UNIVERSITY OF THE WITWATERSRAND, JOHANNESBURG

# COMS 4030A/7047A Adaptive Computation and Machine Learning

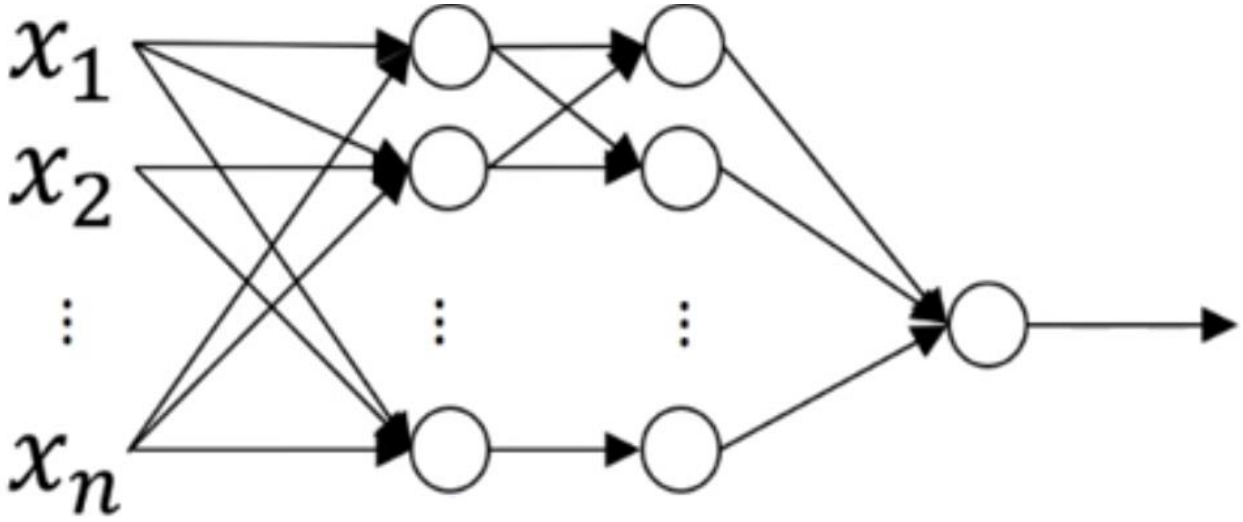
Hima Vadapalli

Semester I, 2022

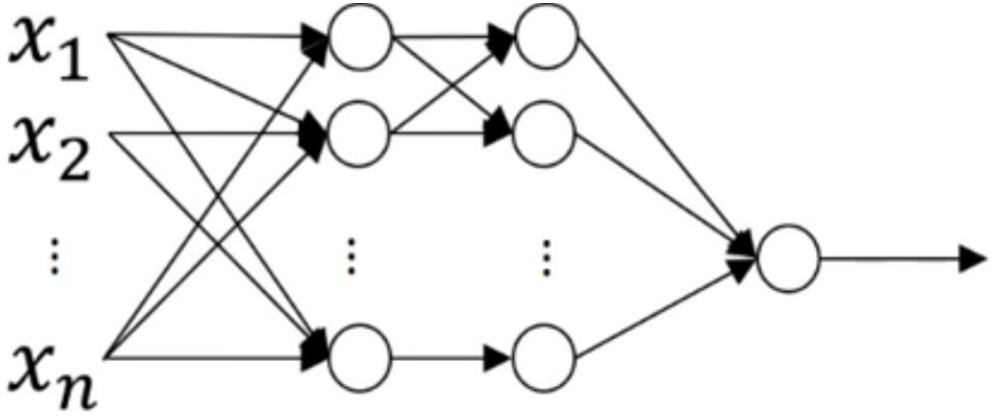
# *Convolutional Neural Networks*

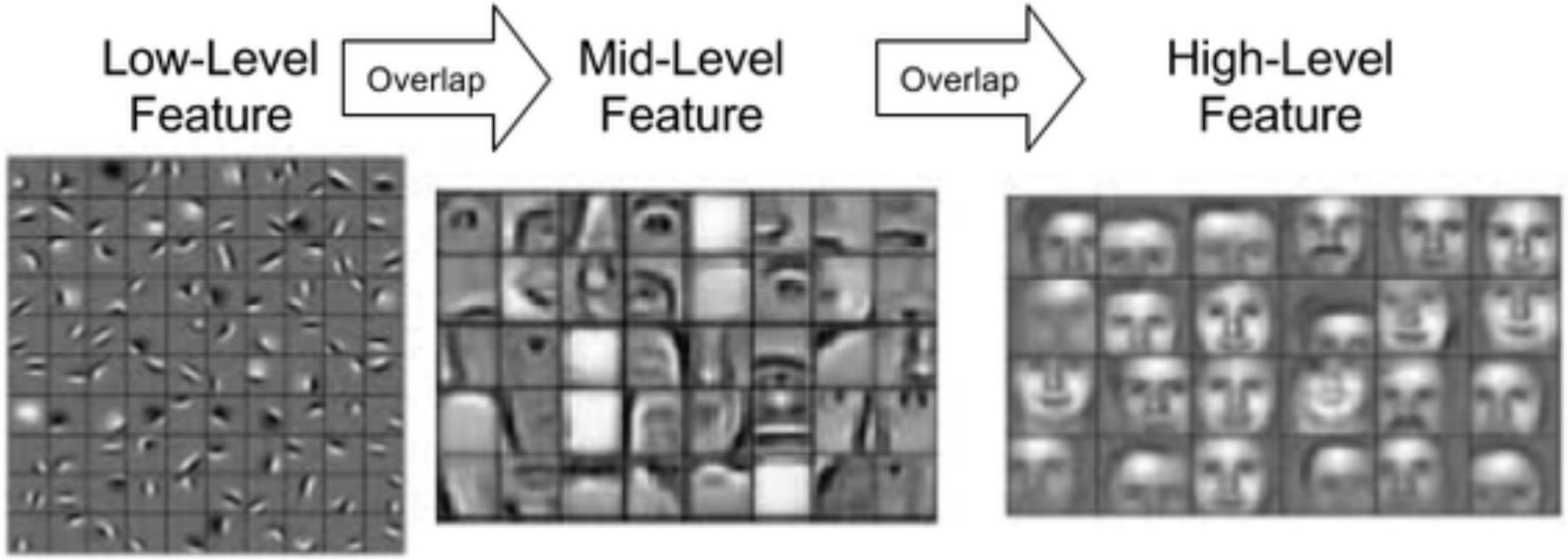
Slides based heavily on course material by Eric Eton Andrew Moore and Andrew Ng

# Problem domain

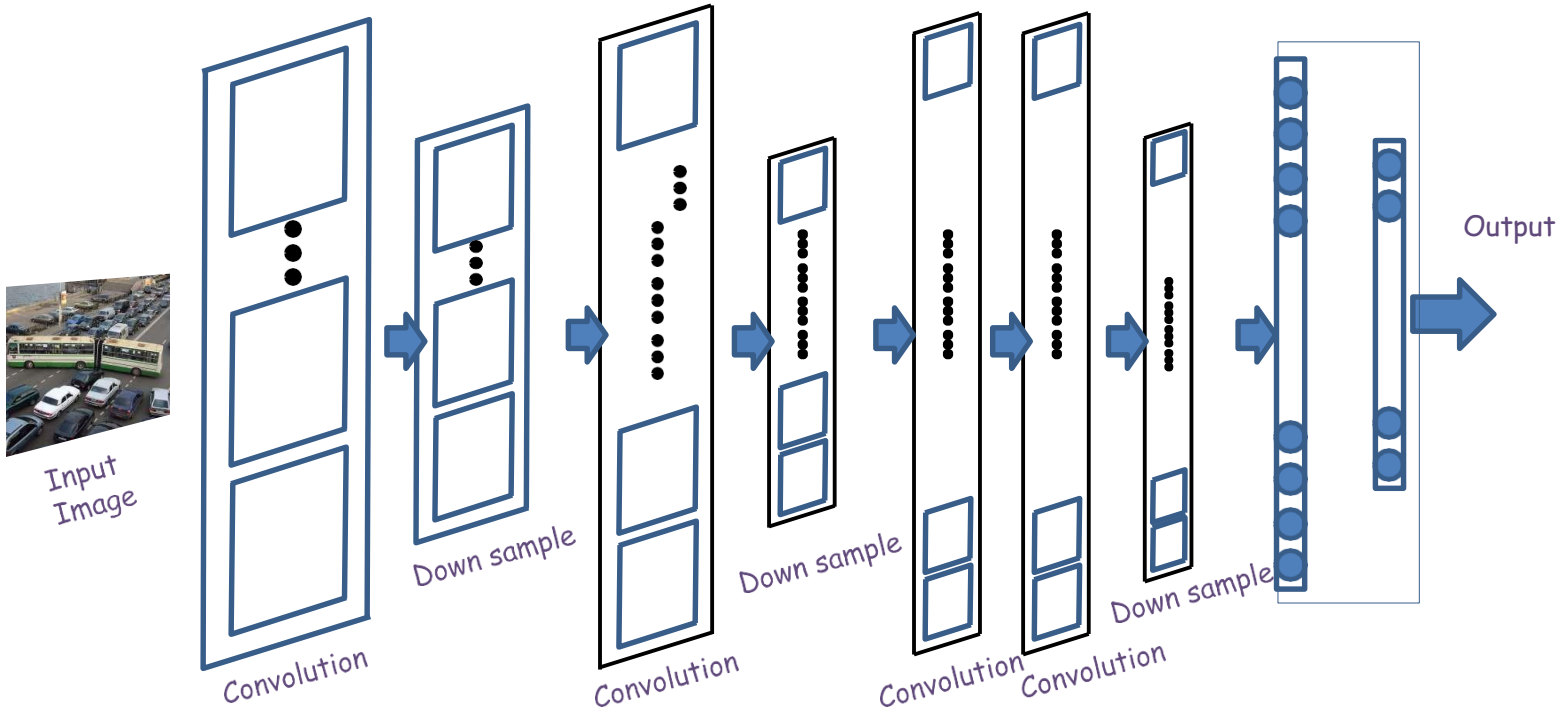


# Problem domain



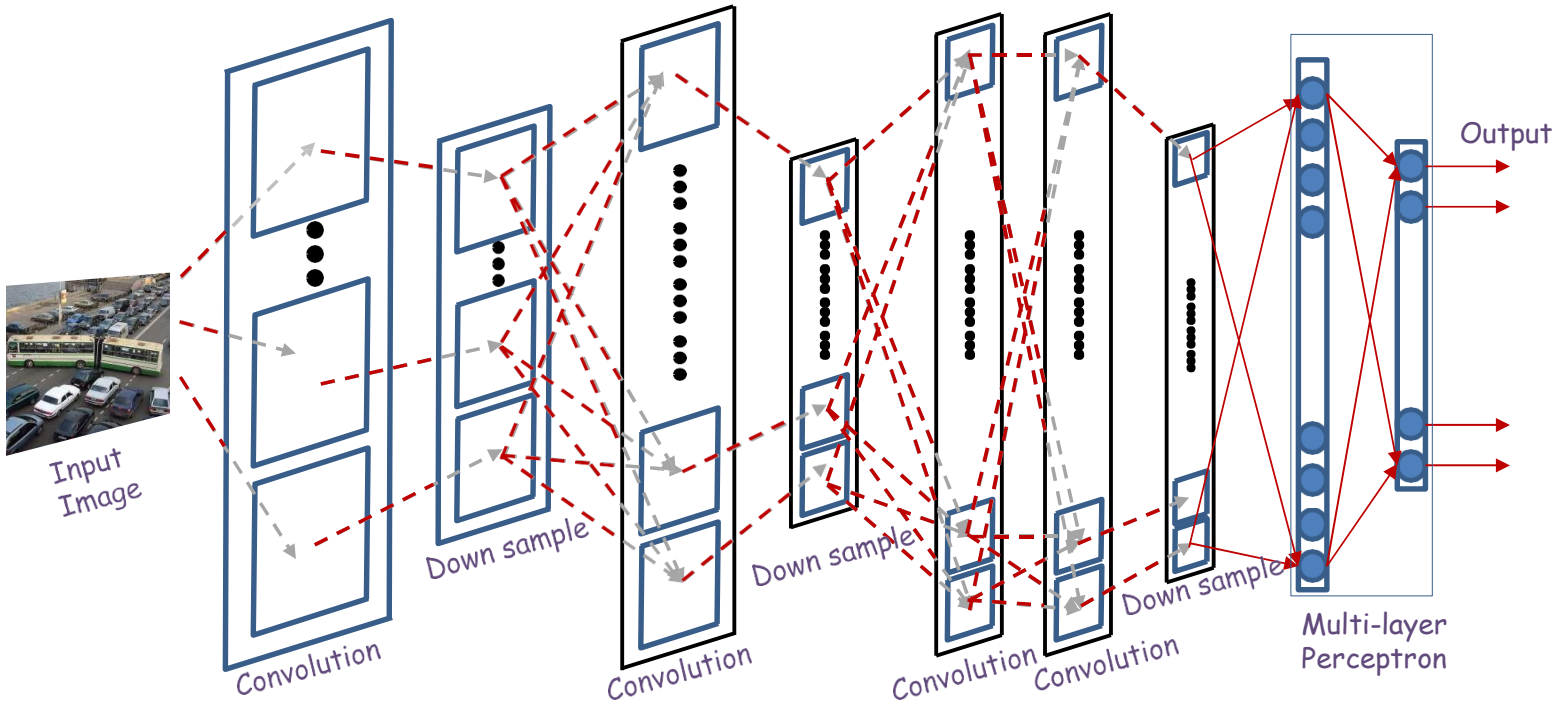



# The general architecture of a convolutional neural network




.

# The general architecture of a convolutional neural network



# What is a convolution?

Example 5x5 image with binary pixels

| | | | | |
|---|---|---|---|---|
| 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 |

Example 3x3 filter

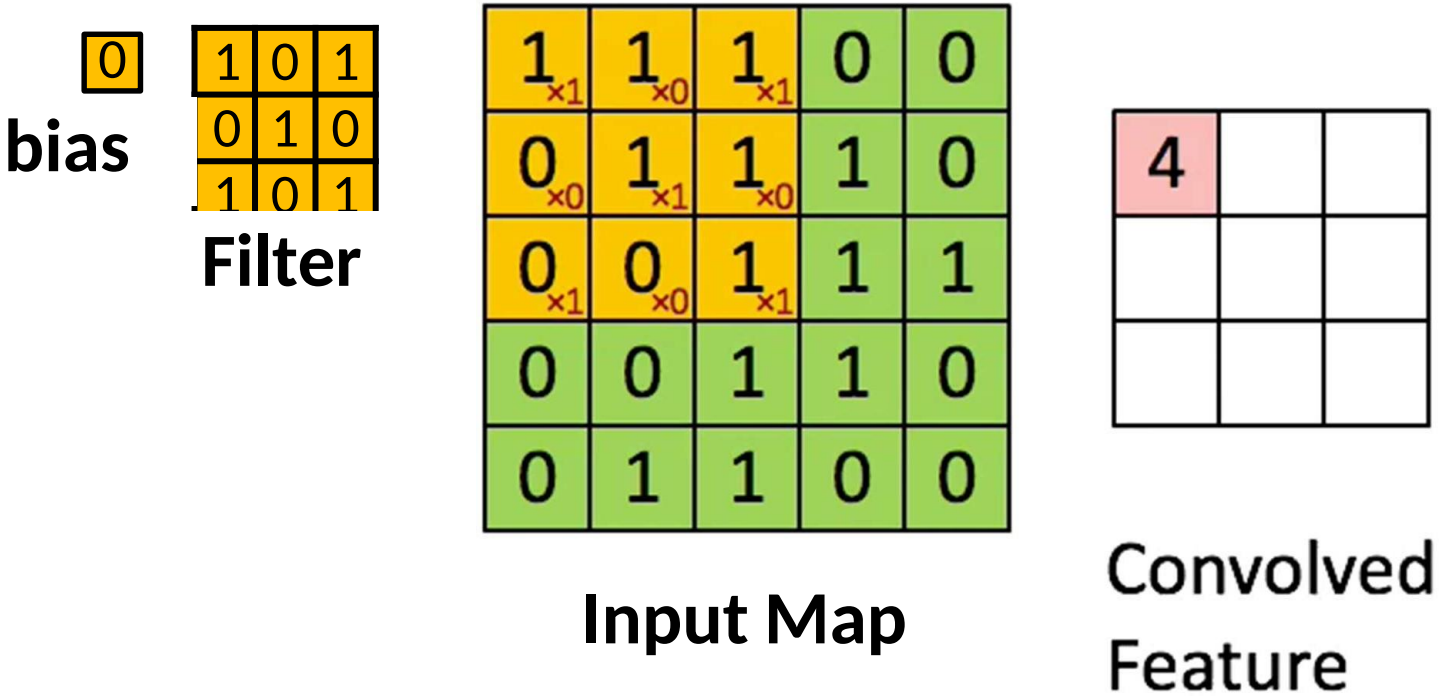
| | | |
|---|---|---|
| 1 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |

bias

| |
|---|
| 0 |

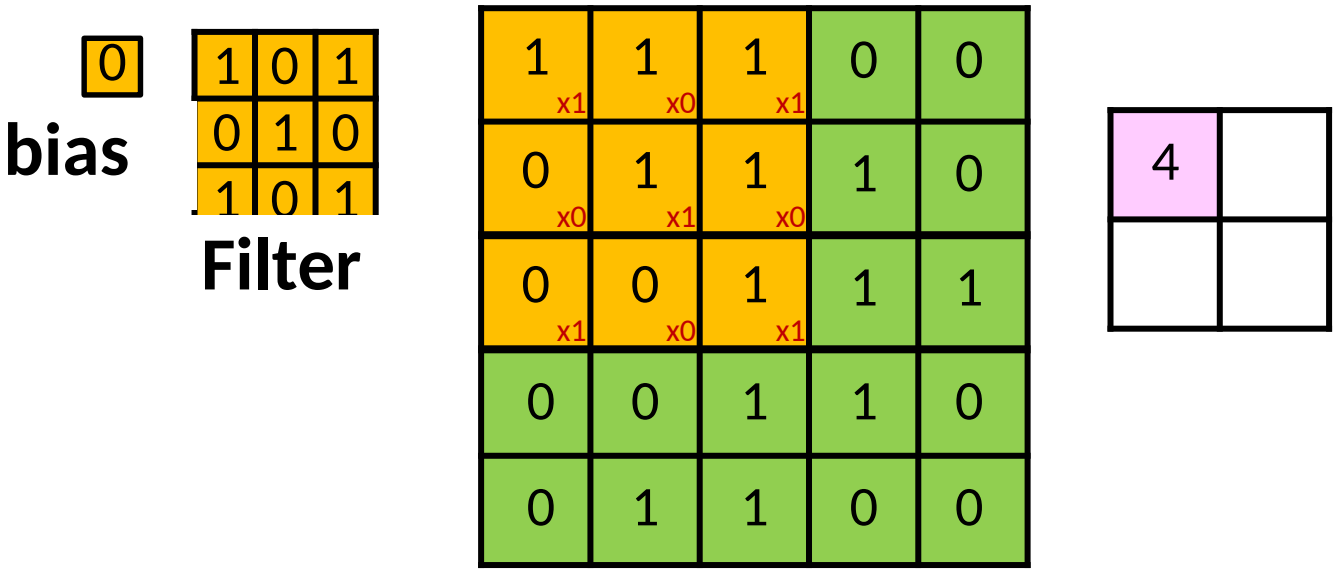
- Scanning an image with a “filter”
  - Note: a filter is really just a perceptron, with weights and a bias

# What is a convolution?



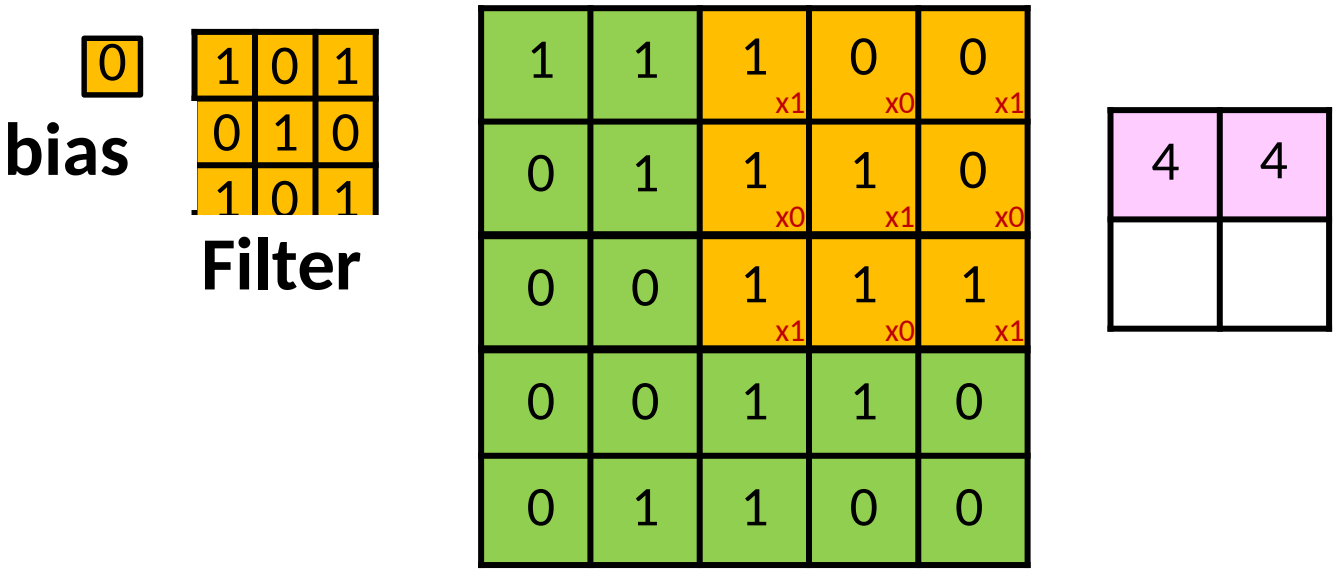

- Scanning an image with a “filter”
  - At each location, the “filter and the underlying map values are multiplied component wise, and the products are added along with the bias

# The “Stride” between adjacent scanned locations need not be 1



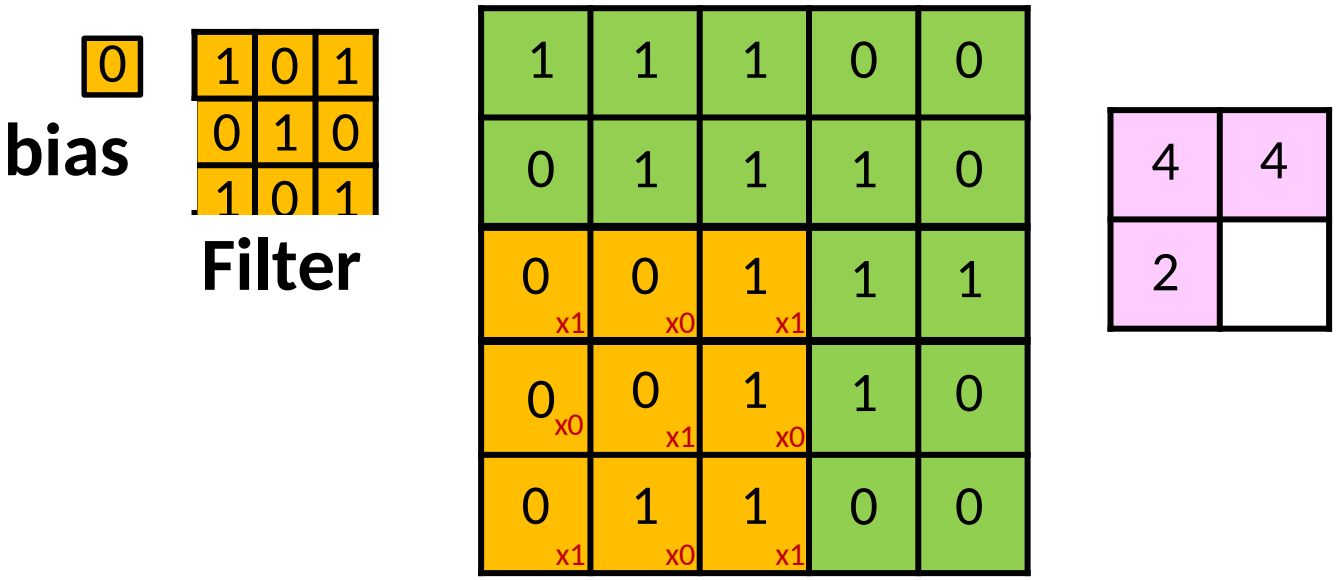

- Scanning an image with a “filter”
  - The filter may proceed by *more* than 1 pixel at a time
  - E.g. with a “stride” of *two* pixels per shift

# The “Stride” between adjacent scanned locations need not be 1



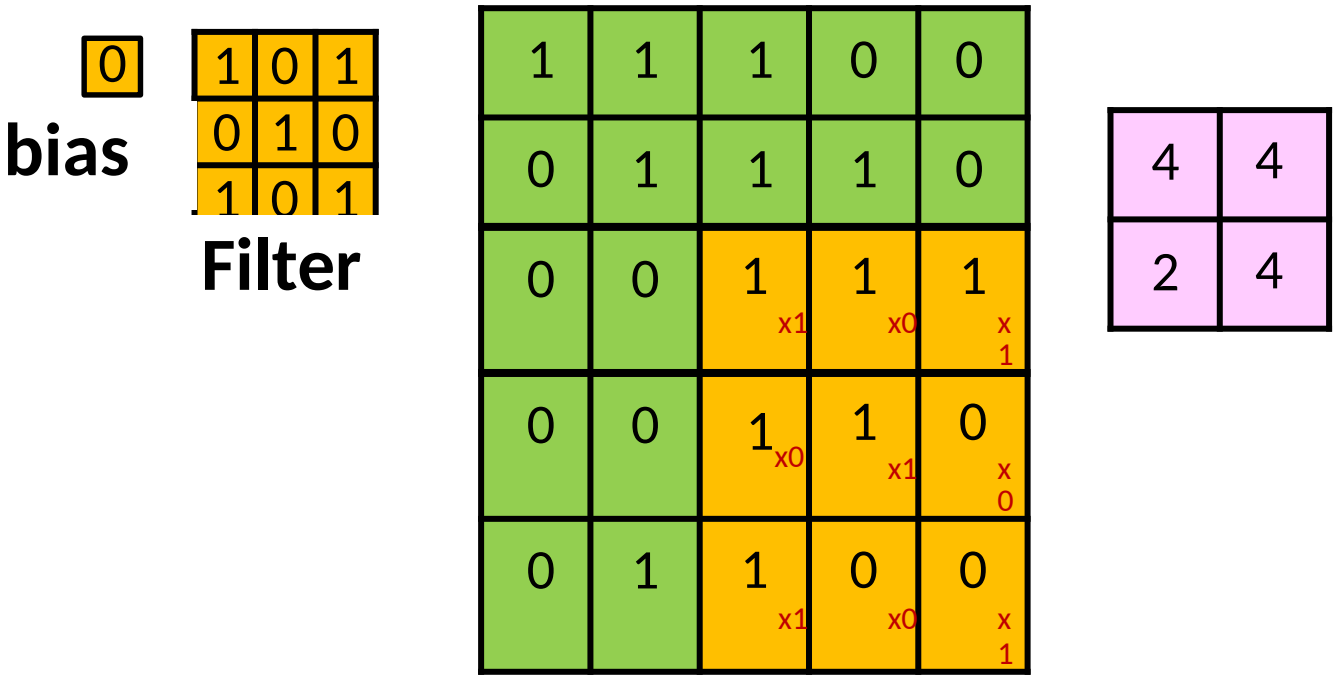

- Scanning an image with a “filter”
  - The filter may proceed by *more* than 1 pixel at a time
  - E.g. with a “hop” of *two* pixels per shift

# The “Stride” between adjacent scanned locations need not be 1



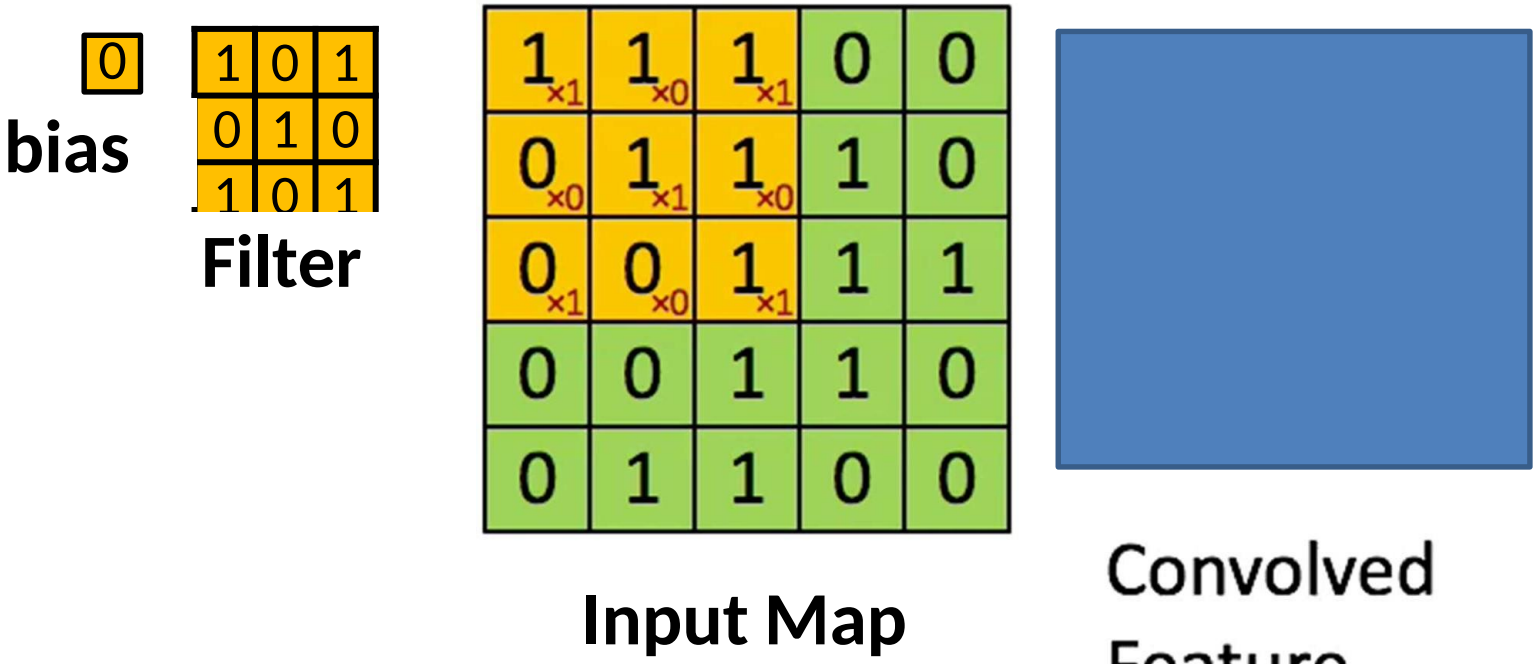

- Scanning an image with a “filter”
  - The filter may proceed by *more* than 1 pixel at a time
  - E.g. with a “hop” of *two* pixels per shift

# The “Stride” between adjacent scanned locations need not be 1



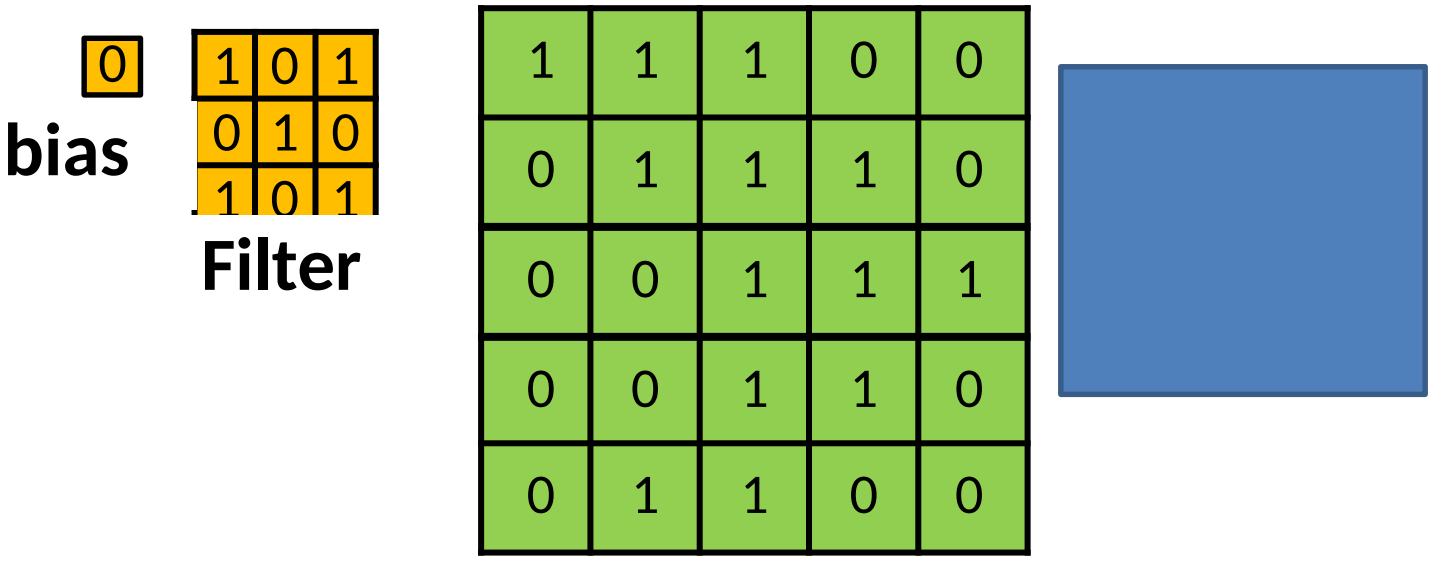

- Scanning an image with a “filter”
  - The filter may proceed by *more* than 1 pixel at a time
  - E.g. with a “hop” of *two* pixels per shift

# The size of the convolution



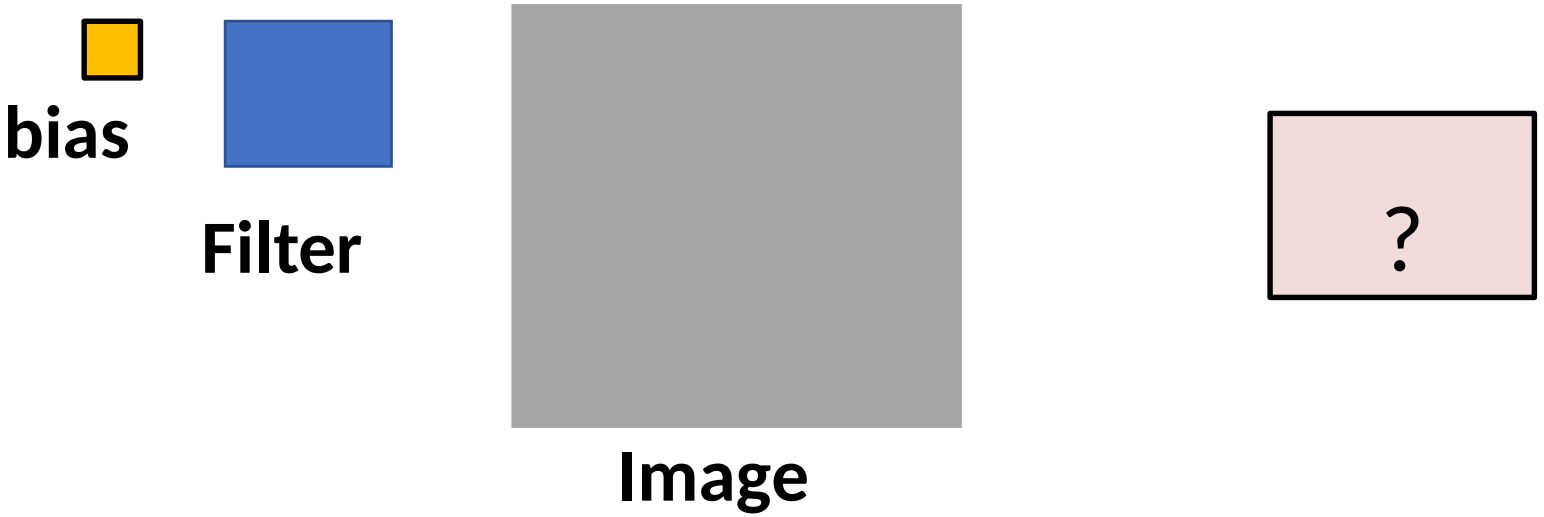

- Image size: 5x5
- Filter: 3x3
- “Stride”: 1
- Output size = ?

# The size of the convolution




- Image size: 5x5
- Filter: 3x3
- Stride: 2
- Output size = ?

# The size of the convolution




- Image size: n x n
- Filter: f x f
- Stride: s
- Output size (each side) = [(n-f)/s]+1
  - Assuming you're not allowed to go beyond the edge of the input

# How much does each pixel contribute?

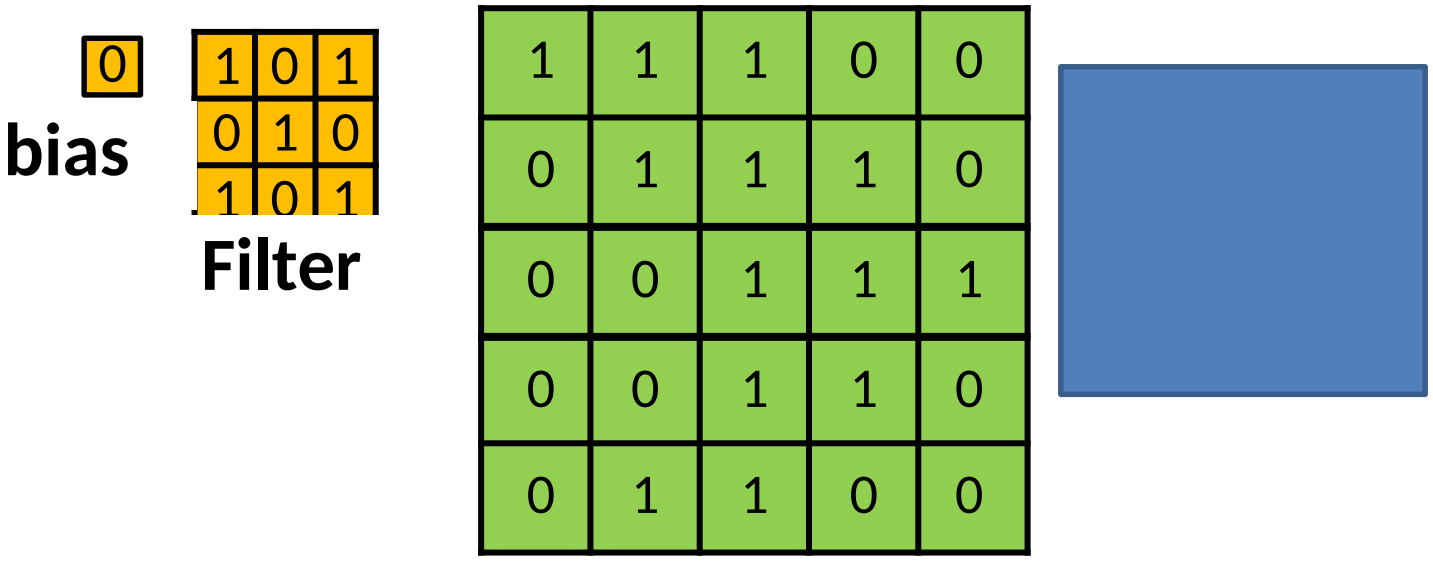
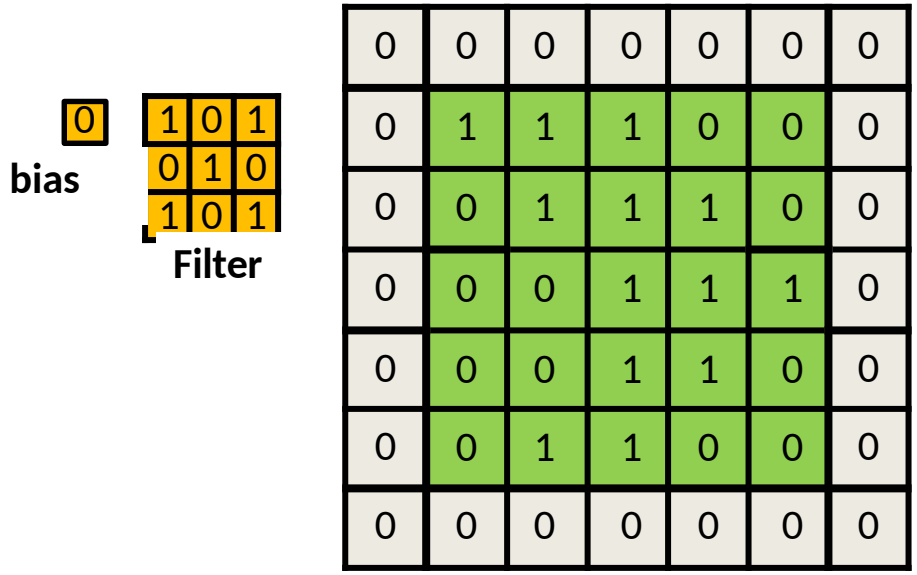



Image size: 5x5

Filter: 3x3

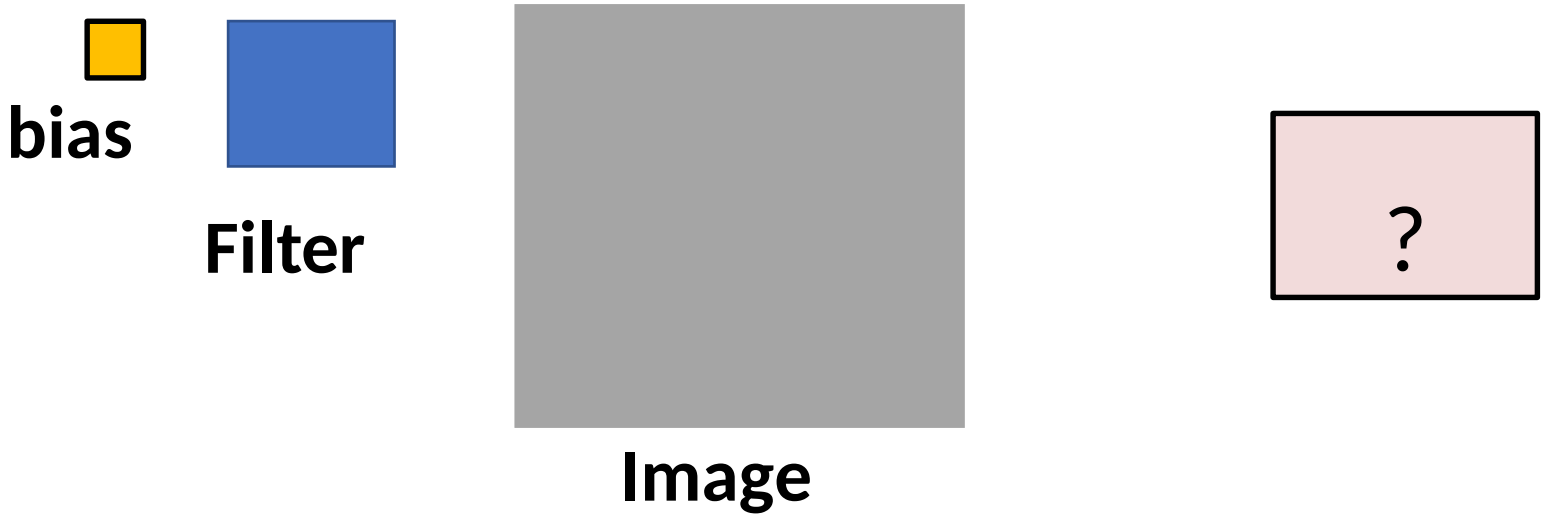
Stride =2

# Solution



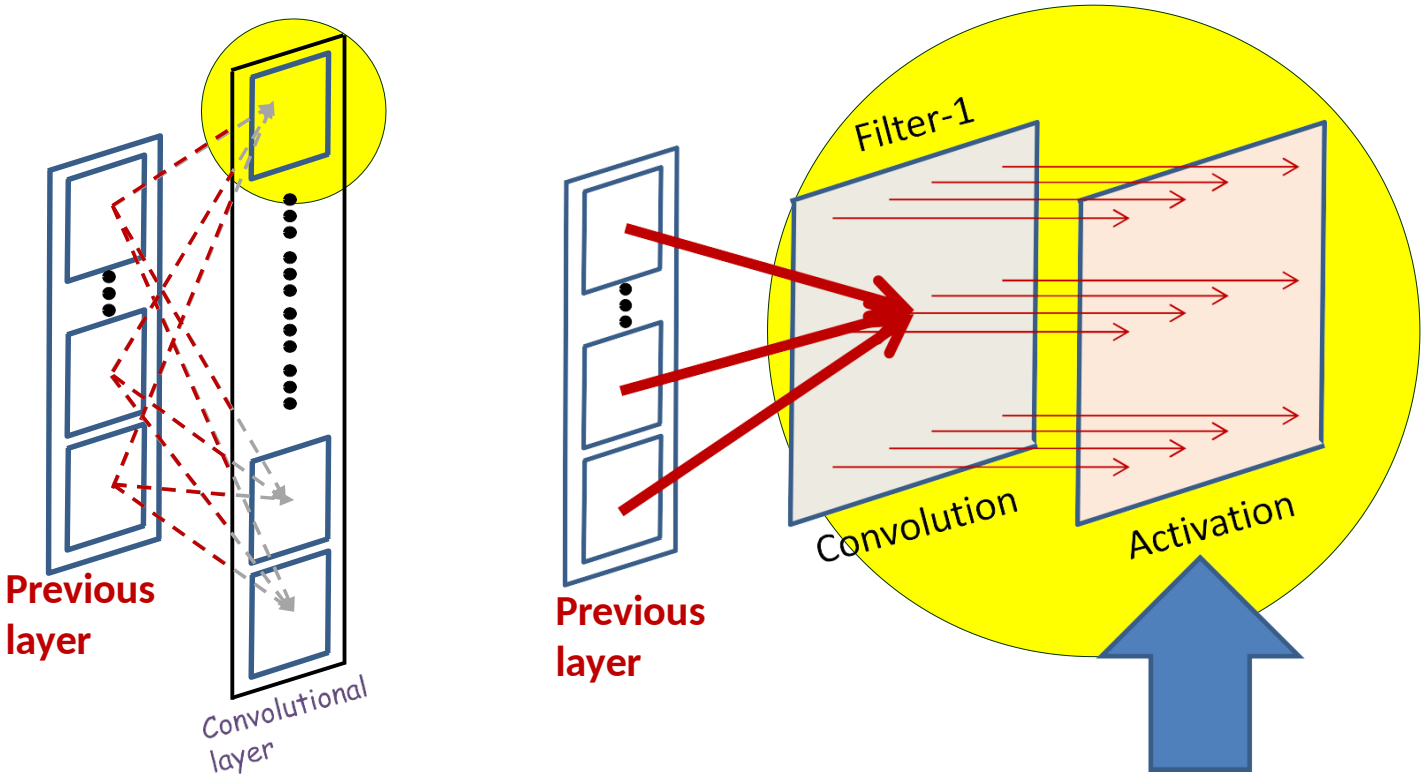

- Zero-pad the input
  - Pad the input image/map all around
  - Pad as symmetrically as possible, such that..
  - **For stride 1, the result of the convolution is the same size as the original image**

# The size of the convolution



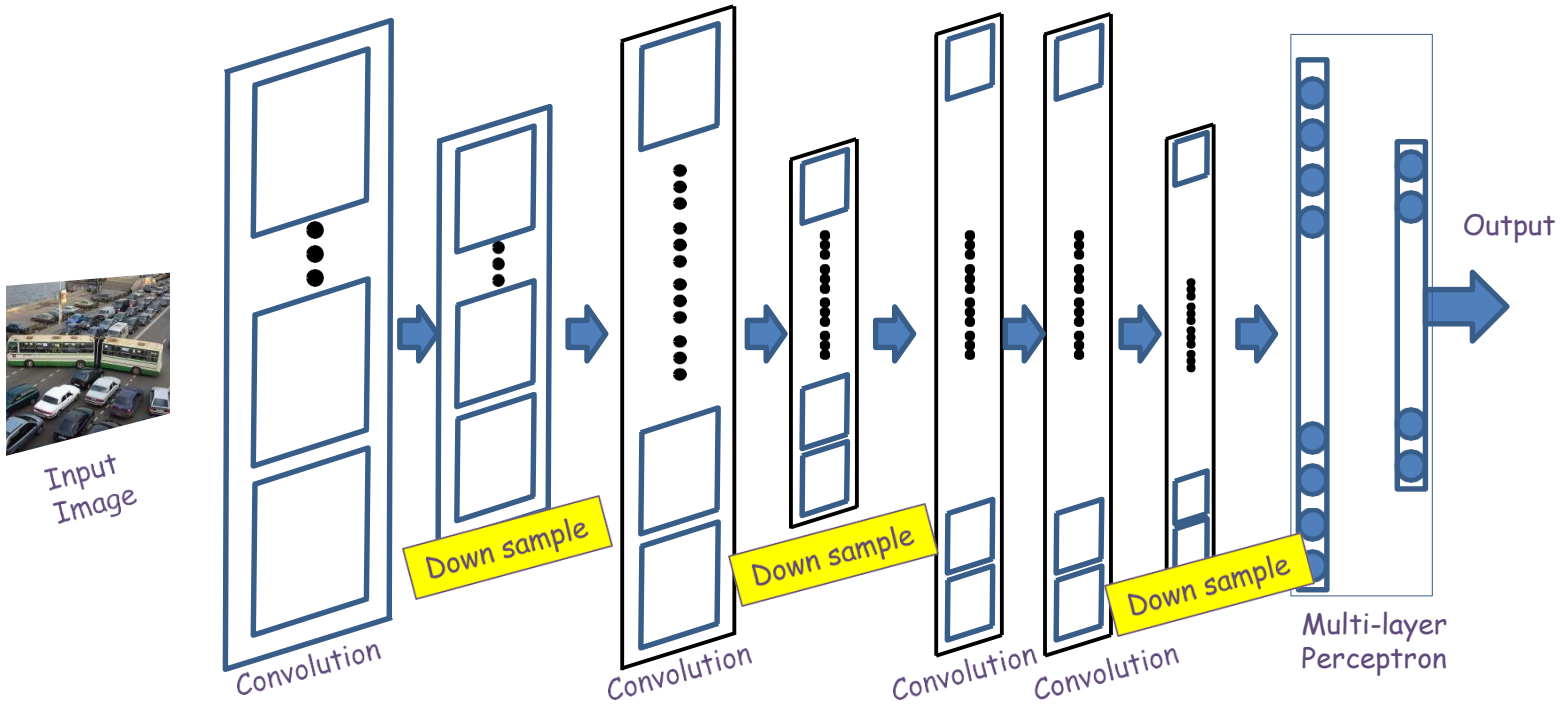

- Image size: n
- Filter: f
- Stride: s
- Padding: p
- Output size (each side) = [(n+2p-f)/s] + 1

# A convolutional layer



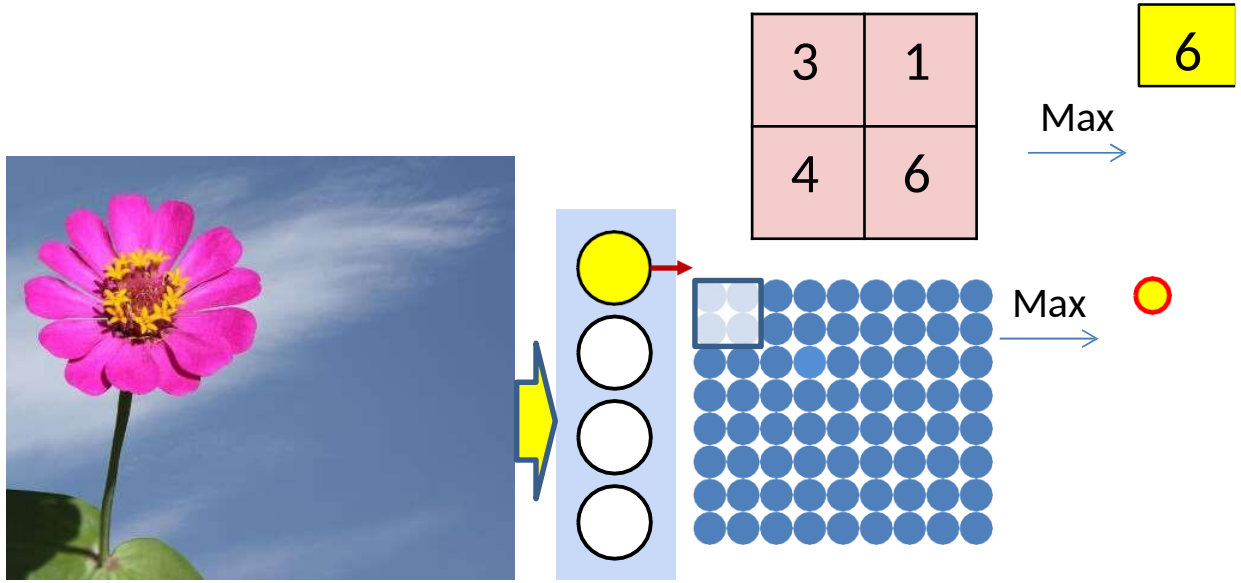

- The convolution operation results in an affine map
- An *Activation* is finally applied to every entry in the map

# The other component
# Downsampling/Pooling



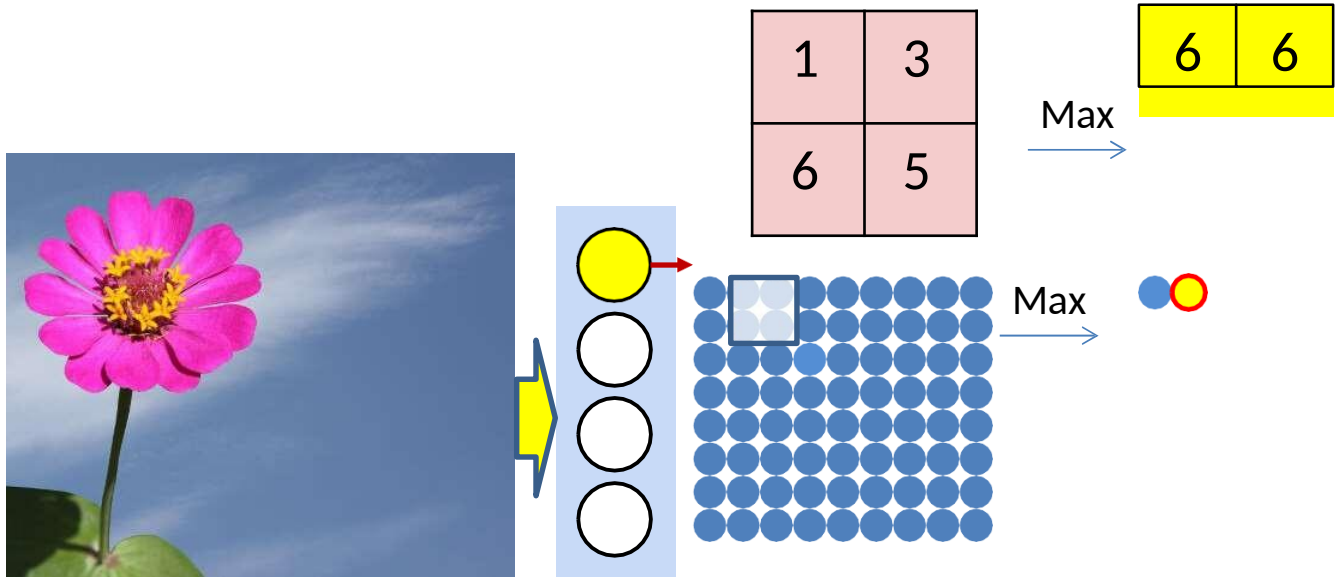

- Convolution (and activation) layers are followed intermittently by "downsampling with pooling" layers
  - Typically (but not always) "max" pooling
  - Often, they alternate with convolution, though this is not necessary

# Max pooling



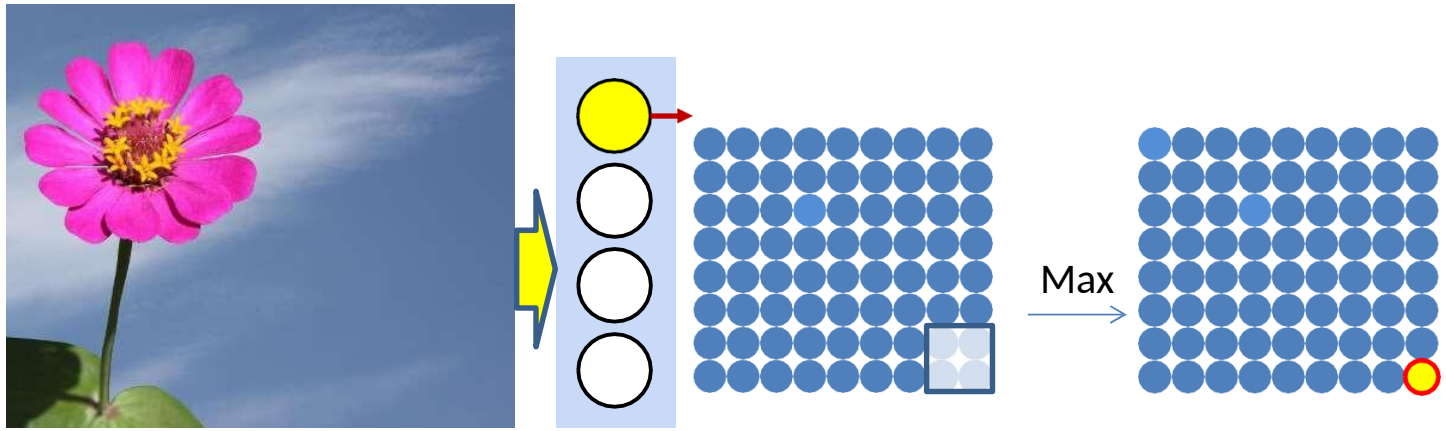

- Max pooling selects the largest from a pool of elements
- Pooling is performed by “scanning” the input

# Recall: Max pooling



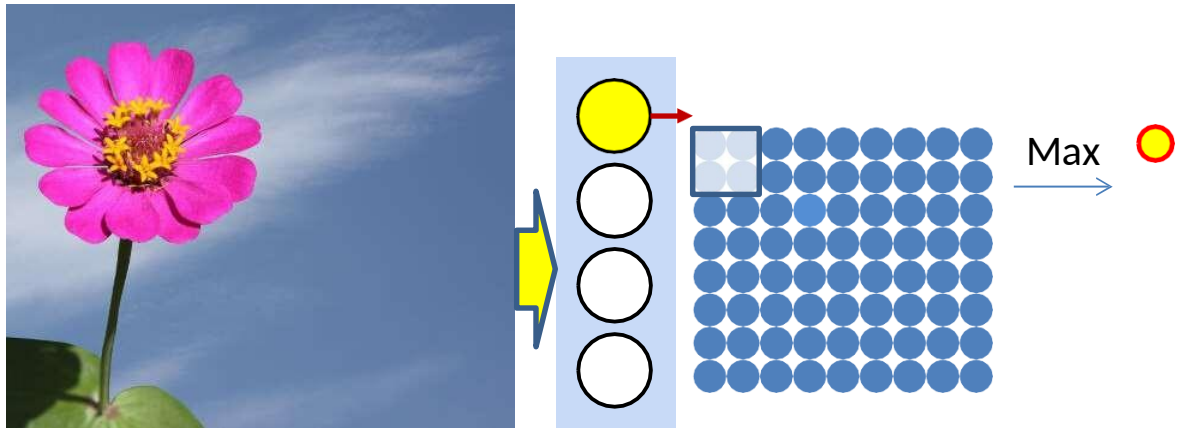

- Max pooling selects the largest from a pool of elements
- Pooling is performed by “scanning” the input

# Recall: Max pooling



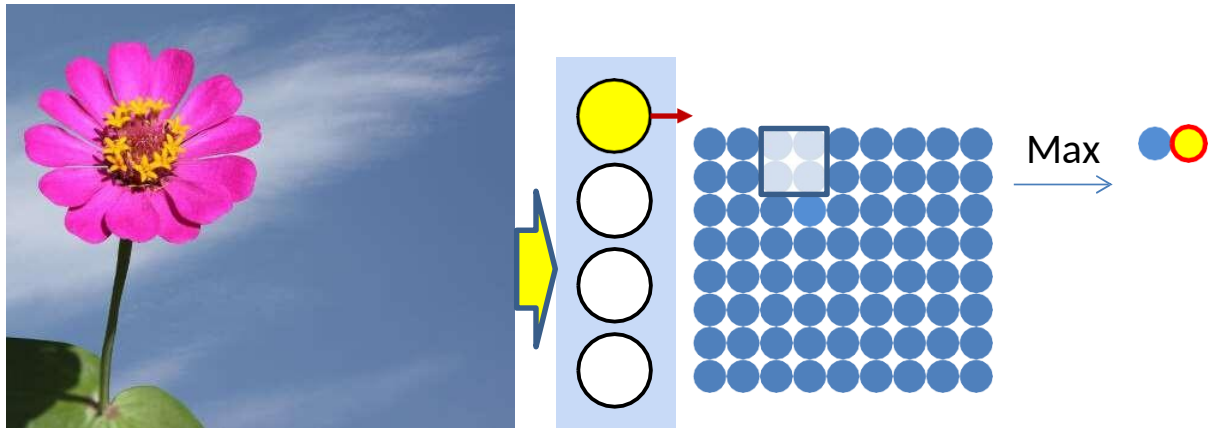

- Max pooling scans with a stride of 1 confer jitter-robustness, but do not constitute downsampling
- Downsampling requires a stride greater than 1

# Downsampling requires Stride>1



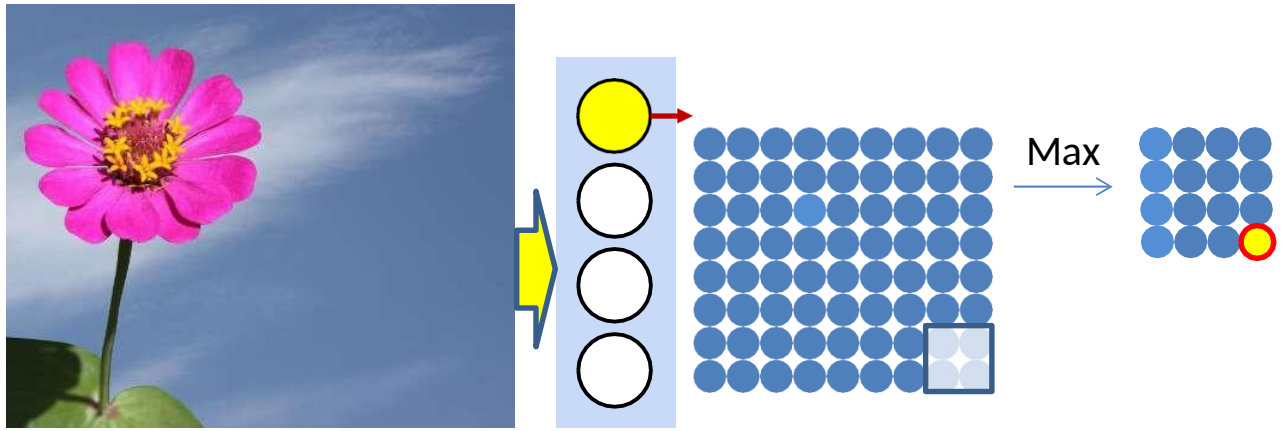

- The “max pooling” operation with “stride” greater than 1 results in an output smaller than the input
  - One output per stride
  - The output is “downsampled”

# Downsampling requires Stride>1




- The “max pooling” operation with “stride” greater than 1 results in an output smaller than the input
  - One output per stride
  - The output is “downsampled”

# Downsampling requires Stride>1

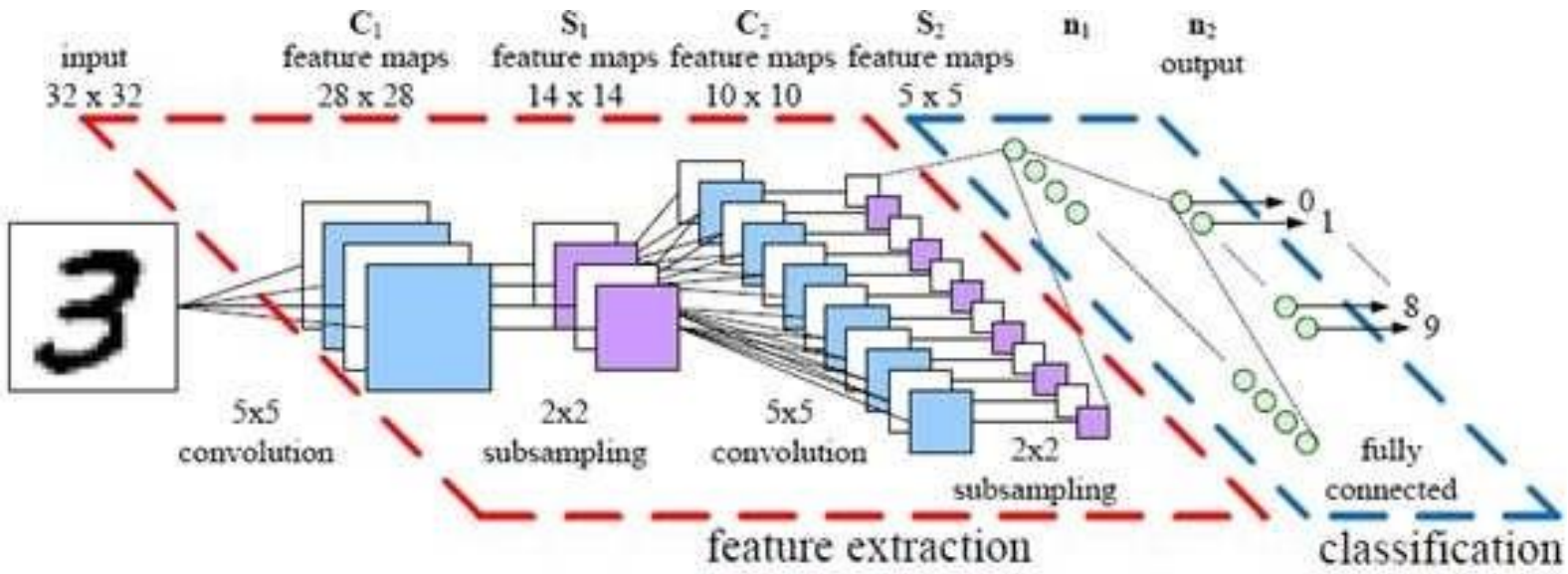



- The “max pooling” operation with “stride” greater than 1 results in an output smaller than the input
  - One output per stride
  - The output is “downsampled”

# Convolutional Neural Networks

- Input: 1 or 3 images
  - Grey scale or color
  - Will assume color to be generic

# Digit classification



# Training

- Training is as in the case of the regular MLP
  - The *only* difference is in the *structure* of the network
- **Training examples of (Image, class) are provided**
- Define a divergence between the desired output and true output of the network in response to any input
- **Network parameters are trained through variants of gradient descent**
- **Gradients are computed through backpropagation**