CS 446/ECE 449: Machine Learning

Shenlong Wang

University of Illinois at Urbana-Champaign, 2024

Convolutional Neural Networks (CNNs)

Goals of this lecture

- Convolutional Neural Networks

Goals of this lecture

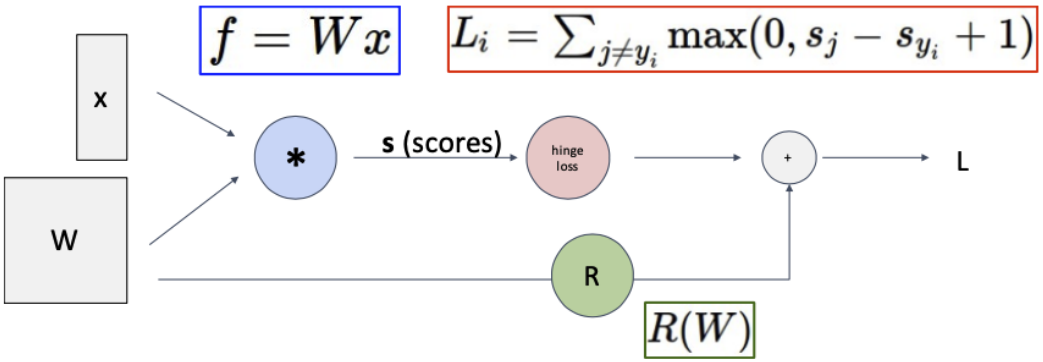
- Convolutional Neural Networks

Reading material

- I. Goodfellow et al.; Deep Learning; Chapters 6-9

Recall: Backpropagation

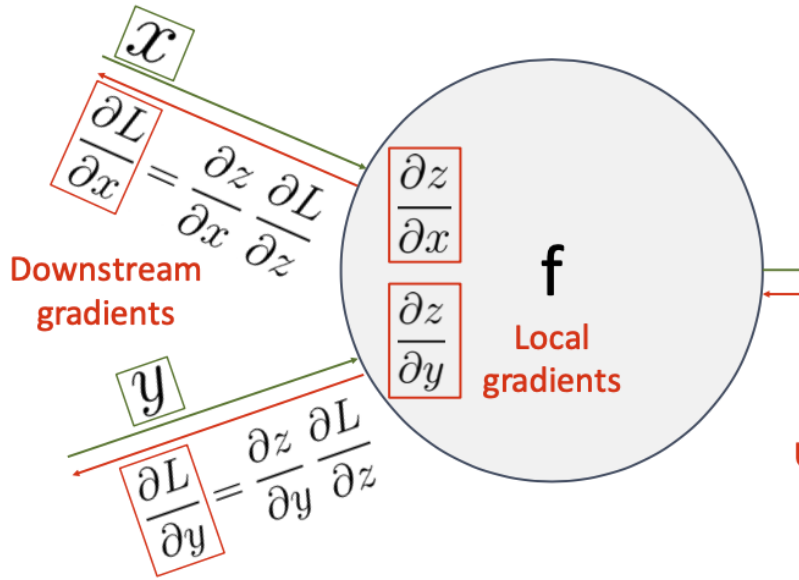
Represent complex expressions as **computational graphs**



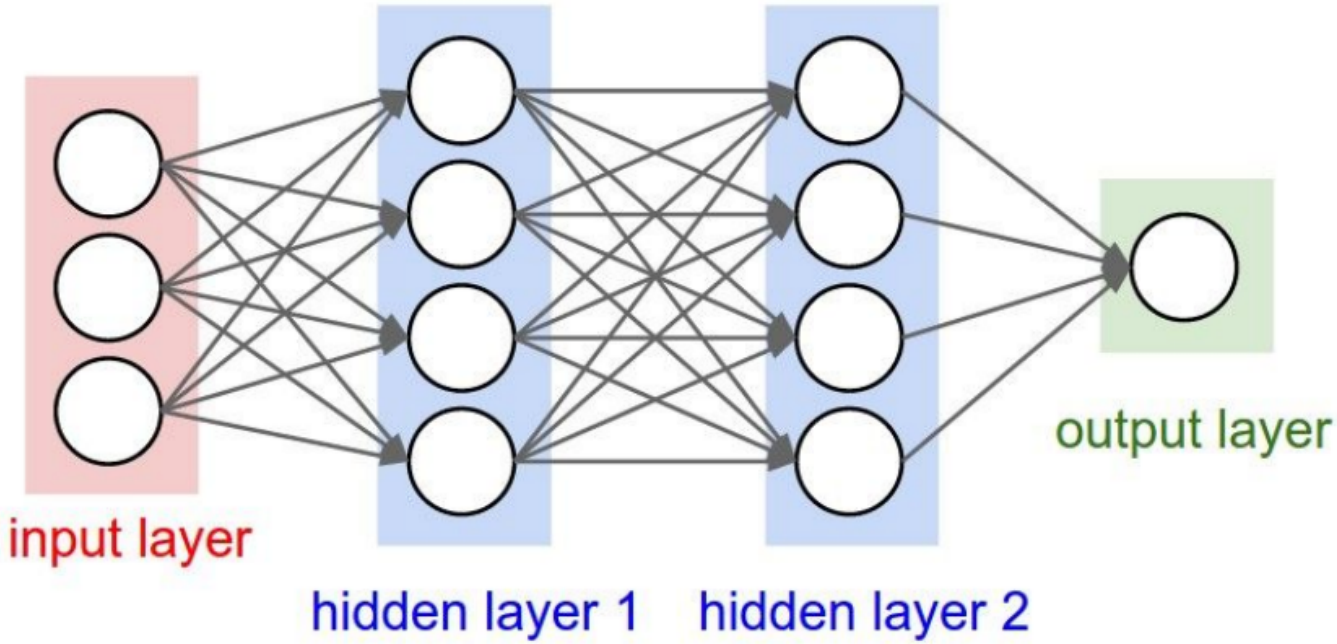
Forward pass computes outputs

Backward pass computes gradients

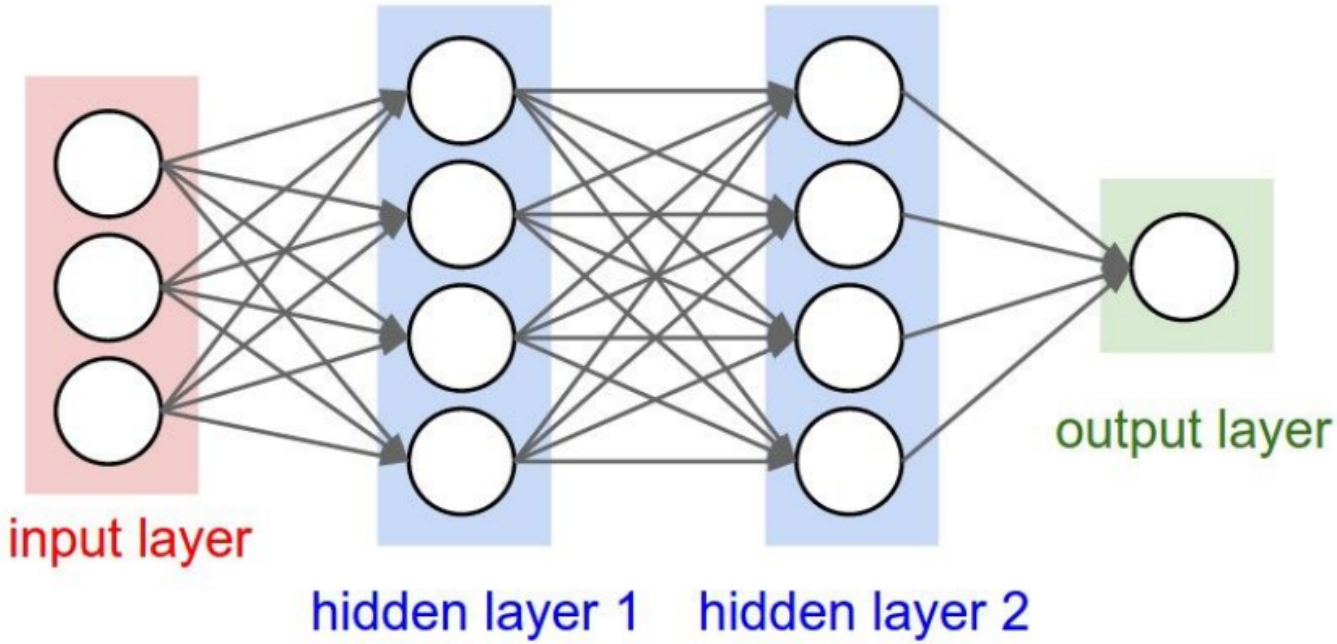
During the backward pass, each
the graph receives **upstream gra**
and multiplies them by **local gra**
compute **downstream gradient**



Recall: Multi-layer Perceptron



Recall: Multi-layer Perceptron

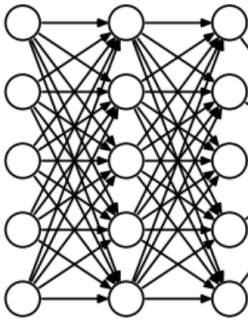


Problems:

- If x is a 500×500 image, z_1 is 1024-dimensional, what is the size of the W_1 ?
- If x is a 500×500 image, x is shifted by one pixel, do you expect the z_1 remain similar?

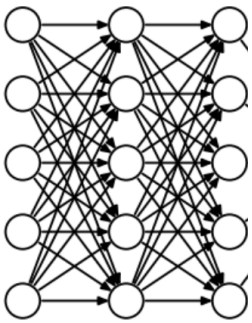
Components of a Fully-Connected Network

Fully-connected layers

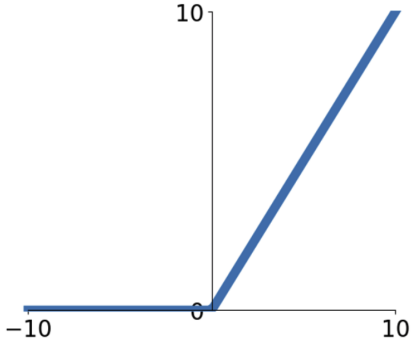


Components of a Fully-Connected Network

Fully-connected layers

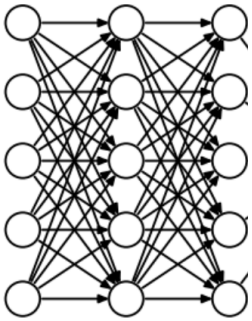


Activation Function

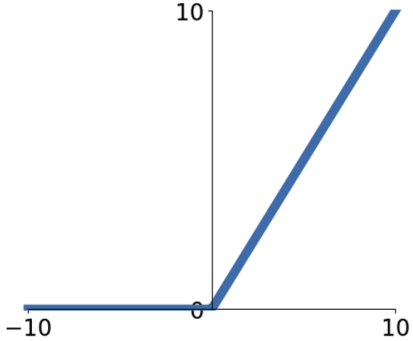


Components of a Convolutional Network

Fully-connected layers

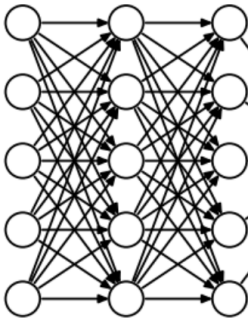


Activation Function

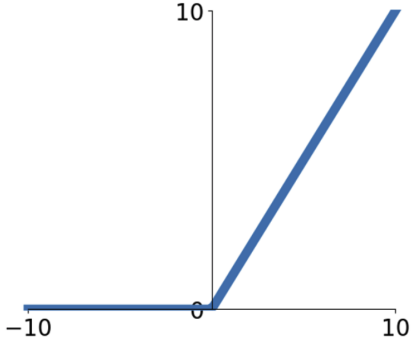


Components of a Convolutional Network

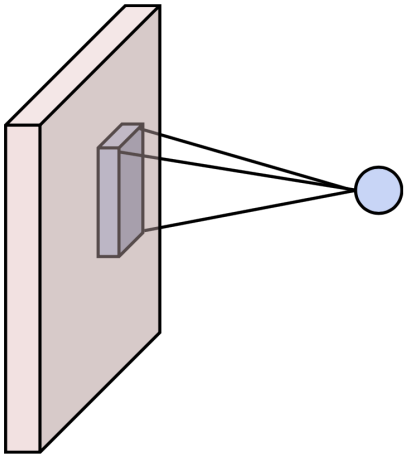
Fully-connected layers



Activation Function

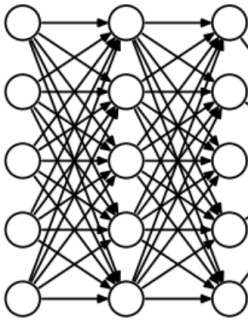


Convolution layers

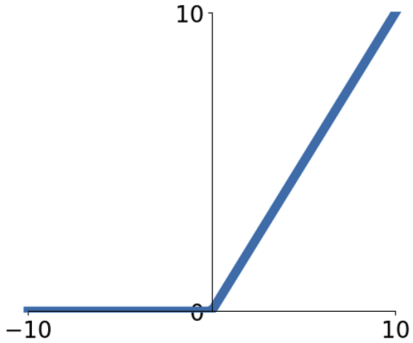


Components of a Convolutional Network

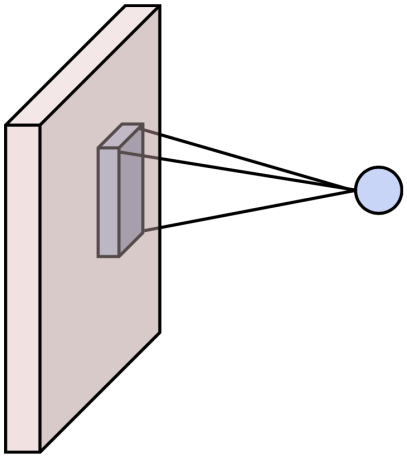
Fully-connected layers



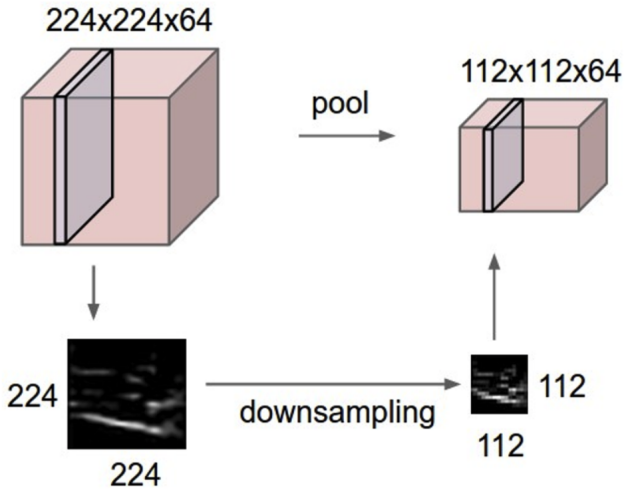
Activation Function



Convolution layers

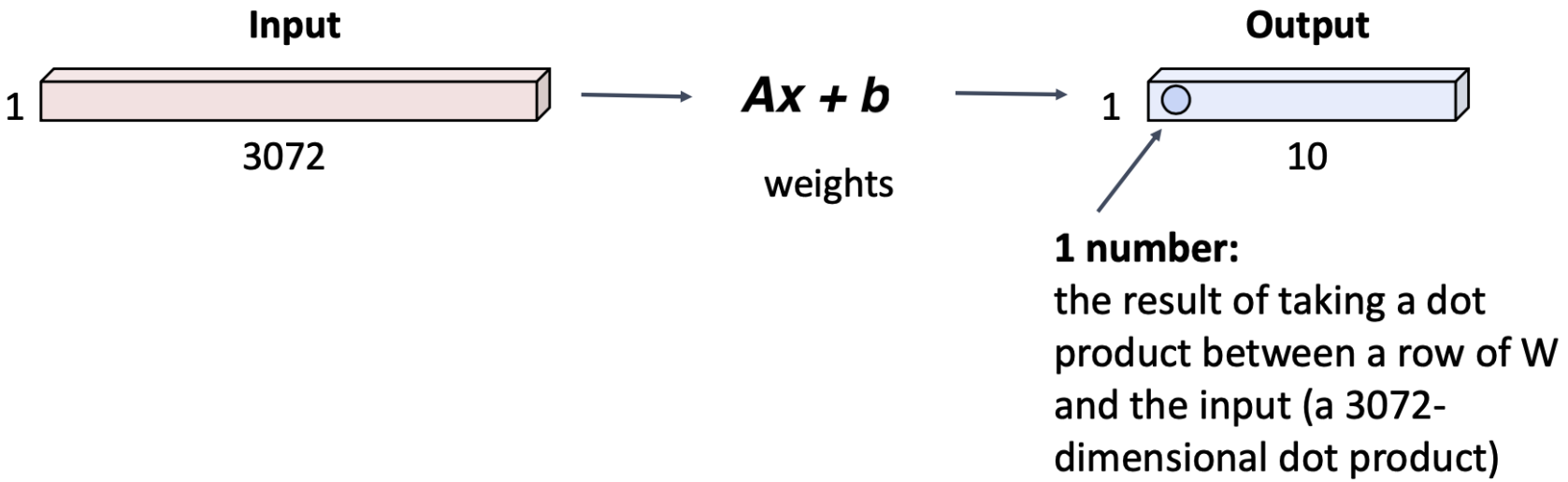


Pooling Layers



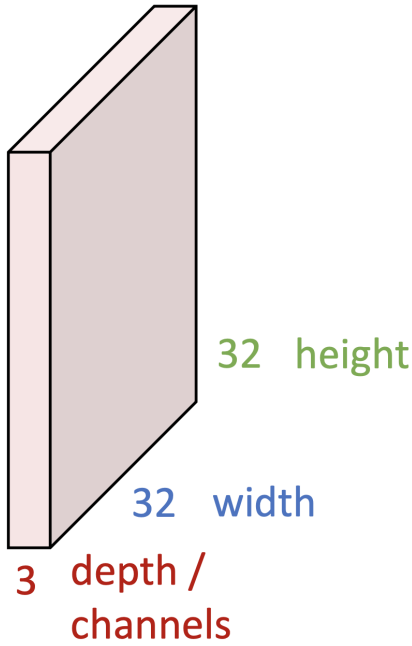
Fully-Connected Layer

32x32x3 image -> stretch to 3072 x 1



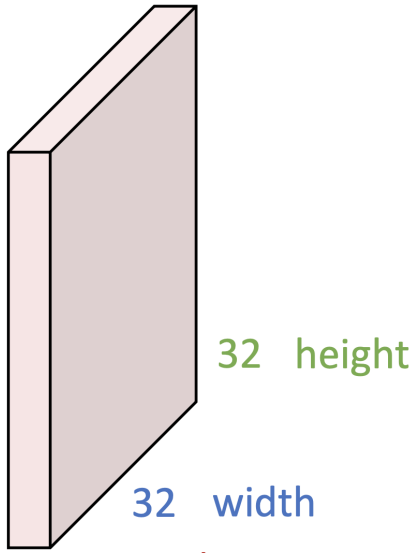
Convolution Layer

3x32x32 image: preserve spatial structure

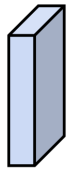


Convolution Layer

3x32x32 image

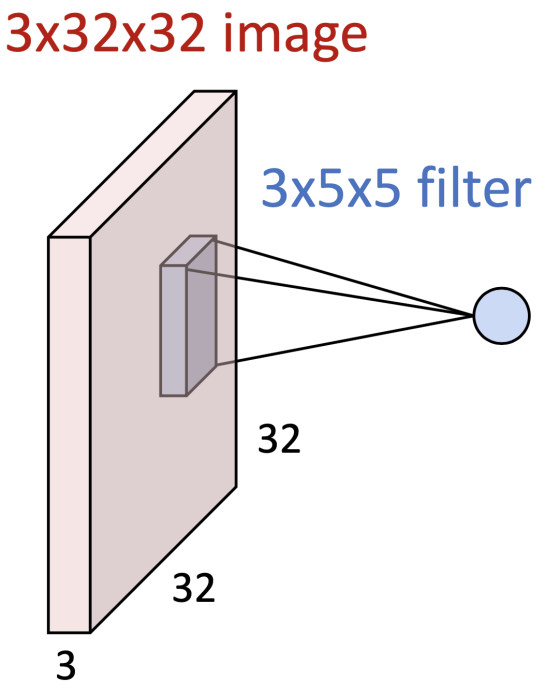


3x5x5 filter



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

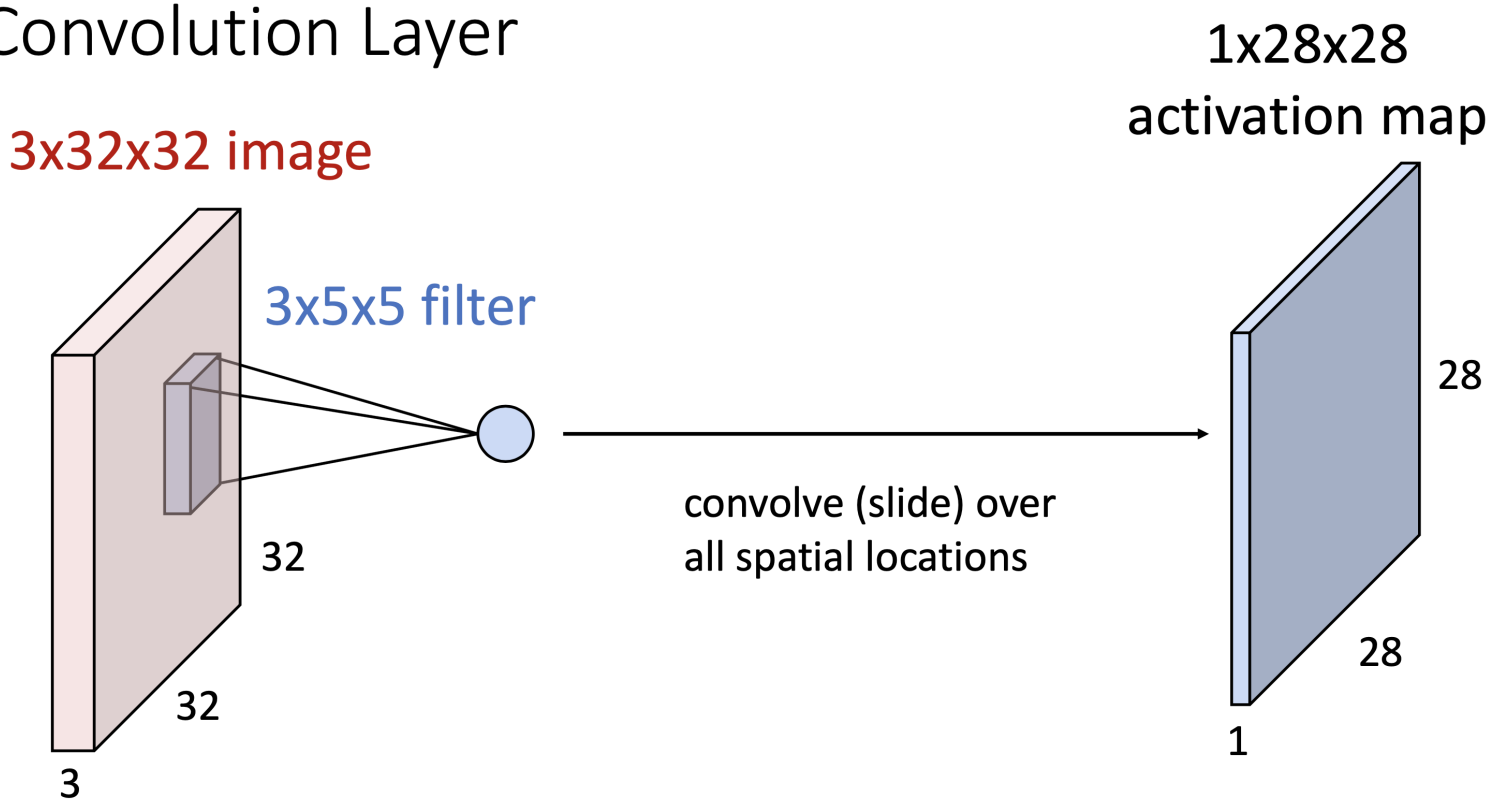
Convolution Layer



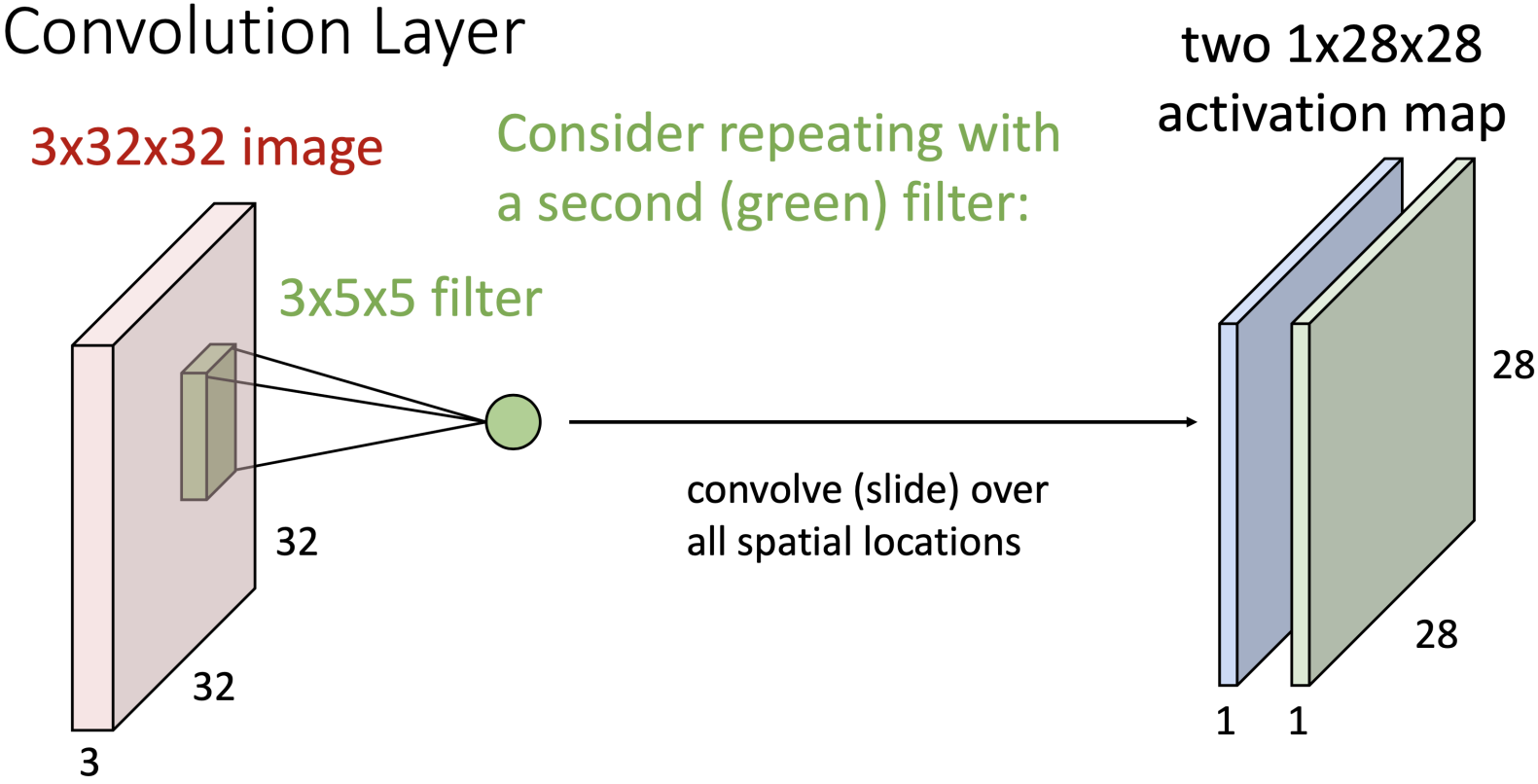
1 number:
the result of taking a dot product between the filter and a small 3x5x5 chunk of the image
(i.e. 3*5*5 = 75-dimensional dot product + bias)

$$w^T x + b$$

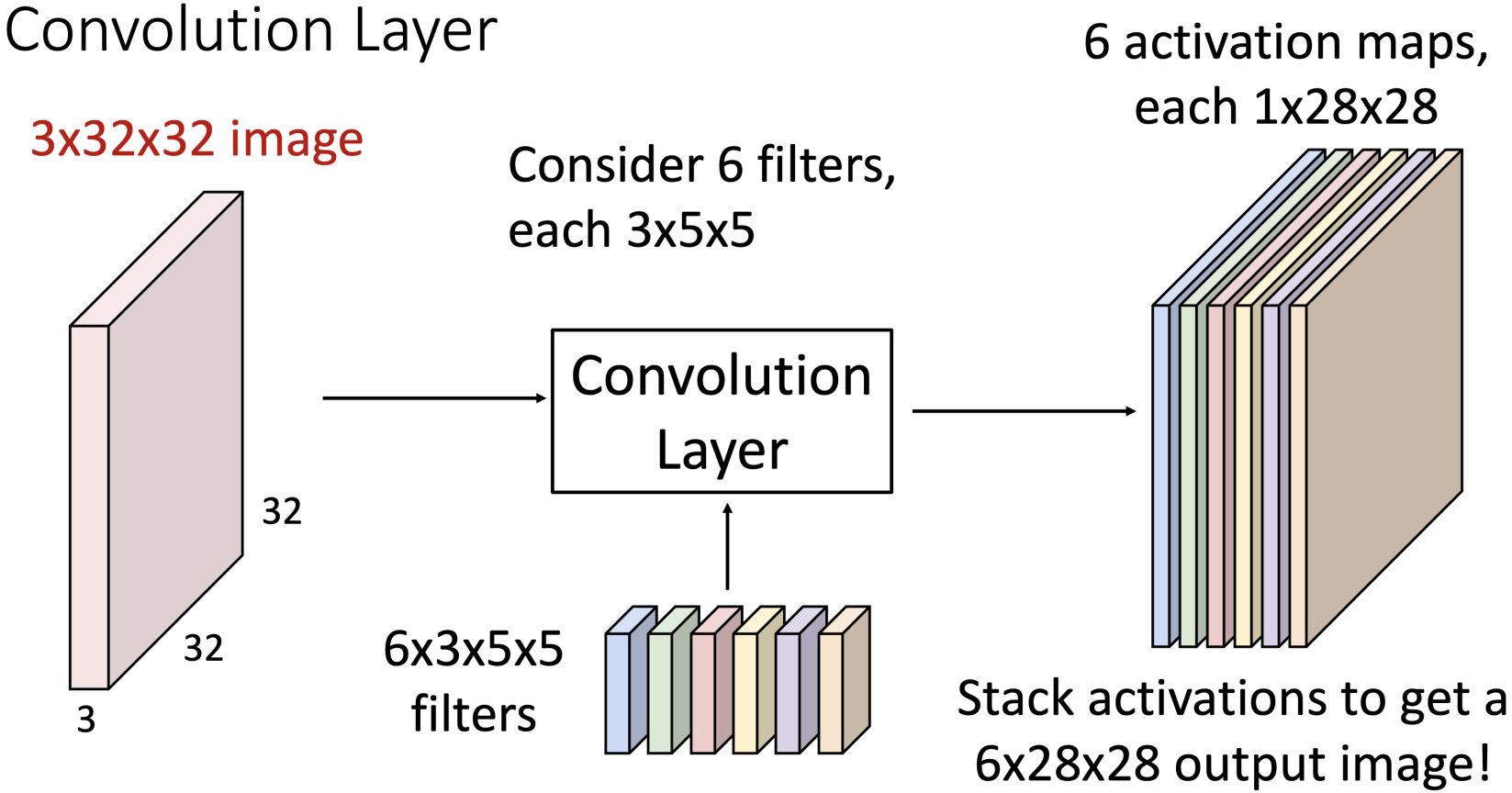
Convolution Layer



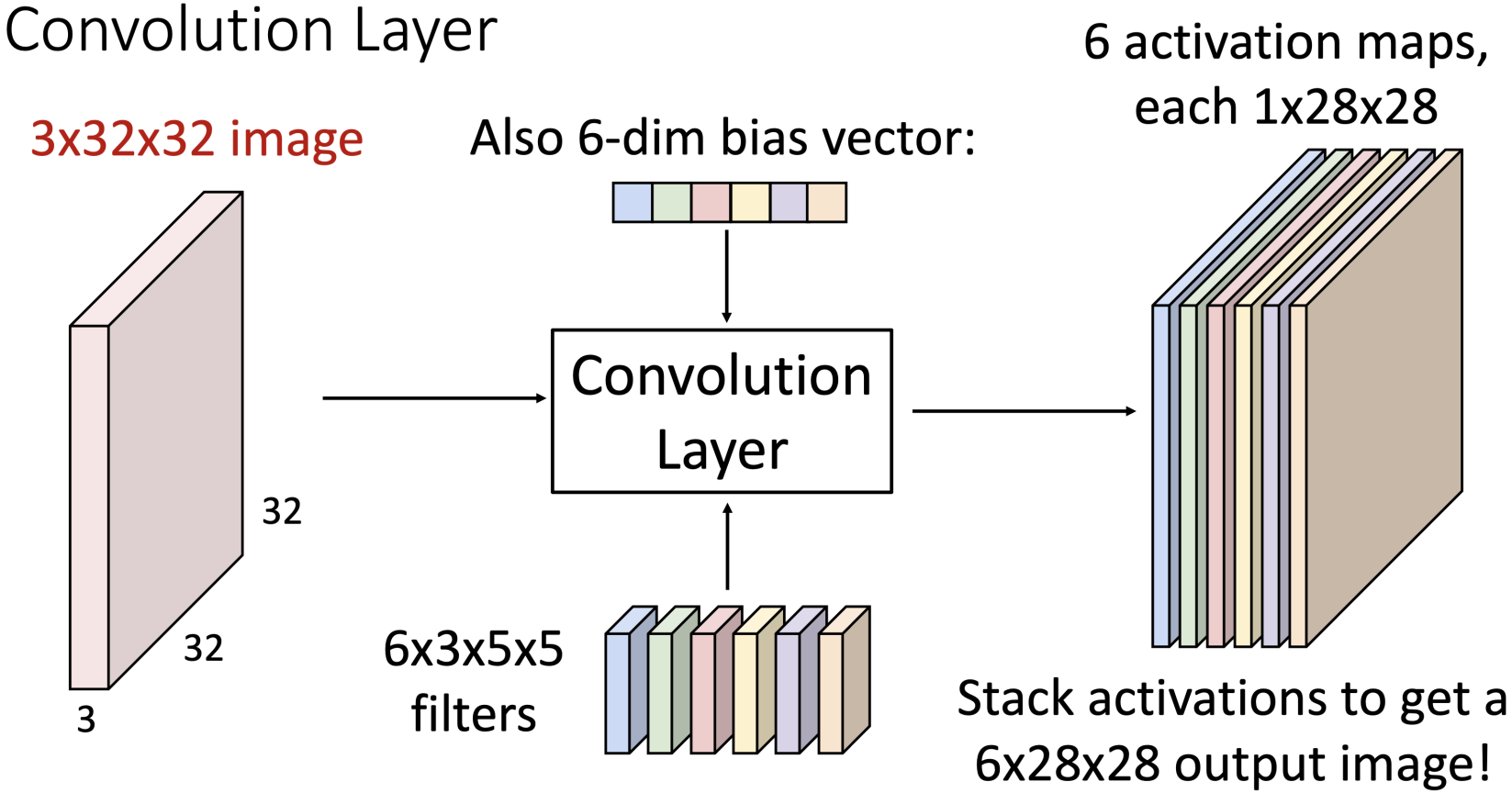
Convolution Layer

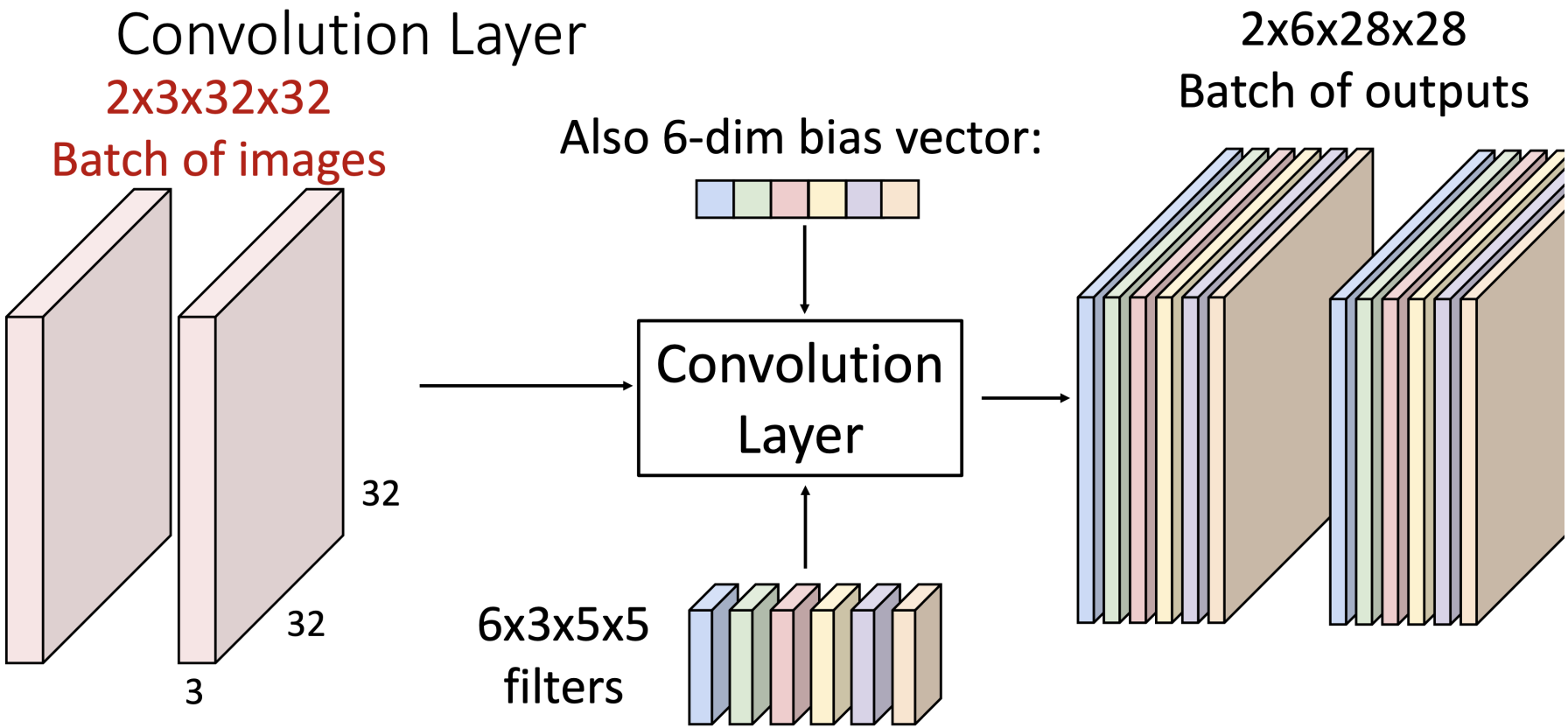


Convolution Layer

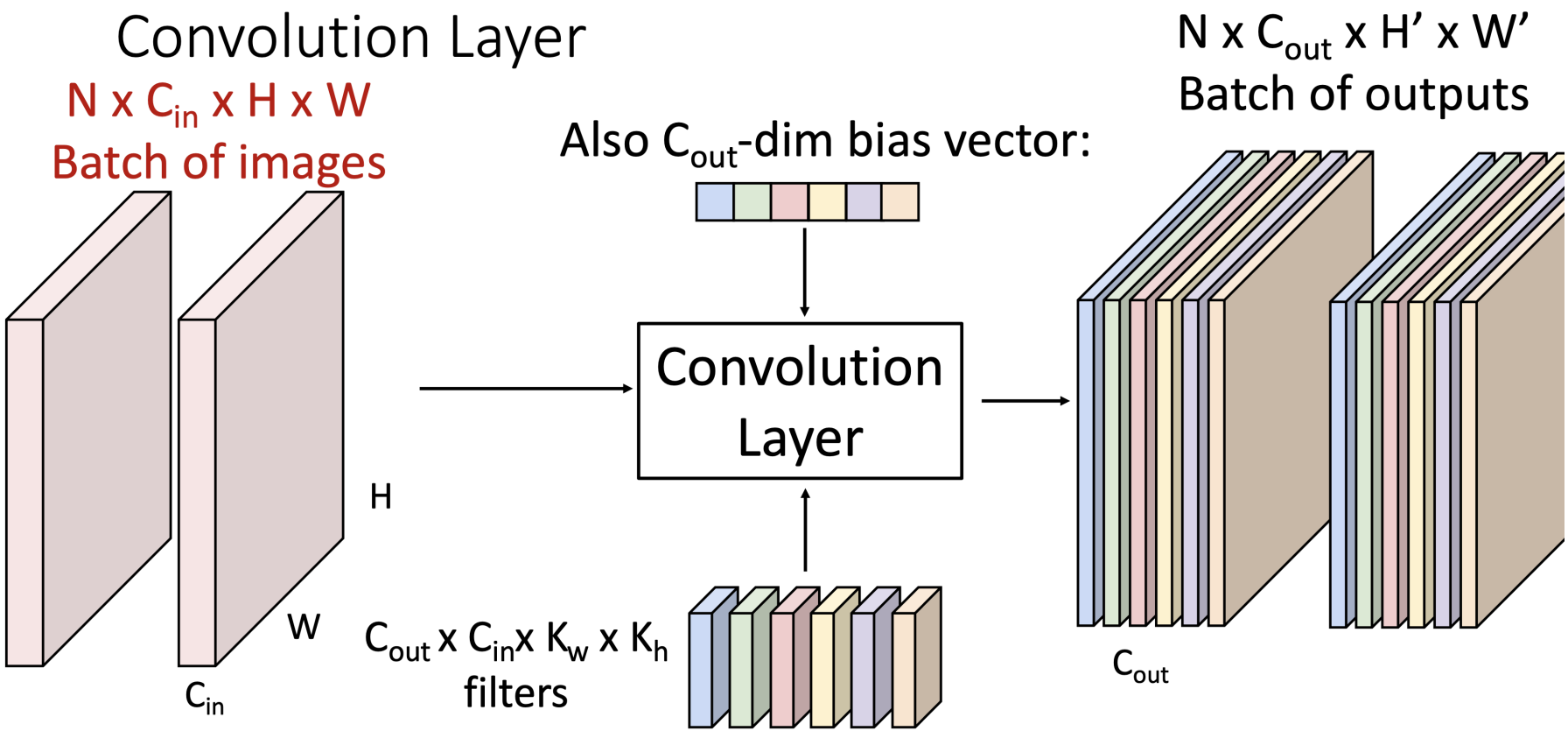


Convolution Layer



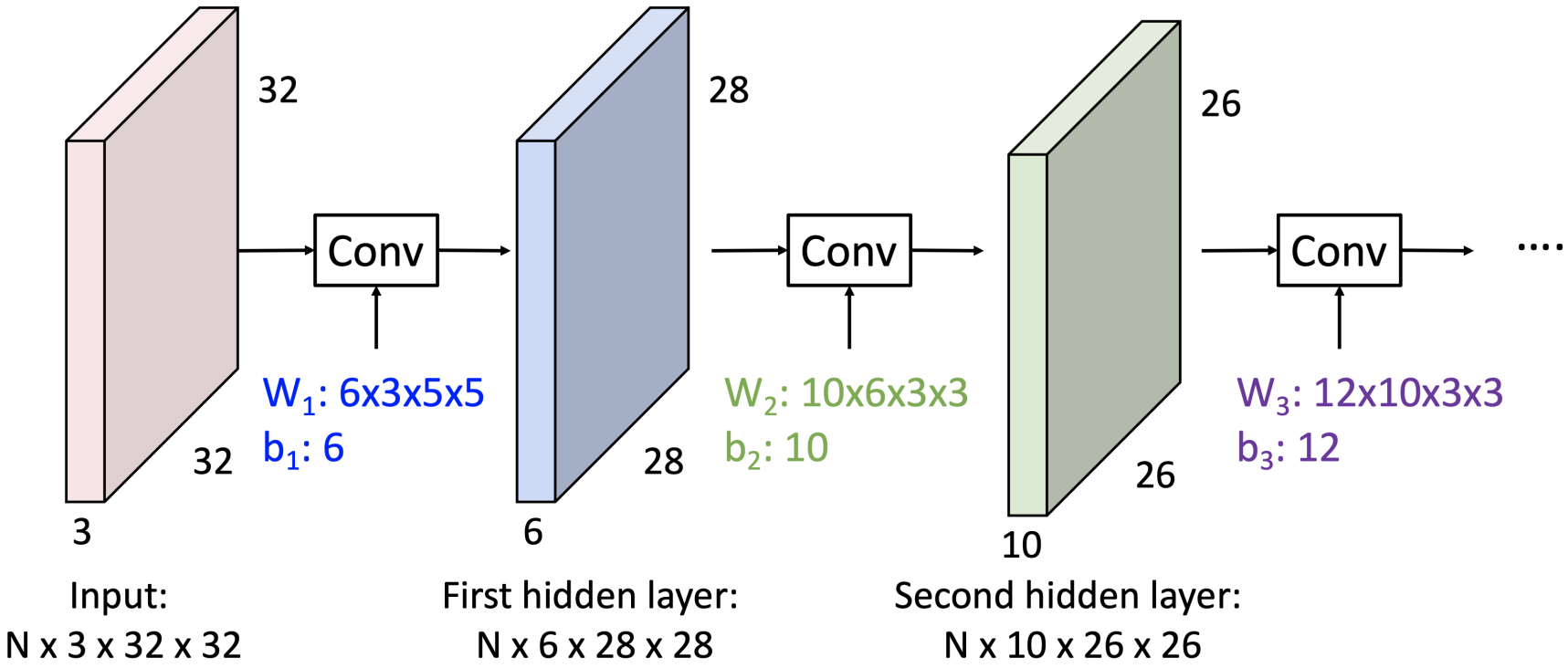

Convolution Layer
2x3x32x32
Batch of images
32
32
3
Also 6-dim bias vector:
Convolution Layer
6x3x5x5
filters
2x6x28x28
Batch of outputs

Convolution Layer



Stacking Convolutions

Q: What happens if we stack two convolution layers?



Input:
N x 3 x 32 x 32

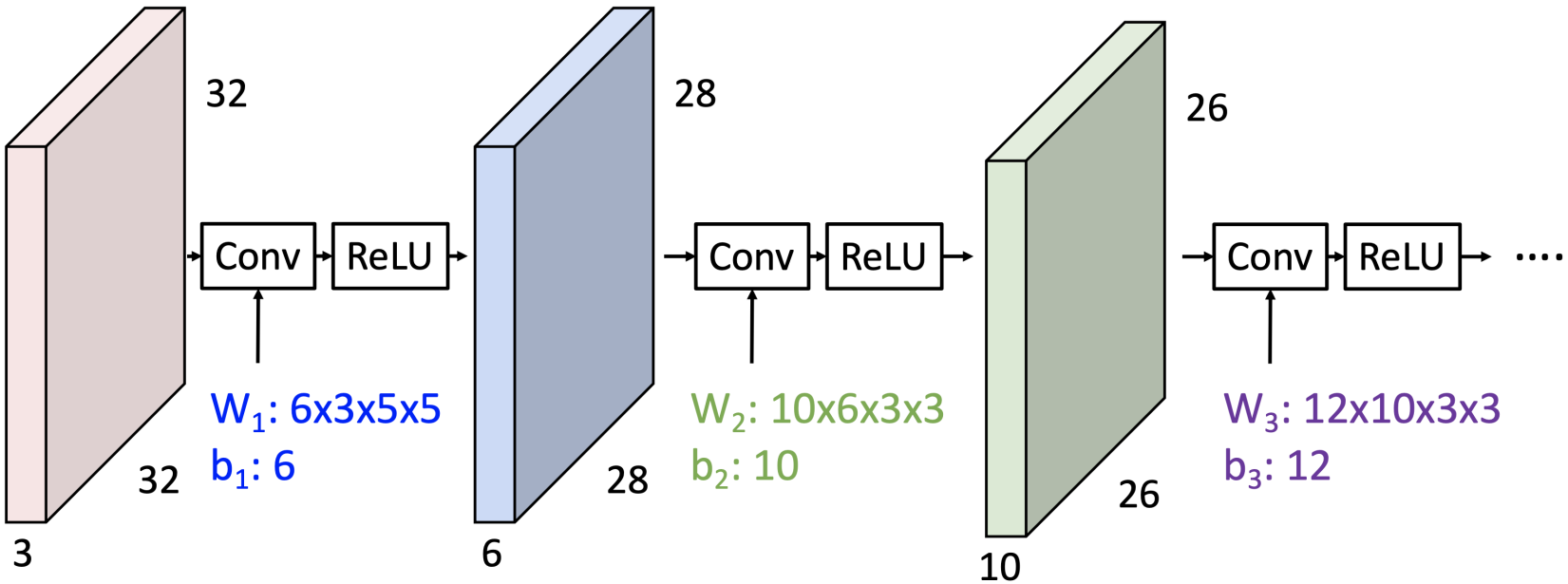
First hidden layer:
N x 6 x 28 x 28

Second hidden layer:
N x 10 x 26 x 26

Stacking Convolutions

Q: What happens if we stack two convolution layers?
A: We get another convolution!

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

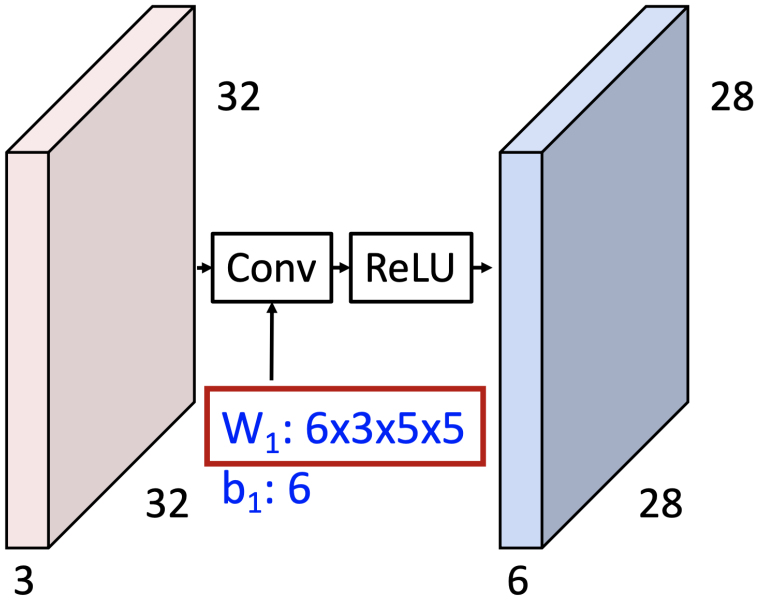


Input:
N x 3 x 32 x 32

First hidden layer:
N x 6 x 28 x 28

Second hidden layer:
N x 10 x 26 x 26

What do convolutional filters learn?

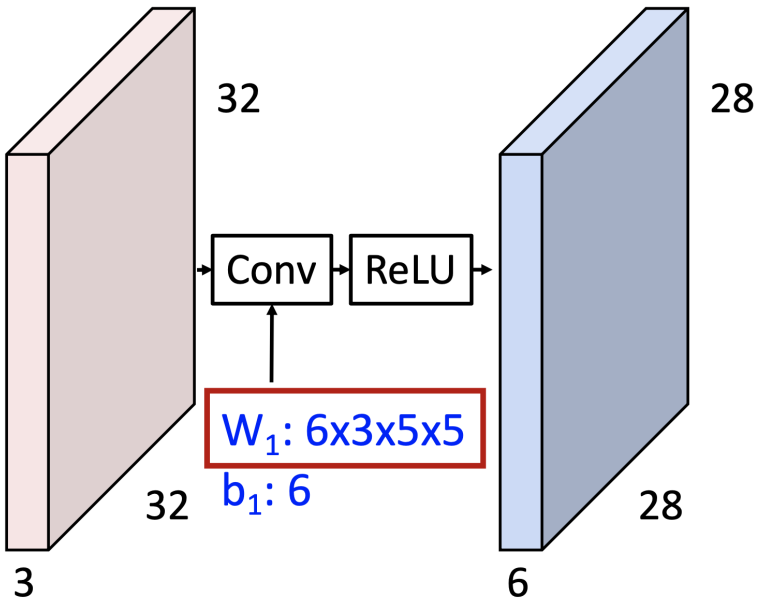


Input:
N x 3 x 32 x 32

First hidden layer:
N x 6 x 28 x 28

Linear classifier: One template per class

What do convolutional filters learn?

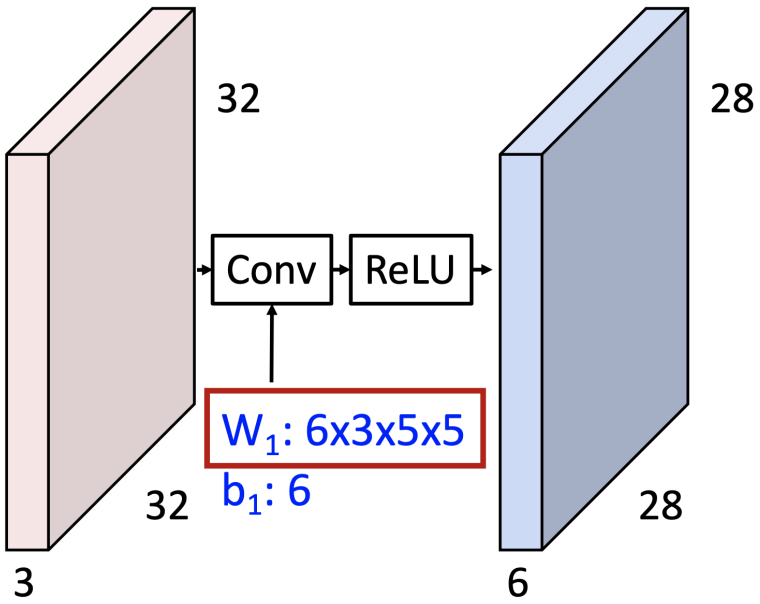


Input:
N x 3 x 32 x 32

First hidden layer:
N x 6 x 28 x 28

MLP: Bank of whole-image templates

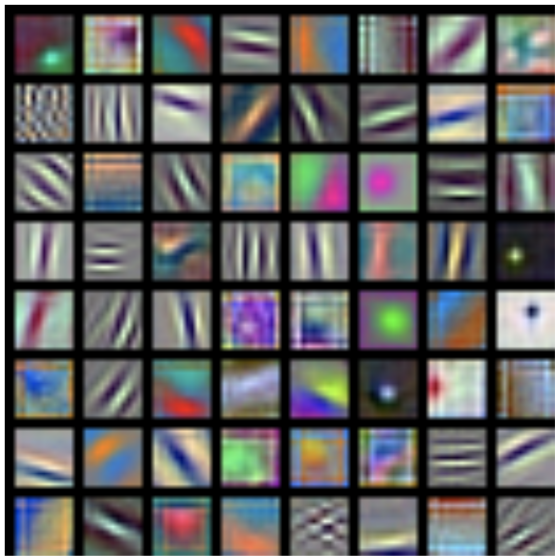
What do convolutional filters learn?



Input:
N x 3 x 32 x 32

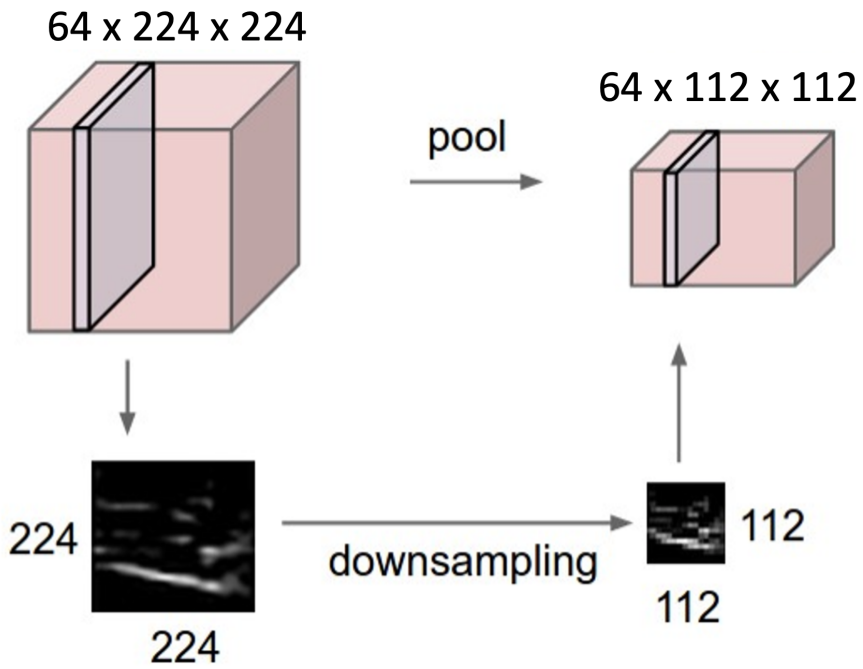
First hidden layer:
N x 6 x 28 x 28

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



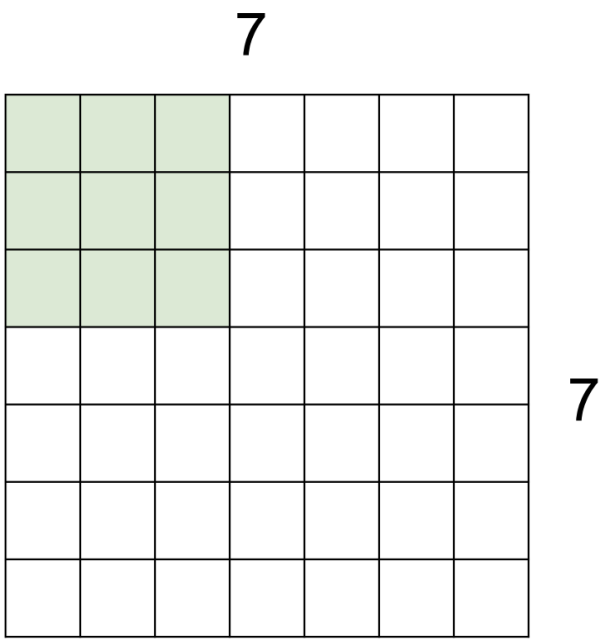
AlexNet: 64 filters, each 3x11x11

Pooling Layers: Another way to downsample



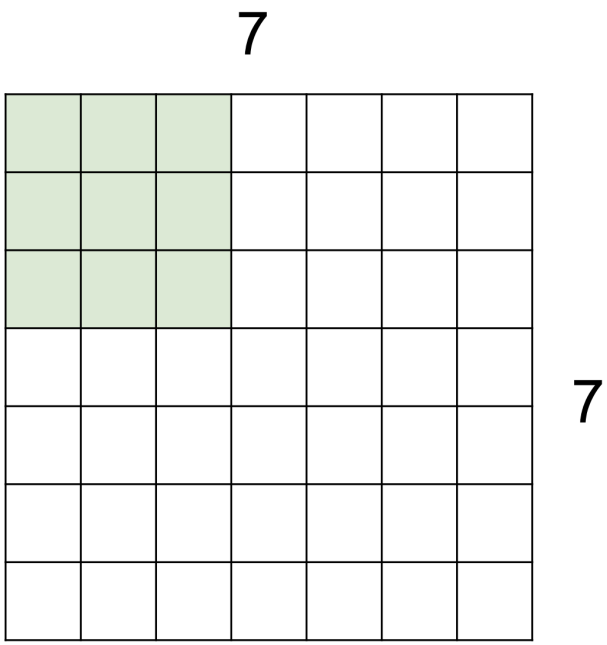
Hyperparameters:
Kernel Size
Stride
Pooling function

Stride



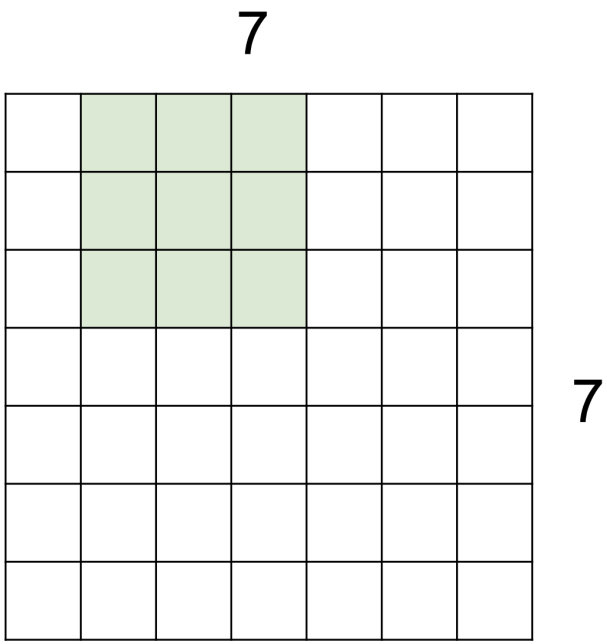
7x7 input (spatially)
assume 3x3 filter

Stride



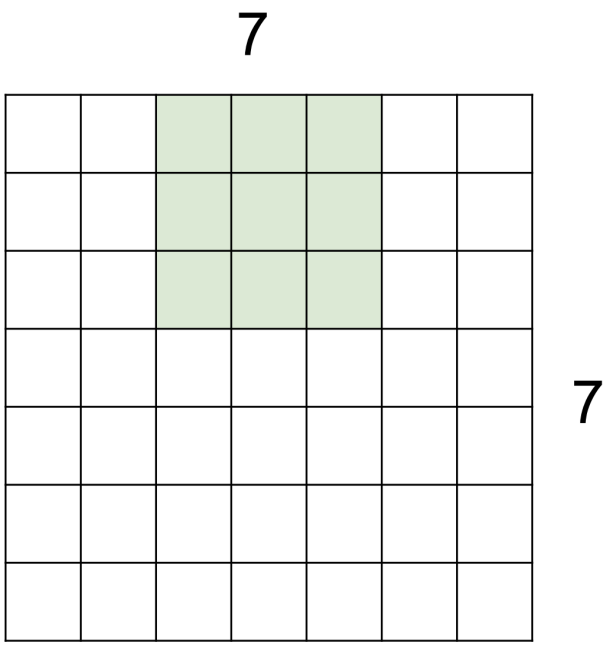
7x7 input (spatially)
assume 3x3 filter

Stride



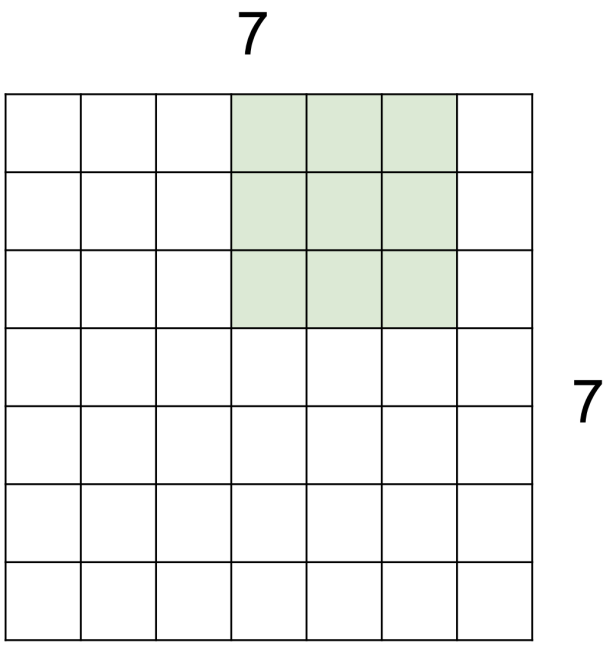
7x7 input (spatially)
assume 3x3 filter

Stride



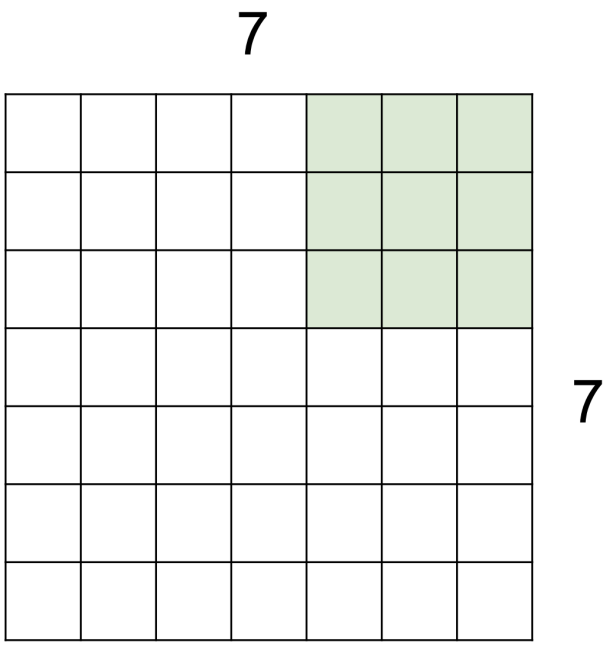
7x7 input (spatially)
assume 3x3 filter

Stride



7x7 input (spatially)
assume 3x3 filter

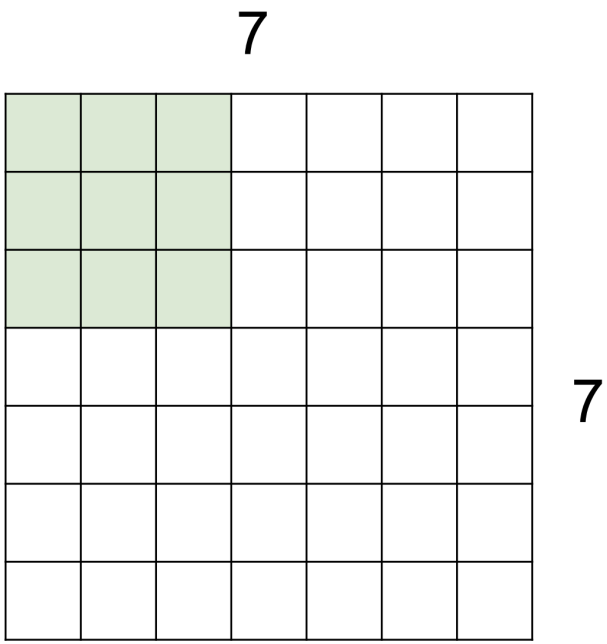
Stride



7x7 input (spatially)
assume 3x3 filter

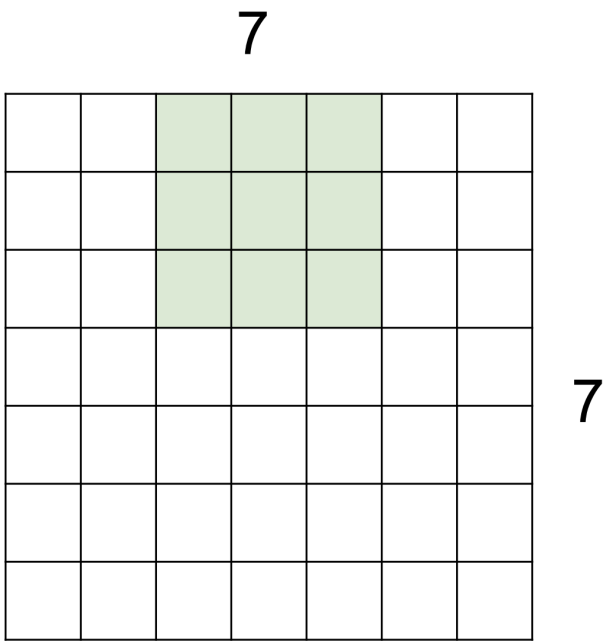
=> 5x5 output

Stride



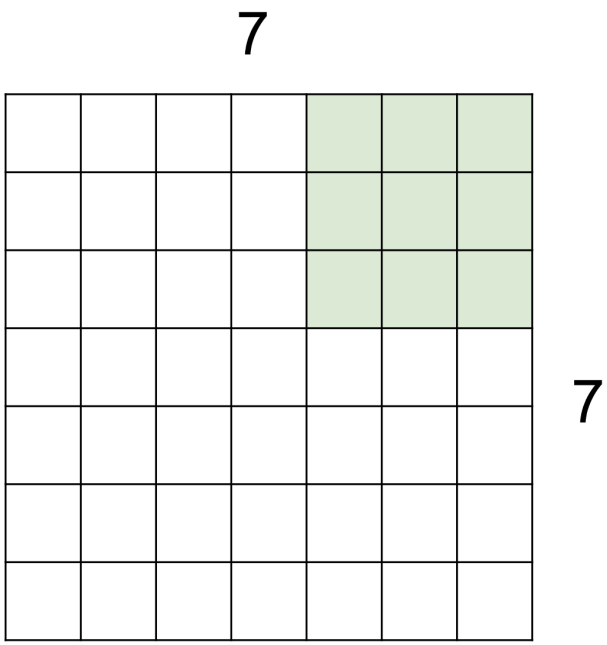
7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

Stride



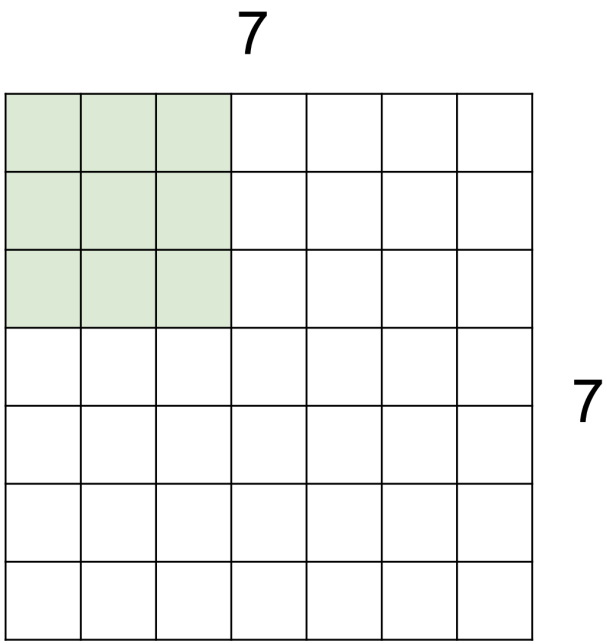
7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

Stride



7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**
=> 3x3 output!

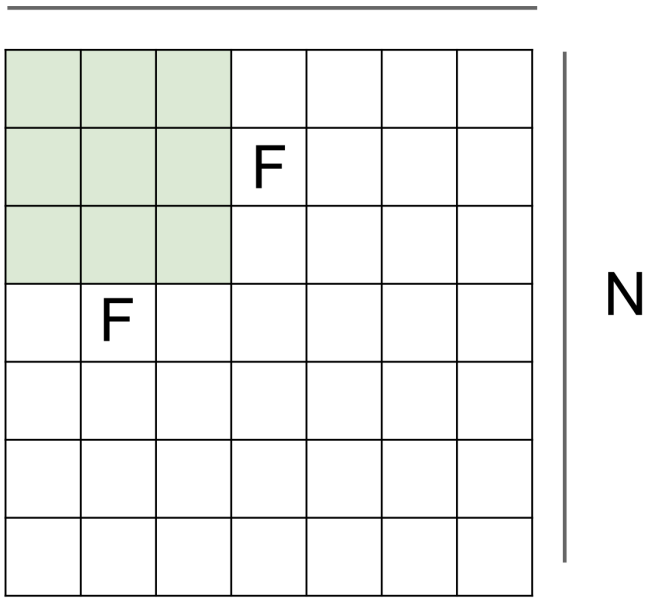
Stride



7x7 input (spatially)
assume 3x3 filter
applied **with stride 3?**

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

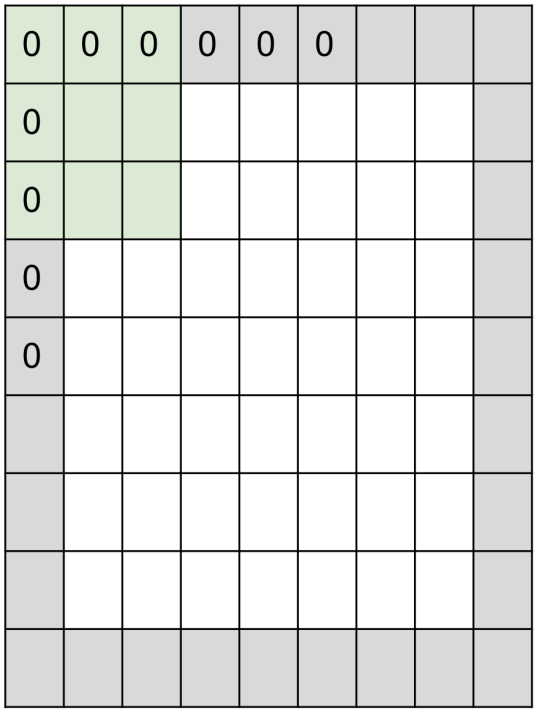
Stride



Output size:
(N - F) / stride + 1

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

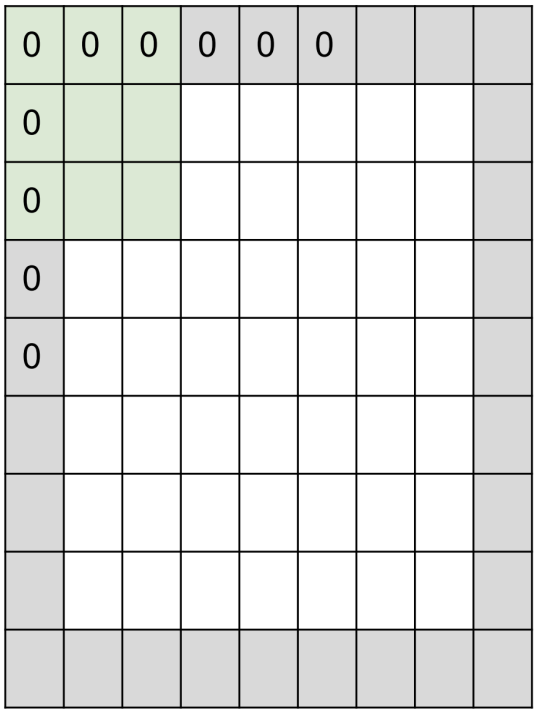
Padding



e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

(recall:)
(N - F) / stride + 1

Padding

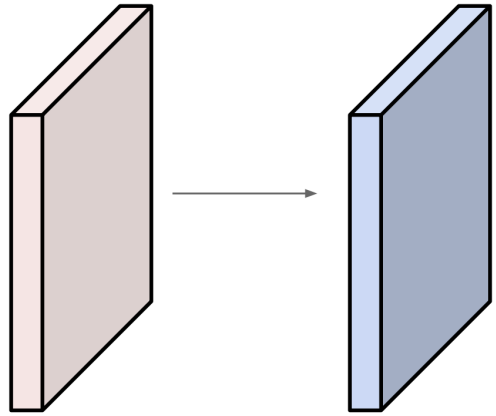


e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

7x7 output!
in general, common to see CONV layers with stride 1, filters of size FxF, and zero-padding with (F-1)/2. (will preserve size spatially)
e.g. F = 3 => zero pad with 1
F = 5 => zero pad with 2
F = 7 => zero pad with 3

Padding

Examples time:

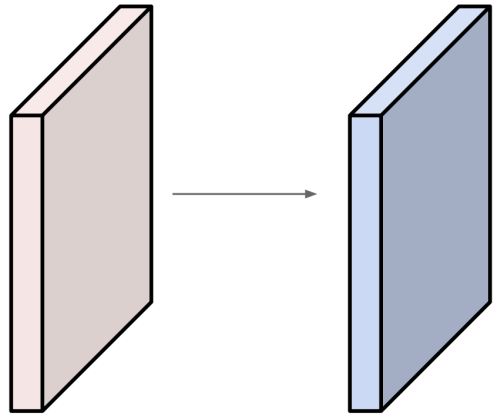


Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

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

Padding

Examples time:

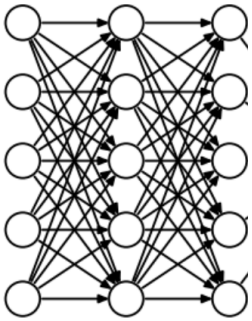


Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

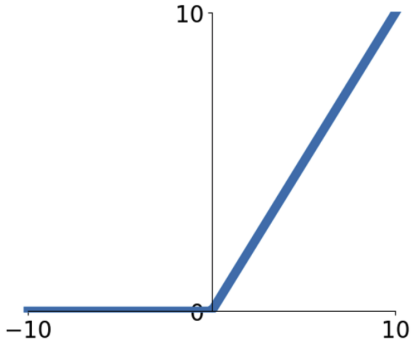
Number of parameters in this layer?
each filter has 5*5*3 + 1 = 76 params (+1 for bias)
=> 76*10 = **760**

Components of a Convolutional Network

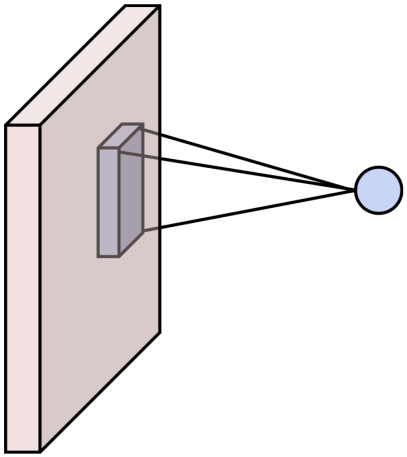
Fully-connected layers



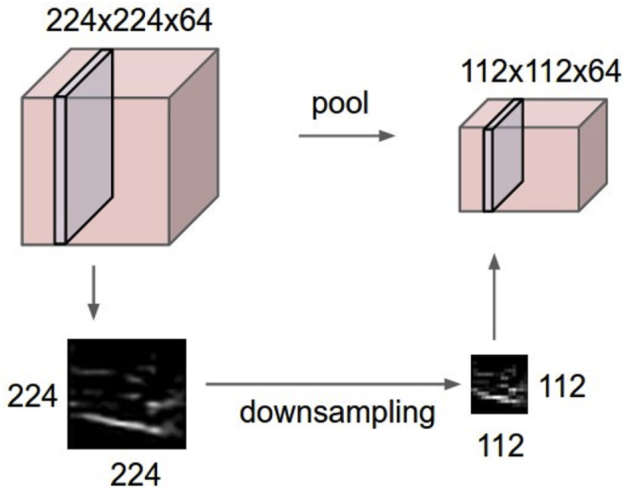
Activation Function



Convolution layers

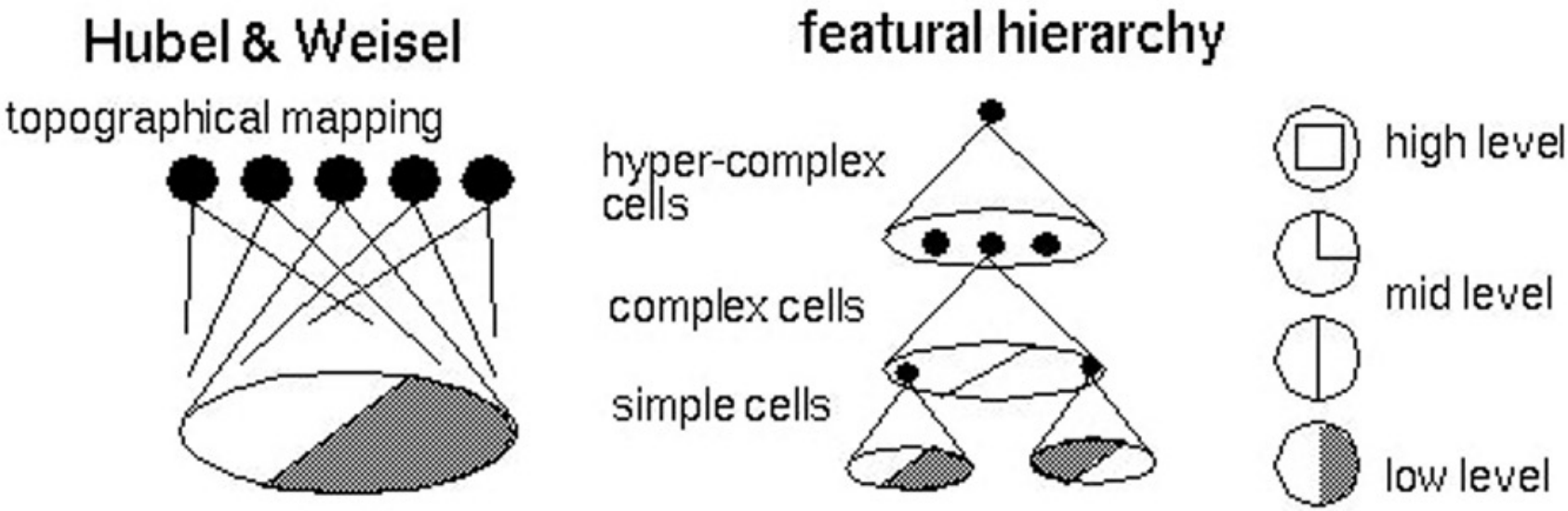


Pooling Layers



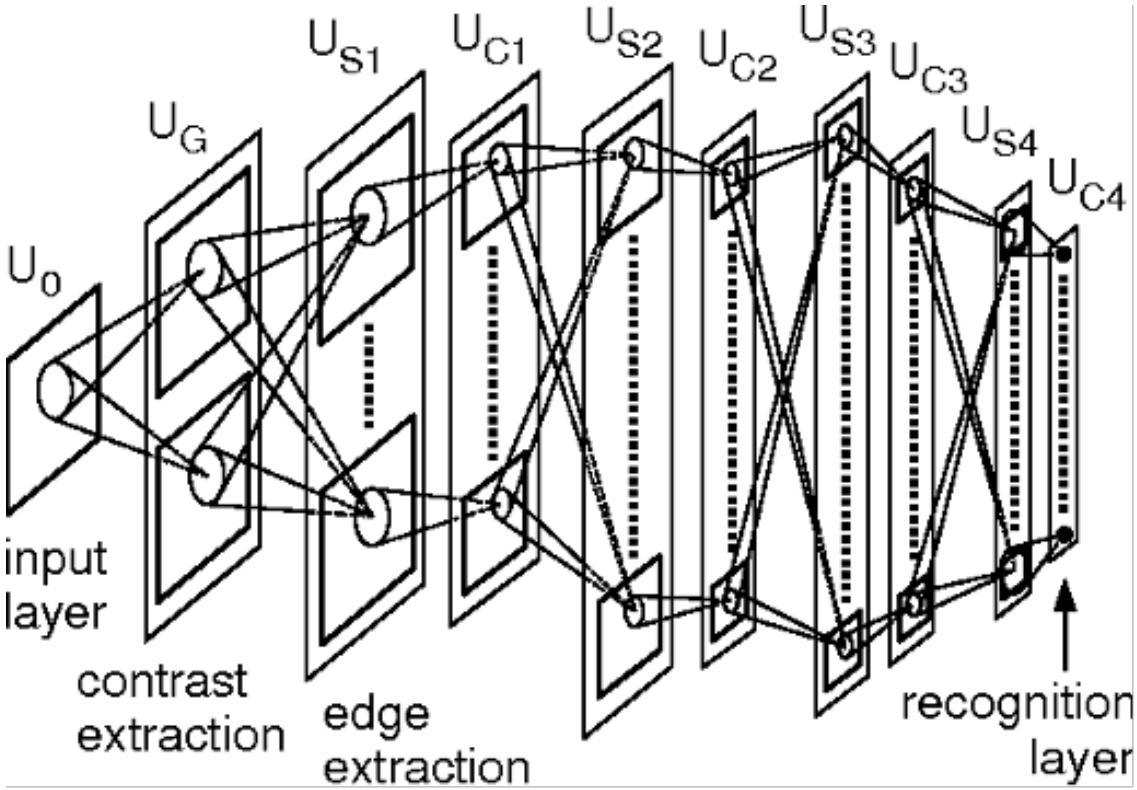
Backstory

- Biological inspiration: D. Hubel and T. Wiesel (1959, 1962, Nobel Prize 1981)
- Visual cortex consists of a hierarchy of simple, complex, and hyper-complex cells



Backstory

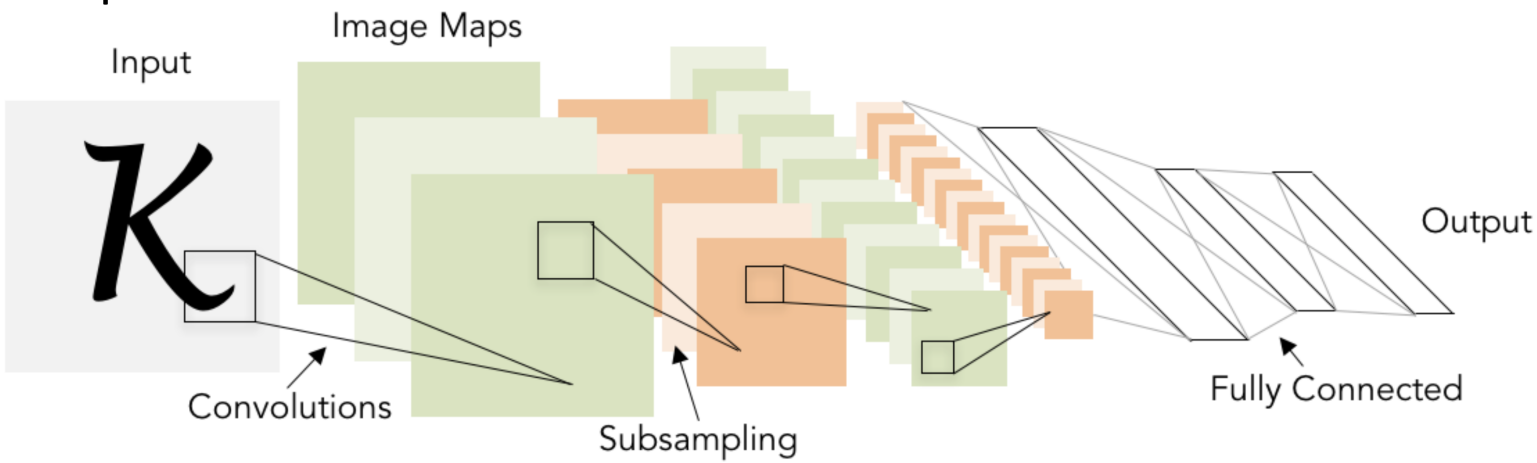
- Neocognitron (Kunihiko Fukushima in 1979)
- Inspired by Hubel and Wiesel, and inspired later CNNs



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

CNN Architectures

AlexNet

CNN Architectures

AlexNet

VGG16

CNN Architectures

AlexNet

VGG16

VGG19

CNN Architectures

AlexNet

VGG16

VGG19

GoogLeNet

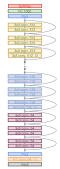
CNN Architectures

AlexNet

VGG16

VGG19

GoogLeNet



ResNet

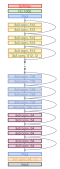
CNN Architectures

AlexNet

VGG16

VGG19

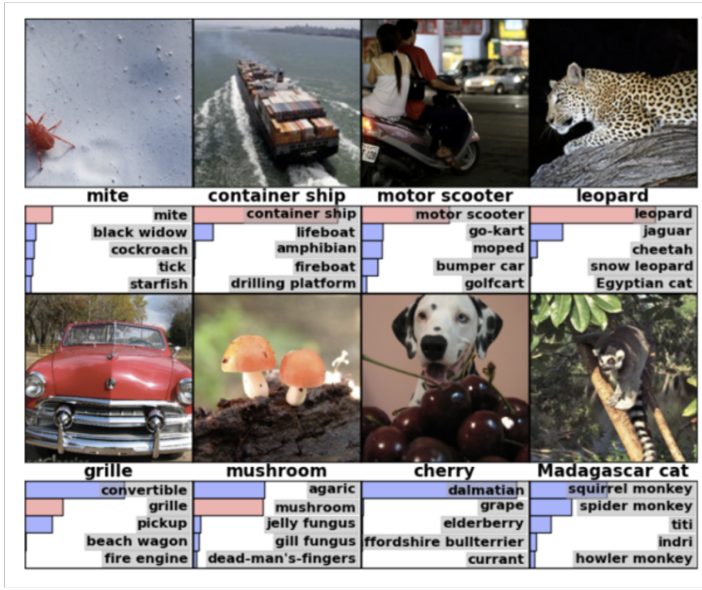
GoogLeNet

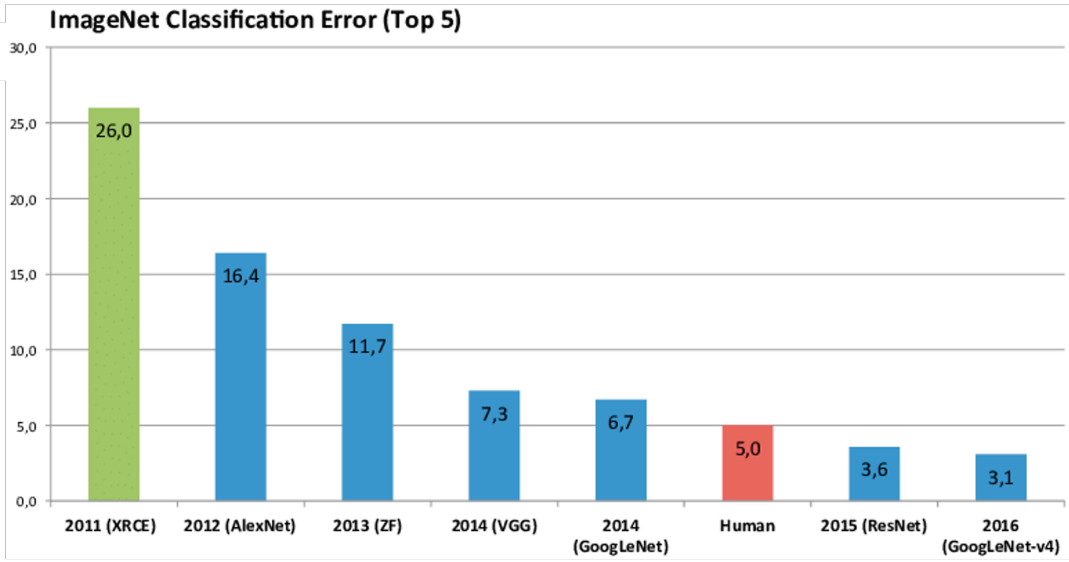


ResNet

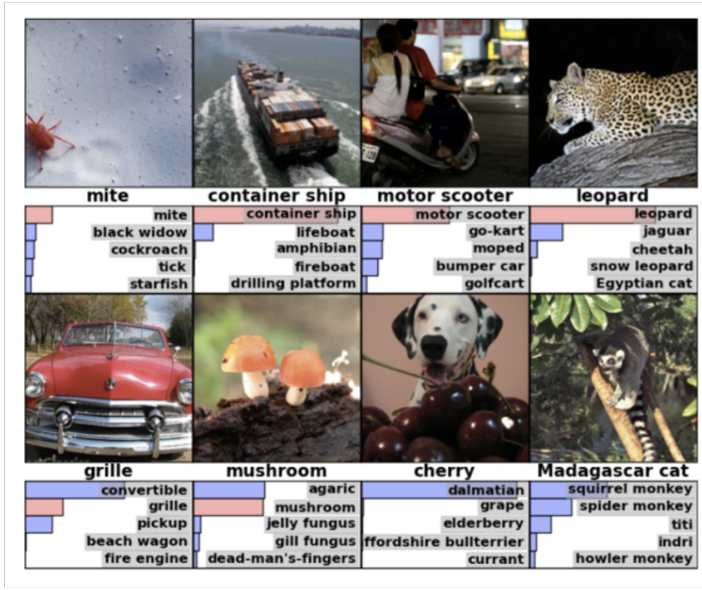
Mo

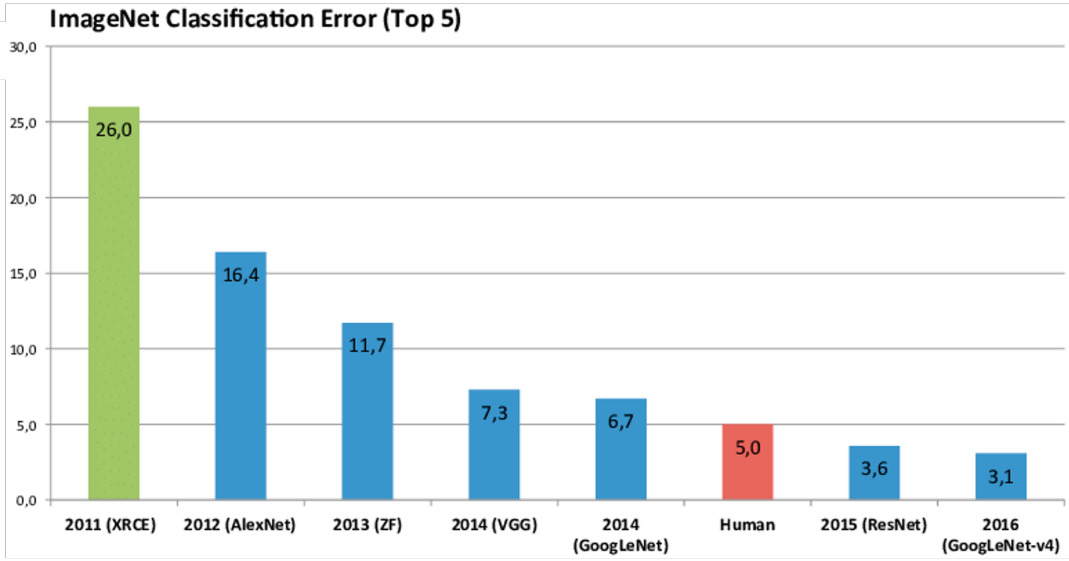
ImageNet

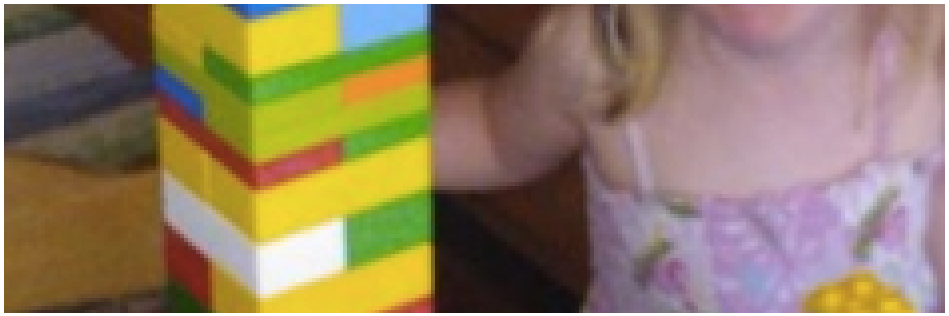

mite
container ship
motor scooter
leopard
mite
black widow
cockroach
tick
starfish
container ship
lifeboat
amphibian
fireboat
drilling platform
motor scooter
go-kart
moped
bumper car
golfcart
leopard
jaguar
cheetah
snow leopard
Egyptian cat
grille
mushroom
cherry
Madagascar cat
convertible
grille
pickup
beach wagon
fire engine
agaric
mushroom
jelly fungus
gill fungus
dead-man's-fingers
dalmatian
grape
elderberry
ffordshire bullterrier
currant
squirrel monkey
spider monkey
titi
indri
howler monkey


ImageNet Classification Error (Top 5)
30,0
25,0
20,0
15,0
10,0
5,0
0,0
26,0
16,4
11,7
7,3
6,7
5,0
3,6
3,1
2011 (XRCE)
2012 (AlexNet)
2013 (ZF)
2014 (VGG)
2014 (GoogLeNet)
Human
2015 (ResNet)
2016 (GoogLeNet-v4)

ImageNet


mite
container ship
motor scooter
leopard
mite
black widow
cockroach
tick
starfish
container ship
lifeboat
amphibian
fireboat
drilling platform
motor scooter
go-kart
moped
bumper car
golfcart
leopard
jaguar
cheetah
snow leopard
Egyptian cat
grille
mushroom
cherry
Madagascar cat
convertible
grille
pickup
beach wagon
fire engine
agaric
mushroom
jelly fungus
gill fungus
dead-man's-fingers
dalmatian
grape
elderberry
ffordshire bullterrier
currant
squirrel monkey
spider monkey
titi
indri
howler monkey


ImageNet Classification Error (Top 5)
30,0
25,0
20,0
15,0
10,0
5,0
0,0
26,0
16,4
11,7
7,3
6,7
5,0
3,6
3,1
2011 (XRCE)
2012 (AlexNet)
2013 (ZF)
2014 (VGG)
2014 (GoogLeNet)
Human
2015 (ResNet)
2016 (GoogLeNet-v4)



Which Architectures should I use?

- For most problems you should use an off-the-shelf architecture (with proven record after many "graduate student descent"); don't try to design your own.

Which Architectures should I use?

- For most problems you should use an off-the-shelf architecture (with proven record after many “graduate student descent”); don’t try to design your own.
- If you just care about accuracy, **ResNet-50** or **ResNet-101** are great choices

Which Architectures should I use?

- For most problems you should use an off-the-shelf architecture (with proven record after many “graduate student descent”); don’t try to design your own.
- If you just care about accuracy, **ResNet-50** or **ResNet-101** are great choices
- If you want an efficient network (real-time, run on mobile, etc), try **MobileNets** and **ShuffleNets**

Which Architectures should I use?

- For most problems you should use an off-the-shelf architecture (with proven record after many "graduate student descent"); don't try to design your own.
- If you just care about accuracy, **ResNet-50** or **ResNet-101** are great choices
- If you want an efficient network (real-time, run on mobile, etc), try **MobileNets** and **ShuffleNets**
- Start from pretrained models.
  `https://pytorch.org/serve/model_zoo.html`

Demo time: `https://huggingface.co/spaces/Xenova/object-detection-webcam_testing`

Important topics of this lecture

Important topics of this lecture

- Getting to know CNNs and its variants

Important topics of this lecture

- Getting to know CNNs and its variants
- Getting to know their use

Important topics of this lecture

- Getting to know CNNs and its variants
- Getting to know their use
- Contrasting CNNs to MLPs

Important topics of this lecture

- Getting to know CNNs and its variants
- Getting to know their use
- Contrasting CNNs to MLPs

Important topics of this lecture

- Getting to know CNNs and its variants
- Getting to know their use
- Contrasting CNNs to MLPs

Next up:

Recurrent neural networks