Title: Lecture - Machine Learning, PHYS 777

Speakers: Mohamed Hibat Allah

Collection/Series: Machine Learning (Elective), PHYS 777, February 24 - March 28, 2025

Subject: Condensed Matter, Other

Date: February 28, 2025 - 9:00 AM

URL: https://pirsa.org/25020017

Lecture 3

Recall the goal of supervised learning (SL): Given a dataset $D = \{(\vec{x}, \vec{y})\}$, fit a function $f(\vec{x})$ to $\vec{y}$.

Feed Forward NN (FFNN):

- $\vec{x} = (x_1, x_2, \ldots, x_{d_x})$.
- $\vec{y} = (y_1, y_2, \ldots, y_{d_y})$.
- $f : \mathbb{R}^{d_x} \longrightarrow \mathbb{R}^{d_y}$.

So far, we have studied two SL algorithms: ("Linear regression" and "Logistic regression").

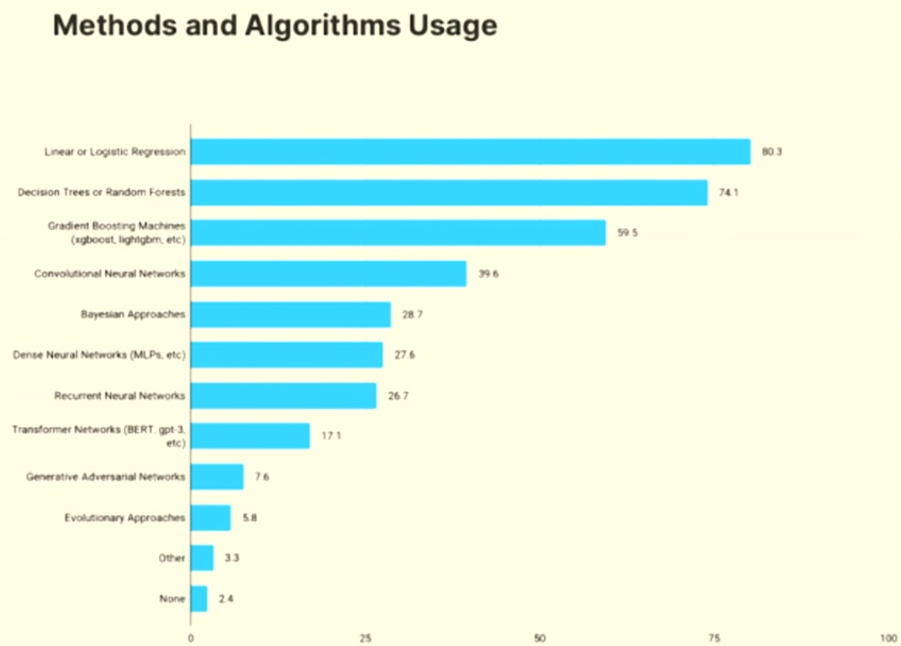
Outline for today: Intro to SL with feed forward neural networks.

→ Architecture.

→ Expressivity of NNs

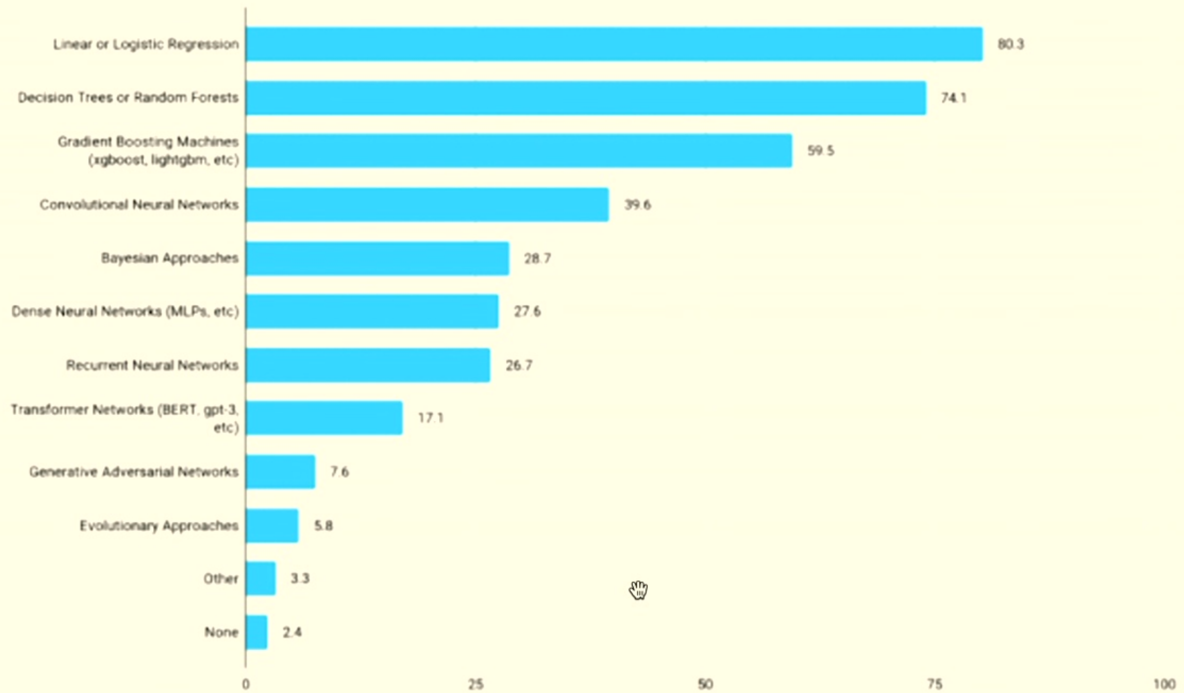
→ Basics of training (Cost functions).

Commonly used algorithms



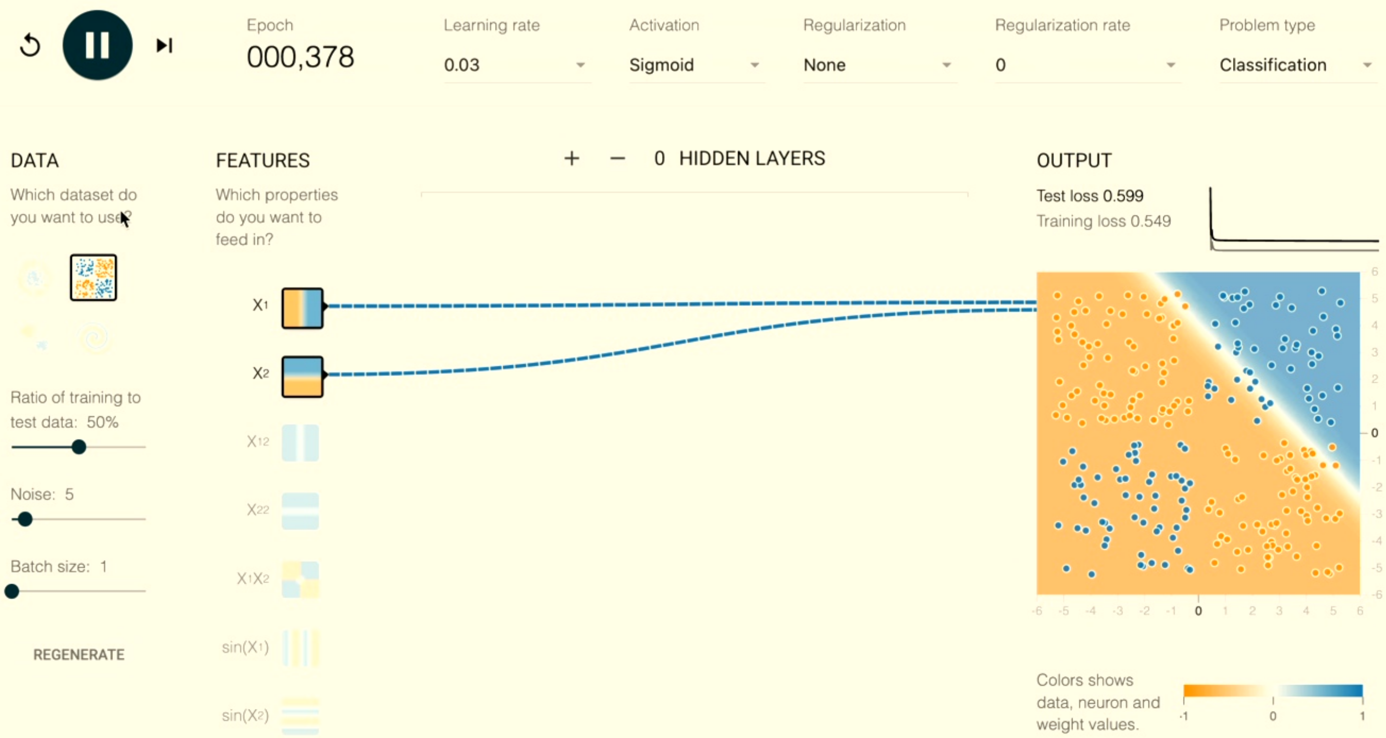
Kaggle | State of ML & Data Science 2021

Methods and Algorithms Usage



Kaggle | State of ML & Data Science 2021

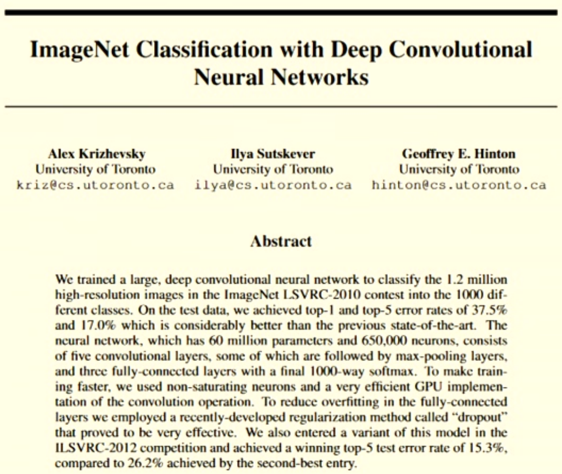
Logistic regression limitation



Historic motivation

Neural networks have been studied since the 70’s, until the breakthrough moment in 2012 in the field of **computer vision**.

Nowadays, neural networks are doing wonders in the field of **natural language processing**.



ImageNet Classification with Deep Convolutional Neural Networks

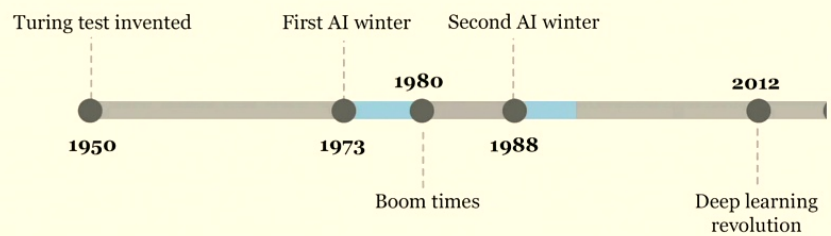
Alex Krizhevsky
University of Toronto
kriz@cs.utoronto.ca

Ilya Sutskever
University of Toronto
ilya@cs.utoronto.ca

Geoffrey E. Hinton
University of Toronto
hinton@cs.utoronto.ca

Abstract

We trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 different classes. On the test data, we achieved top-1 and top-5 error rates of 37.5% and 17.0% which is considerably better than the previous state-of-the-art. The neural network, which has 60 million parameters and 650,000 neurons, consists of five convolutional layers, some of which are followed by max-pooling layers, and three fully-connected layers with a final 1000-way softmax. To make training faster, we used non-saturating neurons and a very efficient GPU implementation of the convolution operation. To reduce overfitting in the fully-connected layers we employed a recently-developed regularization method called “dropout” that proved to be very effective. We also entered a variant of this model in the ILSVRC-2012 competition and achieved a winning top-5 test error rate of 15.3%, compared to 26.2% achieved by the second-best entry.



Credit: towardsdatascience.com

Historic motivation

Neural networks have been studied since the 70's, until the breakthrough moment in 2012 in the field of **computer vision**.

Nowadays, neural networks are doing wonders in the field of **natural language processing**.

ImageNet Classification with Deep Convolutional Neural Networks

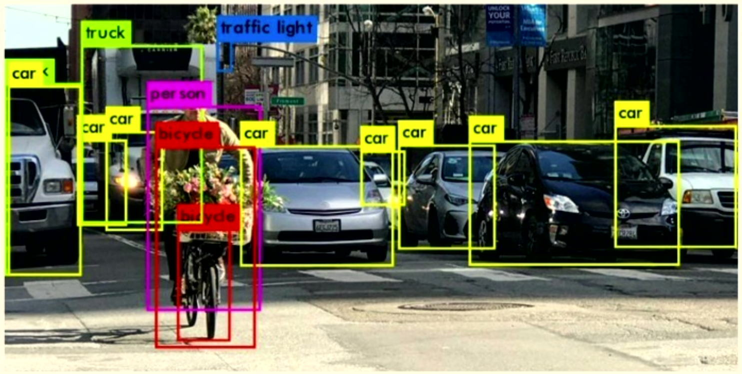
Alex Krizhevsky
University of Toronto
kriz@cs.utoronto.ca

Ilya Sutskever
University of Toronto
ilya@cs.utoronto.ca

Geoffrey E. Hinton
University of Toronto
hinton@cs.utoronto.ca

Abstract

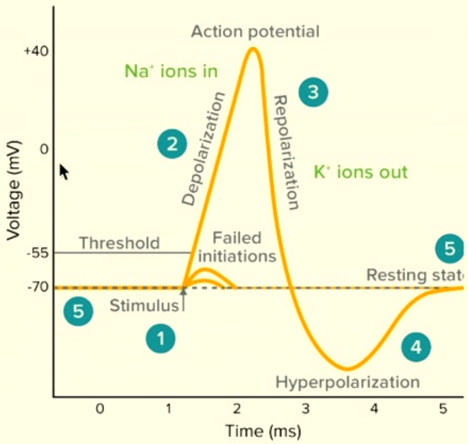
We trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 different classes. On the test data, we achieved top-1 and top-5 error rates of 37.5% and 17.0% which is considerably better than the previous state-of-the-art. The neural network, which has 60 million parameters and 650,000 neurons, consists of five convolutional layers, some of which are followed by max-pooling layers, and three fully-connected layers with a final 1000-way softmax. To make training faster, we used non-saturating neurons and a very efficient GPU implementation of the convolution operation. To reduce overfitting in the fully-connected layers we employed a recently-developed regularization method called "dropout" that proved to be very effective. We also entered a variant of this model in the ILSVRC-2012 competition and achieved a winning top-5 test error rate of 15.3%, compared to 26.2% achieved by the second-best entry.

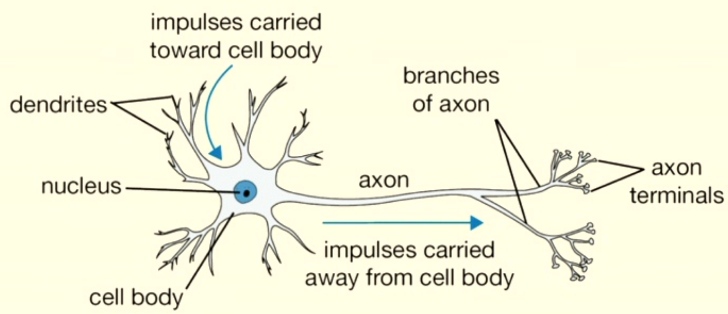


Credit: azati.ai

Inspiration: the brain

- Our brain has $\sim 10^{11}$ neurons, each of which communicates to other $\sim 10^{4}$ neurons

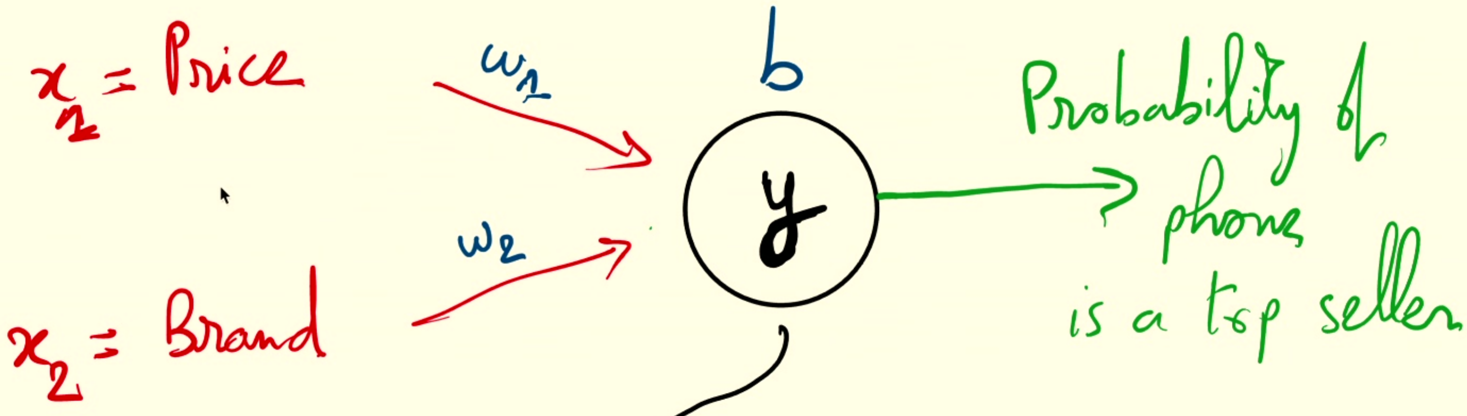




- Neurons receive input signals and accumulate voltage. After some threshold they will fire spiking responses.

credit: www.moleculardevices.com, http://cs231n.github.io/neural-networks-1/

Simplified model of an artificial neuron

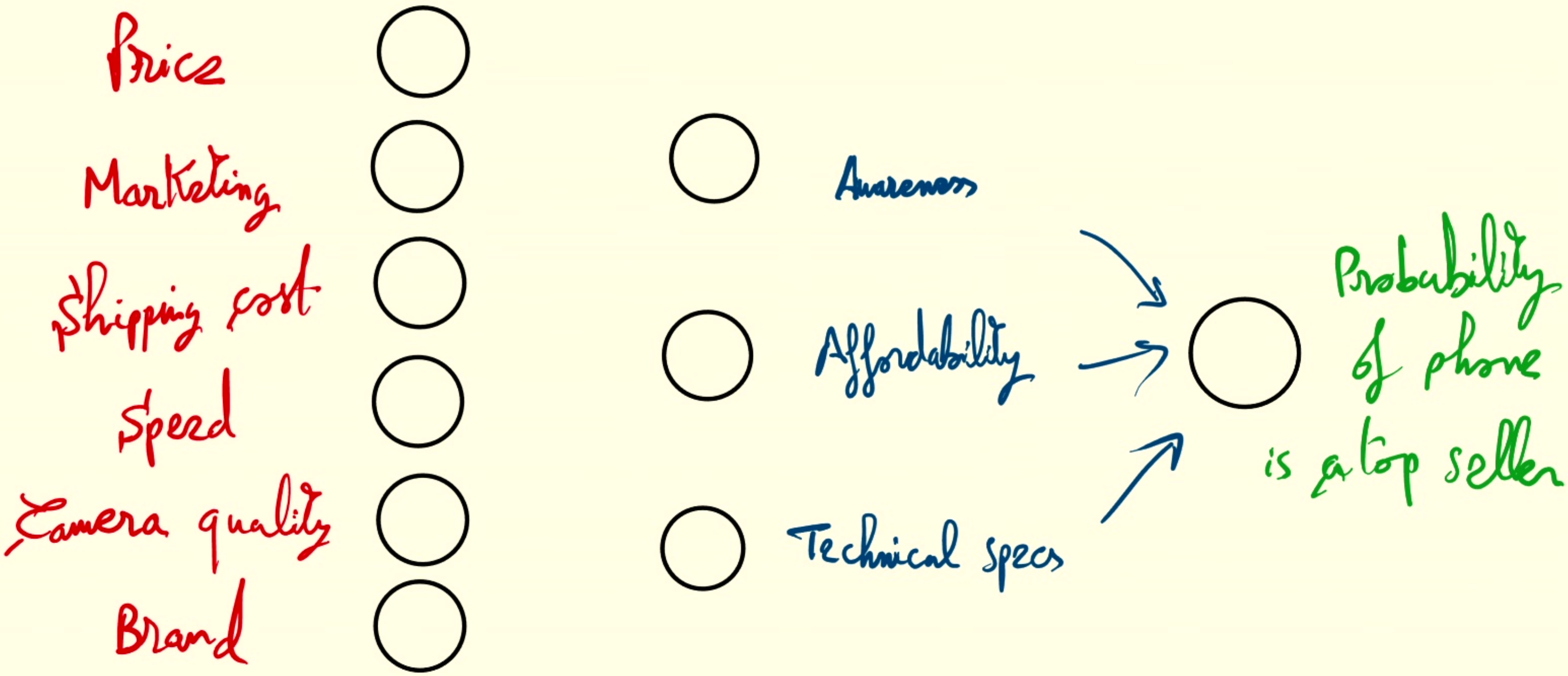


$$y = g(w_1 x_1 + w_2 x_2 + b)$$

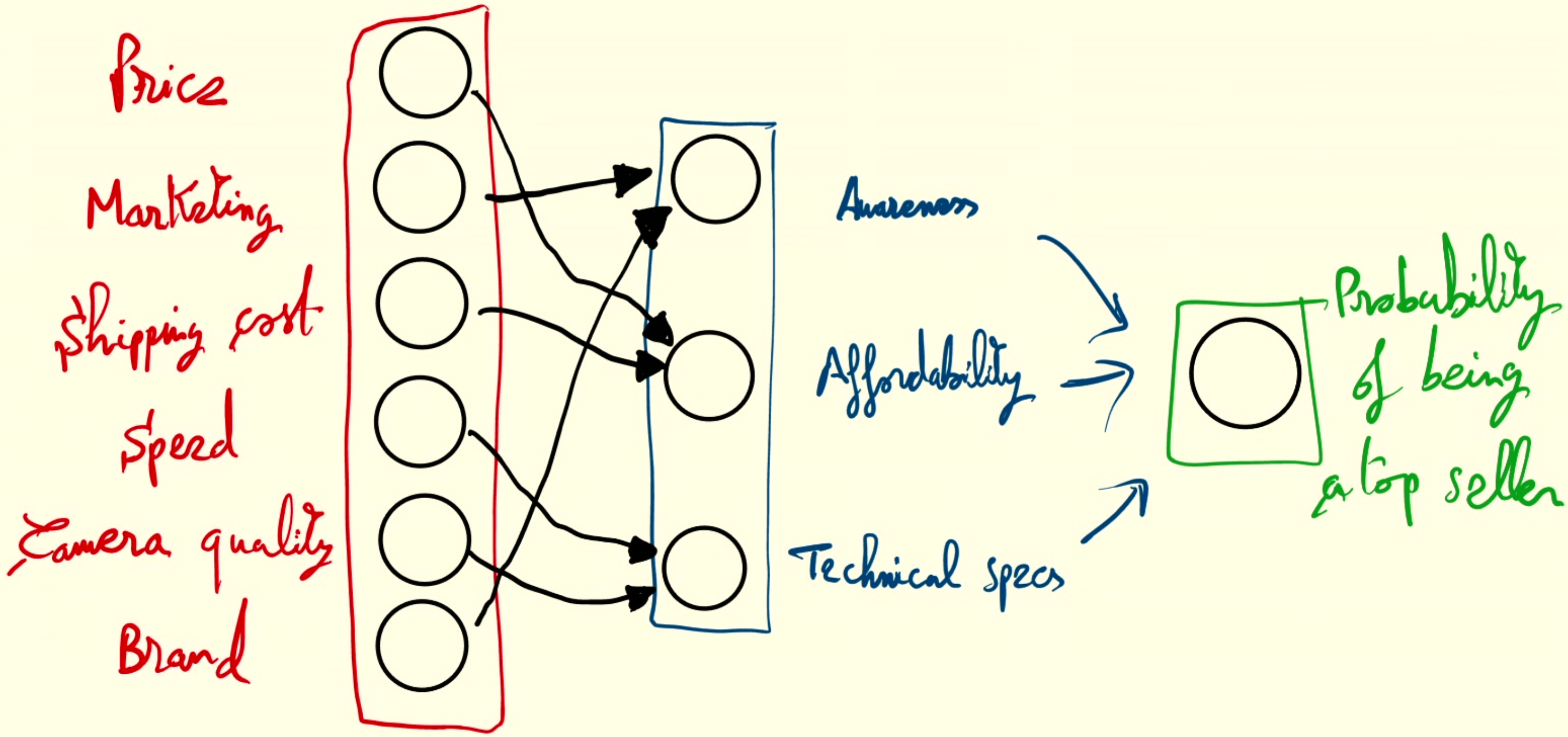
$$g(z) = \frac{1}{1+e^{-z}}$$

« Sigmoid »

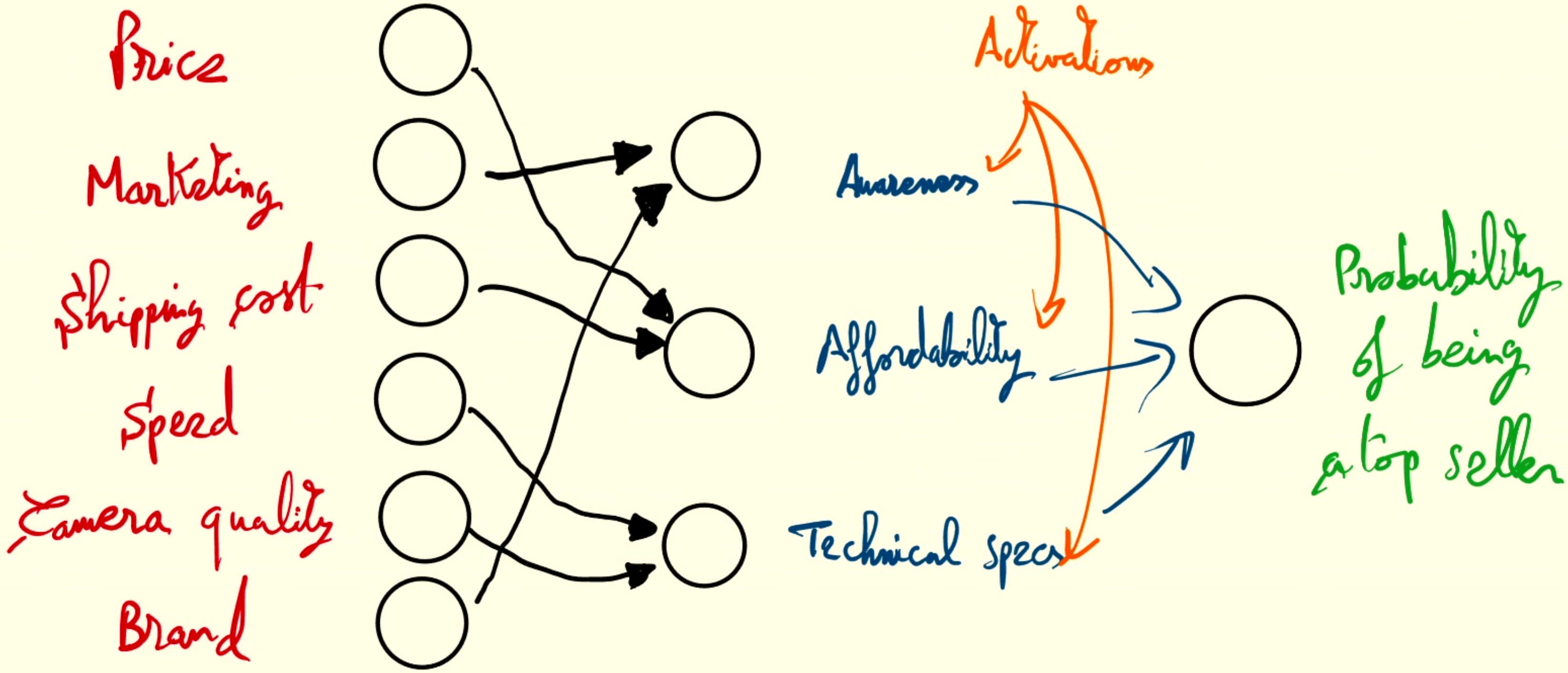
Simplified model of artificial neurons



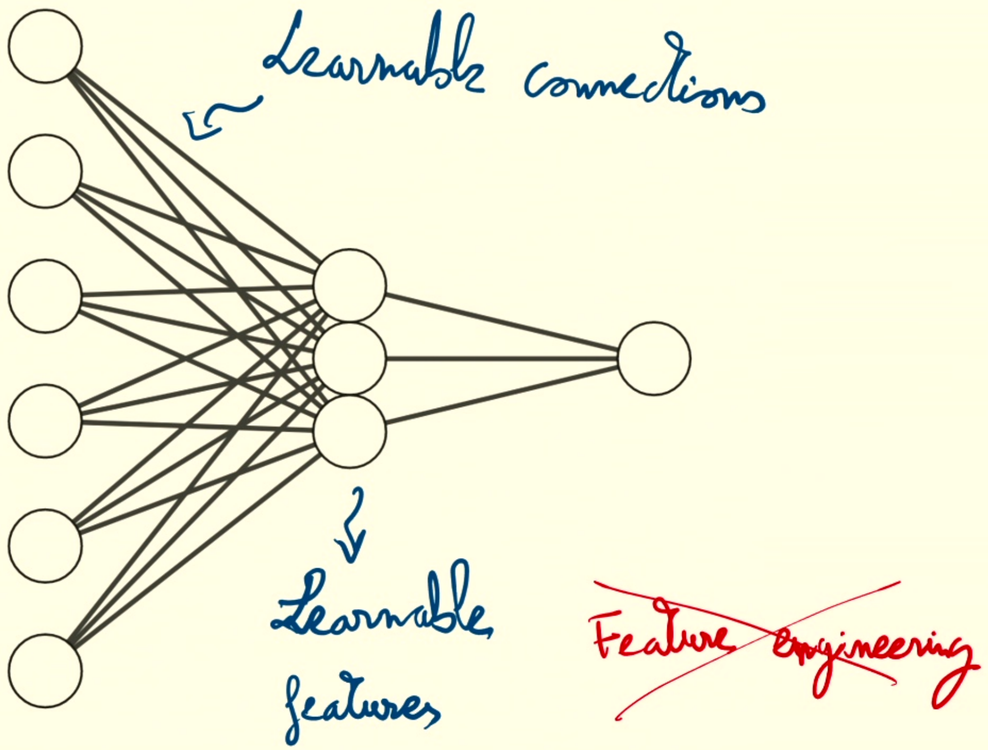
Notion of a layer



Notion of an activation

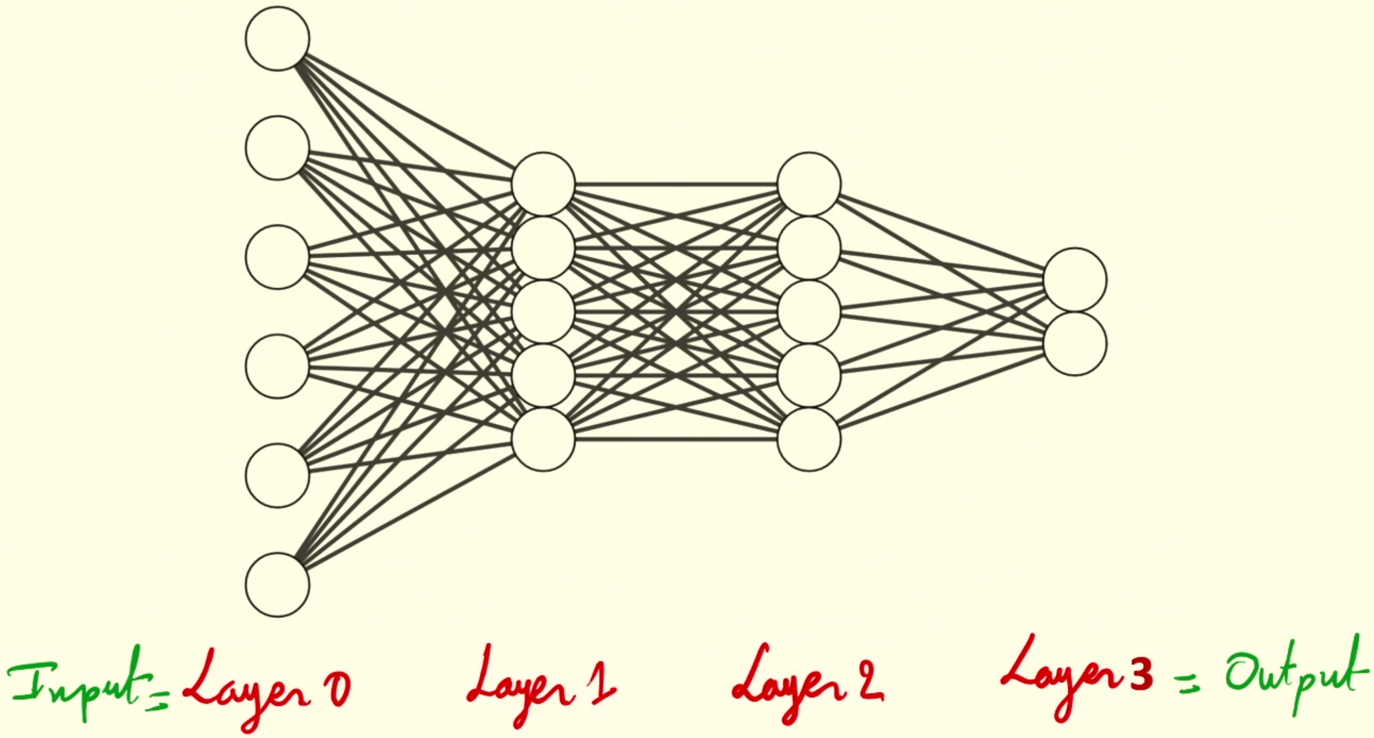


Feed-forward neural networks

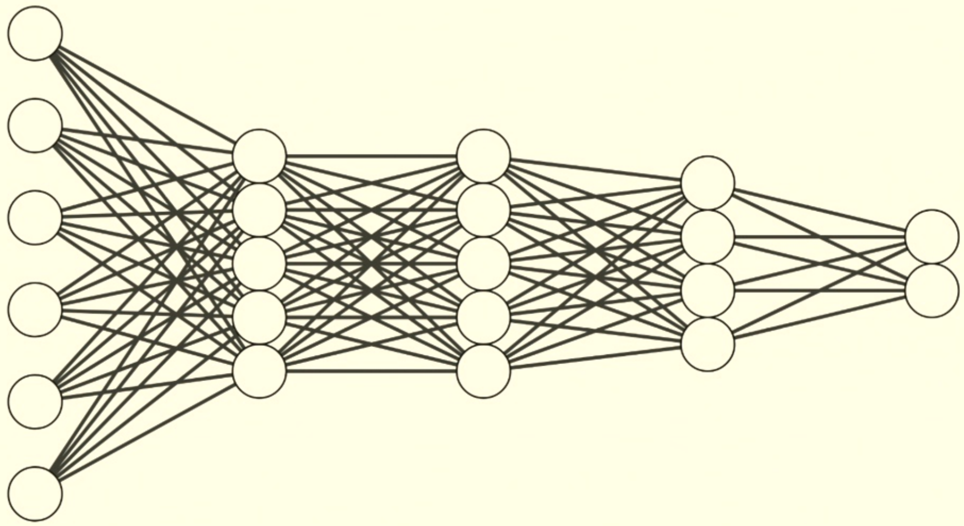


Credit: https://alexlenail.me/NN-SVG/index.html

Feed-forward neural networks

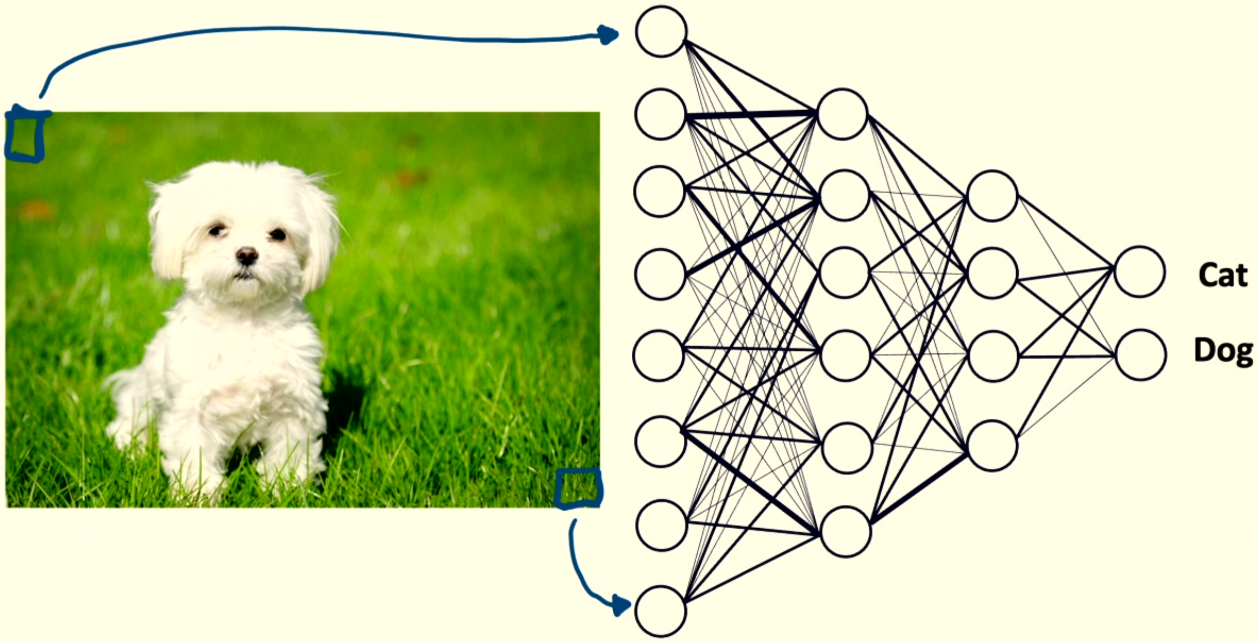


Deep feed-forward neural networks

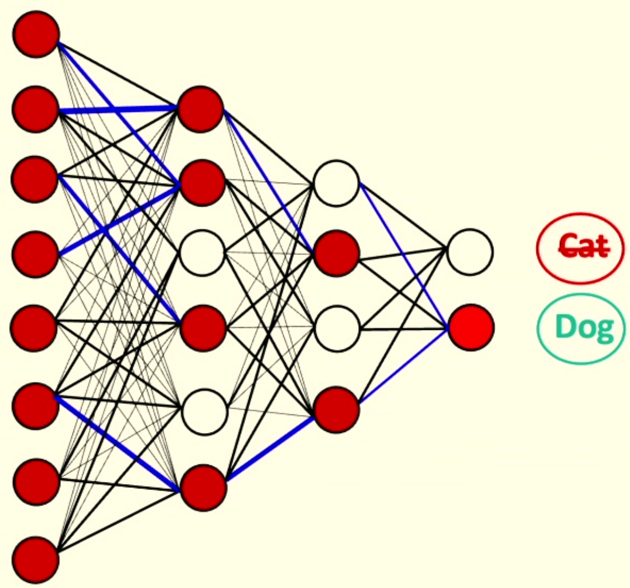


Neural networks can reach 100s of layers.

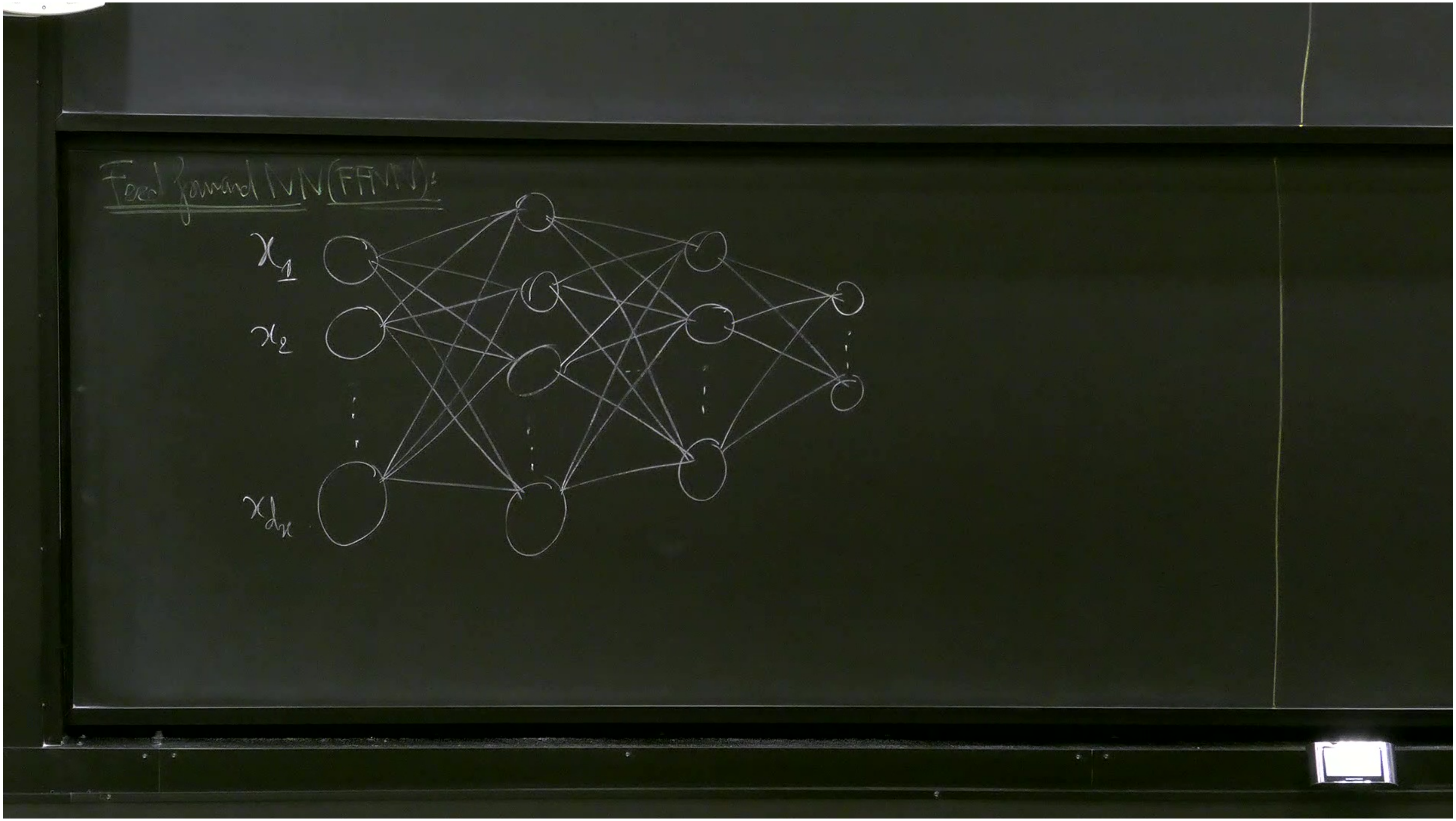
Choose a neural network

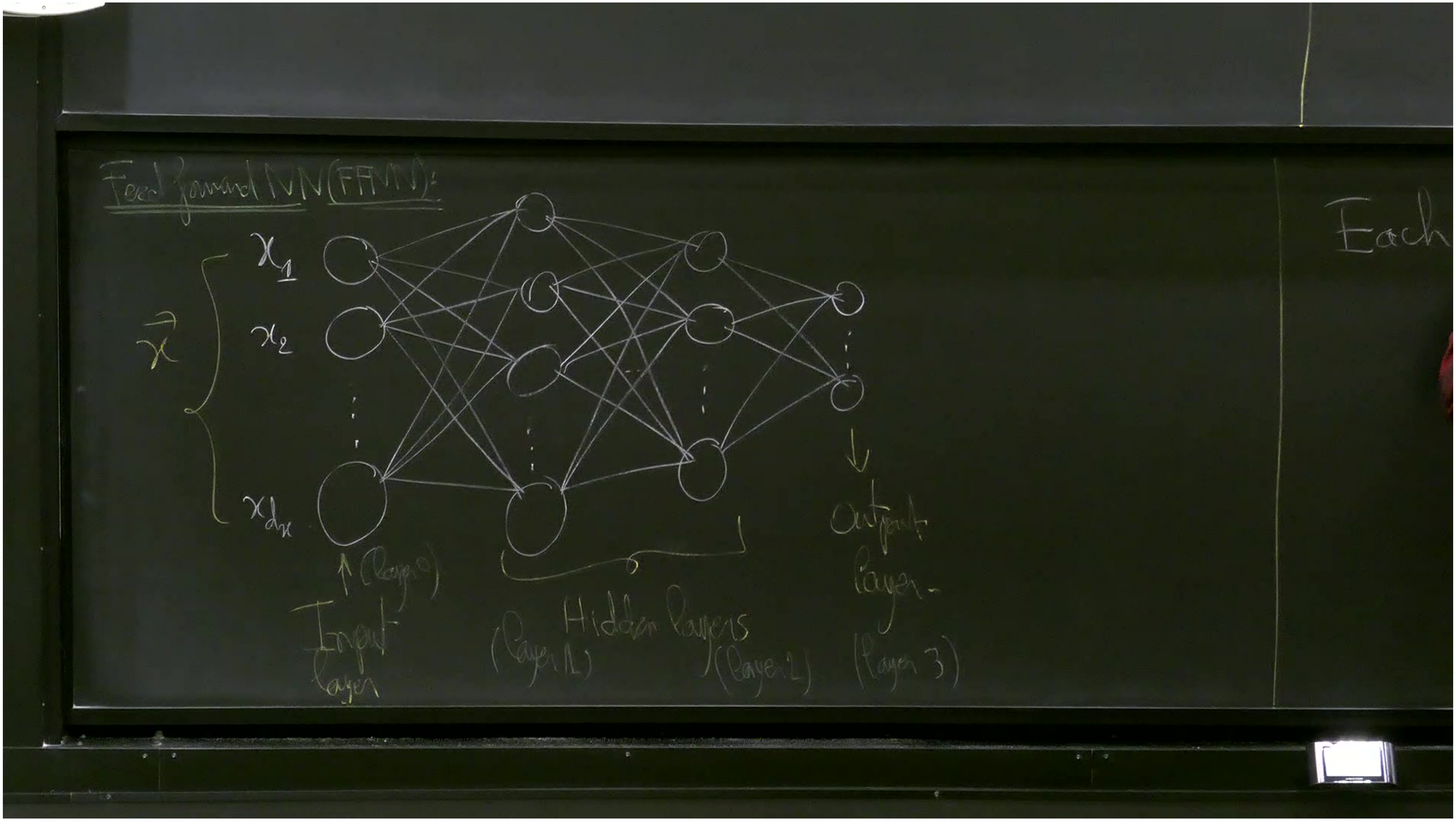


Tuning the neurons couplings (Backpropagation)



Let’s switch to the blackboard to learn the math of Feed-forward NNs


Feed forward NN (FFNN):
x_1
x_2
x_{d_x}


Feed forward NN (FFNN):
x_1
x_2
x_{d_x}
$\vec{x}$
(Layer 0)
Input layer
Hidden layers
(Layer 1)
(Layer 2)
Output layer
(Layer 3)
Each

So far, we have studied most algorithms.

Each "○" is called a "neuron"

Notation:

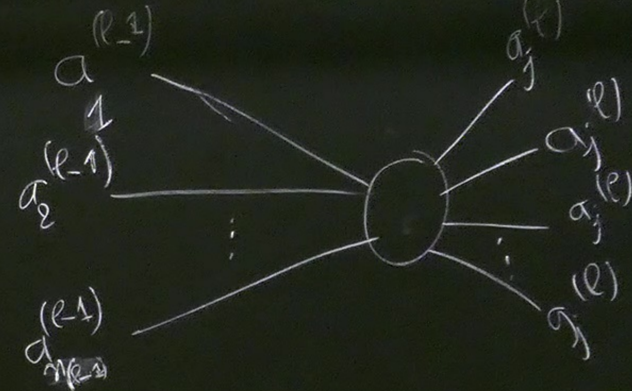
$n_\ell \equiv$ number of neurons in layer ℓ.

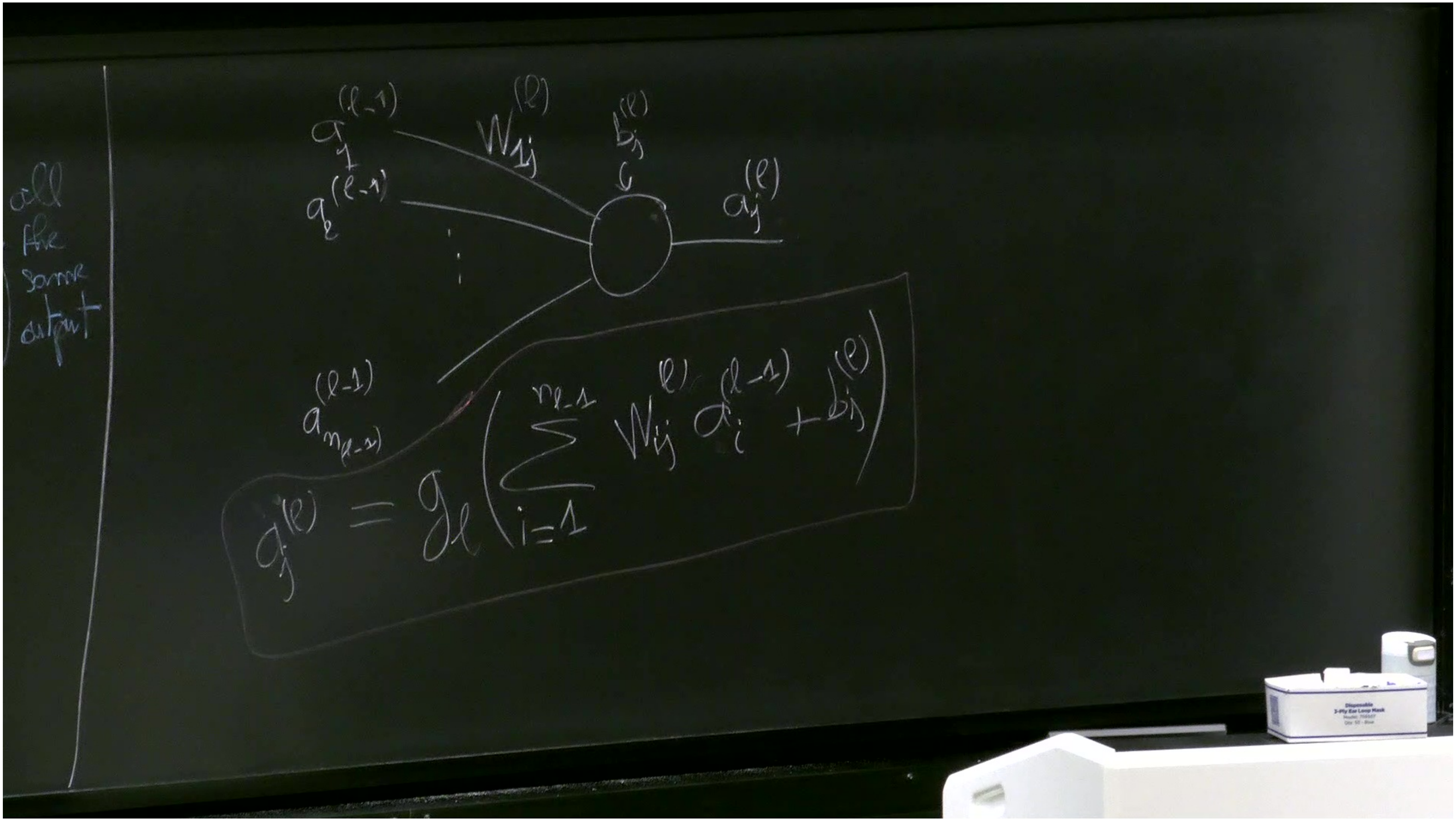
$L \equiv$ largest value of ℓ ($L = 3$ in the previous example).

$a_j^{(\ell)} \equiv$ output from the j^{th} neuron in layer ℓ.

- $n_0 = d_x$
- $n_L = d_y$
- $a_i^{(0)} = x_i \quad (\forall\, 1 \leq i \leq d_x)$

→ Let's zoom in on the j^{th} neuron in layer $\ell \geq 0$




all the same output
$a_1^{(\ell-1)}$
$a_2^{(\ell-1)}$
$a_{n_{(\ell-1)}}^{(\ell-1)}$
$W_{1j}^{(\ell)}$
$b_j^{(\ell)}$
$a_j^{(\ell)}$
$a_j^{(\ell)} = g_\ell\left(\sum_{i=1}^{n_{\ell-1}} W_{ij}^{(\ell)} a_i^{(\ell-1)} + b_j^{(\ell)}\right)$

Input layer

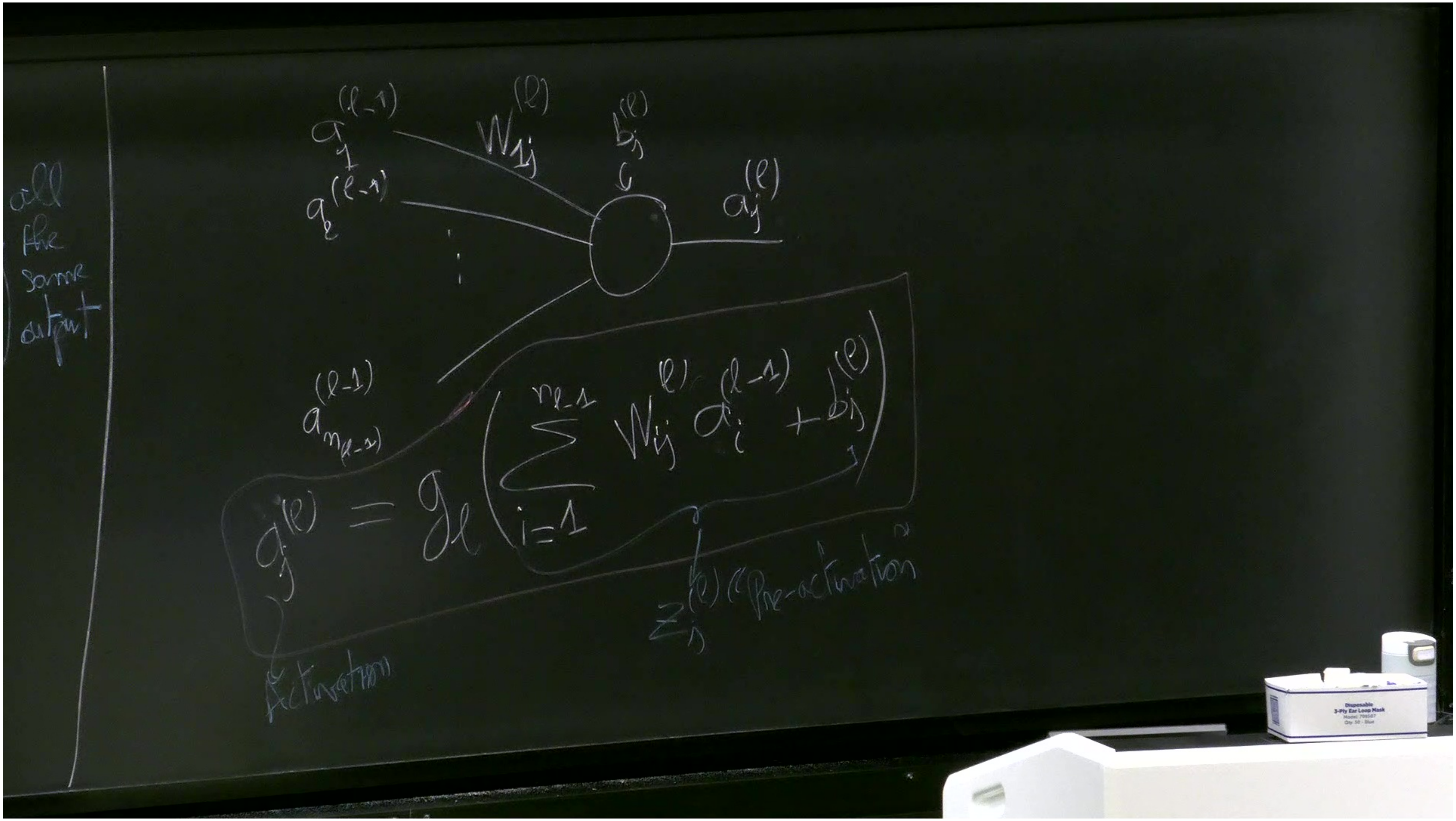
(layer 1) Hidden layers (layer 2) (layer 3)

layer

→ each g_ℓ is a non-linear "activation function"

→ Each link has a "weight" $W_{ij}^{(\ell)}$

→ Each neuron has a "bias" $b_j^{(\ell)}$


all the same output
$a_1^{(\ell-1)}$
$a_2^{(\ell-1)}$
$a_{n_{(\ell-1)}}^{(\ell-1)}$
$W_{1j}^{(\ell)}$
$b_j^{(\ell)}$
$a_j^{(\ell)}$
$a_j^{(\ell)} = g_\ell\left(\sum_{i=1}^{n_{\ell-1}} W_{ij}^{(\ell)} a_i^{(\ell-1)} + b_j^{(\ell)}\right)$
$z_j^{(\ell)}$ "Pre-activation"
Activation

$L \equiv$ largest value of ℓ (...)

$a_j^{(\ell)} \equiv$ output from the j^{th} neuron in layer ℓ.

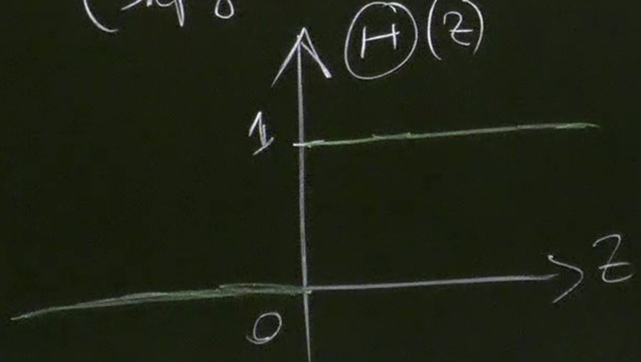
$$\vec{a}^{(\ell)} = \left(a_1^{(\ell)}, a_2^{(\ell)}, \dots, a_{n_\ell}^{(\ell)}\right).$$

$$\boxed{\vec{a}^{(\ell)} = g_\ell\left(W^{(\ell)} \vec{a}^{(\ell-1)} + \vec{b}^{(\ell)}\right)}$$

$$(1 \leq \ell \leq L)$$

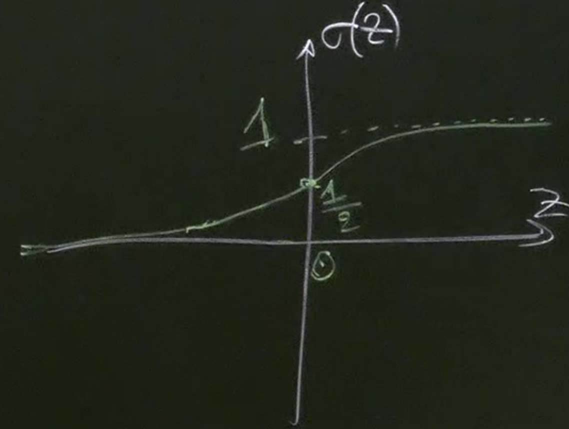
Activation function:

1. Perceptron $\Theta(z)$.
(step function)

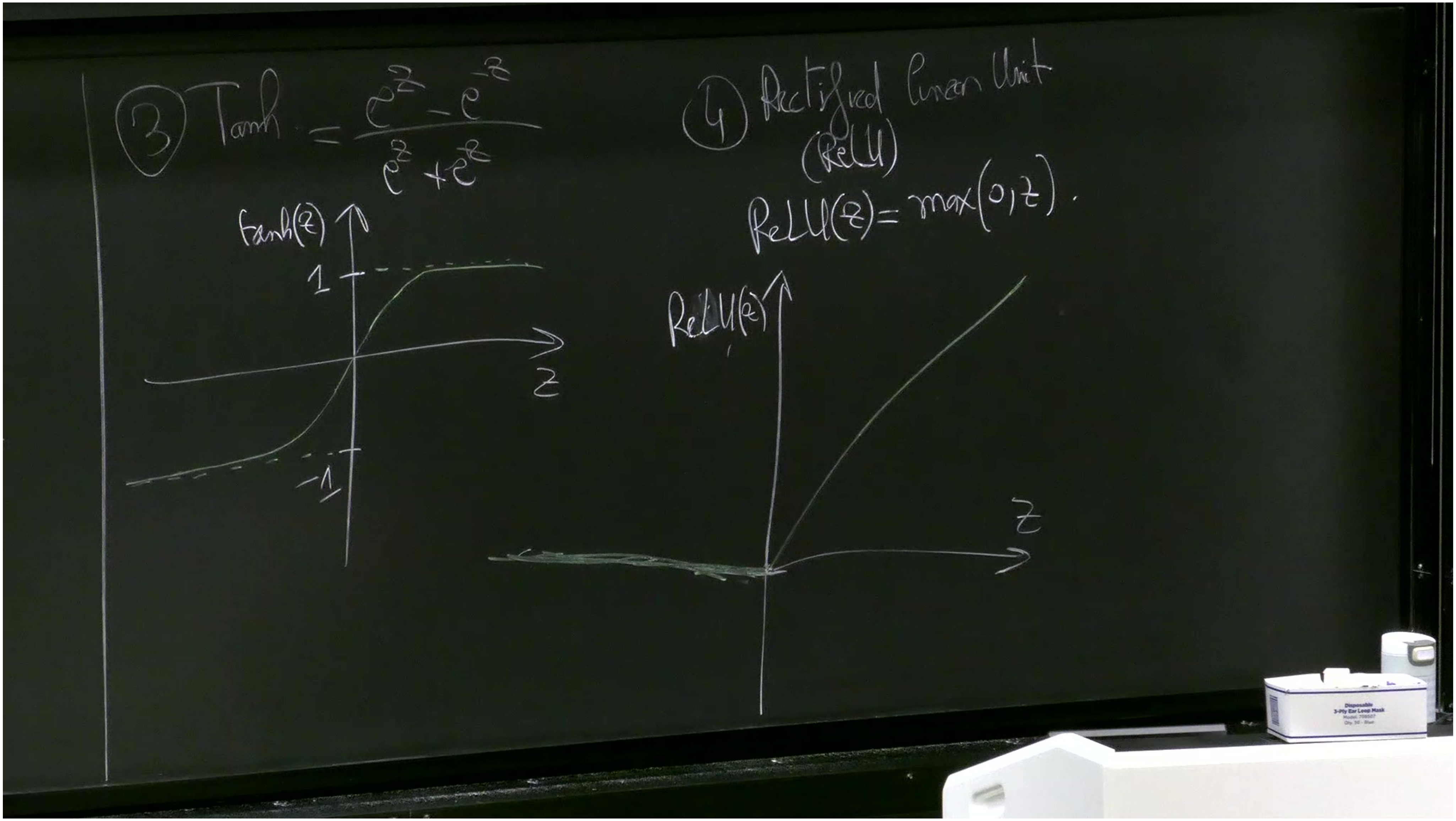


Not practical with gradient descent

2. Sigmoid $\sigma(z) = \dfrac{1}{1+e^{-z}}$



3. T


3 Tanh = $\frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$
tanh(z)
1
-1
z
4 Rectified Linear Unit
(ReLU)
ReLU(z) = max(0, z).
ReLU(z)
z

Input layer

Hidden layers

(Layer 1) (Layer 2) (Layer 3)

$\hat{y}_1$

$\hat{y}_2$

$\hat{y}_3$

$\vdots$

$\hat{y}_{d_y}$

$$\sum_{i=1}^{d_y} \hat{y}_i = 1$$

$$\mathrm{Softmax}(\vec{z})_i = \frac{\exp(z_i)}{\sum_{j=1}^{n_\ell} \exp(z_j)}.$$

$$\vec{z} = (z_1, z_2, \ldots, z_{n_\ell})$$

$a_j^{(\ell)} \equiv$ output from the j^{th} neuron in layer ℓ.

One-hot encoding:

$$y = 0, 1 \longrightarrow \vec{y} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \text{ or } \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

"0" "1"

$$y = 0, 1, 2 \longrightarrow \vec{y} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

"0" "1" "2"

Cost (Loss) functions:

(1) Mean-squared error (MSE)

$$C_{MSE} = \frac{1}{2|D|} \sum_{x \in D} \underbrace{\sum_{i=1}^{L} \left(a_i^{(L)}(x) - y_i(x) \right)^2}_{\| \vec{a}^{(L)}(x) - \vec{y}(x) \|_2^2}$$

② Cross entropy (CE): "Classification"

$$C_{CE} = \frac{-1}{|D|}\sum_{x\in D}\sum_{i=1}^{d_y}\left[y_i(\vec{x})\cdot\log\left(a_i^{(L)}(\vec{x})\right) + \left(1-y_i(\vec{x})\right)\log\left(1-a_i^{(L)}(\vec{x})\right)\right]$$

$$\frac{\partial}{\partial a}\left(-y\log(a) - (1-y)\log(1-a)\right) = 0$$

$$\Rightarrow -\frac{y}{a} + \frac{1-y}{1-a} = 0 \Rightarrow a = y$$

Cost (Loss) function:

① Mean-squared error (MSE) « Regression »

$$C_{MSE} = \frac{1}{2|D|} \sum_{x \in D} \underbrace{\sum_{i=1}^{L} \left(a_i^{(L)}(x) - y_i(x)\right)^2}_{\| \vec{a}^{(L)}(x) - \vec{y}(x) \|_2^2}$$

② C

$C_E = \frac{-1}{|D|}$