Deep Learning

François Rousseau

Lecture #3

This course focuses on "**how to build and understand**", not just "how to use".

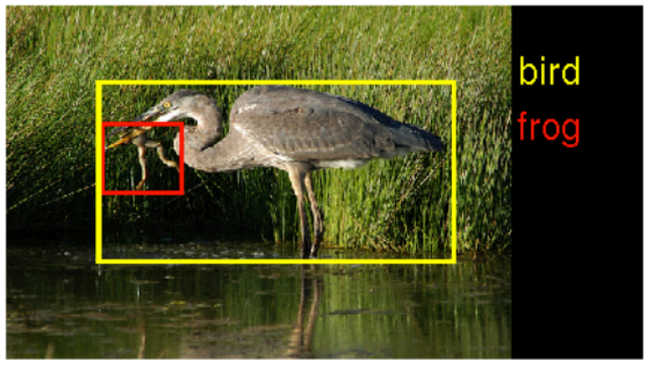
After this teaching unit, you'll know what is deep learning,
when it's applicable and what its limitations are.

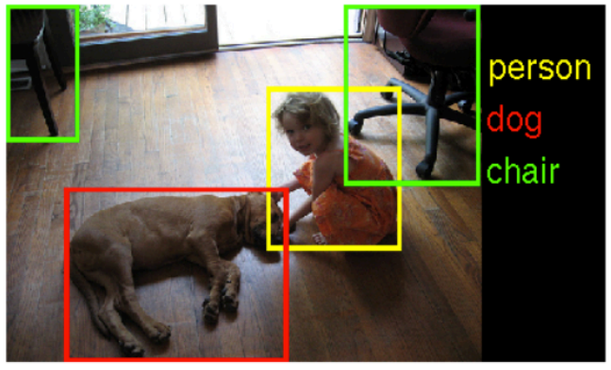
TIPS (for getting through the course)

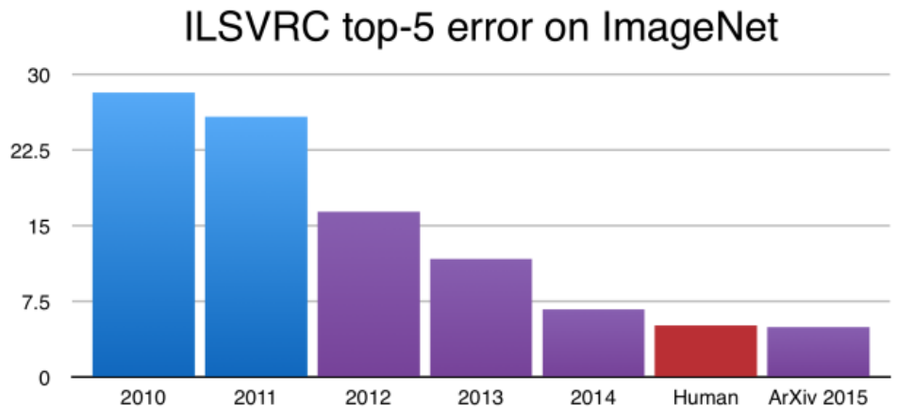
- Take handwritten notes. This will drastically increase your ability to retain the information.
- Write down the equations. If you don't, it will just look like gibberish.
- Ask lots of questions. The more the better!
- Write code yourself, don't just sit there and look at code of others.

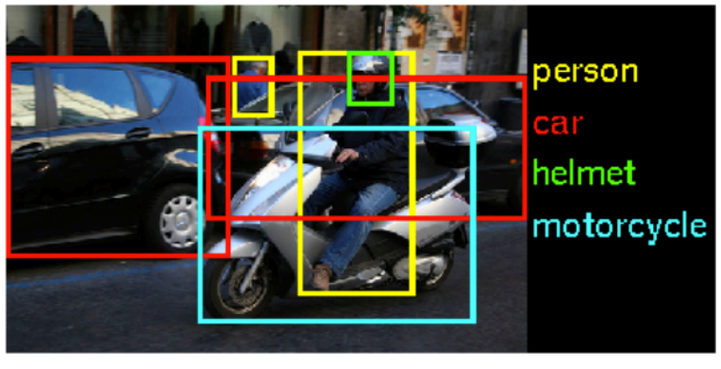
This course includes lectures, lab sessions, flipped classrooms, projects.

Large Scale Visual Recognition Challenge

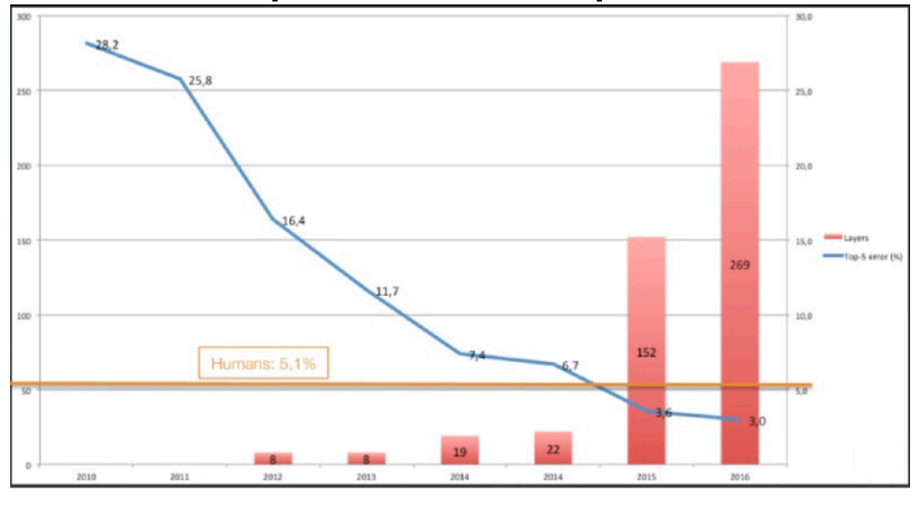








Deeper and deeper …



1.2 million of images associated with 1000 categories

http://image-net.org/challenges/LSVRC

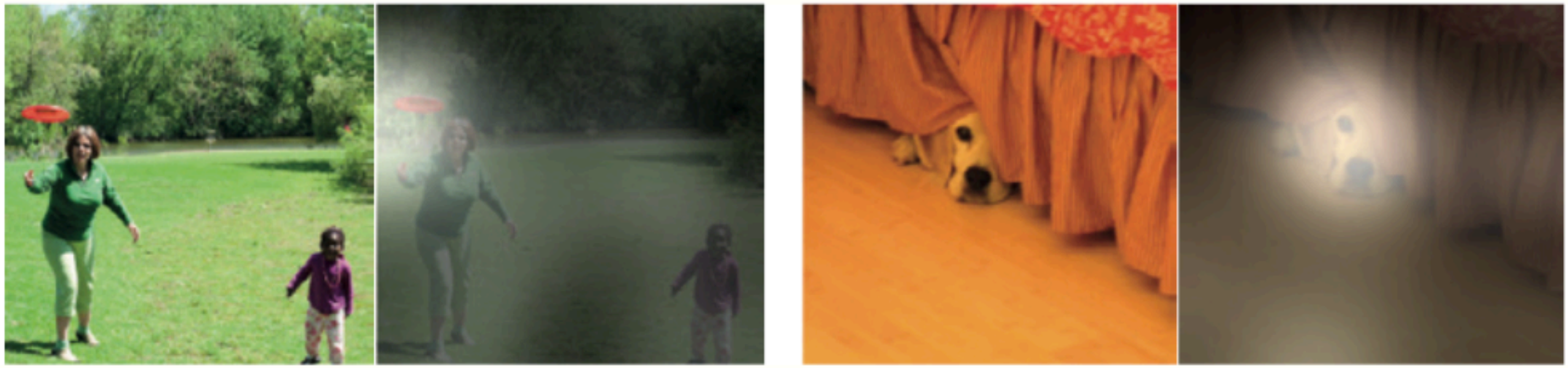
Real Time Recognition

Fig. 11. Real-time scene parsing in natural conditions. Training on SiftFlow dataset. We display one label per component in the final prediction.

Farabet et al., PAMI 2013

From Image to Text

(image captioning)



A woman is throwing a **frisbee** in a park.

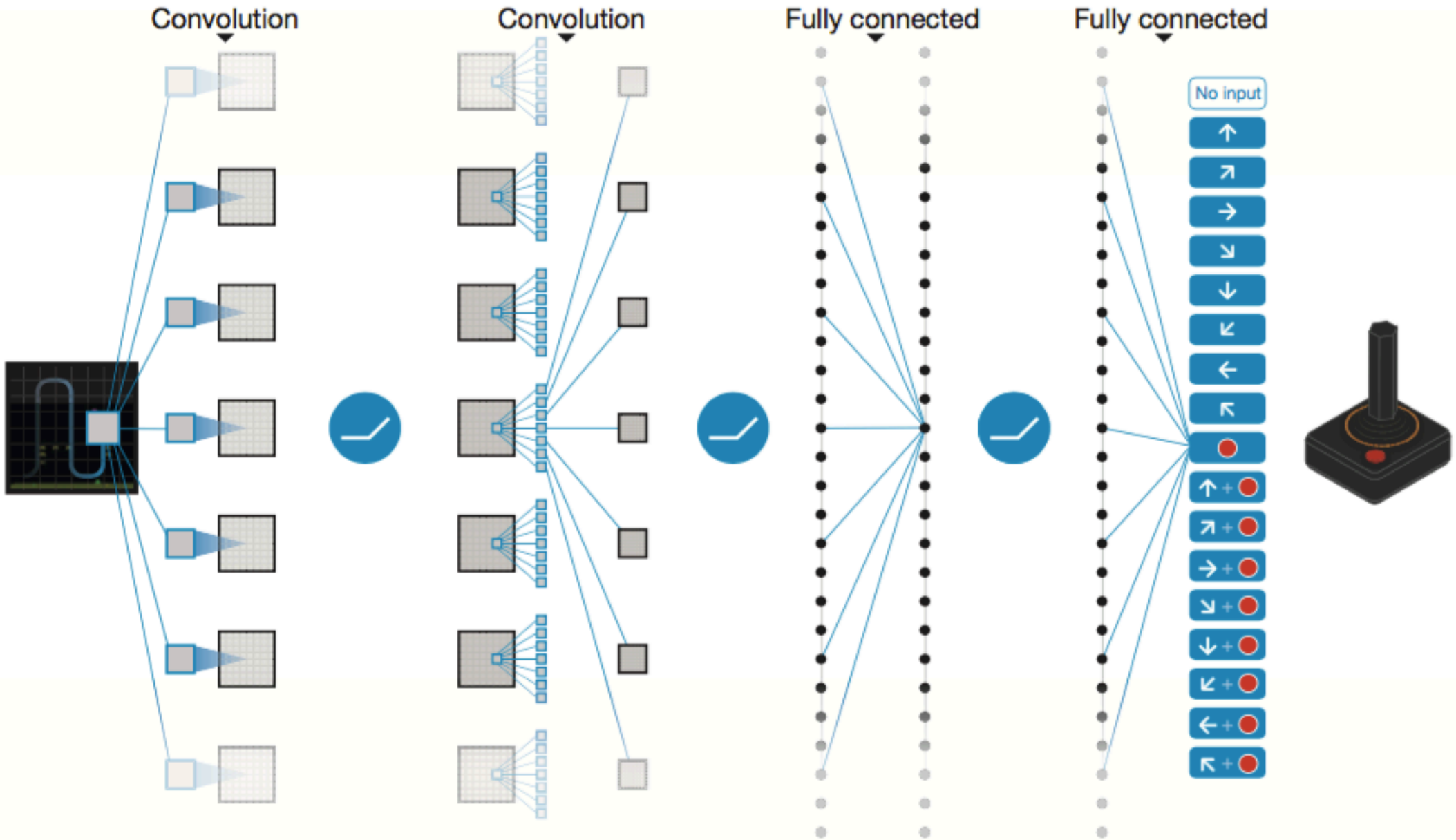
A **dog** is standing on a hardwood floor.

A little **girl** sitting on a bed with a teddy bear.

A group of **people** sitting on a boat in the water.

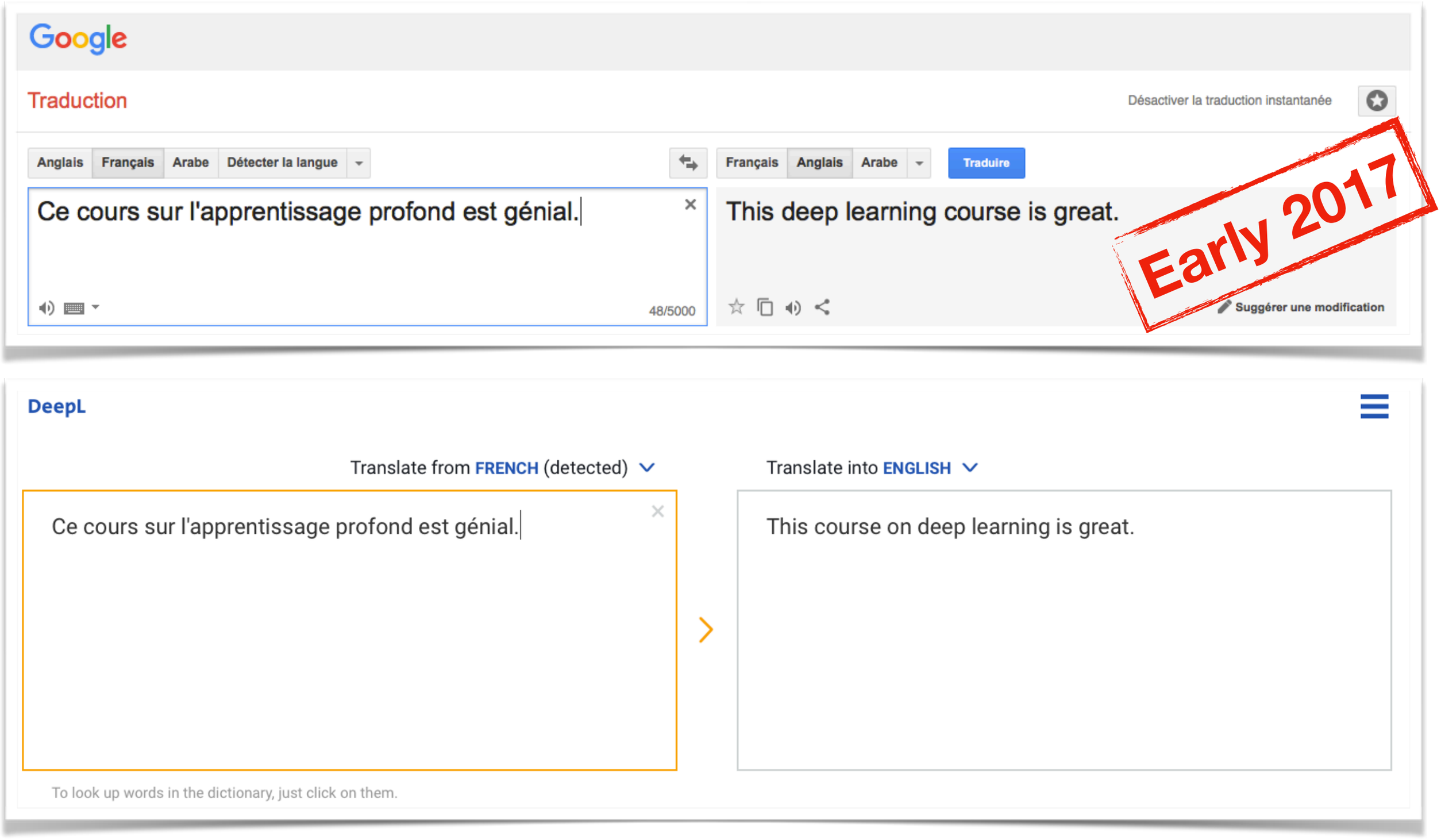
Vinyals et al., ICML 2014

Algorithm playing ATARI



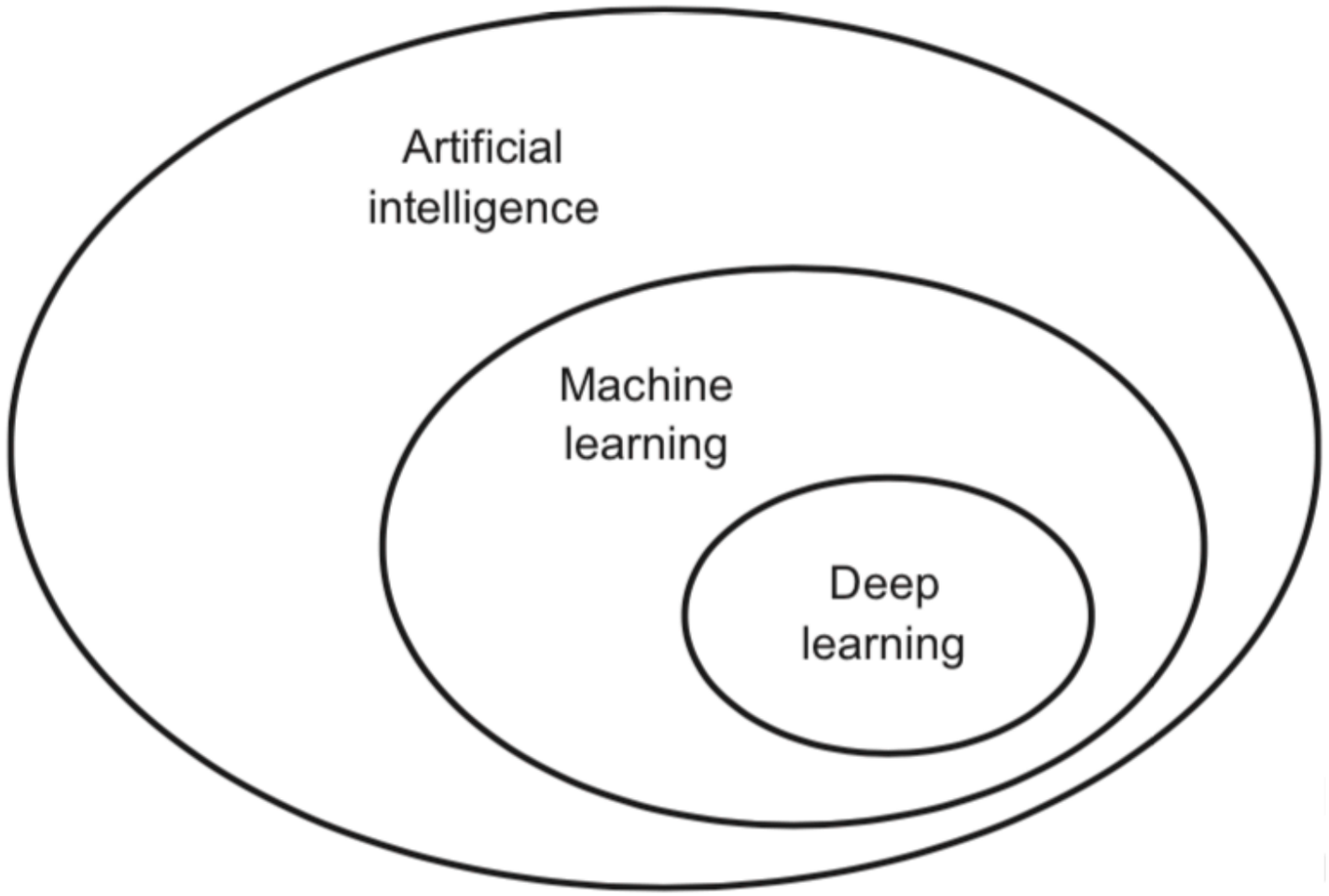
Mnih et al., Nature 2015

Automatic Translation



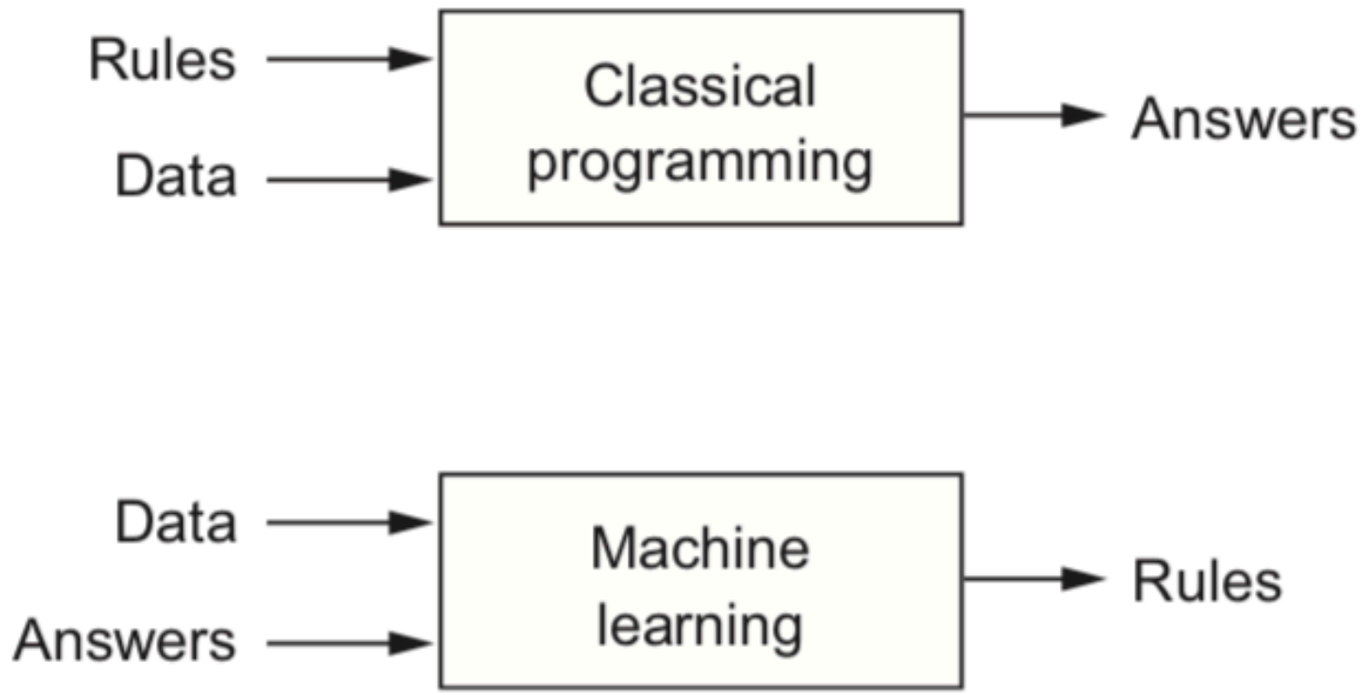
What is Deep Learning?

AI / ML / DL



[Chollet 2018]

Learning *from* data



[Chollet 2018]

Machine Learning

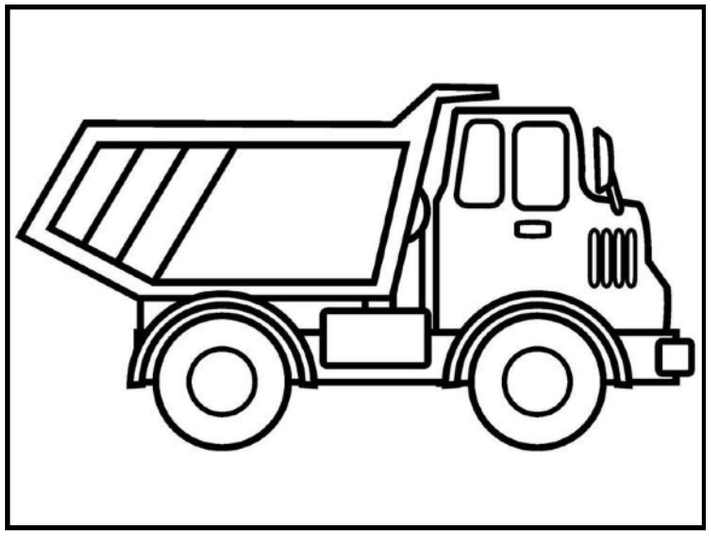
« A computer program is said to learn from experience **E** with respect to some task **T** and some performance measure **P**, if its performance on **T**, as measured by **P**, improves with experience **E**. » (Tom Mitchell, 1998)

Machine Learning Tasks:

- **supervised learning**
- **unsupervised learning**

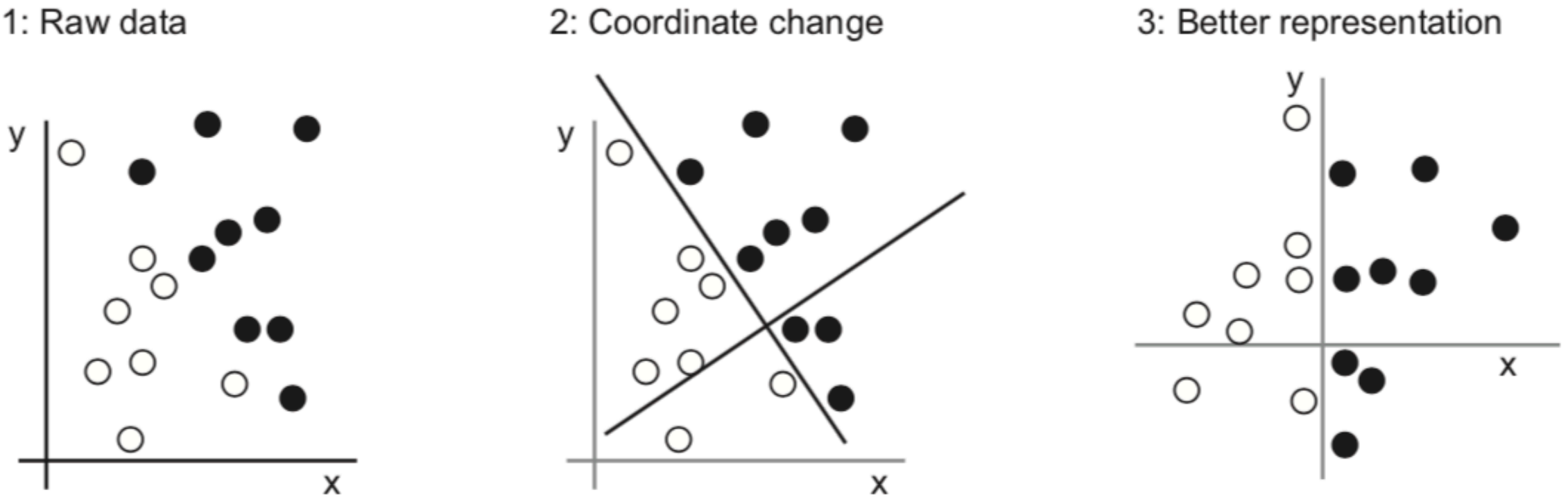
input **output**

CAR



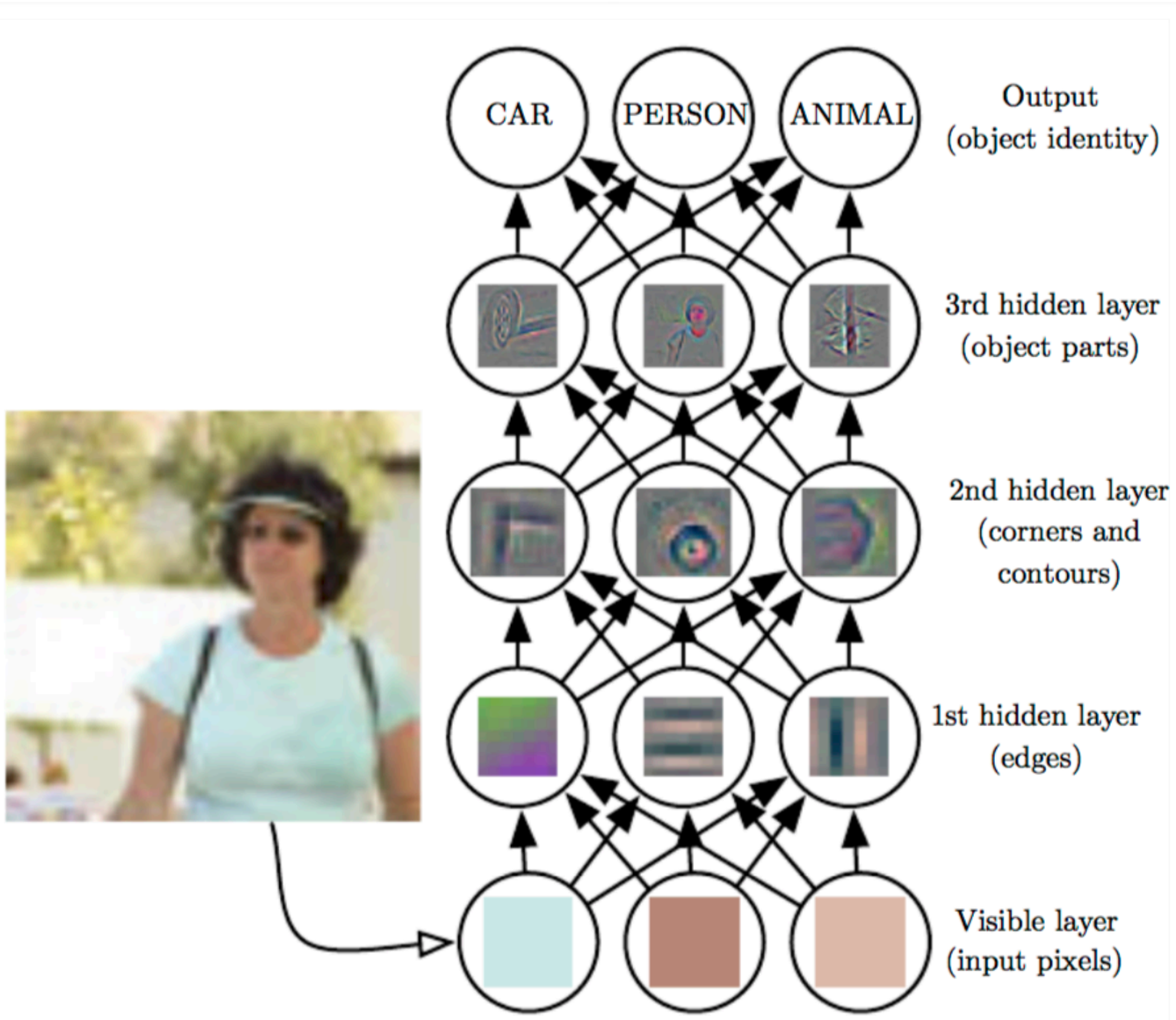
TRUCK

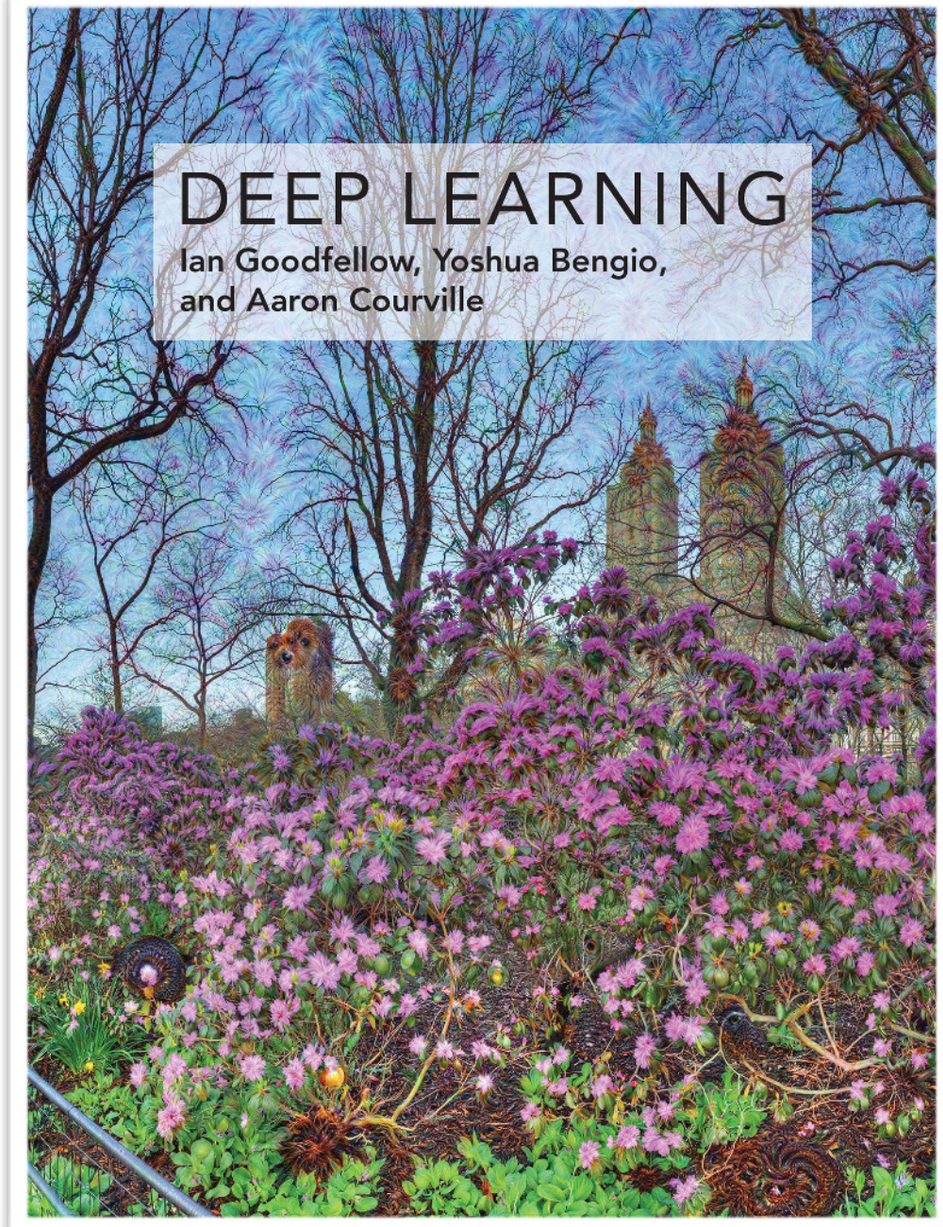
Data representation

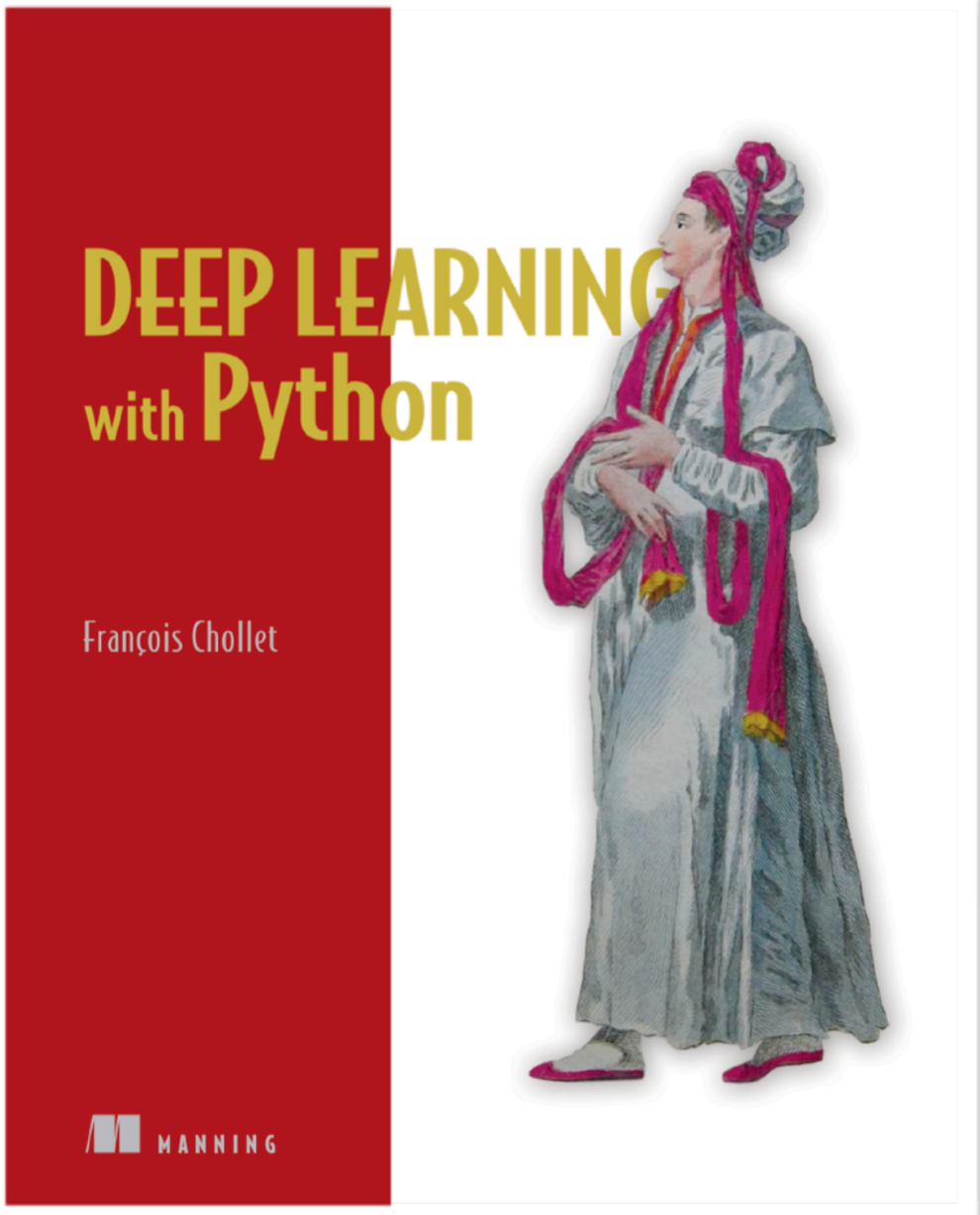


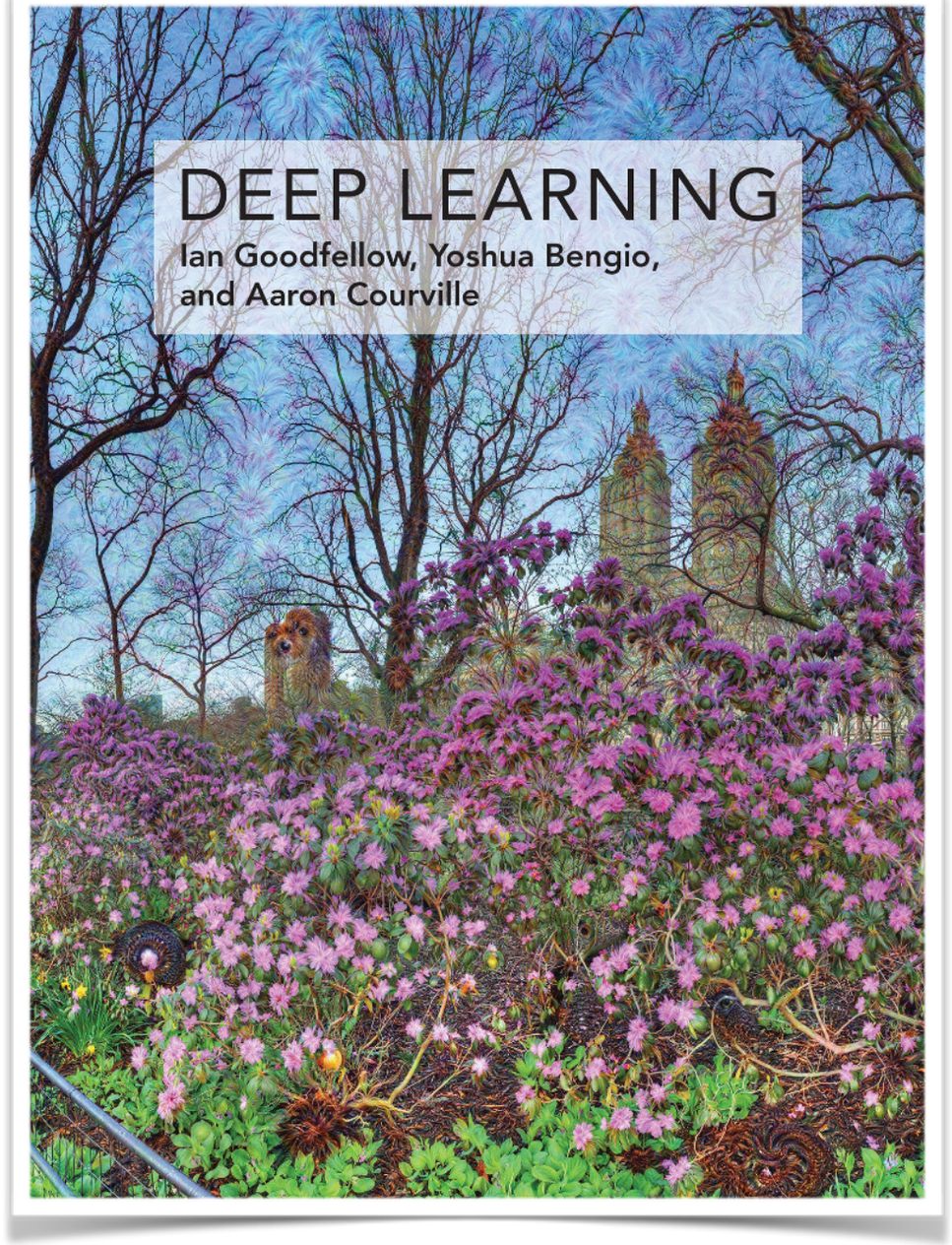
[Chollet 2018]

What is Deep Learning?

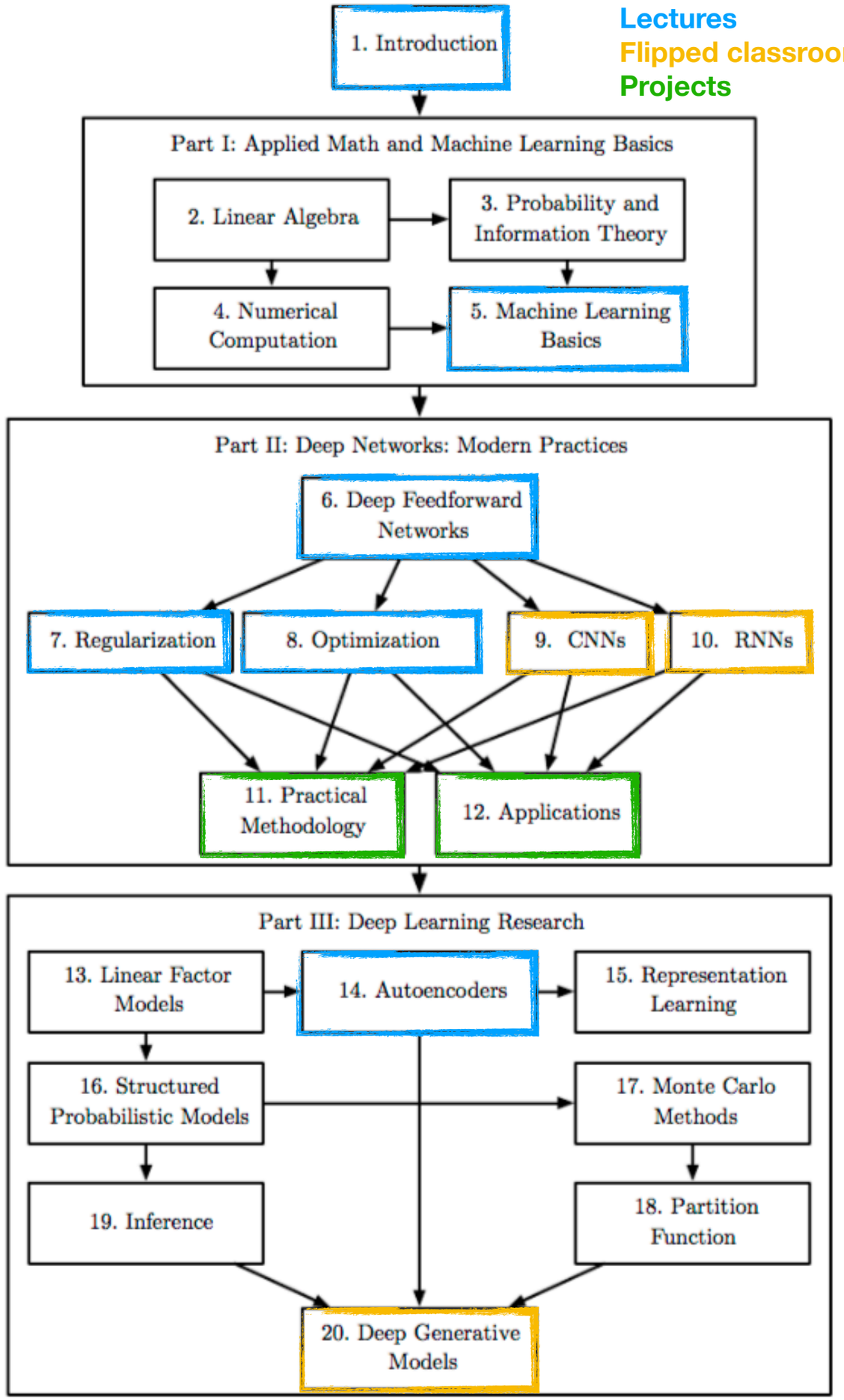



DEEP LEARNING
Ian Goodfellow, Yoshua Bengio,
and Aaron Courville


DEEP LEARNING
with Python
François Chollet
MANNING

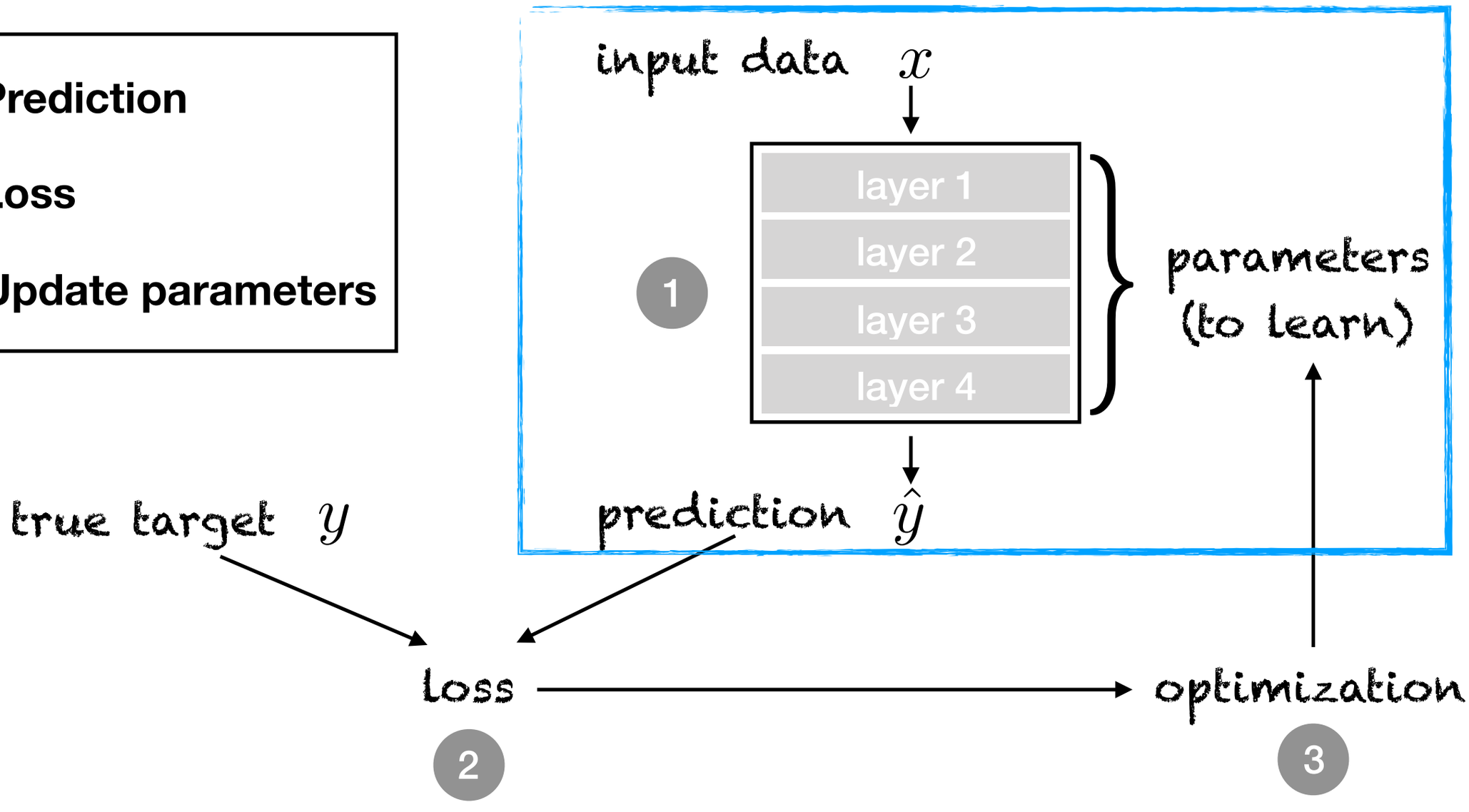


Lectures
Flipped classrooms
Projects



Overview

1. **Prediction**
2. **Loss**
3. **Update parameters**



Feedforward Networks

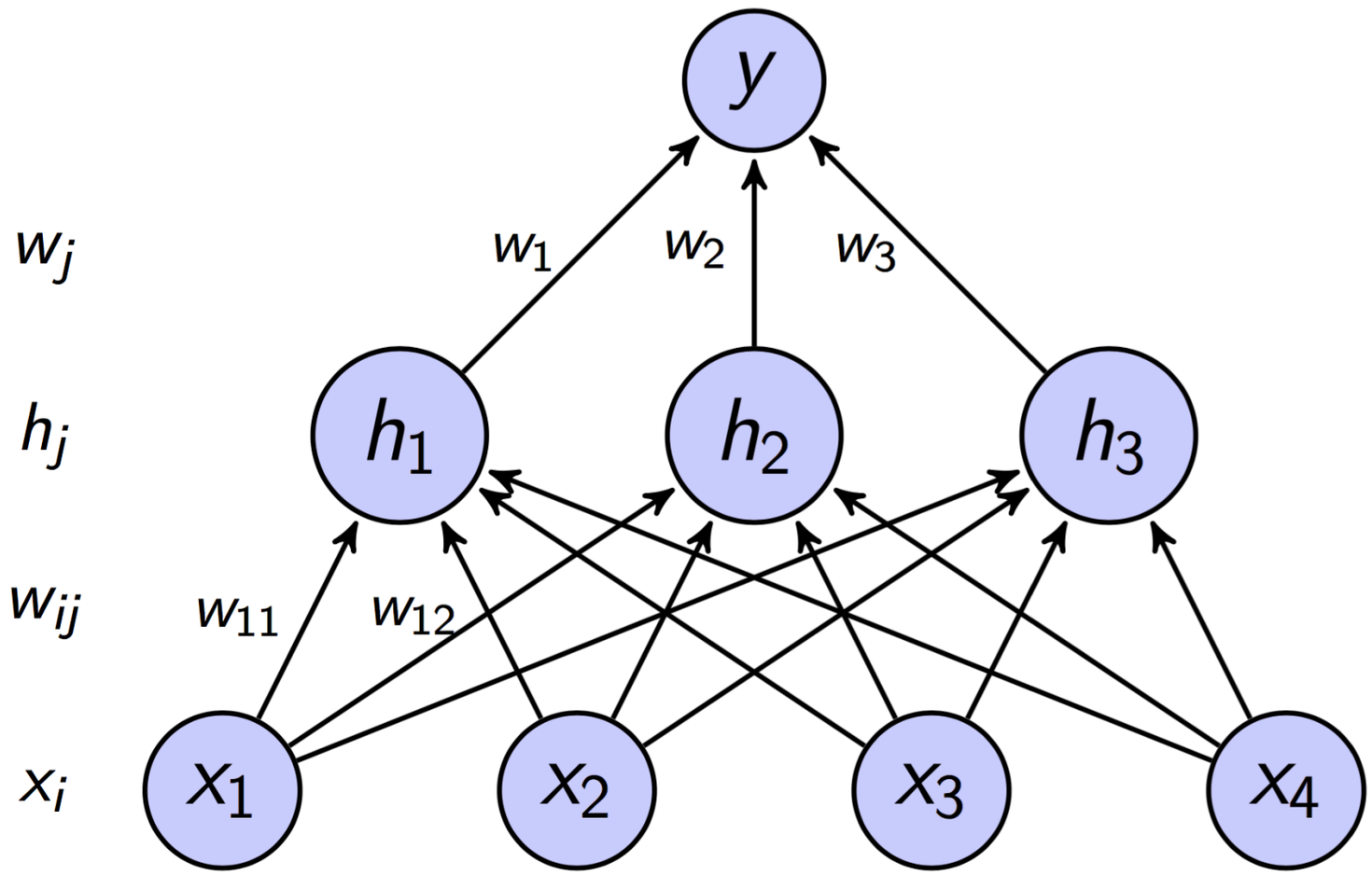
[Chapter 6, Goodfellow et al.]

Feedforward networks

- Objective: approximate a function f^*
- How: learning a mapping: $\boldsymbol{y} = f(\boldsymbol{x}; \boldsymbol{\theta})$

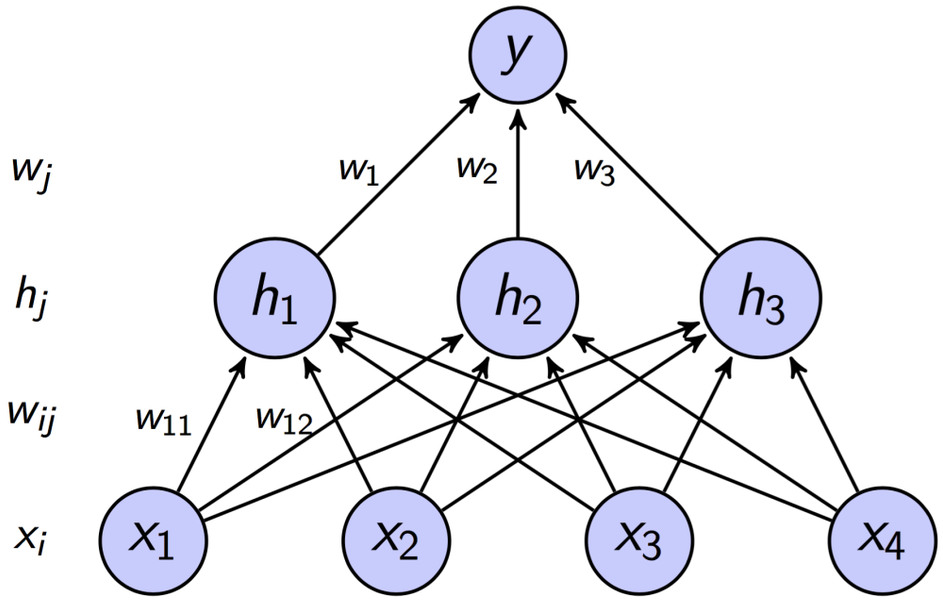
Parameters to learn

Feedforward networks



$$f(x) = \sigma(\sum_j w_j \cdot h_j) = \sigma(\sum_j w_j \cdot \sigma(\sum_i w_{ij} x_i))$$

In practice: network architecture



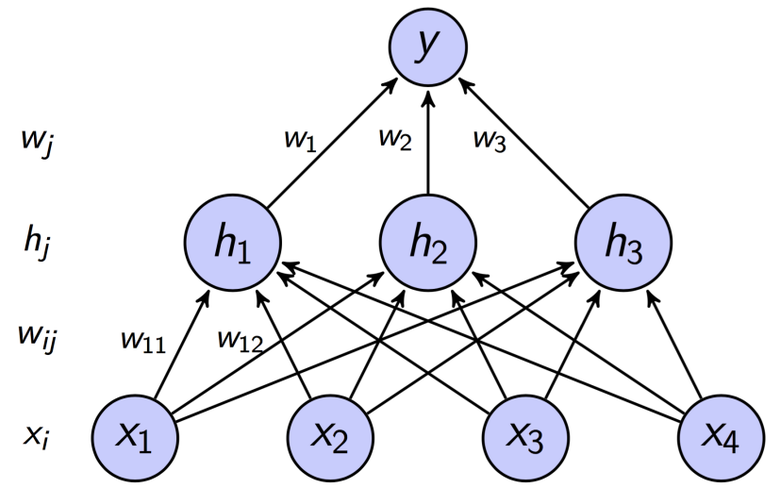

```
from keras import models
from keras import layers
```

```
network = models.Sequential()
network.add(layers.Dense(3, activation='relu', input_shape=(4,)))
network.add(layers.Dense(1, activation='softmax'))
```

Feedforward networks

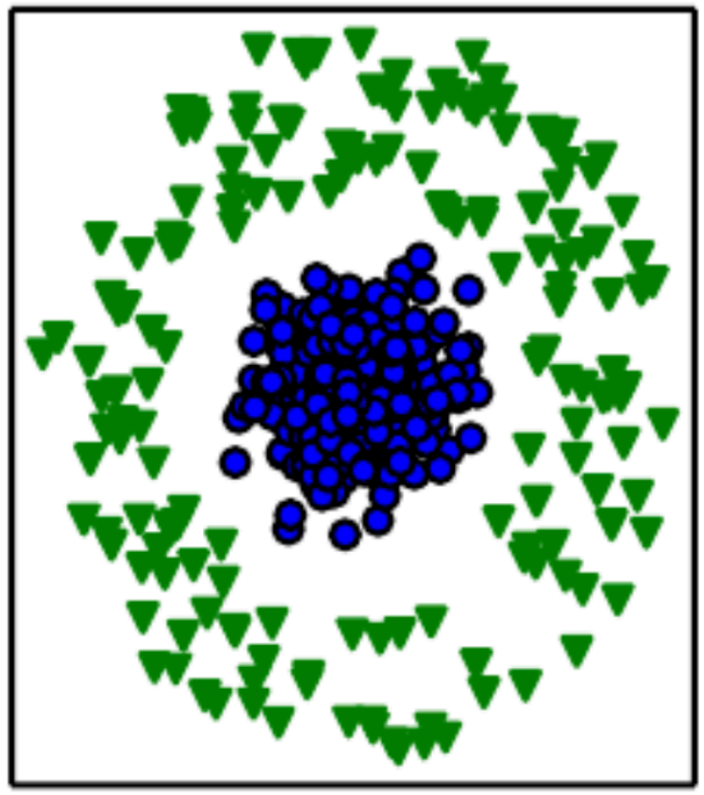
- Feedforward: no feedback connection

- Network: composition of functions



- Neuron: basic unit, inspired from neuroscience

Feedforward networks



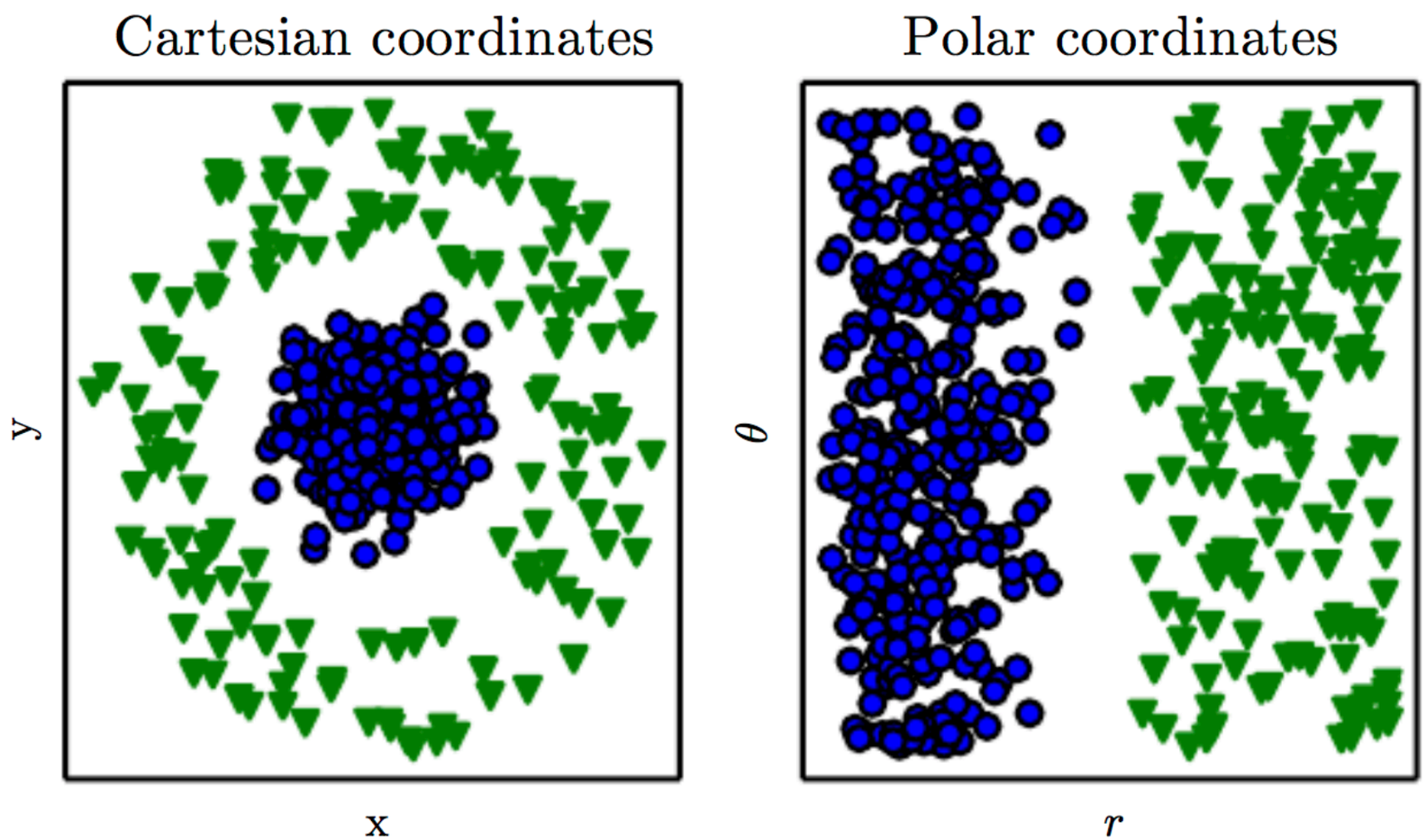
[Goodfellow et al. 2016]

Feedforward networks

- Linear models: $\boldsymbol{y} = \boldsymbol{w}^\top \boldsymbol{x}$
  - efficient, reliable
  - closed form or convex optimization
- Non linear models in deep learning:

$$\boldsymbol{y} = f(\boldsymbol{x}; \boldsymbol{\theta}, \boldsymbol{w}) = \phi(\boldsymbol{x}; \boldsymbol{\theta})^\top \boldsymbol{w}$$

Feedforward networks

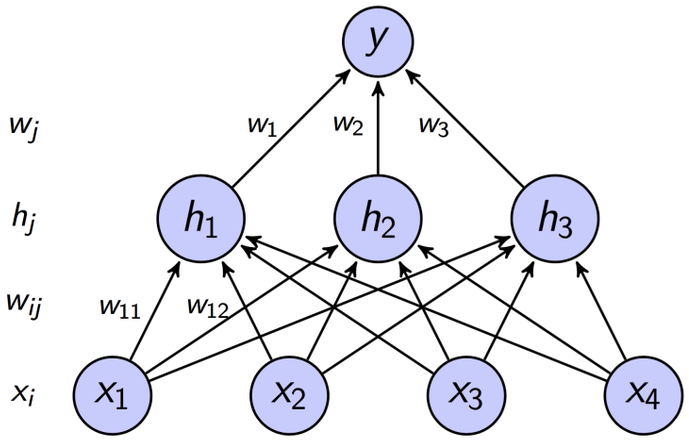


$$\boldsymbol{y} = f(\boldsymbol{x}; \boldsymbol{\theta}, \boldsymbol{w}) = \phi(\boldsymbol{x}; \boldsymbol{\theta})^\top \boldsymbol{w}$$

[Goodfellow et al. 2016]

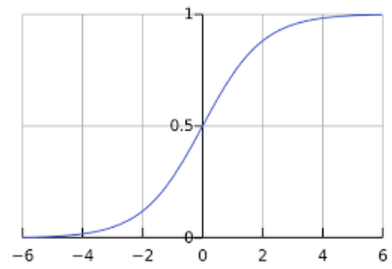
Architecture design

Activation functions



$$f(x) = \sigma(\sum_j w_j \cdot h_j) = \sigma(\sum_j w_j \cdot \sigma(\sum_i w_{ij} x_i))$$

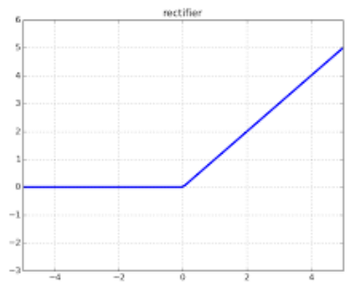
softmax



$$\sigma : \mathbb{R}^K \to (0,1)^K$$

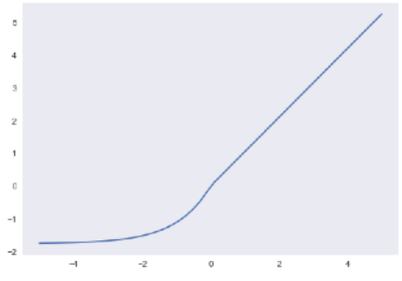
$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \ldots, K.$$

ReLU



$$f(x) = x^+ = \max(0, x)$$

selu



$$\text{selu}(x) = \lambda \begin{cases} x & \text{if } x > 0 \\ \alpha e^x - \alpha & \text{if } x \leqslant 0 \end{cases}$$

etc.

A mostly complete chart of

Neural Networks

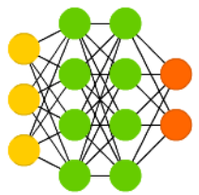
- Backfed Input Cell
- Input Cell
- Noisy Input Cell
- Hidden Cell
- Probablistic Hidden Cell
- Spiking Hidden Cell
- Output Cell
- Match Input Output Cell
- Recurrent Cell
- Memory Cell
- Different Memory Cell
- Kernel
- Convolution or Pool

Perceptron (P)

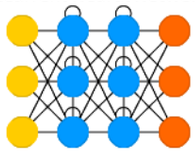
Feed Forward (FF)

Radial Basis Network (RBF)

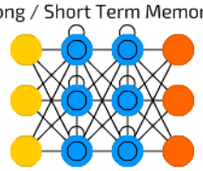
Deep Feed Forward (DFF)



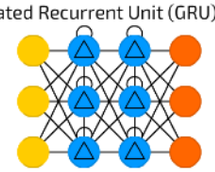
Recurrent Neural Network (RNN)



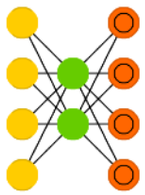
Long / Short Term Memory (LSTM)



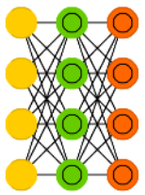
Gated Recurrent Unit (GRU)



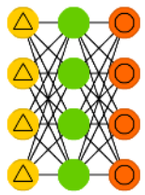
Auto Encoder (AE)



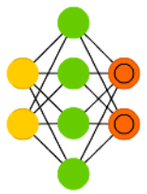
Variational AE (VAE)



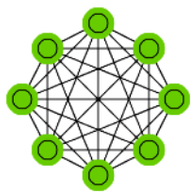
Denoising AE (DAE)



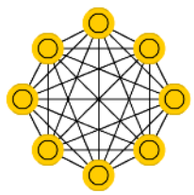
Sparse AE (SAE)



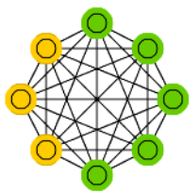
Markov Chain (MC)



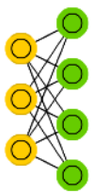
Hopfield Network (HN)



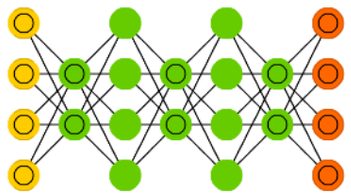
Boltzmann Machine (BM)



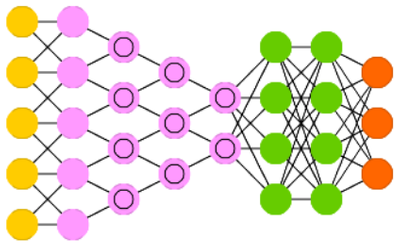
Restricted BM (RBM)



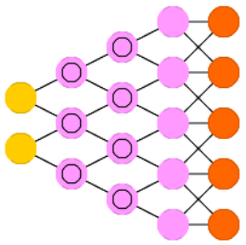
Deep Belief Network (DBN)



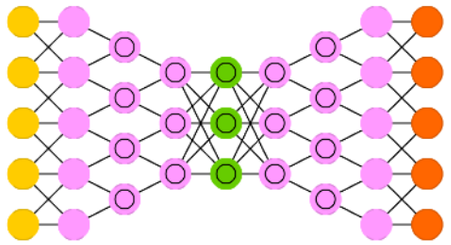
Deep Convolutional Network (DCN)



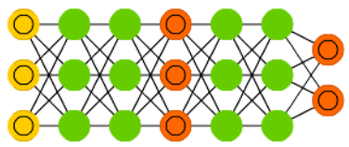
Deconvolutional Network (DN)



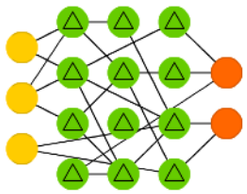
Deep Convolutional Inverse Graphics Network (DCIGN)



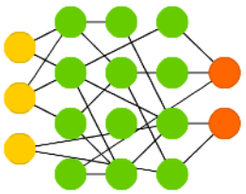
Generative Adversarial Network (GAN)



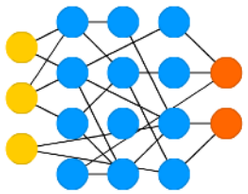
Liquid State Machine (LSM)



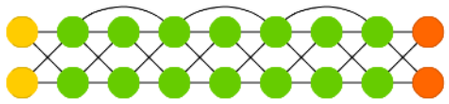
Extreme Learning Machine (ELM)



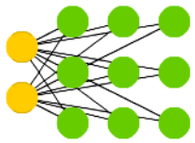
Echo State Network (ESN)



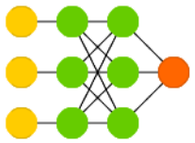
Deep Residual Network (DRN)



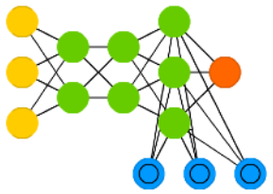
Kohonen Network (KN)



Support Vector Machine (SVM)

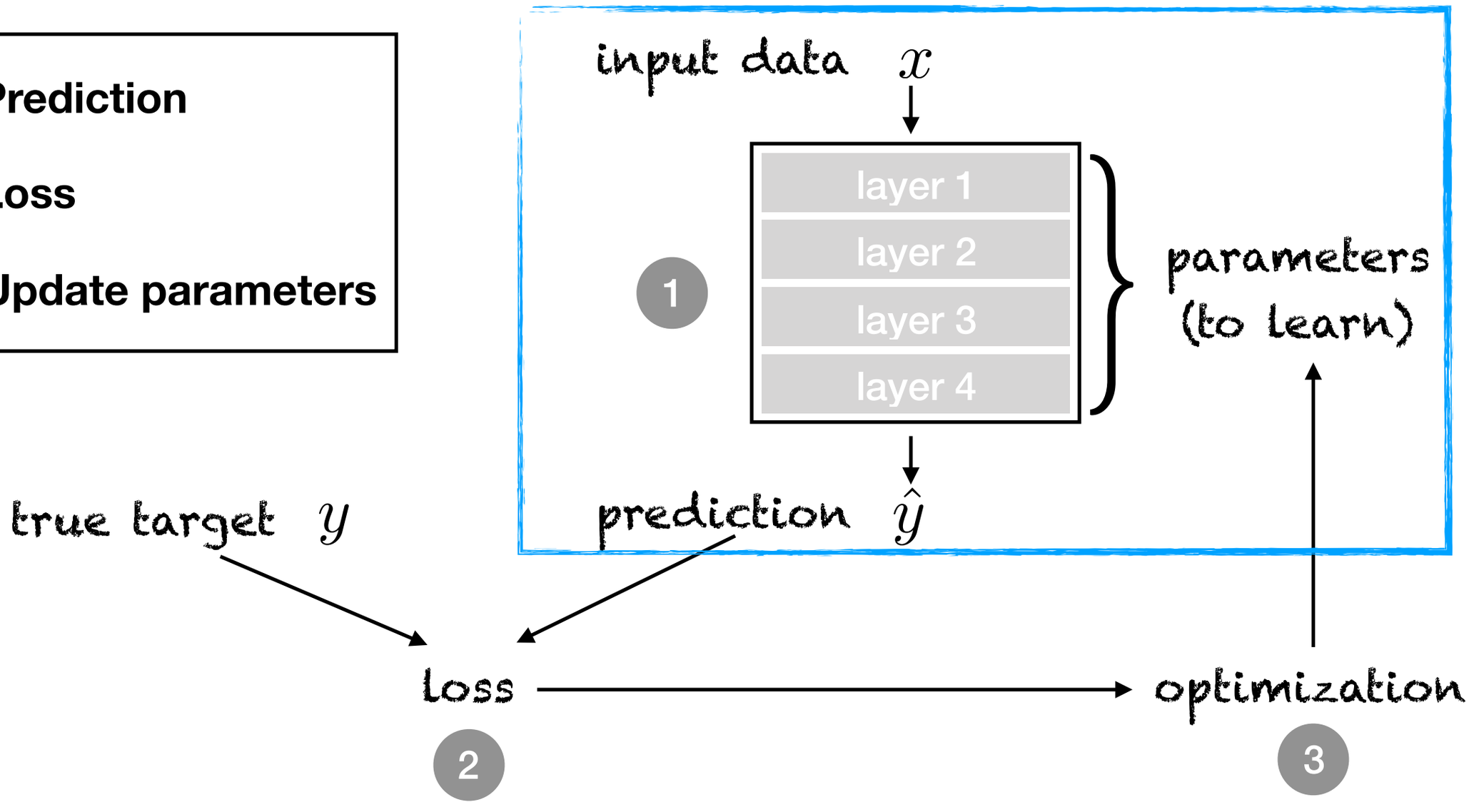


Neural Turing Machine (NTM)



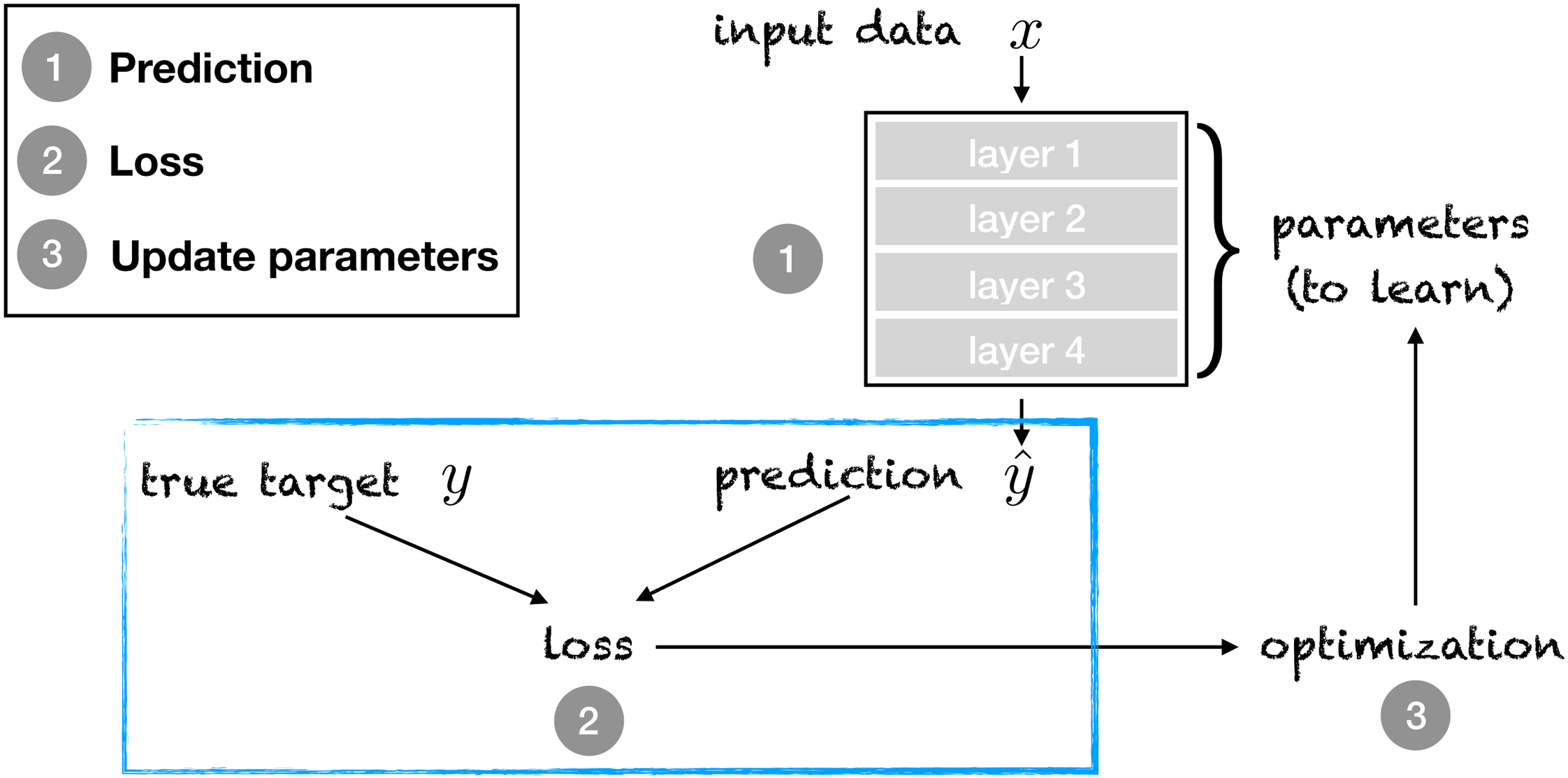
Overview

1. **Prediction**
2. **Loss**
3. **Update parameters**



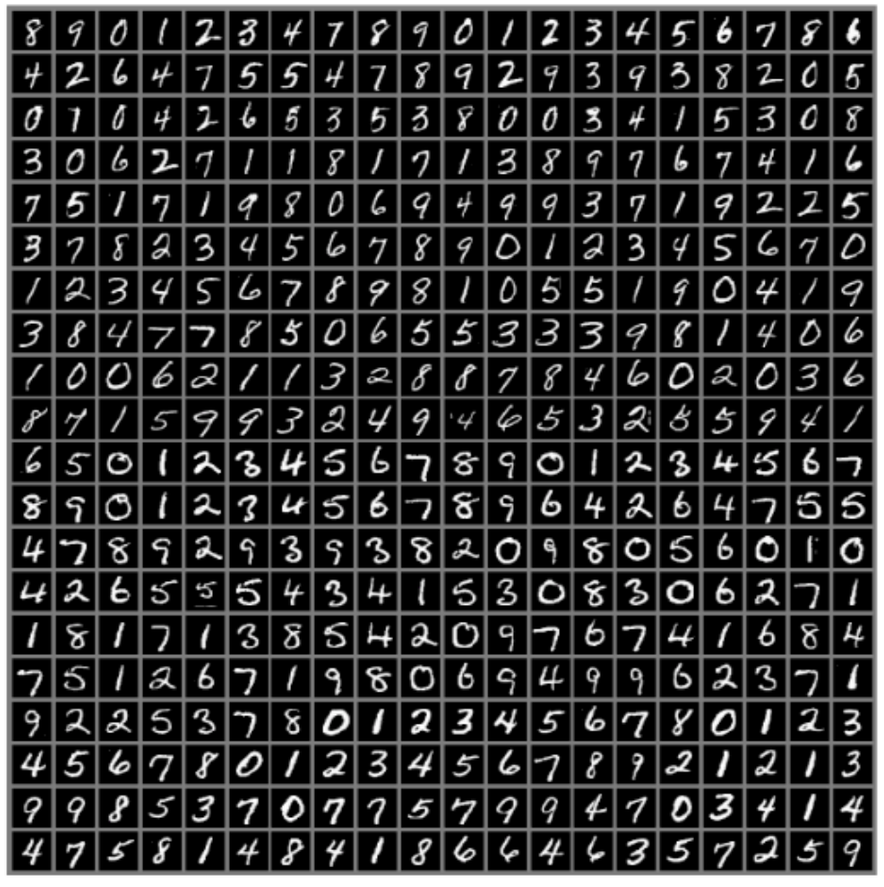
Overview

1. **Prediction**
2. **Loss**
3. **Update parameters**



Cost function

MNIST dataset
  input: image
  output: a class (one-hot encoding)



How to choose the loss function ?

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} |y^{(i)} - f(x^{(i)};\theta)|$$

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \Big(y^{(i)} - f(x^{(i)};\theta)\Big)^2$$

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m} y^{(i)} \log f(x^{(i)};\theta)$$

etc.

Cost Function

(also called loss function or objective function)

- Cost function over the training set:

$$J(\theta) = \mathbb{E}_{(x,y)\sim\hat{p}_{data}} L(f(x;\theta), y)$$

- Empirical risk:

$$\mathbb{E}_{(x,y)\sim\hat{p}_{data}} L(f(x;\theta), y) = \frac{1}{m}\sum_{i=1}^{m} L(f(x^{(i)};\theta), y^{(i)})$$

- Learning: We want to minimize J in the hope that doing so will improve the performance measure P.

Cost Function

(also called loss function or objective function)

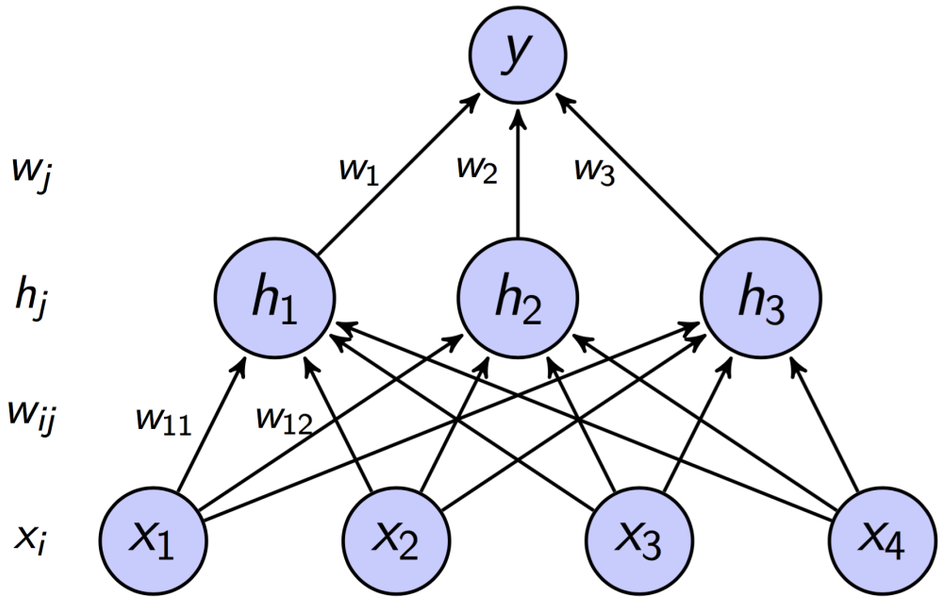
- Maximum likelihood:

$$L(f(x;\theta), y) = -\log p_{model}(y|x)$$

- if $p_{model}(y|x) = \mathcal{N}(y; f(x;\theta), \mathbf{I})$, then:

$$J(\theta) = \frac{1}{2}\mathbb{E}_{(x,y)\sim\hat{p}_{data}}||y - f(x;\theta)||^2 + cst$$

In practice: loss function




```
from keras import models
from keras import layers
```

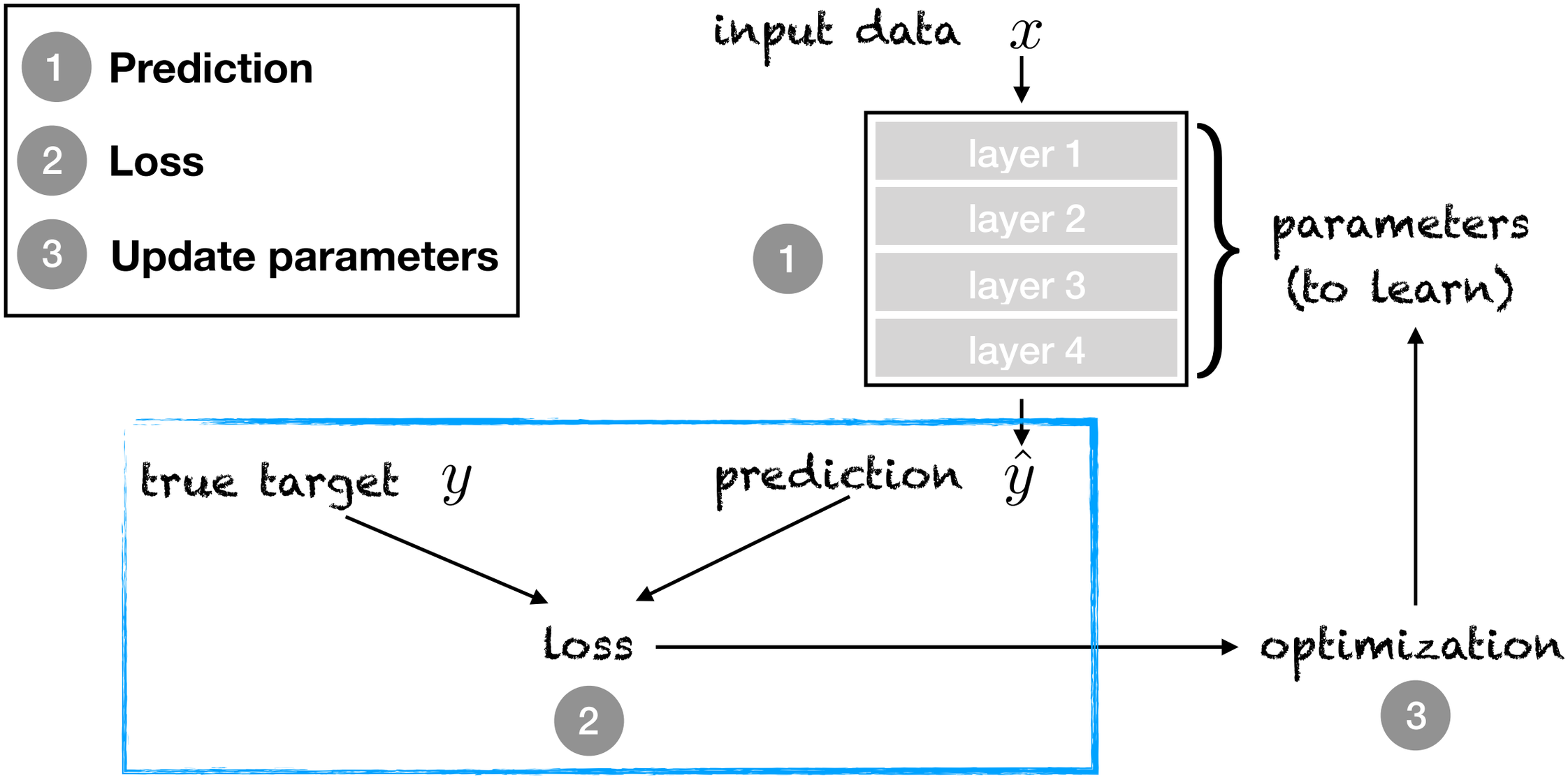
```
network = models.Sequential()
network.add(layers.Dense(3, activation='relu', input_shape=(4,)))
network.add(layers.Dense(1, activation='softmax'))
```

```
from keras.optimizers import RMSprop

model.compile(loss='categorical_crossentropy',
              optimizer=RMSprop(),
              metrics=['accuracy'])
```

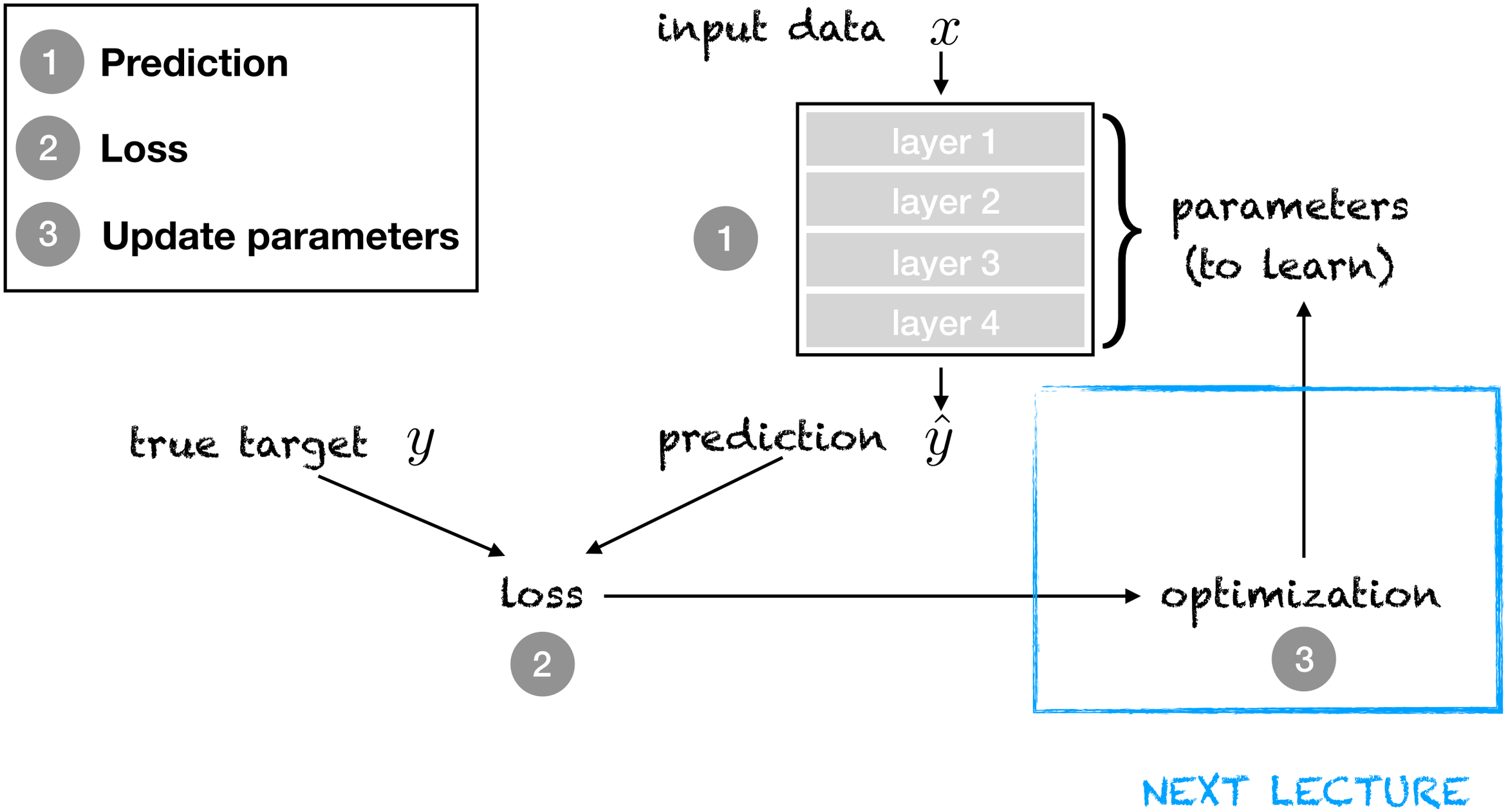
Overview

1. **Prediction**
2. **Loss**
3. **Update parameters**



Overview

1. **Prediction**
2. **Loss**
3. **Update parameters**



Things to know

- AI, ML, DL
- Data representation
- Embedding
- Neural Network Unit
- (Hidden) Layer
- Representation learning
- Feedforward network
- Network width and depth
- Activation functions
- Loss function & Optimization

Short history

Early days of AI. Invention of artificial neuron [McCulloch and Pitts, 1943] & perceptron [Rosenblatt, 1958]

AI Winter. [Minsky and Papert, 1969] showed perceptron only learns linearly separable concepts

Revival in 1980s: Multi-layer Perceptrons (MLP) and Back-propagation [Rumelhart et al., 1986]

Other directions (1990s - present): SVMs, Bayesian Networks

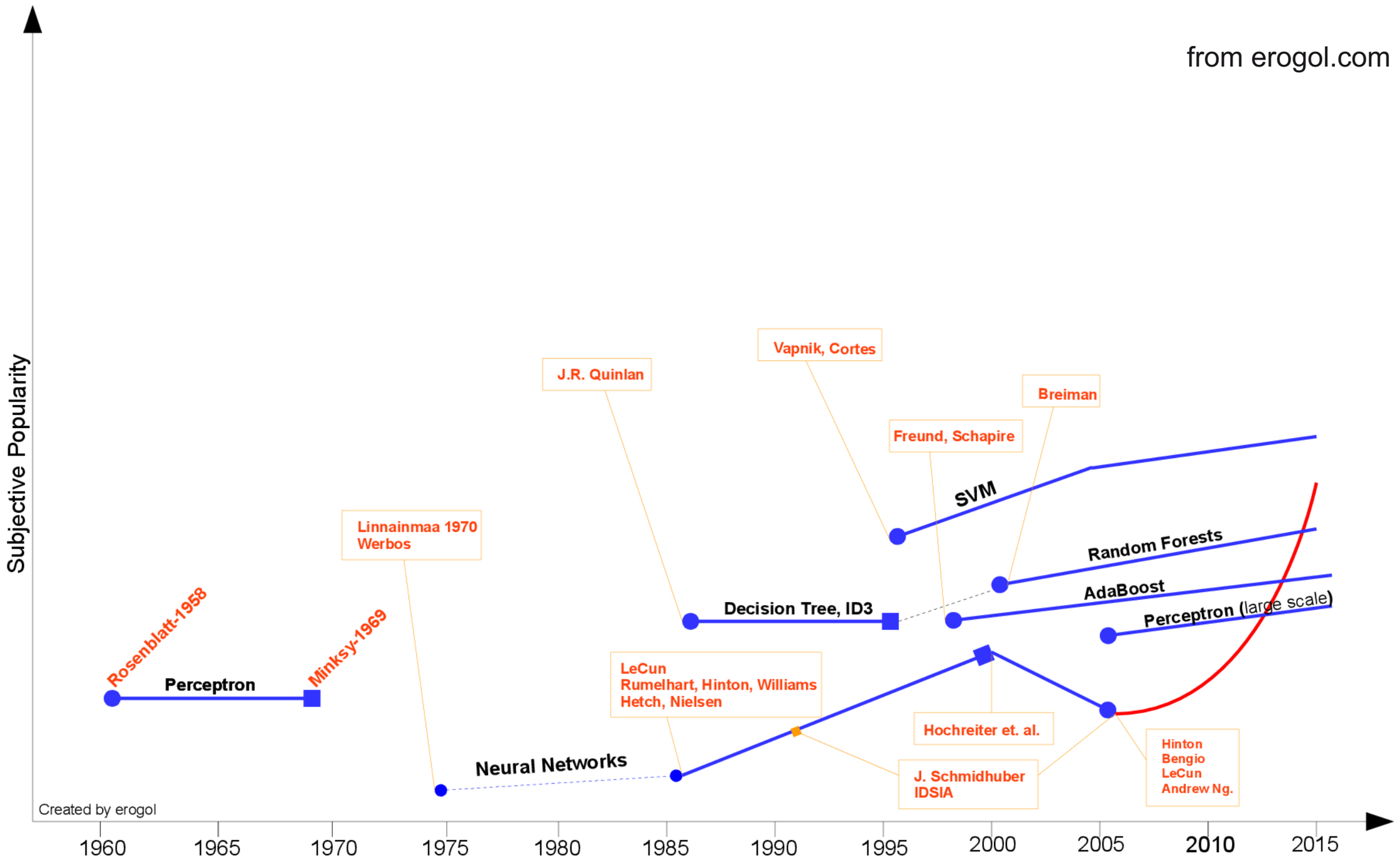
Revival in 2006: Deep learning [Hinton et al., 2006]

Successes in applications: Speech at IBM/Toronto [Sainath et al., 2011], Microsoft [Dahl et al., 2012]. Vision at Google/Stanford [Le et al., 2012]

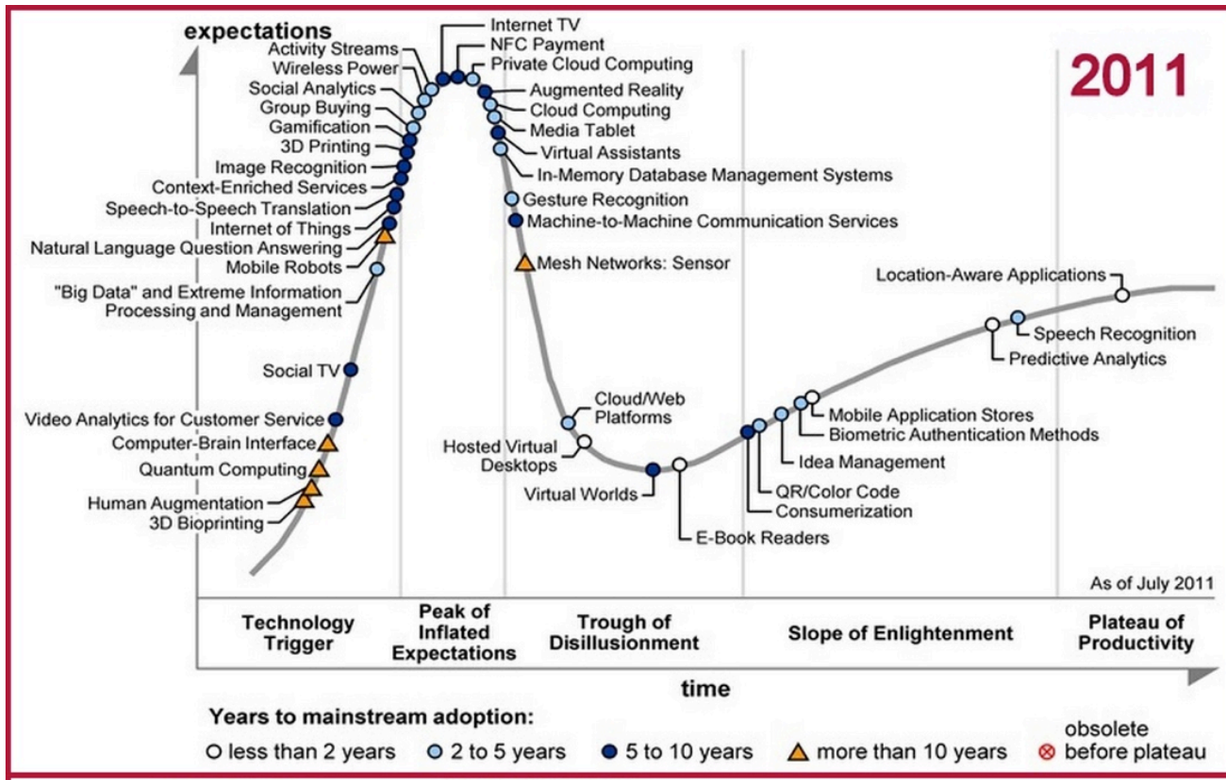
Slide from K. Duh, 2014

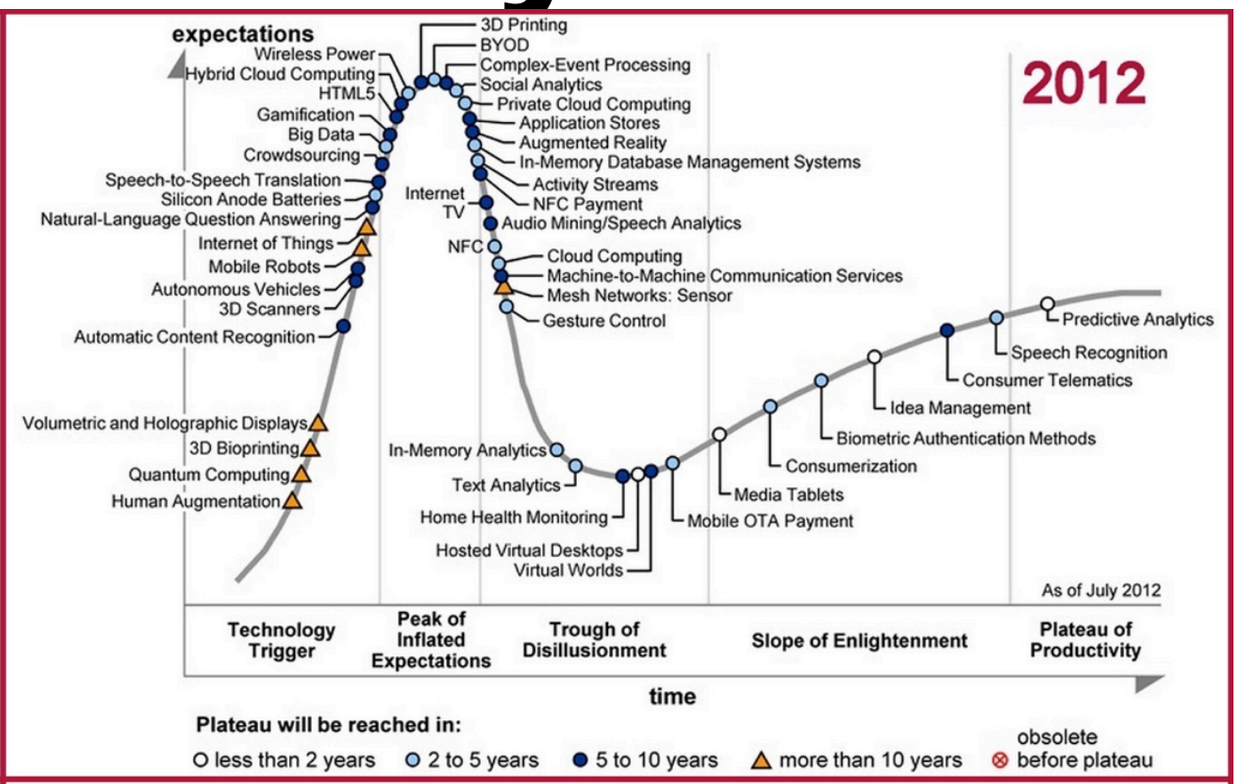
Short history

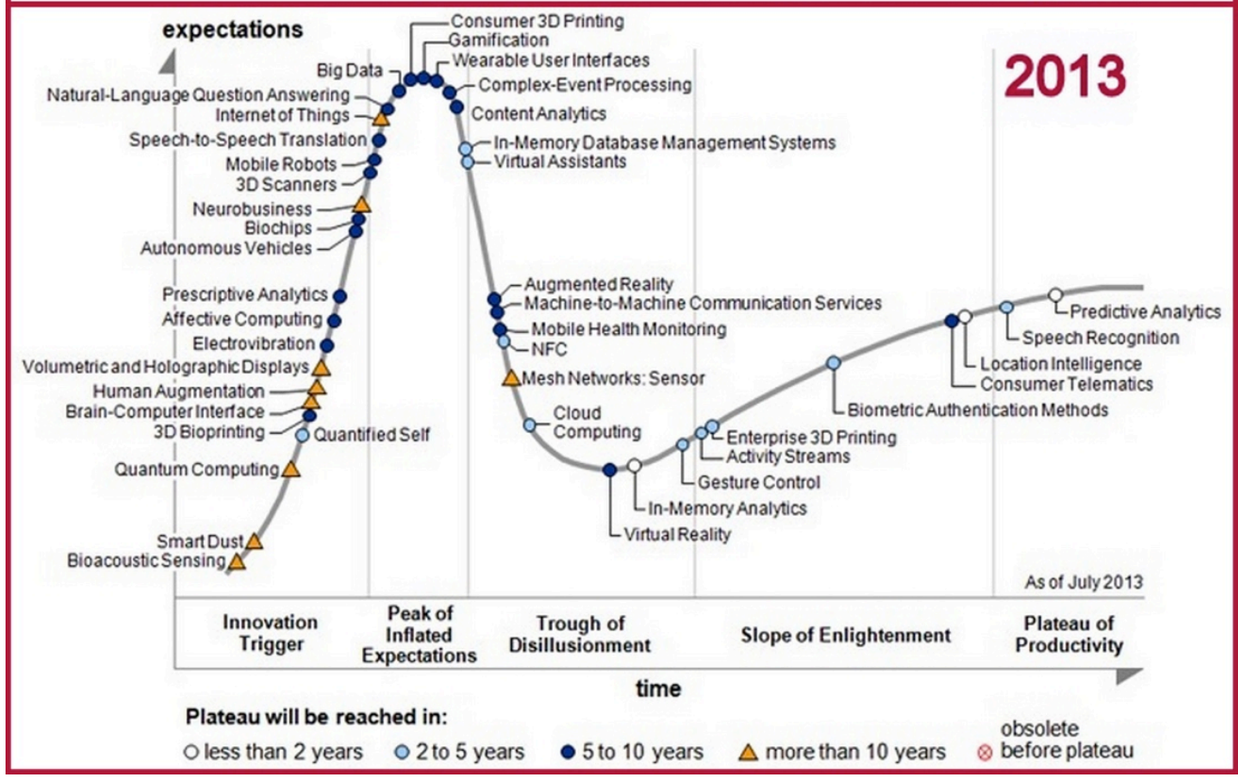
from erogol.com

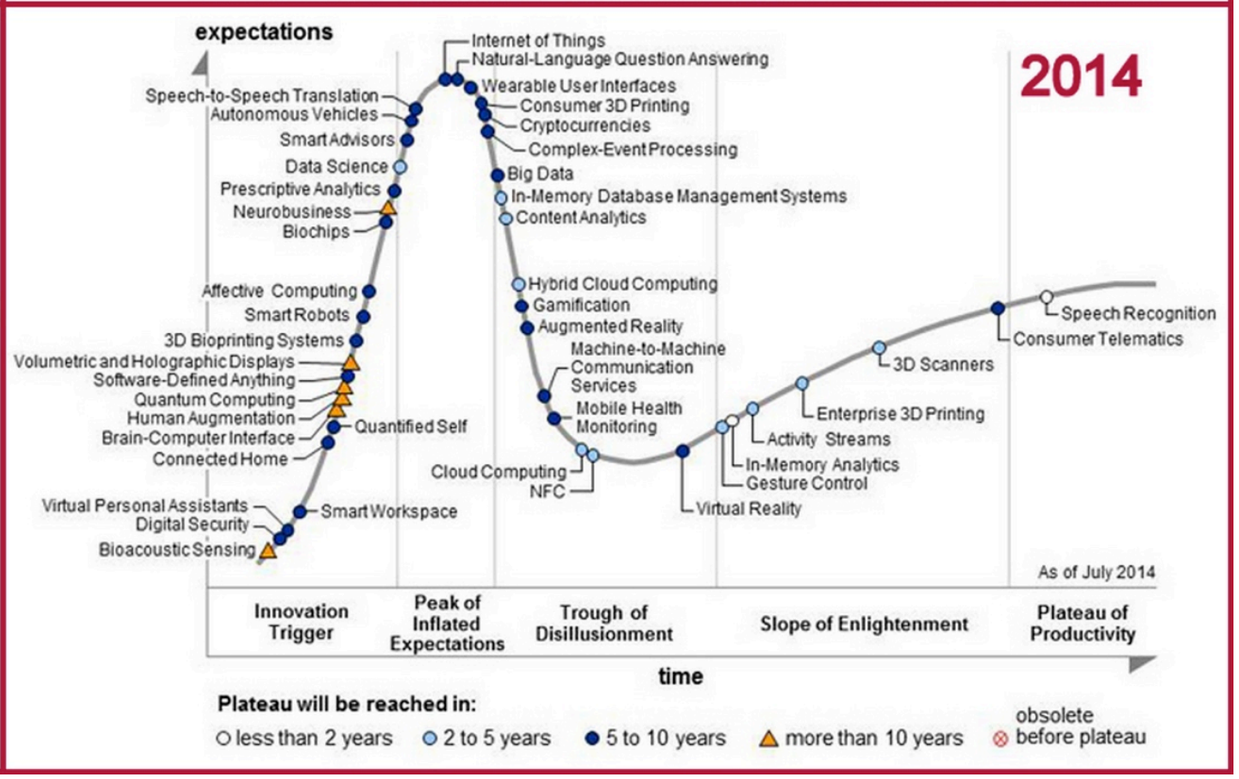


Short history

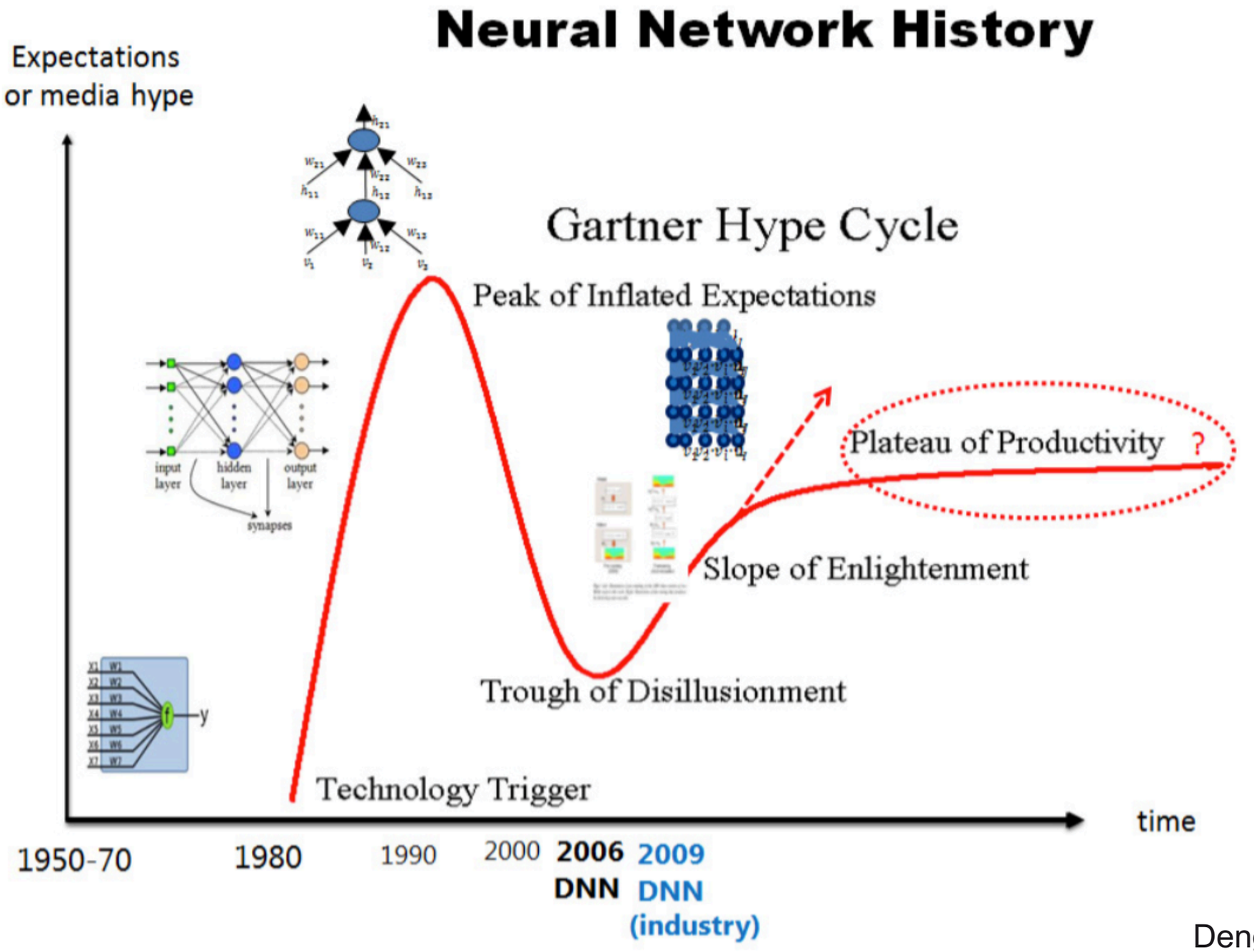








Short history



Deng et al. 2014

History of Deep Learning

Neural Networks 61 (2015) 85–117

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

Review

Deep learning in neural networks: An overview

Jürgen Schmidhuber

The Swiss AI Lab IDSIA, Istituto Dalle Molle di Studi sull'Intelligenza Artificiale, University of Lugano & SUPSI, Galleria 2, 6928 Manno-Lugano, Switzerland

ARTICLE INFO

Article history:
Received 2 May 2014
Received in revised form 12 September 2014
Accepted 14 September 2014
Available online 13 October 2014

Keywords:
Deep learning
Supervised learning
Unsupervised learning
Reinforcement learning
Evolutionary computation

ABSTRACT

In recent years, deep artificial neural networks (including recurrent ones) have won numerous contests in pattern recognition and machine learning. This historical survey compactly summarizes relevant work, much of it from the previous millennium. Shallow and Deep Learners are distinguished by the depth of their *credit assignment paths*, which are chains of possibly learnable, causal links between actions and effects. I review deep supervised learning (also recapitulating the history of backpropagation), unsupervised learning, reinforcement learning & evolutionary computation, and indirect search for short programs encoding deep and large networks.

Why Deep Learning?

- Why is deep learning so popular?
  - Very good performance
  - End-to-end approach

→ **CAR**

- Why now?
  - Hardware: GPU
  - Datasets: ImageNet, Kaggle, etc.
  - User friendly libraries: Keras, Caffe, Torch, Tensorflow, Theano, etc.
  - Algorithmics advances

Schedule

- 4 lectures
- 4 lab sessions
- 2 graded lab sessions
- 4 flipped classrooms
- 1 project
- 1 invited talk

Flipped classroom

- One book chapter + one video to prepare in advance (i.e. reading, taking notes, prepare two or three questions)
- MCQ at the beginning of the session (10mn)
- 20 mn lecture
- 40 mn discussion
- MCQ at the end of the session (10mn)

Lab sessions (3 & 4): MNIST