The Geometry of Deep Learning. Lecture 1: Overview

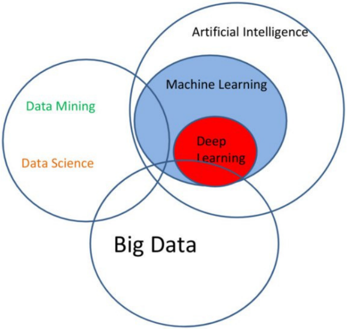
Rita Fioresi, FaBiT, Unibo

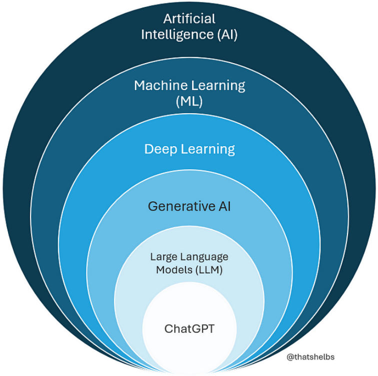
December 11, 2025

Plan of the course

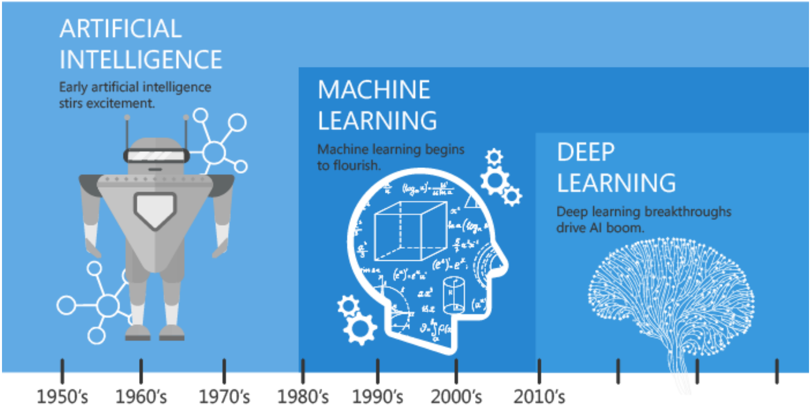
- Introduction to Deep Learning (DL)
- Information Geometry
- Neurogeometry and Deep Learning
- Geometric Deep Learning (GDL)
- Graph Neural Networks (GNN)
- Generative AI (GenAI)

AI, Machine Learning, Deep Learning and GenAI

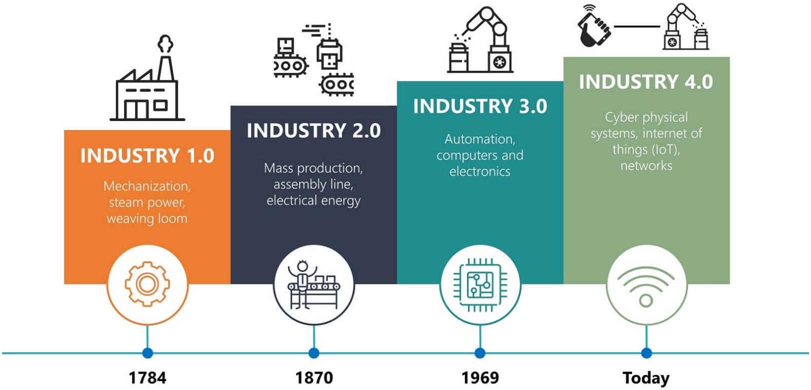




AI, Machine Learning and Deep Learning

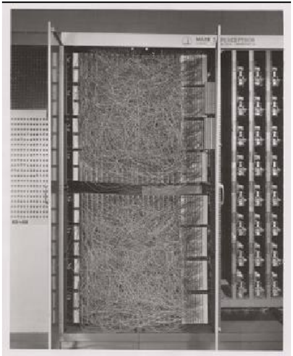


Industry 4.0


INDUSTRY 1.0
Mechanization, steam power, weaving loom
INDUSTRY 2.0
Mass production, assembly line, electrical energy
INDUSTRY 3.0
Automation, computers and electronics
INDUSTRY 4.0
Cyber physical systems, internet of things (IoT), networks
1784
1870
1969
Today

Some History: The Perceptron, a binary classifier

The first neural network: Mark I Perceptron (1957 Rosenblatt). Navy Lab, Cornell University.



First Neural Network

Perceptron (1957 Rosenblatt): algorithm for binary classification.

$$f(x) = \begin{cases} 1 & \text{if } \; w \cdot x + b > 0 \\ 0 & \text{otherwise} \end{cases}$$

Data (input and labels):

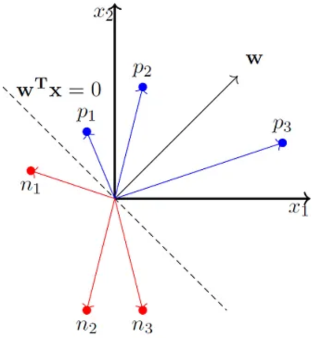
$$D = \{(\mathbf{x}_1, d_1), \dots, (\mathbf{x}_s, d_s)\}$$

Training set with s example.
$x_{j,i}$ value of the i-th feature of the j-th input.
w weight vector.

Functioning of Perceptron: binary classification

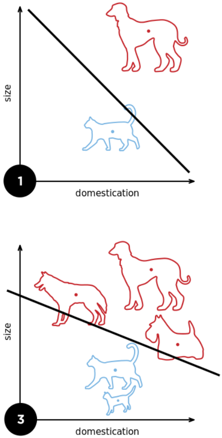
Algorithm: Perceptron Learning Algorithm

$P \leftarrow$ *inputs with label* 1;
$N \leftarrow$ *inputs with label* 0;
Initialize $\mathbf{w}$ randomly;
while !*convergence* **do**
  Pick random $\mathbf{x} \in P \cup N$;
  if $\mathbf{x} \in P$ *and* $\mathbf{w}.\mathbf{x} < 0$ **then**
    $\mathbf{w} = \mathbf{w} + \mathbf{x}$;
  end
  if $\mathbf{x} \in N$ *and* $\mathbf{w}.\mathbf{x} \geq 0$ **then**
    $\mathbf{w} = \mathbf{w} - \mathbf{x}$;
  end
end
//the algorithm converges when all the inputs are classified correctly

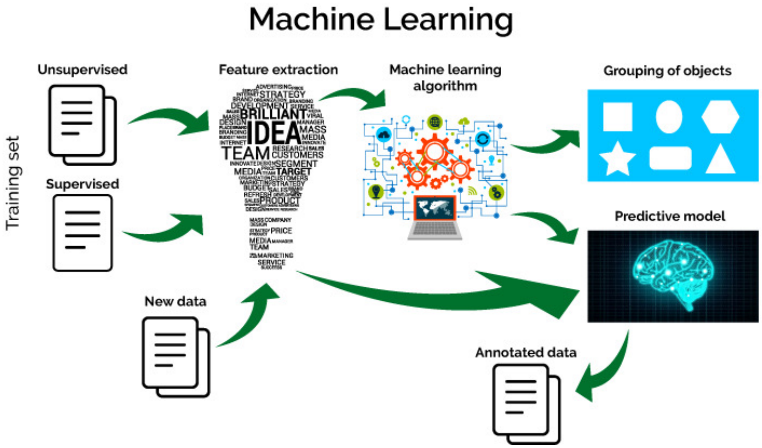


Linear Classification

Perceptron works only for linearly separable datasets.

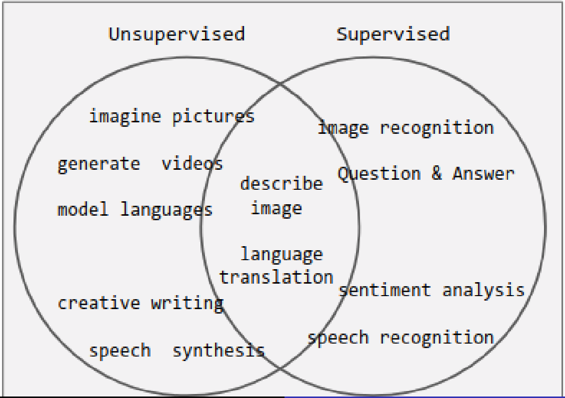


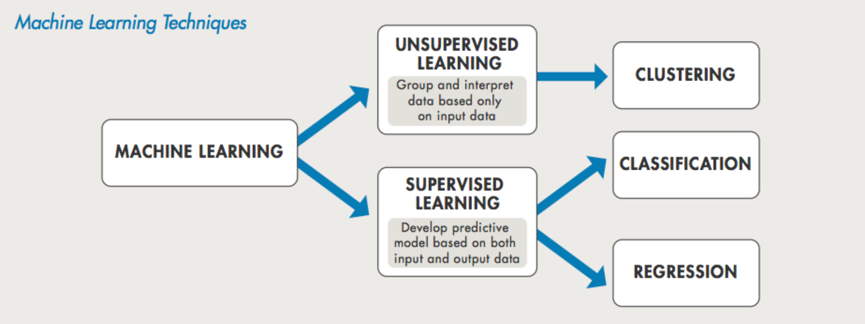
Algorithms of Machine Learning



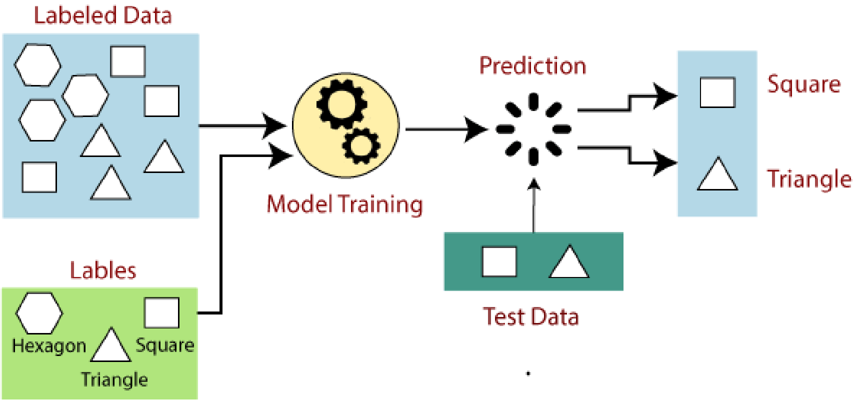
Types of Learning

- **Supervised**: weights are learned with backpropagation (perceptron).
- **Semisupervised, reinforced**: data il learned via interaction with world.
- **Unsupervised**.

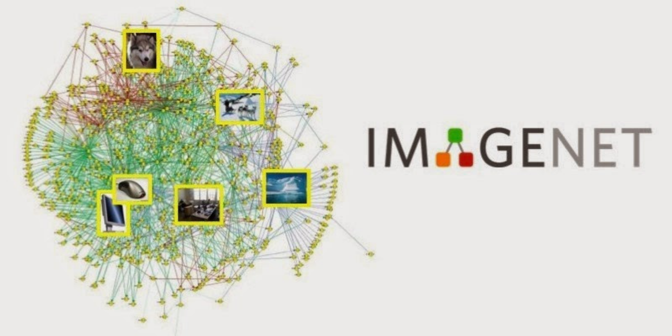



Machine Learning Techniques
MACHINE LEARNING
UNSUPERVISED LEARNING
Group and interpret data based only on input data
SUPERVISED LEARNING
Develop predictive model based on both input and output data
CLUSTERING
CLASSIFICATION
REGRESSION

Machine Learning Cycle (supervised)



Imagenet Challenge ILSVRC: ImageNet Large Scale Visual Recognition Challenge



- **2010** 20000 images, 20 categories, 25% error.
- **2011** 1 million images, 1000 categories: 16% error.
- **2015** 1 millione images, 1000 categories: 4% error.

2017: the challenge is declared won.

Images in Imagenet

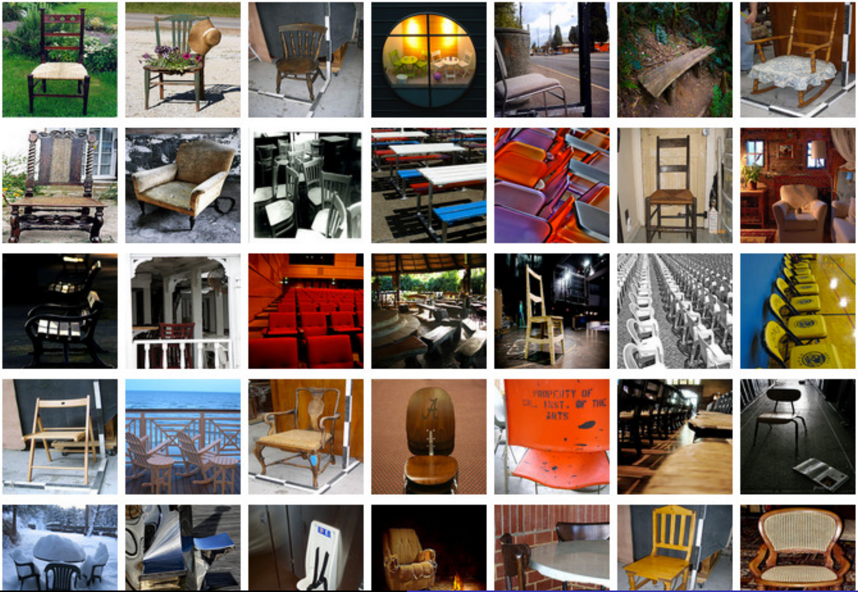
(a) ImageNet Synset: One sample image from each category

(b) Corel-1000 Dataset: Sample images from each category

(c) Caltech-256 Dataset: One sample image from each category

(d) Caltech-101 Dataset: One sample image from each category

Images in Imagenet category "chair"

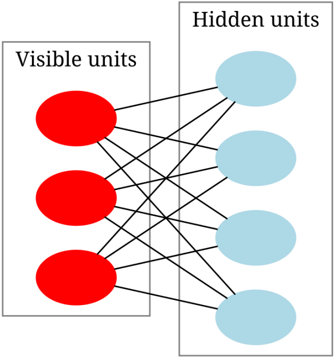


The winners

- **2012 Alexnet**: Error 15% Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton (first CNN)
- **2013 ZF Net**: Error 11% Matthew Zeiler, Rob Fergus, NYU.
- **2014 VGG Net**: Error 7.3% Karen Simonyan, Andrew Zisserman, University of Oxford
- **2014 GoogLeNet**: Error 6.7%
- **2015 Microsoft ResNet**: Error 3.6%

2024 Nobel Prize winner: Geoffrey Hinton

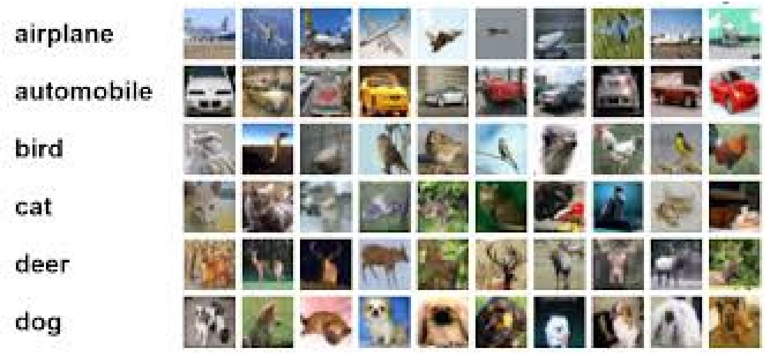
Digression: Restricted Boltzmann Machines



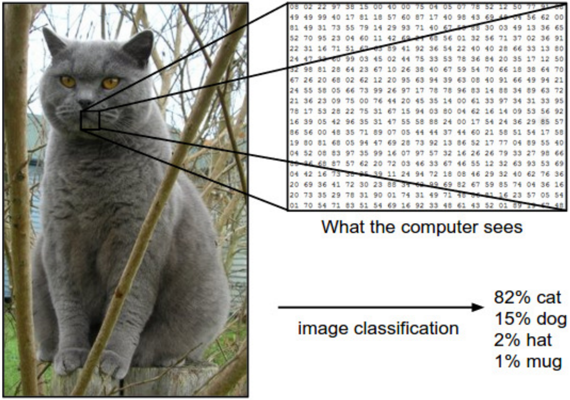
MNIST

Benchmark Datasets/2

CIFAR10



What does a computer see?



Image= set of pixels=matrix
3: RGB channels (Red Green Blue)
Every coordinate is a number between 0 and 255.

Linear Classifier

The problem: 50000 images $32 \times 32 \times 3$ in 10 categories (database CIFAR).
Score (classifier):

$$f(x)_r = \sum_{r=1}^{10} W_{rs} x_s + b_r$$

W weight matrix (inizialized randomly).
x image vector
$f(x)$ gives a score to each class: the higher score gives the answer.

Cross Enthropy Loss via Softmax

The Softmax function:

$$S(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad \text{for } i = 1, \dots, K \text{ and } \mathbf{z} = (z_1, \dots, z_K) \in \mathbb{R}^K.$$

takes as input a vector $z \in \mathbb{R}^K$, and normalizes it into a mass probability distribution.

The Loss function

$$L_i = -log \frac{e^{f_{y_i}}}{\sum_j e^{f_j}} = -f_{y_i} + log \sum_j e^{f_j}, \qquad L = \sum_i L_i$$

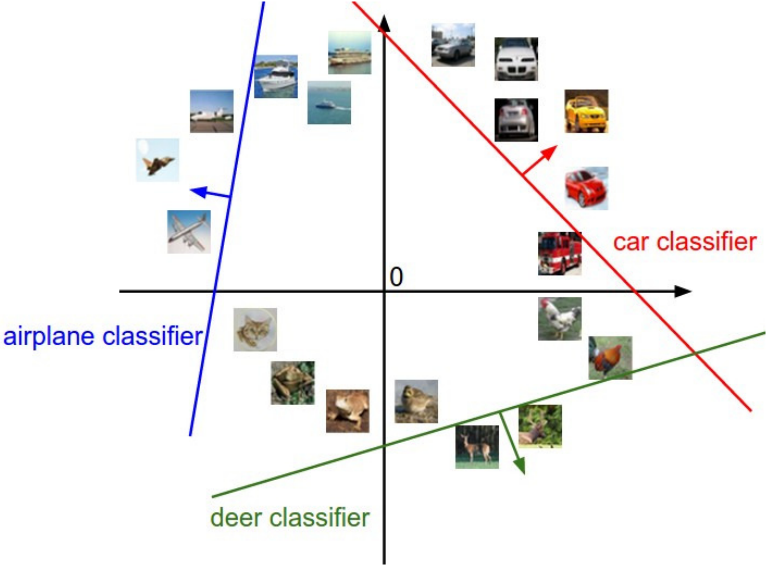
y_i: label of the i-th image class

Bias: b_i influence the score but do not depend on images.

Regularization: added to level the weights.

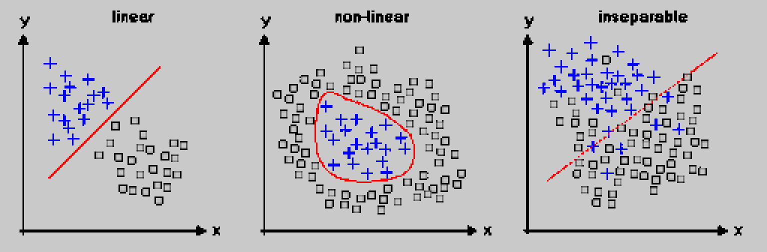
Linearly separable Databases

A linear classifies works only if database is linearly separable:

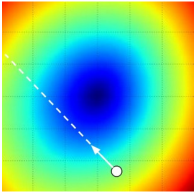


Non Linearly separable datasets

Non linear classifier works only if database is separable by curves and surfaces in some easy way.



Gradient Descent



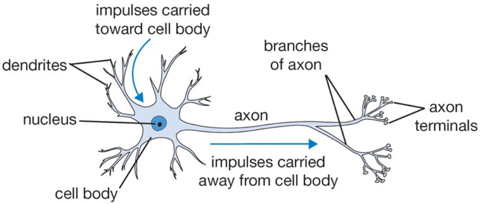
Update of the weights:

$$w_{ij}(t+1) = w_{ij}(t) - \alpha \nabla L_{\text{stoc}}$$

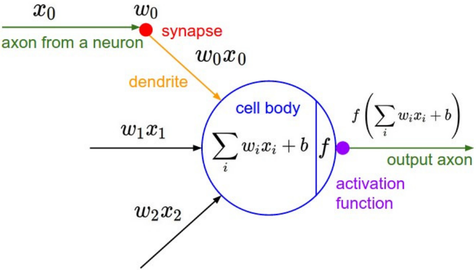
Stochastic Gradient: $\nabla L_{\text{stoc}} = \sum_{i=1}^{32} \nabla L_{\text{rand(i)}}$
True Gradient: $\nabla L = \sum_{i=1}^{50000} \nabla L_i$
Epoch: when we estimate almost all the images were examined via random sampling.
Learning rate: α (thermodynamical analysis: temperature).

A neuron as a linear classifier (perceptron)

Biological neuron

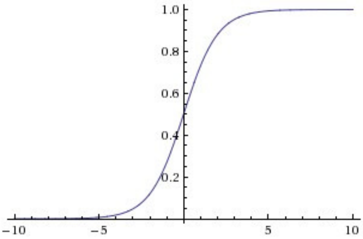


Artificial neuron

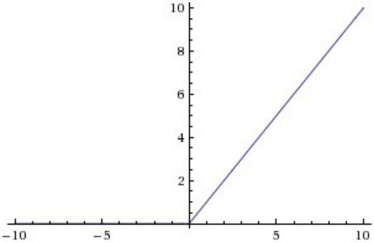


Activation Functions

- Sigmoid



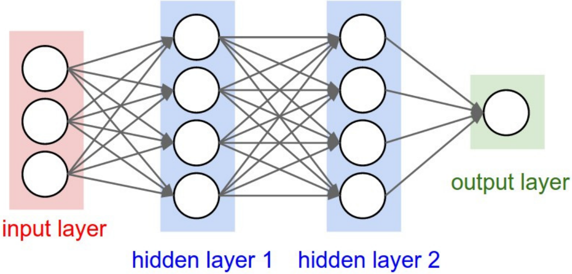
- RELU (Rectifying Linear Unit)



Neural Networks (not convolutional)

Neural Network with two hidden layers

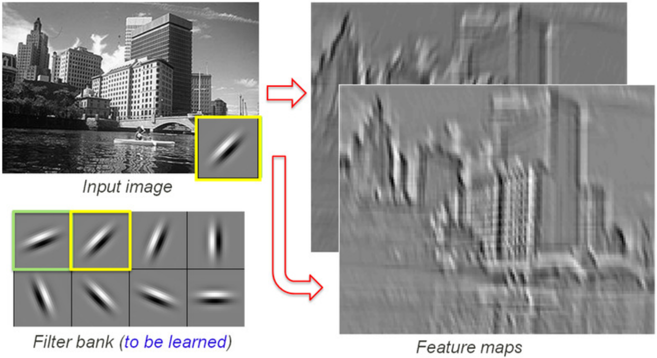
$$f(x) = W_3 max(0, W_2 max(0, W_1 x))$$



Convolutional Neural Networks

Convolutions: extract *features* from a given image.

Key: discrete convolution via filters



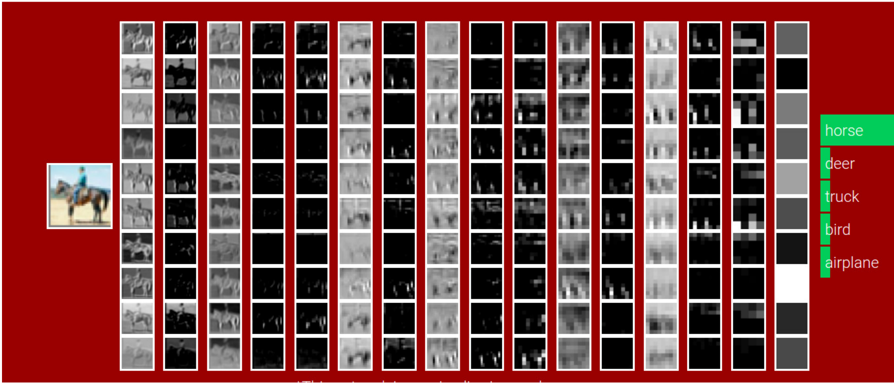
Stanford CS231
Convolutions

Deep Learning in action: CS231 Stanford

CS231n: Convolutional Neural Networks for Visual Recognition

Spring 2020

Previous Years: [Winter 2015] [Winter 2016] [Spring 2017] [Spring 2018] [Spring 2019]

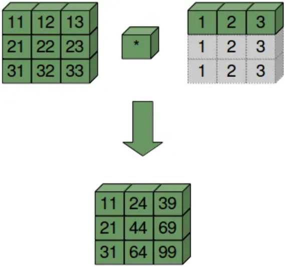


Stanford CS231

Programming Deep Learning/1

Main programming tools: TensorFlow (Google), PyTorch (Facebook).
Environment: Colab
Computer: GPU (NVidia) replacing CPU!
Programming parallel and object oriented:
Forget "if else" and "loops"!
Replace "if else" and "loops: with masks and broadcast:
we read an array at once in the memory.
Numpy: redefines all objects, overloads operators.
High Level Programming. PyTorch allows CNN programming with few lines of code.

- **Broadcasting**



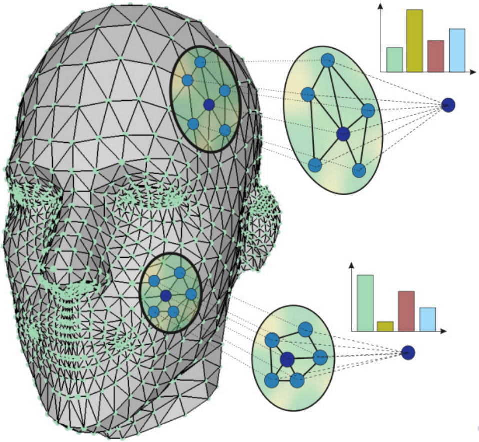
- **Masks**

Source Array	Bool Array	Destination Array		Result after copy: Destination Array
[65, 44, 77]	[T, F, T]	[85, 10, 20]	→	[65, 10, 77]
[25, 22, 31]	[F, F, T]	[15, 12, 32]		[15, 12, 31]
[14, 20, 63]	[F, T, F]	[66, 28, 13]		[66, 20, 13]

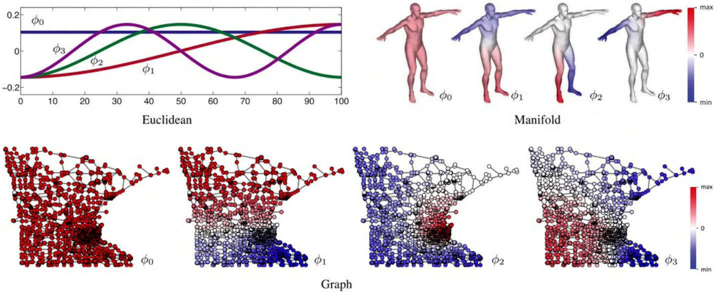
Graph Neural Networks GNN

Geometric Deep Learning: Bronstein et al, 2016.

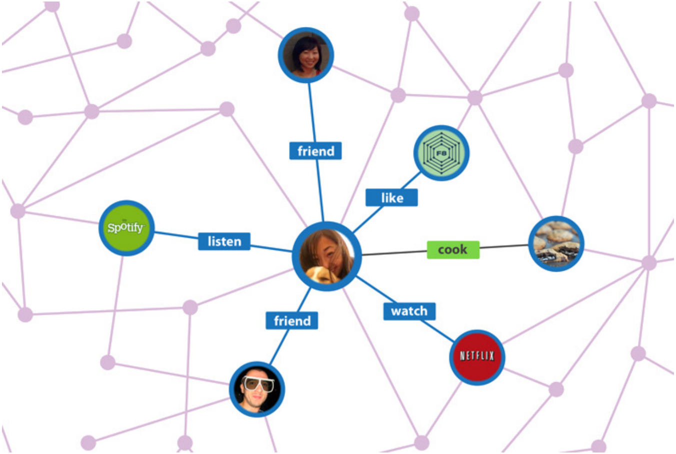
- Do convolutions on graphs (called "non euclidean domains")
- Manipulate complex and heterogeneous datasets (beyond image recognition)
- Effectively work on 3D images



The eigenfunctions of the laplacian form the smoothest-possible basis function over a specific graph (they minimize the Dirichlet energy).



Geometric Deep Learning/3



Generative AI: Transformers and Chat-gpt