Ch6. Neural Network
Not so Deep Learning

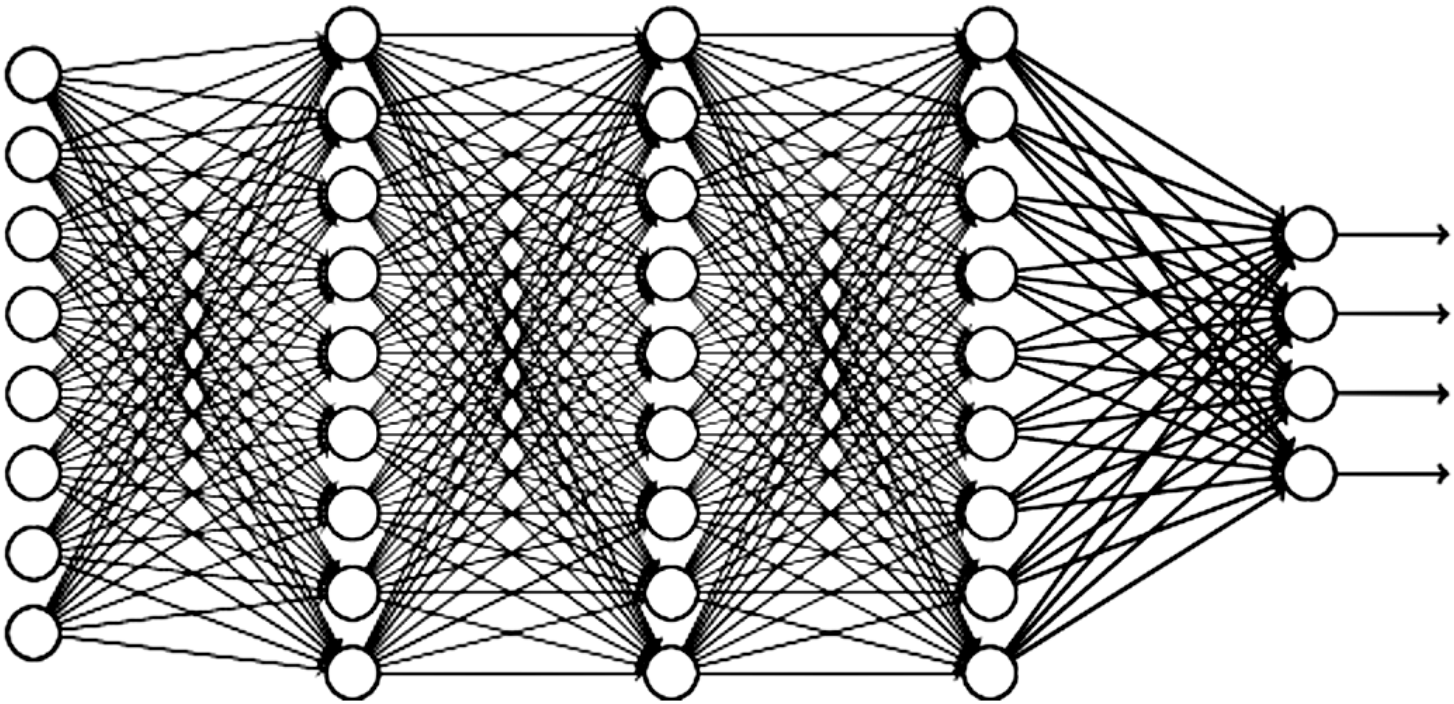
ST4240, 2016/2017

Version 0.1

Alexandre Thiéry

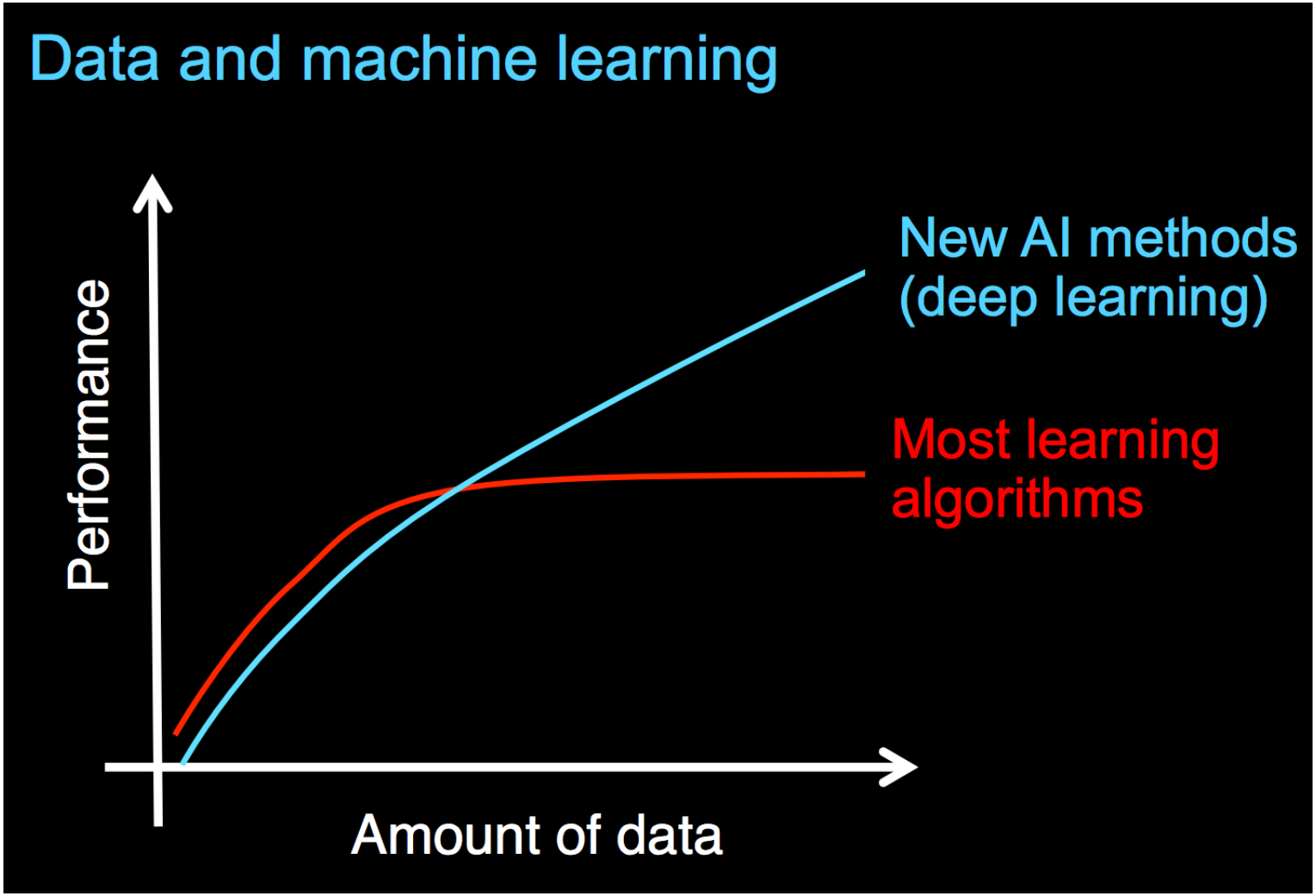
Department of Statistics and Applied Probability

Basic Feed-forward network



Deep Learning: where to get started?

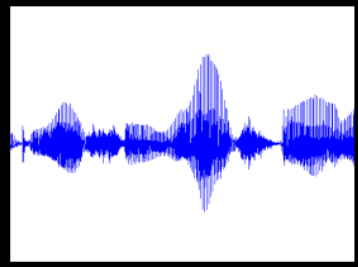
- Learn Python!
- The Python library Keras is superb and simple
- Gentle intro at: http://neuralnetworksanddeeplearning.com/
- http://course.fast.ai/ is great and very practical
- Many blogs and ressources online. Try and experiment!


Data and machine learning
Performance
Amount of data
New AI methods
(deep learning)
Most learning
algorithms

Images

Label image

Audio



Speech recognition

Text

Web search

Deep Learning is good at dealing with images

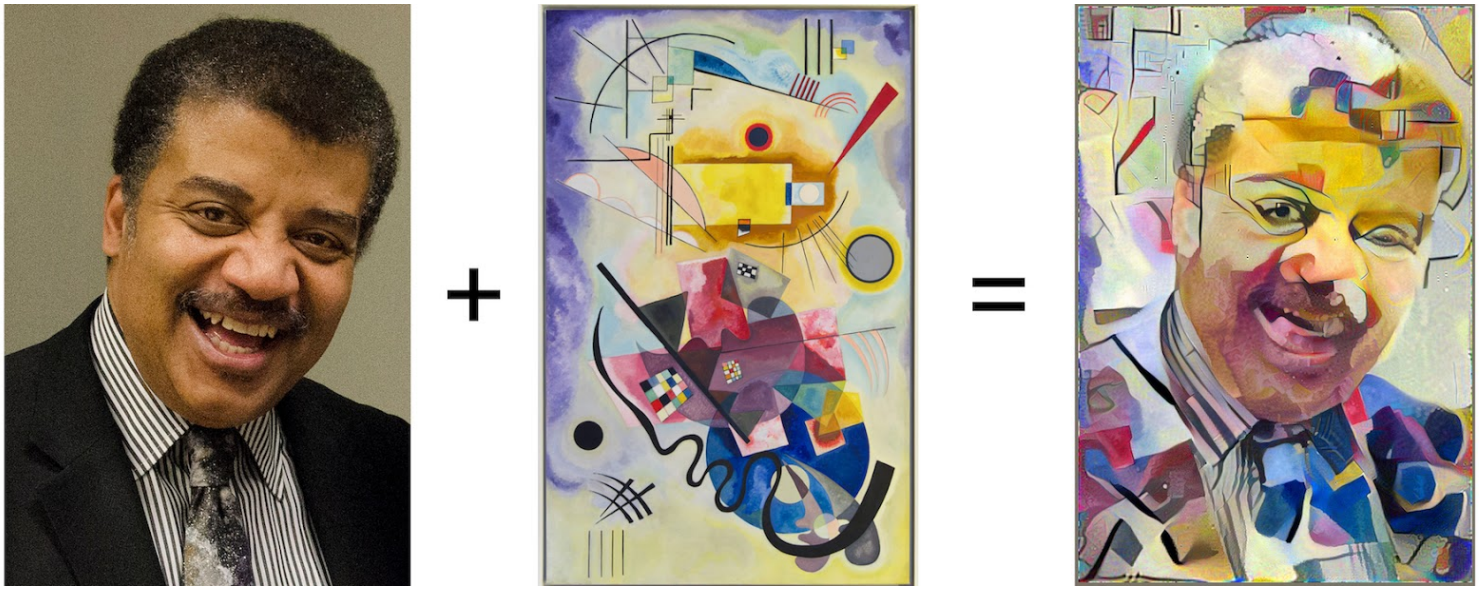
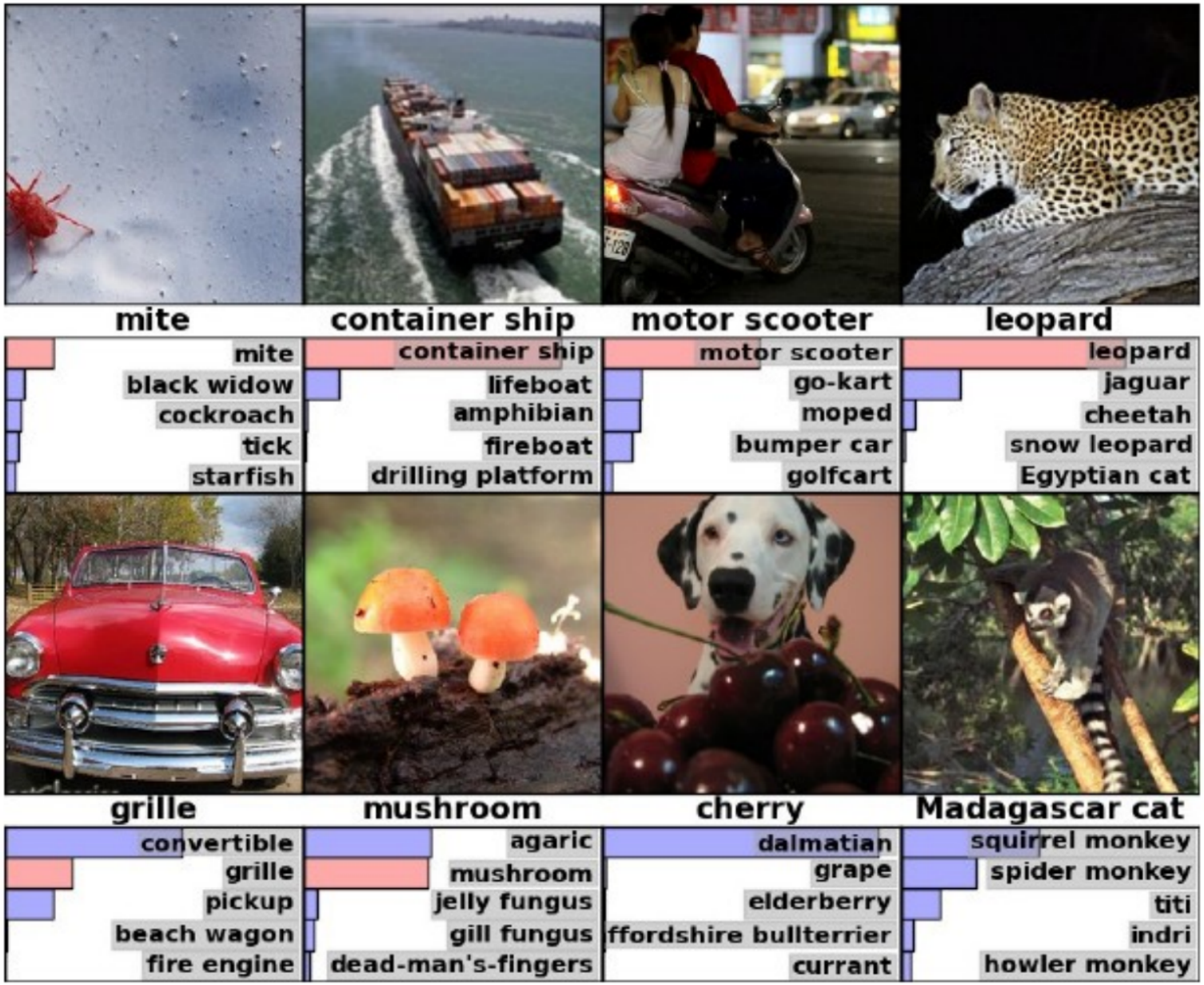
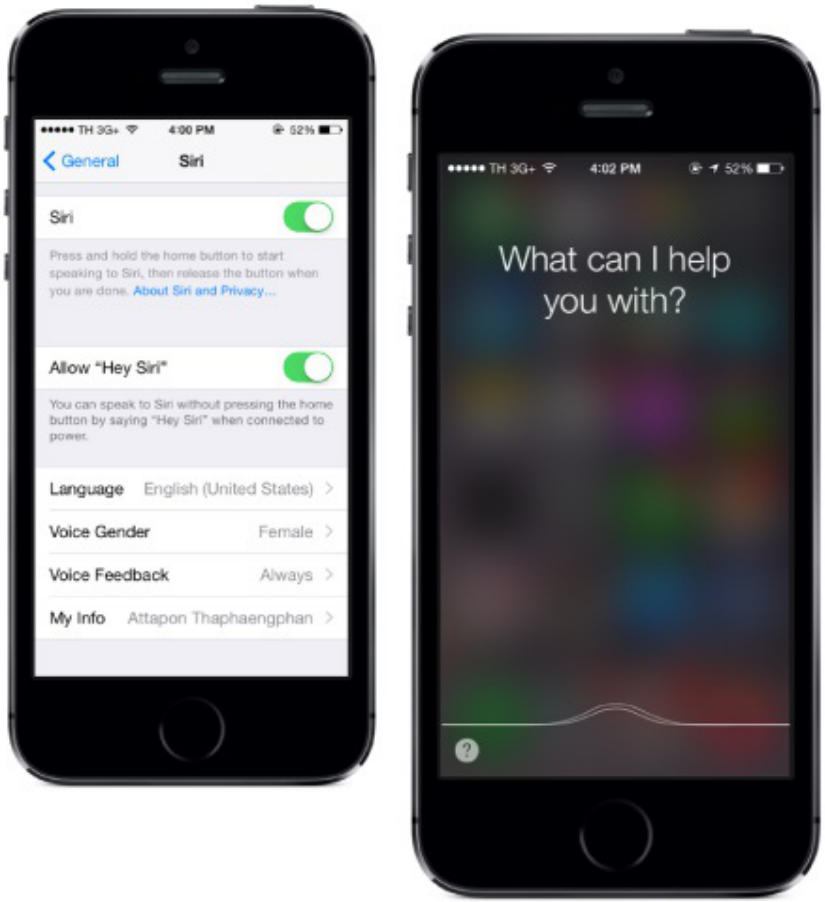


Image classification

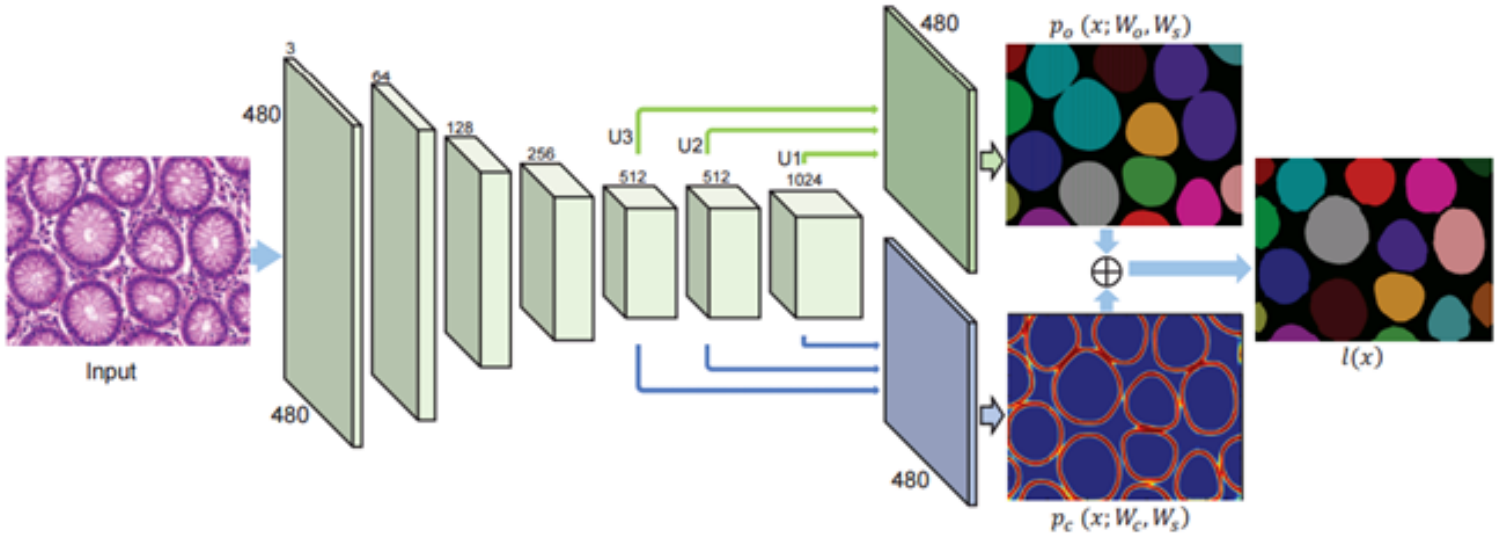


Speech recognition and understanding

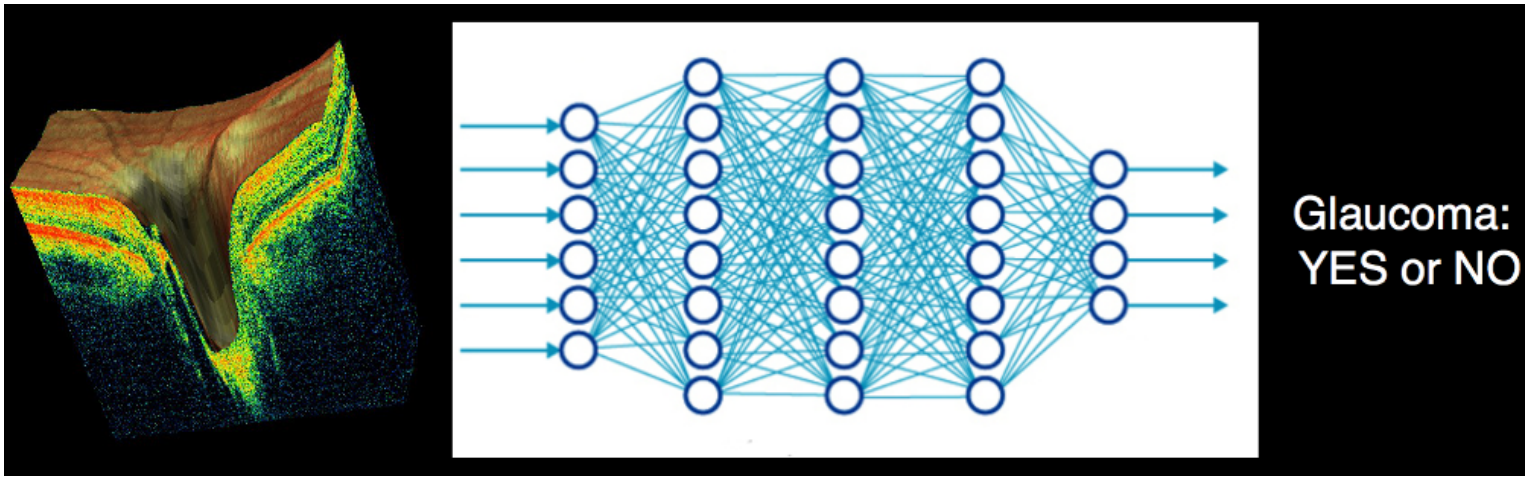


Hey Siri!

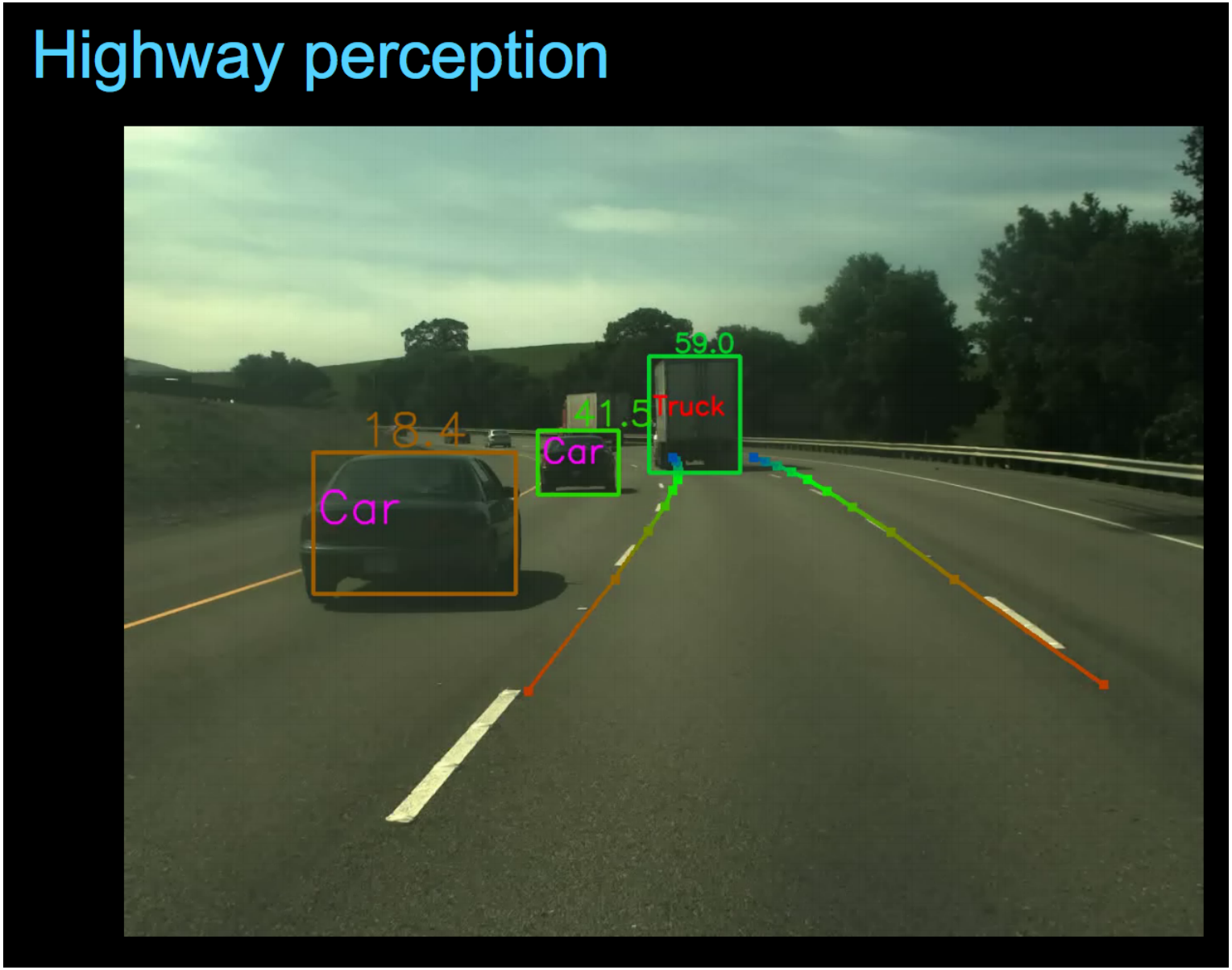
Cancer Detection



Automated Glaucoma Detection...



Self Driving cars...

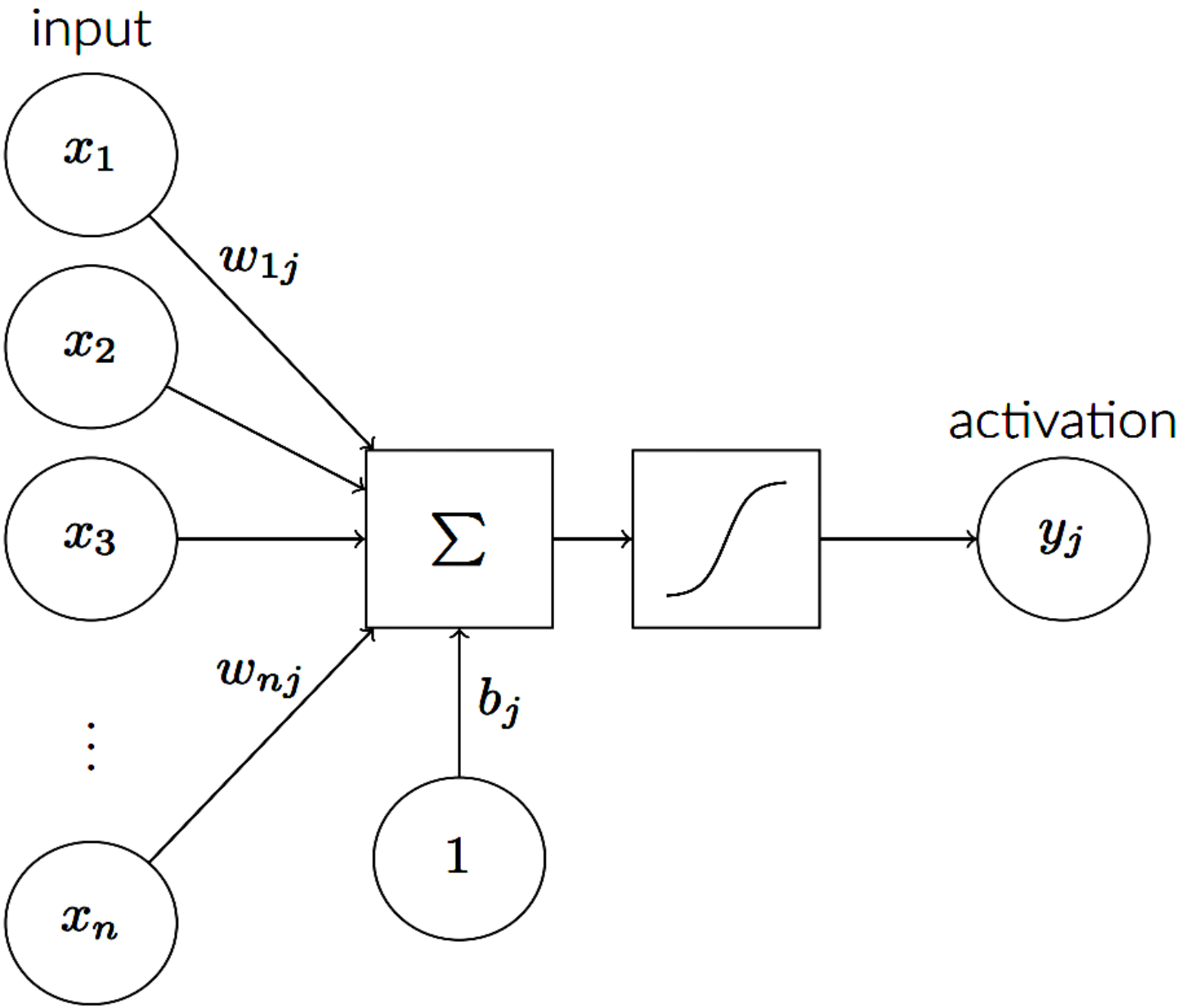


Deep Learning: rules them all?

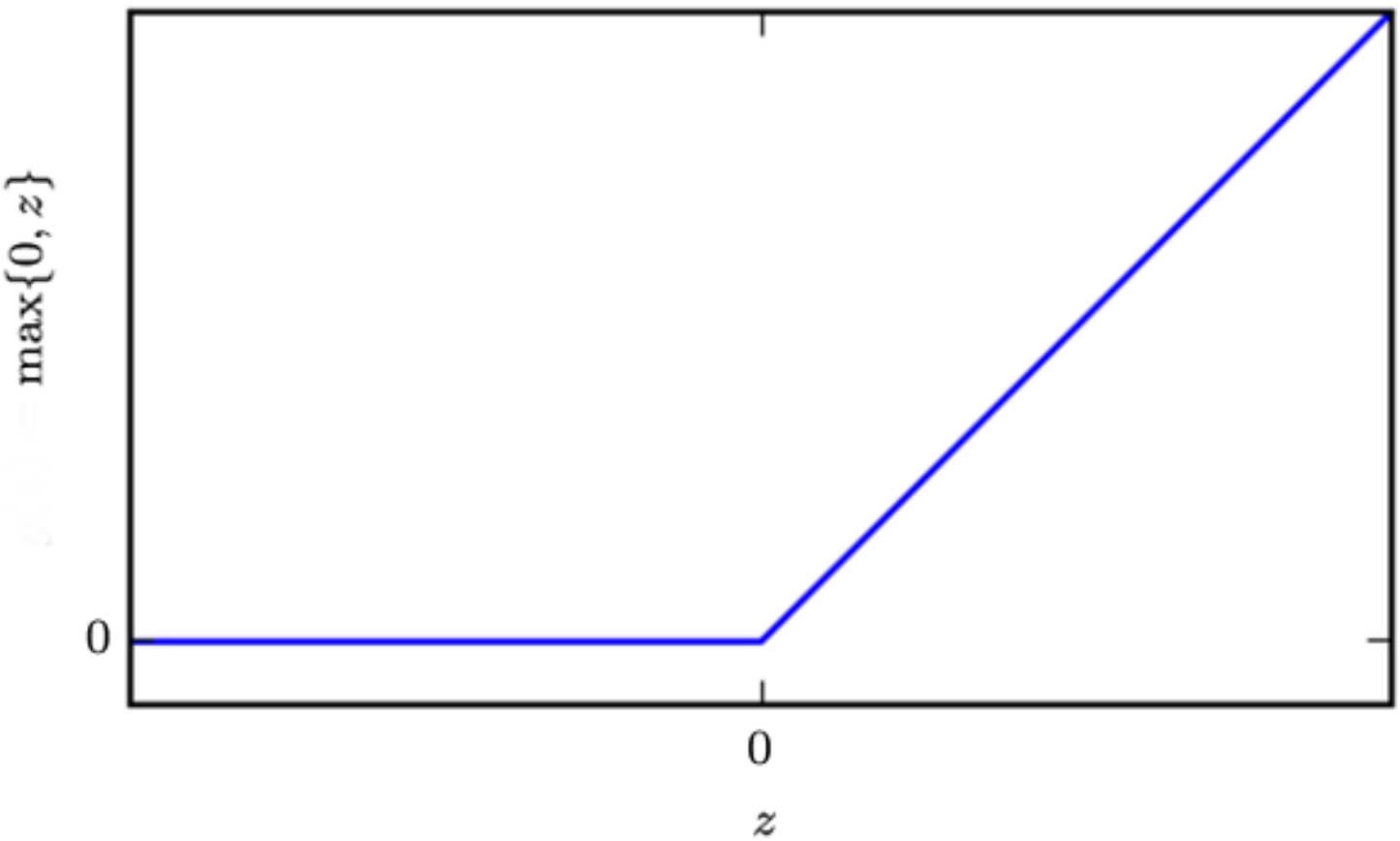
- Deep-Learning often works very well to extract information from complex modalities (images, sound text, etc...)
- It sometimes fails fails spectacularly!
- In many more "simple" settings, approaches such as Random Forest, Gradient boosting, etc.., can perform much better
- Deep Learning is not very good at giving confidence intervals
- It is a big "black-box", which can be a problem in some areas...
- Sometimes difficult and long to implement.

Outline

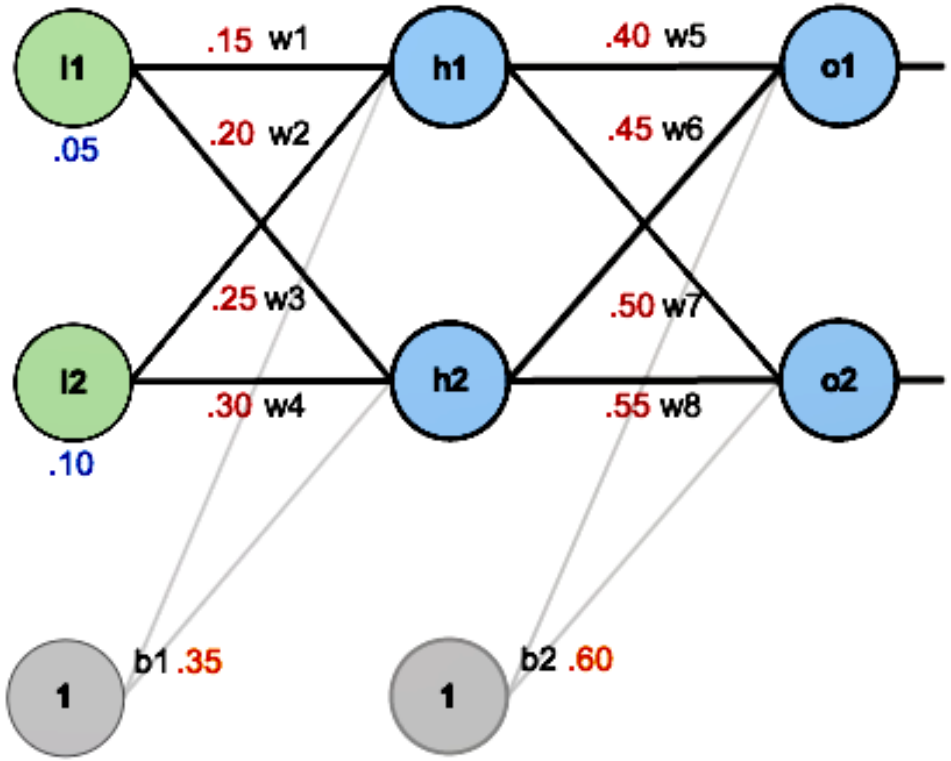
Neuron: Linear combination + Non-linearity



RELU (rectified linear unit) non-linearity

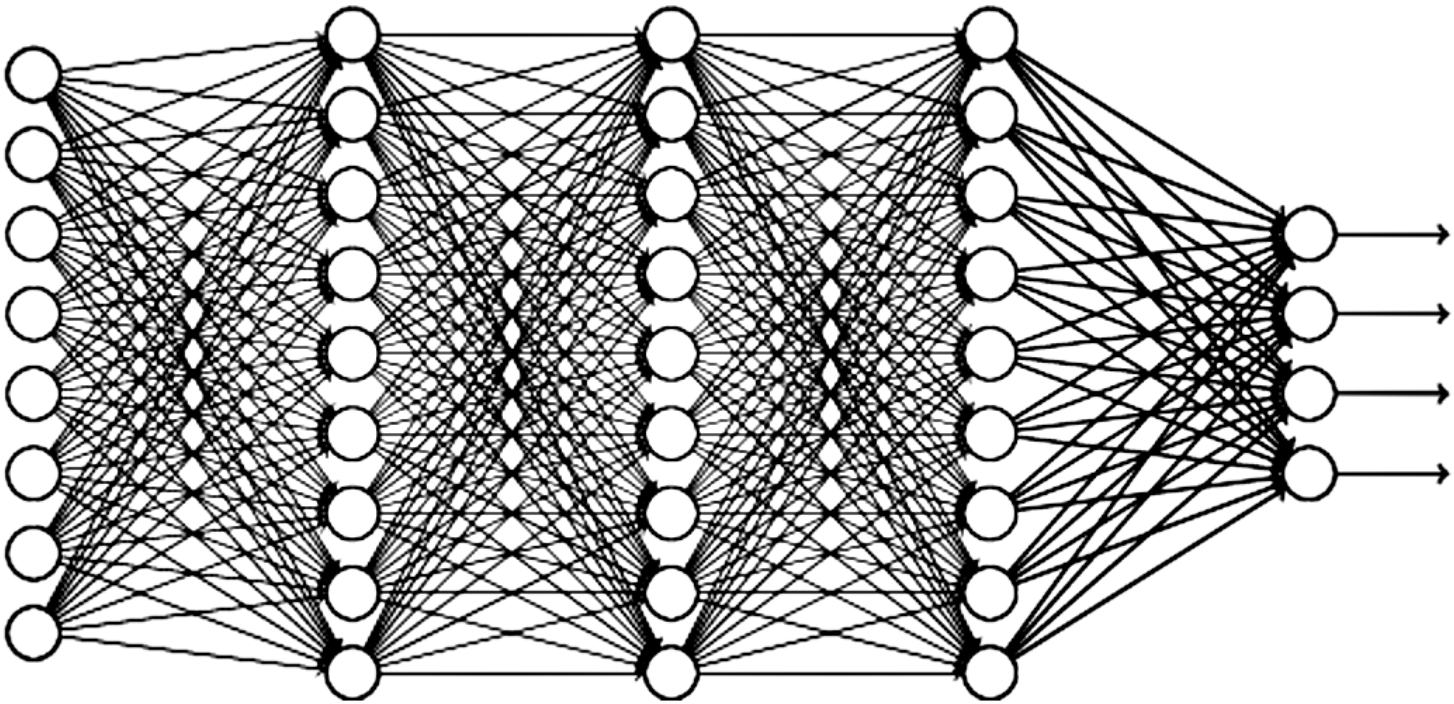


Feedforward example



- **[Exercise]**: Assuming a RELU non-linearity, propagate forward in the neural network to obtain the two outputs.

Feedforward example



- **[Exercise]**: How many unknown parameters?

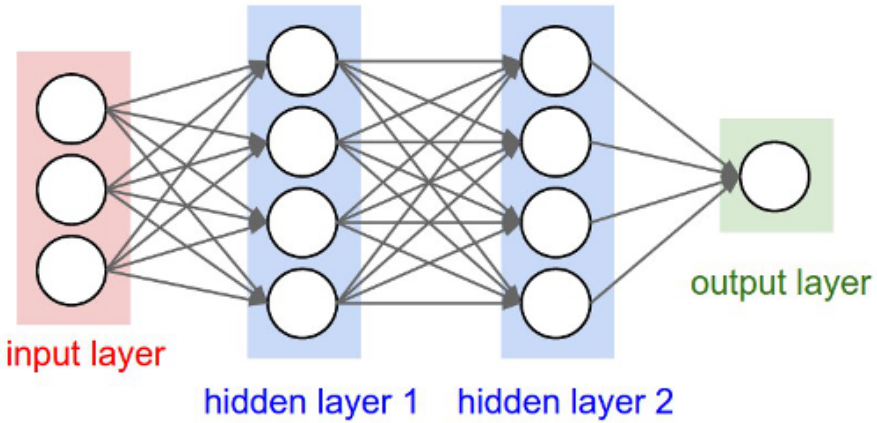
Outline

Usual setting

- Training data: $\{x_i, y_i\}$ with $x_i \in \mathbb{R}^p$ and $y_i \in \mathbb{R}$
- **Goal**: find a function $F : \mathbb{R}^p \to \mathbb{R}$ such that

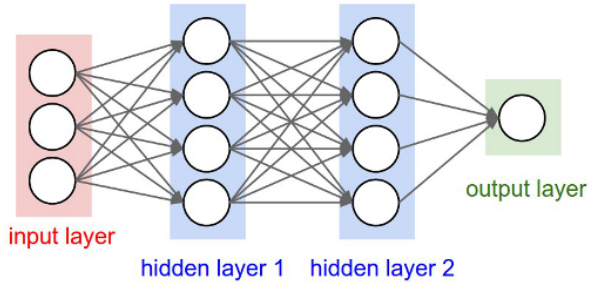
$$y_i \approx F(x_i)$$

- Linear regression: $F(x) = \langle x, \beta \rangle$ (very restrictive)
- Idea: represent $F(x)$ by a neural network



Minimization of a sum of losses

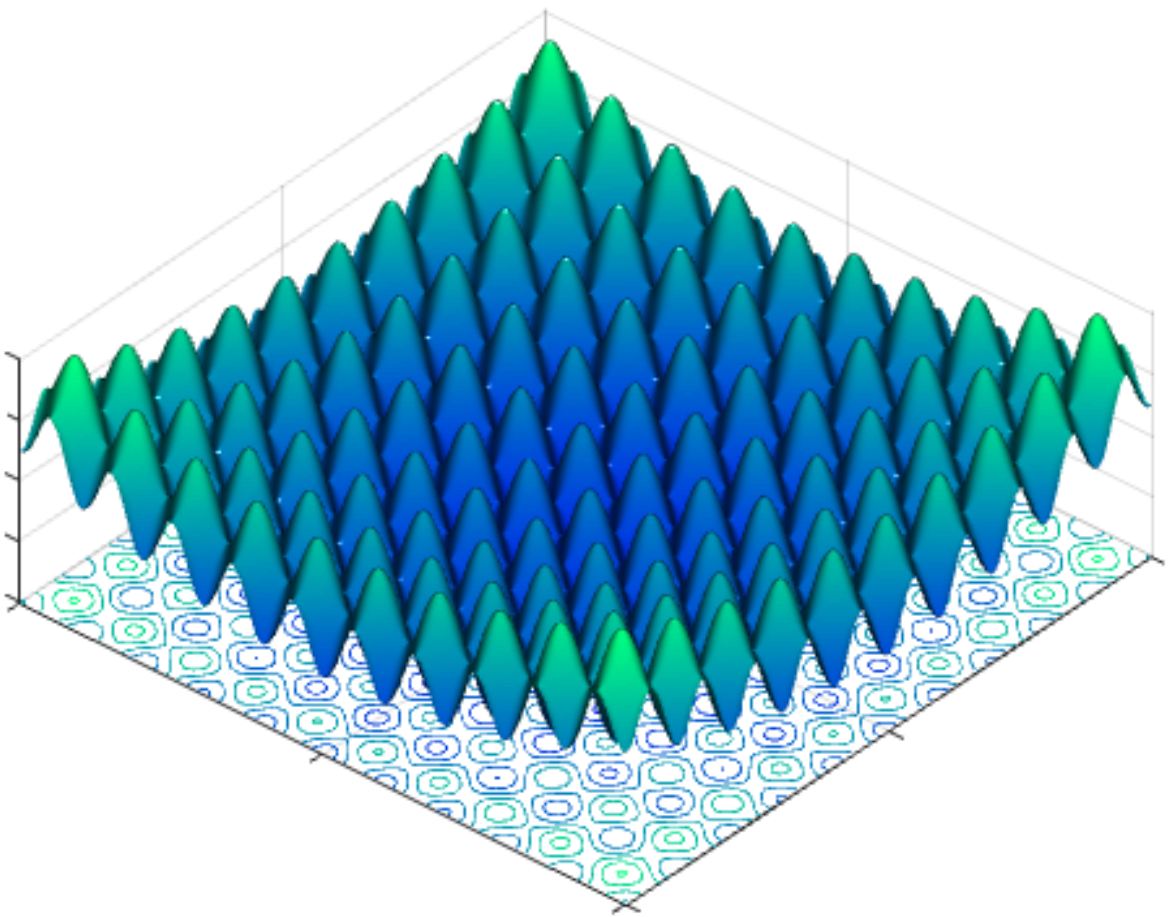
- Represent $F(x)$ by a neural network



- Let us write ω for the set of all unknown weights: $F(x) = F_\omega(x)$
- Minimizing the sum of square:

$$\omega_\star = \textbf{argmin} \left\{ \omega \mapsto \frac{1}{N} \sum_{i=1}^{N} (y_i - F_\omega(x_i))^2 \right\}$$

Gradient descent...



Gradient descent...

- In large scale applications, **stochastic gradient descent** is crucial
- This minimization involves a lot of heavy matrix computations.
- Standard Deep-Learning are very **high-dimensional problems**

$$\mathbf{dim}(\omega) \approx 10^6$$

- Usually a good idea to use Graphical Processing Units (GPUs)

Outline

Usual setting

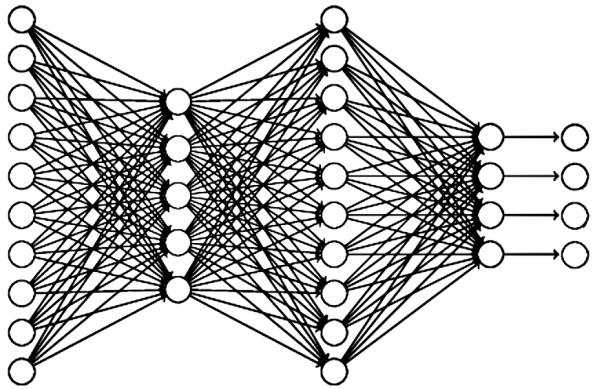
- Training data: $\{x_i, y_i\}$ with $x_i \in \mathbb{R}^p$ and $y_i \in \{1, 2, \ldots, C\}$
- **Probabilistic Prediction**: find a function $F : \mathbb{R}^p \to P_C$ where

$$P_C = \{(p_1, \ldots, p_C) \ : \ p_1 + \ldots + p_C = 1\}$$

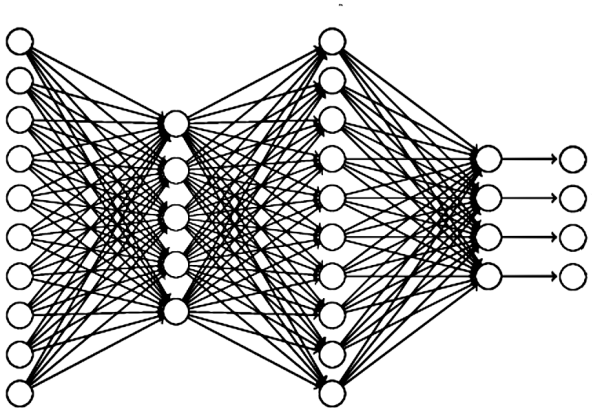
and such that

$$\mathbb{P}(y_i = k \,|\, x_i) \approx F_k(x_i)$$

- Here we have $F(x) = (F_1(x), F_2(x), \ldots, F_C(x))$
- Idea: represent $F(x)$ by a neural network



Probabilistic Output

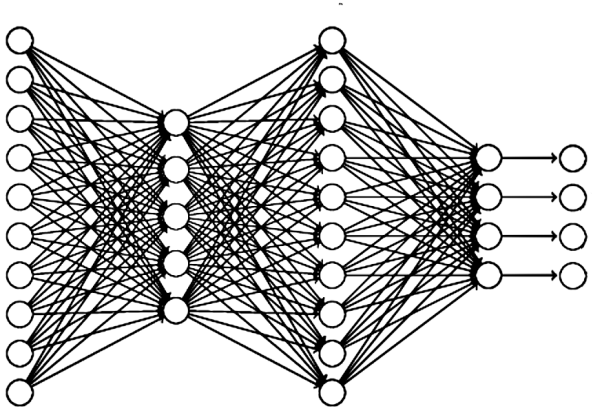


- **[Exercise]**: how to make sure that

$$F_1(x) + \ldots + F_C(x) = 1$$

and $F_k(x) > 0$ for all $1 \leq k \leq C$?

Probabilistic Output: cross entropy minimization



- **[Exercise]**: What is the loss function to minimize?

Overfitting

- In many applications, the number of parameters $\mathbf{dim}(\omega)$ is very large! Often larger than the numbr of training examples!
- **[Exercise]**: How can one reduce/avoid overfitting?
- **[Exercise]**: How can one choose the number of layers and the architecture of the network?