An introduction to AI

Machine Learning & Neural Networks

@ENSPIMA – Bordeaux INP

Jean-Luc.Charles@mailo.com

Sept. 2025

Welcome to the course "An introduction to AI: Machine Learning & Neural Networks"

An introduction to get familiarised with...

- Machine learning (ML)
- Artificial Neural Networks (ANN)
- Training & operating Artificial Neural Networks
- Applications to faults detection.

Welcome to the course "An introduction to AI: Machine Learning & Neural Networks"

My profile

- I taught Python programming (Scientific & Object Oriented) for years at ENSAM
- I started to get interested in ML in 2015
- I have written materials about ML : workshop, project & practical work for engineering schools (ENSAM, ENSEIRB, ENSPIMA...)
- Now I continue my activity as an AI / data processing self worker.

Welcome to the course "An introduction to AI: Machine Learning & Neural Networks"

Resources

- 2 lectures of 1h20
- 4 sessions of practical work MAchine Learning with Python (4 x 3h) on **your laptop**
- Dedicated GitHub repository with all the course material (PDF, notebooks, videos...)

Welcome to the course "An introduction to AI: Machine Learning & Neural Networks"

Resources

- 2 lectures of 1h20
- 4 sessions of practical work MAchine Learning with Python (4 x 3h) on **your laptop**
- Dedicated GitHub repository with all the course material (PDF, notebooks, videos...)

Evaluation

- an MCQ assessed individually (coef. 1/4)
- a report of practical work in pairs (coef. 3/4).

Practical Work: 4 × 3h

Self training: Wake up your Python !

- Start with the 2 notebooks
  `Wake_up_your_Python-part1.ipynb` and `...part2.ipynb`

Practical Work: 4 × 3h

Self training: Wake up your Python !

- Start with the 2 notebooks
  `Wake_up_your_Python-part1.ipynb` and `...part2.ipynb`

Self training: AI & Machine Learning

- 3 Python *notebooks* `ML1_MNIST_en.ipynb`, `ML2_DNN_part1_en.ipynb` and `part2` target the skills:
  - load and pre-process MNIST images
  - build a **dense** neural network with tensorflow [fr] & keras
  - train the network to classify the MNIST images
  - evaluate/operate the trained network.

Practical Work: 4 × 3h

Self training: Wake up your Python !

- Start with the 2 notebooks `Wake_up_your_Python-part1.ipynb` and `...part2.ipynb`

Self training: AI & Machine Learning

- 3 Python *notebooks* `ML1_MNIST_en.ipynb`, `ML2_DNN_part1_en.ipynb` and `part2` target the skills:
  - load and pre-process MNIST images
  - build a **dense** neural network with tensorflow [fr] & keras
  - train the network to classify the MNIST images
  - evaluate/operate the trained network.

Mini-project: application to Predictive Maintenance

- Challenge your skills with a practical use case of fault detection on a motor test bench with a neural network...

Artificial Intelligence ?

Historically[a] *badly chosen* term! Ambiguous current meaning...
Many (contradictory) definitions depending on periods and authors...

[a] first used in 1956 by John McCarthy, researcher at Stanford during the Dartmouth conference

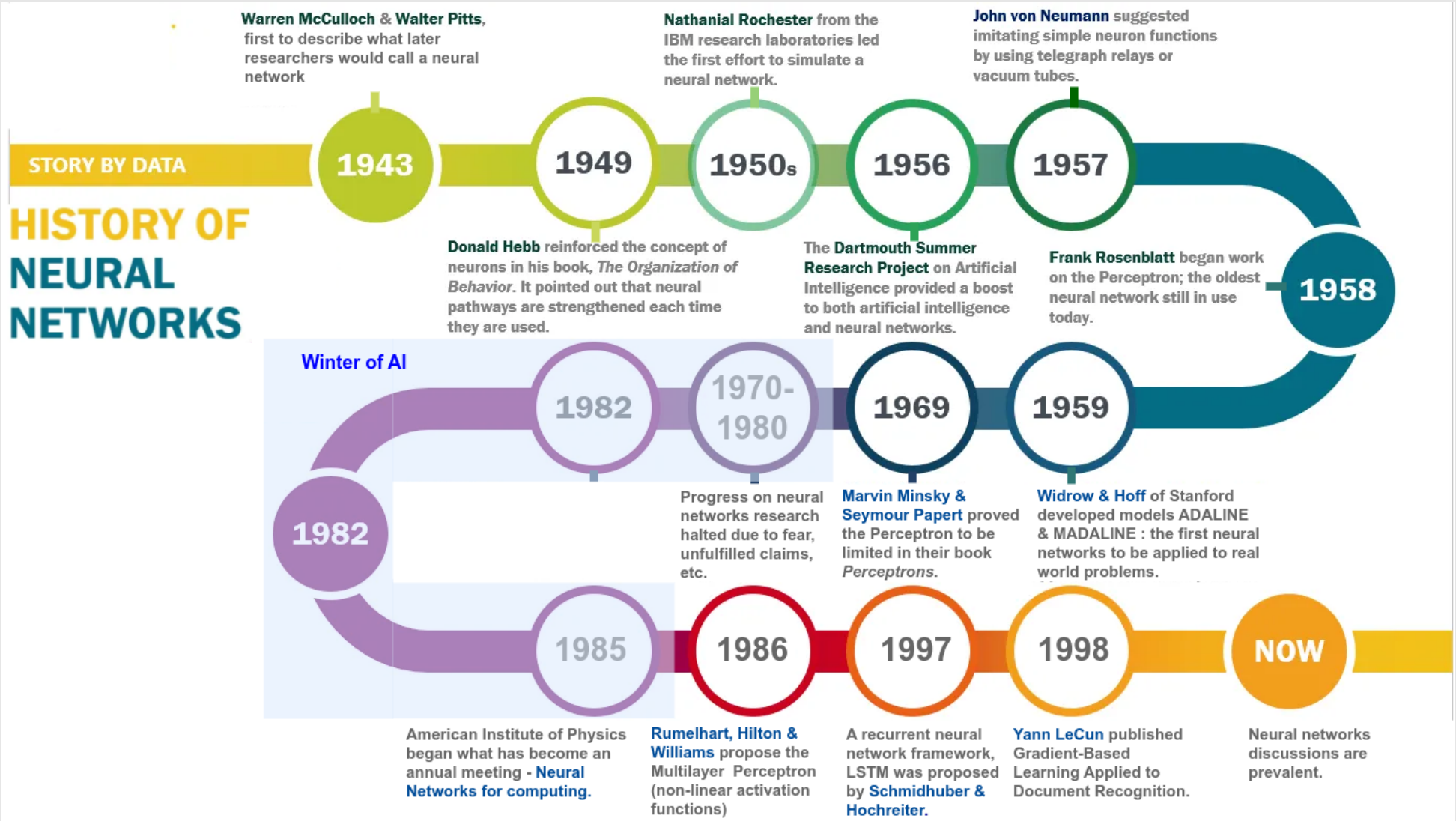
Artificial Intelligence ?

Historically[a] *badly chosen* term! Ambiguous current meaning...
Many (contradictory) definitions depending on periods and authors...

[a] first used in 1956 by John McCarthy, researcher at Stanford during the Dartmouth conference

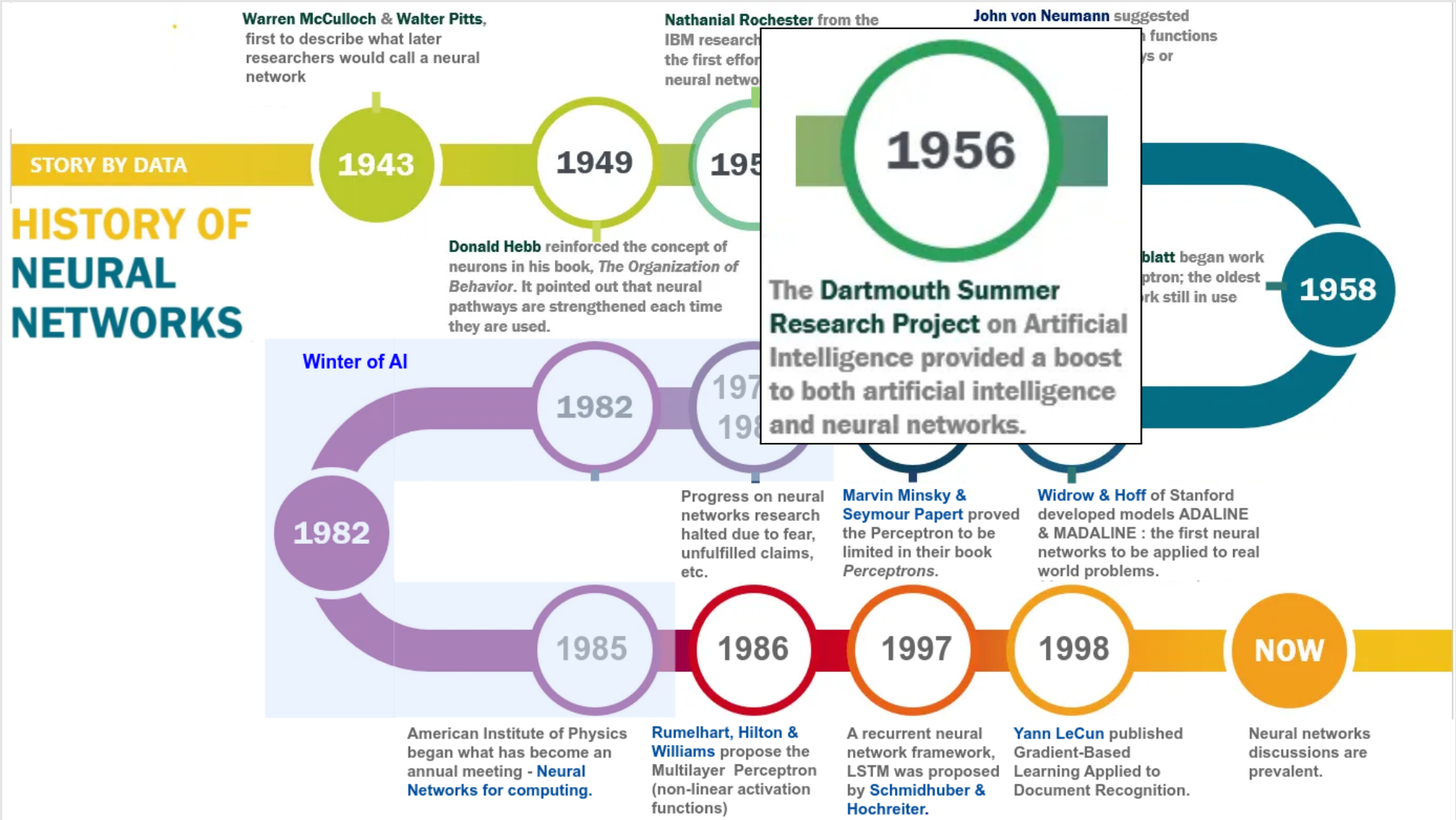
- *"...the science of making computers do things that require intelligence when done by humans."* Alan Turing, 1940
- *"the field of study that gives computers the ability to learn without being explicitly programmed."* Arthur Samuel, 1960
- *"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."* Tom Mitchell, 1997
- Notion of *intelligent agent*, *rational agent*
  "...agent that acts in such a way as to reach the best solution or, in an uncertain environment, the best predictable solution."
  Stuart Russel, Peter Norvig, "Intelligence Artificielle" 2015

AI : the historical way... from 1950 to 2000s



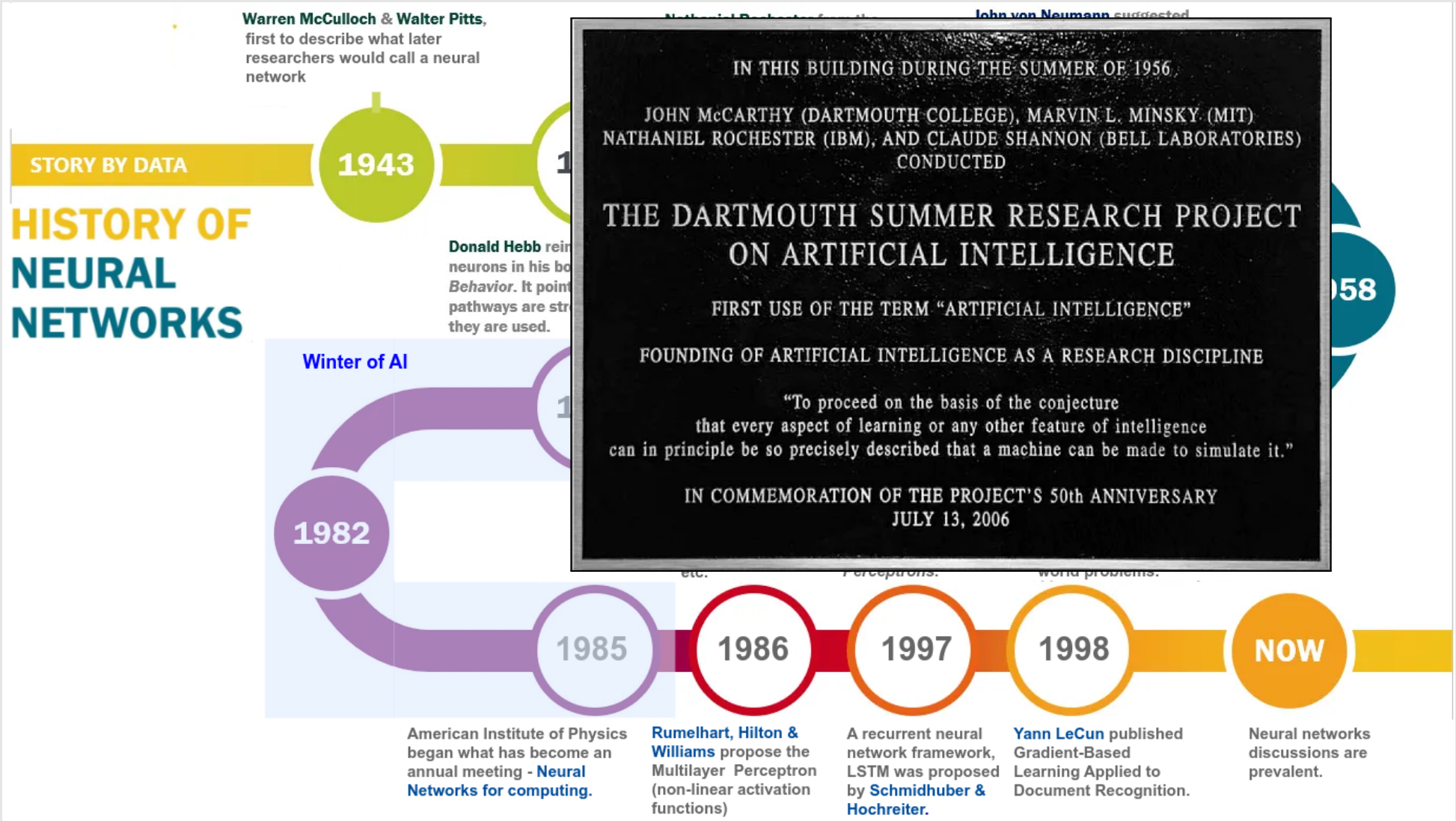
Adapted from Kate Strachni: "Brief History of Neural Netowrks", medium.com

AI : the historical way... from 1950 to 2000s



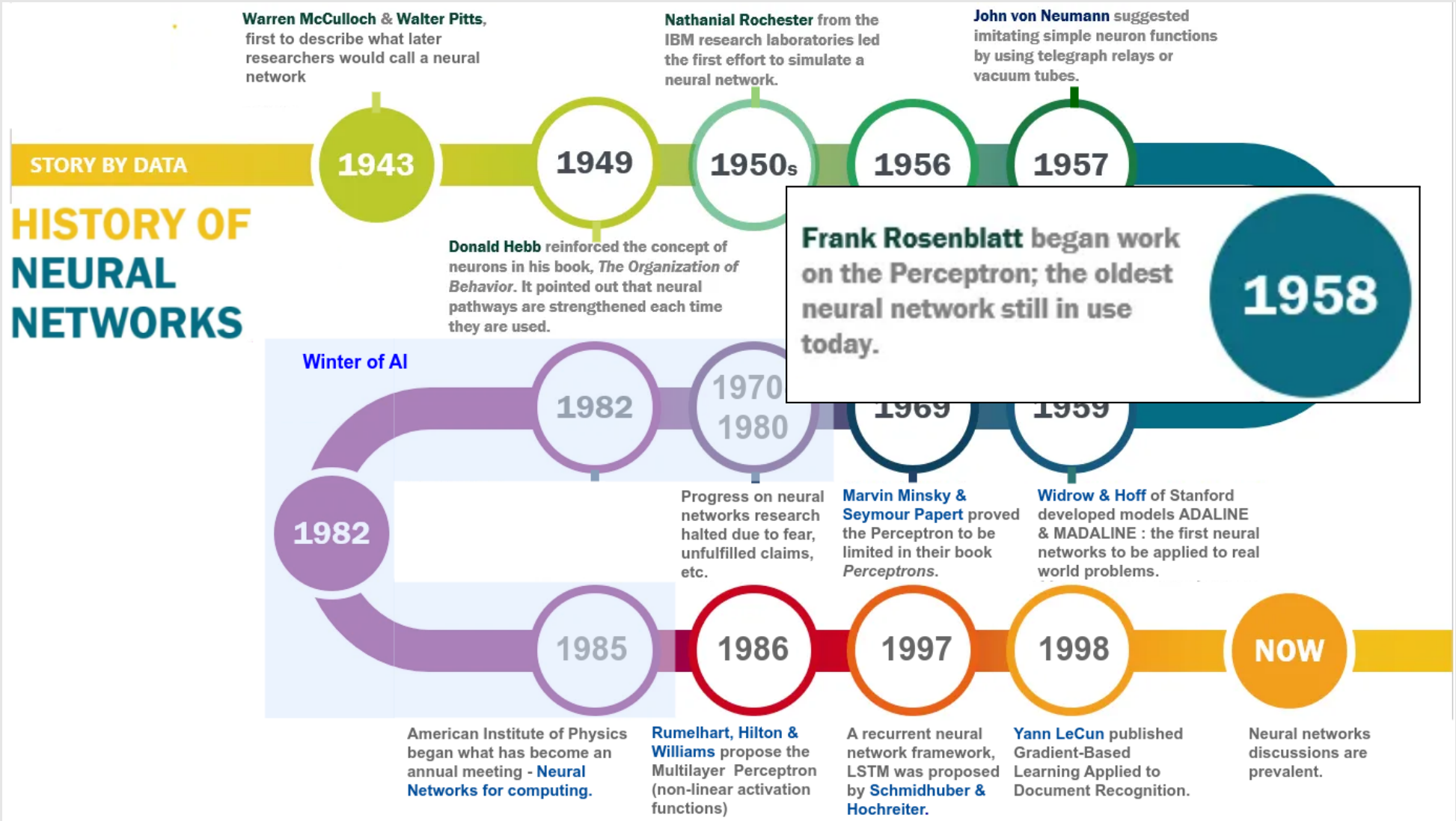
Adapted from Kate Strachni: "Brief History of Neural Netowrks", medium.com

AI : the historical way... from 1950 to 2000s



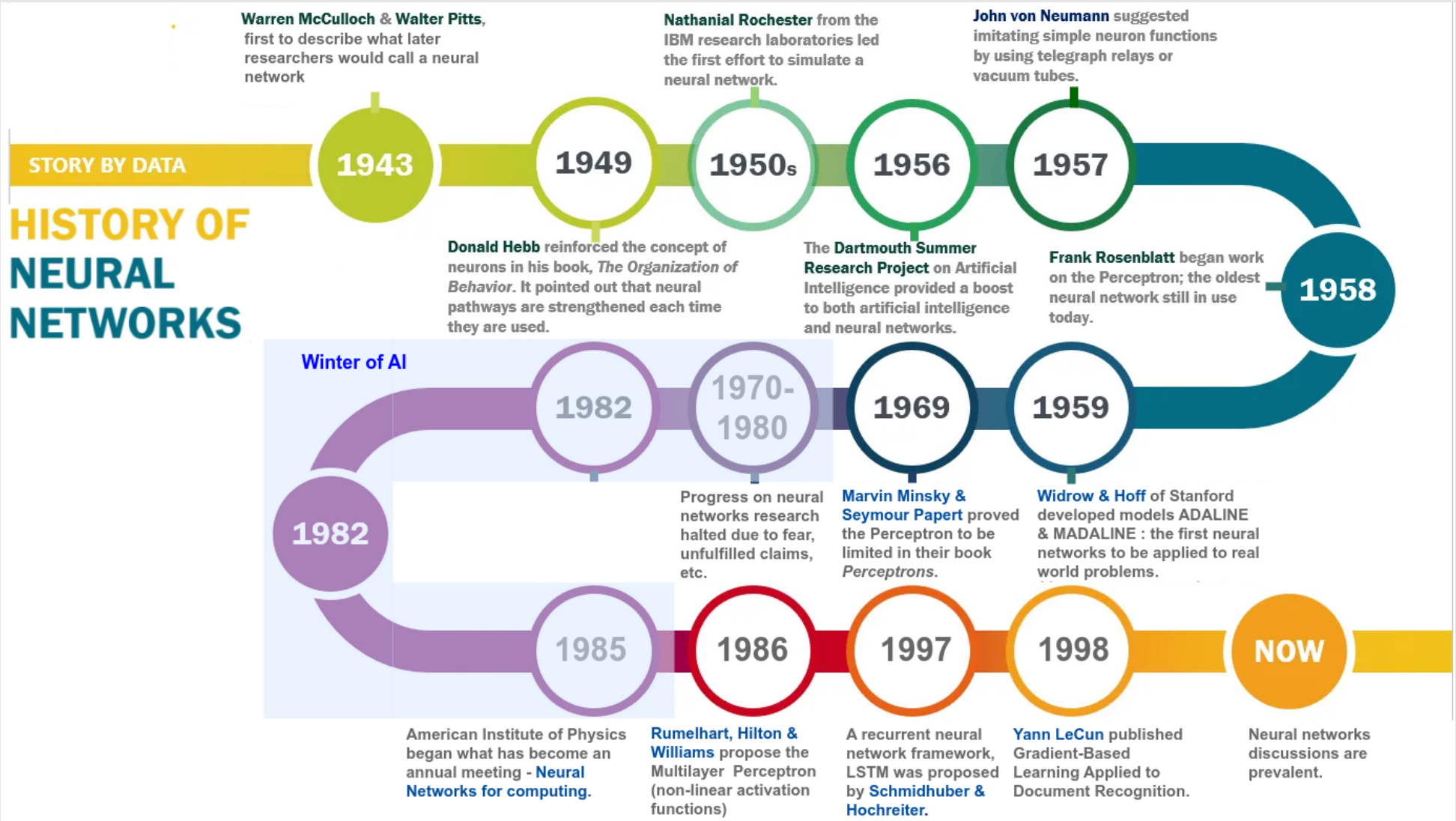
Adapted from Kate Strachni: "Brief History of Neural Netowrks", medium.com

AI : the historical way... from 1950 to 2000s



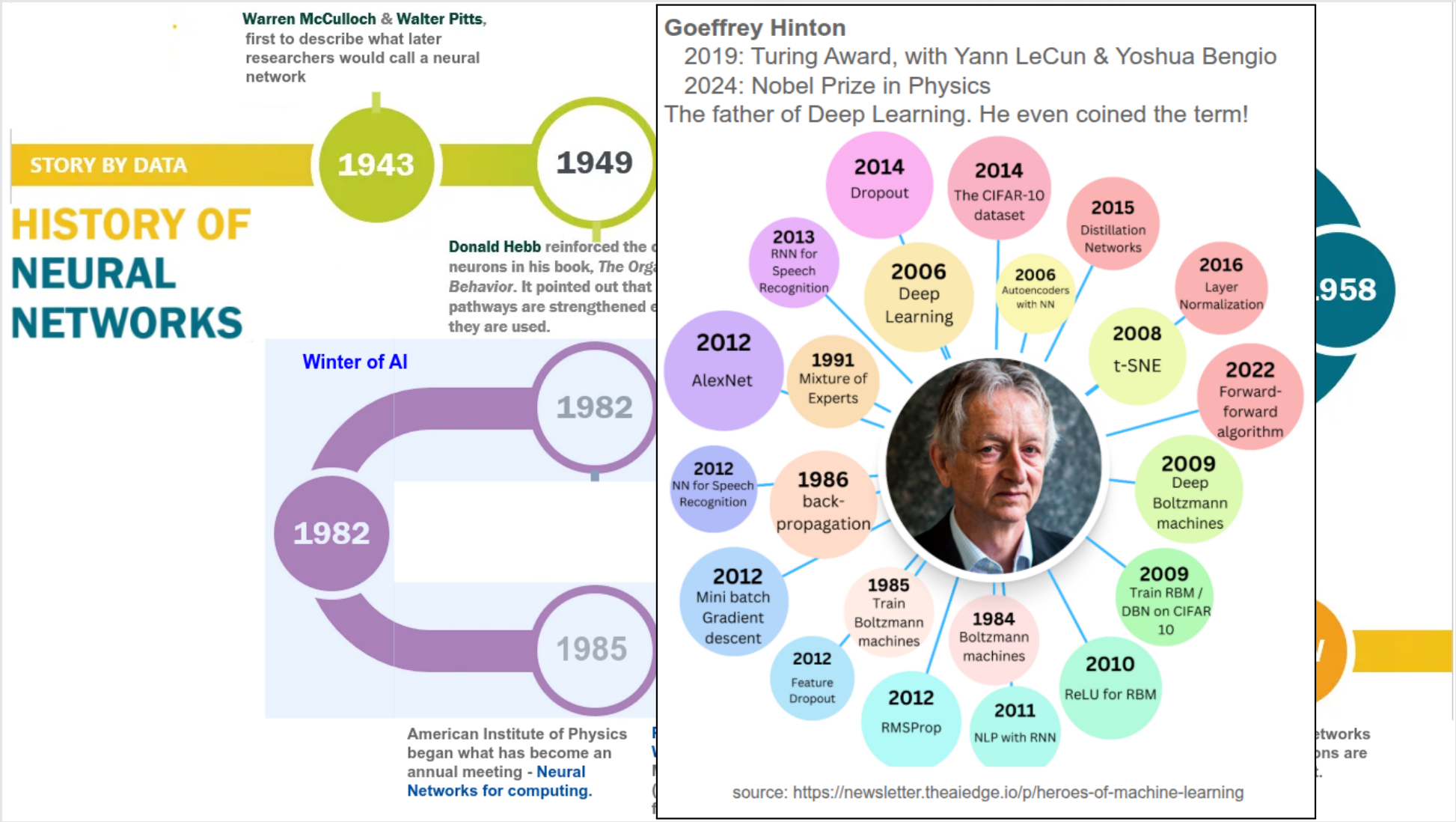
Adapted from Kate Strachni: "Brief History of Neural Netowrks", medium.com

AI : the historical way... from 1950 to 2000s



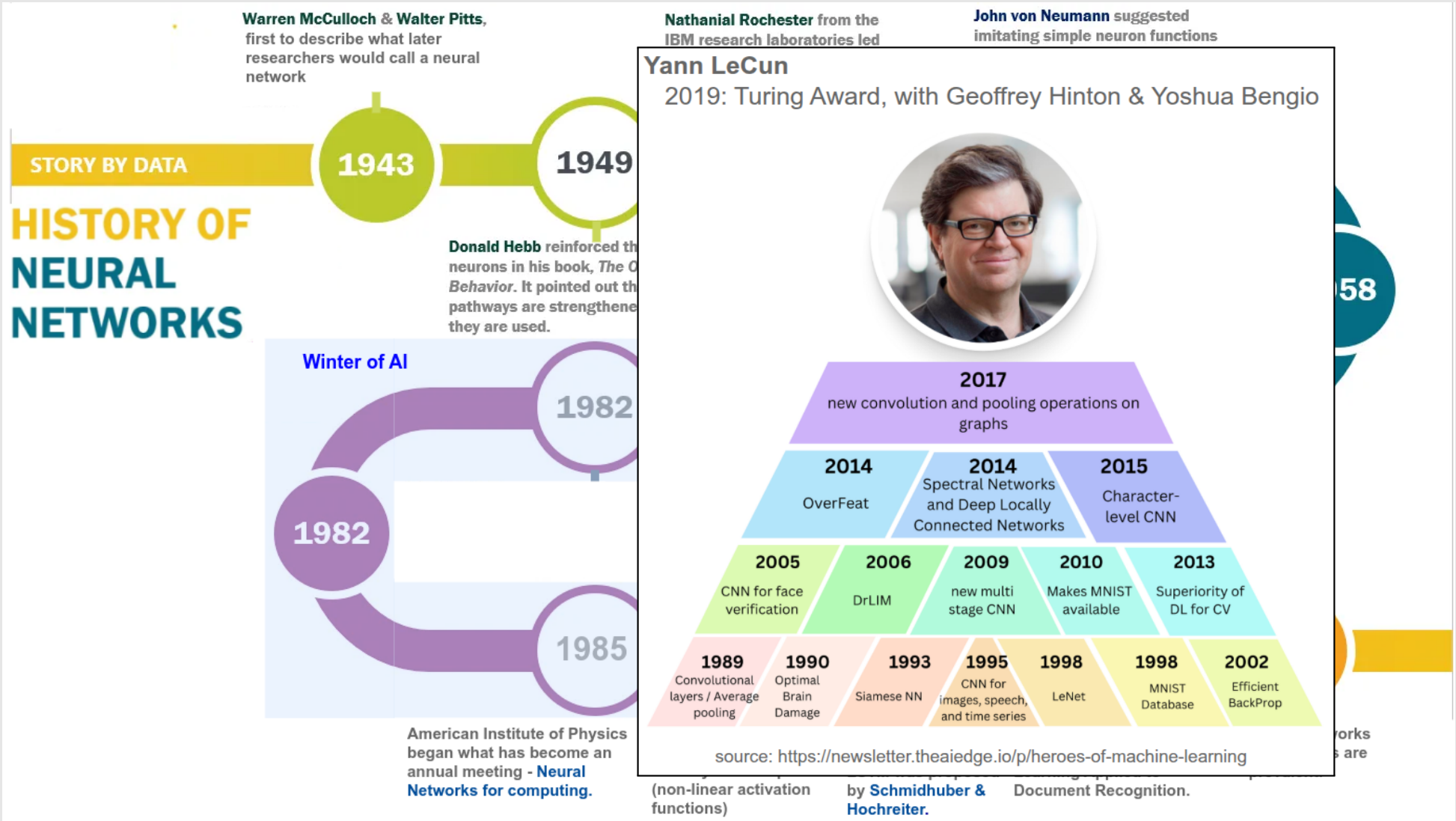
Adapted from Kate Strachni: "Brief History of Neural Netowrks", medium.com

AI : the historical way... from 1950 to 2000s



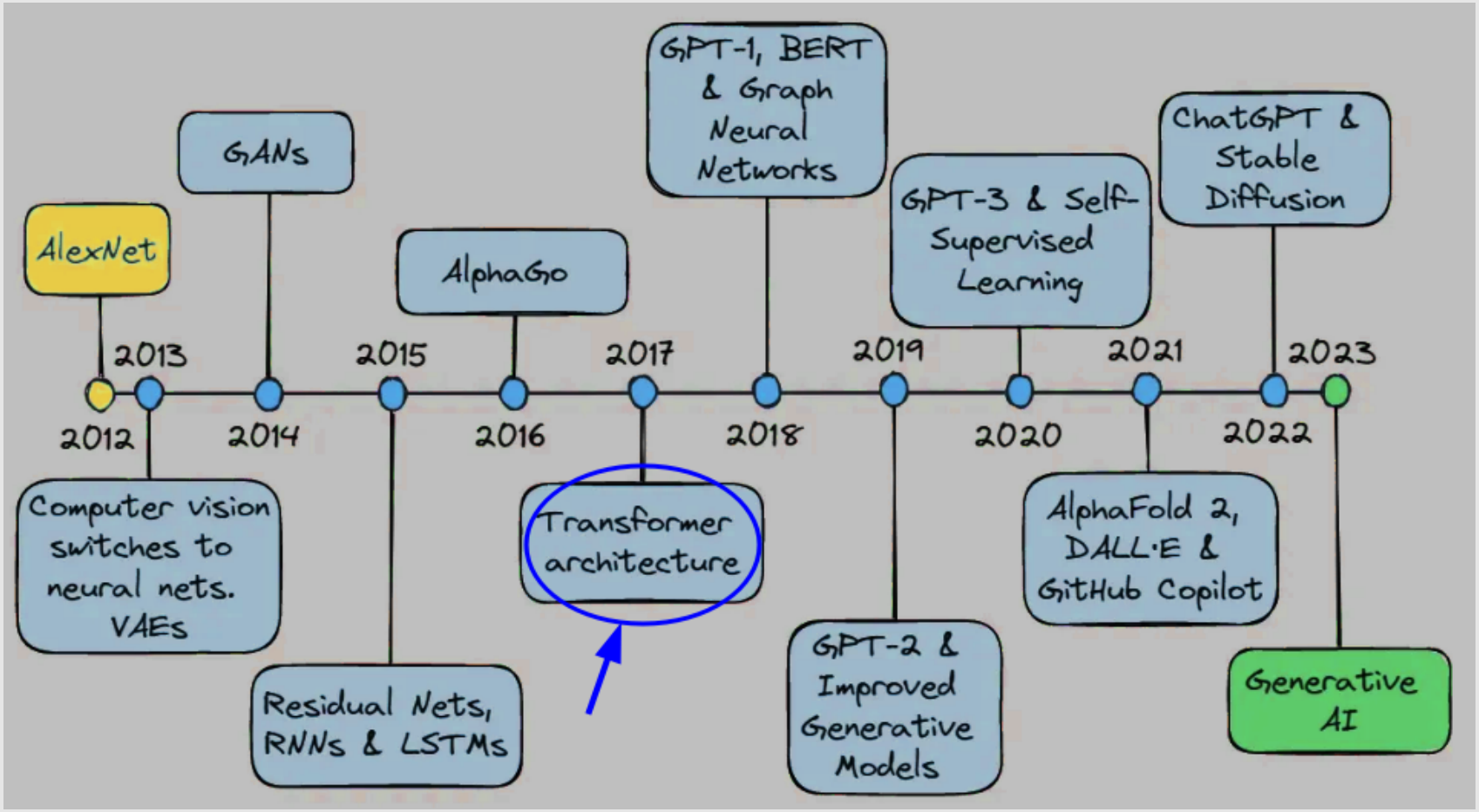
Adapted from Kate Strachni: "Brief History of Neural Netowrks", medium.com

AI : the historical way... from 1950 to 2000s



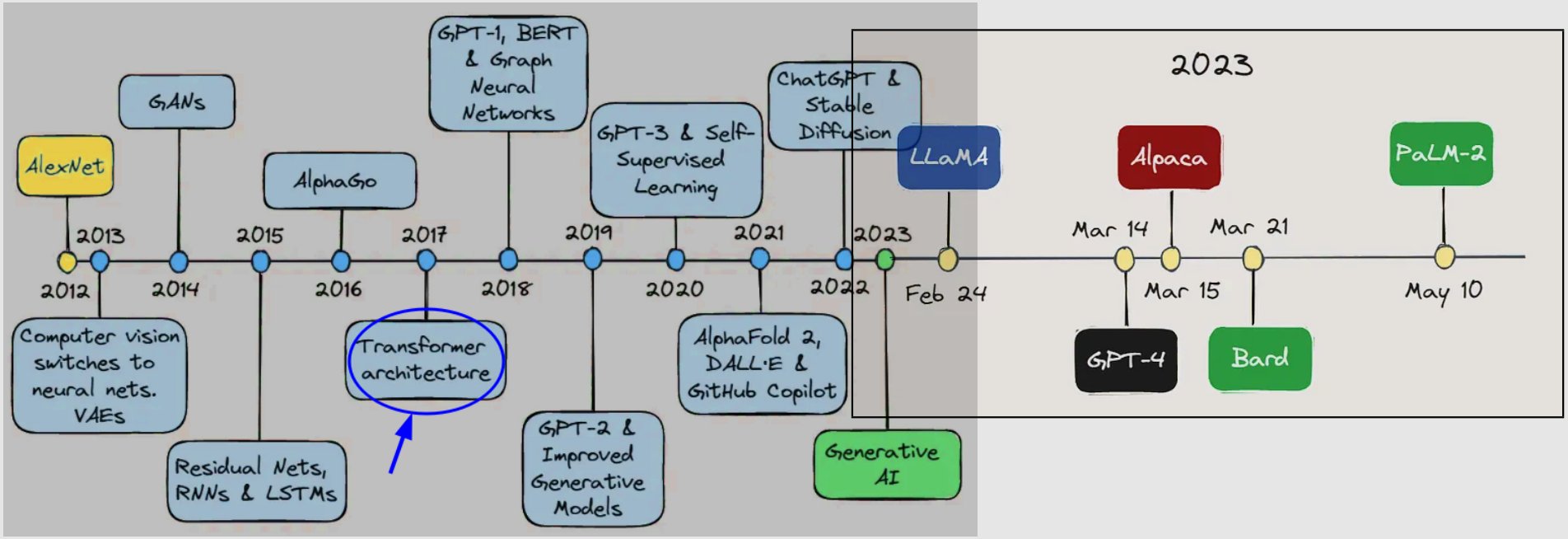
Adapted from Kate Strachni: "Brief History of Neural Netowrks", medium.com

The historical way... post 2012s: the “coming-of-age” of deep learning



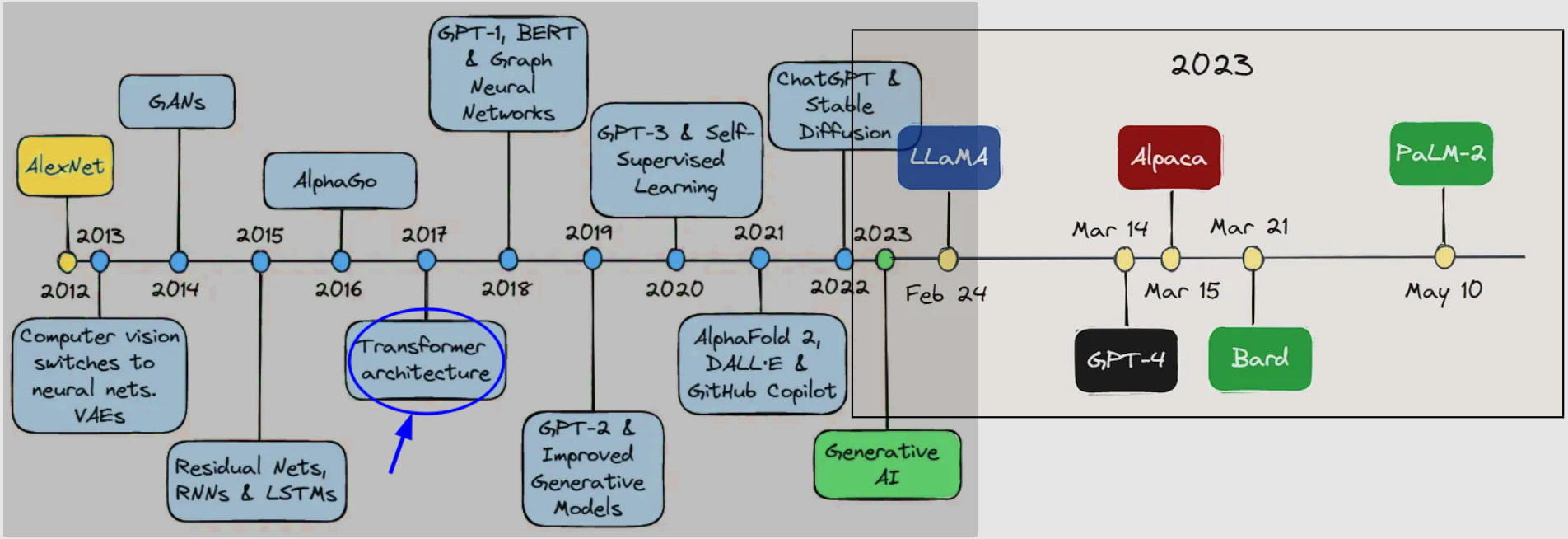
Adapted from Thomas A Dorfe: "Ten Years of AI in Review", kdnugets.com

The historical way... post 2023s: crazy acceleration...



Adapted from Thomas A Dorfe: "Ten Years of AI in Review", medium.com

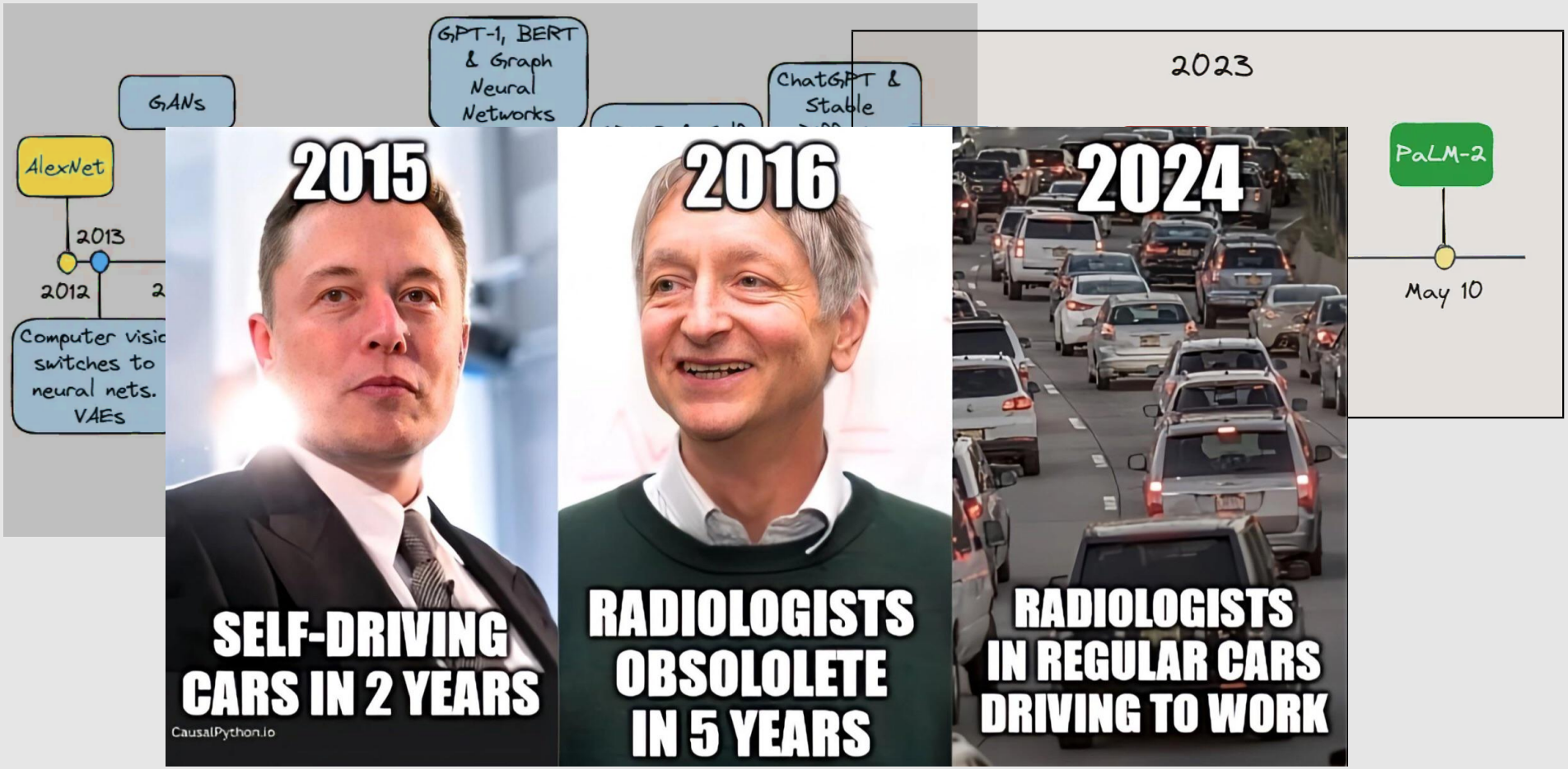
The historical way... post 2023s: crazy acceleration...



Adapted from Thomas A Dorfe: "Ten Years of AI in Review", medium.com

What will happen next?

The historical way... post 2023s: crazy acceleration...



What will happen next?

Artificial Intelligences ?

Narrow/Weak AI

Strong/General AI

Artificial Intelligences ?

Narrow/Weak AI

- AI systems designed for specific, narrow tasks:
  AlphaGo, self-driving cars, robot systems in the medical field...
- Builds systems that **behave like humans** for specific tasks.
- The most common form of AI that we encounter today.
- We already are there... we use it every day!
  anti-spam, facial/voice recognition, language translation...

Strong/General AI

Artificial Intelligences ?

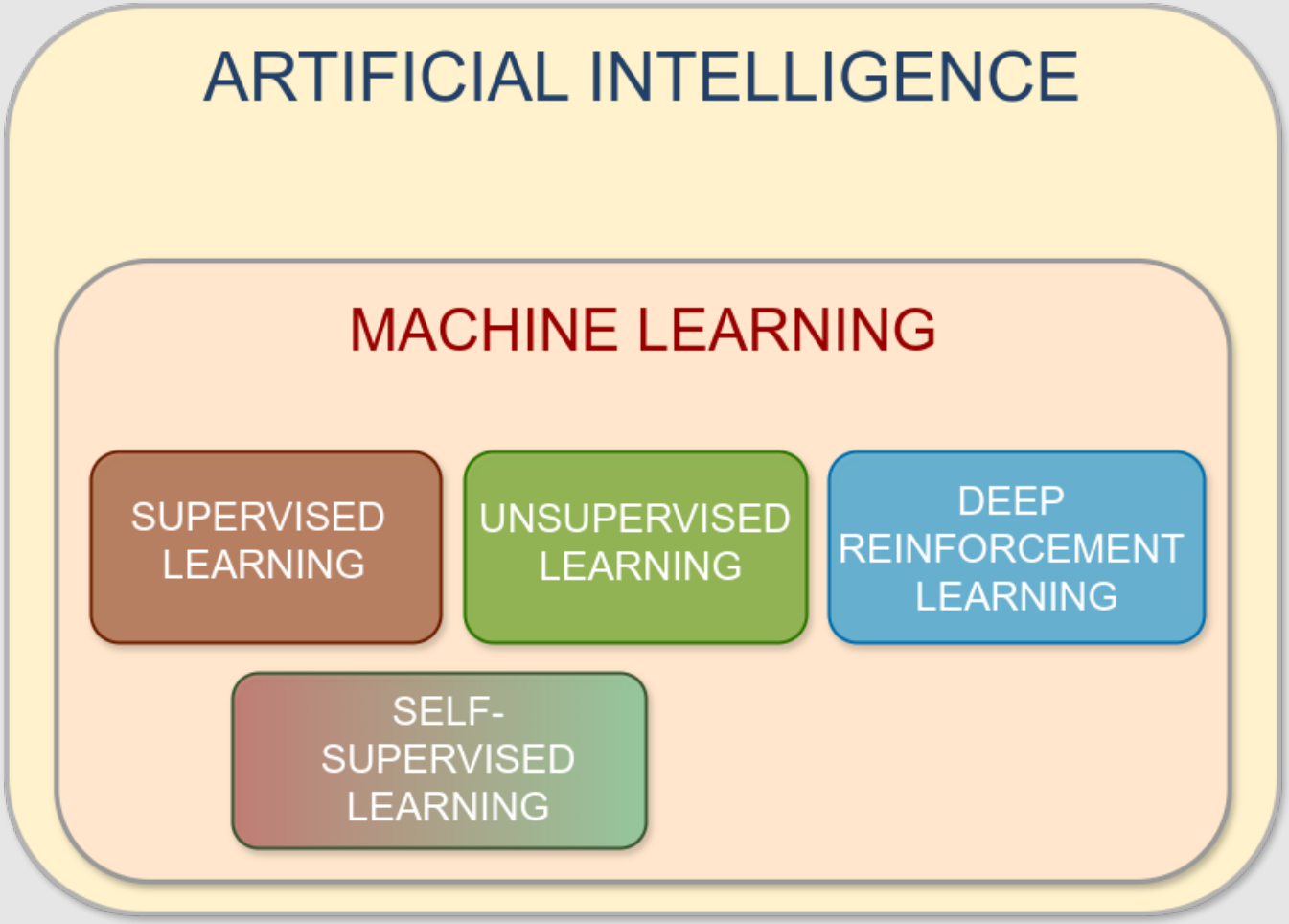
Narrow/Weak AI

- AI systems designed for specific, narrow tasks:
  AlphaGo, self-driving cars, robot systems in the medical field...
- Builds systems that **behave like humans** for specific tasks.
- The most common form of AI that we encounter today.
- We already are there... we use it every day!
  anti-spam, facial/voice recognition, language translation...

Strong/General AI

- Builds systems with the ability to reason in general.
- Could explain how humans think?
- Whe are not yet here... but closer and closer every day
  LLM agent?

Machine Learning: a field of AI



Branches of Machine Learning

Supervised learning

labeled dataset is used to train algorithms:

- **Classification**
  - Images classification
  - Objects detection in images
  - Speech recognition...
- **Regression**
  - Predict a value...
- **Anomaly detection**
  - Spam detection
  - Manufacturing: finding known (learned) defects
  - Weather prediction
  - Diseases classification...

. . .

Branches of Machine Learning

Unsupervised learning

The algorithm attempts to identify patterns / structures within **unlabeled datasets**:

- **Clustering** & **Grouping**
  - Data mining, web data grouping, news grouping...
  - Market segmentation
  - Astronomical data analysis...
- **Anomaly Detection**
  - Manufacturing: finding defects (even new ones)
  - Monitoring abnormal activity: failure, hacker...
  - Fraud detection: fake account on Internet...
- **Dimensionality reduction**
  - Data compressions...

. . .

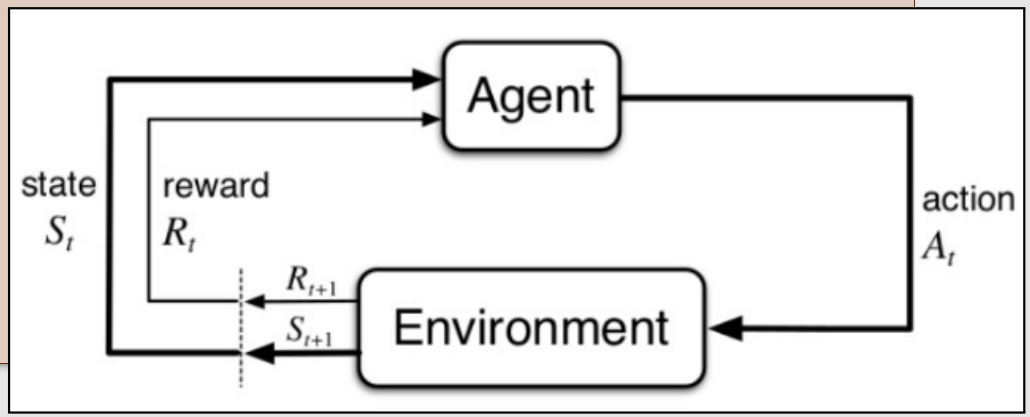
Branches of Machine Learning

Deep Reinforcement Learning DRL

An agent (the model) learns how to drive an environment within a trial and error process by maximising a cumulatd **reward**:

- **Control/command**
  - Controlling drones, mecatronic systems, electric VTVL rocket, robots
  - Factory optimization
  - Financial (stock) trading...
- **Decision making**
  - games (video games)
  - financial analysis...

. . .



Various approaches for ML algorithms

Supervised learning:

- Neural Networks
- Bayesian inference
- Random forest
- Decision Tree
- Support Vector Machine (SVM)
- K-Nearest Neighbor (KNN)
- Linear regression
- Logistic regression...

Unsupervised learning:

- Neural Networks
- Principal Composant Analysis (PCA)
- Singular Value Decomposition (SVD)
- K-mean & Prob. clustering...

Reinforcement learning:

- Neural Networks (Q-learning, Actor-Critic, DDPG, PPO...)
- Monte Carlo...

Self-Supervised learning:

- Neural Networks (SimCLR, Dino...)

Various approaches for ML algorithms

Supervised learning:

- Neural Networks
- Bayesian inference
- Random forest
- Decision Tree
- Support Vector Machine (SVM)
- K-Nearest Neighbor (KNN)
- Linear regression
- Logistic regression...

Unsupervised learning:

- Neural Networks
- Principal Composant Analysis (PCA)
- Singular Value Decomposition (SVD)
- K-mean & Prob. clustering...

Reinforcement learning:

- Neural Networks
- Monte Carlo...

Self-Supervised learning:

- Neural Networks

A large variety of possibities $\rightsquigarrow$ success of **Deep Learning** and **Neural Networks**

Fields & applications of ML

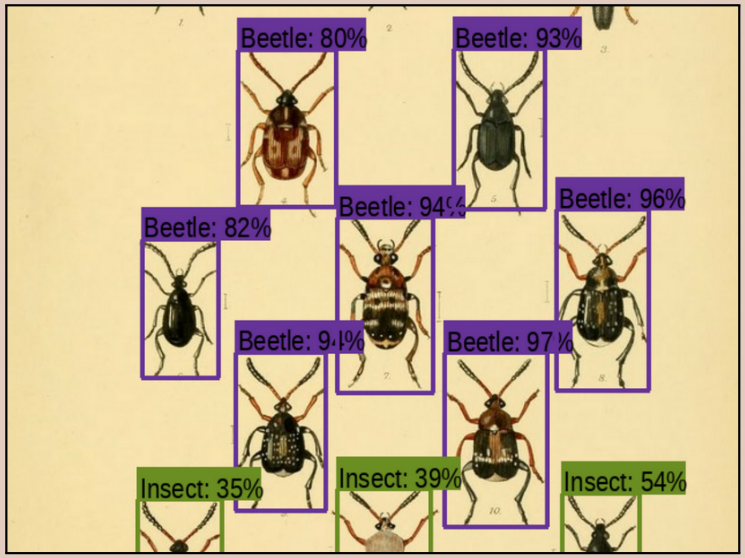
Computer Vision

- Image Classification.
- Object Detection .
- (Semantic) Segmentation.
- Pose Estimation.
- Style transfer.
- OCR (Optical Character Recognition).
- Image Generation
- ...

Fields & applications of ML

Computer Vision

- Image Classification.
- Object Detection .
- (Semantic) Segmentation.
- Pose Estimation.
- Style transfer.
- OCR (Optical Character Recognition).
- Image Generation
- ...



source : Tensorflow tutorials object_detection

Fields & applications of ML

Computer Vision

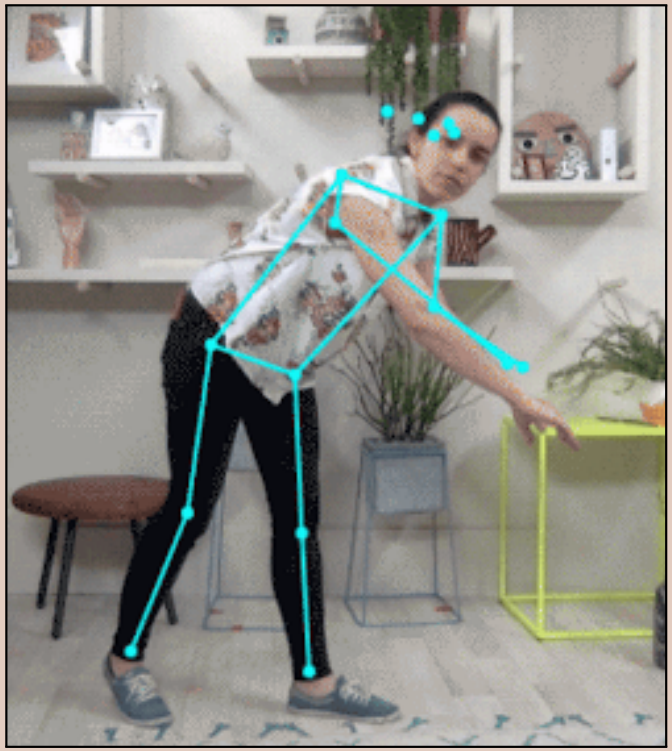
- Image Classification.
- Object Detection .
- (Semantic) Segmentation.
- Pos
- Sty
- OC
- Ima
- ...

source : Tensorflow

Fields & applications of ML

Computer Vision

- Image Classification.
- Object Detection .
- (Semantic) Segmentation.
- Pose Estimation.
- Style transfer.
- OCR (Optical Character Recognition).
- Image Generation
- ...

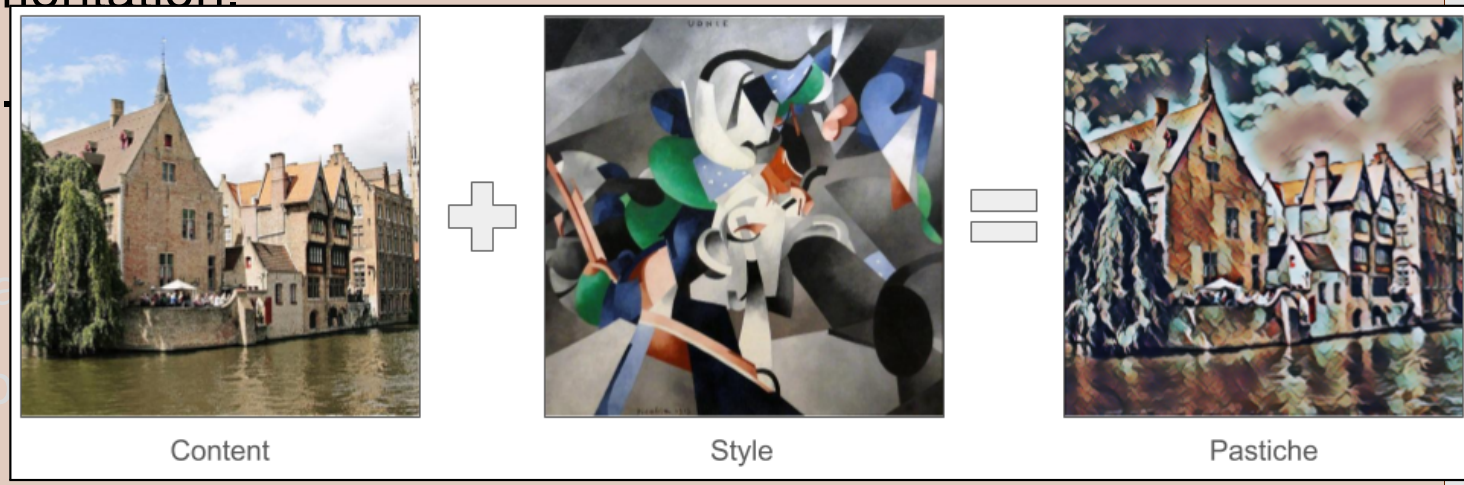


source : Tensorflow pose_detection

Fields & applications of ML

Computer Vision

- Image Classification.
- Object Detection .
- (Semantic) Segmentation.
- Pose Estimation.
- Style transfer.
- OCR (Optical Cha
- Image Generatio
- ...



source : Tensorflow style_transfer

Fields & applications of ML

Computer Vision

- Image Classification.
- Object Detection .
- (Semantic) Segmentation.
- Pose Estimation.
- Style transfer.
- OCR (Optical Character Recognition).
- Image Generation
- ...

source : pyimagesearch.com

Fields & applications of ML

Computer Vision

- Image Classification.
- Object Detection .
- (Semantic) Segmentation.
- Pose Estimation.
- Style transfer.
- OCR (Optical Character Recognition).
- Image Generation
- ...

source : stylegan

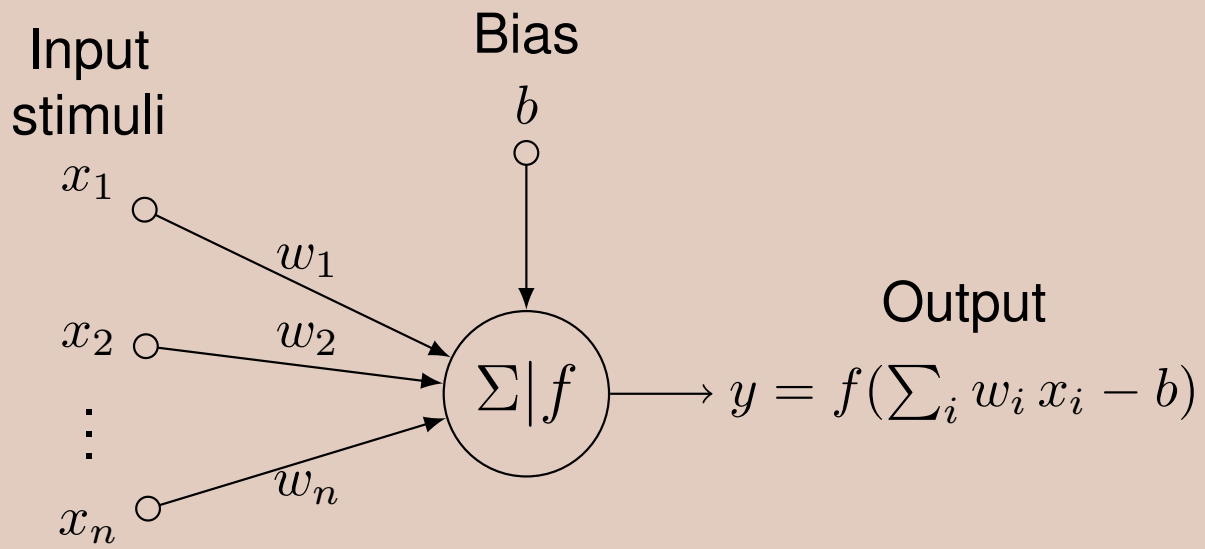
Fields & applications of ML

NLP(Natural Language Processing): some applications...

- Natural Language Understanding (NLU)
- Natural Language Generation (NLG)
- Speech recognition / Speech Synthesis (Text To Speech)
- Machine Translation (languages)
- Virtual agents and ChatBots
- LLM ChatBots (Agent Conversationnel)
- Optical character recognition (OCR)
- ...

Computing aspects

The Artificial neuron model

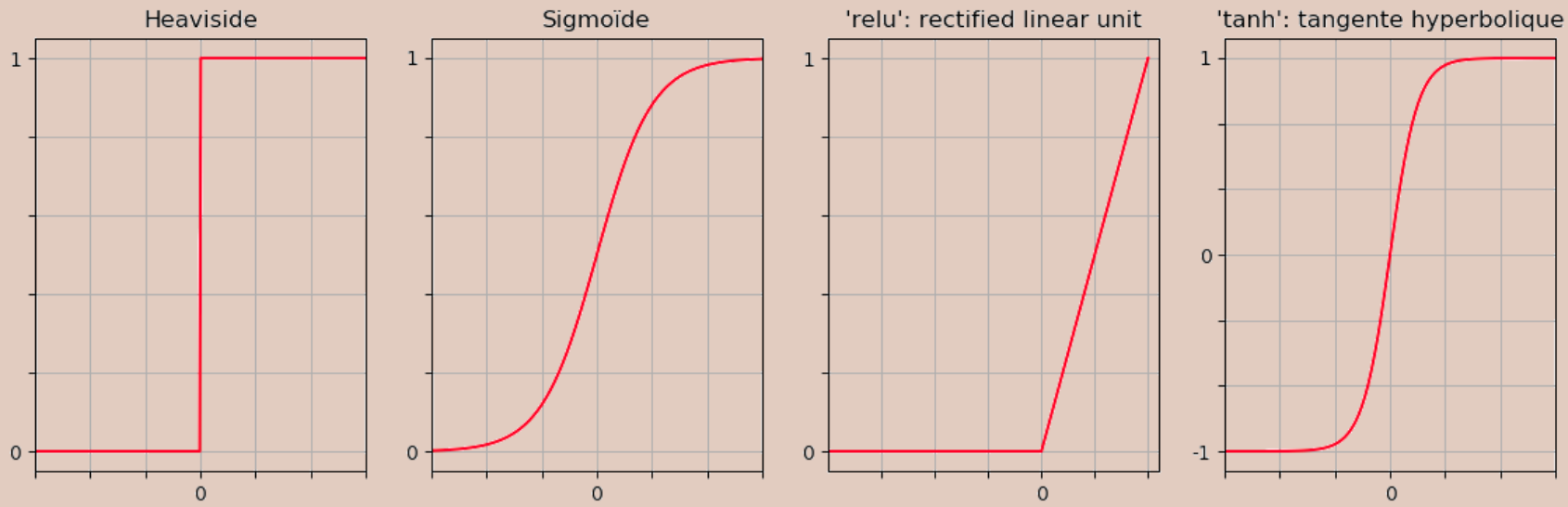


An **artificial neuron**:

- receives the input stimuli $(x_i)_{i=1..n}$ with **weights** (w_i)
- computes the **weighted sum** of the input $\sum_i w_i\, x_i - b$
- outputs its activation $f(\sum_i w_i\, x_i - b)$, computed with a non-linear **activation function** f .

Computing aspects

Common activation functions

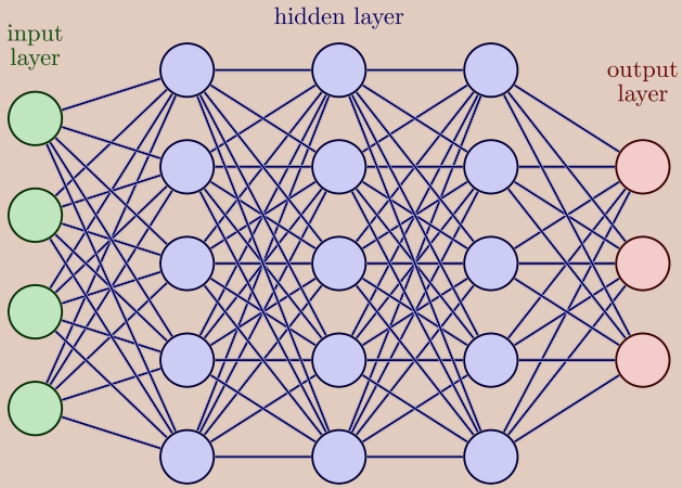


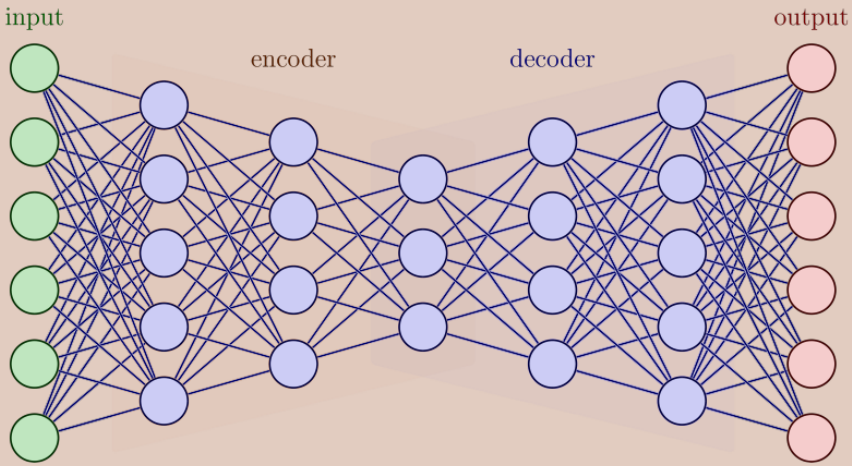
- Introduces a non-linear behavior.
- Sets the range of the neuron output: $[-1, 1]$, $[0, 1]$, $[0, \infty[$...
- The bias b sets the activation threshold of the neuron.

Computing aspects

Neural network

Neurons are grouped by layers to form a **Neural Network**



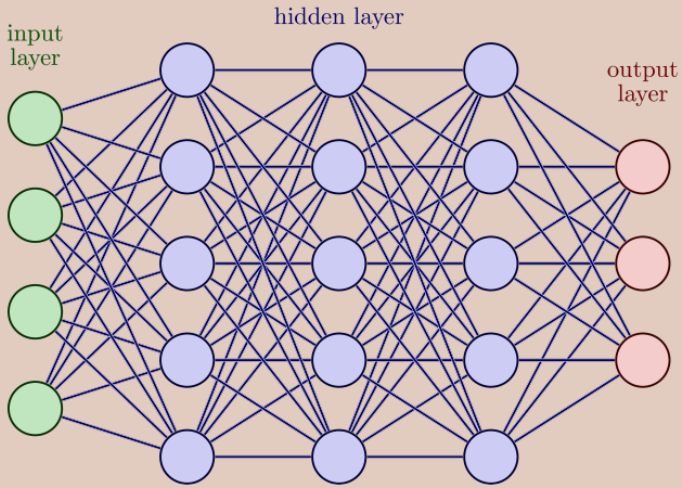


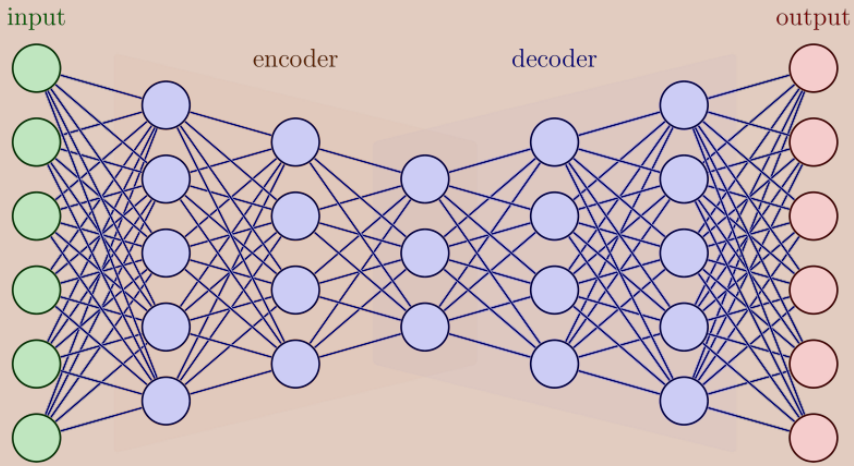
- Two common architectures:
  - The **Dense Neural Network** (DNN), simple, generalist, can perfom greatly when well tuned.
  - The more complex **Convolutional Neural Network** (CNN) specialized in image processing.

Computing aspects

Neural network

Neurons are grouped by layers to form a **Neural Network**





- Two common architectures:
  - The **Dense Neural Network** (DNN), simple, generalist, can perfom greatly when well tuned.
  - The more complex **Convolutional Neural Network** (CNN) specialized in image processing.

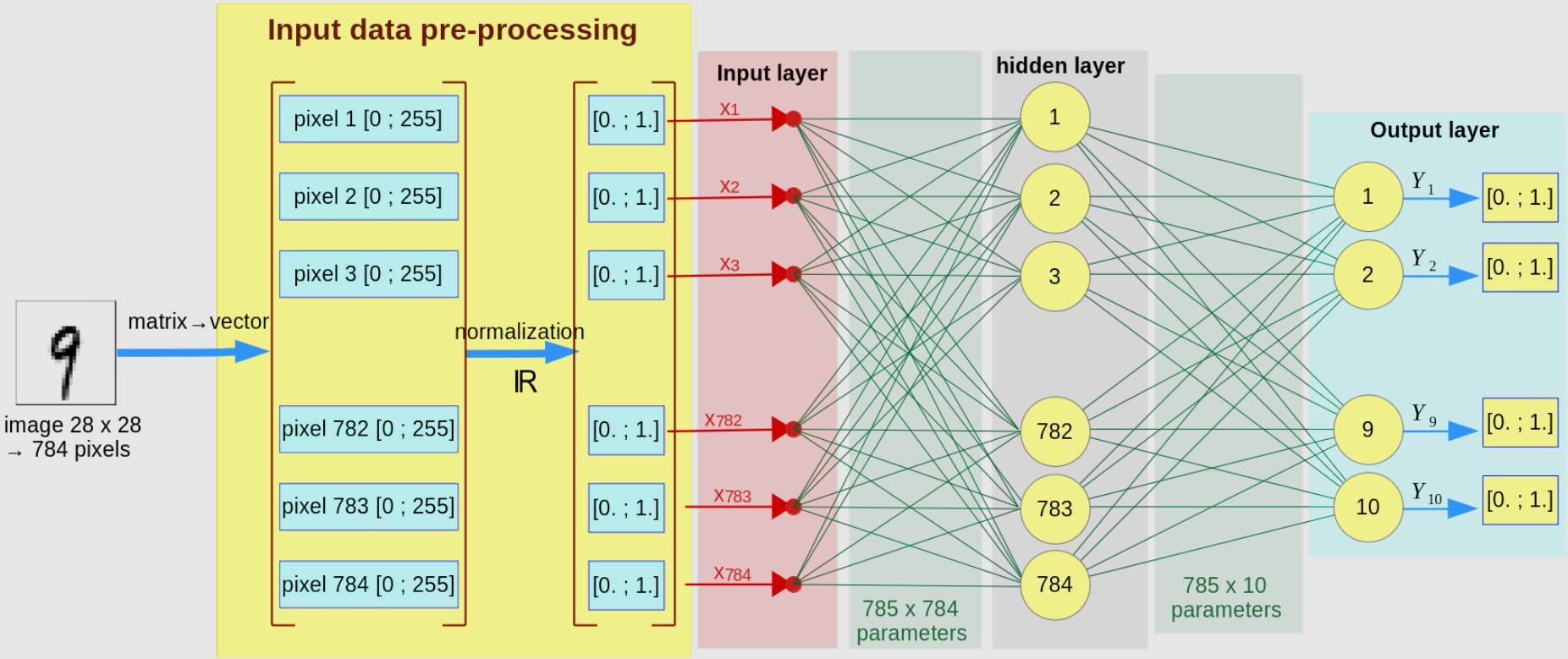
Well known ML example: Trainig a NN to classify the MNIST handwritten digit images

- MNIST: bank of 70000 **labeled images**
  (60000 training images and 10000 test images)

- Grayscale images 28×28 pixels.
- Scores with a dense networks can reach 98% success...
- State of the art for image recognition : **Convolutional Neural Networks** (CNN)
  [will not be covered in this course limited to dense neural networks]

Dense Neural Network architecture

Each matrix $28 \times 28 \rightsquigarrow$ normalized vector of 784 components `float` $\in [0; 1]$.



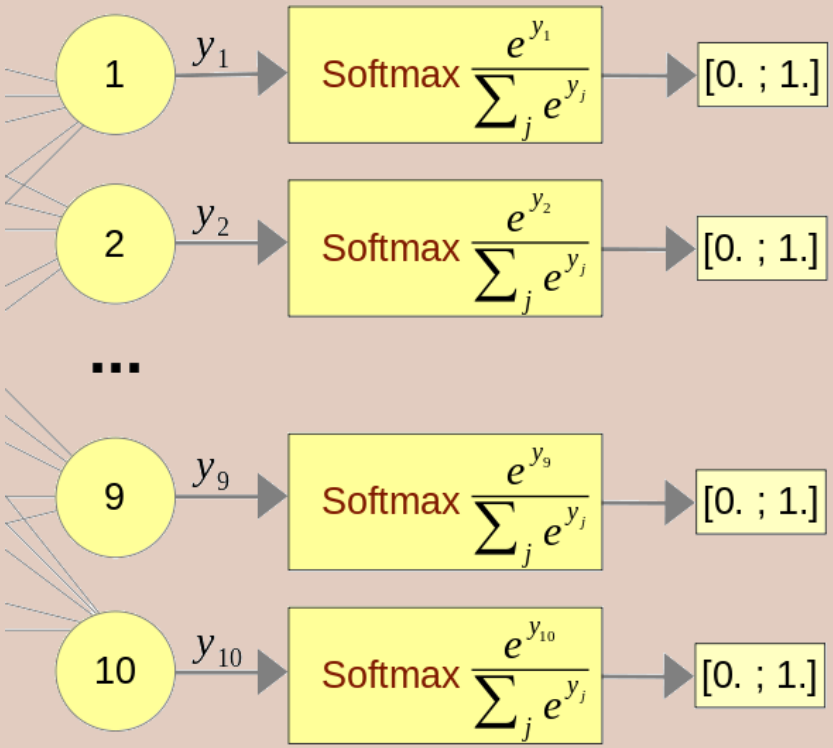
Structure of the neural network:

- The *Input layer* sets the size of the NN inputs to 784 values $\rightsquigarrow$ it has no neuron !
- A *Hidden layer* of 784 neurons (we could have more, or less...), receives the input data. It is connected to the next layer.
- An *Output layer* of 10 neurons (1 neuron for each digit to be recognized).

The *Softmax* activation function for classification

- intermediate layers $\rightsquigarrow$ **relu** promotes NN learning [1].
- **softmax** $\rightsquigarrow$ always used for in the last layer for **Classifying**.

Softmax for the last layer, 10 classes



- The activation of neuron k is $Y_k = e^{y_k} / \sum_i e^{y_i}$ with $y_k = \sum_i \omega_i x_i - b$ calculated by the neuron k.
- The neurons outputs are interpreted as **probabilities** in the interval [0,1].

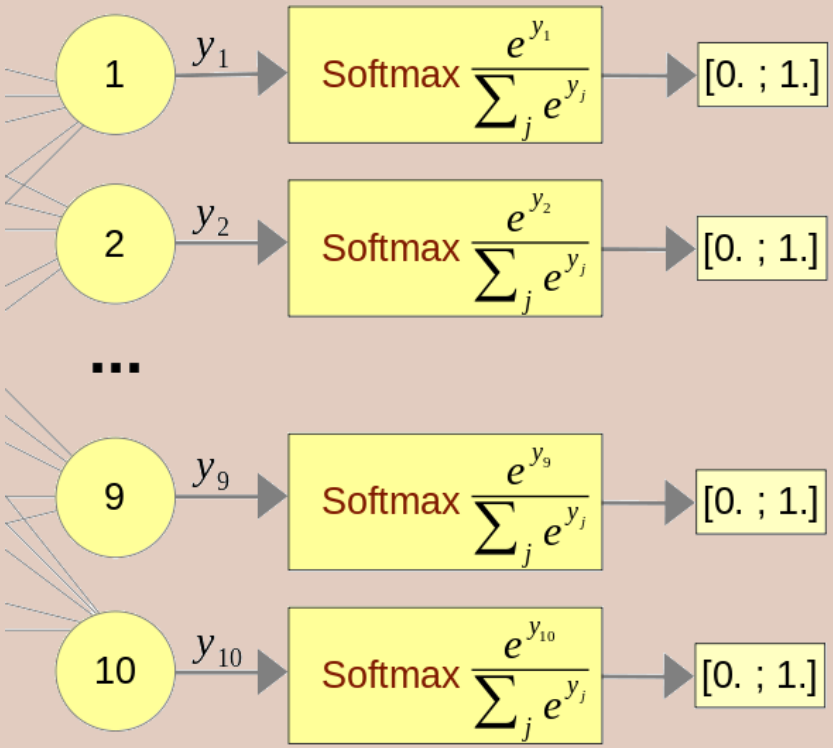
$\rightsquigarrow$ the neuron with the highest probability gives the NN response.

[1] avoids the *vanishing gradient* that appears in *back propagation*

The *Softmax* activation function for classification

- intermediate layers $\rightsquigarrow$ **relu** promotes NN learning [1].
- **softmax** $\rightsquigarrow$ always used for in the last layer for **Classifying**.

Softmax for the last layer, 10 classes



- The activation of neuron k is $Y_k = e^{y_k} / \sum_i e^{y_i}$ with $y_k = \sum_i \omega_i x_i - b$ calculated by the neuron k.
- The neurons outputs are interpreted as **probabilities** in the interval [0,1].

$\rightsquigarrow$ the neuron with the highest probability gives the NN response.

[1] avoids the *vanishing gradient* that appears in *back propagation*

The *One-hot* Coding

One-hot encoding of labels

Purpose: to put the image labels in the format of the NN output

- Image labels: **integers** from 0 to 9.
- Network output: **vector of 10** `float` numbers in the interval [0;1] as calculated by the 10 *softmax* functions of the 10 output neurons.
- *one-hot* coding of an ordered collection of N unique elements:
  - each element is coded by a vector of N null components except one,
  - the *i th* element of the collection $\rightsquigarrow$ vector with a 1 for its *i th* component.

The *One-hot* Coding

One-hot encoding of labels

Purpose: to put the image labels in the format of the NN output

- Image labels: **integers** from 0 to 9.
- Network output: **vector of 10** `float` numbers in the interval [0;1] as calculated by the 10 *softmax* functions of the 10 output neurons.
- *one-hot* coding of an ordered collection of N unique elements:
  - each element is coded by a vector of N null components except one,
  - the *i th* element of the collection $\rightsquigarrow$ vector with a 1 for its *i th* component.

The *One-hot* Coding

One-hot encoding of labels

Purpose: to put the image labels in the format of the NN output

- Image labels: **integers** from 0 to 9.
- Network output: **vector of 10** `float` numbers in the interval [0;1] as calculated by the 10 *softmax* functions of the 10 output neurons.
- *one-hot* coding of an ordered collection of N unique elements:
  - each element is coded by a vector of N null components except one,
  - the *i th* element of the collection $\rightsquigarrow$ vector with a 1 for its *i th* component.

Digit	One_hot vector
0	[1 0 0 0 0 0 0 0 0 0]
1	[0 1 0 0 0 0 0 0 0 0]
2	[0 0 1 0 0 0 0 0 0 0]
3	[0 0 0 1 0 0 0 0 0 0]
4	[0 0 0 0 1 0 0 0 0 0]
5	[0 0 0 0 0 1 0 0 0 0]
6	[0 0 0 0 0 0 1 0 0 0]
7	[0 0 0 0 0 0 0 1 0 0]
8	[0 0 0 0 0 0 0 0 1 0]
9	[0 0 0 0 0 0 0 0 0 1]

The *one-hot* encoding of labels '0' to '9' results in a 10-component vector, like the one computed by the neural network.

Compute the NN error

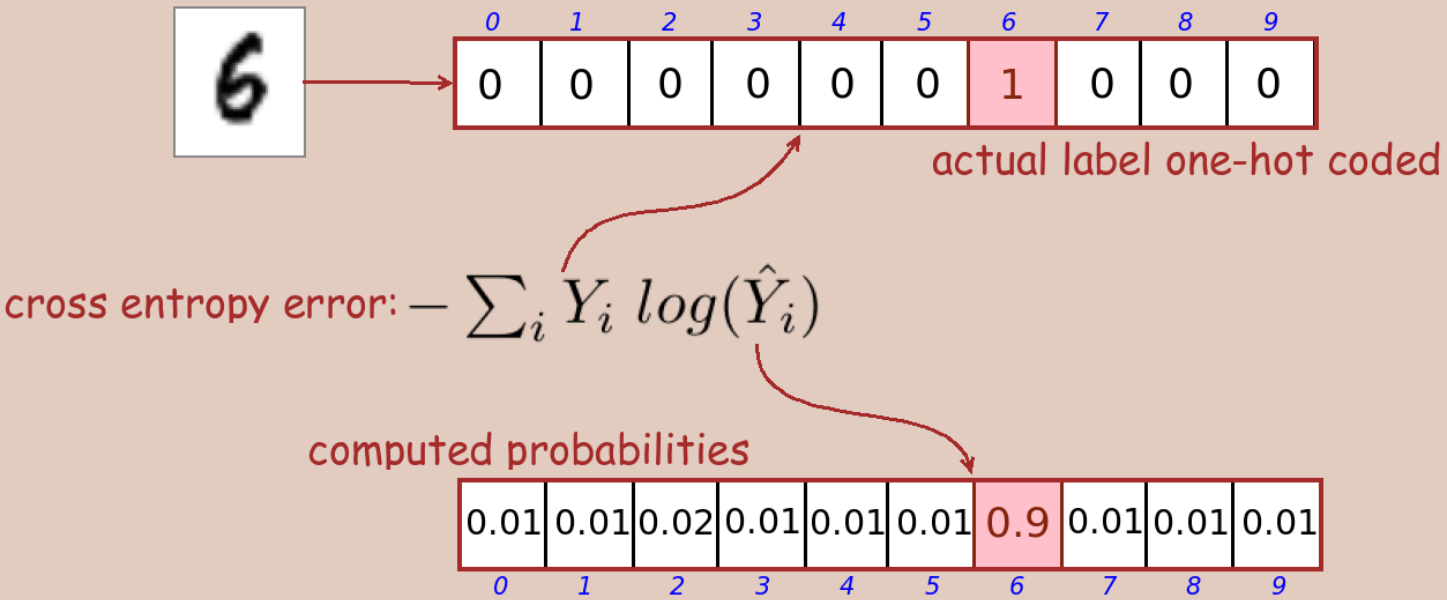
Error function: *Cross entropy error*

- An image processed by the NN $\rightsquigarrow$ vector $\hat{Y}$ of 10 `float` numbers to compare to the *hot-one* encoding Y of the label of the image.
- Error (loss) function $\rightsquigarrow$ *cross entropy error*: $e(Y, \hat{Y}) = -\sum_i Y_i \, log(\hat{Y}_i)$ adapted to the *one-hot* coding

Compute the NN error

Error function: *Cross entropy error*

- An image processed by the NN $\rightsquigarrow$ vector $\hat{Y}$ of 10 `float` numbers to compare to the *hot-one* encoding Y of the label of the image.
- Error (loss) function $\rightsquigarrow$ *cross entropy error*: $e(Y, \hat{Y}) = -\sum_i Y_i \; log(\hat{Y}_i)$ adapted to the *one-hot* coding



The *Optimization* stage

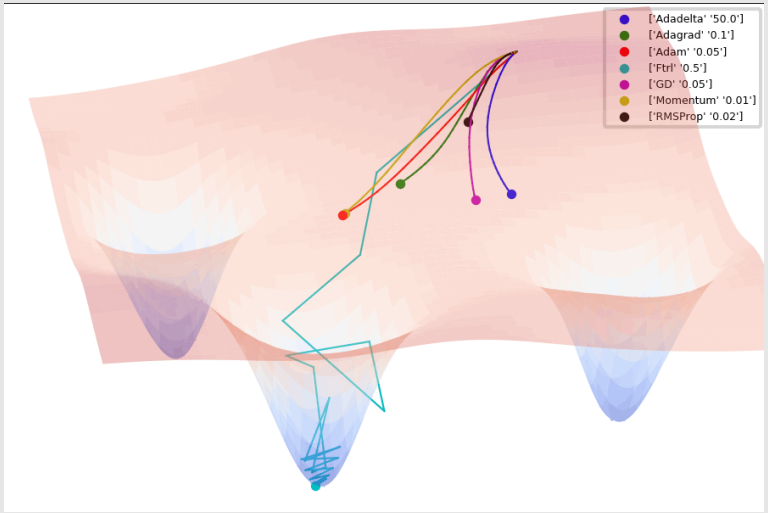
Optimization and *Back Propagation*

- **Feed forward stage** : an optimization algorithm calculates the gradient of the loss function relative to network weights.
- **Back Propagation** : the BP algorithm **modifies** the weights of the network thanks to the gradient of the loss function, iterating from the last layer to the first layer.
- Examples of optimization algorithm used:
  - *Gradient Descent* (GD)
  - *Stochastic Gradient Descent* (SGD)
  - *Adam* (enhanced version of gradient descent)...

  The module tf.keras.optimizers offers Python implementation of several optimization algorithms.

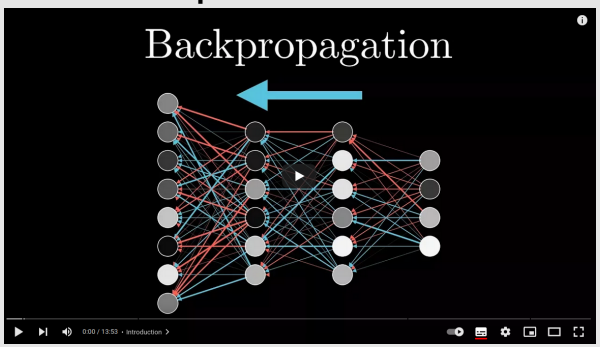
The *Optimization* stage

Visualization of gradient descent algorithm iterations for an ultra-simple loss function with only 2 variables:

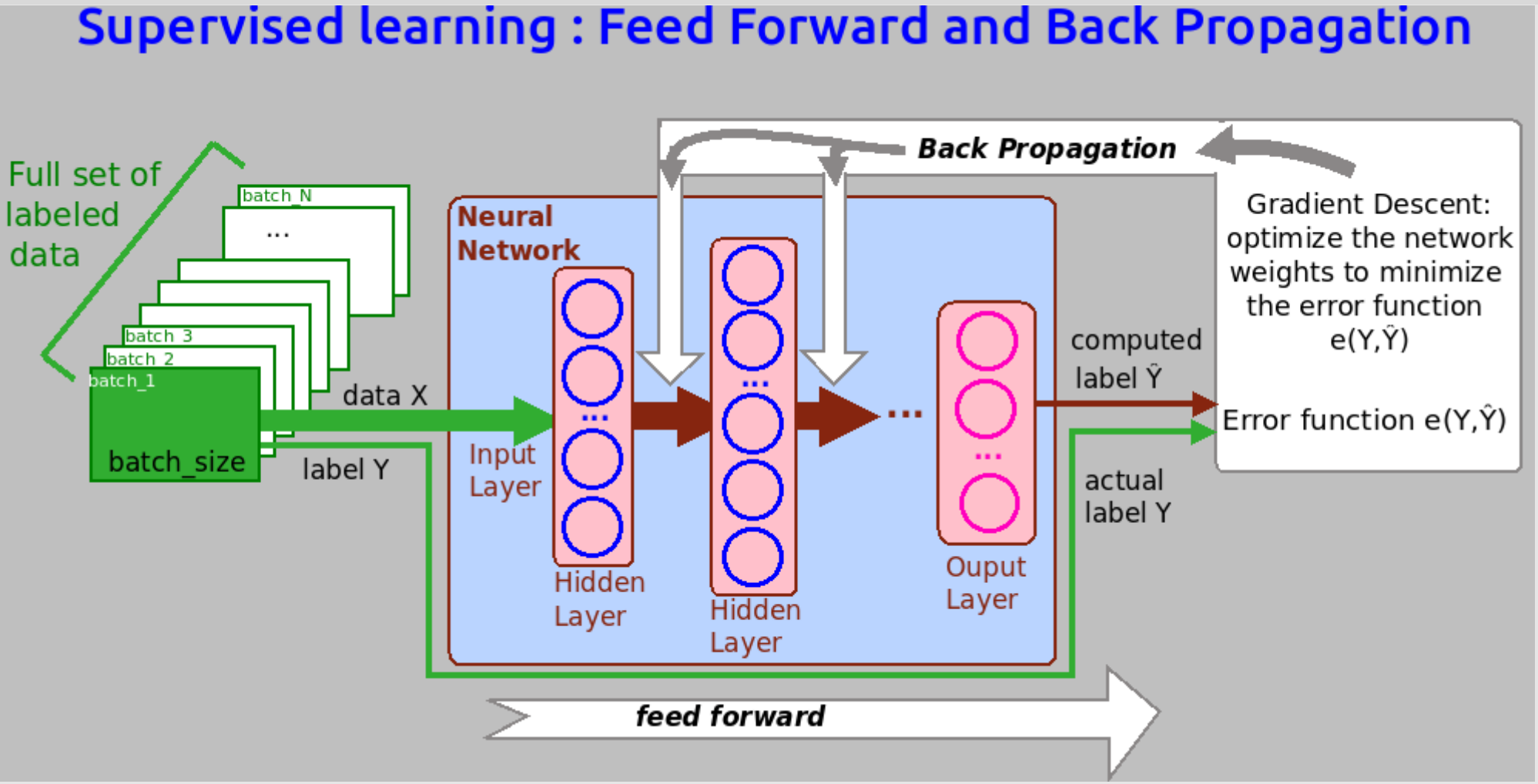


(source: github.com/Jaewan-Yun/optimizer-visualization)

back propagation algorithm explanation video:

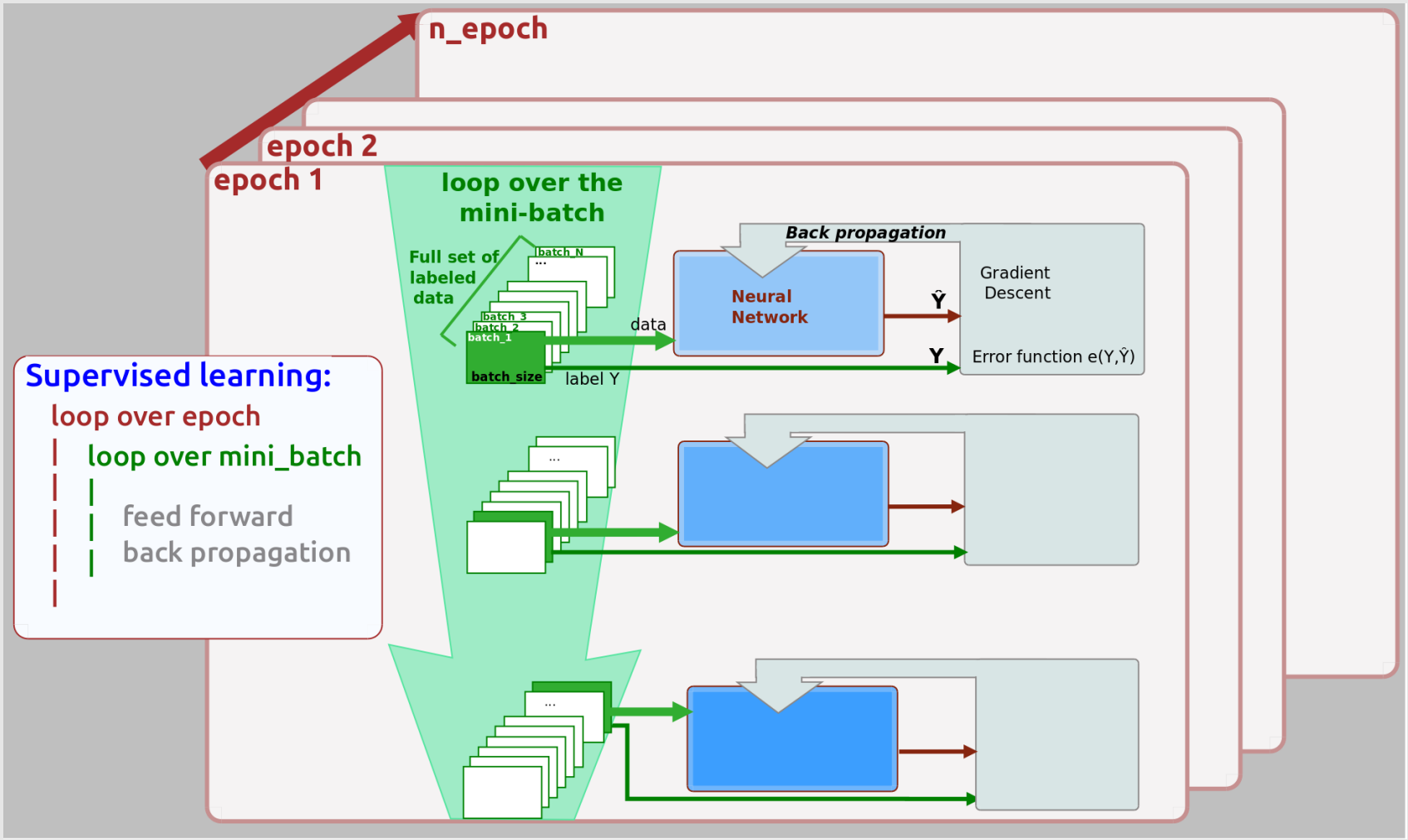


Supervised learning strategy



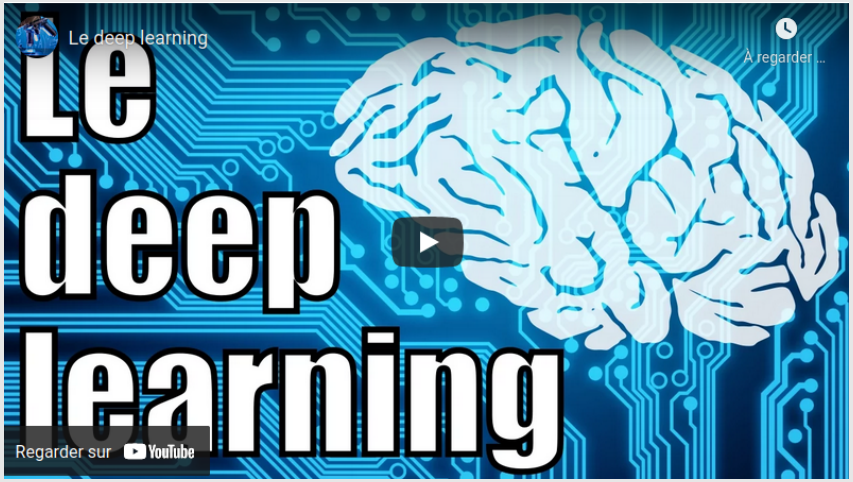
- The dataset is split into (mini) **batches** of size `batch_size`
- After each *feed forward* the *Back Propagation* algorithm modifies the weights neurons to minimize the error e.

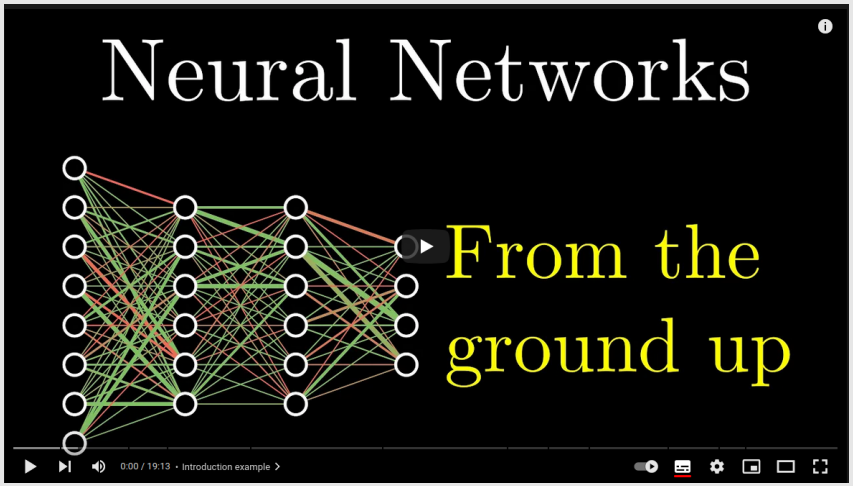
Supervised learning strategy

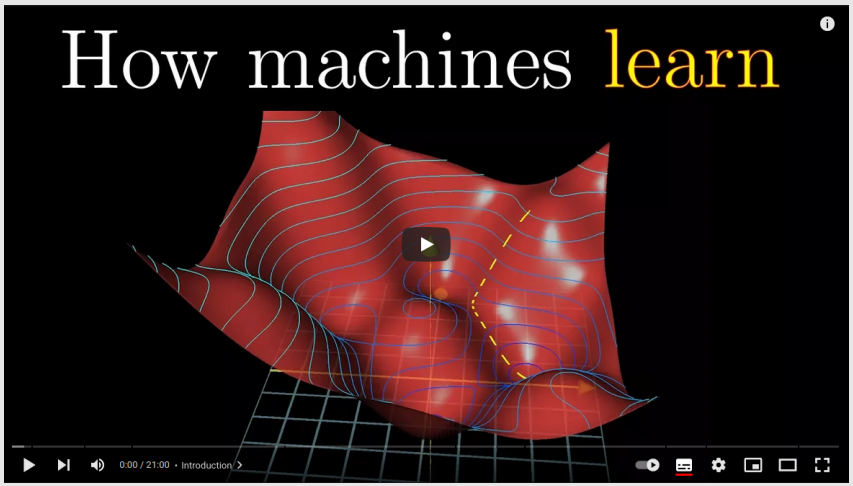


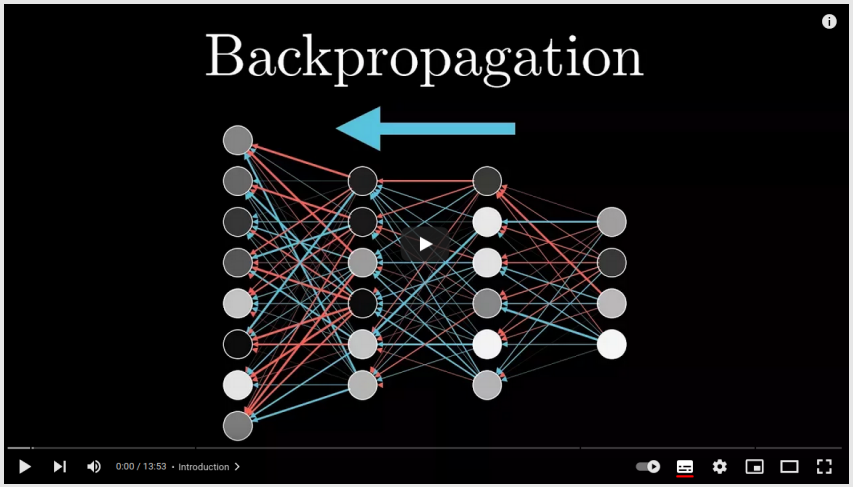
- Training with the whole dataset is repeated `n_epoch` times,
- The network state at end of epoch `n` $\rightsquigarrow$ initial state for epoch `n+1`.

Videos


Le deep learning
Le deep learning
À regarder ...
Regarder sur YouTube


Neural Networks
From the ground up
0:00 / 19:13 • Introduction example


How machines learn
0:00 / 21:00 • Introduction


Backpropagation
0:00 / 13:53 • Introduction

References

[1] *Artificial Intelligence: A Modern Approach (4th Edition)*, By Stuart Russell & Peter Norvig. Pearson, 2020. ISBN 978-0134610993. aima.cs.berkeley.edu

Intelligence artificielle – Une approche moderne – 4e éd., By Stuart Russell & Peter Norvig. Translated by L. Miclet, F. Popineau, & C. Cadet. Paris: Pearson Education France, 2021. ISBN 978-2326002210.

[2] *Deep Learning With Python*, François Chollet, download PDF dowload PDF

[3] *Deep Learning.*, Goodfellow, Ian; Bengio, Yoshua; Courville, Aaron (2016), MIT Pres, ISBN 9780262035613