



HAL
open science

Data analysis with artificial intelligence

Julio Cesar da Silva

► **To cite this version:**

Julio Cesar da Silva. Data analysis with artificial intelligence. École thématique. Ecole CohereX - Science with coherent X-rays at 3rd and 4th generation synchrotron sources, Cargèse (CNRS), France. 2023. ⟨hal-04783980⟩

HAL Id: hal-04783980

<https://hal.science/hal-04783980v1>

Submitted on 14 Nov 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



HAL Authorization

Data analysis with artificial intelligence

Julio Cesar da Silva

CNRS NEEL Institute, Grenoble, France

French CRG Beamline FAME-PIX - ESRF, Grenoble, France

julio-cesar.da-silva@neel.cnrs.fr





612 Superfast

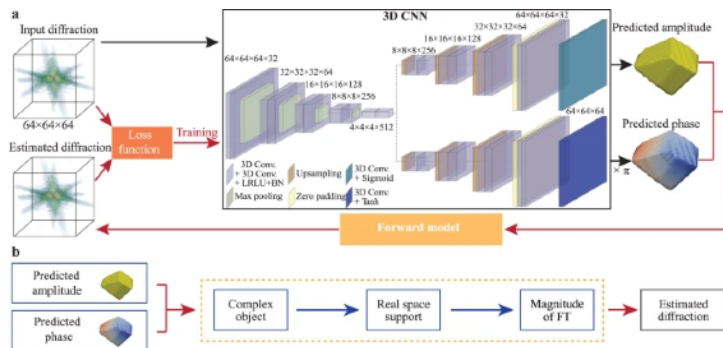
0 1 2 3 4 5 6 7 8 9 10
P R N D
vitesse km/h

FERRARI



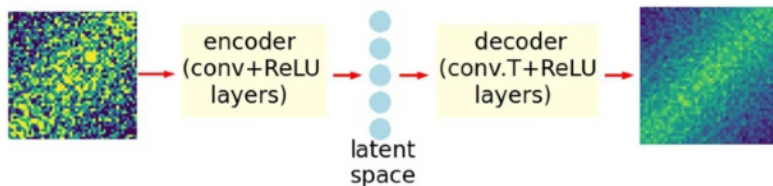
I WANT YOU

AutoPhaseNN for Bragg CDI



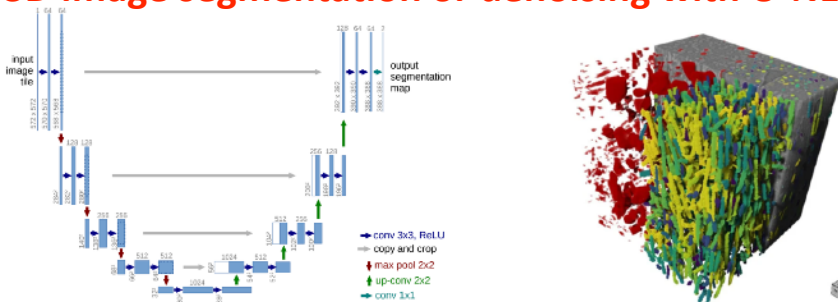
Y. Yao *et al.*, *npj Comput Mater* 8, 124 (2022)

CNN Encoder-Decoder for XPCS



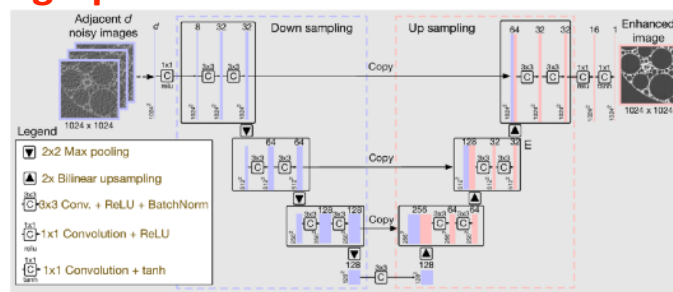
T. Konstantinova *et al.* *Sci Rep* 11, 14756 (2021)

3D image segmentation or denoising with U-NET

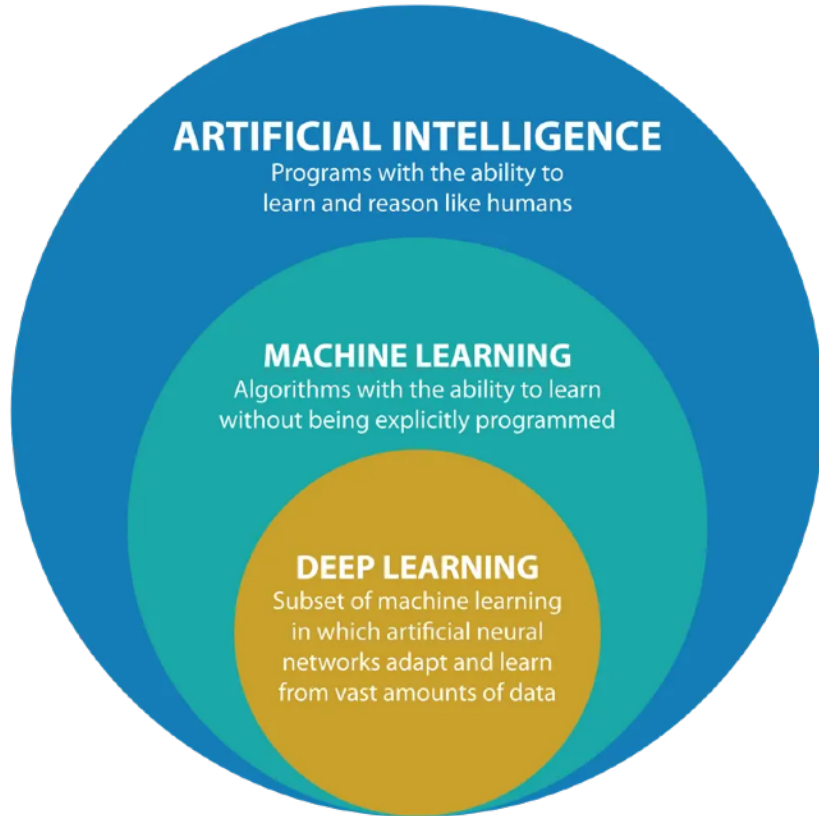


O. Ronneberger *et al.* (2015). *MICCAI 2015. Lecture Notes in Computer Science*, vol 9351. Springer

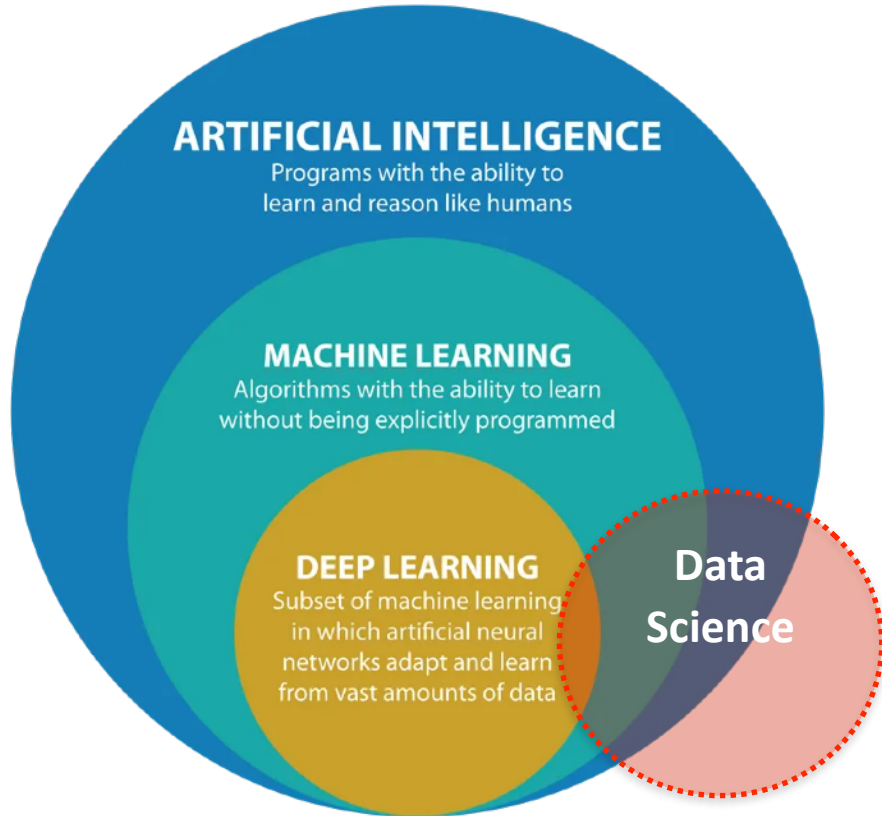
Tomographic Reconstruction with TomoGAN



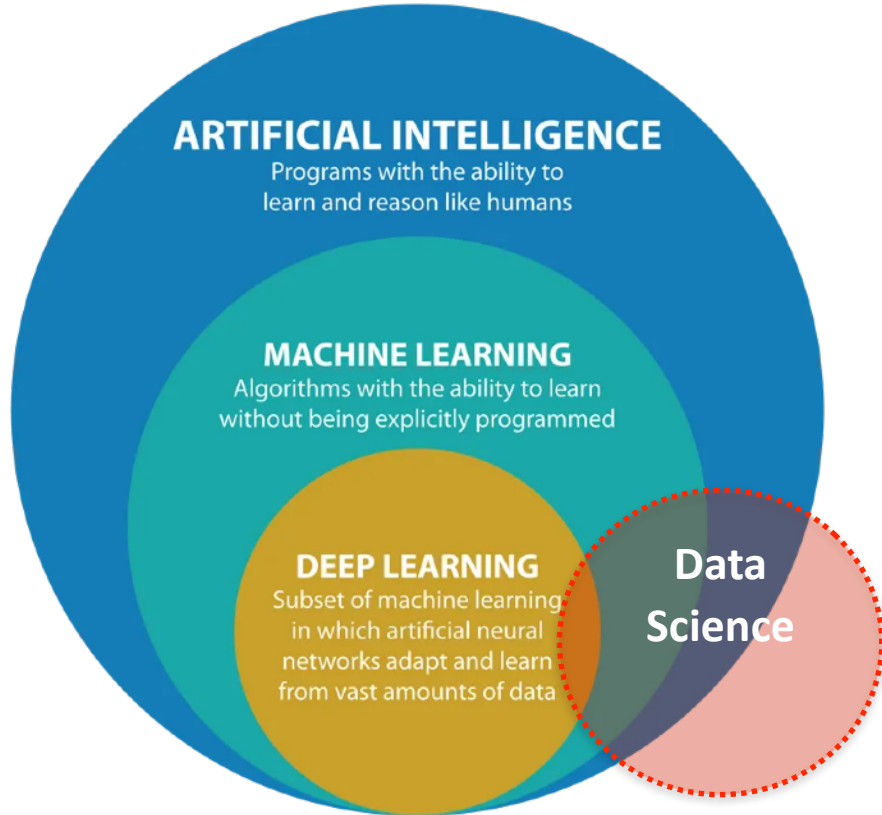
Z. Liu, *et al.* *J. Opt. Soc. Am. A* 37, 422-434 (2020)



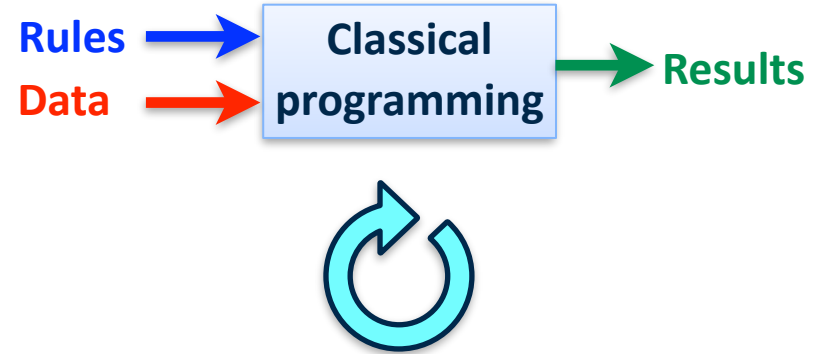
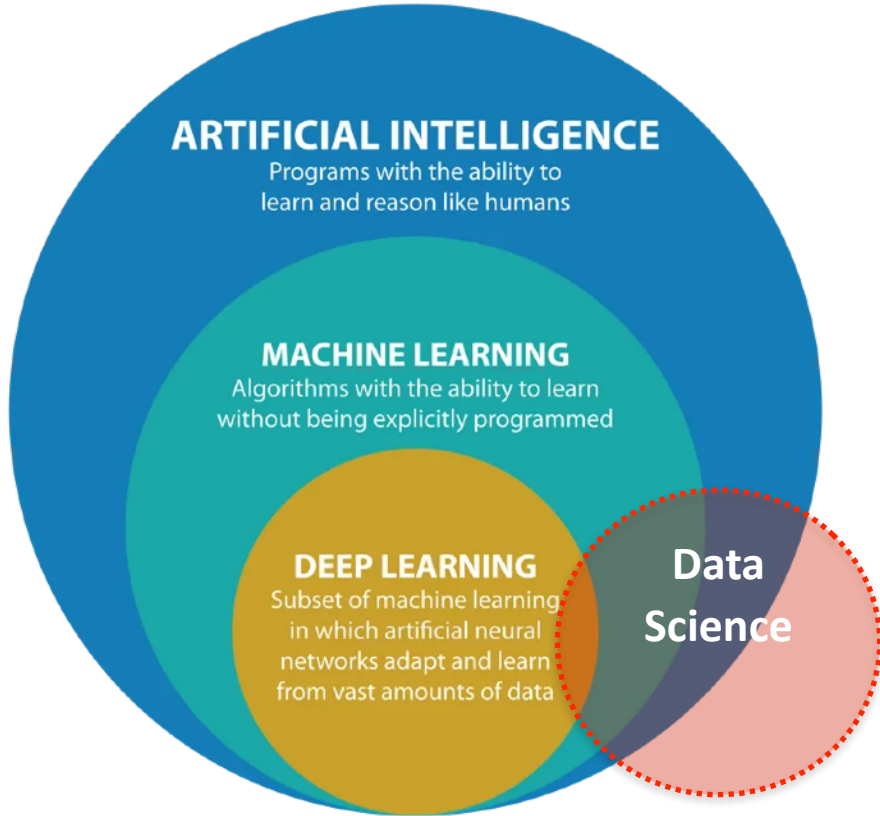
Deep Learning with Python, François Chollet, 2nd edition



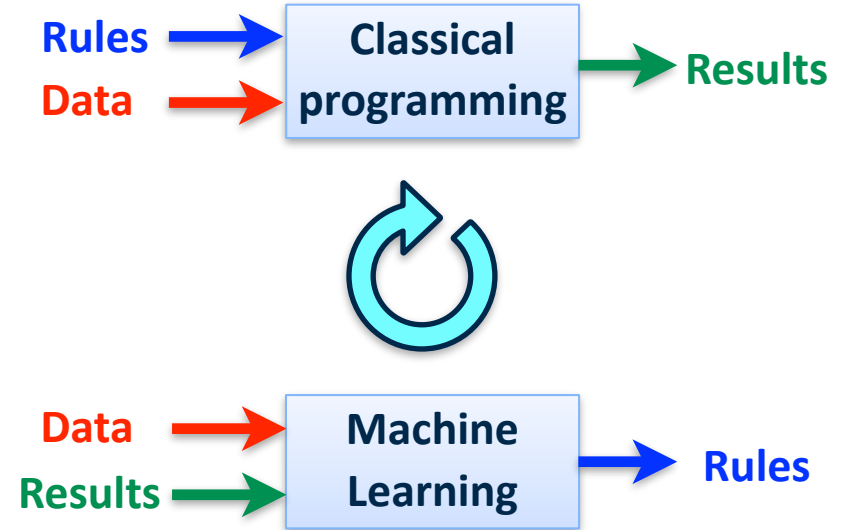
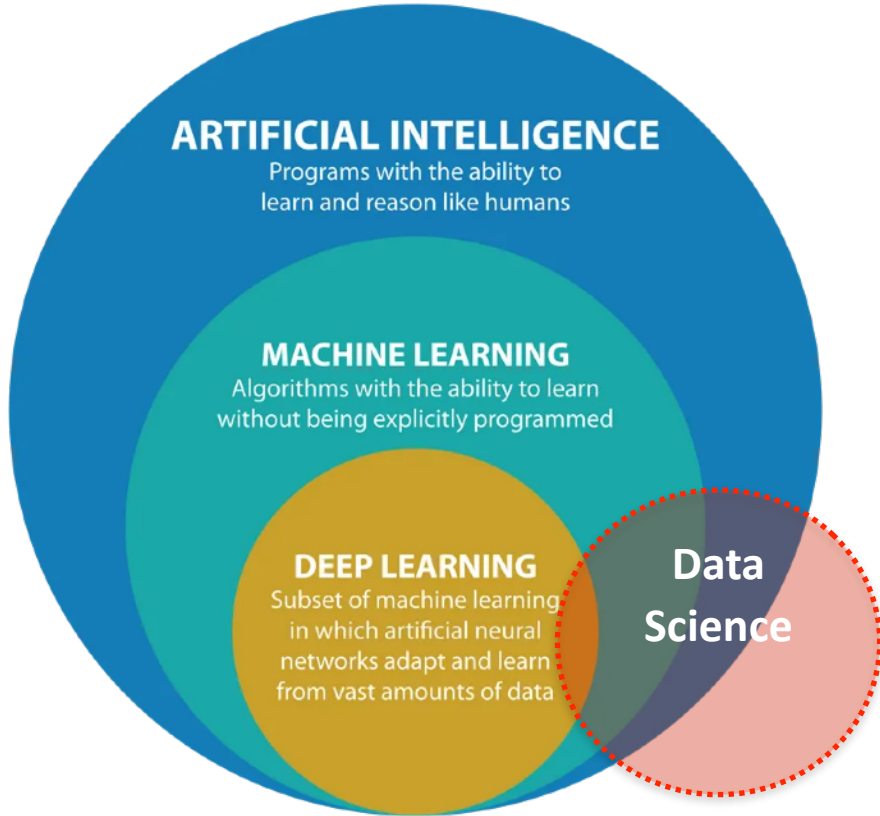
Deep Learning with Python, François Chollet, 2nd edition



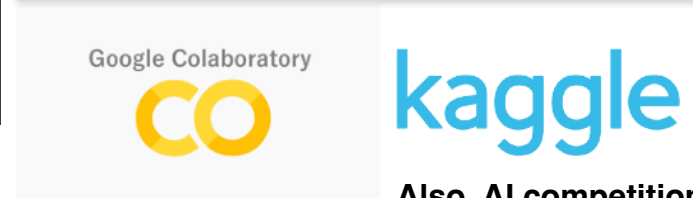
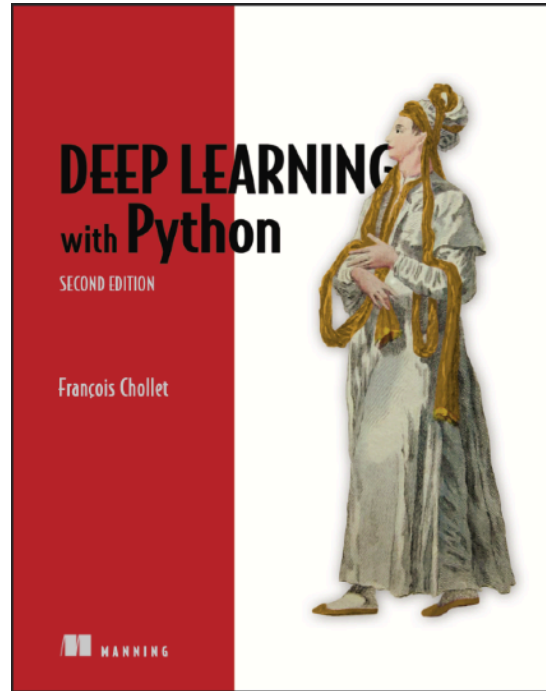
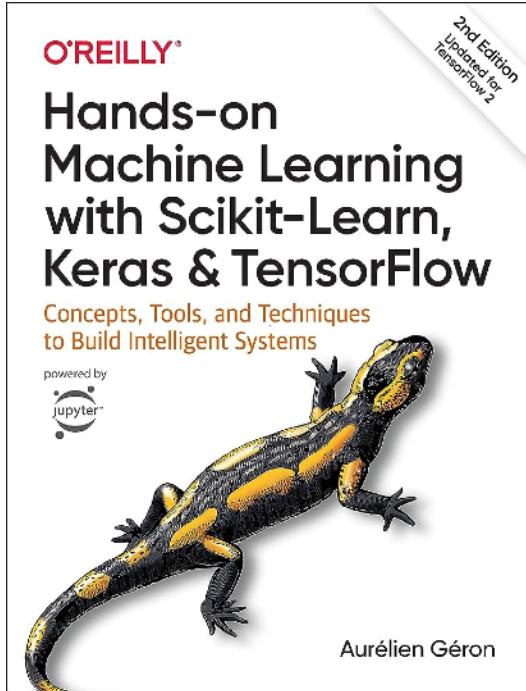
Deep Learning with Python, François Chollet, 2nd edition

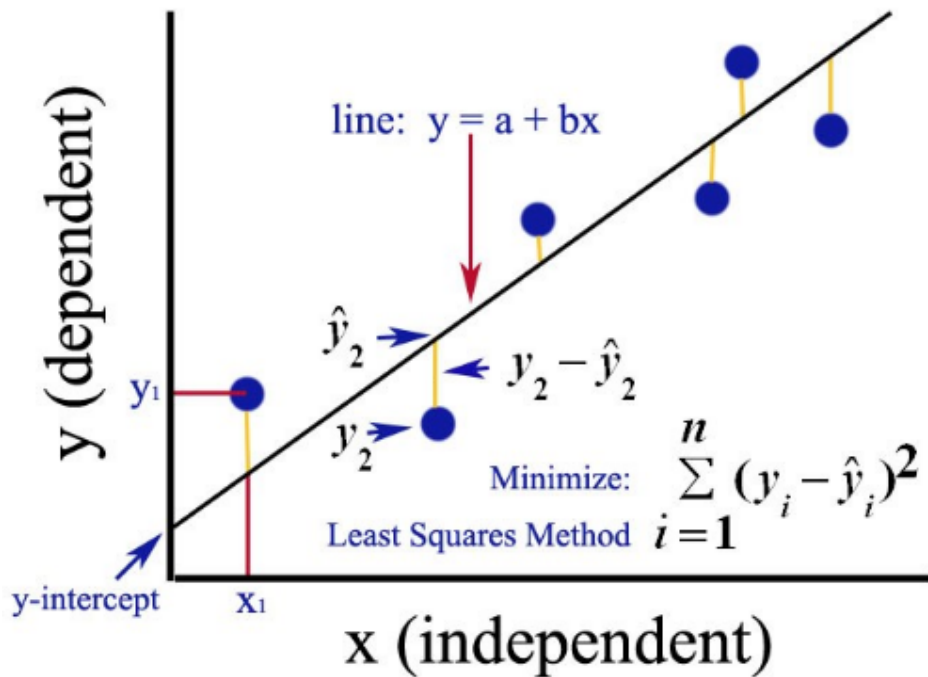


Deep Learning with Python, François Chollet, 2nd edition

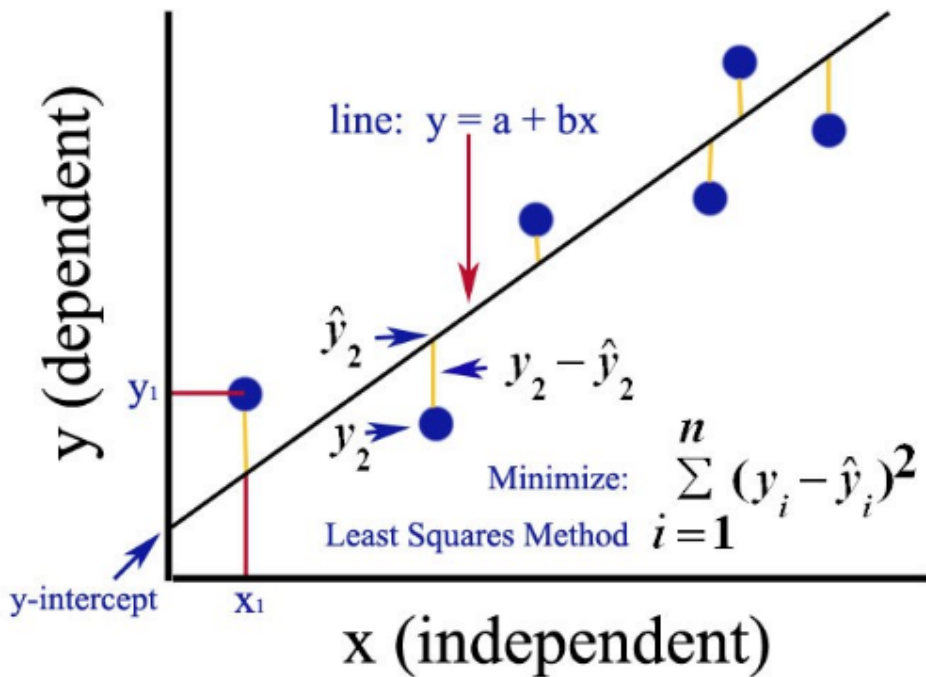


Deep Learning with Python, François Chollet, 2nd edition



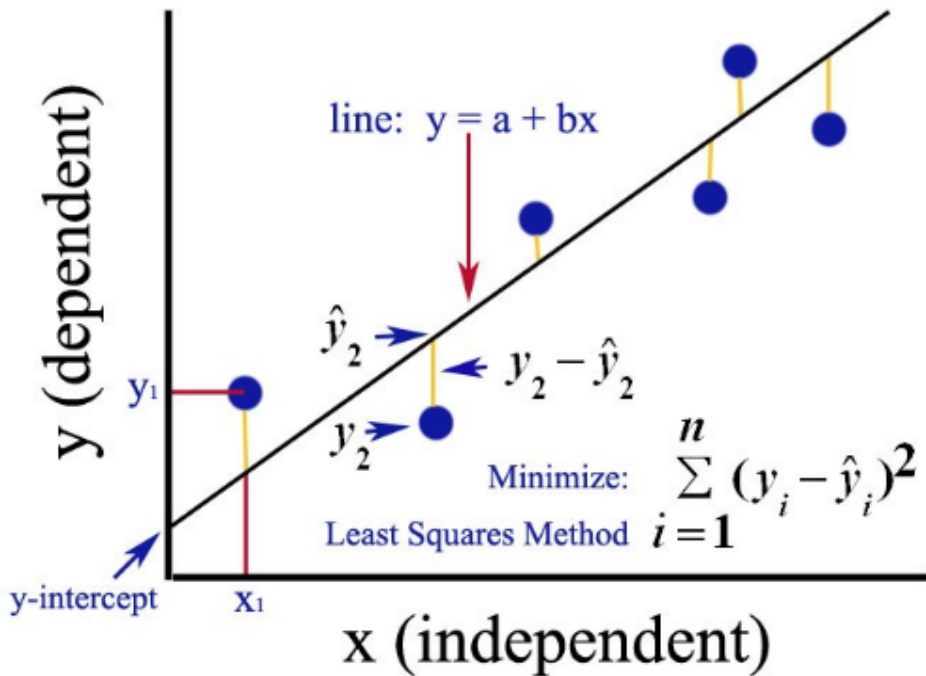


Hands-on Machine Learning with Scikit-learn, Keras & Tensorflow, Aurélien Géron, 2nd edition

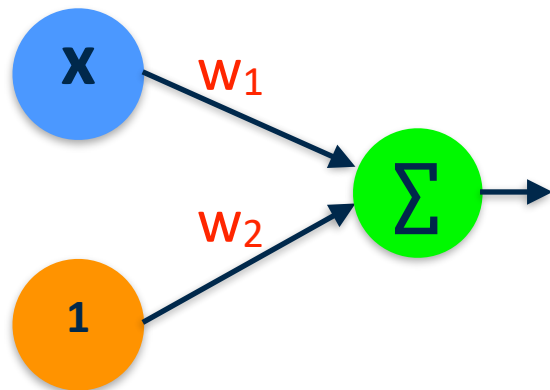


Let's just rethink it in this way:

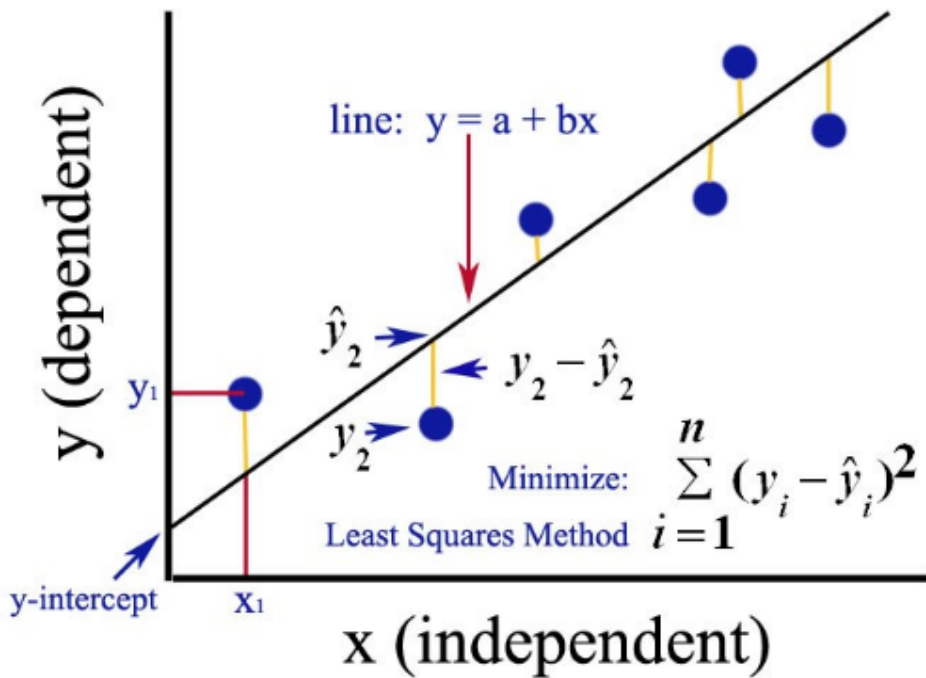
Hands-on Machine Learning with Scikit-learn, Keras & Tensorflow, Aurélien Géron, 2nd edition



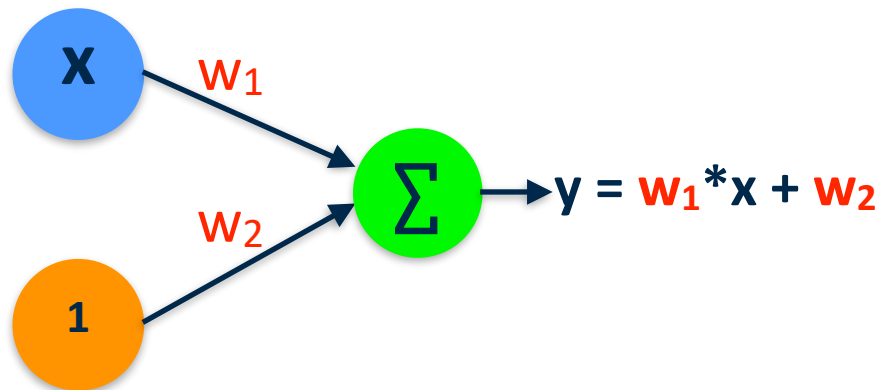
Let's just rethink it in this way:



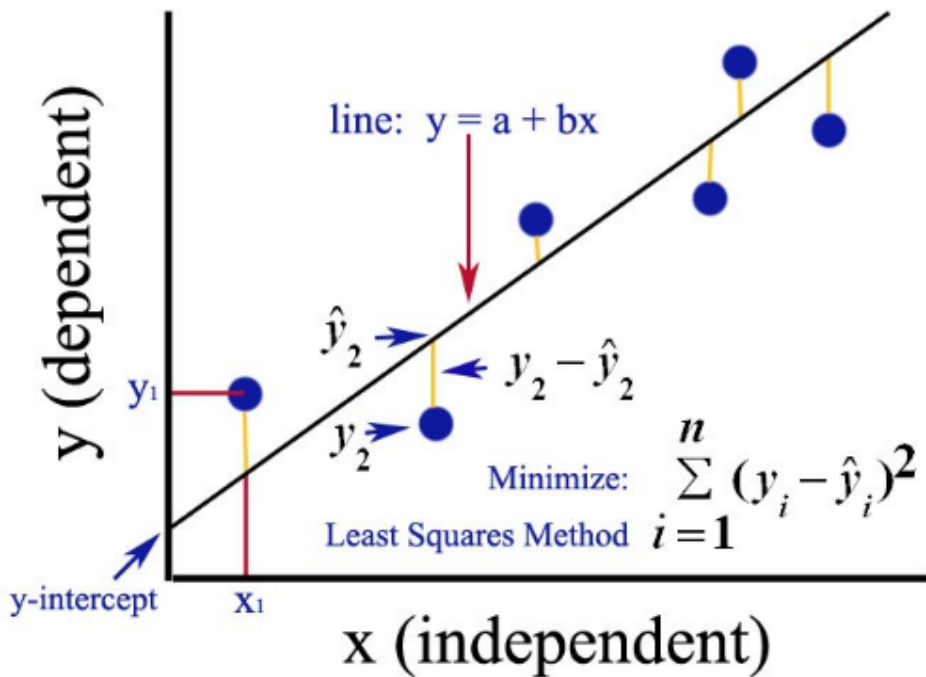
Hands-on Machine Learning with Scikit-learn, Keras & Tensorflow, Aurélien Géron, 2nd edition



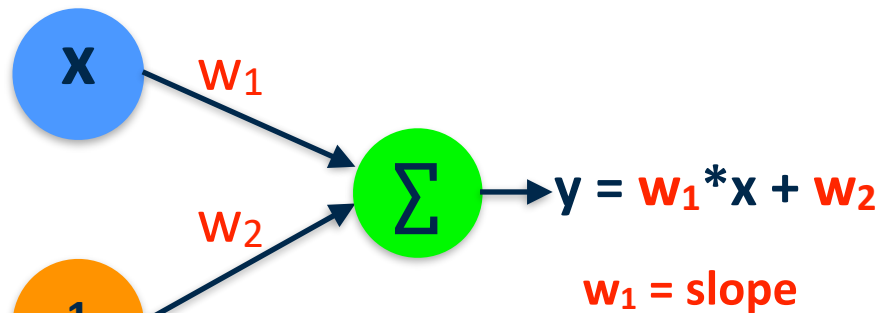
Let's just rethink it in this way:



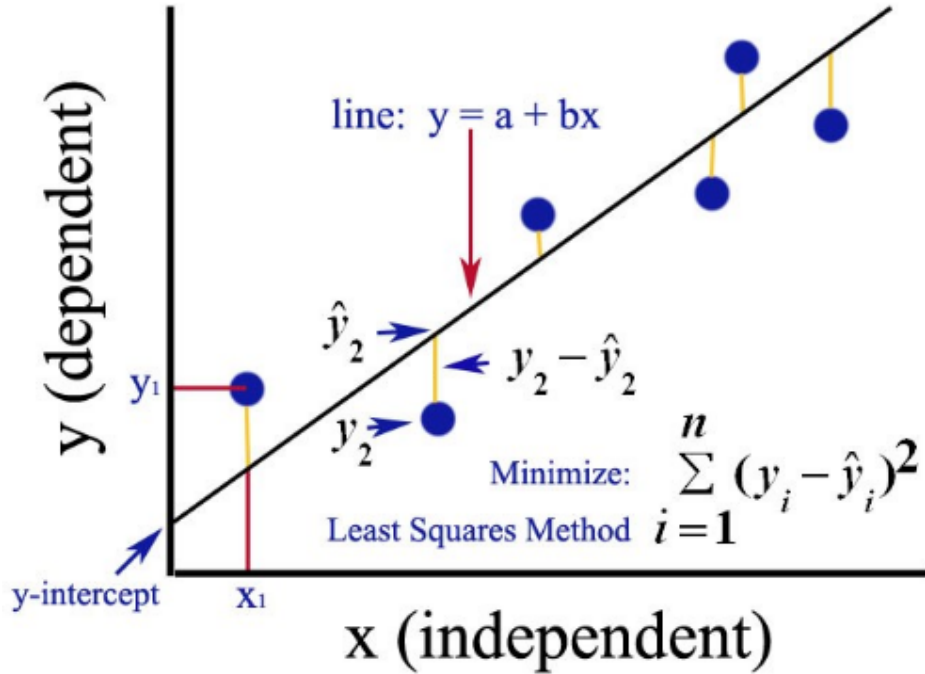
Hands-on Machine Learning with Scikit-learn, Keras & Tensorflow, Aurélien Géron, 2nd edition



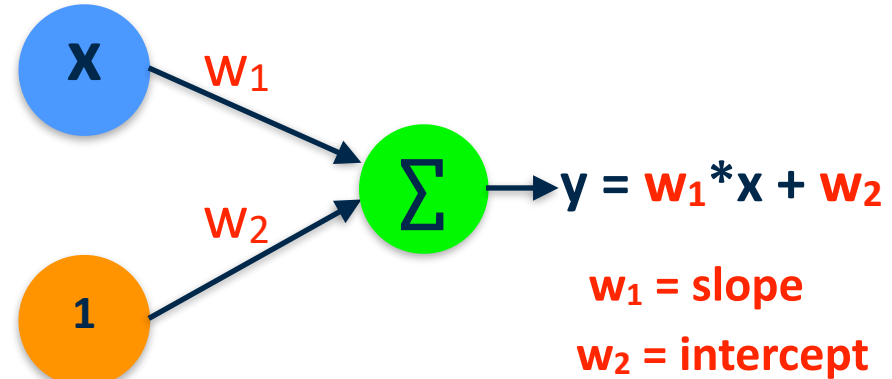
Let's just rethink it in this way:



Hands-on Machine Learning with Scikit-learn, Keras & Tensorflow, Aurélien Géron, 2nd edition



Let's just rethink it in this way:



Hands-on Machine Learning with Scikit-learn, Keras & Tensorflow, Aurélien Géron, 2nd edition

Jupyter notebook Linear models

Unsupervised learning

Clustering

Hierarchical



K-means



Dimensional reduction

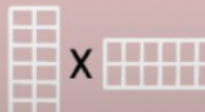
PCA



tSNE



NMF



Supervised Learning

"Machine Learning"

SVM



KNN



Regression



Random forest

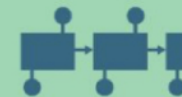


"Deep Learning"

CNN



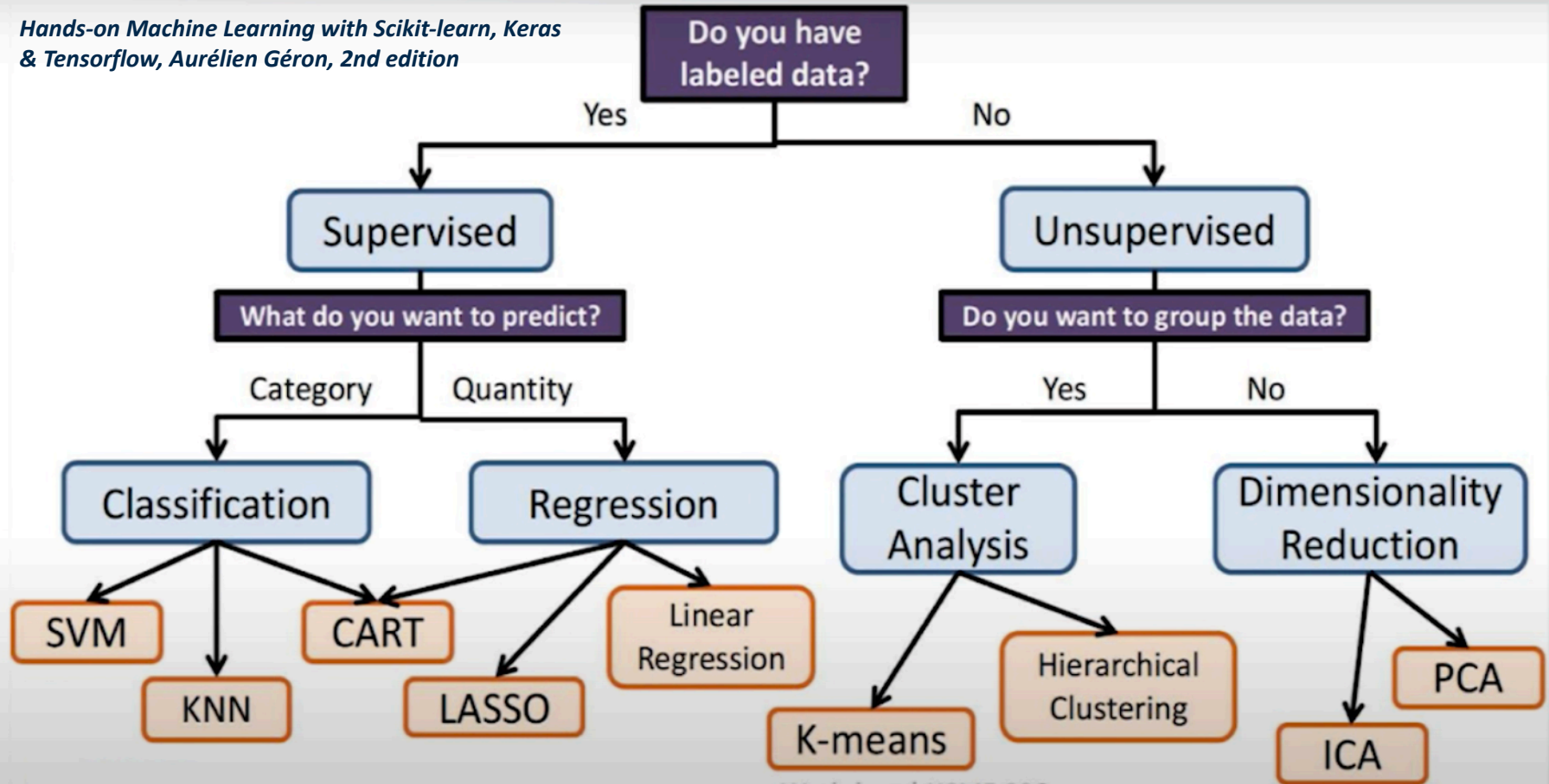
RNN



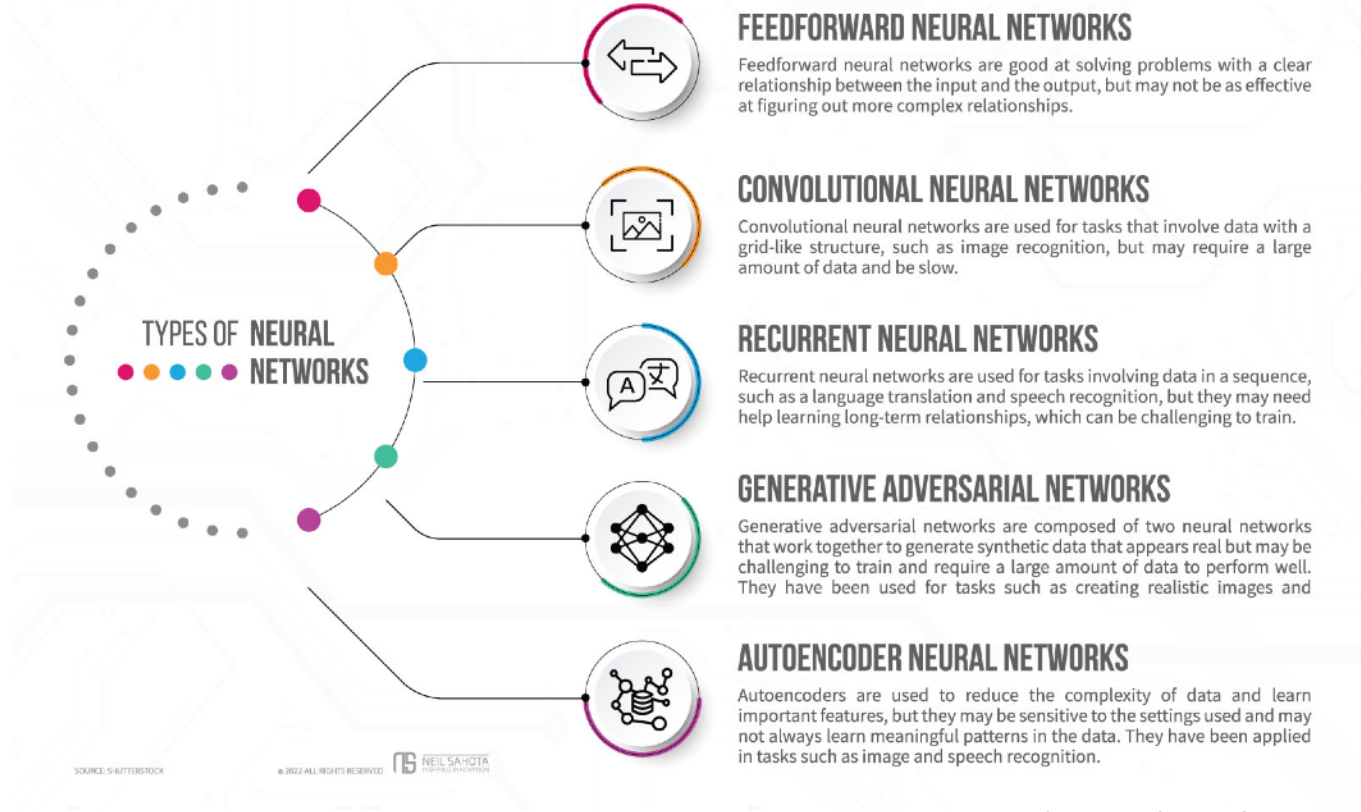
Hands-on Machine Learning with Scikit-learn, Keras & Tensorflow, Aurélien Géron, 2nd edition

The (« non-deep") machine-learning eco-system

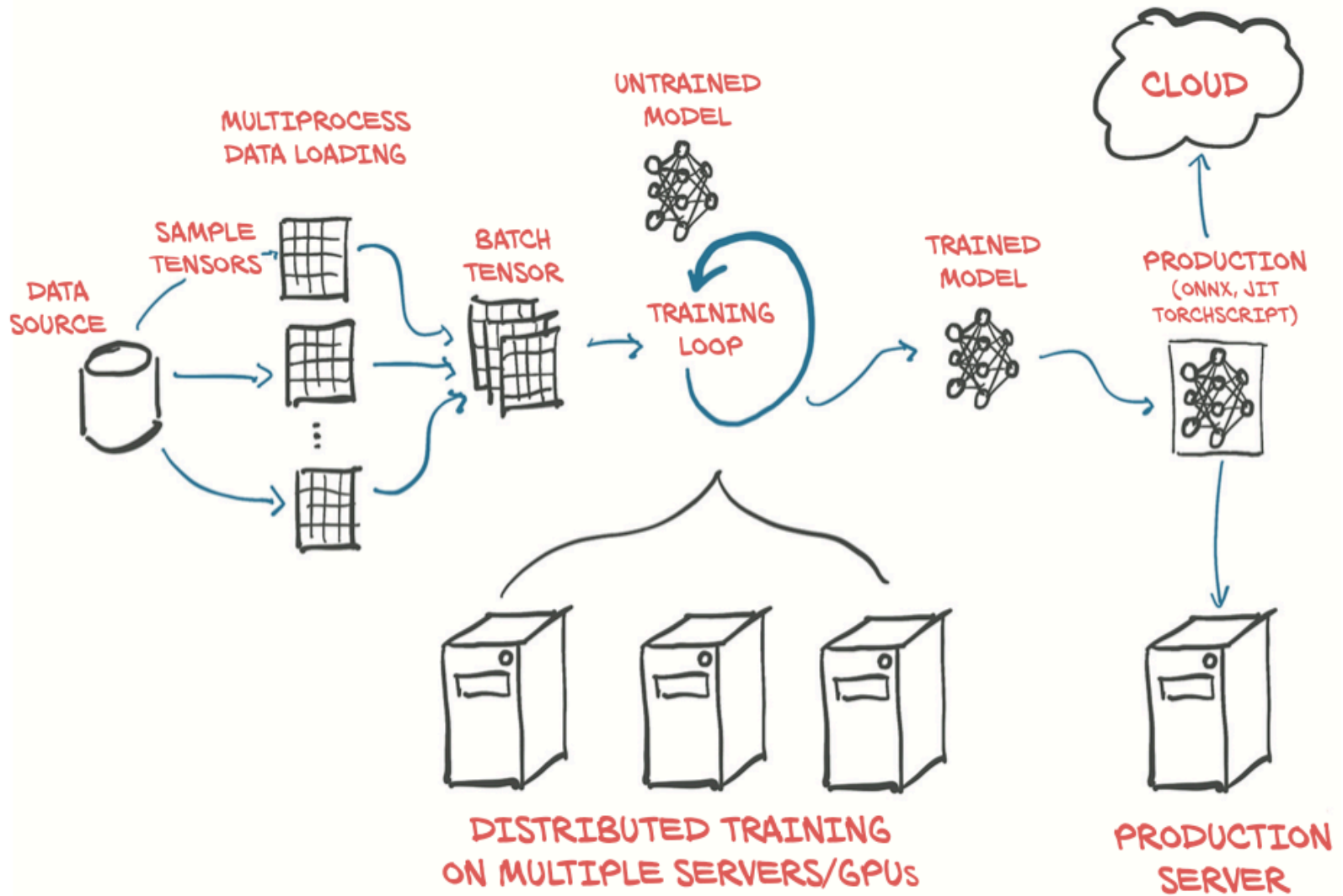
Hands-on Machine Learning with Scikit-learn, Keras & TensorFlow, Aurélien Géron, 2nd edition



Main types of Deep Learning networks



Hands-on Machine Learning with Scikit-learn, Keras & Tensorflow, Aurélien Géron, 2nd edition



Deep Learning with Python, François Chollet, 2nd edition



The laws of Machine Learning



Deep Learning with Python, François Chollet, 2nd edition



1) Only use Machine Learning if the classical approaches fail to solve your problems.



- 1) Only use Machine Learning if the classical approaches fail to solve your problems.
- 2) A neural network can only produce outputs that resemble the data it was trained on.



- 1) Only use Machine Learning if the classical approaches fail to solve your problems.
- 2) A neural network can only produce outputs that resemble the data it was trained on.
- 3) Data preparation is the most important part of the machine and deep learning approaches.

Deep Learning with Python, François Chollet, 2nd edition

Neuron

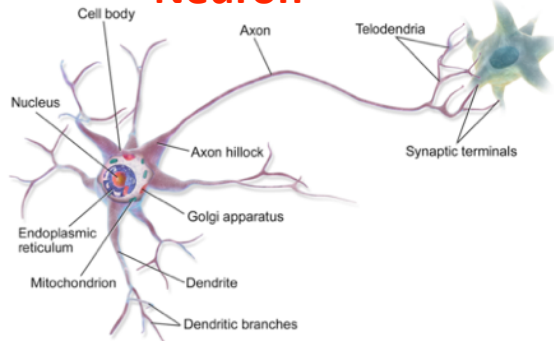
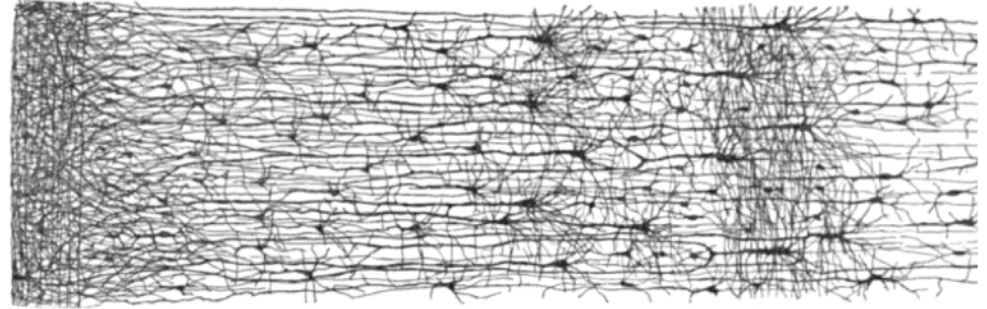


Image by Bruce Blaus (Creative Commons 3.0)
<https://en.wikipedia.org/wiki/Neuron>.

Biological neural network



Drawing of a cortical lamination by S. Ramon y Cajal (public domain).
https://en.wikipedia.org/wiki/Cerebral_cortex

Neuron

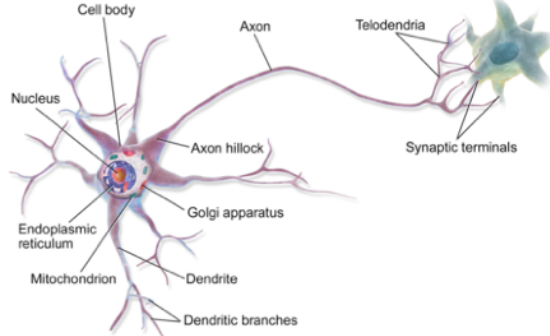
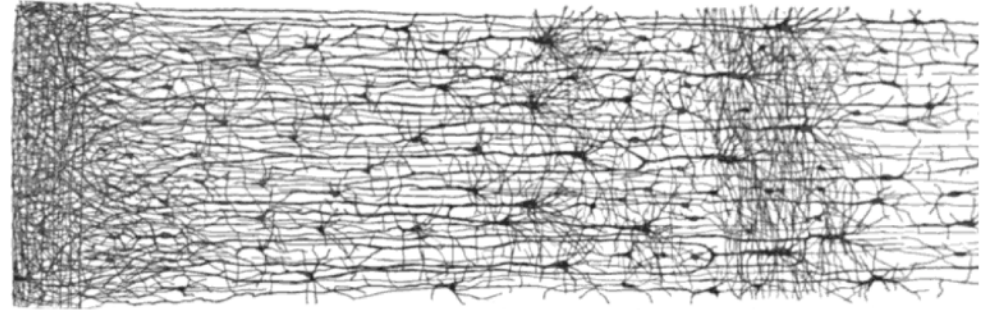


Image by Bruce Blaus (Creative Commons 3.0)
<https://en.wikipedia.org/wiki/Neuron>.

Biological neural network

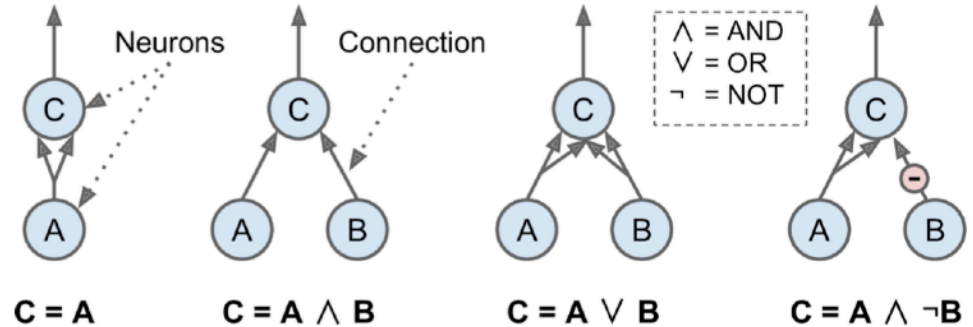


Drawing of a cortical lamination by S. Ramon y Cajal (public domain).
https://en.wikipedia.org/wiki/Cerebral_cortex

McCulloch and Walter Pitts proposed a model for biological neurons, which later became **artificial neurons**.

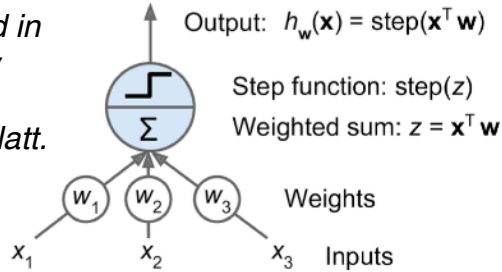
McCulloch, Warren S.; Pitts, Walter (1943). *Bulletin of Mathematical Biophysics*. 5 (4): 115–133.

ANNs performing simple logical computations



Threshold logic unit (TLU)

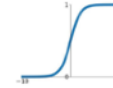
*invented in
1957 by
Frank
Rosenblatt.*



Activation Functions

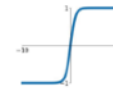
Sigmoid

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



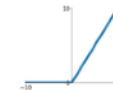
tanh

$$\tanh(x)$$



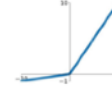
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

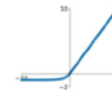


Maxout

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

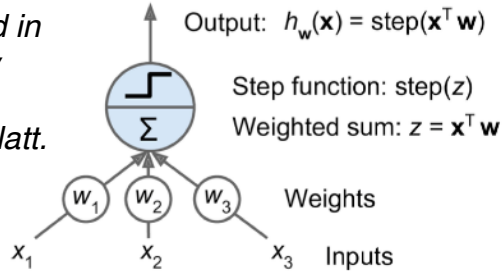
ELU

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



Threshold logic unit (TLU)

invented in 1957 by Frank Rosenblatt.

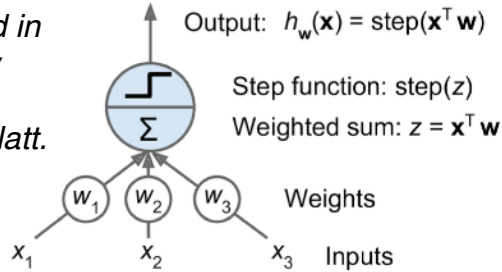


Activation Functions

<p>Sigmoid</p> $y = \frac{1}{1+e^{-x}}$	<p>Tanh</p> $y = \tanh(x)$	<p>Step Function</p> $y = \begin{cases} 0, & x < n \\ 1, & x > n \end{cases}$	<p>Softplus</p> $y = \ln(1+e^x)$
<p>ReLU</p> $y = \begin{cases} 0, & x < 0 \\ x, & x > 0 \end{cases}$	<p>Softsign</p> $y = \frac{x}{(1+ x)}$	<p>ELU</p> $y = \begin{cases} \alpha(e^x-1), & x < 0 \\ x, & x > 0 \end{cases}$	<p>Log of Sigmoid</p> $y = \ln\left(\frac{1}{1+e^{-x}}\right)$
<p>Swish</p> $y = \frac{x}{1+e^{-x}}$	<p>Sinc</p> $y = \frac{\sin(x)}{x}$	<p>Leaky ReLU</p> $y = \max(0.01x, x)$	<p>Mish</p> $y = x(\tanh(\text{softplus}(x)))$

Threshold logic unit (TLU)

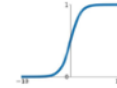
*invented in
1957 by
Frank
Rosenblatt.*



Activation Functions

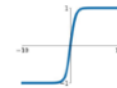
Sigmoid

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



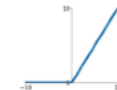
tanh

$$\tanh(x)$$



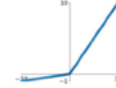
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

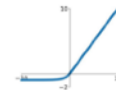


Maxout

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

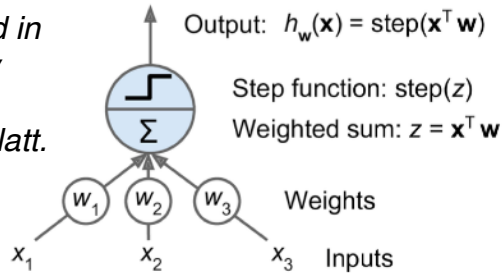
ELU

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



Threshold logic unit (TLU)

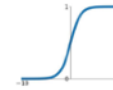
invented in 1957 by Frank Rosenblatt.



Activation Functions

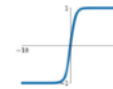
Sigmoid

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



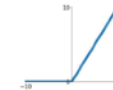
tanh

$$\tanh(x)$$



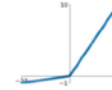
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

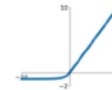


Maxout

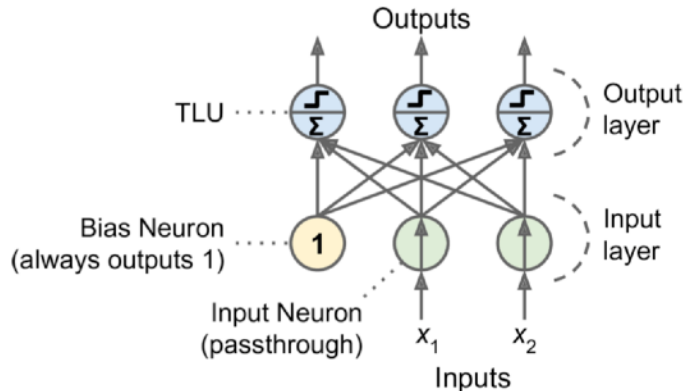
$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

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

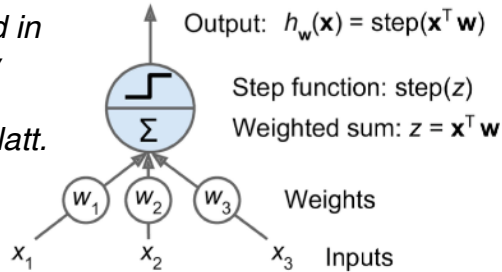


Perceptron (single layer of TLUs)



Threshold logic unit (TLU)

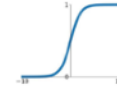
invented in 1957 by Frank Rosenblatt.



Activation Functions

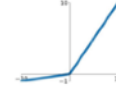
Sigmoid

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



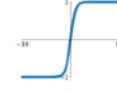
Leaky ReLU

$$\max(0.1x, x)$$



tanh

$$\tanh(x)$$

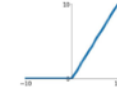


Maxout

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

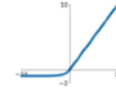
ReLU

$$\max(0, x)$$

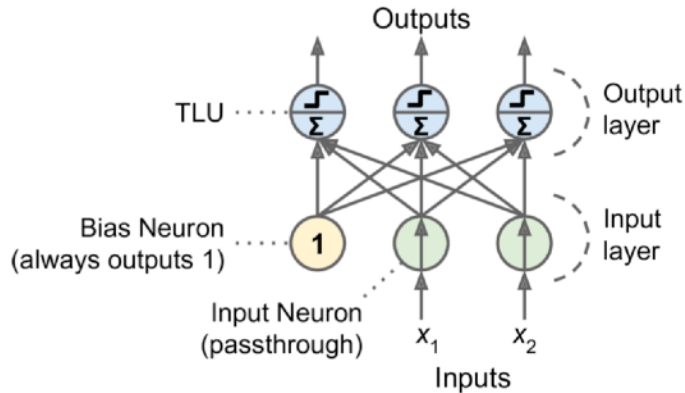


ELU

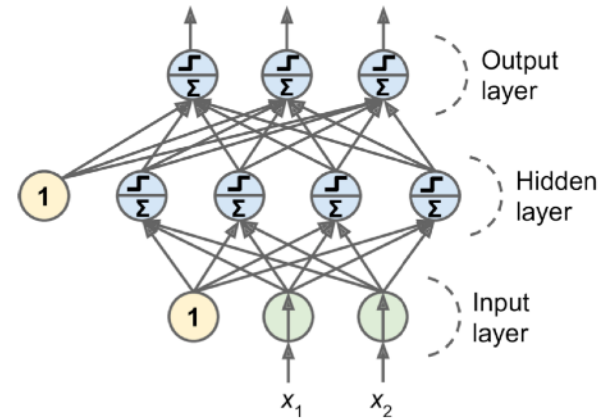
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Perceptron (single layer of TLUs)

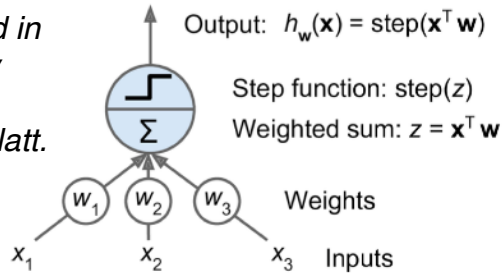


Multi-layer perceptron (deep neural network)



Threshold logic unit (TLU)

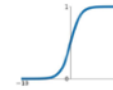
invented in 1957 by Frank Rosenblatt.



Activation Functions

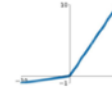
Sigmoid

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



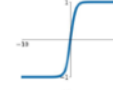
Leaky ReLU

$$\max(0.1x, x)$$



tanh

$$\tanh(x)$$

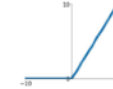


Maxout

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

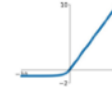
ReLU

$$\max(0, x)$$

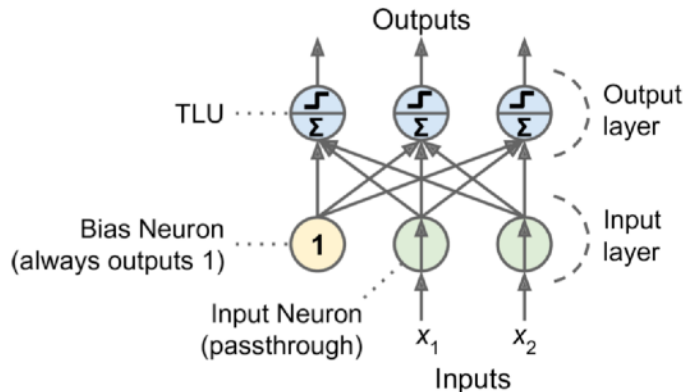


ELU

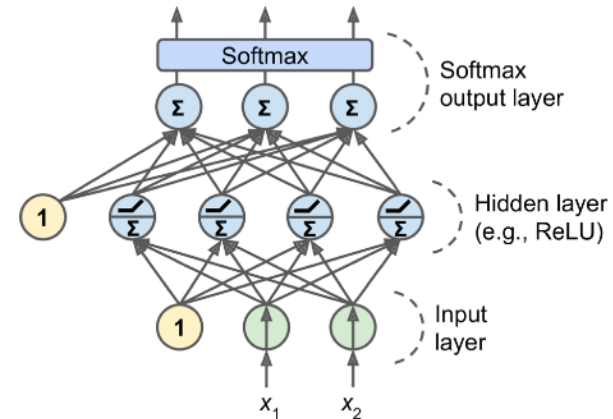
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

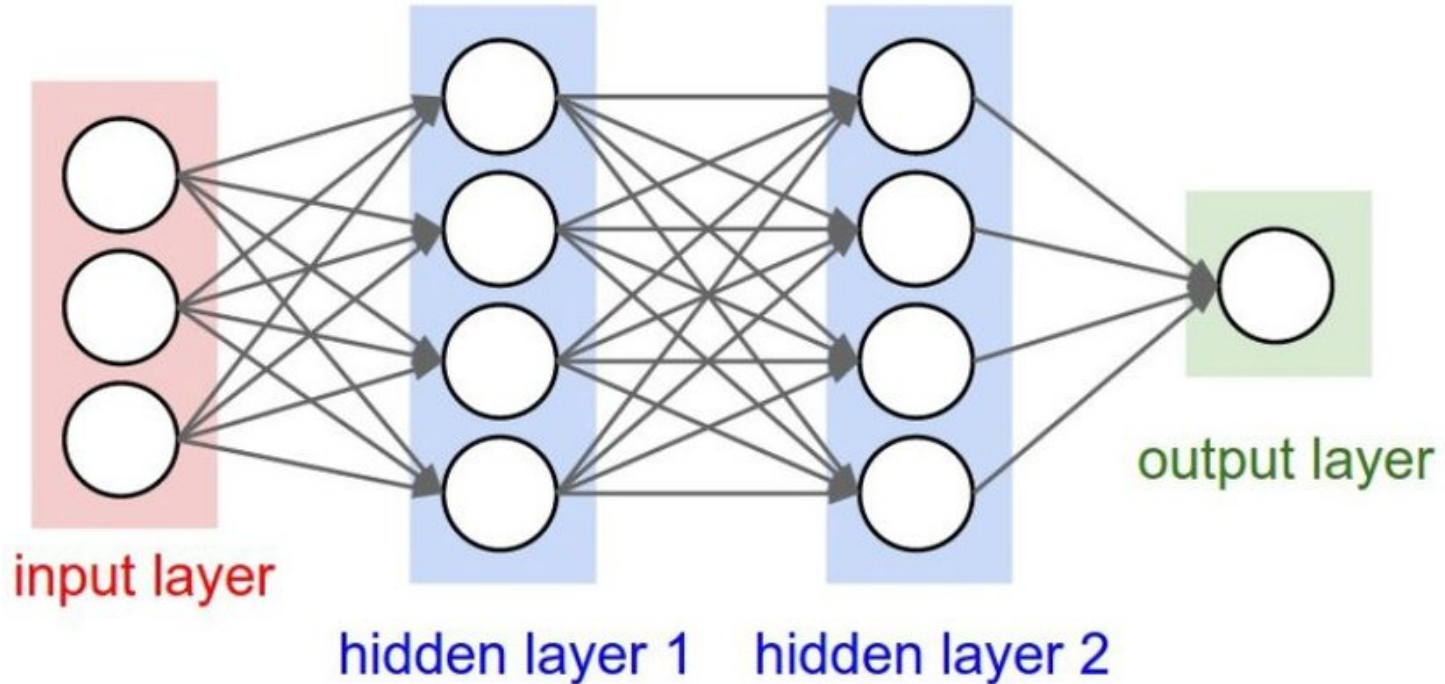


Perceptron (single layer of TLUs)

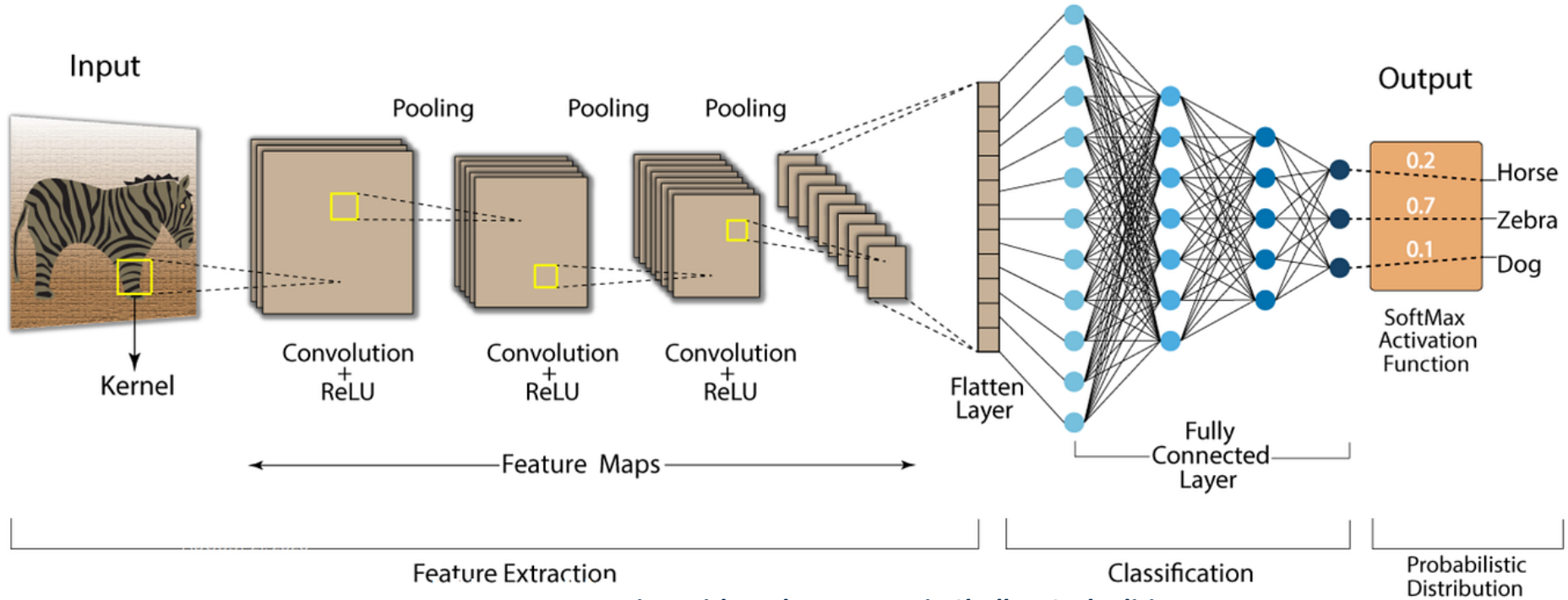


Multi-layer perceptron (deep neural network)



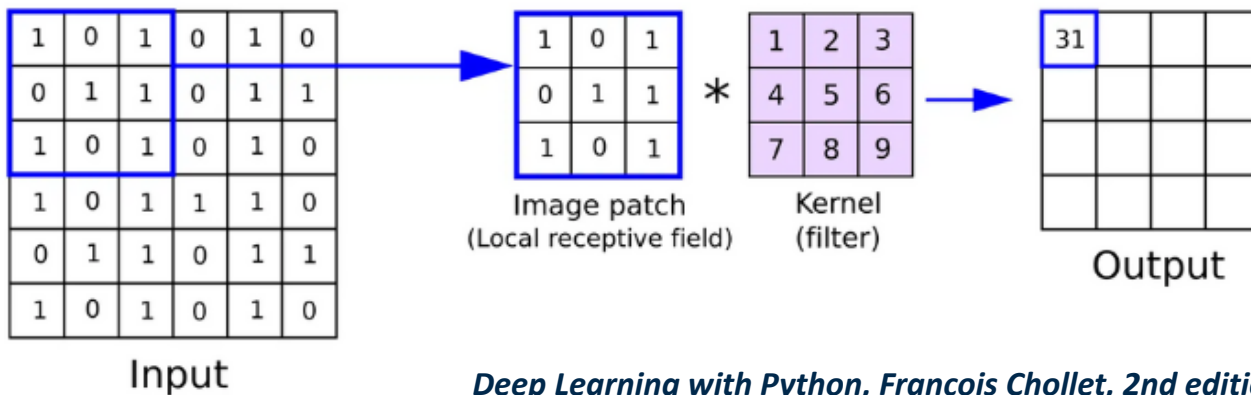
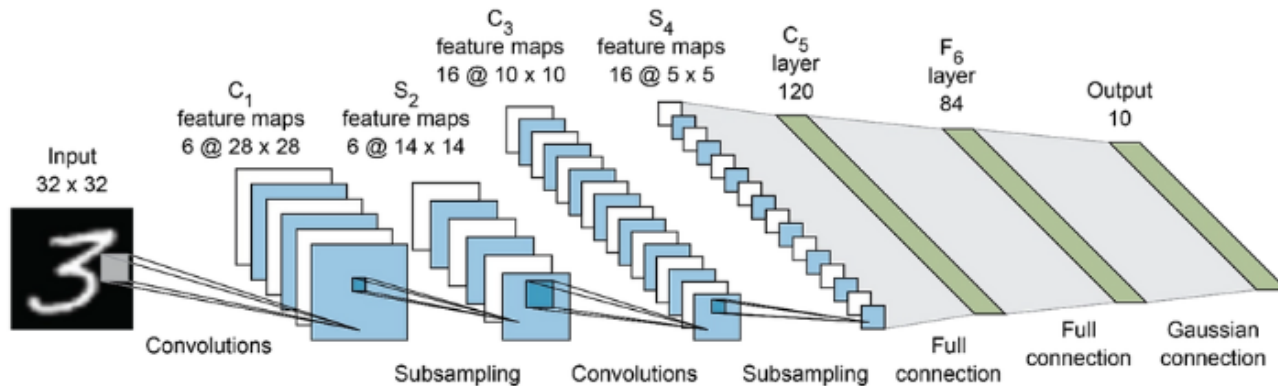


Jupyter notebook Clothes classification



Deep Learning with Python, François Chollet, 2nd edition

The convolution step



Deep Learning with Python, François Chollet, 2nd edition

Max Pooling / Average Pooling

Max Pooling

29	15	28	184
0	100	70	38
12	12	7	2
12	12	45	6

2 x 2
pool size

100	184
12	45

Average Pooling

31	15	28	184
0	100	70	38
12	12	7	2
12	12	45	6

2 x 2
pool size

36	80
12	15

*Deep Learning with Python,
François Chollet, 2nd edition*





0123456789

Output Layer

FC Layer 2

FC Layer 1

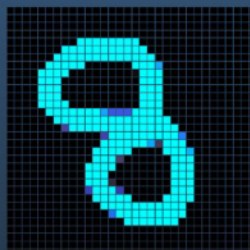
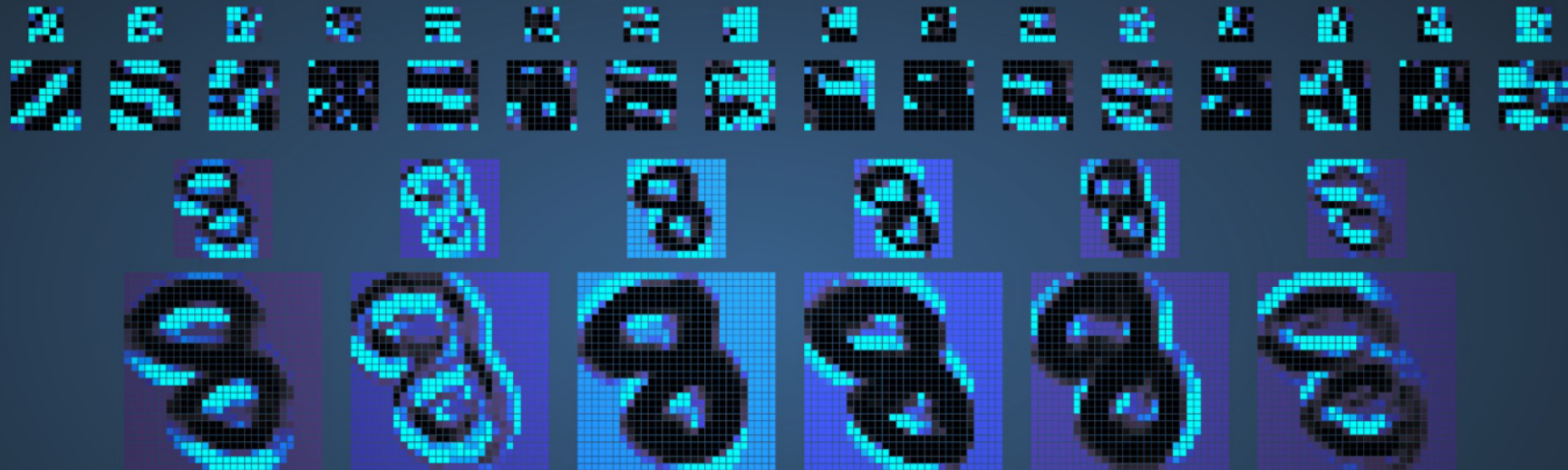
Pooling Layer 2

Convolution Layer 2

Pooling Layer 1

Convolution Layer 1

Input Layer

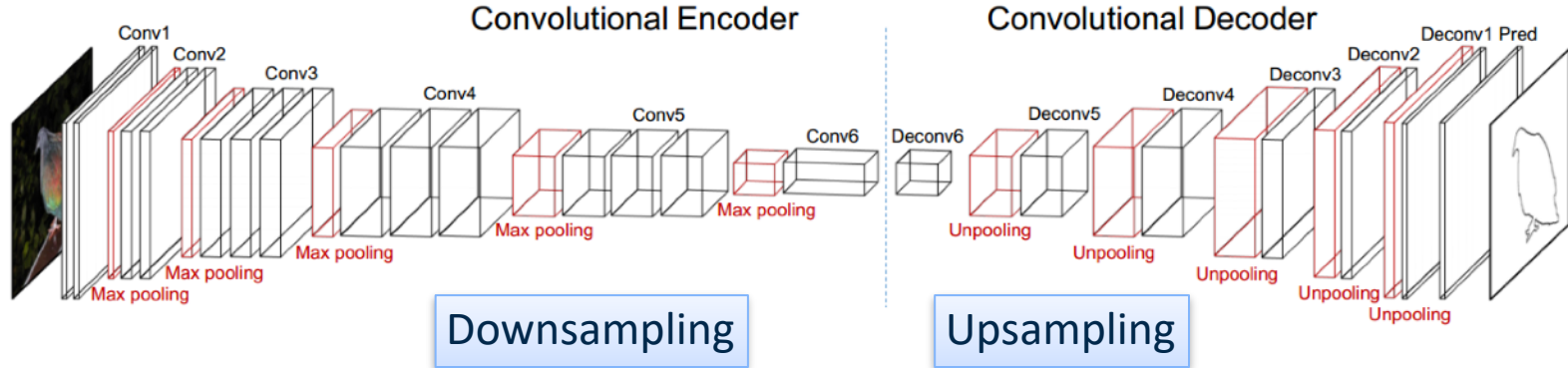


<https://www.kdnuggets.com/2016/11/intuitive-explanation-convolutional-neural-networks.html/3>

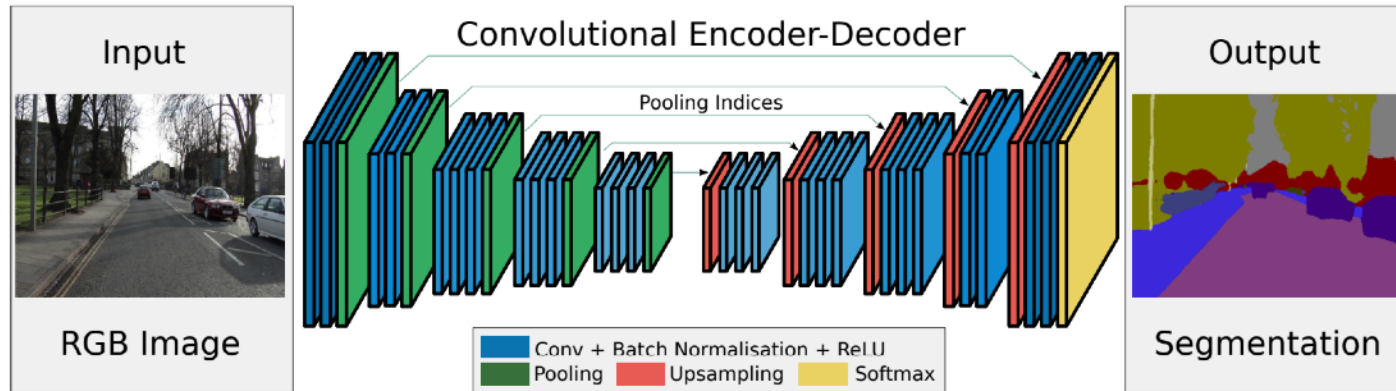
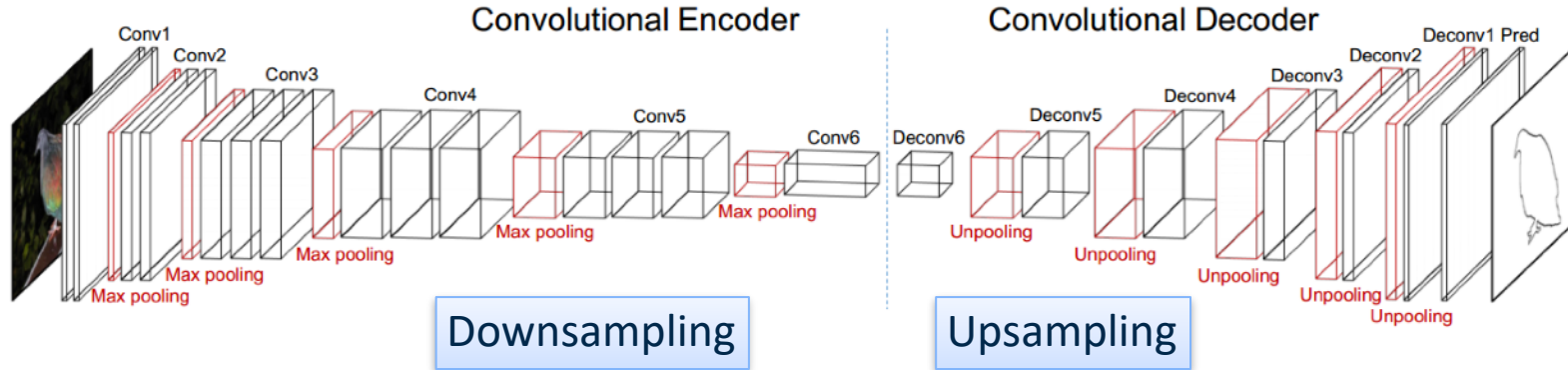
Jupyter notebook Clothes classification with CNN

Jupyter notebook Dogs and cats with and without data augmentation

Encoder-Decoder network



Encoder-Decoder network



0.1	0.5	1.2	-0.7
0.8	-0.2	-0.5	0.3
0.4	0.9	-0.1	-0.2
-0.6	0.1	0.5	0.3

max-pooling

0.8	1.2
0.9	0.5

		X	
X			
	X		
		X	

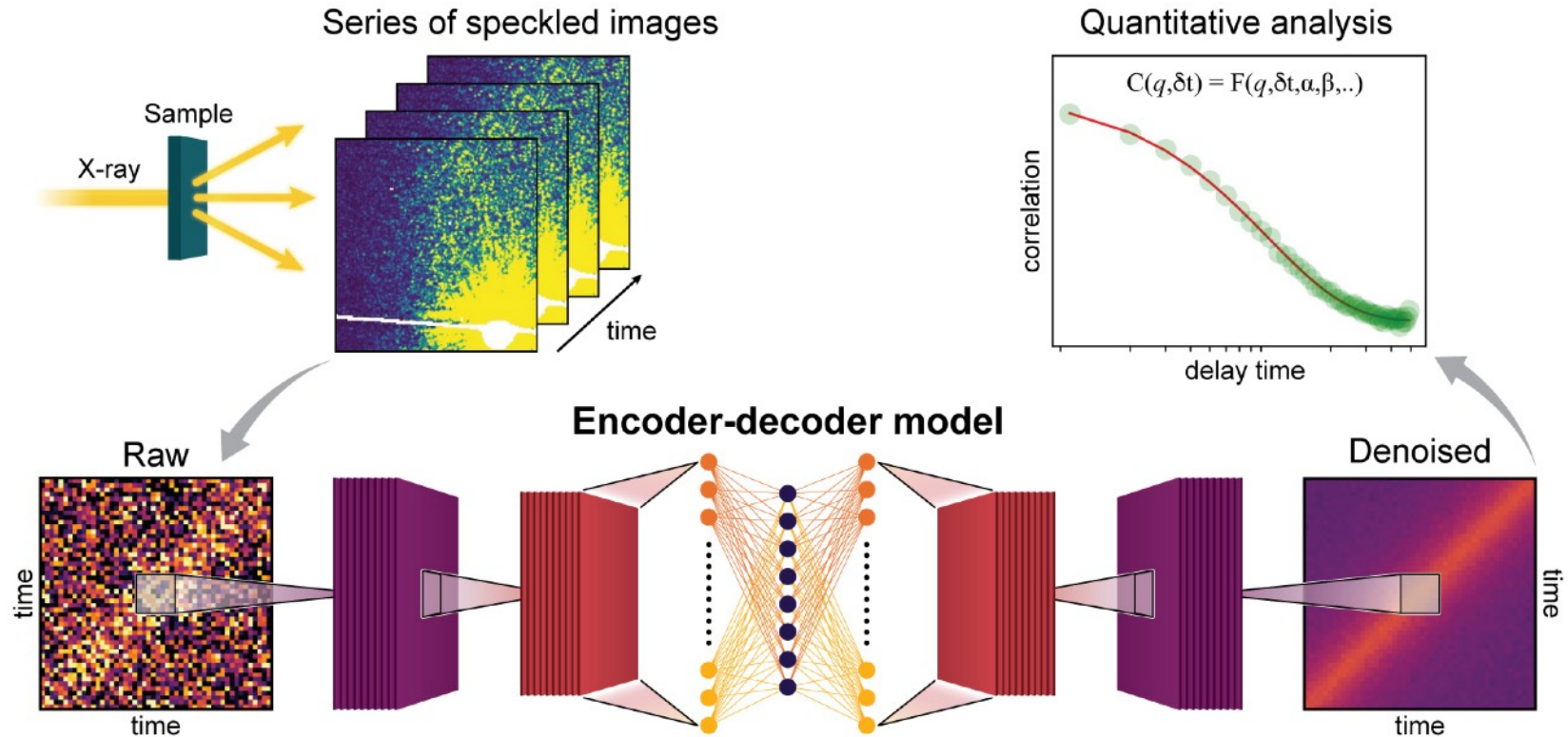
max locations

0	0	0.5	0
1.3	0	0	0
0	0.4	0	0
0	0	0.1	0

unpooling

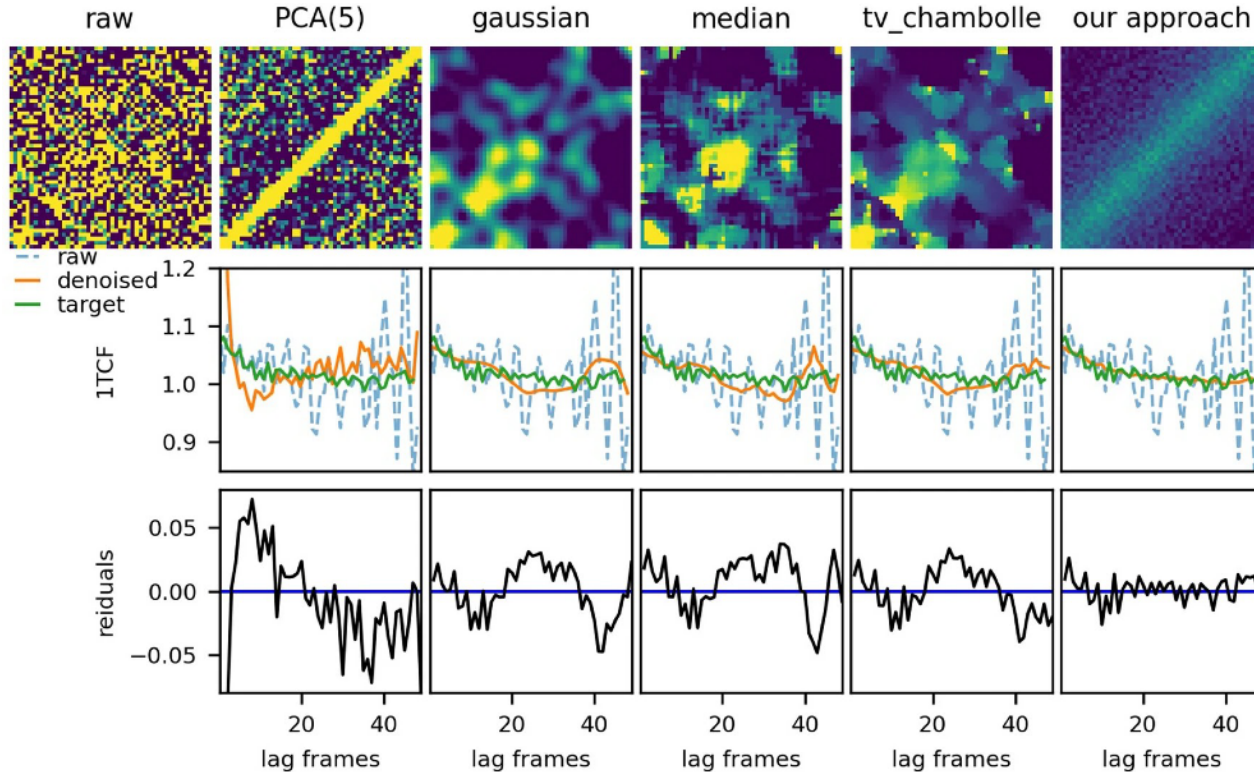
1.3	0.5
0.4	0.1

Encoder - decoder for XPCS data denoising

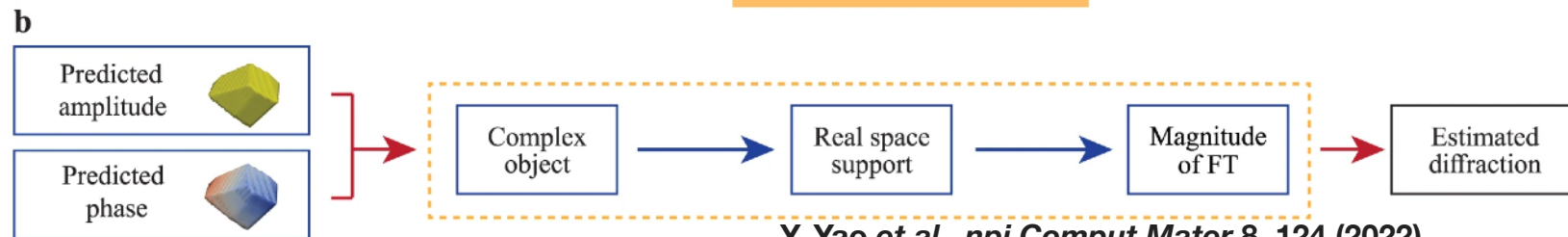
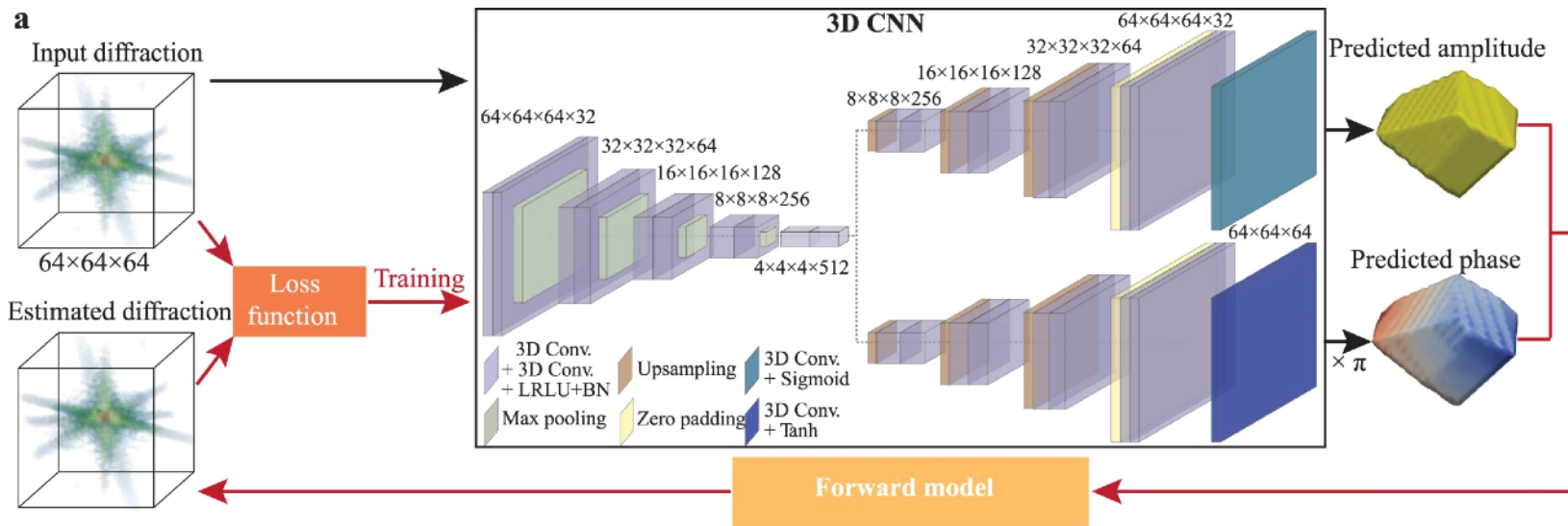


T. Konstantinova et al. *Sci Rep* 11, 14756 (2021)

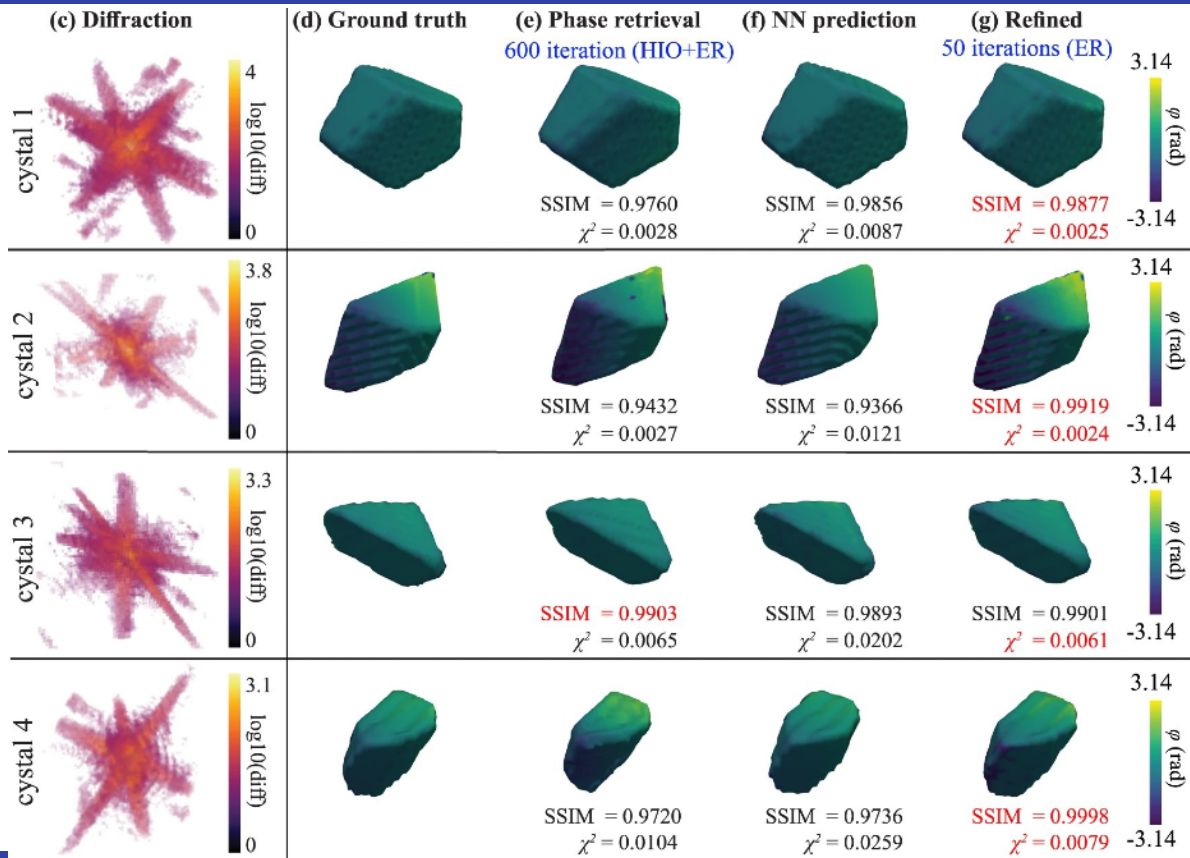
Encoder - decoder for XPCS data denoising

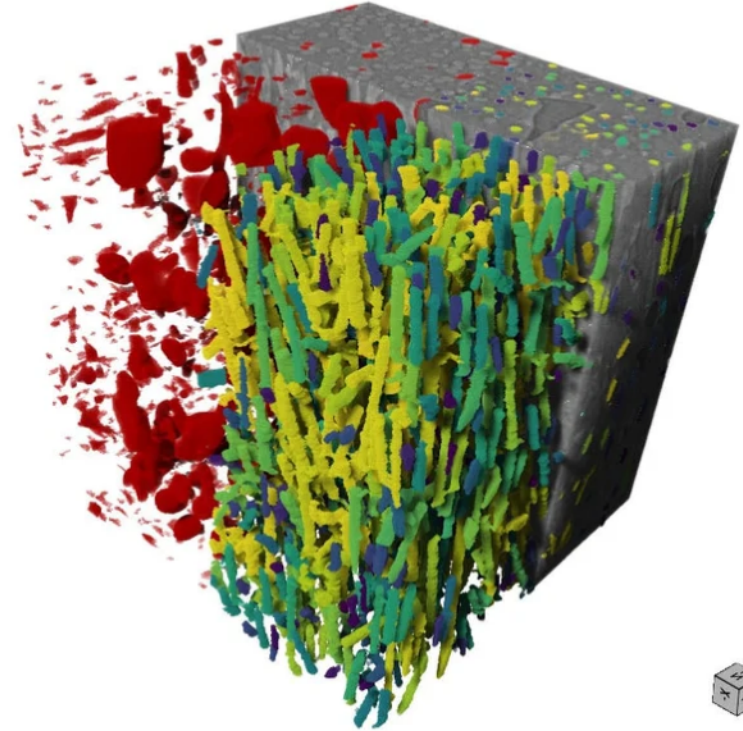
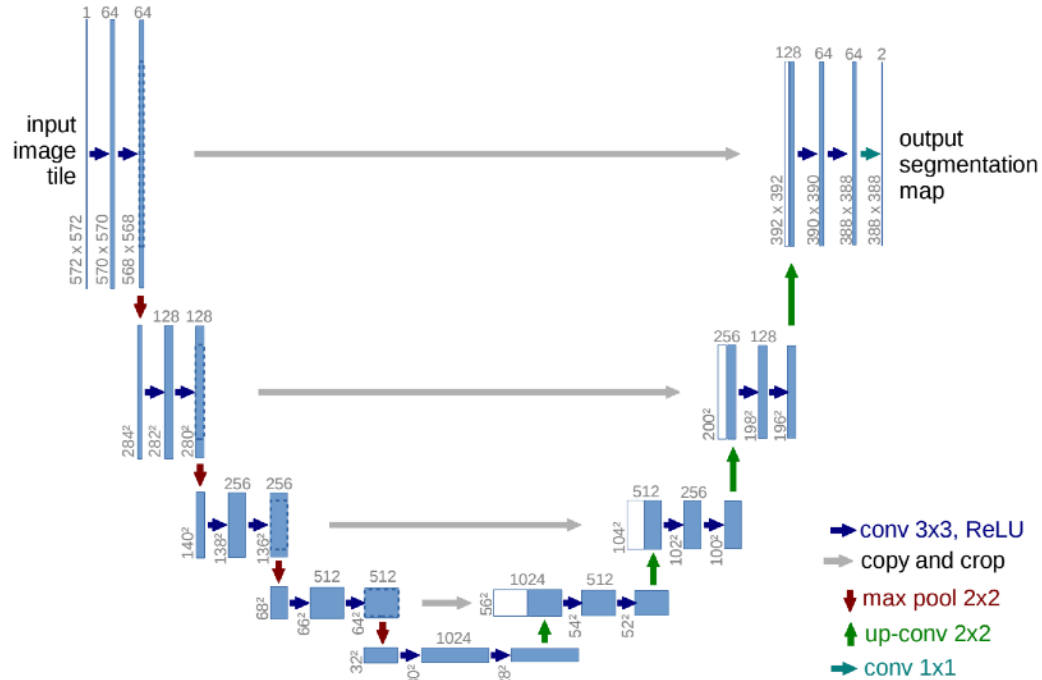


T. Konstantinova *et al. Sci Rep* 11, 14756 (2021)



Y. Yao et al., *npj Comput Mater* 8, 124 (2022)





O. Ronneberger et al. (2015). MICCAI 2015. Lecture Notes in Computer Science, vol 9351. Springer

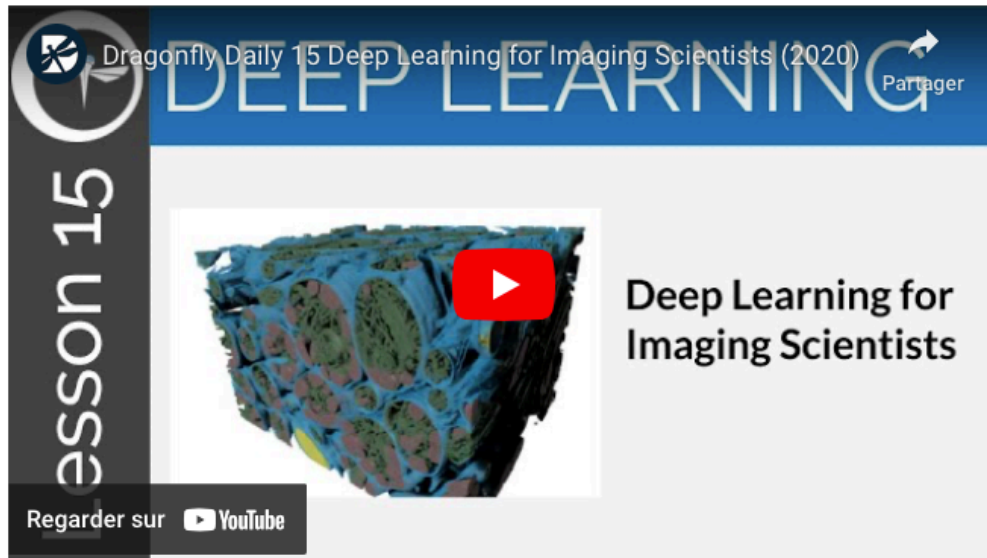




Non-Commercial Licensing

We proudly support academic research...

DRAGONFLY FREE 30-DAY TRIAL



Deep Learning for Imaging Scientists

This video provides the background details for a better understanding of Dragonfly's Deep Learning Tool and how to train deep models for denoising, image segmentation, and super-resolution. Topics include the comparison of image processing to linear regression, fitting and applying functions, perceptron and neural networks as functions, as well as model training and optimization.

References and data examples:

Go to Grant Sanderson's YouTube Channel, 3Brown1Blue, for a deeper explanation on neural networks and how they are trained.

Pharmaceutical tablets example from Ma et al. 2020.

Rat neurons example from Eustaquio et al. 2018.

<https://www.theobjects.com/dragonfly/deep-learning-getting-started.html>

Model: U-Net_millet



Train All Layers

Freeze All Layers

Duplicate Layer

Delete Selected

Select New Layer Type

Filter

Attribute

Value

Add New Layer

Attribute

Value

Input Layer

Output layer

Dropout

Graph Layout: Vertical HorizontalArrange Layers by: Position Size

Reorganize Graph

Fit Graph to View

Back to Model Overview

Go to Editing

Go to Training

Reset

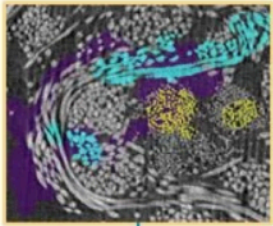
Reload

Save

Close



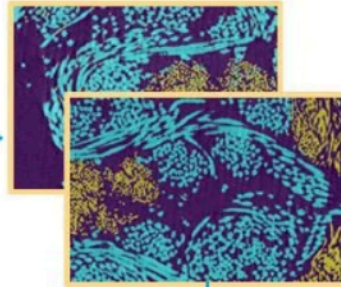
Initial manual input



Random Forest 1

Random Forest 2

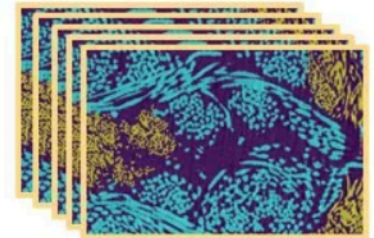
Prediction & correction



U-Net

3D sensor

Segment all




Dragonfly Workstation (Version 2022.1 Build 1231) (E:/calendar files/05-CFRP/2_Cf-PA12_x2_130MB_segmet_artwork_ORSSession)

The interface displays a 3D volume rendering of a material structure, likely a carbon fiber reinforced polymer (CFRP), with different components colored in red, yellow, green, and blue. The volume is shown in a perspective view, with a small 3D navigation cube at the bottom right. To the right of the 3D view, three 2D slices are shown, each with a 1 μm scale bar. The top slice is a grayscale image, while the bottom two are color-coded to match the 3D volume. The software interface includes various toolbars for manipulation, annotation, and window leveling. The right-hand side features a Properties panel with data properties and settings, including a table of labeled classes and their counts.

Properties
Data Properties and Settings

Class	Count
void	38
Fiber <-100um3*	38
2_CF--	38

Basic properties

- Width: 511 px (270.83 μm)
- Height: 511 px (270.83 μm)
- Depth: 512 px (270.36 μm)
- Time steps: 1
- Total voxels: 133,683,952
- Volume: 25,903,964.49 μm³

Statistical properties

- Labeled class count: 2,094
- Labeled voxels count: 14,148,577
- Volume of labeled voxels: 2,106,397.70 μm³ (10.589)

Classes and scalar information

Name	Count	Label
	130,662	2,089
	122,388	2,088
	132,464	2,089
	149,504	2,090
	187,242	2,091
	170,949	2,092
	210,592	2,093
	584,768	2,094

Background class: Drag class here

Labeled voxels count: 0

Scalar information
Measurement: None

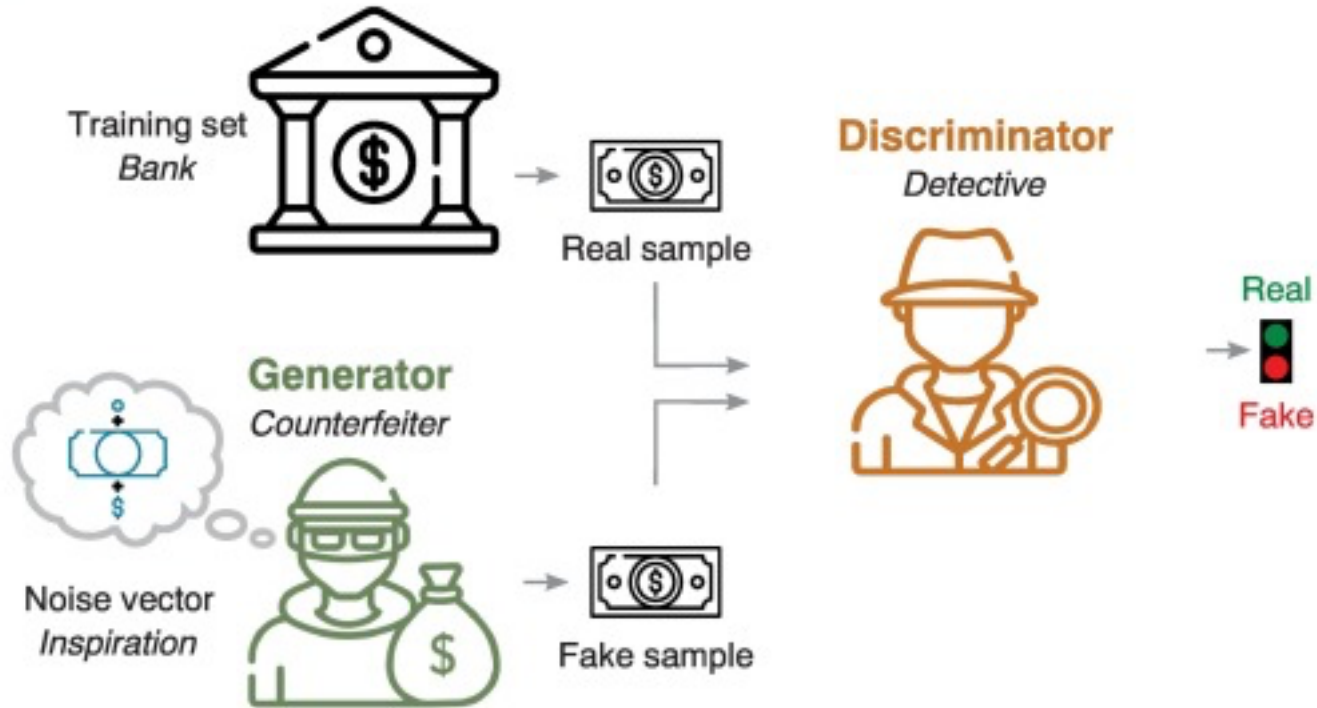
2D settings
 Show only contour
Contour thickness: [Slider]

Opacity and color
Dataset: [um_computed_tomography_130MB]

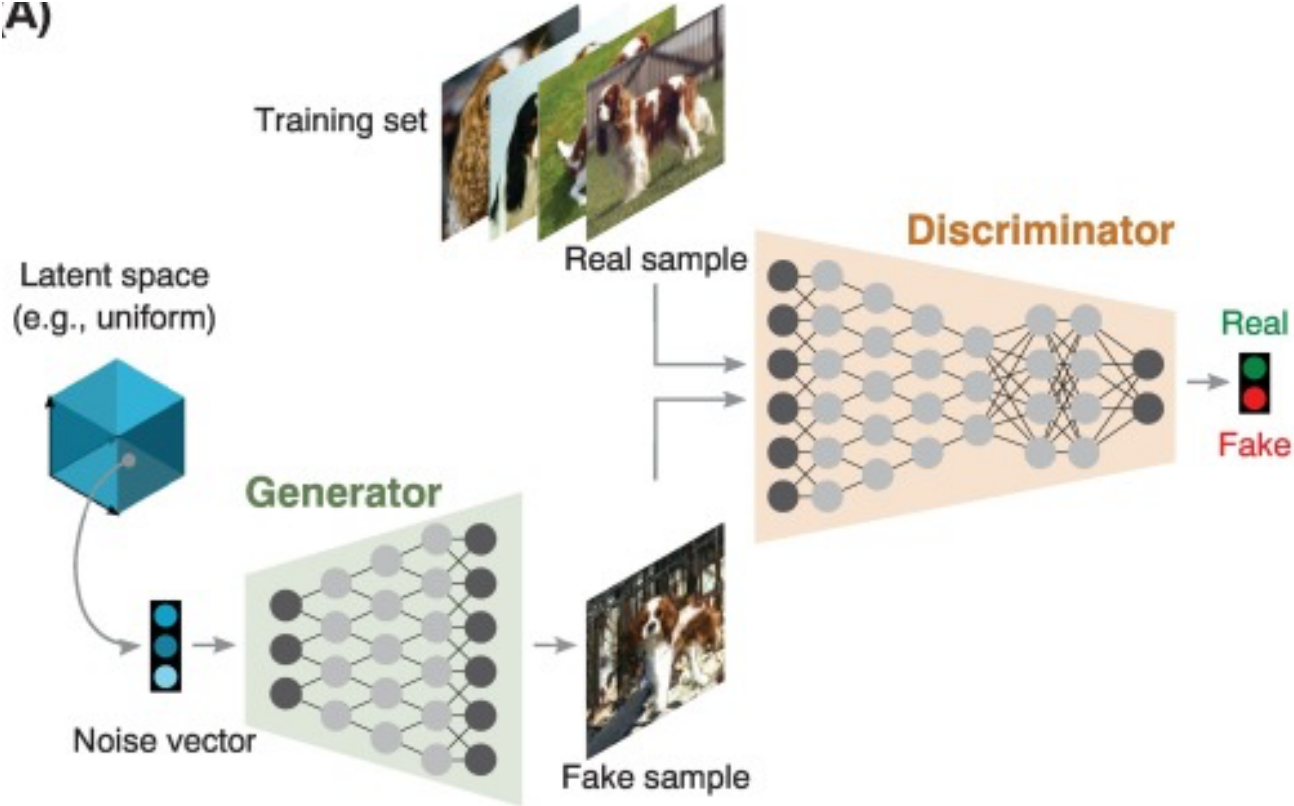
Current state: Leveling (left mouse)

Maintenance expires: 11/6/2022 | New Session... | Preferences

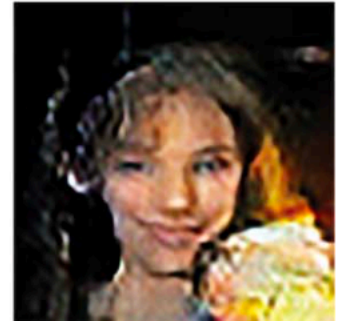
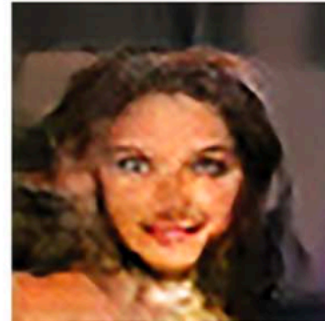
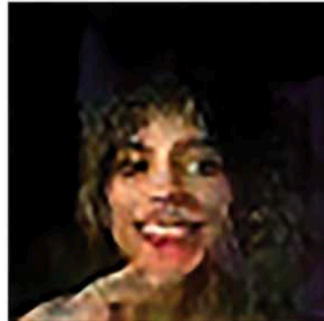
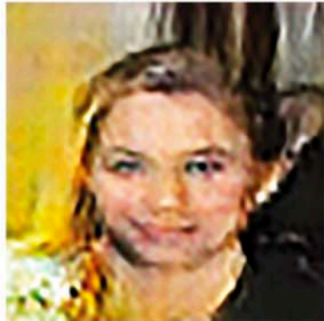
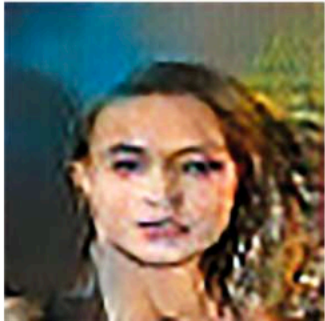
GAN - Generative Adversarial Network

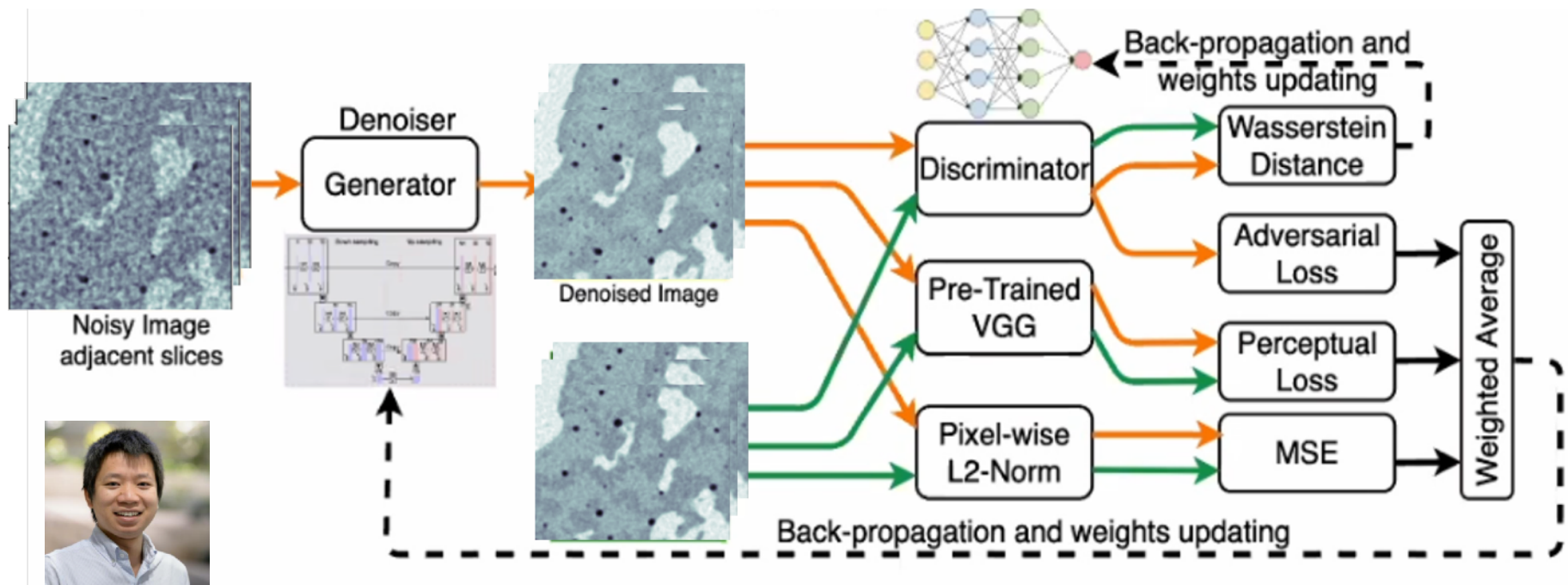


A)



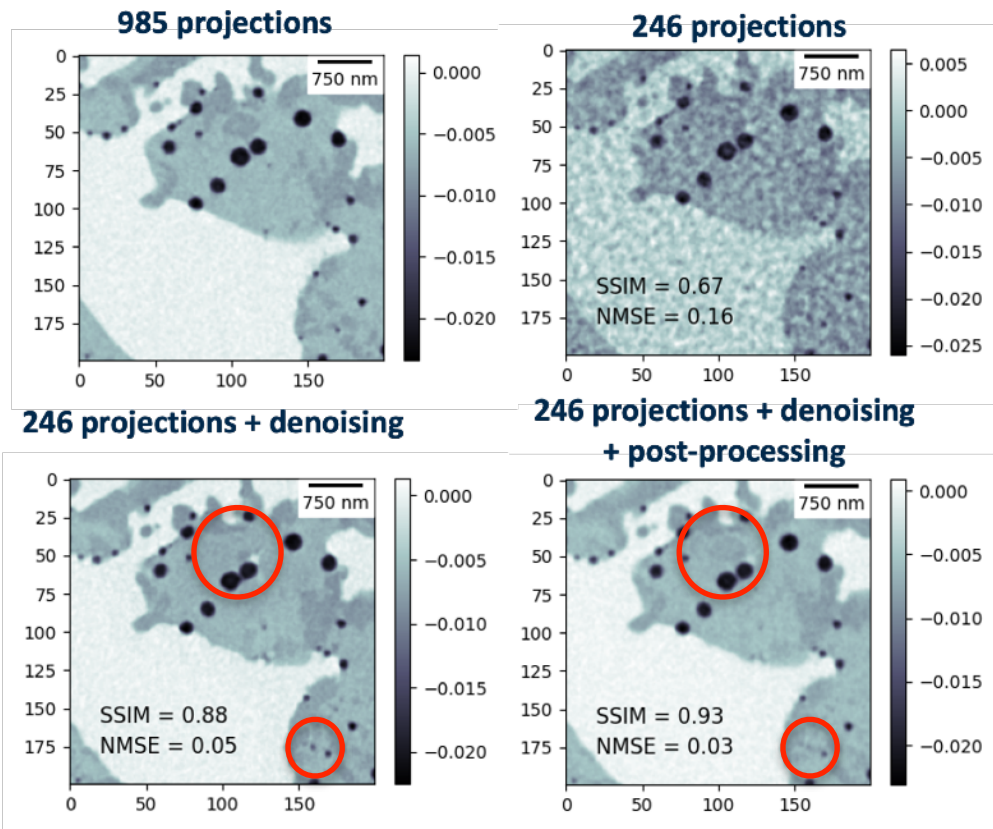
Jupyter notebook Fake Faces with GAN





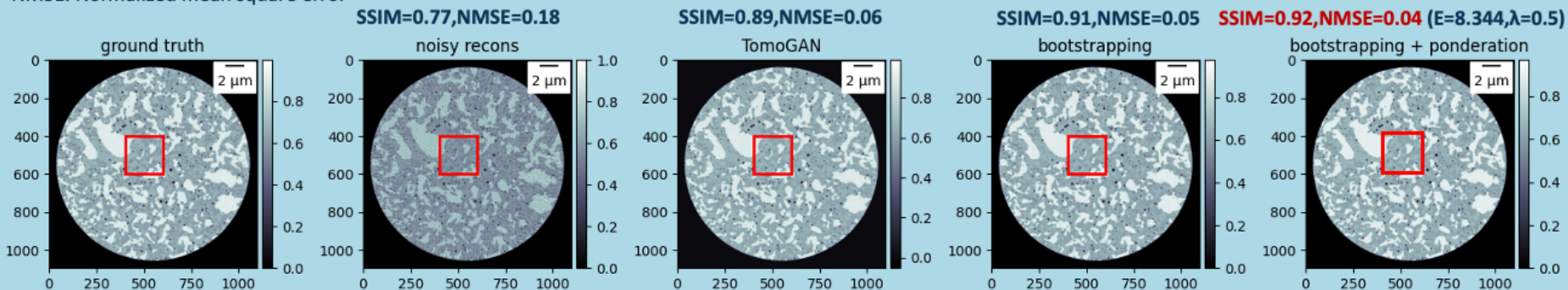
Zhengchun Liu

Jupyter notebook Denoising with TomoGAN

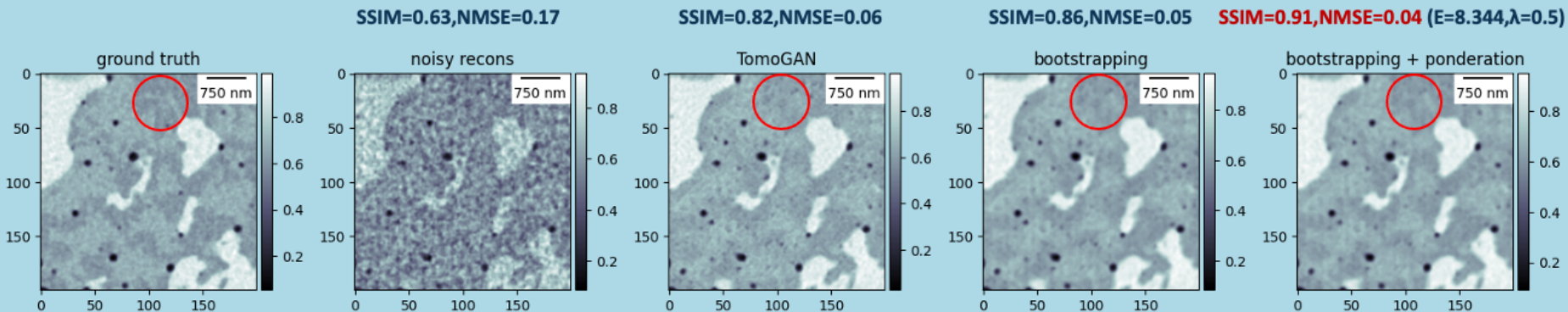


SSIM: Structural SIMilarity
 NMSE: Normalized mean square error

TomoGAN for 1/8 of projections



TomoGAN for 1/8 of projections



Thank you for your attention

Contact info:

`julio-cesar.da-silva@neel.cnrs.fr`

Website:

`sites.google.com/view/jcesardasilva`

Data analysis tools:

`https://github.com/jcesardasilva`

**Flash me for
more info ▼**



QR code