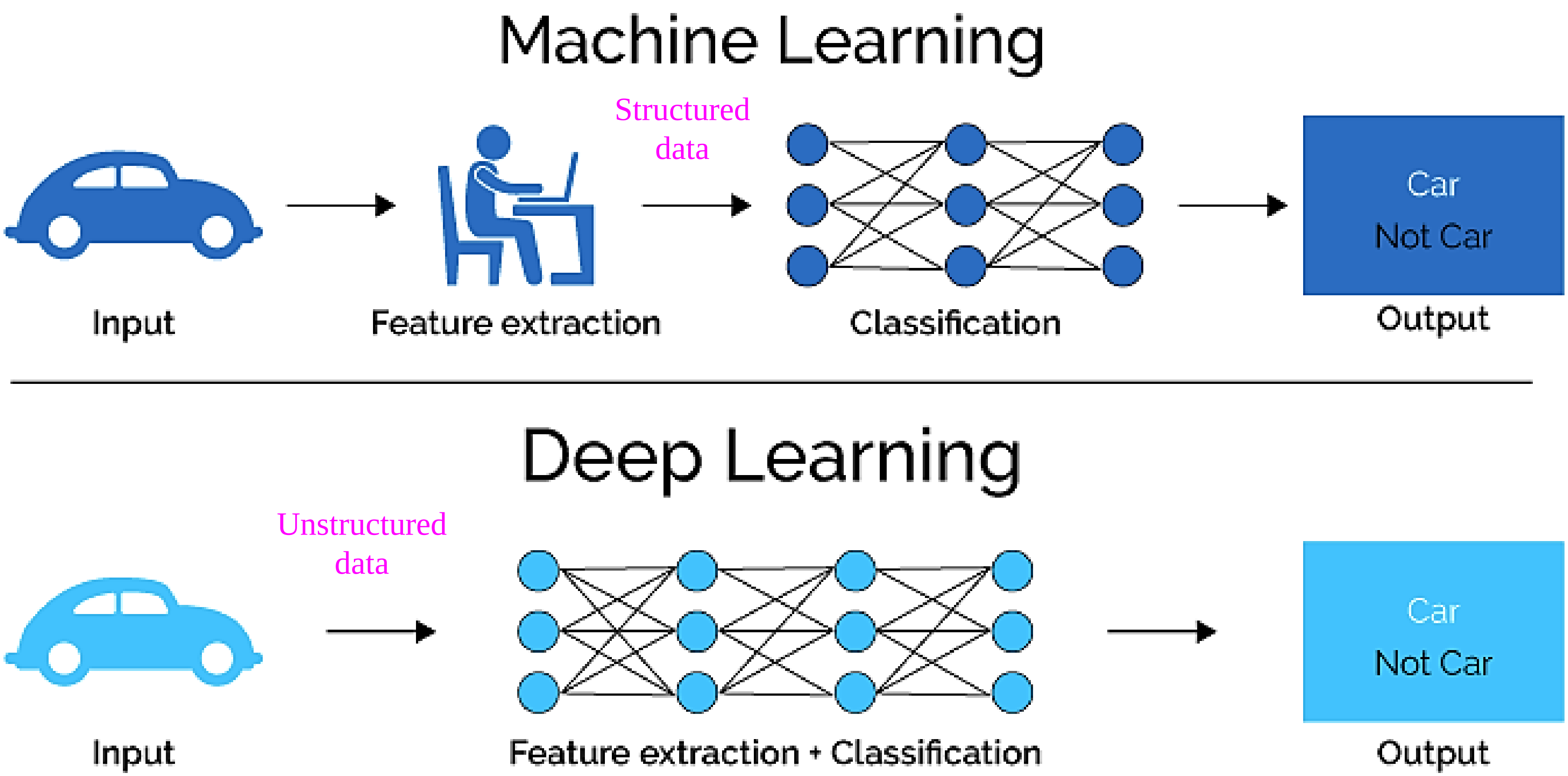
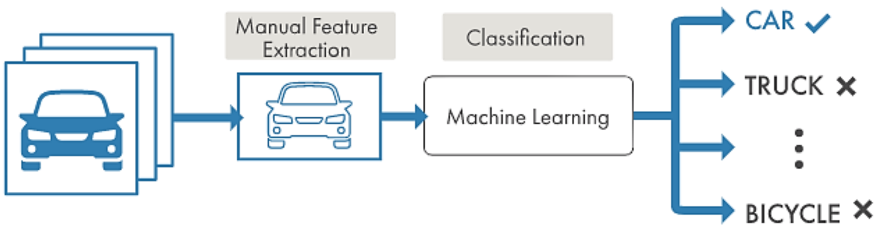
Deep Neural Network (DNN)

Machine learning vs. Deep learning

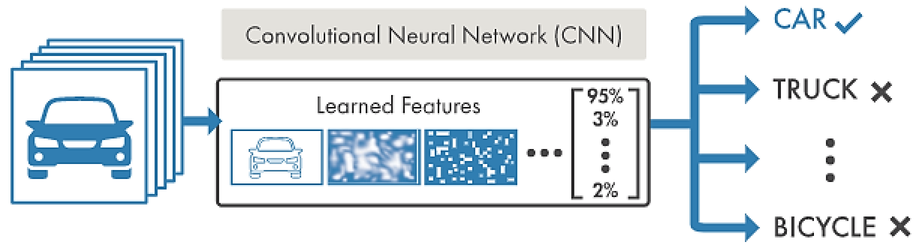


[Adaptation after https://lawtomated.com/a-i-technical-machine-vs-deep-learning/]

MACHINE LEARNING



DEEP LEARNING



A machine learning workflow starts with **relevant features being manually extracted from images**. The features are then used to create a model that categorizes the objects in the image.

With a deep learning workflow, **relevant features are automatically extracted from images**. Deep learning performs "end-to-end learning" – where a network is given raw data and a task to perform, such as classification, and it learns how to do this automatically.

Another key difference is **deep learning algorithms scale with data**, whereas **shallow learning converges**.

Shallow learning refers to machine learning methods that plateau at a certain level of performance when you add more examples and training data to the network

A key advantage of deep learning networks is that they often continue to improve as the size of your data increases.

Deep Learning, MathWorks https://www.mathworks.com/discovery/deep-learning.html

Deep learning represents the very cutting edge of artificial intelligence (AI).

An ANN in its simplest form has two layers: a hidden layer and an output layer – it is called a "shallow neural network"

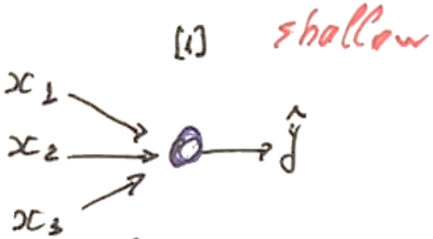
An ANN that is made up of **more than two layers**: multiple hidden layers and an output layer– is called a '**deep neural network**'

A deep learning system is self-teaching, learning as it goes by **filtering information through multiple hidden layers**, in a similar way to humans.

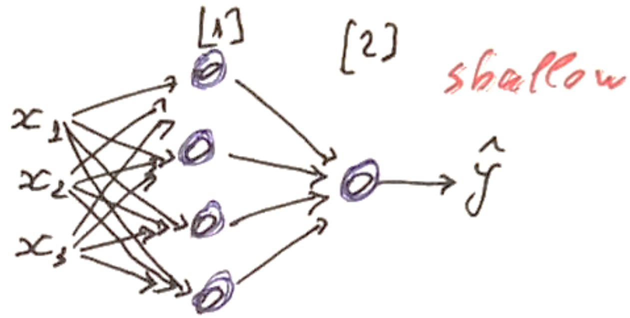
The two are closely connected: one relies on the other to function. Without (shallow) neural networks, there would be no deep learning.

Bernard Marr, Deep Learning Vs Neural Networks - What's The Difference?, https://bernardmarr.com/default.asp?contentID=1789

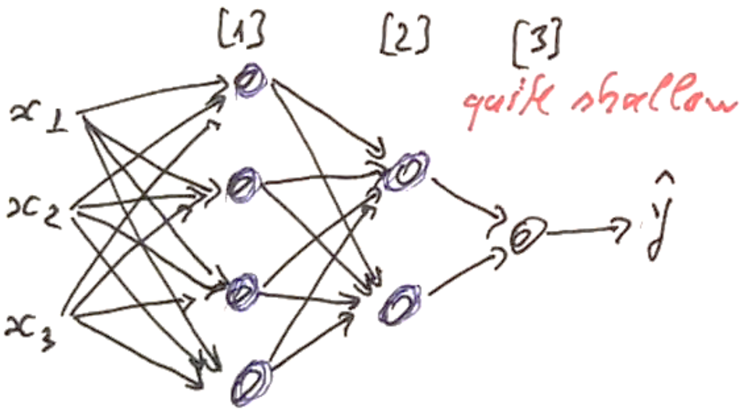
Deep Neural Network – what is it?



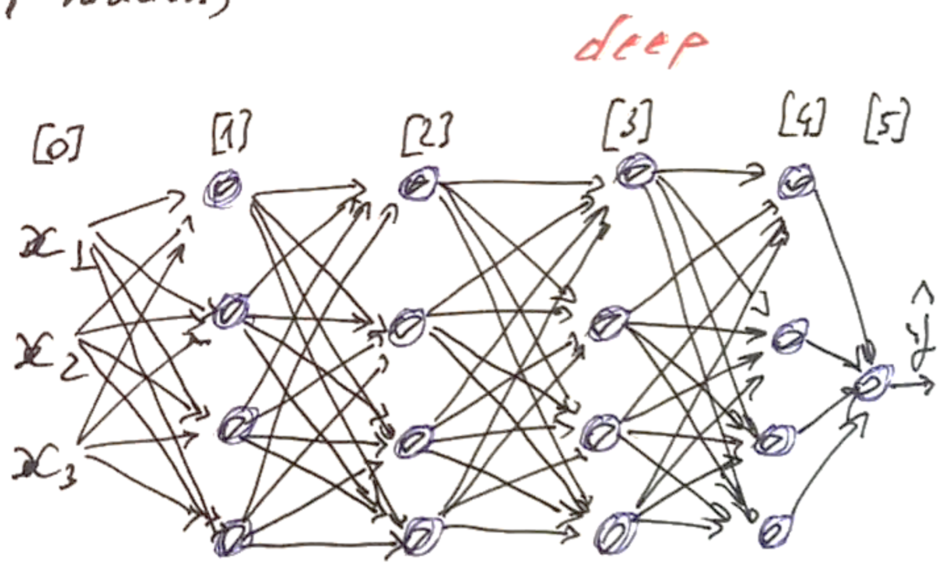
1 layer
logistic regression



2 layers
(1 hidden)



3 layers
(2 hidden)

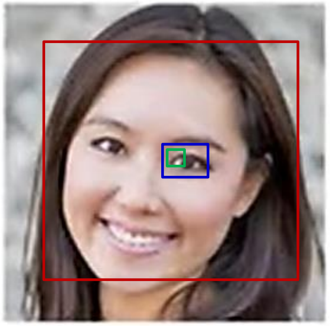


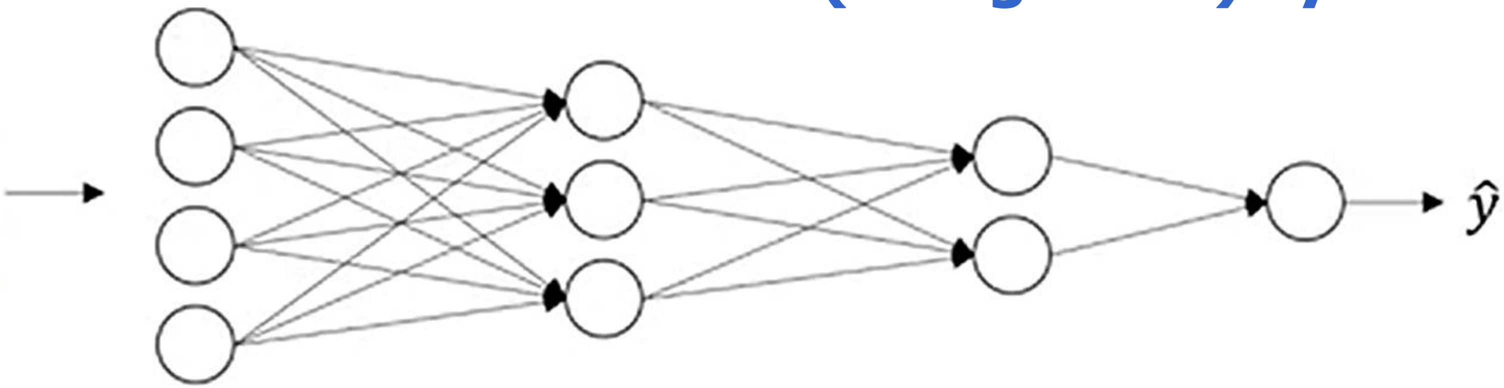
5 layers
(4 hidden)

hidden layers – hyperparameter

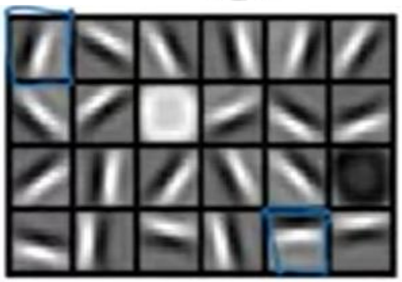
Deep Neural Network – intuition; why deep?

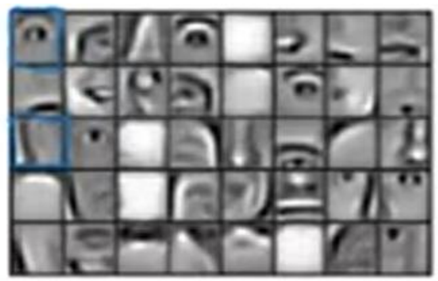
Face detection (recognition) system





Pixels





Simple to complex hieratical representation

Feature detector
Edge detector - where are the edges? (groups pixels to form edges)

Take the detected edges; groups edges together to form part of faces (eye, nose, chin, etc)

Putting together different parts of the faces to form faces

Simple things → Complex things

The complexity of the detected function increases (**edges** => **parts of faces** => **faces**)

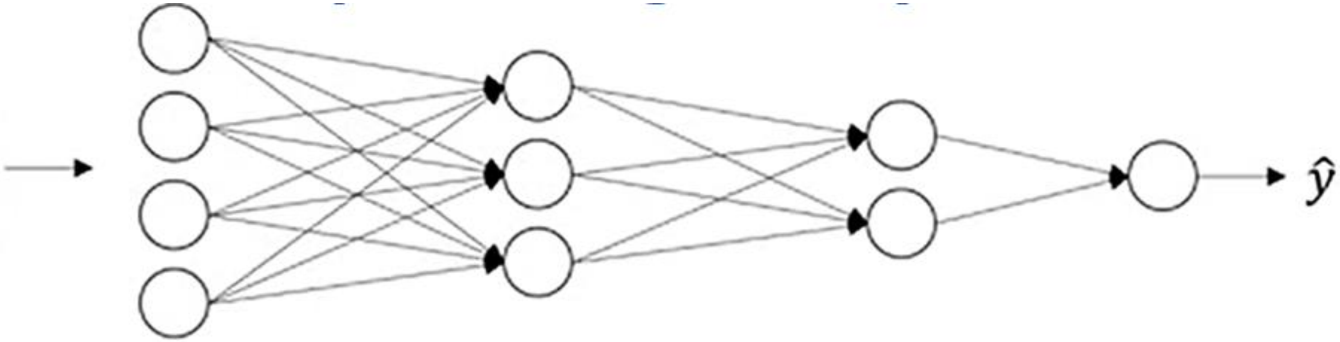
Very small window

Large window

Andrew Ng, Why deep representation?, https://www.coursera.org/lecture/neural-networks-deep-learning/why-deep-representations-rz9xJ

Deep Neural Network – intuition; why deep?

Speech recognition system



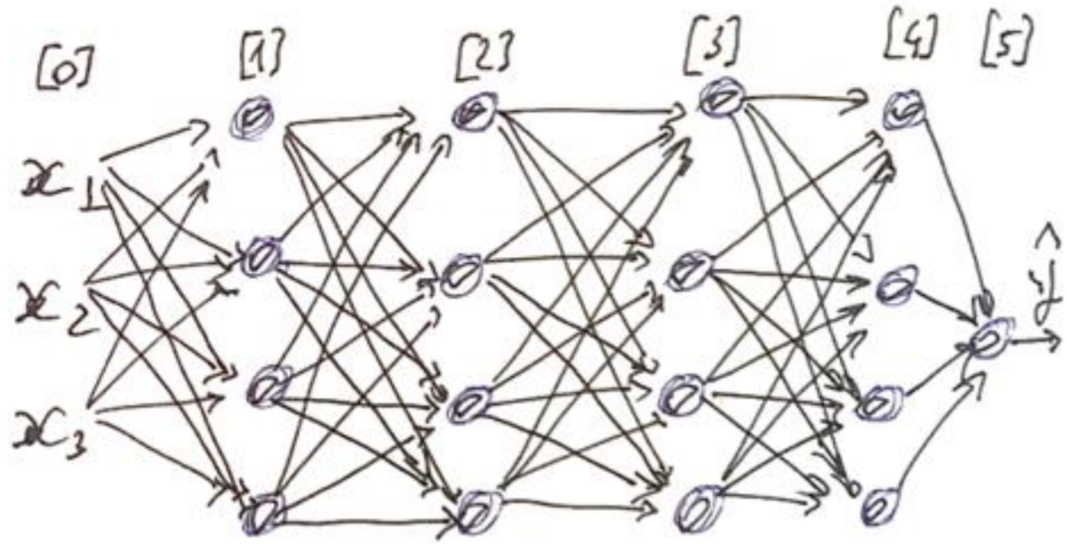
Detect low-level features of audio signal → Detect basic units of sounds (phonems) → Words → Sentences → Phrases

Simple things → Complex things

The complexity of the detected function increases

Andrew Ng, Why deep representation?, https://www.coursera.org/lecture/neural-networks-deep-learning/why-deep-representations-rz9xJ

Forward propagation in DNN



$$a^{[0]} = x$$

$$s^{[1]} = W^{[1]}a^{[0]} + b^{[1]}$$

$$a^{[1]} = f^{[1]}(s^{[1]})$$

$$s^{[2]} = W^{[2]}a^{[1]} + b^{[2]}$$

$$a^{[2]} = f^{[2]}(s^{[2]})$$

$$\vdots$$

$$s^{[5]} = W^{[5]}a^{[4]} + b^{[5]}$$

$$a^{[5]} = f^{[5]}(s^{[5]})$$

$$\hat{y} = a^{[5]}$$

For layer *l*

$$\boldsymbol{s}^{[l]} = \boldsymbol{W}^{[l]}\boldsymbol{a}^{[l-1]} + \boldsymbol{b}^{[l]}$$

$$\boldsymbol{a}^{[l]} = \boldsymbol{f}^{[l]}(\boldsymbol{s}^{[l]})$$

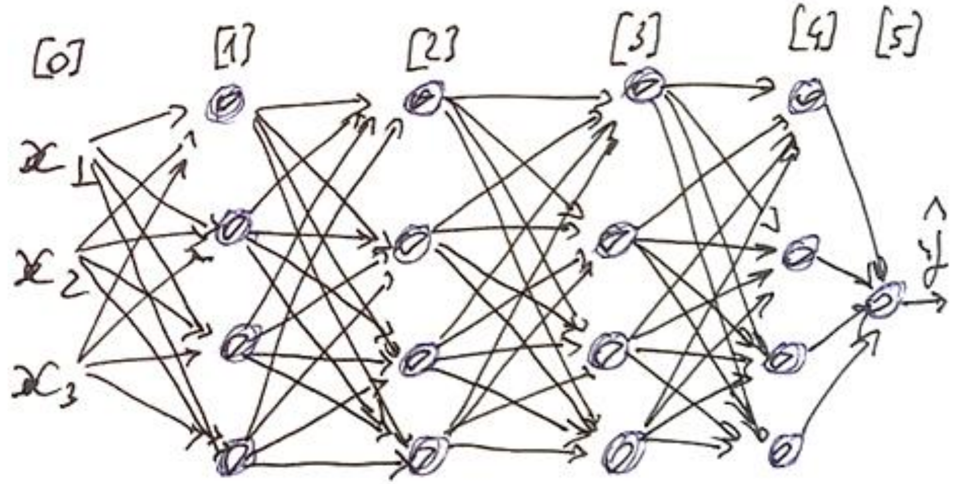
Forward propagation in DNN
Vectorizing for m input examples

input

$$X = \begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{bmatrix}$$

features (n)

training examples (m)



$$A^{[0]} = X$$

$$A^{[l]} = f^{[l]}\left(S^{[l]}\right)$$

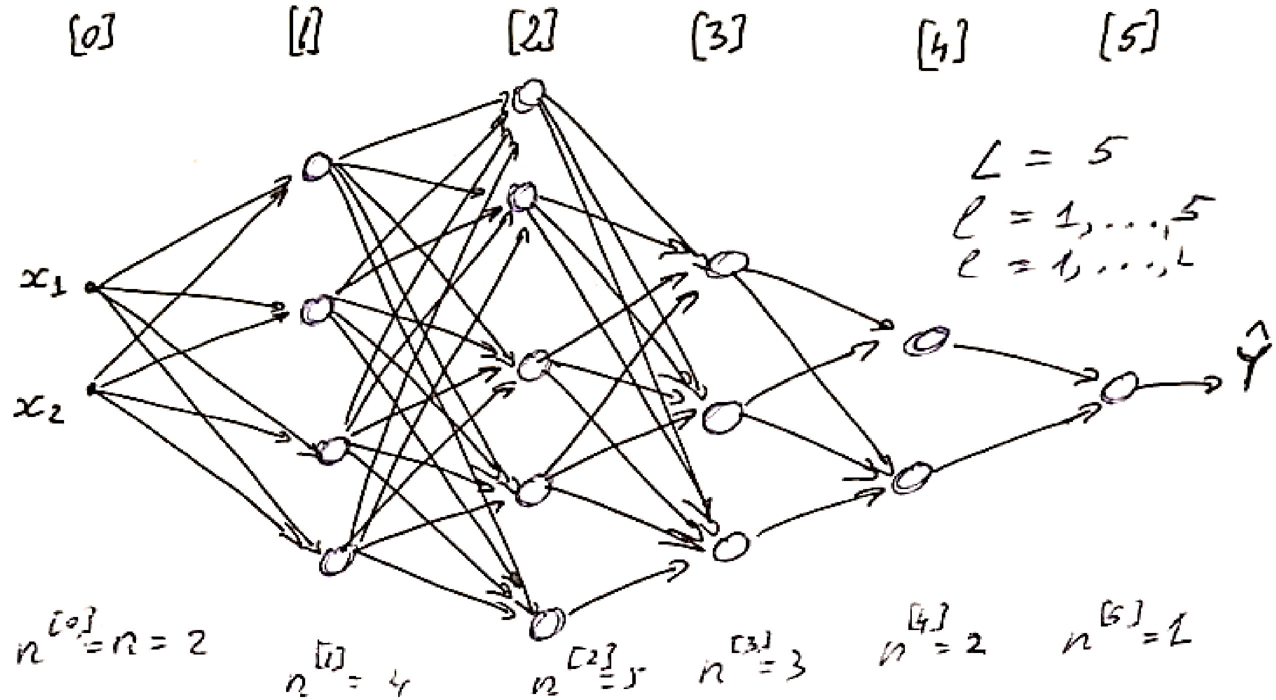
$$S^{[l]} = W^{[l]} A^{[l-1]} + b^{[l]}$$

$L = 5$

For $l = 1$ to L

$$\hat{y} = A^{[5]}$$

Matrix dimensions in DNN



For a single input example.

$x : (n^{[0]}, 1)$

$W^{[l]} : (n^{[l]}, n^{[l-1]})$; $b^{[l]} : (n^{[l]}, 1)$

$z^{[l]} : (n^{[l]}, 1)$; $a^{[l]} : (n^{[l]}, 1)$

$l = 2$ $\quad W^{[2]} : (5, 4)$; $b^{[2]} : (5, 1)$

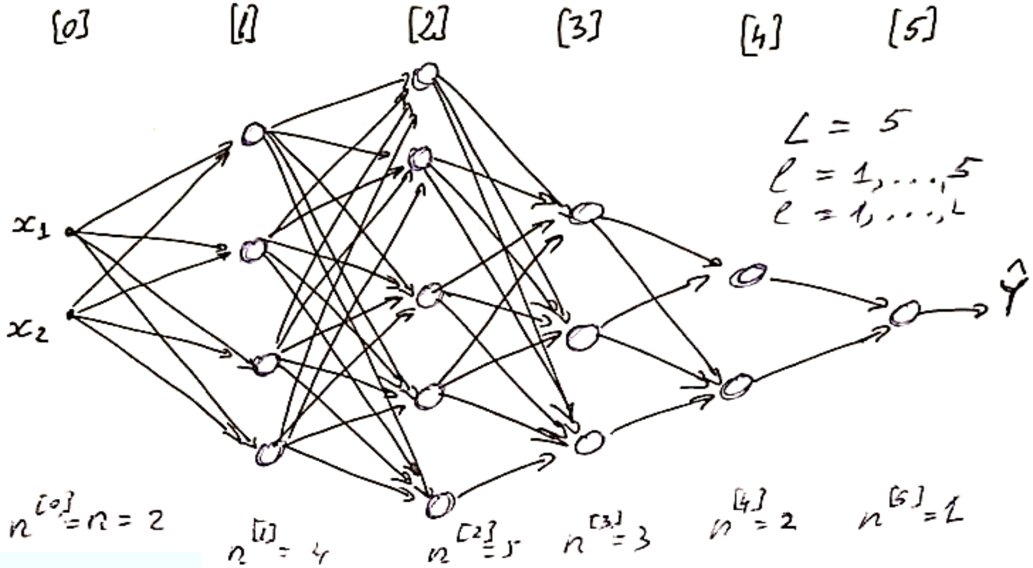
$x : (2, 1)$ $\quad z^{[2]} : (5, 1)$; $a^{[2]} : (5, 1)$

$dW^{[l]} : (n^{[l]}, n^{[l-1]})$

$db^{[l]} : (n^{[l]}, 1)$

Matrix dimensions in DNN

Vectorizing for *m* input examples



For m input examples:

$X : (n^{[0]}, m)$; $A^{[0]} = X$

$w^{[\ell]} : (n^{[\ell]}, n^{[\ell-1]})$; $b^{[\ell]} : (n^{[\ell]}, 1)$

$z^{[\ell]} : (n^{[\ell]}, m)$; $A^{[\ell]} : (n^{[\ell]}, m)$

$\ell = 4$; $\underline{m = 10}$

$X = A^{[0]} : (2, 10)$

$w^{[4]} : (2, 3)$; $b^{[4]} : (2, 1)$

$z^{[4]} : (2, 10)$; $A^{[4]} : (2, 10)$

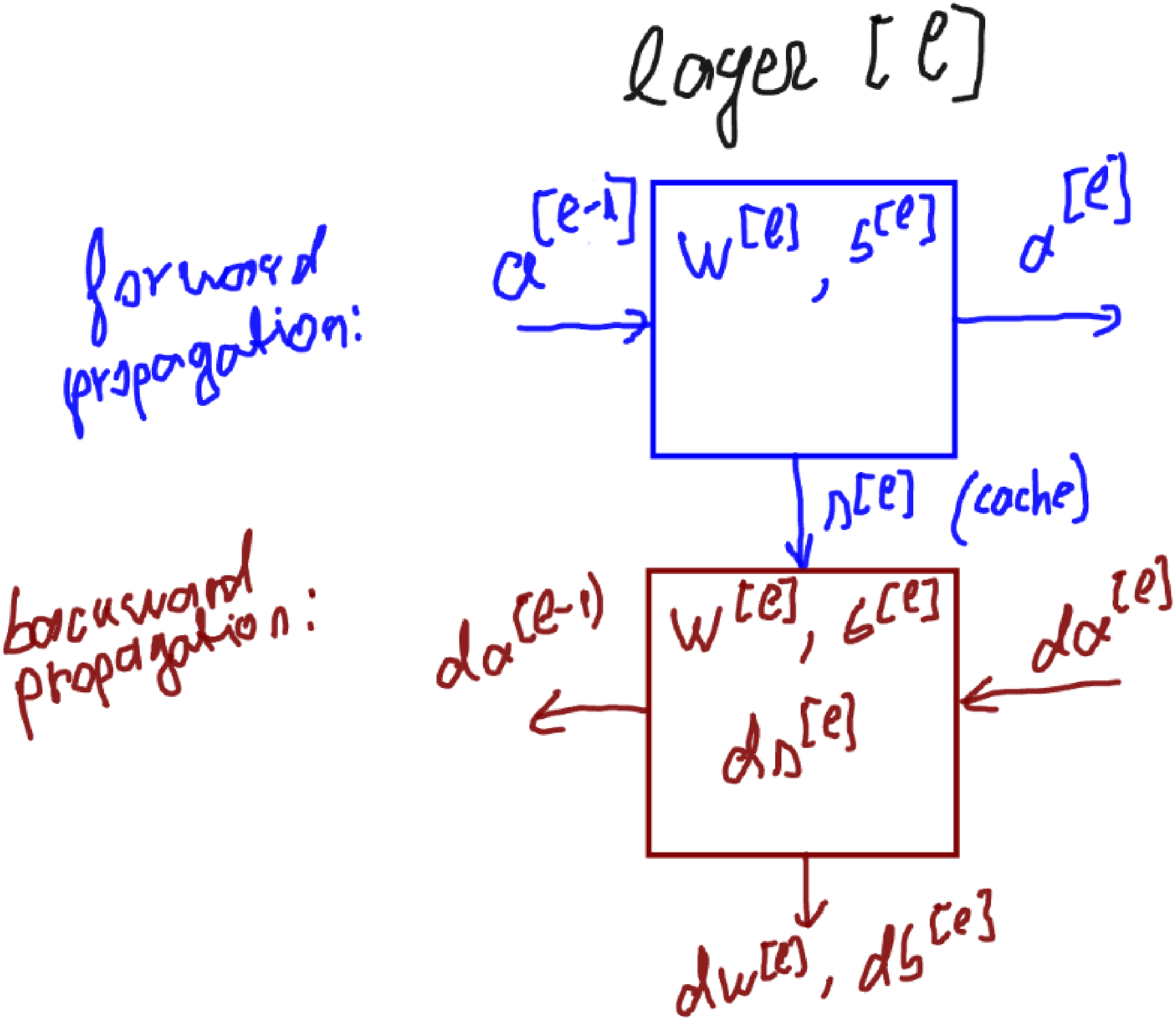
$dw^{[\ell]} : (n^{[\ell]}, n^{[\ell-1]})$

$db^{[\ell]} : (n^{[\ell]}, 1)$

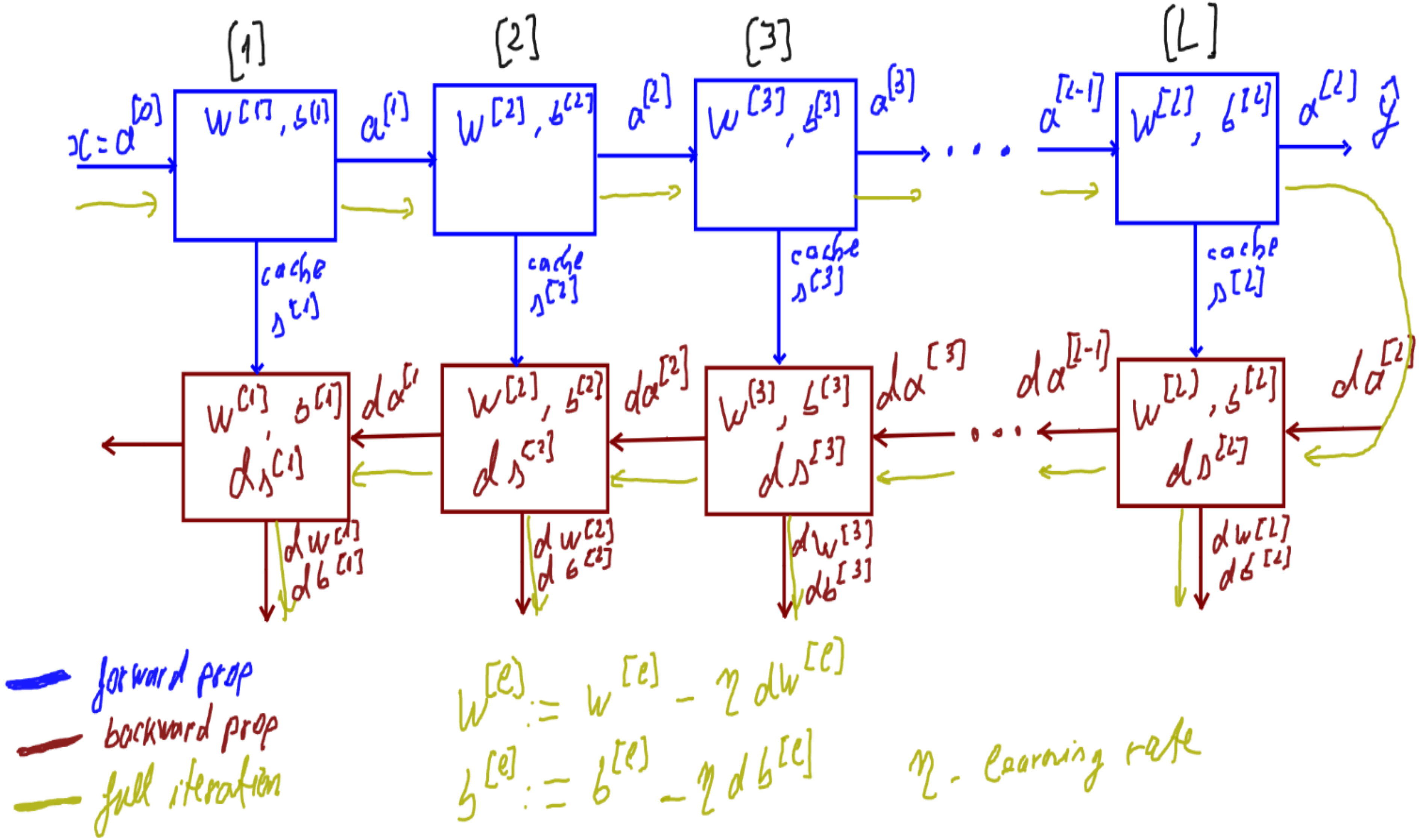
$dz^{[\ell]} : (n^{[\ell]}, m)$

$dA^{[\ell]} : (n^{[\ell]}, m)$

Forward and backward propagation for layer ℓ



Forward and backward propagation for a DNN



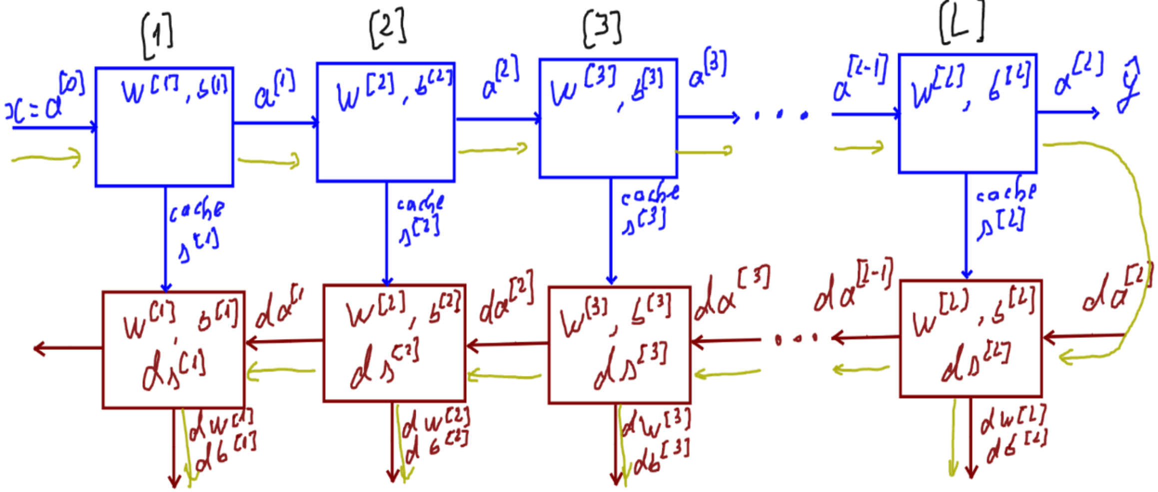
$$W^{[l]} := W^{[l]} - \eta \, dW^{[l]}$$

$$b^{[l]} := b^{[l]} - \eta \, db^{[l]}$$

η - learning rate

Forward and backward propagation for a DNN.

Vectorizing for *m* input examples



$$A^{[0]} = X$$

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

$$A^{[l]} = f^{[l]}\left(S^{[l]}\right)$$

$$\hat{y} = A^{[L]}$$

$$dS^{[L]} = A^{[L]} - Y$$

$$dW^{[L]} = \frac{1}{m} dS^{[L]} {A^{[L-1]}}^{T}$$

$$db^{[L]} = \frac{1}{m} np.sum(dS^{[L]}, axis = 1, keepdims = True)$$

$$dS^{[L-1]} = {W^{[L]}}^{T} dS^{[L]} * f'^{[L-1]}\left(S^{[L-1]}\right)$$

$*$ element-wise multiplication

$$dS^{[1]} = {W^{[2]}}^{T} dS^{[2]} * f'^{[1]}\left(S^{[1]}\right)$$

$$dW^{[1]} = \frac{1}{m} dS^{[1]} {A^{[0]}}^{T} \qquad {A^{[0]}}^{T} = X^{T}$$

$$db^{[1]} = \frac{1}{m} np.sum(dS^{[1]}, axis = 1, keepdims = True)$$

Parameters. Hyperparameters

Parameters: $W^{[1]}, b^{[1]}, W^{[1]}, b^{[1]}, \dots, W^{[L]}, b^{[L]}$

Hyperparameters:

Learning rate
Number of training epochs
Number of hidden layers
Number of neurons in each hidden layer
Activation function for each layer

Minibatch size
Momentum
Regularization
…

Hyperparameters will determine the final value of the parameters

Trial and error to find optimal hyperparameters
Deep learning is an empirical process – try a lot of things and see what works!