I-Form

Advanced Manufacturing Research Centre

A World Leading SFI Research Centre

Science Foundation Ireland sfi For what's next

Introduction to Computer Vision

Dr. Annalina Caputo – annalina.caputo@dcu.ie

HOST INSTITUTION

UCD DUBLIN

PARTNER INSTITUTIONS

DCU

NUI Galway OÉ Gaillimh

Trinity College Dublin Coláiste na Tríonóide, Baile Átha Cliath The University of Dublin

IT Sligo An Institiúid Teicneolaíochta, Sligeach

Waterford Institute of Technology

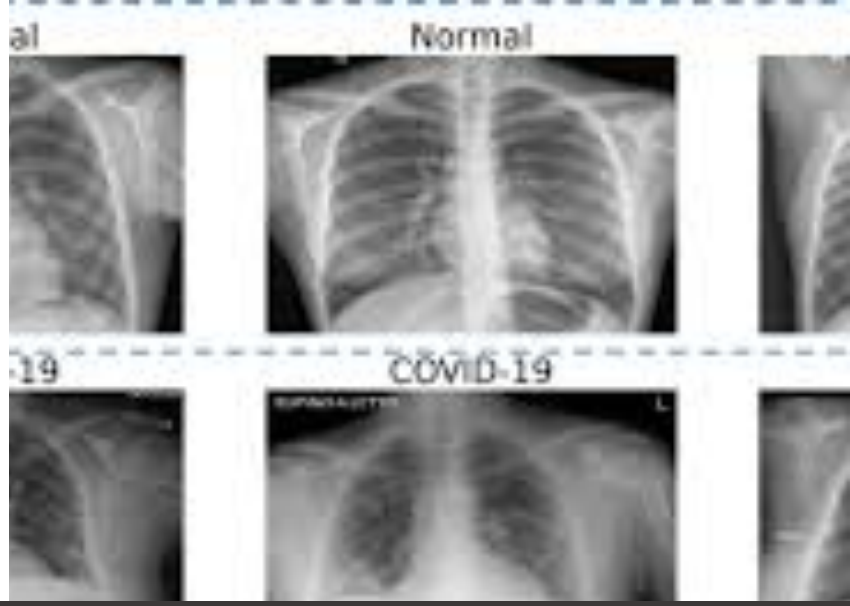
Maynooth University National University of Ireland Maynooth

FUNDED BY:

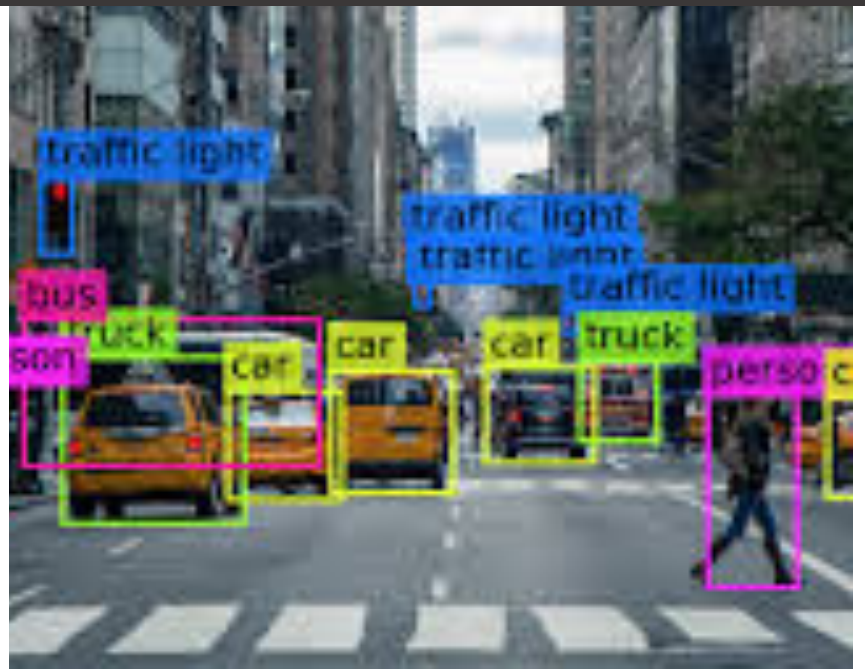
European Union European Regional Development Fund

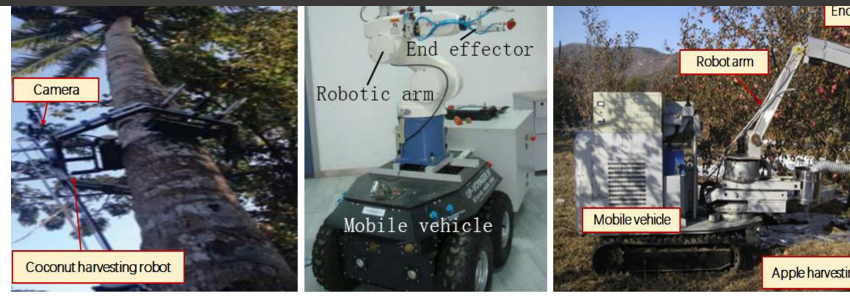
Science Foundation Ireland sfi For what's next

I-Form is an SFI Research Centre funded under the Science Foundation Ireland Research Centres Programme and co-funded under the European Regional Development Fund.



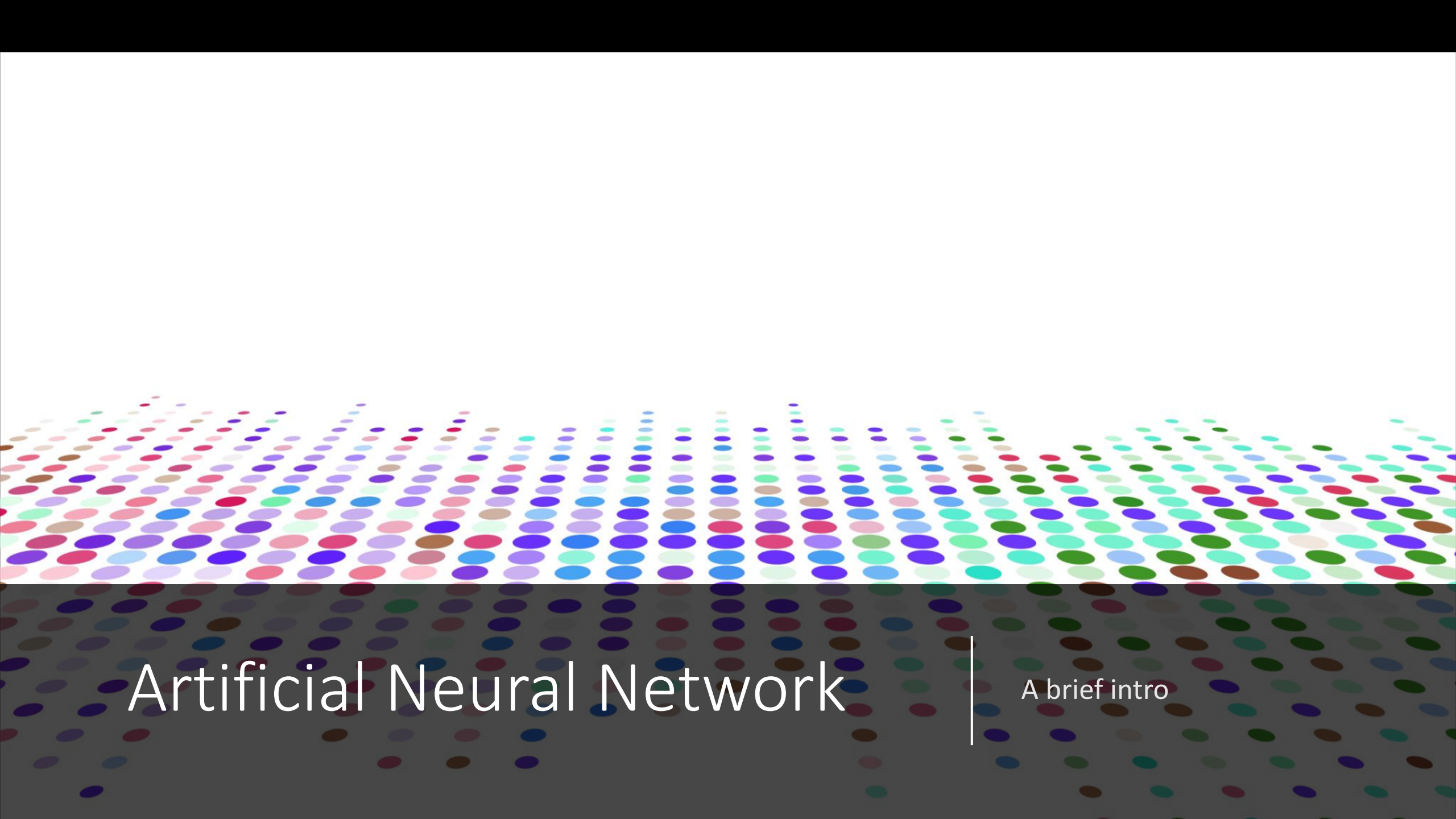
Why Computer Vision?

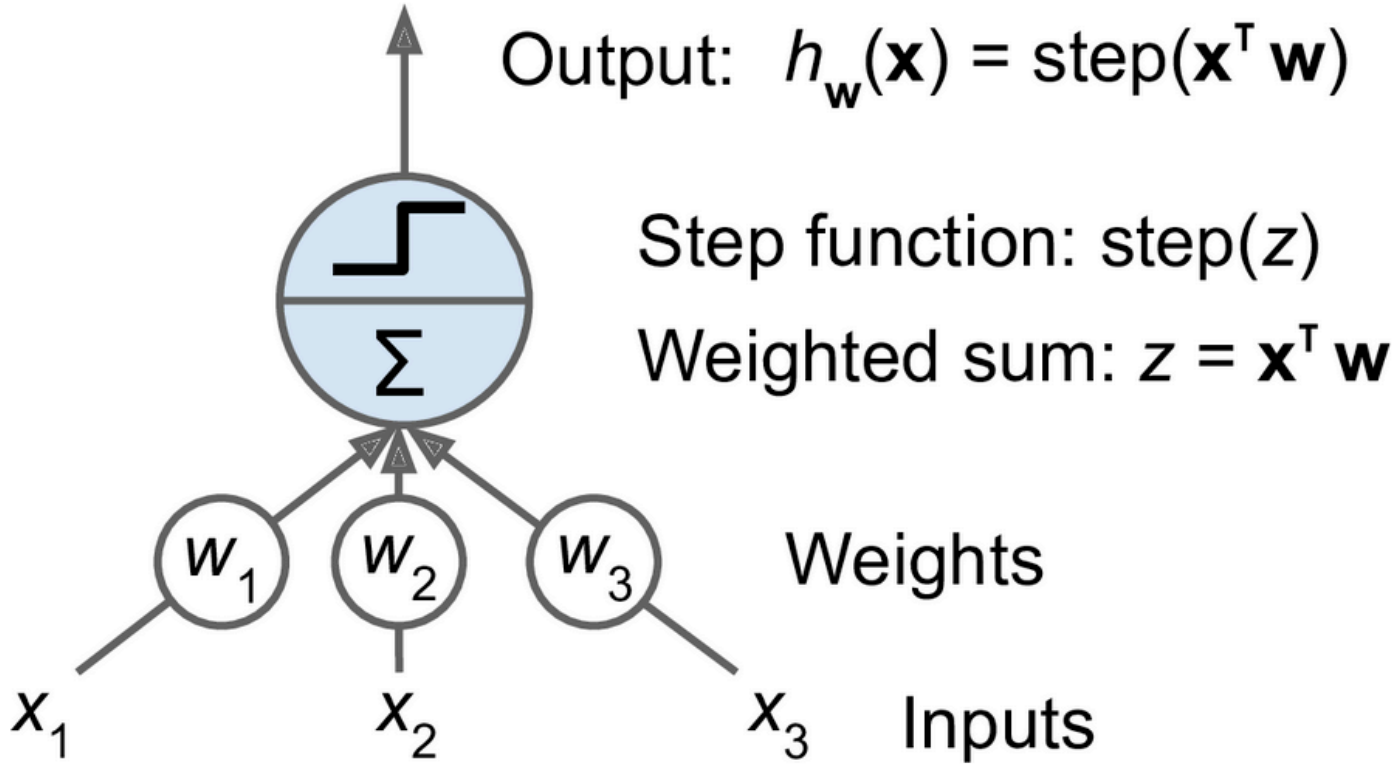




Outline

- Intro to Neural Network and Deep Learning
- Intro to TensorFlow/Keras
- Intro to Convolutional Neural Network
- Hands on TensorFlow


Artificial Neural Network
A brief intro



Basic Neural Network – Threshold Logic Unit

- Perceptron [Rosenblatt, 1957]
- Binary classifier that maps input to an output value
- Basic neural network building block
- Simplest feedforward neural network

Source: Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition

Step functions

Heavside

$$heaviside(z) = \begin{cases} 0 & if\ z < 0 \\ 1 & if\ z \geq 0 \end{cases}$$

Sign

$$sng(z) = \begin{cases} -1 & if\ z < 0 \\ 0 & if\ z = 0 \\ +1 & if\ z > 0 \end{cases}$$

Perceptron

Output

$$h_{W,b}(X) = \phi(XW + b)$$

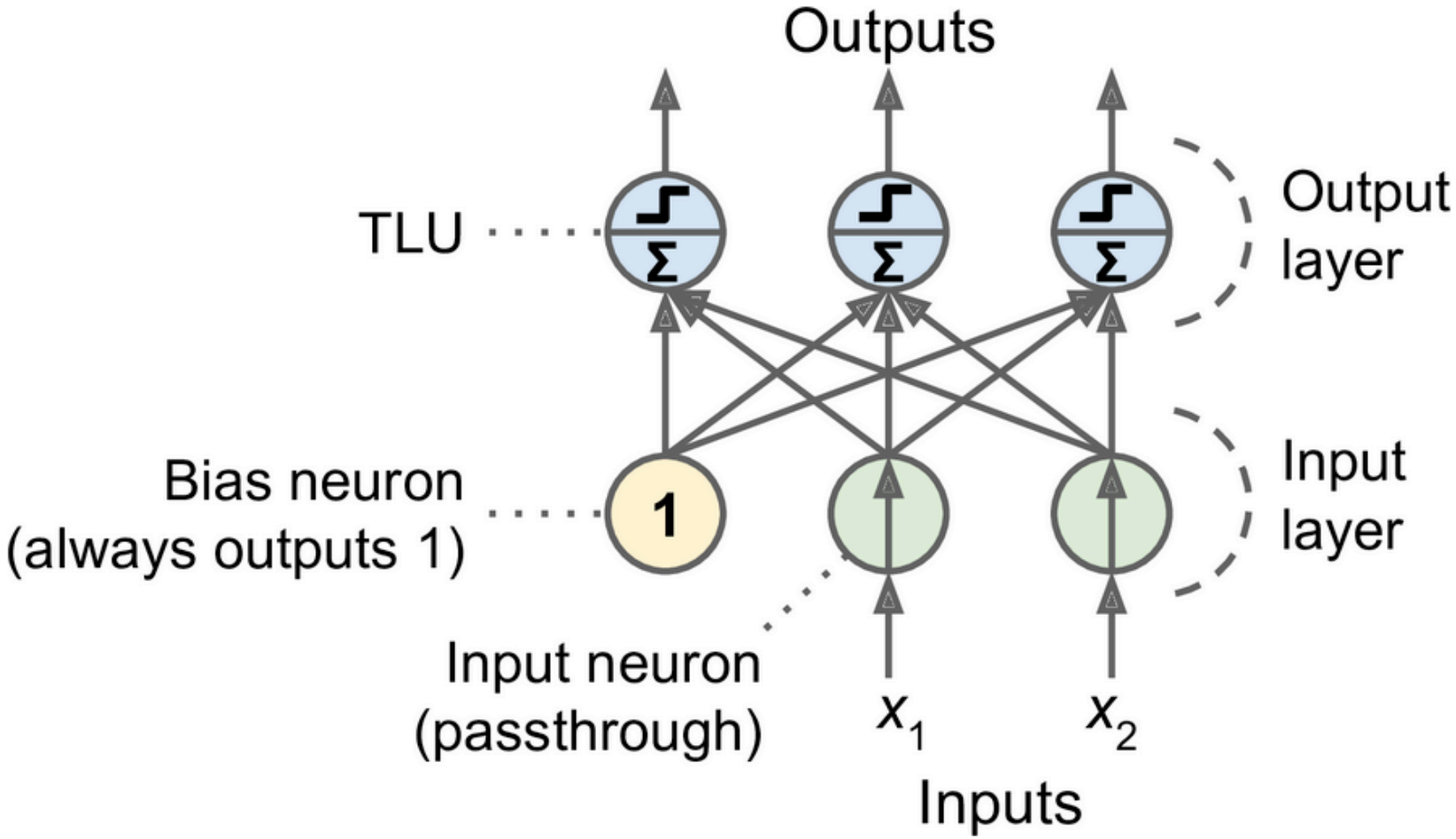
X Matrix input features

W Matrix connection weights

b Connection weights between bias and the neurons

ϕ *activation function*

(in TLU is a step function)



How does a perceptron learn?

- Hebb's rule

"Cells that fire together, wire together"

- Take into account error made during prediction
  - Make prediction for each instance at the time
  - Reinforce connections that help reduce the error

$$w_{i,j}^{t} = w_{i,j}^{t-1} + \eta\left(y_j - \hat{y}_j\right)x_i$$

How does a perceptron learn?

- Hebb's rule

"Cells that fire together, wire together"

- Take into account error made during prediction
  - Make prediction for each instance at the time
  - Reinforce connections that help reduce the error

i^{th} input neuron

j^{th} output neuron

$$w_{i,j}^{t} = w_{i,j}^{t-1} + \eta\left(y_j - \hat{y}_j\right)x_i$$

How does a perceptron learn?

- Hebb's rule

"Cells that fire together, wire together"

- Take into account error made during prediction
  - Make prediction for each instance at the time
  - Reinforce connections that help reduce the error

$$w_{i,j}^{t} = w_{i,j}^{t-1} + \eta\left(y_j - \hat{y}_j\right)x_i$$

New weight for connection

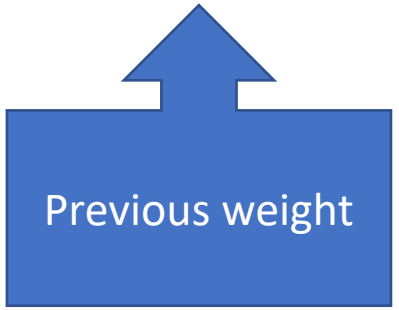
How does a perceptron learn?

- Hebb's rule

"Cells that fire together, wire together"

- Take into account error made during prediction
  - Make prediction for each instance at the time
  - Reinforce connections that help reduce the error

$$w_{i,j}^{t} = w_{i,j}^{t-1} + \eta\left(y_j - \hat{y}_j\right)x_i$$



How does a perceptron learn?

- Hebb's rule

"Cells that fire together, wire together"

- Take into account error made during prediction
  - Make prediction for each instance at the time
  - Reinforce connections that help reduce the error

$$w_{i,j}^{t} = w_{i,j}^{t-1} + \eta\left(y_j - \hat{y}_j\right)x_i$$

Learning rate

How does a perceptron learn?

- Hebb's rule

"Cells that fire together, wire together"

- Take into account error made during prediction
  - Make prediction for each instance at the time
  - Reinforce connections that help reduce the error

$$w_{i,j}^{t} = w_{i,j}^{t-1} + \eta\left(y_j - \hat{y}_j\right)x_i$$

Target output for the j^{th} output neuron

How does a perceptron learn?

- Hebb's rule

"Cells that fire together, wire together"

- Take into account error made during prediction
  - Make prediction for each instance at the time
  - Reinforce connections that help reduce the error

$$w_{i,j}^{t} = w_{i,j}^{t-1} + \eta\left(y_j - \hat{y}_j\right)x_i$$

output for the j^{th} output neuron

How does a perceptron learn?

- Hebb's rule

"Cells that fire together, wire together"

- Take into account error made during prediction
  - Make prediction for each instance at the time
  - Reinforce connections that help reduce the error

$$w_{i,j}^{t} = w_{i,j}^{t-1} + \eta\left(y_j - \hat{y}_j\right)x_i$$

i^{th} input value for the training instance

How does a perceptron learn?

- Hebb's rule

"Cells that fire together, wire together"

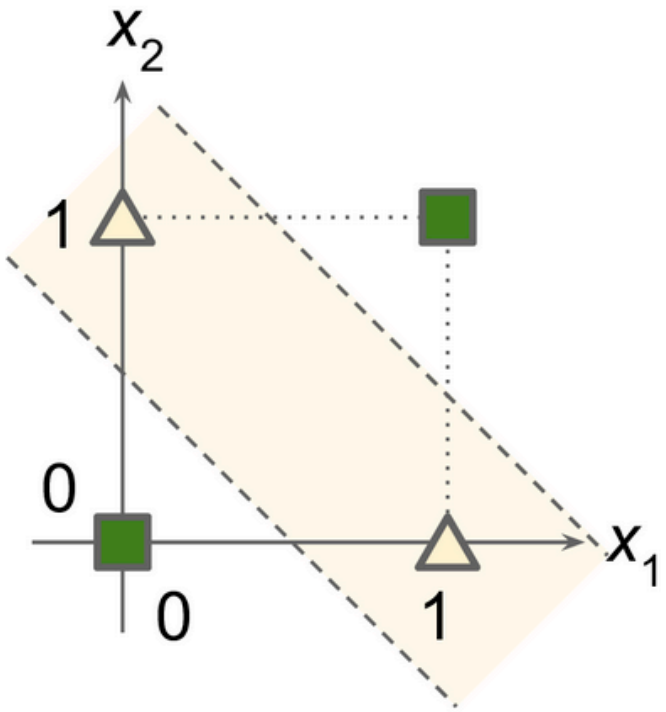
- Take into account error made during prediction
  - Make prediction for each instance at the time
  - Reinforce connections that help reduce the error

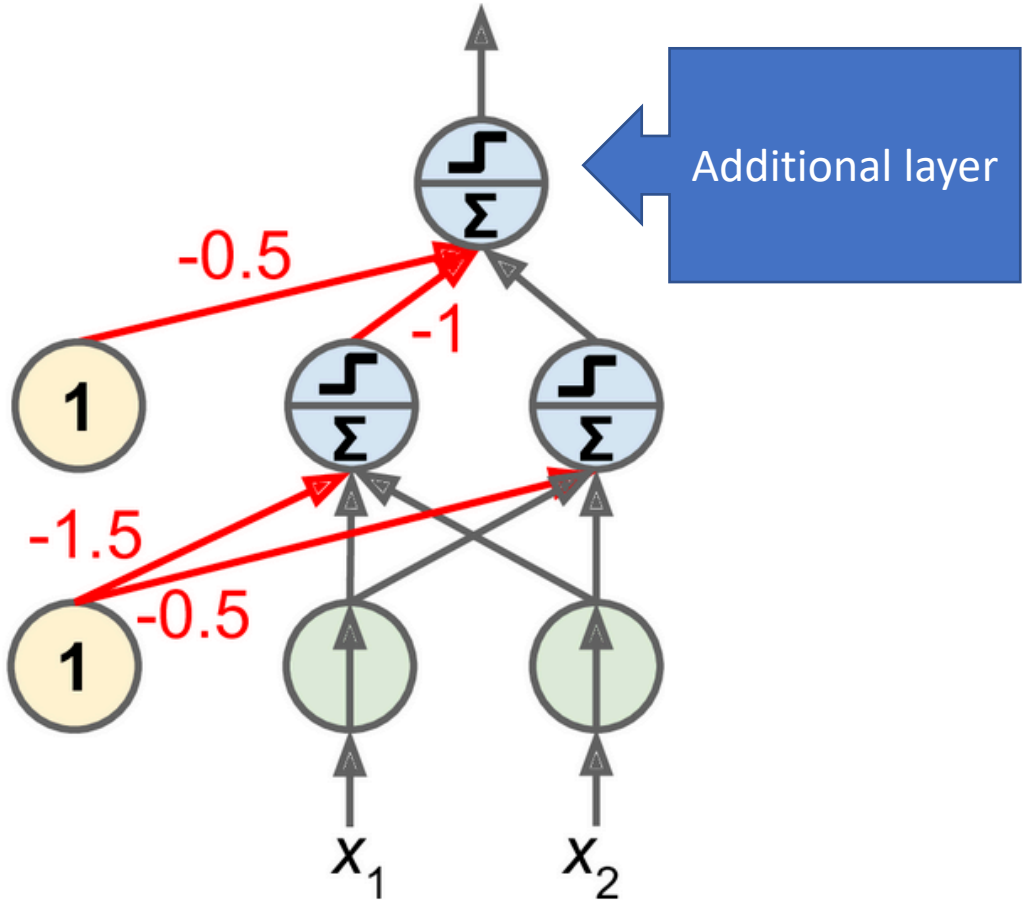
$$w_{i,j}^{t} = w_{i,j}^{t-1} + \eta\left(y_j - \hat{y}_j\right)x_i$$

  - Observation: linear decision boundaries ≡ logistic regression classifiers

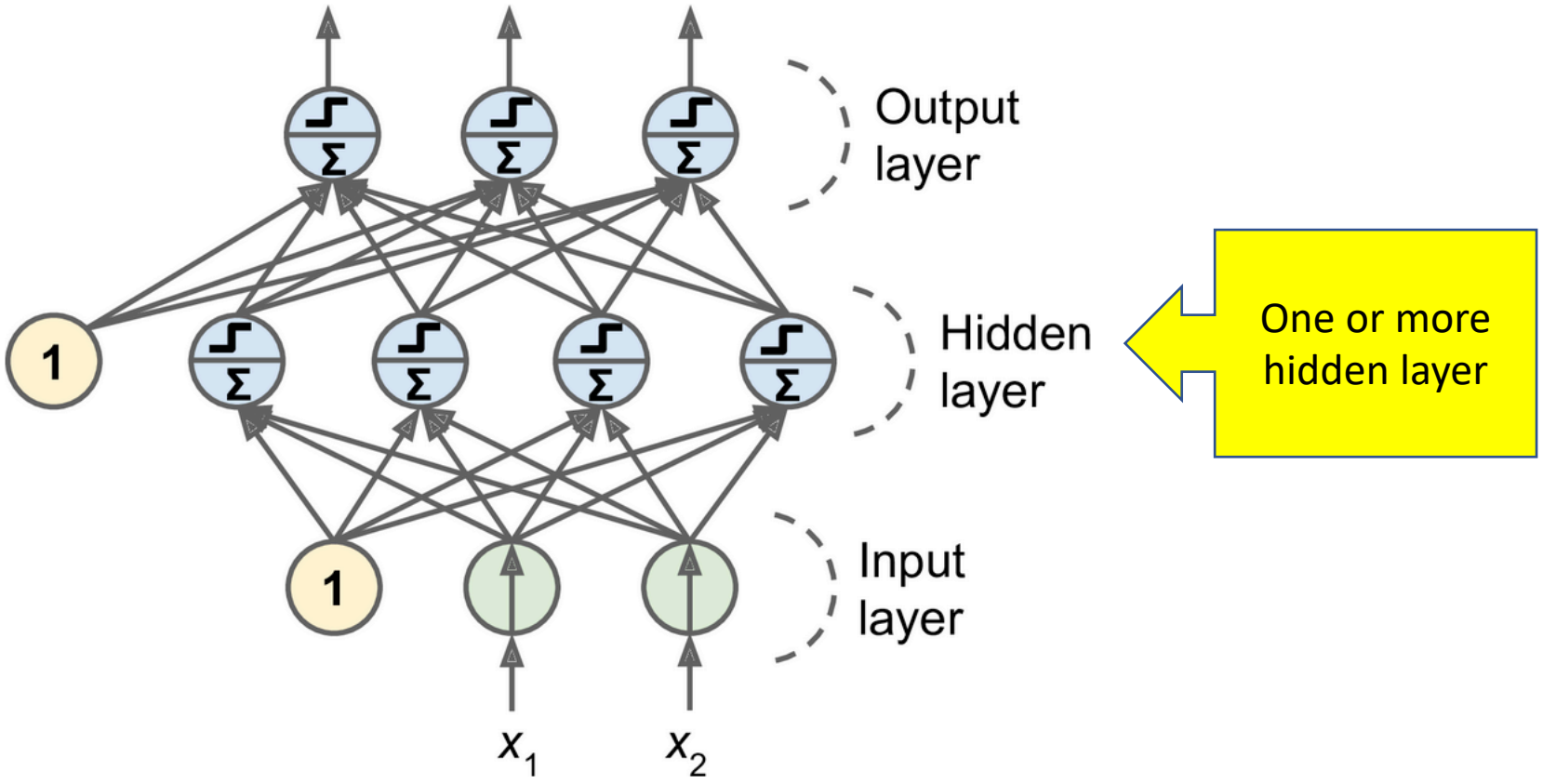
Multilayer Perceptron (MLP)

- Can solve XOR

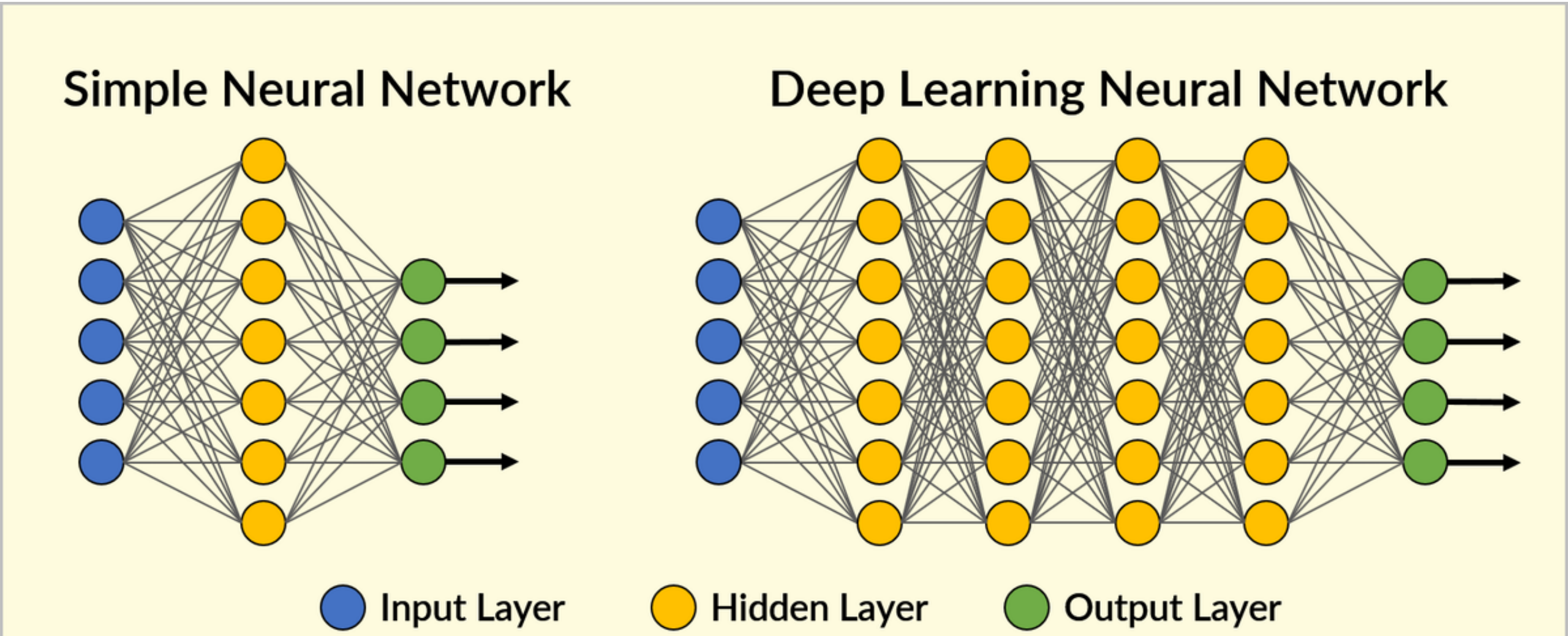




Multilayer Perceptron (MLP)

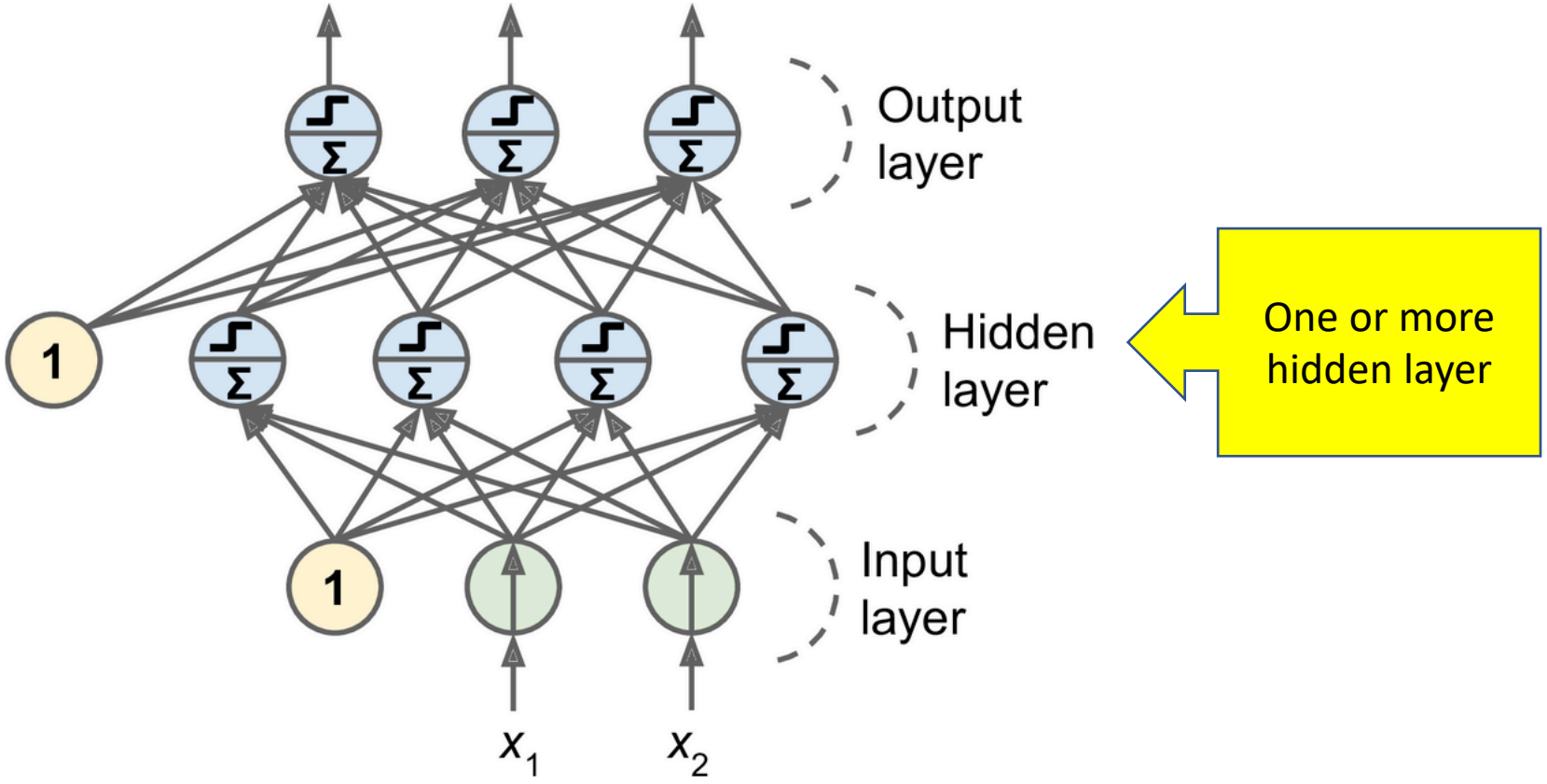


Deep Neural Network



https://www.securityinfowatch.com/video-surveillance/video-analytics/article/21069937/deep-learning-to-the-rescue

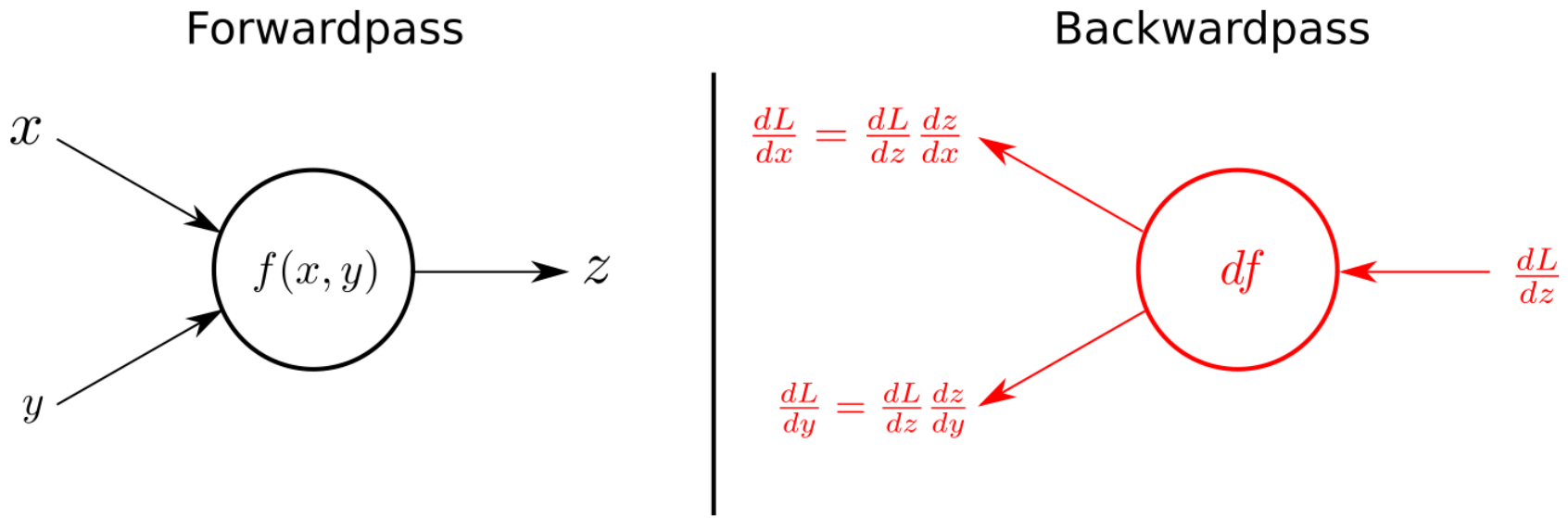
Multilayer Perceptron (MLP)



Feedforward Neural Network

Backpropagation

- Compute the gradient of the network's error wrt every model parameter
- Tweaks every single connection and bias weight to reduce error



https://github.com/Vercaca/NN-Backpropagation

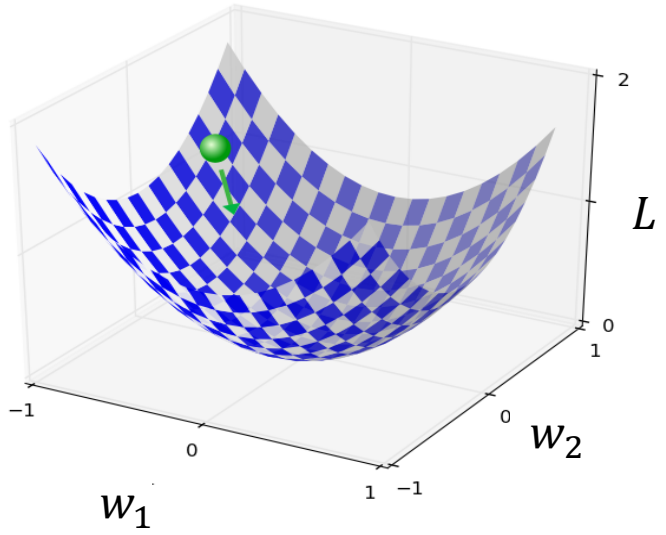
Backpropagation

- Handle mini-batch (small set of training instances)
- Go through training set multiple times (each is called *epoch*)

1. Forward pass: the mini-batch is passed from the input through all the hidden layers up to the output layer – weights for each layer are stored for back pass
2. Compute the error (loss function L)
3. Chain rule: compute how much each output connection contributed to the error
4. Reverse pass: computes how much of these error contributions came from connections in previous layer (apply again chain rule) up to the first input layer
5. Gradient Descent: tweak all the connection weights using the error gradients just computed

Gradient Descent

- L loss function (aka cost function)
- Minimise L -> find where L has a global minimum
- Analytically -> compute the derivatives
  - Don’t work with NN -> billions of weights



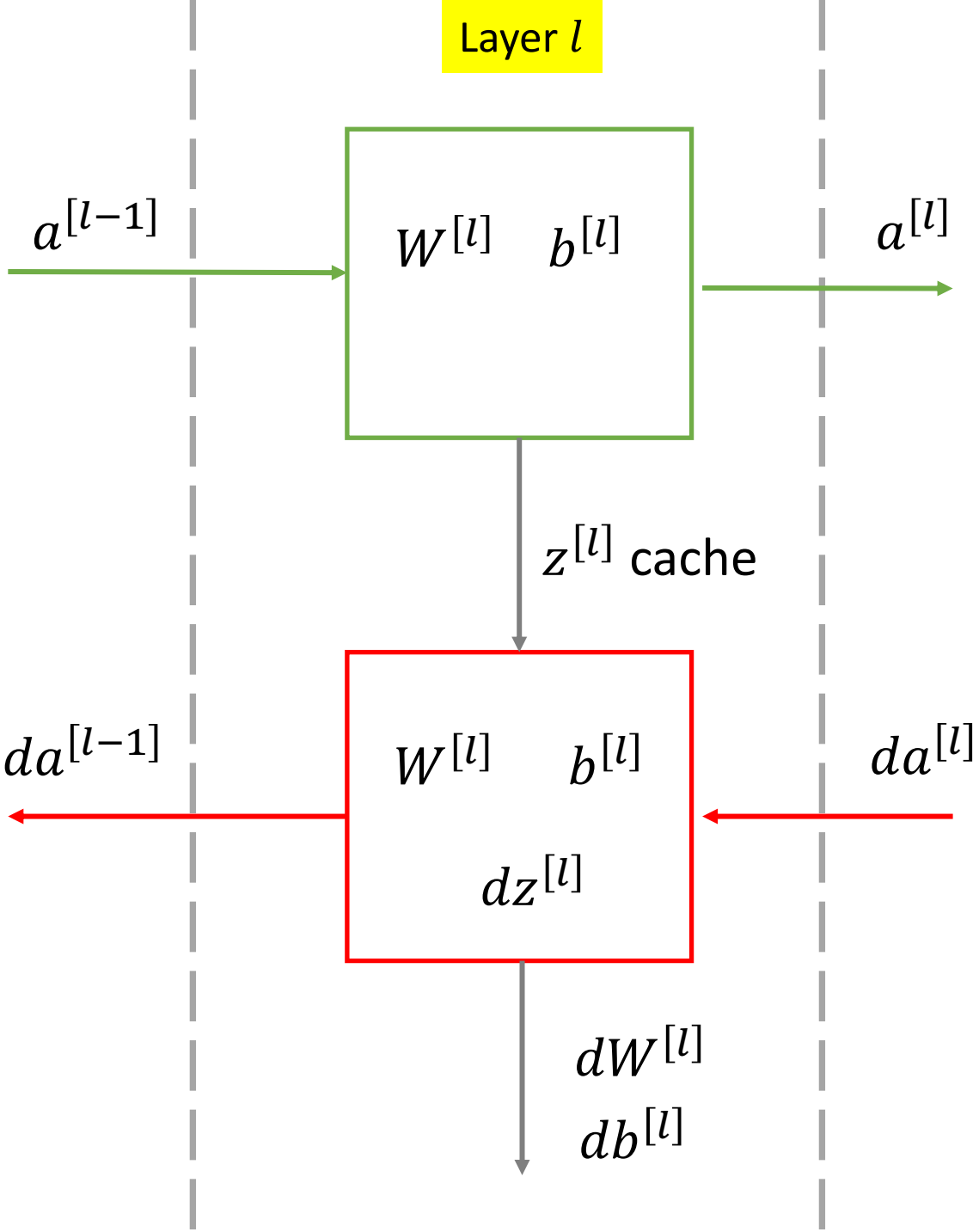
- Construct model using initial random weights
- Repeat until good performance
  - Apply model to the example
  - Compute the lost function

$$L = \frac{1}{m}\sum_{i} L_i$$

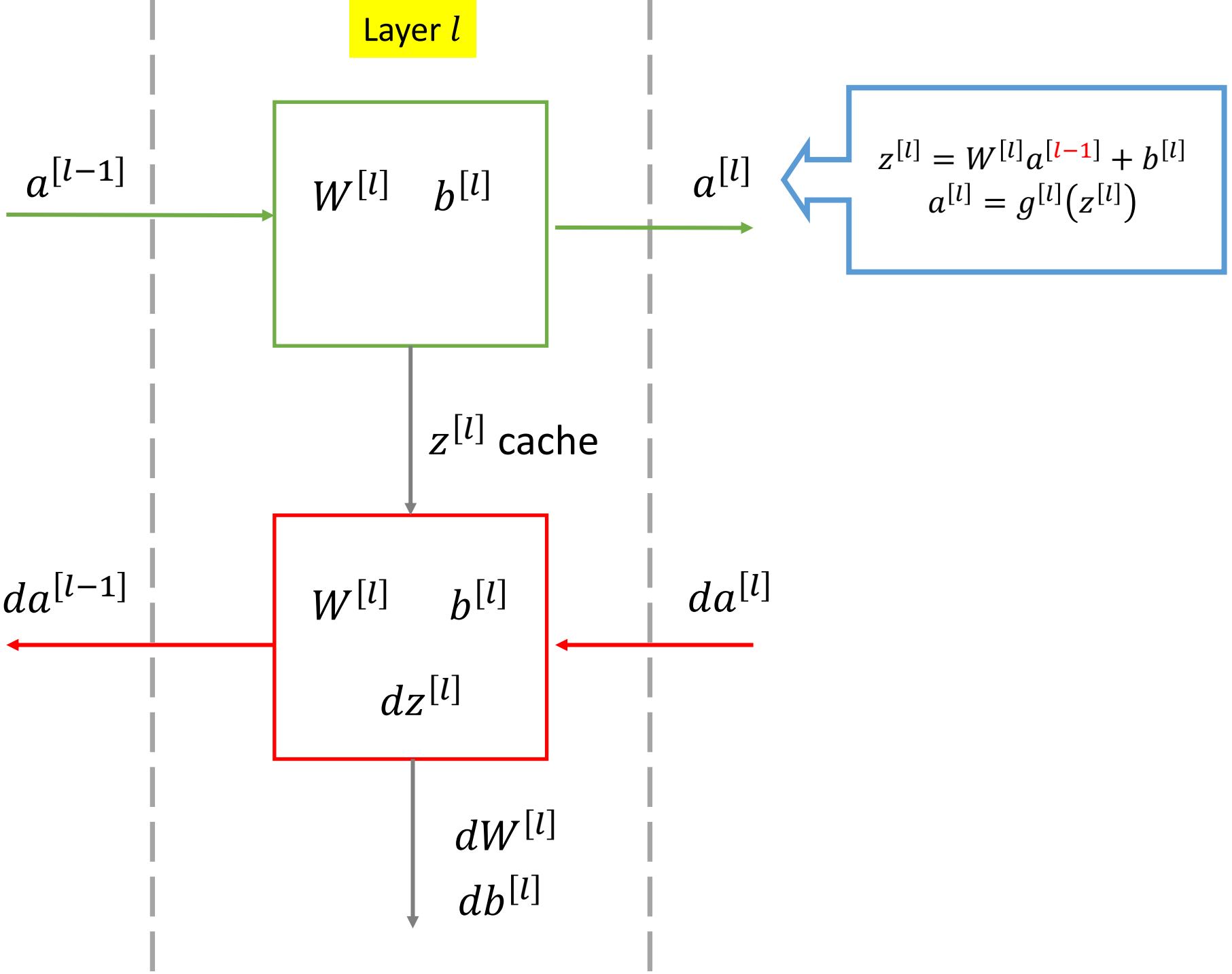
  - Compute derivatives

$$\partial L$$

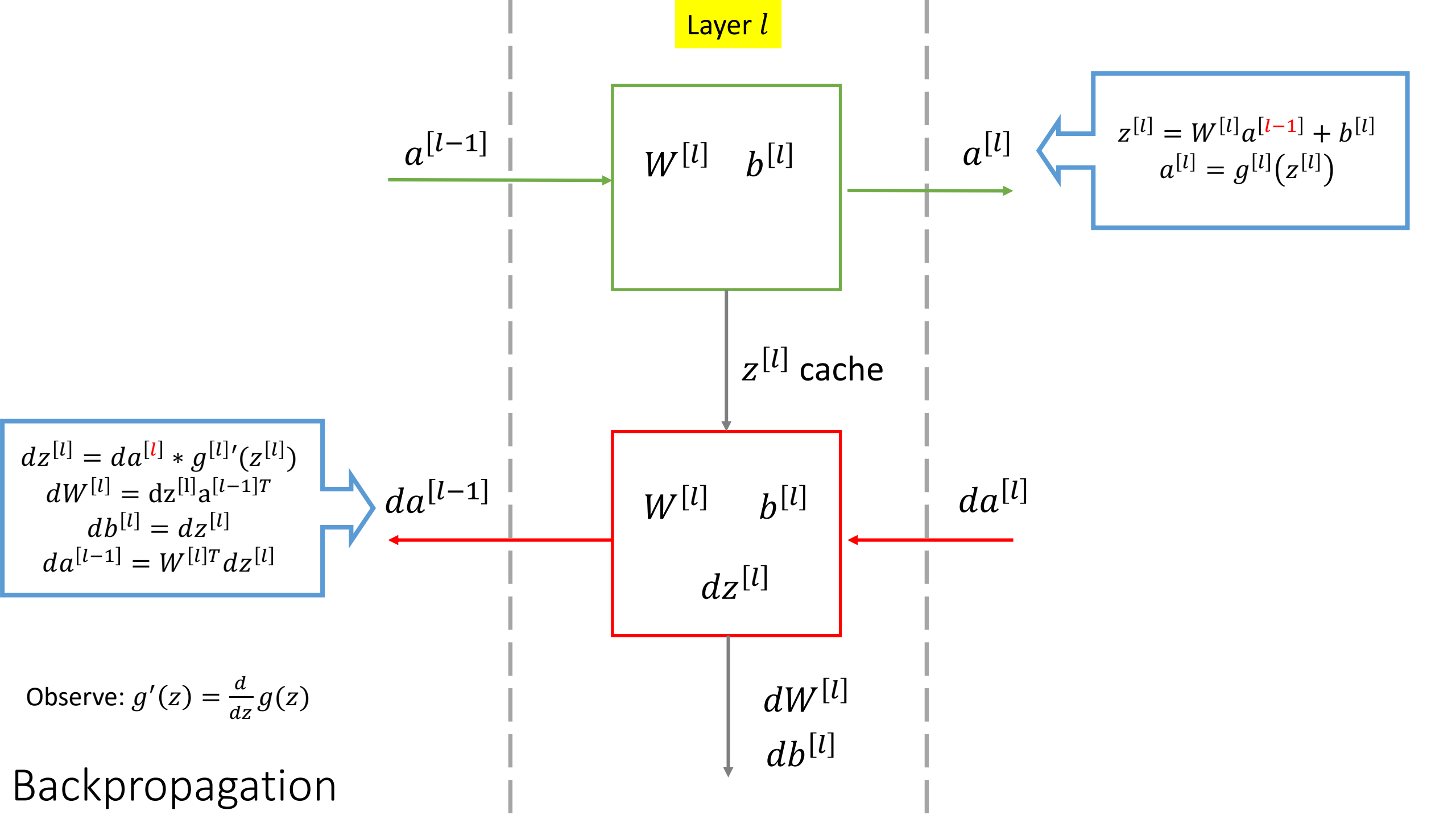
  - Adjust weights using the weights update



Backpropagation

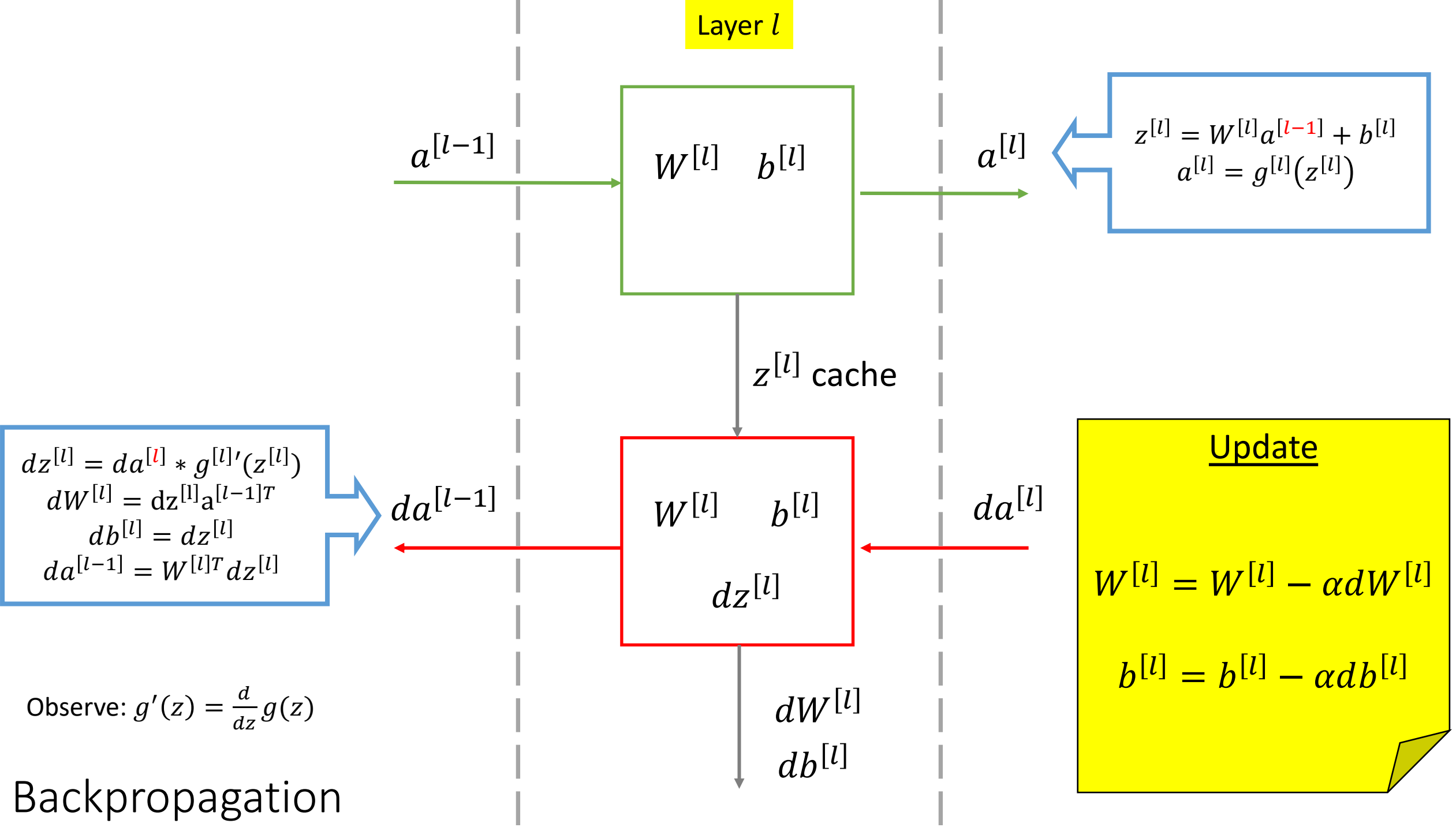


Backpropagation



Observe: $g'(z) = \frac{d}{dz}g(z)$

Backpropagation



Backpropagation

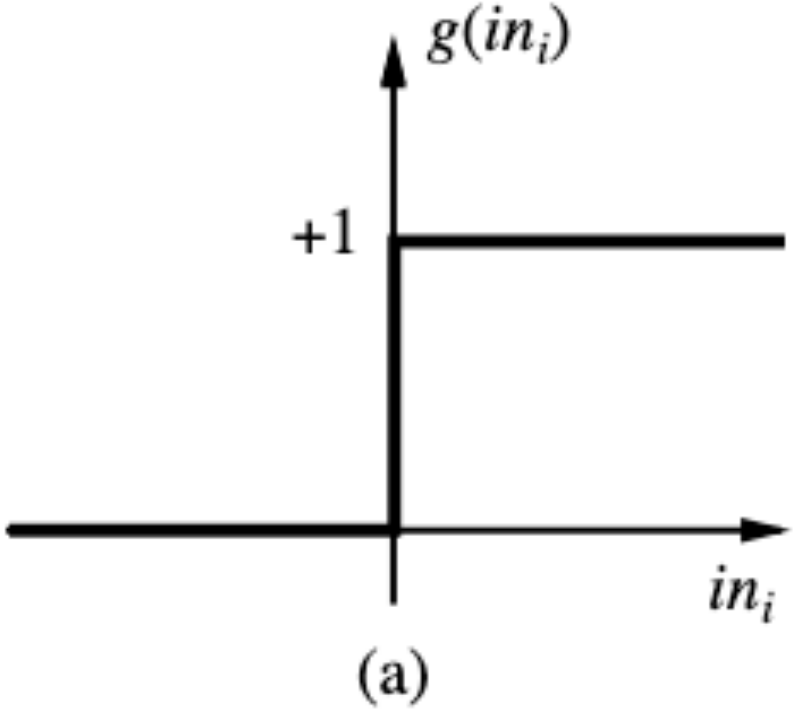
Activation Functions

- In each layer a function g is applied to the product

$$z^{[l]} = W^{[l]}A^{[l-1]} + b^{[1]}$$

$$g^{[l]}\left(z^{[l]}\right)$$

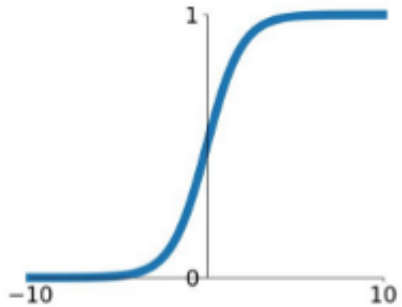
- In MLP g is the step function
- The step function contains only flat segment => no gradient



Activation Functions

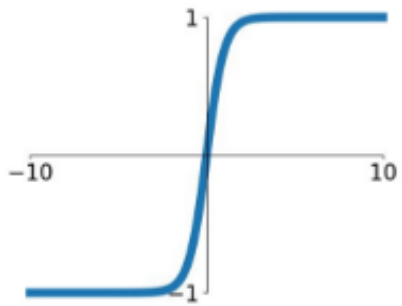
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



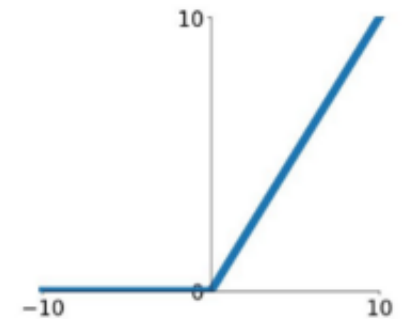
tanh

$$\tanh(x)$$



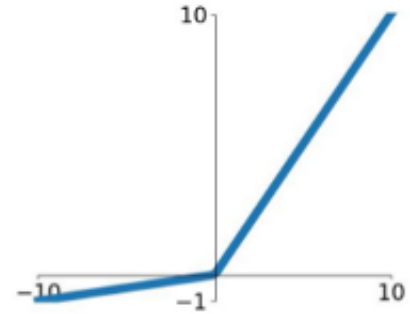
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

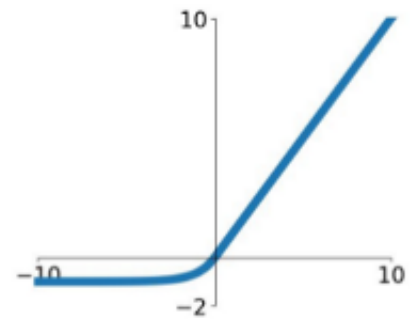


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



https://medium.com/@shrutijadon10104776/survey-on-activation-functions-for-deep-learning-9689331ba092

Introduction to TensorFlow/Keras

TensorFlow

- www.tensorflow.org
- Released under open source license on 9th Nov 2015
- Version 2.4.0
- Open source library for numerical computation using data flow graph
- Originally developed by Google Brain Team
- Provides an extensive suite of functions to build various model from scratch

TensorFlow Main Concepts

N-dimensional vector

Used to represent n-dimensional dataset

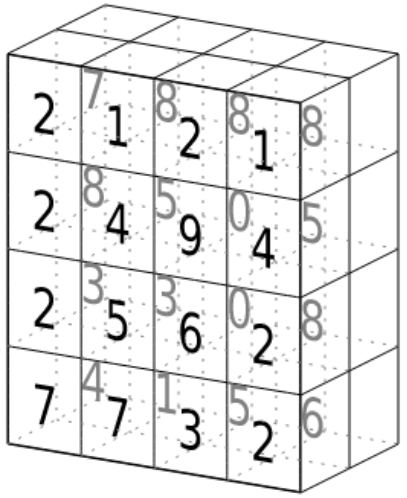
tensor

't'
'e'
'n'
's'
'o'
'r'

tensor of dimensions [6]
(vector of dimension 6)

3	1	4	1
5	9	2	6
5	3	5	8
9	7	9	3
2	3	8	4
6	2	6	4

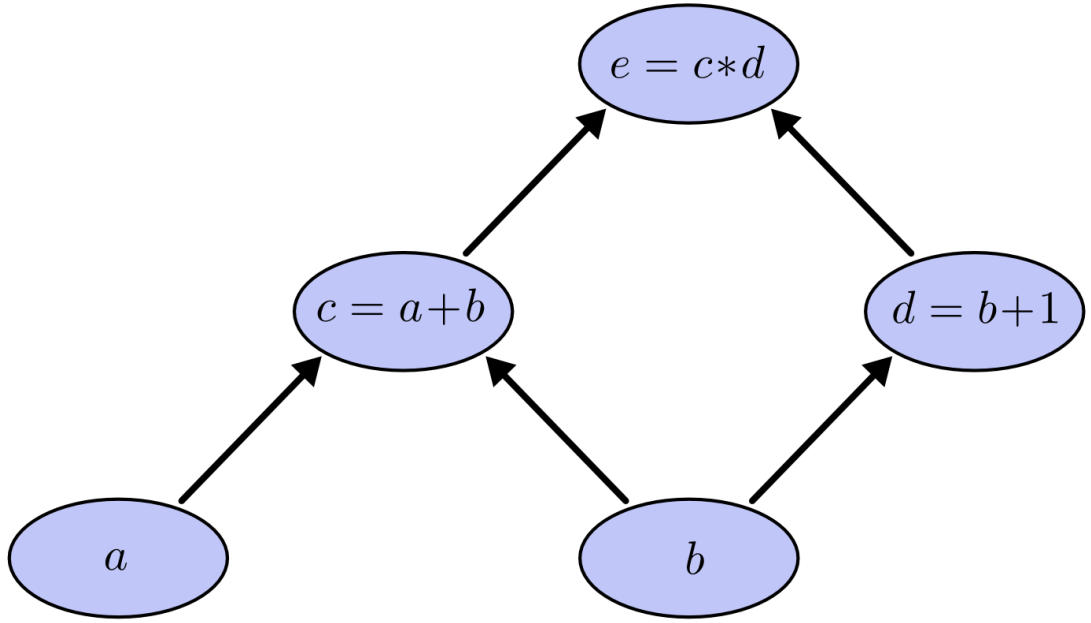
tensor of dimensions [6,4]
(matrix 6 by 4)



tensor of dimensions [4,4,2]

TensorFlow Main Concepts: Computational Graphs flow

- TensorFlow separates the definition of the computations from the execution
- The computation is based on the computational data flow graph
- Leaf and start vertices are always tensors
- Complex operations are represented in hierarchical order
- Traversing the graph in reverse order results in sub-expressions
- Operations at the same level are independent

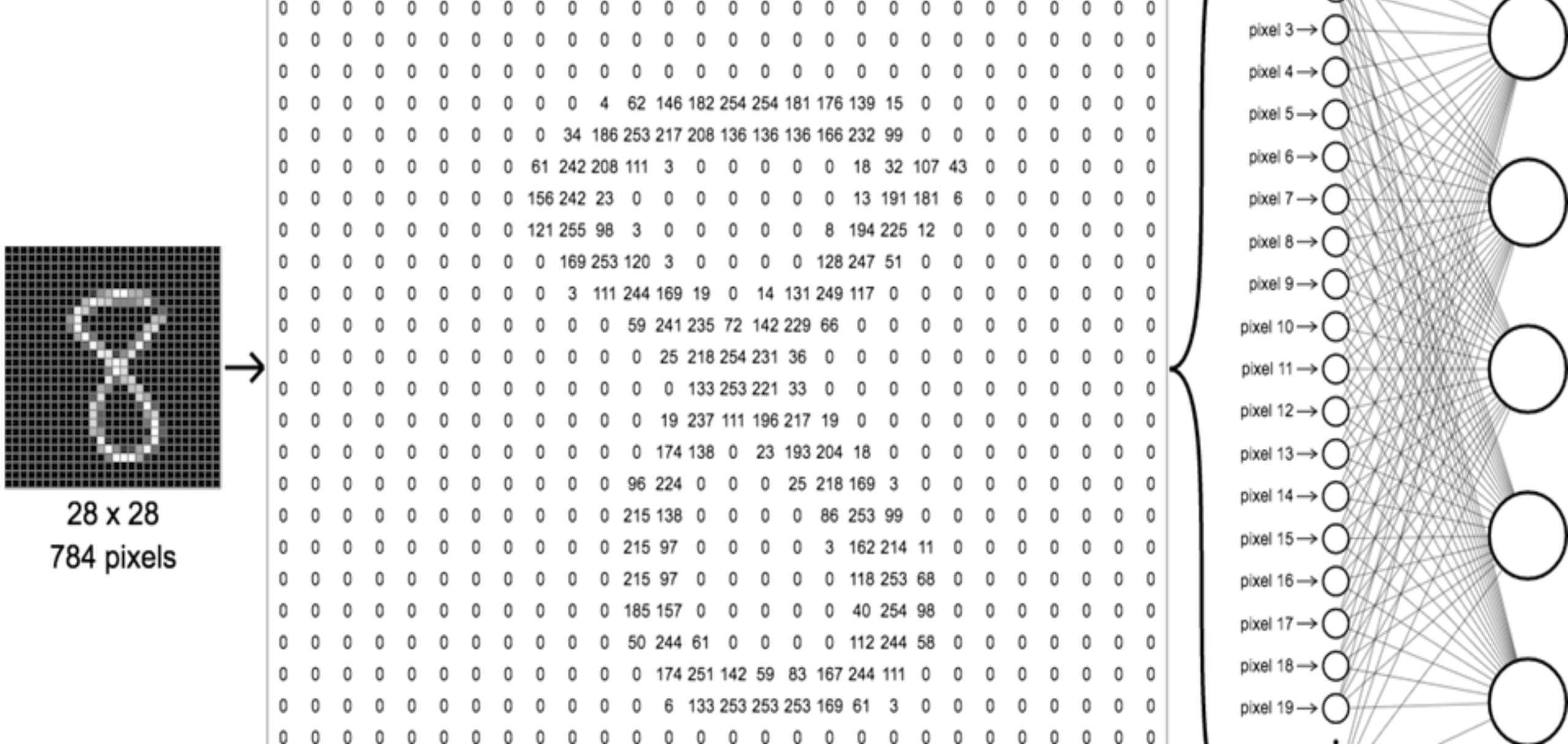


Keras

- www.keras.io
- Minimalist, highly modular, neural network library
- Written in Python
- Run on the top of TensorFlow/Theano and CNTK
- Focus on running fast experiments
- Fast prototyping
- Support convolutional and recurrent networks, and their combination
- Support arbitrary connectivity schema (multi-input, multi-output)
- Run seamlessly on GPU/CPU

Image classification in Keras

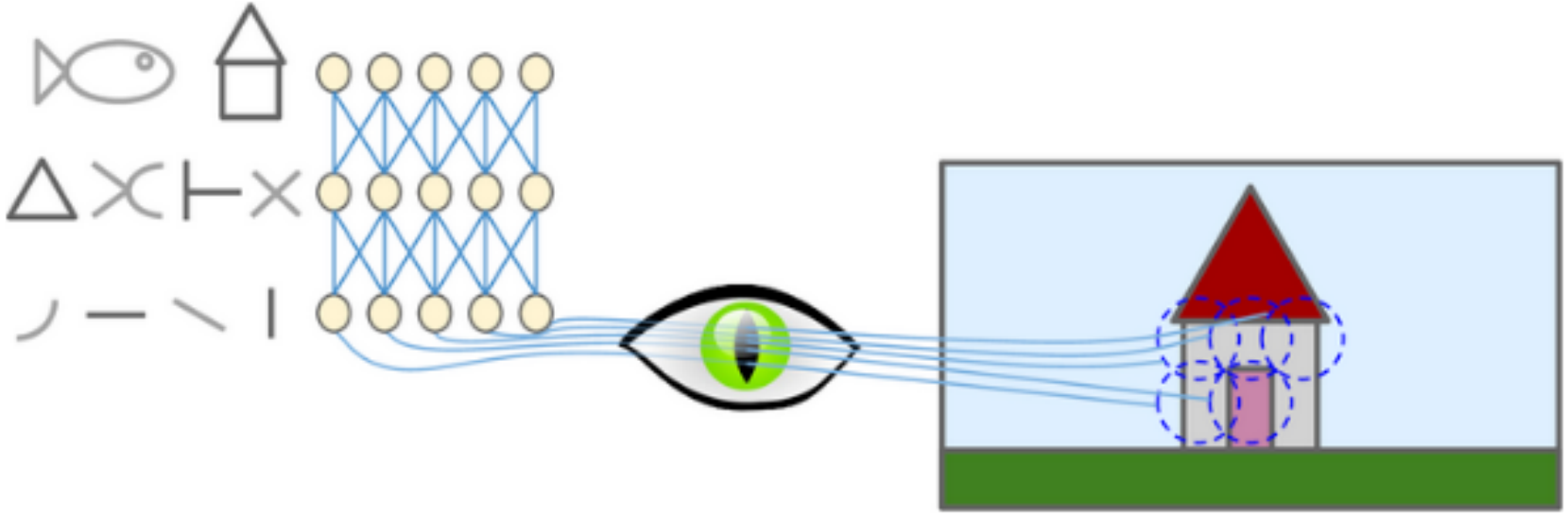
- Fashion MNIST dataset
- 60000 instances
- 28 x 28 images
- 10 categories



Convolutional Neural Network

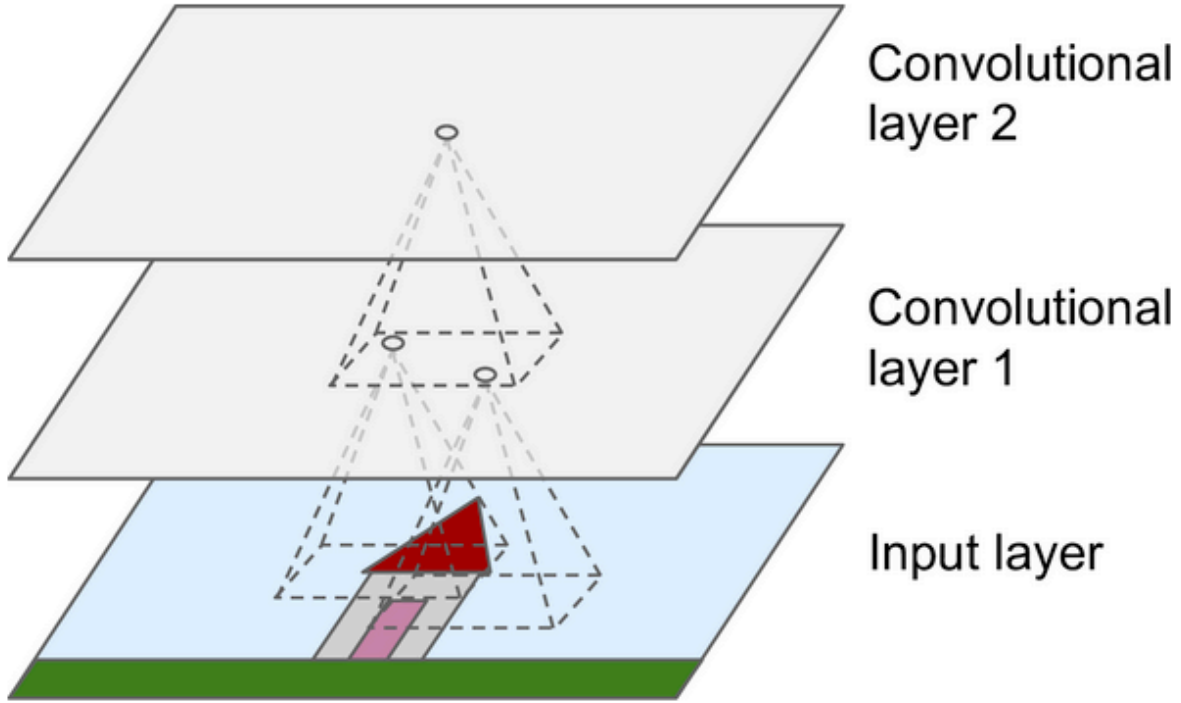
Visual Cortex

- In 1958-1959 Hubel & Wiesel showed some neurons in the visual cortex have *local receptive field*
- Some neurons react only to vertical horizontal lines
- … other to lines with different orientation
- … some react to complex patterns



Convolutional layer

- Idea
  - Neurons in the early layer of the network extract *local* visual features
  - Neurons in the later layers combine these features to extract higher-order features
- Sparse connectivity
- Parameter sharing



Less parameters than fully connected layer!

CNN

cat dog ……

Convolution

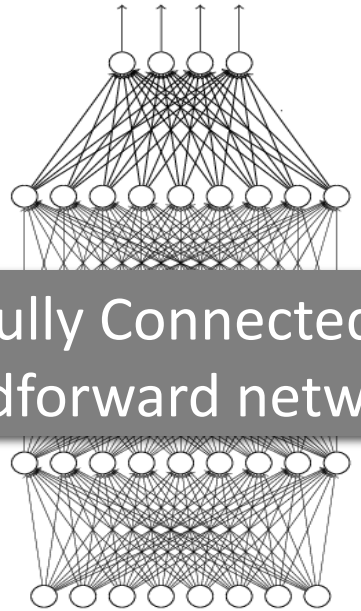
Max Pooling

Convolution

Max Pooling

Can repeat many times

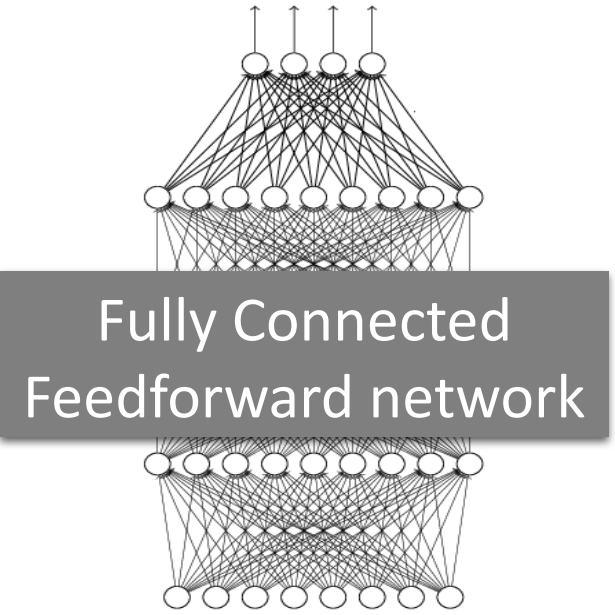
Flatten



Fully Connected Feedforward network

http://speech.ee.ntu.edu.tw/~tlkagk/courses/ML_2017/Lecture/CNN.pptx

CNN



1. Some pattern are much smaller than the whole image

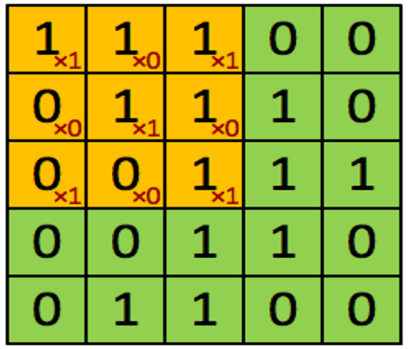
2. Same patterns appear in different regions

3. Subsampling the pixel will not change the object

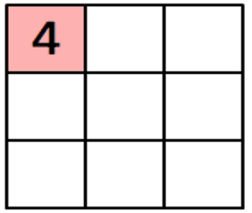
http://speech.ee.ntu.edu.tw/~tlkagk/courses/ML_2017/Lecture/CNN.pptx

Filters

- Neuron weights represented as a small image the size of receptive field
- If neurons at the same layer use all the same filter they will output a *feature map*
- No need to define filters, the convolutional layer learns them!

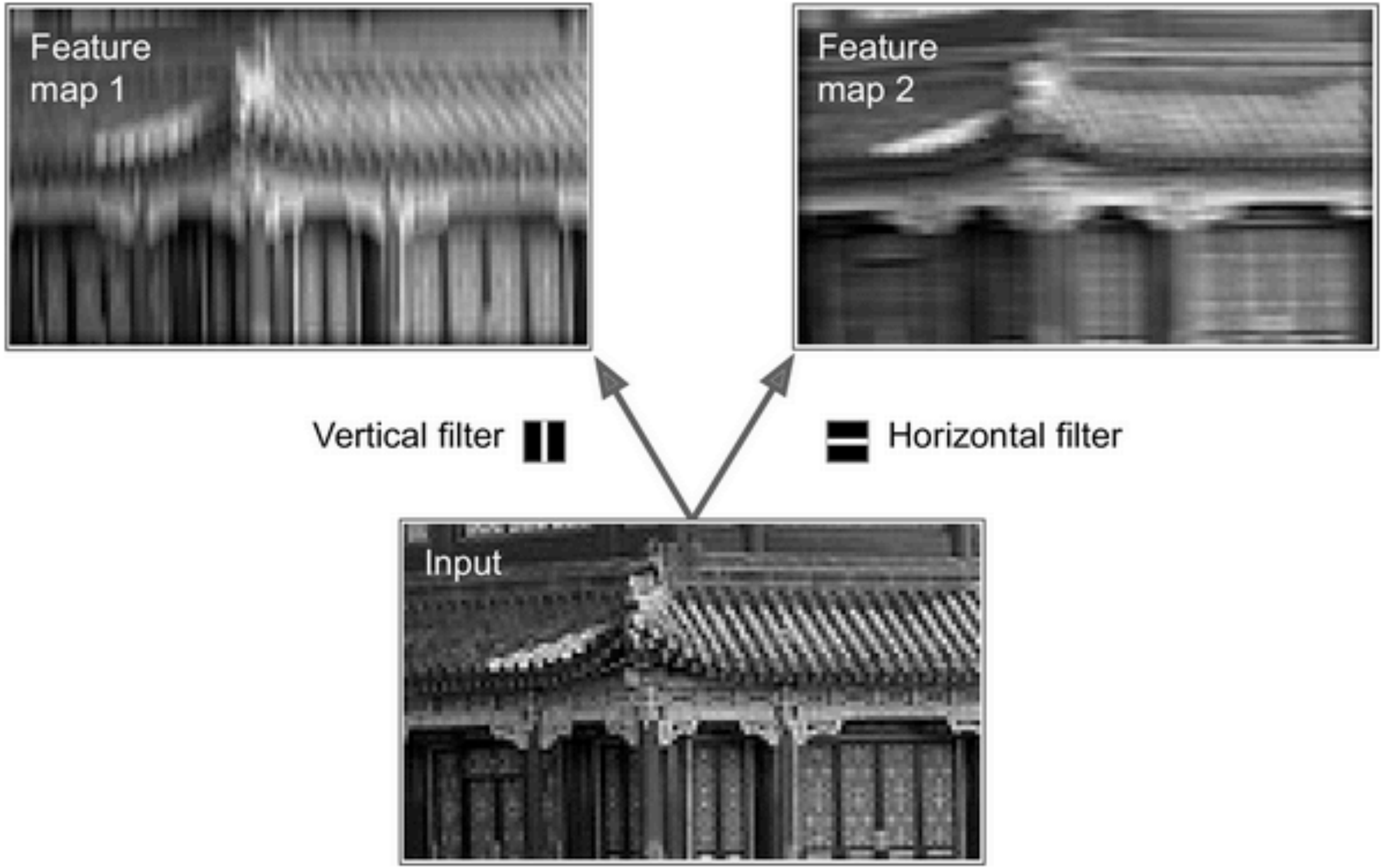


Image



Convolved Feature

http://cs229.stanford.edu/section/cs229_deep_learning_friday.pptx


Feature
map 1
Feature
map 2
Vertical filter
Horizontal filter
Input

Filters

Stride = 1

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

6 x 6 image

*

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

=

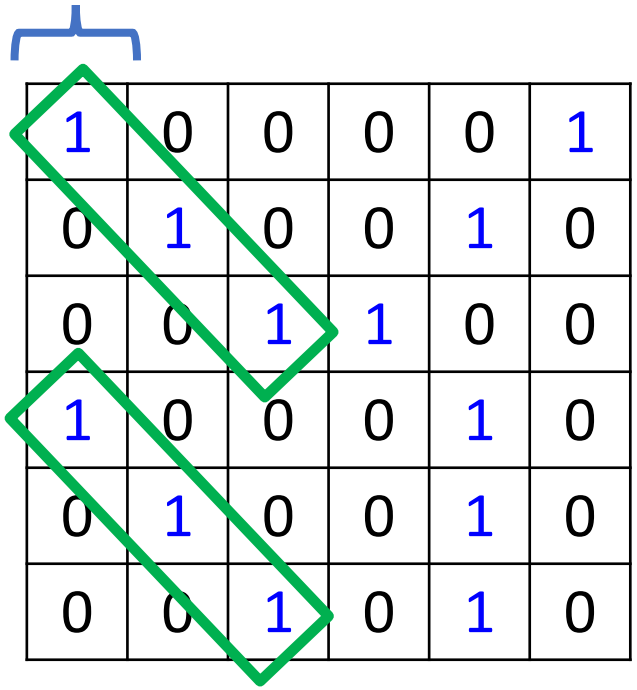
3	-1	-3	-1
-3	1	0	-3
-3	-3	0	1
3	-2	-2	-1

Repeat for every filter

1. Some pattern are much smaller than the whole image
=>
Each filter detects a small pattern (3x3)

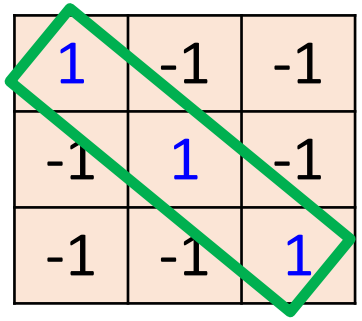
Filters

Stride = 1



6 x 6 image

*



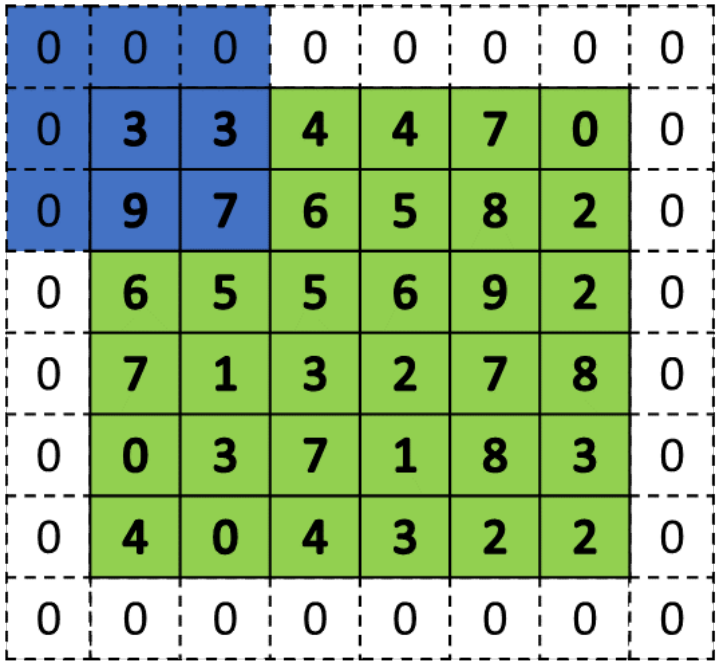
Filter 1

=

3	-1	-3	-1
-3	1	0	-3
-3	-3	0	1
3	-2	-2	-1

2. Same patterns appear in different regions

Padding



$6 \times 6 \rightarrow 8 \times 8$

$*$

1	0	-1
1	0	-1
1	0	-1

3×3

$=$

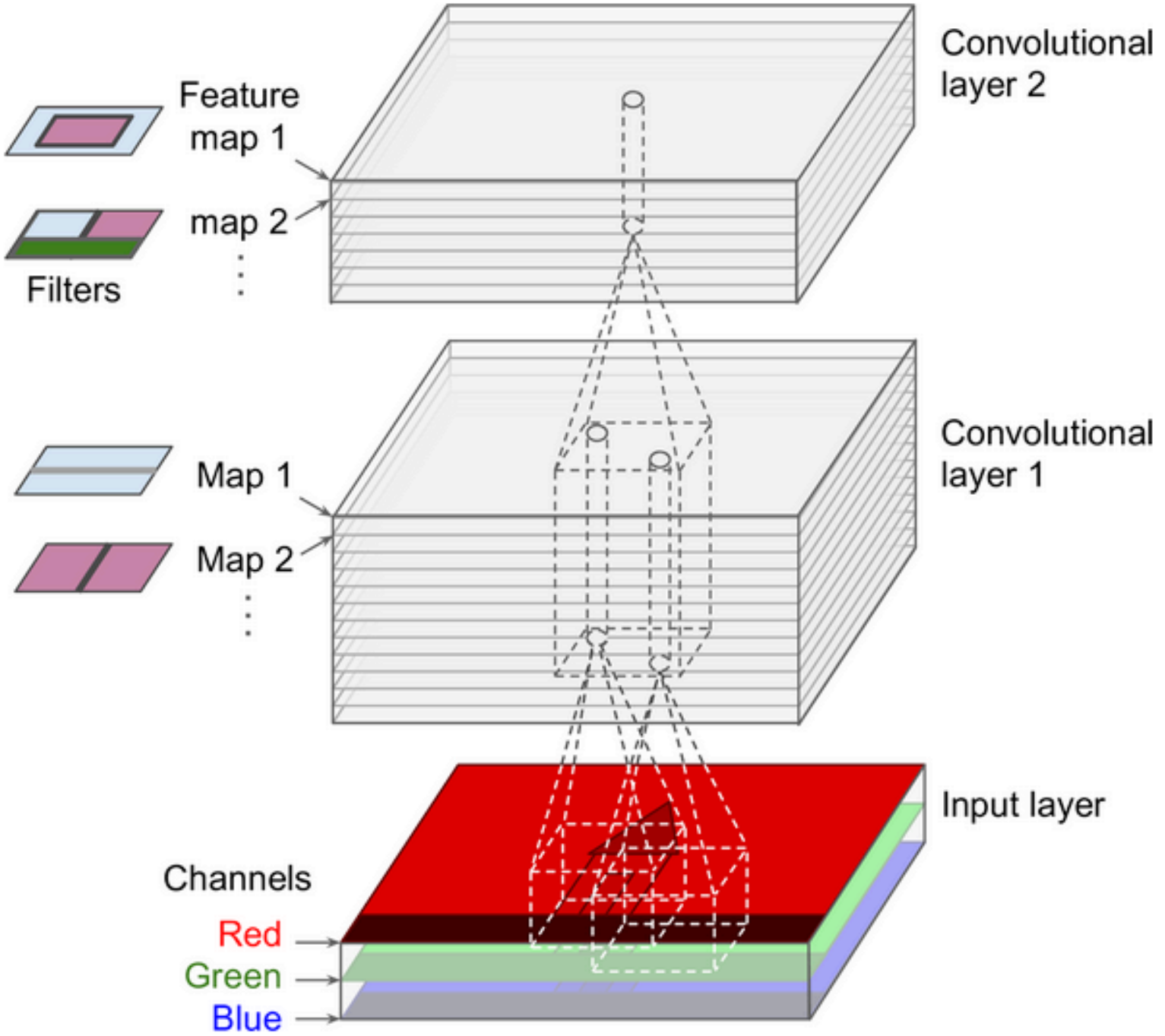
-10	-13	1			
-9	3	0			

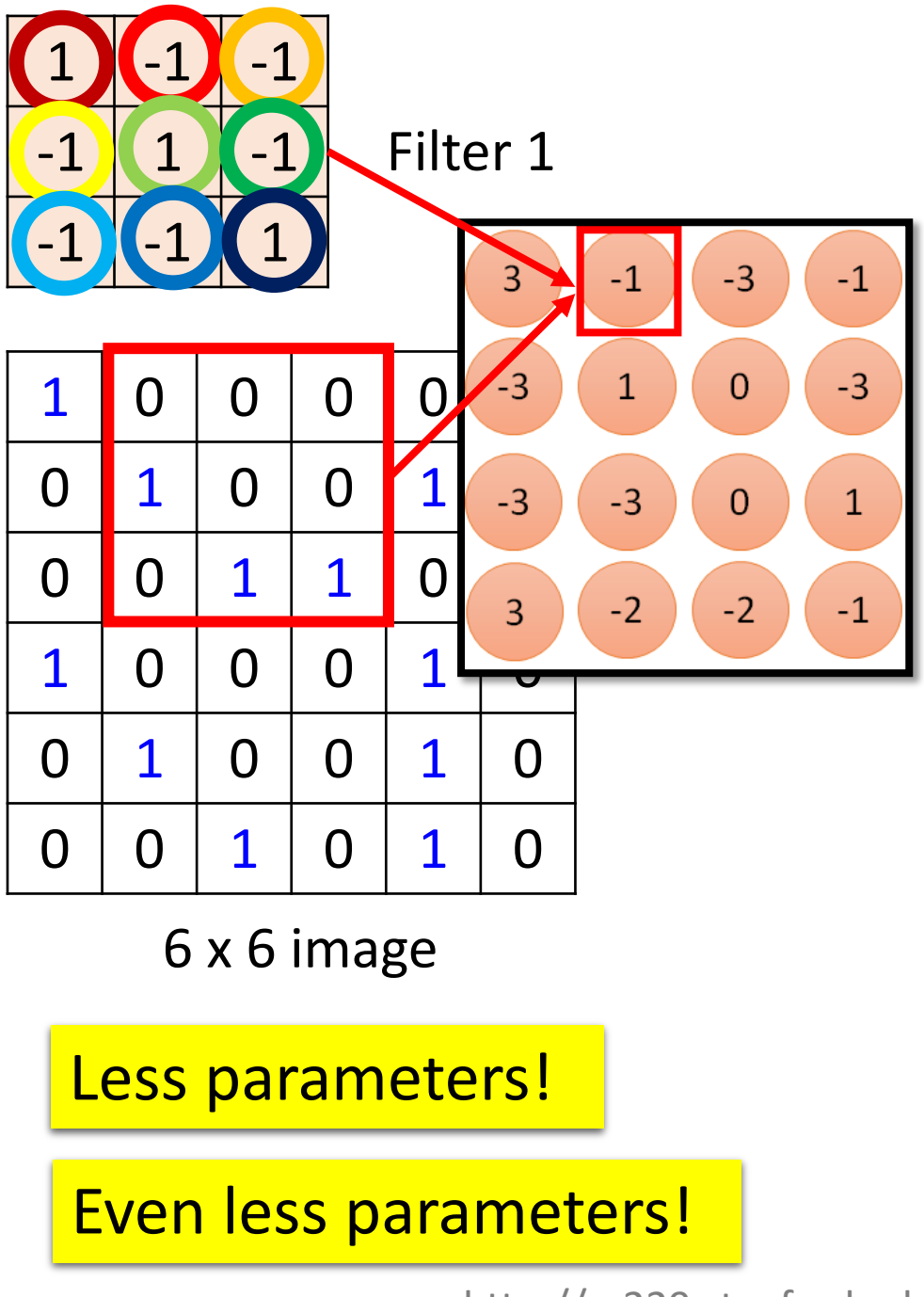
6×6

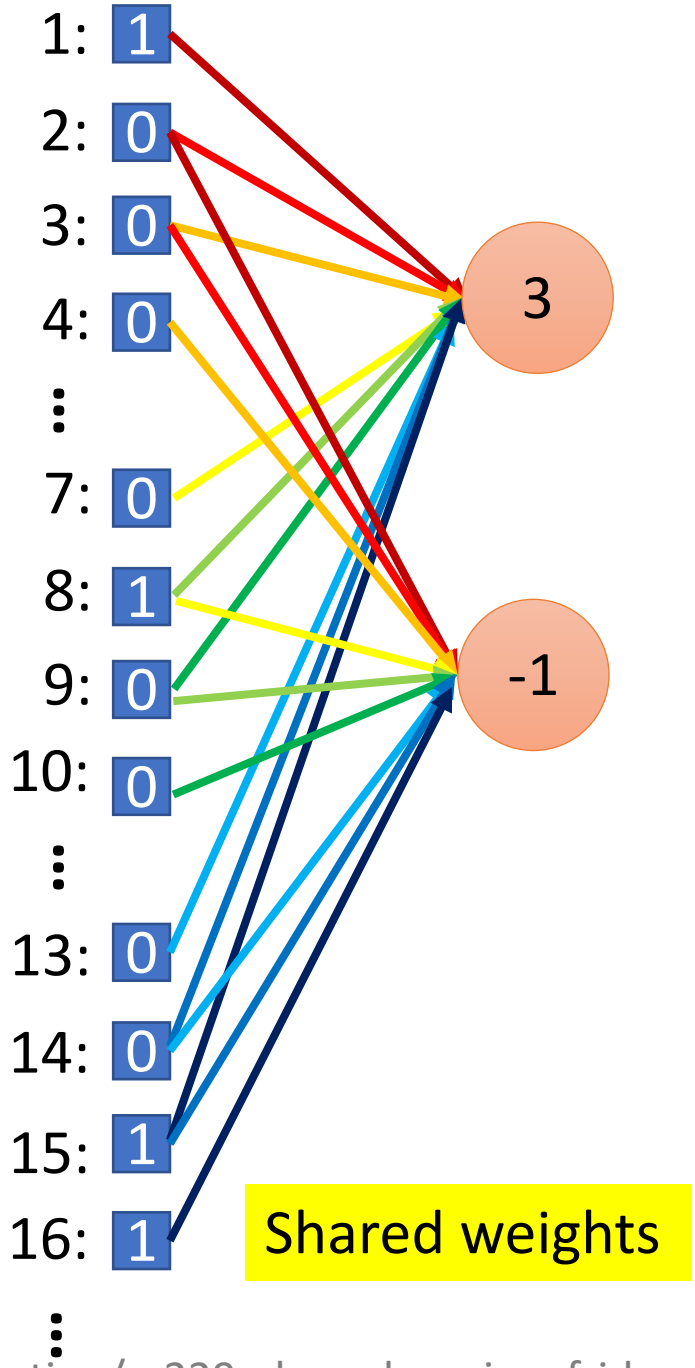
http://datahacker.rs/what-is-padding-cnn/

- Input images are usually in three channels: RGB
- Convolutional layer has multiple filters
- 1 feature map x filter

  => 3D
- Neurons in different feature maps use different parameter
- A neuron receptive field extends across all the previous layer of the feature maps
  => applies multiple trainable filters to its input
- Translational invariance







Less parameters!

Even less parameters!

Shared weights

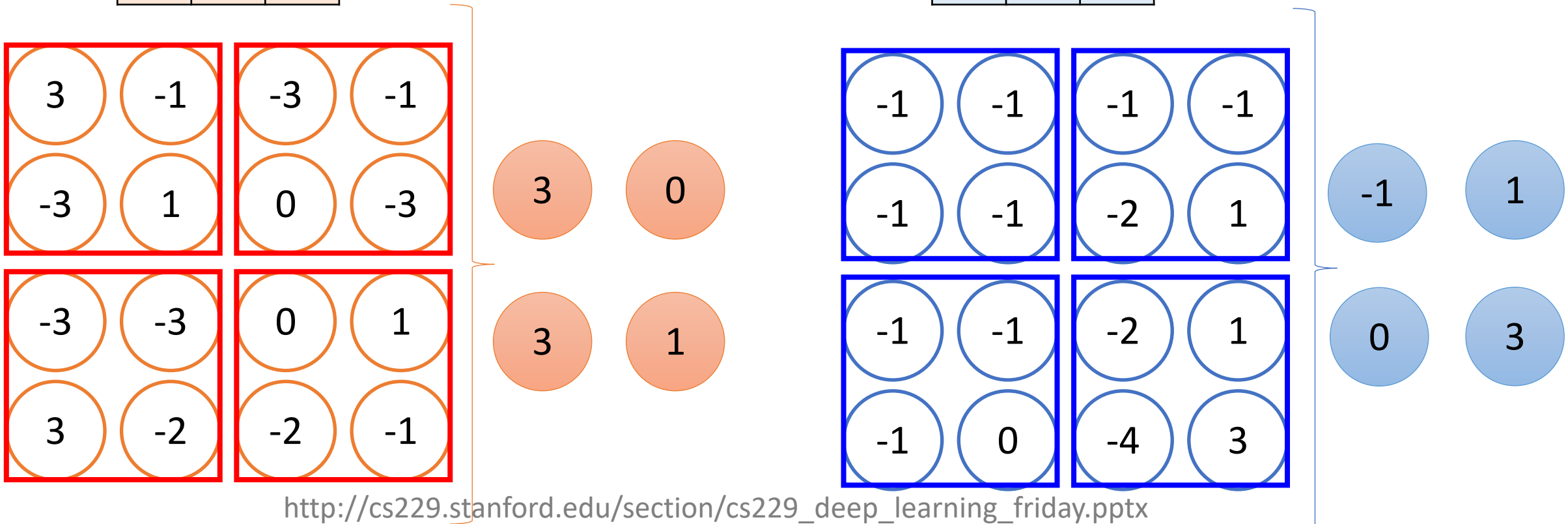
Max Pooling

1	-1	-1
-1	1	-1
-1	-1	1

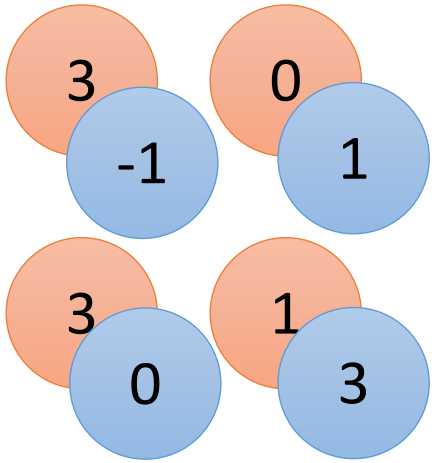
Filter 1

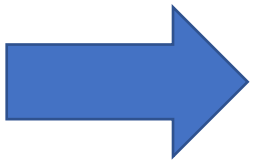
-1	1	-1
-1	1	-1
-1	1	-1

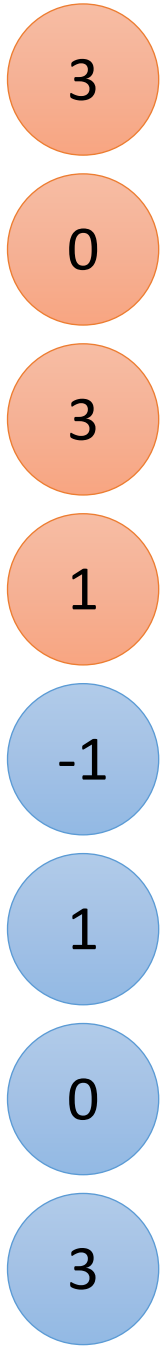
Filter 2

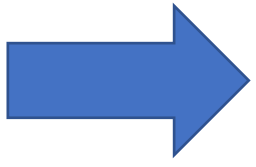


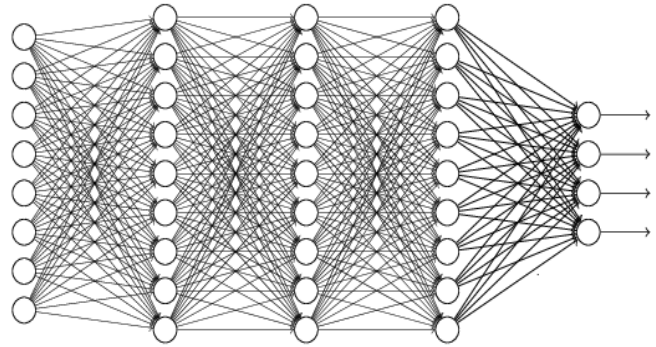
Flatten







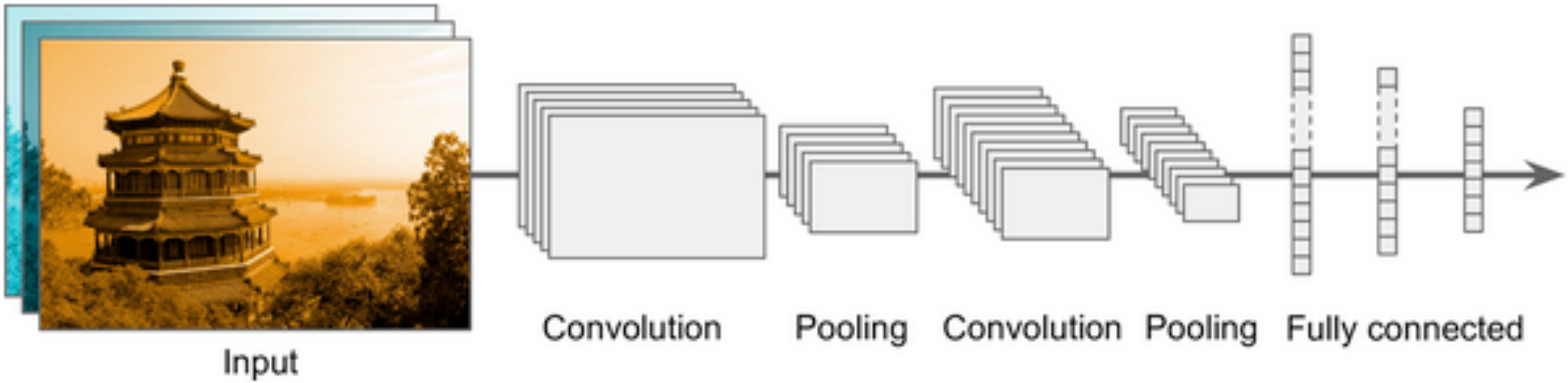




Fully Connected Feedforward network

http://cs229.stanford.edu/section/cs229_deep_learning_friday.pptx

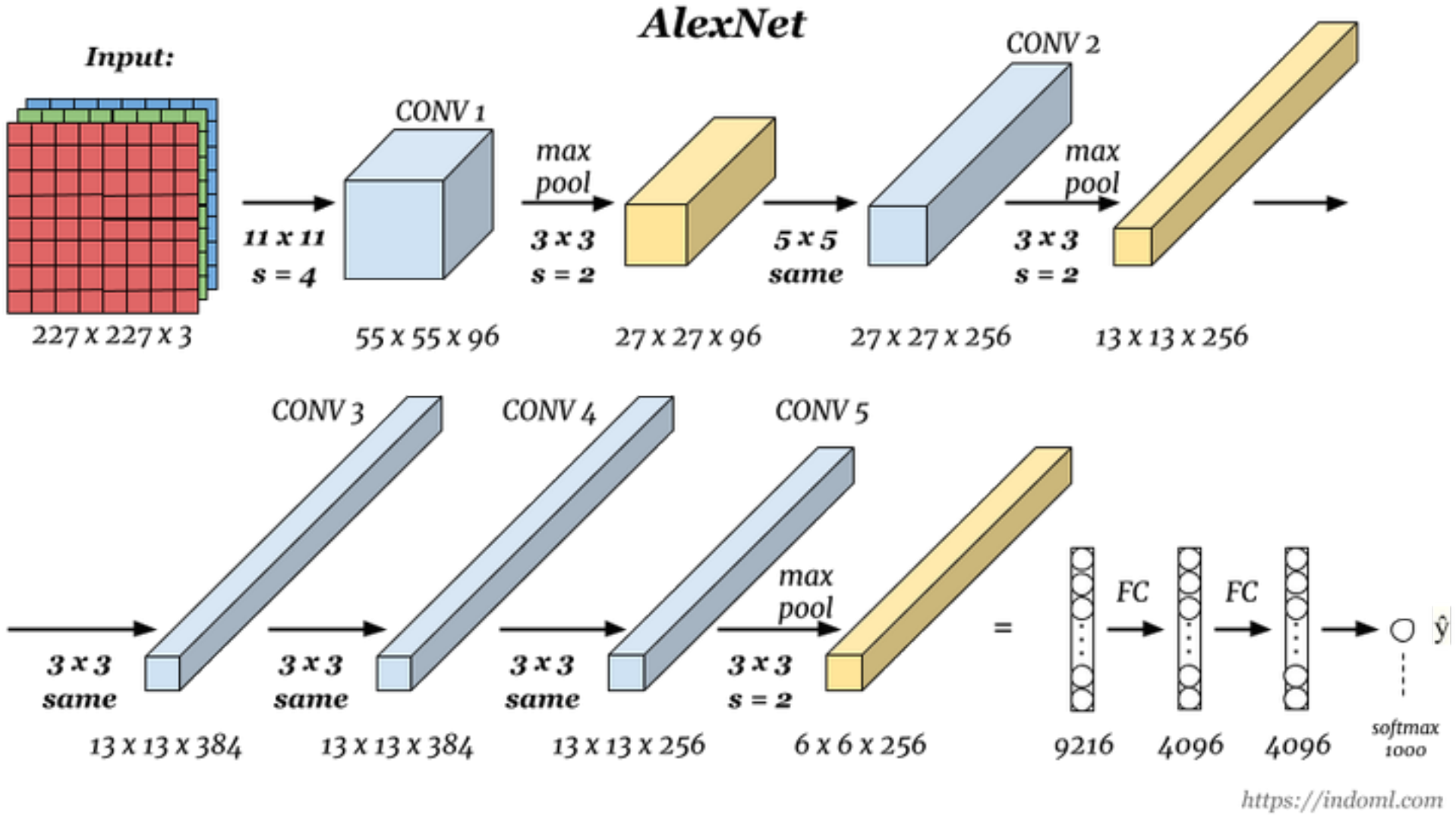
Recap


Input
Convolution
Pooling
Convolution
Pooling
Fully connected

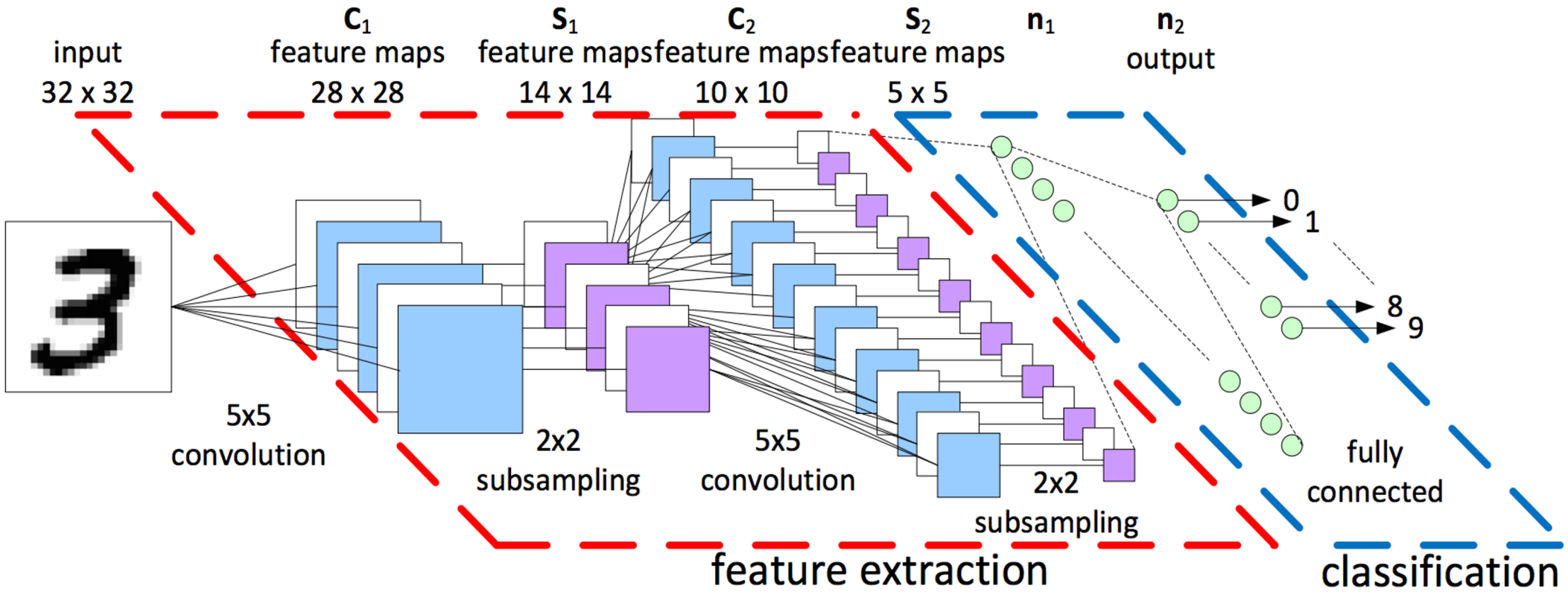
Recap

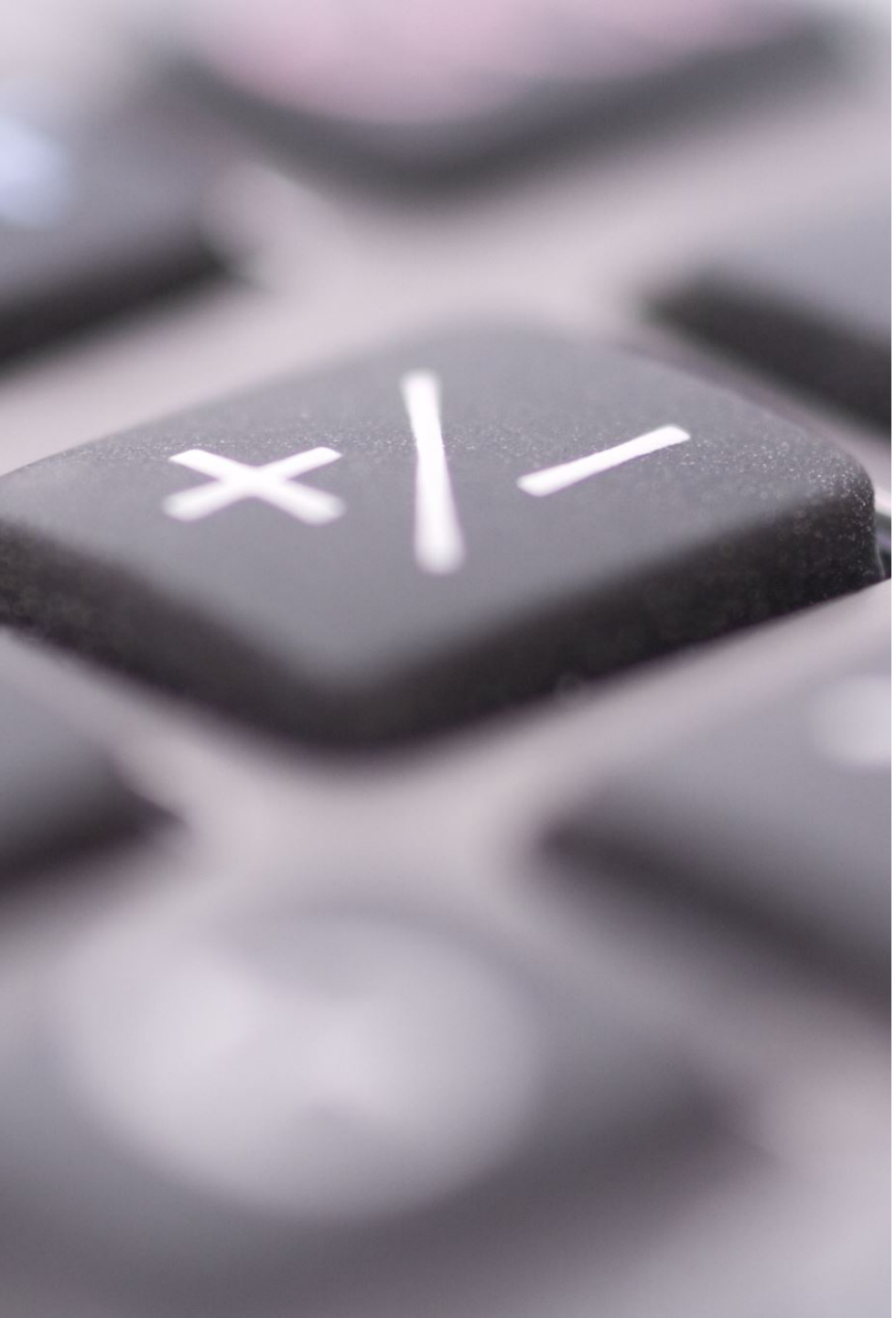
- Number of channel (e.g. RGB) in input must match the filter
- Multiple filters can be used in a convolutional layer to detect multiple features
- The output of the layer will have the same number of channel as the number of filter
- Finally to make up a convolutional layer we add the bias and the activation function (ReLU or tanh)
- Notation
  - f filter
  - s stride
  - p padding

AlexNet


AlexNet
Input:
227 x 227 x 3
11 x 11
s = 4
CONV 1
55 x 55 x 96
max
pool
3 x 3
s = 2
27 x 27 x 96
5 x 5
same
CONV 2
27 x 27 x 256
max
pool
3 x 3
s = 2
13 x 13 x 256
3 x 3
same
CONV 3
13 x 13 x 384
3 x 3
same
CONV 4
13 x 13 x 384
3 x 3
same
13 x 13 x 256
3 x 3
s = 2
CONV 5
max
pool
6 x 6 x 256
=
9216
FC
4096
FC
4096
softmax
1000
ŷ
https://indoml.com

CNN in action





Resources

- http://neuralnetworksanddeeplearning.com/chap1.html
- Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow
- https://github.com/ageron/handson-ml2/
- http://cs229.stanford.edu/syllabus-spring2020.html
- https://colab.research.google.com/drive/1-6s9KHSa-vzrRqfq8yQ_urBeJu_G6zma?usp=sharing
- https://colab.research.google.com/drive/12IzTI1uHmzEG9ACQwfdl0lVo6W9NPfN1?usp=sharing