Convolutional Neural Networks

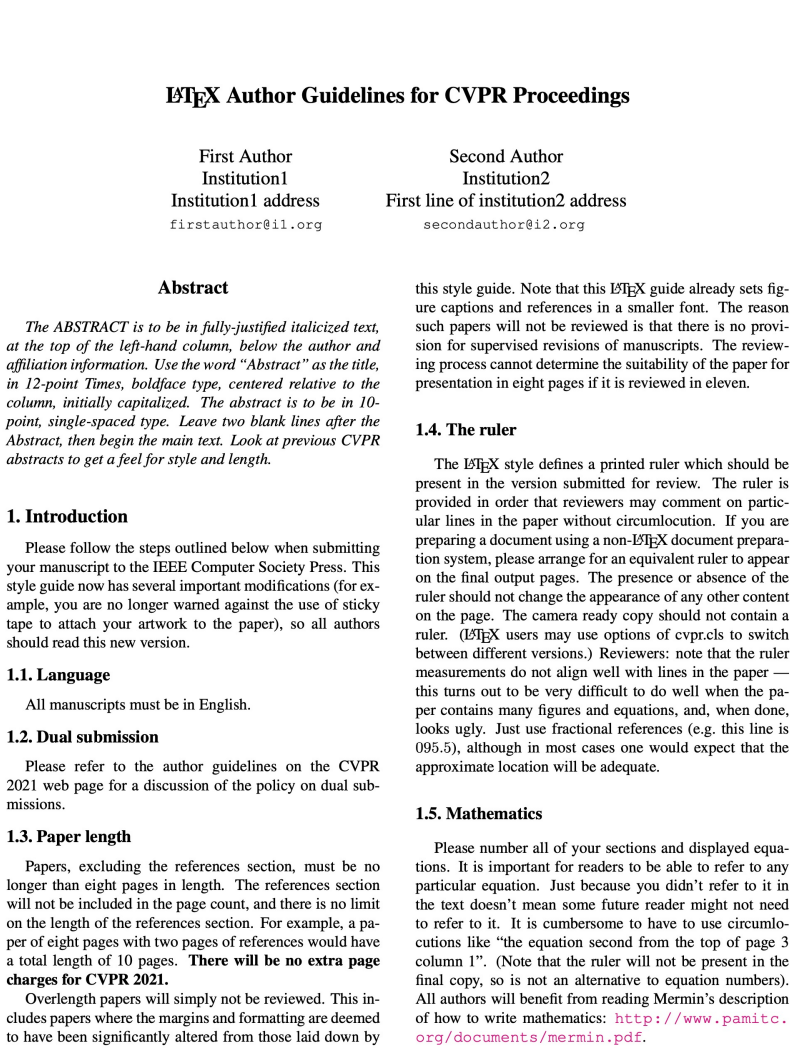
CS5330, Huaizu Jiang

Fall 2021, Northeastern University

Announcement

- Grades of pa2 have been released
  - Attend the office hour for the solution
- Pa4 has been released
  - Due on Nov 22 at 11:59pm
  - Plan accordingly (it's due before the Thanksgiving recess)
  - Pay attention to how many late days you have used
- Grades of the project proposal abstract have been released
  - Check the grades with your group members (if any)
  - Talk to me if necessary
  - Format issue of the report

Format for the Final Project Report



LaTeX Author Guidelines for CVPR Proceedings

First Author
Institution1
Institution1 address
firstauthor@i1.org

Second Author
Institution2
First line of institution2 address
secondauthor@i2.org

Abstract

The ABSTRACT is to be in fully-justified italicized text, at the top of the left-hand column, below the author and affiliation information. Use the word "Abstract" as the title, in 12-point Times, boldface type, centered relative to the column, initially capitalized. The abstract is to be in 10-point, single-spaced type. Leave two blank lines after the Abstract, then begin the main text. Look at previous CVPR abstracts to get a feel for style and length.

1. Introduction

Please follow the steps outlined below when submitting your manuscript to the IEEE Computer Society Press. This style guide now has several important modifications (for example, you are no longer warned against the use of sticky tape to attach your artwork to the paper), so all authors should read this new version.

1.1. Language

All manuscripts must be in English.

1.2. Dual submission

Please refer to the author guidelines on the CVPR 2021 web page for a discussion of the policy on dual submissions.

1.3. Paper length

Papers, excluding the references section, must be no longer than eight pages in length. The references section will not be included in the page count, and there is no limit on the length of the references section. For example, a paper of eight pages with two pages of references would have a total length of 10 pages. **There will be no extra page charges for CVPR 2021.**

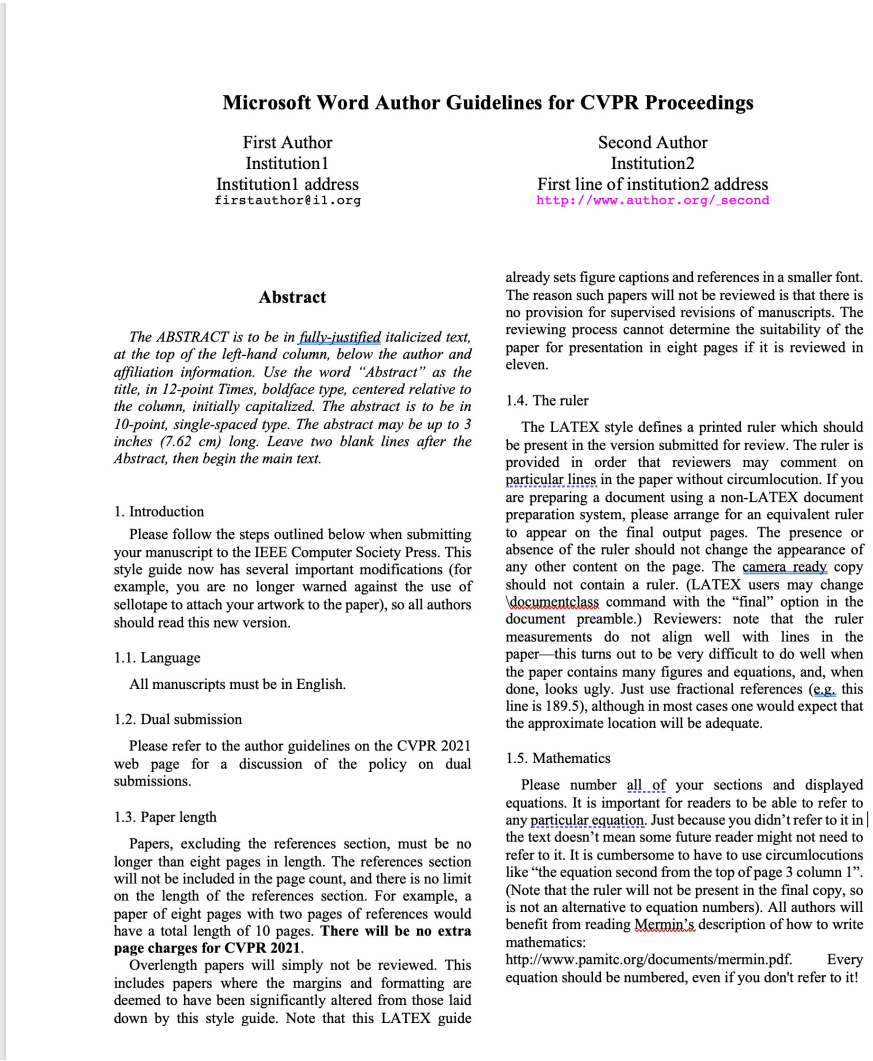
Overlength papers will simply not be reviewed. This includes papers where the margins and formatting are deemed to have been significantly altered from those laid down by this style guide. Note that this LaTeX guide already sets figure captions and references in a smaller font. The reason such papers will not be reviewed is that there is no provision for supervised revisions of manuscripts. The reviewing process cannot determine the suitability of the paper for presentation in eight pages if it is reviewed in eleven.

1.4. The ruler

The LaTeX style defines a printed ruler which should be present in the version submitted for review. The ruler is provided in order that reviewers may comment on particular lines in the paper without circumlocution. If you are preparing a document using a non-LaTeX document preparation system, please arrange for an equivalent ruler to appear on the final output pages. The presence or absence of the ruler should not change the appearance of any other content on the page. The camera ready copy should not contain a ruler. (LaTeX users may use options of cvpr.cls to switch between different versions.) Reviewers: note that the ruler measurements do not align well with lines in the paper — this turns out to be very difficult to do well when the paper contains many figures and equations, and, when done, looks ugly. Just use fractional references (e.g. this line is 095.5), although in most cases one would expect that the approximate location will be adequate.

1.5. Mathematics

Please number all of your sections and displayed equations. It is important for readers to be able to refer to any particular equation. Just because you didn't refer to it in the text doesn't mean some future reader might not need to refer to it. It is cumbersome to have to use circumlocutions like "the equation second from the top of page 3 column 1". (Note that the ruler will not be present in the final copy, so is not an alternative to equation numbers). All authors will benefit from reading Mermin's description of how to write mathematics: http://www.pamitc.org/documents/mermin.pdf.



Microsoft Word Author Guidelines for CVPR Proceedings

First Author
Institution1
Institution1 address
firstauthor@i1.org

Second Author
Institution2
First line of institution2 address
http://www.author.org/_second

Abstract

The ABSTRACT is to be in fully-justified italicized text, at the top of the left-hand column, below the author and affiliation information. Use the word "Abstract" as the title, in 12-point Times, boldface type, centered relative to the column, initially capitalized. The abstract is to be in 10-point, single-spaced type. The abstract may be up to 3 inches (7.62 cm) long. Leave two blank lines after the Abstract, then begin the main text.

1. Introduction

Please follow the steps outlined below when submitting your manuscript to the IEEE Computer Society Press. This style guide now has several important modifications (for example, you are no longer warned against the use of sellotape to attach your artwork to the paper), so all authors should read this new version.

1.1. Language

All manuscripts must be in English.

1.2. Dual submission

Please refer to the author guidelines on the CVPR 2021 web page for a discussion of the policy on dual submissions.

1.3. Paper length

Papers, excluding the references section, must be no longer than eight pages in length. The references section will not be included in the page count, and there is no limit on the length of the references section. For example, a paper of eight pages with two pages of references would have a total length of 10 pages. **There will be no extra page charges for CVPR 2021**.

Overlength papers will simply not be reviewed. This includes papers where the margins and formatting are deemed to have been significantly altered from those laid down by this style guide. Note that this LATEX guide already sets figure captions and references in a smaller font. The reason such papers will not be reviewed is that there is no provision for supervised revisions of manuscripts. The reviewing process cannot determine the suitability of the paper for presentation in eight pages if it is reviewed in eleven.

1.4. The ruler

The LATEX style defines a printed ruler which should be present in the version submitted for review. The ruler is provided in order that reviewers may comment on particular lines in the paper without circumlocution. If you are preparing a document using a non-LATEX document preparation system, please arrange for an equivalent ruler to appear on the final output pages. The presence or absence of the ruler should not change the appearance of any other content on the page. The camera ready copy should not contain a ruler. (LATEX users may change \documentclass command with the "final" option in the document preamble.) Reviewers: note that the ruler measurements do not align well with lines in the paper—this turns out to be very difficult to do well when the paper contains many figures and equations, and, when done, looks ugly. Just use fractional references (e.g. this line is 189.5), although in most cases one would expect that the approximate location will be adequate.

1.5. Mathematics

Please number all of your sections and displayed equations. It is important for readers to be able to refer to any particular equation. Just because you didn't refer to it in the text doesn't mean some future reader might not need to refer to it. It is cumbersome to have to use circumlocutions like "the equation second from the top of page 3 column 1". (Note that the ruler will not be present in the final copy, so is not an alternative to equation numbers). All authors will benefit from reading Mermin's description of how to write mathematics: http://www.pamitc.org/documents/mermin.pdf. Every equation should be numbered, even if you don't refer to it!

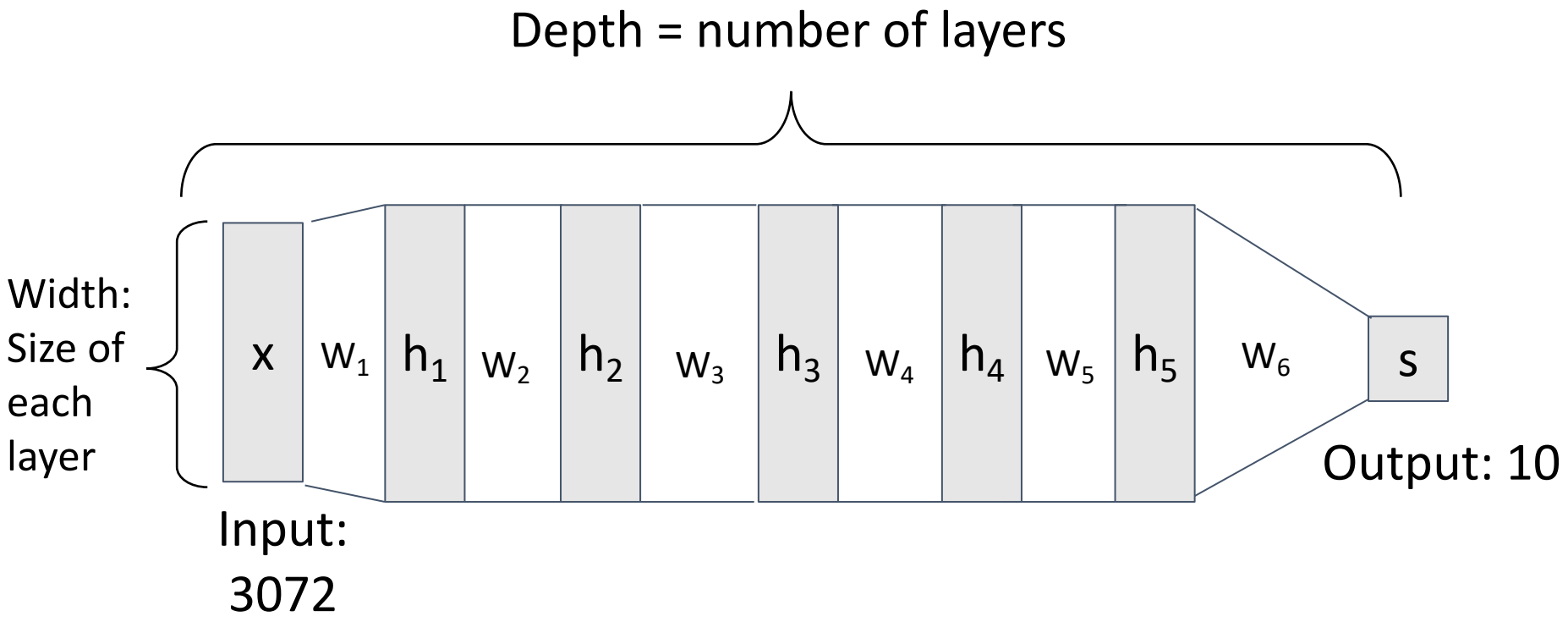
1. No point deductions about the format for the abstract
2. The requirement will be enforced for the final project

Remember:

No late days for the final project

Recap

Deep Neural Networks



$$s = W_6 \max(0, W_5 \max(0, W_4 \max(0, W_3 \max(0, W_2 \max(0, W_1 x)))))$$

Slide credit: D Fouhey & J Johnson

Problem: How to compute gradients?

$$s = W_2 \max(0, W_1 x) + b_2$$ Nonlinear score function

$$L_i = \sum_{j \neq y_i} \max\left(0, s_j - s_{y_i} + 1\right)$$ Per-element data loss

$$R(W) = \sum_k W_k^2$$ L2 Regularization

$$L(W_1, W_2) = \frac{1}{N} \sum_{i=1}^{N} L_i + \lambda R(W_1) + \lambda R(W_2)$$ Total loss

What do we need to optimize with SGD?

Compute $\frac{\partial L}{\partial W_1}, \frac{\partial L}{\partial W_2}$

Slide credit: D Fouhey & J Johnson

(Bad) Idea: Derive $\nabla_W L$ on paper

$$s = f(x; W) = Wx$$

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

$$= \sum_{j \neq y_i} \max(0, W_{j,:} \cdot x + W_{y_i,:} \cdot x + 1)$$

$$L = \frac{1}{N} \sum_{i=1}^{N} L_i + \lambda \sum_k W_k^2$$

$$= \frac{1}{N} \sum_{i=1}^{N} \sum_{j \neq y_i} \max(0, W_{j,:} \cdot x + W_{y_i,:} \cdot x + 1) + \lambda \sum_k W_k^2$$

$$\nabla_W L = \nabla_W \left(\frac{1}{N} \sum_{i=1}^{N} \sum_{j \neq y_i} \max(0, W_{j,:} \cdot x + W_{y_i,:} \cdot x + 1) + \lambda \sum_k W_k^2 \right)$$

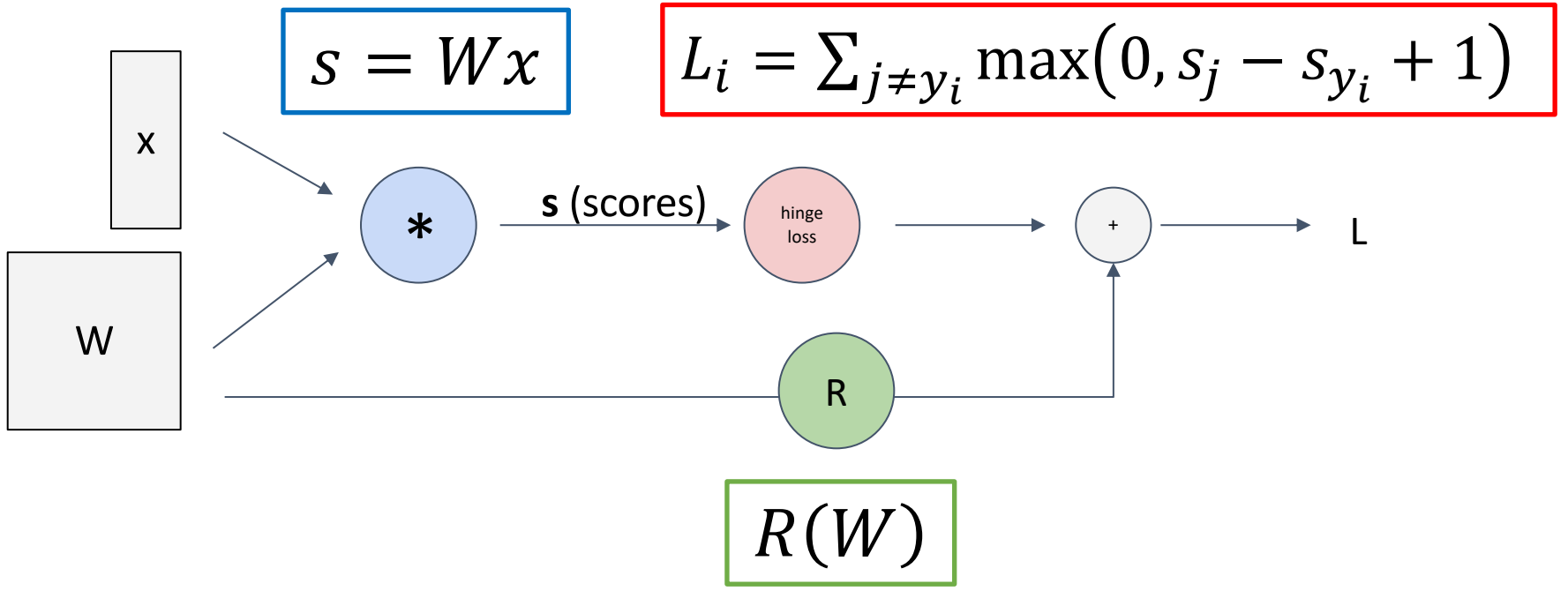
Problem: Very tedious: Lots of matrix calculus, need lots of paper

Problem: What if we want to change loss? E.g. use softmax instead of SVM? Need to re-derive from scratch. Not modular!

Problem: Not feasible for very complex models!

Slide credit: D Fouhey & J Johnson

Better Idea: **Computational Graphs**

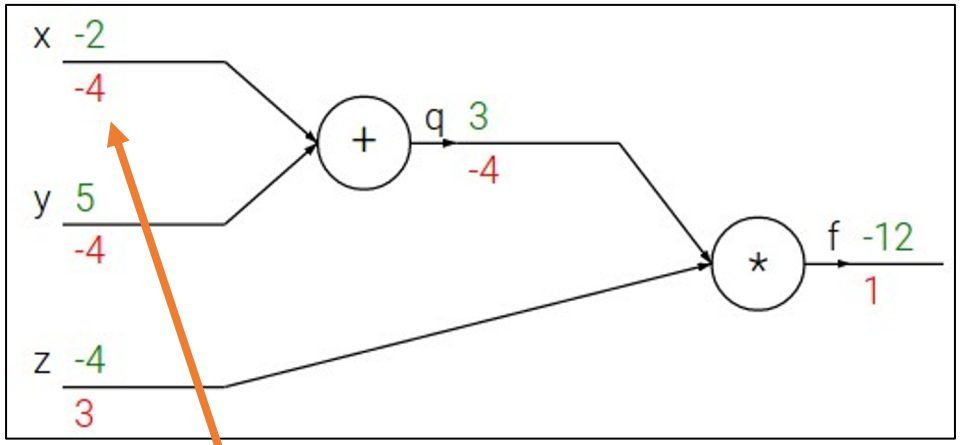


Slide credit: D Fouhey & J Johnson

Backpropagation: Simple Example

$$f(x, y, z) = (x + y) \cdot z$$

e.g. x = -2, y = 5, z = -4



1. Forward pass: Compute outputs

$$q = x + y \qquad f = q \cdot z$$

2. Backward pass: Compute derivatives

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

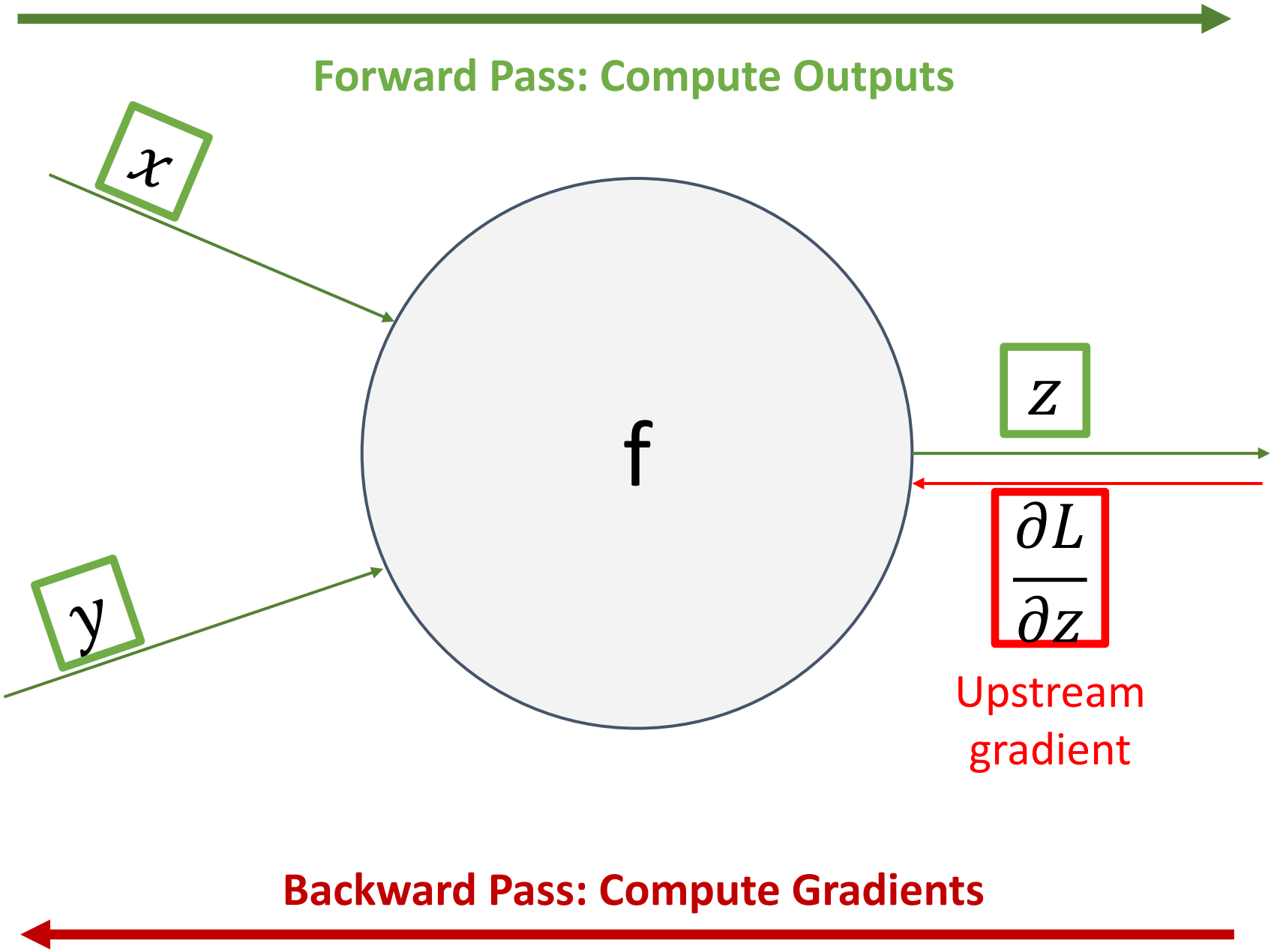
Chain Rule

$$\frac{\partial f}{\partial x} = \frac{\partial q}{\partial x}\frac{\partial f}{\partial q}$$

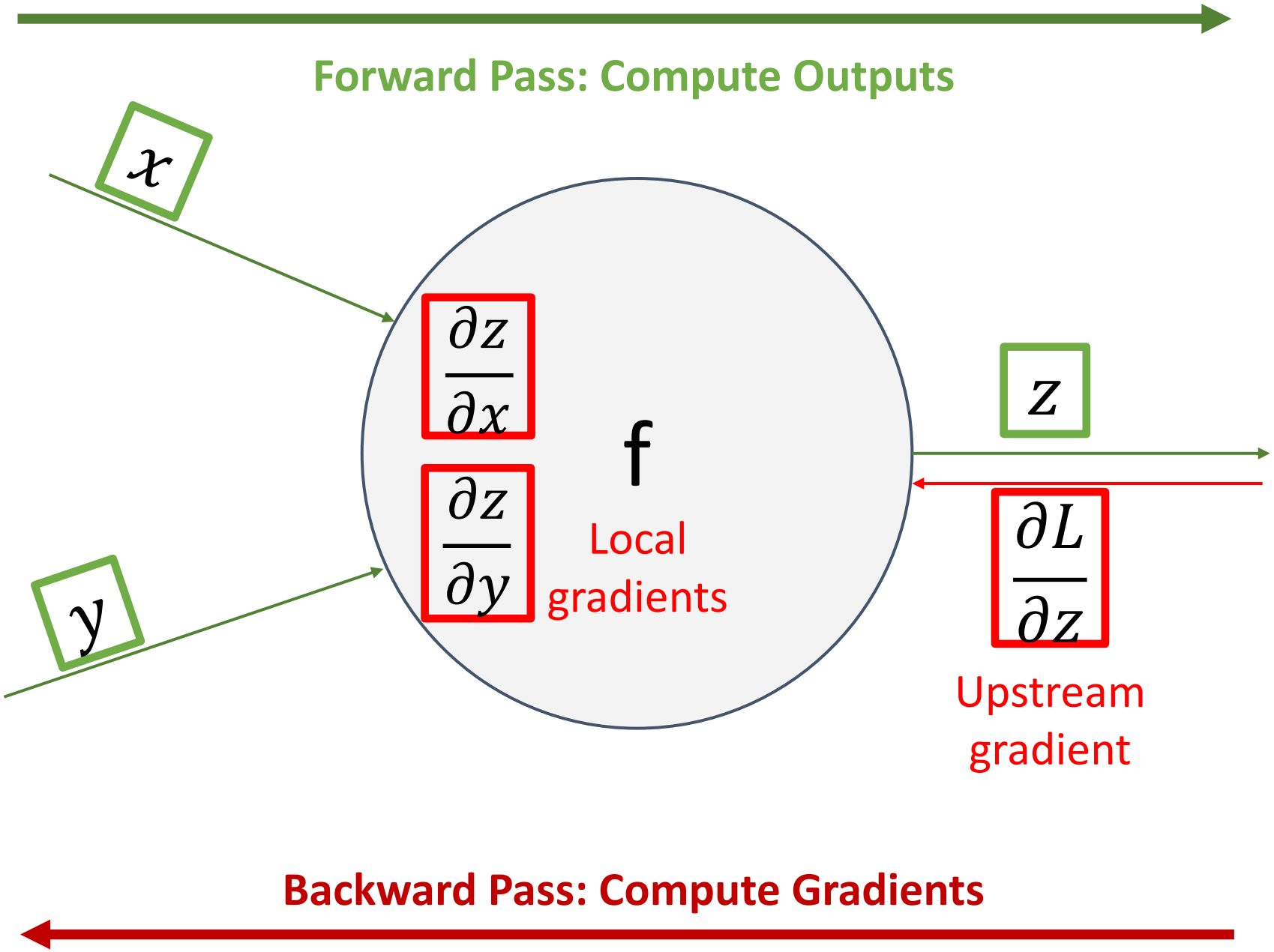
$$\frac{\partial q}{\partial x} = 1$$

Downstream Gradient **Local Gradient** **Upstream Gradient**

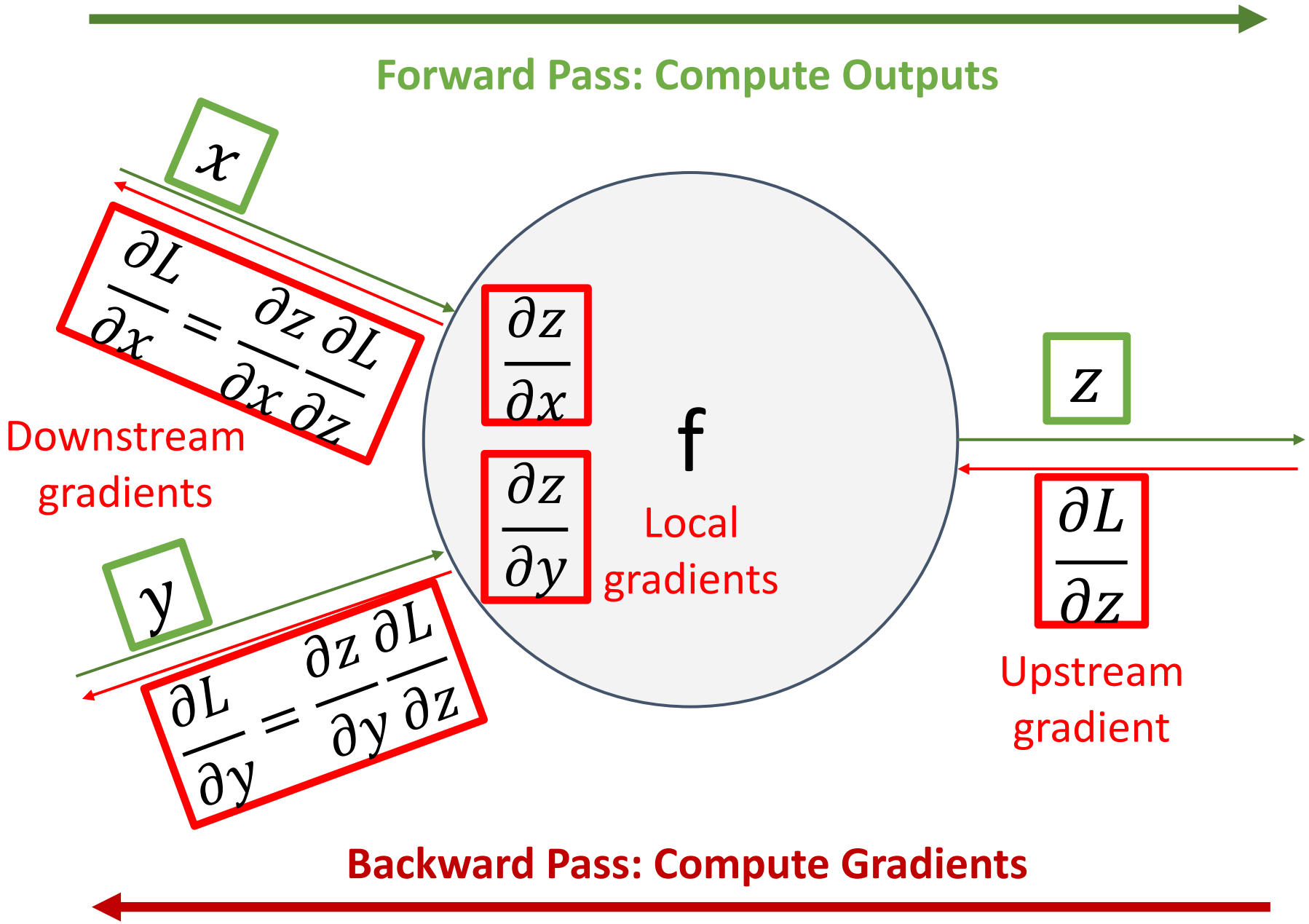
Slide credit: D Fouhey & J Johnson



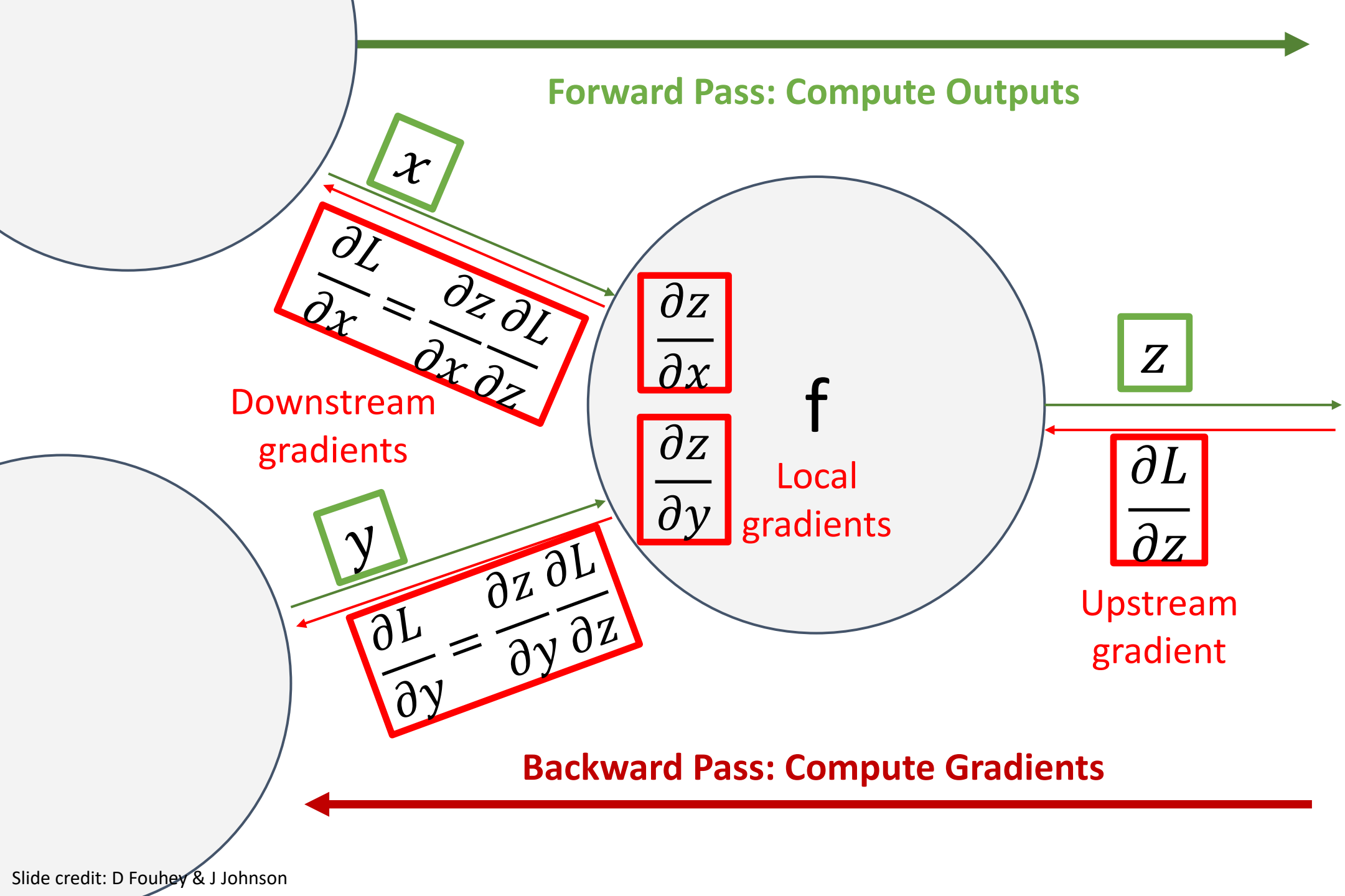
Slide credit: D Fouhey & J Johnson



Slide credit: D Fouhey & J Johnson

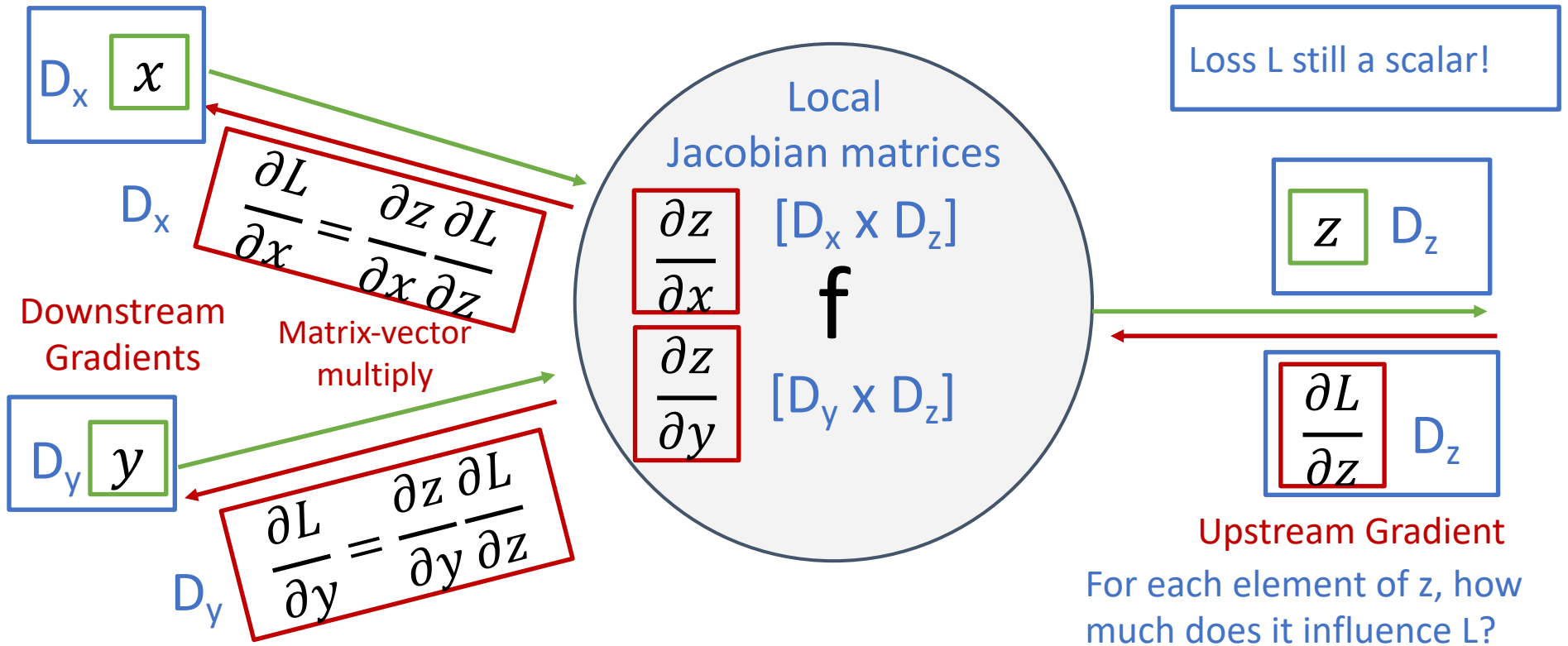


Slide credit: D Fouhey & J Johnson



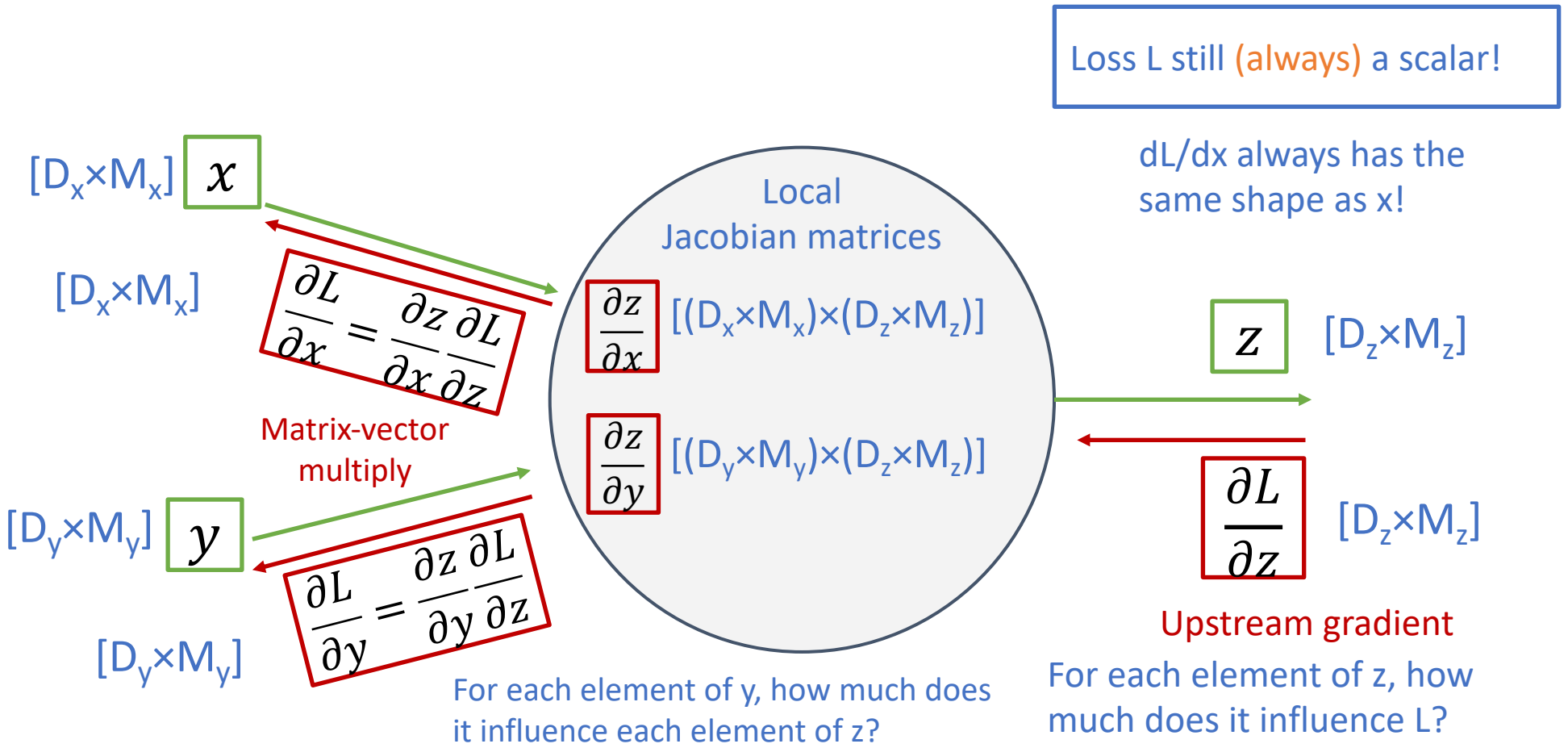
Slide credit: D Fouhey & J Johnson

Backprop with Vectors



Slide credit: D Fouhey & J Johnson

Backprop with Matrices (or Tensors):



Slide credit: D Fouhey & J Johnson

Extra Resources

- Prof. Erik Learned-Miller's notes on vector, matrix, and tensor derivatives

  https://compsci682-fa21.github.io/docs/vecDerivs.pdf

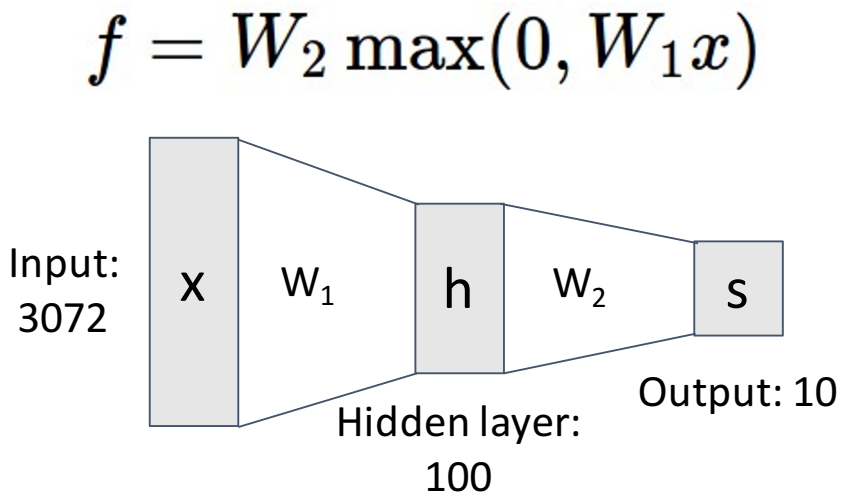
- Prof. Justin Johnson's notes for a derivation of backprop for matrix multiplication

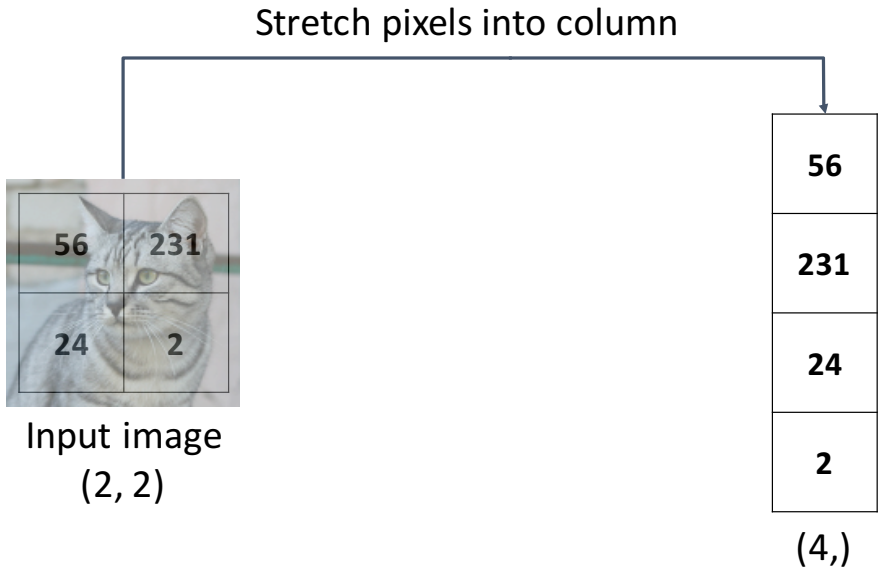
  https://web.eecs.umich.edu/~justincj/teaching/eecs442/notes/linear-backprop.html

Today’s Class

Convolutional Neural Networks (may take longer than 2 classes)

Problem: So far our classifiers don't respect the spatial structure of images!



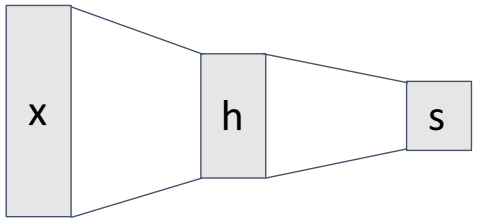


Solution: Define new computational nodes that operate on images!

Slide credit: D Fouhey & J Johnson

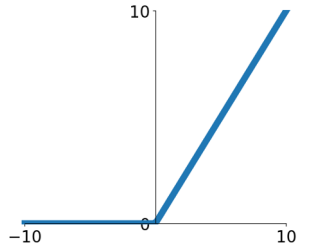
Components of a Fully-Connected Network

Fully-Connected Layers



$$y = Wx + b$$

Activation Function

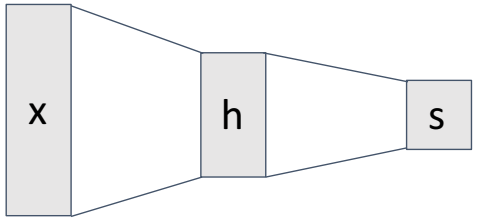


$$y = \max(0, x)$$

Slide credit: D Fouhey & J Johnson

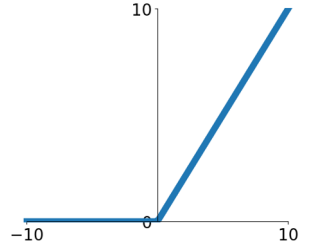
Components of a Convolutional Network

Fully-Connected Layers



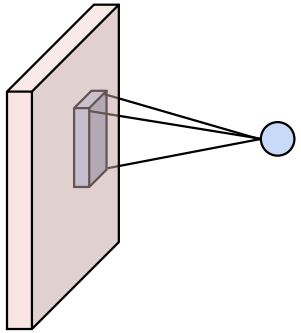
$$y = Wx + b$$

Activation Function

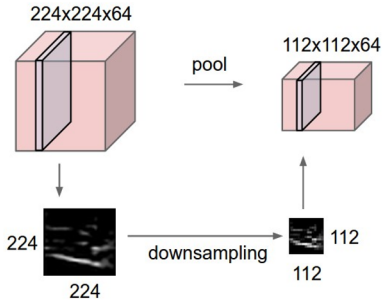


$$y = \max(0, x)$$

Convolution Layers



Pooling Layers



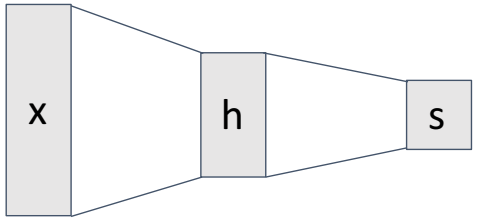
Normalization

$$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \varepsilon}}$$

Slide credit: D Fouhey & J Johnson

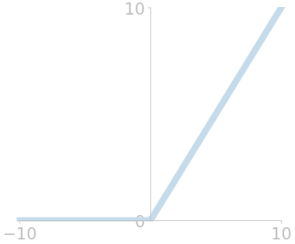
Components of a Convolutional Network

Fully-Connected Layers



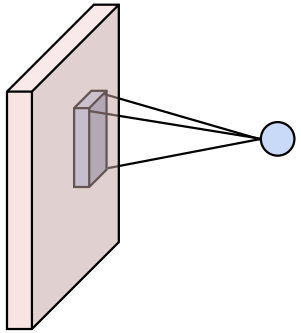
$$y = Wx + b$$

Activation Function

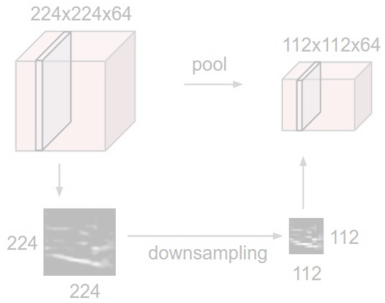


$$y = \max(0, x)$$

Convolution Layers



Pooling Layers



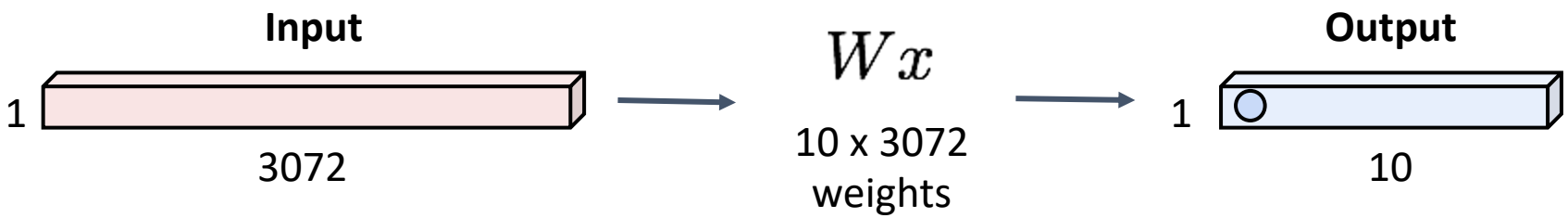
Normalization

$$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \varepsilon}}$$

Slide credit: D Fouhey & J Johnson

Fully-Connected Layer

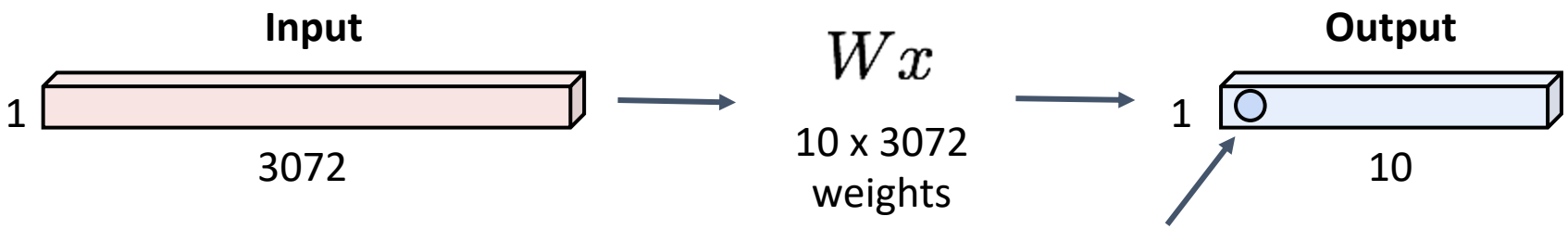
32x32x3 image -> stretch to 3072 x 1



Slide credit: D Fouhey & J Johnson

Fully-Connected Layer

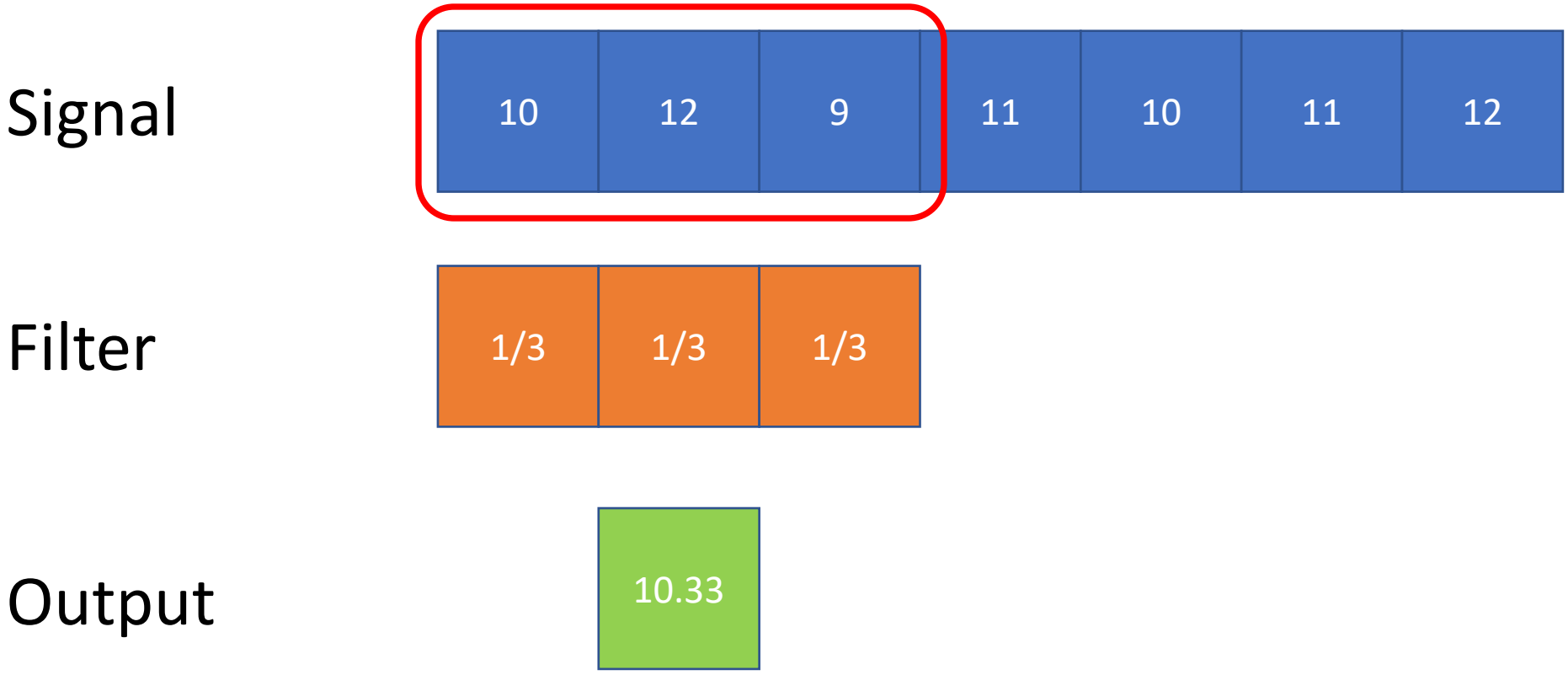
32x32x3 image -> stretch to 3072 x 1



1 number:
the result of taking a dot product between a row of W and the input (a 3072-dimensional dot product)

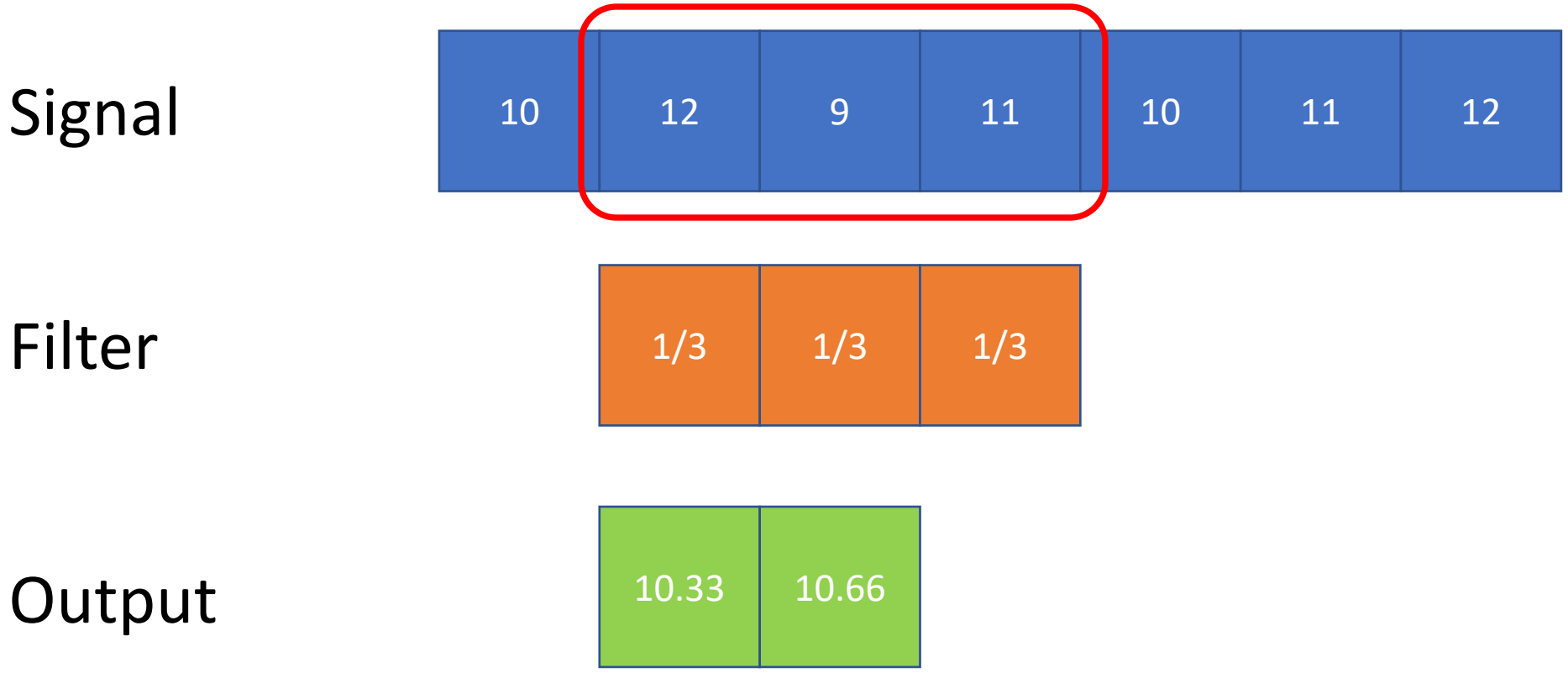
Slide credit: D Fouhey & J Johnson

Recap: 1D Case of Convolution



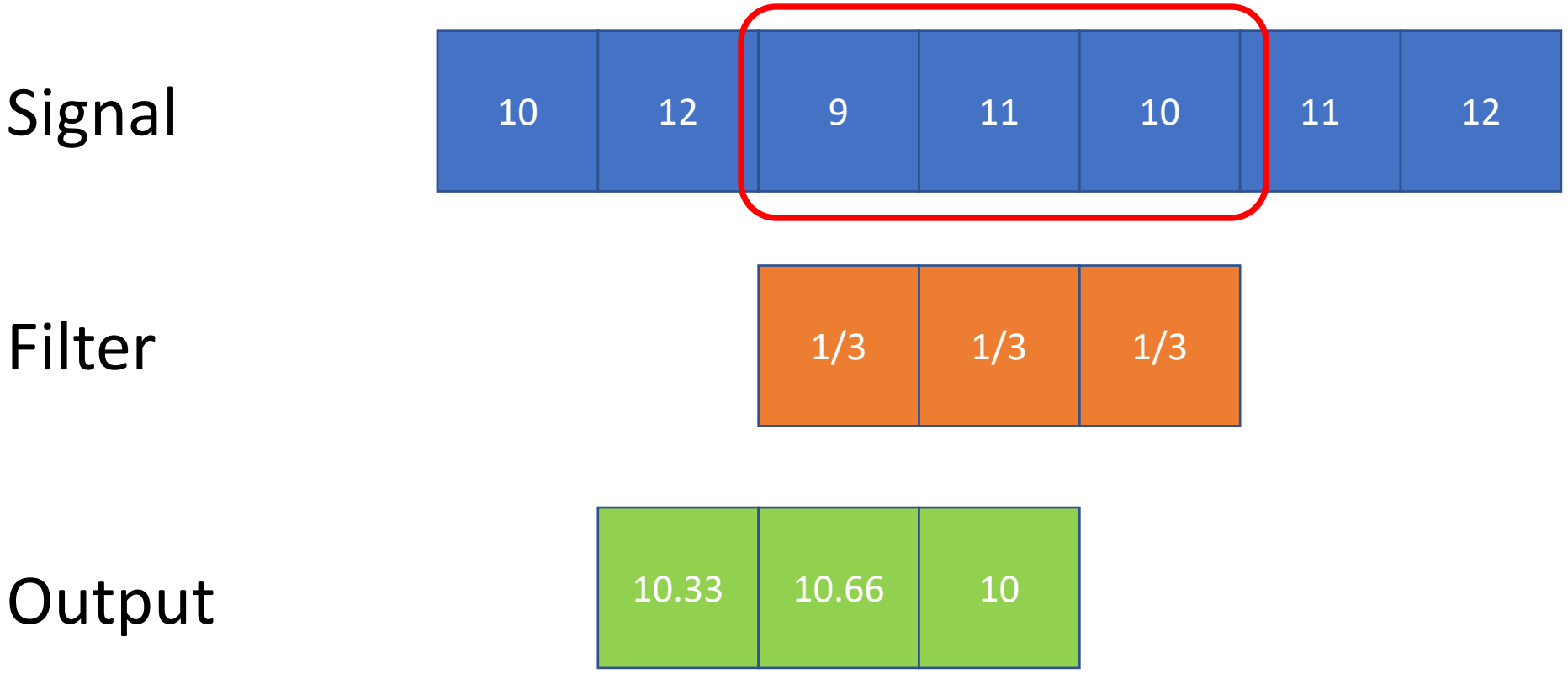
Slide adapted from : J Johnson

Recap: 1D Case of Convolution



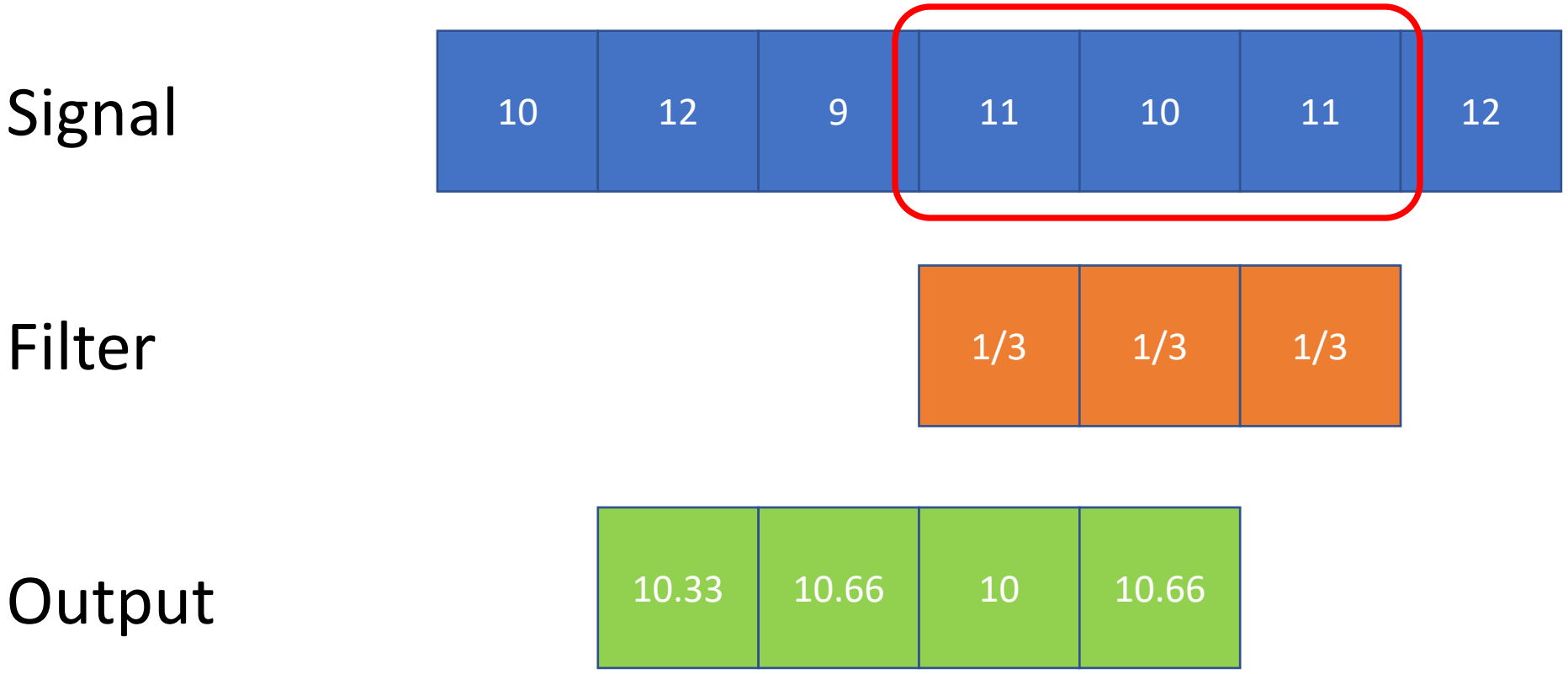
Slide adapted from : J Johnson

Recap: 1D Case of Convolution



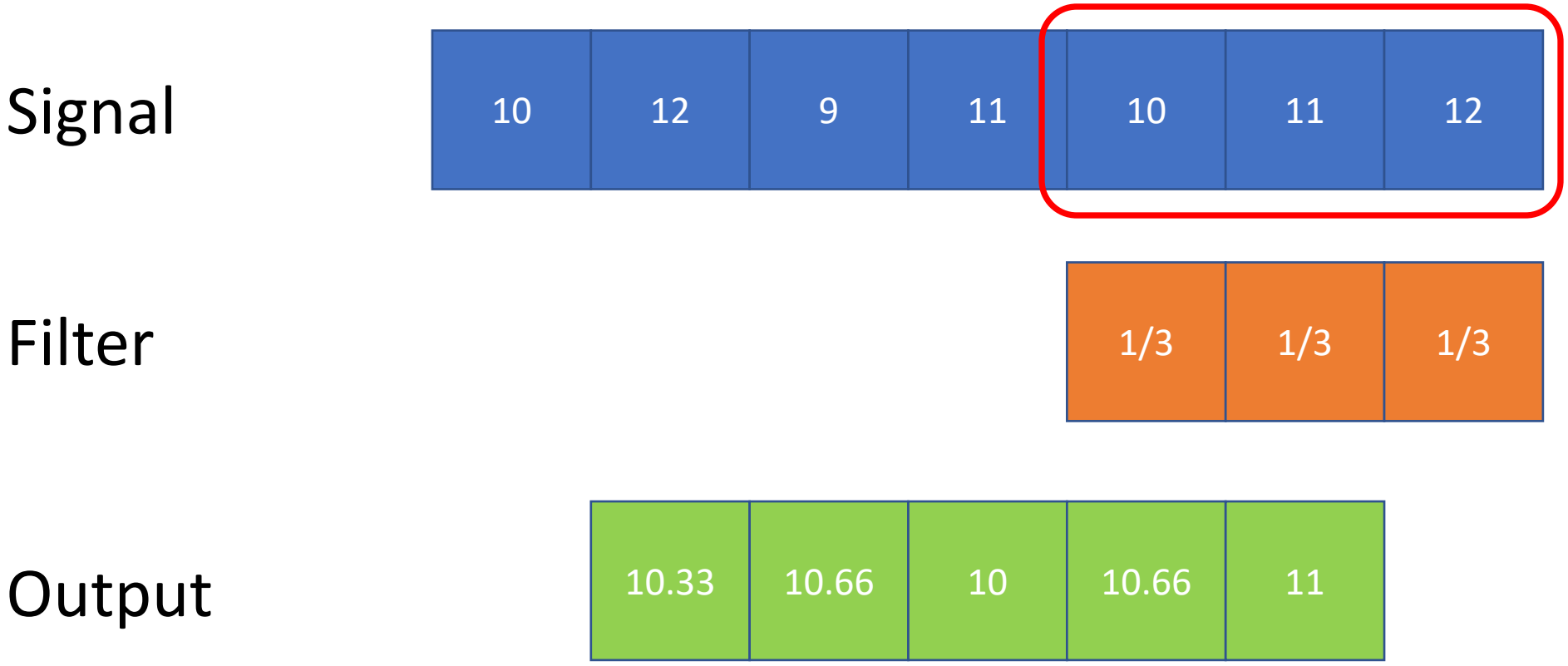
Slide adapted from : J Johnson

Recap: 1D Case of Convolution



Slide adapted from : J Johnson

Recap: 1D Case of Convolution



Slide adapted from : J Johnson

Recap: Image filtering

$g[\cdot,\cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[\cdot,\cdot]$

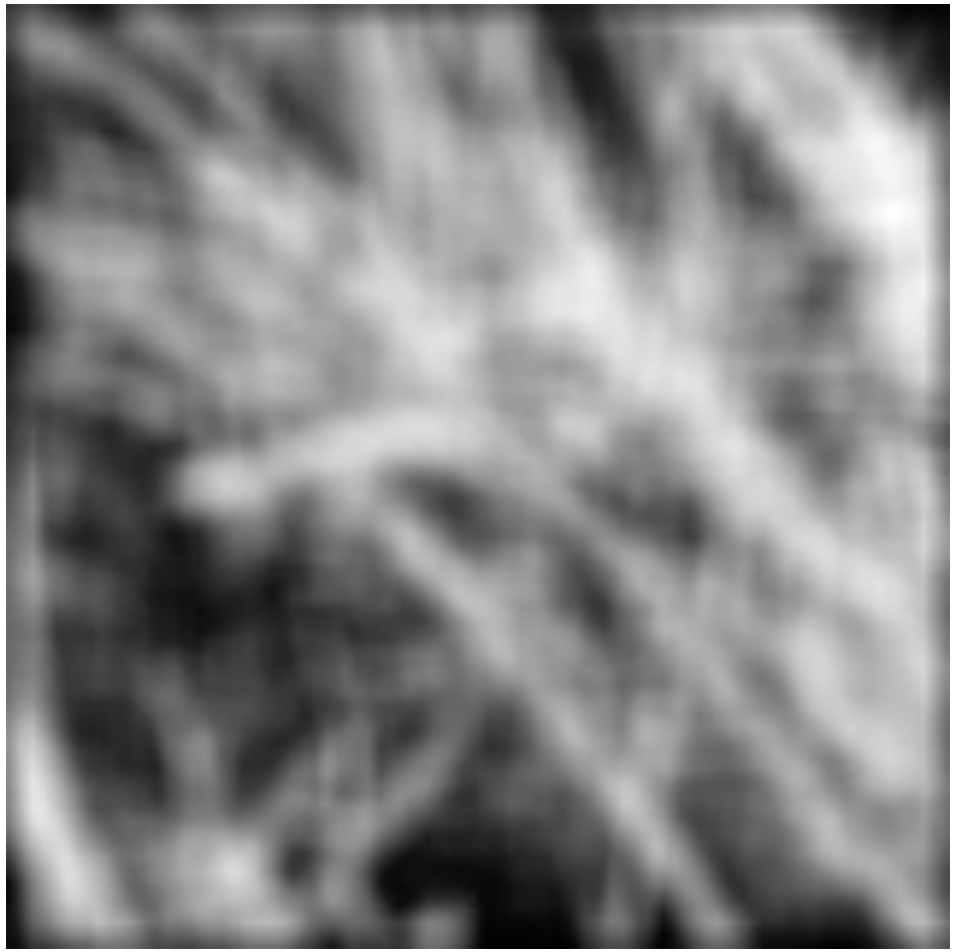
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[\cdot,\cdot]$

$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k,n+l]$$

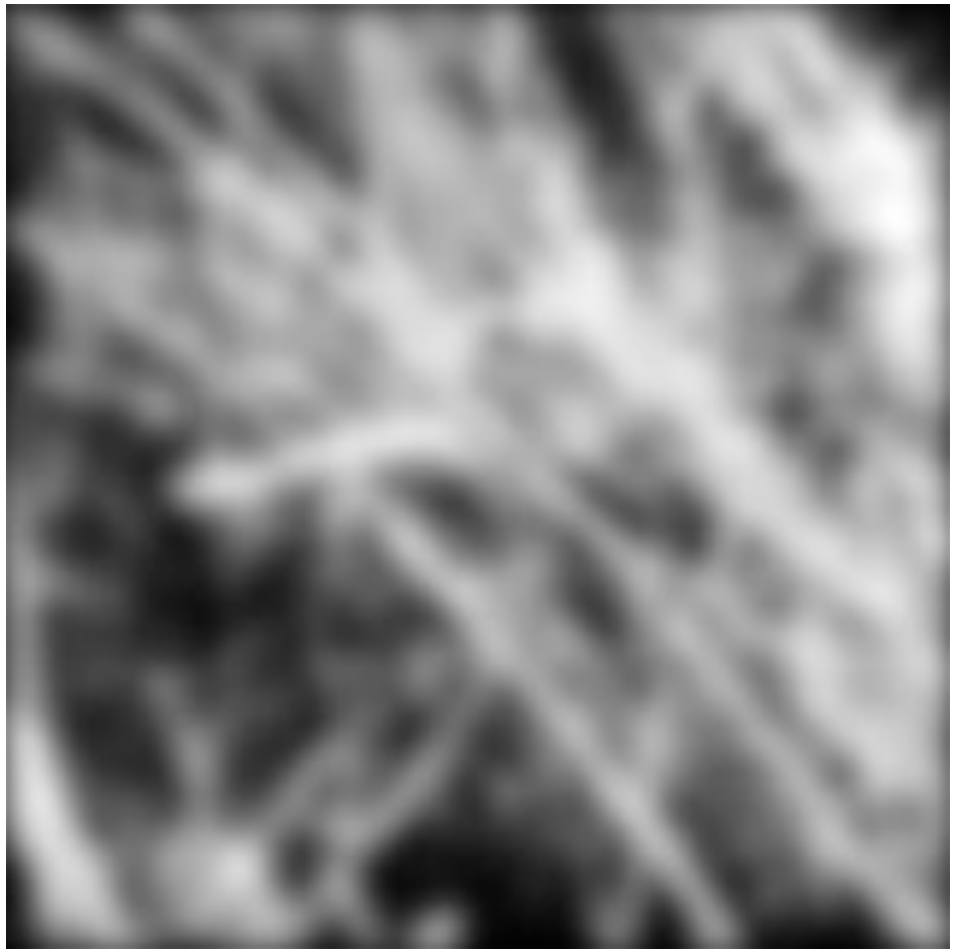
Credit: S. Seitz

Recap: Smoothing with box filter



Slide credit: D Hoiem

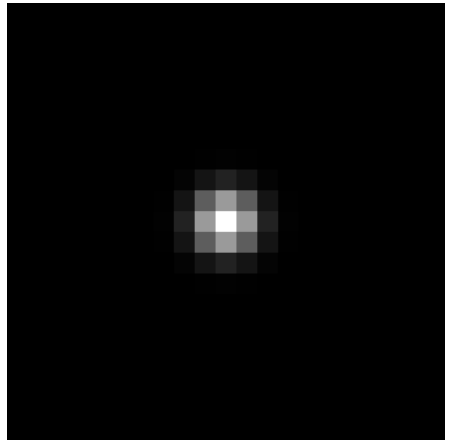
Recap: Smoothing with Gaussian filter



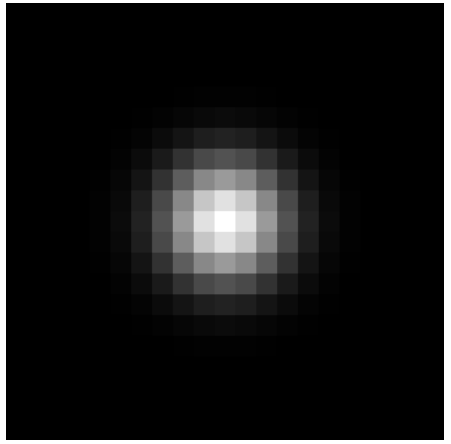
Slide credit: D Hoiem

Recap: Gaussian Filters

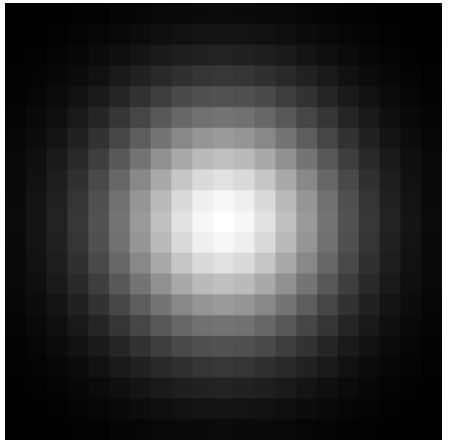
σ = 1
filter = 21x21



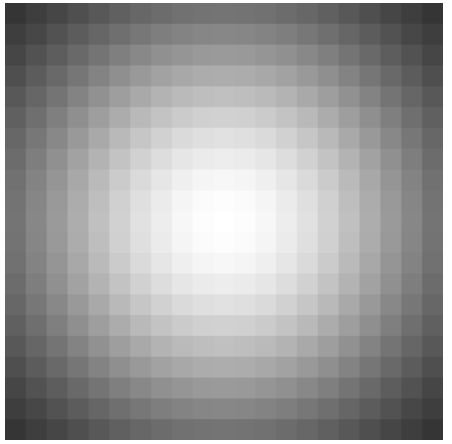
σ = 2
filter = 21x21



σ = 4
filter = 21x21



σ = 8
filter = 21x21

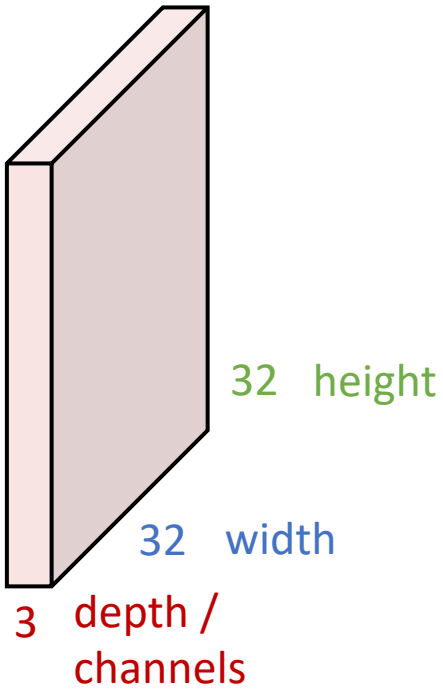


Note: filter visualizations are independently normalized throughout the slides so you can see them better

Slide credit: J Johnson

Convolution Layer

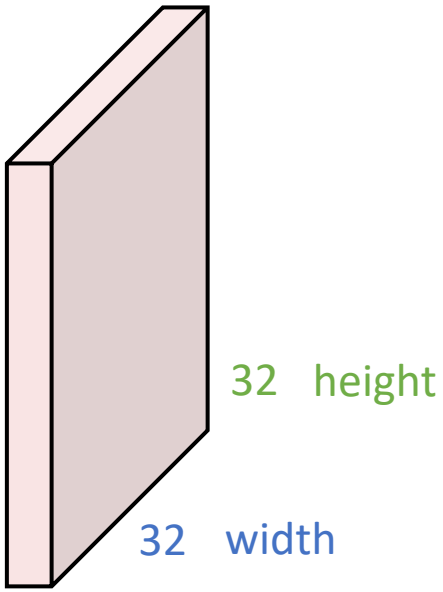
3x32x32 image: preserve spatial structure



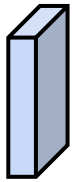
Slide credit: D Fouhey & J Johnson

Convolution Layer

3x32x32 image



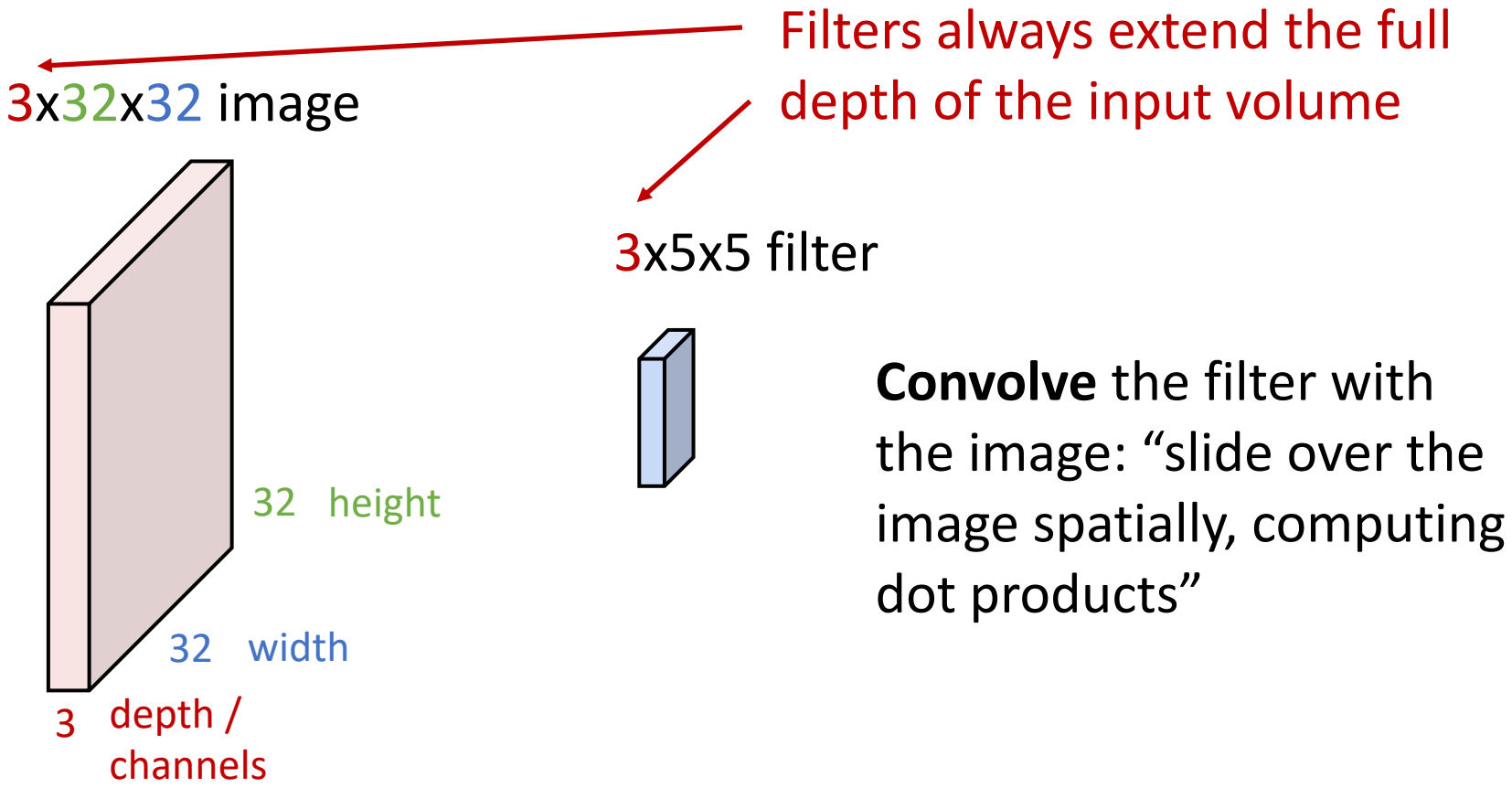
3x5x5 filter



Convolve the filter with the image: “slide over the image spatially, computing dot products”

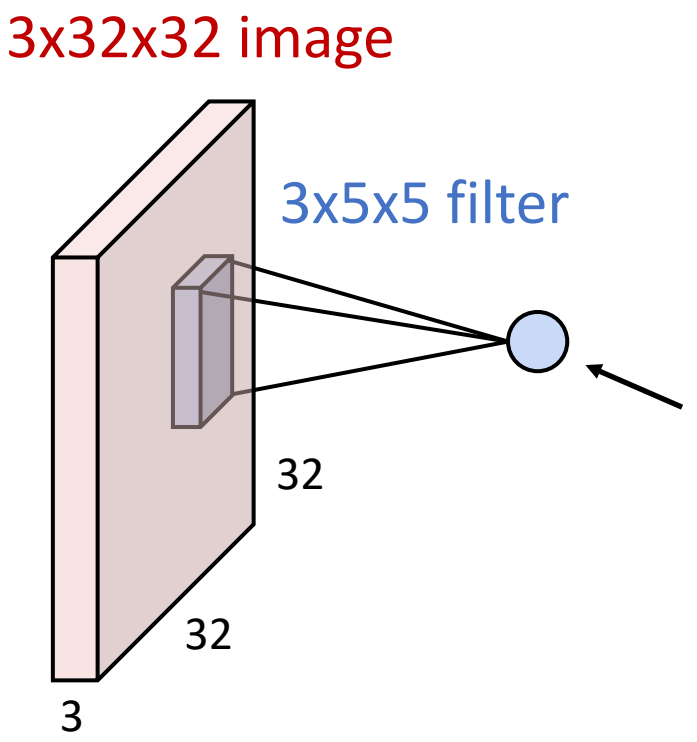
Slide credit: D Fouhey & J Johnson

Convolution Layer



Slide credit: D Fouhey & J Johnson

Convolution Layer

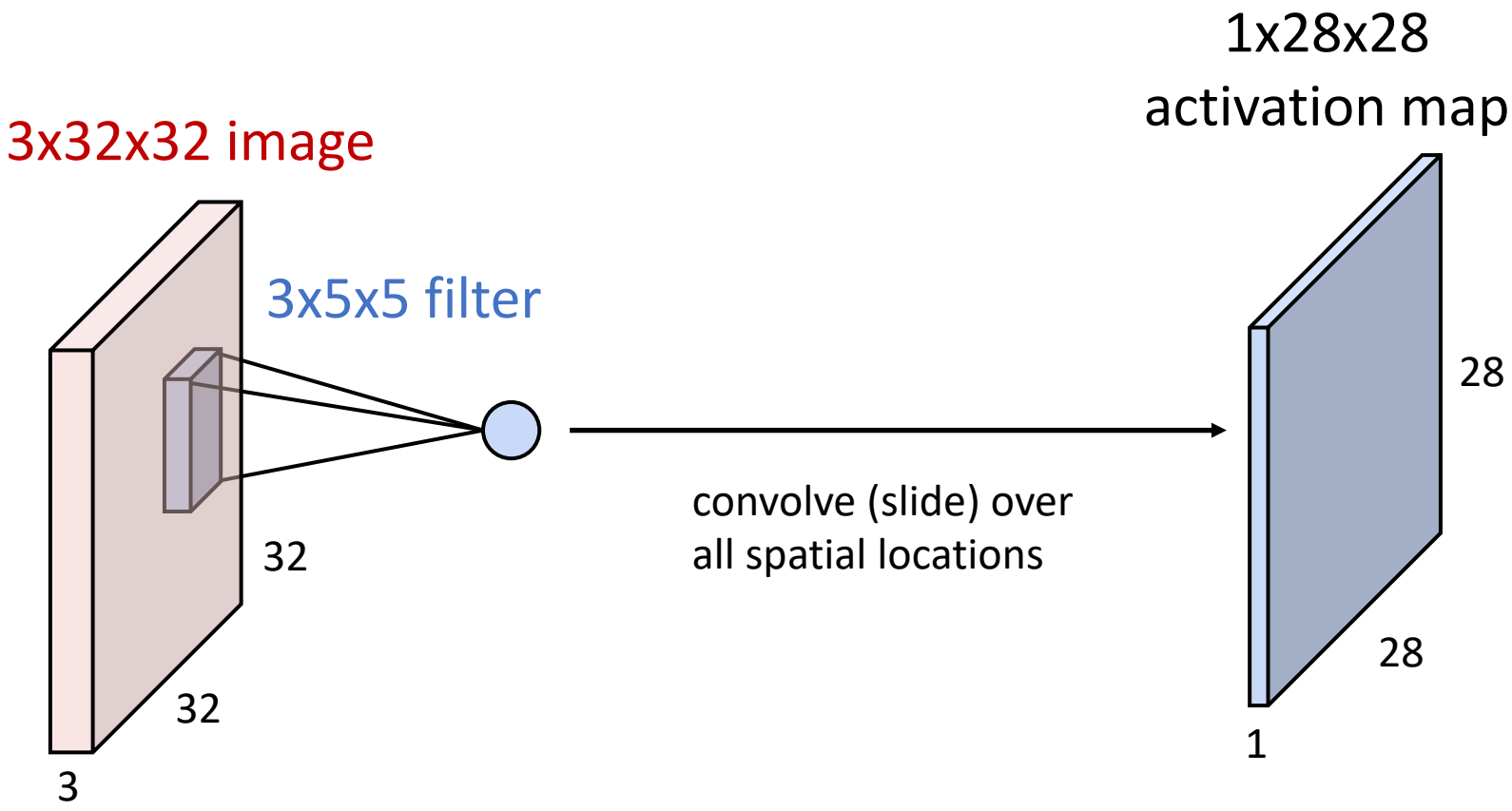


1 number:
the result of taking a dot product between the filter and a small 3x5x5 chunk of the image
(i.e. 3*5*5 = 75-dimensional dot product + bias)

$$w^T x + b$$

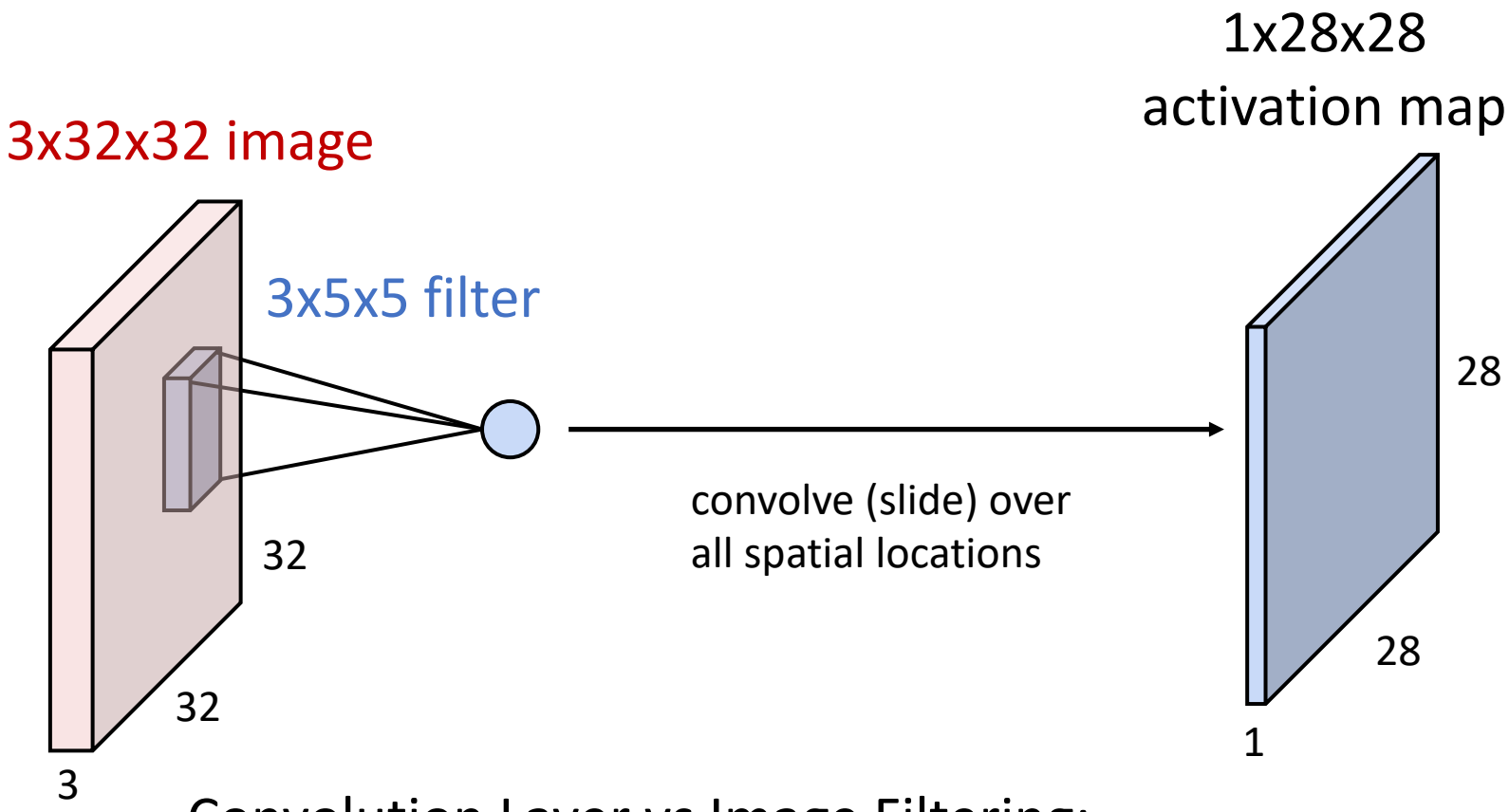
Slide credit: D Fouhey & J Johnson

Convolution Layer



Slide credit: D Fouhey & J Johnson

Convolution Layer



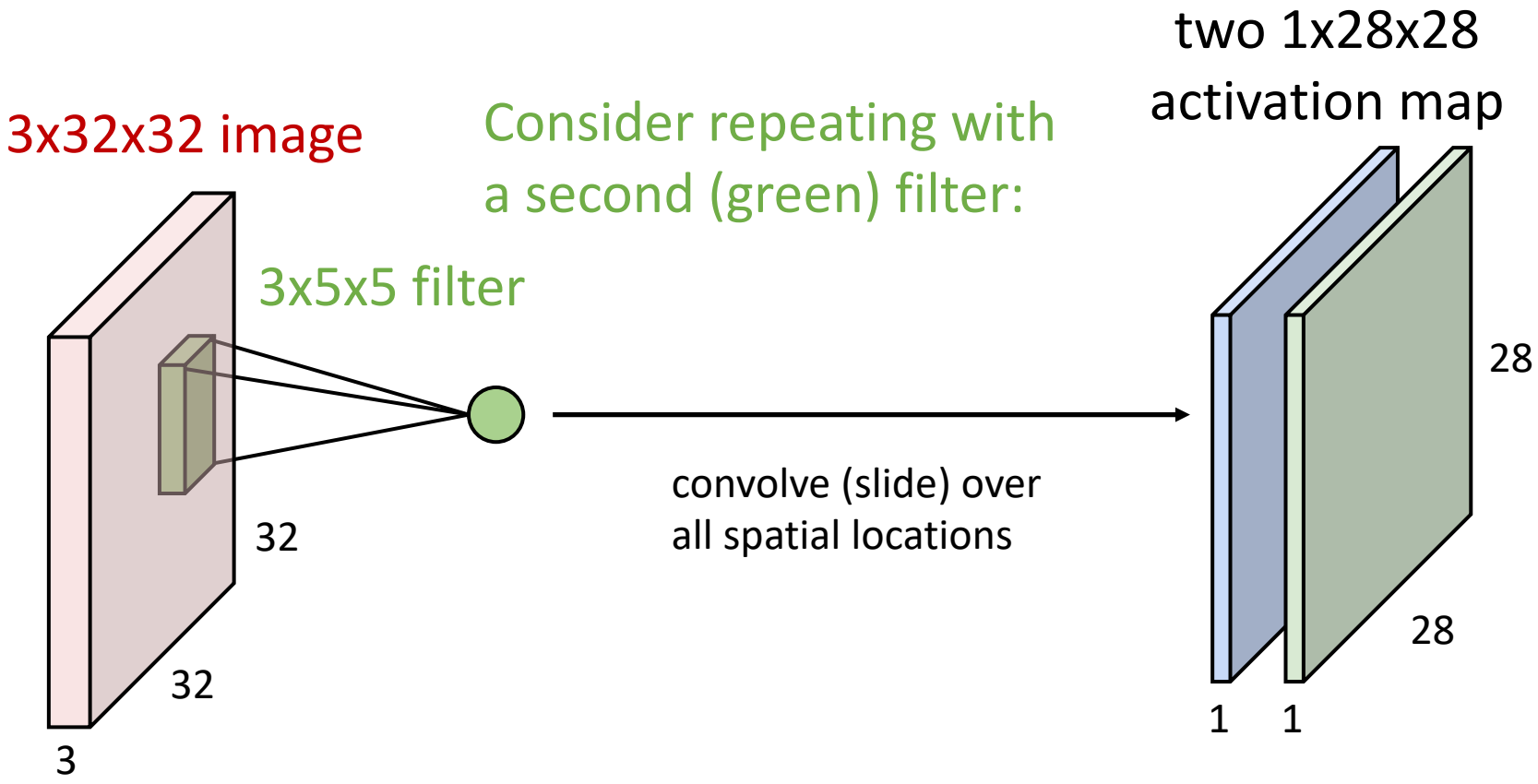
Convolution Layer vs Image Filtering:

>1 input and output channels

Forget about convolution vs cross-correlation

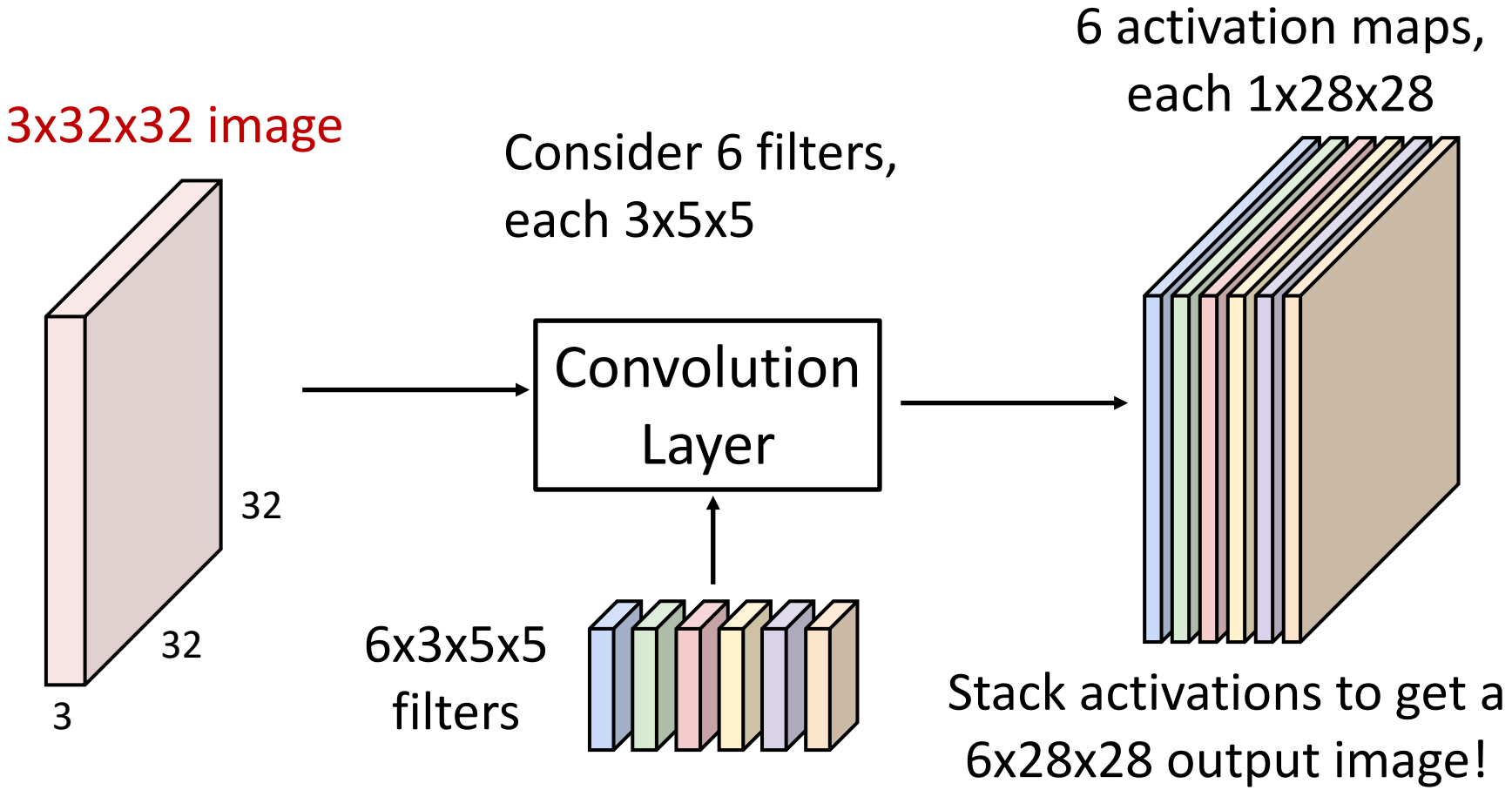
Slide credit: D Fouhey & J Johnson

Convolution Layer



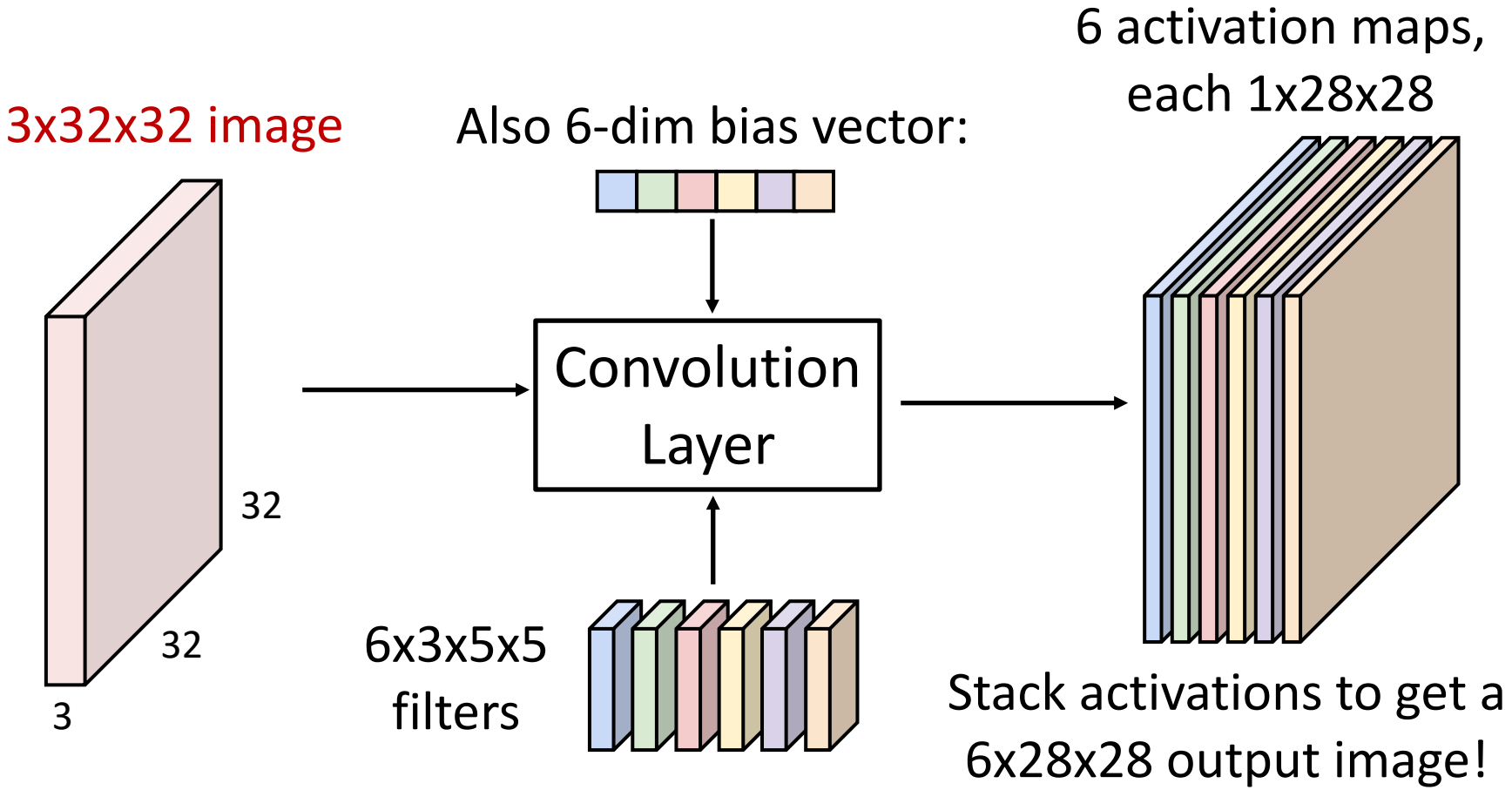
Slide credit: D Fouhey & J Johnson

Convolution Layer



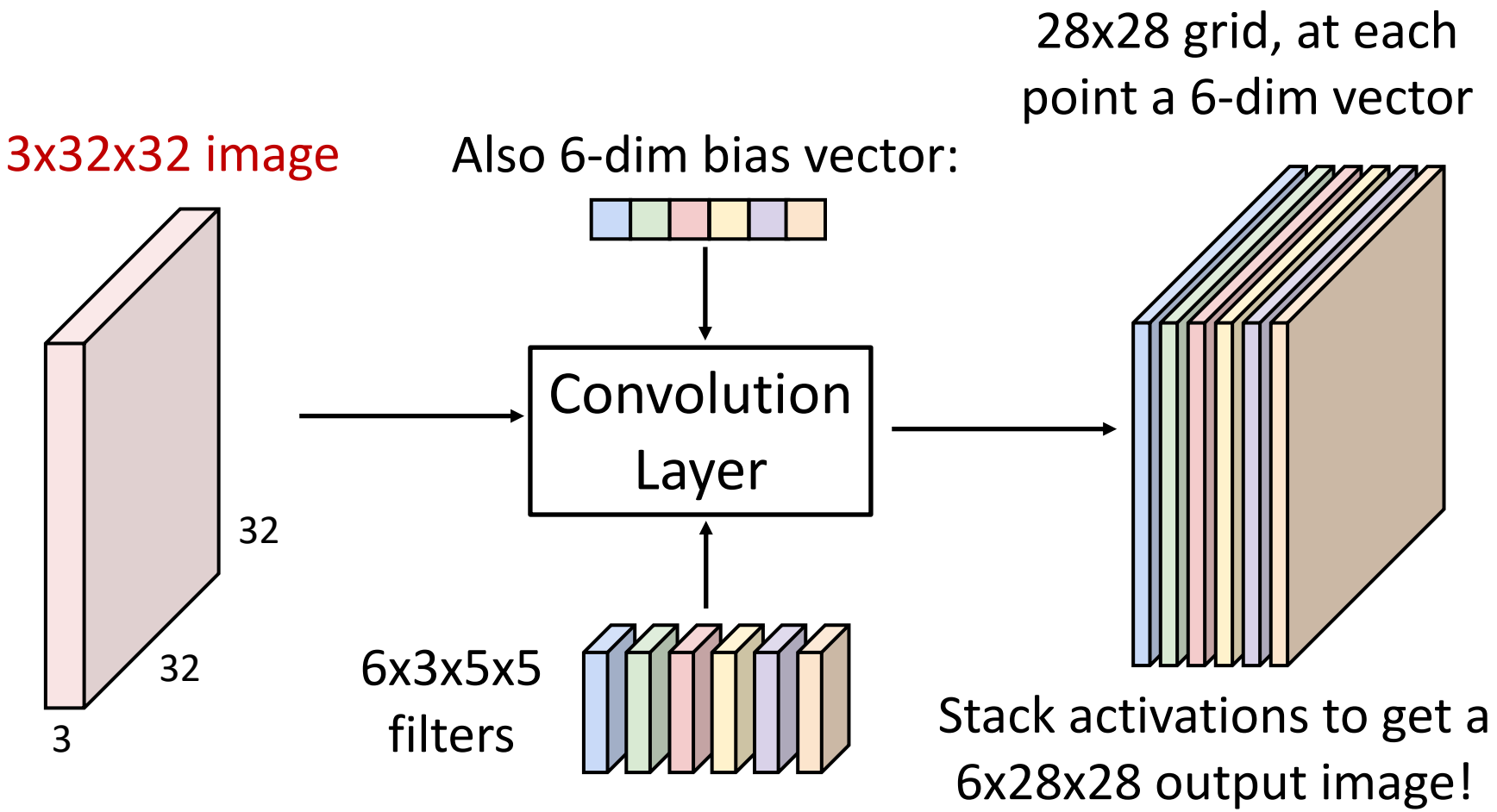
Slide credit: D Fouhey & J Johnson

Convolution Layer



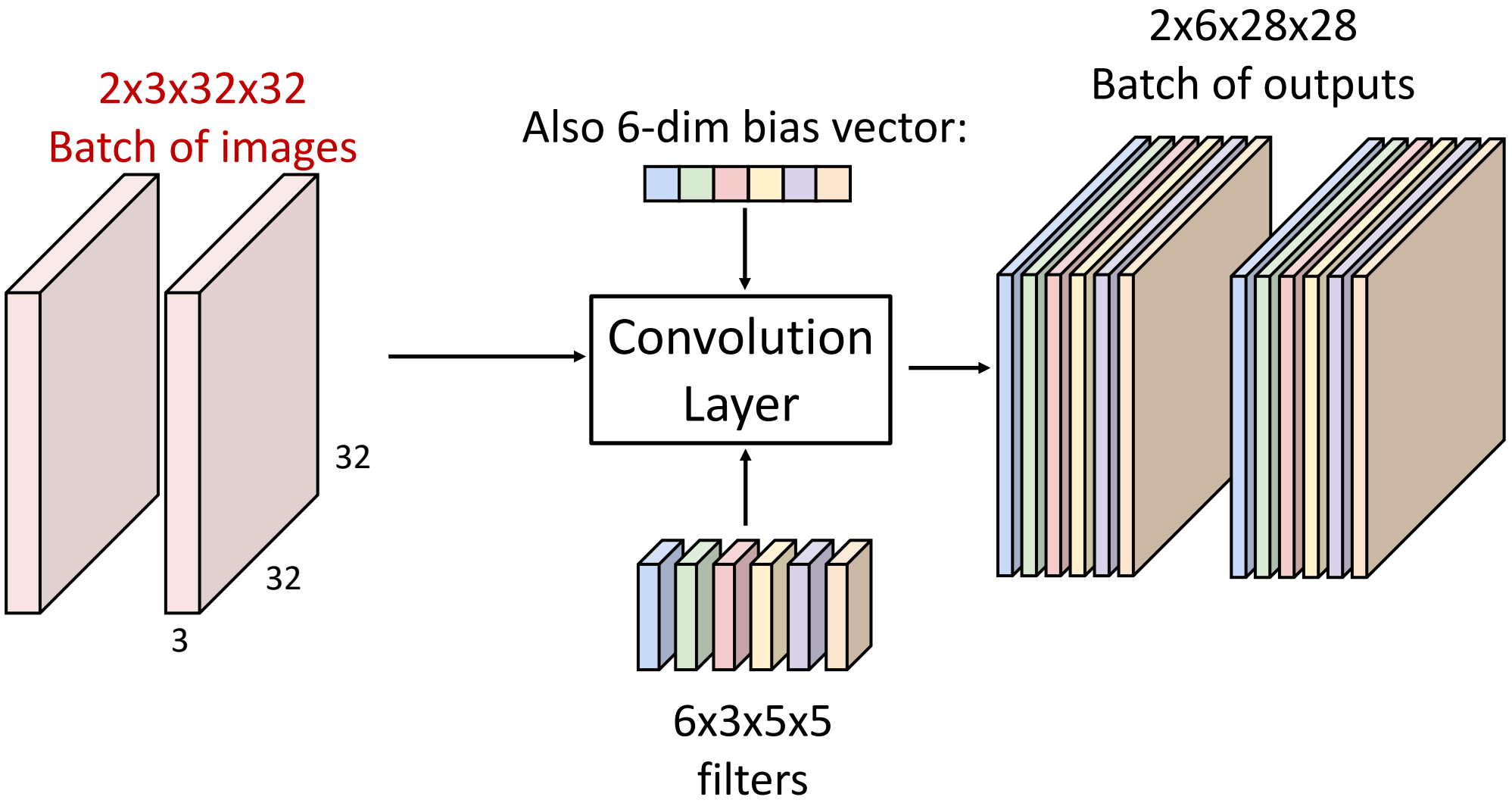
Slide credit: D Fouhey & J Johnson

Convolution Layer



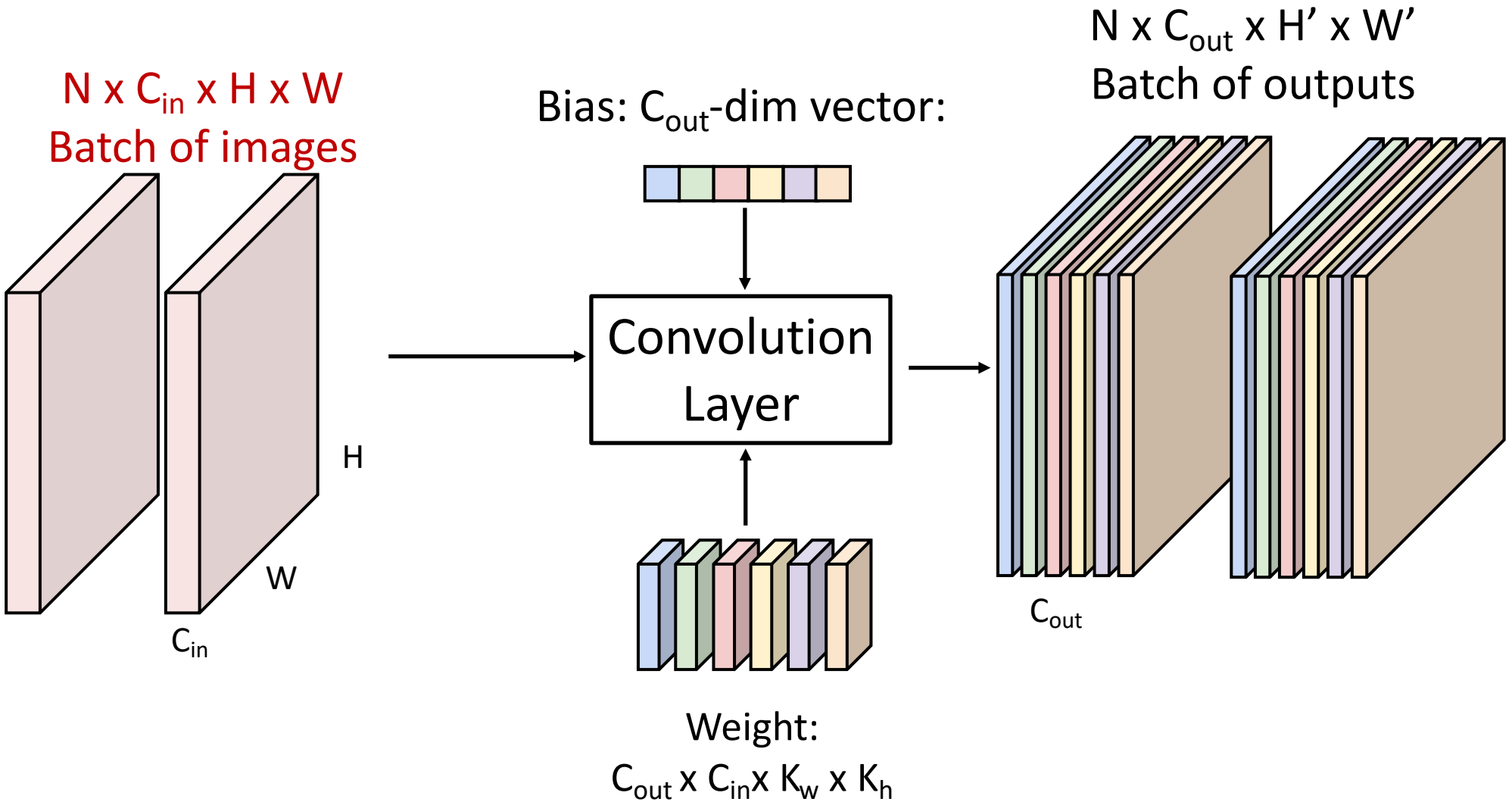
Slide credit: D Fouhey & J Johnson

Convolution Layer



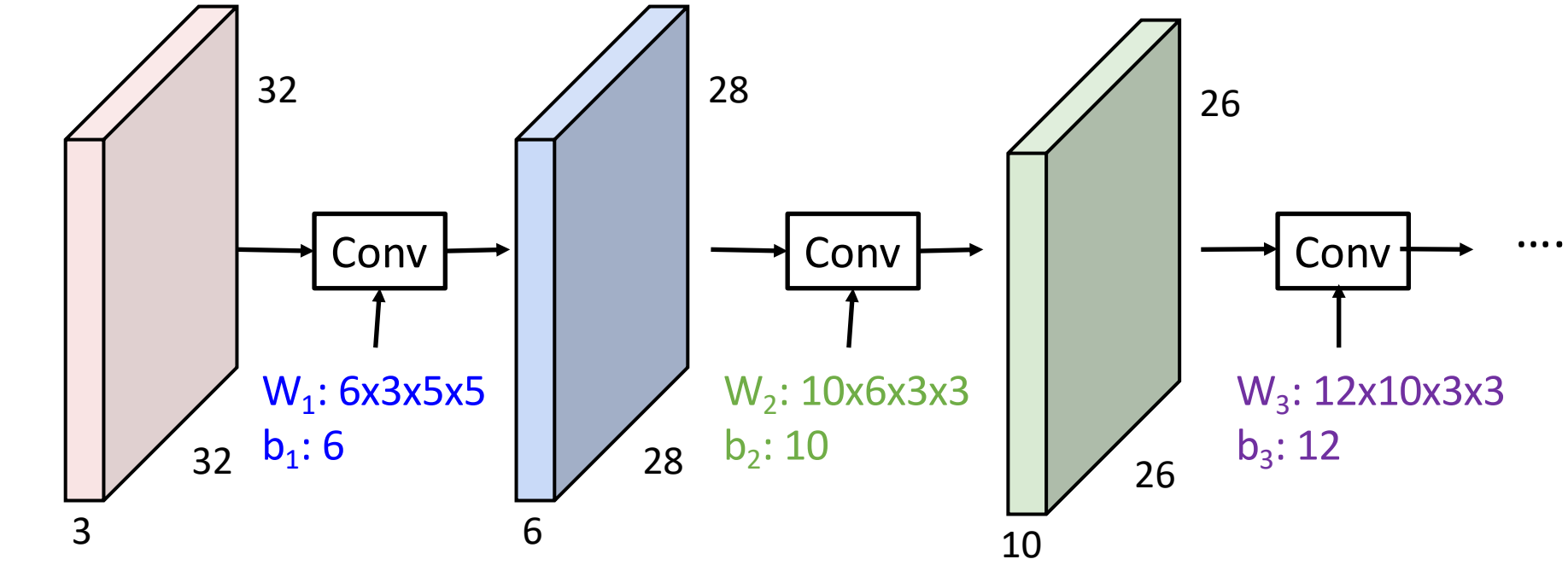
Slide credit: D Fouhey & J Johnson

Convolution Layer



Slide credit: D Fouhey & J Johnson

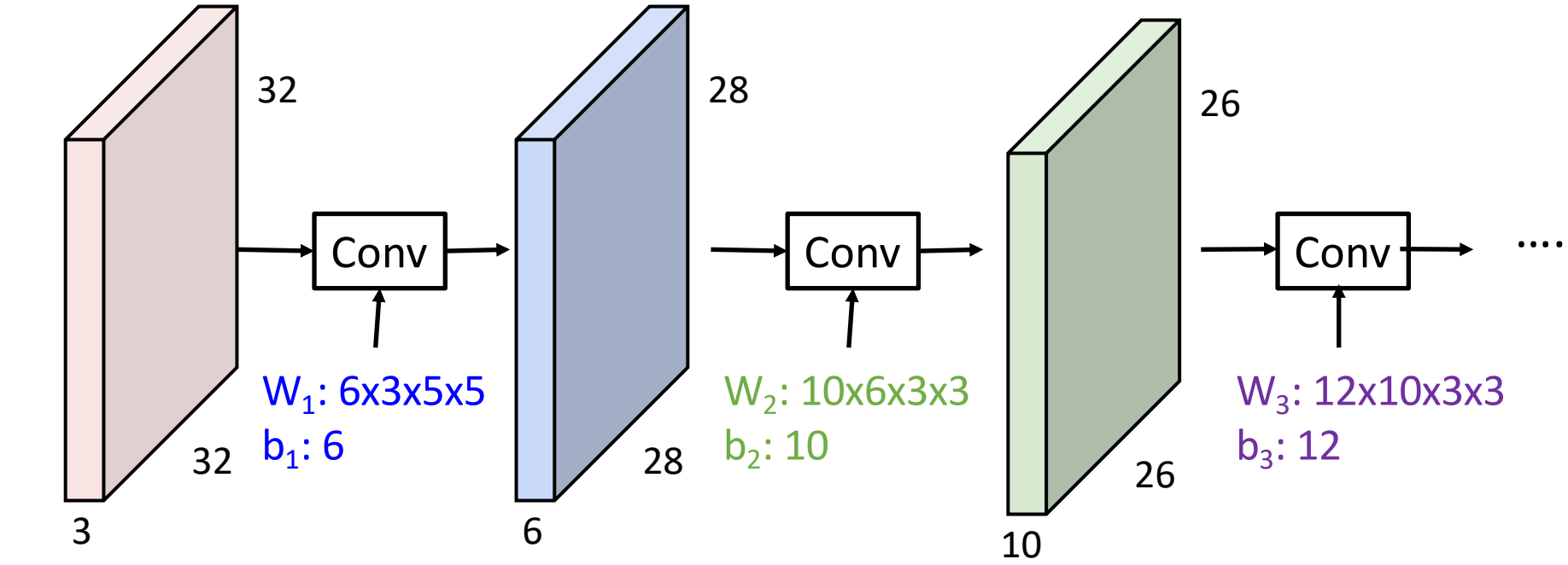
Stacking Convolutions



Slide credit: D Fouhey & J Johnson

Stacking Convolutions

Q: What happens if we stack two convolution layers?



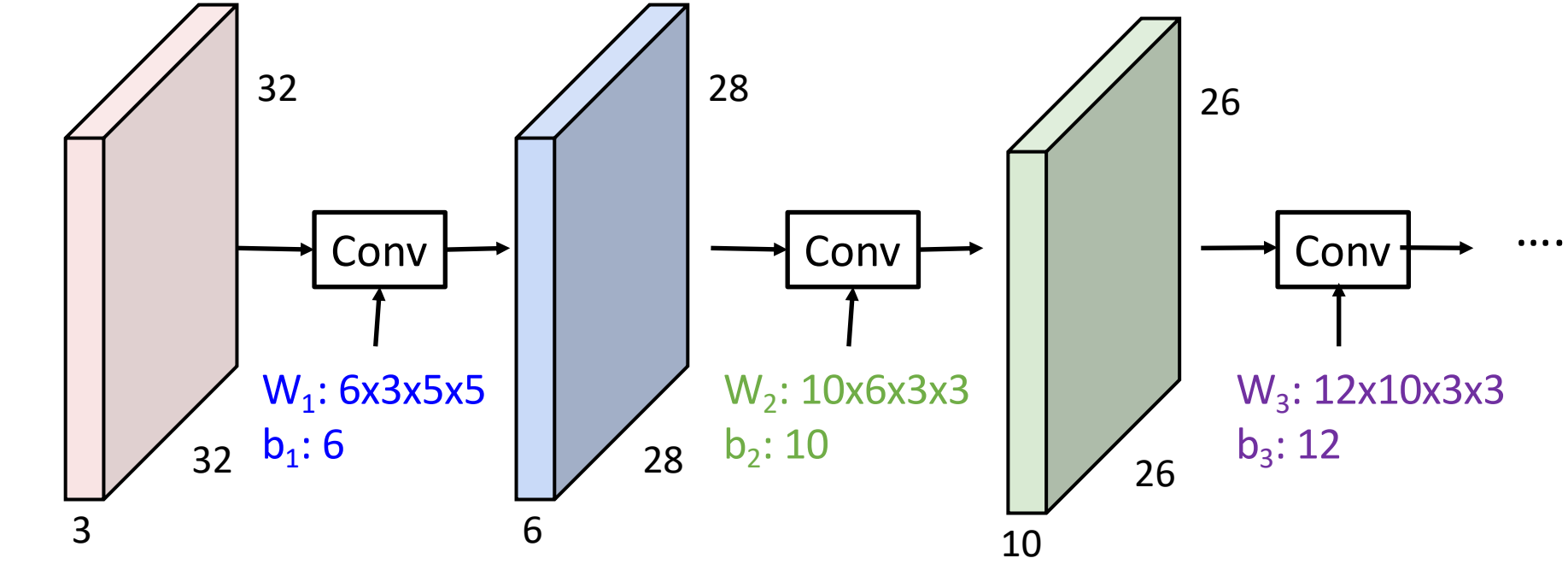
Slide credit: D Fouhey & J Johnson

Stacking Convolutions

(Recall $y=W_2W_1x$ is a linear classifier)

Q: What happens if we stack two convolution layers?

A: It's equivalent to just one convolution layer!



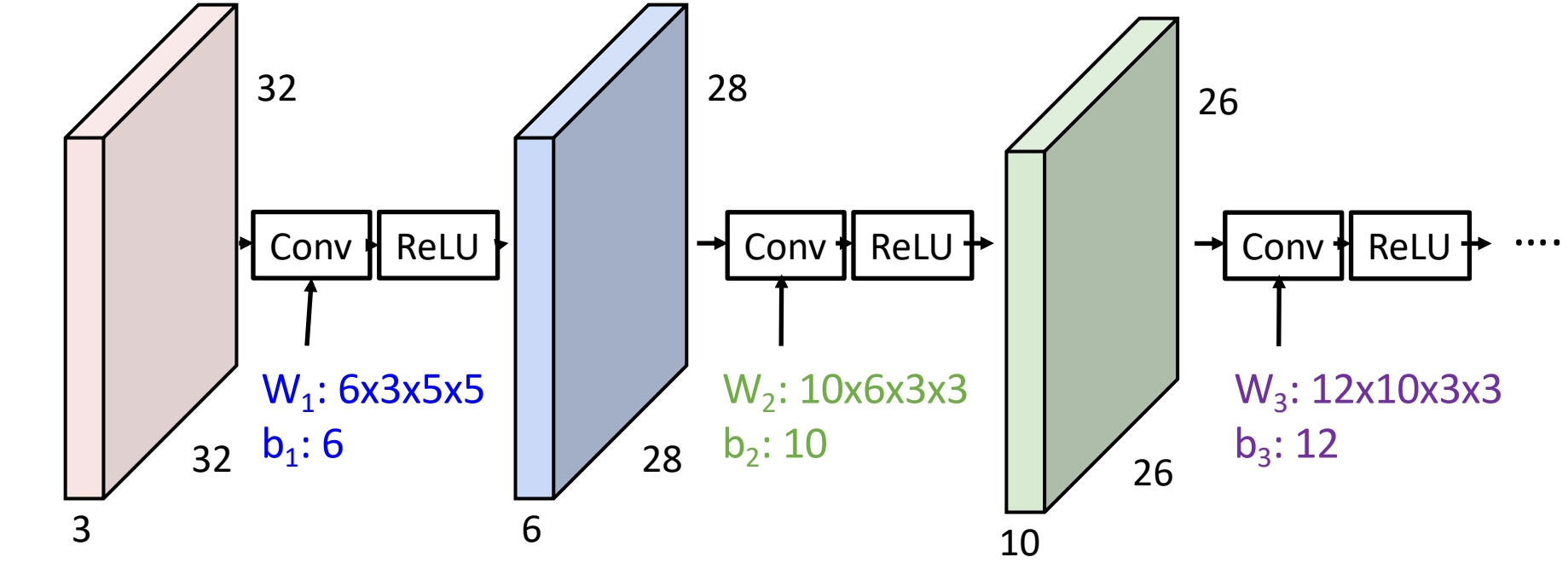
Slide credit: D Fouhey & J Johnson

Stacking Convolutions

(Recall $y=W_2W_1x$ is a linear classifier)

Q: What happens if we stack two convolution layers?

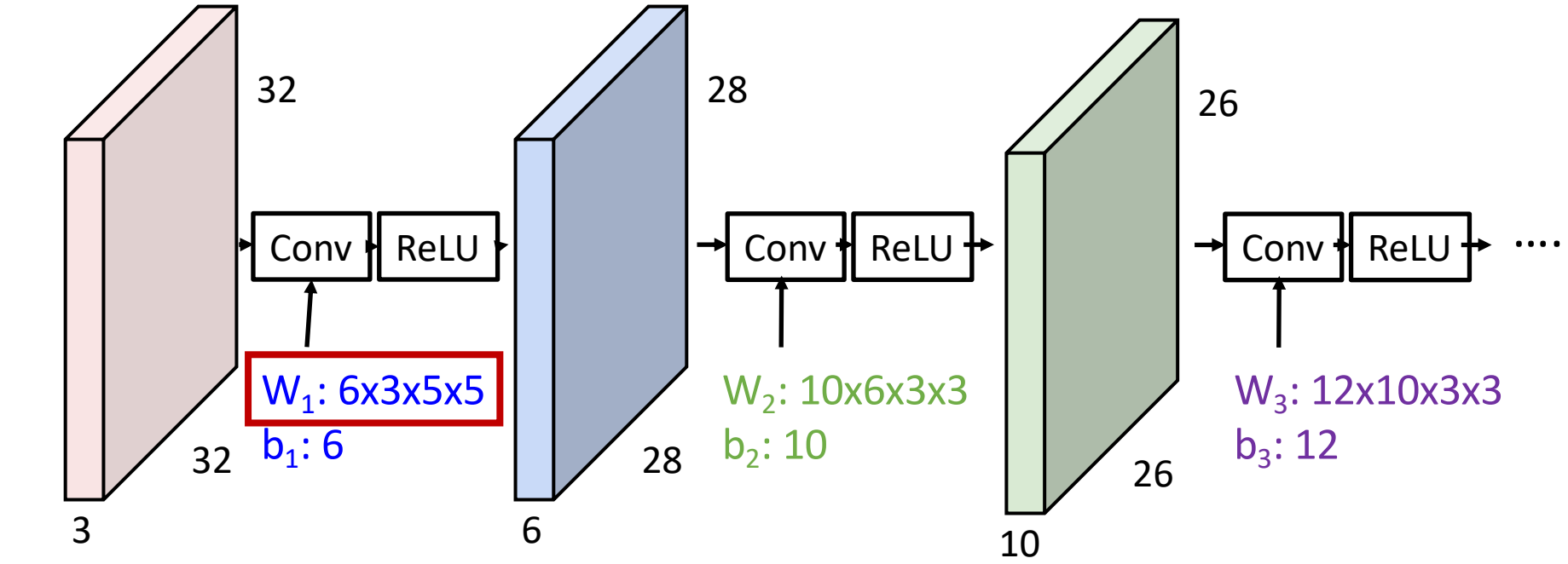
A: It's equivalent to just one convolution layer!



Solution: Add a nonlinearity between each conv layer

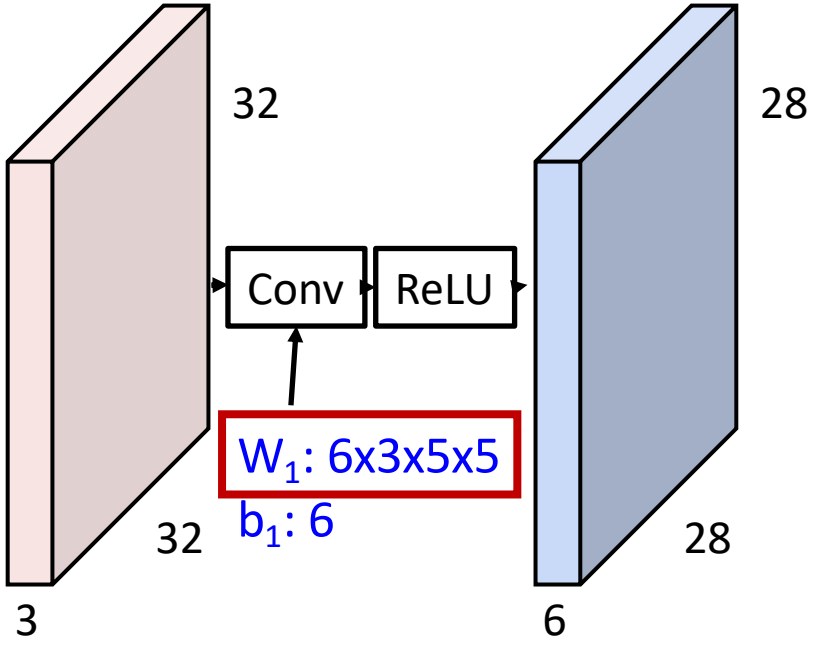
Slide credit: D Fouhey & J Johnson

What do Conv Filters Learn?



Slide credit: D Fouhey & J Johnson

What do Conv Filters Learn?



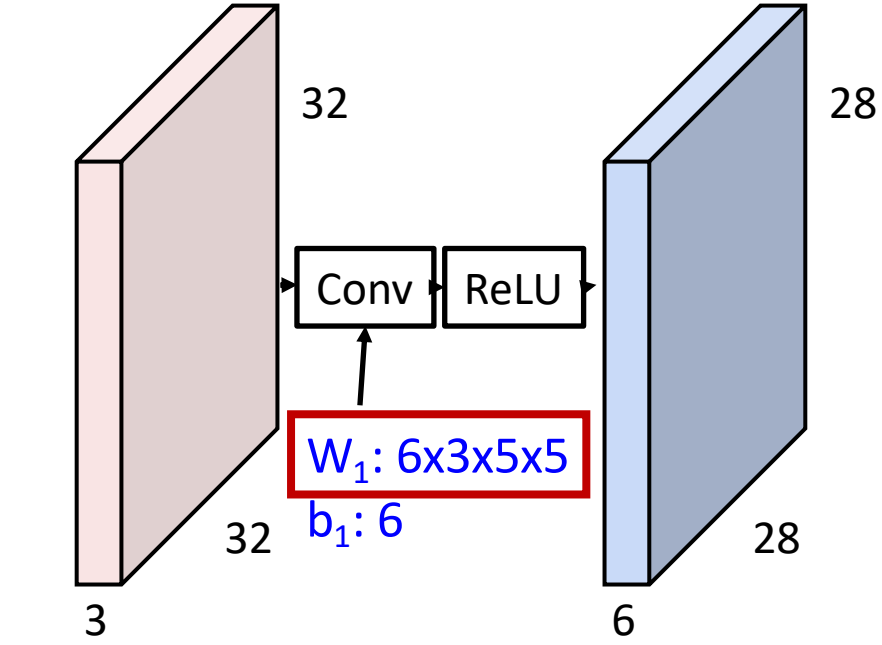
Input:
N x 3 x 32 x 32

First hidden layer:
N x 6 x 28 x 28

Linear classifier:
One template per class

Slide credit: D Fouhey & J Johnson

What do Conv Filters Learn?



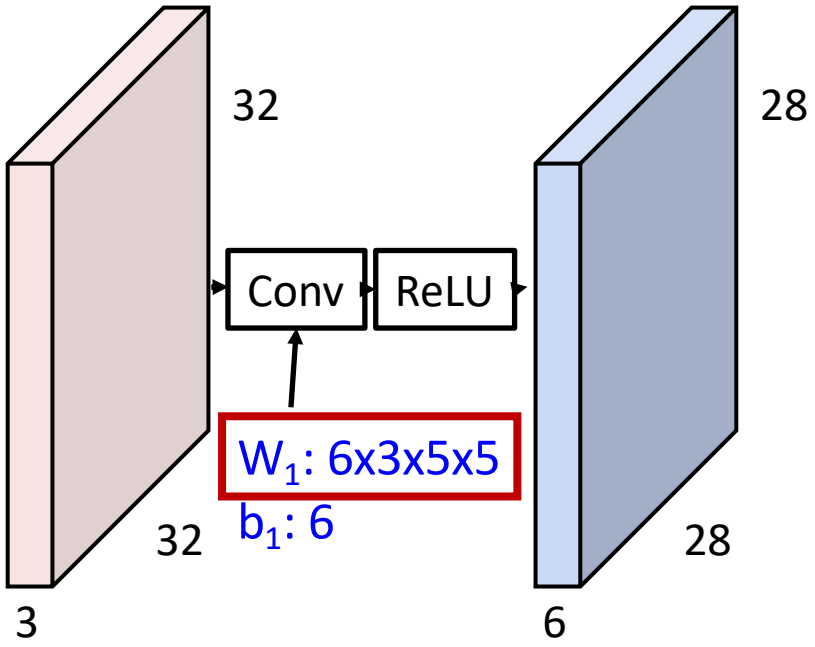
Input:
N x 3 x 32 x 32

First hidden layer:
N x 6 x 28 x 28

MLP: Bank of whole-image templates

Slide credit: D Fouhey & J Johnson

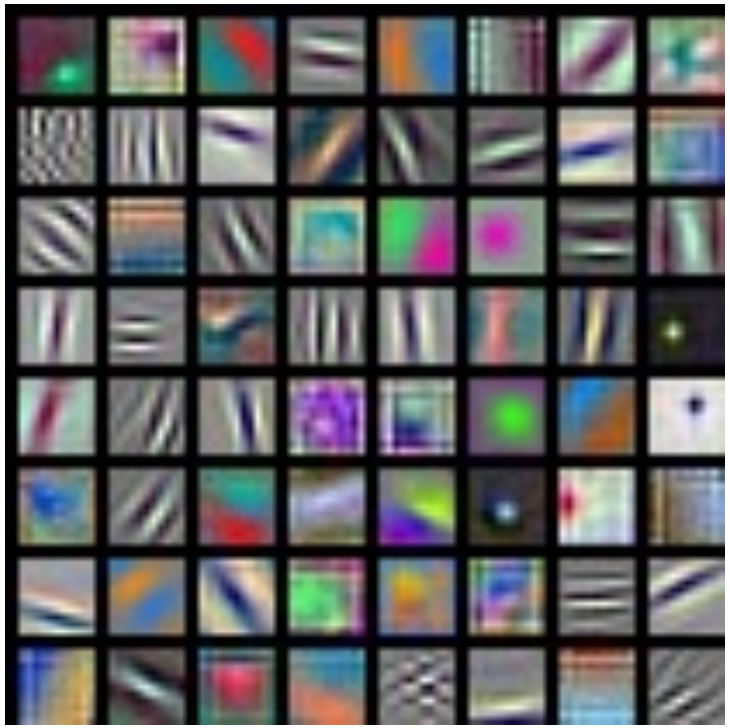
What do Conv Filters Learn?



Input:
N x 3 x 32 x 32

First hidden layer:
N x 6 x 28 x 28

First-layer conv filters: local image templates (Often learns oriented edges, opposing colors)

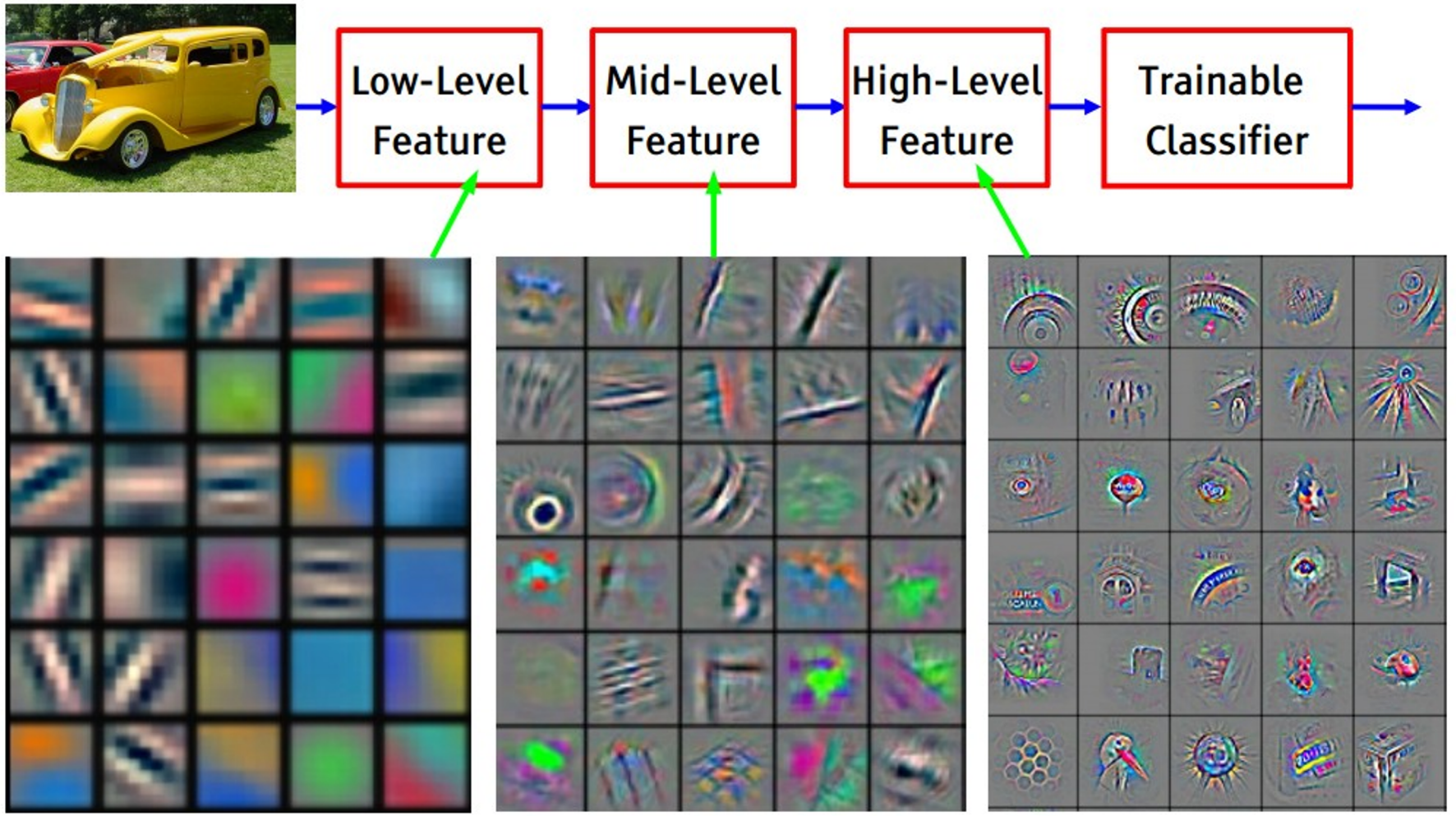


AlexNet: 64 filters, each 3x11x11

Slide credit: D Fouhey & J Johnson

Filters in ConvNets

[From recent Yann LeCun slides]

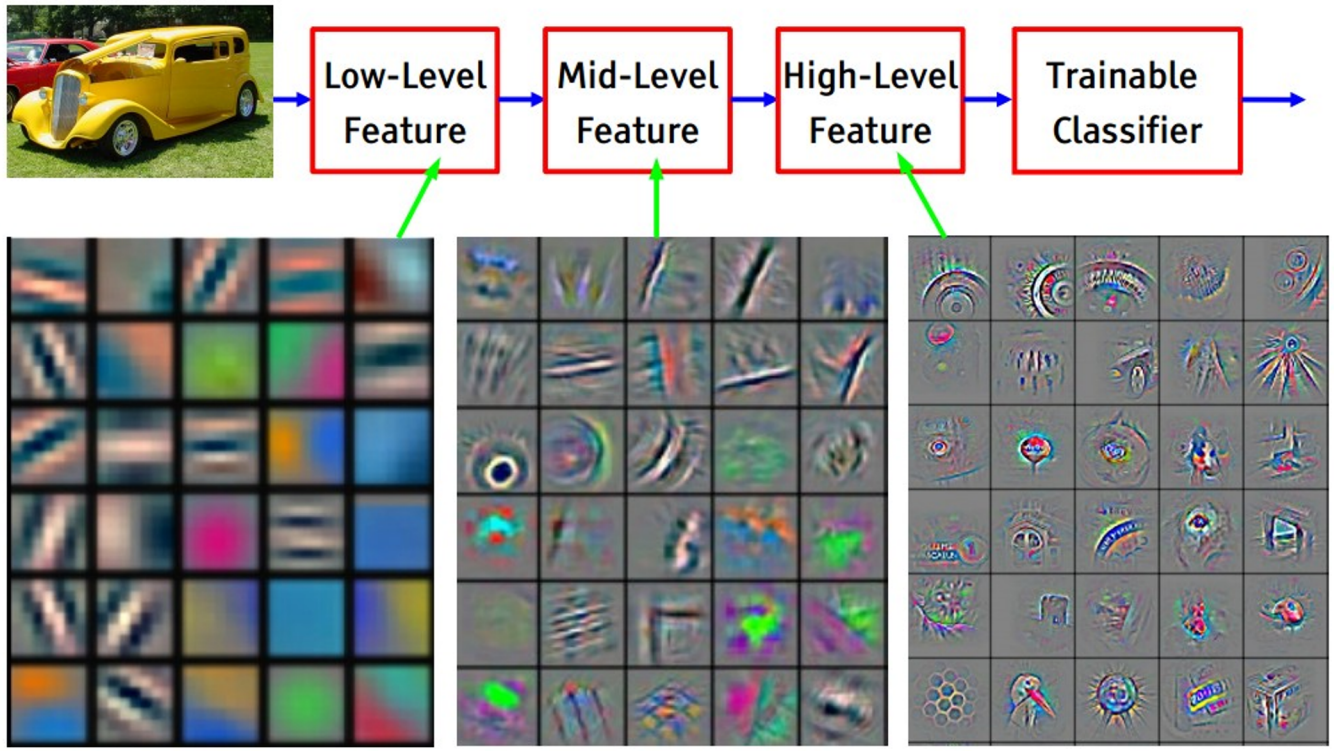


Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

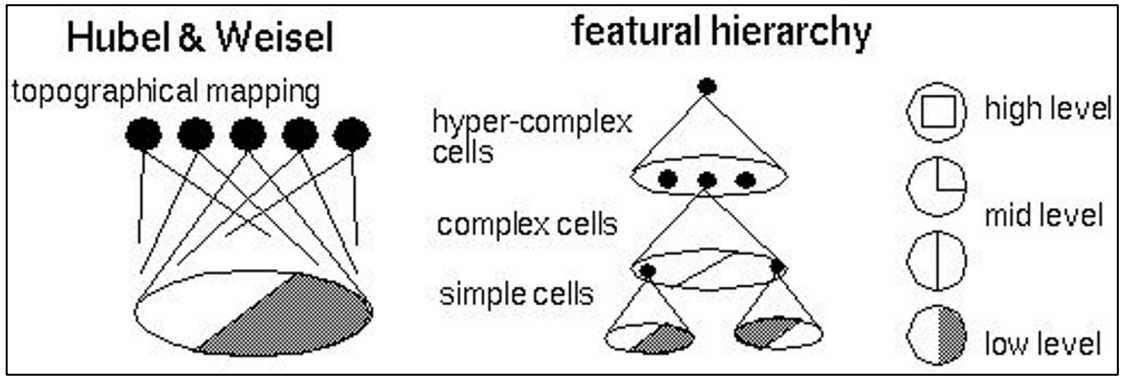
Slide credit: E Learned-Miller

Filters in ConvNets

[From recent Yann LeCun slides]



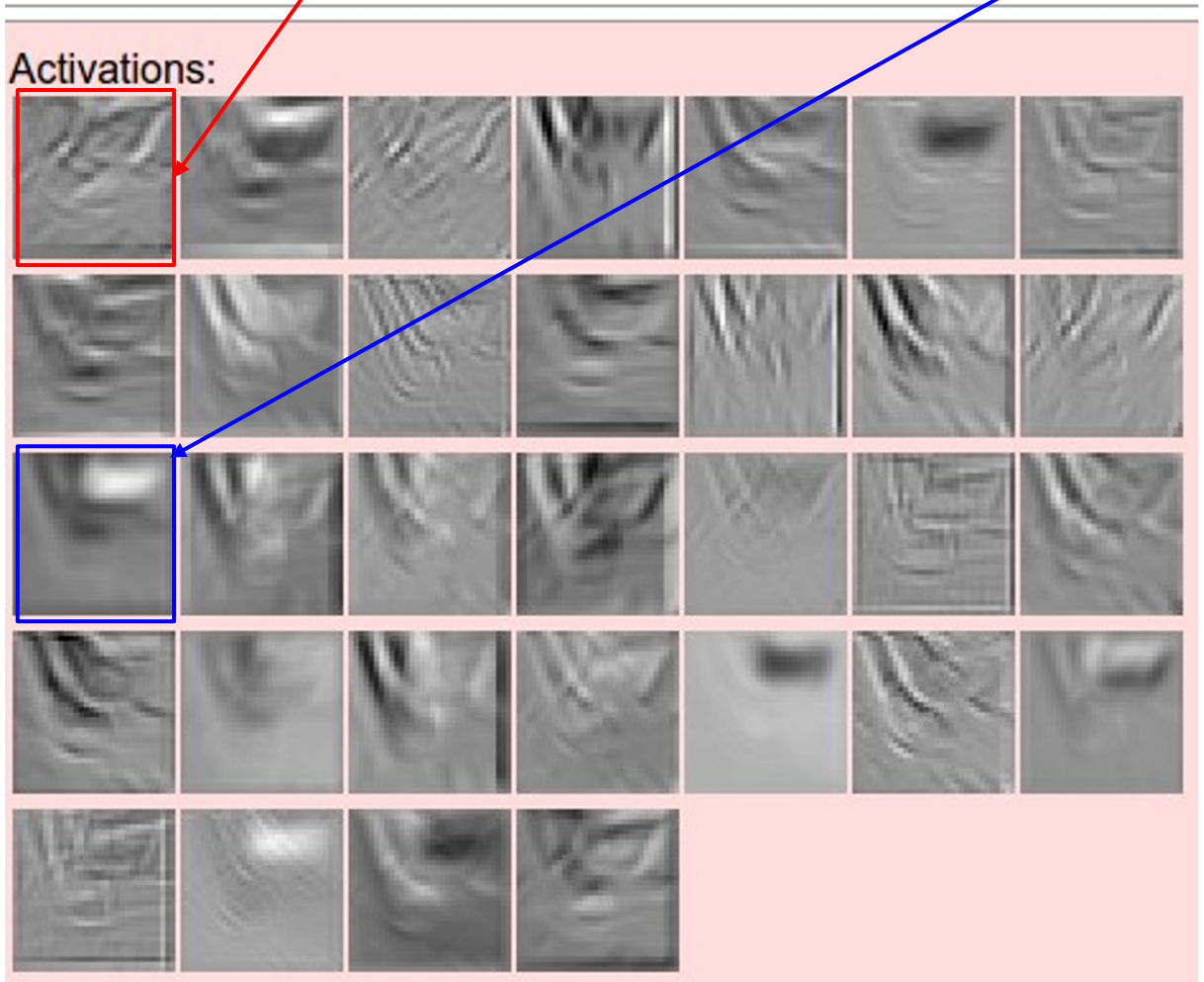
Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]



Slide credit: E Learned-Miller

one filter =>

one activation map



example 5x5 filters

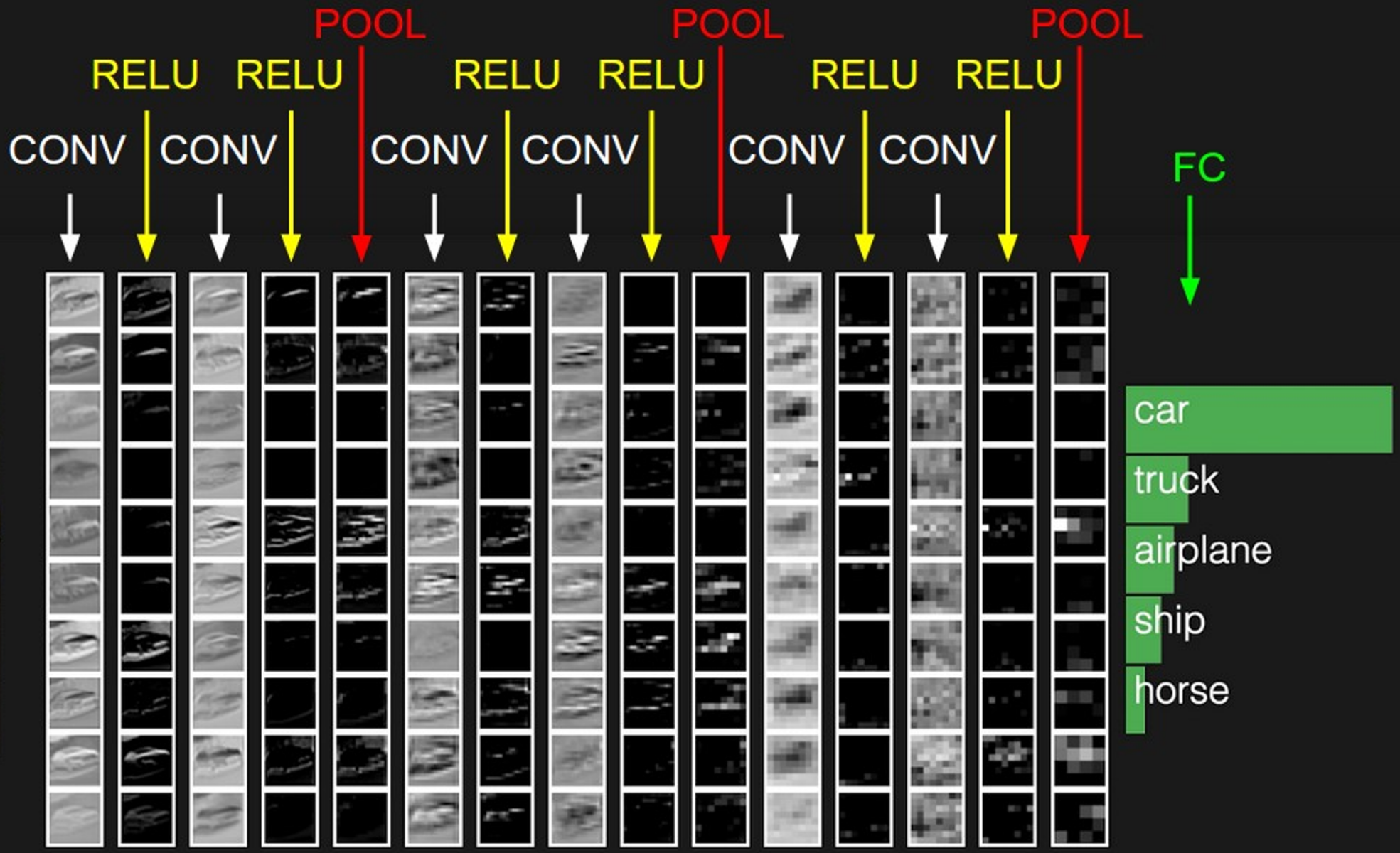
(32 total)

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

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

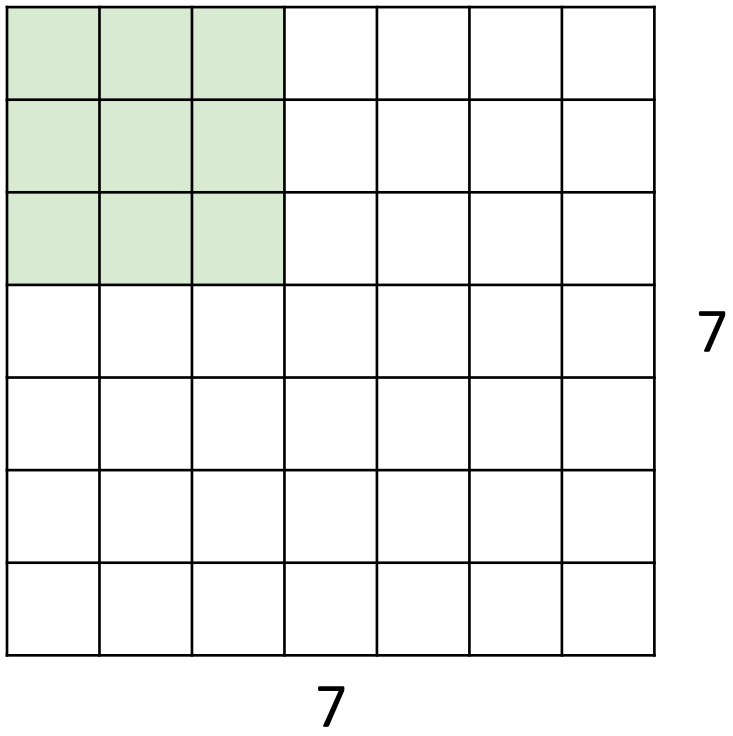
elementwise multiplication and sum of a filter and the signal (image)

Slide credit: E Learned-Miller



Slide credit: E Learned-Miller

Convolution Spatial Dimensions

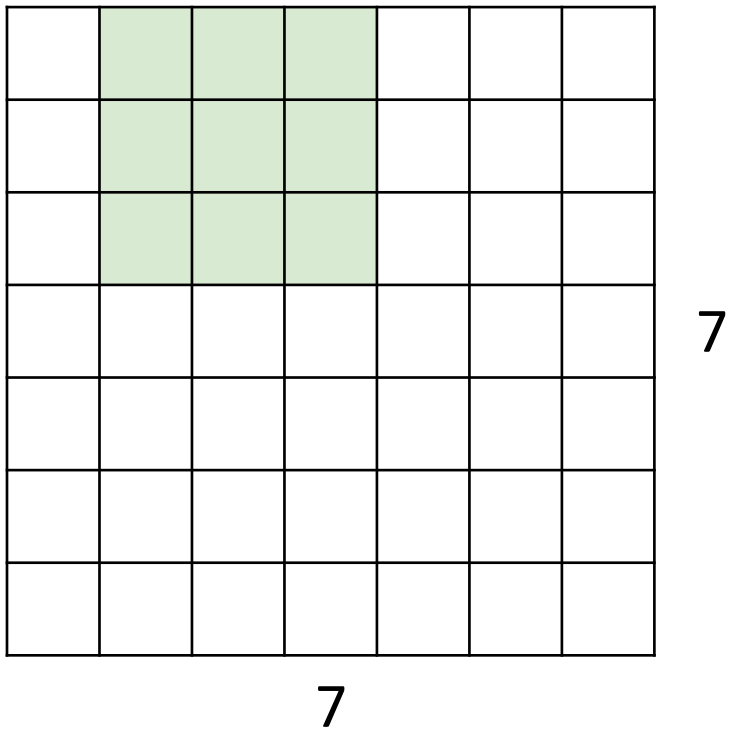


Input: 7x7
Filter: 3x3

Q: How big is output?

Slide credit: D Fouhey & J Johnson

Convolution Spatial Dimensions

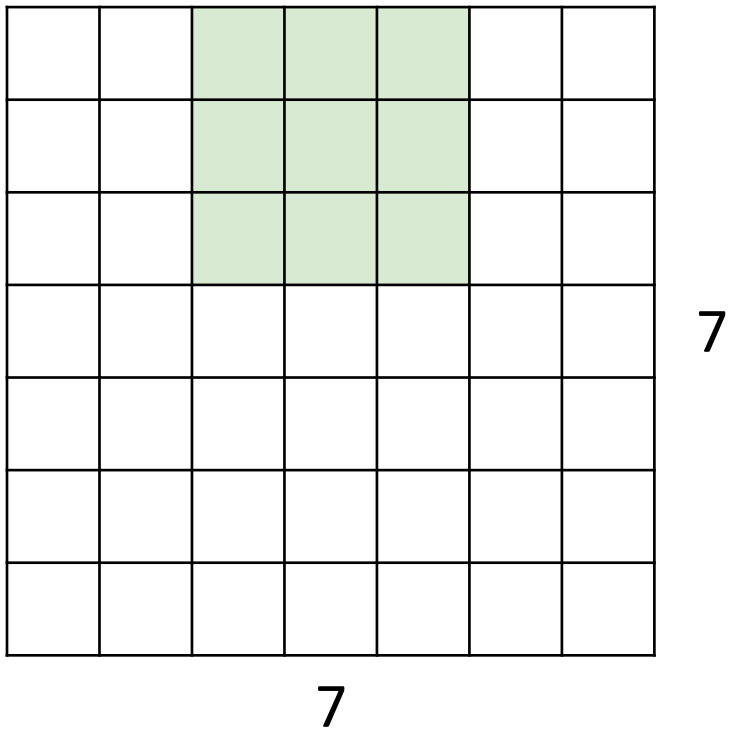


Input: 7x7
Filter: 3x3

Q: How big is output?

Slide credit: D Fouhey & J Johnson

Convolution Spatial Dimensions

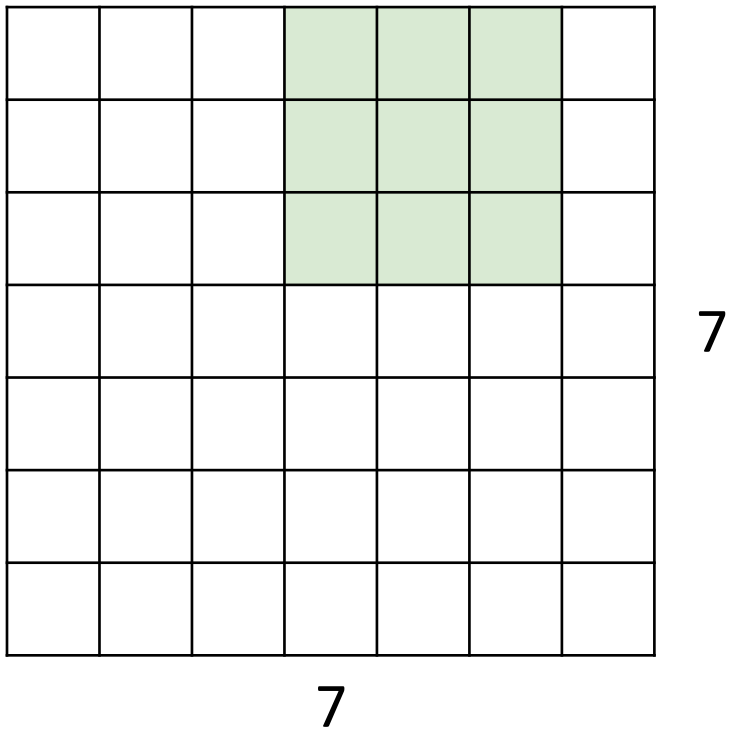


Input: 7x7
Filter: 3x3

Q: How big is output?

Slide credit: D Fouhey & J Johnson

Convolution Spatial Dimensions

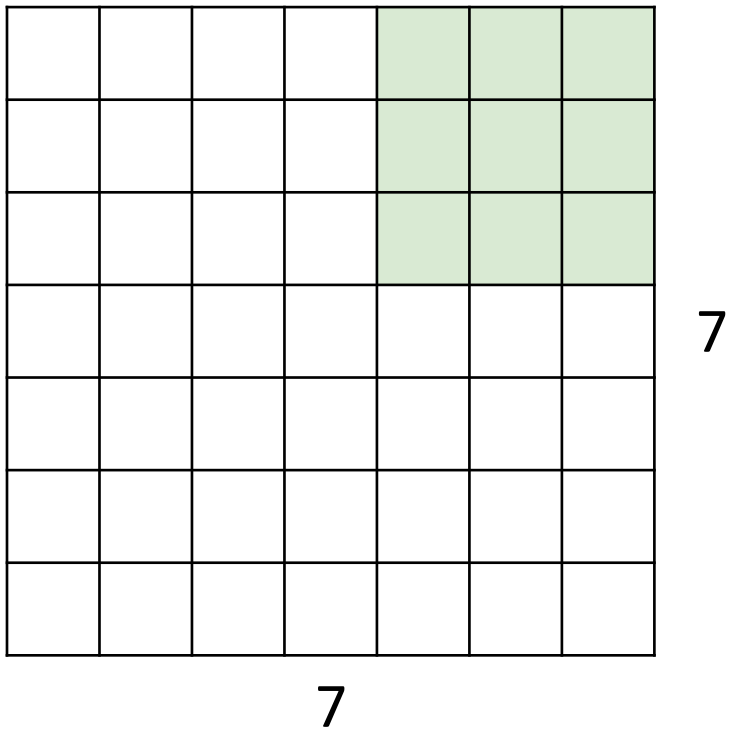


Input: 7x7
Filter: 3x3

Q: How big is output?

Slide credit: D Fouhey & J Johnson

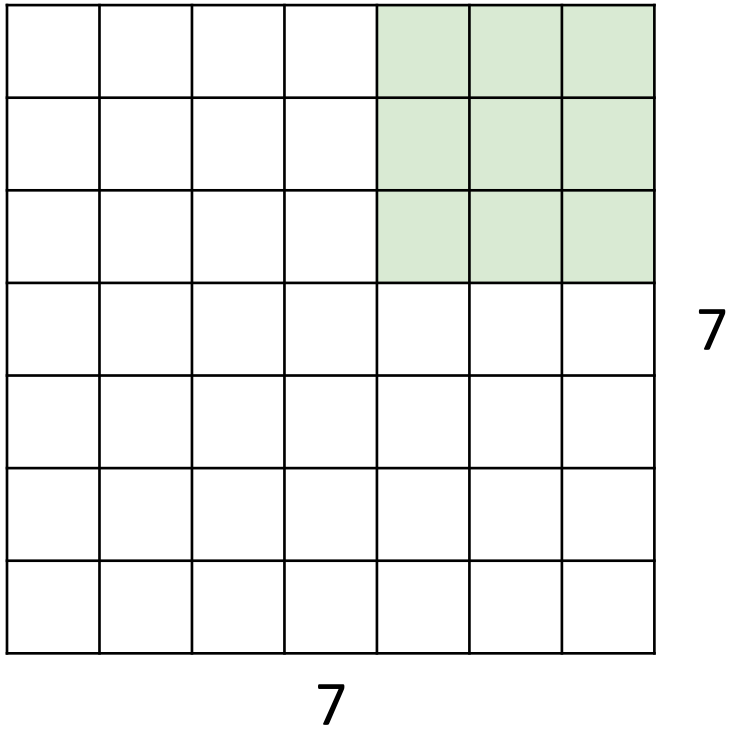
Convolution Spatial Dimensions



Input: 7x7
Filter: 3x3
Output: 5x5

Slide credit: D Fouhey & J Johnson

Convolution Spatial Dimensions



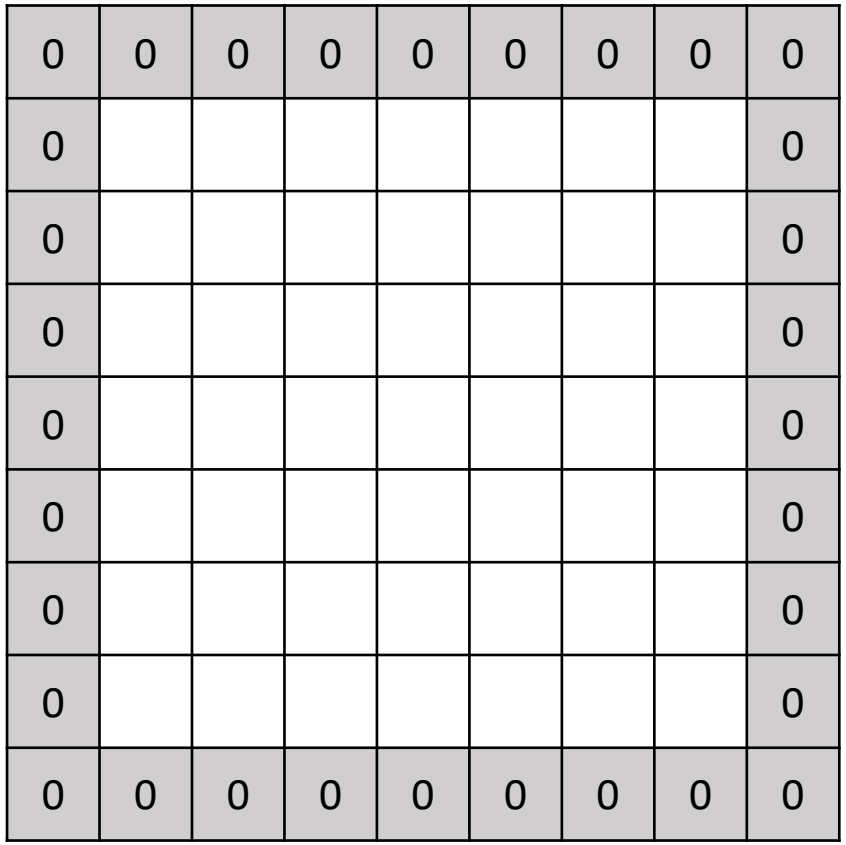
Input: 7x7
Filter: 3x3
Output: 5x5

In general:
Input: W
Filter: K
Output: W – K + 1

Problem:
Feature maps "shrink" with each layer!

Slide credit: D Fouhey & J Johnson

Convolution Spatial Dimensions



Input: 7x7
Filter: 3x3
Output: 5x5

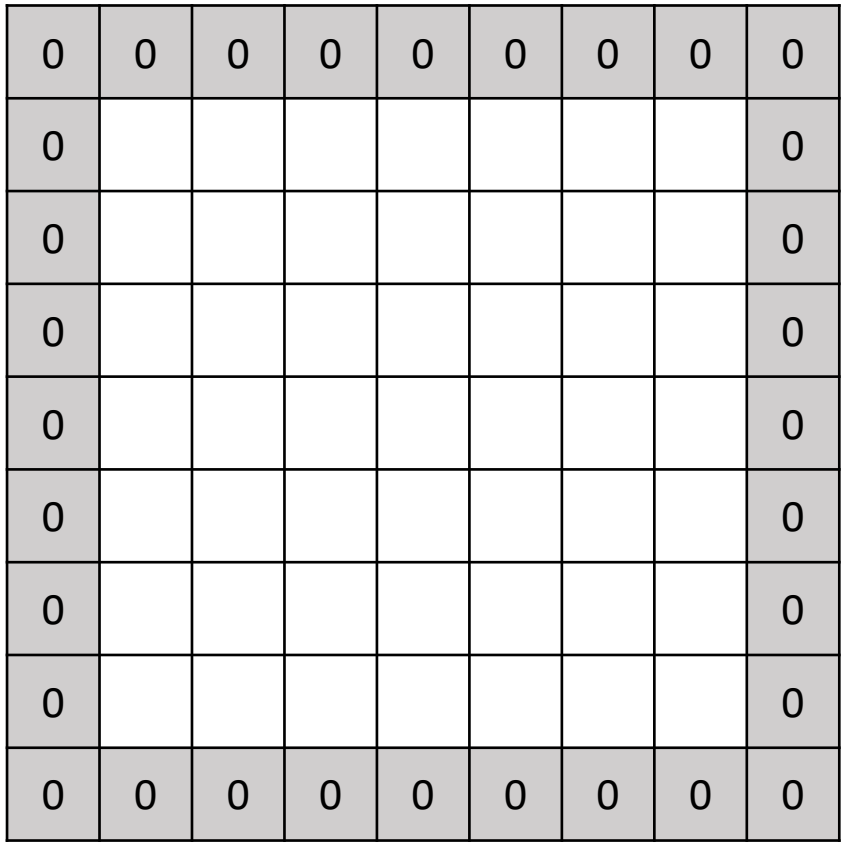
In general:
Input: W
Filter: K
Padding: P

Problem:
Feature maps "shrink" with each layer!

Solution: **padding**
Add zeros around the input

Slide credit: D Fouhey & J Johnson

Convolution Spatial Dimensions



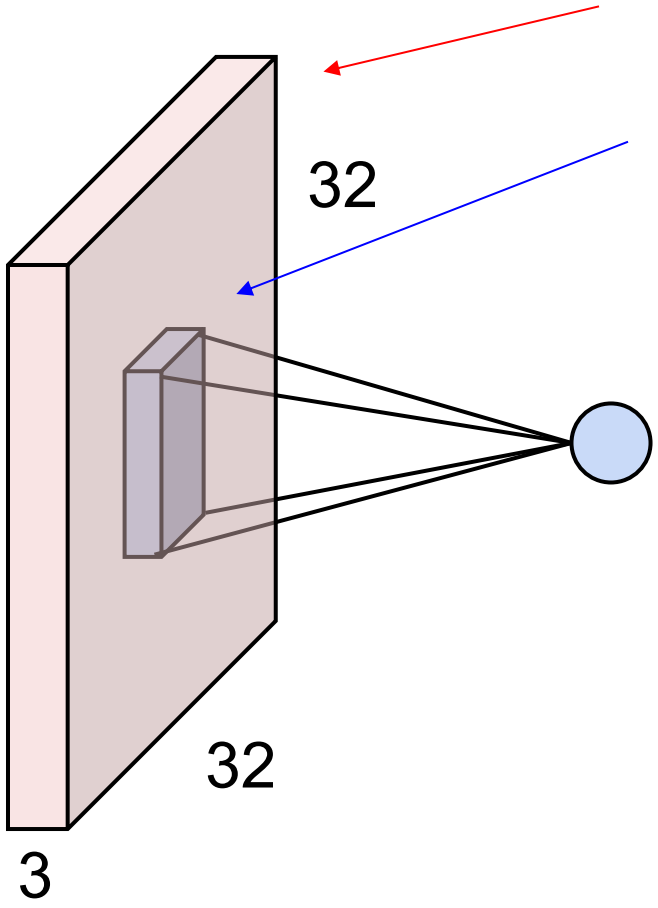
Input: 7x7
Filter: 3x3
Output: 5x5

In general:
Input: W
Filter: K
Padding: P
Output: W – K + 1 + 2P

Very common: “same padding”
Set P = (K – 1) / 2
Then output size = input size

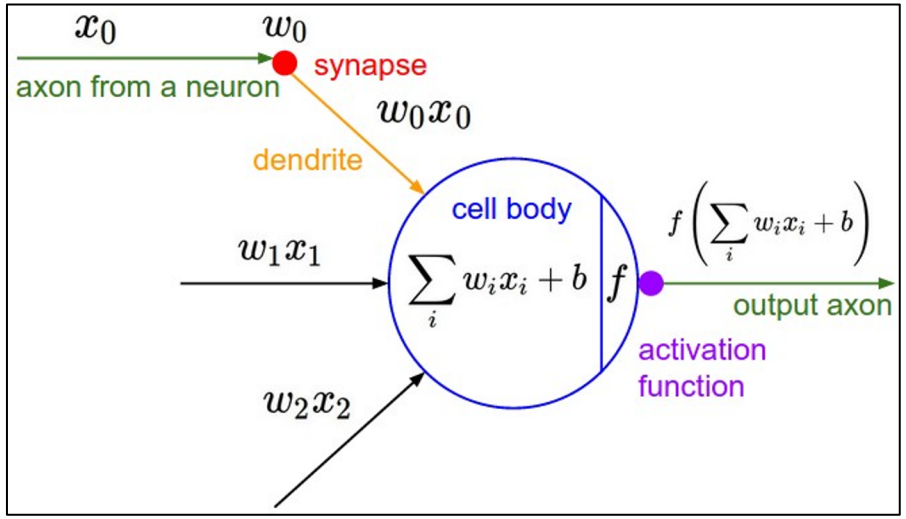
Slide credit: D Fouhey & J Johnson

The brain/neuron view of CONV Layer



32x32x3 image

5x5x3 filter

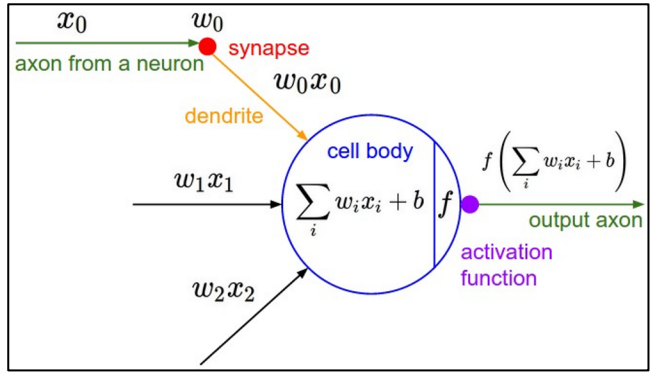


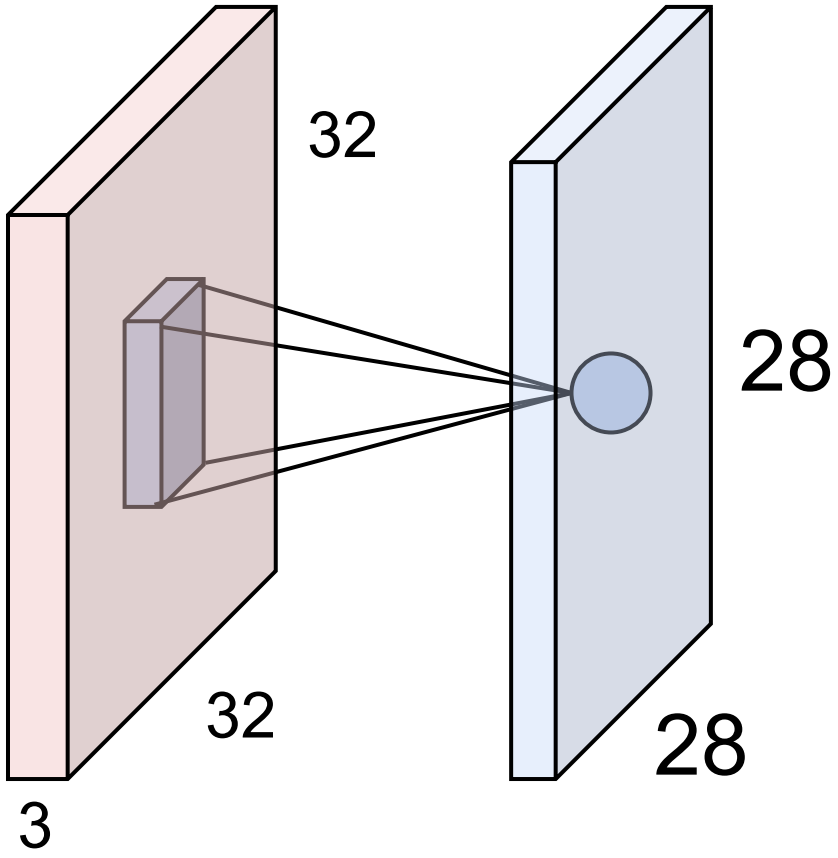
It's just a neuron with local connectivity...

1 number:
the result of taking a dot product between the filter and this part of the image
(i.e. 5*5*3 = 75-dimensional dot product)

Slide credit: E Learned-Miller

The brain/neuron view of CONV Layer





An activation map is a 28x28 sheet of neuron outputs:

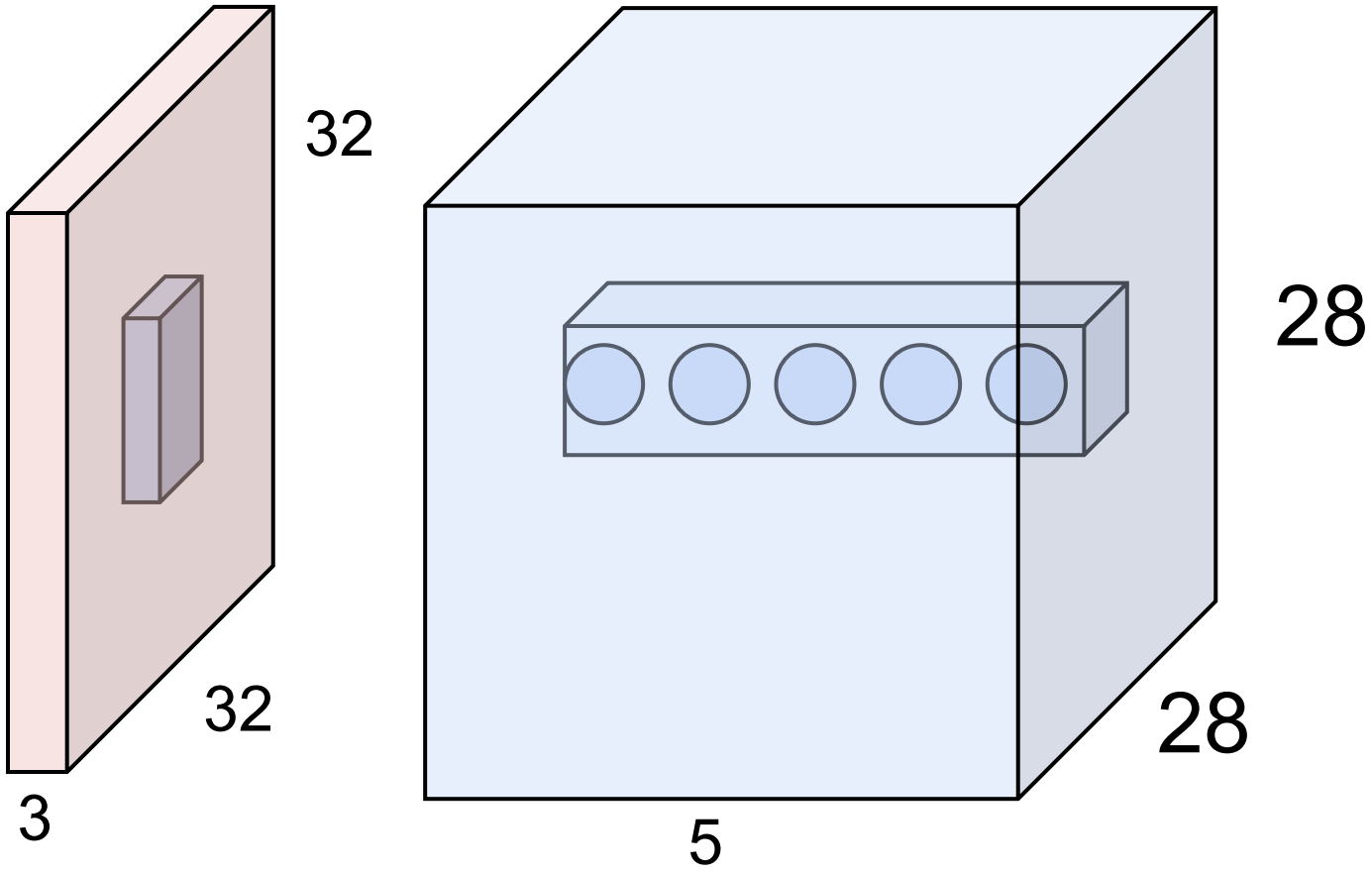
1. Each is connected to a small region in the input
2. All of them share parameters

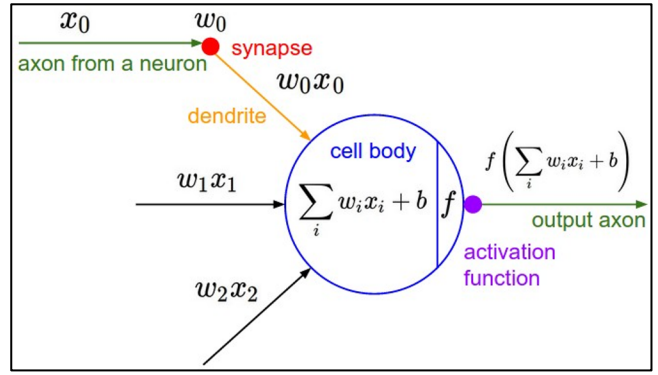
A major advantage of CONV layer!

"5x5 filter" -> "5x5 receptive field for each neuron"

Slide credit: E Learned-Miller

The brain/neuron view of CONV Layer





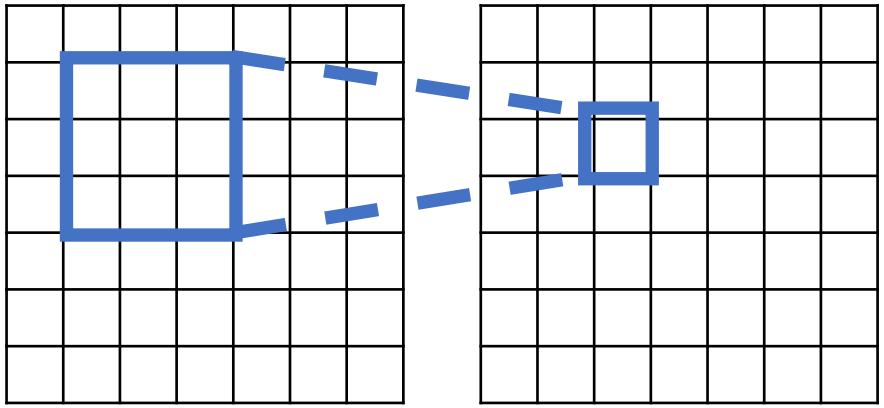
E.g. with 5 filters,
CONV layer consists of neurons arranged in a 3D grid (28x28x5)

There will be 5 different neurons all looking at the same region in the input volume

Slide credit: E Learned-Miller

Receptive Fields

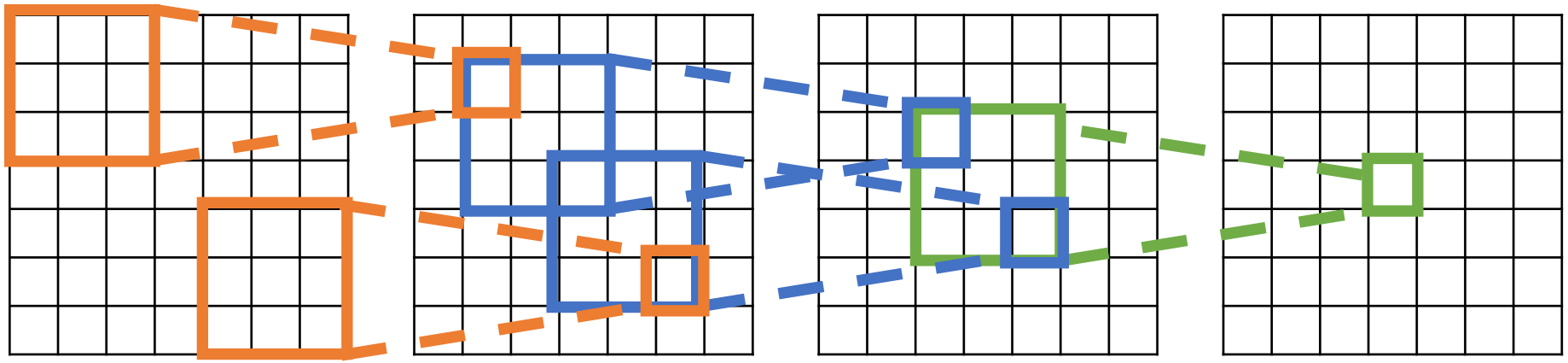
For convolution with kernel size K, each element in the output depends on a K x K **receptive field** in the input



Slide credit: D Fouhey & J Johnson

Receptive Fields

Each successive convolution adds K – 1 to the receptive field size
With L layers the receptive field size is 1 + L * (K – 1)

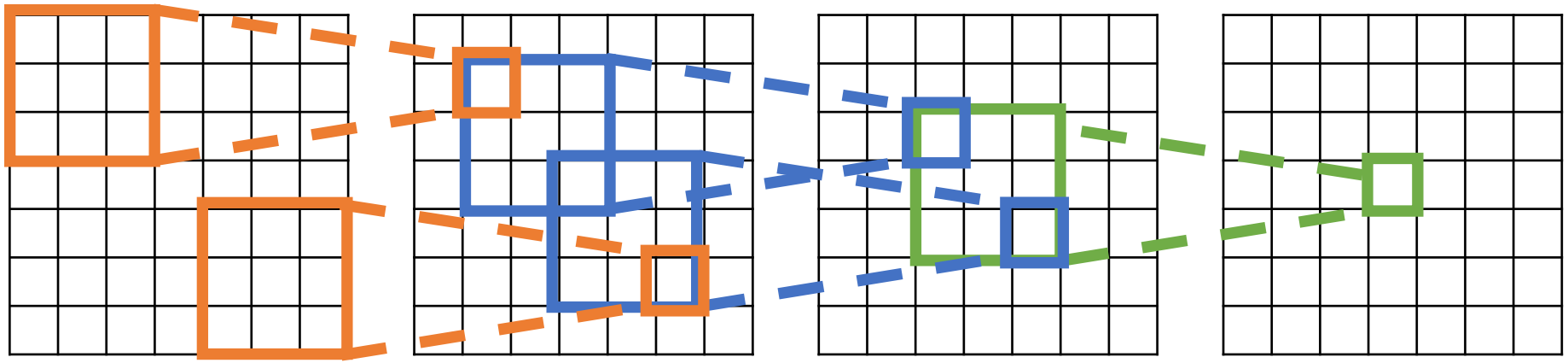


Careful – ”receptive field in the input”
vs “receptive field in the previous layer”
Hopefully clear from context!

Slide credit: D Fouhey & J Johnson

Receptive Fields

Each successive convolution adds K – 1 to the receptive field size
With L layers the receptive field size is 1 + L * (K – 1)

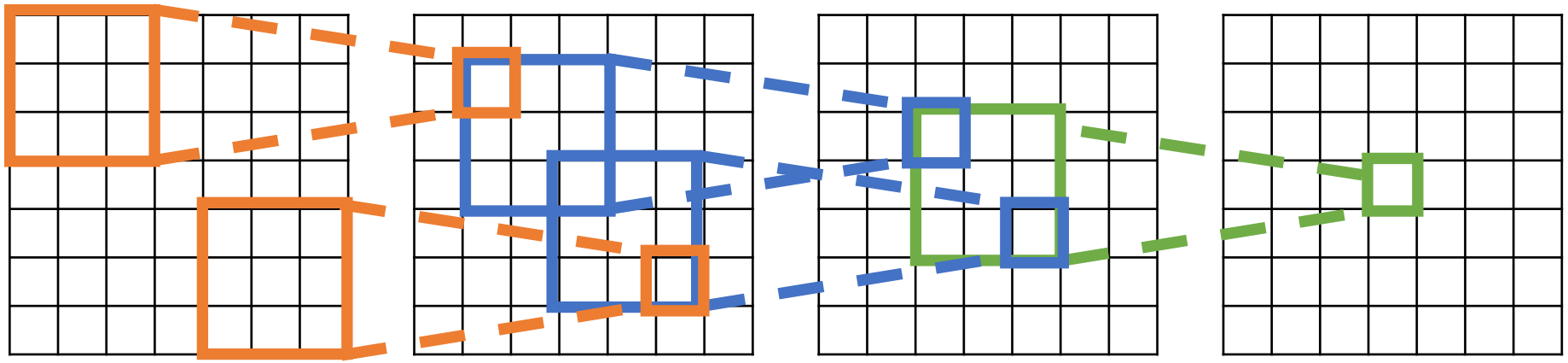


Problem: For large images we need many layers for each output to “see” the whole image image

Slide credit: D Fouhey & J Johnson

Receptive Fields

Each successive convolution adds K – 1 to the receptive field size
With L layers the receptive field size is 1 + L * (K – 1)

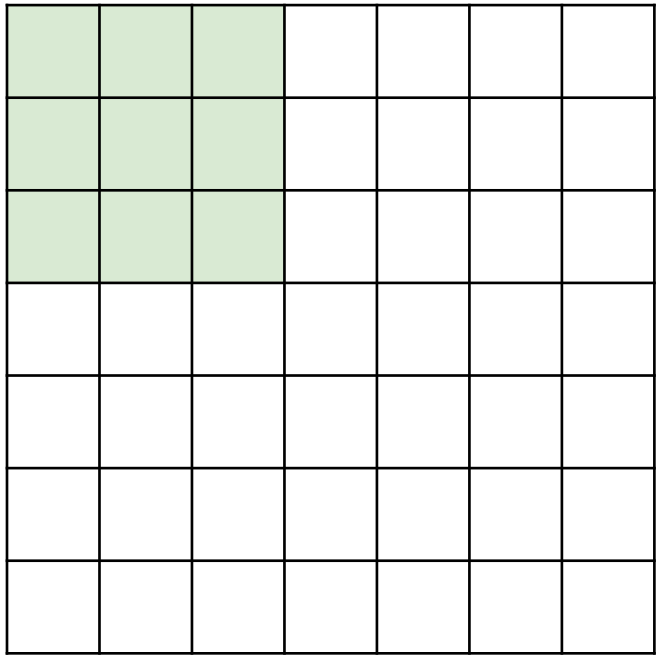


Problem: For large images we need many layers for each output to “see” the whole image image

Solution: Downsample inside the network

Slide credit: D Fouhey & J Johnson

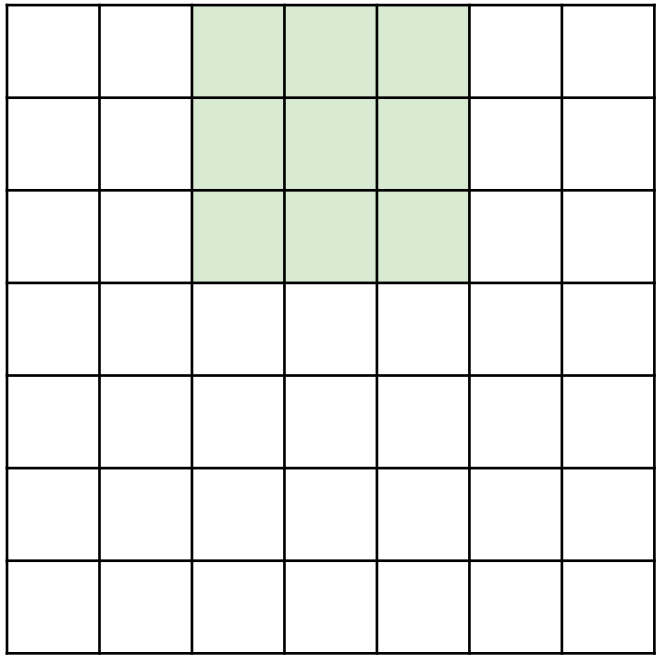
Strided Convolution



Input: 7x7
Filter: 3x3
Stride: 2

Slide credit: D Fouhey & J Johnson

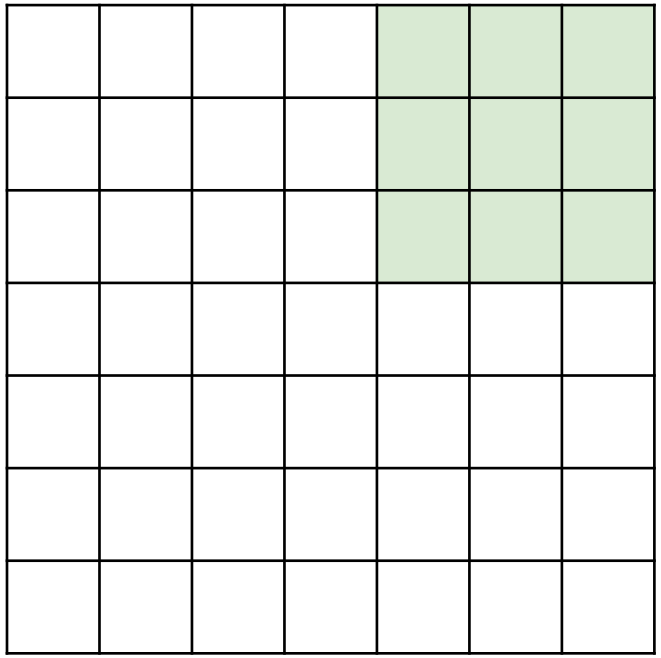
Strided Convolution



Input: 7x7
Filter: 3x3
Stride: 2

Slide credit: D Fouhey & J Johnson

Strided Convolution

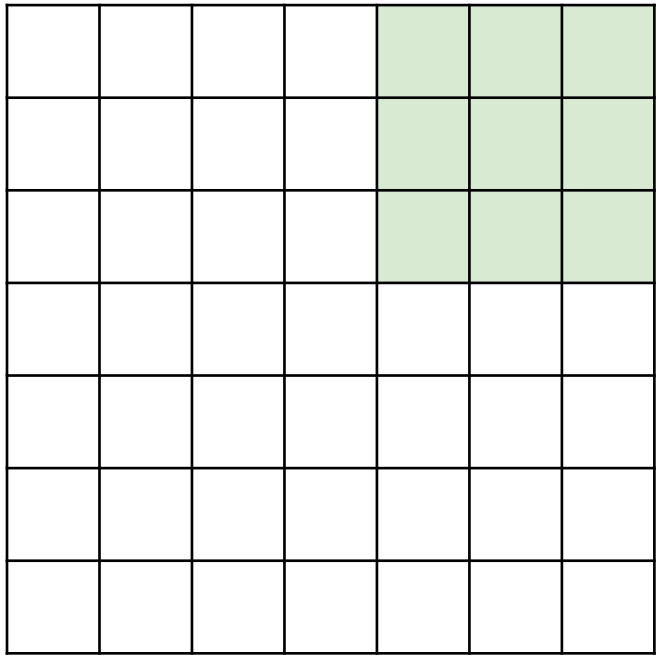


Input: 7x7
Filter: 3x3
Stride: 2

Output: 3x3

Slide credit: D Fouhey & J Johnson

Strided Convolution



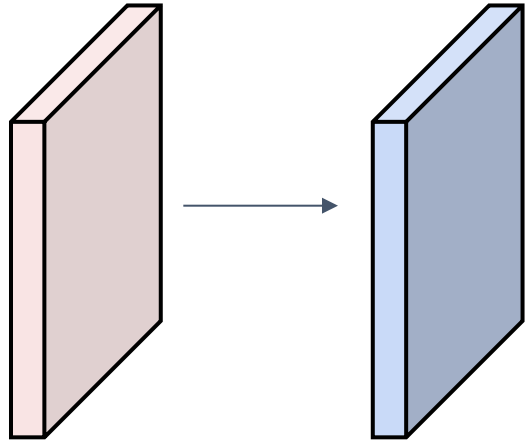
Input: 7x7
Filter: 3x3
Stride: 2

Output: 3x3

In general:
Input: W
Filter: K
Padding: P
Stride: S
Output: (W – K + 2P) / S + 1

Slide credit: D Fouhey & J Johnson

Convolution Example

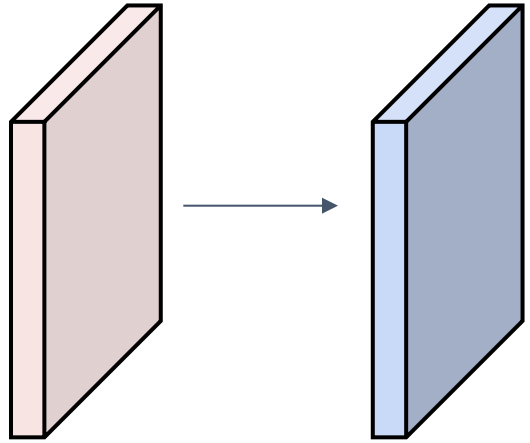


Input volume: 3 x 32 x 32
10 5x5 filters with stride 1, pad 2

Output volume size: ?

Slide credit: D Fouhey & J Johnson

Convolution Example

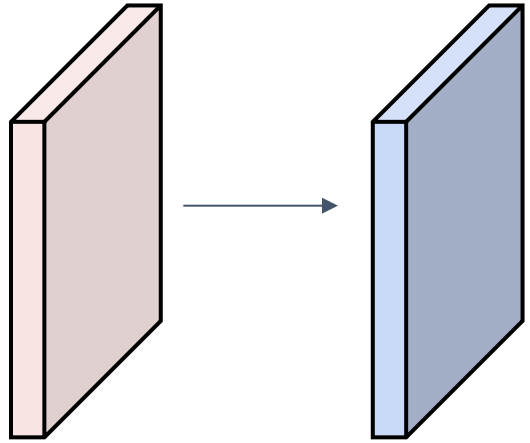


Input volume: 3 x **32** x **32**
10 **5x5** filters with stride **1**, pad **2**

Output volume size:
(**32**+2*__2__-**5**)/**1**+1 = 32 spatially, so
10 x 32 x 32

Slide credit: D Fouhey & J Johnson

Convolution Example

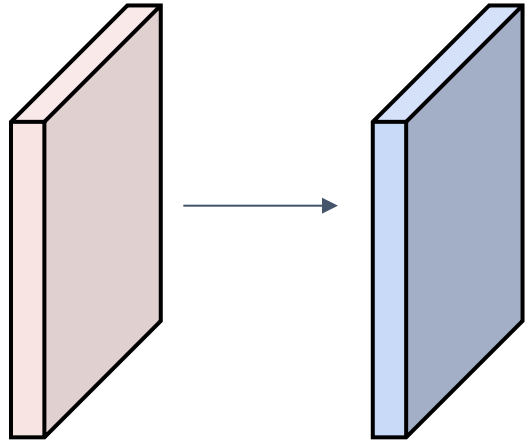


Input volume: 3 x 32 x 32
10 5x5 filters with stride 1, pad 2

Output volume size: 10 x 32 x 32
Number of learnable parameters: ?

Slide credit: D Fouhey & J Johnson

Convolution Example

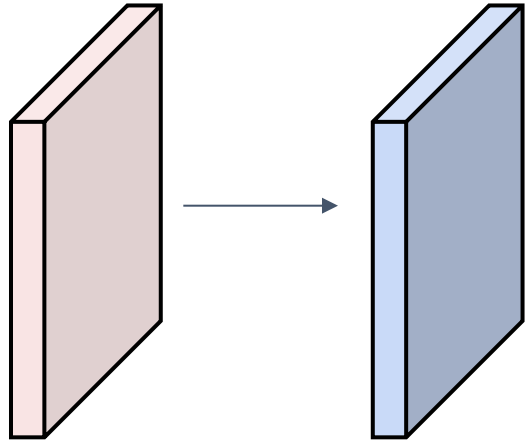


Input volume: 3 x 32 x 32
10 5x5 filters with stride 1, pad 2

Output volume size: 10 x 32 x 32
Number of learnable parameters: **760**
Parameters per filter: 3*5*5 + 1 (for bias) = 76
10 filters, so total is 10 * 76 = **760**

Slide credit: D Fouhey & J Johnson

Convolution Example

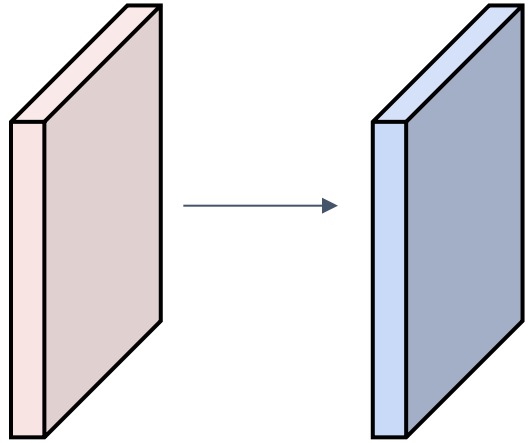


Input volume: 3 x 32 x 32
10 5x5 filters with stride 1, pad 2

Output volume size: 10 x 32 x 32
Number of learnable parameters: 760
Number of multiply-add operations: ?

Slide credit: D Fouhey & J Johnson

Convolution Example



Input volume: **3** x 32 x 32
10 **5x5** filters with stride 1, pad 2

Output volume size: **10 x 32 x 32**
Number of learnable parameters: 760
Number of multiply-add operations: **768,000**
10*32*32 = 10,240 outputs; each output is the inner product of two **3**x**5x5** tensors (75 elems); total = 75*10240 = **768K**

Slide credit: D Fouhey & J Johnson

Convolution Summary

Input: C_{in} x H x W
Hyperparameters:
- **Kernel size**: K_H x K_W
- **Number filters**: C_{out}
- **Padding**: P
- **Stride**: S

Weight matrix: C_{out} x C_{in} x K_H x K_W
giving C_{out} filters of size C_{in} x K_H x K_W

Bias vector: C_{out}

Output size: C_{out} x H' x W' where:
- H' = (H – K + 2P) / S + 1
- W' = (W – K + 2P) / S + 1

Slide credit: D Fouhey & J Johnson

Convolution Summary

Input: C_{in} x H x W
Hyperparameters:
- **Kernel size**: K_H x K_W
- **Number filters**: C_{out}
- **Padding**: P
- **Stride**: S

Weight matrix: C_{out} x C_{in} x K_H x K_W
giving C_{out} filters of size C_{in} x K_H x K_W

Bias vector: C_{out}

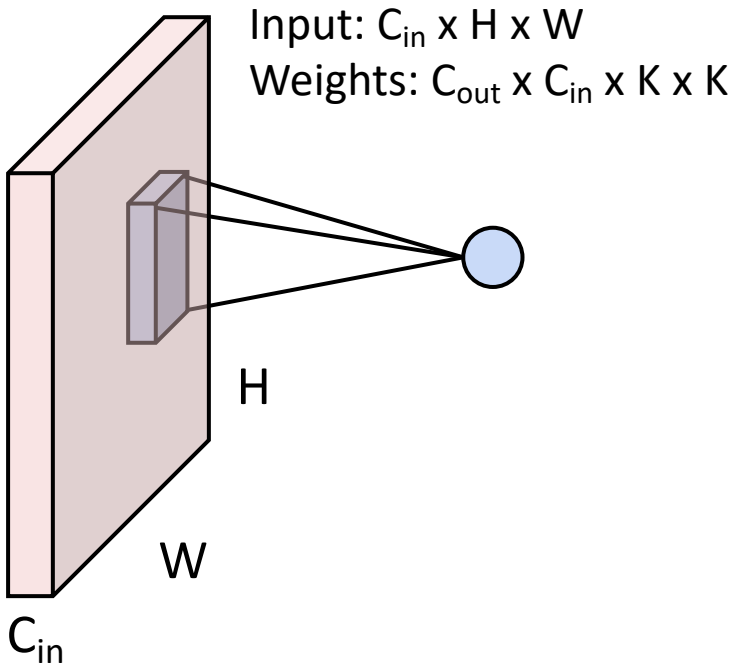
Output size: C_{out} x H' x W' where:
- H' = (H – K + 2P) / S + 1
- W' = (W – K + 2P) / S + 1

Common settings:
$K_H = K_W$ (Small square filters)
P = (K – 1) / 2 ("Same" padding)
C_{in}, C_{out} = 32, 64, 128, 256 (powers of 2)
K = 3, P = 1, S = 1 (3x3 conv)
K = 5, P = 2, S = 1 (5x5 conv)
K = 1, P = 0, S = 1 (1x1 conv)
K = 3, P = 1, S = 2 (Downsample by 2)

Slide credit: D Fouhey & J Johnson

Other types of convolution

So far: 2D Convolution

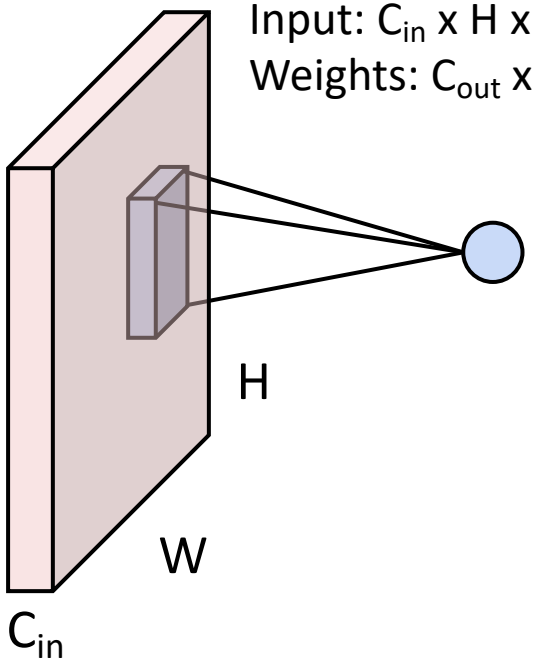


Slide credit: D Fouhey & J Johnson

Other types of convolution

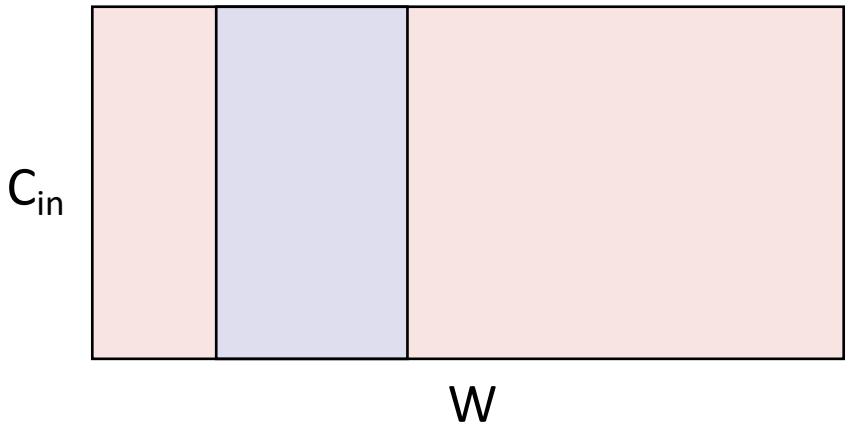
So far: 2D Convolution

Input: C_{in} x H x W
Weights: C_{out} x C_{in} x K x K



1D Convolution

Input: C_{in} x W
Weights: C_{out} x C_{in} x K

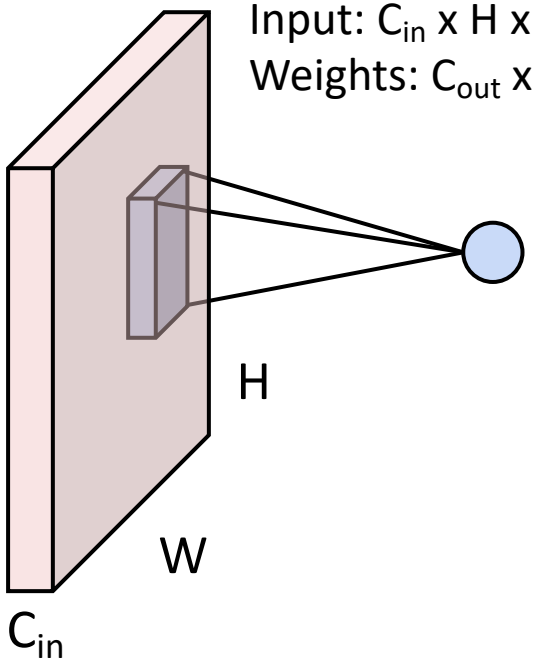


Slide credit: D Fouhey & J Johnson

Other types of convolution

So far: 2D Convolution

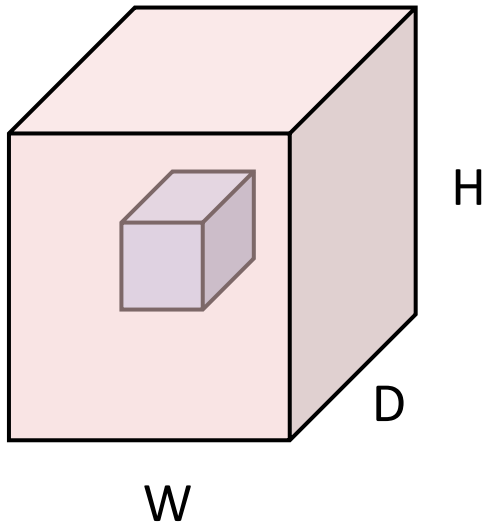
Input: C_{in} x H x W
Weights: C_{out} x C_{in} x K x K



3D Convolution

Input: C_{in} x H x W x D
Weights: C_{out} x C_{in} x K x K x K

C_{in}-dim vector
at each point
in the volume



Slide credit: D Fouhey & J Johnson

PyTorch Convolution Layer

Conv2d

CLASS `torch.nn.Conv2d(`*`in_channels`*`, `*`out_channels`*`, `*`kernel_size`*`, `*`stride=1`*`, `*`padding=0`*`, `*`dilation=1`*`, `*`groups=1`*`, `*`bias=True`*`, `*`padding_mode='zeros'`*`)` [SOURCE]

Applies a 2D convolution over an input signal composed of several input planes.

In the simplest case, the output value of the layer with input size (N, C_{in}, H, W) and output $(N, C_{\text{out}}, H_{\text{out}}, W_{\text{out}})$ can be precisely described as:

$$\text{out}(N_i, C_{\text{out}_j}) = \text{bias}(C_{\text{out}_j}) + \sum_{k=0}^{C_{\text{in}}-1} \text{weight}(C_{\text{out}_j}, k) \star \text{input}(N_i, k)$$

Slide credit: D Fouhey & J Johnson

PyTorch Convolution Layers

Conv2d

CLASS `torch.nn.Conv2d(`*in_channels, out_channels, kernel_size, stride=1, padding=0, dilation=1, groups=1, bias=True, padding_mode='zeros'*`)` [SOURCE]

Conv1d

CLASS `torch.nn.Conv1d(`*in_channels, out_channels, kernel_size, stride=1, padding=0, dilation=1, groups=1, bias=True, padding_mode='zeros'*`)` [SOURCE]

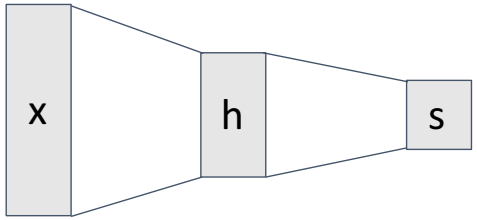
Conv3d

CLASS `torch.nn.Conv3d(`*in_channels, out_channels, kernel_size, stride=1, padding=0, dilation=1, groups=1, bias=True, padding_mode='zeros'*`)` [SOURCE]

Slide credit: D Fouhey & J Johnson

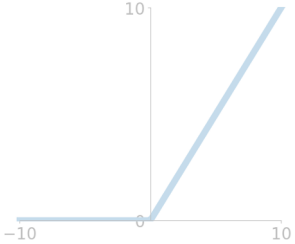
Components of a Convolutional Network

Fully-Connected Layers



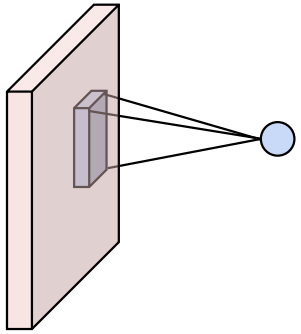
$$y = Wx + b$$

Activation Function

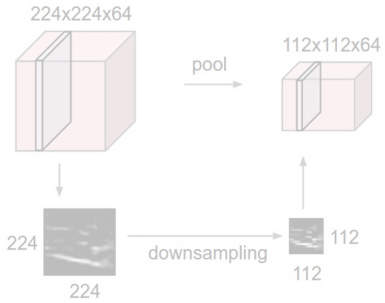


$$y = \max(0, x)$$

Convolution Layers



Pooling Layers



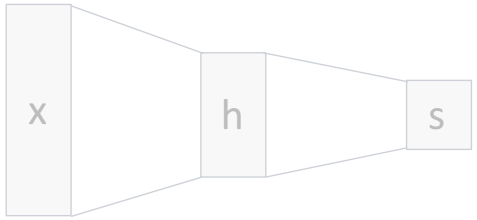
Normalization

$$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \varepsilon}}$$

Slide credit: D Fouhey & J Johnson

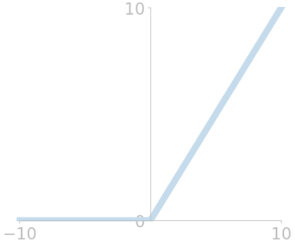
Components of a Convolutional Network

Fully-Connected Layers



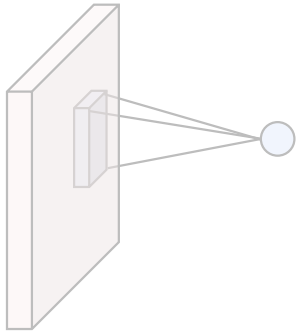
$$y = Wx + b$$

Activation Function

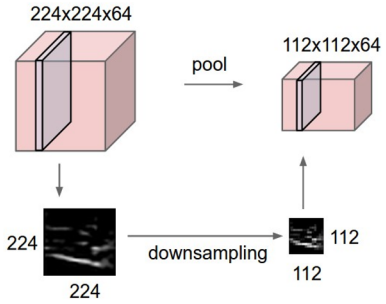


$$y = \max(0, x)$$

Convolution Layers



Pooling Layers

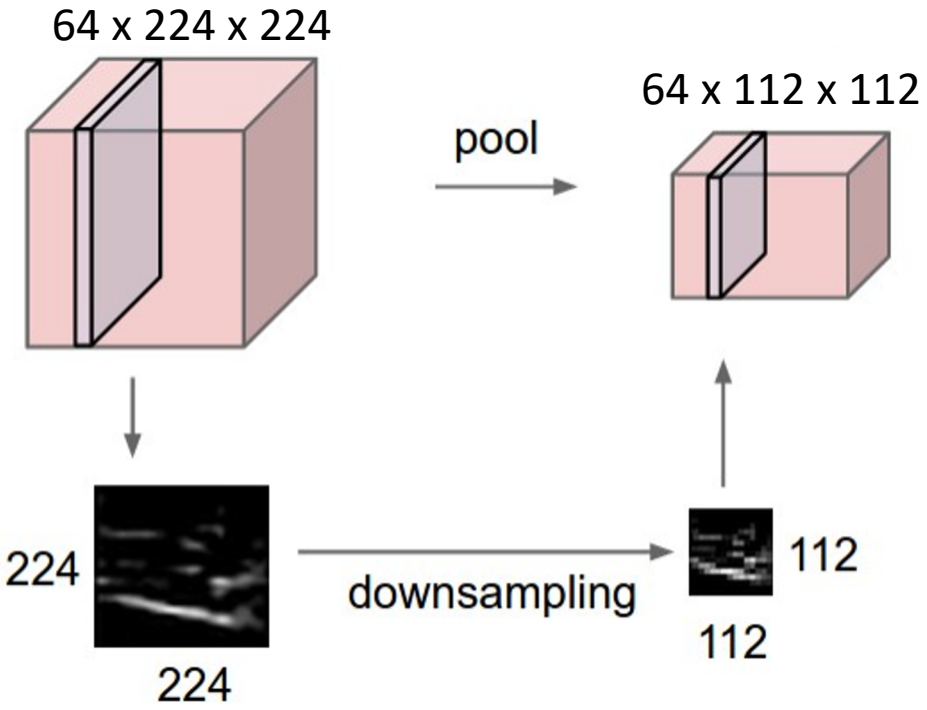


Normalization

$$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \varepsilon}}$$

Slide credit: D Fouhey & J Johnson

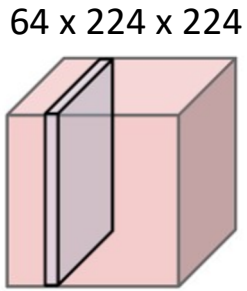
Pooling Layers: Downampling



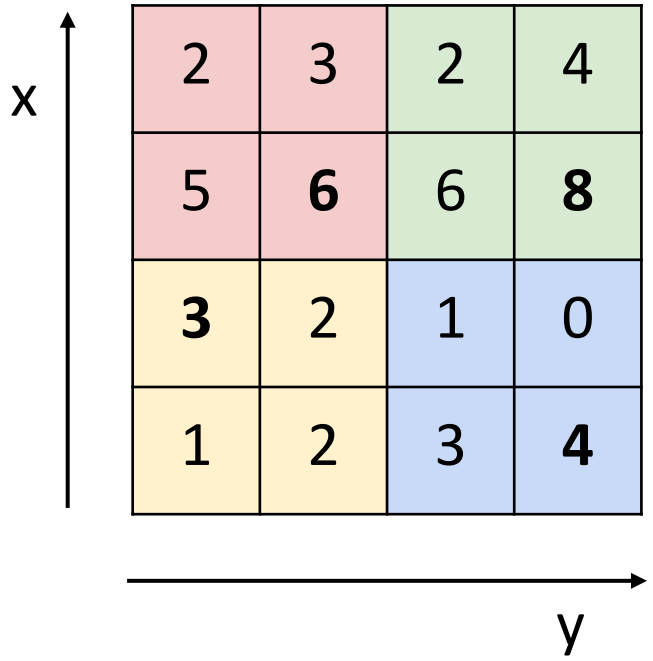
Hyperparameters:
Kernel Size
Stride
Pooling function

Slide credit: D Fouhey & J Johnson

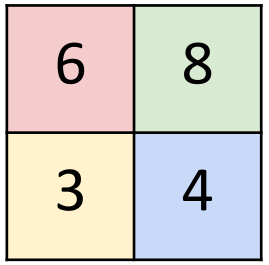
Max Pooling



Single depth slice



Max pooling with 2x2 kernel size and stride 2

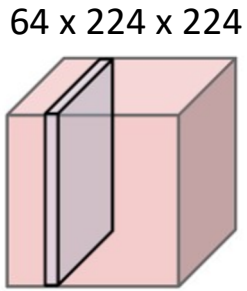


Introduces **invariance** to small spatial shifts

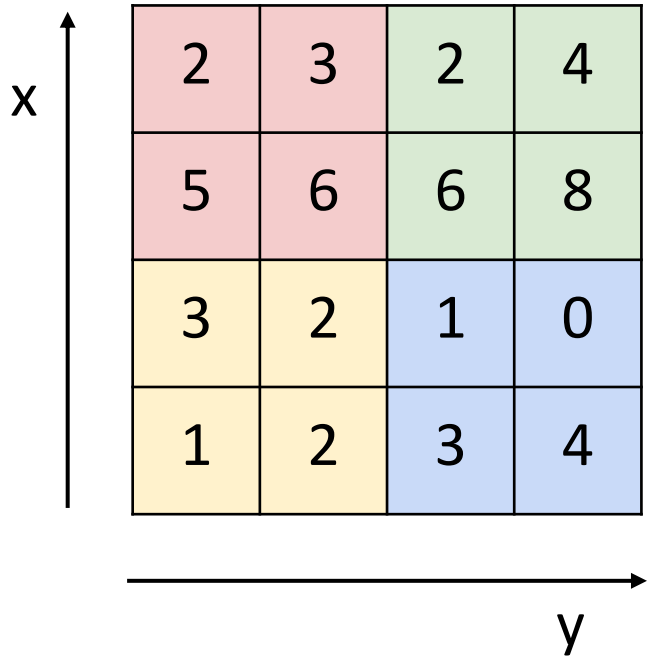
No learnable parameters!

Slide credit: D Fouhey & J Johnson

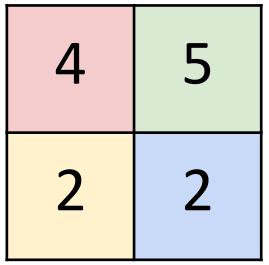
Average Pooling



Single depth slice



Avg pooling with 2x2 kernel size and stride 2



Introduces **invariance** to small spatial shifts

No learnable parameters!

Slide credit: D Fouhey & J Johnson

Pooling Summary

Input: C x H x W

Hyperparameters:

- Kernel size: K
- Stride: S
- Pooling function (max, avg)

Output: C x H’ x W’ where

- H’ = (H – K) / S + 1
- W’ = (W – K) / S + 1

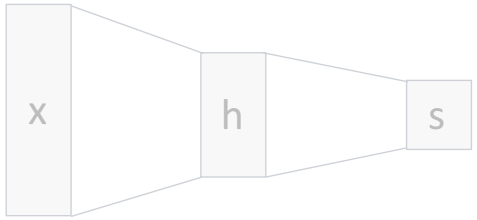
Learnable parameters: None!

Common settings:
max, K = 2, S = 2
max, K = 3, S = 2 (AlexNet)

Slide credit: D Fouhey & J Johnson

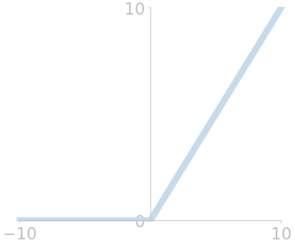
Components of a Convolutional Network

Fully-Connected Layers



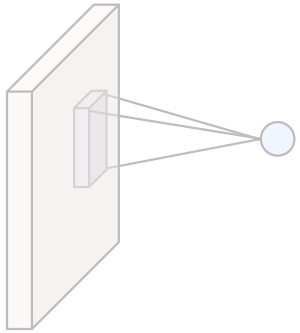
$$y = Wx + b$$

Activation Function

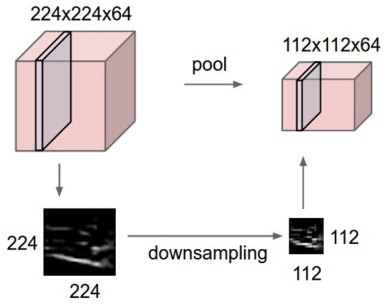


$$y = \max(0, x)$$

Convolution Layers



Pooling Layers



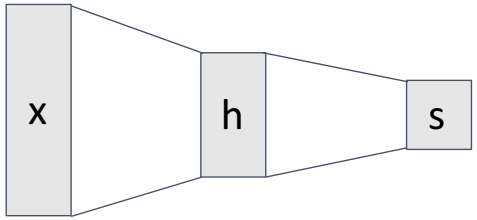
Normalization

$$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \varepsilon}}$$

Slide credit: D Fouhey & J Johnson

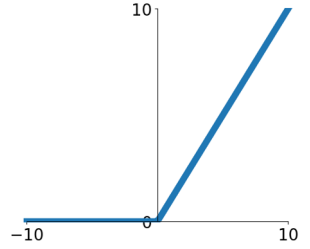
Components of a Convolutional Network

Fully-Connected Layers



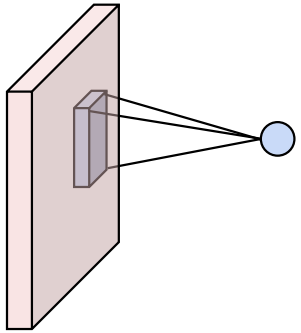
$$y = Wx + b$$

Activation Function

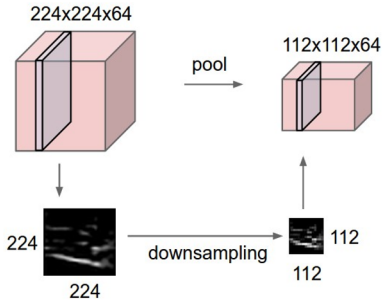


$$y = \max(0, x)$$

Convolution Layers



Pooling Layers



Normalization

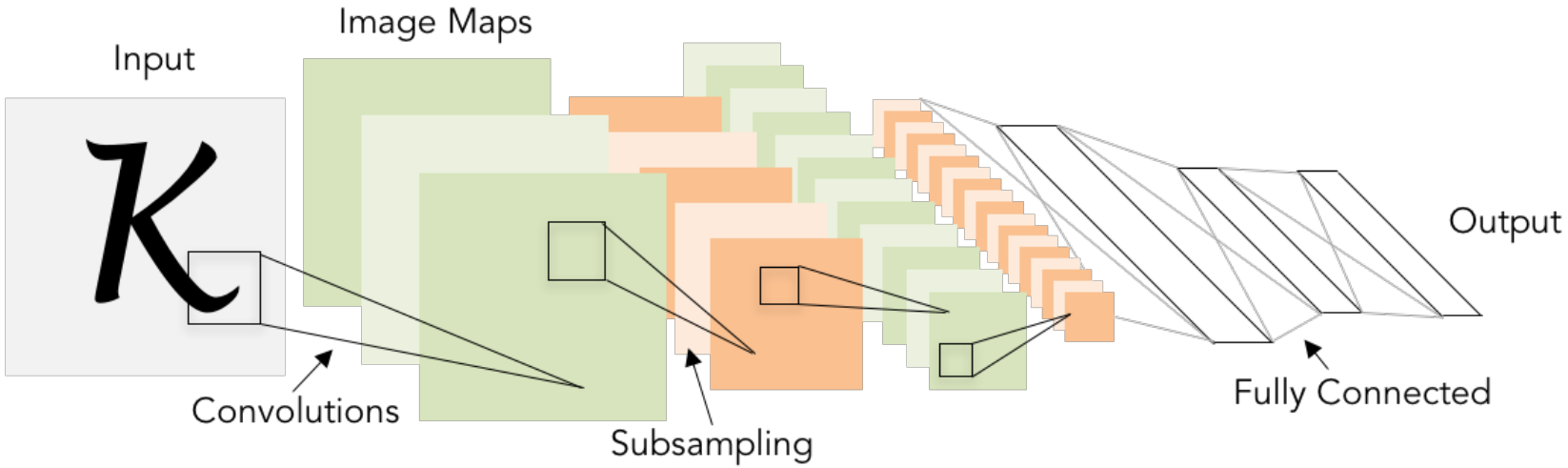
$$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \varepsilon}}$$

Slide credit: D Fouhey & J Johnson

Convolutional Networks

Classic architecture:
[Conv, ReLU, Pool] x N, flatten, [FC, ReLU] x N, FC

Example: LeNet-5

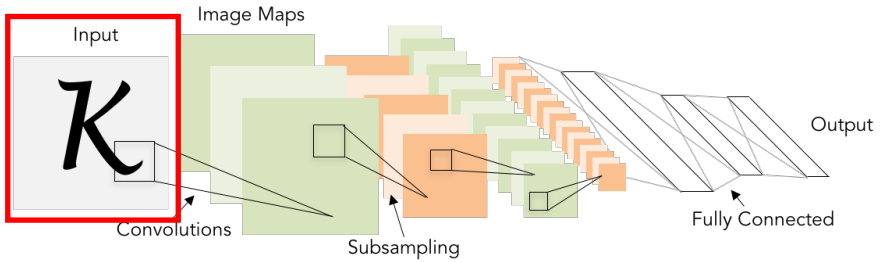


Lecun et al, “Gradient-based learning applied to document recognition”, 1998

Slide credit: D Fouhey & J Johnson

Example: LeNet-5

Layer	Output Size	Weight Size
Input	1 x 28 x 28	

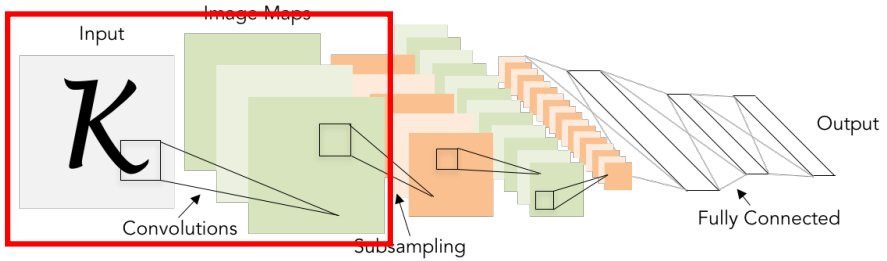


Lecun et al, “Gradient-based learning applied to document recognition”, 1998

Slide credit: D Fouhey & J Johnson

Example: LeNet-5

Layer	Output Size	Weight Size
Input	1 x 28 x 28	
Conv (C_{out}=20, K=5, P=2, S=1)	20 x 28 x 28	20 x 1 x 5 x 5
ReLU	20 x 28 x 28	

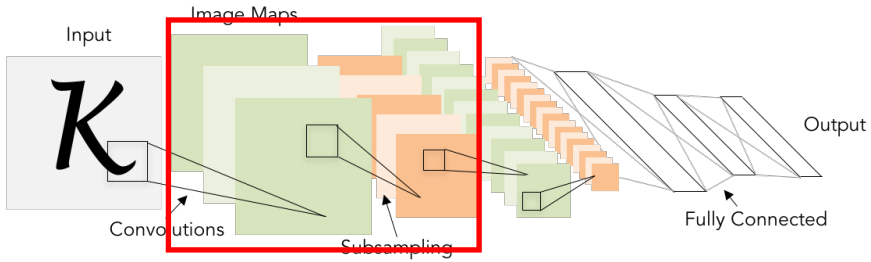


Lecun et al, “Gradient-based learning applied to document recognition”, 1998

Slide credit: D Fouhey & J Johnson

Example: LeNet-5

Layer	Output Size	Weight Size
Input	1 x 28 x 28	
Conv (C_{out}=20, K=5, P=2, S=1)	20 x 28 x 28	20 x 1 x 5 x 5
ReLU	20 x 28 x 28	
MaxPool(K=2, S=2)	20 x 14 x 14	

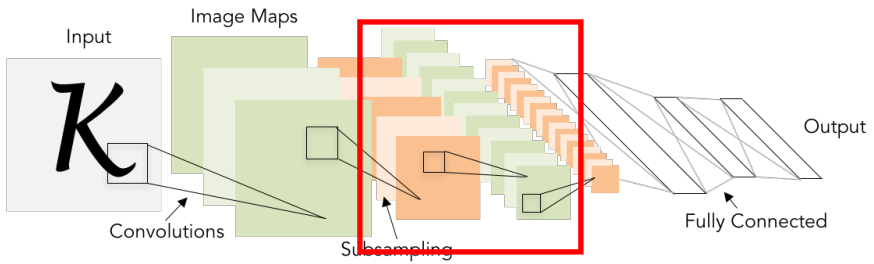


Lecun et al, “Gradient-based learning applied to document recognition”, 1998

Slide credit: D Fouhey & J Johnson

Example: LeNet-5

Layer	Output Size	Weight Size
Input	1 x 28 x 28	
Conv (C_{out}=20, K=5, P=2, S=1)	20 x 28 x 28	20 x 1 x 5 x 5
ReLU	20 x 28 x 28	
MaxPool(K=2, S=2)	20 x 14 x 14	
Conv (C_{out}=50, K=5, P=2, S=1)	50 x 14 x 14	50 x 20 x 5 x 5
ReLU	50 x 14 x 14	

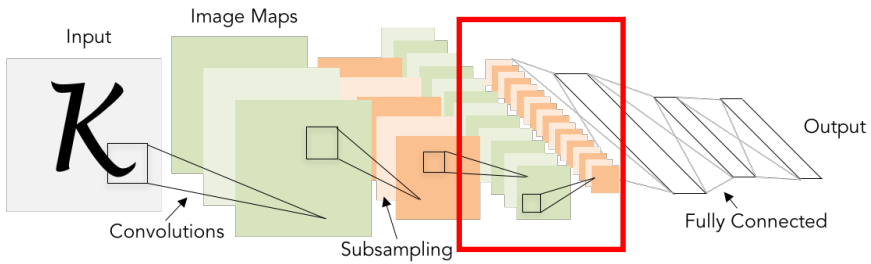


Lecun et al, “Gradient-based learning applied to document recognition”, 1998

Slide credit: D Fouhey & J Johnson

Example: LeNet-5

Layer	Output Size	Weight Size
Input	1 x 28 x 28	
Conv (C_{out}=20, K=5, P=2, S=1)	20 x 28 x 28	20 x 1 x 5 x 5
ReLU	20 x 28 x 28	
MaxPool(K=2, S=2)	20 x 14 x 14	
Conv (C_{out}=50, K=5, P=2, S=1)	50 x 14 x 14	50 x 20 x 5 x 5
ReLU	50 x 14 x 14	
MaxPool(K=2, S=2)	50 x 7 x 7	

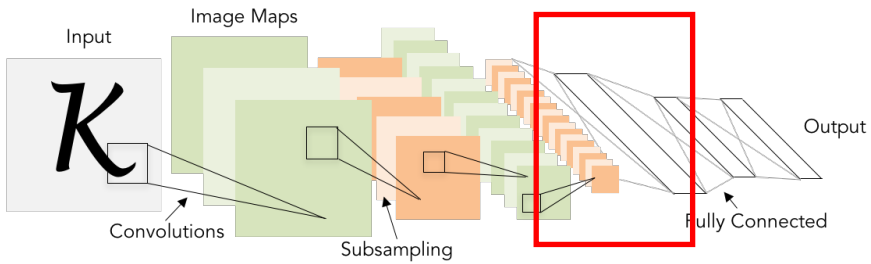


Lecun et al, “Gradient-based learning applied to document recognition”, 1998

Slide credit: D Fouhey & J Johnson

Example: Lenet-5

Layer	Output Size	Weight Size
Input	1 x 28 x 28	
Conv (C_{out}=20, K=5, P=2, S=1)	20 x 28 x 28	20 x 1 x 5 x 5
ReLU	20 x 28 x 28	
MaxPool(K=2, S=2)	20 x 14 x 14	
Conv (C_{out}=50, K=5, P=2, S=1)	50 x 14 x 14	50 x 20 x 5 x 5
ReLU	50 x 14 x 14	
MaxPool(K=2, S=2)	50 x 7 x 7	
Flatten	2450	

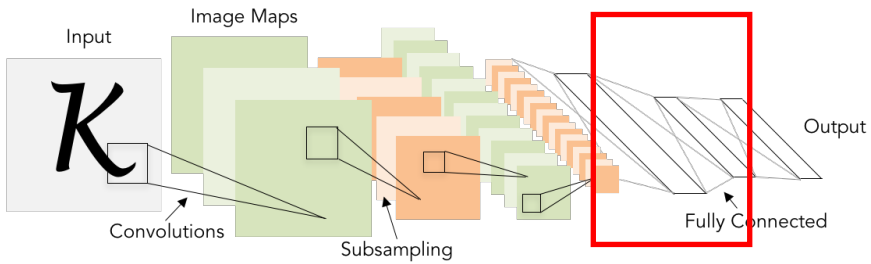


Lecun et al, “Gradient-based learning applied to document recognition”, 1998

Slide credit: D Fouhey & J Johnson

Example: Lenet-5

Layer	Output Size	Weight Size
Input	1 x 28 x 28	
Conv (C_{out}=20, K=5, P=2, S=1)	20 x 28 x 28	20 x 1 x 5 x 5
ReLU	20 x 28 x 28	
MaxPool(K=2, S=2)	20 x 14 x 14	
Conv (C_{out}=50, K=5, P=2, S=1)	50 x 14 x 14	50 x 20 x 5 x 5
ReLU	50 x 14 x 14	
MaxPool(K=2, S=2)	50 x 7 x 7	
Flatten	2450	
Linear (2450 -> 500)	500	2450 x 500
ReLU	500	

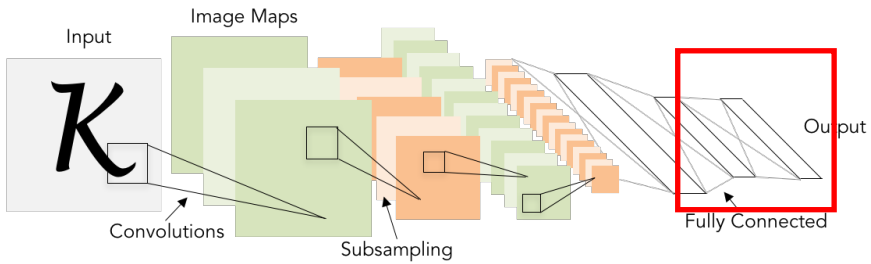


Lecun et al, “Gradient-based learning applied to document recognition”, 1998

Slide credit: D Fouhey & J Johnson

Example: Lenet-5

Layer	Output Size	Weight Size
Input	1 x 28 x 28	
Conv (C_{out}=20, K=5, P=2, S=1)	20 x 28 x 28	20 x 1 x 5 x 5
ReLU	20 x 28 x 28	
MaxPool(K=2, S=2)	20 x 14 x 14	
Conv (C_{out}=50, K=5, P=2, S=1)	50 x 14 x 14	50 x 20 x 5 x 5
ReLU	50 x 14 x 14	
MaxPool(K=2, S=2)	50 x 7 x 7	
Flatten	2450	
Linear (2450 -> 500)	500	2450 x 500
ReLU	500	
Linear (500 -> 10)	10	500 x 10

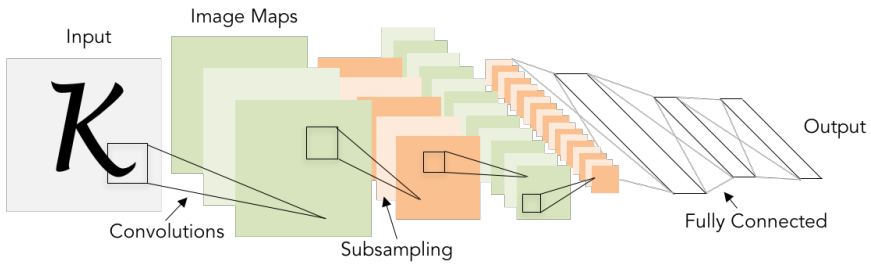


Lecun et al, “Gradient-based learning applied to document recognition”, 1998

Slide credit: D Fouhey & J Johnson

Example: Lenet-5

Layer	Output Size	Weight Size
Input	1 x 28 x 28	
Conv (C_{out}=20, K=5, P=2, S=1)	20 x 28 x 28	20 x 1 x 5 x 5
ReLU	20 x 28 x 28	
MaxPool(K=2, S=2)	20 x 14 x 14	
Conv (C_{out}=50, K=5, P=2, S=1)	50 x 14 x 14	50 x 20 x 5 x 5
ReLU	50 x 14 x 14	
MaxPool(K=2, S=2)	50 x 7 x 7	
Flatten	2450	
Linear (2450 -> 500)	500	2450 x 500
ReLU	500	
Linear (500 -> 10)	10	500 x 10



As we go through the network:

Spatial size **decreases**
(using pooling or strided conv)

Number of channels **increases**
(total "volume" is preserved!)

Lecun et al, "Gradient-based learning applied to document recognition", 1998

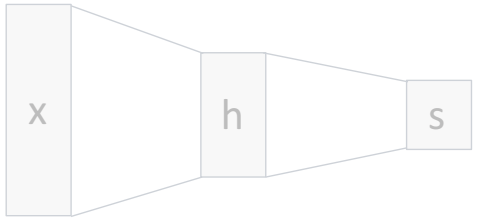
Slide credit: D Fouhey & J Johnson

Problem: Deep Networks very hard to train!

Slide credit: D Fouhey & J Johnson

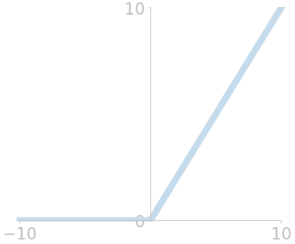
Components of a Convolutional Network

Fully-Connected Layers



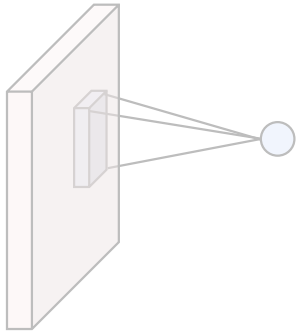
$$y = Wx + b$$

Activation Function

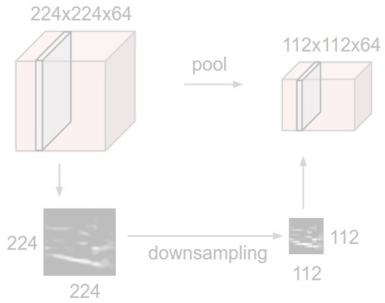


$$y = \max(0, x)$$

Convolution Layers



Pooling Layers



Normalization

$$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \varepsilon}}$$

Slide credit: D Fouhey & J Johnson

Batch Normalization

Idea: “Normalize” the outputs of each layer so they have zero mean and unit variance

Why? Helps reduce “internal covariate shift”, improves optimization

Ioffe and Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift”, ICML 2015

Slide credit: D Fouhey & J Johnson

Batch Normalization

Idea: "Normalize" the outputs of each layer so they have zero mean and unit variance

Why? Helps reduce "internal covariate shift", improves optimization

We can normalize a batch of activations like this:

$$\hat{x} = \frac{x - E[x]}{\sqrt{Var[x]}}$$

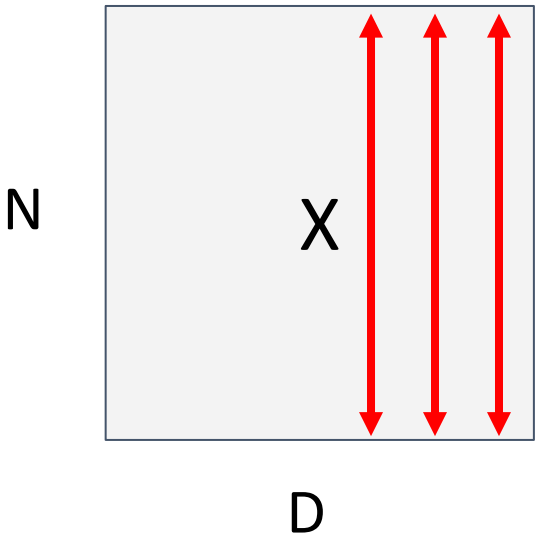
This is a **differentiable function**, so we can use it as an operator in our networks and backprop through it!

Ioffe and Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift", ICML 2015

Slide credit: D Fouhey & J Johnson

Batch Normalization

Input: $x \in \mathbb{R}^{N \times D}$



$$\mu_j = \frac{1}{N}\sum_{i=1}^{N} x_{i,j}$$

Per-channel mean, shape is D

$$\sigma_j^2 = \frac{1}{N}\sum_{i=1}^{N}\left(x_{i,j} - \mu_j\right)^2$$

Per-channel std, shape is D

$$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \varepsilon}}$$

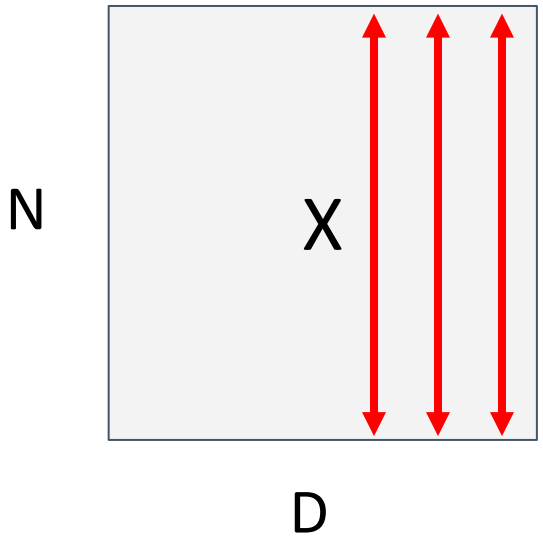
Normalized x, Shape is N x D

Ioffe and Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift", ICML 2015

Slide credit: D Fouhey & J Johnson

Batch Normalization

Input: $x \in \mathbb{R}^{N \times D}$



$$\mu_j = \frac{1}{N}\sum_{i=1}^{N} x_{i,j}$$

Per-channel mean, shape is D

$$\sigma_j^2 = \frac{1}{N}\sum_{i=1}^{N}\left(x_{i,j} - \mu_j\right)^2$$

Per-channel std, shape is D

$$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \varepsilon}}$$

Normalized x, Shape is N x D

Problem: What if zero-mean, unit variance is too restrictive?

Ioffe and Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift", ICML 2015

Slide credit: D Fouhey & J Johnson

Batch Normalization

Input: $x \in \mathbb{R}^{N \times D}$

Learnable scale and shift parameters:

$$\gamma, \beta \in \mathbb{R}^{D}$$

Learning $\gamma = \sigma, \beta = \mu$ will recover the identity function (in expectation)

$$\mu_j = \frac{1}{N}\sum_{i=1}^{N} x_{i,j}$$

Per-channel mean, shape is D

$$\sigma_j^2 = \frac{1}{N}\sum_{i=1}^{N} \left(x_{i,j} - \mu_j\right)^2$$

Per-channel std, shape is D

$$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \varepsilon}}$$

Normalized x, Shape is N x D

$$y_{i,j} = \gamma_j \hat{x}_{i,j} + \beta_j$$

Output, Shape is N x D

Slide credit: D Fouhey & J Johnson

Batch Normalization

Problem: Estimates depend on minibatch; can't do this at test-time!

Input: $x \in \mathbb{R}^{N \times D}$

Learnable scale and shift parameters:

$$\gamma, \beta \in \mathbb{R}^{D}$$

Learning $\gamma = \sigma, \beta = \mu$ will recover the identity function (in expectation)

$$\mu_j = \frac{1}{N}\sum_{i=1}^{N} x_{i,j}$$

Per-channel mean, shape is D

$$\sigma_j^2 = \frac{1}{N}\sum_{i=1}^{N} (x_{i,j} - \mu_j)^2$$

Per-channel std, shape is D

$$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \varepsilon}}$$

Normalized x, Shape is N x D

$$y_{i,j} = \gamma_j \hat{x}_{i,j} + \beta_j$$

Output, Shape is N x D

Slide credit: D Fouhey & J Johnson

Batch Normalization: Test-Time

Input: $x \in \mathbb{R}^{N \times D}$

Learnable scale and shift parameters:

$$\gamma, \beta \in \mathbb{R}^{D}$$

Learning $\gamma = \sigma, \beta = \mu$ will recover the identity function (in expectation)

$\mu_j =$ (Running) average of values seen during training — Per-channel mean, shape is D

$\sigma_j^2 =$ (Running) average of values seen during training — Per-channel std, shape is D

$$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \varepsilon}}$$

Normalized x, Shape is N x D

$$y_{i,j} = \gamma_j \hat{x}_{i,j} + \beta_j$$

Output, Shape is N x D

Slide credit: D Fouhey & J Johnson

Batch Normalization: Test-Time

Input: $x \in \mathbb{R}^{N \times D}$

Learnable scale and shift parameters:

$\gamma, \beta \in \mathbb{R}^{D}$

During testing batchnorm becomes a linear operator!
Can be fused with the previous fully-connected or conv layer

$\mu_j =$ (Running) average of values seen during training — Per-channel mean, shape is D

$\sigma_j^2 =$ (Running) average of values seen during training — Per-channel std, shape is D

$$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \varepsilon}}$$

Normalized x, Shape is N x D

$$y_{i,j} = \gamma_j \hat{x}_{i,j} + \beta_j$$

Output, Shape is N x D

Slide credit: D Fouhey & J Johnson

Batch Normalization for ConvNets

Batch Normalization for
fully-connected networks

$$x : N \times D$$

Normalize ↓

$$\mu, \sigma : 1 \times D$$

$$\gamma, \beta : 1 \times D$$

$$y = \frac{(x - \mu)}{\sigma}\gamma + \beta$$

Batch Normalization for
convolutional networks
(Spatial Batchnorm, BatchNorm2D)

$$x : N \times C \times H \times W$$

Normalize ↓ ↓ ↓

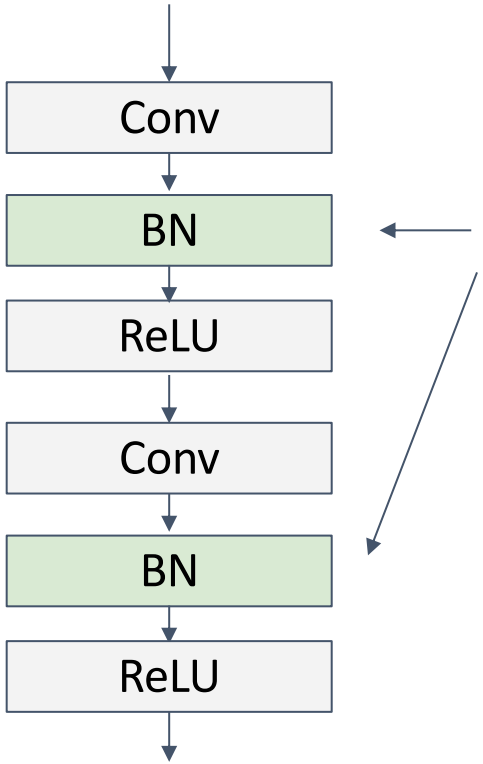
$$\mu, \sigma : 1 \times C \times 1 \times 1$$

$$\gamma, \beta : 1 \times C \times 1 \times 1$$

$$y = \frac{(x - \mu)}{\sigma}\gamma + \beta$$

Slide credit: D Fouhey & J Johnson

Batch Normalization

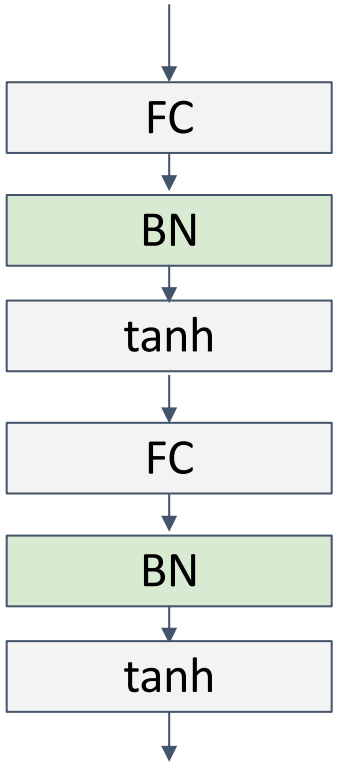


Usually inserted after Fully Connected or Convolutional layers, and before nonlinearity.

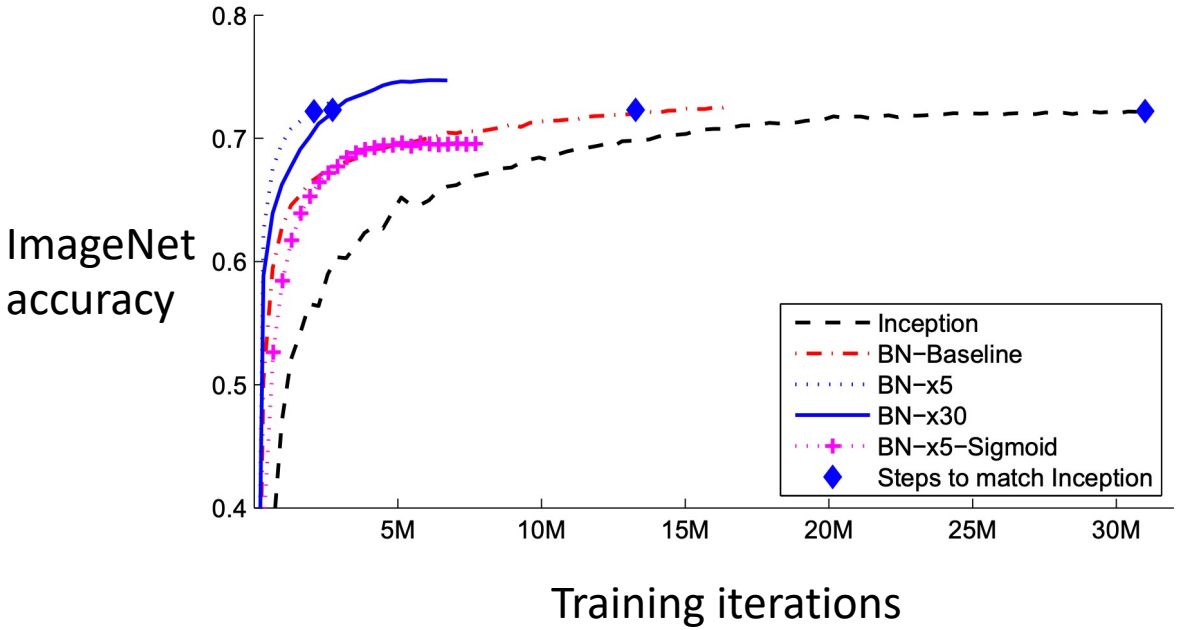
$$\hat{x} = \frac{x - E[x]}{\sqrt{Var[x]}}$$

Slide credit: D Fouhey & J Johnson

Batch Normalization

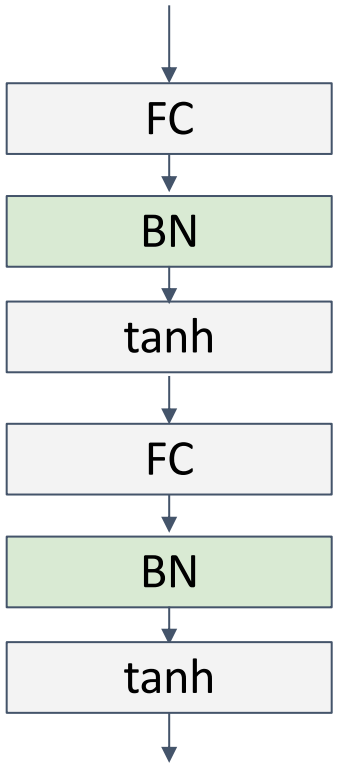


- Makes deep networks **much** easier to train!
- Allows higher learning rates, faster convergence
- Networks become more robust to initialization
- Acts as regularization during training
- Free at test-time: can be fused with conv!



Slide credit: D Fouhey & J Johnson

Batch Normalization

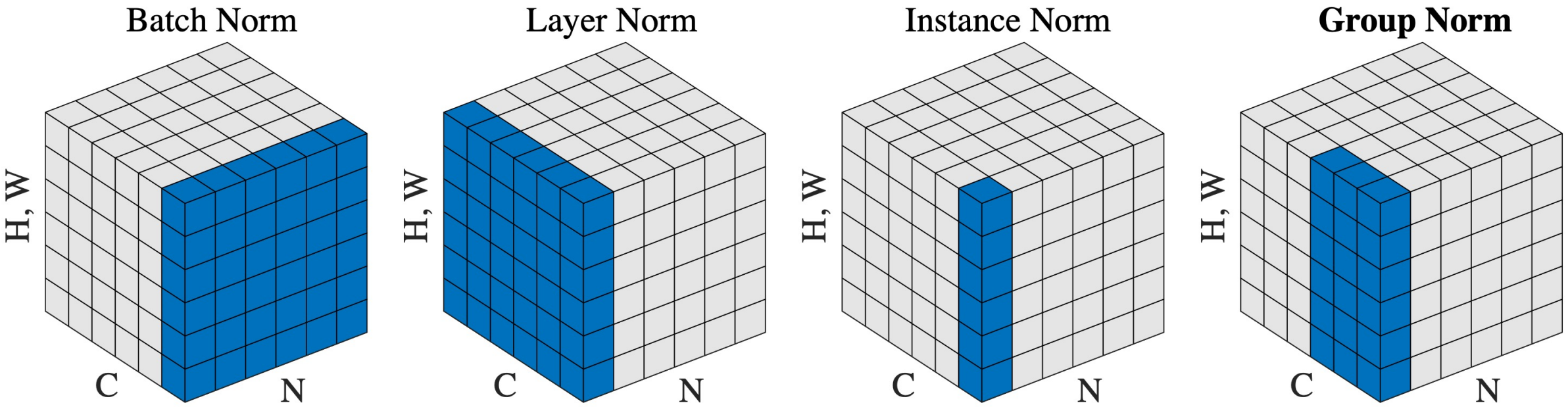


Ioffe and Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift", ICML 2015

- Makes deep networks **much** easier to train!
- Allows higher learning rates, faster convergence
- Networks become more robust to initialization
- Acts as regularization during training
- Free at test-time: can be fused with conv!
- Not well-understood theoretically (yet)
- Behaves differently during training and testing: this is a very common source of bugs!

Slide credit: D Fouhey & J Johnson

Different Normalization Layers



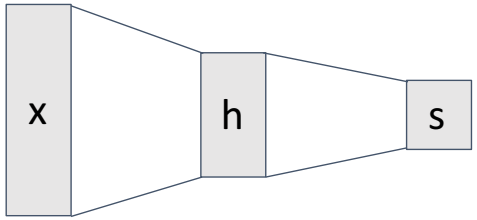
[Wu and He. Group Normalization. ECCV 2018. Best paper honorable mention.]

Devils in the details (mainly for PyTorch)

1. To reliably estimate BN statistics (running mean and average), you need at least 8 samples on each GPU
2. If you don’t have enough samples on each GPU
    1. Synchronized BN layers
    2. Group normalization layers
3. Instance normalization layers are useful for some applications: such as style transfer, dense correspondence

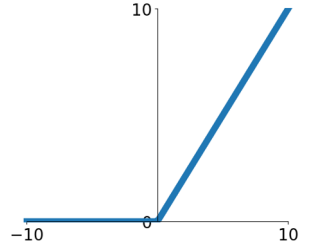
Components of a Convolutional Network

Fully-Connected Layers



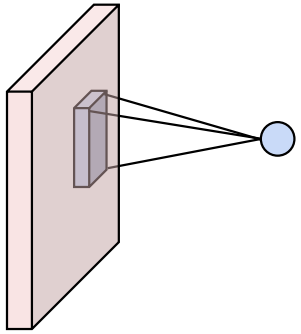
$$y = Wx + b$$

Activation Function

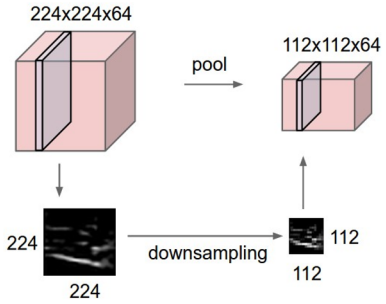


$$y = \max(0, x)$$

Convolution Layers



Pooling Layers



Normalization

$$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \varepsilon}}$$

Slide credit: D Fouhey & J Johnson

Next Class

More about Convolutional Neural Networks