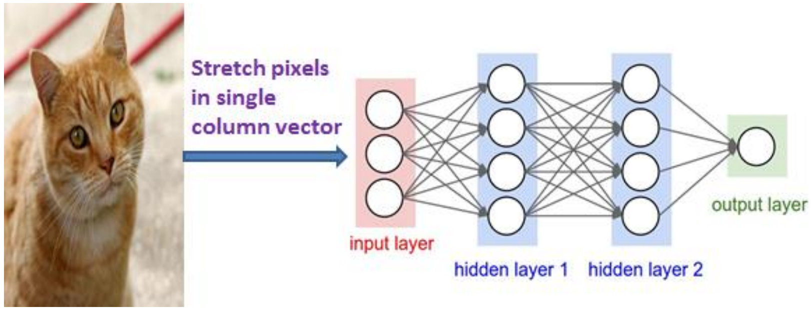
Machine Learning

Convolutional Neural Network

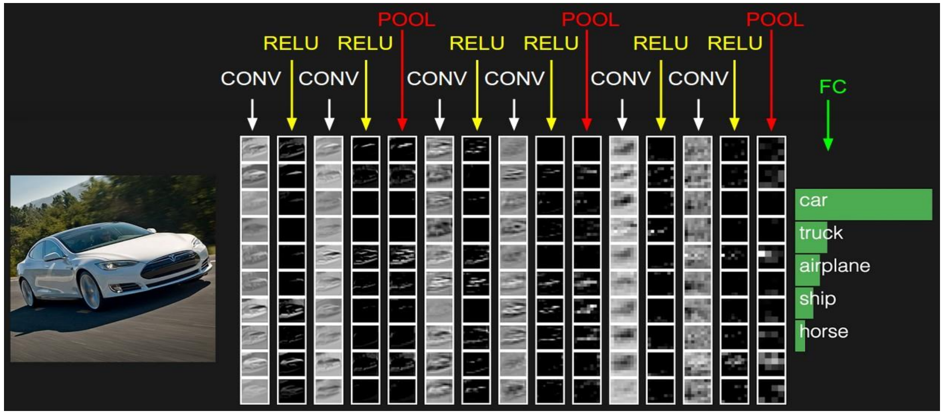
Indian Institute of Information Technology
Sri City, Chittoor

This Class

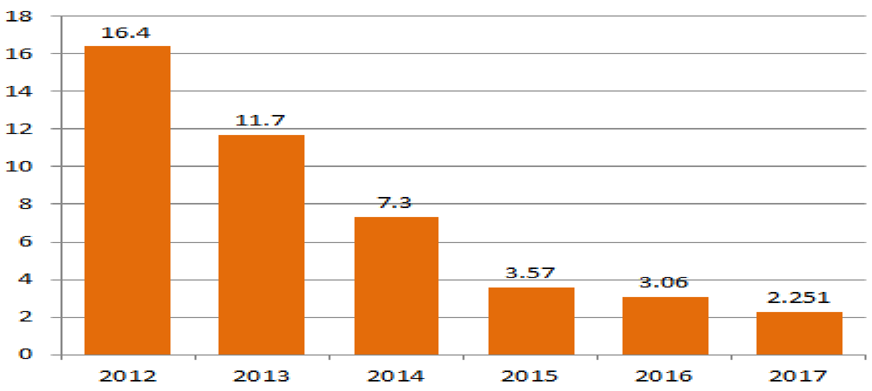
- Neural Network and Image
  - Dimensionality
  - Local relationship



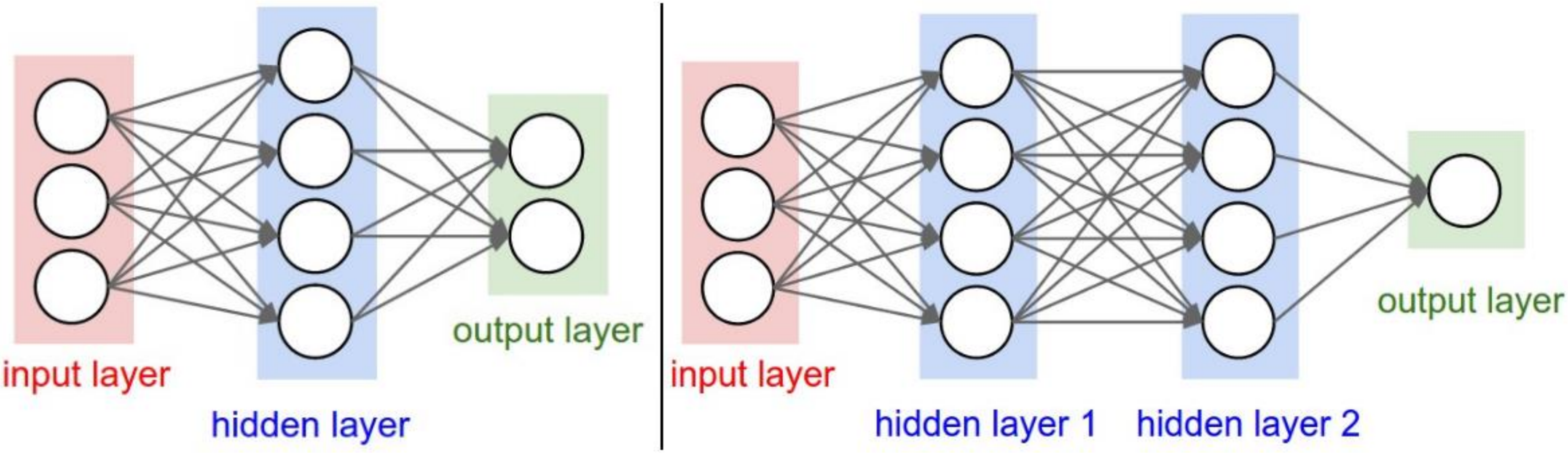
- Convolutional Neural Network (CNN)
  - Convolution Layer
  - Non-linearity Layer
  - Pooling Layer
  - Fully Connected Layer
  - Classification Layer



- ImageNet Challenge
  - Progress
  - Human Level Performance

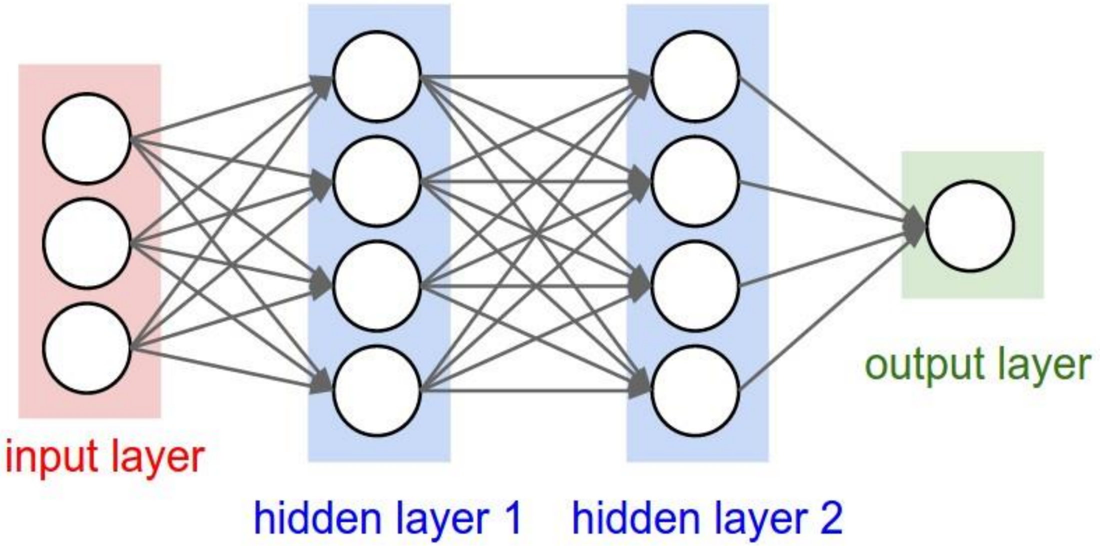


Neural Networks



Source: http://cs231n.github.io

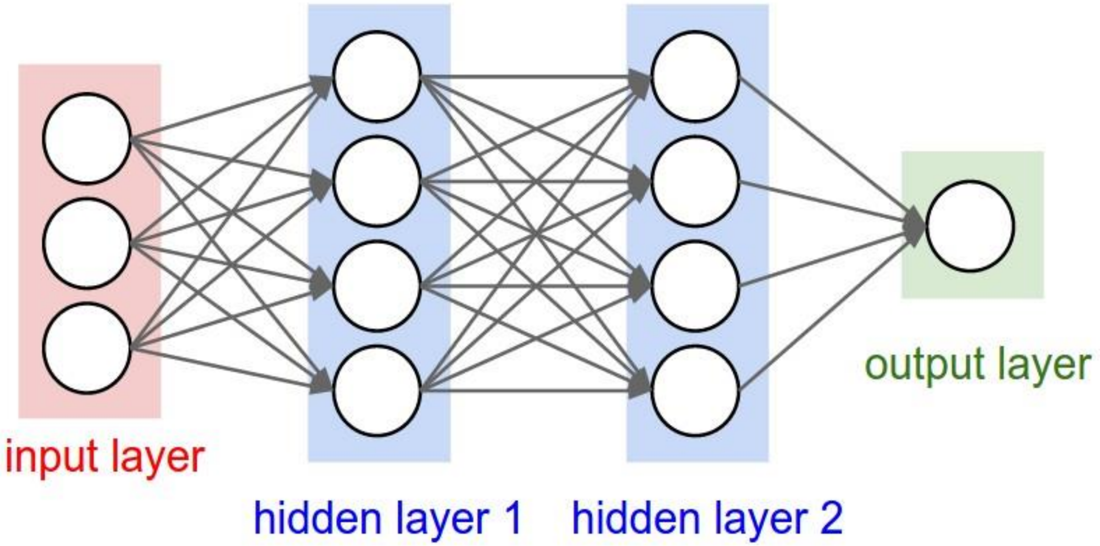
Multi-layer Neural Network & Image



How to apply NN over Image?

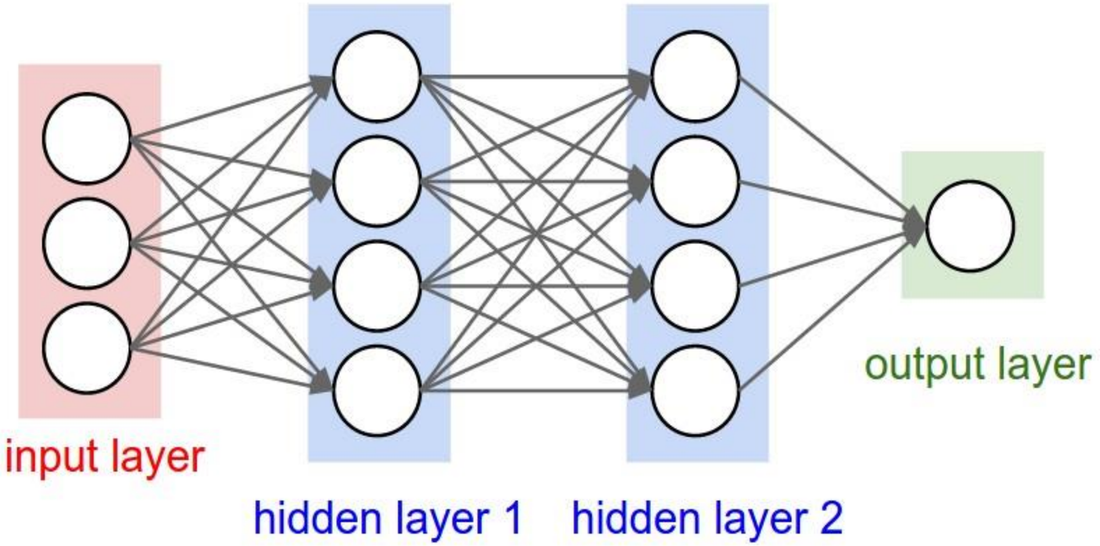
Multi-layer Neural Network & Image

Stretch pixels in single column vector



Multi-layer Neural Network & Image

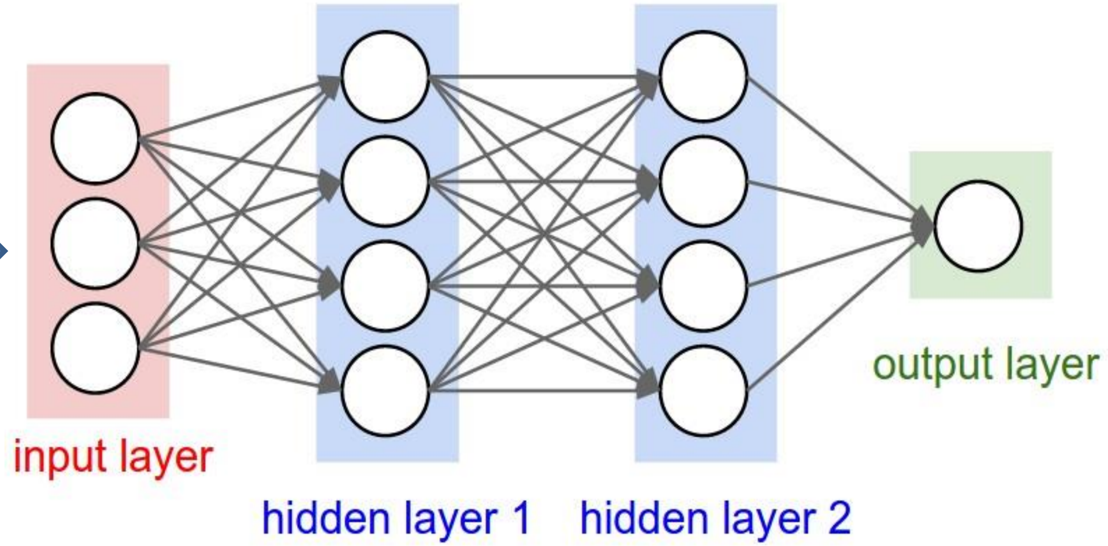
Stretch pixels in single column vector



Problems ?

Multi-layer Neural Network & Image

Stretch pixels in single column vector



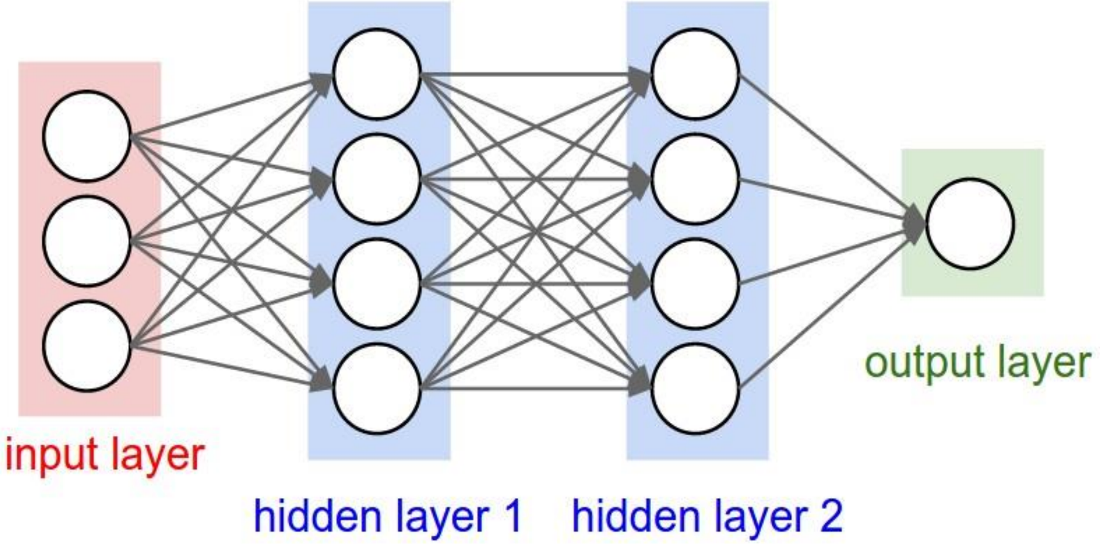
Problems:

High dimensionality

Local relationship

Multi-layer Neural Network & Image

Stretch pixels in single column vector



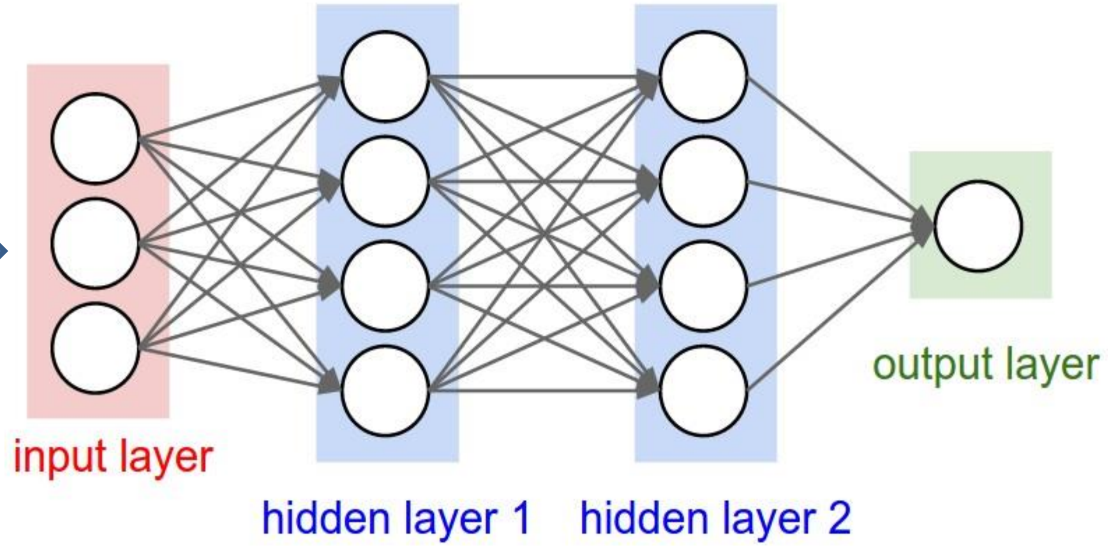
Problems: **Solution ?**

High dimensionality

Local relationship

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

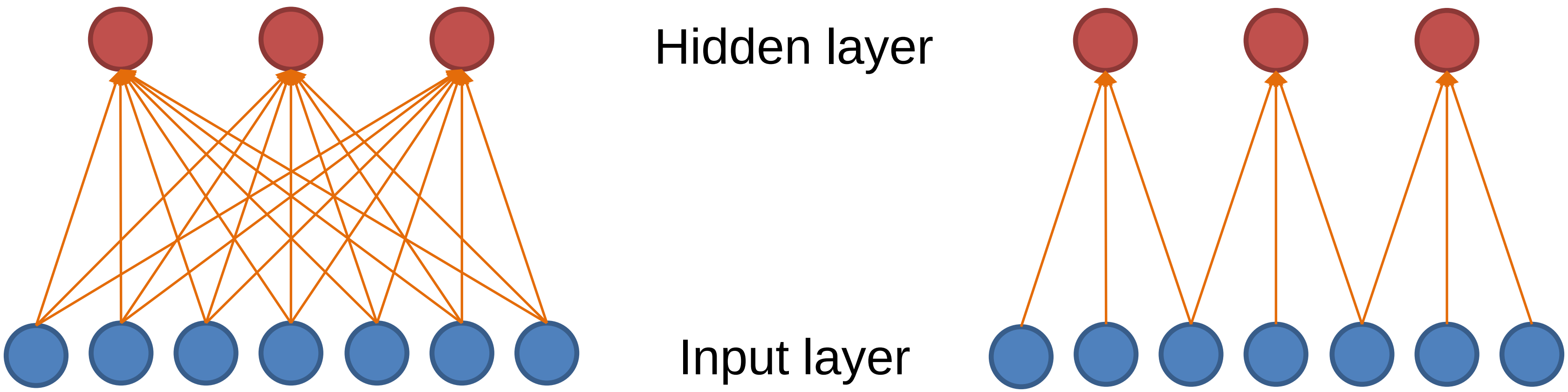
  CNN,

  ConvNet,

  DCN

- CNN = a multi-layer neural network with
  1. Local connectivity
  2. Weight sharing

CNN: Local Connectivity

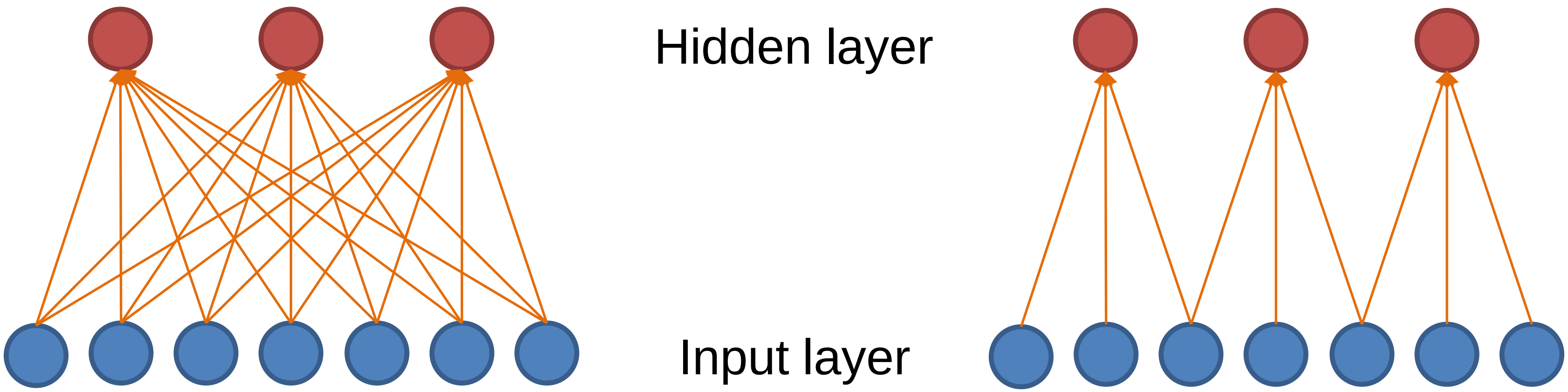


Global connectivity

Local connectivity

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

CNN: Local Connectivity

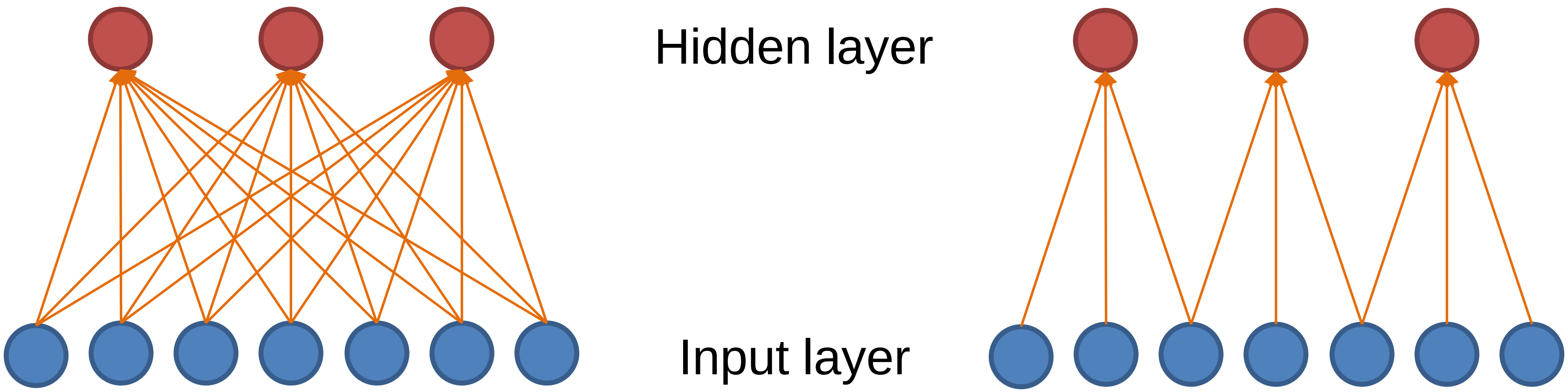


Global connectivity

Local connectivity

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

CNN: Local Connectivity

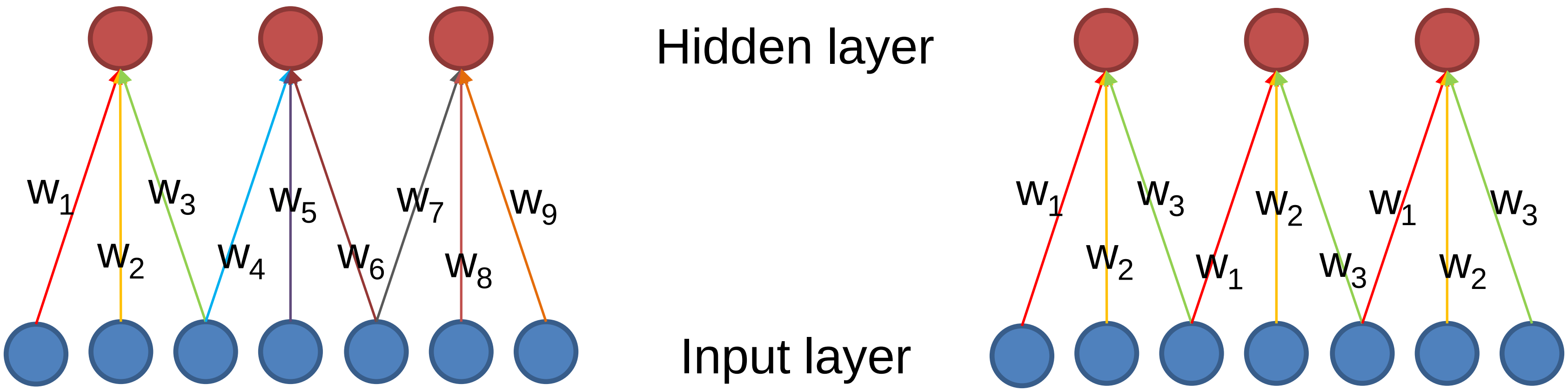


Global connectivity

Local connectivity

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

CNN: Weight Sharing

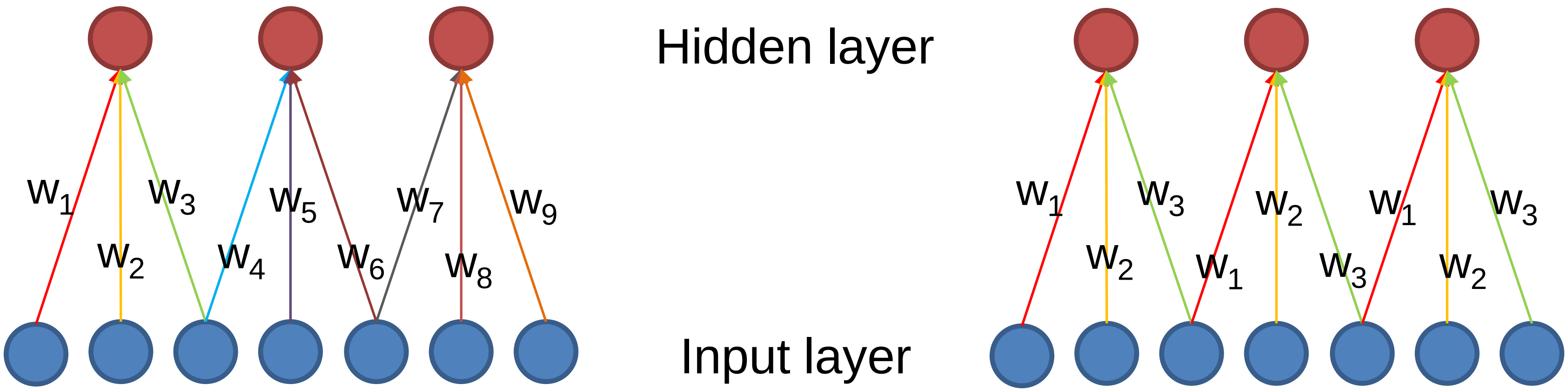


Without weight sharing

With weight sharing

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

CNN: Weight Sharing

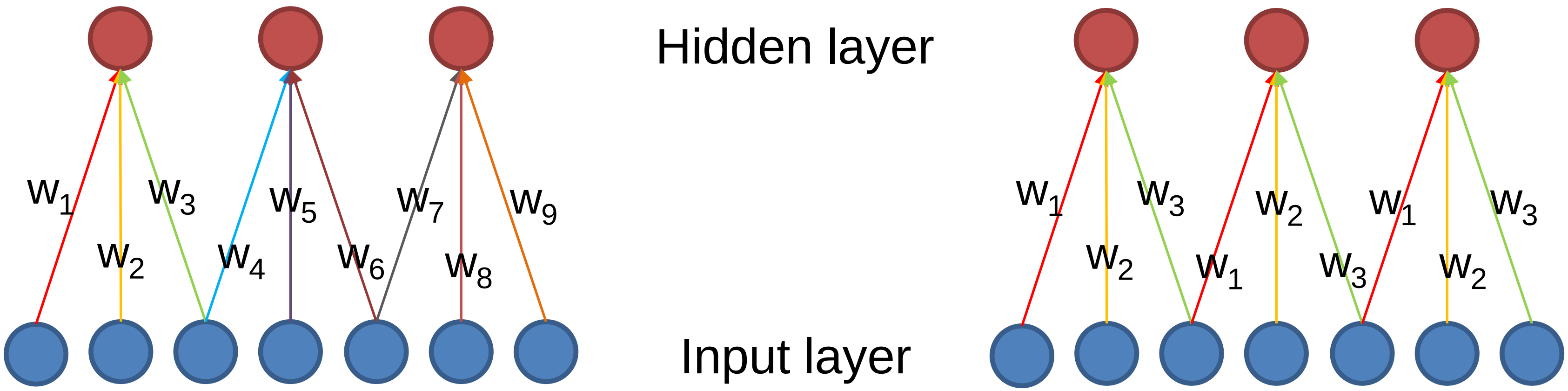


Without weight sharing

With weight sharing

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

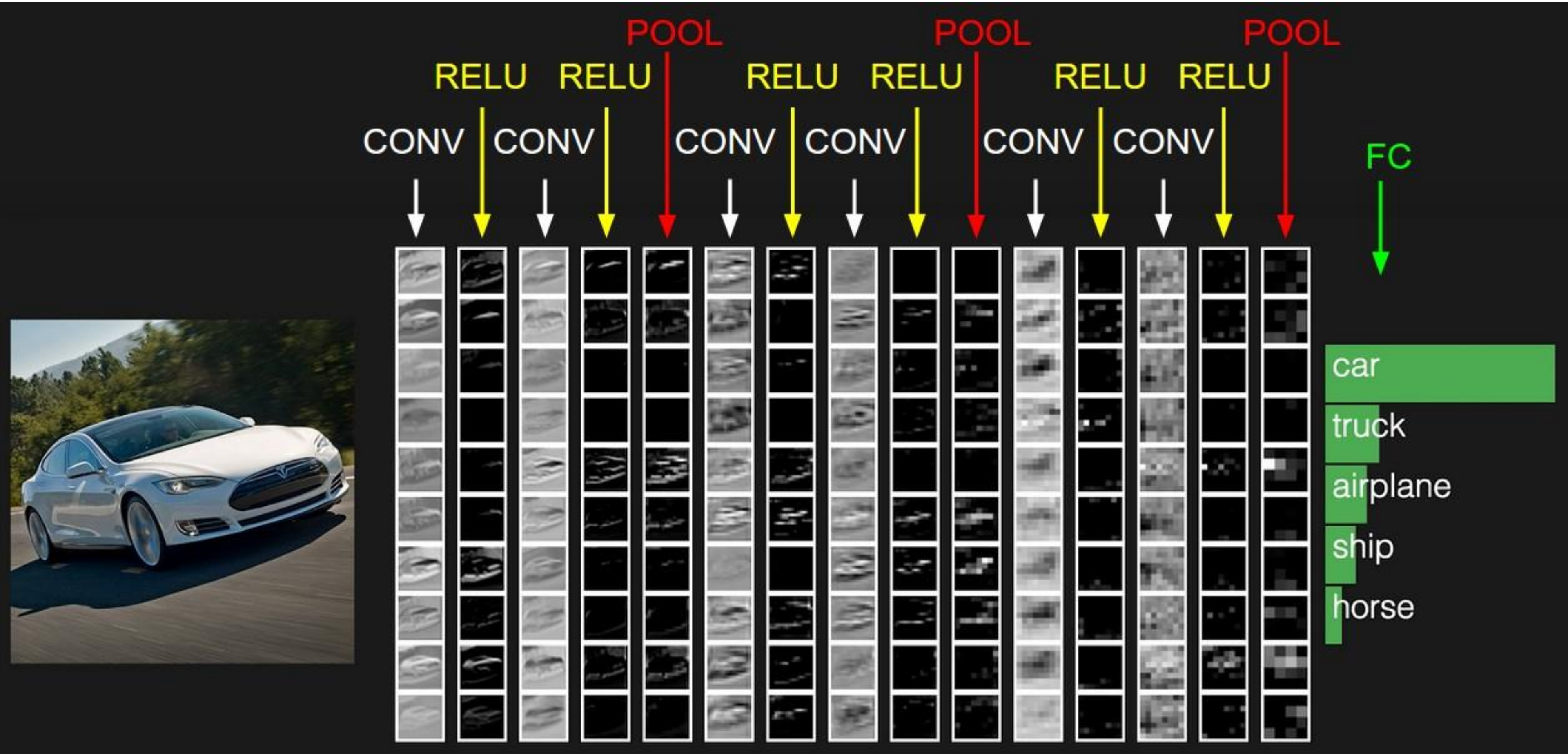
CNN: Weight Sharing



Without weight sharing **With** weight sharing

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

Convolutional Neural Networks



Source: cs231n, Stanford University

Layers used to build ConvNets

Input Layer (Input image)

Convolutional Layer

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

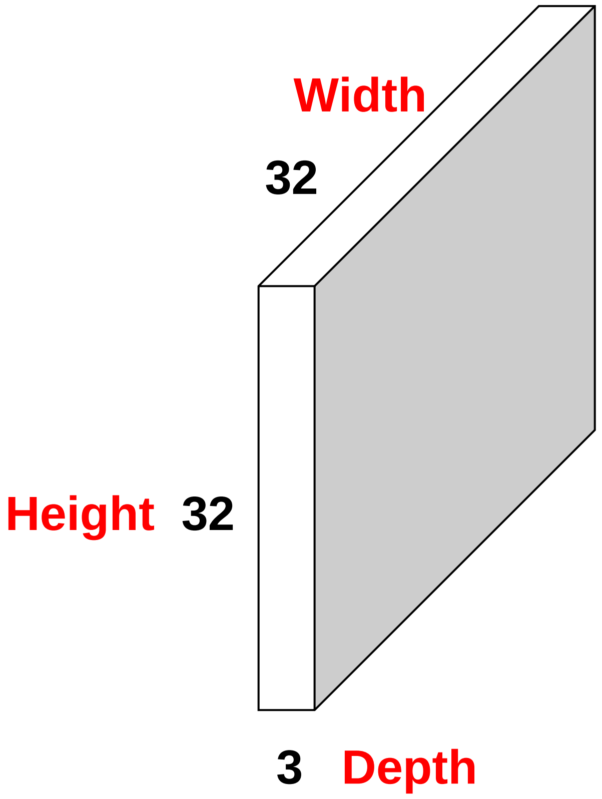
Pooling Layer (such as Max Pooling, Average Pooling, etc.)

Fully-Connected Layer

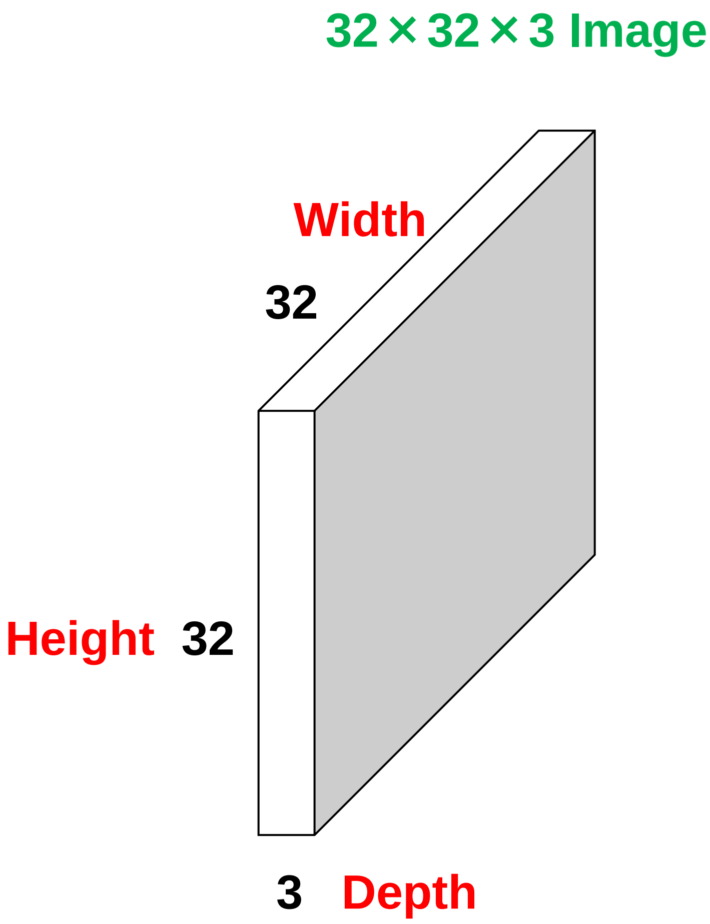
Classification Layer (Softmax, etc.)

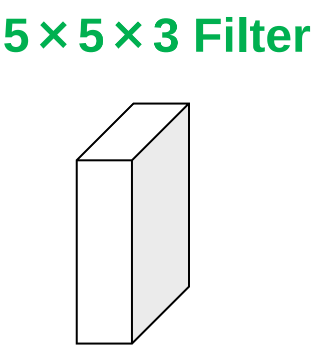
Convolutional Layer

32 × 32 × 3 Image -> preserve spatial structure



Convolutional Layer





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

Convolutional Layer

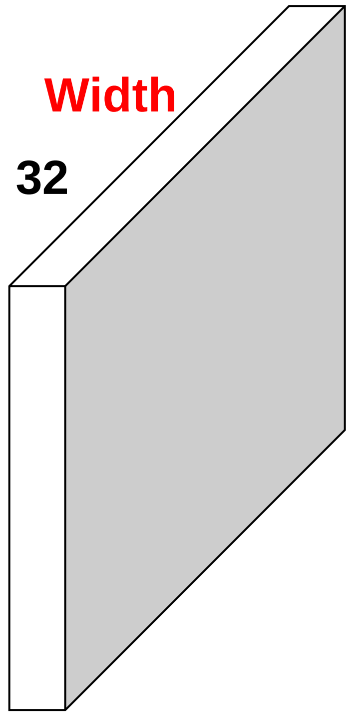
Handling multiple input channels

$32 \times 32 \times 3$ Image

Filters always extend the full depth of the input volume

Width

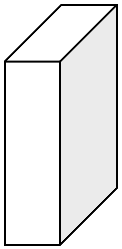
32



Height 32

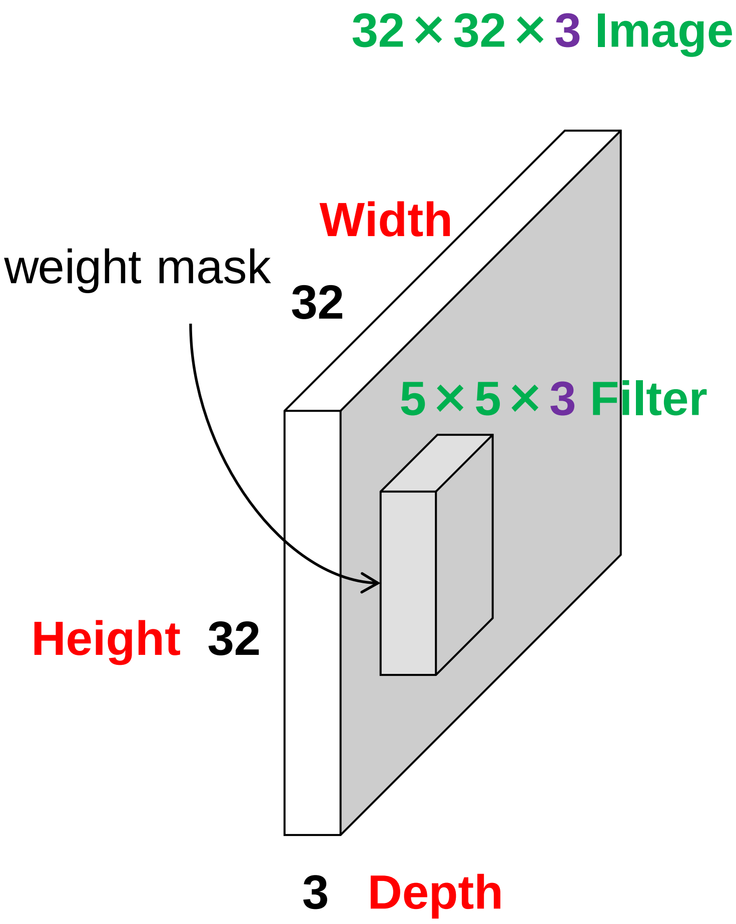
3 Depth

$5 \times 5 \times 3$ Filter

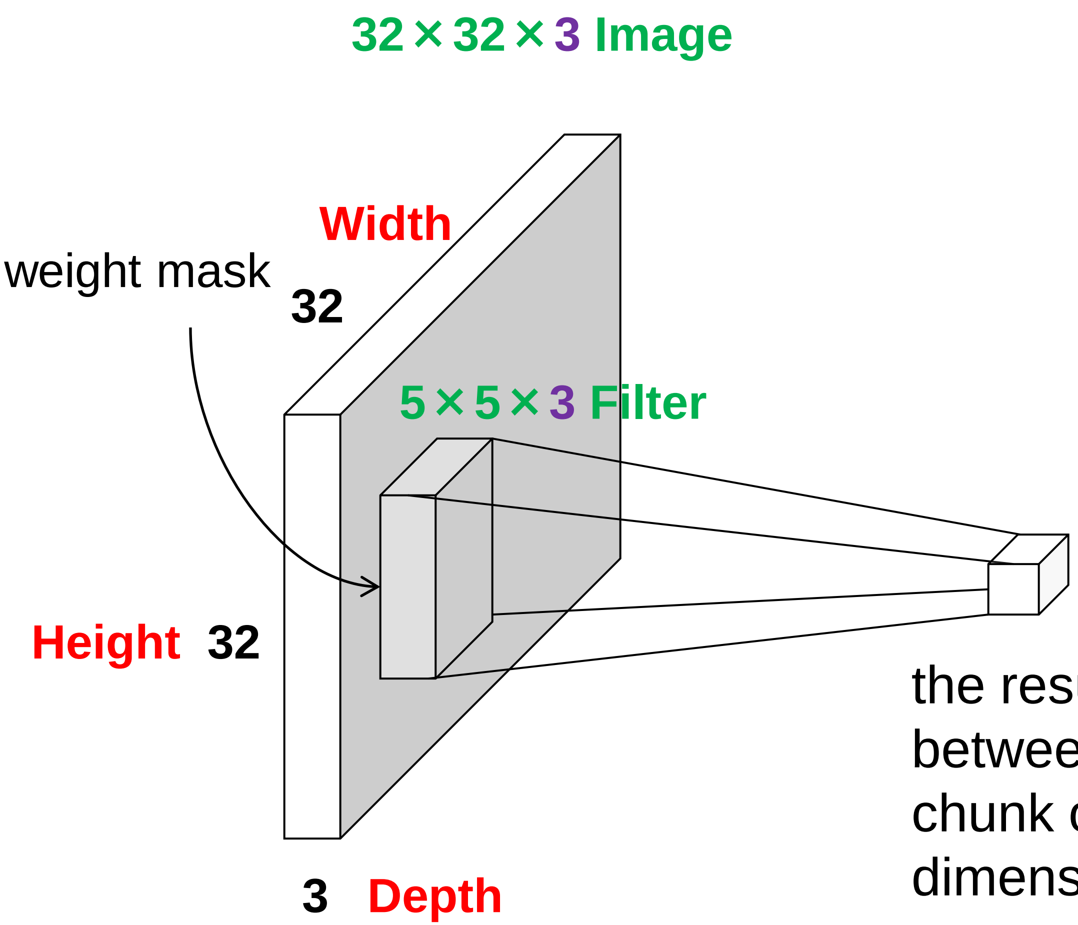


Convolve the filter with the image i.e. “slide over the image spatially, computing dot products”

Convolutional Layer



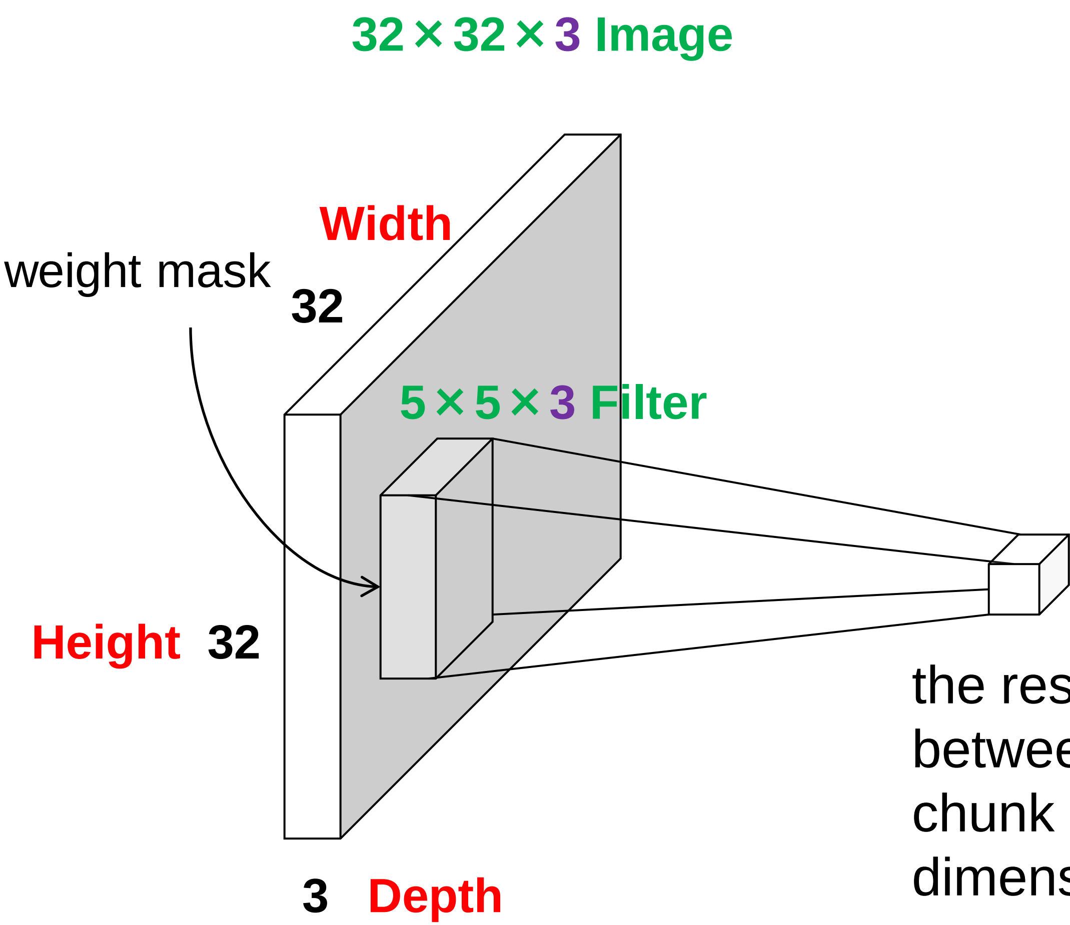
Convolutional Layer



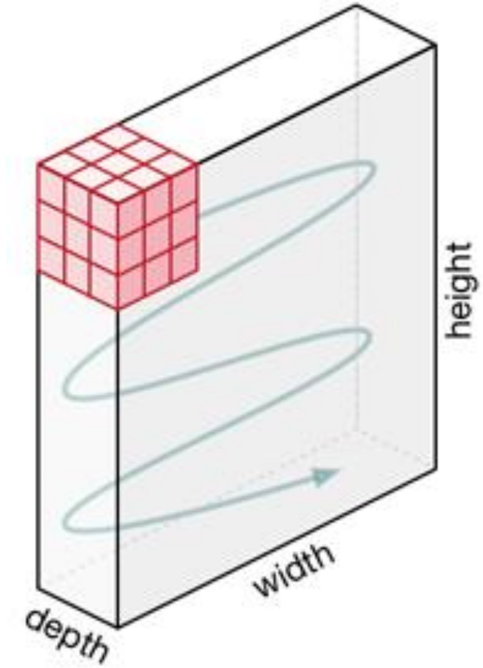
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image (i.e. 5*5*3 = 75-dimensional dot product + bias)

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

Convolutional Layer



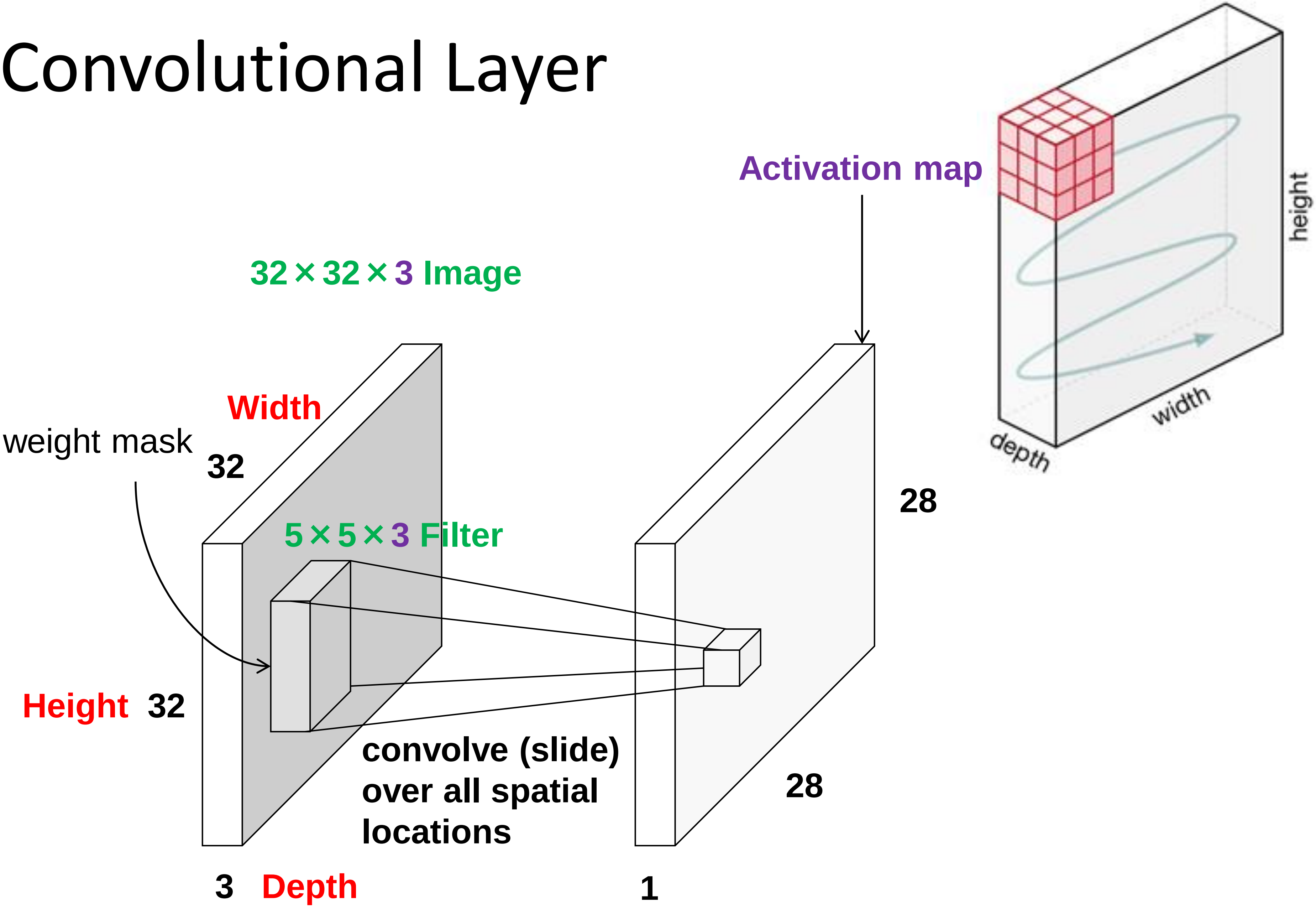
A single value



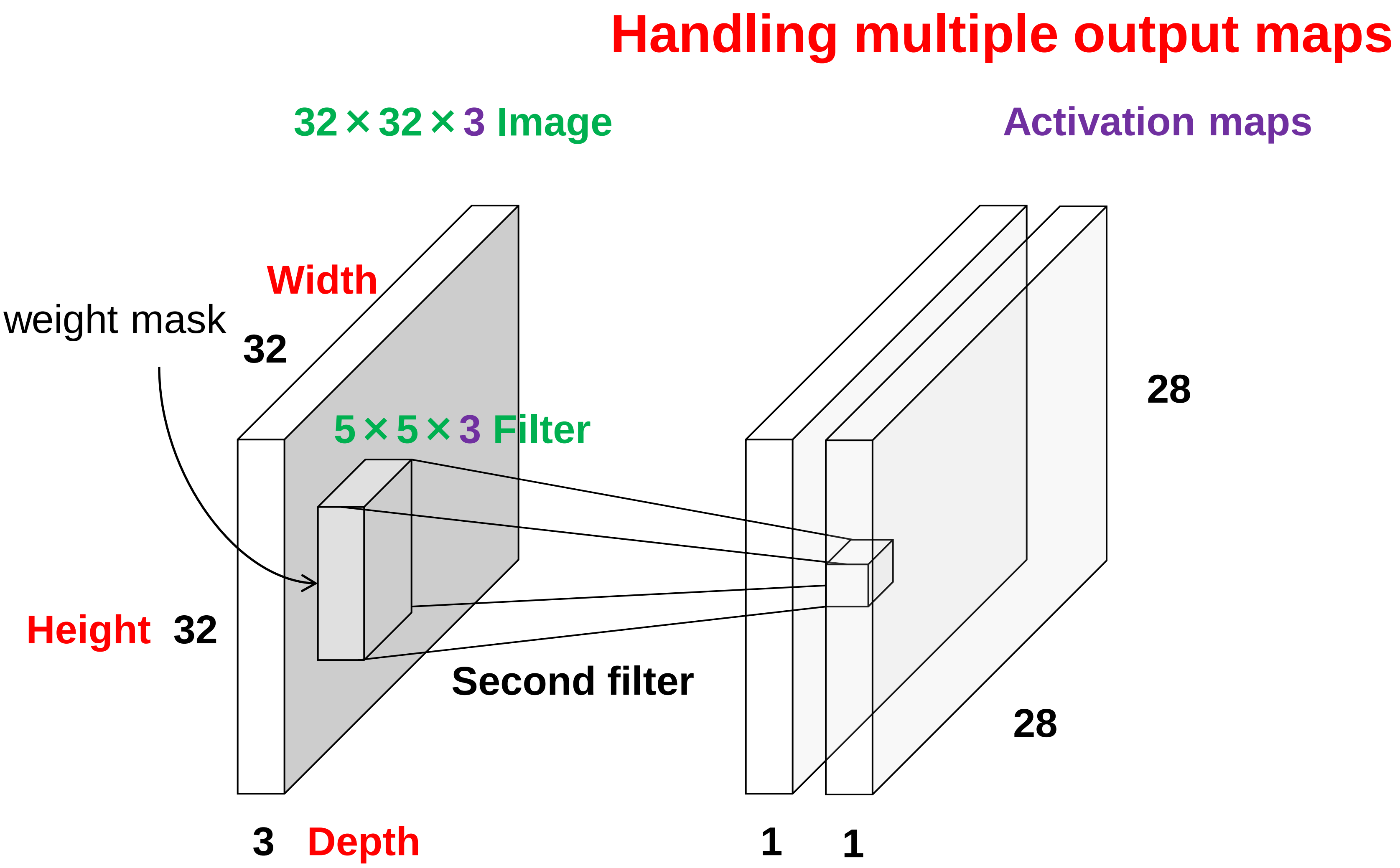
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image (i.e. 5*5*3 = 75-dimensional dot product + bias)

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

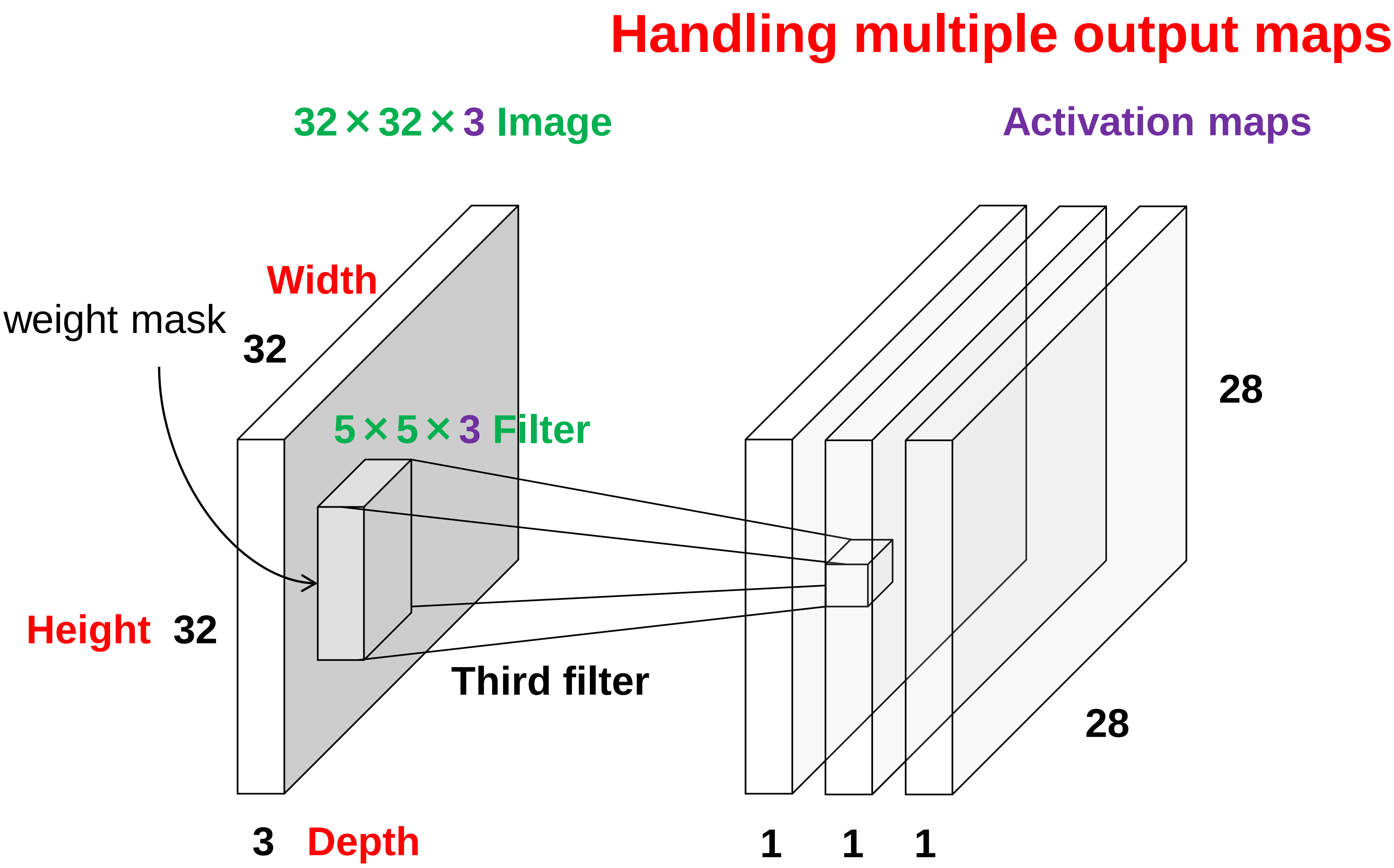
Convolutional Layer



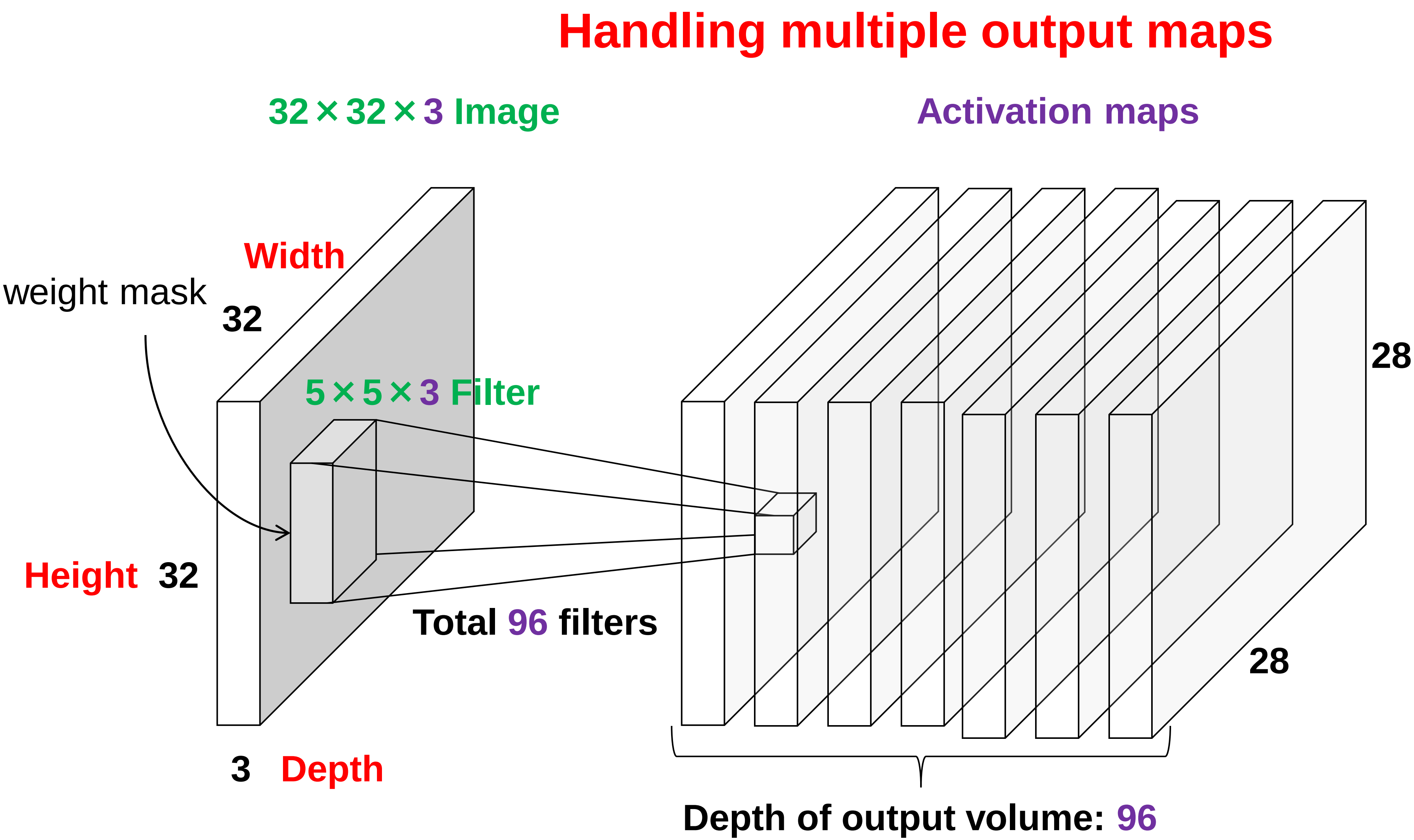
Convolutional Layer



Convolutional Layer



Convolutional Layer



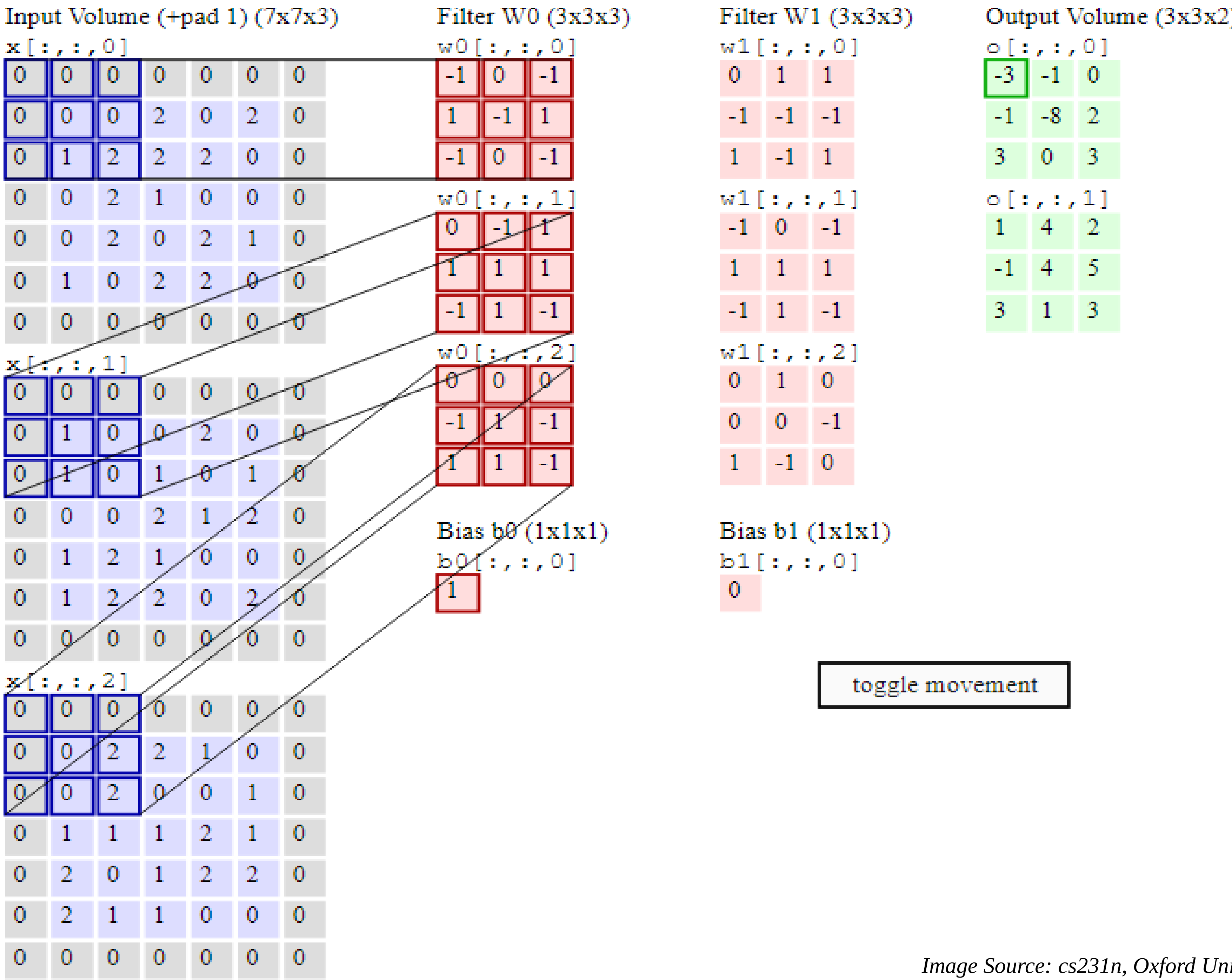


Image Source: cs231n, Oxford University

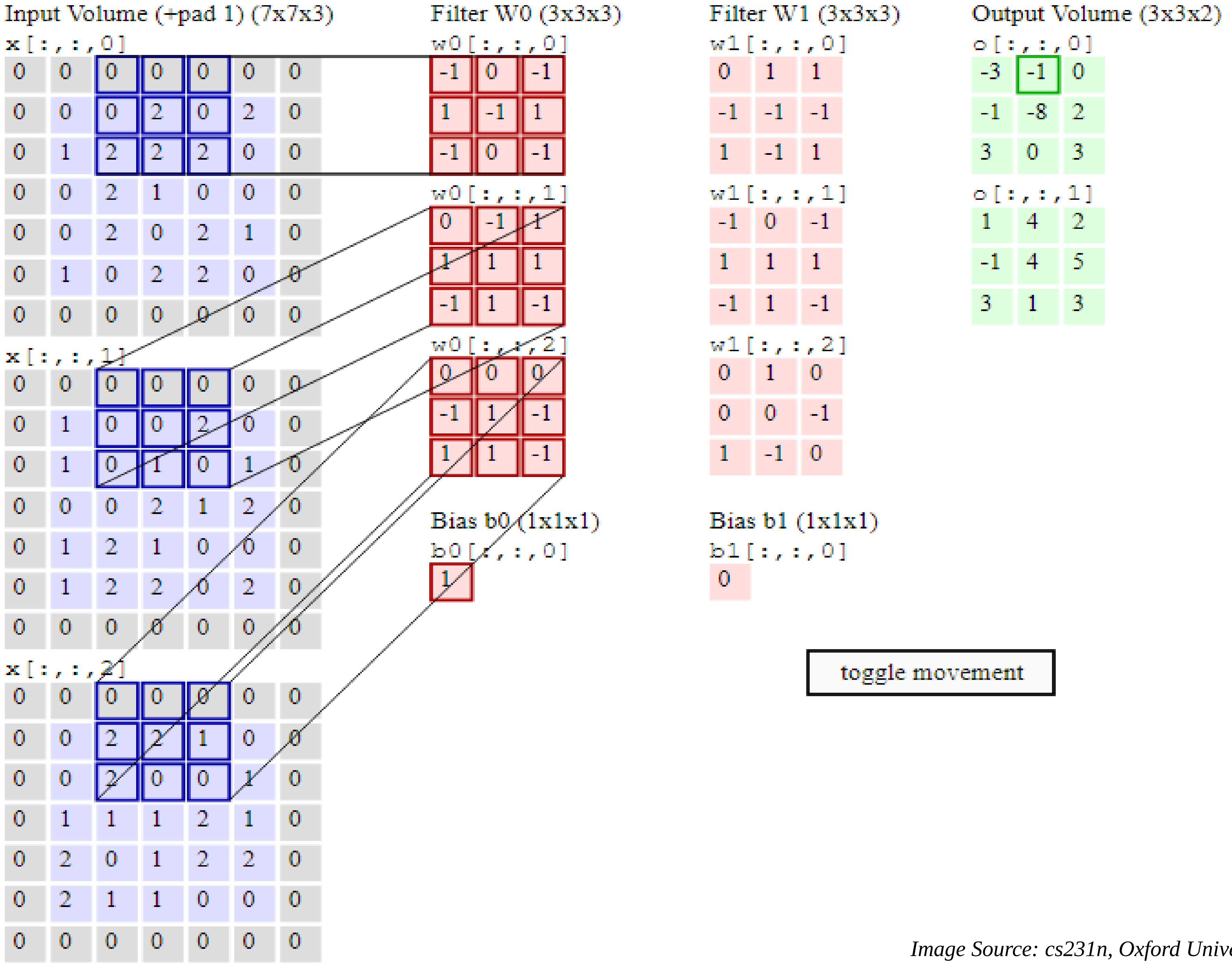


Image Source: cs231n, Oxford University

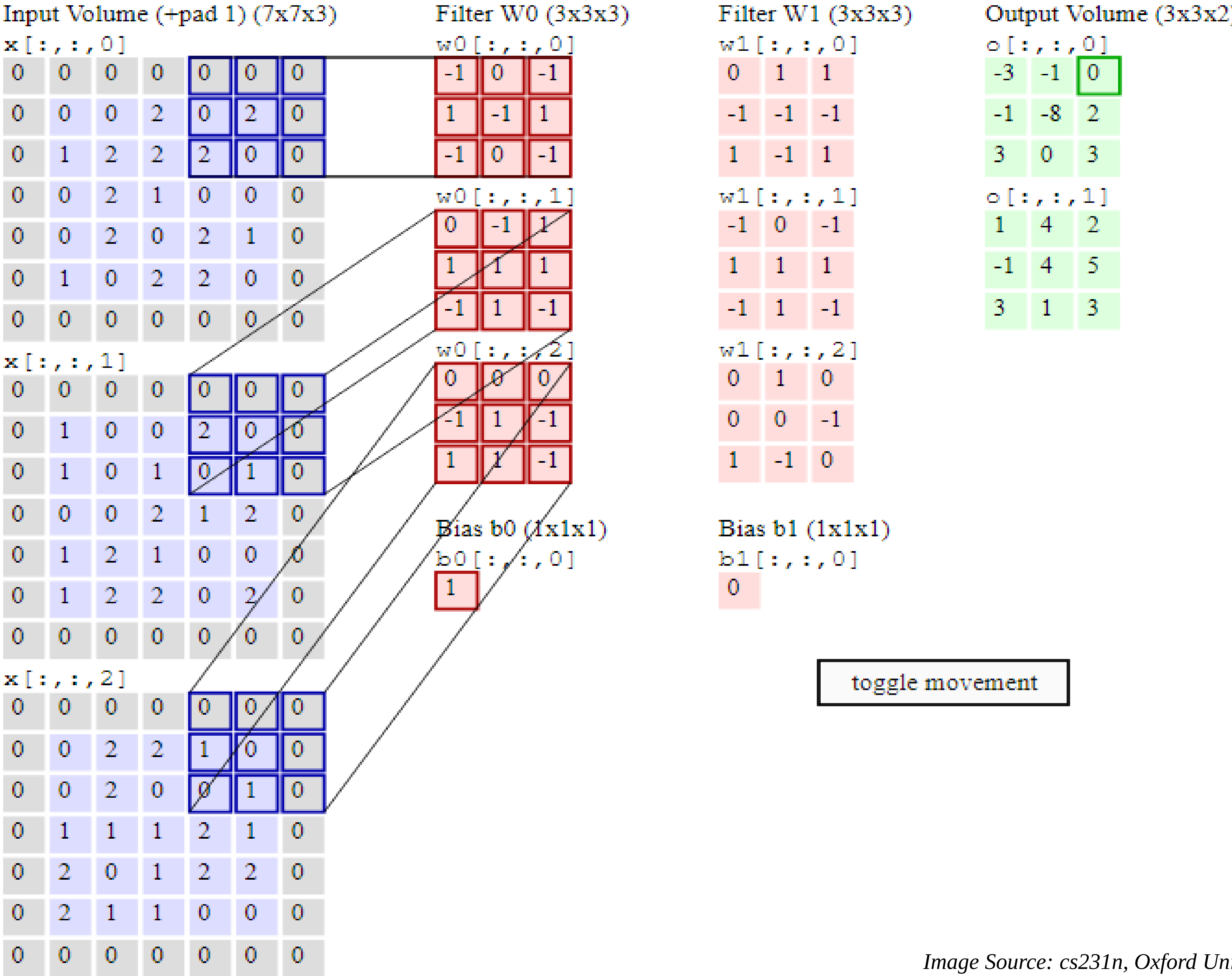
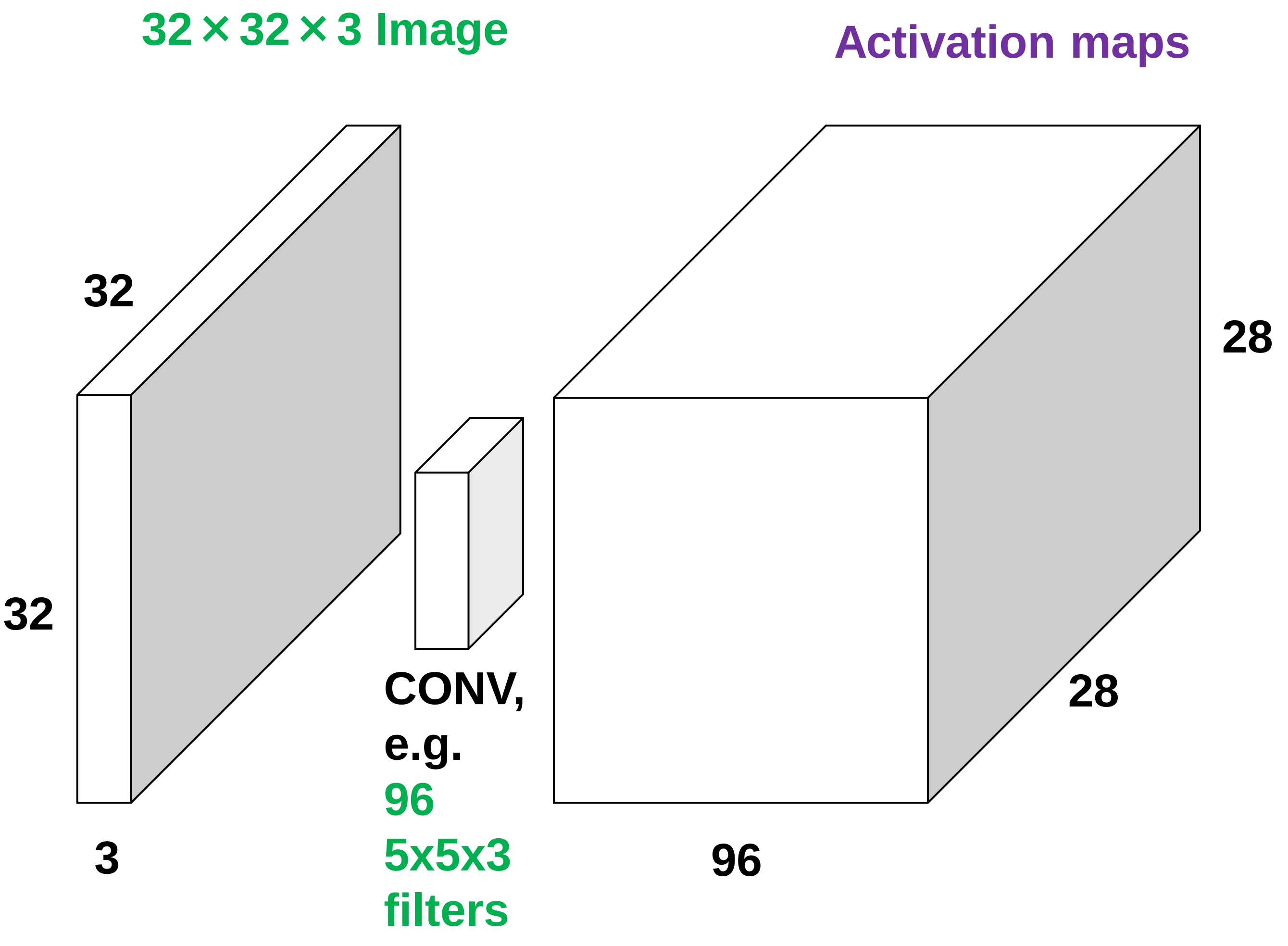


Image Source: cs231n, Oxford University

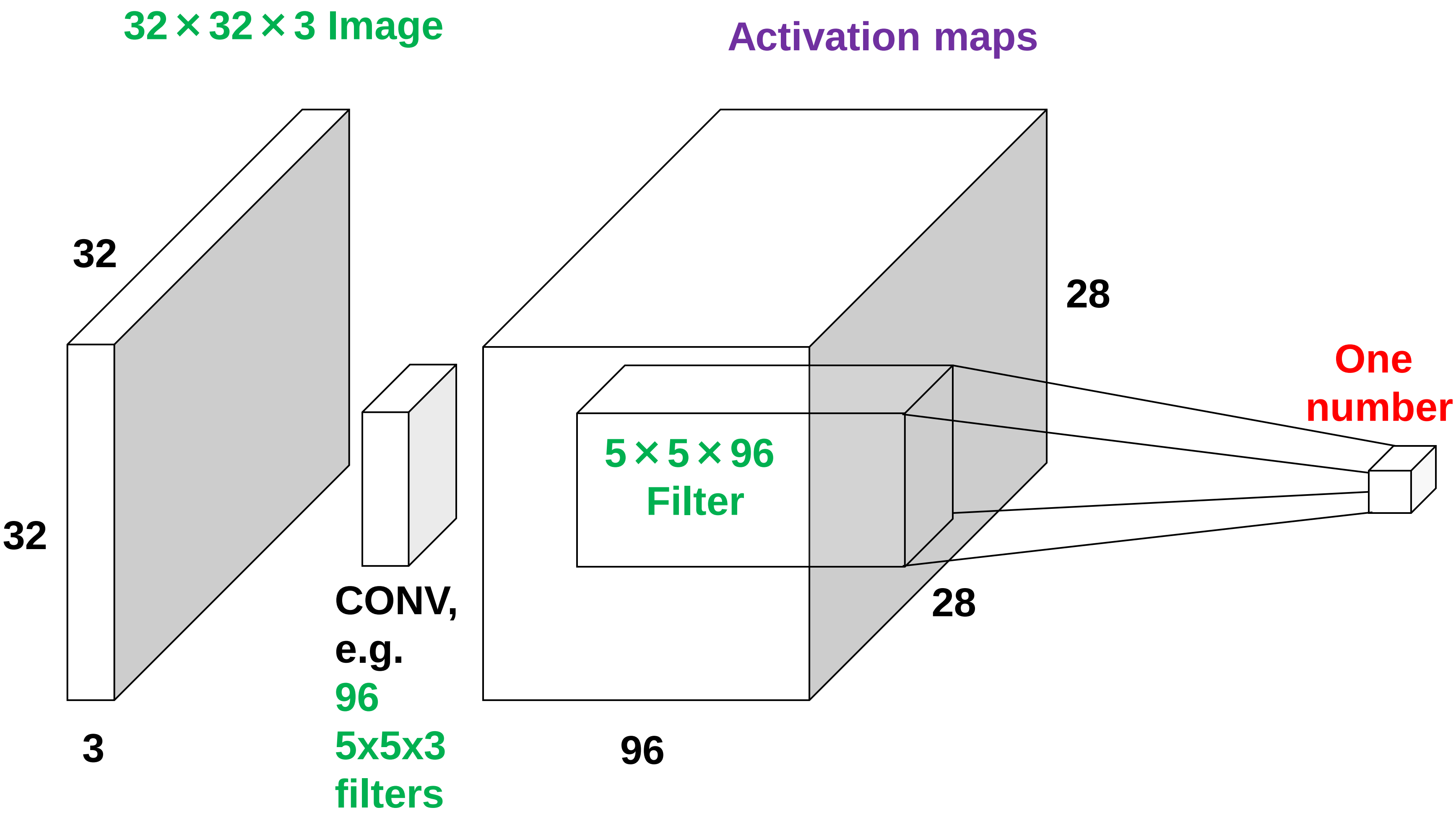
Convolutional Layer

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



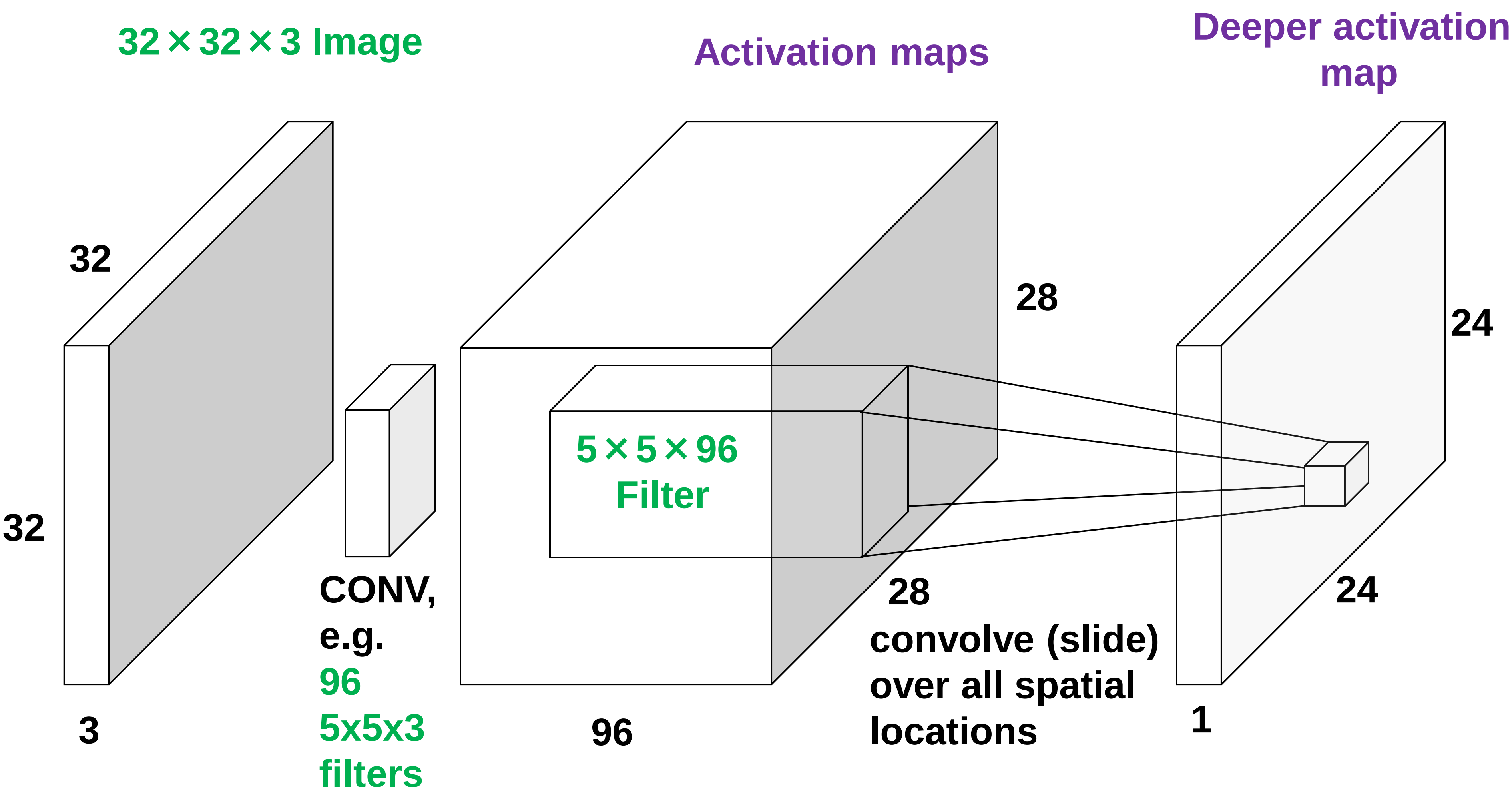
Convolutional Layer

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



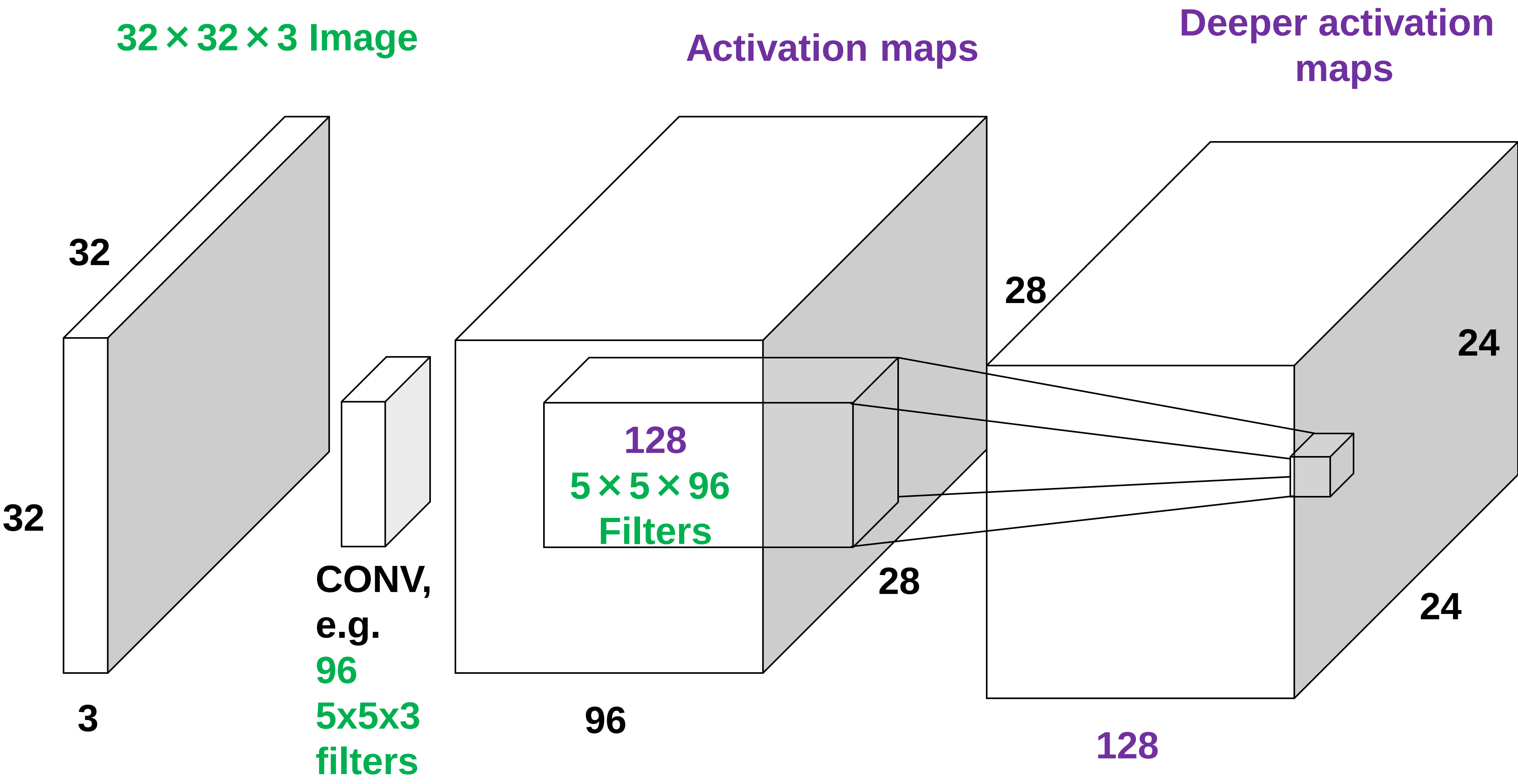
Convolutional Layer

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



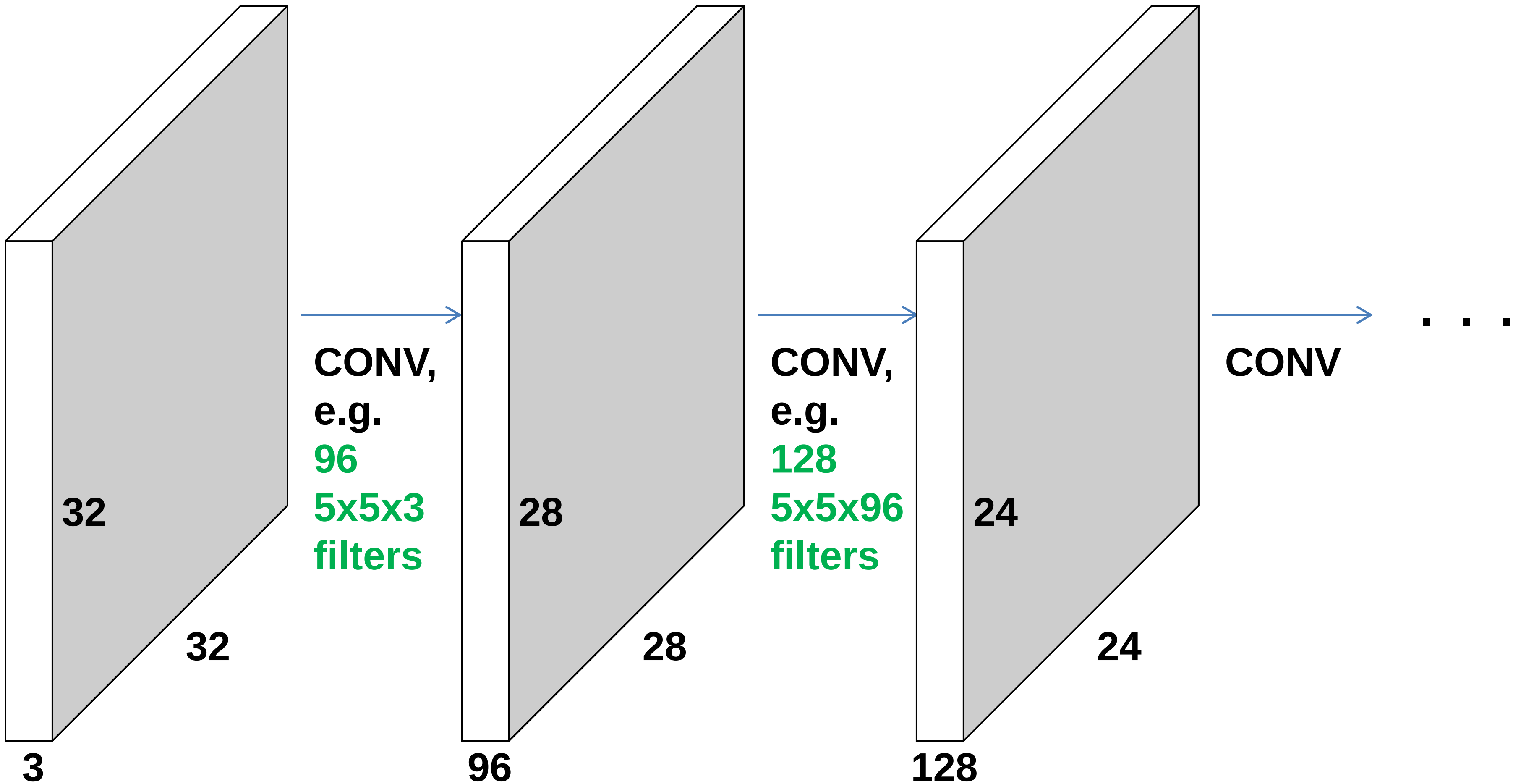
Convolutional Layer

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



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

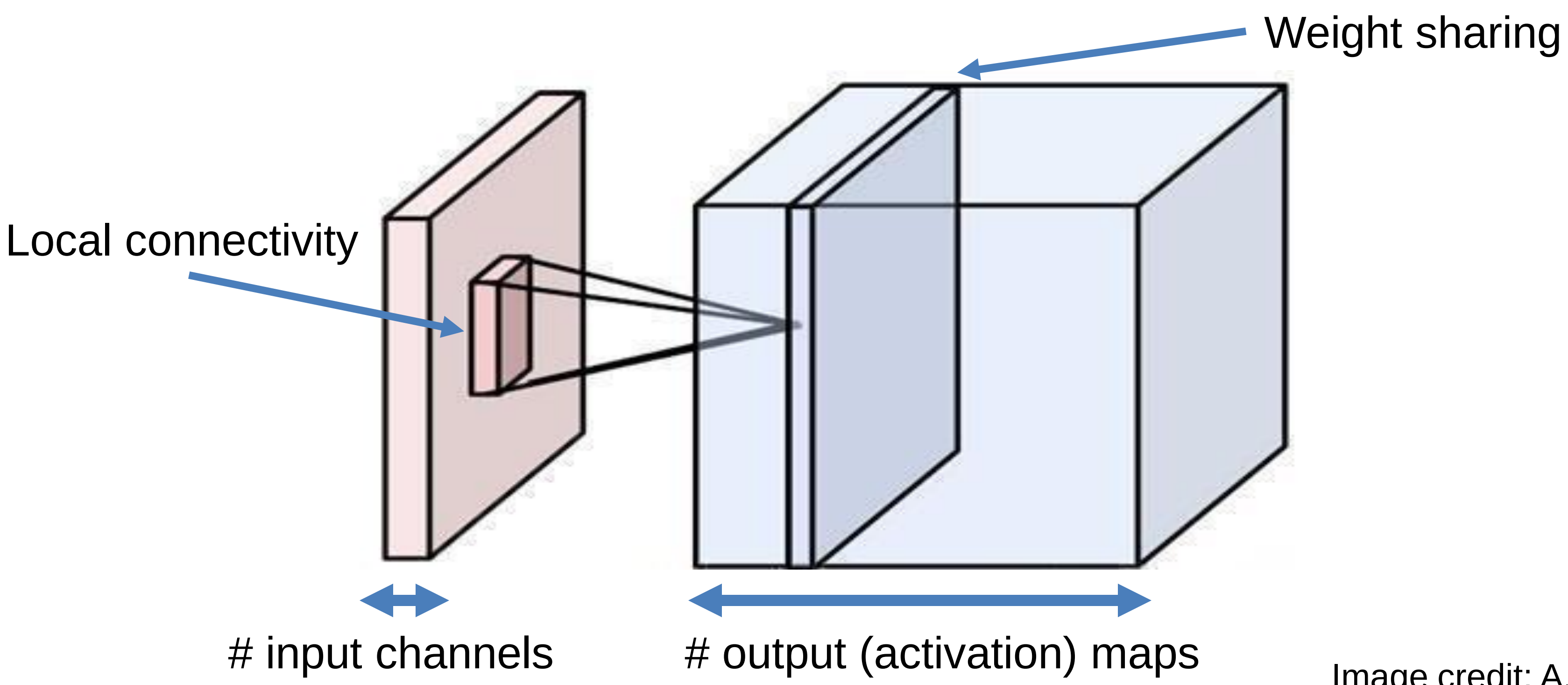
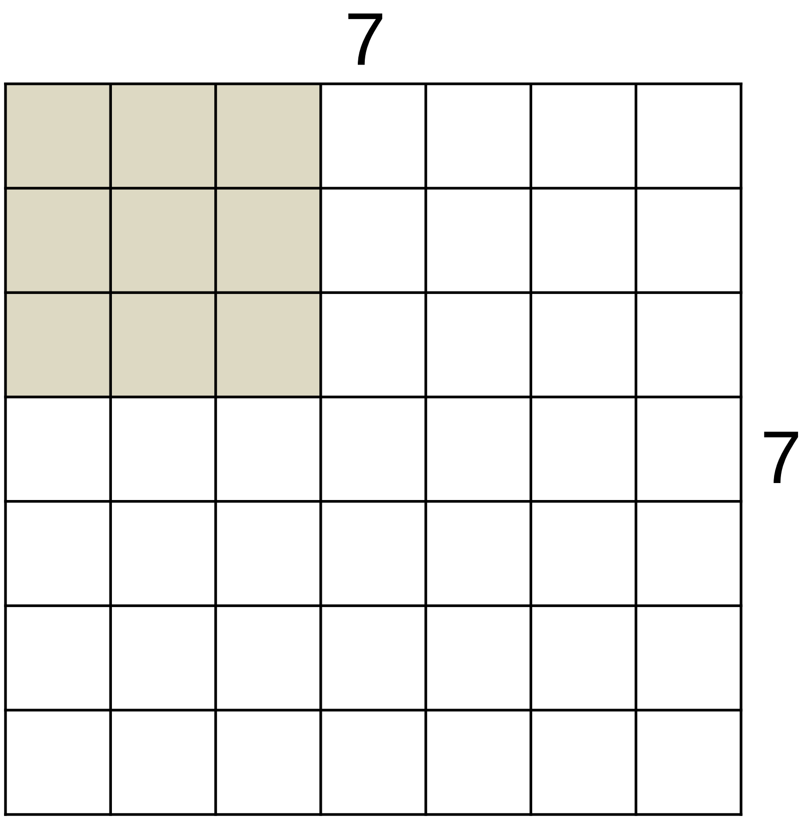


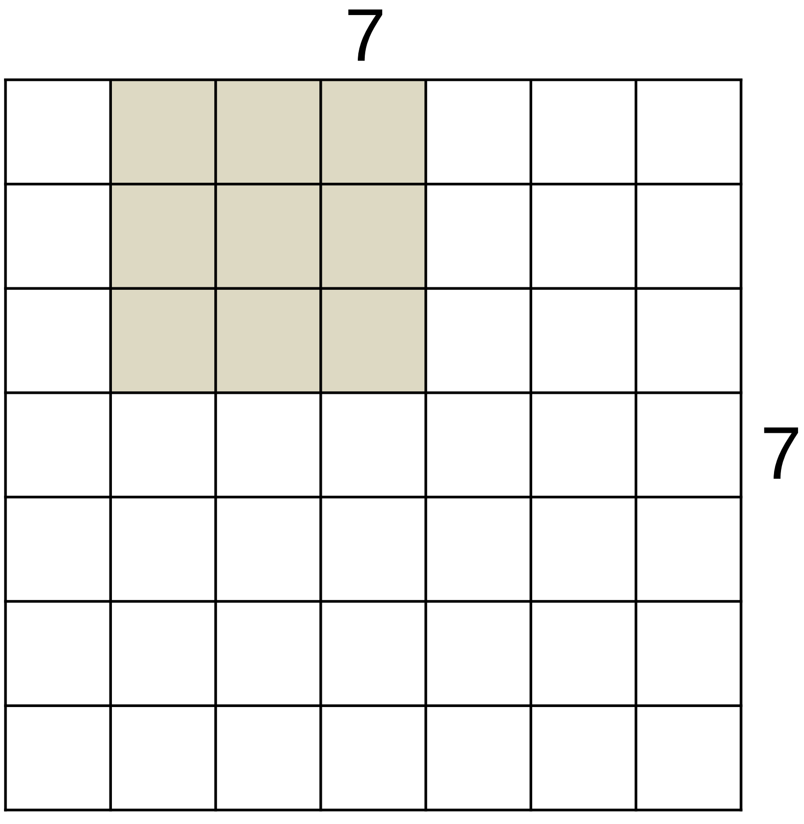
Image credit: A. Karpathy

A closer look at spatial dimensions



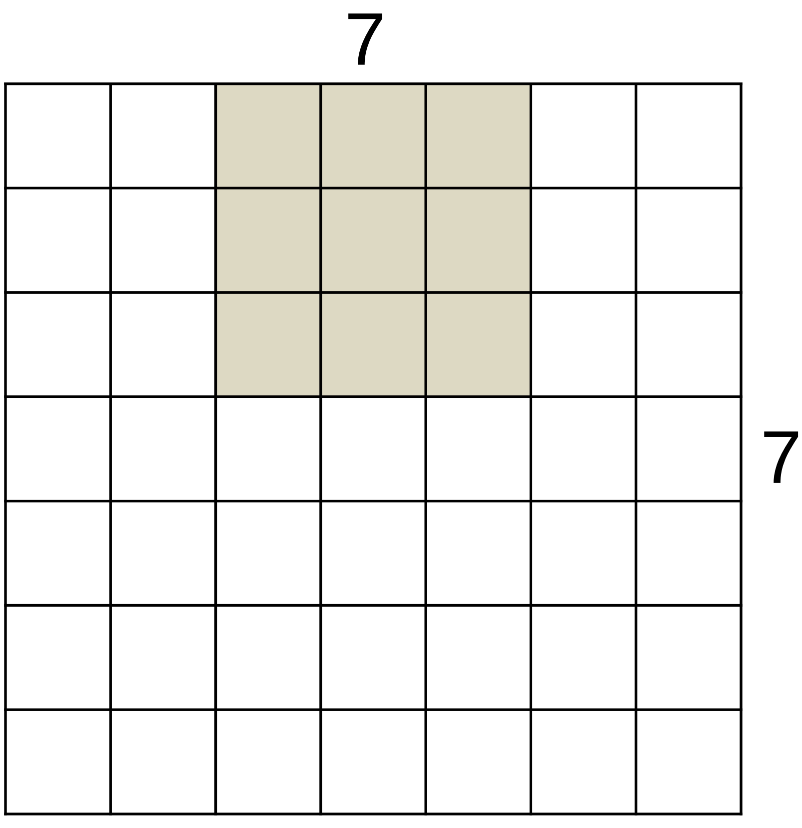
7×7 input (spatially)
assume 3×3 filter

A closer look at spatial dimensions



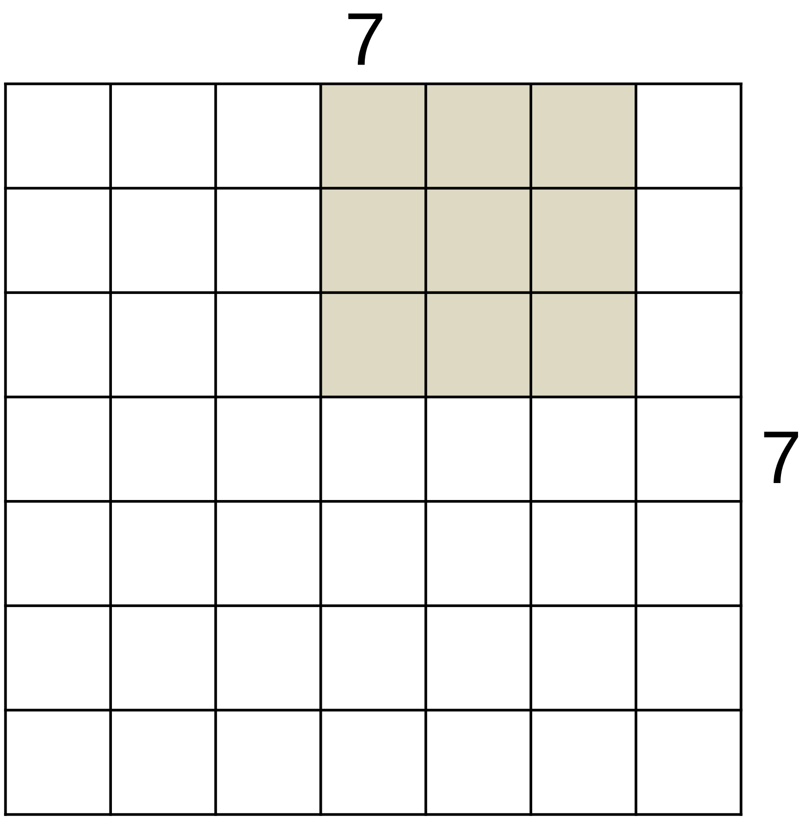
7×7 input (spatially)
assume 3×3 filter

A closer look at spatial dimensions



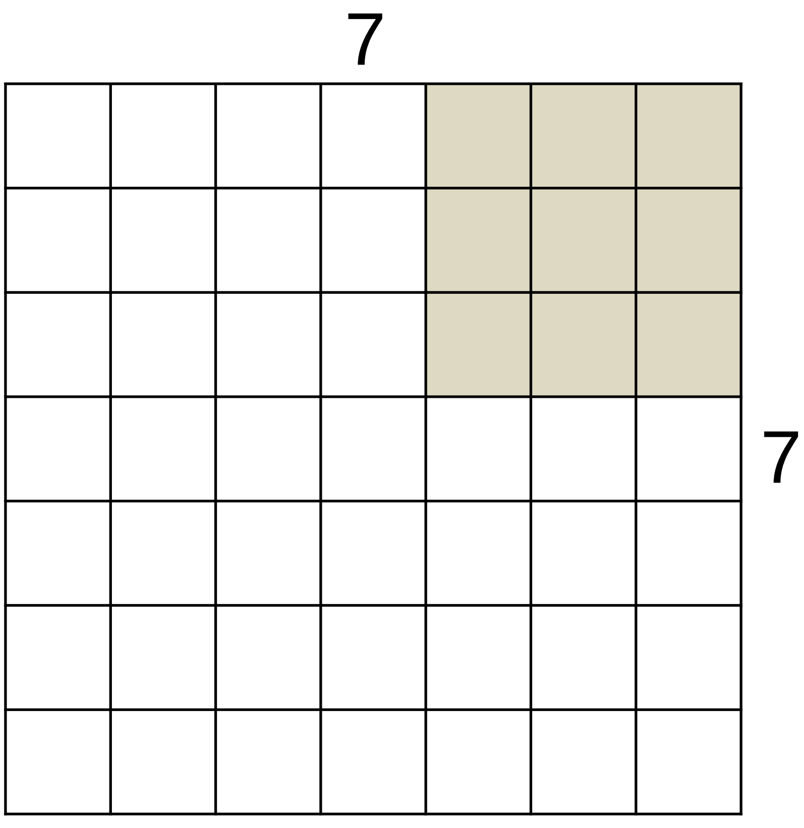
7×7 input (spatially)
assume 3×3 filter

A closer look at spatial dimensions



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

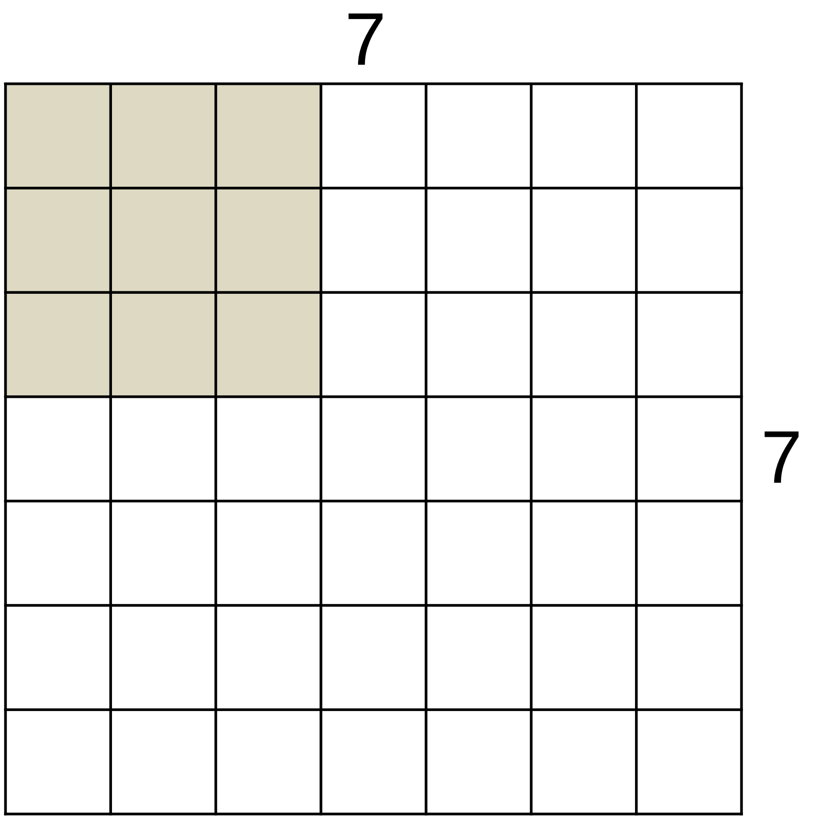
A closer look at spatial dimensions



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

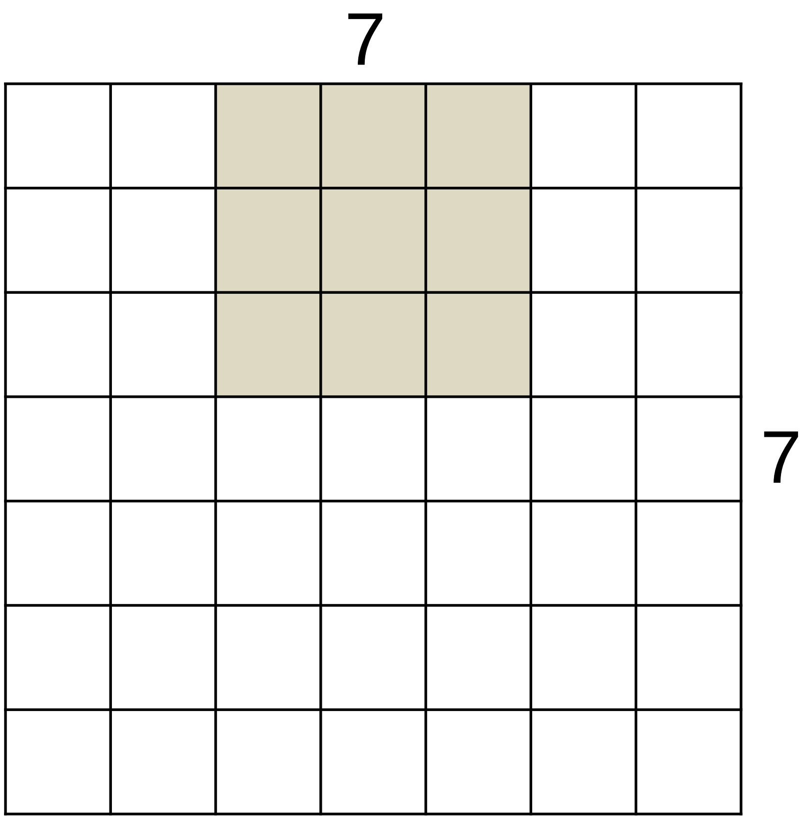
5×5 output

A closer look at spatial dimensions



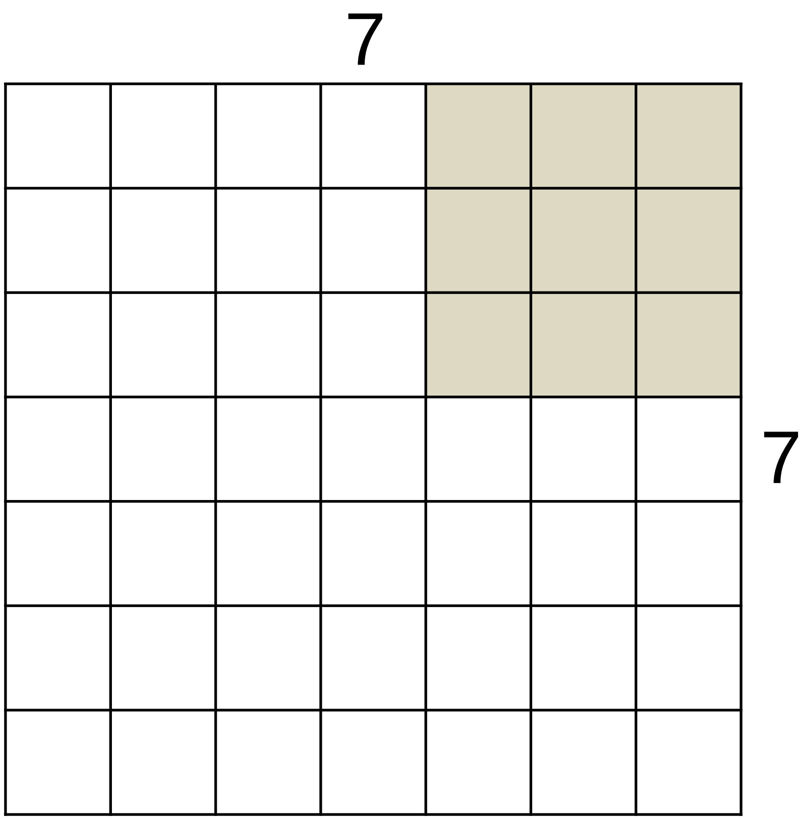
7×7 input (spatially)
assume 3×3 filter
applied with **stride 2**

A closer look at spatial dimensions



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

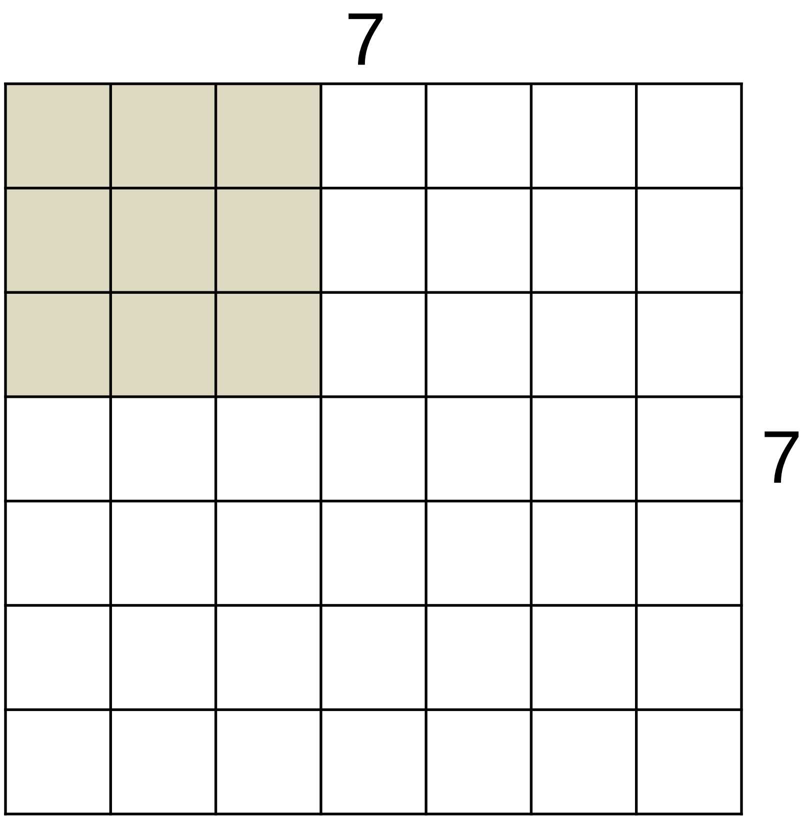
A closer look at spatial dimensions



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

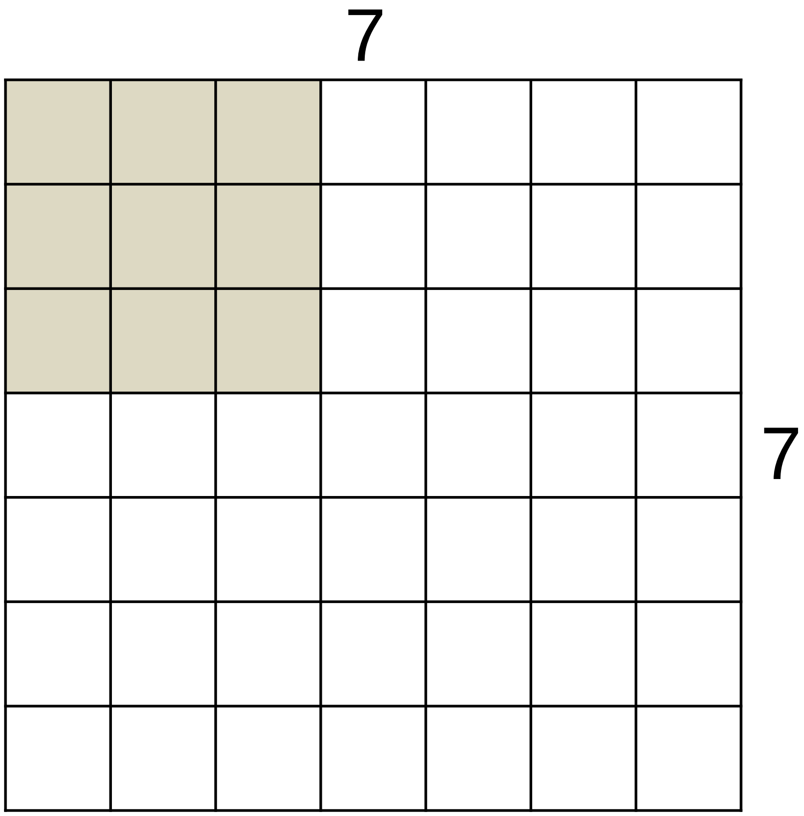
⇒ 3×3 output

A closer look at spatial dimensions



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

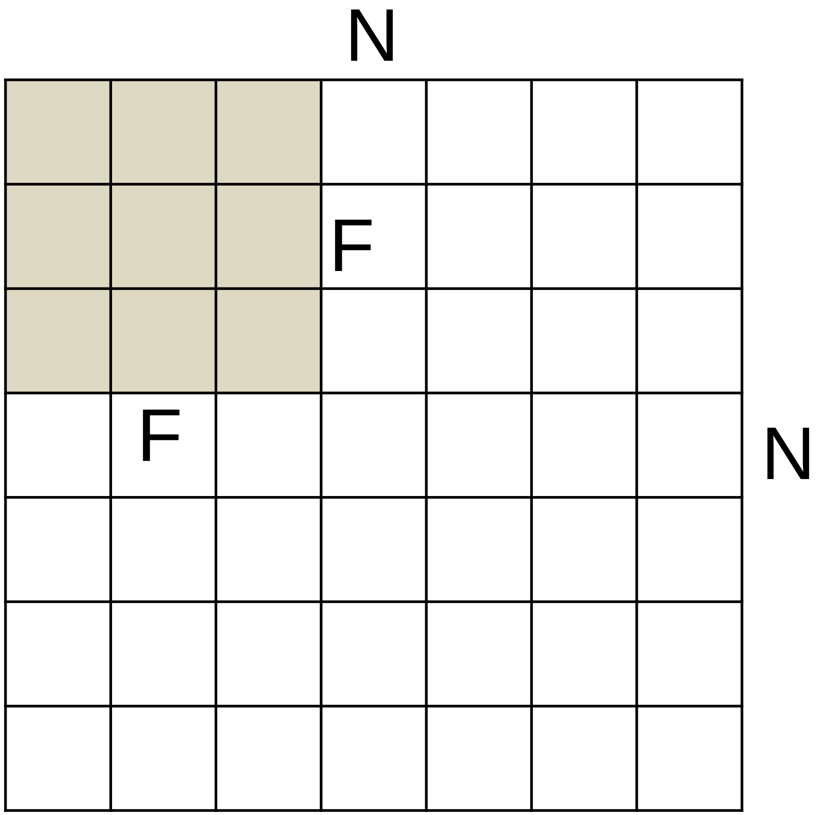
A closer look at spatial dimensions



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.

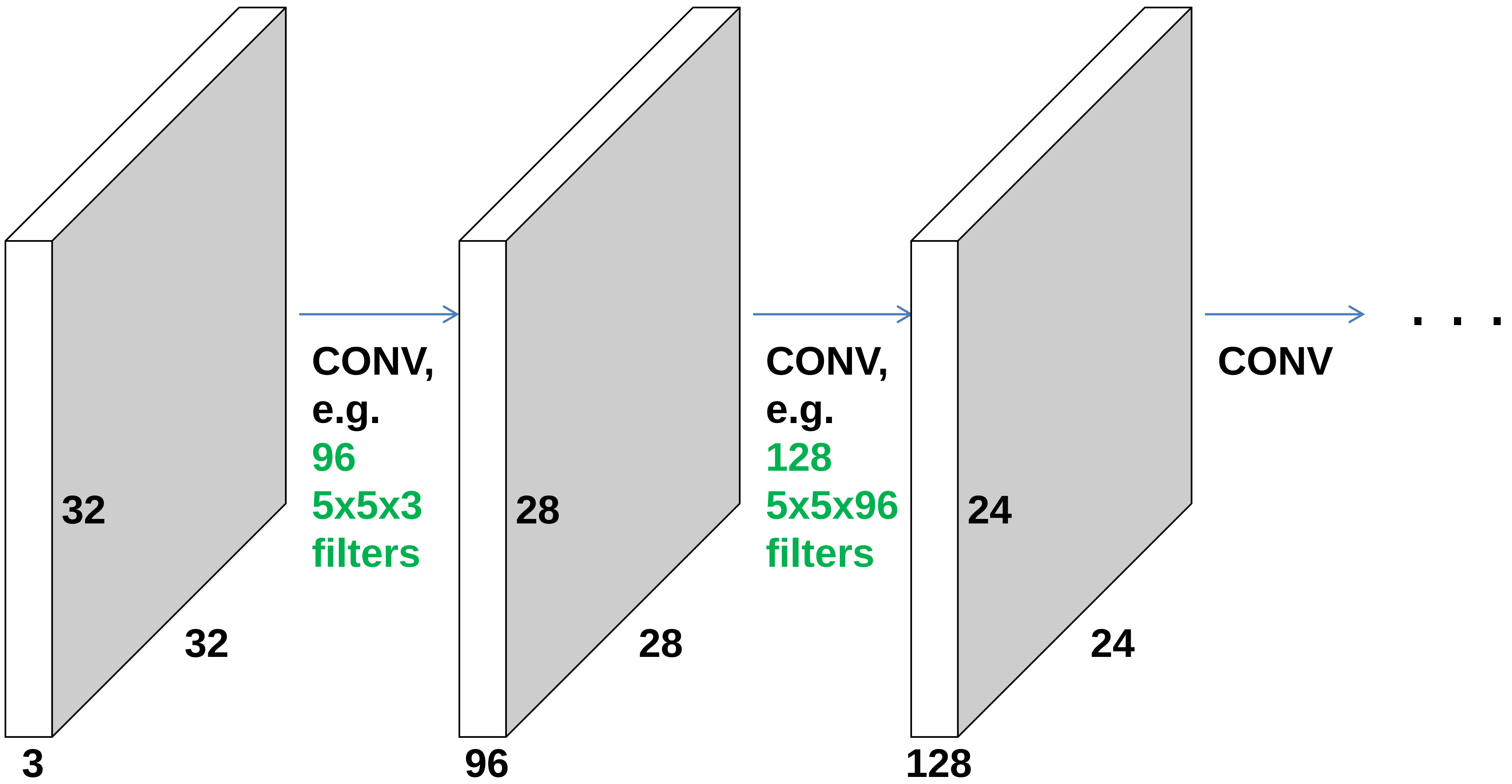
A closer look at spatial dimensions



Output size
(N - F) / stride + 1

e.g. N = 7, F = 3
stride 1 => (7 - 3)/1 + 1 = 5
stride 2 => (7 - 3)/2 + 1 = 3
stride 3 => (7 - 3)/3 + 1 = 2.33

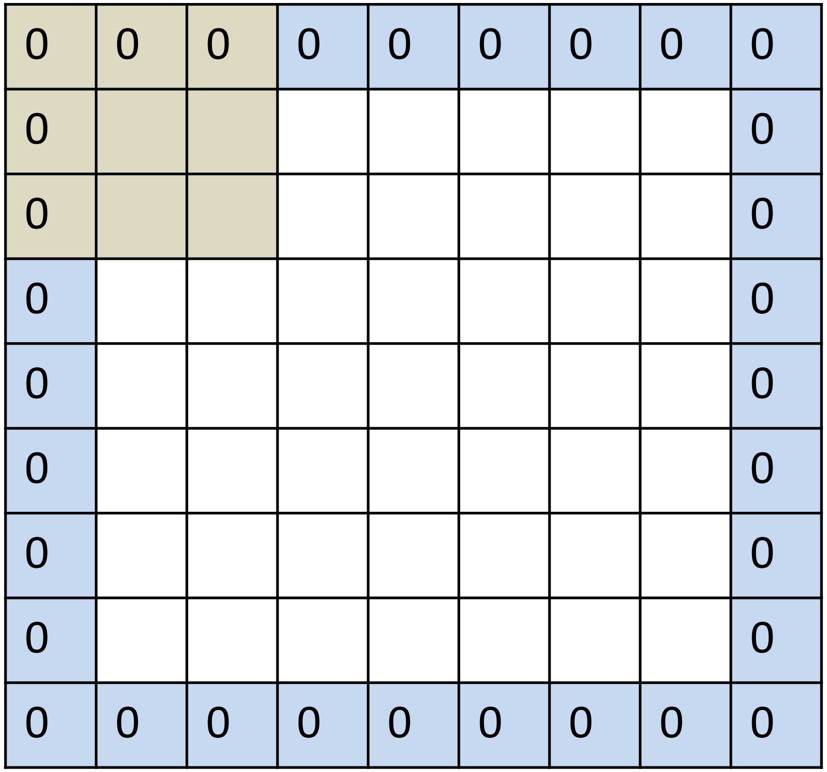
A closer look at spatial dimensions



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.

Source: cs231n, Stanford University

In practice: common to zero pad



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

What is the output dimension?

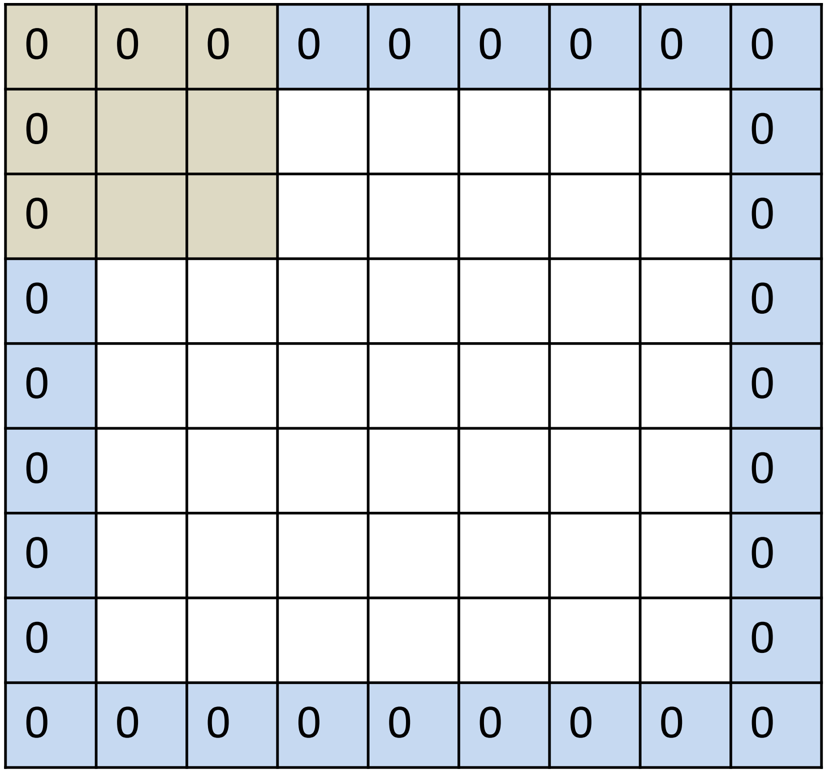
In practice: common to zero pad

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



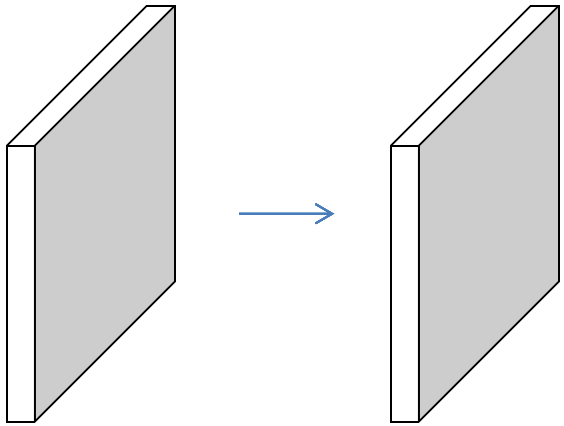
e.g. **input 7×7** (spatially)
3×3 filter, applied with **stride 1**
pad with 1 pixel border

7×7 Output
in general, common to see CONV layers with stride 1, filters of size F×F, and zero-padding with (F-1)/2. (will preserve size spatially)
e.g.
F = 3 => zero pad with 1
F = 5 => zero pad with 2
F = 7 => zero pad with 3

Example

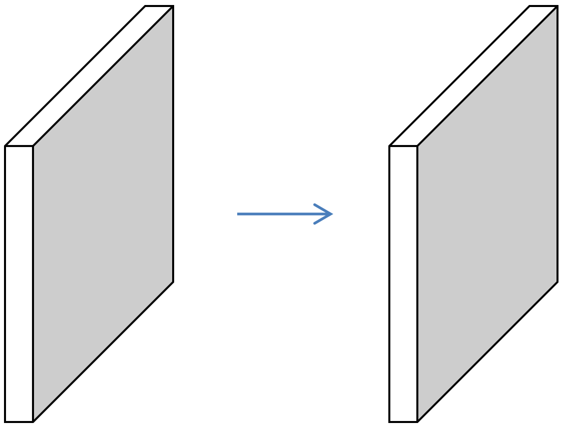
Input volume: **32x32x3**

10 5x5 filters with stride 1, pad 2



Output volume size: ?

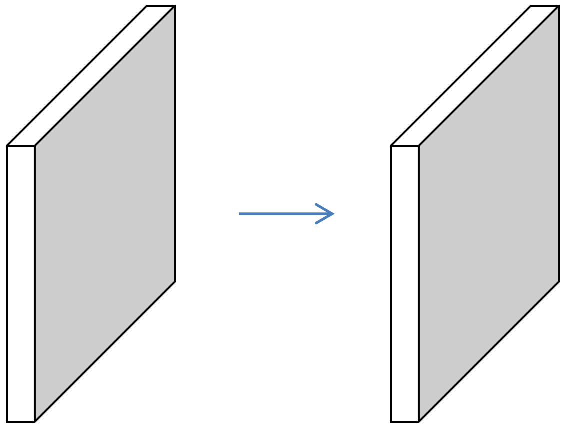
Example



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

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

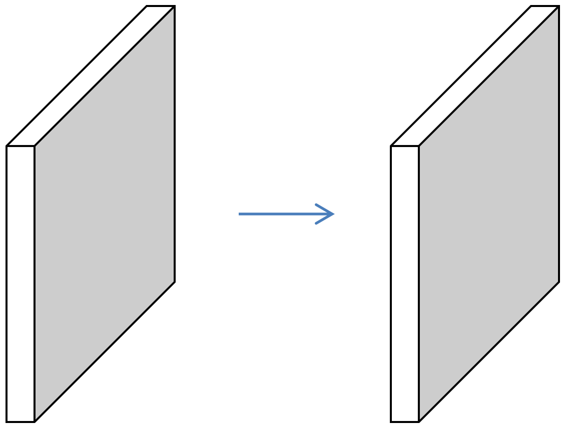
Example



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)

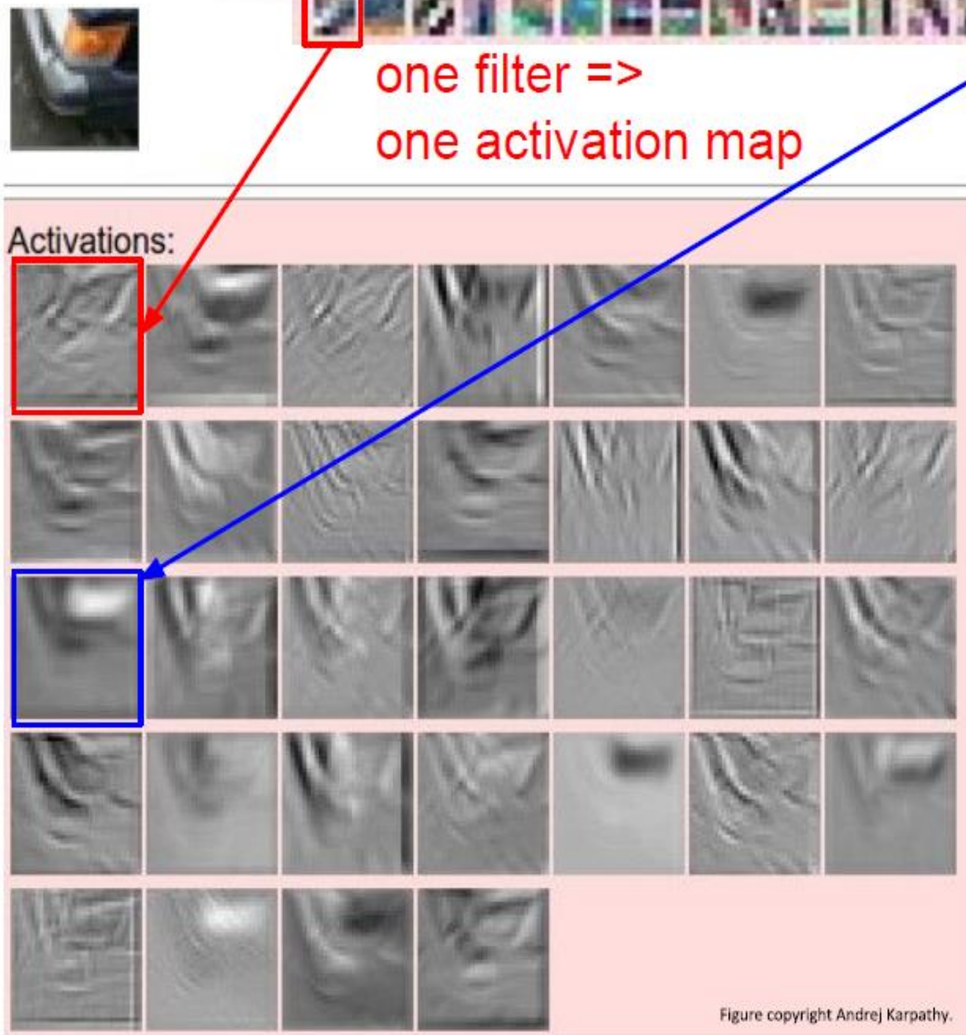
=> 76*10 = **760**

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.

Source: cs231n, Stanford University

Convolution as feature extraction



example 5x5 filters
(32 total)

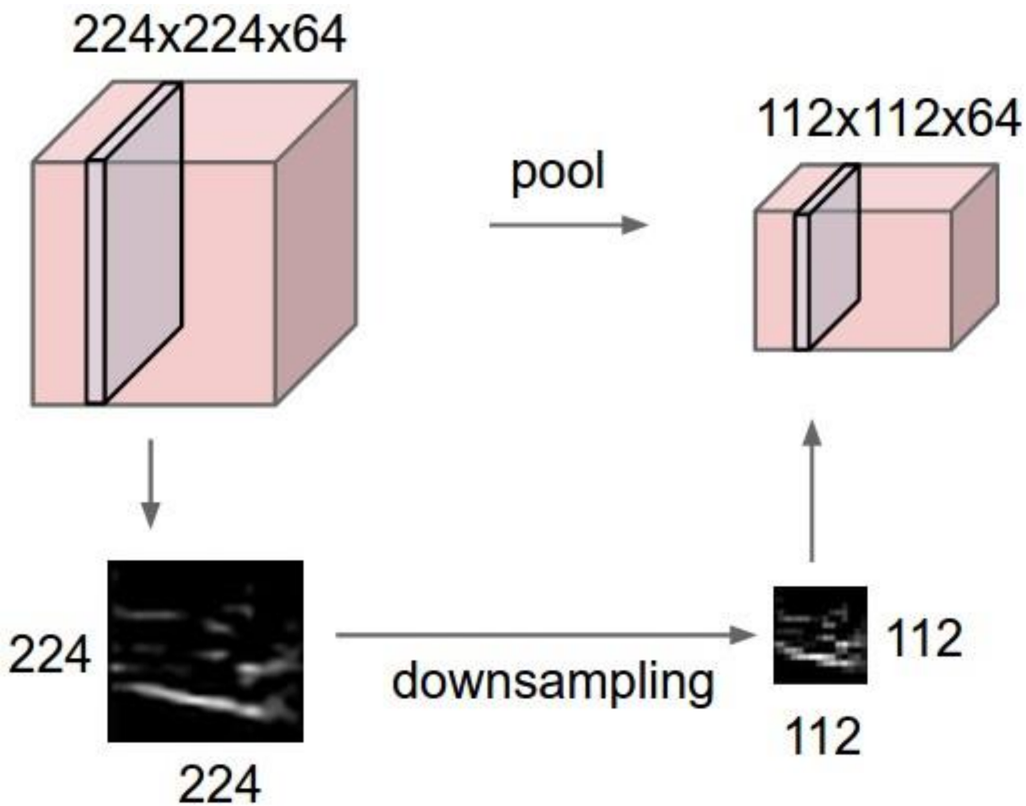
We call the layer convolutional because it is related to convolution of two signals:

$$f[x,y] * g[x,y] = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f[n_1,n_2] \cdot g[x-n_1, y-n_2]$$

elementwise multiplication and sum of a filter and the signal (image)

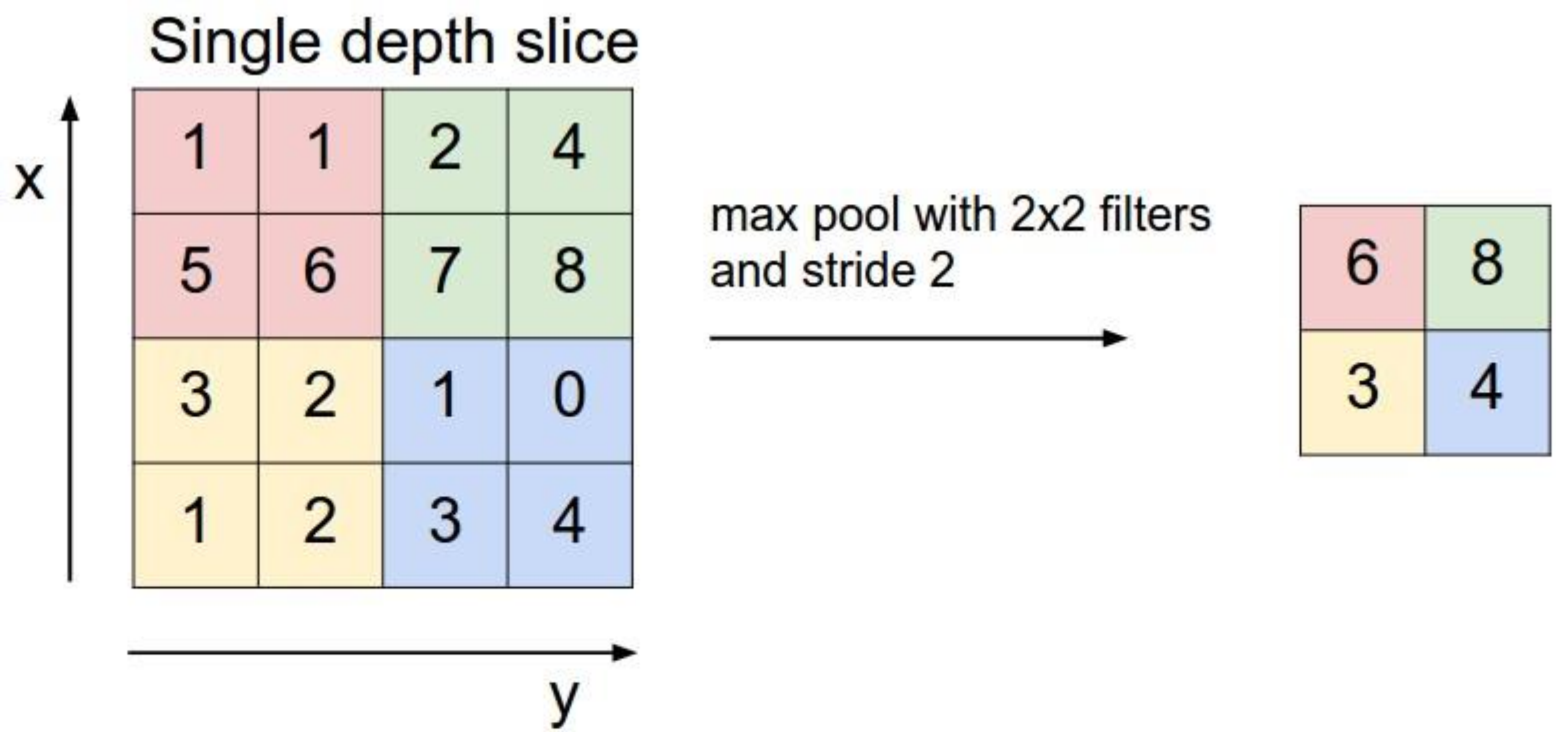
Pooling Layer

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



Source: cs231n, Stanford University

Max Pooling



Source: cs231n, Stanford University

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

Source: cs231n, Stanford University

Fully Connected Layer

- Connect every neuron in one layer to every neuron in another layer

- Same as the traditional multi-layer perceptron neural network

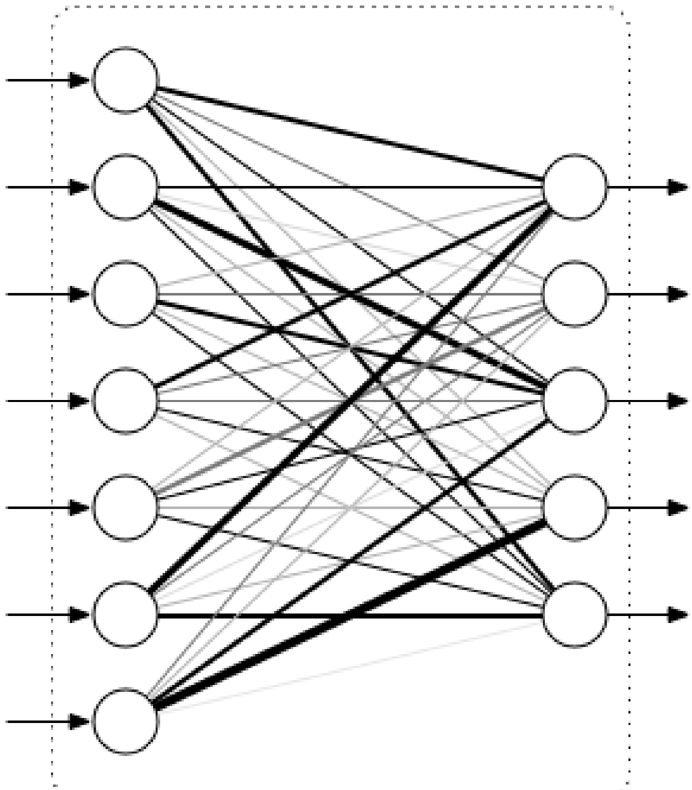


Image Source: machinethink.net

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

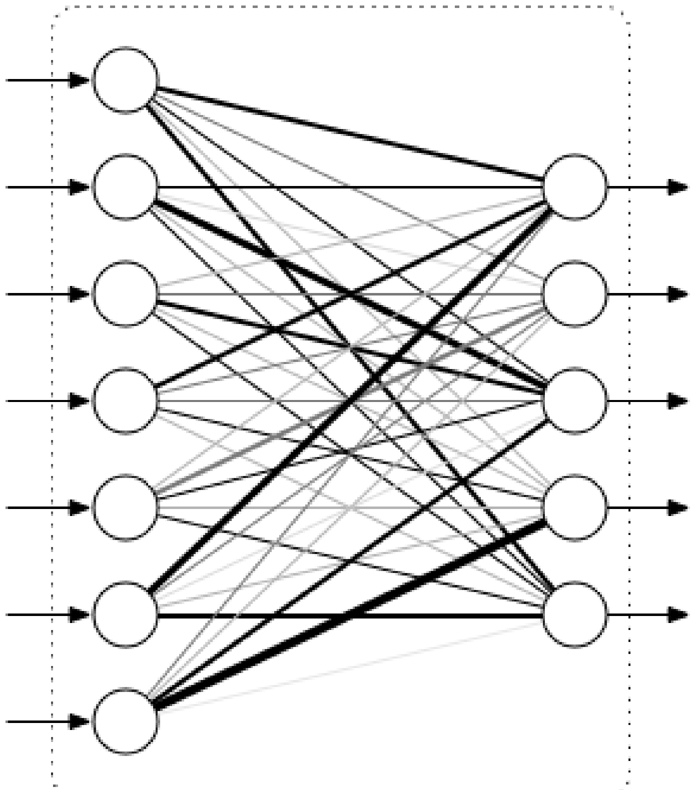


Image Source: machinethink.net

Loss/Classification Layer

- SVM Classifier (SVM Loss/Hinge Loss/Max-margin Loss)

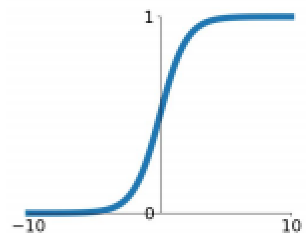
- Softmax Classifier (Softmax Loss/Cross-entropy Loss)

Non-linearity Layer

Activation Functions

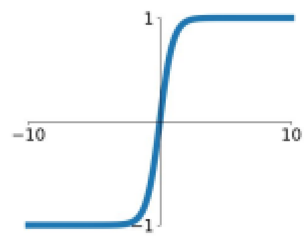
Sigmoid

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



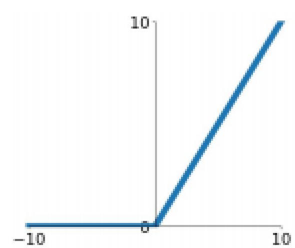
tanh

$\tanh(x)$



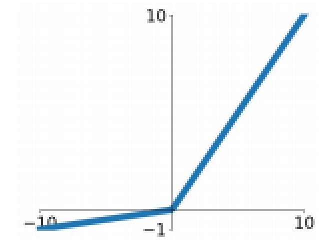
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

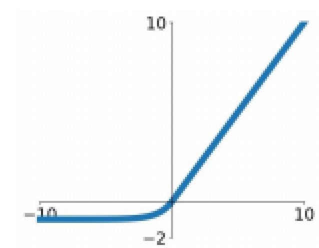


Maxout

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

ELU

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



Source: cs231n, Stanford University

A typical CNN structure

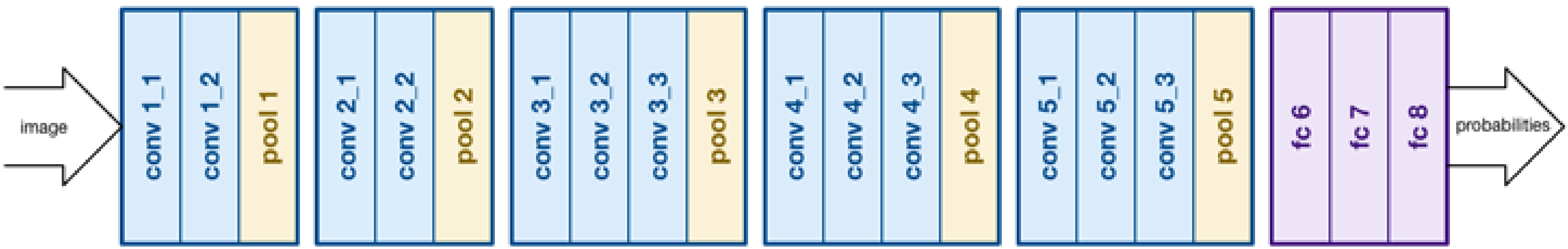
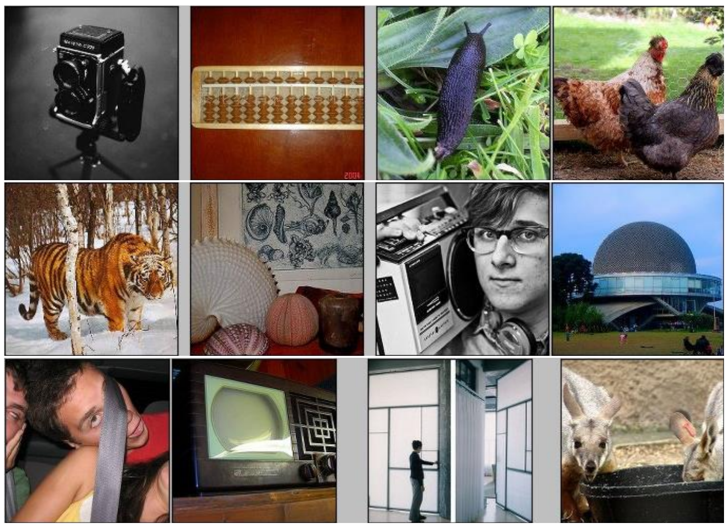


Image Source: machinethink.net

ImageNet Challenge

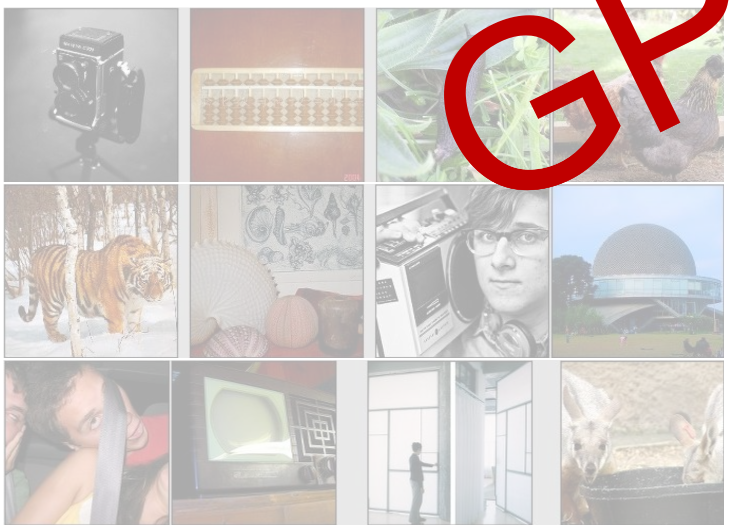


- ~14 million labeled images, 20k classes
- Images gathered from Internet
- Human labels via Amazon MTurk
- Challenge: 1.2 million training images, 1000 classes

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

ImageNet Challenge

IMAGENET

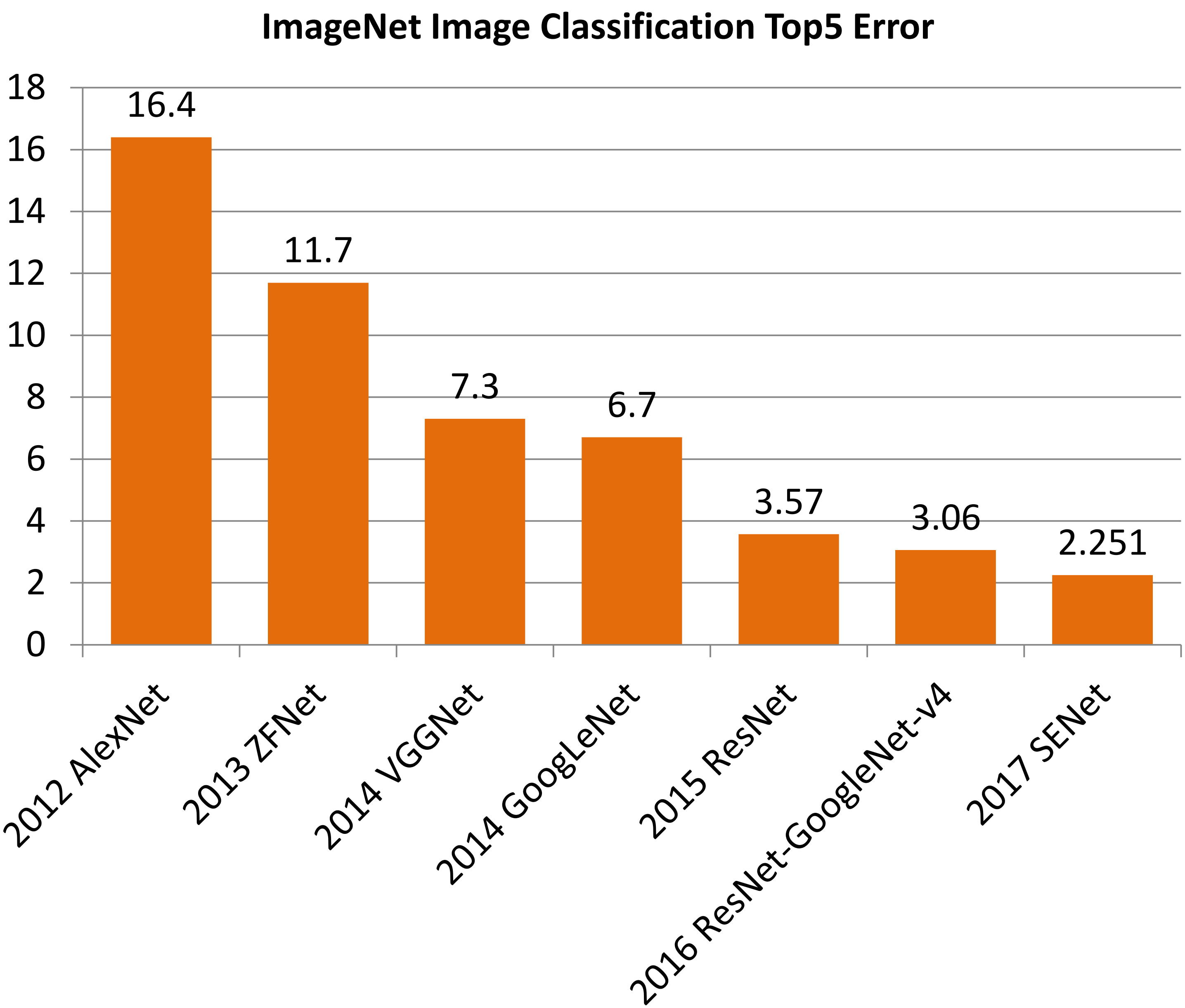


- ~14 million labeled images, 20k classes
- Images gathered from Internet
- Human labels via Amazon MTurk
- Challenge: 1.2 million training images, 1000 classes

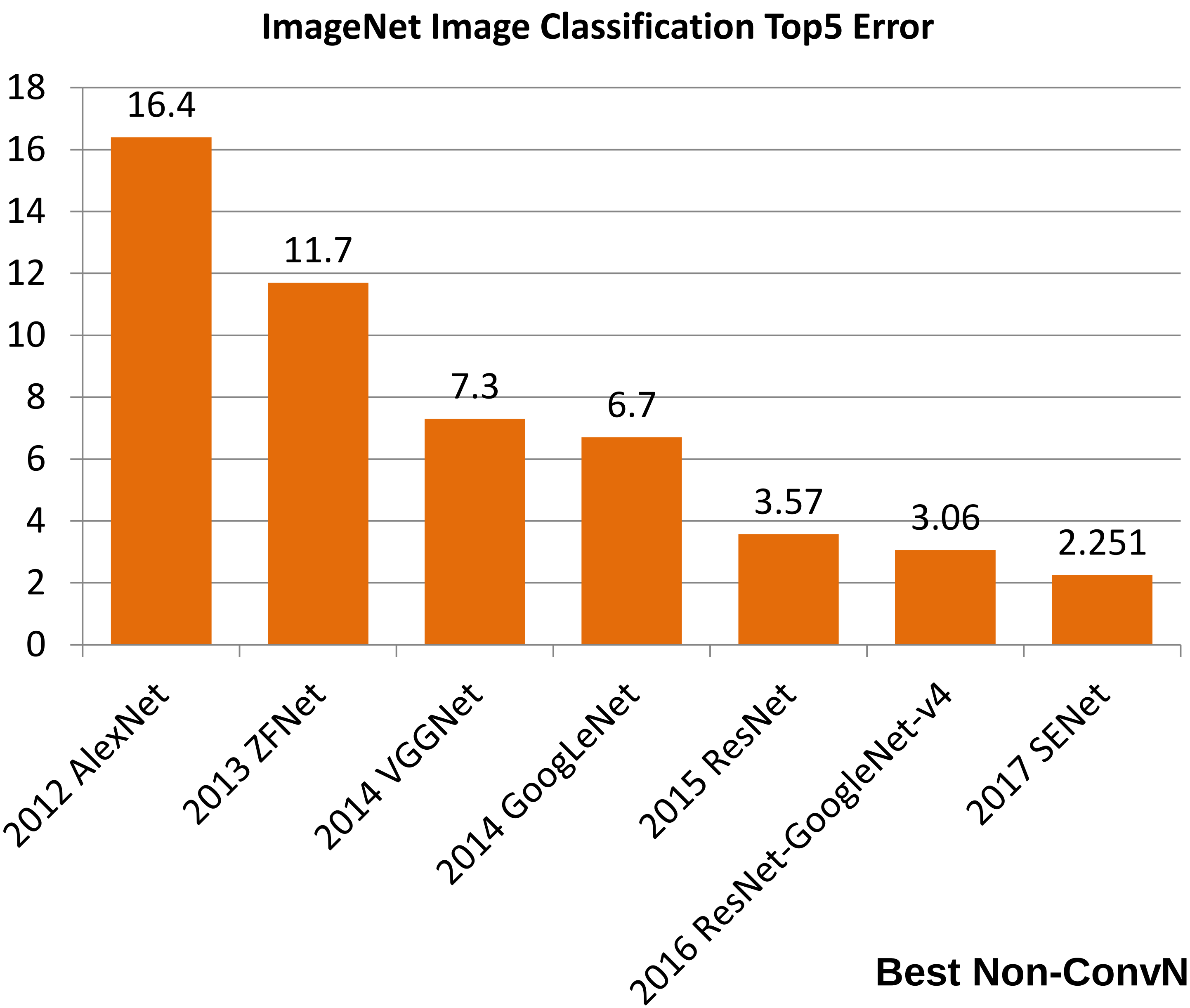
GPUs + Data

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

Progress on ImageNet Challenge



Progress on ImageNet Challenge



Best Non-ConvNet in 2012: 26.2%

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

Acknowledgements

- Thanks to the following researchers for making their teaching/research material online
  - Forsyth
  - Steve Seitz
  - Noah Snavely
  - J.B. Huang
  - Derek Hoiem
  - D. Lowe
  - A. Bobick
  - S. Lazebnik
  - K. Grauman
  - R. Zaleski
  - Antonio Torralba
  - Rob Fergus
  - Leibe
  - And many more ………..

Next Class

Training Aspects of CNN

- Activation Functions
- Dataset Preparation
- Data Preprocessing
- Weight Initialization
- Optimization Methods
- Learning Rate
- Transfer Learning
- Generalization