Convolutional Neural Networks (CNNs)

CMSC 478

UMBC

Outline

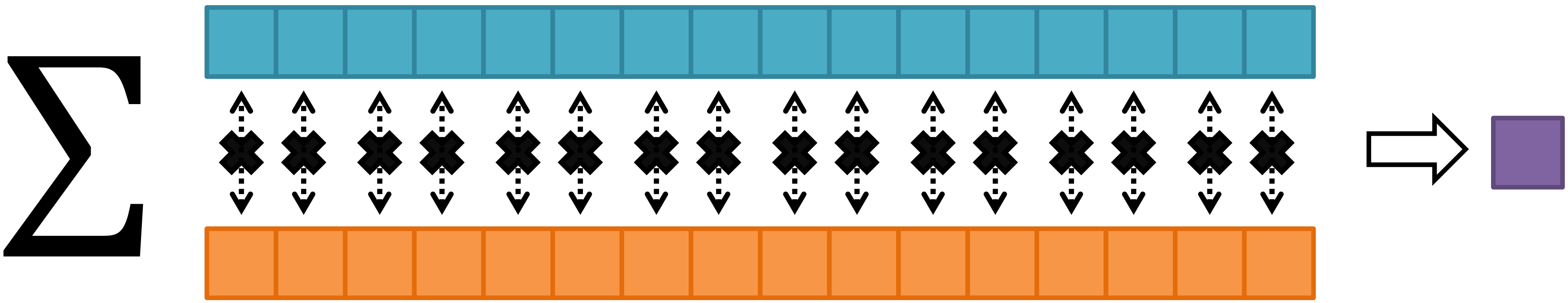
Convolutional Neural Networks

- What *is* a convolution?
- Multidimensional Convolutions
- Typical Convnet Operations
- Deep convnets

Recurrent Neural Networks

- Types of recurrence
- A basic recurrent cell
- BPTT: Backpropagation through time

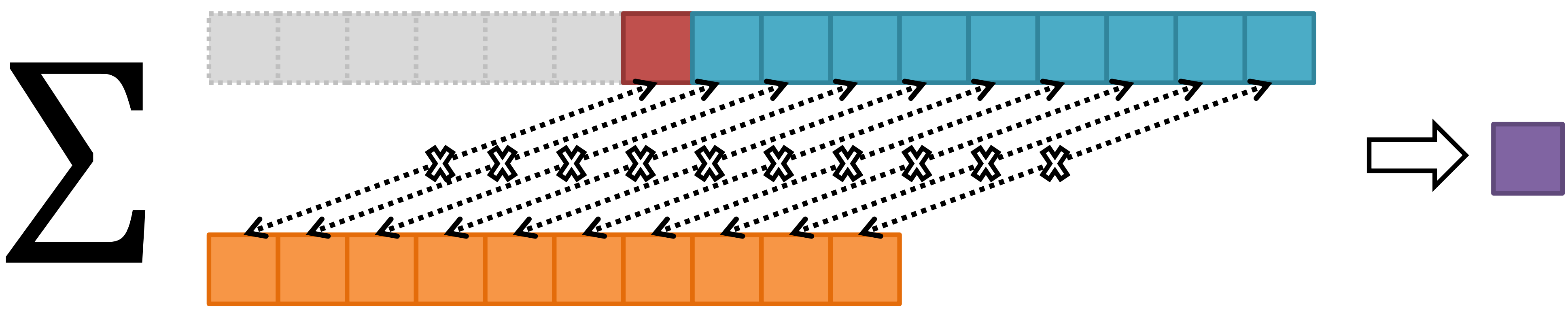
Dot Product



$$x^T y = \sum_k x_k y_k$$

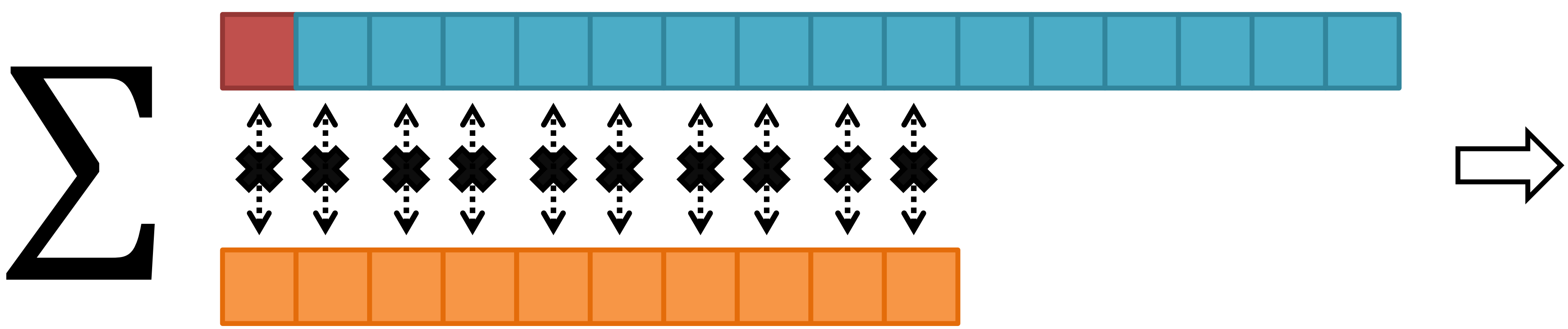
Convolution: Modified Dot Product Around a Point

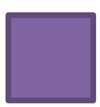
$$(x^T y)_i = \sum_{k<K} x_{k+i} y_k$$



Convolution: Modified Dot Product Around a Point

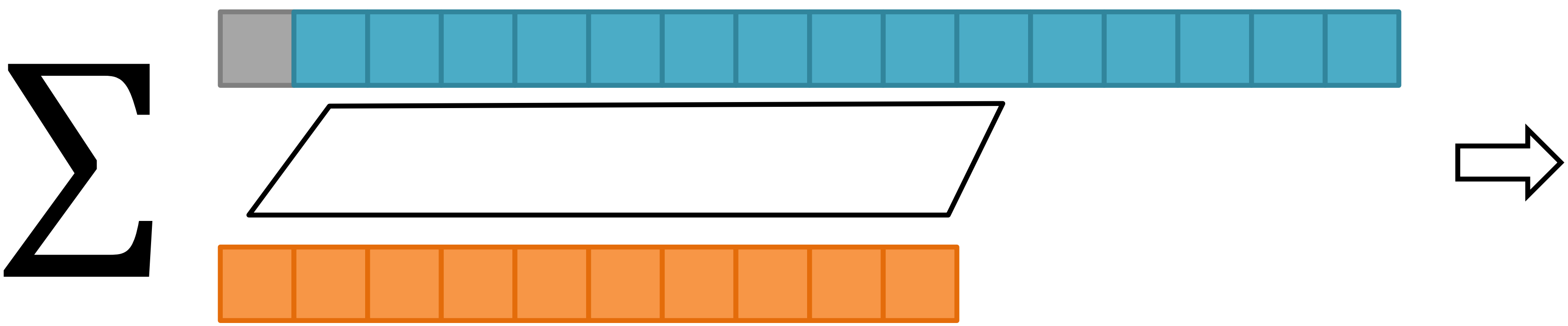
$$(x^T y)_i = \sum_k x_{k+i} y_k$$

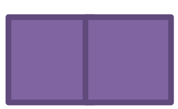


$(x \star y)[i] =$ 

Convolution: Modified Dot Product Around a Point

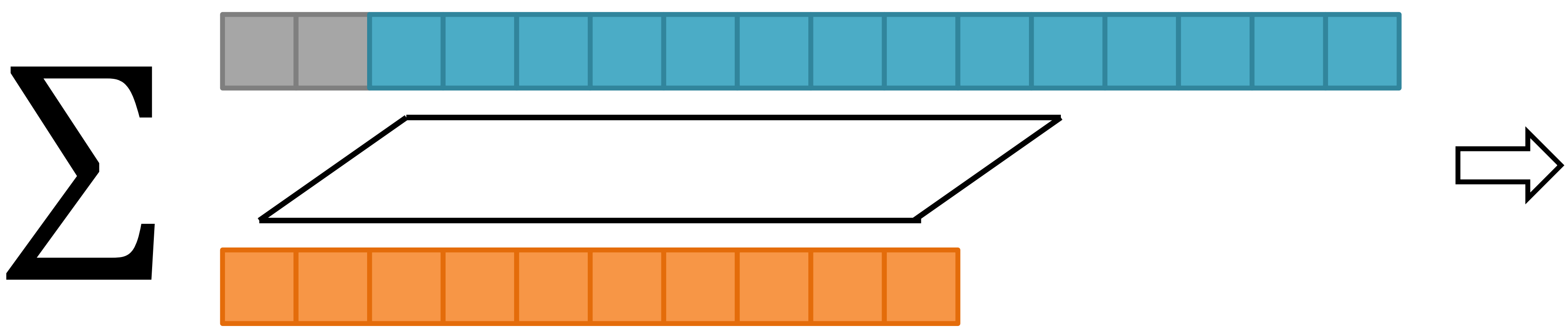
$$(x^T y)_i = \sum_k x_{k+i} y_k$$



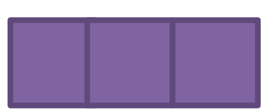
$(x \star y)[i] =$ 

Convolution: Modified Dot Product Around a Point

$$(x^T y)_i = \sum_k x_{k+i} y_k$$

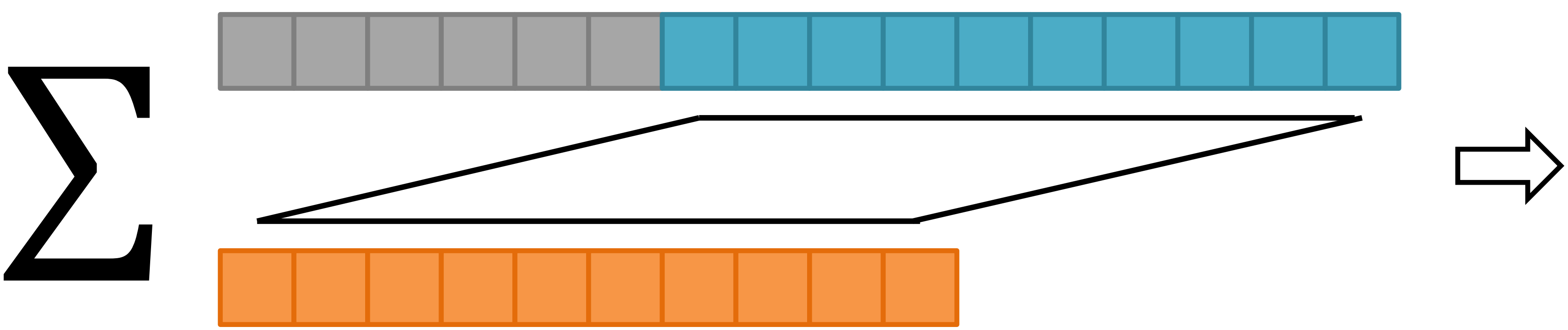


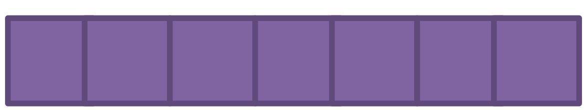
$$(x \star y)[i] =$$



Convolution: Modified Dot Product Around a Point

$$(x^T y)_i = \sum_k x_{k+i} y_k$$



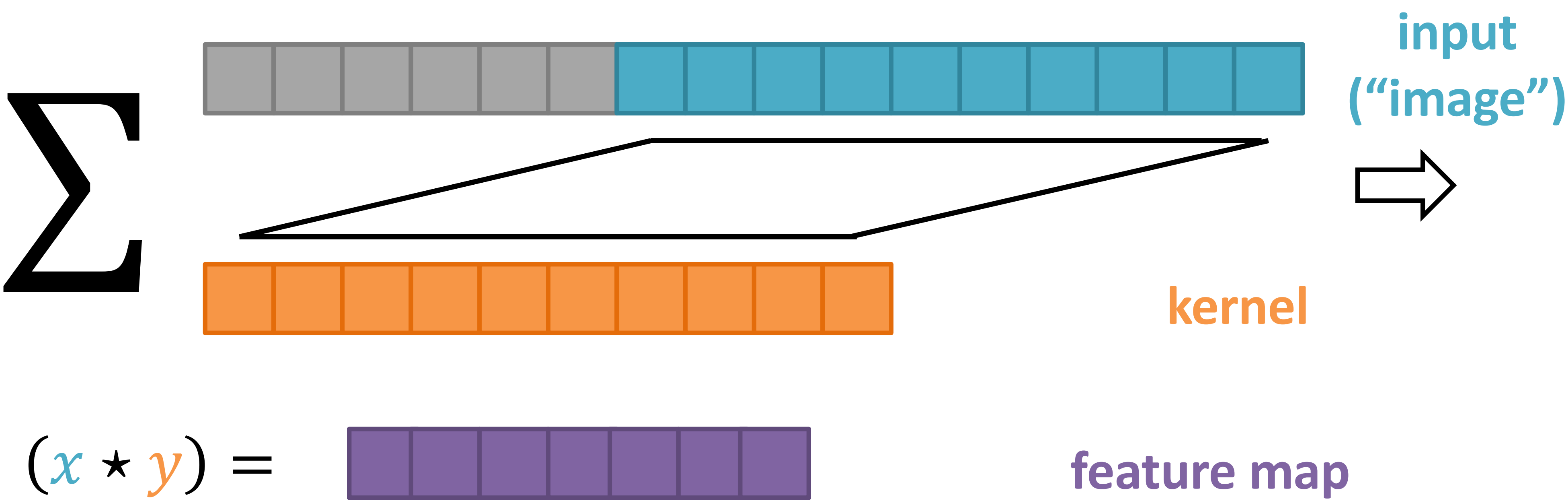
$(x \star y)[i] =$ 

Convolution/cross-correlation

Convolution: Modified Dot Product Around a Point

$$(x^T y)_i = \sum_k x_{k+i} y_k$$

1-D convolution



$$(x \star y) =$$

Convolution/cross-correlation

Outline

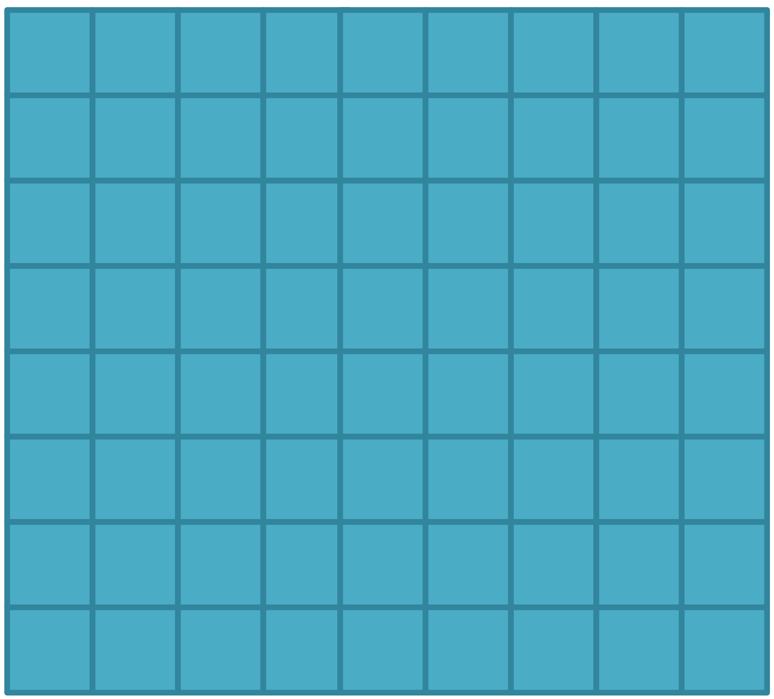
Convolutional Neural Networks

- What *is* a convolution?
- Multidimensional Convolutions
- Typical Convnet Operations
- Deep convnets

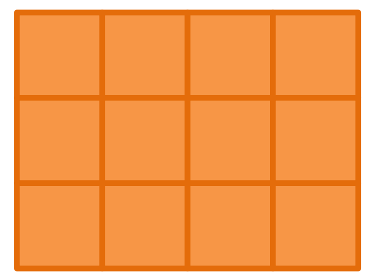
Recurrent Neural Networks

- Types of recurrence
- A basic recurrent cell
- BPTT: Backpropagation through time

2-D Convolution



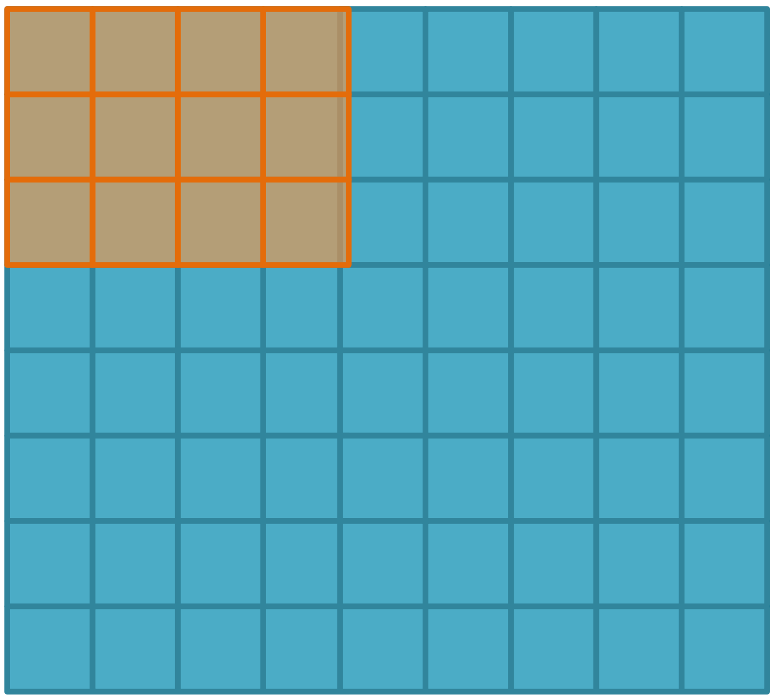
input
("image")



kernel

width: shape of the kernel
(often square)

2-D Convolution

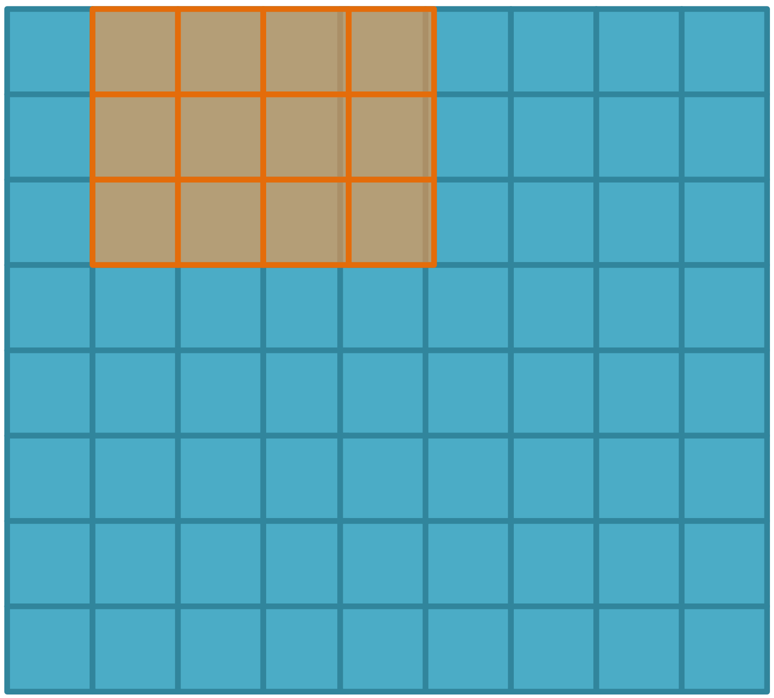


input
("image")

stride(s): how many spaces to move the kernel

width: shape of the kernel (often square)

2-D Convolution

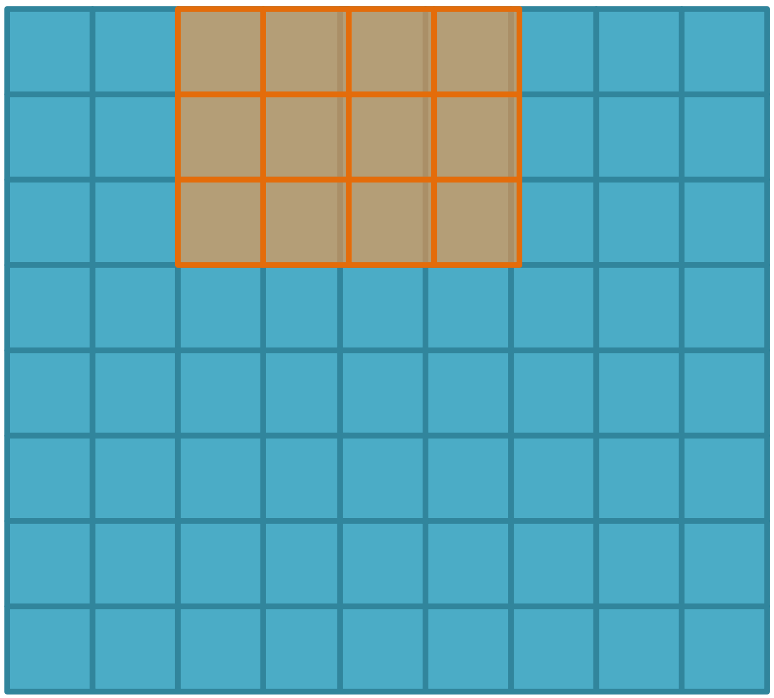


stride(s): how many spaces to move the kernel

stride=1

width: shape of the kernel (often square)

2-D Convolution



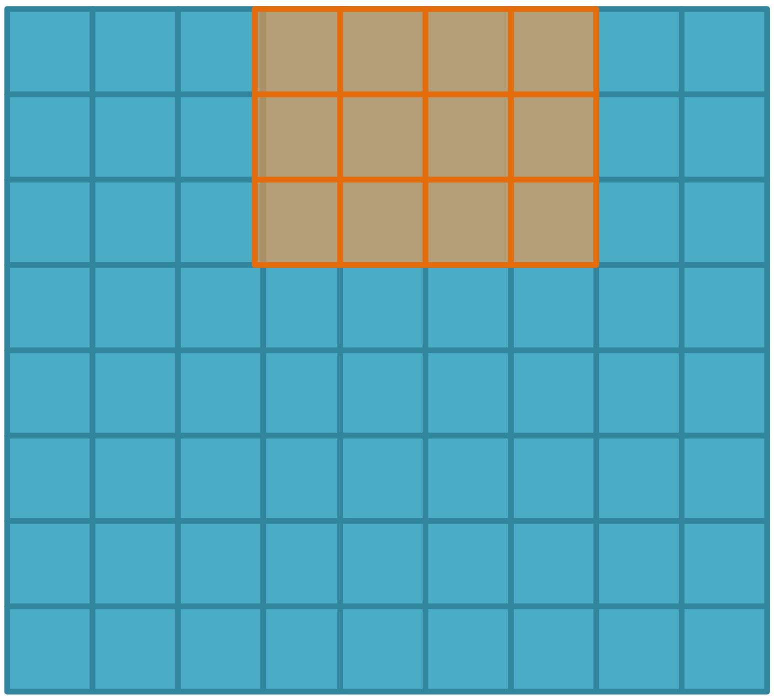
input
("image")

stride(s): how many spaces to move the kernel

stride=1

width: shape of the kernel (often square)

2-D Convolution



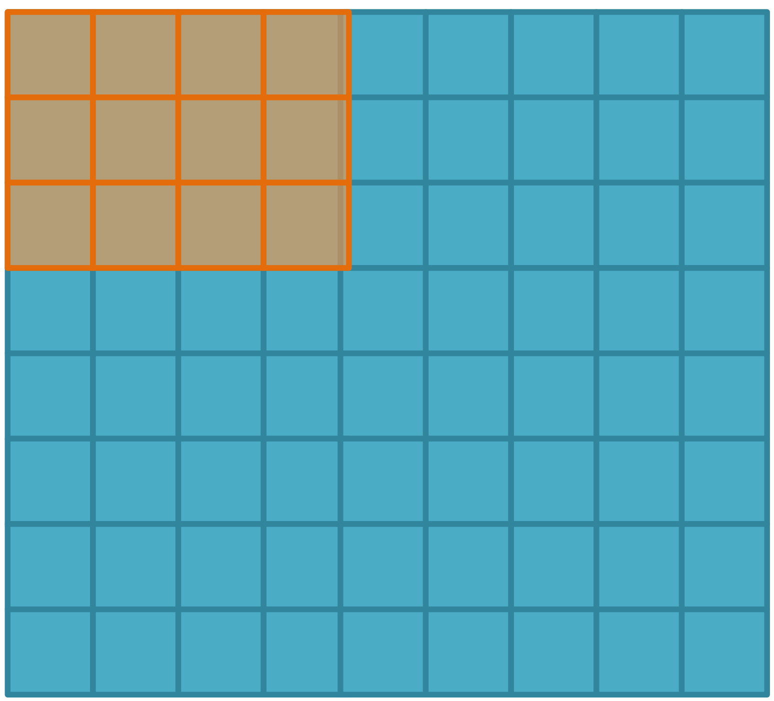
input
("image")

stride(s): how many spaces to move the kernel

stride=1

width: shape of the kernel (often square)

2-D Convolution



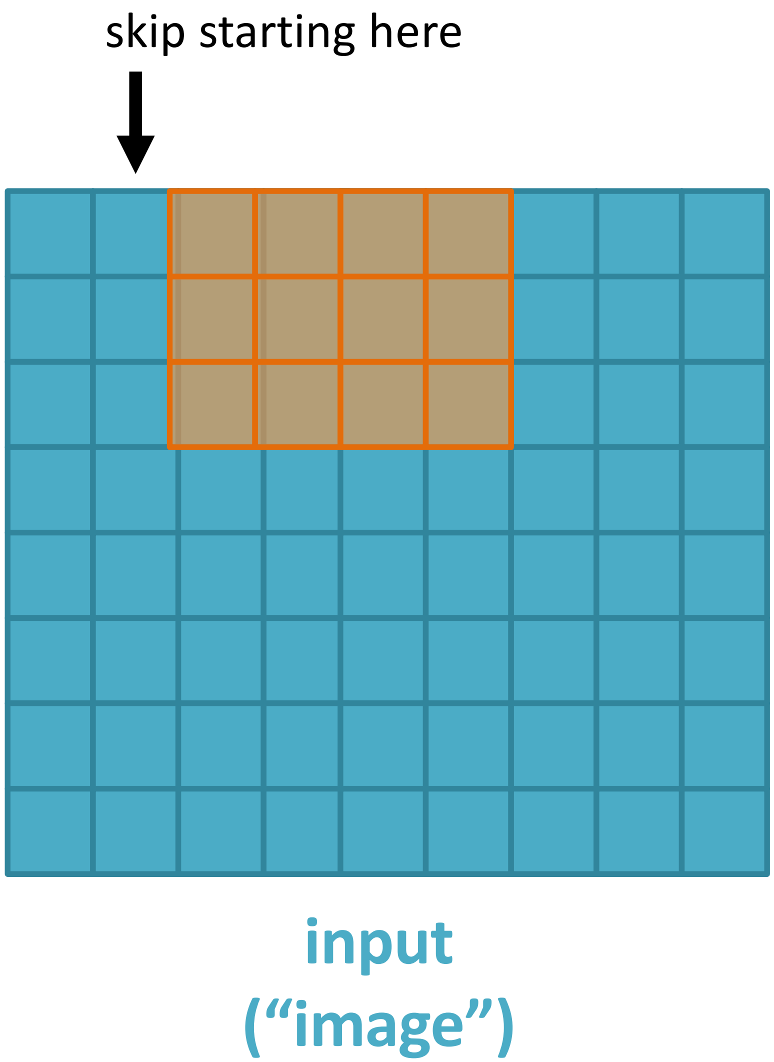
input
(“image”)

stride(s): how many spaces to move the kernel

stride=2

width: shape of the kernel (often square)

2-D Convolution

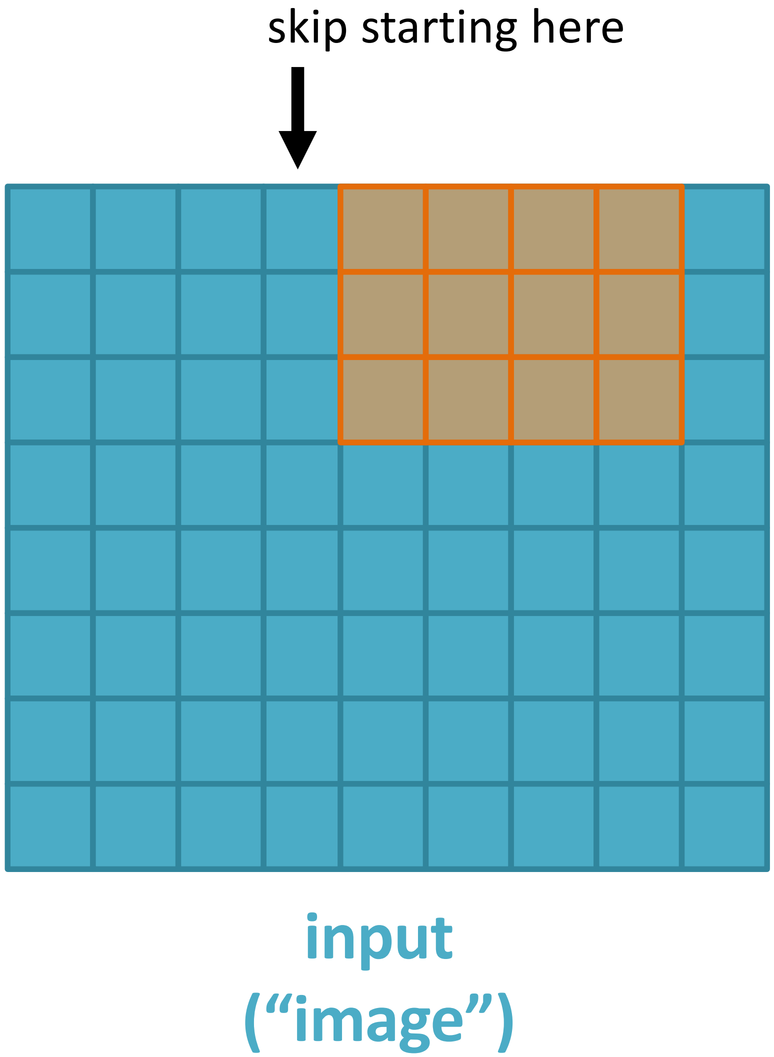


stride(s): how many spaces to move the kernel

stride=2

width: shape of the kernel (often square)

2-D Convolution

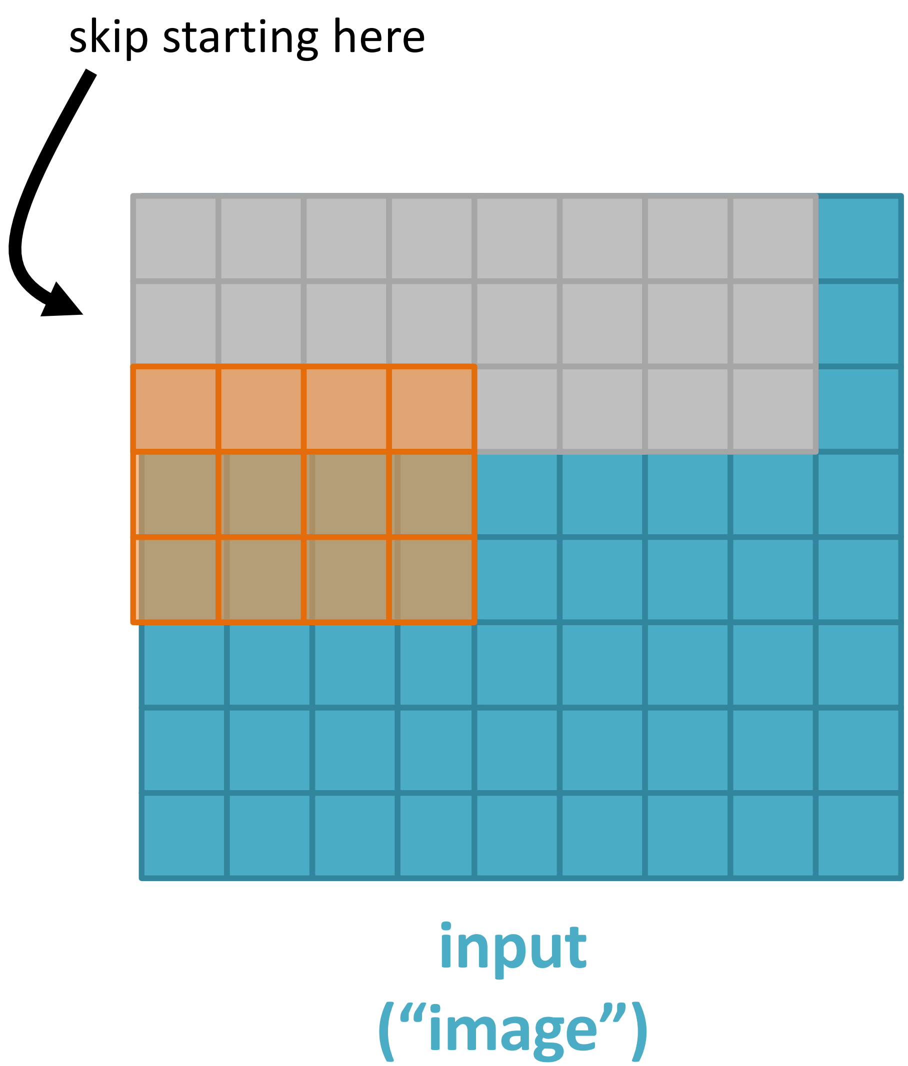


stride(s): how many spaces to move the kernel

stride=2

width: shape of the kernel (often square)

2-D Convolution

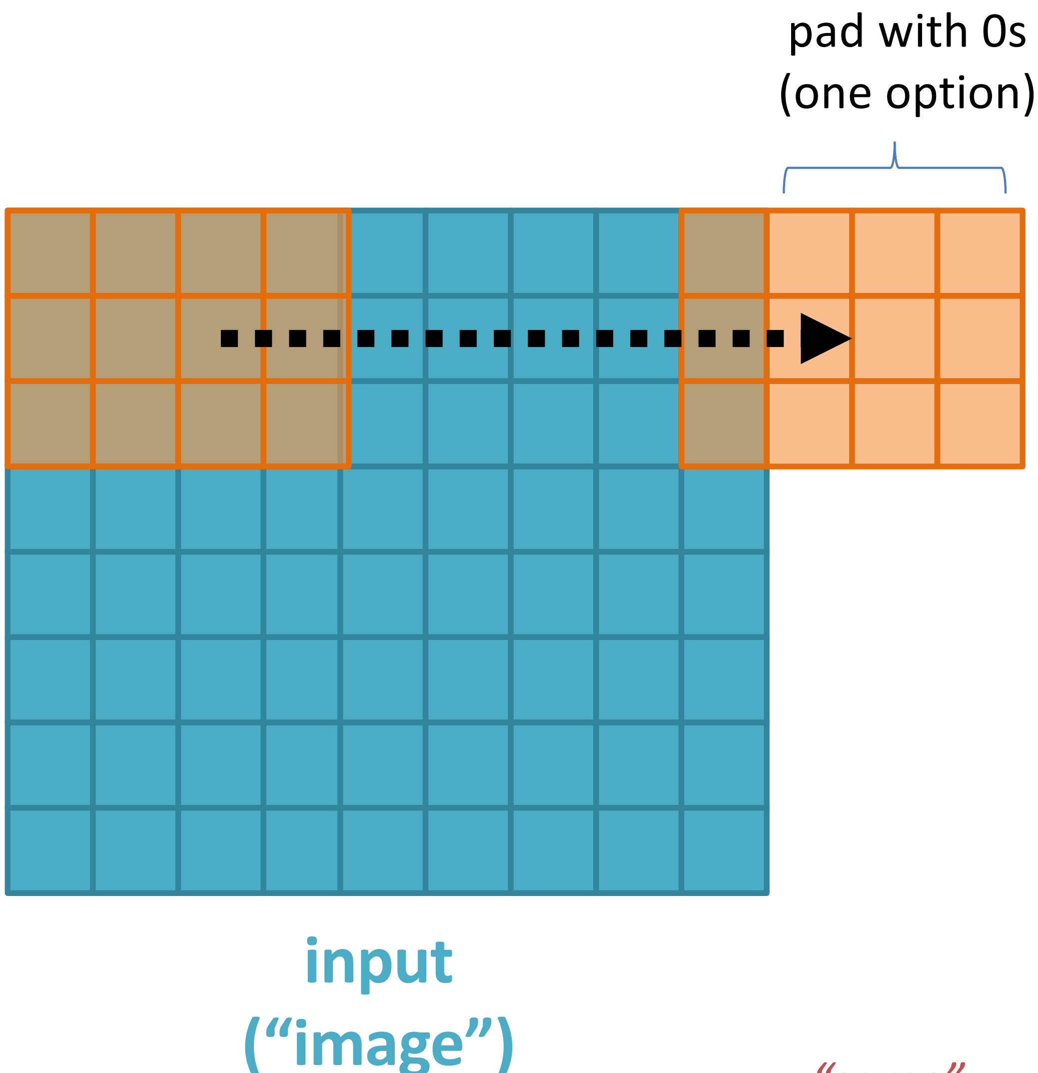


stride(s): how many spaces to move the kernel

stride=2

width: shape of the kernel (often square)

2-D Convolution



stride(s): how many spaces to move the kernel

padding: how to handle input/kernel shape mismatches

width: shape of the kernel (often square)

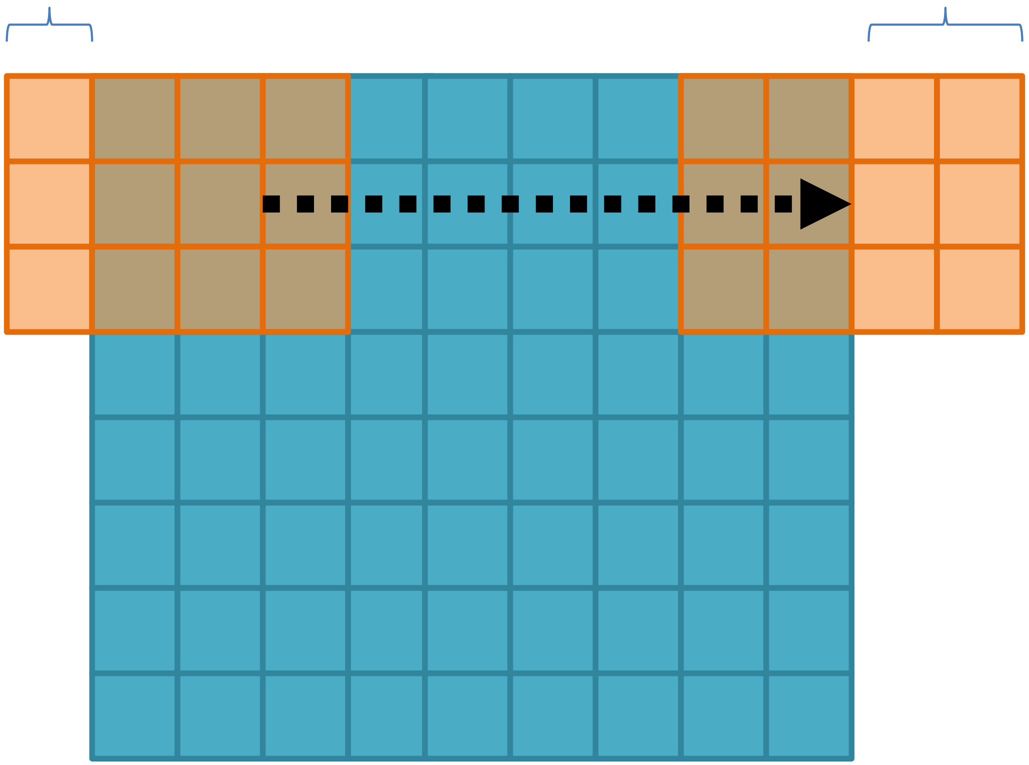
"same": input.shape == output.shape

"different": input.shape ≠ output.shape

2-D Convolution

pad with 0s
(another option)

pad with 0s
(another option)



input
("image")

stride(s): how many spaces to move the kernel

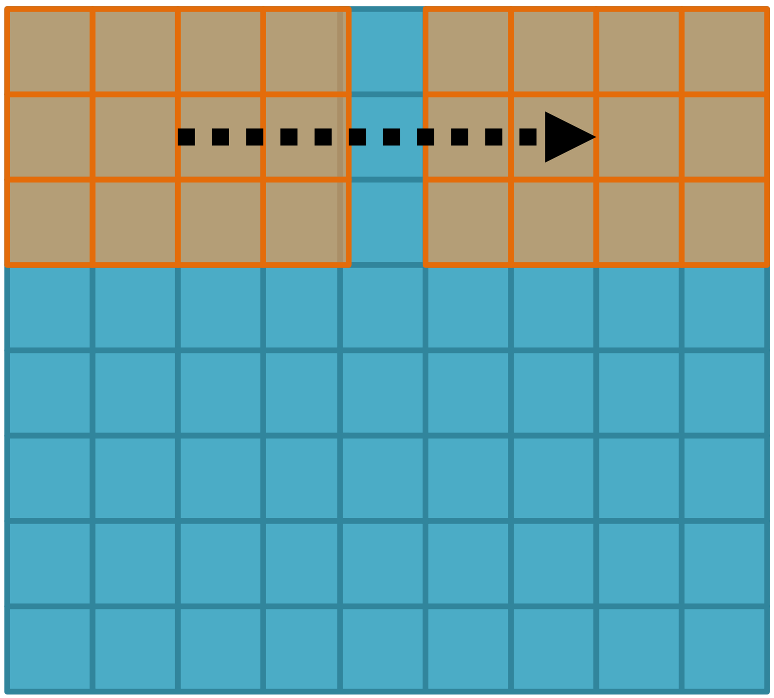
padding: how to handle input/kernel shape mismatches

width: shape of the kernel (often square)

"same":
input.shape == output.shape

"different":
input.shape ≠ output.shape

2-D Convolution



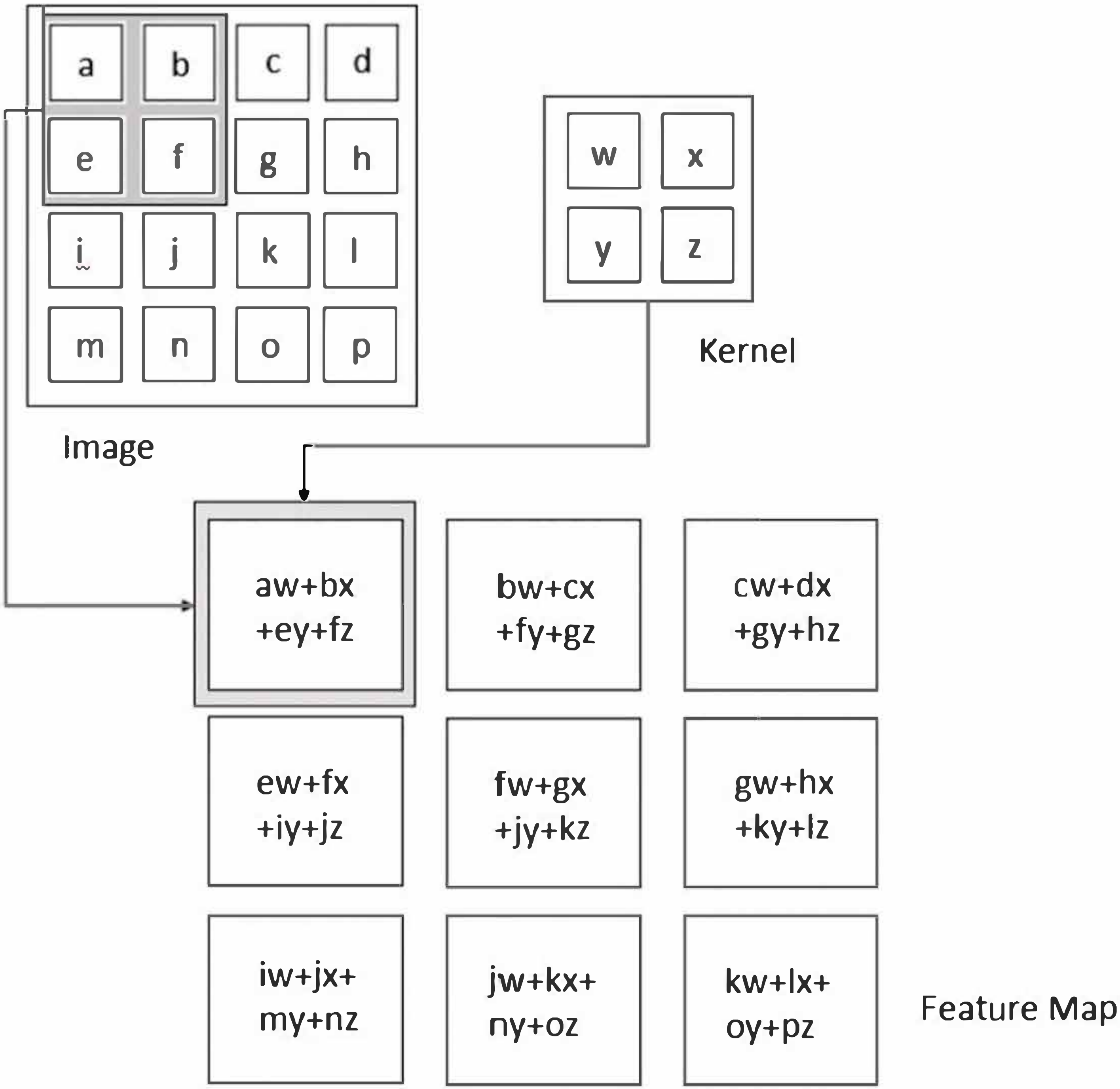
stride(s): how many spaces to move the kernel

padding: how to handle input/kernel shape mismatches

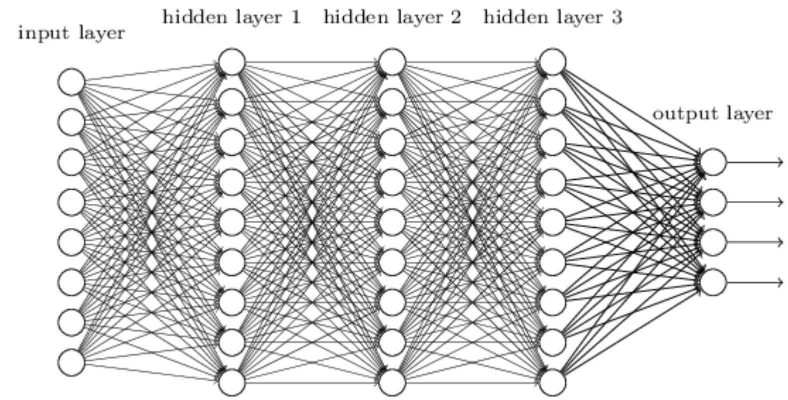
width: shape of the kernel (often square)

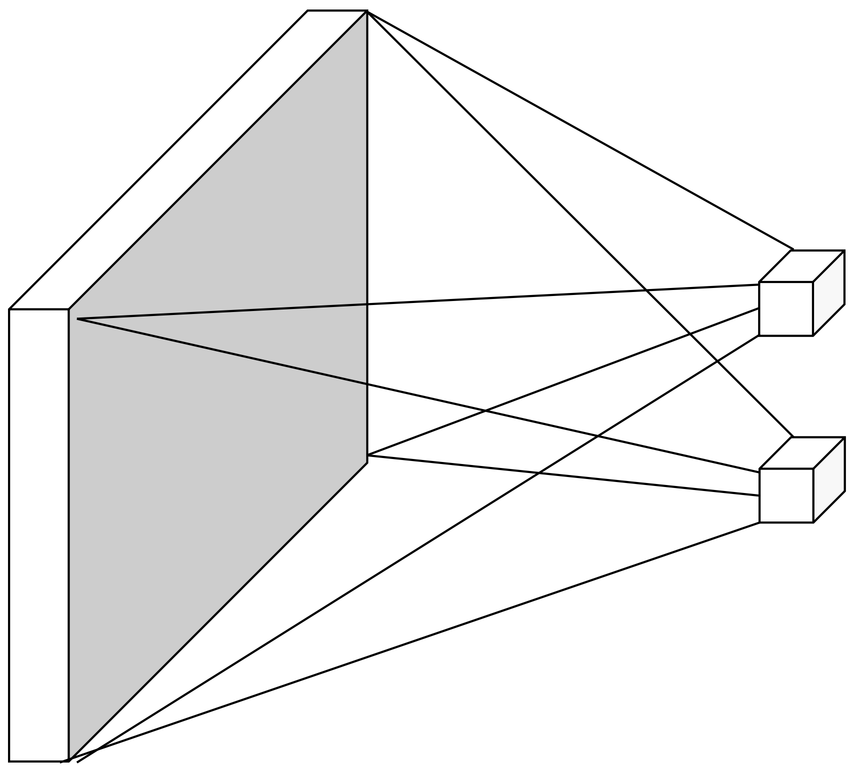
"same":
input.shape == output.shape

"different":
input.shape ≠ output.shape


a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
Image
w
x
y
z
Kernel
aw+bx
+ey+fz
bw+cx
+fy+gz
cw+dx
+gy+hz
ew+fx
+iy+jz
fw+gx
+jy+kz
gw+hx
+ky+lz
iw+jx+
my+nz
jw+kx+
ny+oz
kw+lx+
oy+pz
Feature Map

From fully connected to convolutional networks



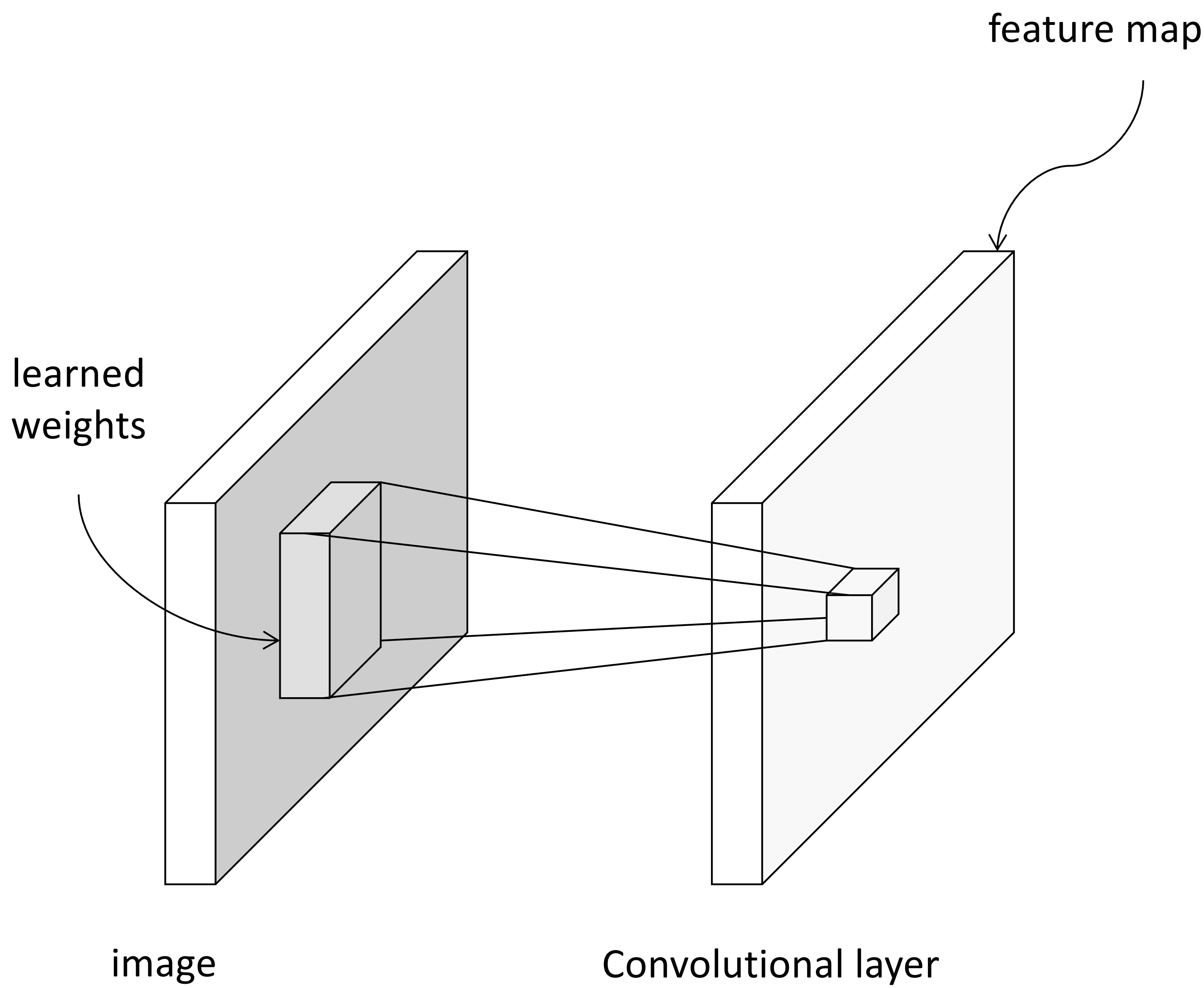


image

Fully connected layer

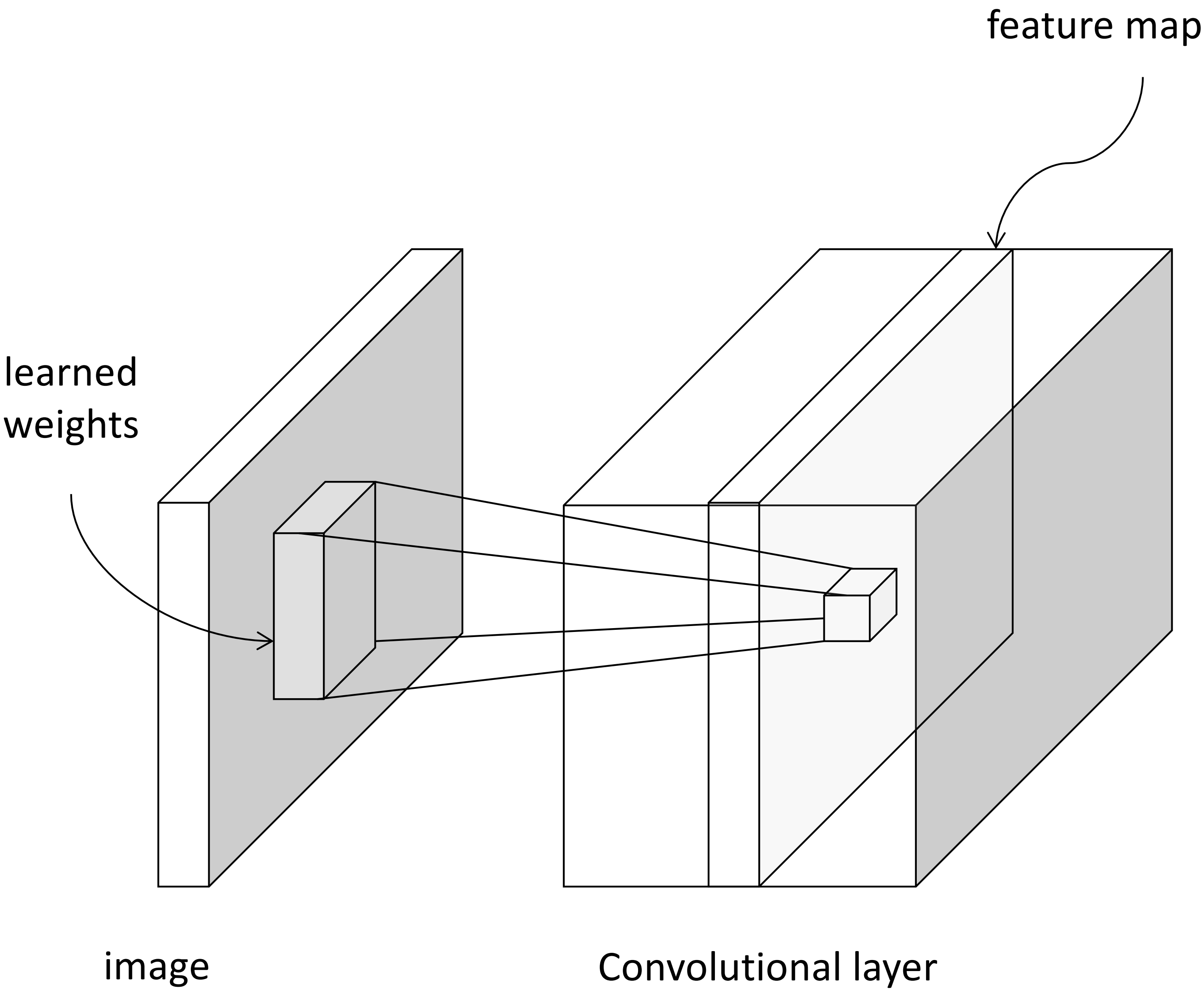
Slide credit: Svetlana Lazebnik

From fully connected to convolutional networks



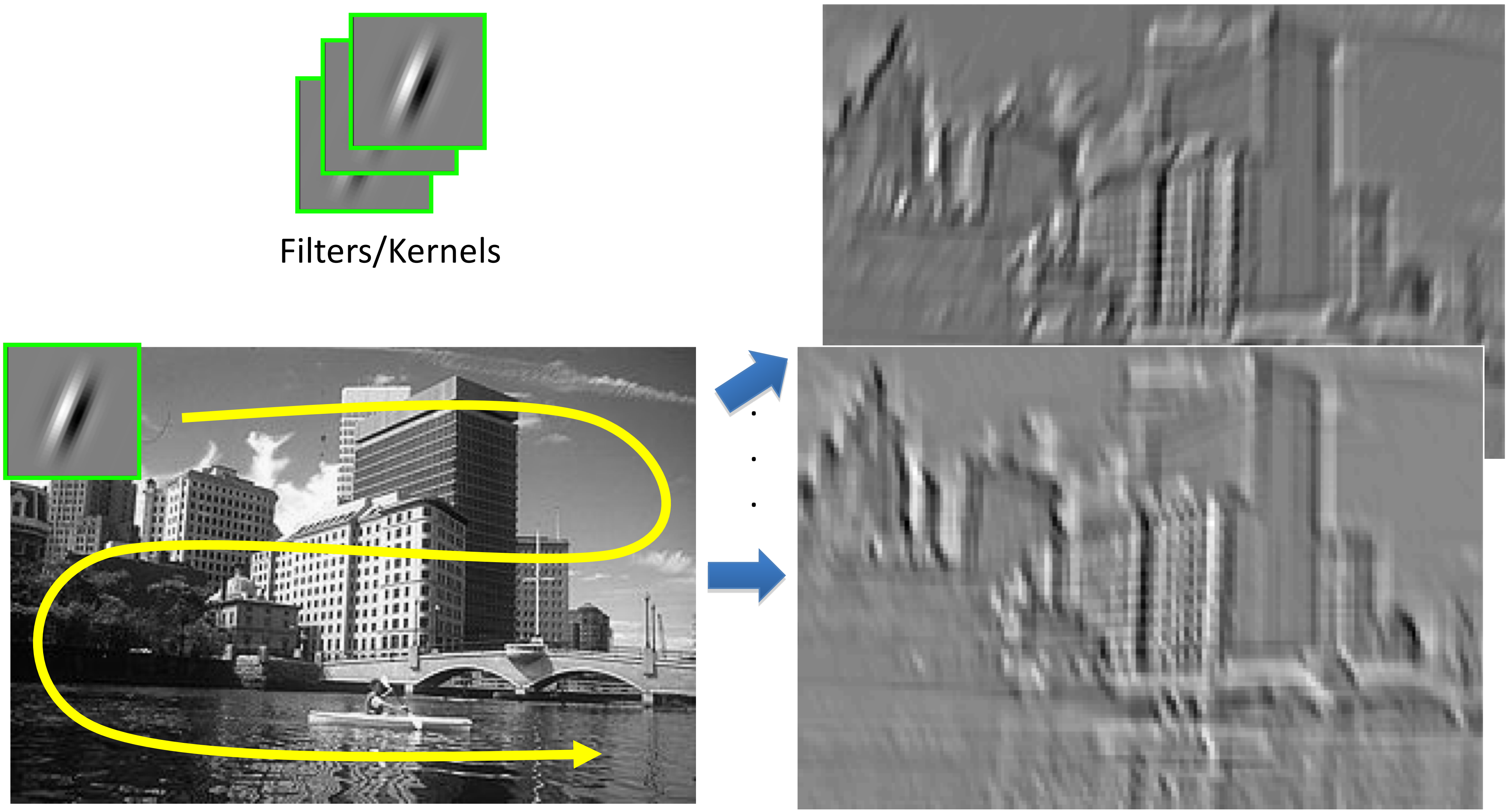
Slide credit: Svetlana Lazebnik

From fully connected to convolutional networks



Slide credit: Svetlana Lazebnik

Convolution as feature extraction

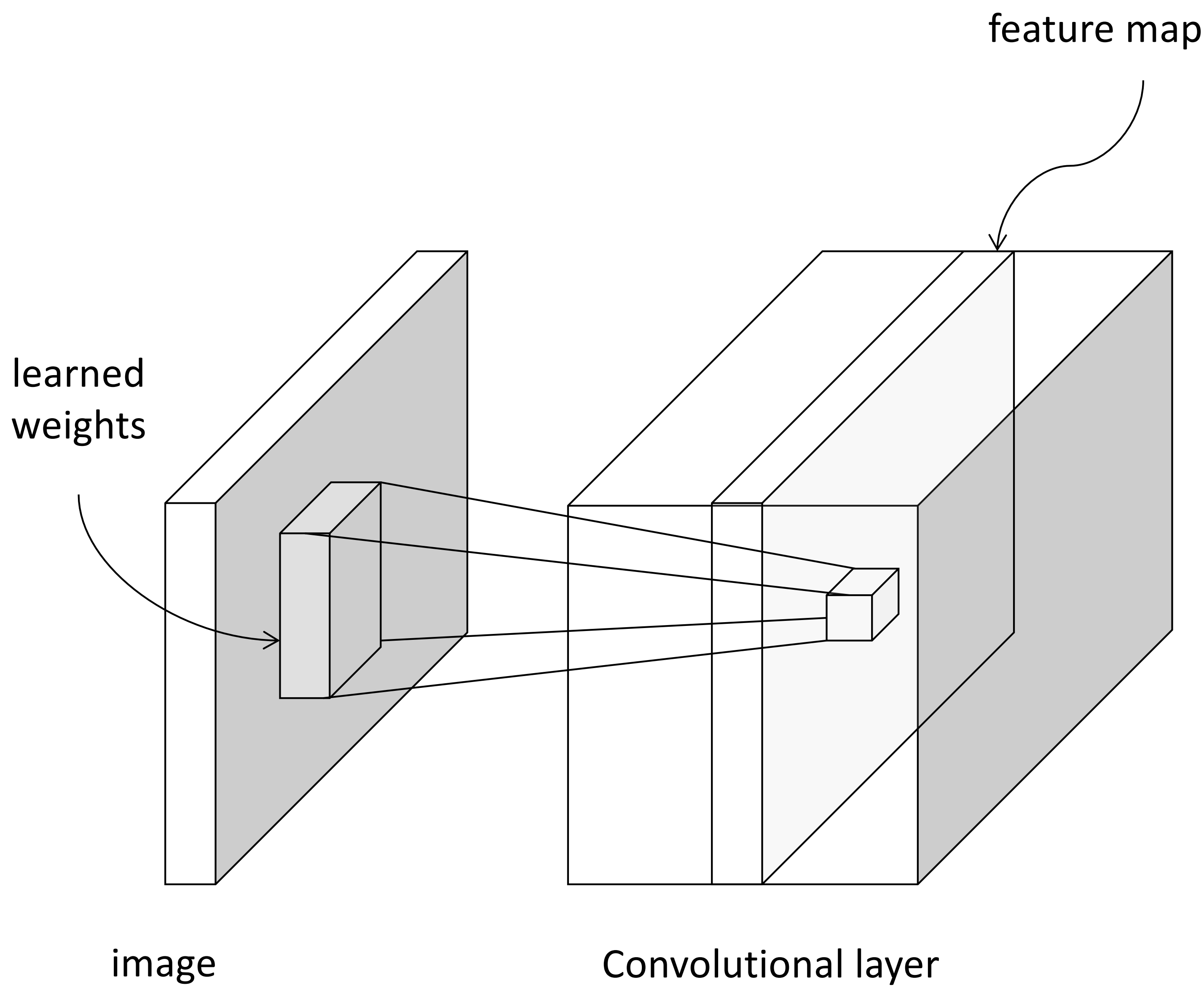


Input

Feature Map

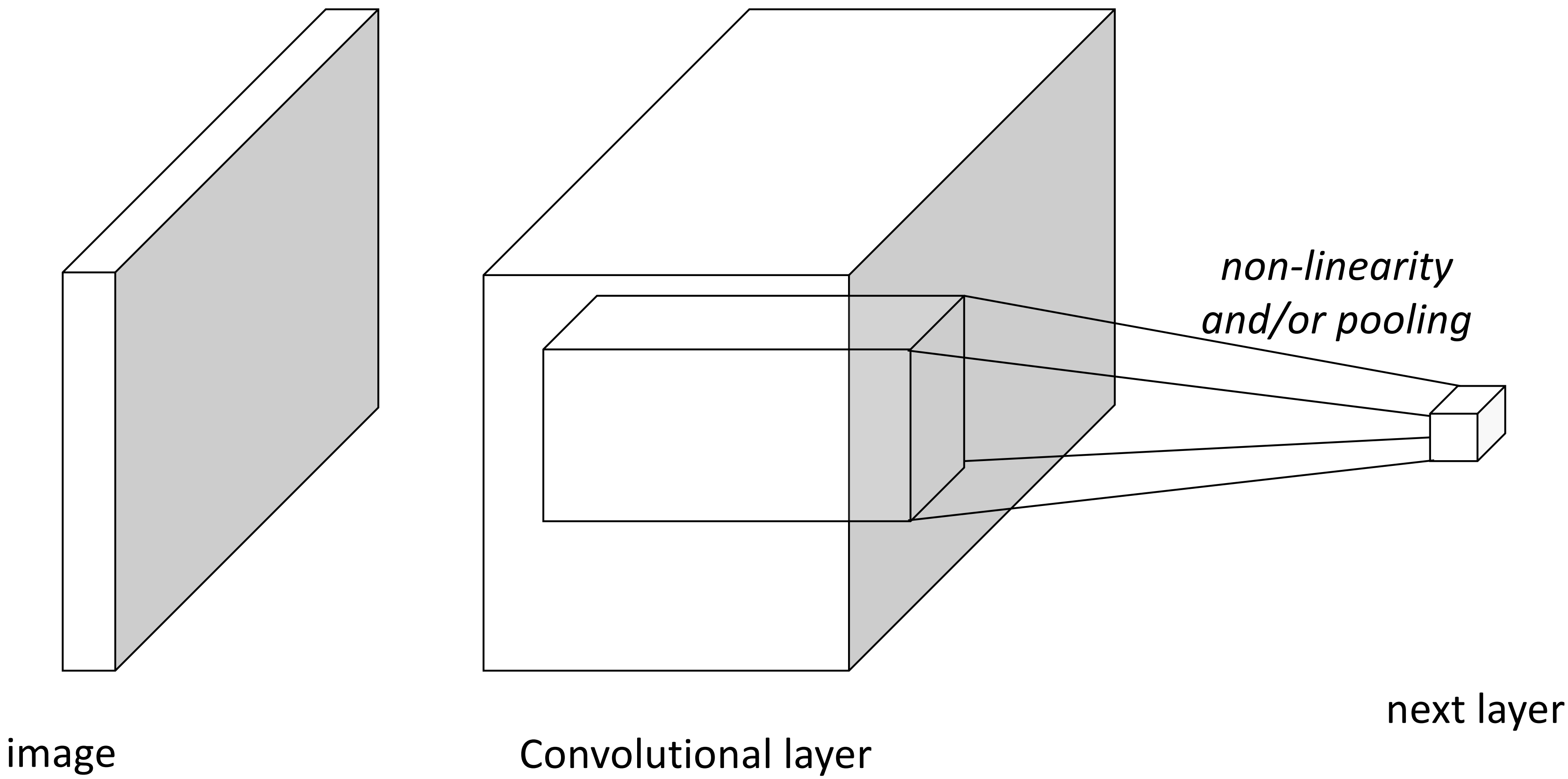
Slide credit: Svetlana Lazebnik

From fully connected to convolutional networks



Slide credit: Svetlana Lazebnik

From fully connected to convolutional networks



Outline

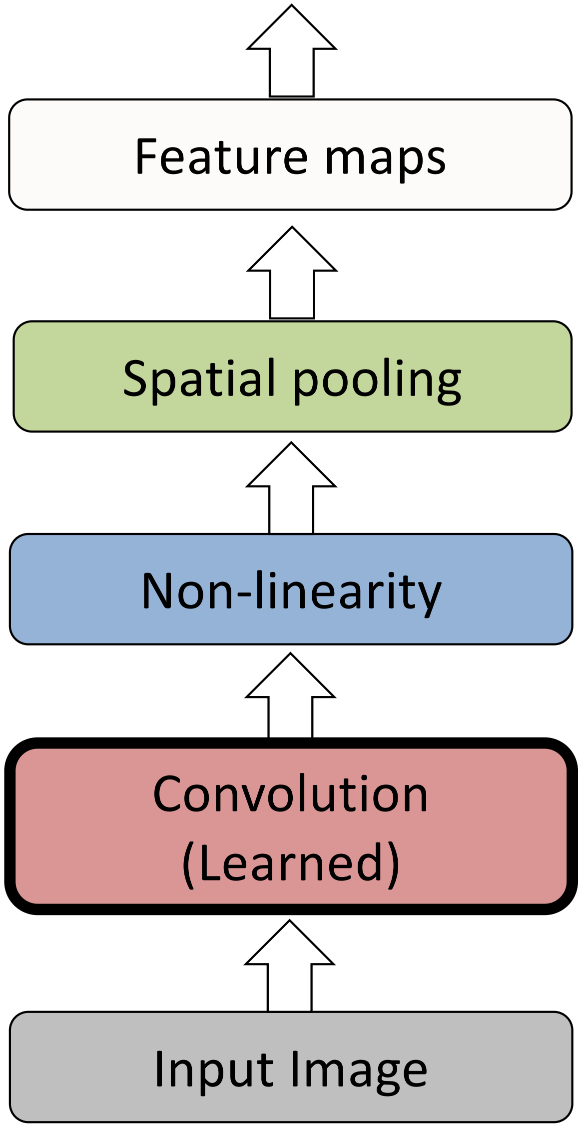
Convolutional Neural Networks

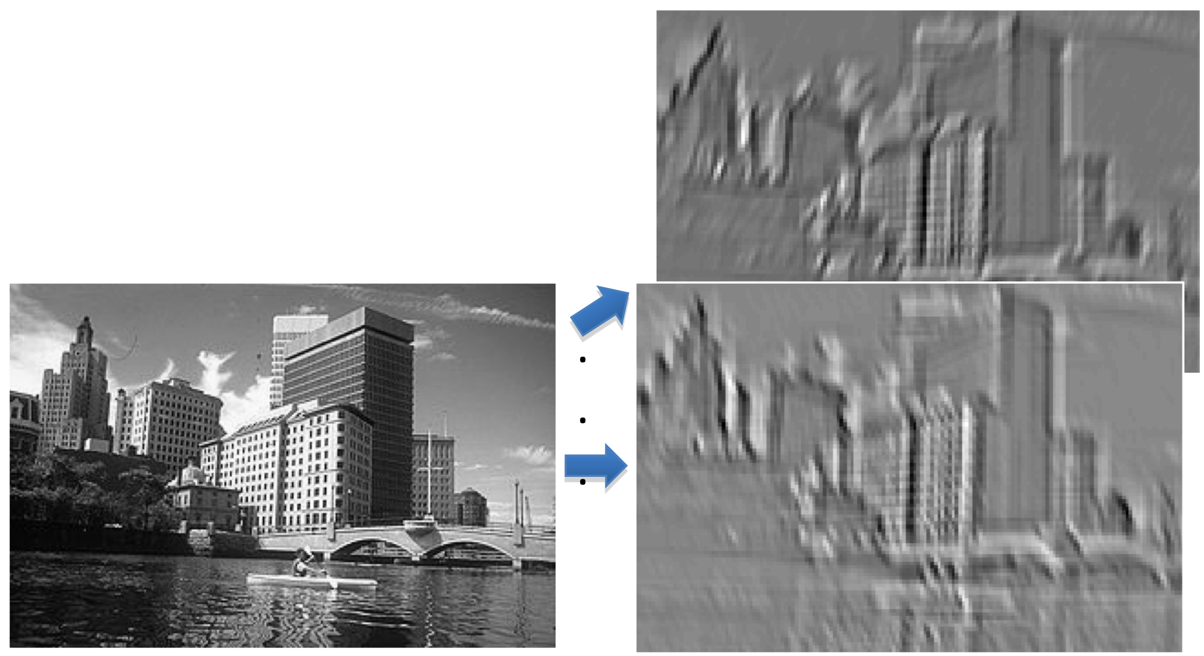
- What *is* a convolution?
- Multidimensional Convolutions
- Typical Convnet Operations
- Deep convnets

Recurrent Neural Networks

- Types of recurrence
- A basic recurrent cell
- BPTT: Backpropagation through time
- Solving vanishing gradients problem

Key operations in a CNN



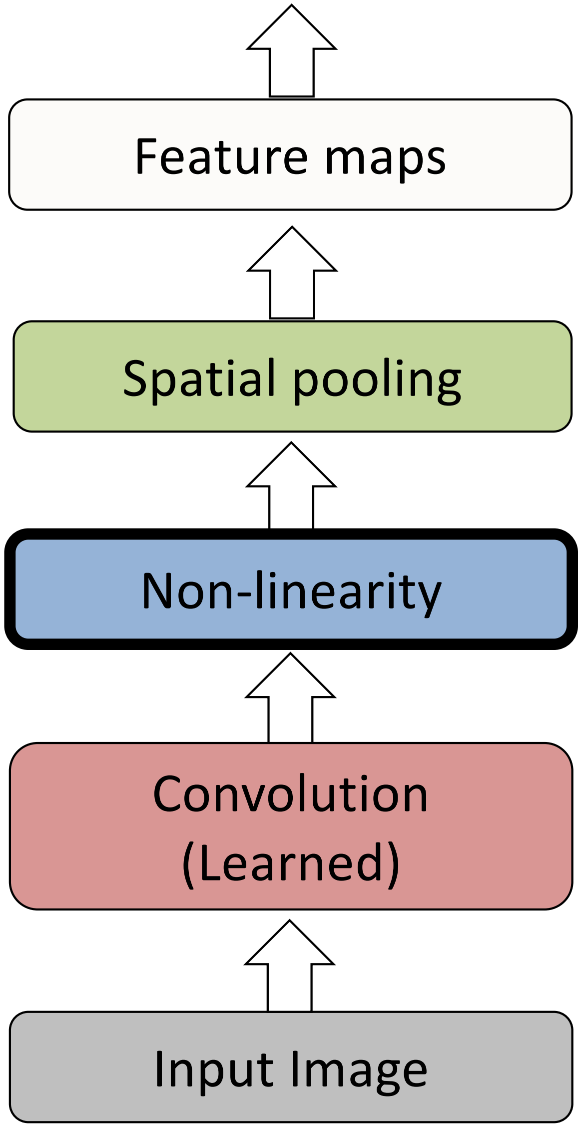


Input

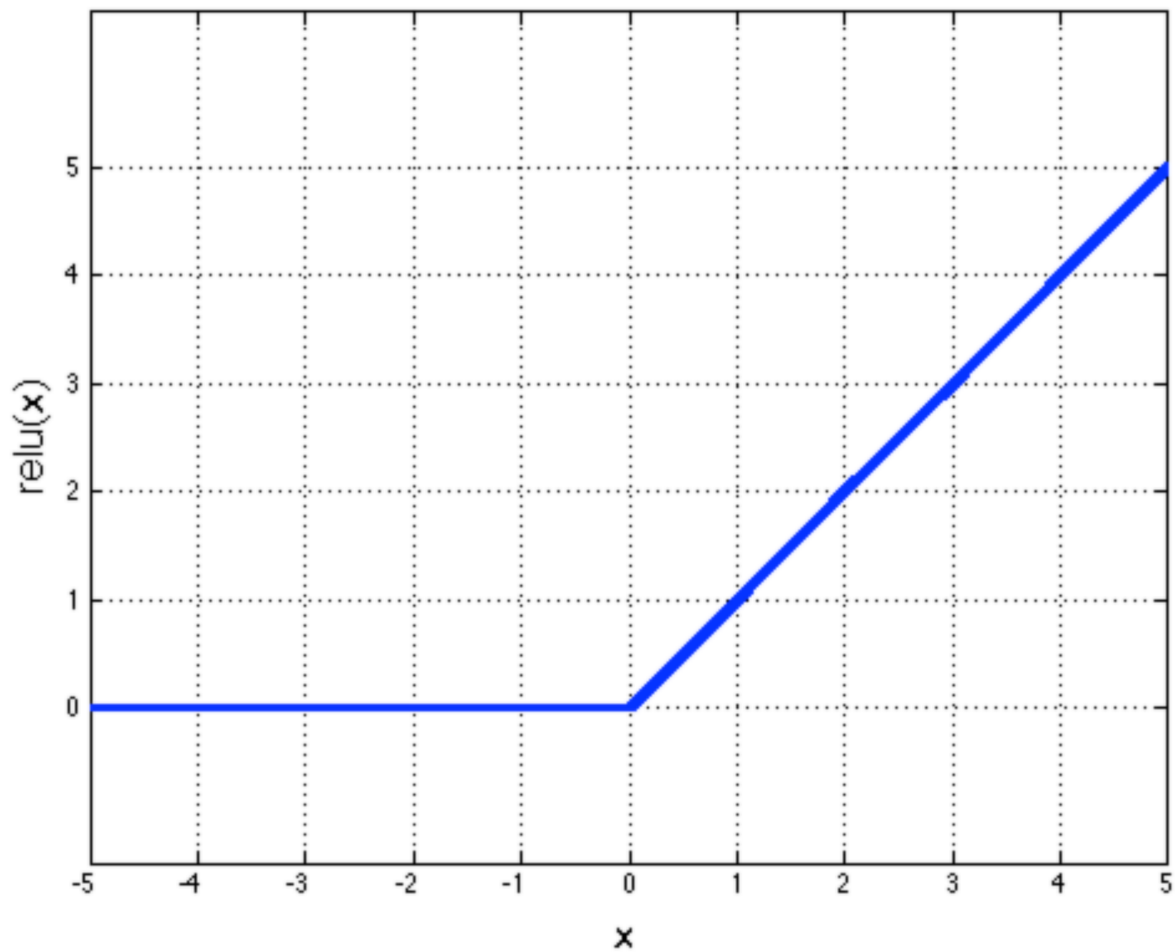
Feature Map

Slide credit: Svetlana Lazebnik, R. Fergus, Y. LeCun

Key operations

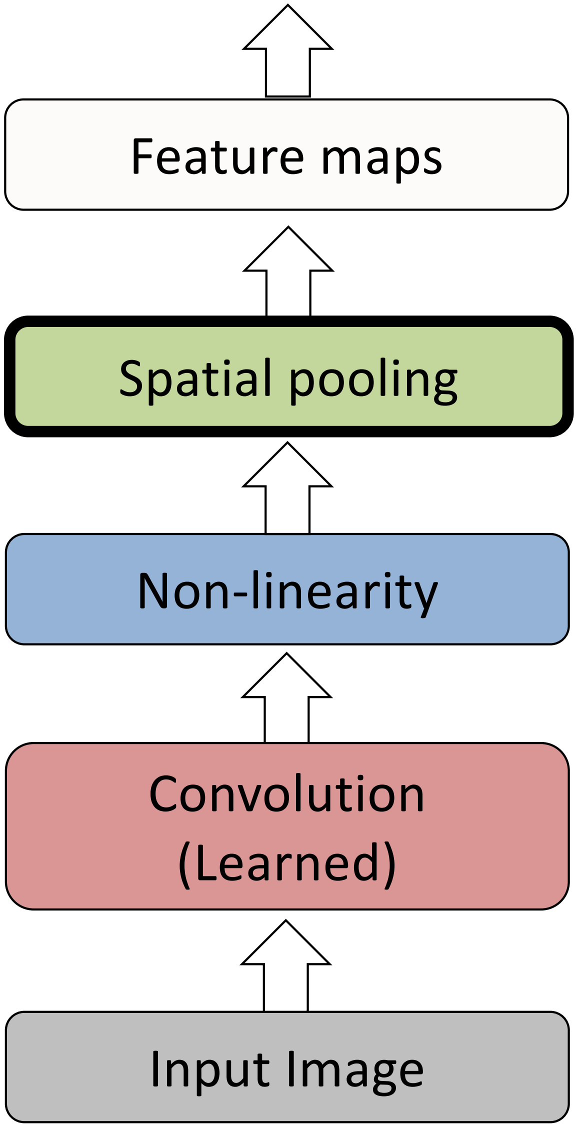


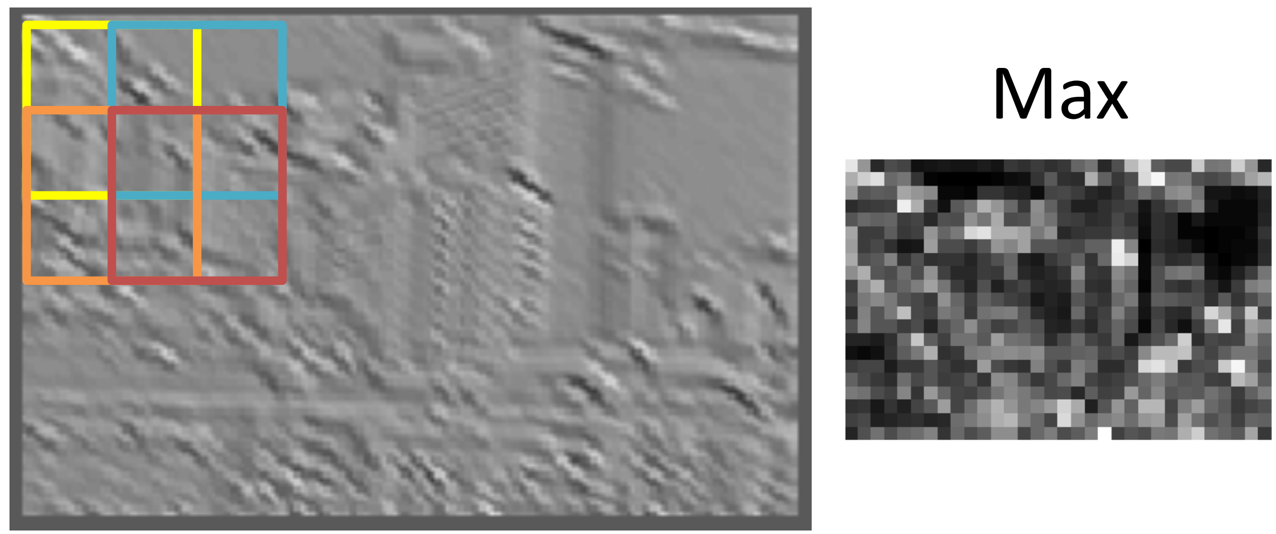
Example: Rectified Linear Unit (ReLU)



Slide credit: Svetlana Lazebnik, R. Fergus, Y. LeCun

Key operations





Design principles

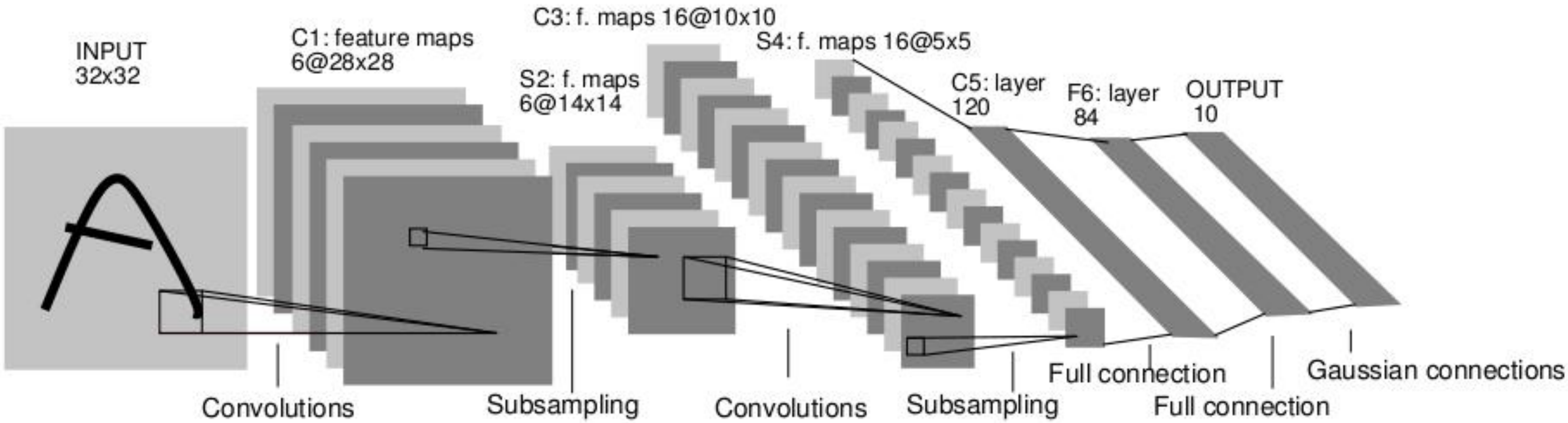
Reduce filter sizes (except possibly at the lowest layer), factorize filters aggressively

Use 1x1 convolutions to reduce and expand the number of feature maps judiciously

Use skip connections and/or create multiple paths through the network

Slide credit: Svetlana Lazebnik

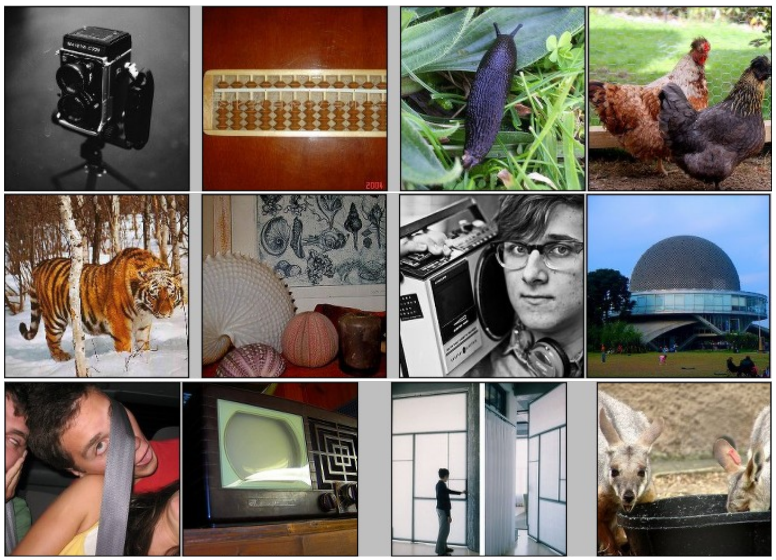
LeNet-5



Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86(11): 2278–2324, 1998.

Slide credit: Svetlana Lazebnik

ImageNet



~14 million labeled images, 20k classes

Images gathered from Internet

Human labels via Amazon MTurk

ImageNet Large-Scale Visual Recognition Challenge (ILSVRC): 1.2 million training images, 1000 classes

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

Slide credit: Svetlana Lazebnik

Slide credit: Svetlana Lazebnik

http://www.inference.vc/deep-learning-is-easy/

Outline

Convolutional Neural Networks

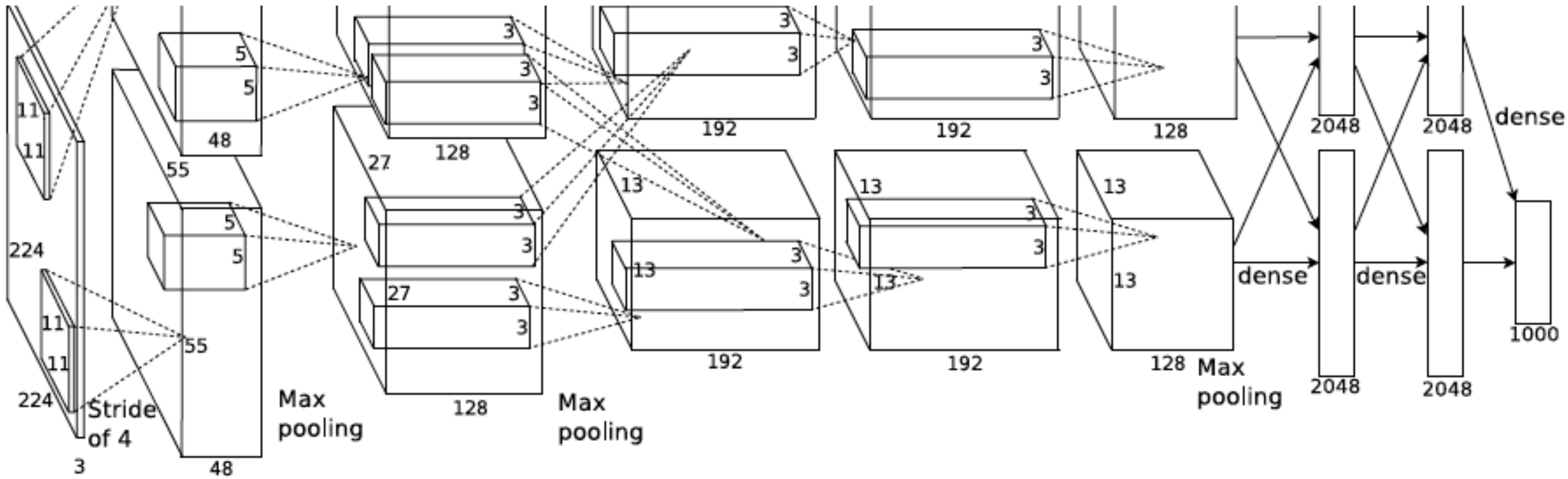
- What *is* a convolution?
- Multidimensional Convolutions
- Typical Convnet Operations
- Deep convnets

Recurrent Neural Networks

- Types of recurrence
- A basic recurrent cell
- BPTT: Backpropagation through time
- Solving vanishing gradients problem

Just FYI

AlexNet: ILSVRC 2012 winner



Similar framework to LeNet but:

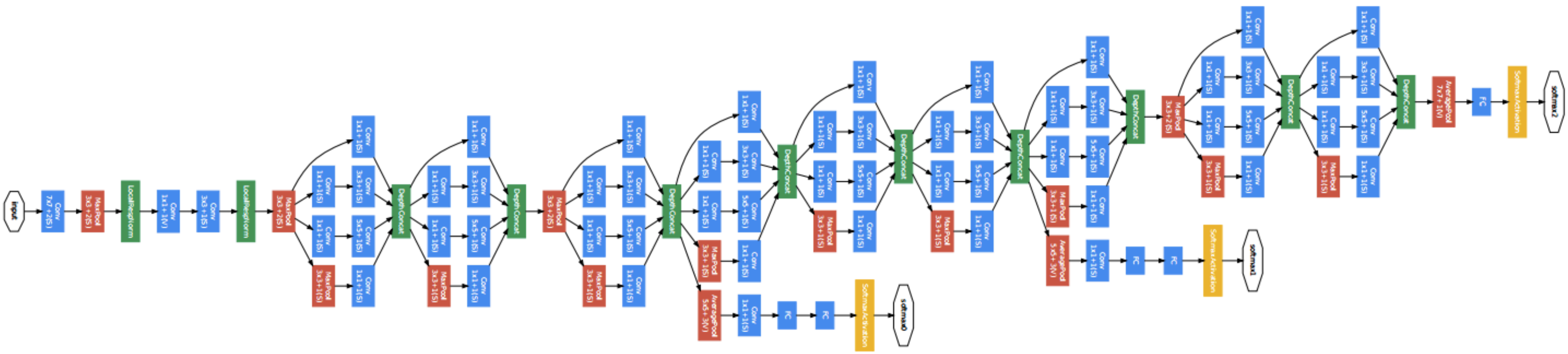
- Max pooling, ReLU nonlinearity
- More data and bigger model (7 hidden layers, 650K units, 60M params)
- GPU implementation (50x speedup over CPU): Two GPUs for a week
- Dropout regularization

A. Krizhevsky, I. Sutskever, and G. Hinton, [ImageNet Classification with Deep Convolutional Neural Networks], NIPS 2012

Slide credit: Svetlana Lazebnik

Just FYI

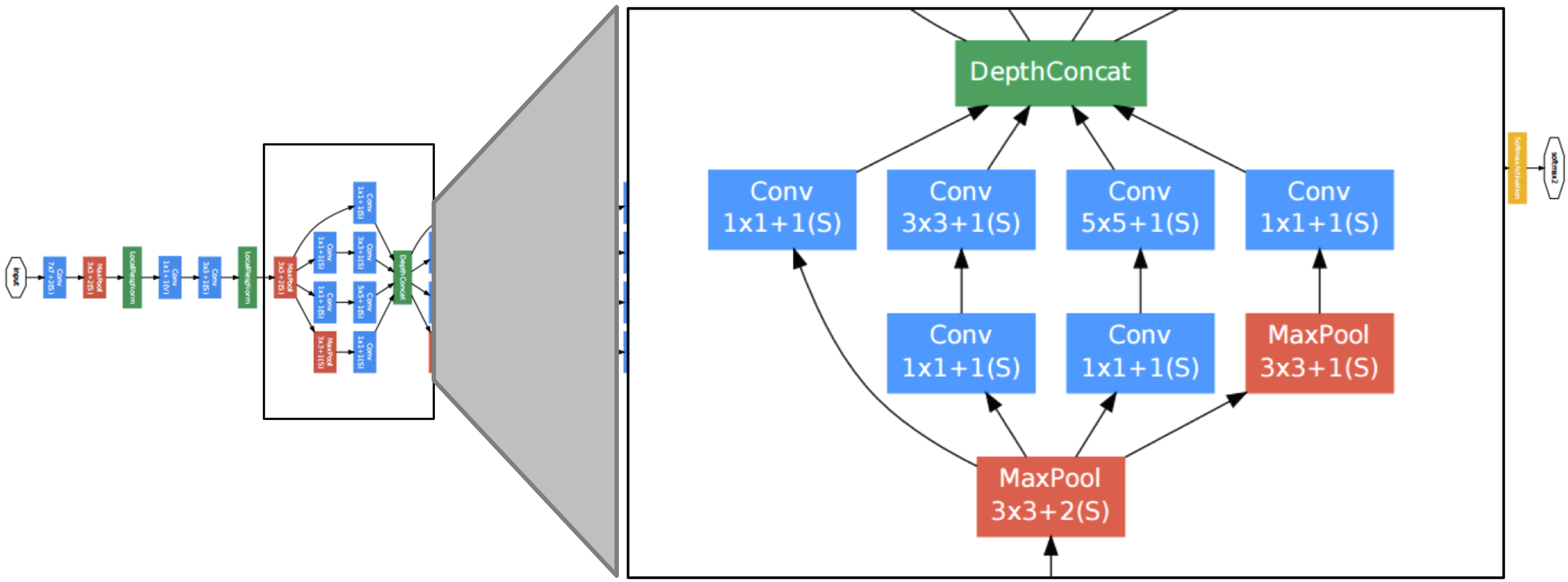
GoogLeNet



Szegedy et al., 2015

Just FYI

GoogLeNet

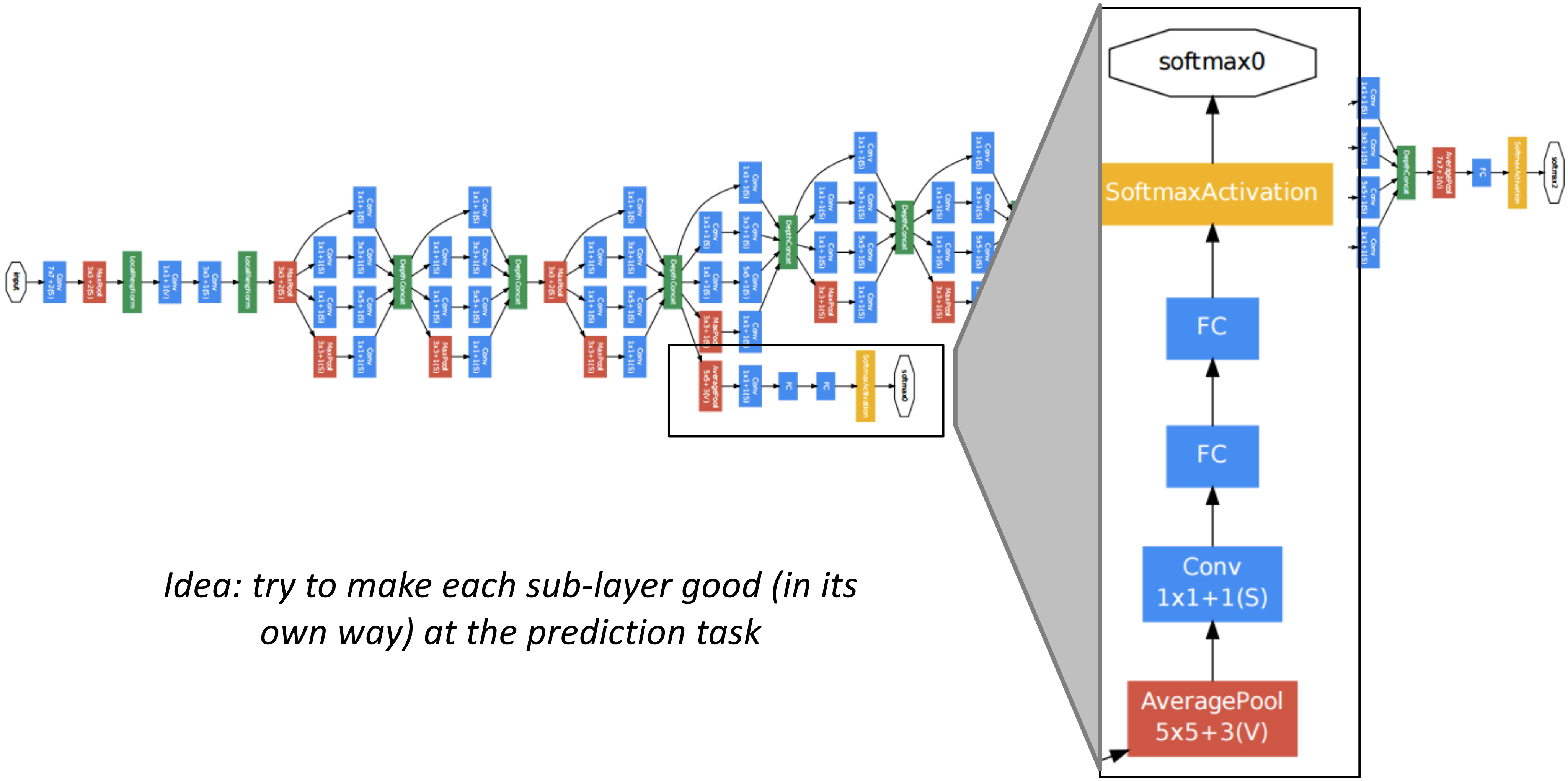


Slide credit: Svetlana Lazebnik

Szegedy et al., 2015

Just FYI

GoogLeNet: Auxiliary Classifier at Sub-levels



Idea: try to make each sub-layer good (in its own way) at the prediction task

Szegedy et al., 2015

Slide credit: Svetlana Lazebnik

Just FYI

GoogLeNet

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

Slide credit: Svetlana Lazebnik

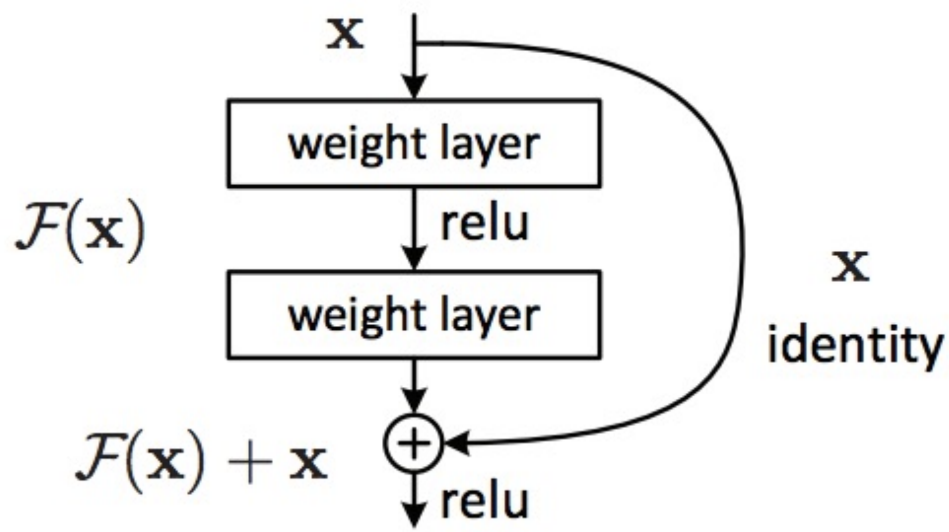
Szegedy et al., 2015

Just FYI

ResNet (Residual Network)

Make it easy for network layers to represent the identity mapping

Skipping 2+ layers is intentional & needed



He et al. "Deep Residual Learning for Image Recognition" (2016)

Slide credit: Svetlana Lazebnik

Just FYI

Summary: ILSVRC 2012-2015

Team	Year	Place	Error (top-5)	External data
SuperVision	2012	-	16.4%	no
SuperVision	2012	1st	15.3%	ImageNet 22k
Clarifai (7 layers)	2013	-	11.7%	no
Clarifai	2013	1st	11.2%	ImageNet 22k
VGG (16 layers)	2014	2nd	7.32%	no
GoogLeNet (19 layers)	2014	1st	6.67%	no
ResNet (152 layers)	2015	1st	3.57%	
Human expert*			5.1%	

http://karpathy.github.io/2014/09/02/what-i-learned-from-competing-against-a-convnet-on-imagenet/

Slide credit: Svetlana Lazebnik

Rapid Progress due to CNNs

Classification: ImageNet Challenge top-5 error

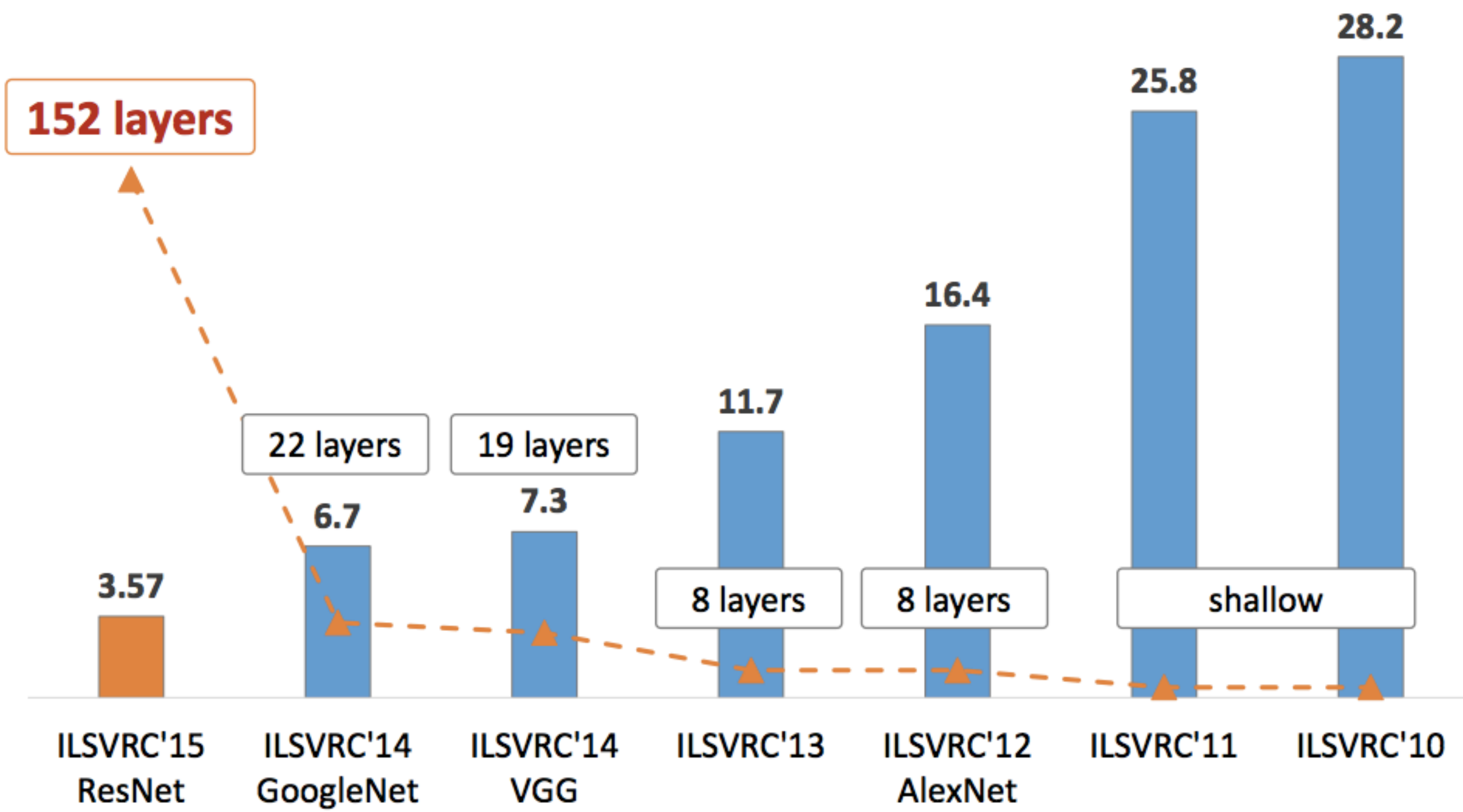


Figure source: [Kaiming He]

Slide credit: Svetlana Lazebnik

Slide Credit

http://slazebni.cs.illinois.edu/spring17/lec01_cnn_architectures.pdf

http://slazebni.cs.illinois.edu/spring17/lec02_rnn.pdf