Lecture 5:
Image Classification with CNNs

Administrative: Assignment 1

Assignment 1 Due **Wednesday 4/16** at 11:59pm

- K-Nearest Neighbor
- Linear classifiers: SVM, Softmax
- Two-layer neural network
- Image features

Administrative: Project Proposal

Due Thursday 4/23

TA expertise is posted on the webpage.

http://cs231n.stanford.edu/office_hours.html

Administrative: OAE Letters

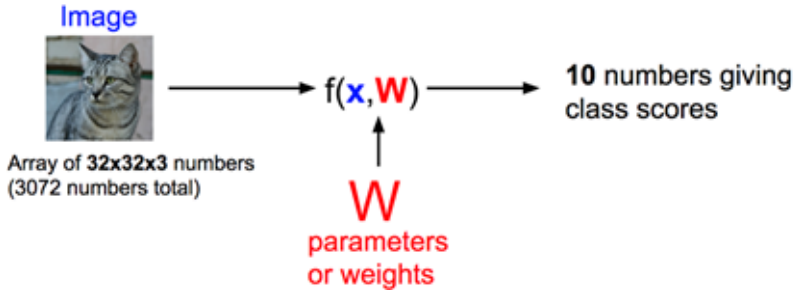
Due this Friday 4/17

- Assignments
- Midterm

CS231n: Deep Learning for Computer Vision

- Deep Learning Basics (Lecture 2 – 4)
- Perceiving and Understanding the Visual World (Lecture 5 – 12)
- Generative and Interactive Visual Intelligence (Lecture 13 – 16)
- Human-Centered Applications and Implications (Lecture 17 – 18)

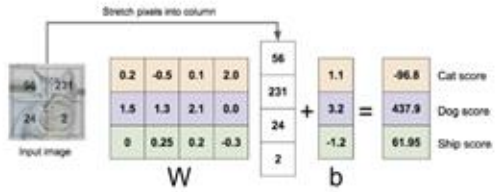
Recap: Image Classification with Linear Classifier



$$f(x,W) = Wx + b$$

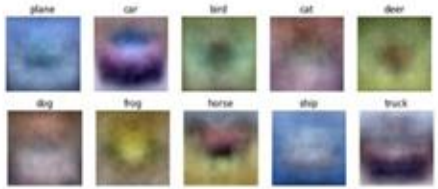
Algebraic Viewpoint

$$f(x,W) = Wx$$



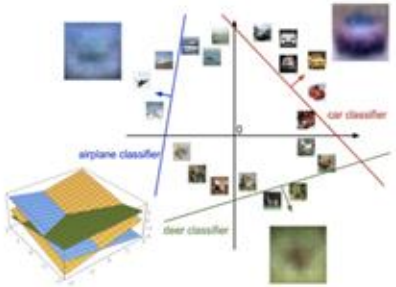
Visual Viewpoint

One template per class



Geometric Viewpoint

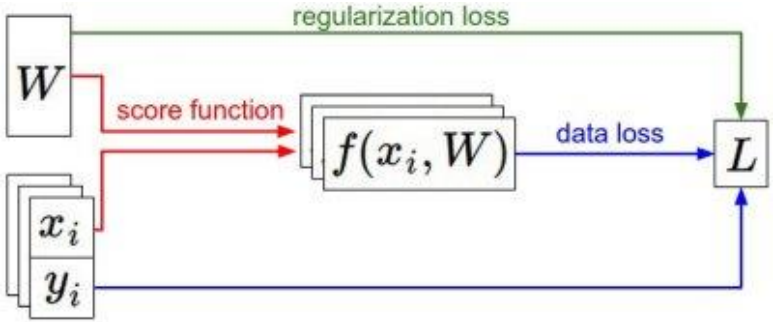
Hyperplanes cutting up space



Recap: Loss Function

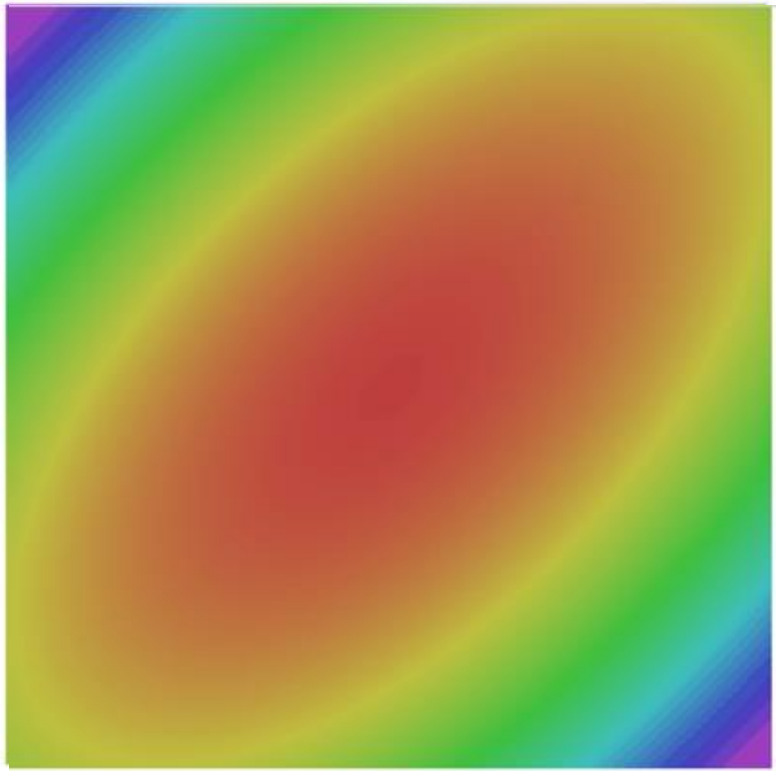
- We have some dataset of (x,y)
- We have a score function: $s = f(x; W) = Wx$
- We have a loss function:

$$L_i = -\log\left(\frac{e^{sy_i}}{\sum_j e^{s_j}}\right)$$ Softmax



$$L = \frac{1}{N}\sum_{i=1}^{N} L_i + R(W)$$ Full loss

Recap: Optimization



SGD

SGD+Momentum

RMSProp

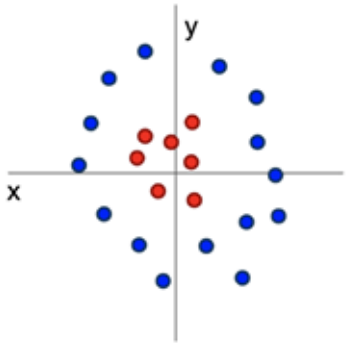
Adam

Problem: Linear Classifiers are not very powerful

Visual Viewpoint

Linear classifiers learn one template per class

Geometric Viewpoint

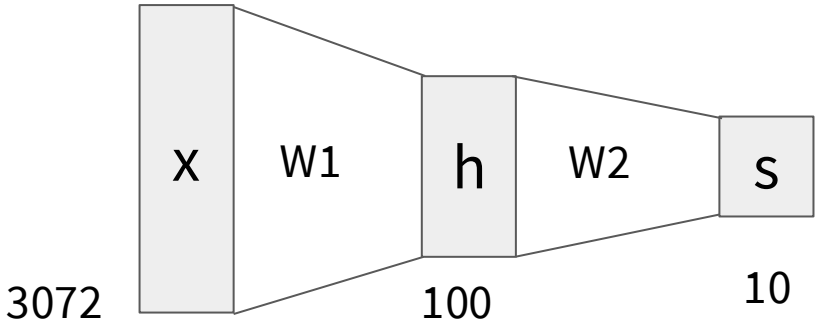


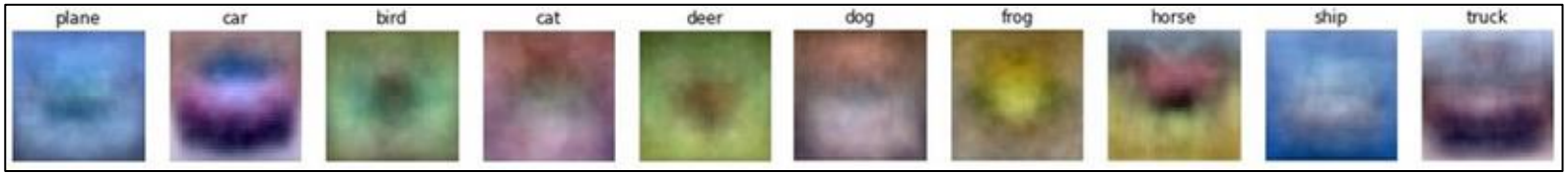
Linear classifiers can only draw linear decision boundaries

Last time: Neural Networks

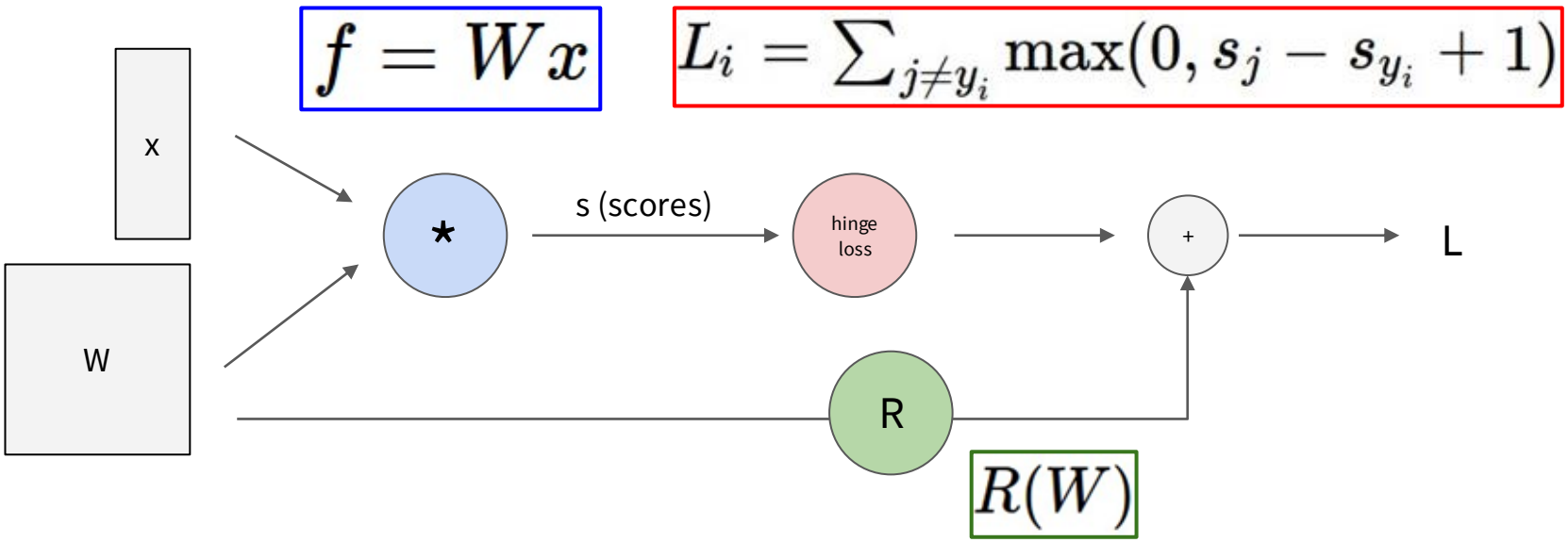
Linear score function: $f = Wx$

2-layer Neural Network $f = W_2 \max(0, W_1 x)$

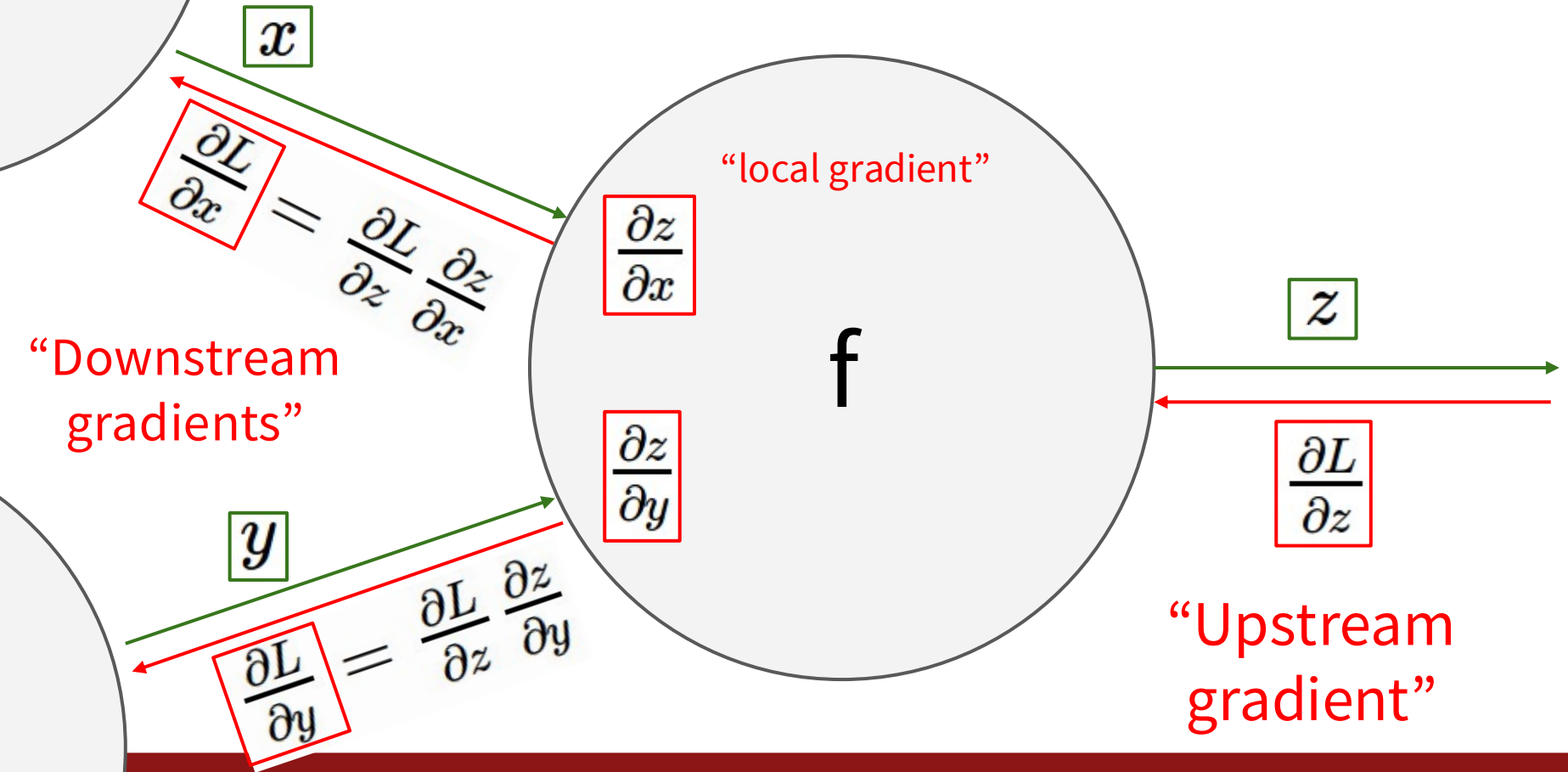




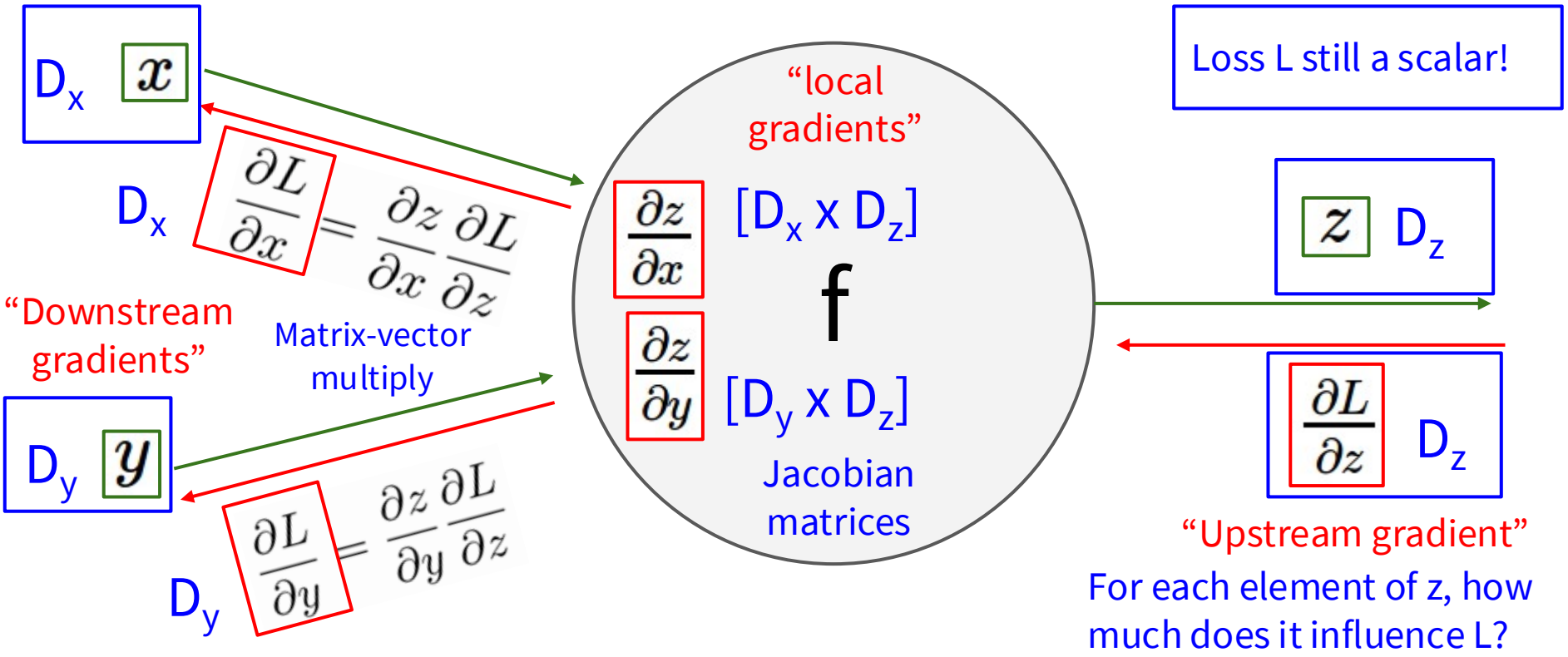
Last time: Computation Graph



Last time: Backpropagation



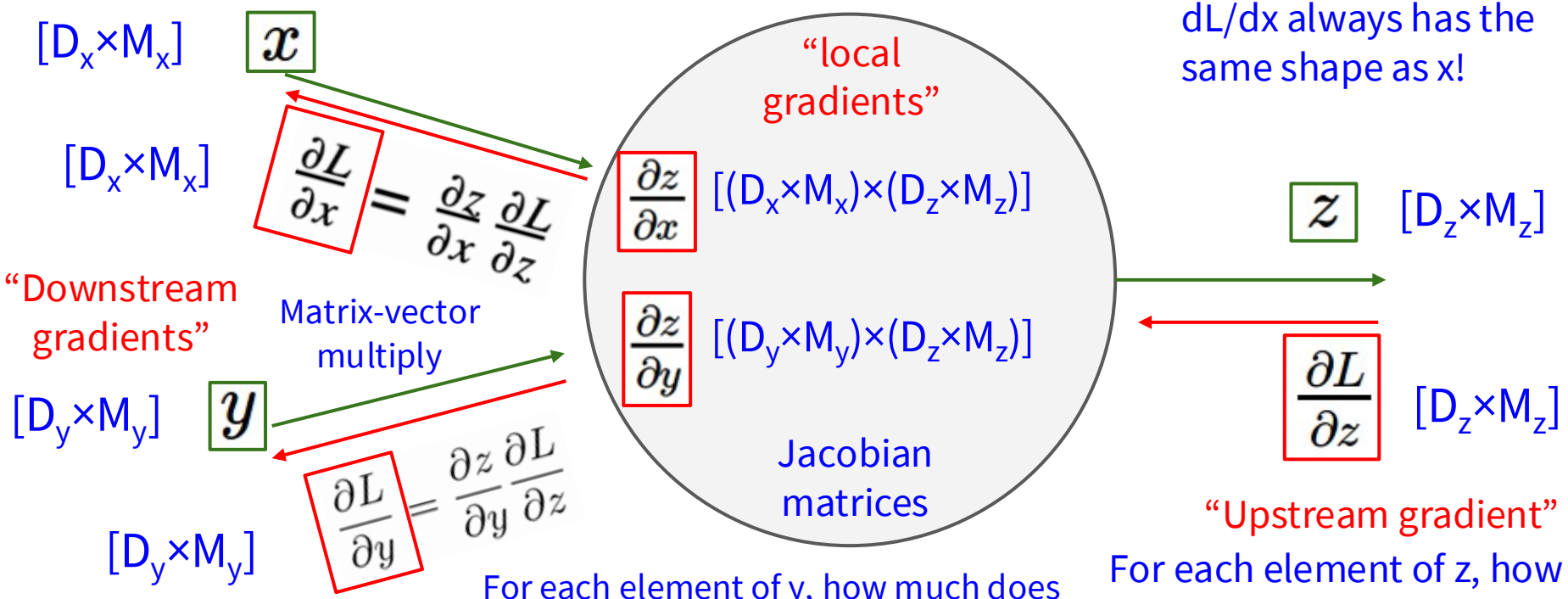
Backprop with Vectors



Backprop with Matrices (or Tensors)

Loss L still a scalar!

dL/dx always has the same shape as x!



For each element of y, how much does it influence each element of z?

"Upstream gradient"

For each element of z, how much does it influence L?

CS231n: Deep Learning for Computer Vision

- Deep Learning Basics (Lecture 2 – 4)
- Perceiving and Understanding the Visual World (Lecture 5 – 12)
- Generative and Interactive Visual Intelligence (Lecture 13 – 16)
- Human-Centered Applications and Implications (Lecture 17 – 18)

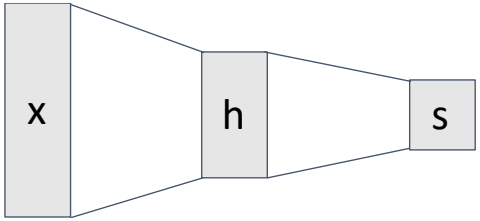
CS231n: Deep Learning for Computer Vision

- Deep Learning Basics (Lecture 2 – 4)
- Perceiving and Understanding the Visual World (Lecture 5 – 12)
- Generative and Interactive Visual Intelligence (Lecture 13 – 16)
- Human-Centered Applications and Implications (Lecture 17 – 18)

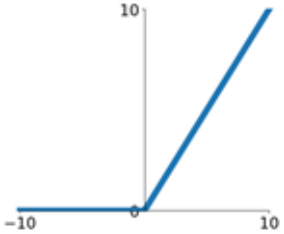
Today: Convolutional Networks

We have already seen these

Fully-Connected Layer



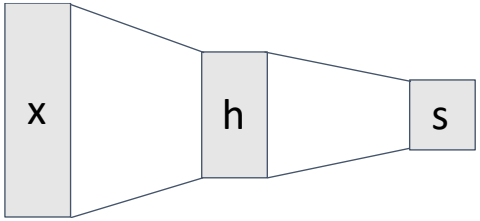
Activation Function



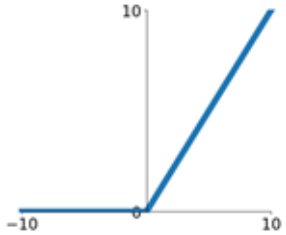
Today: Convolutional Networks

We have already seen these

Fully-Connected Layer

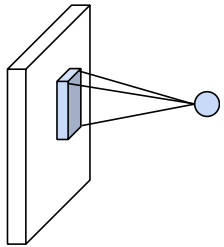


Activation Function



Today: Image-specific operators

Convolution Layer



Pooling Layer

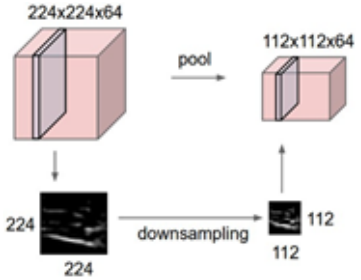


Image Classification: A core task in Computer Vision

(assume given a set of labels)
{dog, cat, truck, plane, ...}

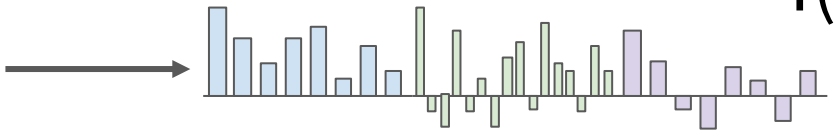
cat
dog
bird
deer
truck

Pixel space

$$f(x) = Wx$$

Class scores

Image features

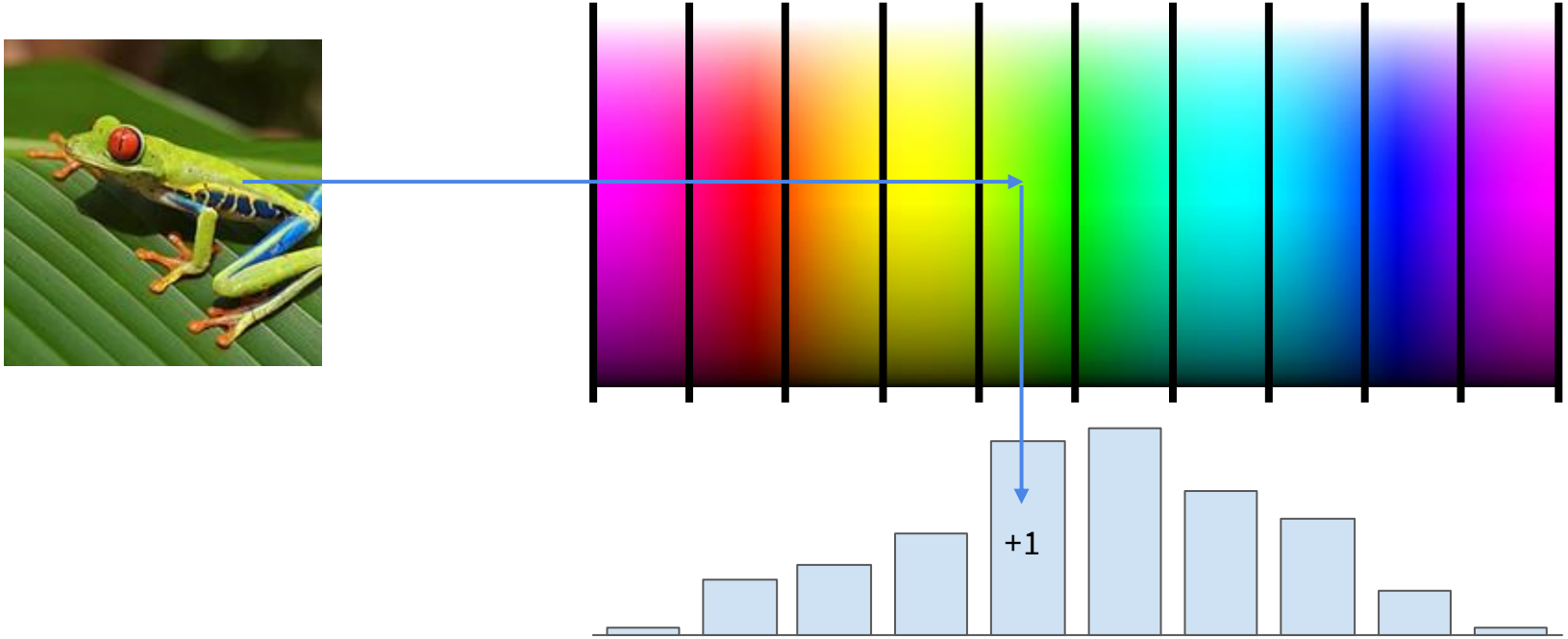


f(x) = Wx

Feature Representation

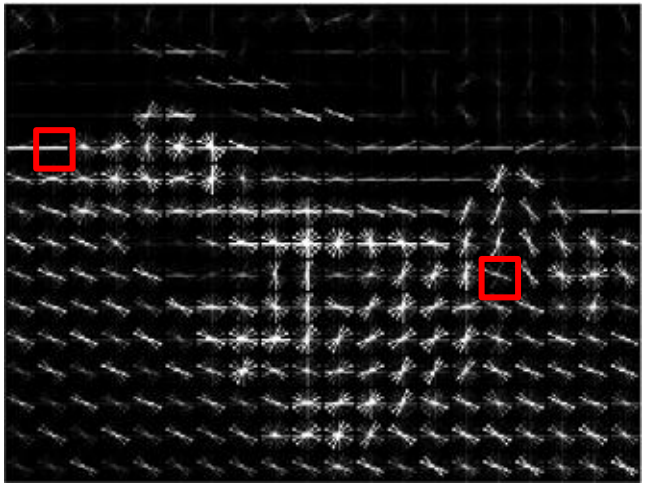
Class scores

Example: Color Histogram



Example: Histogram of Oriented Gradients (HoG)

Divide image into 8x8 pixel regions
Within each region quantize edge direction into 9 bins



Example: 320x240 image gets divided into 40x30 bins; in each bin there are 9 numbers so feature vector has 30*40*9 = 10,800 numbers

Lowe, “Object recognition from local scale-invariant features”, ICCV 1999
Dalal and Triggs, "Histograms of oriented gradients for human detection," CVPR 2005

Image Features

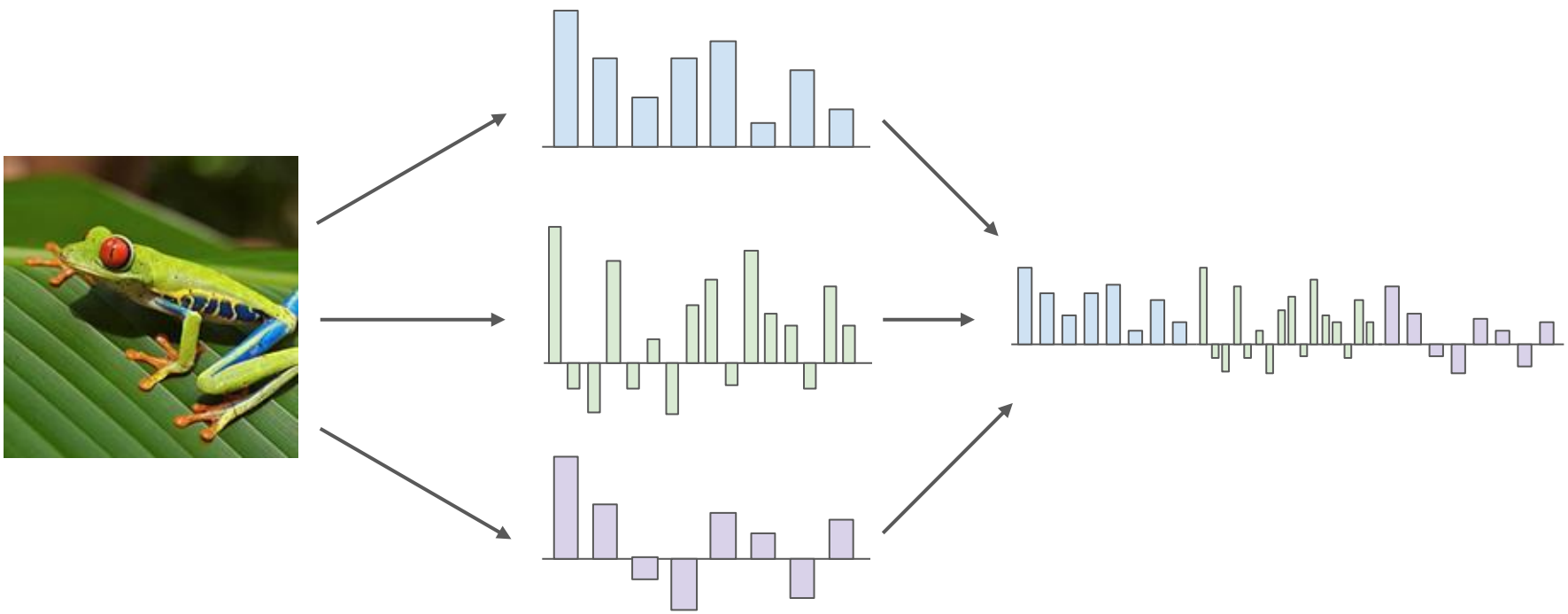


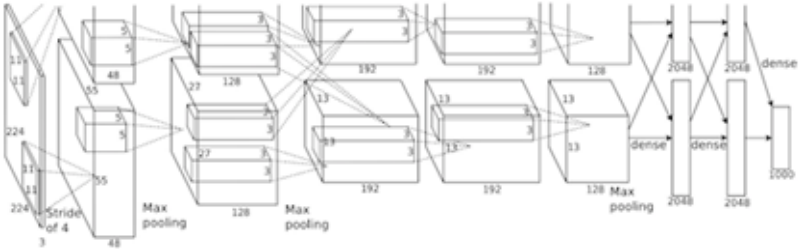
Image features vs. ConvNets

Feature Extraction

f

10 numbers giving scores for classes

training



Krizhevsky, Sutskever, and Hinton, "Imagenet classification with deep convolutional neural networks", NIPS 2012. Figure copyright Krizhevsky, Sutskever, and Hinton, 2012. Reproduced with permission.

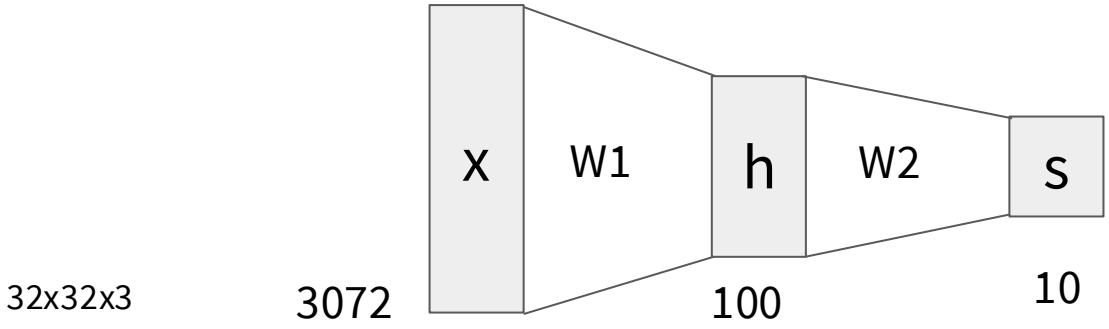
10 numbers giving scores for classes

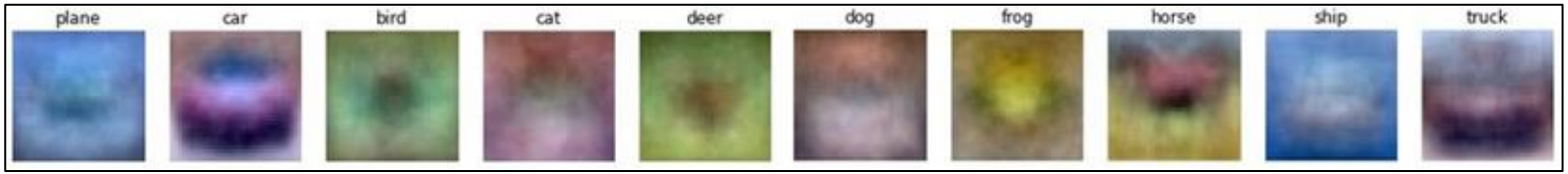
training

Last Time: Neural Networks

Linear score function: $f = Wx$

2-layer Neural Network $f = W_2 \max(0, W_1 x)$



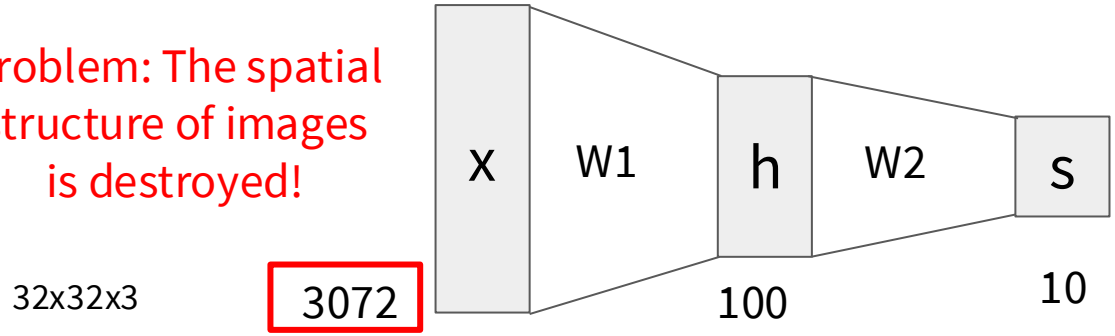


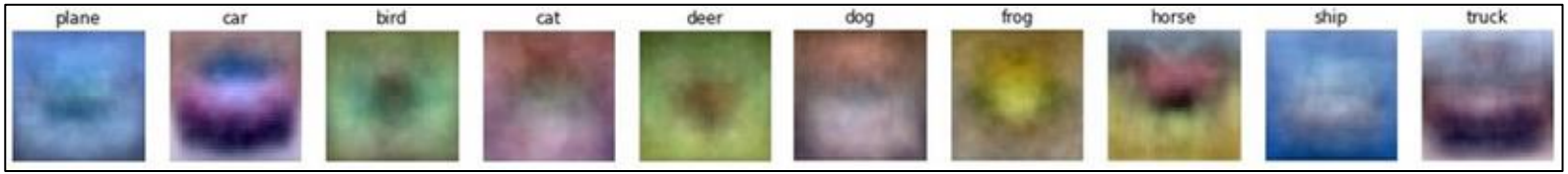
Last Time: Neural Networks

Linear score function: $f = Wx$

2-layer Neural Network $f = W_2 \max(0, W_1 x)$

Problem: The spatial structure of images is destroyed!





Next: Convolutional Neural Networks

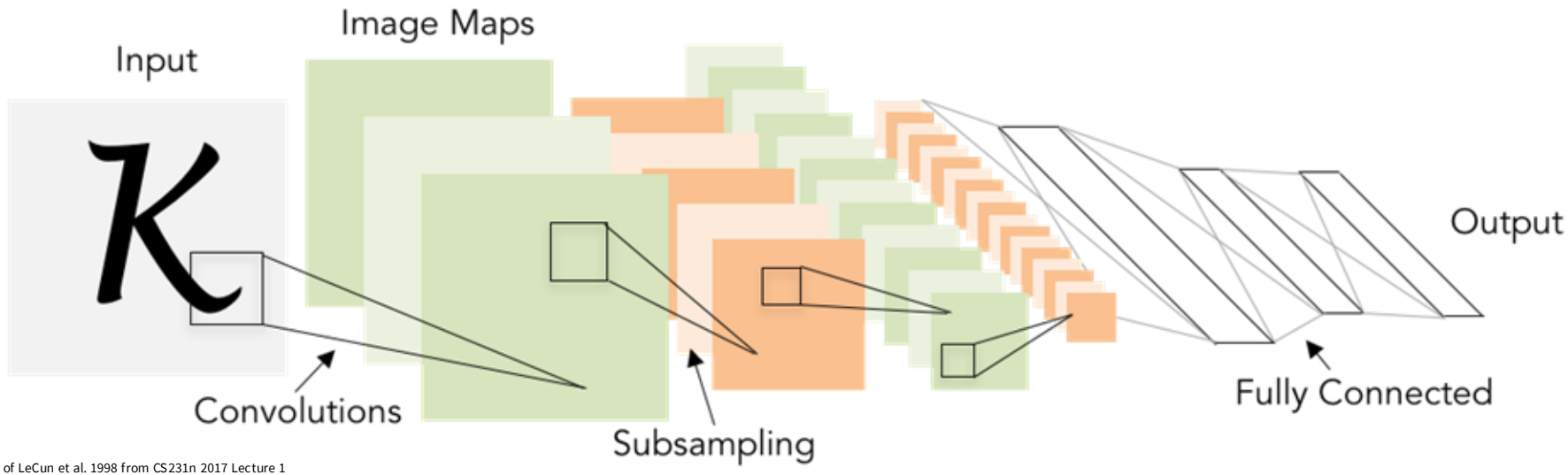


Illustration of LeCun et al. 1998 from CS231n 2017 Lecture 1

Next: Convolutional Neural Networks

Convolution and **pooling** operators extract features while respecting 2D image structure

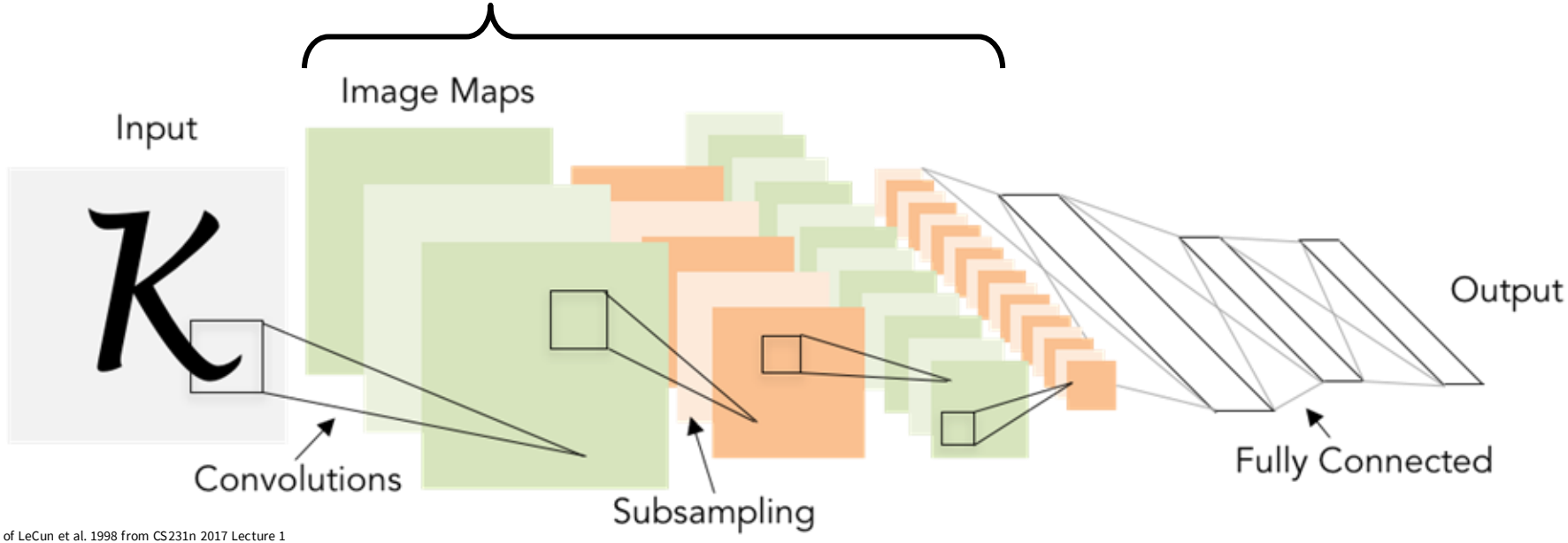


Illustration of LeCun et al. 1998 from CS231n 2017 Lecture 1

Next: Convolutional Neural Networks

Convolution and **pooling** operators extract features while respecting 2D image structure

Fully-Connected layers form an MLP at the end to predict scores

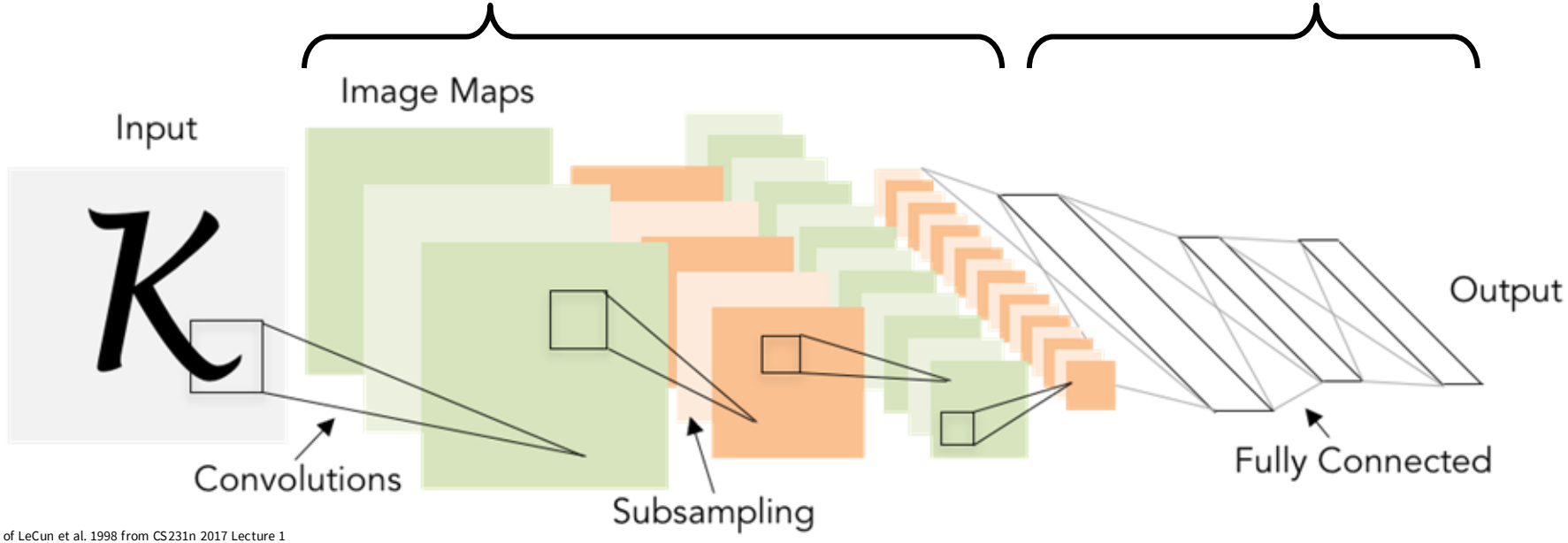


Illustration of LeCun et al. 1998 from CS231n 2017 Lecture 1

Next: Convolutional Neural Networks

Trained end-to-end with backprop + gradient descent

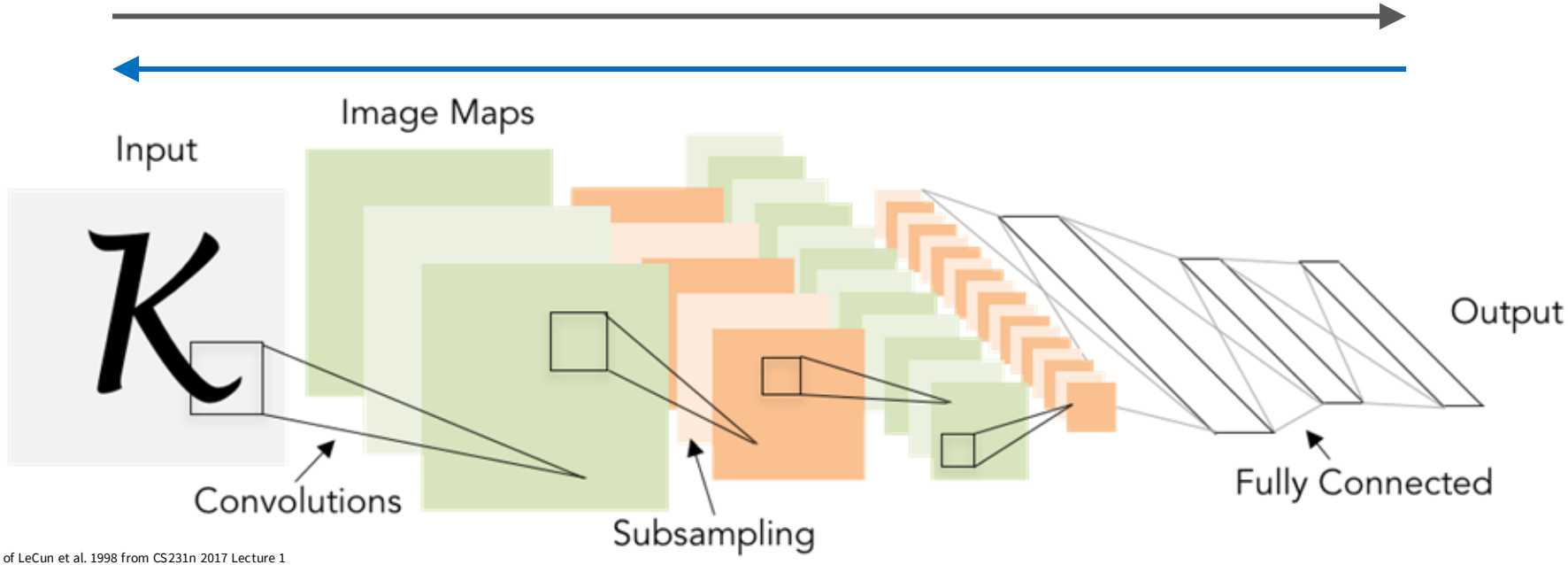
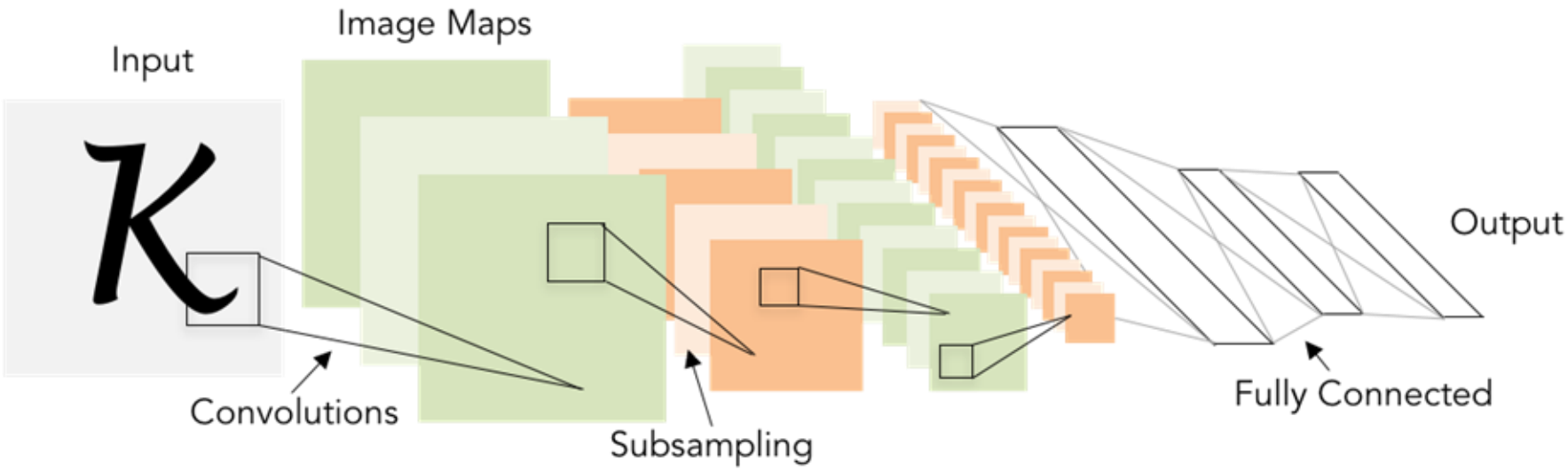


Illustration of LeCun et al. 1998 from CS231n 2017 Lecture 1

A bit of history:

Gradient-based learning applied to document recognition
[LeCun, Bottou, Bengio, Haffner 1998]



A bit of history:

ImageNet Classification with Deep Convolutional Neural Networks [Krizhevsky, Sutskever, Hinton, 2012]

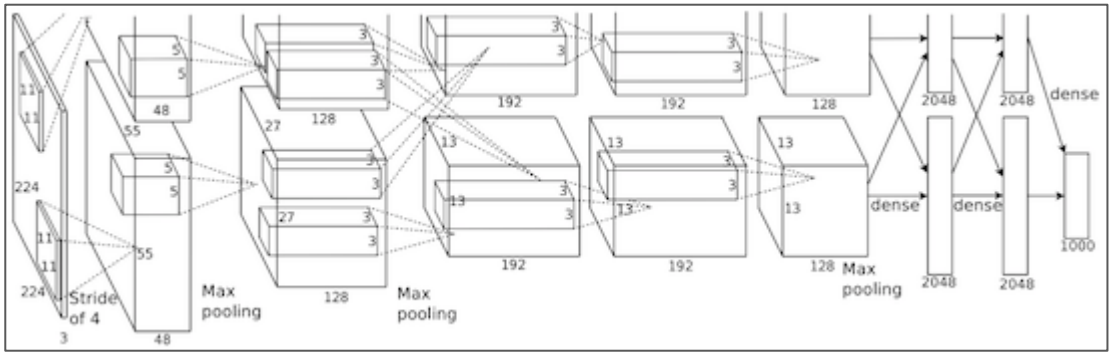
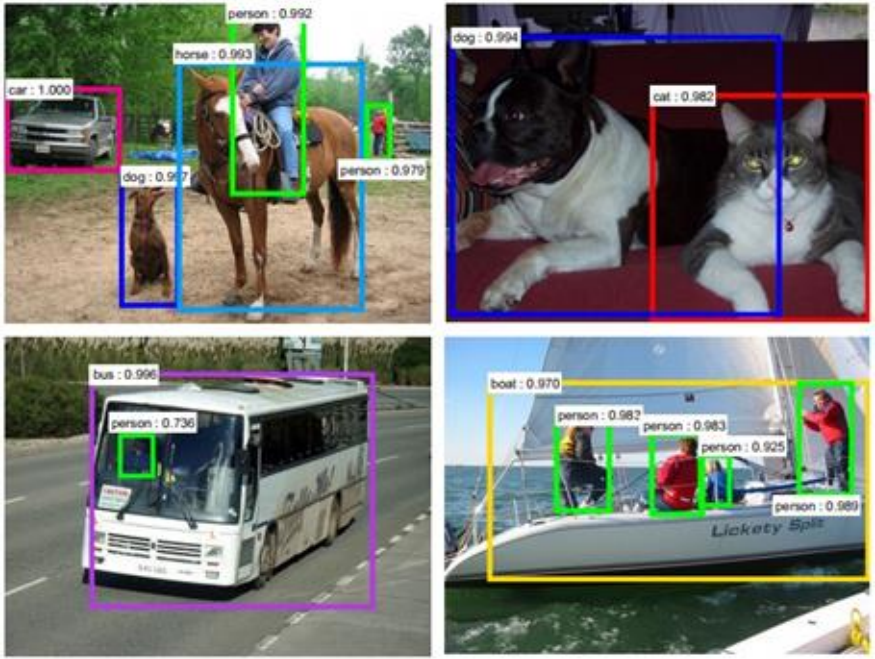


Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

"AlexNet"

~2012 – 2020: ConvNets dominate all vision tasks

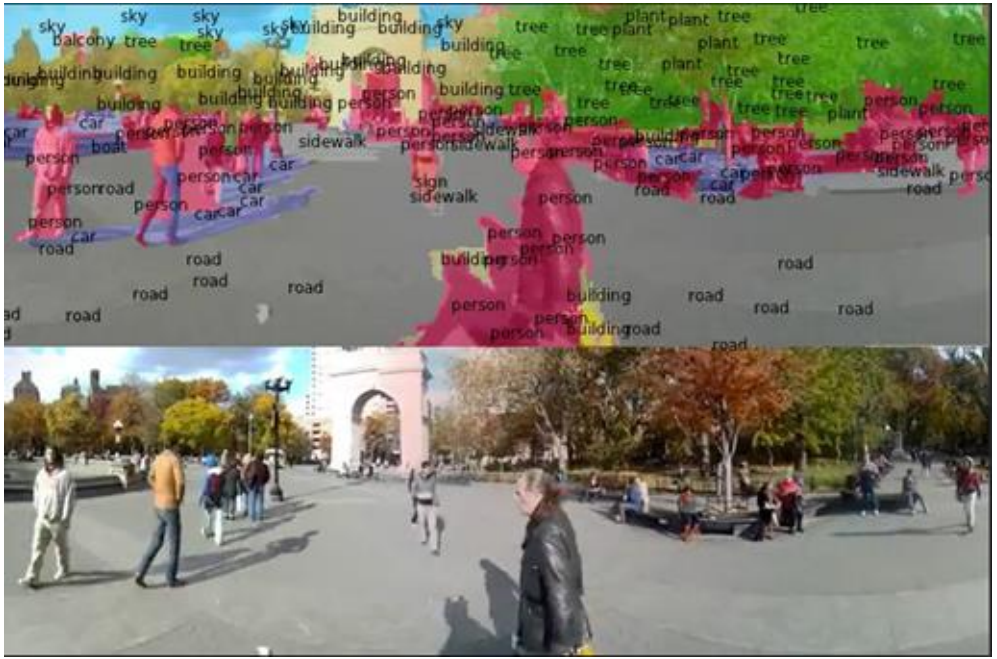
Detection



Figures copyright Shaoqing Ren, Kaiming He, Ross Girschick, Jian Sun, 2015. Reproduced with permission.

[Faster R-CNN: Ren, He, Girshick, Sun 2015]

Segmentation



Figures copyright Clement Farabet, 2012. Reproduced with permission.

[Farabet et al., 2012]

~2012 – 2020: ConvNets dominate all vision tasks

Image Captioning

A white teddy bear sitting in the grass

A man in a baseball uniform throwing a ball

A woman is holding a cat in her hand

A man riding a wave on top of a surfboard

A cat sitting on a suitcase on the floor

A woman standing on a beach holding a surfboard

All images are CC0 Public domain:
https://pixabay.com/en/luggage-antique-cat-1643010/
https://pixabay.com/en/teddy-plush-bears-cute-teddy-bear-1623436/
https://pixabay.com/en/surf-wave-summer-sport-litoral-1668716/
https://pixabay.com/en/woman-female-model-portrait-adult-983967/
https://pixabay.com/en/handstand-lake-meditation-496008/
https://pixabay.com/en/baseball-player-shortstop-infield-1045263/

Captions generated by Justin Johnson using Neuraltalk2

[Vinyals et al., 2015]
[Karpathy and Fei-Fei, 2015]

~2012 – 2020: ConvNets dominate all vision tasks

Text-to-Image Generation

Rombach et al, "High-Resolution Image Synthesis with Latent Diffusion Models", CVPR 2022

A zombie in the style of Picasso

An image of a half mouse half octopus

A painting of a squirrel eating a burger

A watercolor painting of a chair that looks like an octopus

A shirt with the inscription: "I love generative models!"

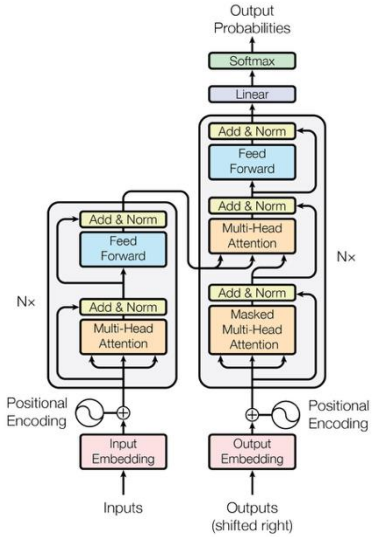
~2012 – 2020: ConvNets dominate all vision tasks

CS231n: Convolutional Neural Networks for Visual Recognition

This class used to be focused on ConvNets!

2021 - Present: Transformers have taken over

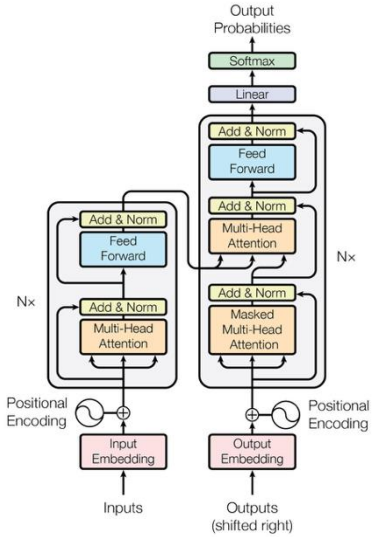
2017: Transformers for language tasks



Vaswani et al, “Attention is all you need”, NeurIPS 2017

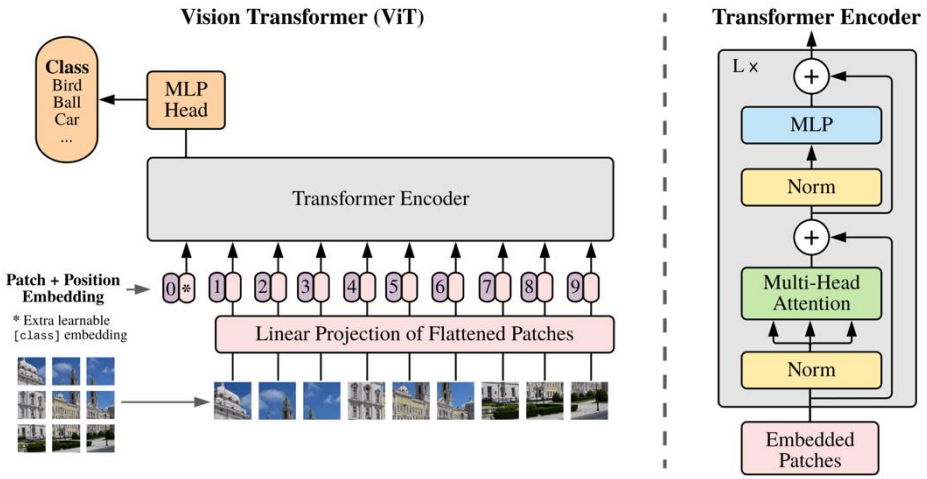
2021 - Present: Transformers have taken over

2017: Transformers for language tasks



Vaswani et al, "Attention is all you need", NeurIPS 2017

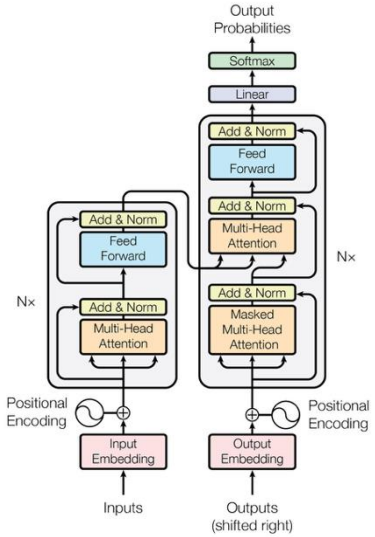
2021: Transformers for vision tasks



Dosovitskiy et al, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale", ICLR 2021

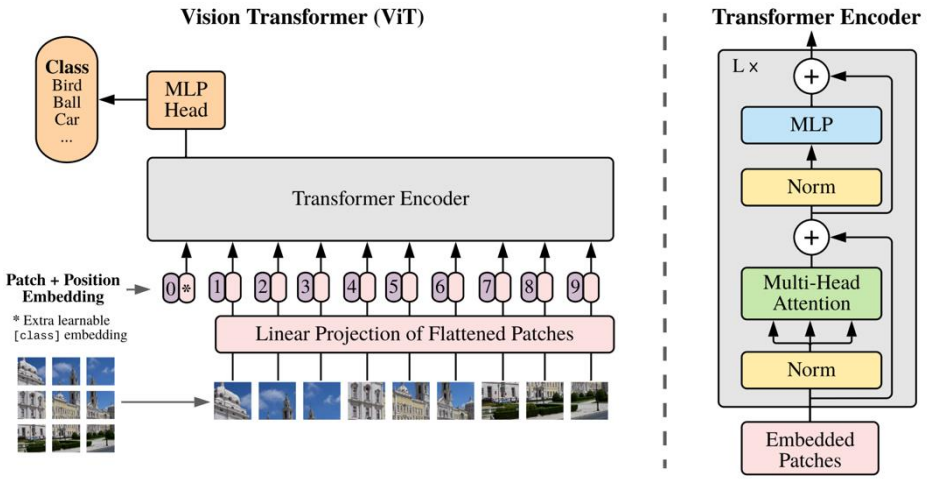
2021 - Present: Transformers have taken over

2017: Transformers for language tasks



Vaswani et al, "Attention is all you need", NeurIPS 2017

2021: Transformers for vision tasks



Dosovitskiy et al, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale", ICLR 2021

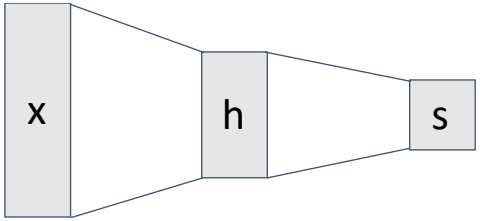
Wait until Lecture 8!

Convolutional Neural Networks

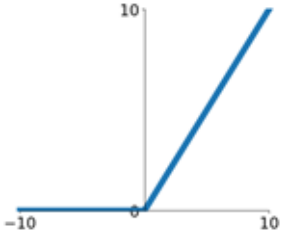
Today: Convolutional Networks

Fully-Connected Layer

We have already seen these



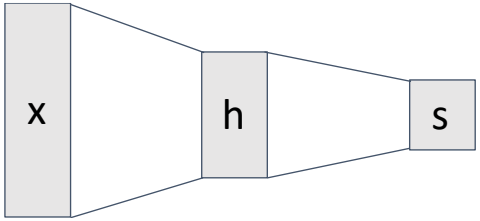
Activation Function



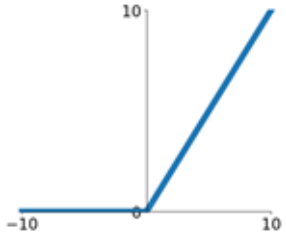
Today: Convolutional Networks

We have already seen these

Fully-Connected Layer

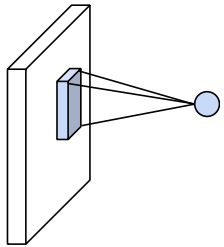


Activation Function

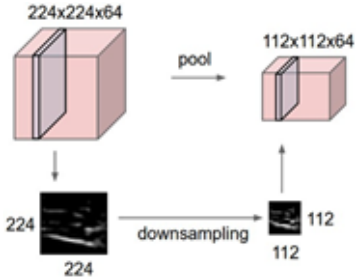


Today: Image-specific operators

Convolution Layer



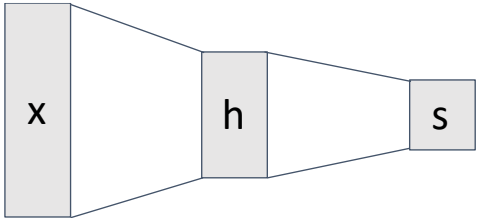
Pooling Layer



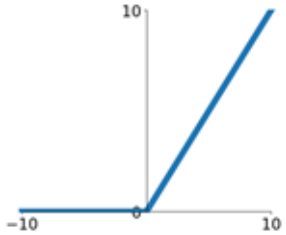
Today: Convolutional Networks

We have already seen these

Fully-Connected Layer

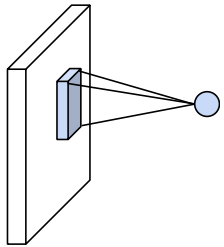


Activation Function

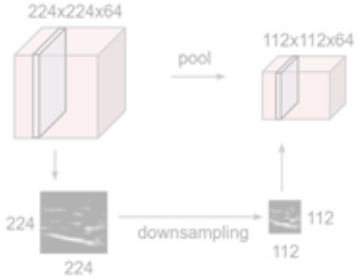


Today: Image-specific operators

Convolution Layer

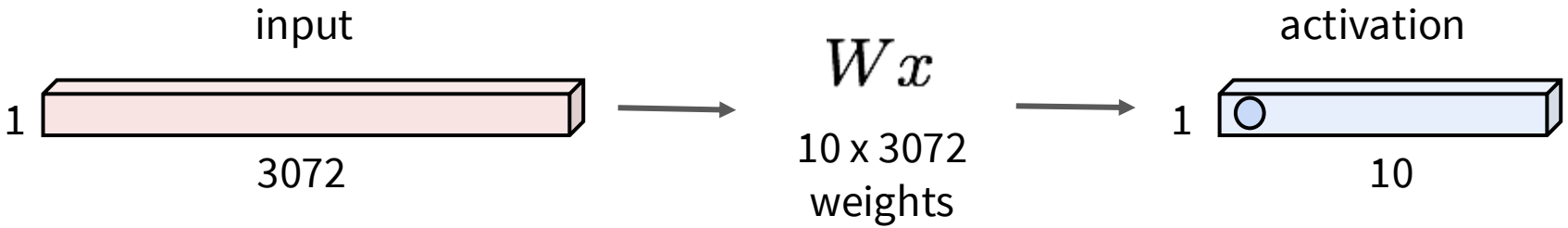


Pooling Layer



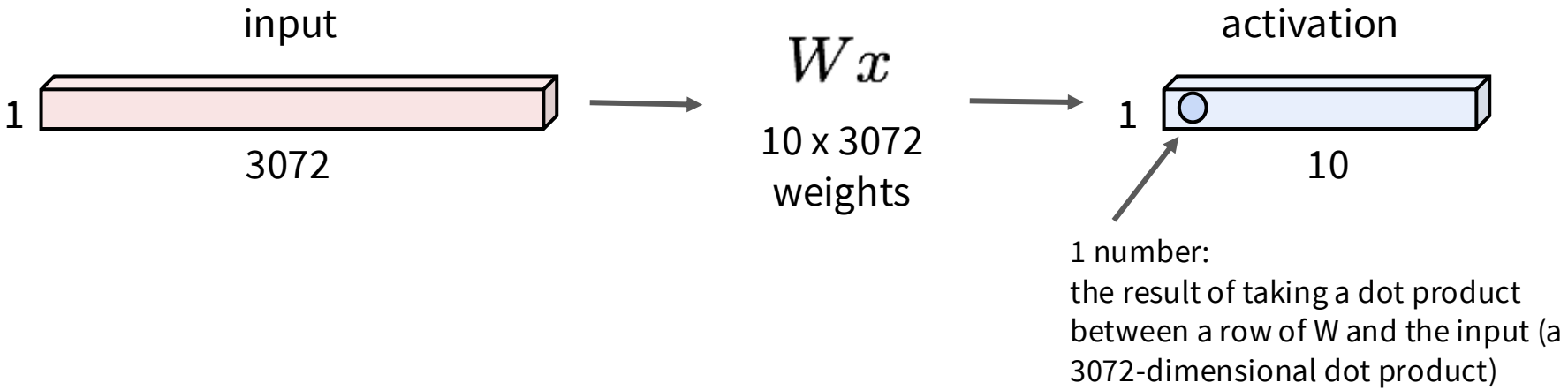
Recap: Fully Connected Layer

32x32x3 image -> stretch to 3072 x 1



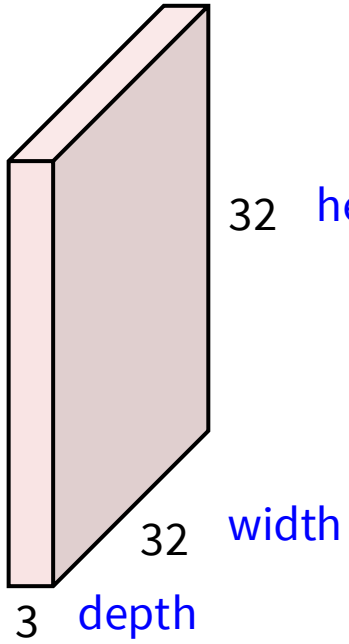
Fully Connected Layer

32x32x3 image -> stretch to 3072 x 1



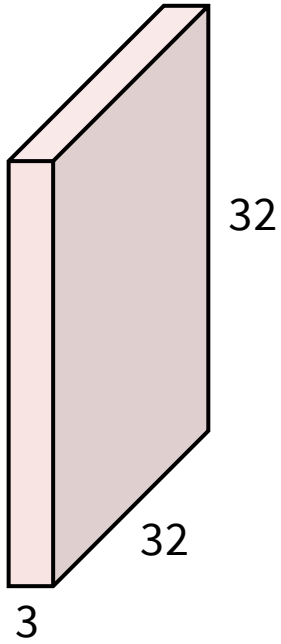
Convolution Layer

32x32x3 image -> preserve spatial structure

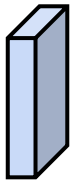


Convolution Layer

32x32x3 image



5x5x3 filter

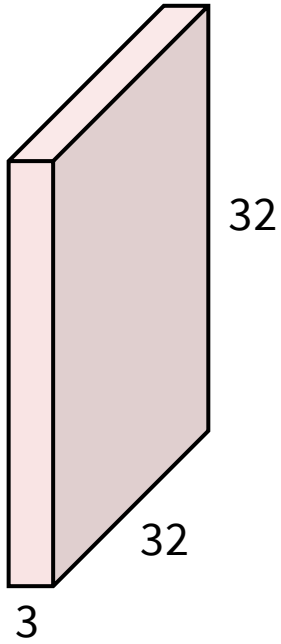


Convolve the filter with the image
i.e. “slide over the image spatially,
computing dot products”

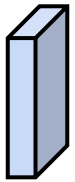
Convolution Layer

Filters always extend the full depth of the input volume

32x32x3 image

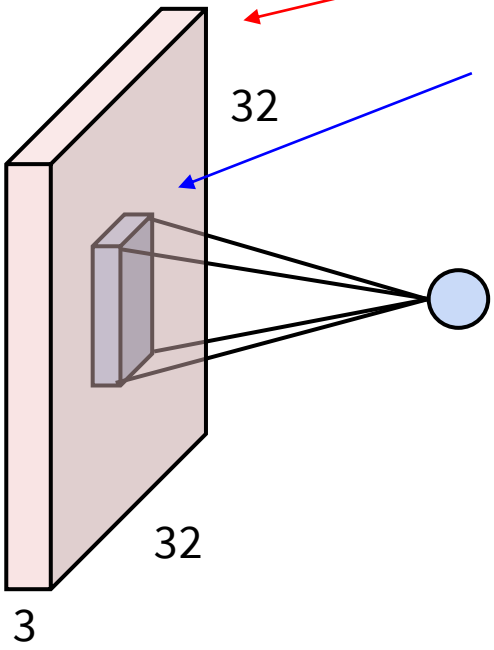


5x5x3 filter



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

Convolution Layer



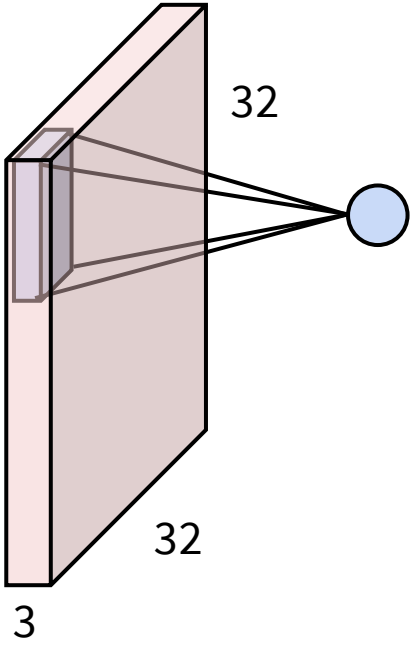
32x32x3 image

5x5x3 filter w

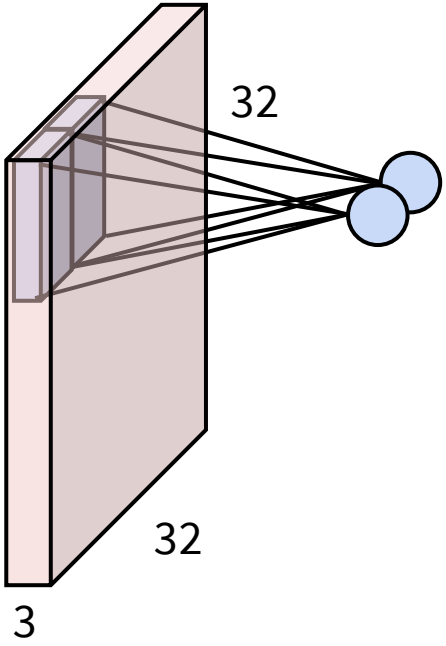
1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)

$$w^T x + b$$

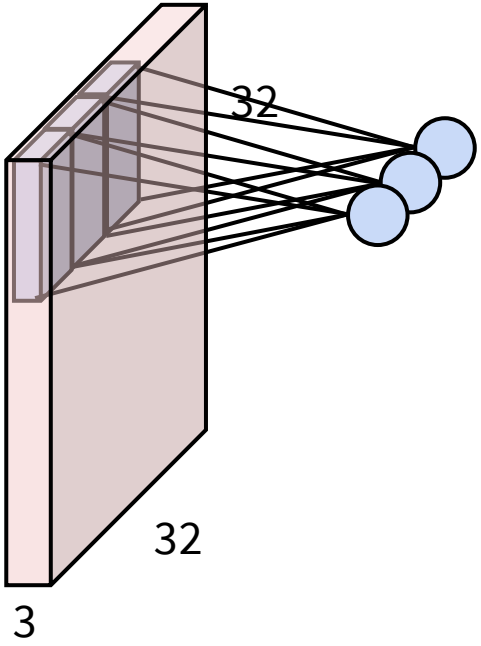
Convolution Layer



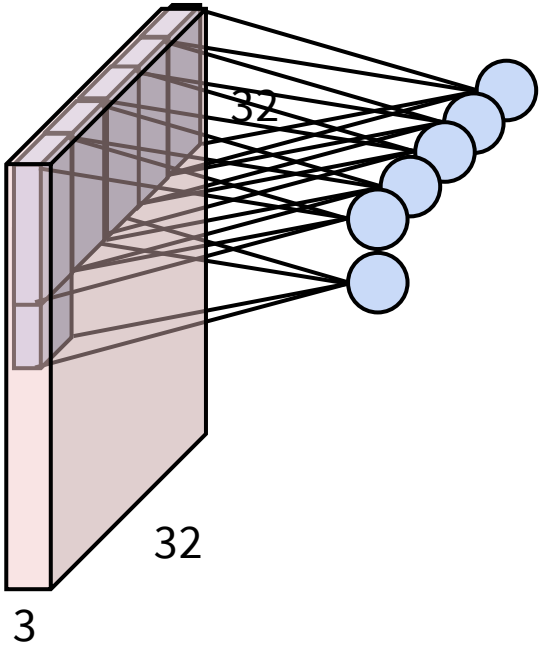
Convolution Layer



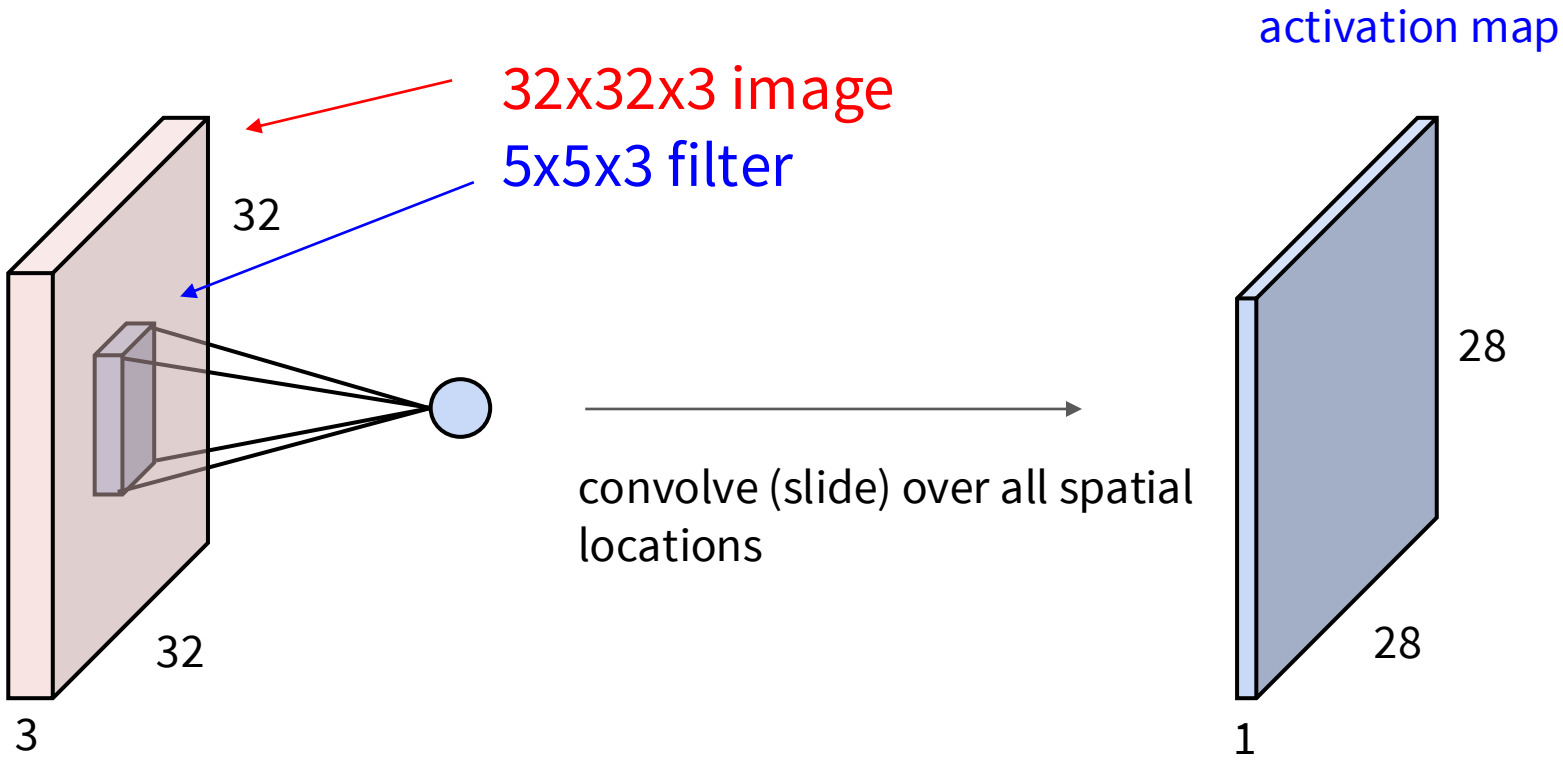
Convolution Layer



Convolution Layer

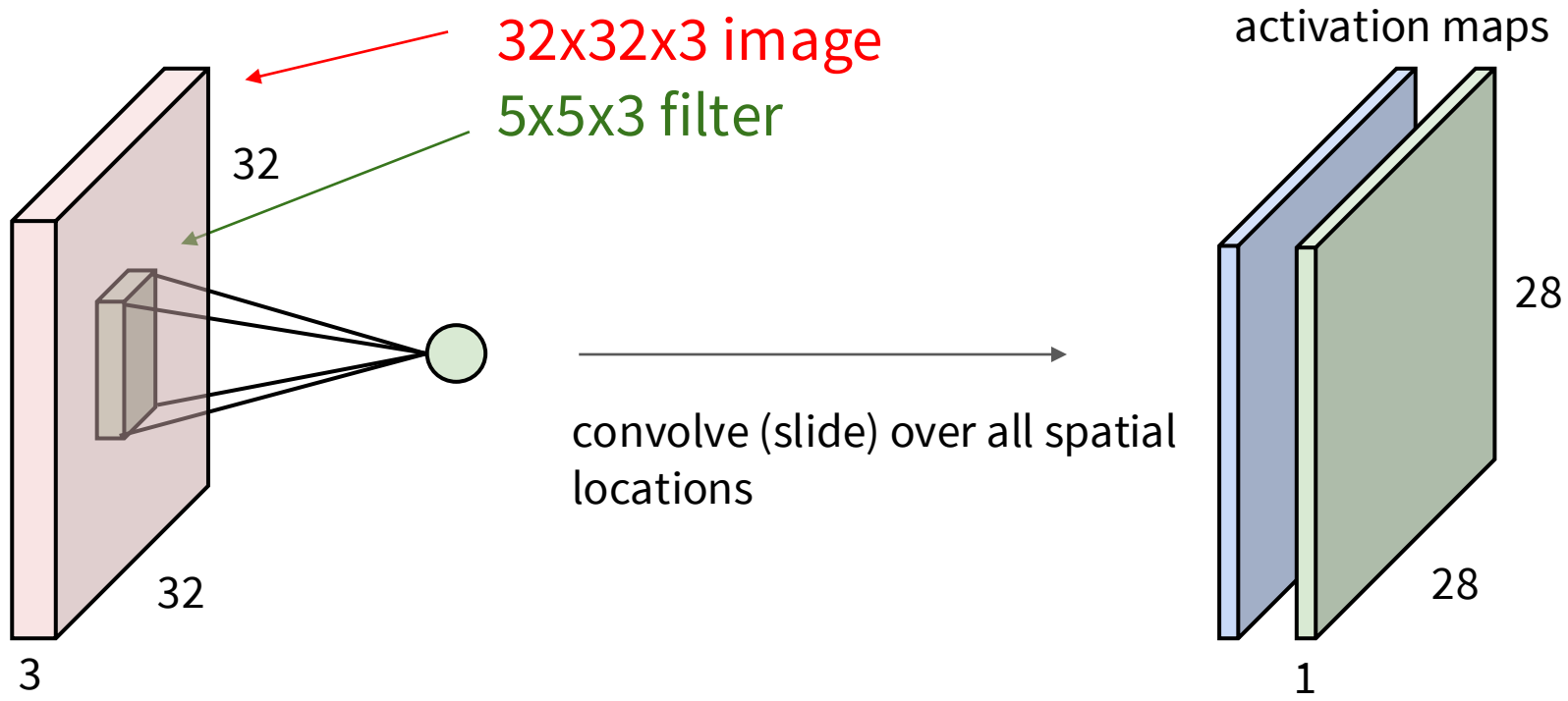


Convolution Layer

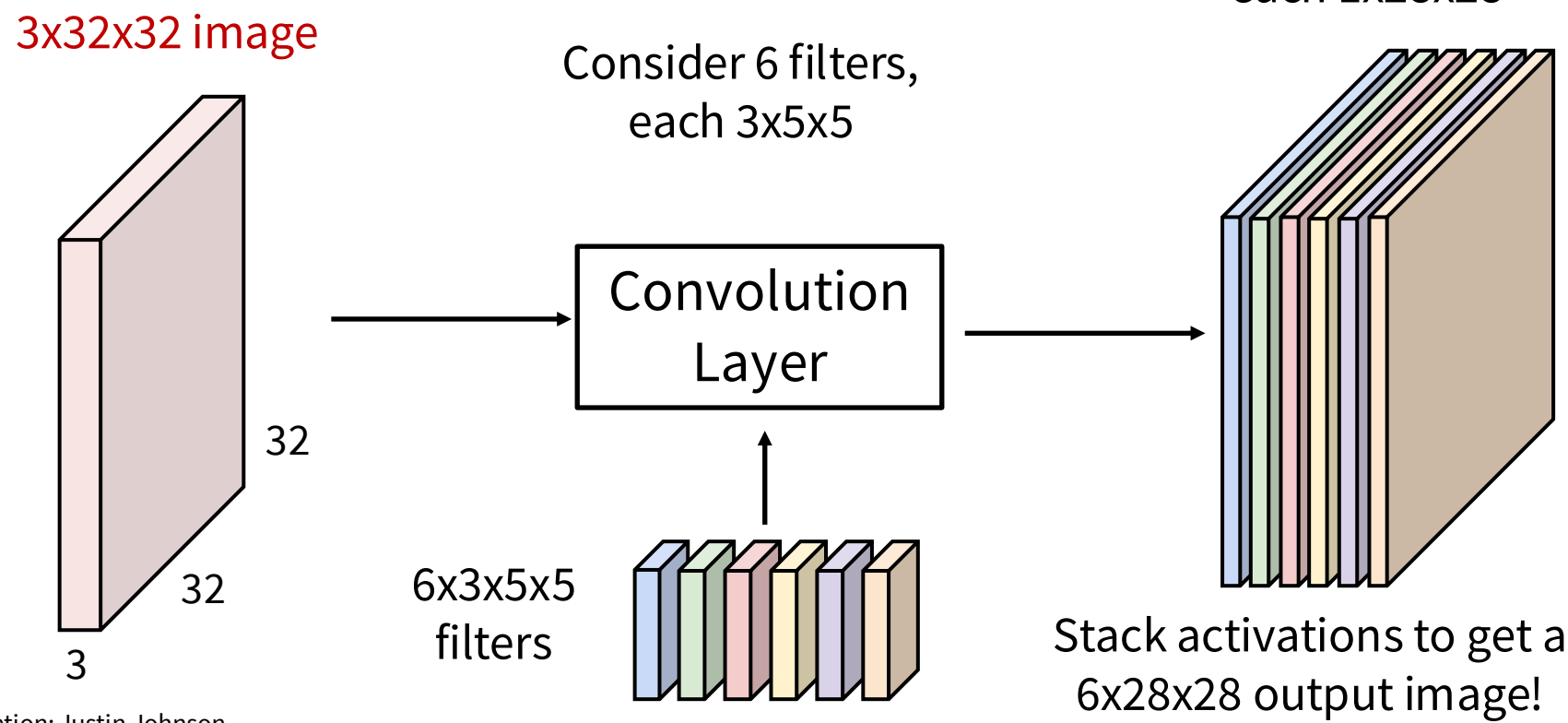


Convolution Layer

consider a second, green filter

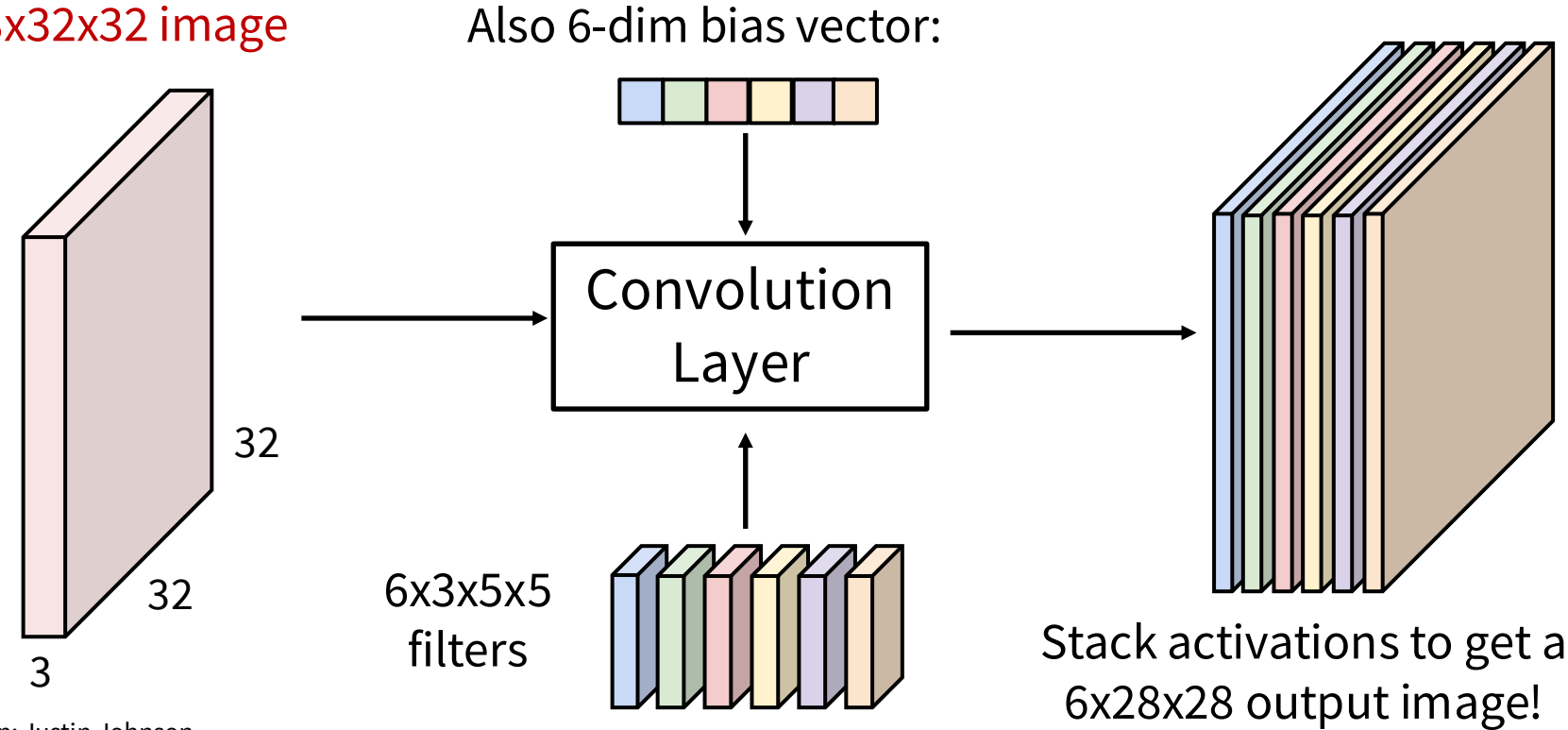


Convolution Layer



Slide inspiration: Justin Johnson

Convolution Layer



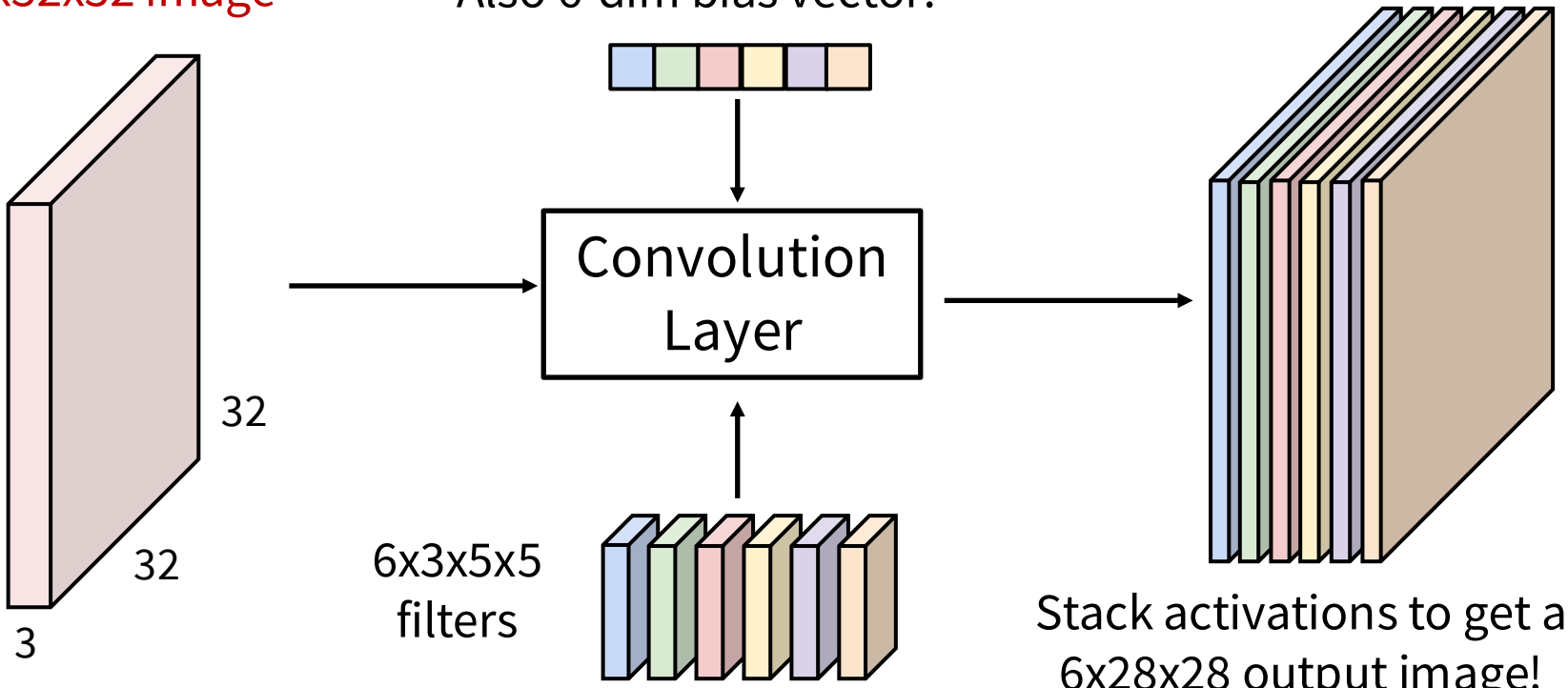
Slide inspiration: Justin Johnson

Convolution Layer

28x28 grid, at each point a 6-dim vector

3x32x32 image

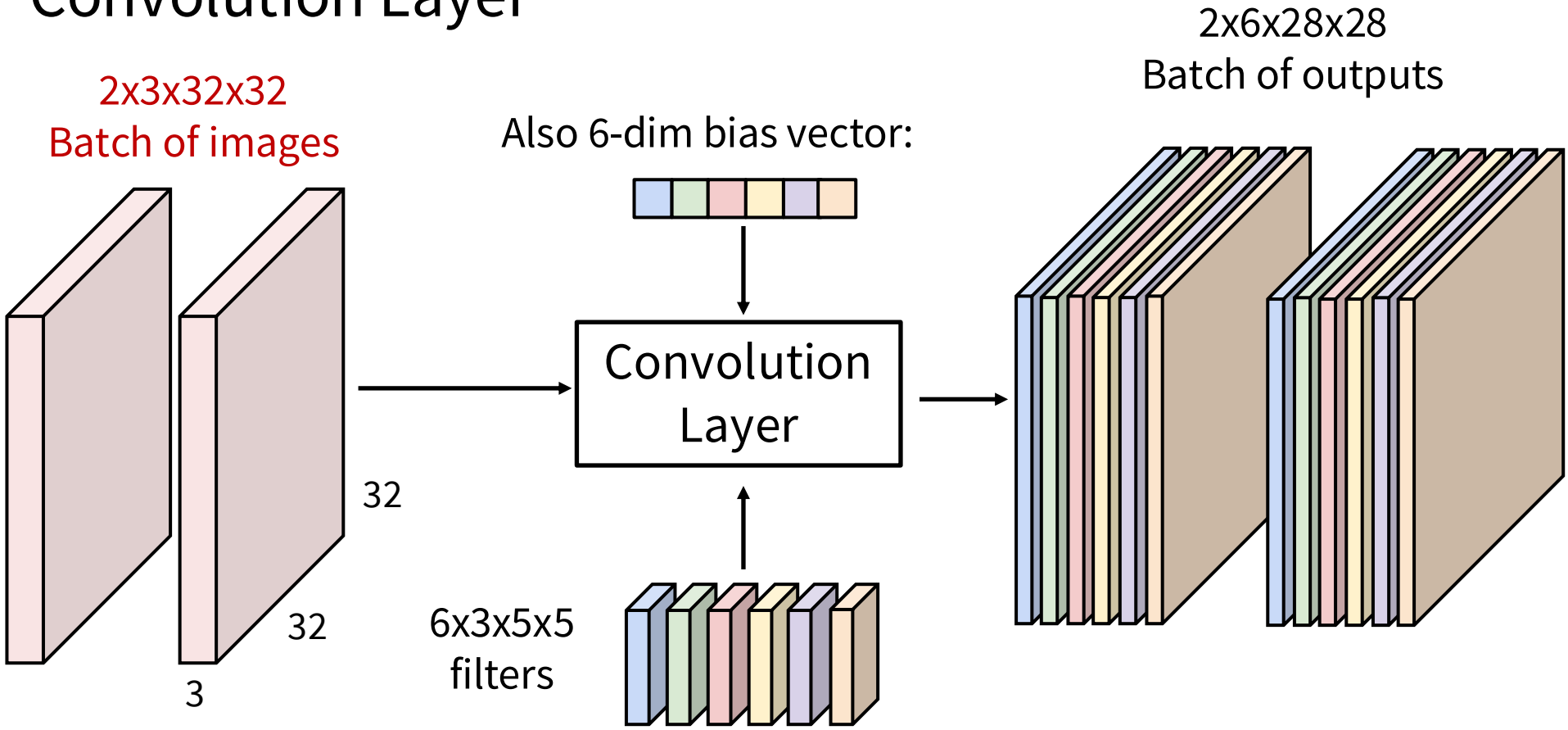
Also 6-dim bias vector:



Stack activations to get a 6x28x28 output image!

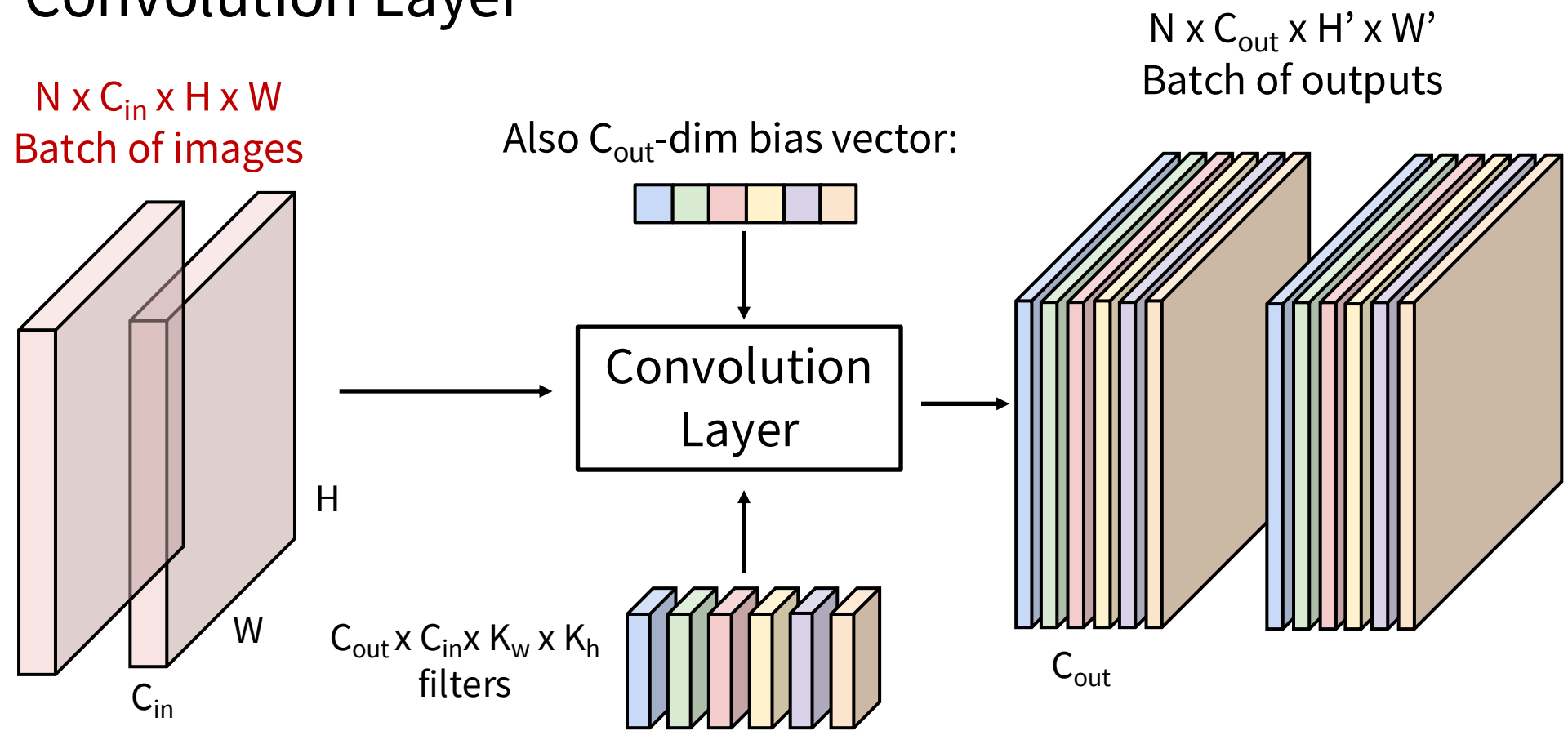
Slide inspiration: Justin Johnson

Convolution Layer



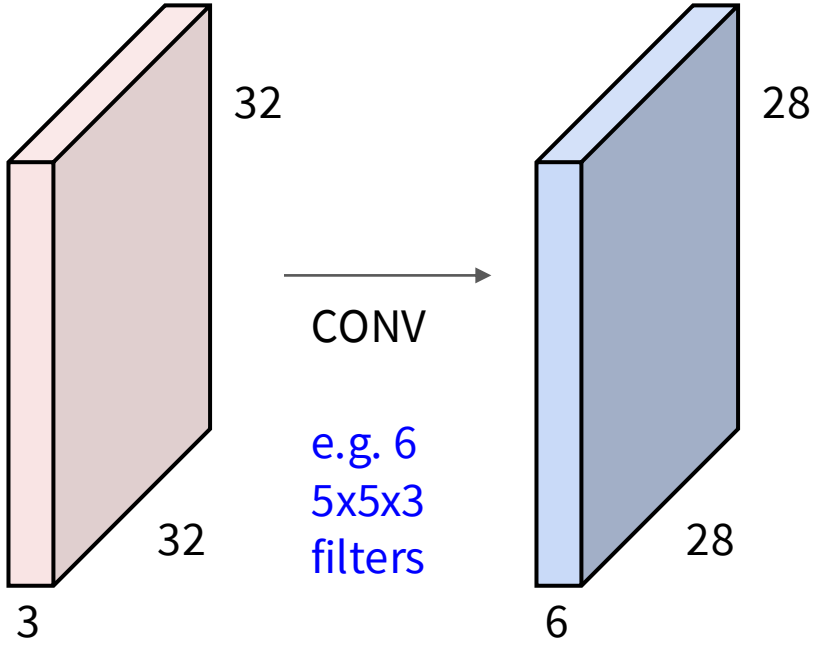
Slide inspiration: Justin Johnson

Convolution Layer

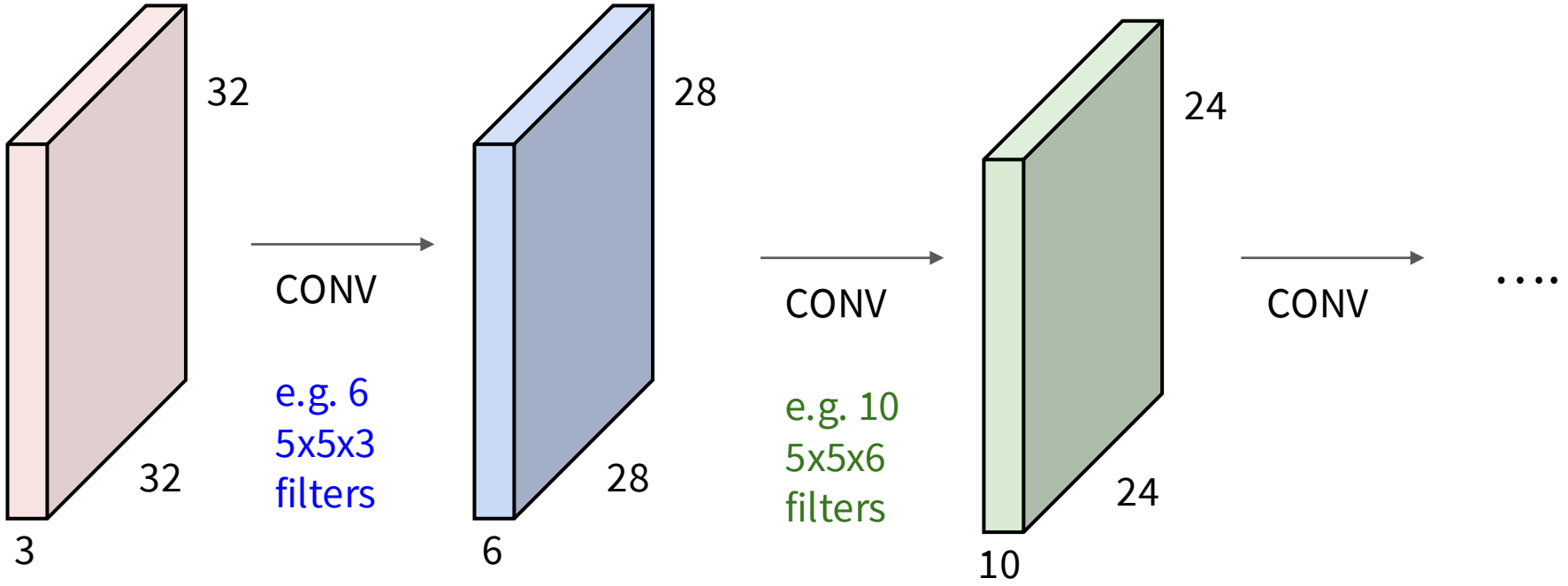


Slide inspiration: Justin Johnson

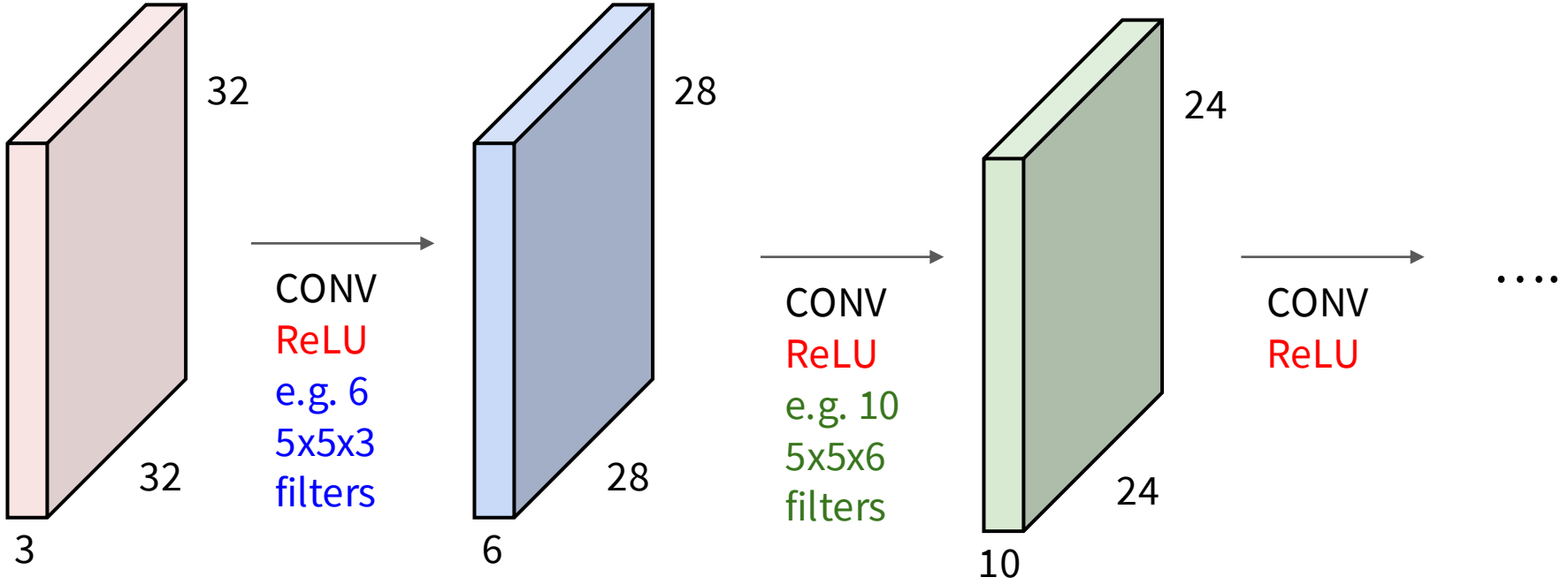
A ConvNet is a neural network with Conv layers



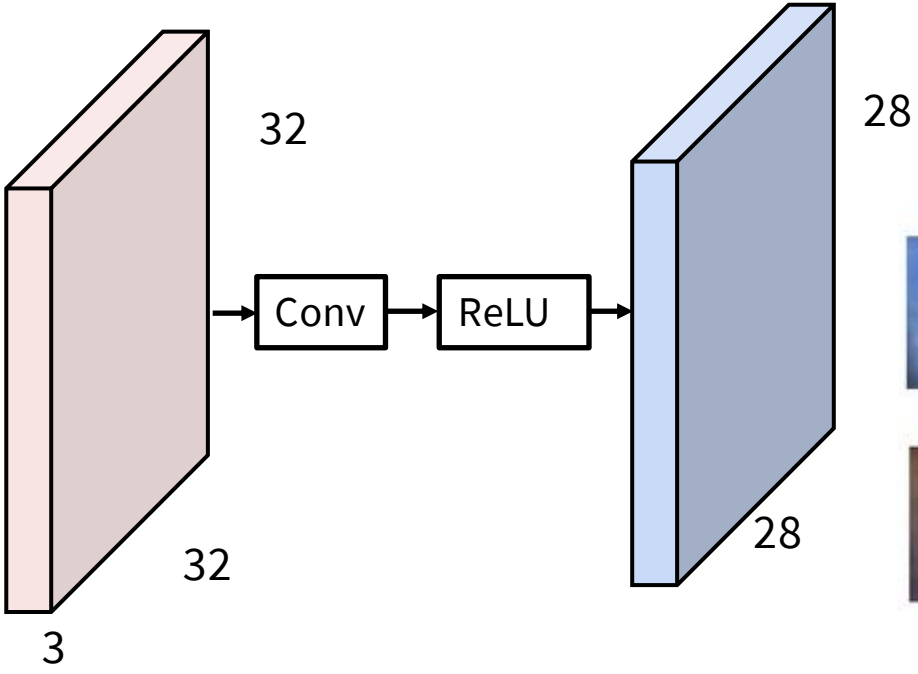
A ConvNet is a neural network with Conv layers



A ConvNet is a neural network with Conv layers with activation functions!

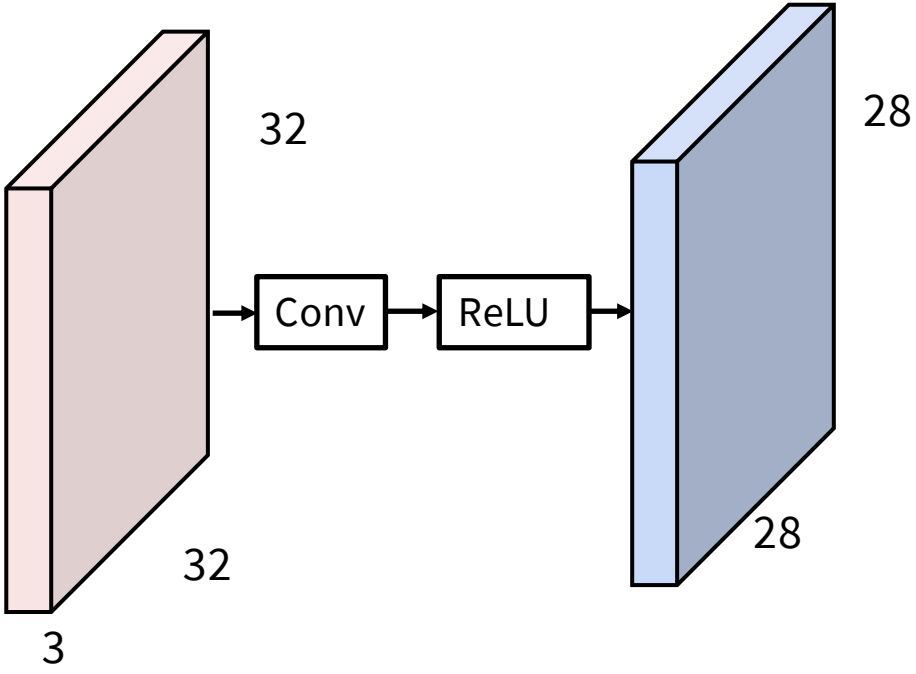


What do Conv filters learn?



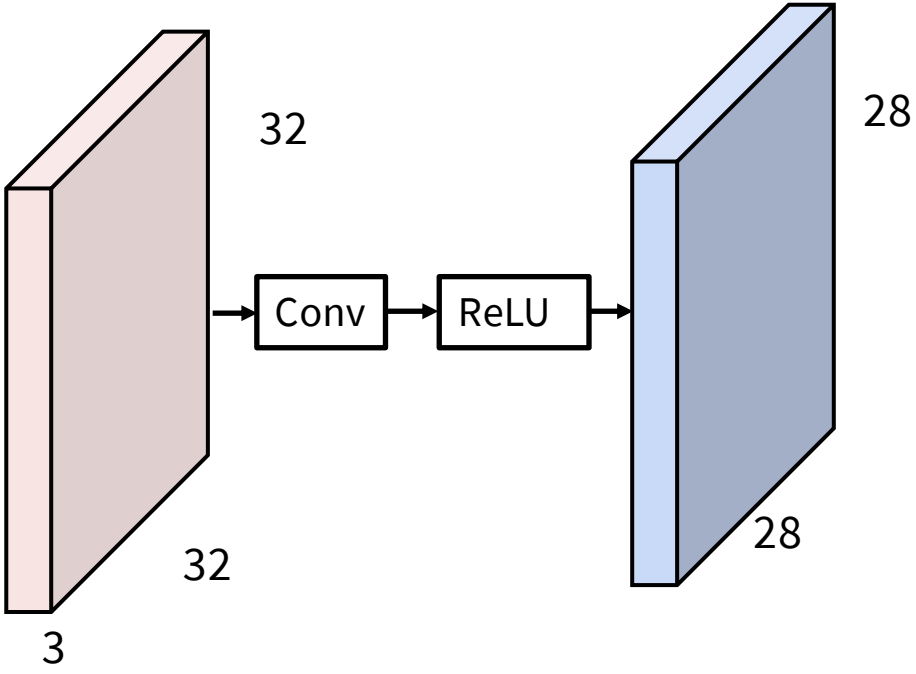
Linear classifier: One template per class

What do Conv filters learn?

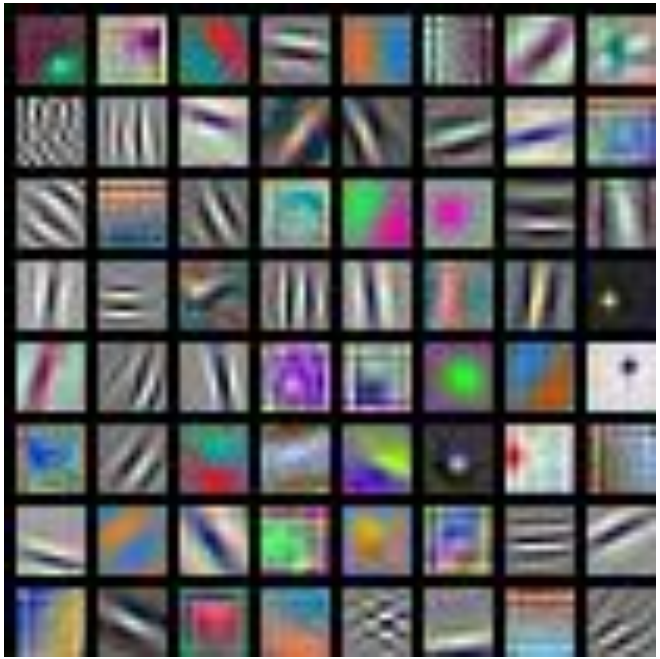


MLP: Bank of whole-image templates

What do Conv filters learn?

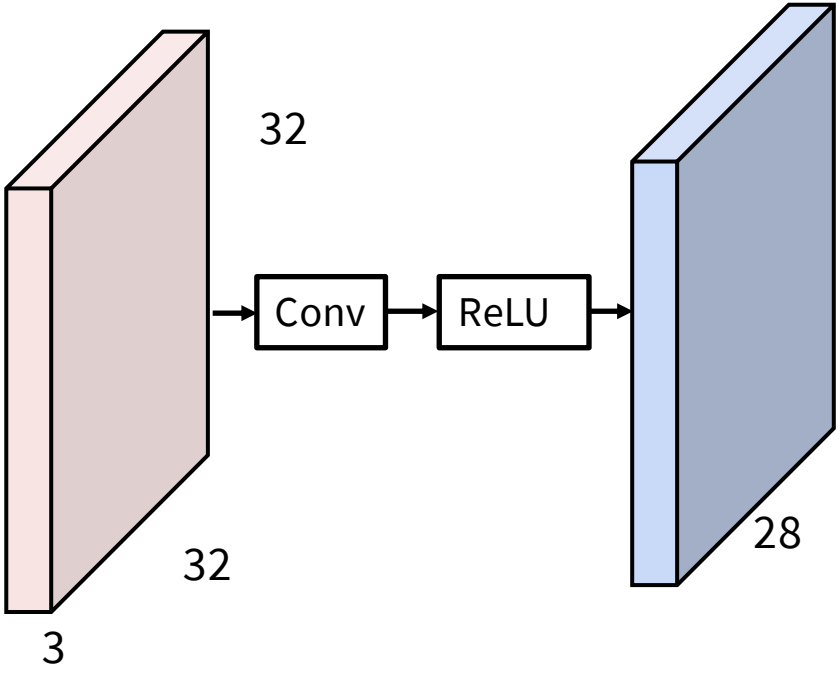


First-layer conv filters: local image templates
(Often learns oriented edges, opposing colors)

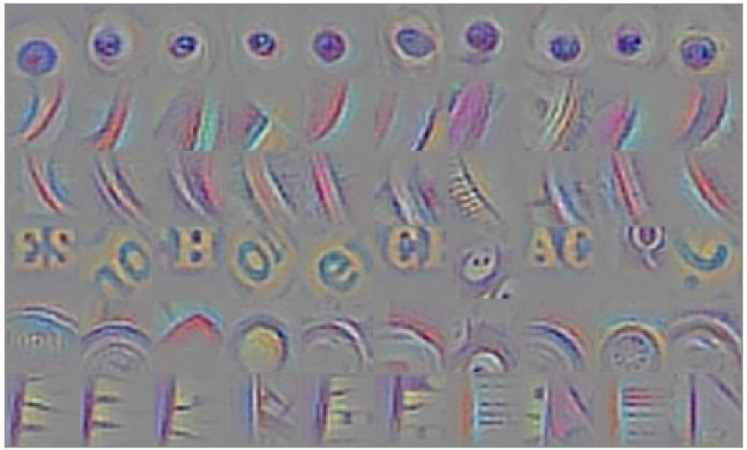


AlexNet: 64 filters, each 3x11x11

What do Conv filters learn?



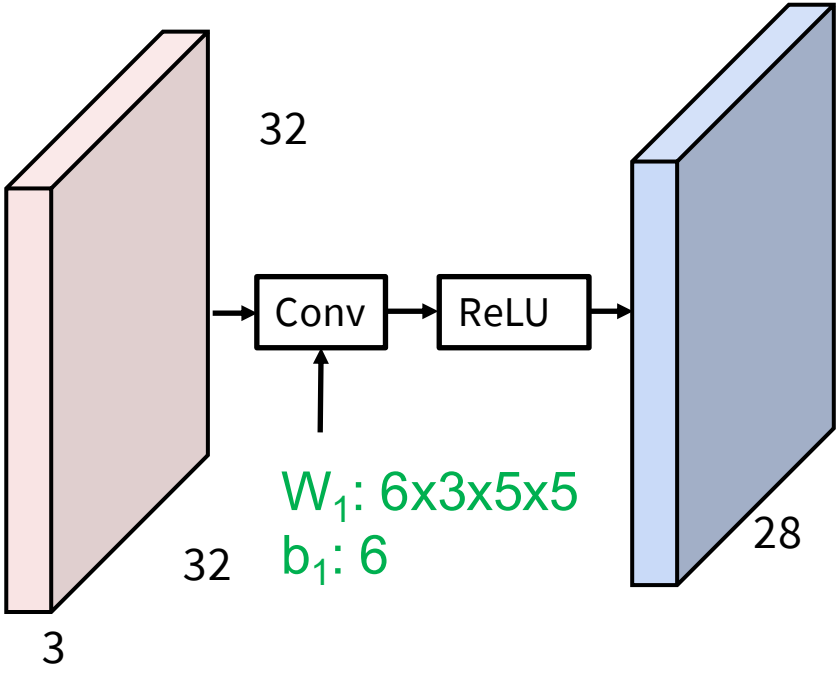
Deeper conv layers: Harder to visualize
Tend to learn larger structures e.g. eyes, letters



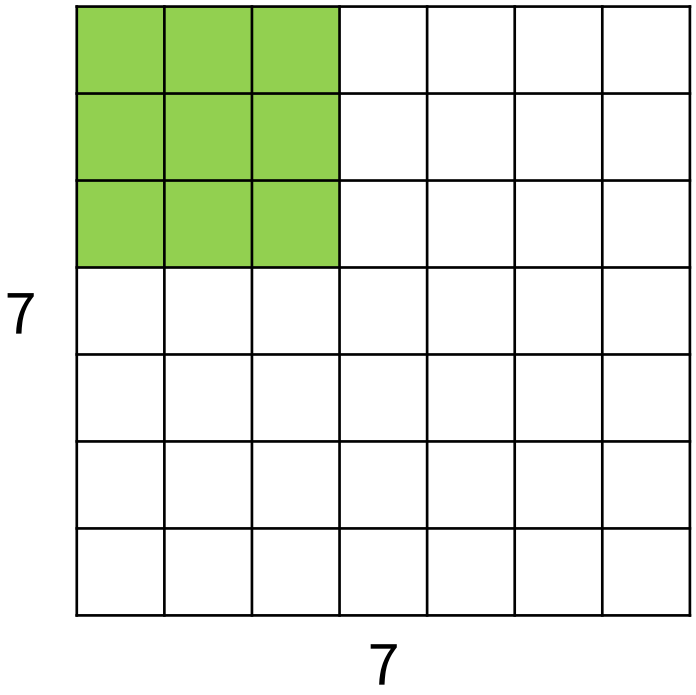
6th layer conv layer from an ImageNet model

Visualization from [Springenberg et al, ICLR 2015]

Convolution: Spatial Dimensions

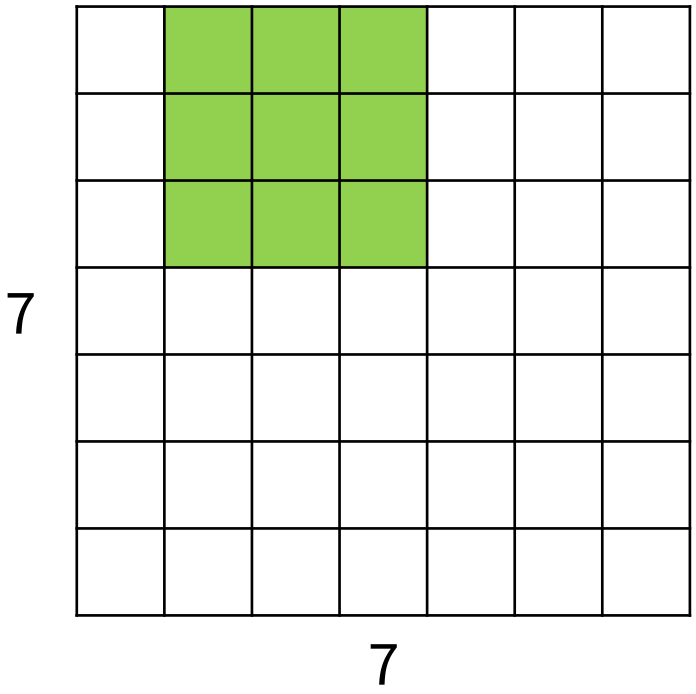


Convolution: Spatial Dimensions



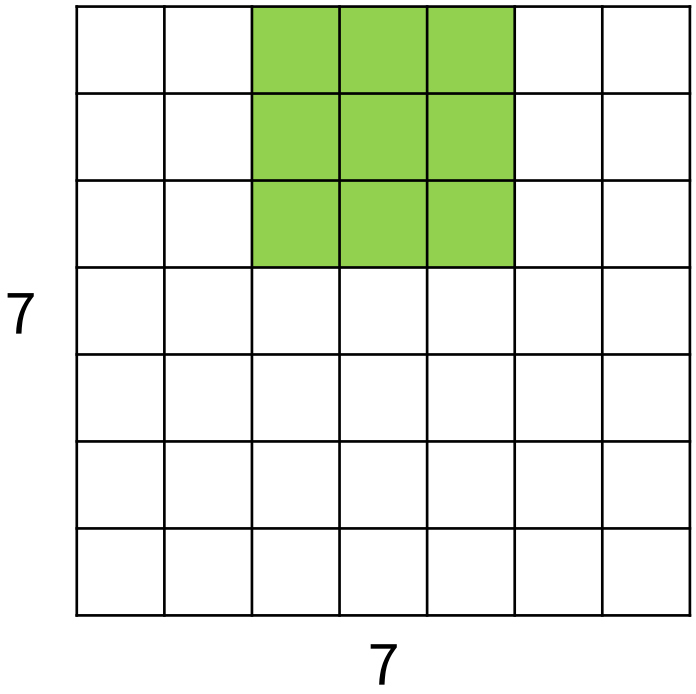
Input: 7x7
Filter: 3x3

Convolution: Spatial Dimensions



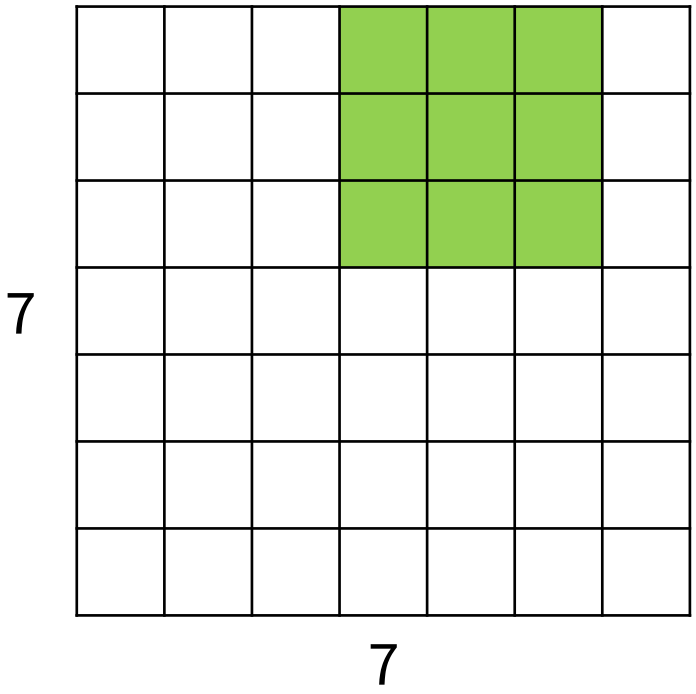
Input: 7x7
Filter: 3x3

Convolution: Spatial Dimensions



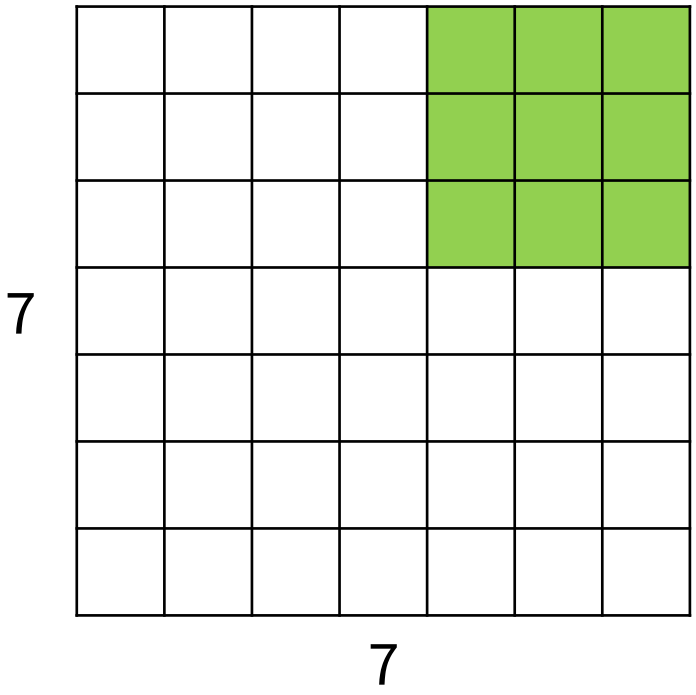
Input: 7x7
Filter: 3x3

Convolution: Spatial Dimensions



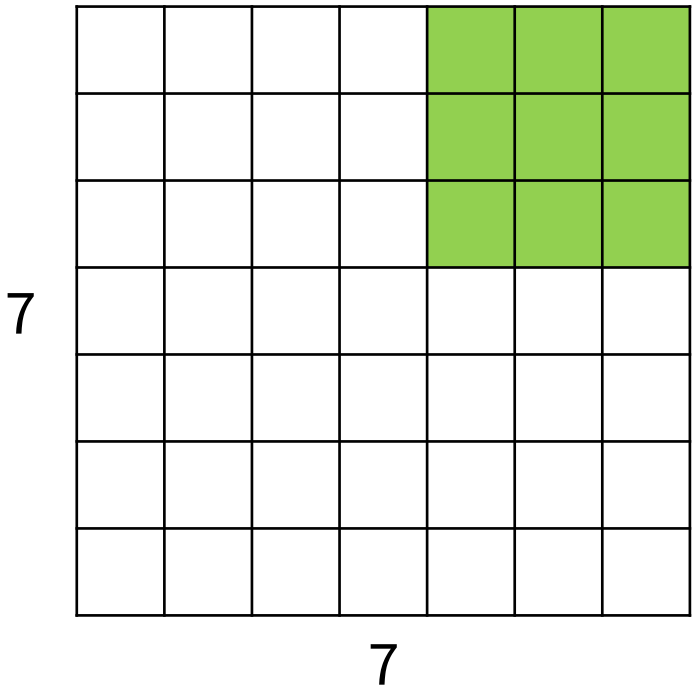
Input: 7x7
Filter: 3x3

Convolution: Spatial Dimensions



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

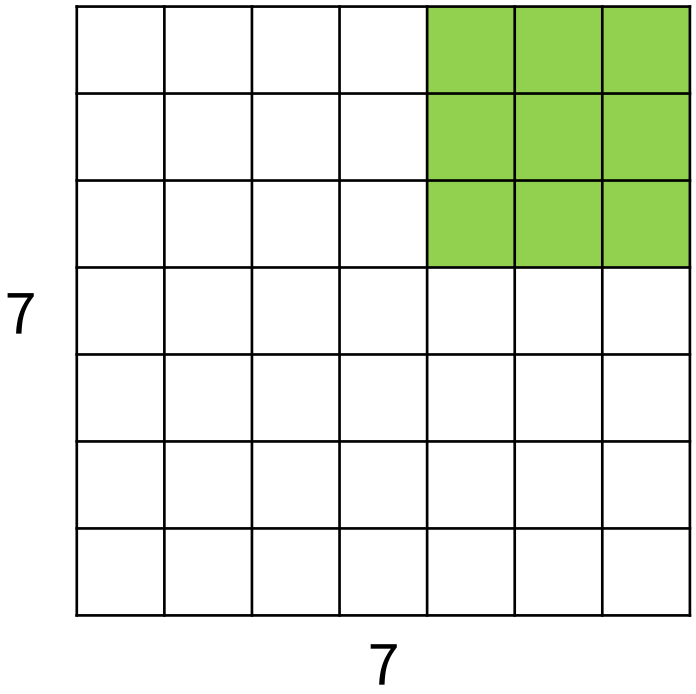
Convolution: Spatial Dimensions



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

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

Convolution: Spatial Dimensions

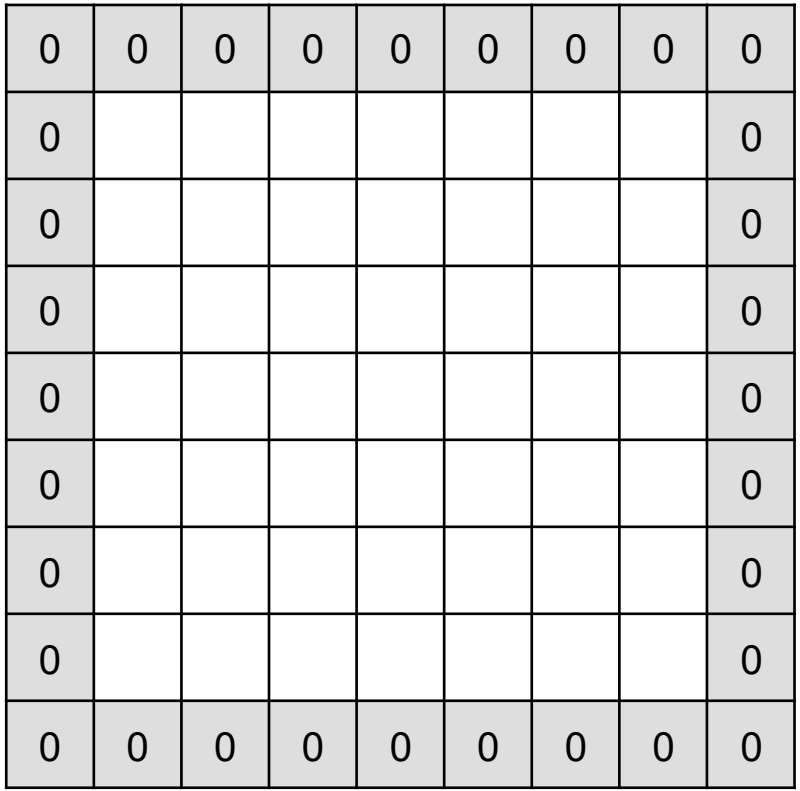


Input: 7x7
Filter: 3x3
Output: 5x5

Problem: Feature maps shrink with each layer!

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

Convolution: Spatial Dimensions



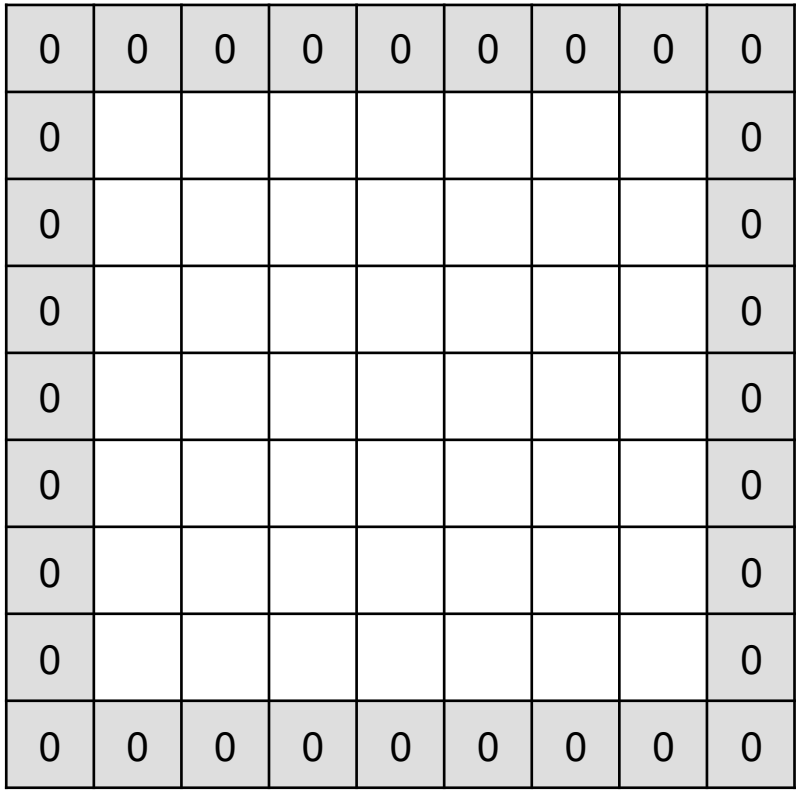
Input: 7x7
Filter: 3x3
Output: 5x5

Problem: Feature maps shrink with each layer!

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

Solution: Add **padding** around the input before sliding the filter

Convolution: Spatial Dimensions



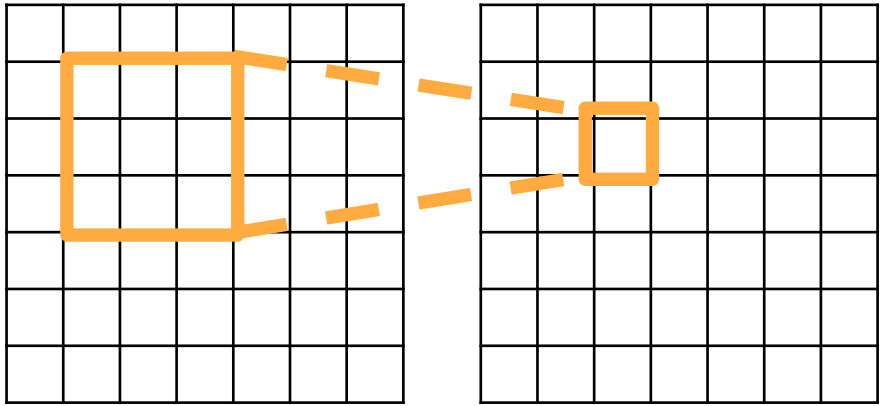
Input: 7x7
Filter: 3x3
Output: 5x5

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

Common setting:
P = (K – 1) / 2
Means output has same size as input

Receptive Fields

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

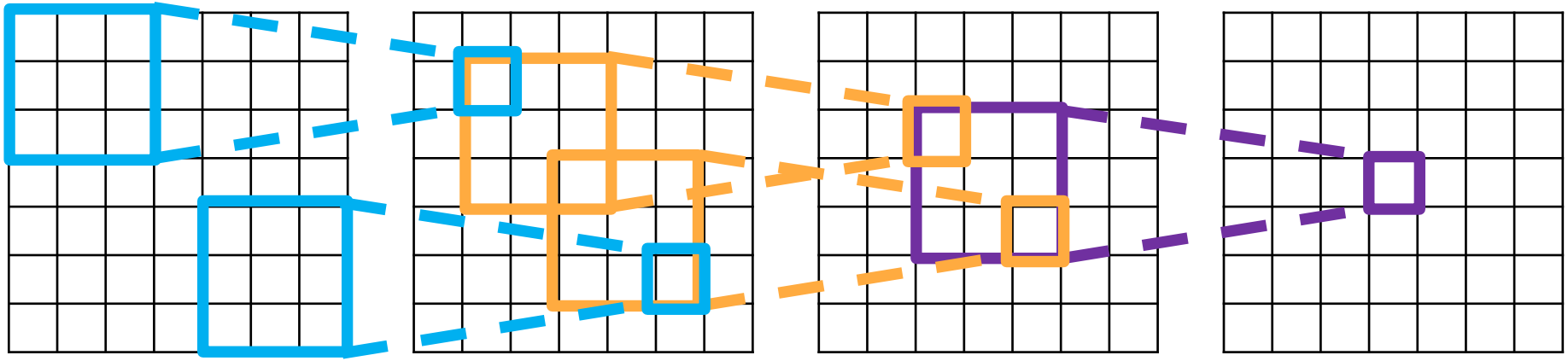


Input Output

Slide inspiration: Justin 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)



Input

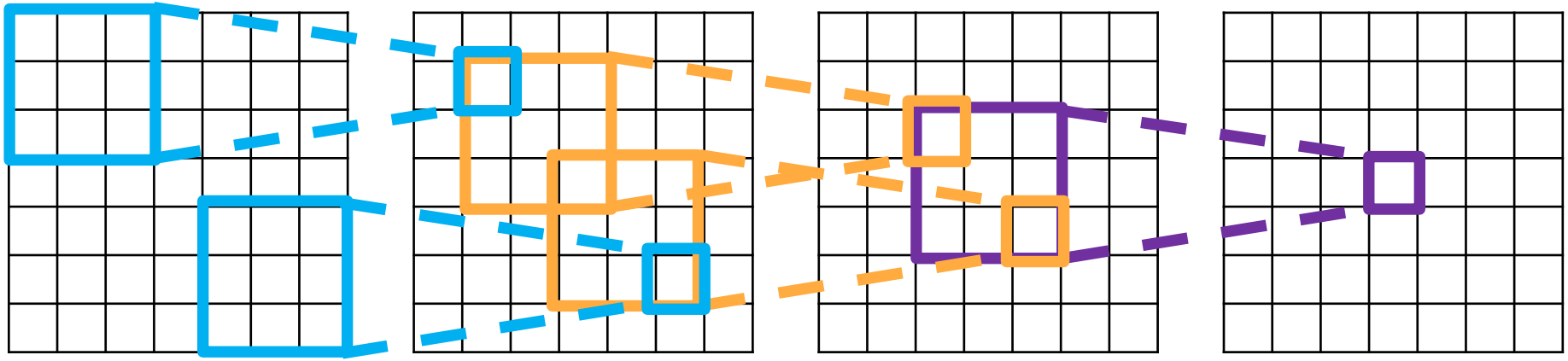
Output

Be careful – “receptive field in the input” vs. “receptive field in the previous layer”

Slide inspiration: Justin 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)



Input

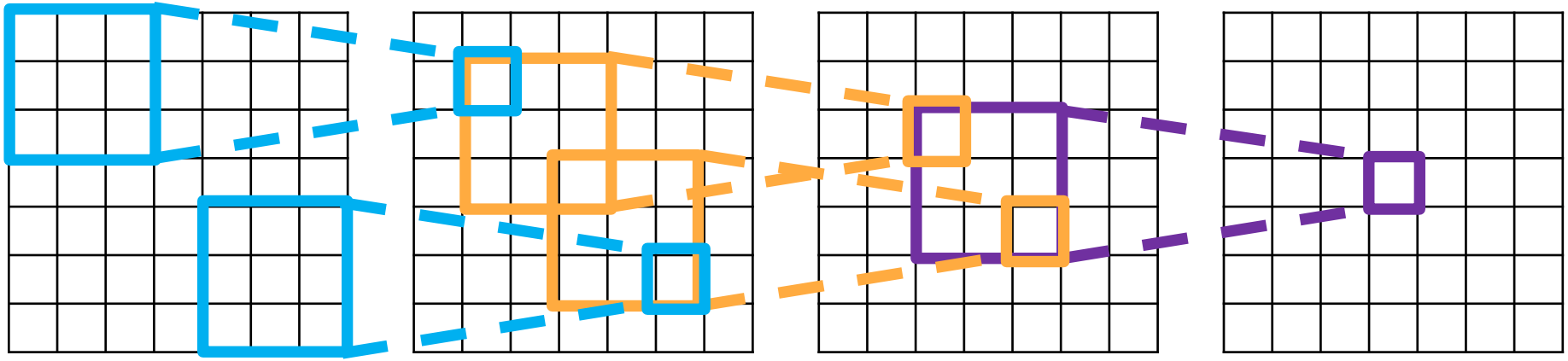
Problem: For large images we need many layers for each output to “see” the whole image image

Output

Slide inspiration: Justin 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)



Input

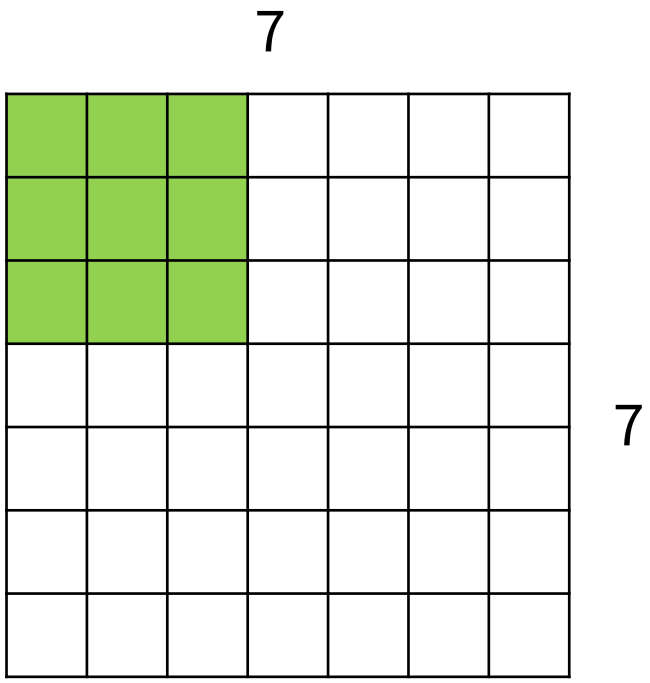
Output

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

Solution: Downsample inside the network

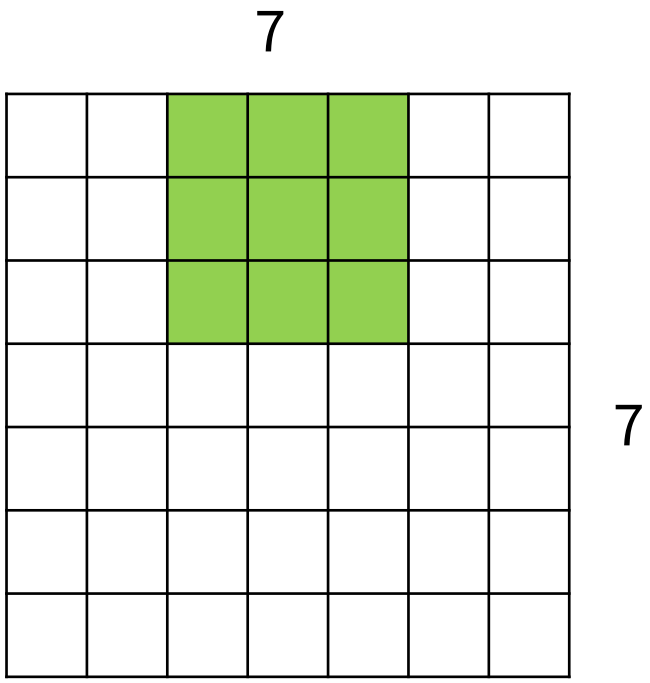
Slide inspiration: Justin Johnson

Strided Convolution



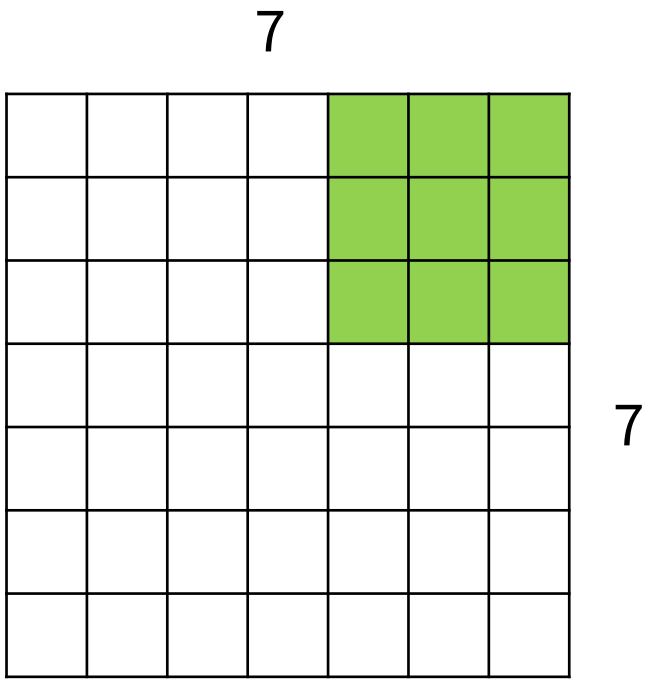
Input: 7x7
Filter: 3x3
Stride: 2

Strided Convolution



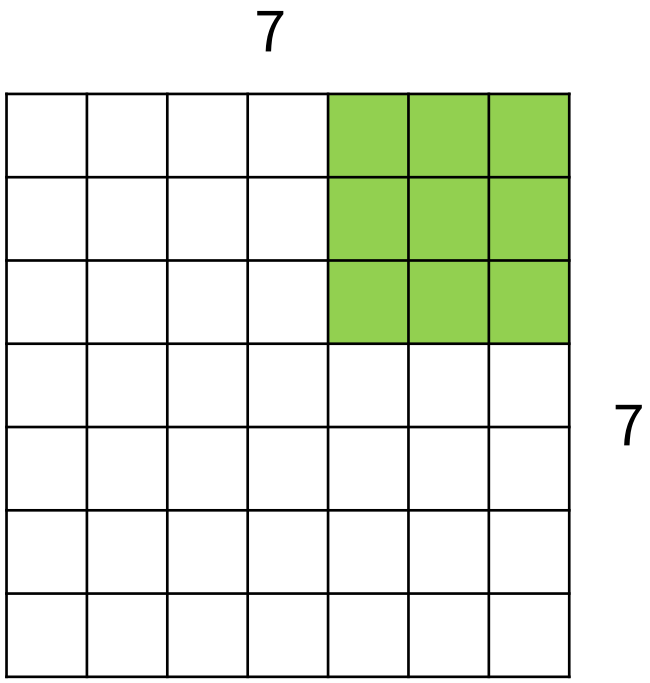
Input: 7x7
Filter: 3x3
Stride: 2

Strided Convolution



Input: 7x7
Filter: 3x3
Stride: 2
Output: 3x3

Strided Convolution



Input: 7x7
Filter: 3x3
Stride: 2
Output: 3x3

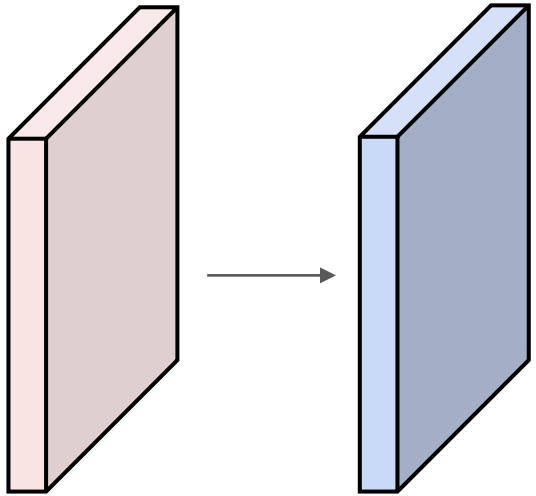
In general:
Input: W
Filter: K
Padding: P
Stride: S

→ Output:
(W – K + 2P) / S + 1

Convolution Example

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

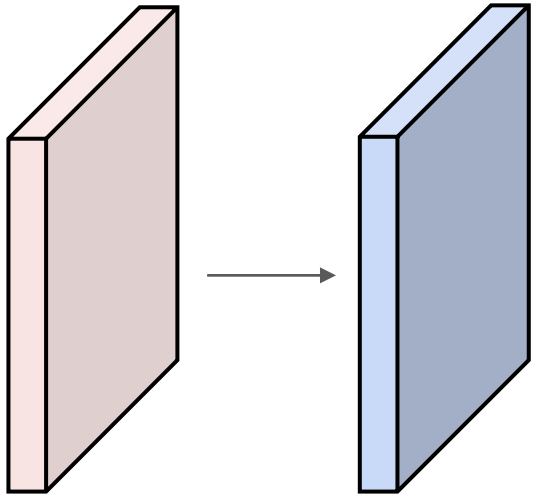
Output volume size: ?



Convolution Example

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

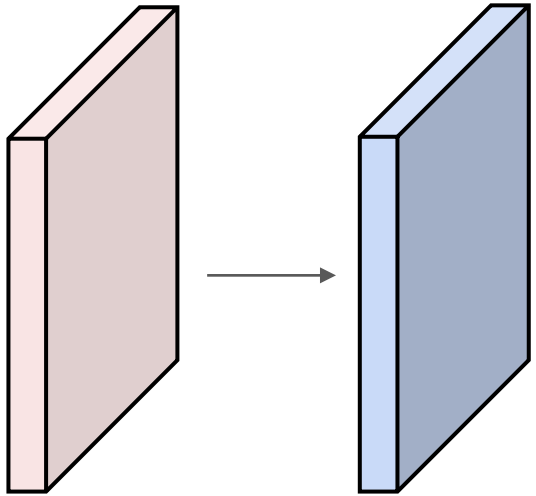
Output volume size: **10** x **32** x **32**
32 = (**32**+2*__2__-**5**)/**1**+1



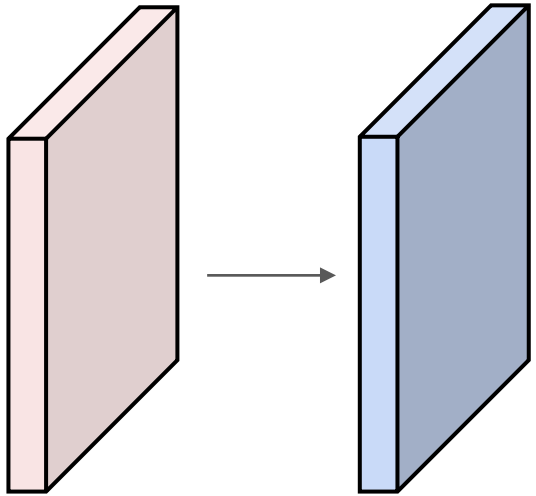
Convolution Example

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: ?



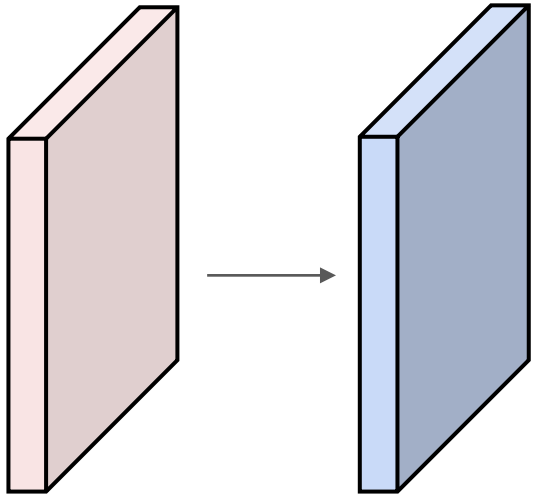
Convolution Example



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**

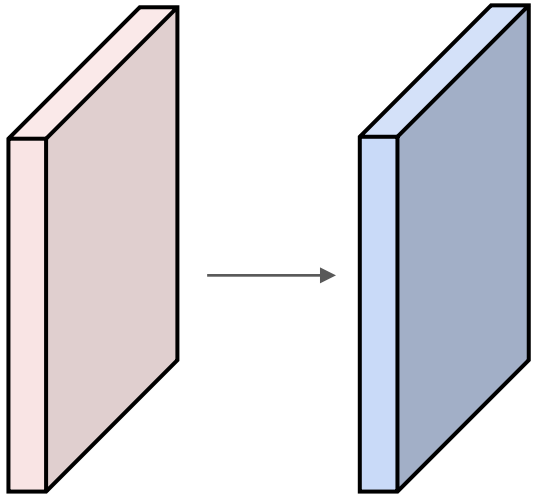
Convolution Example



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?

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**

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)

PyTorch Convolution Layer

CLASS torch.nn.Conv2d(*in_channels*, *out_channels*, *kernel_size*, *stride=1*, *padding=0*, *dilation=1*, *groups=1*, *bias=True*, *padding_mode='zeros'*, *device=None*, *dtype=None*) [SOURCE]

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

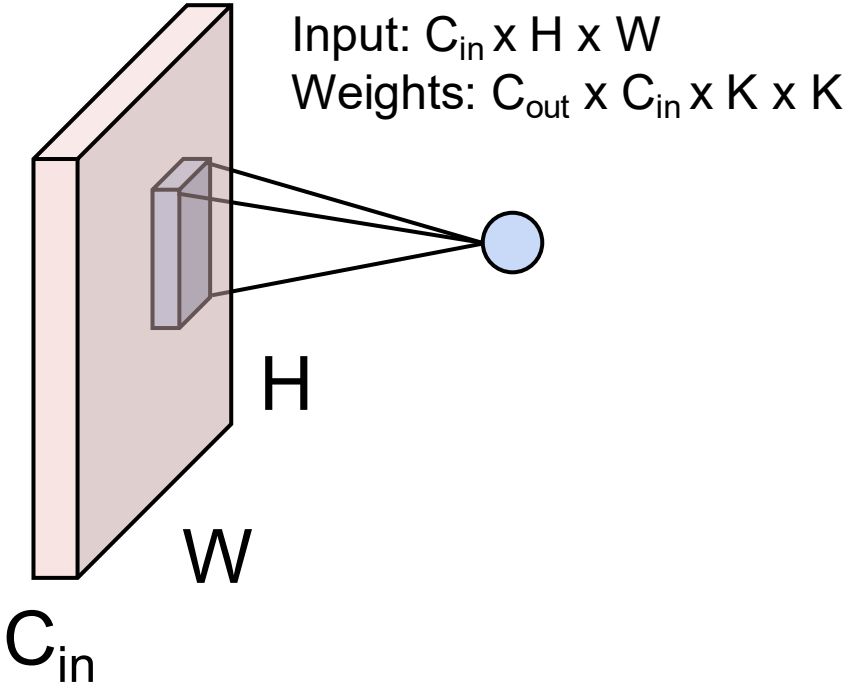
In the simplest case, the output value of the layer with input size (N, C_{in}, H, W) and output $(N, C_{\text{out}}, H_{\text{out}}, W_{\text{out}})$ can be precisely described as:

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

We didn't talk about groups or dilation…

Other Types of Convolution

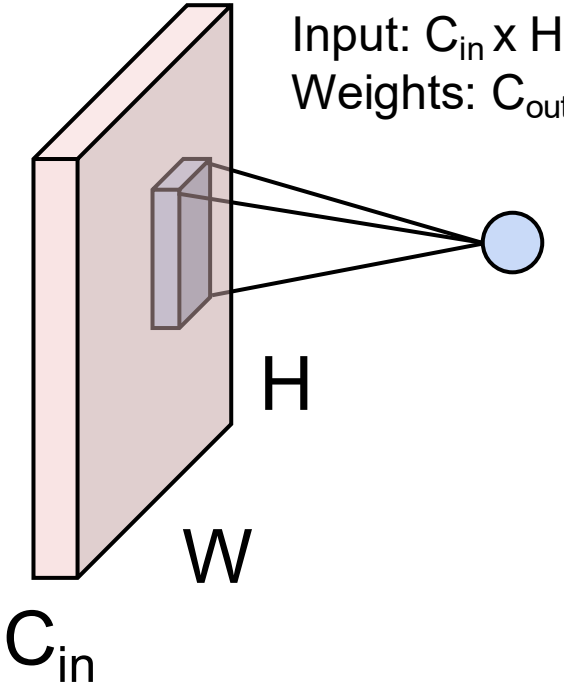
So far: 2D Convolution



Other Types of Convolution

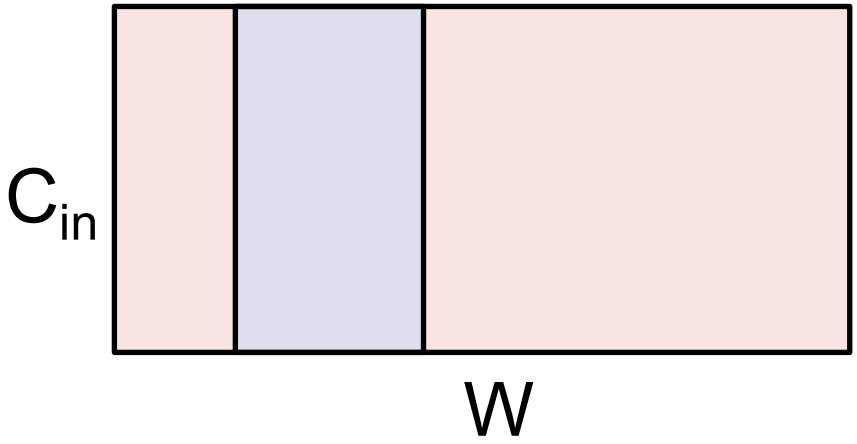
So far: 2D Convolution

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



1D Convolution

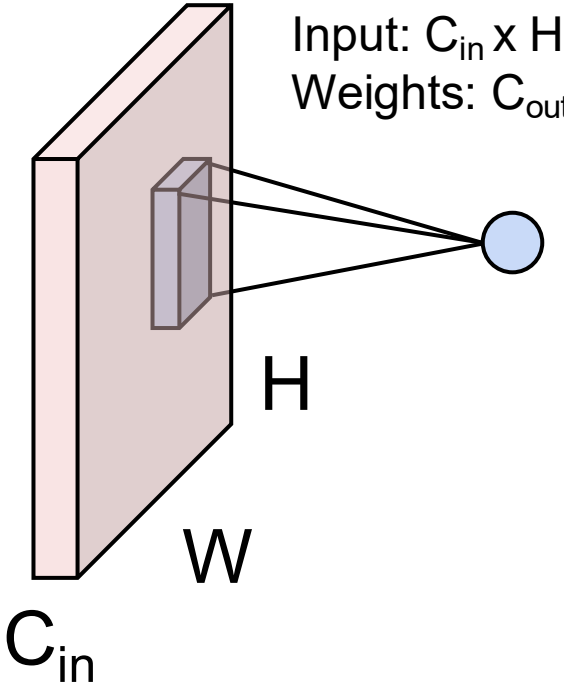
Input: C_{in} x W
Weights: C_{out} x C_{in} x K



Other Types of Convolution

So far: 2D Convolution

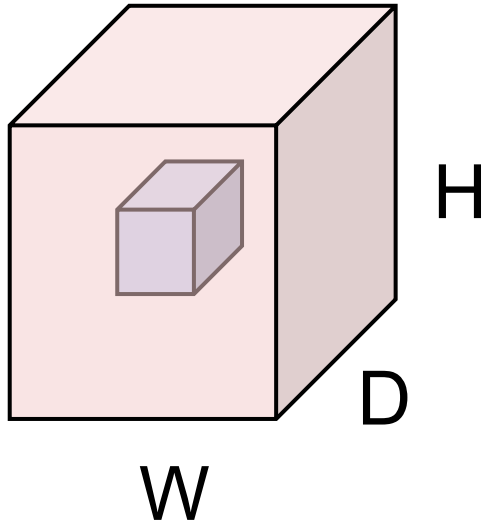
Input: C_{in} x H x W
Weights: C_{out} x C_{in} x K x K



3D Convolution

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

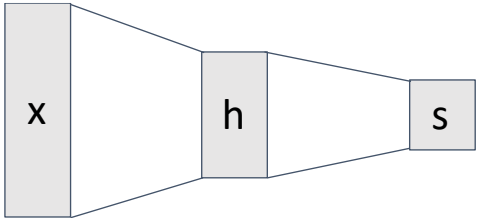
C_{in}-dim vector
at each point
in the volume



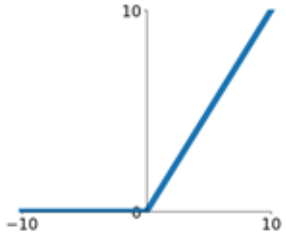
Convolutional Networks

We have already seen these

Fully-Connected Layer

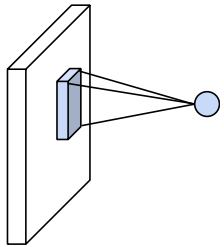


Activation Function

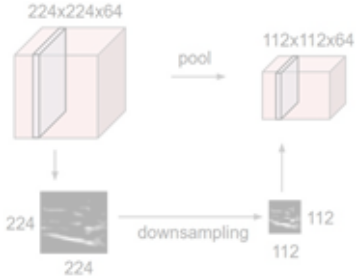


Today: Image-specific operators

Convolution Layer



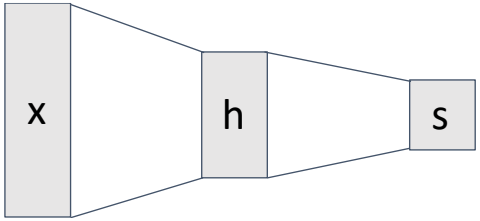
Pooling Layer



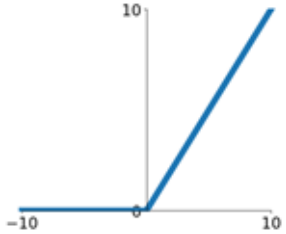
Convolutional Networks

We have already seen these

Fully-Connected Layer

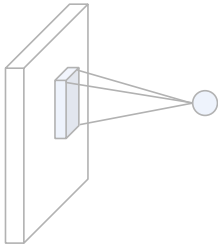


Activation Function

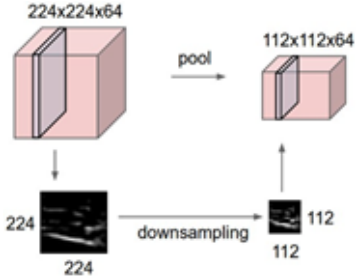


Today: Image-specific operators

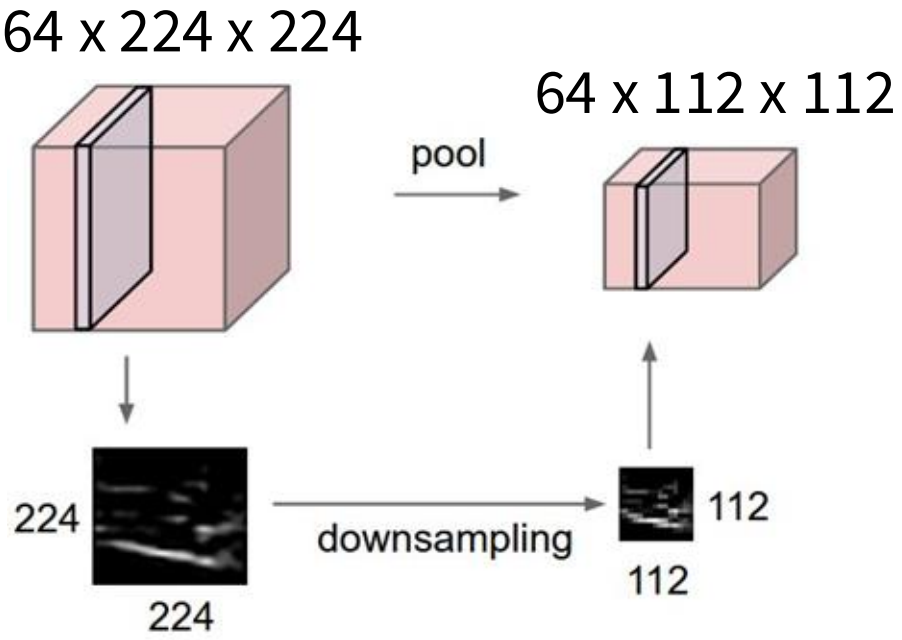
Convolution Layer



Pooling Layer



Pooling Layers: Another way to downsample



Given an input C x H x W,
downsample each 1 x H x W plane

Hyperparameters:
Kernel Size
Stride
Pooling function

Pooling Layers: Another way to downsample

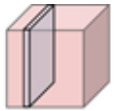
Single depth slice

1	1	2	4
5	**6**	7	**8**
3	2	1	0
1	2	3	**4**

x

y

64 x 224 x 224



Max pooling with 2x2 kernel size and stride 2

6	8
3	4

Gives **invariance** to small spatial shifts. No learnable parameters.

Pooling Summary

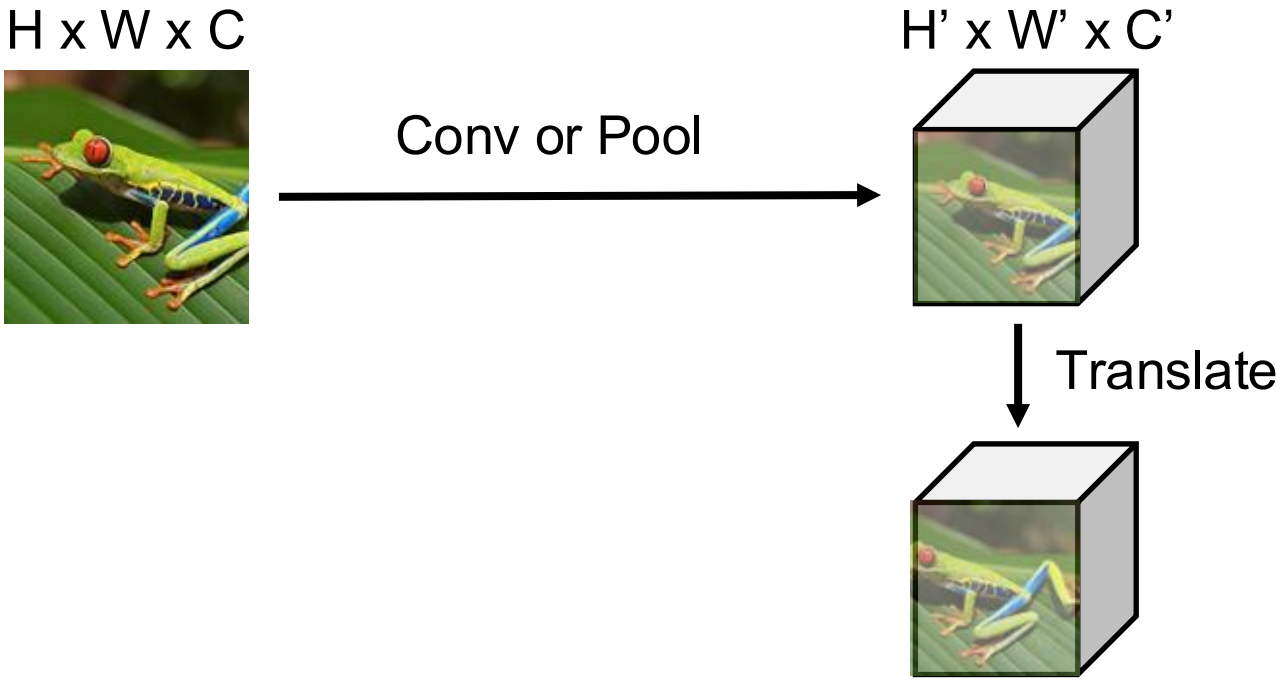
Input: C x H x W
Hyperparameters:
- **Kernel size**: K
- **Stride**: S
- **Pooling function**: max, avg

Output size: C x H' x W' where:
- H' = (H – K) / S + 1
- W' = (W – K) / S + 1

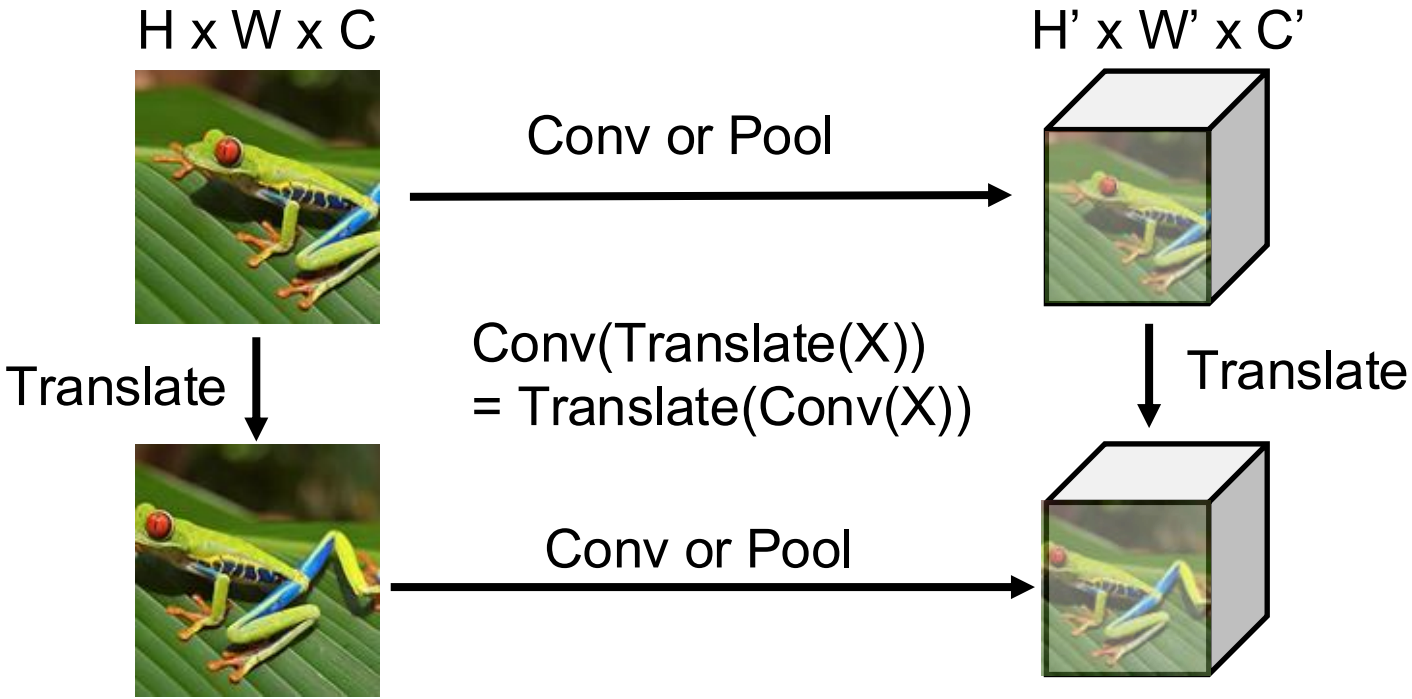
No learnable parameters

Common setting:
max, K=2, S=2 => Gives 2x downsampling

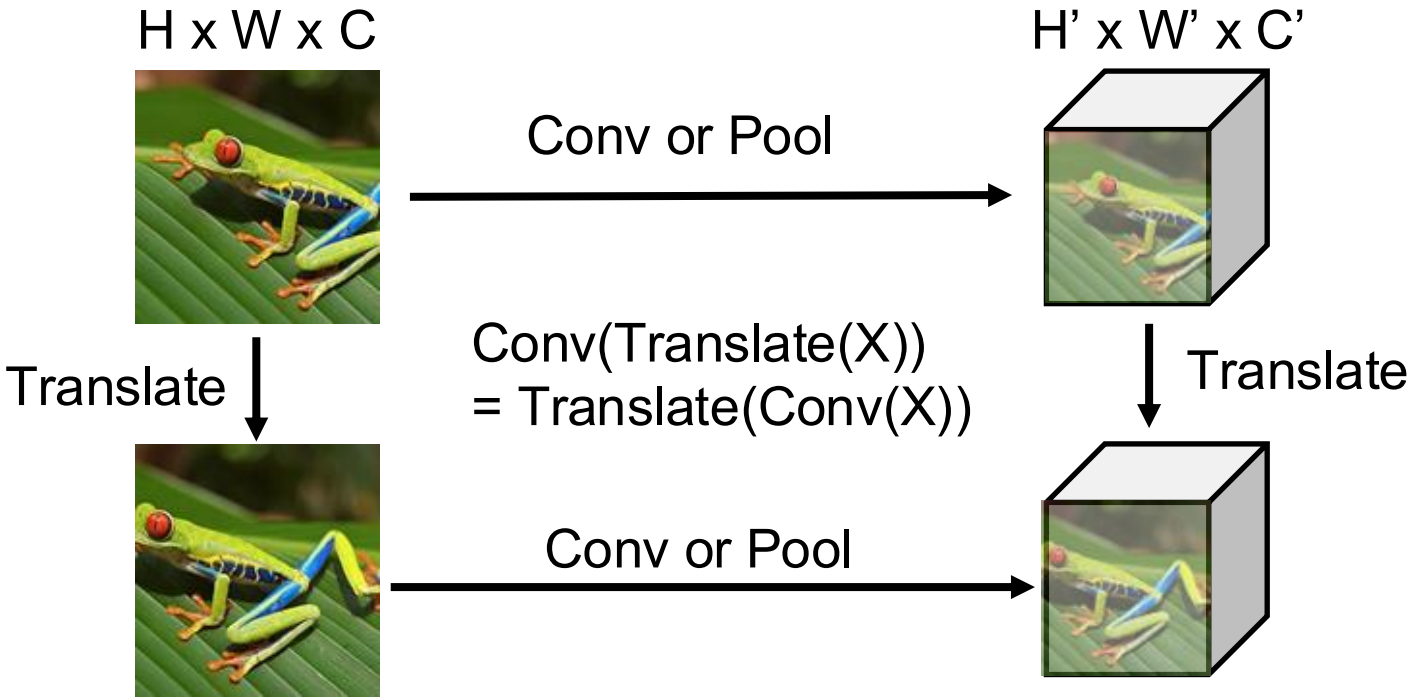
Convolution and Pooling: Translation Equivariance



Convolution and Pooling: Translation Equivariance



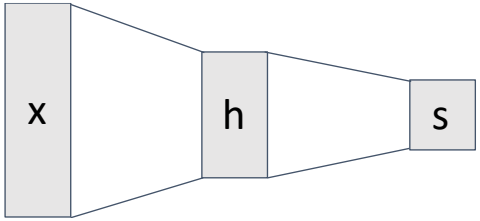
Convolution and Pooling: Translation Equivariance



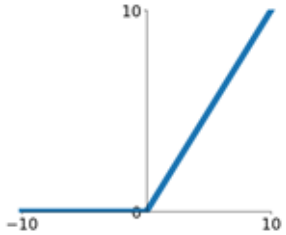
Intuition: Features of images don't depend on their location in the image

Summary: Convolutional Networks

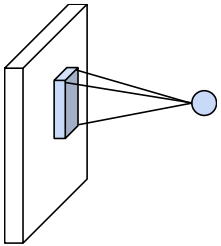
Fully-Connected Layer



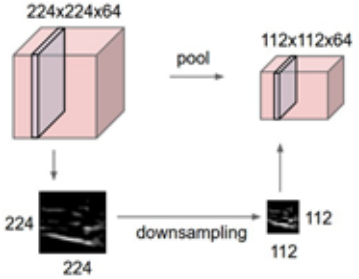
Activation Function



Convolution Layer



Pooling Layer



Next time: CNN Architectures