Convolutional Neural Networks and Optimization tricks

Frederic Precioso

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Recap

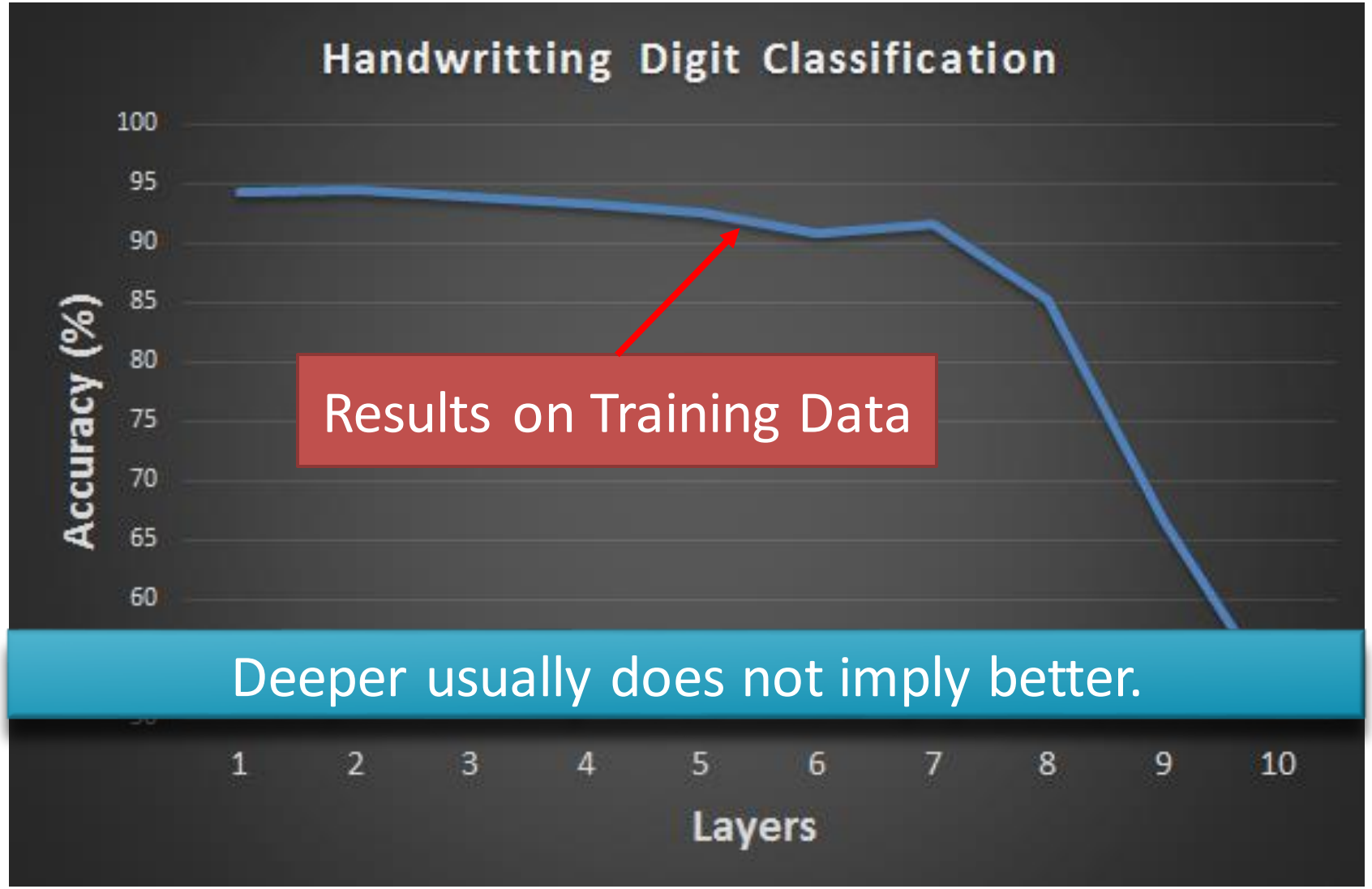
https://www.youtube.com/watch?v=IHZwWFHWa-w

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Recipe of Deep Learning

- ***Hard to get the power of Deep …***



UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Structure the network?

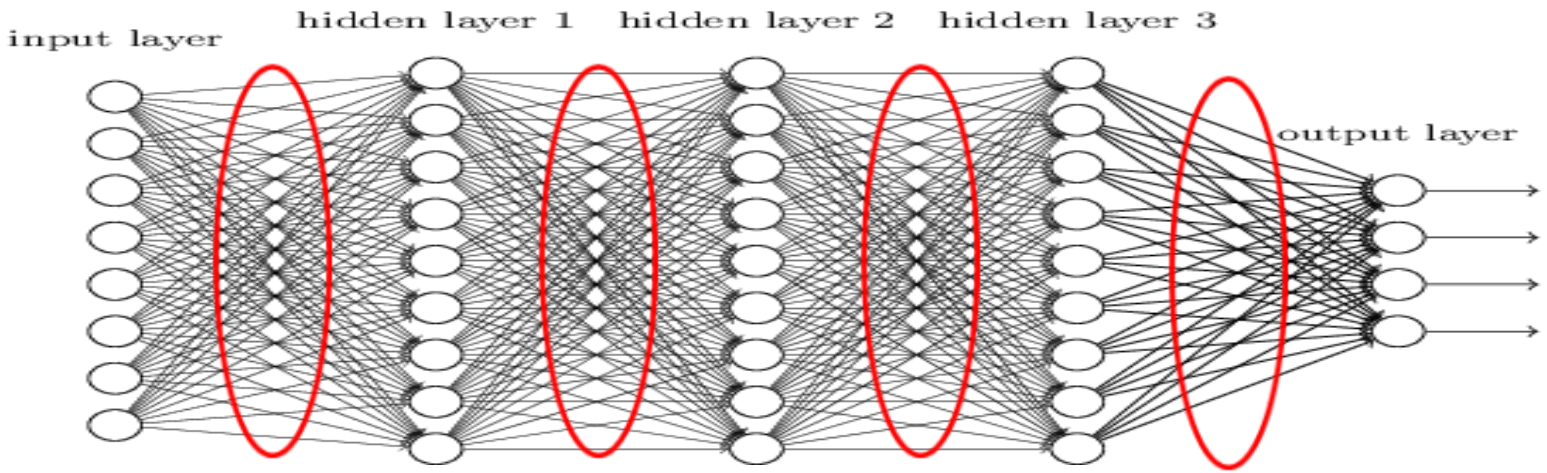
- Can we put any structure reducing the space of exploration and providing useful properties (invariance, robustness…)?

$$y = s(w_{13}\, s(w_{11}x_1 + w_{21}x_2 - w_{01}) + w_{23}\, s(w_{12}x_1 + w_{22}x_2 - w_{02}) - w_{03})$$

z_1 : $s(w_{11}x_1 + w_{21}x_2 - w_{01})$

z_2 : $s(w_{12}x_1 + w_{22}x_2 - w_{02})$

z_3 : $s(w_{13}\, z_1 + w_{23}\, z_2 - w_{03})$



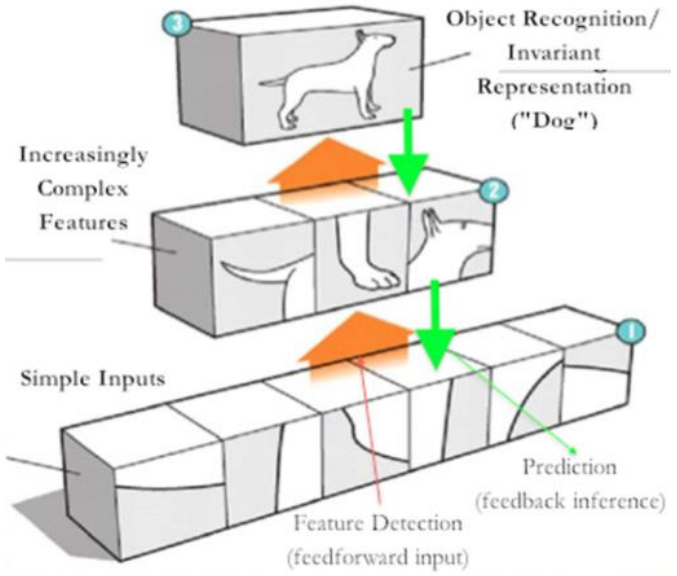
Invariance to spatial changes

Convolutional Neural Networks (aka CNN, ConvNet)

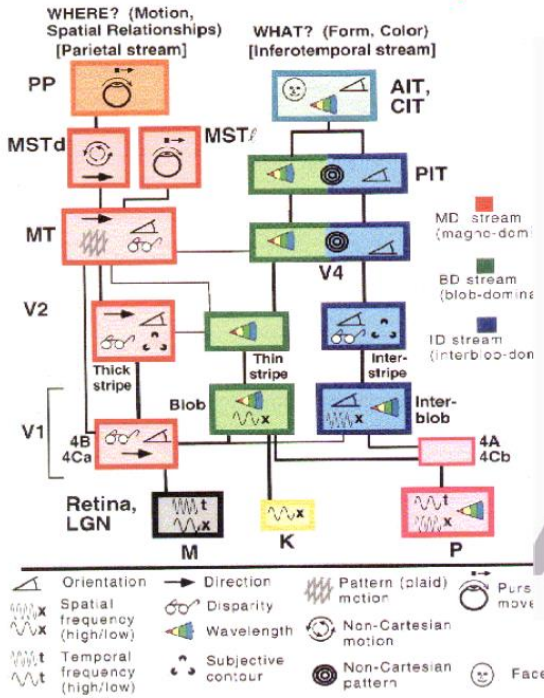
UNIVERSITÉ CÔTE D'AZUR

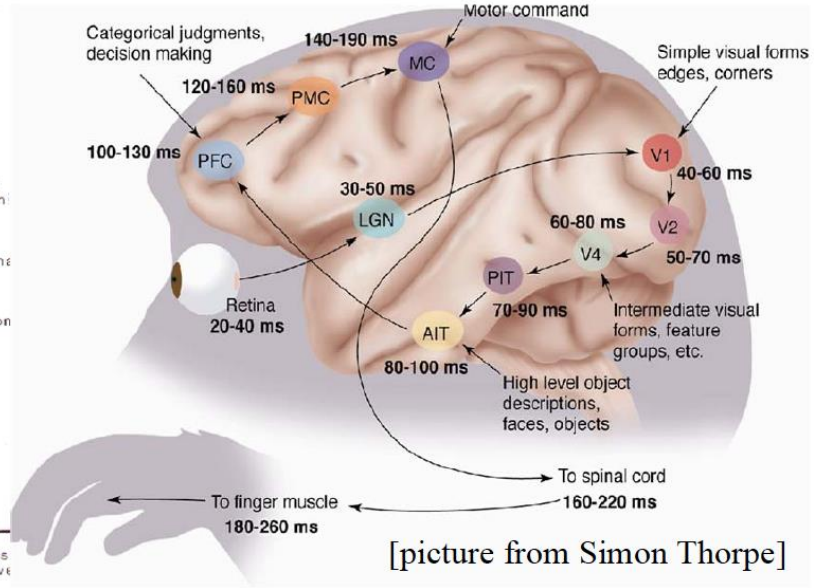
univ-cotedazur.fr

The Mammalian Visual Cortex Inspires CNN



- The ventral (recognition) pathway in the visual cortex has multiple stages
- Retina - LGN - V1 - V2 - V4 - PIT - AIT ....
- Lots of intermediate representations



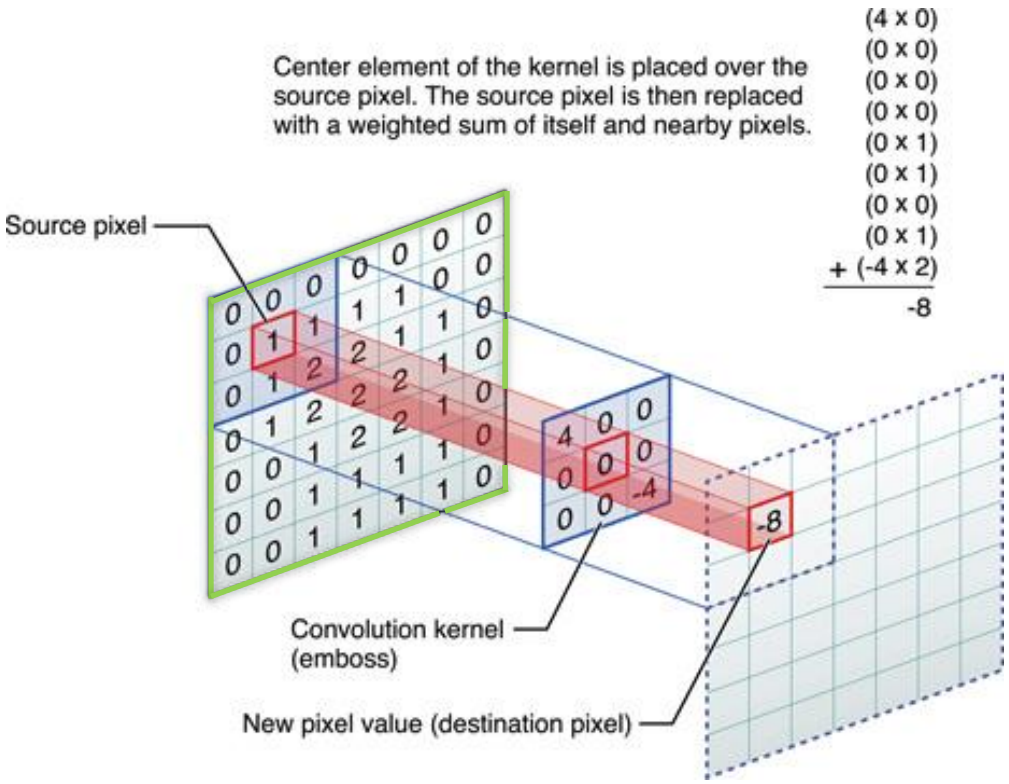


[Gallant & Van Essen]

UNIVERSITÉ CÔTE D'AZUR

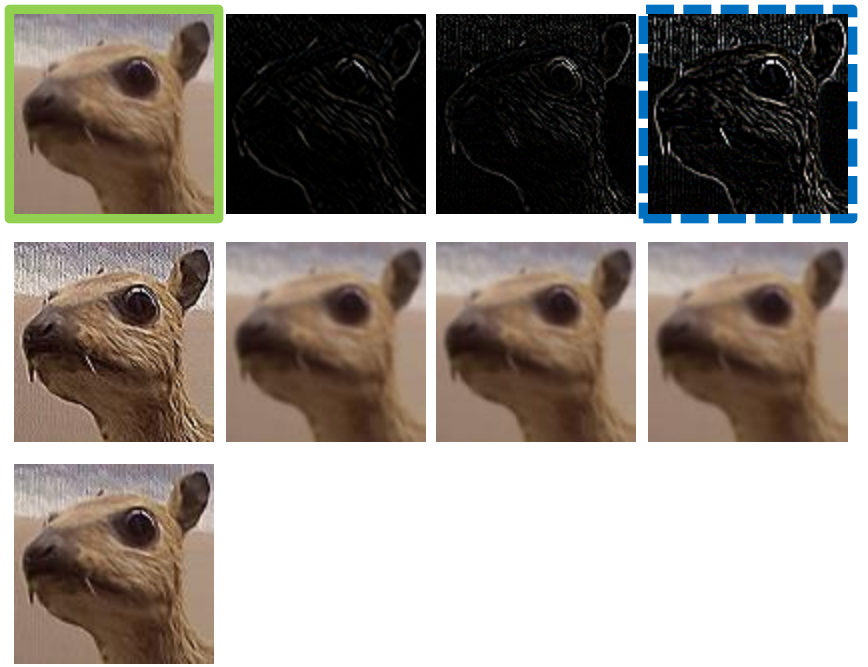
univ-cotedazur.fr

Image Convolution



Original Image

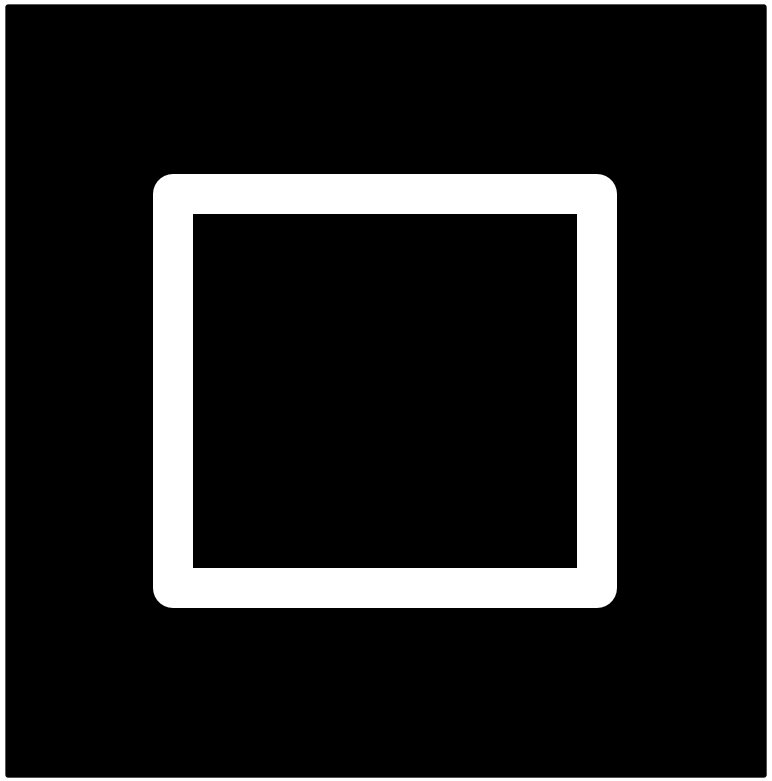
Convolved Image (here Edges)



UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Image Convolution



Image

-1
0
1

Filter to extract horizontal edges

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Image Convolution

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

Image

-1
0
1

Filter to extract horizontal edges

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Image Convolution

New Image

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

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

0	*	-1	$= (\mathbf{0} \times -1) + (\mathbf{0} \times 0) + (\mathbf{1} \times 1) = 1$
0		0	
1		1	

Filter

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Image Convolution

New Image

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

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

0	*	-1	=	$(\mathbf{0} \times -1) + (\mathbf{0} \times 0) + (\mathbf{1} \times 1) = 1$
0		0		
1		1		

Filter

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Image Convolution

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

New Image

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

0	-1	
0	0	
1	1	

Filter

$$= (\mathbf{0} \times -1) + (\mathbf{0} \times 0) + (\mathbf{1} \times 1) = 1$$

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Image Convolution

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

New Image

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

0	-1	
0	0	
1	1	

Filter

$$= (\mathbf{0} \times -1) + (\mathbf{0} \times 0) + (\mathbf{1} \times 1) = 1$$

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Image Convolution

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

New Image

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

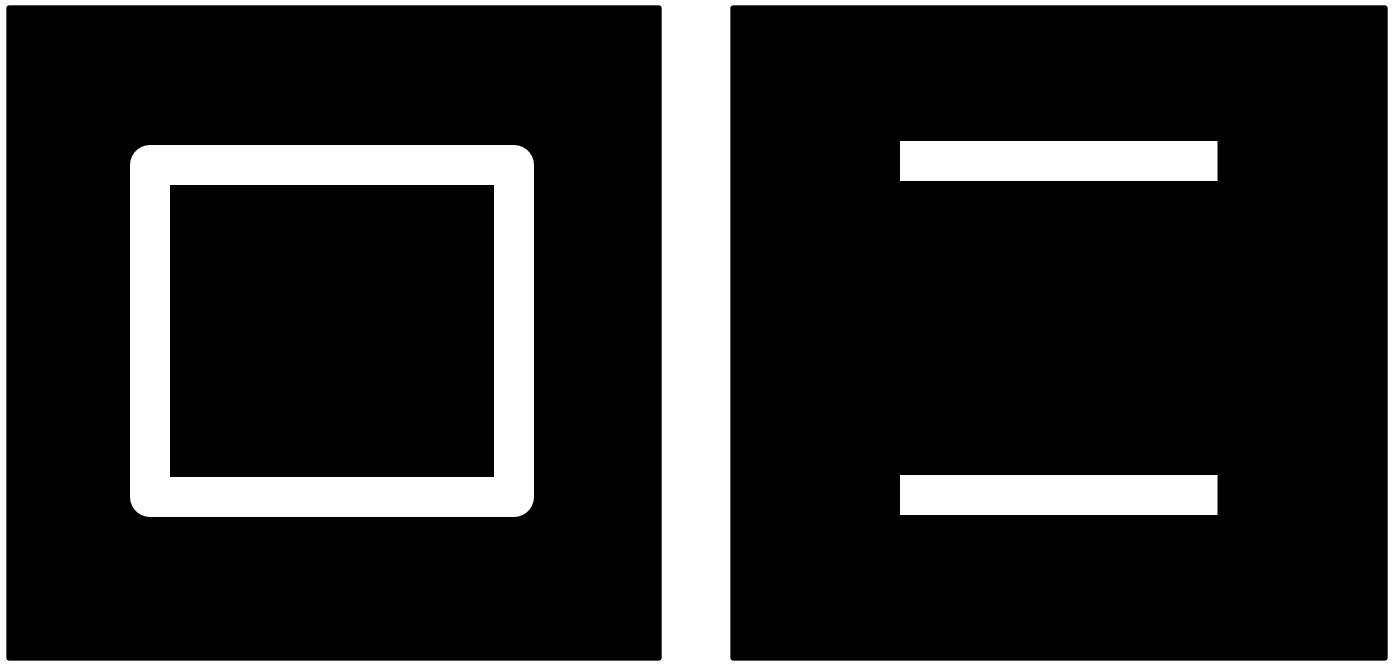
Horizontal Edge Detector

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

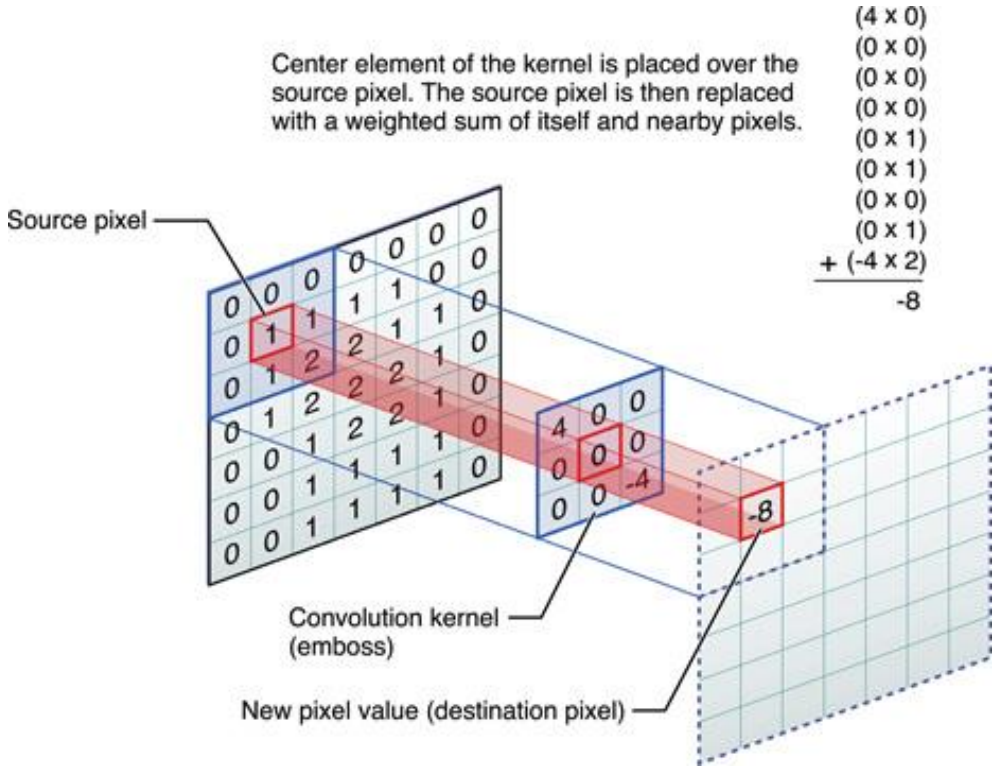
Image Convolution

New Image

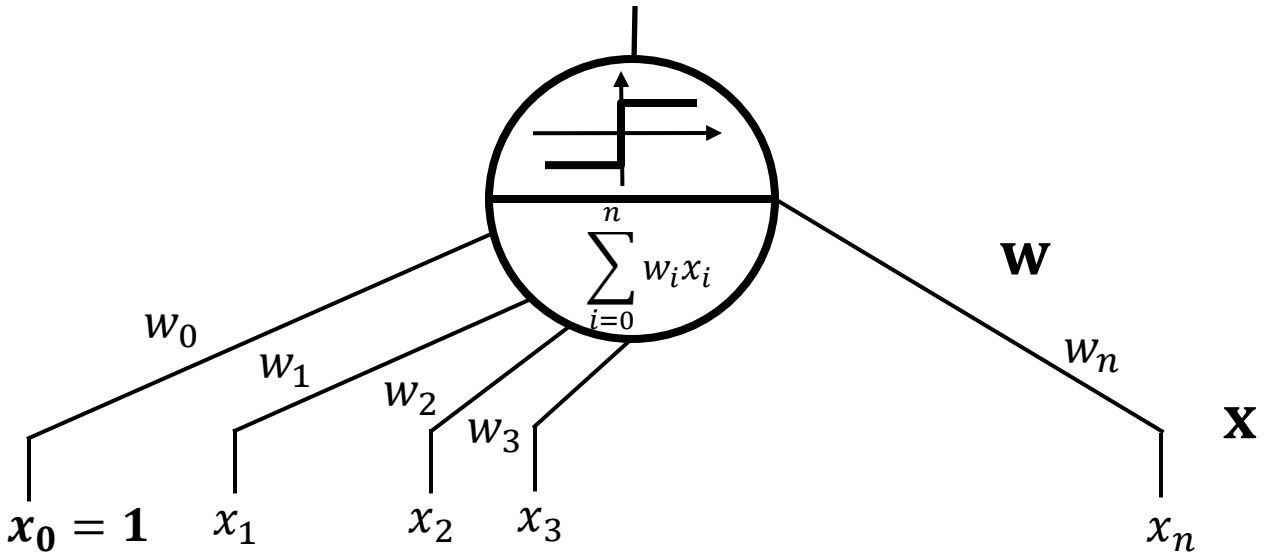


Horizontal Edge Detector

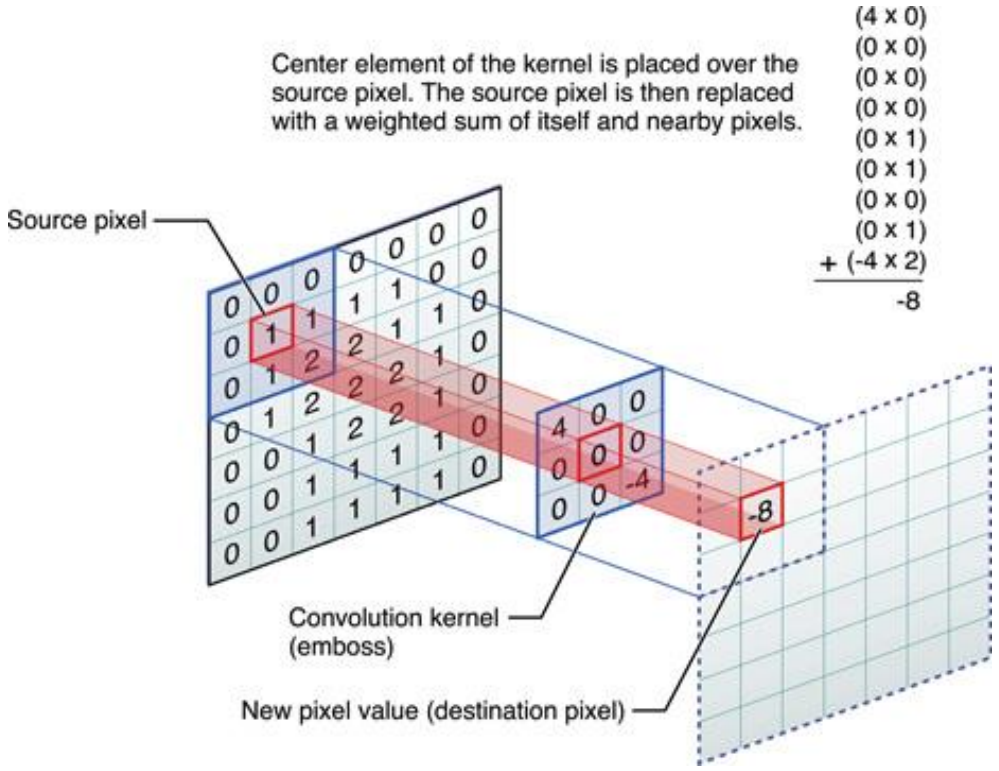
Deep representation by CNN

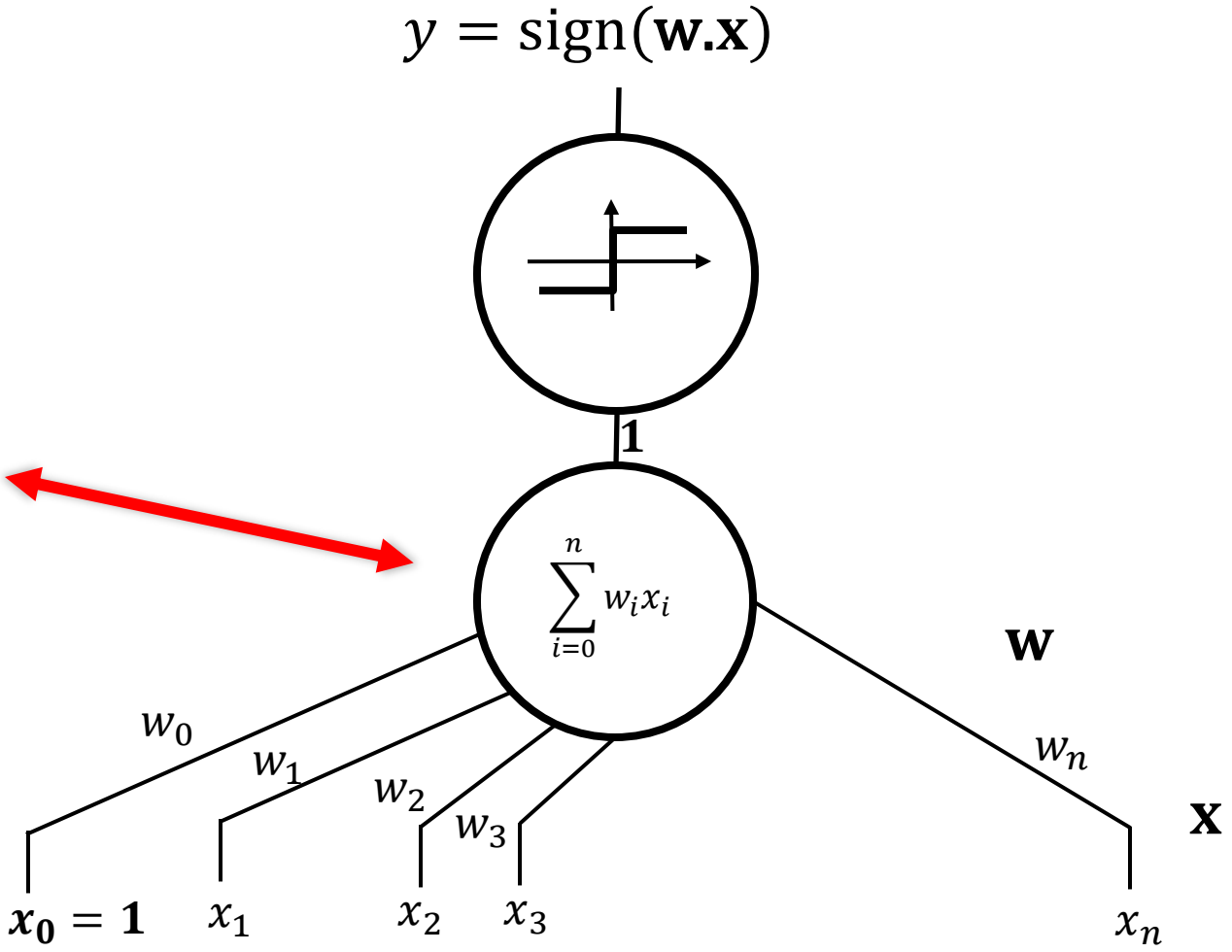


$$y = \text{sign}(\mathbf{w}.\mathbf{x})$$



Deep representation by CNN





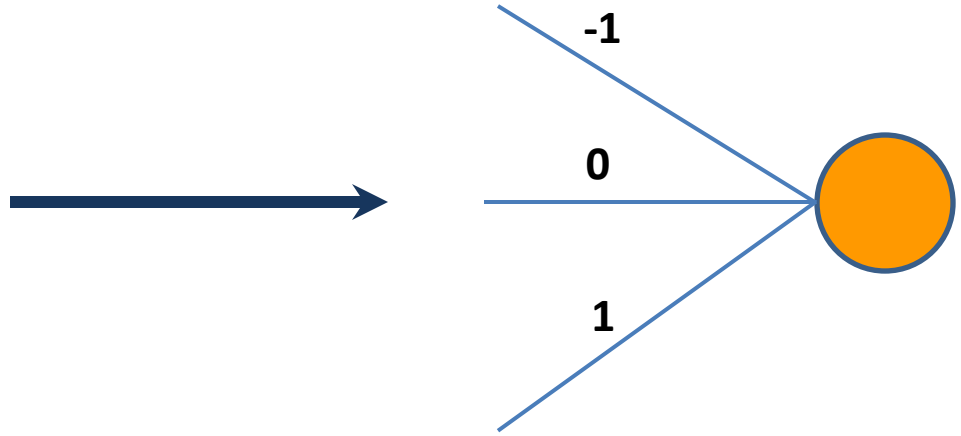
UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

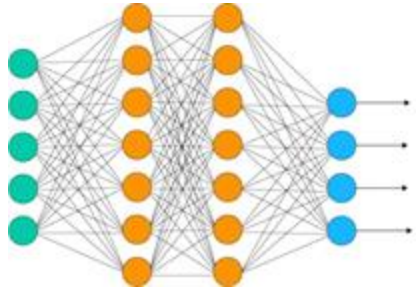
Filter in CNN

-1
0
1

Filter



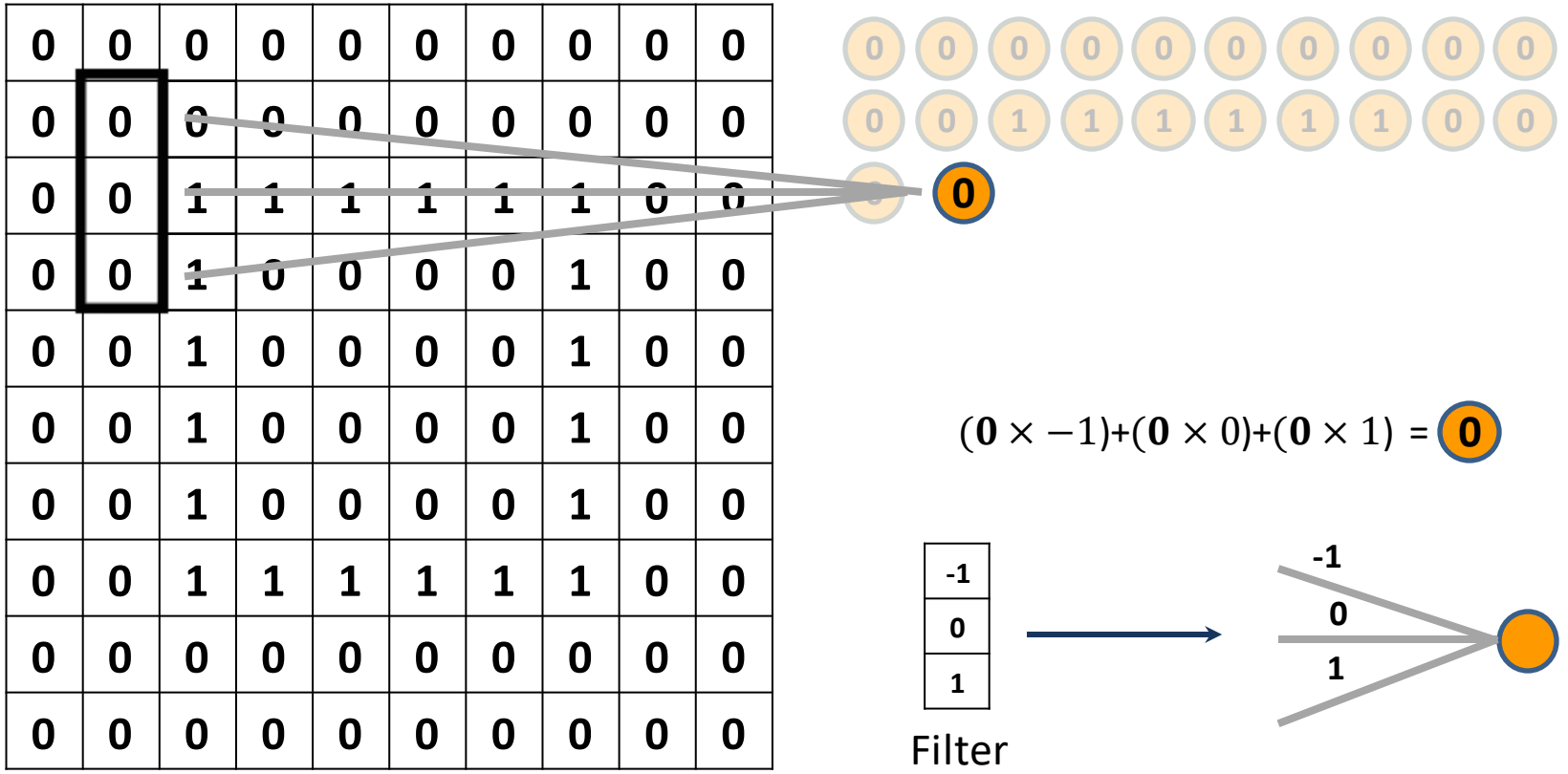
In CNN, the filter becomes a neuron



UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

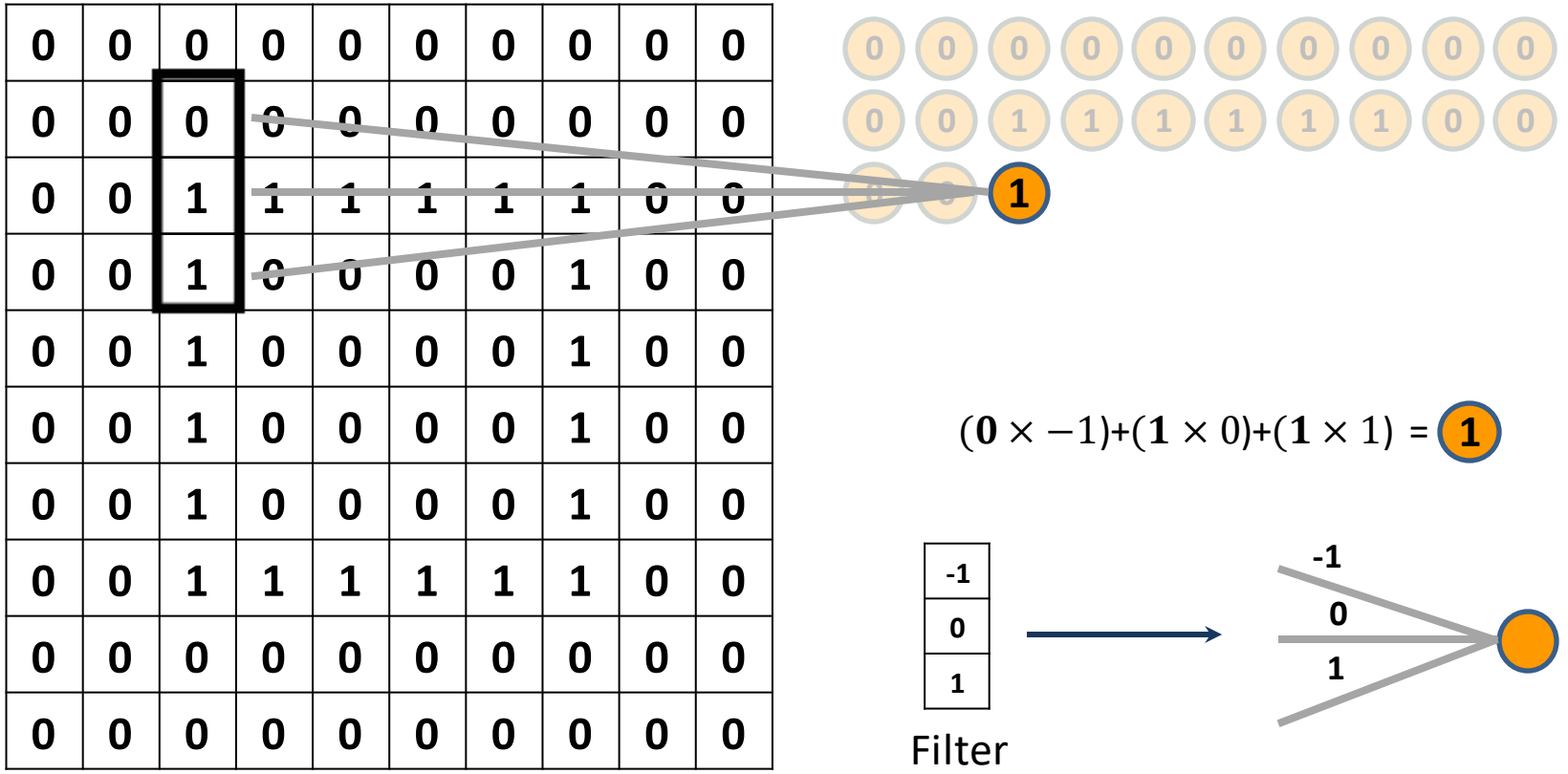
Filter in CNN



UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

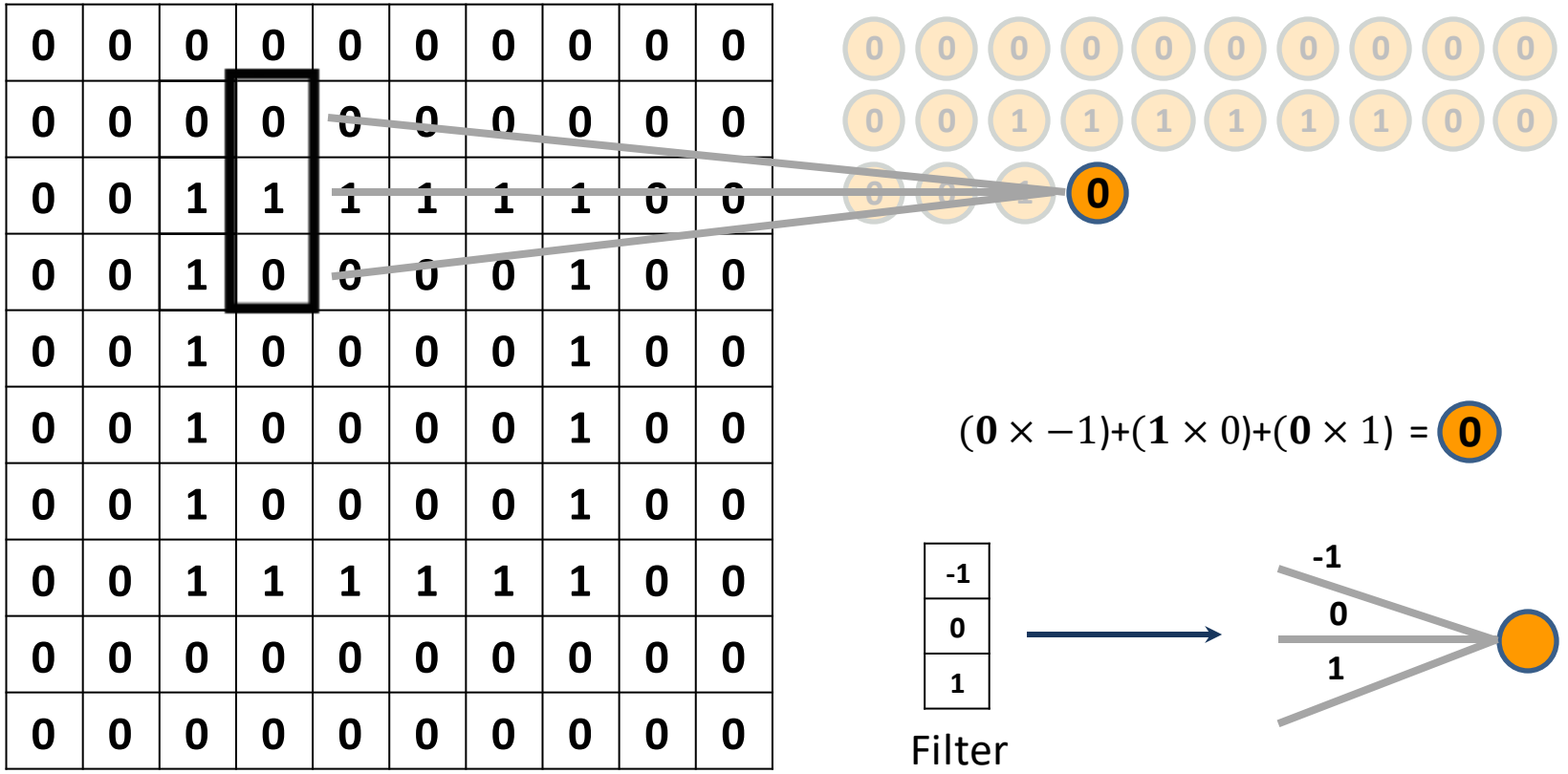
Filter in CNN



UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

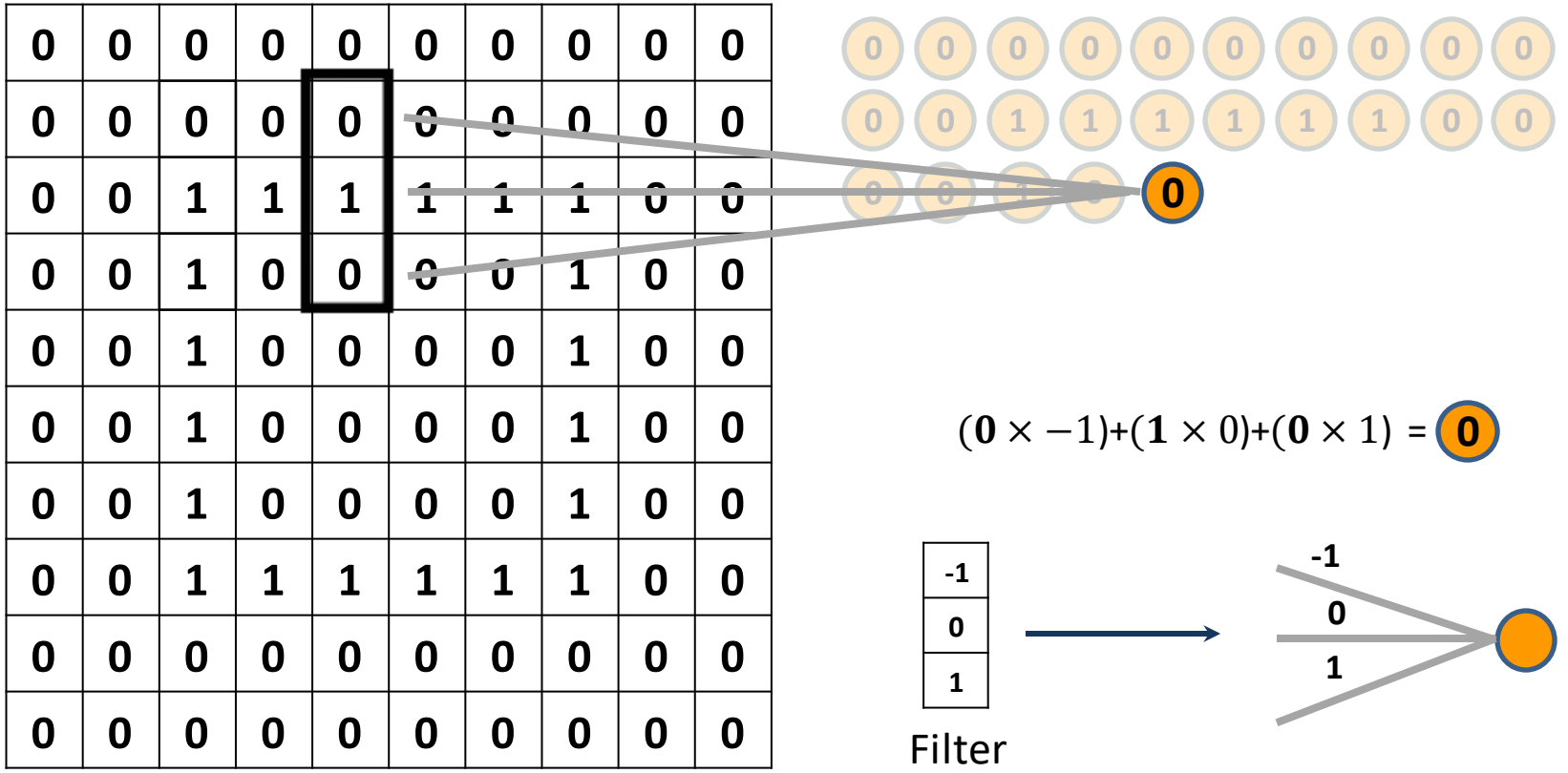
Filter in CNN



UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

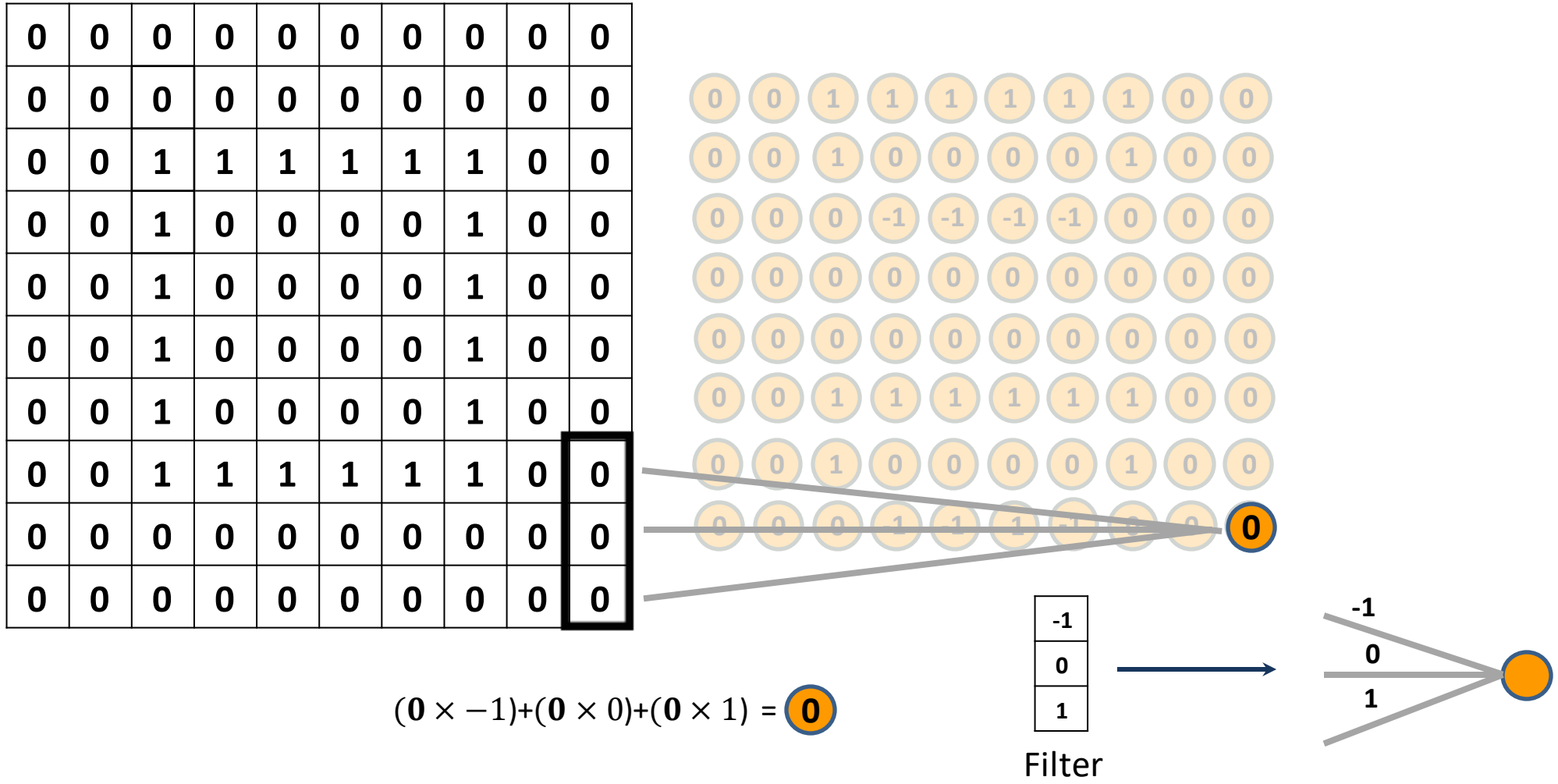
Filter in CNN



UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

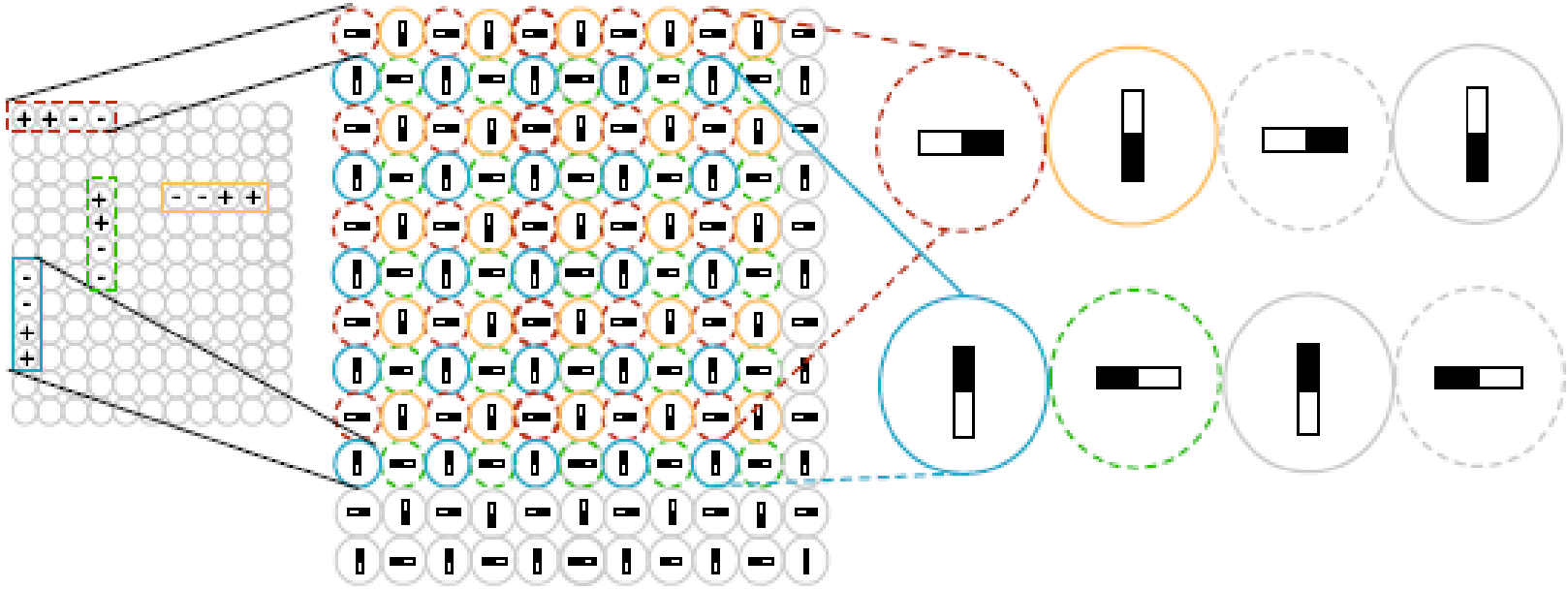
Filter in CNN



UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

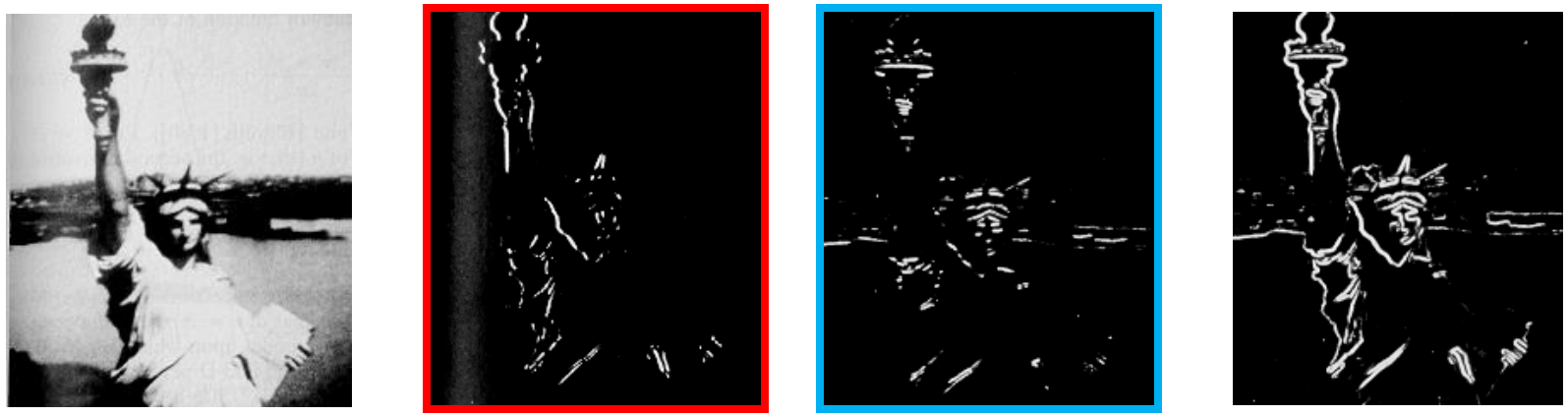
Convolution in nature



Octopus Retina Neurons

Simple Cells (IF Chip 0)

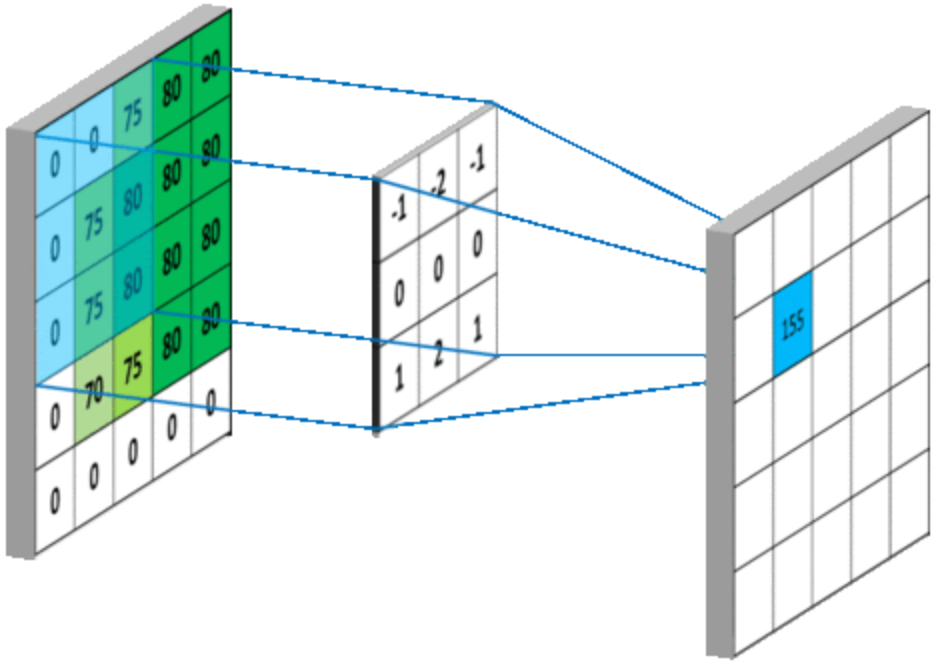
Complex Cells (IF Chip 1)



UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

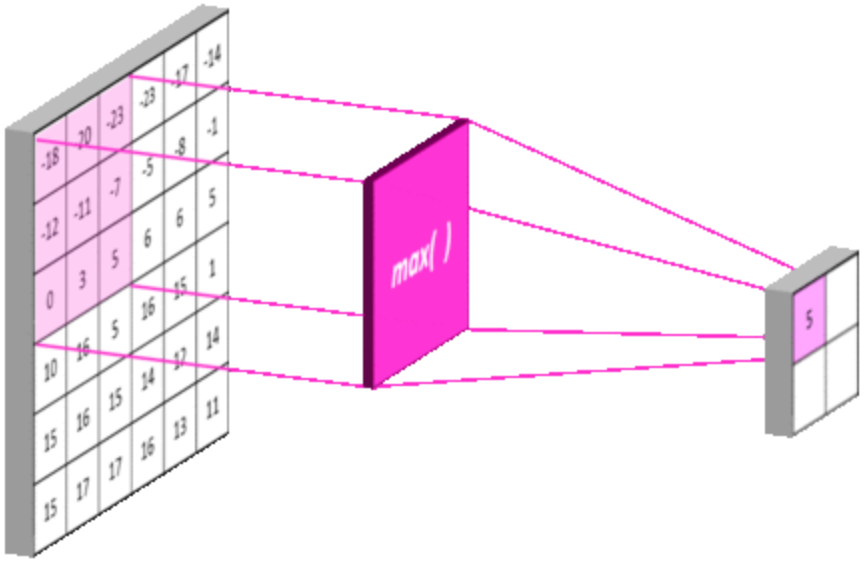
Image Convolution



UNIVERSITÉ CÔTE D'AZUR

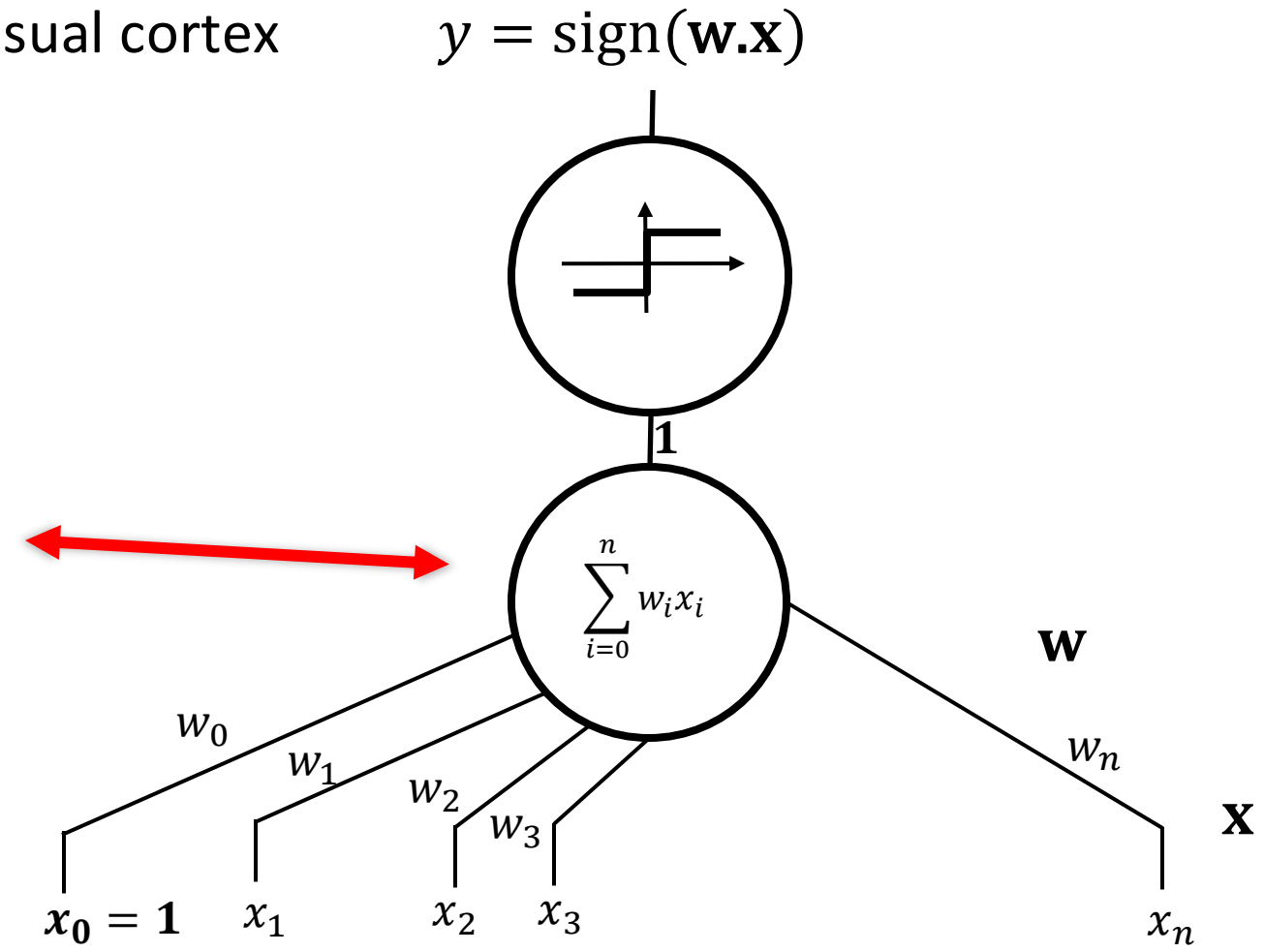
univ-cotedazur.fr

Max pooling



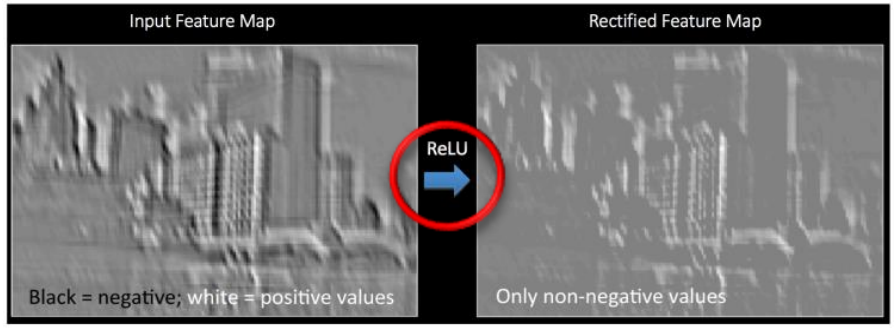
Convolution in nature

1. Hubel and Wiesel have worked on visual cortex of cats (1962)
2. Convolution

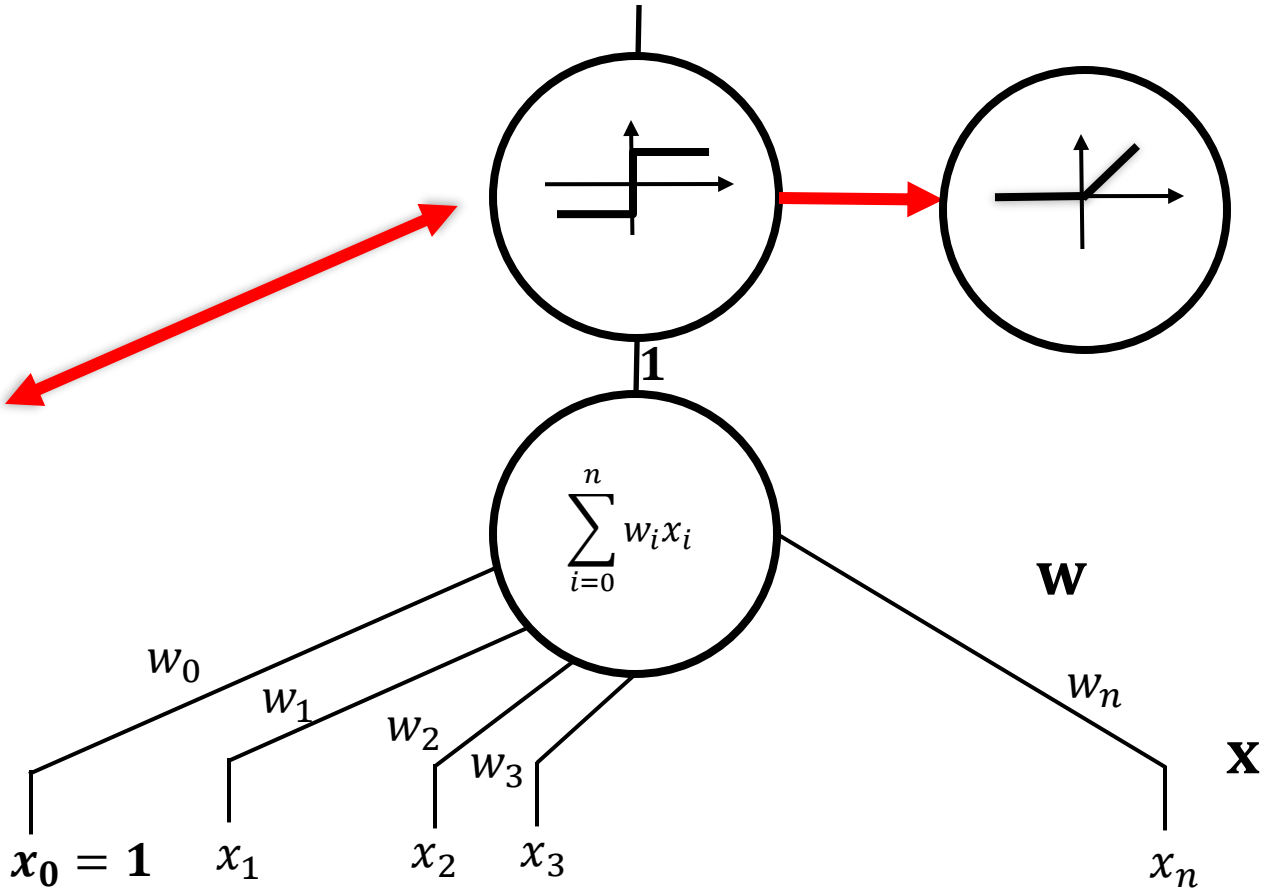


Convolution in nature

1. Hubel and Wiesel have worked on visual cortex of cats (1962)
2. Convolution
3. Activation

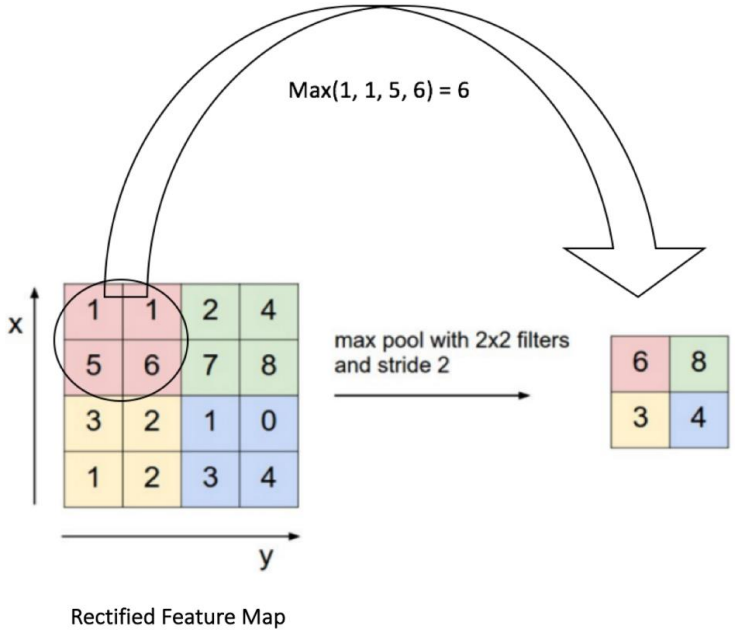


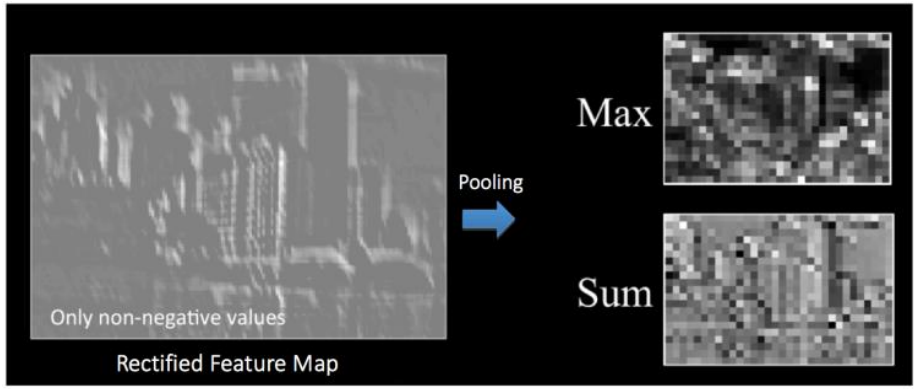
$y = \text{sign}(\mathbf{w}.\mathbf{x})$



Convolution in nature

1. Hubel and Wiesel have worked on visual cortex of cats (1962)
2. Convolution
3. Activation
4. Pooling

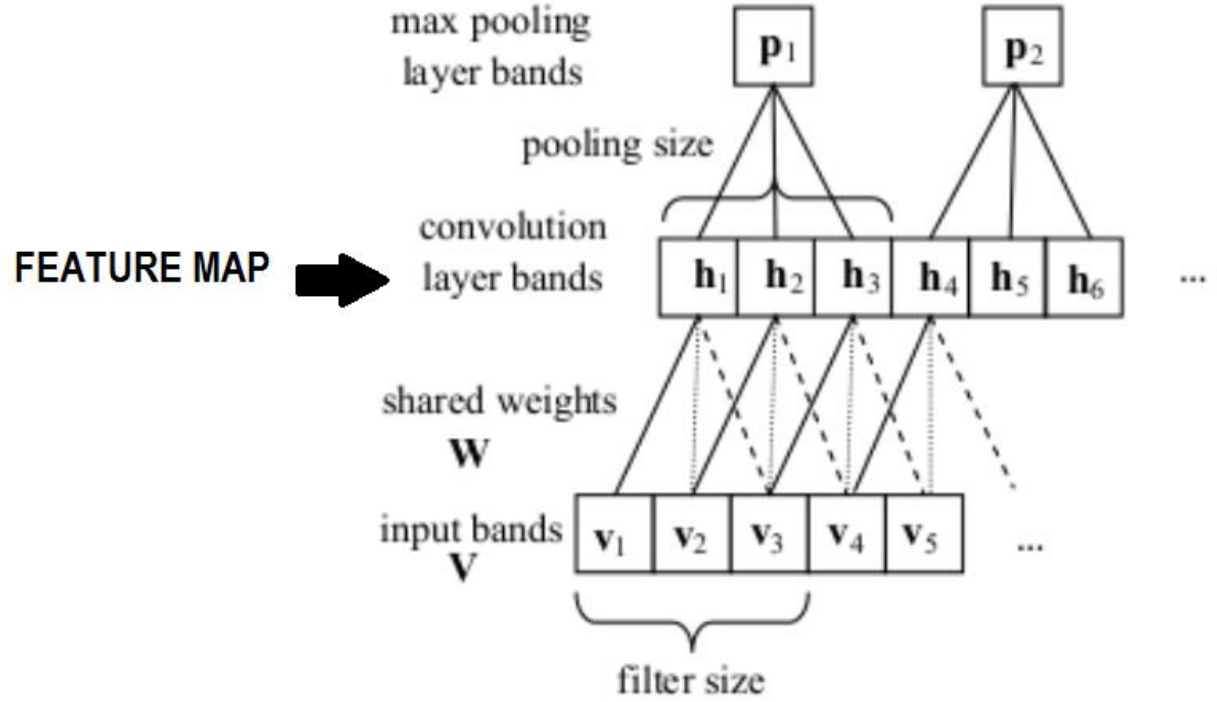




Deep representation by CNN

Yann Lecun, [LeCun et al., 1998]

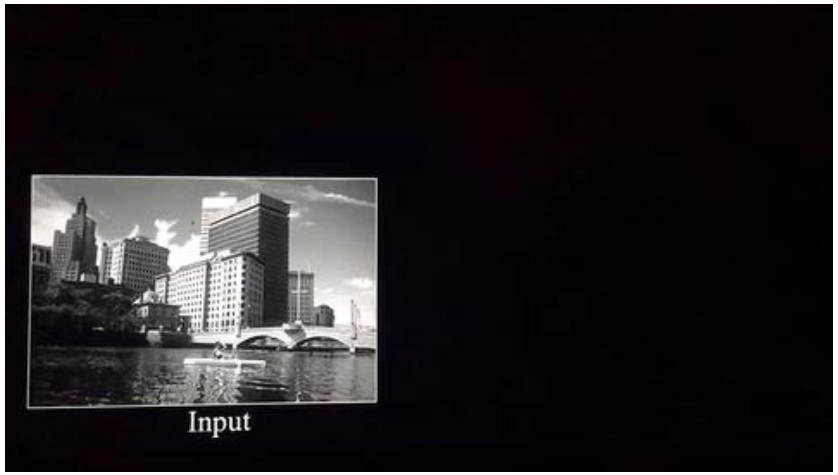
1. Subpart of the field of vision and translation invariant
2. S cells: convolution with filters
3. C cells: max pooling

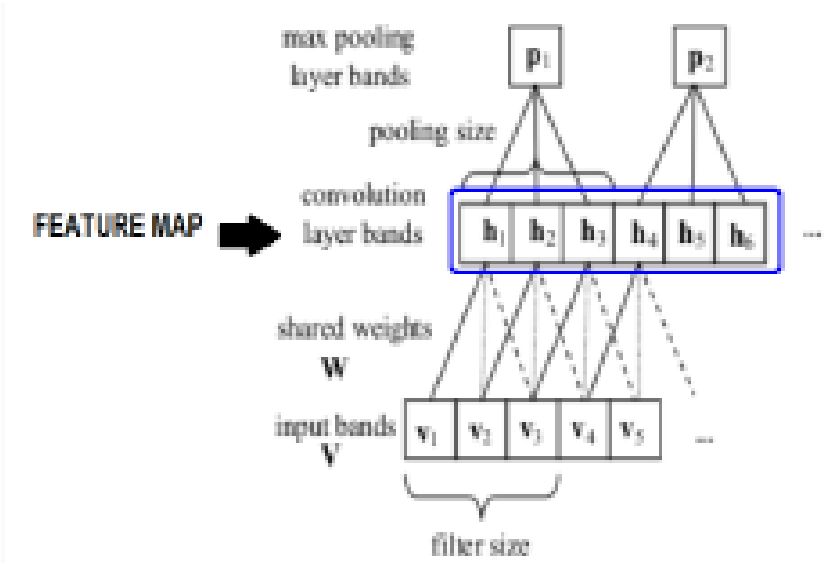


Deep representation by CNN

Yann Lecun, [LeCun et al., 1998]

1. Subpart of the field of vision and translation invariant
2. S cells: convolution with filters
3. C cells: max pooling

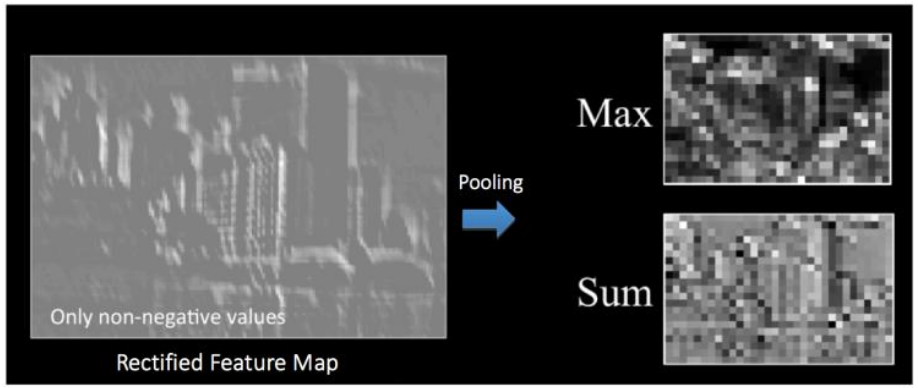


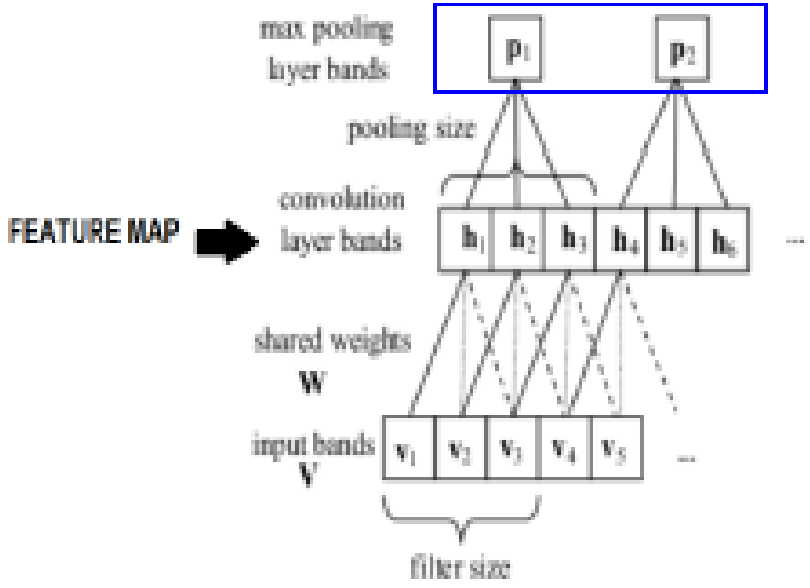


Deep representation by CNN

Yann Lecun, [LeCun et al., 1998]

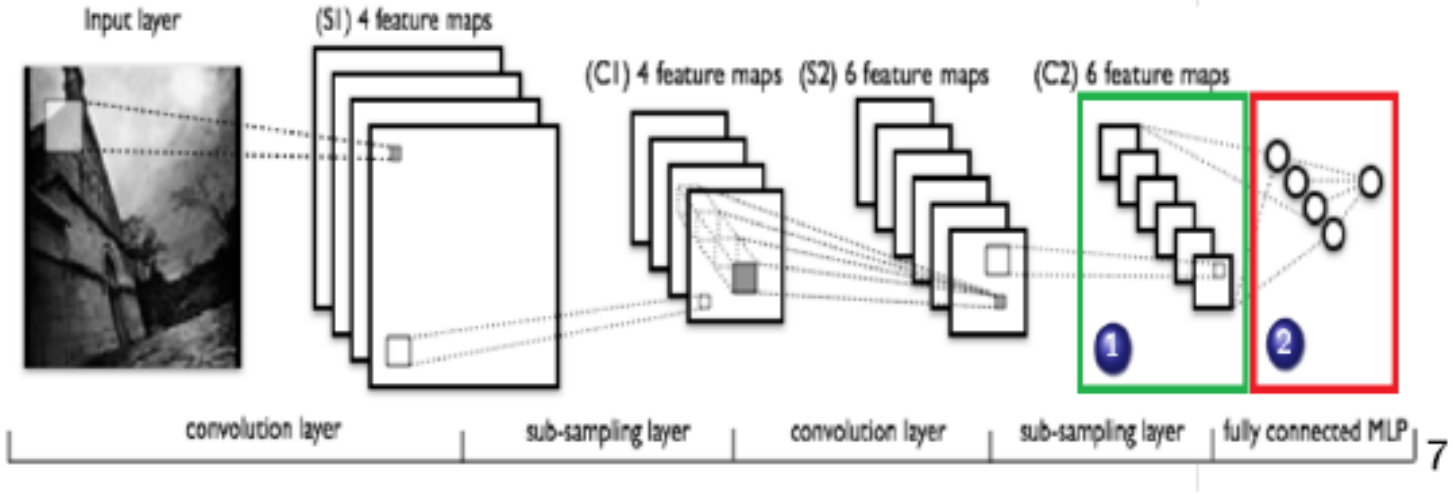
1. Subpart of the field of vision and translation invariant
2. S cells: convolution with filters
3. C cells: max pooling





Deep representation by CNN

- feature map = result of the convolution
- convolution with a filter extract characteristics (*edge detectors*)
- extract parallelised characteristics at each layer

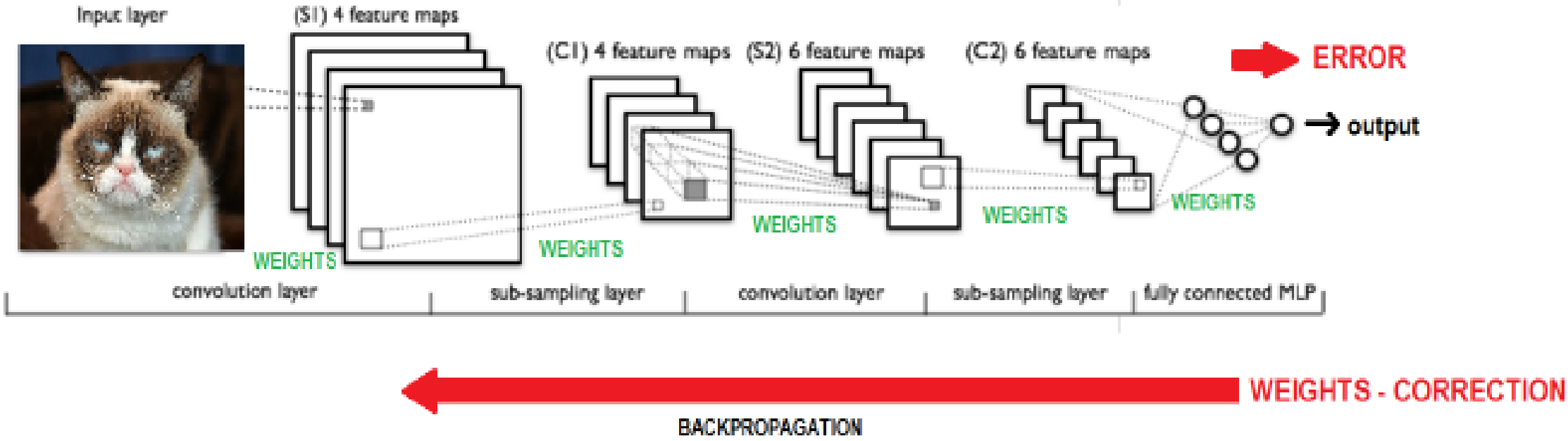


1. final representation of our data
2. classifier (MLP)

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

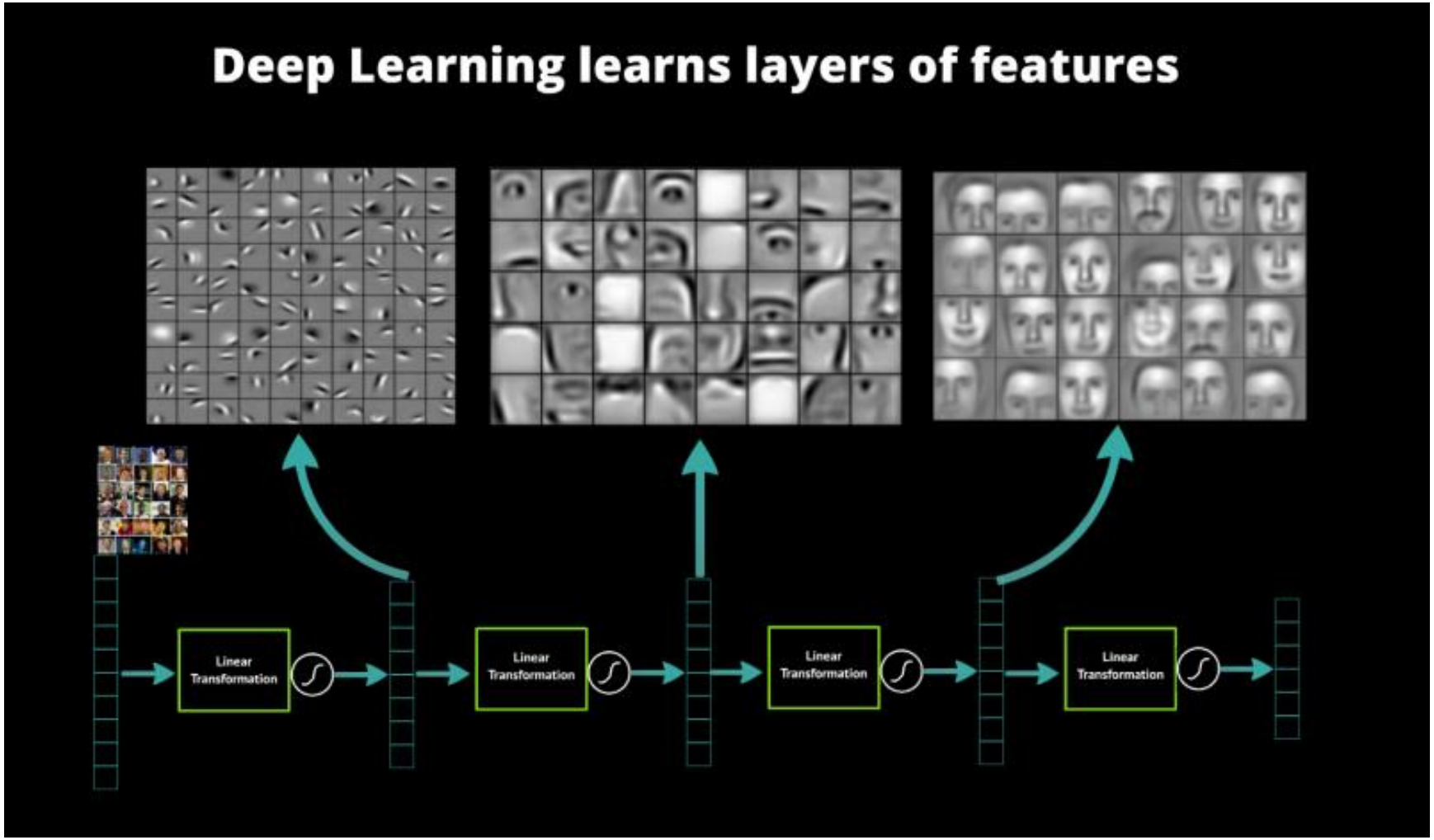
Deep representation by CNN



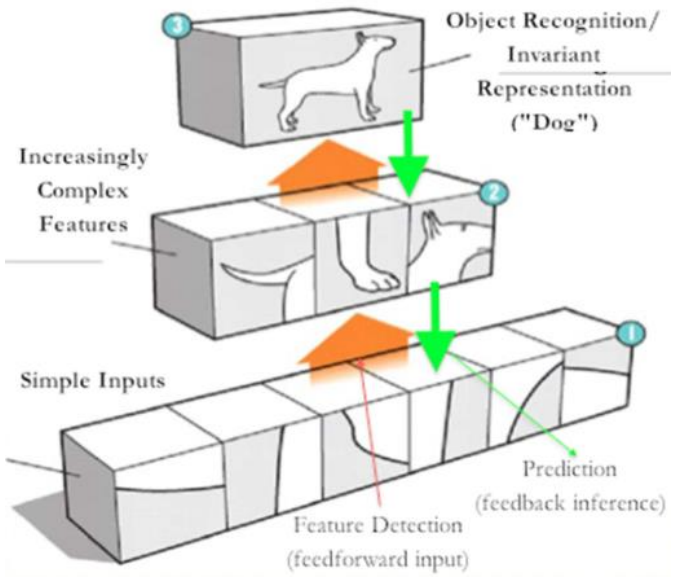
UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

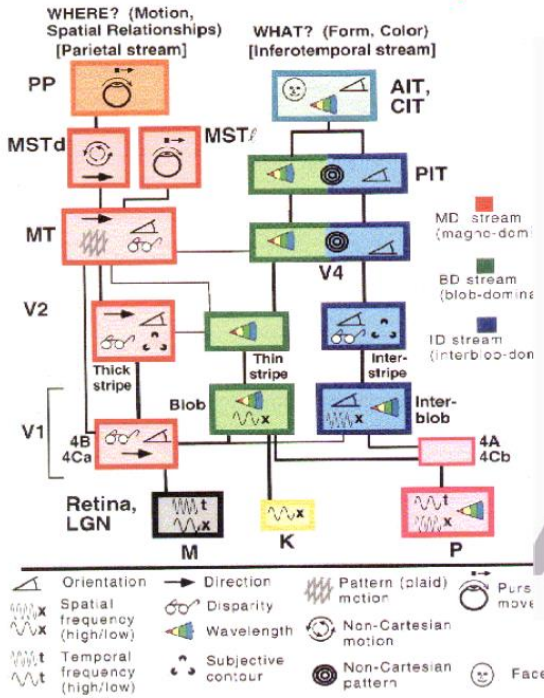
Deep representation by CNN

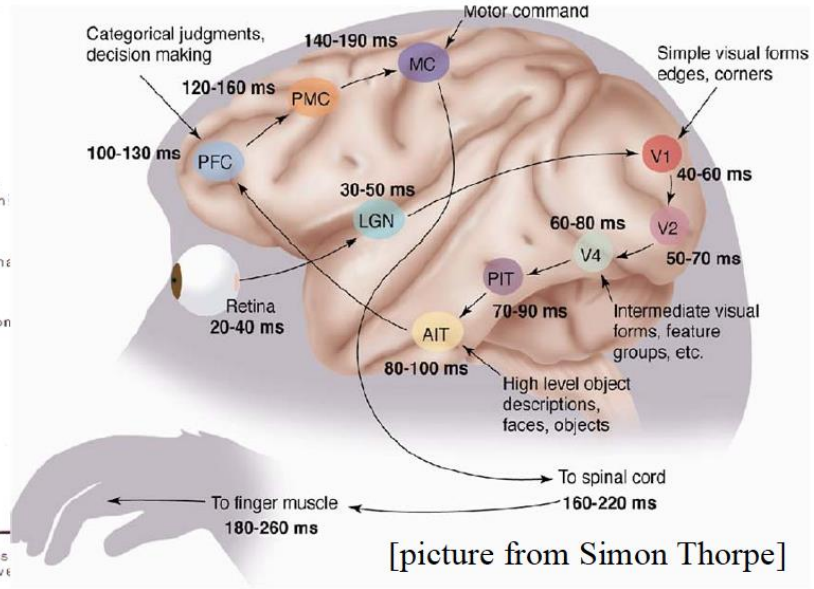


The Mammalian Visual Cortex Inspires CNN



- The ventral (recognition) pathway in the visual cortex has multiple stages
- Retina - LGN - V1 - V2 - V4 - PIT - AIT ....
- Lots of intermediate representations



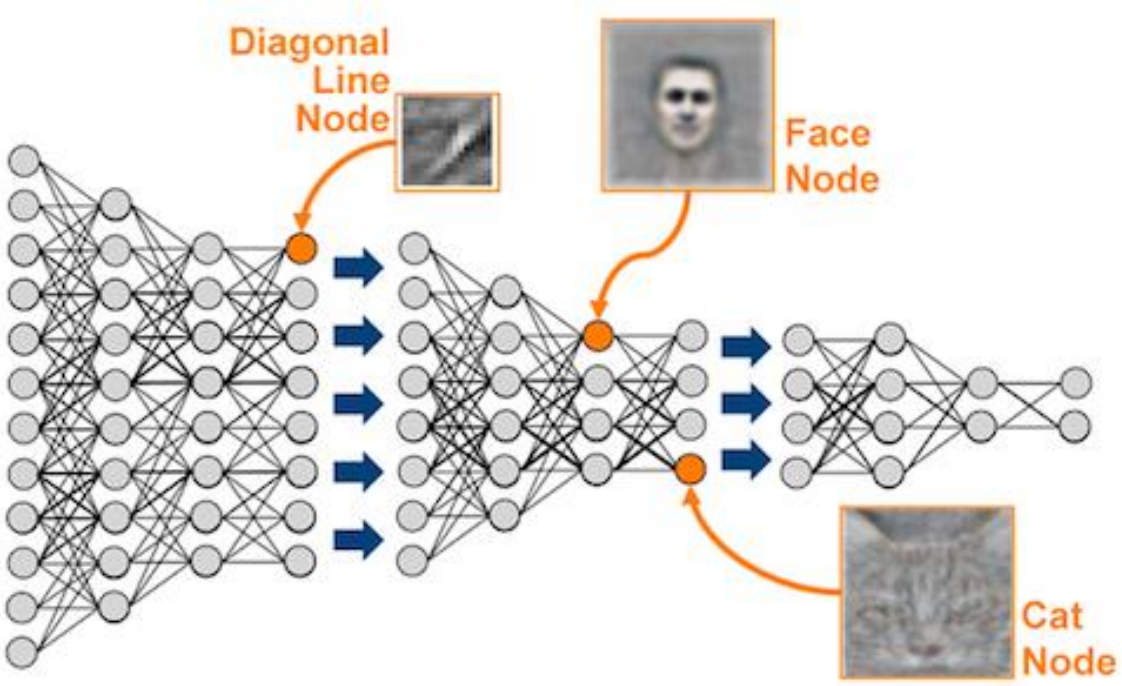


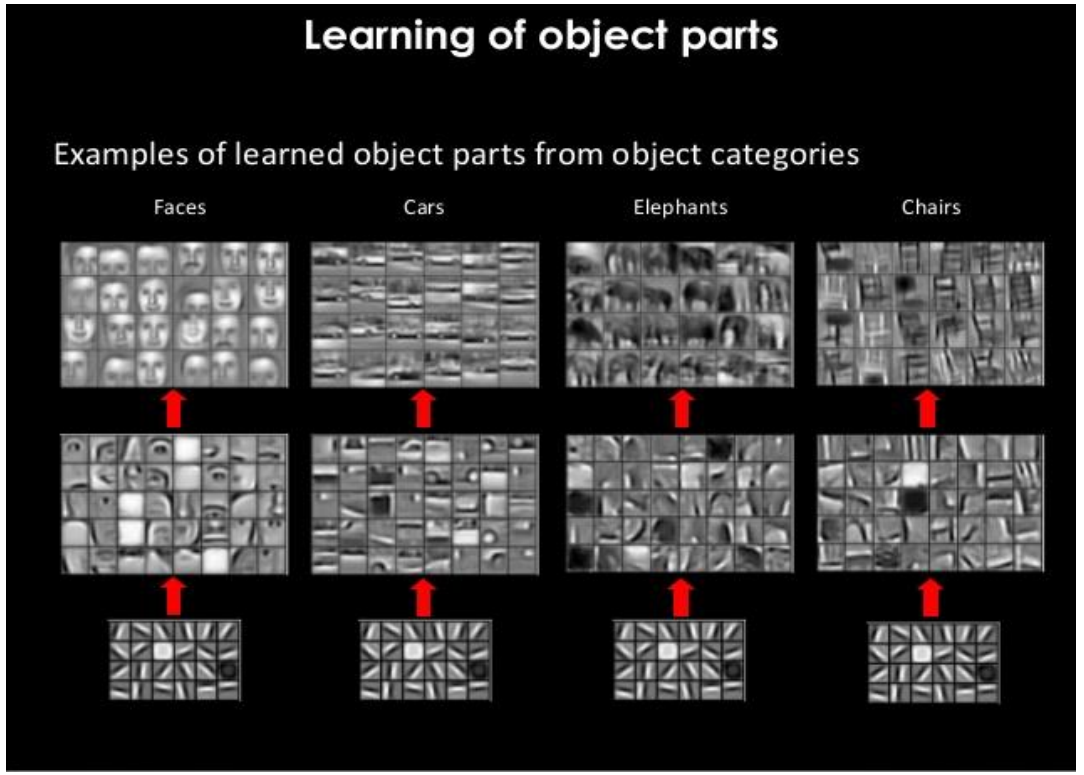
[Gallant & Van Essen]

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Deep representation by CNN

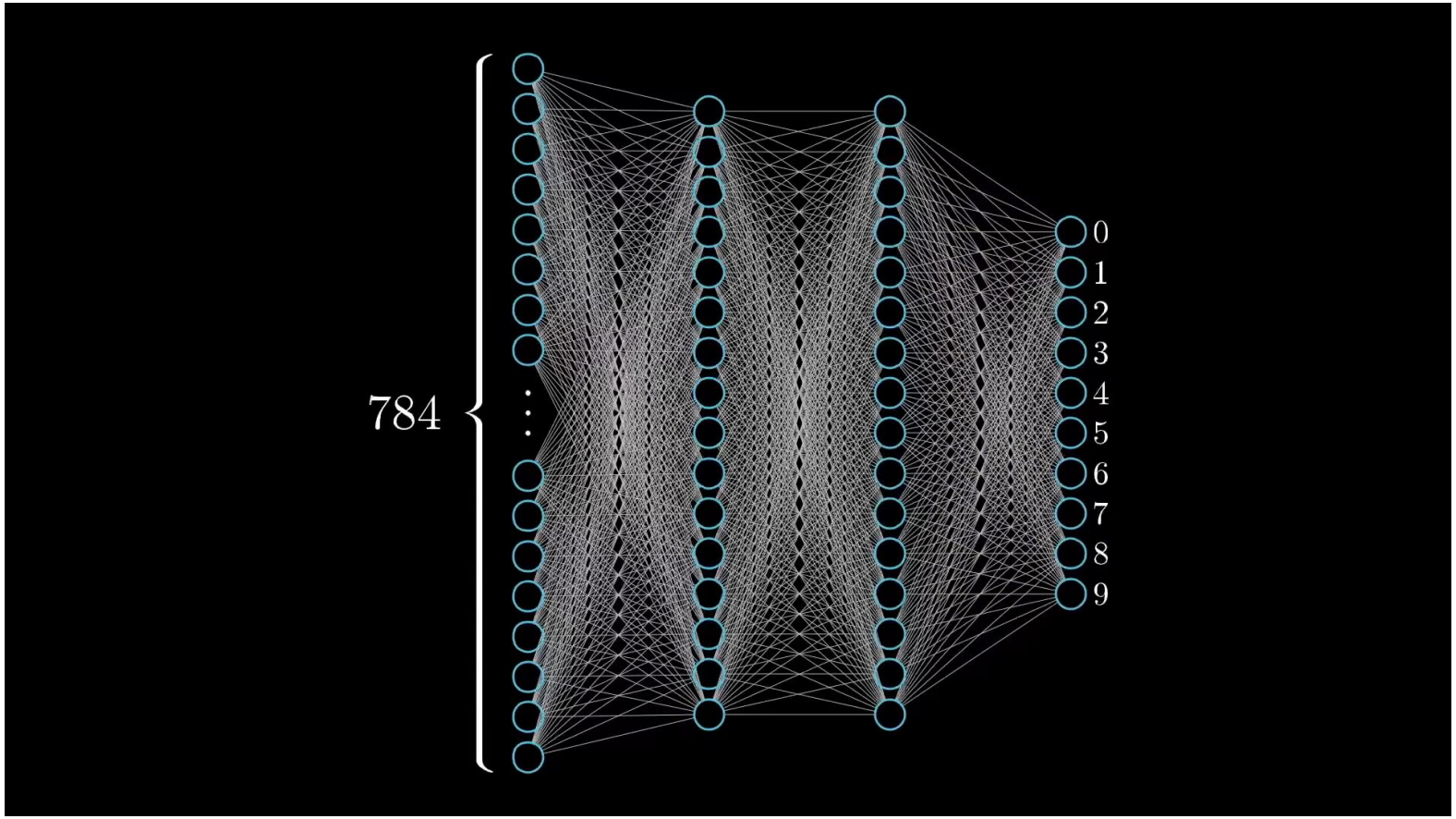




UNIVERSITÉ CÔTE D'AZUR

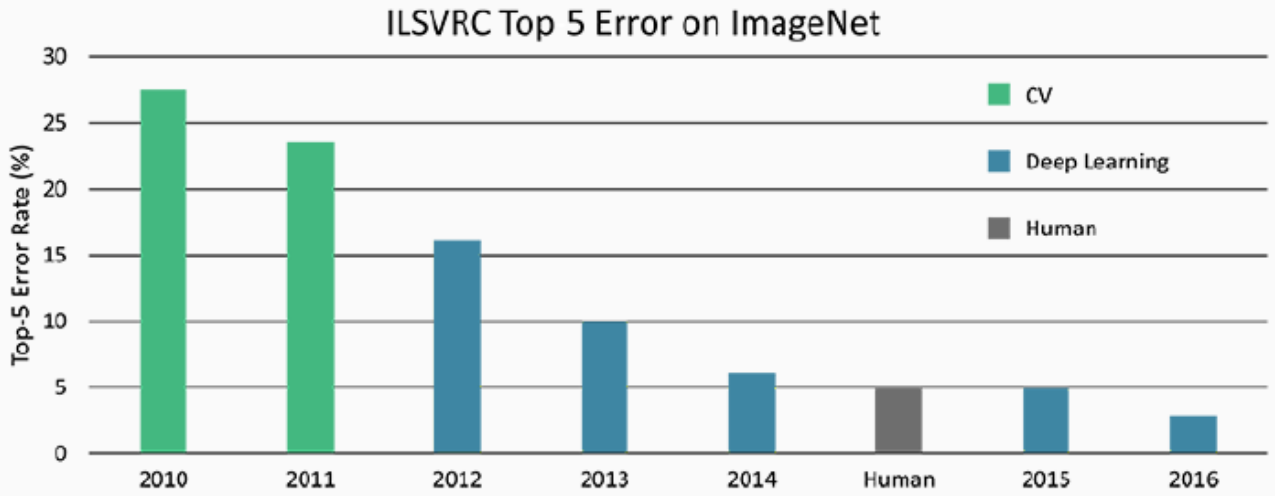
univ-cotedazur.fr

Deep representation by CNN



Deep representation by CNN

- Deep Networks are as good as humans at recognition, identification...



How much does a deep network understands those tasks?

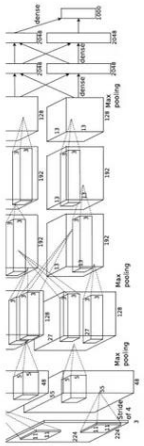
UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Deep = Many hidden layers

http://cs231n.stanford.edu/slides/winter1516_lecture8.pdf

8 layers



16.4%

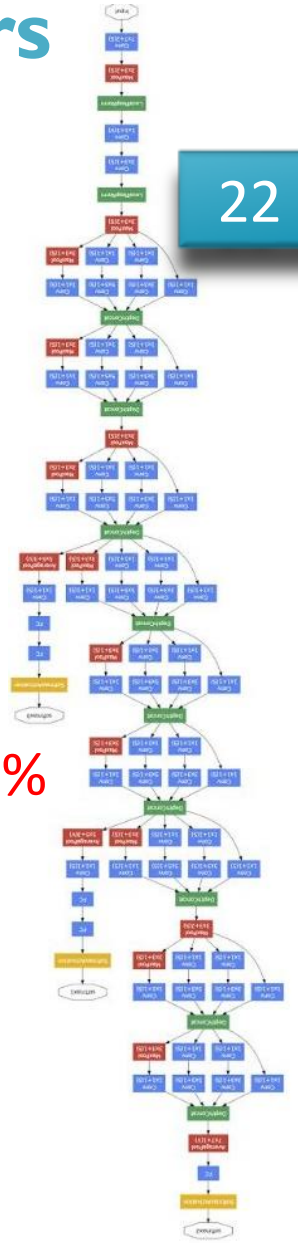
AlexNet (2012)

19 layers

7.3%

VGG (2014)

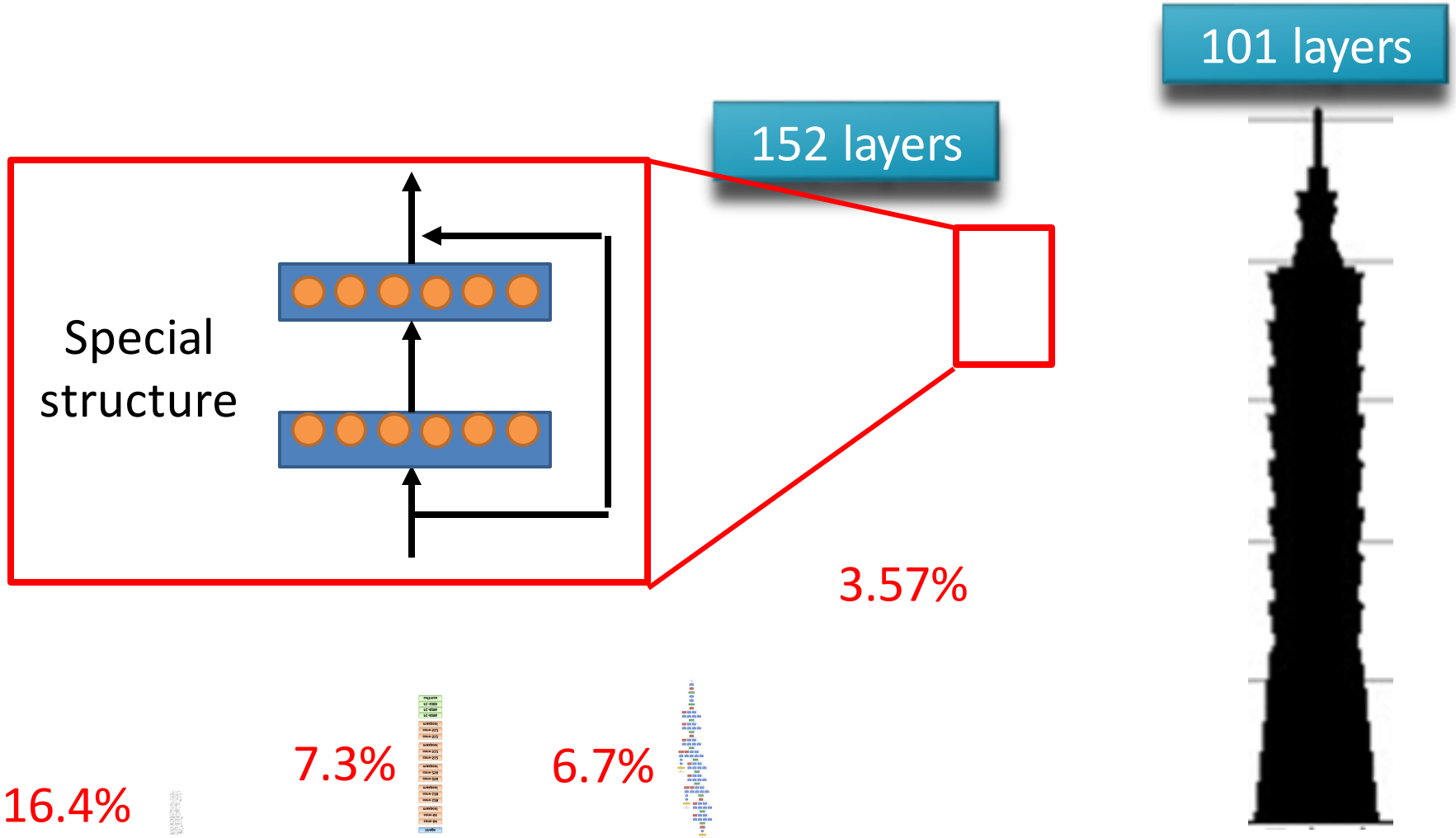
22 layers



6.7%

GoogleNet (2014)

Deep = Many hidden layers



Optimisation, Tips and Tricks

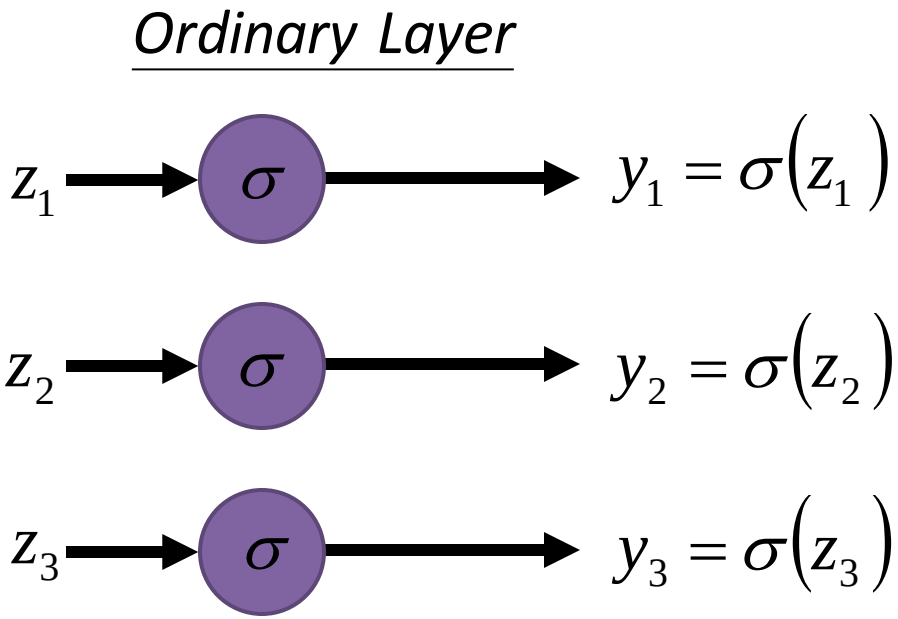
For all deep neural architectures

Basic recipes for neural networks

Which preprocessings?

- **Split data into train/val/test splits**. Validation set is used for tuning hyper-parameters, which is extremely important for training neural networks
- **Normalize the features in your data to have zero mean and unit variance.** It can make the features in the same scale.
- If your data is very high-dimensional, consider using a dimensionality reduction technique (such as PCA).
- If the size of your dataset is small, you can also do data augmentation (e.g., CV: horizontally flipping, random crops and color jittering; NLP: synonym substitution).

Which output layer? Softmax layer!



In general, the output of network can be any value.

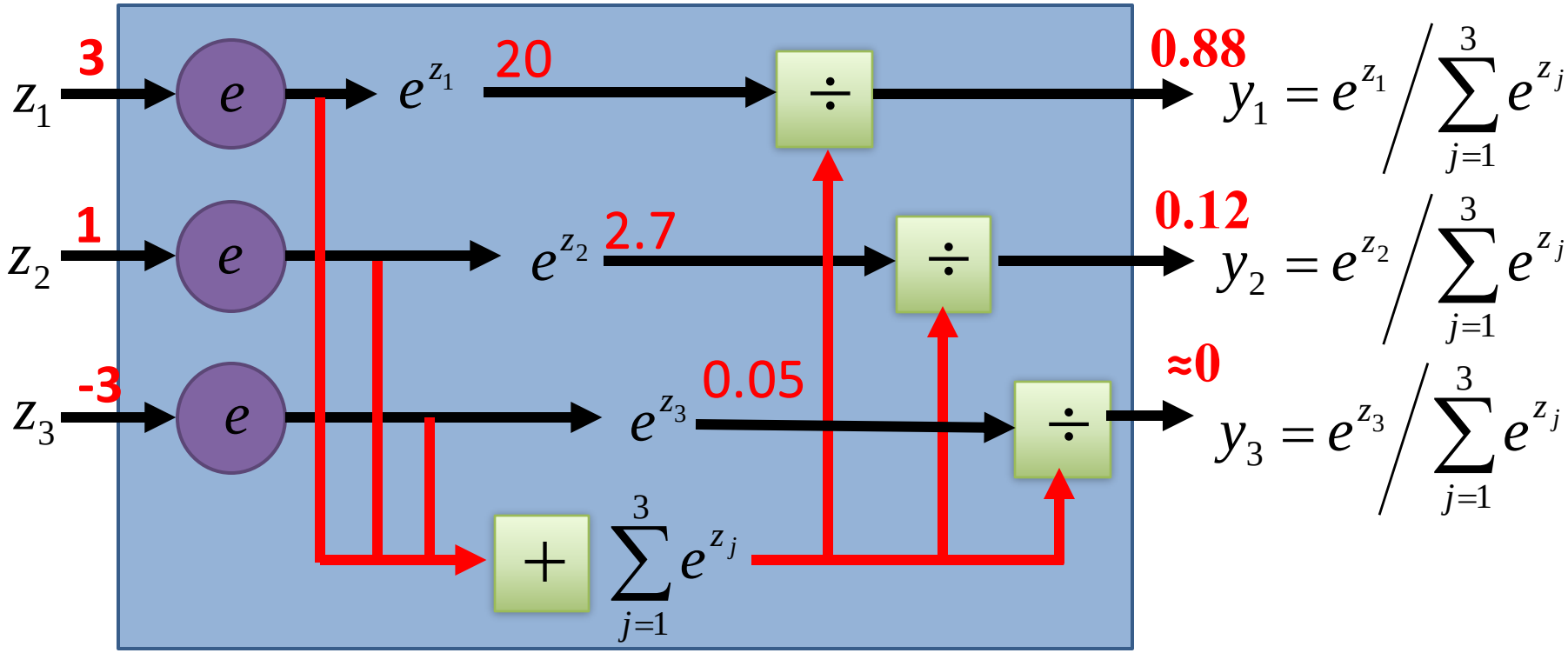
May not be easy to interpret

UNIVERSITÉ CÔTE D'AZUR
univ-cotedazur.fr

Which output layer? Softmax layer!

- **Softmax layer as the output layer**
  (Monotonicity and Non-locality of Softmax)

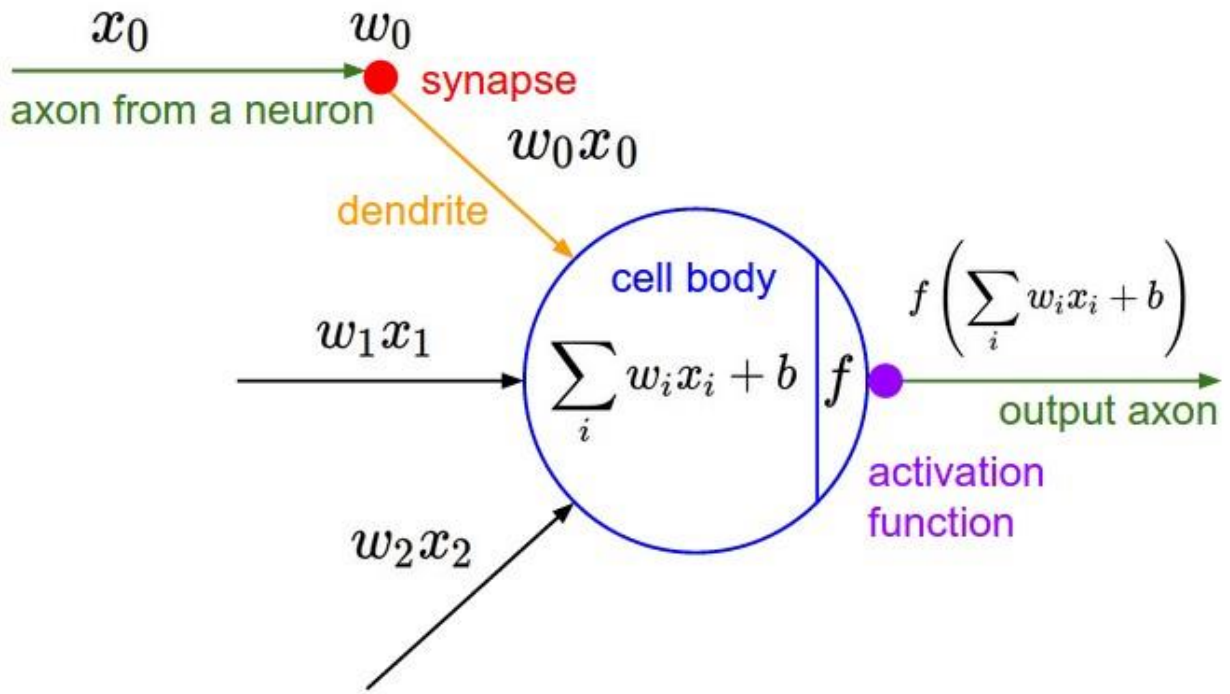
Probability:
- $1 > y_i > 0$
- $\sum_i y_i = 1$



UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

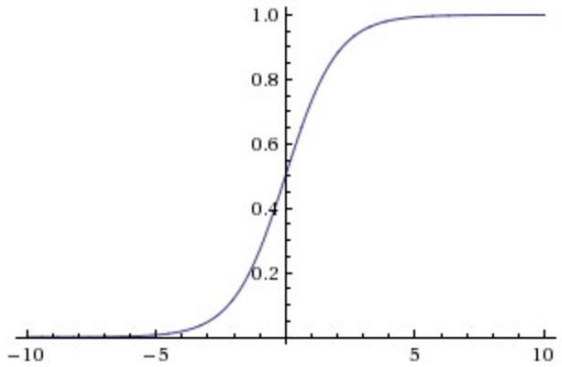
Which activation function?



Each neuron performs a dot product with the input and its weights, adds the bias and applies the non-linearity (or activation function)

Which activation function?

- ***Sigmoid***



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

Pros:

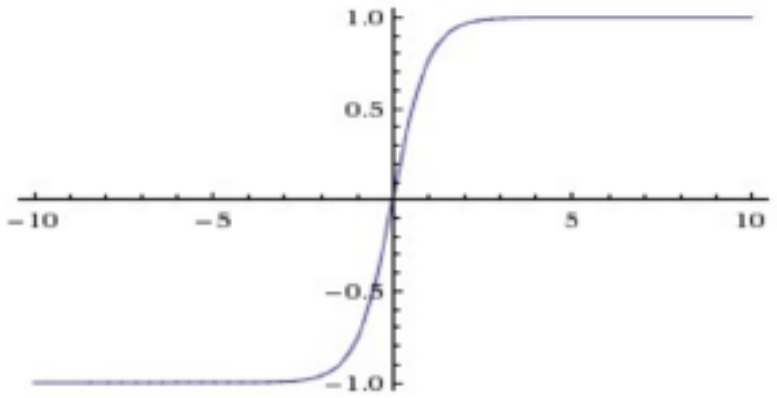
- The output is between 0 and 1, can be used as the output layer.
- Easy to compute derivation: $\sigma(x)(1-\sigma(x))$

Cons:

- Saturating when receiving strong signals, have derivatives of 0 at both ends, drive other gradients in previous layers towards 0.
- Exploding gradient problems.
  Trick: clipping the gradients (if the gradient is exceeding a threshold, then pushed down to that threshold)

Which activation function?

- ***tanh***



$$\tanh(x) = \frac{1-e^{-2x}}{1+e^{-2x}} = 2\sigma(2x) - 1$$

Pros:

- The output is centered around 0.
- Easy to compute derivative $(tanh(x))' = 1 - tanh^2(x)$
- Converges faster than Sigmoid

Cons:

- Saturating, vanishing or exploding gradient problems

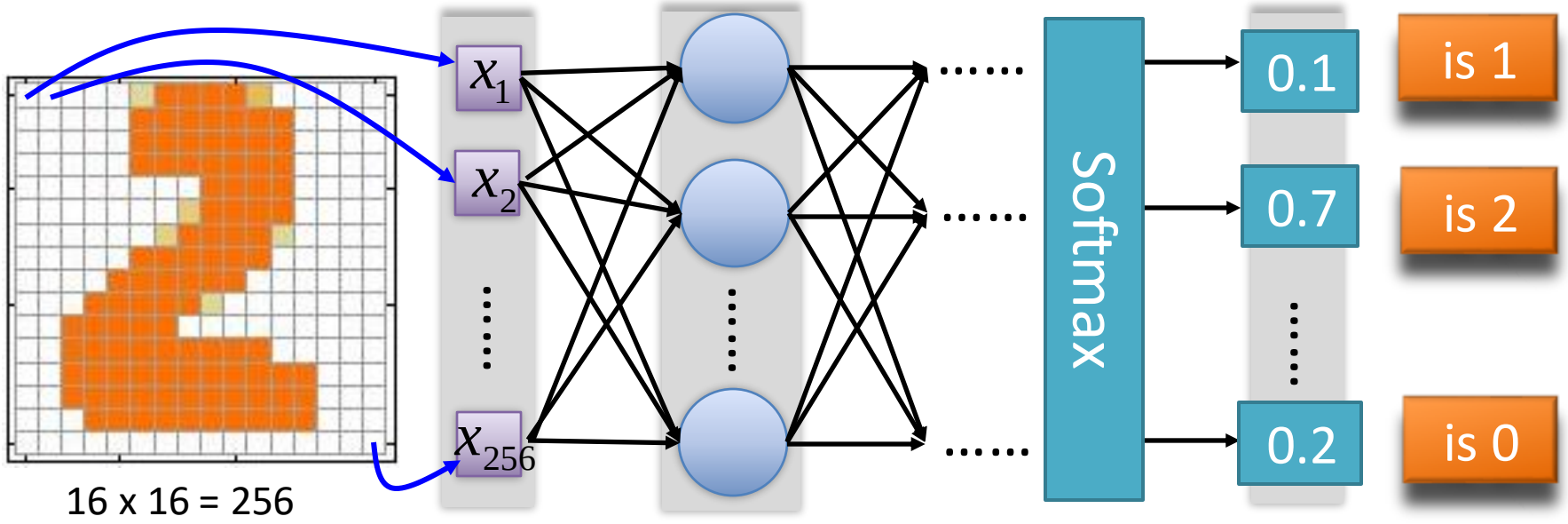
Training network parameters

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

How to set network parameters?

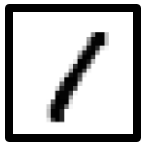
$$\boldsymbol{W} = \{W^1, b^1, W^2, b^2, \cdots W^L, b^L\}$$



Ink → 1

No ink → 0

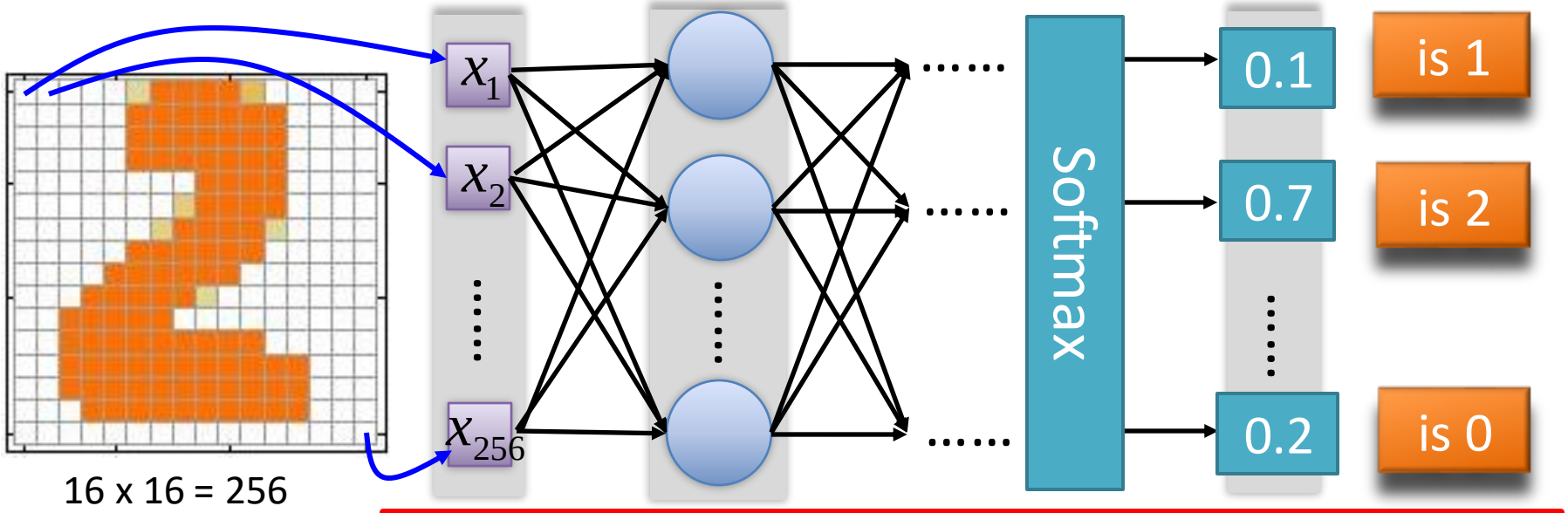
Set the network parameters $\boldsymbol{W}$ such that ……

Input:  → y_1 has the maximum value

Input:  → y_2 has the maximum value

How to set network parameters?

$$\boldsymbol{W} = \{W^1, b^1, W^2, b^2, \cdots W^L, b^L\}$$



Ink → 1

No ink → 0

Set the network parameters $\boldsymbol{W}$ such that ……

Input … value

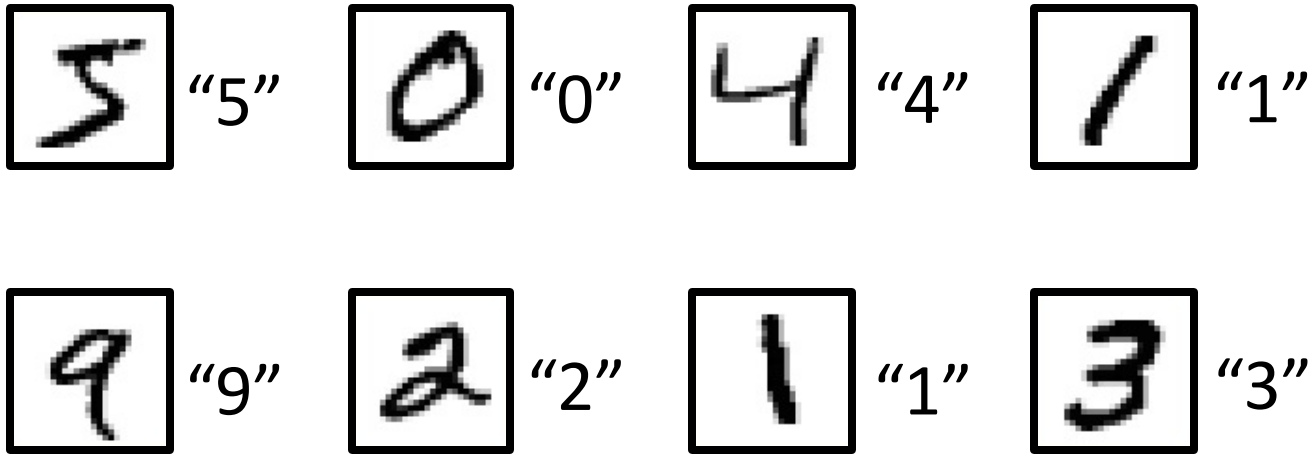
Input … value

How to let the neural network achieve this

UNIVERSITÉ CÔTE D'AZUR
univ-cotedazur.fr

How to train the network?

- Preparing training data: images and their labels

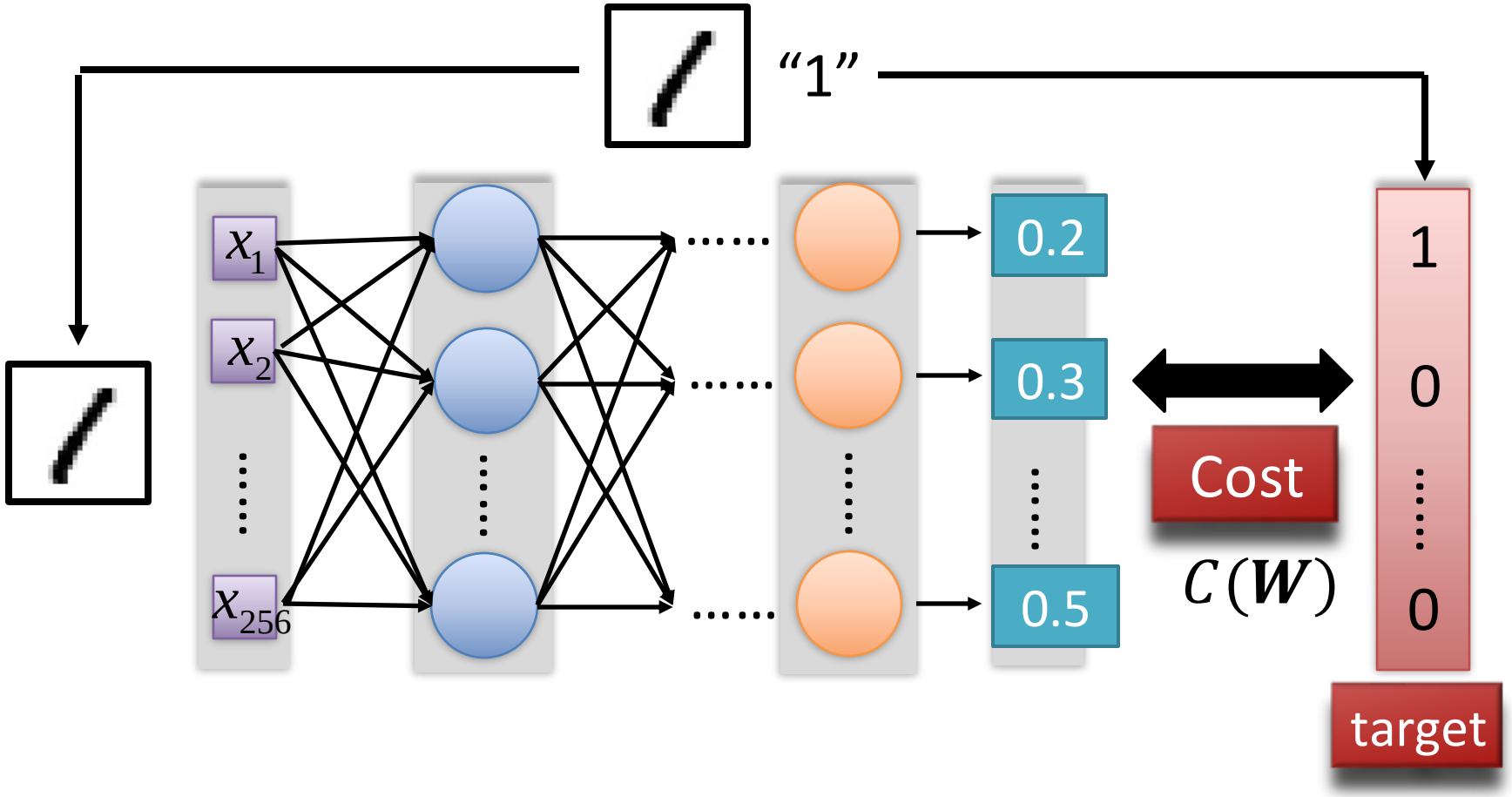


Using the training data to find the network parameters.

UNIVERSITÉ CÔTE D'AZUR
univ-cotedazur.fr

How to train the network?

Given a set of network parameters $\boldsymbol{W}$, each example has a cost value.



UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

How to define the loss L?

You should choose the right loss function based on your problem and your data
(here y is the true/expected answer, $f(x)$ the answer predicted by the network).

Classification

- Cross-entropy loss: $L(x) = -(y\,ln(f(x)) + (1-y)ln(1-f(x)))$
- Hinge Loss (max-margin loss, 0-1 loss): $L(x) = max(0, 1-yf(x))$
- …

Regression

- Mean square loss (or Quadratic Loss): $L(x) = (f(x)-y)^2$
- Mean absolute loss
- …

Retrieval

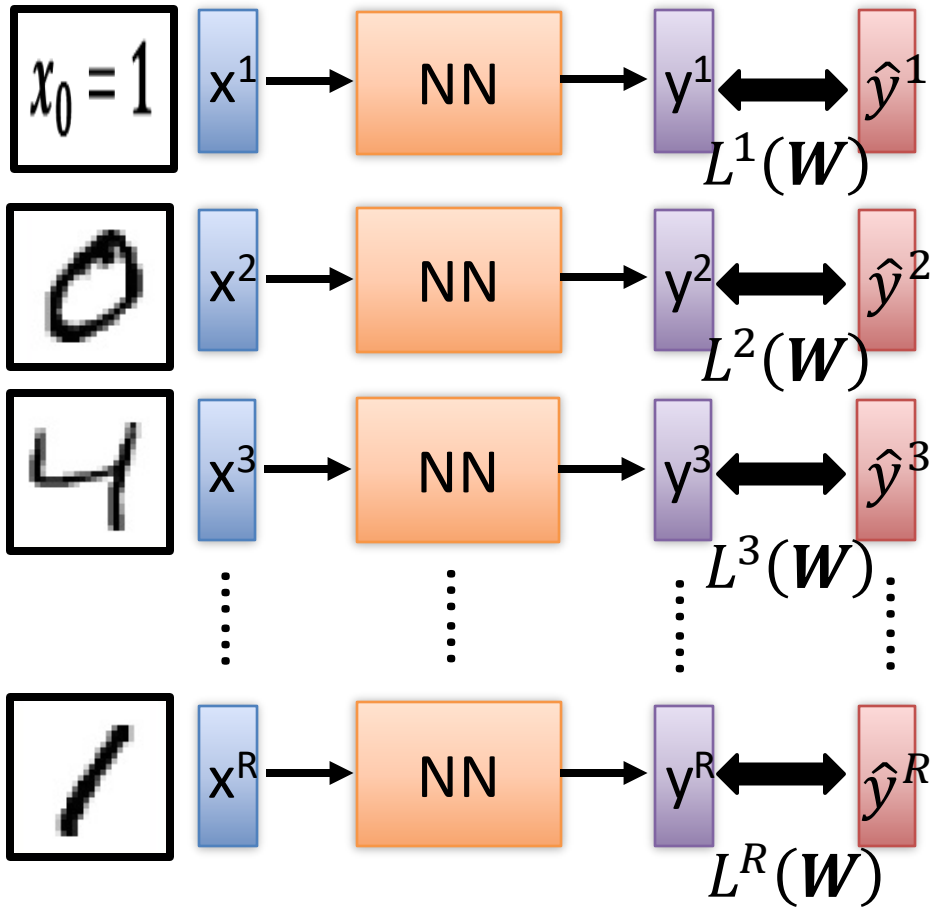
- Triplet loss
- Cosine similarity, χ^2,…
- …

If the loss is minimized but accuracy is low, you should check the loss function. Maybe it is not the appropriate one for your task.

UNIVERSITÉ CÔTE D'AZUR
univ-cotedazur.fr

Total Cost?

For all training data …



Total Cost:

$$C(\boldsymbol{W}) = \sum_{r=1}^{R} L^r(\boldsymbol{W})$$

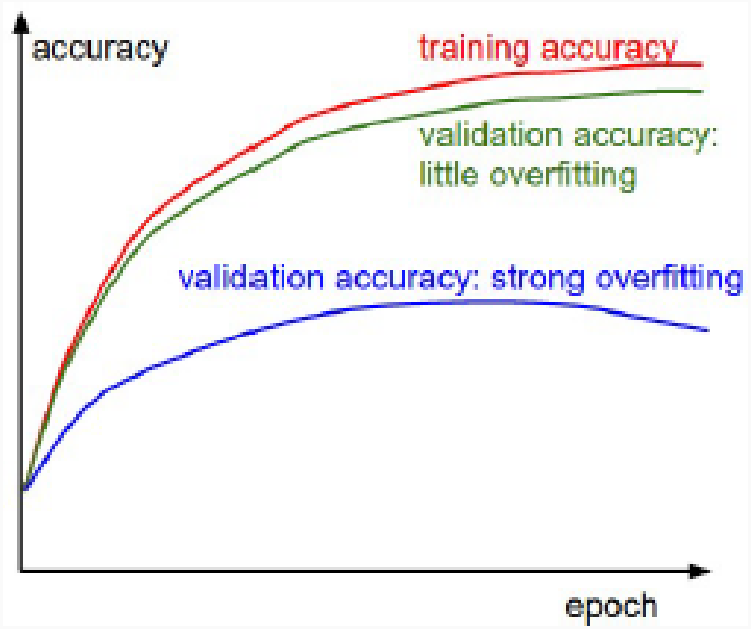
How bad the network parameters $\boldsymbol{W}$ is on this task

Find the network parameters $\boldsymbol{W}^*$ that minimize this value

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Babysit your (deep) network!



babysitting your deep network

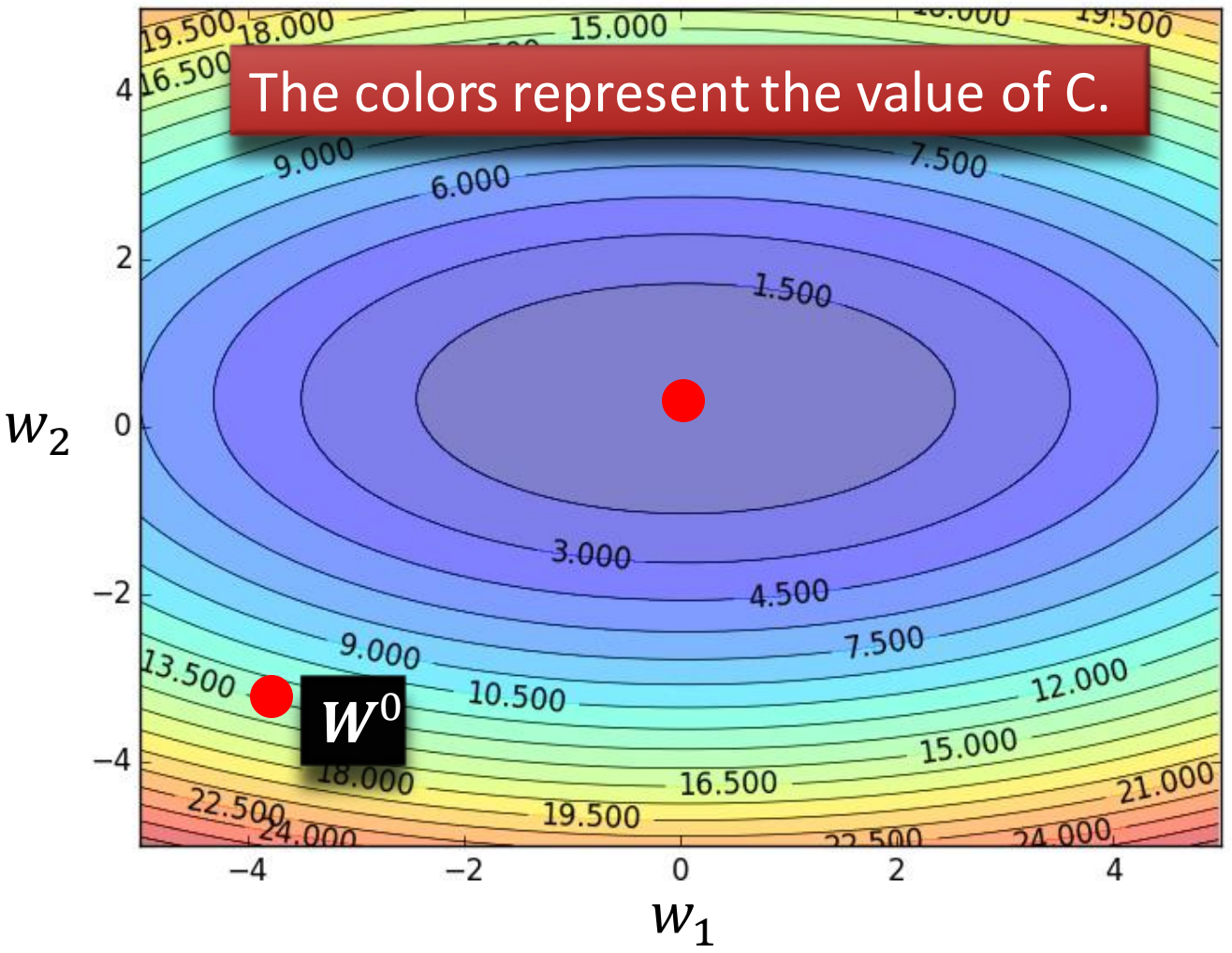
1. overfitting and underfitting
2. check accuracy before training [Saxe et al., 2011]
3. Y. Bengio : "*check if the model is powerful enough to overfit, if not then change model structure or make model larger*"

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

How to train the network? Gradient descent

Error Surface



$\boldsymbol{W} = \{w_1, w_2\}$

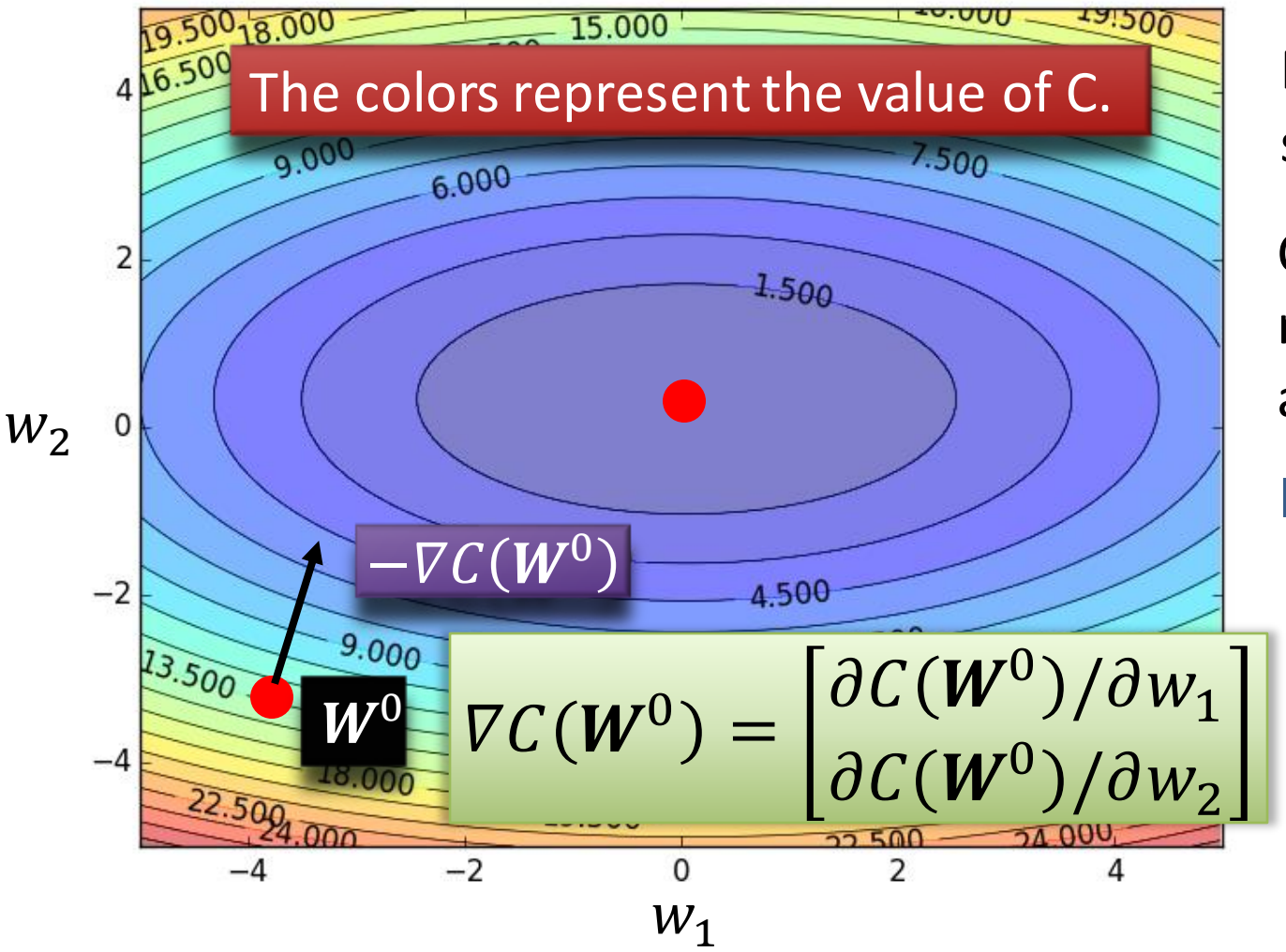
Randomly pick a starting point $\boldsymbol{W}^0$

Compute the negative gradient at $\boldsymbol{W}^0$

→ $-\nabla C(\boldsymbol{W}^0)$

How to train the network? Gradient descent

Error Surface



$\boldsymbol{W} = \{w_1, w_2\}$

Randomly pick a starting point $\boldsymbol{W}^0$

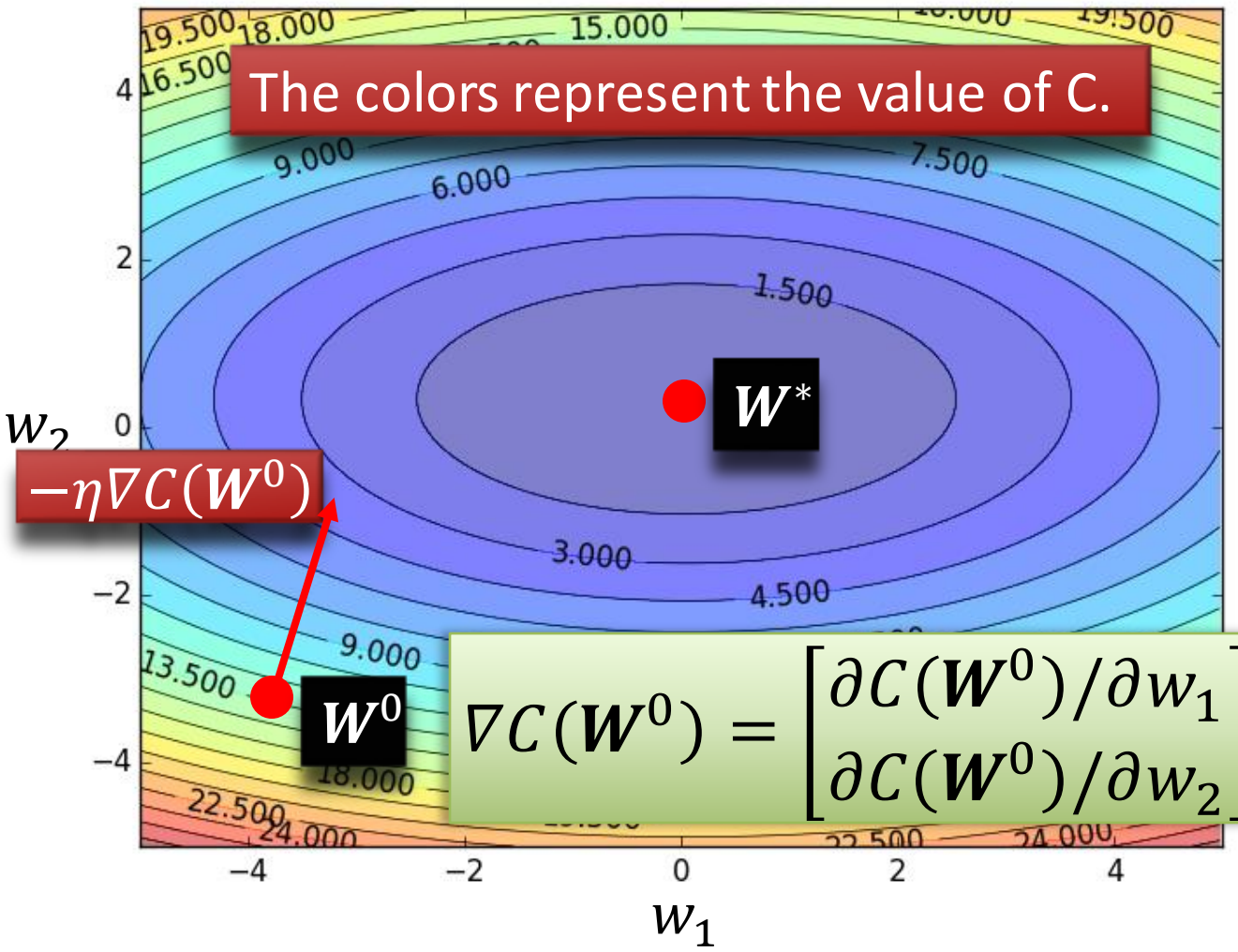
Compute the negative gradient at $\boldsymbol{W}^0$

→ $-\nabla C(\boldsymbol{W}^0)$

UNIVERSITÉ CÔTE D'AZUR
univ-cotedazur.fr

How to train the network? Gradient descent

Error Surface



$W = \{w_1, w_2\}$

Randomly pick a starting point W^0

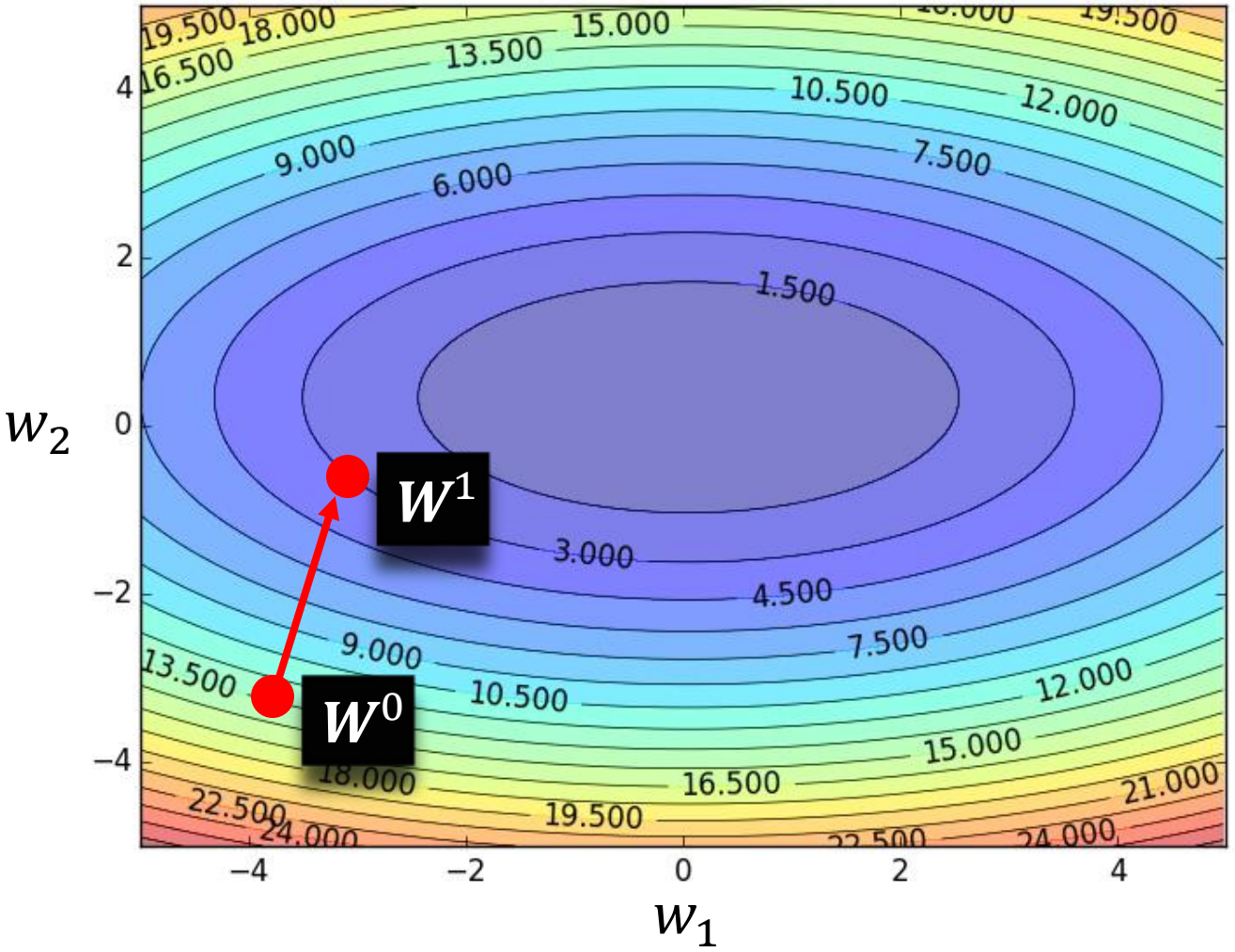
Compute the negative gradient at W^0

→ $-\nabla C(W^0)$

Times the learning rate η

→ $-\eta\nabla C(W^0)$

How to train the network? Gradient descent



Randomly pick a starting point $\boldsymbol{W}^0$

Compute the negative gradient at $\boldsymbol{W}^0$

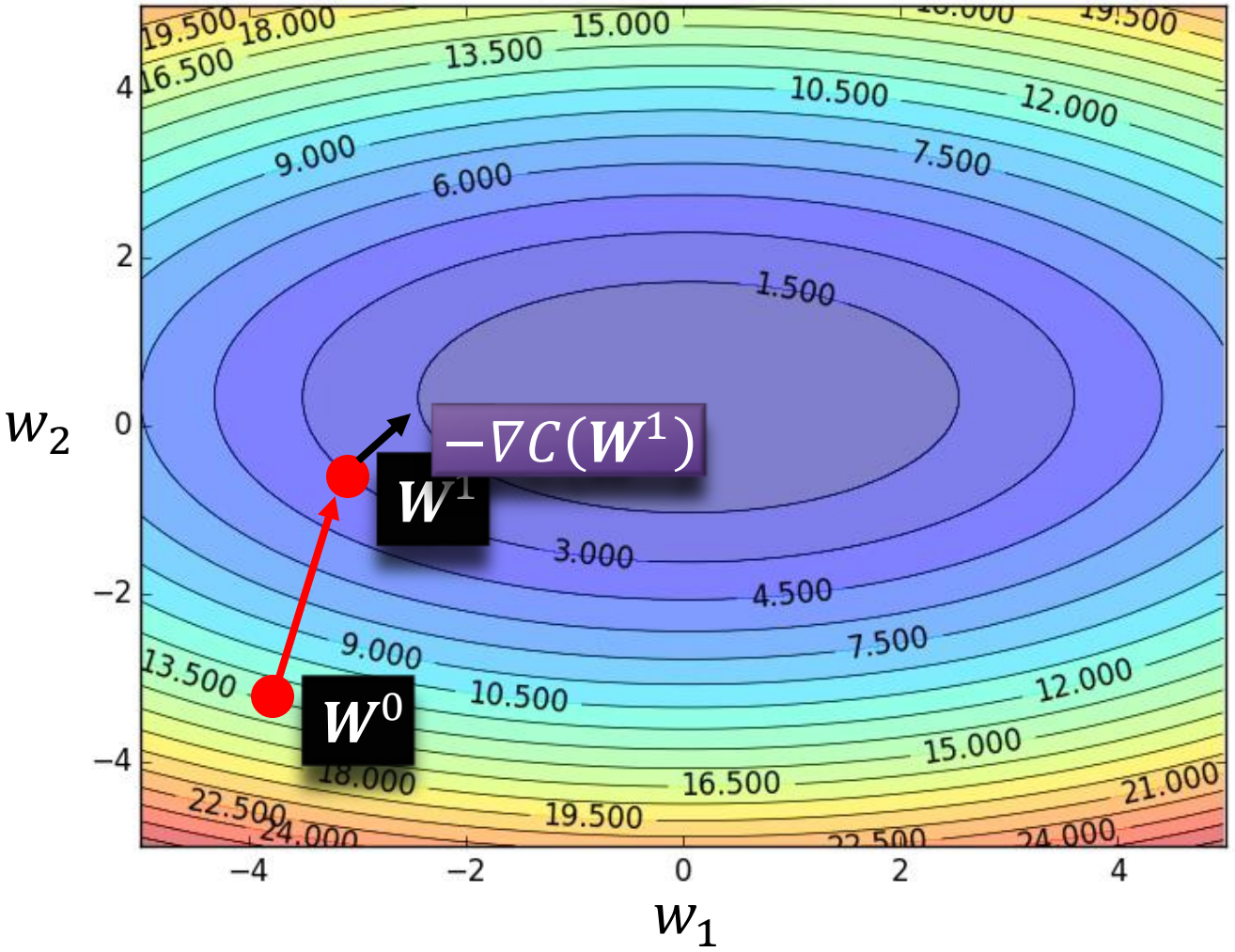
→ $-\nabla C(\boldsymbol{W}^0)$

Times the learning rate η

→ $-\eta\nabla C(\boldsymbol{W}^0)$

UNIVERSITÉ CÔTE D'AZUR
univ-cotedazur.fr

How to train the network? Gradient descent



Randomly pick a starting point $\boldsymbol{W}^0$

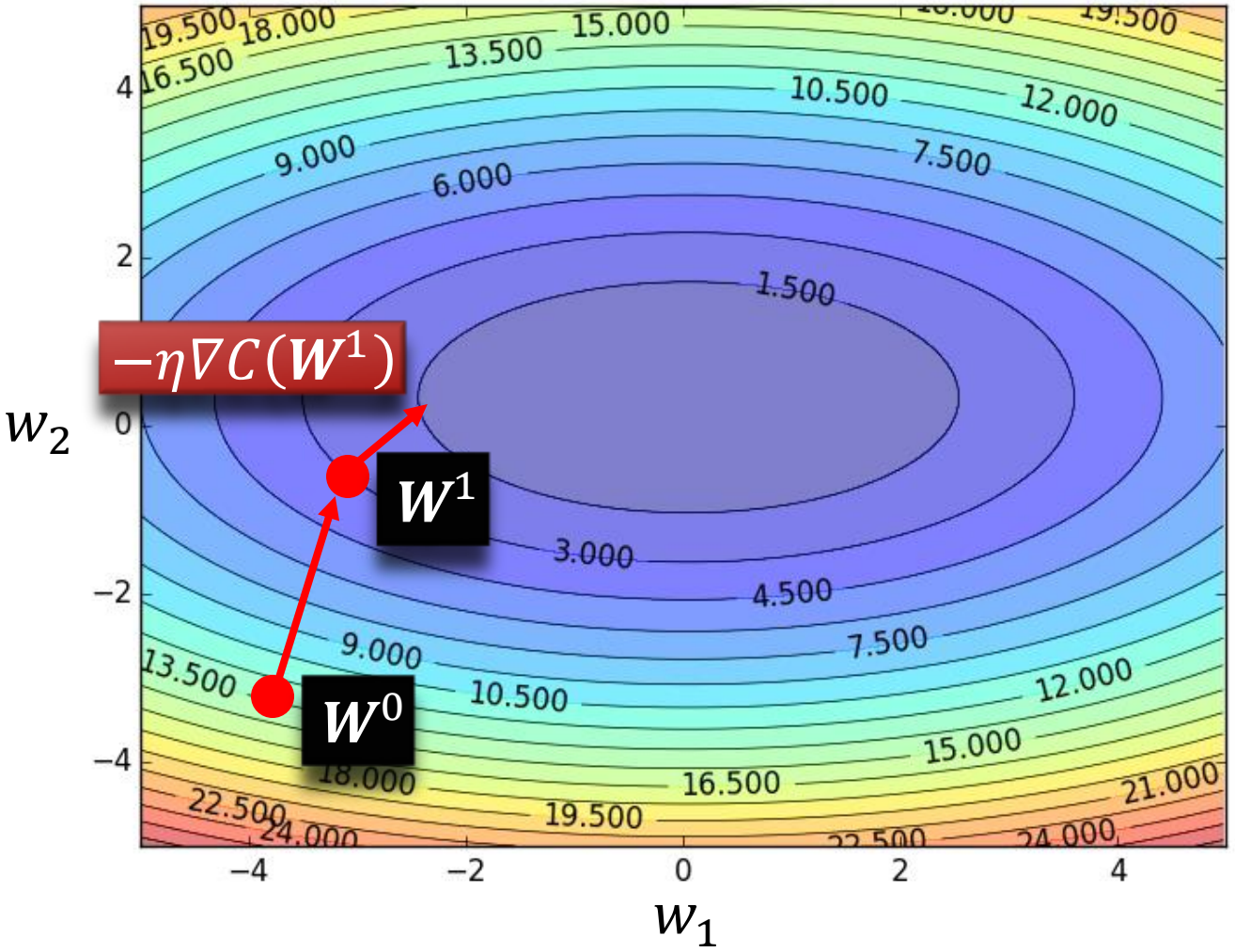
Compute the negative gradient at $\boldsymbol{W}^0$

→ $-\nabla C(\boldsymbol{W}^0)$

Times the learning rate η

→ $-\eta \nabla C(\boldsymbol{W}^0)$

How to train the network? Gradient descent



Randomly pick a starting point $\boldsymbol{W}^0$

Compute the negative gradient at $\boldsymbol{W}^0$

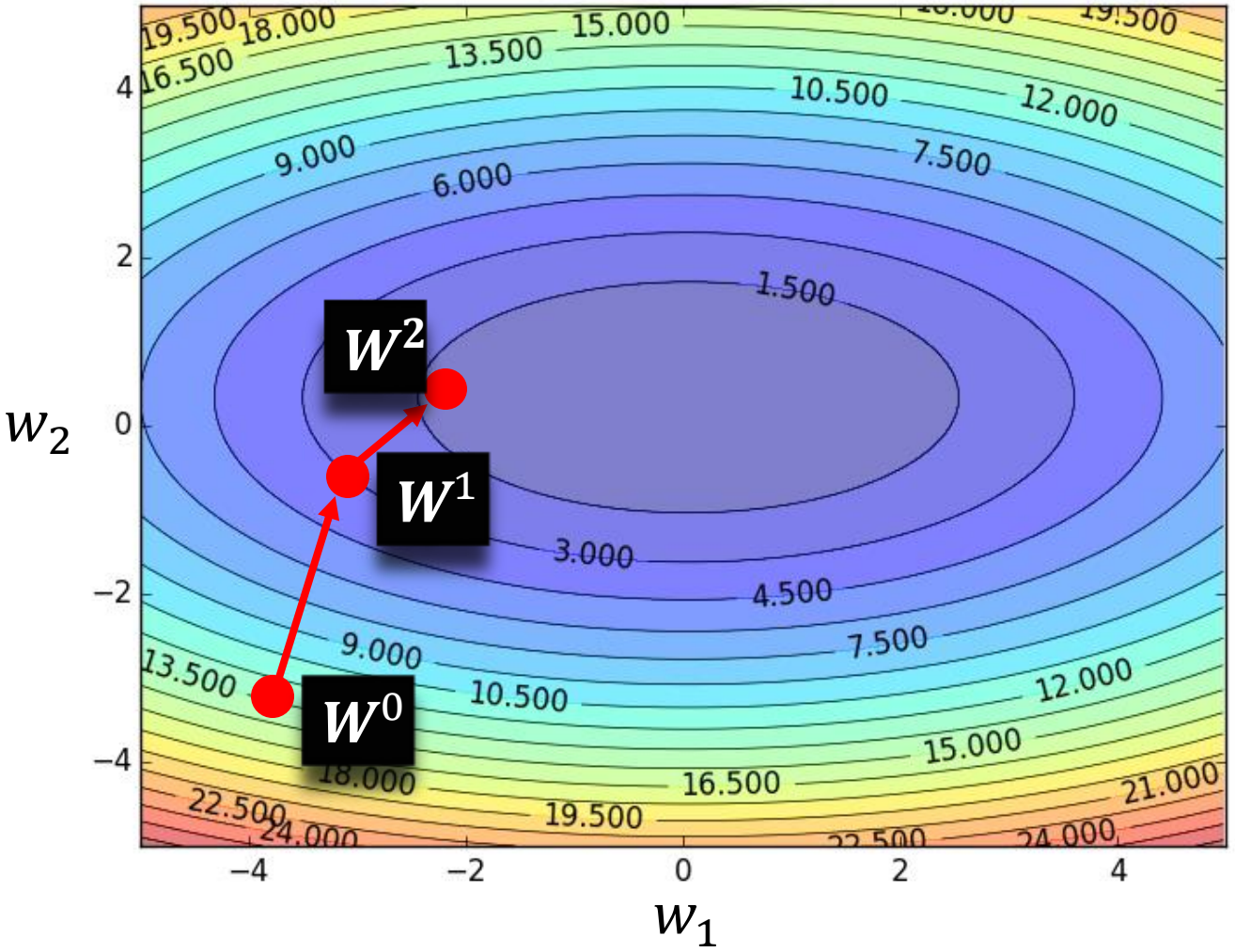
→ $-\nabla C(\boldsymbol{W}^0)$

Times the learning rate η

→ $-\eta\nabla C(\boldsymbol{W}^0)$

UNIVERSITÉ CÔTE D'AZUR
univ-cotedazur.fr

How to train the network? Gradient descent



Randomly pick a starting point $\boldsymbol{W}^0$

Compute the negative gradient at $\boldsymbol{W}^0$

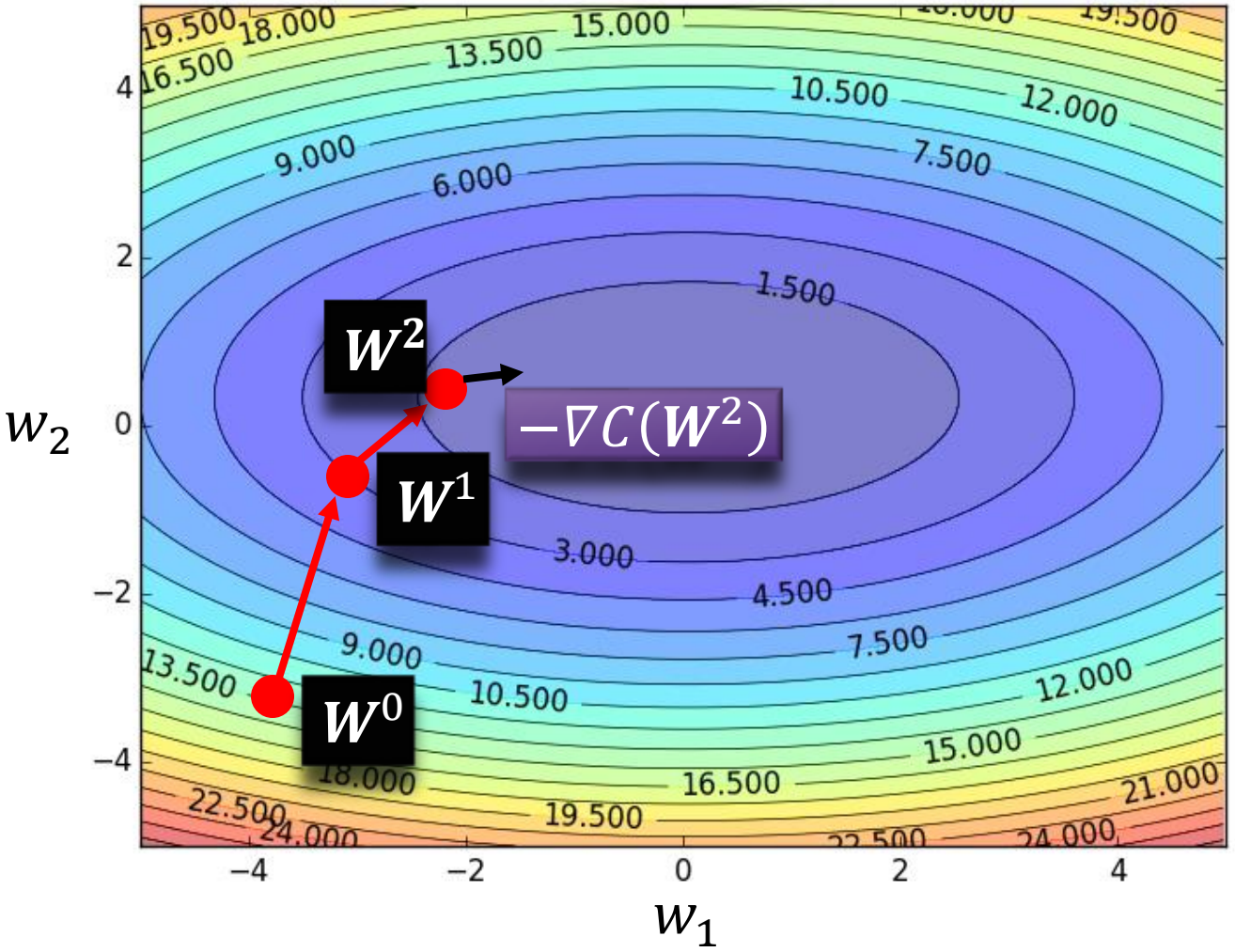
→ $-\nabla C(\boldsymbol{W}^0)$

Times the learning rate η

→ $-\eta\nabla C(\boldsymbol{W}^0)$

UNIVERSITÉ CÔTE D'AZUR
univ-cotedazur.fr

How to train the network? Gradient descent



Randomly pick a starting point $\boldsymbol{W}^0$

Compute the negative gradient at $\boldsymbol{W}^0$

→ $-\nabla C(\boldsymbol{W}^0)$

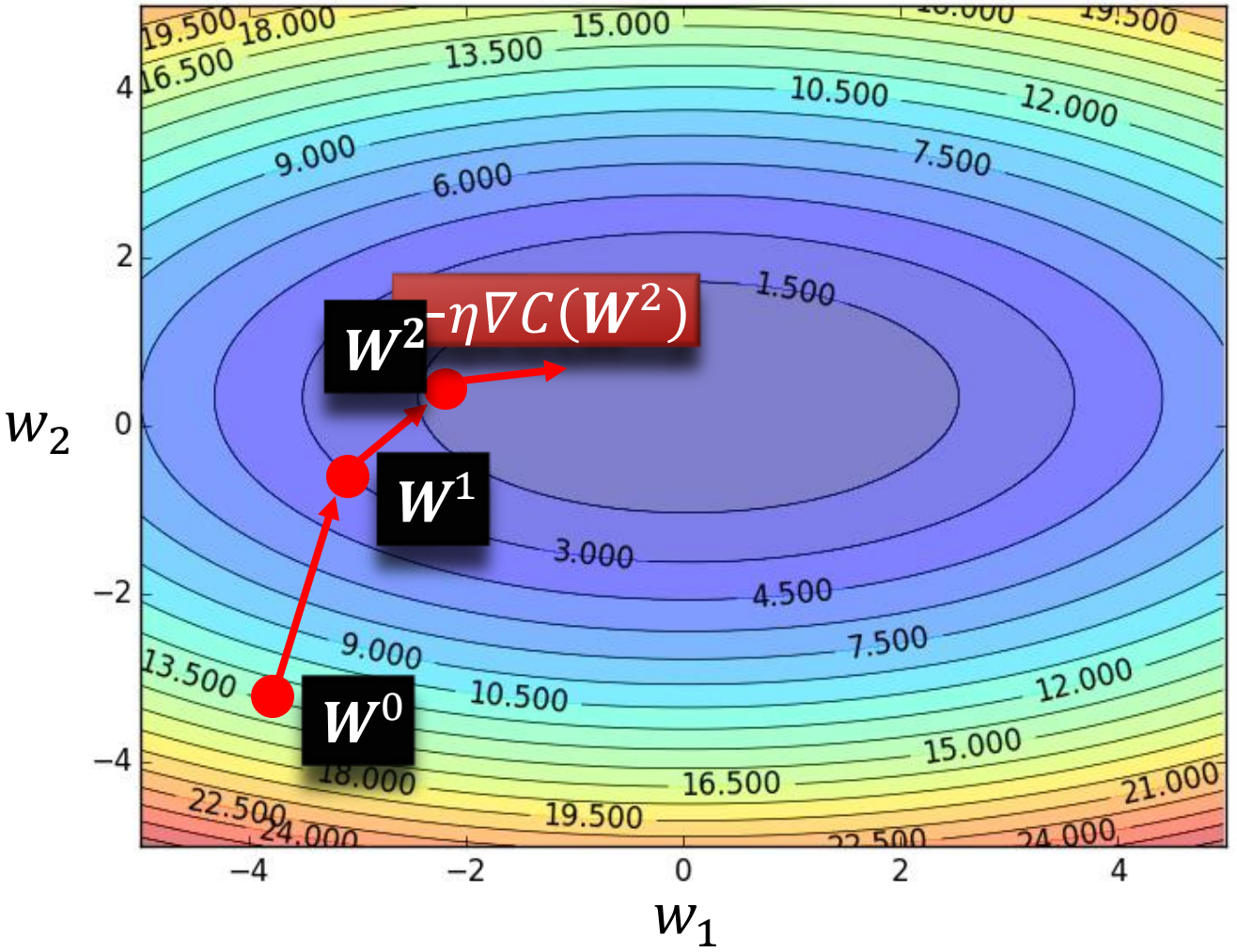
Times the learning rate η

→ $-\eta \nabla C(\boldsymbol{W}^0)$

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

How to train the network? Gradient descent



Randomly pick a starting point $\boldsymbol{W}^0$

Compute the negative gradient at $\boldsymbol{W}^0$

→ $-\nabla C(\boldsymbol{W}^0)$

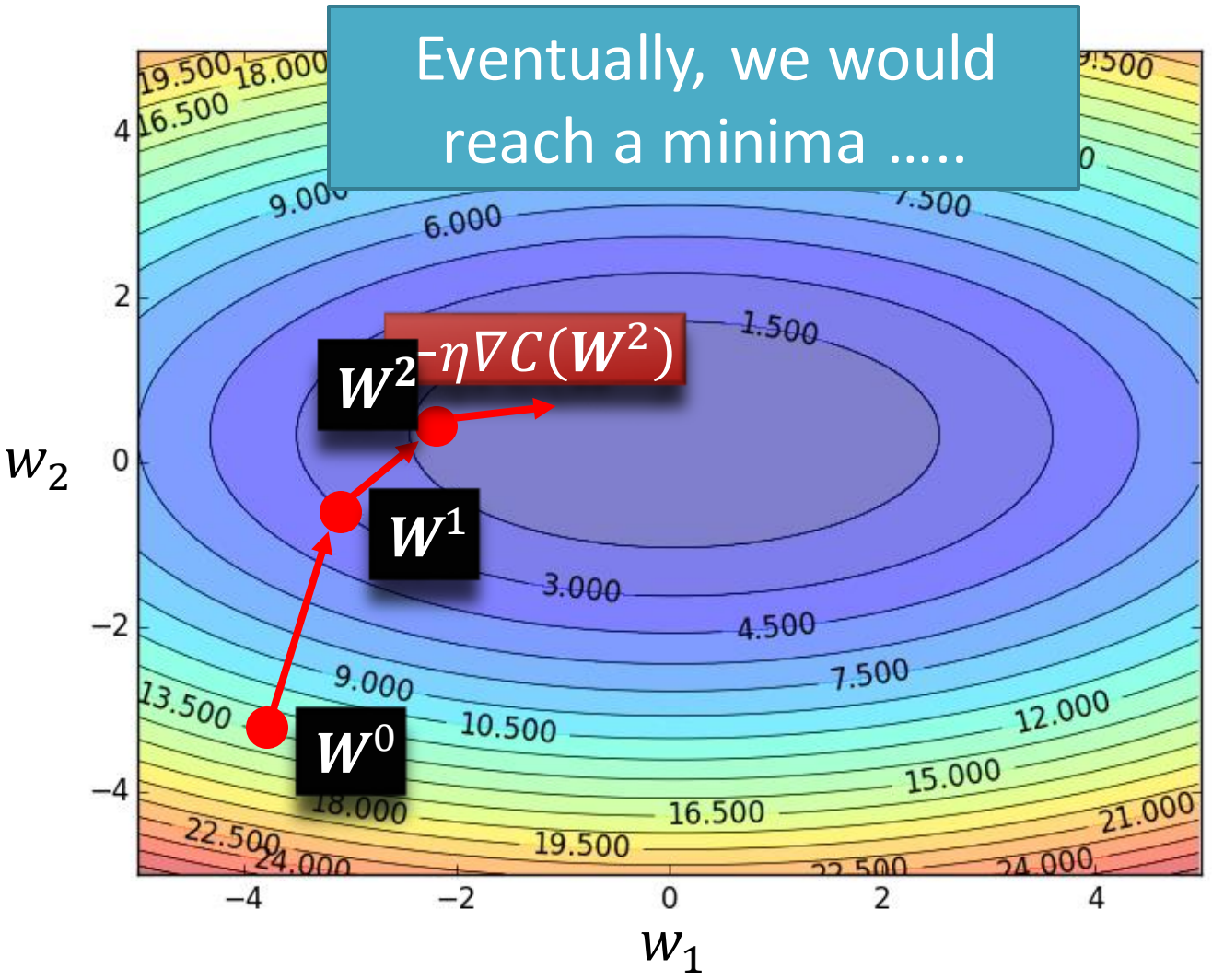
Times the learning rate η

→ $-\eta\nabla C(\boldsymbol{W}^0)$

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

How to train the network? Gradient descent



Randomly pick a starting point $\boldsymbol{W}^0$

Compute the negative gradient at $\boldsymbol{W}^0$

→ $-\nabla C(\boldsymbol{W}^0)$

Times the learning rate η

→ $-\eta\nabla C(\boldsymbol{W}^0)$

UNIVERSITÉ CÔTE D'AZUR

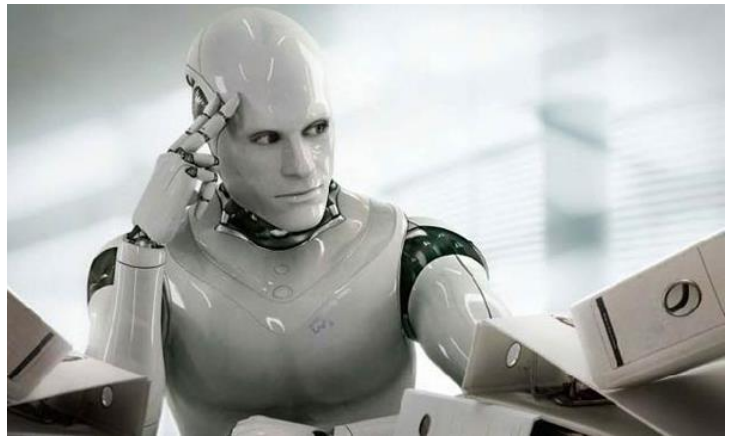
univ-cotedazur.fr

Gradient Descent

This is the "learning" of machines in deep learning ……

➡ Even alpha go using this approach.

What People imagine ……

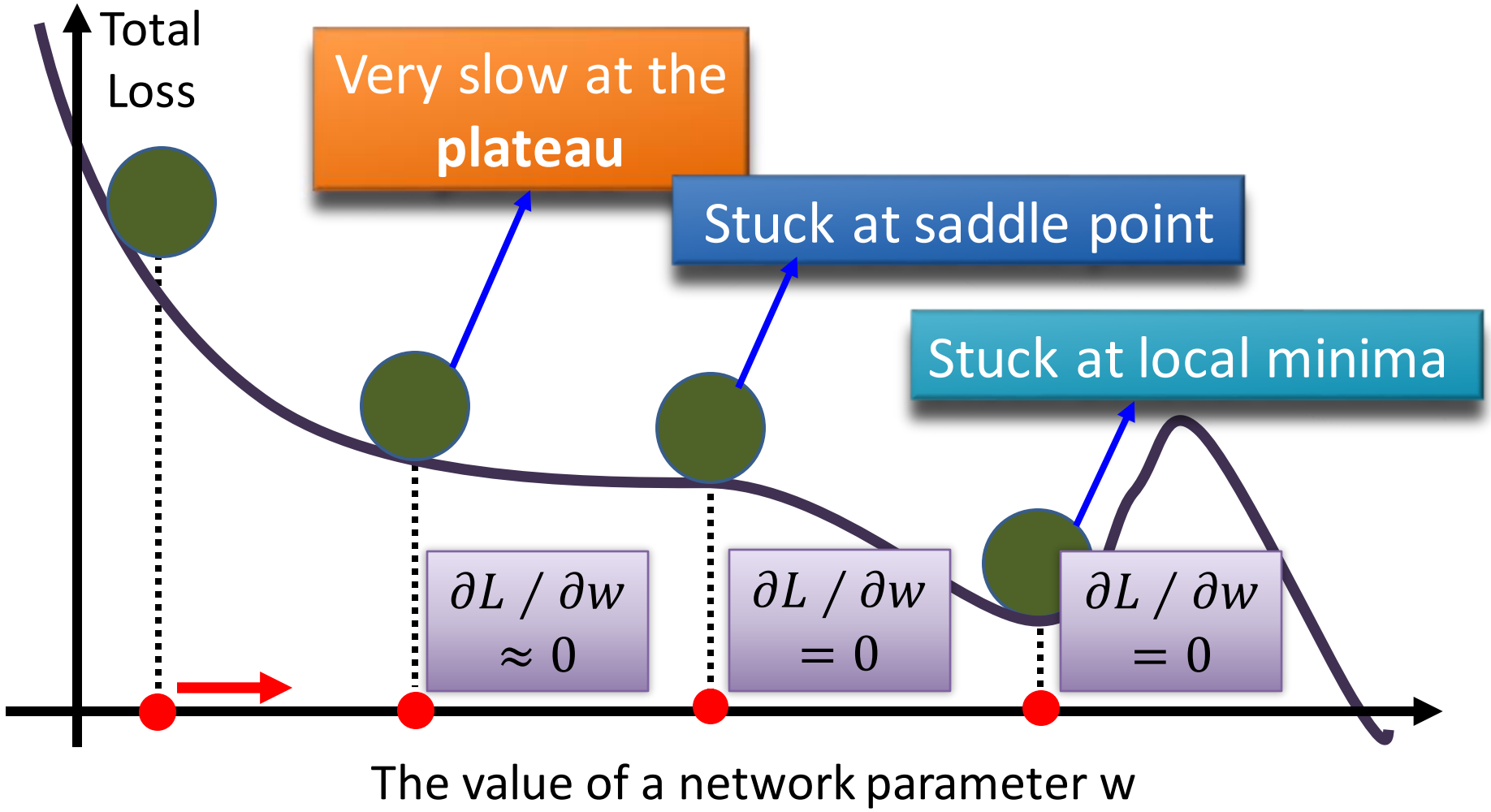


In reality …..

UNIVERSITÉ CÔTE D'AZUR

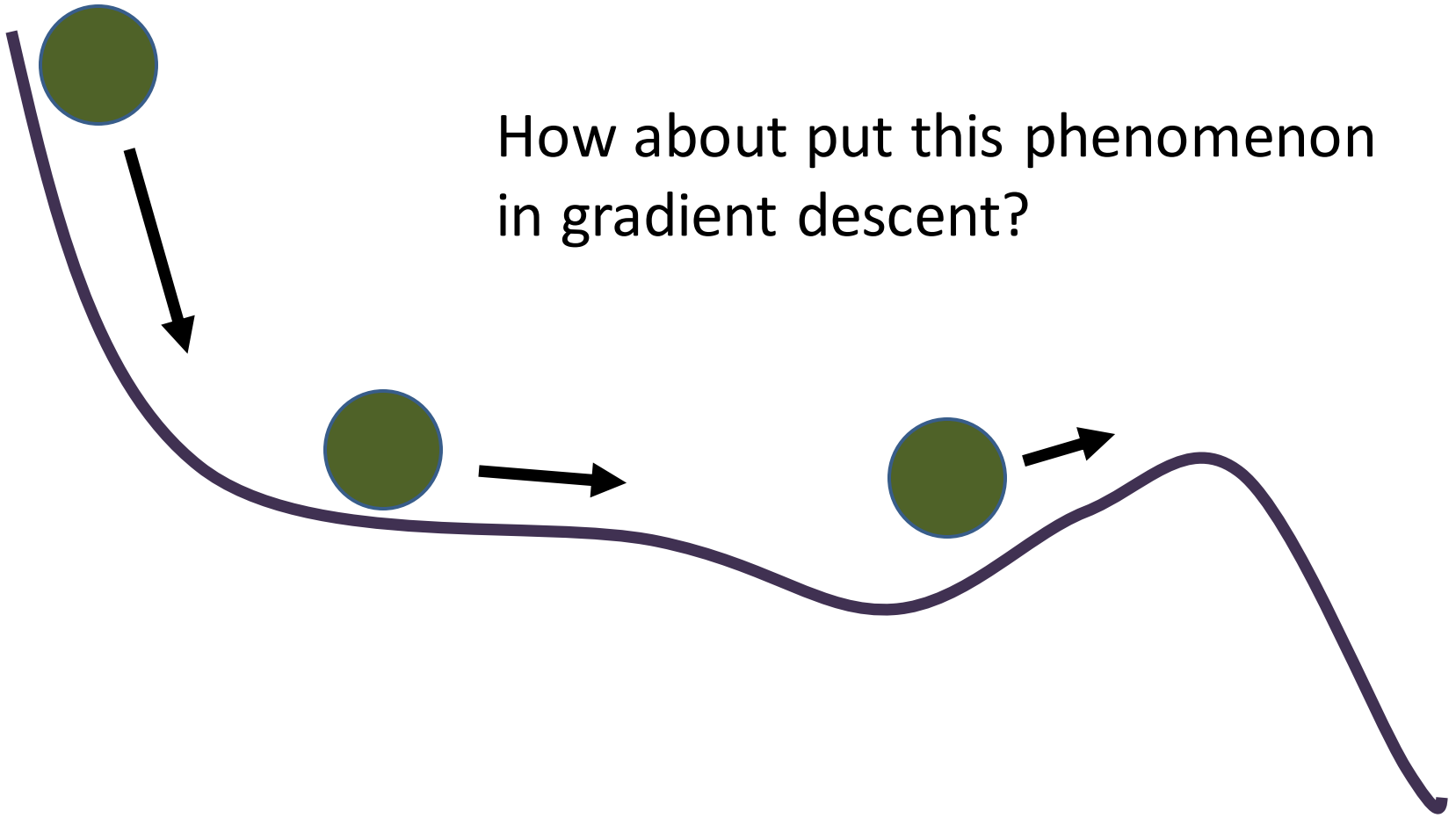
univ-cotedazur.fr

Hard to find optimal network parameters by gradient descent…



In physical world ……

- **Momentum = "a little flick"**

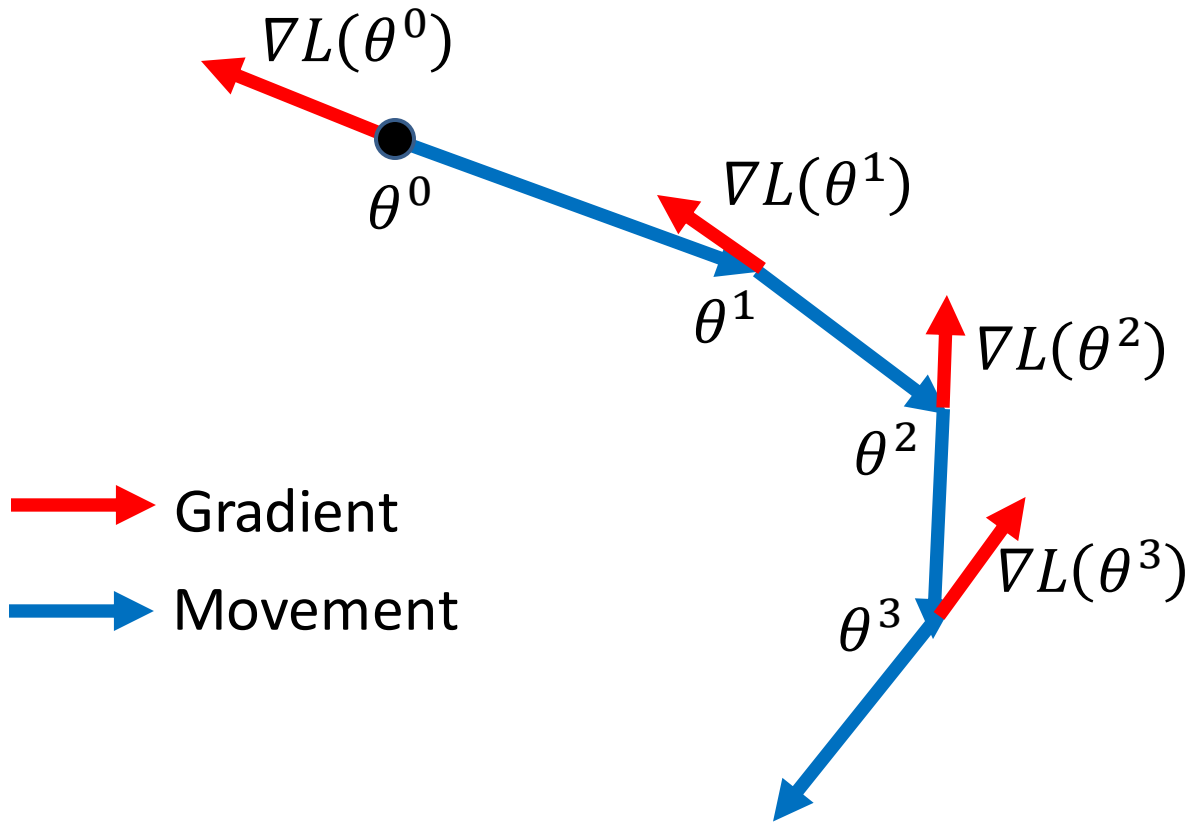


How about put this phenomenon in gradient descent?

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Vanilla Gradient Descent



Start at position θ^0

Compute gradient at θ^0

Move to $\theta^1 = \theta^0 - \eta\nabla L(\theta^0)$

Compute gradient at θ^1

Move to $\theta^2 = \theta^1 - \eta\nabla L(\theta^1)$

⋮

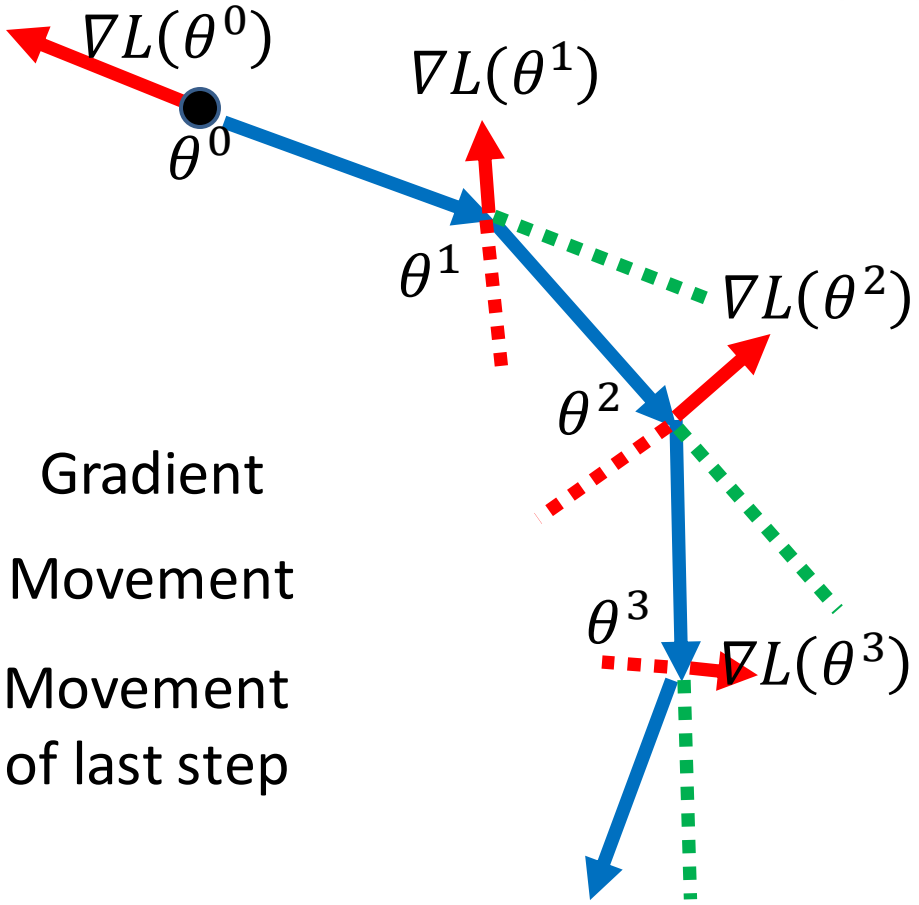
Stop until $\nabla L(\theta^t) \approx 0$

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Momentum = "a little flick"

Movement: movement of last step minus gradient at present



Start at point θ^0

Movement $v^0=0$

Compute gradient at θ^0

Movement $v^1 = \lambda v^0 - \eta \nabla L(\theta^0)$

Move to $\theta^1 = \theta^0 + v^1$

Compute gradient at θ^1

Movement $v^2 = \lambda v^1 - \eta \nabla L(\theta^1)$

Move to $\theta^2 = \theta^1 + v^2$

Movement not just based on gradient, but previous movement.

Momentum = "a little flick"

Movement: movement of last step minus gradient at present

v^i is actually the weighted sum of all the previous gradient: $\nabla L(\theta^0), \nabla L(\theta^1), \ldots \nabla L(\theta^{i-1})$

$v^0 = 0$

$v^1 = -\eta \nabla L(\theta^0)$

$v^2 = -\lambda \eta \nabla L(\theta^0) - \eta \nabla L(\theta^1)$

⋮

Start at point θ^0

Movement $v^0=0$

Compute gradient at θ^0

Movement $v^1 = \lambda v^0 - \eta \nabla L(\theta^0)$

Move to $\theta^1 = \theta^0 + v^1$

Compute gradient at θ^1

Movement $v^2 = \lambda v^1 - \eta \nabla L(\theta^1)$

Move to $\theta^2 = \theta^1 + v^2$

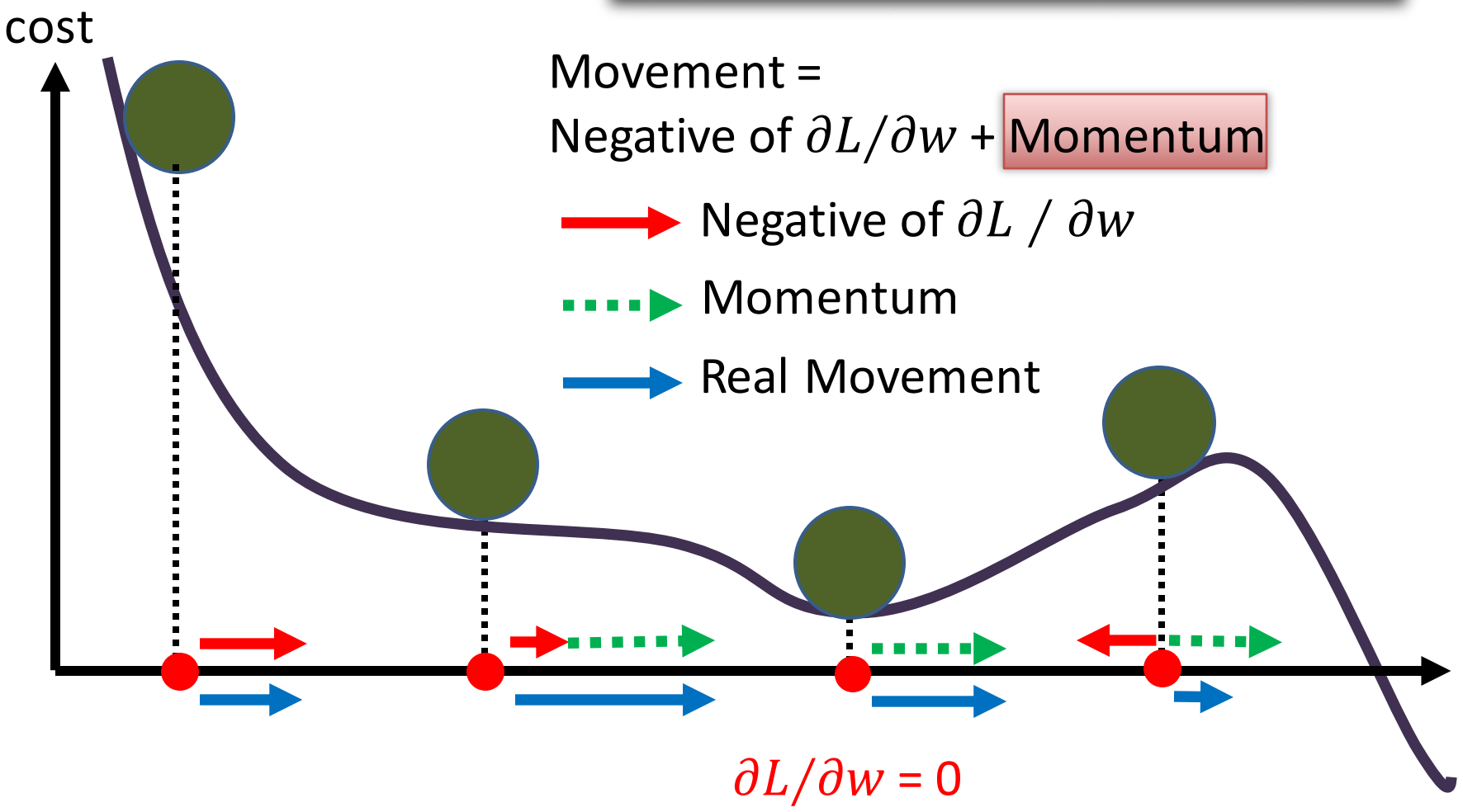
Movement not just based on gradient, but previous movement

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Momen ”

Still not guarantee reaching global minima, but give some hope ……



UNIVERSITÉ
COTE D'AZUR
univ-cotedazur.fr

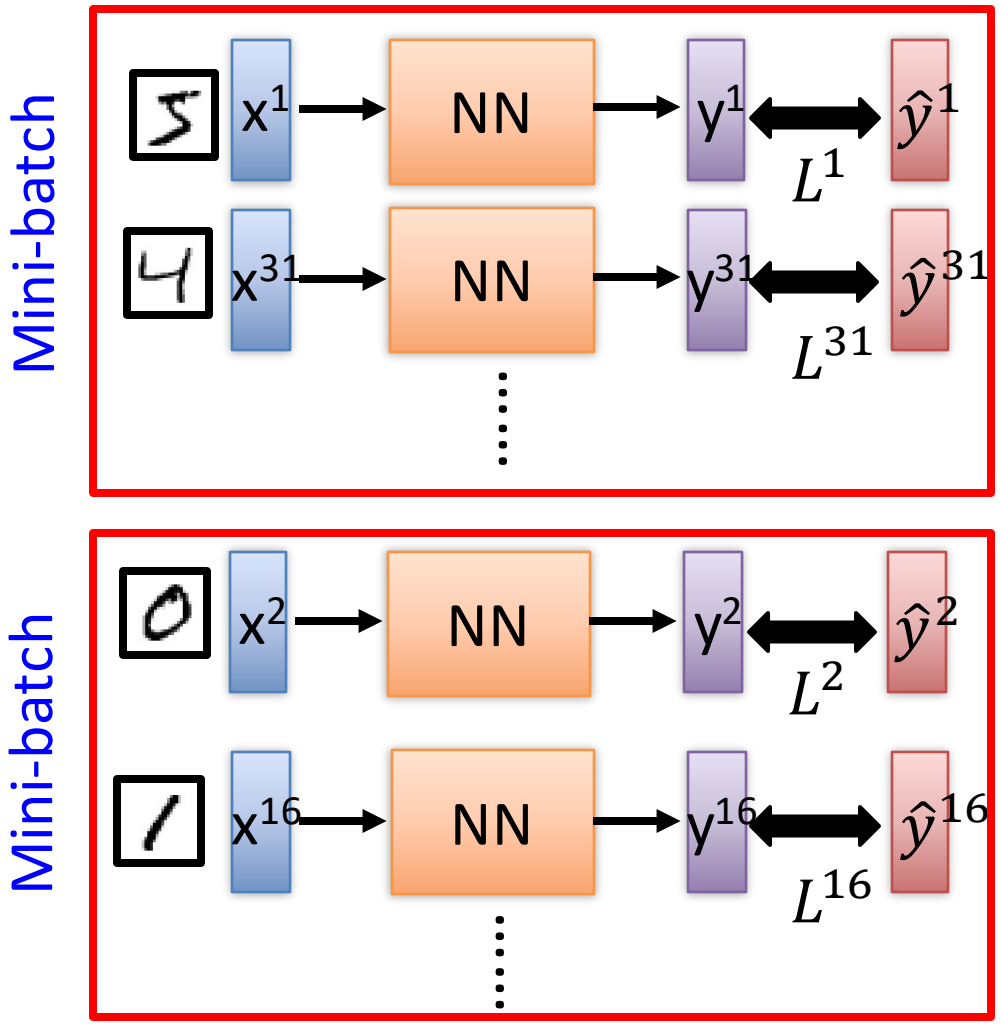
Deep tricks

Deep network initialization

- ***How to initialize the weights?***
  - ***All zero initialization (W=0)****: not good when the network* ***is deep****, every neuron computes the same output, have same gradients during back-propagation*
  - *Zero-mean Gaussian with 0.01 stddev, (W~N (0, 0.01))*
  - *Uniform Distribution*
  - *Orthogonal: The Eigen values of an orthogonally initialized matrix are one.*
  - ***Glorot normal/uniform*** *(aka* ***Xavier normal/uniform****): zero-mean gaussian initialization with variance scaled by number of input neurons + number of output neurons (Glorot et al., 2010), keeping the signal in a reasonable range of values through many layers* ***(W ~$\mathcal{N}$(0, 2/(input_neurons+output_neurons))*** *)*
  - ***He normal/uniform****: zero-mean gaussian initialization scaled by number of input neurons* ***(W ~ $\mathcal{N}$(0, 1/input_neurons)*** *)*

Train by *Mini-Batches* of training samples

Faster | Better!



- Randomly initialize $\boldsymbol{W}^0$
- Pick the 1st batch

  $C = L^1 + L^{31} + \cdots$

  $\boldsymbol{W}^1 \leftarrow \boldsymbol{W}^0 - \eta \nabla C(\boldsymbol{W}^0)$
- Pick the 2nd batch

  $C = L^2 + L^{16} + \cdots$

  $\boldsymbol{W}^2 \leftarrow \boldsymbol{W}^1 - \eta \nabla C(\boldsymbol{W}^1)$

  ⋮
- Until all mini-batches have been picked

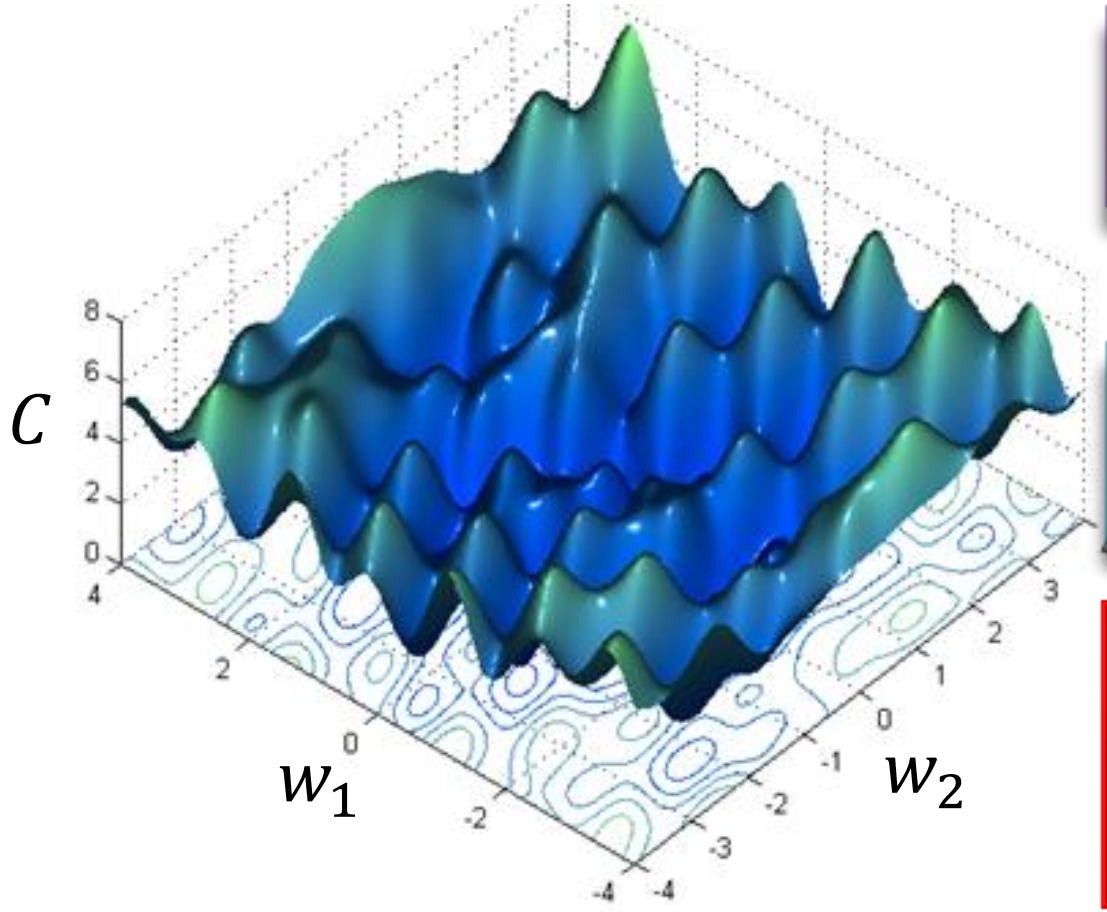
one epoch

Repeat the above process

UNIVERSITÉ CÔTE D'AZUR
univ-cotedazur.fr

Gradient descent…Problem of local minima

- Gradient descent never guarantee global minima but…



Different initial point W^0

Reach different minima, thus different results

The Loss Surfaces of Multilayer Networks (AISTAT 2015)
A. Choromanska, M. Henaff, M. Mathieu, G. Ben Arous, Y. LeCun

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Keep normalizing internal features throughout the deep network

- ***By integrating Batch-Normalization Layer***

Internal covariate shift: a change in the distribution of activations owing to parameter updates might slow learning

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

- Faster learning
- Increase accuracy
- Potentially allows a higher learning rate
- Prevent from bad initialization

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

How to easily design the right network architecture: Transfer Learning!!

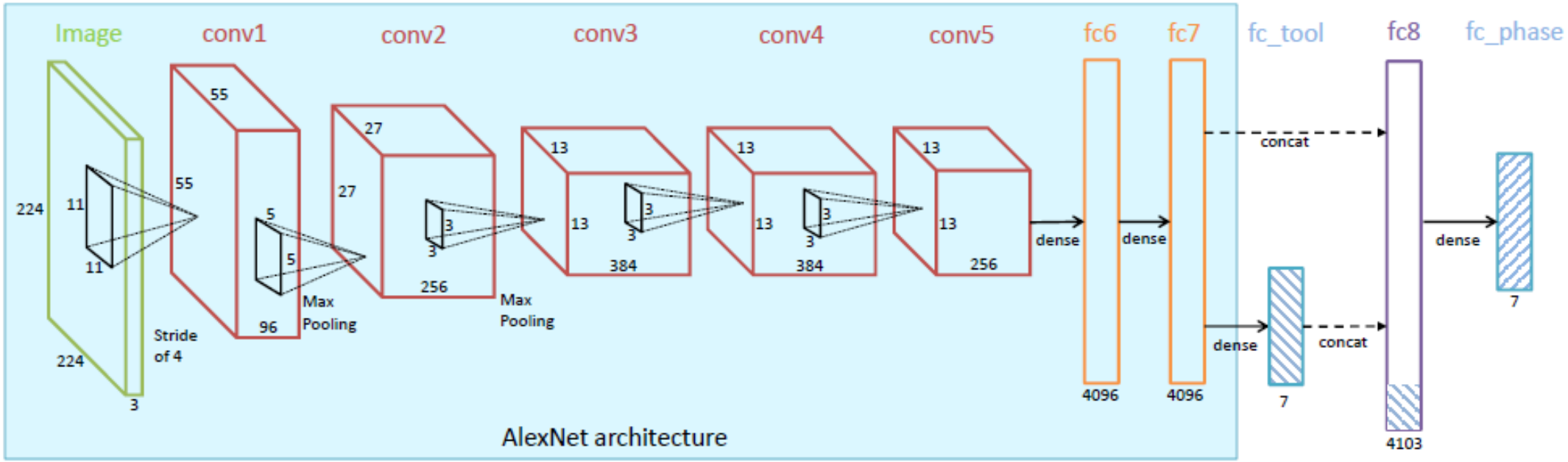


Fig. 2: EndoNet architecture (best seen in color). The layers shown in the turquoise rectangle are the same as in the AlexNet architecture.

Grasper Bipolar Hook Clipper

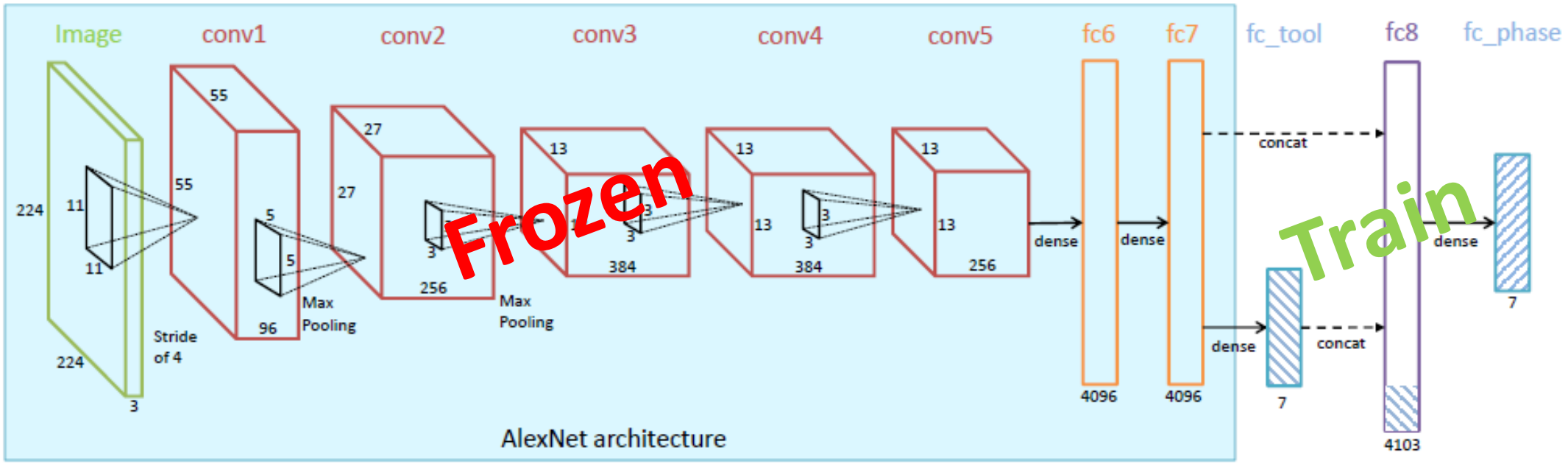
Scissors Irrigator Specimen bag

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

How to easily design the right network architecture: Transfer Learning!!

Very small training set

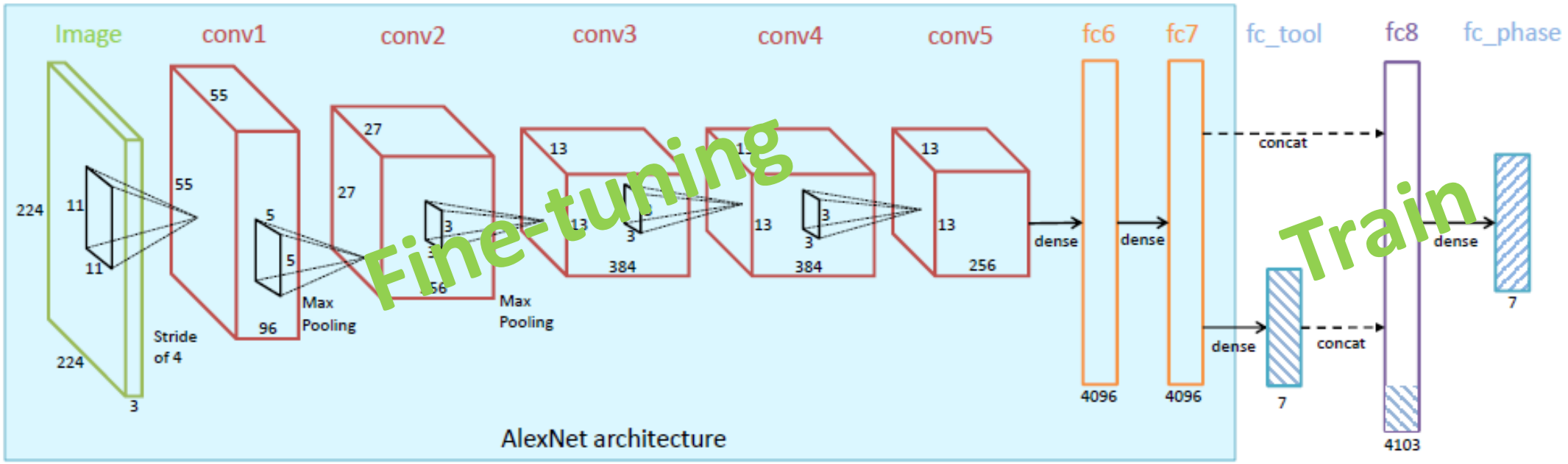


UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

How to easily design the right network architecture: Transfer Learning!!

Small training set

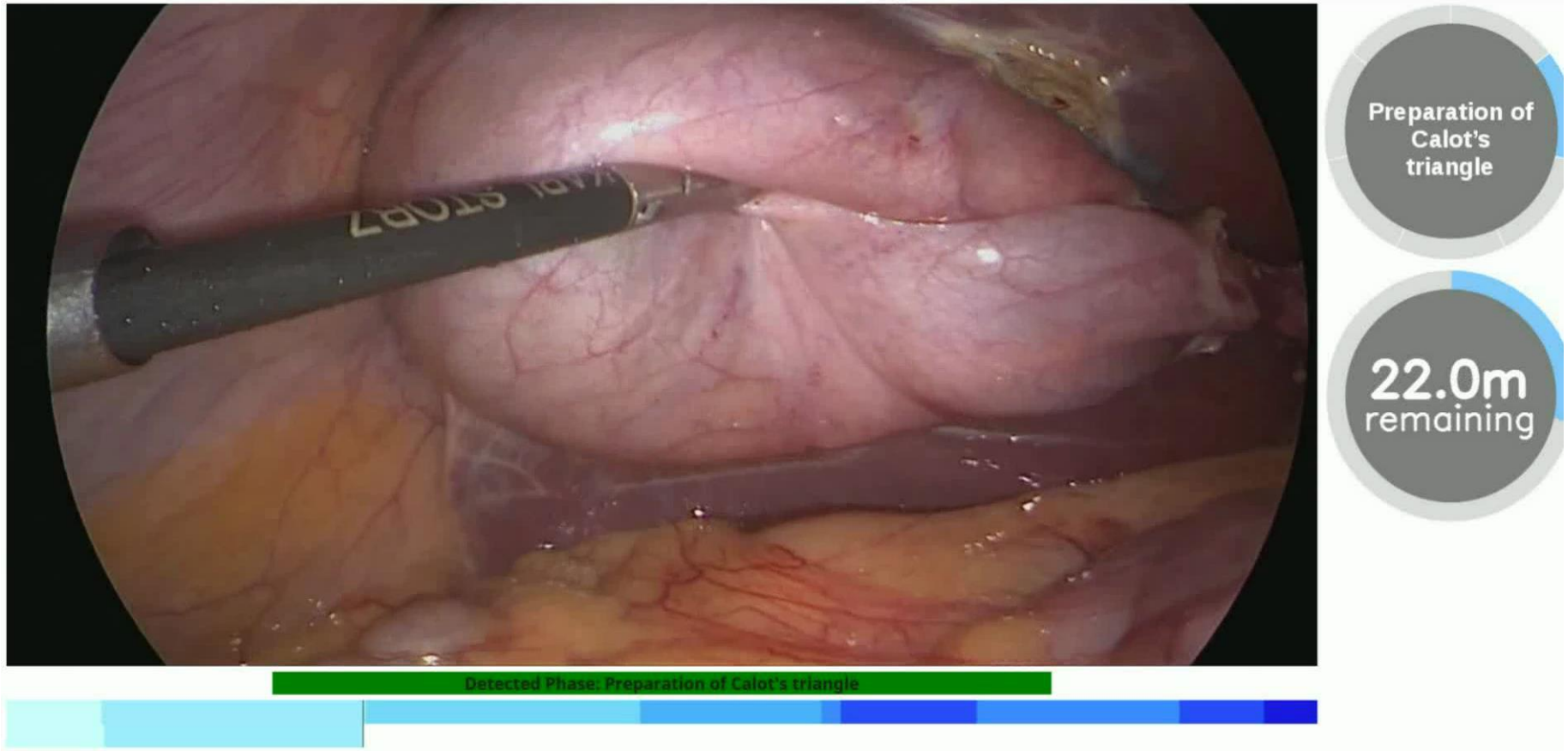


UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Challenge Surgery: Laparoscopic Surgery

Surgery workflow analysis



UNIVERSITÉ CÔTE D'AZUR

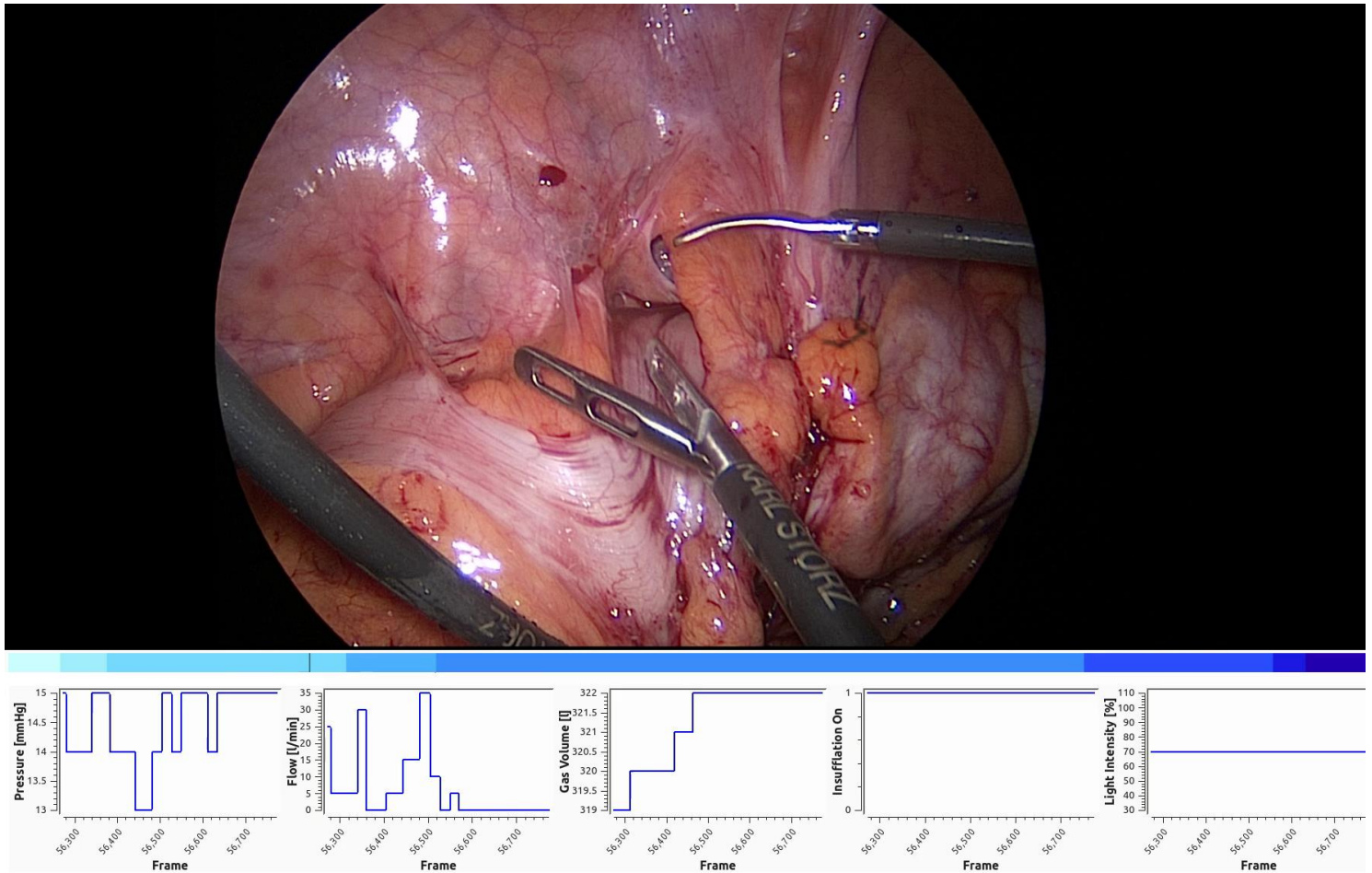
univ-cotedazur.fr

Task

Phase detection

Video

Surgical Devices

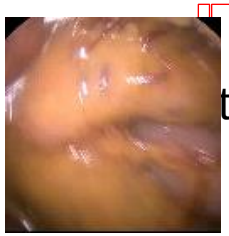


UNIVERSITÉ CÔTE D'AZUR

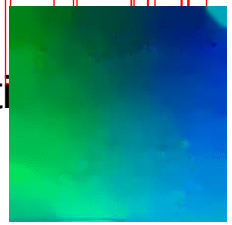
univ-cotedazur.fr

Method

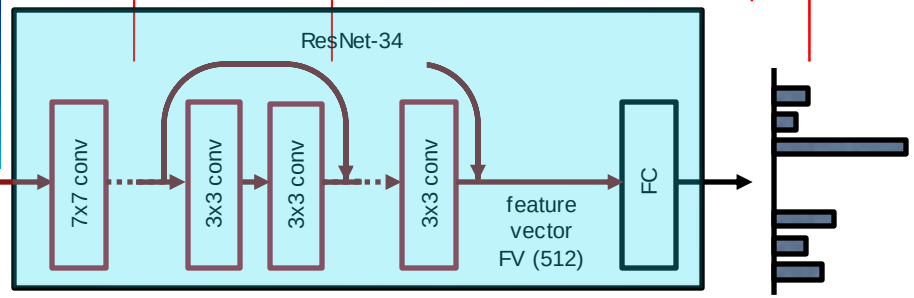
Temporal Network



224x224x3



224x224x20



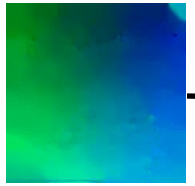
Number of target classes:
Rectal resection: 11
Sigmoid resection: 10
Proctolectomy: 12

Spatial network accuracy:
Rectal resection: 62.91%
Sigmoid resection: 63.01%
Proctolectomy: 63.26%

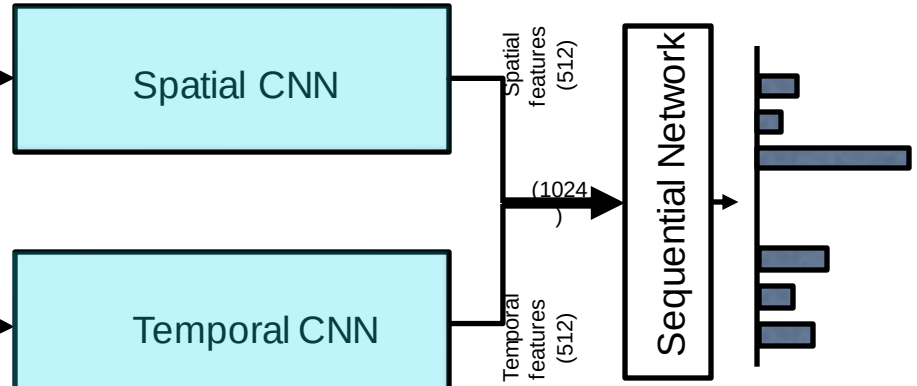
Temporal network accuracy:
Rectal resection: 49.88%
Sigmoid resection: 48.56%
Proctolectomy: 46.96%

Final Network

224x224x3



224x224x20



Final Network Accuracy:

Rectal resection (8): **80.7%**	sigmoid resection (7): **73.5%**	Proctocolectomy (1): **71.3%**
Rectal resection (6): **79.9%**	sigmoid resection (1): **54.7%**	Proctocolectomy (4): **73.9%**

Results for rectal resection video #8; GT in green, prediction in red

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

And the winner is…

	Data used	Average Jaccard	Median Jaccard	Accuracy	
1	Video	40%	38%	61%	Team UCA - UCLan
2	Video + Device	38%	38%	60%	Team NCT
3	Video	25%	25%	57%	Team TUM
4	Device	16%	16%	36%	Team TUM
5	Video	8%	7%	21%	Team FFL

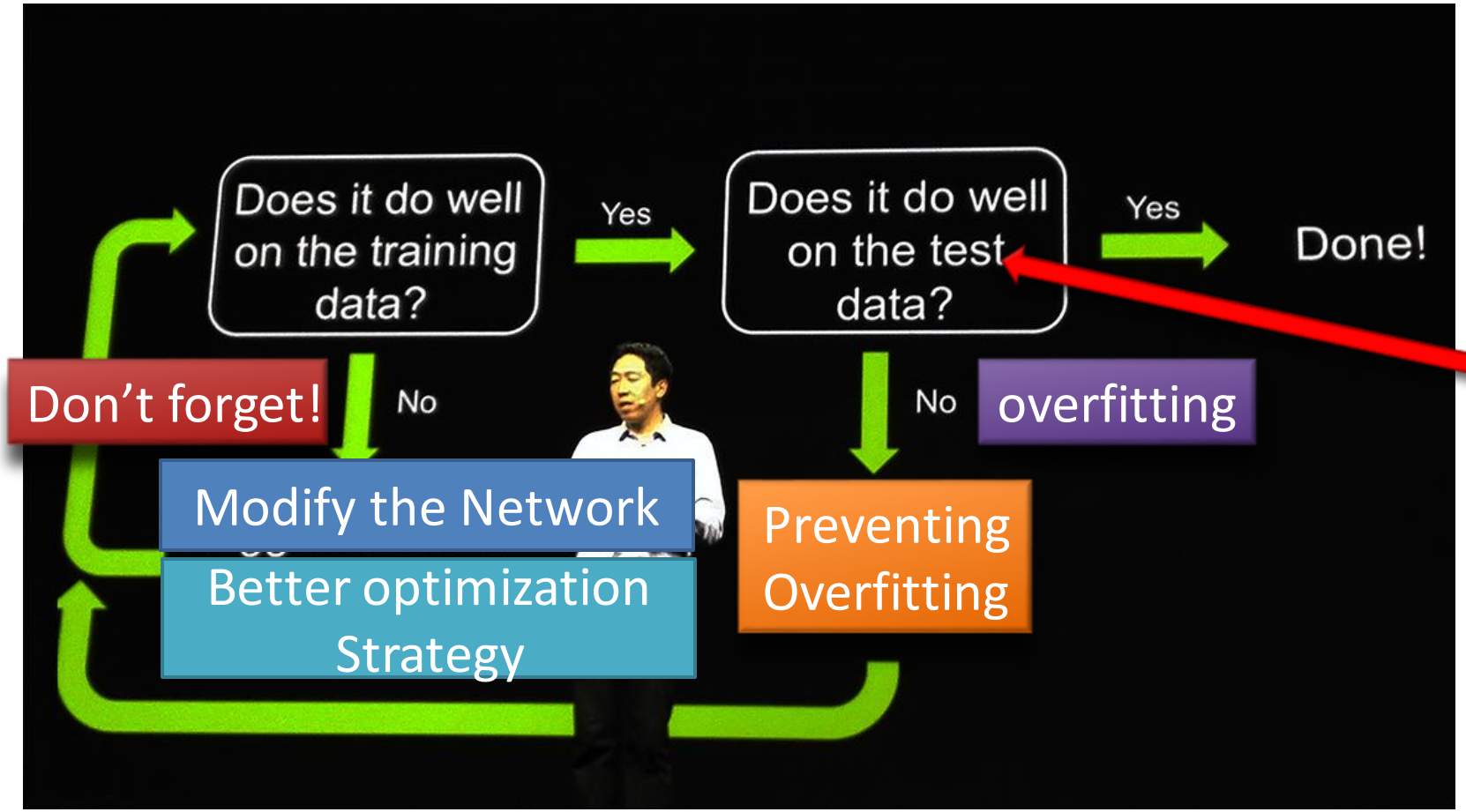
UNIVERSITÉ
COTE D'AZUR
univ-cotedazur.fr

Deep recipes

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Recipes of Deep Learning



Validation set

http://www.gizmodo.com.au/2015/04/the-basic-recipe-for-machine-learning-explained-in-a-single-powerpoint-slide/

Recipes of Deep Learning

Modify the Network

- New activation functions, for example, ReLU or LeakyReLU…

Better optimization Strategy

- Adaptive learning rates

Prevent Overfitting

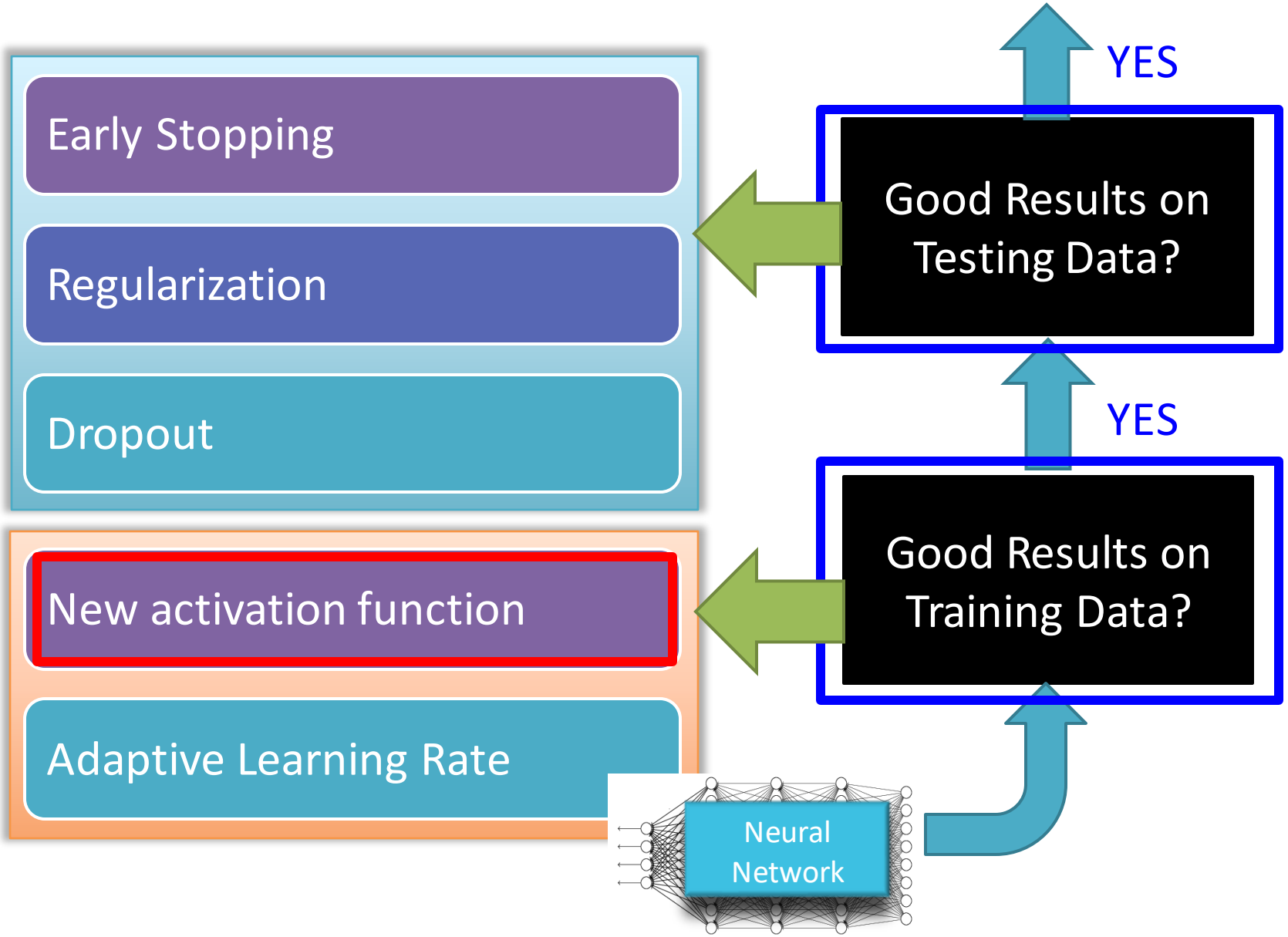
- Dropout

Only use this approach when you already obtained good results on the training data.

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Recipes of Deep Learning

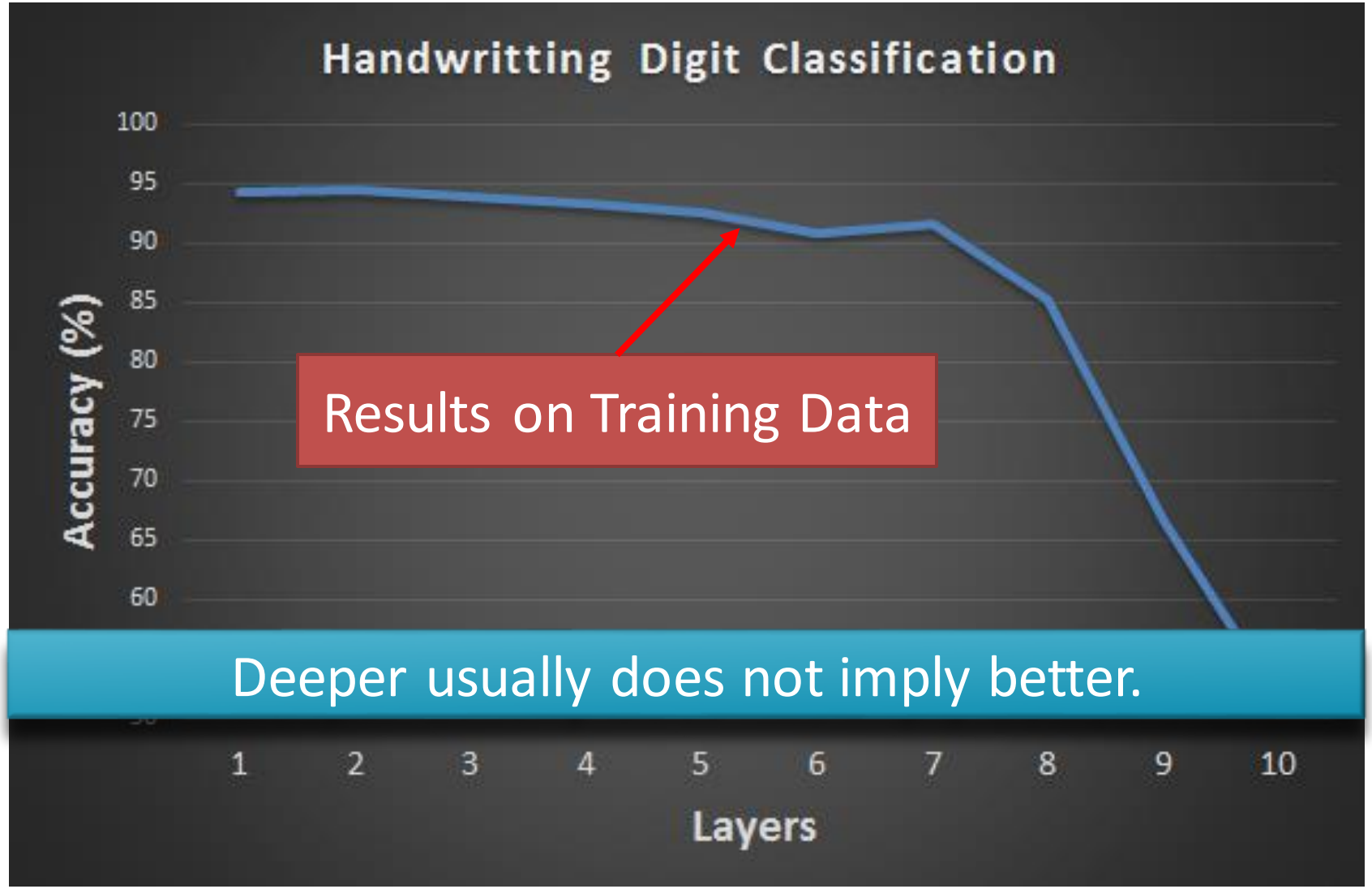


UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Recipe of Deep Learning

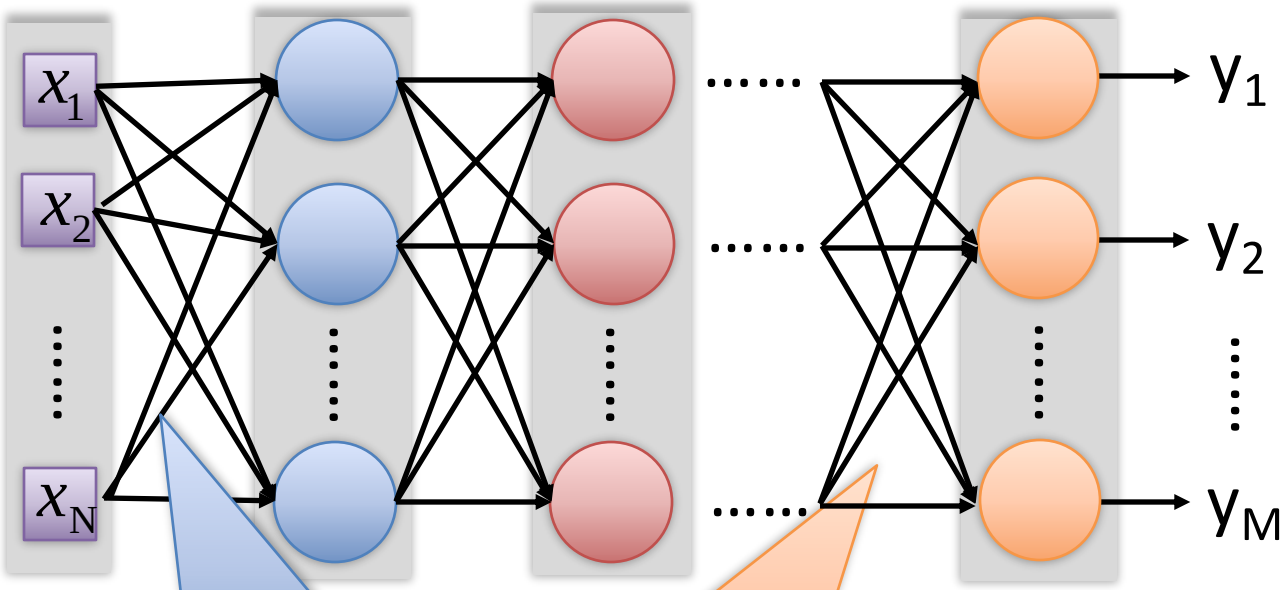
- ***Hard to get the power of Deep …***



UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Vanishing Gradient Problem



Smaller gradients

Learn very slow

Almost random

Larger gradients

Learn very fast

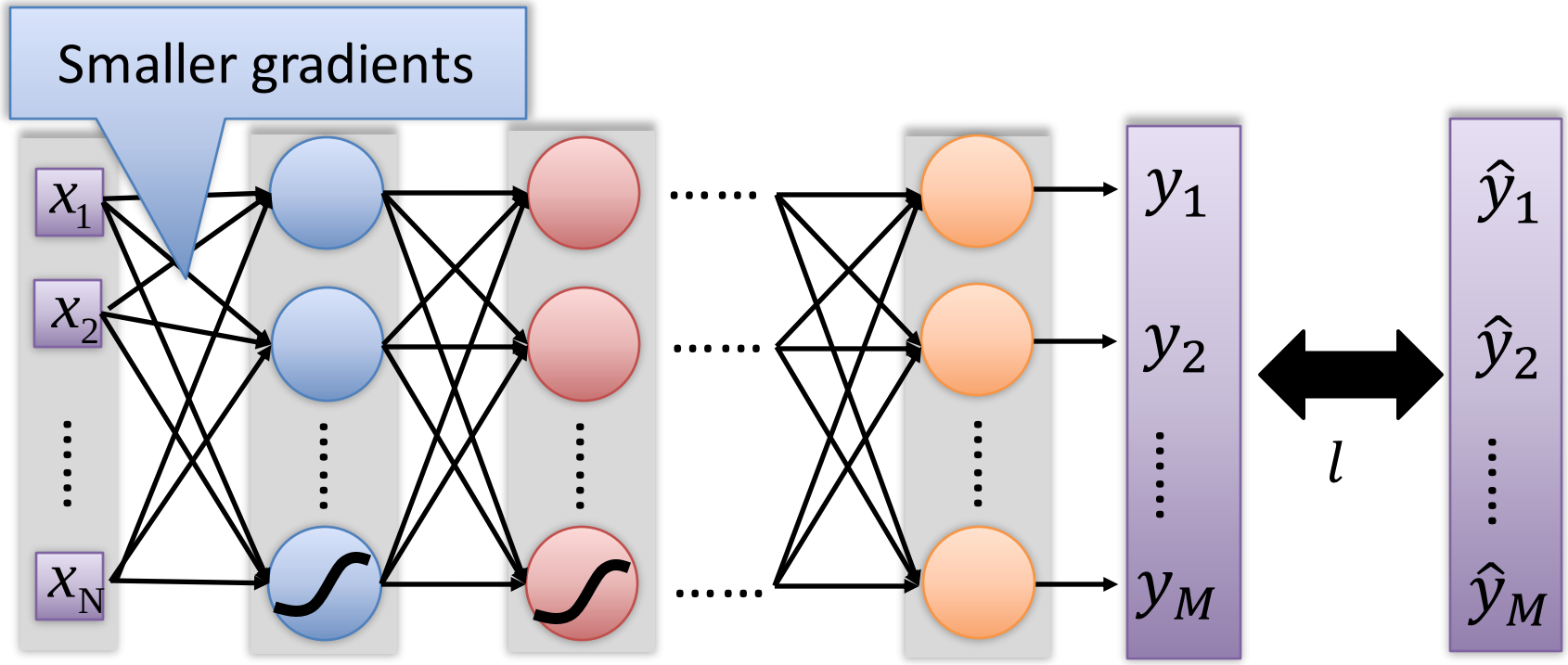
Already converge

based on random!?

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Vanishing Gradient Problem



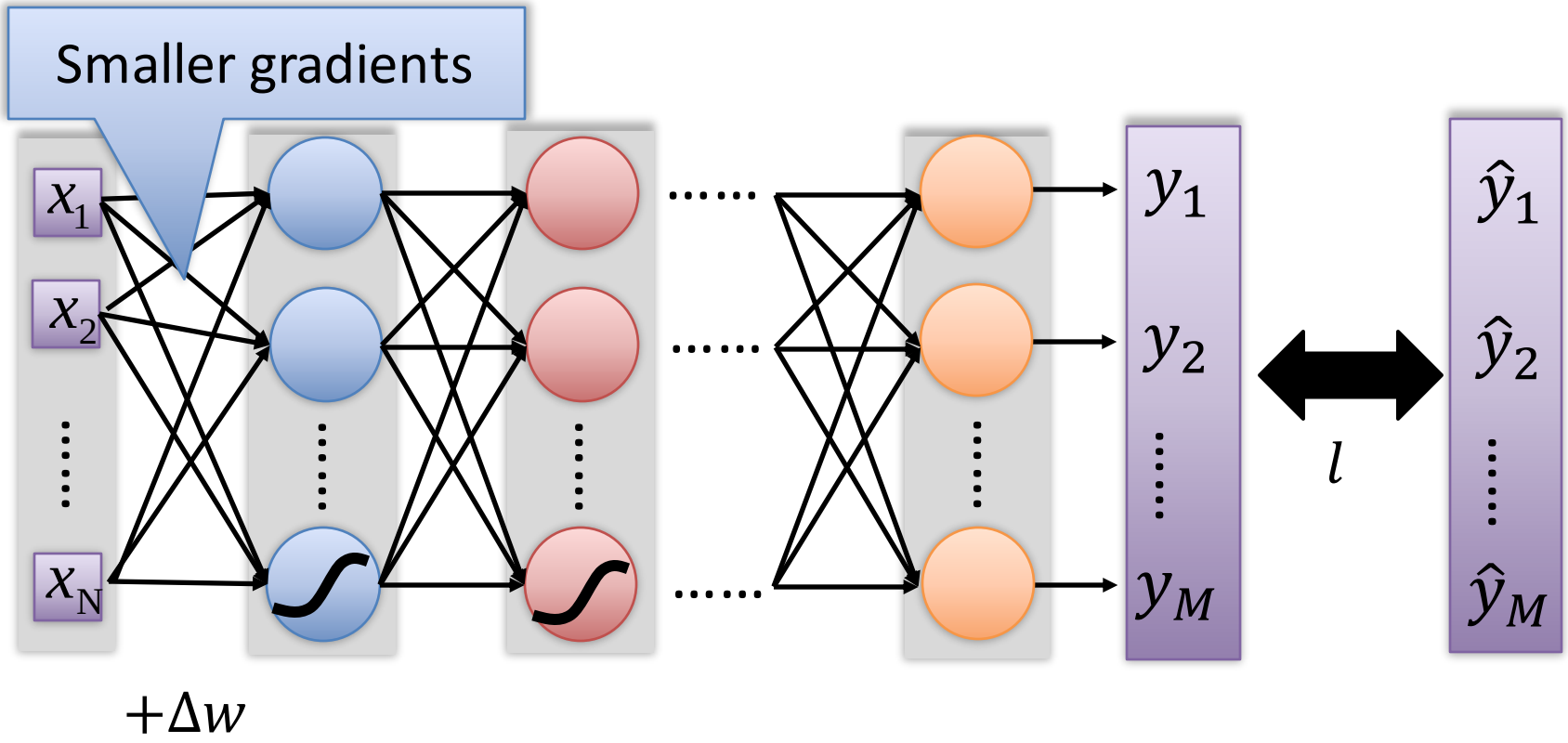
Intuitive way to compute the derivatives …

$$\frac{\partial l}{\partial w} =? \frac{\Delta l}{\Delta w}$$

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Vanishing Gradient Problem



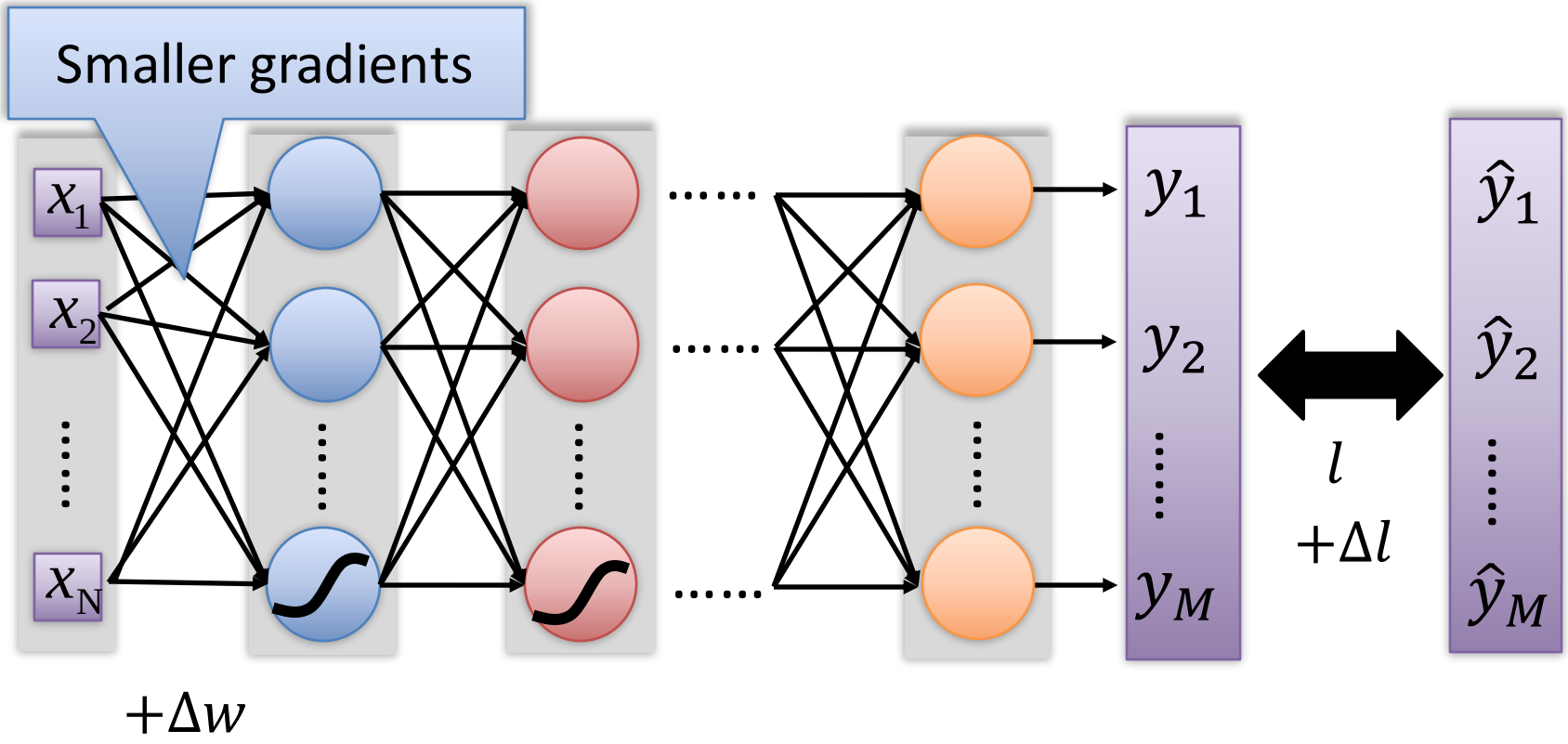
Intuitive way to compute the derivatives …

$$\frac{\partial l}{\partial w} =? \frac{\Delta l}{\Delta w}$$

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Vanishing Gradient Problem



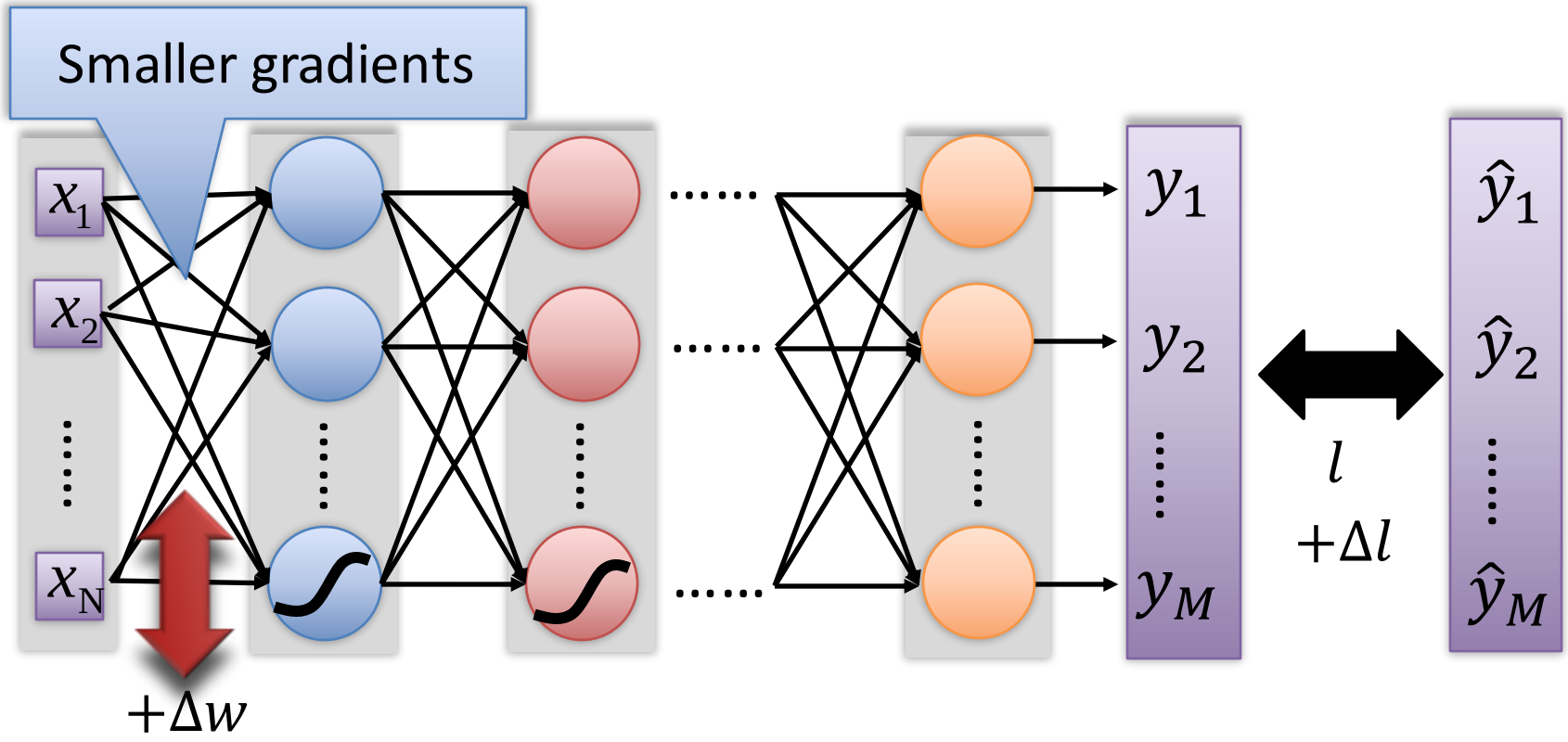
Intuitive way to compute the derivatives …

$$\frac{\partial l}{\partial w} =? \frac{\Delta l}{\Delta w}$$

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Vanishing Gradient Problem



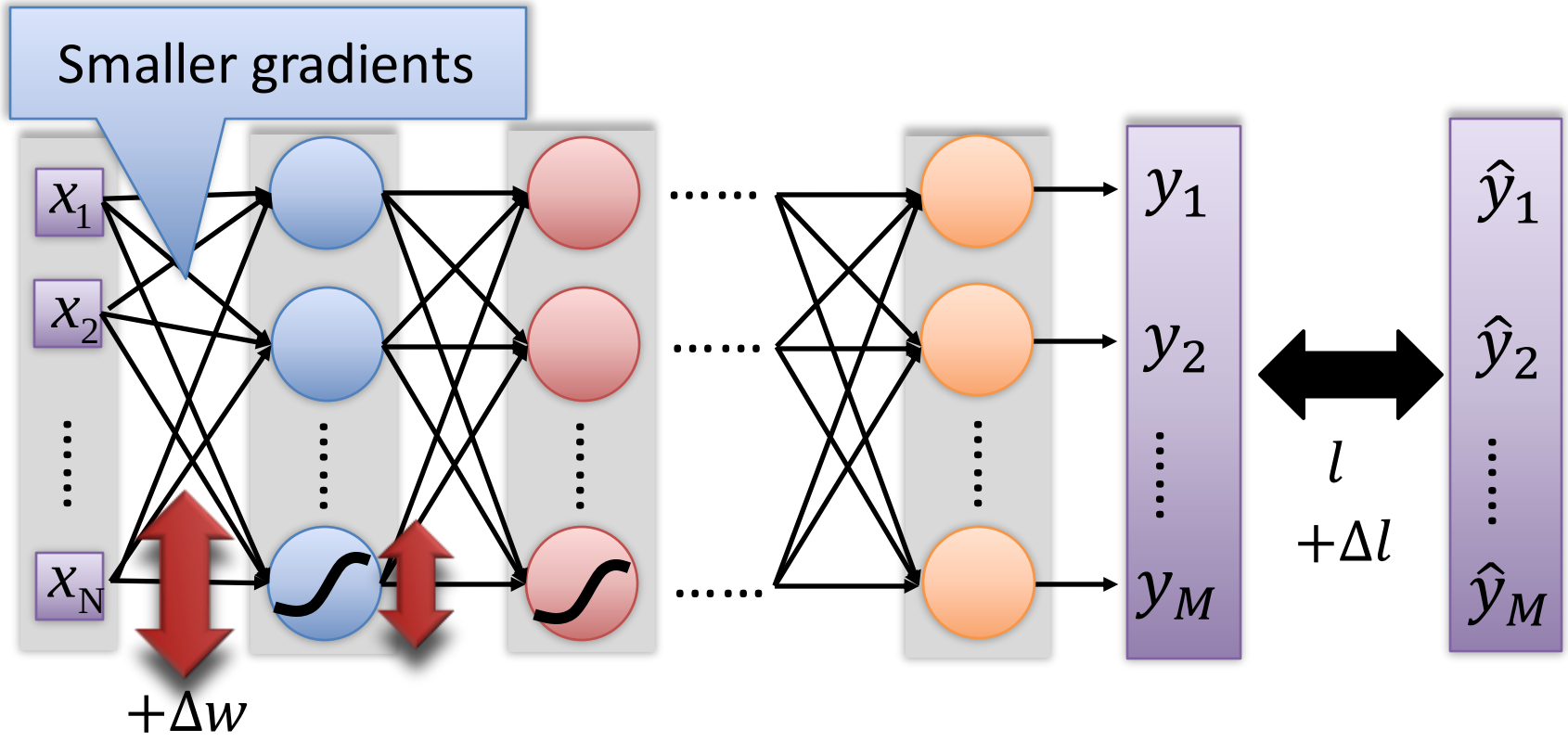
Intuitive way to compute the derivatives …

$$\frac{\partial l}{\partial w} =? \frac{\Delta l}{\Delta w}$$

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Vanishing Gradient Problem

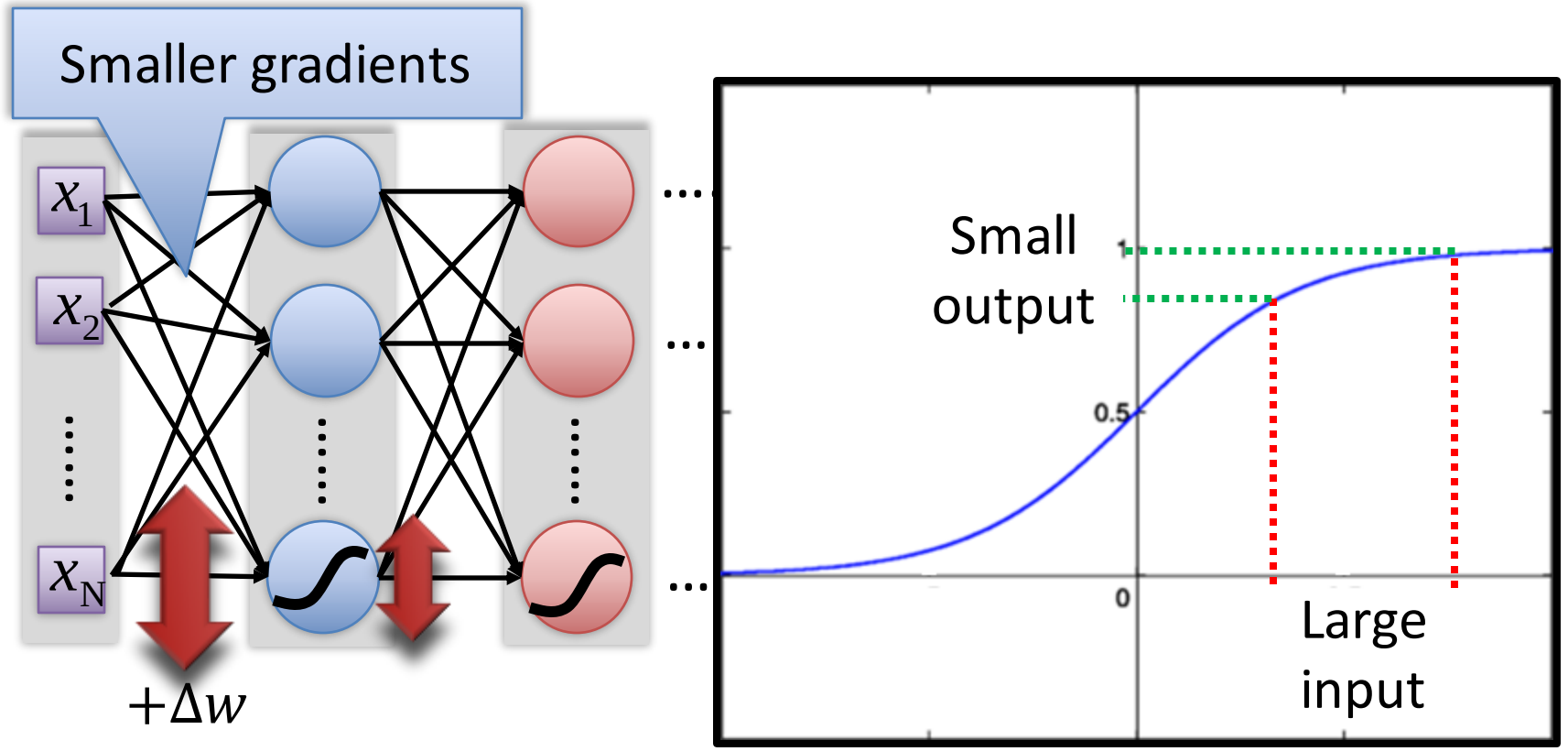


Intuitive way to compute the derivatives …

$$\frac{\partial l}{\partial w} =? \frac{\Delta l}{\Delta w}$$

UNIVERSITÉ CÔTE D'AZUR
univ-cotedazur.fr

Vanishing Gradient Problem

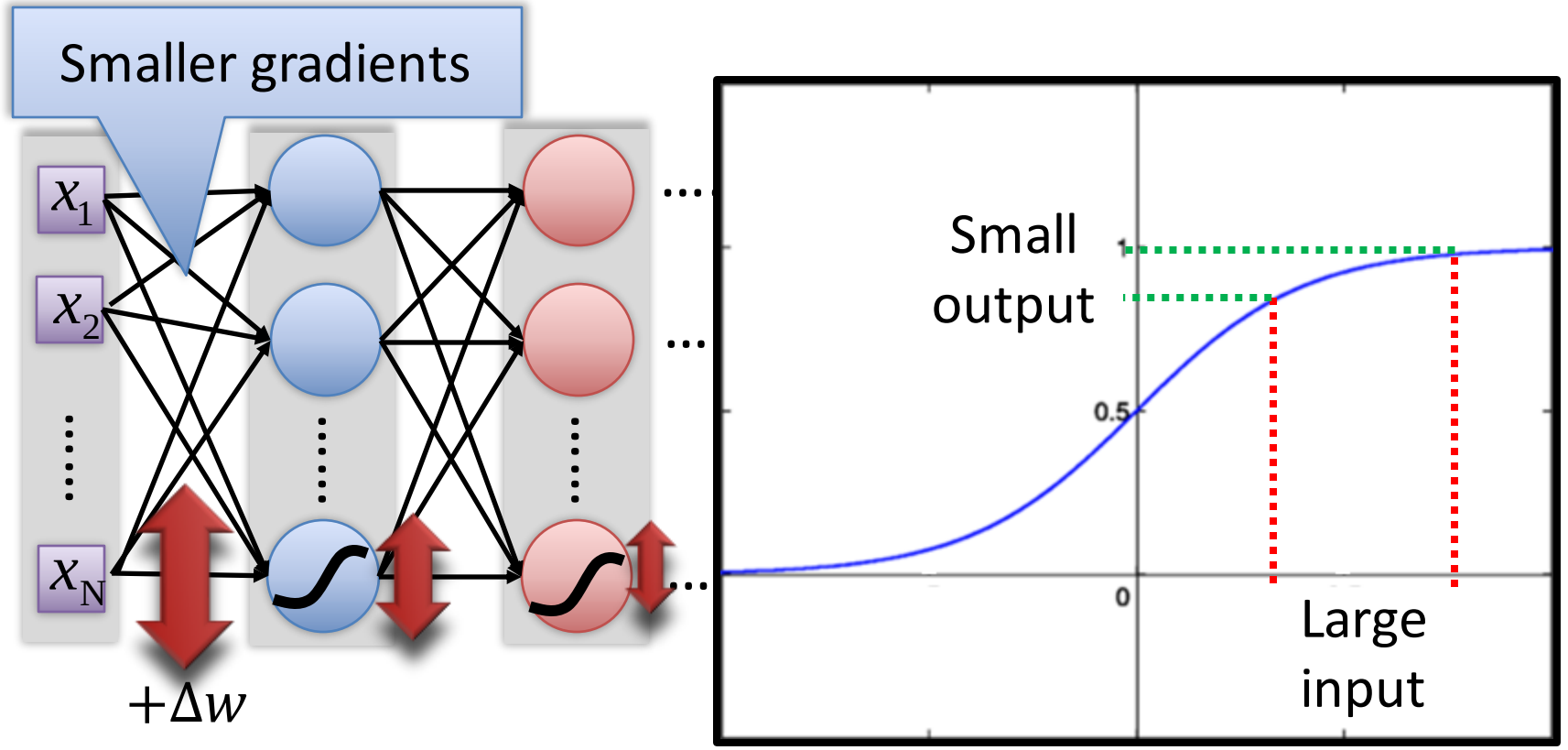


Intuitive way to compute the derivatives …

$$\frac{\partial l}{\partial w} =? \frac{\Delta l}{\Delta w}$$

UNIVERSITÉ CÔTE D'AZUR
univ-cotedazur.fr

Vanishing Gradient Problem



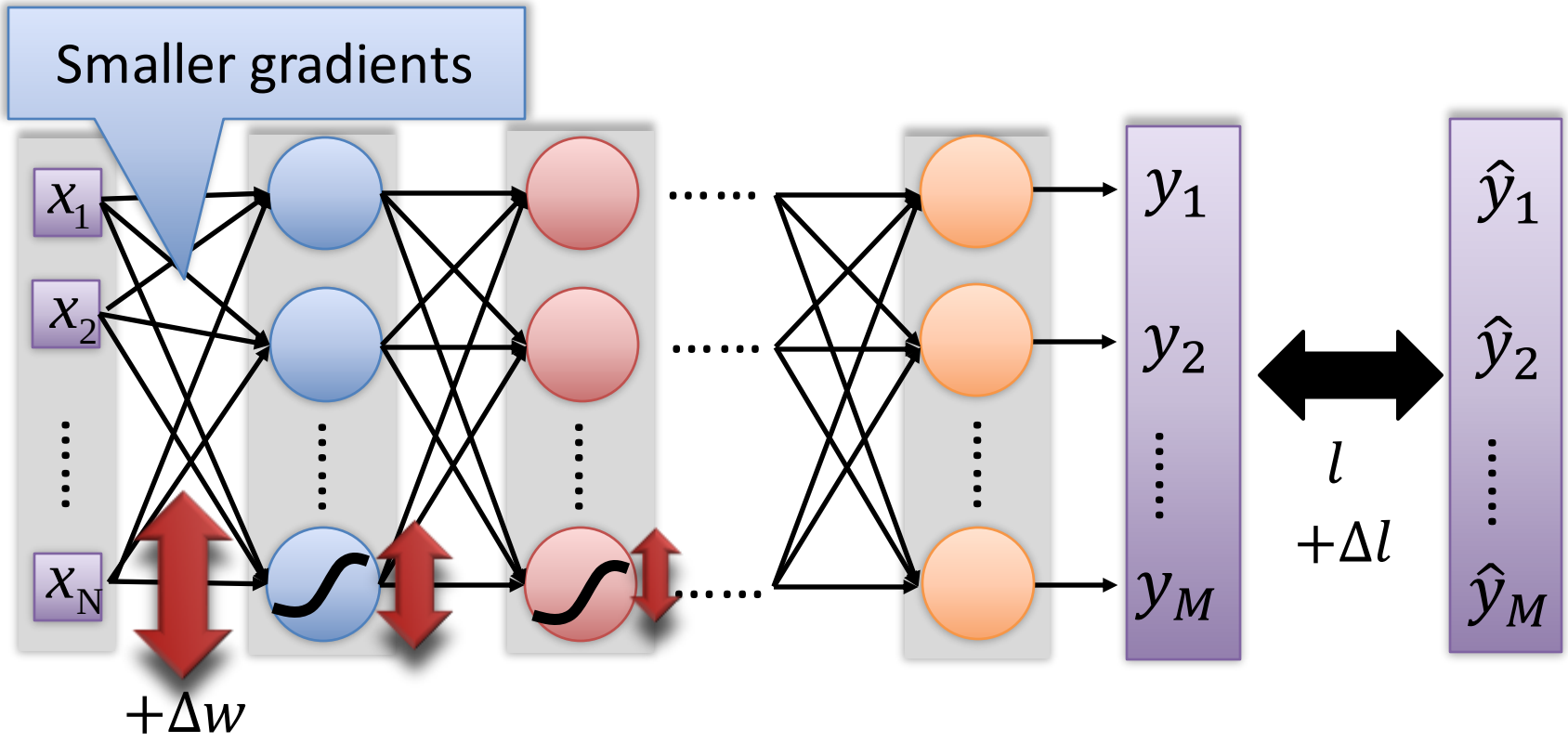
Intuitive way to compute the derivatives …

$$\frac{\partial l}{\partial w} =? \frac{\Delta l}{\Delta w}$$

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Vanishing Gradient Problem



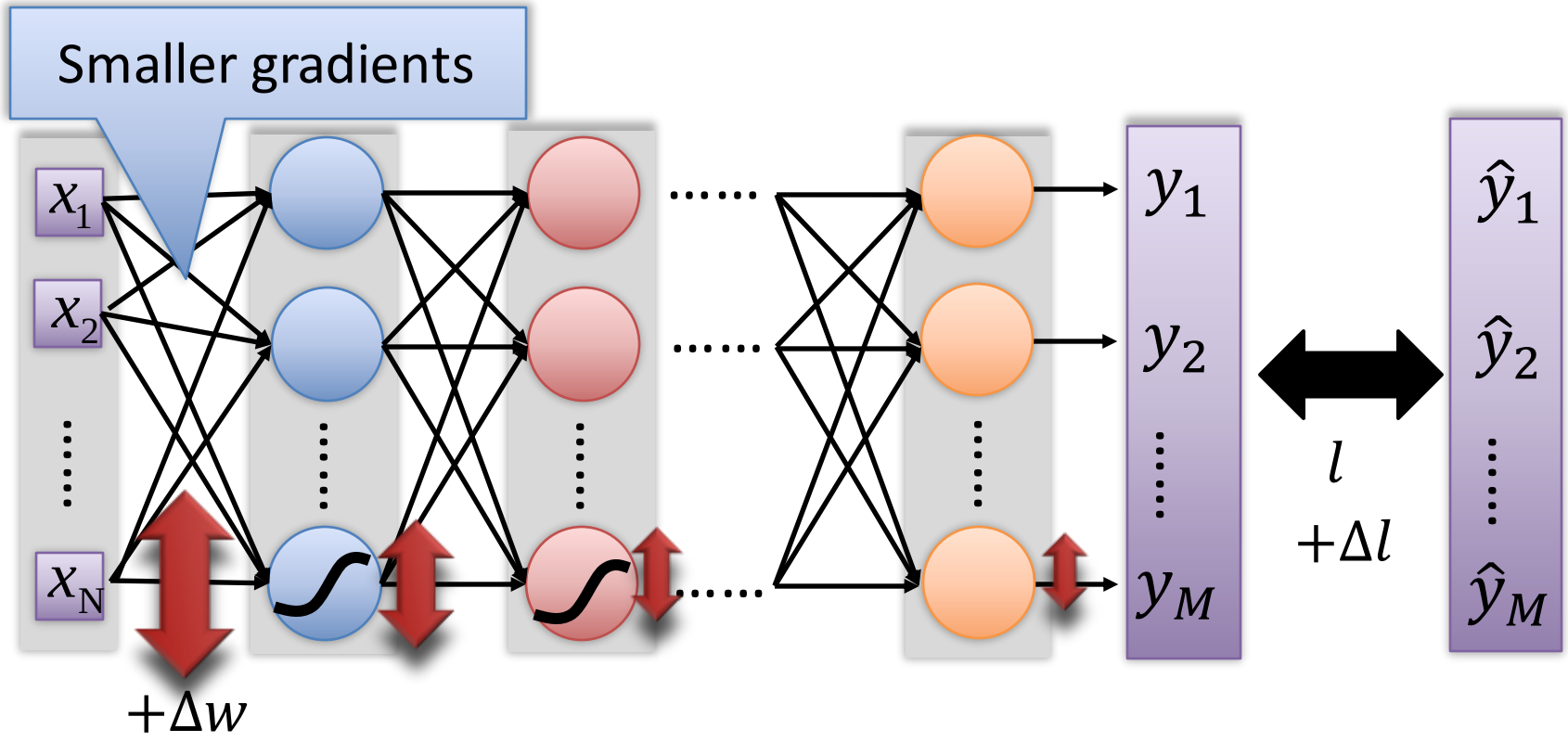
Intuitive way to compute the derivatives …

$$\frac{\partial l}{\partial w} =? \frac{\Delta l}{\Delta w}$$

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Vanishing Gradient Problem



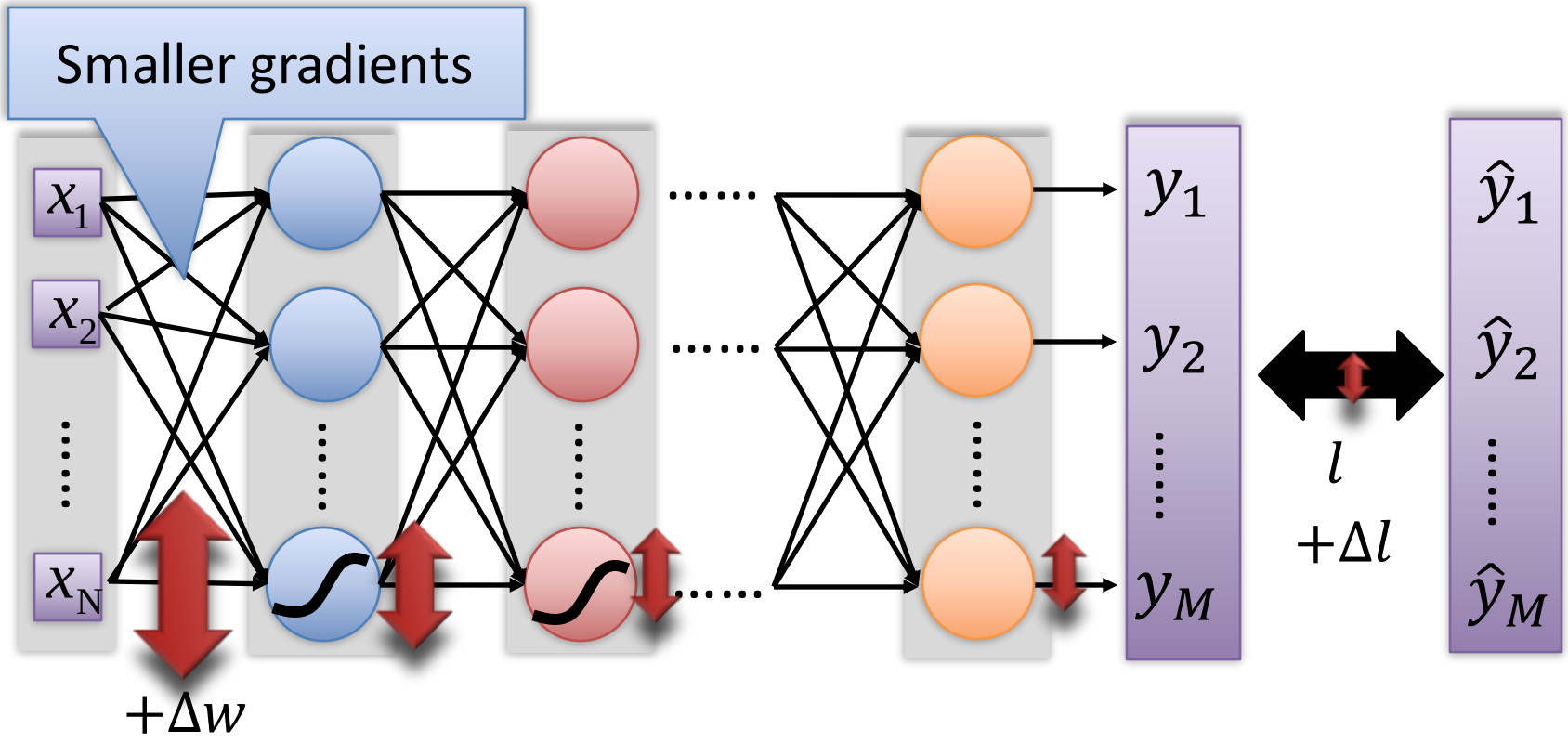
Intuitive way to compute the derivatives …

$$\frac{\partial l}{\partial w} =? \frac{\Delta l}{\Delta w}$$

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Vanishing Gradient Problem



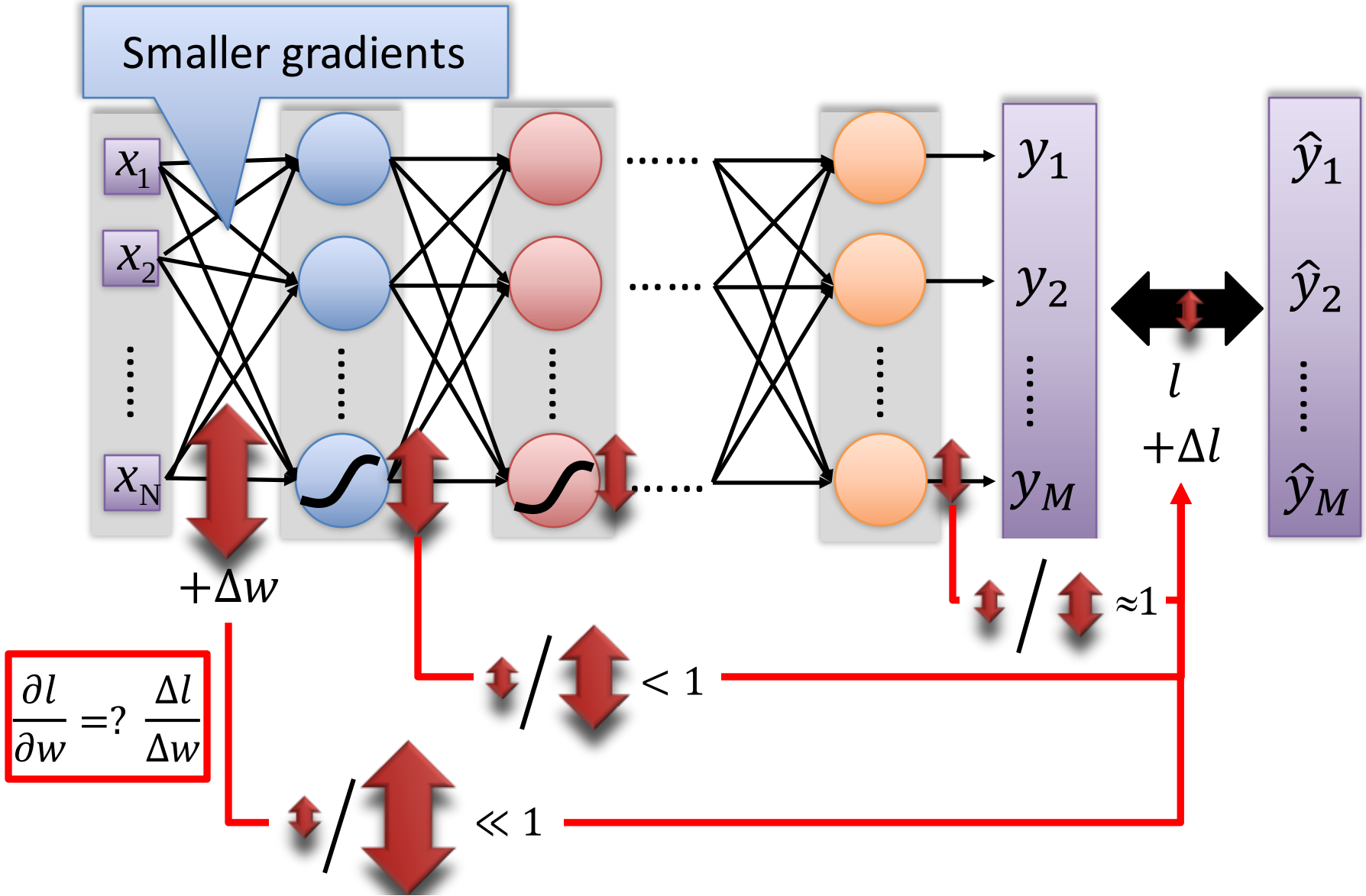
Intuitive way to compute the derivatives …

$$\frac{\partial l}{\partial w} =? \frac{\Delta l}{\Delta w}$$

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

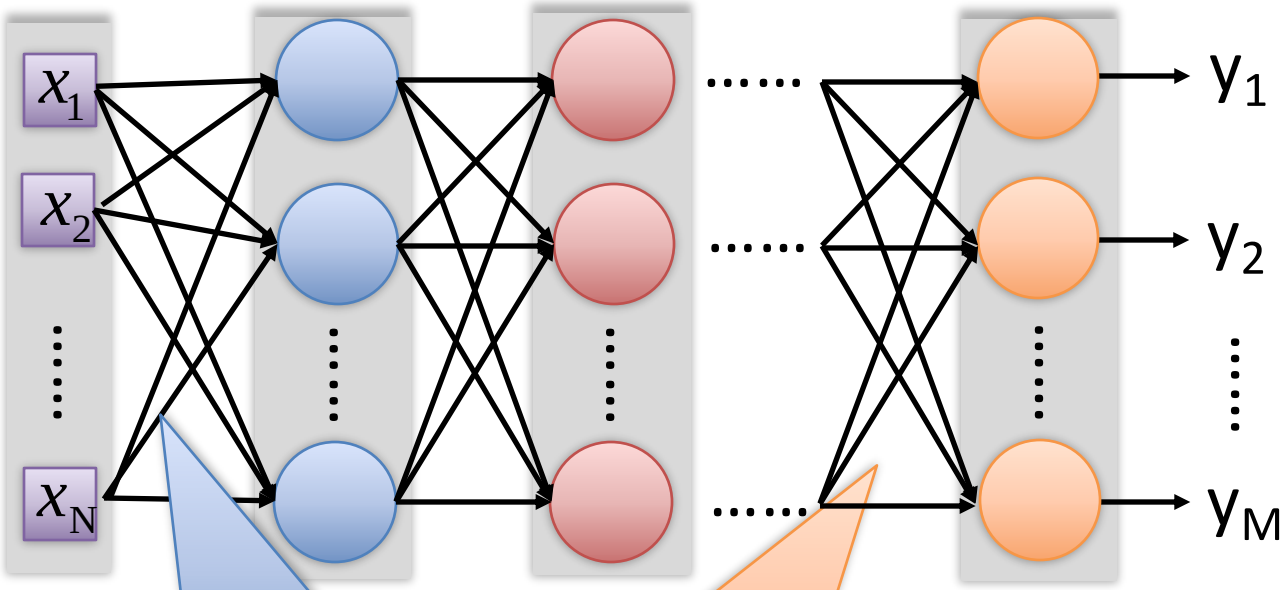
Vanishing Gradient Problem



UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Vanishing Gradient Problem



Smaller gradients

Learn very slow

Almost random

Larger gradients

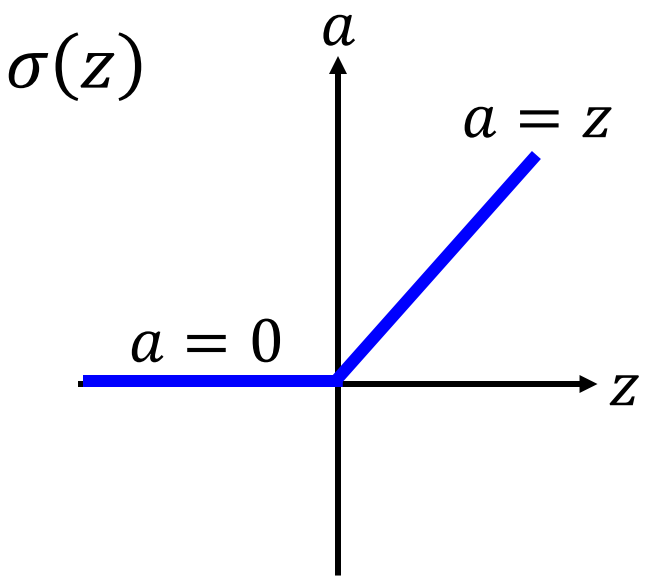
Learn very fast

Already converge

based on random!?

Vanishing Gradient Problem

- ***Rectified Linear Unit (ReLU)***



[Xavier Glorot, AISTATS'11]
[Andrew L. Maas, ICML'13]
[Kaiming He, arXiv'15]

Reason:

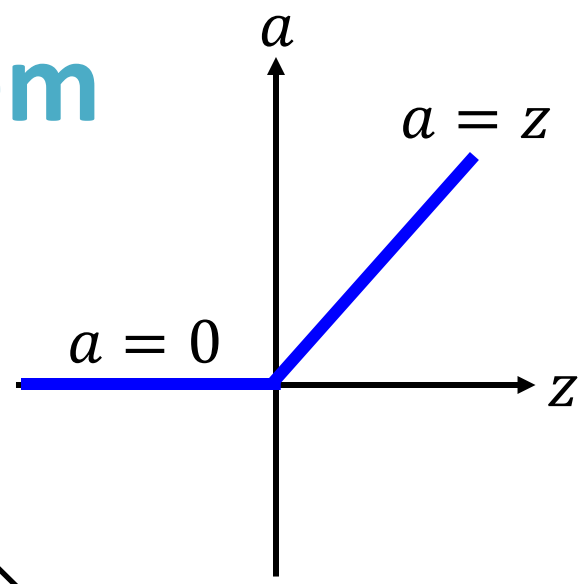
1. Fast to compute
2. Biological reason
3. Infinite sigmoid with different biases
4. Vanishing gradient problem

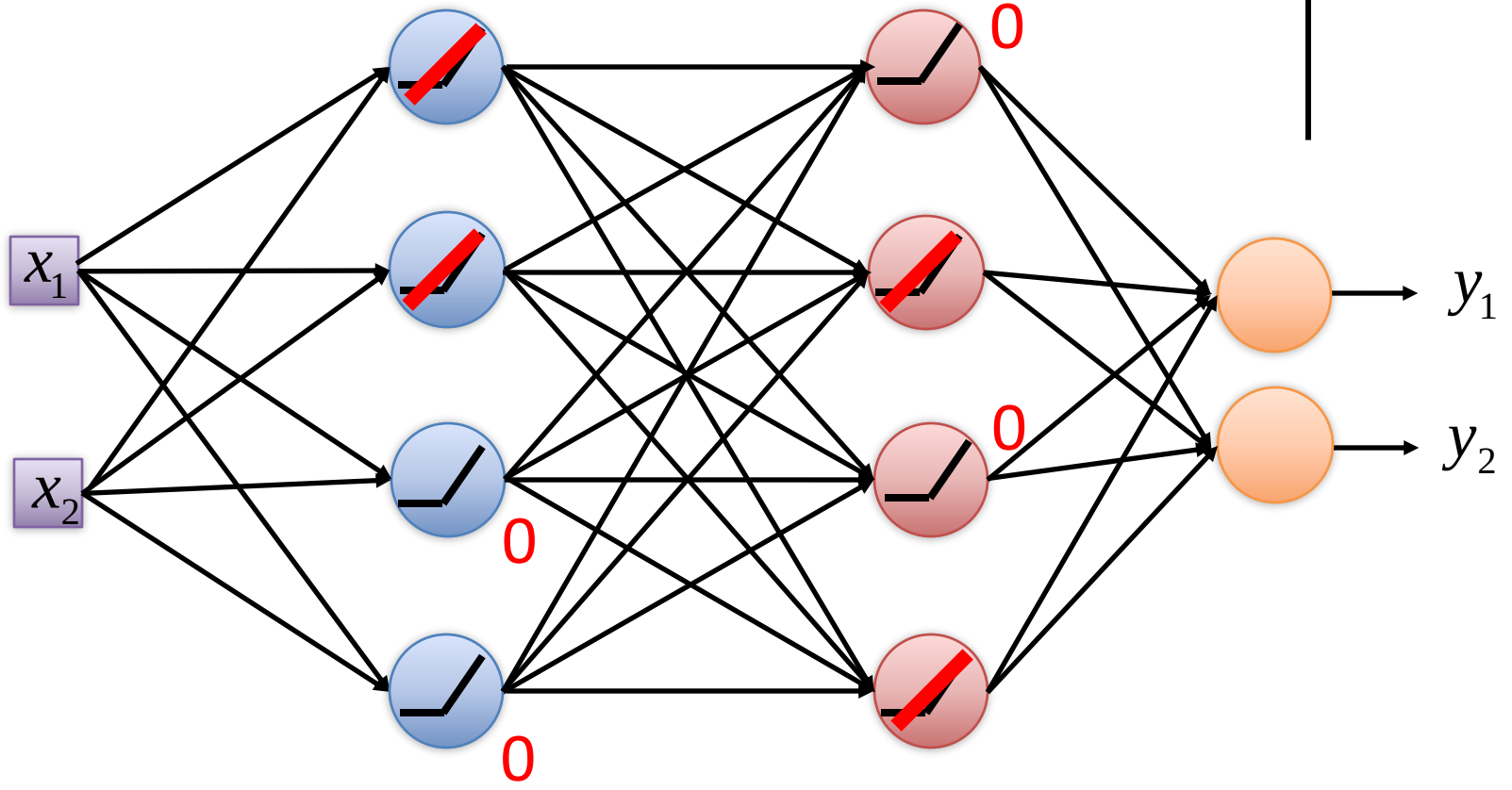
UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Vanishing Gradient Problem

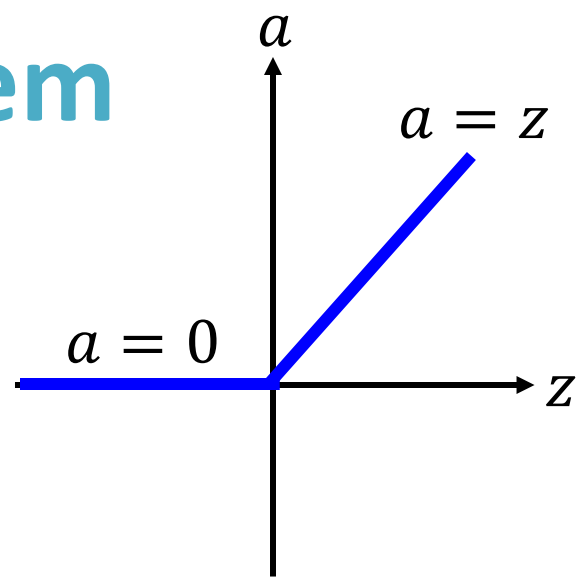
- ***Rectified Linear Unit (ReLU)***

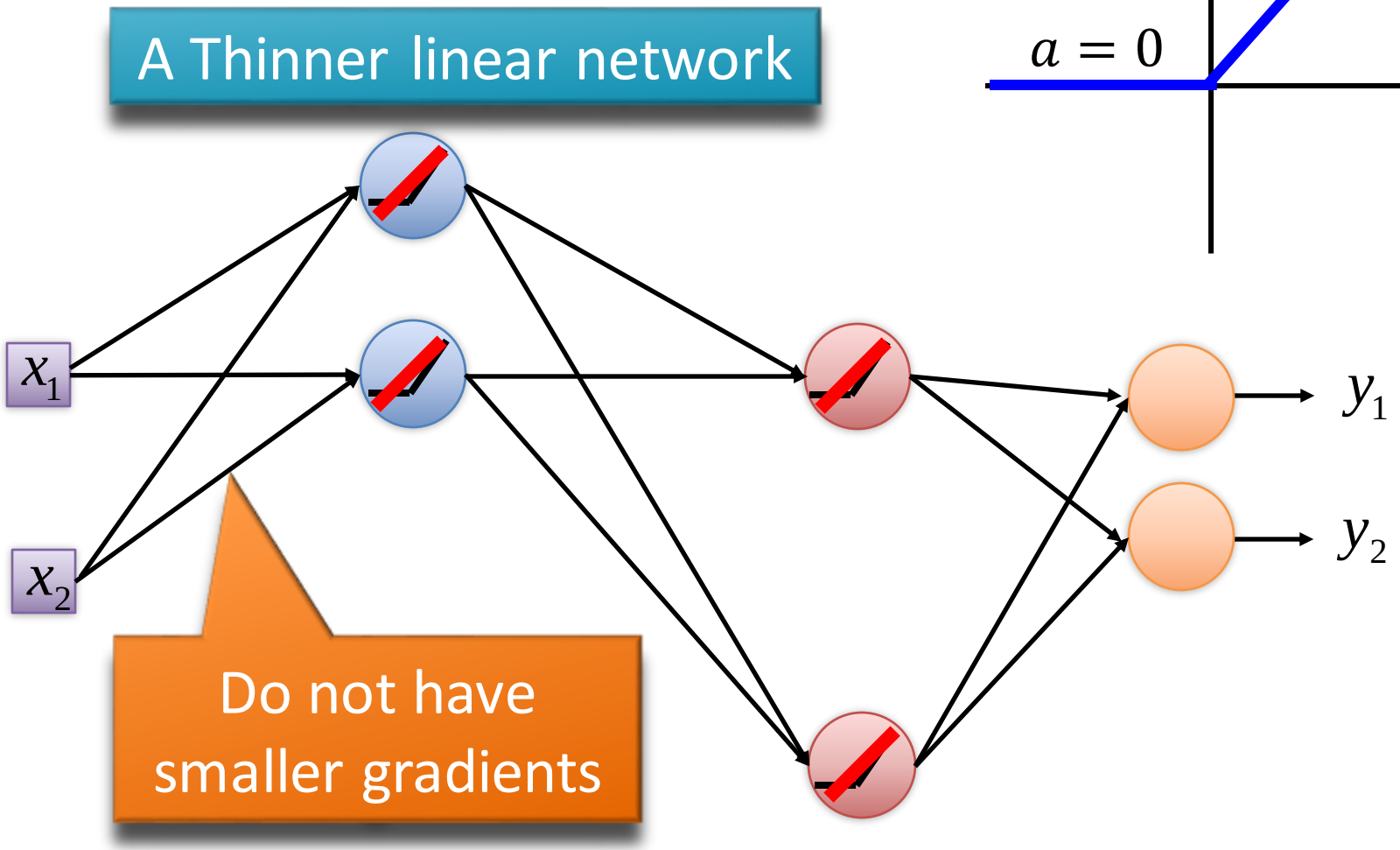




Vanishing Gradient Problem

- ***Rectified Linear Unit (ReLU)***

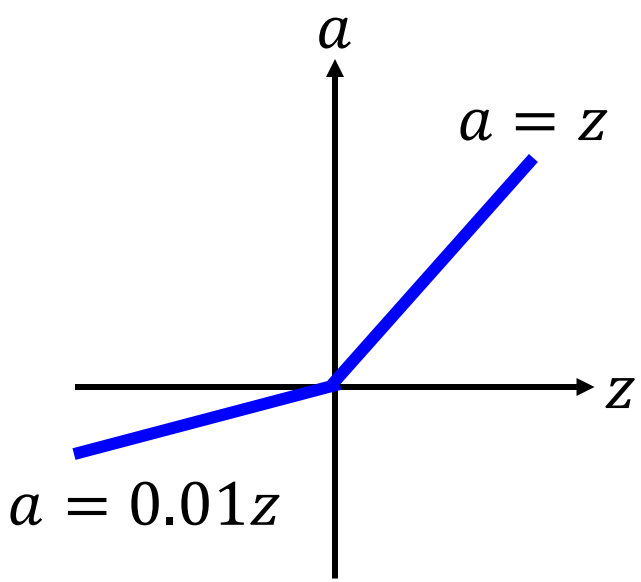




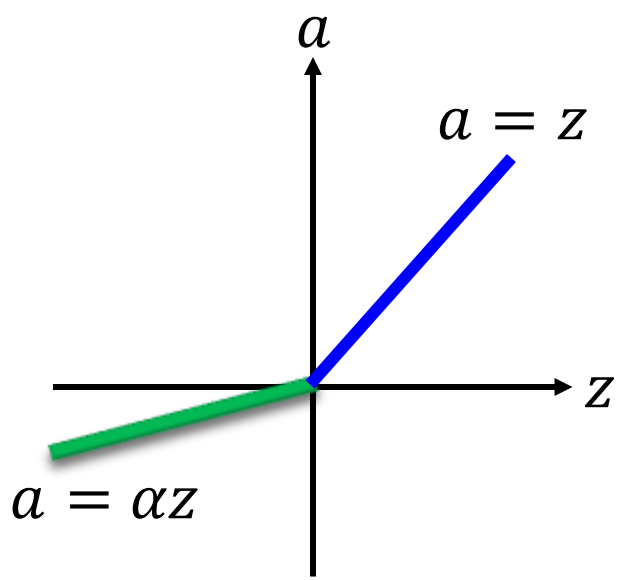
Vanishing Gradient Problem

- ***ReLU - Variant***

Leaky ReLU



Parametric ReLU



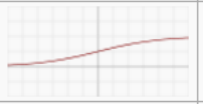
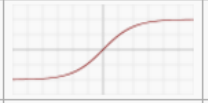
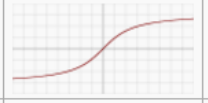
α also learned by gradient descent

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Vanishing Gradient Problem

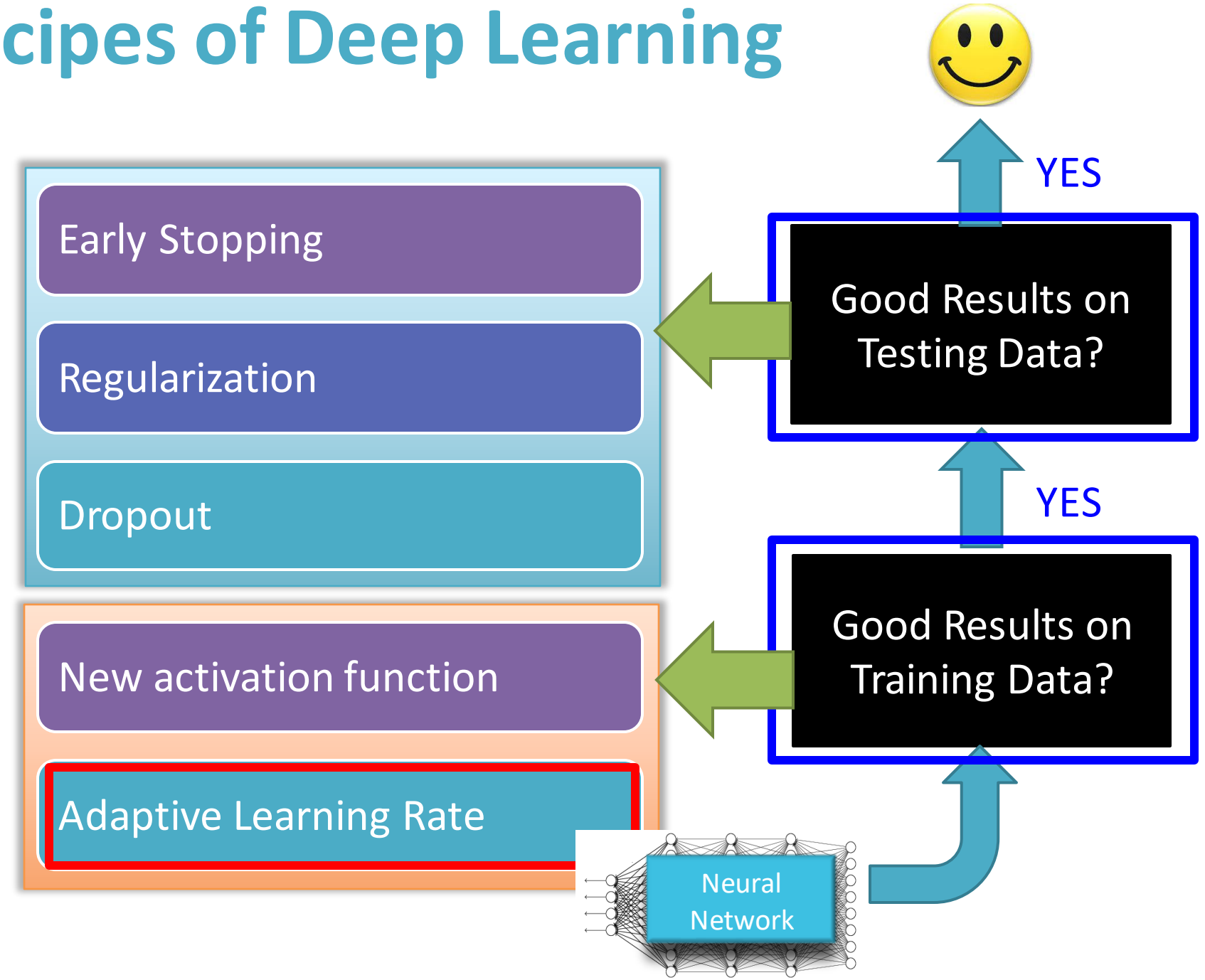
- ***Activation functions***

Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1-f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2+1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU) [2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU) [3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1+e^x)$	$f'(x) = \frac{1}{1+e^{-x}}$

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

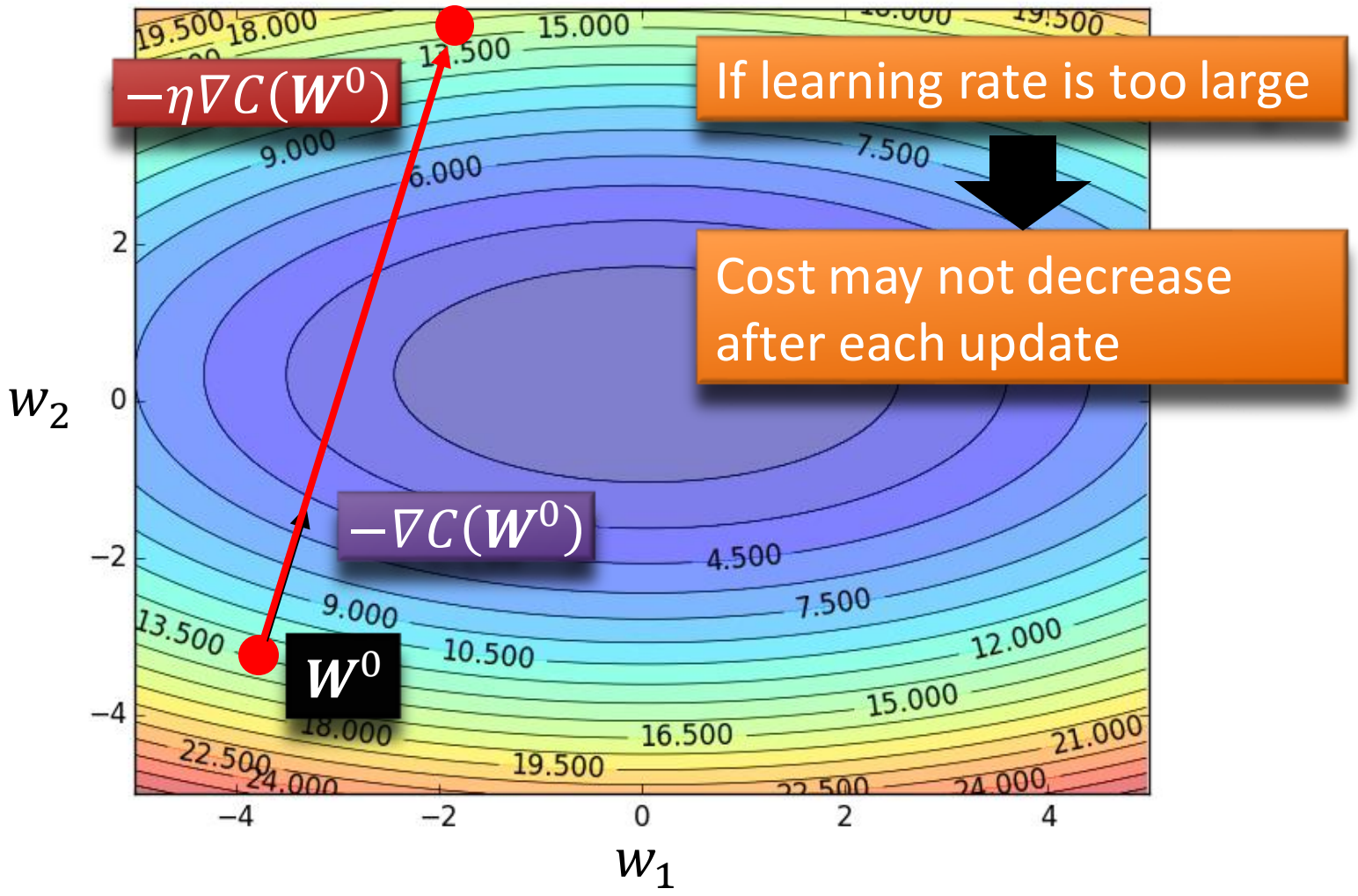
Recipes of Deep Learning



Recipes of Deep Learning

- ***Adaptive Learning Rate***

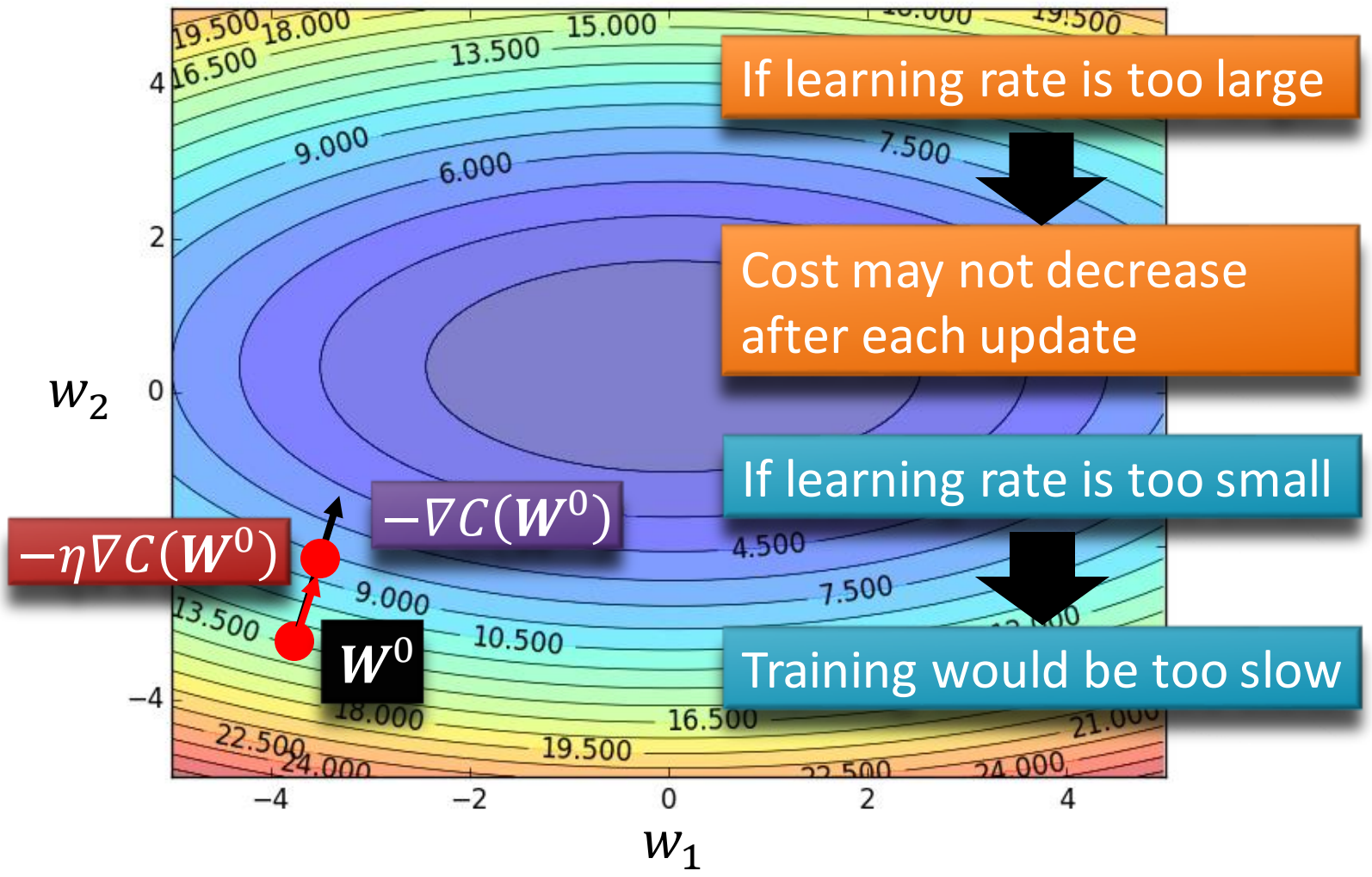
Set the learning rate η carefully



Recipes of Deep Learning

- ***Adaptive Learning Rate***

Can we give different parameters different learning rates?



If learning rate is too large

Cost may not decrease after each update

If learning rate is too small

Training would be too slow

Learning Rates

- Popular & Simple Idea: Reduce the learning rate by some factor every few epochs.
  - At the beginning, we are far from the destination, so we use larger learning rate
  - After several epochs, we are close to the destination, so we reduce the learning rate
  - E.g. 1/t decay: $\eta^t = \eta/\sqrt{t+1}$
- Learning rate cannot be one-size-fits-all
  - Giving different parameters different learning rates

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Adaptative Learning Rates

- ***Adagrad***

Original Gradient Descent

$$\boldsymbol{W}^t \leftarrow \boldsymbol{W}^{t-1} - \eta \nabla C(\boldsymbol{W}^{t-1})$$

Each parameter w are considered separately

$$w^{t+1} \leftarrow w^t - \eta_w g^t \qquad g^t = \frac{\partial C(\boldsymbol{W}^t)}{\partial w}$$

Parameter dependent learning rate

$$\eta_w = \frac{\eta}{\sqrt{\sum_{i=0}^{t}(g^i)^2}}$$

constant

g^i is $\partial L / \partial w$ obtained at the i-th update

Summation of the square of the previous derivatives

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Adaptative Learning Rates

$$\eta_w = \frac{\eta}{\sqrt{\sum_{i=0}^{t}(g^i)^2}}$$

- ***Adagrad***

	g^0
w_1	0.1

	g^0
w_2	20.0

Learning rate:

$$\frac{\eta}{\sqrt{0.1^2}} = \frac{\eta}{0.1} \qquad \frac{\eta}{\sqrt{0.1^2 + 0.2^2}} = \frac{\eta}{0.22}$$

Learning rate:

$$\frac{\eta}{\sqrt{20^2}} = \frac{\eta}{20} \qquad \frac{\eta}{\sqrt{20^2 + 10^2}} = \frac{\eta}{22}$$

Observation: 1. Learning rate is smaller and smaller for all parameters

2. Smaller derivatives, larger learning rate, and vice versa

Why?

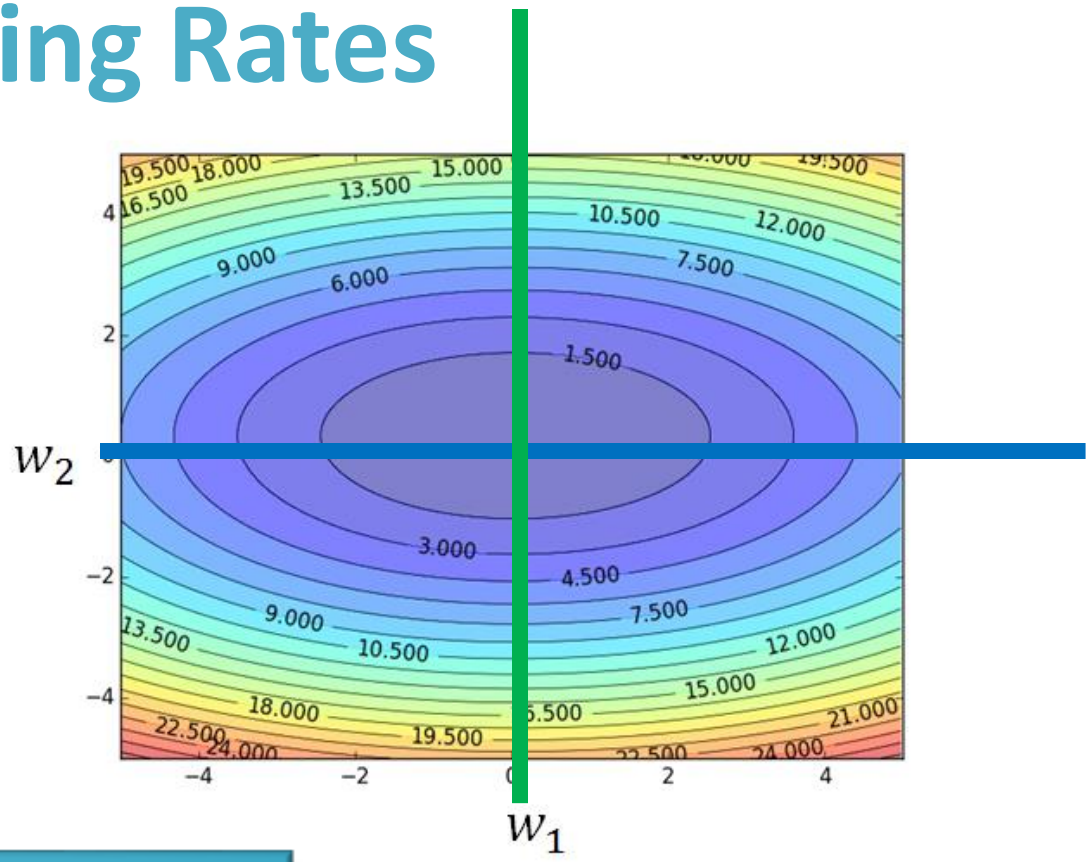
UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Adaptative Learning Rates

Larger derivatives

Smaller Learning Rate



Smaller Derivatives

Larger Learning Rate

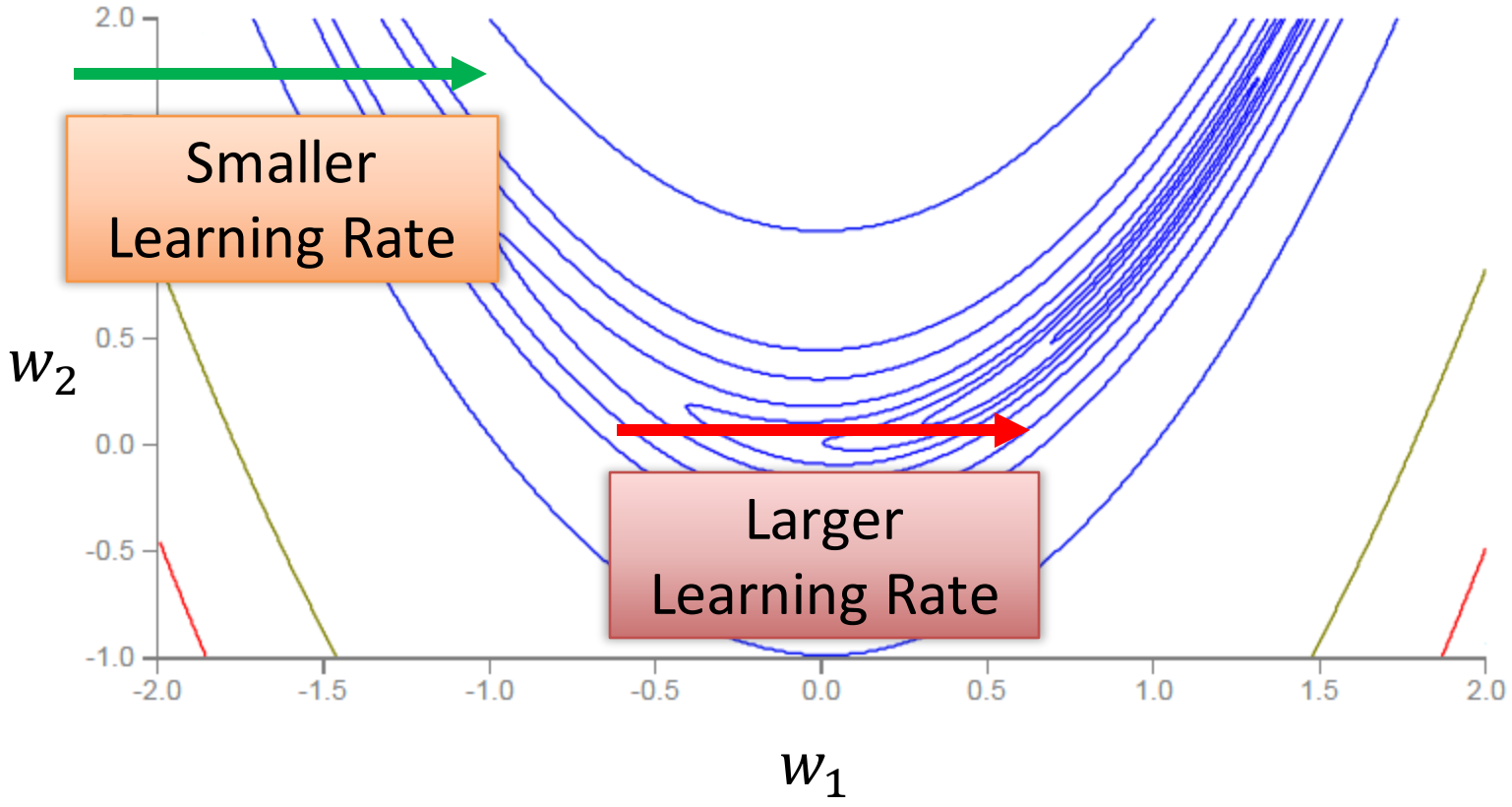
2. Smaller derivatives, larger learning rate, and vice versa

Why?

Adaptative Learning Rates

- ***RMSProp***

Error Surface can be very complex when training NN.



Adaptative Learning Rates

- ***RMSProp***

$$w^1 \leftarrow w^0 - \frac{\eta}{\sigma^0} g^0 \qquad \sigma^0 = g^0$$

$$w^2 \leftarrow w^1 - \frac{\eta}{\sigma^1} g^1 \qquad \sigma^1 = \sqrt{\alpha(\sigma^0)^2 + (1-\alpha)(g^1)^2}$$

$$w^3 \leftarrow w^2 - \frac{\eta}{\sigma^2} g^2 \qquad \sigma^2 = \sqrt{\alpha(\sigma^1)^2 + (1-\alpha)(g^2)^2}$$

$$\vdots$$

$$w^{t+1} \leftarrow w^t - \frac{\eta}{\sigma^t} g^t \qquad \sigma^t = \sqrt{\alpha(\sigma^{t-1})^2 + (1-\alpha)(g^t)^2}$$

Root Mean Square of the gradients with previous gradients being decayed

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Adaptative Learning Rates

- ***Adam*** RMSProp + Momentum

Algorithm 1: *Adam*, our proposed algorithm for stochastic optimization. See section 2 for details, and for a slightly more efficient (but less clear) order of computation. g_t^2 indicates the elementwise square $g_t \odot g_t$. Good default settings for the tested machine learning problems are $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. All operations on vectors are element-wise. With β_1^t and β_2^t we denote β_1 and β_2 to the power t.

Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
$m_0 \leftarrow 0$ (Initialize 1st moment vector) → for momentum
$v_0 \leftarrow 0$ (Initialize 2nd moment vector) → for RMSprop
$t \leftarrow 0$ (Initialize timestep)
while θ_t not converged **do**
$t \leftarrow t + 1$
$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
$m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
$v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
$\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
$\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
$\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
end while
return θ_t (Resulting parameters)

Adaptative Learning Rates

- ***Not the whole story …***

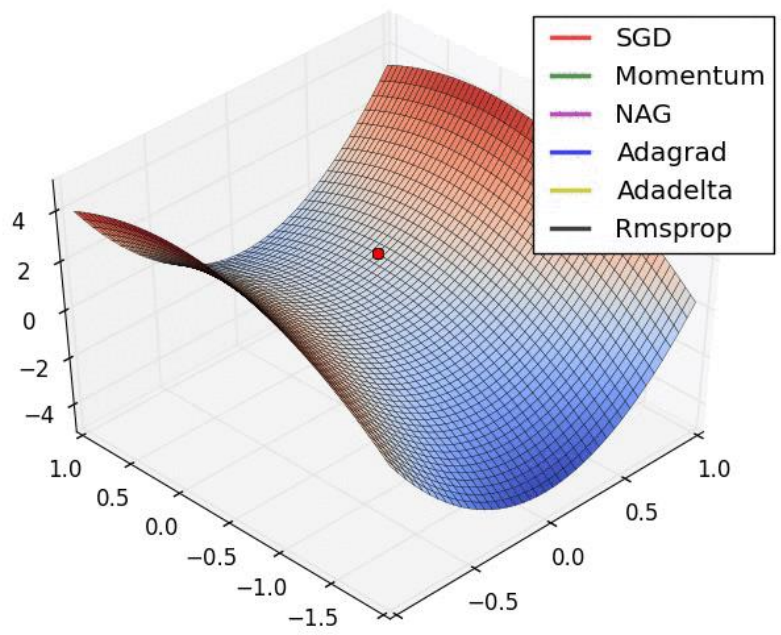
- Adagrad [John Duchi, JMLR'11]
- RMSprop
  - https://www.youtube.com/watch?v=O3sxAc4hxZU
- Adadelta [Matthew D. Zeiler, arXiv'12]
- Adam [Diederik P. Kingma, ICLR'15] = RMSprop + Momentum
- AdaSecant [Caglar Gulcehre, arXiv'14]
- "No more pesky learning rates" [Tom Schaul, arXiv'12]
- Nadam
  - http://cs229.stanford.edu/proj2015/054_report.pdf

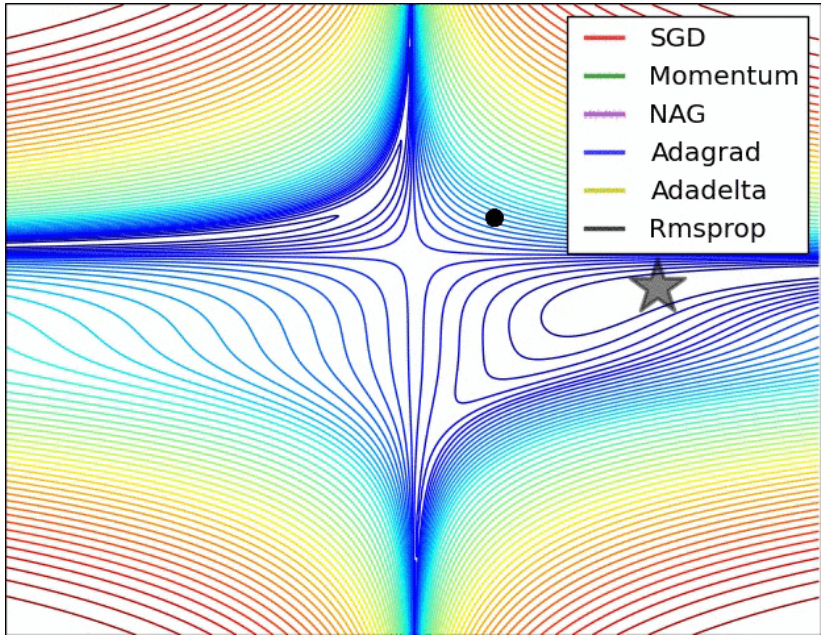
UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Adaptative Learning Rates

- ***Not the whole story …***

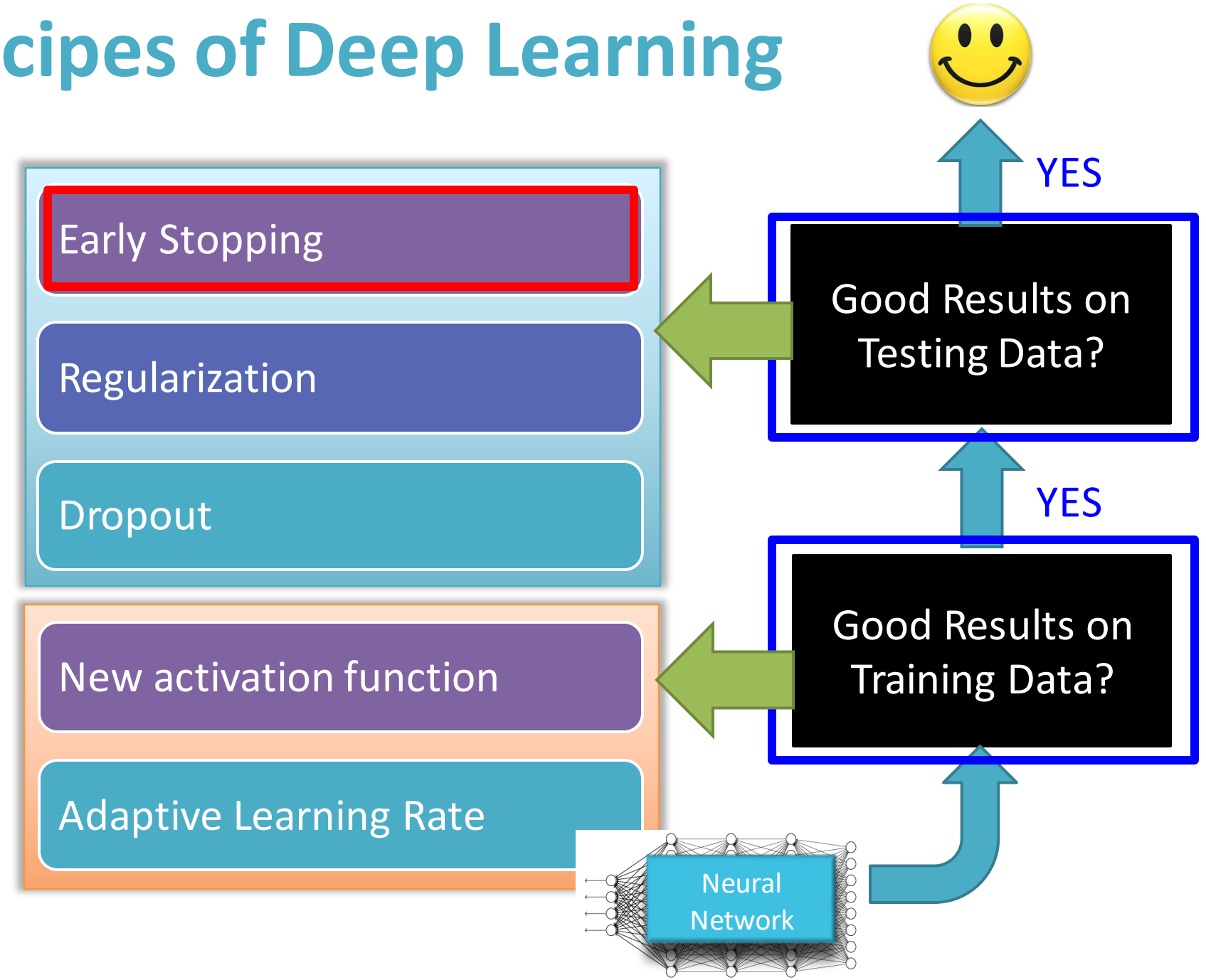




UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Recipes of Deep Learning



Recipes of Deep Learning

- ***Panacea for Overfitting***

- Have more training data
- ***Create*** more training data (?)

Handwriting recognition:

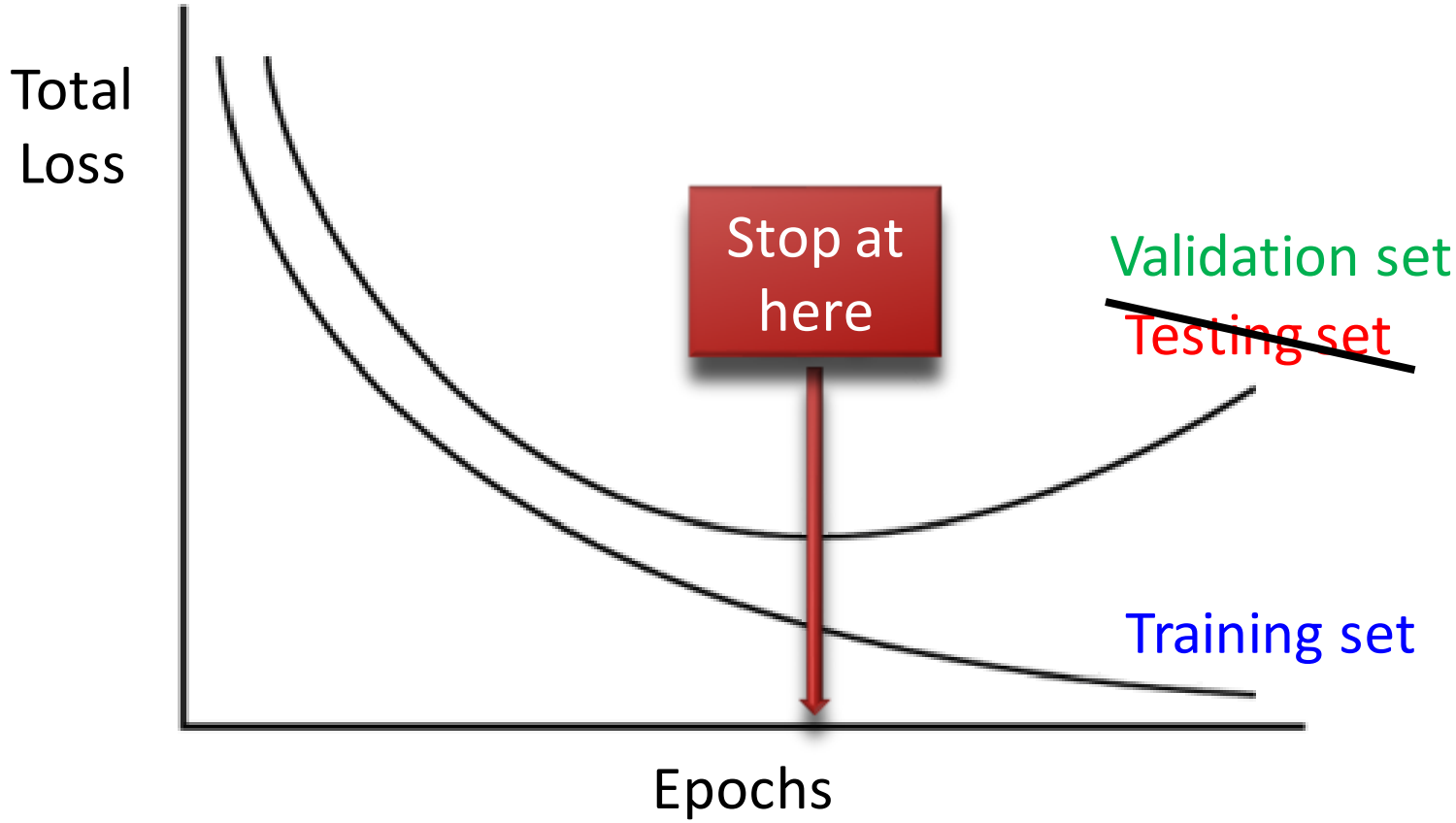
Original Training Data:

Created Training Data:

Shift 15°

Recipes of Deep Learning

- ***Early stopping***

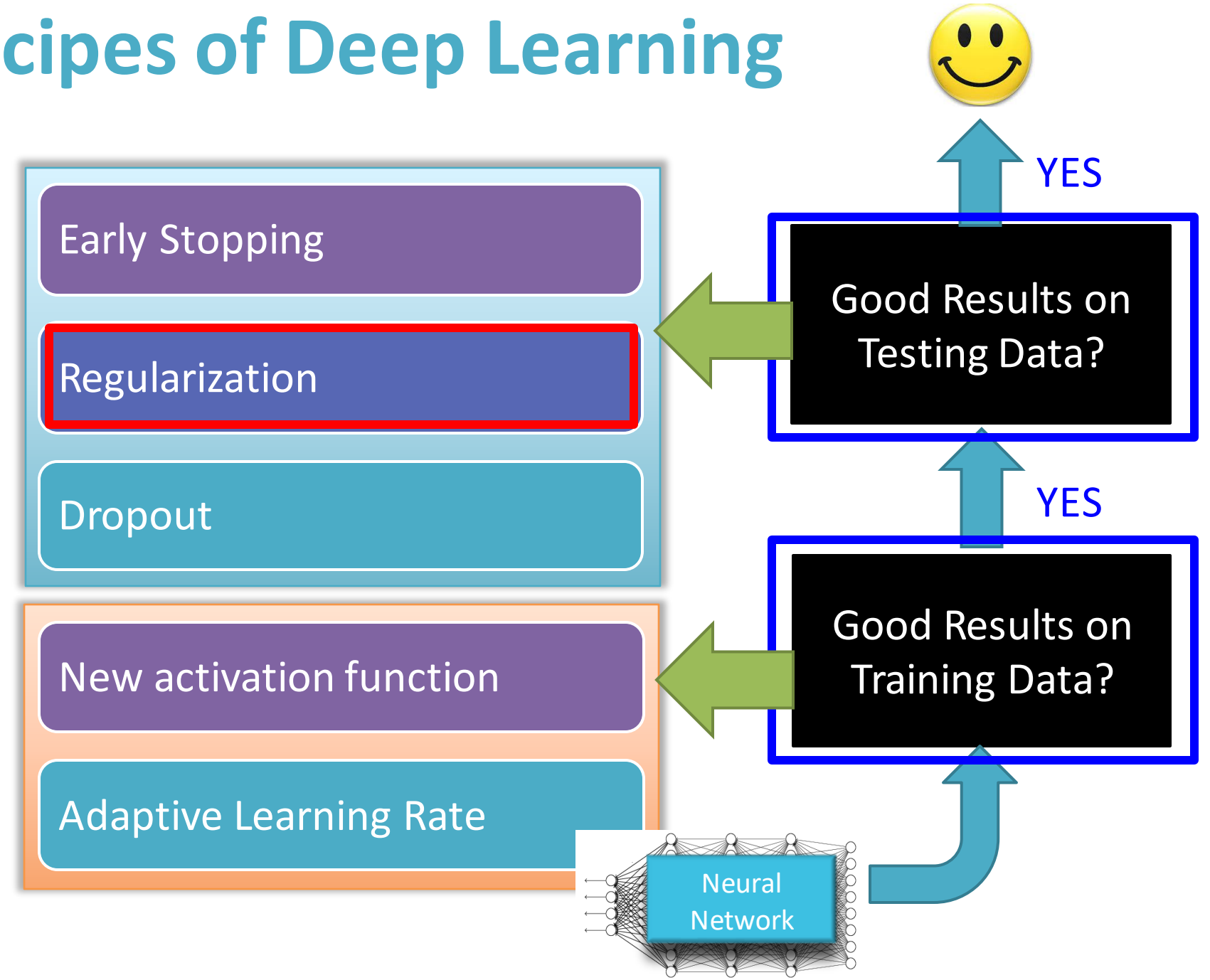


Keras: http://keras.io/getting-started/faq/#how-can-i-interrupt-training-when-the-validation-loss-isnt-decreasing-anymore

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Recipes of Deep Learning



Recipes of Deep Learning

- ***Regularization***
- New loss function to be minimized
  - Find a set of weight not only minimizing original cost but also close to zero

$$\mathrm{L}'(W) = L(W) + \lambda \frac{1}{2} \|W\|_2$$ → Regularization term

$$W = \{w_1, w_2, \ldots\}$$

L2 regularization also known as ridge, or weight decay
(***The most widely used regularization method***):

$$\|W\|_2 = (w_1)^2 + (w_2)^2 + \ldots$$

(usually not consider biases)

Regularization

L2 regularization:

$$\|W\|_2 = (w_1)^2 + (w_2)^2 + \dots$$

- New loss function to be minimized

$$\mathbf{L}'(W) = L(W) + \lambda \frac{1}{2}\|W\|_2 \quad \text{Gradient:} \quad \frac{\partial \mathrm{L}'}{\partial w} = \frac{\partial \mathrm{L}}{\partial w} + \lambda w$$

Update:

$$w^{t+1} \rightarrow w^t - \eta \frac{\partial \mathrm{L}'}{\partial w} = w^t - \eta\left(\frac{\partial \mathrm{L}}{\partial w} + \lambda w^t\right)$$

$$= \underline{(1-\eta\lambda)w^t} - \eta \underline{\frac{\partial \mathrm{L}}{\partial w}}$$

Weight Decay

Closer to zero

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Regularization

- New loss function to be minimized
  - Find a set of weight not only minimizing original cost but also close to zero

$$\mathrm{L}'(W) = L(W) + \lambda \frac{1}{2} |W| \rightarrow \text{Regularization term}$$

$$W = \{w_1, w_2, \ldots\}$$

L1 regularization also known as Lasso, produce sparse results:

$$|W| = |w_1| + |w_2| + \ldots$$

(usually not consider biases)

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Regularization

L1 regularization:

$$\|W\|_1 = |w_1| + |w_2| + \dots$$

- New loss function to be minimized

$$\mathbf{L}'(W) = L(W) + \lambda \frac{1}{2} \|W\|_1 \qquad \frac{\partial \mathrm{L}'}{\partial w} = \frac{\partial \mathrm{L}}{\partial w} + \lambda \operatorname{sgn}(w)$$

Update:

$$w^{t+1} \rightarrow w^t - \eta \frac{\partial \mathrm{L}'}{\partial w} = w^t - \eta \left(\frac{\partial \mathrm{L}}{\partial w} + \lambda \operatorname{sgn}(w^t) \right)$$

$$= w^t - \eta \frac{\partial \mathrm{L}}{\partial w} - \underline{\eta \lambda \operatorname{sgn}(w^t)}$$

Regularization

- New loss function to be minimized
  - Find a set of weight not only minimizing original cost but also close to zero

$$\mathrm{L}'(W) = L(W) + \lambda_1 \frac{1}{2}|W| + \lambda_2 \frac{1}{2}\|W\|_2$$

$$W = \{w_1, w_2, \ldots\}$$

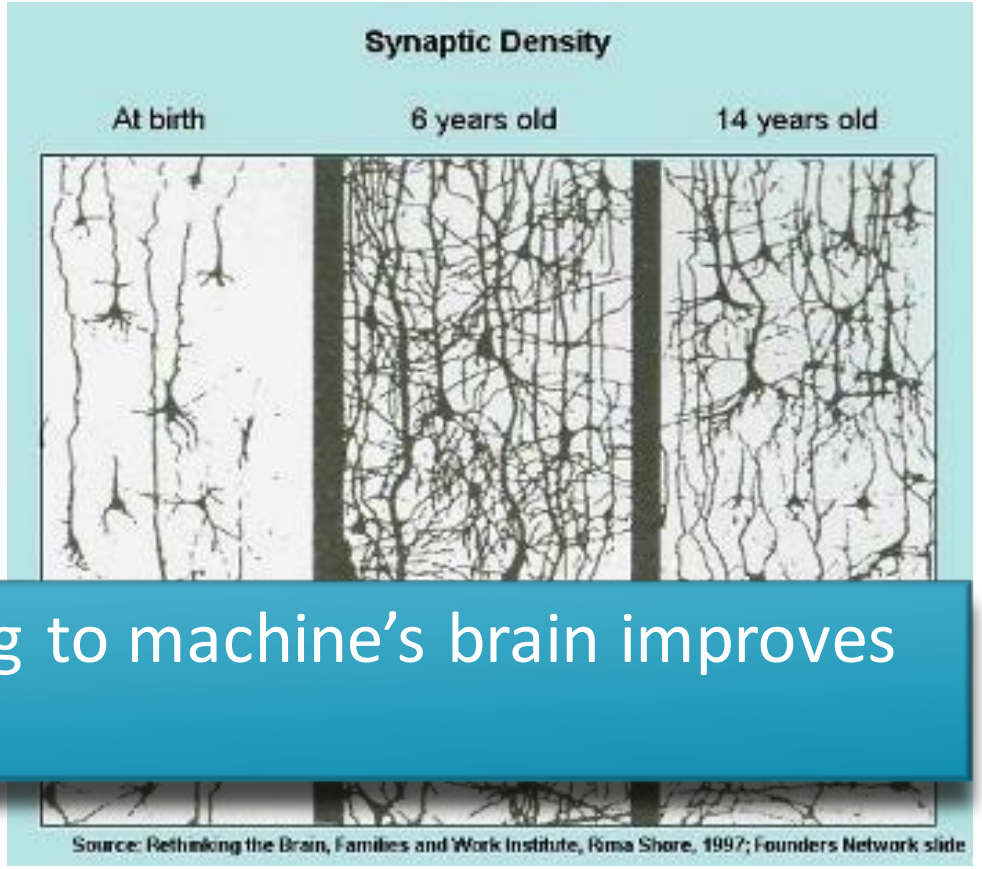
L12 regularization: $\lambda_1 L_1 + \lambda_2 L_2$, also known as elastic net regularization

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Regularization

- Our brain prunes out the useless link between neurons.

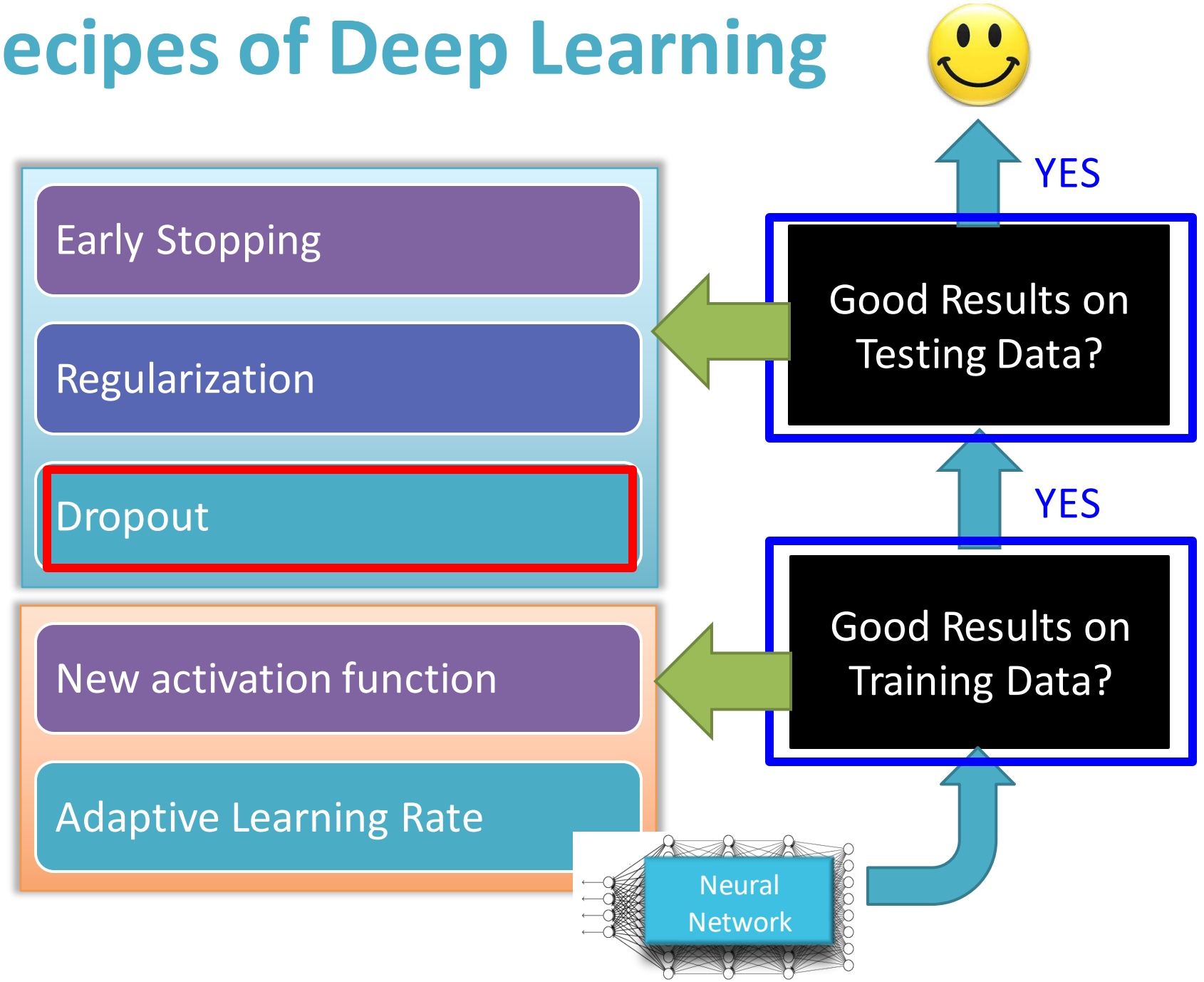


Doing the same thing to machine's brain improves the performance.

UNIVERSITÉ CÔTE D'AZUR

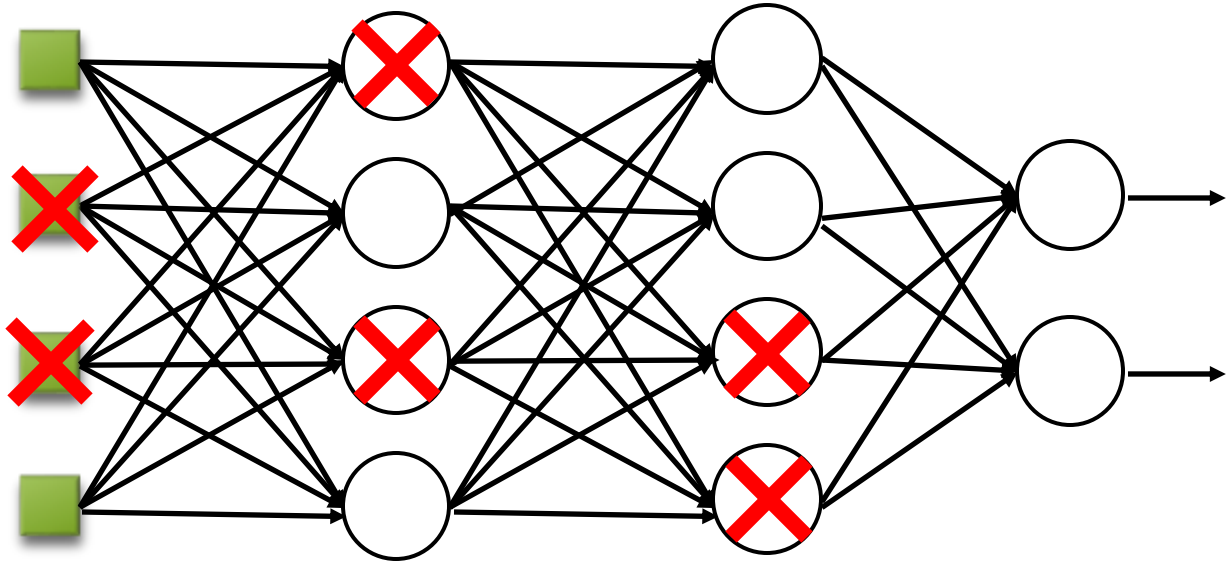
univ-cotedazur.fr

Recipes of Deep Learning



Dropout

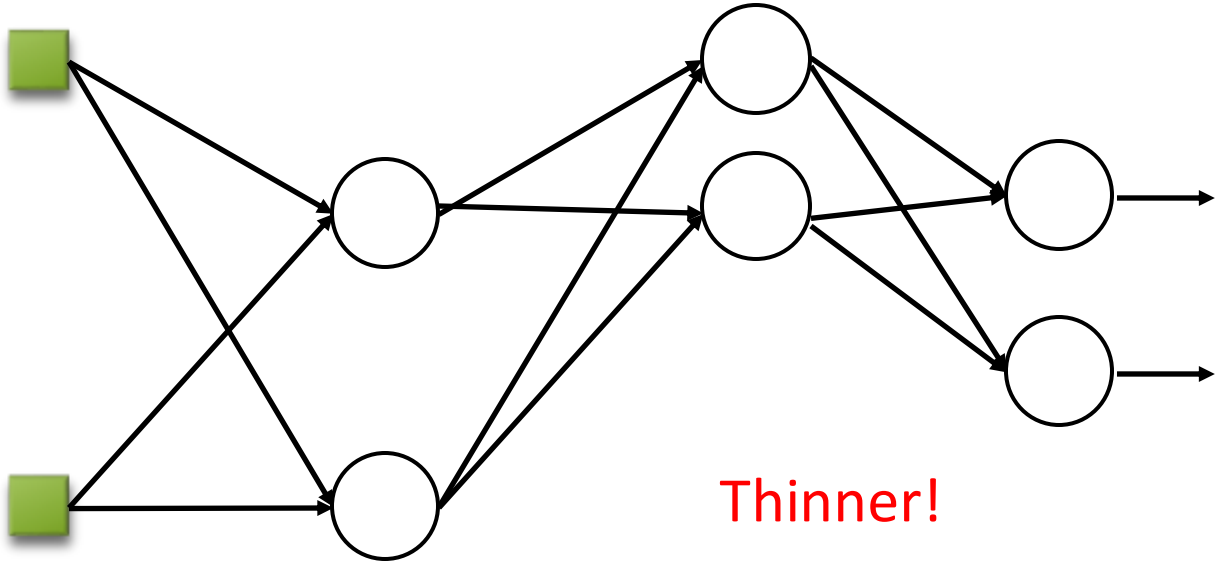
Training:



- **Each time before updating the parameters**
  - Each neuron has p% to dropout

Dropout

Training:



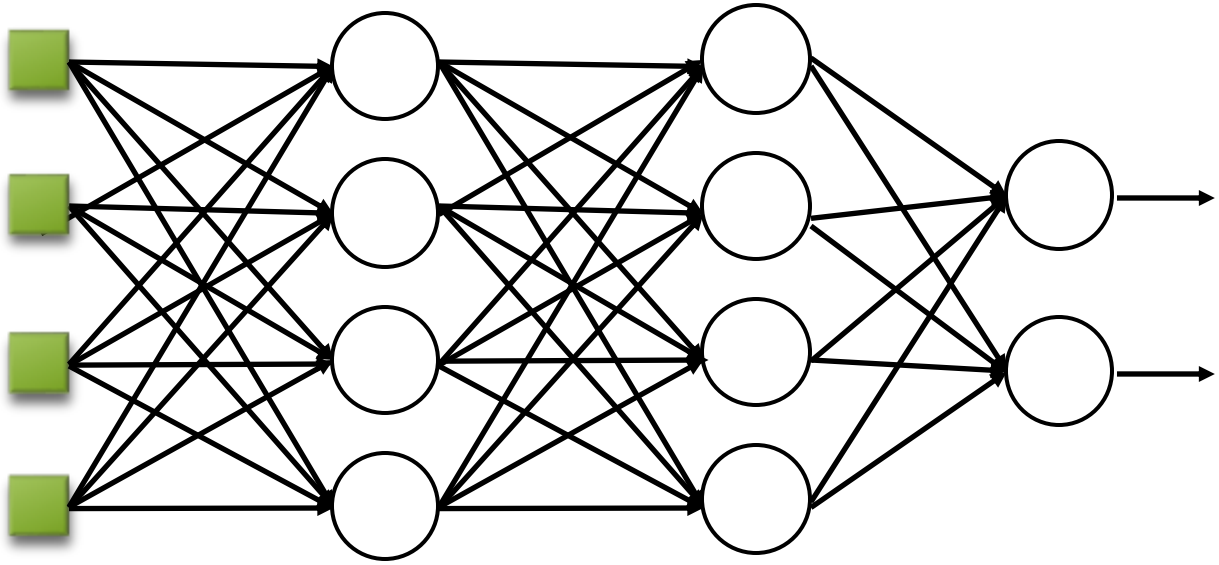
- **Each time before updating the parameters**
  - Each neuron has p% to dropout

    The structure of the network is changed.
  - Using the new network for training

For each mini-batch, we resample the dropout neurons

Dropout

Testing:



➢ **No dropout**

- If the dropout rate at training is p%,
  all the weights times 1-p%
- Assume that the dropout rate is 50%.
  If a weight $w = 1$ by training, set $w = 0.5$ for testing.

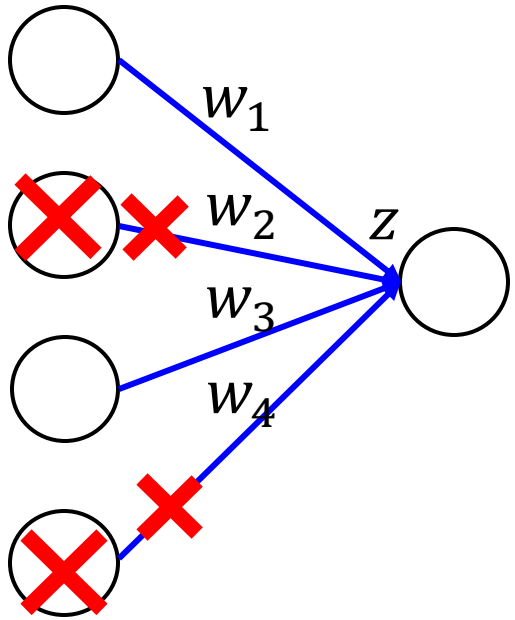
UNIVERSITÉ CÔTE D'AZUR
univ-cotedazur.fr

Dropout – intuitive reason

- Why the weights should multiply (1-p)% (dropout rate) when testing?

Training of Dropout

Assume dropout rate is 50%

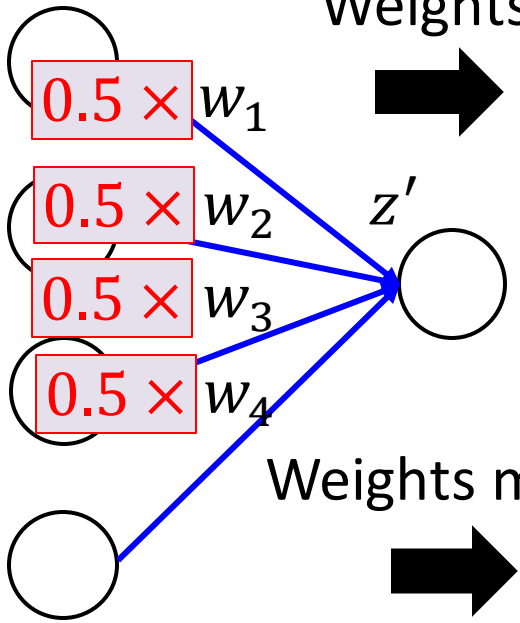


Testing of Dropout

No dropout

Weights from training

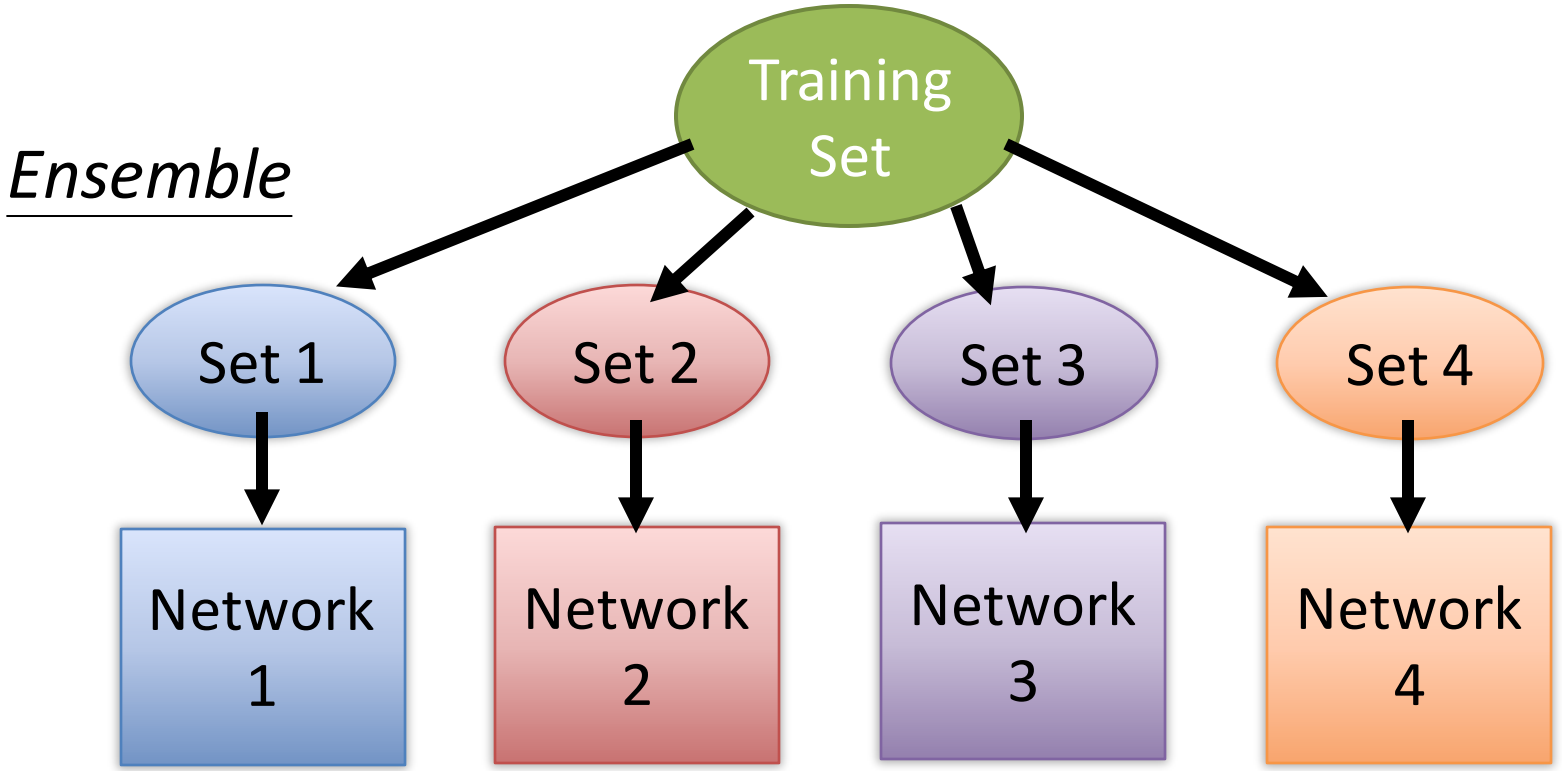
$z' \approx 2z$



Weights multiply 1-p%

$z' \approx z$

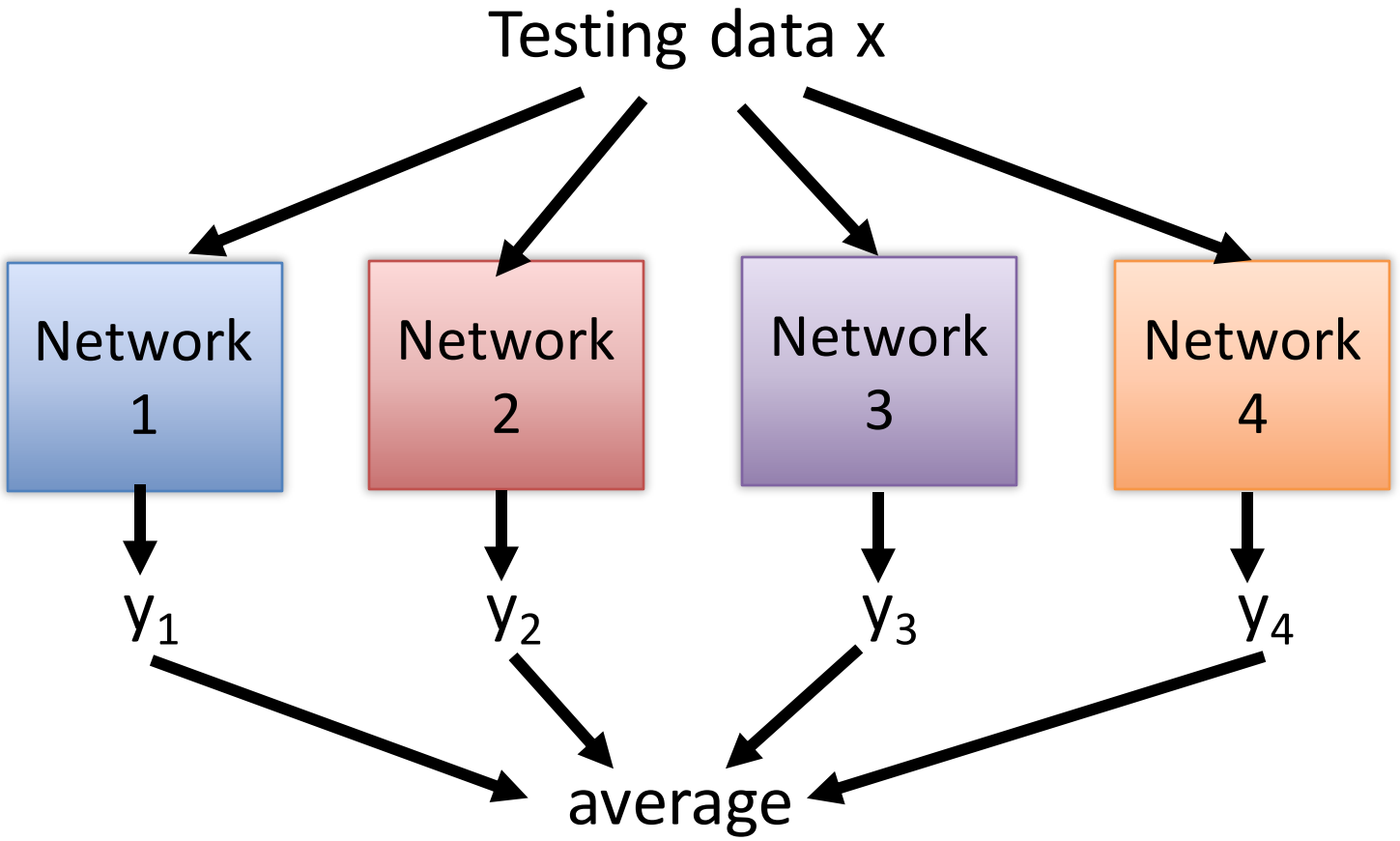
Dropout is a kind of ensemble technique



Train a bunch of networks with different structures

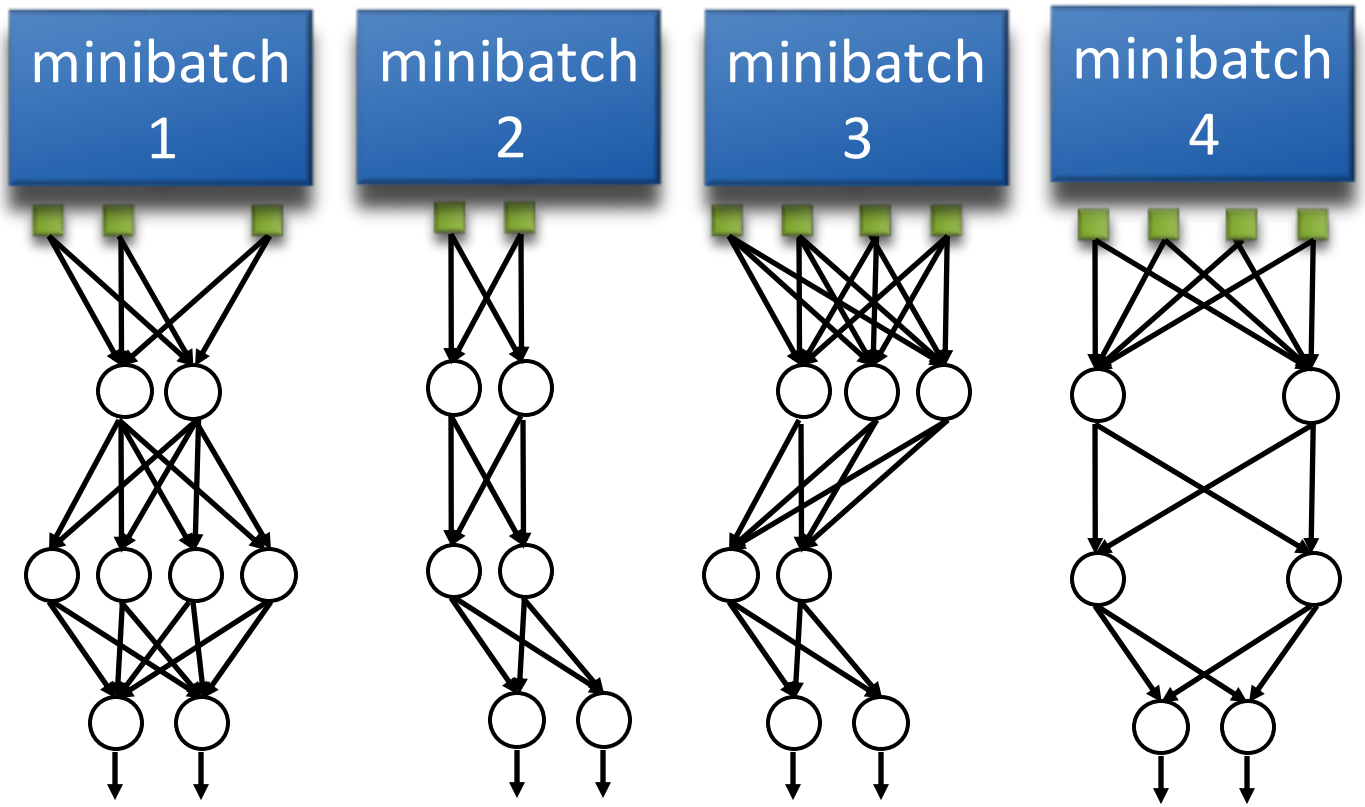
Dropout is a kind of ensemble technique

Ensemble



UNIVERSITÉ CÔTE D'AZUR
univ-cotedazur.fr

Dropout is a kind of ensemble technique



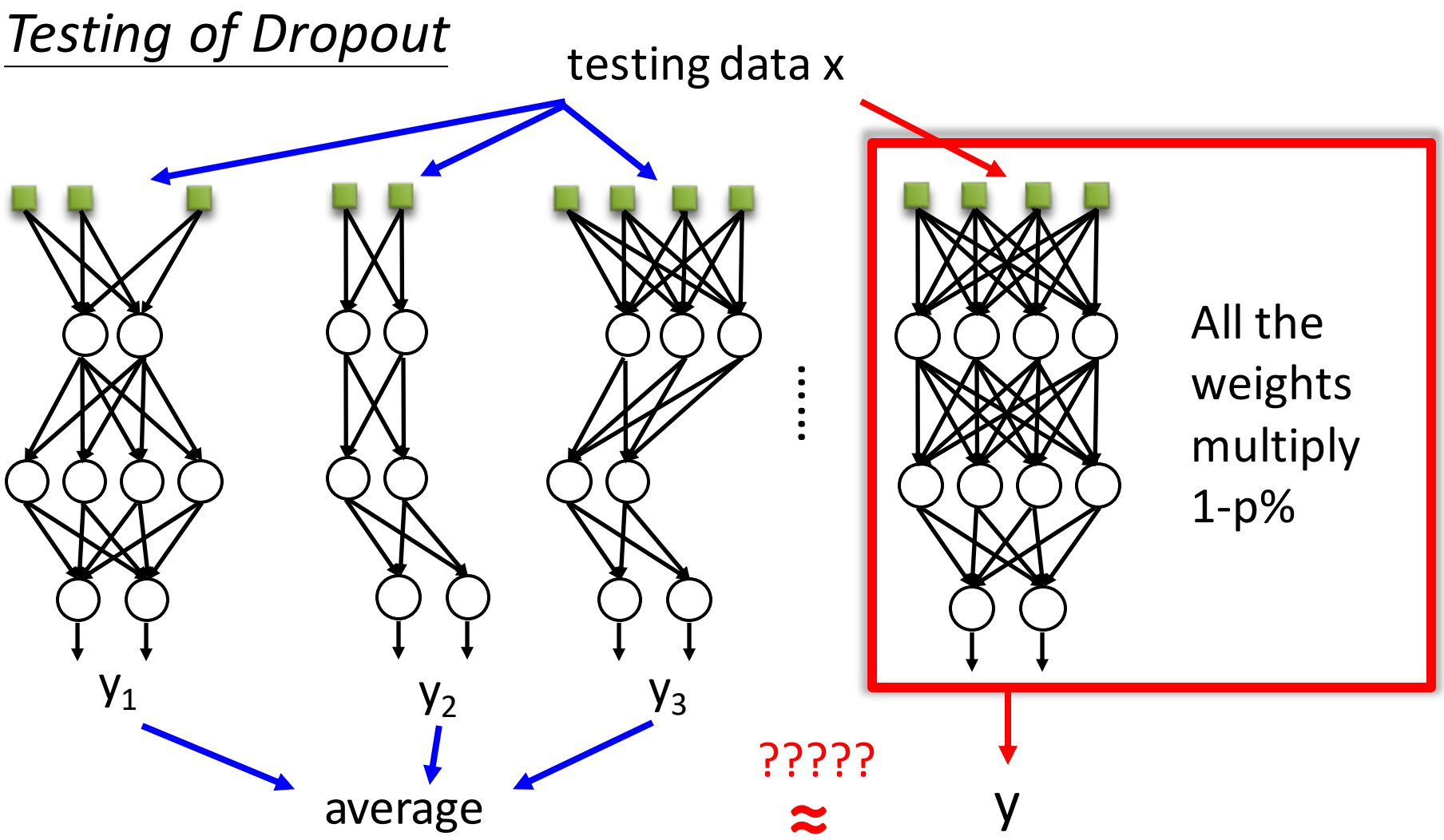
Training of Dropout

M neurons

2^M possible networks

- Using one mini-batch to train one network
- Some parameters in the network are shared

Dropout is a kind of ensemble technique

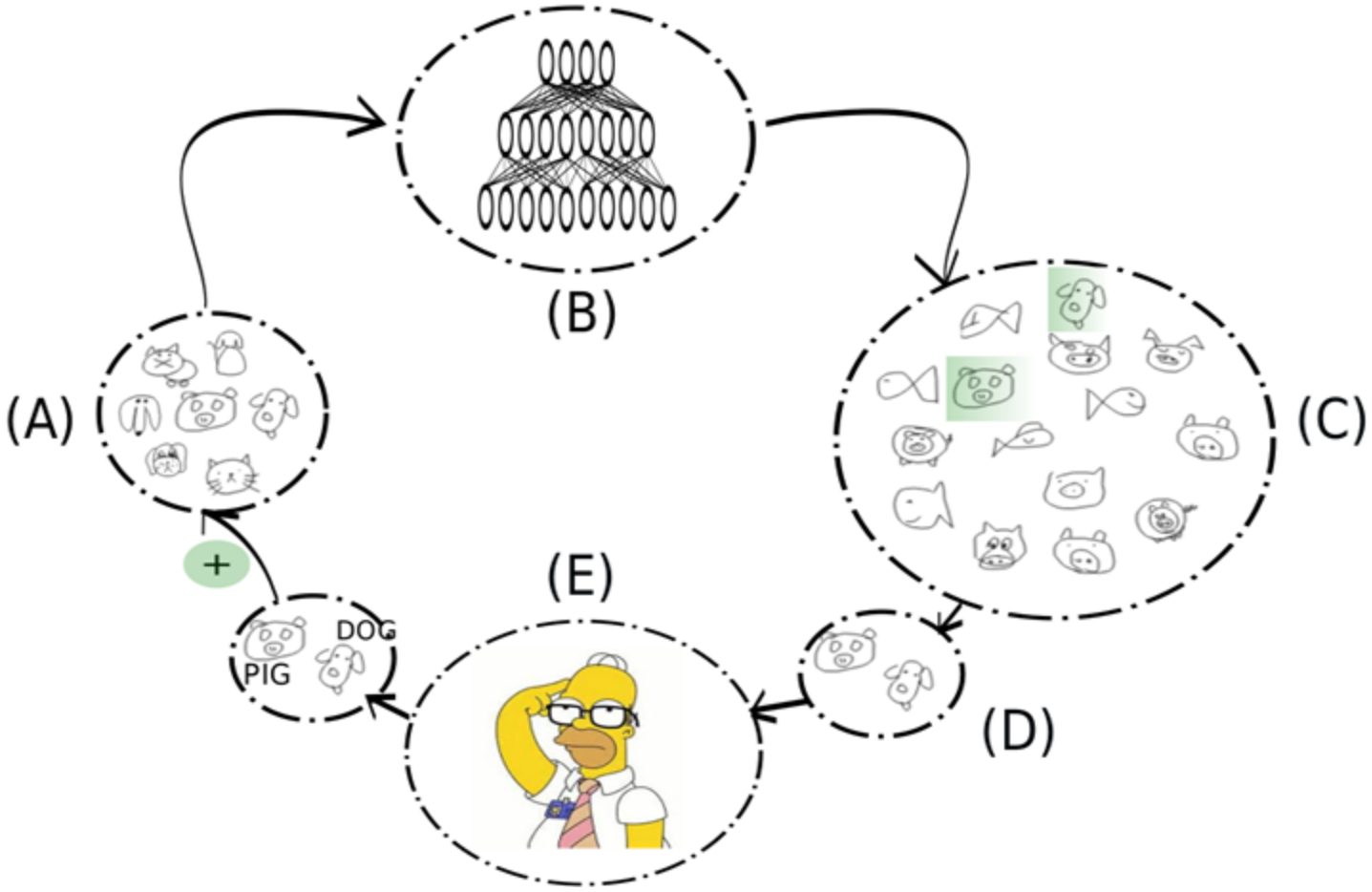


Benefit Dropout to design an Active Learning Strategy

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

ACTIVE Supervised Classification



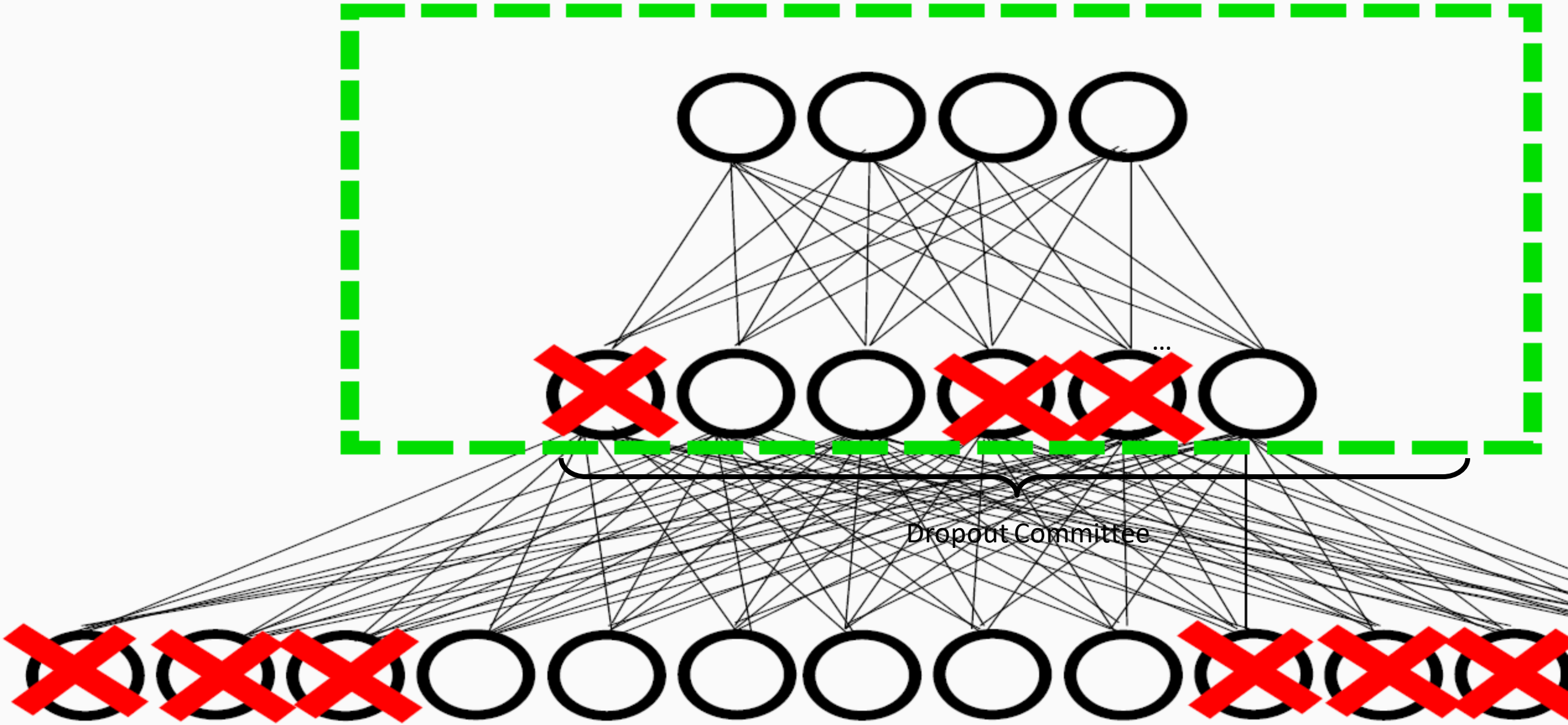
(A)=labelled data

(B)=model

(C)=**POOL** of unlabelled data

(D)=**BATCH** of query

(E) ORACLE (eg human)


...
Dropout Committee

UNIVERSITÉ CÔTE D'AZUR
univ-cotedazur.fr

Sample Selection with margin-like idea

- Our own metric based on *"how a partial CNN may change its decision to be in accordance with the majority"*.
- Committee = $\{pCNN_i\}$ with p^i the output probability vector of $pCNN_i$
- Given a sample ***x***, we first establish its most probable label based $\mathbf{LABEL}(\mathbf{x}) = \underset{j}{\operatorname{argmax}} \sum_{pCNN_i} 1_{j=\underset{k}{argmax}\ p^i(y=k|\mathbf{x})}$

Sample Selection with margin-like idea

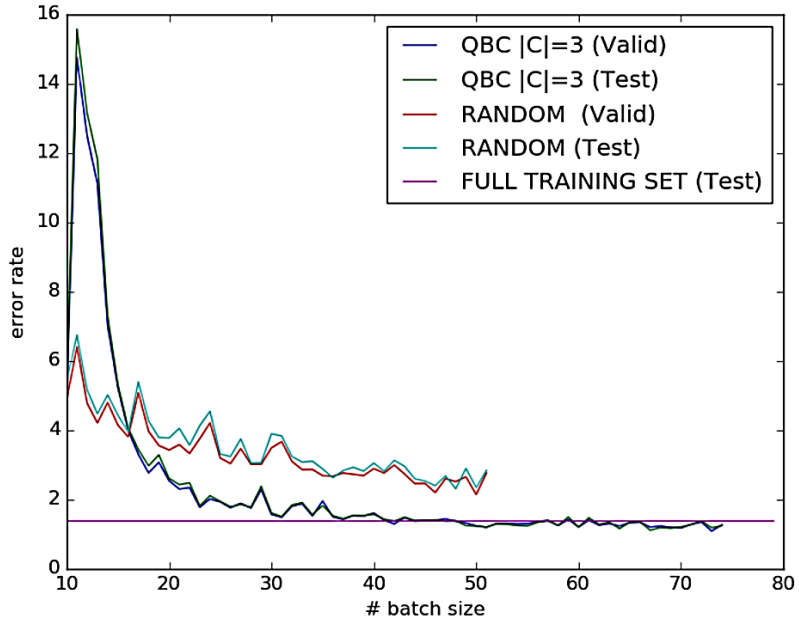
- Inspired from Random Forest margin function to produce a ranking of candidates for selection, we want accounting for the confidence of the CNN in the scoring function.

$$rg(\mathbf{x}, \mathbf{LABEL}(\mathbf{x})) = \sum_{pCNN_i} \max_j p^i(y = j \mid \mathbf{x}) - p^i(y = \mathbf{LABEL}(\mathbf{x}) \mid \mathbf{x})$$

$$query \leftarrow \max_{\mathbf{x}} rg(\mathbf{x}, \mathbf{LABEL}(\mathbf{x}))$$

Majority voting on MNIST

- For **MNIST** again, with the previous selection function (majority voting) 30% of the database was required.



METHOD	MEAN error rate	MIN error rate
QBDC	1.10	0.99
RANDOM	2.13	1.78

Average and best error rate on MNIST test set (30%)

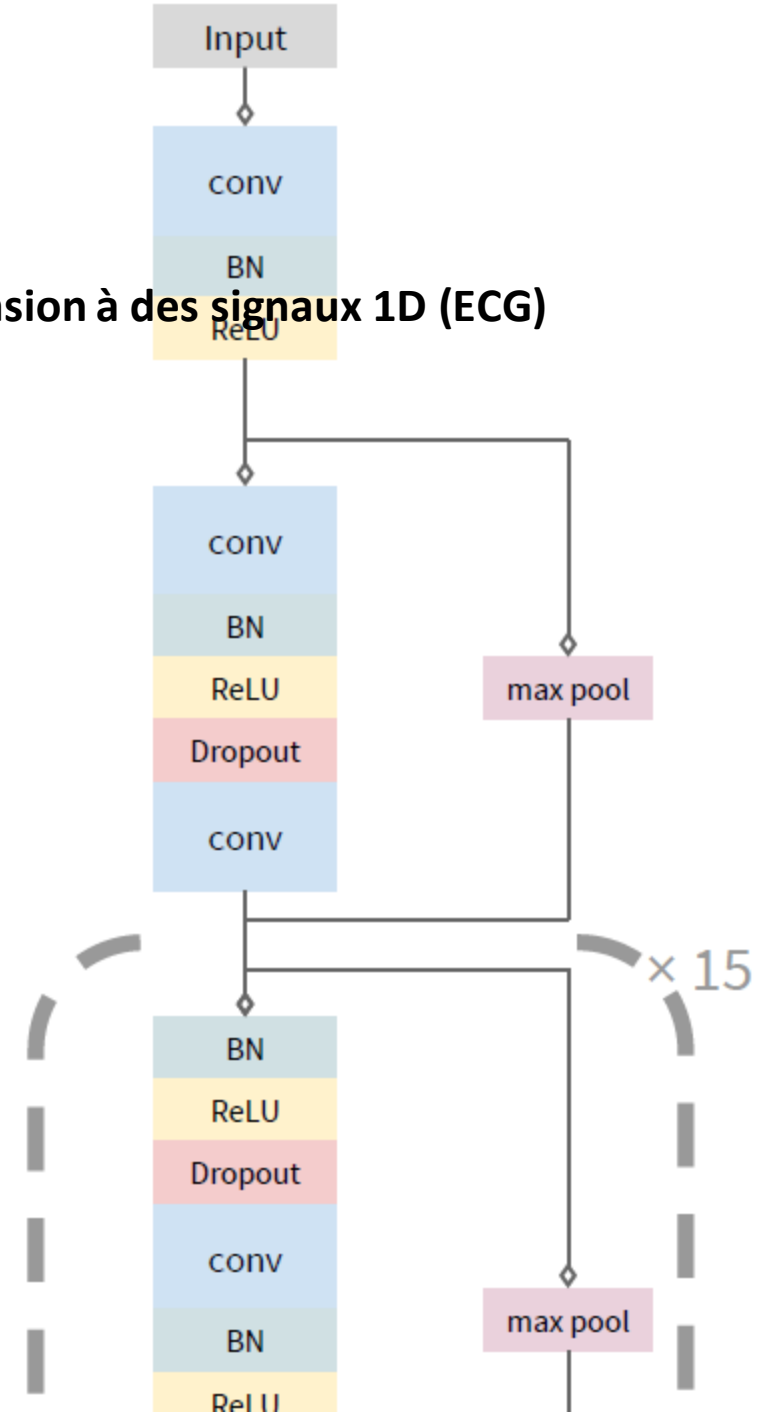
univ-cotedazur.fr

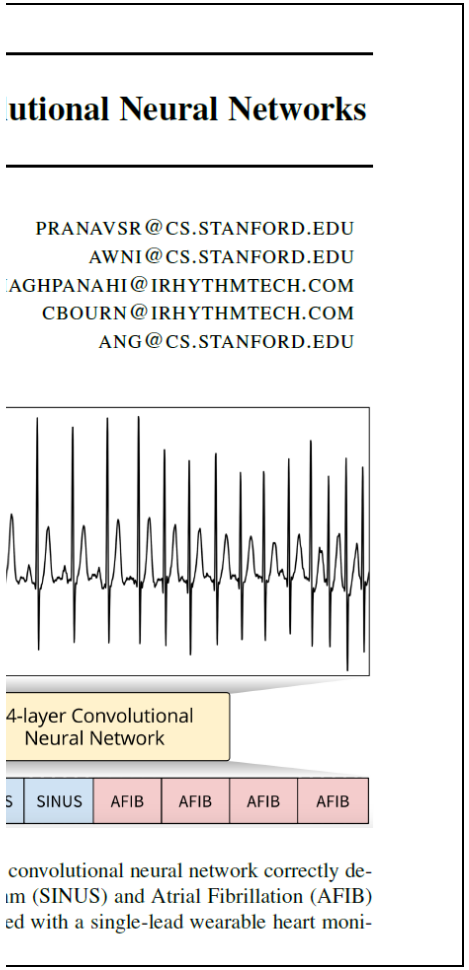
And many other tricks and tips but with these ones, you should be fine to start…

CNN for other domains than images

other application domains

Extension à des signaux 1D (ECG)





utional Neural Networks

PRANAVSR@CS.STANFORD.EDU
AWNI@CS.STANFORD.EDU
AGHPANAHI@IRHYTHMTECH.COM
CBOURN@IRHYTHMTECH.COM
ANG@CS.STANFORD.EDU

4-layer Convolutional Neural Network

S | SINUS | AFIB | AFIB | AFIB | AFIB

convolutional neural network correctly de-
m (SINUS) and Atrial Fibrillation (AFIB)
ed with a single-lead wearable heart moni-

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Many other application domains

Extension à des signaux 1D (Spectres de Masse)

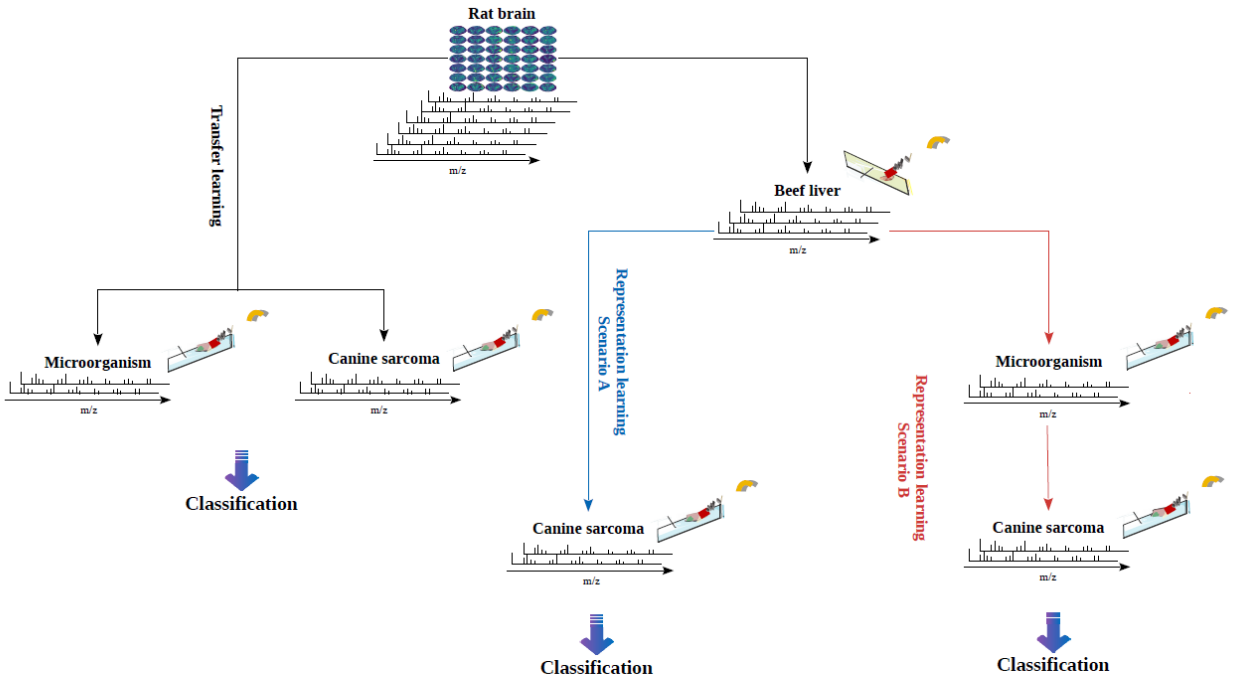


Figure 3: Workflow diagram of 1D-CNN classification by transfer learning and cumulative representation learning on SpiderMass datasets

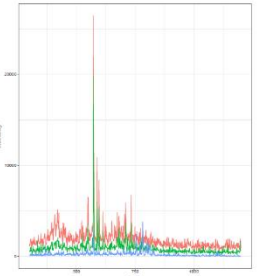


Figure 2: An example of three spectra from canine myxosarcoma type indicating the within-class variation

Nature Communications : https://www.nature.com/articles/s41467-020-19354-z

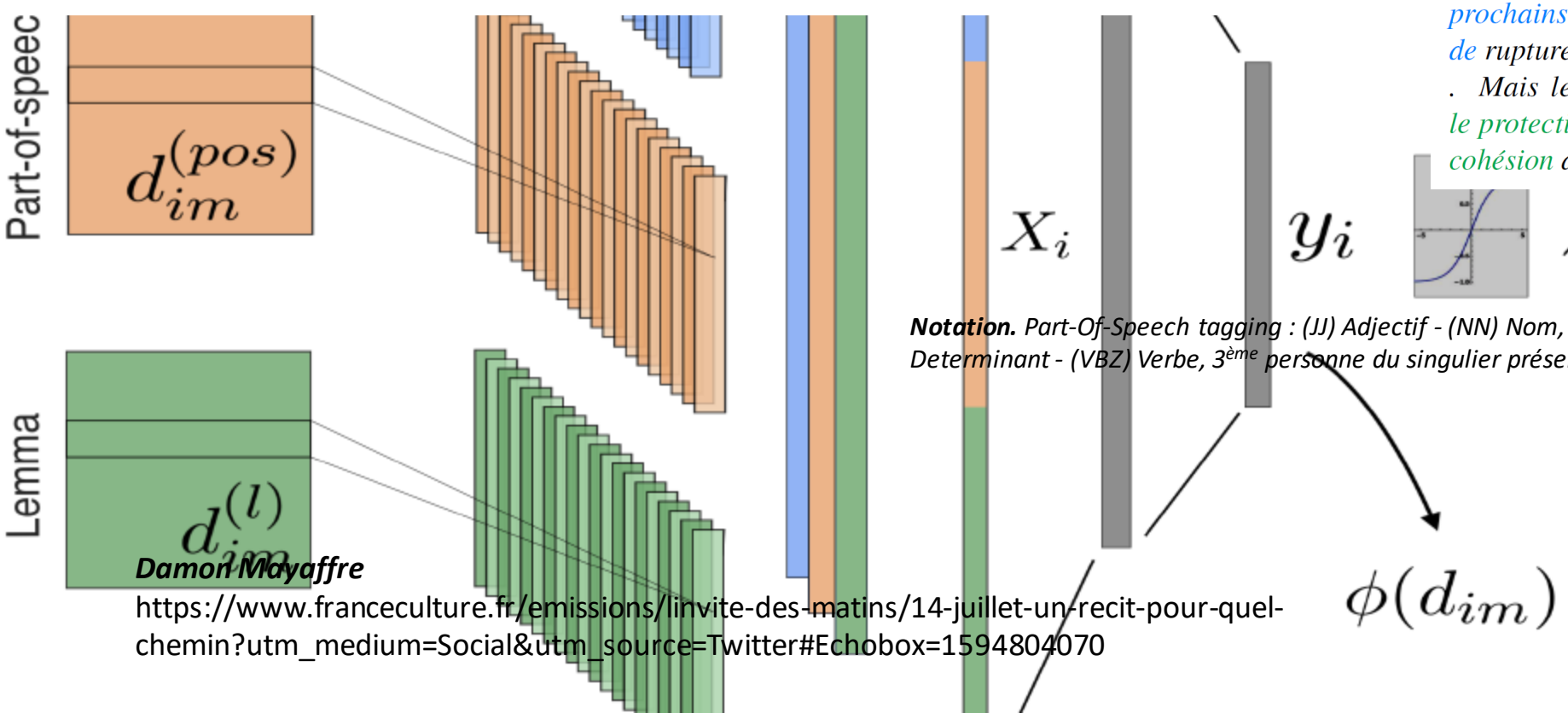
ɔmains

[...] and incredibly strong . Our unemployment is at a historic low . This JJ economic prosperity gives us NN , reserves , and resources to handle any threat that comes our way . DT VBZ

Discours présidentiel
D.J. Trump, 11 Mars 2020

Discours présidentiel
E. Macron, 12 Mars 2020

[...] qui tiennent fermement leur destin en main . Les prochaines semaines et les prochains mois VER:futu des décisions de rupture en ce sens . Je les assumerai . Mais le temps , aujourd'hui , est à le protection de nos concitoyens et à le cohésion de le Nation [...]



Notation. *Part-Of-Speech tagging : (JJ) Adjectif - (NN) Nom, singulier ou pluriel - (DT) Determinant - (VBZ) Verbe, 3ème personne du singulier présent, VER:futu, verbe au futur.*

Damon Mayaffre
https://www.franceculture.fr/emissions/invite-des-matins/14-juillet-un-recit-pour-quel-chemin?utm_medium=Social&utm_source=Twitter#Echobox=1594804070