Lecture 2

Introduction to Neural Networks

Minh-Tan Pham, Matthieu Le Lain

BUT2 INFO, 2023-2024

IUT de Vannes, Université Bretagne Sud

minh-tan.pham@univ-ubs.fr

Planning

- Image classification
- Linear Classifier
- Neural networks
- Lab Assignments

Image Classification

Image Classification

- **Motivation of image classification (recognition):** assign one label (from a fixed set of categories) to each input image

- **Example:** given the following image, assign probabilities to 4 labels {dog, cat, hat, mug}

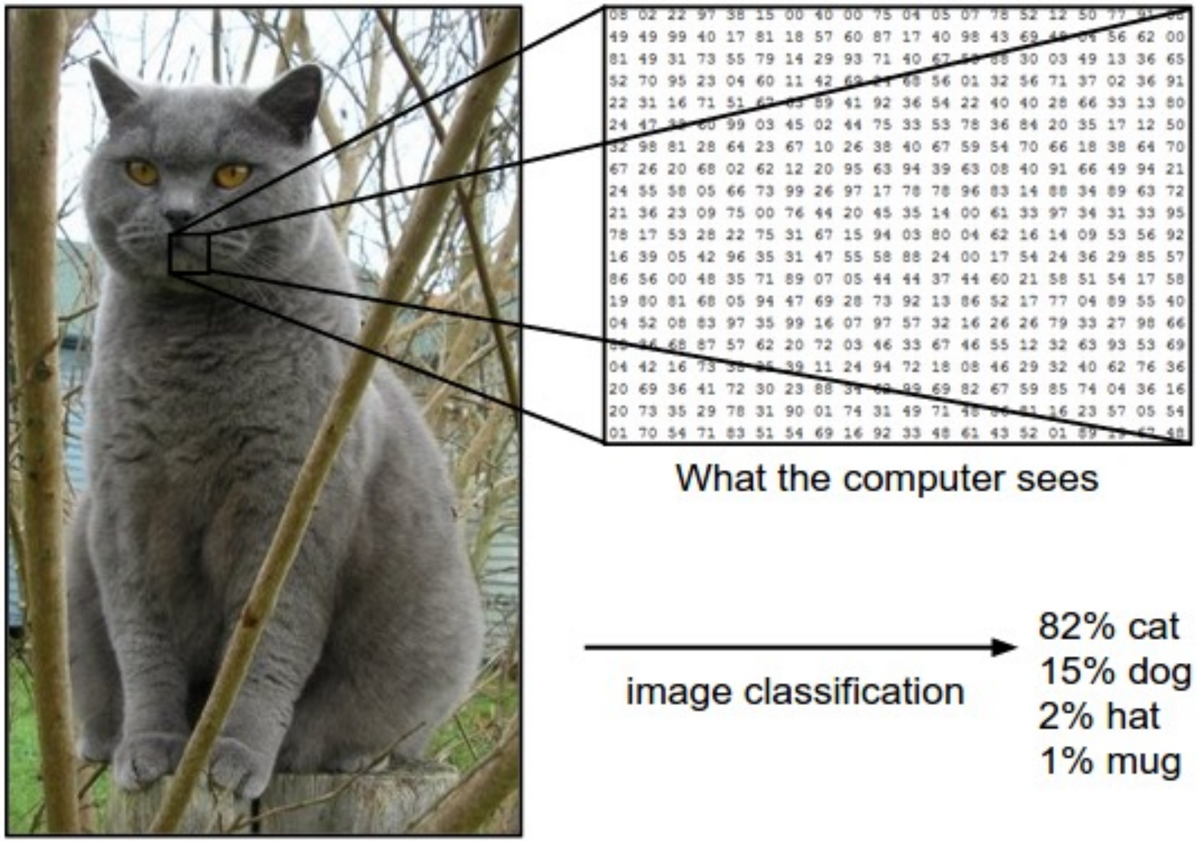


Image credits: [cs231n]

Image Classification

- **Challenges:** the task is trivial for human, but very challenging for a computer !

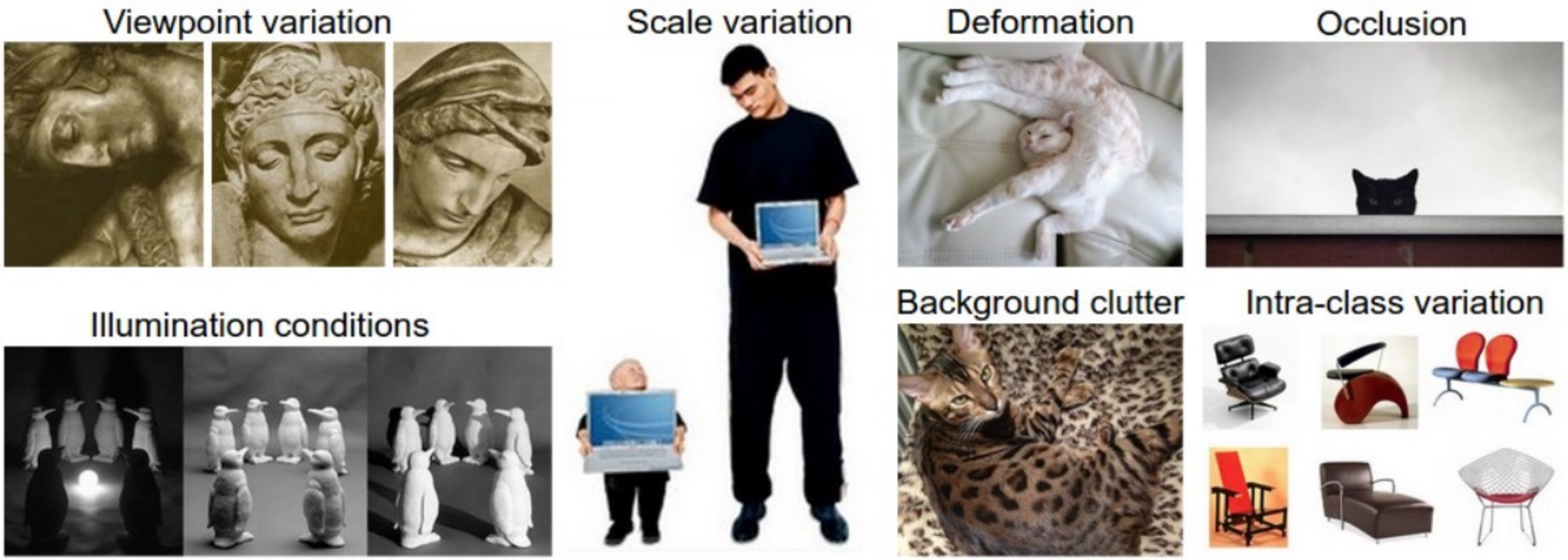


Image Classification

- **Data-driven approach:** learn from a training dataset including labeled images

In practice, we may have thousands of classes and hundreds of thousands of images per class.

Nearest Neighbor classifier

- **Principle:** compare the test image to each training image ➔ choose the closest one (lowest distance)
- L1 distance $$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$
- L2 distance $$d_2(I_1, I_2) = \sqrt{\sum_p (I_1^p - I_2^p)^2}$$

test image

56	32	10	18
90	23	128	133
24	26	178	200
2	0	255	220

\-

training image

10	20	24	17
8	10	89	100
12	16	178	170
4	32	233	112

=

pixel-wise absolute value differences

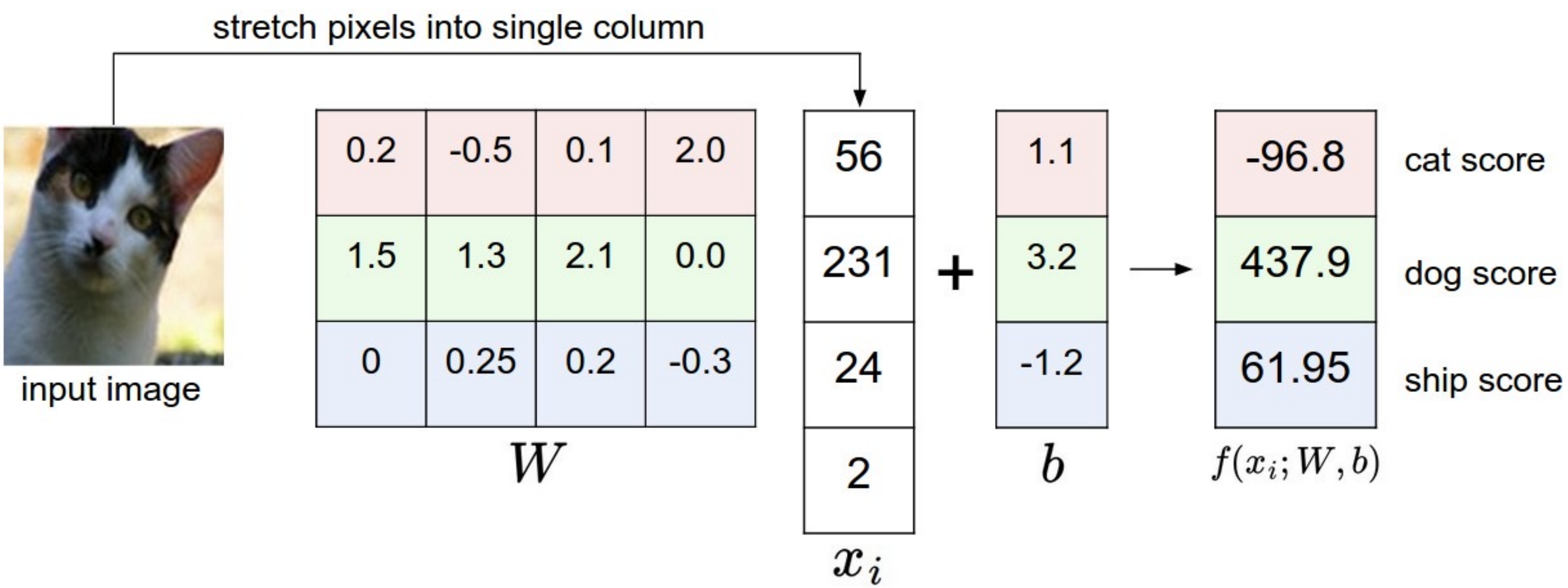
46	12	14	1
82	13	39	33
12	10	0	30
2	32	22	108

→ 456

➔ It does not work because of the previous challenges !!!!

Linear Classification

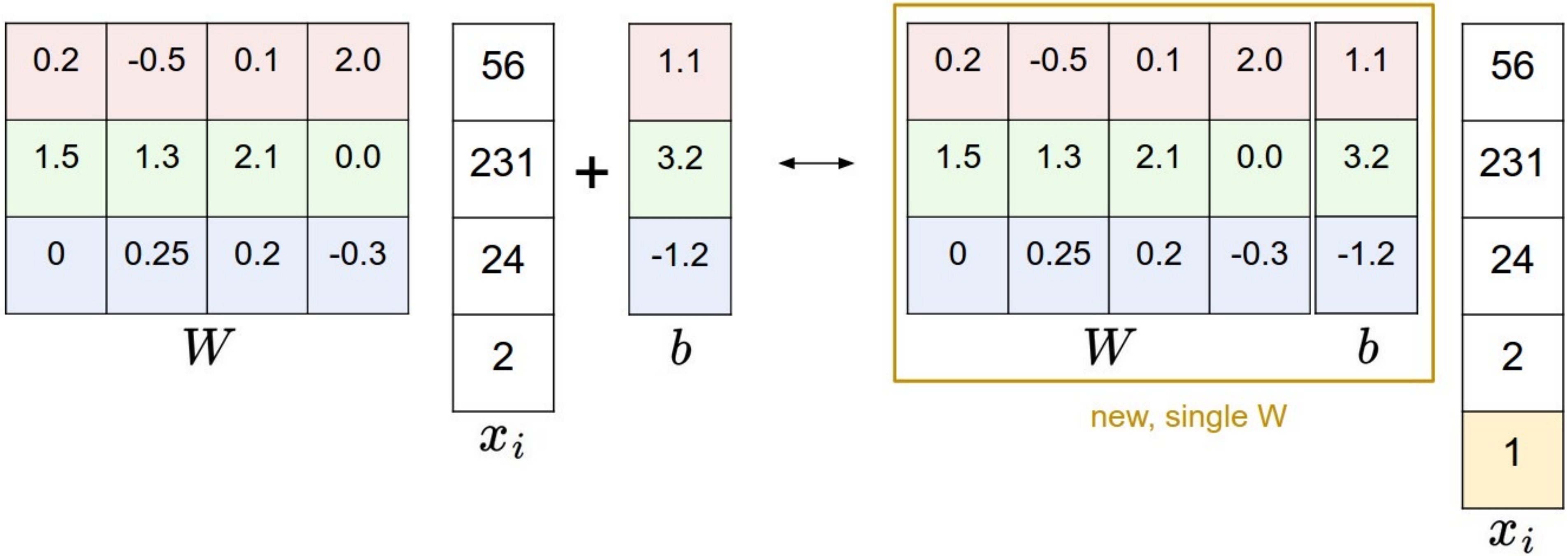
- **Principle:** use the linear mapping $f(x_i, W, b) = Wx_i + b$



- We need to learn the parameters (weights **W** and bias vector **b**) using the training dataset.
- Once they are learned, simply pass the test image and compute the class probabilities
- Matrix multiplication and addition → very fast

Linear Classification

- **Bias trick:** in common, we simplify the couple **(W,b)** as one



- We extend the vector x_i with one additional dimension (bias dimension)
- Our linear mapping now becomes

$$f(x_i, W) = W x_i$$

Linear Classification

- **Training:** input each pair of image and label $(\boldsymbol{x_i}, \boldsymbol{y_i})$ to update weights **W** (by using a loss function). The training objective is to minimize the loss

- Usually, we use the **cross-entropy loss.**

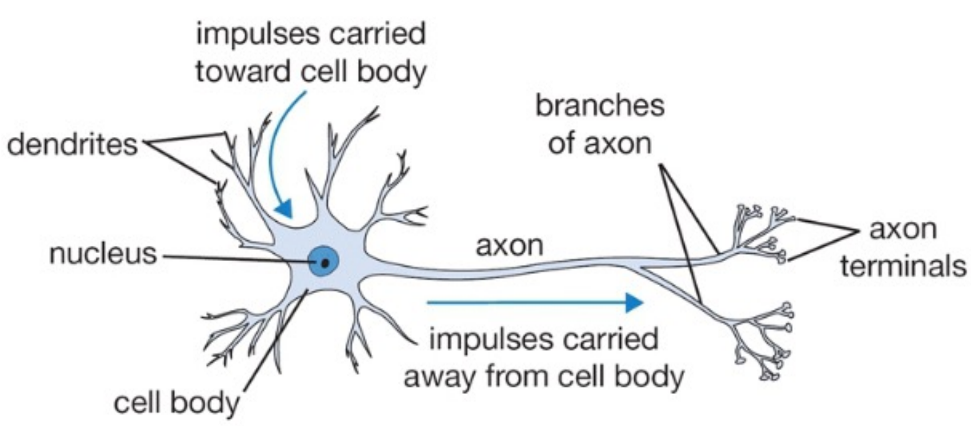
- Testing: use the computed W to predict a label $\boldsymbol{\hat{y}_k}$ for a test image $\boldsymbol{x_k}$

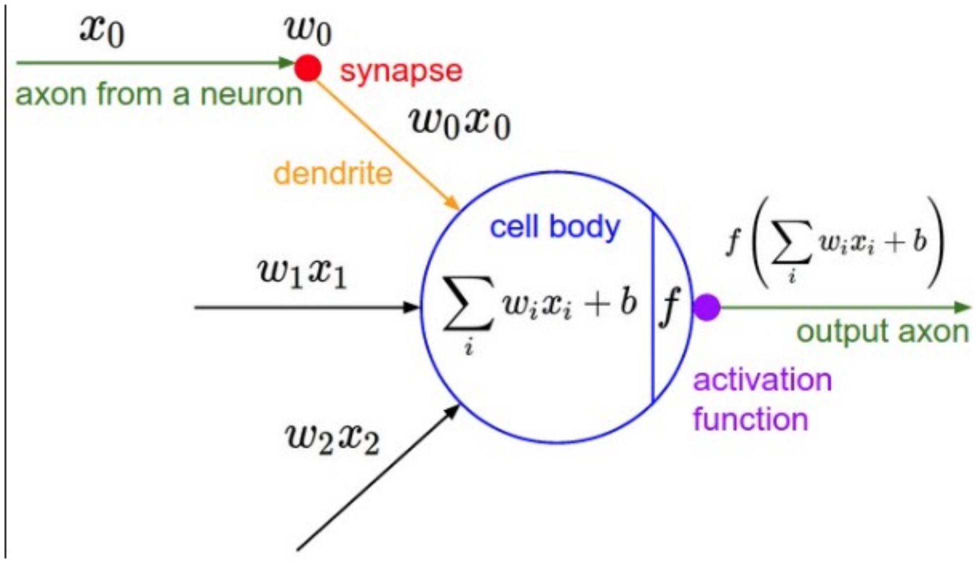
- **Further reading:** *loss function, gradient descent, back-propagation*

Neural networks

Neural networks

Inspired by the goal of modeling biological neural systems with an 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)
- Biological neuron models are much more complex !!!

Neural networks

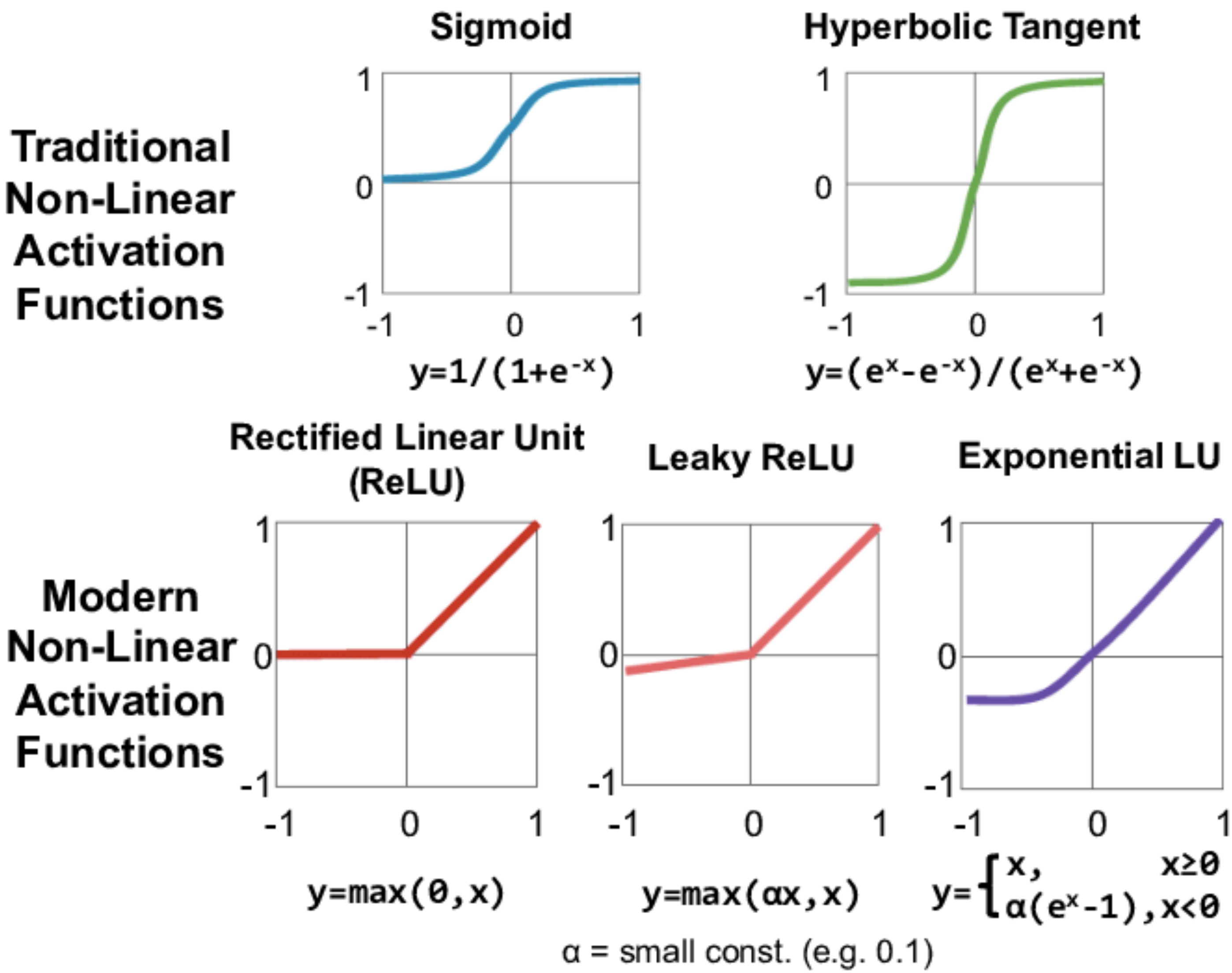
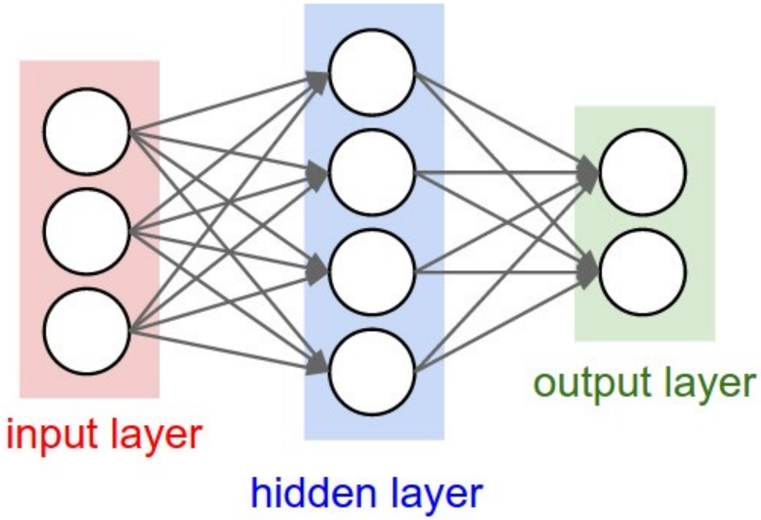


Image credits: Efficient Processing of Deep Neural Networks: A Tutorial and Survey

Neural networks

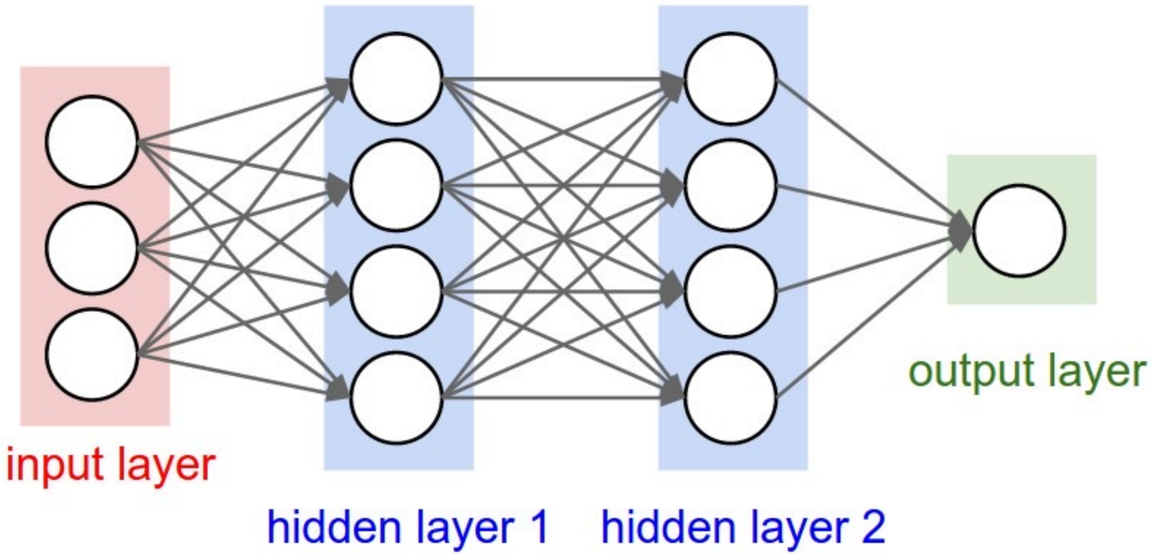
One-layer neural networks



- This network has 4 + 2 = 6 neurons (not counting the inputs), [3 x 4] + [4 x 2] = 20 weights and 4 + 2 = 6 biases, for a total of 26 learnable parameters.
- Attention: the output layer neuros commonly do not have an activation function !!! It is usually take to represent the class score (i.e. classification layer)

Neural networks

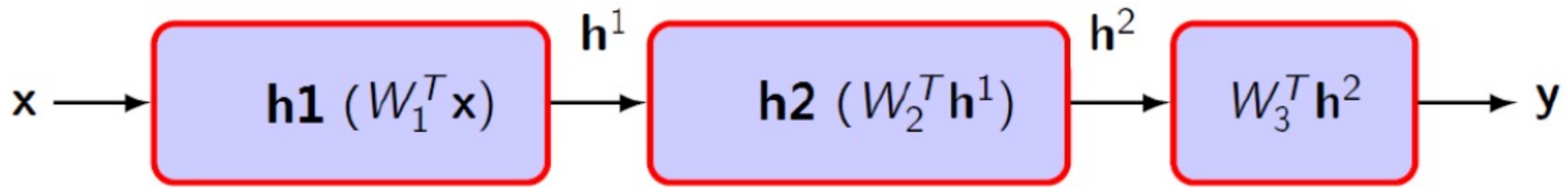
Two-layer neural networks



- The network has 4 + 4 + 1 = 9 neurons, [3 x 4] + [4 x 4] + [4 x 1] = 12 + 16 + 4 = 32 weights and 4 + 4 + 1 = 9 biases, for a total of 41 learnable parameters.
- Also named **Multi-Layer Perceptrons (MLPs)**

Neural networks

Simplified notations

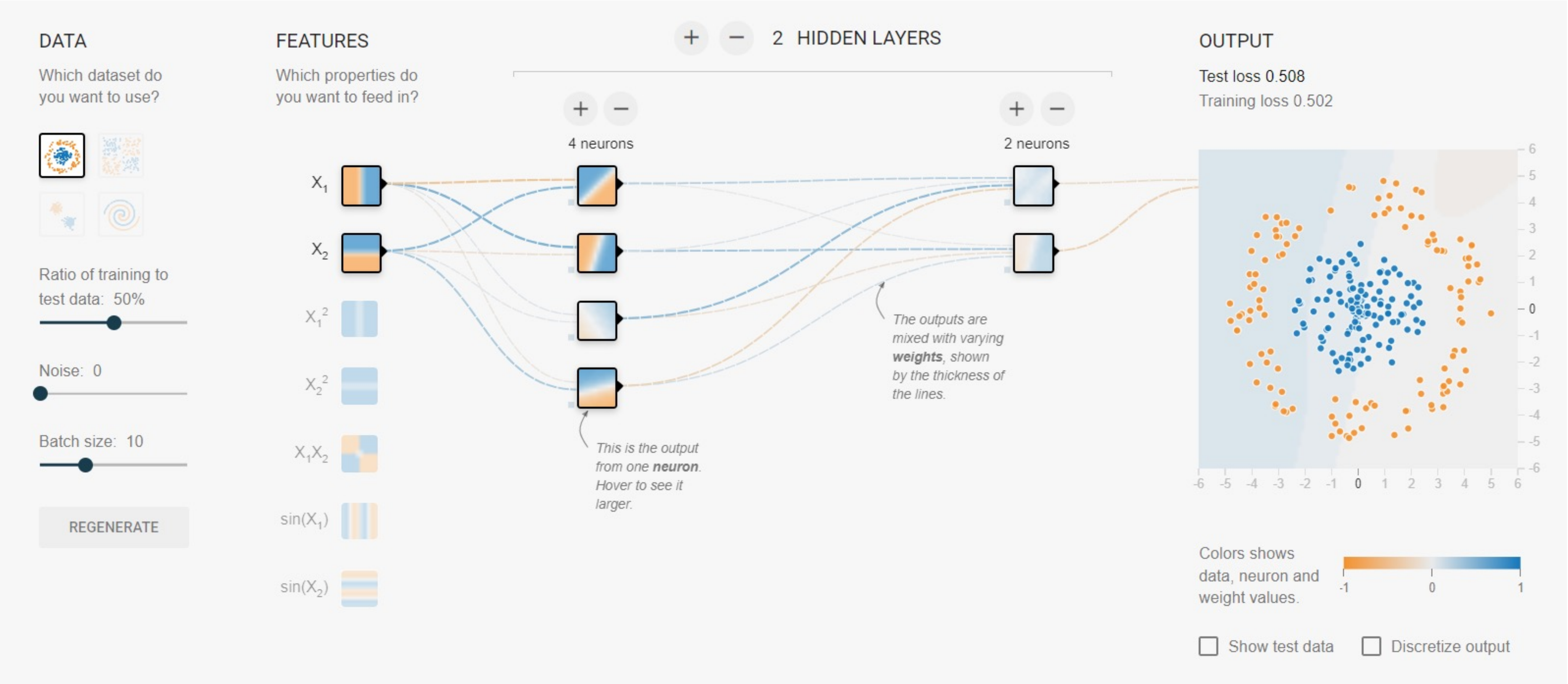


- **x**: input
- **y**: output
- $\boldsymbol{W_i}$: set of parameters (weights) of the i^{th} layer
- $\boldsymbol{h^i}$: output of the i^{th} layer
- $\boldsymbol{hi}$: activation function of the i^{th} layer (i.e., *sigmoid, ReLU, leaky ReLU, etc.*)

Neural networks

Visualization of neural networks

https://playground.tensorflow.org/



Lab assignments

Practical lab: Training neural networks on the MNIST dataset (for handwritten digit recognition) using Pytorch.

- **Download, complete and submit** the notebook from Moodle **(R3A13_lab1_assign.ipynb)**

- Note that you need to submit **your compiled notebook with all outputs**

References and Sources

- **Convolutional neural networks for visual recognition** - Stanford

  https://cs231n.github.io/
- **Neural networks and deep learning** (free online book)

  http://neuralnetworksanddeeplearning.com/index.html
- **Machine learning course** – Oxford

  https://www.cs.ox.ac.uk/people/nando.defreitas/machinelearning/
- **Neural network course** - Hugo Larochelle

  https://info.usherbrooke.ca/hlarochelle/neural_networks/content.html
- **Deep learning course** – François Fleuret

  https://fleuret.org/dlc/