Convolutional Neural Networks

Machine Learning

(Largely based on slides from Fei-Fei Li & Justin Johnson & Serena Yeung)

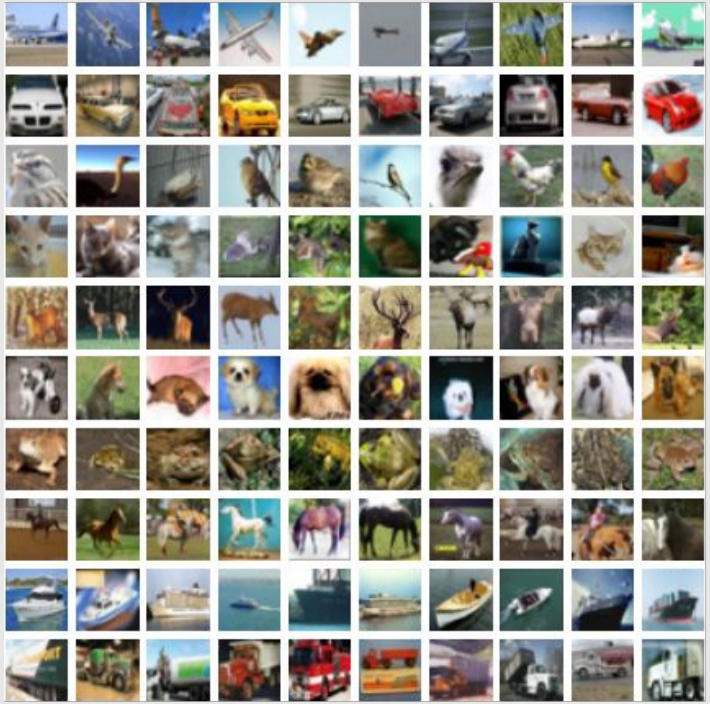
Prof. Sandra Avila
Institute of Computing (IC/Unicamp)

MC886, October 14, 2019

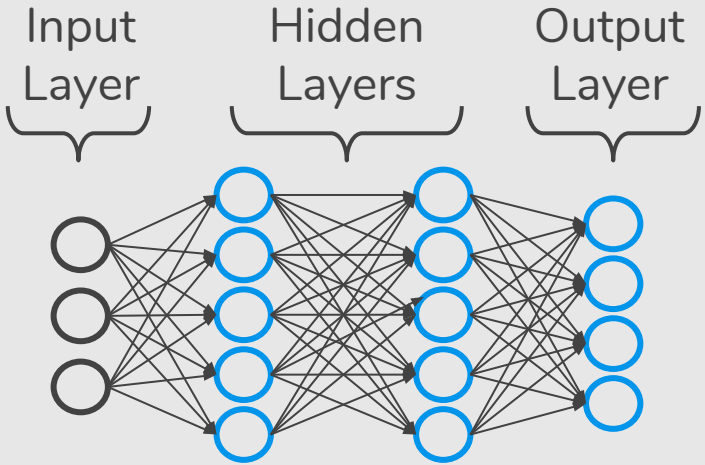
Today's Agenda

- **Neural Networks vs. Convolutional Networks**
- **What is a convolution?**
- Convolutional Neural Networks
  - Convolution Layer
  - Pooling Layer
  - Fully-connected Layer

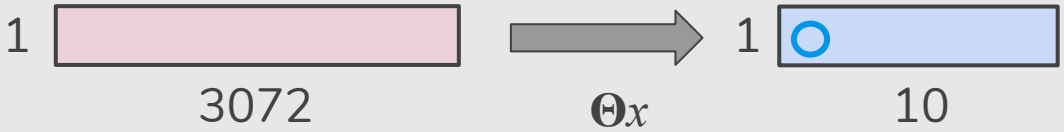
Neural Networks



CIFAR-10



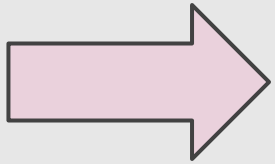
32 × 32 × 3 image ⇒ stretch to 3072 × 1



Neural Networks

Neural Networks

0	0	3	2
1	1	0	1
4	2	1	2
0	2	1	5



0
0
3
2
1
1
0
1
⋮
2
1
5

What is a Convolution?

Convolution is the process of adding each element of the image to its local neighbors, **weighted by the kernel**.

What is a Convolution?

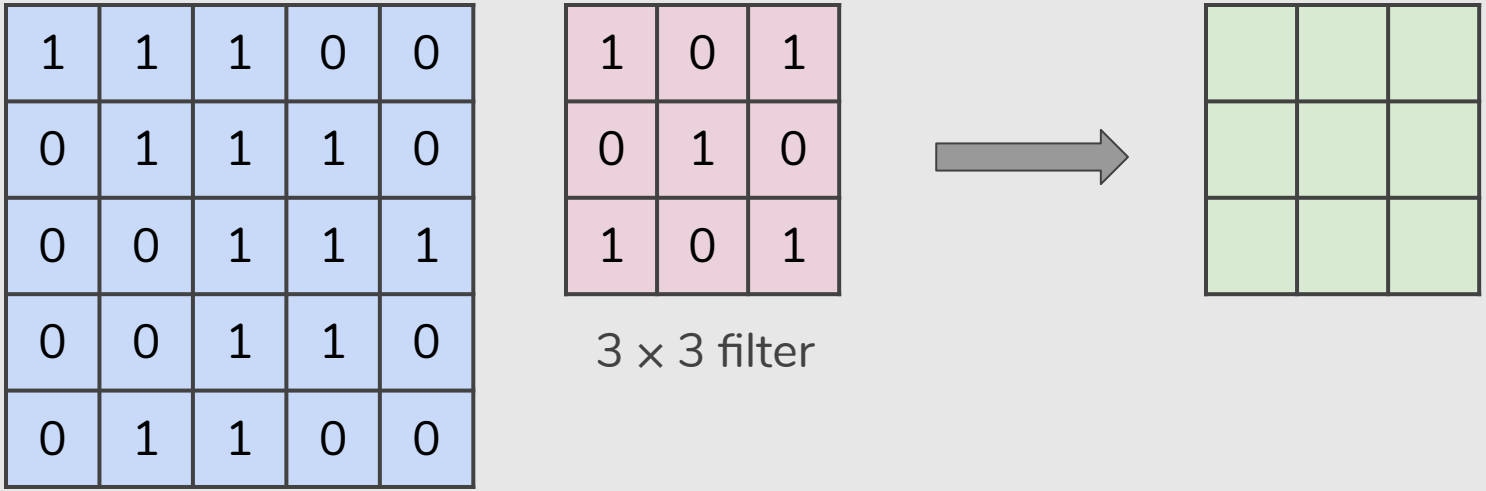
1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

5 x 5 matrix
(image)

1	0	1
0	1	0
1	0	1

3 x 3 filter

What is a Convolution?



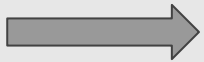
What is a Convolution?

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

5 x 5 matrix
(image)

1	0	1
0	1	0
1	0	1

3 x 3 filter



4		

1*1 + 1*0 + 1*1 +
0*0 + 1*1 + 1*0 +
0*1 + 0*0 + 1*1 = 4

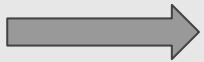
What is a Convolution?

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

5 x 5 matrix
(image)

1	0	1
0	1	0
1	0	1

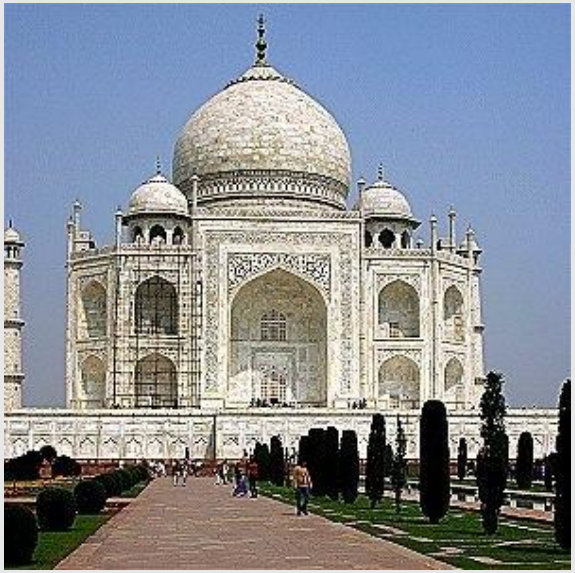
3 x 3 filter



4	3	4
2	4	3
2	3	4

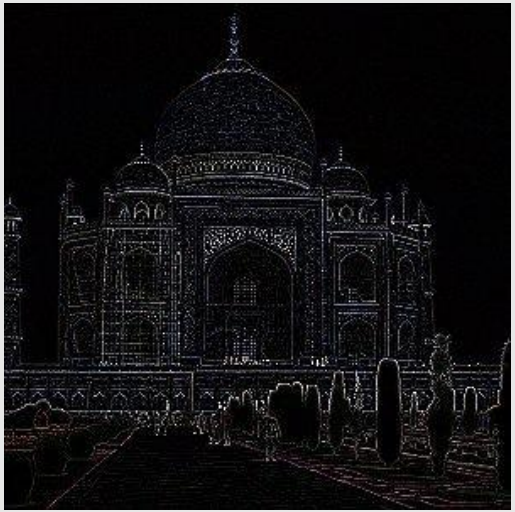
1*1 + 1*0 + 1*1 +
1*0 + 1*1 + 0*0 +
1*1 + 0*0 + 0*1 = 4

What is a Convolution?

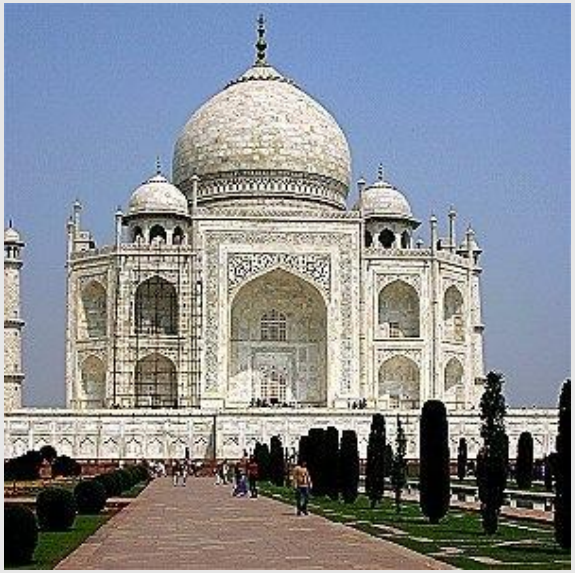


Edge Detection

0	1	0
1	-4	1
0	1	0



What is a Convolution?



Emboss

-2	-1	0
-1	1	1
0	1	2

What is a Convolution?

-1	-1	-1
-1	8	-1
-1	-1	-1

Edge Detection

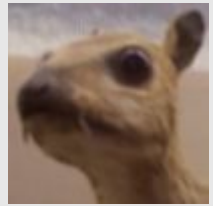
0	-1	0
-1	5	-1
0	-1	0

Sharpen

1/9

1	1	1
1	1	1
1	1	1

Box blur



1/16

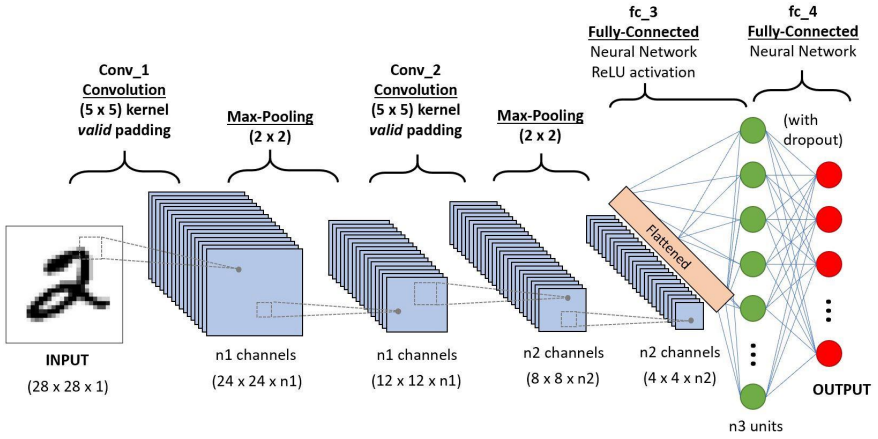
1	2	1
2	4	2
1	2	1

Gaussian blur 3 × 3

Today's Agenda

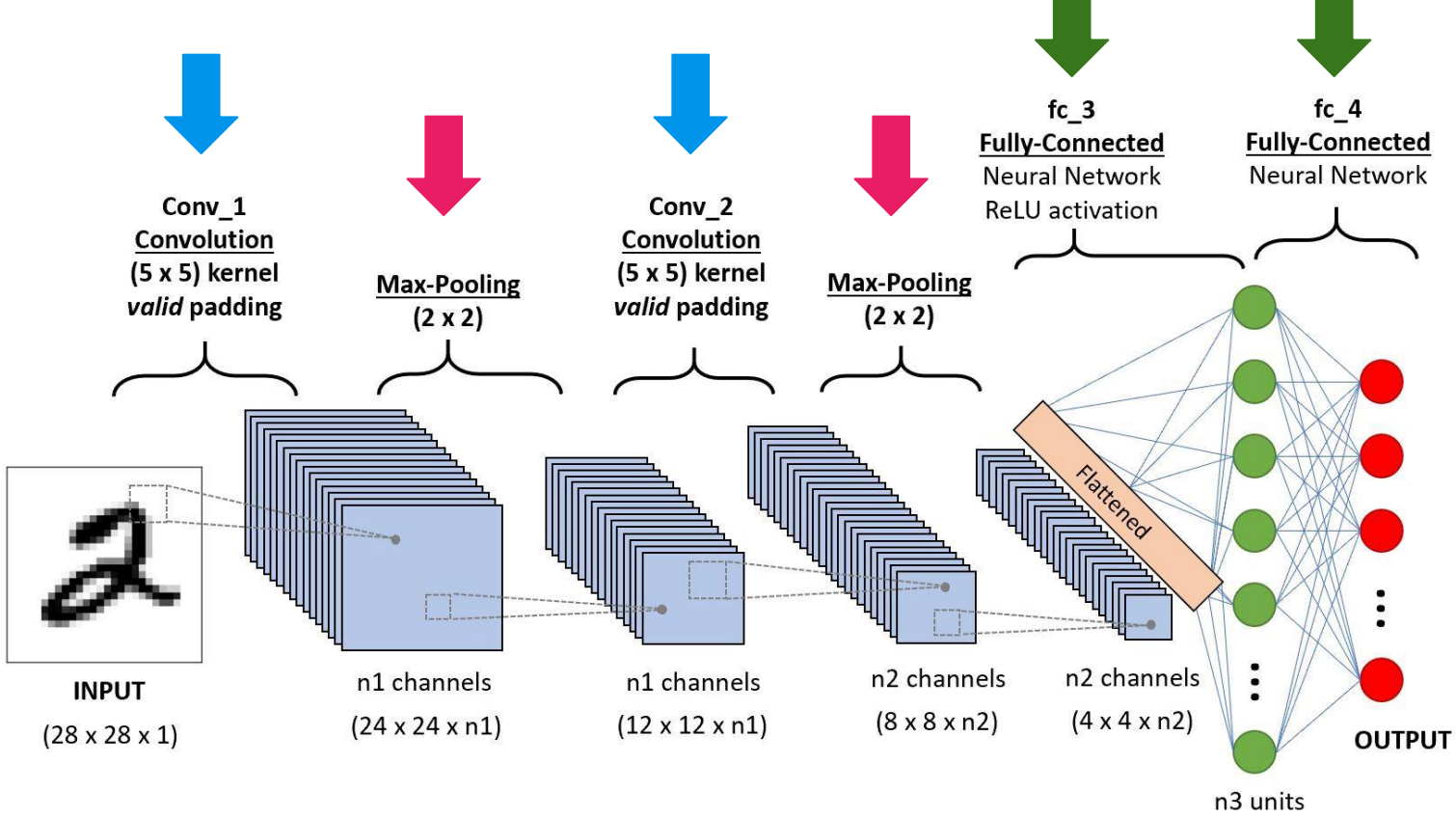
- Neural Networks *vs.* Convolutional Networks
- What is a convolution?
- **Convolutional Neural Networks**
  - Convolution Layer
  - Pooling Layer
  - Fully-connected Layer

Convolutional Neural Networks (CNNs)



"Gradient-based learning applied to document recognition", 1998 http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf

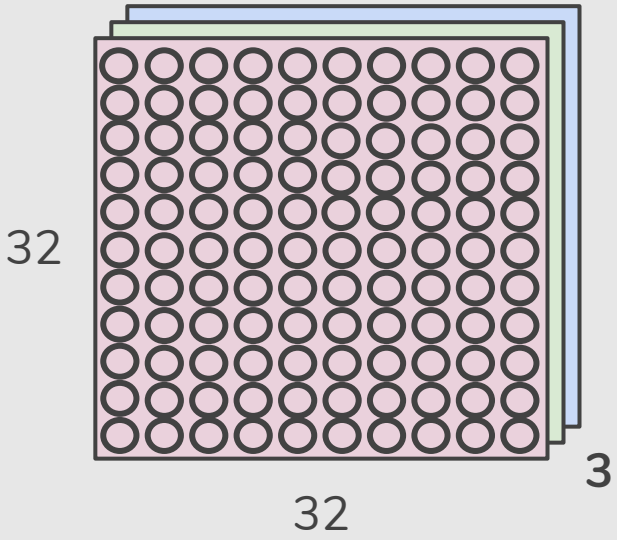
https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53



There are a few distinct types of layers (e.g., CONV/POOL/FC are by far the most popular).

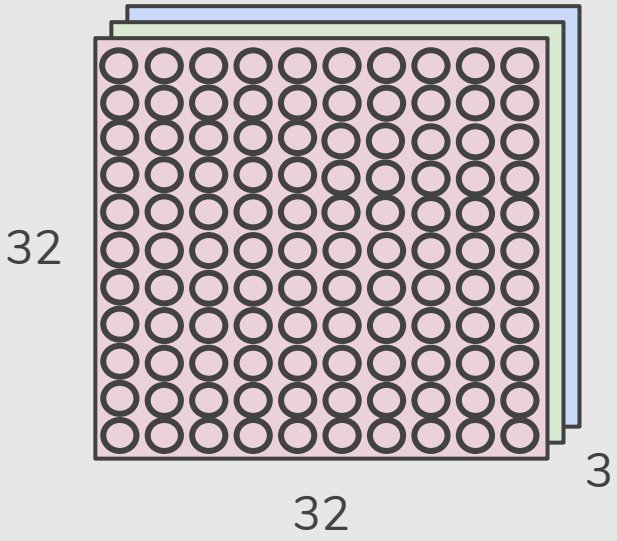
Convolution Layer

32 × 32 × 3 image ⇒ preserve spatial structure



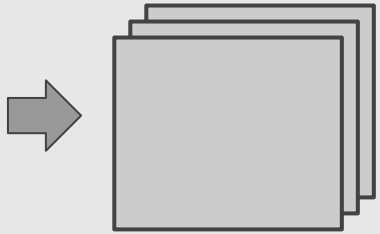
Convolution Layer

32 x 32 x 3 image ⇒ preserve spatial structure



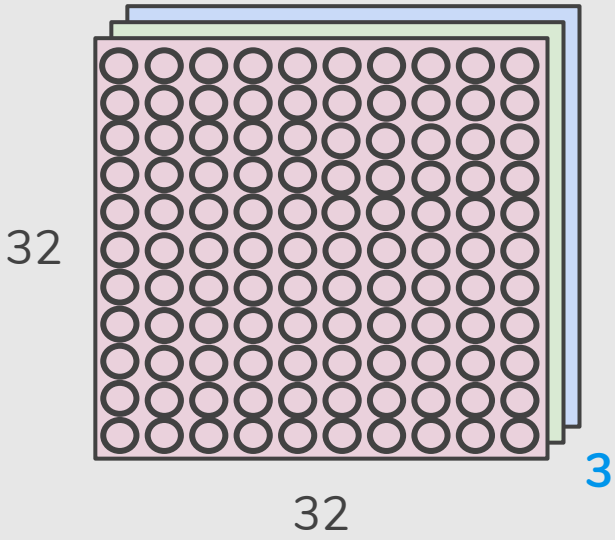
Convolve the filter with the image i.e. “slide over the image spatially, computing dot products”

5 x 5 x 3 filter

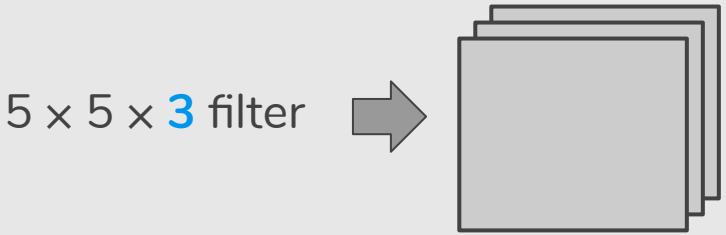


Convolution Layer

32 x 32 x 3 image ⇒ preserve spatial structure



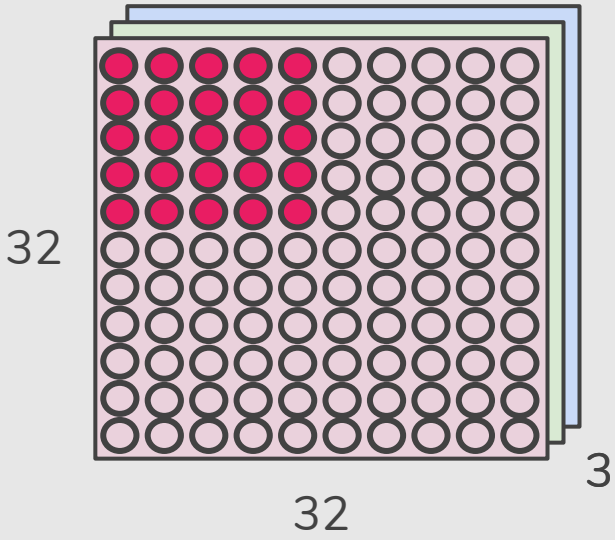
Convolve the filter with the image i.e. "slide over the image spatially, computing dot products"



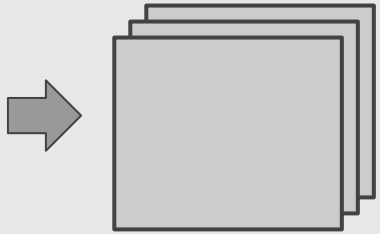
Filters always extend the full depth of the input volume

Convolution Layer

32 x 32 x 3 image ⇒ preserve spatial structure

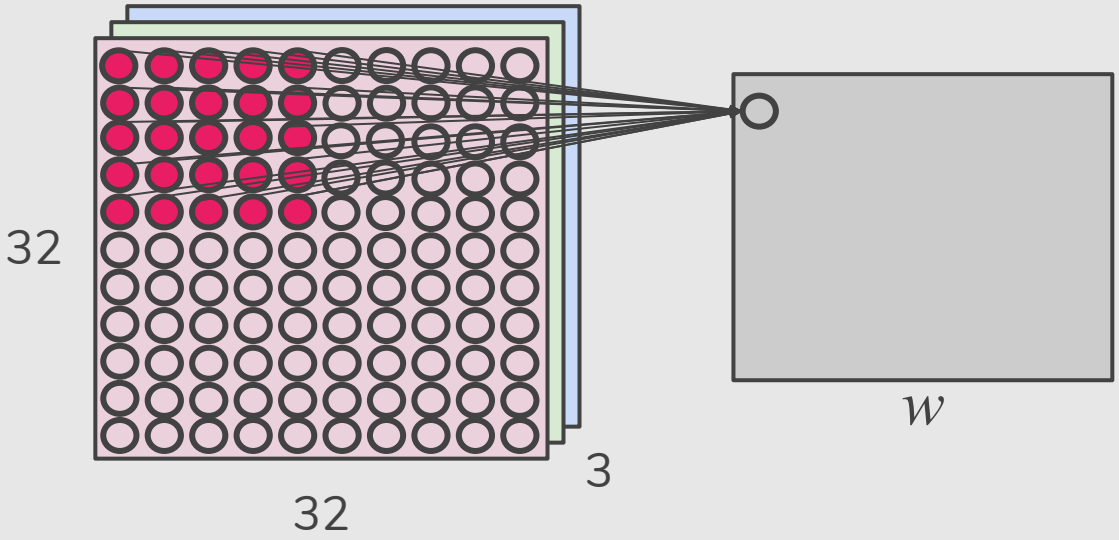


5 x 5 x 3 filter



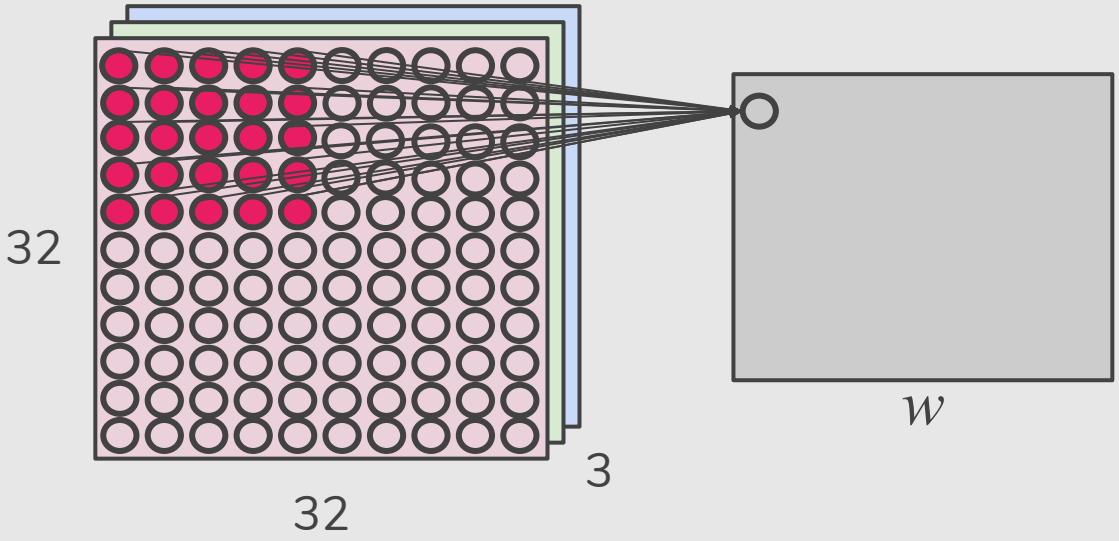
Convolution Layer

32 × 32 × 3 image ⇒ preserve spatial structure



Convolution Layer

32 x 32 x 3 image ⇒ preserve spatial structure



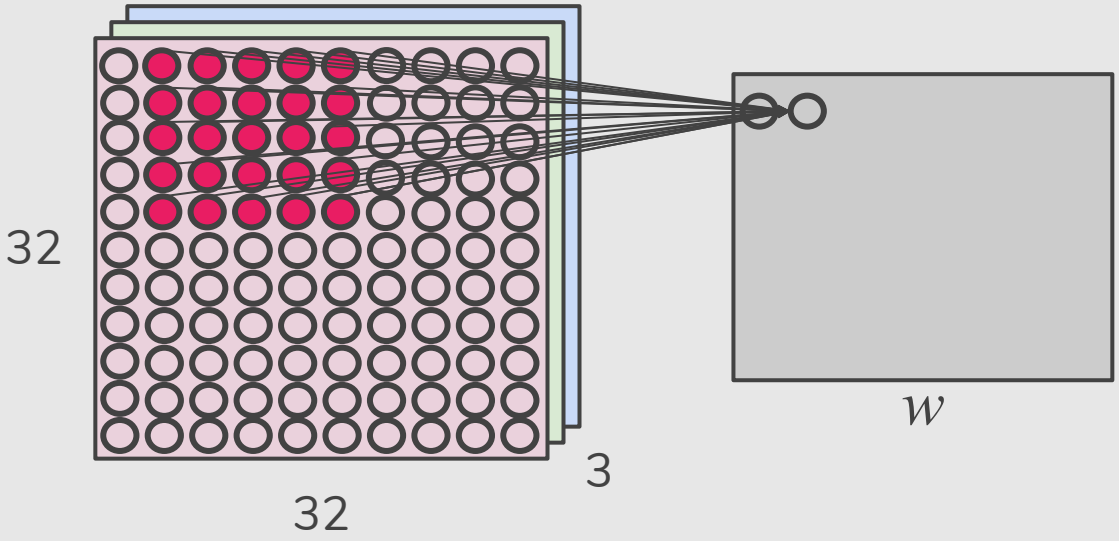
1 number:

5*5*3 = 75-dimensional dot product + bias)

$w^T x + b$

Convolution Layer

32 x 32 x 3 image ⇒ preserve spatial structure



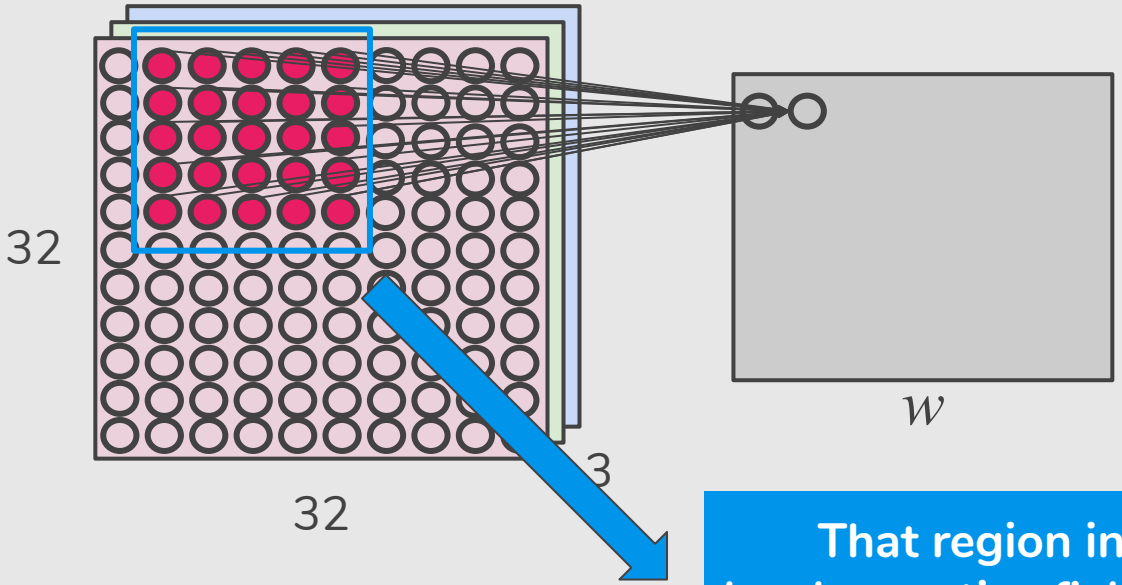
1 number:

5*5*3 = 75-dimensional dot product + bias)

$w^T x + b$

Convolution Layer

32 x 32 x 3 image ⇒ preserve spatial structure



1 number:

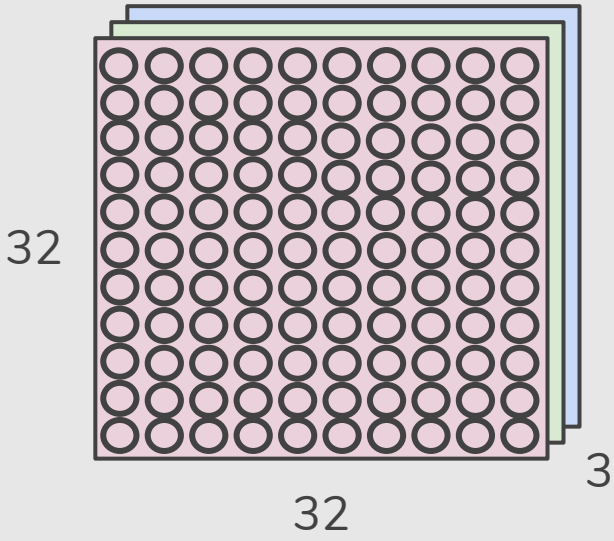
5*5*3 = 75-dimensional dot product + bias)

$w^T x + b$

That region in the input image is called the *local receptive field* for the hidden neuron.

Convolution Layer

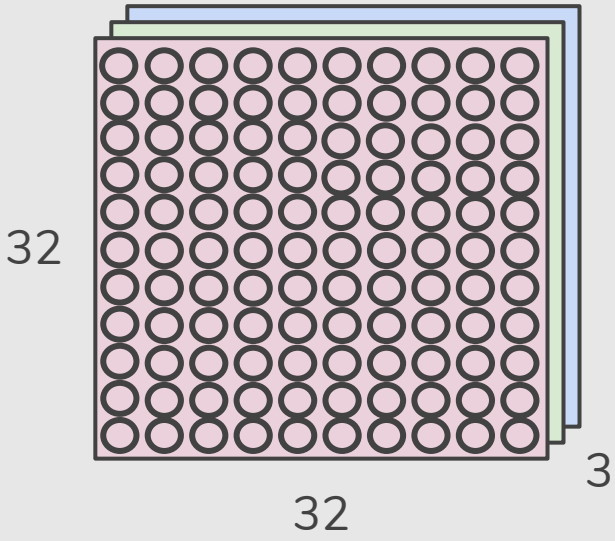
32 × 32 × 3 image ⇒ preserve spatial structure



Convolve (slide) over
all spatial locations

Convolution Layer

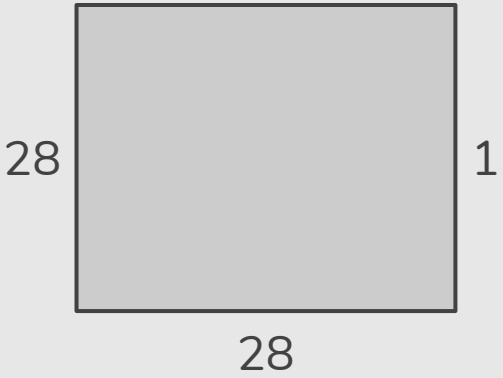
32 x 32 x 3 image ⇒ preserve spatial structure



Convolve (slide) over all spatial locations

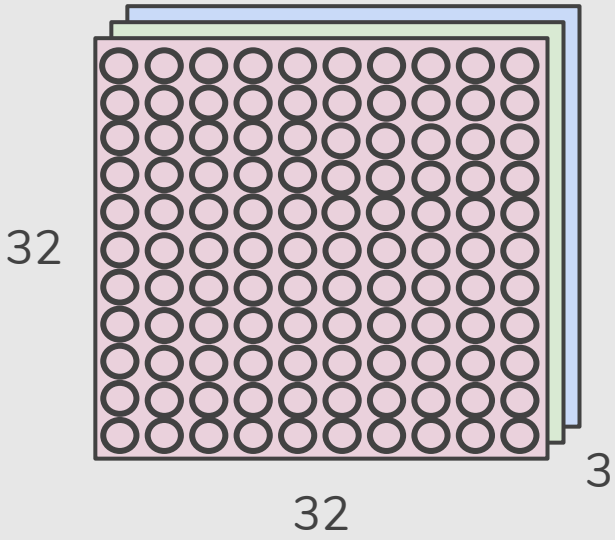
32 x 32 x 3 image
5 x 5 x 3 filter

activation map



Convolution Layer

32 x 32 x 3 image ⇒ preserve spatial structure

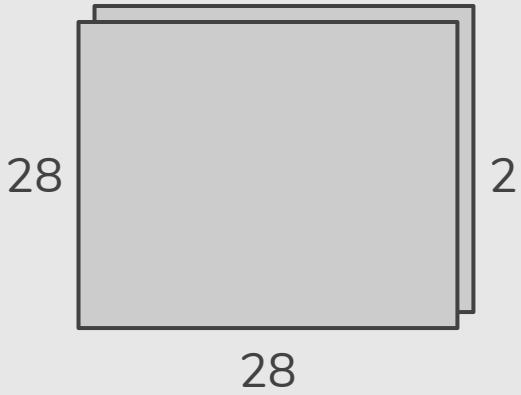


Convolve (slide) over all spatial locations

32 x 32 x 3 image
5 x 5 x 3 filter

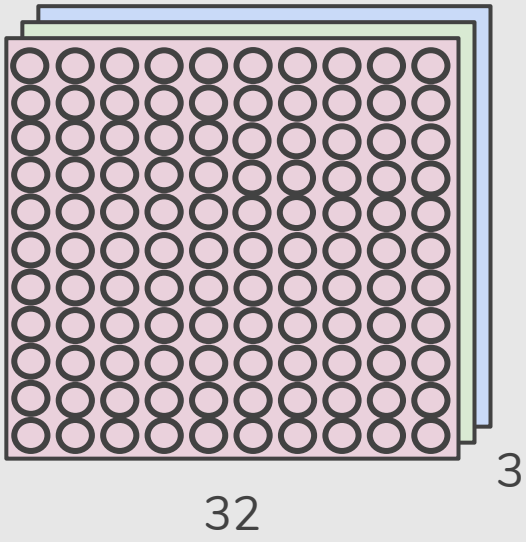
(considering a second filter)

activation maps



Convolution Layer

32 x 32 x 3 image ⇒ preserve spatial structure



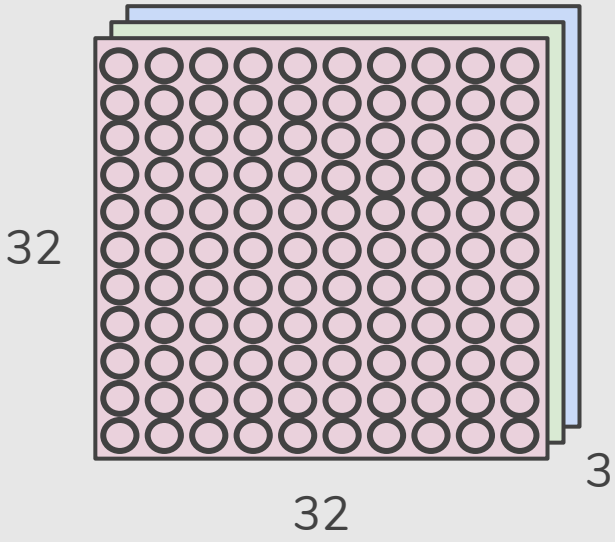
Convolve (slide) over all spatial locations

32 x 32 x 3 image
5 x 5 x 3 filter

If we had 6 5 x 5 x 3 filters ...

Convolution Layer

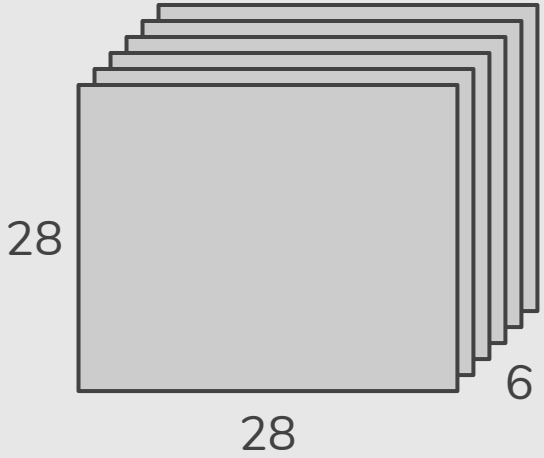
32 x 32 x 3 image ⇒ preserve spatial structure



Convolve (slide) over all spatial locations

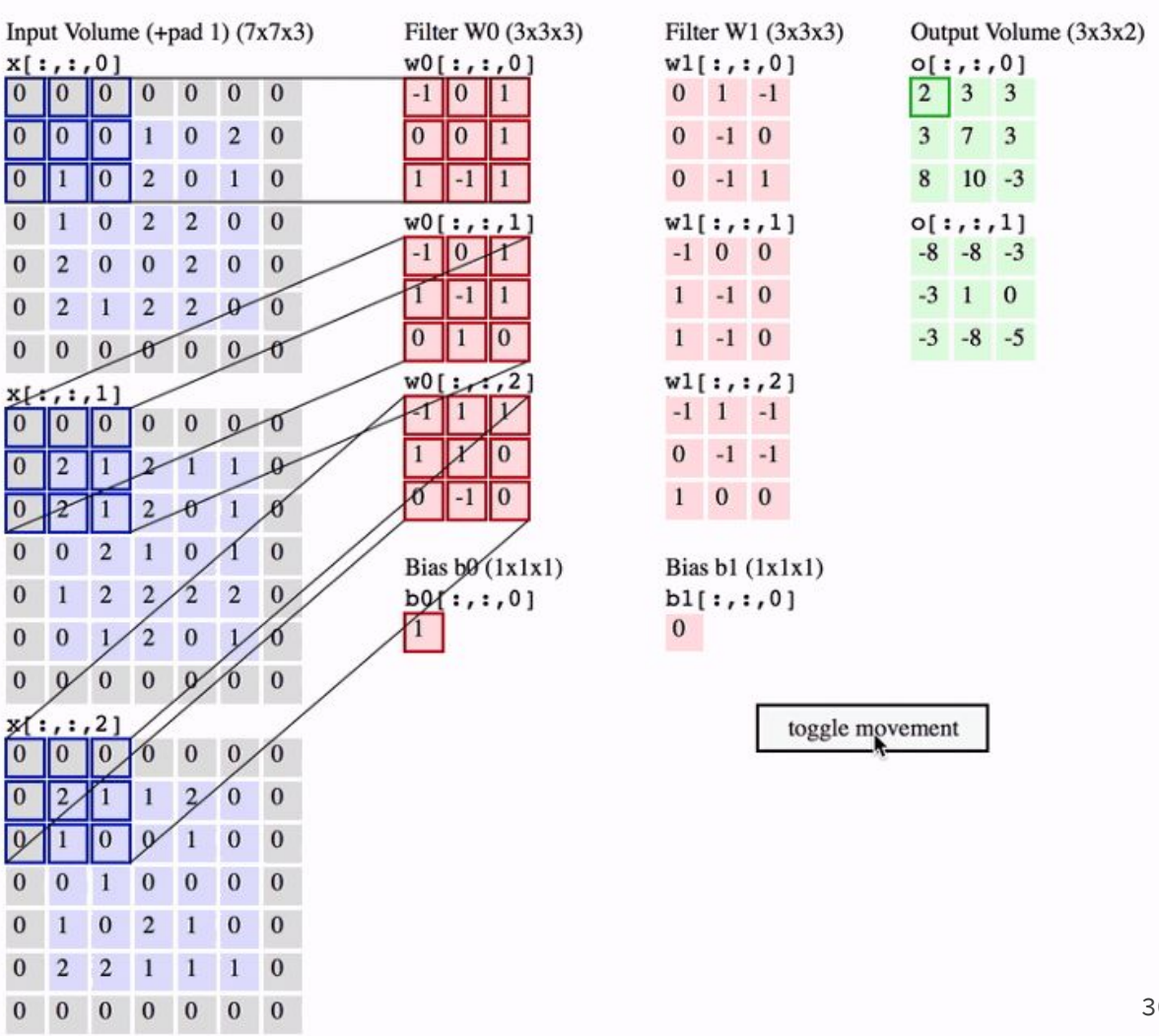
32 x 32 x 3 image
5 x 5 x 3 filter

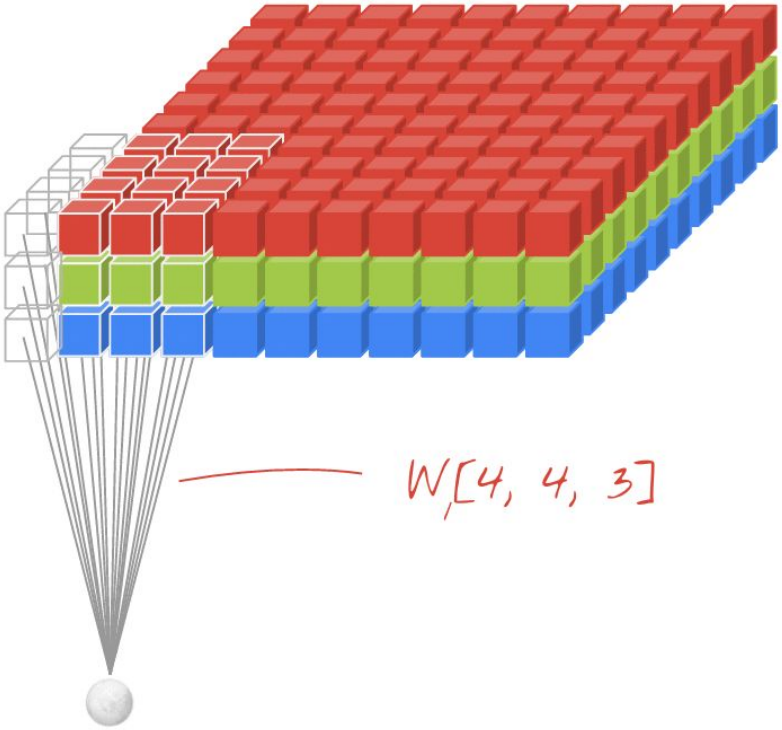
6 activation maps



If we had 6 5 x 5 x 3 filters ...

http://cs231n.github.io/convolutional-networks

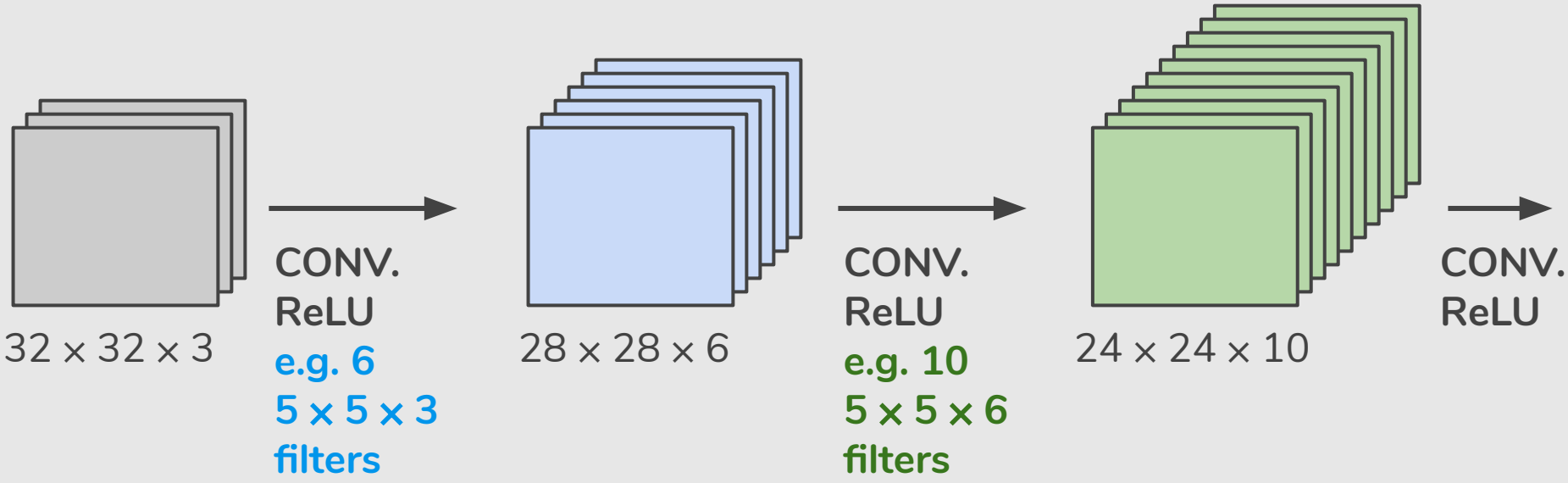




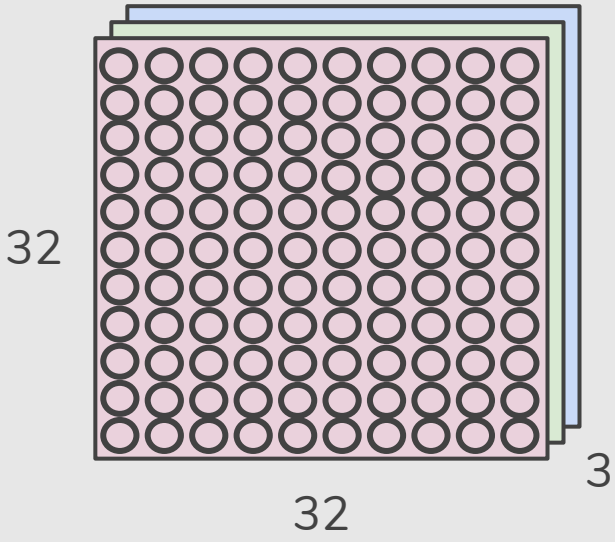
Credit: Martin Görner, @martin_gorner (twitter)

Convolutional Networks

Sequence of Convolutional Layers, interspersed with activation functions.



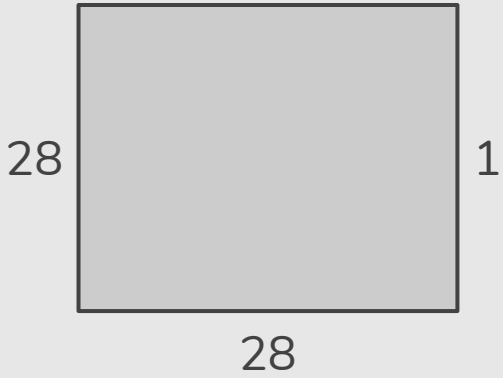
A Closer Look at Spatial Dimensions



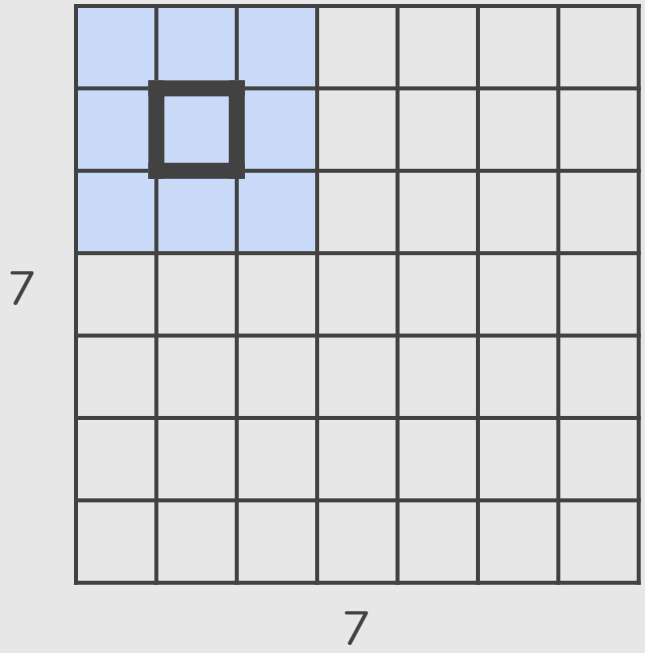
Convolve (slide) over
all spatial locations

32 x 32 x 3 image
5 x 5 x 3 filter

activation map

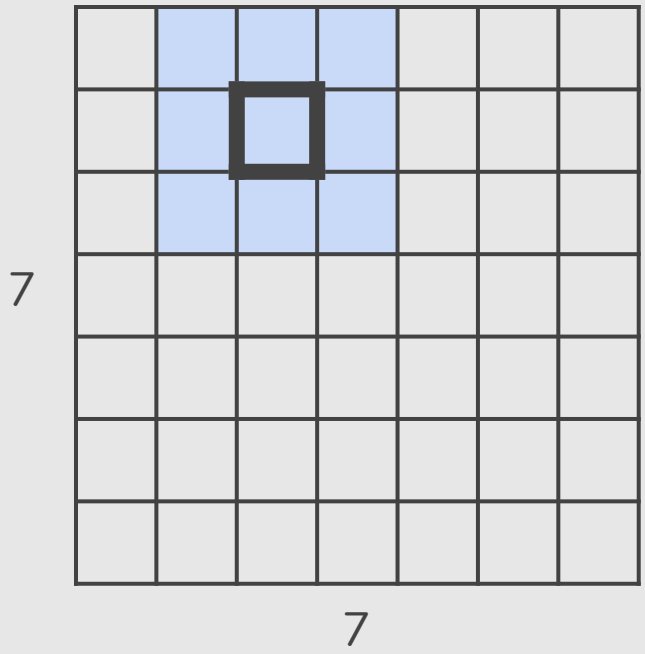


A Closer Look at Spatial Dimensions



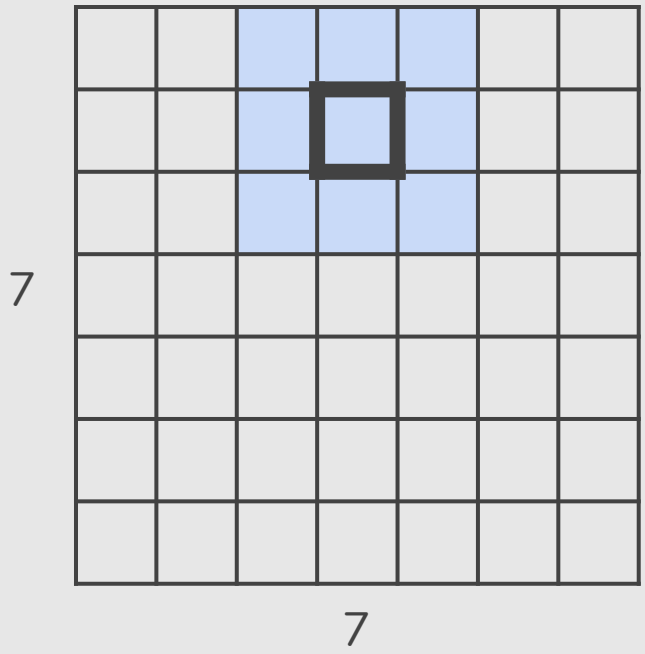
7 × 7 input (spatially)
assume 3 × 3 filter

A Closer Look at Spatial Dimensions



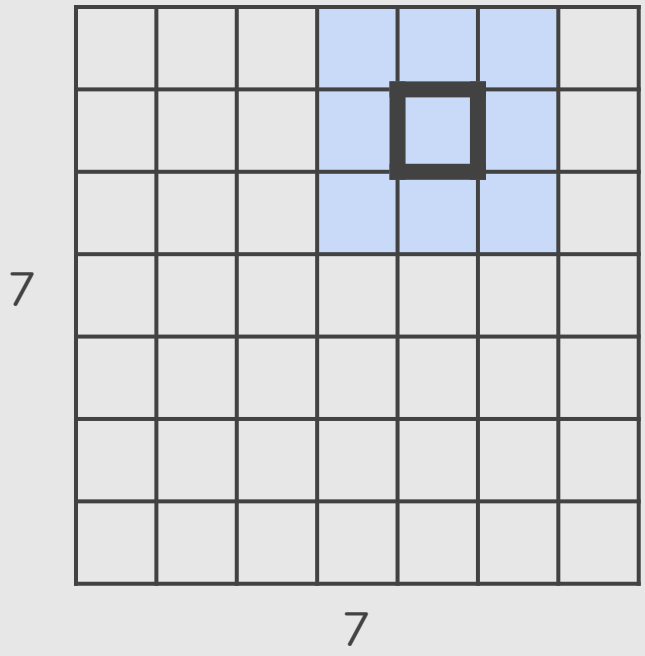
7 × 7 input (spatially)
assume 3 × 3 filter

A Closer Look at Spatial Dimensions



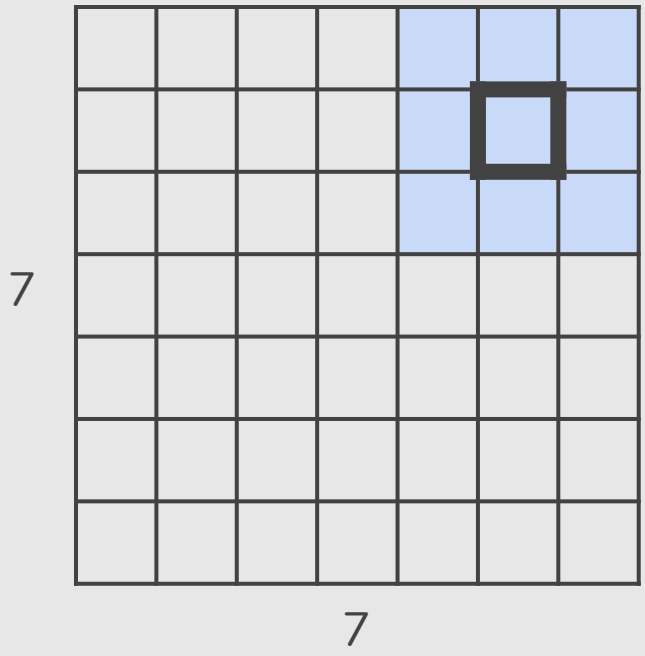
7 × 7 input (spatially)
assume 3 × 3 filter

A Closer Look at Spatial Dimensions



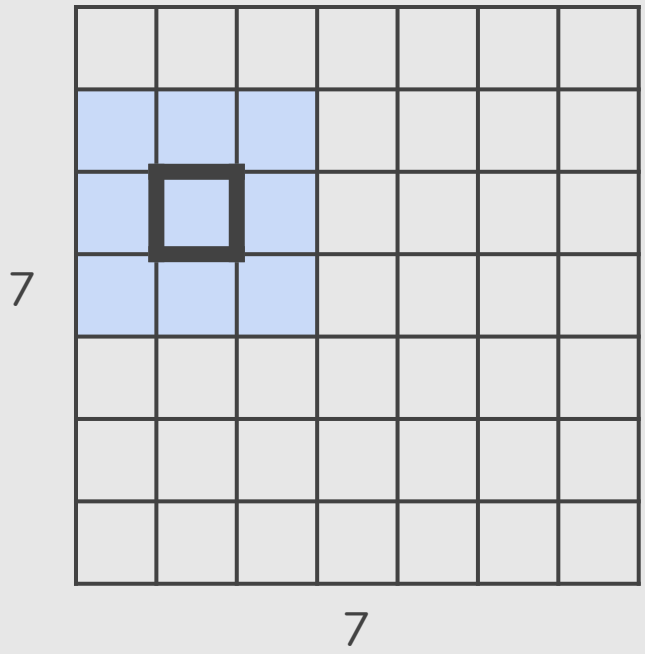
7×7 input (spatially)
assume 3×3 filter

A Closer Look at Spatial Dimensions



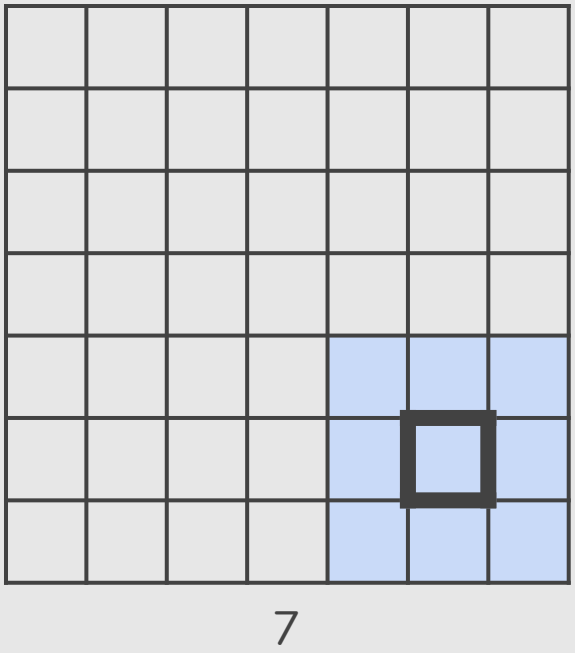
7 × 7 input (spatially)
assume 3 × 3 filter

A Closer Look at Spatial Dimensions



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

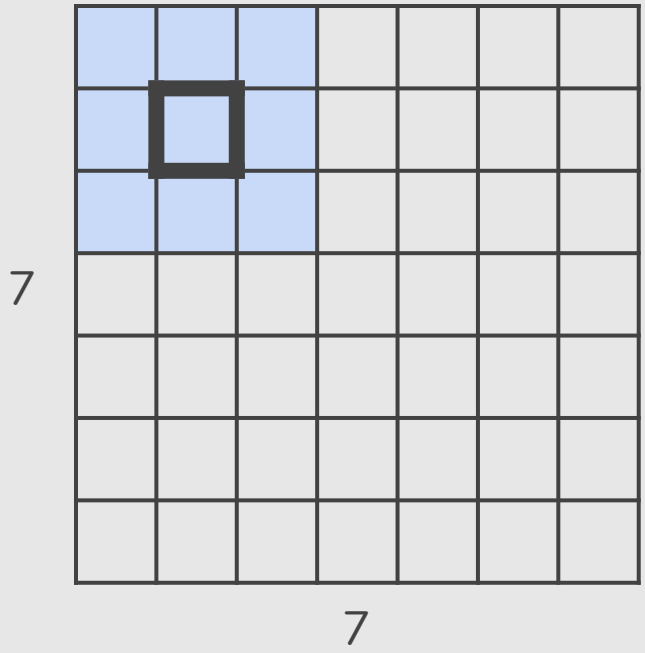
A Closer Look at Spatial Dimensions



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

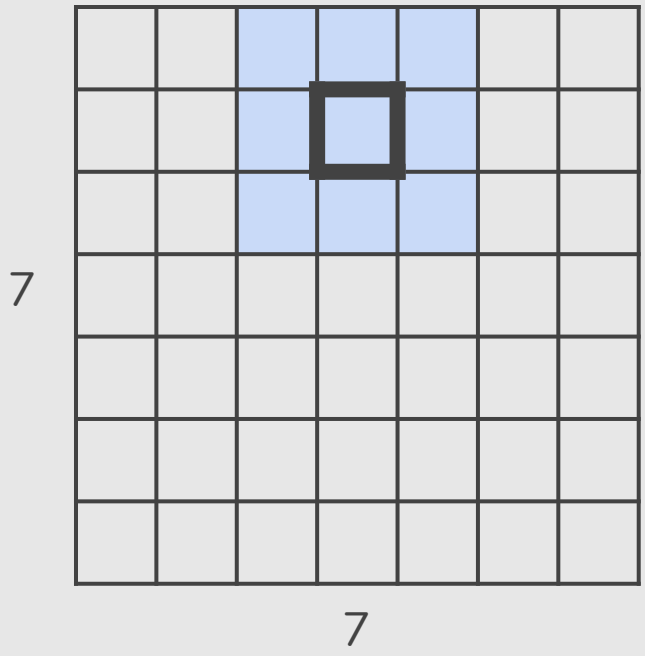
⇒ 5 × 5 output

A Closer Look at Spatial Dimensions



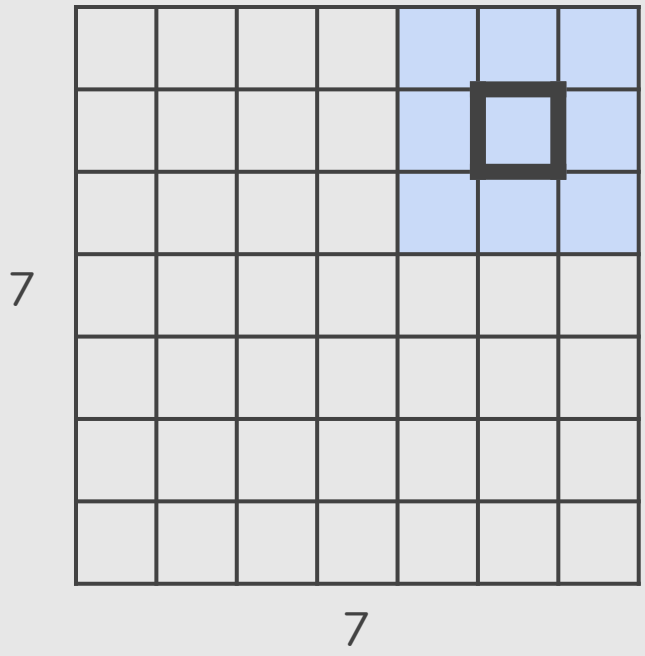
7 × 7 input (spatially)
assume 3 × 3 filter
applied with **stride 2**

A Closer Look at Spatial Dimensions



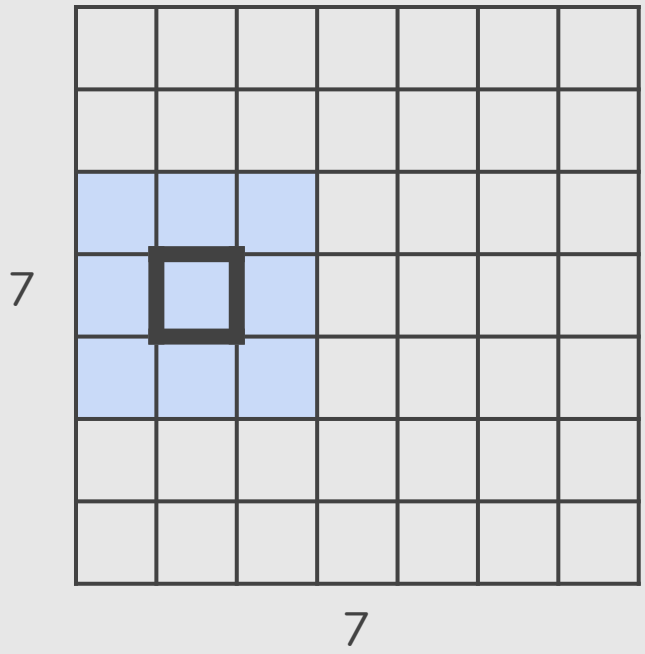
7 × 7 input (spatially)
assume 3 × 3 filter
applied with **stride 2**

A Closer Look at Spatial Dimensions



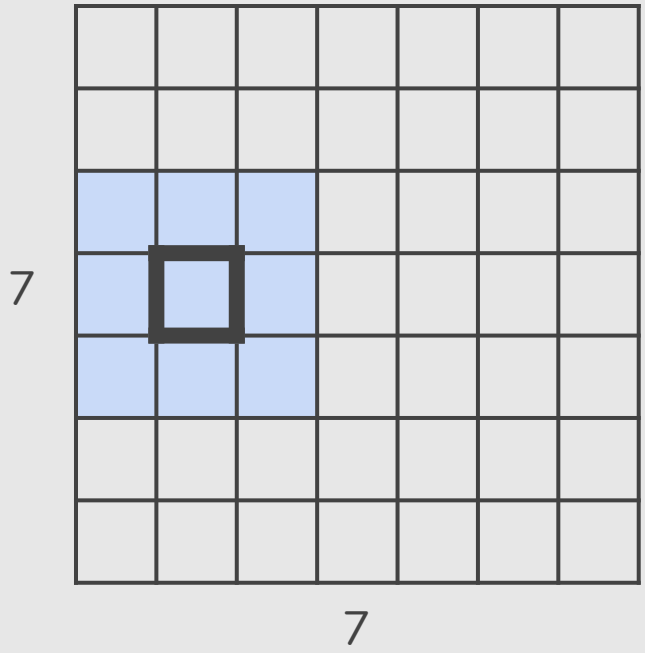
7 × 7 input (spatially)
assume 3 × 3 filter
applied with **stride 2**

A Closer Look at Spatial Dimensions



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

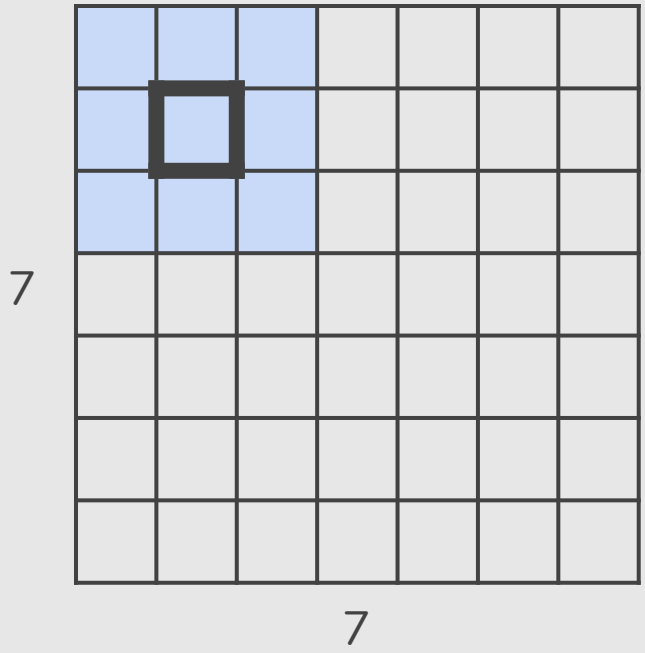
A Closer Look at Spatial Dimensions



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

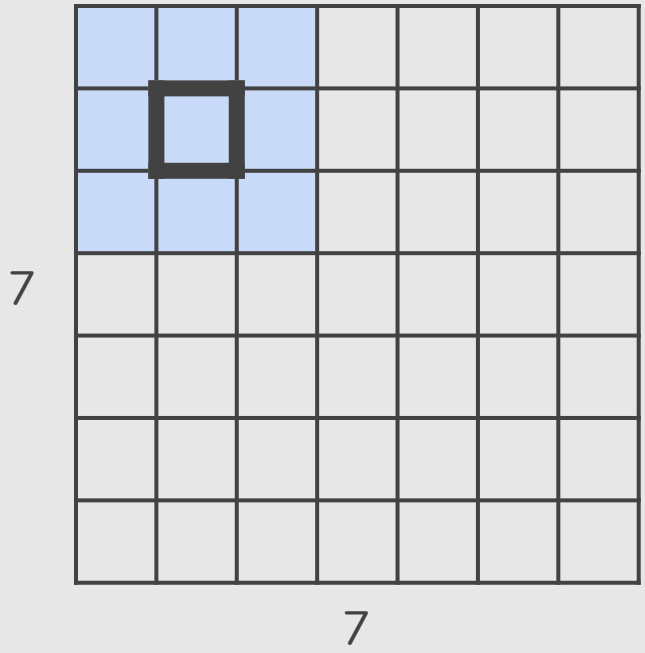
⇒ 3 × 3 output

A Closer Look at Spatial Dimensions



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

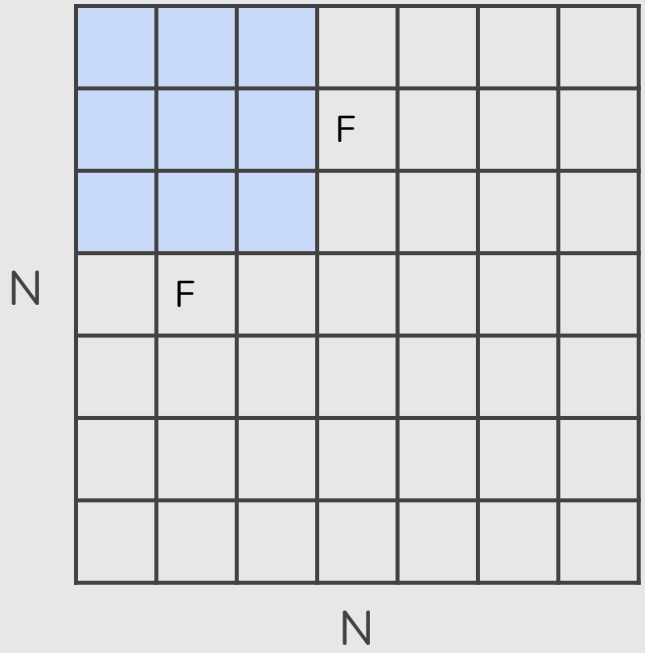
A Closer Look at Spatial Dimensions



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

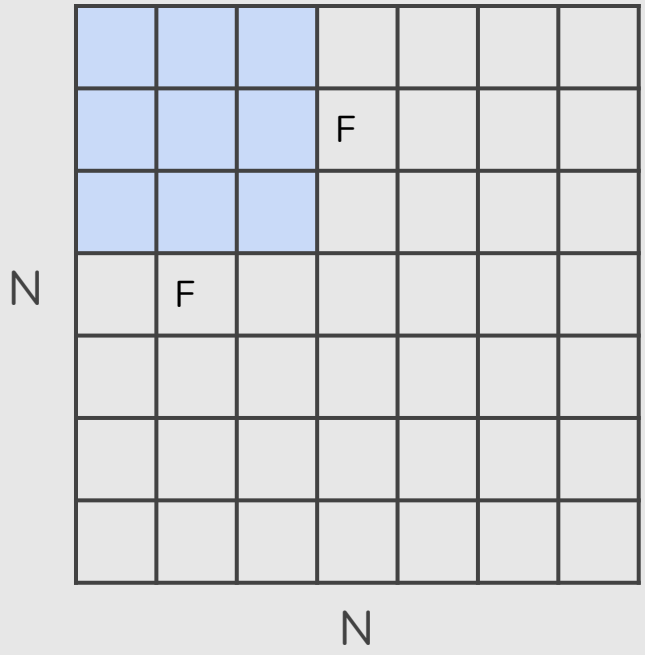
Doesn't fit!
cannot apply 3 × 3 filter on 7 × 7 input with stride 3.

A Closer Look at Spatial Dimensions



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

A Closer Look at Spatial Dimensions

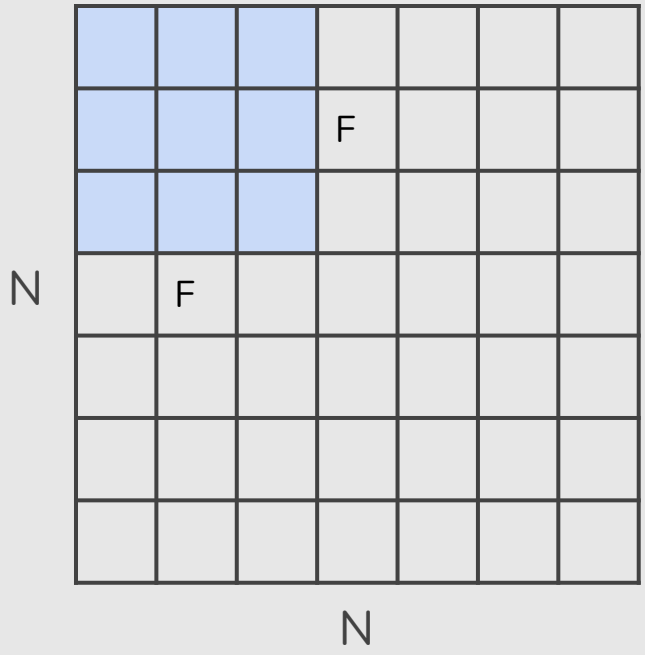


Output size:
(N - F) / stride + 1

e.g. N = 7, F = 3:

stride 1 ⇒ (7 - 3)/1 + 1 = 5

A Closer Look at Spatial Dimensions



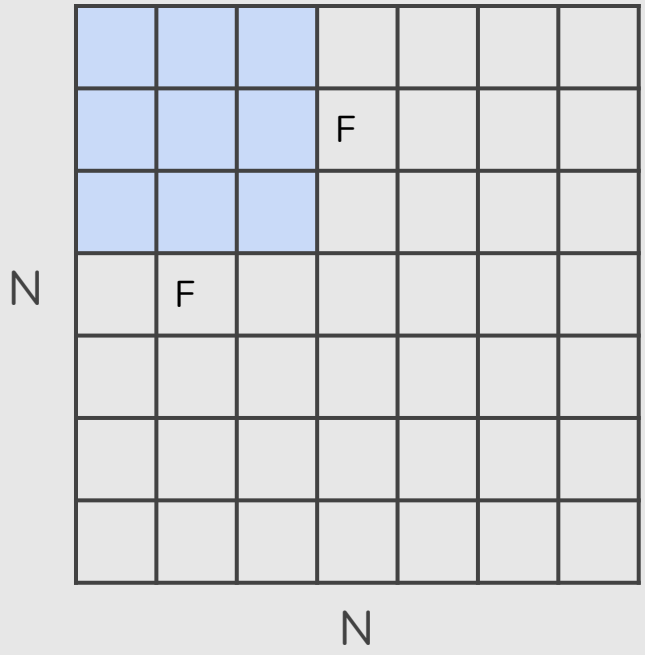
Output size:
(N - F) / stride + 1

e.g. N = 7, F = 3:

stride 1 ⇒ (7 - 3)/1 + 1 = 5

stride 2 ⇒ (7 - 3)/2 + 1 = 3

A Closer Look at Spatial Dimensions



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

e.g. N = 7, F = 3:

stride 1 ⇒ (7 - 3)/1 + 1 = 5

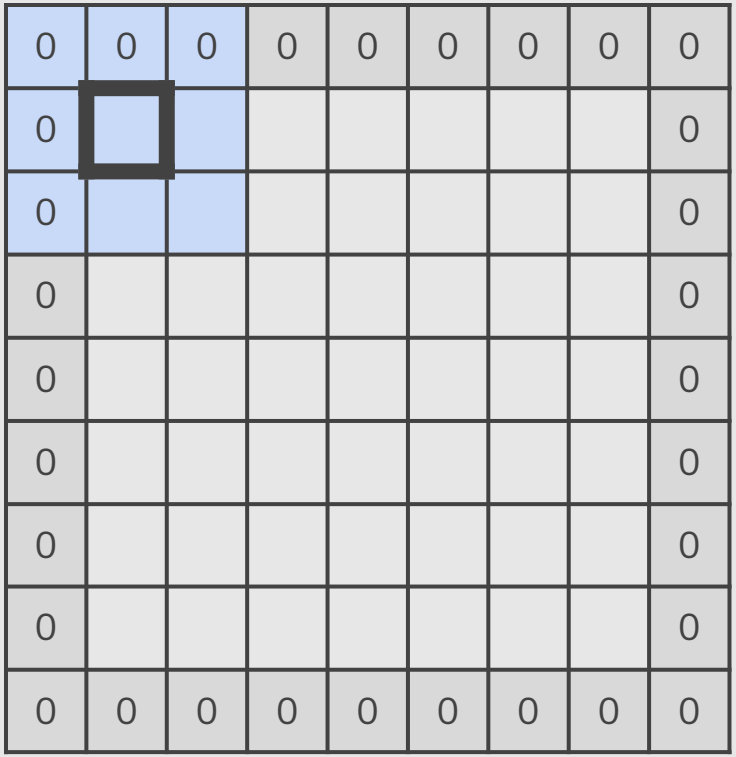
stride 2 ⇒ (7 - 3)/2 + 1 = 3

stride 3 ⇒ (7 - 3)/3 + 1 = 2.33

In Practice: Common to zero pad the border

0	0	0	0	0	0	0	0	0
0								0
0								0
0								0
0								0
0								0
0								0
0								0
0	0	0	0	0	0	0	0	0

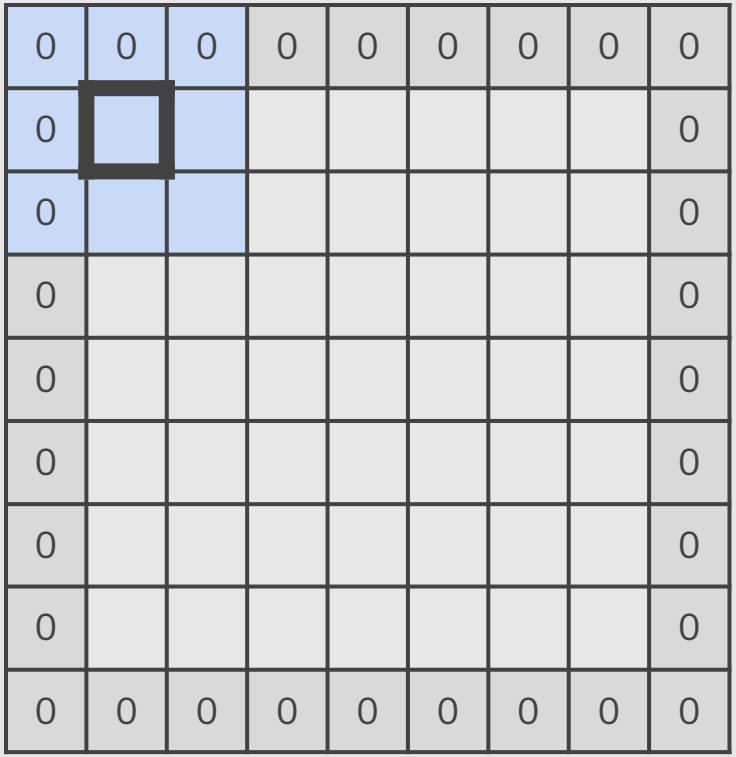
In Practice: Common to zero pad the border



7 × 7 input,
3 × 3 filter applied
with **stride 1 with pad 1**

What is the output?

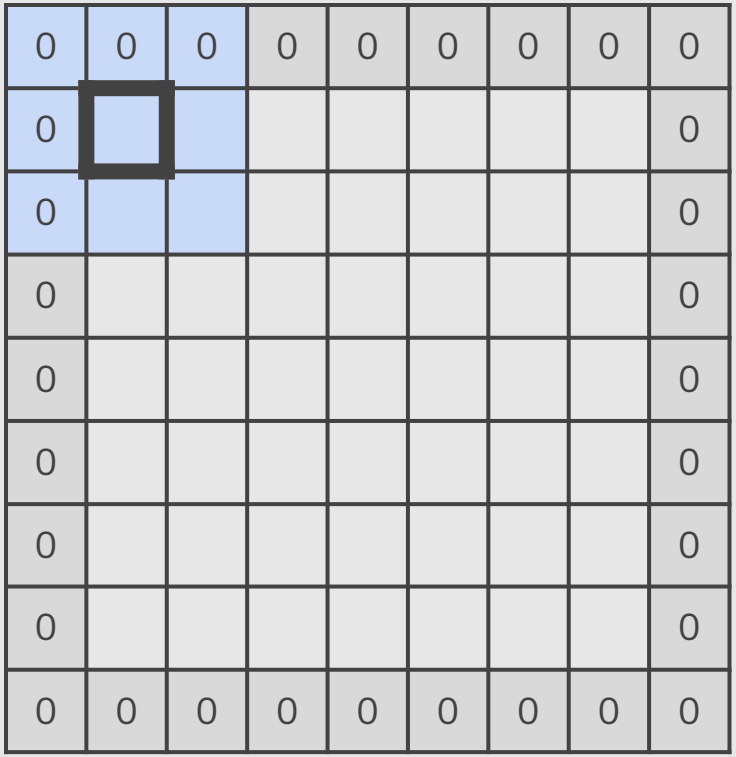
In Practice: Common to zero pad the border



7 × 7 input,
3 × 3 filter applied
with **stride 1 with pad 1**

What is the output?
7 × 7 output

In Practice: Common to zero pad the border



In general, common to see CONV layers with stride 1, filters of size F × F, and zero-padding with (F-1)/2 **(will preserve size spatially)**.

e.g. F = 3 ⇒ zero pad with 1

F = 5 ⇒ zero pad with 2

F = 7 ⇒ zero pad with 3

Partial Convolution based Padding

Guilin Liu Kevin J. Shih Ting-Chun Wang Fitsum A. Reda
Karan Sapra Zhiding Yu Andrew Tao Bryan Catanzaro
NVIDIA
{guilinl, kshih, tingchunw, freda, ksapra, zhidingy, atao, bcatanzaro}@nvidia.com

Abstract

In this paper, we present a simple yet effective padding scheme that can be used as a drop-in module for existing convolutional neural networks. We call it ***partial convolution based padding****, with the intuition that the padded region can be treated as holes and the original input as non-holes. Specifically, during the convolution operation, the convolution results are re-weighted near image borders based on the ratios between the padded area and the convolution sliding window area. Extensive experiments with various deep network models on ImageNet classification and semantic segmentation demonstrate that the proposed padding scheme consistently outperforms standard zero padding with better accuracy. The code is available at https://github.com/NVIDIA/partialconv*

1. Introduction

Convolutional operation often requires padding when part of the filter extends beyond the input image or feature map. Standard approaches include zero padding (extend with zeros), reflection padding (reflect the input values across the border axis) and replication padding (extend by replicating the values along borders). Among them, the most commonly used scheme is zero padding, as was adopted by [13]. Besides its simplicity, zero padding is also computationally more efficient than the other two schemes. Yet, there is no consensus on which padding scheme is the best yet. In this work, we conduct extensive experiments on the ImageNet classification task using these three padding

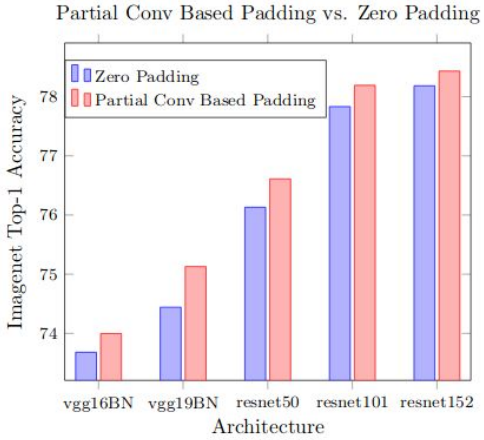


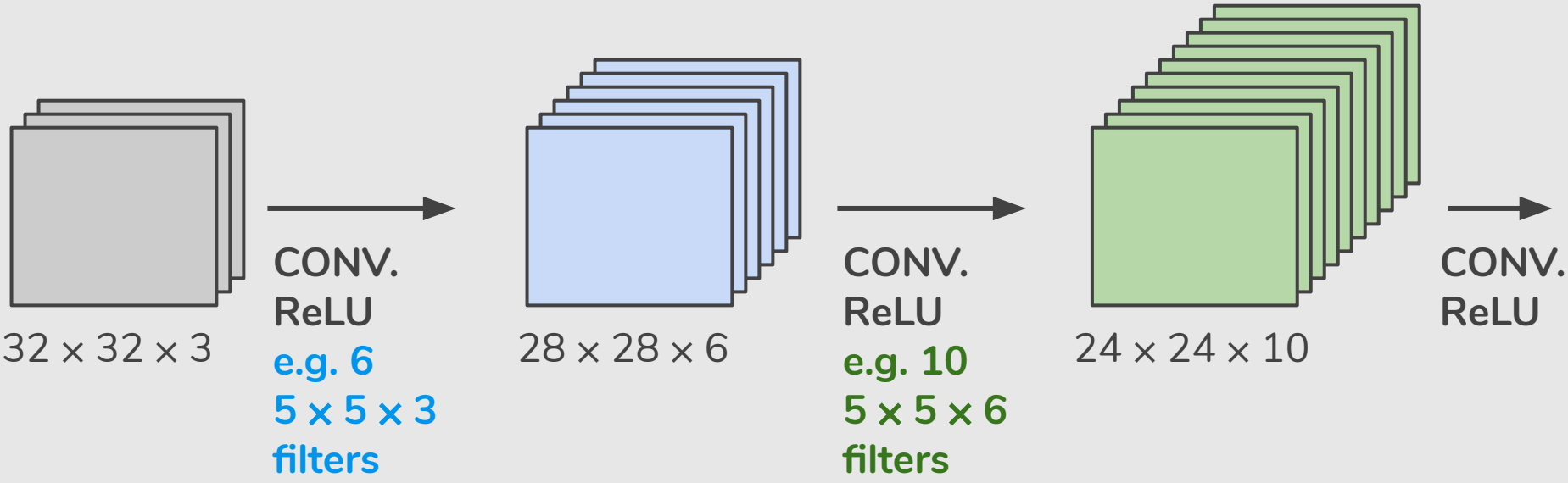
Figure 1. Comparison of the ImageNet classification top-1 accuracy with center crop between partial convolution based padding (in red) and zero padding (in blue) on VGG and ResNet networks. vgg16BN and vgg19BN represent the vgg16 network and vgg19 network with batch normalization layers.

adding extra unrelated data to the input. Reflection and replication padding attempt to pad with plausible data values by re-using what is along the borders of the input. These two types of padding lead to unrealistic image patterns since only some parts of the input are replicated. Moreover, for all these three padding schemes, the added or replicated fea-

https://arxiv.org/abs/1811.11718

Shrinking too fast is not good, doesn't work well.

32 → 28 → 24 → ...



Number of Parameters

Input volume: **32 × 32 × 3**

10 5 × 5 filters with stride 1, pad 2

Number of parameters in this layer?

Number of Parameters

Input volume: **32 x 32 x 3**

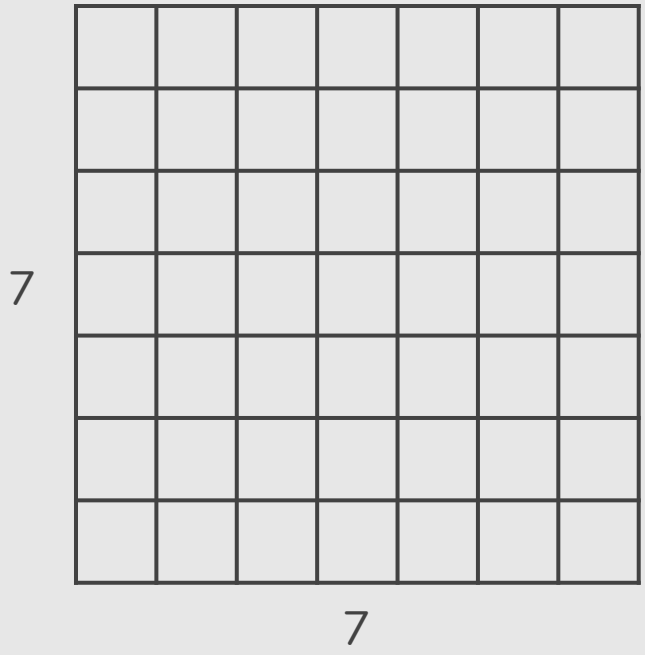
10 **5 x 5** filters with stride 1, pad 2

Number of parameters in this layer?

Each filter has 5*5*3 + 1 = **76** parameters (+1 for bias)

=> **76***10 = 760

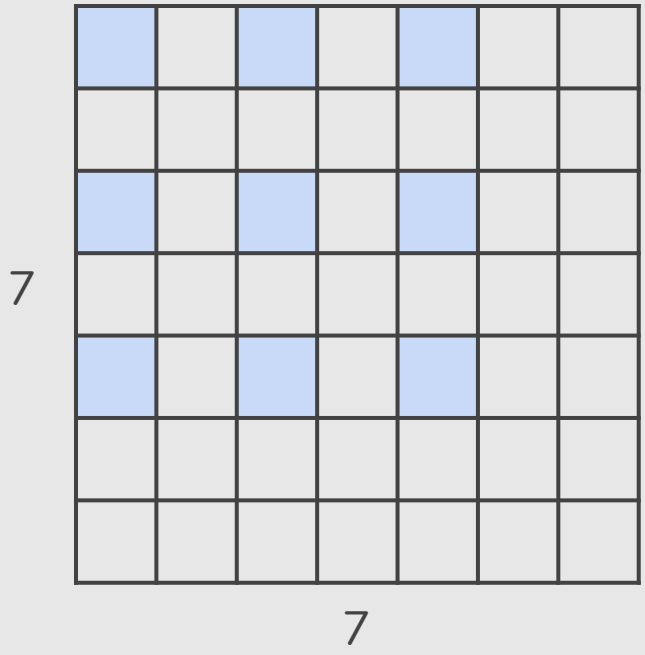
Dilated Convolution



7 × 7 input (spatially)
assume 3 × 3 filter
with **stride 1 with dilation 2**

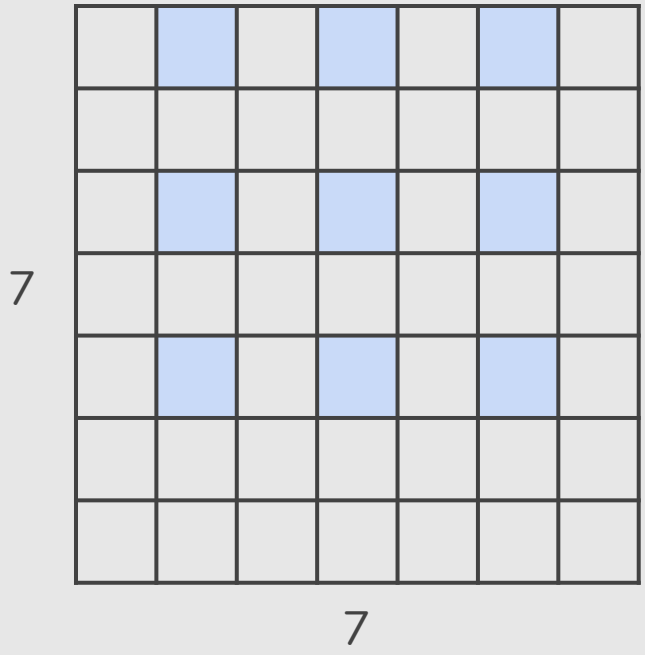
"Multi-Scale Context Aggregation by Dilated Convolutions", ICLR 2016, https://arxiv.org/abs/1511.07122

Dilated Convolution



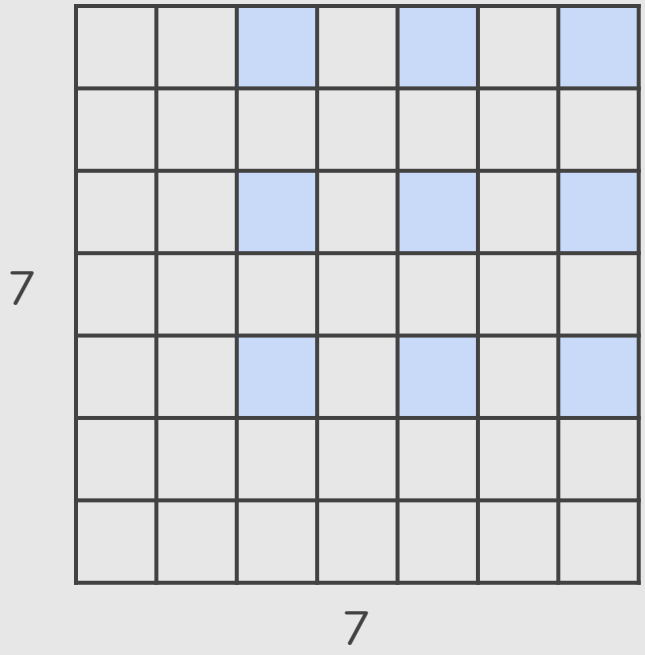
7 × 7 input (spatially)
assume 3 × 3 filter
with **stride 1 with dilation 2**

Dilated Convolution



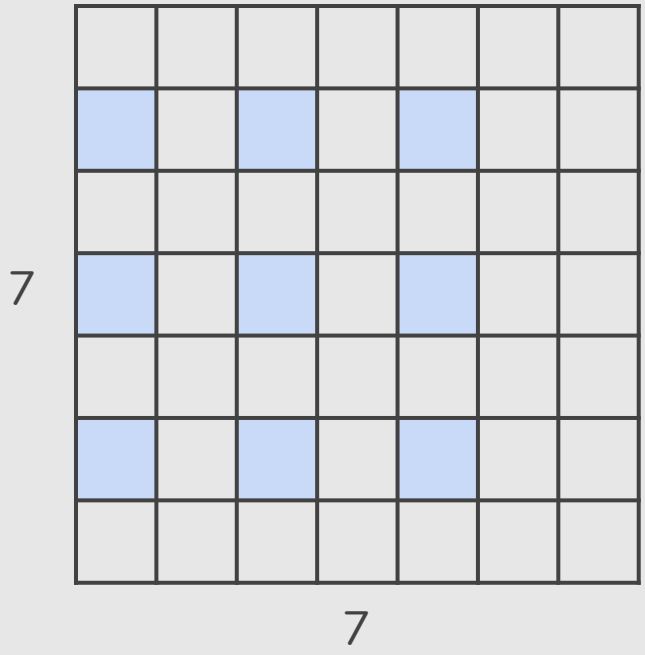
7 × 7 input (spatially)
assume 3 × 3 filter
with **stride 1 with dilation 2**

Dilated Convolution



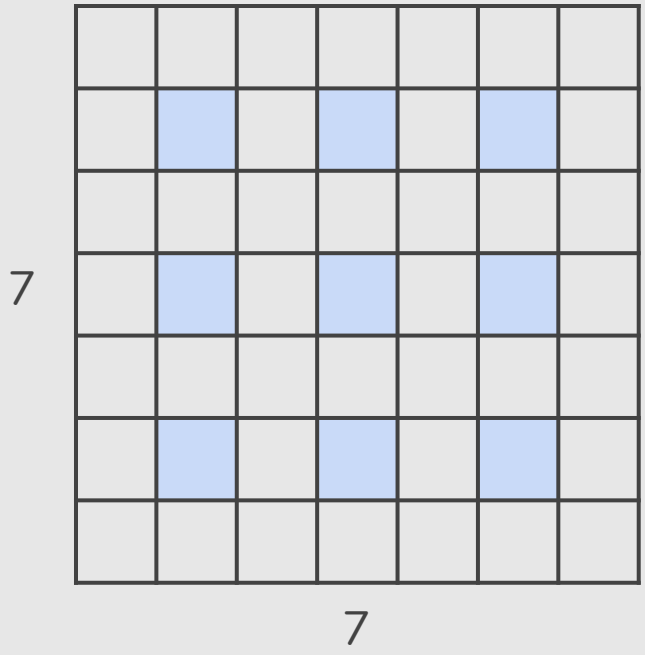
7 × 7 input (spatially)
assume 3 × 3 filter
with **stride 1 with dilation 2**

Dilated Convolution



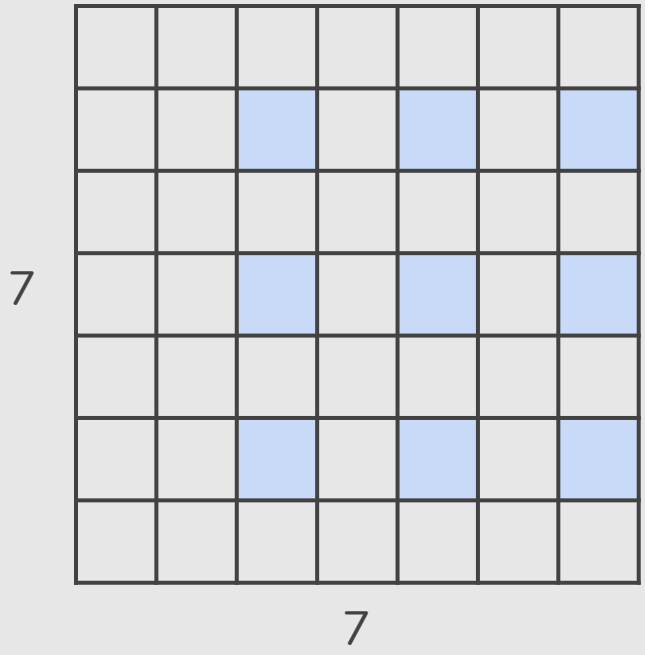
7 × 7 input (spatially)
assume 3 × 3 filter
with **stride 1 with dilation 2**

Dilated Convolution



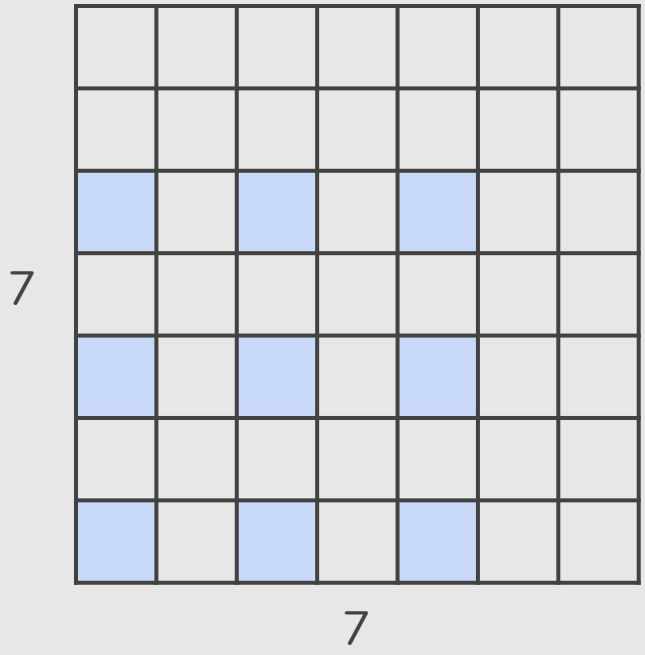
7 × 7 input (spatially)
assume 3 × 3 filter
with **stride 1 with dilation 2**

Dilated Convolution



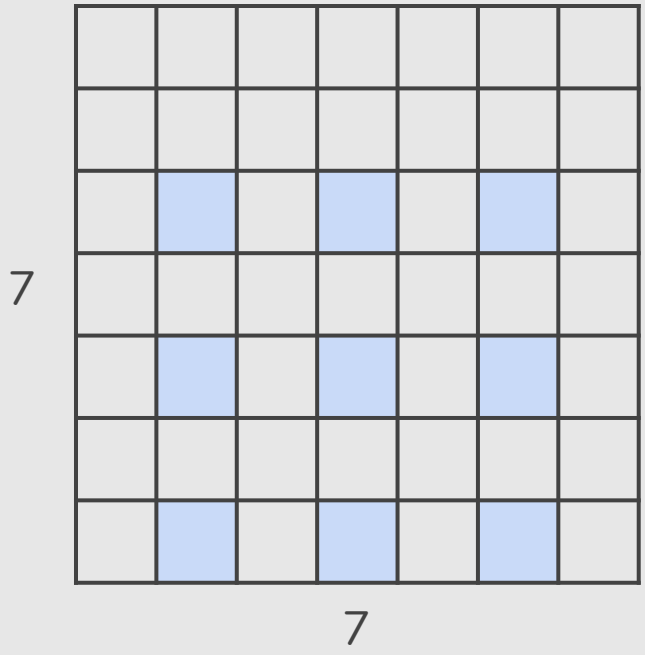
7 × 7 input (spatially)
assume 3 × 3 filter
with **stride 1 with dilation 2**

Dilated Convolution



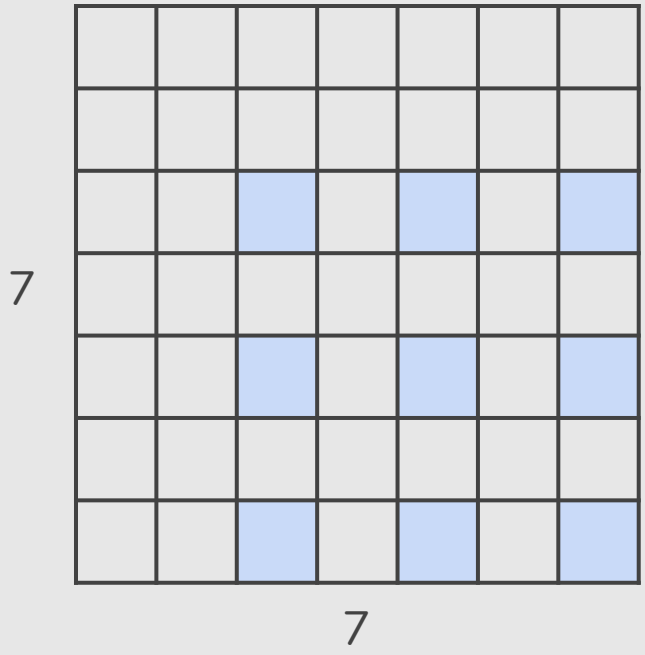
7 × 7 input (spatially)
assume 3 × 3 filter
with **stride 1 with dilation 2**

Dilated Convolution

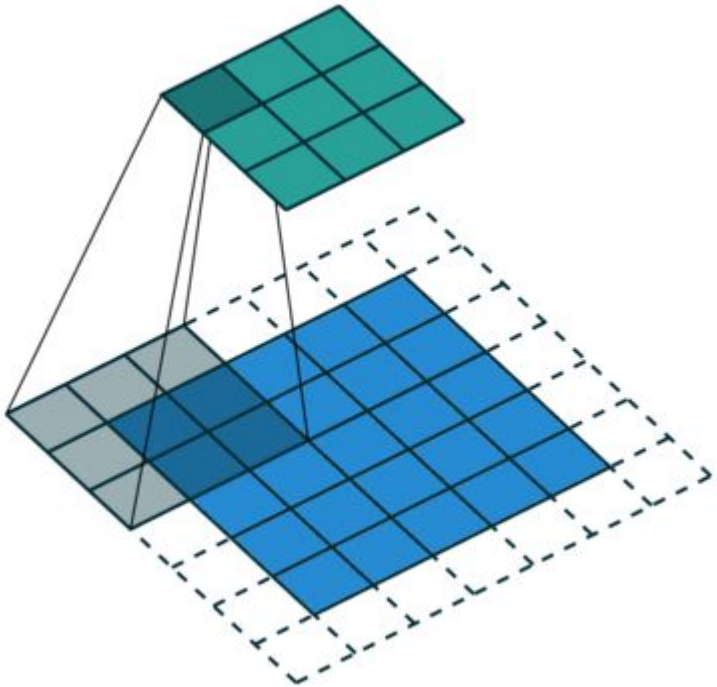


7 × 7 input (spatially)
assume 3 × 3 filter
with **stride 1 with dilation 2**

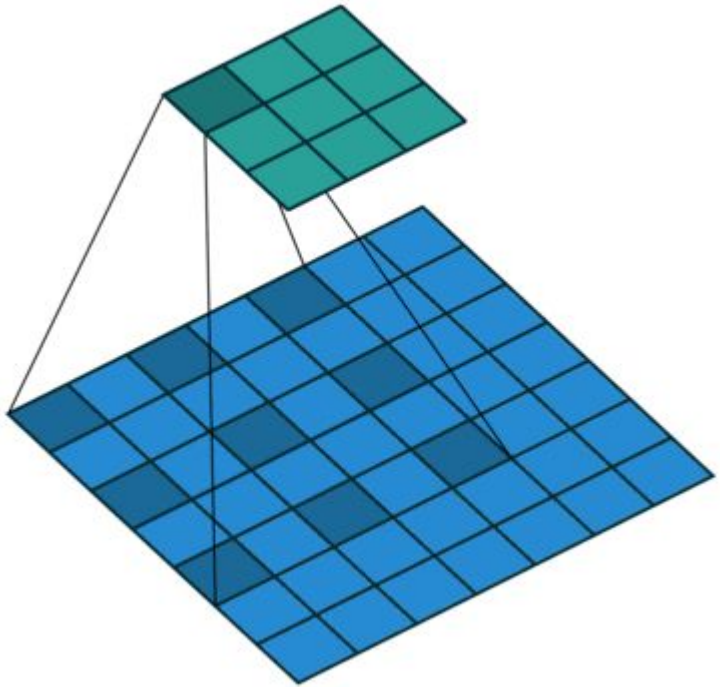
Dilated Convolution



7 × 7 input (spatially)
assume 3 × 3 filter
with **stride 1 with dilation 2**



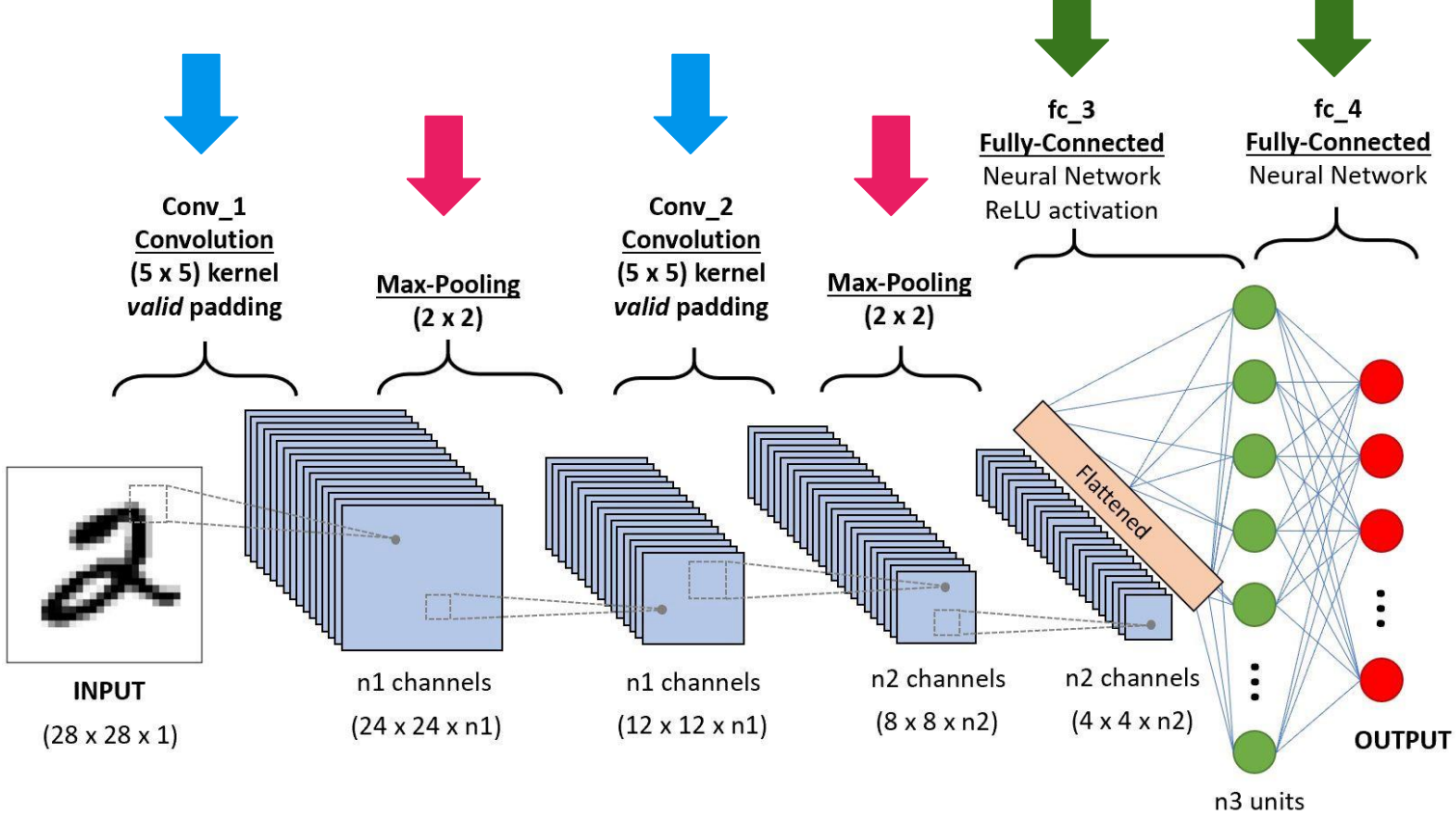
Standard Convolution



Dilated Convolution

Convolutions

“A Guide to Convolution Arithmetic for Deep Learning” https://arxiv.org/pdf/1603.07285 (Janeiro, 2018)



There are a few distinct types of layers (e.g., CONV/POOL/FC are by far the most popular).

Pooling Layer

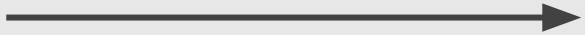
- Makes the representations smaller and more manageable
- Operates over each activation map independently

Pooling Layer

- Makes the representations smaller and more manageable
- Operates over each activation map independently

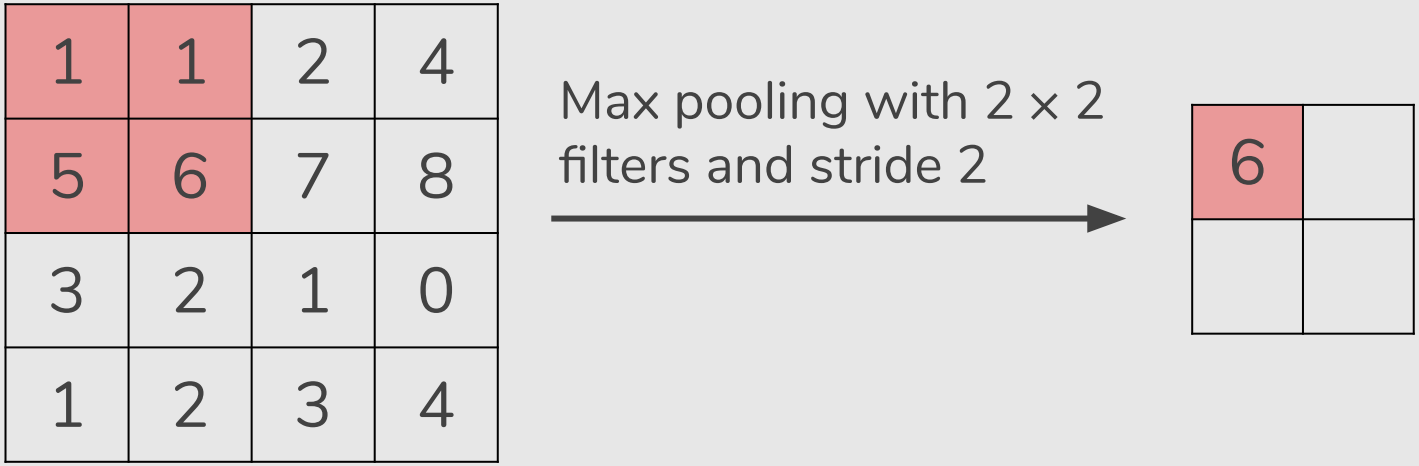
1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

Max pooling with 2 x 2 filters and stride 2



Pooling Layer

- Makes the representations smaller and more manageable
- Operates over each activation map independently



Pooling Layer

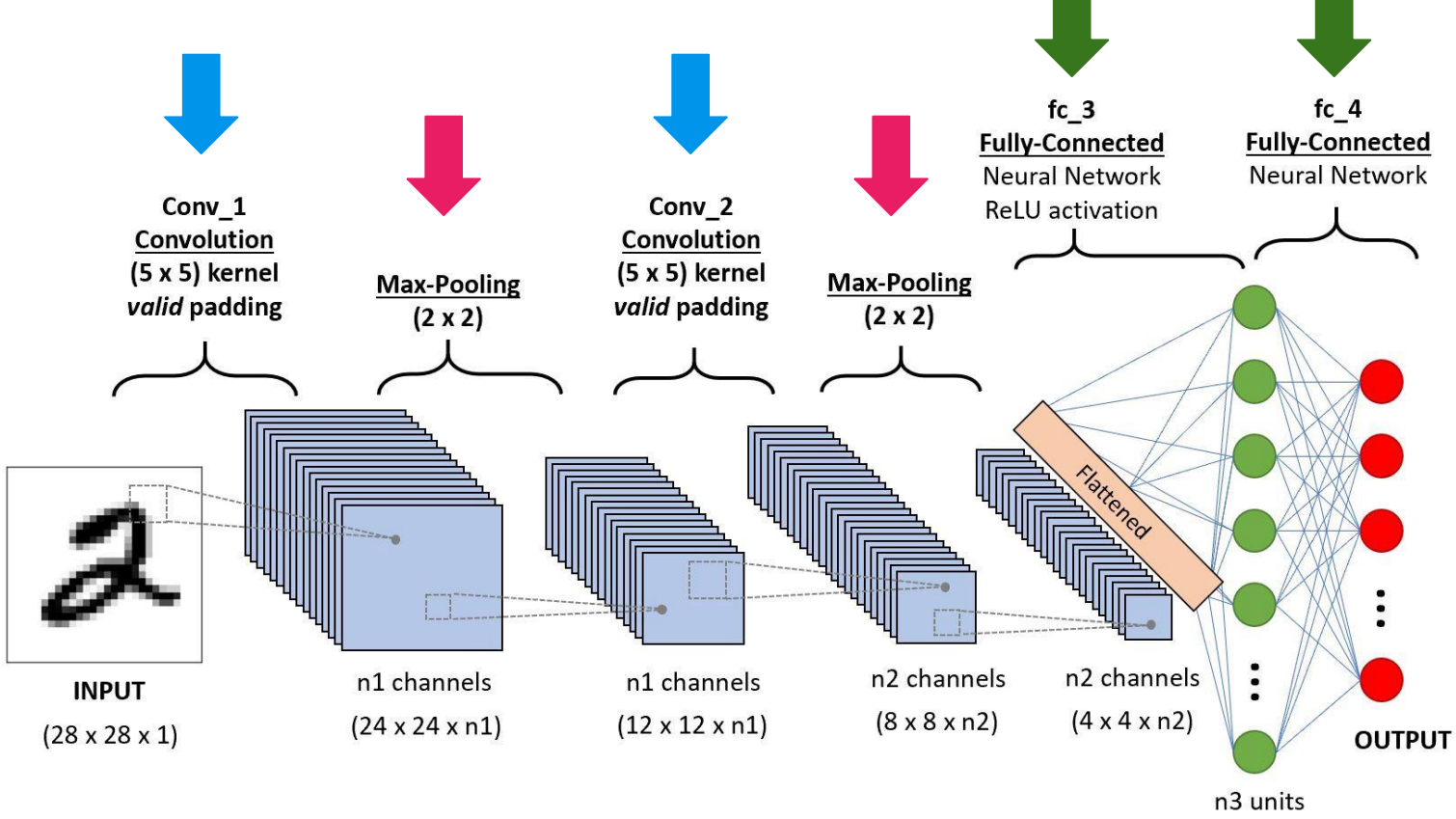
- Makes the representations smaller and more manageable
- Operates over each activation map independently

Pooling Layer

- Makes the representations smaller and more manageable
- Operates over each activation map independently

Pooling Layer

- Makes the representations smaller and more manageable
- Operates over each activation map independently

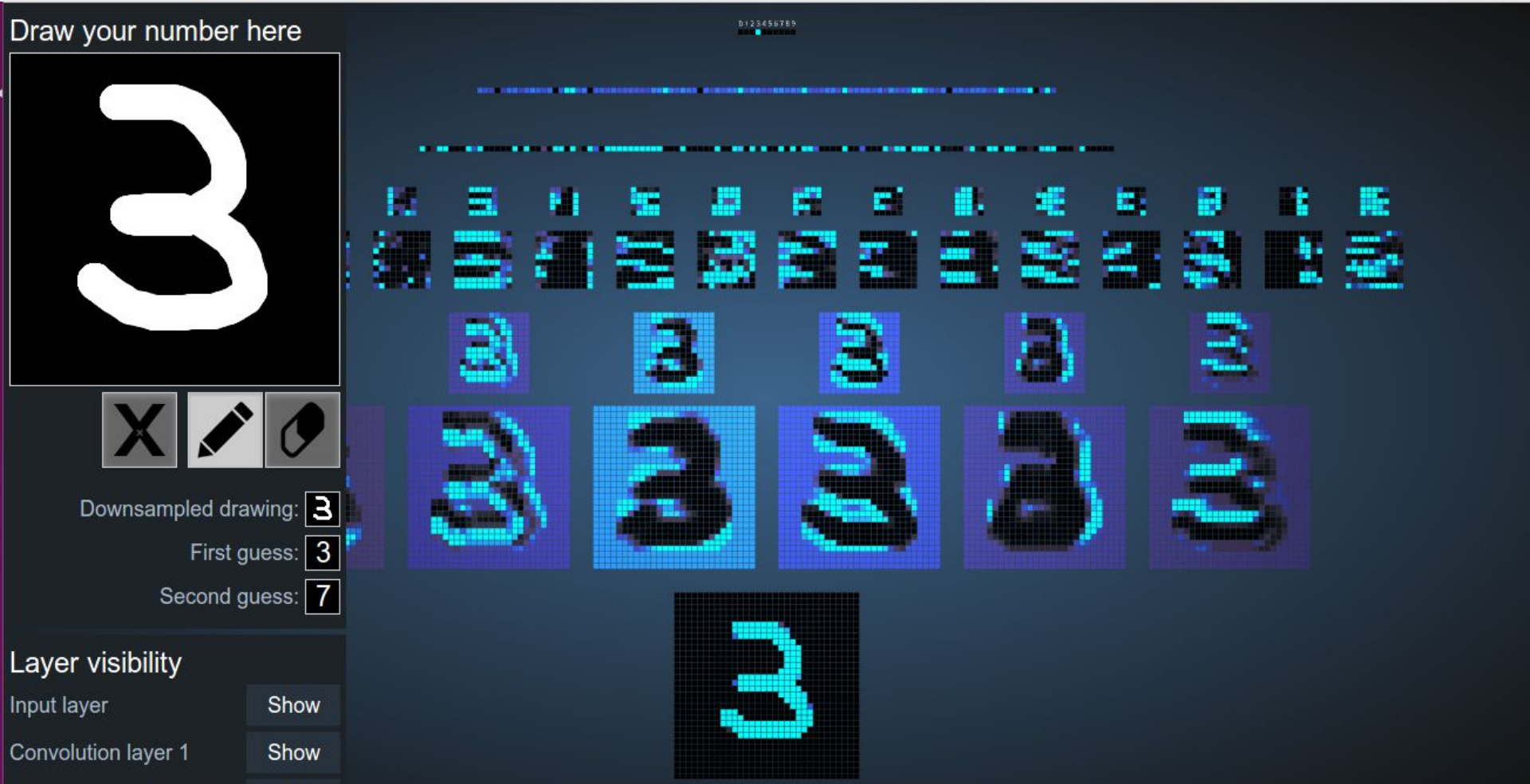


There are a few distinct types of layers (e.g., CONV/POOL/FC are by far the most popular).

Fully Connected Layer

- Contains neurons that connect to the entire input volume, as in ordinary Neural Networks

http://scs.ryerson.ca/~aharley/vis/conv/flat.html



Visualizing a CNN trained on Handwritten Digits

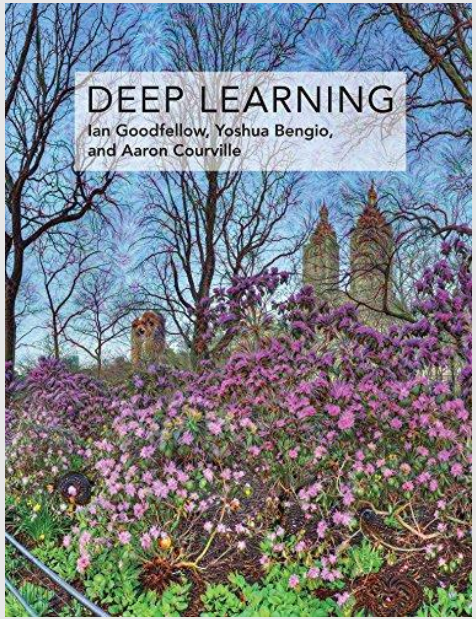
- Input image: 1024 pixels (32 x 32 image)
- CONV 1 (+ RELU): 6 5 × 5 (stride 1) filters
- POOL 1: 2 × 2 max pooling (with stride 2)
- CONV 2 (+ RELU): 16 5 × 5 (stride 1) filters
- POOL 2: 2 × 2 max pooling (with stride 2)
- 2 FC layers:
  - 120 neurons in the first FC layer
  - 100 neurons in the second FC layer
- Output layer: 10 neurons in the third FC

http://neuralnetworksanddeeplearning.com/chap6.html#final_conv

Convolutional neural networks in practice

We've now seen the core ideas behind convolutional neural networks. Let's look at how they work in practice, by implementing some convolutional networks, and applying them to the MNIST digit classification problem. The program we'll use to do this is called `network3.py`, and it's an improved version of the programs `network.py` and `network2.py` developed in earlier chapters*. If you wish to follow along, the code is available on GitHub. Note that we'll work through the code for `network3.py` itself in the next section. In this section, we'll use `network3.py` as a library to build convolutional networks.

*Note also that `network3.py` incorporates ideas from the Theano library's documentation on convolutional neural nets (notably the implementation of LeNet-5), from Misha Denil's implementation of dropout, and from Chris Olah.



“Deep Learning”, Goodfellow & Bengio & Courville, 2016.

http://www.deeplearningbook.org/contents/convnets.html

Chapter 9

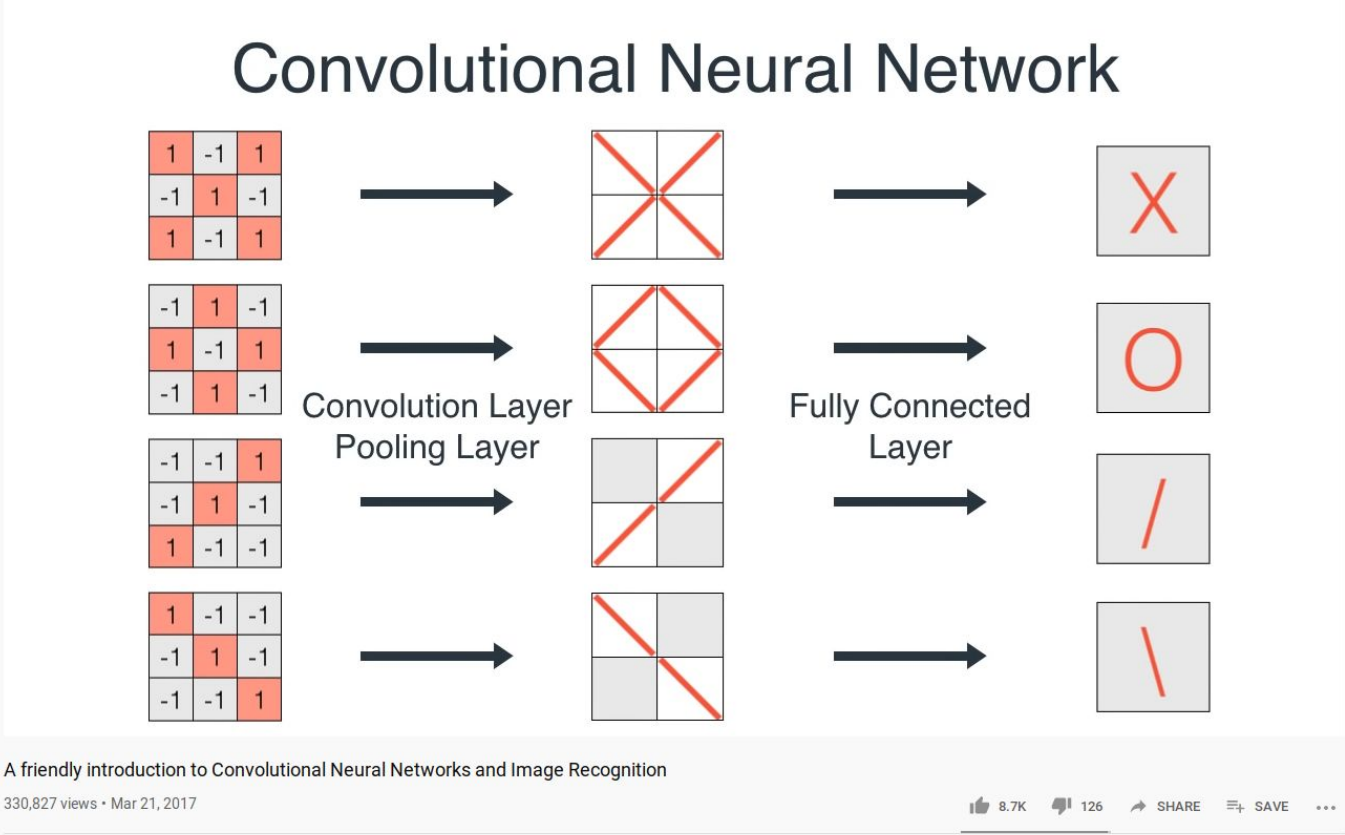
Convolutional Networks

Convolutional networks (LeCun, 1989), also known as **convolutional neural networks**, or CNNs, are a specialized kind of neural network for processing data that has a known grid-like topology. Examples include time-series data, which can be thought of as a 1-D grid taking samples at regular time intervals, and image data, which can be thought of as a 2-D grid of pixels. Convolutional networks have been tremendously successful in practical applications. The name “convolutional neural network” indicates that the network employs a mathematical operation called **convolution**. Convolution is a specialized kind of linear operation. *Convolutional networks are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers.*

In this chapter, we first describe what convolution is. Next, we explain the motivation behind using convolution in a neural network. We then describe an operation called **pooling**, which almost all convolutional networks employ. Usually, the operation used in a convolutional neural network does not correspond precisely to the definition of convolution as used in other fields, such as engineering or pure mathematics. We describe several variants on the convolution function that are widely used in practice for neural networks. We also show how convolution may be applied to many kinds of data, with different numbers of dimensions. We then discuss means of making convolution more efficient. Convolutional networks stand out as an example of neuroscientific principles influencing deep learning. We discuss these neuroscientific principles, then conclude with comments about the role convolutional networks have played in the history of deep learning. One topic this chapter does not address is how to choose the architecture of your convolutional network. The goal of this chapter is to describe the kinds of tools that convolutional networks provide, while chapter 11 describes general guidelines

326

“A friendly introduction to Convolutional Neural Networks and Image Recognition” https://youtu.be/2-Ol7ZB0MmU



References

Machine Learning Books

- Hands-On Machine Learning with Scikit-Learn and TensorFlow, Chap. 11 & 13

Machine Learning Courses

- https://www.coursera.org/learn/neural-networks
- “The 3 popular courses on Deep Learning”: https://medium.com/towards-data-science/the-3-popular-courses-for-deeplearning-ai-ac37d4433bd