

Indian Institute of Information Technology, Allahabad



Convolutional Neural Networks

By

Dr. Shiv Ram Dubey

Computer Vision and Biometrics Lab

Department of Information Technology

Indian Institute of Information Technology, Allahabad



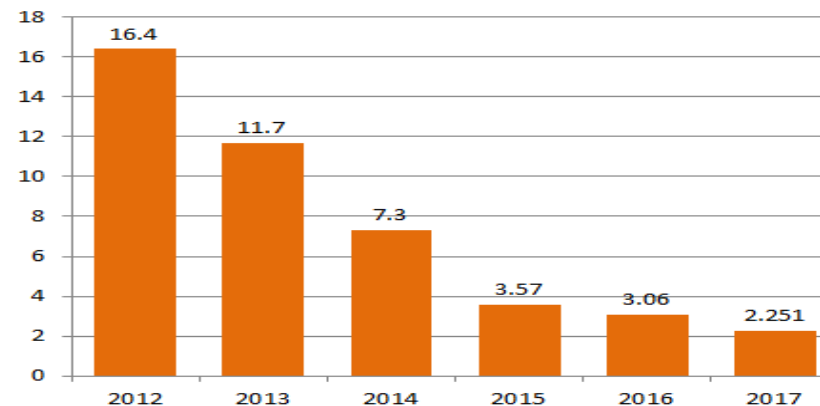
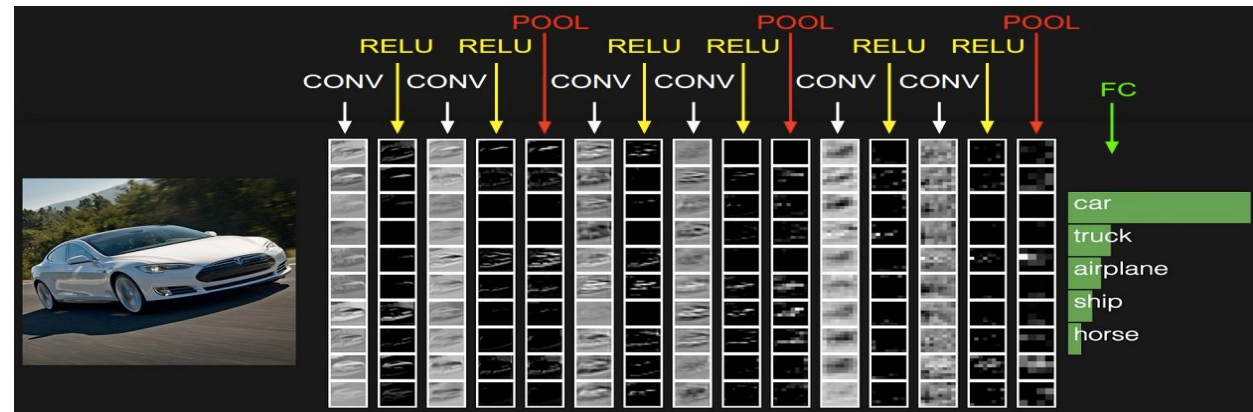
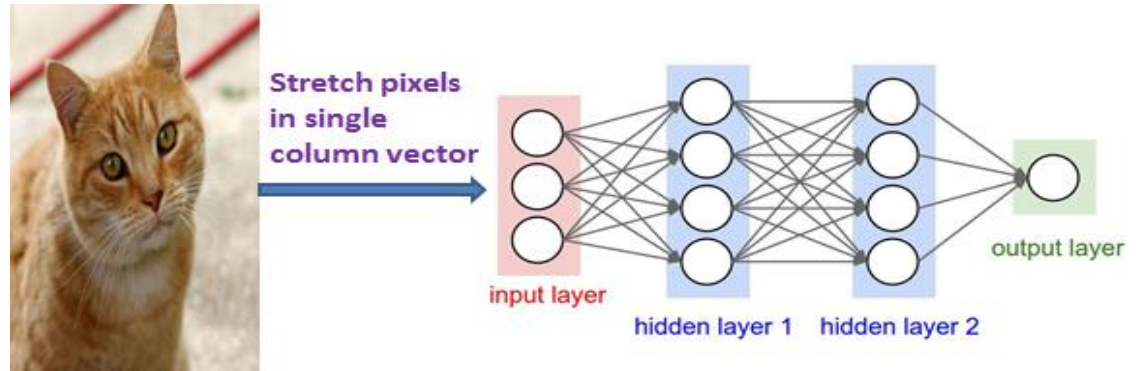
DISCLAIMER

The content (text, image, and graphics) used in this slide are adopted from many sources for Academic purposes. Broadly, the sources have been given due credit appropriately. However, there is a chance of missing out some original primary sources. The authors of this material do not claim any copyright of such material.

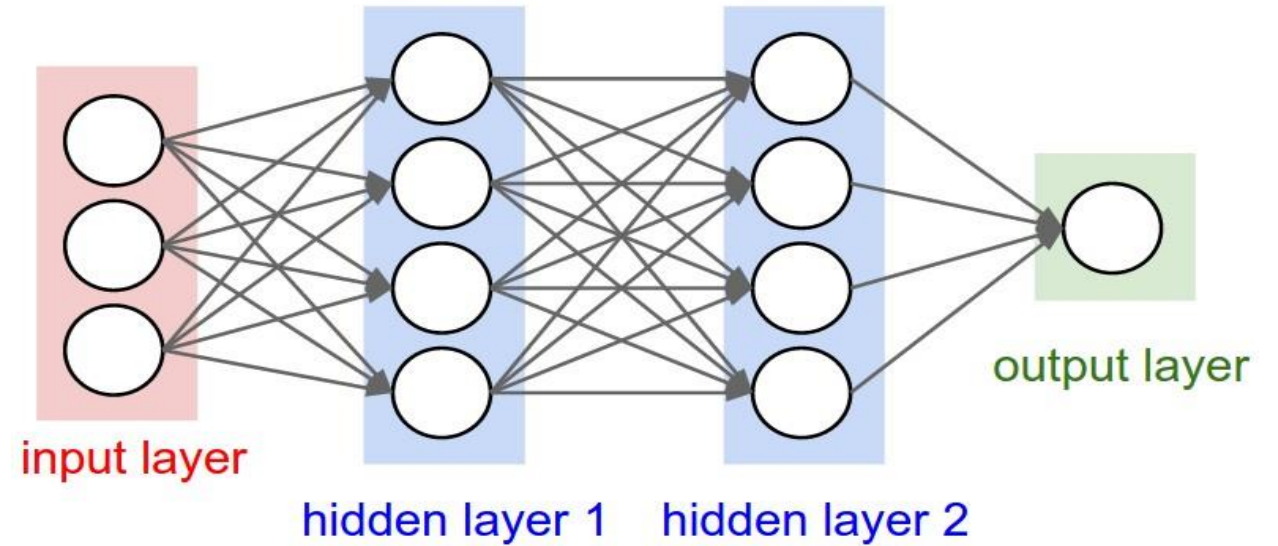


THIS SESSION

- **Neural Network and Image**
 - Dimensionality
 - Local relationship
- **Convolutional Neural Network (CNN)**
 - Convolution Layer
 - Non-linearity Layer
 - Pooling Layer
 - Fully Connected Layer
 - Classification Layer
- **ImageNet Challenge**
 - Progress
 - Human Level Performance
- **Transfer Learning**



MULTI-LAYER NEURAL NETWORK & IMAGE



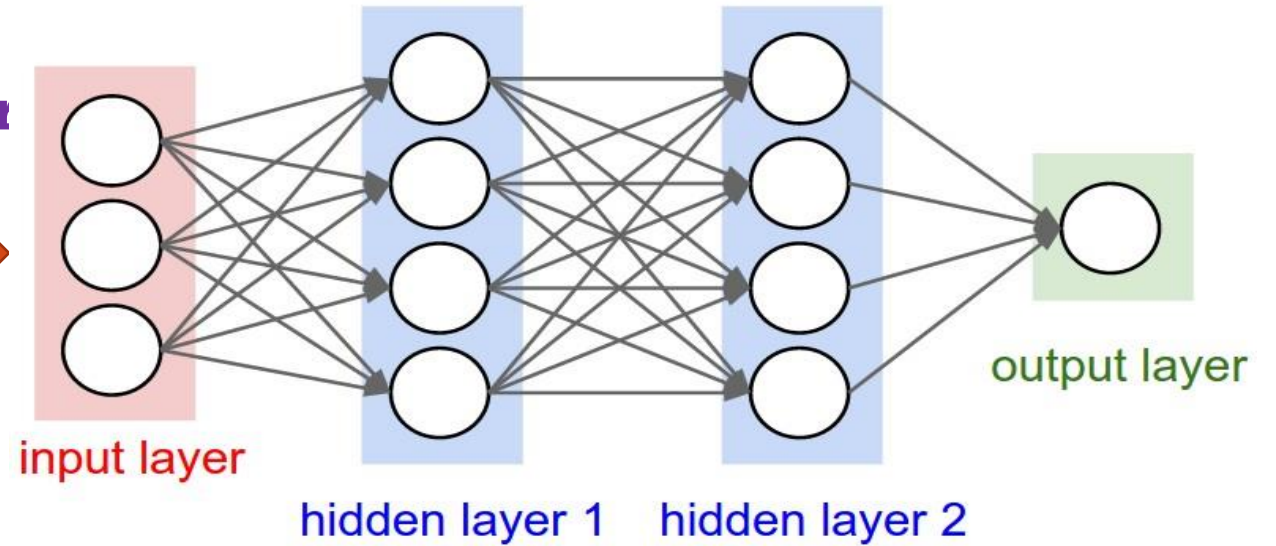
How to apply NN over Image?



MULTI-LAYER NEURAL NETWORK & IMAGE



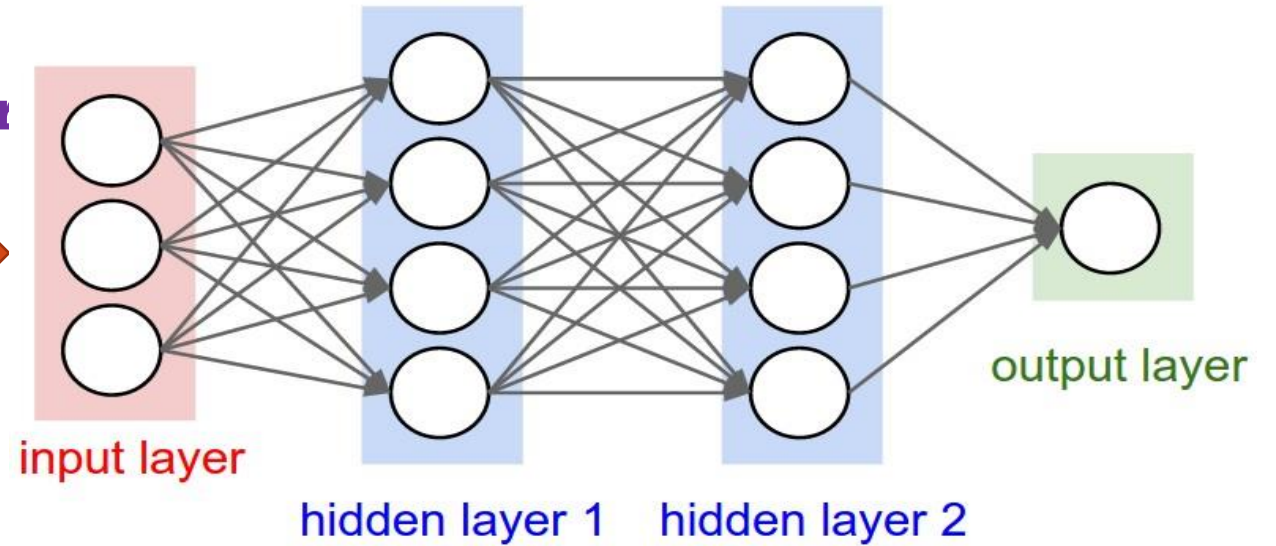
Stretch pixels in
single column vector



MULTI-LAYER NEURAL NETWORK & IMAGE



Stretch pixels in
single column vector



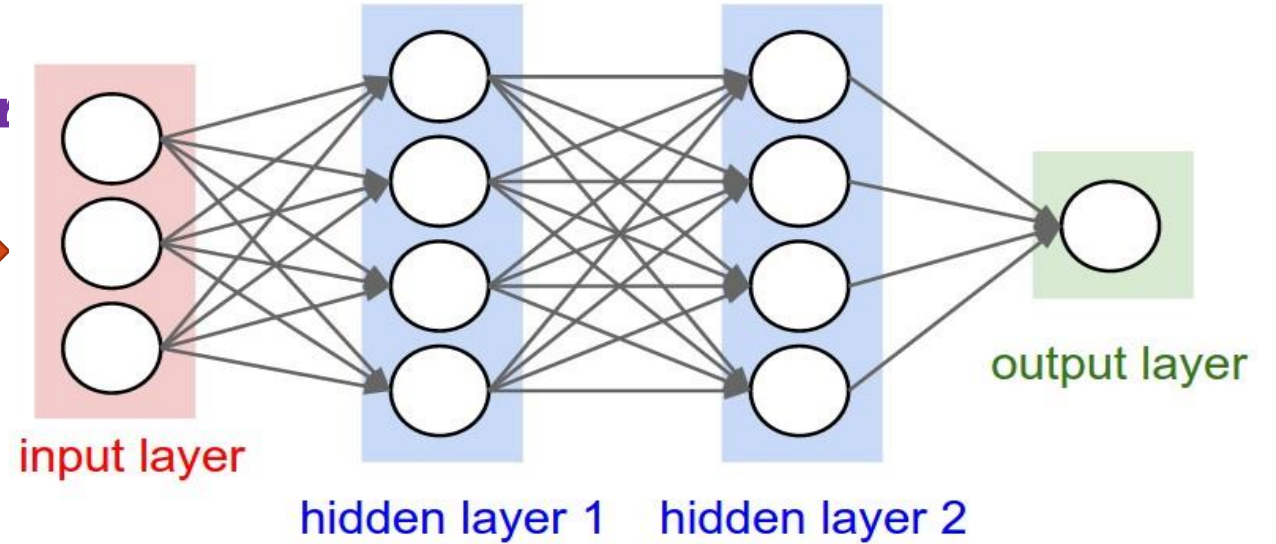
Problems ?



MULTI-LAYER NEURAL NETWORK & IMAGE



Stretch pixels in
single column vector



Problems:

High dimensionality

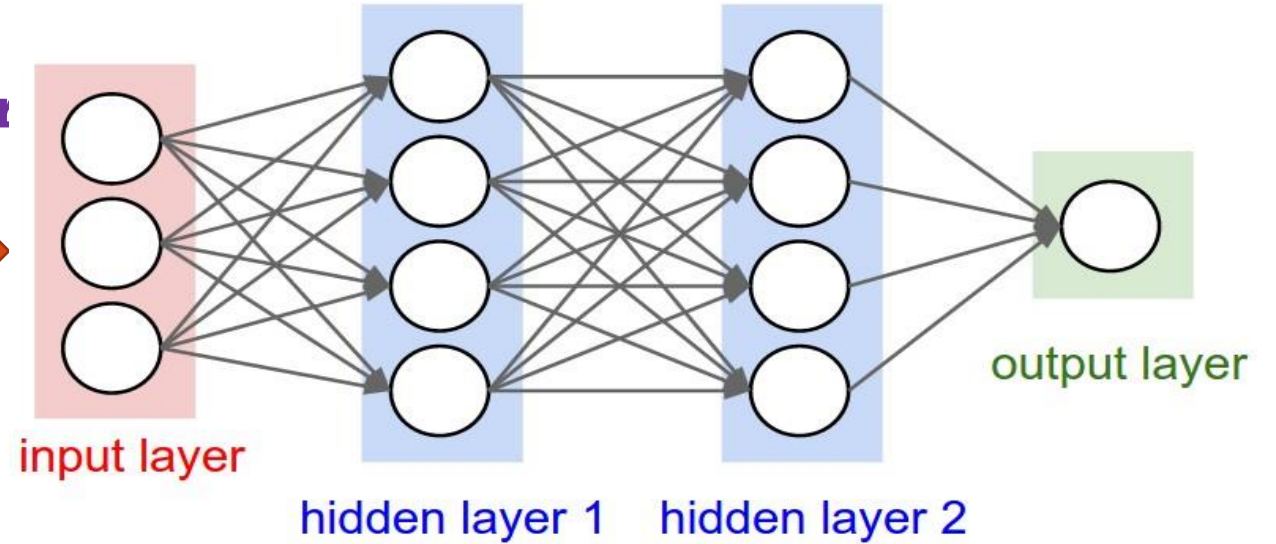
Local relationship



MULTI-LAYER NEURAL NETWORK & IMAGE



Stretch pixels in
single column vector



Problems:

High dimensionality

Local relationship

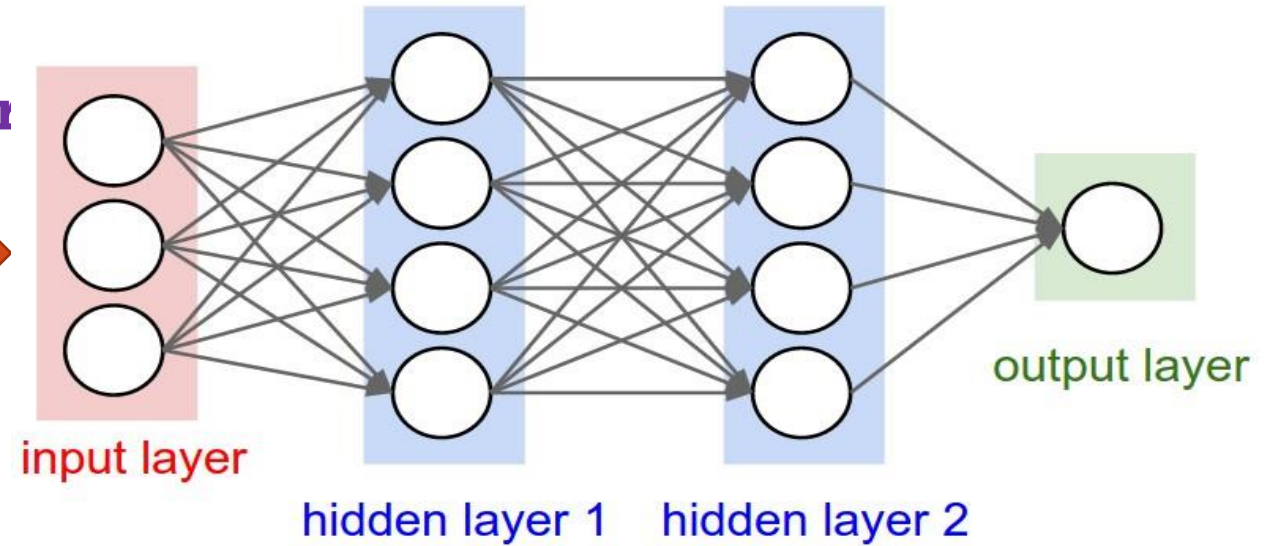
Solution ?



MULTI-LAYER NEURAL NETWORK & IMAGE



Stretch pixels in
single column vector



Problems:

High dimensionality

Local relationship

Solution:

Convolutional Neural Network

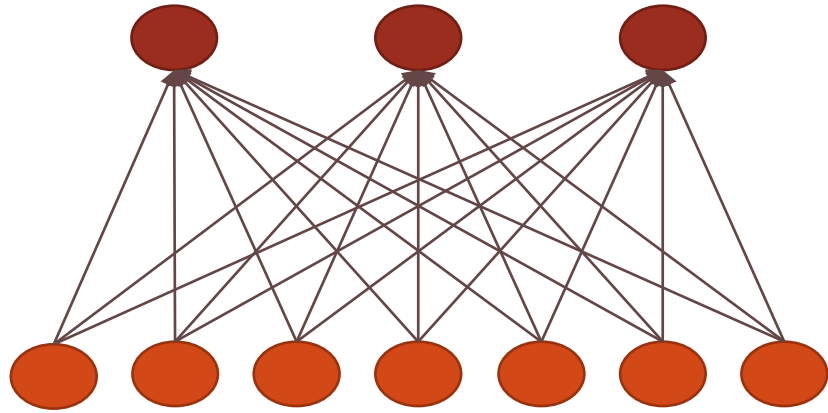


CONVOLUTIONAL NEURAL NETWORKS

- Also known as
 - CNN,
 - ConvNet,
 - DCN
- CNN = a multi-layer neural network with
 1. Local connectivity
 2. Weight sharing



CNN: LOCAL CONNECTIVITY

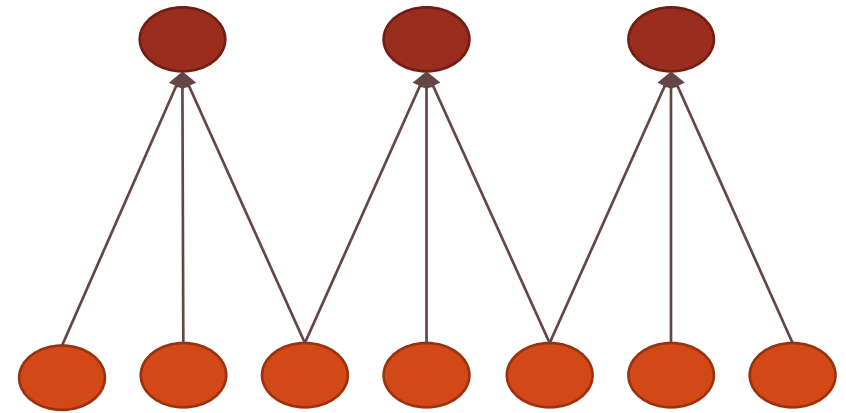


Hidden layer

Input layer

Global connectivity

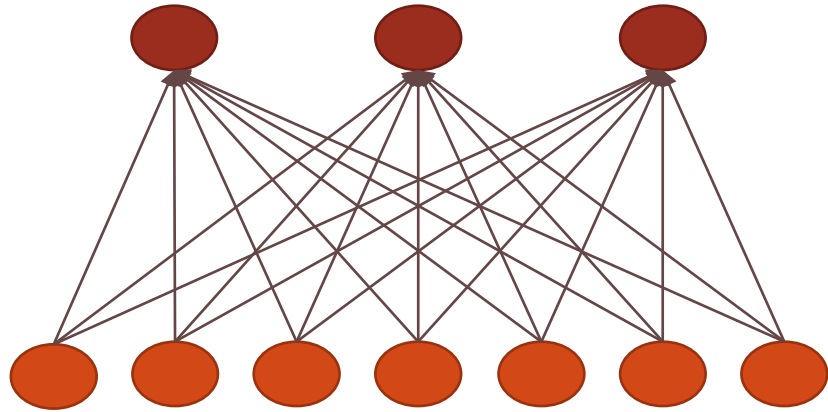
- # input units (neurons): 7
- # hidden units: 3



Local connectivity



CNN: LOCAL CONNECTIVITY

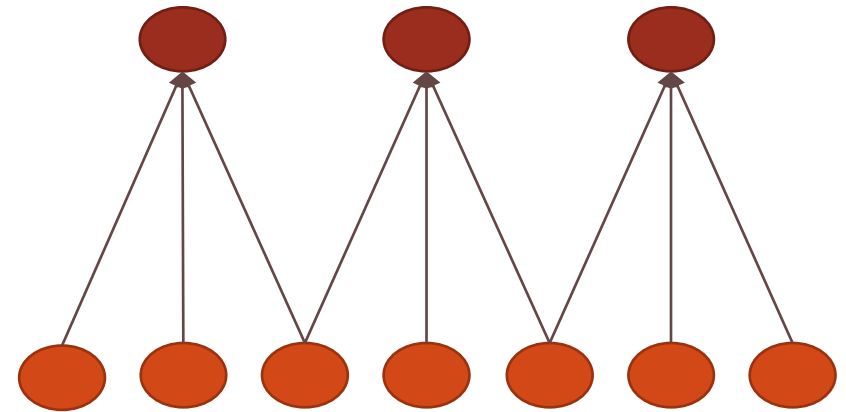


Hidden layer

Input layer

Global connectivity

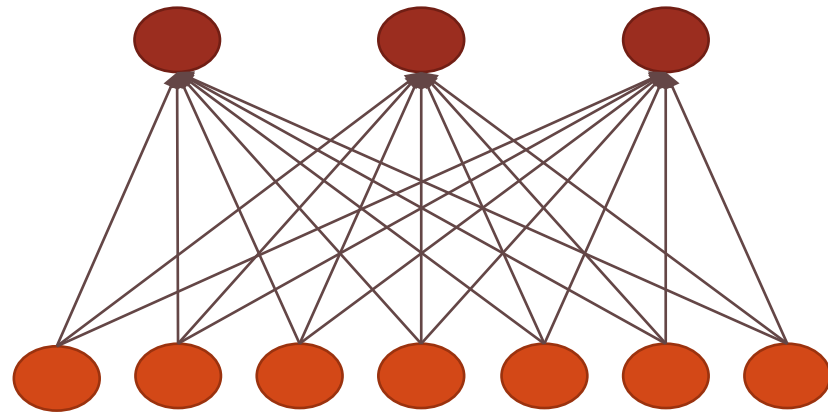
- # input units (neurons): 7
- # hidden units: 3
- **Number of parameters**
 - Global connectivity: ?
 - Local connectivity: ?



Local connectivity



CNN: LOCAL CONNECTIVITY

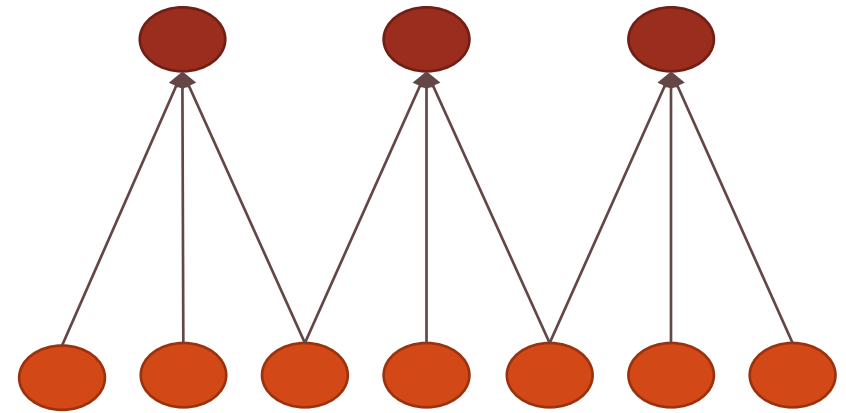


Hidden layer

Input layer

Global connectivity

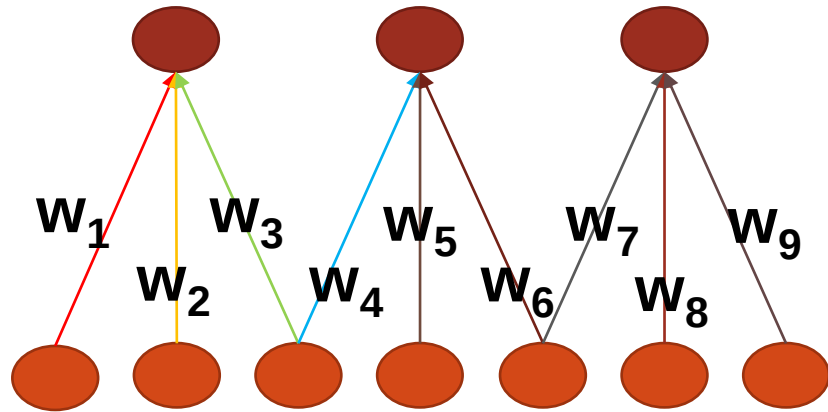
- # input units (neurons): 7
- # hidden units: 3
- **Number of parameters**
 - Global connectivity: $3 \times 7 = 21$
 - Local connectivity: $3 \times 3 = 9$



Local connectivity



CNN: WEIGHT SHARING

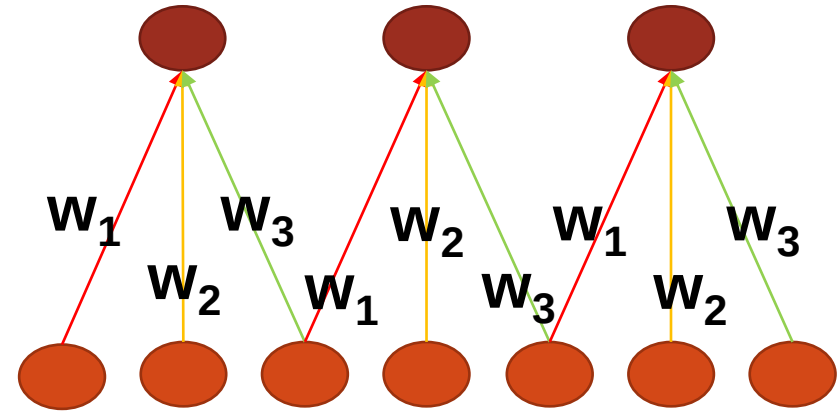


Without weight sharing

- # input units (neurons): 7
- # hidden units: 3

Hidden layer

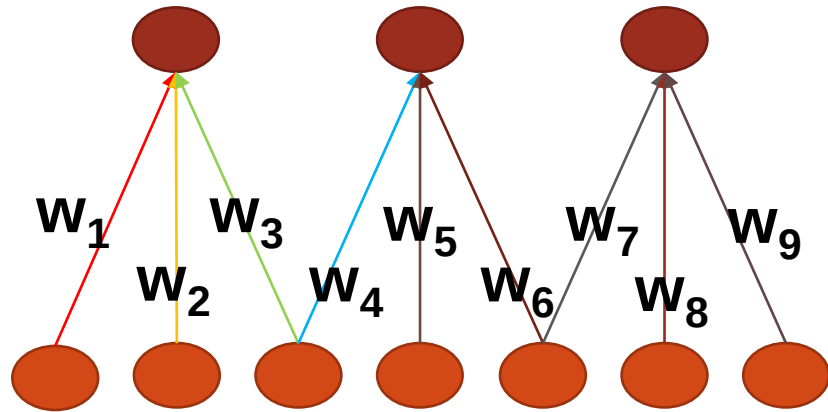
Input layer



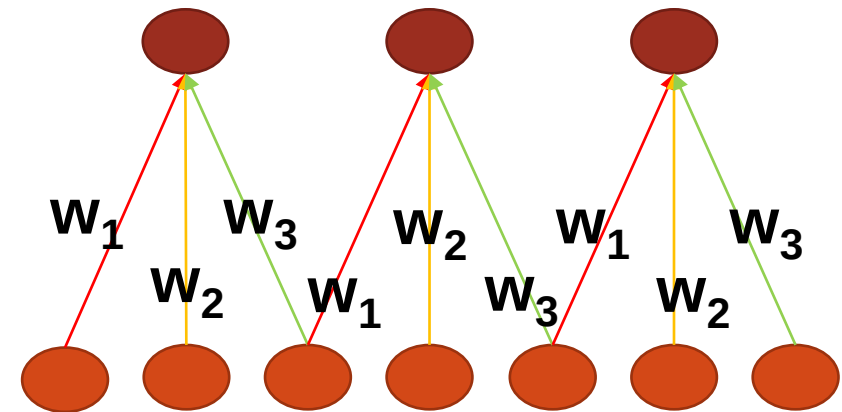
With weight sharing



CNN: WEIGHT SHARING



Without weight sharing

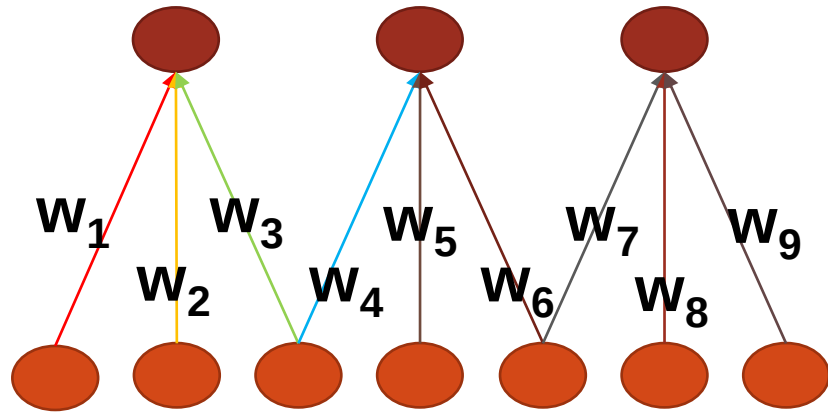


With weight sharing

- # input units (neurons): 7
- # hidden units: 3
- **Number of parameters**
 - Without weight sharing: ?
 - With weight sharing : ?



CNN: WEIGHT SHARING

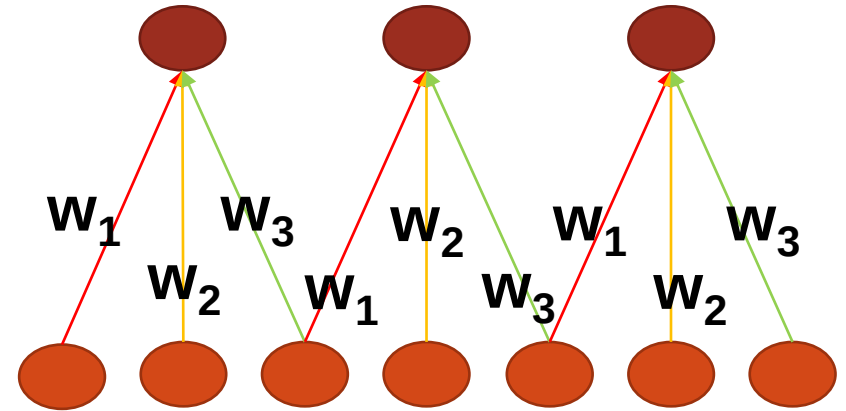


Without weight sharing

- # input units (neurons): 7
- # hidden units: 3
- **Number of parameters**
 - Without weight sharing: $3 \times 3 = 9$
 - With weight sharing : $3 \times 1 = 3$

Hidden layer

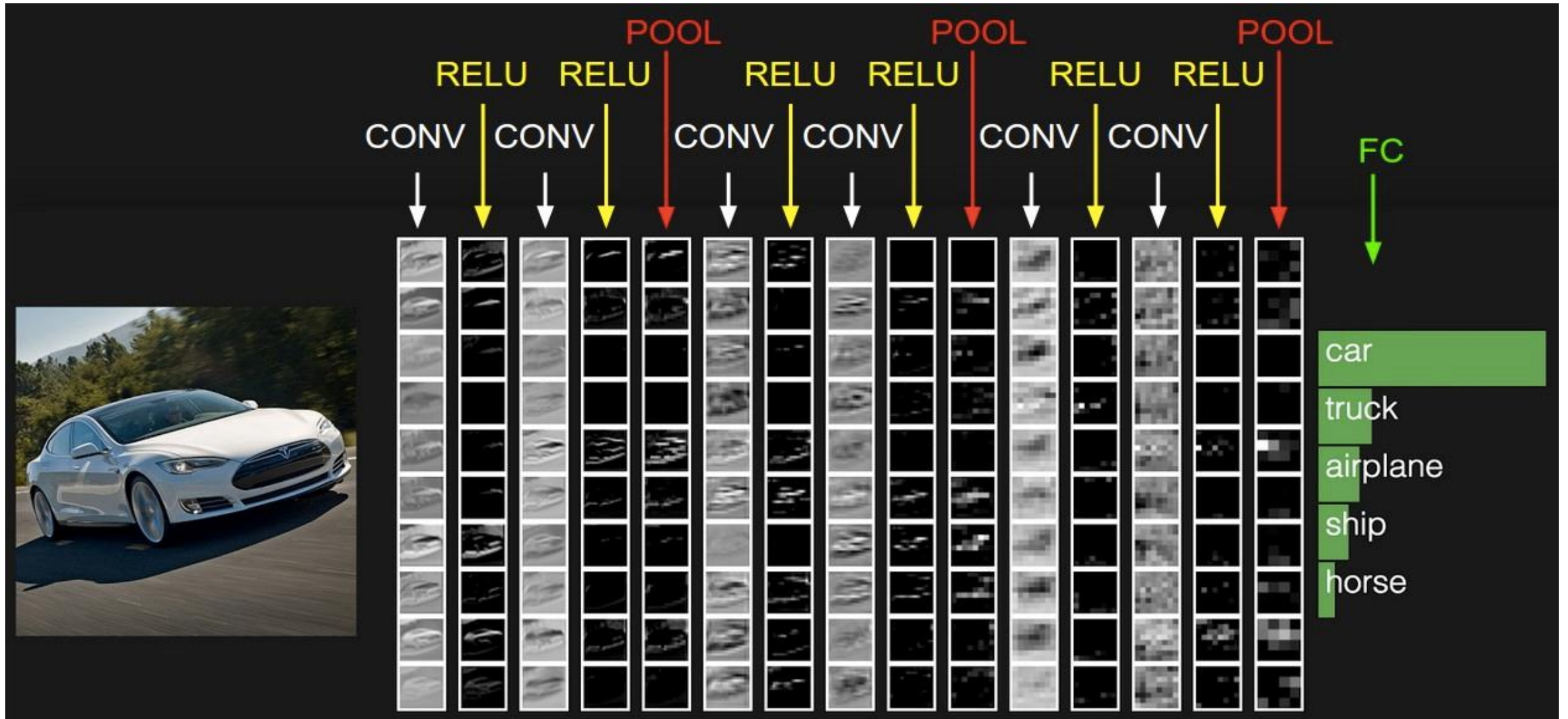
Input layer



With weight sharing



CONVOLUTIONAL NEURAL NETWORKS



LAYERS USED TO BUILD CONVNETS

Input Layer (Input image)

Convolutional Layer

Non-linearity Layer (such as Sigmoid, Tanh, ReLU, PReLU, ELU, Swish, etc.)

Pooling Layer (such as Max Pooling, Average Pooling, etc.)

Fully-Connected Layer

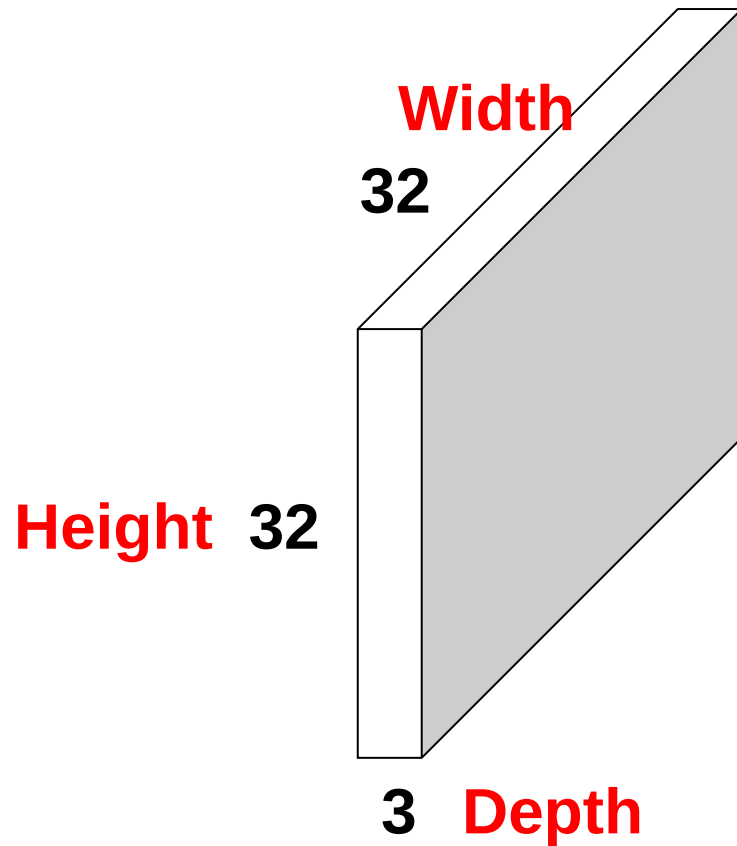
Classification Layer (Softmax, etc.)



CONVOLUTIONAL LAYER

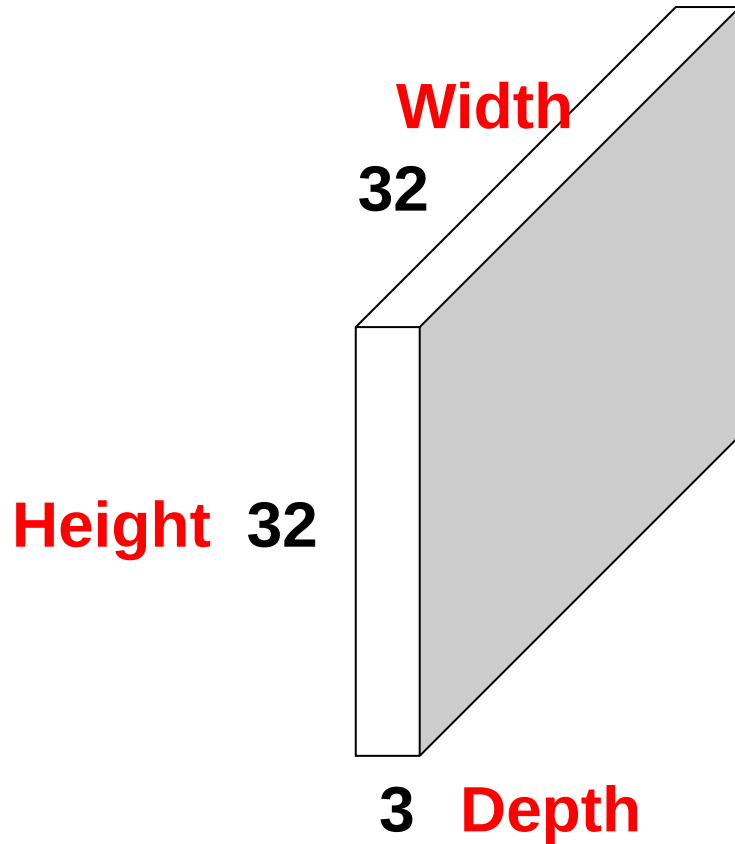
32×32×3 Image

-> preserve spatial structure

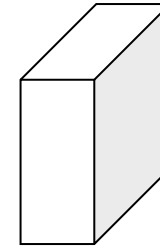


CONVOLUTIONAL LAYER

32×32×3 Image



5×5×3 Filter



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

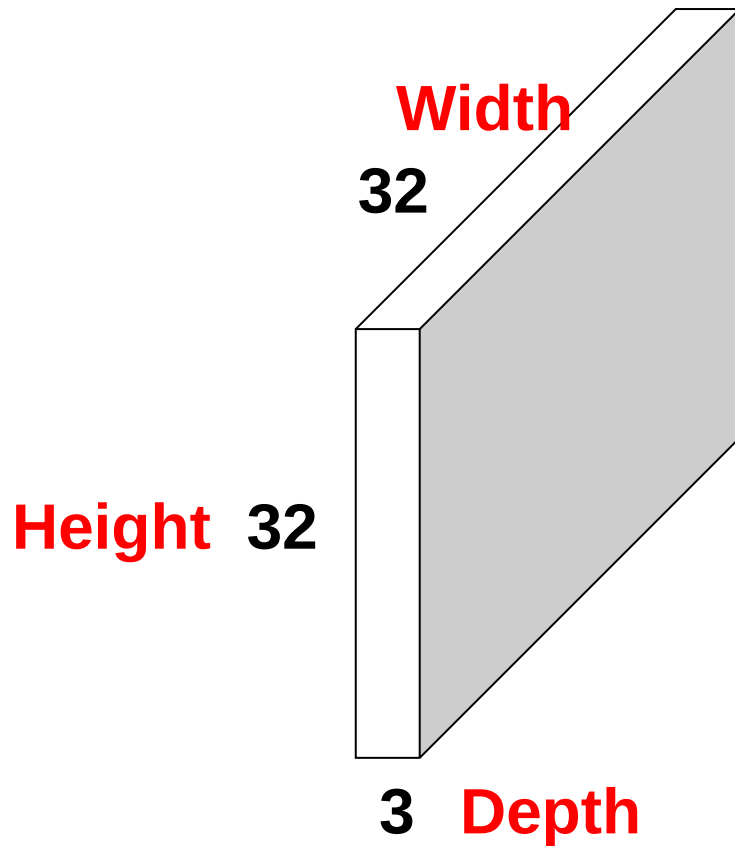


CONVOLUTIONAL LAYER

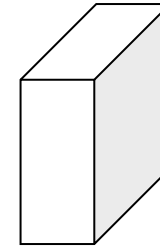
Handling multiple input channels

Filters always extend the full depth of the input volume

32×32×3 Image



5×5×3 Filter

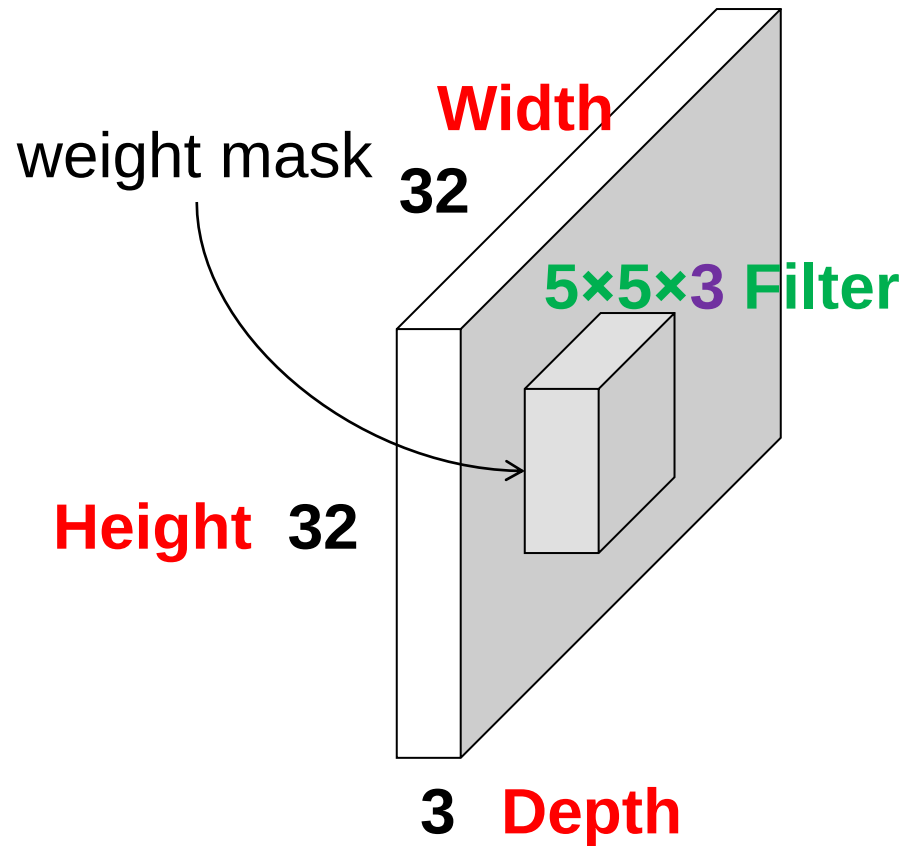


Convolve the filter with the image i.e. “slide over the image spatially, computing dot products”



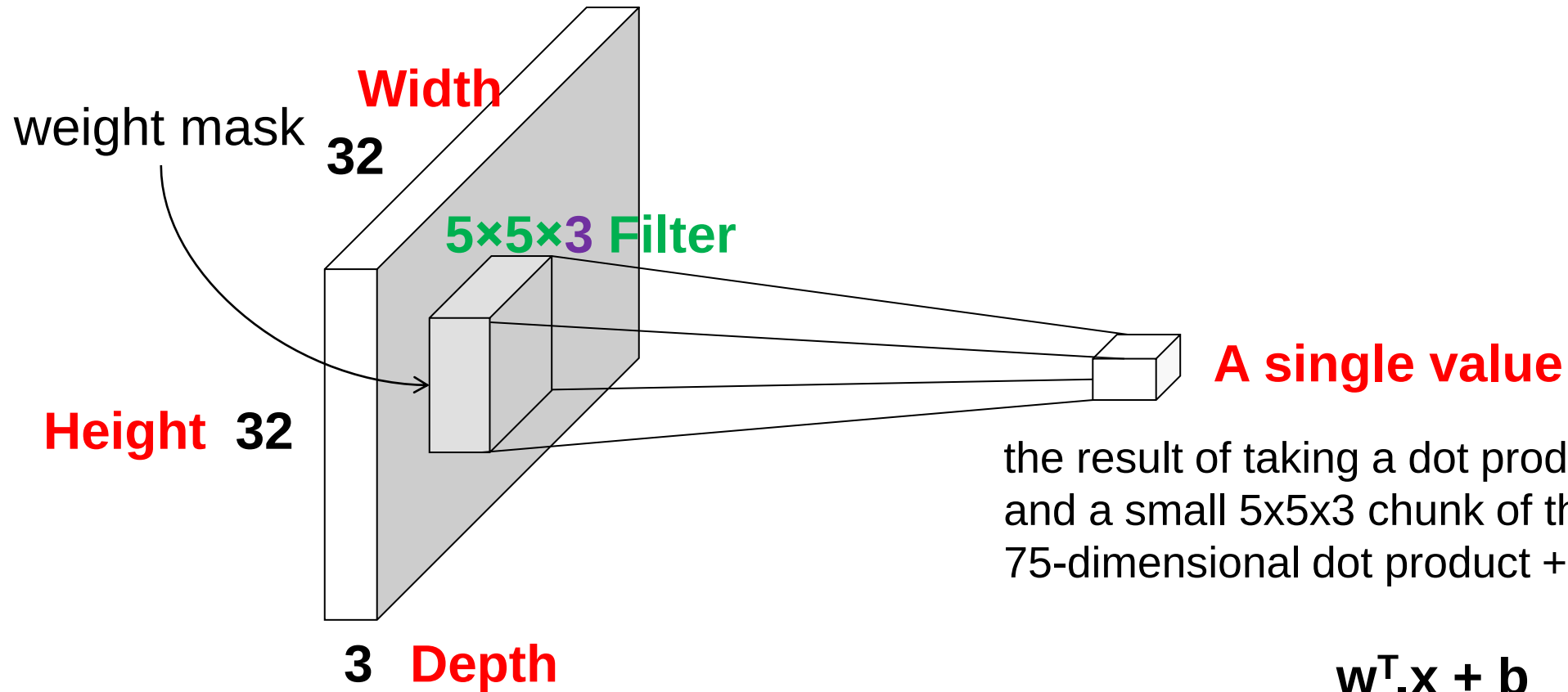
CONVOLUTIONAL LAYER

32×32×3 Image



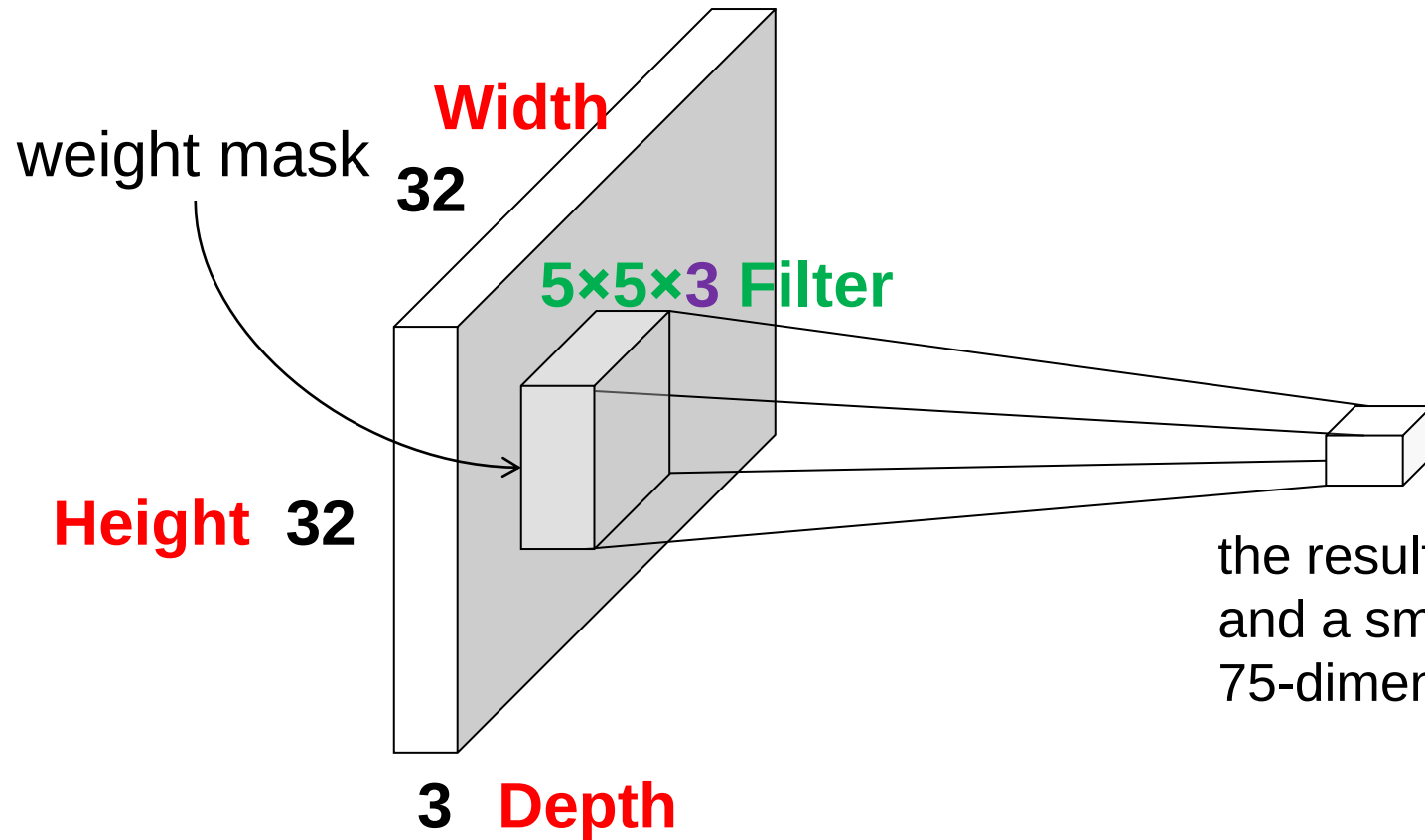
CONVOLUTIONAL LAYER

32×32×3 Image



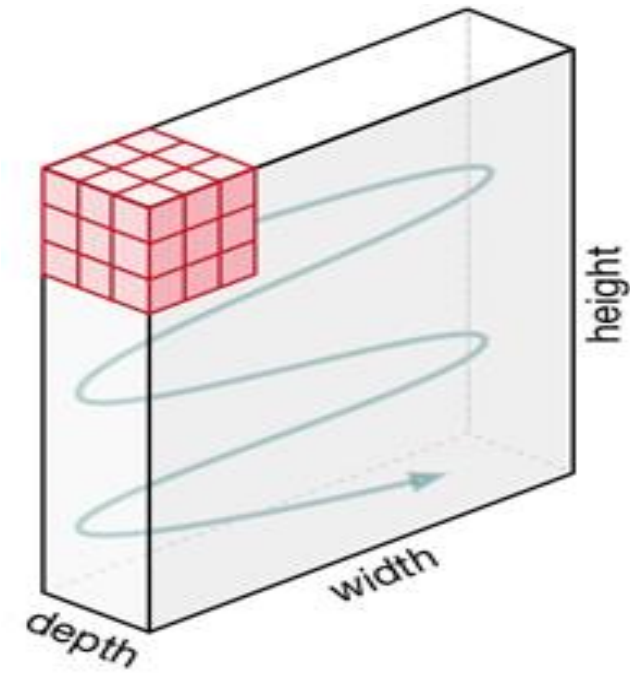
CONVOLUTIONAL LAYER

32×32×3 Image

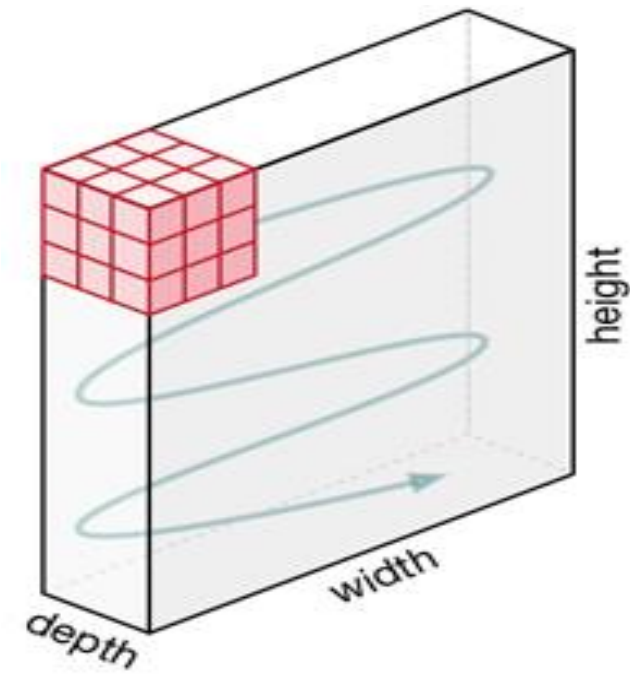
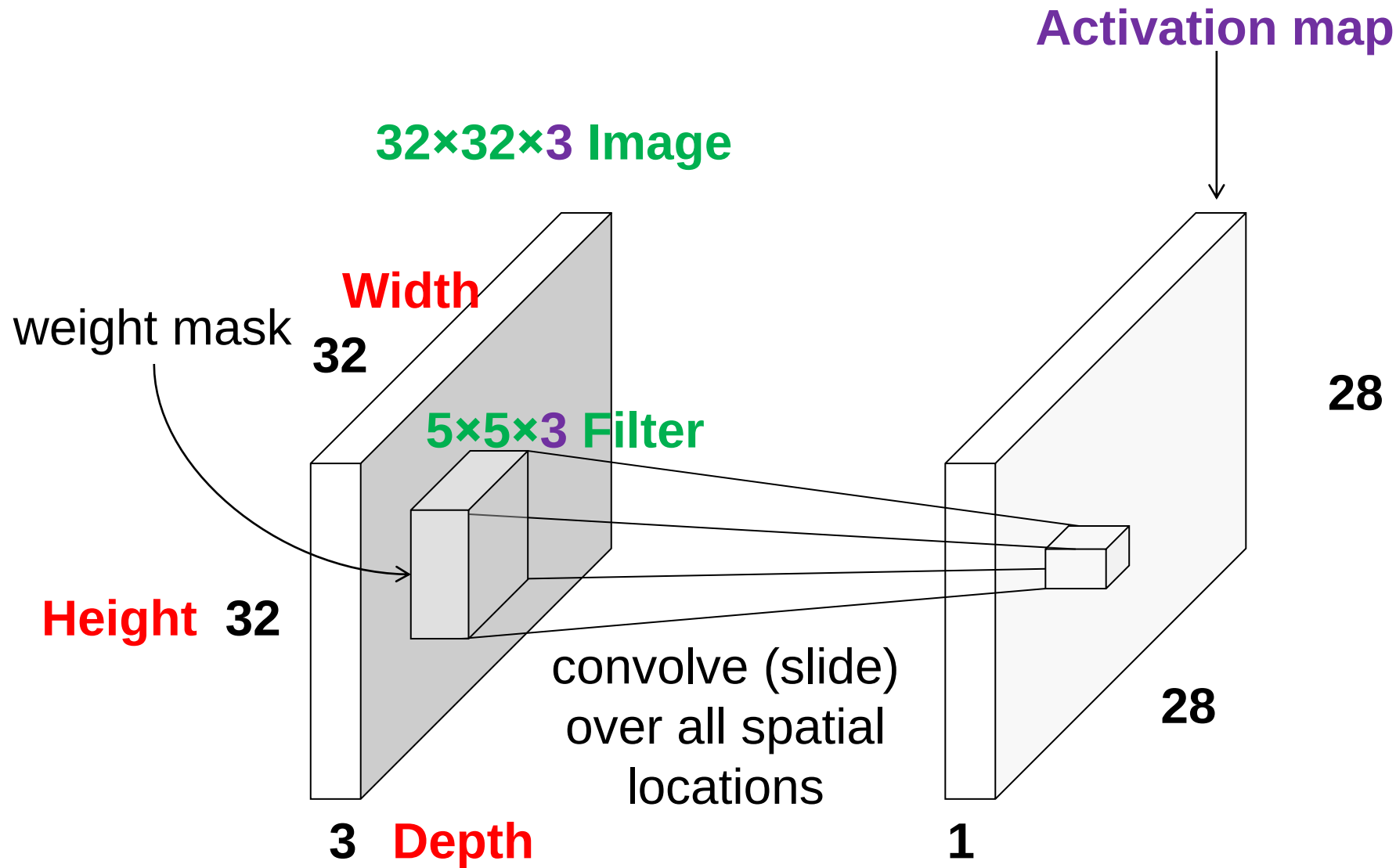


the result of taking a dot product between the filter and a small 5x5x3 chunk of the image (i.e. $5 \times 5 \times 3 = 75$ -dimensional dot product + bias)

$$\mathbf{w}^T \cdot \mathbf{x} + \mathbf{b}$$



CONVOLUTIONAL LAYER

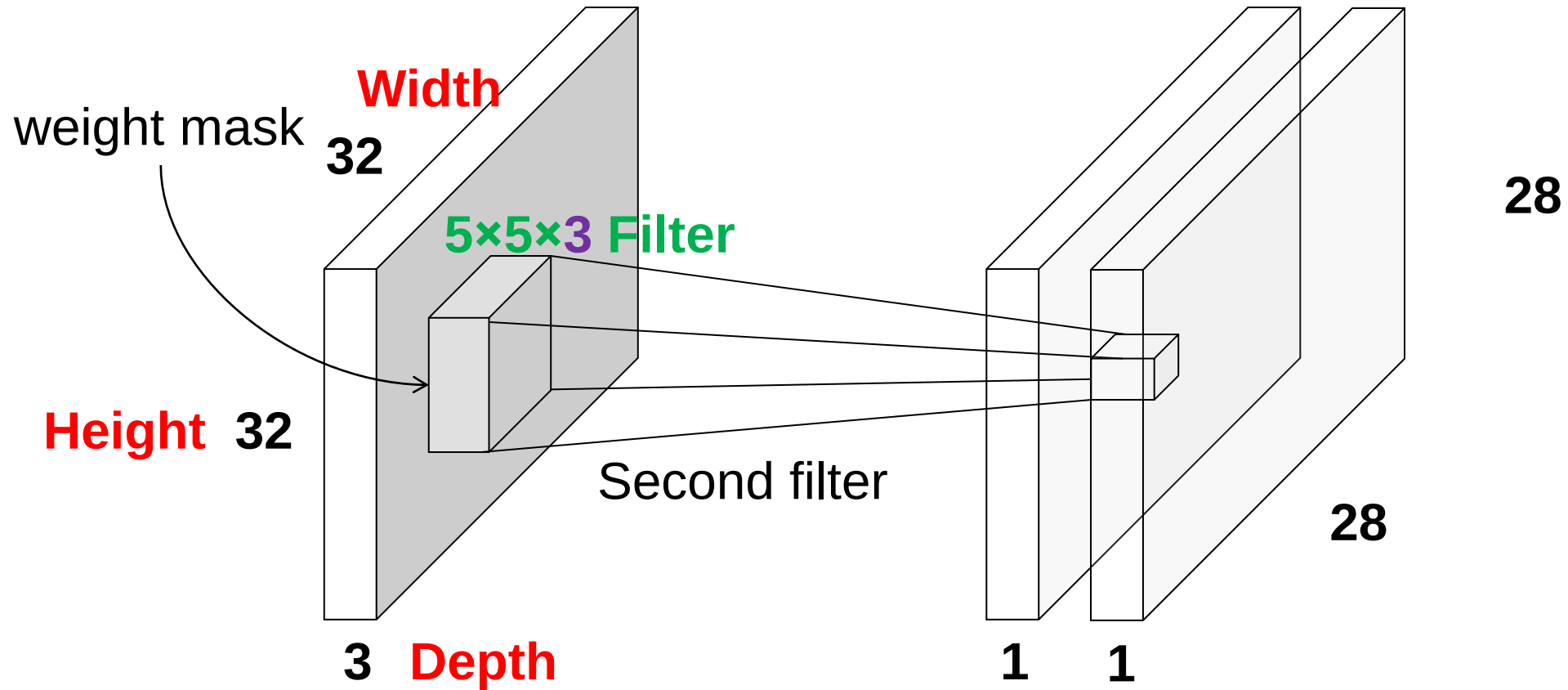


CONVOLUTIONAL LAYER

Handling multiple output maps

32×32×3 Image

Activation maps

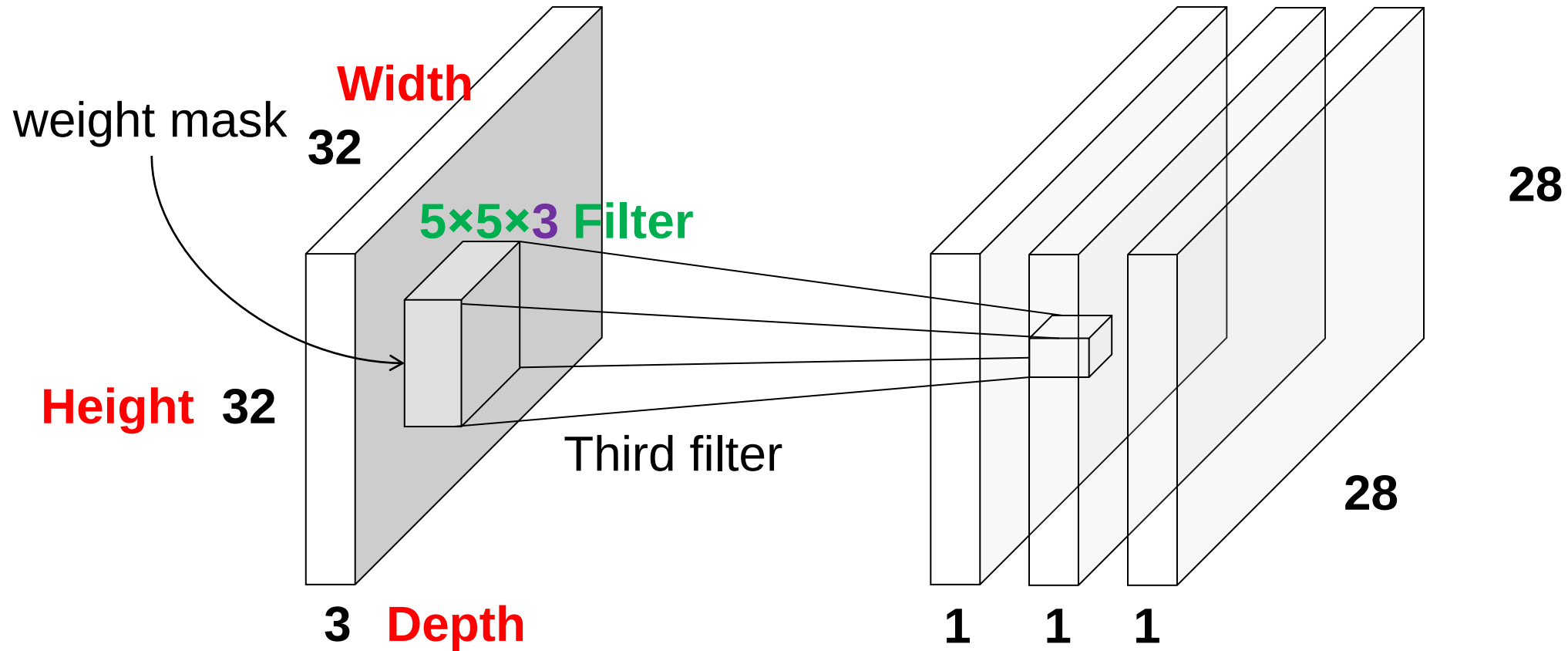


CONVOLUTIONAL LAYER

Handling multiple output maps

32×32×3 Image

Activation maps

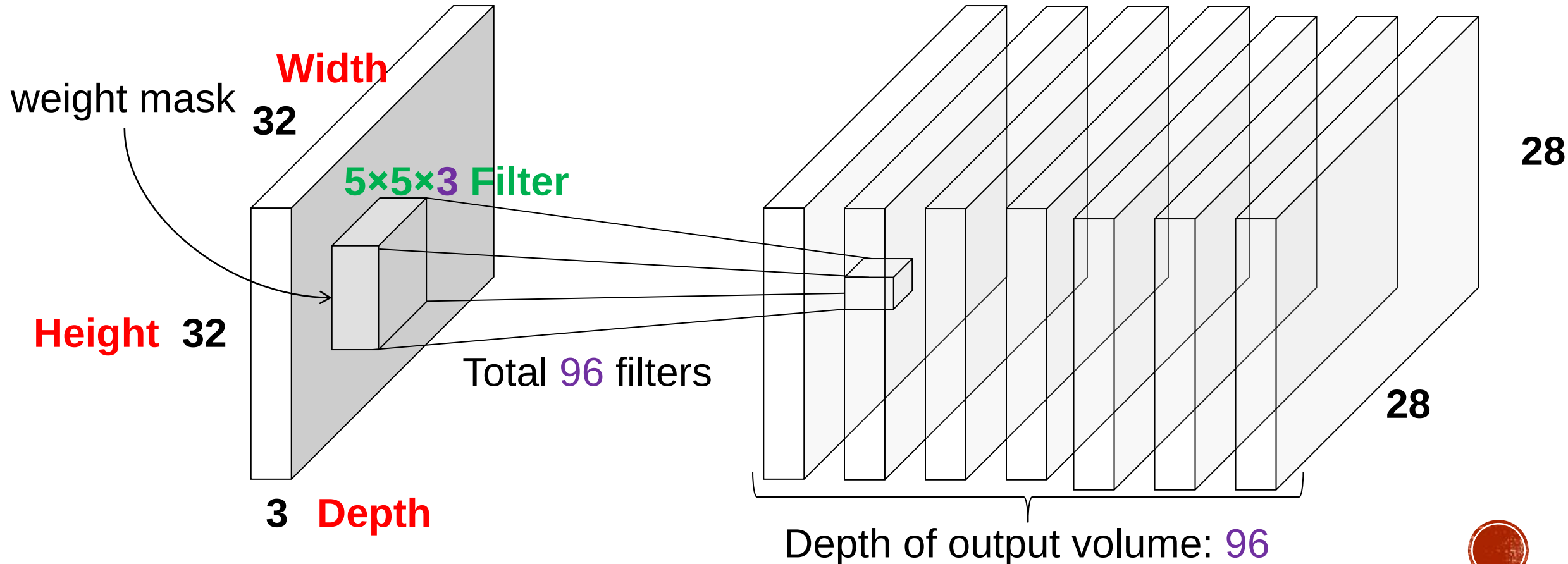


CONVOLUTIONAL LAYER

Handling multiple output maps

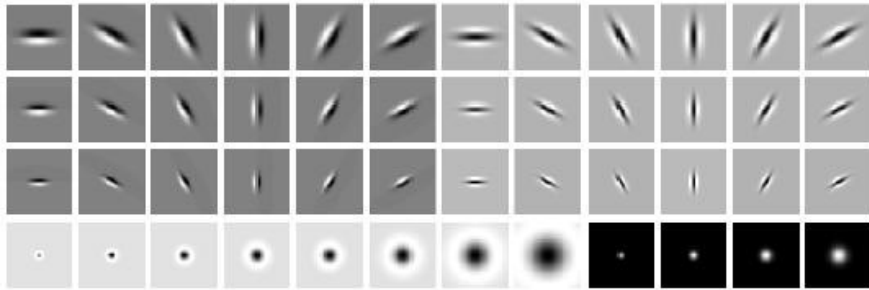
32×32×3 Image

Activation maps

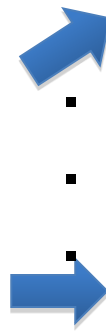


CONVOLUTION AND TRADITIONAL FEATURE EXTRACTION

bank of K filters



K feature maps



image

feature map



Input Volume (+pad 1) (7x7x3)

$x[:, :, 0]$

0	0	0	0	0	0	0
0	0	0	2	0	2	0
0	1	2	2	2	0	0
0	0	2	1	0	0	0
0	0	2	0	2	1	0
0	1	0	2	2	0	0
0	0	0	0	0	0	0

$x[:, :, 1]$

0	0	0	0	0	0	0
0	1	0	0	2	0	0
0	1	0	1	0	1	0
0	0	0	2	1	2	0
0	1	2	1	0	0	0
0	1	2	2	0	2	0
0	0	0	0	0	0	0

$x[:, :, 2]$

0	0	0	0	0	0	0
0	0	2	2	1	0	0
0	0	2	0	0	1	0
0	1	1	1	2	1	0
0	2	0	1	2	2	0
0	2	1	1	0	0	0
0	0	0	0	0	0	0

Filter W0 (3x3x3)

$w0[:, :, 0]$

-1	0	-1
1	-1	1
-1	0	-1

$w0[:, :, 1]$

0	-1	1
1	1	1
-1	1	-1

$w0[:, :, 2]$

0	0	0
-1	1	-1
1	1	-1

Bias $b0$ (1x1x1)

$b0[:, :, 0]$

1

Filter W1 (3x3x3)

$w1[:, :, 0]$

0	1	1
-1	-1	-1
1	-1	1

$w1[:, :, 1]$

-1	0	-1
1	1	1
-1	1	-1

$w1[:, :, 2]$

0	1	0
0	0	-1
1	-1	0

Bias $b1$ (1x1x1)

$b1[:, :, 0]$

0

Output Volume (3x3x2)

$o[:, :, 0]$

-3	-1	0
-1	-8	2
3	0	3

$o[:, :, 1]$

1	4	2
-1	4	5
3	1	3

toggle movement



Input Volume (+pad 1) (7x7x3)

$x[:, :, 0]$						
0	0	0	0	0	0	0
0	0	0	2	0	2	0
0	1	2	2	2	0	0
0	0	2	1	0	0	0
0	0	2	0	2	1	0
0	1	0	2	2	0	0
0	0	0	0	0	0	0
$x[:, :, 1]$						
0	0	0	0	0	0	0
0	1	0	0	2	0	0
0	1	0	1	0	1	0
0	0	0	2	1	2	0
0	1	2	1	0	0	0
0	1	2	2	0	2	0
0	0	0	0	0	0	0
$x[:, :, 2]$						
0	0	0	0	0	0	0
0	0	2	2	1	0	0
0	0	2	0	0	1	0
0	1	1	1	2	1	0
0	2	0	1	2	2	0
0	2	1	1	0	0	0
0	0	0	0	0	0	0

Filter W0 (3x3x3)

$w0[:, :, 0]$		
-1	0	-1
1	-1	1
-1	0	-1
$w0[:, :, 1]$		
0	-1	1
1	1	1
-1	1	-1
$w0[:, :, 2]$		
0	0	0
-1	1	-1
1	1	-1
Bias $b0$ (1x1x1)		
$b0[:, :, 0]$		
1		

Filter W1 (3x3x3)

$w1[:, :, 0]$		
0	1	1
-1	-1	-1
1	-1	1
$w1[:, :, 1]$		
-1	0	-1
1	1	1
-1	1	-1
$w1[:, :, 2]$		
0	1	0
0	0	-1
1	-1	0
Bias $b1$ (1x1x1)		
$b1[:, :, 0]$		
0		

Output Volume (3x3x2)

$o[:, :, 0]$		
-3	-1	0
-1	-8	2
3	0	3
$o[:, :, 1]$		
1	4	2
-1	4	5
3	1	3

toggle movement



Input Volume (+pad 1) (7x7x3)

 $x[:, :, 0]$

0	0	0	0	0	0	0
0	0	0	2	0	2	0
0	1	2	2	2	0	0
0	0	2	1	0	0	0
0	0	2	0	2	1	0
0	1	0	2	2	0	0
0	0	0	0	0	0	0

 $x[:, :, 1]$

0	0	0	0	0	0	0
0	1	0	0	2	0	0
0	1	0	1	0	1	0
0	0	0	2	1	2	0
0	1	2	1	0	0	0
0	1	2	2	0	2	0
0	0	0	0	0	0	0

 $x[:, :, 2]$

0	0	0	0	0	0	0
0	0	2	2	1	0	0
0	0	2	0	0	1	0
0	1	1	1	2	1	0
0	2	0	1	2	2	0
0	2	1	1	0	0	0
0	0	0	0	0	0	0

Filter W0 (3x3x3)

 $w0[:, :, 0]$

-1	0	-1
1	-1	1
-1	0	-1

 $w0[:, :, 1]$

0	-1	1
1	1	1
-1	1	-1

 $w0[:, :, 2]$

0	0	0
-1	1	-1
1	1	-1

Bias b0 (1x1x1)

 $b0[:, :, 0]$

1

Filter W1 (3x3x3)

 $w1[:, :, 0]$

0	1	1
-1	-1	-1
1	-1	1

 $w1[:, :, 1]$

-1	0	-1
1	1	1
-1	1	-1

 $w1[:, :, 2]$

0	1	0
0	0	-1
1	-1	0

Bias b1 (1x1x1)

 $b1[:, :, 0]$

0

Output Volume (3x3x2)

 $o[:, :, 0]$

-3	-1	0
-1	-8	2
3	0	3

 $o[:, :, 1]$

1	4	2
-1	4	5
3	1	3

toggle movement

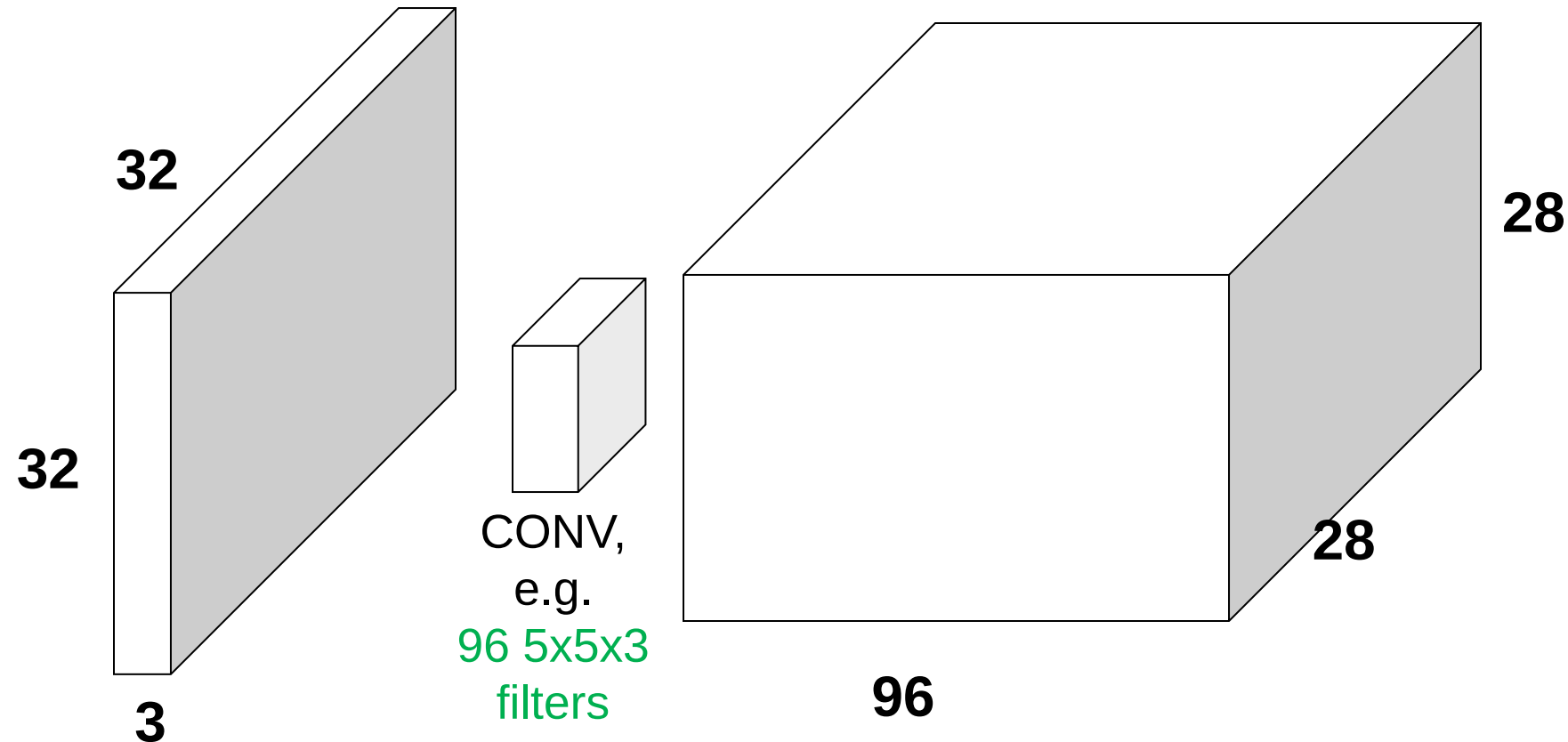


CONVOLUTIONAL LAYER

Preview: ConvNet is a sequence of Convolution Layers, interspersed with activation functions

32×32×3 Image

Activation maps

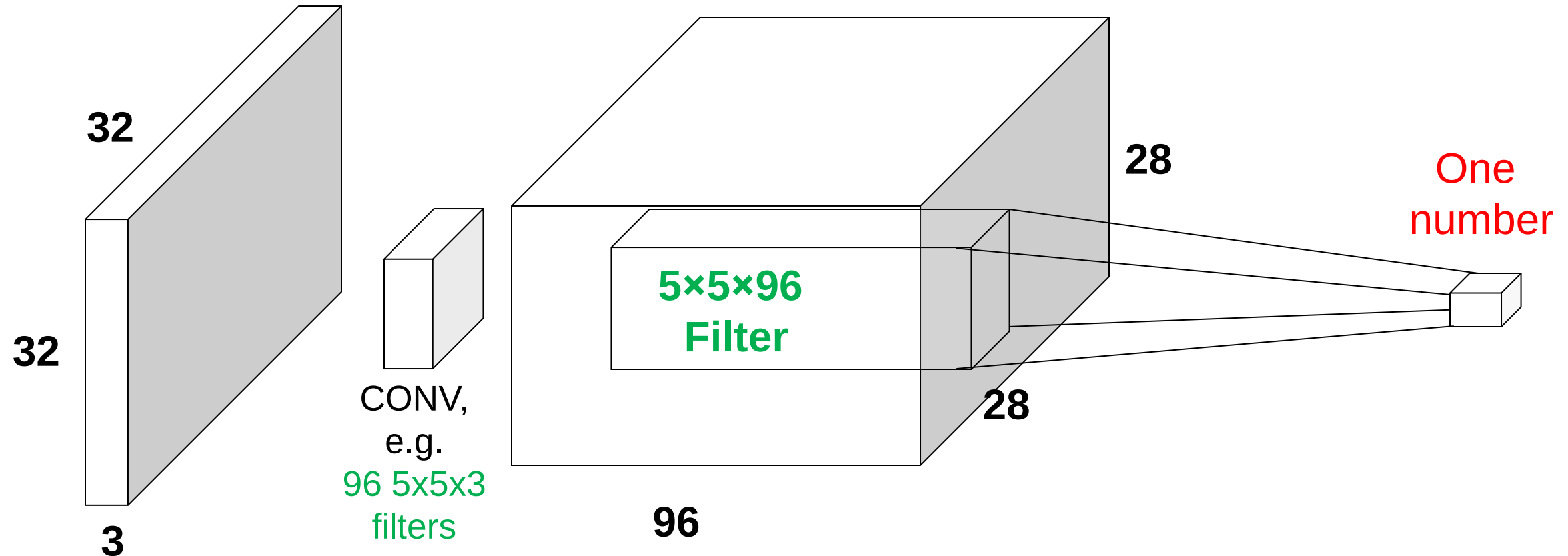


CONVOLUTIONAL LAYER

Preview: ConvNet is a sequence of Convolution Layers, interspersed with activation functions

32×32×3 Image

Activation maps



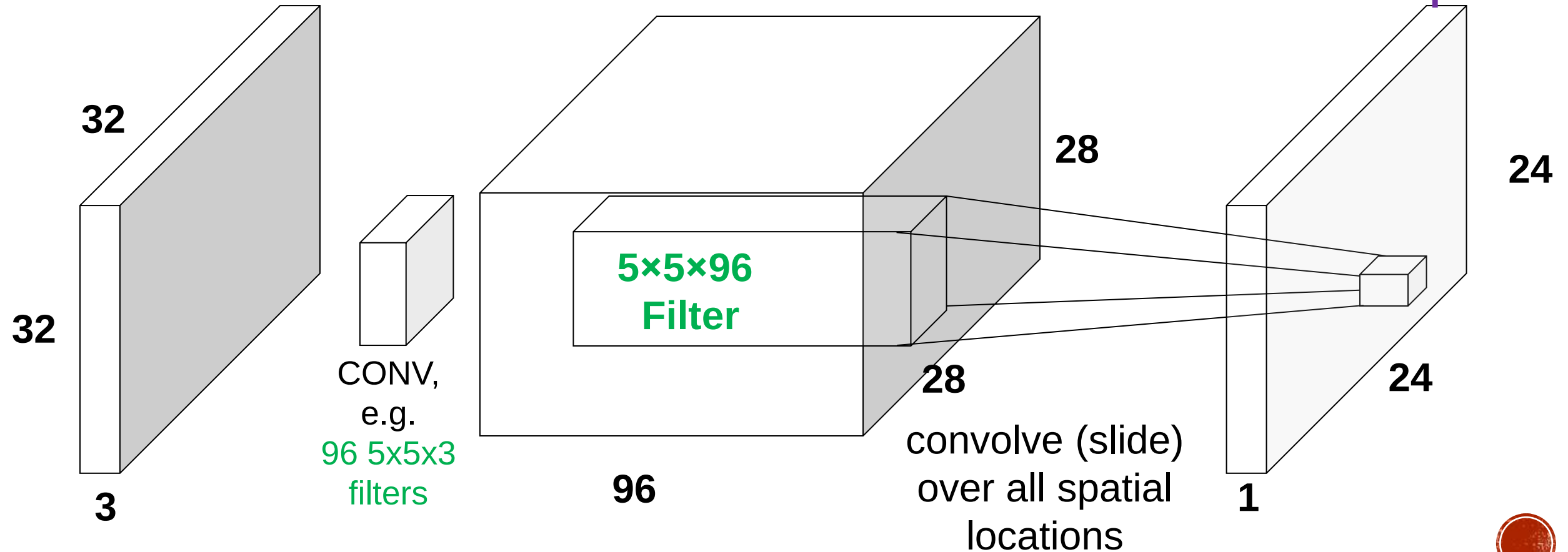
CONVOLUTIONAL LAYER

Preview: ConvNet is a sequence of Convolution Layers, interspersed with activation functions

32×32×3 Image

Activation maps

Deeper activation map



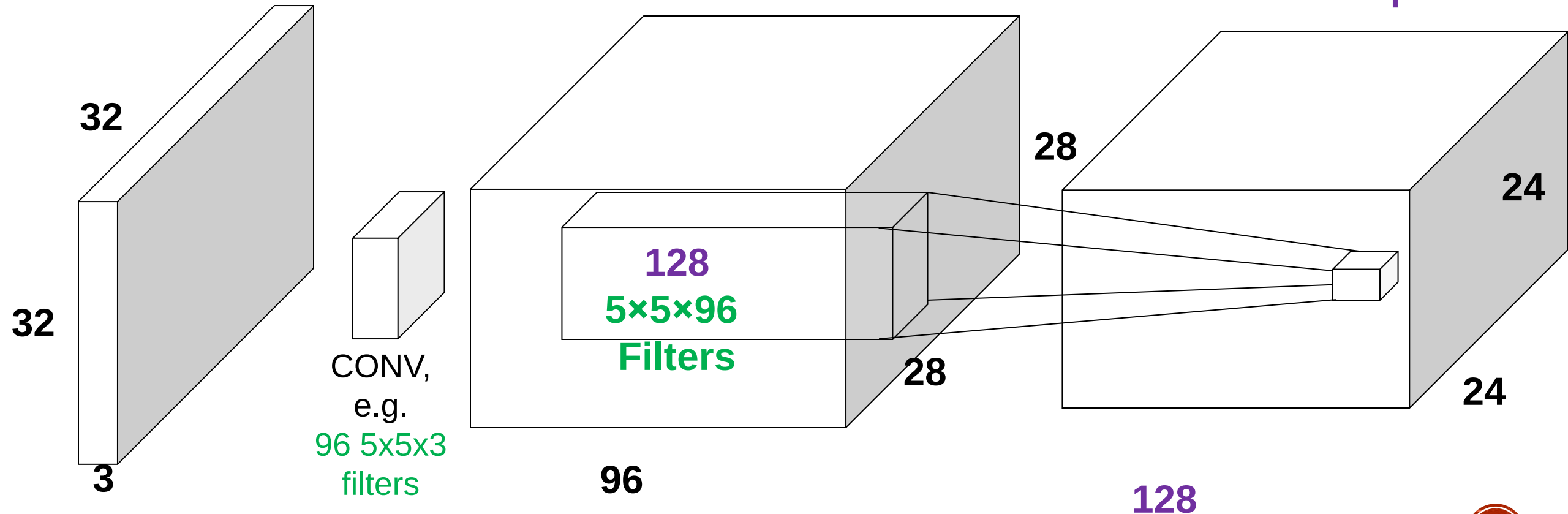
CONVOLUTIONAL LAYER

Preview: ConvNet is a sequence of Convolution Layers, interspersed with activation functions

32×32×3 Image

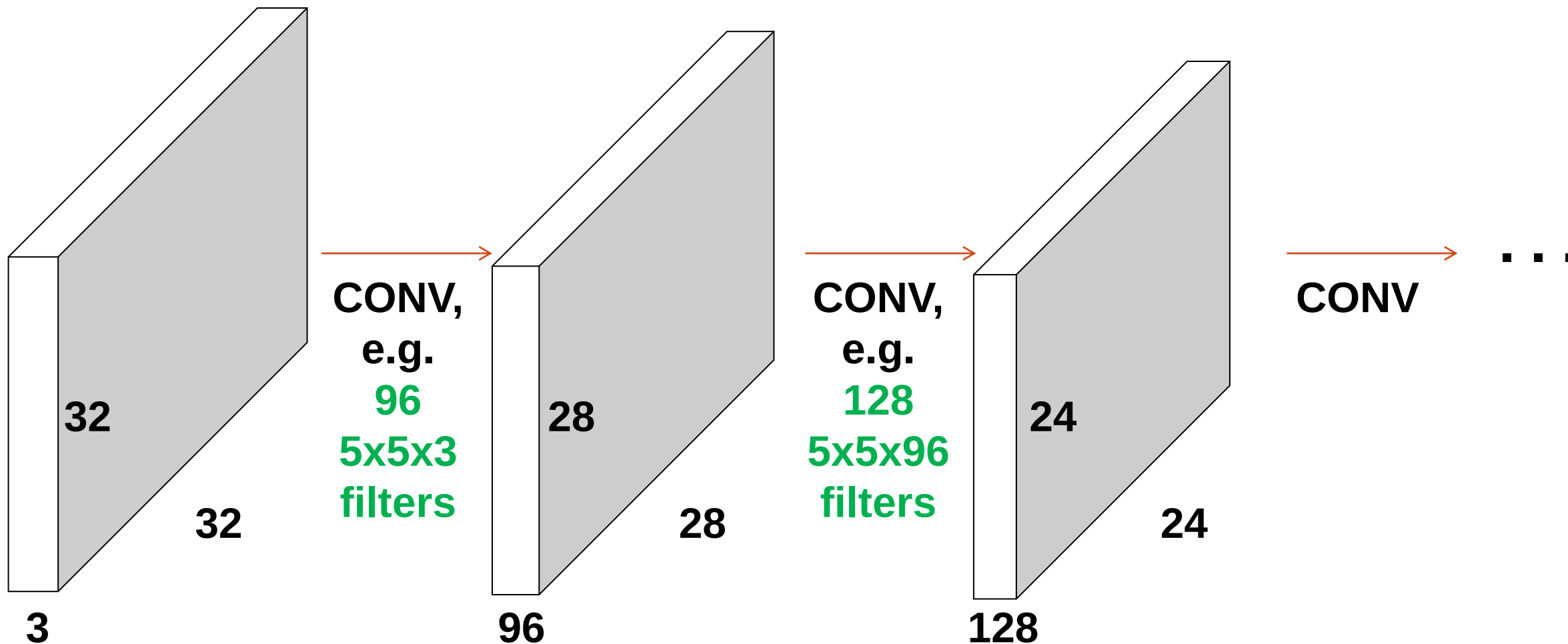
Activation maps

Deeper activation maps



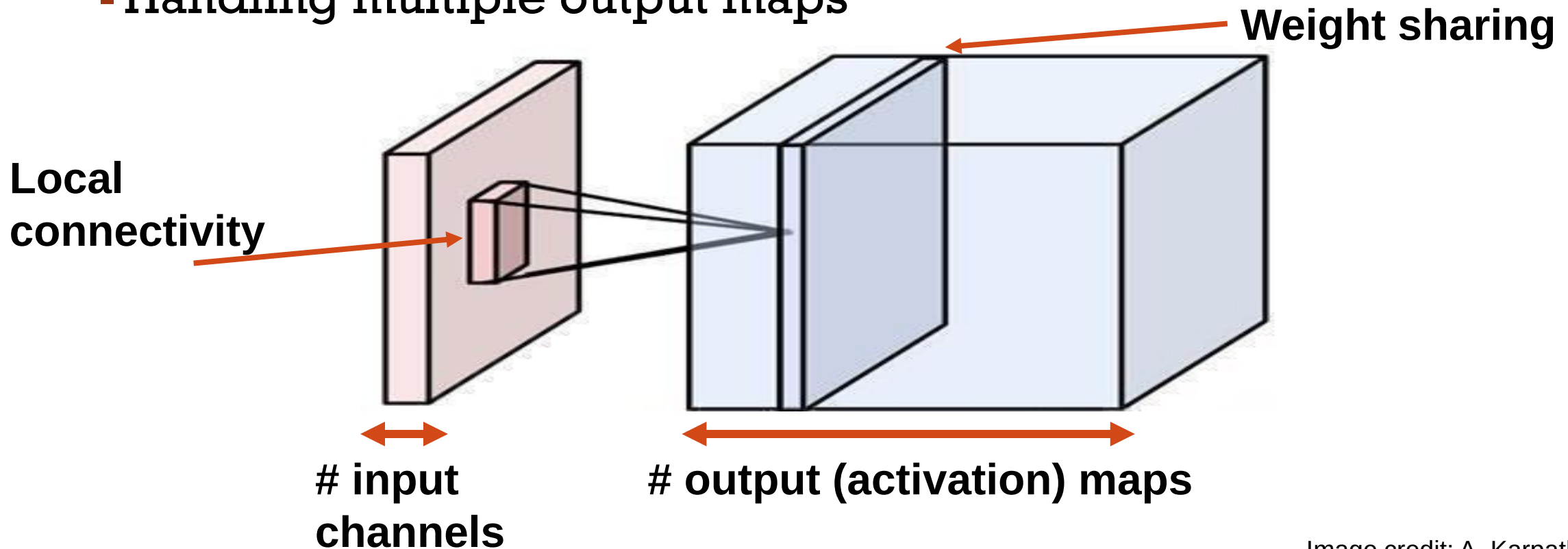
MULTILAYER CONVOLUTION

Preview: ConvNet is a sequence of Convolution Layers, interspersed with activation functions



ANY CONVOLUTION LAYER

- Local connectivity
- Weight sharing
- Handling multiple input channels
- Handling multiple output maps



A CLOSER LOOK AT SPATIAL DIMENSIONS

7

7×7 input (spatially)
assume 3×3 filter

7



A CLOSER LOOK AT SPATIAL DIMENSIONS

7

7×7 input (spatially)
assume 3×3 filter

7



A CLOSER LOOK AT SPATIAL DIMENSIONS

7

7×7 input (spatially)
assume 3×3 filter

7



A CLOSER LOOK AT SPATIAL DIMENSIONS

7

7×7 input (spatially)
assume 3×3 filter

7



A CLOSER LOOK AT SPATIAL DIMENSIONS

7

7×7 input (spatially)
assume 3×3 filter

7



5×5 output



A CLOSER LOOK AT SPATIAL DIMENSIONS

7

7

7×7 input (spatially)
assume 3×3 filter
applied with stride 2



A CLOSER LOOK AT SPATIAL DIMENSIONS

7

7

7×7 input (spatially)
assume 3×3 filter
applied with stride 2



A CLOSER LOOK AT SPATIAL DIMENSIONS

7

7×7 input (spatially)
assume 3×3 filter
applied with stride 2

7



3×3 output



A CLOSER LOOK AT SPATIAL DIMENSIONS

7

7

7×7 input (spatially)
assume 3×3 filter
applied with stride 3



A CLOSER LOOK AT SPATIAL DIMENSIONS

7

7

7×7 input (spatially)
assume 3×3 filter
applied with stride 3

doesn't fit!
cannot apply 3x3 filter on
7x7 input with stride 3.



A CLOSER LOOK AT SPATIAL DIMENSIONS

N

			F			
	F					

Output size

$$(N - F) / \text{stride} + 1$$

e.g. $N = 7, F = 3$

stride 1 $\Rightarrow (7 - 3)/1 + 1 = 5$

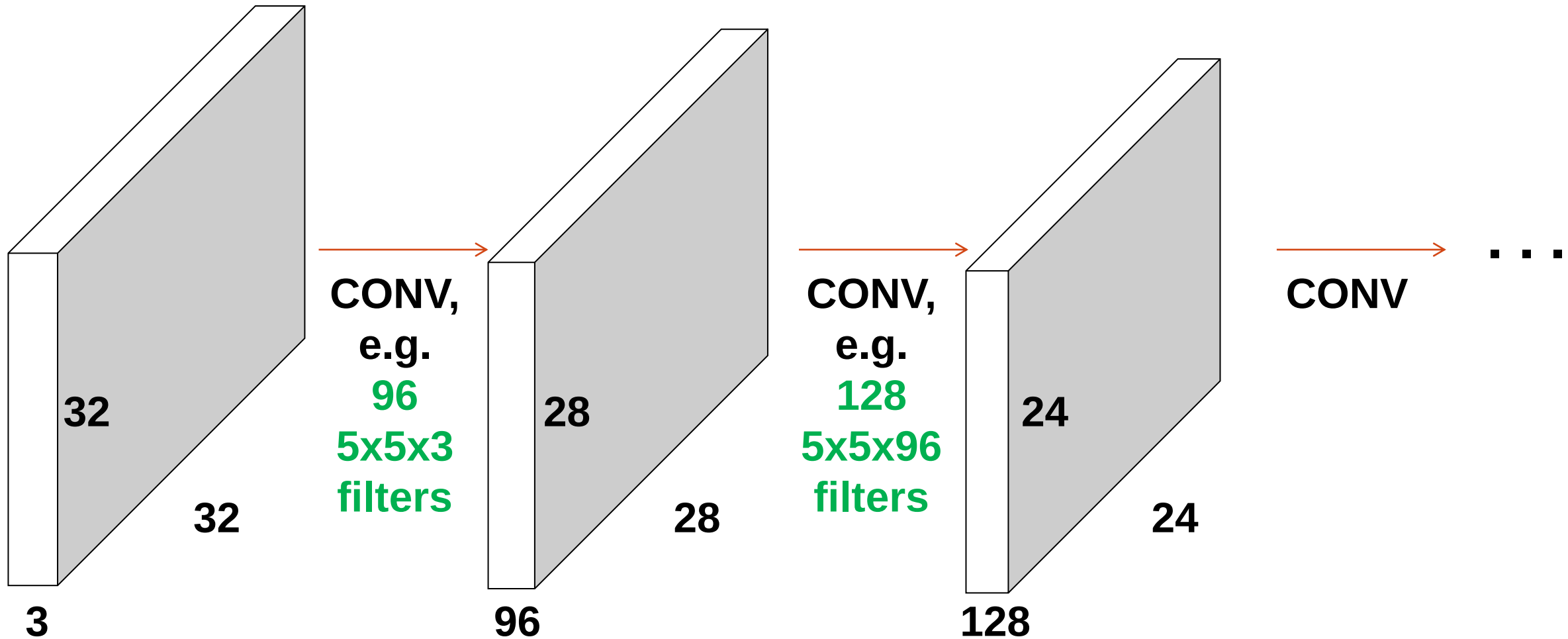
stride 2 $\Rightarrow (7 - 3)/2 + 1 = 3$

stride 3 $\Rightarrow (7 - 3)/3 + 1 = 2.33$

N



A CLOSER LOOK AT SPATIAL DIMENSIONS



E.g. 32x32 input convolved repeatedly with 5x5 filters shrinks volumes spatially! (32 -> 28 -> 24 ...). Shrinking too fast is not good, doesn't work well.



IN PRACTICE: COMMON TO ZERO PAD

0	0	0	0	0	0	0	0	0
0								0
0								0
0								0
0								0
0								0
0								0
0								0
0	0	0	0	0	0	0	0	0

e.g. input 7×7 (spatially)
3×3 filter, applied with stride 1
pad with 1 pixel border

What is the output dimension?



IN PRACTICE: COMMON TO ZERO PAD

0	0	0	0	0	0	0	0	0
0								0
0								0
0								0
0								0
0								0
0								0
0								0
0	0	0	0	0	0	0	0	0

e.g. input 7×7 (spatially)
3×3 filter, applied with stride 1
pad with 1 pixel border

7×7 Output



IN PRACTICE: COMMON TO ZERO PAD

0	0	0	0	0	0	0	0	0
0								0
0								0
0								0
0								0
0								0
0								0
0								0
0	0	0	0	0	0	0	0	0

e.g. input 7×7 (spatially)
3×3 filter, applied with stride 1
pad with 1 pixel border

7×7 Output

in general, common to see CONV layers with
stride 1, filters of size $F \times F$, and zero-padding
with

$(F-1)/2$. (will preserve size spatially)

e.g.

$F = 3 \Rightarrow$ zero pad with 1

$F = 5 \Rightarrow$ zero pad with 2

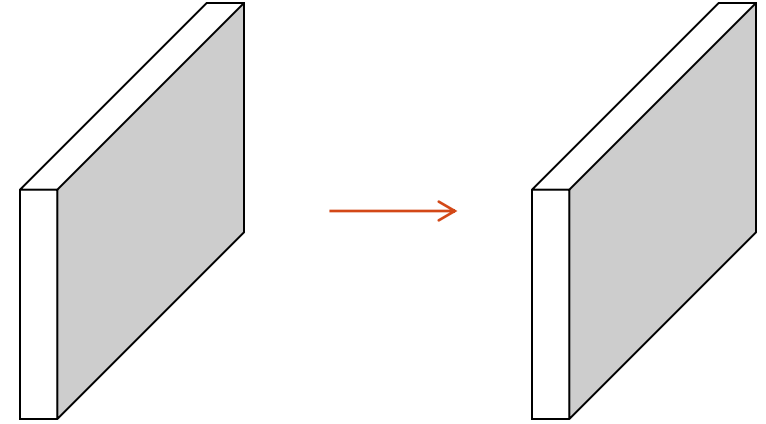
$F = 7 \Rightarrow$ zero pad with 3



EXAMPLE

Input volume: 32x32x3
10 5x5 filters with stride 1, pad 2

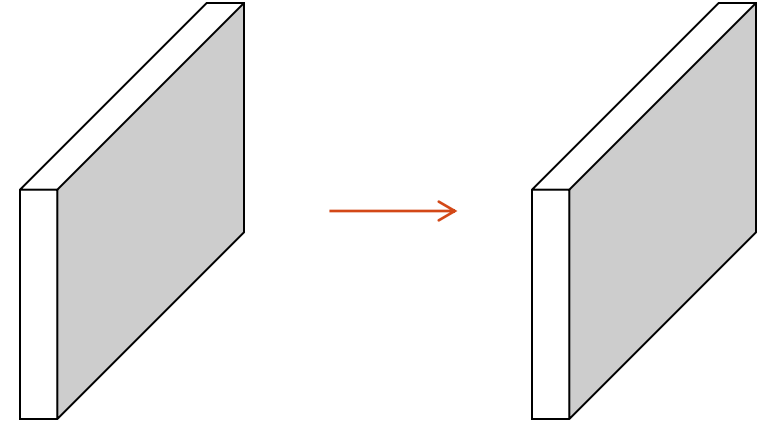
Output volume size: ?



EXAMPLE

Input volume: 32x32x3
10 5x5 filters with stride 1, pad 2

Output volume size:
 $(32+2*2-5)/1+1 = 32$ spatially,
so
32x32x10

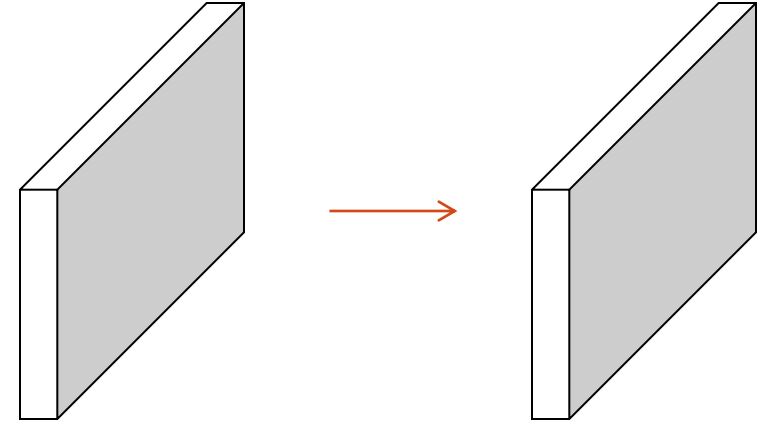


EXAMPLE

Input volume: 32x32x3

10 5x5 filters with stride 1, pad 2

Number of parameters in this layer?



EXAMPLE

Input volume: 32x32x3

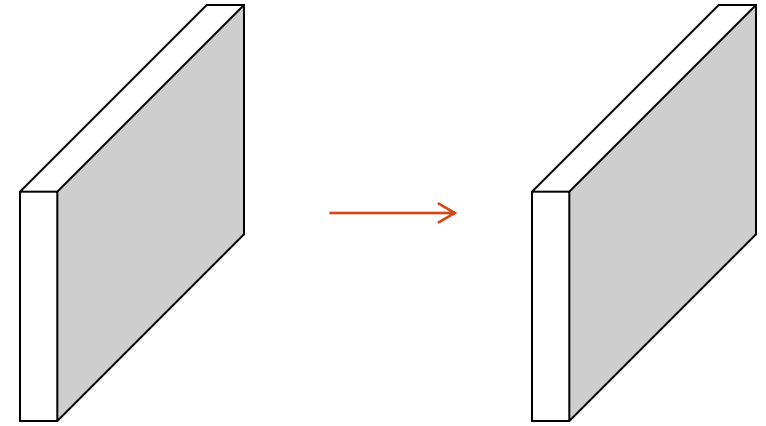
10 5x5 filters with stride 1, pad 2

Number of parameters in this layer?

each filter has

$5*5*3 + 1 = 76$ params (+1 for bias)

$\Rightarrow 76*10 = 760$

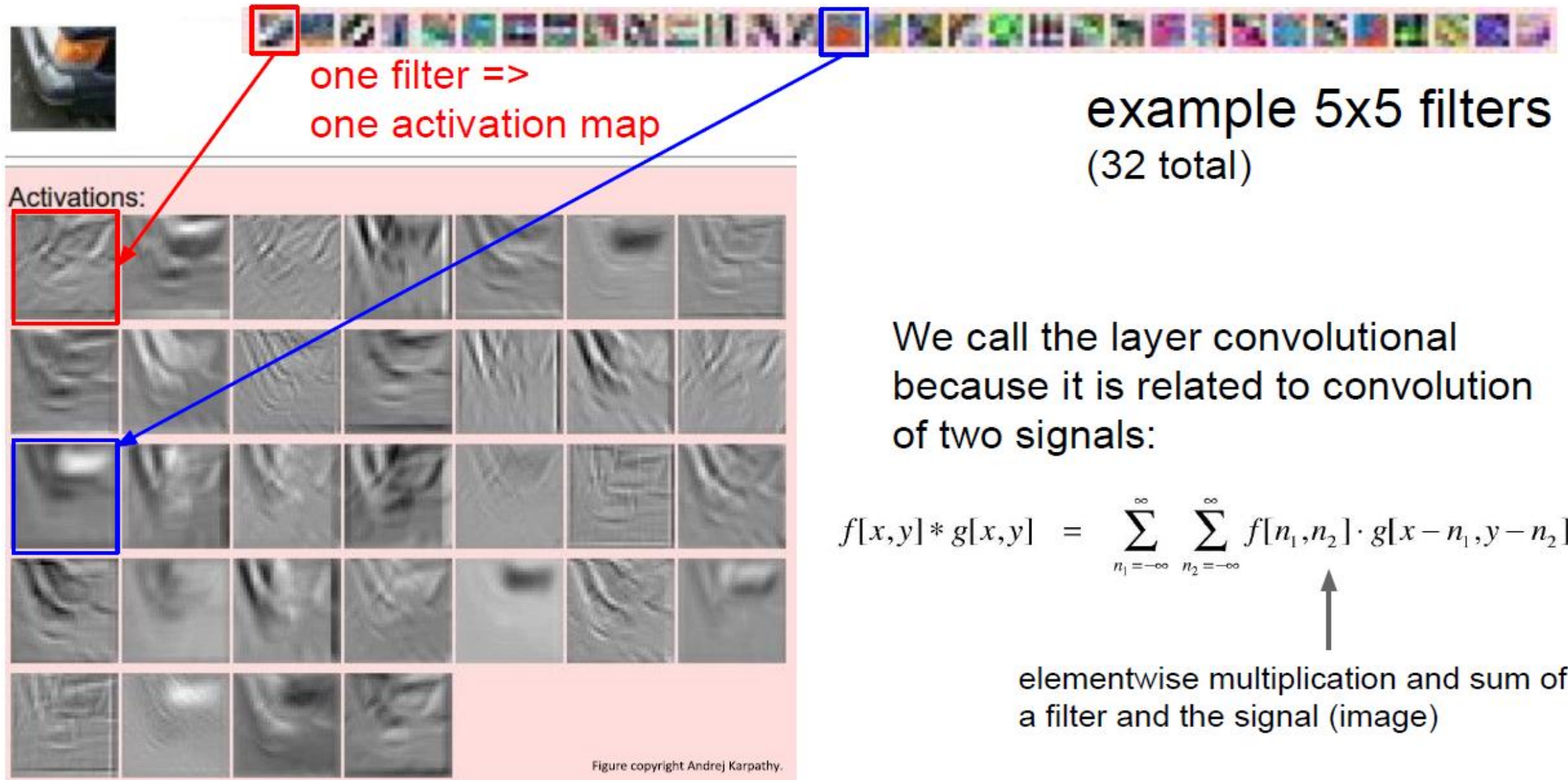


Summary. To summarize, the Conv Layer:

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
 - Number of filters K ,
 - their spatial extent F ,
 - the stride S ,
 - the amount of zero padding P .
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
 - $W_2 = (W_1 - F + 2P)/S + 1$
 - $H_2 = (H_1 - F + 2P)/S + 1$ (i.e. width and height are computed equally by symmetry)
 - $D_2 = K$
- With parameter sharing, it introduces $F \cdot F \cdot D_1$ weights per filter, for a total of $(F \cdot F \cdot D_1) \cdot K$ weights and K biases.

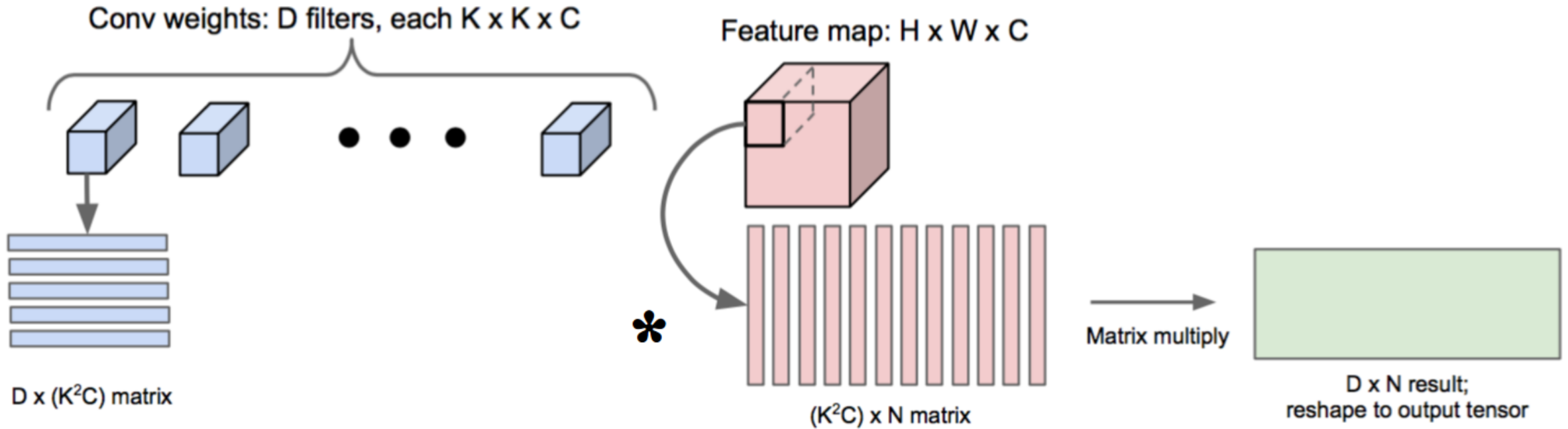


CONVOLUTION AS FEATURE EXTRACTION



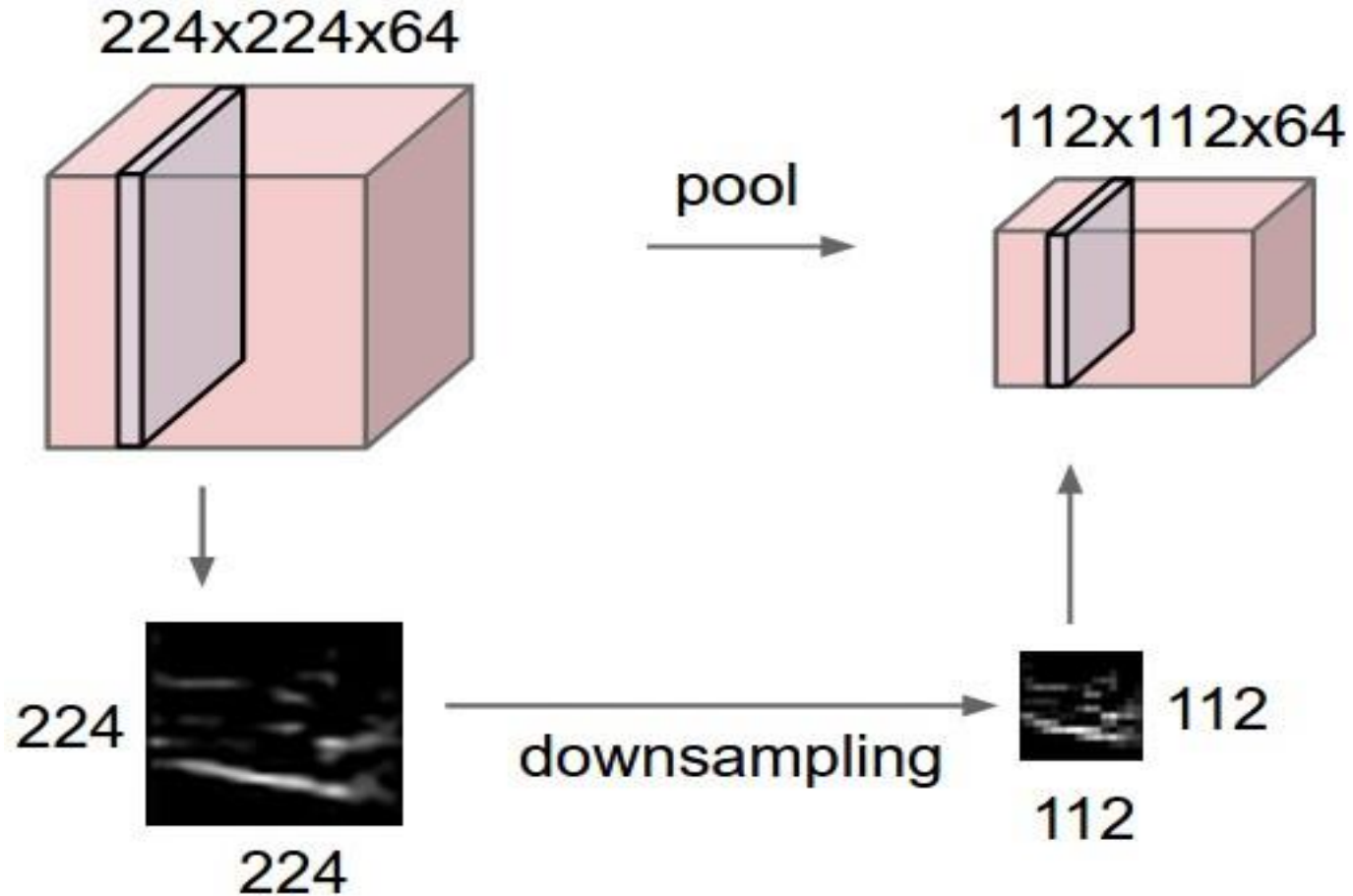
EFFICIENT IMPLEMENTATION OF CONVOLUTIONS

- Reshape all image neighborhoods into columns (im2col operation), do matrix-vector multiplication

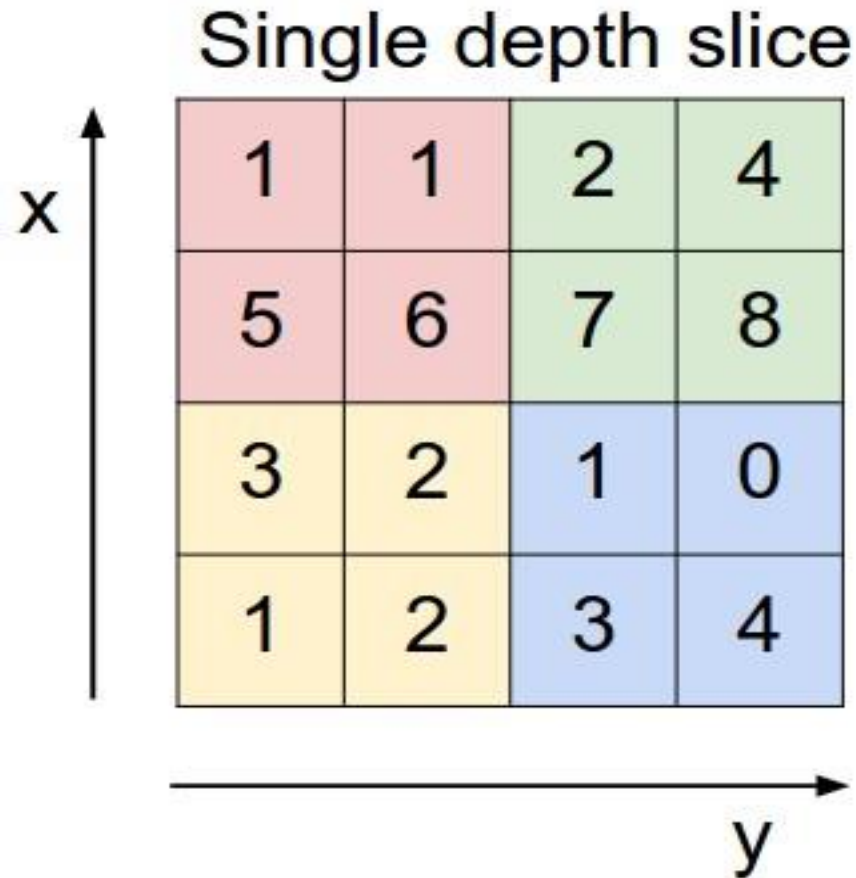


POOLING LAYER

- makes the representations smaller and more manageable
- operates over each activation map independently:



MAX POOLING



max pool with 2x2 filters
and stride 2

6	8
3	4

Backward pass: upstream
gradient is passed back only to
the unit with max value



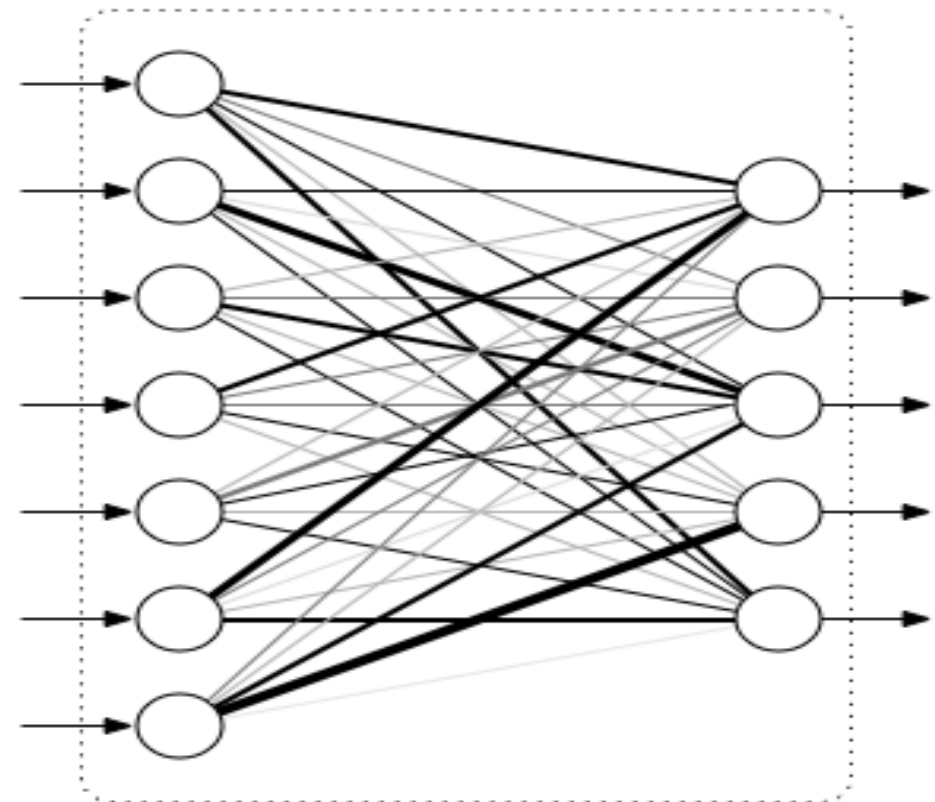
POOLING LAYER

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires two hyperparameters:
 - their spatial extent F ,
 - the stride S ,
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
 - $W_2 = (W_1 - F)/S + 1$
 - $H_2 = (H_1 - F)/S + 1$
 - $D_2 = D_1$
- Introduces zero parameters since it computes a fixed function of the input
- Note that it is not common to use zero-padding for Pooling layers



FULLY CONNECTED LAYER

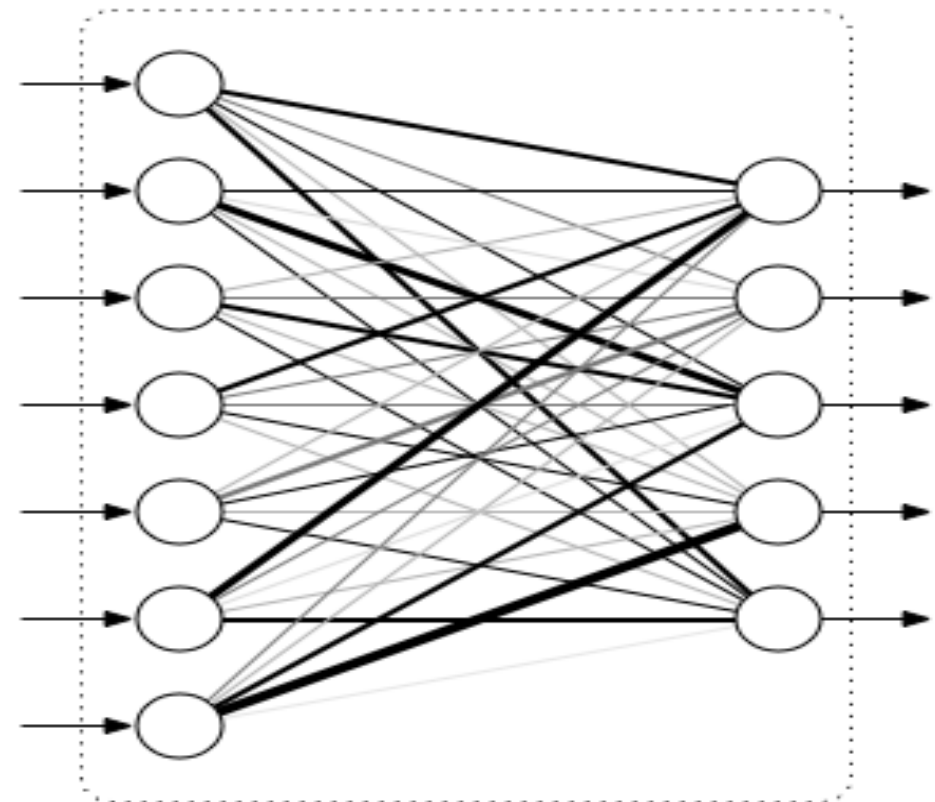
- Connect every neuron in one layer to every neuron in another layer
- Same as the traditional multi-layer perceptron neural network



FULLY CONNECTED LAYER

- Connect every neuron in one layer to every neuron in another layer
- Same as the traditional multi-layer perceptron neural network

No. of Neurons (Last FC)
= No. of classes

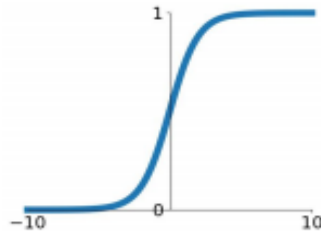


NON-LINEARITY LAYER

Activation Functions

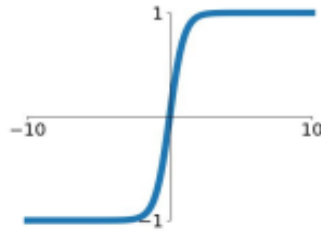
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



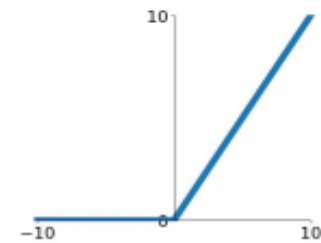
tanh

$$\tanh(x)$$



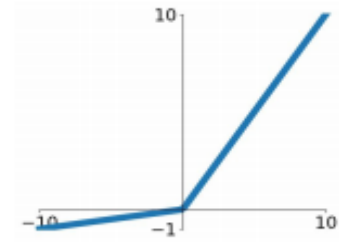
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

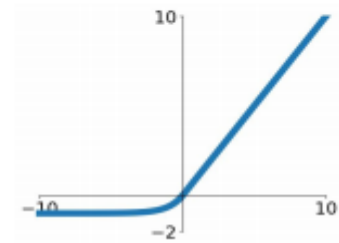


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

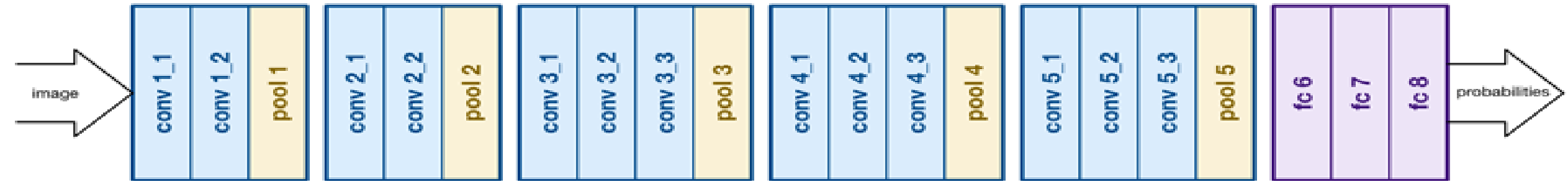


CLASSIFICATION/LOSS LAYER

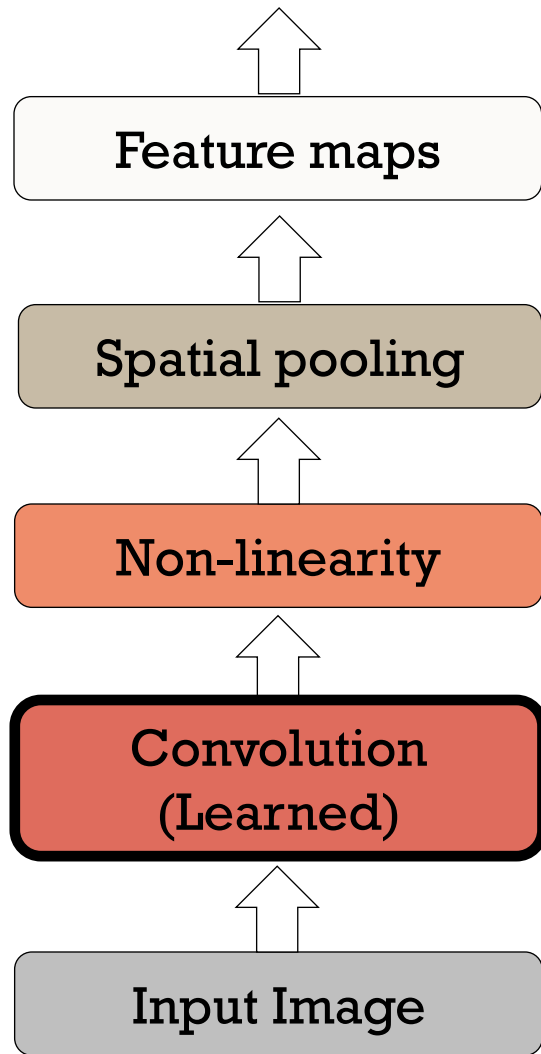
- SVM Classifier
 - SVM Loss/Hinge Loss/Max-margin Loss
- Softmax Classifier
 - Softmax Loss/Cross-entropy Loss



A TYPICAL CNN STRUCTURE



SUMMARY: CNN PIPELINE



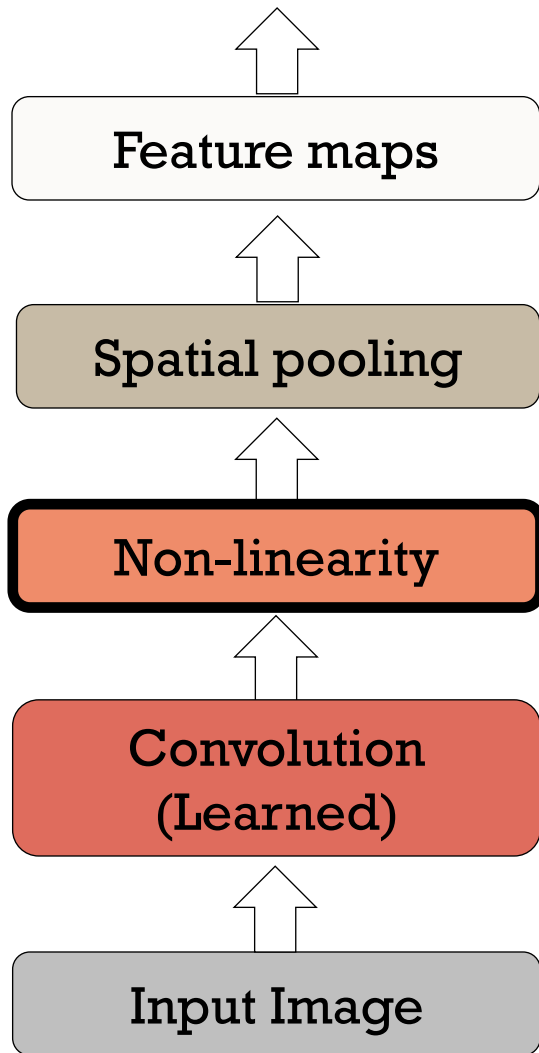
Input



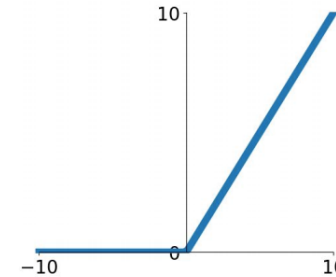
Feature Map



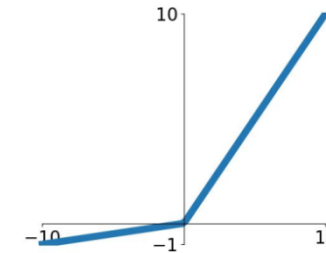
SUMMARY: CNN PIPELINE



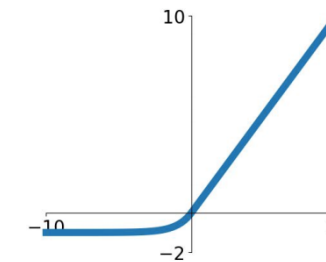
ReLU
 $\max(0, x)$



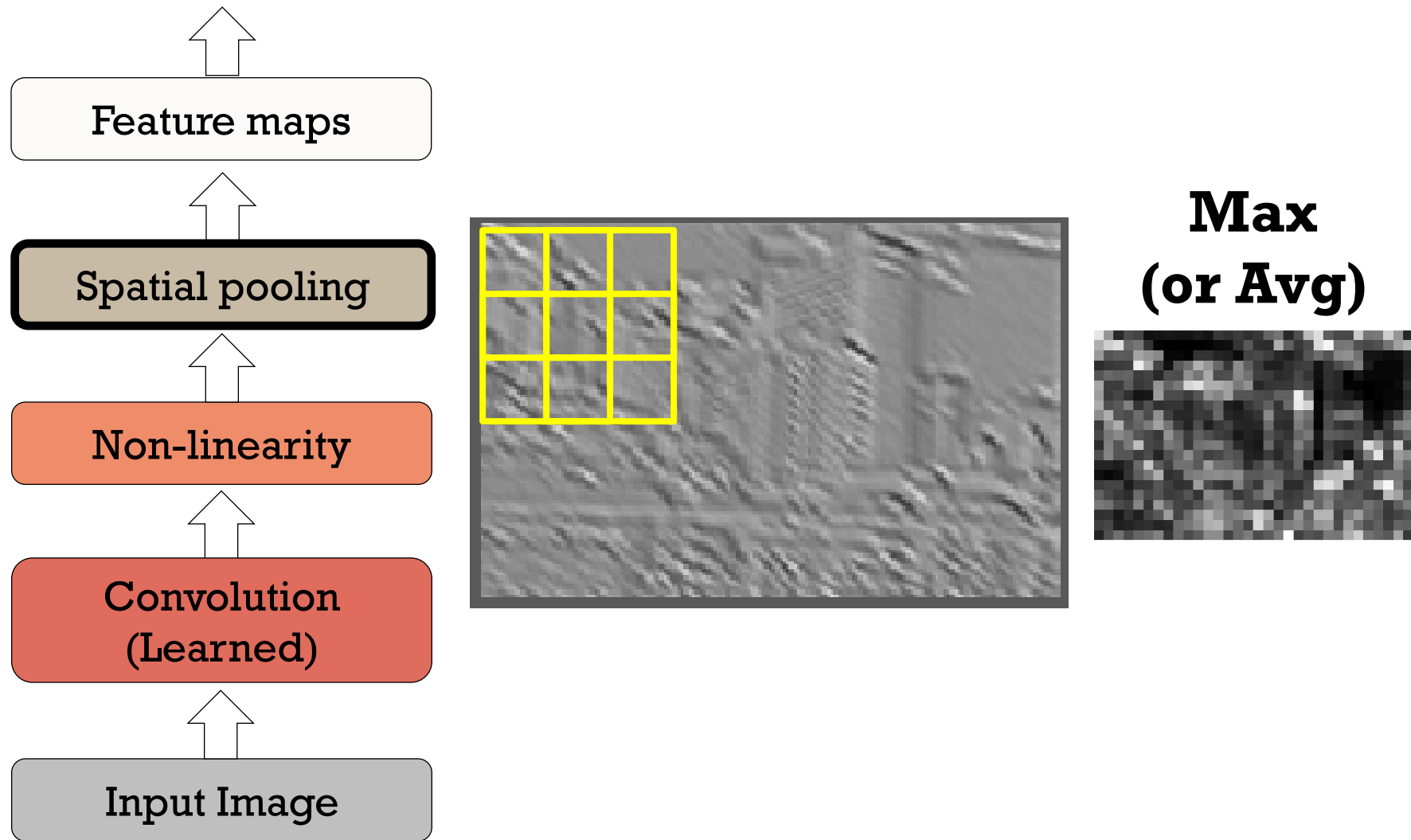
Leaky ReLU
 $\max(0.1x, x)$



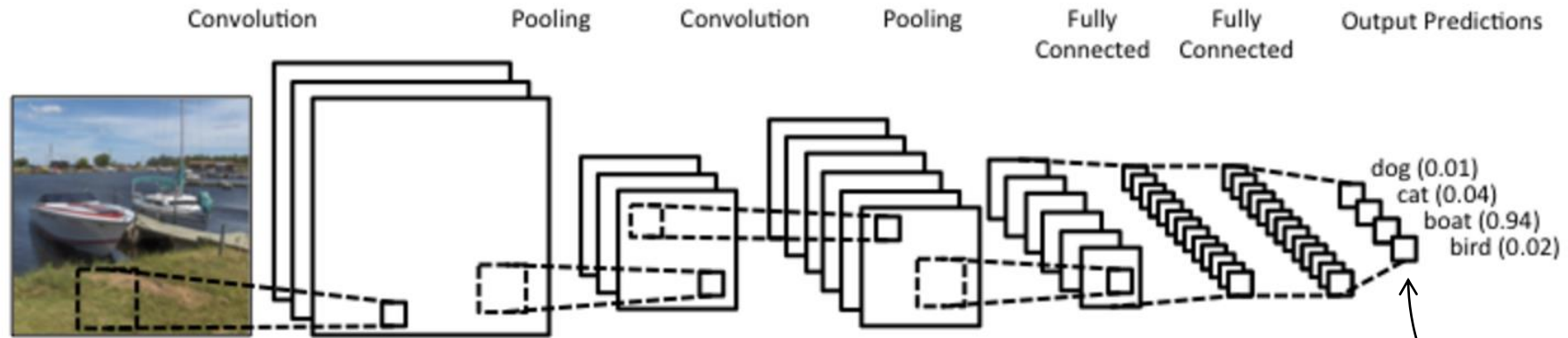
ELU
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



SUMMARY: CNN PIPELINE



SUMMARY: CNN PIPELINE

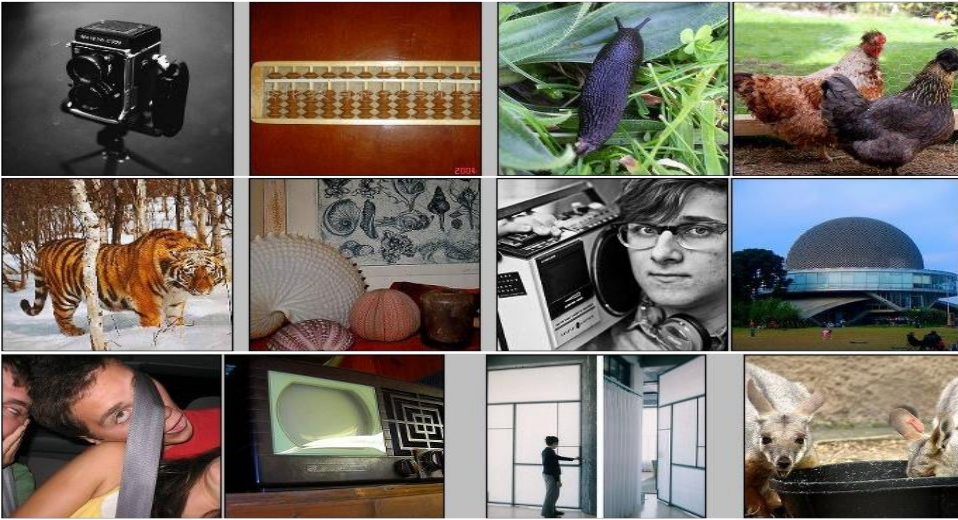


Softmax layer:



IMAGENET CHALLENGE

IM  GENET



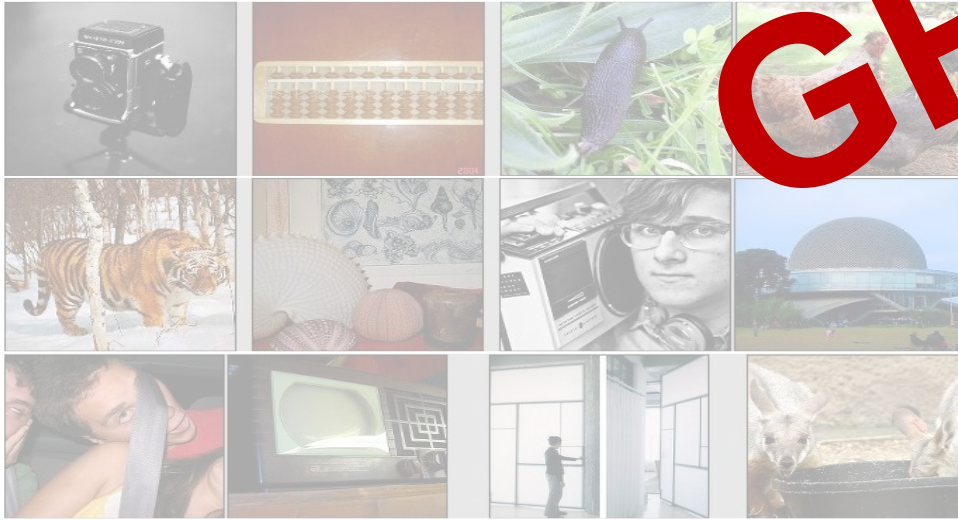
- ~14 million labeled images, 20k classes
- Images gathered from Internet
- Human labels via Amazon MTurk
- Challenge: 1.2 million training images, 1000 classes

www.image-net.org/challenges/LSVRC/



IMAGENET CHALLENGE

IMAGENET



GPUS
+
Data

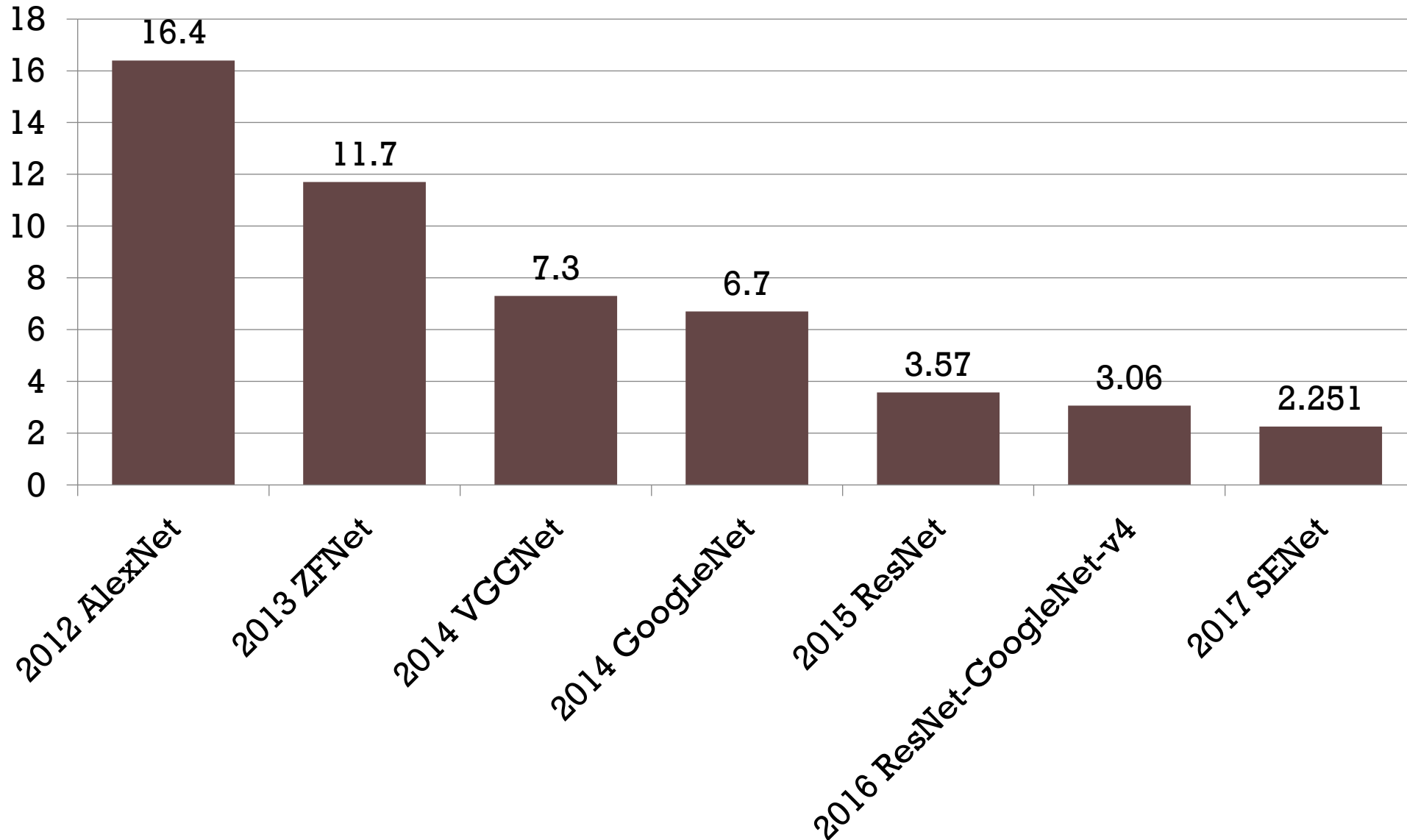
- 14 million labeled images, 20k classes
- Images gathered from Internet
- Human labels via Amazon MTurk
- Challenge 1.2 million training images, 1000 classes

www.image-net.org/challenges/LSVRC/



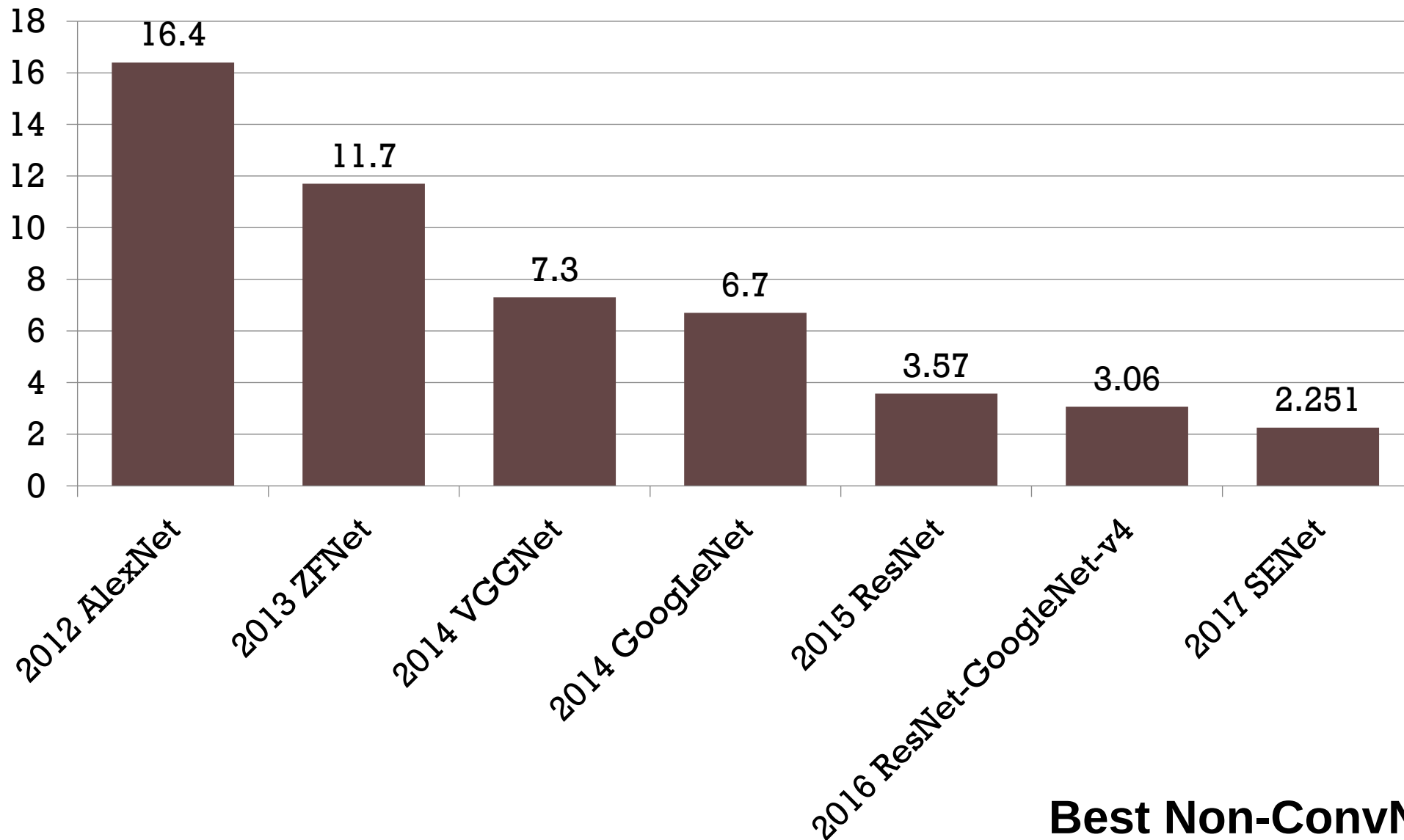
PROGRESS ON IMAGENET CHALLENGE

ImageNet Image Classification Top5 Error



PROGRESS ON IMAGENET CHALLENGE

ImageNet Image Classification Top5 Error



Best Non-ConvNet in 2012: 26.2%

TRANSFER LEARNING WITH CNNs

1. Train on Imagenet



Donahue et al, “DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition”, ICML 2014

Razavian et al, “CNN Features Off-the-Shelf: An Astounding Baseline for Recognition”, CVPR Workshops 2014

Source: cs231n

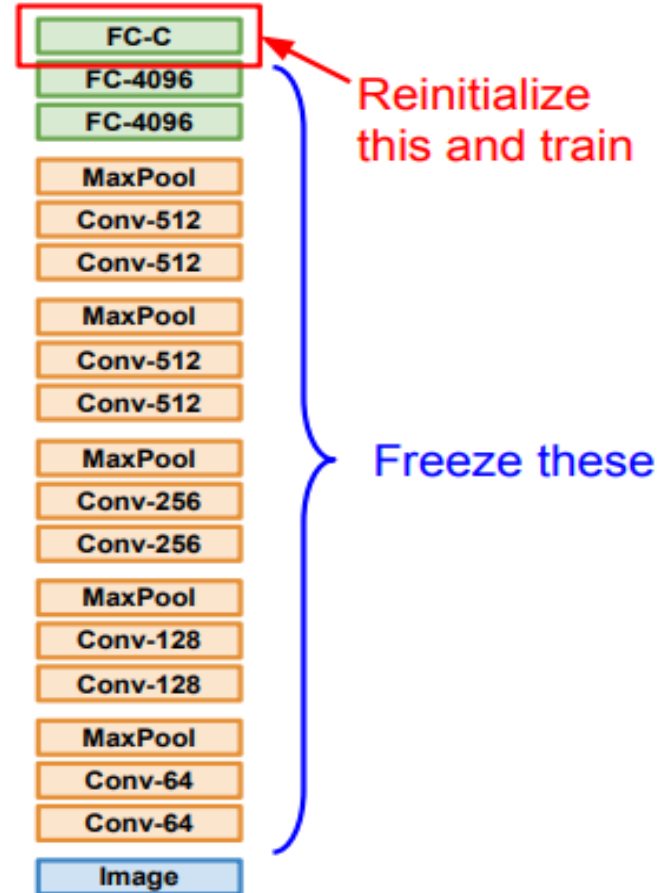


TRANSFER LEARNING WITH CNNs

1. Train on Imagenet



2. Small Dataset (C classes)



Donahue et al, “DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition”, ICML 2014

Razavian et al, “CNN Features Off-the-Shelf: An Astounding Baseline for Recognition”, CVPR Workshops 2014

Source: cs231n

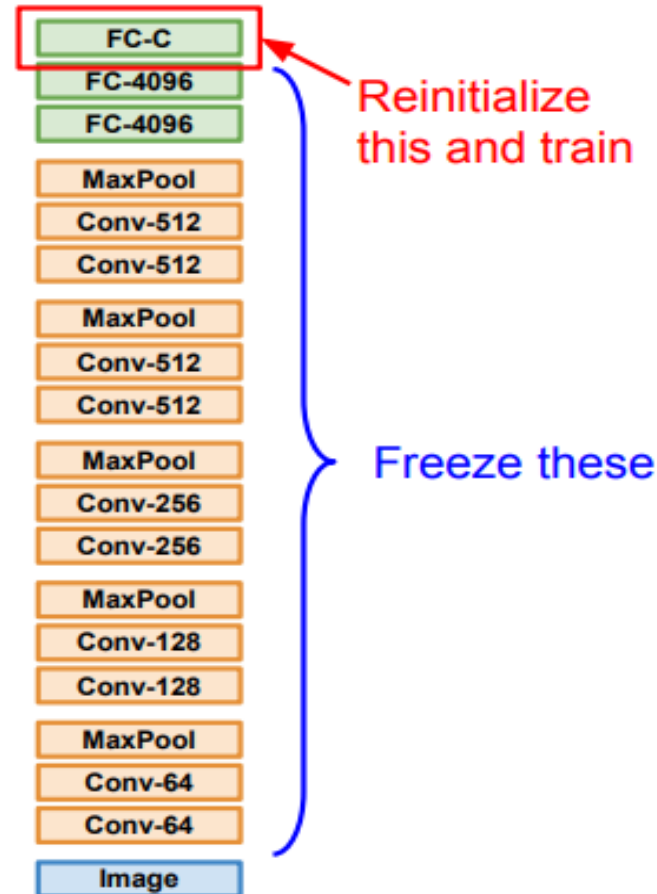


TRANSFER LEARNING WITH CNNs

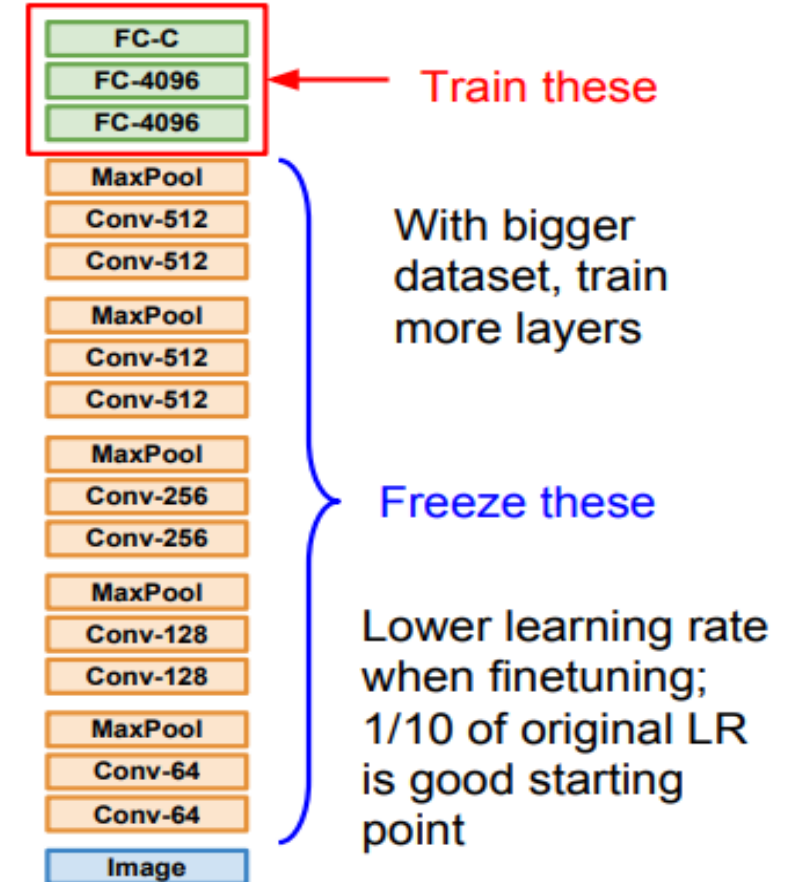
1. Train on Imagenet



2. Small Dataset (C classes)



3. Bigger dataset



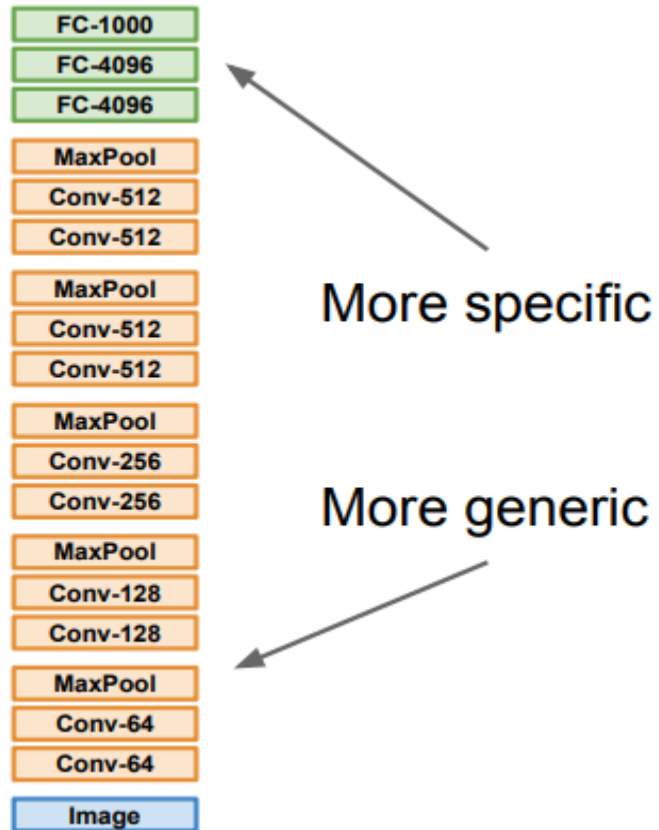
Donahue et al, "DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition", ICML 2014

Razavian et al, "CNN Features Off-the-Shelf: An Astounding Baseline for Recognition", CVPR Workshops 2014

Source: cs231n



TRANSFER LEARNING WITH CNNs



TRANSFER LEARNING WITH CNNs



More specific

More generic

	very similar dataset	very different dataset
very little data		
quite a lot of data		



TRANSFER LEARNING WITH CNNs



More specific

More generic

	very similar dataset	very different dataset
very little data	Use Linear Classifier on top layer	
quite a lot of data		



TRANSFER LEARNING WITH CNNs



More specific

More generic

	very similar dataset	very different dataset
very little data	Use Linear Classifier on top layer	
quite a lot of data	Finetune a few layers	



TRANSFER LEARNING WITH CNNs



More specific

More generic

	very similar dataset	very different dataset
very little data	Use Linear Classifier on top layer	
quite a lot of data	Finetune a few layers	Finetune a larger number of layers



TRANSFER LEARNING WITH CNNs



More specific

More generic

	very similar dataset	very different dataset
very little data	Use Linear Classifier on top layer	You're in trouble... Try linear classifier from different stages
quite a lot of data	Finetune a few layers	Finetune a larger number of layers



TRANSFER LEARNING WITH CNNs

Takeaway for your projects and beyond:

Have some dataset of interest but it has $< \sim 1\text{M}$ images?

1. Find a very large dataset that has similar data, train a big ConvNet there
2. Transfer learn to your dataset

Deep learning frameworks provide a “Model Zoo” of pretrained models so you don’t need to train your own

Caffe: <https://github.com/BVLC/caffe/wiki/Model-Zoo>

TensorFlow: <https://github.com/tensorflow/models>

PyTorch: <https://github.com/pytorch/vision>

Matconvnet: <http://www.vlfeat.org/matconvnet/pretrained/>



THINGS TO REMEMBER

- **Neural network and Image**
 - Neuroscience, Perceptron, Problems due to High Dimensionality and Local Relationship
- **Convolutional neural network (CNN)**
 - Convolution Layer,
 - Nonlinearity Layer,
 - Pooling Layer,
 - Fully Connected Layer,
 - Loss/Classification Layer
- **Progress on ImageNet challenge**
 - Latest SENet, Winner 2017



ACKNOWLEDGEMENT

Thanks to the following courses and corresponding researchers for making their teaching/research material online

- Convolutional Neural Networks for Visual Recognition, Stanford University
- Deep Learning, Stanford University
- Introduction to Deep Learning, University of Illinois at Urbana-Champaign
- Introduction to Deep Learning, Carnegie Mellon University
- Natural Language Processing with Deep Learning, Stanford University
- And Many More Publicly Available Resources



Questions?

