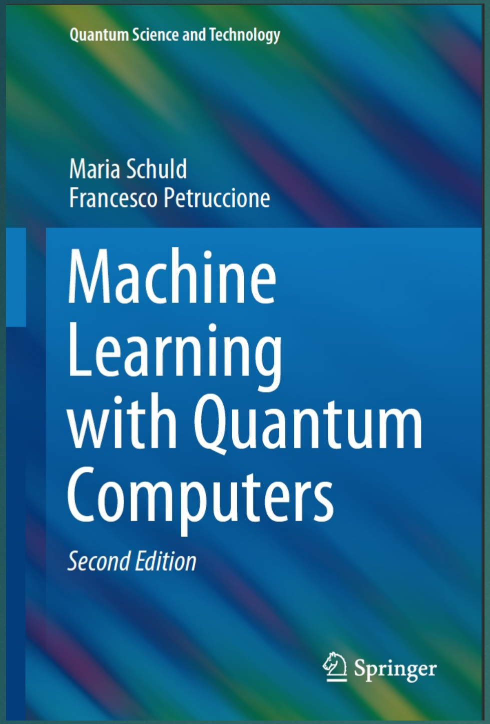
Ciência de Dados Quântica 22/23

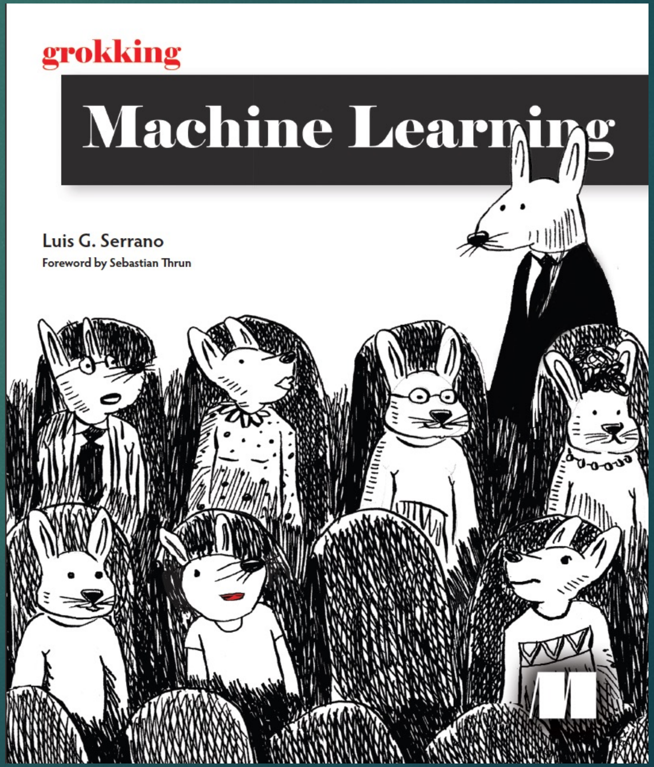
Classical Machine Learning:
Neural Networks

LUÍS PAULO SANTOS

Material de Consulta

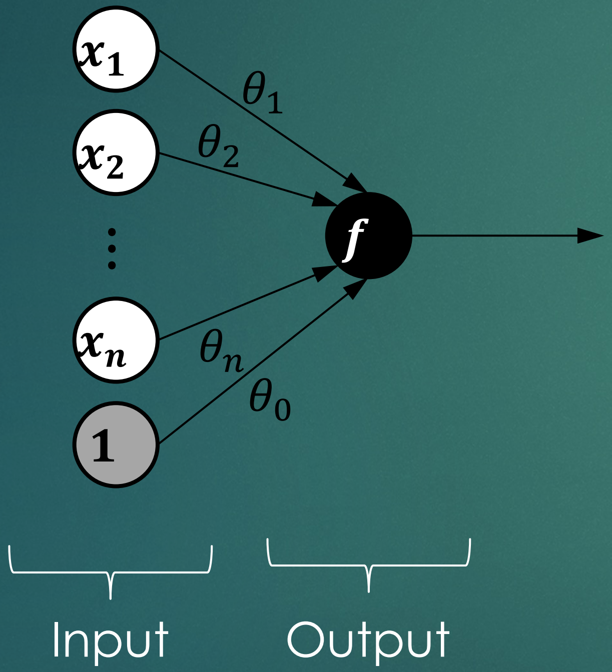


- Maria Schuld, Francesco Petruccione
  "***Machine Learning with Quantum Computers***"
  Cap. 2



- Luís G. Serrano
  "**grokking *Machine Learning***"
  Cap. 10 e apêndice B

Perceptrons

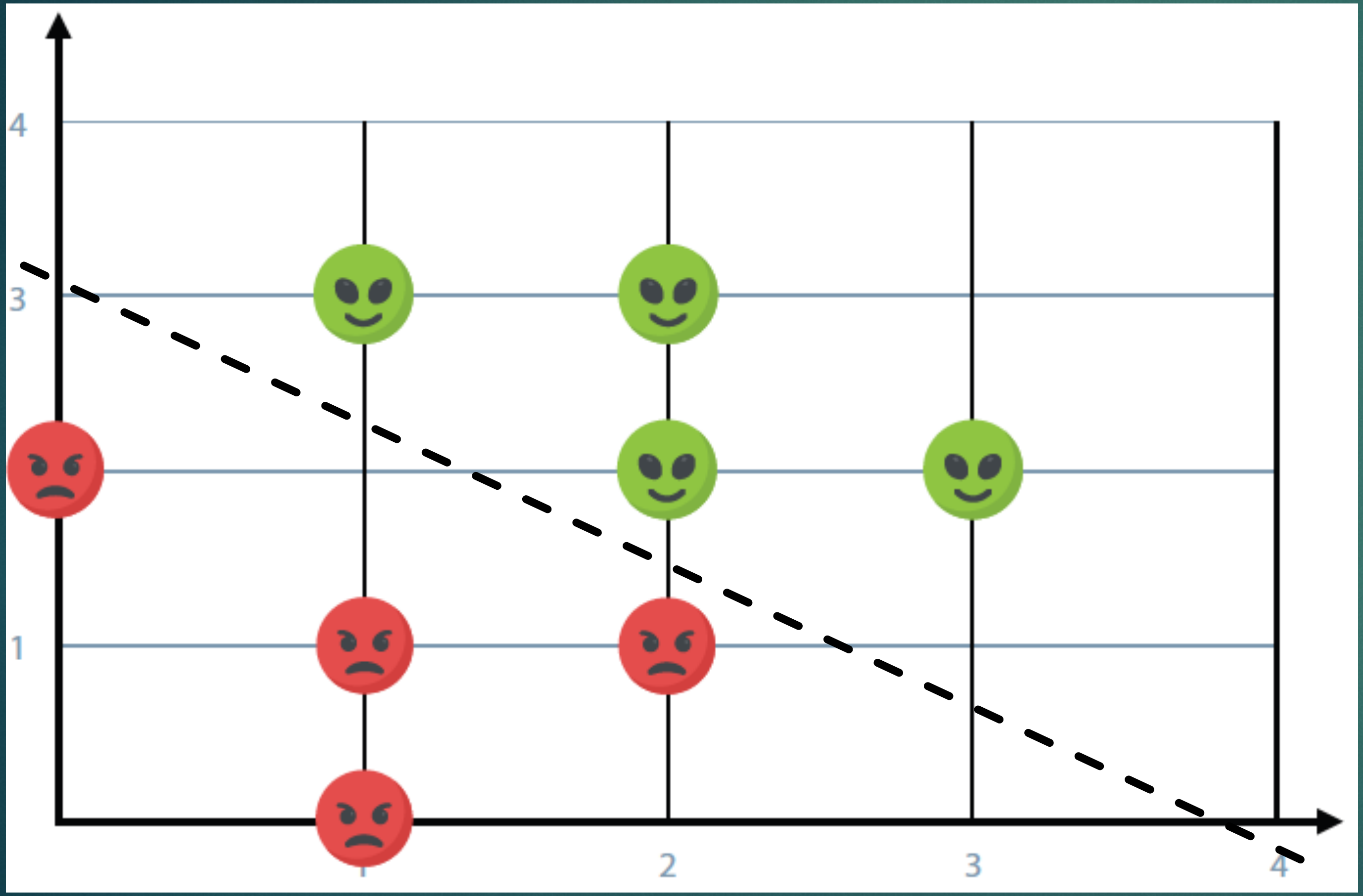


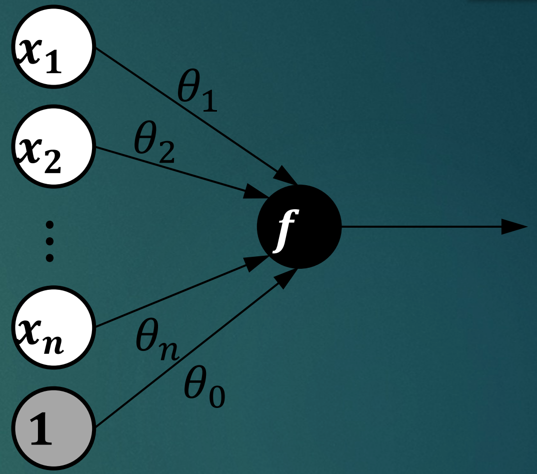
Perceptron Classifier:
$f_{activation}(\quad) = step(\quad)$

$$\hat{y}(x) = f_{activation}\left(\theta_0 + \sum_{i=1}^{n} \theta_i x_i\right)$$

Logistic Classifier:
$f_{activation}(\quad) = \sigma(\quad)$

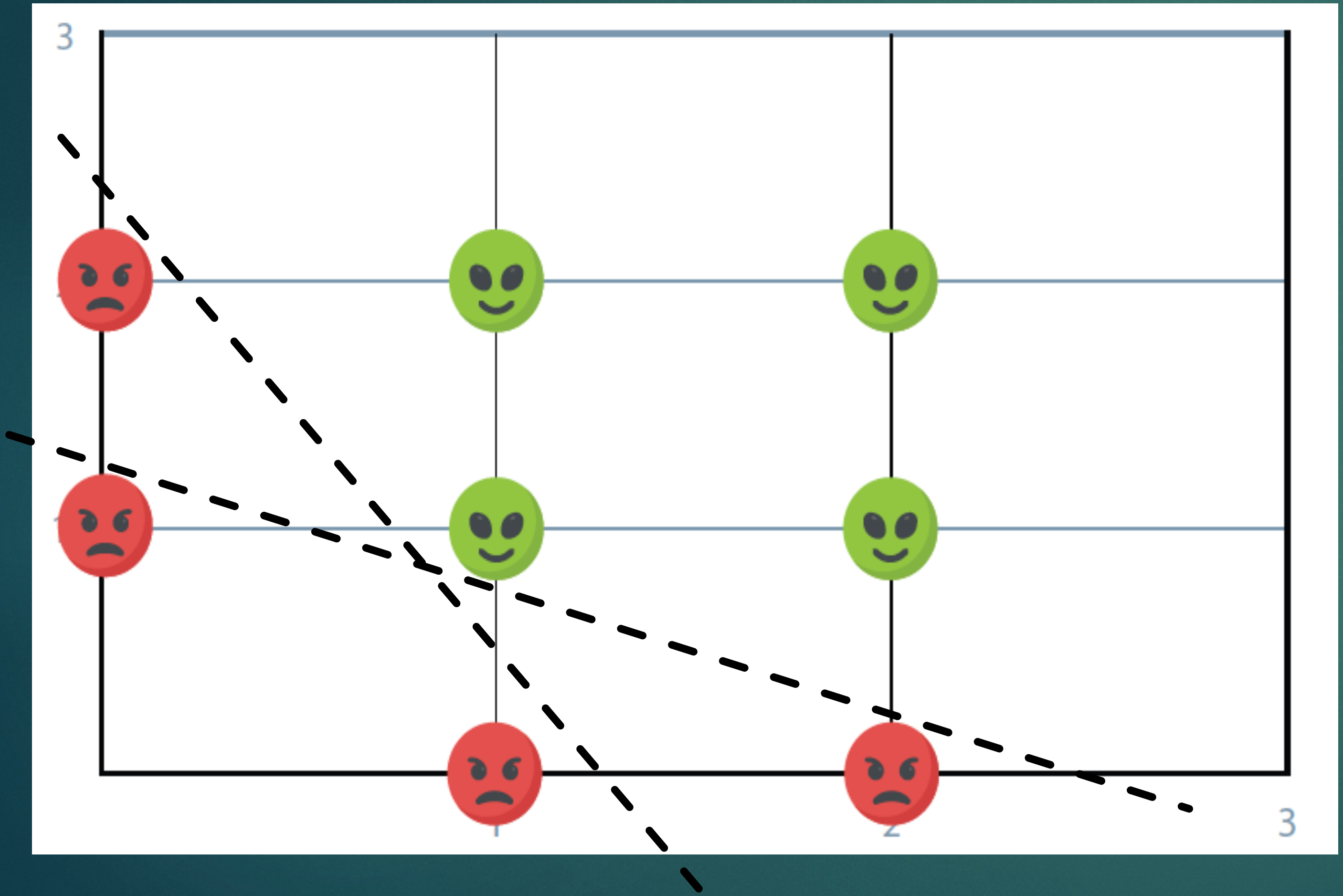
Neural Networks





$$\hat{y}(x) = \sigma\left(\theta_0 + \sum_{i=1}^{n} \theta_i x_i\right)$$

Neural Networks

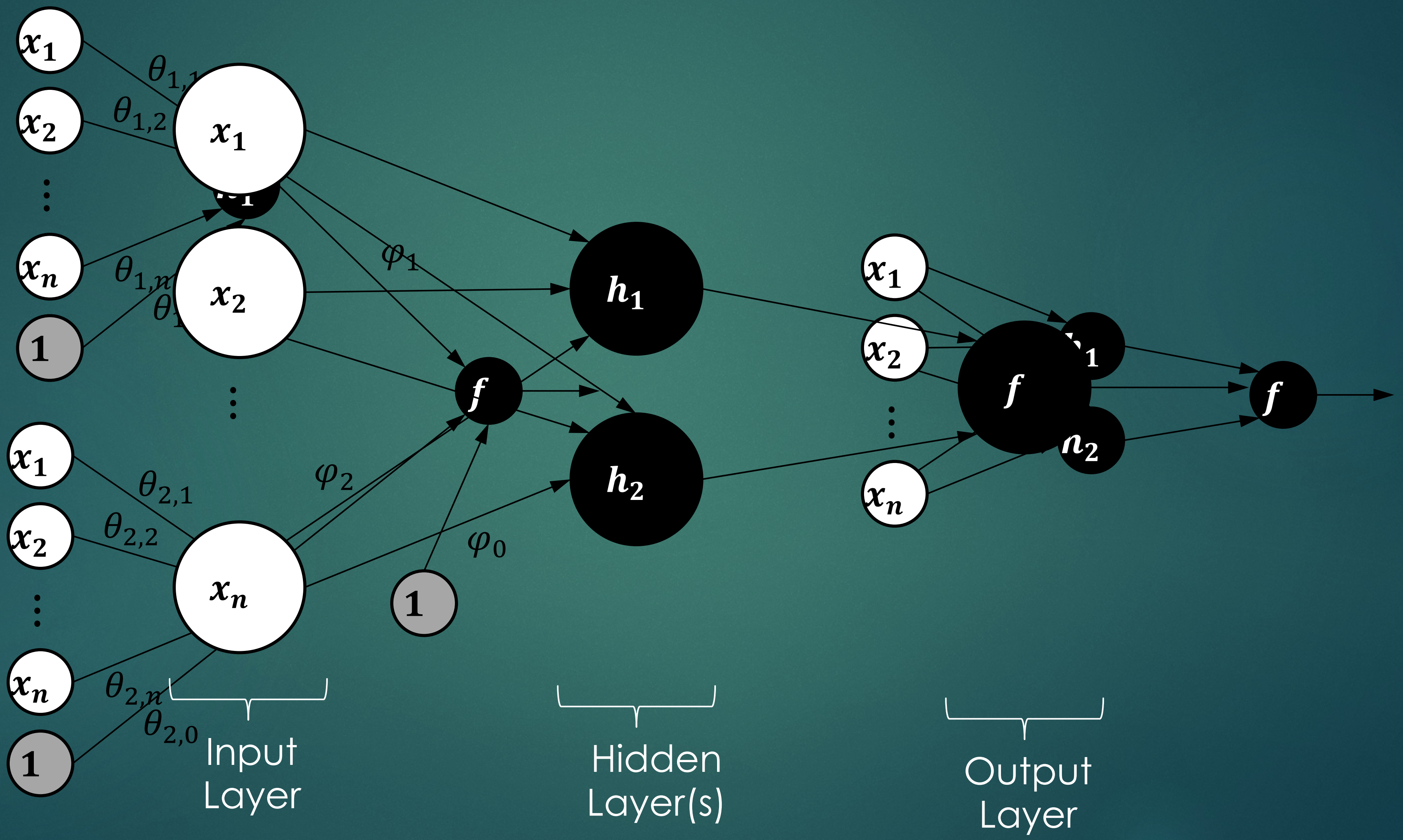


$$h_1(x) = \sigma\left(\theta_{1,0} + \sum_{i=1}^{n} \theta_{1,i} x_i\right)$$

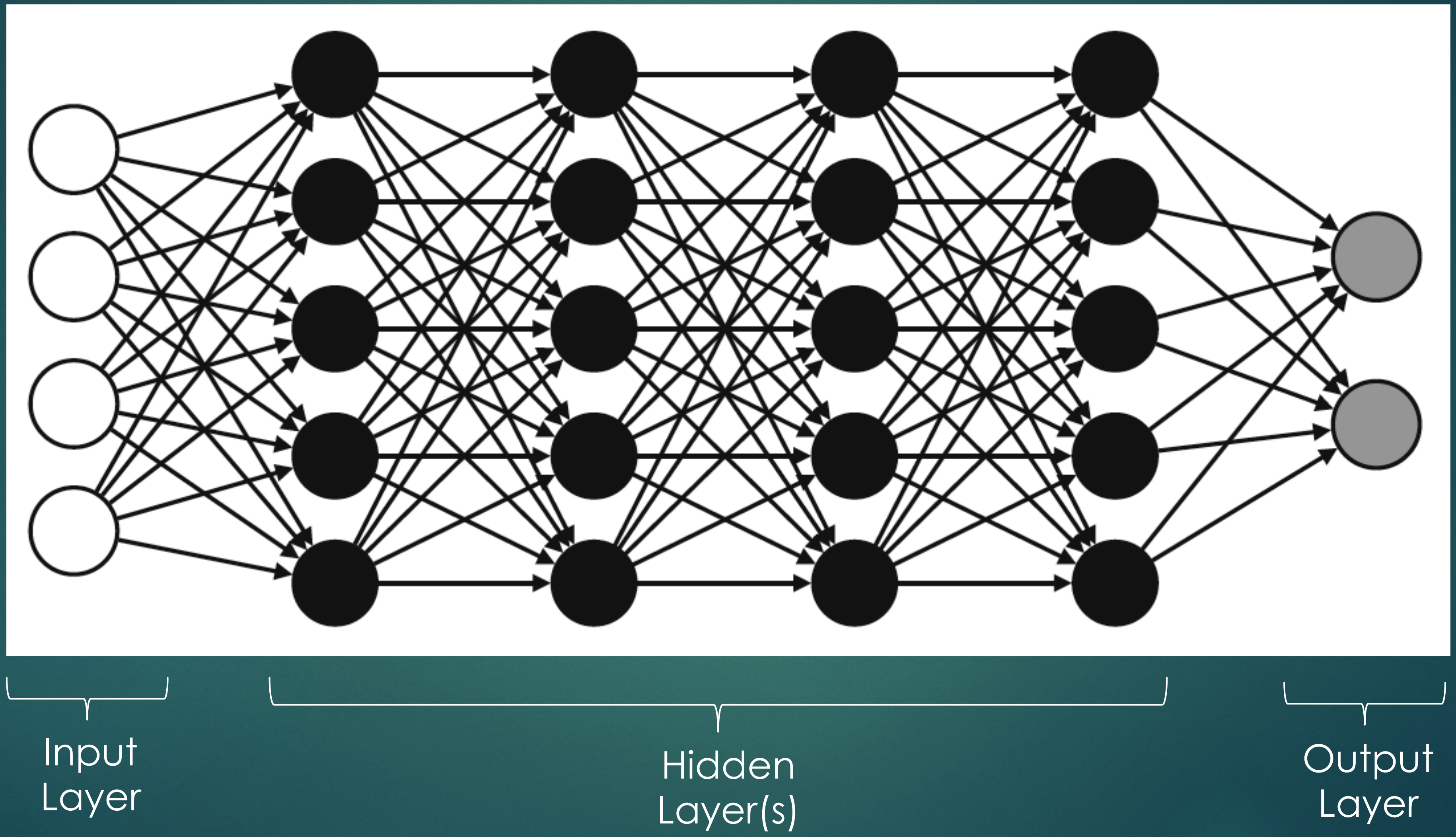
$$h_2(x) = \sigma\left(\theta_{2,0} + \sum_{i=1}^{n} \theta_{2,i} x_i\right)$$

$$\hat{y}(x) = \sigma\left(\varphi_0 + \sum_{i=1}^{H} \varphi_i h_i(x)\right)$$

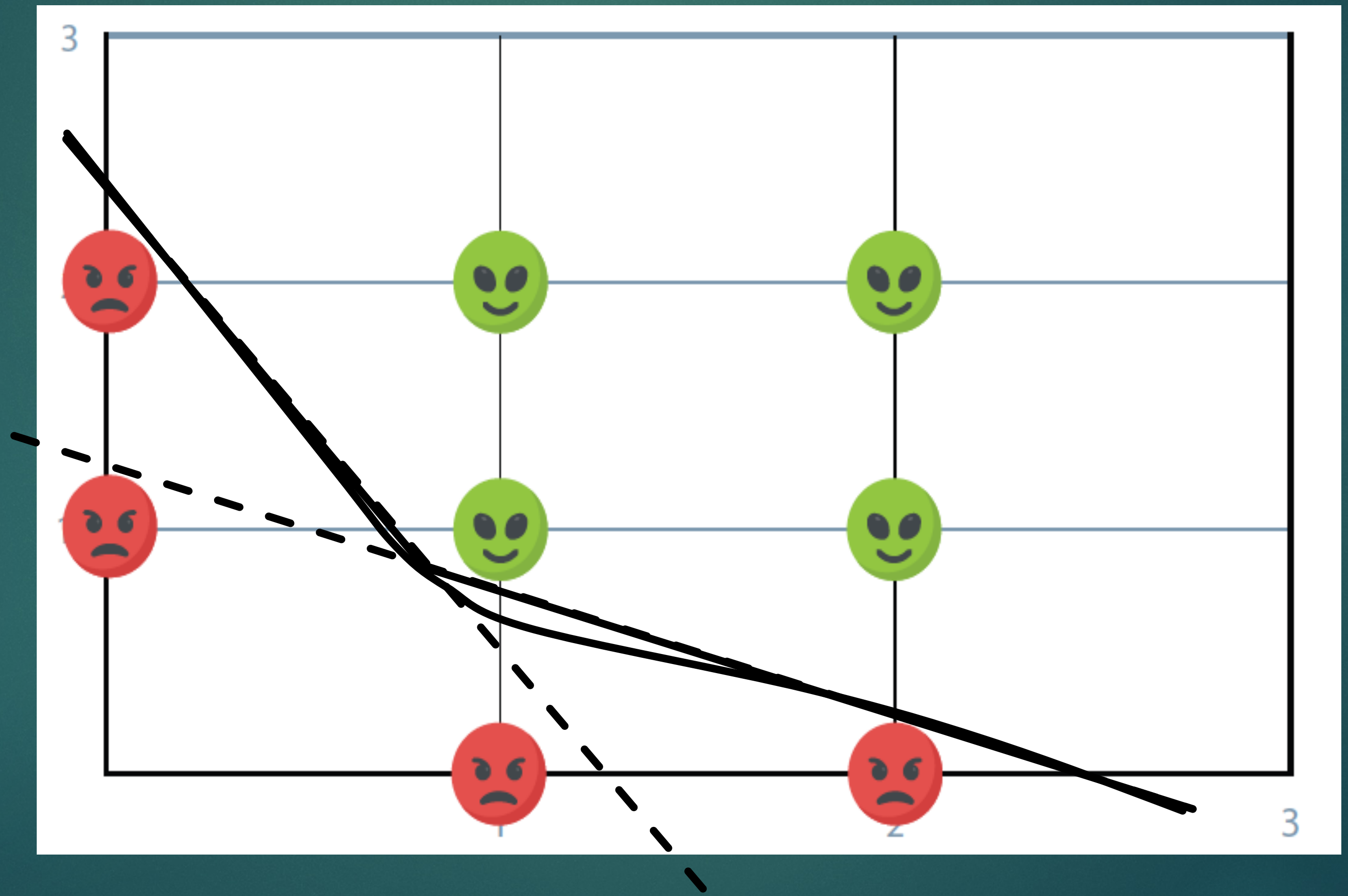
Neural Networks



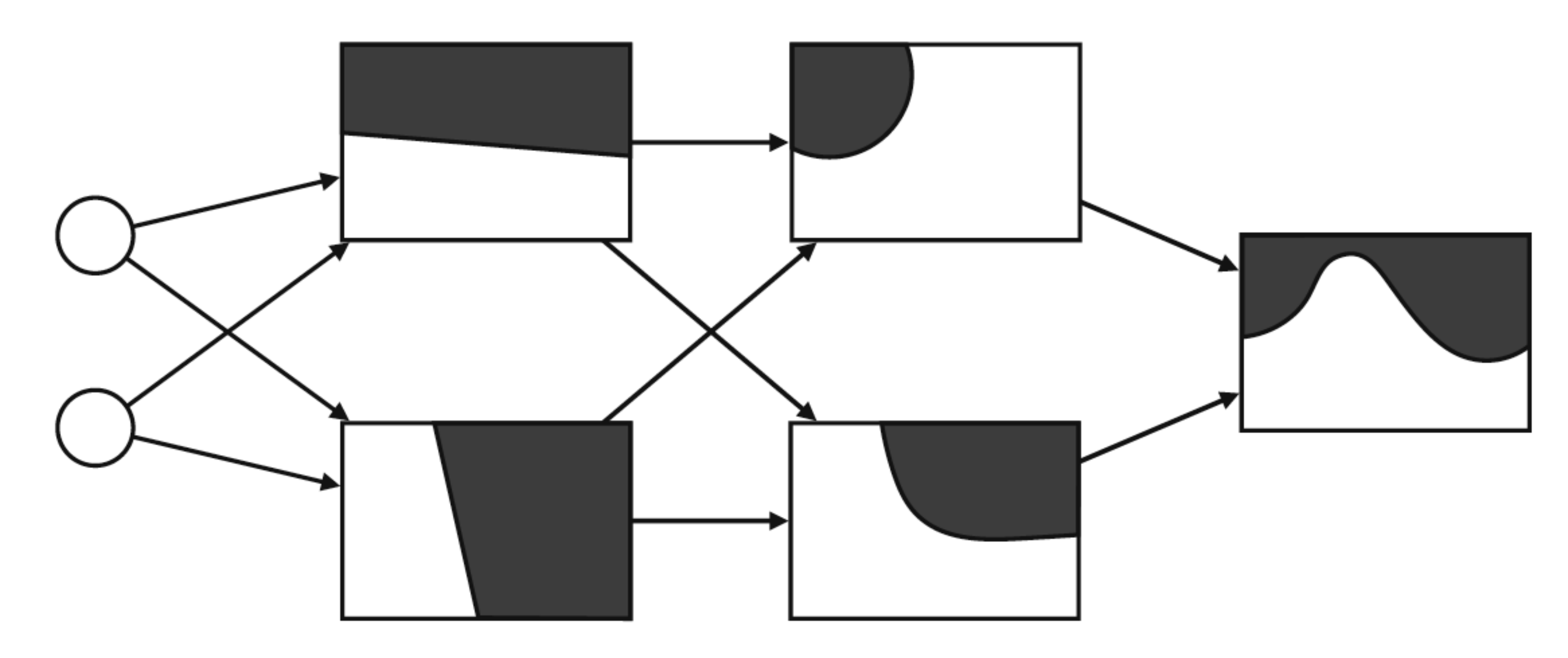
Fully connected feed forward NN



Neural Networks boundaries



Neural Networks boundaries



Universal Approximation Theorem

Neural Networks, Vol. 2, pp. 359–366, 1989
0893-6080/89 $3.00 + .00

ORIGINAL CONTRIBUTION

Multilayer Feedforward Networks are Universal Approximators

KURT HORNIK
Technische Universität Wien

MAXWELL STINCHCOMBE AND HALBERT WHITE
University of California, San Diego

(*Received* 16 *September* 1988; *revised and accepted* 9 *March* 1989)

Abstract—*This paper rigorously establishes that standard multilayer feedforward networks with as few as one hidden layer using arbitrary squashing functions are capable of approximating any Borel measurable function from one finite dimensional space to another to any desired degree of accuracy, provided sufficiently many hidden units are available. In this sense, multilayer feedforward networks are a class of universal approximators.*

"…**multilayer feed-forward networks** with as few as one hidden layer **are universal approximators.**"

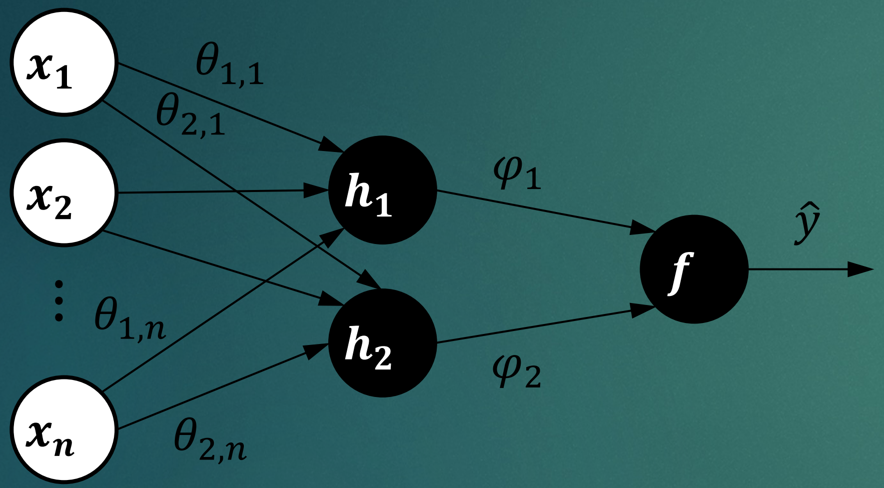
[https://en.wikipedia.org/wiki/Universal_approximation_theorem]

Error Function: log loss

$$LL(y,\hat{y}) = -y\,ln(\hat{y}) - (1-y)ln(1-\hat{y}) = \begin{cases} 0 & \leftarrow y = \hat{y} \\ -ln(\hat{y}) & \leftarrow \hat{y} \neq y \land y = 1 \\ -ln(1-\hat{y}) & \leftarrow \hat{y} \neq y \land y = 0 \end{cases}$$

- Review: why the log loss?
  - logarithms suit probabilities well:
    the logarithm of a product is the sum of the individual factors logarithms
  - the logarithm has an analytical simple derivative
  - the value of the log loss grows with the difference $y - \hat{y}$
  - The log loss is related to cross entropy from information theory

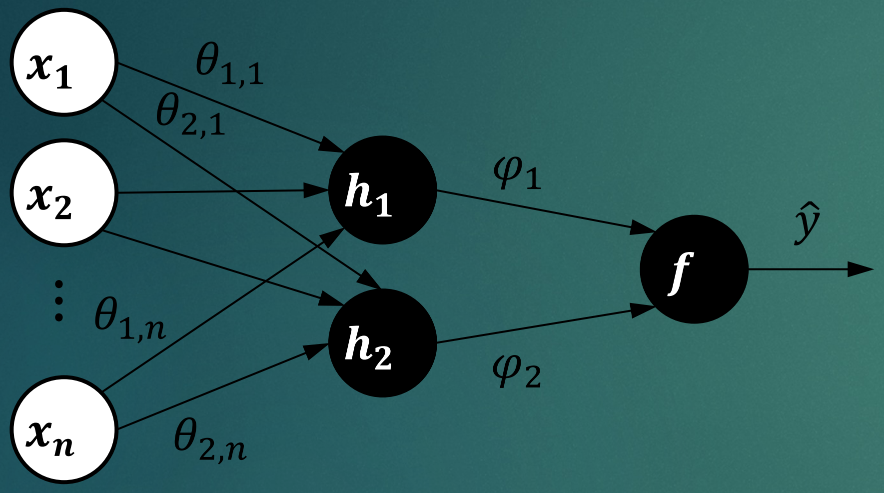
Gradient Descent



$$\Theta^{(t+1)} = \Theta^{(t)} - \eta \nabla_\Theta LL(\hat{y}(x), y)$$

$$\nabla_\Theta = \begin{pmatrix} \frac{\partial}{\partial \varphi_0} \\ \vdots \\ \frac{\partial}{\partial \varphi_H} \\ \frac{\partial}{\partial \theta_{1,0}} \\ \vdots \\ \frac{\partial}{\partial \theta_{1,n}} \\ \vdots \\ \frac{\partial}{\partial \theta_{H,0}} \\ \vdots \\ \frac{\partial}{\partial \theta_{H,n}} \end{pmatrix}$$

Gradient Descent: back propagation



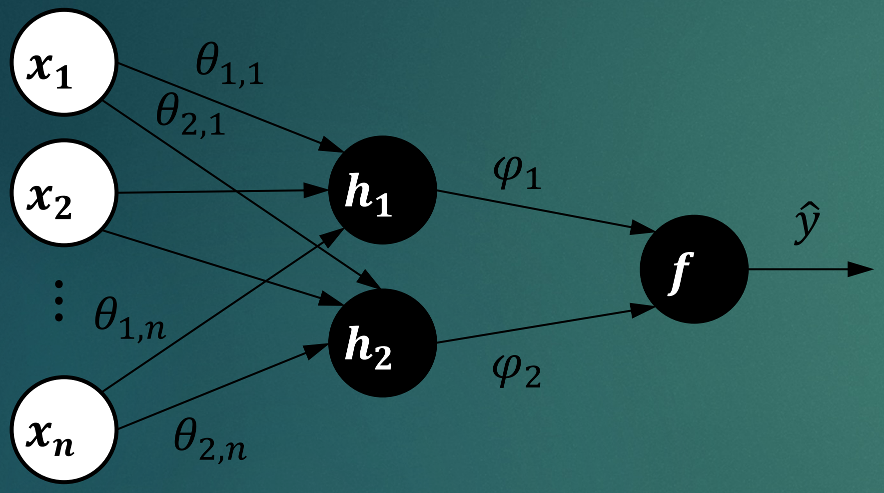
$$\hat{y} = \sigma\left(\varphi_0 + \sum_{j=1}^{H} \varphi_j h_j\right)$$

$$s = \varphi_0 + \sum_{j=1}^{H} \varphi_j h_j \qquad \hat{y} = \sigma(s)$$

$$h_j = \sigma\left(\theta_{j,0} + \sum_{i=1}^{n} \theta_{j,i} x_i\right)$$

$$r_j = \theta_{j,0} + \sum_{i=1}^{n} \theta_{j,i} x_i \qquad h_j = \sigma(r_j)$$

Gradient Descent: back propagation



$$LL(y, \hat{y}) = -y\, ln(\hat{y}) - (1 - y) ln(1 - \hat{y})$$

$$\frac{\partial ln(u)}{\partial u} = \frac{1}{u}$$

$$\frac{\partial \sigma(u)}{\partial u} = \sigma(u)\big(1 - \sigma(u)\big)$$

$$\frac{\partial u}{\partial v} = \frac{\partial t}{\partial v}\frac{\partial u}{\partial t}$$

Gradient Descent: back propagation

$$\frac{\partial LL}{\partial \hat{y}} = -\frac{y}{\hat{y}} - \frac{-(1-y)}{(1-\hat{y})} = \frac{-(y-\hat{y})}{\hat{y}(1-\hat{y})}$$

$$\frac{\partial \hat{y}}{\partial s} = \sigma(s)\big(1-\sigma(s)\big) = \hat{y}(1-\hat{y})$$

$$\frac{\partial LL}{\partial s} = \frac{\partial LL}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial s} = \frac{-(y-\hat{y})}{\hat{y}(1-\hat{y})}\hat{y}(1-\hat{y}) = -(y-\hat{y})$$

<u>Remember:</u>

$$LL(y,\hat{y}) = -y\, ln(\hat{y}) - (1-y)ln(1-\hat{y})$$

$$s = \varphi_0 + \sum_{j=1}^{H} \varphi_j h_j \qquad \hat{y} = \sigma(s)$$

Gradient Descent: back propagation

$$\frac{\partial s}{\partial \varphi_{j>0}} = h_j \qquad \frac{\partial s}{\partial \varphi_0} = 1$$

$$\frac{\partial s}{\partial h_j} = \varphi_j$$

$$\frac{\partial h_j}{\partial r_j} = \sigma(r_j)\left(1 - \sigma(r_j)\right) = h_j(1 - h_j)$$

$$\frac{\partial r_j}{\partial \theta_{j,i>0}} = x_i \qquad \frac{\partial r_j}{\partial \theta j_{,0}} = 1$$

<u>Remember:</u>

$$s = \varphi_0 + \sum_{j=1}^{H} \varphi_j h_j$$

$$h_j = \sigma\left(\theta_{j,0} + \sum_{i=1}^{n} \theta_{j,i} x_i\right)$$

$$r_j = \theta_{j,0} + \sum_{i=1}^{n} \theta_{j,i} x_i \qquad h_j = \sigma(r_j)$$

Gradient Descent: back propagation

By applying the previous results:

$$\frac{\partial LL}{\partial s} = -(y - \hat{y})$$

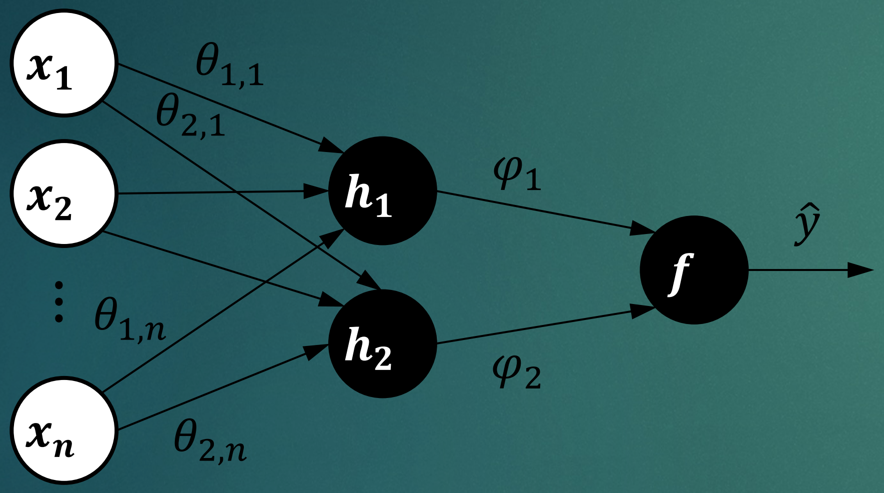
$$\frac{\partial LL}{\partial \varphi_{j>0}} = \frac{\partial s}{\partial \varphi_{j>0}} \frac{\partial LL}{\partial s} = -(y - \hat{y})h_j$$

$$\frac{\partial LL}{\partial \varphi_0} = \frac{\partial s}{\partial \varphi_0} \frac{\partial LL}{\partial s} = -(y - \hat{y})$$

$$\frac{\partial LL}{\partial \theta_{j,i>0}} = \frac{\partial r_j}{\partial \theta_{j,i>0}} \frac{\partial h_j}{\partial r_j} \frac{\partial s}{\partial h_j} \frac{\partial LL}{\partial s} = -(y - \hat{y})\varphi_j h_j \left(1 - h_j\right) x_i$$

$$\frac{\partial LL}{\partial \theta_{j,0}} = \frac{\partial r_j}{\partial \theta_{j,0}} \frac{\partial h_j}{\partial r_j} \frac{\partial s}{\partial h_j} \frac{\partial LL}{\partial s} = -(y - \hat{y})\varphi_j h_j \left(1 - h_j\right)$$

Gradient Descent: back propagation



$$\Theta^{(t+1)} = \Theta^{(t)} - \eta \nabla_{\Theta} LL(\hat{y}(x), y)$$

$$\varphi_{j>0}^{(t+1)} = \varphi_{j>0}^{(t)} - \eta[-(y - \hat{y})h_j]$$

$$\varphi_0^{(t+1)} = \varphi_0^{(t)} - \eta[-(y - \hat{y})]$$

$$\theta_{j,i>0}^{(t+1)} = \theta_{j,i>0}^{(t)} - \eta[-(y - \hat{y})\varphi_j h_j (1 - h_j) x_i]$$

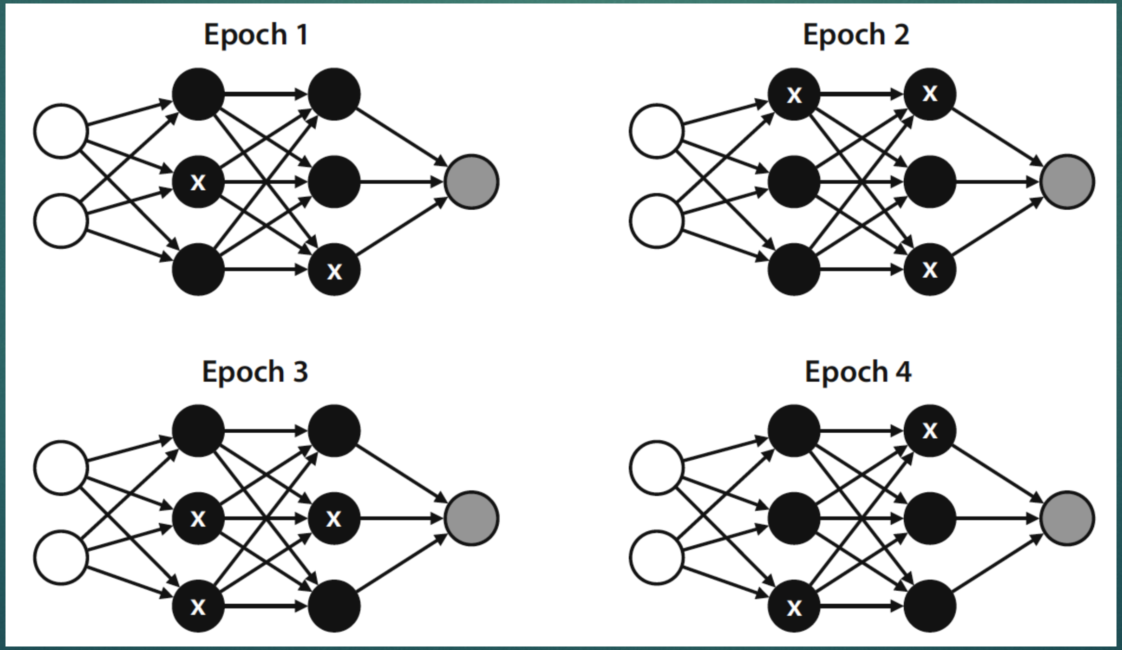
$$\theta_{j,0}^{(t+1)} = \theta_{j,0}^{(t)} - \eta[-(y - \hat{y})\varphi_j h_j (1 - h_j)]$$

Neural Networks: overfitting

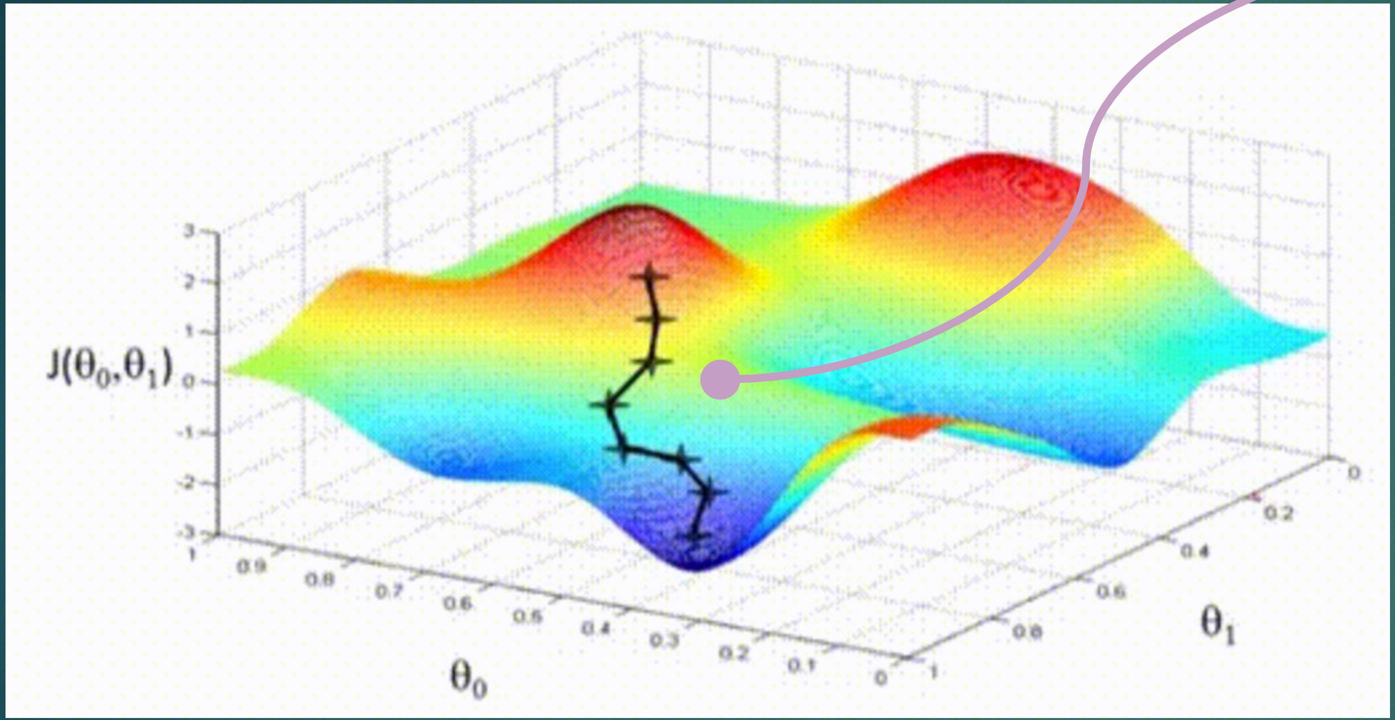
- Regularization

$$C_{\Theta}(x, y) = LL(\hat{y}(x), y) + \lambda * ||\Theta||_{\ell}$$

- Dropout
  - at each epoch randomly select some of the nodes not to be trained



Neural Networks: vanishing gradients

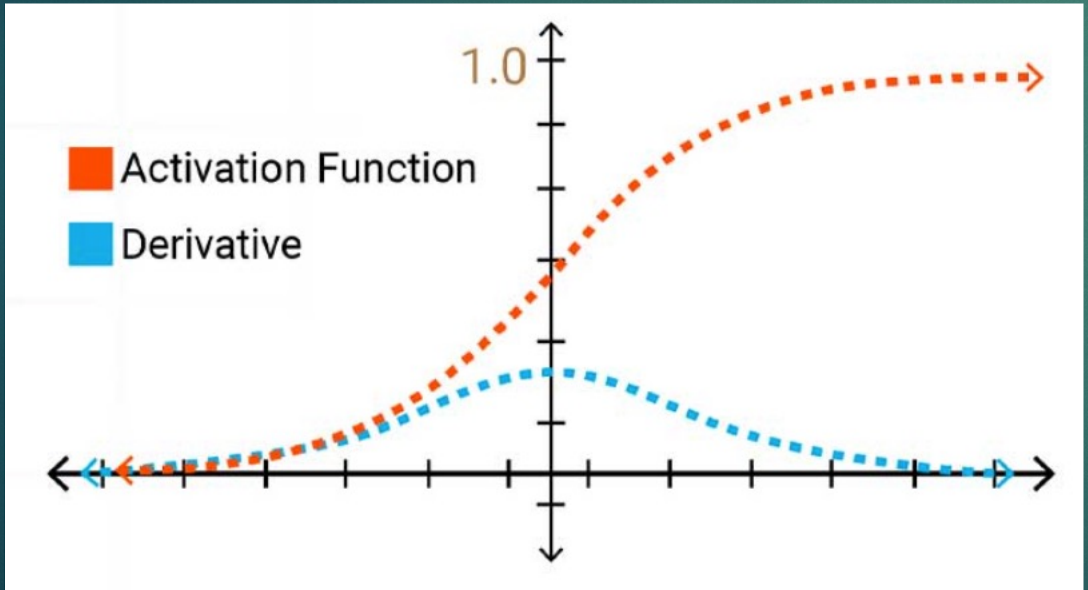


flat regions ($\nabla_\Theta \approx 0$) hinder gradient descent from progressing to the global minimum

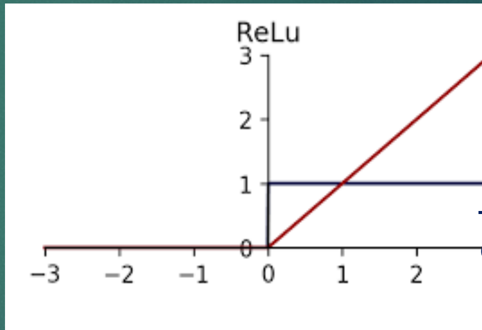
Neural Networks: vanishing gradients

- $\Theta^{(0)}$ initialization strategy

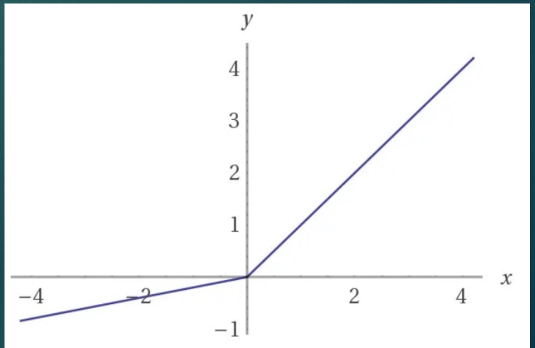
- use activation functions with non-zero derivatives



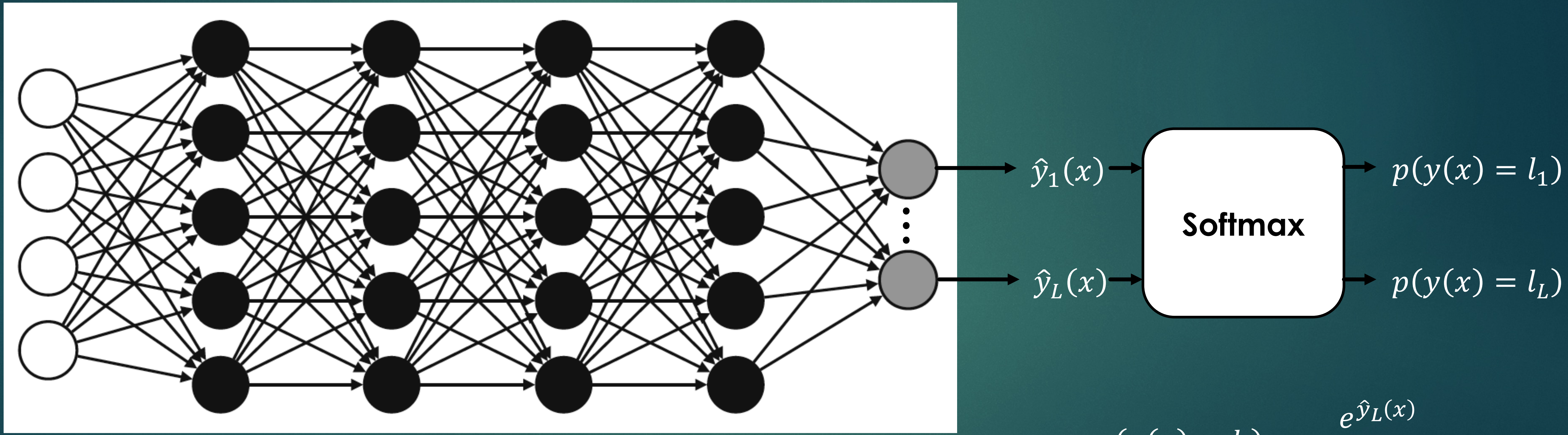
[https://www.engati.com/glossary/vanishing-gradient-problem]



$\frac{\partial}{\partial x}$

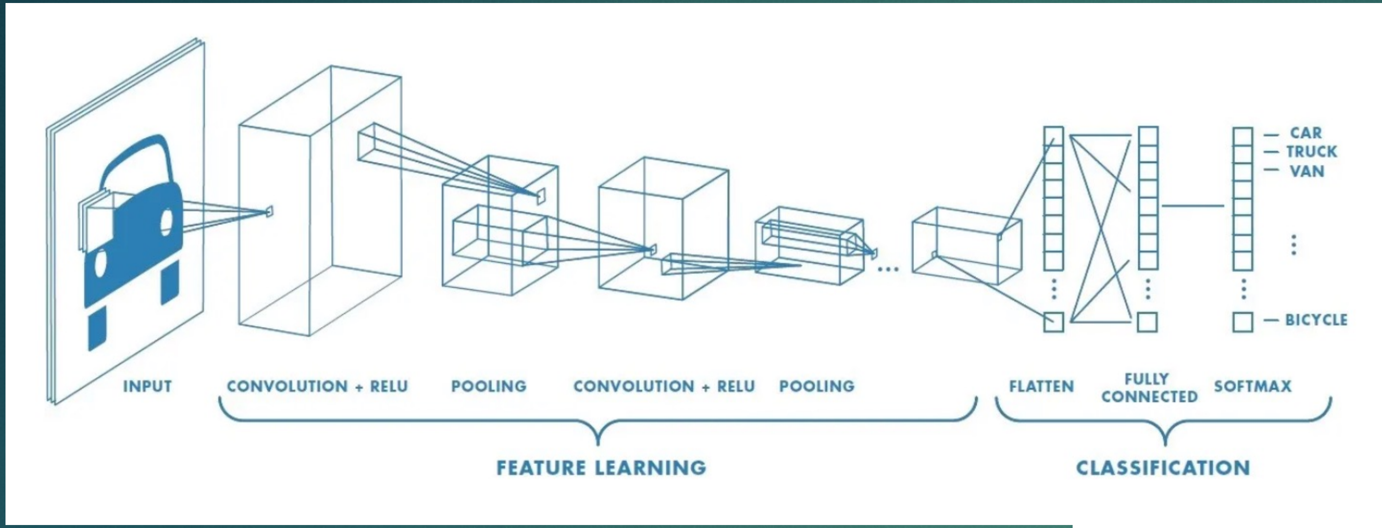


Multi label NN

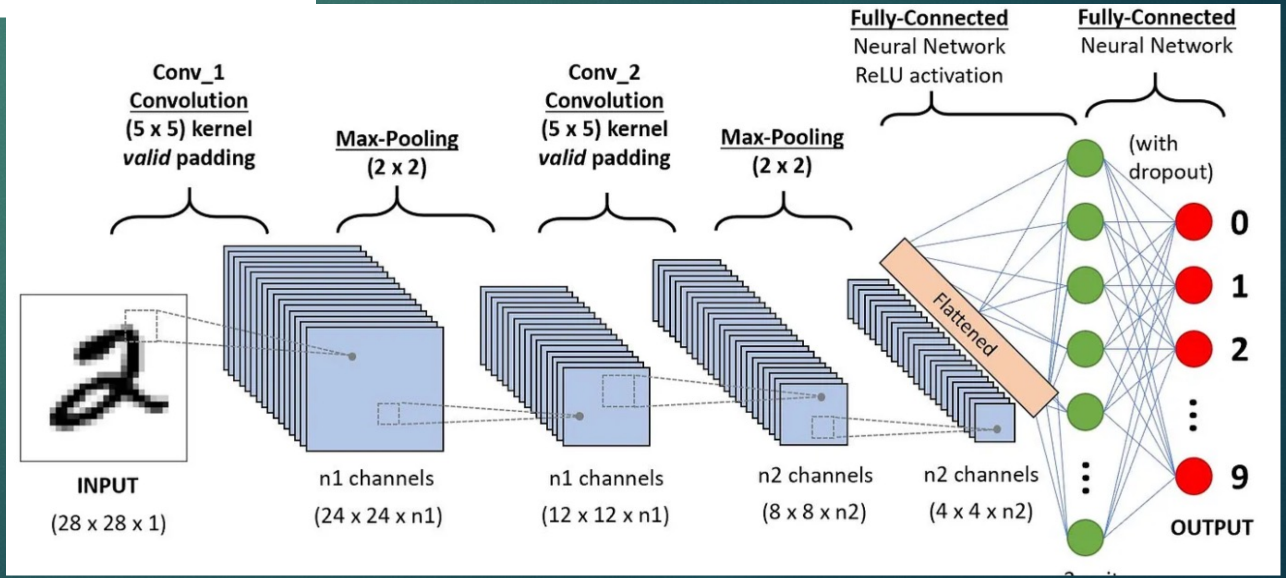


$$p(y(x) = l_l) = \frac{e^{\hat{y}_L(x)}}{\sum_{j=1}^{L} e^{\hat{y}_j(x)}}$$

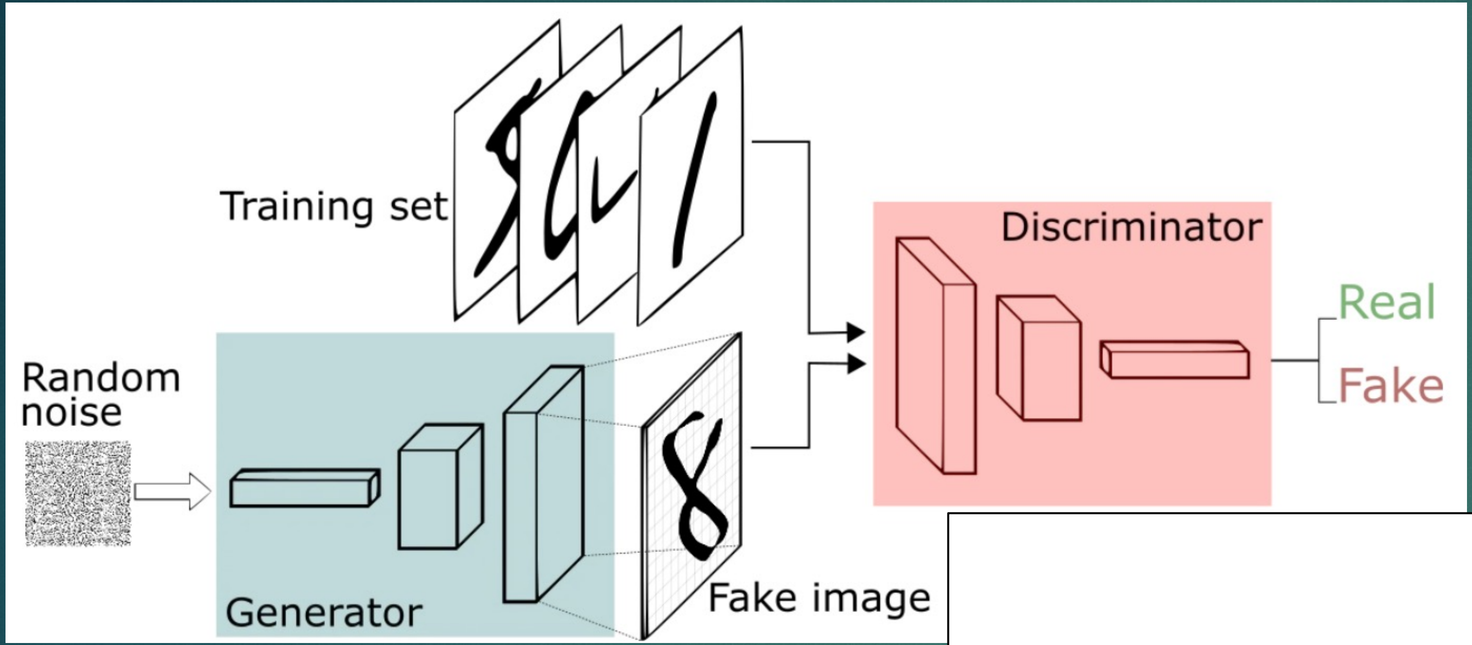
Convolution Neural Networks



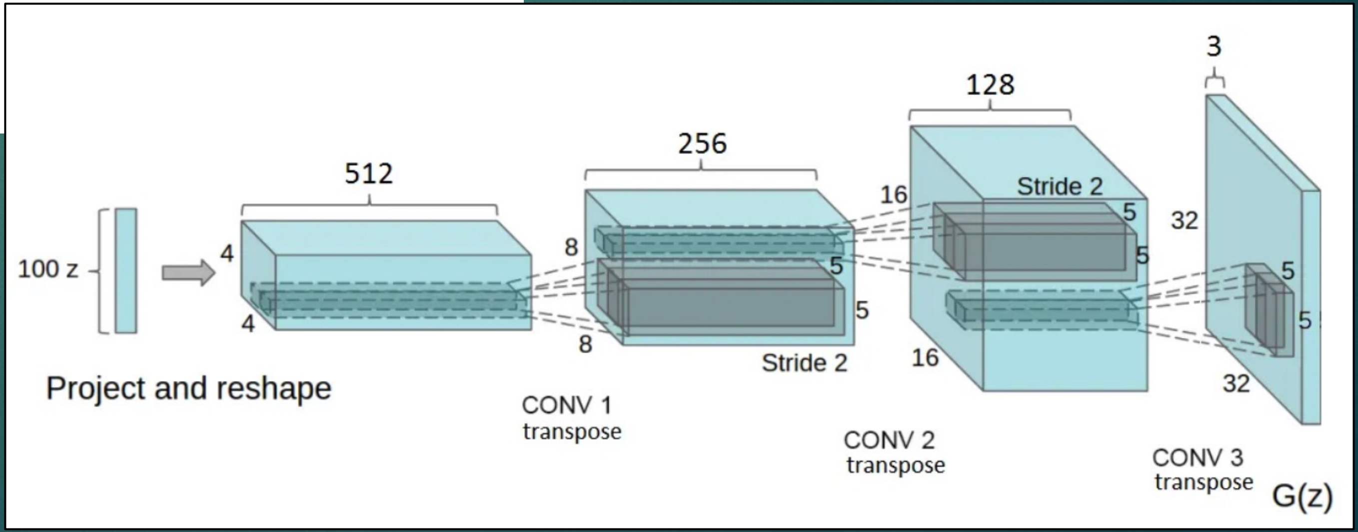
[TowardsDataScience.com]



Generative Adversarial Networks

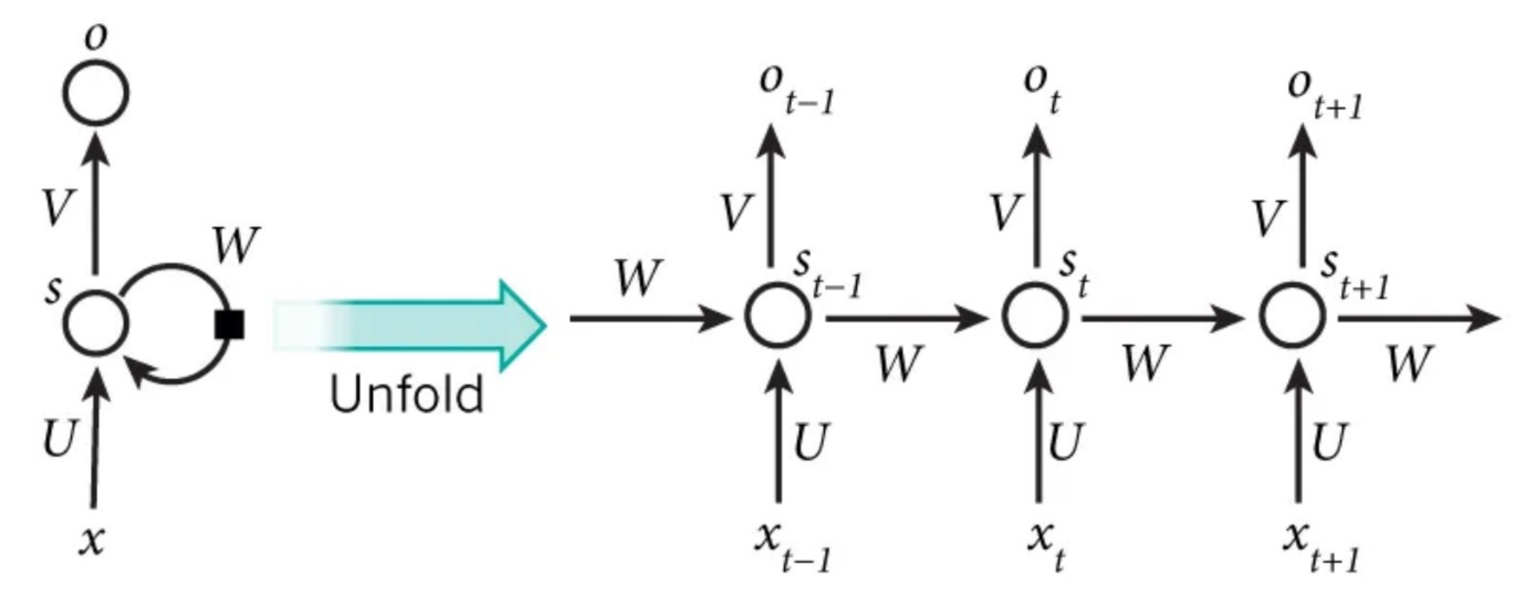


[Medium.com]



Recurrent Neural Networks

- RNNs perform the same task for every element of a sequence, with the output being dependent on the previous computations
- RNNs behave as if they had a "memory" which captures information about what has been calculated so far.



[Medium.com]