THE UNIVERSITY OF MELBOURNE

Convolutional Neural Networks (Part 1)

COMP90086 Computer Vision

Qiuhong Ke

Before we start…

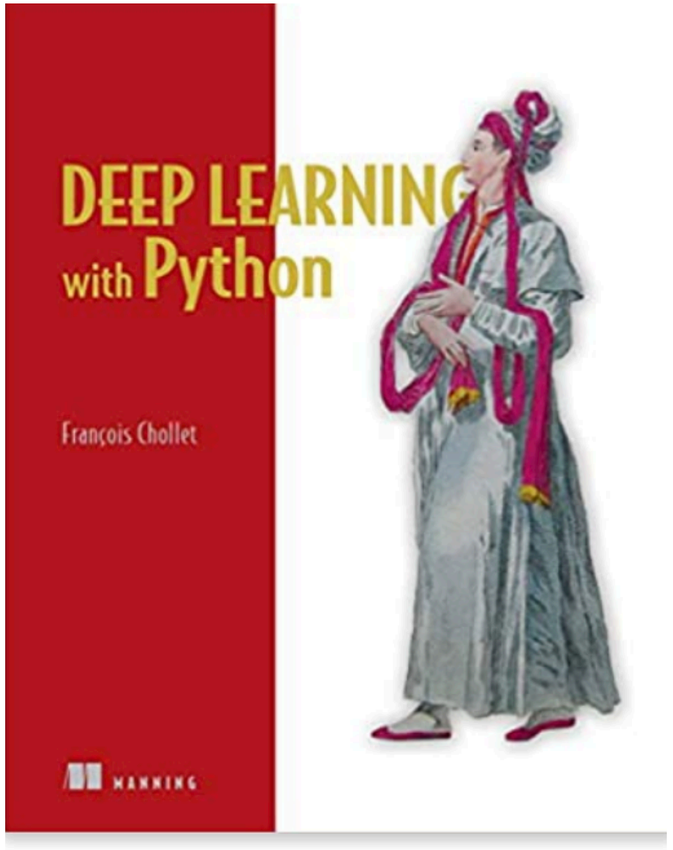
About me

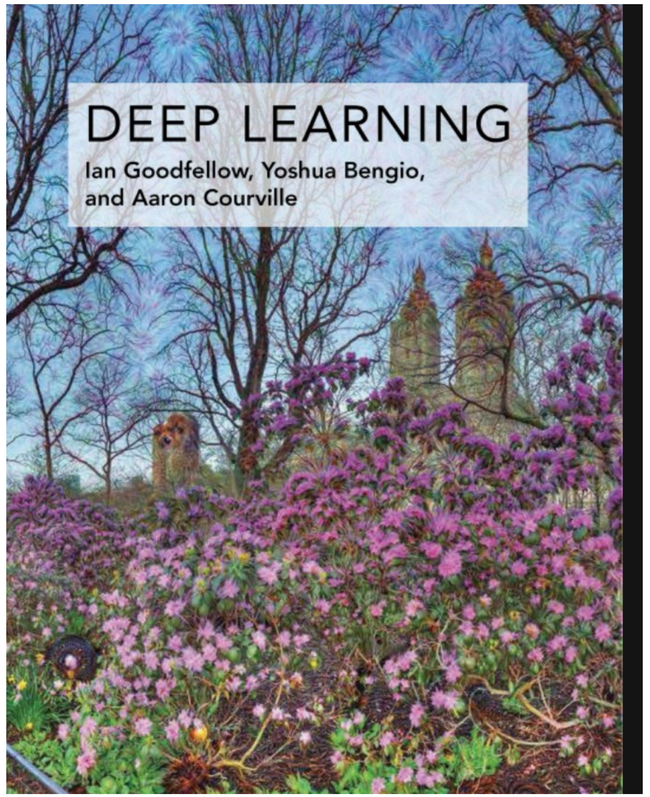
- 2015.02-2018.04: PhD in UWA
- 2018.05-2019.12: Post-doc in MPII
- From 2020.01: Lecturer in UniMelb
- Research: Action recognition and prediction
- Contact: ~~qiuhong.ke@unimelb.edu.com;~~
  qiuhong.ke@unimelb.edu.au

Before we start

Books & resources

- Books:
  - Deep Learning with Python, by Francois Chollet, available in unimelb library
  - Deep Learning, by Ian Goodfellow and Yoshua Bengio and Aaron Courville, https://www.deeplearningbook.or
- Tool: Keras, https://keras.io/





Outline

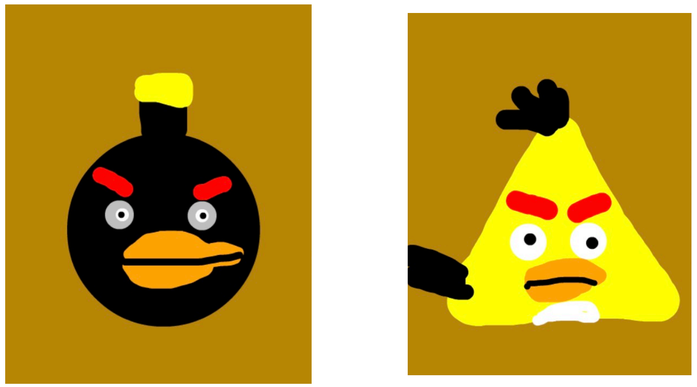
- Introduction on Deep Learning
- Neural Networks: Recap of Multi-layer Perceptron
- Convolutional Neural Networks

Angry birds

VS

Source: by Papachan (CC BY)
https://www.sketchport.com/drawing/4524248689278976/bomb-and-chuck

Bird classification



Input → Feature engineering → Classification → Output

Color or
Body shape or
…

Decision Trees
…

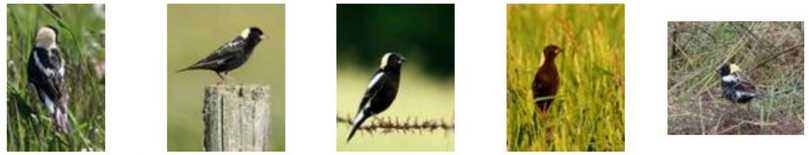
Bird class

Source: by Papachan (CC BY)
https://www.sketchport.com/drawing/4524248689278976/bomb-and-chuck

In real life…

Boat tailed Grackle

Bobolink



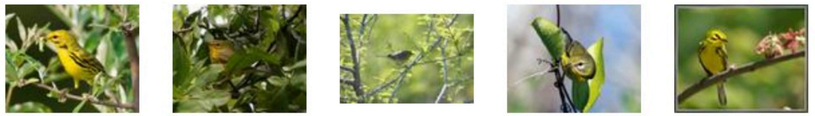
Bohemian Waxwing

Brandt Cormorant

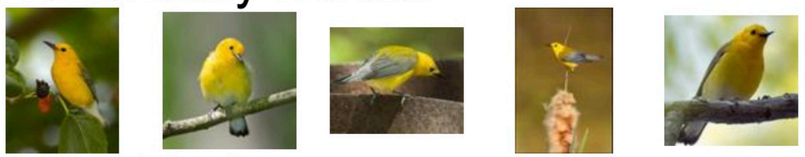
Brewer Blackbird

Pomarine Jaeger

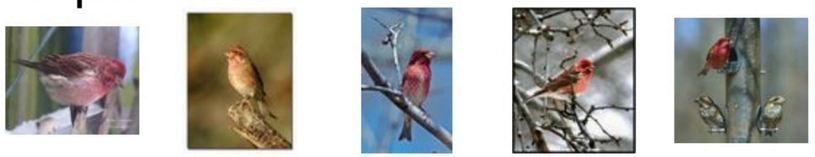
Prairie Warbler



Prothonotary Warbler



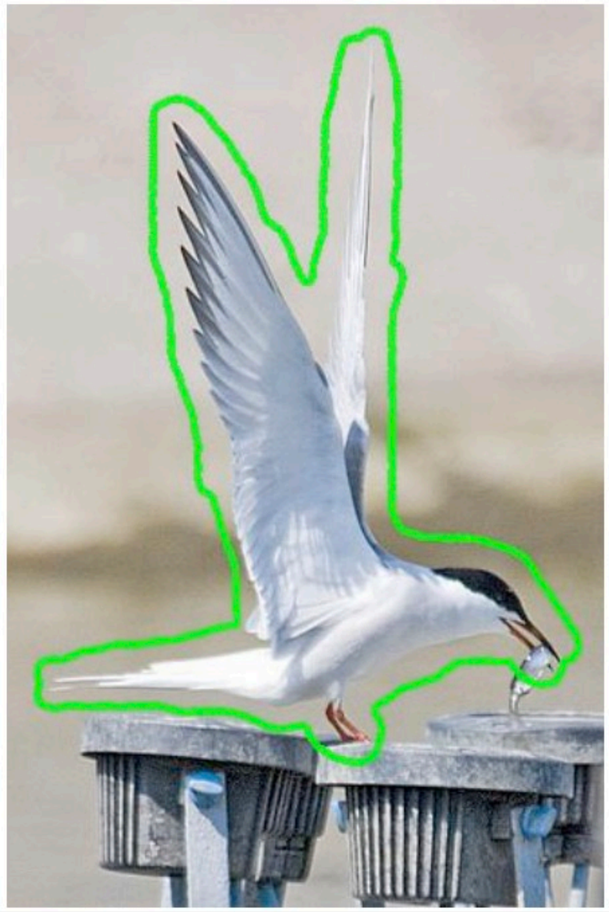
Purple Finch



Red bellied Woodpecker

Source: Welinder, Peter, et al. "Caltech-UCSD birds 200." (2010).

More features for real birds…



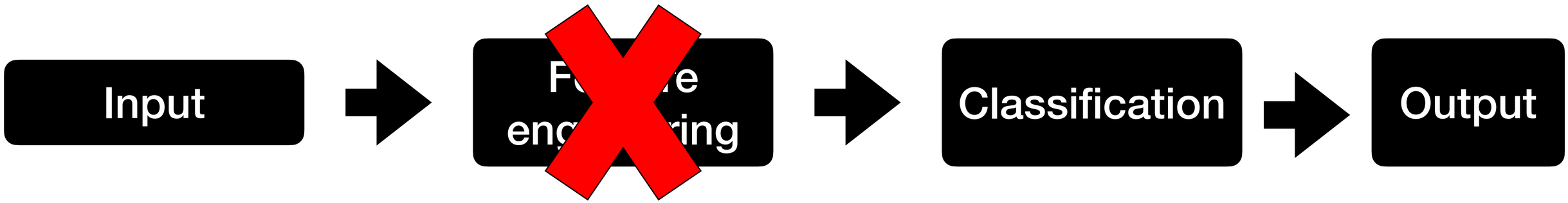
forehead_color	**black**	**black**	**black**
breast_pattern	**solid**	**solid**	**solid**
breast_color	**white**	**white**	**white**
head_pattern	**plain**	**capped**	plain
back_color	**white**	white	**black**
wing_color	grey/white	**grey**	**white**
leg_color	**orange**	orange	**orange**
size	**medium**	large	**medium**
bill_shape	needle	dagger	**dagger**
wing_shape	**pointed**	**tapered**	**long**
...	...	...	...
primary_color	white	white	white

Problems?

- time-consuming
- Require expert knowledge
- Require feature selection

Source: Welinder, Peter, et al. "Caltech-UCSD birds 200." (2010).

‘REAL Bird’ classification

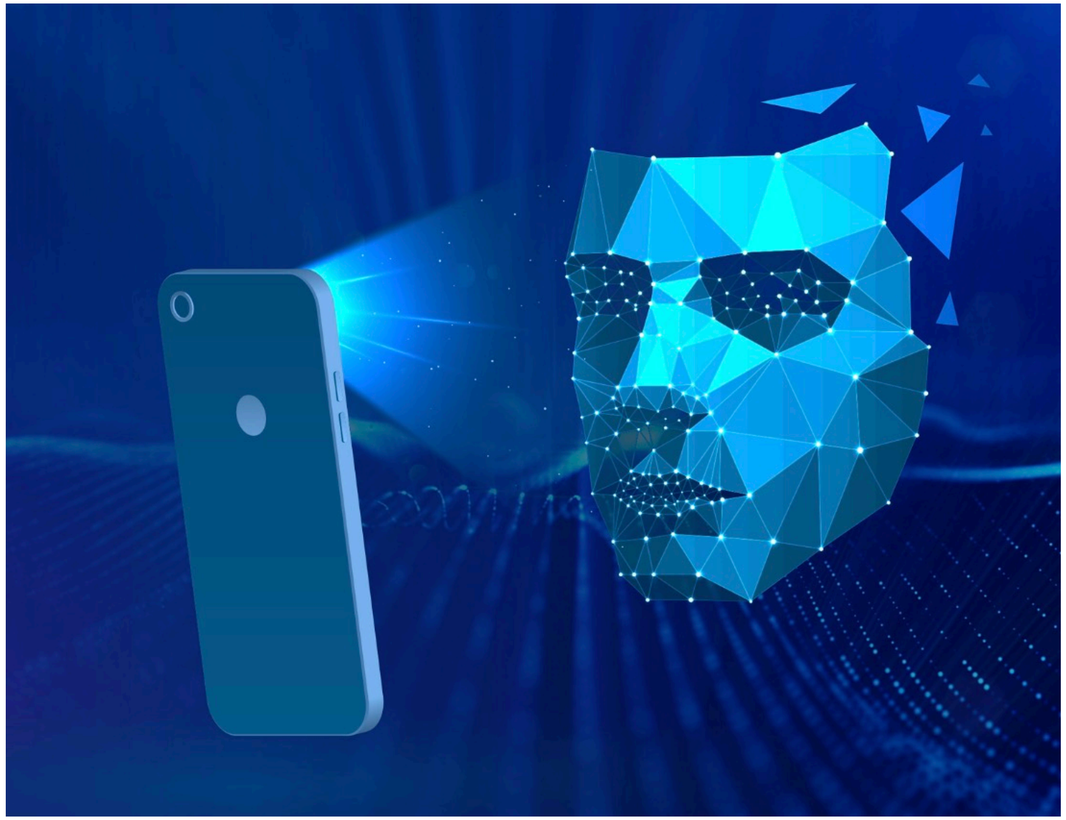


Use deep learning models!

Bird class

Deep learning is everywhere

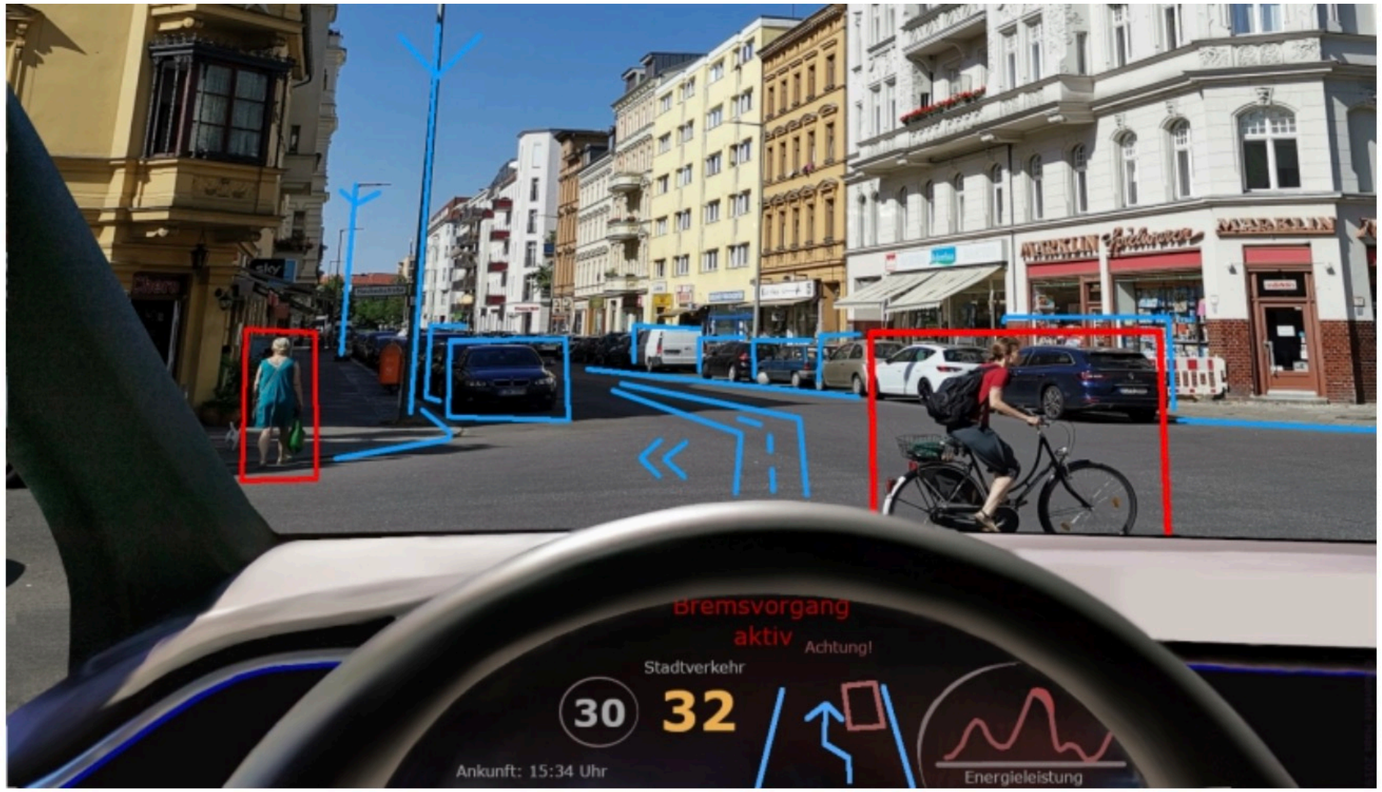
Face recognition



By Mike MacKenzie
https://www.flickr.com/photos/mikemacmarketing/30188201497

Deep learning is everywhere

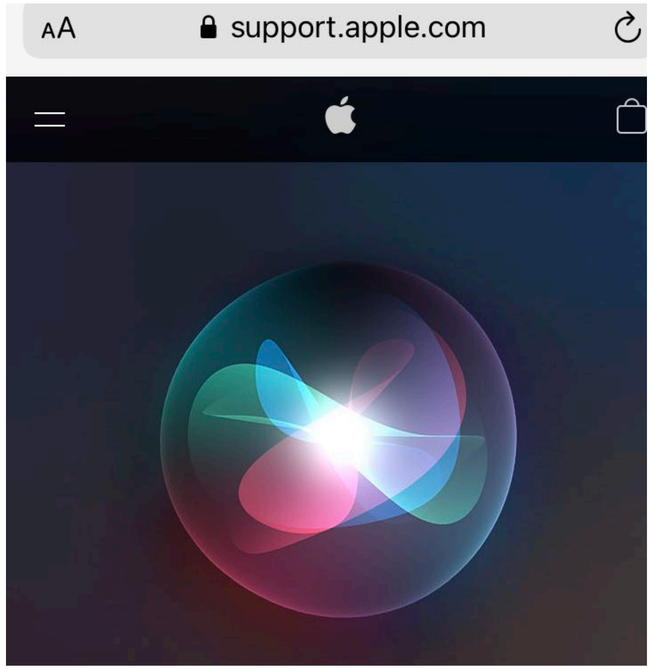
Self driving cars



Source: https://usa.streetsblog.org/2020/12/15/was-2020-the-year-driverless-cars-became-inevitable/

Deep learning is everywhere

Virtual Assistants:
- Amazon Alexa
- Cortana
- Google Assistant
- Siri



Deep learning is everywhere

Chatbots: AI application to chat online (to solve customer problems)

by mohamed hassan
source: https://www.stockvault.net/photo/259374/chatbot-technology

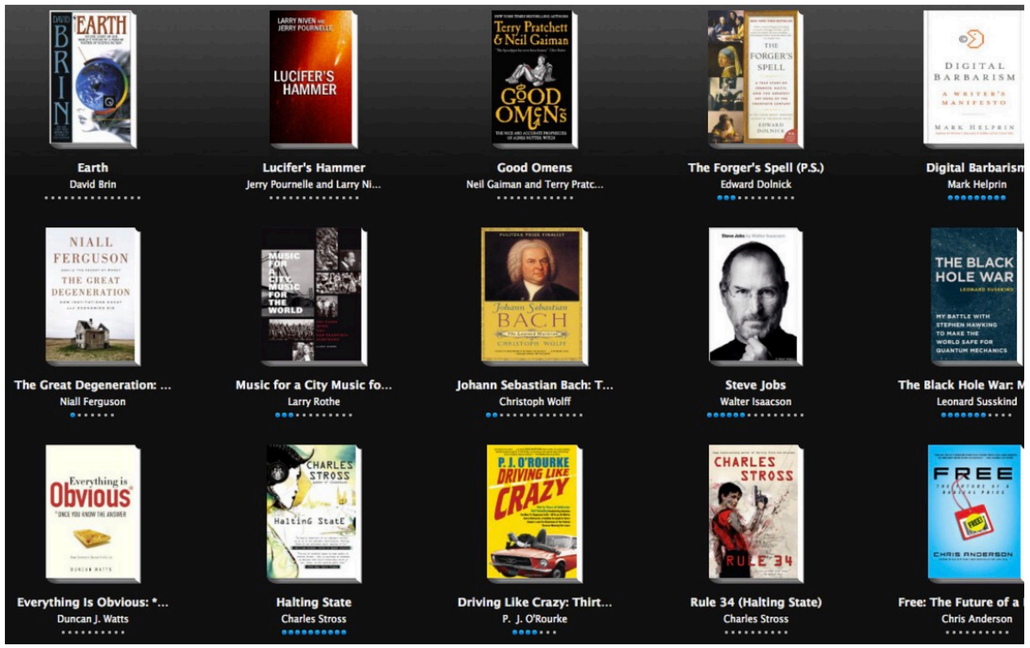
Deep learning is everywhere

Healthcare: medical research,
medicine discovery, and disease diagnosis

by Xinh Studio
source:https://thenounproject.com/term/healthcare/1243846/

Deep learning is everywhere

Recommendation:
books, videos…

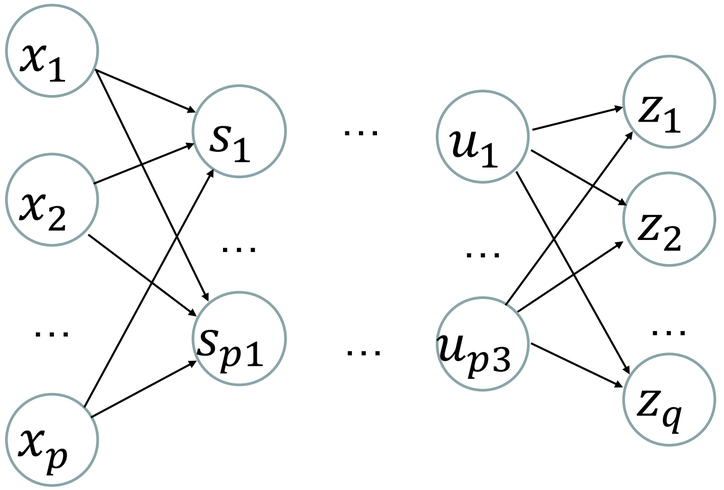


by Jose Camões Silva
https://www.flickr.com/photos/josecamoessilva/9115095115

What is Deep learning

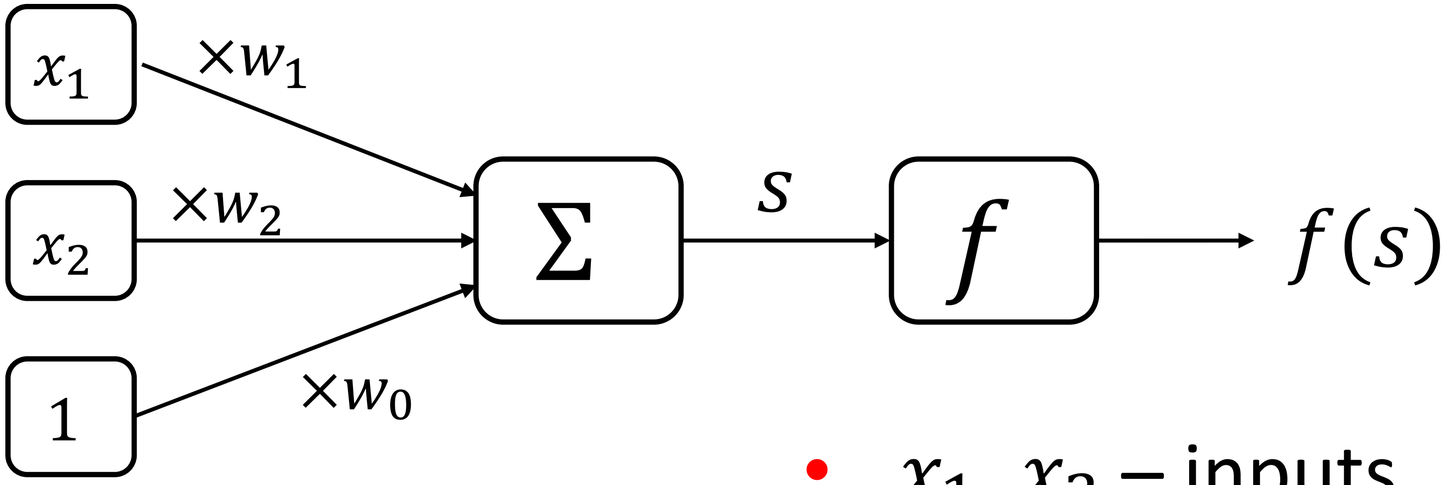
Input → Feature learning + classification → Output

Deep neural network



many hidden layers

Perceptron



$$s = \sum_{i=0}^{m} x_i w_i$$

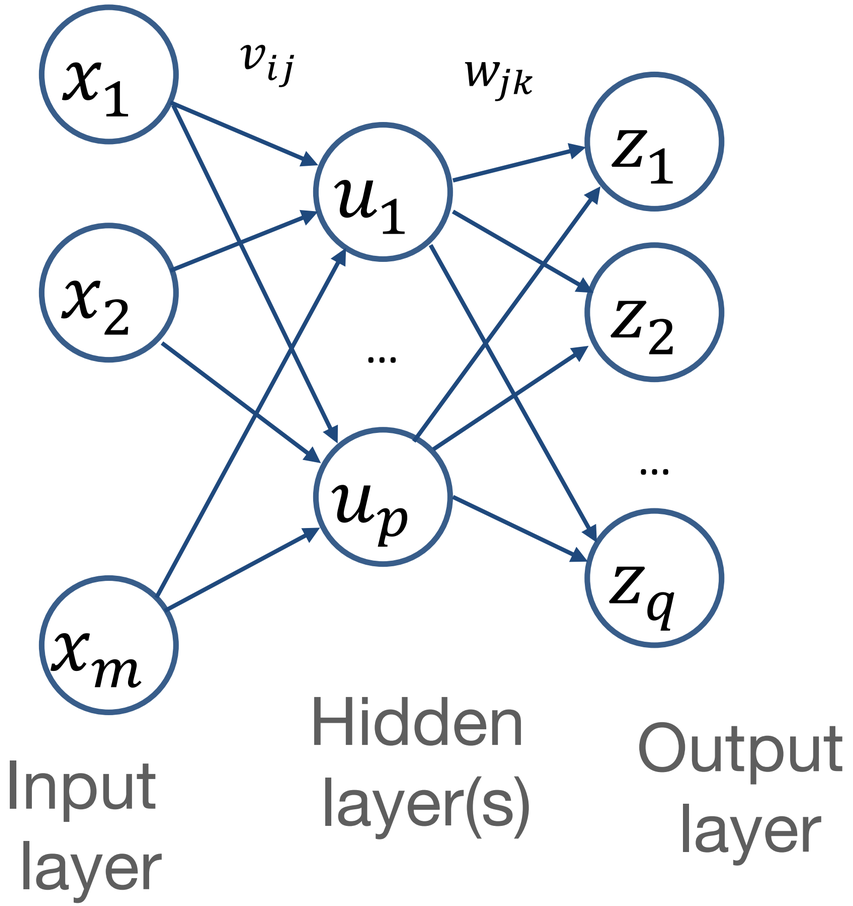
- x_1, x_2 – inputs
- w_1, w_2 – synaptic weights
- w_0 – bias weight
- f – activation function

Multi-layer perceptron: function composition

$$u_j = g(r_j)$$

$$r_j = \sum_{i=0}^{m} x_i v_{ij}$$

g(x): activation function

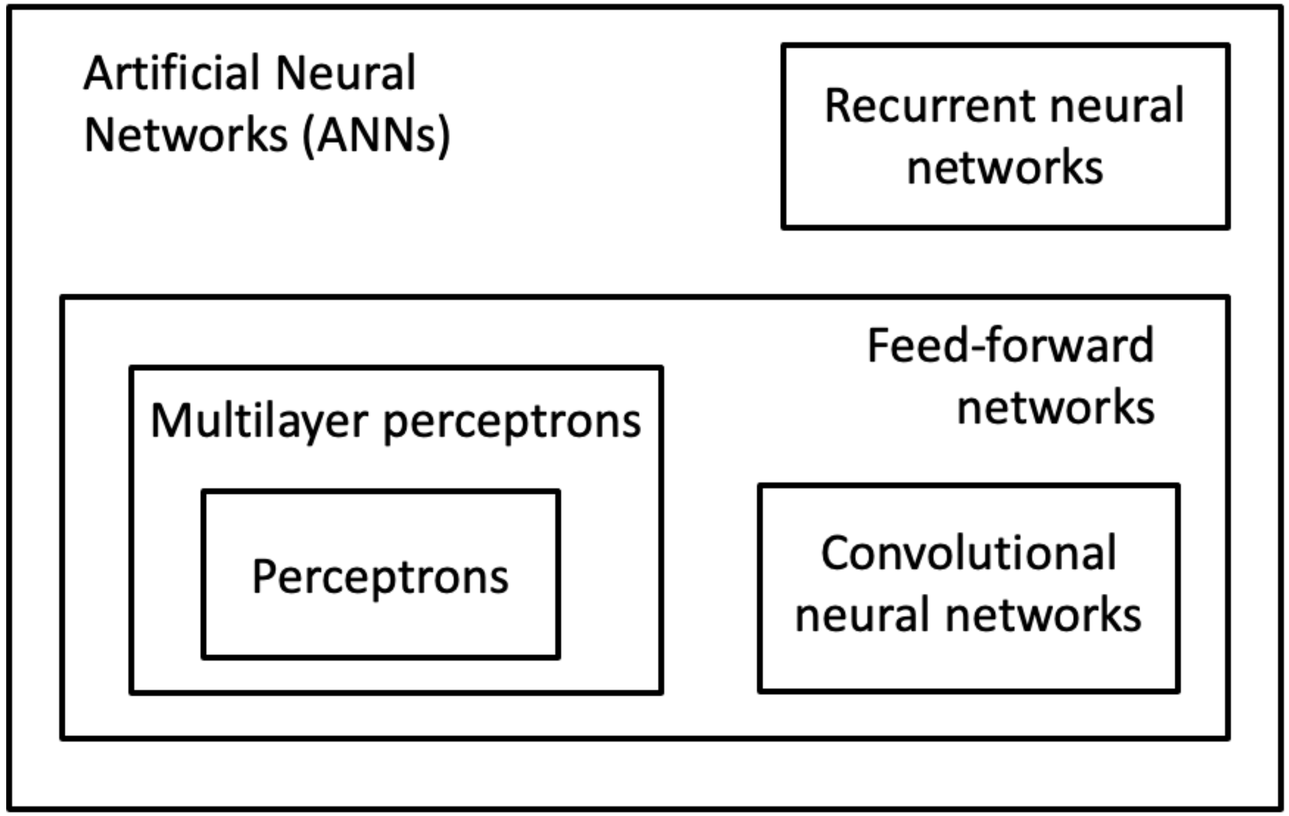


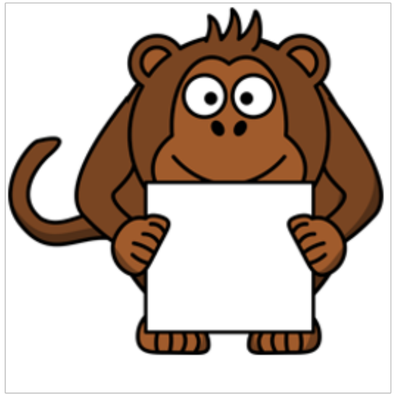
$$z_k = h(s_k)$$

$$s_k = \sum_{j=0}^{p} u_j w_{jk}$$

h(x): activation function

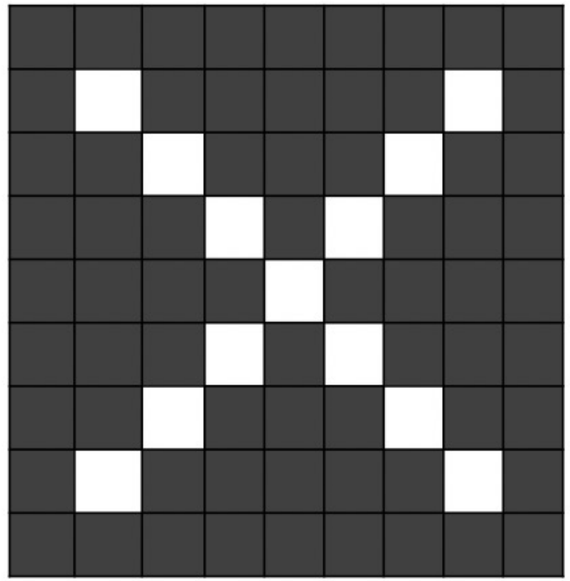
Animals in the zoo

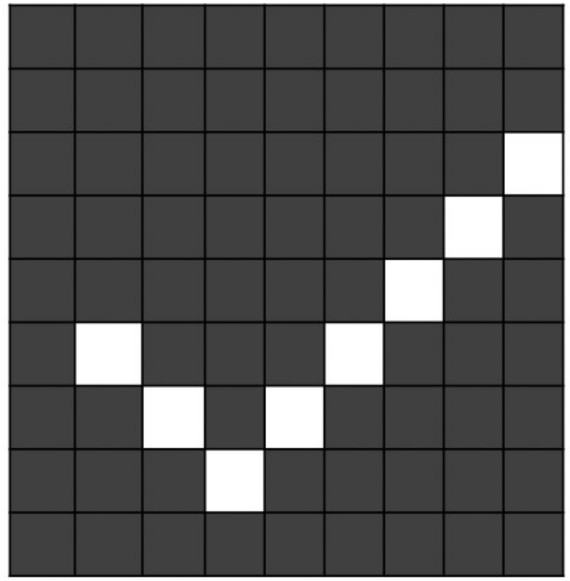




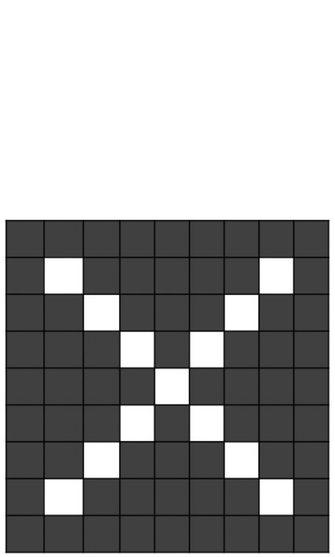
art: OpenClipartVectors at pixabay.com (CC0)


vs



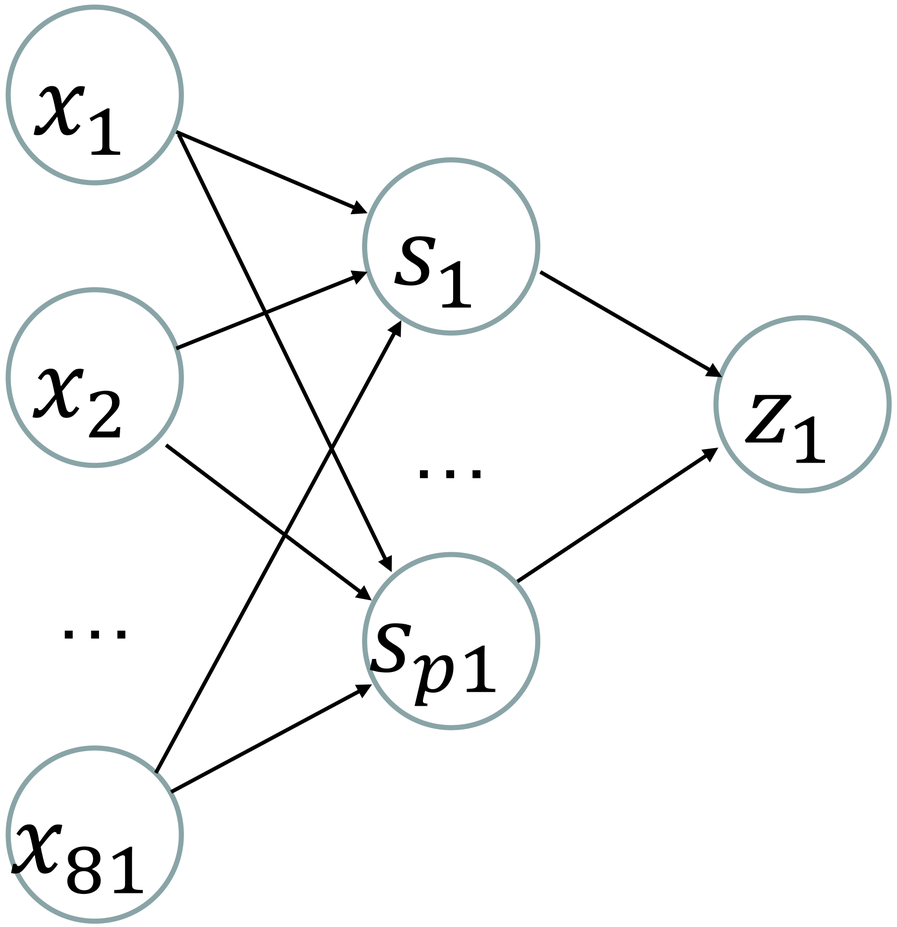


Multi-layer perceptron: A fully connected network

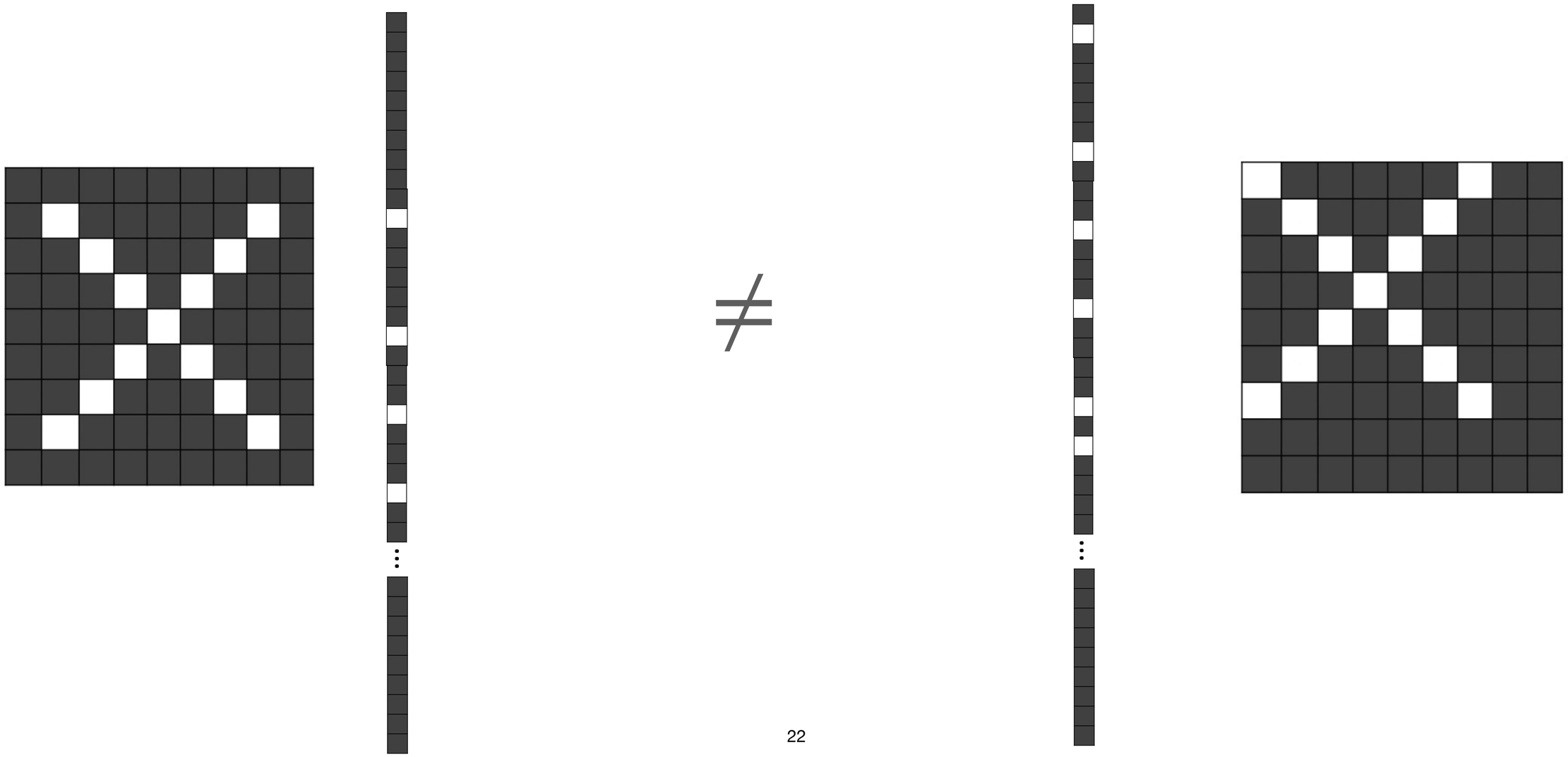


9x9

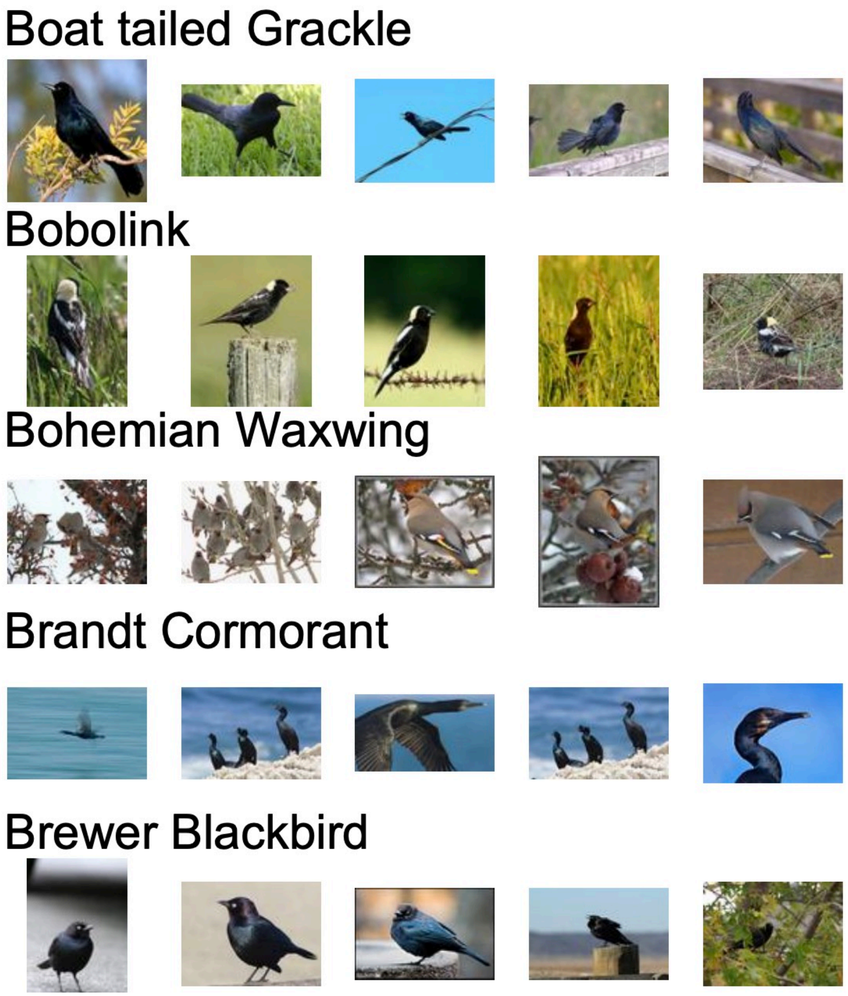
81x1

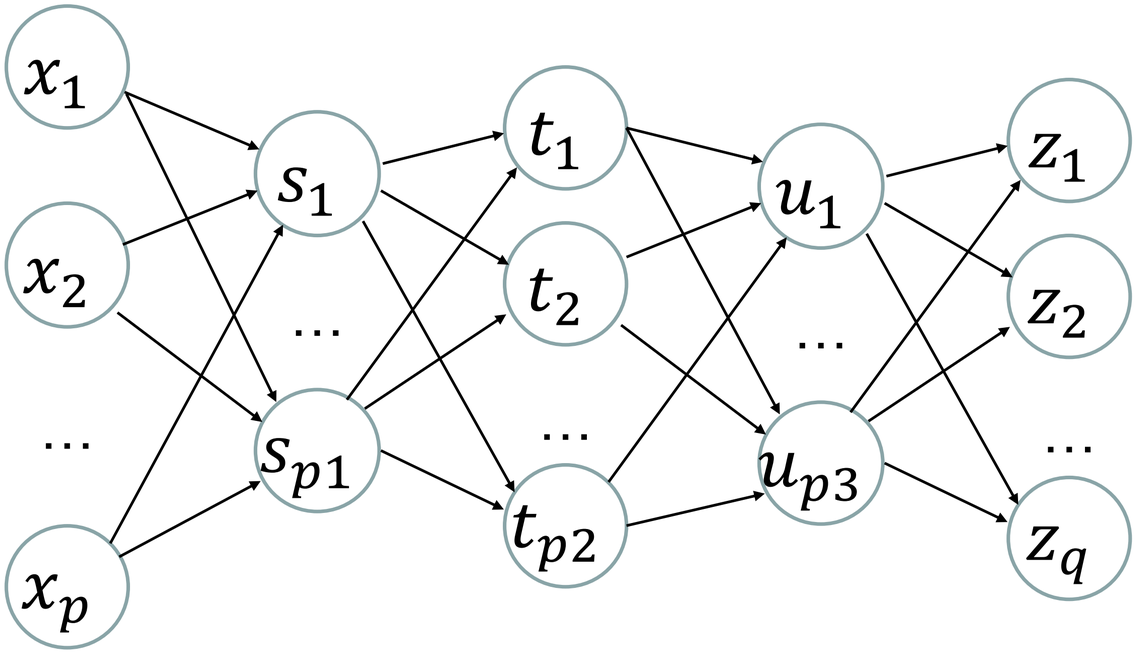


Multi-layer perceptron: Not spatial invariant



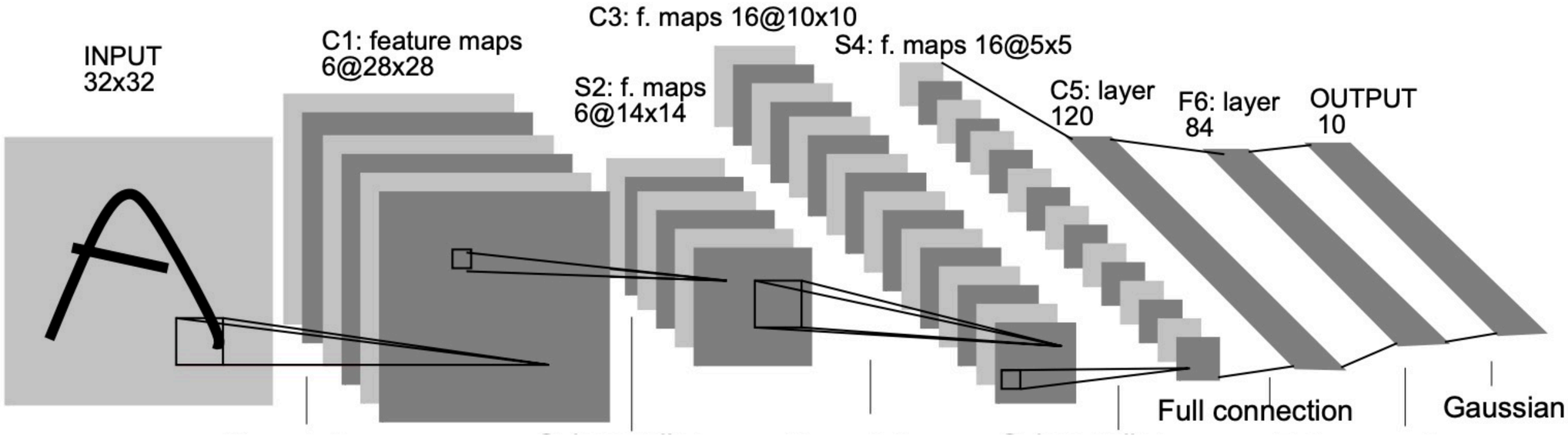
Multi-layer perceptron: cannot build deep neural network





Source: Welinder, Peter, et al. "Caltech-UCSD birds 200." (2010).

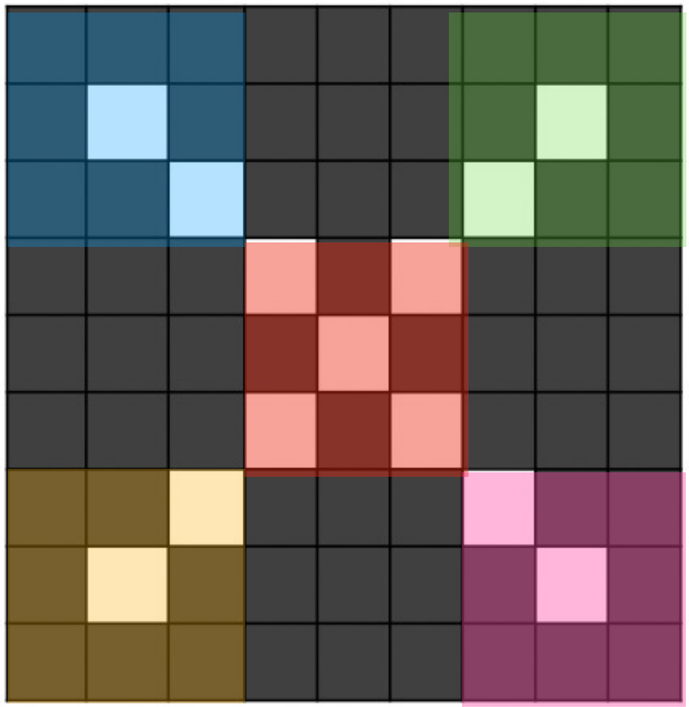
Convolutional, Max-Pooling, and Fully Connected (FC) layers

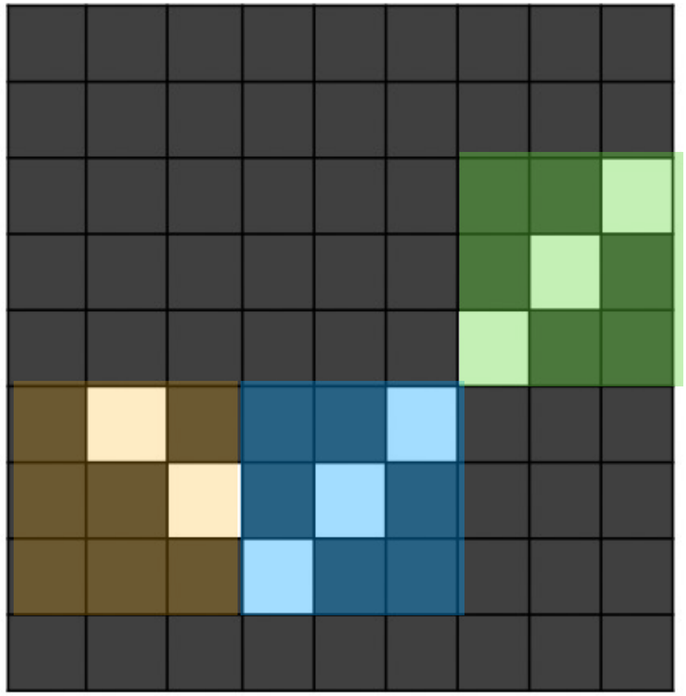


LeCun, Yann, et al. "Gradient-based learning applied to document recognition." *Proceedings of the IEEE* 86.11 (1998): 2278-2324.

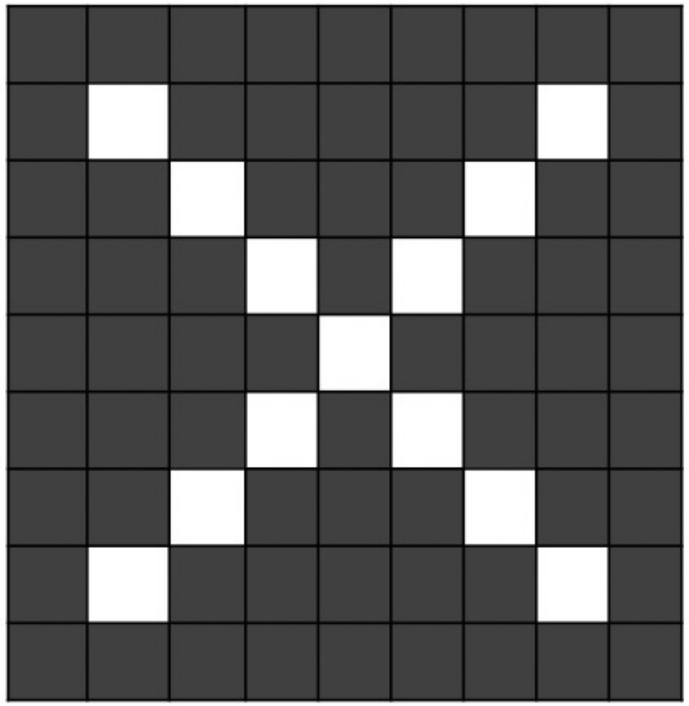
Motivation of convolution

- An image == several patches
- Combine the local features of the patches-> image feature



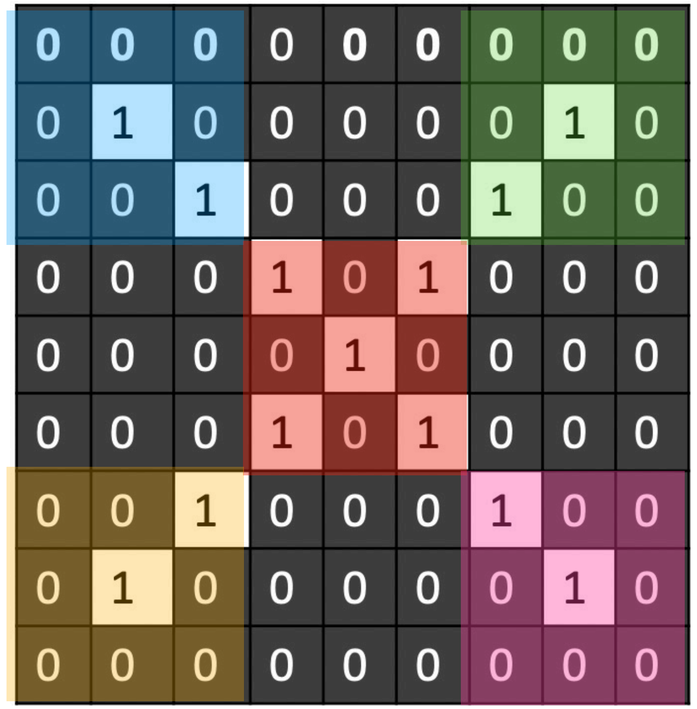


Identify different patterns at local patches



0	**0**	**0**	0	**0**	**0**	**0**	**0**	**0**
0	1	0	0	0	0	0	1	0
0	0	1	0	0	0	1	0	0
0	0	0	1	0	1	0	0	0
0	0	0	0	1	0	0	0	0
0	0	0	1	0	1	0	0	0
0	0	1	0	0	0	1	0	0
0	1	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0

Identify different patterns



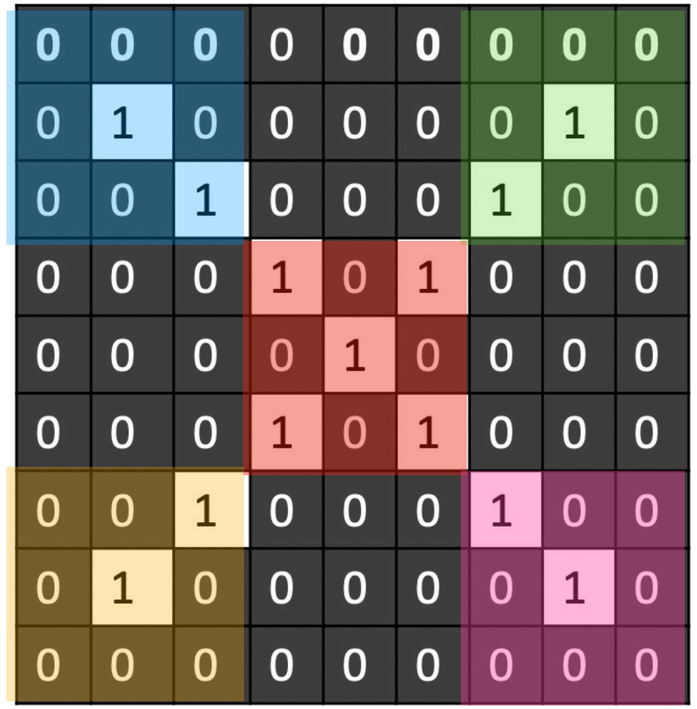
Element-wise multiplication

Sum (0 0 0 / 0 1 0 / 0 0 1) × 0 0 0 / 0 1 0 / 0 0 1) =2

Sum (0 0 0 / 0 1 0 / 1 0 0) × 0 0 0 / 0 1 0 / 0 0 1)=1

Filter (kernel)

Input and kernel have the same pattern: high response



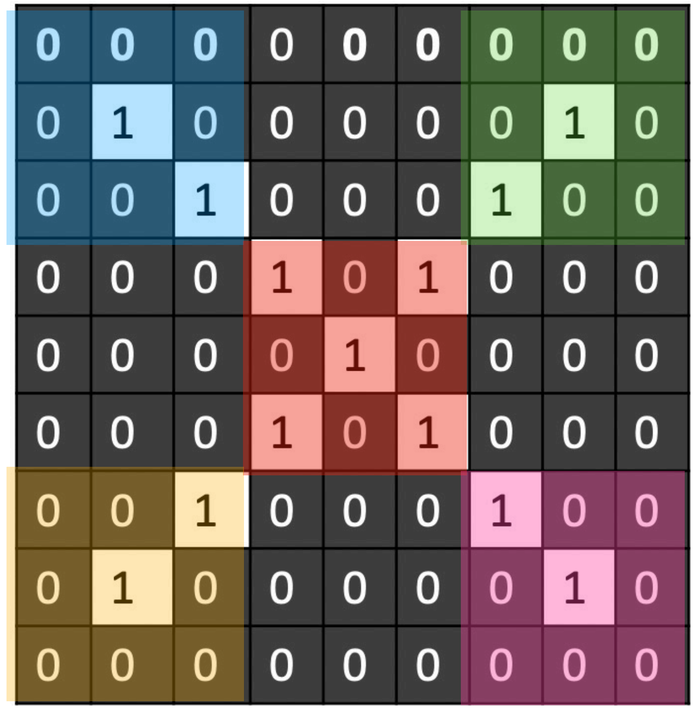
Element-wise multiplication

Sum ($\begin{matrix}0&0&0\\0&1&0\\0&0&1\end{matrix}$ $\times$ $\begin{matrix}0&0&0\\0&1&0\\1&0&0\end{matrix}$) =1

Sum ($\begin{matrix}0&0&0\\0&1&0\\1&0&0\end{matrix}$ $\times$ $\begin{matrix}0&0&0\\0&1&0\\1&0&0\end{matrix}$)=2

Filter (kernel)

Identify different patterns



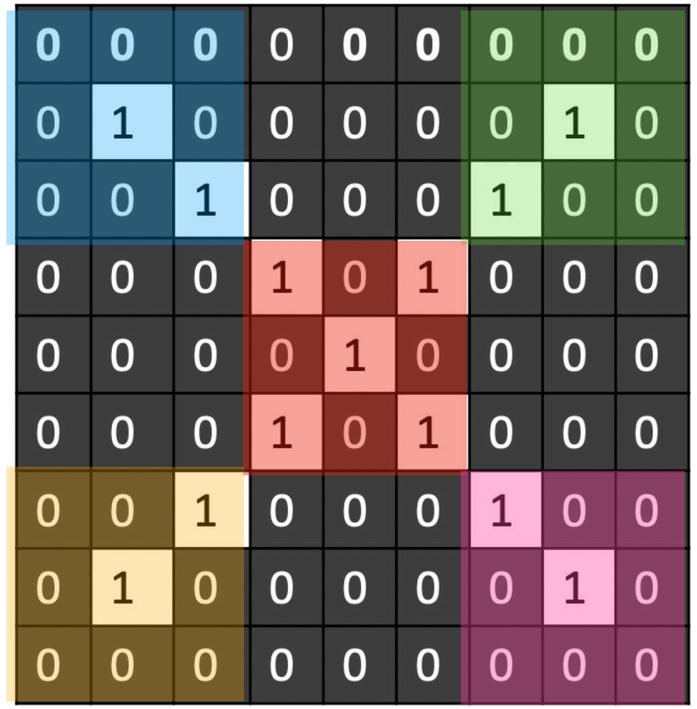
Element-wise multiplication

Sum (0 0 0 / 0 1 0 / 0 0 1) × 0 0 0 / 0 1 0 / 0 0 1) =2

Sum (1 0 1 / 0 1 0 / 1 0 1) × 0 0 0 / 0 1 0 / 0 0 1)=2

Filter (kernel)

Different kernels identify different patterns



Element-wise multiplication

Sum (0 0 0 / 0 1 0 / 0 0 1) × (1 0 1 / 0 1 0 / 1 0 1) =2

Sum (1 0 1 / 0 1 0 / 1 0 1) × (1 0 1 / 0 1 0 / 1 0 1)=5

Filter (kernel)

Convolution on 2D

Use kernel to perform element-wise multiplication and sum for every local patch

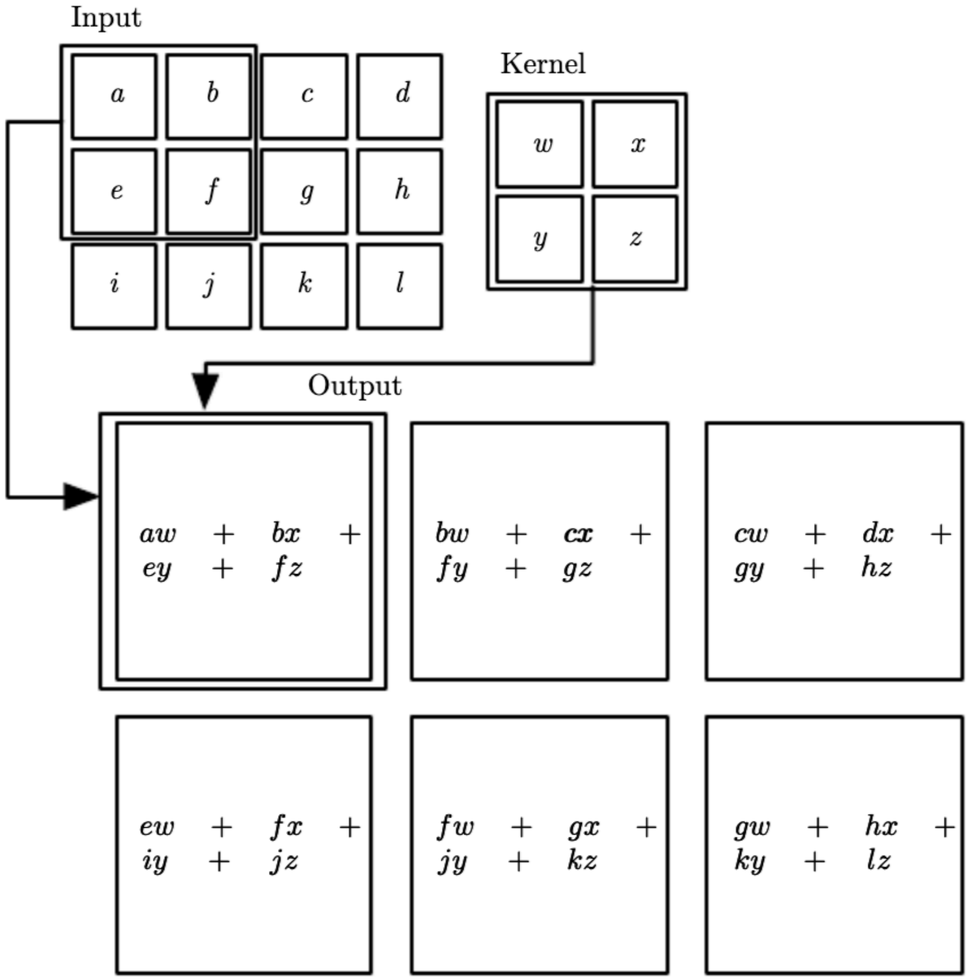
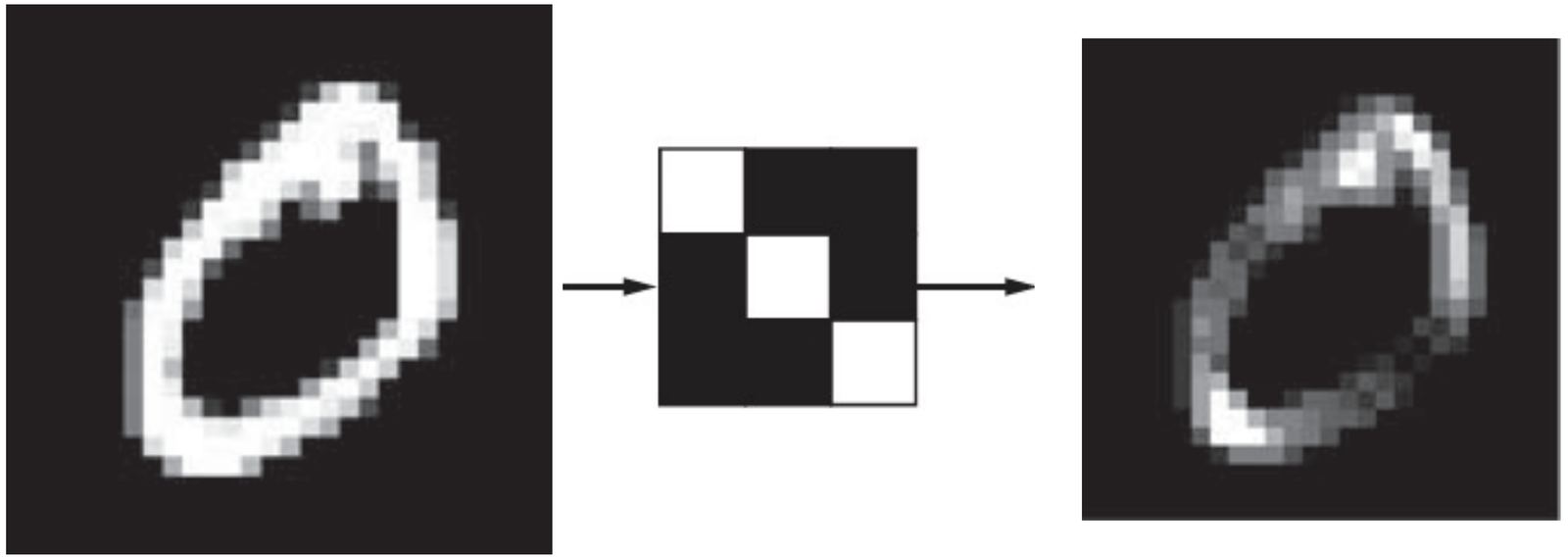


Figure 9.1 in Deep learning by Ian Goodfellow and Yoshua Bengio and Aaron Courville

Response map (Feature map)

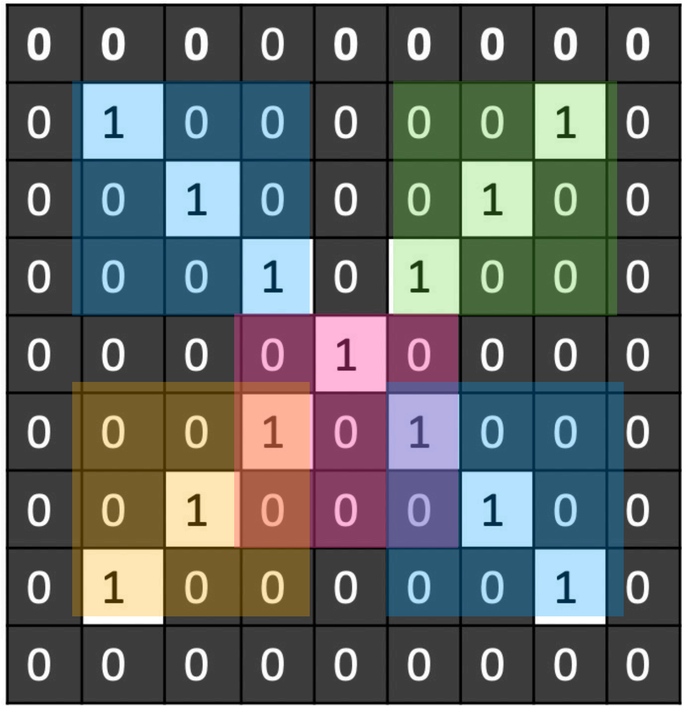


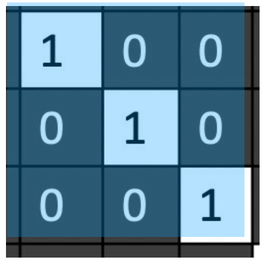
Input kernel

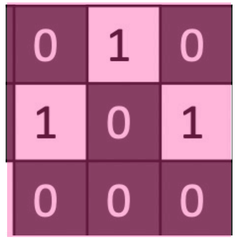
Feature map: 2D map of the presence of a pattern at different locations in an input

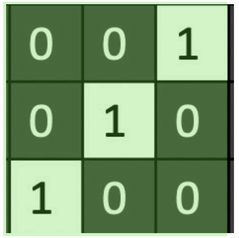
Figure 5.3 in Deep learning with python by Francois Chollet

Different kernels identify different patterns: use multiple filters in each layer





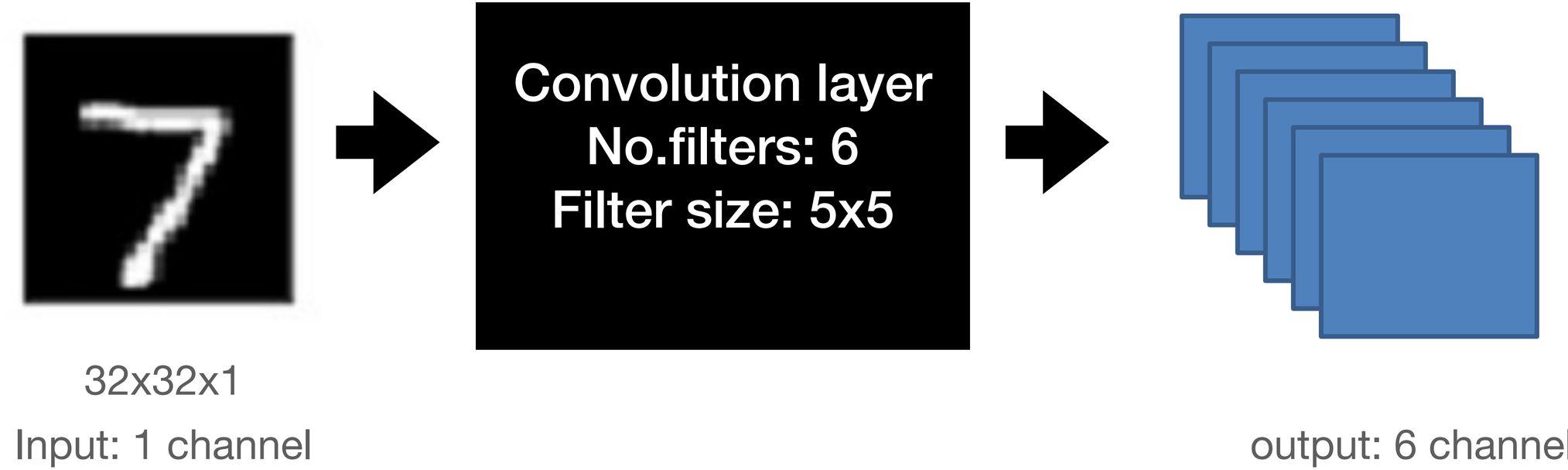




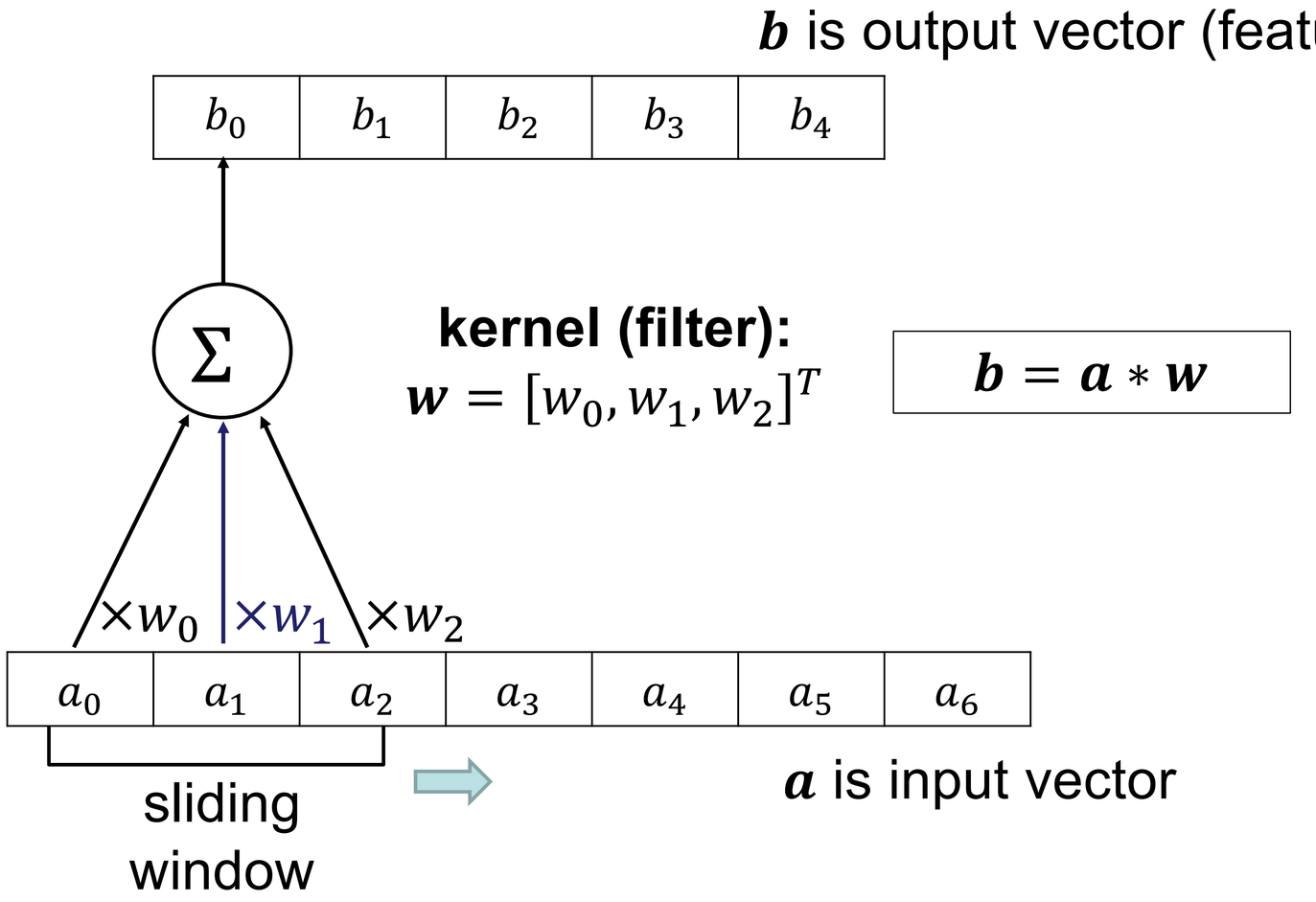
The number of filters decides the number of output feature maps

Two key hyperparameters in Convolution

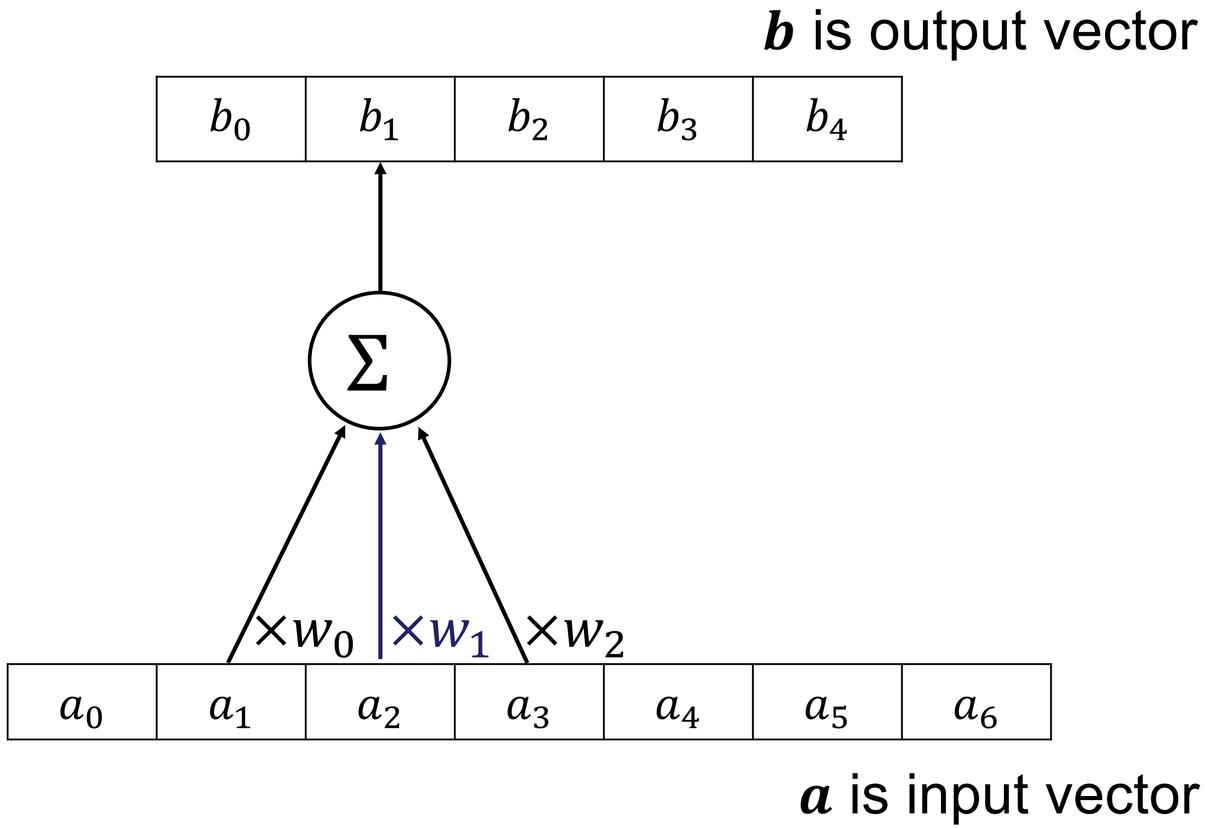
Filter (kernel) size: Size of the patches extracted from the inputs
Number of filters: Depth (channel) of the output feature map



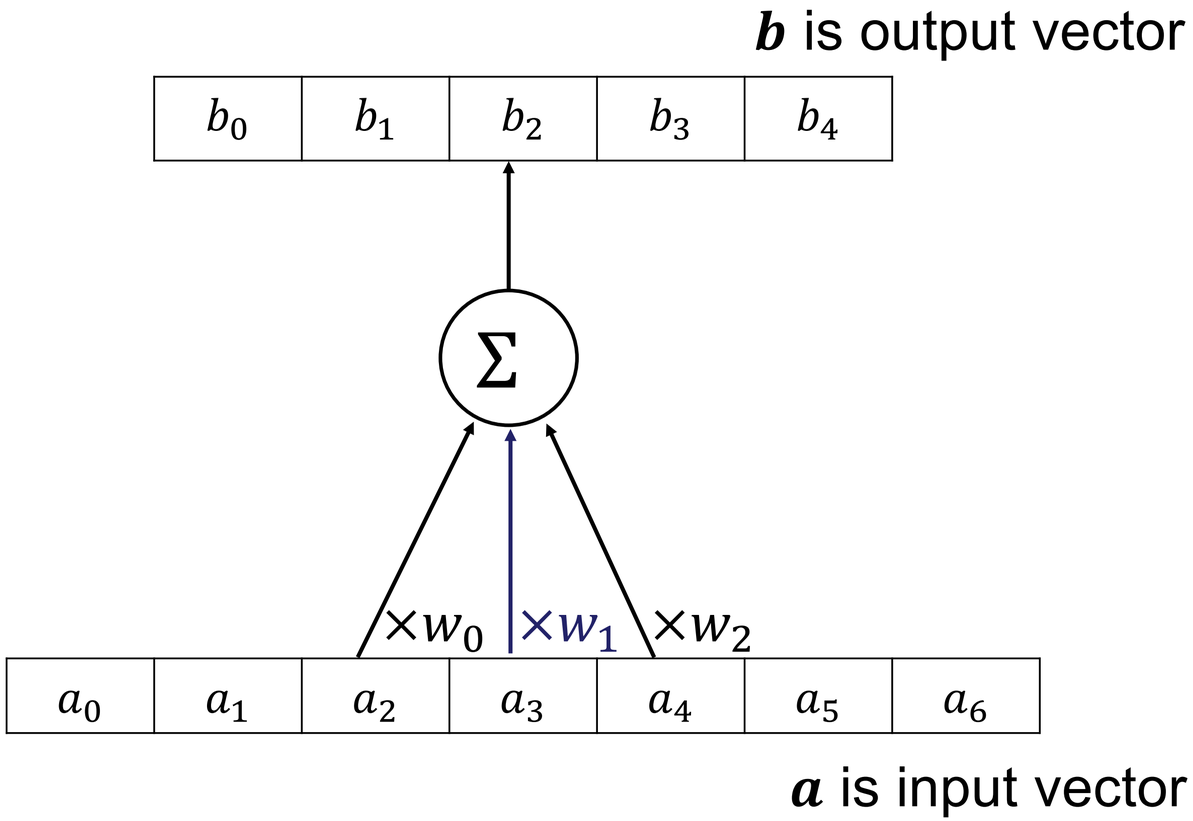
Convolution on 1D



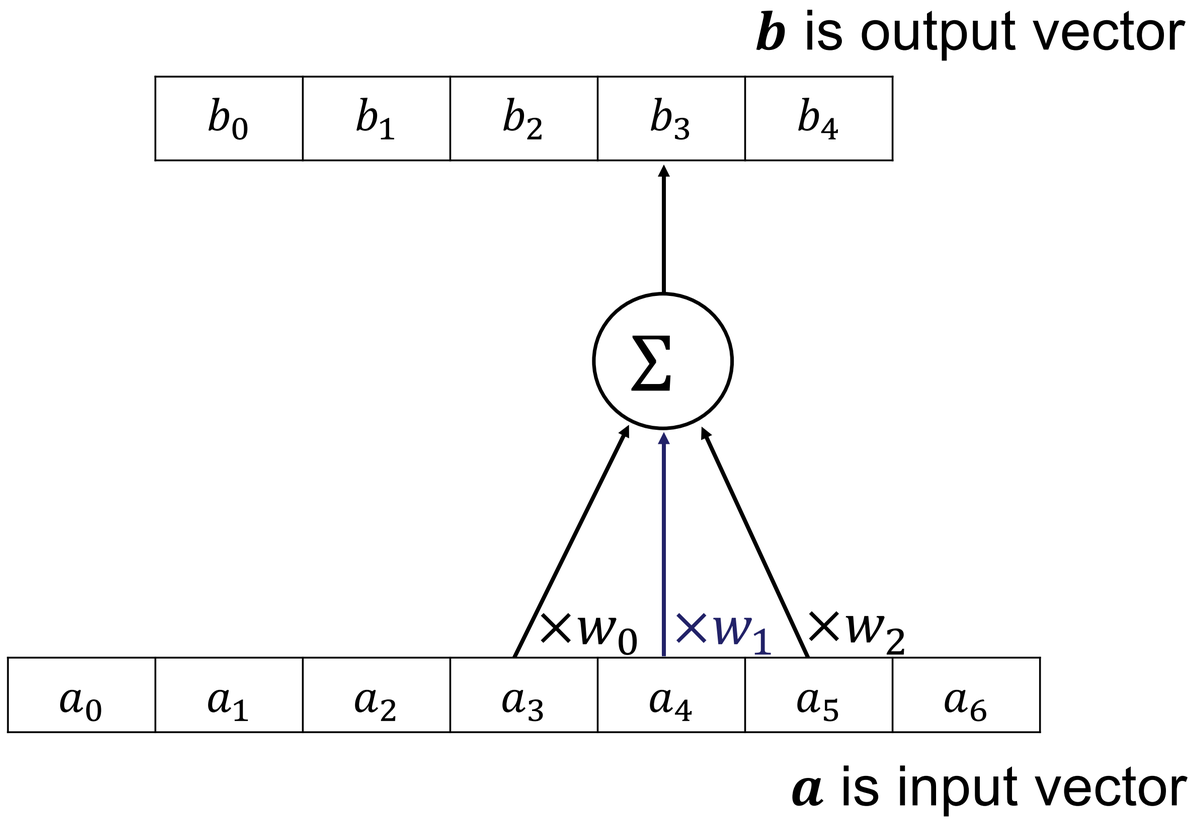
Convolution on 1D



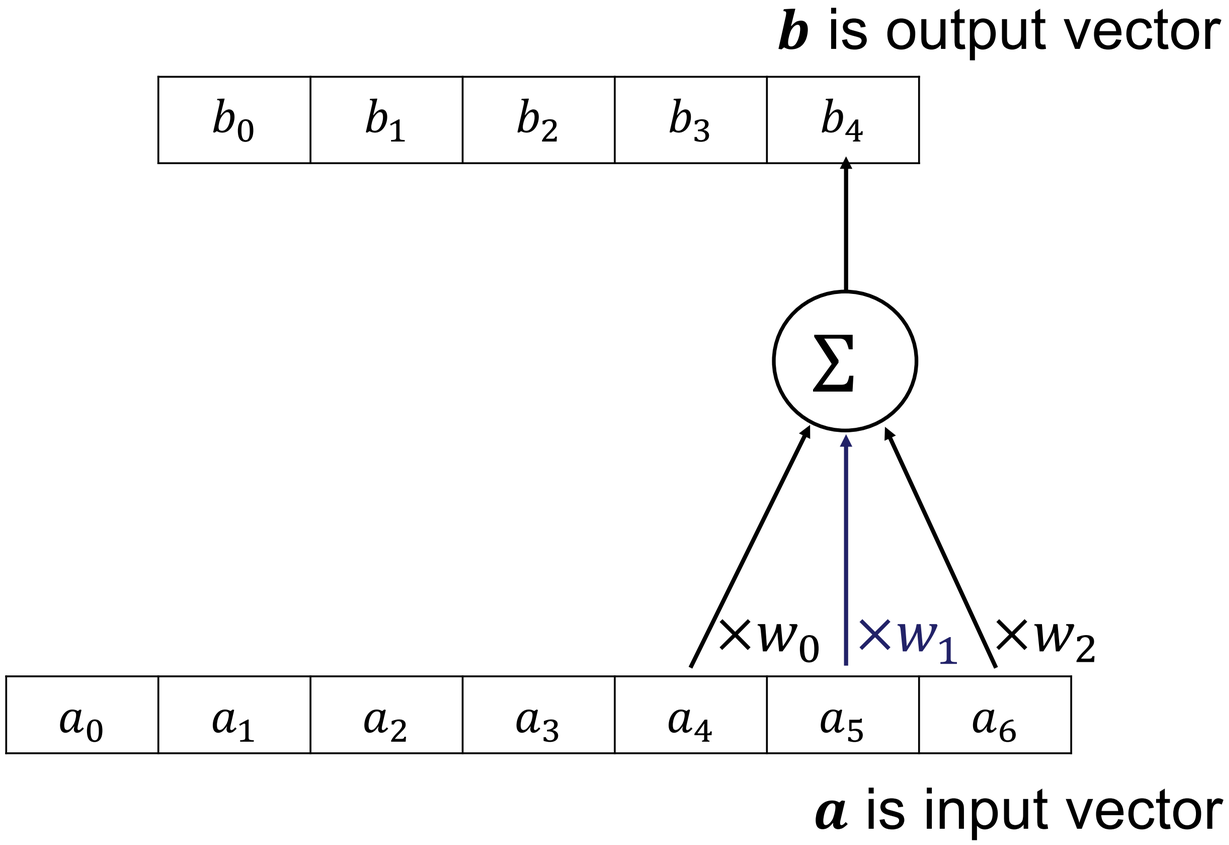
Convolution on 1D



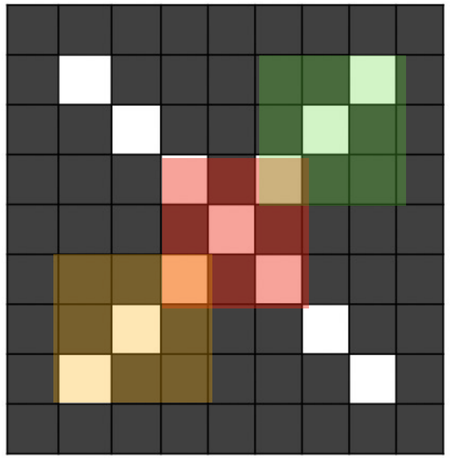
Convolution on 1D



Convolution on 1D

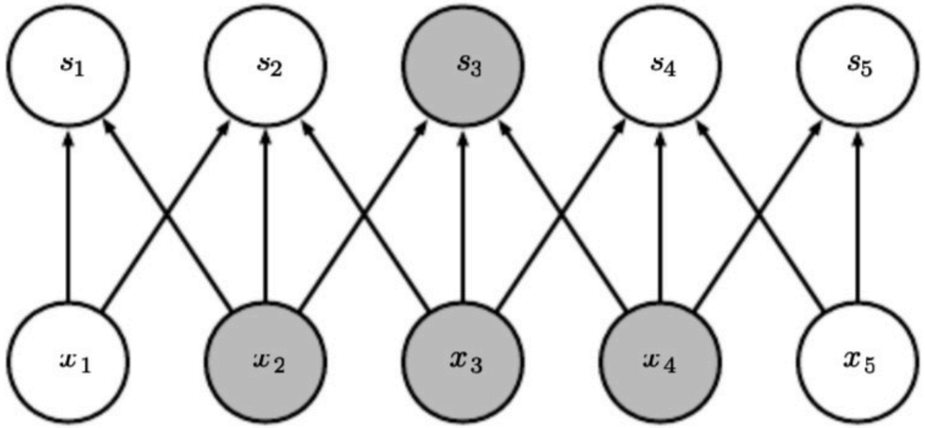


Advantage: learn translation-invariant pattern



Advantage: weight sharing and sparse connection

Convolutional layer:



Fully connected layer:
Each arrow is a weight (no sharing)

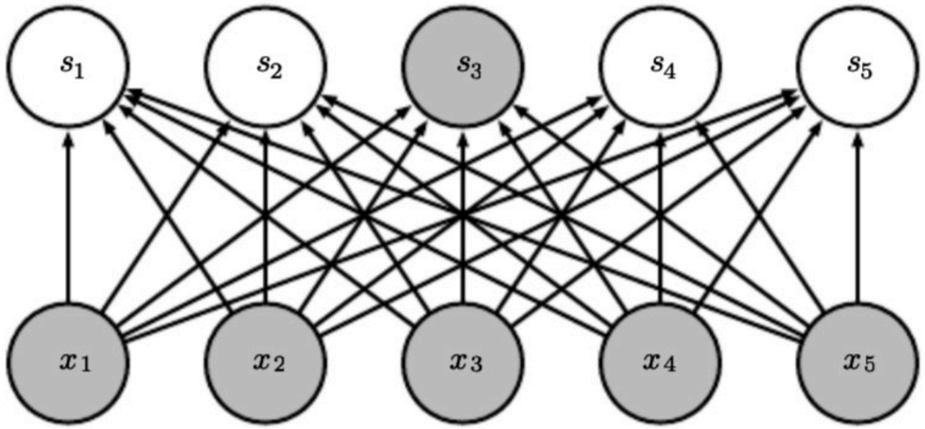


Figure 9.3 in Deep learning by Ian Goodfellow and Yoshua Bengio and Aaron Courville

Advantage: learn hierarchical pattern

More layers: larger size of receptive field (larger window of the input is seen)

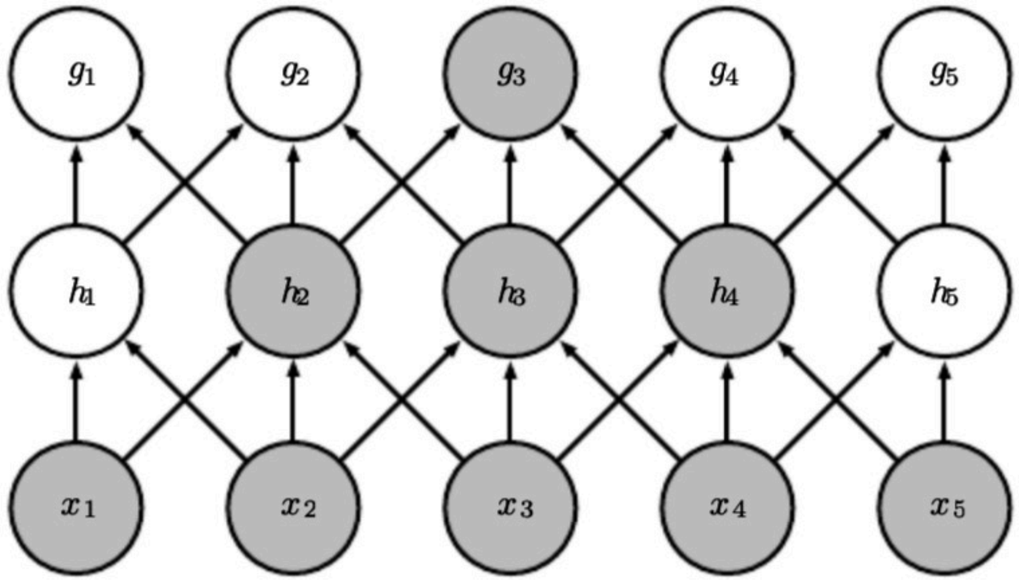


Figure 9.4 in Deep learning by Ian Goodfellow and Yoshua Bengio and Aaron Courville

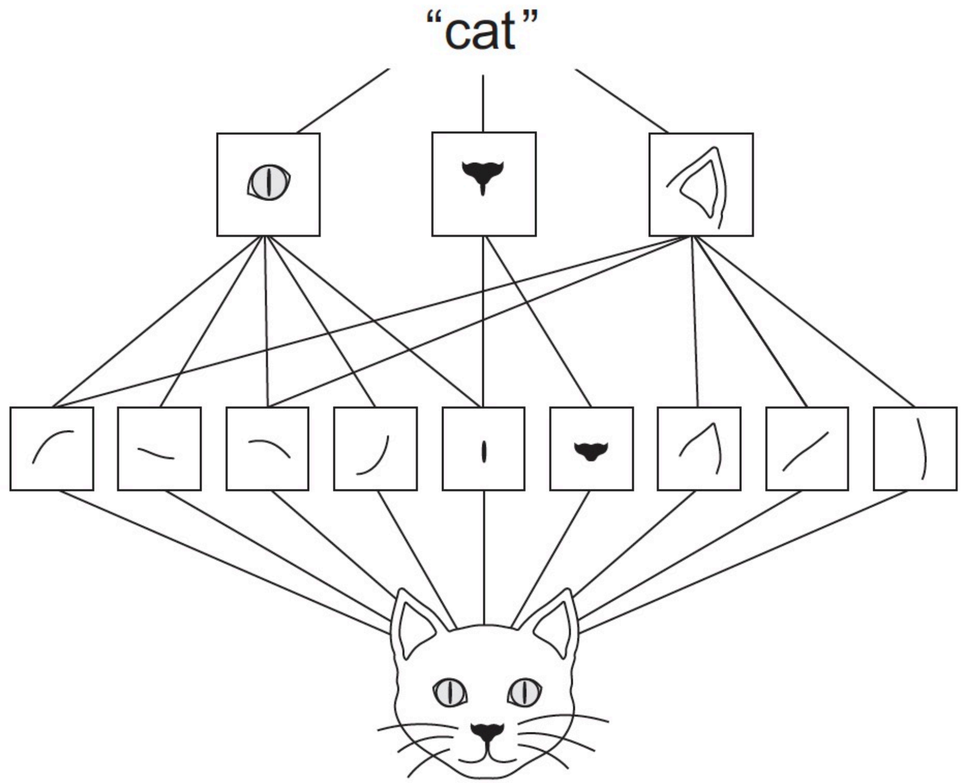
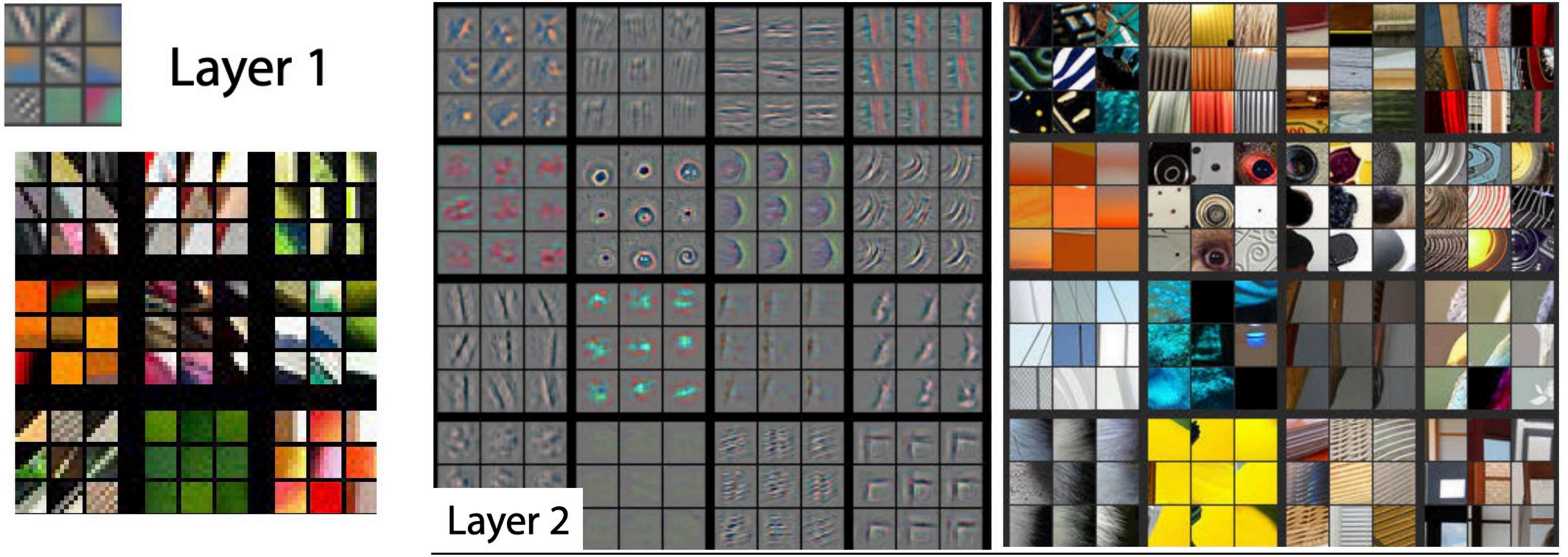
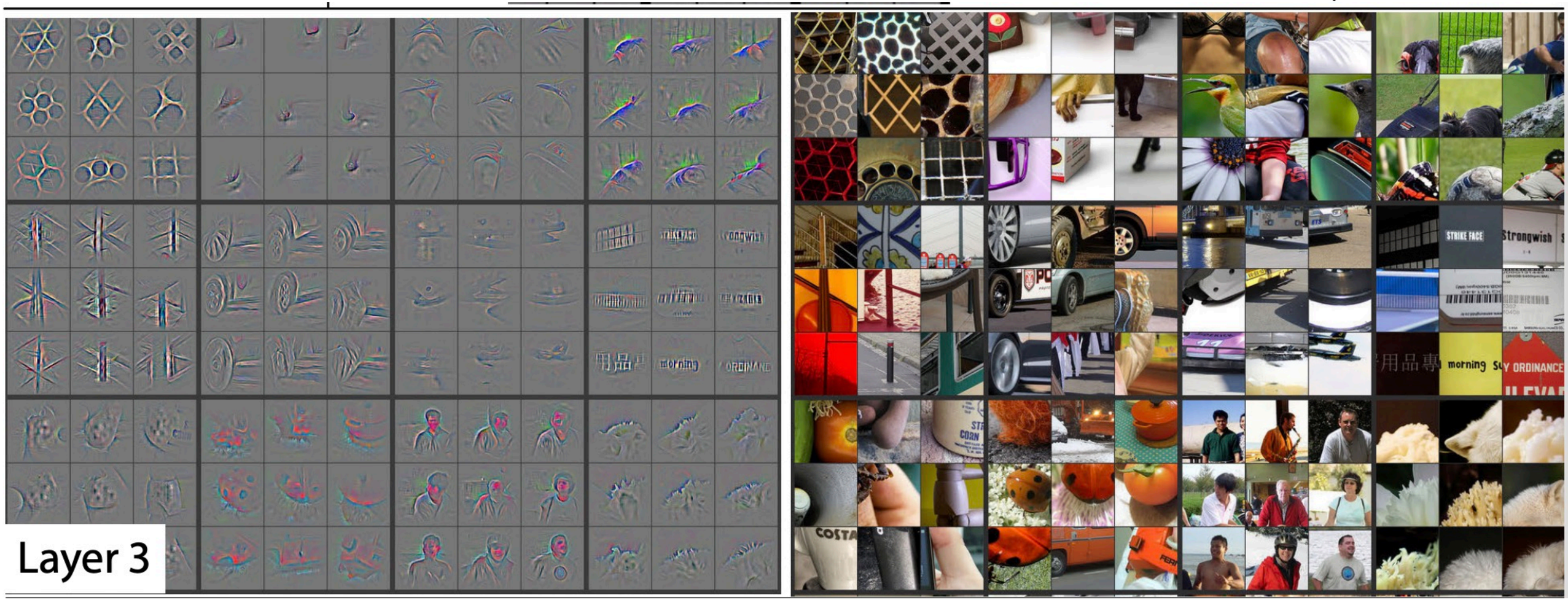


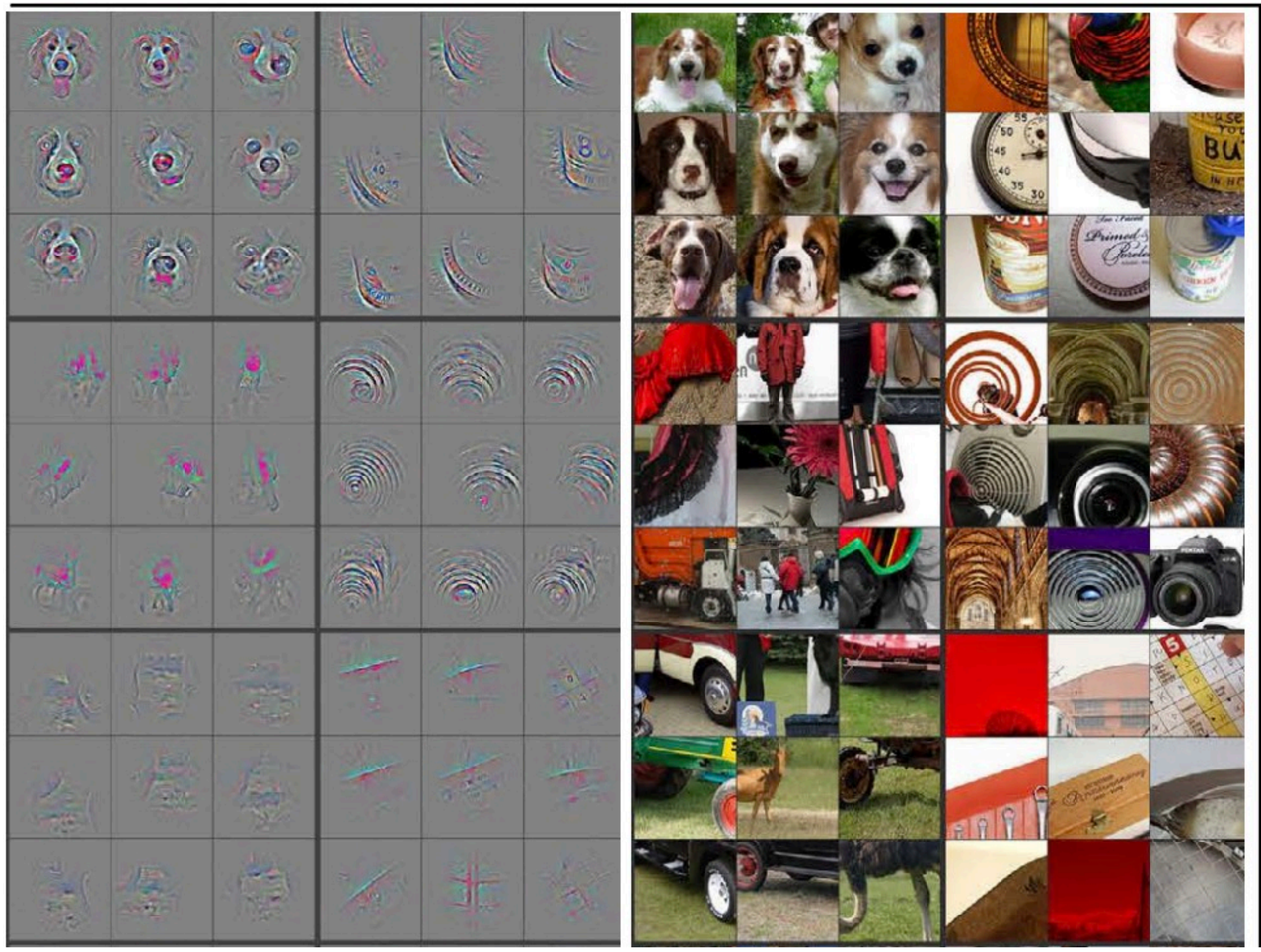
Figure 5.2 in Deep learning with python by Francois Chollet



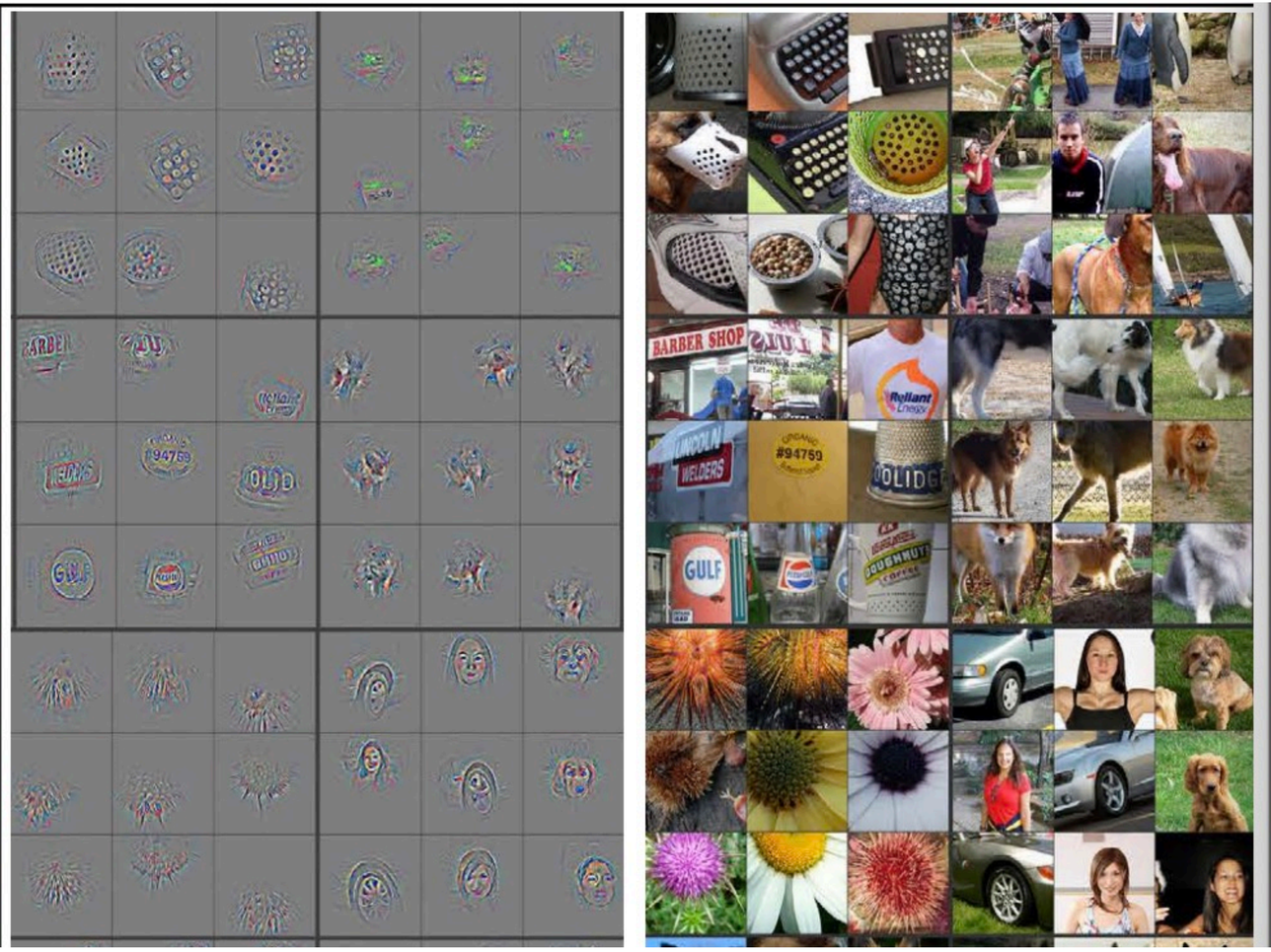
Zeiler, Matthew D., and Rob Fergus. "Visualizing and understanding convolutional networks." *European conference on computer vision*. Springer, Cham, 2014.



Zeiler, Matthew D., and Rob Fergus. "Visualizing and understanding convolutional networks." *European conference on computer vision*. Springer, Cham, 2014.



Layer 4



Layer 5

Zeiler, Matthew D., and Rob Fergus. "Visualizing and understanding convolutional networks." *European conference on computer vision*. Springer, Cham, 2014.

output size $\neq$ input size

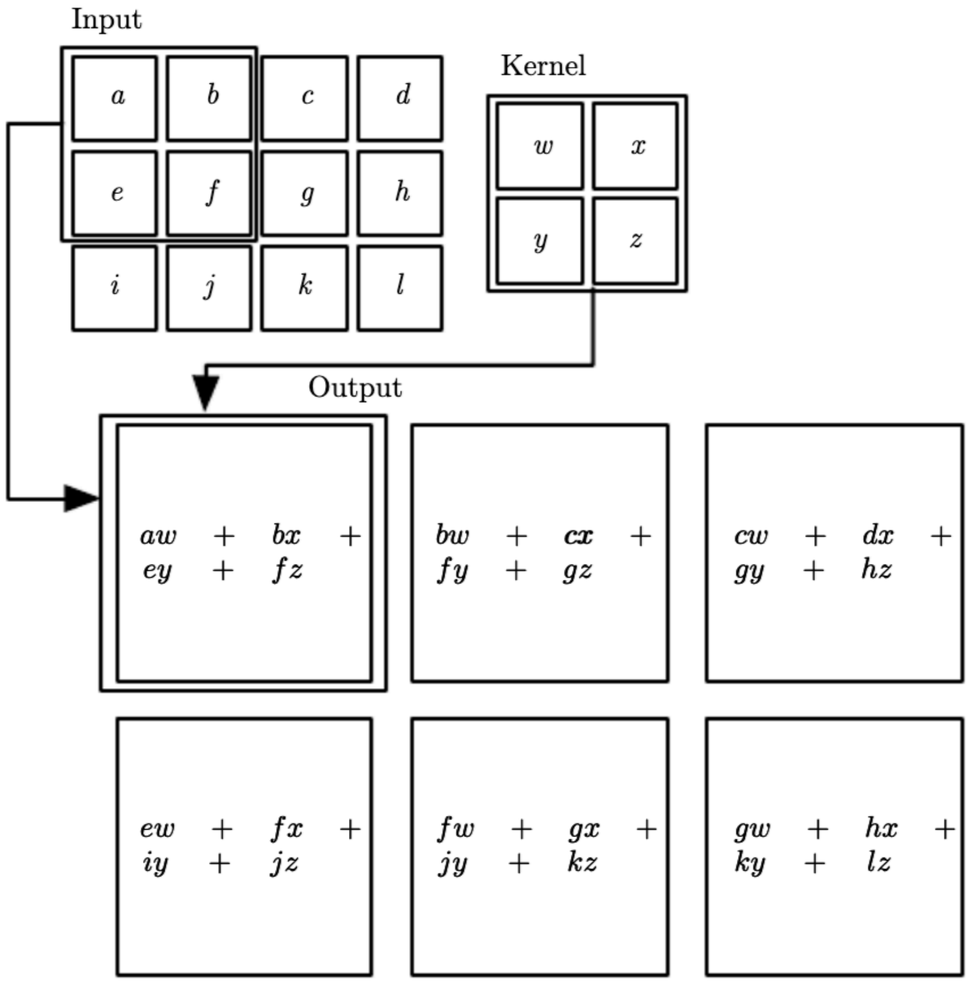


Figure 9.1 in Deep learning by Ian Goodfellow and Yoshua Bengio and Aaron Courville

Padding

adding an appropriate number of rows and columns on each side of the input feature map

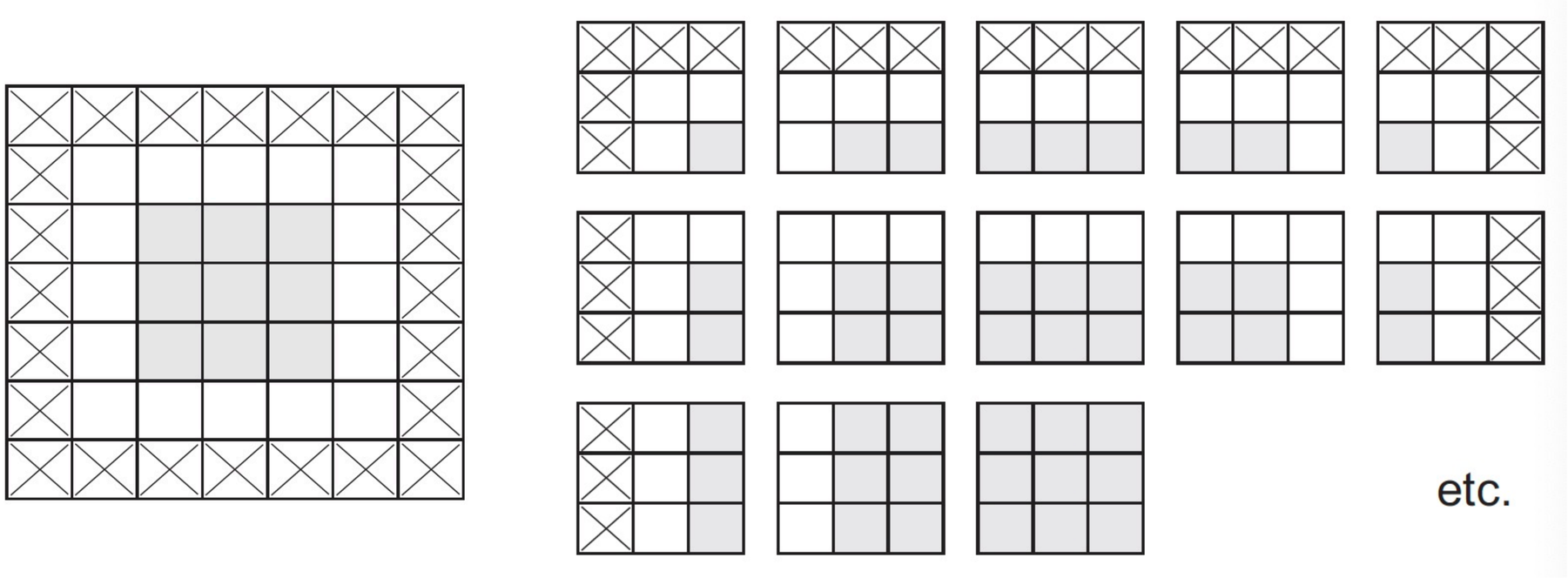


Figure 5.6 in Deep learning with python by Francois Chollet

Stride

the distance between two successive windows

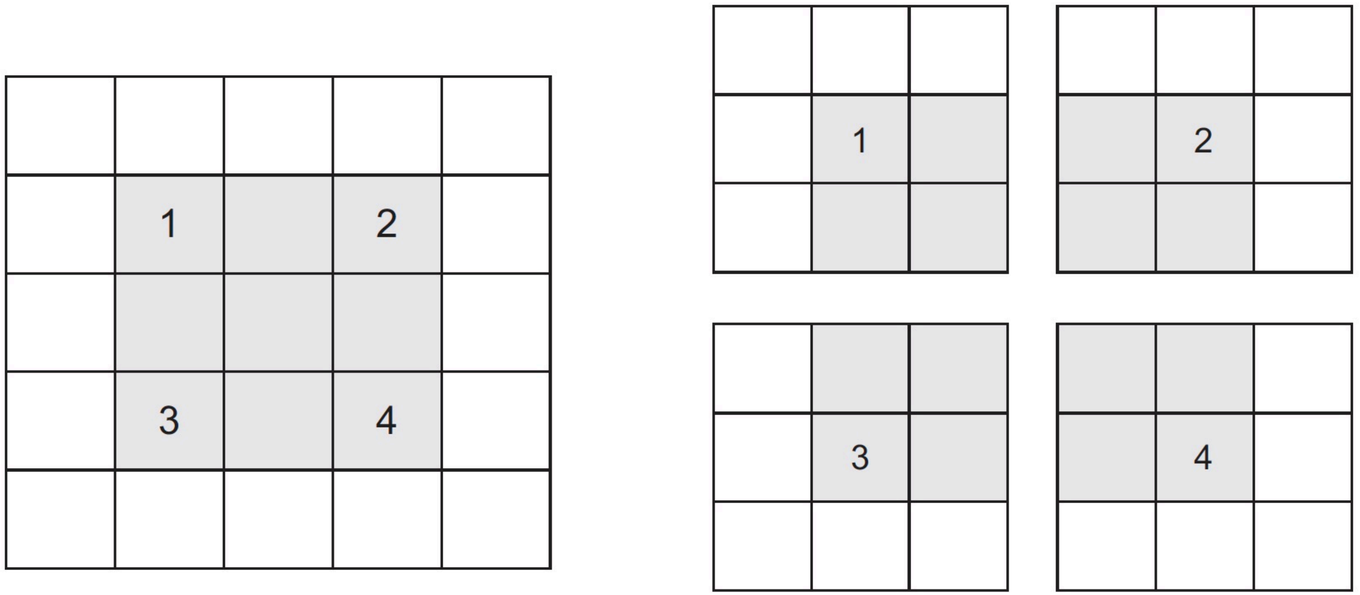


Figure 5.5 in Deep learning with python by Francois Chollet

No padding: output_size=ceiling((input_size-kernel_size+1)/stride)

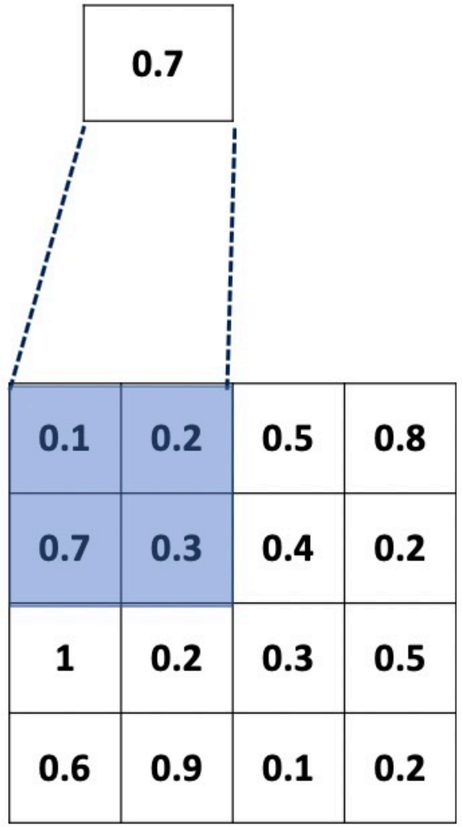
padding: output_size=ceiling(input_size /stride)

Convolutional layer

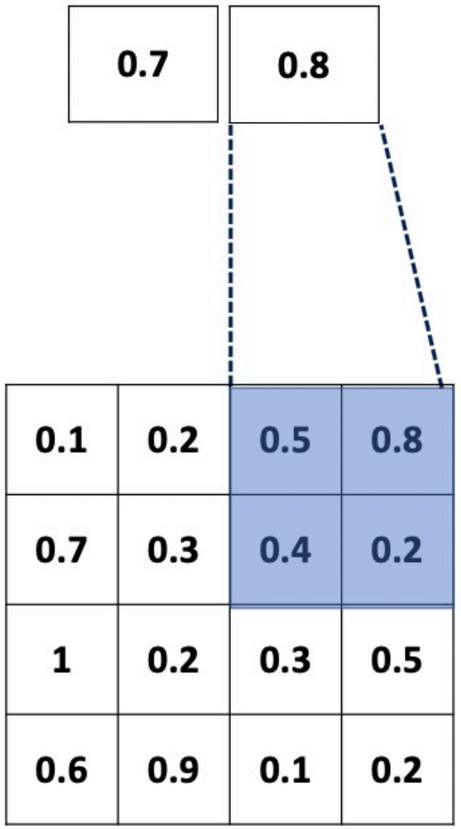
filters: the number of filters in the convolution
kernel_size: the height and width of the 2D convolution window
padding: one of "valid" or “same”
stride: the strides of the convolution along the height and width

```
layer=keras.layers.Conv2D(
    32,
    (3, 3),
    padding='same',
    strides=1,
    activation='relu')
```

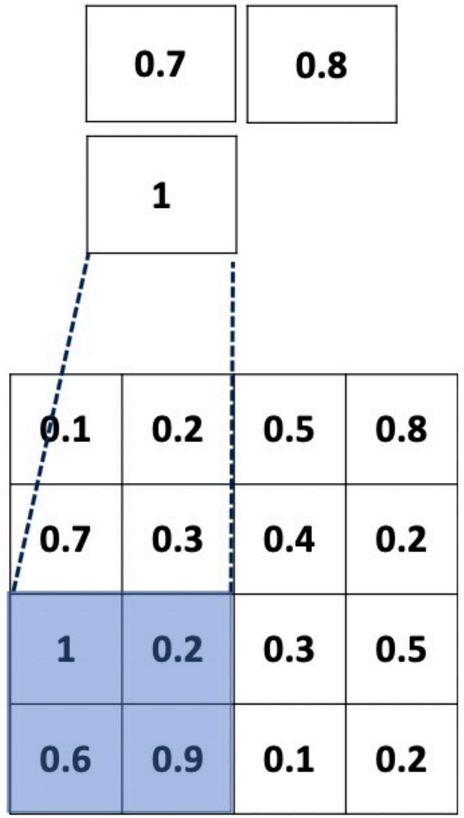
Max-pooling layer



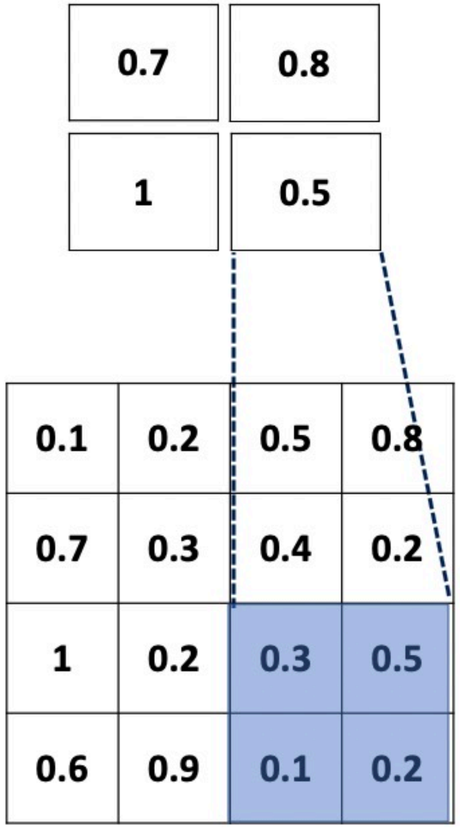
Max-pooling layer



Max-pooling layer

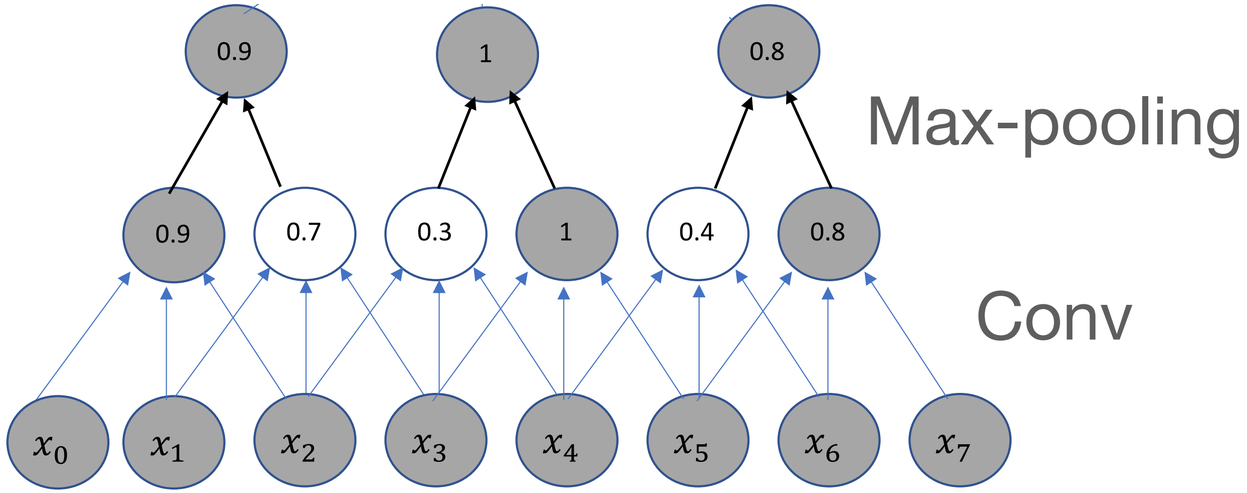


Max-pooling layer

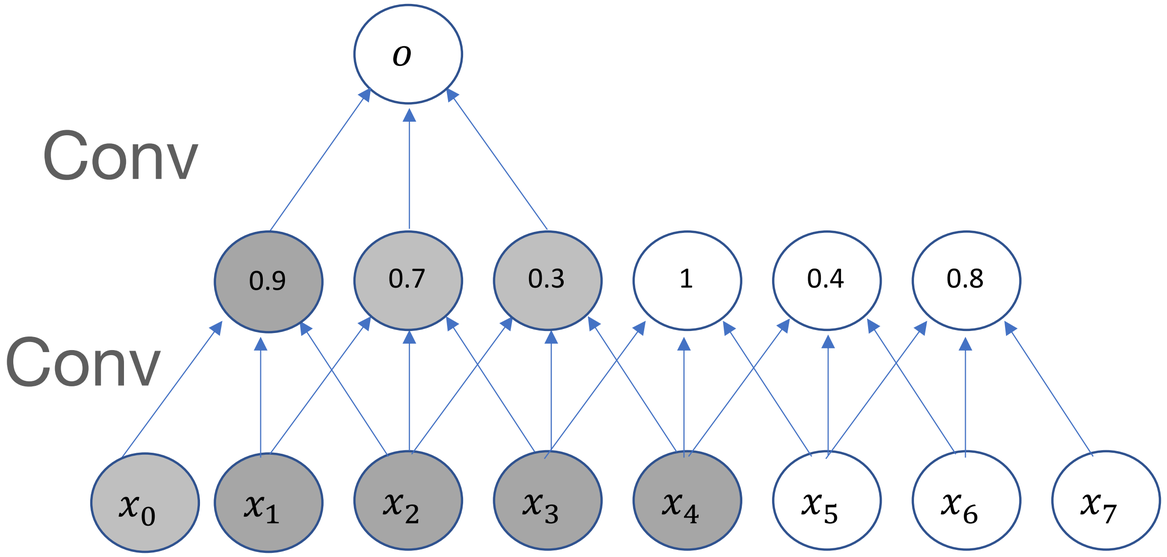


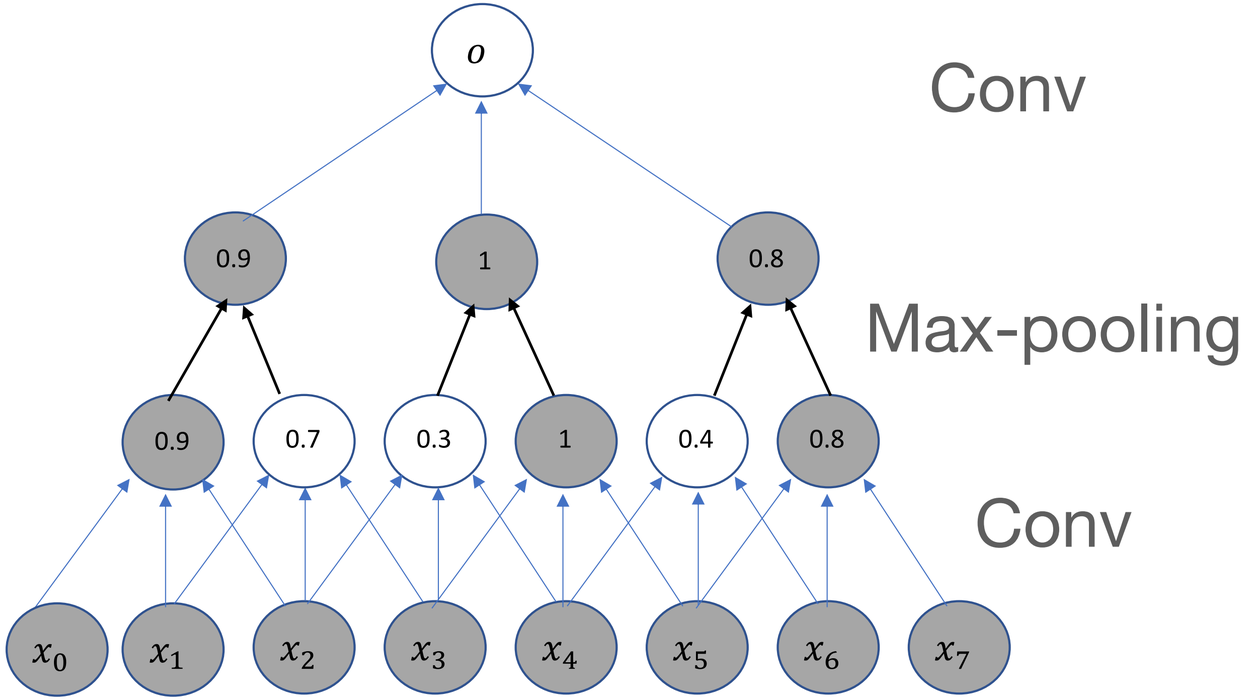

```
tf.keras.layers.MaxPooling2D(
    pool_size=(2, 2), strides=None, padding="valid")
```

Advantage: downsample feature map, reduce computational burden



Advantage: increase size of receptive field (window of the input is seen)





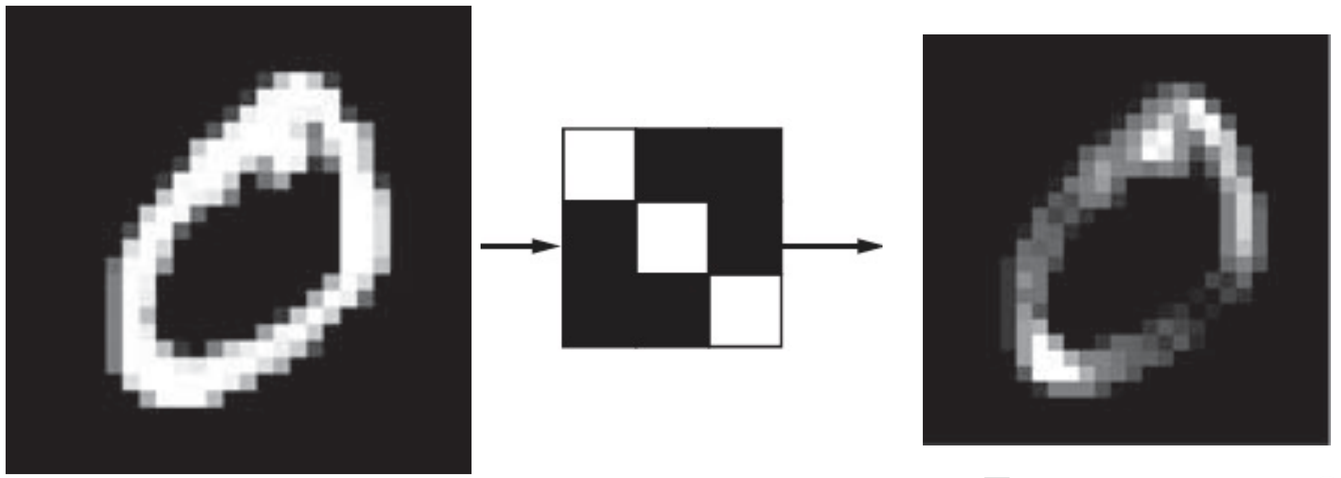
Other pooling method: Average pooling

taking the average value over the patch

Why max-pooling to downsample feature map?

Average pooling: dilute feature-presence information

Convolution with stride>1: miss feature-presence information



Input kernel

Feature map: 2D map of the presence of a pattern at different locations in an input

Figure 5.3 in Deep learning with python by Francois Chollet

Summary

- Why/what is deep learning?
- Disadvantages of multilayer perceptron?
- How to perform convolution?
- Advantages of convolutional layer
- How to perform max-pooling?
- Advantages of max-pooling?
- What is stride/padding?
- How to compute output size of feature maps?