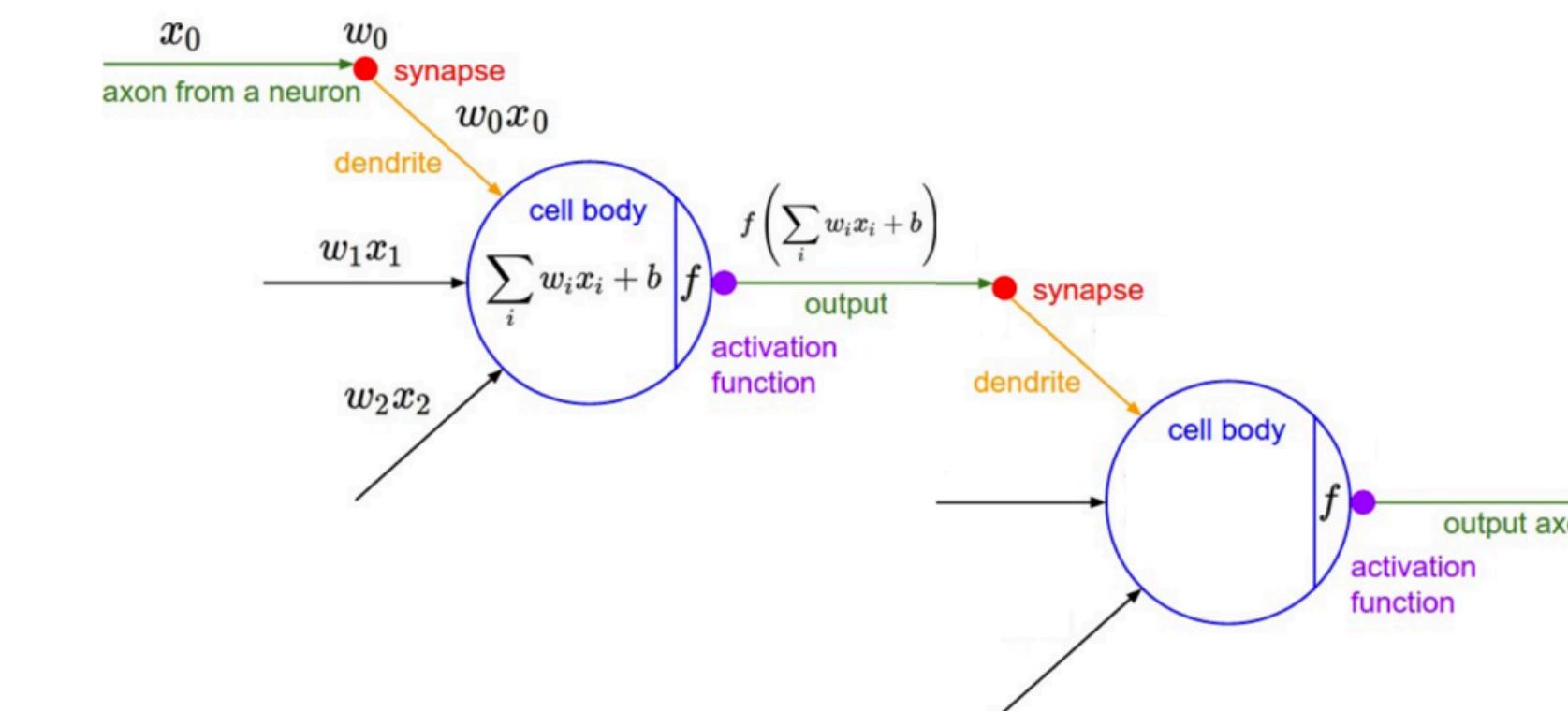
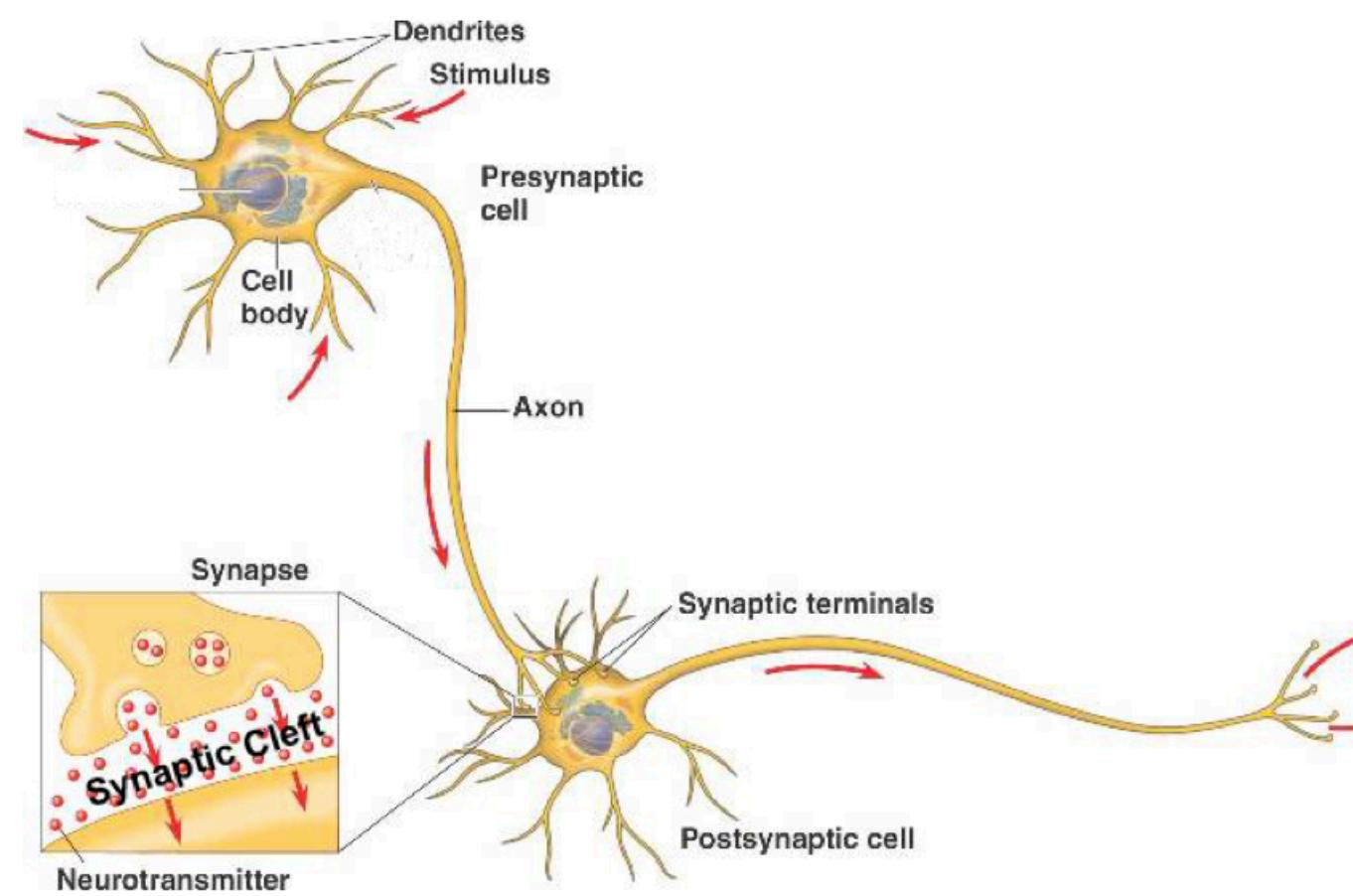
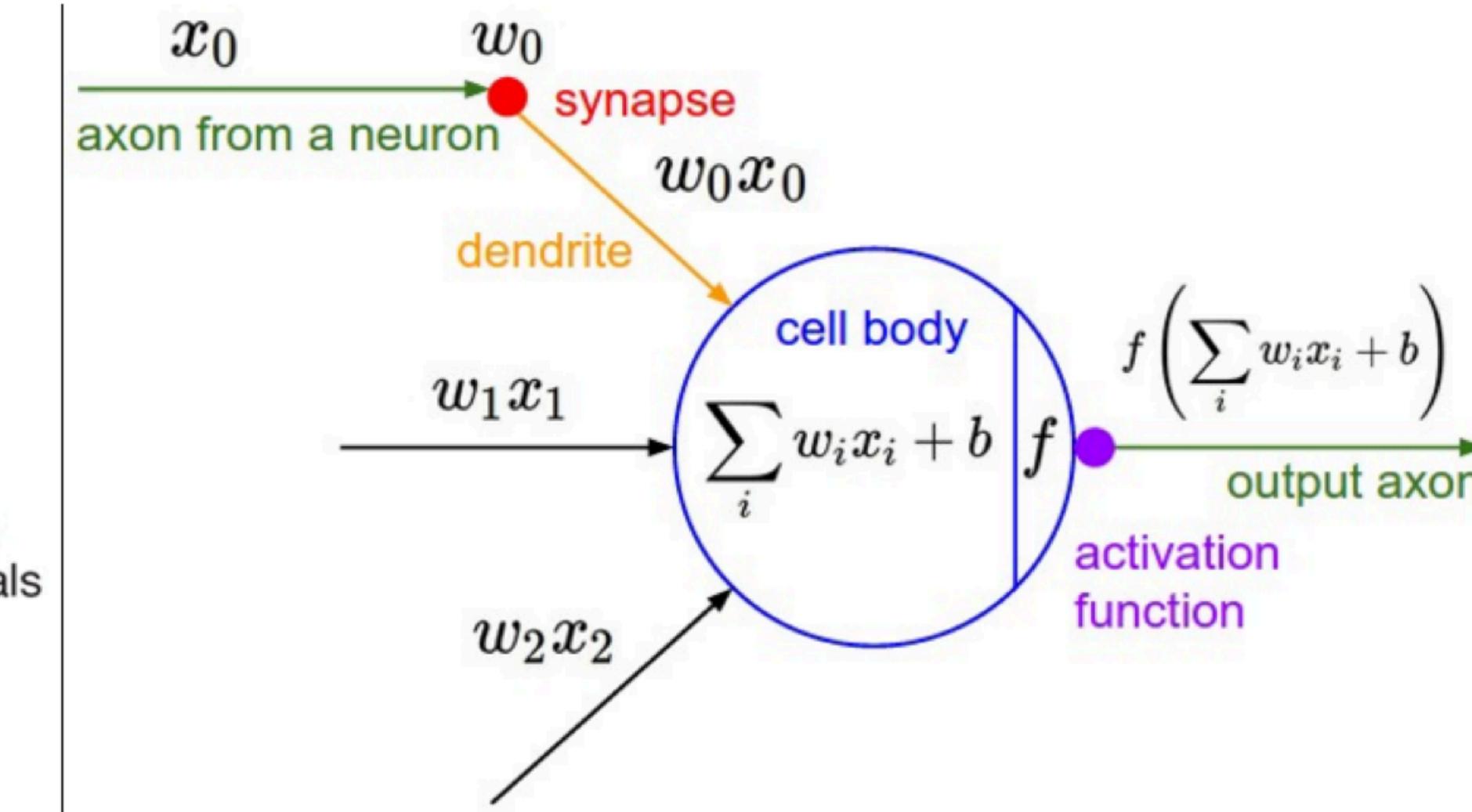
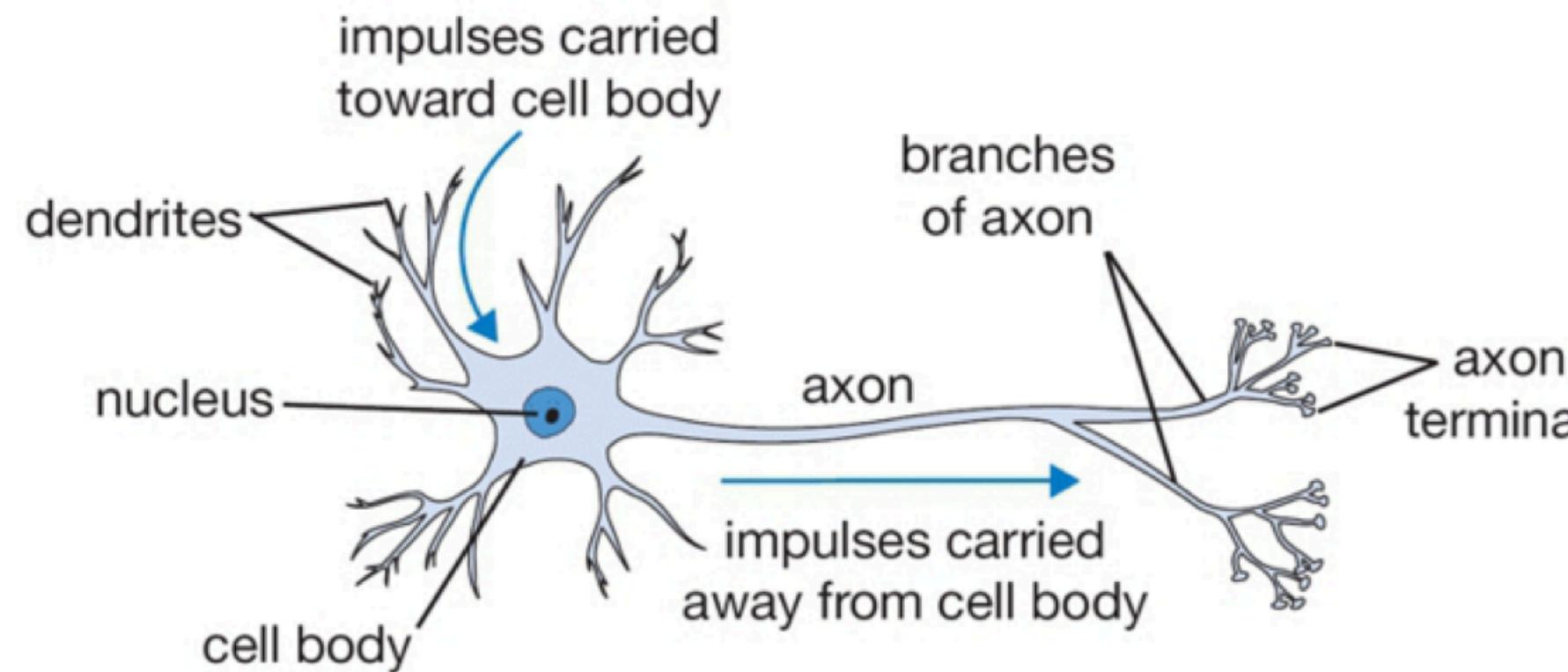


Neural Network

Multi-Layer Perceptron & Deep Learning

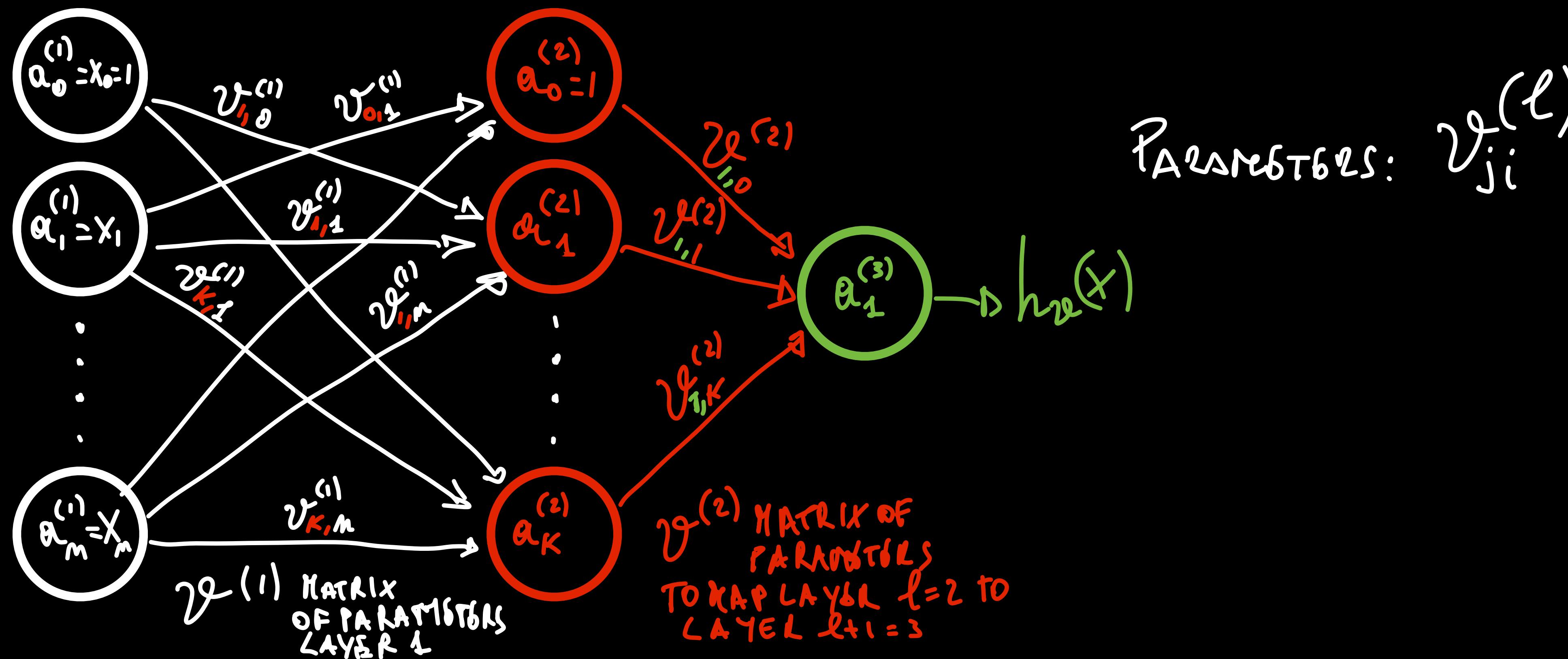
Model Representation



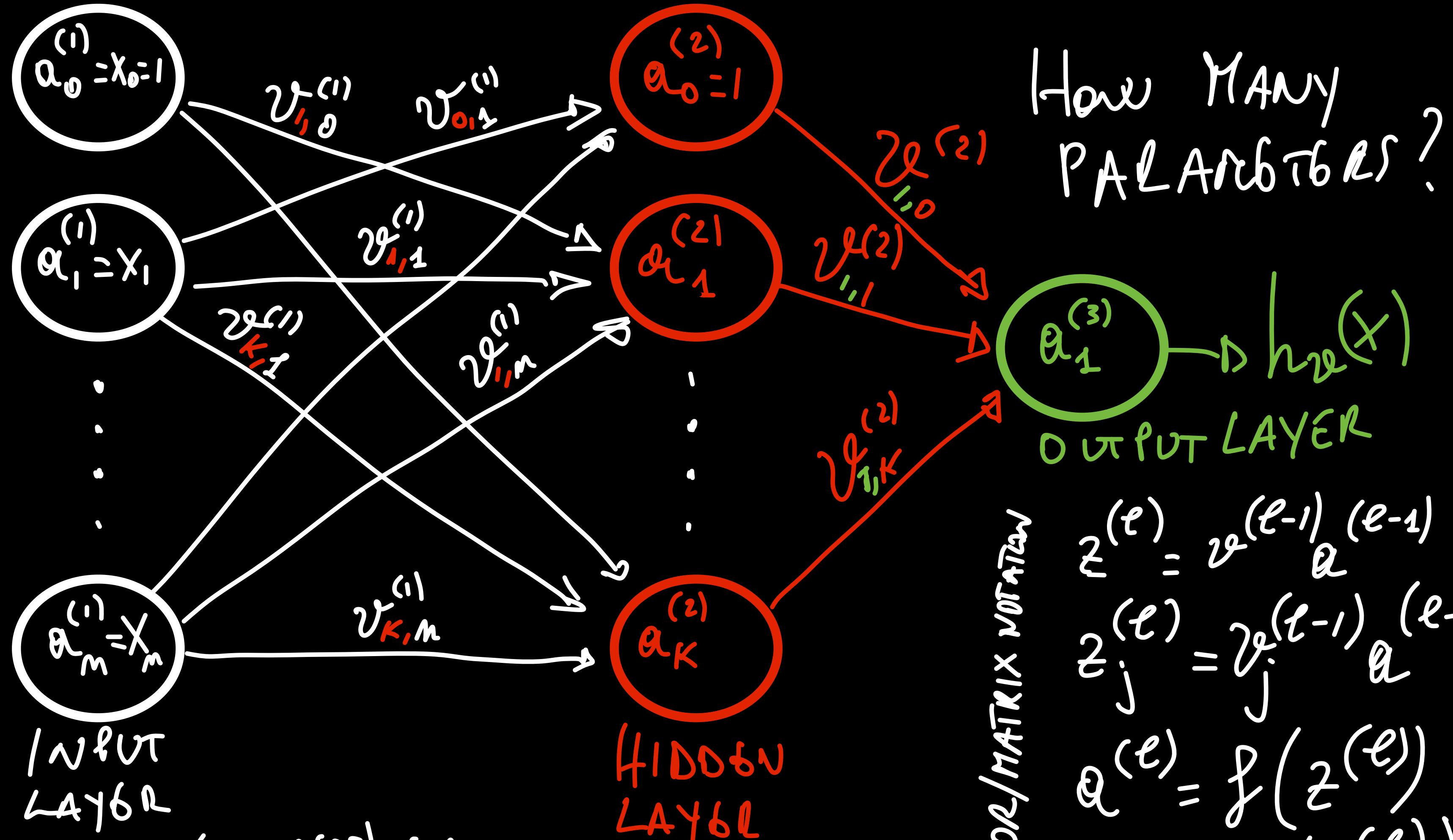
NEURAL NETWORK - MODEL REPRESENTATION

THE MULTI-LAYER PERCEPTRON (MLP)

- MOTIVATIONS:
- 1) WE NEED TO FIT DATA WHICH LIVES IN NON-LINEAR SPACE
(E.G., XOR PROBLEM)
 - 2) GIVEN A SET OF DATA (E.G., IMAGES, TEXT DOCUMENTS)
WE NEED TO LEARN A GOOD REPRESENTATION OF
THEM (AND NOT AN HAND CRAFTED ONE!)



$\ell \equiv \text{LAYER}$
 $i \equiv \text{AYER M PARTNER}$
 $j \equiv \text{AYER M ARRIVO}$
 $a_i^{(\ell)} \equiv i\text{-th ACTIVATION UNIT OF THE LAYER } \ell$



A NEURAL NET (ALSO DEEP) IS A COMPOSITION OF DERIVABLE FUNCTIONS

$$\begin{aligned}
 h_v(x) &= a_1^{(3)} = f\left(v_{1,0}^{(2)} + v_{1,1}^{(2)} a_1^{(2)} + \dots + v_{1,K}^{(2)} a_K^{(2)}\right) = \\
 &= f\left(v_{1,0}^{(2)} + v_{1,1}^{(2)} f\left(v_{1,0}^{(1)} + v_{1,1}^{(1)} a_1^{(1)} + \dots + v_{1,m}^{(1)} a_m^{(1)}\right) + \dots + v_{1,K}^{(2)} f\left(v_{K,0}^{(1)} + \dots + v_{K,m}^{(1)} a_m^{(1)}\right)\right)
 \end{aligned}$$

TENSOR/MATRIX NOTATION

$$z^{(l)} = v^{(l-1)} a$$

$$z_j^{(l)} = v_j^{(l-1)} a^{(l-1)} = v_{j,0}^{(l-1)} a_0 + v_{j,1}^{(l-1)} a_1 + \dots + v_{j,m}^{(l-1)} a_m$$

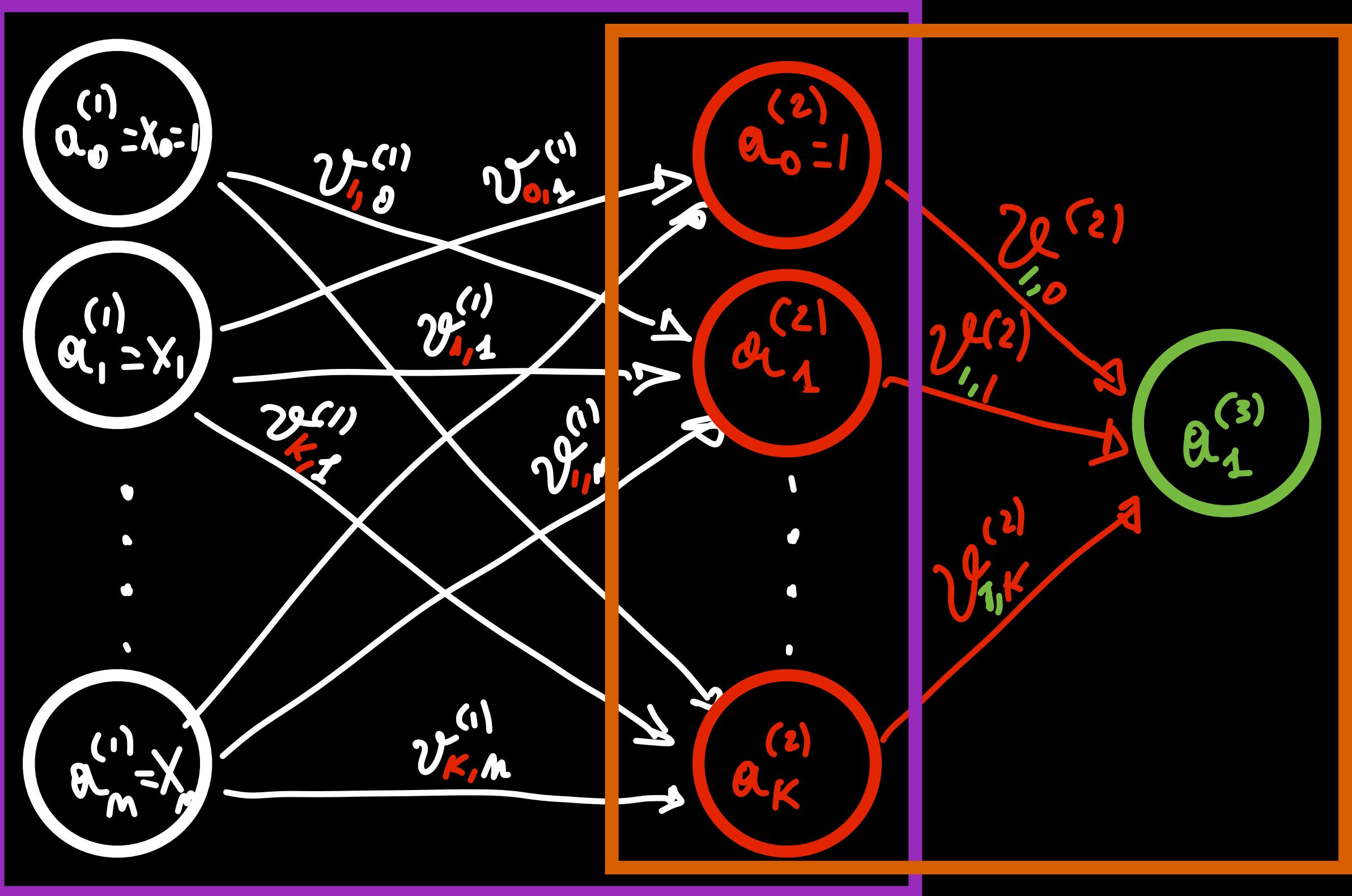
$$a^{(l)} = f(z^{(l)})$$

$$a_j^{(l)} = f(z_j^{(l)})$$

$f = f$ = DERIVABLE FUNCTION (E.g. SIGMOID)

How MANY
PARAMETERS?

$a_1^{(3)}$ $\rightarrow h_v(x)$
OUTPUT LAYER



A NEURAL NETWORK
IS COMPOSED
BY TWO PARTS:

- USEFUL TO BAR A FEATURES SPACE IN A SUPERVISED WAY DEPENDING ON THE TASK (HENCE HOW TO TRANSFORM x IN $q^{(2)}$)
- A SOFTMAX (LOGISTIC REGRESSOR IF BINARY CLASSIFIER) WHICH TAKE AS INPUT THE LEARNED FEATURES

“Universal Approximation Theorem”

A feedforward network with a single layer is sufficient to represent any function, but the layer may be infeasibly large and may fail to learn and generalize correctly (Ian Goodfellow).

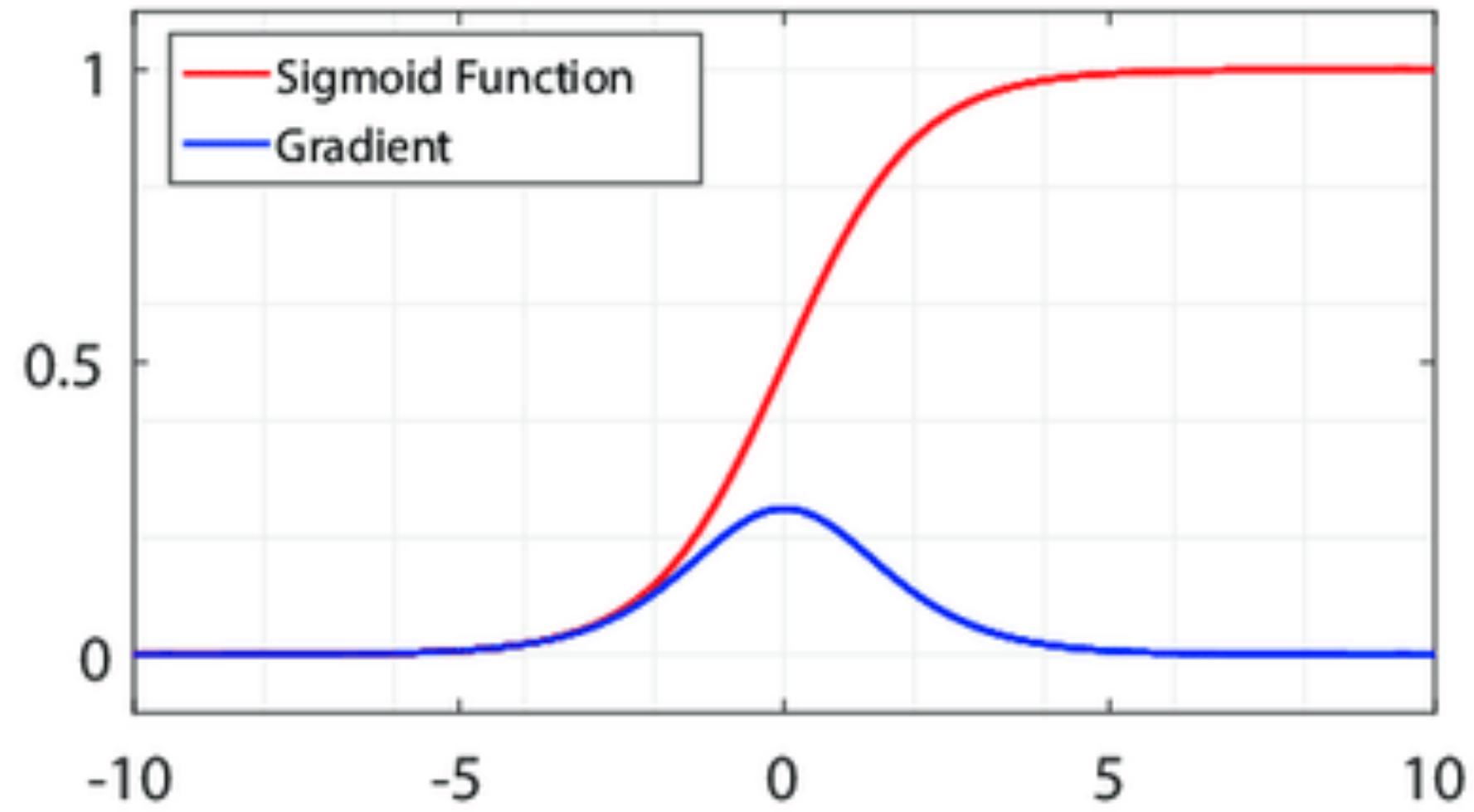
- YOU CAN ADD AS MANY AS HIDDEN LAYERS YOU NEED TO MAKE THE NETWORK DEEP (A DEEP NETWORK)
- IN ORDER TO FIT YOUR NON-LINEAR PROBLEM
- DIFFERENT HIDDEN LAYER CAN HAVE DIFFERENT NUMBER OF NEURON
- NEURONS USUALLY USE THE SAME ACTIVATION FUNCTION

Activation Functions

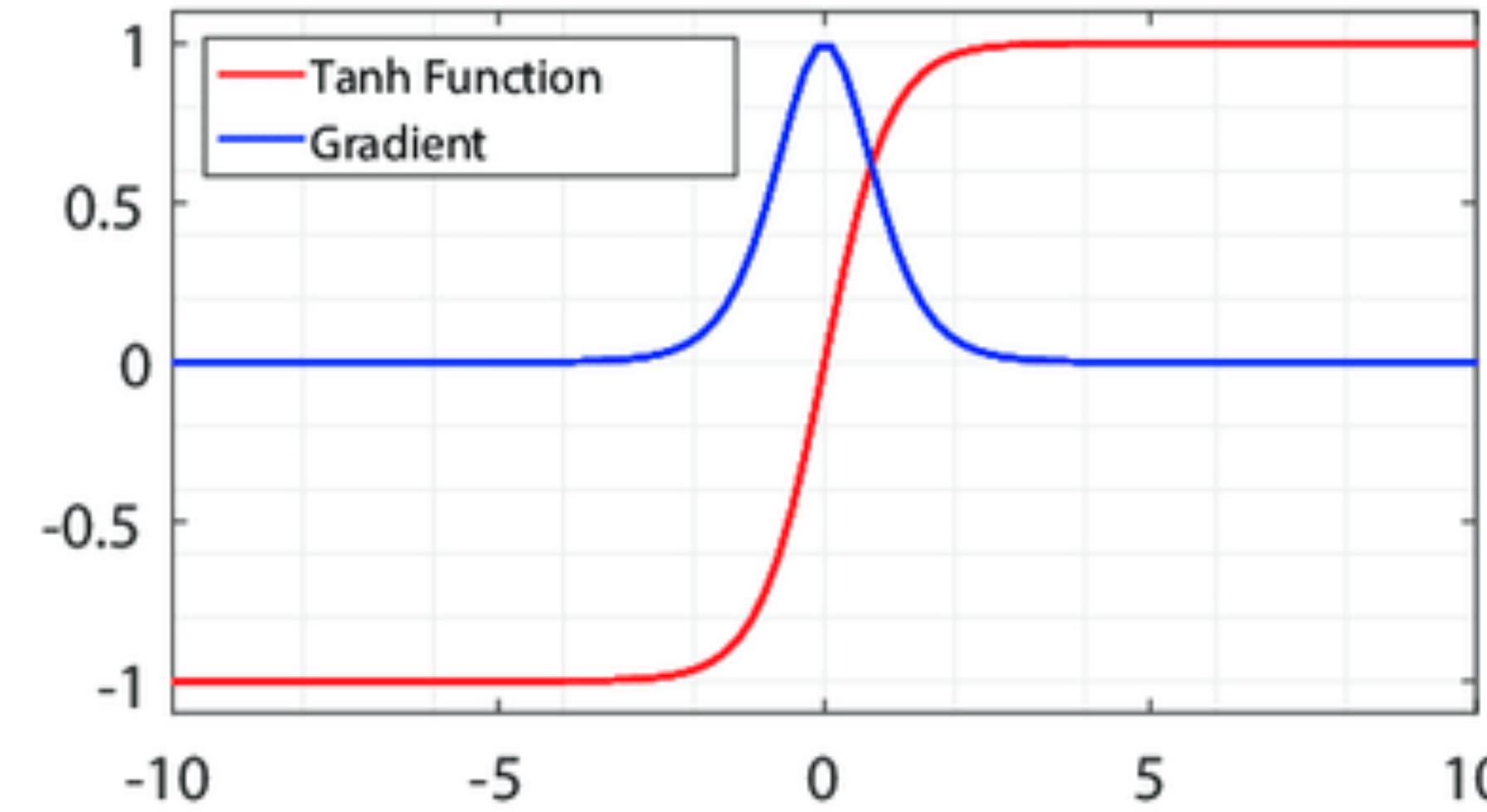
Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
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		$f(x) = \log_e(1 + e^x)$	$f'(x) = \frac{1}{1 + e^{-x}}$

EVERY ACTIVATION FUNCTION YOU WANT TO USE HAS TO BE DIFFERENTIABLE
 ⇒ COMPOSITION (NEURAL NET) OF DIFFERENTIABLE FUNCTIONS IS DIFFERENTIABLE

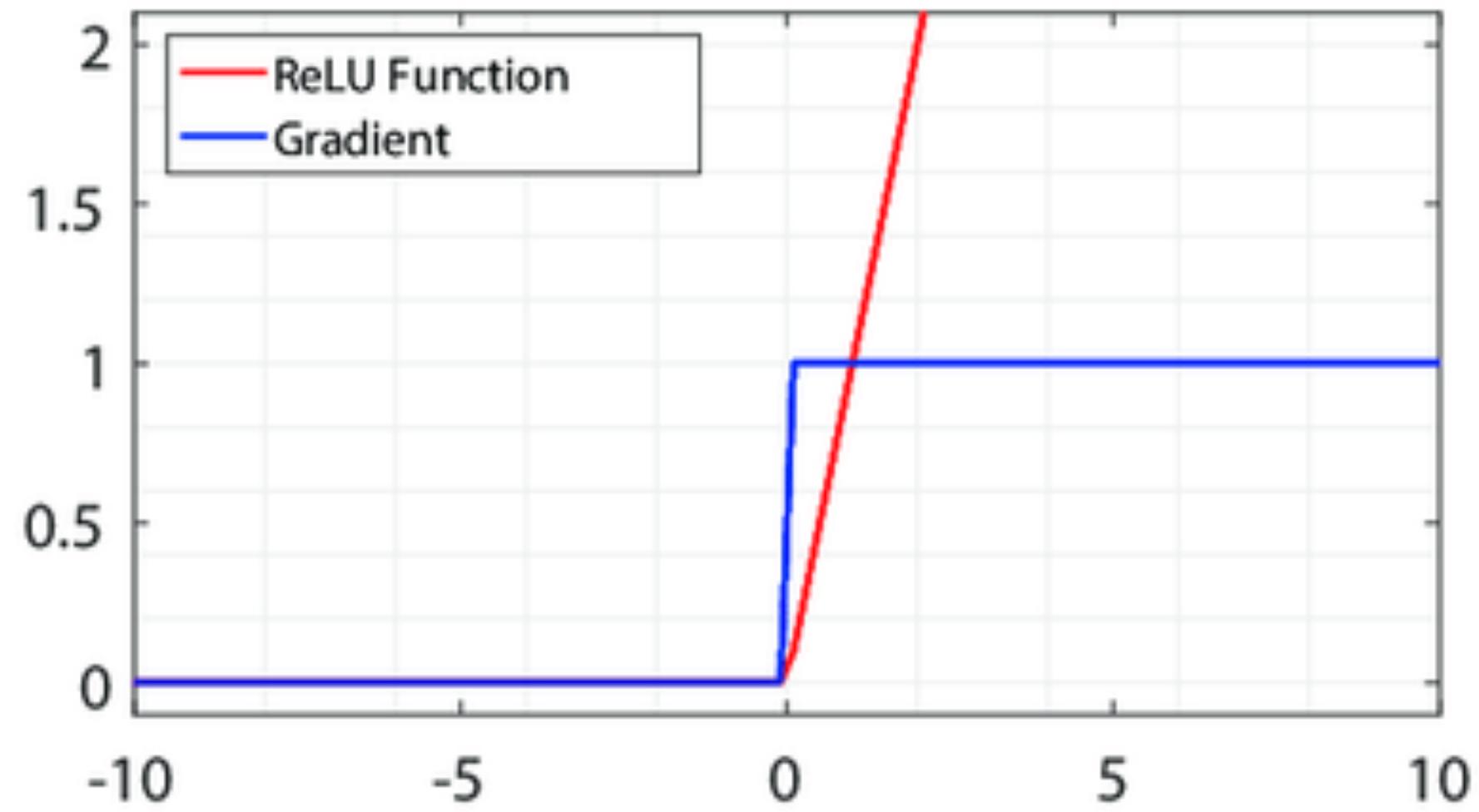
Most USED ACTIVATION FUNCTIONS AND THEIR DERIVATIVES



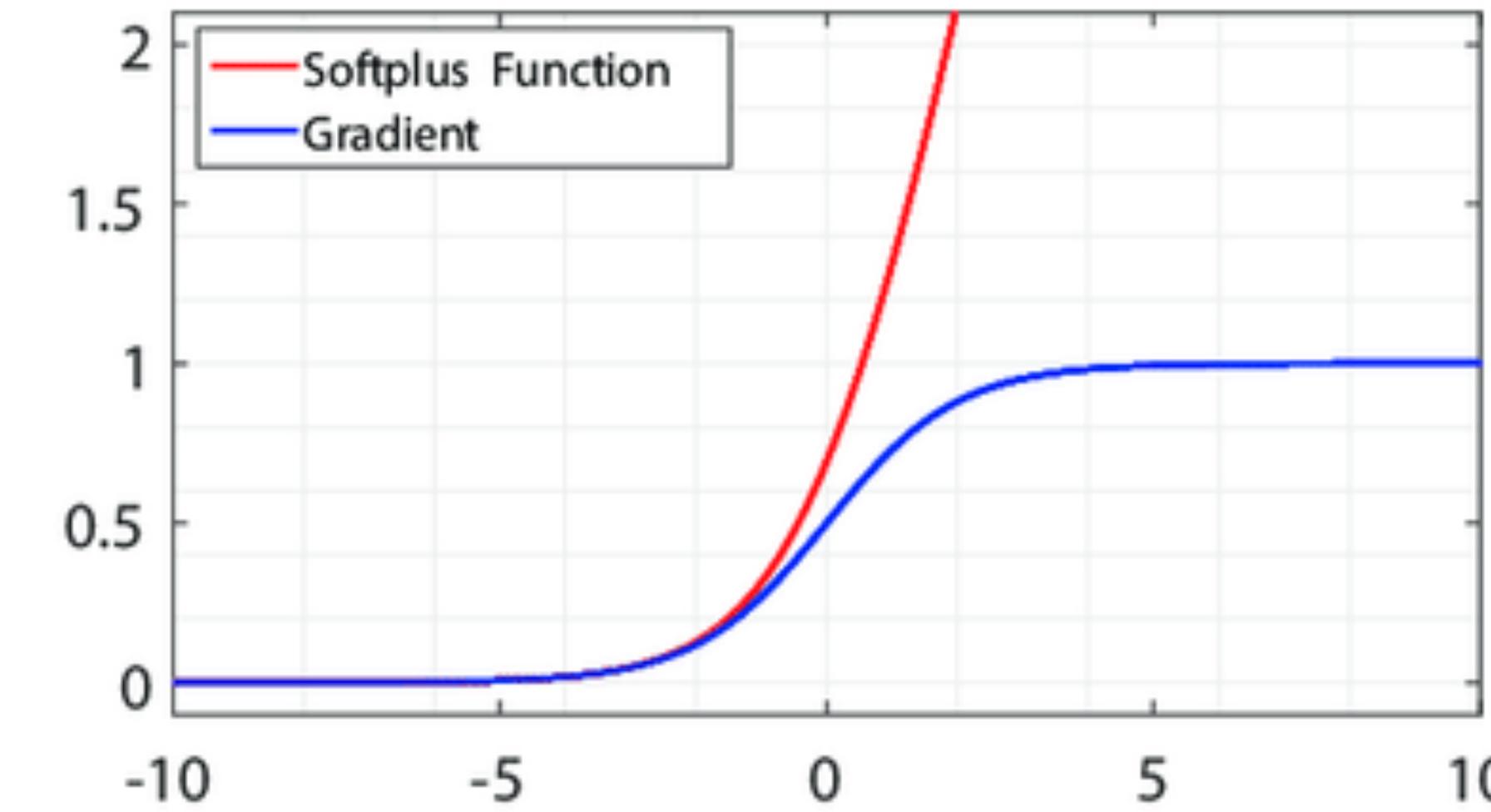
(a)



(b)

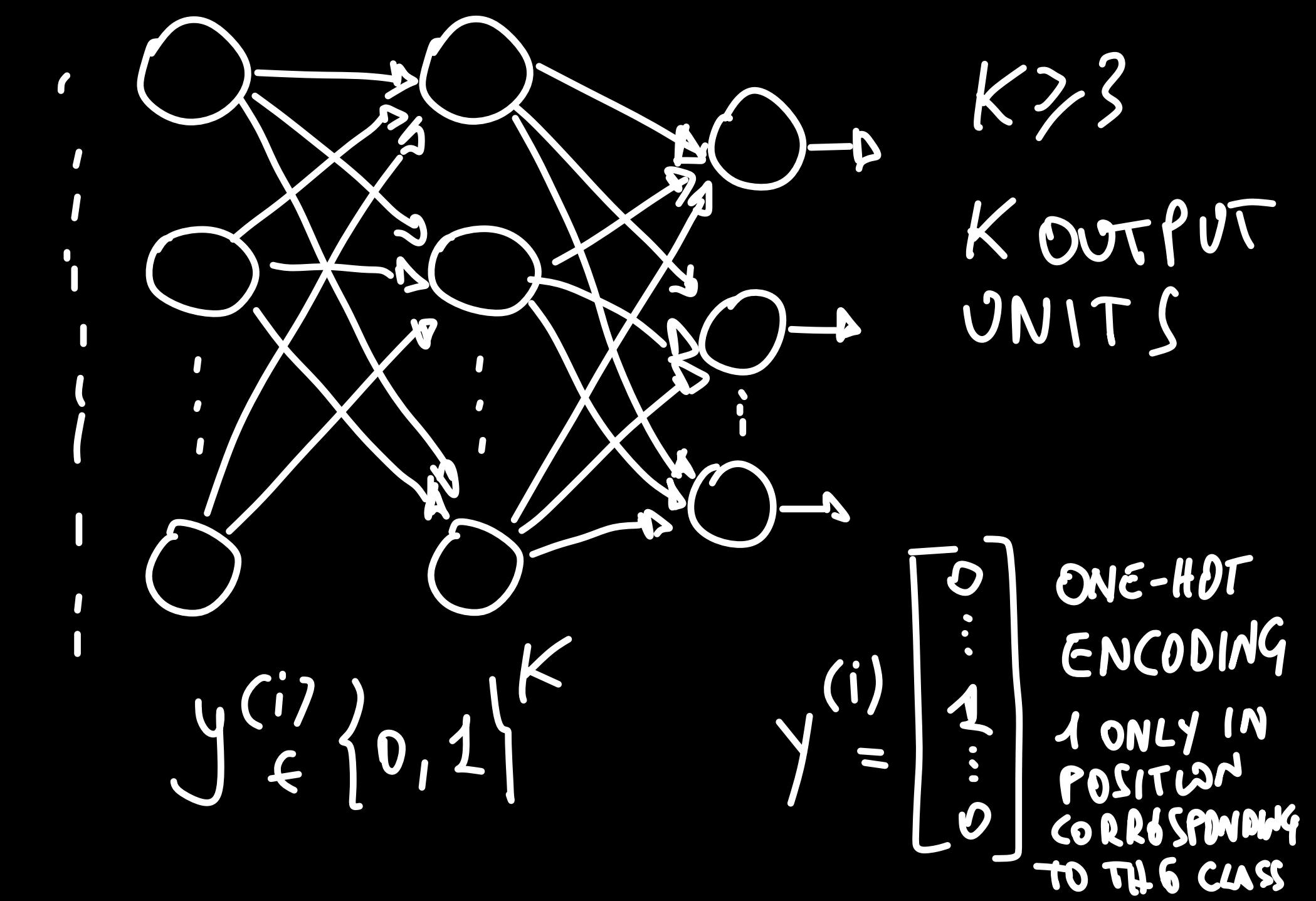
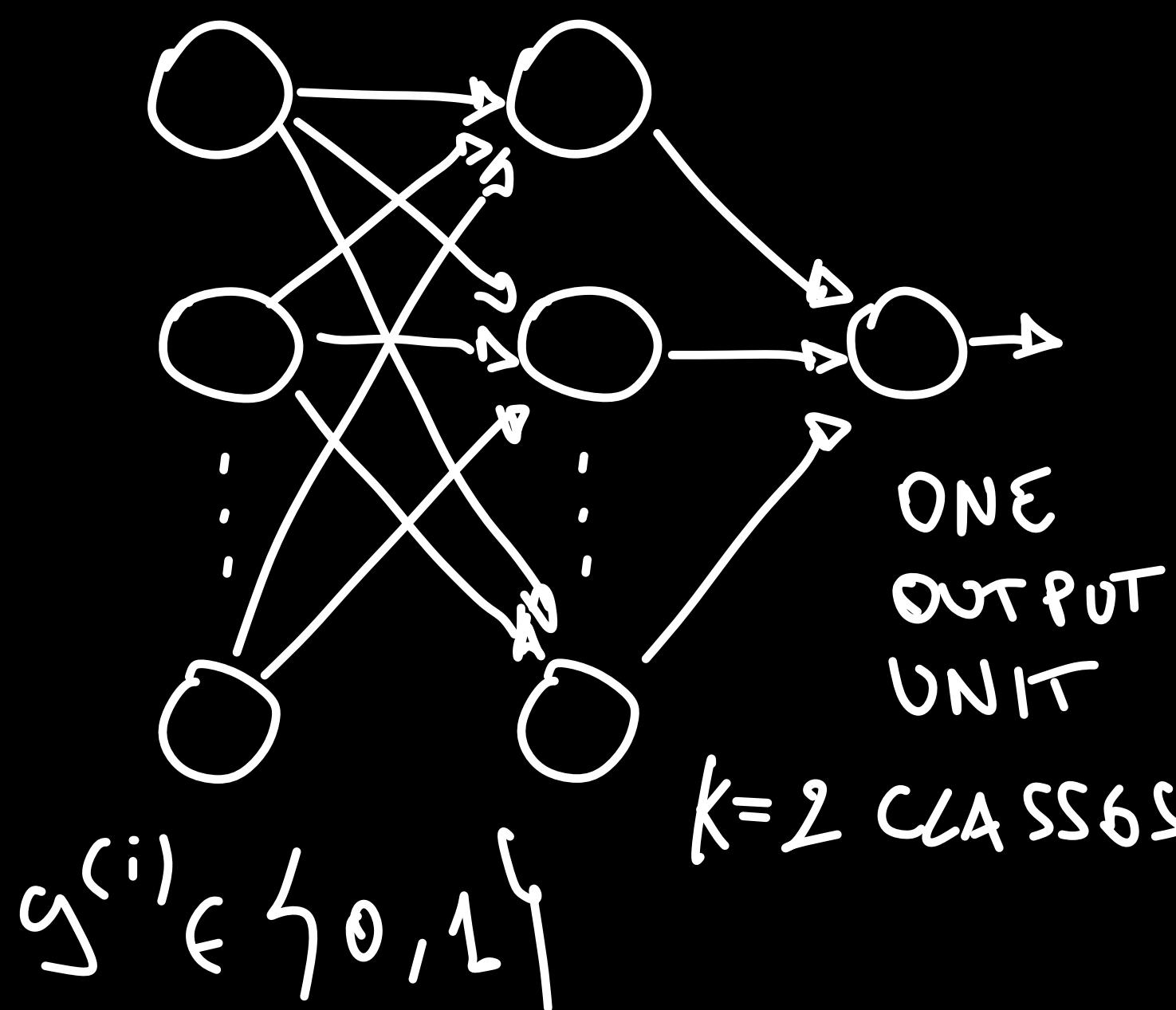


(c)

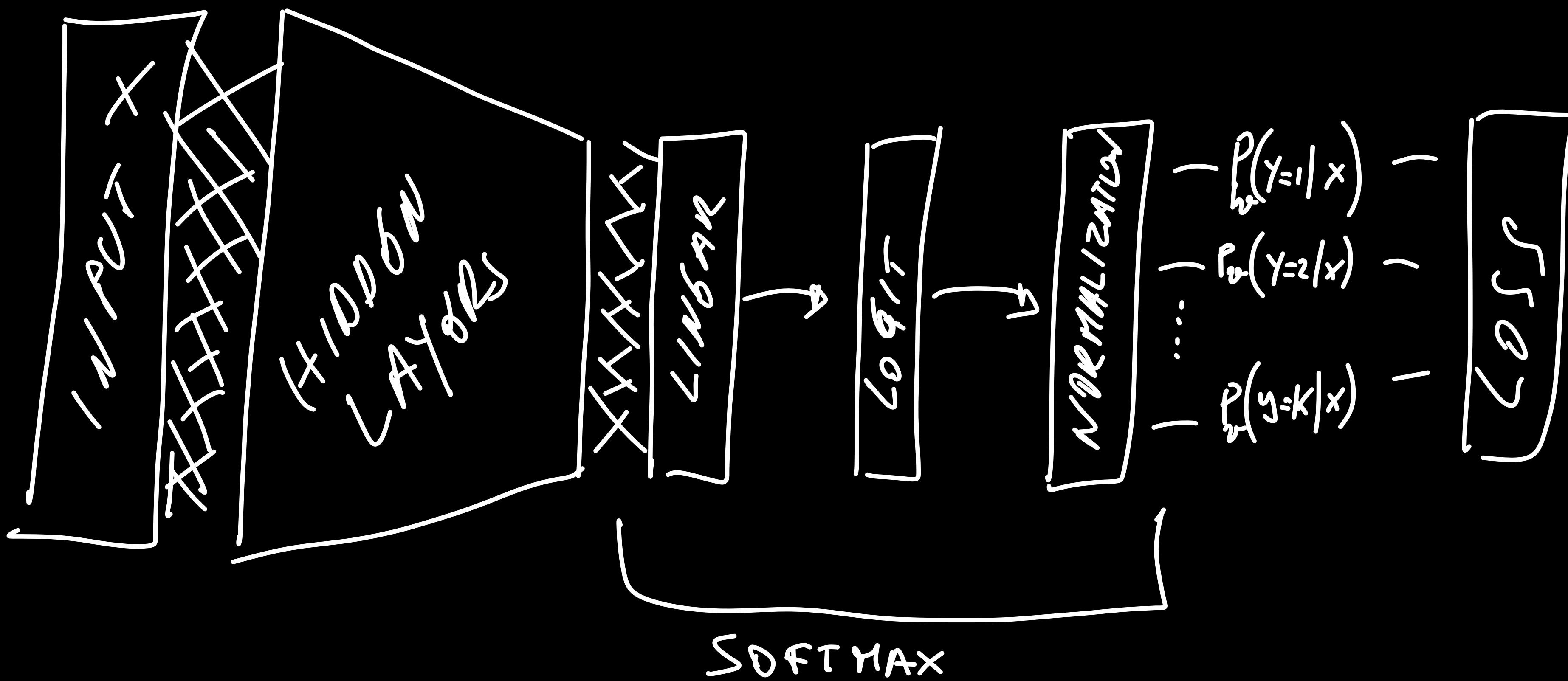


(d)

- NEURAL NETWORKS ALLOW TO TRAIN COMPLEX NON-LINEAR HYPOTHESES AND FEATURES SPACES GIVEN A TASK
- HOW TO USE FOR REGRESSION? JUST USE LINEAR REGRESSION IN THE LAST LAYER (AND THE CORRESPONDING LOSS FUNCTION)
- HOW TO EXTEND FOR MULTI-CLASS CLASSIFICATION?



NEURAL NETWORK "STANDARD" STRUCTURE



IS IT POSSIBLE TO USE FOR REGRESSION (NOT
CONSIDERING LOGIT AND NORMALIZATION)

NEURAL NETWORK - TRAINING & INFERENCE

- SUPPOSE WE HAVE FOUND AN ARCHITECTURE
- DURING TRAINING PHASES OUR GOAL IS TO LEARN THE BEST PARAMETERS

FOR OUR MODEL \Rightarrow WE NEED A LOSS

FUNCTION + AN ALGORITHM FOR LEARNING
THEN

- WHEN PARAMETERS HAVE BEEN OBTAINED WE USE THEM TO INFER ON NEW PATTERNS

COST FUNCTION (TO BE USED FOR CLASSIFICATION)

OUR GOAL IS TO FIND \hat{w} SUCH THAT

$$\hat{w} = \min (\hat{J}(w))$$

WHICH THE COST FUNCTION IS THE FOLLOWING

$$J(w) = -\frac{1}{m} \sum_{i=1}^m \sum_{c=1}^K \left[y_c^{(i)} \log \left(h_w(x^{(i)}) \right) + (1 - y_c^{(i)}) \log \left(1 - h_w(x^{(i)}) \right) \right]$$

↑ CROSS-ENTROPY FUNCTION

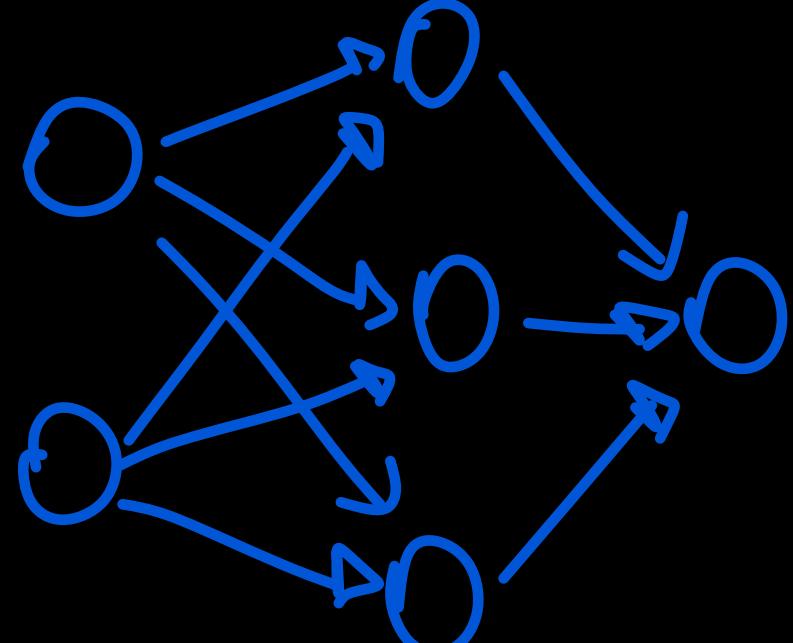
Σ over CROSS-ENTROPY OF EACH OUTPUT NEURON

TO REACH OUR GOAL WE SHOULD BE ABLE TO
EFFICIENTLY COMPUTE $\frac{\partial J(w)}{\partial w_j^{(i)}} \forall j, i, \ell$

NEURAL NETWORK AND OVERRFITTING

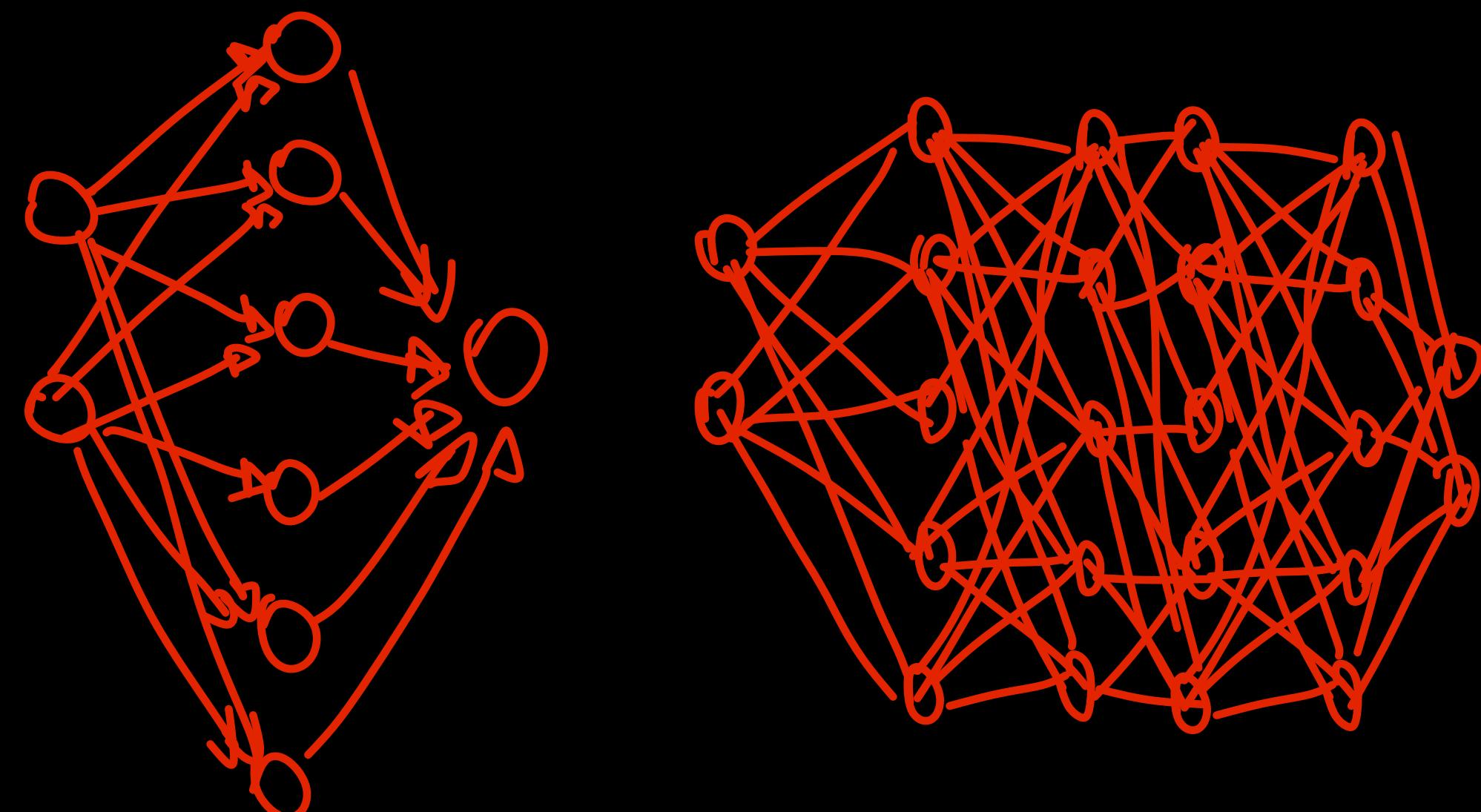
"SMALL" NEURAL NET

- FEWER PARAMETERS
- MORE PRONE TO UNDERFITTING
- COMPUTATIONALLY CHEAPER



"LARGE" NEURAL NET

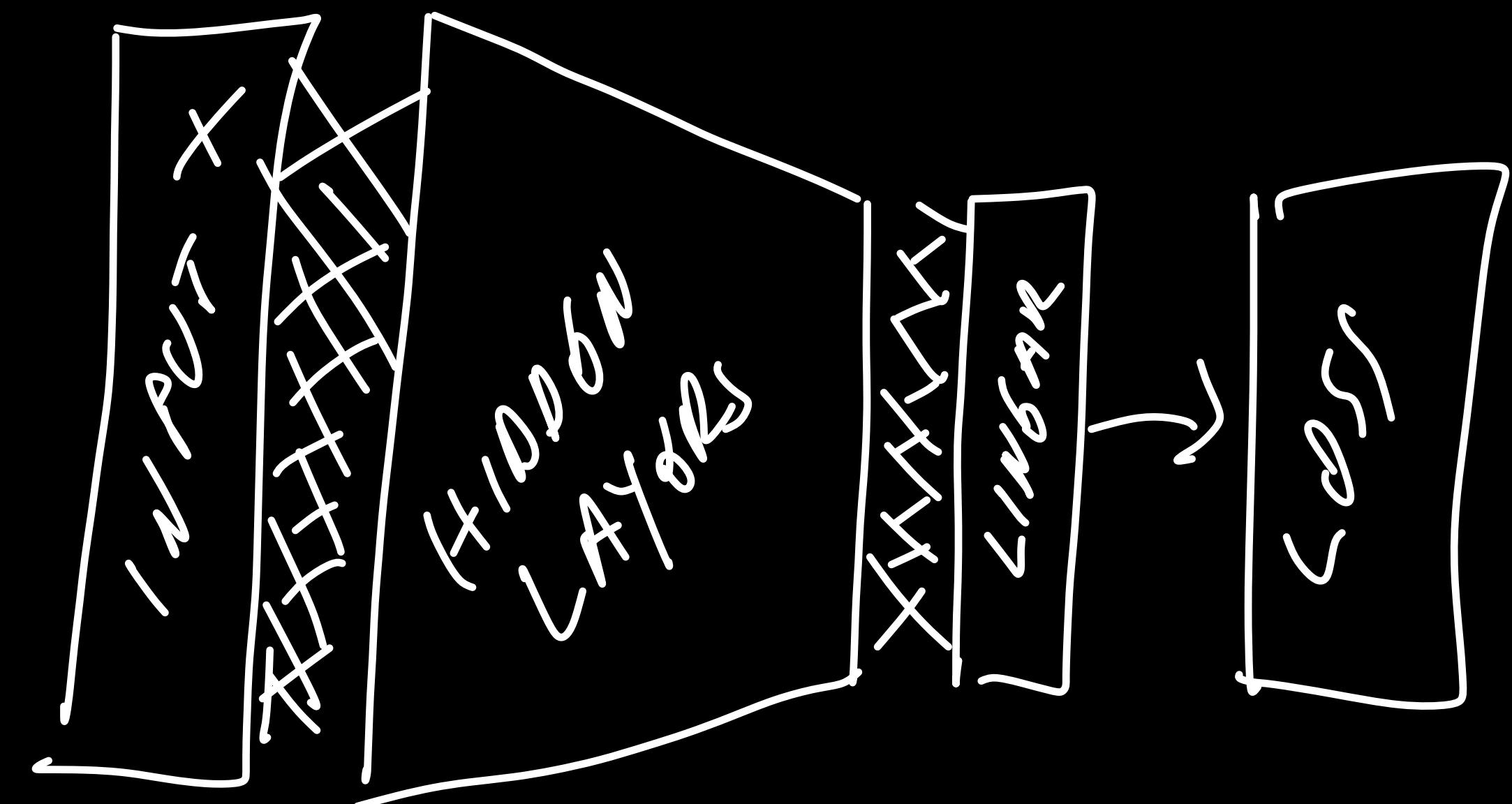
- MORE PARAMETERS
- MORE PRONE TO OVERRFITTING
- COMPUTATIONALLY EXPENSIVE



USE REGULARIZATION (λ) TO ADDRESS OVERRFITTING ON LARGE NET

How TO USE FOR REGRESSION?

- WE NEED TO CHANGE THE COST FUNCTION
- EACH OUTPUT NODE IS A LINEAR REGRESSION
- COST FUNCTION SHOULD TAKE INTO ACCOUNT
THE OUTPUT OF ALL REGRESSIONS

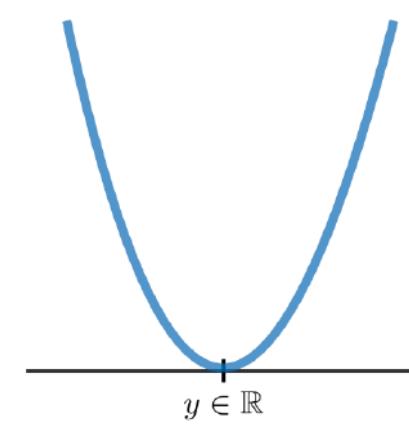
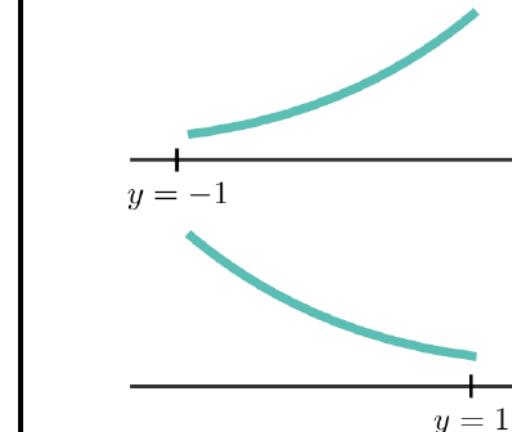
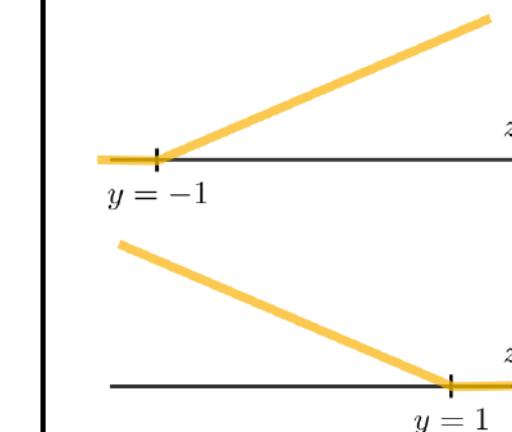
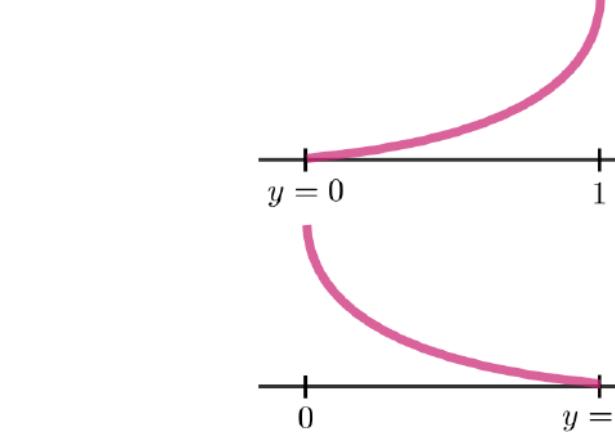


Neural Network at Glance

$h_{\theta}(x^{(i)})$

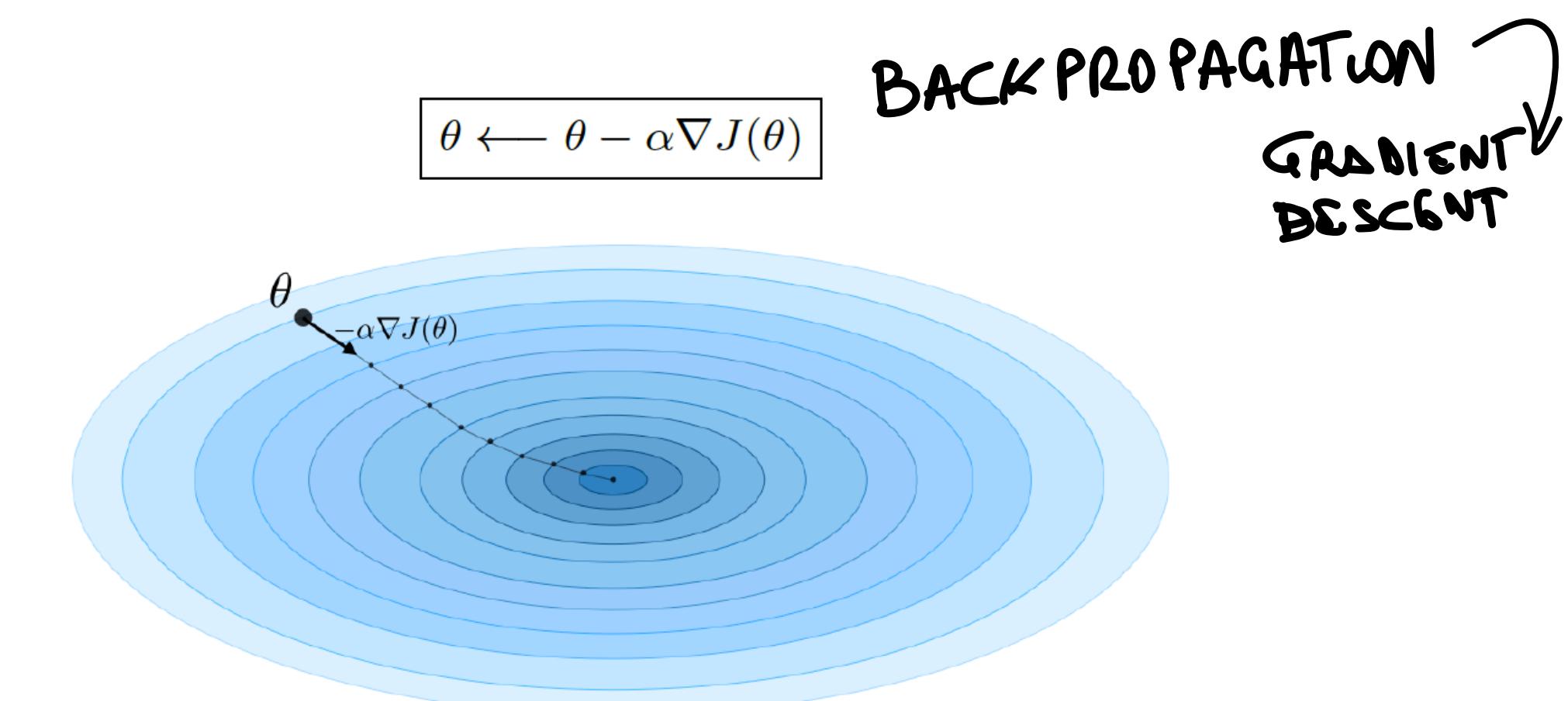
	Regression	Classifier
Outcome	Continuous	Class

$$L : (z, y) \in \mathbb{R} \times Y \longmapsto L(z, y) \in \mathbb{R}$$

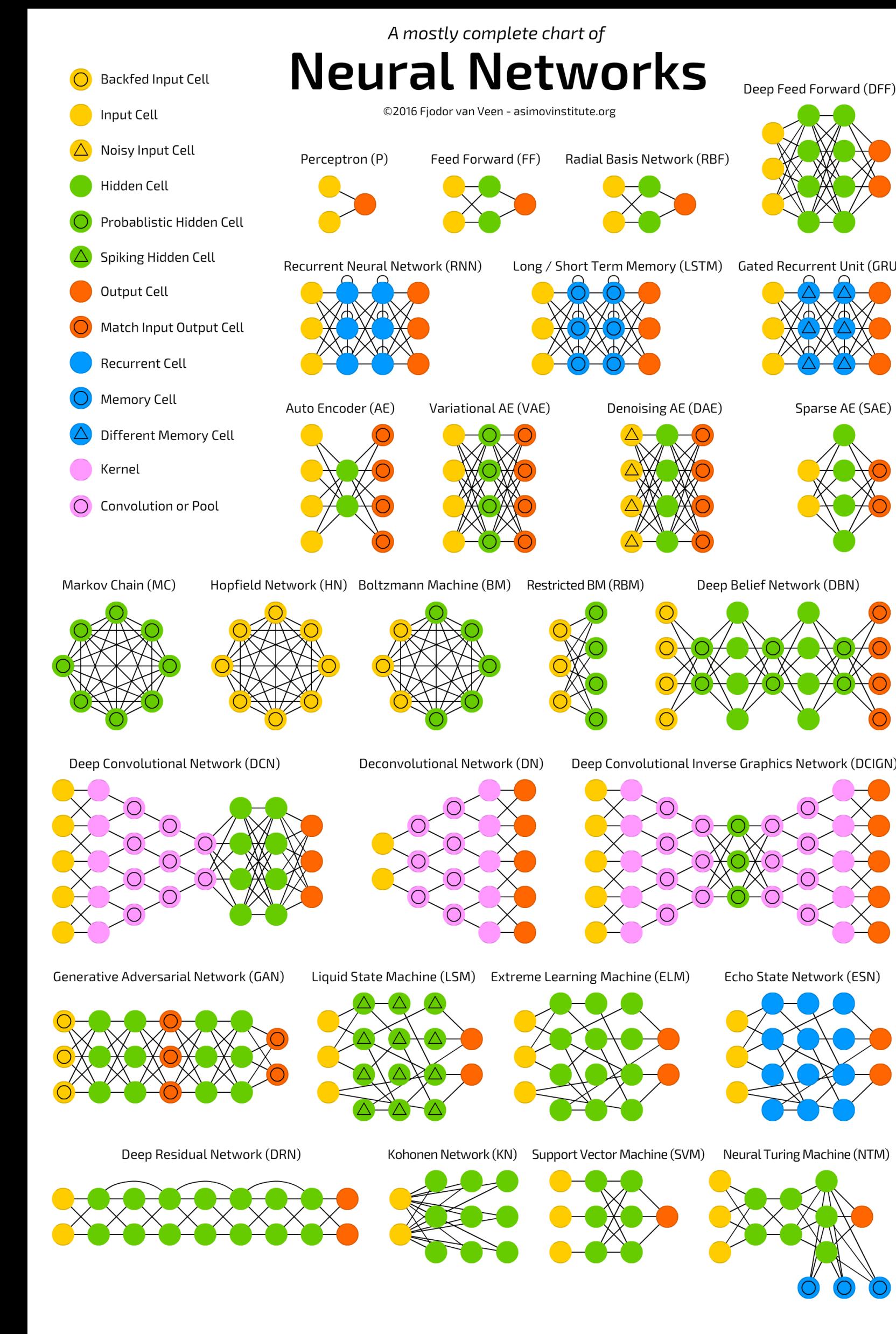
Least squared	Logistic	Hinge	Cross-entropy
$\frac{1}{2}(y - z)^2$	$\log(1 + \exp(-yz))$	$\max(0, 1 - yz)$	$-(y \log(z) + (1 - y) \log(1 - z))$
			
Linear regression	Logistic regression	SVM	Neural Network

$$J(\theta) = \sum_{i=1}^m L(h_{\theta}(x^{(i)}), y^{(i)})$$

$$\theta \leftarrow \theta - \alpha \nabla J(\theta)$$



Neural Network Zoo

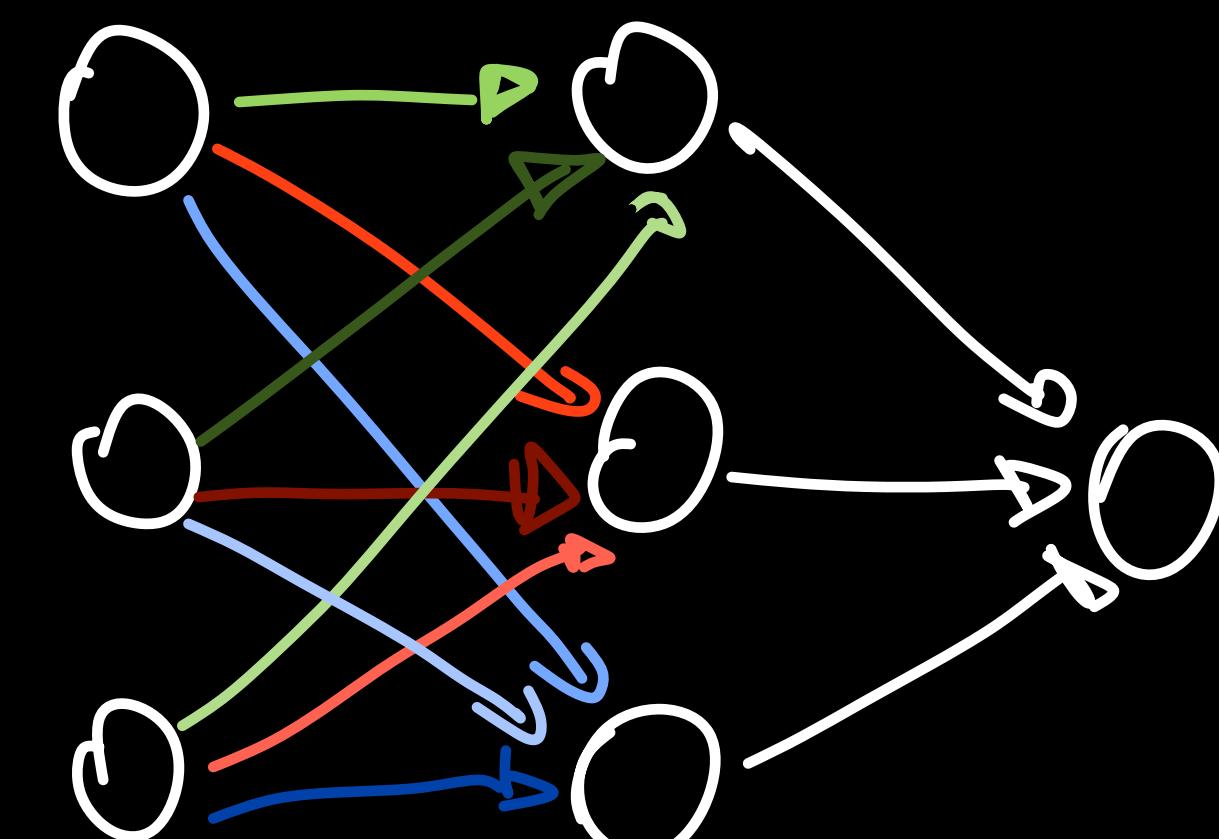
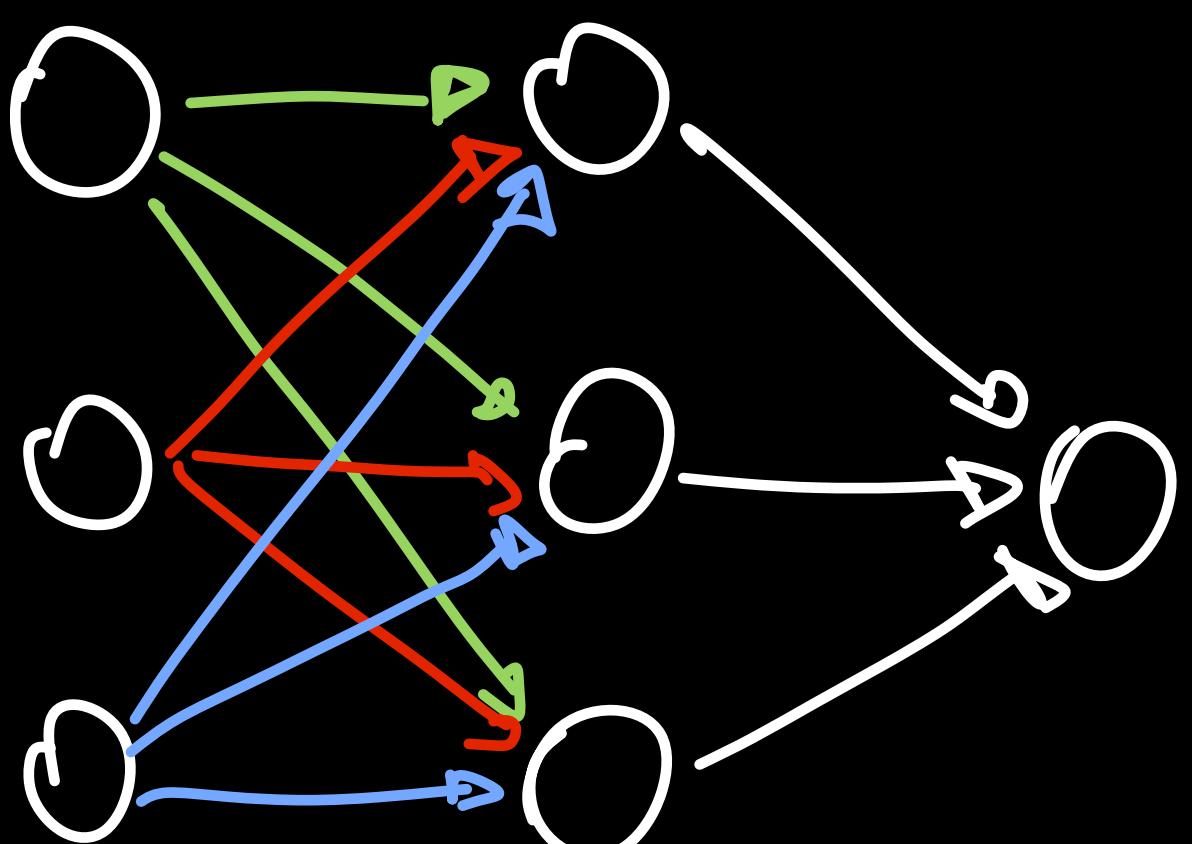


PARAMETERS INITIALIZATION

HOW TO INITIALIZE THE PARAMETERS MATRIX \mathcal{W} ?

SYMMETRY BREAKING: THE PARAMETERS DEPEND ON N_S

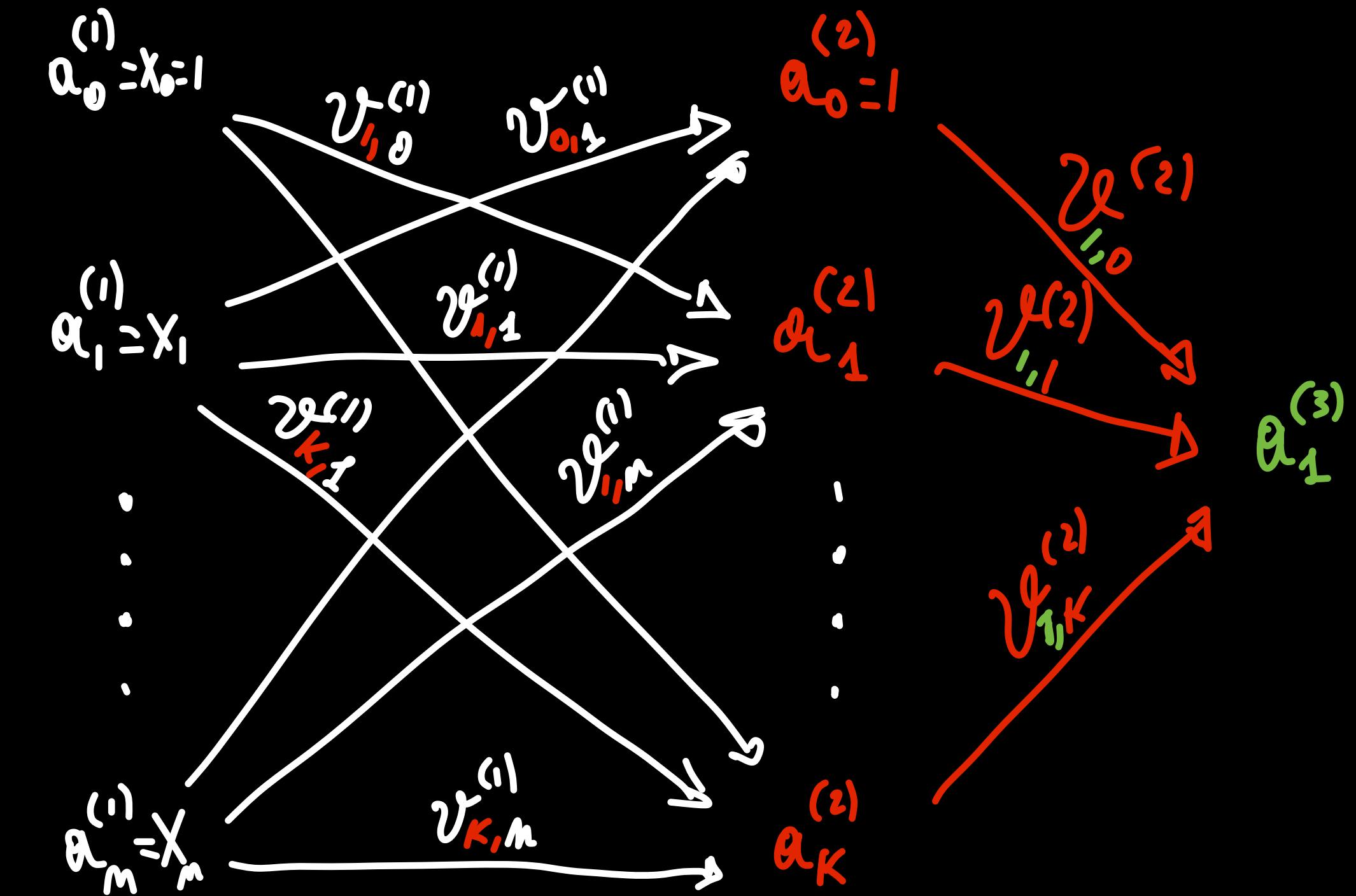
FROM THE SAME NODES HAVE TO BE INITIALIZED WITH
DIFFERENT VALUES (SMALL: $v_{j,i}^{(l)}$ USUALLY $\in [-\varepsilon, \varepsilon]$ WITH
 $\varepsilon = 1, 3 -$)



PROBLEM: HOW TO
LEARN PARAMETERS
IN A NEURAL NETWORK
WITH A NUMBER
OF HIDDEN LAYERS ≥ 1 ?
HOW TO COMPUTE THE GRADIENT
(DERIVATIVES) WITH RESPECT TO PARAMETERS?

How to LEARN PARAMETERS?

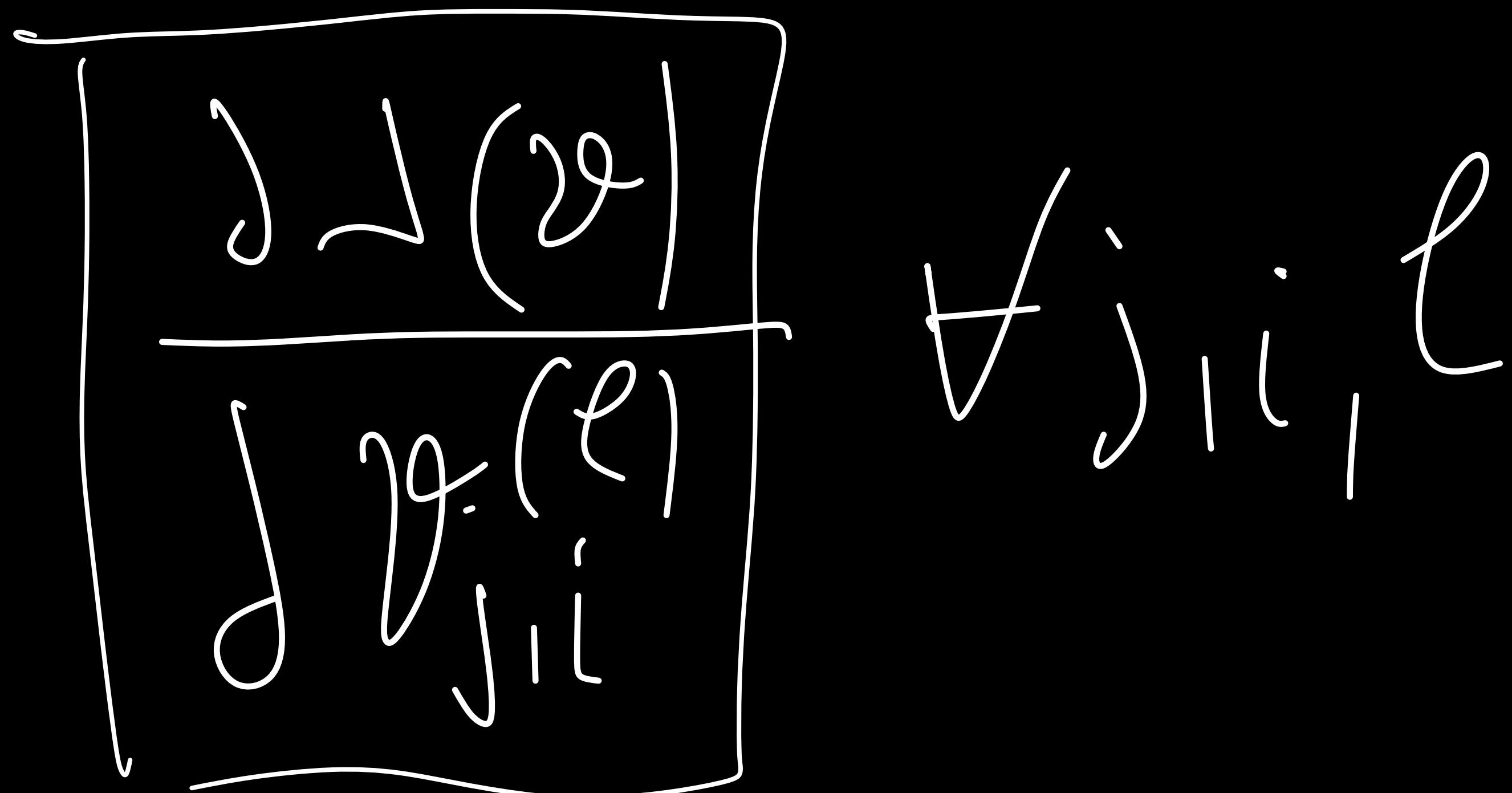
- ONE NEURON ✓
 \Rightarrow GRADIENT DESCENT
 (EVOLUTION OF ROSENBLATT LEARNING ALGORITHM)
- SOFTMAX ✓
 \Rightarrow GRADIENT DESCENT
 LOSS FUNCTION ACTS ON 1 LAYER
- NN WITH HIDDEN LAYER? ✓



VALUES OF HIDDEN NODES ARE NOT KNOWN IN ADVANCE \exists ARE NOT GIVEN WITH TRAINING SET

\Rightarrow GRADIENT DESCENT SHOULD BE REVISING \Rightarrow BACKPROPAGATION

HENCE OUR PROBLEM IS HOW TO COMPUTE



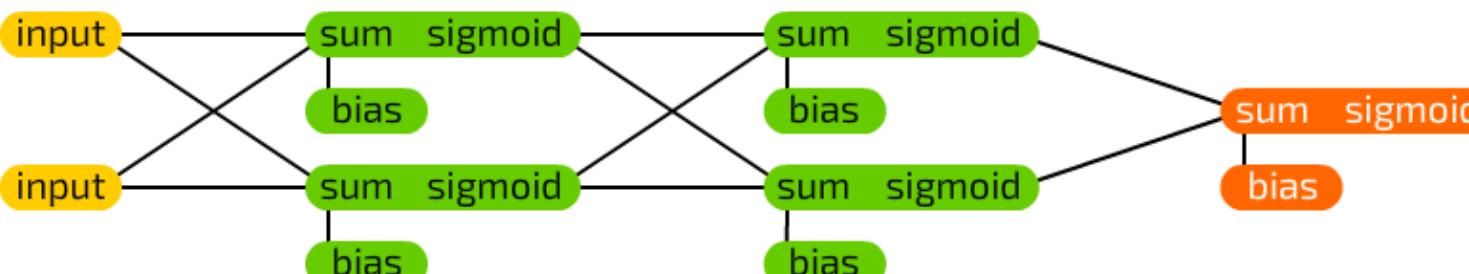
SOLUTION IS BASED ON THE FACT THAT
NEURAL NETWORK ARE COMPUTATIONAL GRAPHS
AND WE CAN USE CHAIN RULE TO DRIVE \Rightarrow BACKPROPAGATION

COMPUTATIONAL GRAPHS

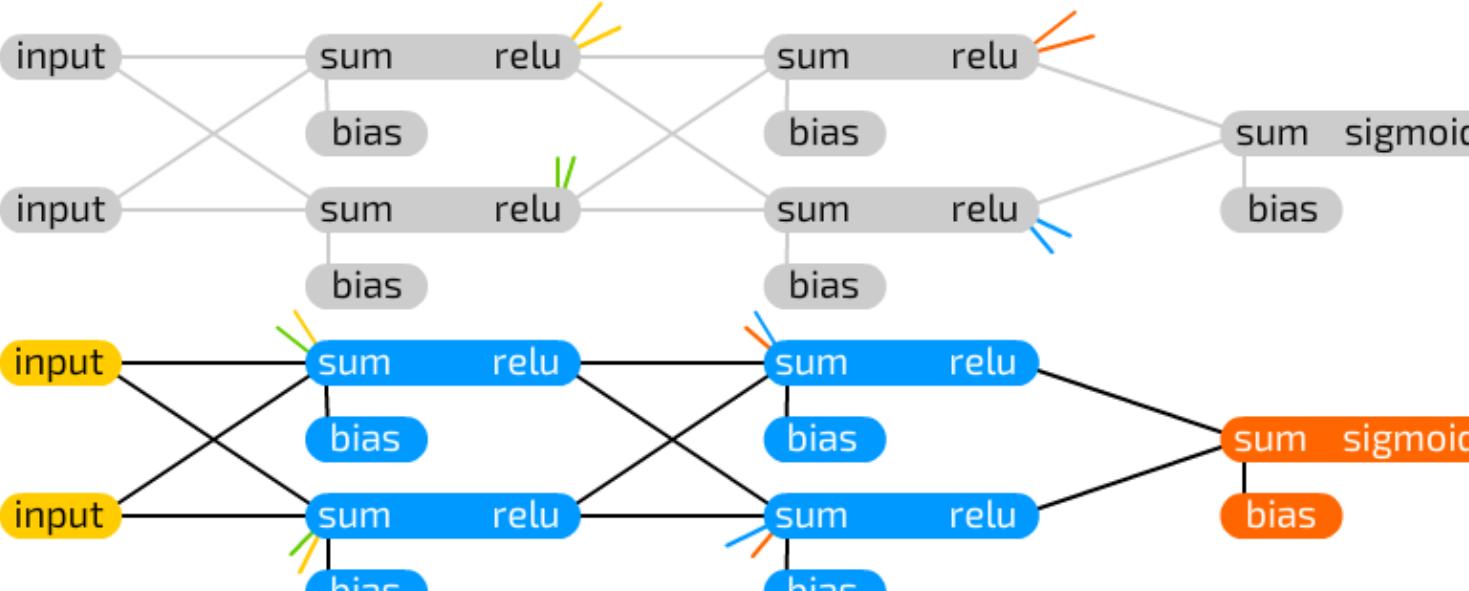
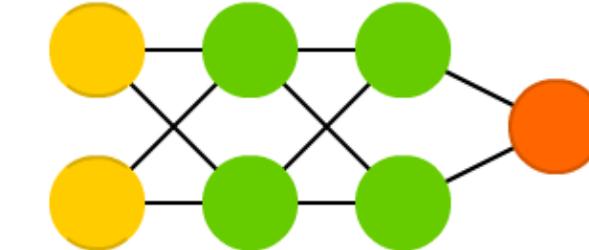
An informative chart to build

Neural Network Graphs

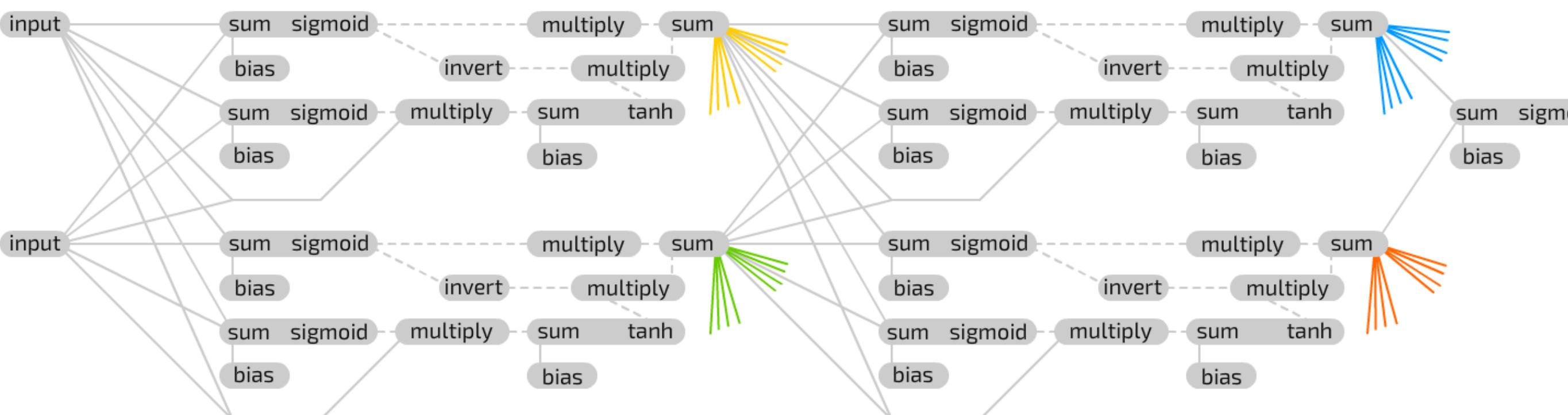
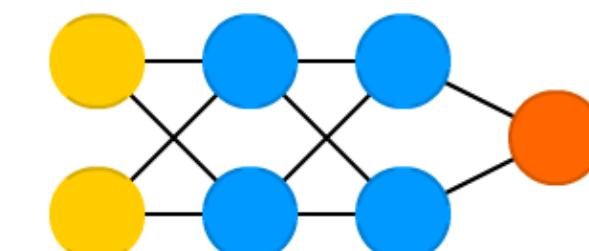
©2016 Fjodor van Veen - asimovinstitute.org



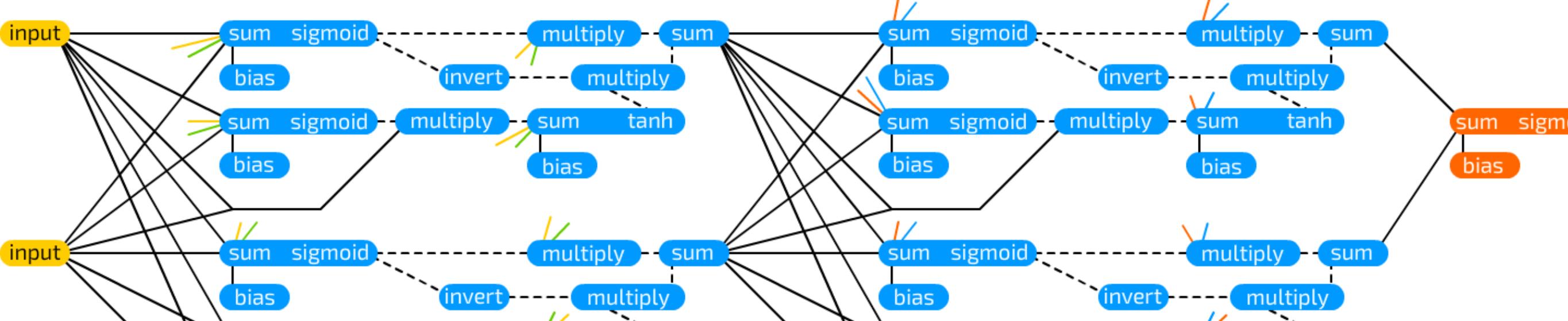
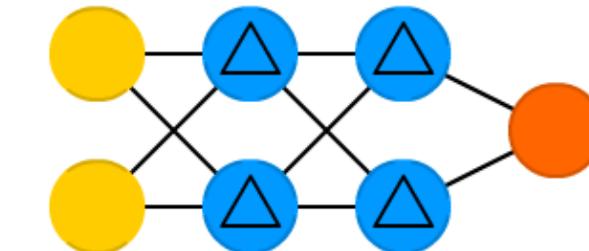
Deep Feed Forward Example



Deep Recurrent Example
(previous iteration)



Deep GRU Example
(previous iteration)



Deep GRU Example