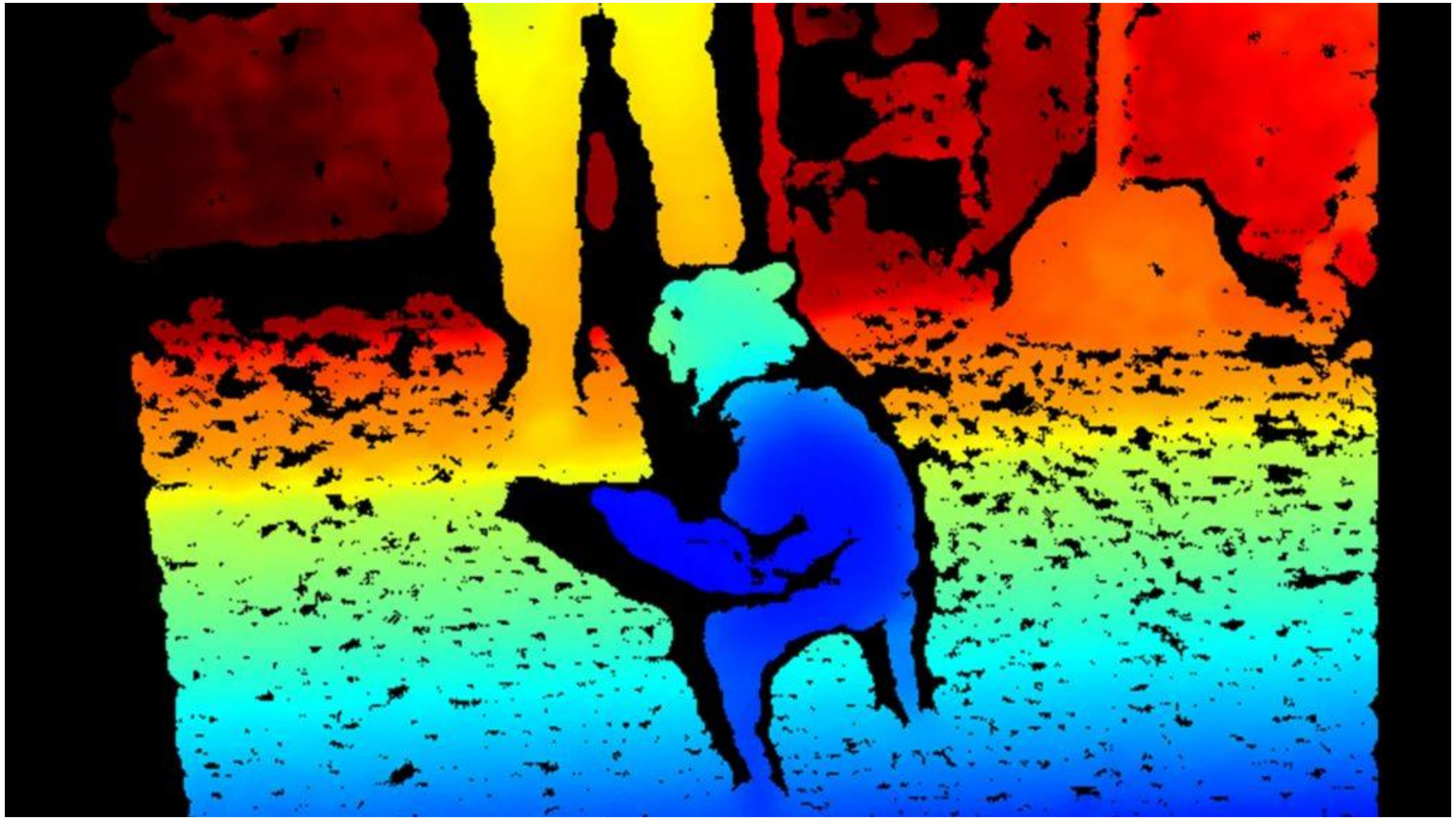
NN basics



References

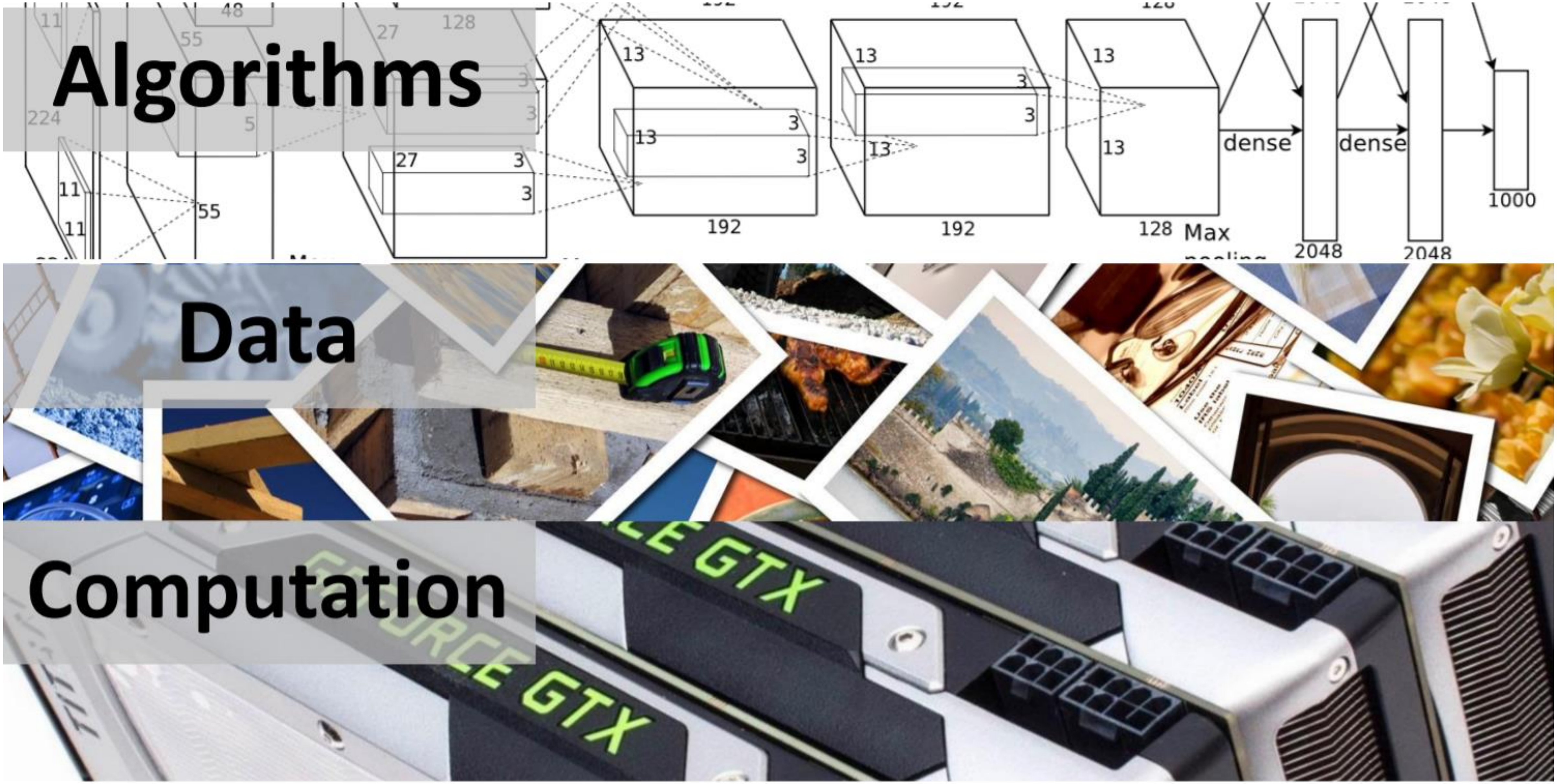
- http://cs231n.stanford.edu/index.html
- http://www.cs.cornell.edu/courses/cs5670/2019sp/lectures/lectures.html
- http://www.cs.cmu.edu/~16385/

- What is a neural network?
- The object recognition challenge
- Neural networks history.

What is a neural network

- **Artificial neural networks** (**ANN**) are computing systems vaguely inspired by the biological neural networks that constitute animal brains. Such systems "learn" to perform tasks by considering examples, generally without being programmed with task-specific rules.
  - [Wikipedia]

What does a NN needs?

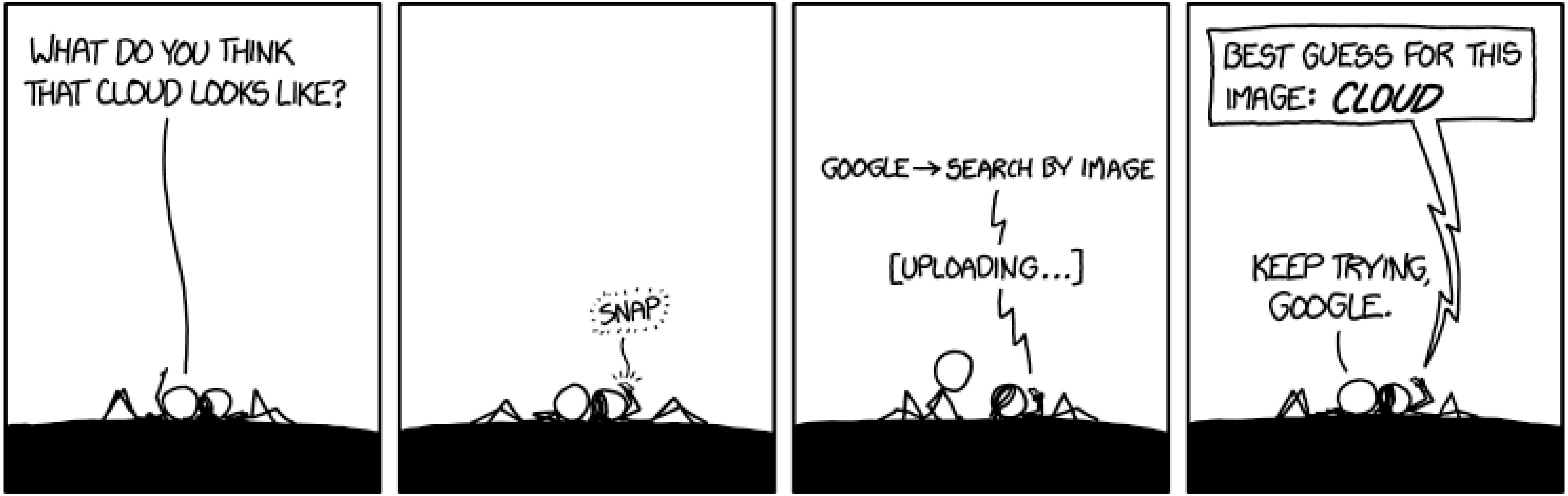


What a neural network can do?

- Image based:
  - Object recognition
  - Human pose detection
  - 3D reconstruction from a signal image
  - Image captioning
  - Style transfer
- Non image based:
  - Language translation
  - Game playing
- And much-much more…

Object recognition

Object recognition



Human pose detection

3D reconstruction from a single image

Image captioning

a little girl sitting on a bench holding an umbrella.

a herd of sheep grazing on a lush green hillside.

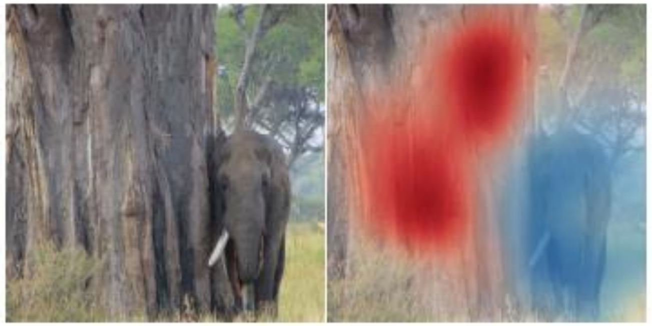
a close up of a fire hydrant on a sidewalk.

a yellow plate topped with meat and broccoli.

a zebra standing next to a zebra in a dirt field.

a stainless steel oven in a kitchen with wood cabinets.

two birds sitting on top of a tree branch.



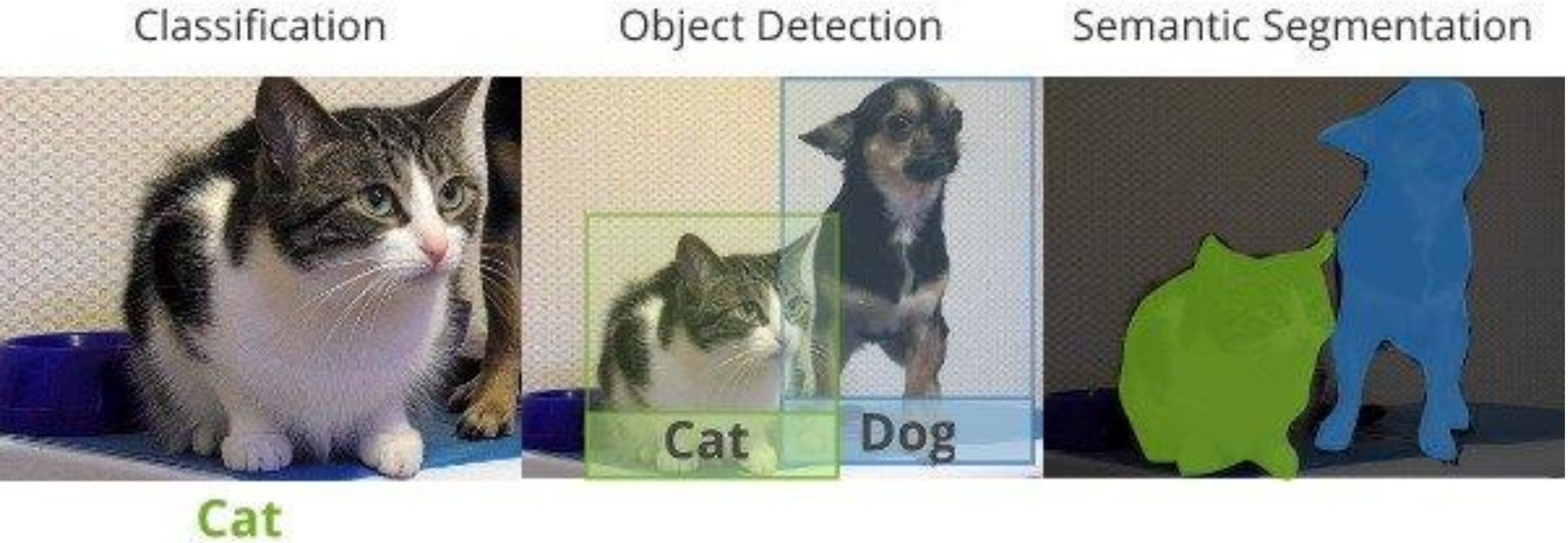
an elephant standing next to rock wall.

a man riding a bike down a road next to a body of water.

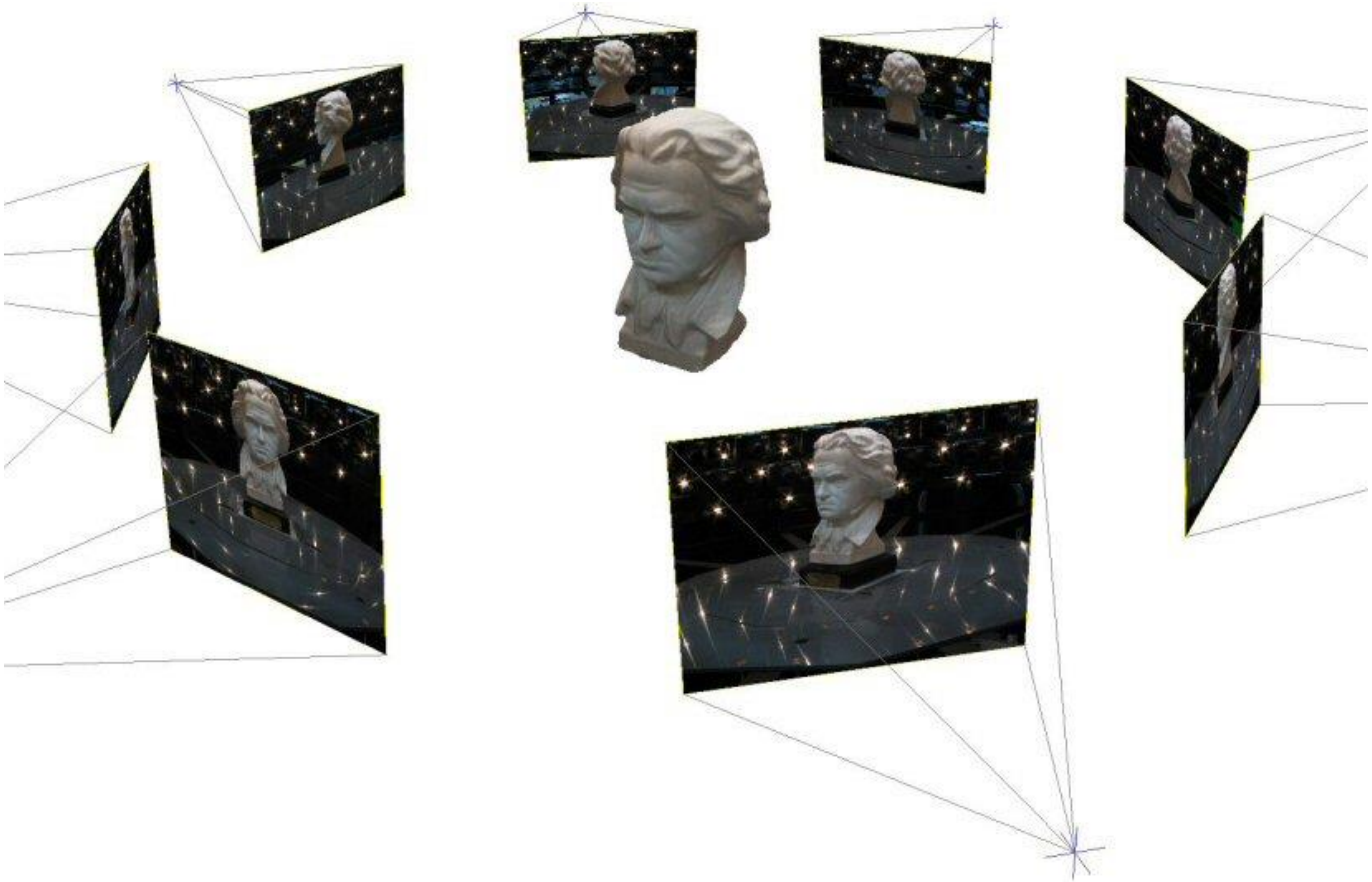
Style transfer

Object recognition challenges

- As we've seen before- object recognition is hard!



Challenge: variable viewpoint



Challenge: variable illumination

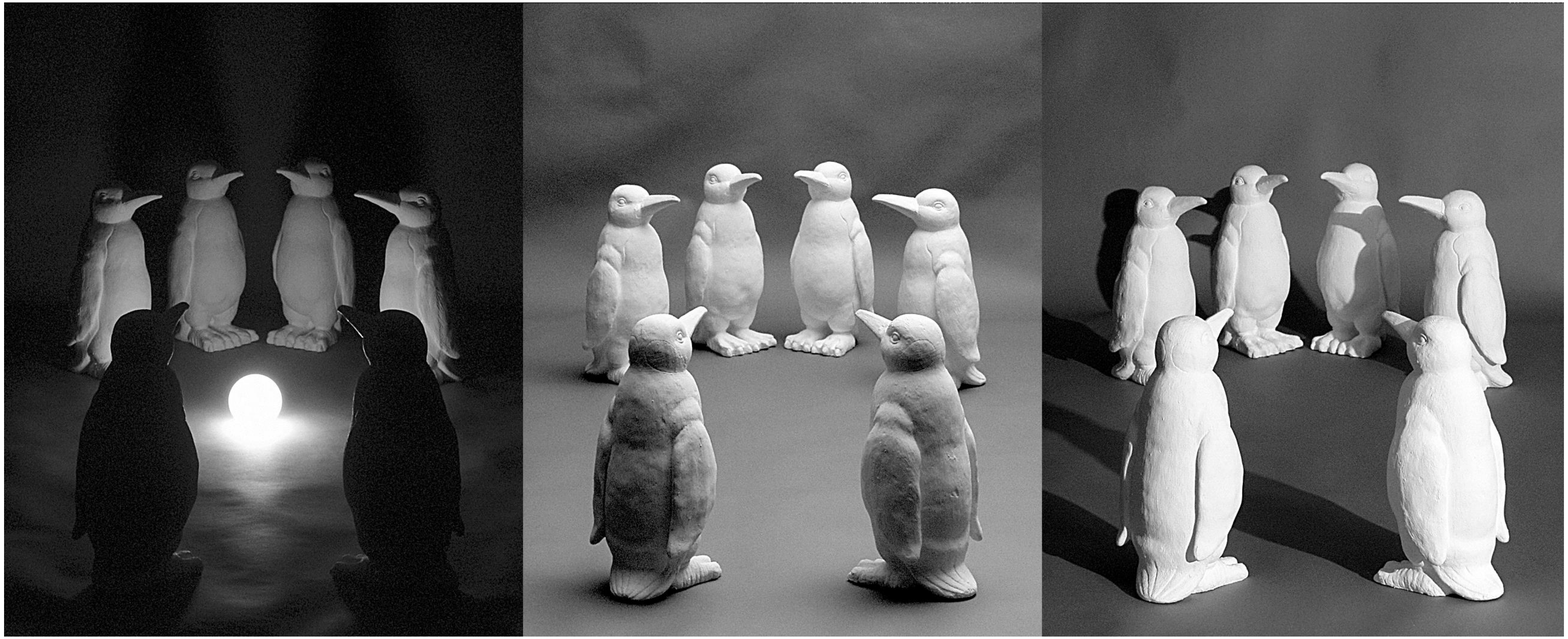


image credit: J. Koenderink

Challenge: scale

Challenge: deformation

Challenge: occlusion

Challenge: background clutter

Challenge: intra-class variations

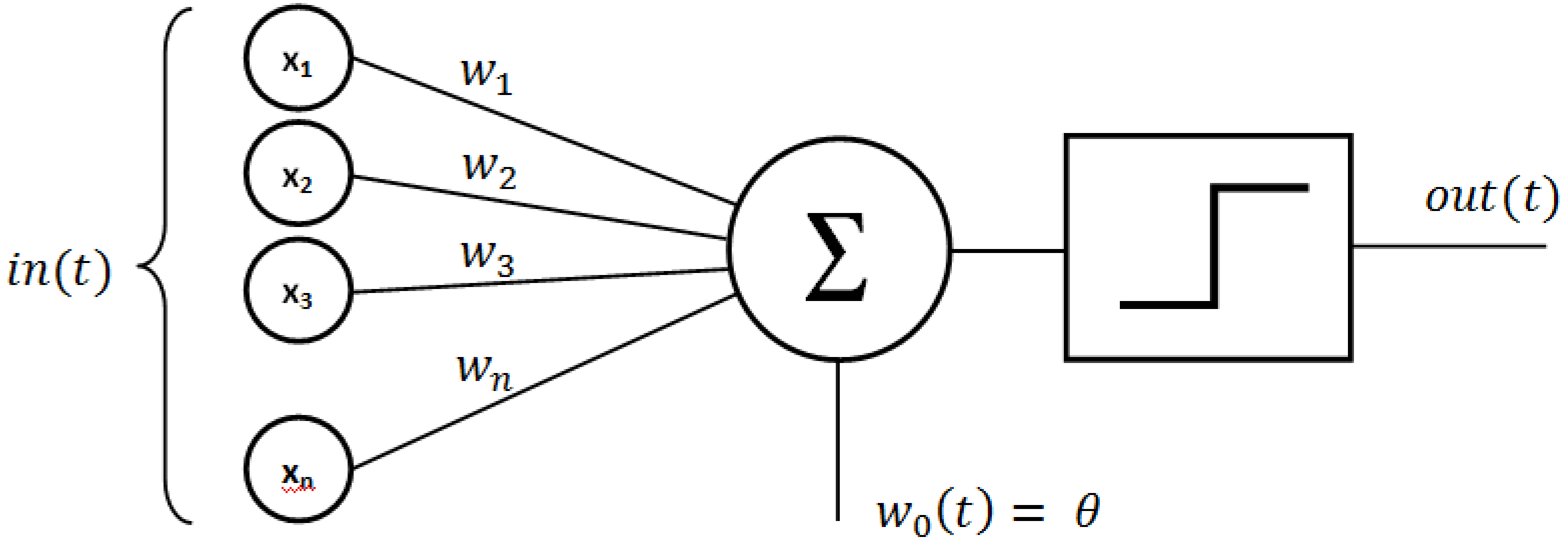
Svetlana Lazebnik

Object recognition challenges

- We’ve already seen that this is a hard problem to tackle with “classic” CV algorithms like SIFT and template matching.
  - Template matching does a relatively good job to find the same template instance in an image.
  - SIFT can extend this to find the instance with changing viewpoint/scale/illumination and rotation.
- What happens when want to find similar object that are not the same?
  - NN for the saving!

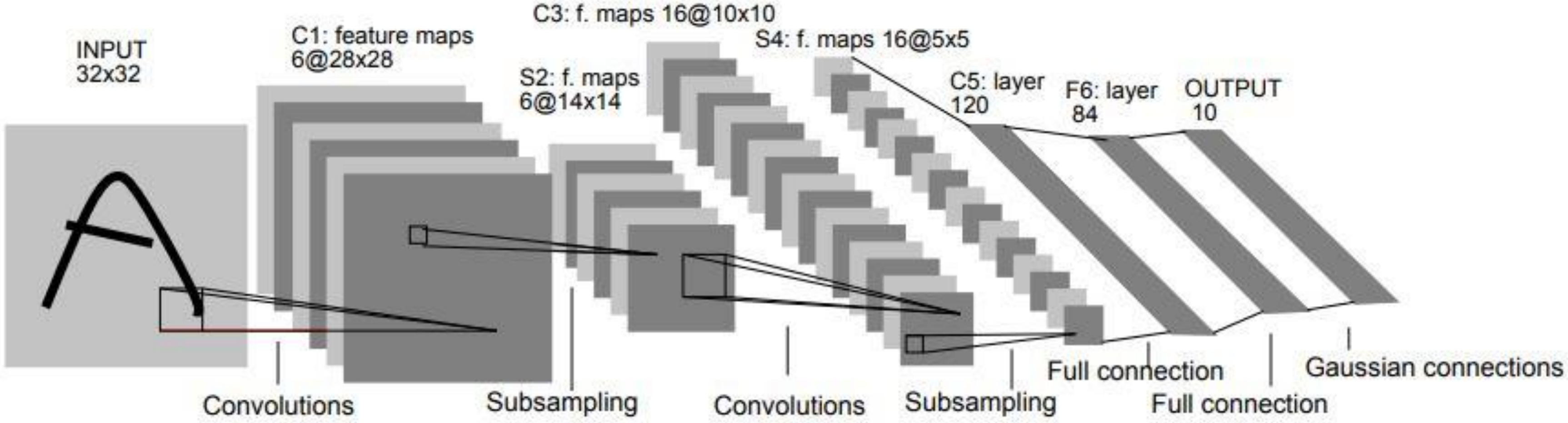
perceptron

- The basic building block of all NN.
- First introduced in 1958 at Cornell Aeronautical Laboratory by Frank Rosenblatt.
- We will talk more about it in a moment…



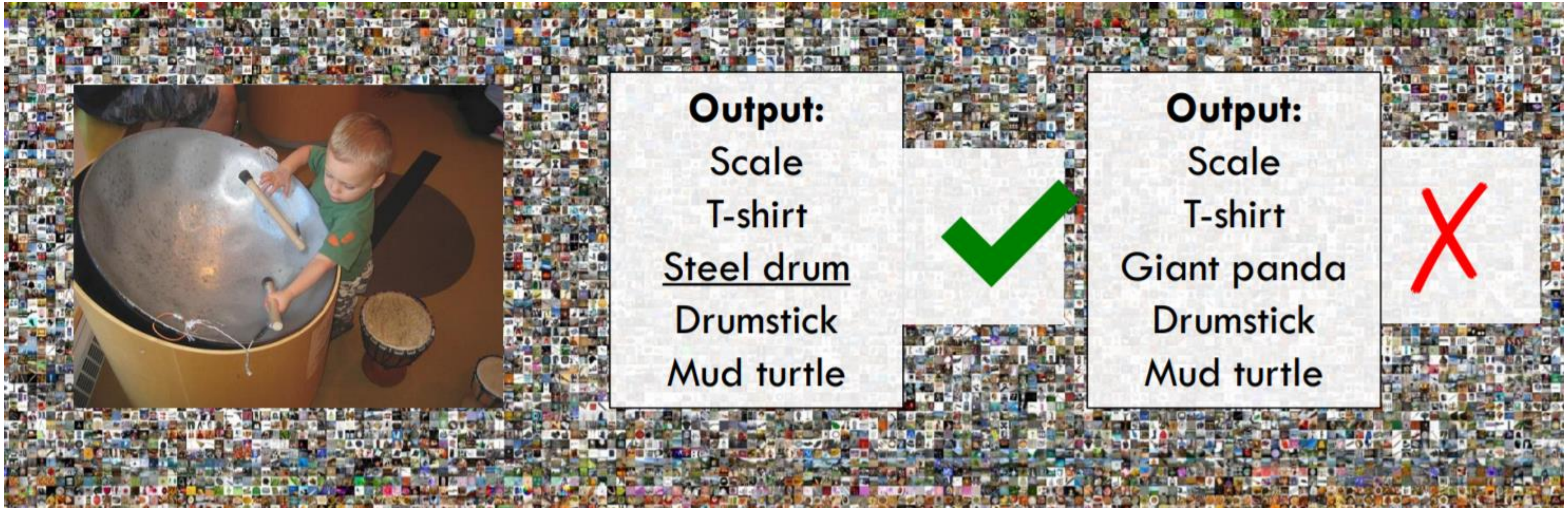
MNIST + LeNet-5

- MNIST is a large dataset of handwritten digits used in training of LeNet-5.
- LeNet-5 is the first known NN to solve a major computer vision problem:
  - Classifies digits, was applied by several banks to recognize hand-written numbers on checks.
  - Used 7 trainable layers with a total of **60K** params (sounds a lot?).
  - Yann LeCun at el., 1998, 23000 citations.

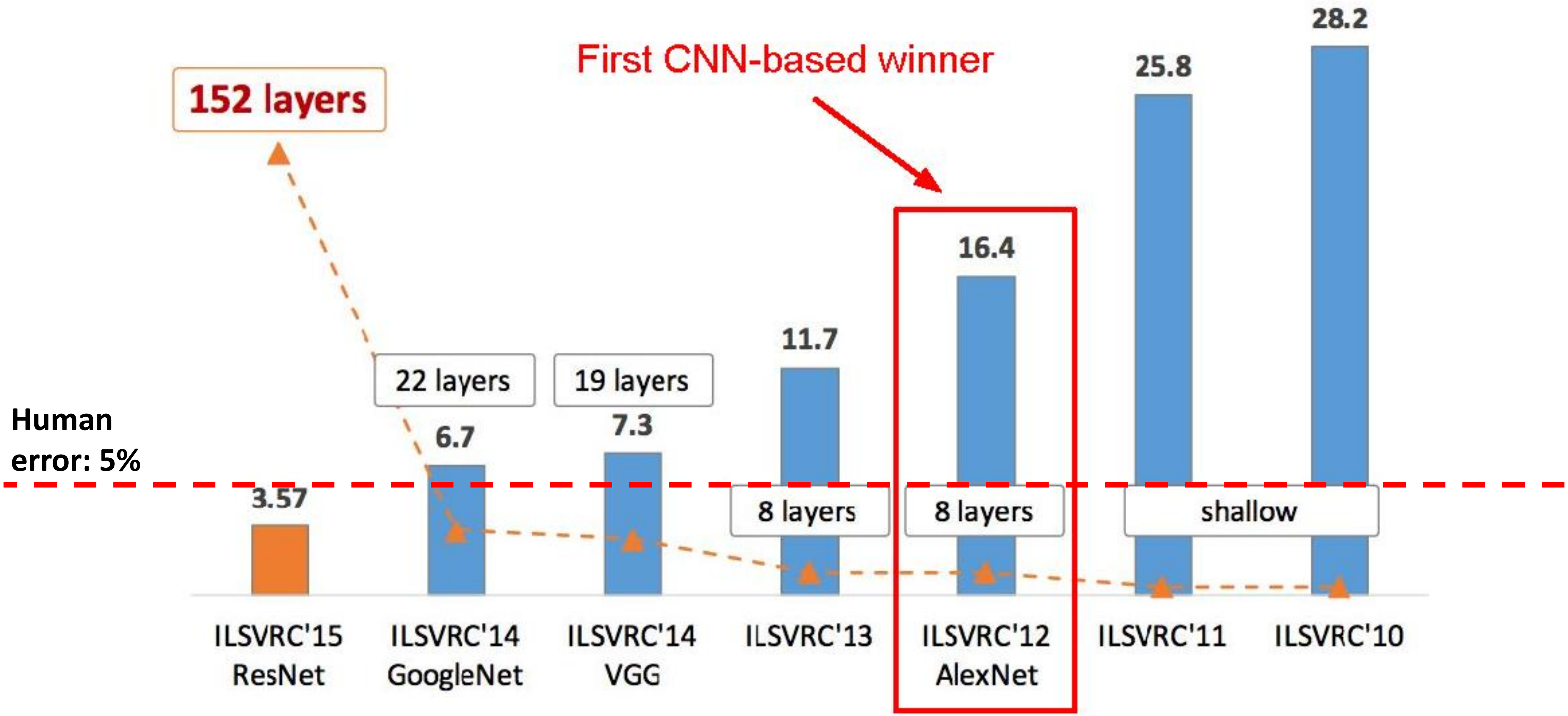


IMAGENET Large Scale Visual Recognition Challenge (ILSVRC)

- ImageNet is an image database most known for its ILSVRC challenge, and specifically for the image classification contest:
  - 1000 object classes
  - 1,431,167 images
  - Winner has the minimum mean labeling error out of 5 gausses for a given unknown test set.



ILSVRC winners



perceptron

hyperplane

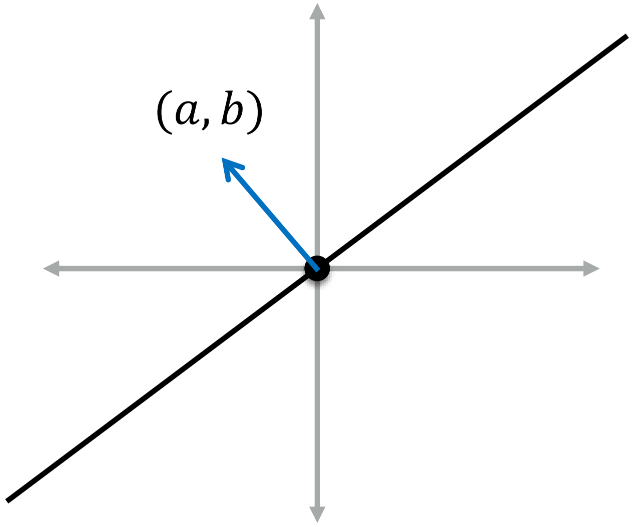
- Paramtrization of a line in 2D:

$$ax + by + c = 0$$

  - if $c = 0$:

$$ax + by = 0 \leftrightarrow (a, b) \cdot (x, y) = 0 \leftrightarrow (a, b) \perp (x, y)$$

    - (a, b) defines the normal to the line



hyperplane

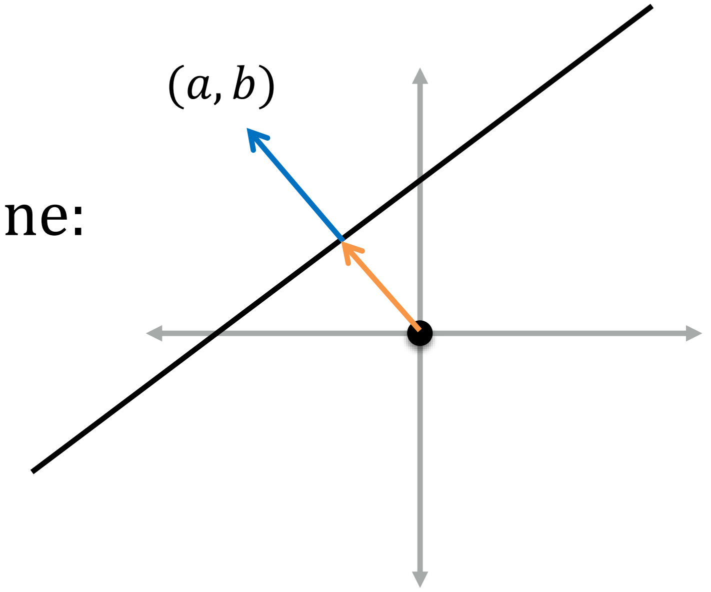
- Paramtrization of a line in 2D:

$$ax + by + c = 0$$

  - if $c = 0$:

$$ax + by = 0 \leftrightarrow (a, b) \cdot (x, y) = 0 \leftrightarrow (a, b) \perp (x, y)$$

    - (a, b) defines the normal to the line
  - if $c \neq 0$:
    - This is the **bias** factor.
    - Defines the distance of $(0{,}0)$ from the line:
      - Point-line distance: $\mathrm{d} = \frac{|ax+by+c|}{\sqrt{a^2+b^2}}$
    - $bias = \frac{|c|}{\sqrt{a^2+b^2}}$



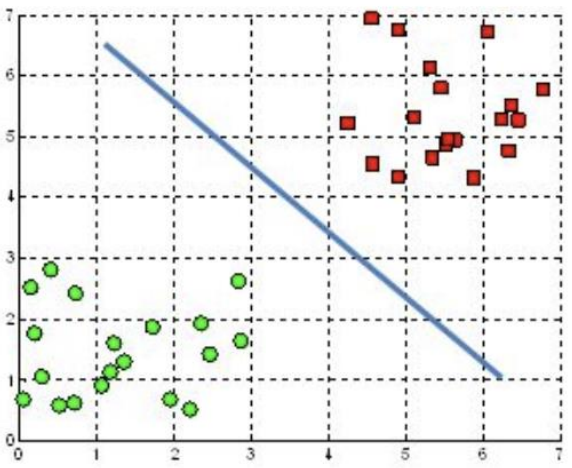
hyperplane

- This is the same for 3D representation of a plane as well:

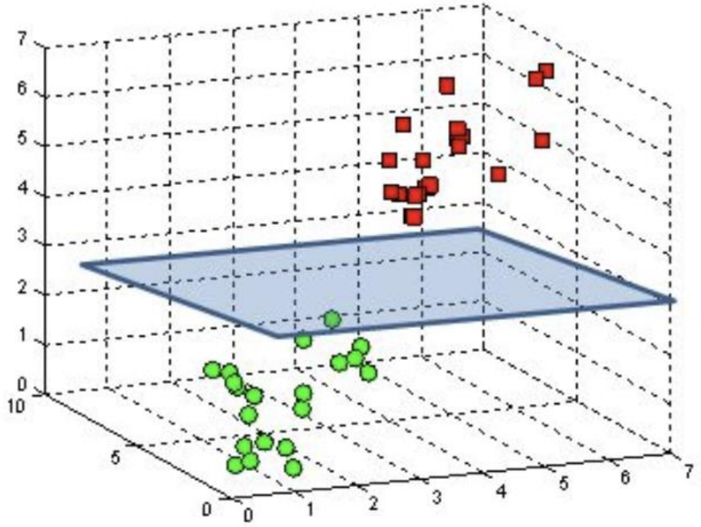
$$ax + by + cz + d = 0$$

  - (a, b, c) defines the normal to the plane, d defines the bias of the plane from $(0,0,0)$.
- And the same representation can be done for ND space. The ND plane is called a **hyperplane.**

A hyperplane in $\mathbb{R}^2$ is a line



A hyperplane in $\mathbb{R}^3$ is a plane



hyperplane

- Writing the hyperplane representation vector vise will result the equation below:

$$[w_1 \ \cdots w_n]\begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} + b = w^T x + b = 0$$

- Points x above the hyperplane (in the direction of the normal) will result in $w^T x + b > 0$, and points x below the hyperplane will result in $w^T x + b < 0$.

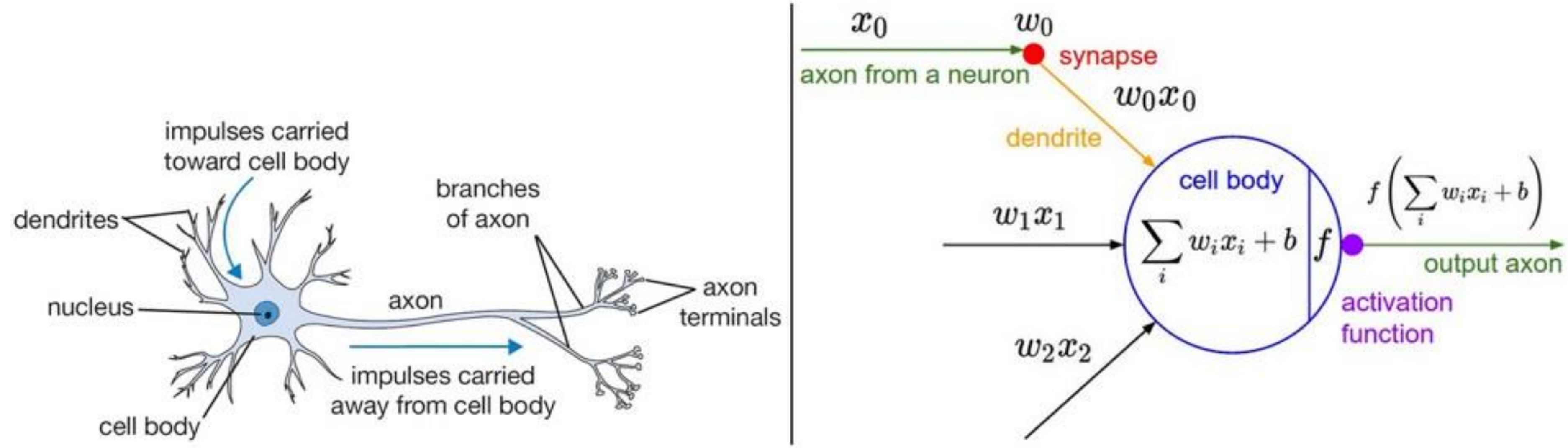
hyperplane

- **Another option** is to write the hyperplane representation with **homogenous vectors,** this will result with the (more compact) equation below:

$$[w_1 \ \cdots w_n \ \ b]\begin{bmatrix} x_1 \\ \vdots \\ x_n \\ 1 \end{bmatrix} = w^T x = 0$$

- Points x above the hyperplane (in the direction of the normal) will result in $w^T x > 0$, and points x below the hyperplane will result in $w^T x < 0$.

Side note: Inspiration from Biology

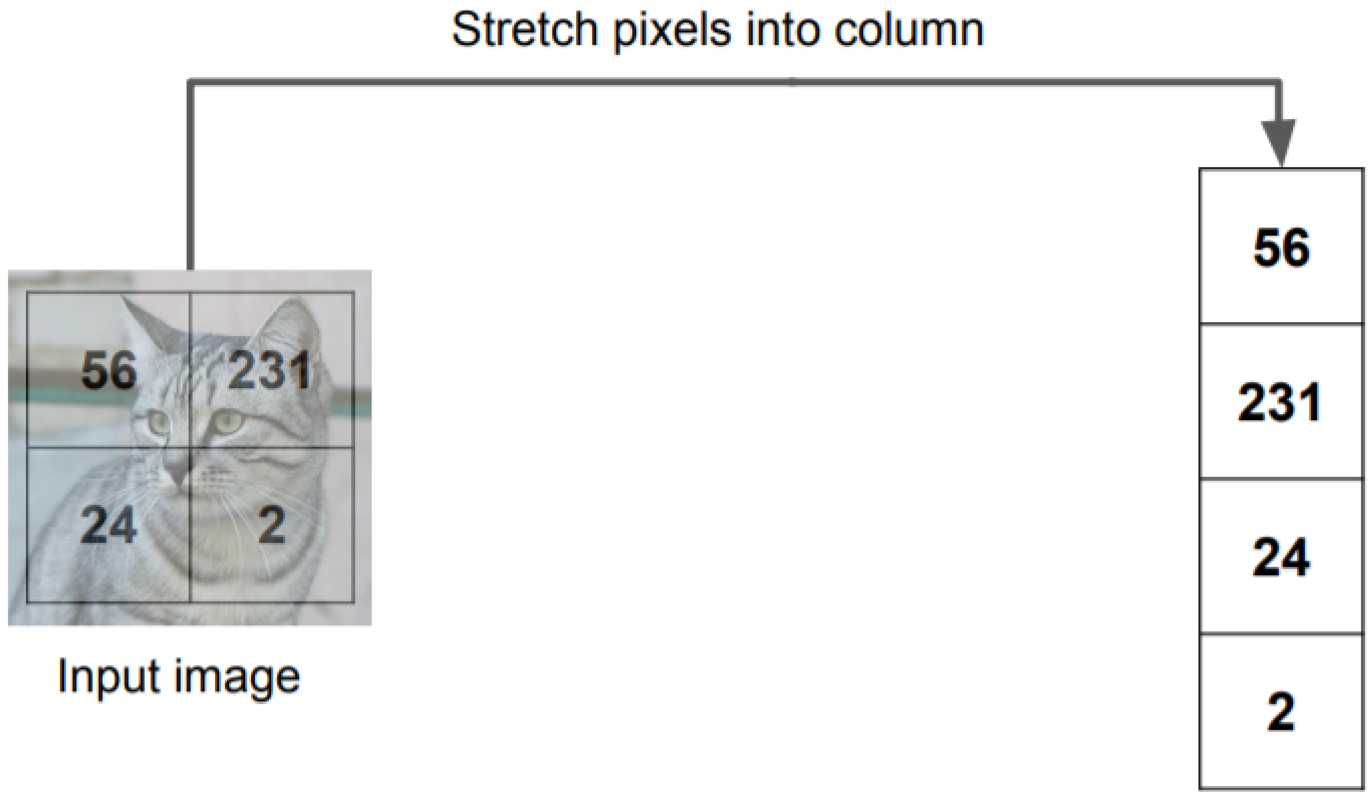


A cartoon drawing of a biological neuron (left) and its mathematical model (right).

- In this example the activation function is $f(x) = \begin{cases} -1, & x < 0 \\ 1, & x \geq 0 \end{cases}$
- Neural nets/perceptrons are **loosely** inspired by biology.
- But they certainly are **not** a model of how the brain works, or even how neurons work.

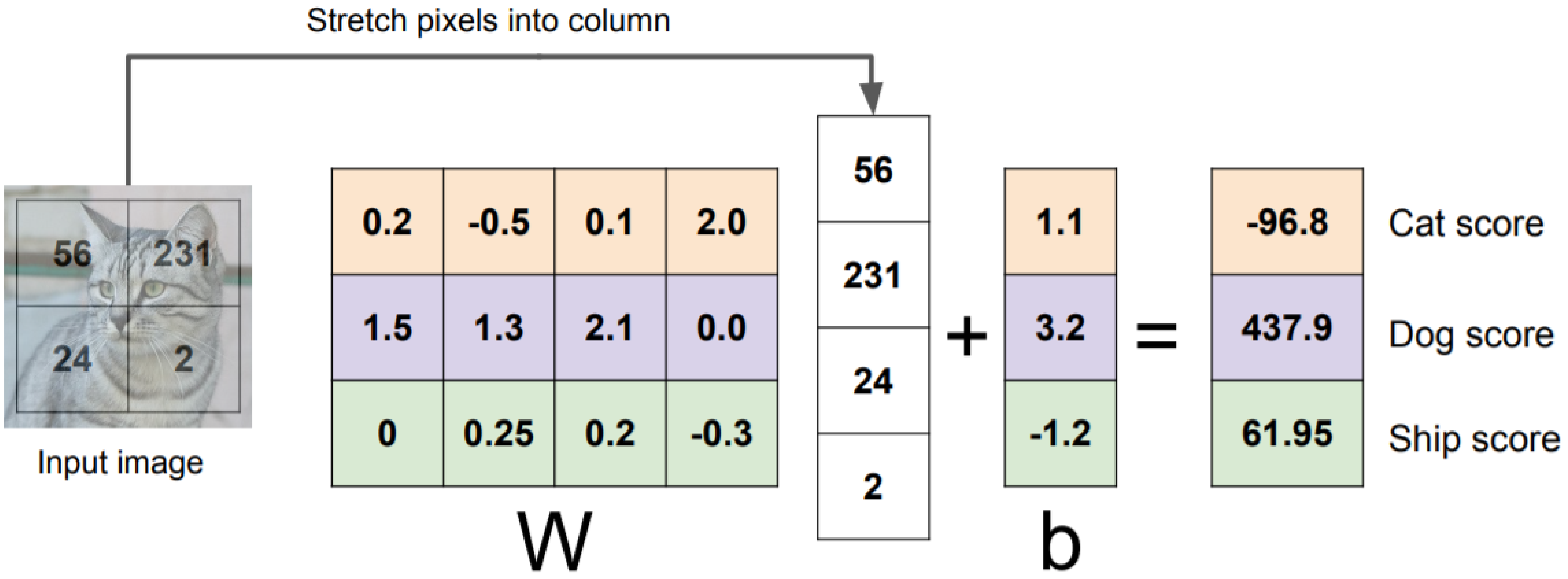
Hyperplanes and image classification

- Let’s say an image is a vector in 4D.



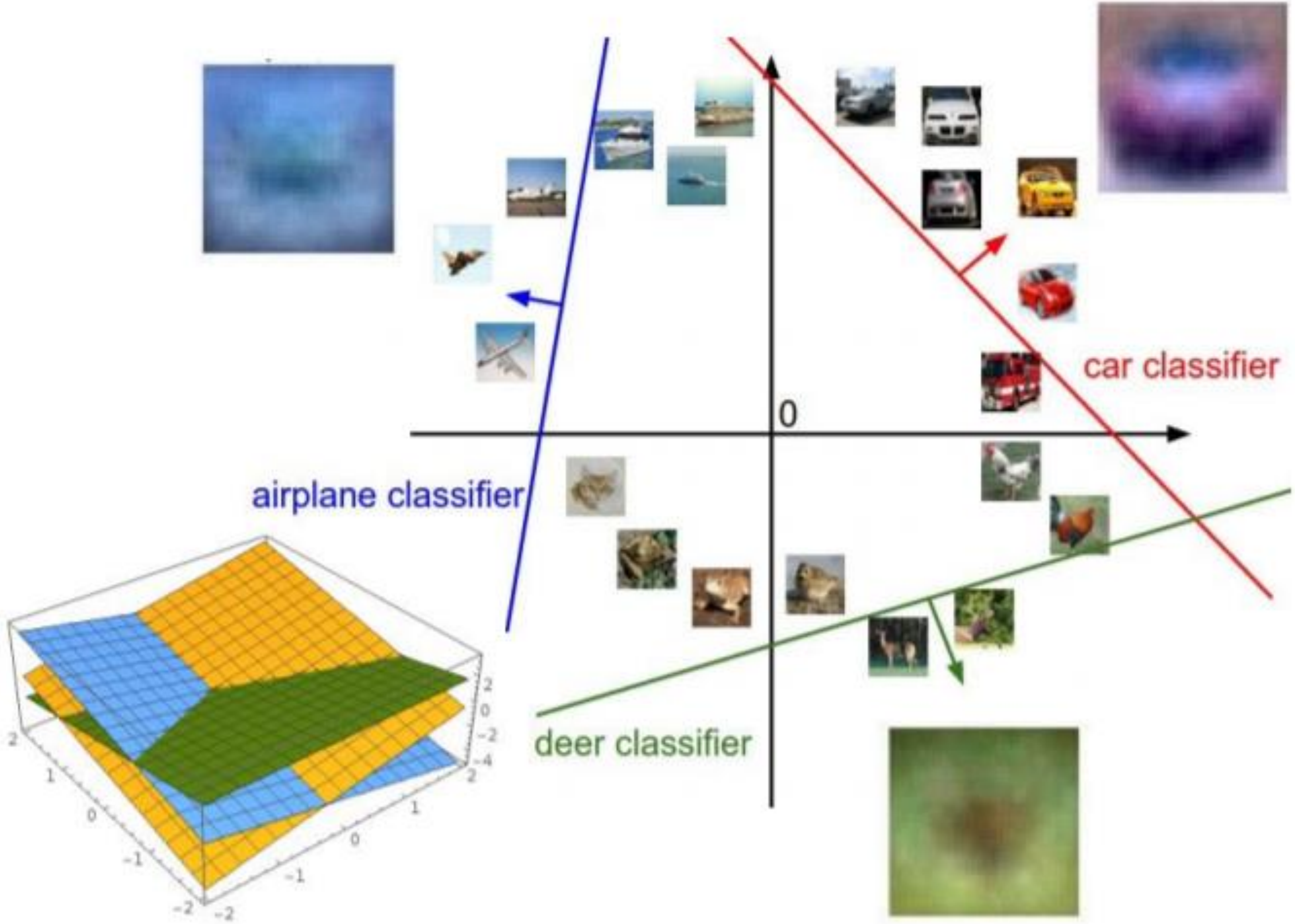
Hyperplanes and image classification

- We want to find a hyperplane in 4D space that puts all cats' vectors in one side of it, and all other images in the other side.
  - Let's assume there are 2 more classes. In total: cats, dogs and ships.
  - Find 3 separating planes, one for each class.



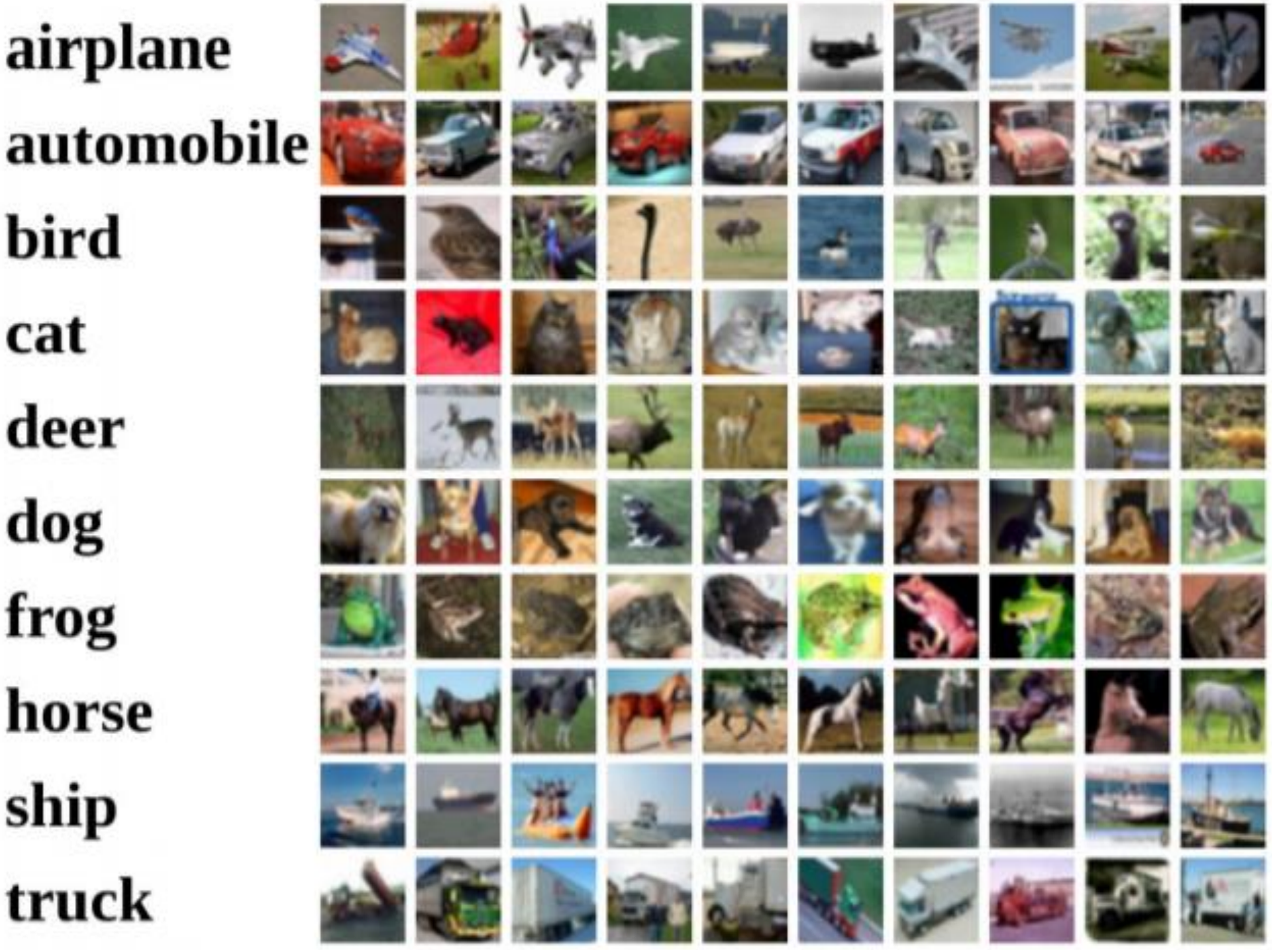
Hyperplanes and image classification

- Another example.



CIFAR10 dataset

- 32X32X3=3072 DOFs in this problem, and images vary a lot. This is not possible to linearly separate.



50,000 training images
each image is **32x32x3**

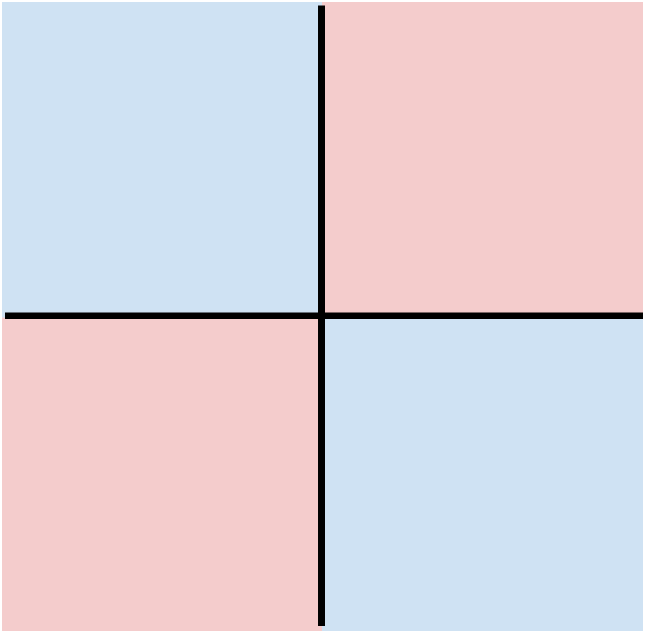
10,000 test images.

More none linear separable examples

Hard cases for a linear classifier

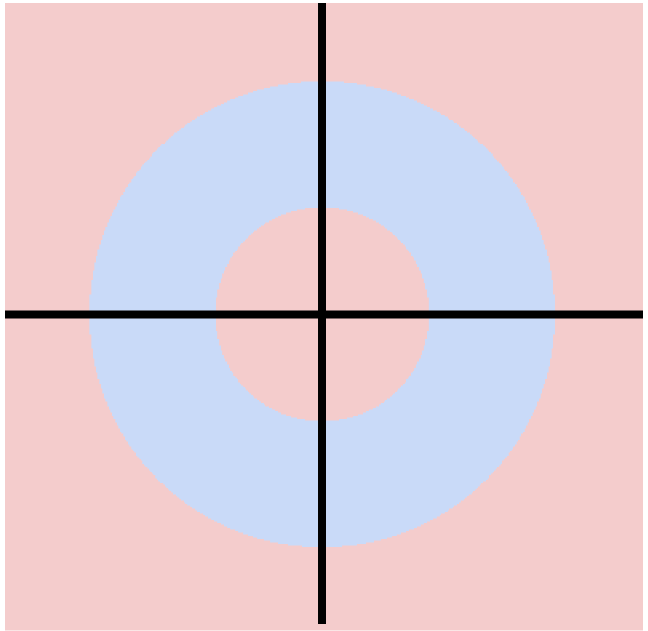
Class 1:
First and third quadrants

Class 2:
Second and fourth quadrants



Class 1:
1 <= L2 norm <= 2

Class 2:
Everything else



Class 1:
Three modes

Class 2:
Everything else