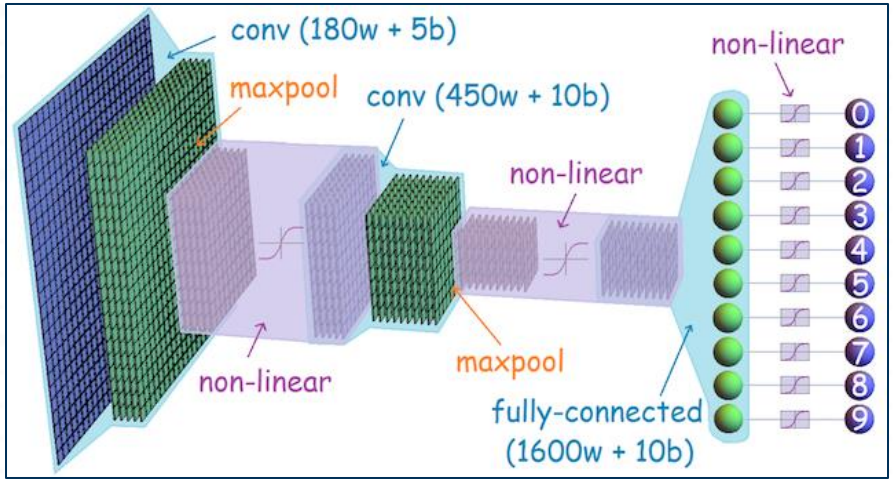
ECE 4252/8803: Fundamentals of Machine Learning (FunML)
Fall 2024

Lecture 14: Convolutional Neural Networks



OLIVES @GeorgiaTech

Georgia Tech

Overview

In this Lecture..

Locally Connected Layer

- Limitations of ANNNs
- Convolutional Layers

Introduction to Convolutional Neural Networks

Layers used to build Convolutional Neural Networks

- Convolutional layer
- Pooling Layer
- Fully-connected layers

Examples of Deep CNNs Architectures

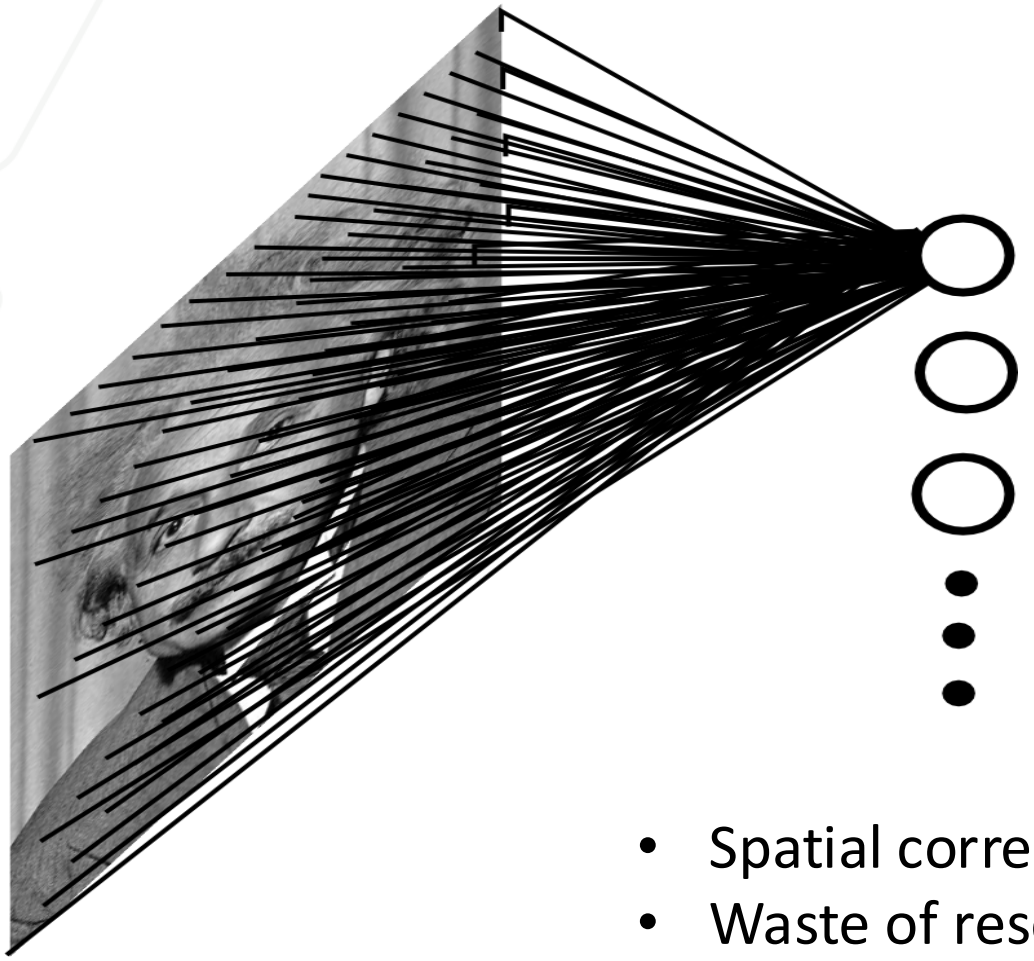
Training CNNs

- Optimization

Visualization of Convolutional Neural Networks

MLPs/ANNs

Limitations of Multi Layer Perceptrons



Example of a single layer Neural Network:

Input: 200x200 pixels image (vectorized image $\boldsymbol{x} \in \mathbb{R}^{4\times10^4}$)

For a single neuron, $\boldsymbol{w}_i \in \mathbb{R}^{4\times10^4}$ (4×10^4 weights)

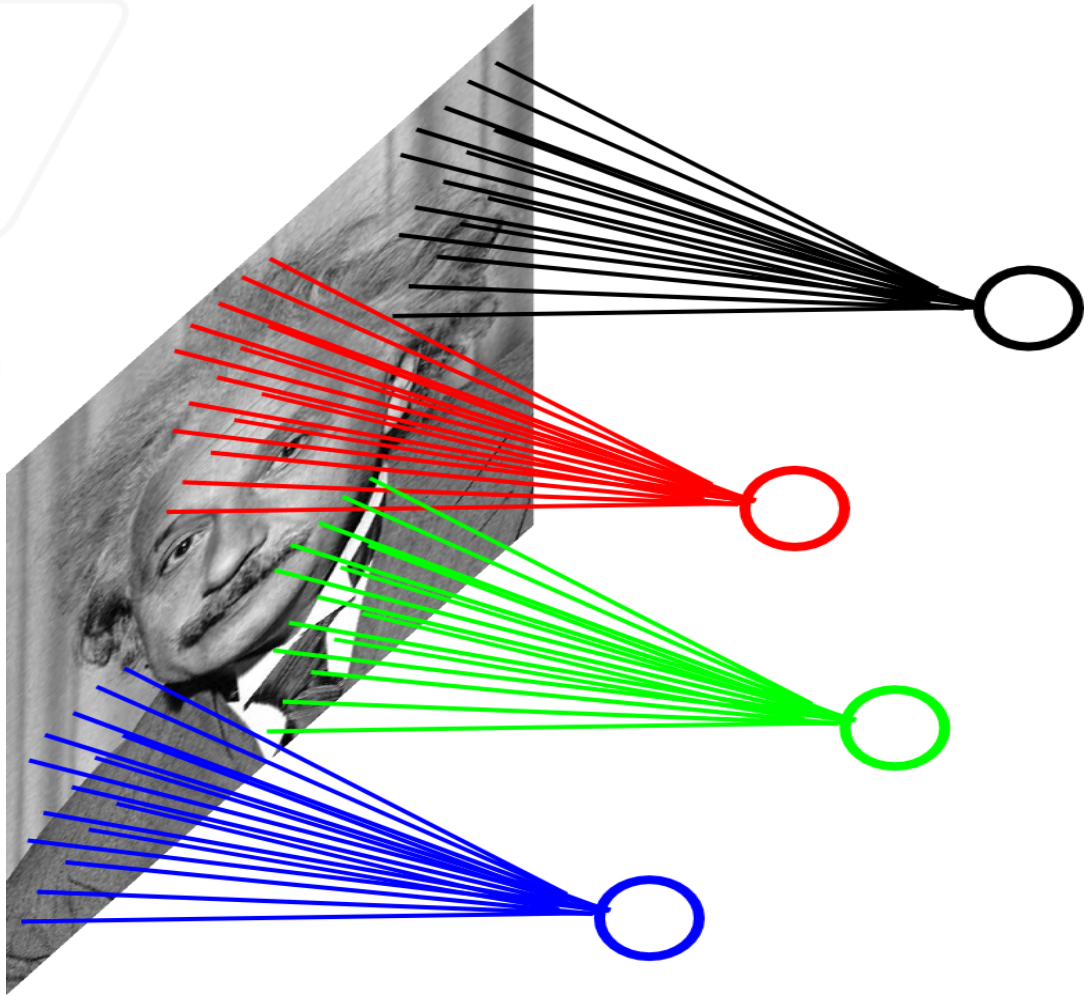
For 100 neurons: 4×10^4 x100

~4M parameters!!!

- Spatial correlation is local
- Waste of resources while not enough training samples…

Locally Connected Neural networks

Local Connectivity



Example of a locally connected layer:

Input: 200x200 pixels image (vectorized image $\boldsymbol{x} \in \mathbb{R}^{4\times10^4}$)

For a single neuron (filter), $\boldsymbol{w}_i \in \mathbb{R}^{3\times3}$ (9 **shared** weights)

For 100 neurons: 9 x100

900 parameters

Locally Connected Neural networks

Local Connectivity

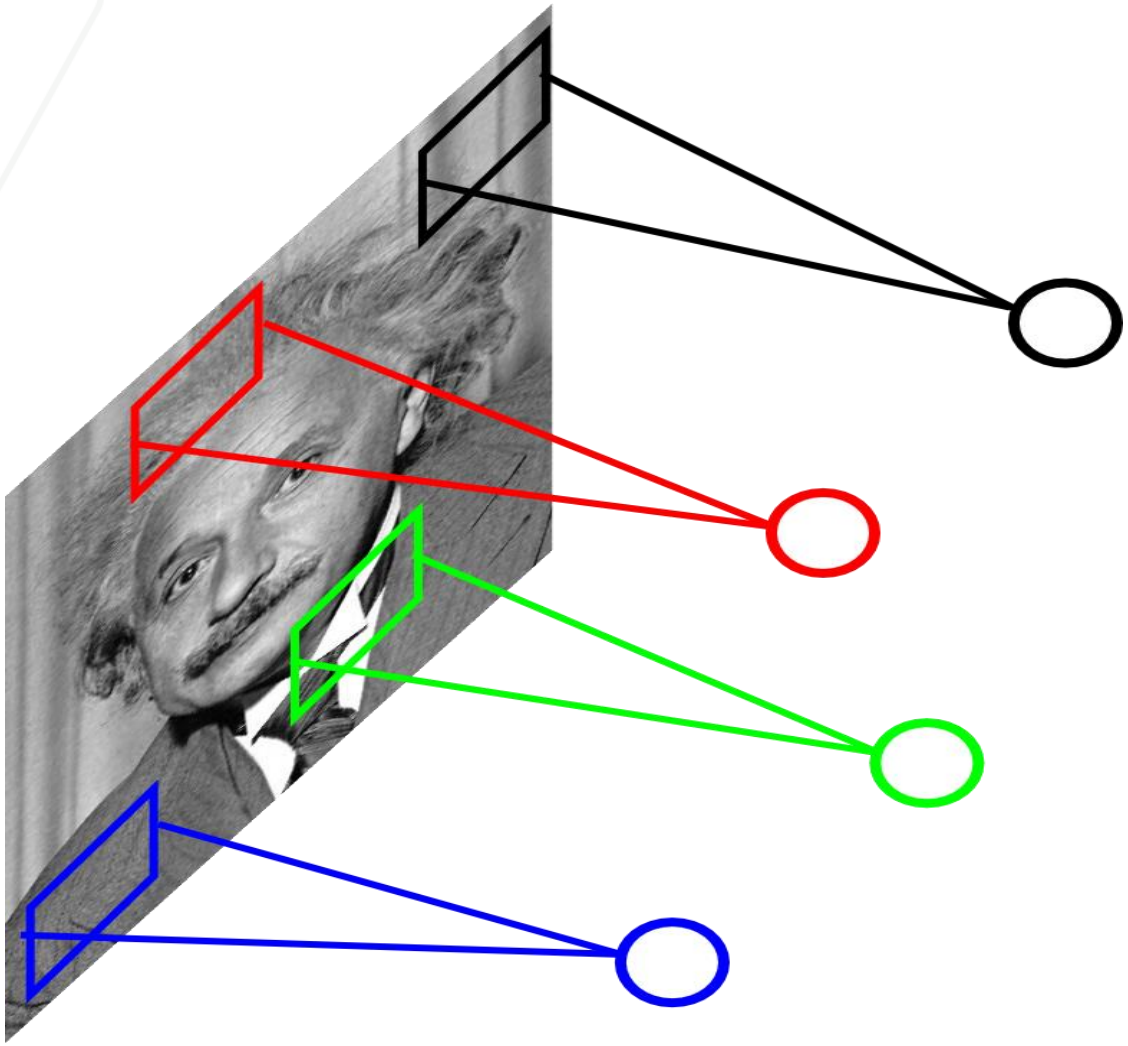
This is good when:

- The inputs are (roughly) registered (e.g., face recognition)

- The object can be anywhere

Locally Connected Neural networks

Local Connectivity



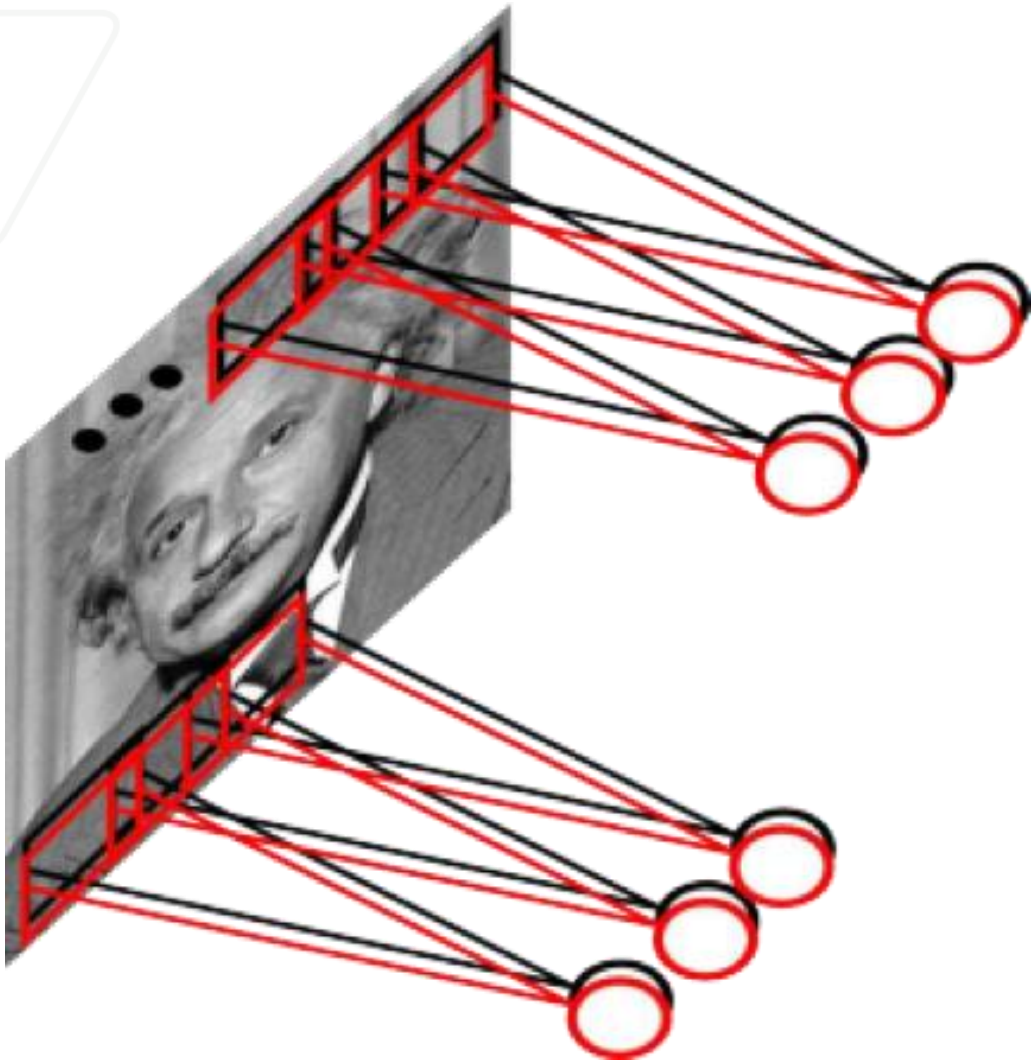
STATIONARITY?

Statistics are similar at different locations

Material adapted from Marc'Aurelio Ranzato, Deep Learning CVPR 2012 Tutorial

Locally Connected Neural networks

Parameter Sharing



- Connect each neuron to an input patch
- Share the same parameters across different locations (assuming input is stationary):

Convolutions with learned kernels

Material adapted from Marc'Aurelio Ranzato, Deep Learning CVPR 2012 Tutorial

Locally Connected Neural networks

ANNs vs CNNs

Artificial Neural Networks:

- number of parameters scales quadratically with the size of the input
- does not leverage stationarity

Convolutional layers:

- connect each neuron to a patch of the input
- share the weight across spatial locations

Material adapted from Marc'Aurelio Ranzato, Deep Learning CVPR 2012 Tutorial

Convolution

1D and 2D Convolutions

- One-dimensional:

$$y(t) = \int_{-\infty}^{+\infty} x(t-a)w(a)da$$

- Two-dimensional:

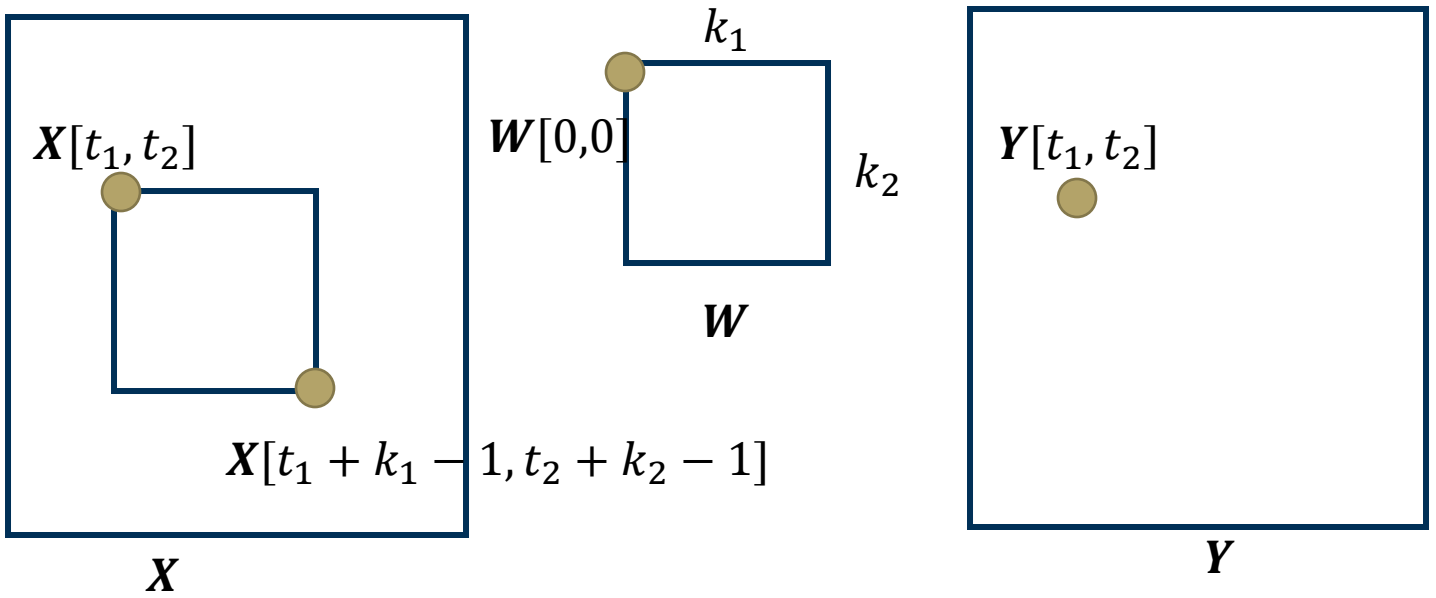
$$y(t_1, t_2) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} x(t_1 - a, t_2 - b)w(a, b)\, da\, db$$

Convolution

2D Discrete Convolution

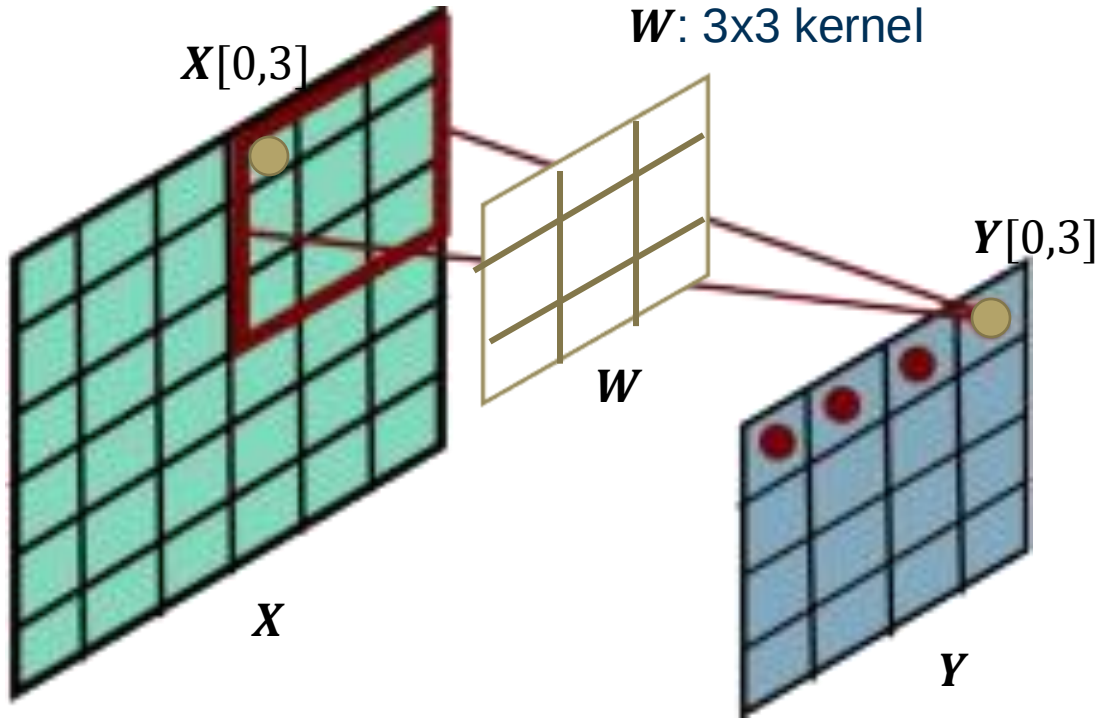
- Two-dimensional (discrete):

$$\boldsymbol{Y}[t_1, t_2] = \sum_{a=0}^{k_1-1} \sum_{b=0}^{k_2-1} \boldsymbol{X}[t_1 + a, t_2 + b]\boldsymbol{W}[a, b]$$



Convolutional Layer

2D Discrete Convolution

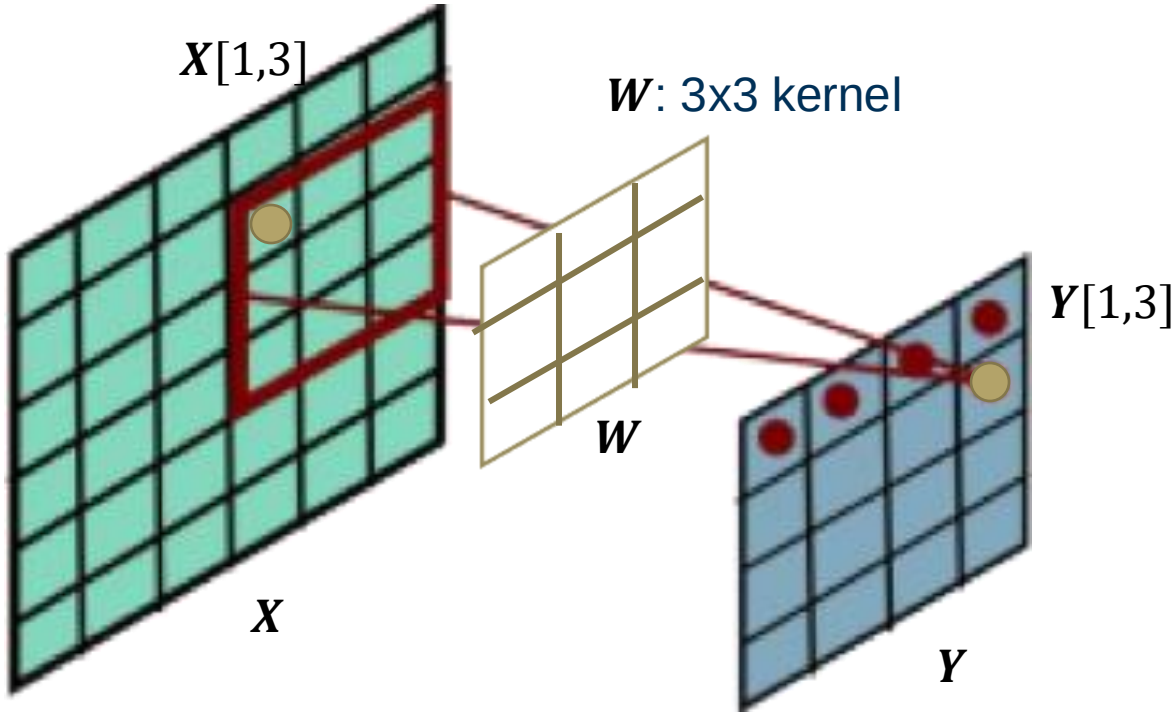


$$\boldsymbol{Y}[0,3] = \sigma\left(\sum_{a=0}^{3-1}\sum_{b=0}^{3-1} \boldsymbol{X}[0+a, 3+b]\boldsymbol{W}[a,b]\right)$$ (σ : non-linear activation)

Material adapted from Marc'Aurelio Ranzato, Deep Learning CVPR 2012 Tutorial

Convolutional Layer

2D Discrete Convolution

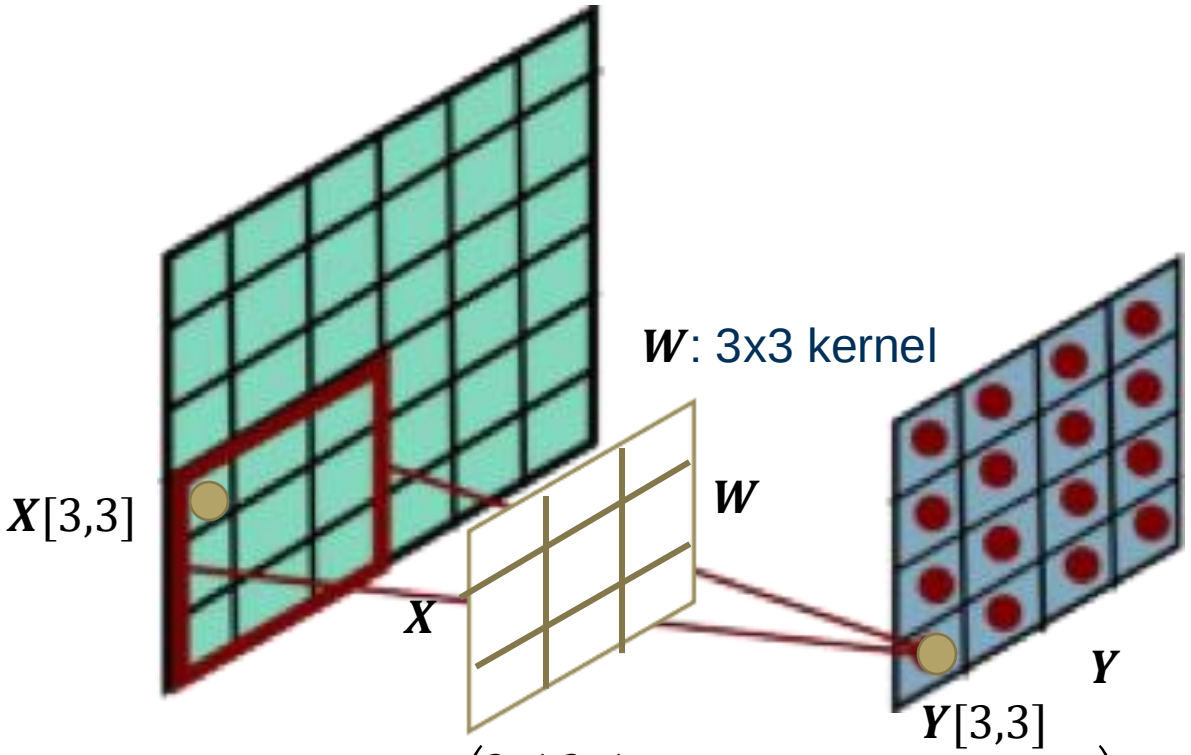


$$Y[1,3] = \sigma\left(\sum_{a=0}^{3-1}\sum_{b=0}^{3-1} X[1+a, 3+b]W[a,b]\right)$$ (σ : non-linear activation)

Material adapted from Marc'Aurelio Ranzato, Deep Learning CVPR 2012 Tutorial

Convolutional Layer

2D Discrete Convolution



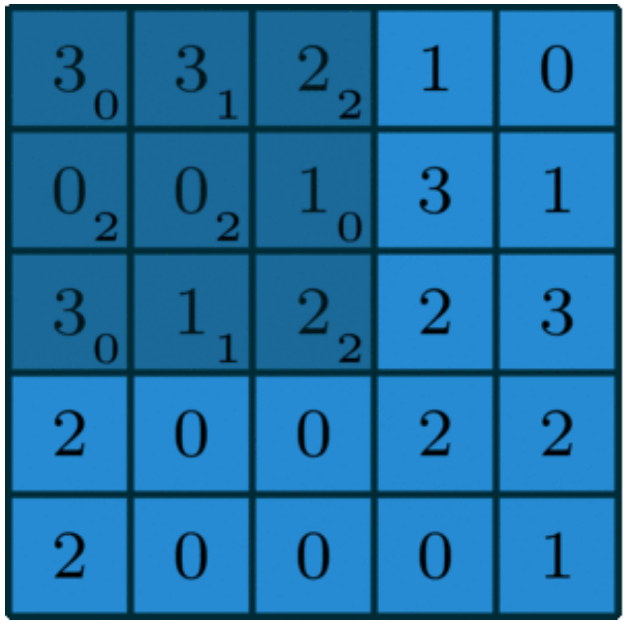
$$\boldsymbol{Y}[3,3] = \sigma\left(\sum_{a=0}^{3-1}\sum_{b=0}^{3-1} \boldsymbol{X}[3+a, 3+b]\boldsymbol{W}[a,b]\right)$$ (σ : non-linear activation)

Material adapted from Marc'Aurelio Ranzato, Deep Learning CVPR 2012 Tutorial

Convolutional Layer

Kernel Intuition

A 3x3 kernel slides over the input features, performing an element-wise multiplication with the current patch of the input, summing up the results into a single output at every location it slides over.



input features (5×5 matrix)

output features (3×3 matrix)

Kernel: $\begin{bmatrix} 0 & 1 & 2 \\ 2 & 2 & 0 \\ 0 & 1 & 2 \end{bmatrix}$

Convolutional Layer

Kernel

The Convolutional layer is the core building block of a Convolutional Network that does most of the computational heavy lifting.

The properties of convolutional layer:

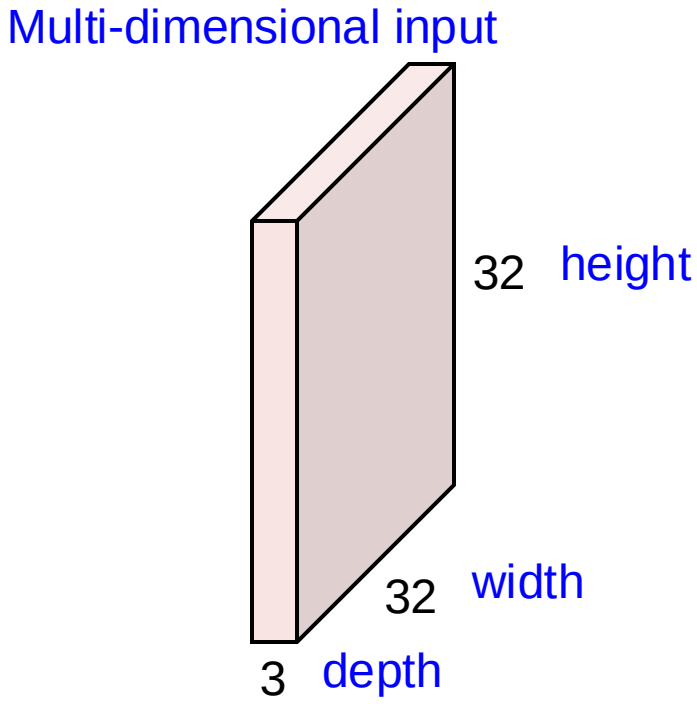
- Local Connectivity
- Parameter sharing
- Spatial arrangement

In this course: Kernels = Filters

Convolutional Layer

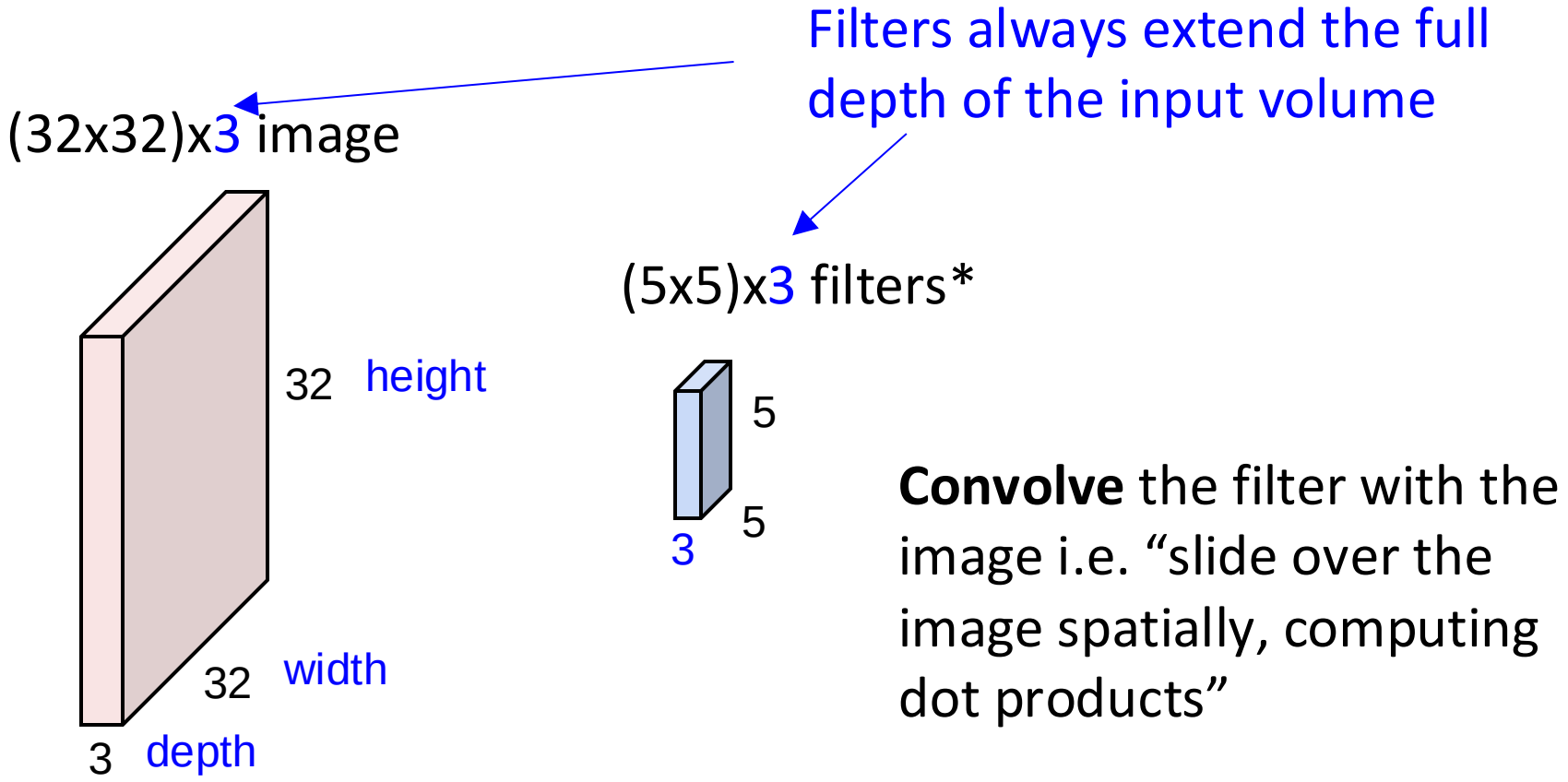
Spatial Arrangement

(32x32)x3 image → preserve spatial structure



Convolutional Layer

Spatial Arrangement



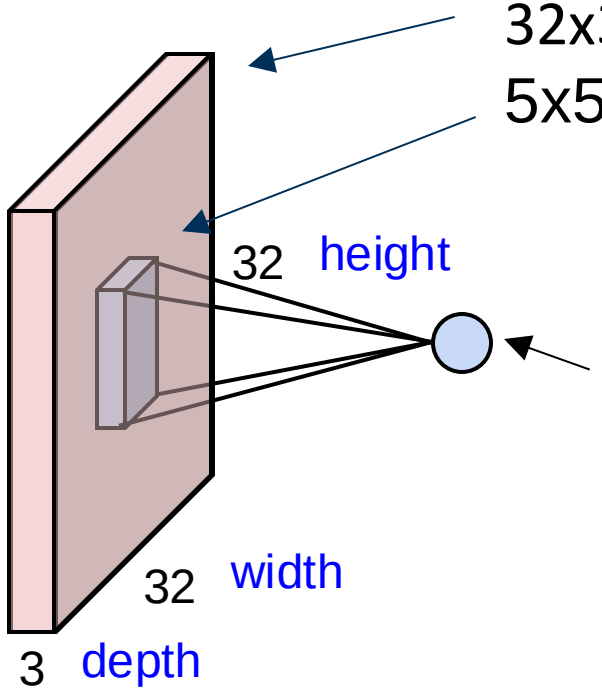
Convolve the filter with the image i.e. “slide over the image spatially, computing dot products”

* A collection of 3 unique 2-dimensional filters, of which dimension is 5x5

Convolutional Layer

Spatial Arrangement

consider the first filter

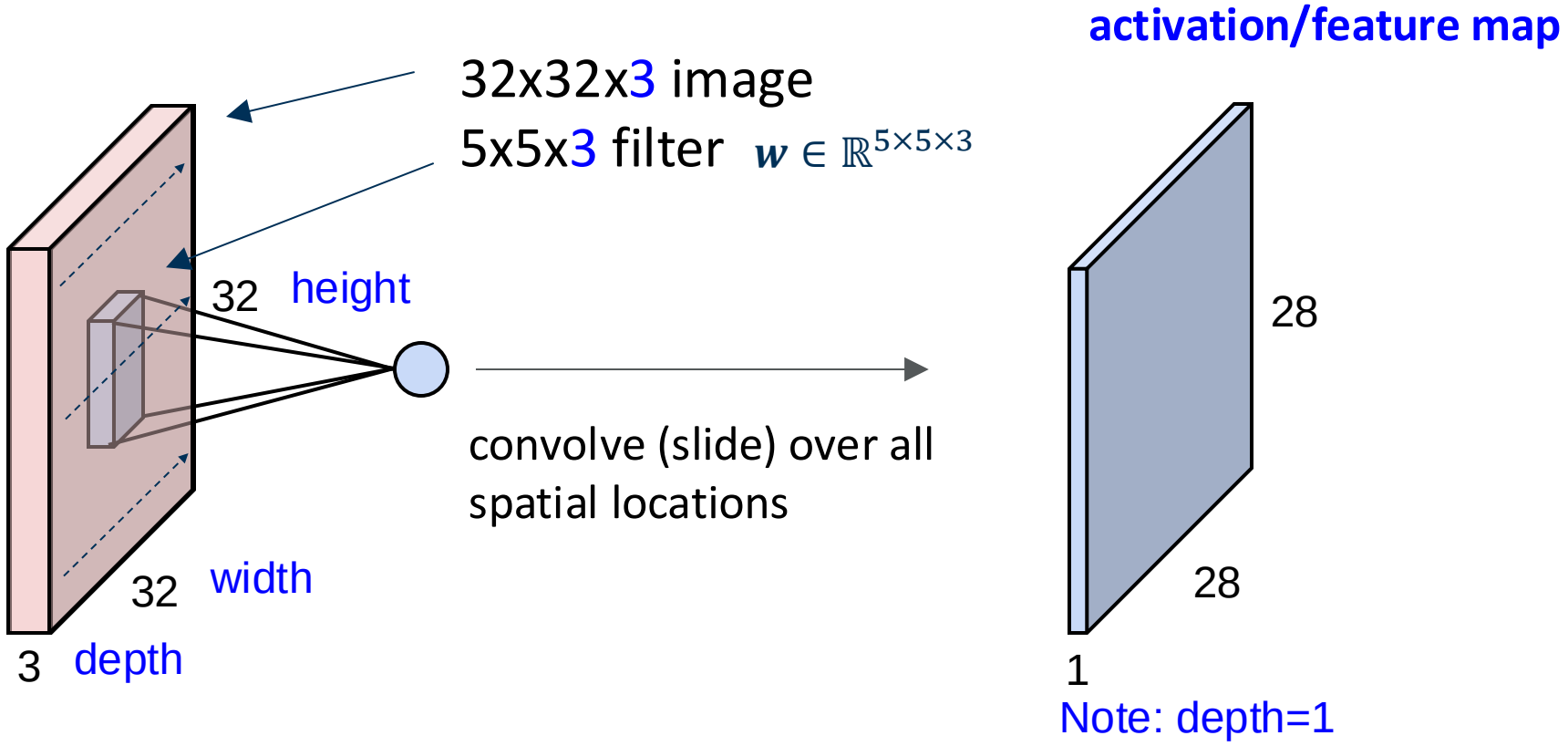


output 1 scalar number:
the result of taking a dot product between filter and a small 5x5x3 patch of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)

$w^T x + b$ (vectorized)
where $x \in \mathbb{R}^{5\times5\times3}$

Convolutional Layer

Spatial Arrangement

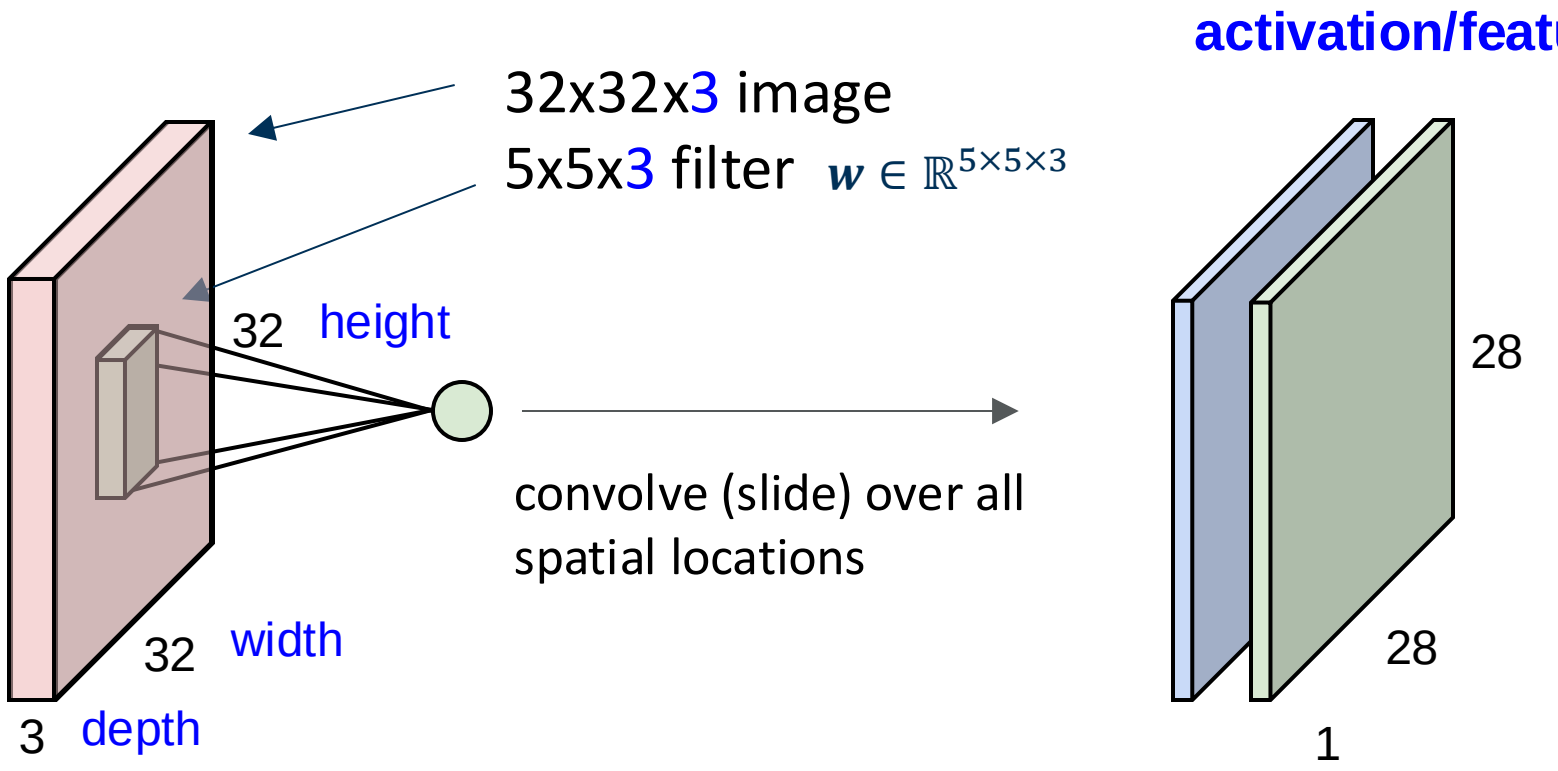


Material adapted from : Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Convolutional Layer

Spatial Arrangement

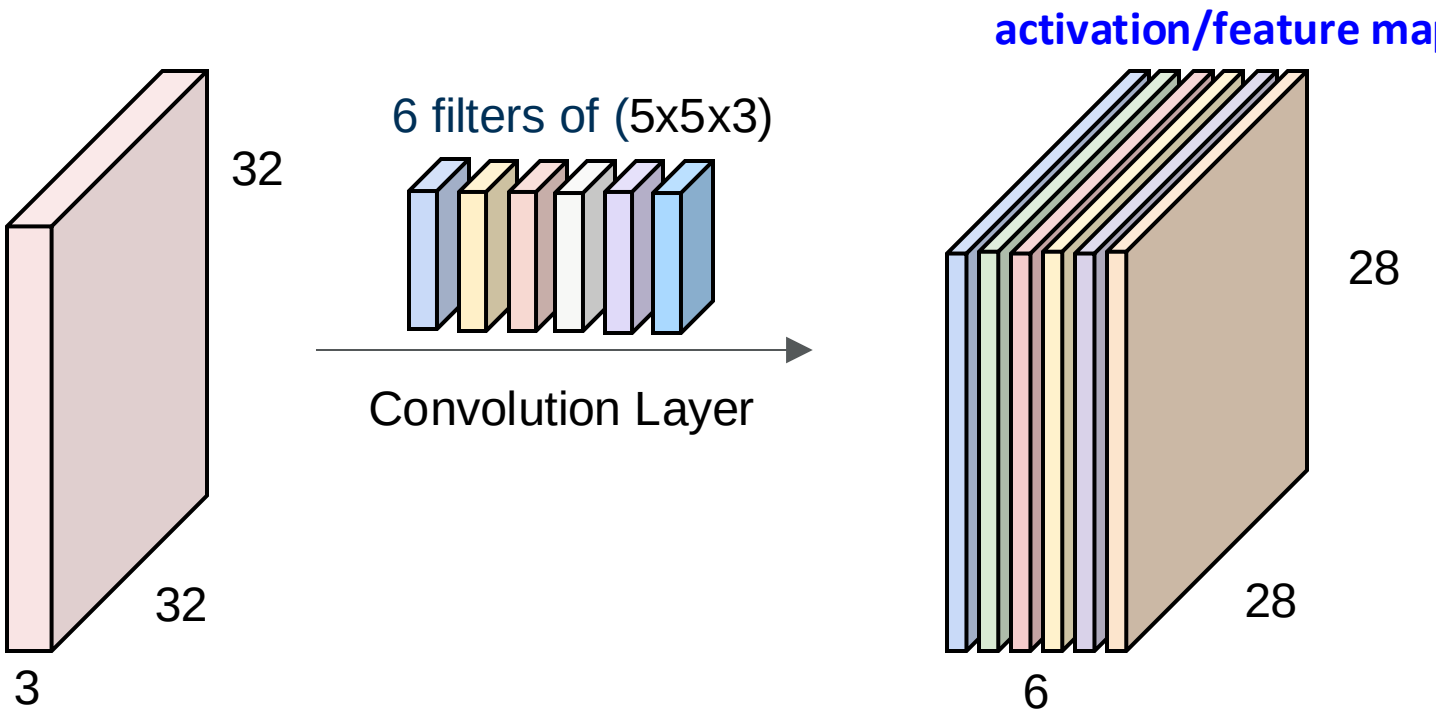
consider a second filter



Convolutional Layer

Spatial Arrangement

If we had 6 filters of size 5x5x3, the outputs are 6 separate activation maps:

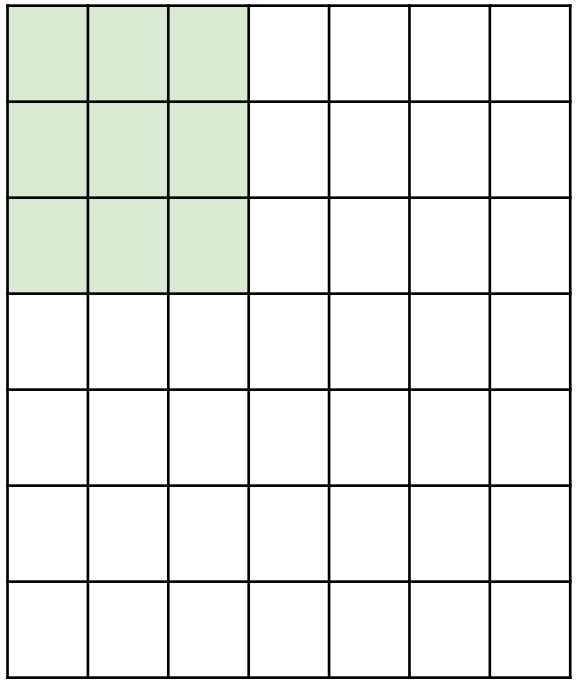


We stack these up to get a set of output feature maps of size 28x28x6!

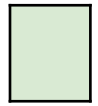
Convolutional Layer

Spatial Arrangement

A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter

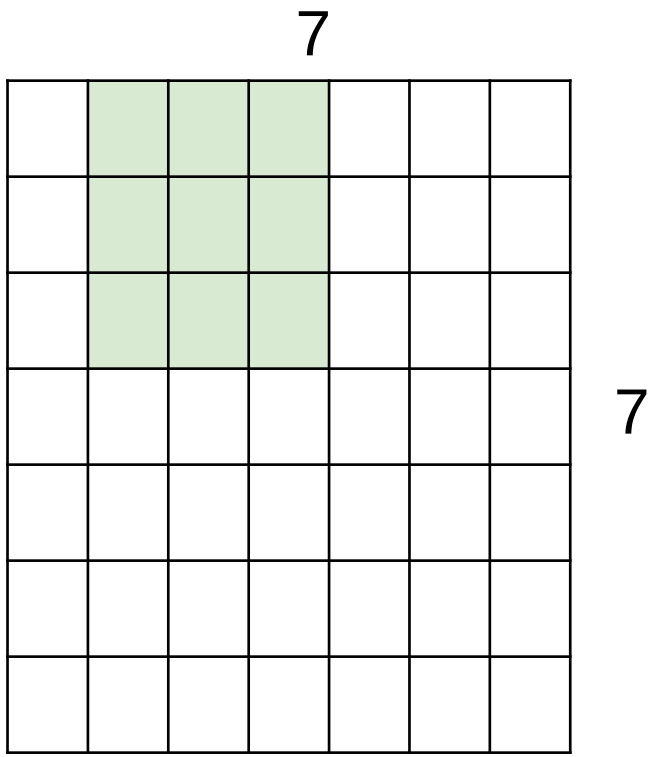


output

Convolutional Layer

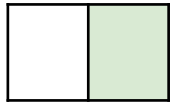
Spatial Arrangement

A closer look at spatial dimensions:



input

7x7 input (spatially)
assume 3x3 filter

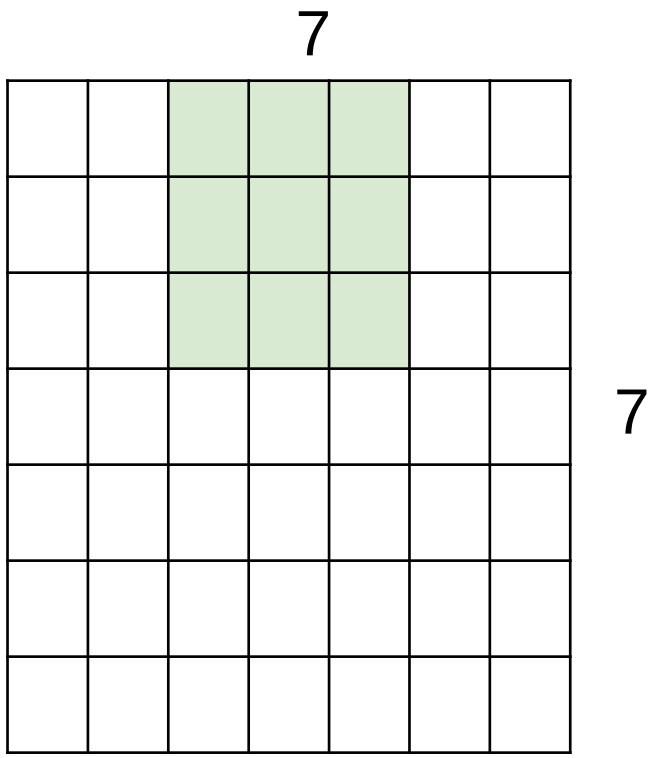


output

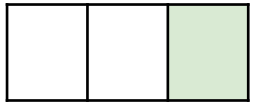
Convolutional Layer

Spatial Arrangement

A closer look at spatial dimensions:



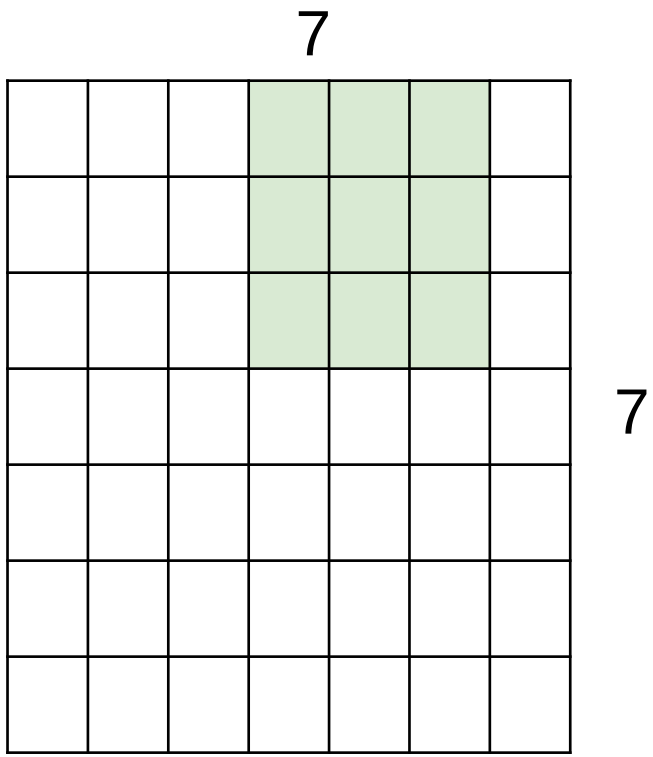
7x7 input (spatially)
assume 3x3 filter



Convolutional Layer

Spatial Arrangement

A closer look at spatial dimensions:

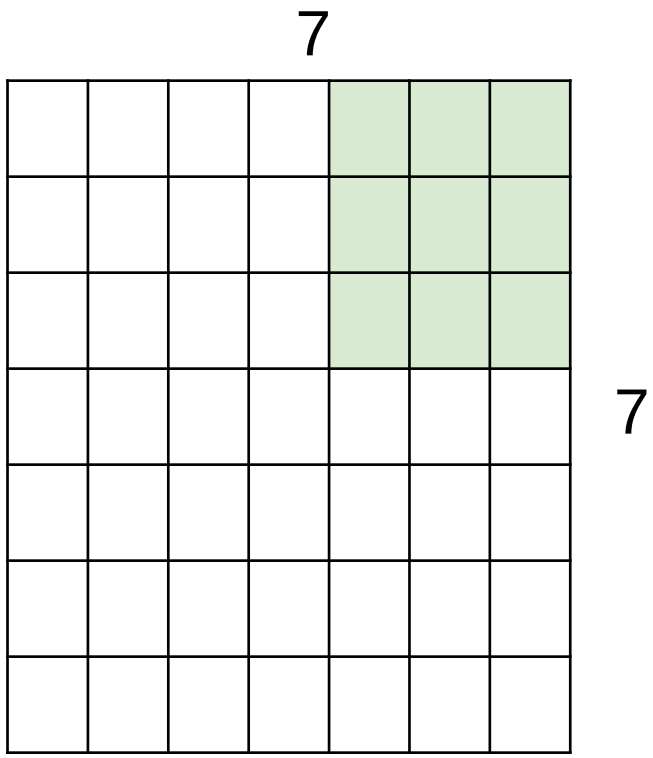


7x7 input (spatially)
assume 3x3 filter

Convolutional Layer

Spatial Arrangement

A closer look at spatial dimensions:



input

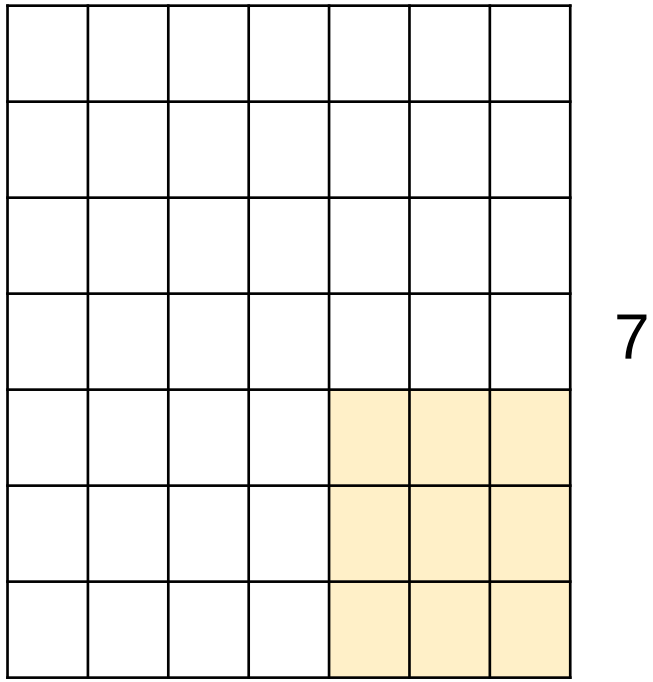
7x7 input (spatially)
assume 3x3 filter

output

Convolutional Layer

Spatial Arrangement

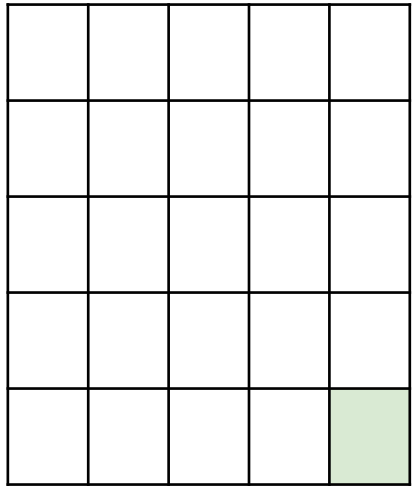
A closer look at spatial dimensions:



input

7x7 input (spatially)
assume 3x3 filter

=> 5x5 output

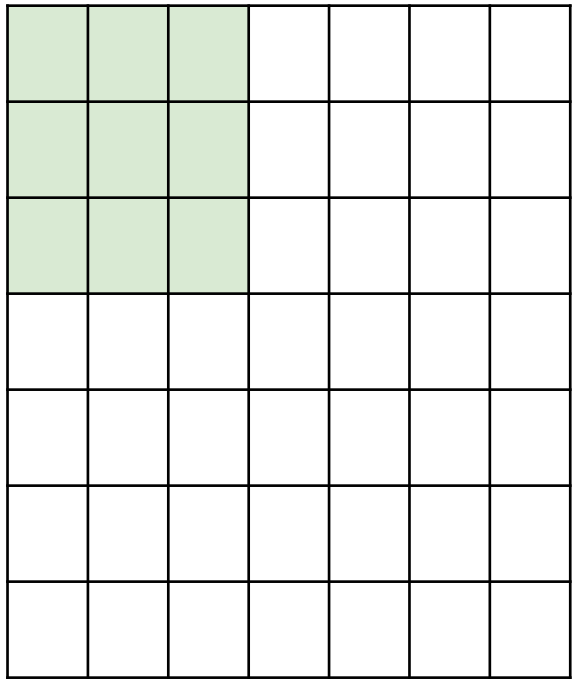


output

Convolutional Layer

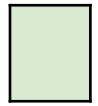
Spatial Arrangement

A closer look at spatial dimensions:



input

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

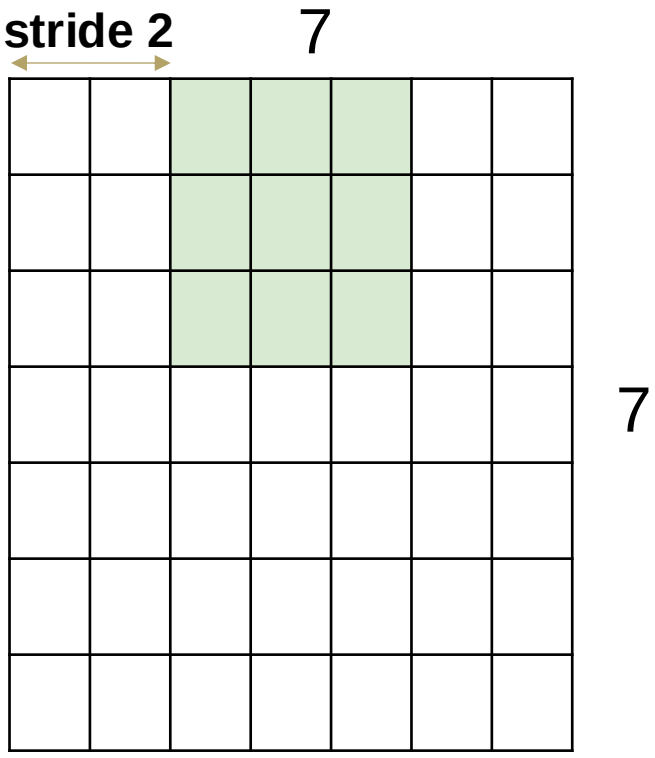


output

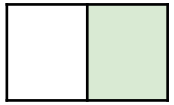
Convolutional Layer

Spatial Arrangement

A closer look at spatial dimensions:



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

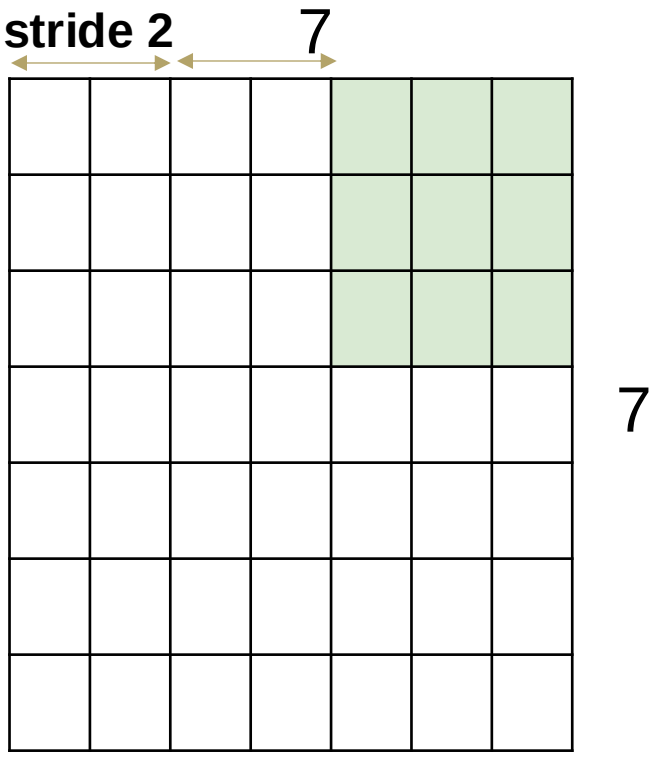


output

Convolutional Layer

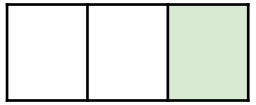
Spatial Arrangement

A closer look at spatial dimensions:



input

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

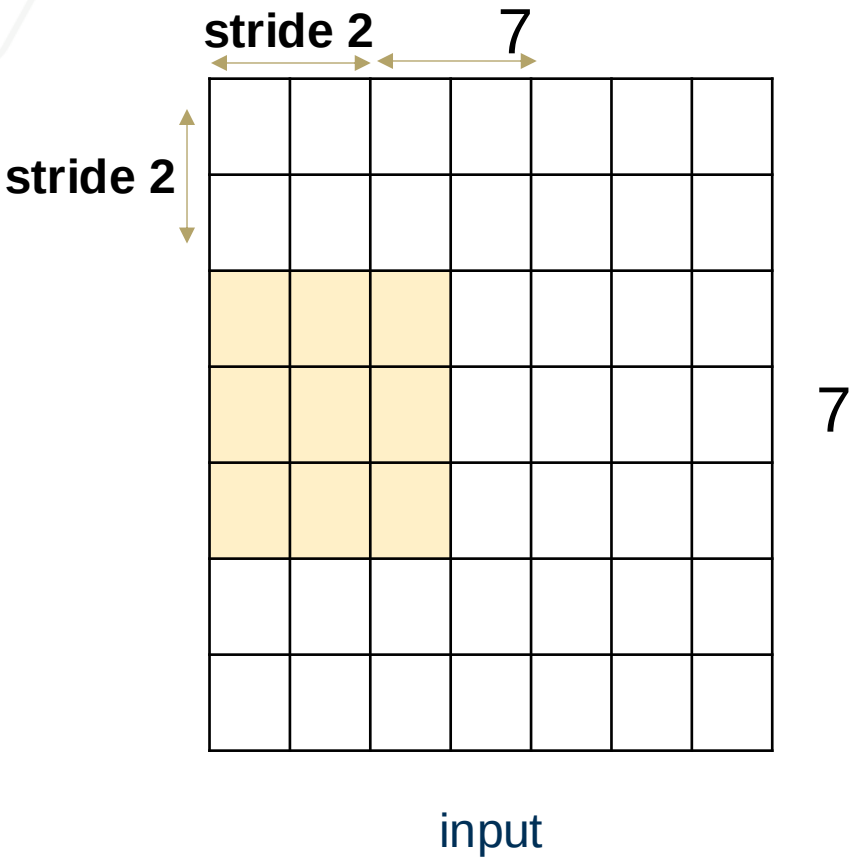


output

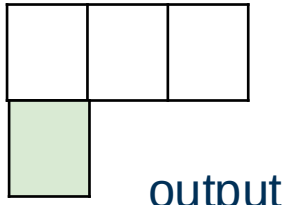
Convolutional Layer

Spatial Arrangement

A closer look at spatial dimensions:



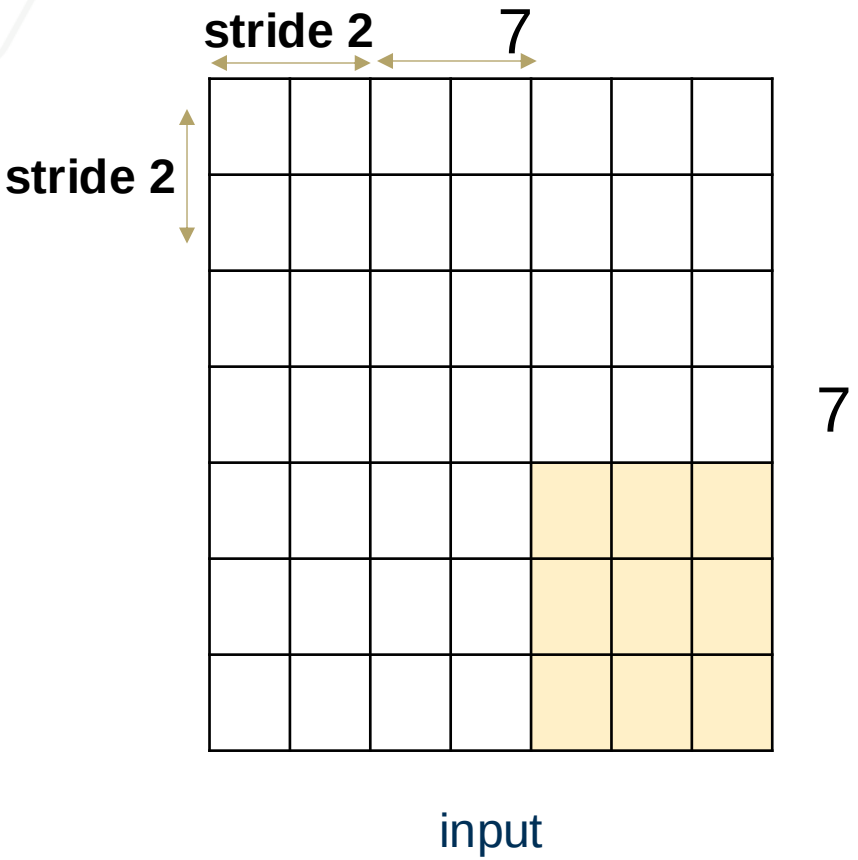
7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**



Convolutional Layer

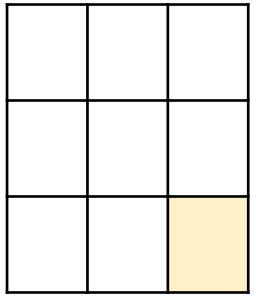
Spatial Arrangement

A closer look at spatial dimensions:



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

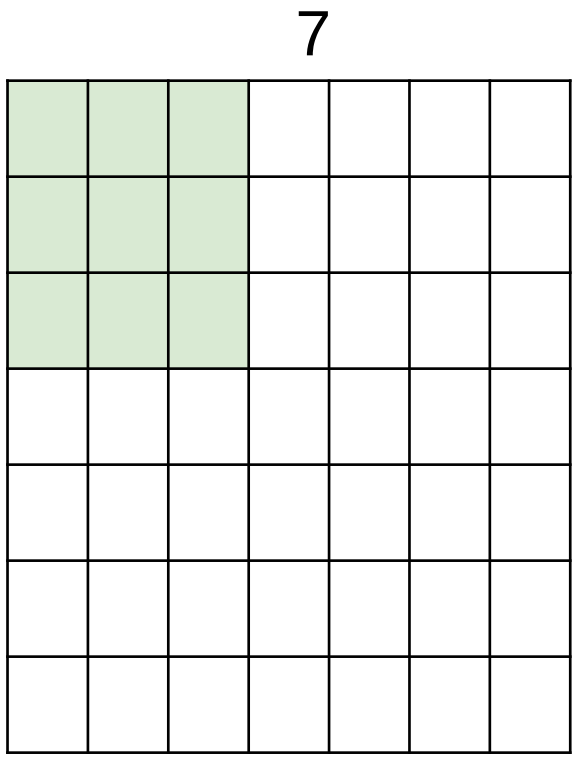
=> 3x3 output!



Convolutional Layer

Spatial Arrangement

A closer look at spatial dimensions:

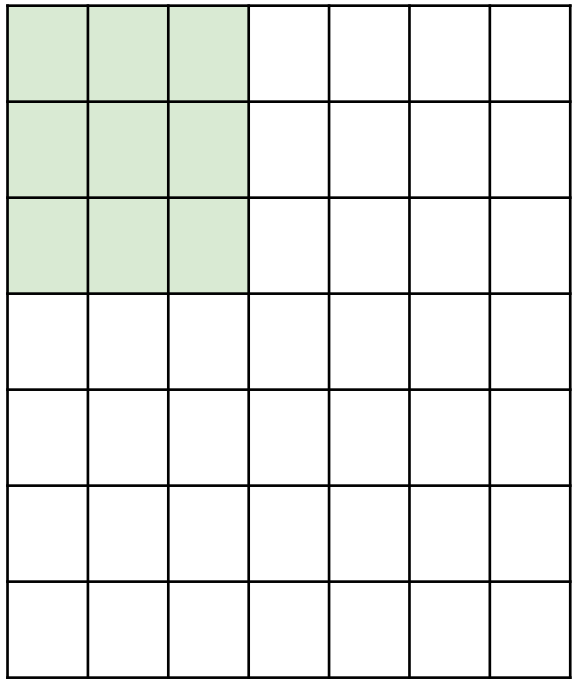


7x7 input (spatially)
assume 3x3 filter
applied **with stride 3?**

Convolutional Layer

Spatial Arrangement

A closer look at spatial dimensions:

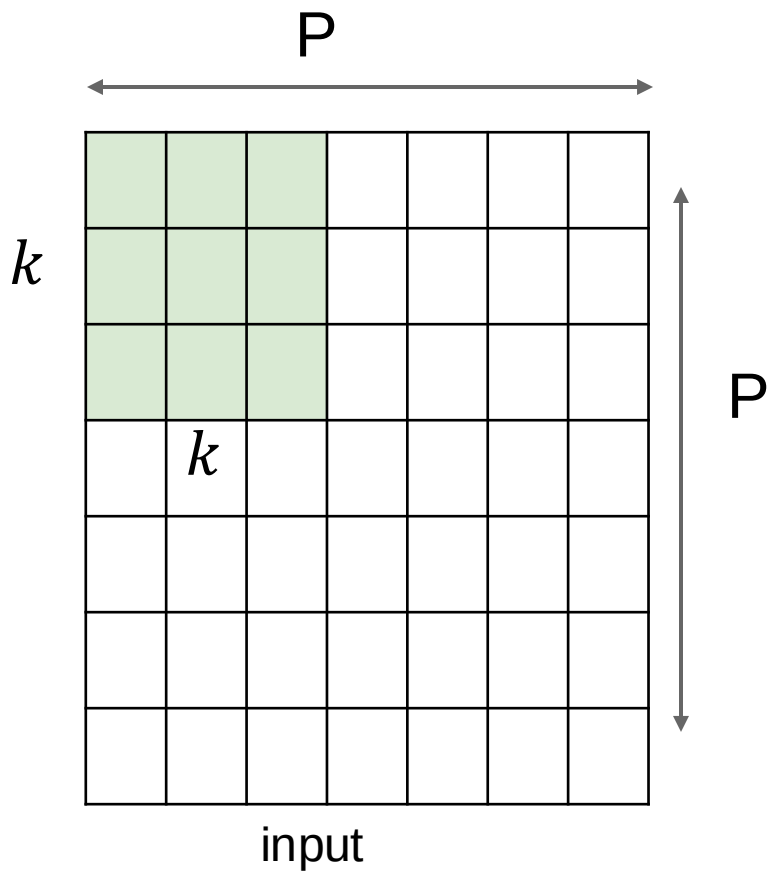


7x7 input (spatially)
assume 3x3 filter
applied **with stride 3?**

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

Convolutional Layer

Spatial Arrangement



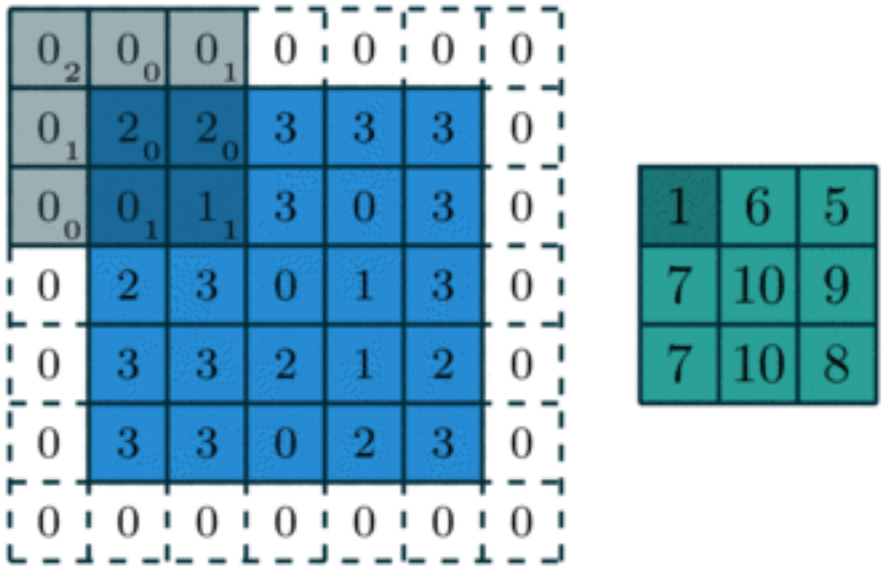
- Input size: P x P
- Filter (kernel) size: kx k
- Output size: **(P - k) / stride + 1**

e.g., P = 7, k = 3:
stride 1 => (7 - 3)/1 + 1 = 5
stride 2 => (7 - 3)/2 + 1 = 3
stride 3 => (7 - 3)/3 + 1 = 2.33

Convolutional Layer

Spatial Arrangement

zero-padding & strides



kernel: $\begin{pmatrix} 2 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \end{pmatrix}$

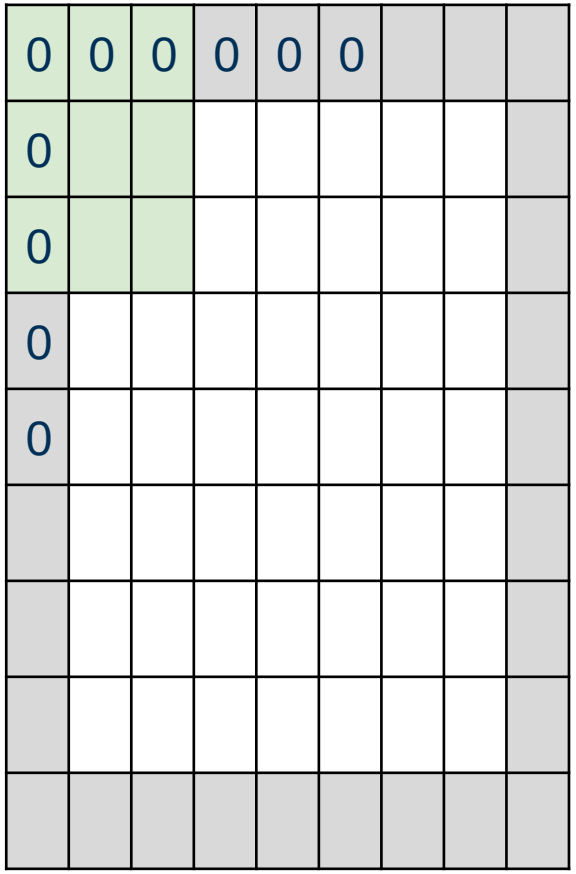
Padded with 1x1 border of zeros

stride 2: kernel moves 2 pixels at the input

Output size: (P - k) / stride + 1 = (7 - 3) / 2 + 1=3

Convolutional Layer

Spatial Arrangement



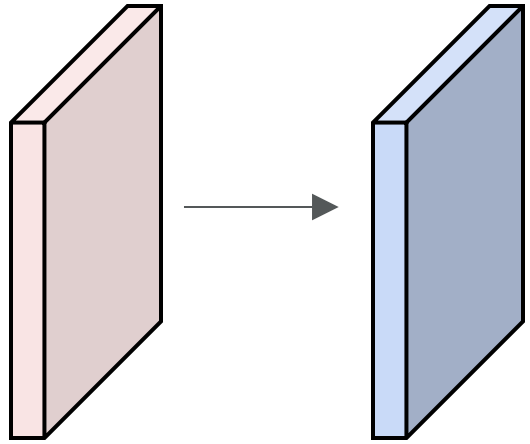
e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

7x7 output!
in general, if convolutional layers with stride 1, filters of size kxk, and zero-padding with (k-1)/2. (will preserve size spatially)
e.g. k = 3 => zero pad with 1
k = 5 => zero pad with 2
k = 7 => zero pad with 3

Convolutional Layer

Output Dimensions

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



- Output volume size:

(32+2*2-5)/1+1 = 32 spatially, so

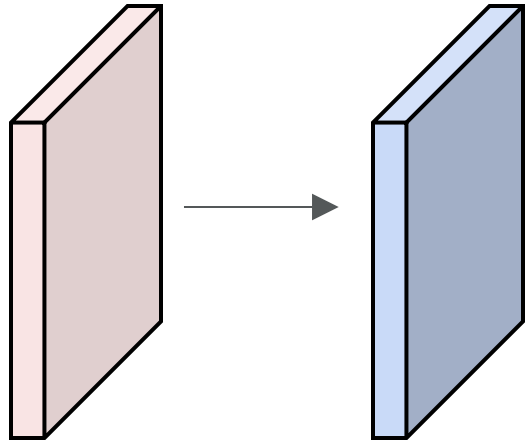
32x32x10

$$\frac{Input\ Size + 2 \times Padding\ - Filter\ size}{Stride} + 1$$

Convolutional Layer

Number of Parameters

- 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**

Convolutional Layer

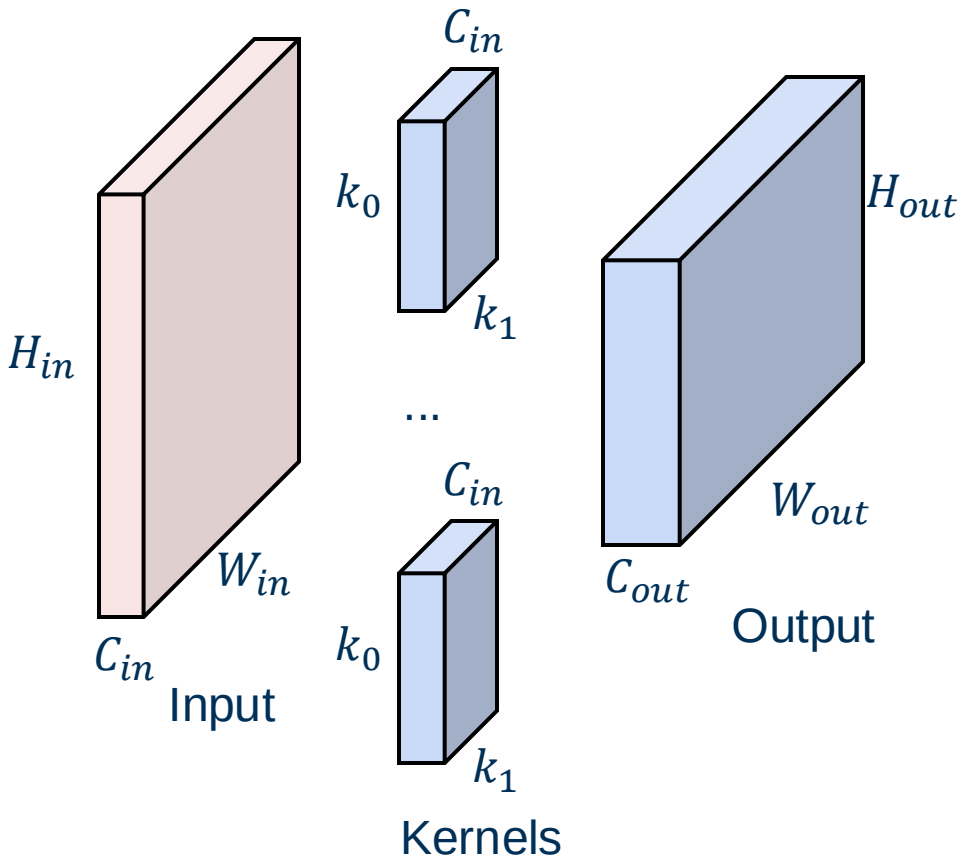
Dimensions

In general, for a convolutional layer:

- Input: $(N, C_{in}, H_{in}, W_{in})$
- Kernel: $(C_{out}, C_{in}, k_0, k_1)$
- Output: $(N, C_{out}, H_{out}, W_{out})$

where

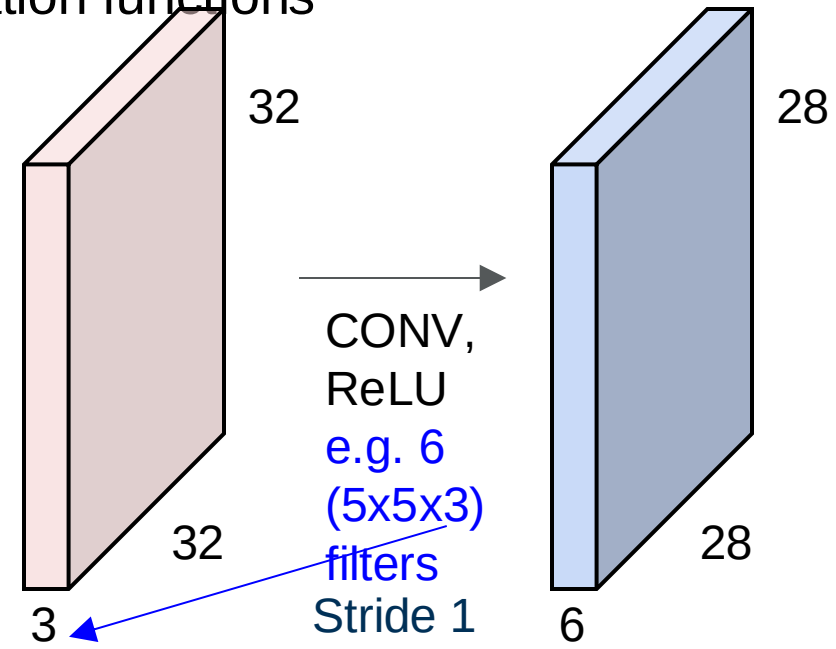
- $H_{out} = \left\lfloor \frac{H_{in} + 2 \times Padding_H - k_0}{Stride_H} + 1 \right\rfloor$
- $W_{out} = \left\lfloor \frac{W_{in} + 2 \times Padding_W - k_1}{Stride_W} + 1 \right\rfloor$
- N: number of samples
- $\lfloor \quad \rfloor$: floor operation, $\lfloor x \rfloor$ is the largest integer *i*, such that *i* <= *x*.



Convolutional Neural networks

Introduction

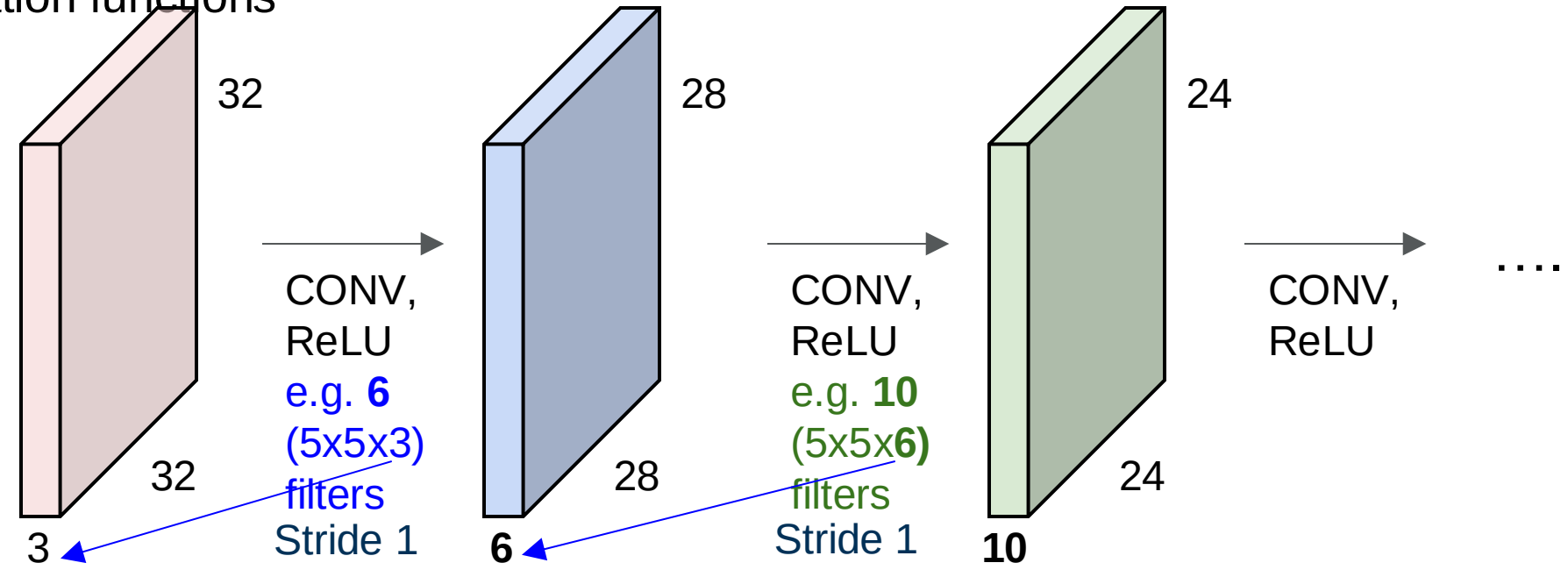
Preview: ConvNet is (initially) a sequence of Convolutional Layers, interspersed with activation functions



Convolutional Neural networks

Introduction

Preview: ConvNet is (initially) a sequence of Convolutional Layers, interspersed with activation functions

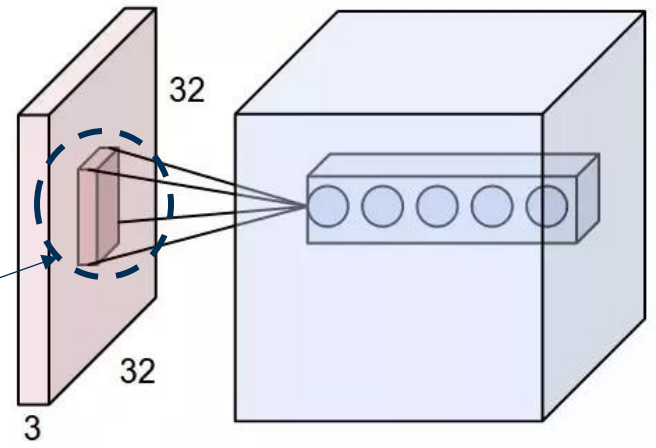


Convolutional Neural networks

Introduction

- Fully-connected networks:
  - Output of each neuron depends on the entire input
- Convolutional networks :
  - Output of a neuron only depends on a **local region** of the **input**. This region in the input is the **receptive field** for that neuron.

local receptive fields
(5x5 region in the input)

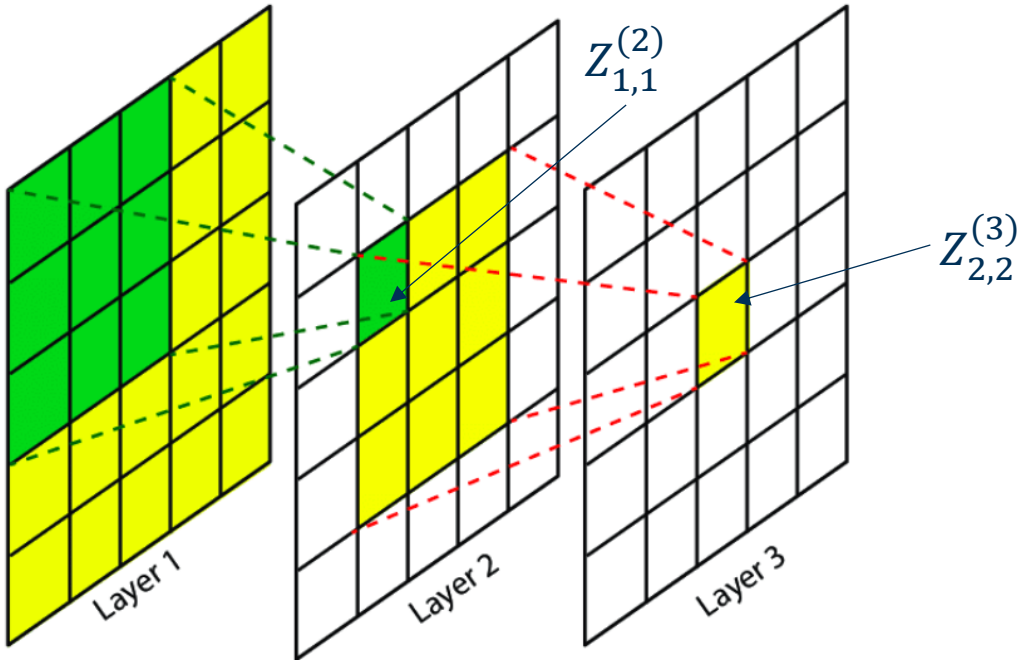


Luo, Wenjie, et al. "Understanding the effective receptive field in deep convolutional neural networks." *NeurIPS* 2016

Convolutional Neural networks

Introduction

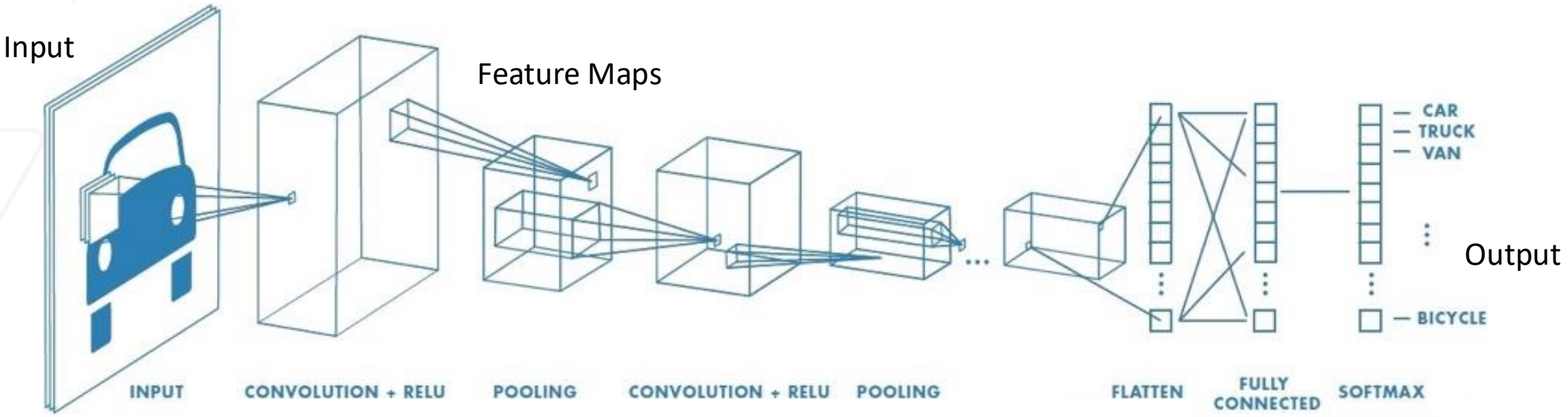
- The receptive field of $Z_{1,1}^{(2)}$ in the layer 1 has dimension of 3x3 (green region)
- The receptive field of $Z_{2,2}^{(3)}$ in the layer 1 has dimension of 5x5 (yellow region)



- Stacking more convolutional layers increases the receptive field size

Convolutional Neural networks

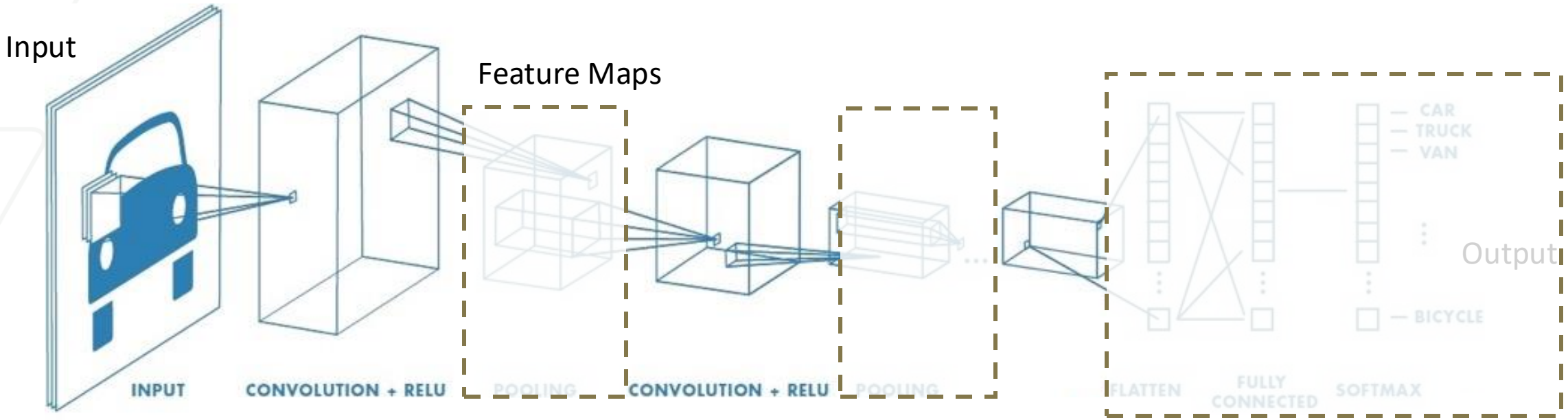
Introduction



different layers in a CNN

Convolutional Neural networks

Introduction



different layers in a CNN

Overview

In this Lecture..

Locally Connected Layer

- Limitations of ANNNs
- Convolutional Layers

Introduction to Convolutional Neural Networks

Layers used to build Convolutional Neural Networks

- Convolutional layer
- Pooling Layer
- Fully-connected layers

Examples of Deep CNNs Architectures

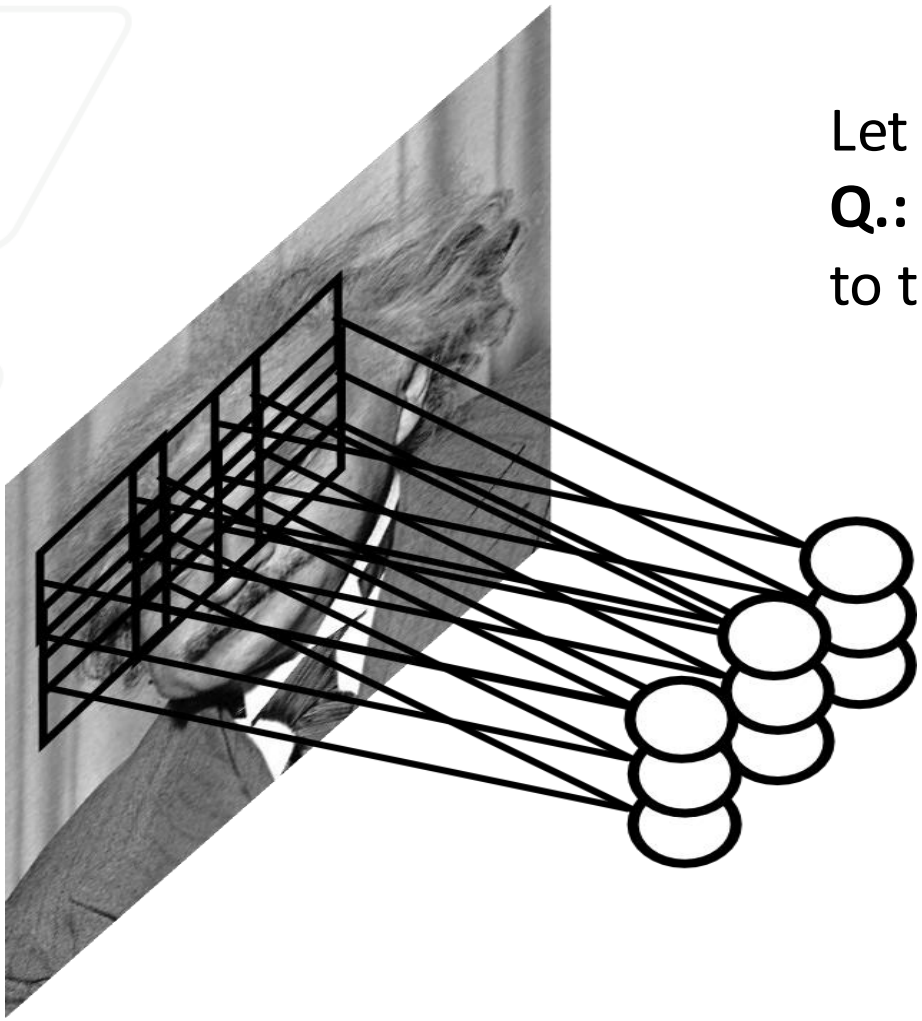
Training CNNs

- Optimization

Visualization of Convolutional Neural Networks

Convolutional Neural networks

Pooling Layer



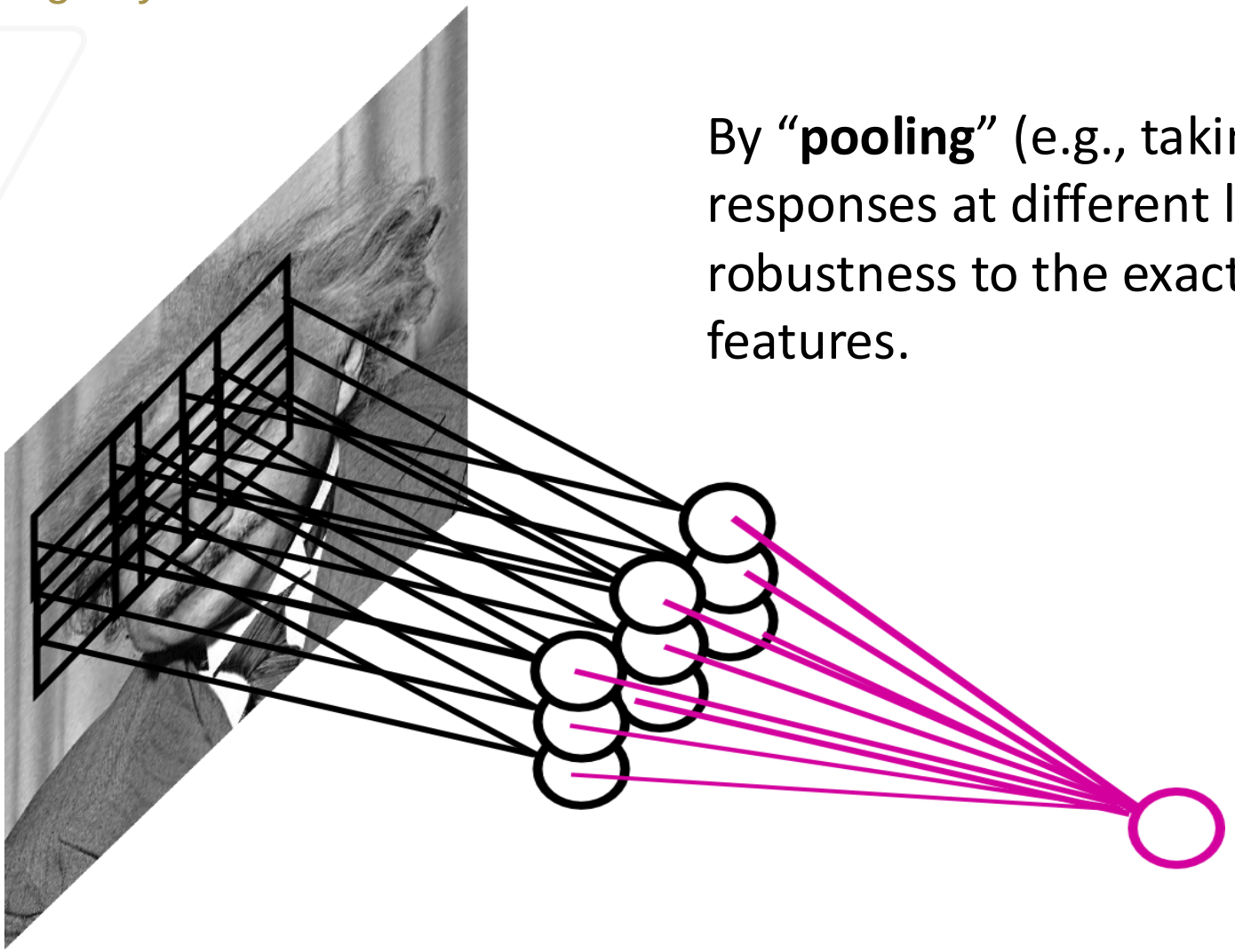
Let us assume the filter is an “eye” detector.

Q.: how can we make the detection robust to the exact location of the eye?

Material adapted from Marc'Aurelio Ranzato, Deep Learning CVPR 2012 Tutorial

Convolutional Neural networks

Pooling Layer



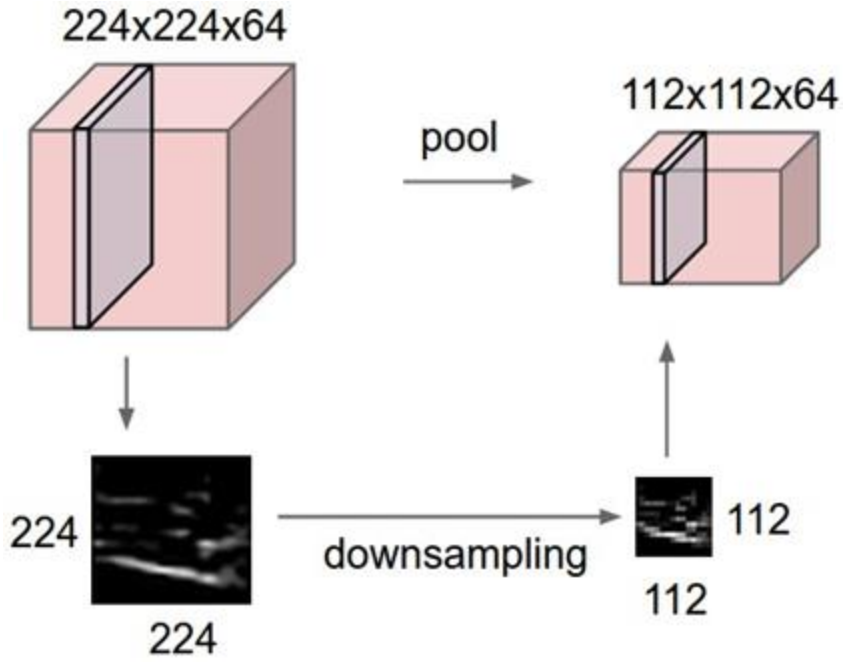
By “**pooling**” (e.g., taking max) filter responses at different locations we gain robustness to the exact spatial location of features.

Material adapted from Marc'Aurelio Ranzato, Deep Learning CVPR 2012 Tutorial

Convolutional Neural networks

Pooling Layer

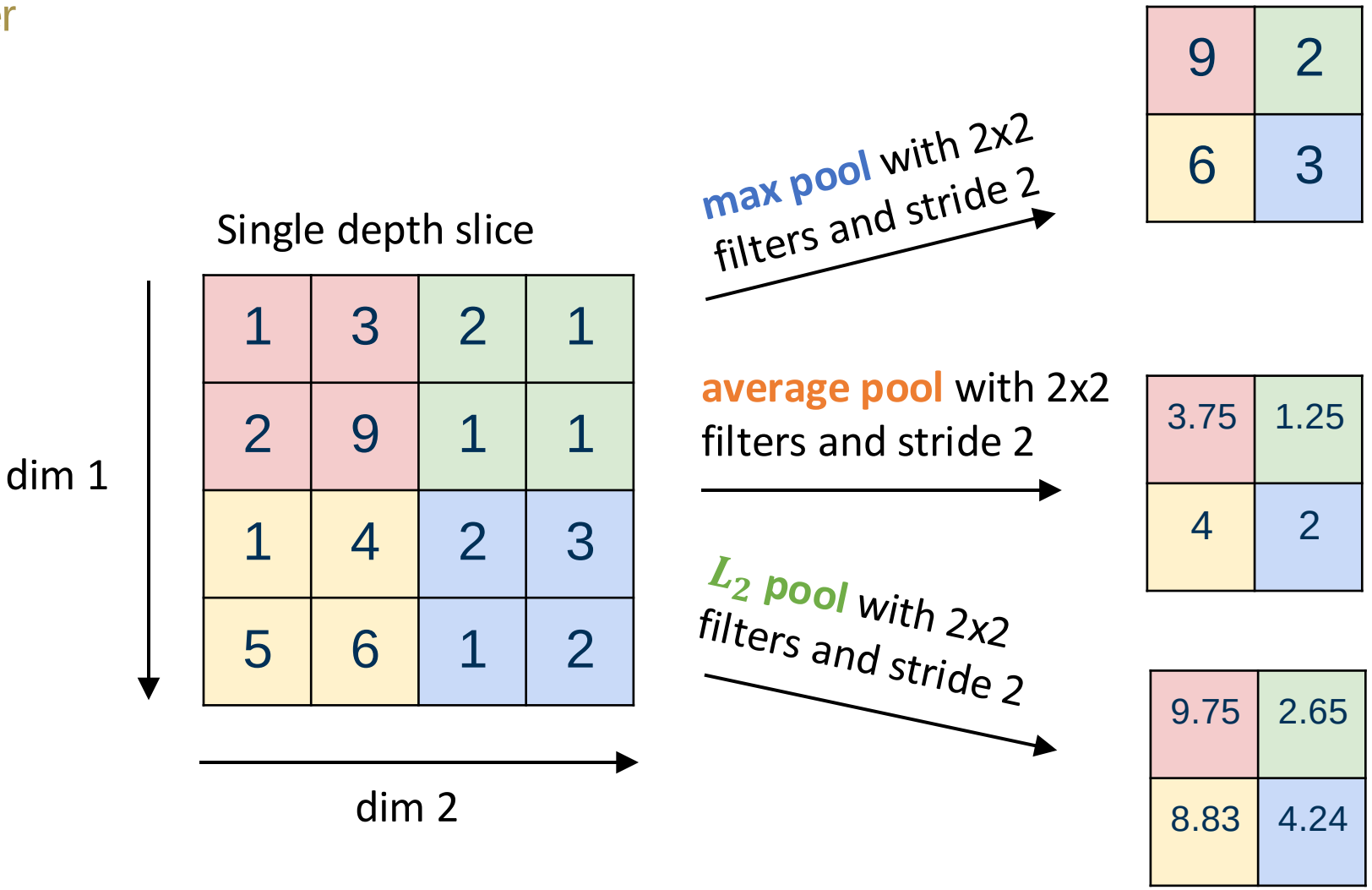
- makes the activation maps smaller and more manageable
- operates over each activation map independently



Material adapted from Marc'Aurelio Ranzato, Deep Learning CVPR 2012 Tutorial

Convolutional Neural networks

Pooling Layer

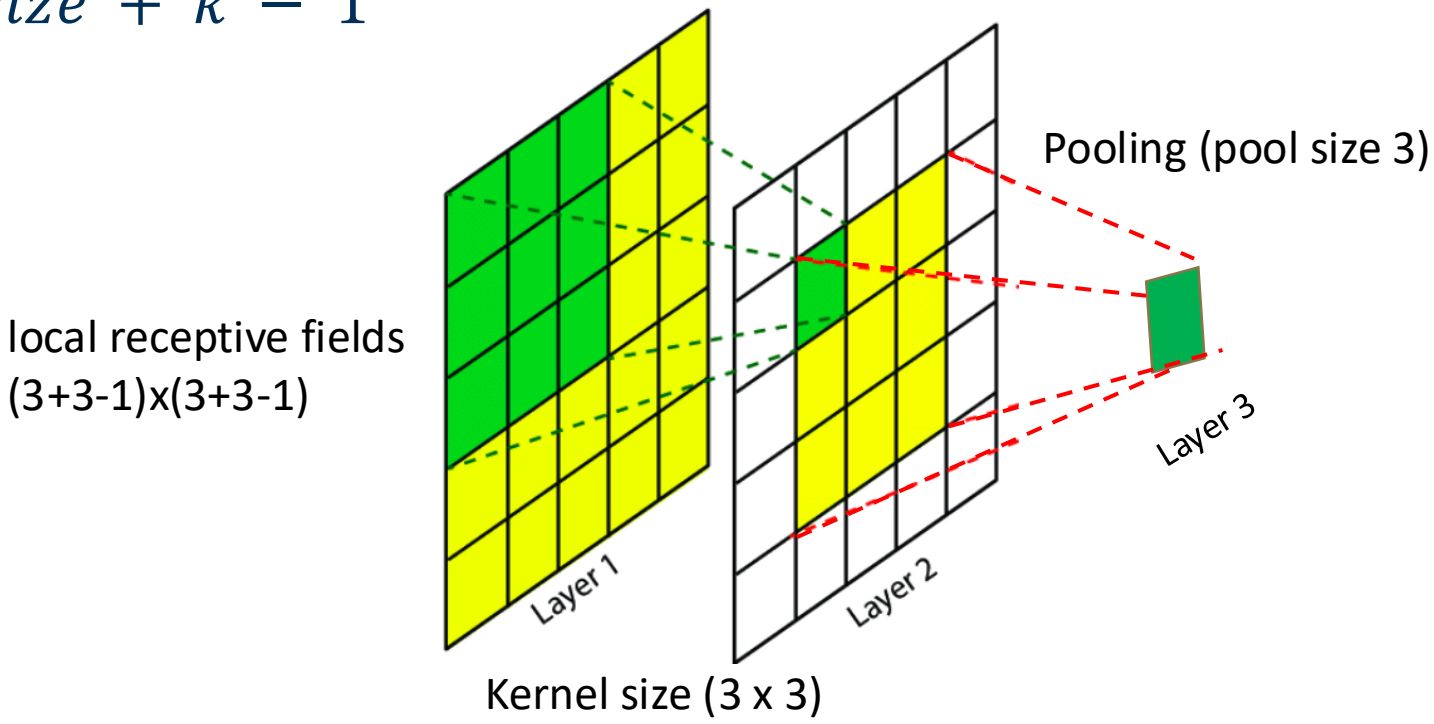


Material adapted from Marc'Aurelio Ranzato, Deep Learning CVPR 2012 Tutorial

Convolutional Neural networks

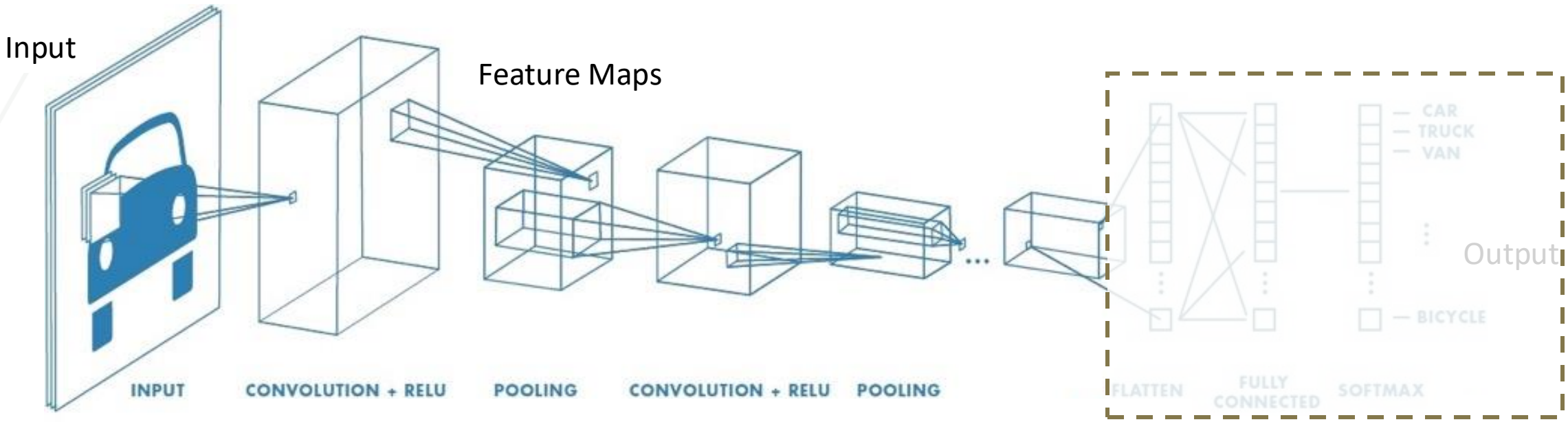
Pooling Layer: Receptive Field Size

- If filters are k x k and stride 1
- then each neuron in pooling layer 3 depends upon a patch in layer 1 of size: $pool\_size + k - 1$



Convolutional Neural networks

Introduction

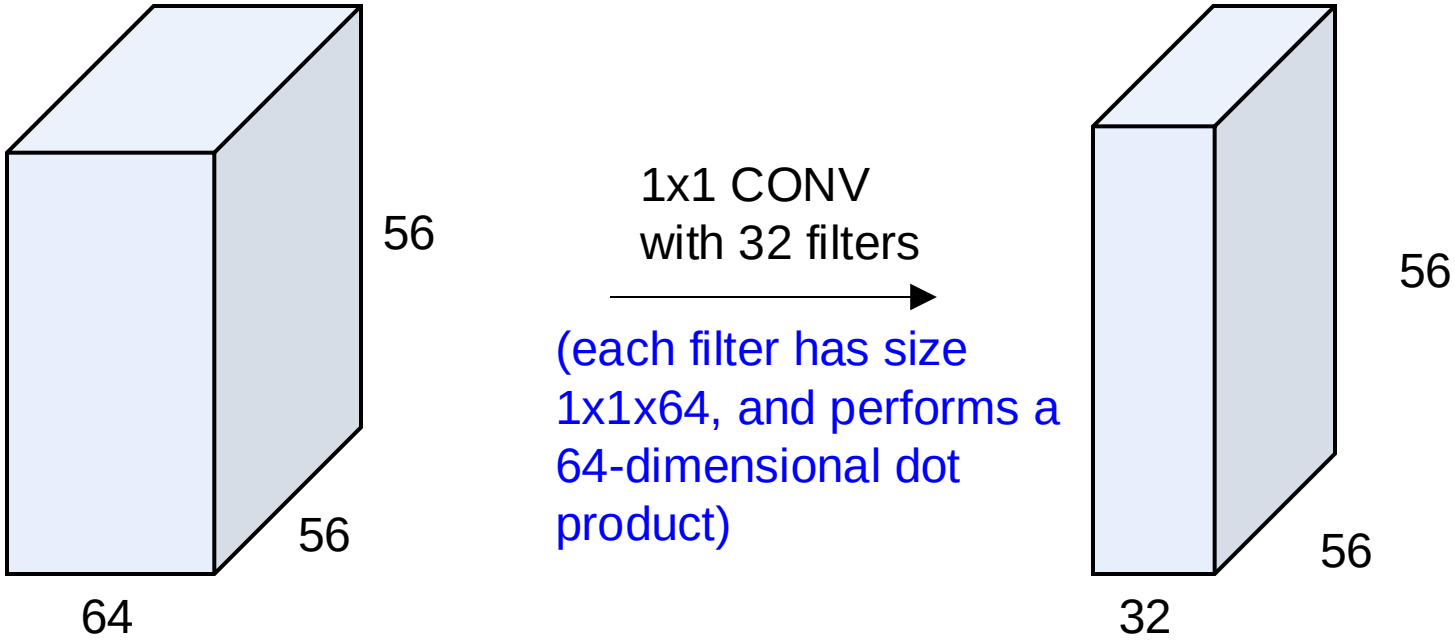


different layers in a CNN

Convolutional Neural networks

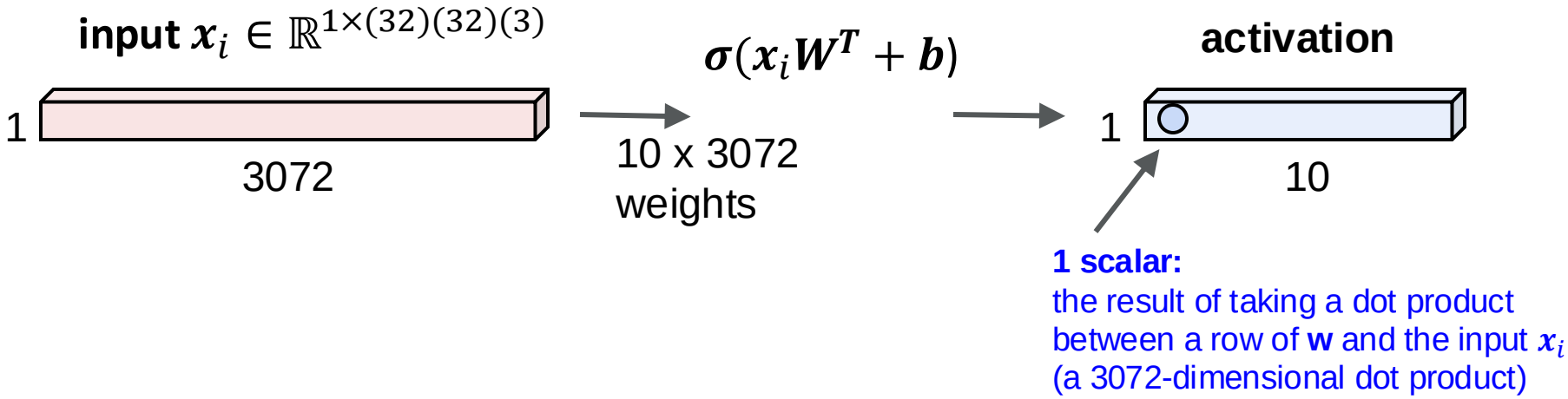
Fully-connected Layer: 1x1 Convolution

1x1 convolution layers make perfect sense



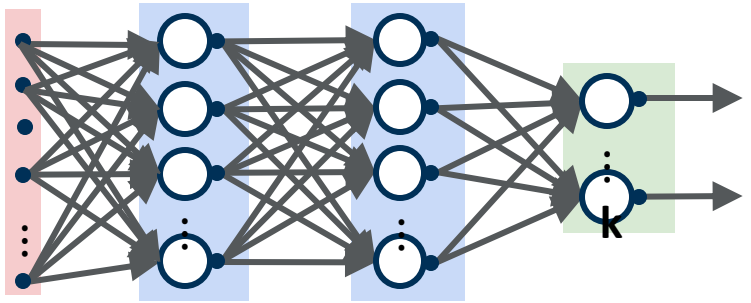
Fully-Connected Layer as 1x1 Convolution

32x32x3 image -> stretch to 3072 x 1

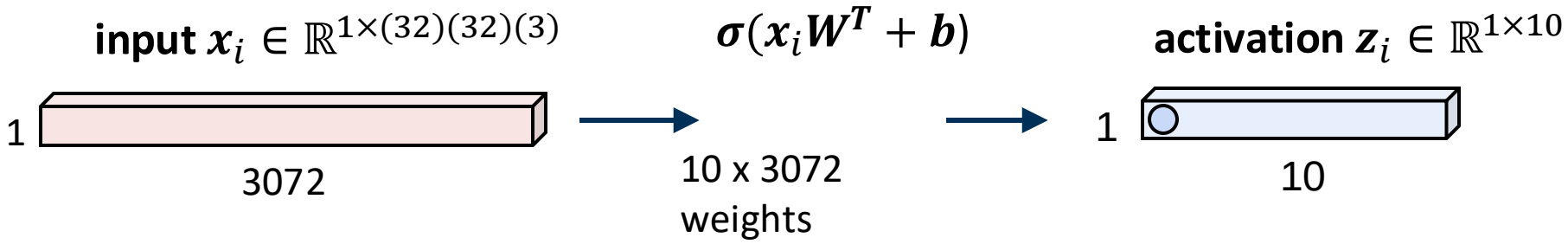


Convolutional Neural networks

Fully-connected Layer: 1x1 Convolution



Neurons in a fully connected layer have full connections to all activations in the previous layer, as seen in Artificial Neural Networks. Their activations can hence be computed with a matrix multiplication followed by a bias offset.



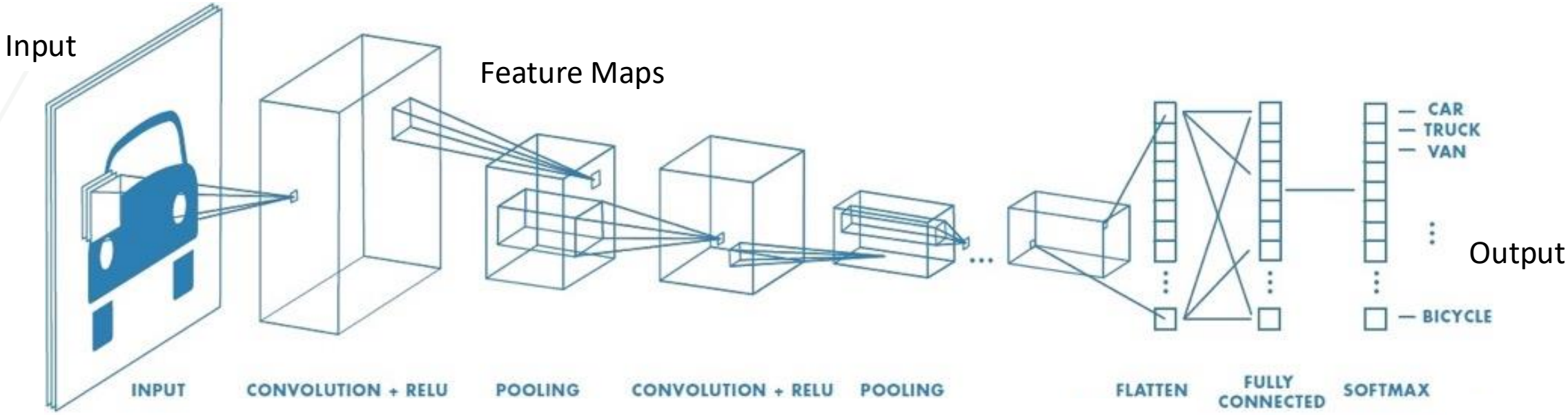
$$W^T \in \mathbb{R}^{(32)(32)(3)\times 10},\ b \in \mathbb{R}^{1\times 10}$$

Convolutional Neural networks

Fully-connected Layer: 1x1 Convolution

Three main types of layers to build ConvNet architectures:

- Convolutional Layer, Pooling Layer, and Fully-Connected Layer (ANN)



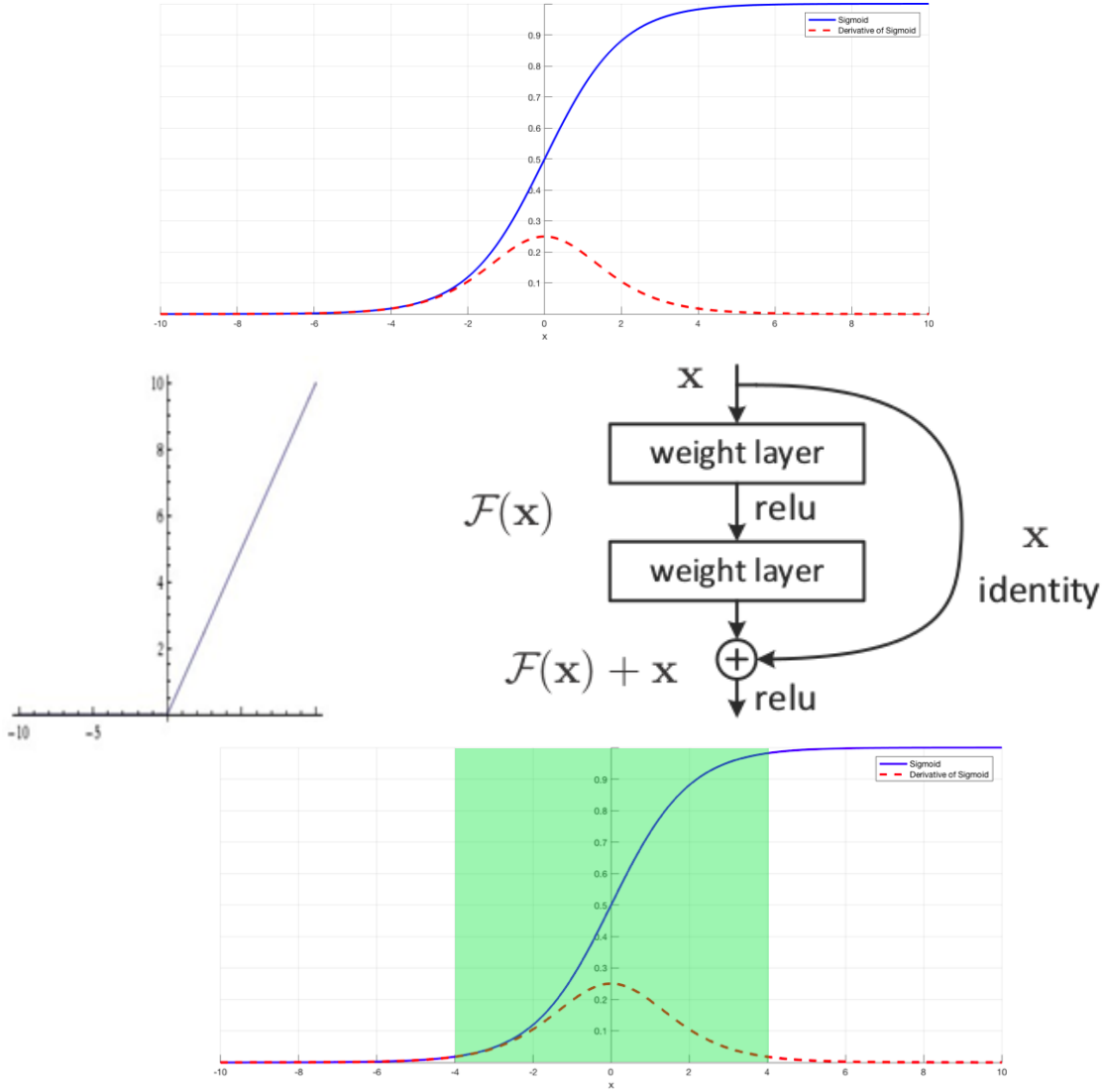
different layers in a CNN

Convolutional Neural networks

Vanishing Gradient Problem

- Using the sigmoid, the gradient could become smaller and smaller and in deep networks, that may cause the training to be ineffective as no change could take place [check the derivative of the sigmoid in the plot]

- Possible solutions:
  - Use ReLU
  - Use Residual networks
  - Normalization



Terminology

- $Distribution$: (sample space) the set of all possible samples
- $Dataset$: a set of samples drawn from a distribution
- $Batch$: a subset of samples drawn from the dataset
- $Sample$: a single data object represented as a set of features
- $Feature$: value of a single attribute, property, in a sample. Could be numeric or categorical.

Appendix A: Notations

- x_i: a single feature
- $\boldsymbol{x}_i$: feature vector (a data sample)
- $\boldsymbol{x}_{:,i}$: feature vector of all data samples
- $\boldsymbol{X}$: matrix of feature vectors (dataset)
- N: number of data samples
- P: number of features in a feature vector
- α: learning rate
- Bold letter/symbol: vector
- Bold capital letters/symbol: matrix