# ICT303 – Advanced Machine Learning and Artificial Intelligence

## Topic 5: Convolutional Neural Networks (CNN) – Part I

Hamid Laga
H.Laga@murdoch.edu.au
Office: 245.1.020

# How to Get in Touch with the Teaching Team

- Internal and External Students
  - Email: H.Laga@murdoch.edu.au.
- Important
  - In any communication, please make sure that you
    - Start the subject of your email with ICT303
    - Include your student ID, name, and the lab slot in which you are enrolled.
  - We will do all our best to answer your queries within 24 hrs.

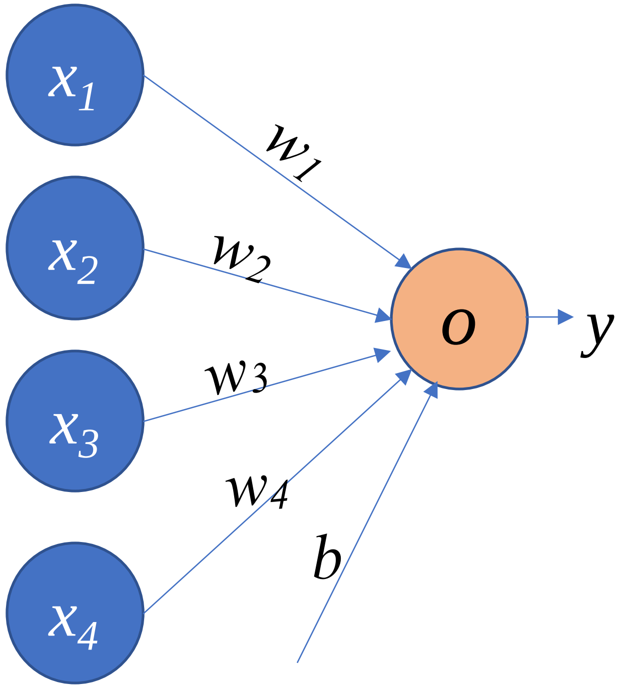
# In this Lecture

- Introduction
  - Recap of Linear regression, perceptron and MLP
  - Why we need CNNs?
- Convolutional Neural Network (CNN)
  - Convolutions
  - Convolution layers
- Examples of CNNs
  - LeNet
- Summary
- Learning objectives
  - Understand CNN and the different components that compose it
  - Implement CNNs in Python, NumPy and PyTorch
- Additional readings
  - Chapter 7 of the textbook, available at: https://d2l.ai/

# Linear Methods for Regression and Classification (Recap)

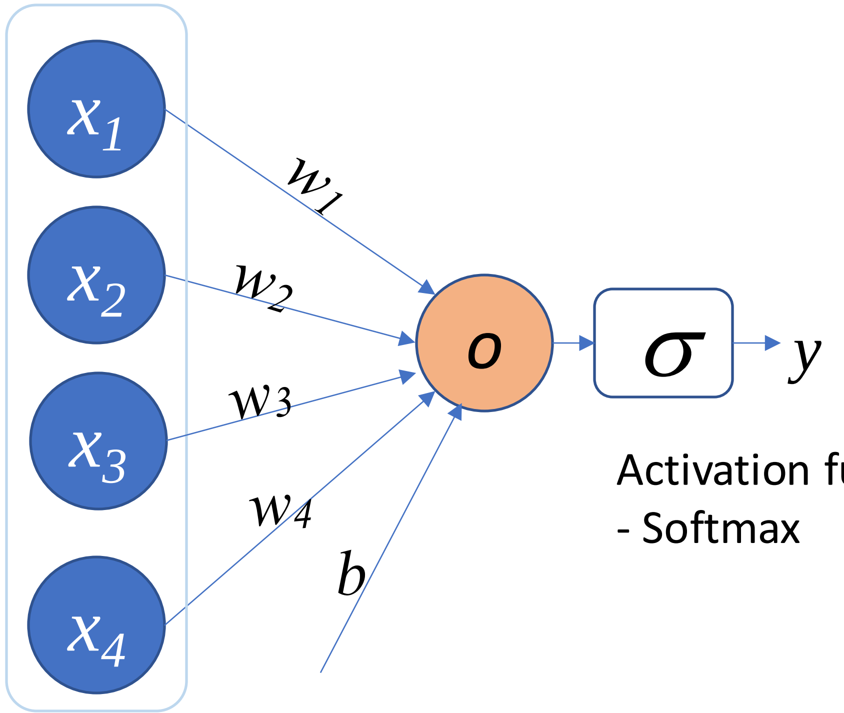
- Linear regression

A dense (fully connected, or linear) layer has parameters
$\mathbf{W} \in \mathbb{R}^{m \times n}, \mathbf{b} \in \mathbb{R}^m$, it computes output $\mathbf{y} = \mathbf{W}\mathbf{x} + \mathbf{b} \in \mathbb{R}^m$




$$o = y = \langle W, X \rangle + b$$

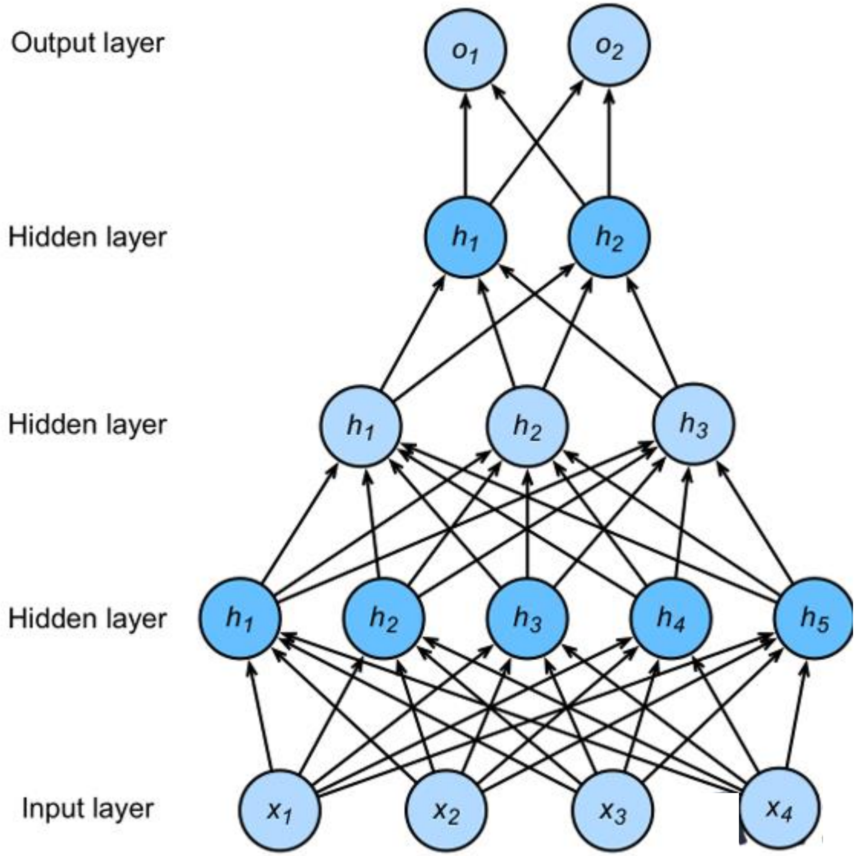
- Classification




Activation functions
- Softmax

# Multilayer Perceptron (MLP)

- Stacks multiple hidden layers (dense + activation) to get deeper models
- The activation function leads to non-linear models
  - Sigmoid, ReLU
- Hyper parameters
  - No. of hidden layers
  - No. of outputs of each hidden layers
- They are universal approximators
  - Can approximate any function



# MLPs limitations

- Design an MLP composed of one single hidden layer
  - Input: RGB image of size 300 x 300 pixels
  - Output: The class of the object depicted in the image
- Problem
  - Consider the case where we:
    - have 1000 possible object classes (horse, dogs, tables, ….)
    - use a simple MLP composed of 1 input layer, two hidden layers, and one output layer
  - How many parameters the network will have?
  - If we randomly shuffle the image pixels, what would be the output produced by the network?

# MLPs limitations

- Design an MLP composed of one single hidden layer
  - Input: RGB image of size 300 x 300 pixels
  - Output: The class of the object depicted in the image
- Problem
  - Consider the case where we:
    - have 1000 possible object classes (horse, dogs, tables, ….)
    - use a simple MLP composed of 1 input layer, two hidden layers, and one output layer
  - How many parameters the network will have?
  - If we randomly shuffle the image pixels, what would be the output produced by the network?

**For 1 Single Hidden Layer (for 1 training example):**

x = (300 x 300 x 3,1)
W = (1000,300x300x3)
b = (1000,1)
z = Wx + b

Total Parameters = 1000 x 300 x 300 x 3 + 1000 = 270 001 000

**For 2 Hidden Layers (needed for processing m training examples):**

x = (300 x 300 x 3,m)
W1 = (n_1,300x300x3)
b1 = (n_1,m)
W2 = (1000,n_1)
b2 = (1000,m)

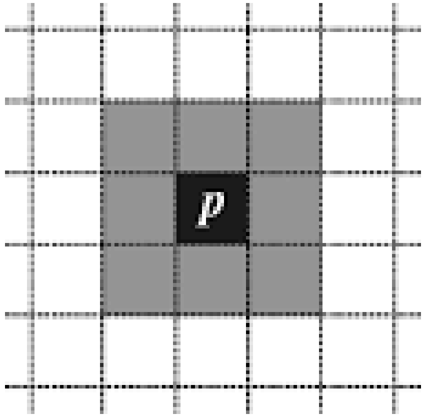
Total Parameters = n_1 x 300 x 300 x 3 + 1000 * n_1 + n_1 * m + 1000 * m

# From MLPS to Convolution Neural Networks (CNN)

- Issues with MLPs
  - Images are structured into grids of pixels
    - MLPs treat them as a flattened vector ignoring the local relations between pixels
    - If you take an image and randomly shuffle its pixels, the MLP will produce the same result!
  - Have a very large number of parameters
    - Thus, they are difficult to train for complex problems

- Convolutional Neural Networks (CNN or ConvNets)
  - A powerful family of neural networks specifically designed to
    - Capture the local spatial relationships between pixels
    - Reduce the number of parameters compared to MLPs
  - Uses convolutions

# Convolution



A pixel p and its
8 neighbors

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix},$$

A 3 x 3 filter

$$\begin{aligned} J(x,y) \; = \; & aI(x-1,y-1) + bI(x,y-1) + cI(x+1,y-1) + \\ & dI(x-1,y) + eI(x,y) + fI(x+1,y) + \\ & gI(x-1,y+1) + hI(x,y+1) + iI(x+1,y+1) \end{aligned}$$

# Convolution - Example

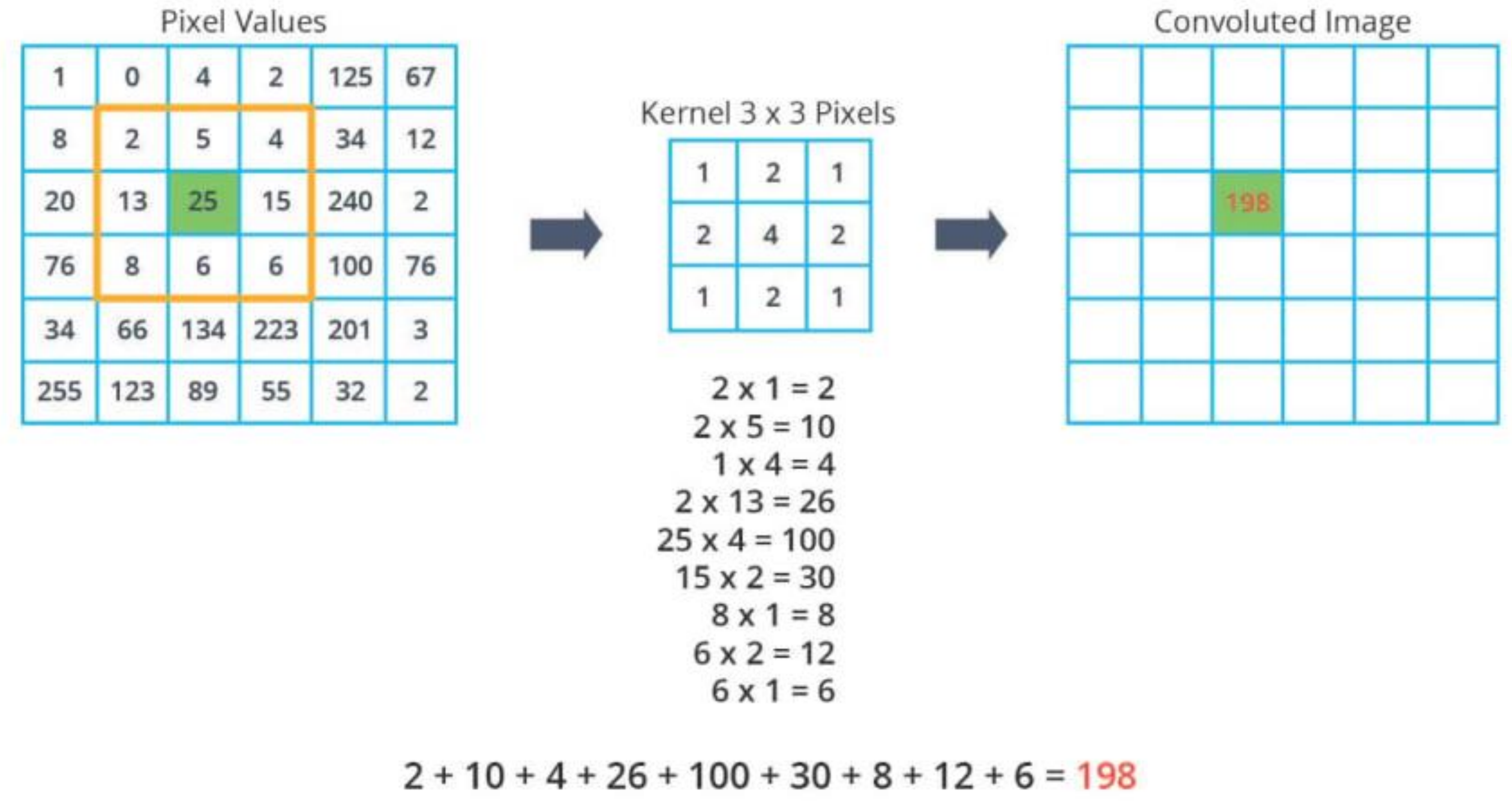



Image courtesy of: https://dev.to/sandeepbalachandran/machine-learning-convolution-with-color-images-2p41

# Illustration of Convolution

Given this 5x5 image, convolve with a 3x3 filter

| | | | | |
|---|---|---|---|---|
| 1 | 2 | 2 | 3 | 1 |
| 1 | 1 | 3 | 3 | 2 |
| 3 | 2 | 1 | 2 | 3 |
| 2 | 3 | 2 | 3 | 2 |
| 2 | 1 | 2 | 3 | 1 |

| | | |
|---|---|---|
| 1 | 2 | 1 |
| 1 | 2 | 1 |
| 1 | 2 | 1 |

# Illustration of Convolution

## Given this 5x5 image, convolve with a 3x3 filter

| | | | | |
|---|---|---|---|---|
| 1 | 2 | 1 | 3 | 1 |
| 1 | 2 | 1 | 3 | 2 |
| 1 | 2 | 1 | 2 | 3 |
| 2 | 3 | 2 | 3 | 2 |
| 2 | 1 | 2 | 3 | 1 |

Move the filter over the entire image from left to right, top to bottom

1x1 + 2x2 + 1x2 + 1x1 + 2x1 + 1x3 + 1x3 + 2x2 + 1x1

| | | |
|---|---|---|
| 21 | | |
| | | |
| | | |

| | | | | |
|---|---|---|---|---|
| 1 | 2 | 2 | 3 | 1 |
| 1 | 1 | 3 | 3 | 2 |
| 3 | 2 | 1 | 2 | 3 |
| 2 | 3 | 2 | 3 | 2 |
| 2 | 1 | 2 | 3 | 1 |

# Illustration of Convolution

## Given this 5x5 image, convolve with a 3x3 filter

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 2 | 1 | 1 |
| 1 | 1 | 2 | 1 | 2 |
| 3 | 1 | 2 | 1 | 3 |
| 2 | 3 | 2 | 3 | 2 |
| 2 | 1 | 2 | 3 | 1 |

Move the filter over the entire image from left to right, top to bottom

1x2 + 2x2 + 1x3 + 1x1 + 2x3 + 1x3 + 1x2 + 2x1 + 1x2

| | | | | |
|---|---|---|---|---|
| 1 | 2 | 2 | 3 | 1 |
| 1 | 1 | 3 | 3 | 2 |
| 3 | 2 | 1 | 2 | 3 |
| 2 | 3 | 2 | 3 | 2 |
| 2 | 1 | 2 | 3 | 1 |

| | | |
|---|---|---|
| 21 | 25 | |
| | | |
| | | |

# Illustration of Convolution

## Given this 5x5 image, convolve with a 3x3 filter

| | | | | |
|---|---|---|---|---|
| 1 | 2 | 1 | 2 | 1 |
| 1 | 1 | 1 | 2 | 1 |
| 3 | 2 | 1 | 2 | 1 |
| 2 | 3 | 2 | 3 | 2 |
| 2 | 1 | 2 | 3 | 1 |

Move the filter over the entire image from left to right, top to bottom

1x2 + 2x3 + 1x1 + 1x3 + 2x3 + 1x2 + 1x1 + 2x2 + 1x3

| | | |
|---|---|---|
| 21 | 25 | 27 |
| | | |
| | | |

| | | | | |
|---|---|---|---|---|
| 1 | 2 | 2 | 3 | 1 |
| 1 | 1 | 3 | 3 | 2 |
| 3 | 2 | 1 | 2 | 3 |
| 2 | 3 | 2 | 3 | 2 |
| 2 | 1 | 2 | 3 | 1 |

# Illustration of Convolution

## Given this 5x5 image, convolve with a 3x3 filter

| | | | | |
|---|---|---|---|---|
| 1 | 2 | 2 | 3 | 1 |
| 1 | 2 | 1 | 3 | 2 |
| 1 | 2 | 1 | 2 | 3 |
| 1 | 2 | 1 | 3 | 2 |
| 2 | 1 | 2 | 3 | 1 |

Move the filter over the entire image from left to right, top to bottom

1x1 + 2x1 + 1x3 + 1x3 + 2x2 + 1x1 + 1x2 + 2x3 + 1x2

| | | |
|---|---|---|
| 21 | 25 | 27 |
| 25 | | |
| | | |

| | | | | |
|---|---|---|---|---|
| 1 | 2 | 2 | 3 | 1 |
| 1 | 1 | 3 | 3 | 2 |
| 3 | 2 | 1 | 2 | 3 |
| 2 | 3 | 2 | 3 | 2 |
| 2 | 1 | 2 | 3 | 1 |

# Illustration of Convolution

## Given this 5x5 image, convolve with a 3x3 filter

| | | | | |
|---|---|---|---|---|
| 1 | 2 | 2 | 3 | 1 |
| 1 | 1 | 2 | 1 | 2 |
| 3 | 1 | 2 | 1 | 3 |
| 2 | 1 | 2 | 1 | 2 |
| 2 | 1 | 2 | 3 | 1 |

Move the filter over the entire image from left to right, top to bottom

1x1 + 2x3 + 1x3 + 1x2 + 2x1 + 1x2 + 1x3 + 2x2 + 1x3

| | | | | |
|---|---|---|---|---|
| 1 | 2 | 2 | 3 | 1 |
| 1 | 1 | 3 | 3 | 2 |
| 3 | 2 | 1 | 2 | 3 |
| 2 | 3 | 2 | 3 | 2 |
| 2 | 1 | 2 | 3 | 1 |

| | | |
|---|---|---|
| 21 | 25 | 27 |
| 25 | 26 | |
| | | |

# Illustration of Convolution

## Given this 5x5 image, convolve with a 3x3 filter

| | | | | |
|---|---|---|---|---|
| 1 | 2 | 2 | 3 | 1 |
| 1 | 1 | 2 | 1 | 2 |
| 3 | 1 | 2 | 1 | 3 |
| 2 | 1 | 2 | 1 | 2 |
| 2 | 1 | 2 | 3 | 1 |

Move the filter over the entire image from left to right, top to bottom

1x1 + 2x3 + 1x3 + 1x2 + 2x1 + 1x2 + 1x3 + 2x2 + 1x3

| | | | | |
|---|---|---|---|---|
| 1 | 2 | 2 | 3 | 1 |
| 1 | 1 | 3 | 3 | 2 |
| 3 | 2 | 1 | 2 | 3 |
| 2 | 3 | 2 | 3 | 2 |
| 2 | 1 | 2 | 3 | 1 |

| | | |
|---|---|---|
| 21 | 25 | 27 |
| 25 | 26 | |
| | | |

# Illustration of Convolution

## Given this 5x5 image, convolve with a 3x3 filter

| | | | | |
|---|---|---|---|---|
| 1 | 2 | 2 | 3 | 1 |
| 1 | 1 | 3 | 3 | 2 |
| 1 | 2 | 1 | 2 | 3 |
| 1 | 2 | 1 | 3 | 2 |
| 1 | 2 | 1 | 3 | 1 |

Move the filter over the entire image from left to right, top to bottom

1x3 + 2x2 + 1x1 + 1x2 + 2x3 + 1x2 + 1x2 + 2x1 + 1x2

| | | |
|---|---|---|
| 21 | 25 | 27 |
| 25 | 26 | 29 |
| 24 | | |

| | | | | |
|---|---|---|---|---|
| 1 | 2 | 2 | 3 | 1 |
| 1 | 1 | 3 | 3 | 2 |
| 3 | 2 | 1 | 2 | 3 |
| 2 | 3 | 2 | 3 | 2 |
| 2 | 1 | 2 | 3 | 1 |

# Illustration of Convolution

## Given this 5x5 image, convolve with a 3x3 filter

| | | | | |
|---|---|---|---|---|
| 1 | 2 | 2 | 3 | 1 |
| 1 | 1 | 3 | 3 | 2 |
| 3 | 1 | 2 | 1 | 3 |
| 2 | 1 | 2 | 1 | 2 |
| 2 | 1 | 2 | 1 | 1 |

Move the filter over the entire image from left to right, top to bottom

1x2 + 2x1 + 1x2 + 1x3 + 2x2 + 1x3 + 1x1 + 2x2 + 1x3

| | | | | |
|---|---|---|---|---|
| 1 | 2 | 2 | 3 | 1 |
| 1 | 1 | 3 | 3 | 2 |
| 3 | 2 | 1 | 2 | 3 |
| 2 | 3 | 2 | 3 | 2 |
| 2 | 1 | 2 | 3 | 1 |

| | | |
|---|---|---|
| 21 | 25 | 27 |
| 25 | 26 | 29 |
| 24 | 24 | |

# Illustration of Convolution

## Given this 5x5 image, convolve with a 3x3 filter


Move the filter over the entire image from left to right, top to bottom

1x1 + 2x2 + 1x3 + 1x2 + 2x3 + 1x2 + 1x2 + 2x3 + 1x1

| | | |
|---|---|---|
| 21 | 25 | 27 |
| 25 | 26 | 29 |
| 24 | 24 | 27 |

# Convolution - Example

- Edge detection with Sobel filter (or Sobel kernel)
  - Convolving the image I with this operator will result in Jv, the image of vertical edges

| | | |
|---|---|---|
| $-1$ | $0$ | $1$ |
| $-2$ | $0$ | $2$ |
| $-1$ | $0$ | $1$ |

  - Convolving the image I with this operator will result in Jh, the image of horizontal edges

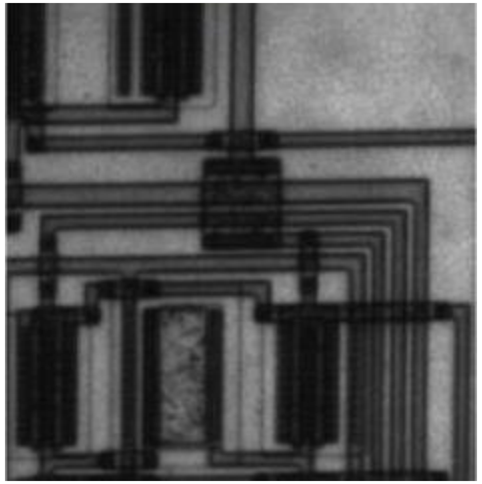
| | | |
|---|---|---|
| $1$ | $2$ | $1$ |
| $0$ | $0$ | $0$ |
| $-1$ | $-2$ | $-1$ |

- Combine the two filters to give a single measure of gradient magnitude

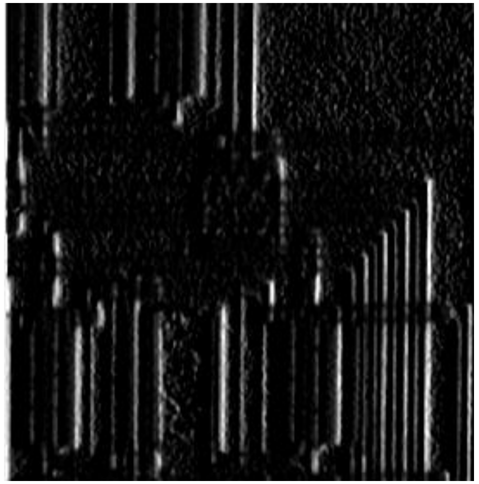
$$J(x,y) = \sqrt{J_h(x,y)^2 + J_v(x,y)^2}$$

# Convolution - Example

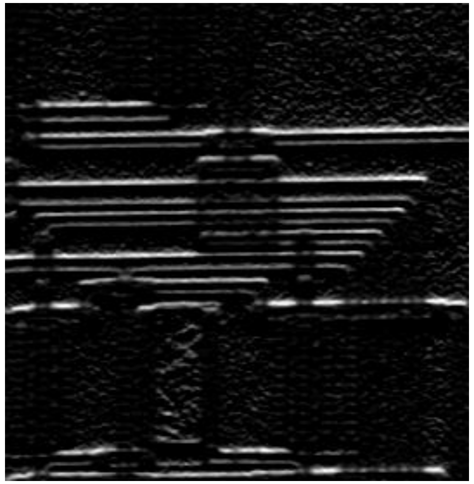
- Edge detection with Sobel filter (or Sobel kernel)



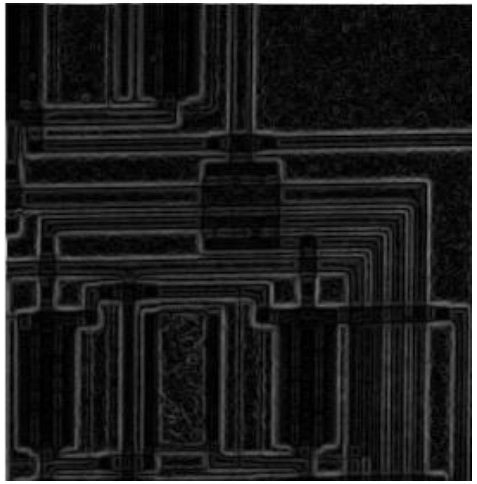
Input image



Vertical edges Iy



Horizontal edges Ix



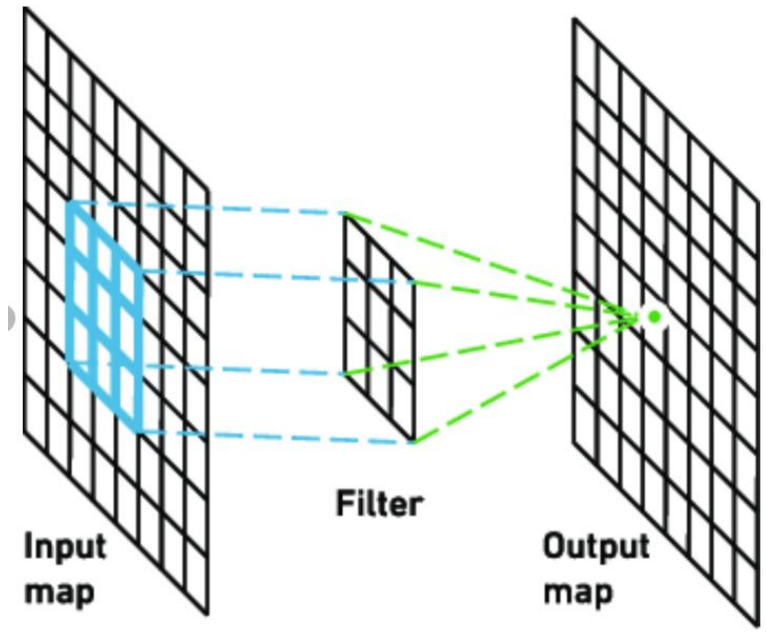
$G = \sqrt{Ix^2 + Iy^2}$

# Convolution

- Question
  - Can you implement the Edge detector using neurons?

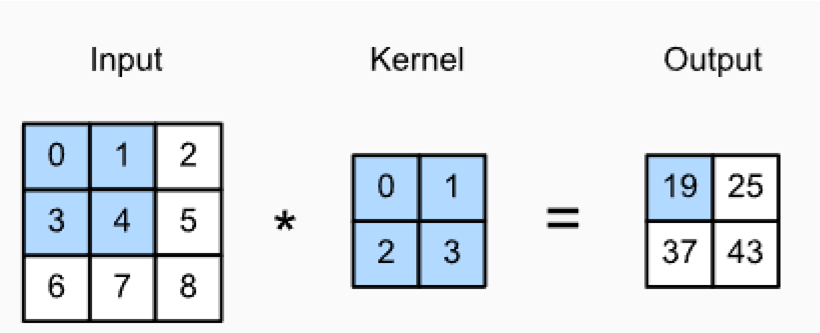
# Convolution Layer

- At each image pixel p, unlike dense, or fully-connected, layer, convolutional layers take a weighted sum of the pixels around p
  - Think about it as a filter BUT the values of the filter (kernel) are unknown
  - The goal of training is to learn the values of the elements of the filter that best suit the task at hand



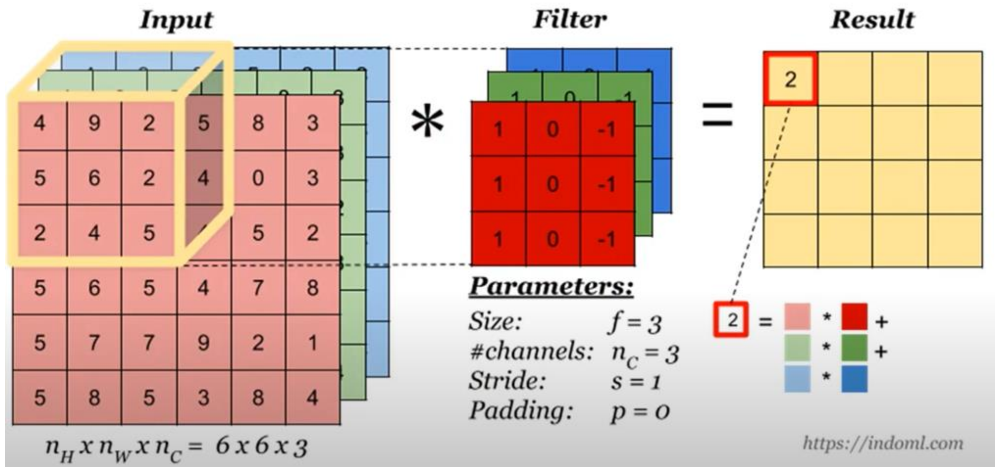
# Convolution Layer – one filter case

- Greyscale image
  - Has d = 1 channel (for the grey level)
  - The filter is then of size s x s (e.g., s can be 3, 5, 7, ..)




https://d2l.ai/chapter_convolutional-neural-networks/conv-layer.html

- Color image
  - Has d=3 channels (one for R, one for G, and another for B)
  - The filter is then of size d x s x s (e.g., s can be 3, 5, 7, ..)




https://www.youtube.com/watch?v=3myNsOGhc3A&list=PLbNJQ-D5RH18U7BXm3NH8_6mnguQ45uRh&index=5

# Convolution Layer – one filter case

- Convolutional layer
  - Cross correlates the input image and the kernel and adds a bias to produce an output
- The two (learnable) parameters of a convolutional layer are
  - The kernel values
  - The bias
- The hyper parameters (not learnable, set by the user) are
  - The kernel size
    - It defines the receptive field size
    - Receptive field refers to all the elements from the previous layer that may affect the calculation of the output value
- The output of a convolutional layer is called feature map

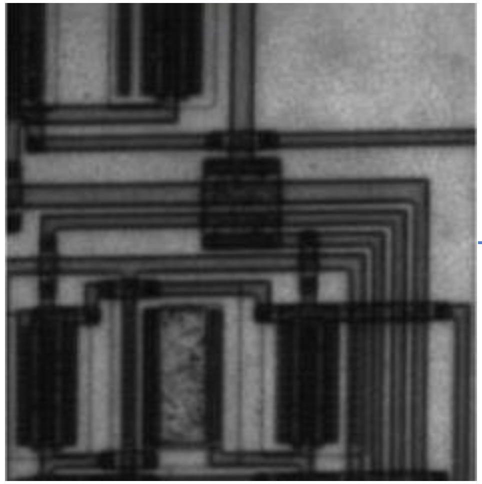
# Convolution Layer – Multiple Feature Maps

- Apply different kernels to the input
  - Each kernel will produce one output, called feature map
  - Stack the feature maps together to produce a multichannel output

- Hyper parameters
  - The kernel size
    - It defines the receptive field size
    - Receptive field refers to all the elements from the previous layer that may affect the calculation of the output value
  - No. of output channels at each layer
    - It defines the no. of different filters to learn at each layer

# Convolution Layer – Multiple Feature Maps

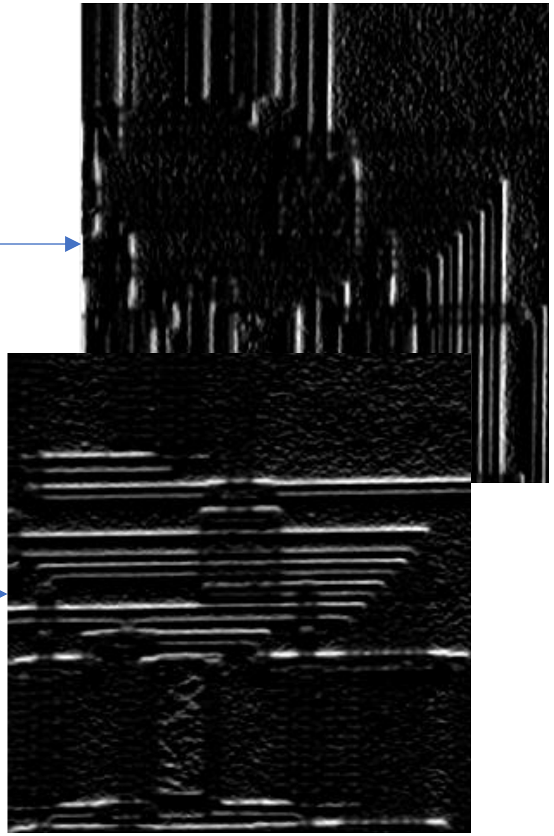
- Example – Edge detection with Sobel filter (or Sobel kernel)



Input image

$$\begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array}$$

$$\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array}$$



# Convolution Layer – Practical Considerations

- Padding
  - When you convolve an image with a filter, we will lose pixels at the boundary of the image

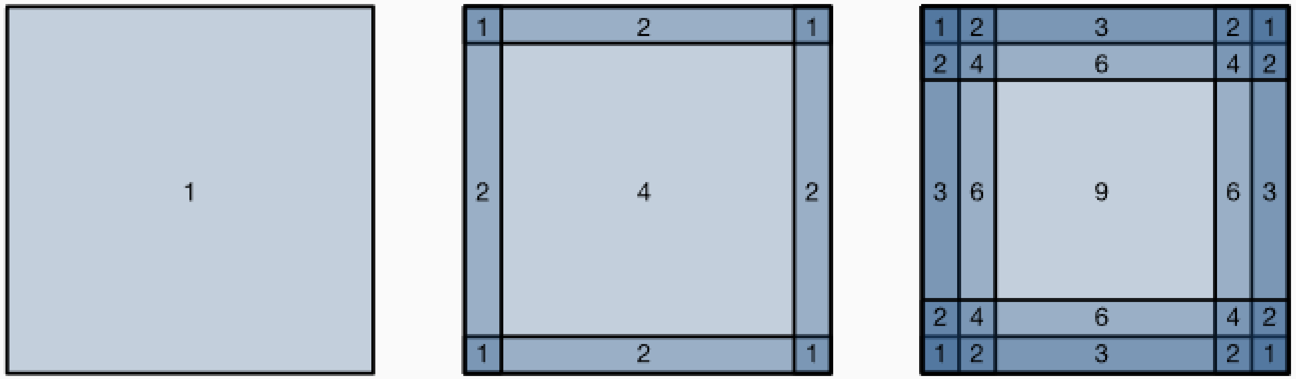
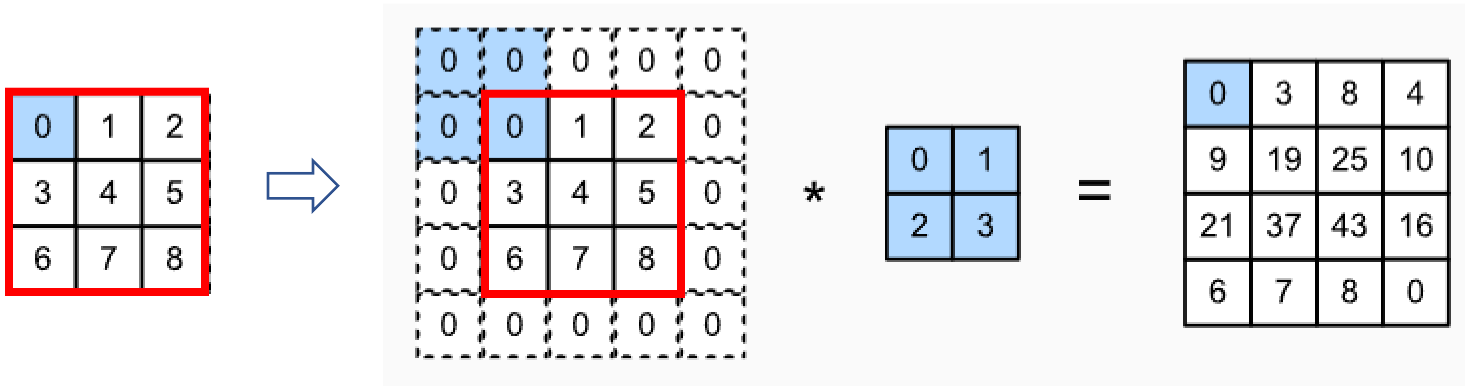



*Fig. 7.3.1* Pixel utilization for convolutions of size $1 \times 1$, $2 \times 2$, and $3 \times 3$ respectively.

https://d2l.ai/chapter_convolutional-neural-networks/padding-and-strides.html

# Convolution Layer – Practical Considerations

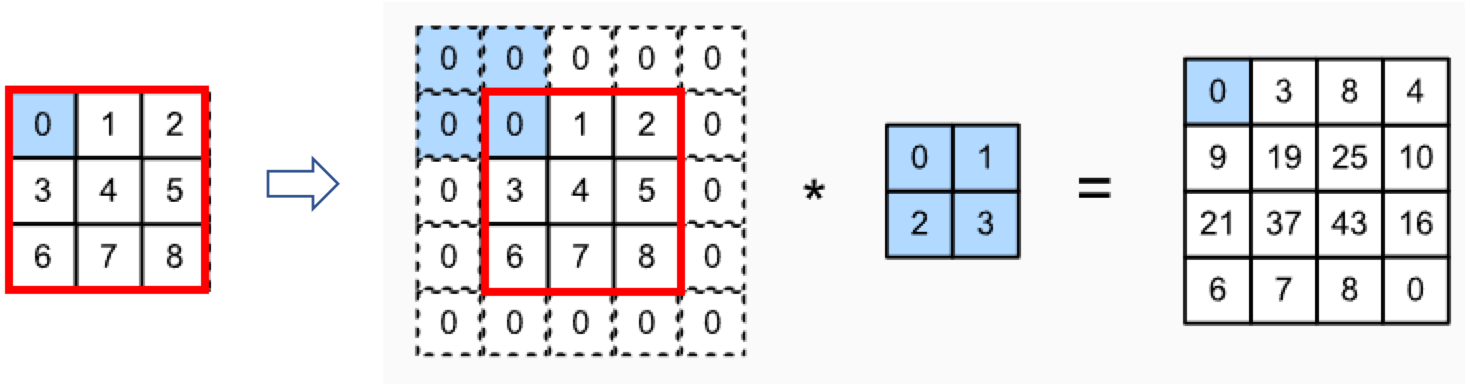
- Padding
  - When you convolve an image with a filter, we will loose pixels at the boundary of the image
- Solution → Padding
  - Add extra pixels (set to 0) of filler around the boundary of the input image




https://d2l.ai/chapter_convolutional-neural-networks/padding-and-strides.html

# Convolution Layer – Practical Considerations

- Padding
  - When you convolve an image with a filter, we will loose pixels at the boundary of the image
- Solution → Padding
  - Add extra pixels (set to 0) of filler around the boundary of the input image
  - Typically, we use kernels with odd height and width, e.g., 1, 3, 5, 7




https://d2l.ai/chapter_convolutional-neural-networks/padding-and-strides.html

# Convolution Layer – Practical Considerations

- Typically,
  - The convolution window starts at the upper-left corner, and then we slide it over all locations both down and to the right
  - We slide the window one element at a time
  - For computational efficient (and to downsample)
    - Slide the window more than one elements, skipping the intermediate locations
  - The number of rows (and columns) skipped is called stride

- Hyper parameters
  - The kernel size
  - No. of output channels
  - Padding
  - The stride

# Illustration of Convolution with Stride

Given this 5x5 image, convolve with a 3x3 filter

| | | | | |
|---|---|---|---|---|
| 1 | 2 | 2 | 3 | 1 |
| 1 | 1 | 3 | 3 | 2 |
| 3 | 2 | 1 | 2 | 3 |
| 2 | 3 | 2 | 3 | 2 |
| 2 | 1 | 2 | 3 | 1 |

| | | |
|---|---|---|
| 1 | 2 | 1 |
| 1 | 2 | 1 |
| 1 | 2 | 1 |

# Illustration of Convolution with Stride

## Given this 5x5 image, convolve with a 3x3 filter **with Stride = 2**

| | | | | |
|---|---|---|---|---|
| 1 | 2 | 1 | 3 | 1 |
| 1 | 2 | 1 | 3 | 2 |
| 1 | 2 | 1 | 2 | 3 |
| 2 | 3 | 2 | 3 | 2 |
| 2 | 1 | 2 | 3 | 1 |

Move the filter over the entire image from left to right, top to bottom

1x1 + 2x2 + 1x2 + 1x1 + 2x1 + 1x3 + 1x3 + 2x2 + 1x1

| | |
|---|---|
| 21 | |
| | |

| | | | | |
|---|---|---|---|---|
| 1 | 2 | 2 | 3 | 1 |
| 1 | 1 | 3 | 3 | 2 |
| 3 | 2 | 1 | 2 | 3 |
| 2 | 3 | 2 | 3 | 2 |
| 2 | 1 | 2 | 3 | 1 |

# Illustration of Convolution with Stride

## Given this 5x5 image, convolve with a 3x3 filter **with Stride = 2**

| | | | | |
|---|---|---|---|---|
| 1 | 2 | 1 | 2 | 1 |
| 1 | 1 | 1 | 2 | 1 |
| 3 | 2 | 1 | 2 | 1 |
| 2 | 3 | 2 | 3 | 2 |
| 2 | 1 | 2 | 3 | 1 |

Move the filter over the entire image from left to right, top to bottom **skipping 2 columns or rows respectively**

1x2 + 2x3 + 1x1 + 1x3 + 2x3 + 1x2 + 1x1 + 2x2 + 1x3

| | |
|---|---|
| 21 | 28 |
| | |

| | | | | |
|---|---|---|---|---|
| 1 | 2 | 2 | 3 | 1 |
| 1 | 1 | 3 | 3 | 2 |
| 3 | 2 | 1 | 2 | 3 |
| 2 | 3 | 2 | 3 | 2 |
| 2 | 1 | 2 | 3 | 1 |

# Illustration of Convolution with Stride

## Given this 5x5 image, convolve with a 3x3 filter **with Stride = 2**

| | | | | |
|---|---|---|---|---|
| 1 | 2 | 2 | 3 | 1 |
| 1 | 1 | 3 | 3 | 2 |
| 1 | 2 | 1 | 2 | 3 |
| 1 | 2 | 1 | 3 | 2 |
| 1 | 2 | 1 | 3 | 1 |

Move the filter over the entire image from left to right, top to bottom **skipping 2 columns or rows respectively**

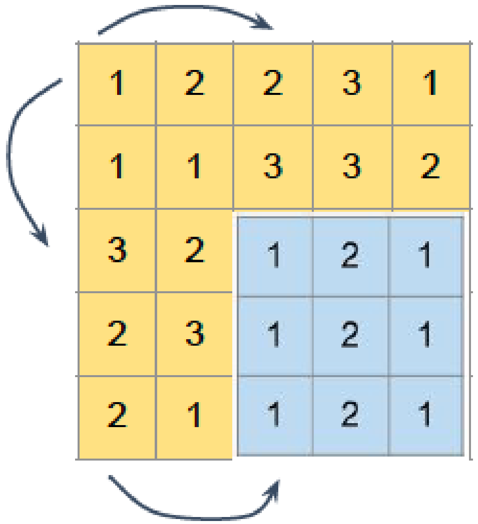
1x3 + 2x2 + 1x1 + 1x2 + 2x3 + 1x2 + 1x2 + 2x1 + 1x2

| | |
|---|---|
| 21 | 28 |
| 24 | |

| | | | | |
|---|---|---|---|---|
| 1 | 2 | 2 | 3 | 1 |
| 1 | 1 | 3 | 3 | 2 |
| 3 | 2 | 1 | 2 | 3 |
| 2 | 3 | 2 | 3 | 2 |
| 2 | 1 | 2 | 3 | 1 |

# Illustration of Convolution with Stride

## Given this 5x5 image, convolve with a 3x3 filter **with Stride = 2**




Move the filter over the entire image from left to right, top to bottom **skipping 2 columns or rows respectively**

1x1 + 2x2 + 1x3 + 1x2 + 2x3 + 1x2 + 1x2 + 2x3 + 1x1

| | |
|---|---|
| 21 | 28 |
| 24 | 27 |

| | | | | |
|---|---|---|---|---|
| 1 | 2 | 2 | 3 | 1 |
| 1 | 1 | 3 | 3 | 2 |
| 3 | 2 | 1 | 2 | 3 |
| 2 | 3 | 2 | 3 | 2 |
| 2 | 1 | 2 | 3 | 1 |

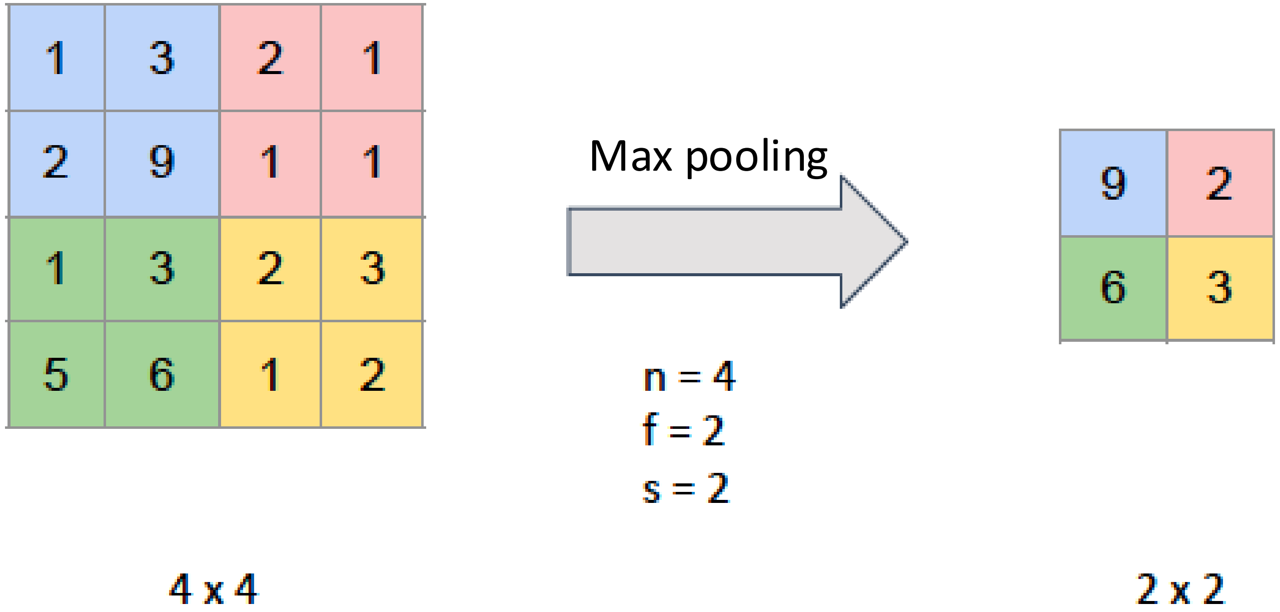
# Pooling Layer

- Pooling is the operation of aggregating the values within a window around a pixel to produce one singe value, e.g.,
  - Max pooling
    - takes the maximum of all the values within a window around a pixel
  - Min pooling
    - takes the minimum of all the values within a window around a pixel
  - Average pooling
    - takes the minimum of all the values within a window around a pixel

# Pooling Layer

- Pooling is the operation of aggregating the values within a window around a pixel to produce one singe value, e.g.,
  - Max pooling
    - takes the maximum of all the values within a window around a pixel
  - Min pooling
    - takes the minimum of all the values within a window around a pixel
  - Average pooling
    - takes the minimum of all the values within a window around a pixel
- It is important when stacking multiple layers
  - Reduces the size of the output of a layer before feeding it to the next layer
- Advantages
  - Reduce the size of the feature map
  - Although the size of the filters remain the same, the receptive field will increase with the depth of the network – thus subsequent layers will capture bigger context

# Illustration of Max Pooling



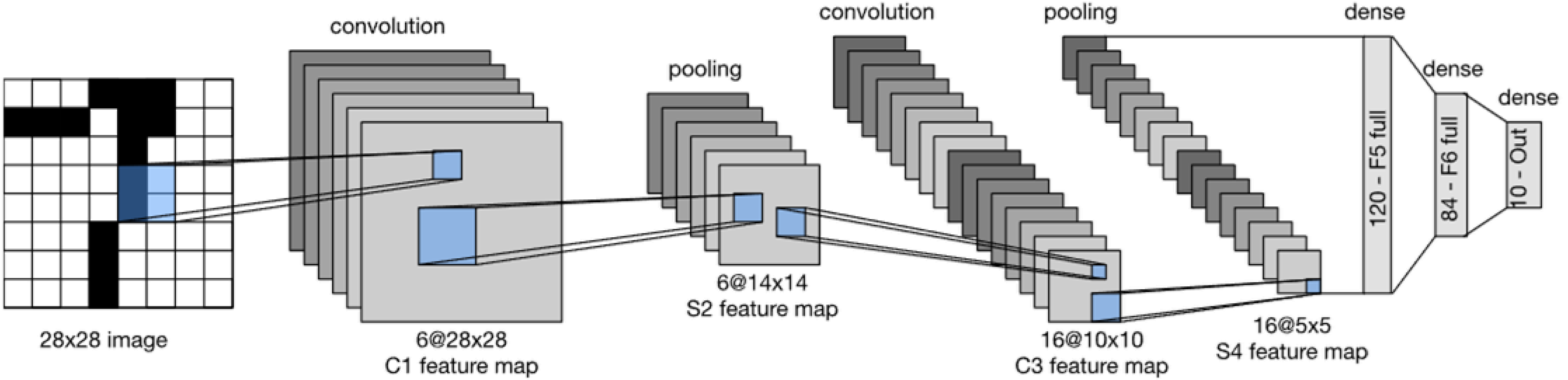
# In this Lecture

- Introduction
  - Linear regression, perceptron and MLP
  - Why we need CNNs
- Convolutional Neural Network (CNN)
  - Convolutions
  - Convolution layers
- Examples of CNNs
  - LeNet
- Summary

- Learning objectives
  - Understand CNN and the different components that compose it
  - Implement CNNs in Python, NumPy and PyTorch
- Additional readings
  - Chapter 7 of the textbook, available at: https://d2l.ai/

# Convolutional Neural Networks: LeNet

- LeNet, the first published CNN (by Yann LeCun in 1998), used to recognize handwritten digits in images
  - Used to recognize digits for processing deposits in ATM machines (1990s)
  - Some ATM machines are still using this code!




Convolutional encoder composed of 2 convolutional layers (5x5 kernel), each layer has sigmoid activation and is followed by 2x2 average pooling and stride 2

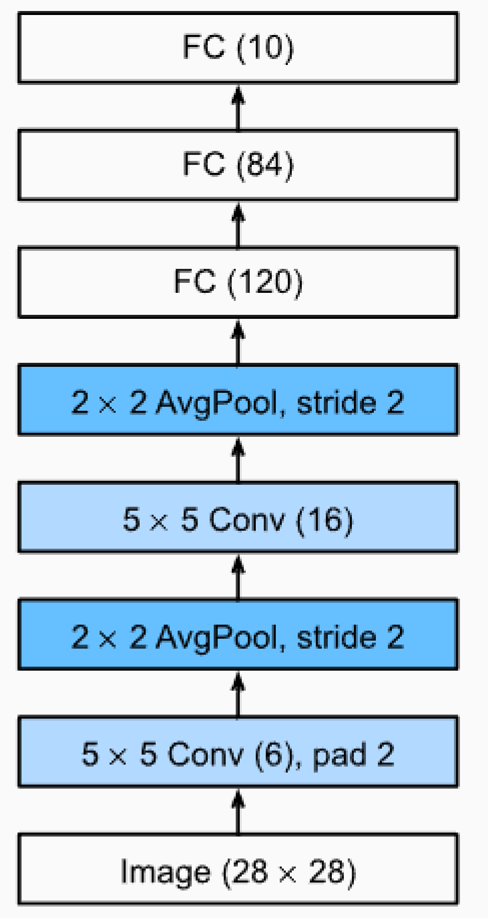
Two fully connected layers



https://d2l.ai/chapter_convolutional-neural-networks/padding-and-strides.html

# Convolutional Neural Networks: LeNet

- LeNet, the first published CNN (by Yann LeCun in 1998), used to recognize handwritten digits in images
  - Also used to recognize digits for processing deposits in ATM machin ATM machines are still using this code!
- In the lab,
  - You will create the network and train it on the MNIST dataset






https://d2l.ai/chapter_convolutional-neural-networks/padding-and-strides.html

# Summary

- We derived the structure of convolutional layers
- In the lab
  - You will create and train LeNet
- Next week
  - Training (including data preparation), validation and testing
  - Tuning the hyper parameters
  - Modern Convolutional Neural Networks

# Questions