Indian Institute of Information Technology, Allahabad

CVBL IIIT Allahabad

Introduction to Machine Learning

Convolutional Neural Networks

By

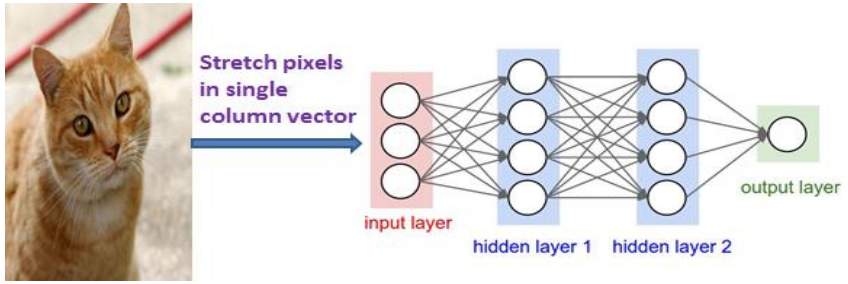
Dr. Shiv Ram Dubey
Associate Professor
Computer Vision and Biometrics Lab
Department of Information Technology
Indian Institute of Information Technology, Allahabad
Web: https://profile.iiita.ac.in/srdubey/

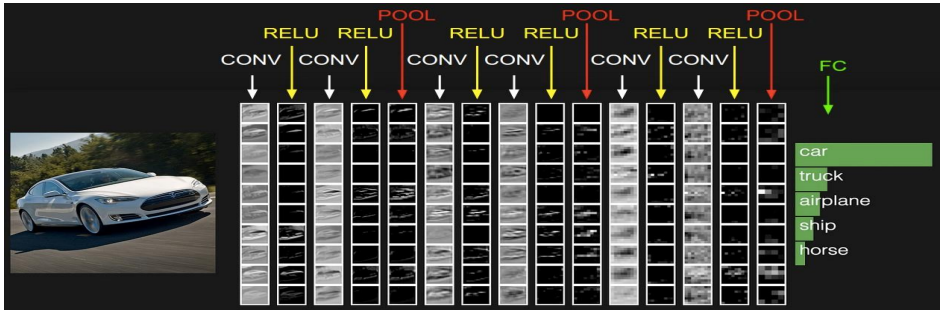
DISCLAIMER

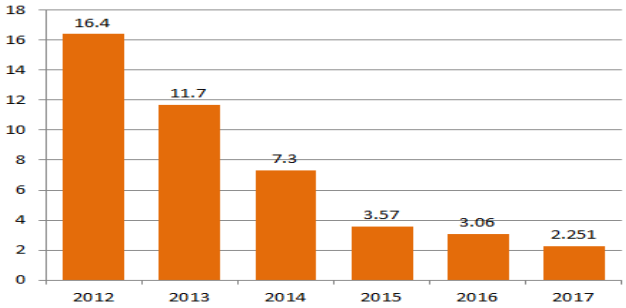
The content (text, image, and graphics) used in this slide are adopted from many sources for academic purposes. Broadly, the sources have been given due credit appropriately. However, there is a chance of missing out some original primary sources. The authors of this material do not claim any copyright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

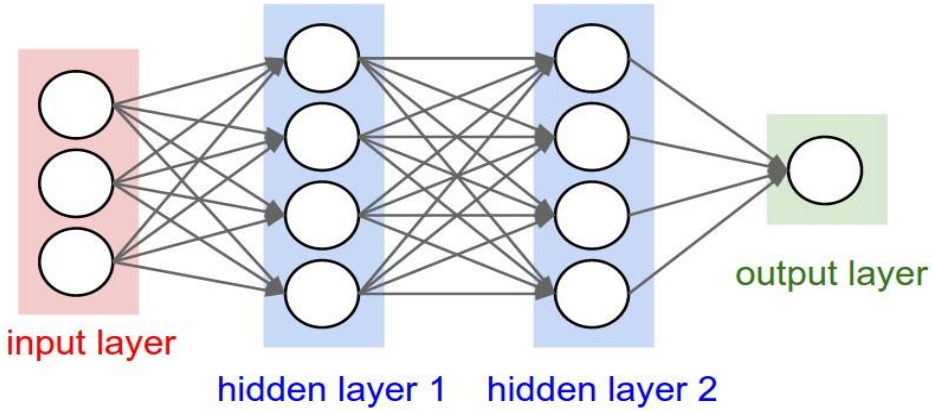






Multi-layer Neural Network & Image

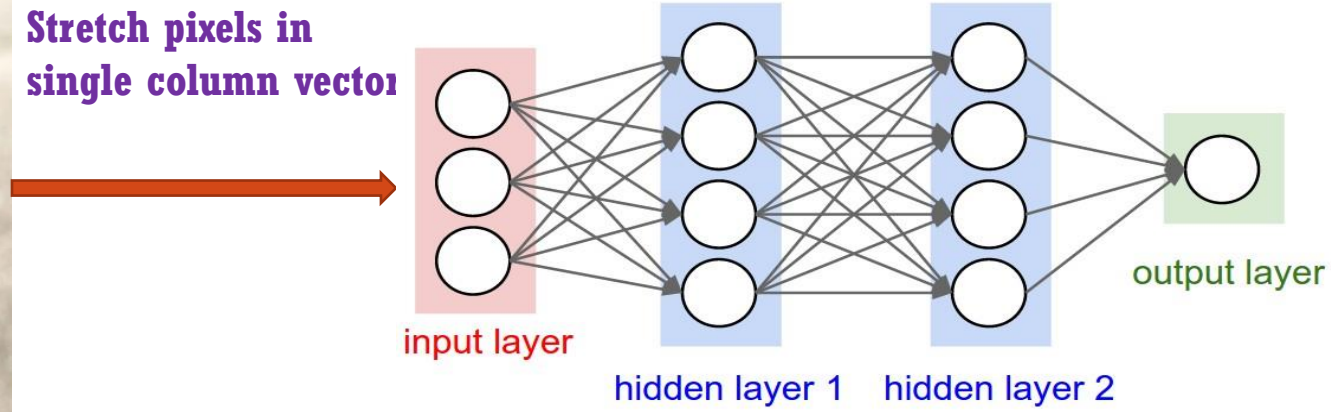




How to apply NN over Image?

Multi-layer Neural Network & Image

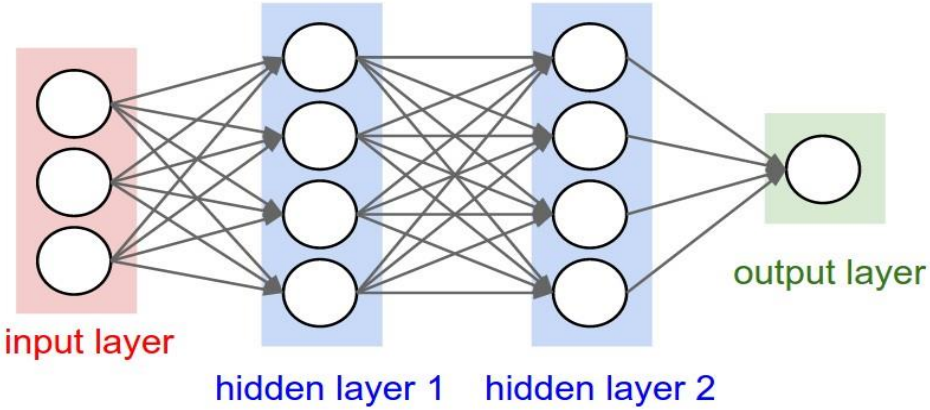




Multi-layer Neural Network & Image



Stretch pixels in
single column vecto

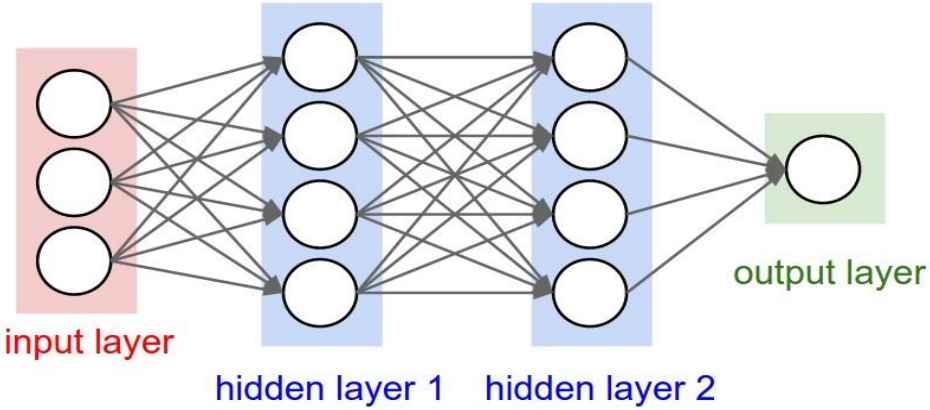


Problems ?

Multi-layer Neural Network & Image



Stretch pixels in single column vector



Problems:

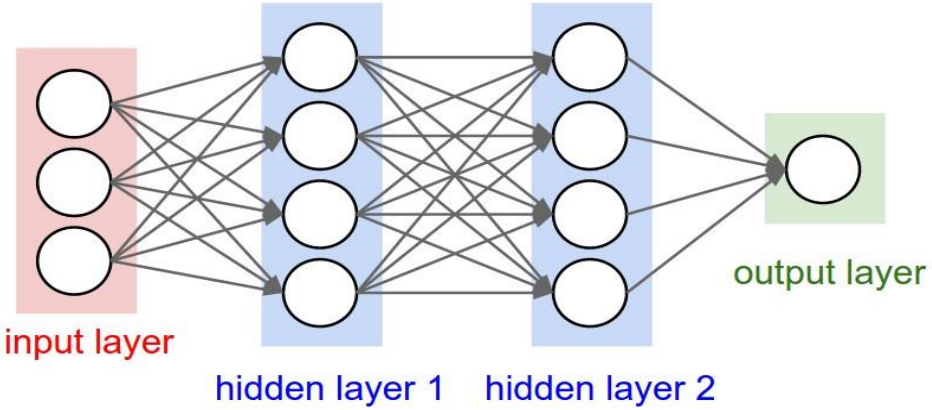
High dimensionality

Local relationship

Multi-layer Neural Network & Image



Stretch pixels in single column vector



Problems:

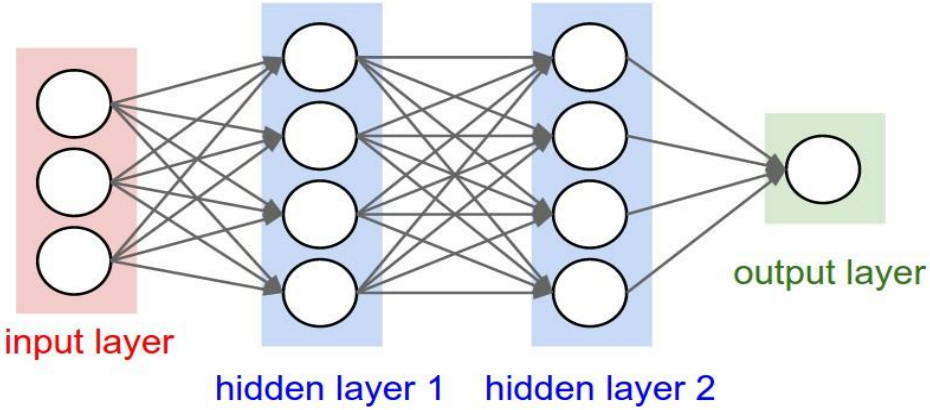
- High dimensionality
- Local relationship

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

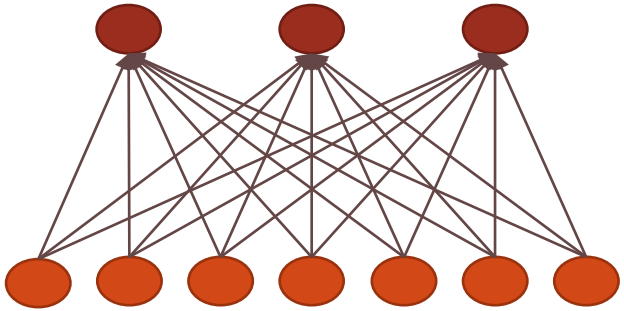
  CNN,

  ConvNet,

  DCN

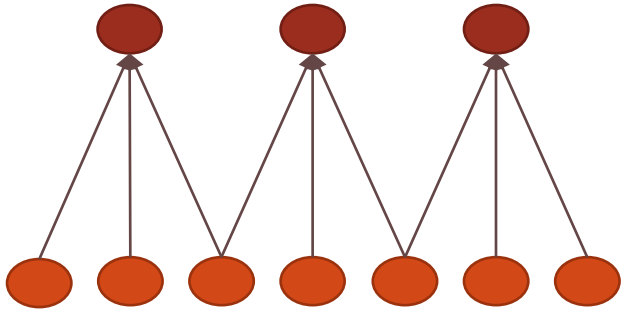
- CNN = a multi-layer neural network with
  1. Local connectivity
  2. Weight sharing

CNN: Local Connectivity



Hidden layer

Input layer

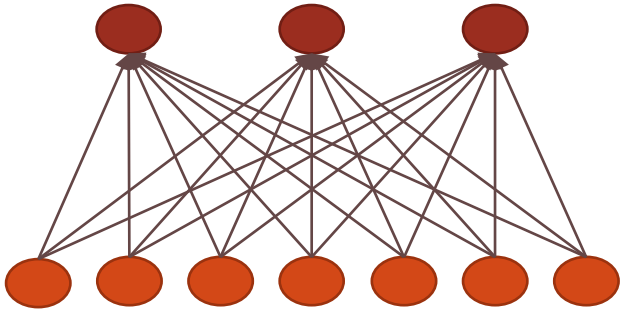


Global connectivity

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

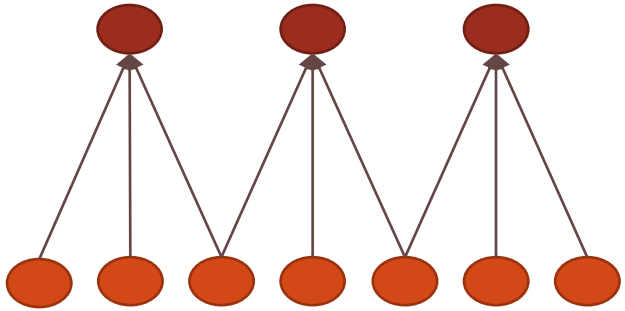
Local connectivity

CNN: Local Connectivity



Hidden layer

Input layer

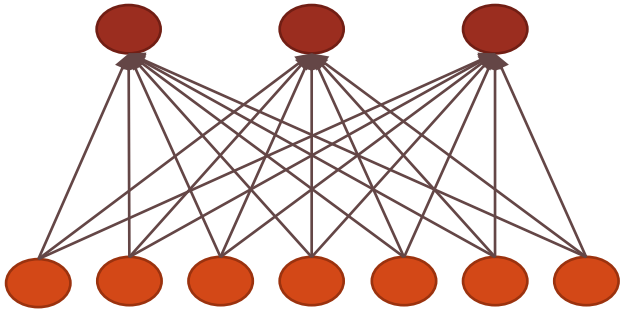


Global connectivity

Local connectivity

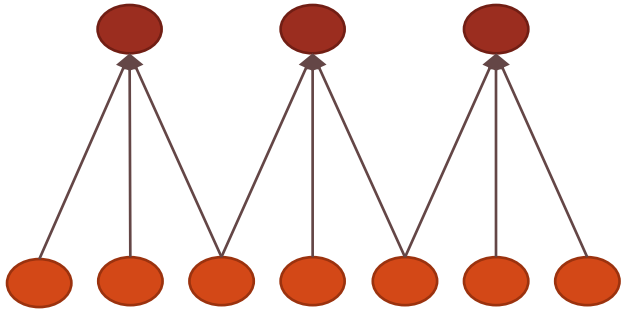
- # input units (neurons): 7
- # hidden units: 3
- **Number of parameters**
  - Global connectivity: ?
  - Local connectivity: ?

CNN: Local Connectivity



Hidden layer

Input layer

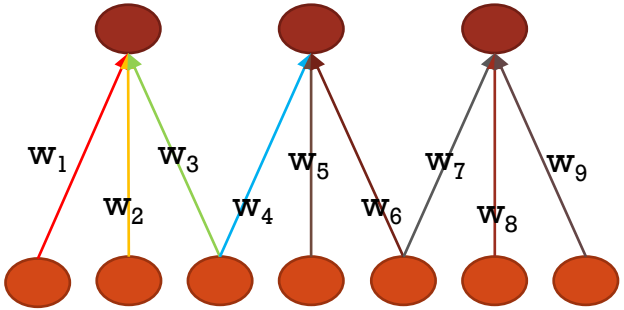


Global connectivity

Local connectivity

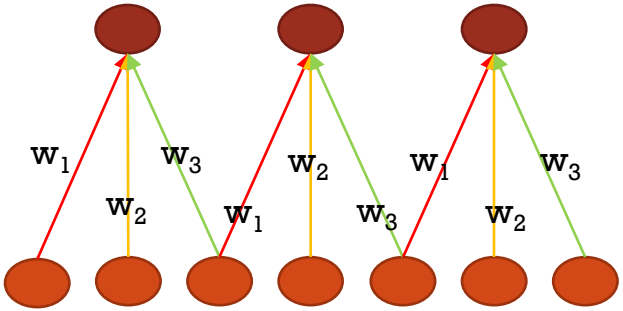
- # input units (neurons): 7
- # hidden units: 3
- **Number of parameters**
  - Global connectivity: **3 x 7 = 21**
  - Local connectivity: **3 x 3 = 9**

CNN: Weight Sharing



Hidden layer

Input layer

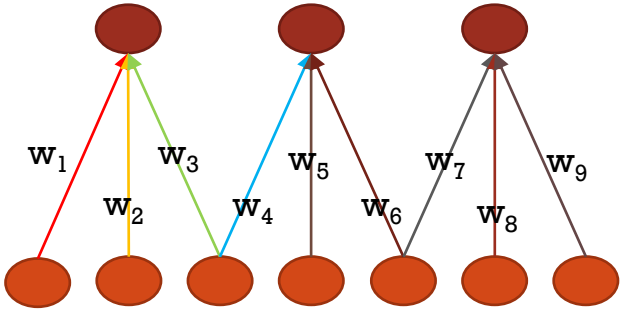


Without weight sharing

With weight sharing

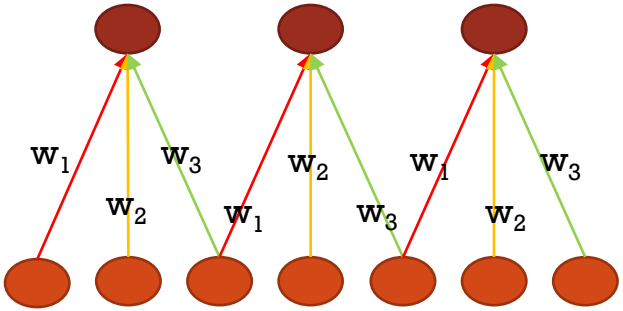
- # input units (neurons): 7
- # hidden units: 3

CNN: Weight Sharing



Hidden layer

Input layer

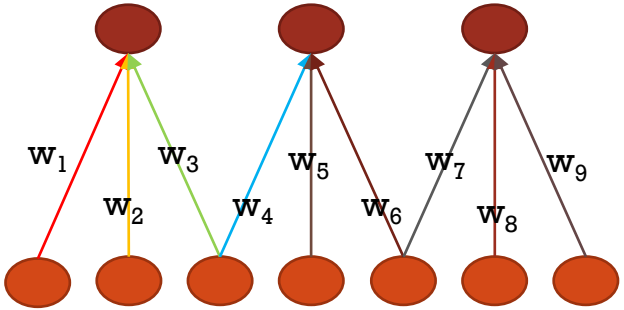


Without weight sharing

With weight sharing

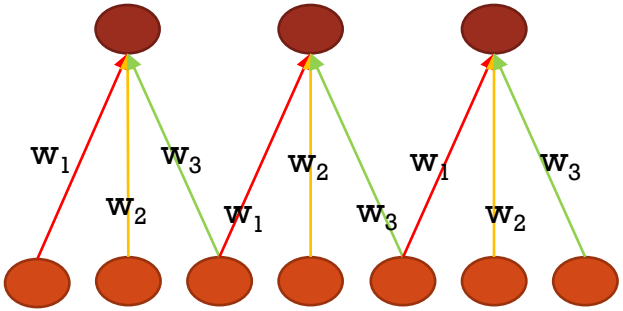
- # input units (neurons): 7
- # hidden units: 3
- Number of parameters
  - Without weight sharing: ?
  - With weight sharing : ?

CNN: Weight Sharing



Hidden layer

Input layer

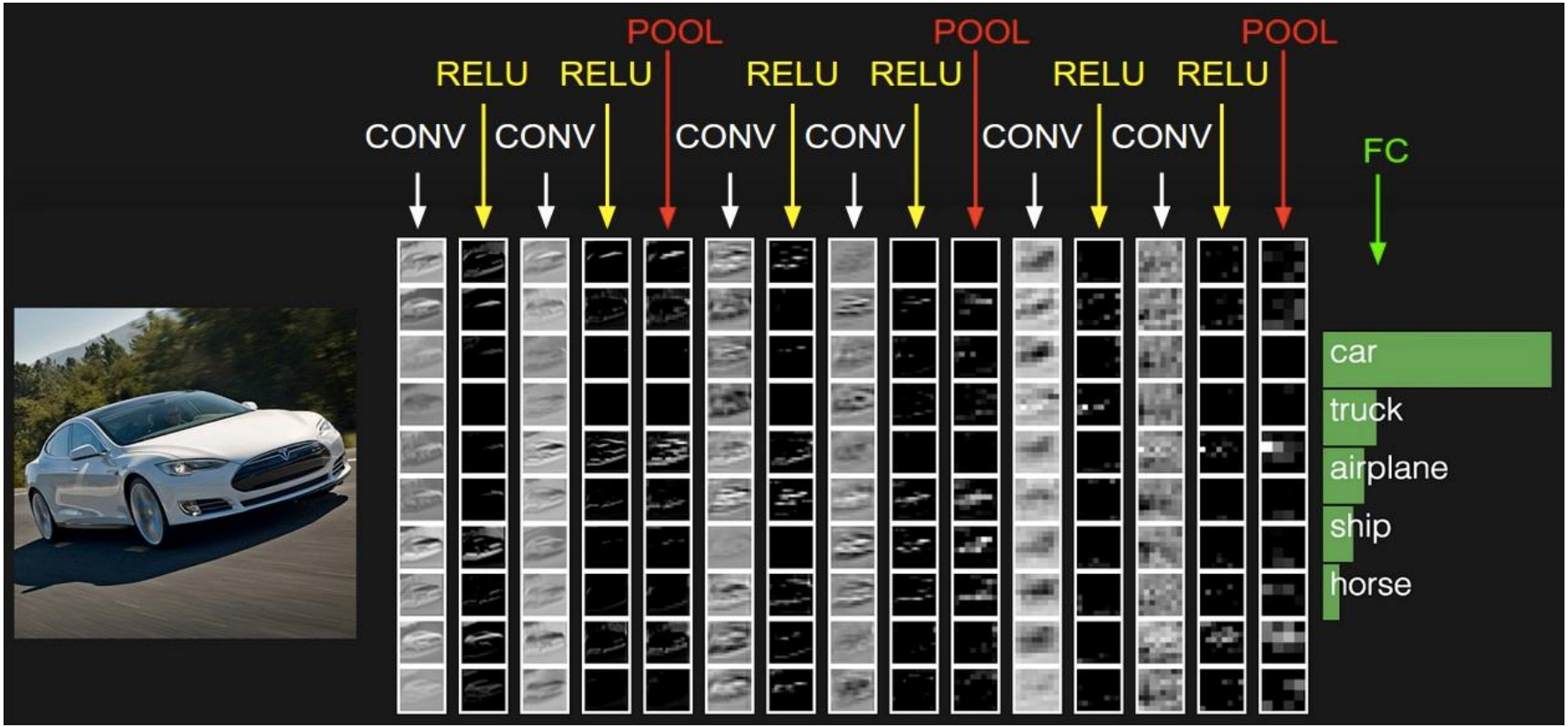


Without weight sharing

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.)

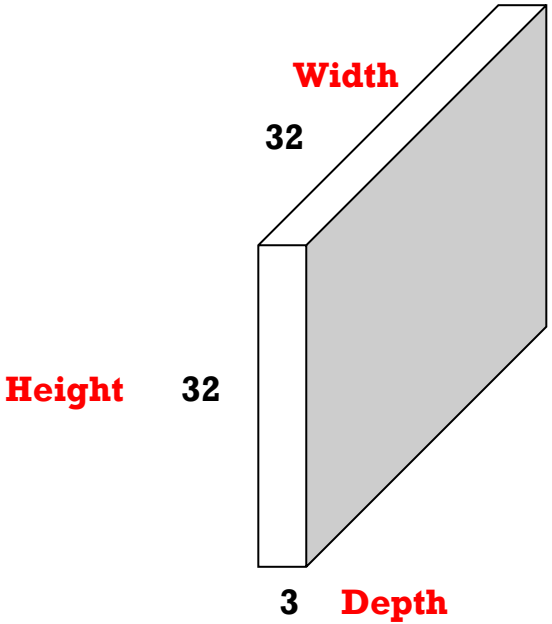
Fully-Connected Layer

Classification Layer (Softmax, etc.)

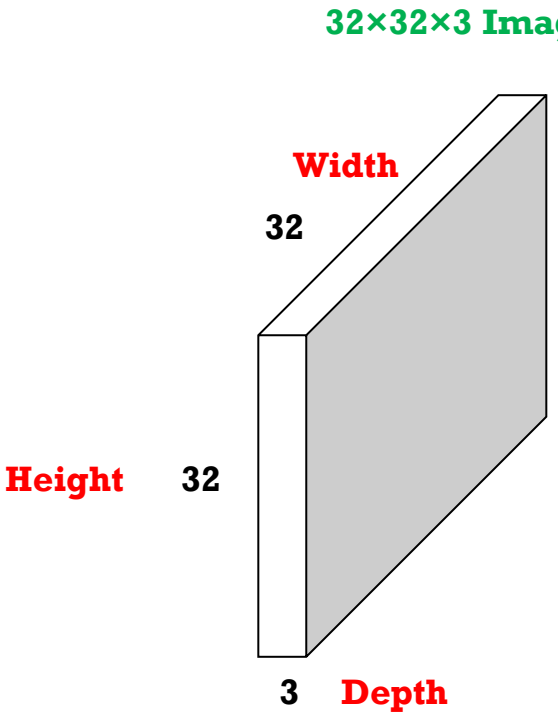
Convolutional Layer

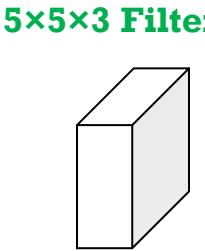
32×32×3 Image

-> preserve spatial structure



Convolutional Layer





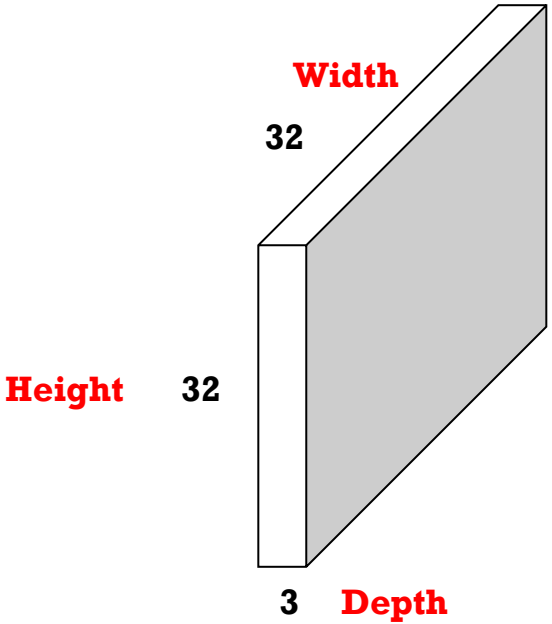
Convolve the filter with the image i.e. "slide over the image spatially, computing dot products"

Convolutional Layer

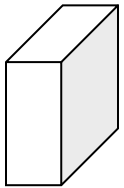
Handling multiple input channels

Filters always extend the full depth of the input volume

32×32×3 Image

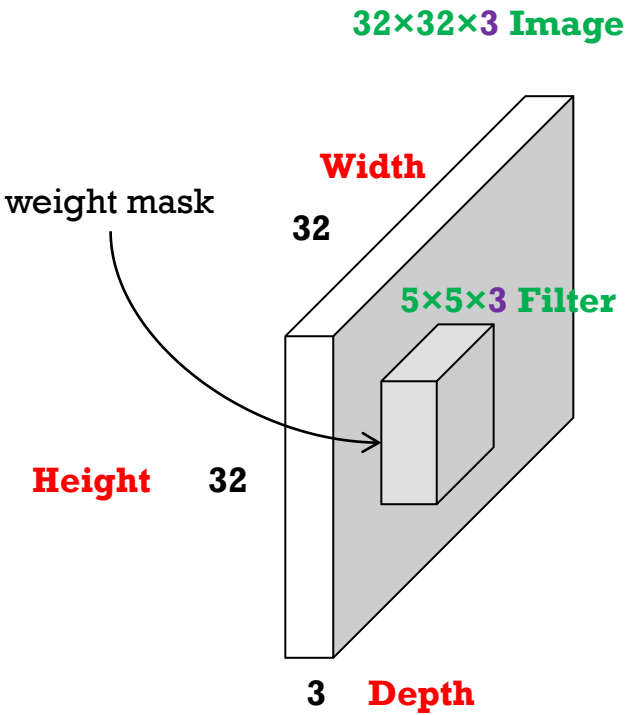


5×5×3 Filter

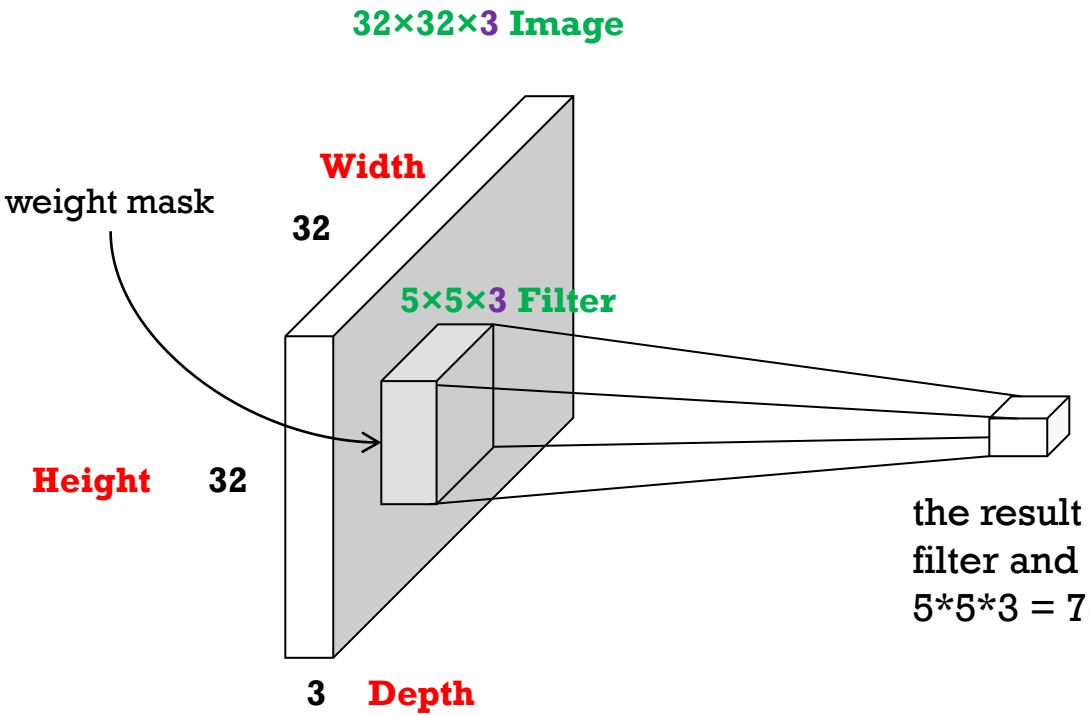


Convolve the filter with the image i.e. "slide over the image spatially, computing dot products"

Convolutional Layer



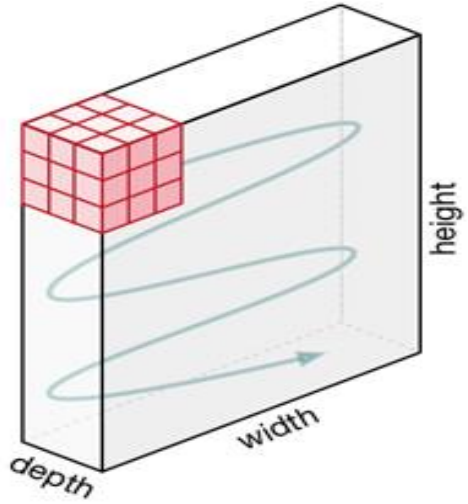
Convolutional Layer

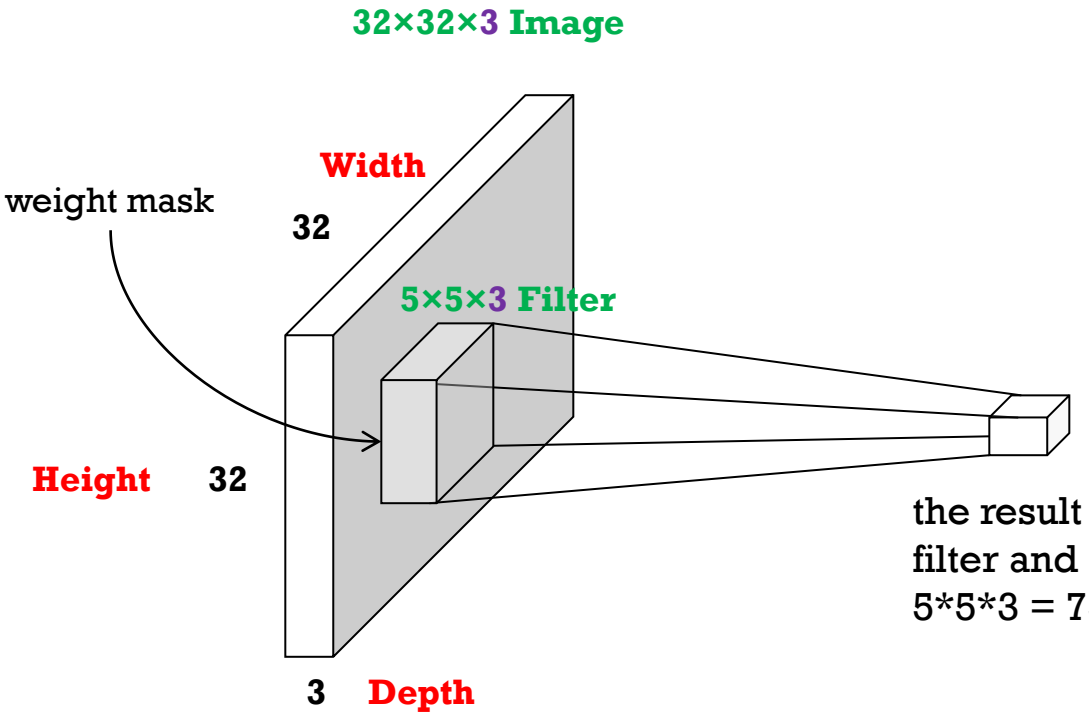


the result of taking a dot product between the filter and a small 5x5x3 chunk of the image (i.e. 5*5*3 = 75-dimensional dot product + bias)

$$w^T.x + b$$

Convolutional Layer

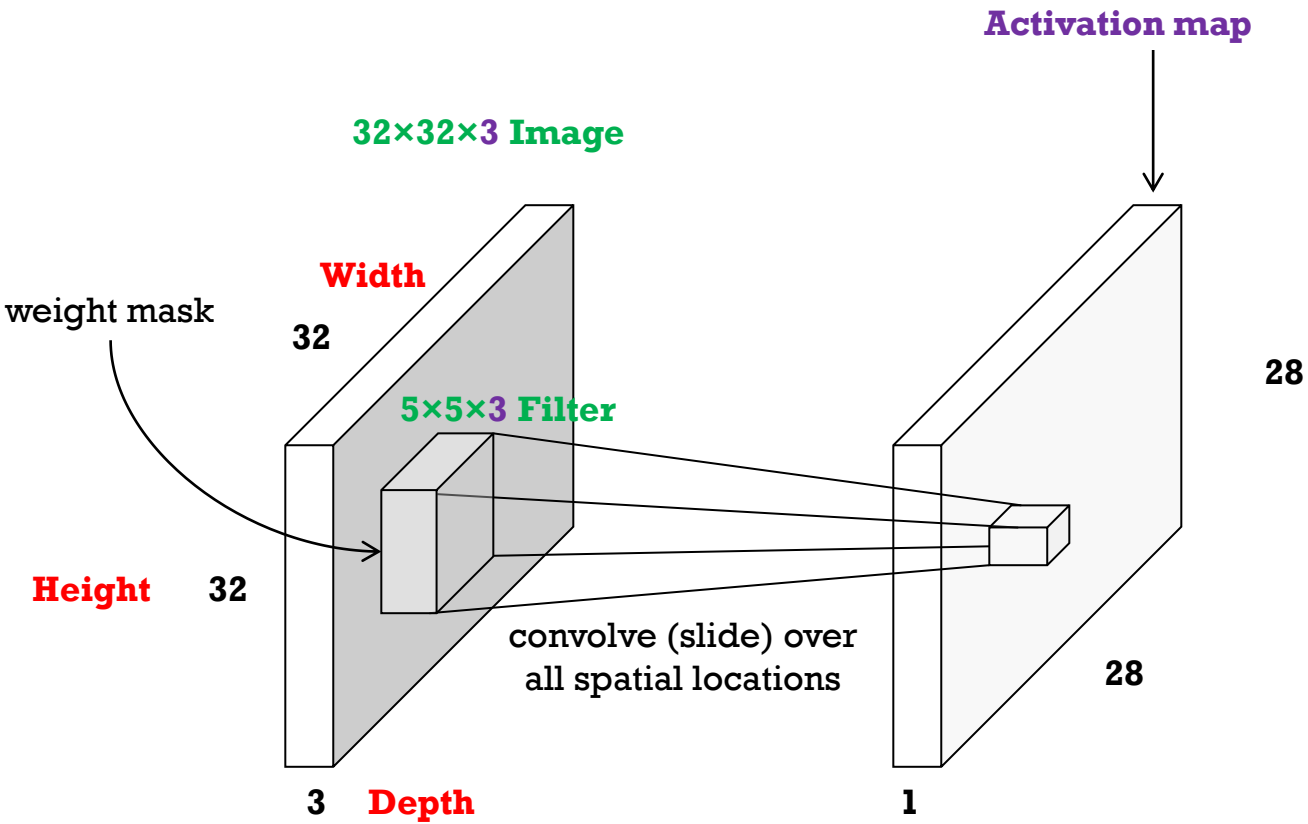


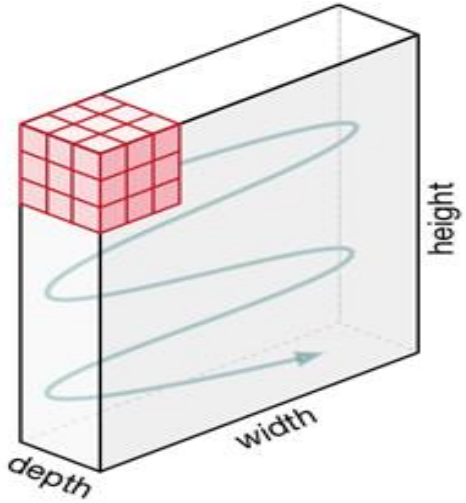


the result of taking a dot product between the filter and a small 5x5x3 chunk of the image (i.e. 5*5*3 = 75-dimensional dot product + bias)

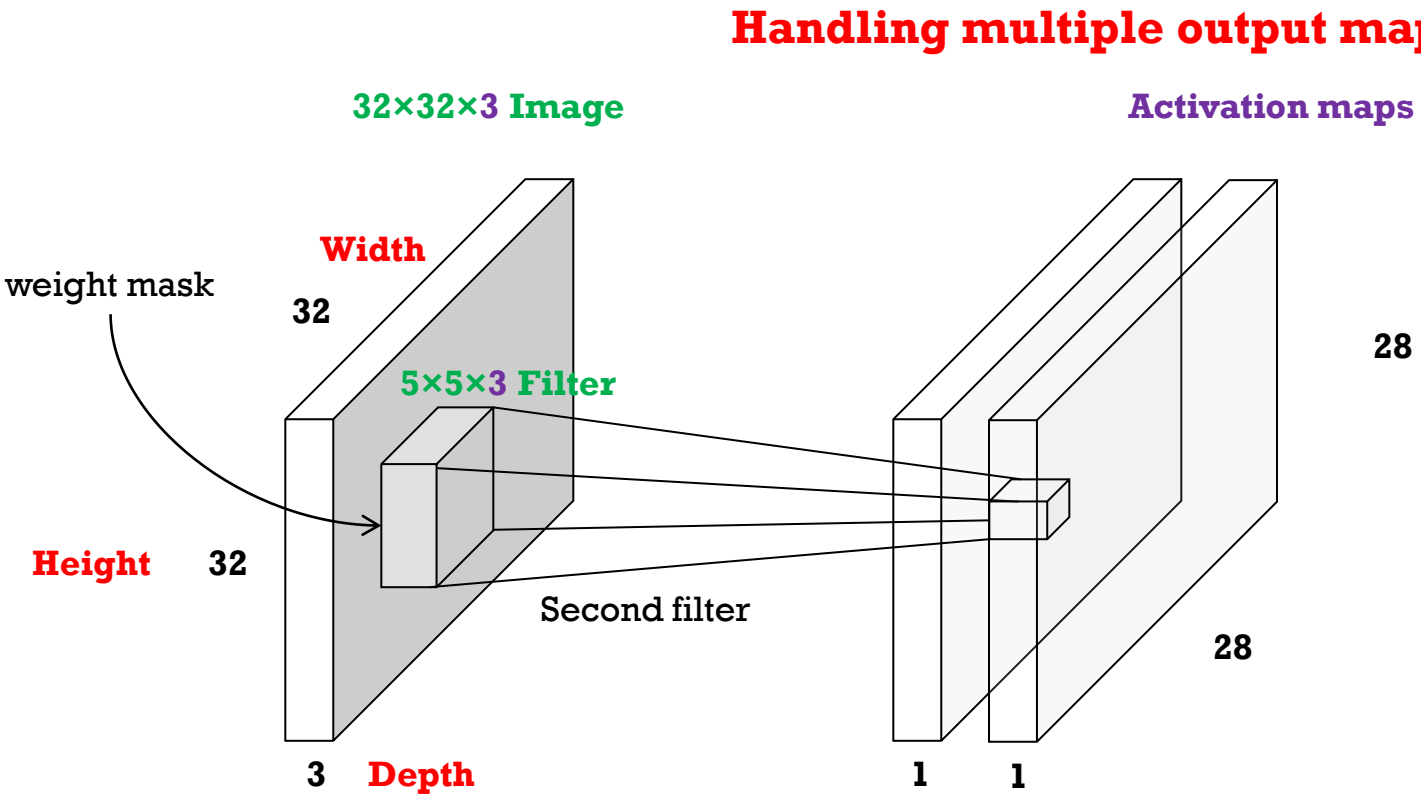
$$w^T.x + b$$

Convolutional Layer

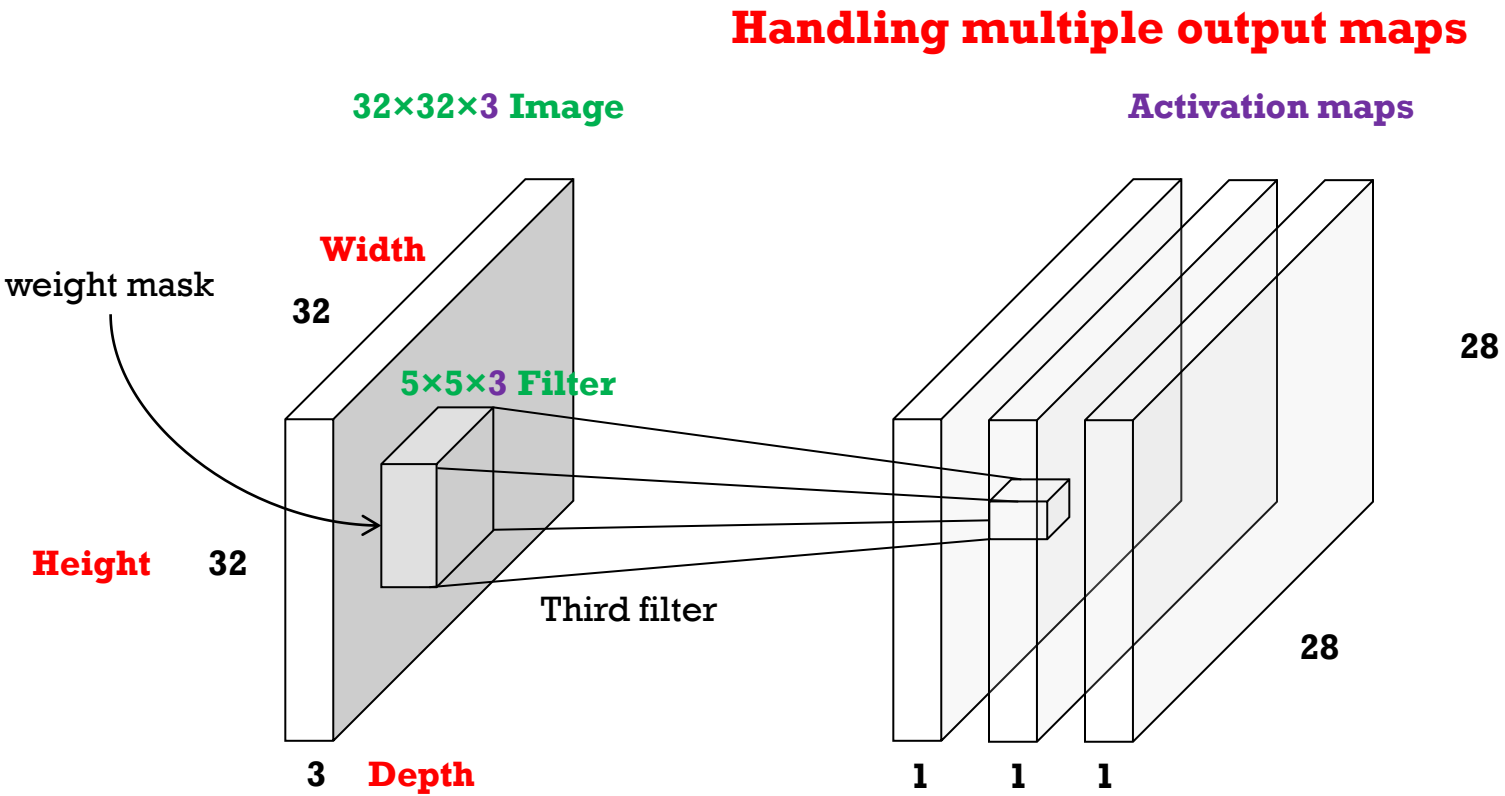




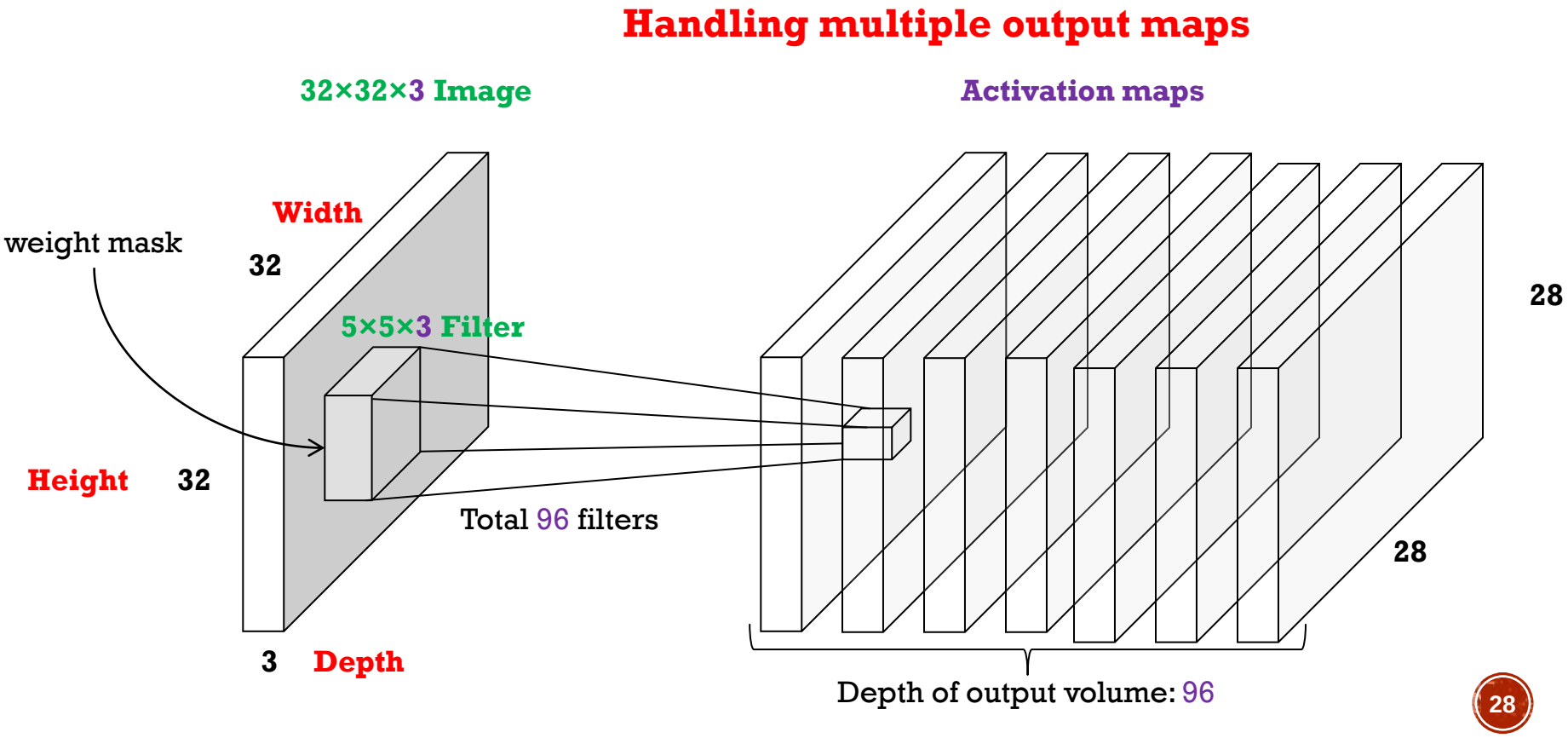
Convolutional Layer



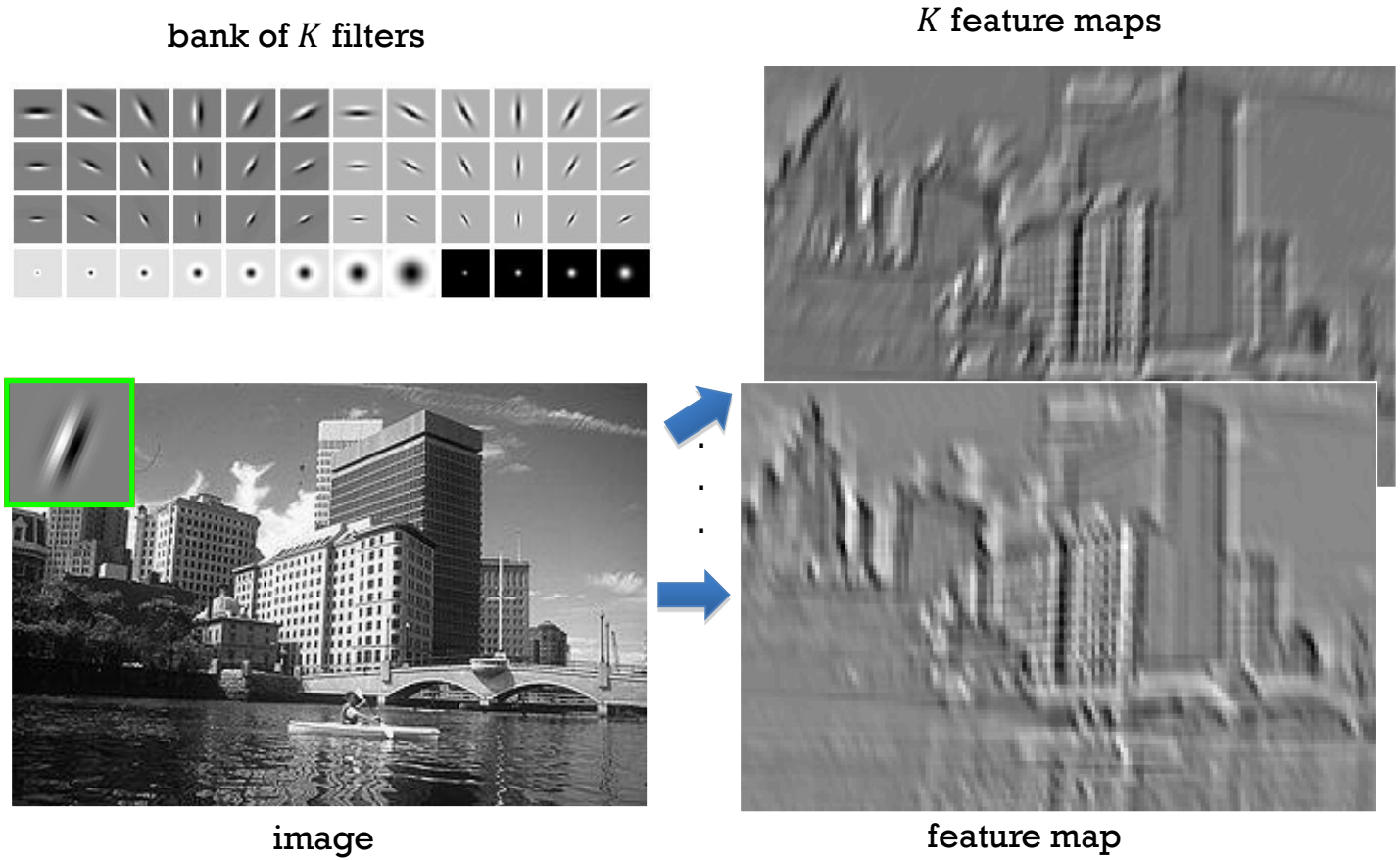
Convolutional Layer



Convolutional Layer



Convolution and Traditional Feature Extraction



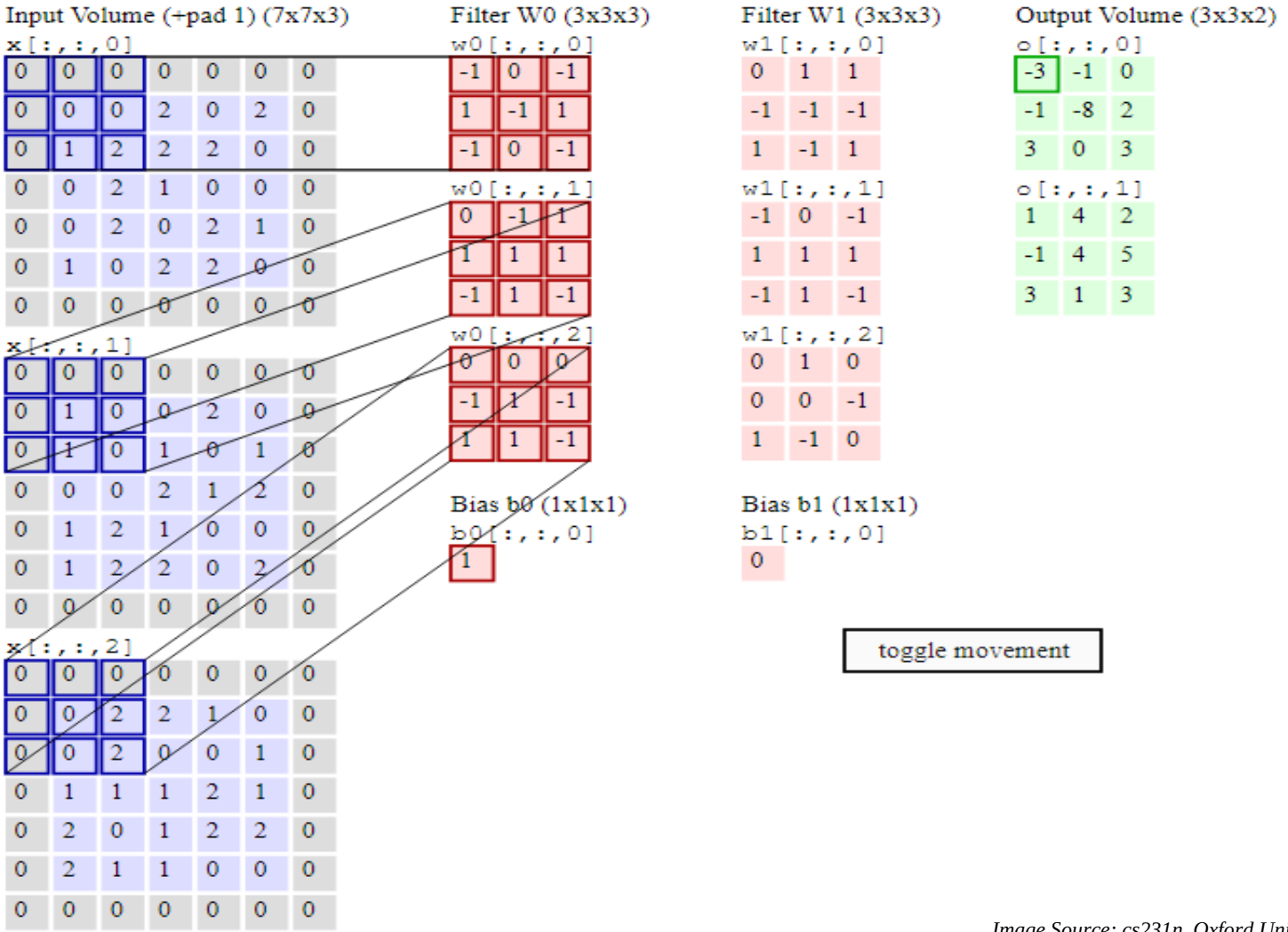


Image Source: cs231n, Oxford University

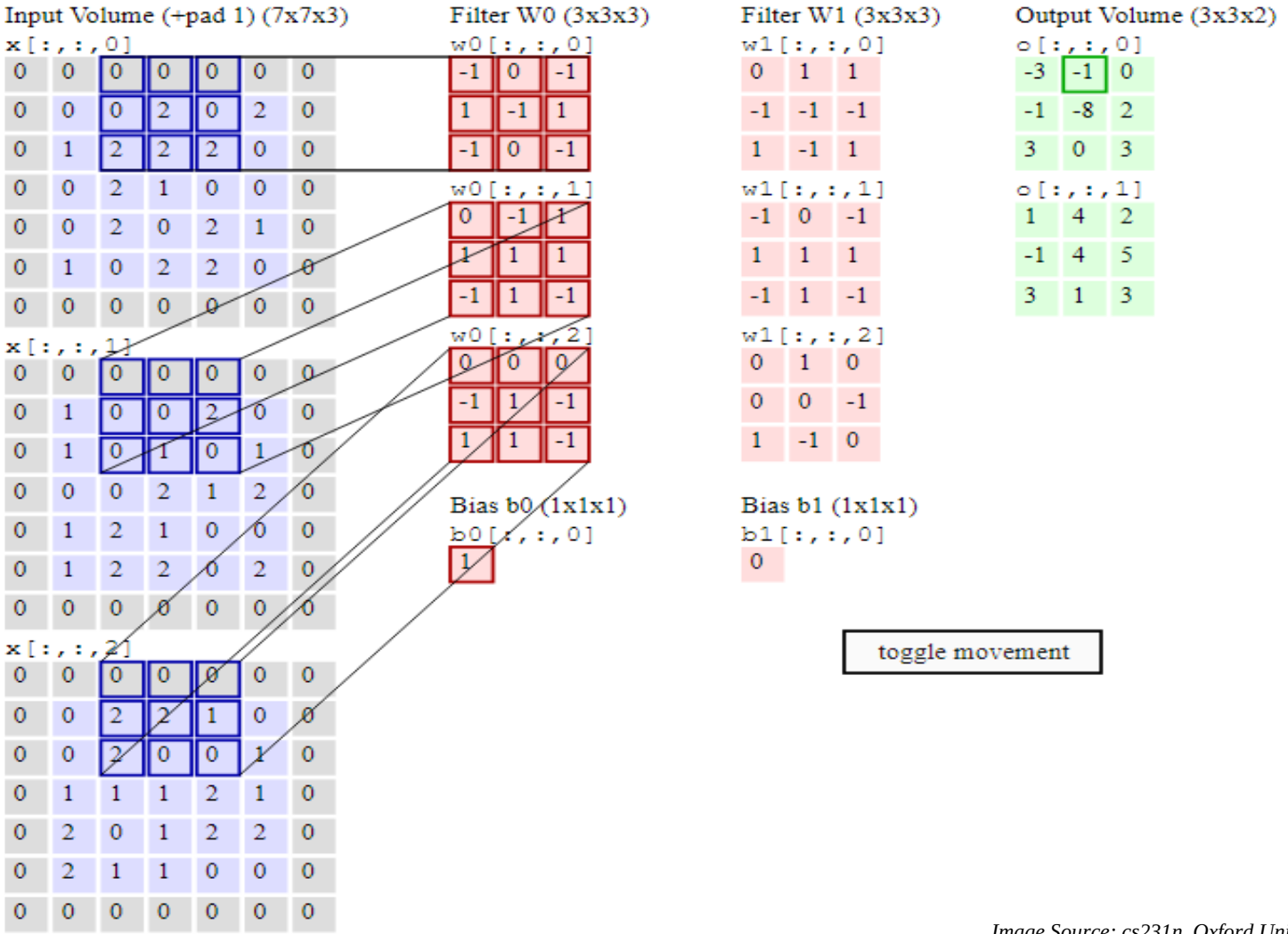


Image Source: cs231n, Oxford University

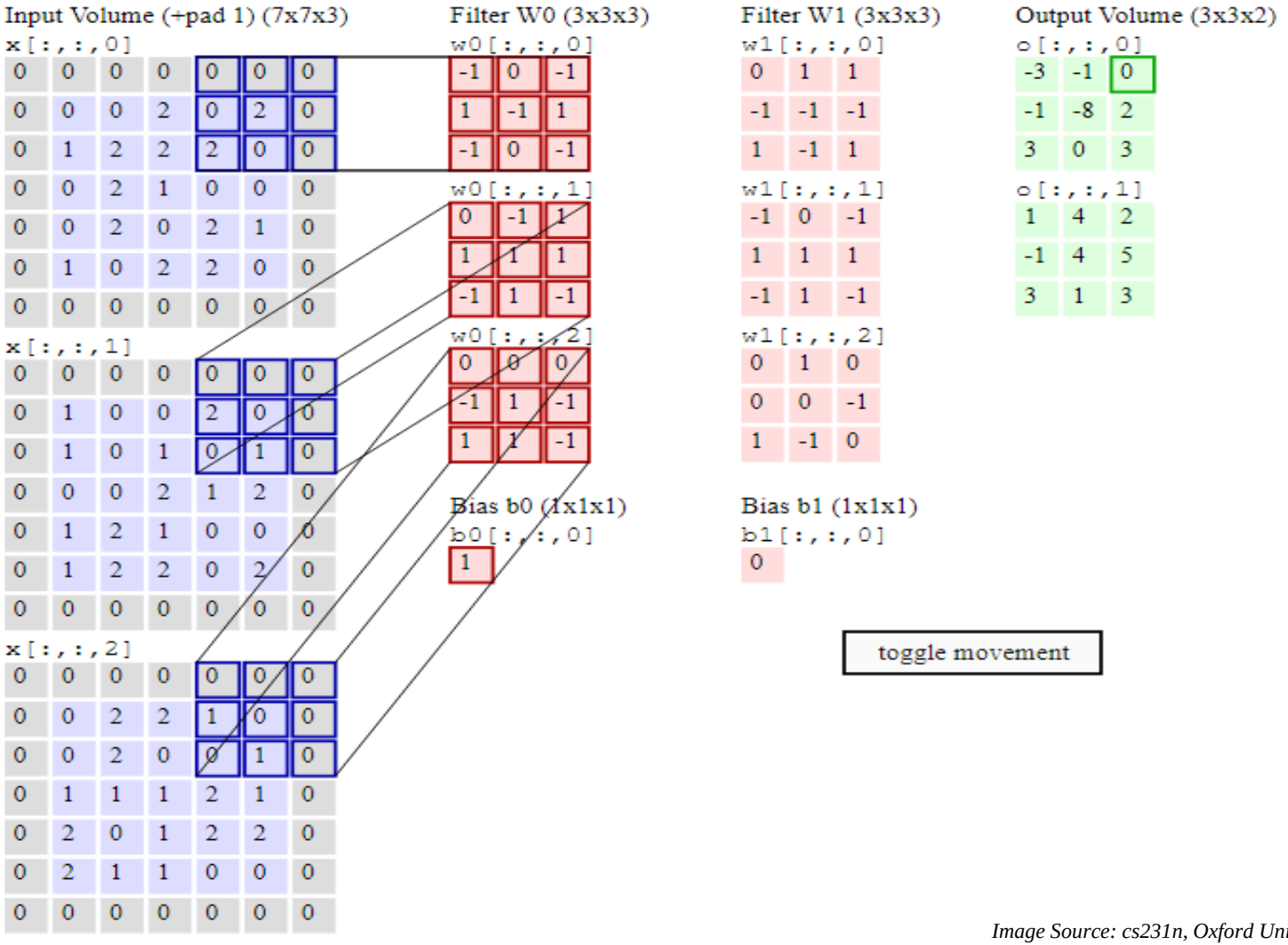
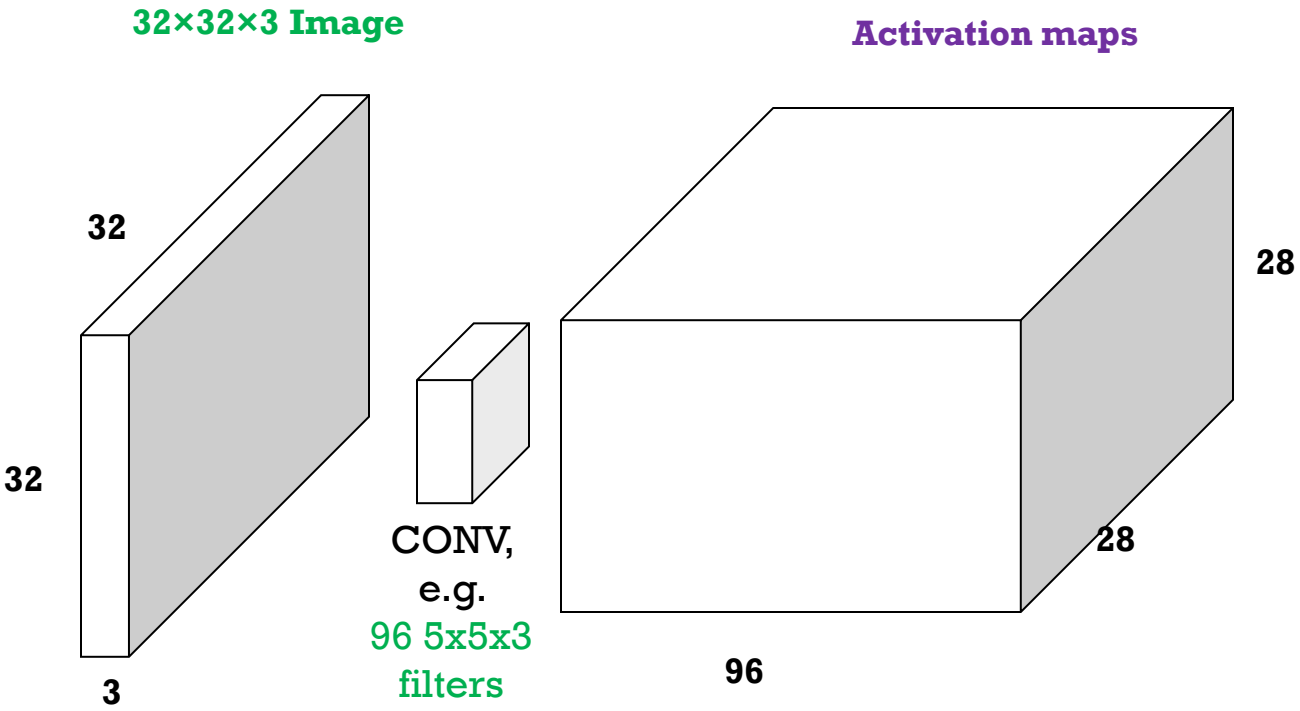


Image Source: cs231n, Oxford University

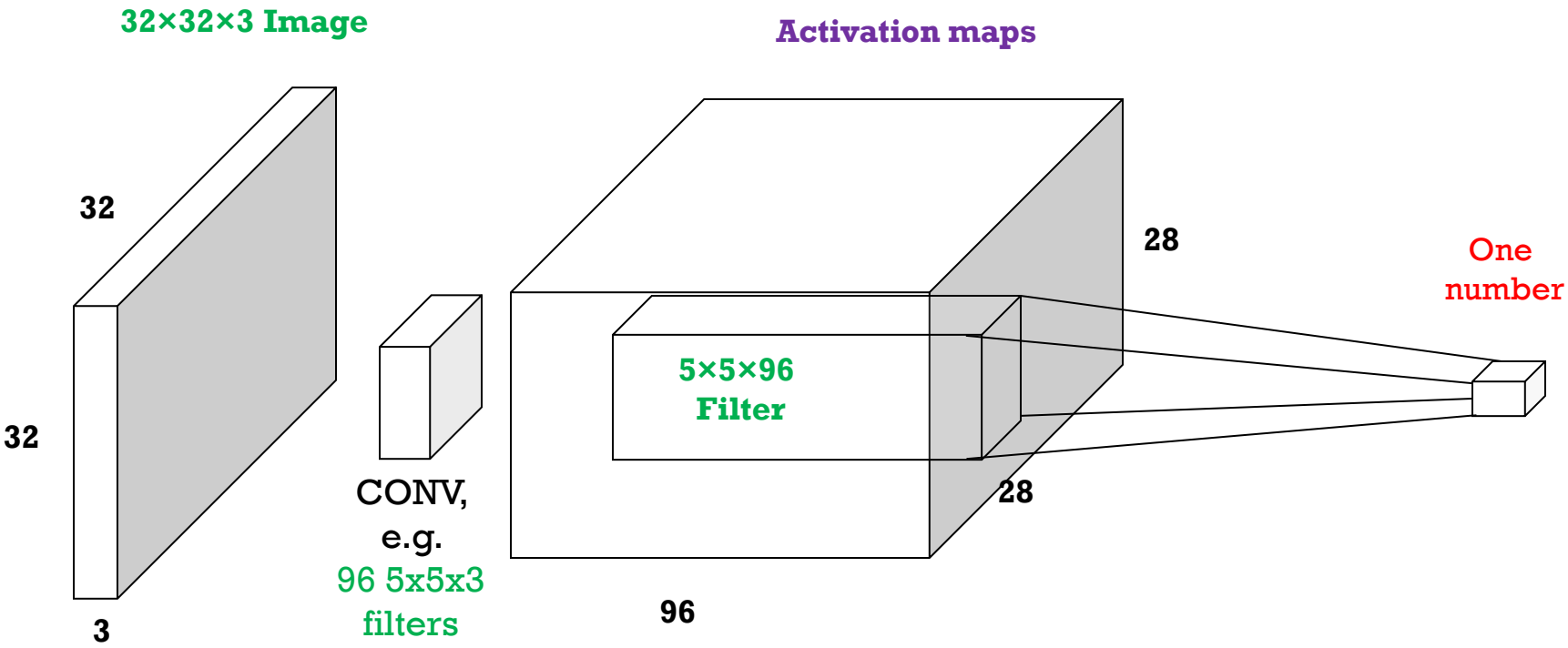
Convolutional Layer

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



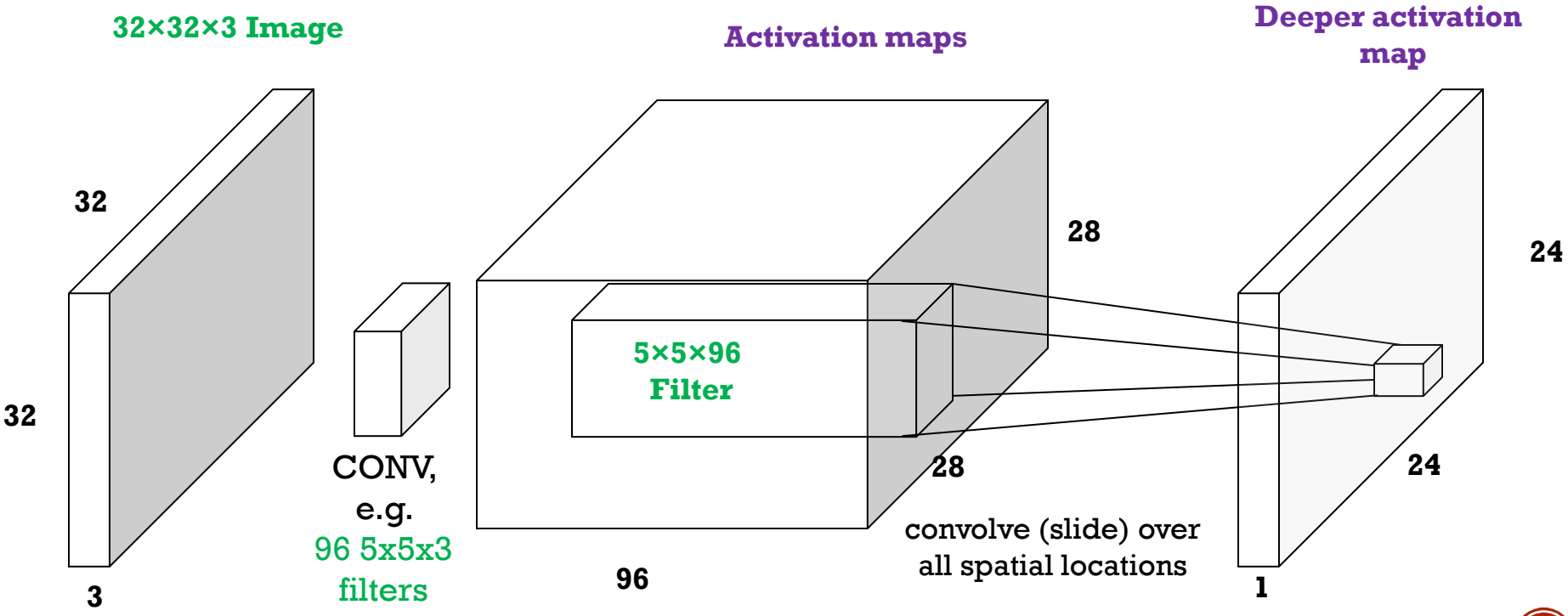
Convolutional Layer

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



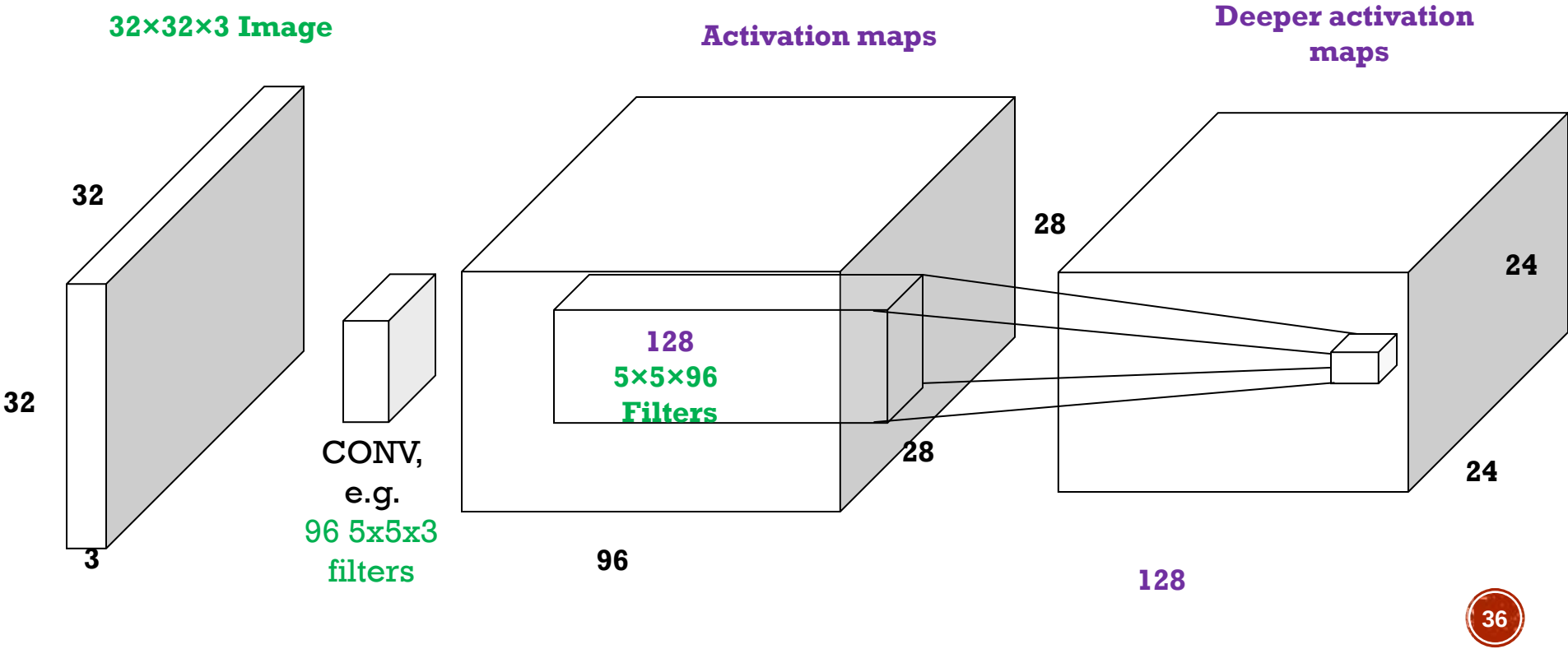
Convolutional Layer

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



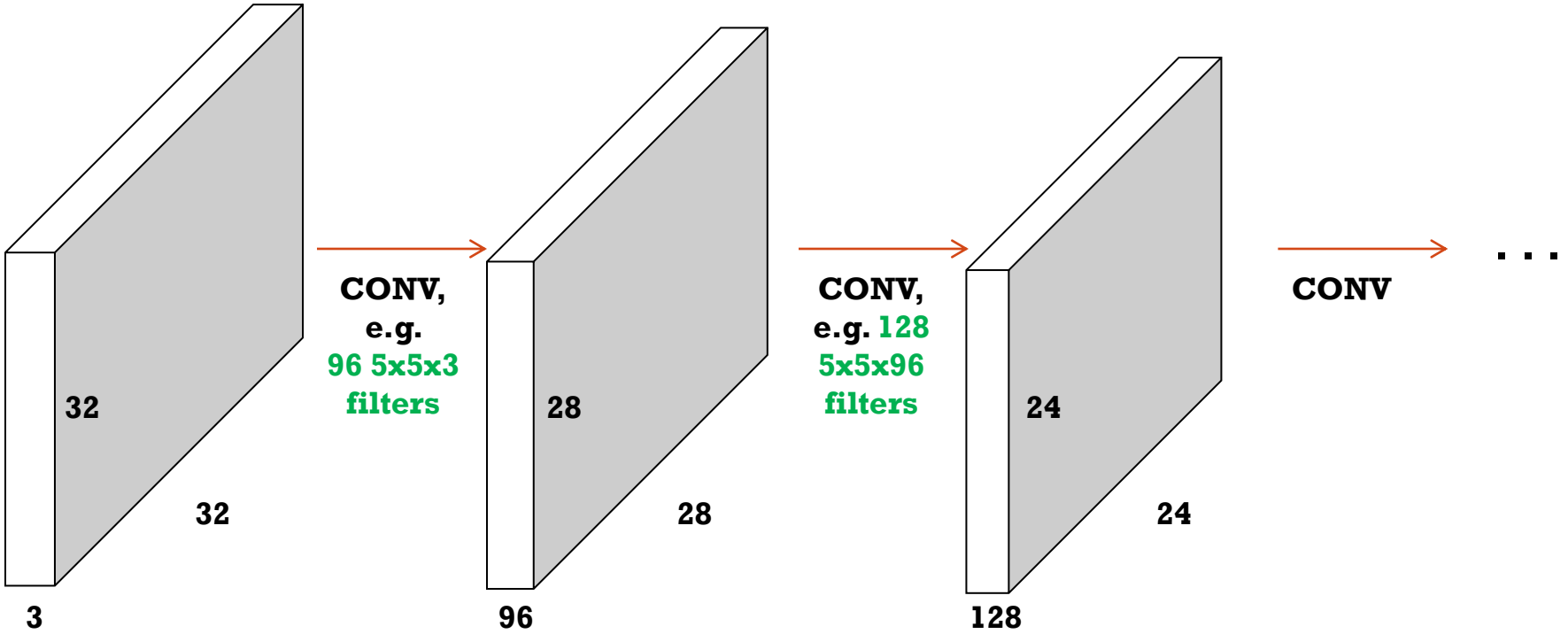
Convolutional Layer

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

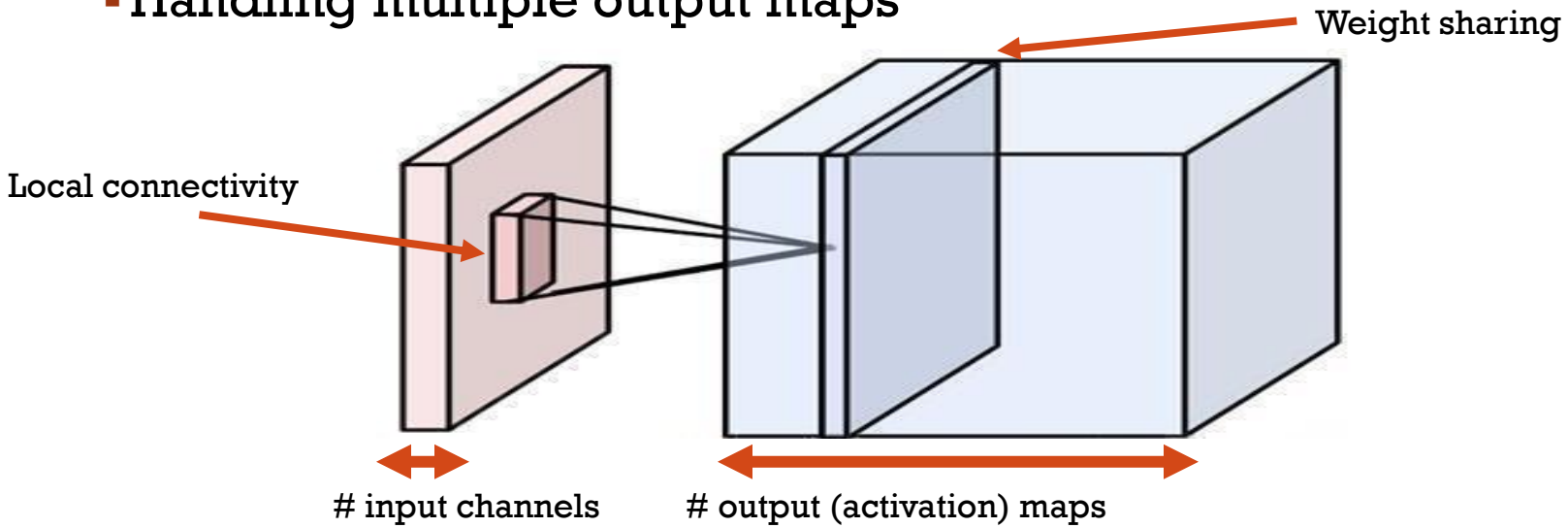
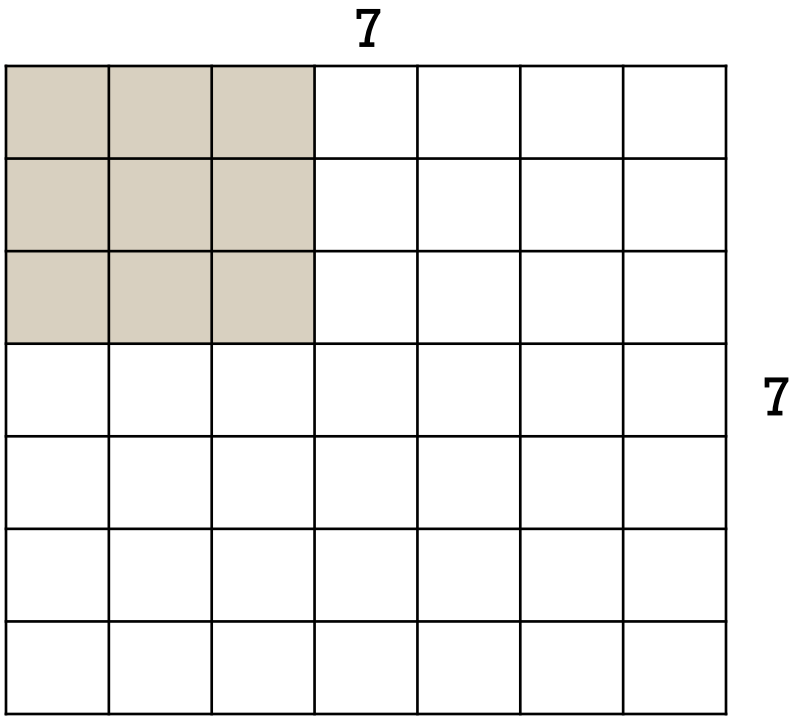


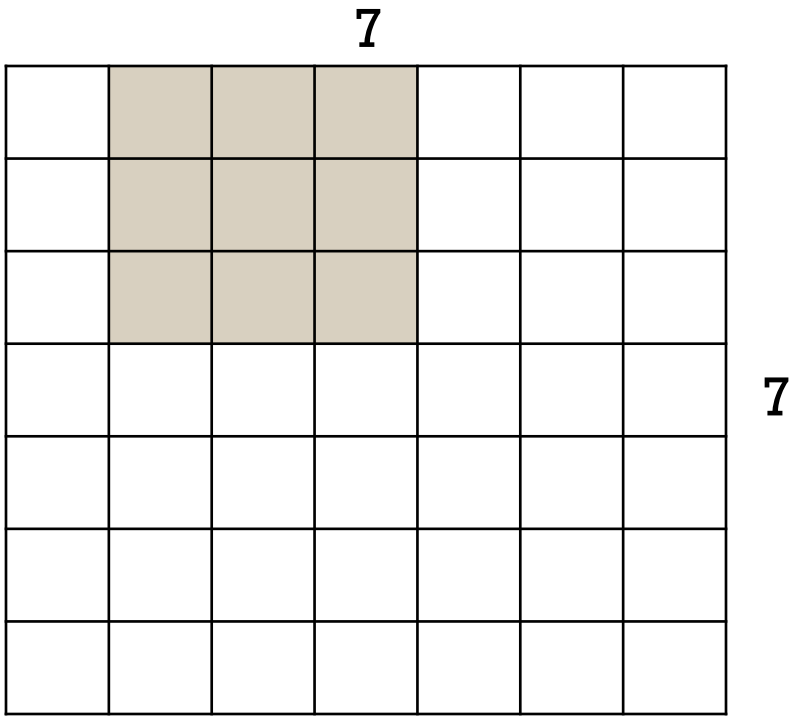
Image credit: A. Karpathy

A closer look at spatial dimensions



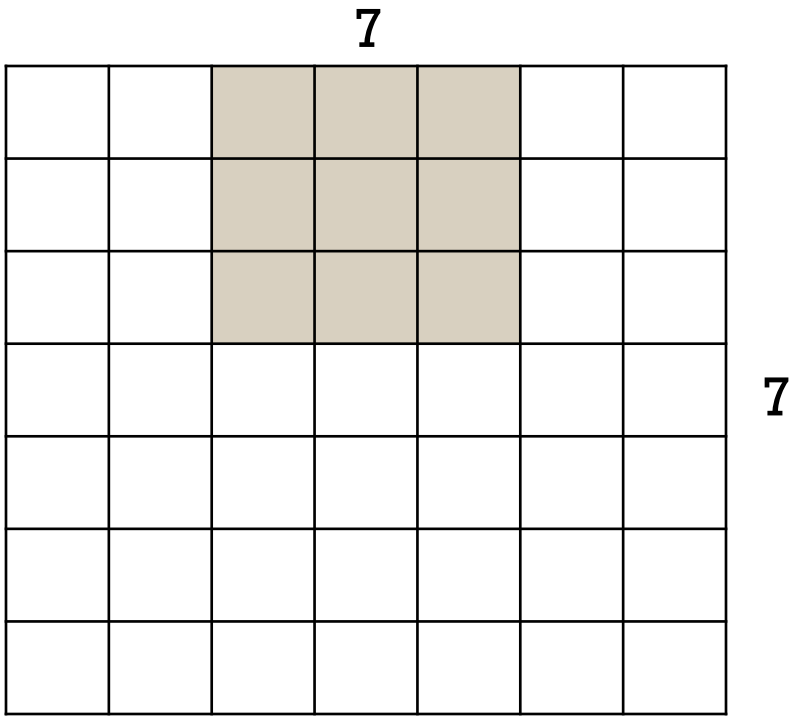
7×7 input (spatially)
assume 3×3 filter

A closer look at spatial dimensions



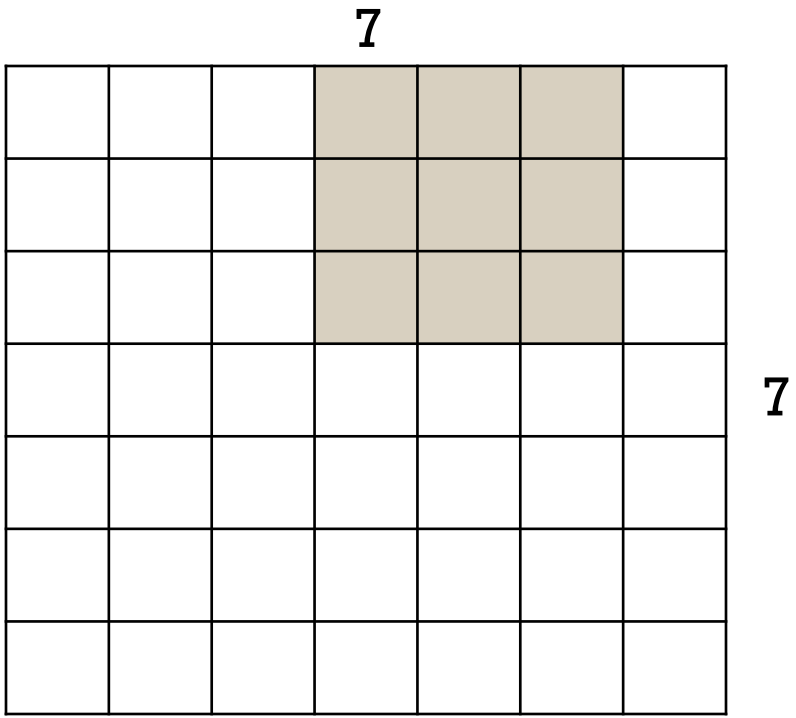
7×7 input (spatially)
assume 3×3 filter

A closer look at spatial dimensions



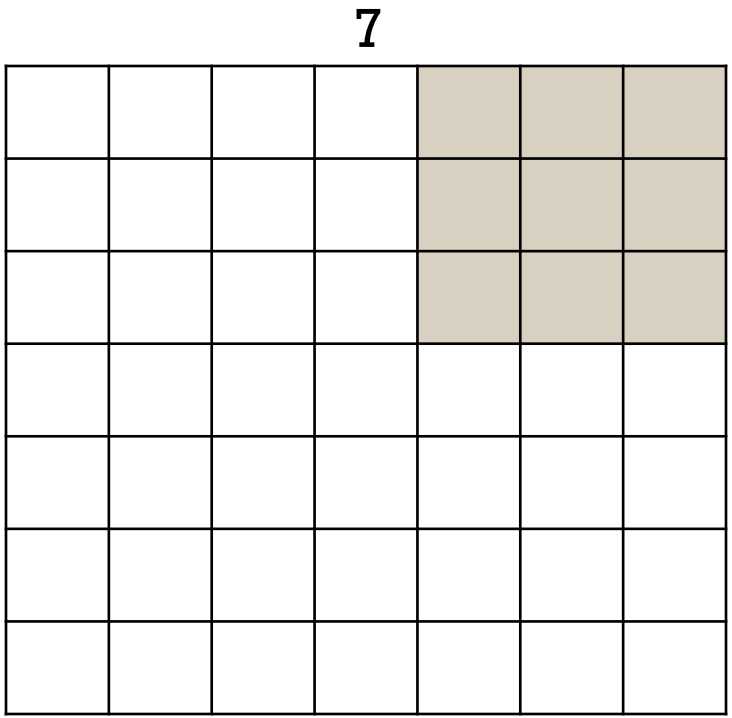
7×7 input (spatially)
assume 3×3 filter

A closer look at spatial dimensions



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

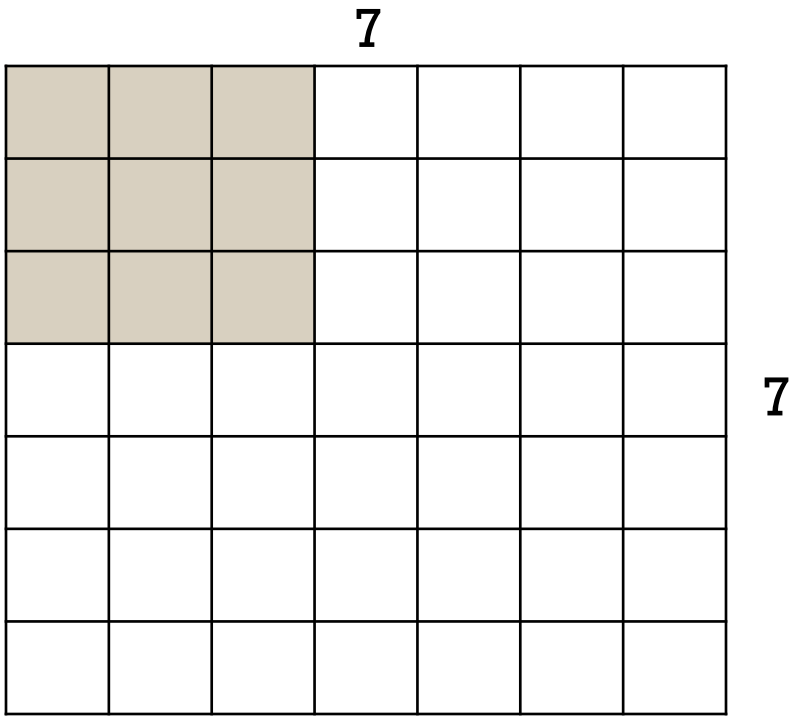
A closer look at spatial dimensions



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

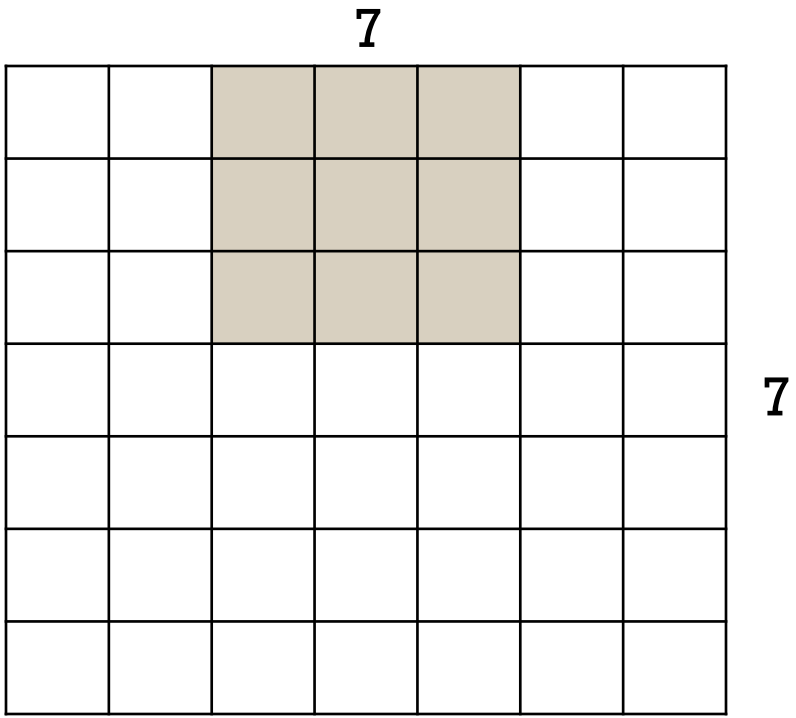
→ 5×5 output

A closer look at spatial dimensions



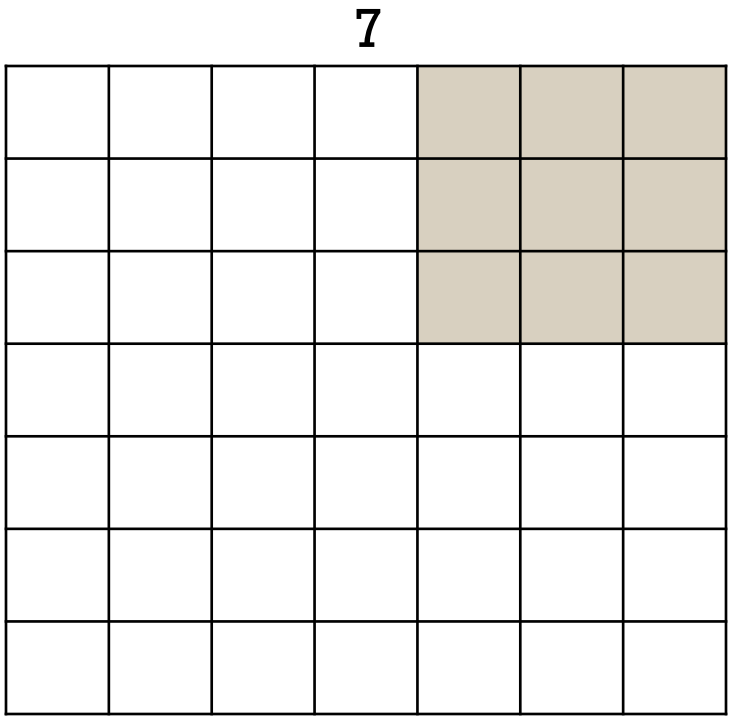
7×7 input (spatially)
assume 3×3 filter
applied with stride 2

A closer look at spatial dimensions



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

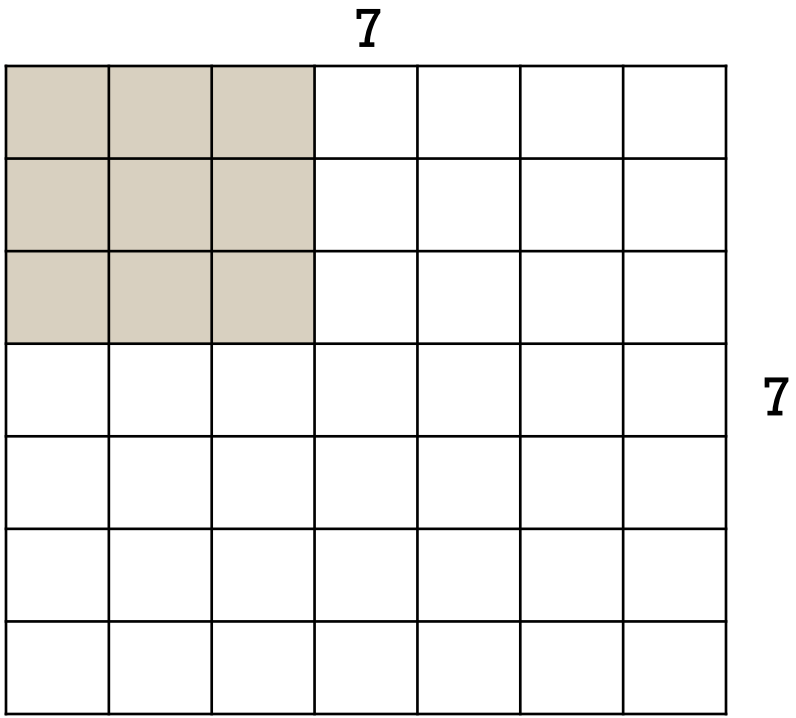
A closer look at spatial dimensions



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

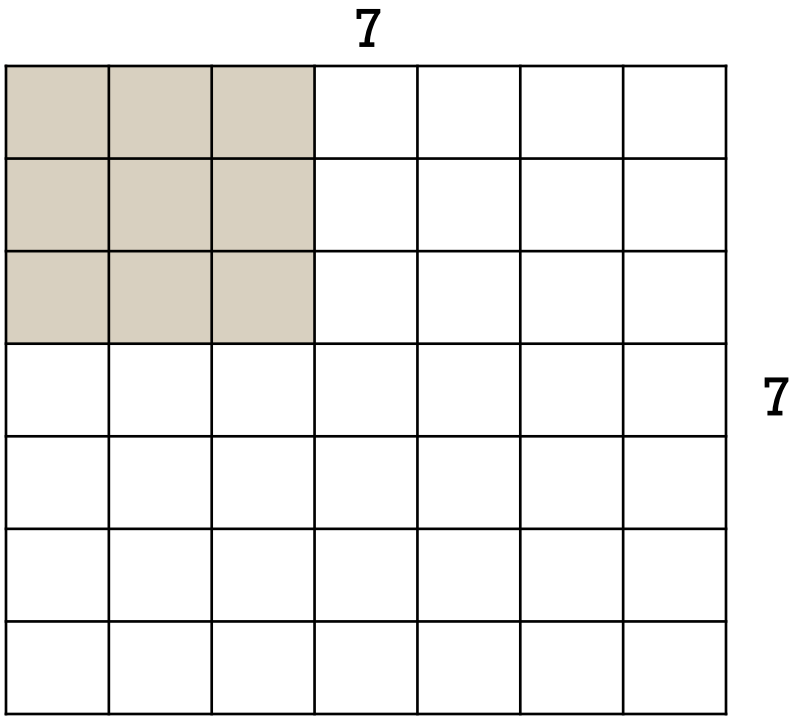
3×3 output

A closer look at spatial dimensions



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

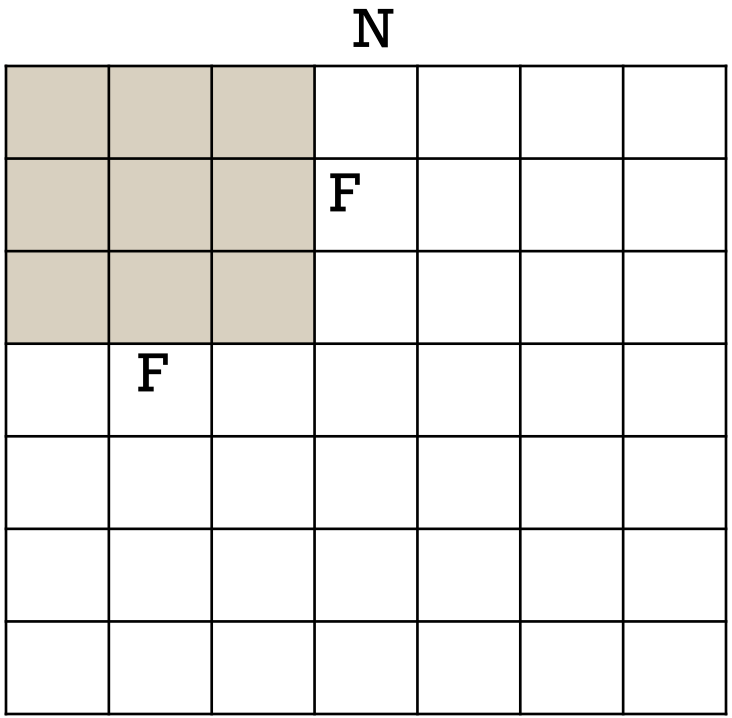
A closer look at spatial dimensions



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.

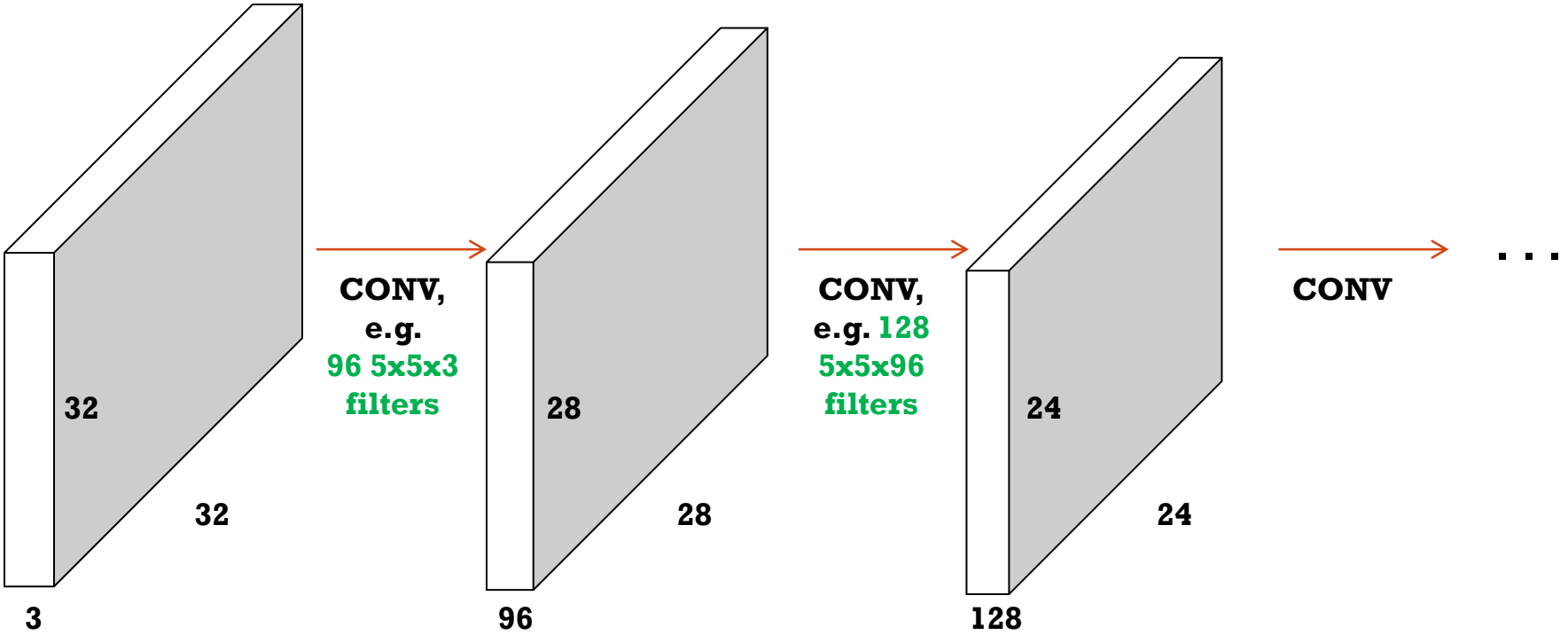
A closer look at spatial dimensions



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

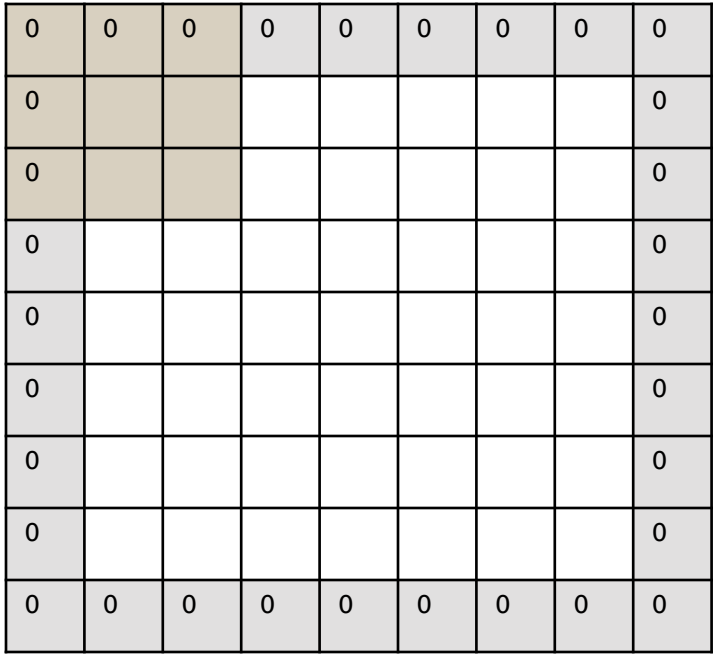
A closer look at spatial dimensions



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

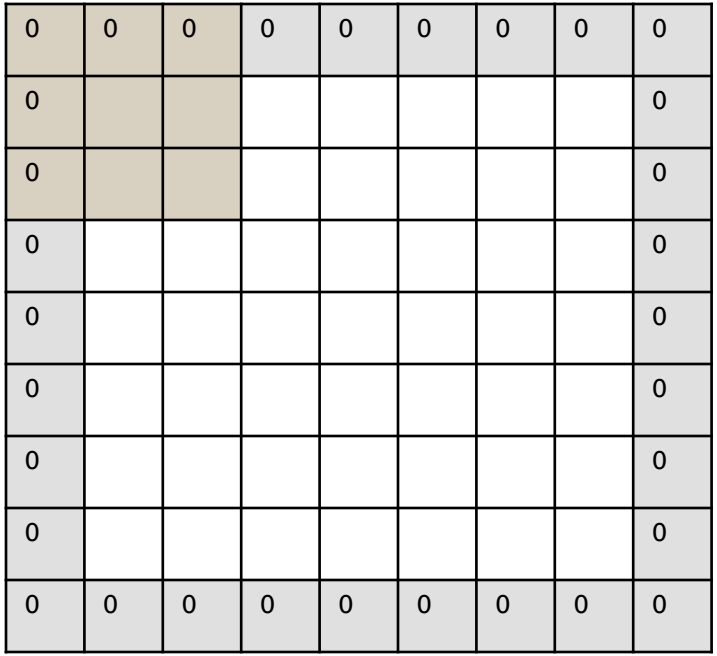
In practice: common to zero pad



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

What is the output dimension?

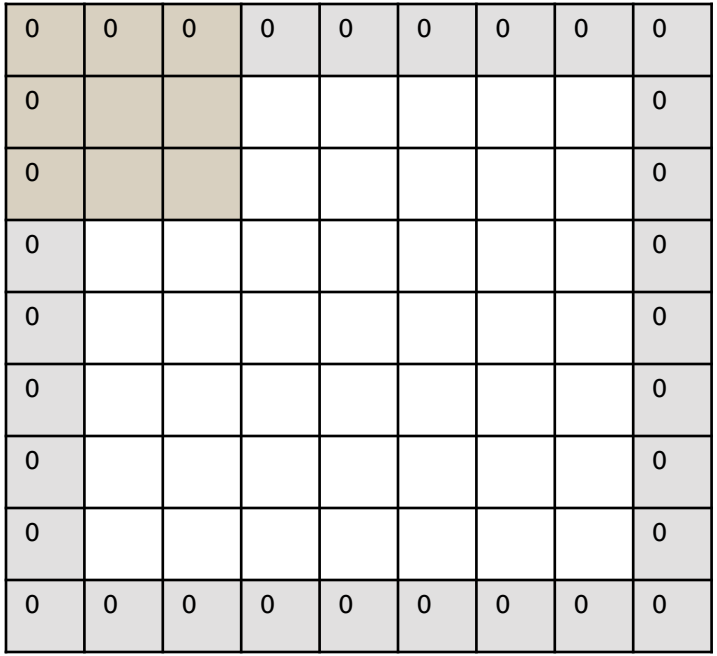
In practice: common to zero pad



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

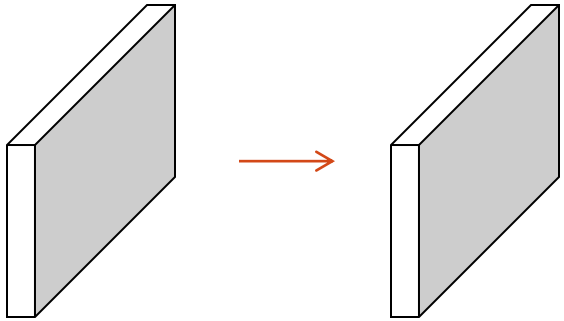
7×7 Output

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

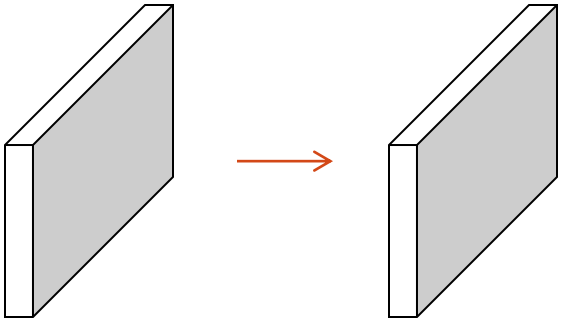
Input volume: 32x32x3
10 5x5 filters with stride 1, pad 2

Output volume size: ?



Example

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

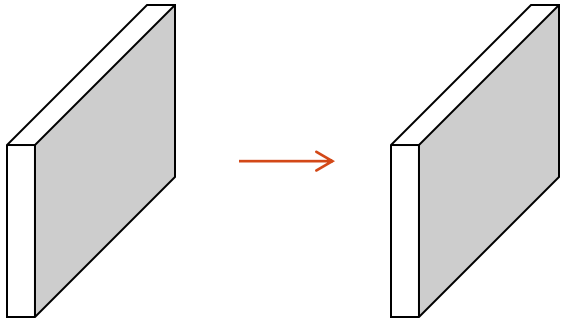


Output volume size:
(32+2*2-5)/1+1 = 32 spatially,
so
32x32x10

Example

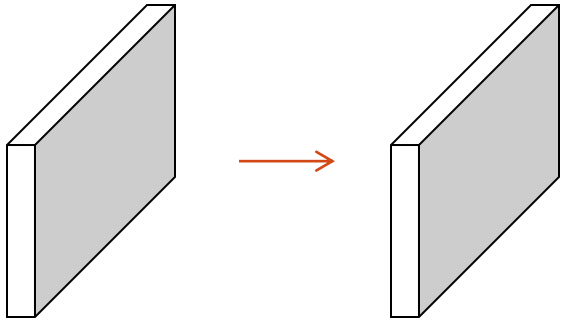
Input volume: 32x32x3
10 5x5 filters with stride 1, pad 2

Number of parameters in this layer?



Example

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



Number of parameters in this layer?

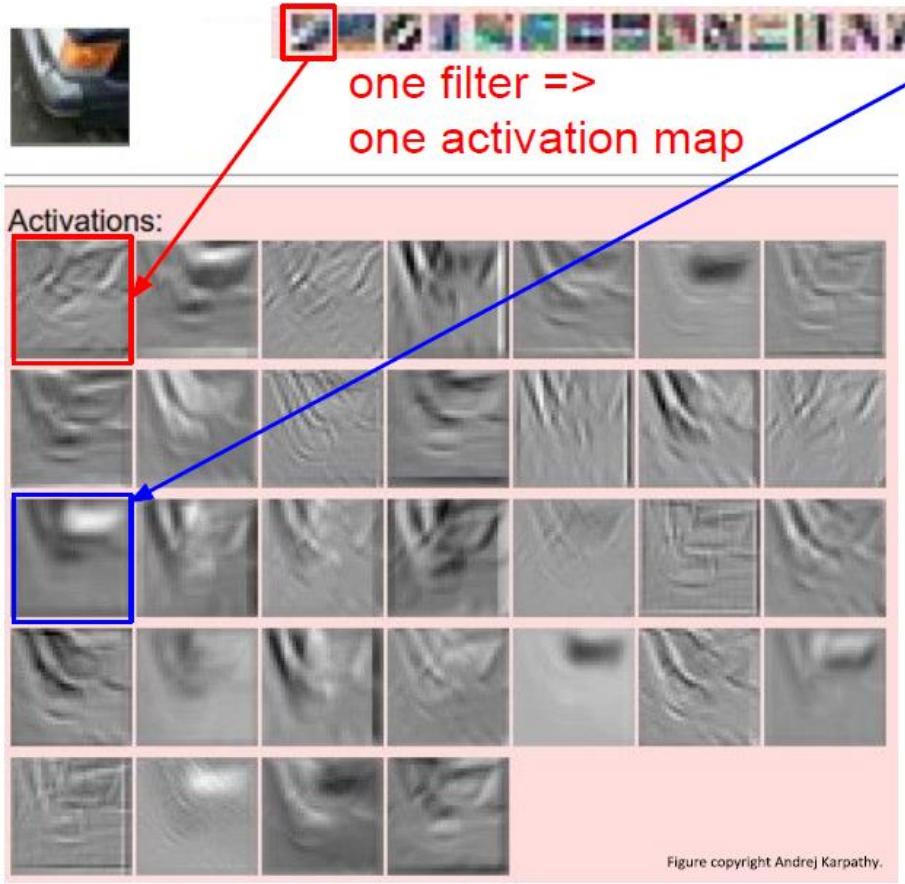
each filter has
5*5*3 + 1 = 76 params (+1 for bias)
=> 76*10 = 760

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:

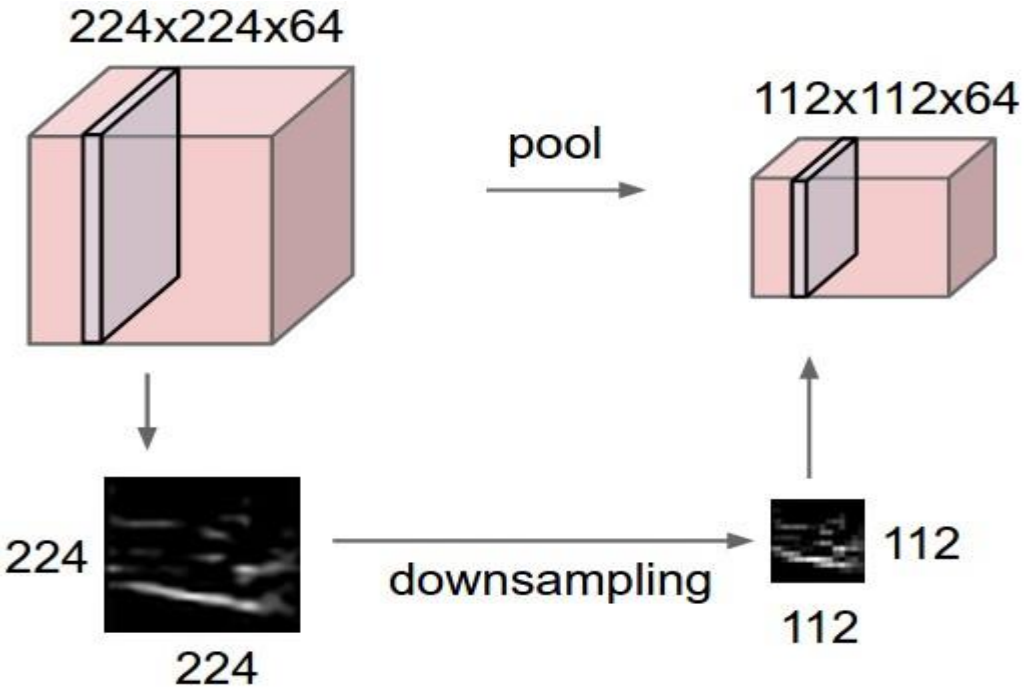
$$f[x,y] * g[x,y] = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f[n_1,n_2] \cdot g[x-n_1, y-n_2]$$

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

Source: cs231n, Stanford University

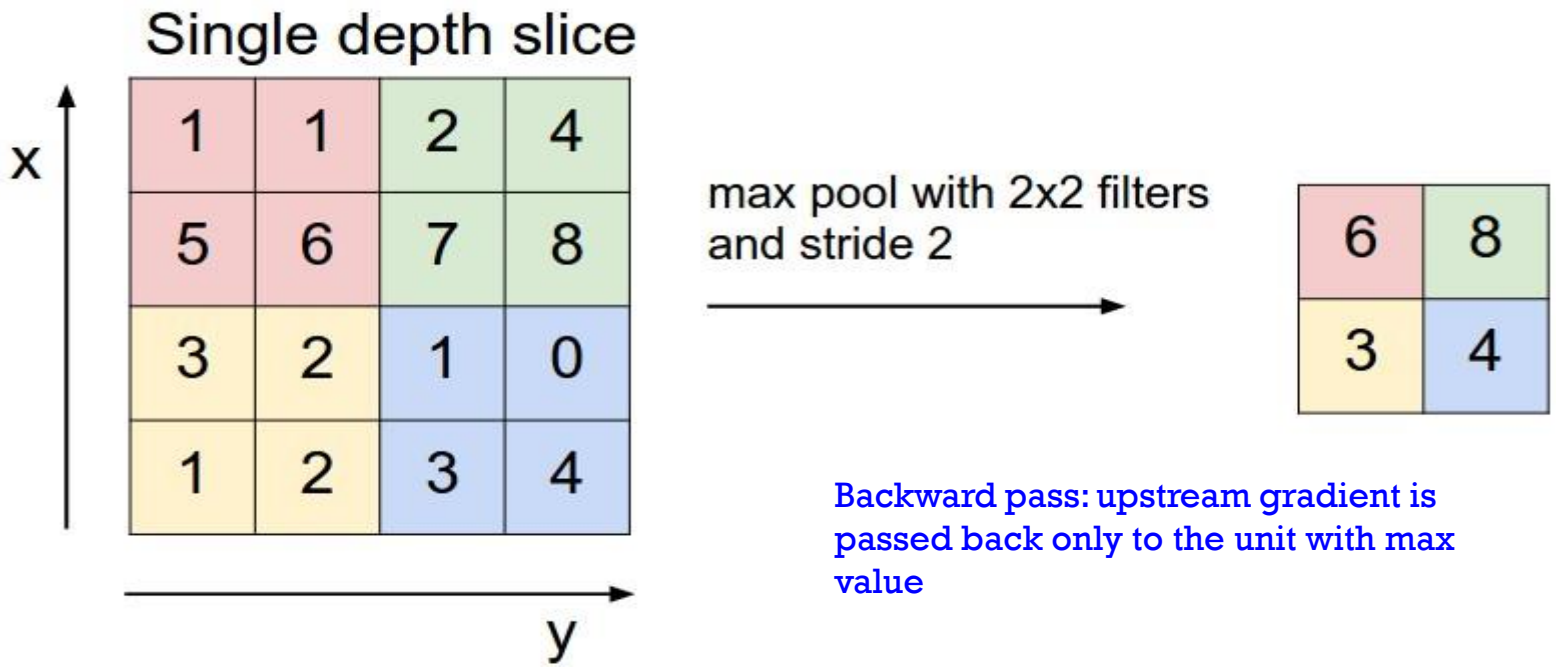
Pooling Layer

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



Source: cs231n, Stanford University

Max Pooling



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

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

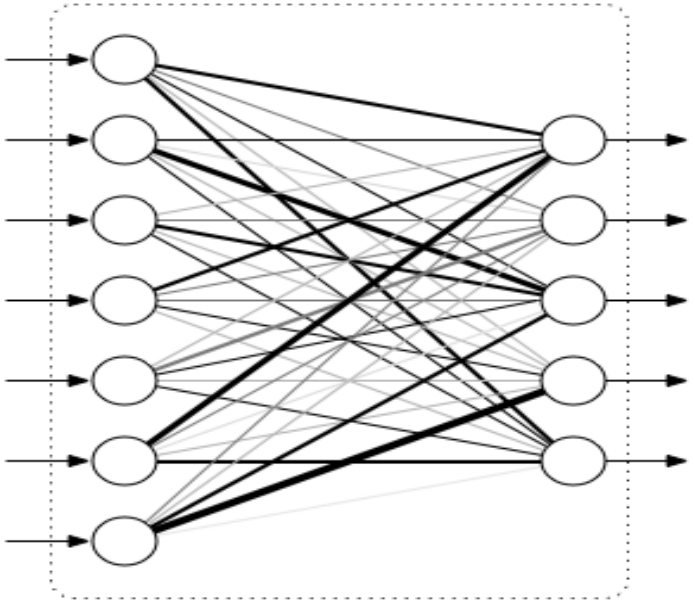


Image Source: machinethink.net

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

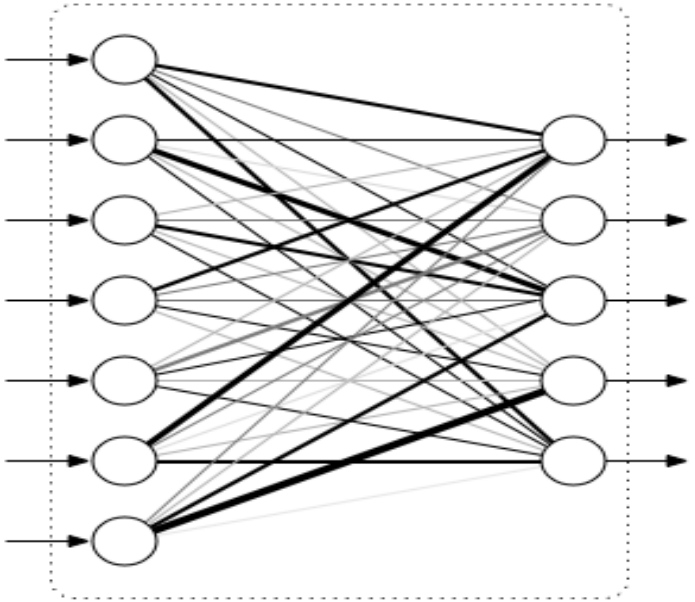


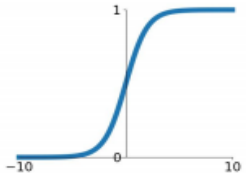
Image Source: machinethink.net

Non-linearity Layer

Activation Functions

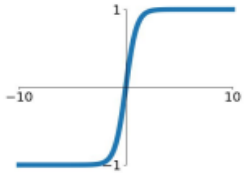
Sigmoid

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



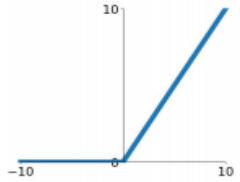
tanh

$$\tanh(x)$$



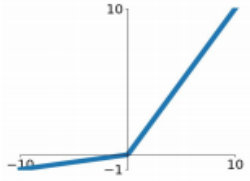
ReLU

$$\max(0, x)$$



Leaky ReLU

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

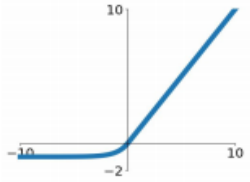


Maxout

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

ELU

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



Source: cs231n, Stanford University

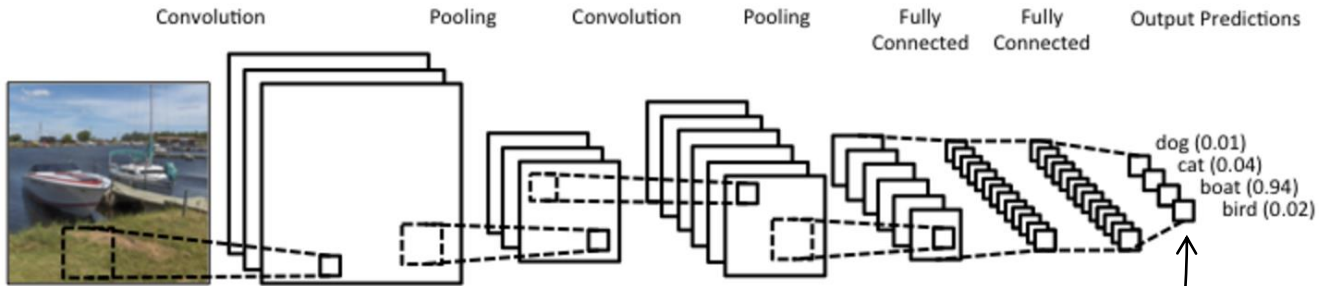
Classification/Loss Layer

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

- Softmax Classifier
- Softmax Loss/Cross-entropy Loss

A typical CNN structure

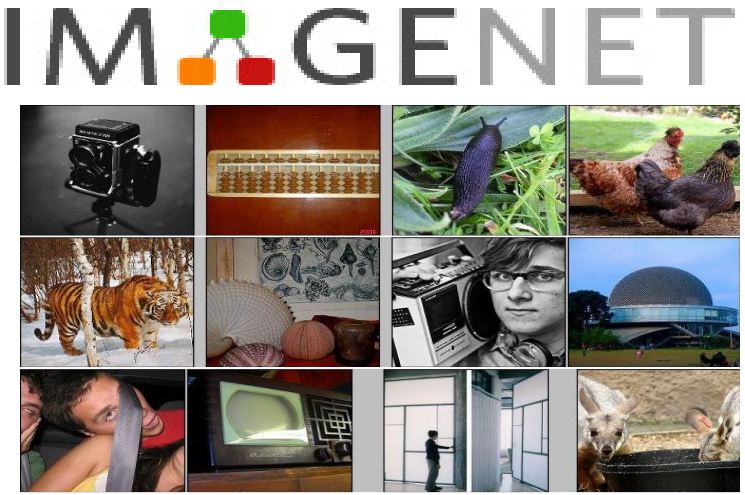




Softmax layer:

Image Source: machinethink.net

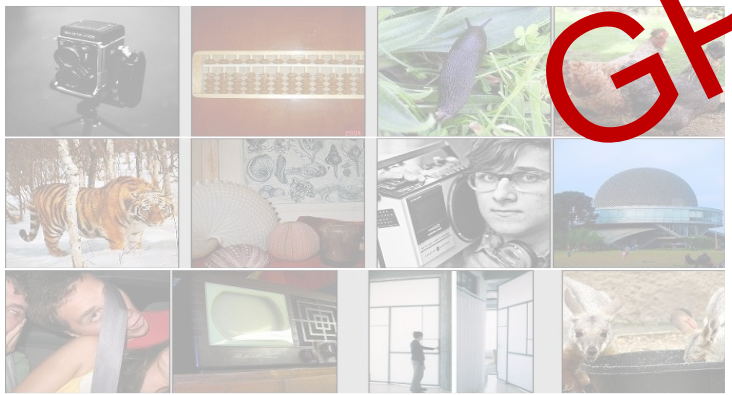
ImageNet Challenge



- ~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

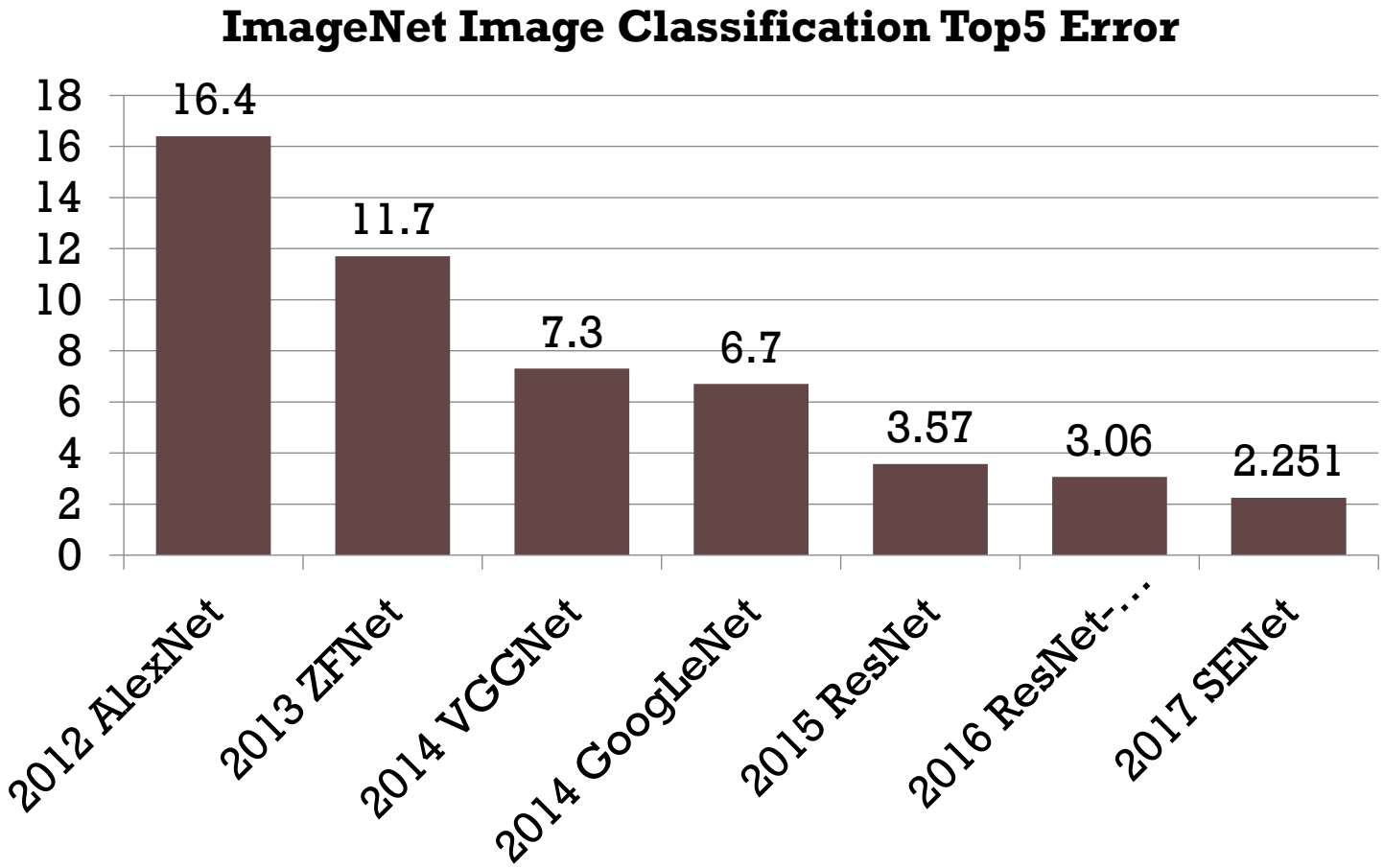


- 14 million labeled images, 20k classes
- Images gathered from Internet
- Human labels via Amazon MTurk
- Challenge: 1.2 million training images, 1000 classes

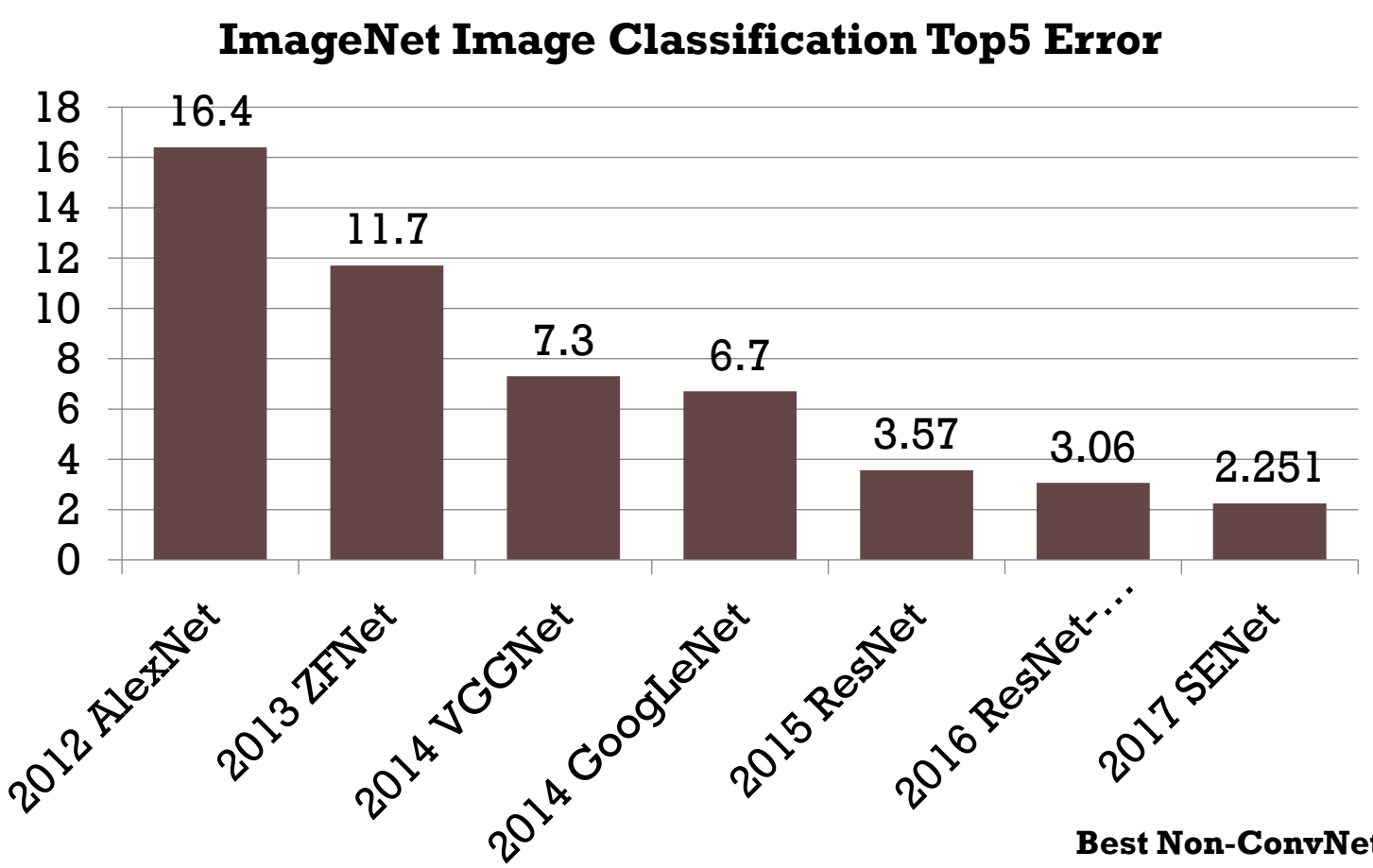
GPUs + Data

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

Progress on ImageNet Challenge



Progress on ImageNet Challenge





Best Non-ConvNet in 2012: 26.2%

Transfer Learning with CNNs

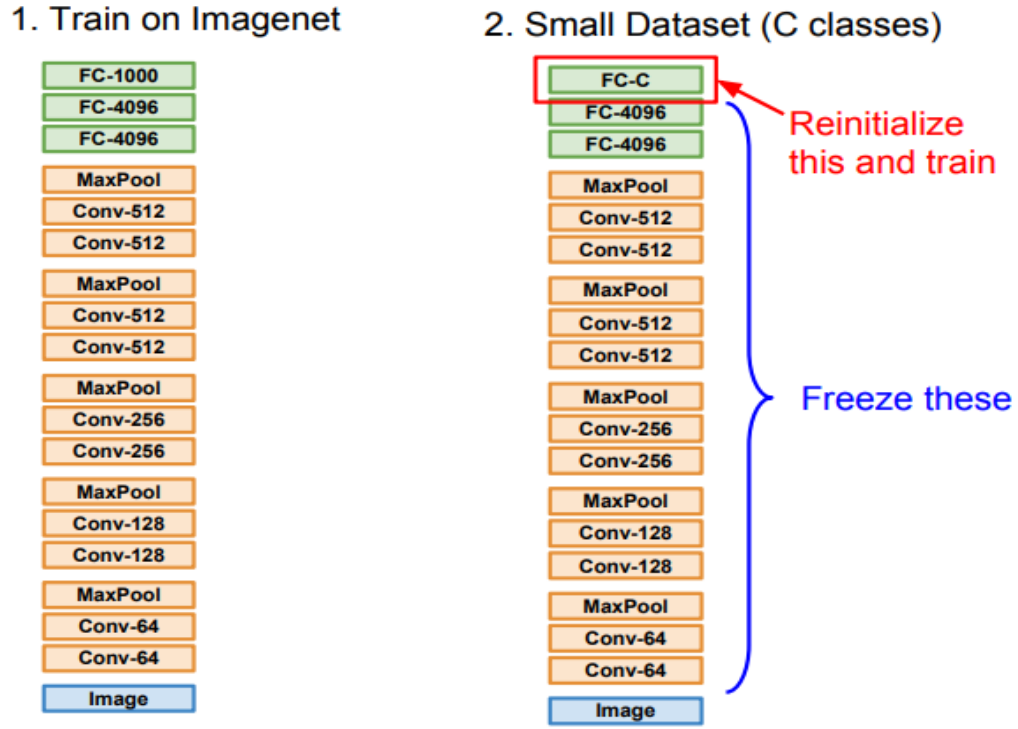
1. Train on Imagenet



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

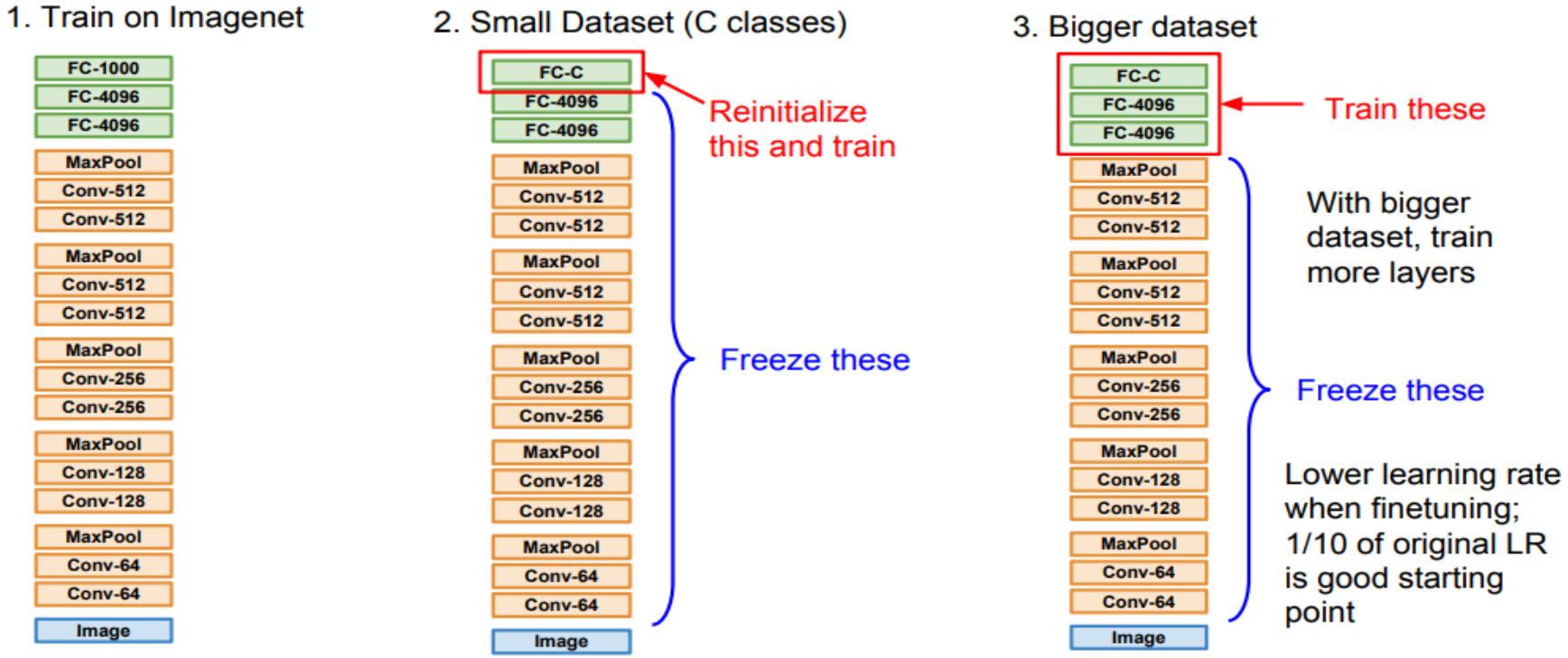


Donahue et al, "DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition", ICML 2014

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

Source: cs231n

Transfer Learning with CNNs

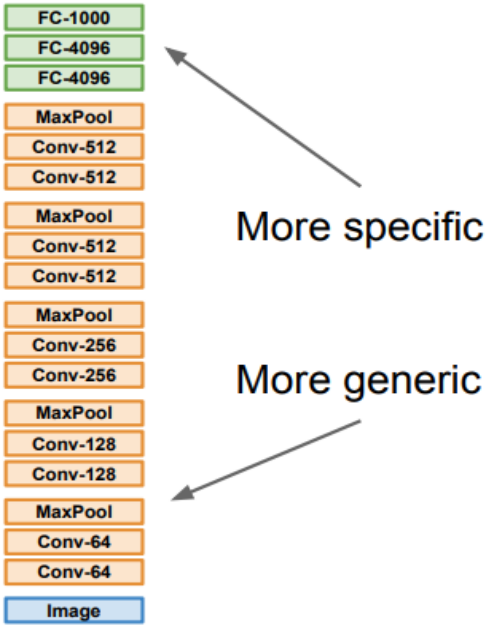


Donahue et al, "DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition", ICML 2014

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

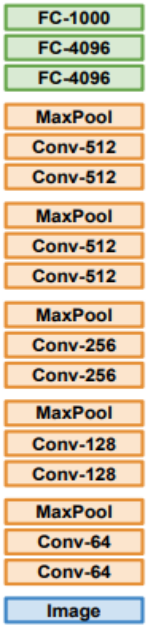
Source: cs231n

Transfer Learning with CNNs



Source: cs231n

Transfer Learning with CNNs



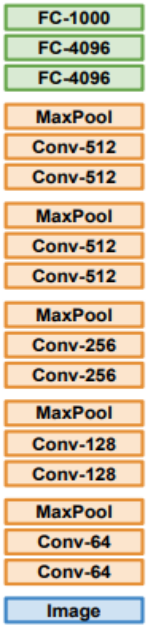
More specific

More generic

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

Source: cs231n

Transfer Learning with CNNs



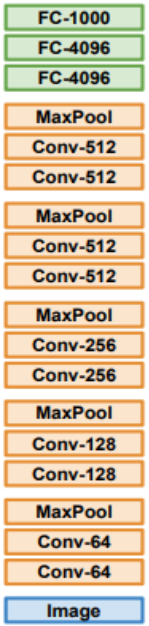
More specific

More generic

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

Source: cs231n

Transfer Learning with CNNs



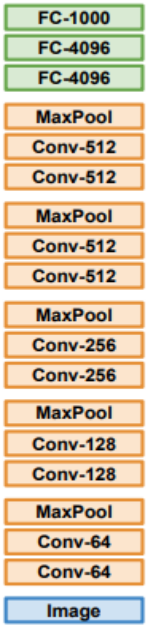
More specific

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	

Source: cs231n

Transfer Learning with CNNs



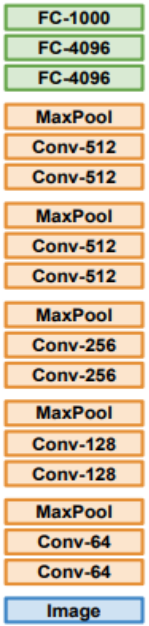
More specific

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

Source: cs231n

Transfer Learning with CNNs



	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

Source: cs231n

Transfer Learning with CNNs

Takeaway for your projects and beyond:

Have some dataset of interest but it has < ~1M 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/

Source: cs231n

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

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

THANK YOU