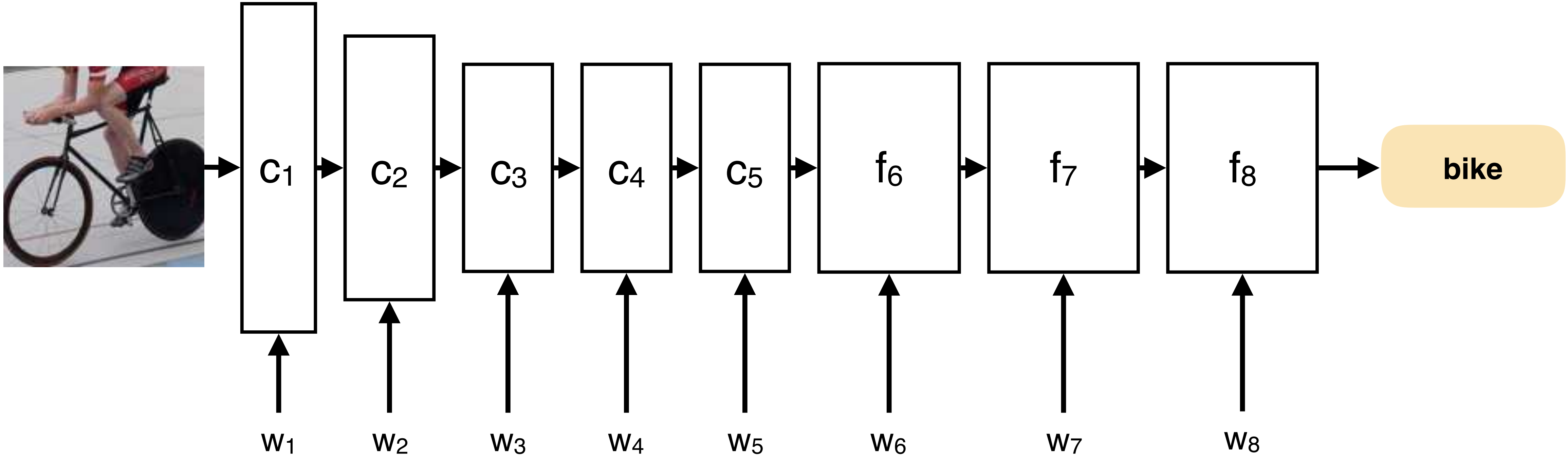
- Different filters (weights) reveal a different characteristics of the input.

$*$

1	0	-1
2	0	-2
1	0	-1

Convolutional Neural Networks in a Nutshell

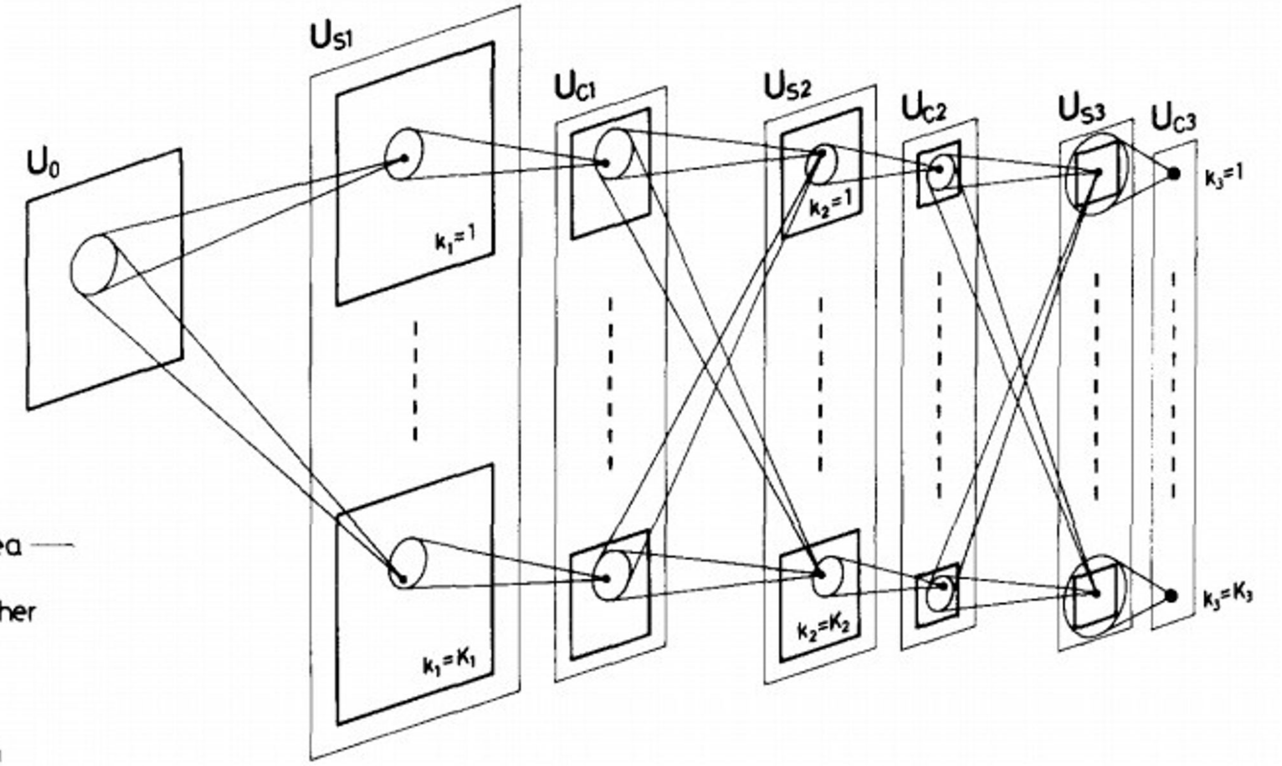
- A neural network model that consists of a sequence of local & translation invariant layers
  - Many identical copies of the same neuron: Weight/parameter sharing
  - Hierarchical feature learning

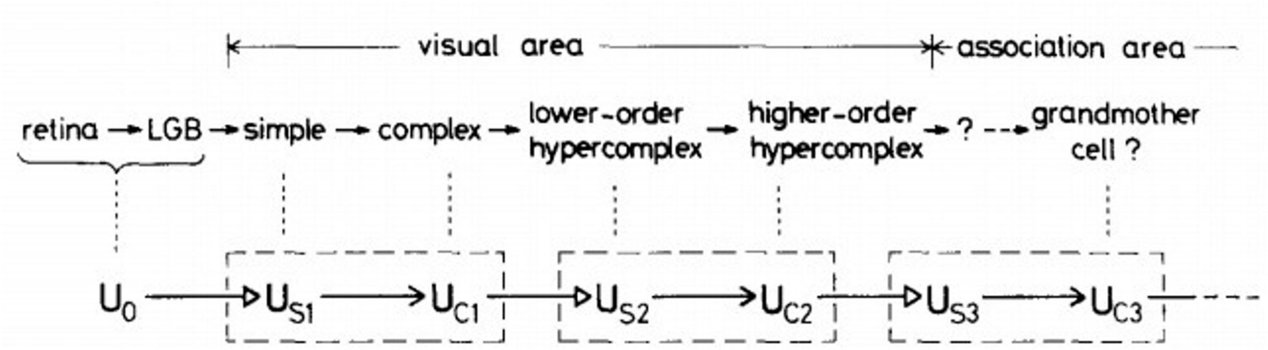


A. Krizhevsky, I. Sutskever, and G. E. Hinton. **Imagenet classification with deep convolutional neural networks**. In NIPS 2012.

A bit of history

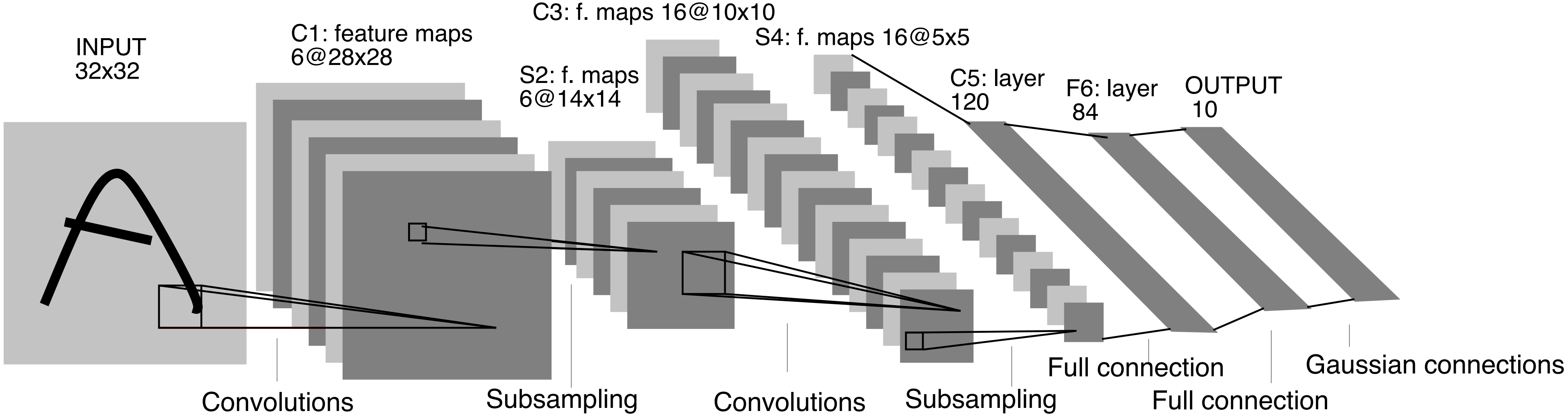
- Neocognitron model by Fukushima (1980)
- The first convolutional neural network (CNN) model
- so-called “sandwich” architecture
  - simple cells act like filters
  - complex cells perform pooling
- Difficult to train
  - No backpropagation yet





A bit of history

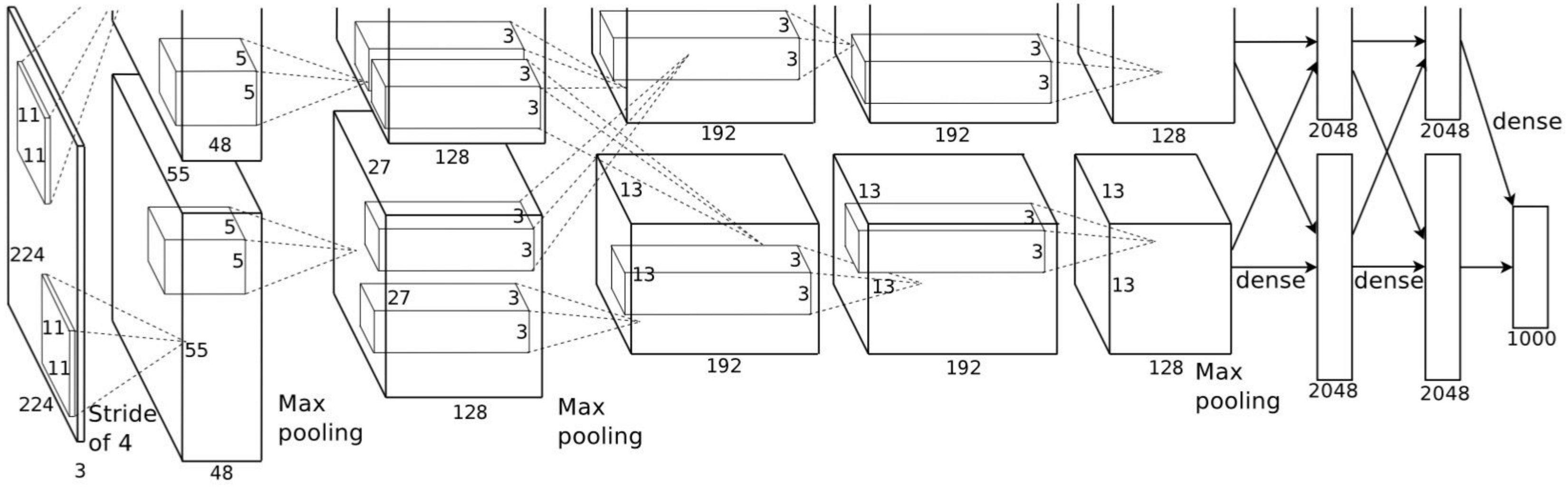
- LeNet-5 model



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.

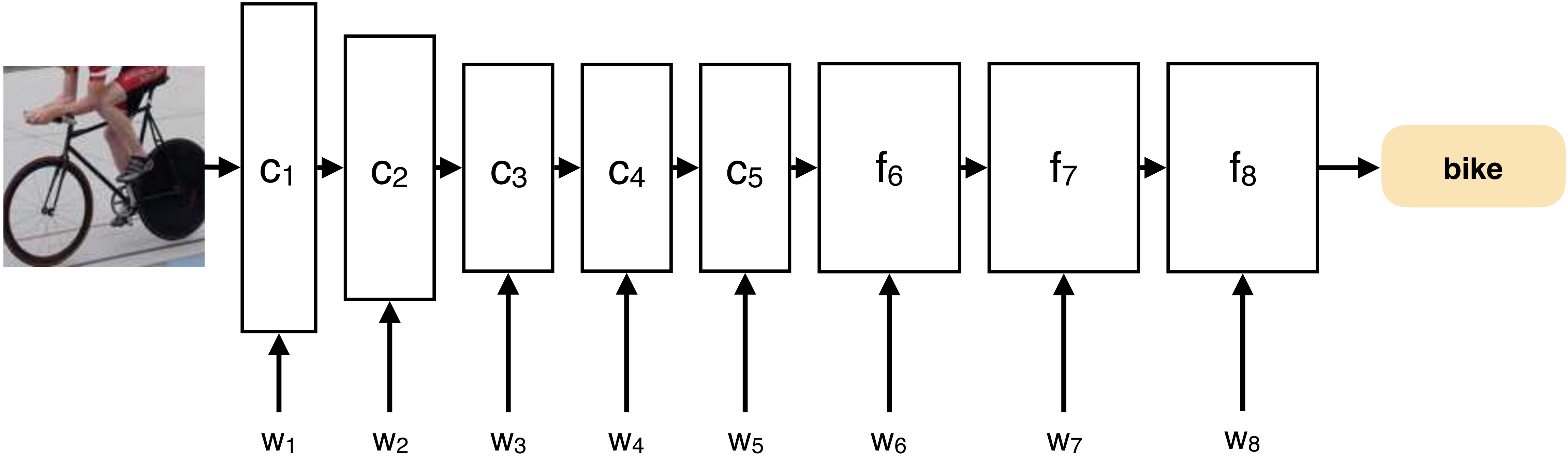
A bit of history

- AlexNet model



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

Convolutional Neural Network

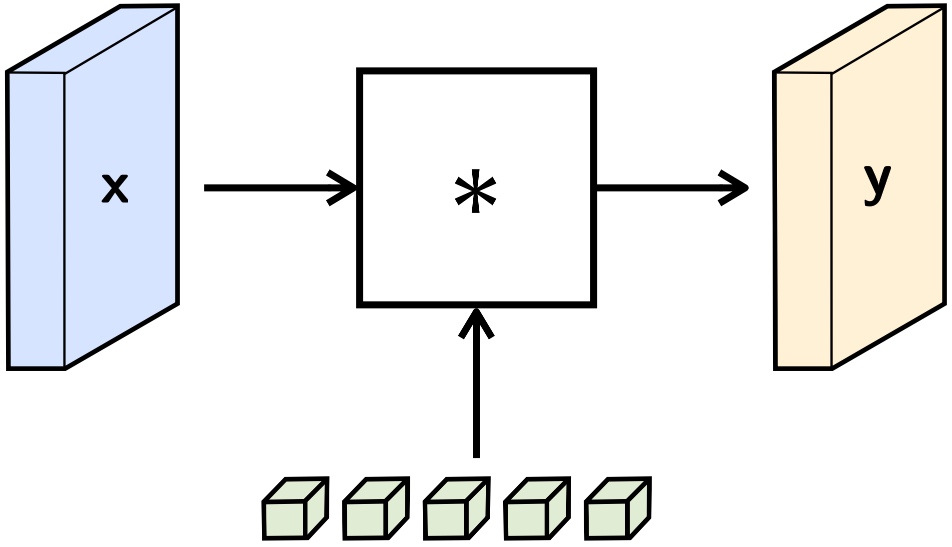


A. Krizhevsky, I. Sutskever, and G. E. Hinton. **Imagenet classification with deep convolutional neural networks**. In NIPS 2012.

Convolutional layer

- Learn a filter bank (a set of filters) once
- Use them over the input data to extract features

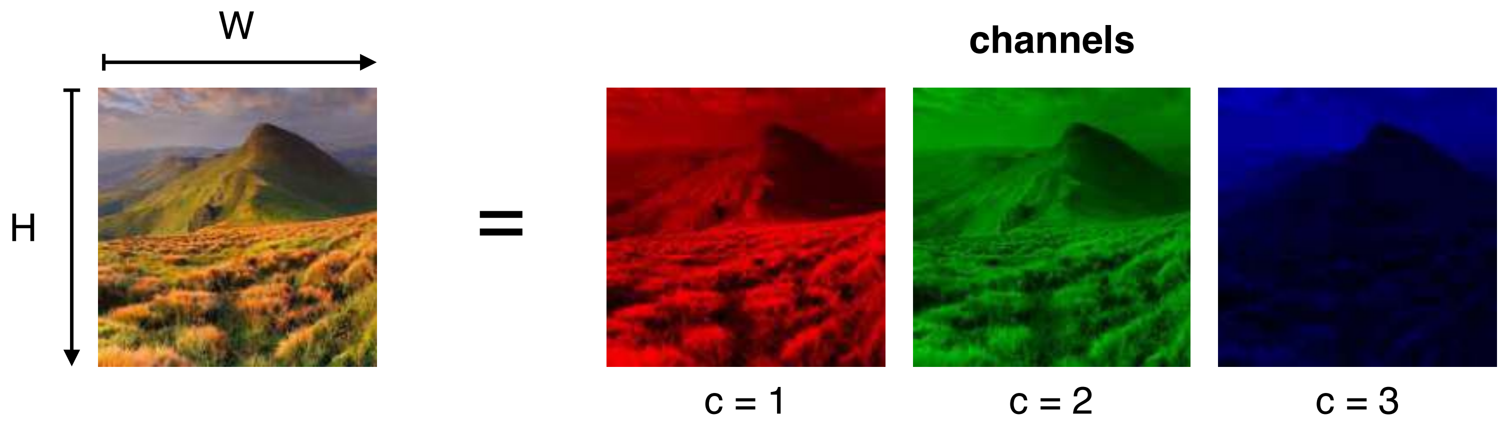
$$\boxed{\mathbf{y} = F * \mathbf{x} + b}$$

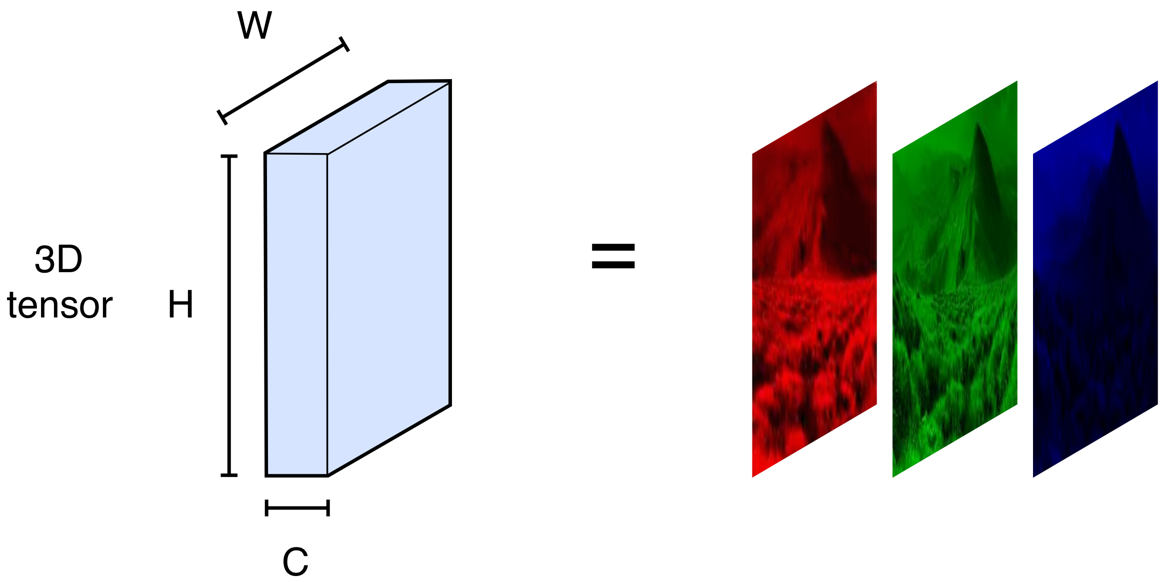


input data $\mathbf{x}$ filter bank F output data $\mathbf{y}$

Data = 3D Tensors

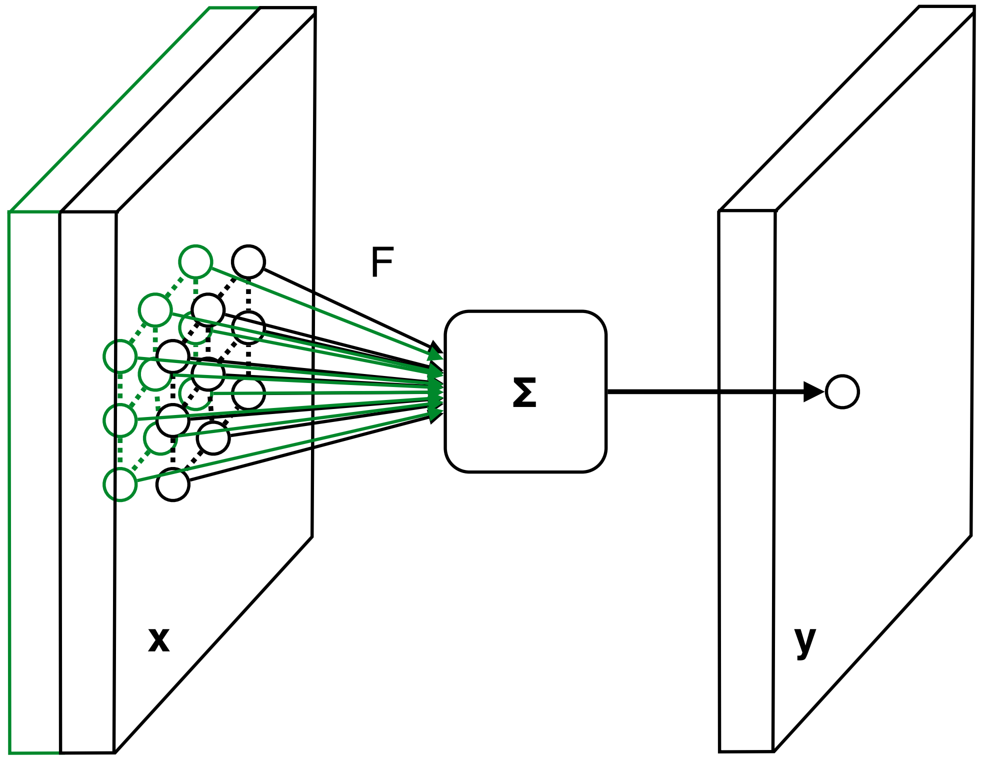
- There is a vector of feature channels (e.g. RGB) at each spatial location (pixel).

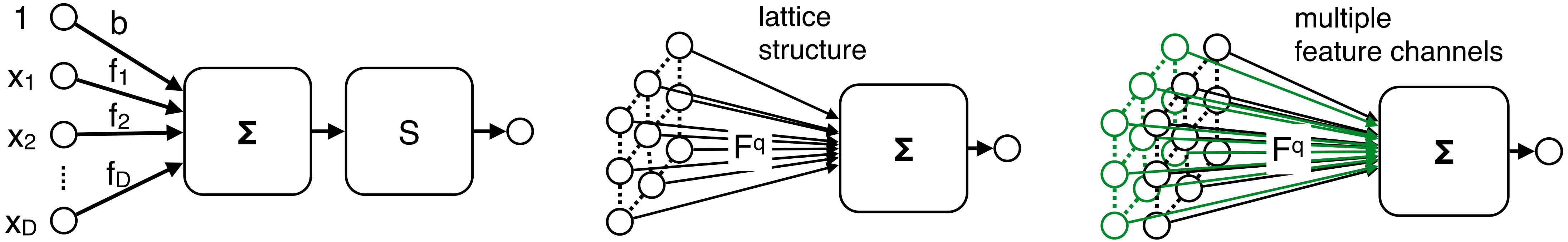




Convolutions with 3D Filters

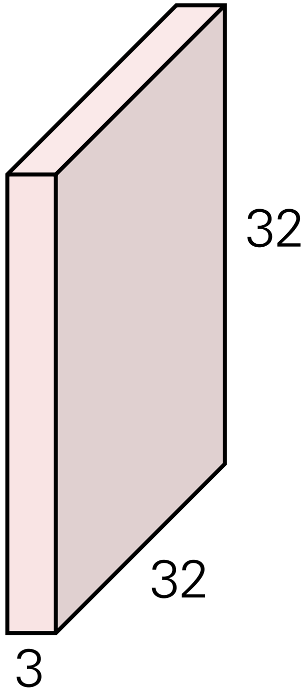
- Each filter acts on multiple input channels
  - **Local**
    Filters look locally
  - **Translation invariant**
    Filters act the same everywhere



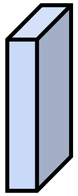


Convolutional Layer

32x32x3 input

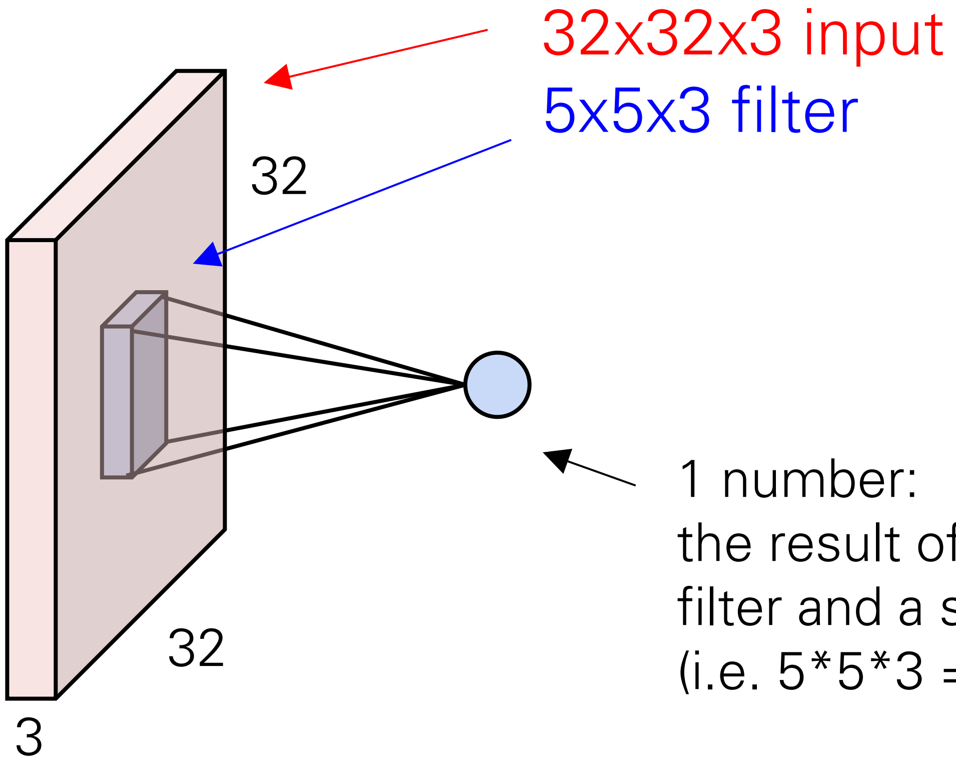


5x5x3 filter



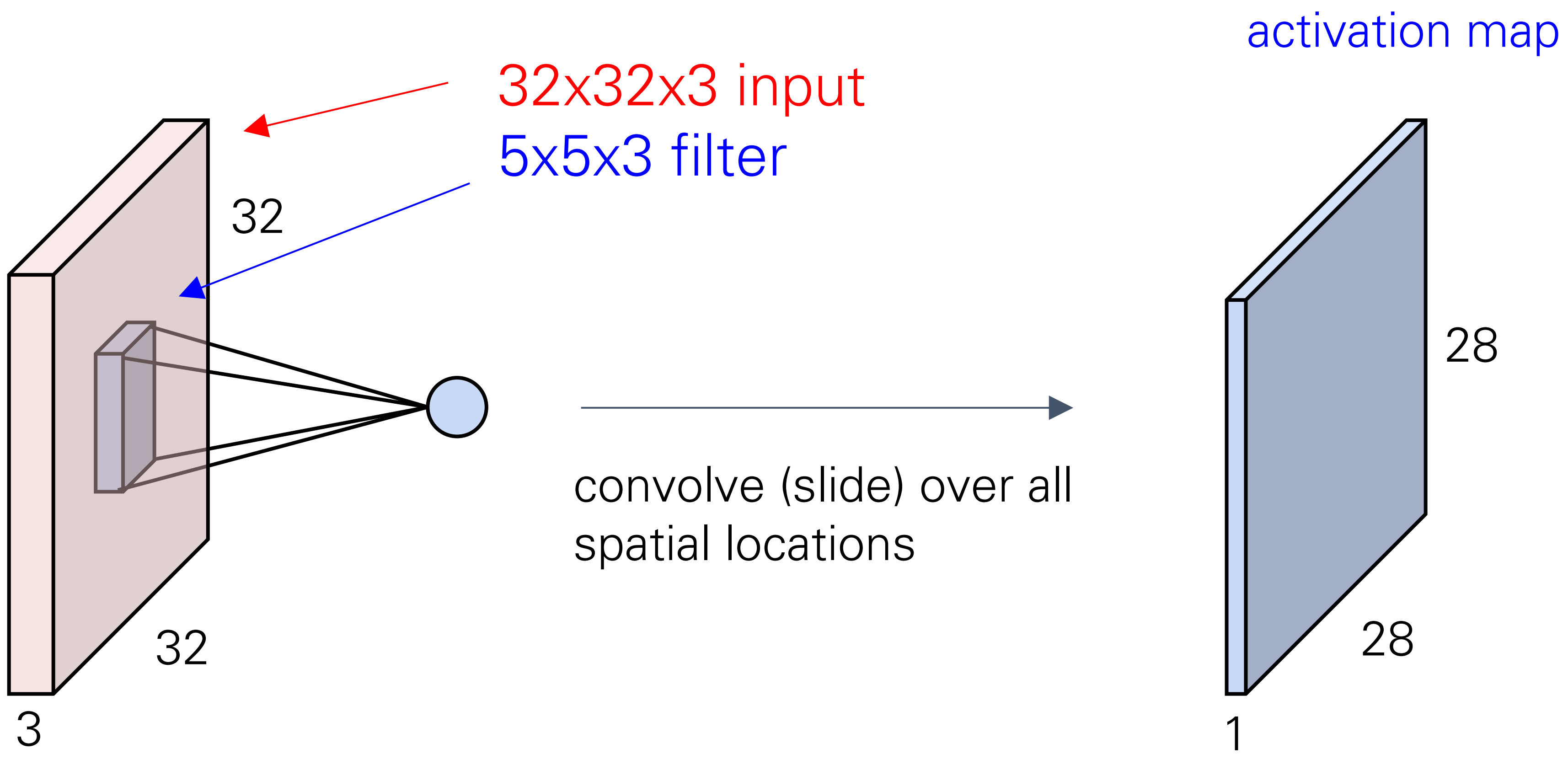
Convolve the filter with the input
i.e. “slide over the image spatially,
computing dot products”

Convolutional Layer



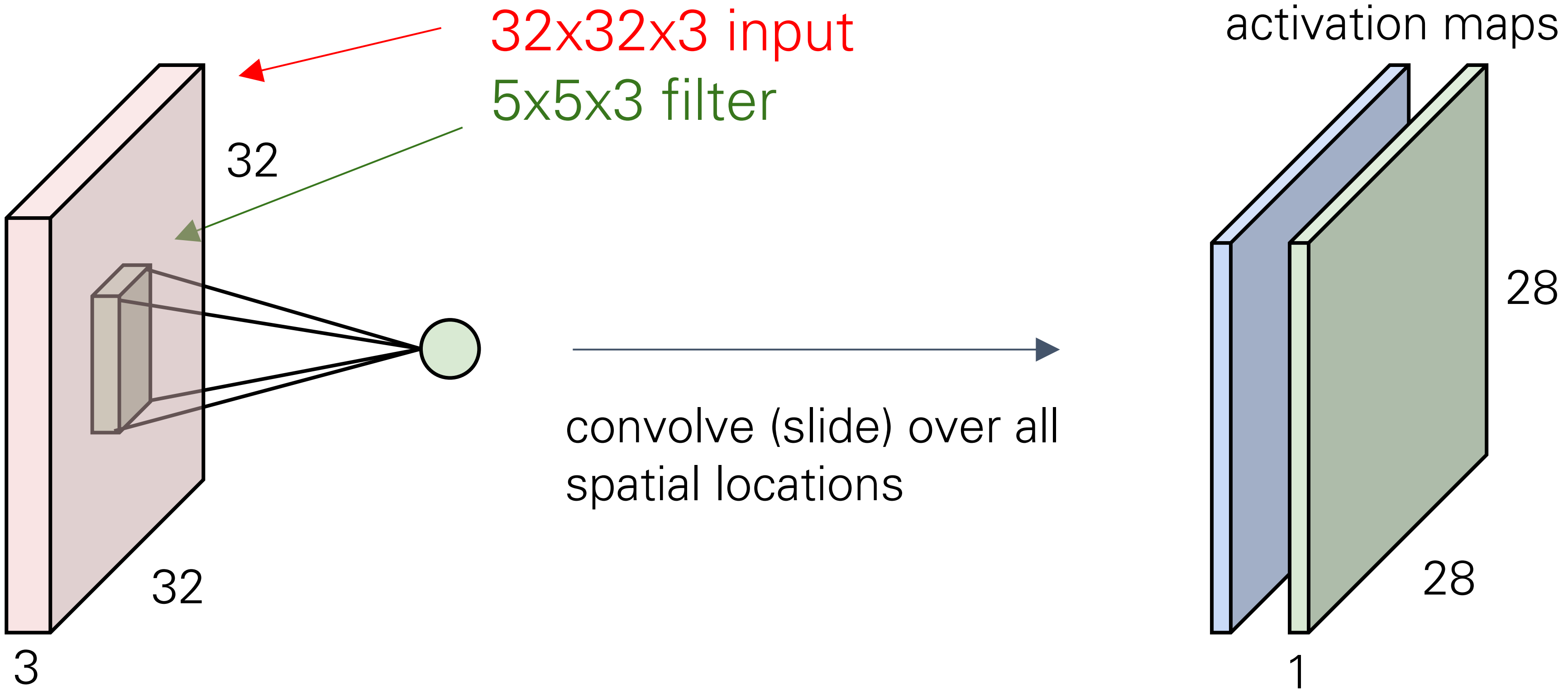
1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the input
(i.e. 5*5*3 = 75-dimensional dot product + bias)

Convolutional Layer



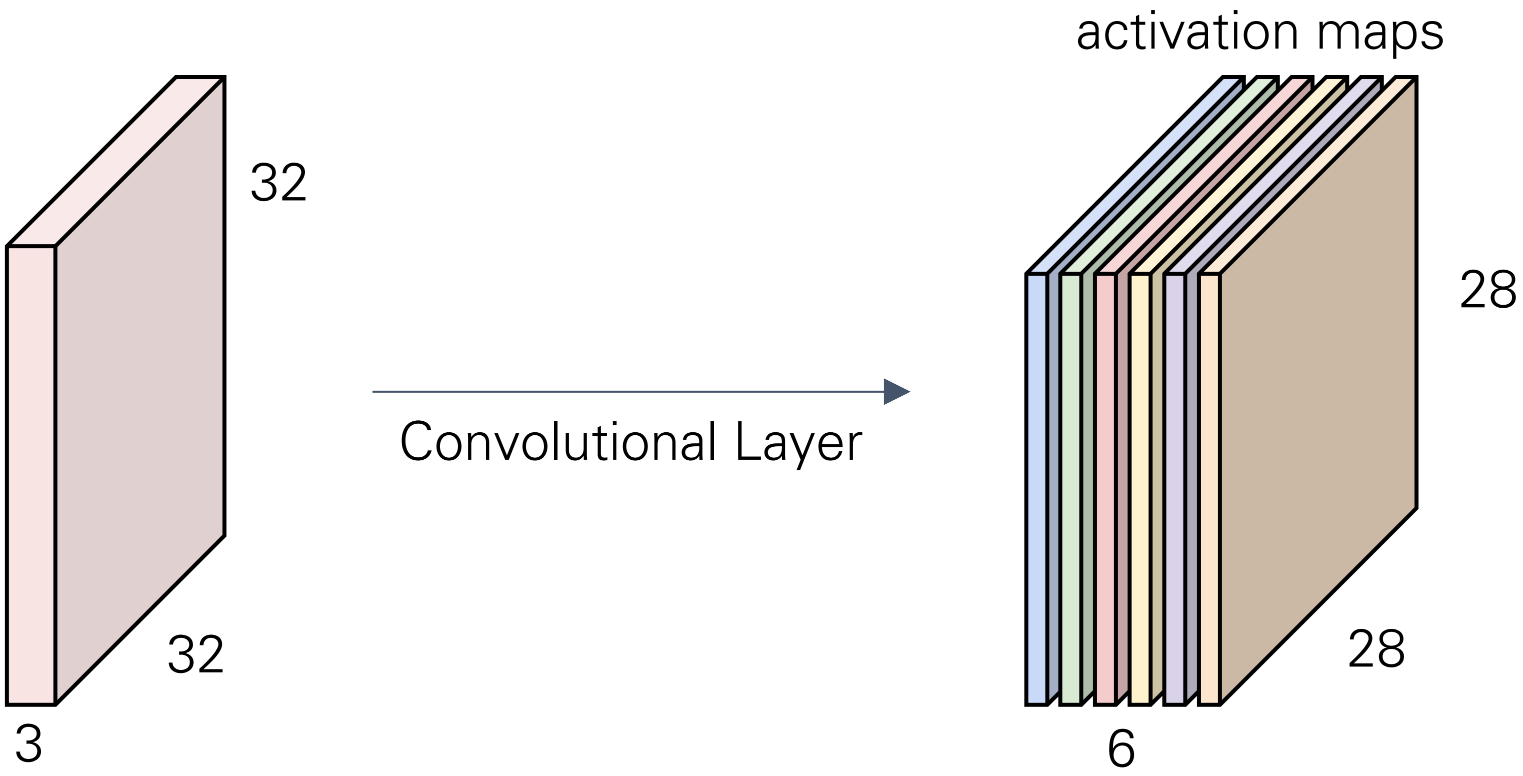
Convolutional Layer

consider a second, green filter



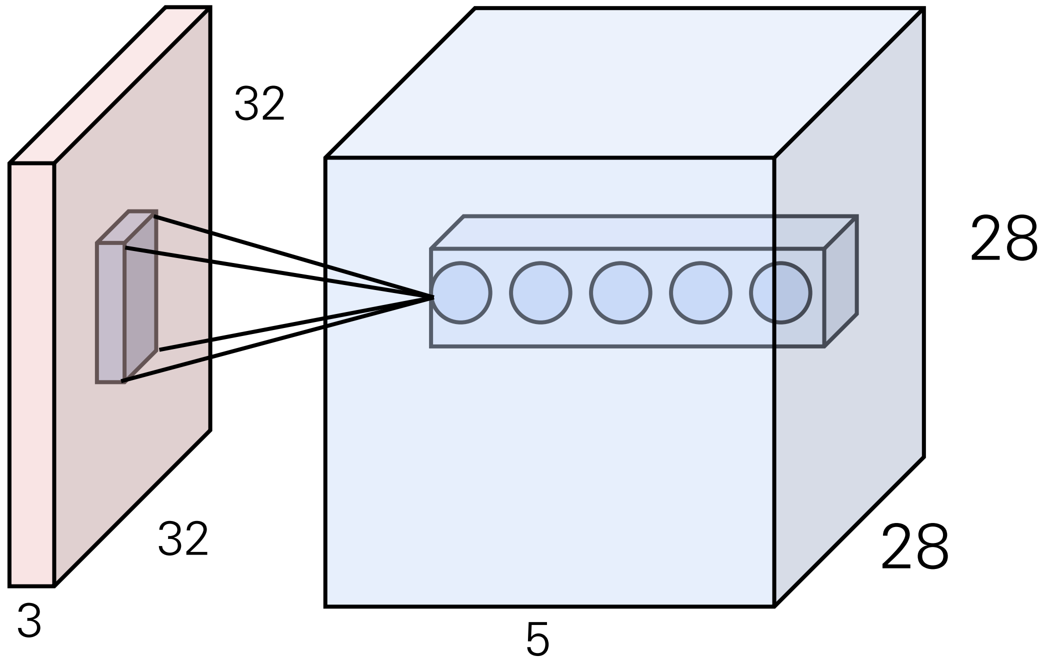
Convolutional Layer

- Multiple filters produce multiple output channels
- For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:

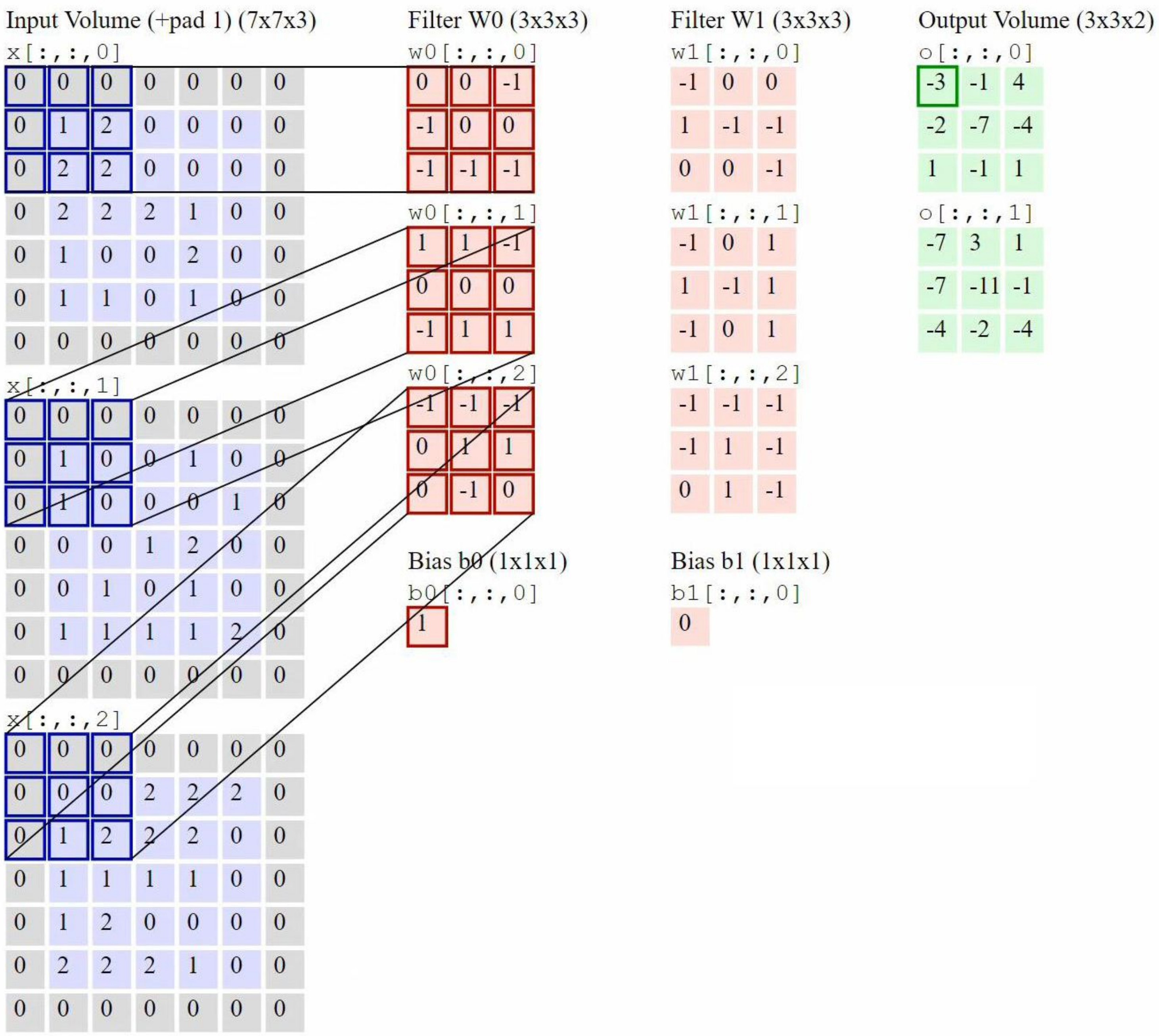


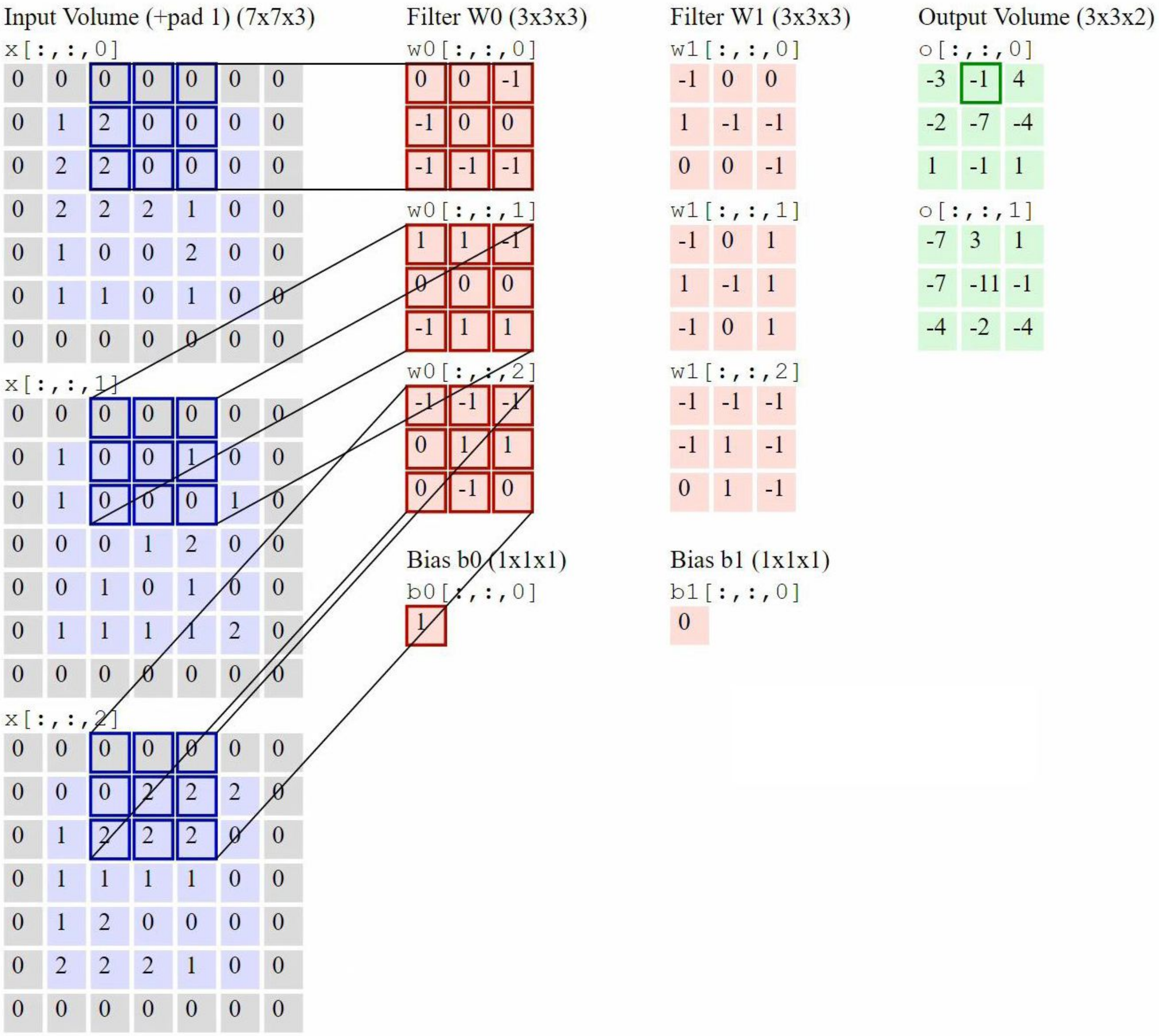
We stack these up to get an output of size 28x28x6.

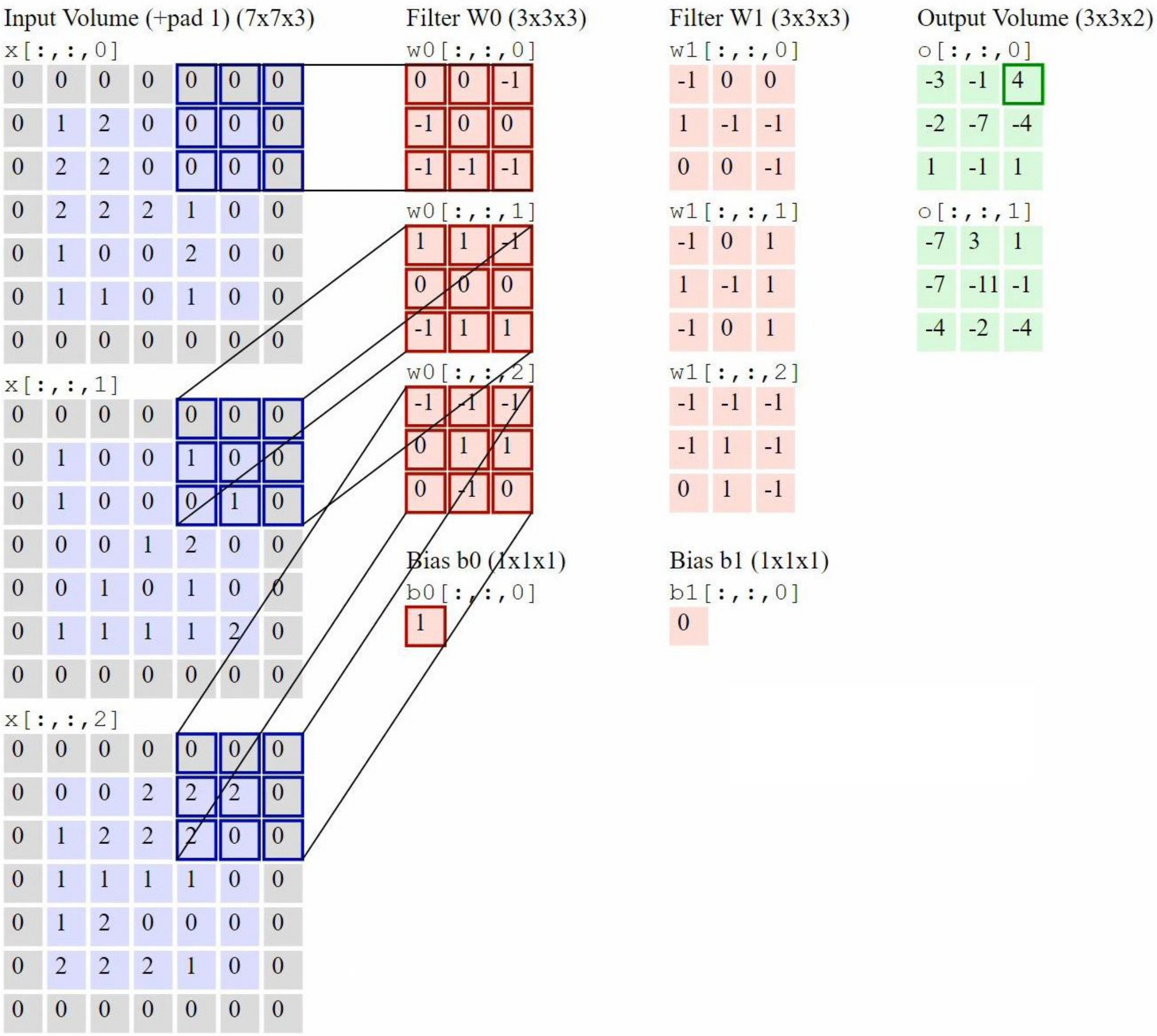
Spatial Arrangement of Output Volume

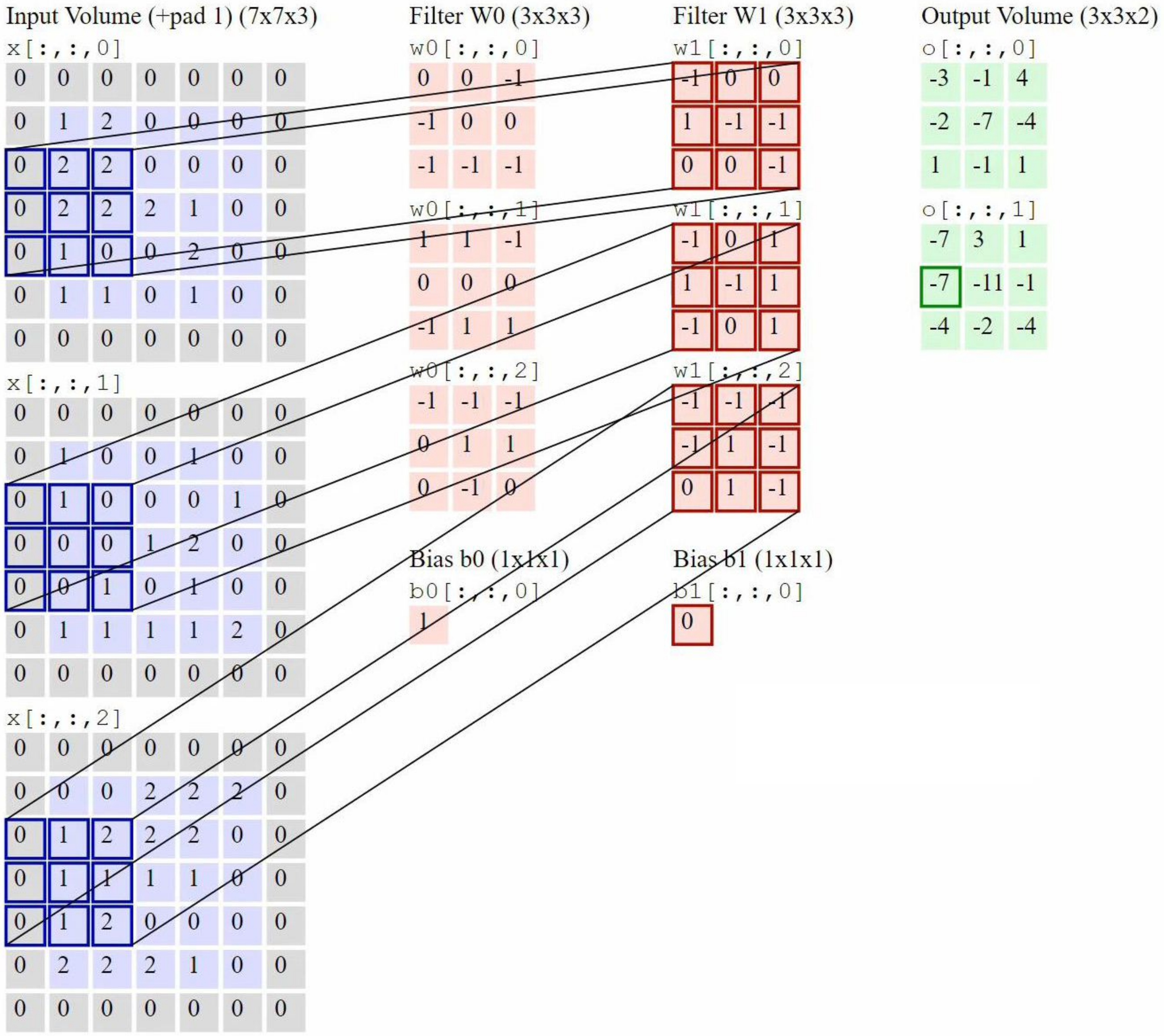


- **Depth:** number of filters
- **Stride:** filter step size (when we "slide" it)
- **Padding:** zero-pad the input

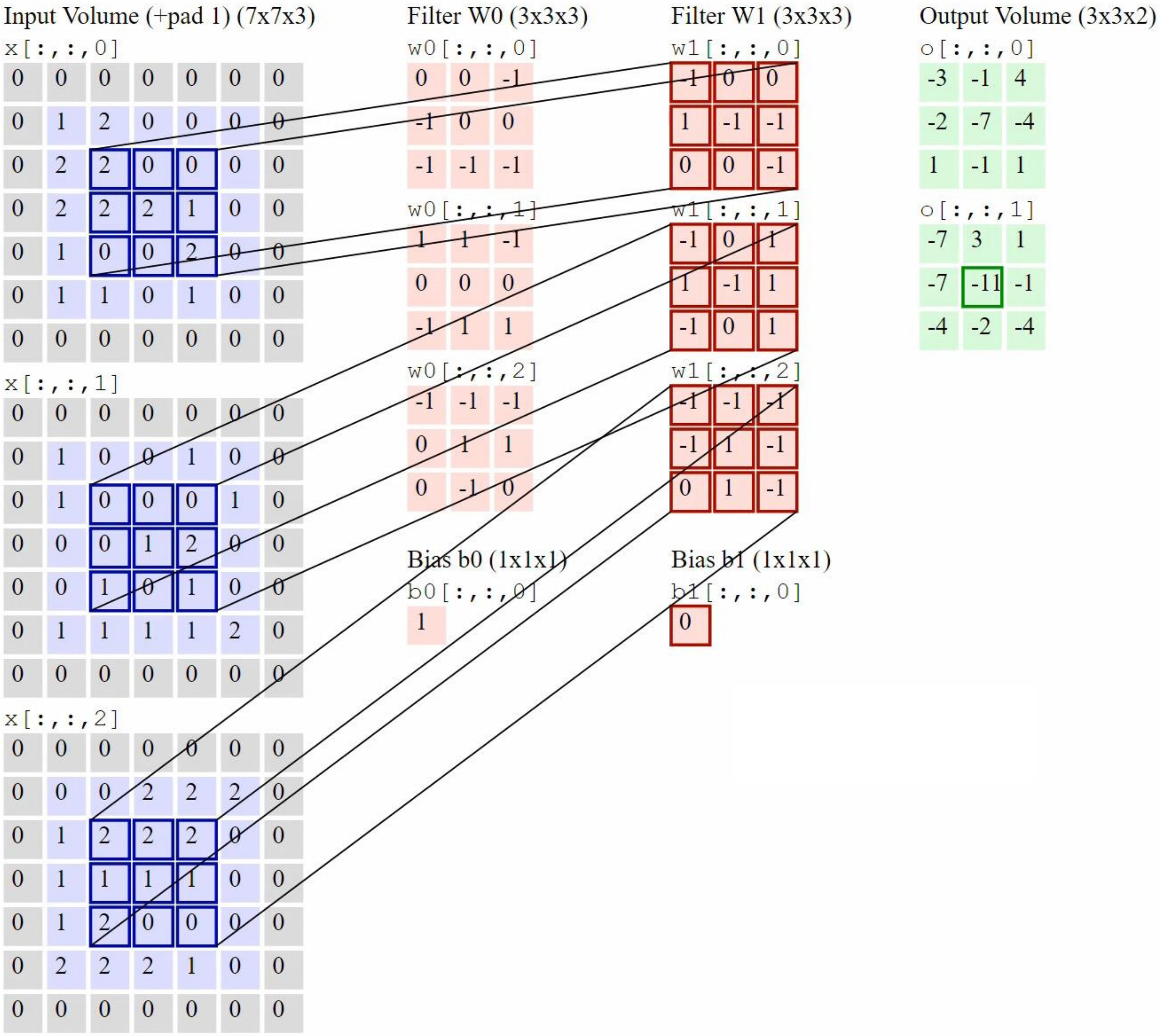

Input Volume (+pad 1) (7x7x3)
x[:,:,0]
0 0 0 0 0 0 0
0 1 2 0 0 0 0
0 2 2 0 0 0 0
0 2 2 2 1 0 0
0 1 0 0 2 0 0
0 1 1 0 1 0 0
0 0 0 0 0 0 0
x[:,:,1]
0 0 0 0 0 0 0
0 1 0 0 1 0 0
0 1 0 0 0 1 0
0 0 0 1 2 0 0
0 0 1 0 1 0 0
0 1 1 1 1 2 0
0 0 0 0 0 0 0
x[:,:,2]
0 0 0 0 0 0 0
0 0 0 2 2 2 0
0 1 2 2 2 0 0
0 1 1 1 1 0 0
0 1 2 0 0 0 0
0 2 2 2 1 0 0
0 0 0 0 0 0 0
Filter W0 (3x3x3)
w0[:,:,0]
0 0 -1
-1 0 0
-1 -1 -1
w0[:,:,1]
1 1 -1
0 0 0
-1 1 1
w0[:,:,2]
-1 -1 -1
0 1 1
0 -1 0
Bias b0 (1x1x1)
b0[:,:,0]
1
Filter W1 (3x3x3)
w1[:,:,0]
-1 0 0
1 -1 -1
0 0 -1
w1[:,:,1]
-1 0 1
1 -1 1
-1 0 1
w1[:,:,2]
-1 -1 -1
-1 1 -1
0 1 -1
Bias b1 (1x1x1)
b1[:,:,0]
0
Output Volume (3x3x2)
o[:,:,0]
-3 -1 4
-2 -7 -4
1 -1 1
o[:,:,1]
-7 3 1
-7 -11 -1
-4 -2 -4

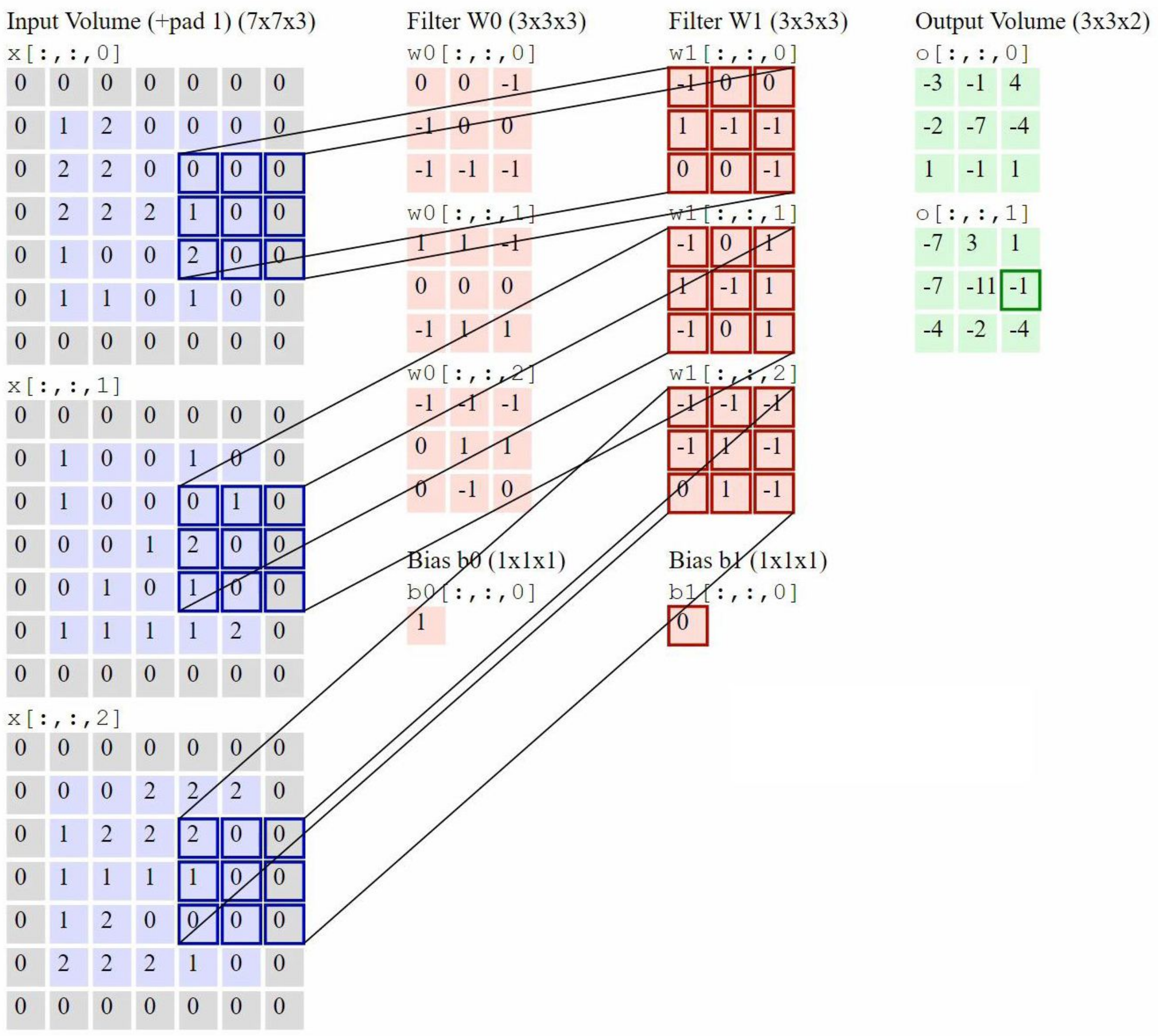

Input Volume (+pad 1) (7x7x3)
x[:,:,0]
0 0 0 0 0 0 0
0 1 2 0 0 0 0
0 2 2 0 0 0 0
0 2 2 2 1 0 0
0 1 0 0 2 0 0
0 1 1 0 1 0 0
0 0 0 0 0 0 0
x[:,:,1]
0 0 0 0 0 0 0
0 1 0 0 1 0 0
0 1 0 0 0 1 0
0 0 0 1 2 0 0
0 0 1 0 1 0 0
0 1 1 1 1 2 0
0 0 0 0 0 0 0
x[:,:,2]
0 0 0 0 0 0 0
0 0 0 2 2 2 0
0 1 2 2 2 0 0
0 1 1 1 1 0 0
0 1 2 0 0 0 0
0 2 2 2 1 0 0
0 0 0 0 0 0 0
Filter W0 (3x3x3)
w0[:,:,0]
0 0 -1
-1 0 0
-1 -1 -1
w0[:,:,1]
1 1 -1
0 0 0
-1 1 1
w0[:,:,2]
-1 -1 -1
0 1 1
0 -1 0
Bias b0 (1x1x1)
b0[:,:,0]
1
Filter W1 (3x3x3)
w1[:,:,0]
-1 0 0
1 -1 -1
0 0 -1
w1[:,:,1]
-1 0 1
1 -1 1
-1 0 1
w1[:,:,2]
-1 -1 -1
-1 1 -1
0 1 -1
Bias b1 (1x1x1)
b1[:,:,0]
0
Output Volume (3x3x2)
o[:,:,0]
-3 -1 4
-2 -7 -4
1 -1 1
o[:,:,1]
-7 3 1
-7 -11 -1
-4 -2 -4

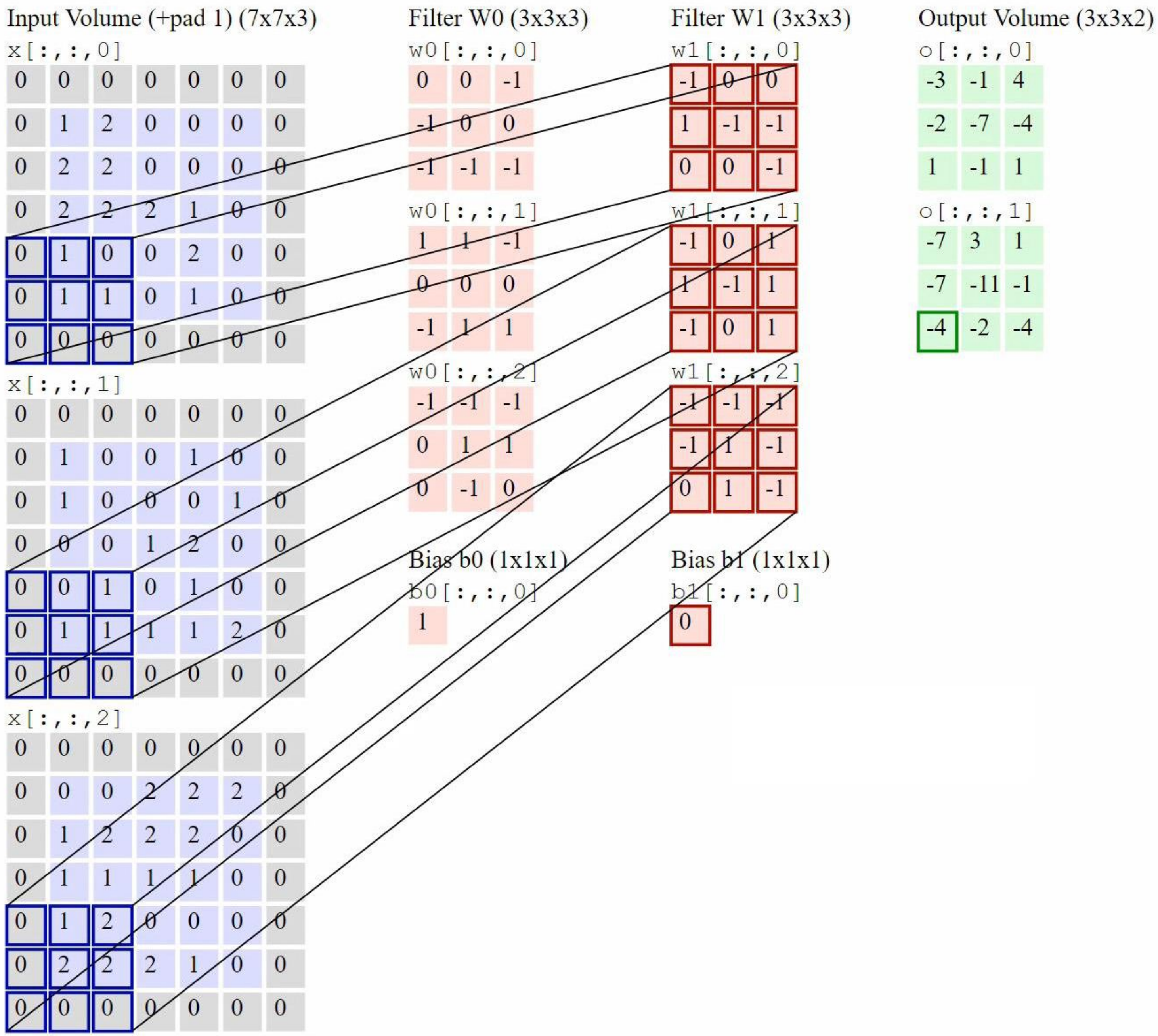

Input Volume (+pad 1) (7x7x3)
x[:,:,0]
0 0 0 0 0 0 0
0 1 2 0 0 0 0
0 2 2 0 0 0 0
0 2 2 2 1 0 0
0 1 0 0 2 0 0
0 1 1 0 1 0 0
0 0 0 0 0 0 0
x[:,:,1]
0 0 0 0 0 0 0
0 1 0 0 1 0 0
0 1 0 0 0 1 0
0 0 0 1 2 0 0
0 0 1 0 1 0 0
0 1 1 1 1 2 0
0 0 0 0 0 0 0
x[:,:,2]
0 0 0 0 0 0 0
0 0 0 2 2 2 0
0 1 2 2 2 0 0
0 1 1 1 1 0 0
0 1 2 0 0 0 0
0 2 2 2 1 0 0
0 0 0 0 0 0 0
Filter W0 (3x3x3)
w0[:,:,0]
0 0 -1
-1 0 0
-1 -1 -1
w0[:,:,1]
1 1 -1
0 0 0
-1 1 1
w0[:,:,2]
-1 -1 -1
0 1 1
0 -1 0
Bias b0 (1x1x1)
b0[:,:,0]
1
Filter W1 (3x3x3)
w1[:,:,0]
-1 0 0
1 -1 -1
0 0 -1
w1[:,:,1]
-1 0 1
1 -1 1
-1 0 1
w1[:,:,2]
-1 -1 -1
-1 1 -1
0 1 -1
Bias b1 (1x1x1)
b1[:,:,0]
0
Output Volume (3x3x2)
o[:,:,0]
-3 -1 4
-2 -7 -4
1 -1 1
o[:,:,1]
-7 3 1
-7 -11 -1
-4 -2 -4

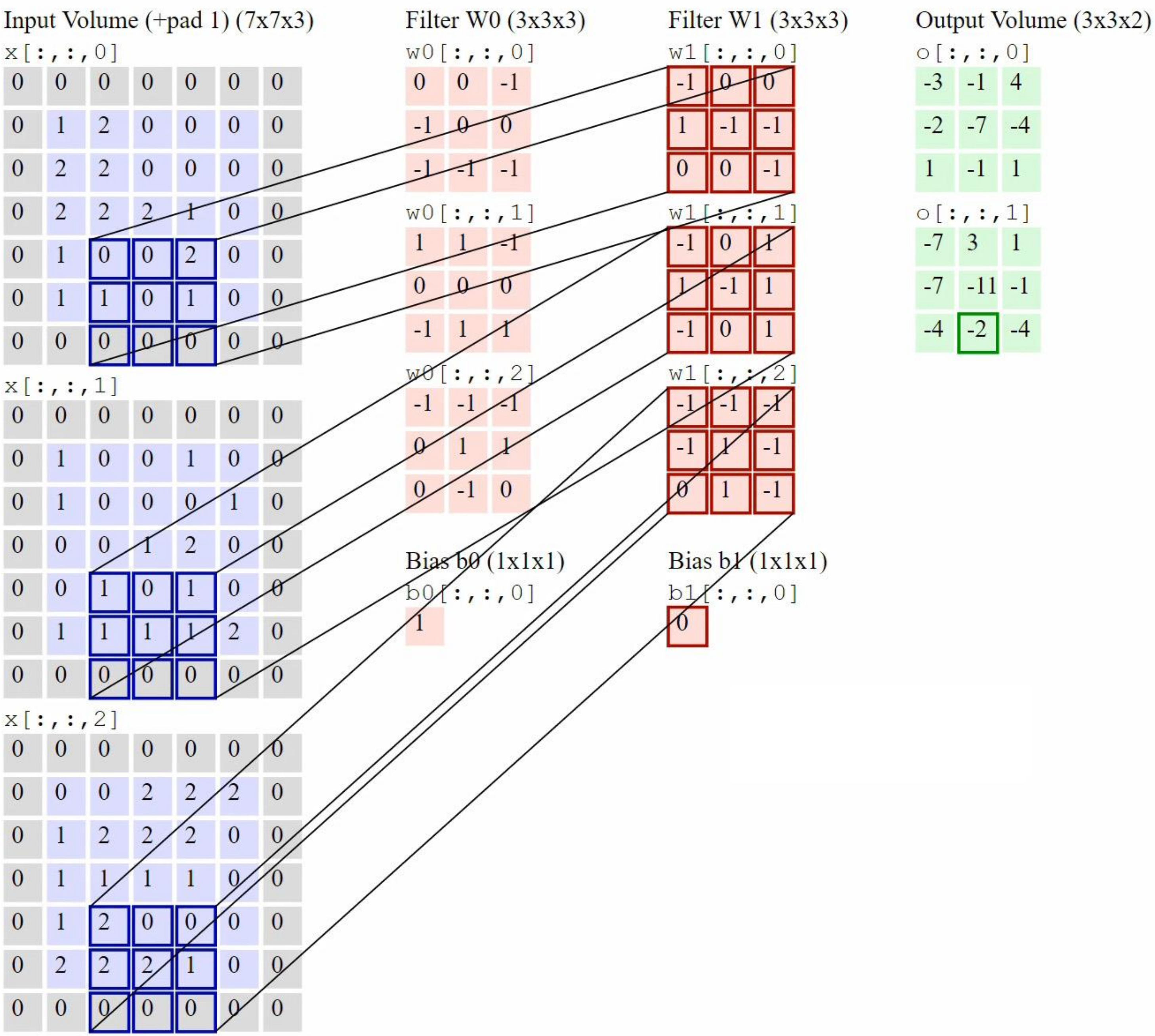


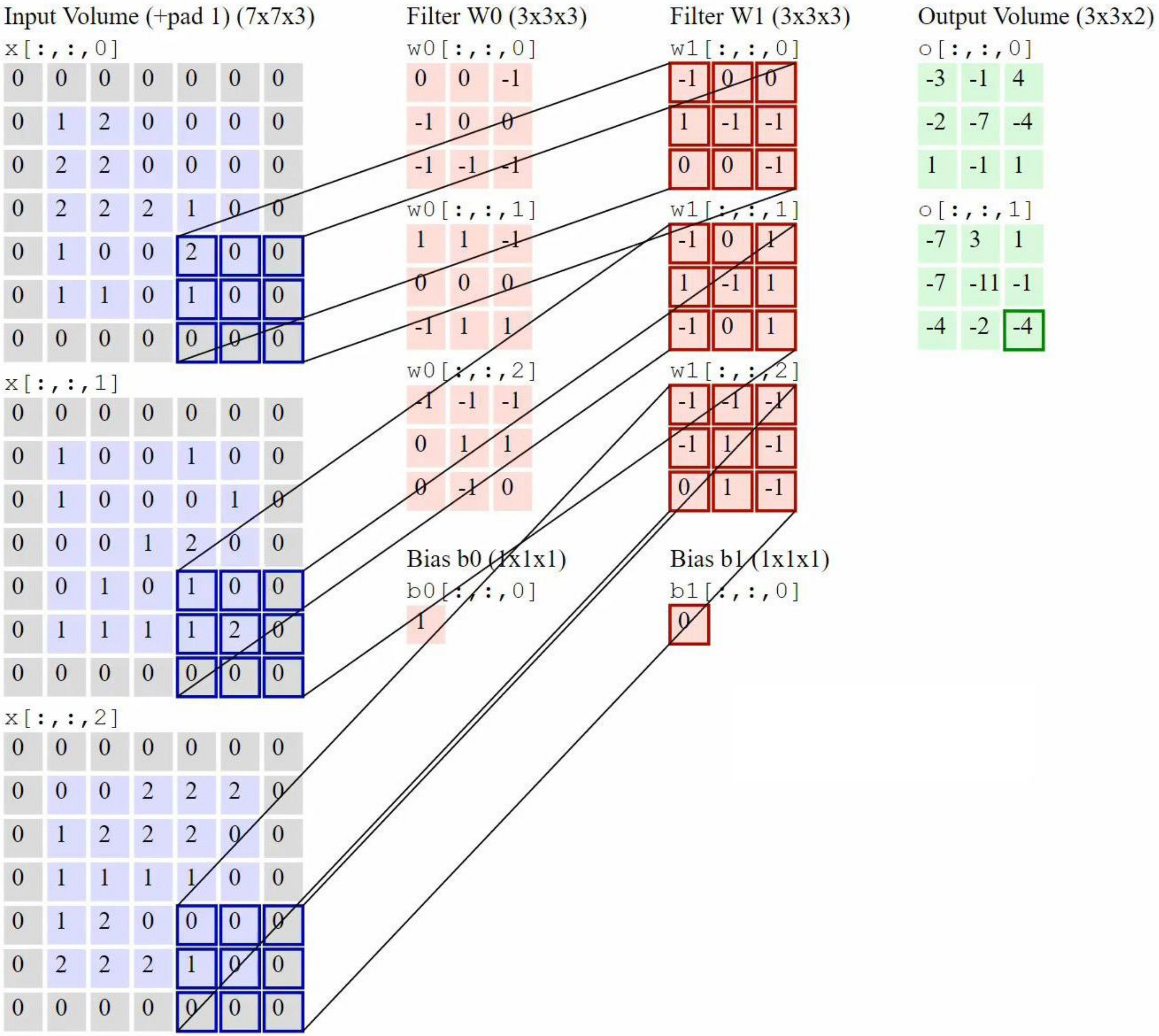
Input Volume (+pad 1) (7x7x3)
x[:,:,0]
0 0 0 0 0 0 0
0 1 2 0 0 0 0
0 2 2 0 0 0 0
0 2 2 2 1 0 0
0 1 0 0 2 0 0
0 1 1 0 1 0 0
0 0 0 0 0 0 0
x[:,:,1]
0 0 0 0 0 0 0
0 1 0 0 1 0 0
0 1 0 0 0 1 0
0 0 0 1 2 0 0
0 0 1 0 1 0 0
0 1 1 1 1 2 0
0 0 0 0 0 0 0
x[:,:,2]
0 0 0 0 0 0 0
0 0 0 2 2 2 0
0 1 2 2 2 0 0
0 1 1 1 1 0 0
0 1 2 0 0 0 0
0 2 2 2 1 0 0
0 0 0 0 0 0 0
Filter W0 (3x3x3)
w0[:,:,0]
0 0 -1
-1 0 0
-1 -1 -1
w0[:,:,1]
1 1 -1
0 0 0
-1 1 1
w0[:,:,2]
-1 -1 -1
0 1 1
0 -1 0
Bias b0 (1x1x1)
b0[:,:,0]
1
Filter W1 (3x3x3)
w1[:,:,0]
-1 0 0
1 -1 -1
0 0 -1
w1[:,:,1]
-1 0 1
1 -1 1
-1 0 1
w1[:,:,2]
-1 -1 -1
-1 1 -1
0 1 -1
Bias b1 (1x1x1)
b1[:,:,0]
0
Output Volume (3x3x2)
o[:,:,0]
-3 -1 4
-2 -7 -4
1 -1 1
o[:,:,1]
-7 3 1
-7 -11 -1
-4 -2 -4


Input Volume (+pad 1) (7x7x3)
x[:,:,0]
x[:,:,1]
x[:,:,2]
Filter W0 (3x3x3)
w0[:,:,0]
w0[:,:,1]
w0[:,:,2]
Filter W1 (3x3x3)
w1[:,:,0]
w1[:,:,1]
w1[:,:,2]
Bias b0 (1x1x1)
b0[:,:,0]
Bias b1 (1x1x1)
b1[:,:,0]
Output Volume (3x3x2)
o[:,:,0]
o[:,:,1]


Input Volume (+pad 1) (7x7x3)
x[:,:,0]
x[:,:,1]
x[:,:,2]
Filter W0 (3x3x3)
w0[:,:,0]
w0[:,:,1]
w0[:,:,2]
Bias b0 (1x1x1)
b0[:,:,0]
Filter W1 (3x3x3)
w1[:,:,0]
w1[:,:,1]
w1[:,:,2]
Bias b1 (1x1x1)
b1[:,:,0]
Output Volume (3x3x2)
o[:,:,0]
o[:,:,1]

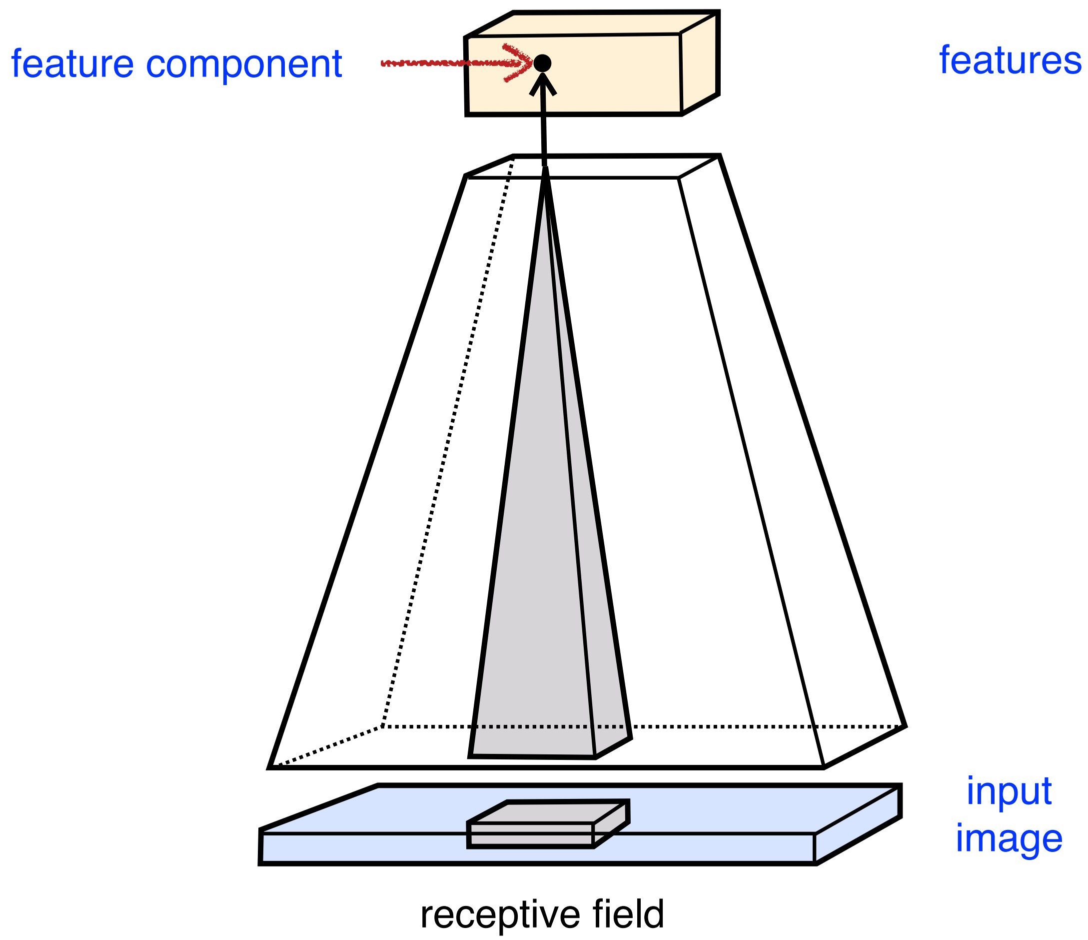

Input Volume (+pad 1) (7x7x3)
x[:,:,0]
0 0 0 0 0 0 0
0 1 2 0 0 0 0
0 2 2 0 0 0 0
0 2 2 2 1 0 0
0 1 0 0 2 0 0
0 1 1 0 1 0 0
0 0 0 0 0 0 0
x[:,:,1]
0 0 0 0 0 0 0
0 1 0 0 1 0 0
0 1 0 0 0 1 0
0 0 0 1 2 0 0
0 0 1 0 1 0 0
0 1 1 1 1 2 0
0 0 0 0 0 0 0
x[:,:,2]
0 0 0 0 0 0 0
0 0 0 2 2 2 0
0 1 2 2 2 0 0
0 1 1 1 1 0 0
0 1 2 0 0 0 0
0 2 2 2 1 0 0
0 0 0 0 0 0 0
Filter W0 (3x3x3)
w0[:,:,0]
0 0 -1
-1 0 0
-1 -1 -1
w0[:,:,1]
1 1 -1
0 0 0
-1 1 1
w0[:,:,2]
-1 -1 -1
0 1 1
0 -1 0
Bias b0 (1x1x1)
b0[:,:,0]
1
Filter W1 (3x3x3)
w1[:,:,0]
-1 0 0
1 -1 -1
0 0 -1
w1[:,:,1]
-1 0 1
1 -1 1
-1 0 1
w1[:,:,2]
-1 -1 -1
-1 1 -1
0 1 -1
Bias b1 (1x1x1)
b1[:,:,0]
0
Output Volume (3x3x2)
o[:,:,0]
-3 -1 4
-2 -7 -4
1 -1 1
o[:,:,1]
-7 3 1
-7 -11 -1
-4 -2 -4

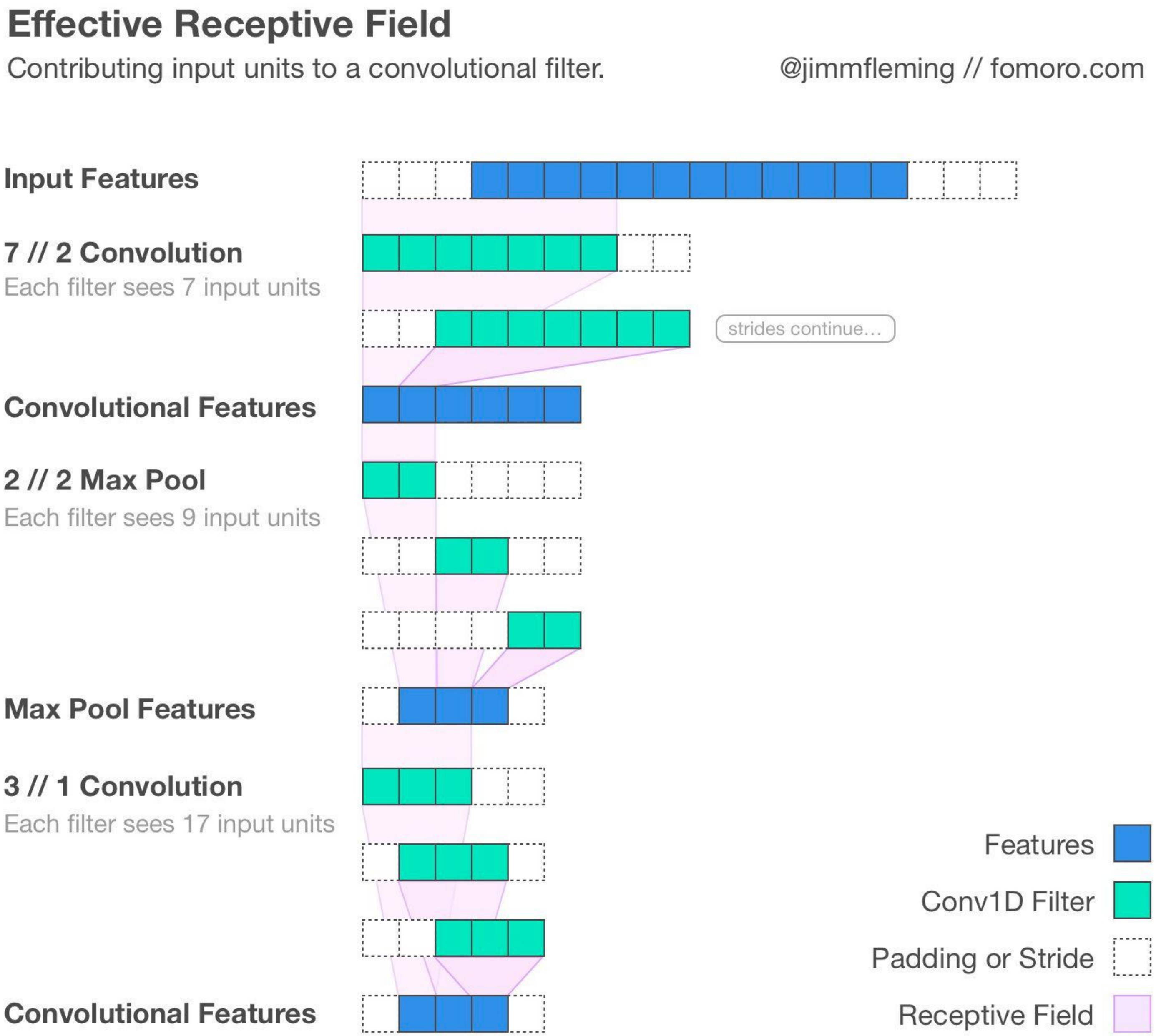

Input Volume (+pad 1) (7x7x3)
x[:,:,0]
0 0 0 0 0 0 0
0 1 2 0 0 0 0
0 2 2 0 0 0 0
0 2 2 2 1 0 0
0 1 0 0 2 0 0
0 1 1 0 1 0 0
0 0 0 0 0 0 0
x[:,:,1]
0 0 0 0 0 0 0
0 1 0 0 1 0 0
0 1 0 0 0 1 0
0 0 0 1 2 0 0
0 0 1 0 1 0 0
0 1 1 1 1 2 0
0 0 0 0 0 0 0
x[:,:,2]
0 0 0 0 0 0 0
0 0 0 2 2 2 0
0 1 2 2 2 0 0
0 1 1 1 1 0 0
0 1 2 0 0 0 0
0 2 2 2 1 0 0
0 0 0 0 0 0 0
Filter W0 (3x3x3)
w0[:,:,0]
0 0 -1
-1 0 0
-1 -1 -1
w0[:,:,1]
1 1 -1
0 0 0
-1 1 1
w0[:,:,2]
-1 -1 -1
0 1 1
0 -1 0
Bias b0 (1x1x1)
b0[:,:,0]
1
Filter W1 (3x3x3)
w1[:,:,0]
-1 0 0
1 -1 -1
0 0 -1
w1[:,:,1]
-1 0 1
1 -1 1
-1 0 1
w1[:,:,2]
-1 -1 -1
-1 1 -1
0 1 -1
Bias b1 (1x1x1)
b1[:,:,0]
0
Output Volume (3x3x2)
o[:,:,0]
-3 -1 4
-2 -7 -4
1 -1 1
o[:,:,1]
-7 3 1
-7 -11 -1
-4 -2 -4


Input Volume (+pad 1) (7x7x3)
x[:,:,0]
0 0 0 0 0 0 0
0 1 2 0 0 0 0
0 2 2 0 0 0 0
0 2 2 2 1 0 0
0 1 0 0 2 0 0
0 1 1 0 1 0 0
0 0 0 0 0 0 0
x[:,:,1]
0 0 0 0 0 0 0
0 1 0 0 1 0 0
0 1 0 0 0 1 0
0 0 0 1 2 0 0
0 0 1 0 1 0 0
0 1 1 1 1 2 0
0 0 0 0 0 0 0
x[:,:,2]
0 0 0 0 0 0 0
0 0 0 2 2 2 0
0 1 2 2 2 0 0
0 1 1 1 1 0 0
0 1 2 0 0 0 0
0 2 2 2 1 0 0
0 0 0 0 0 0 0
Filter W0 (3x3x3)
w0[:,:,0]
0 0 -1
-1 0 0
-1 -1 -1
w0[:,:,1]
1 1 -1
0 0 0
-1 1 1
w0[:,:,2]
-1 -1 -1
0 1 1
0 -1 0
Bias b0 (1x1x1)
b0[:,:,0]
1
Filter W1 (3x3x3)
w1[:,:,0]
-1 0 0
1 -1 -1
0 0 -1
w1[:,:,1]
-1 0 1
1 -1 1
-1 0 1
w1[:,:,2]
-1 -1 -1
-1 1 -1
0 1 -1
Bias b1 (1x1x1)
b1[:,:,0]
0
Output Volume (3x3x2)
o[:,:,0]
-3 -1 4
-2 -7 -4
1 -1 1
o[:,:,1]
-7 3 1
-7 -11 -1
-4 -2 -4

Convolutional layers

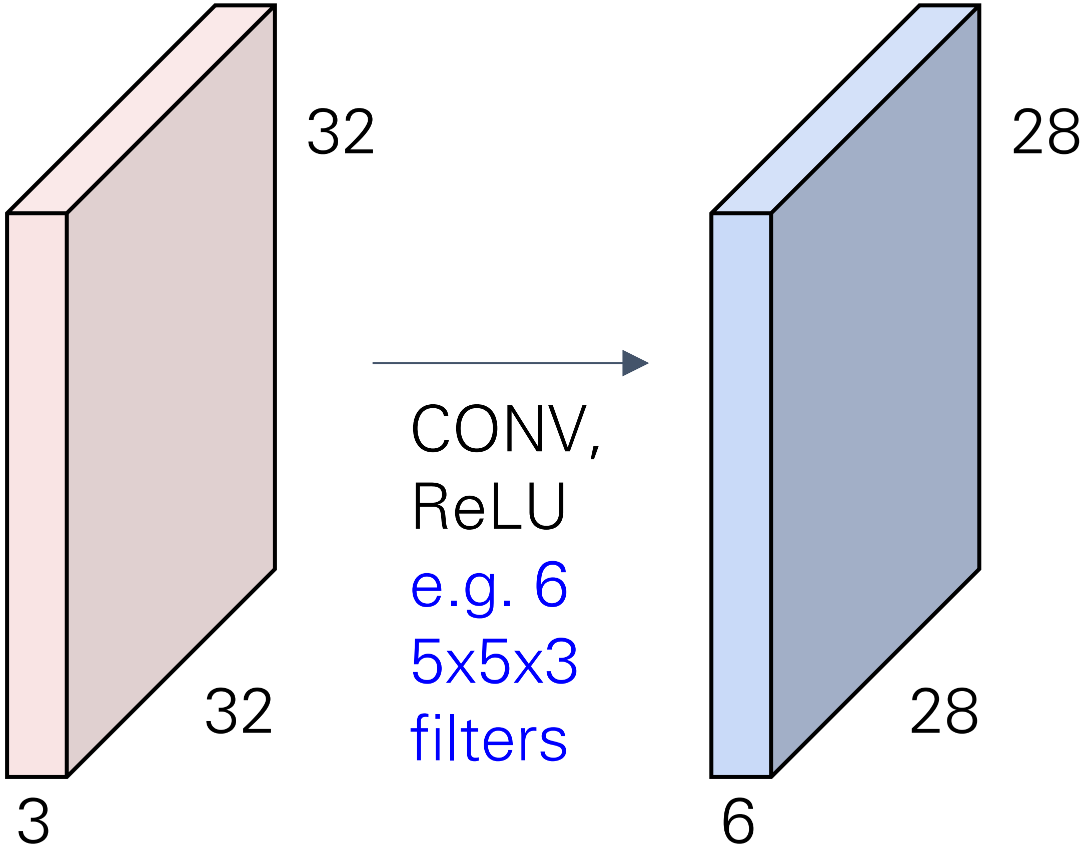
- Local receptive field
- Each column of hidden units looks at a different input patch



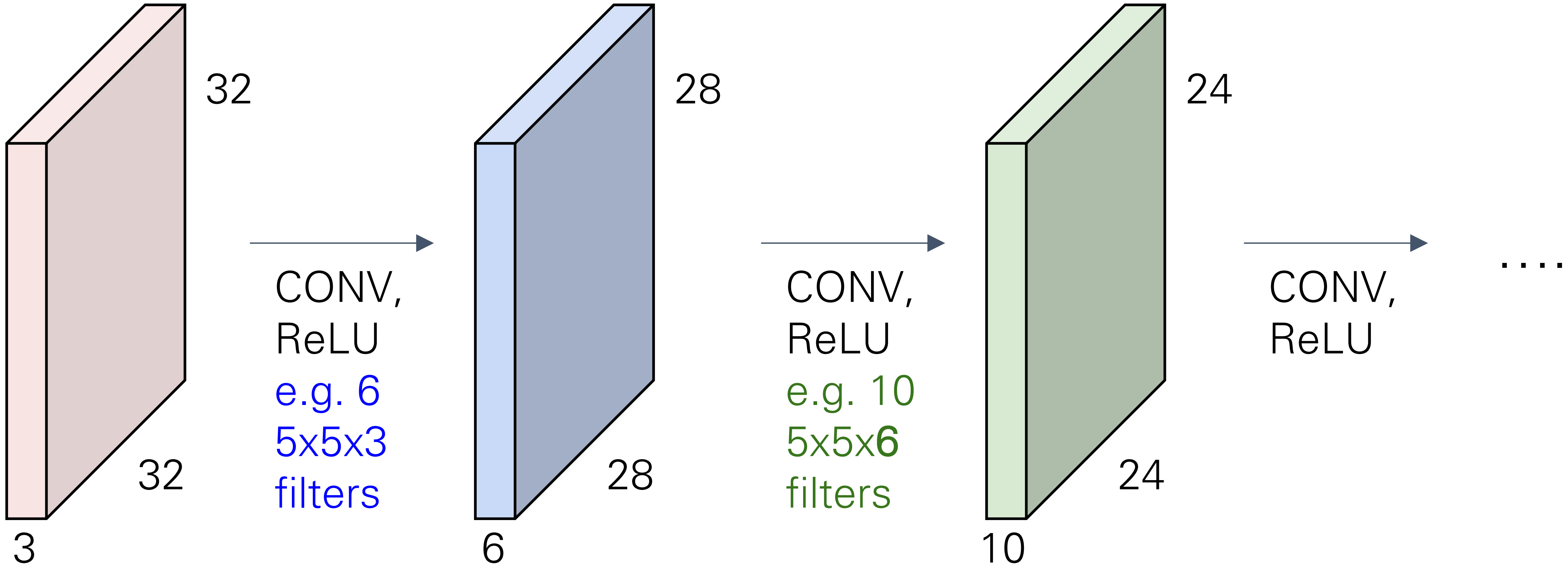


[http://fomoro.com/tools/receptive-fields/index.html]

Convolutional layers

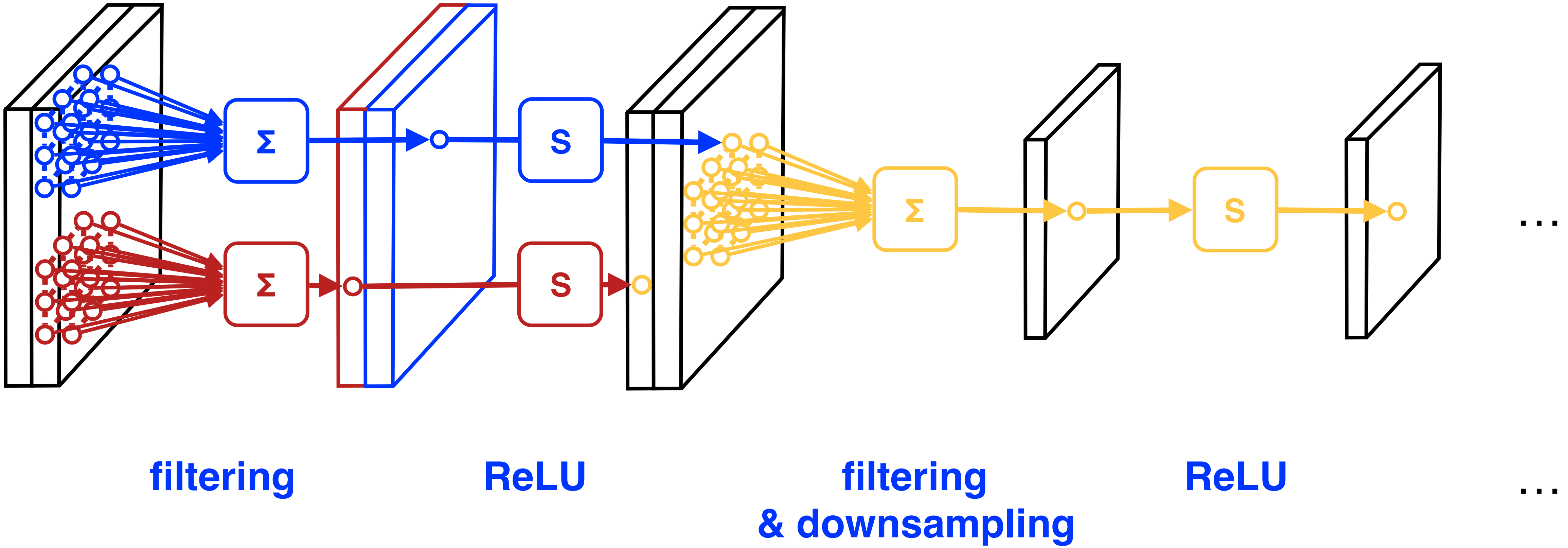


Repeat linear / non-linear operators



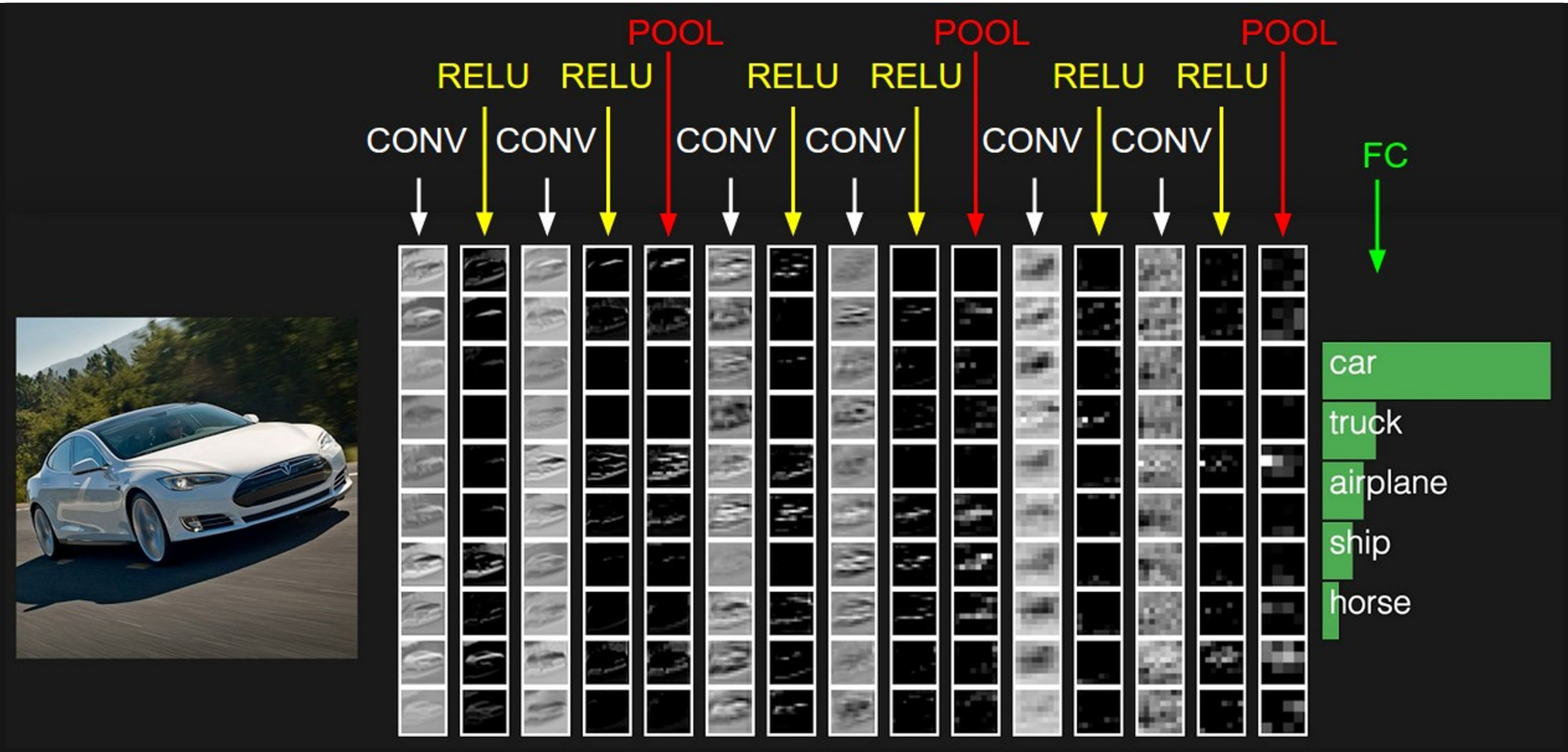
Linear/Non-linear Chains

- The basic blueprint of most architectures
- Stack multiple layers of convolutions



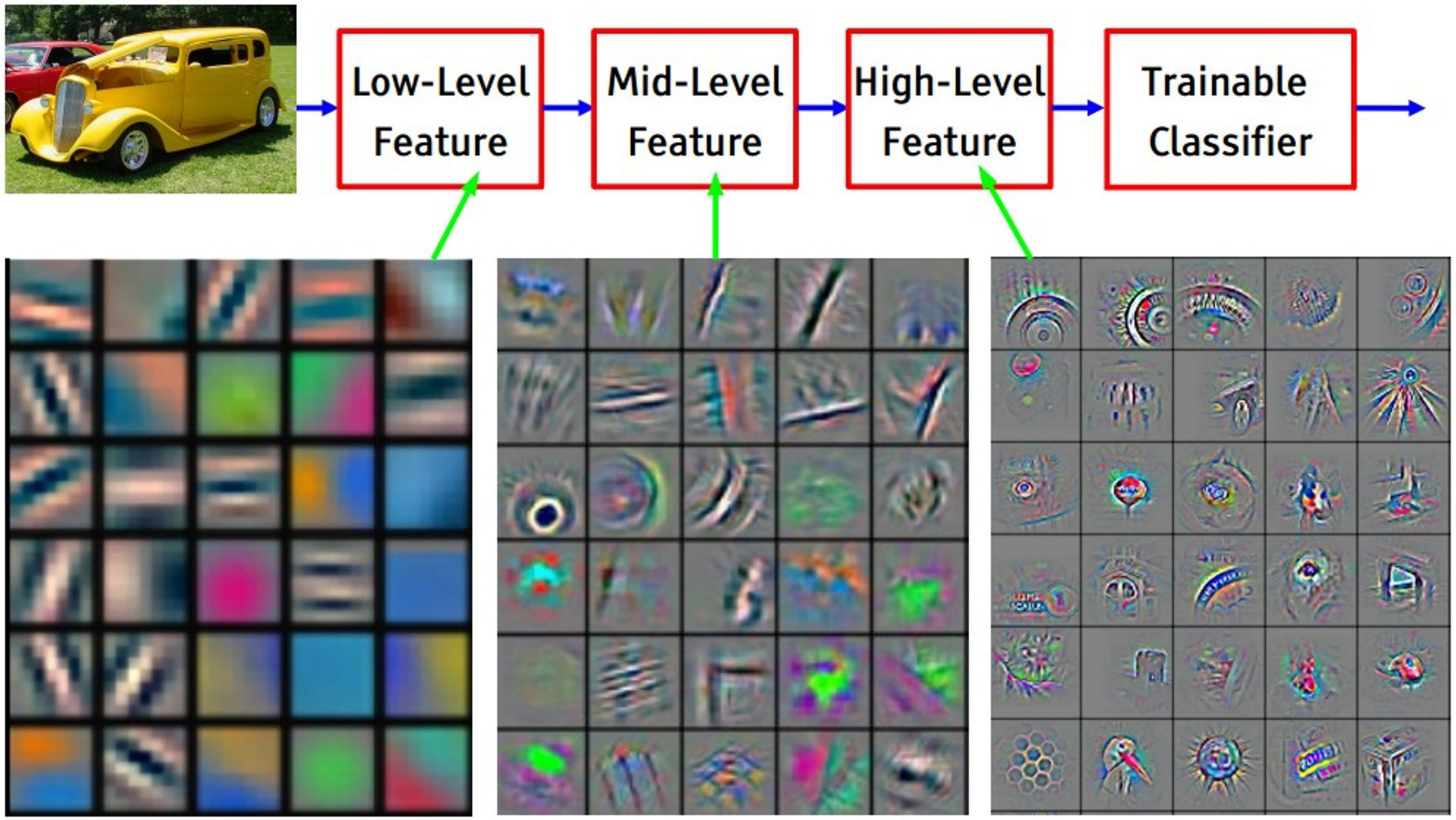
Feature Learning

- Hierarchical layer structure allows to learn hierarchical filters (features).



Feature Learning

- Hierarchical layer structure allows to learn hierarchical filters (features).

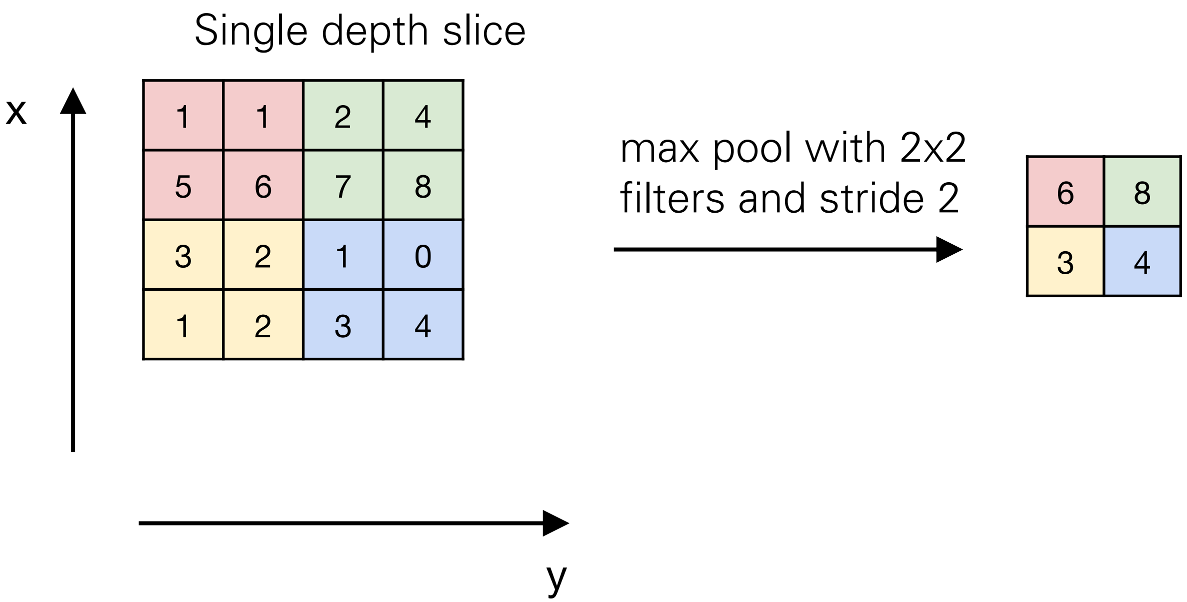


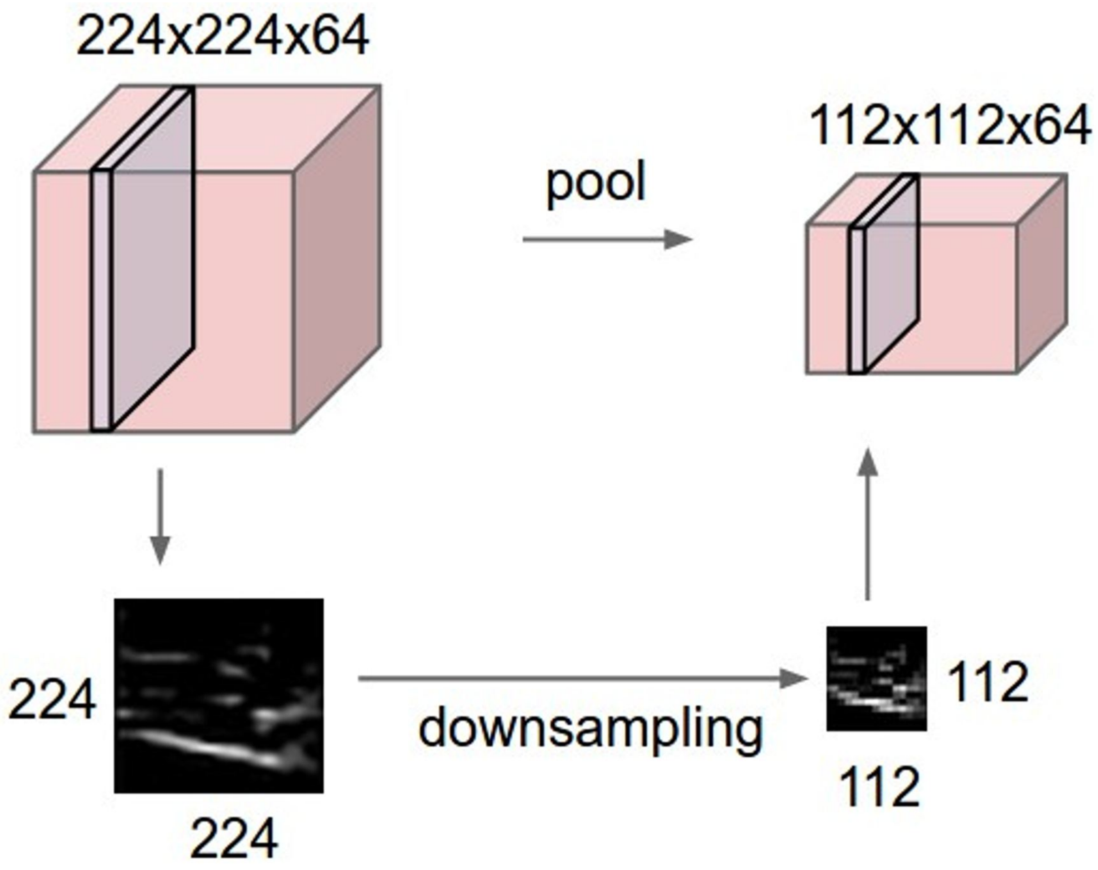
Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Slide credit: Yann LeCun

Pooling layer

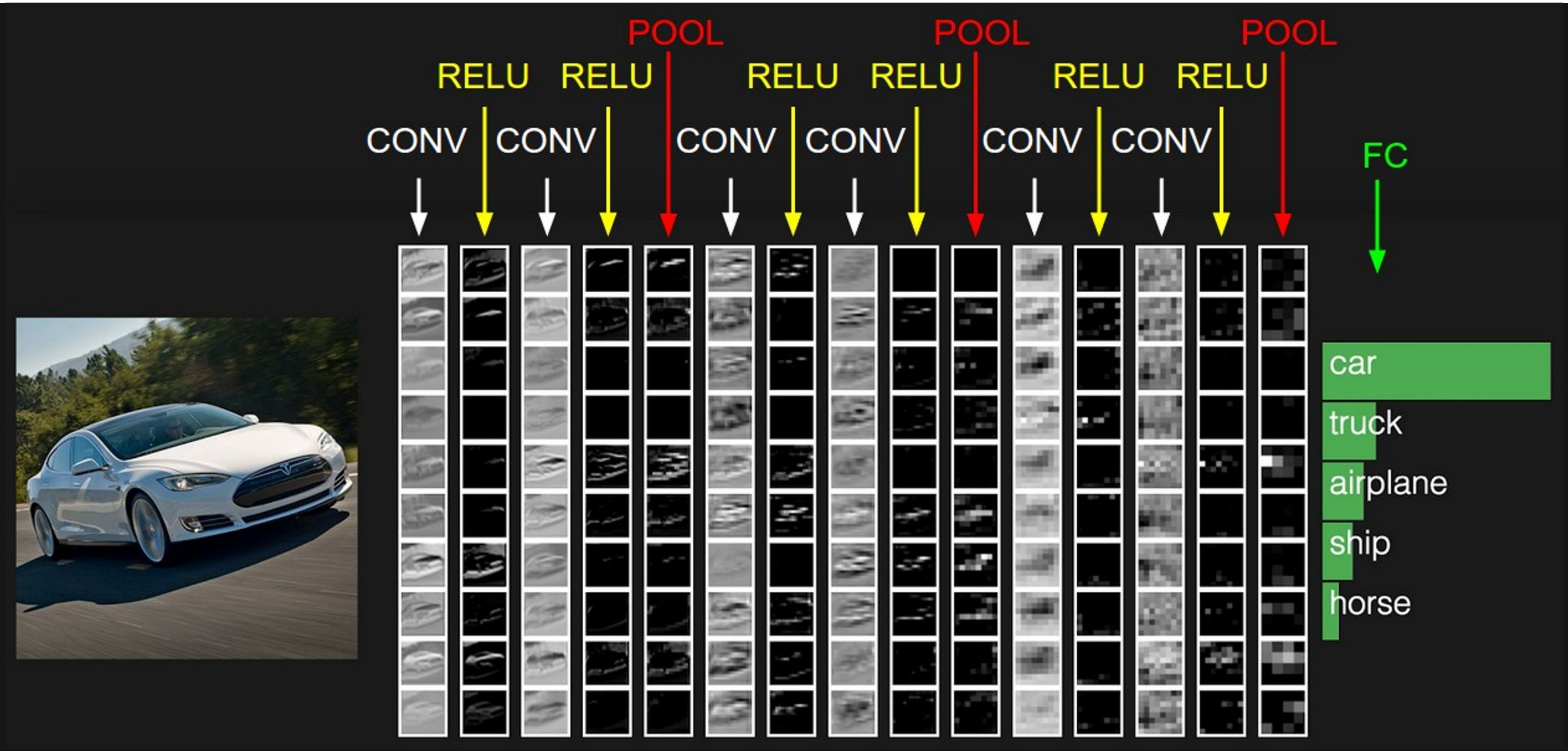
- makes the representations smaller and more manageable
- operates over each activation map independently:
- Max pooling, average pooling, etc.





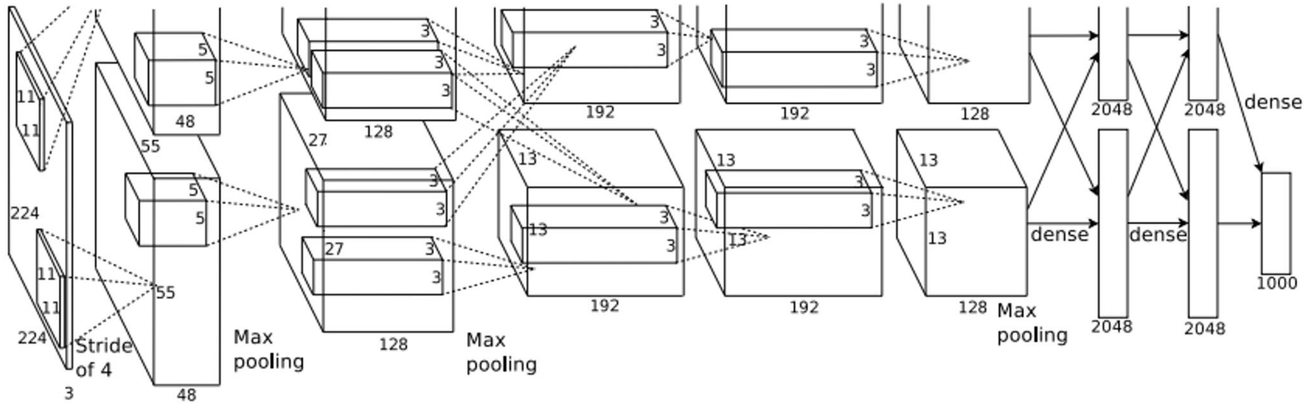
Fully connected layer

- contains neurons that connect to the entire input volume, as in ordinary Neural Networks



Case Study: AlexNet

[Krizhevsky et al. 2012]



Full (simplified) AlexNet architecture:
[227x227x3] INPUT
[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0
[27x27x96] MAX POOL1: 3x3 filters at stride 2
[27x27x96] NORM1: Normalization layer
[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2
[13x13x256] MAX POOL2: 3x3 filters at stride 2
[13x13x256] NORM2: Normalization layer
[13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1
[13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1
[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1
[6x6x256] MAX POOL3: 3x3 filters at stride 2
[4096] FC6: 4096 neurons
[4096] FC7: 4096 neurons
[1000] FC8: 1000 neurons (class scores)

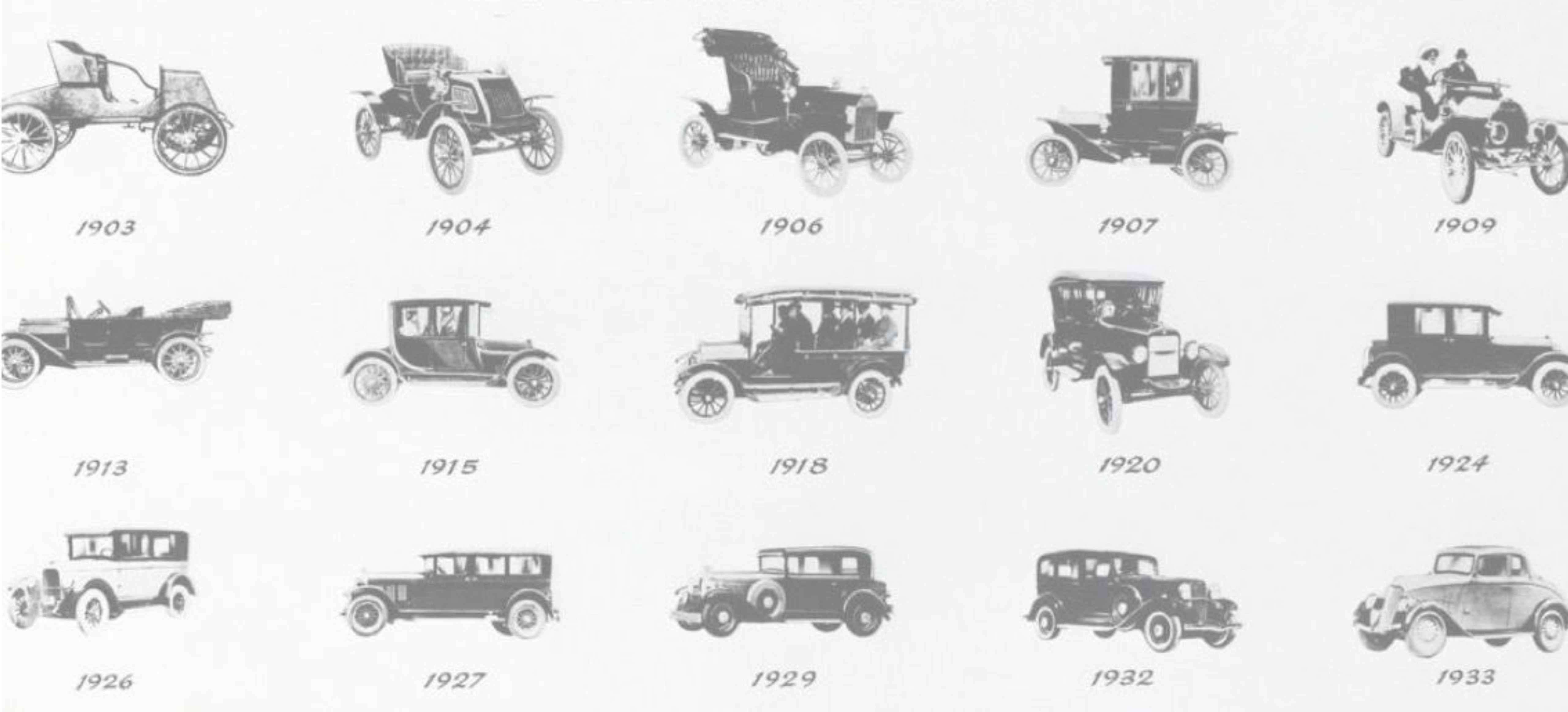
Details/Retrospectives:
- first use of ReLU
- used Norm layers (not common anymore)
- heavy data augmentation
- dropout 0.5
- batch size 128
- SGD Momentum 0.9
- Learning rate 1e-2, reduced by 10 manually when val accuracy plateaus
- L2 weight decay 5e-4
- 7 CNN ensemble: 18.2% -> 15.4%

Convolutional Neural Network Demo

- ConvNetJS demo: training on CIFAR-10
- http://cs.stanford.edu/people/karpathy/convnetjs/demo/cifar10.html

Three Years of Progress

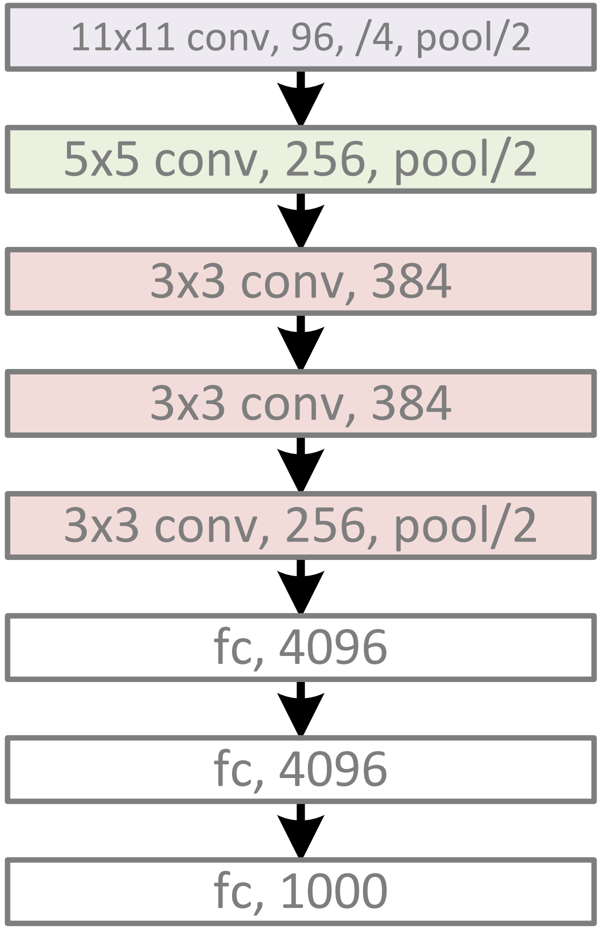
From AlexNet (2012) to ResNet (2015)



Revolution of Depth

AlexNet, 8 layers

(ILSVRC 2012)

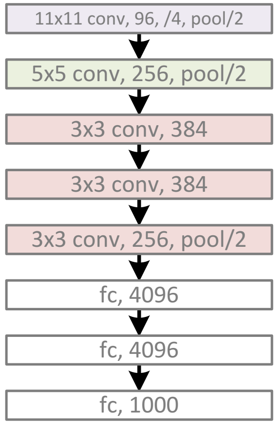


- 5 convolutional layers
- 3 fully connected layers
- ReLU
- End-to-end (no pre-training)
- Data augmentation

Revolution of Depth

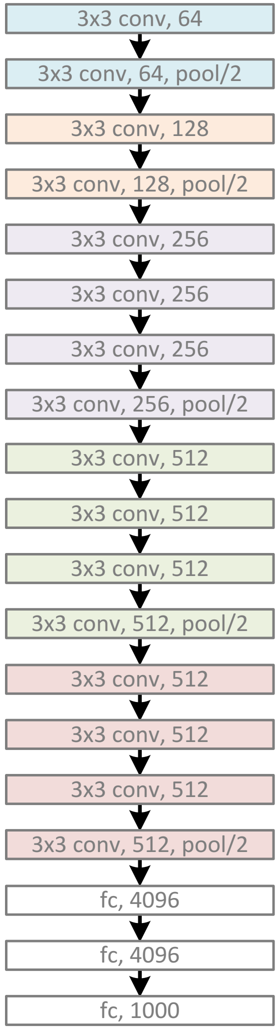
AlexNet, 8 layers

(ILSVRC 2012)



VGG, 19 layers

(ILSVRC 2014)



- Very deep
- Simply deep

INPUT: [224x224x3] memory: 224*224*3=150K params: 0 (not counting biases)
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

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

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 **conv1-256**	conv3-256 conv3-256 **conv3-256**
maxpool		
conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**
maxpool		
conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**
maxpool		
FC-4096		
FC-4096		
FC-1000		
soft-max		

VGG-16 Net

INPUT: [224x224x3] memory: 224*224*3=150K params: 0 (not counting biases)
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

Note:

Most memory is in early CONV

Most params are in late FC

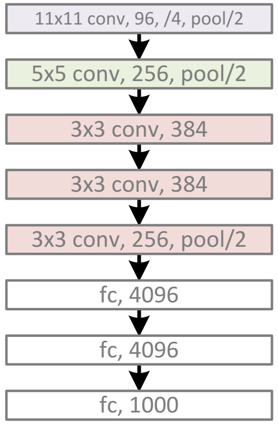
TOTAL memory: 24M * 4 bytes ~= 93MB / image
(only forward! ~*2 for bwd)
TOTAL params: 138M parameters

VGG-16 Net

Revolution of Depth

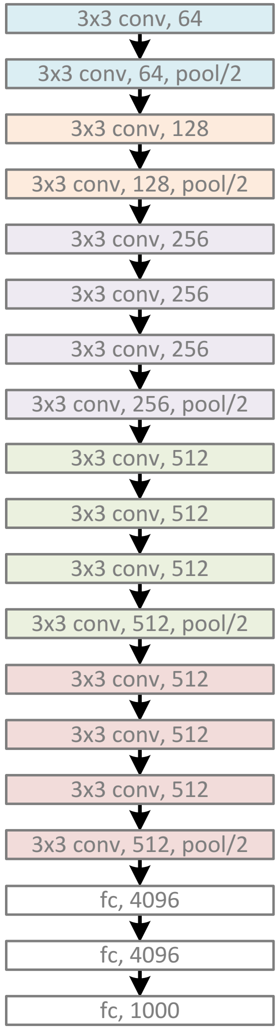
AlexNet, 8 layers

(ILSVRC 2012)



VGG, 19 layers

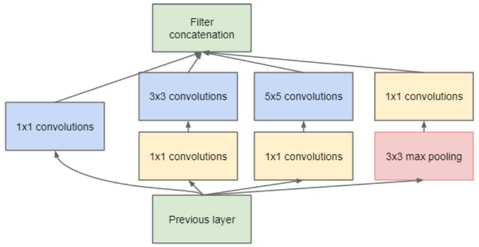
(ILSVRC 2014)

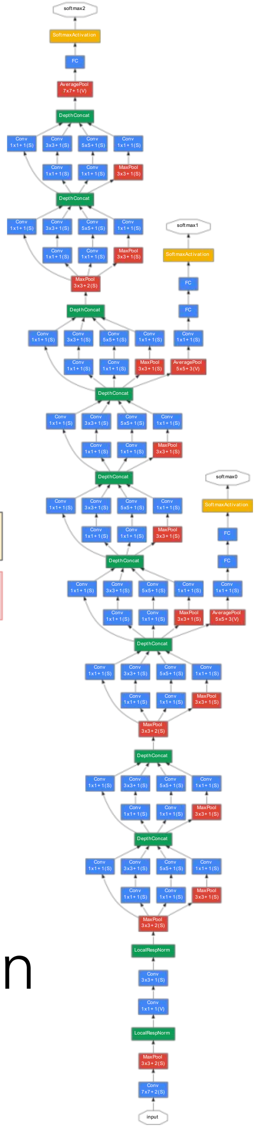


- Very deep
- Simply deep

GoogLeNet,
22 layers

(ILSVRC 2014)

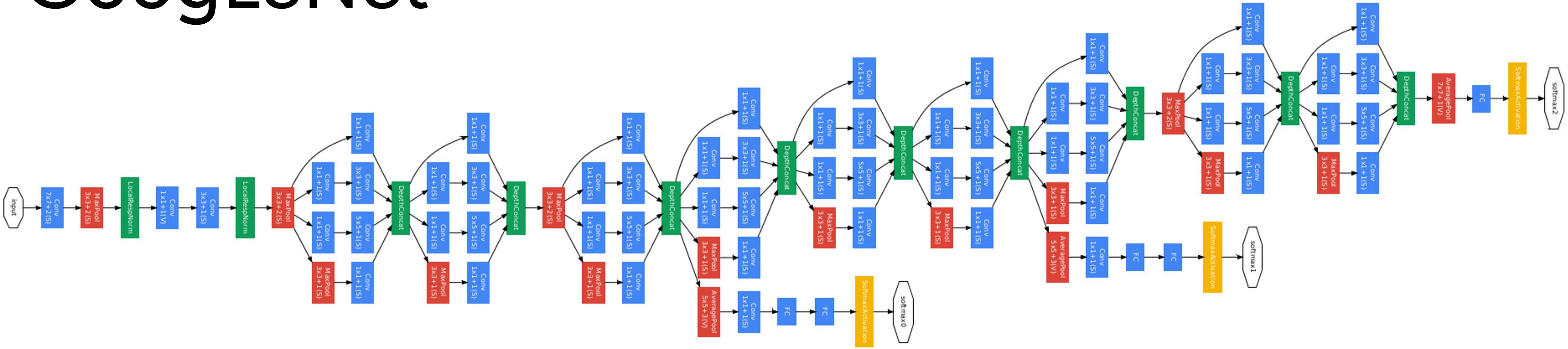


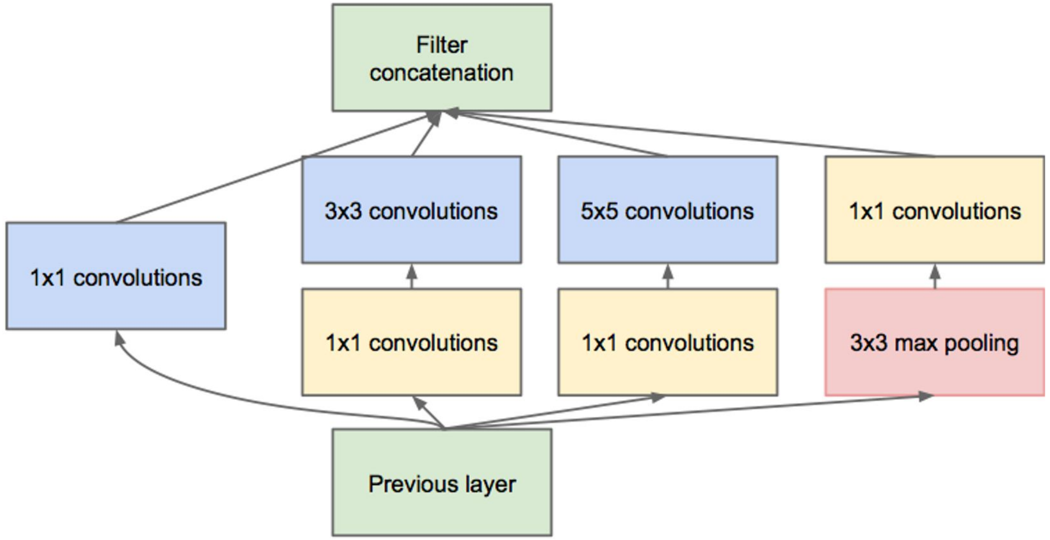


- Branching
- Bootleneck
- Skip connection

GoogLeNet

[Szegedy et al., 2014]





Inception module

GoogLeNet

[Szegedy et al., 2014]

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

Fun features:

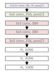
- Only 5 million params!
(Removes FC layers completely)

Compared to AlexNet:
- 12X less params
- 2x more compute
- 6.67% (vs. 16.4%)

Revolution of Depth

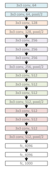
AlexNet, 8 layers

(ILSVRC 2012)



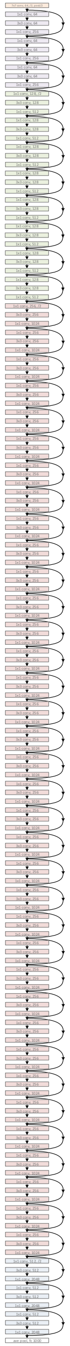
VGG, 19 layers

(ILSVRC 2014)



ResNet,
152 layers

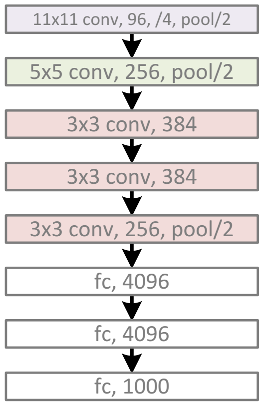
(ILSVRC 2015)



Revolution of Depth

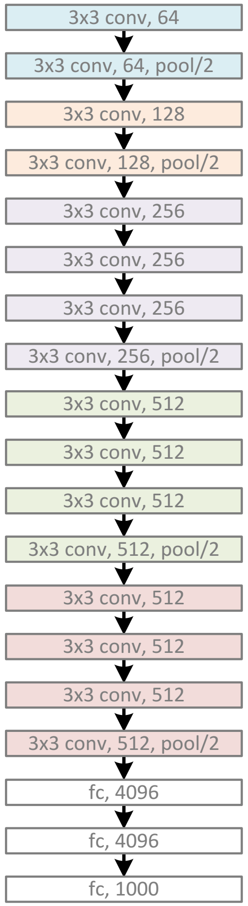
AlexNet, 8 layers

(ILSVRC 2012)



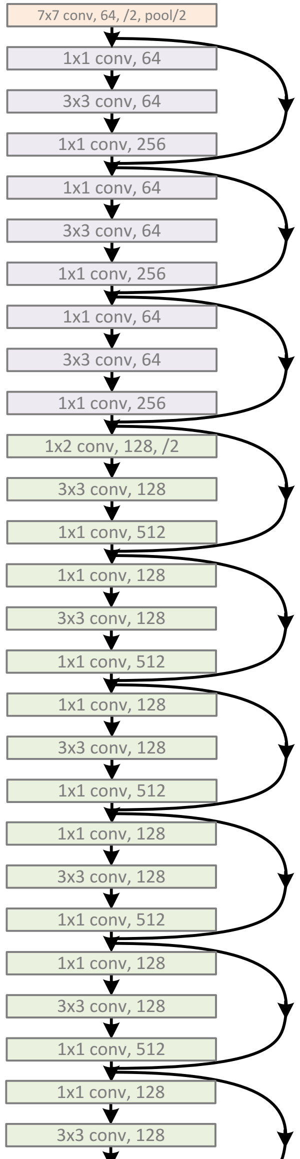
VGG, 19 layers

(ILSVRC 2014)



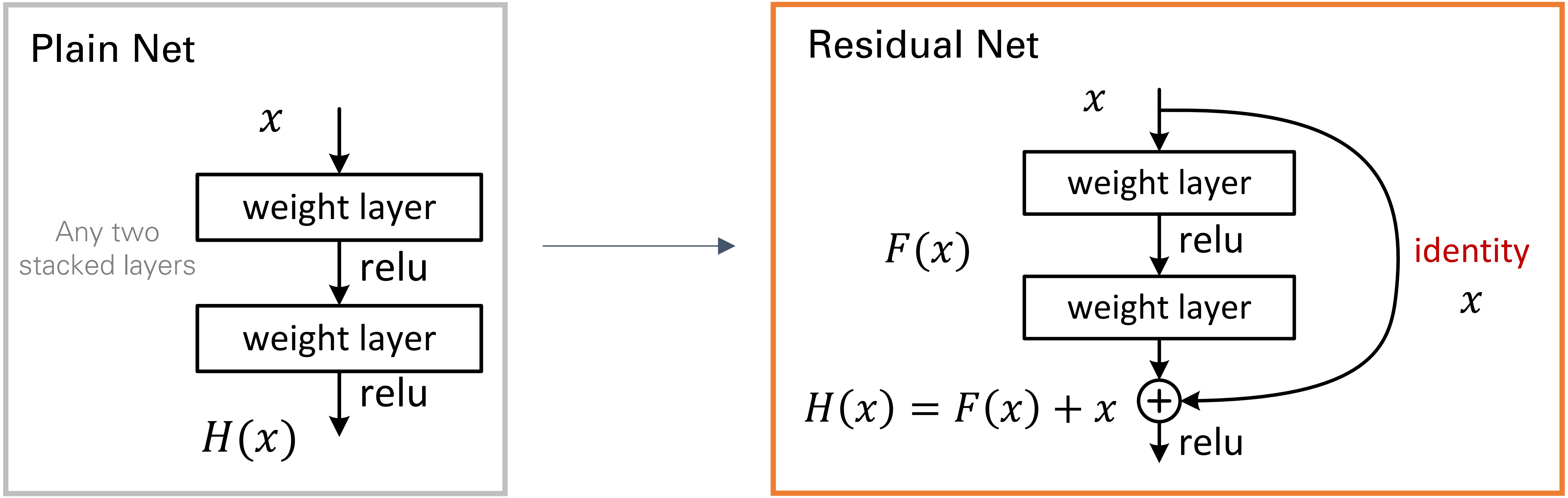
ResNet,
152 layers

(ILSVRC 2015)

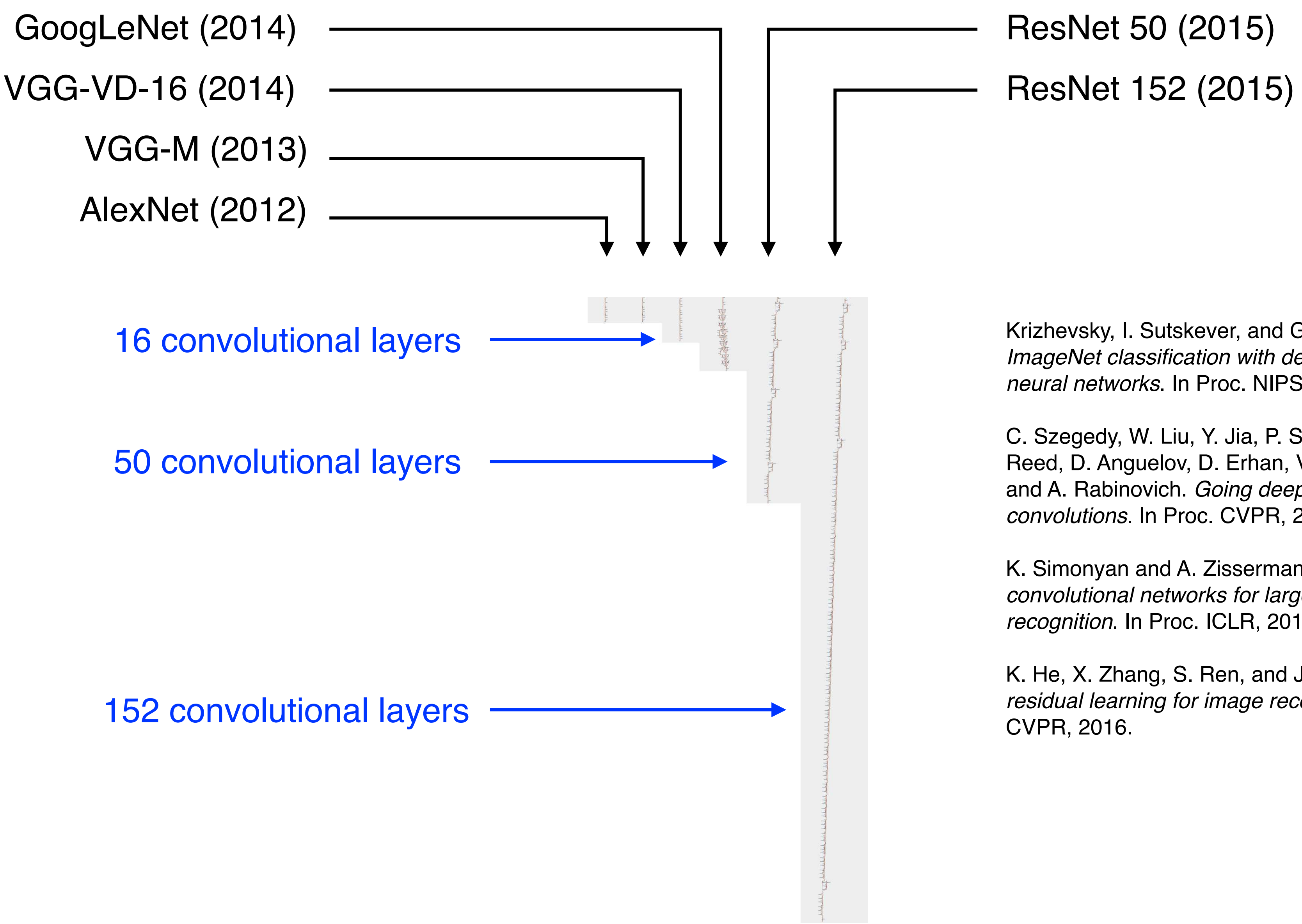


Residual Net (ResNet)

[He et al., 2015]



How deep is enough?



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

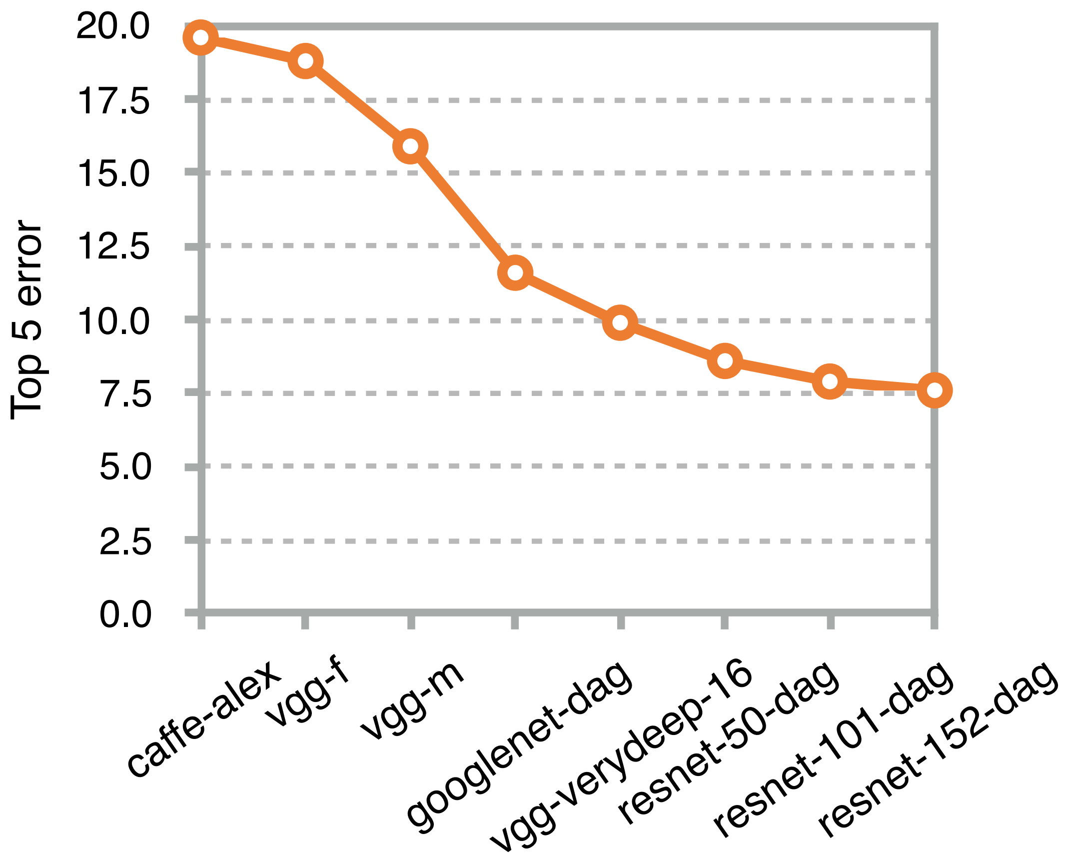
C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. *Going deeper with convolutions*. In Proc. CVPR, 2015.

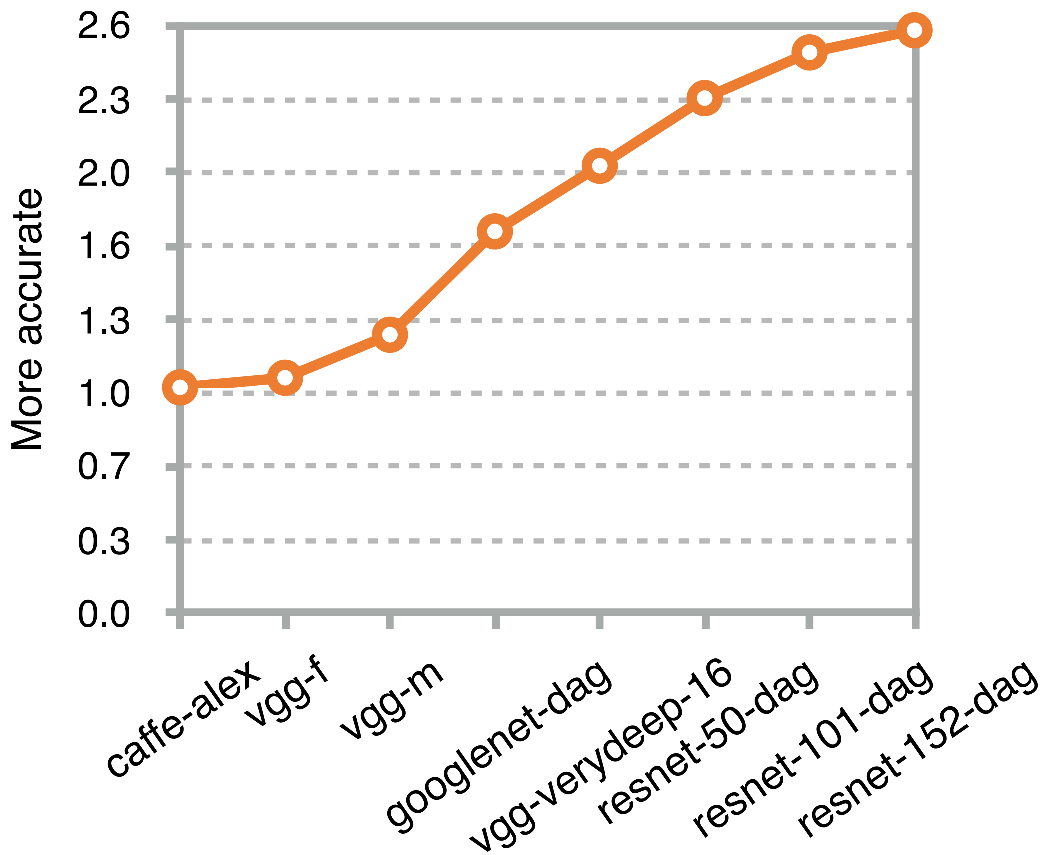
K. Simonyan and A. Zisserman. *Very deep convolutional networks for large-scale image recognition*. In Proc. ICLR, 2015.

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

How deep is enough?

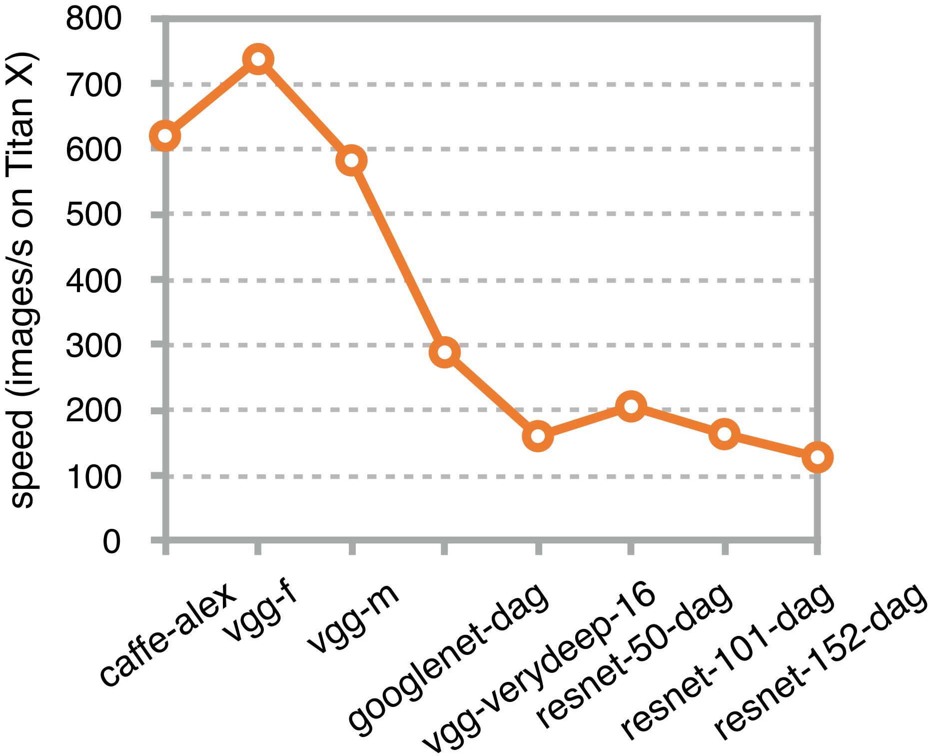
- 3 × more accurate in 3 years

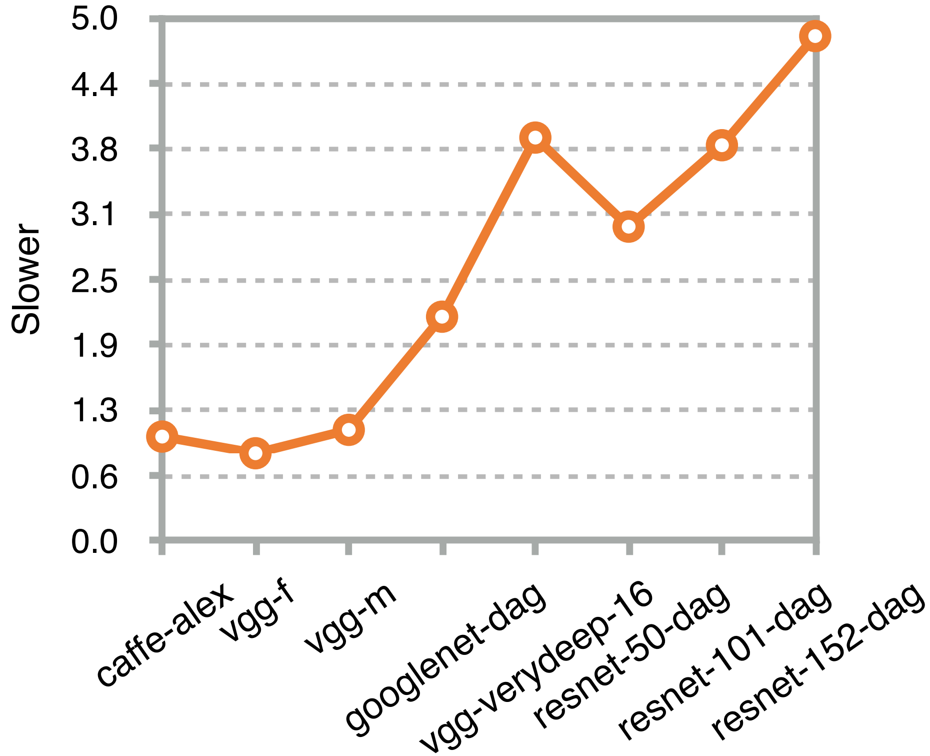




Speed

- 5 × slower

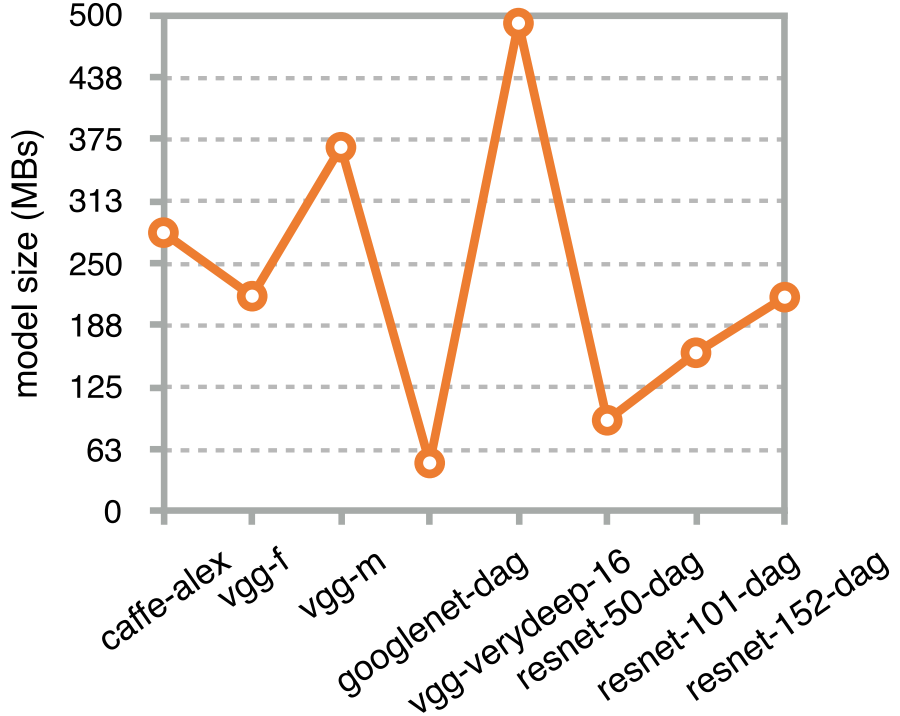


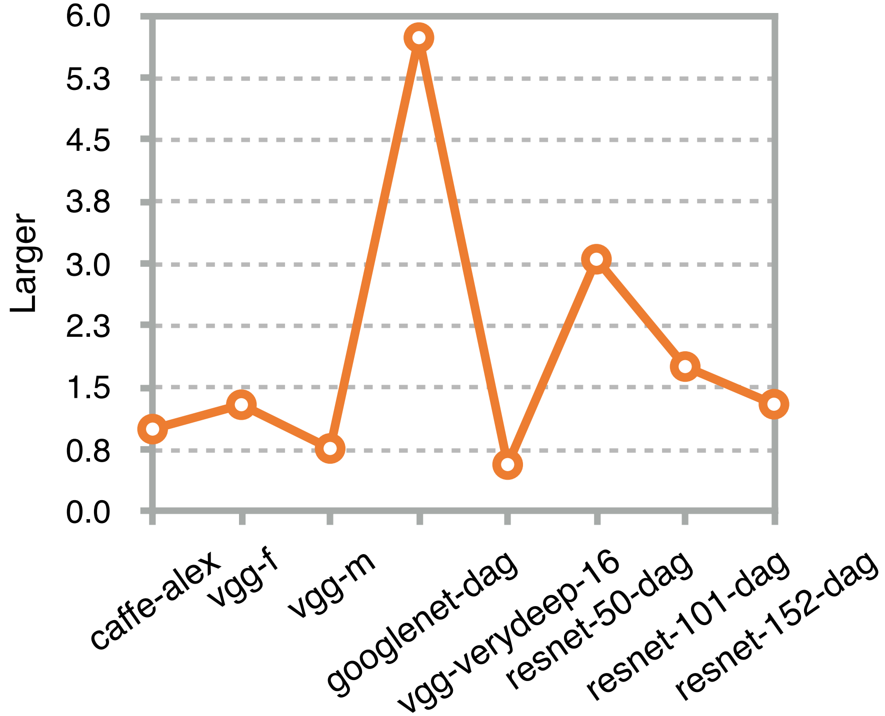


- **Remark:** 101 ResNet layers same size/speed as 16 VGG-VD layers
- **Reason:** Far fewer feature channels (quadratic speed/space gain)
- **Moral:** Optimize your architecture

Model Size

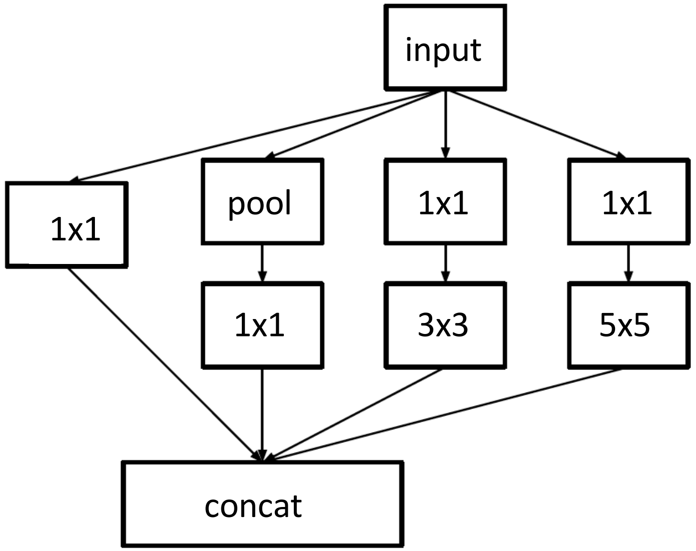
- Num. of parameters is about the same



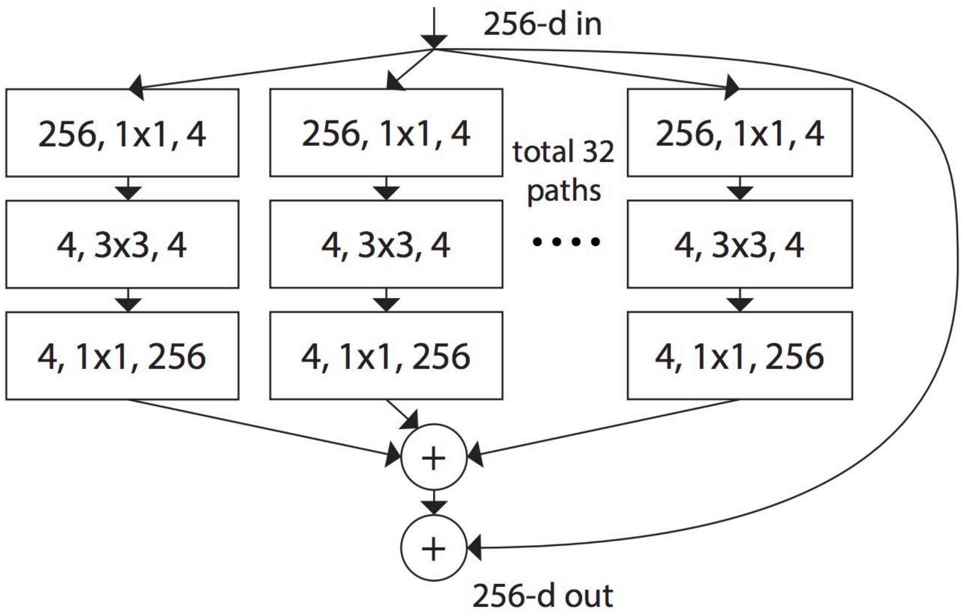


- **Remark:** 101 ResNet layers same size/speed as 16 VGG-VD layers
- **Reason:** Far fewer feature channels (quadratic speed/space gain)
- **Moral:** Optimize your architecture

ResNeXt: Both Wider and Deeper



Inception:
heterogeneous multi-branch



ResNeXt:
uniform multi-branch

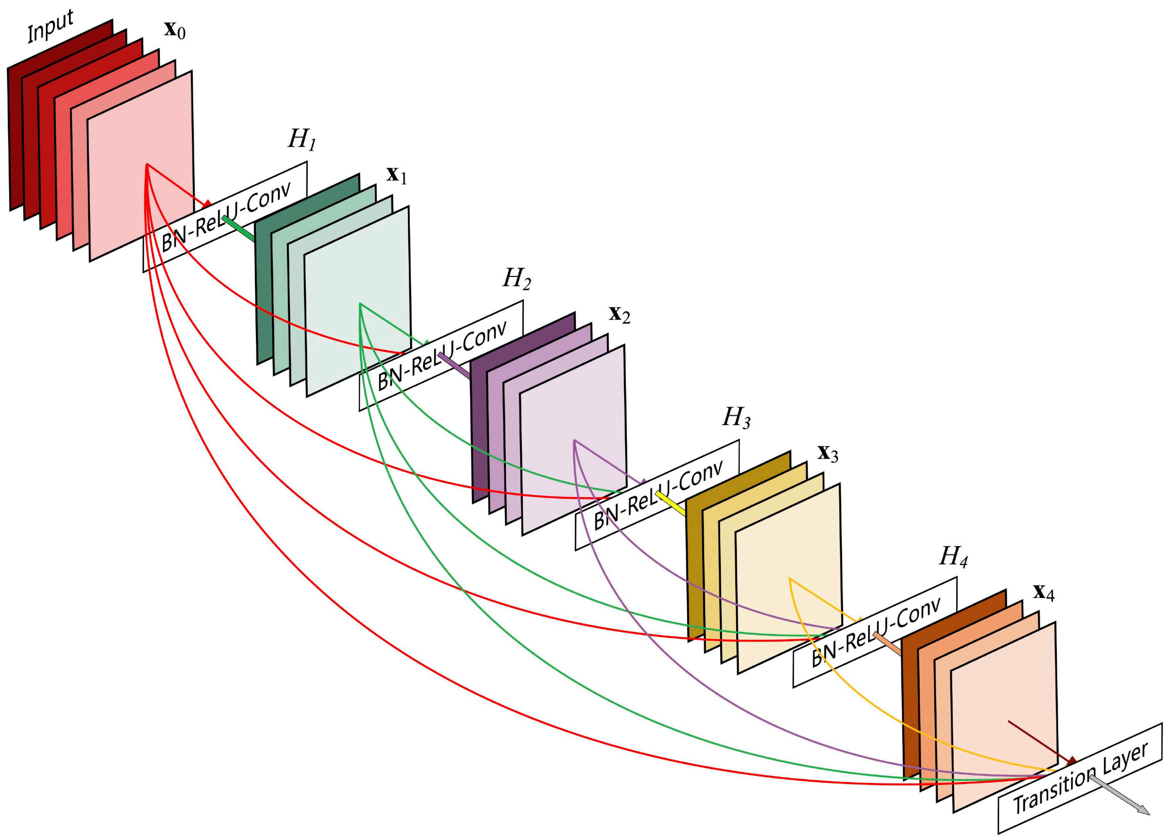
- shortcut, bottleneck, and multi-branch
- Better accuracy (when having the same FLOPs/#params as ResNet)

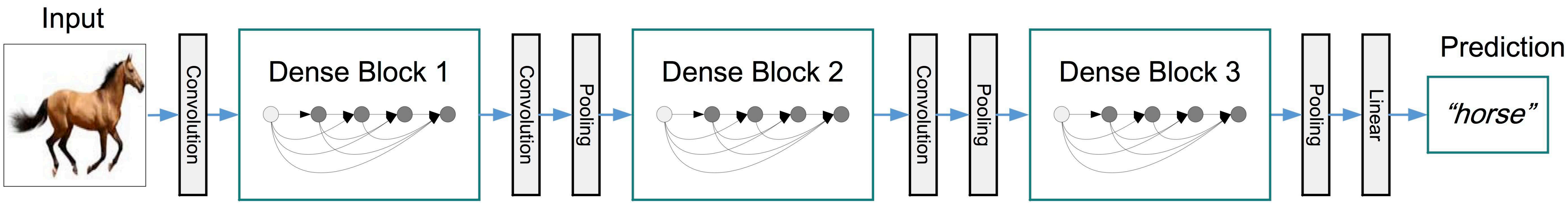
Saining Xie et al. "Aggregated Residual Transformations for Deep Neural Networks". CVPR 2017

DenseNet

- 201 layers, 20M parameters

- Densely connected blocks
- Alleviates vanishing gradient
- Strengthens feature propagation
- Encourages feature reuse

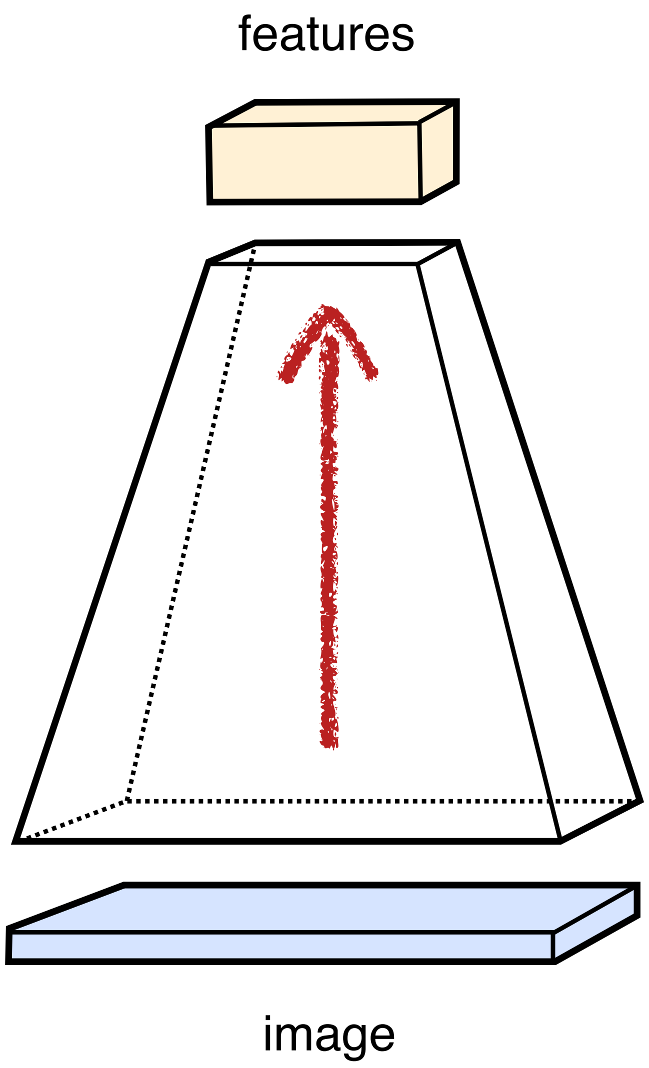




G. Huang, Z. Liu, L. van der Maaten, & K. Weinberger, **"Densely connected convolutional networks"**, CVPR 2017

Design Guidelines

Design Guidelines



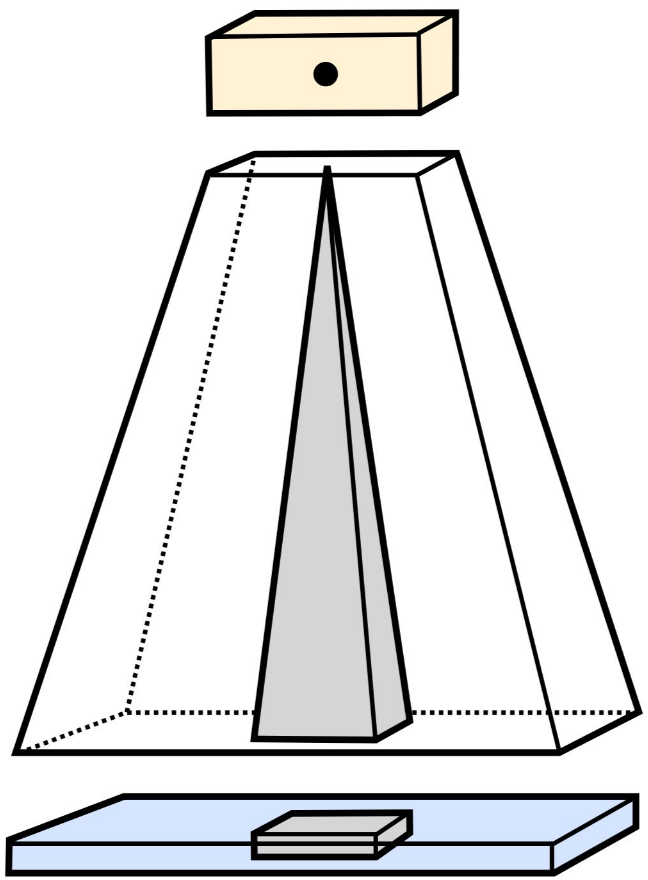
Guideline 1: Avoid tight bottlenecks

- **From bottom to top**
  - The spatial resolution H×W decreases
  - The number of channels C increases
- **Guideline**
  - Avoid tight information bottleneck
  - Decrease the data volume H × W × C slowly

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

C. Szegedy, V. Vanhoucke, S. Ioffe, and J. Shlens. **Rethinking the inception architecture for computer vision**. In CVPR 2016.

Receptive Field

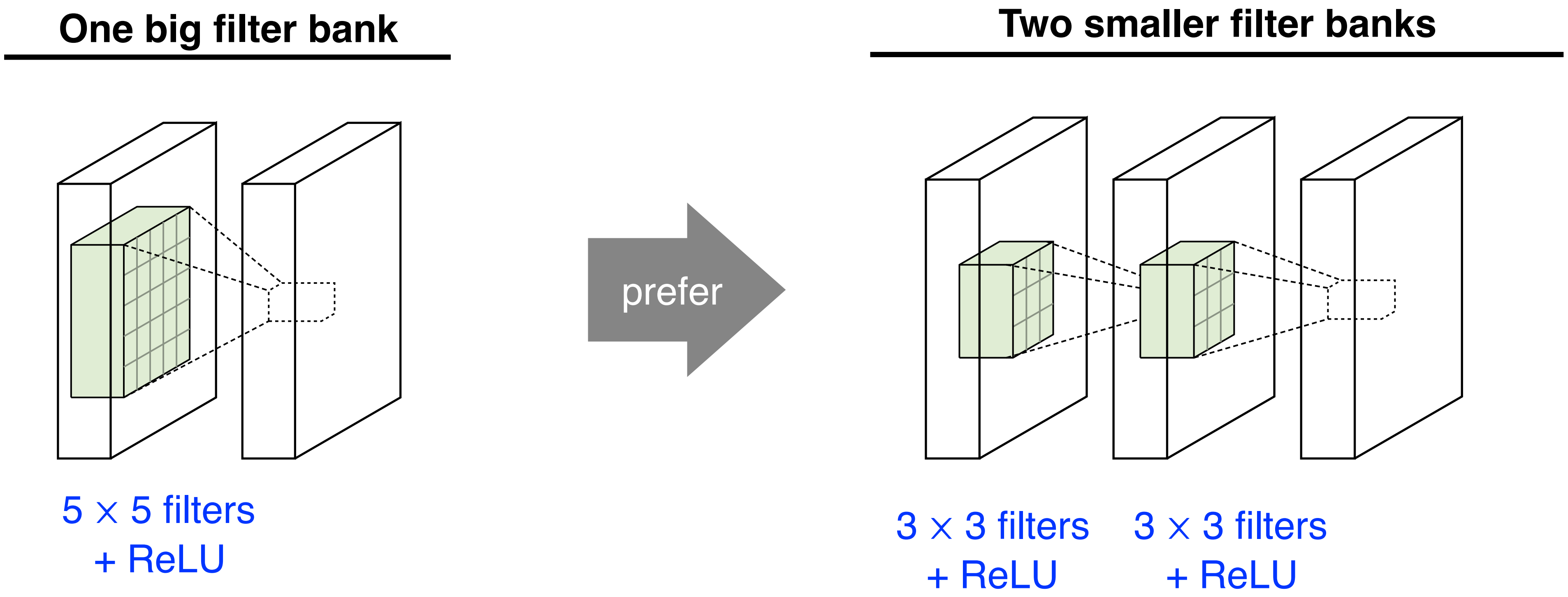


Must be large enough

- **Receptive field of a neuron**
  - The image region influencing a neuron
  - Anything happening outside is invisible to the neuron
- **Importance**
  - Large image structures cannot be detected by neurons with small receptive fields
- **Enlarging the receptive field**
  - Large filters
  - Chains of small filters

Design Guidelines

Guideline 2: Prefer small filter chains



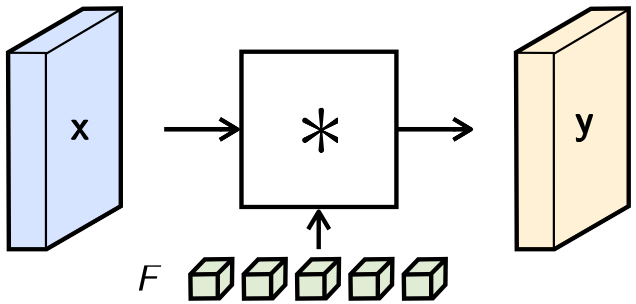
- **Remark:** 101 ResNet layers same size/speed as 16 VGG-VD layers
- **Reason:** Far fewer feature channels (quadratic speed/space gain)
- **Moral:** Optimize your architecture

Design Guidelines

Guideline 3:

Keep
the number
of channels
at bay

$H \times W \times C$



$H_f \times W_f \times \underline{C \times K}$

C = num. input channels

K = num. output channels

Num. of operations

$$\frac{H \times H_f}{\text{stride}} \times \frac{W \times W_f}{\text{stride}} \times \underline{C \times K}$$

Num. of parameters

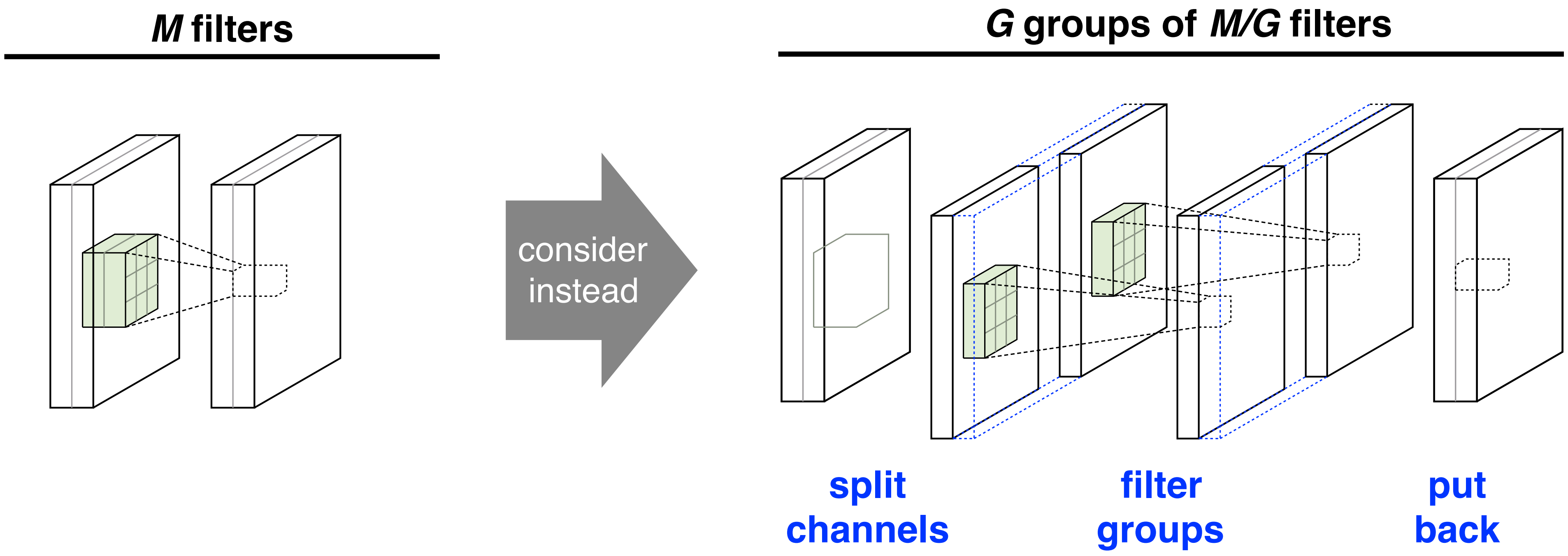
$$H_f \times W_f \times \underline{C \times K}$$

$$\boxed{\text{complexity} \propto \underline{C \times K}}$$

Design Guidelines

Guideline 4:

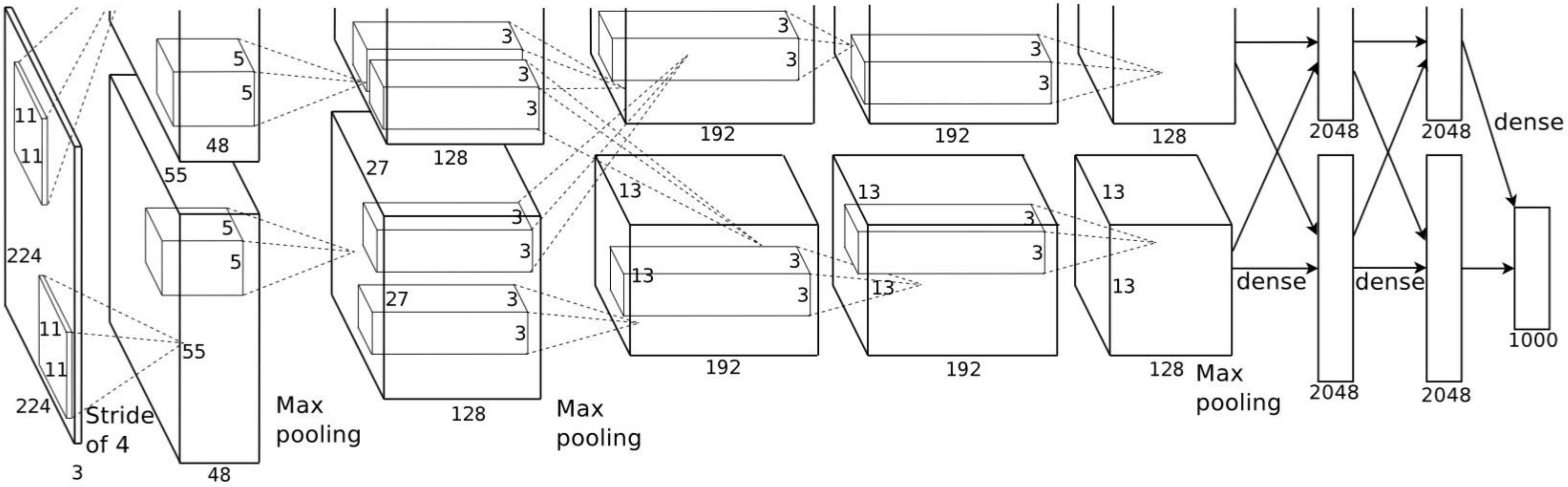
Less computations with filter groups



Did we see this before?

complexity $\propto (C \times K) / G$

AlexNet

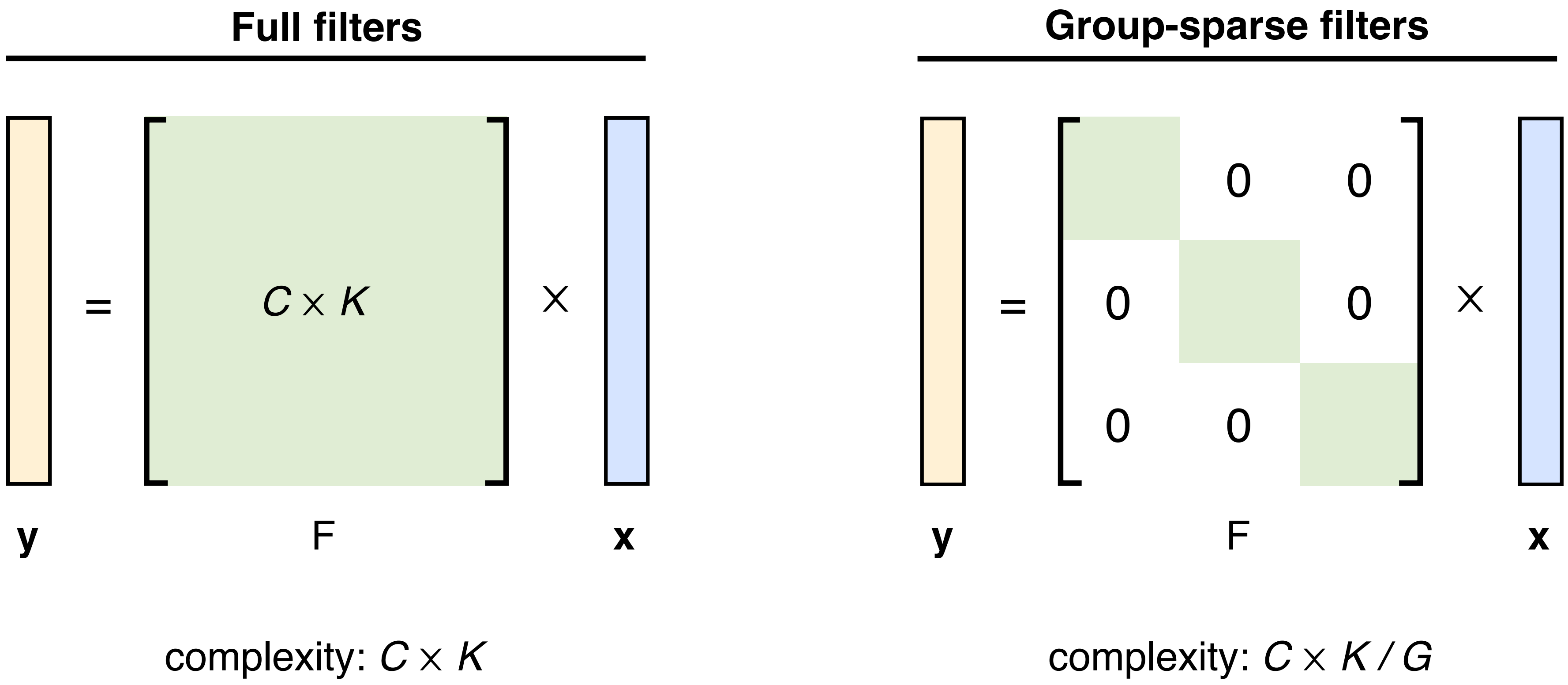


A. Krizhevsky, I. Sutskever, and G. E. Hinton. **Imagenet classification with deep convolutional neural networks**. In NIPS 2012.

Design Guidelines

Guideline 4:

Less computations with filter groups

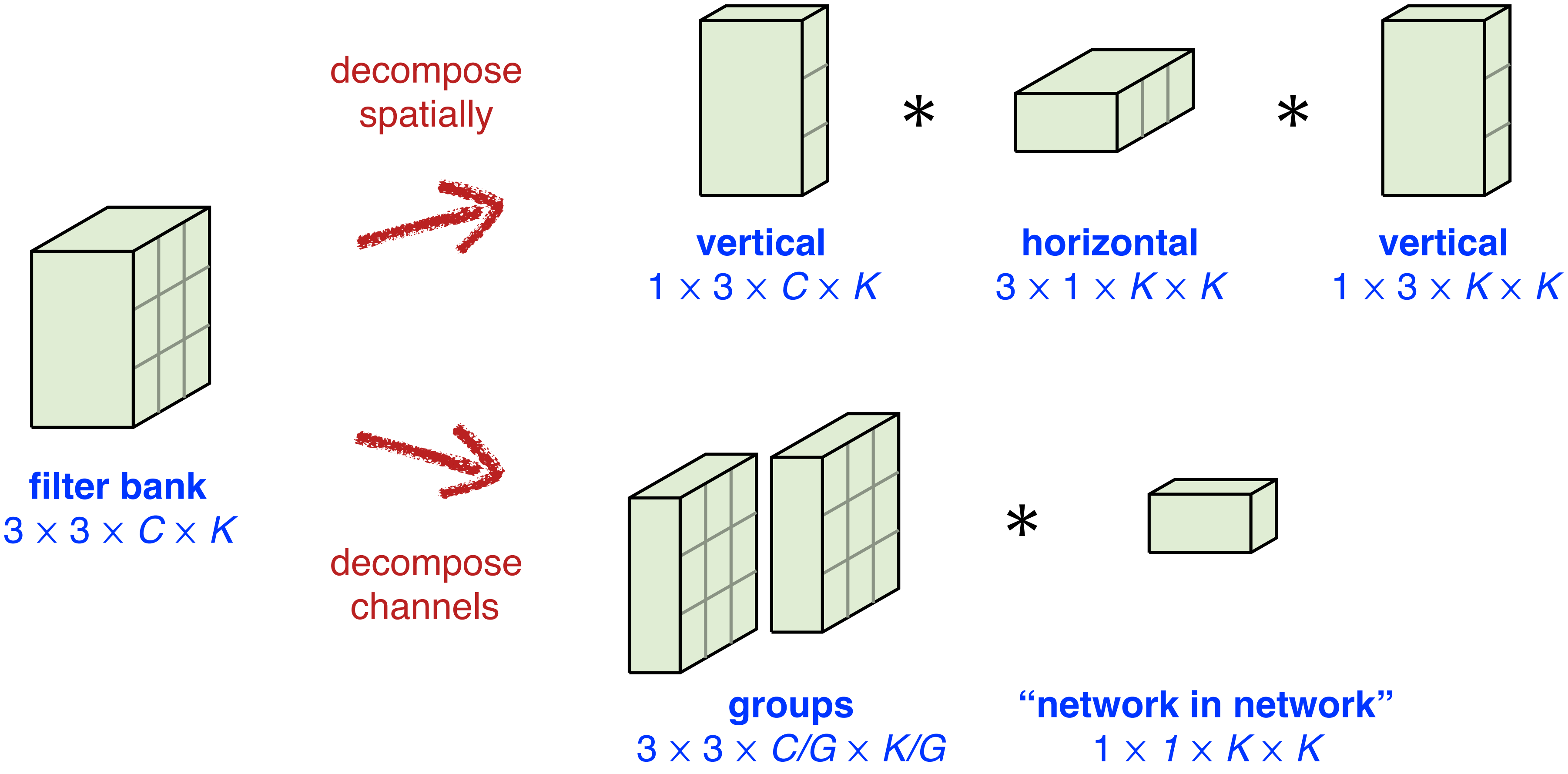


Groups = filters, seen as a matrix, have a “block” structure

Design Guidelines

Guideline 5:

Low-rank decompositions

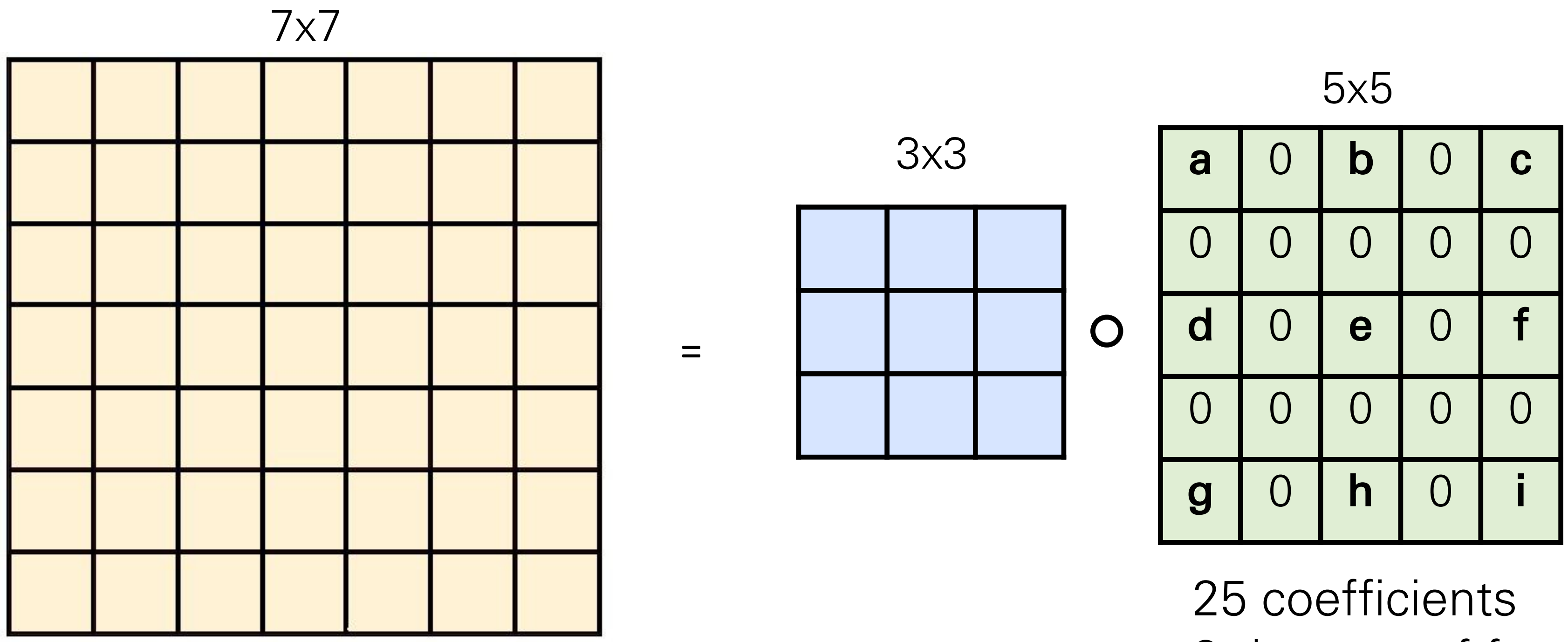


Make sure to mix the information

Design Guidelines

Guideline 6:

Dilated Convolutions



49 coefficients
18 degrees of freedom

25 coefficients
9 degrees of freedom

Exponential expansion of the receptive field without loss of resolution

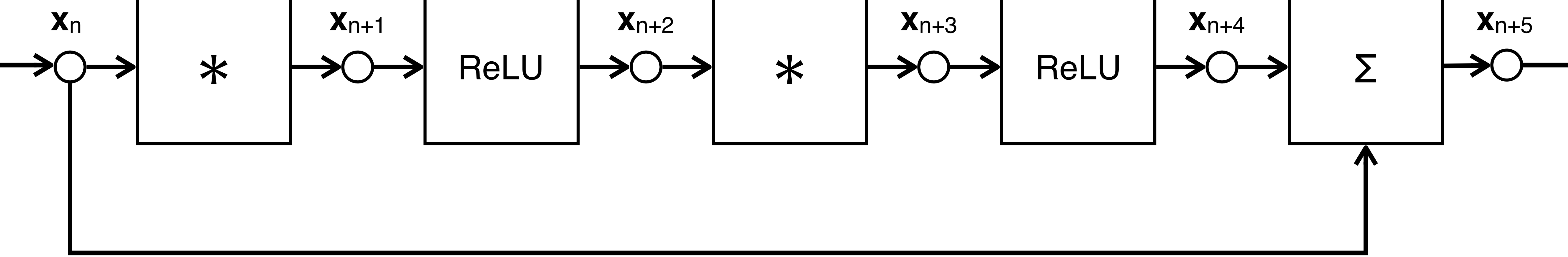
A Closer Look to Residual Learning

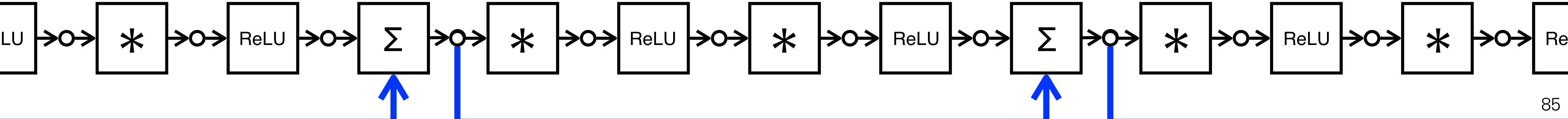
Residual Learning

Fixed identity
// learned residual

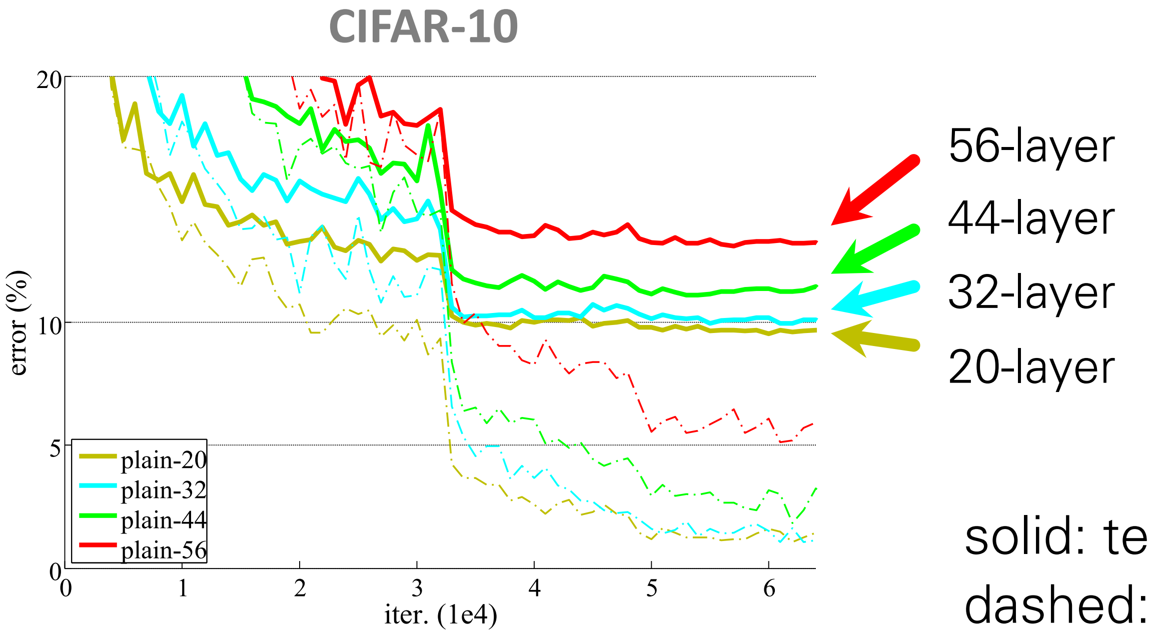
$$\mathbf{x}_{n+5} = \underbrace{\mathbf{x}_n}_{\text{identity}} + \underbrace{(\phi_{\mathrm{ReLU}} \circ \phi_* \circ \phi_{\mathrm{ReLU}} \circ \phi_*)(\mathbf{x}_n)}_{\text{residual}}$$

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

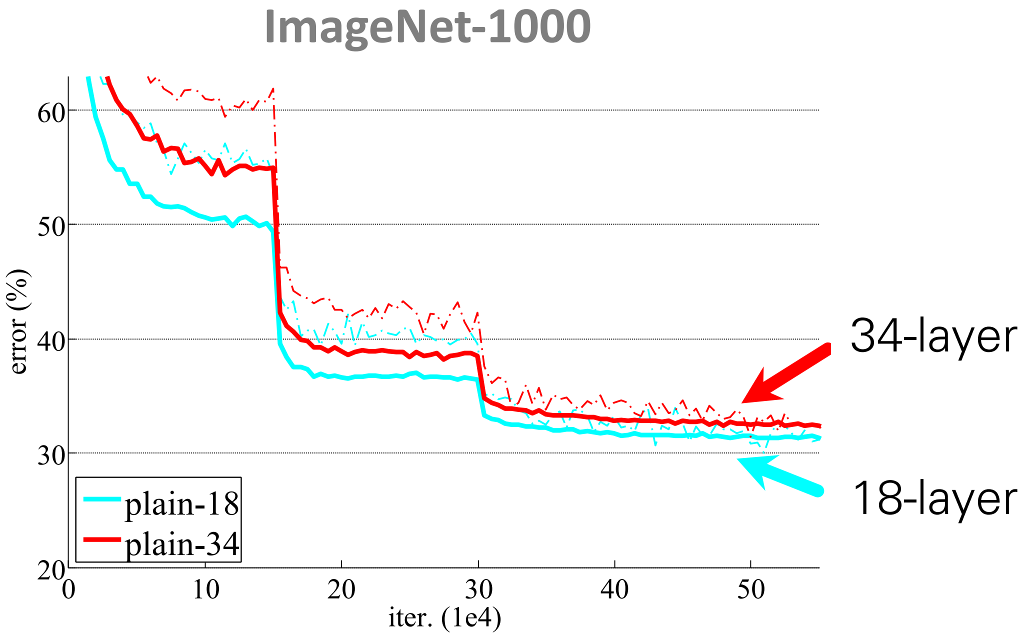




Residual Learning



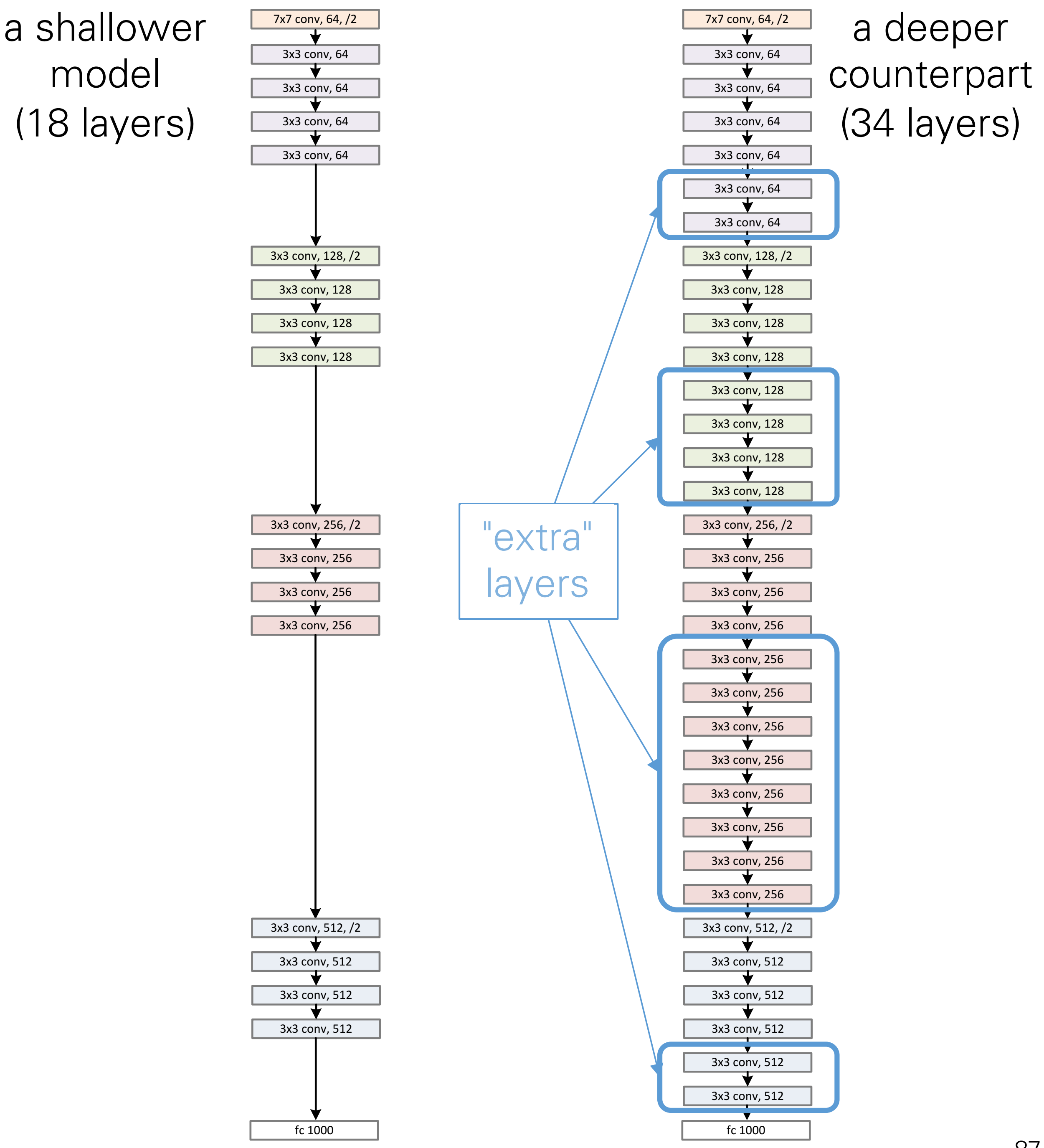
solid: test/val
dashed: train



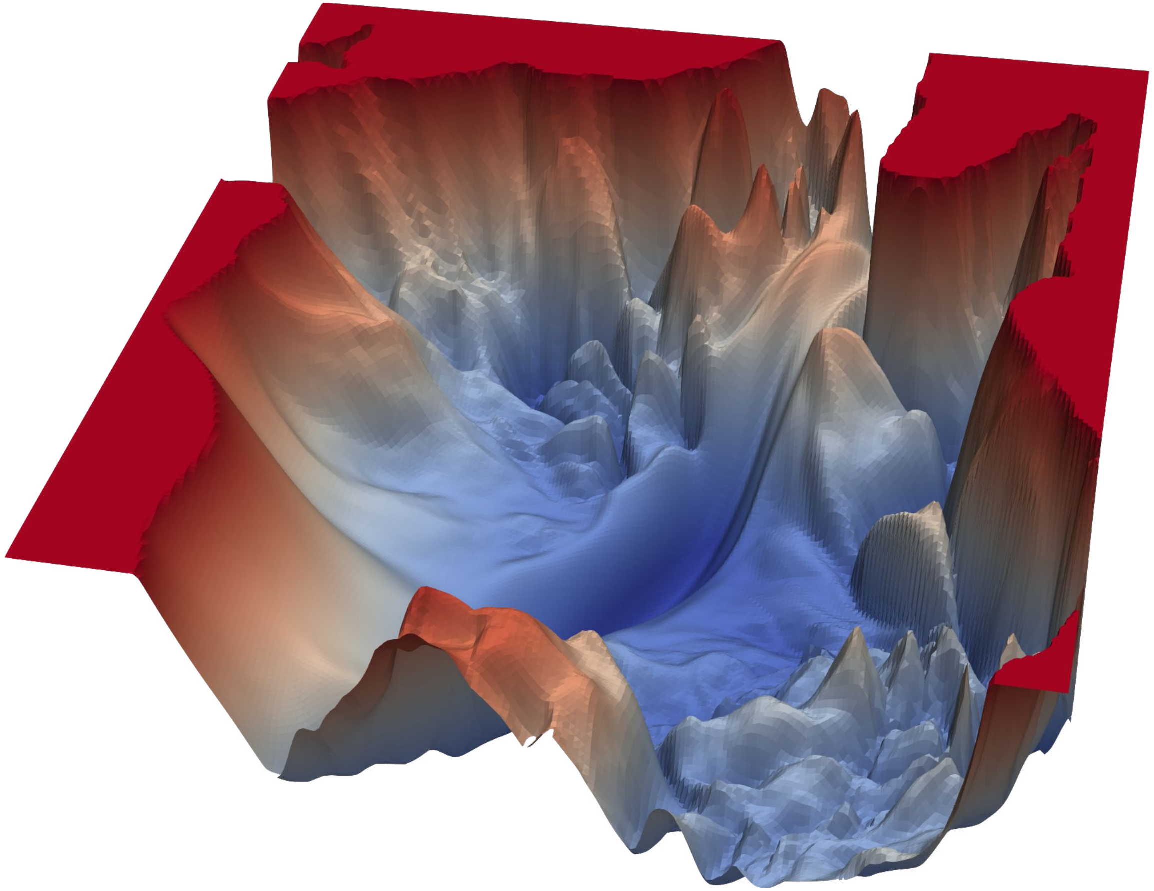
- "Overly deep" plain nets have **higher training error**
- A general phenomenon, observed in many datasets
- This is optimization issue, deeper models are harder to optimize

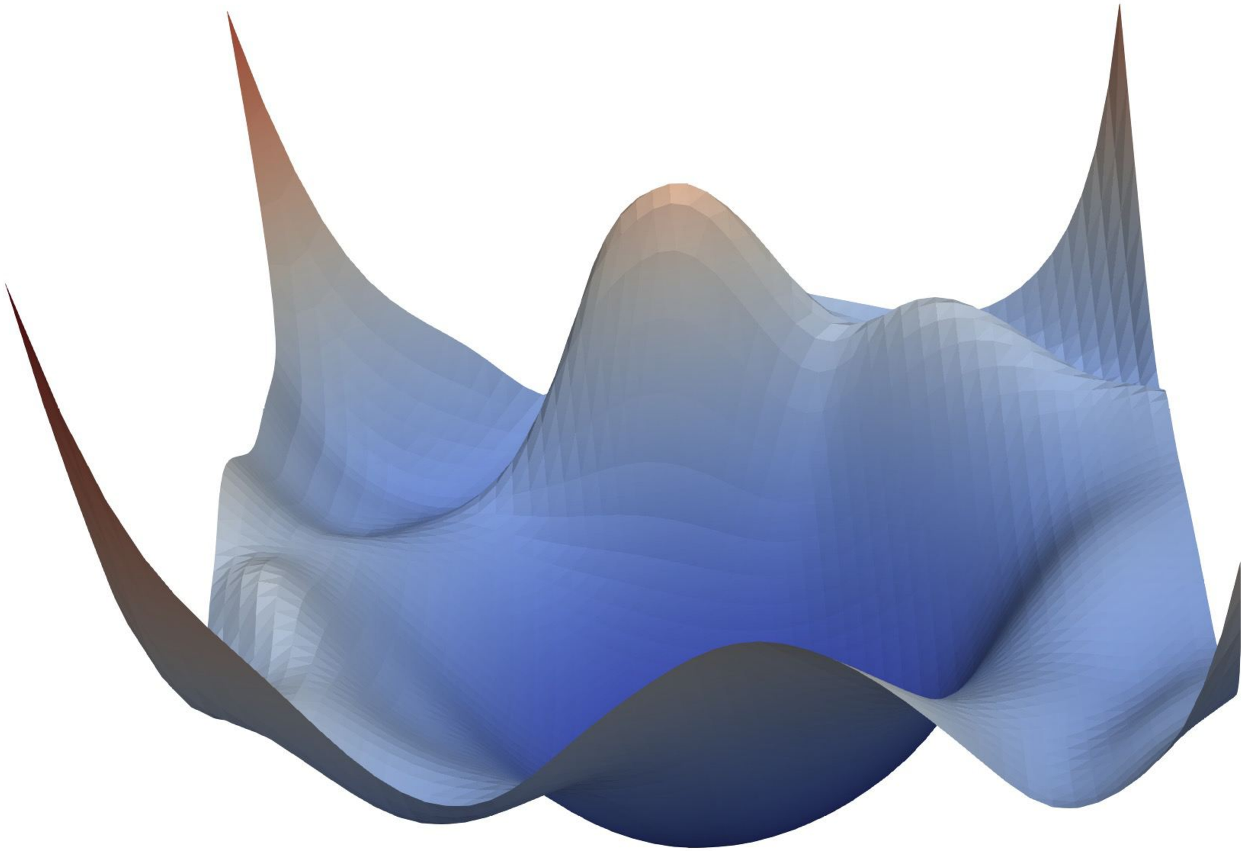
Residual Learning

- Richer solution space
- A deeper model should not have **higher training error**
- A solution by construction:
  - original layers: copied from a
  - learned shallower model
  - extra layers: set as identity
  - at least the same training error



Residual Learning





- The loss surface of a 56-layer net using the CIFAR-10 dataset, both without (left) and with (right) residual connections.

Hao Li et al., "Visualizing the Loss Landscape of Neural Nets". ICLR 2018

Transfer Learning with Convolutional Neural Networks

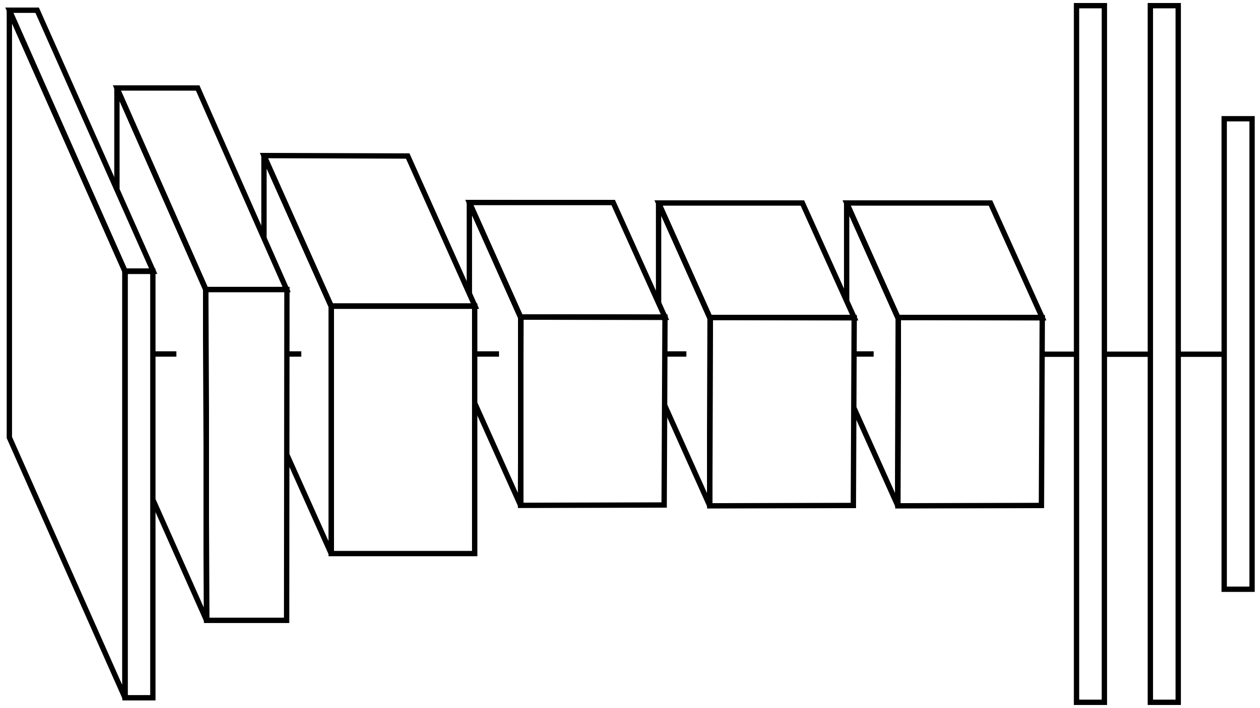
Beyond CNNs

- Do features extracted from the CNN generalize other tasks and datasets?
  - Donahue et al. (2013), Chatfield et al. (2014), Razavian et al. (2014), Yosinski et al. (2014), etc.

- CNN activations as deep features
- Finetuning CNNs

CNN activations as deep features

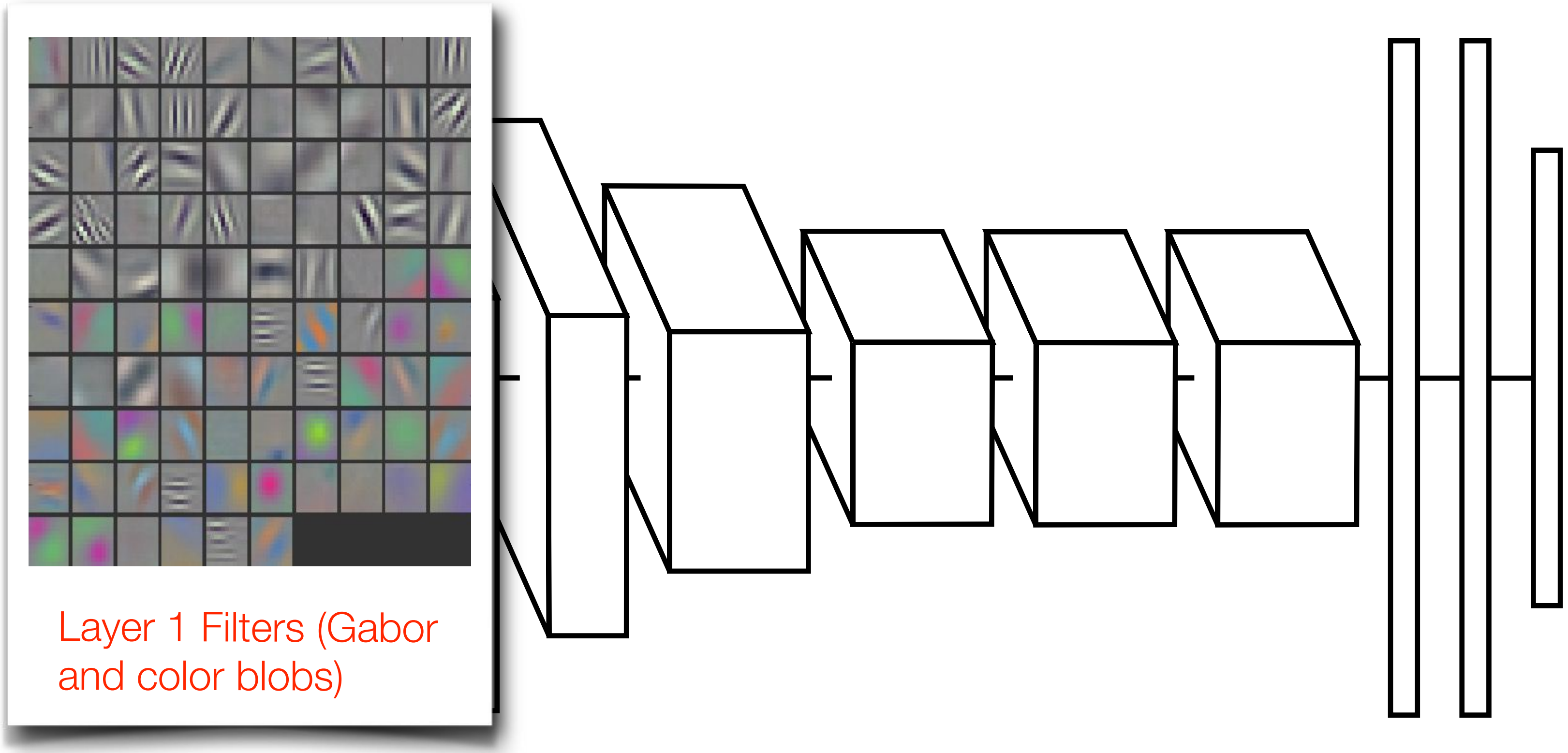
- CNNs discover effective representations. Why not to use them?



Slide credit: Jason Yosinski

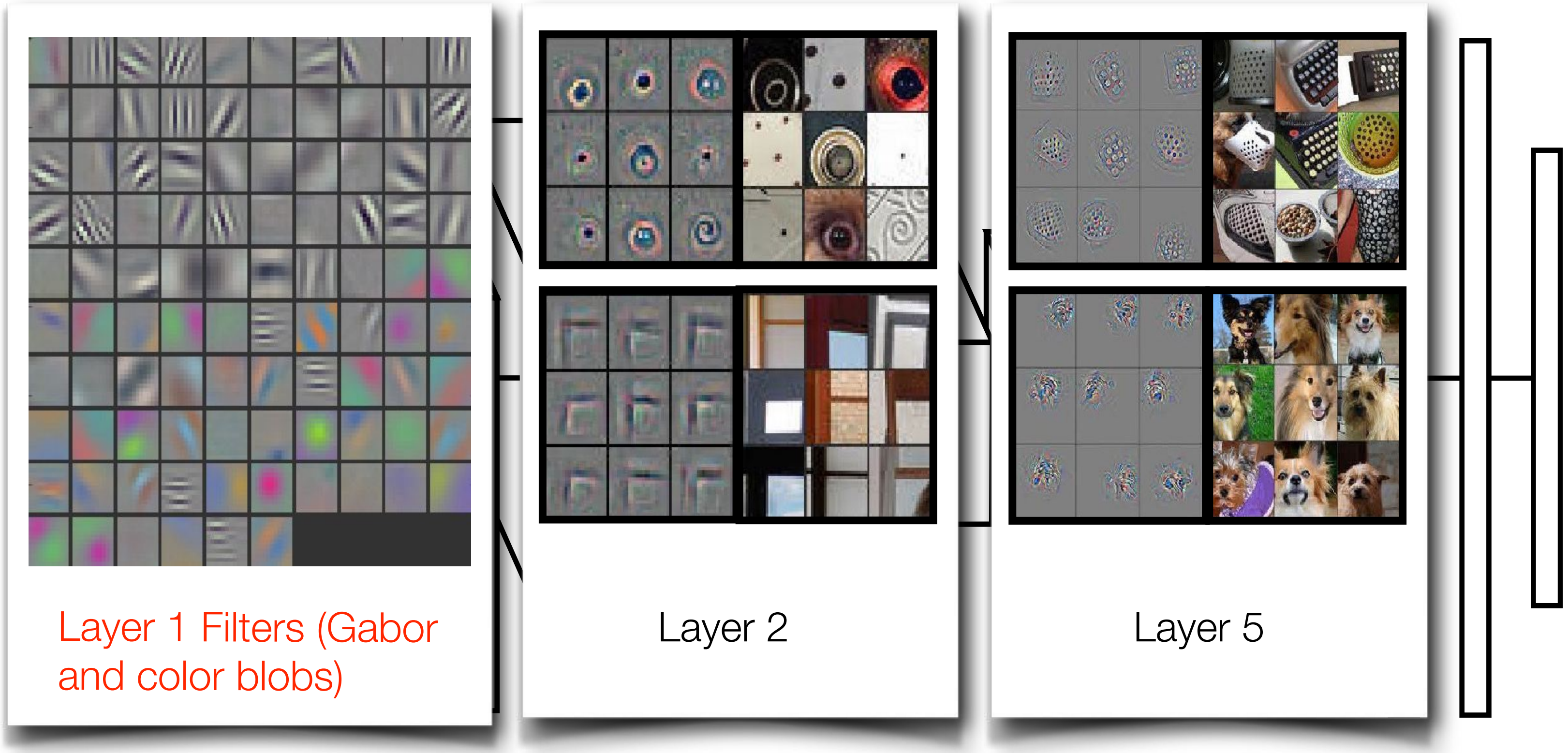
CNN activations as deep features

- CNNs discover effective representations. Why not to use them?



CNN activations as deep features

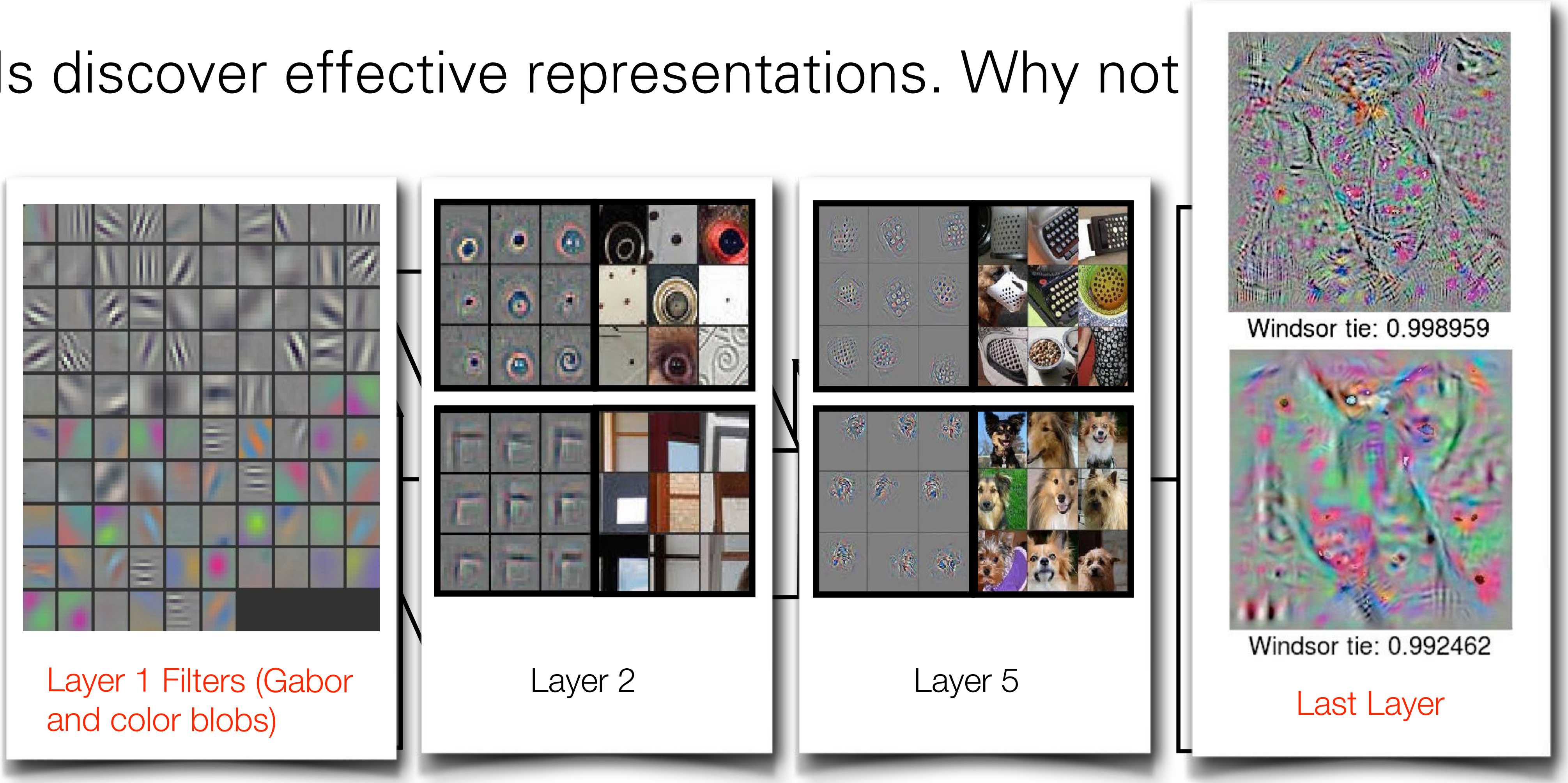
- CNNs discover effective representations. Why not to use them?



Zeiler et al., 2014

CNN activations as deep features

- CNNs discover effective representations. Why not



Zeiler et al., 2014

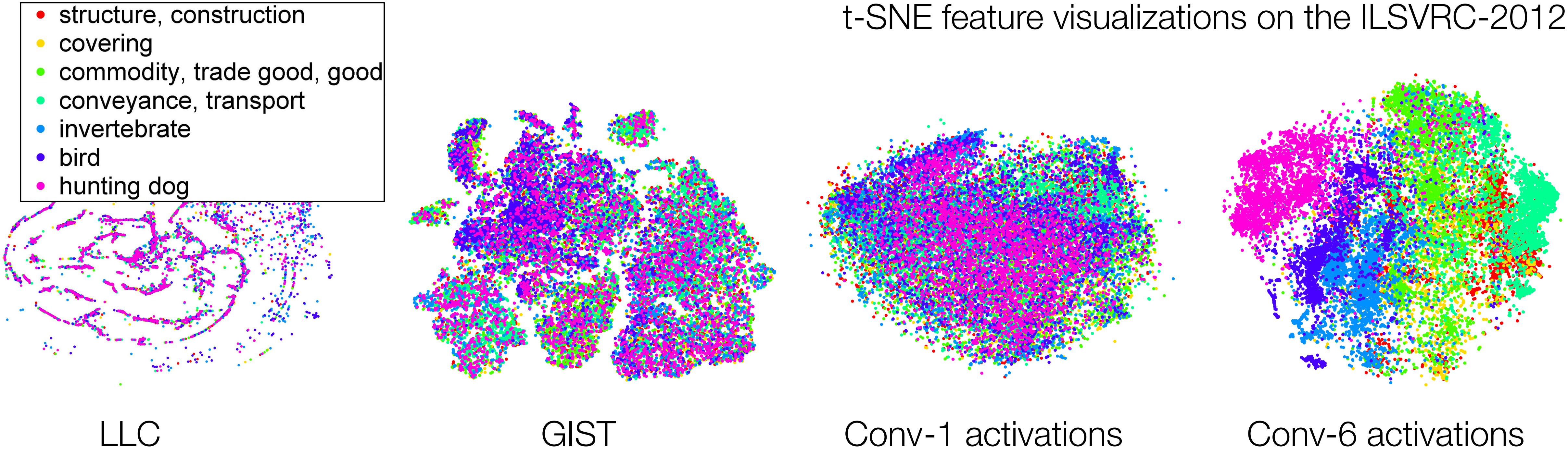
Nguyen et al., 2014

Slide credit: Jason Yosinski

CNNs as deep features

• CNNs discover effective representations. Why not to use them?

t-SNE feature visualizations on the ILSVRC-2012



Donahue et al. **DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition**, 2014

Transfer Learning with CNNs

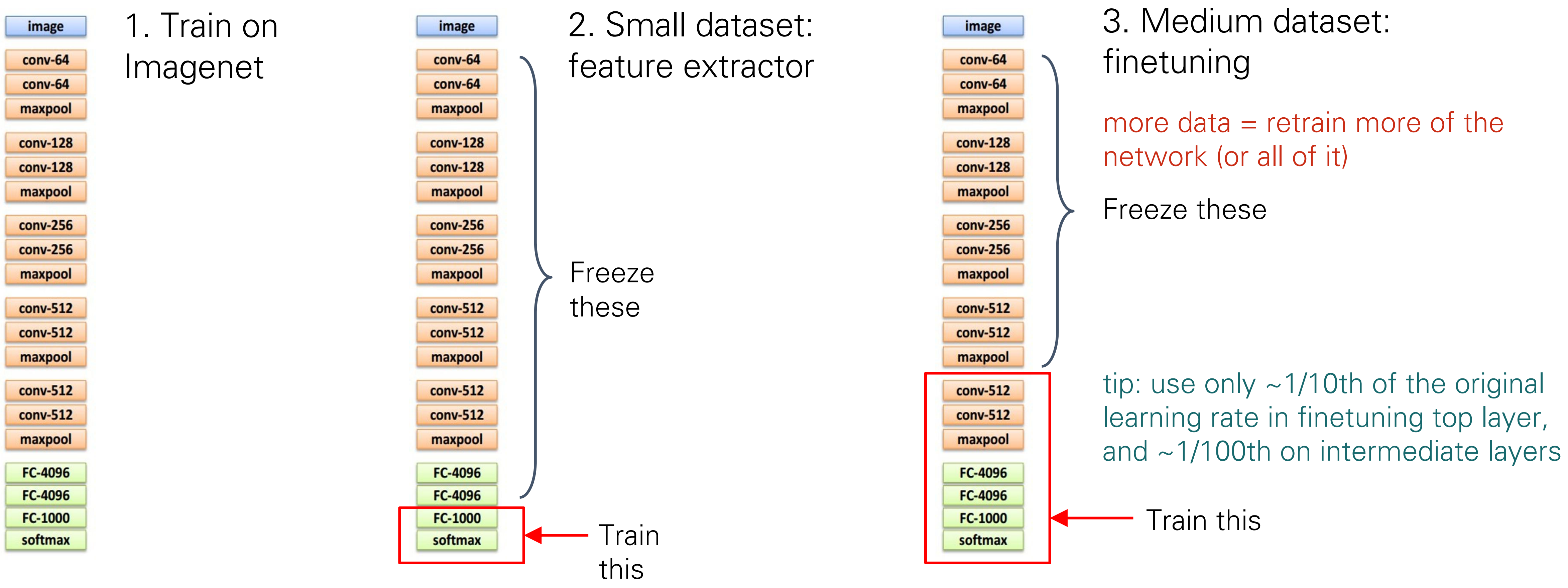
- A CNN trained on a (large enough) dataset generalizes to other visual tasks

A. Joulin, L.J.P. van der Maaten, A. Jabri, and N. Vasilache **Learning visual features from Large Weakly supervised Data.** ECCV 2016

Slide credit: Joan Bruna

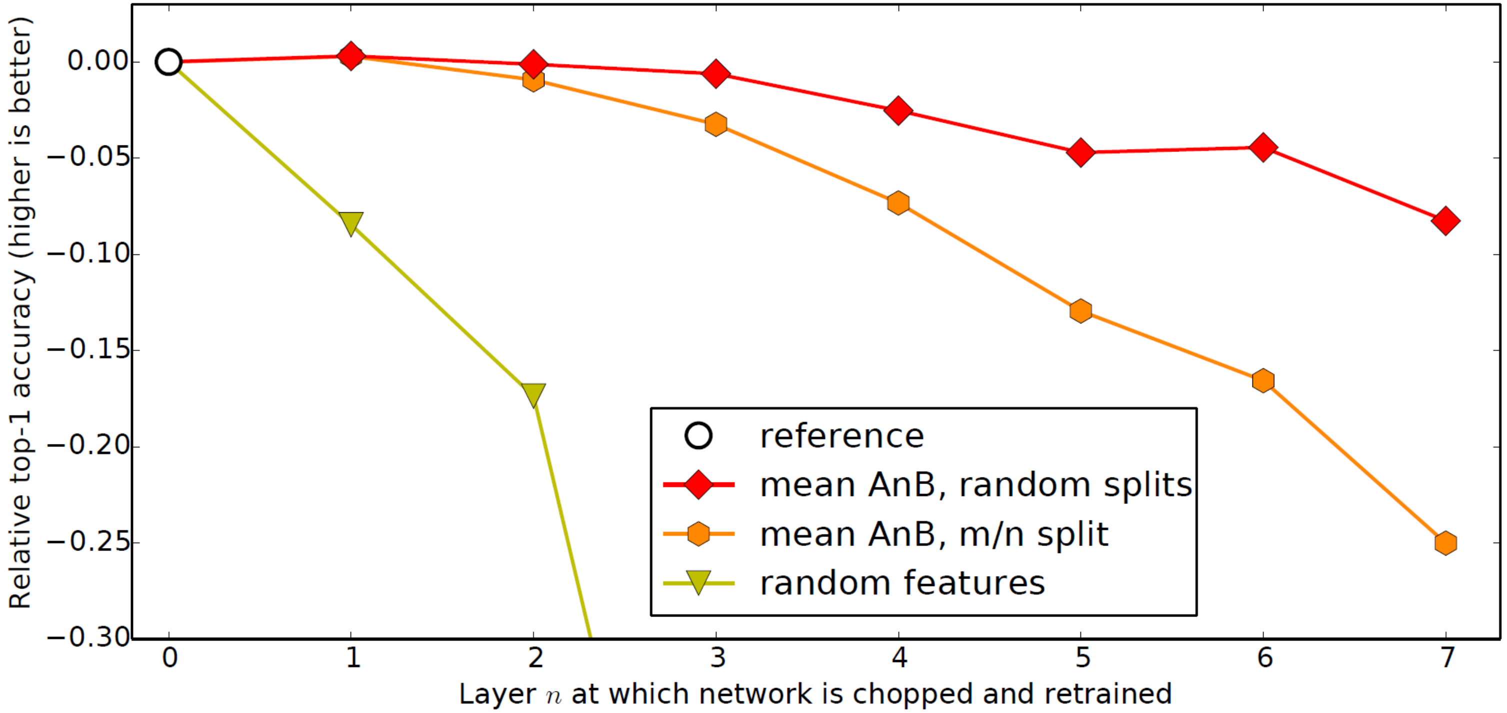
Transfer Learning with CNNs

- Keep layers 1-7 of our ImageNet-trained model fixed
- Train a new softmax classifier on top using the training images of the new dataset.



How transferable are features in CNN networks?

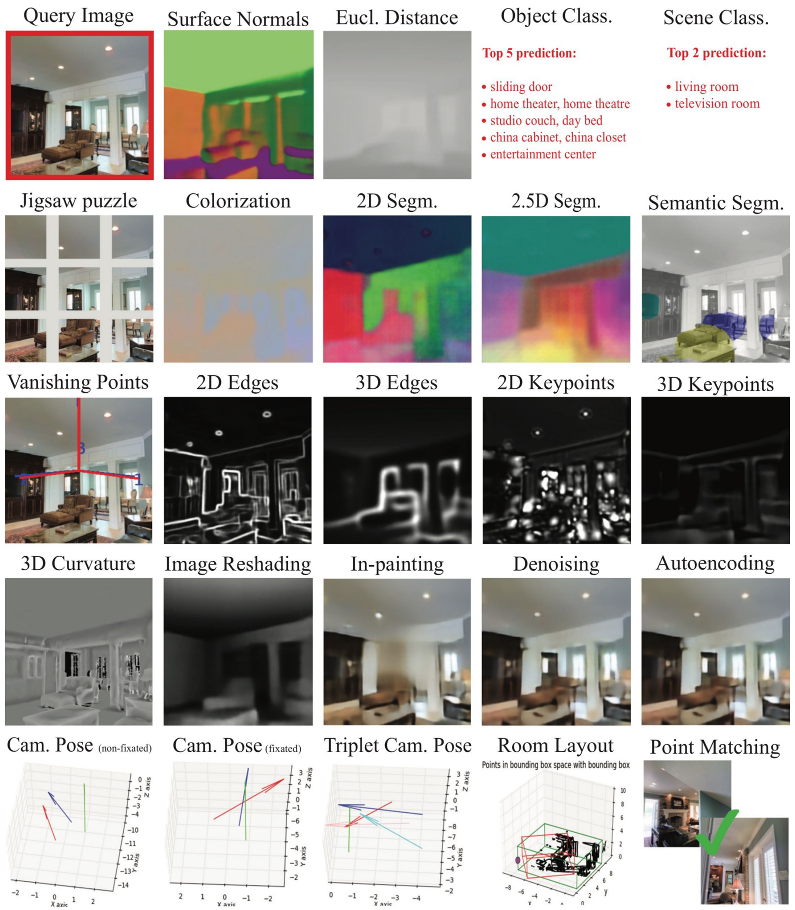
- Divide ImageNet into man-made objects A (449 classes) and natural objects B (551 classes)
- The transferability of features decreases as the distance between the base task and target task increases

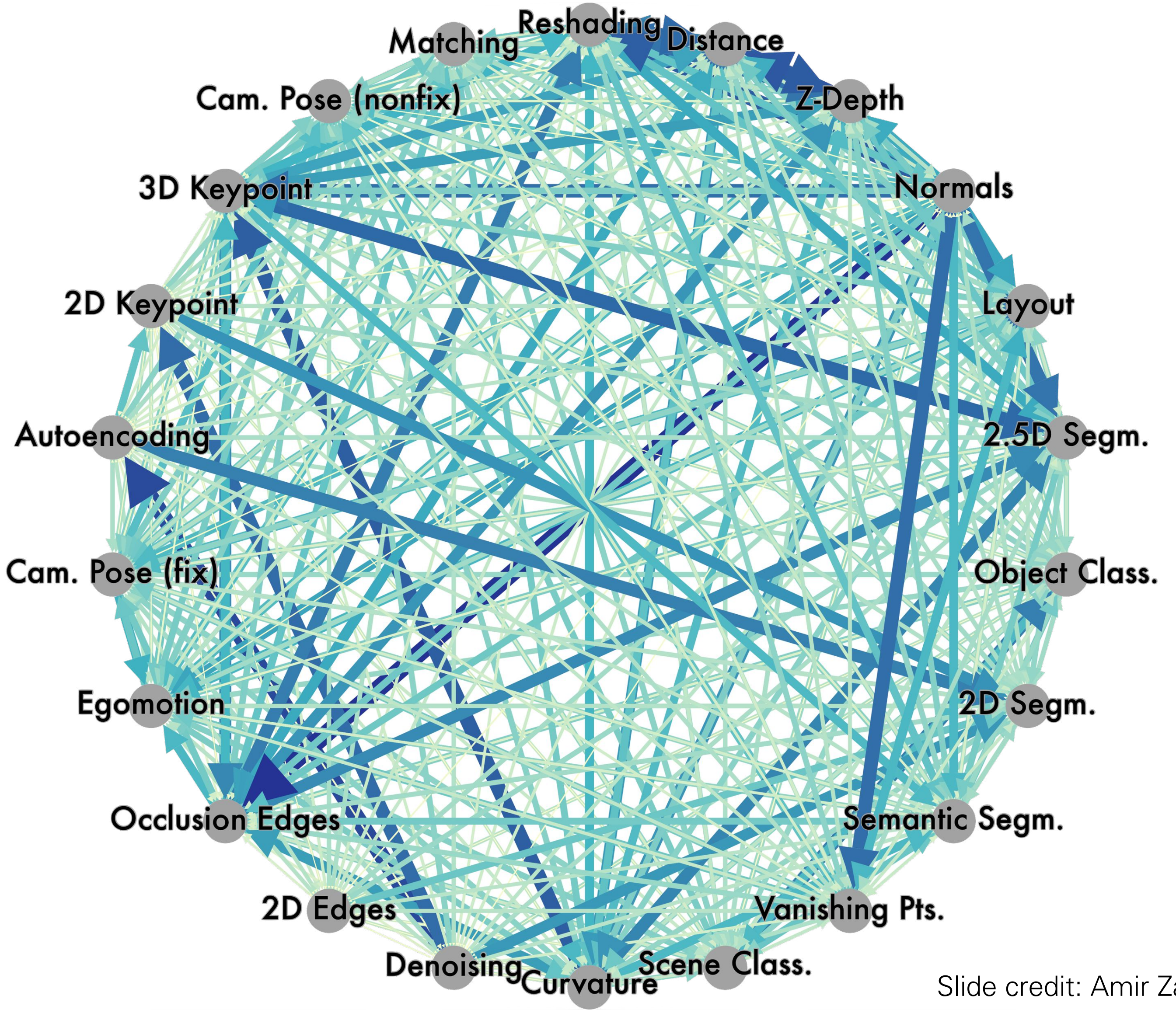


Slide credit: Xiaogan Wang

How transferable are features in CNN networks?

- An open research problem





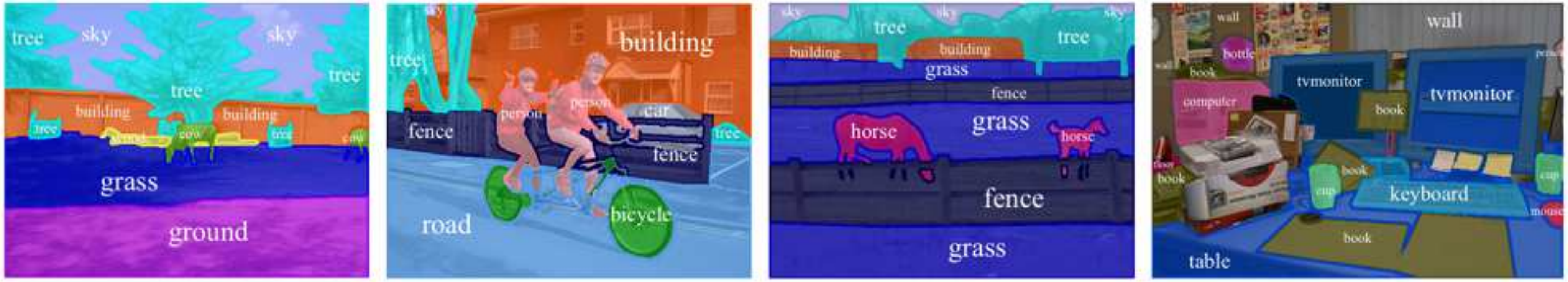
Slide credit: Amir Zamir

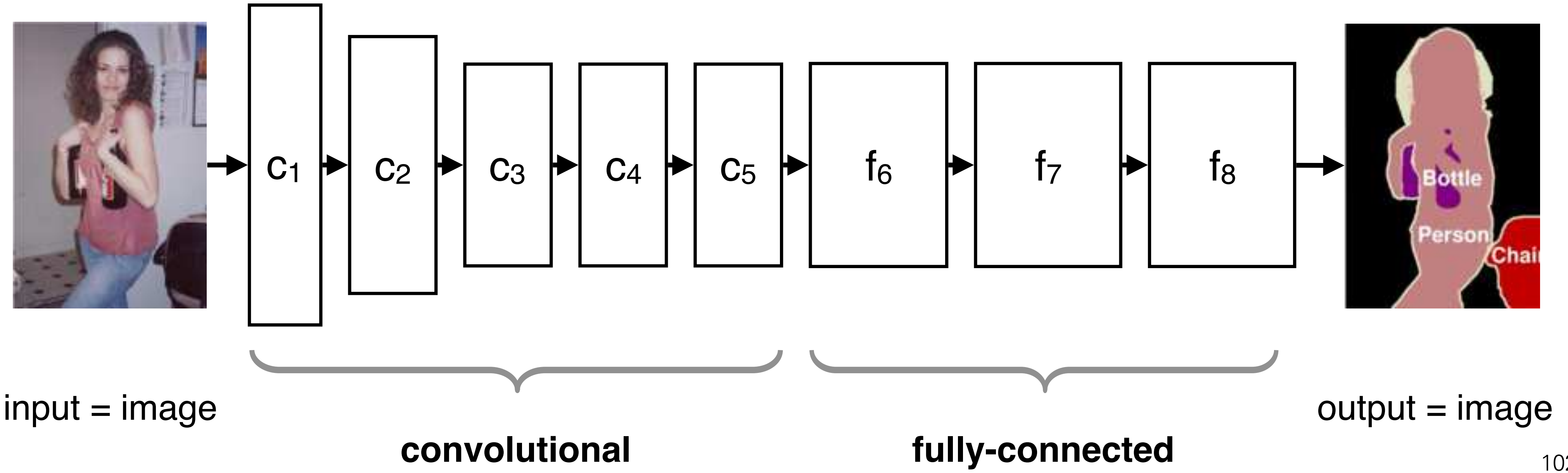
A. Zamir, A. Sax, W. Shen, L. Guibas, J. Malik, S. Savarese. **Taskonomy: Disentangling Task Transfer Learning**. CVPR 2018.

Semantic Segmentation

Semantic Image Segmentation

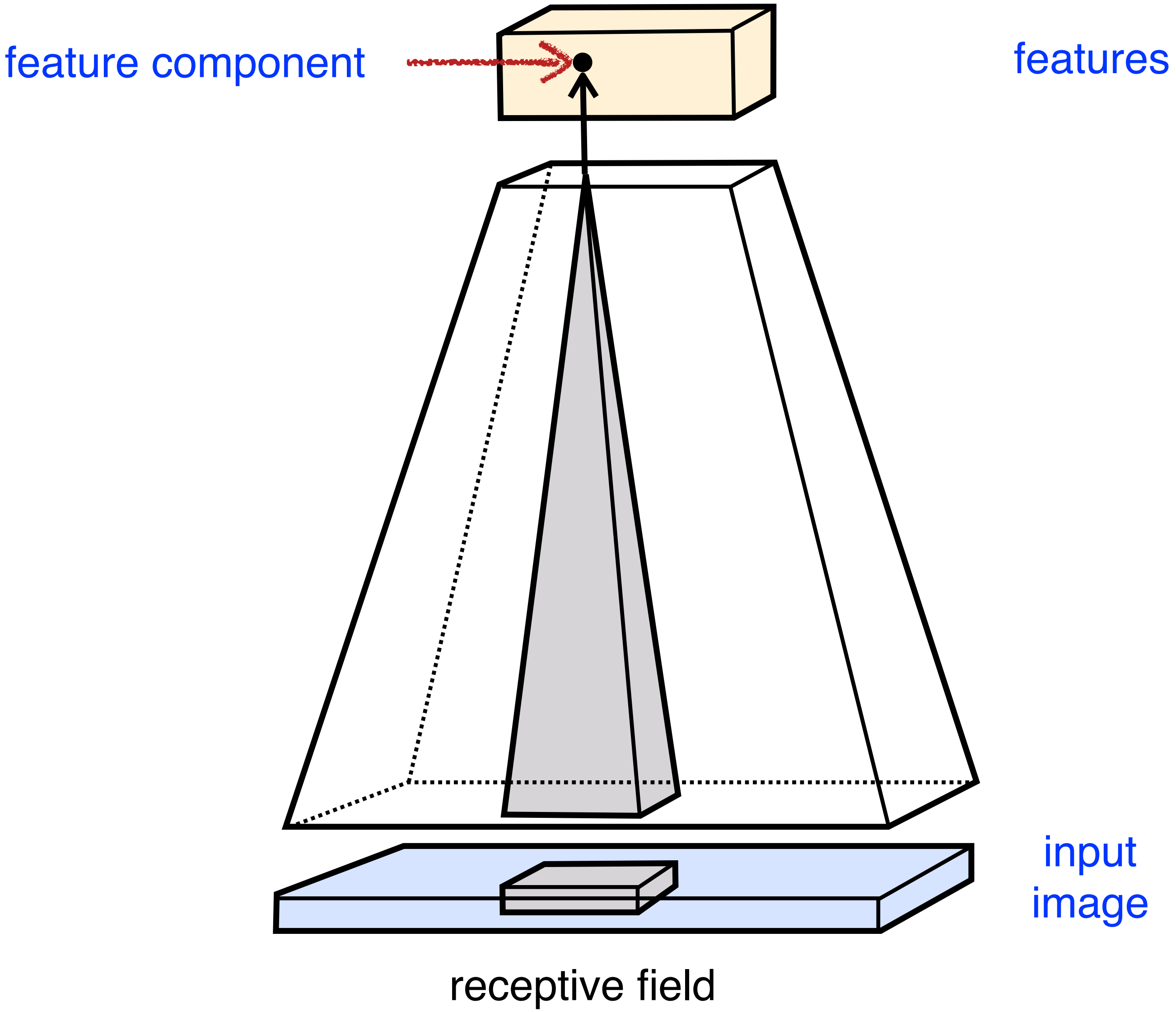
- Label individual pixels





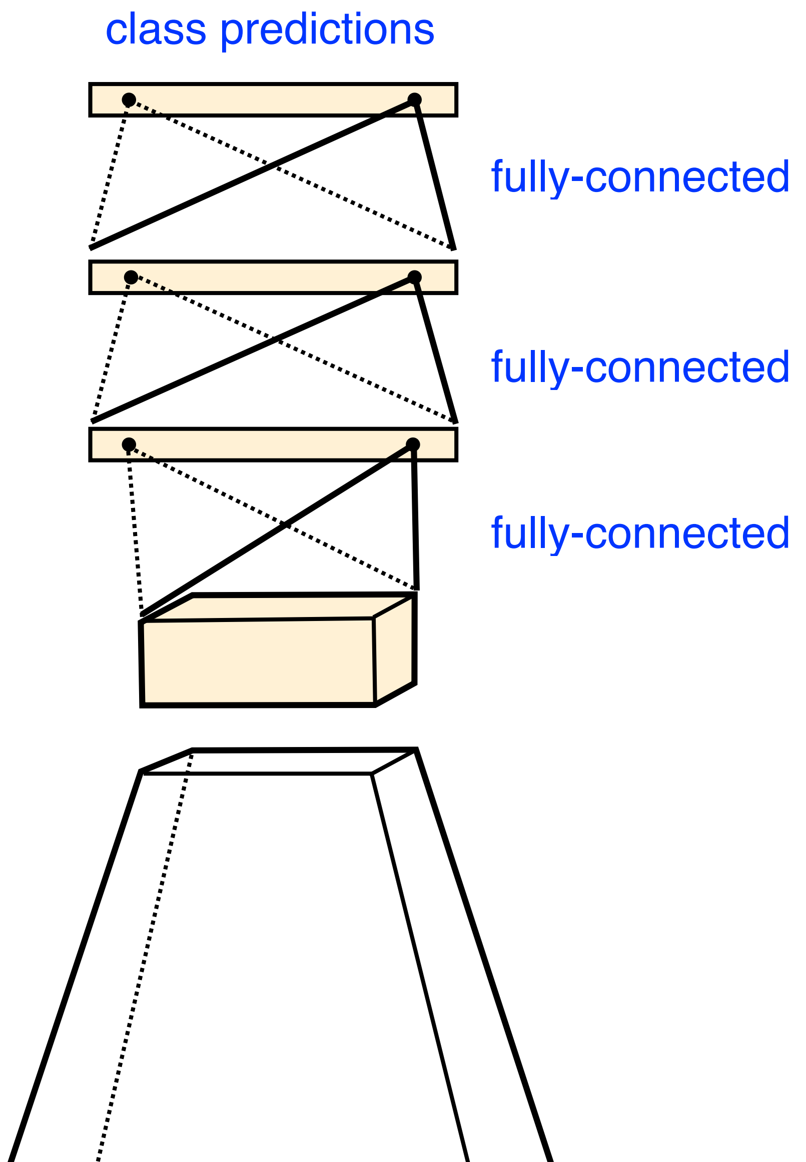
Convolutional Layers

- Local receptive field



Fully Connected Layers

- Global receptive field



Convolutional vs. Fully Connected

- Comparing the receptive fields

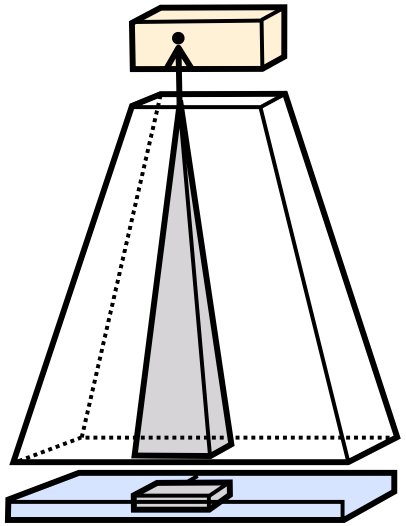
Downsampling filters

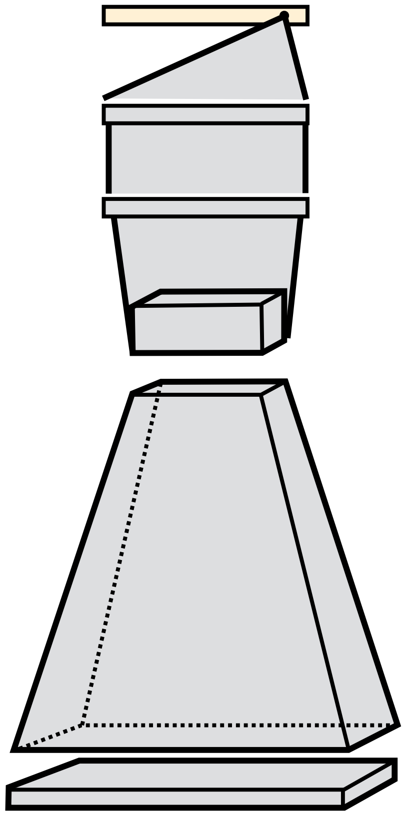
Responses are spatially selective, can be used to localize things.

Upsampling filters

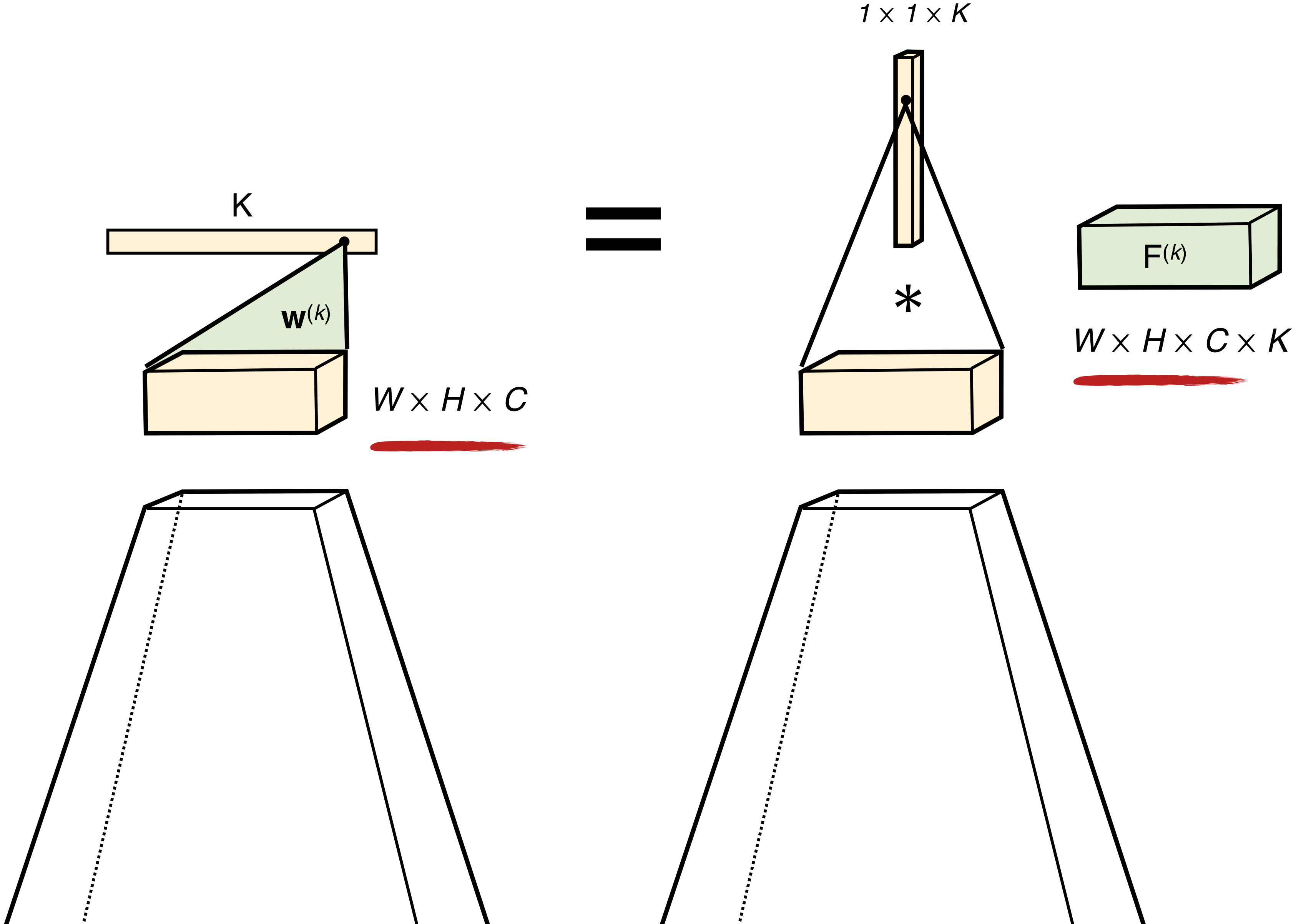
Responses are global, do not characterize well position

Which one is more useful for pixel level labelling?

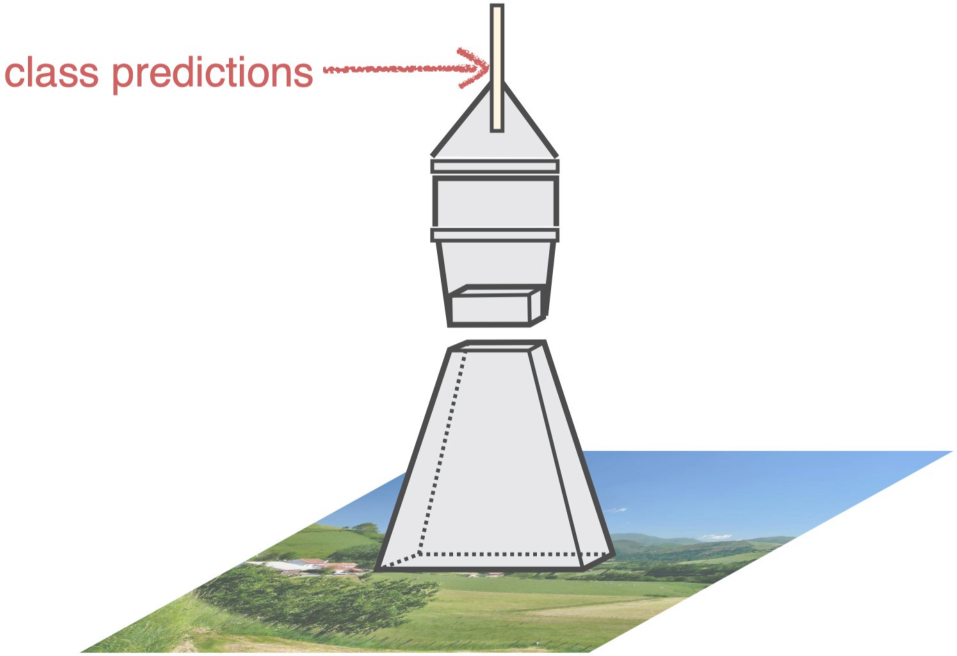




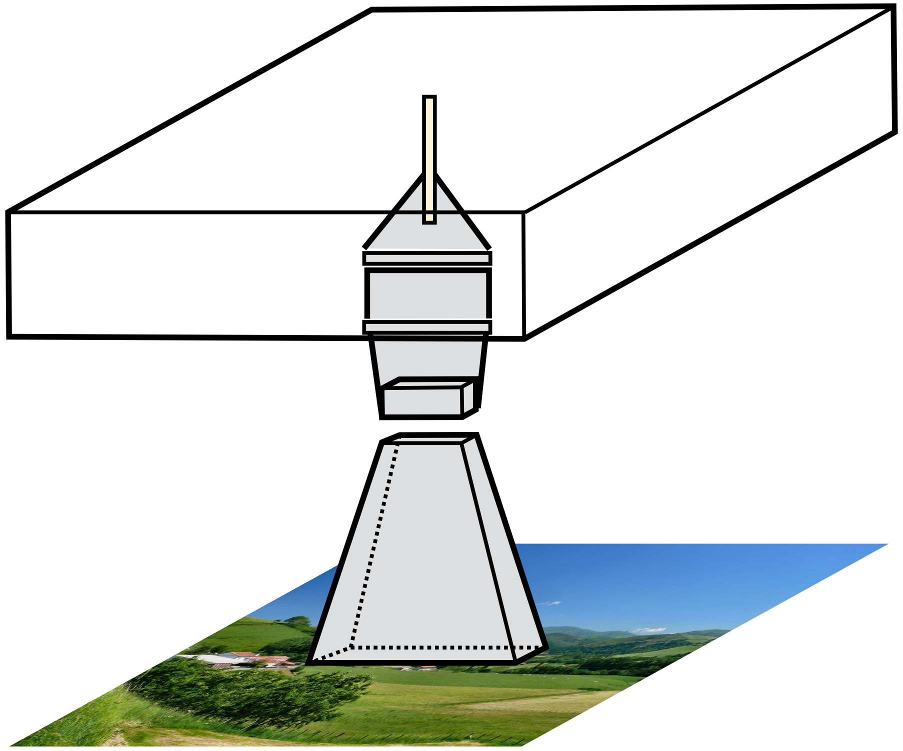
Fully-Connected Layer = Large Filter



Fully-Convolutional Neural Networks



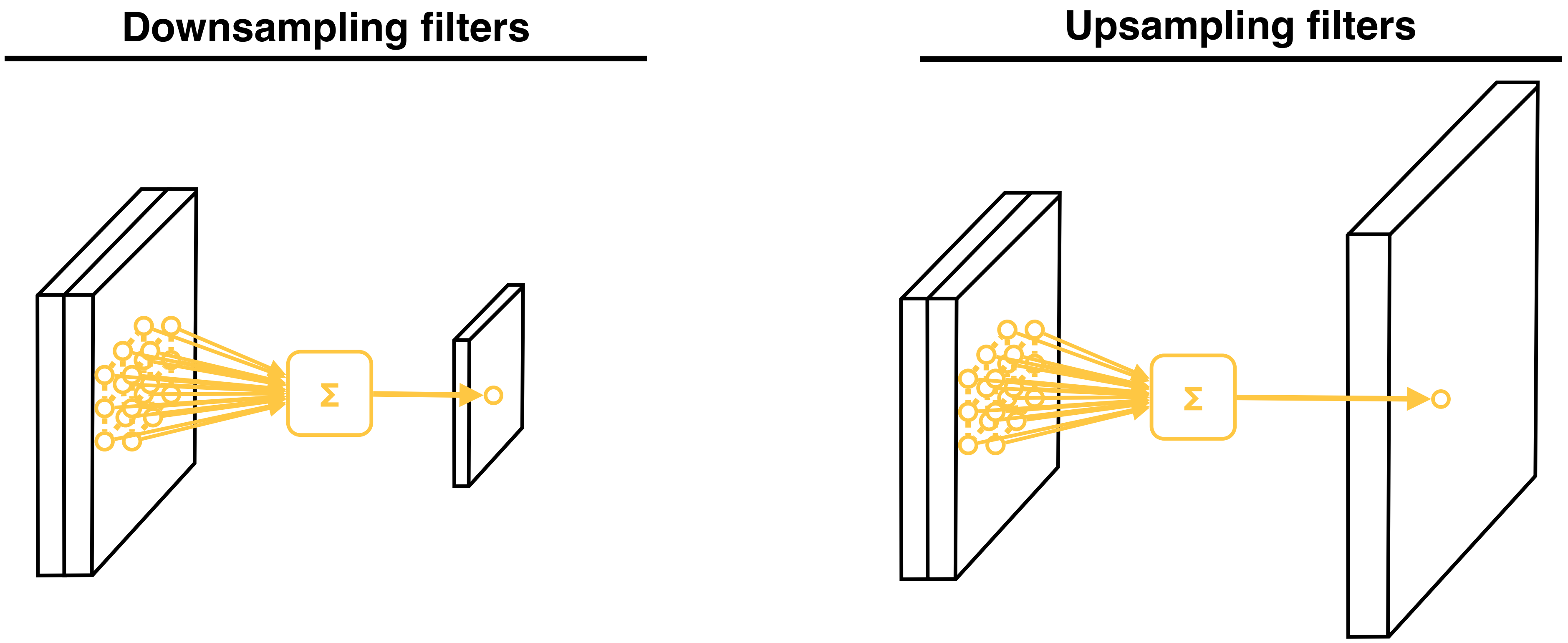
Fully-Convolutional Neural Networks



- **Dense evaluation**
  - Apply the whole network convolutional
  - Estimates a vector of class probabilities at each pixel
- **Downsampling**
  - In practice most network downsample the data fast
  - The output is very low resolution (e.g. 1/32 of original)

Upsampling The Resolution

- Interpolating filter

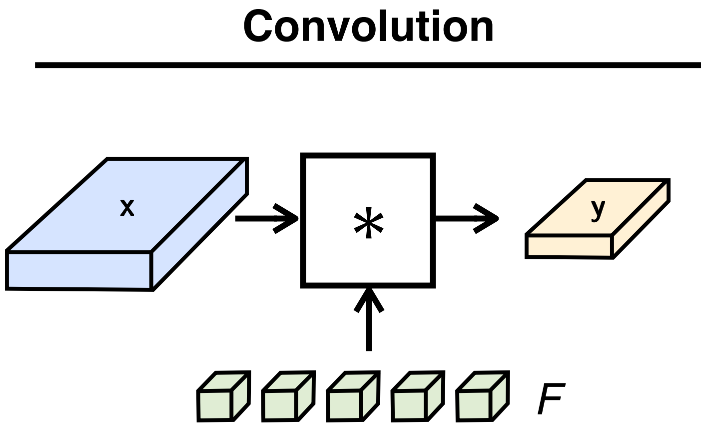


Upsampling filters allow to increase the resolution of the output

Very useful to get full-resolution segmentation results

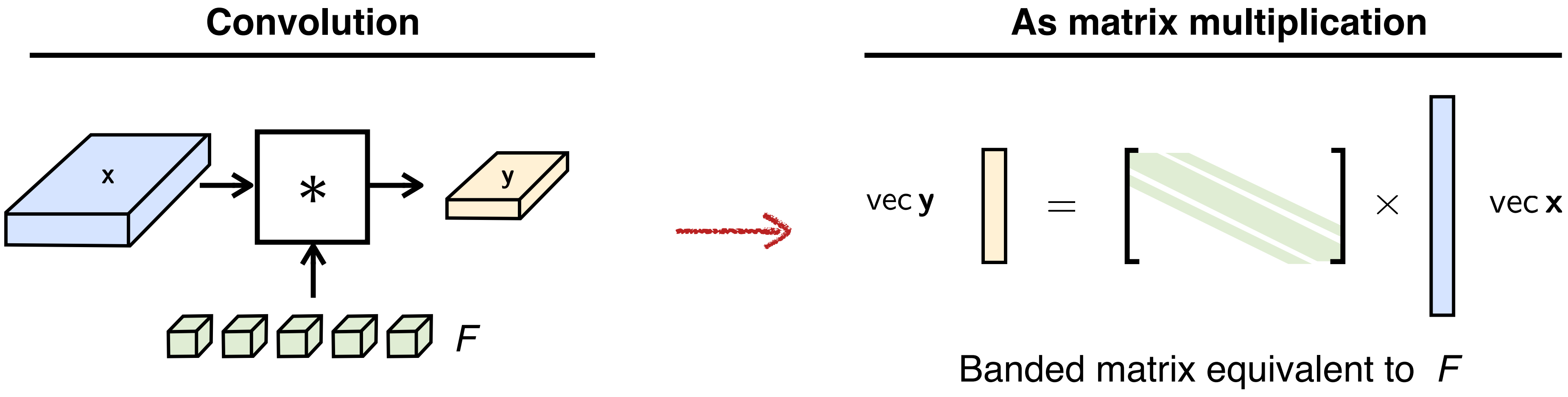
Deconvolution Layer

- Or convolution transpose



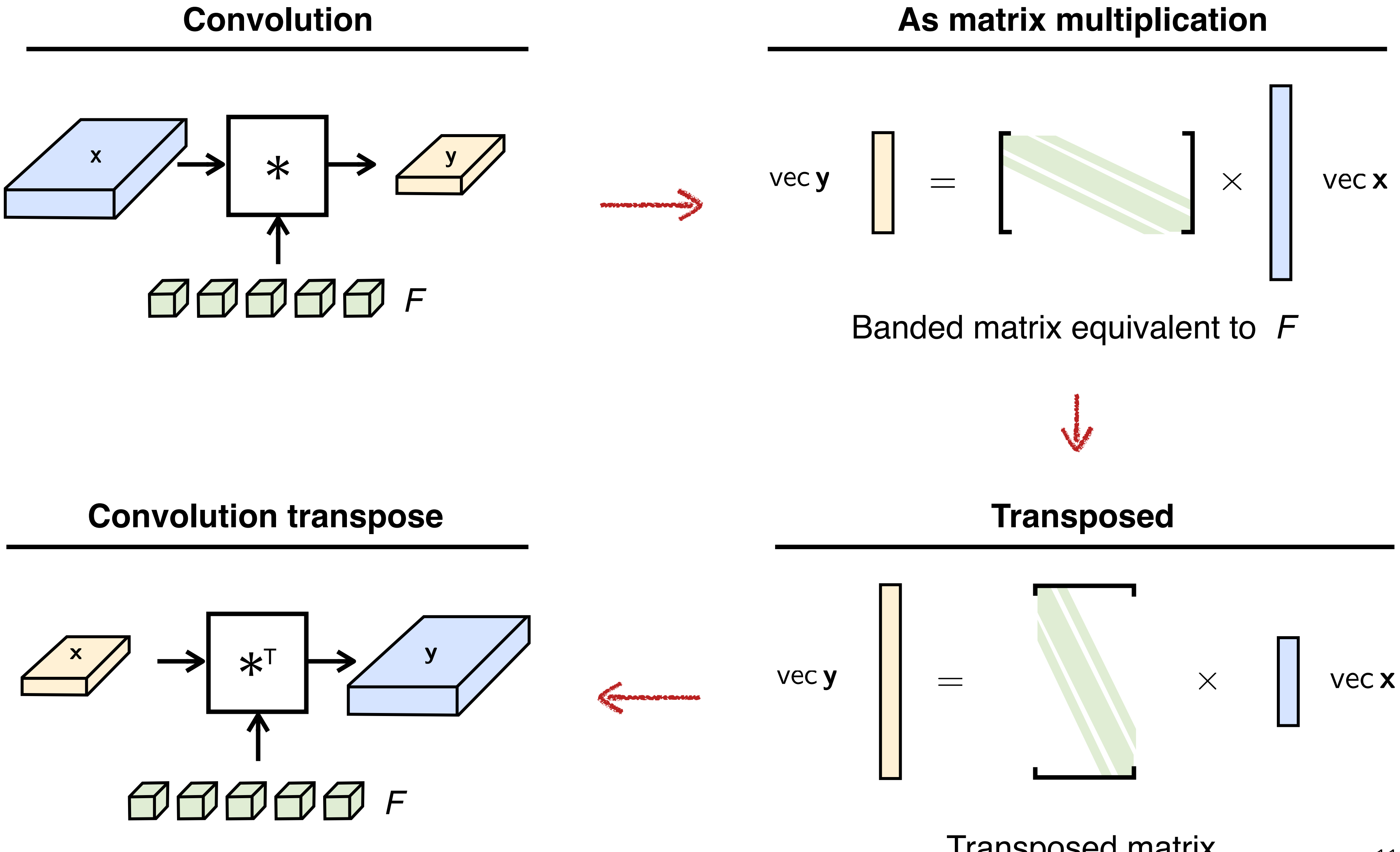
Deconvolution Layer

- Or convolution transpose



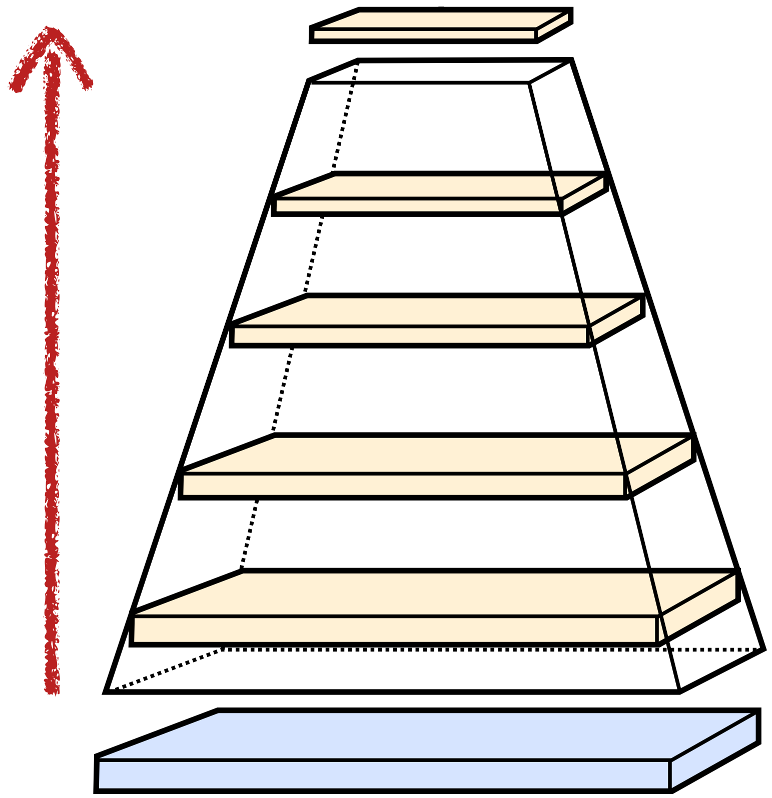
Deconvolution Layer

- Or convolution transpose



U-Architectures

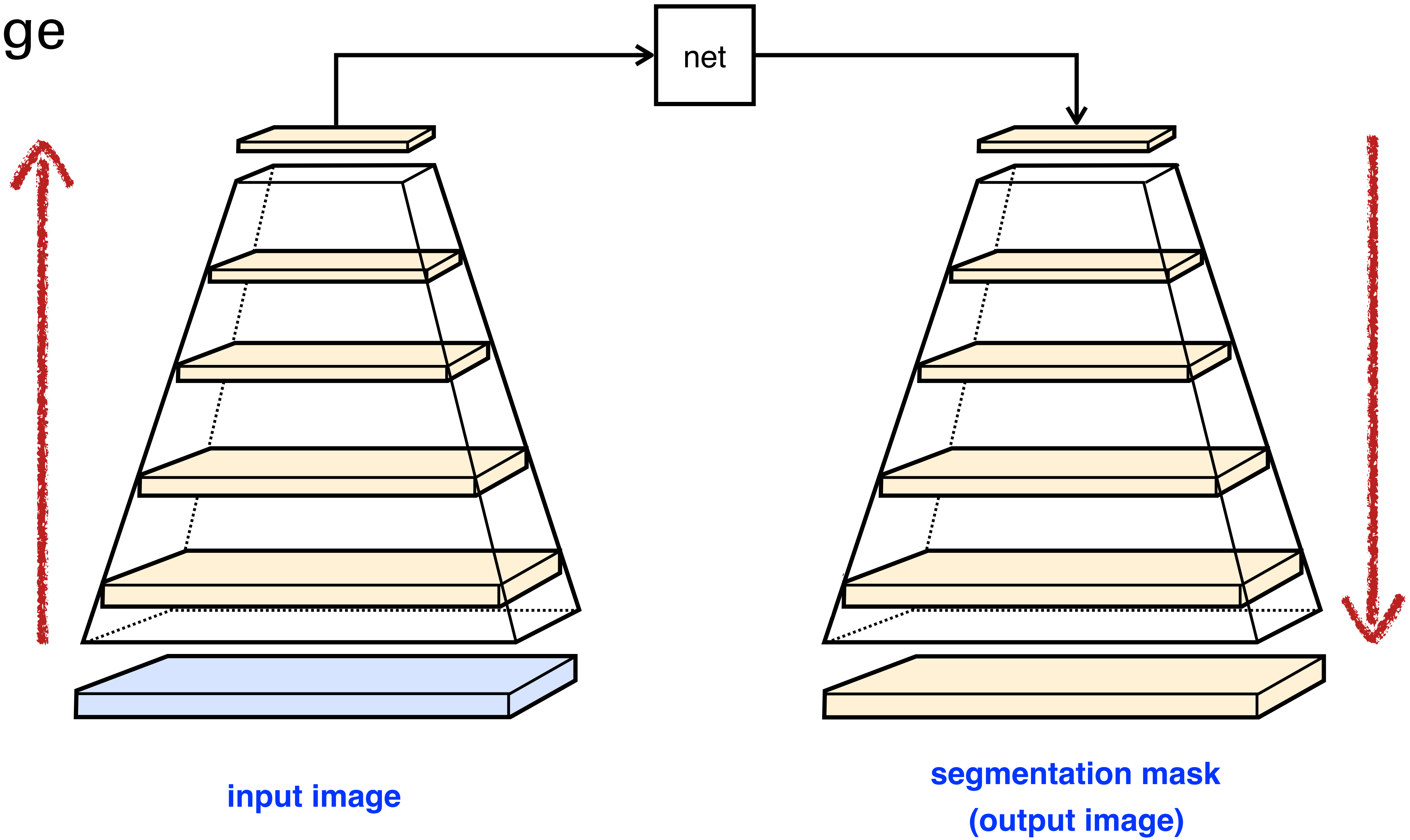
- Image to image



input image

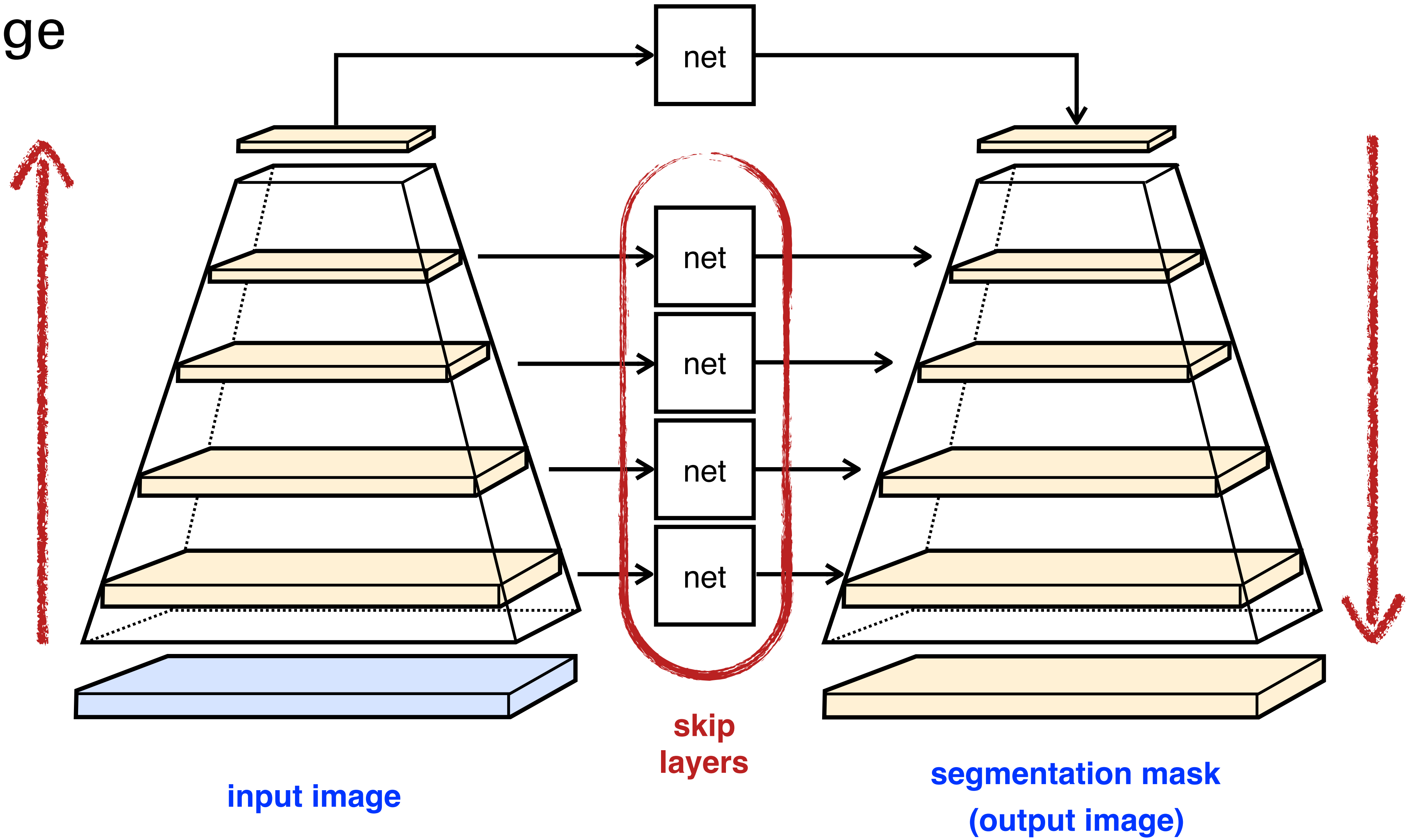
U-Architectures

- Image to image



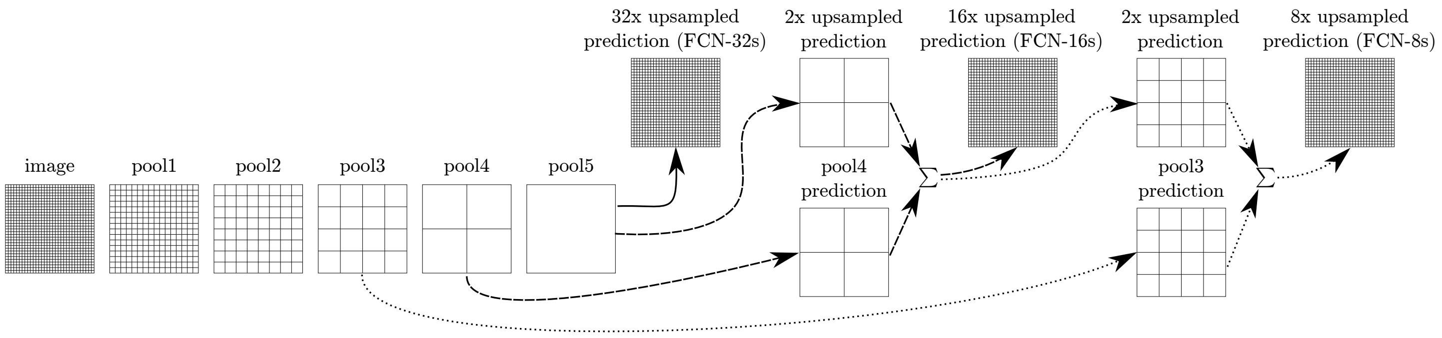
U-Architectures

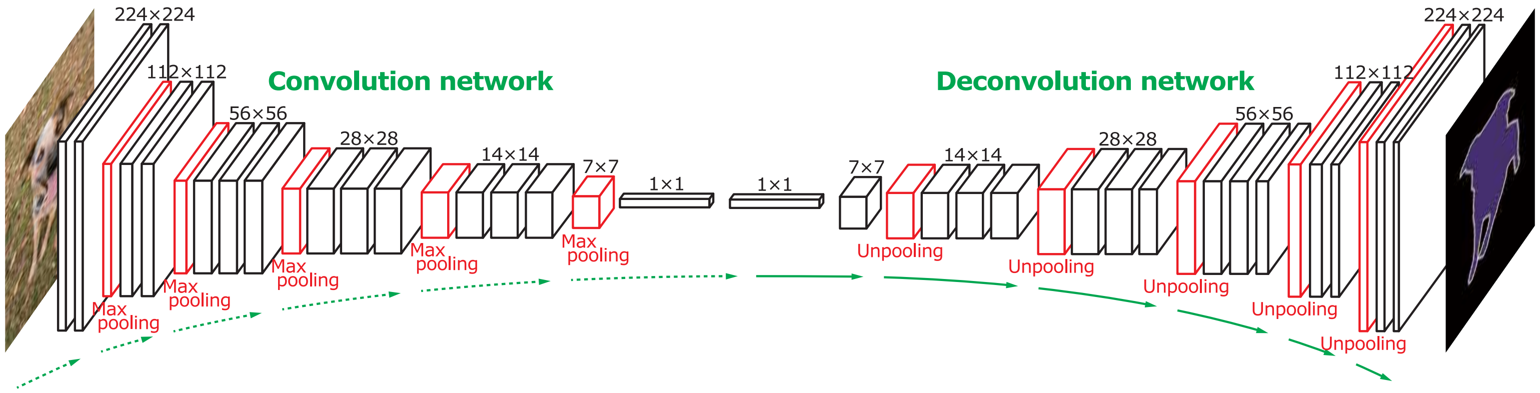
- Image to image

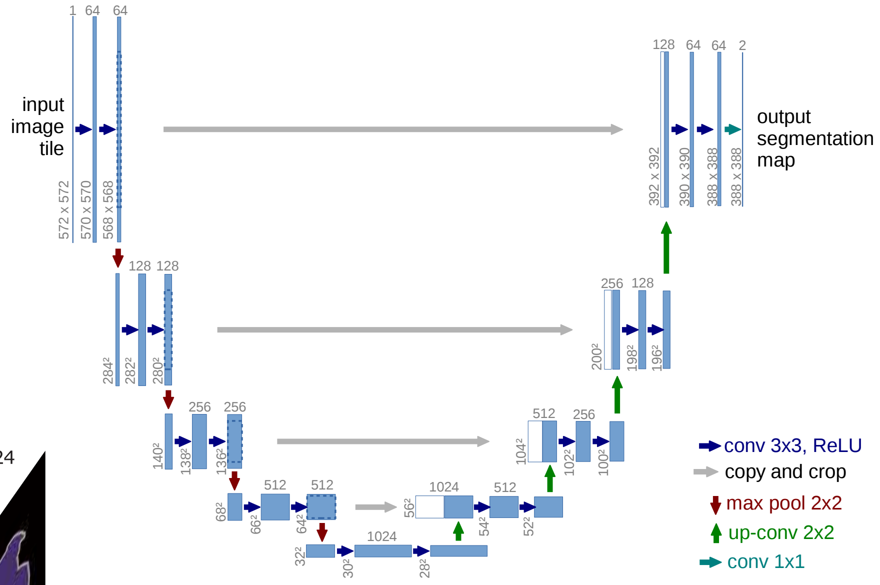


U-Architectures

- Several variants: FCN, U-arch, deconvolution, ...

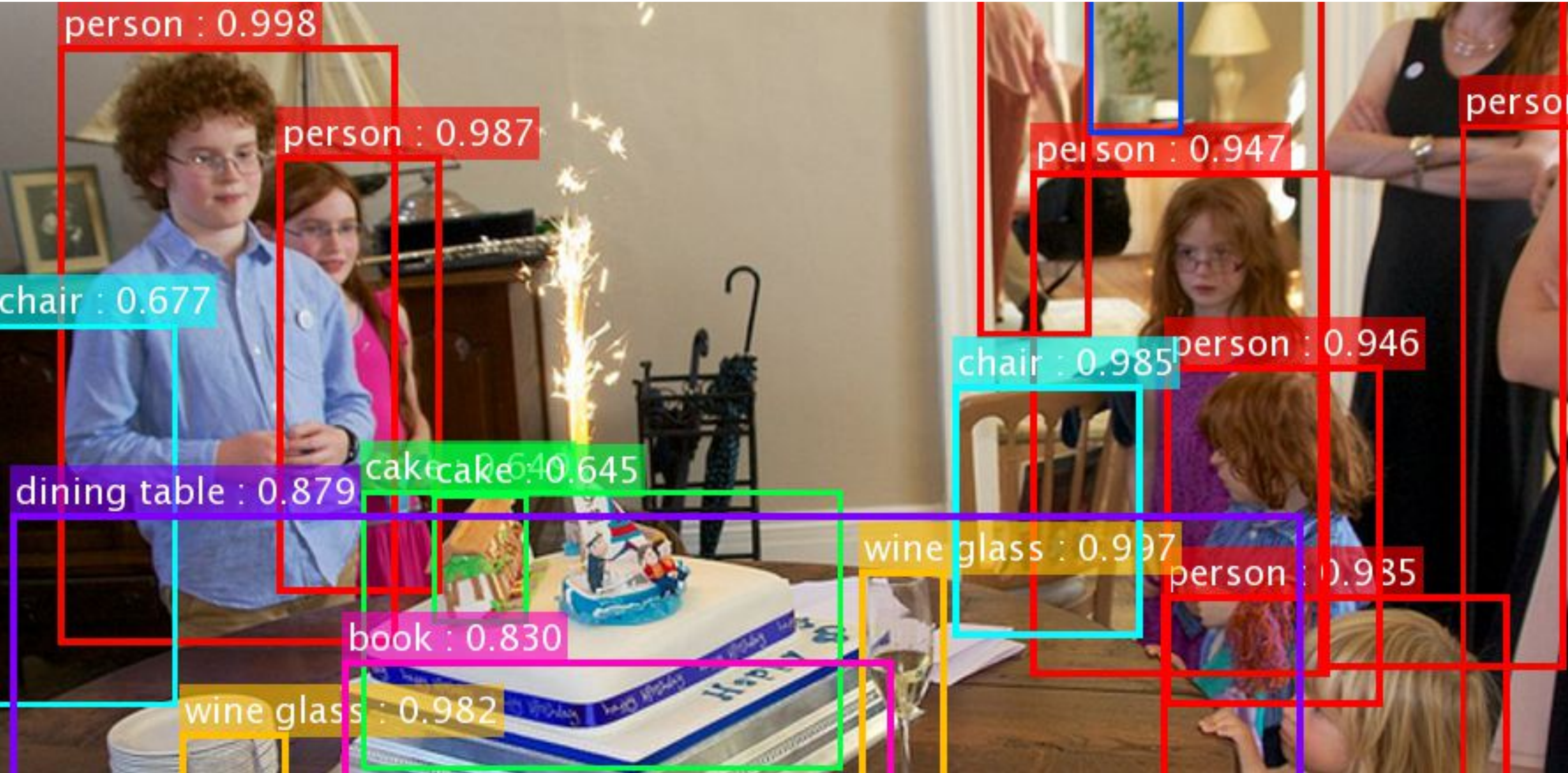






J. Long, E. Shelhamer, and T. Darrell. **Fully convolutional models for semantic segmentation.** In CVPR 2015
H. Noh, S. Hong, and B. Han. **Learning deconvolution network for semantic segmentation.** In ICCV 2015
O. Ronneberger, P. Fischer, and T. Brox. **U-net: Convolutional networks for biomedical image segmentation.** In MICCAI 2015

Object Detection



MS COCO Dataset Images

MS COCO Annotations

- 80 different categories

MS COCO

Dataset Images

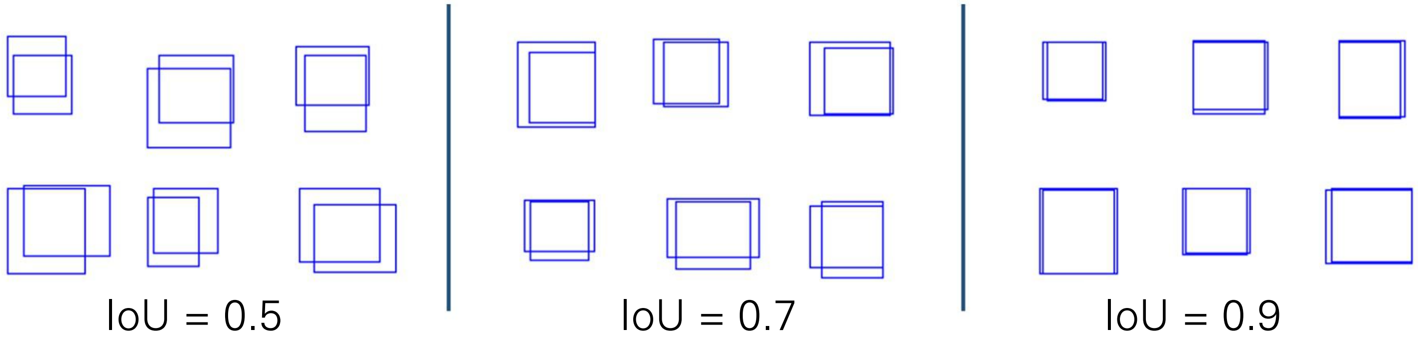
+

Annotations

COCO Object Detection Average Precision (%)

- Area under a detector's precision-recall curve, averaged over…
  - Object categories
  - True positive overlap requirement (IoU from 0.5 to 0.95; see below)

boxes



masks

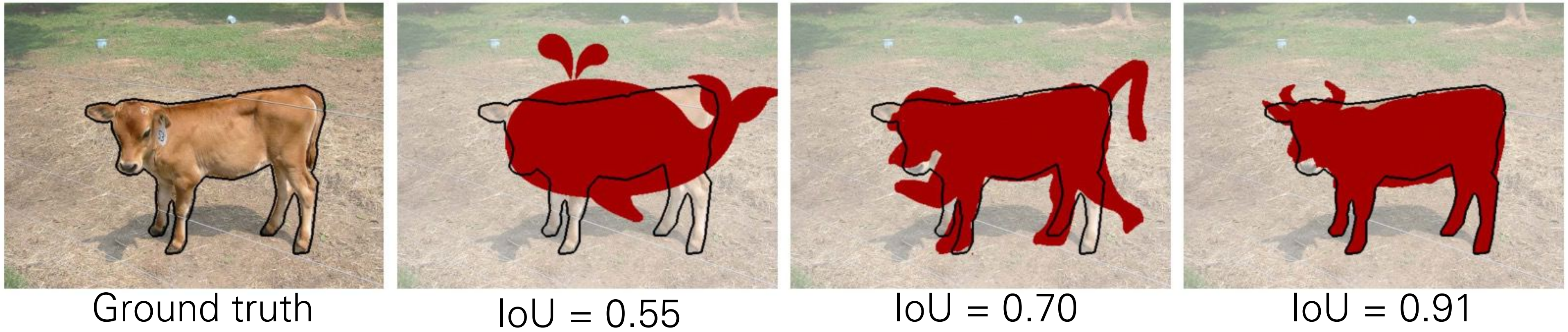
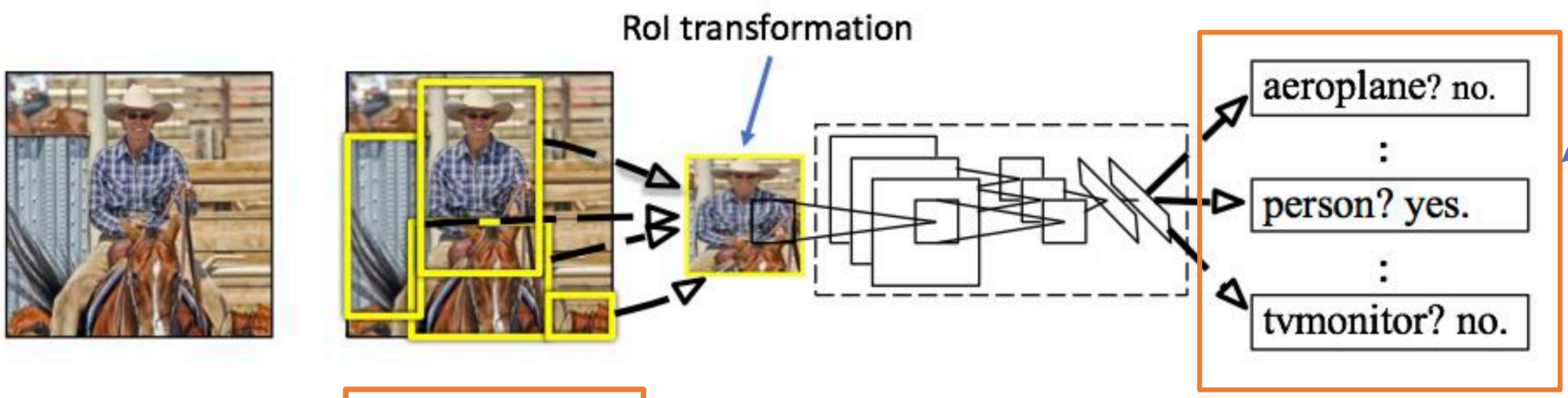


Figure credits: Dollár and Zitnick (top), Krähenbühl and Kulton (bottom)

More than one "stage" (≈proposal based; but doesn't require proposals)

classification of reduced output space elements



Cascade-like reduction in output space

Input image

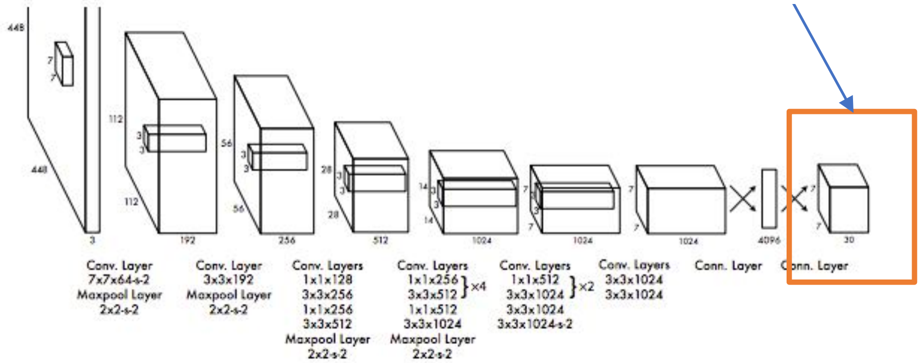
Object / region proposals

Deep Learning region classifier

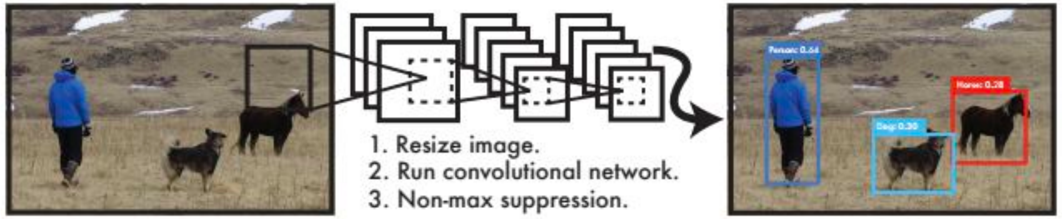
Region classification, box regression

One stage

Direct classification
Of all output space elements



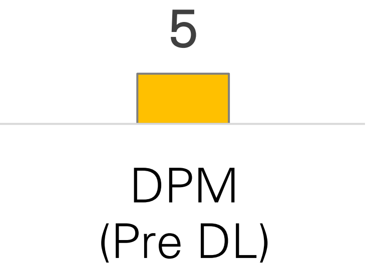
Redmond et al. You Only Look Once:
Unified Real-time Object Detection. In CVPR 2016



"You only look once»
"Single shot"

COCO Object Detection Average Precision (%)

Past
(best circa
2012)

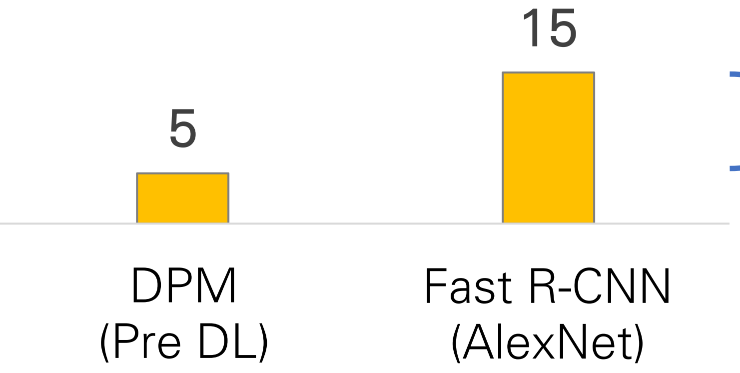


Felzenszwalb, Girshick, McAllester, Ramanan. Object Detection with Discriminatively Trained Part Based Models. PAMI 2010.

COCO Object Detection Average Precision (%)

Past
(best circa
2012)

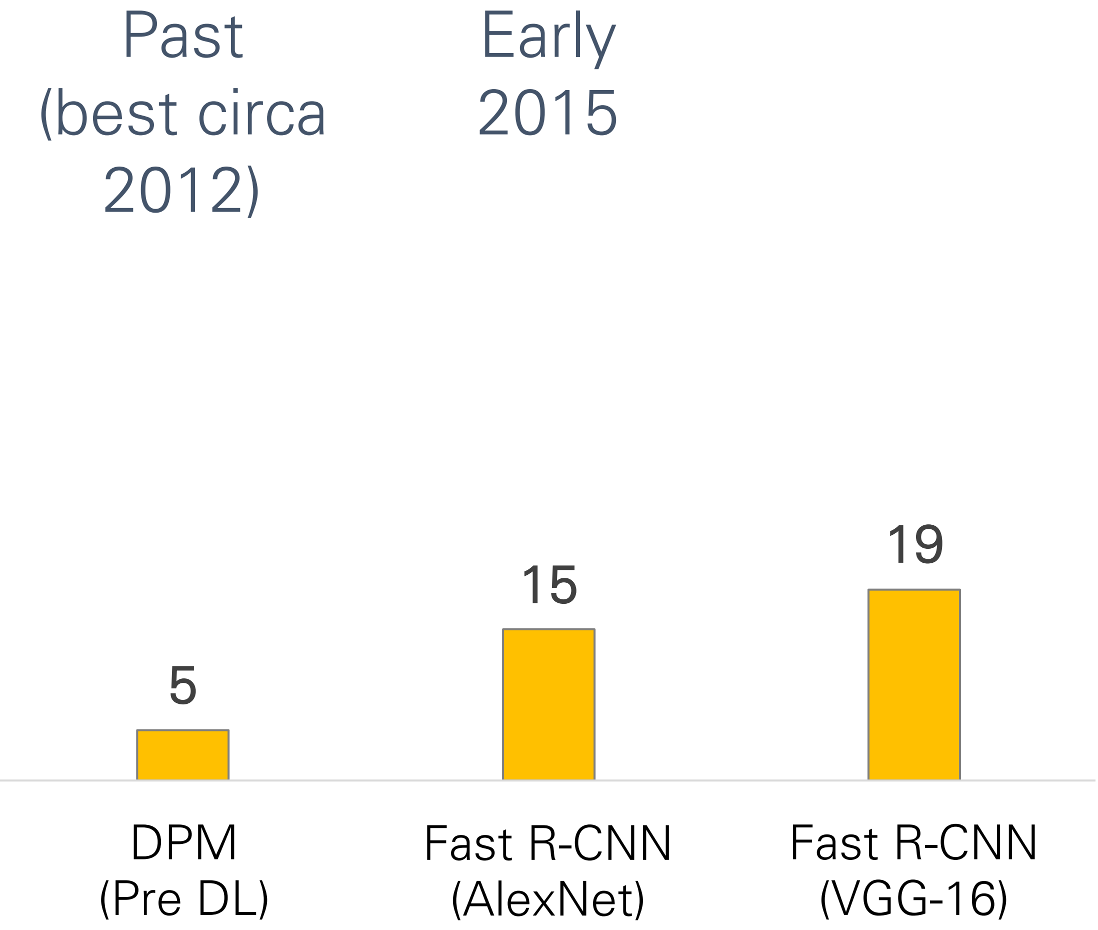
Early
2015



Movement to
DL methods

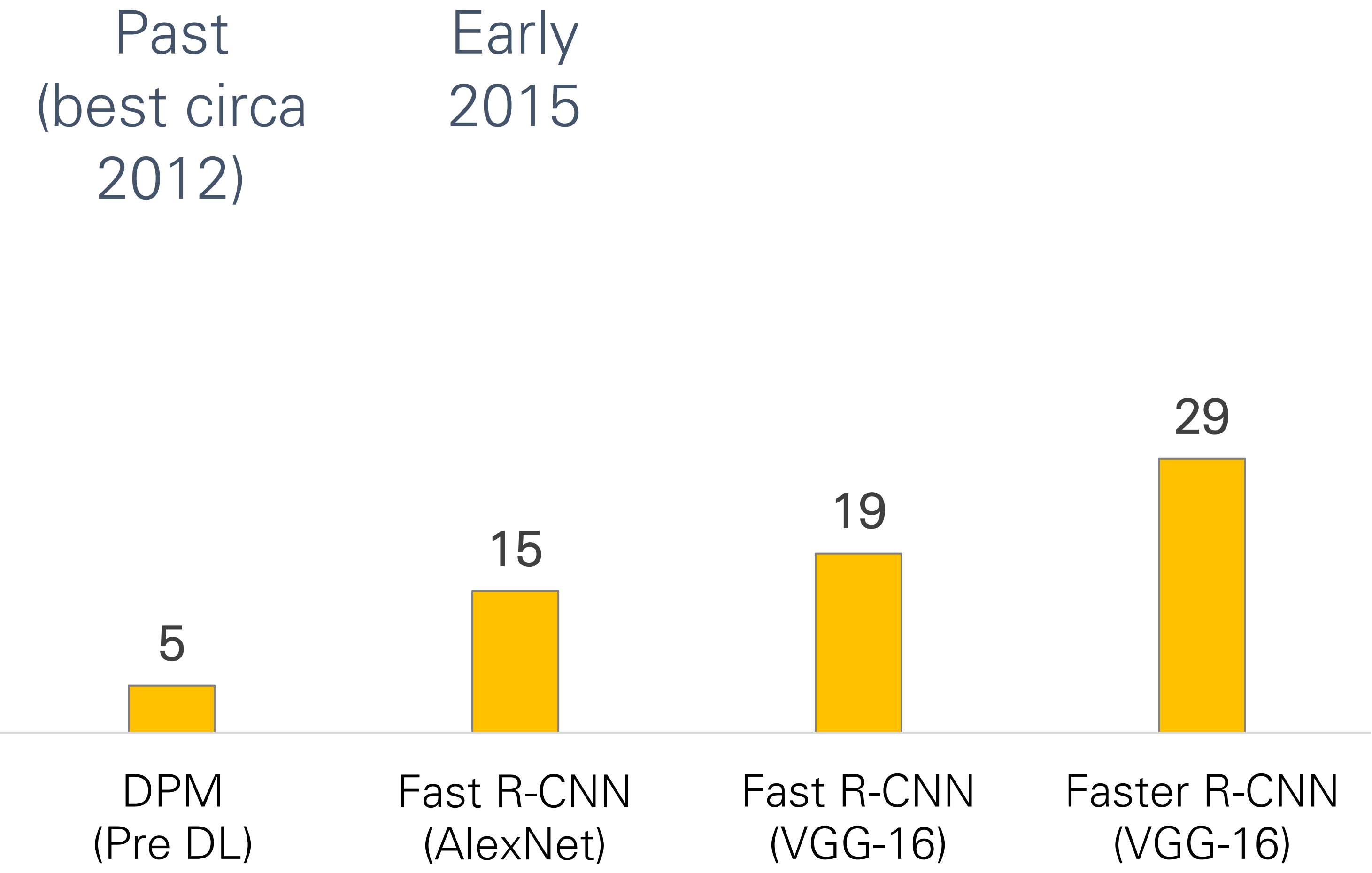
Girshick. Fast R-CNN. ICCV 2015.

COCO Object Detection Average Precision (%)



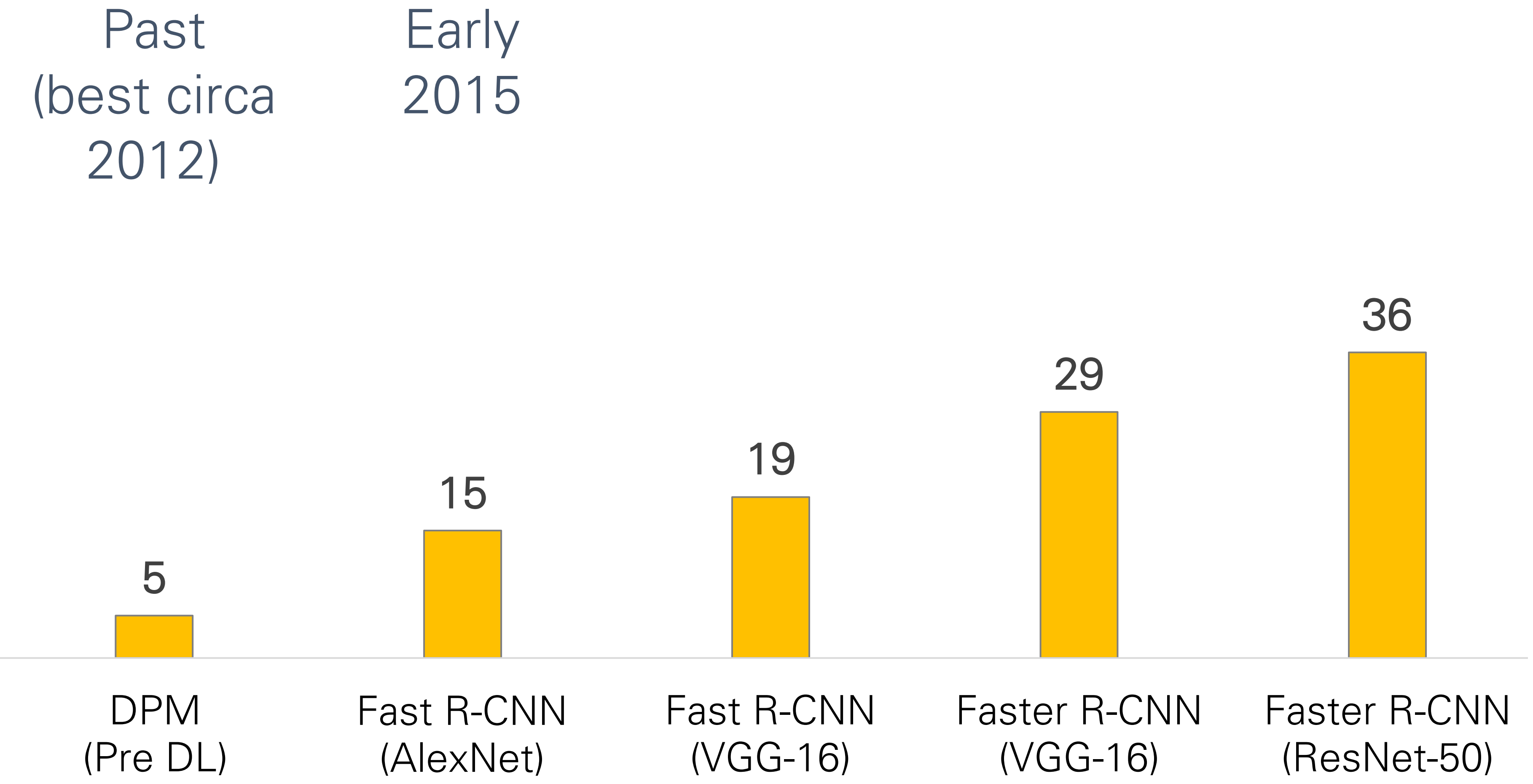
Girshick. Fast R-CNN. ICCV 2015.

COCO Object Detection Average Precision (%)



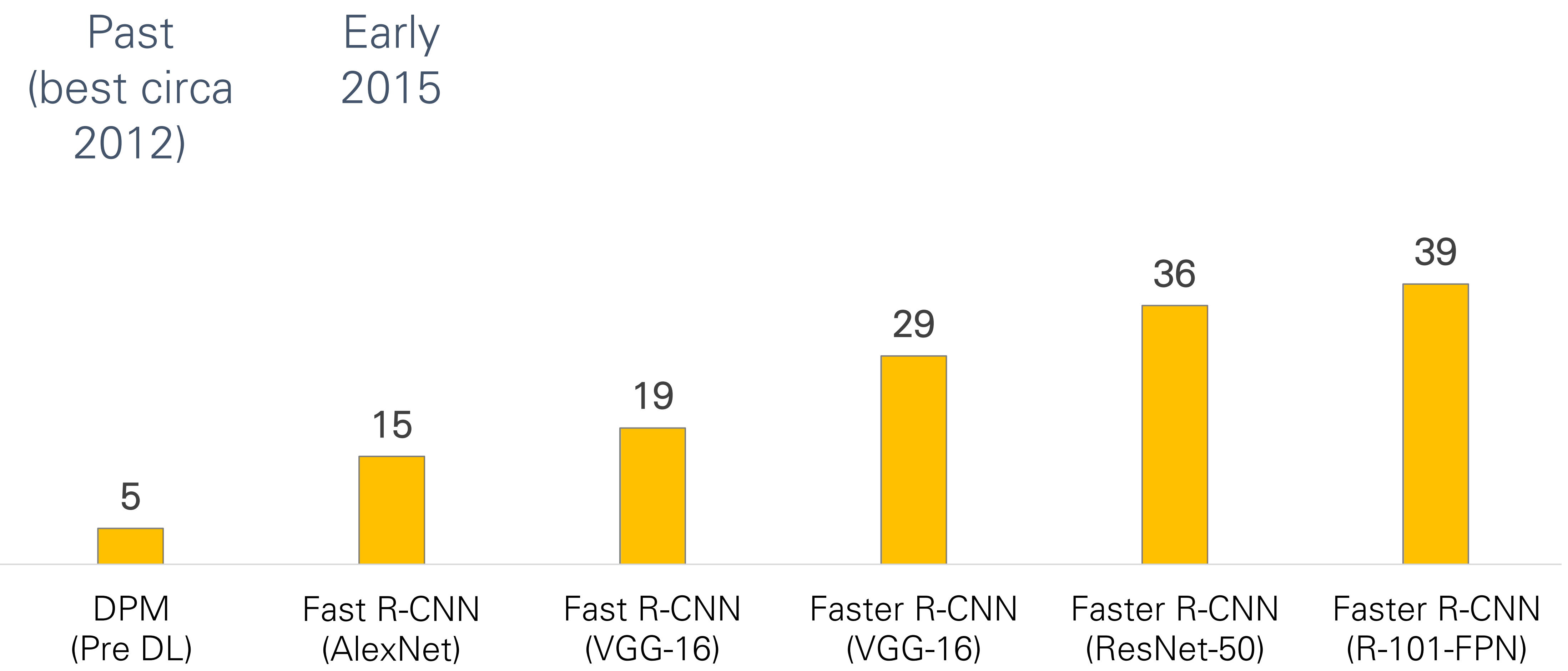
Ren, He, Girshick, Sun. Faster R-CNN: Towards Real-Time Object Detection. NIPS 2015.

COCO Object Detection Average Precision (%)



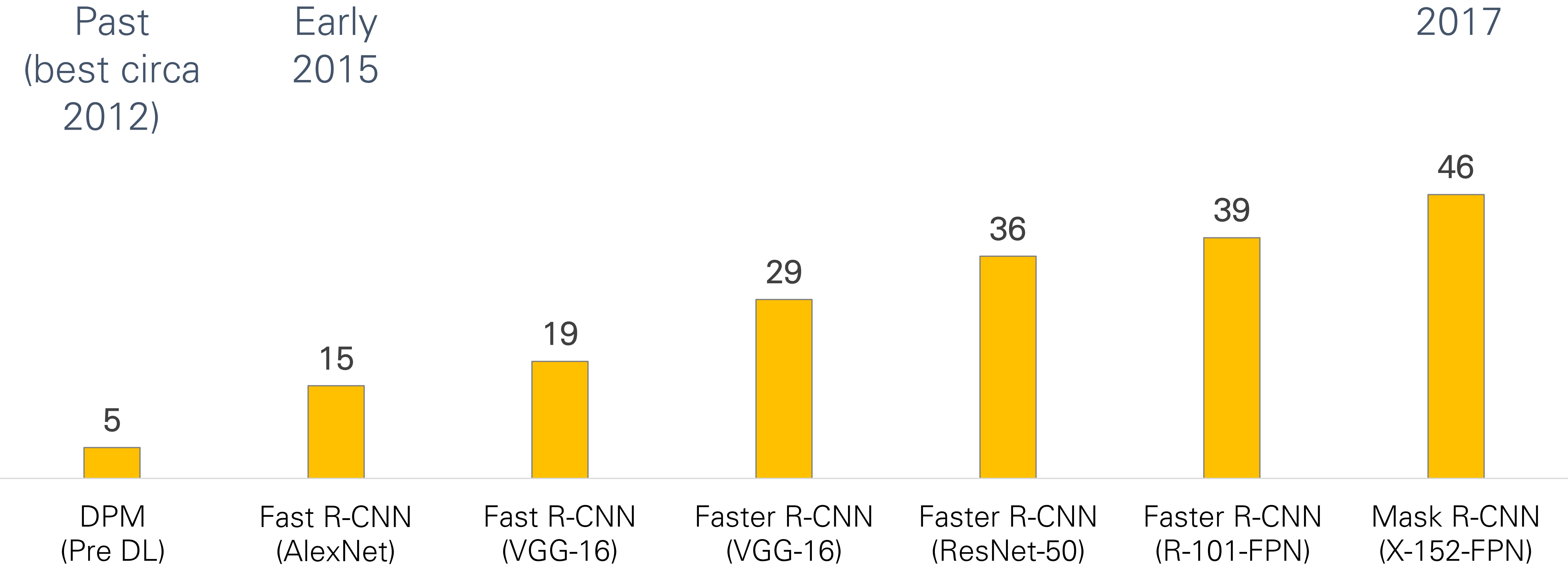
Ren, He, Girshick, Sun. Faster R-CNN: Towards Real-Time Object Detection. NIPS 2015.

COCO Object Detection Average Precision (%)



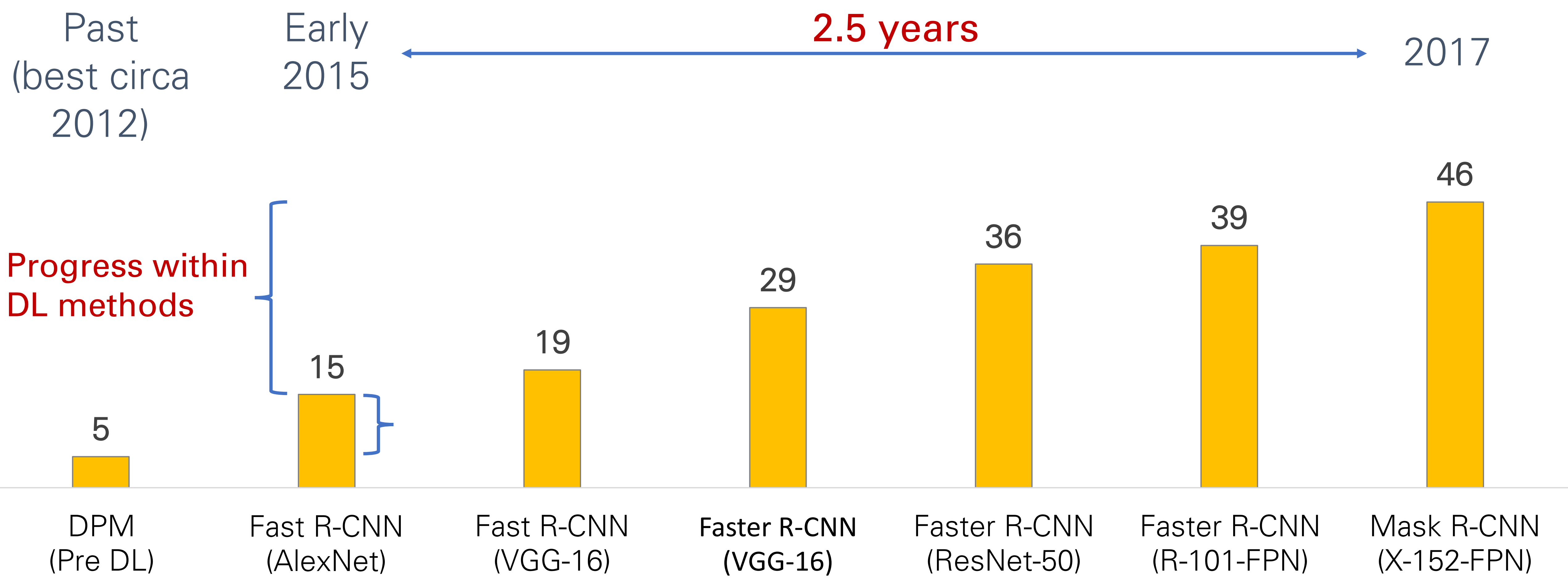
Lin et al. Feature Pyramid Networks. CVPR 2017.

COCO Object Detection Average Precision (%)



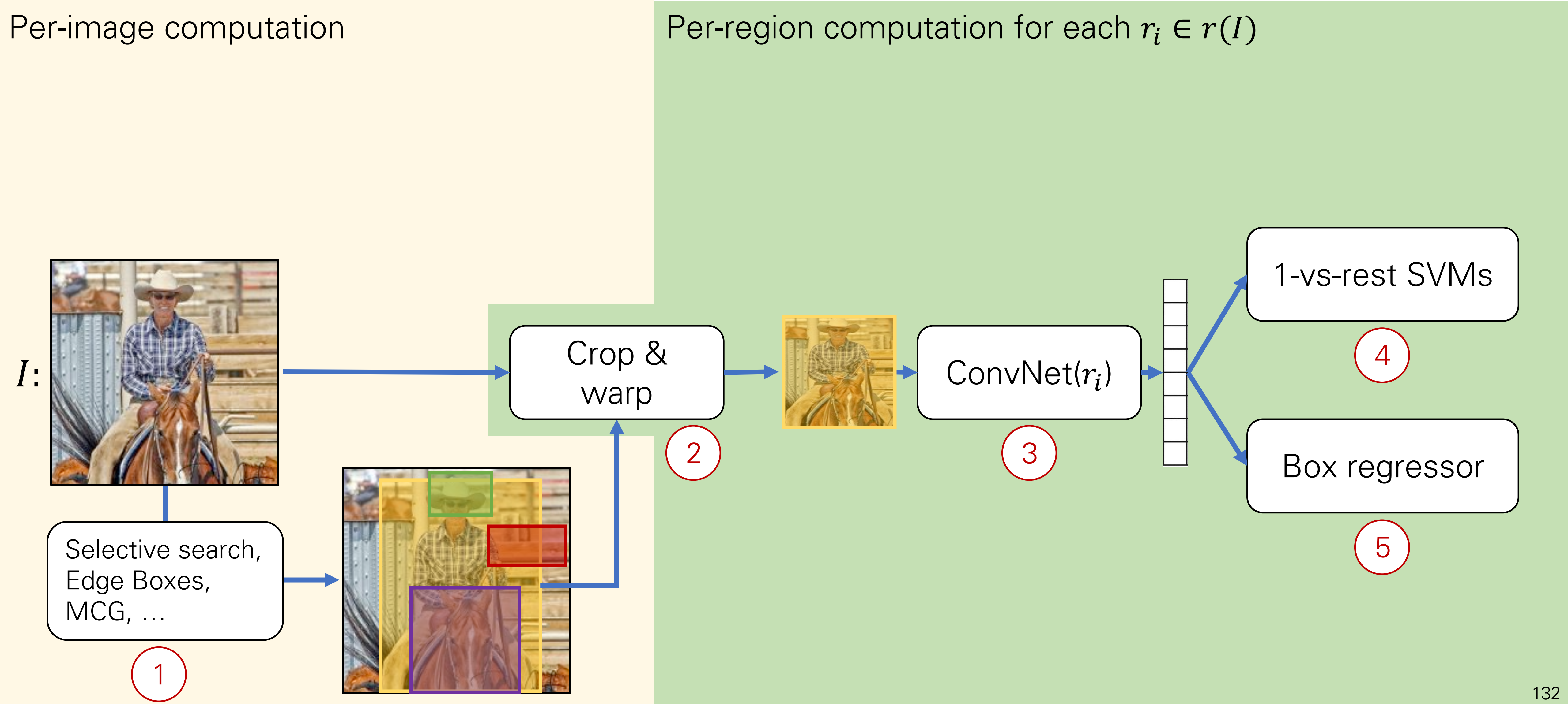
He, Gkioxari, Dollár, Girshick. Mask R-CNN. ICCV 2017.

COCO Object Detection Average Precision (%)

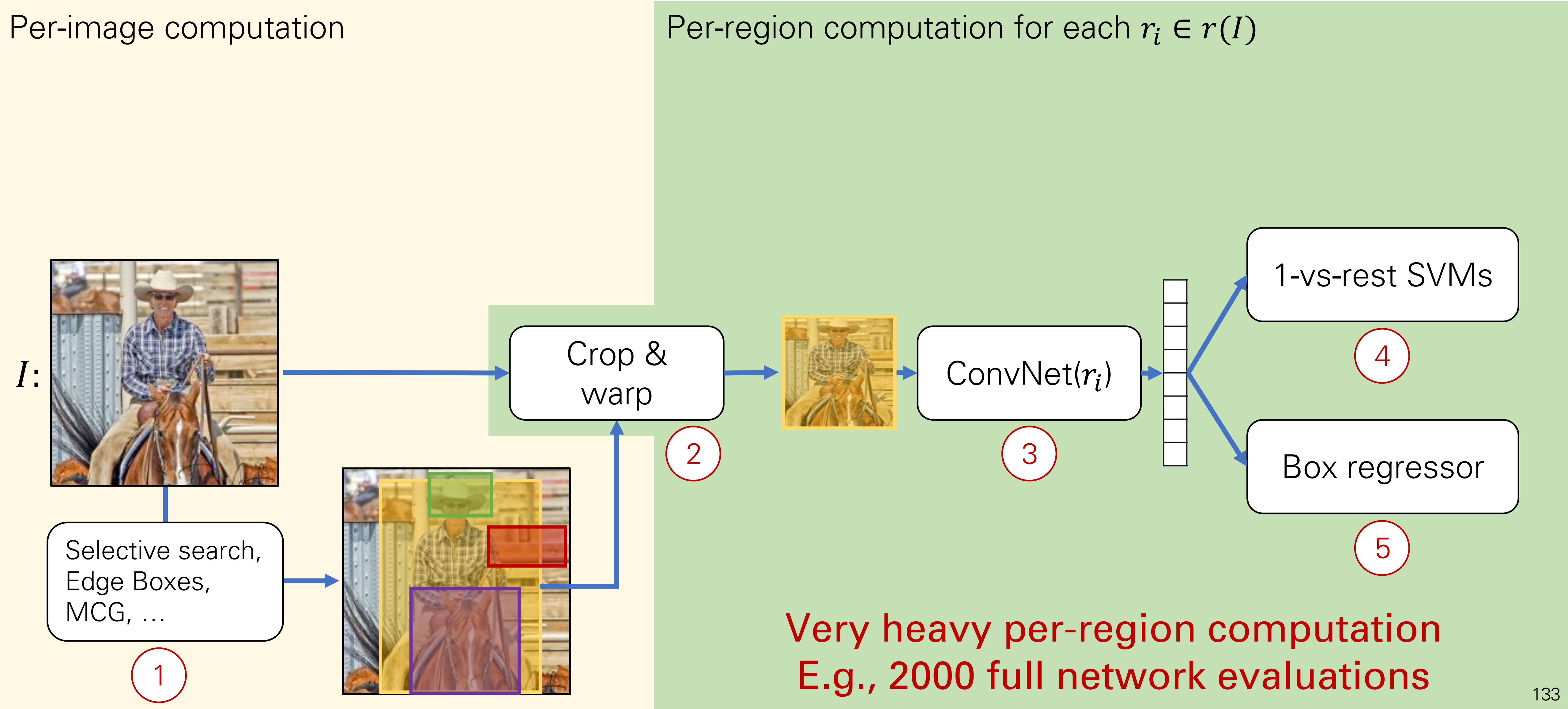


"Slow" R-CNN

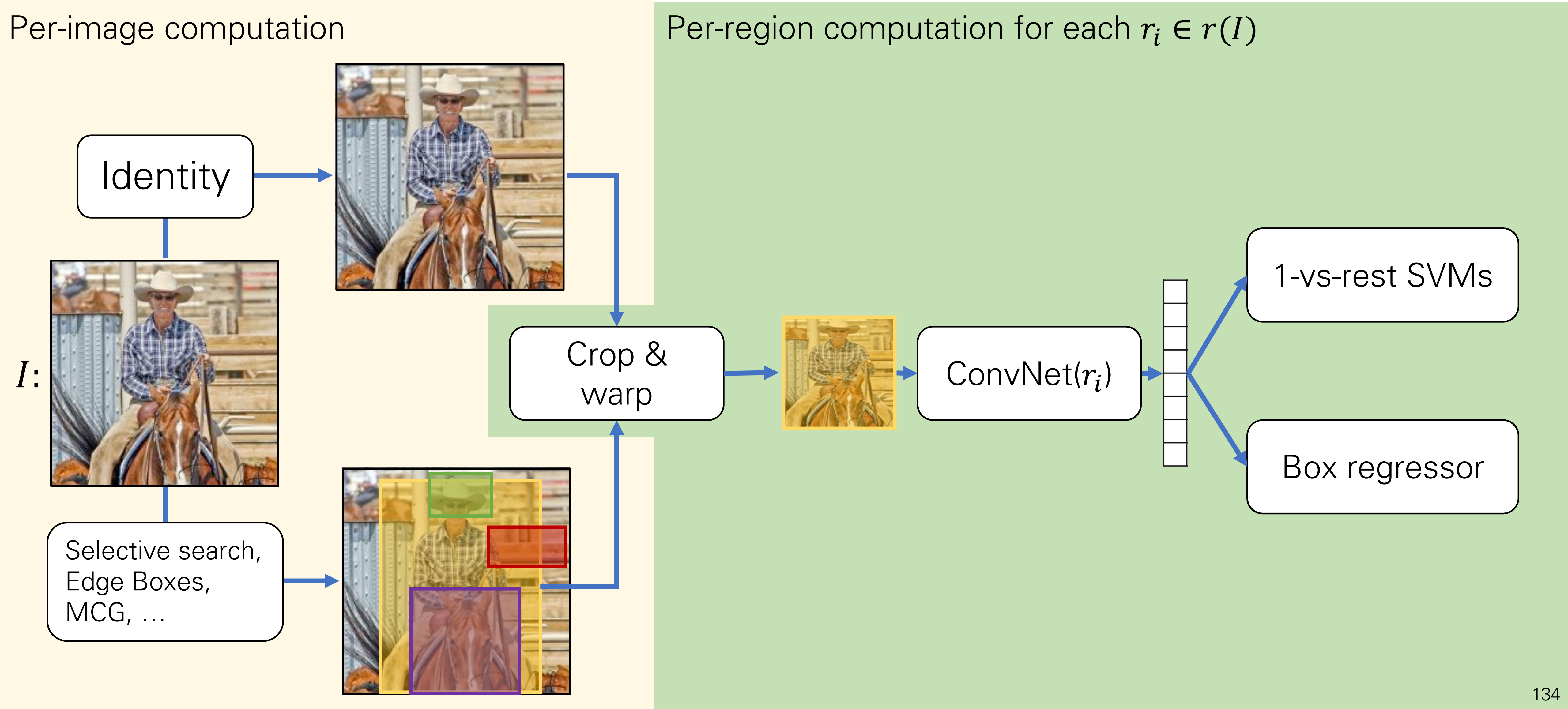
Girshick, Donahue, Darrell, Malik. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. CVPR 2014.



"Slow" R-CNN

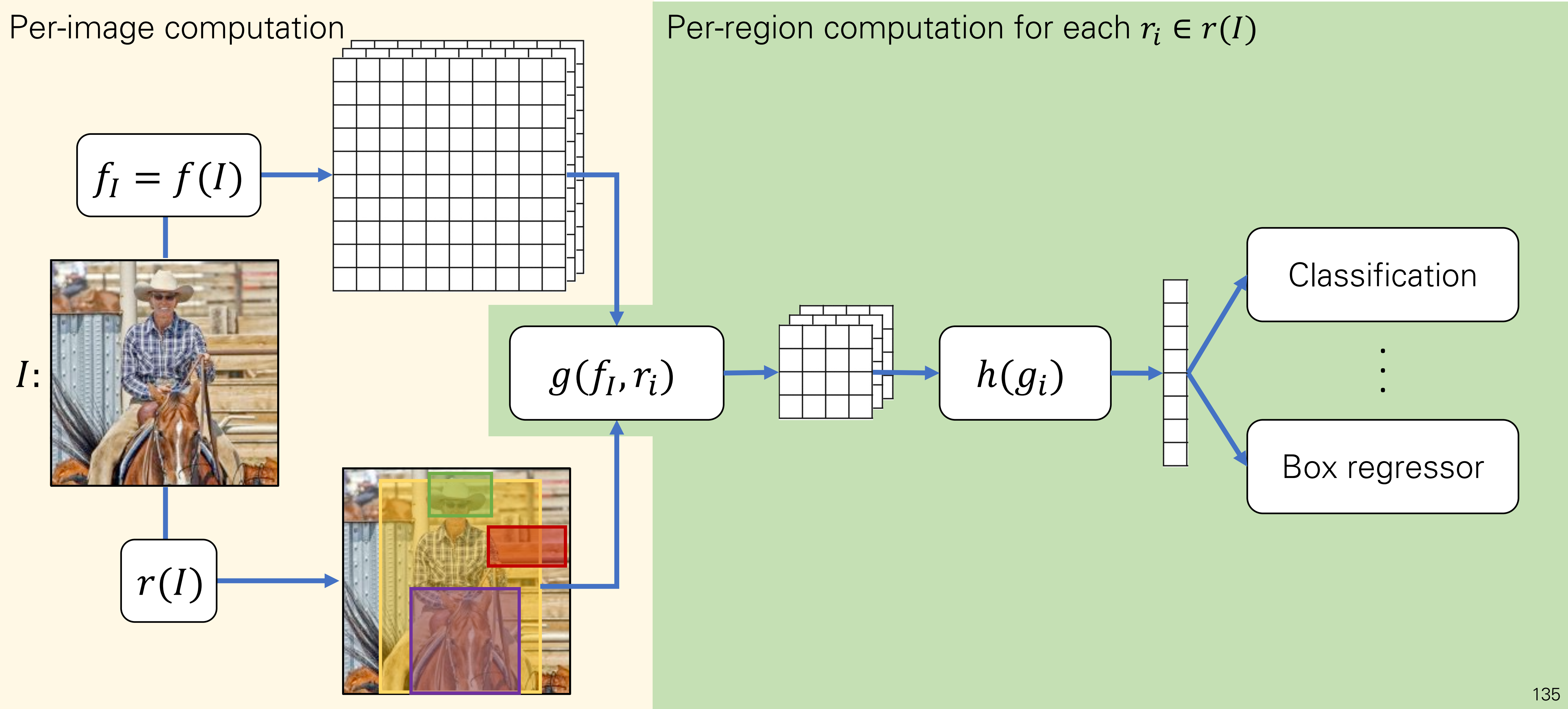

Per-image computation
Per-region computation for each $r_i \in r(I)$
I:
Selective search,
Edge Boxes,
MCG, …
1
Crop &
warp
2
ConvNet(r_i)
3
1-vs-rest SVMs
4
Box regressor
5
Very heavy per-region computation
E.g., 2000 full network evaluations

"Slow" R-CNN



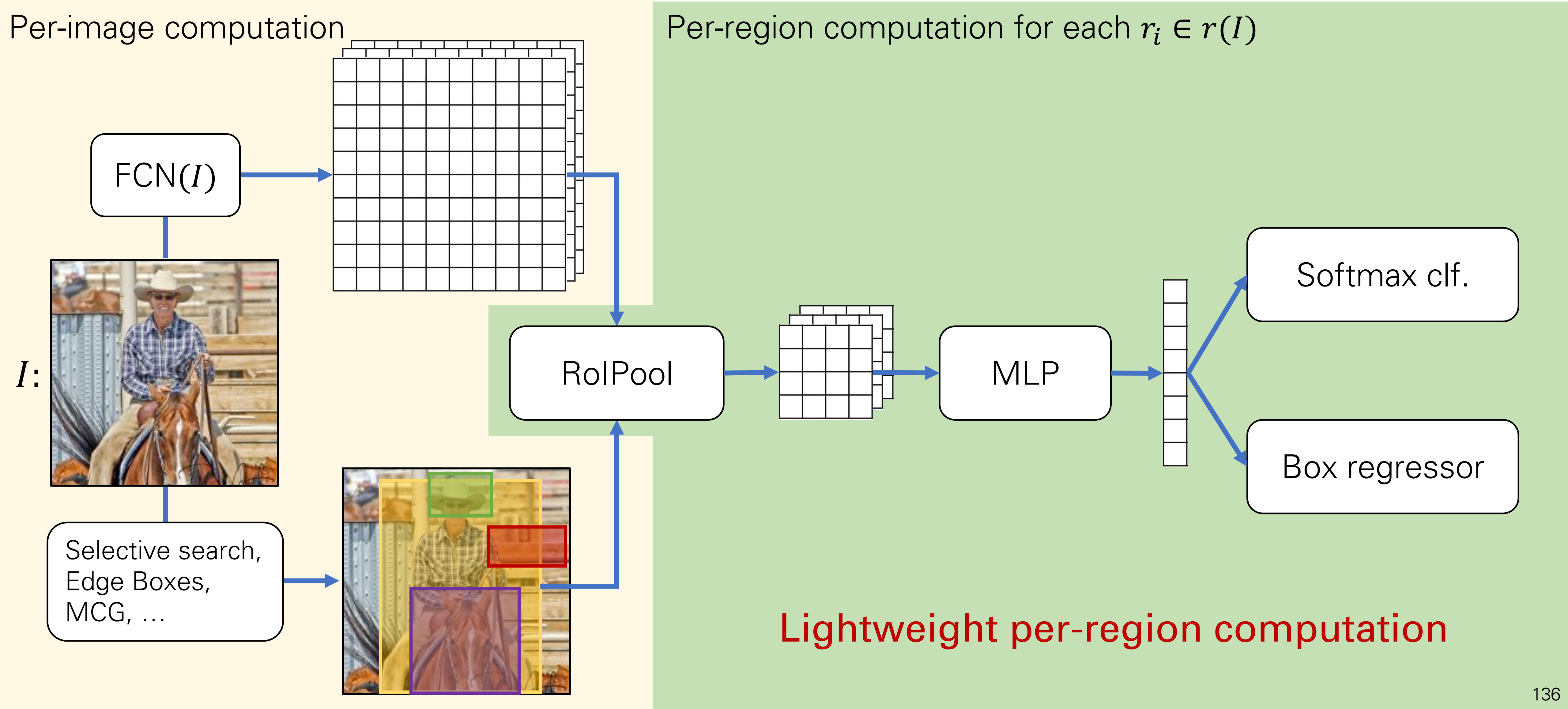
Per-image computation
Per-region computation for each $r_i \in r(I)$
Identity
I:
Selective search, Edge Boxes, MCG, …
Crop & warp
ConvNet(r_i)
1-vs-rest SVMs
Box regressor

Generalized R-CNN Approach to Detection

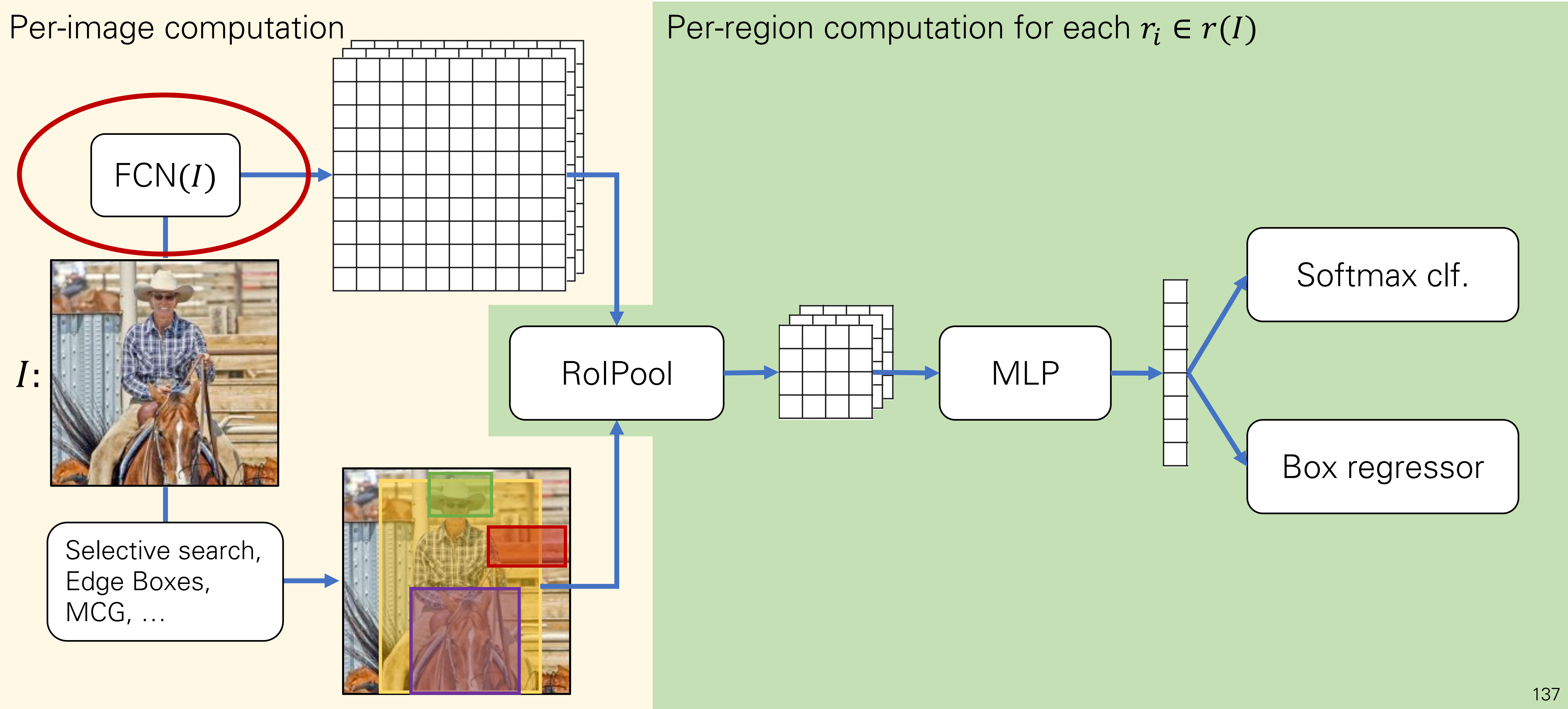


Girshick. Fast R-CNN. ICCV 2015.

Fast R-CNN



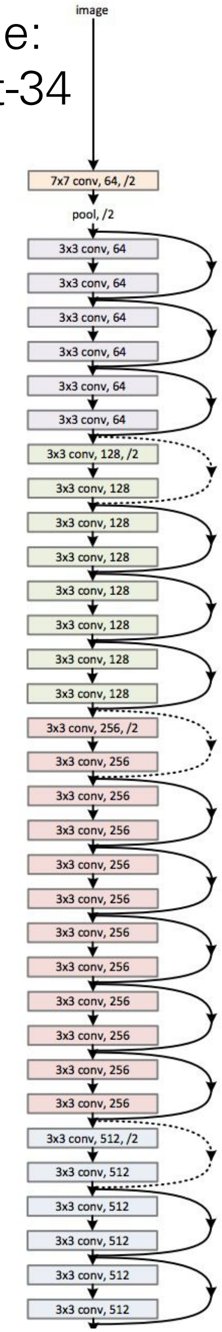
Fast R-CNN

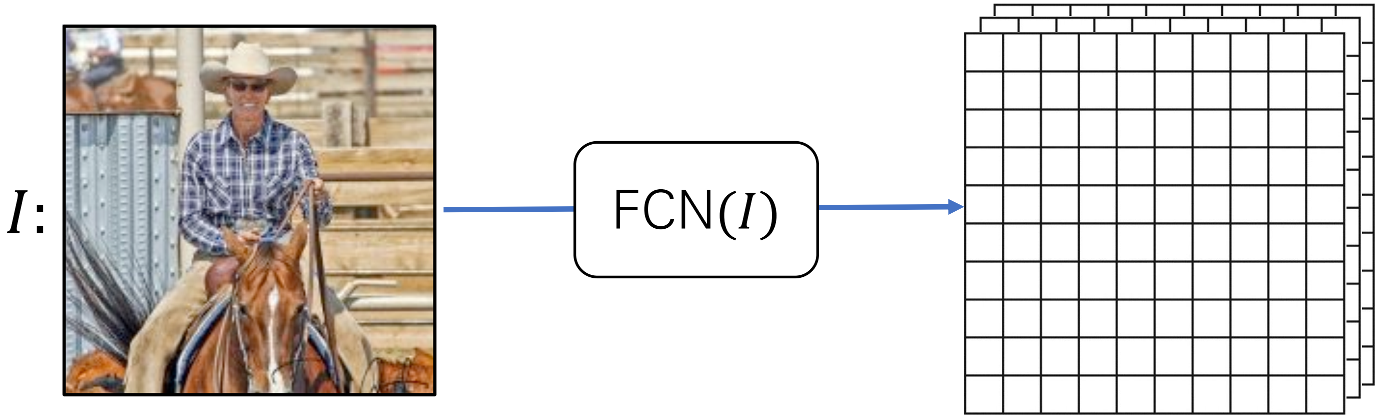

Per-image computation
Per-region computation for each $r_i \in r(I)$
FCN(I)
I:
Selective search,
Edge Boxes,
MCG, …
RoIPool
MLP
Softmax clf.
Box regressor

Whole-image FCN

- Use **any standard ConvNet** as the "**backbone architecture**"
  - AlexNet, VGG, ResNet, Inception, Inception-ResNet, ResNeXt, DenseNet, …
  - Use the first N layers with spatial extent (e.g., up to "conv5")

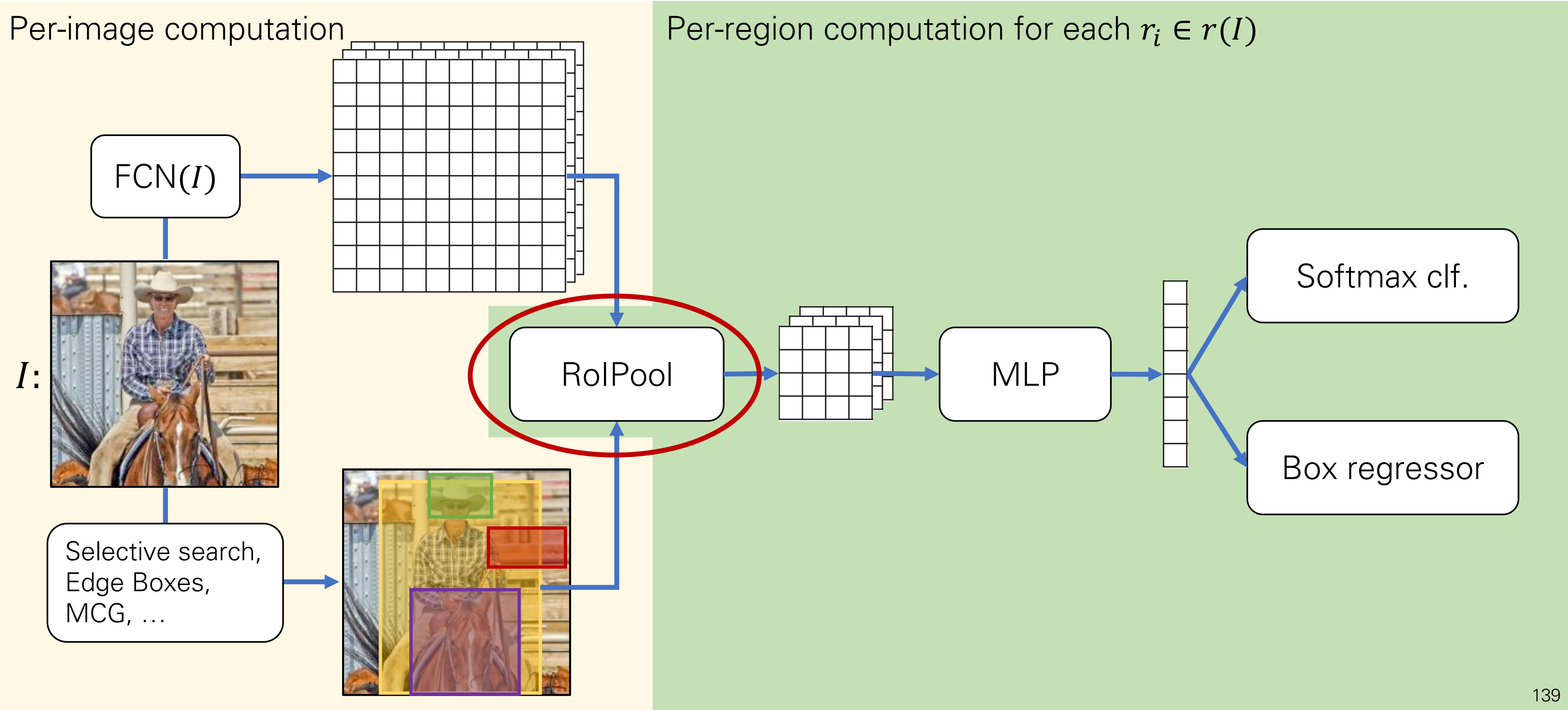
Example: ResNet-34





Example feature map dimensions: (512, H/16, W/16)

Fast R-CNN

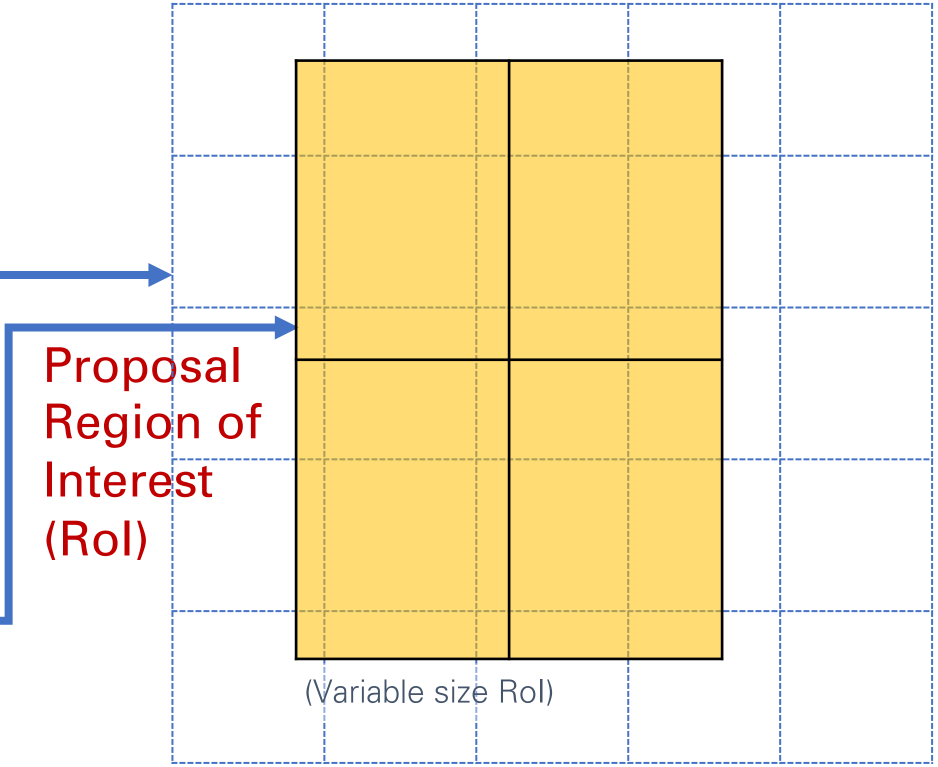


Per-image computation
Per-region computation for each $r_i \in r(I)$
FCN(I)
I:
Selective search,
Edge Boxes,
MCG, …
RoIPool
MLP
Softmax clf.
Box regressor

RoIPool (on each Proposal)

Transform **arbitrary size proposal** into a **fixed-dimensional** representation (e.g., 2x2)

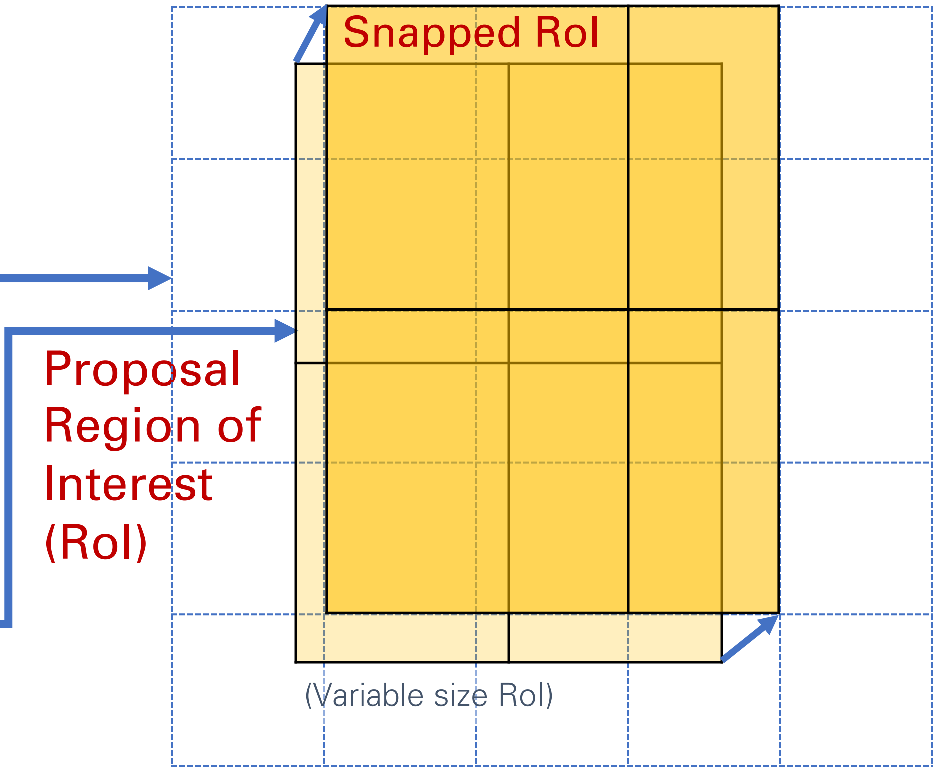
f_I = FCN(I)



RoIPool (on each Proposal)

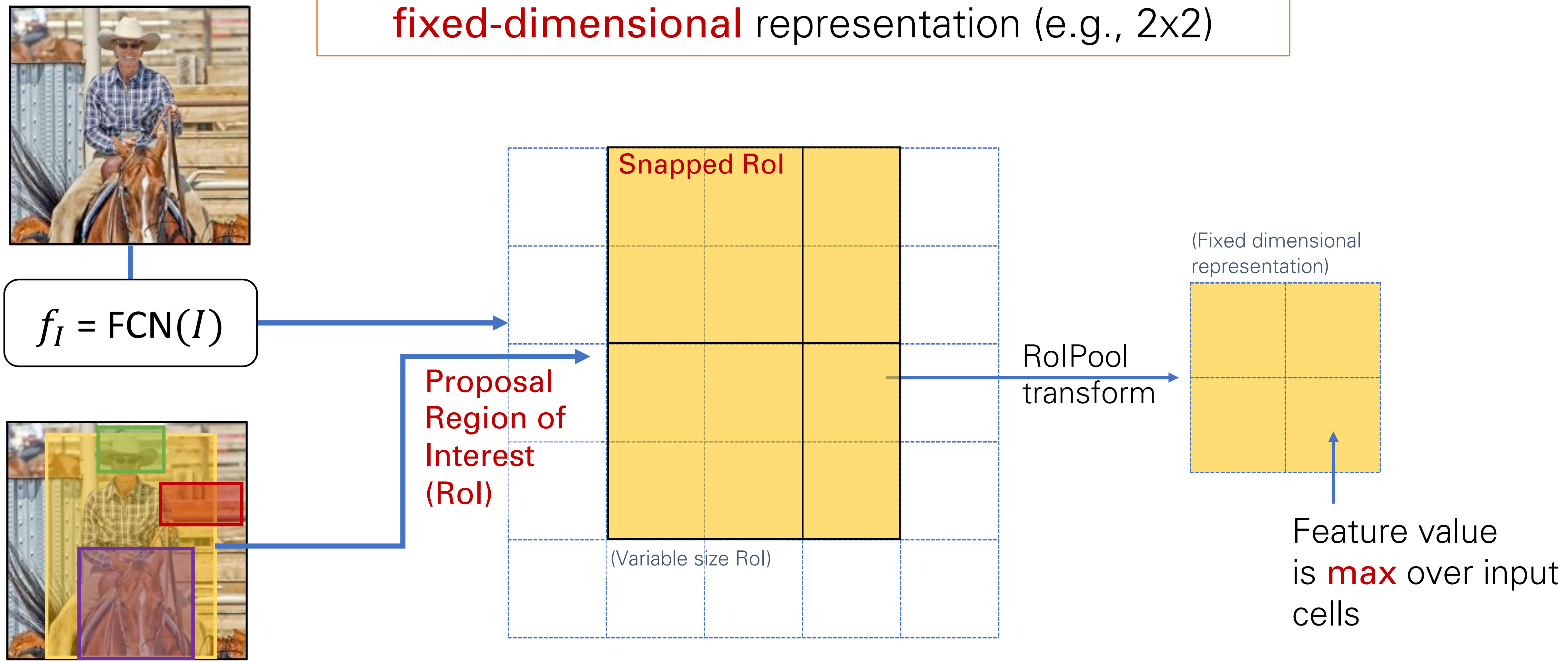
Transform **arbitrary size proposal** into a **fixed-dimensional** representation (e.g., 2x2)

f_I = FCN(I)

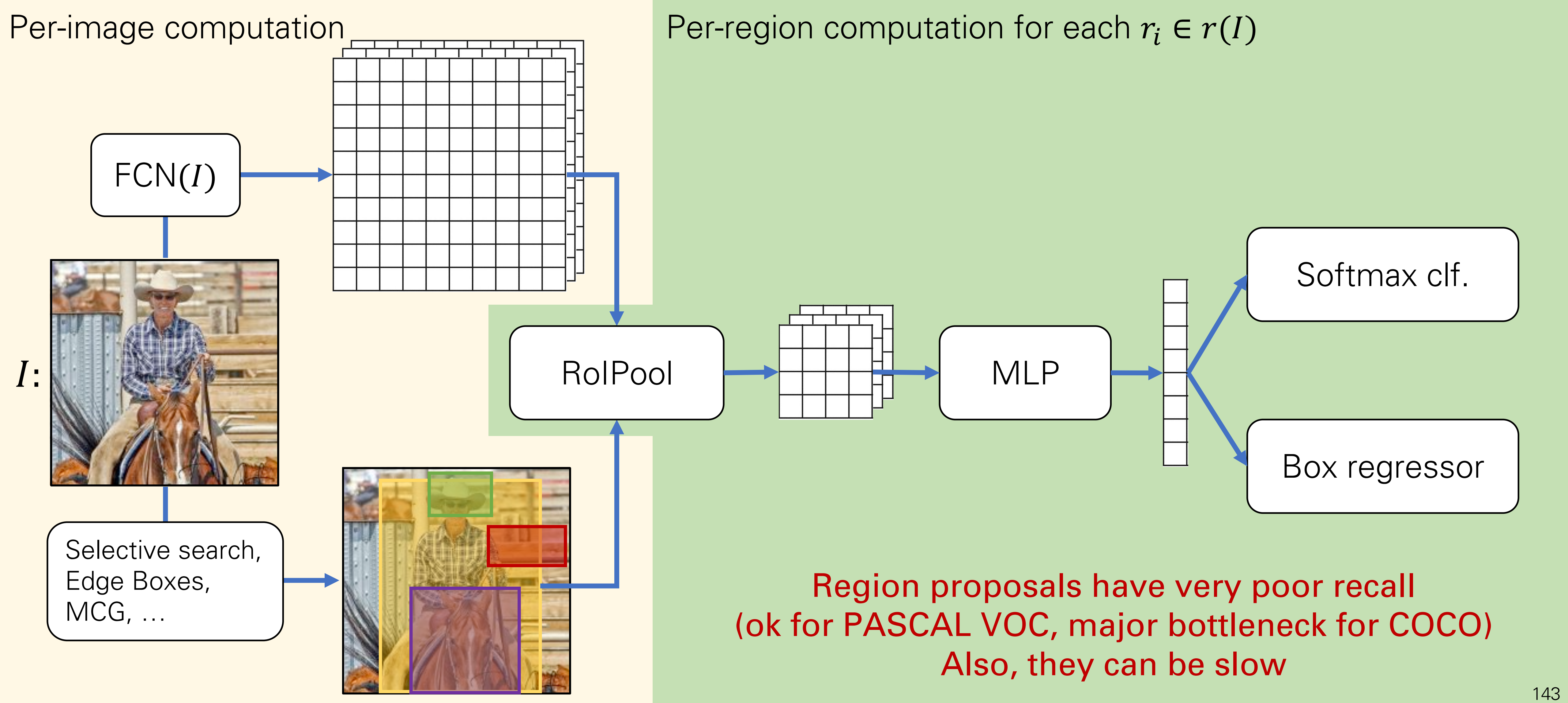


RoIPool (on each Proposal)

Transform **arbitrary size proposal** into a **fixed-dimensional** representation (e.g., 2x2)



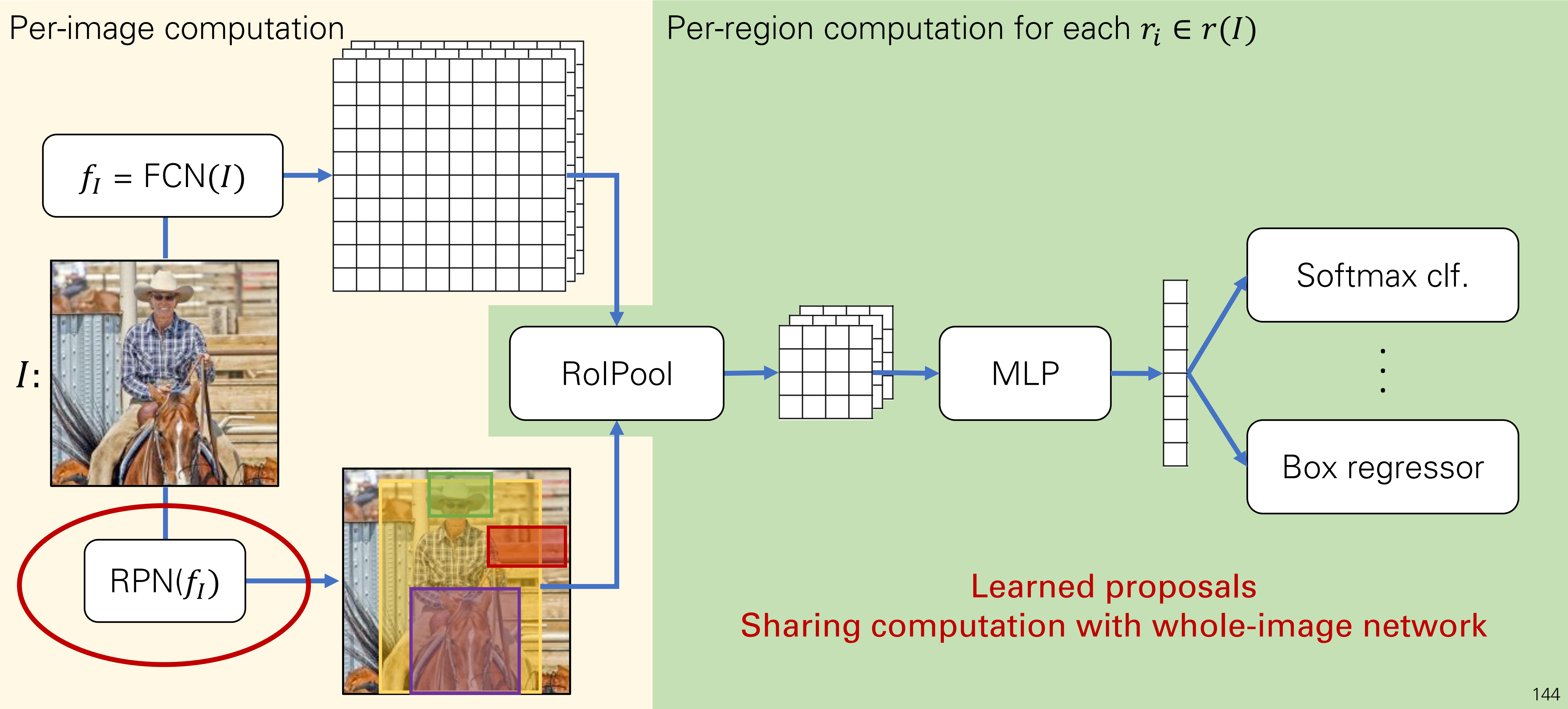
Fast R-CNN



Region proposals have very poor recall
(ok for PASCAL VOC, major bottleneck for COCO)
Also, they can be slow

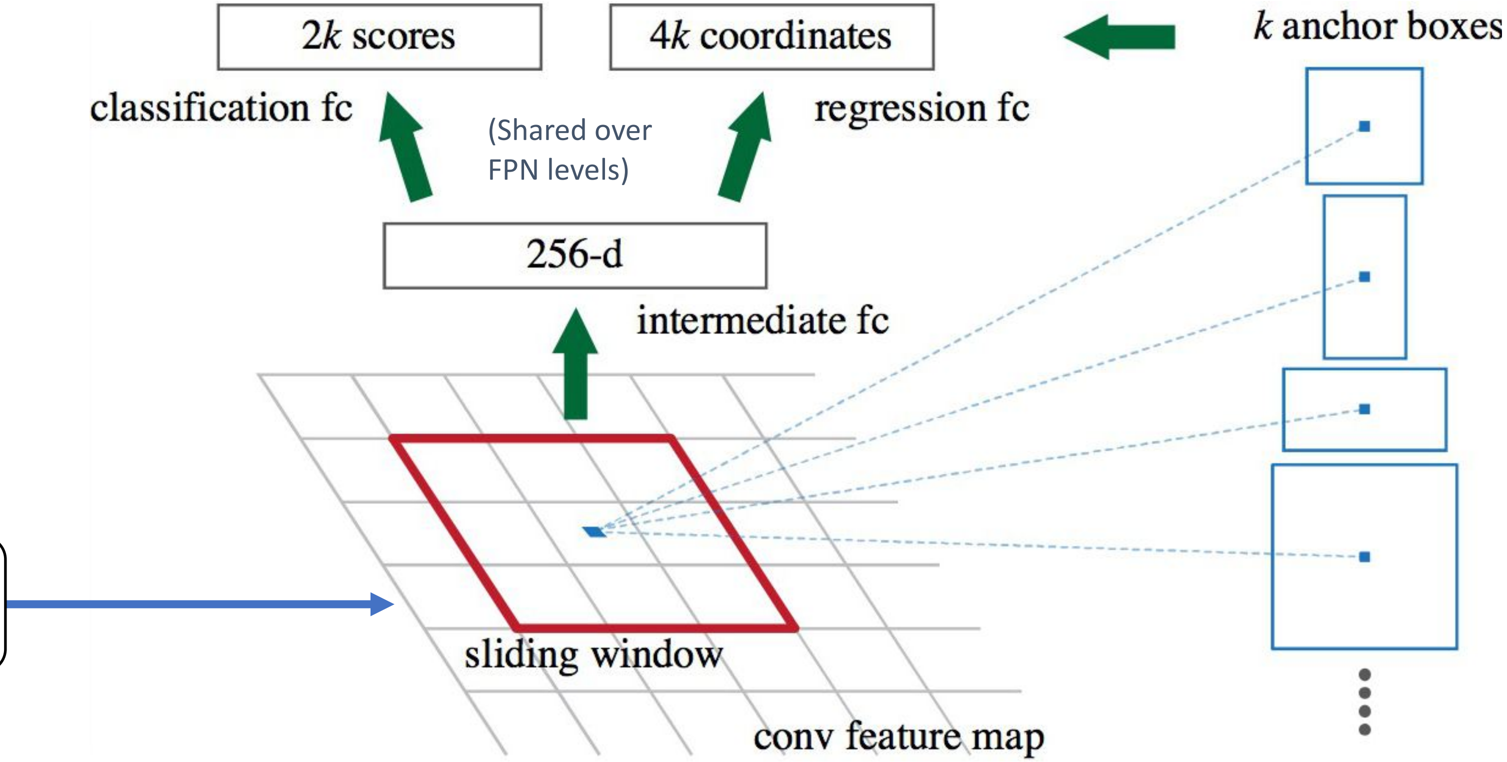
Faster R-CNN

Ren, He, Girshick, Sun. Faster R-CNN: Towards Real-Time Object Detection. NIPS 2015.



Region Proposal Network (RPN)

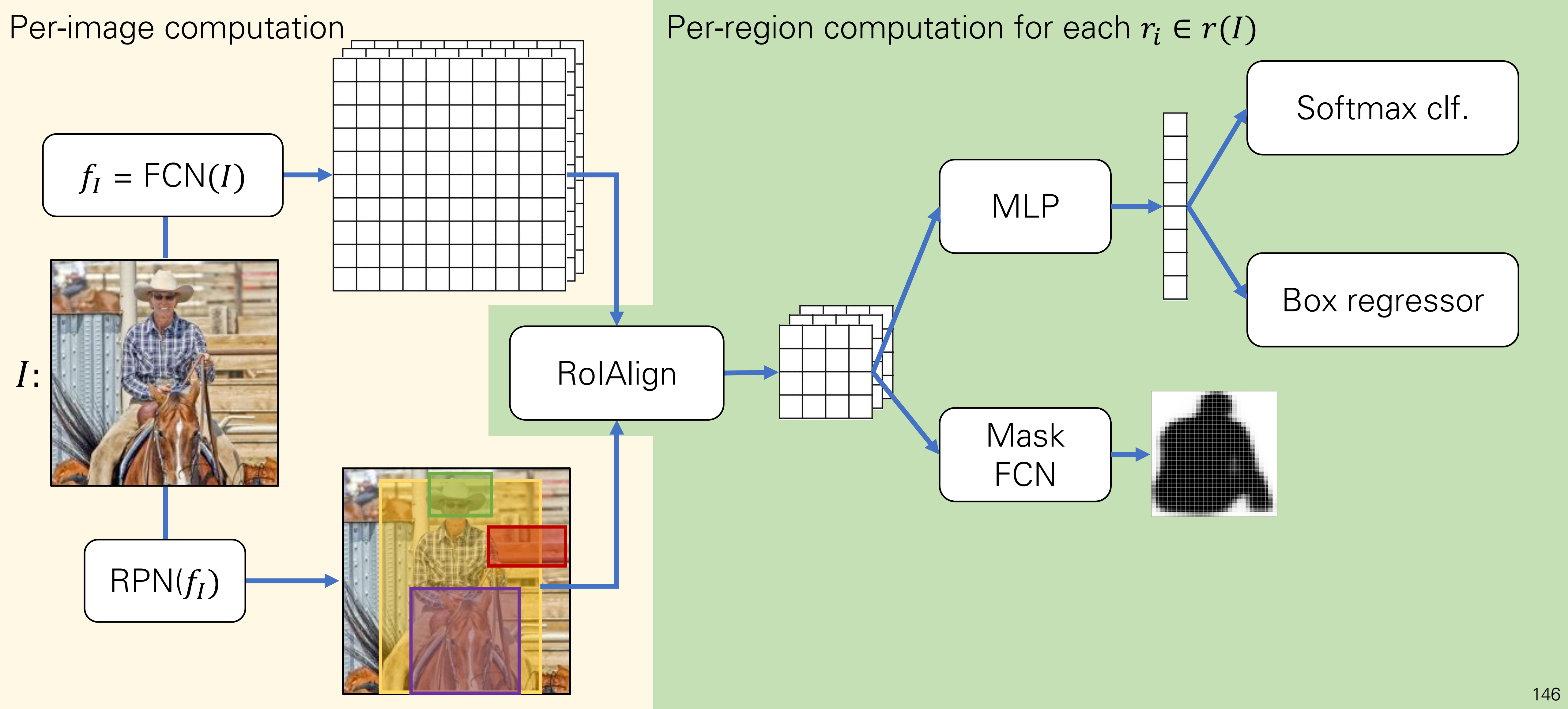
Proposals = sliding window object/not-object classifier + box regression
inside the same network



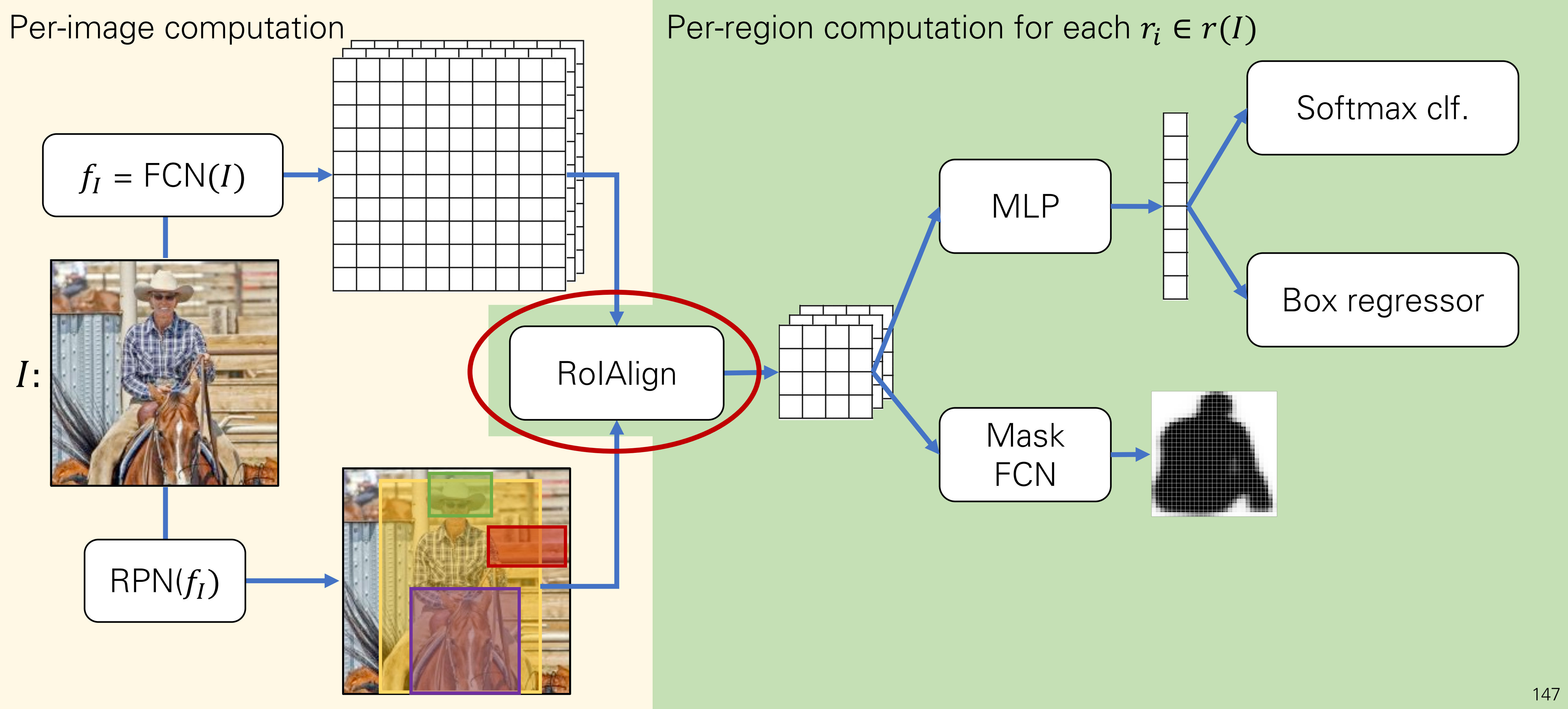
Anchors are prototypical object boxes

Mask R-CNN

He, Gkioxari, Dollár, Girshick. Mask R-CNN. ICCV 2017.

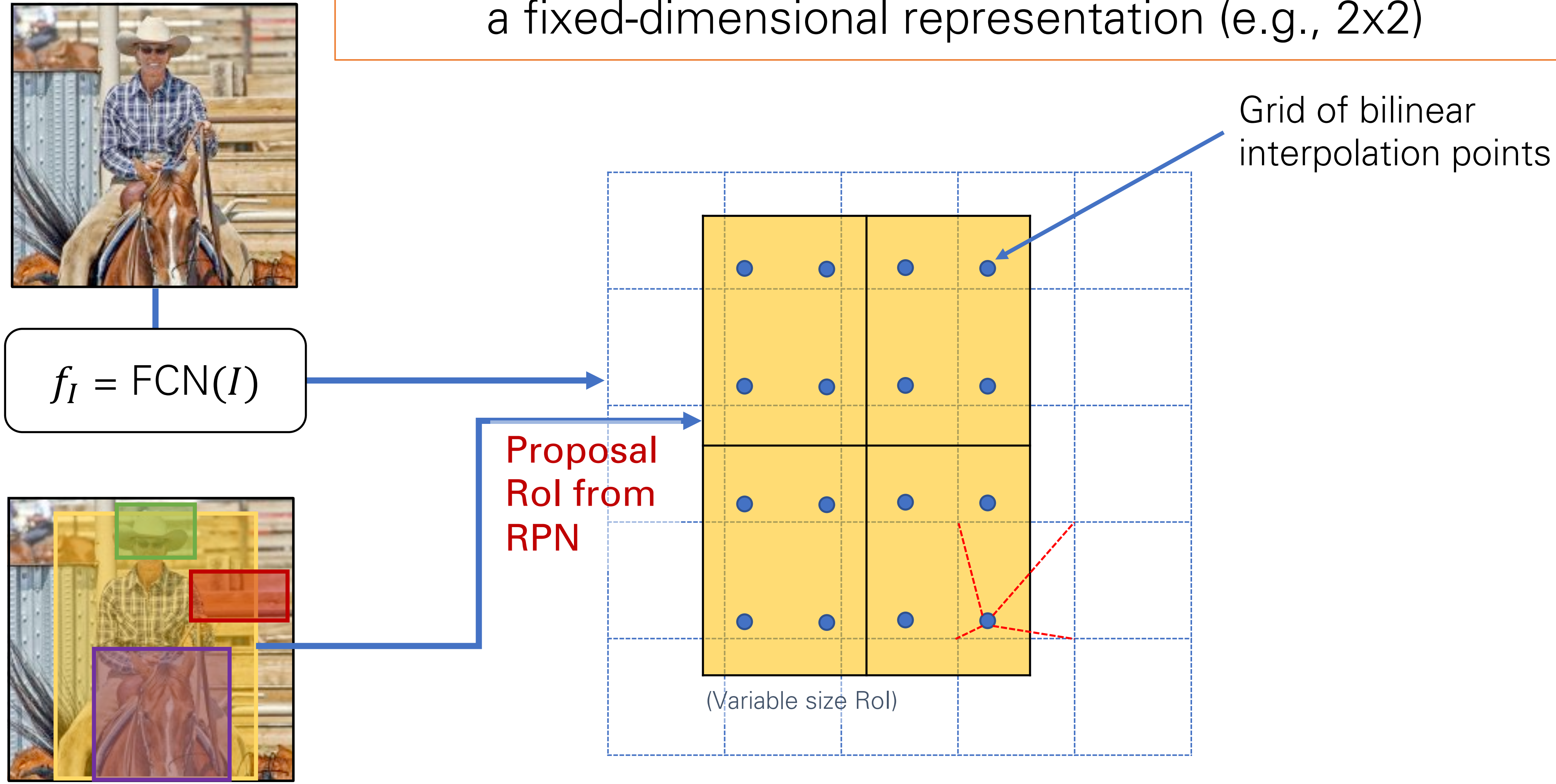


Mask R-CNN


Per-image computation
Per-region computation for each $r_i \in r(I)$
f_I = FCN(I)
I:
RPN(f_I)
RoIAlign
MLP
Softmax clf.
Box regressor
Mask FCN

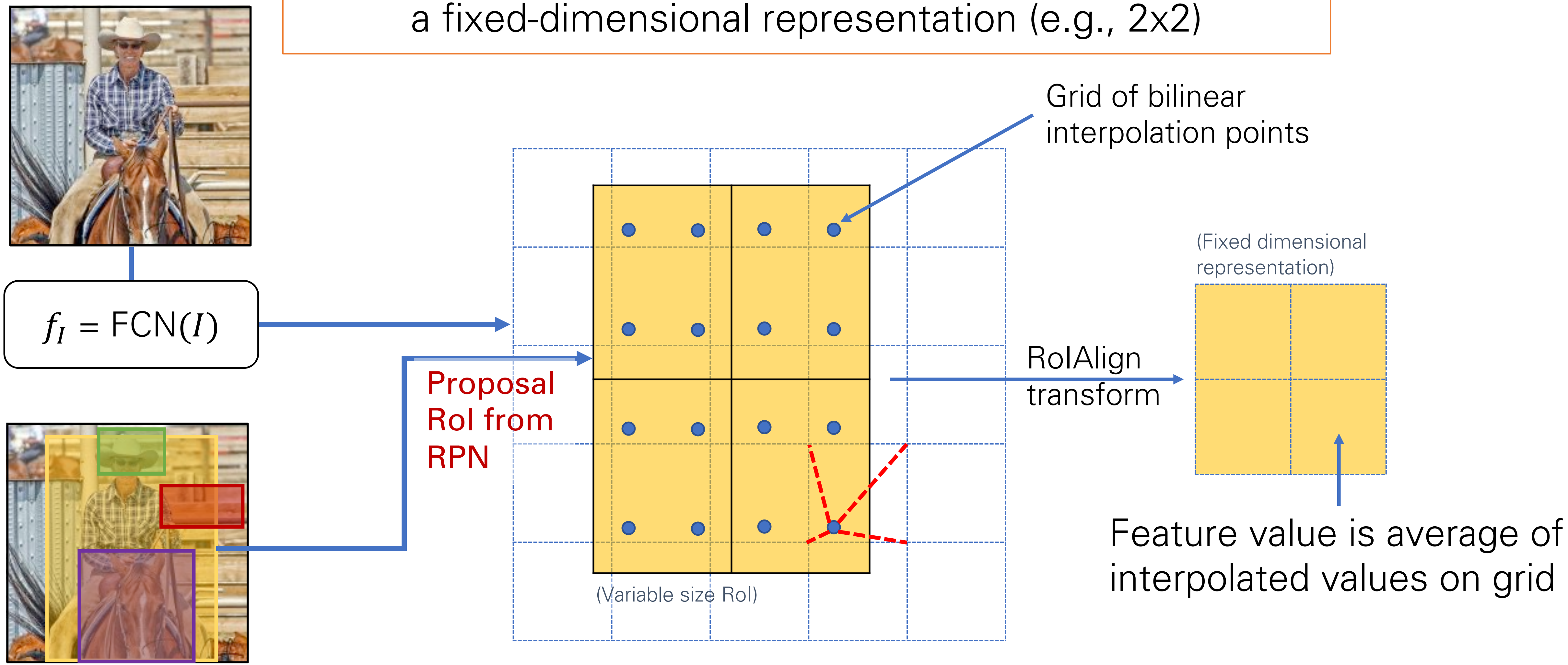
RoIAlign (on each Proposal)

Smoothly transform RoI features into
a fixed-dimensional representation (e.g., 2x2)



RoIAlign (on each Proposal)

Smoothly transform RoI features into
a fixed-dimensional representation (e.g., 2x2)



Compare to RoIPool

Preserve alignment or not?

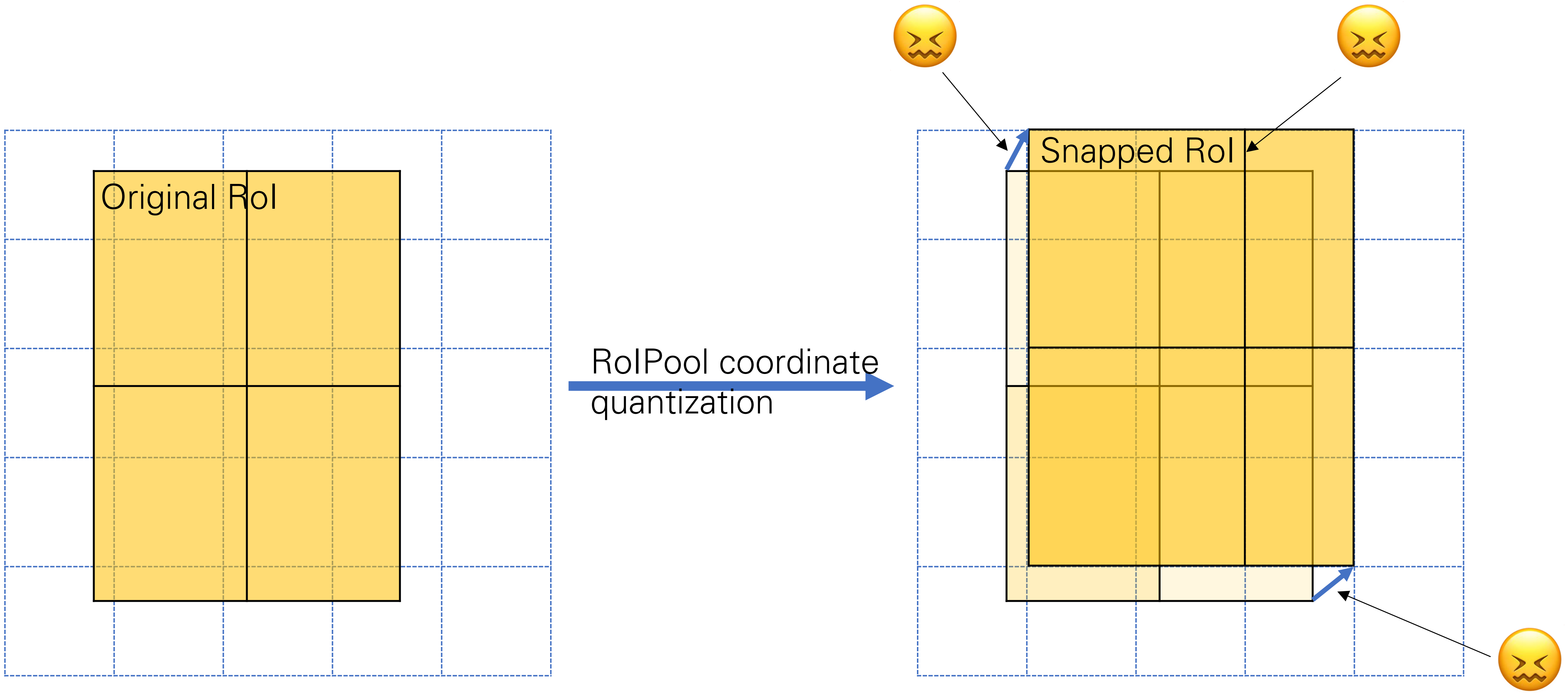
	align?	bilinear?	agg.	AP	AP_{50}	AP_{75}
RoIPool [12]			max	26.9	48.8	26.4
RoIWarp [10]		✓	max	27.2	49.2	27.1
		✓	ave	27.1	48.9	27.1
RoIAlign	✓	✓	max	**30.2**	**51.0**	**31.8**
	✓	✓	ave	**30.3**	**51.2**	**31.5**

+20% relative at high IoU

(c) **RoIAlign** (ResNet-50-C4): Mask results with various RoI layers. Our RoIAlign layer improves AP by ~3 points and AP_{75} by ~5 points. Using proper alignment is the only factor that contributes to the large gap between RoI layers.

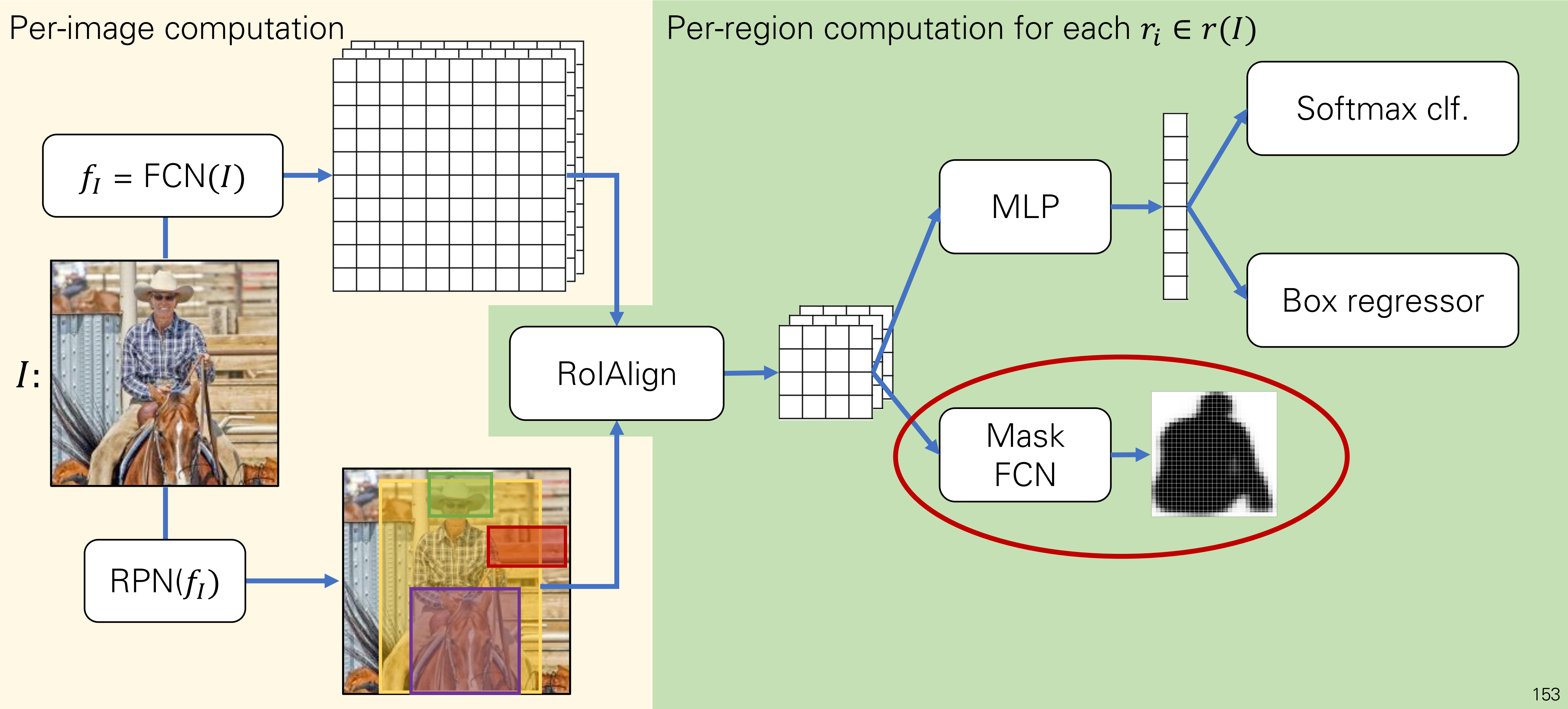
Compare to RoIPool

Quantization breaks pixel-to-pixel alignment



Instance Segmentation

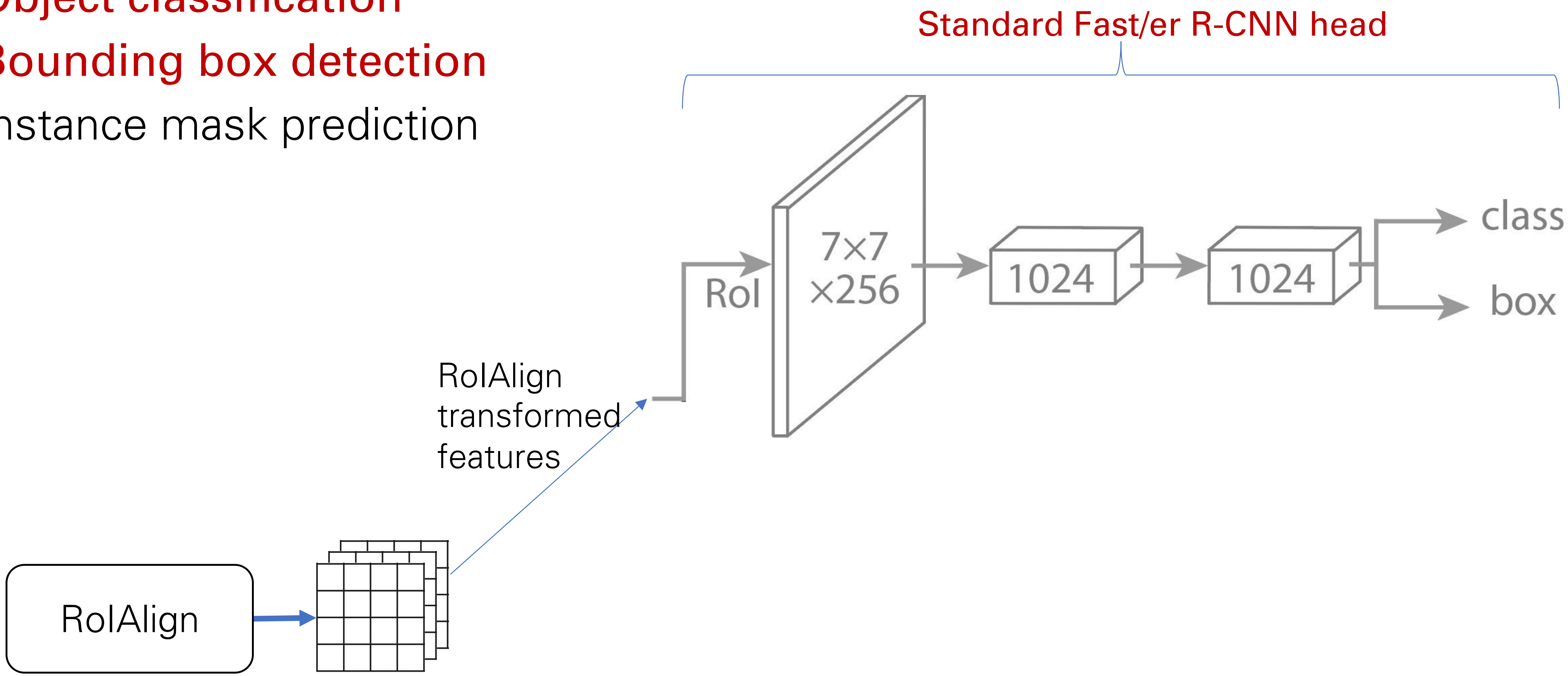
Mask R-CNN



Per-image computation
Per-region computation for each $r_i \in r(I)$
f_I = FCN(I)
I:
RPN(f_I)
RoIAlign
MLP
Softmax clf.
Box regressor
Mask FCN

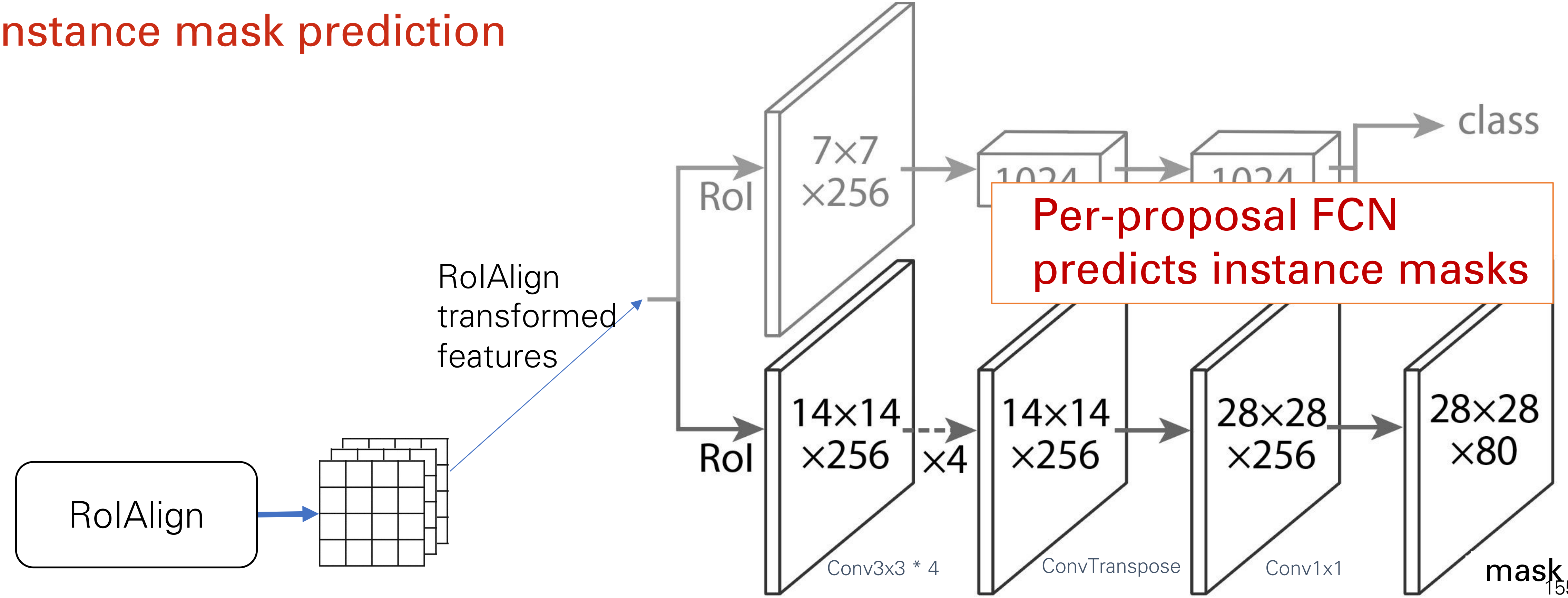
Mask Head (on each Proposal)

- Task specific heads for …
  - Object classification
  - Bounding box detection
  - Instance mask prediction

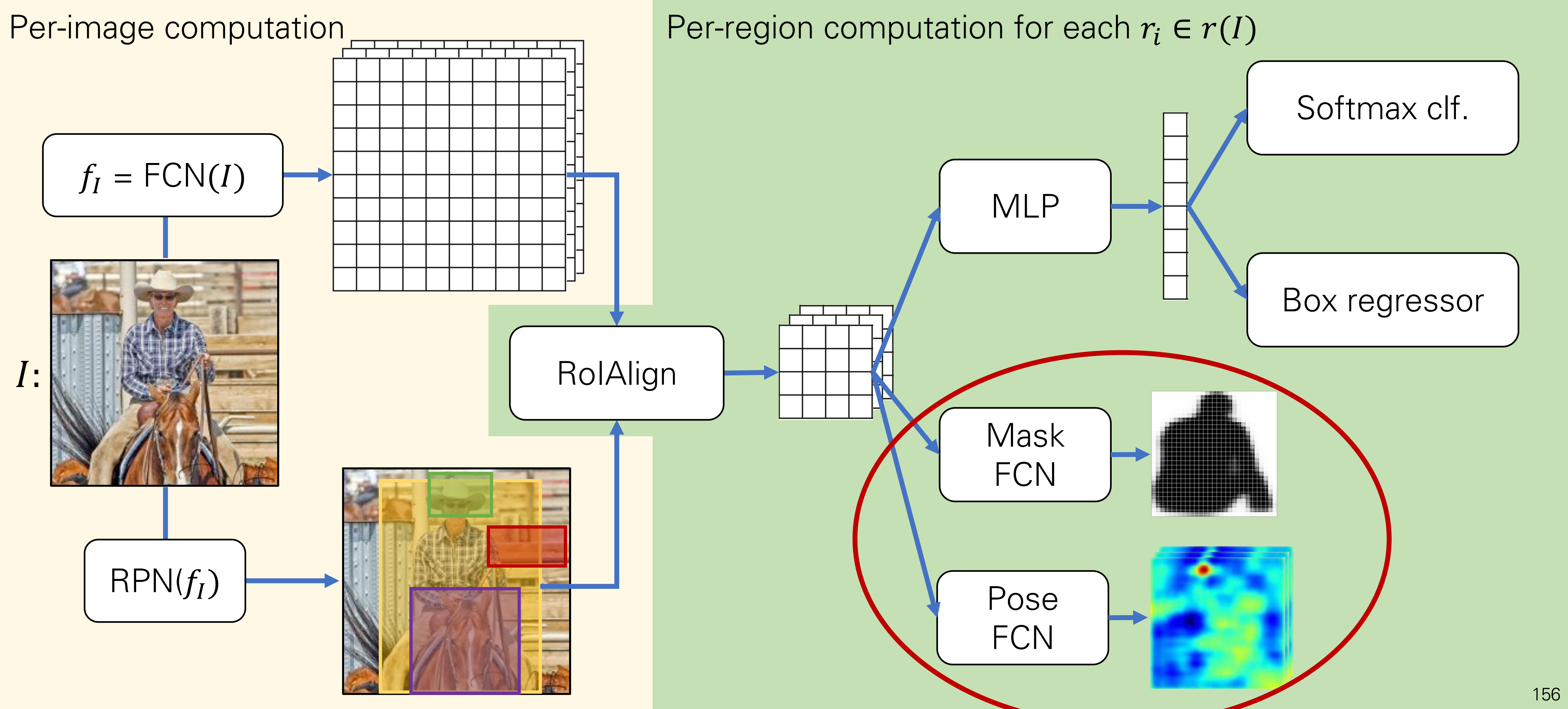


Mask Head (on each Proposal)

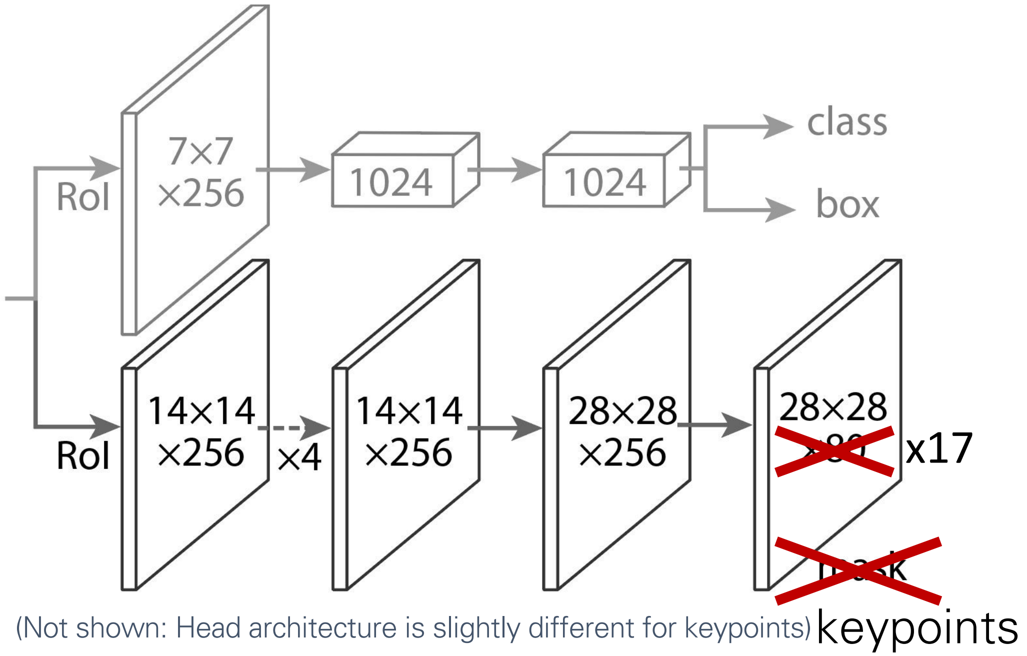
- Task specific heads for …
  - Object classification
  - Bounding box detection
  - Instance mask prediction



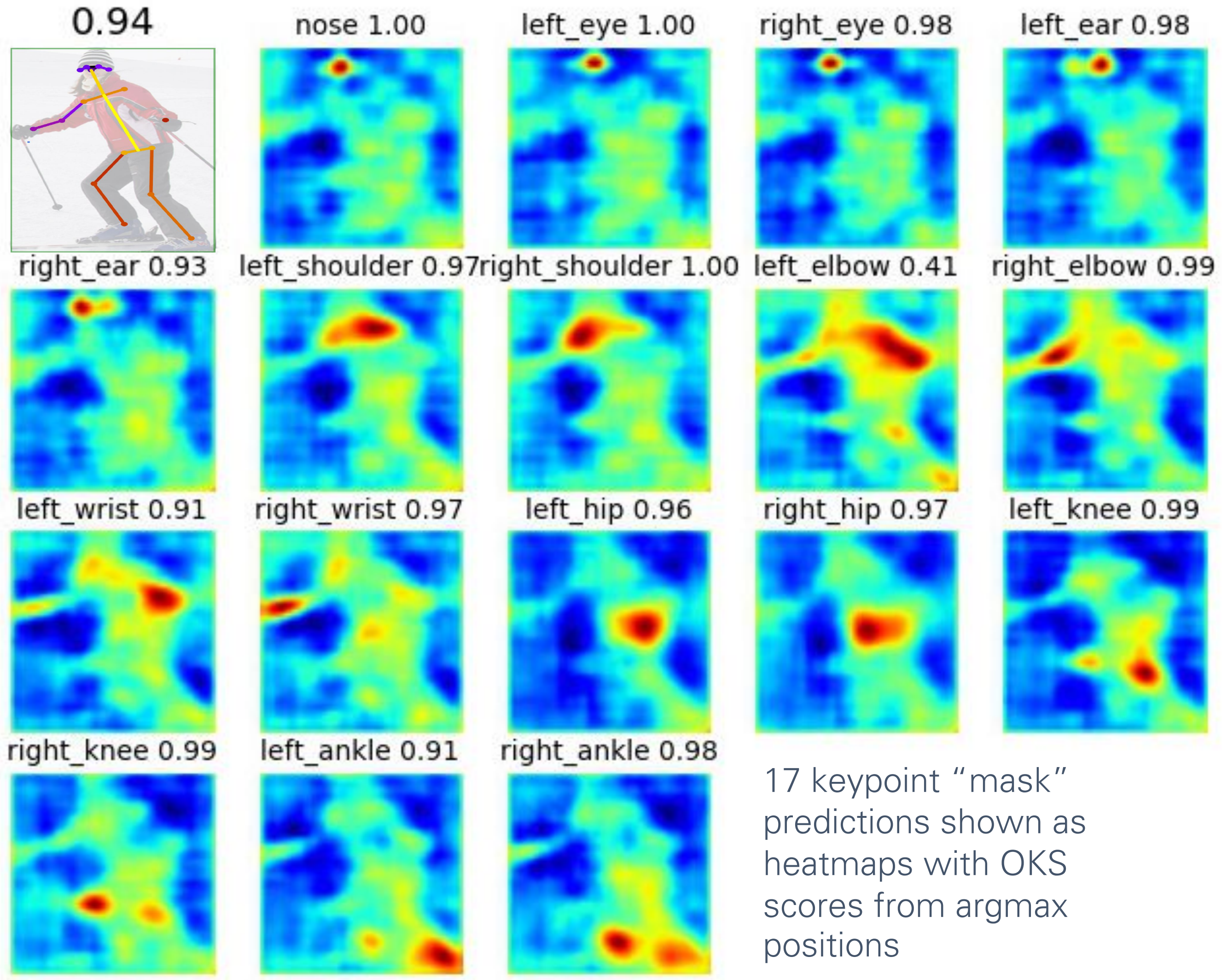
Mask R-CNN: Extension to 2D Human Pose



Pose Head



(Not shown: Head architecture is slightly different for keypoints)



17 keypoint "mask" predictions shown as heatmaps with OKS scores from argmax positions

- Add keypoint head (28x28x17)
- Predict one "mask" for each keypoint
- Softmax over spatial locations (encodes one keypoint per mask "prior")

Mask R-CNN: Training

- Same as “image centric” Fast/er R-CNN training
- But with training targets for masks

Example Mask Training Targets

Image with training proposal

28x28 mask target

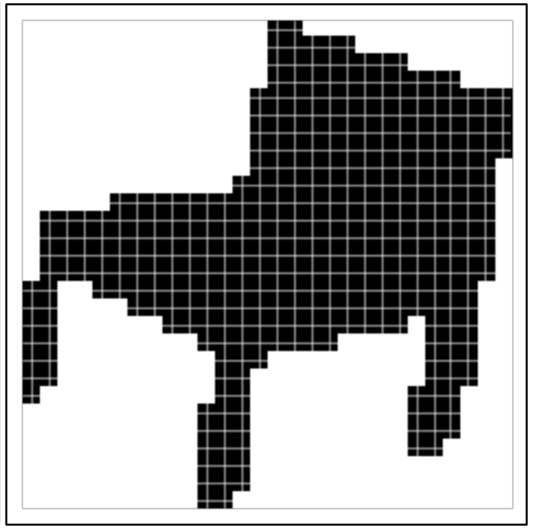
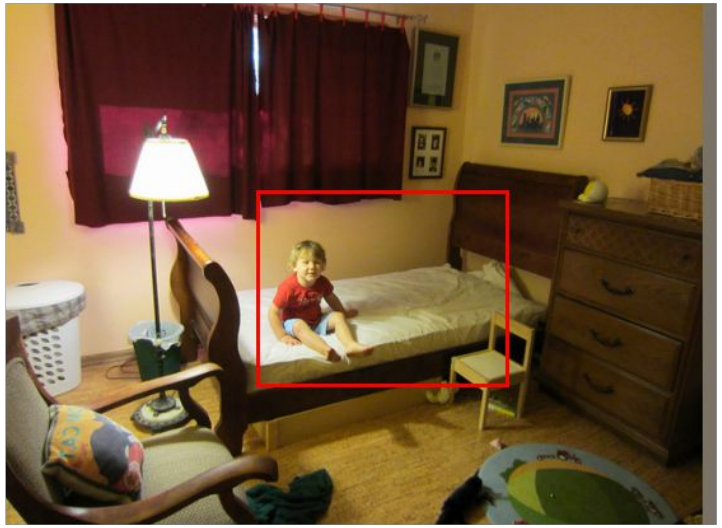
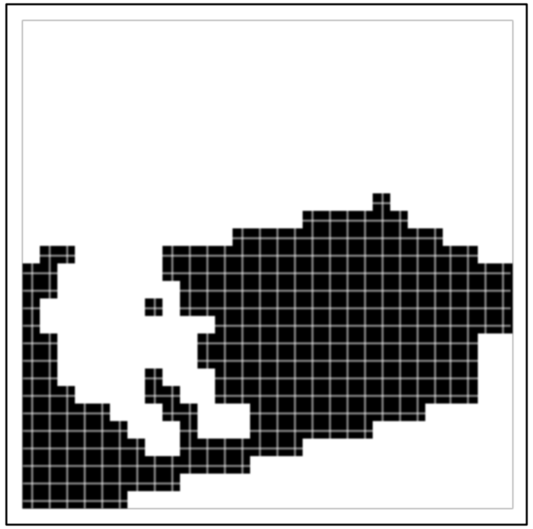
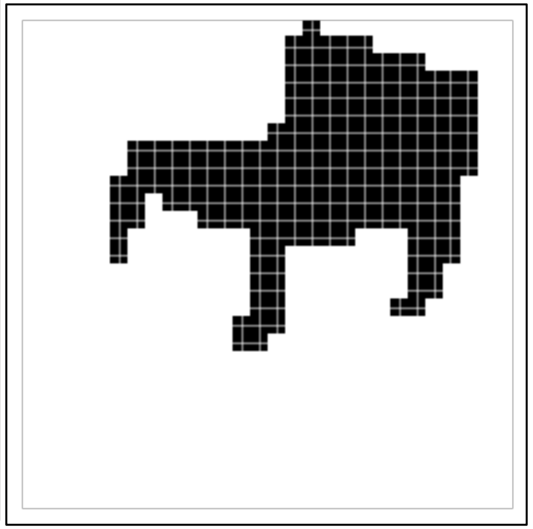


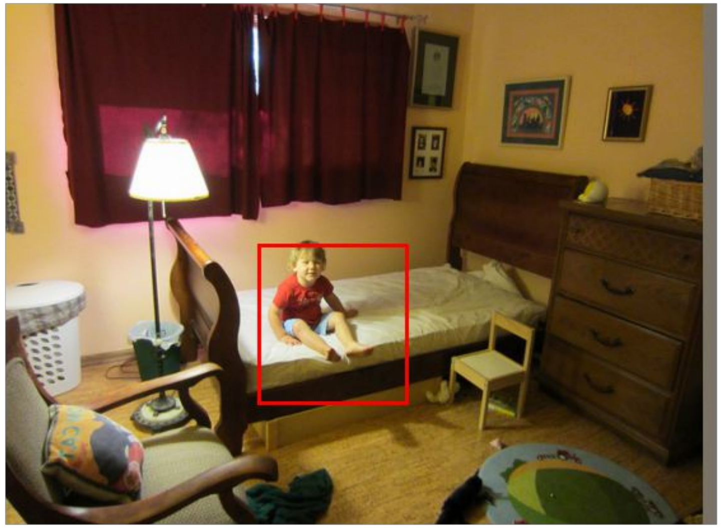
Image with training proposal

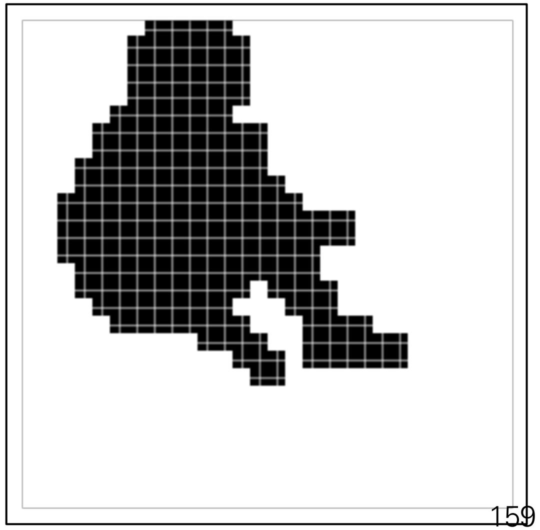


28x28 mask target







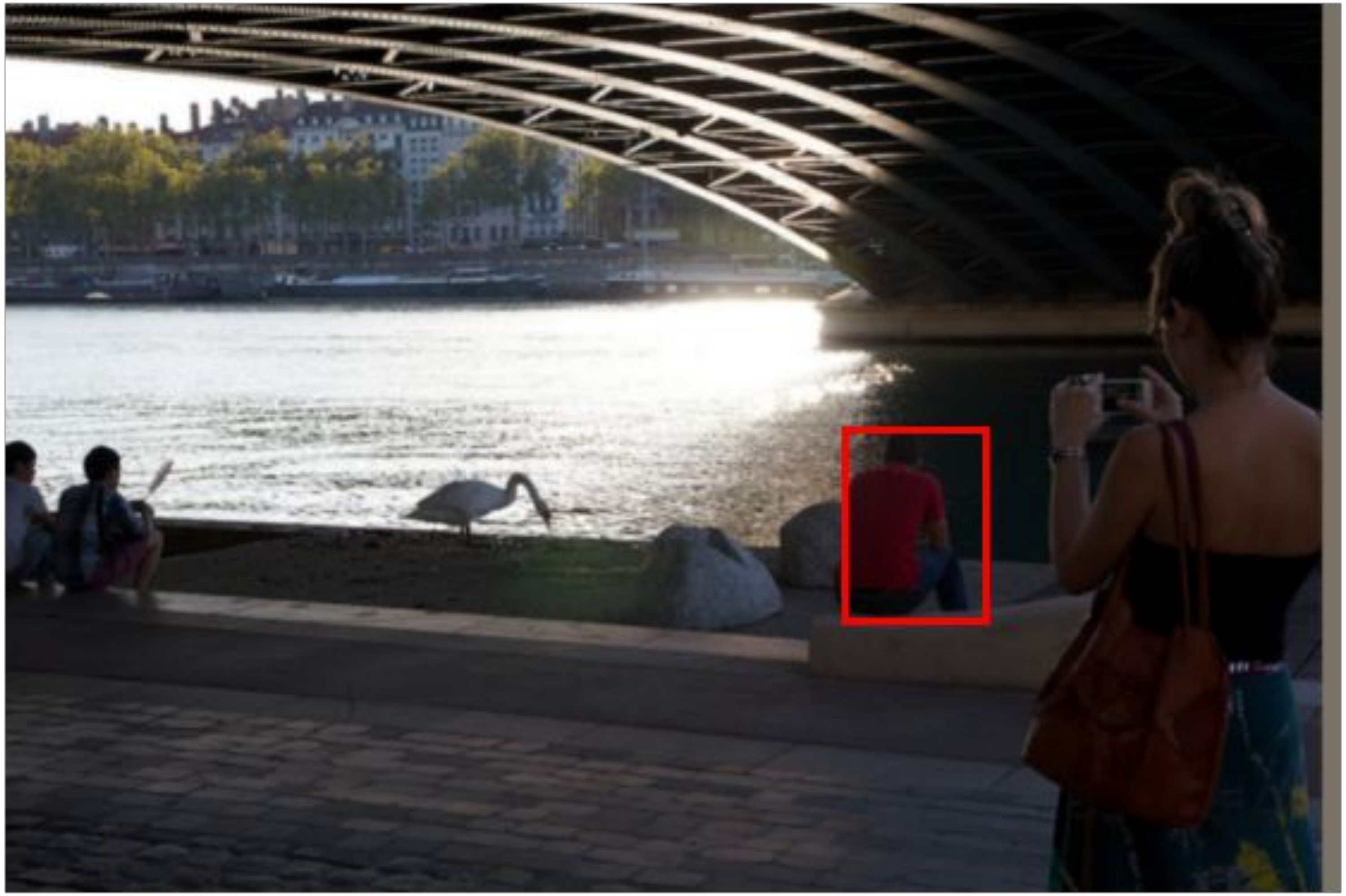


Mask R-CNN: Inference

1. **Perform Faster R-CNN inference**
   - Run backbone FCN
   - Generate proposals with RPN
   - Score the proposals with clf. head
   - Refine proposals with box regressor
   - Apply NMS and take the top K (= 100, e.g.)

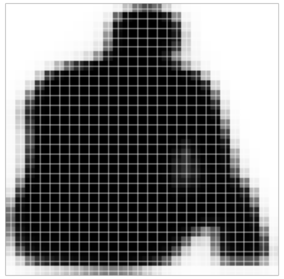
2. **Run RoIAlign and mask head on top-*K* refined, post-NMS boxes**
   - Fast (only compute masks for top-K detections)
   - Improves accuracy (uses refined detection boxes, not proposals)

Mask Prediction



Validation image with box detection shown in red

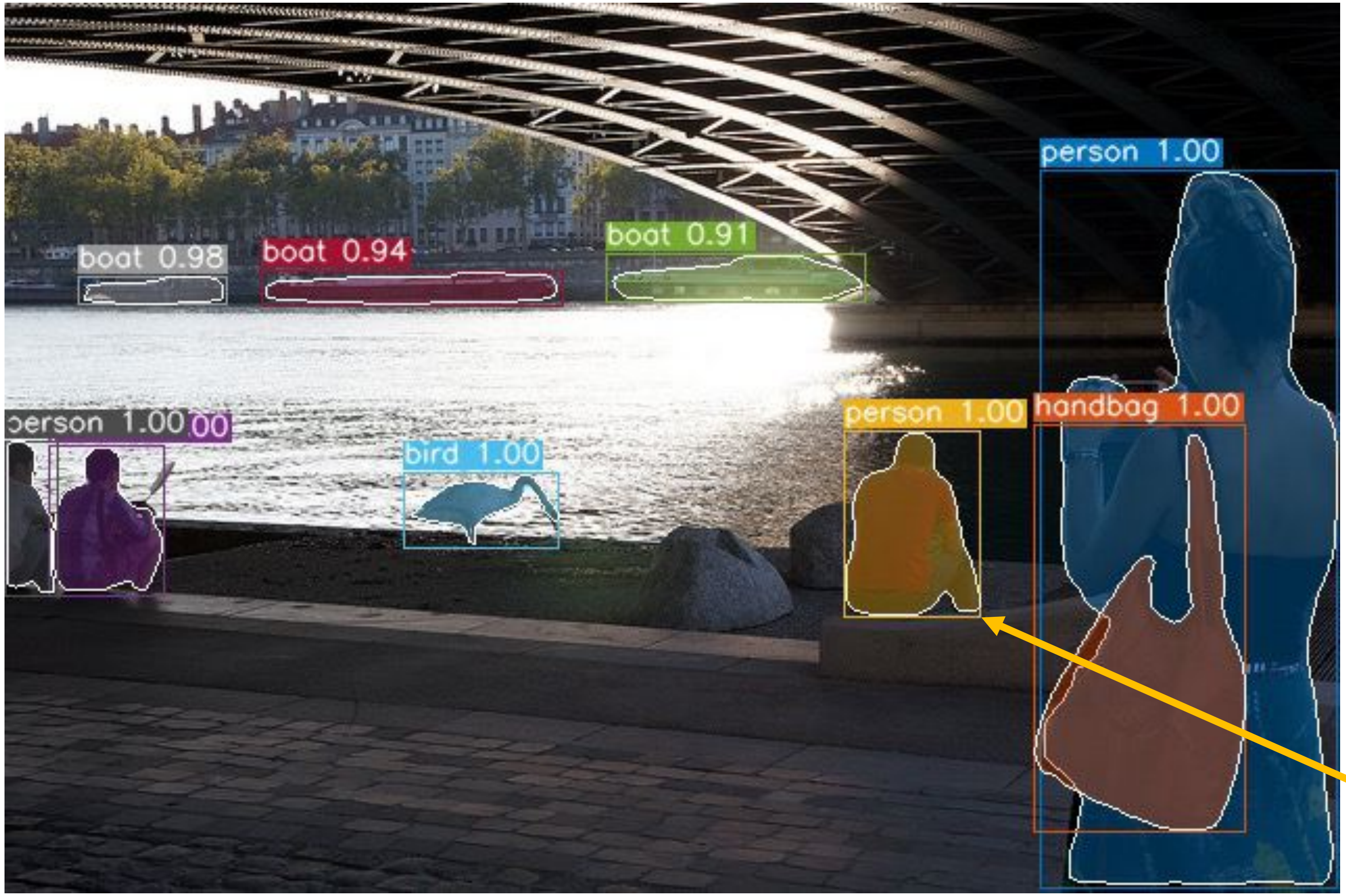
28x28 soft prediction from Mask R-CNN
(enlarged)



Soft prediction **resampled to image coordinates**
(bilinear and bicubic interpolation work equally well)

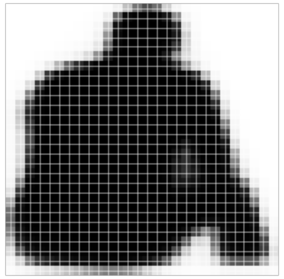
Final prediction (threshold at 0.5)

Mask Prediction



Validation image with box detection shown in red

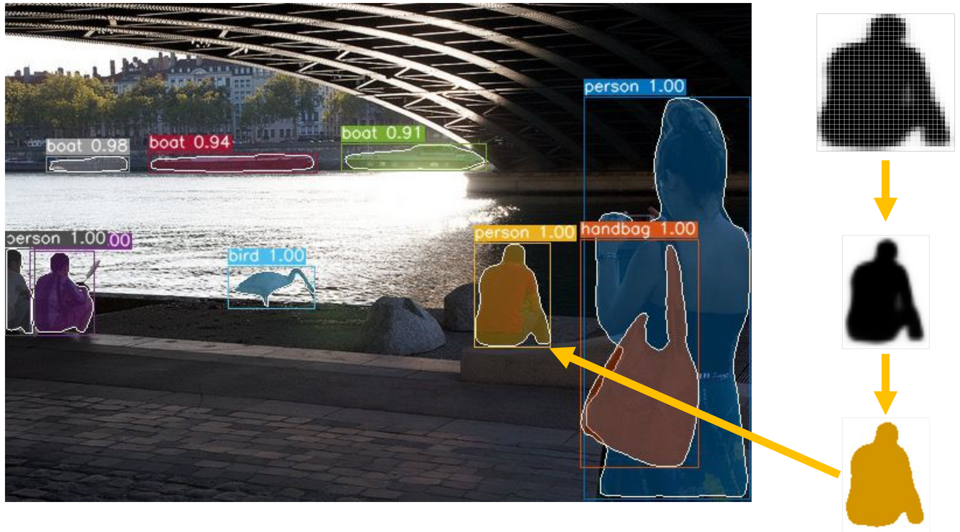
28x28 soft prediction from Mask R-CNN
(enlarged)

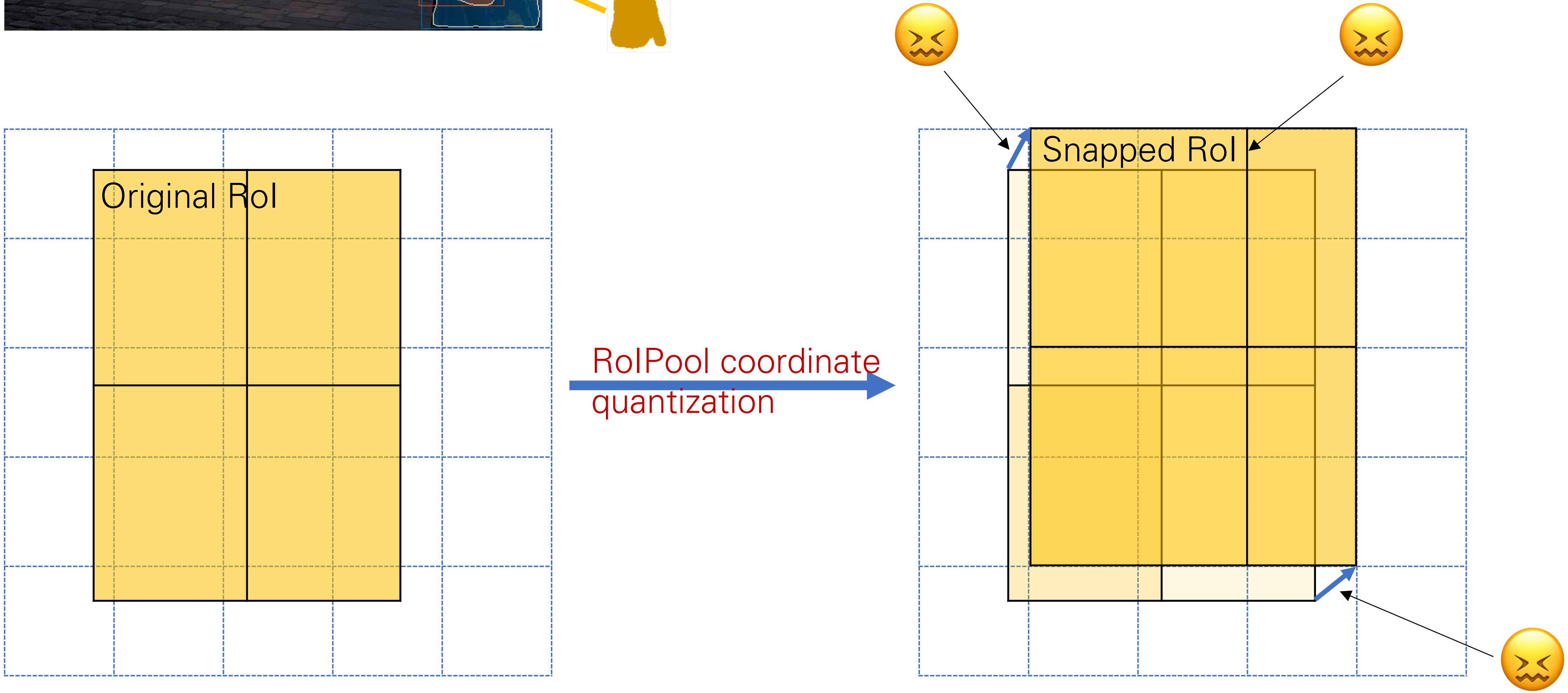


Soft prediction **resampled to image coordinates**
(bilinear and bicubic interpolation work equally well)

Final prediction (threshold at 0.5)

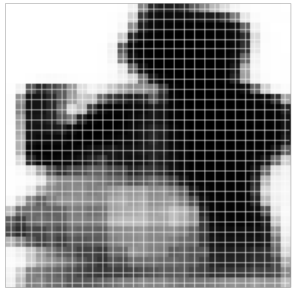
Quantization breaks pixel-to-pixel alignment





Mask Prediction

28x28 soft prediction



Resized soft prediction

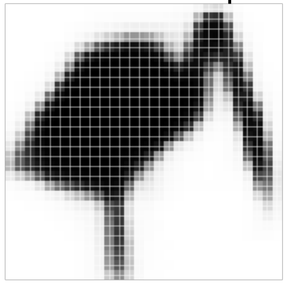
Final mask

Validation image with box detection shown in red

Mask Prediction

Validation image with box detection shown in red

28x28 soft prediction

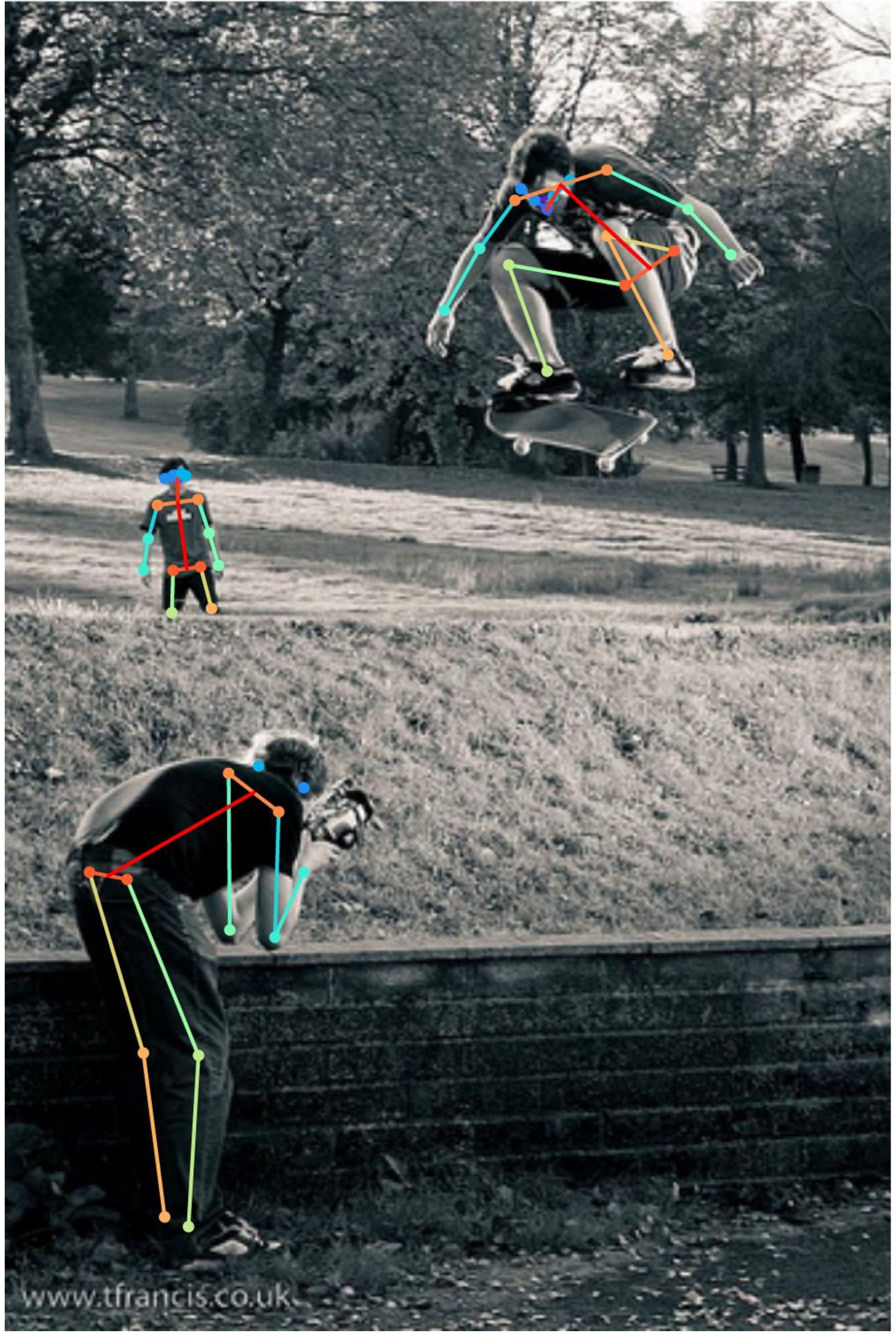


Resized Soft prediction

Final mask

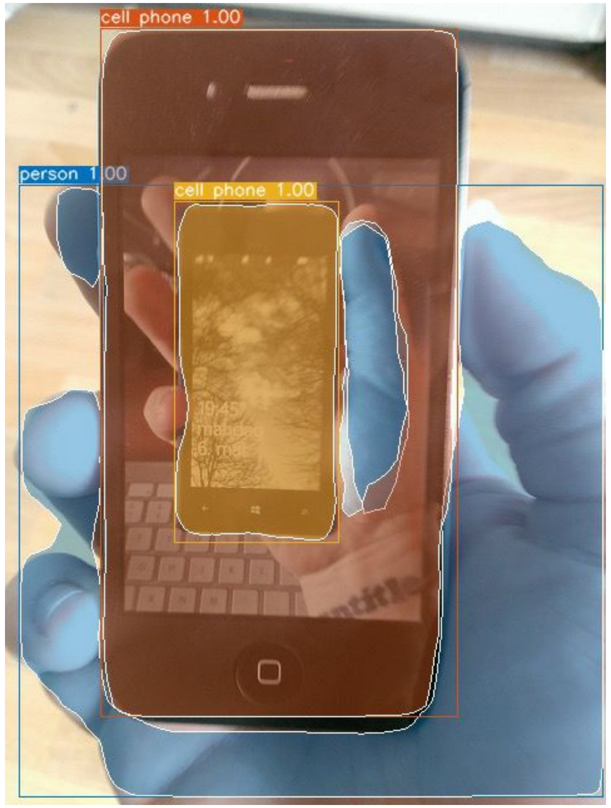

person1.00
person1.00
person.91
surfboard1.00
surfboard1.00
person1.00
surfboard.98
person1.00
surfboard1.00
person.98
surfboard1.00
person.74


person1.00
person.88
tv.98
tv.84
person1.00
person1.00
bottle.97
wine glass.99
dining table.95
wine glass1.00
wine glass1.00


www.tfrancis.co.uk

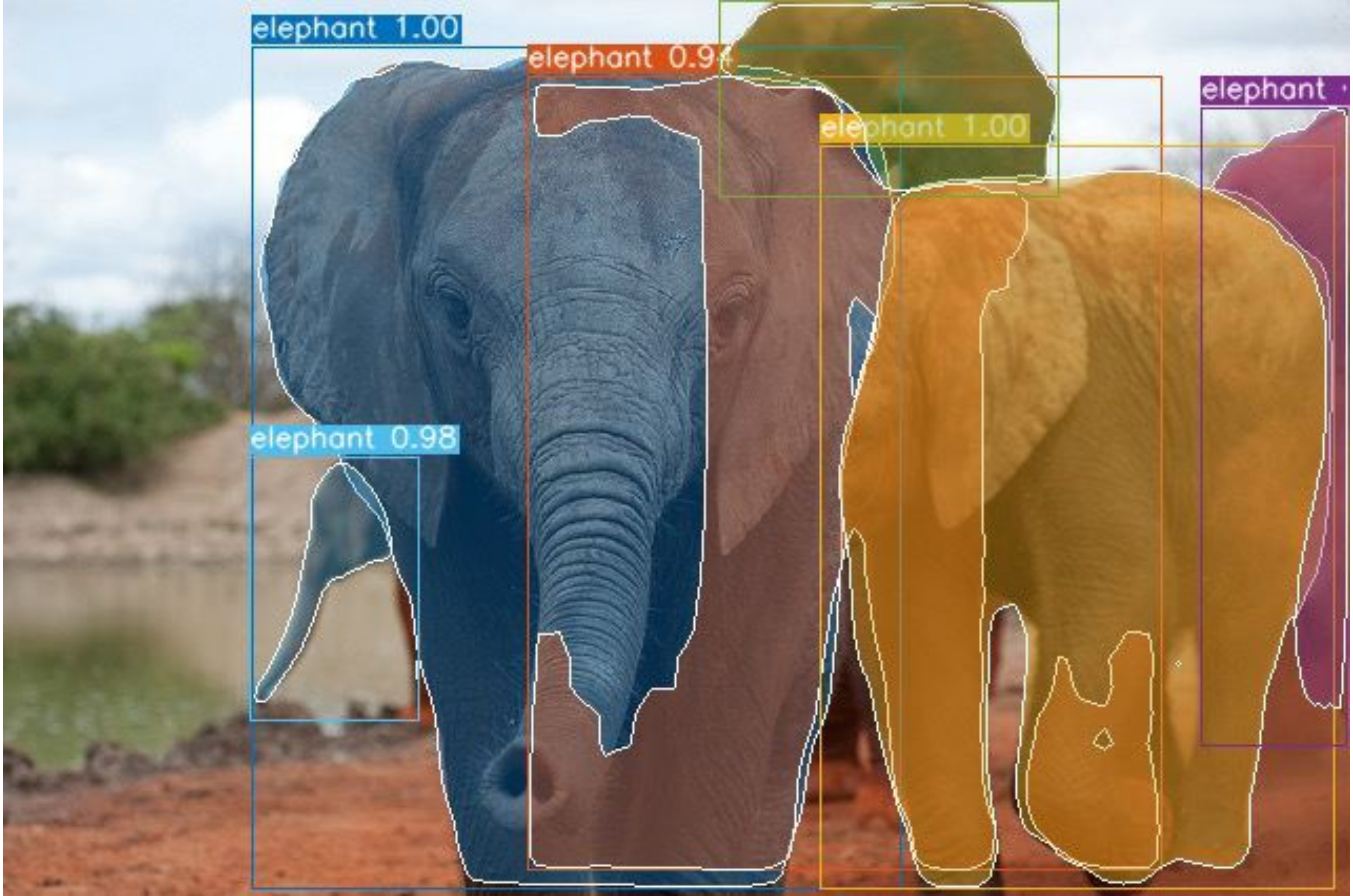
Is Object Detection Solved?

- Obviously no; there are **frequently silly errors**
- But it is getting frustratingly good
- The errors are often reasonable
- The bottlenecks are raw recognition and “reasoning”




tv 1.00
person 1.00
bottle 0.92
bottle 0.93
sink 0.99
sink 0.99
sink 0.96
bed 0.98

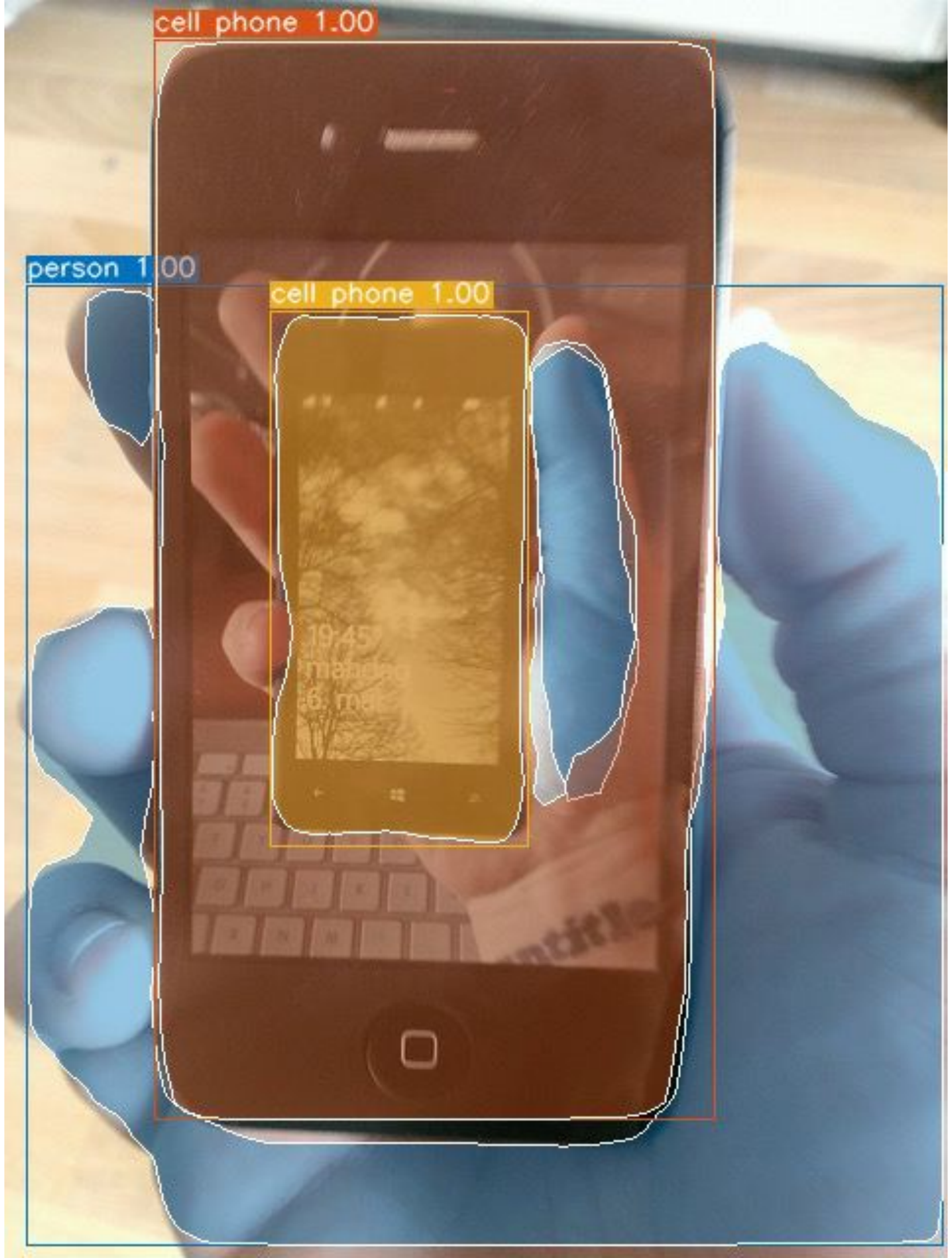
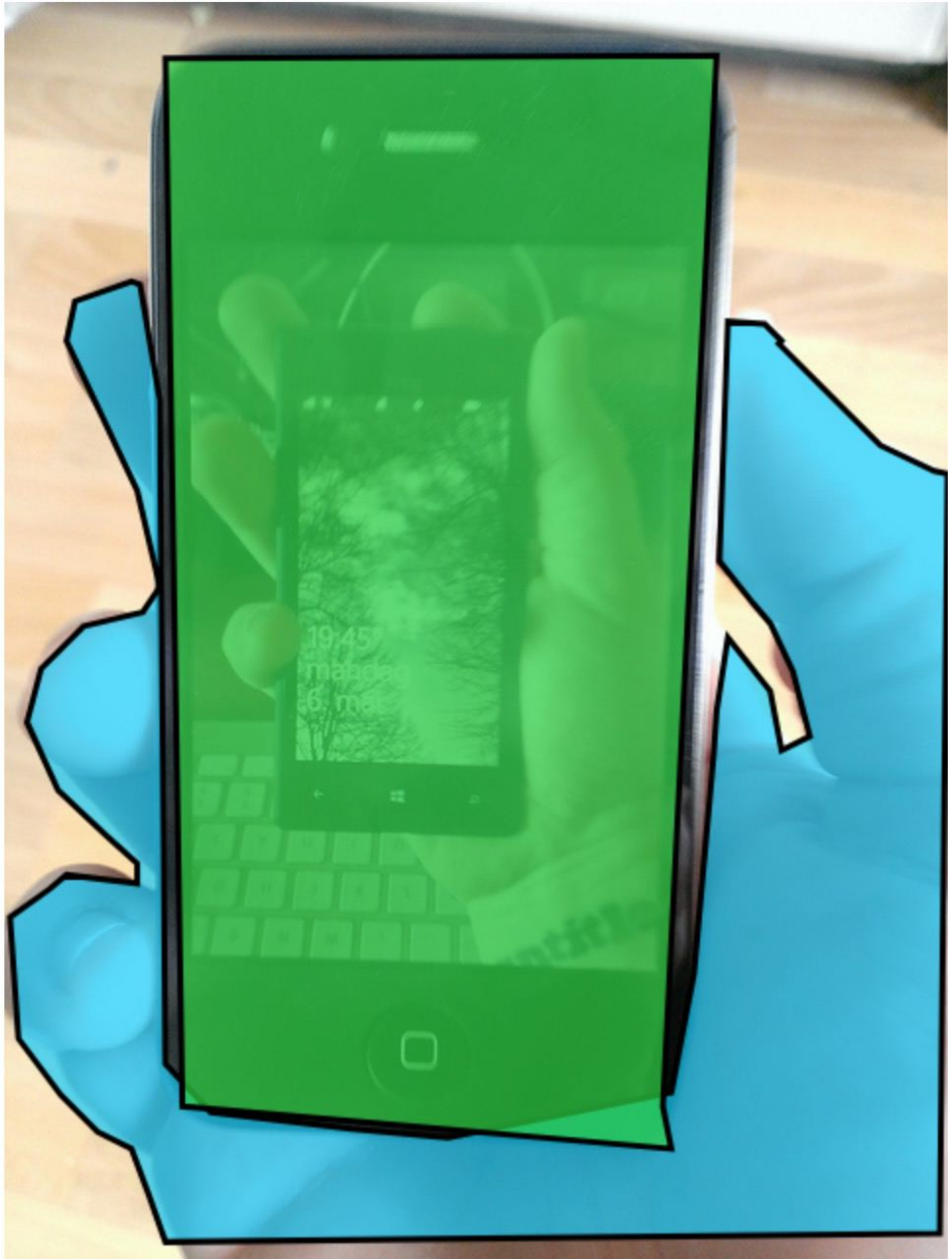

bed 1.00
bed 0.98
bed 0.90
bed 0.94
11


elephant 1.00
elephant 0.94
elephant 1.00
elephant
elephant 0.98


person 1.00
person 0.99
person 0.95
person 0.94
person 0.93
person 1.00
cup 0.9
person 0.97
bottle 0.94
bottle 0.99
bottle 0.99
bottle 0.99
bottle 0.95

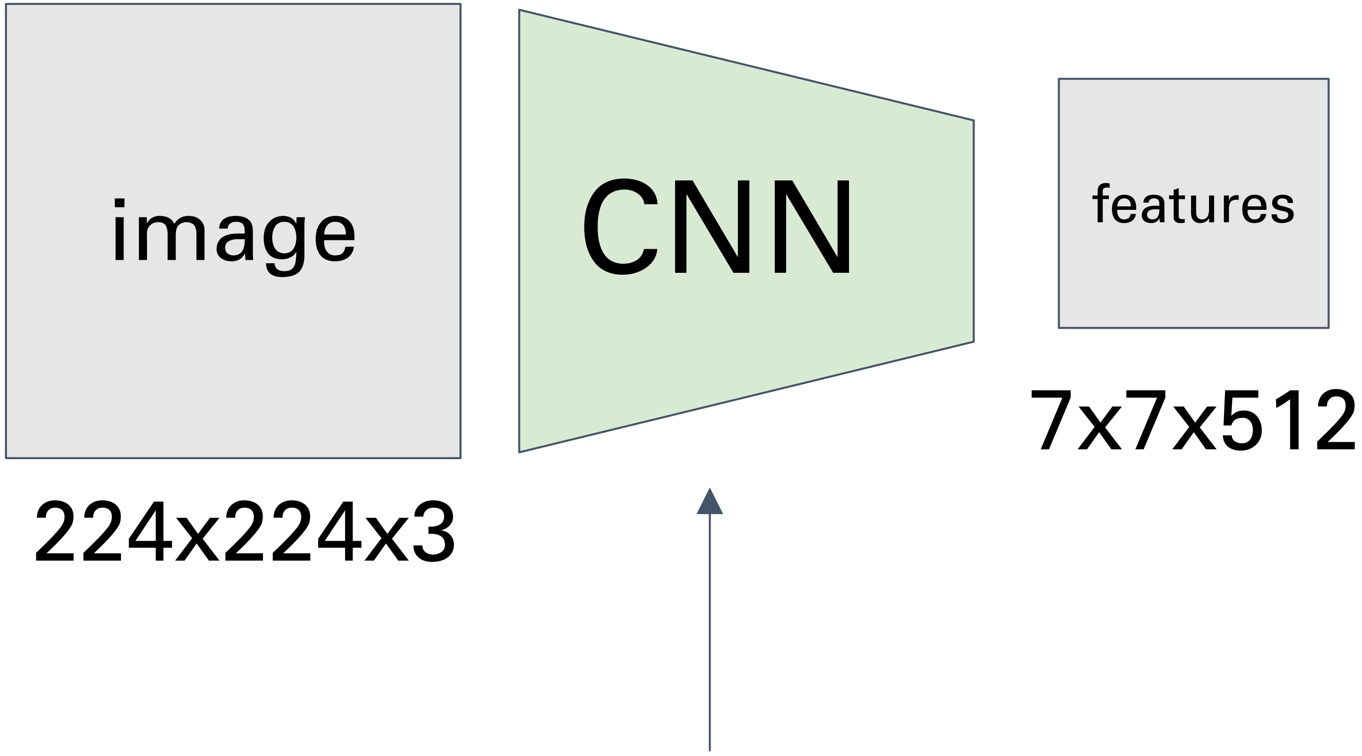

person 1.00
tie 0.96


person 0.95
vase 0.97
orange 0.96
orange 0.97
book 0.


cell phone 1.00
person 1.00
cell phone 1.00


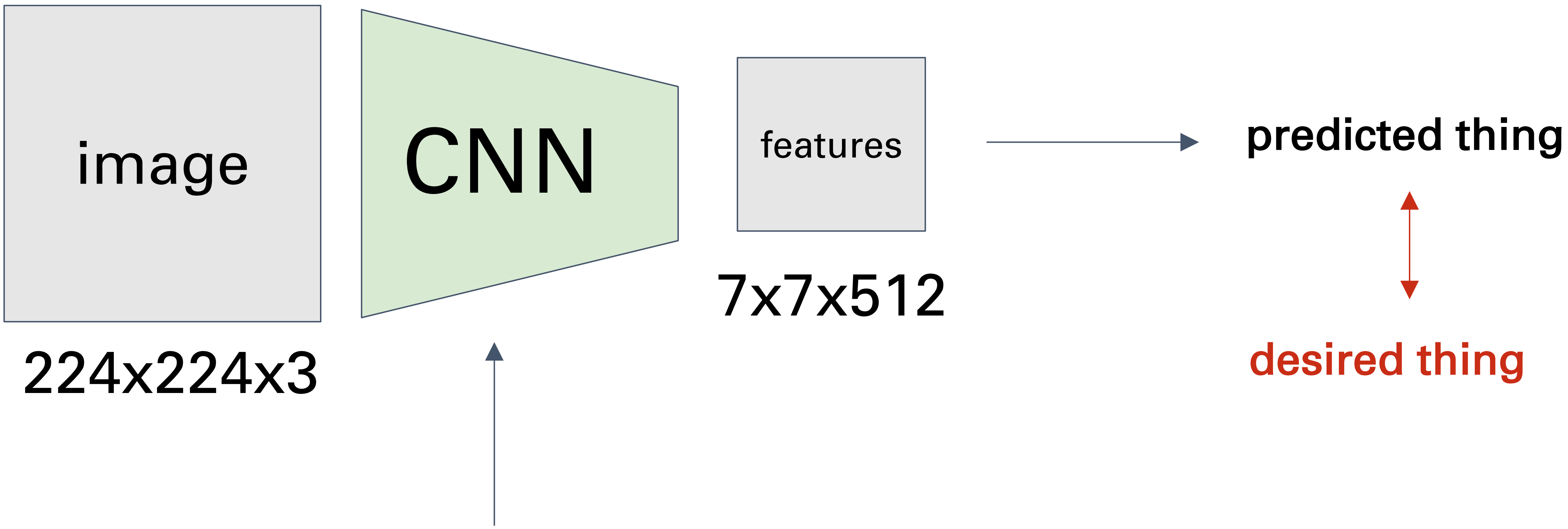
Addressing other tasks…

Addressing other tasks...



A block of compute with a few million parameters.

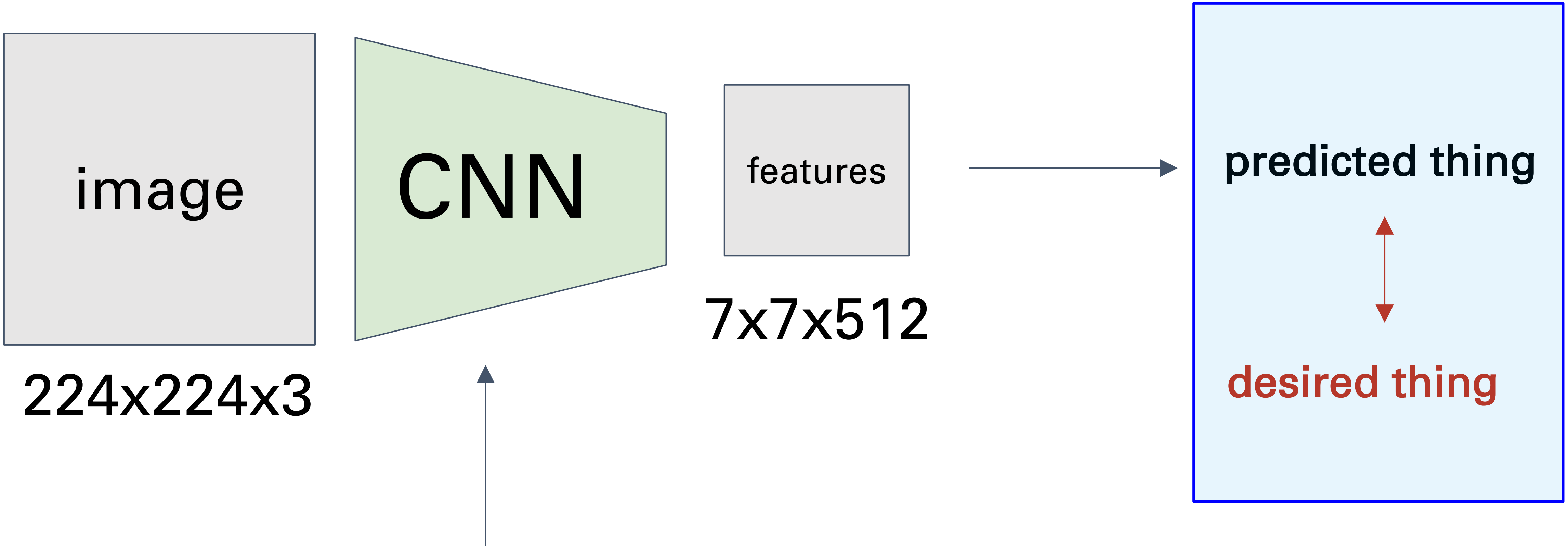
Addressing other tasks...



A block of compute with a few million parameters.

Addressing other tasks...

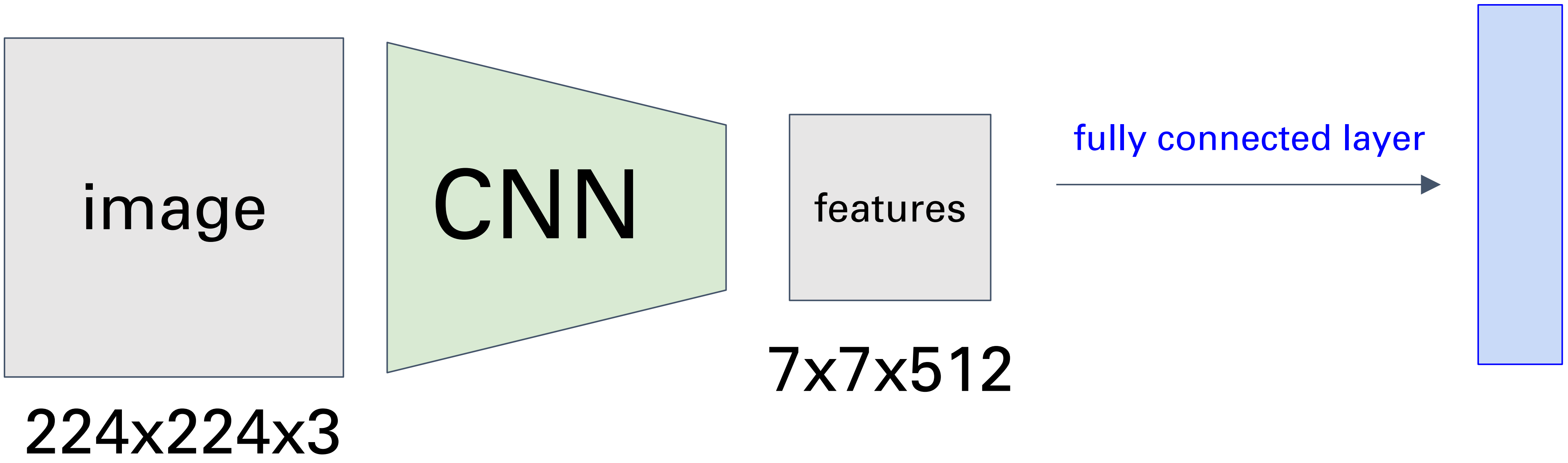
this part
changes from
task to task



A block of compute with a
few million parameters.

Image Classification

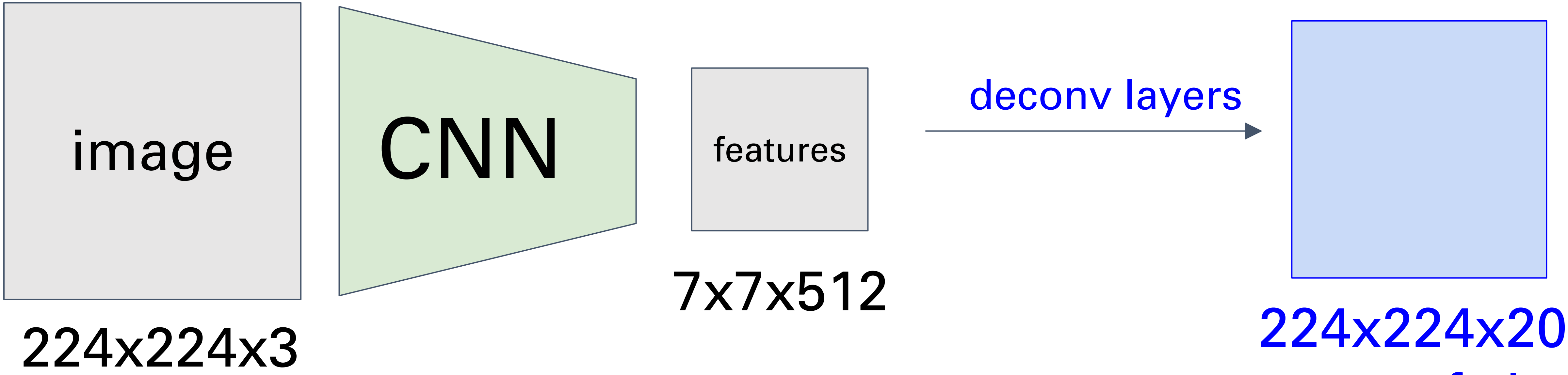
thing = a vector of probabilities for different classes



e.g. vector of 1000 numbers giving probabilities for different classes.

Segmentation

image class “map”



224x224x20
array of class
probabilities
at each pixel.

Localization

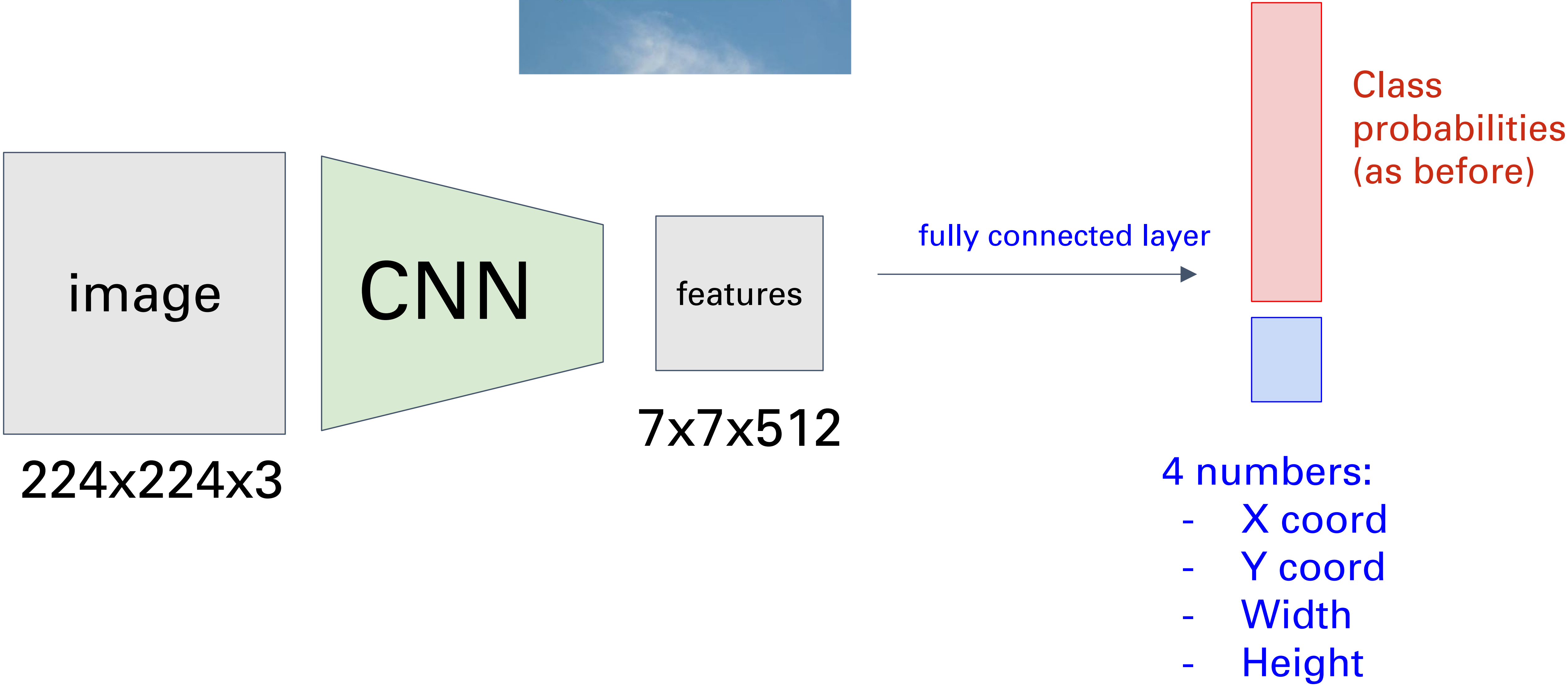
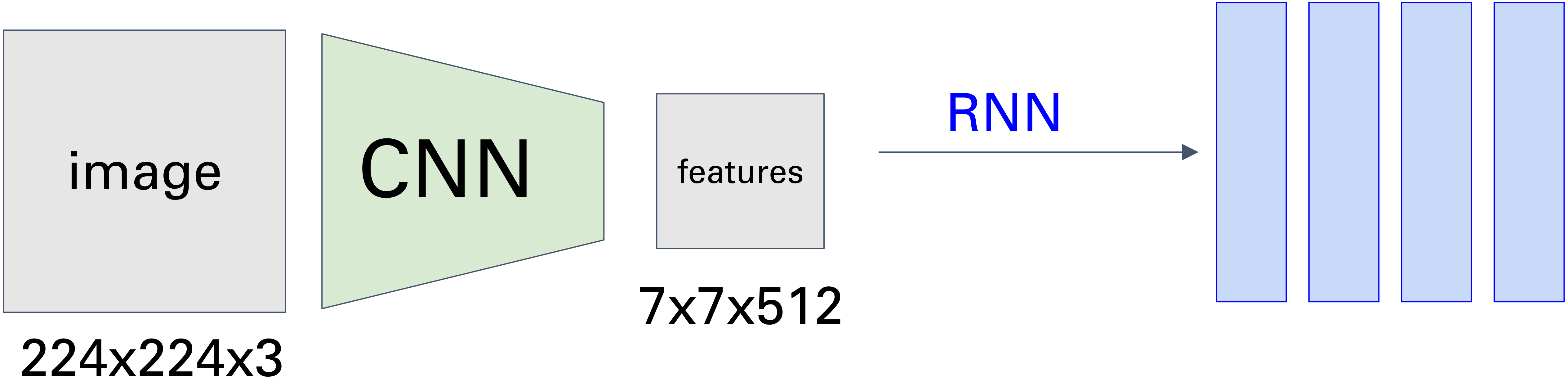


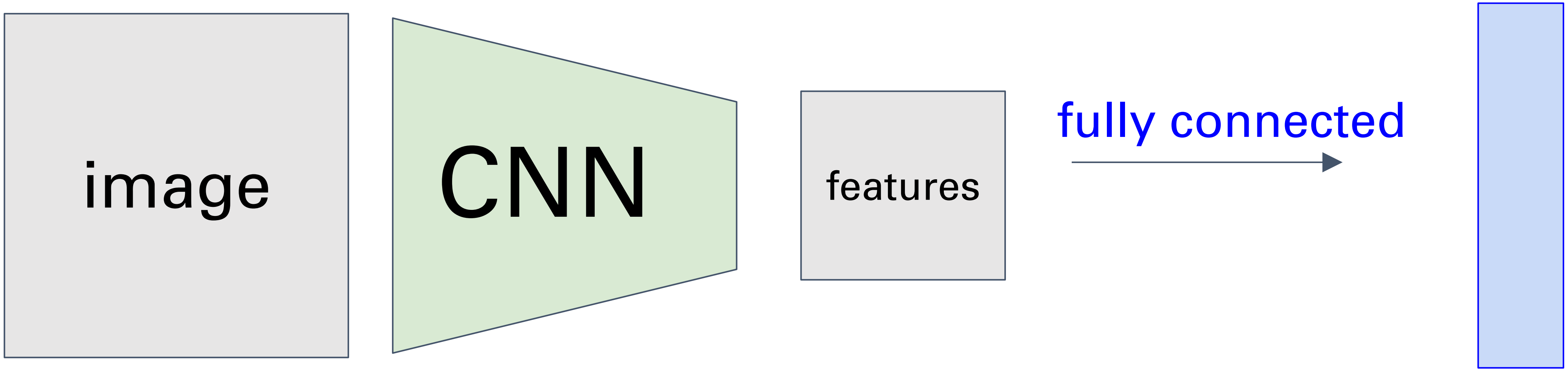
Image Captioning



A sequence of 10,000-dimensional vectors giving probabilities of different words in the caption.

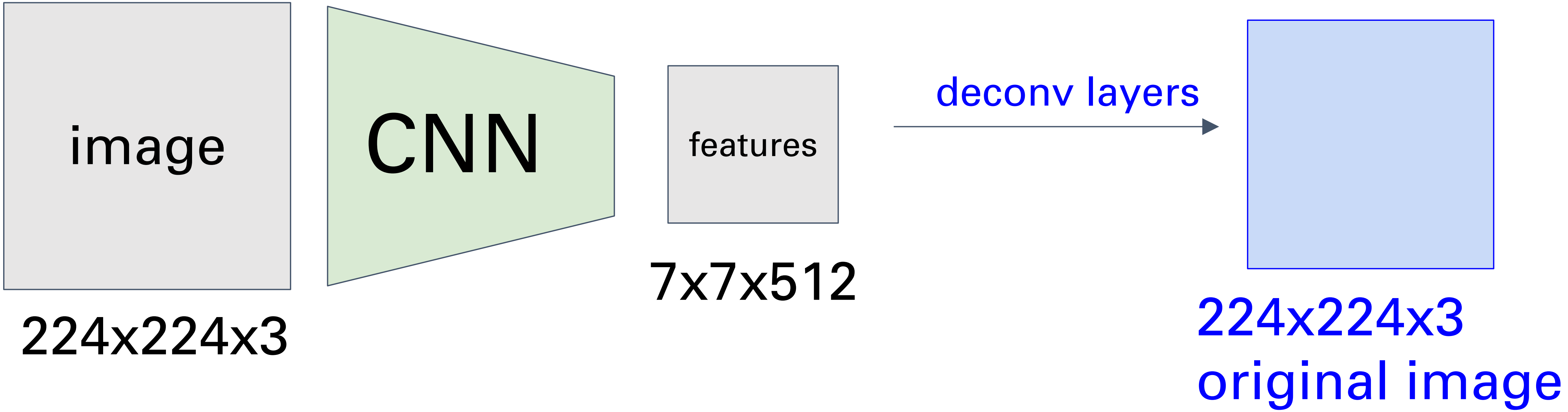
Reinforcement Learning

Mnih et al. 2015

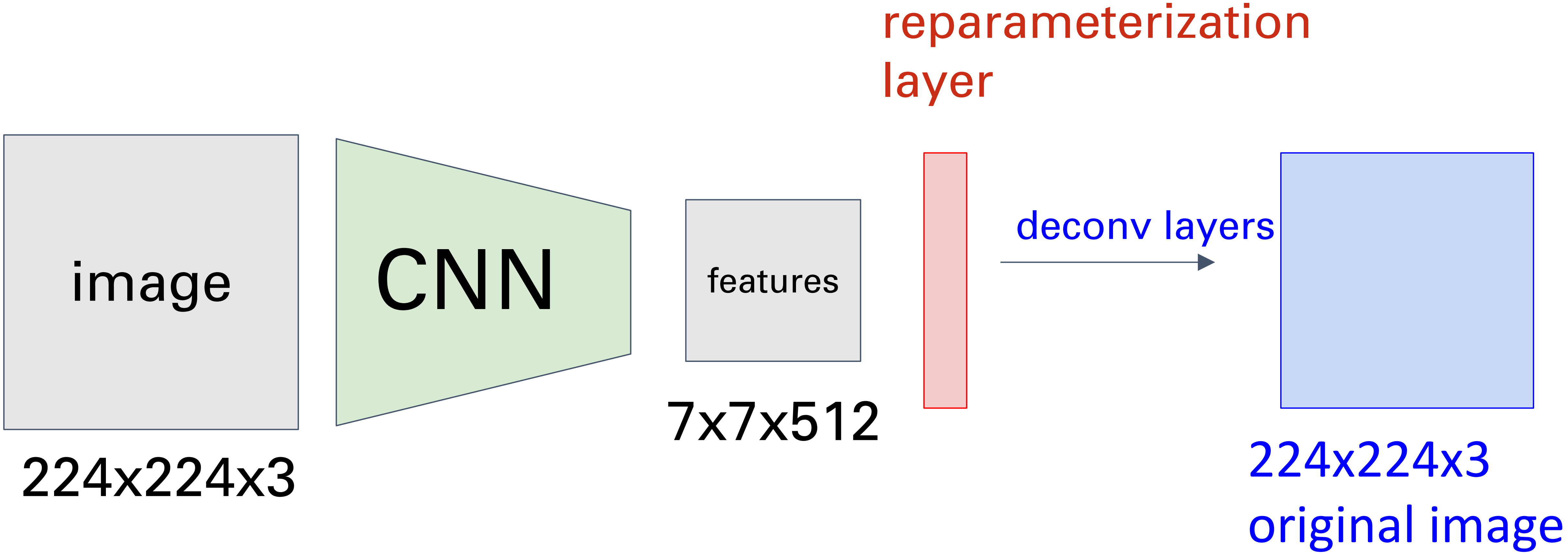


e.g. vector of 8 numbers giving probability of wanting to take any of the 8 possible ATARI actions.

Autoencoders

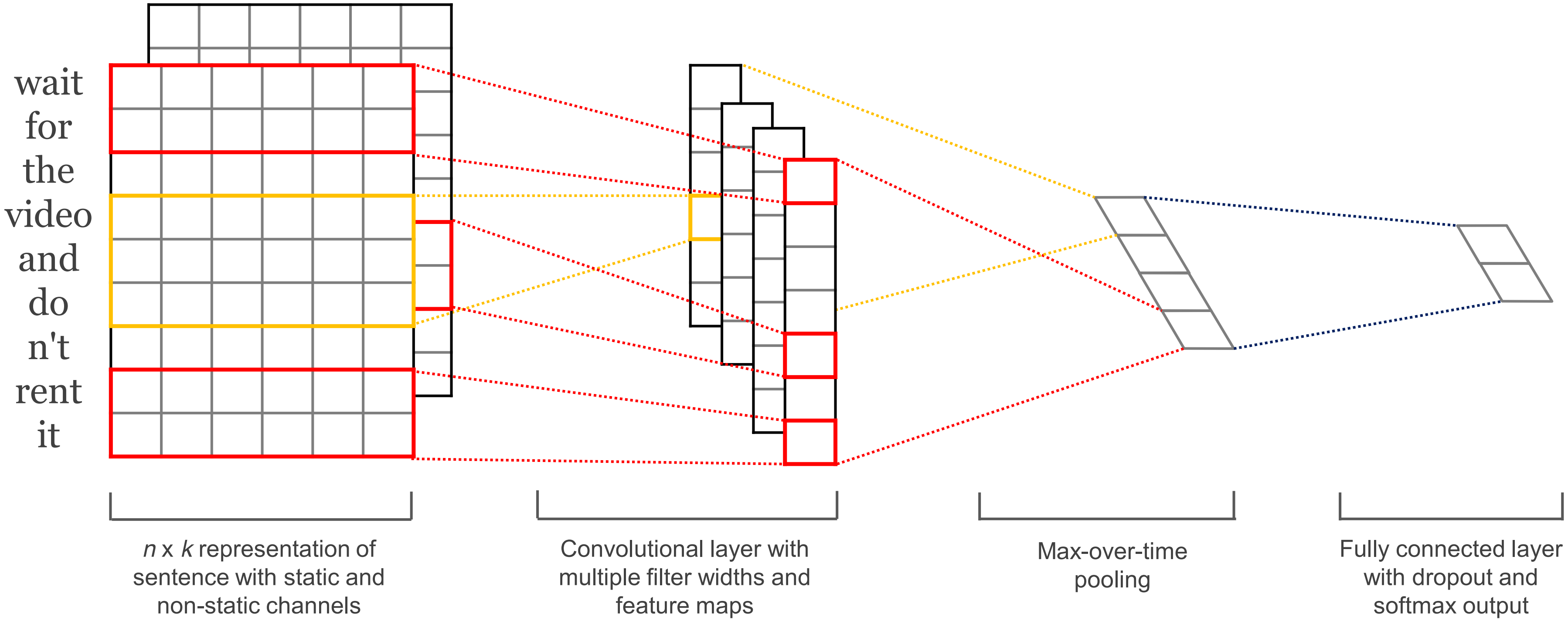


Variational Autoencoders



[Kingma et al.], [Rezende et al.], [Salimans et al.]

Addressing other tasks…

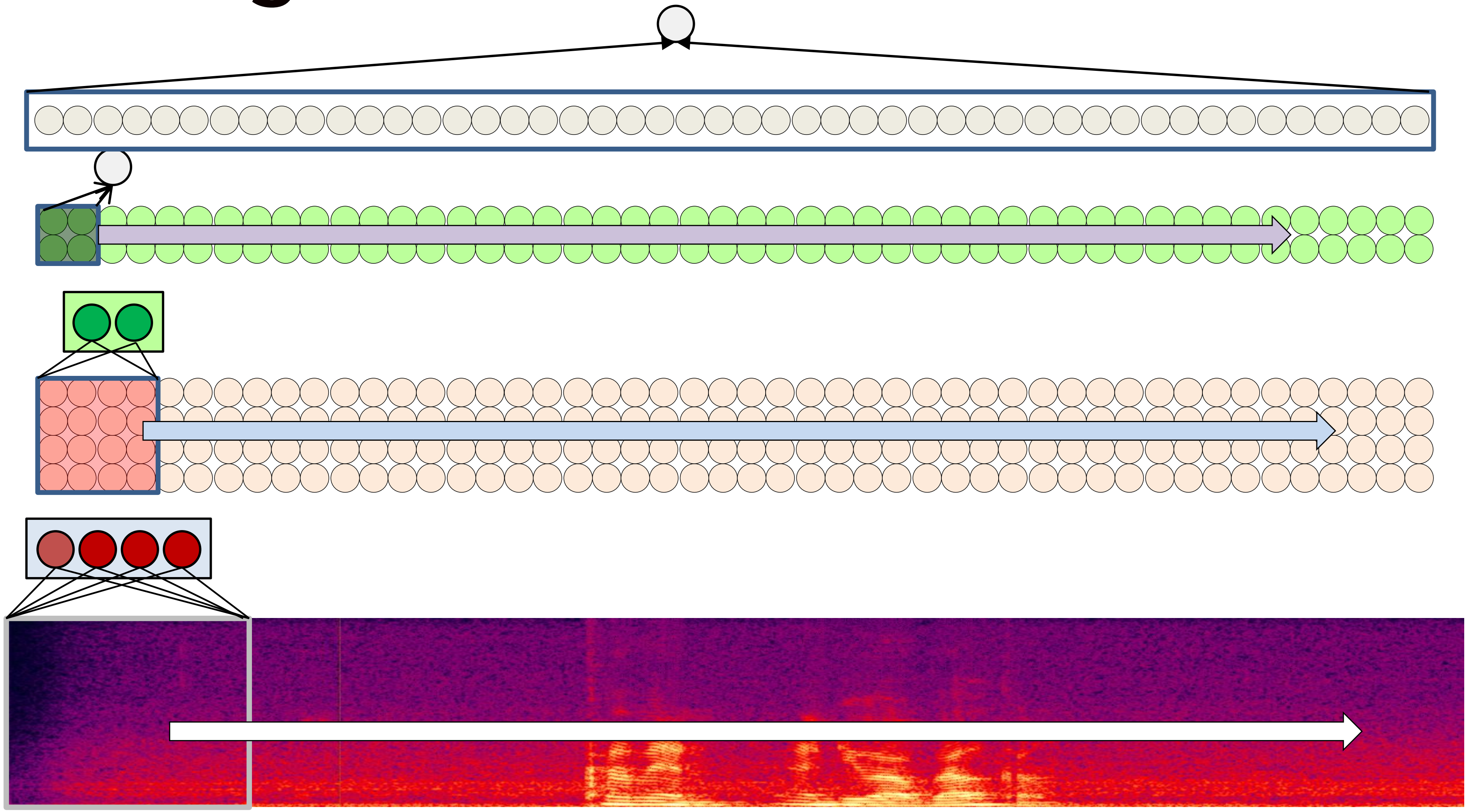


- 1D convolution ≈ Time Delay Neural Networks (Waibel et al. 1989, Collobert and Weston 2011)
- Two main paradigms:
  - **Context window modeling:** For tagging, etc. get the surrounding context before tagging
  - **Sentence modeling:** Do convolution to extract n-grams, pooling to combine over whole sentence

Figure credit: Yoon Kim

Slide adapted from Graham Neubig

Addressing other tasks...



- **CNNs for audio processing:** MFCC features + Time Delay Neural Networks

Figure credit: Bhiksha Raj

Next lecture:
Understanding and Visualizing ConvNets