Intro to Deep Learning

Sungwook E. Hong (KASI)

2025. 8. 19 @ A3Net Summer School

What is Learning?

Supervised

From given sets of $\{\vec{x}_i\}$ and $\{\vec{y}_i\}$,
find a function $F(\ldots)$ that satisfies

$$F(\overrightarrow{x_i}) \approx \overrightarrow{y_i}$$

Outputs that we need to reproduce

Inputs

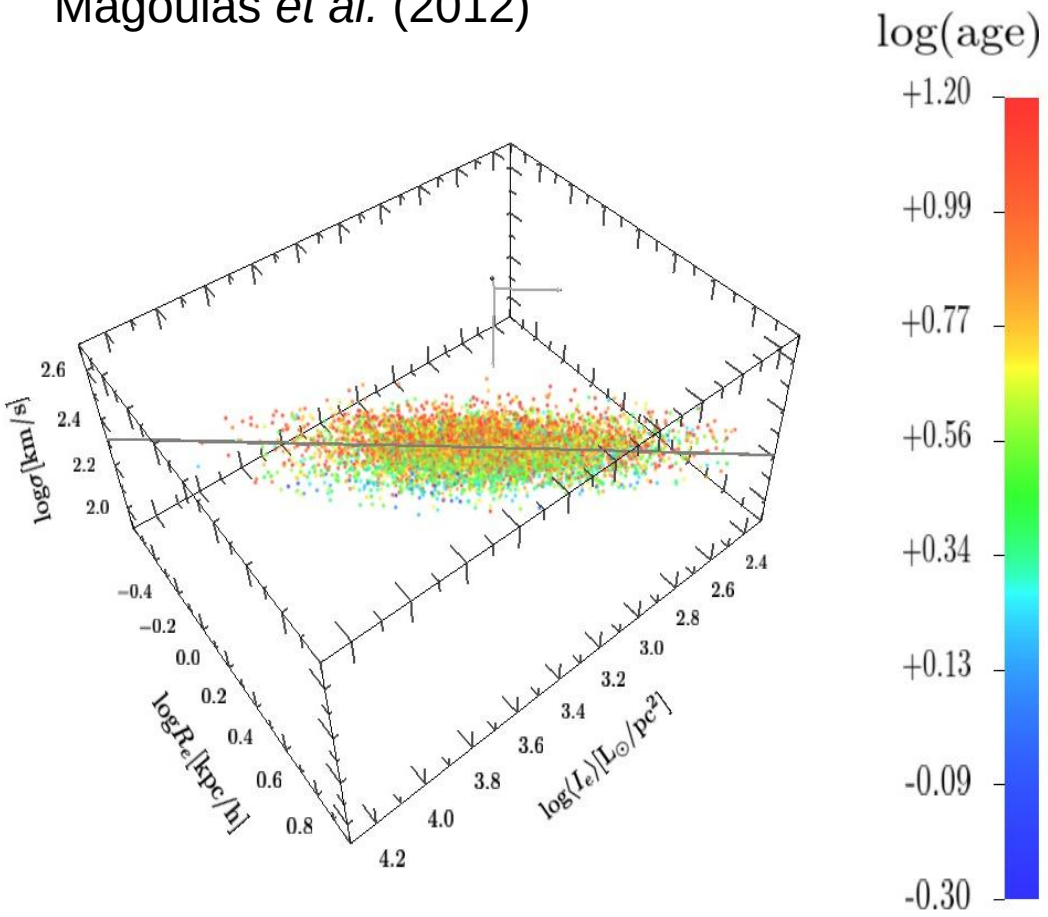
Unsupervised

From a given set of $\{\vec{x}_i\}$,
find a function $F(\ldots)$ that produces a useful $\{\vec{y}_i\}$ by

$$F(\overrightarrow{x_i}) = \overrightarrow{y_i}$$

Outputs that we can define

Magoulas *et al.* (2012)



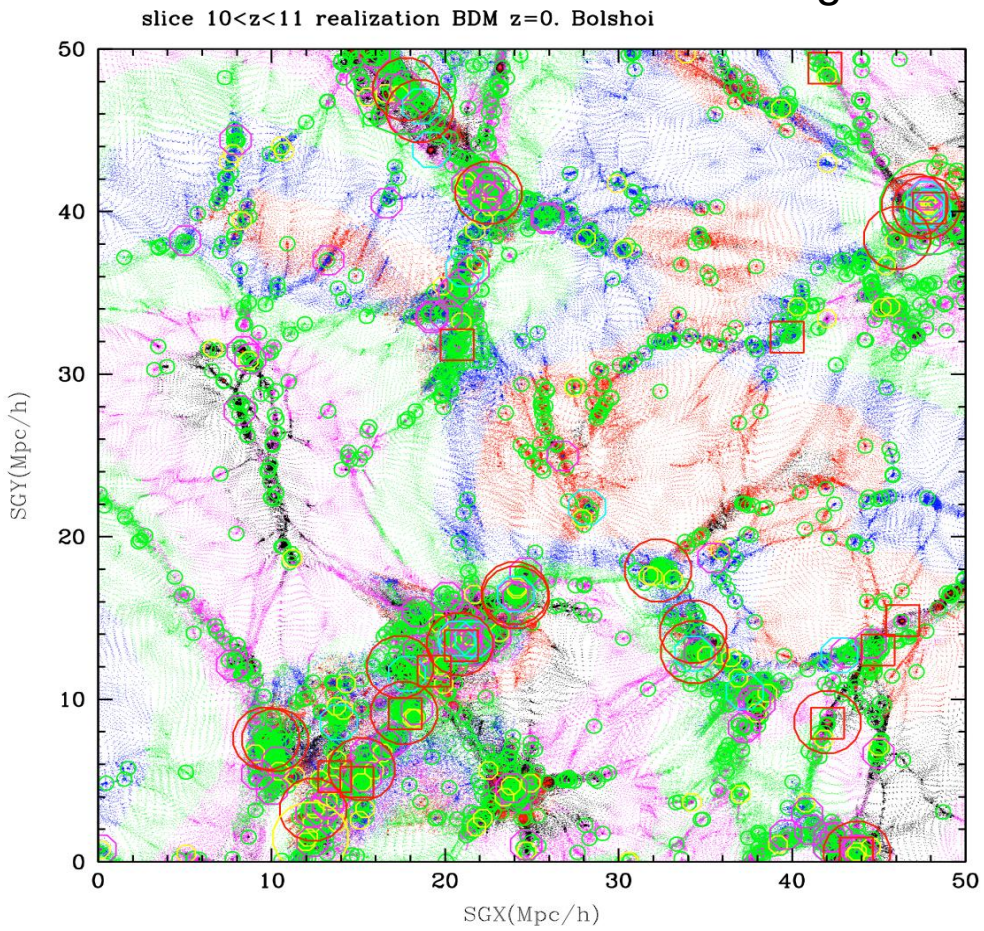
Fitting with a given functional form

→Supervised

Inputs: effective radius & velocity dispersion

Output: surface brightness

© Bolshoi Cosmological Simulation



Finding dark matter halos

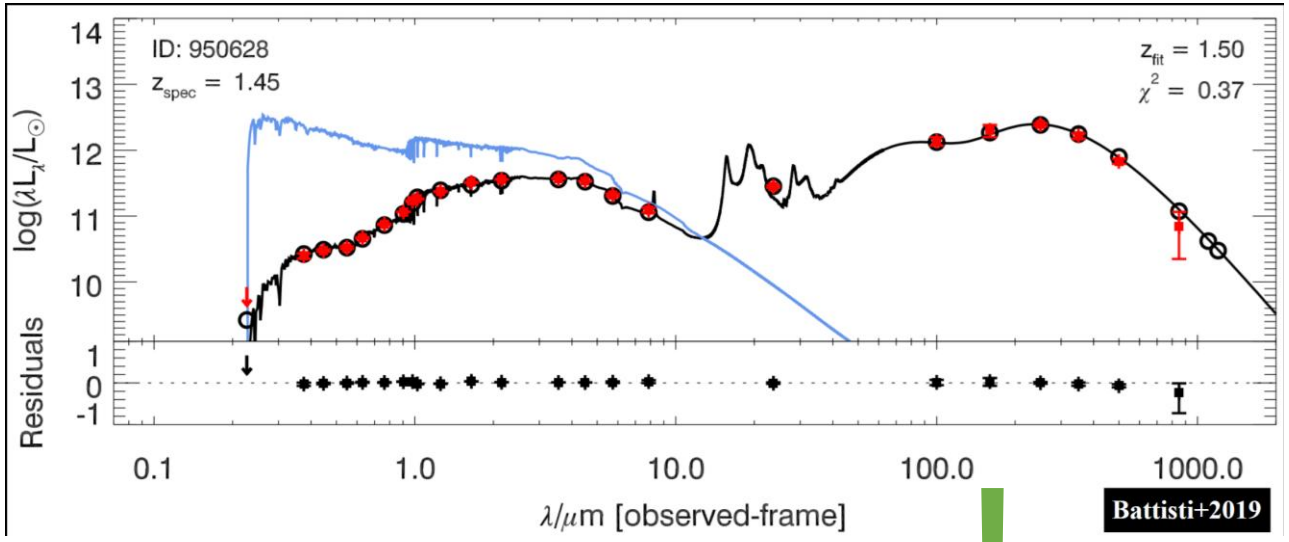
→Unsupervised

Inputs: particle positions

Outputs: halo IDs

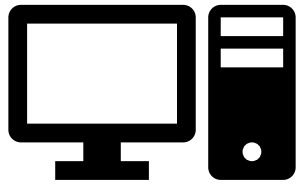
In Classical (Machine) Learning...

Wavelength	Intensity
X.XXXXXX	X.XXXXXX
X.XXXXXX	X.XXXXXX
X.XXXXXX	X.XXXXXX
X.XXXXXX	X.XXXXXX
X.XXXXXX	X.XXXXXX
…	…



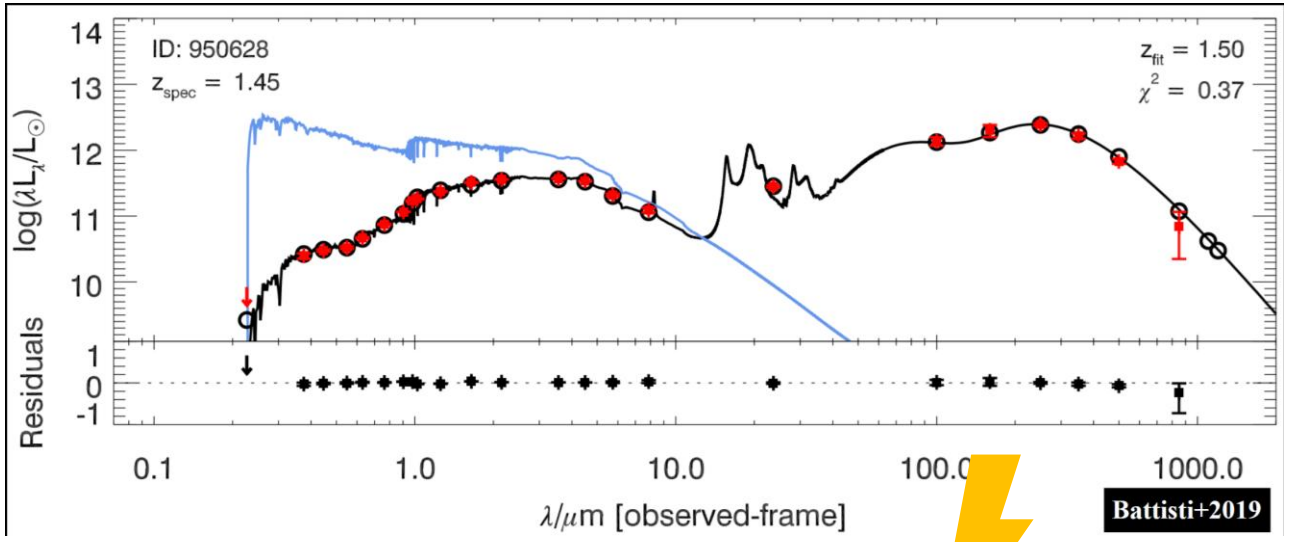
Find fitting formulae based on theory & experiences

Object type / redshift / ...



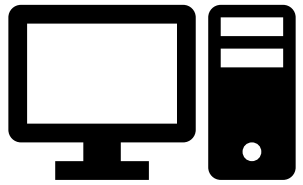
In Machine Learning...

Wavelength	Intensity
X.XXXXXX	X.XXXXXX
X.XXXXXX	X.XXXXXX
X.XXXXXX	X.XXXXXX
X.XXXXXX	X.XXXXXX
X.XXXXXX	X.XXXXXX
…	…



Set the learning strategies

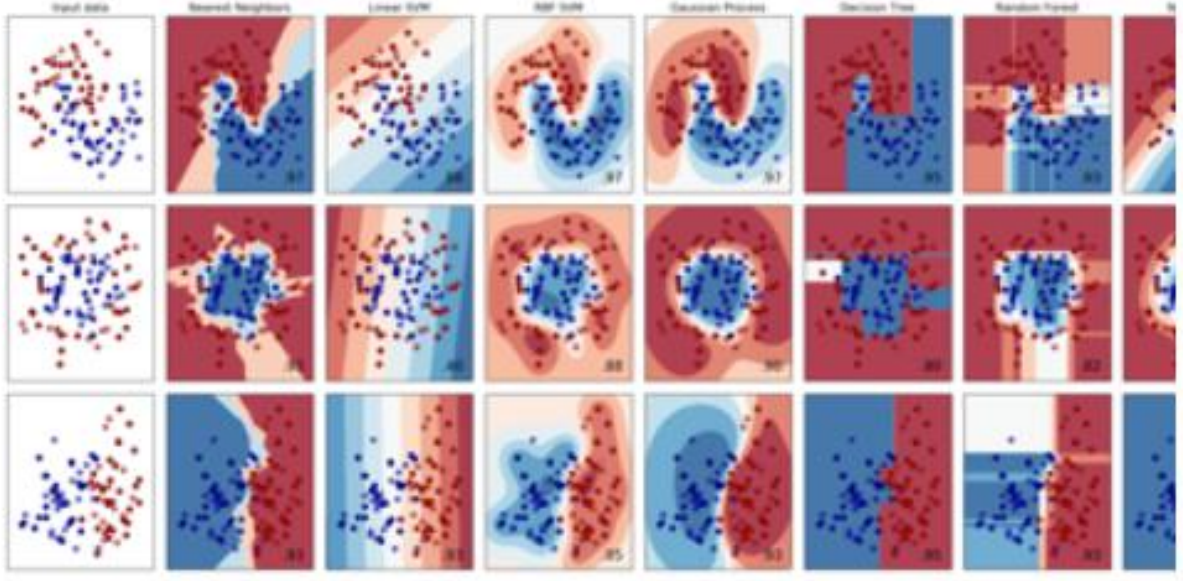
Object type / redshift / ...



Classification

Identifying which category an object belongs to.

Applications: Spam detection, image recognition.
Algorithms: SVM, nearest neighbors, random forest, and more...



© scikit-learn

Type: Supervised

Outputs: Vector

Each element means the probability of being each type

Example

Classify each image to either red, green, or blue!

Inputs	Outputs
	(1, 0, 0)
	(0, 1, 0)
	(0, 0, 1)

Regression

Predicting a continuous-valued attribute associated with an object.

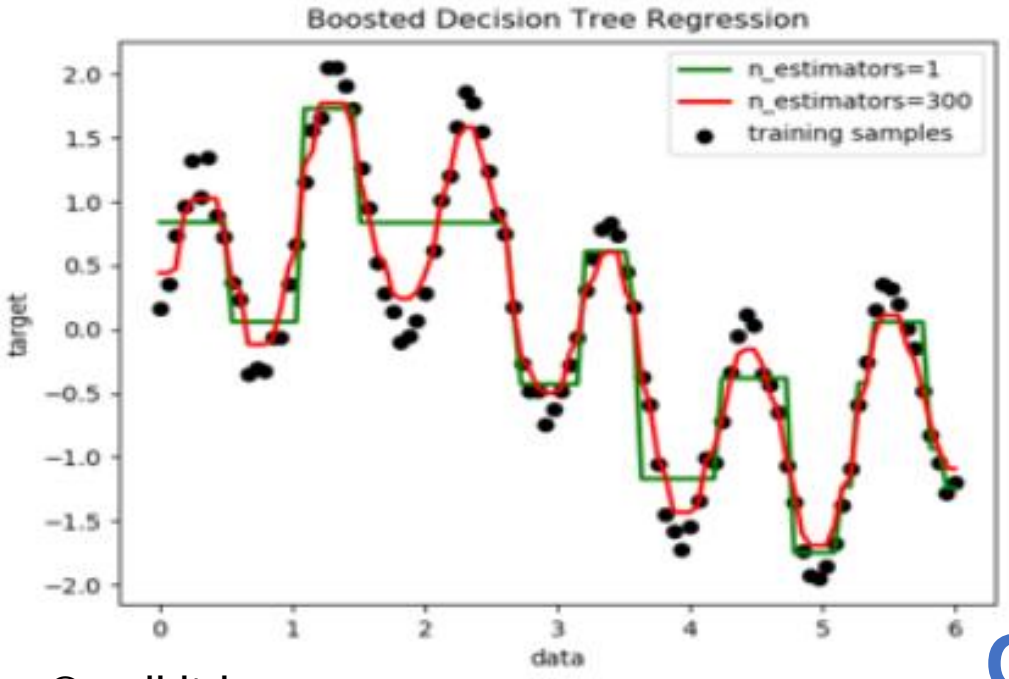
Applications: Drug response, Stock prices.
Algorithms: SVR, nearest neighbors, random forest, and more...

Type: Supervised

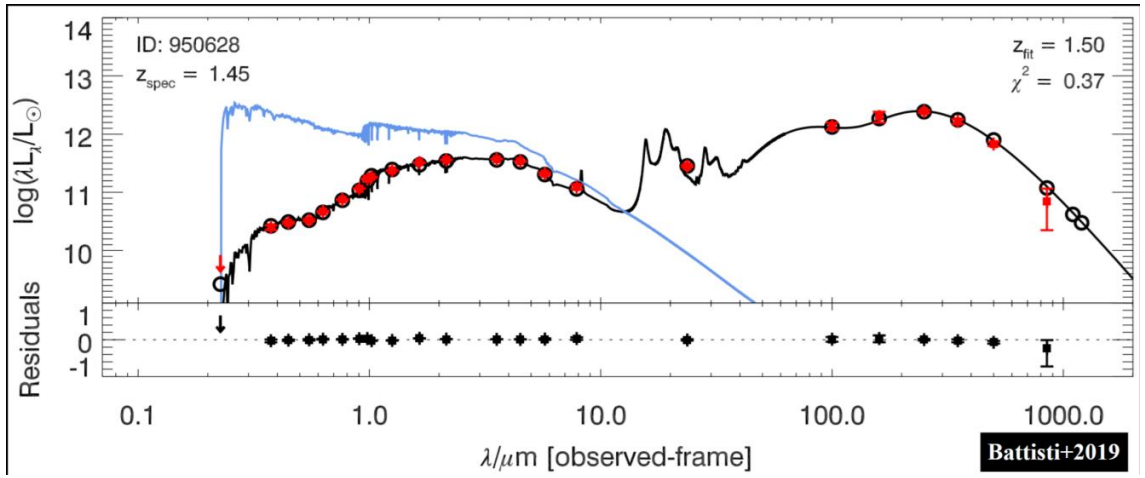
Outputs: Vector

Example

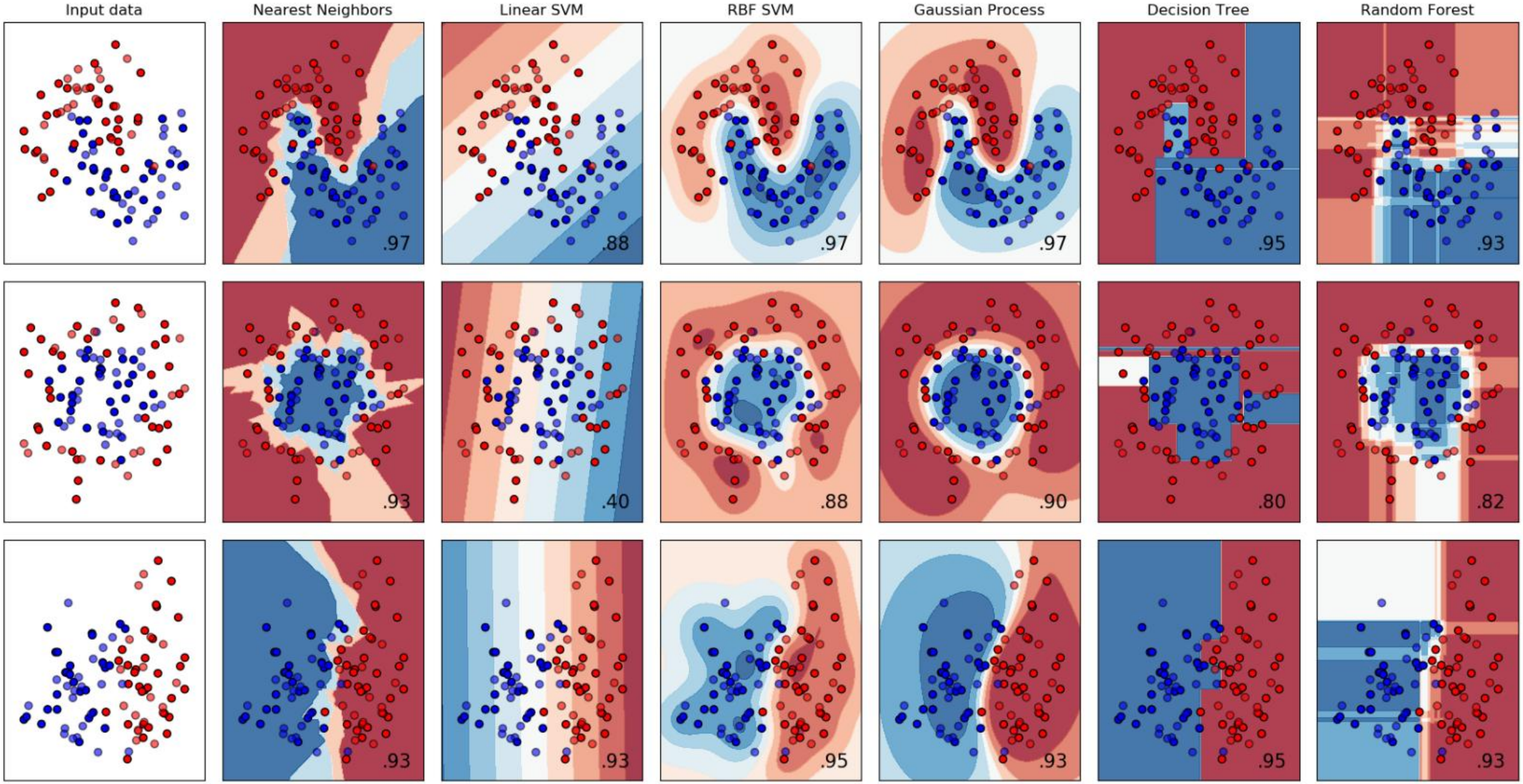
Find the object type and redshift from the spectrum



© scikit-learn



Classification ⟺ **Regression**


Input data
Nearest Neighbors
Linear SVM
RBF SVM
Gaussian Process
Decision Tree
Random Forest
.97
.88
.97
.97
.95
.93
.93
.40
.88
.90
.80
.82
.93
.93
.95
.93
.95
.93

Clustering

Automatic grouping of similar objects into sets.

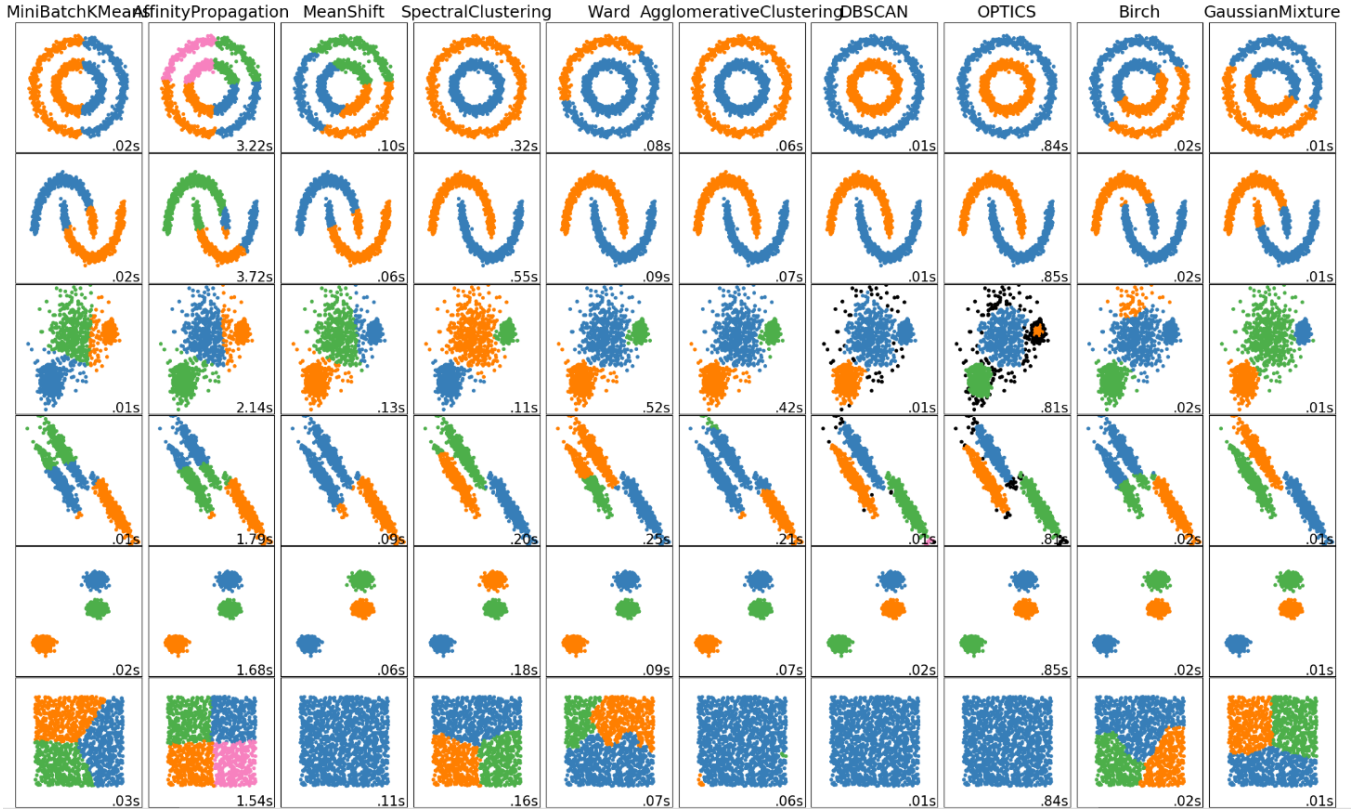
Applications: Customer segmentation, Grouping experiment outcomes

Algorithms: k-Means, spectral clustering, mean-shift, and more...

© scikit-learn

Type: Unsupervised

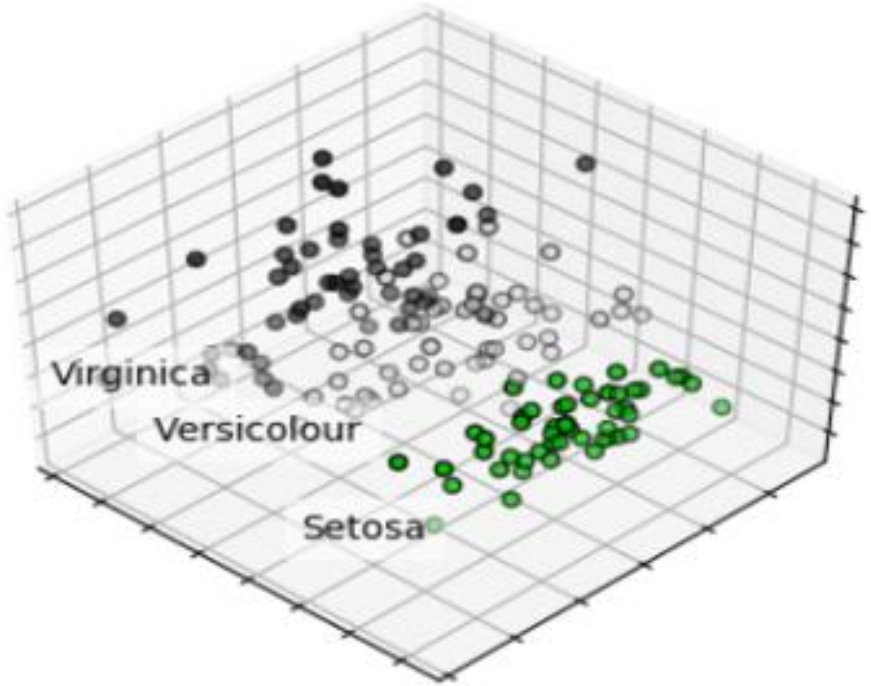
Outputs: Group ID



Dimensionality reduction

Reducing the number of random variables to consider.

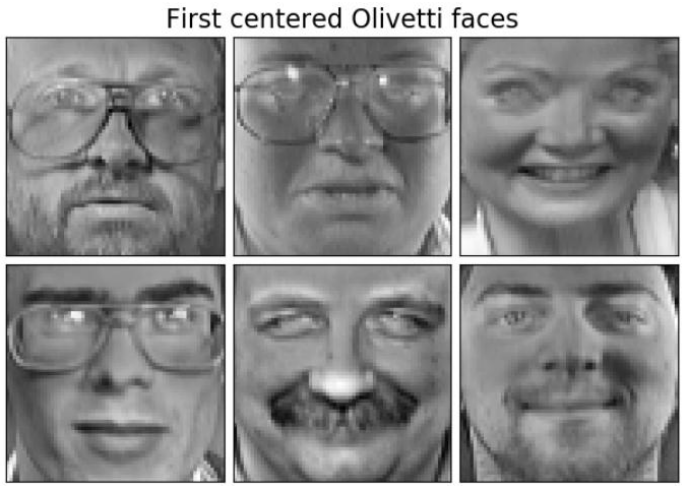
Applications: Visualization, Increased efficiency
Algorithms: k-Means, feature selection, non-negative matrix factorization, and more...

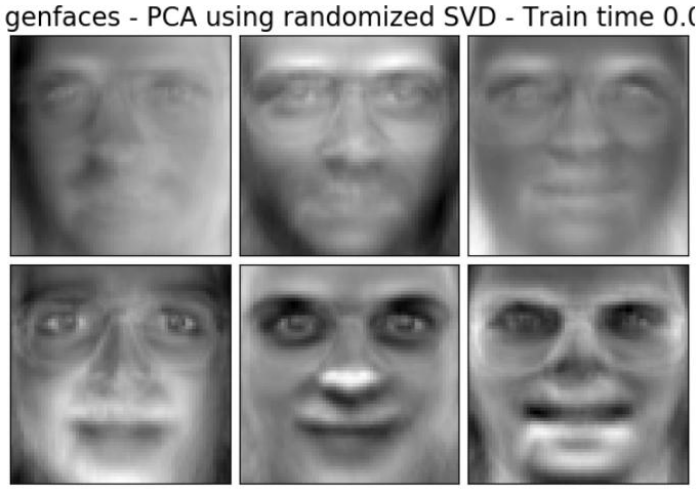


© scikit-learn

Type: Unsupervised

Outputs: Vectors with lower dimensions





Machine Learning

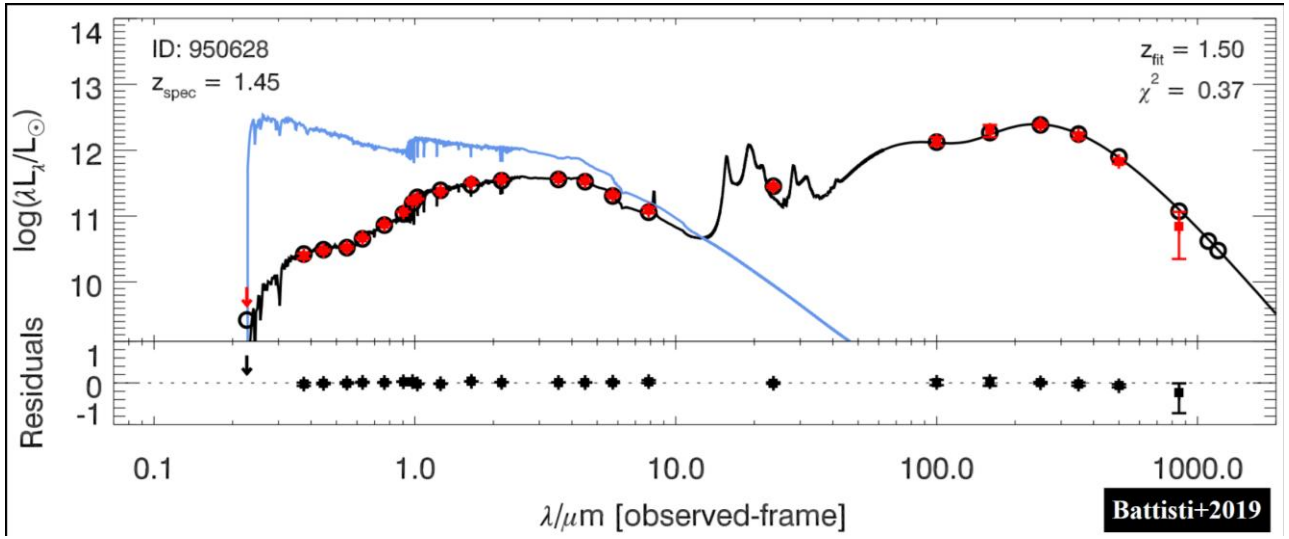
- All results should be lead by physics
  - Software is no replacement for physical understanding
- ML is a useful technique in three cases:
  - No known or understandable physical mechanism exists:
    - Dark energy
  - Physics known, but computing the required quantities is challenging:
    - Photometric redshifts
    - Large numbers of simulations - non-linear emulators
  - Paradigm testing: looking for things not predicted by any physics or models
    - Unknown unknown
    - Model-independent tests

Techniques (examples):

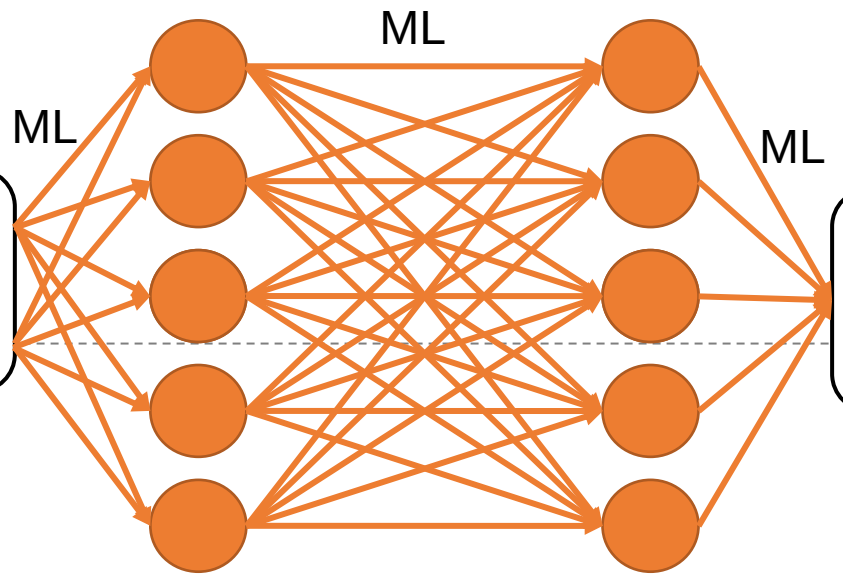
1. Classification/catagoriszation
   - Support Vector Machines
   - Naive Bayes
   - Perceptron/CNN
   - Decision Trees
2. Clustering
   - Hierarchical
   - k-means
   - DBSCAN
   - Density
3. Interpolation/Regression
   - Gaussian Process Regression
   - Kernel Ridge Regression
   - Multivariate adaptive regression splines

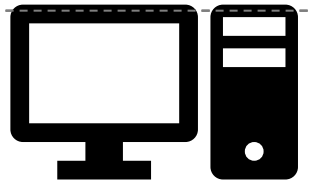
Then, In Deep Learning…

Wavelength	Intensity
X.XXXXXX	X.XXXXXX
X.XXXXXX	X.XXXXXX
X.XXXXXX	X.XXXXXX
X.XXXXXX	X.XXXXXX
X.XXXXXX	X.XXXXXX
…	…

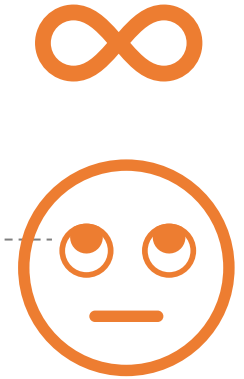


- Wavelength
- Intensity



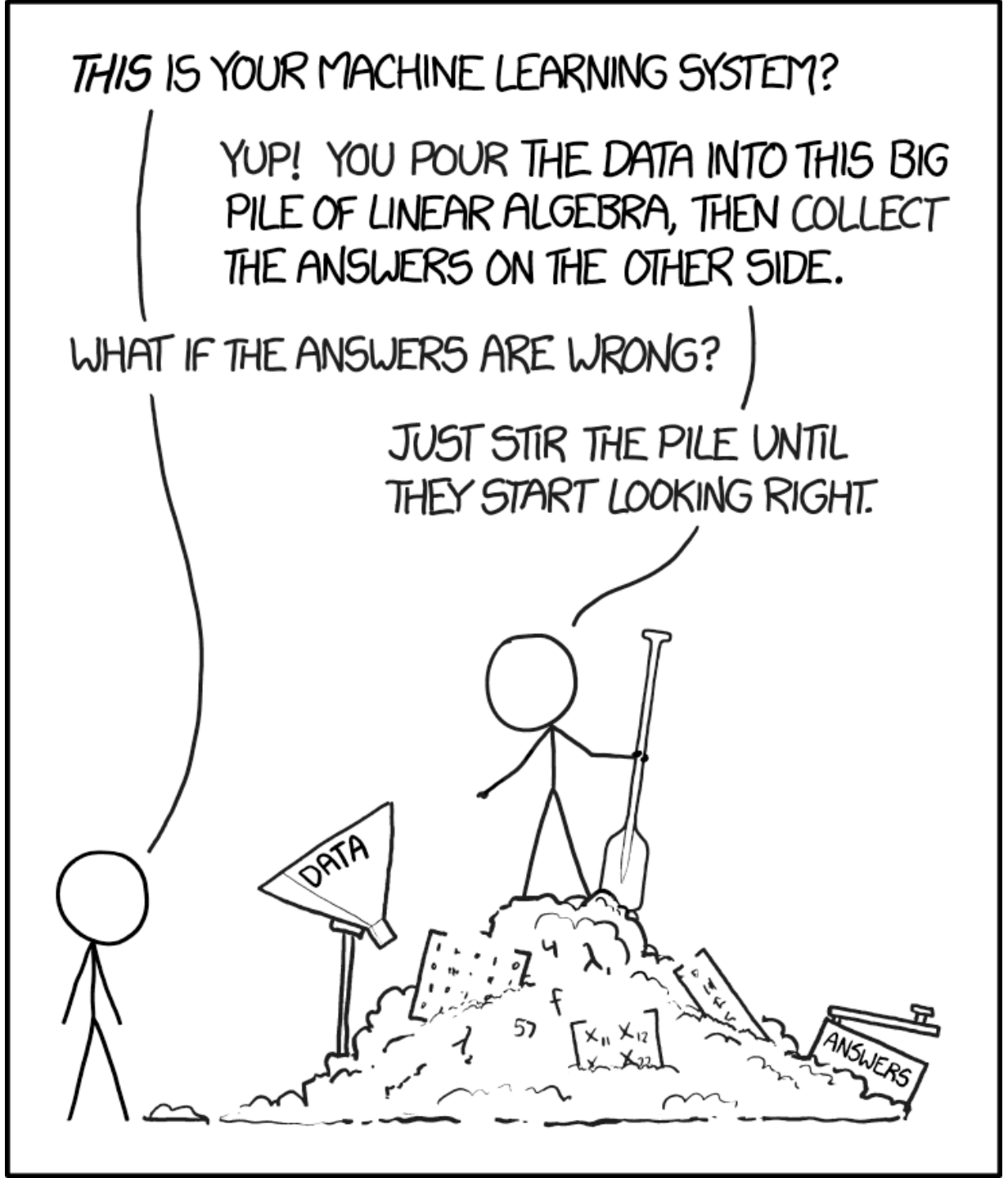


- Object type
- Redshift



Compared to ML, DL is…

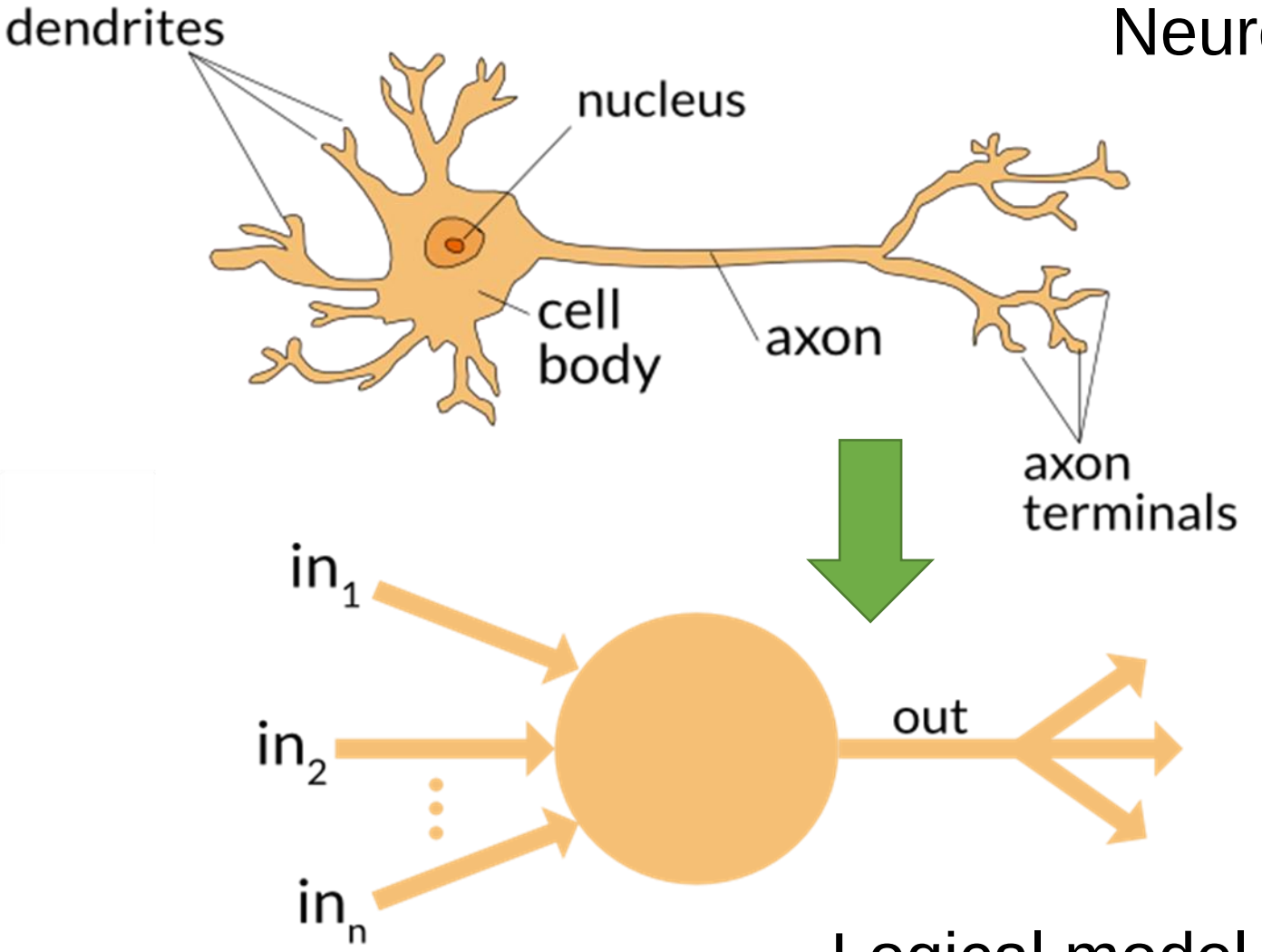
- Harder to understand the underlying physics
- Having too many degrees of freedom to build models
- Requiring more data and computational resources (> GPUs) for successful learning
- Harder to guarantee the success of learning mathematically
- Capable of solving very difficult problems, once successful



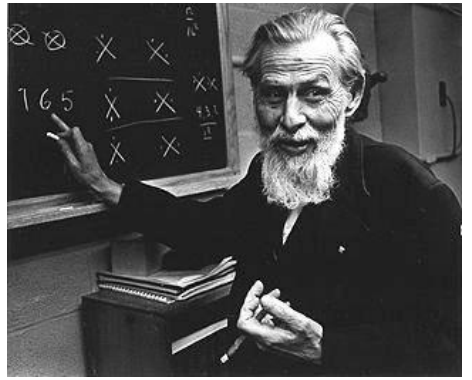
So... What is deep learning?

A brief history... in 1943

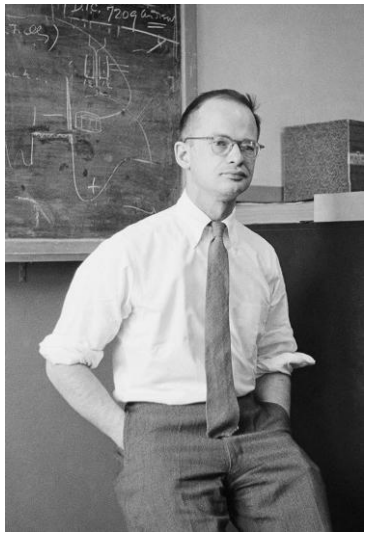
Neuron



Logical model

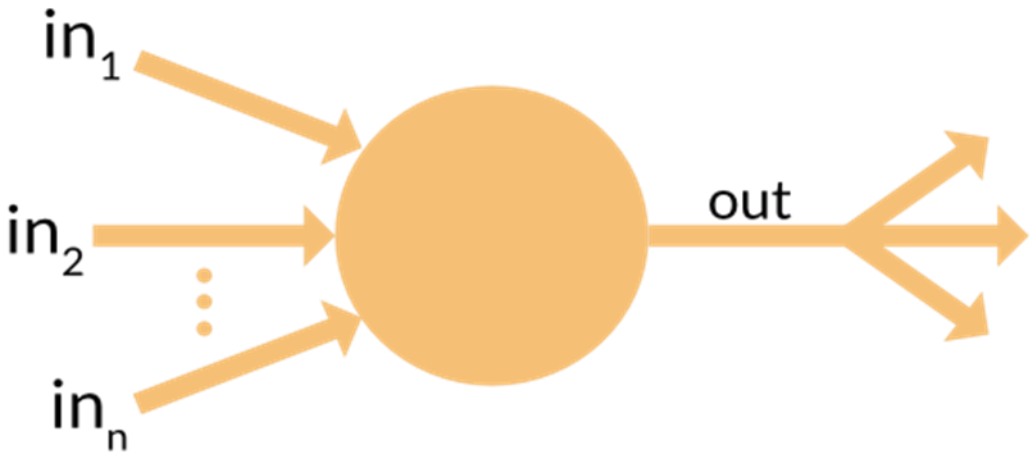


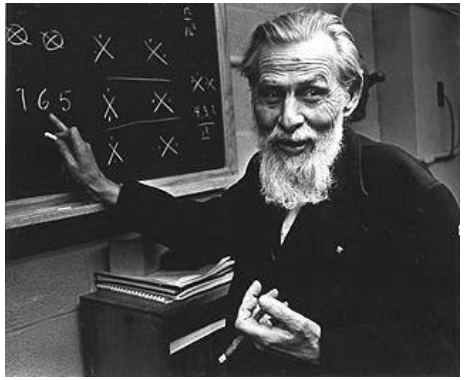
Warren McCulloch



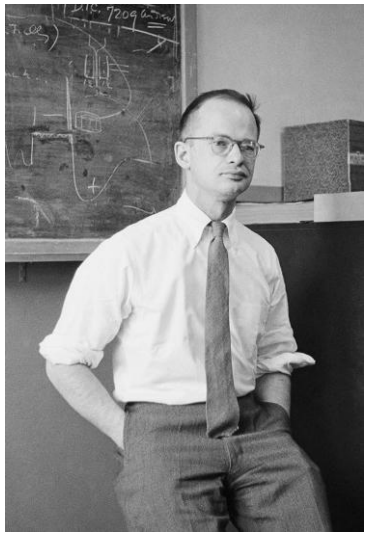
Walter Pitts

A brief history... in 1943





Warren McCulloch



Walter Pitts

Activation

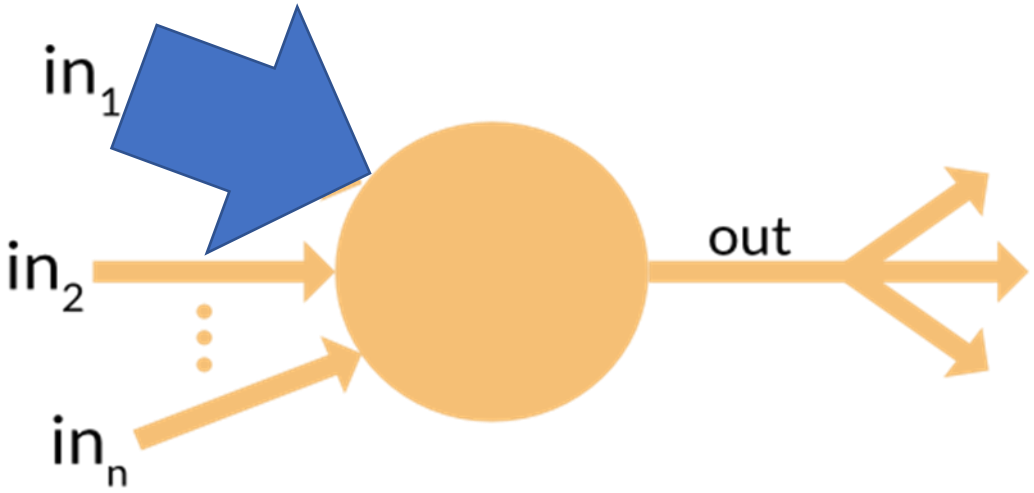
Threshold

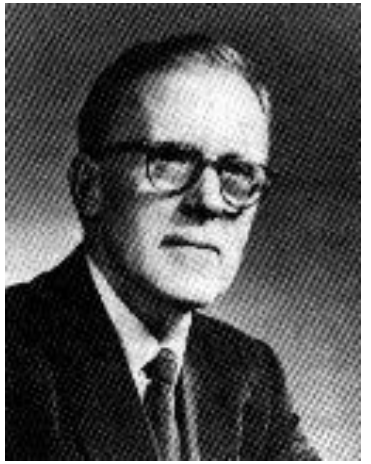
$$Y = \begin{cases} 1, & \sum_{i \in E} X_i \geq \Theta \text{ and } \sum_{j \in I} X_j = 0 \\ 0, & \sum_{i \in E} X_i < \Theta \text{ or } \sum_{j \in I} X_j > 0 \end{cases}$$

Excitation

Inhibition

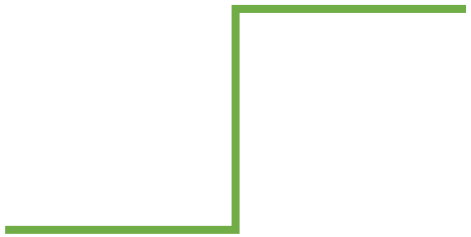
A brief history... in 1949





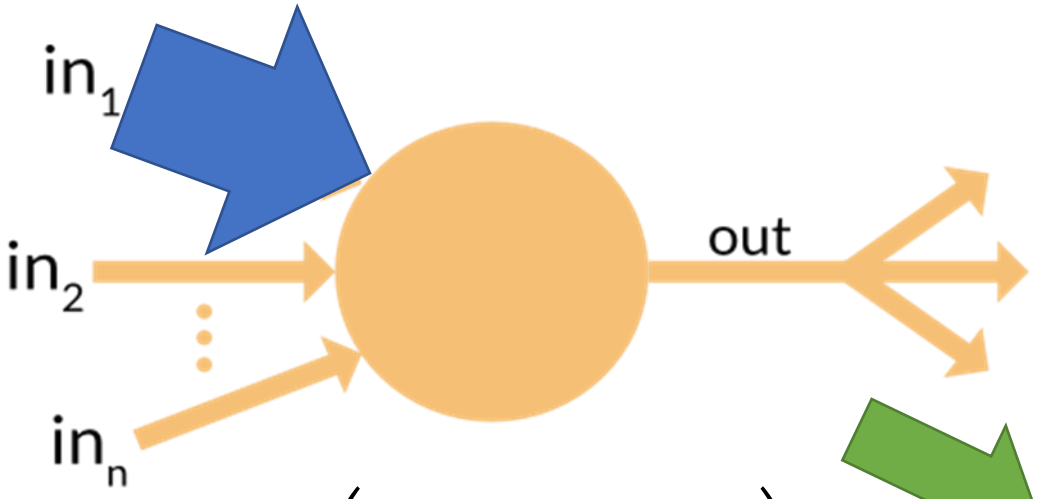
Donald Hebb

$$Y = H\left(\sum_i \omega_i X_i - \Theta\right)$$



$$X_i \uparrow \text{ and } Y \uparrow \Longrightarrow \frac{\mathrm{d}\omega_i}{\mathrm{d}t} > 0 \text{ (Synaptic strength)}$$

A brief history... in 1957



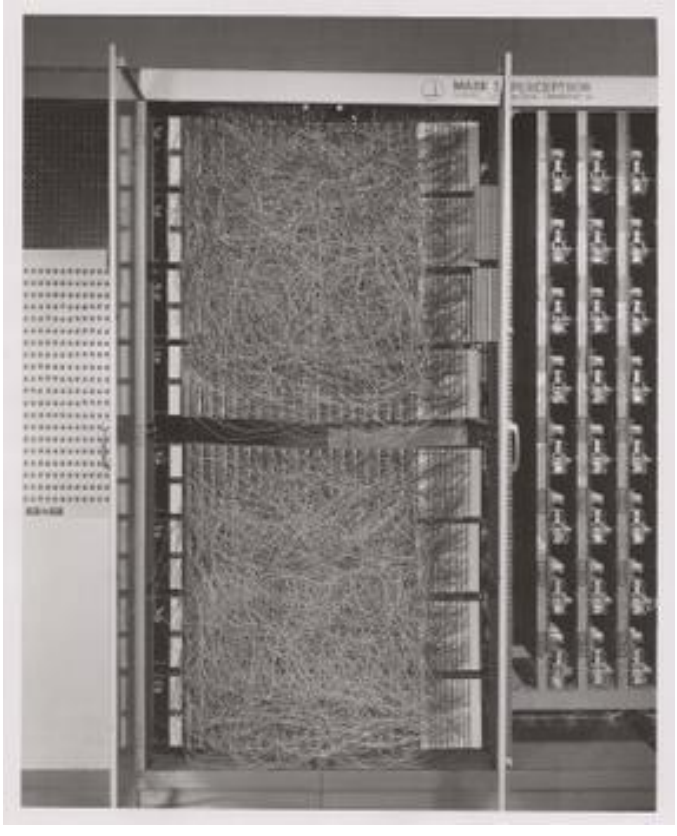
$$Y = H\left(\sum_i \omega_i X_i - \Theta\right)$$

$$\frac{\mathrm{d}\omega_i}{\mathrm{d}t} = \eta X_i Y$$

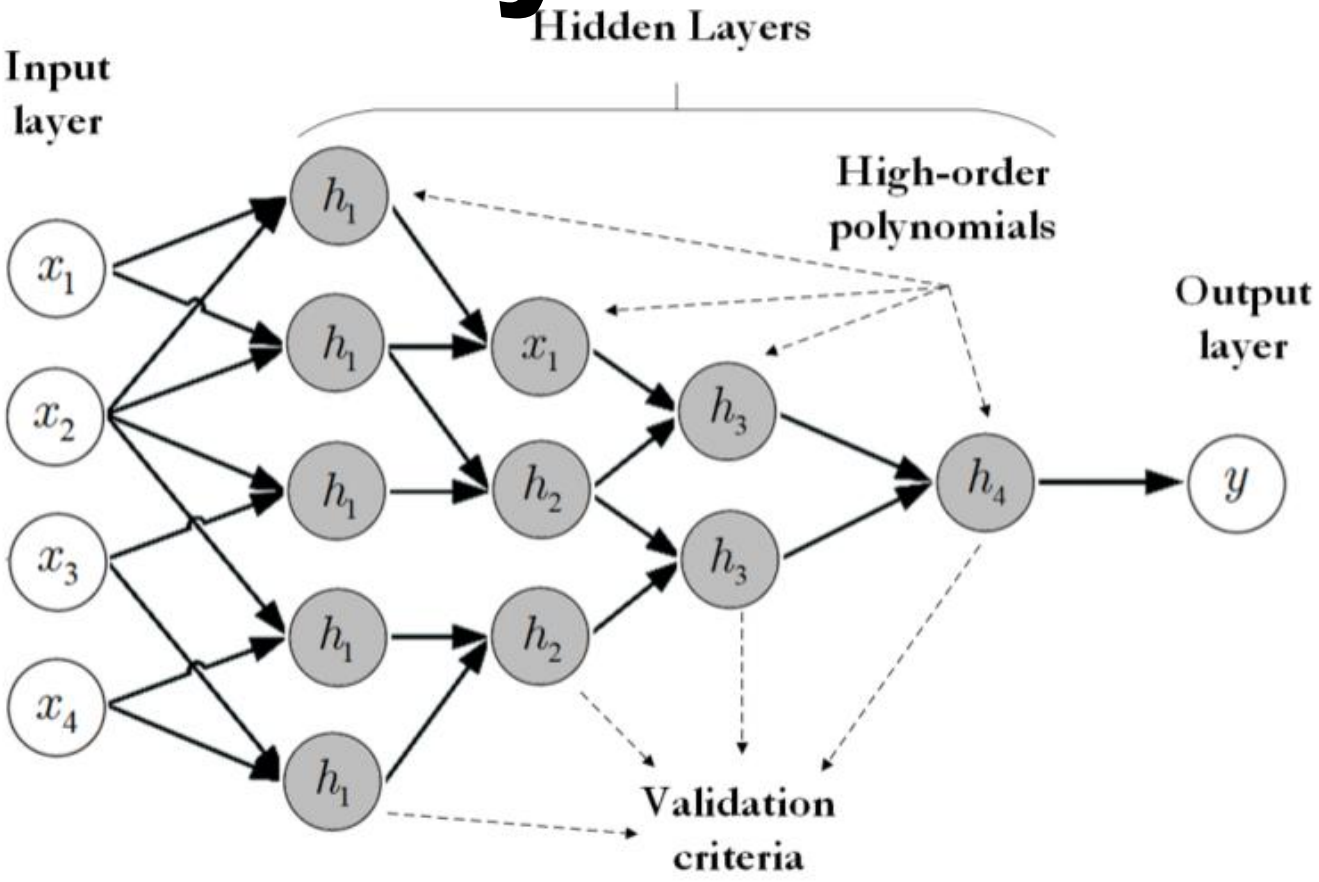
Perceptron

The first artificial neural network

Frank Rosenblatt



A brief history… in 1966



Alexey Ivakhnenko

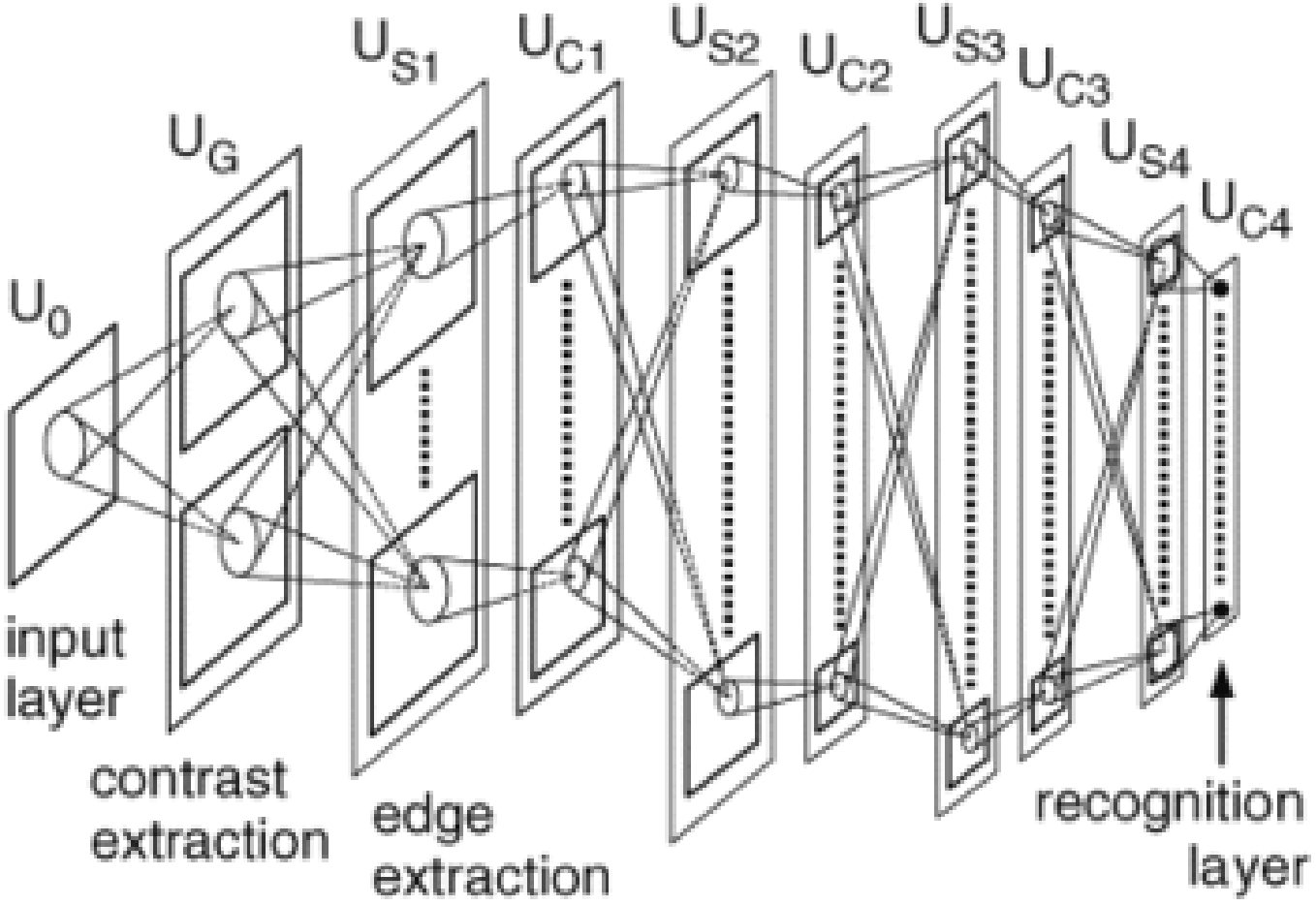
Radaideh & Kozlowski (2019)

$$y = a_0 + \sum_{i=1}^{d} a_i x_i + \sum_{i=1}^{d}\sum_{j=1}^{d} a_{ij} x_i x_j + \sum_{i=1}^{d}\sum_{j=1}^{d}\sum_{k=1}^{d} a_{ijk} x_i x_j x_k + \ldots$$

Group Method of Data Handling (GMDH)

One of the first deep (hidden-layer) learning algorithms

A brief history... in 1980

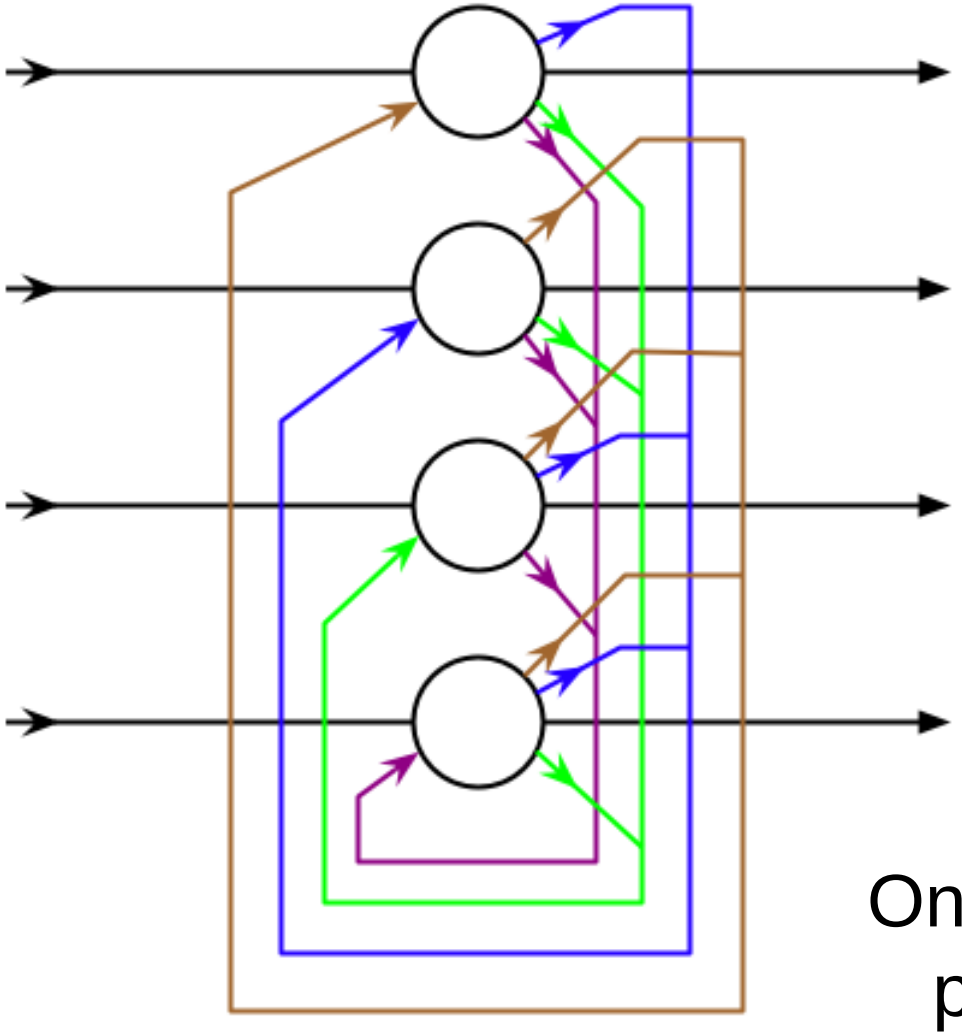


Kunihiko Fukushima

Neocognitron

One of the first convolutional neural networks

A brief history... in 1982

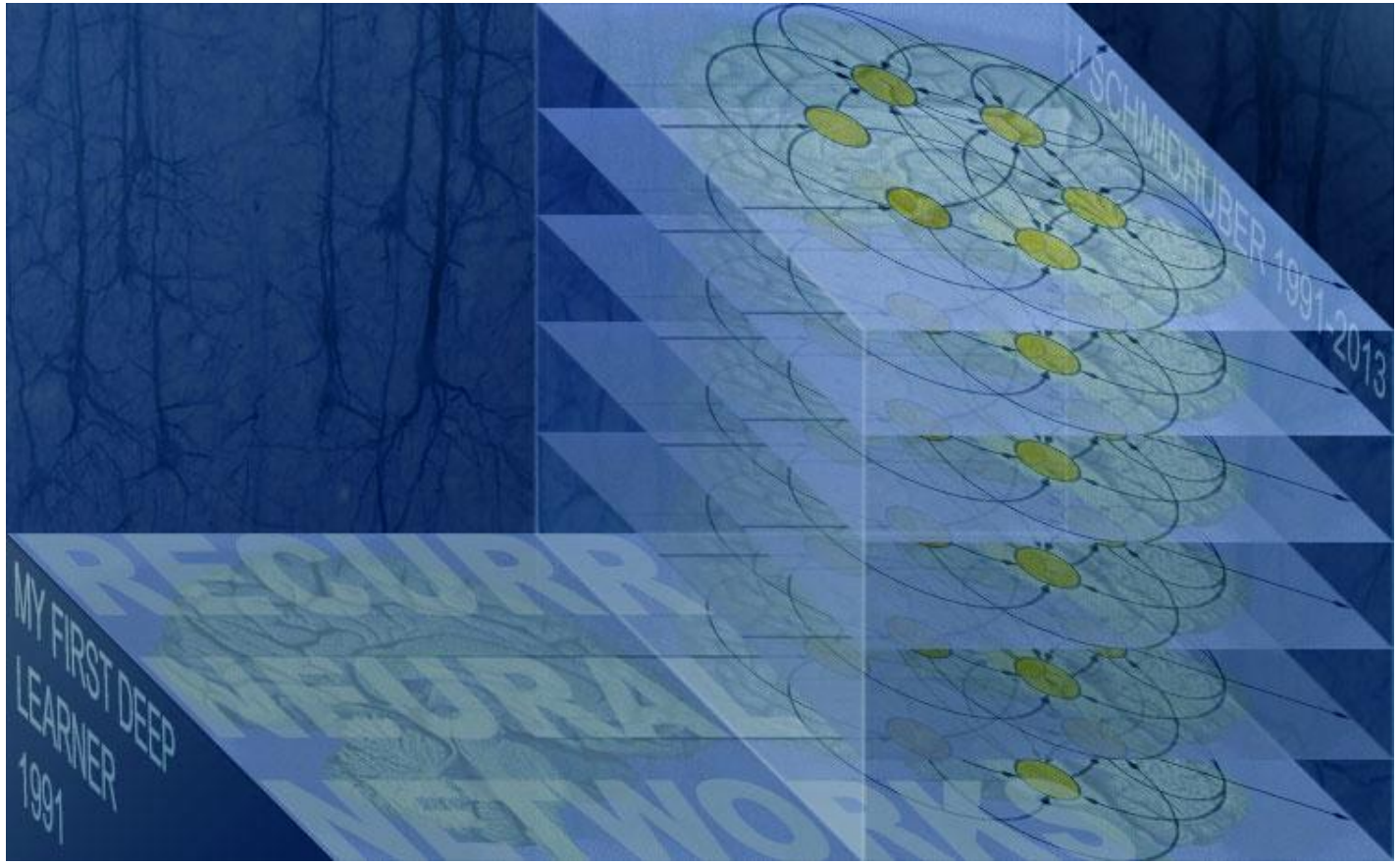


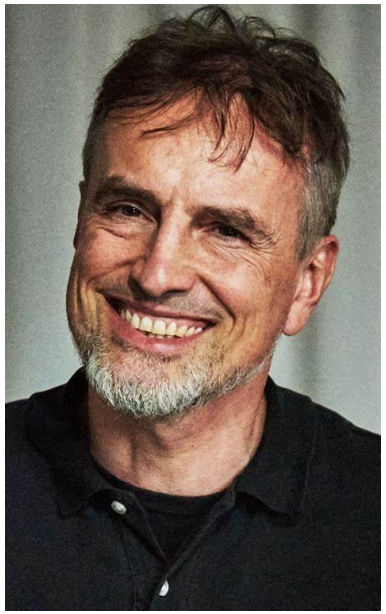
John Hopfield

Hopfield Network

One of the first recursive neural networks → preserves information from former steps

A brief history... in 1991





Jurgen Schmidhuber

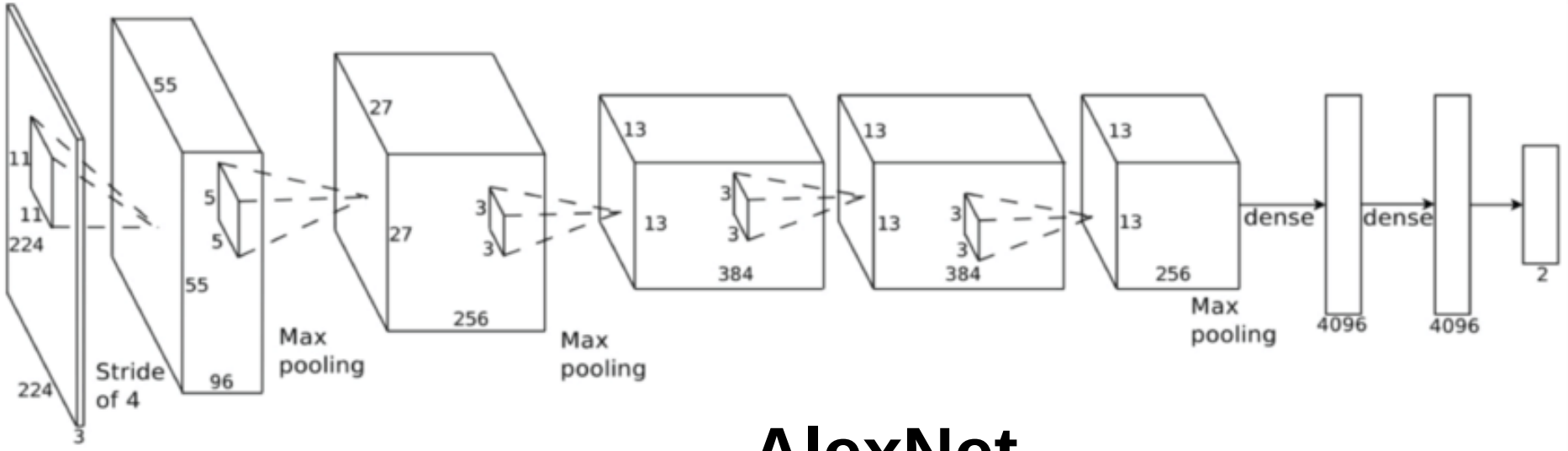
Neural Sequence Chunker

Very deep learning with > 1000 layers

A brief history... in 2011

IMAGENET

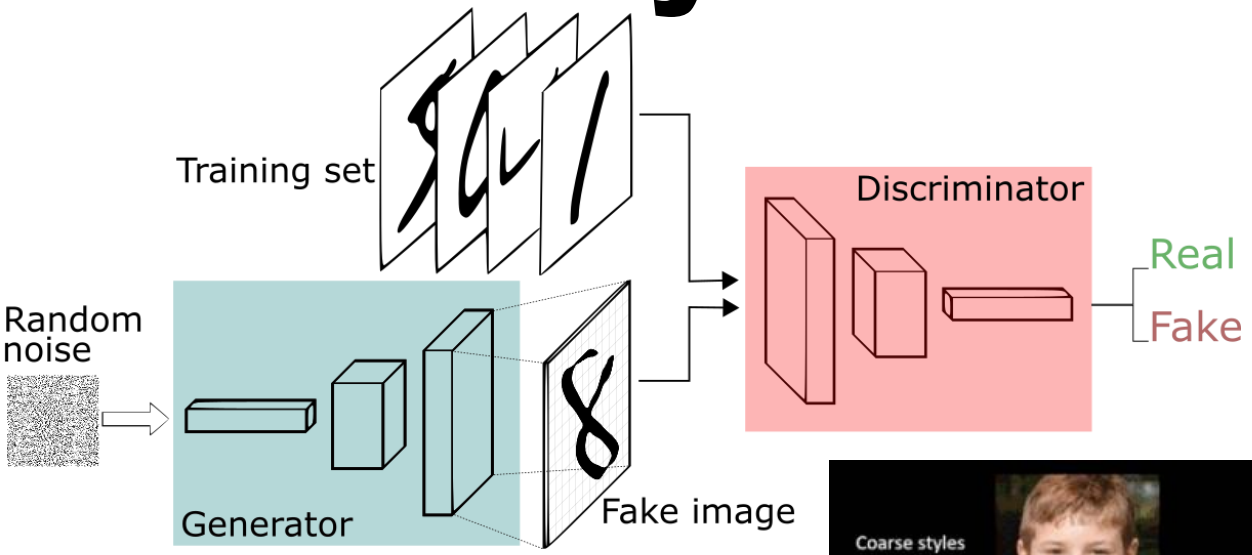
Alex Krizhevsky



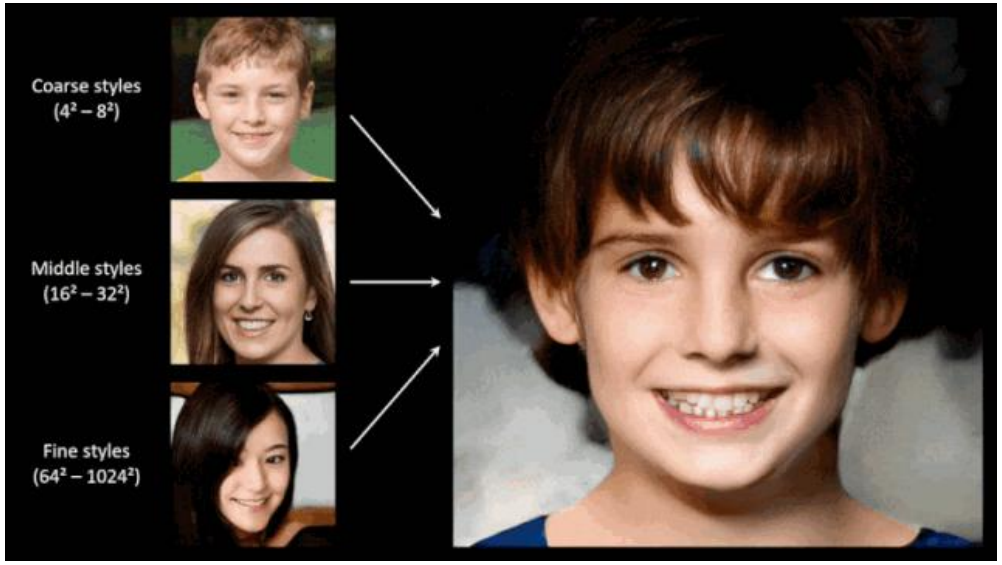
AlexNet

Putting (convolutional) deep learning as a superior algorithm

A brief history... in 2014

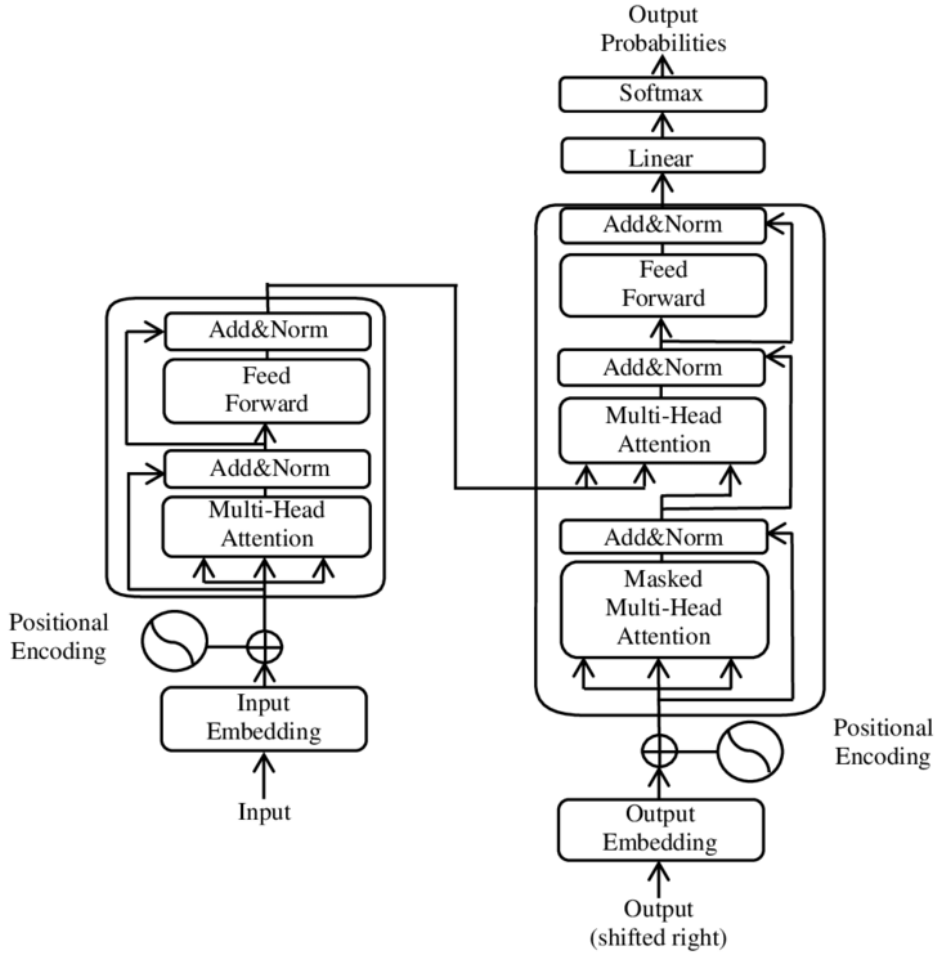


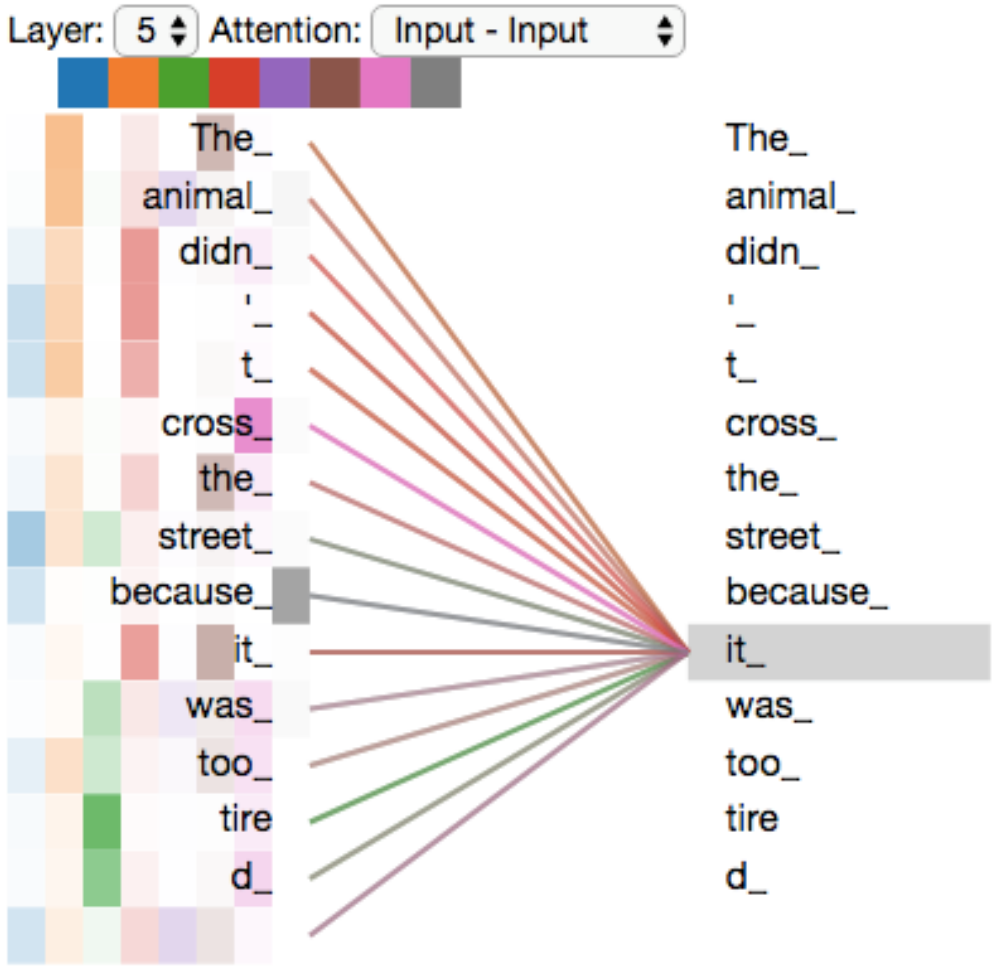
Ian Goodfellow



Generative Adversarial Network

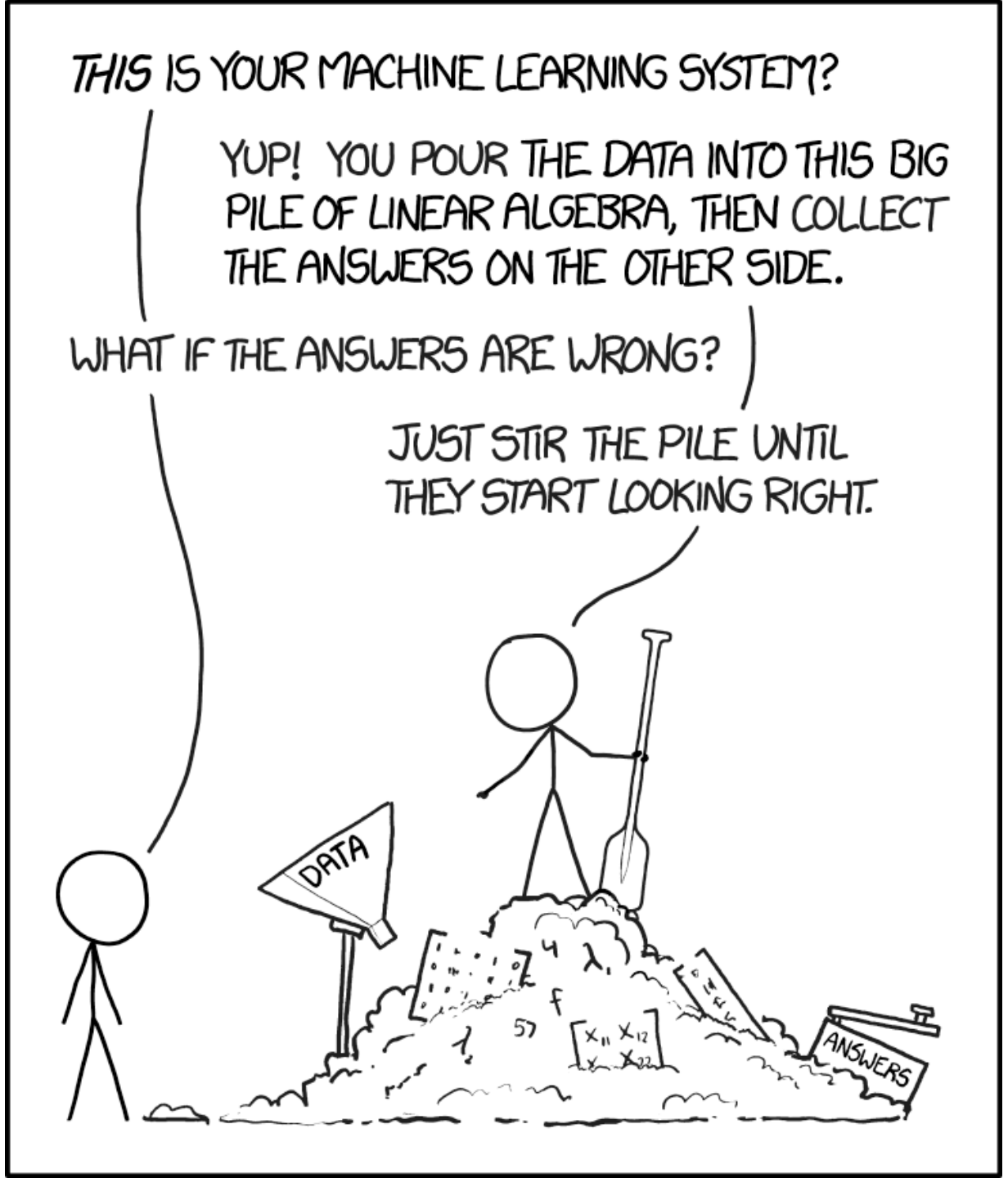
A brief history... in 2017





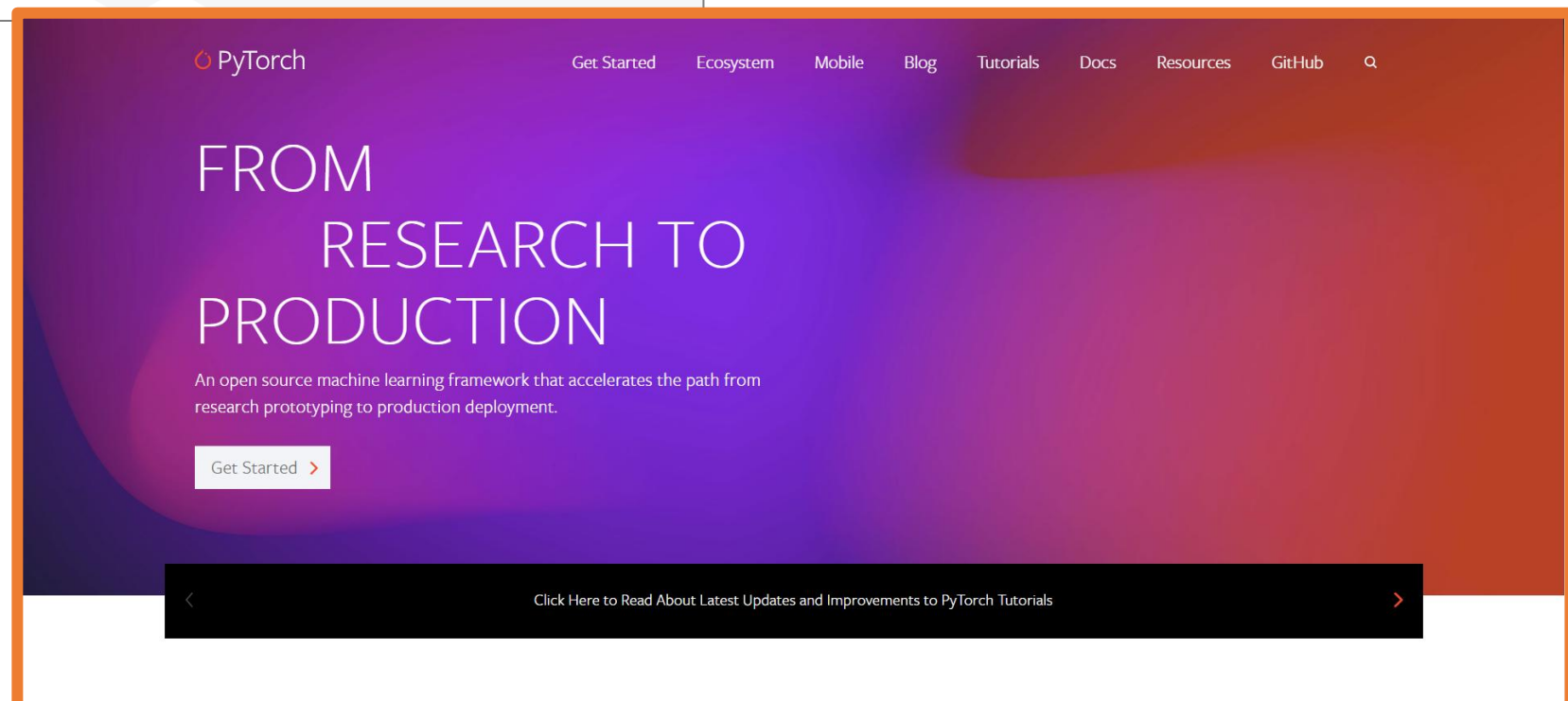
Attention Is All You Need

Transformer model → GPT

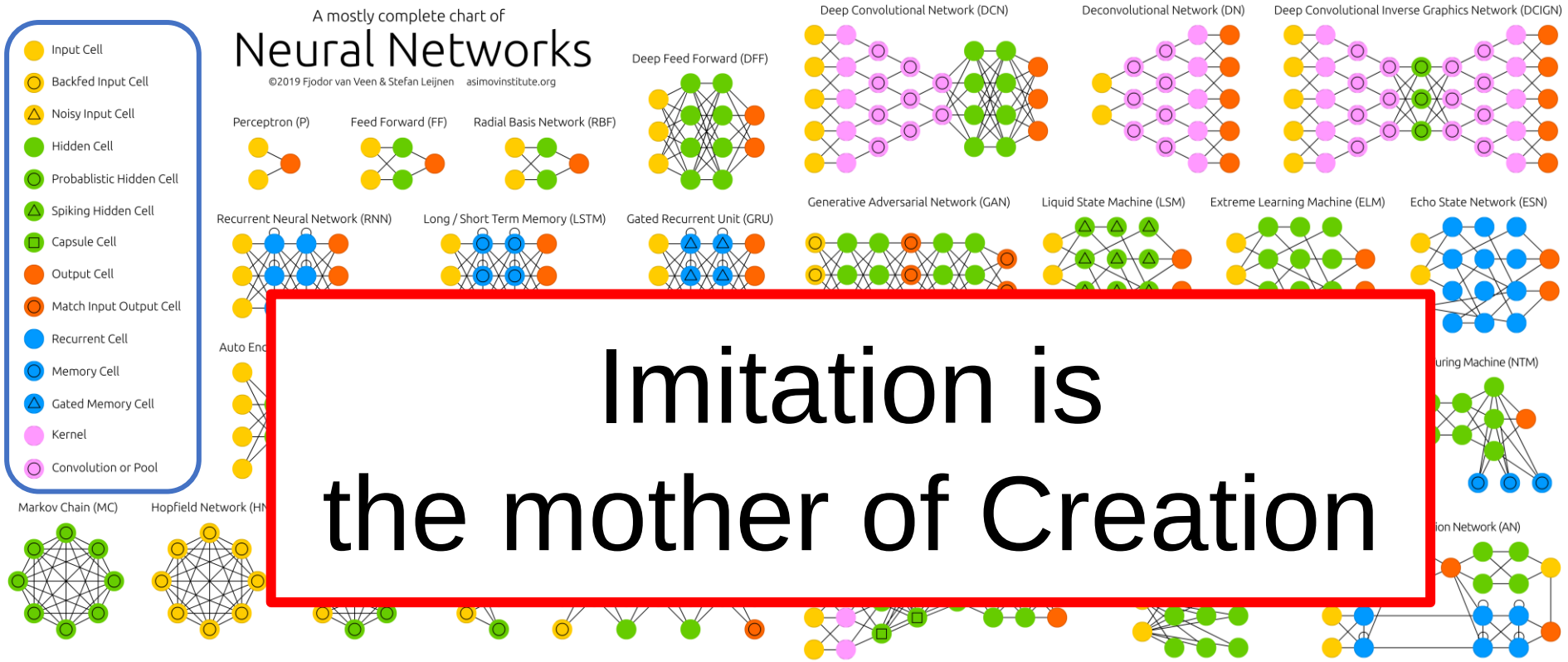


Let’s do it!

Tensorflow
(http://www.tensorflow.org/)



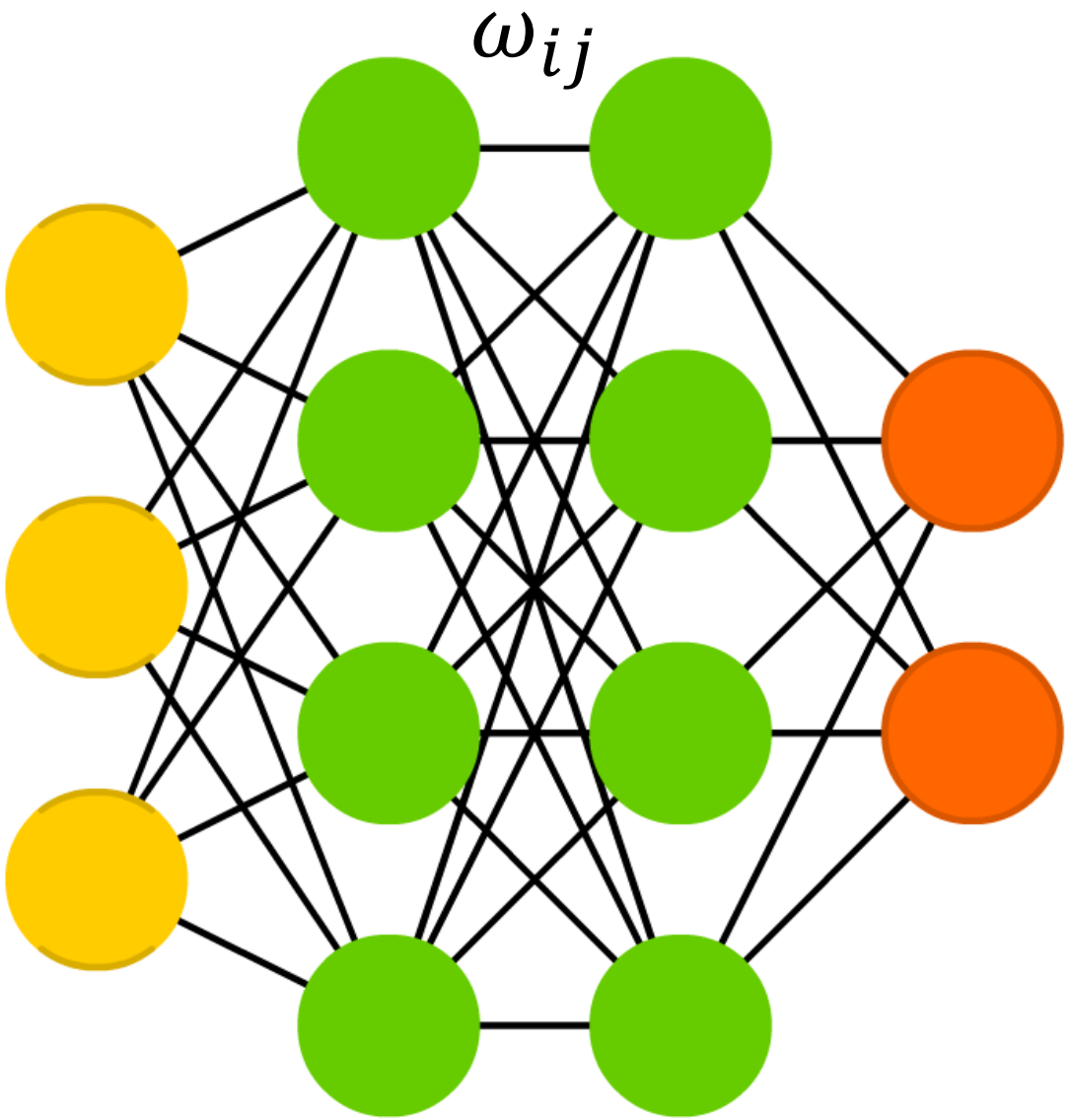
PyTorch
(http://pytorch.org/)



Imitation is the mother of Creation

https://www.asimovinstitute.org/neural-network-zoo/

Layer = Lego block



`torch.nn.Linear(`*`n_element_before`*`,` *`n_element_after`*`)`

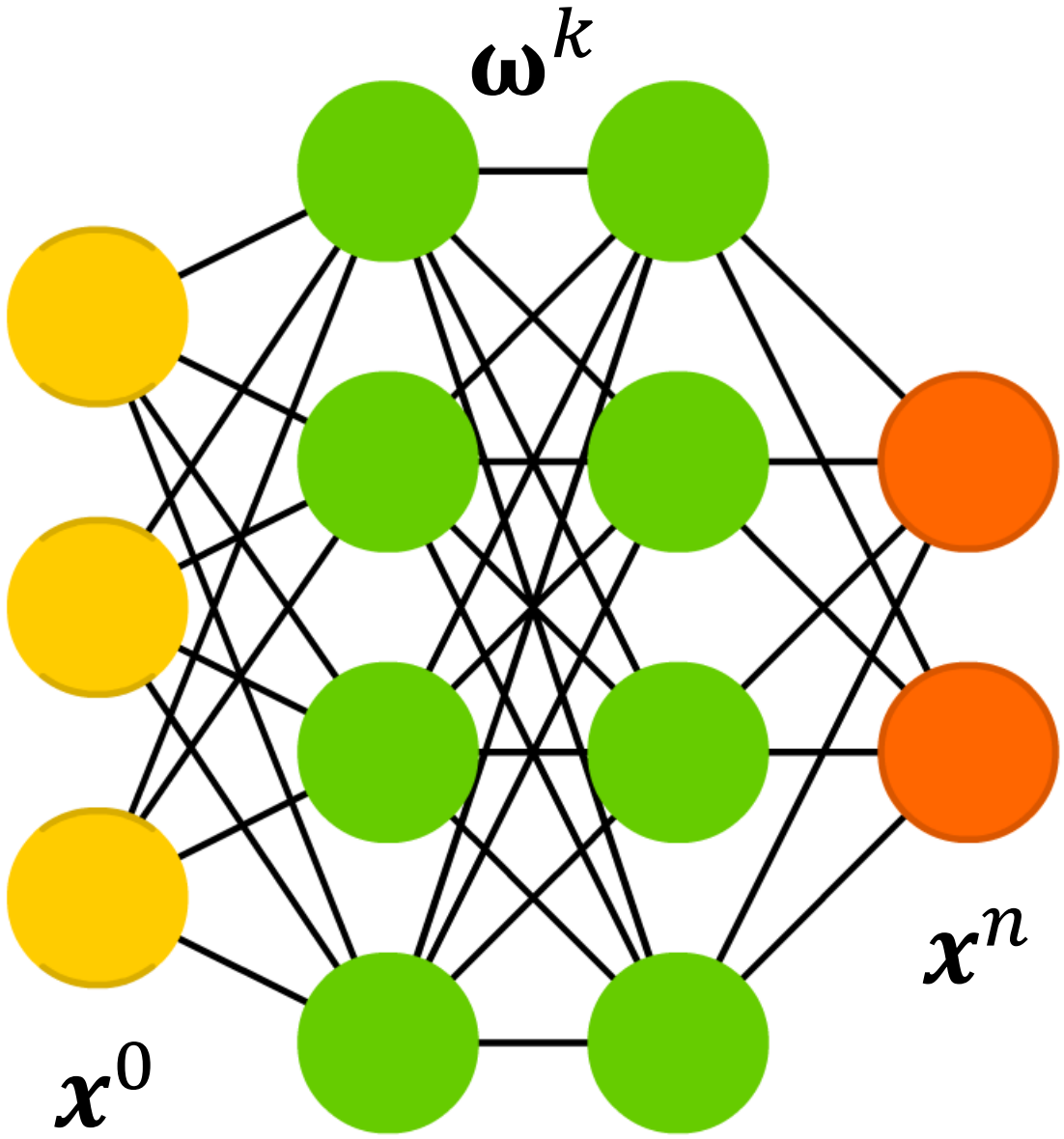
Fully Connected Layer

- Calculate the vector by using linear algebra from the vector at the former layer

$$y_i = \sum_j \omega_{ij} x_j + b_i$$

- Ideal if each vector element has a different meaning
  (e.g., different physical parameters)

- Useful even when different vector elements share a similar meaning
  (to reduce the contamination by noise)



For any set of $\{\boldsymbol{\omega}^k, \boldsymbol{b}^k\}$ with

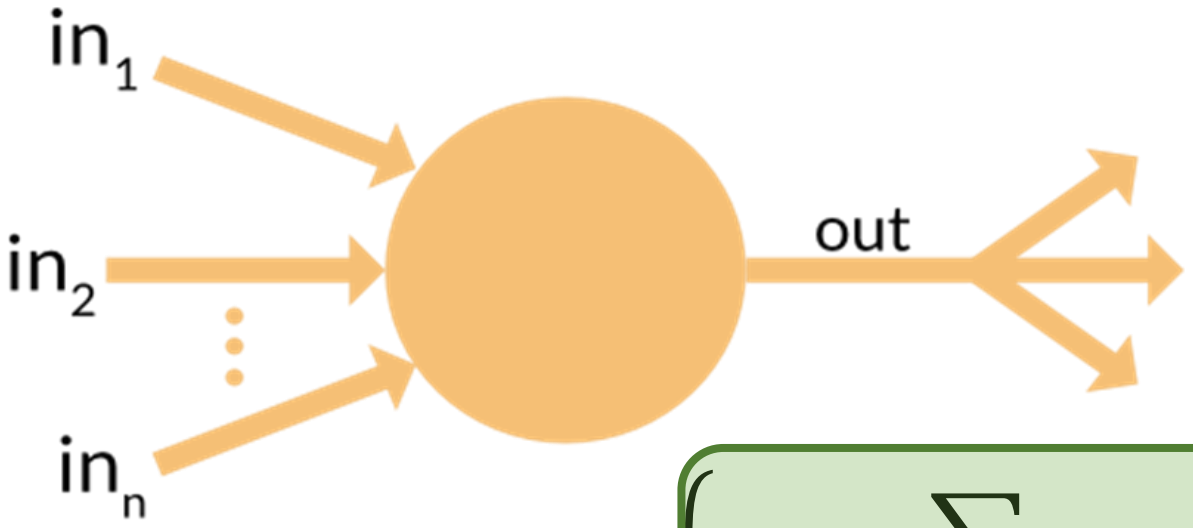
$$\boldsymbol{x}^k = \boldsymbol{\omega}^k \boldsymbol{x}^{k-1} + \boldsymbol{b}^k$$

$$(k = 1, \ldots, n),$$

there always exists $\{\boldsymbol{\omega}, \boldsymbol{b}\}$ that gives the same result with

$$\boldsymbol{x}^n = \boldsymbol{\omega} \boldsymbol{x}^0 + \boldsymbol{b}$$

What's the benefit of deep layers?



Activation

$$Y = \begin{cases} 1, & \sum_{i \in E} X_i \geq \Theta \text{ and } \sum_{j \in I} X_j = 0 \\ 0, & \sum_{i \in E} X_i < \Theta \quad \text{or} \quad \sum_{j \in I} X_j > 0 \end{cases}$$

Let's apply nonlinear functions for each layer!

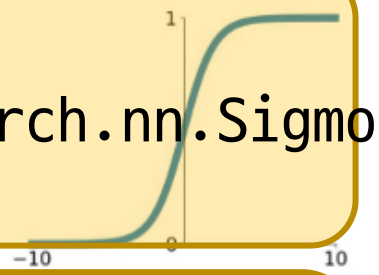
→ Activation Funtion

Activation Functions

Sigmoid

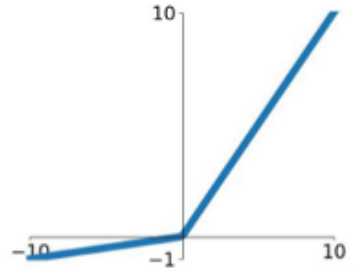
$\sigma(x) = \frac{1}{1+e^{-x}}$

`torch.nn.Sigmoid()`



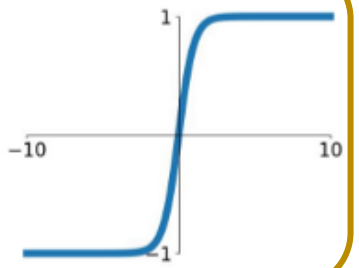
Leaky ReLU

$\max(0.1x, x)$



tanh

$\tanh(x)$



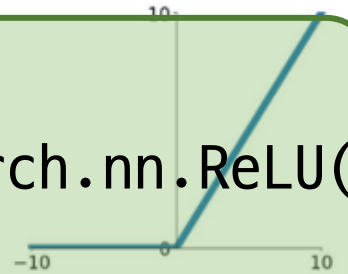
Maxout

$\max(w_1^T x + b_1, w_2^T x + b_2)$

ReLU

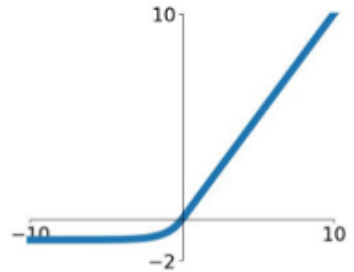
$\max(0, x)$

`torch.nn.ReLU()`



ELU

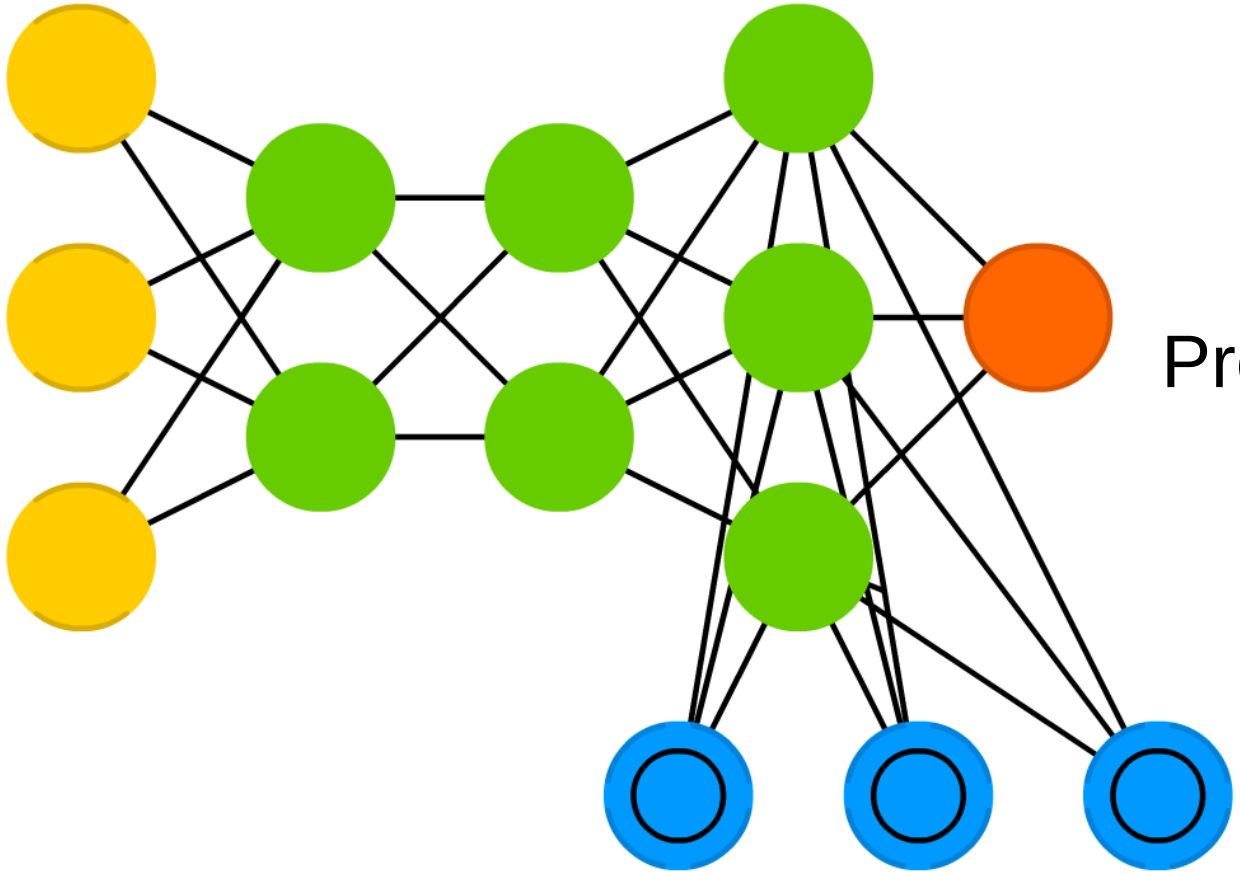
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



https://medium.com/@shrutijadon10104776/survey-on-activation-functions-for-deep-learning-9689331ba092

Softmax $y_i = \frac{\exp x_i}{\sum_i \exp x_i} \rightarrow \sum_i y_i = 1$ `torch.nn.Softmax(`*dim*`)`

Mostly used at the last layer in classification problems



X_i
Inputs

y_i
Predictions

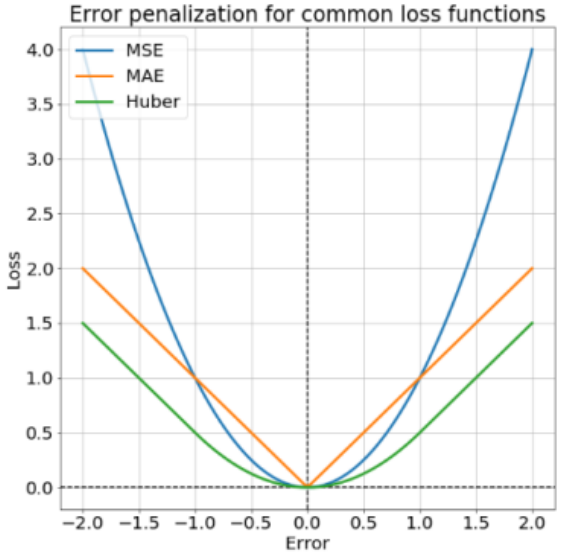
Y_i
Truths

$y_i \approx Y_i$

→ **Minimize** $\mathcal{L}(\{y_i\}, \{Y_i\})$

Loss Function

Regression



- **MSE** Mean square error (L2)
  - Strongly penalizes outliers
  - High sensitivity near minimum
- **MAE** Mean absolute error (L1)
  - Scales linearly with size of error
  - Low sensitivity near minimum
- **Huber**
  - Similar to MSE near minimum
  - Similar to MAE away from minimum

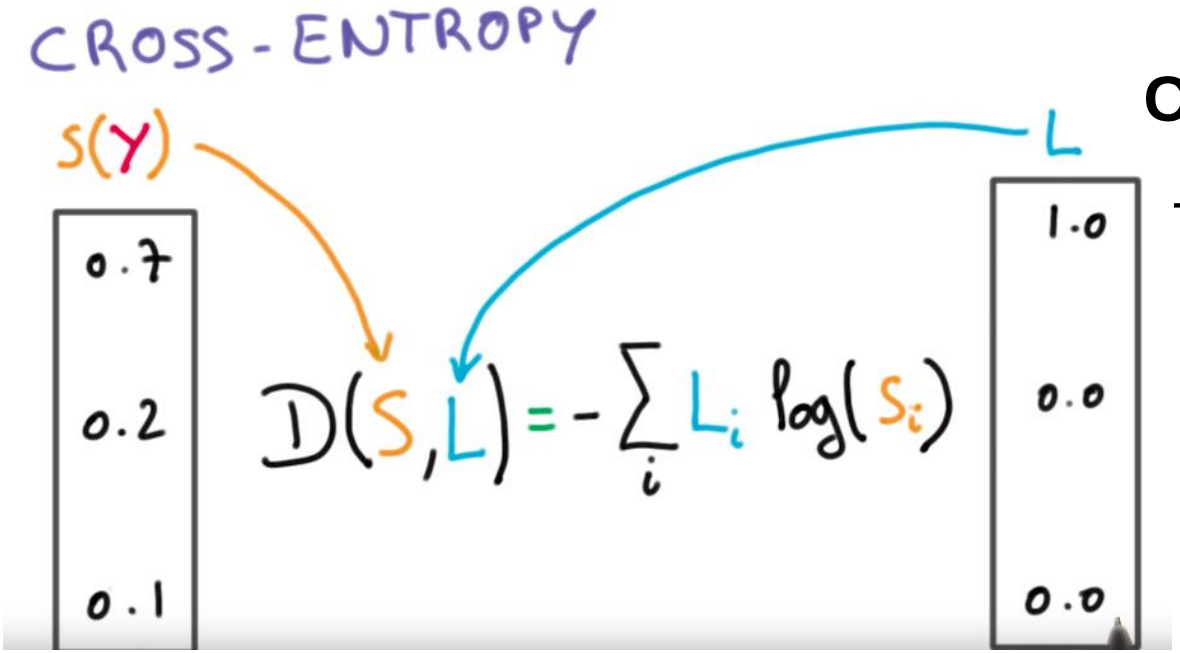
$$\mathcal{L} = \sum_i (y_i - Y_i)^2$$

`torch.nn.MSELoss()`

$$\mathcal{L} = \sum_i |y_i - Y_i|$$

`torch.nn.L1Loss()`

https://datascience103579984.wordpress.com/2020/01/26/introduction-to-tensorflow-in-python-from-datacamp/2/



Classification

`torch.nn.CrossEntropyLoss()`

https://medium.com/data-science-bootcamp/understand-cross-entropy-loss-in-minutes-9fb263caee9a

Before you start ML/DL…

- Is it really necessary?
  - Do you want to increase precision, or understand the nature?
  - Isn’t classic fitting enough?

- Can you trust ML/DL results?
  - Peers/referees do not easily believe both your methods and results!
  - Please perform other analyses to convince them!

Input/Output Data

- Which parameters will you use?
- **Normalization** (0~1? Logarithmic scale? Probability?)

Parameters showing the strength of my character

- Level (Lv. 1 ~ 99)
- Level / Max. Level (0 ~ 1)
- Experience Point (Exp. 1 ~ ∞)
- Log (Experience Point) (0 ~ ∞)
- Ranking (1st ~ ∞)

Input/Output Data

- Which parameters will you use?
- **Normalization** (0~1? Logarithmic scale? Probability?)
- Data format (Scalar? 1/2/3...-dimensional vector?)

Parameters showing the size of a galaxy

- Magnitude
- Magnitude – Standard Magnitude
- Stellar Mass
- Log (Stellar Mass)
- Mass Function

Infinite Limits

After explaining to a student through various lessons and examples that

$$\lim_{x \to 8^+} \frac{1}{x-8} = \infty$$

What if they also used other numbers?

I tried to check if he really understood that, so I gave him a different example.

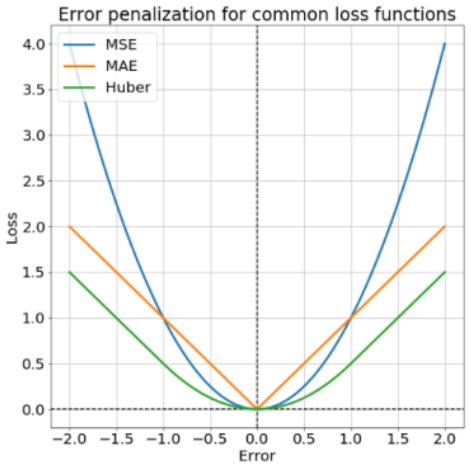
This was the result:

$$\lim_{x \to 5^+} \frac{1}{x-5} = 5$$

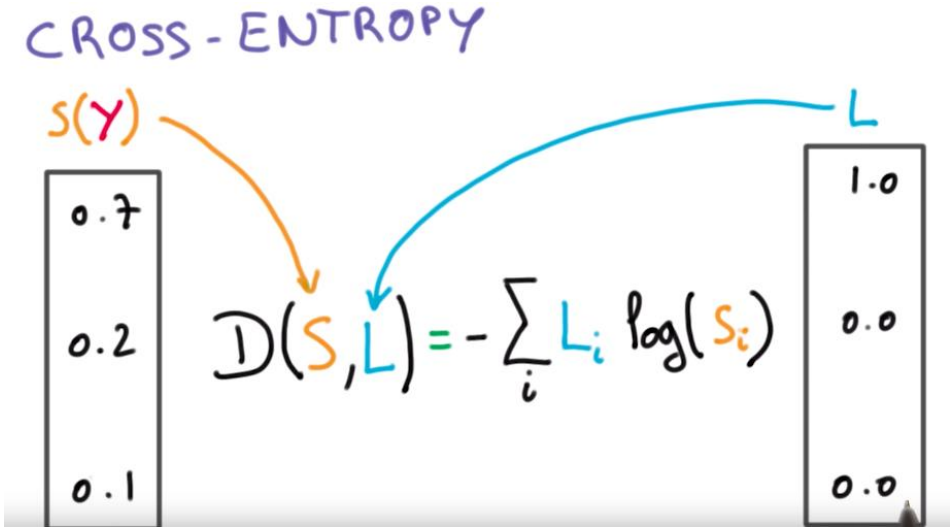
This is a math joke…..

https://www.slideserve.com/phong/infinite-limits

Loss Function

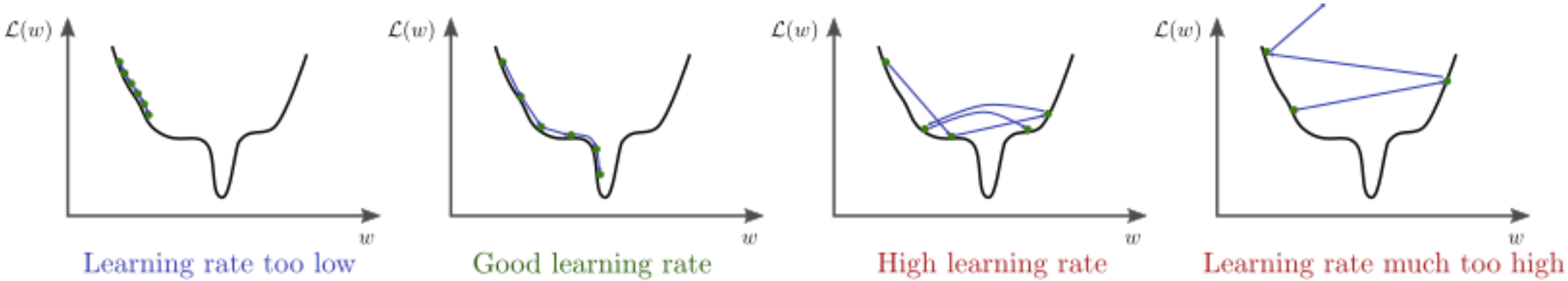


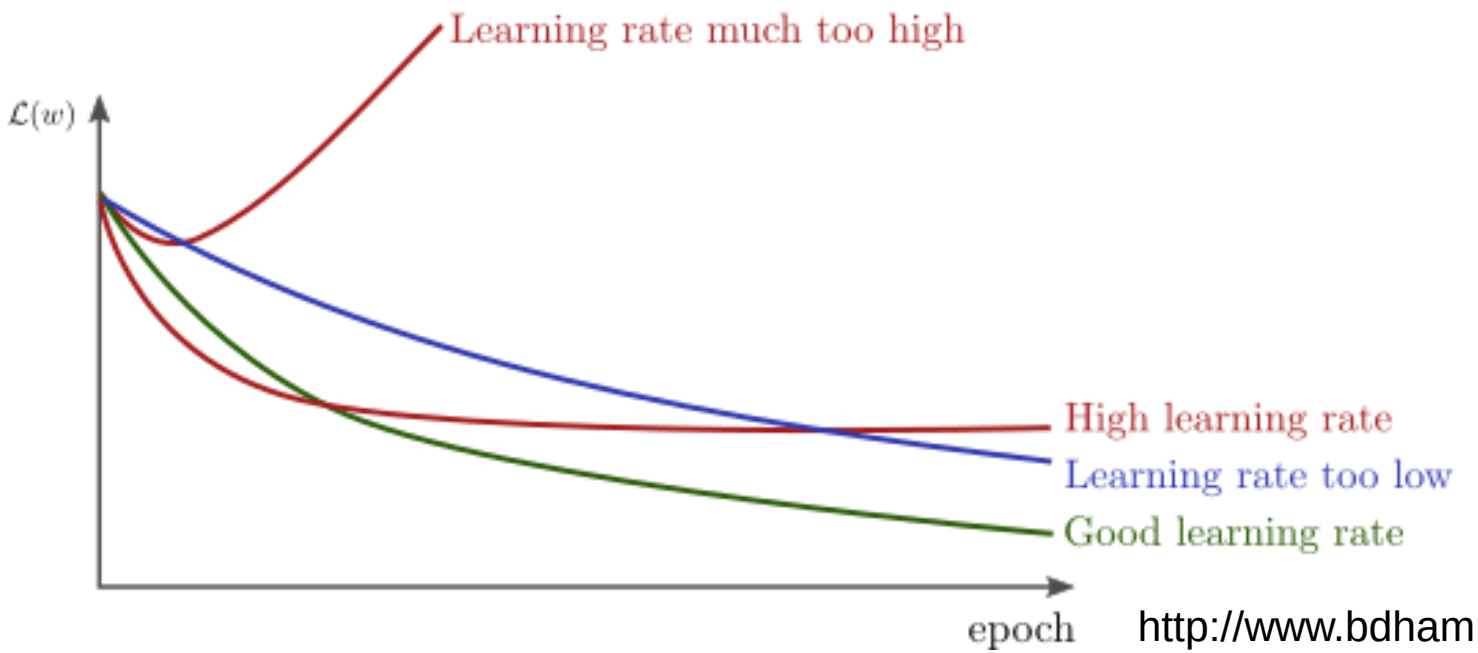
- **MSE**
  - Strongly penalizes outliers
  - High sensitivity near minimum
- **MAE**
  - Scales linearly with size of error
  - Low sensitivity near minimum
- **Huber**
  - Similar to MSE near minimum
  - Similar to MAE away from minimum



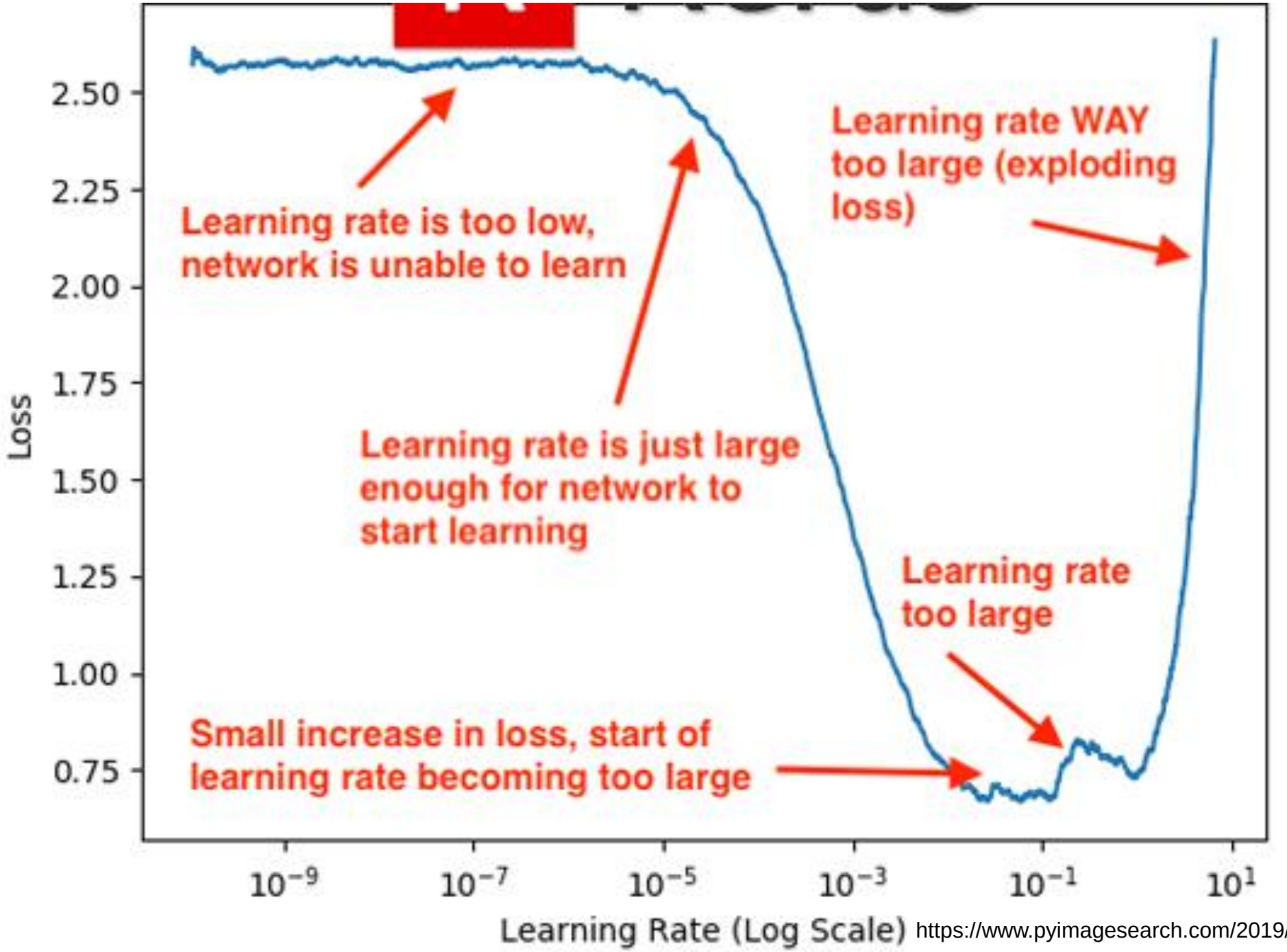
- What if there exists an unavoidable bias in your data?
- What if the given loss function fails to reproduce some important physical properties?
- What if the DL structure is too complicated?

Learning Rate

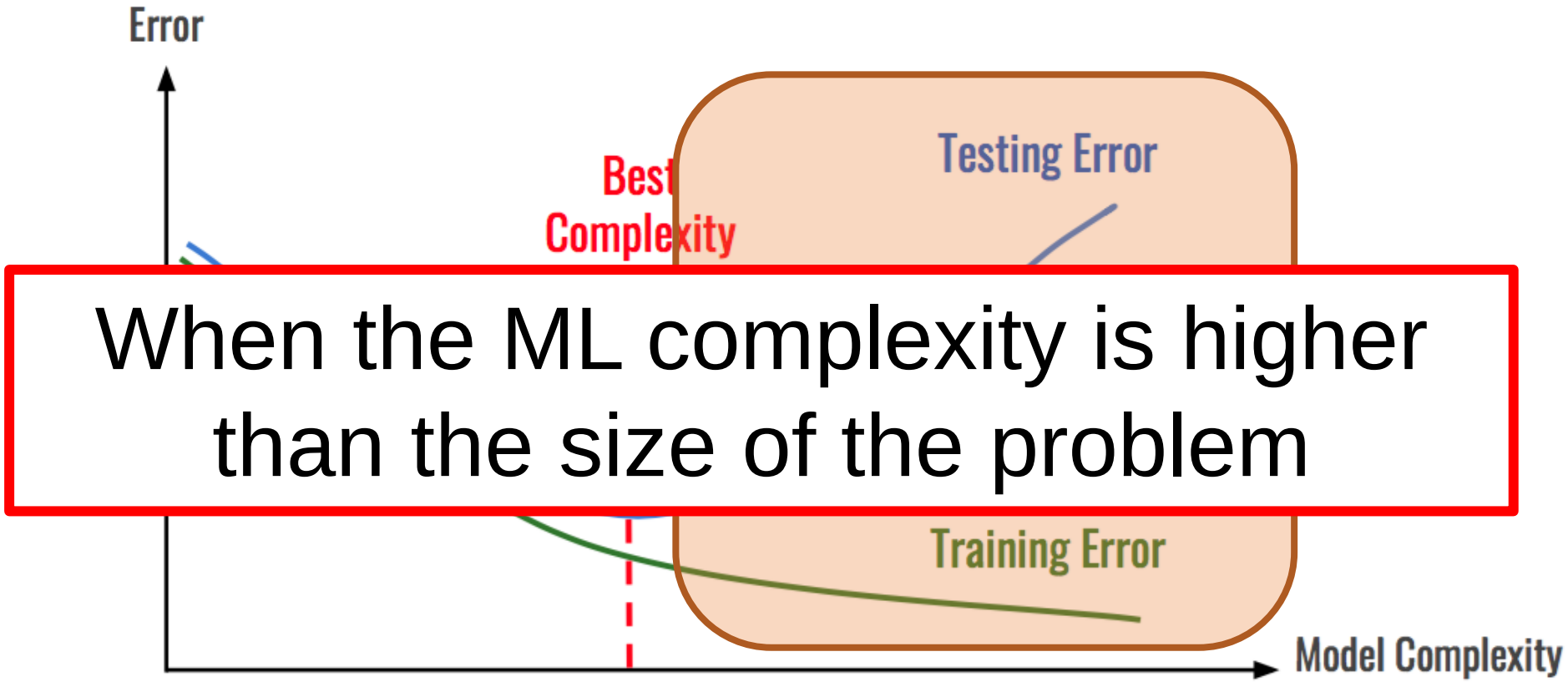




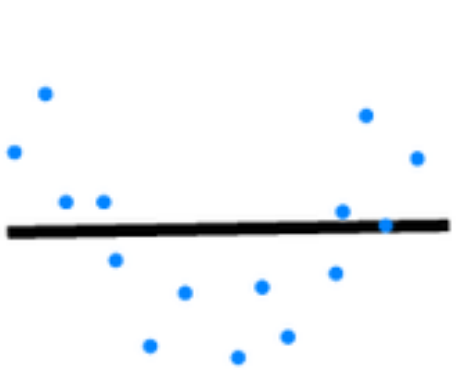
http://www.bdhammel.com/learning-rates/



https://www.pyimagesearch.com/2019/08/05/keras-learning-rate-finder/



When the ML complexity is higher than the size of the problem

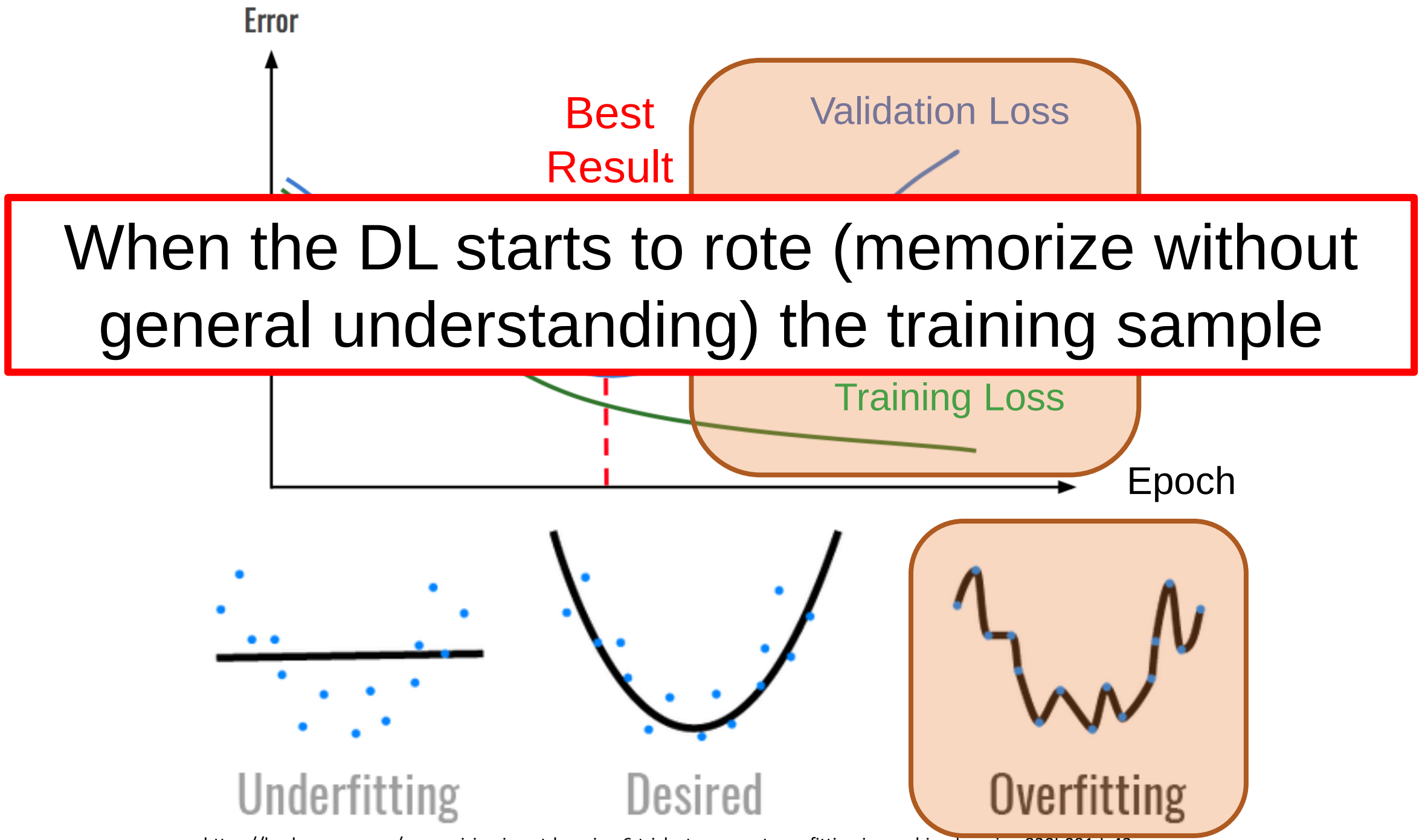


Underfitting

Desired

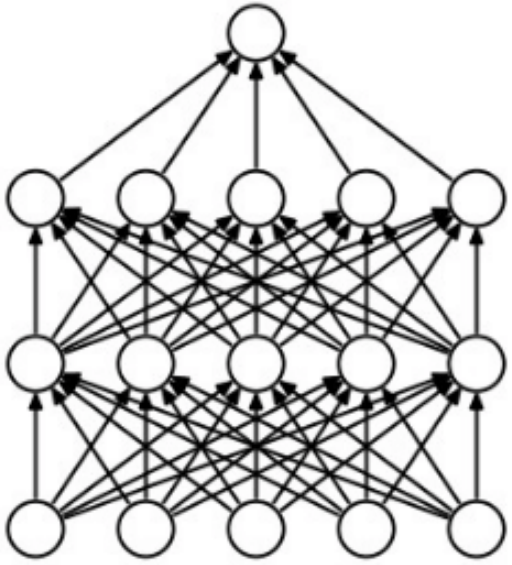
Overfitting

https://hackernoon.com/memorizing-is-not-learning-6-tricks-to-prevent-overfitting-in-machine-learning-820b091dc42

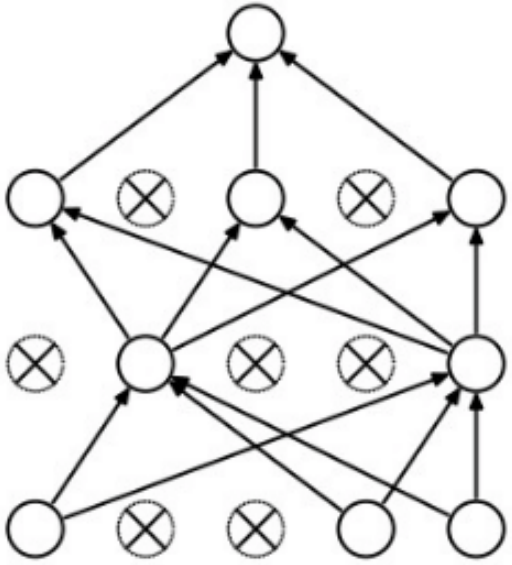


When the DL starts to rote (memorize without general understanding) the training sample

https://hackernoon.com/memorizing-is-not-learning-6-tricks-to-prevent-overfitting-in-machine-learning-820b091dc42



(a) Standard Neural Net



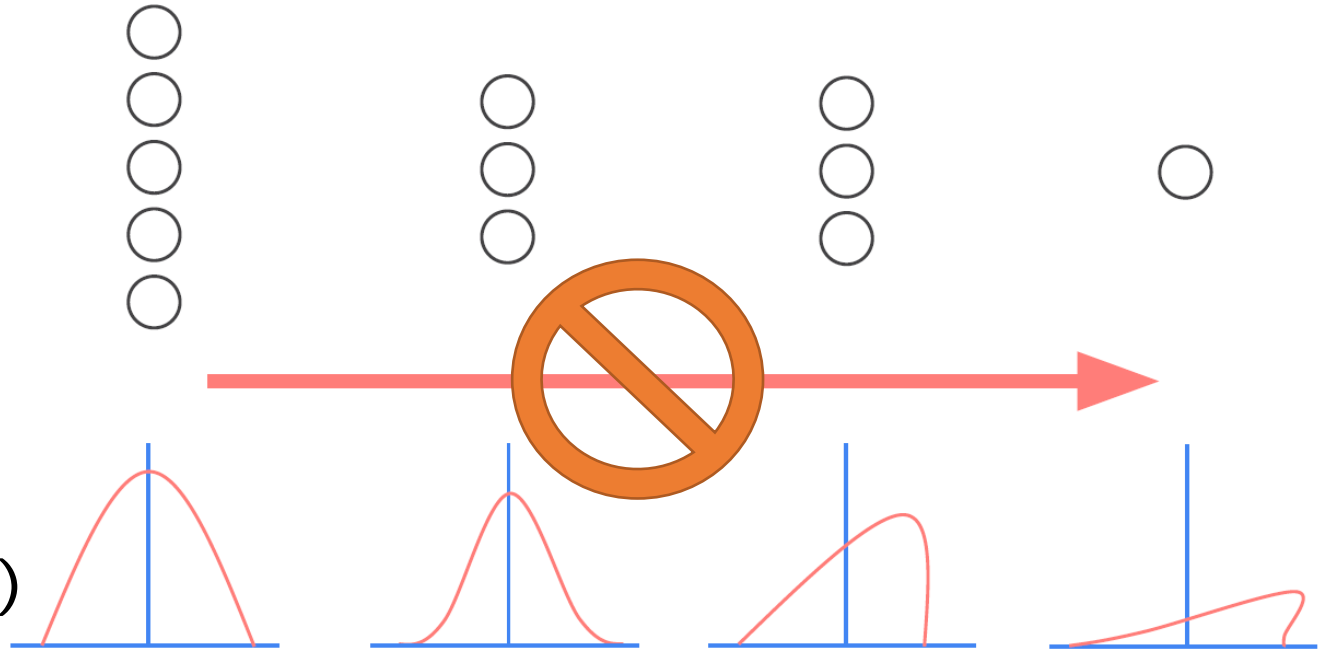
(b) After applying dropout.

Dropout

`torch.nn.Dropout(`*rate*`)`

Batch Normalization

`torch.nn.BatchNorm1d(`*num_features*`)`

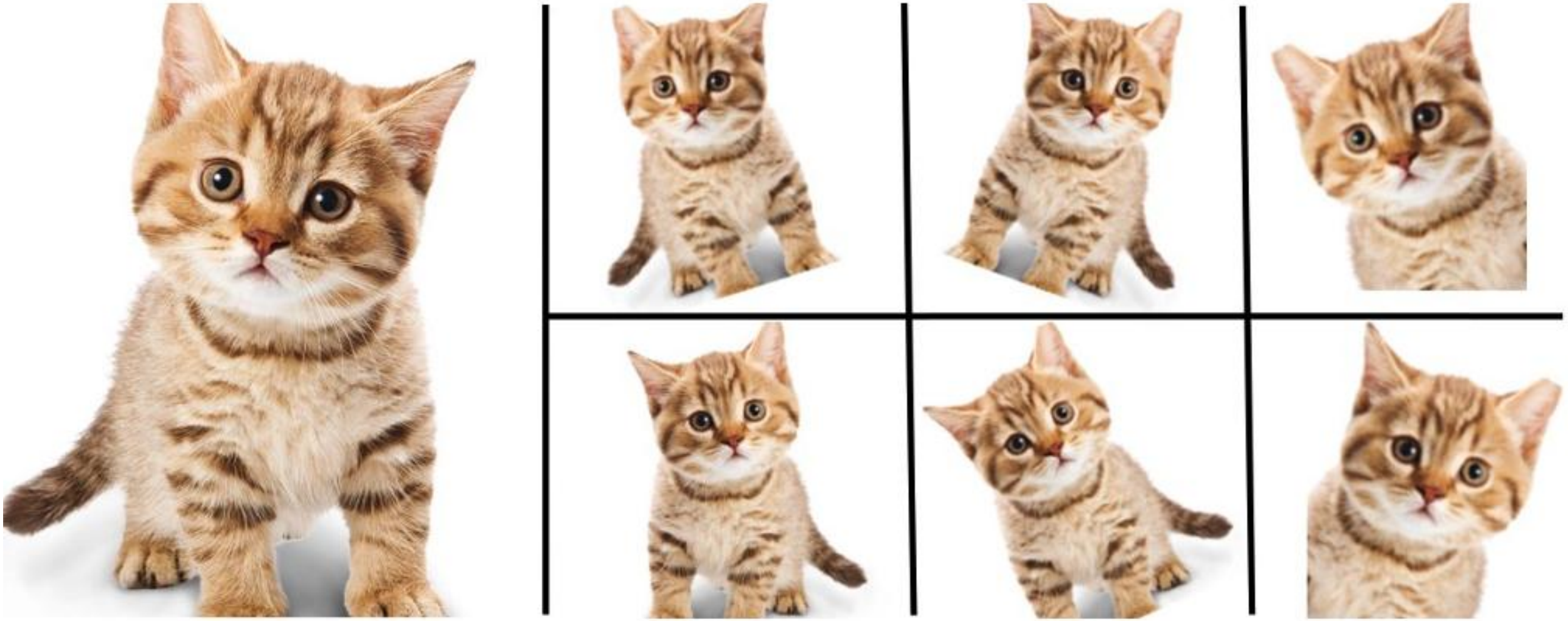


多多益善

The more, the better

Paik Nam June (1987)

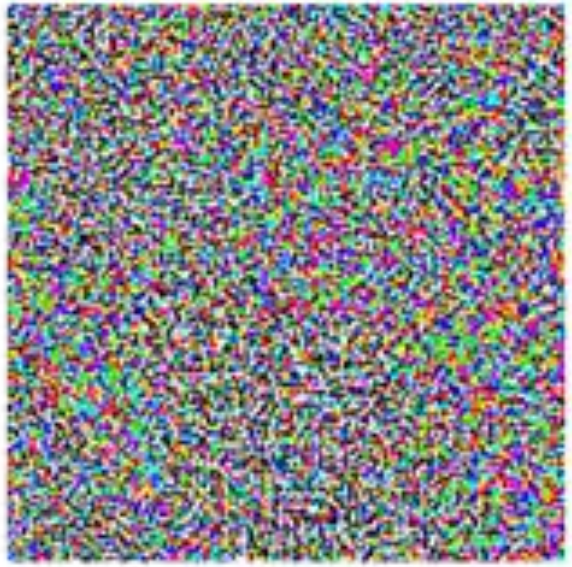
Data Augmentation



https://medium.com/nanonets/how-to-use-deep-learning-when-you-have-limited-data-part-2-data-augmentation-c26971dc8ced

https://pyimagesearch.com/2020/10/19/adversarial-images-and-attacks-with-keras-and-tensorflow/

$+ .007 \times$



$=$

$\boldsymbol{x}$

"panda"
57.7% confidence

$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$

"nematode"
8.2% confidence

$\boldsymbol{x} + \epsilon\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$

"gibbon"
99.3 % confidence

Adding noise may be beneficial for making ML/DL
to concentrate only on what's important
and forget about what's not.