Autoencoders and U-Net

Third ML-INFN Hackathon: Advanced Level
21-24 November 2022, Bari (Italy)

Francesca Lizzi

INFN Pisa

22 November 2022

Istituto Nazionale di Fisica Nucleare

Outline:

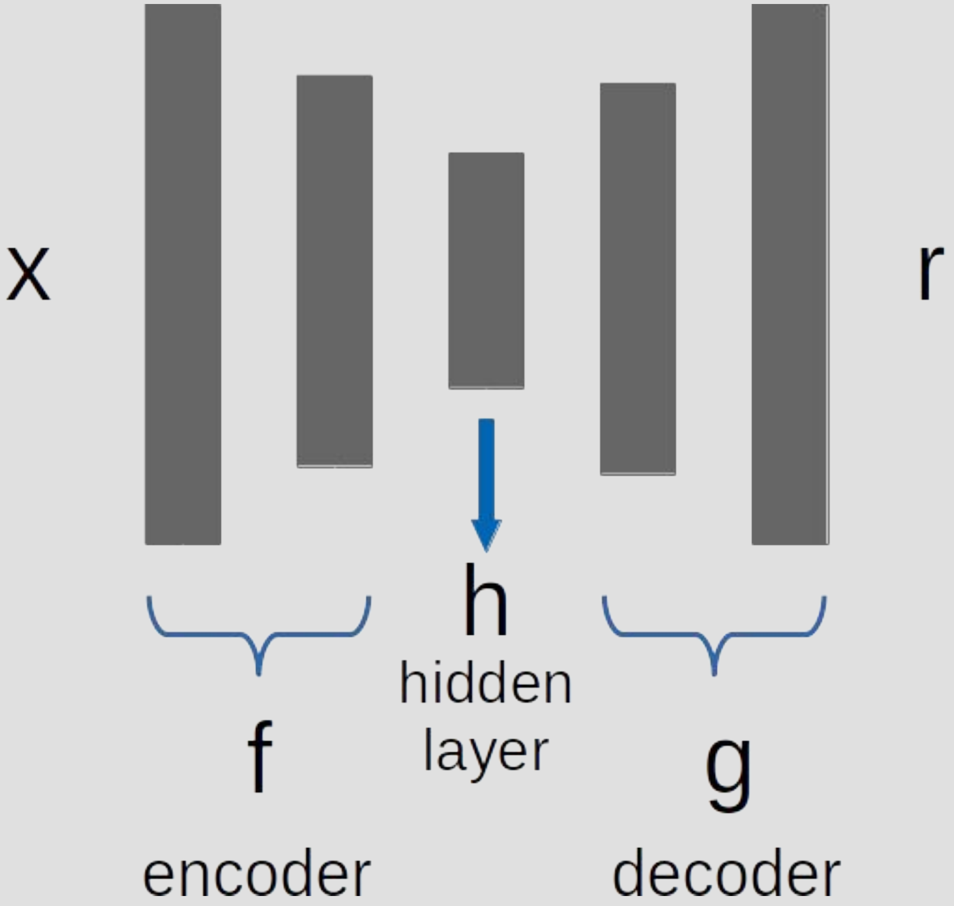
1. Autoencoder: introduction and general intuition
2. Undercomplete Autoencoder
3. Regularized Autoencoder:
    a. Denoising Autoencoder
    b. Sparse Autoencoder
4. From Autoencoder to U-Nets: medical image segmentation
    a. Transposed Convolution
    b. Skip Connections
5. An application of U-Nets: introduction to the hackathon exercise

An AutoEncoder Neural Network is an unsupervised learning algorithm that is trained to attempt to copy its input to its output.

It is made of two main parts:

- an encoder -> $\mathbf{h} = f(x)$
- a decoder -> $\mathbf{r} = g(\mathbf{h})$

h is a hidden layer (latent space) that describes a code used to represent the input.



If an autoencoder learns $g(f(\mathbf{x}))=\mathbf{x}$ everywhere...
It is not useful!

We want to insert something that avoid the perfect copy of the input!

The first thing we can do to avoid the learning of identity is to let the latent space be smaller than the input.

REMINDER : Capacity is an informal term and it is the capability of a NN to use the information in a significant way. More neurons, more layers correspond to more capacity.

Undercomplete Autoencoder

The learning process is simply the minimization of a loss function:

$$L(x, g(f(x)))$$

The loss function can be chosen among the ones that penalize $g(f(\mathbf{x}))$ for being dissimilar to $\mathbf{x}$.

Regularized Autoencoder

We have just seen that we can act on capacity to avoid the autoencoder to learn the identity function.

Can we build a non-linear and overcomplete autoencoder that does not learn the identity?

YES

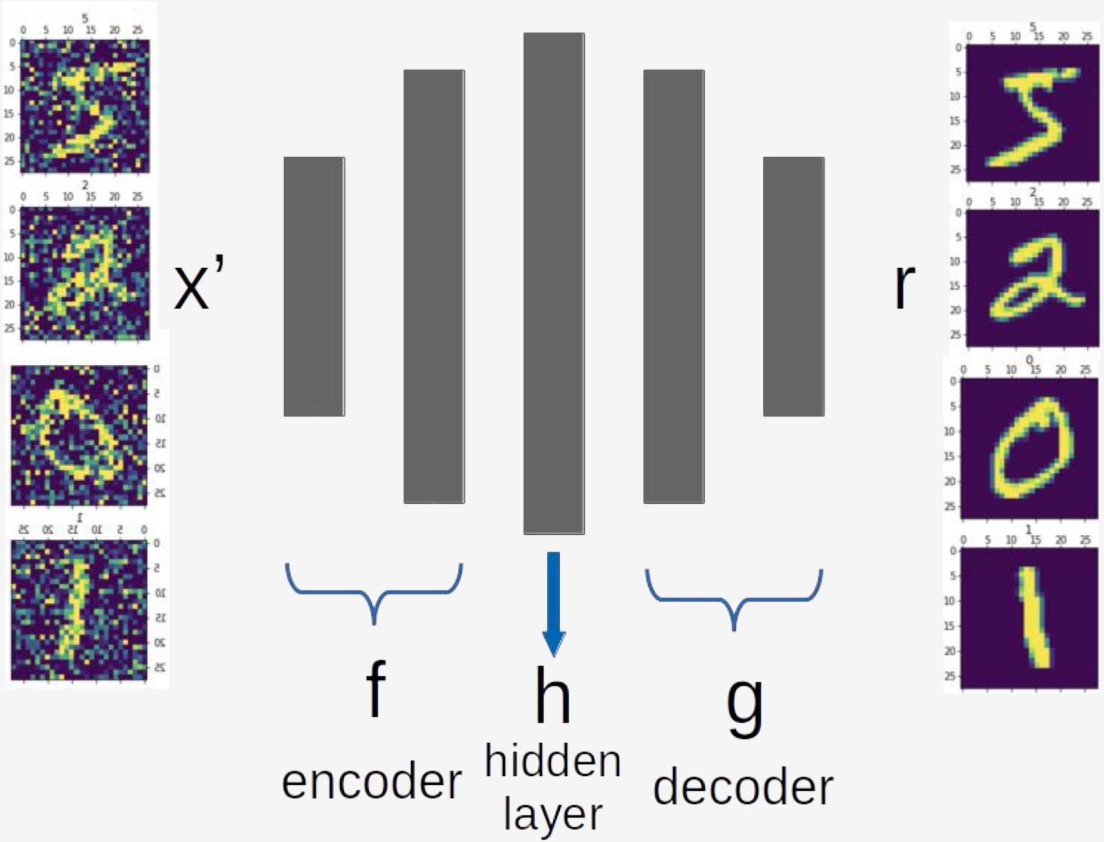
How? Instead of limiting capacity we can make something that pushes the model to have other properties

Denoising autoencoder (DAE):

DAE receives corrupted data points as input and it is trained to predict the original one as output.

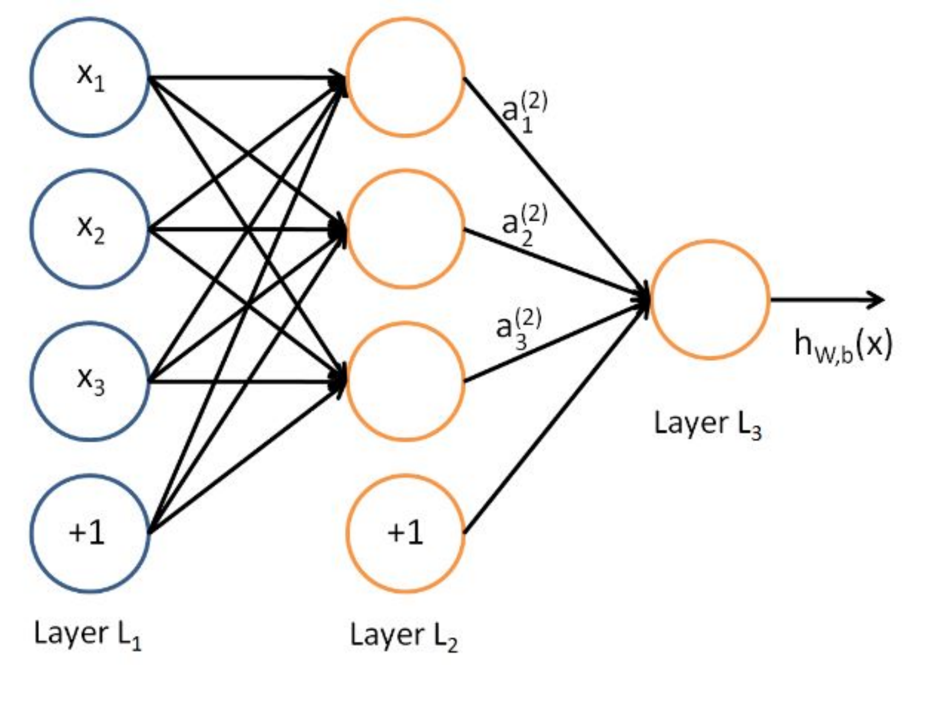
$$L(x, g(f(x')))$$

Is this an unsupervised algorithm?



Sparse autoencoder

Informally, we think that a neuron is "active" if its output value is close to 1 or "inactive" if it is close to 0: in a sparse AE we constrain the neurons of the hidden layer to be inactive for most of the time.



We insert in the loss function a **SPARSITY penalty $\Omega(\mathbf{h})$** on the code layer h:

$$L(x, g(f(x))) + \Omega(\mathbf{h})$$

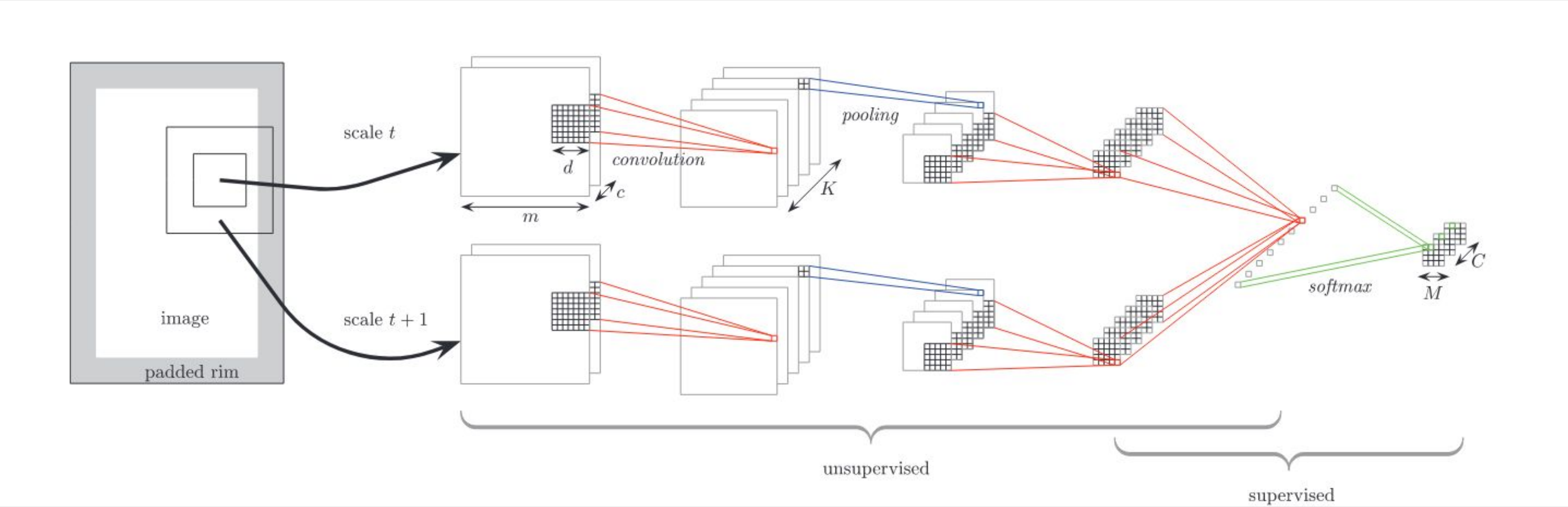
Traditionally autoencoders were used for:

- dimensionality reduction
- learning features

Nowadays, they are used also as generative models (you will see it in a later lesson by Francesco Vaselli and Matteo Barbetti)

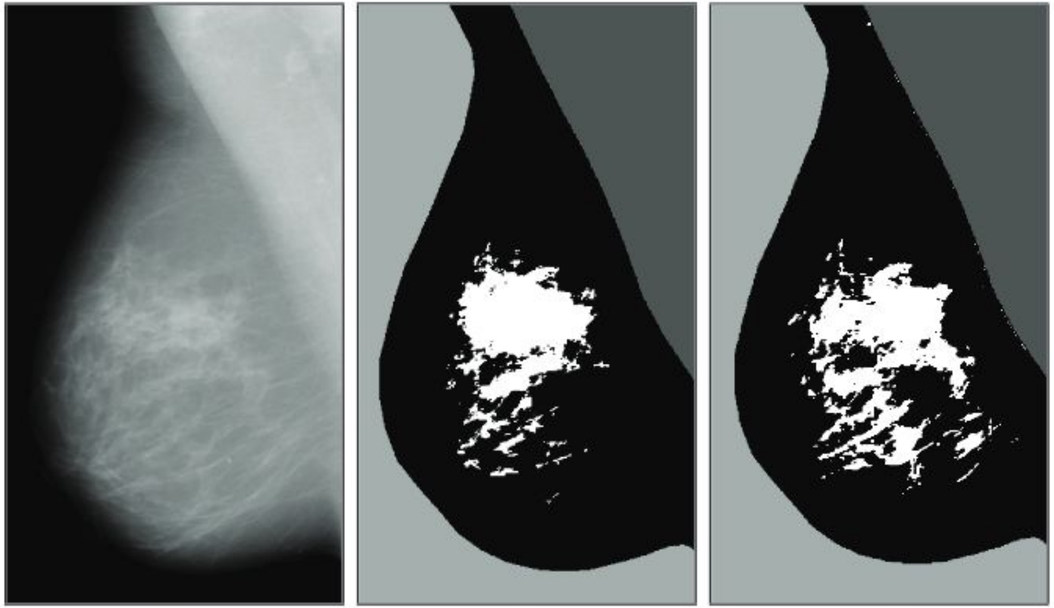
Convolutional Sparse AutoEncoder (CSAE) for breast density

In this study, authors used a CSAE to extract features and then trained a classifier for breast density.



M. Kallenberg *et al*., "Unsupervised Deep Learning Applied to Breast Density Segmentation and Mammographic Risk Scoring," in *IEEE Transactions on Medical Imaging*, vol. 35, no. 5, pp. 1322-1331, May 2016, doi: 10.1109/TMI.2016.2532122.

Convolutional Sparse AutoEncoder (CSAE) for breast density



(a) (b) (c)

Fig. 3. Automated MD thresholding. Depicted are (a) original image, (b) dense tissue according to expert Cumulus-like threshold, and (c) dense tissue according to CSAE.

M. Kallenberg *et al.*, "Unsupervised Deep Learning Applied to Breast Density Segmentation and Mammographic Risk Scoring," in *IEEE Transactions on Medical Imaging*, vol. 35, no. 5, pp. 1322-1331, May 2016, doi: 10.1109/TMI.2016.2532122.

Segmentation of medical images

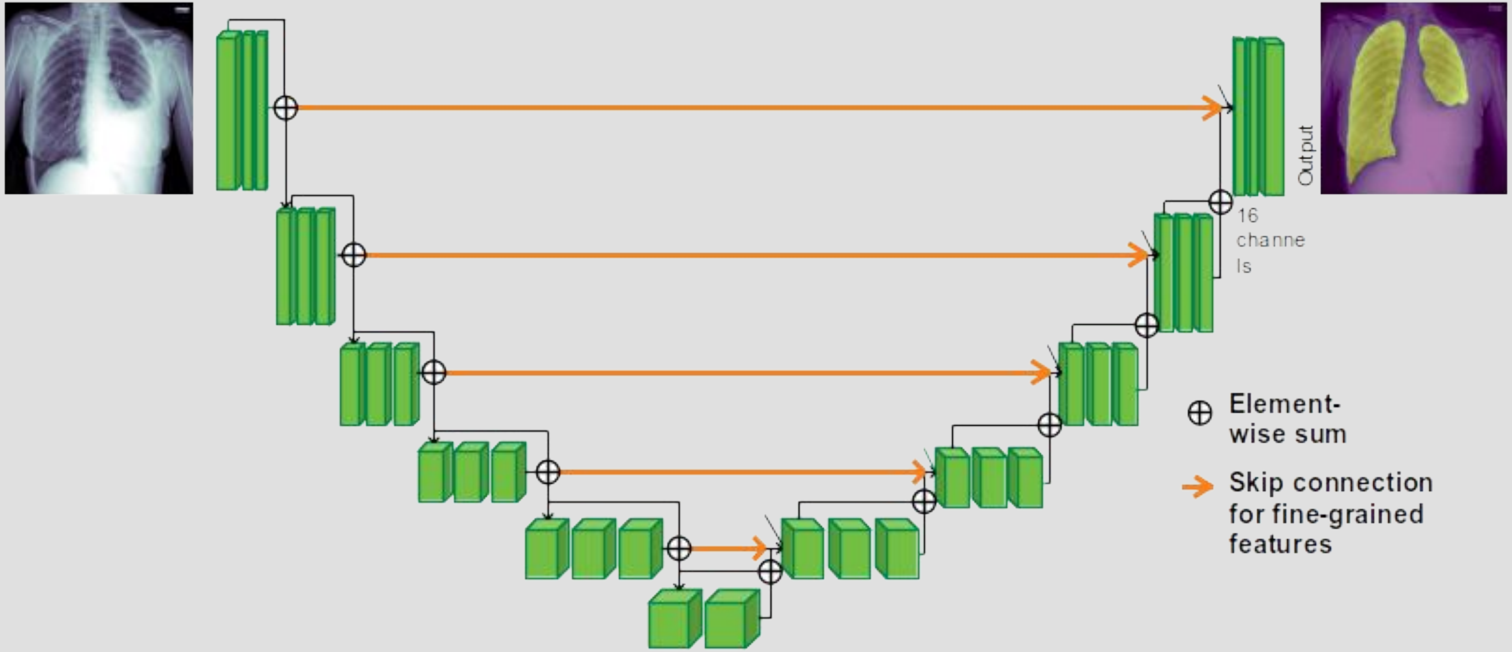
Medical images have some **peculiarities** with respect to other images:

- They are usually in **high resolution** (ex: a DM are usually about 4000x4000);
- We need to find **very small details** with respect the whole image;
- We could use a patch-wise approach but it is not always possible;

It is not often possible to make a priori considerations on distributions or constraints but we need to delete on the images those parts that are not relevant for our scopes. -> SEGMENTATION

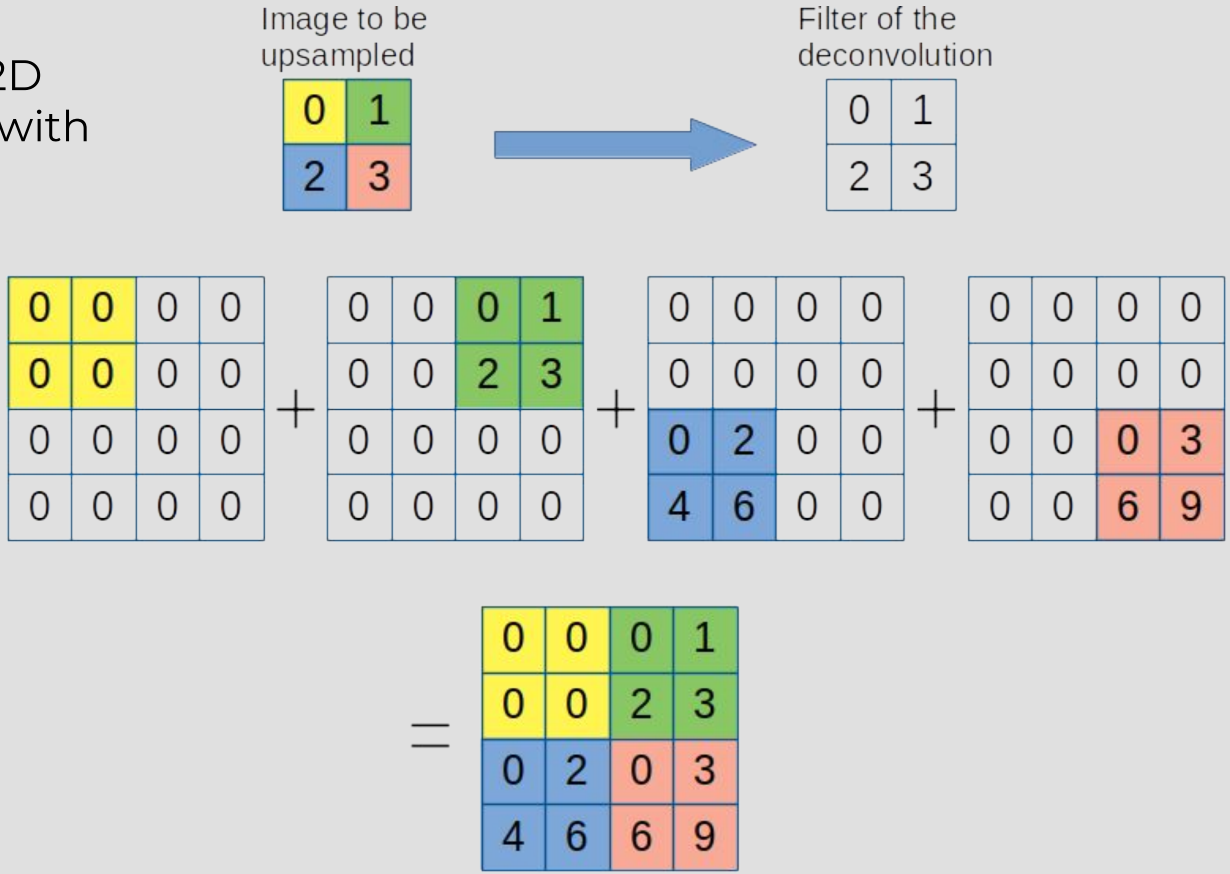
U-Nets

U-Nets are Fully Convolutional Neural Networks (FCNN) and the state-of-the-art method for medical image segmentation.
They have an encoder-decoder structure as autoencoders.



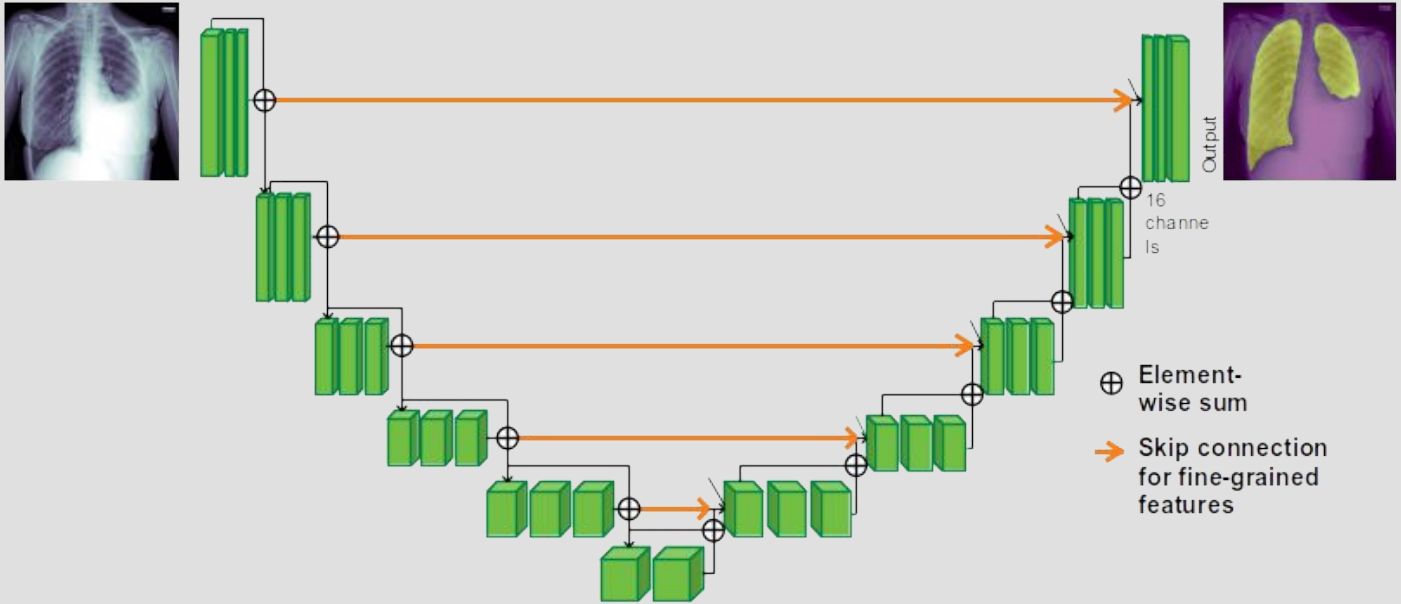
Transposed 2D Convolution (deconvolution)

Transposed 2D Convolution with stride 2



AE and U-Nets: skip connections

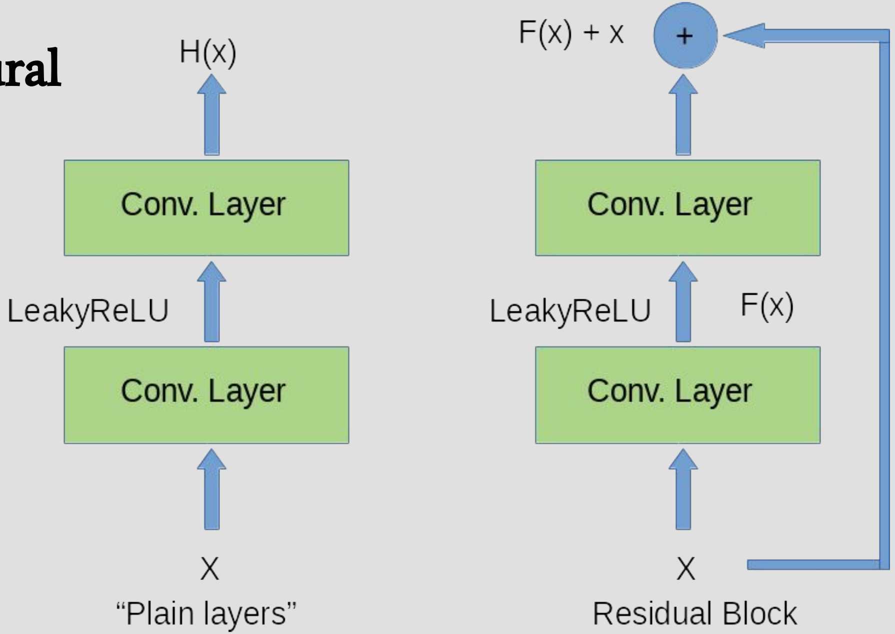
They are both made of an encoder and a decoder.
U-Nets are supervised learning algorithms while AE are unsupervised.
U-Nets exploit the skip connections (in orange in the Figure).



Skip connections

Skip connections are common in Convolutional Neural Networks. They consist in connecting different layers through **addition or concatenation** .

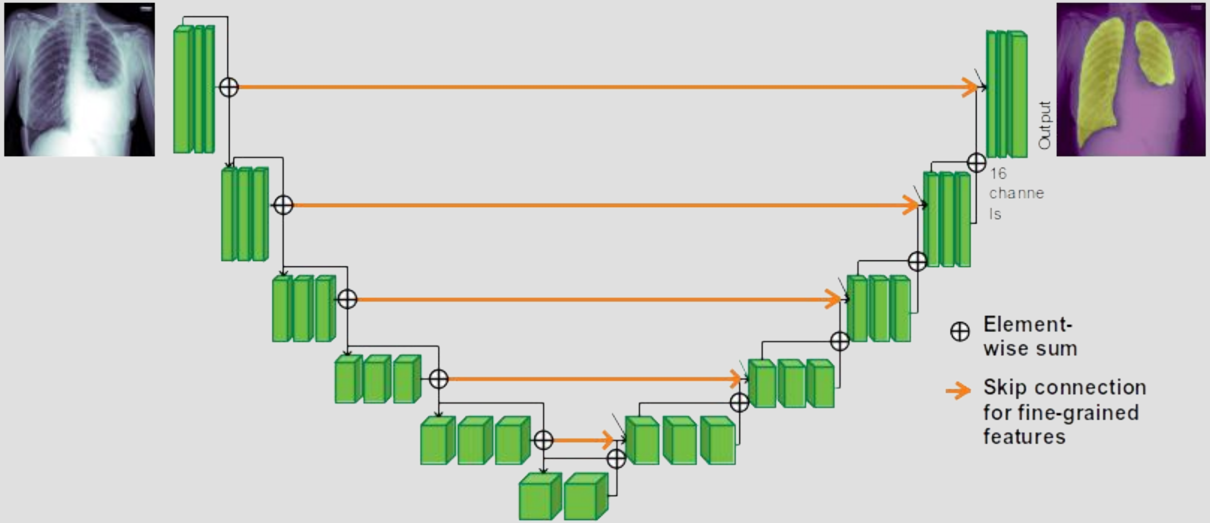
Residual Neural Network (ResNet):



Short skip connections:

- In ResNet, addition is used;
- Reduce the problem of the vanishing gradient;
- Preserve information through the (many) layers;

Skip connections in U-Nets

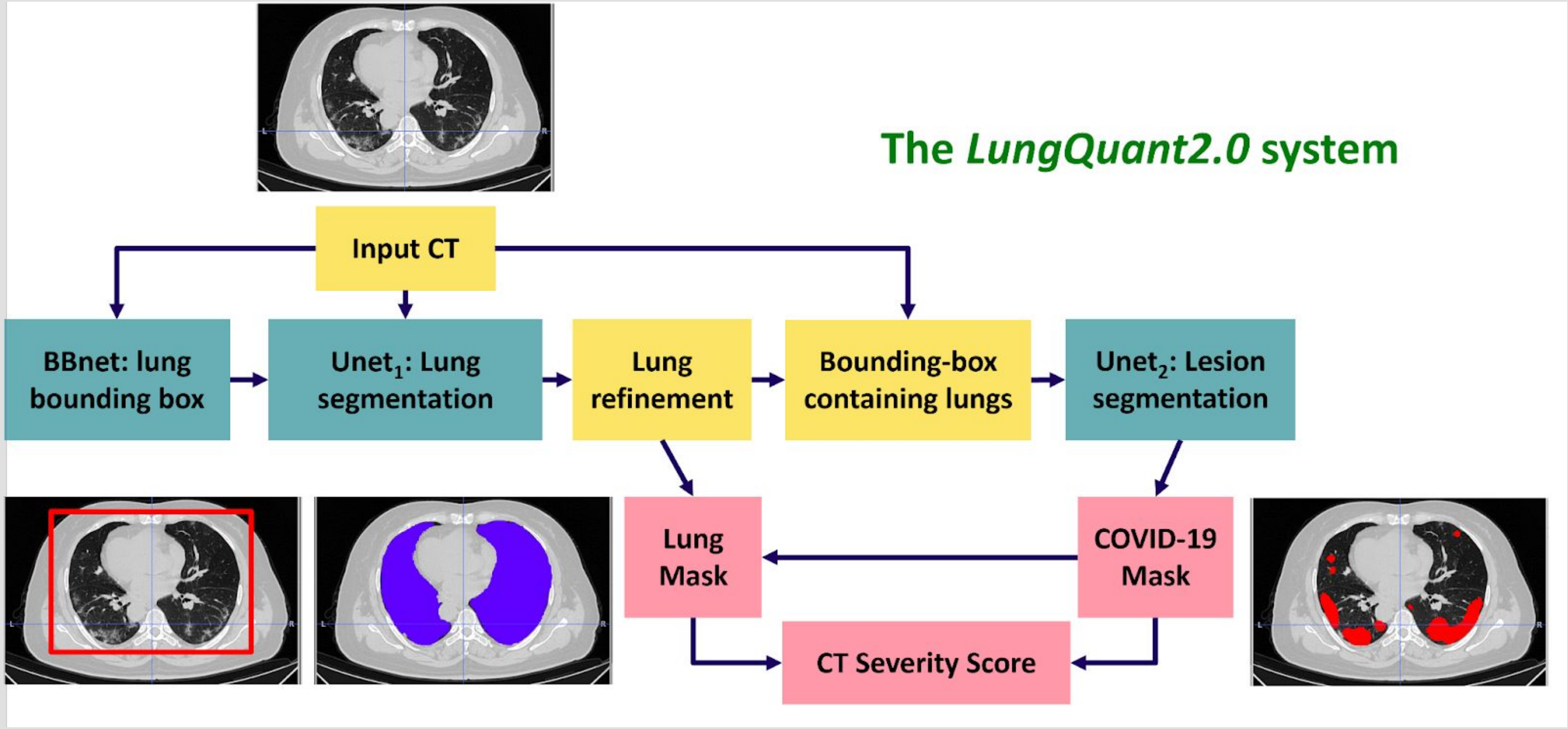


Long skip connections:

- In U-Nets, concatenation is used;
- Reduce the problem of the vanishing gradient;
- Preserve information that contains fine-grained details.

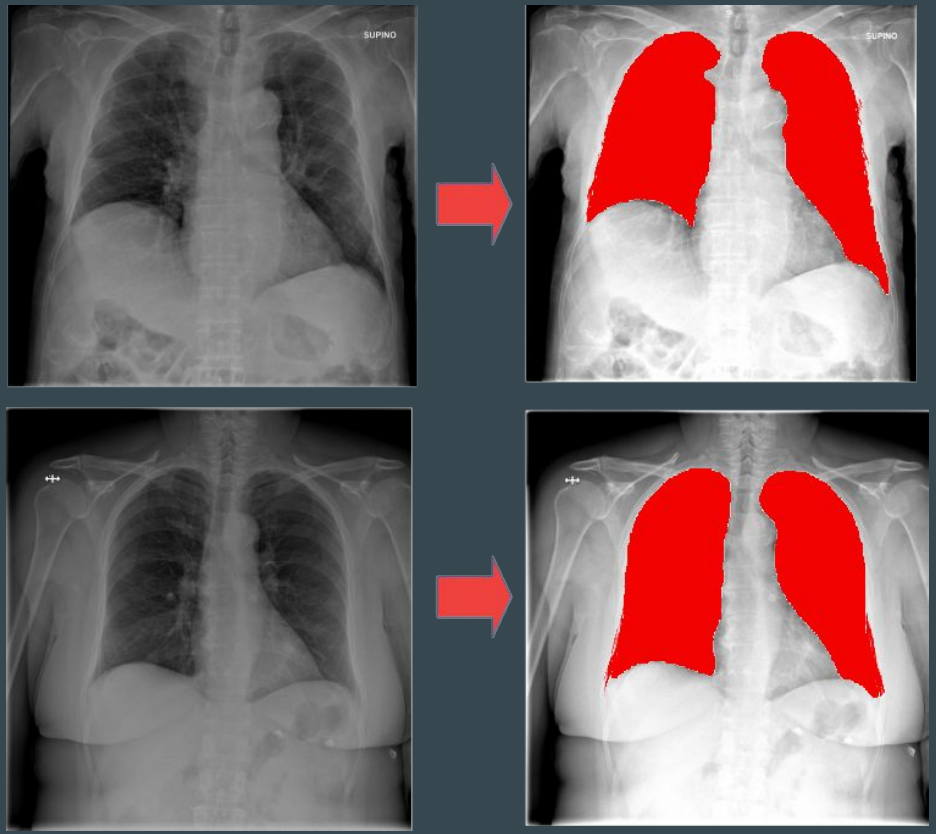
For both addition and concatenation we should check carefully the sizes of the parts of the network we are connecting. They have to match except for the addition/concatenation axis.

Example of U-Nets:



Lizzi F et al. Quantification of pulmonary involvement in COVID-19 pneumonia by means of a cascade of two U-nets: training and assessment on multiple datasets using different annotation criteria. International Journal of Computer Assisted Radiology and Surgery, 2021.

Chest X-Ray images



These are the images we are going to use this afternoon in the exercise.

As you can see, most of the pixels in this image do not belong to the lungs. So they are not useful if we want to analyze lungs.

We need a way to delete them.

This afternoon, we will write a U-Nets with Keras

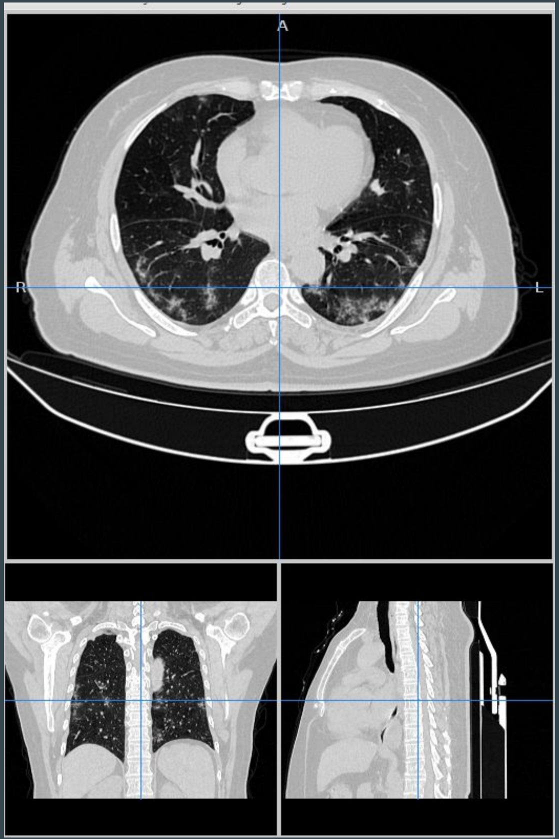
Exercise structure:

Data Generator	• To load data • To do real time data augmentation
Architecture	• You have to write the decoder
Loss and Metrics	• We will define the DSC loss • We will define the DSC metric
Train and Test	• You will train the architecture • You will test the trained U-Net

Thank you for your kind attention!
Questions?

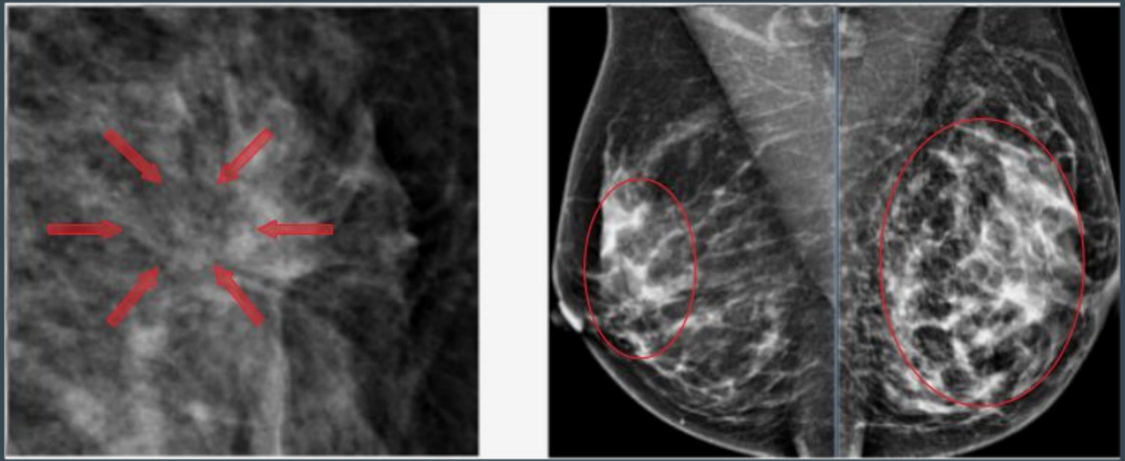
francesca.lizzi@pi.infn.it

Some examples:



Computed Tomography of a COVID-19 patient

Breast cancer signs on mammograms: left architectural distortion, right asymmetry



Undercomplete Autoencoder:

- **if the decoder is linear and the loss function is a MSE, an autoencoder learns to span the same subspace of the Principal Component Analysis (PCA);**

- **autoencoders with non-linear encoder (f) and decoder (g) can learn a “more powerful” generalization of the PCA.**

Interesting exercise to see this behaviour:
https://towardsdatascience.com/autoencoders-vs-pca-when-to-use-which-73de063f5d7
https://towardsdatascience.com/dimensionality-reduction-with-autoencoders-versus-pca-f47666f80743

We can imagine to build a very complex (high capacity) autoencoder and use a hidden layer of dimension 1 -> what will happen in this case?

Sparse autoencoder

Informally, we think that a neuron is “active” if its output value is close to 1 or “inactive” if it is close to 0: in a sparse AE we constrain the neurons of the hidden layer to be inactive for most of the time.

We insert in the loss function a SPARSITY penalty $\Omega(\mathbf{h})$ on the code layer h:

$$L(x, g(f(x))) + \Omega(\mathbf{h})$$

Let’s $a_j^{(2)}(x)$ be the activation of the neuron j in the hidden layer with respect to an input X. We can define the average activation as:

$$\hat{\rho}_j = \frac{1}{m} \sum_{i=1}^{m} \left[a_j^{(2)}(x^{(i)}) \right]$$

$$\hat{\rho}_j = \rho,$$

ρ is called SPARSITY PARAMETER

Sparse autoencoder

We can now choose sparsity penalty to be added to the loss function.

$$L(x, g(f(x))) + \Omega(h)$$

There are many choices, for example:

$$\sum_{j=1}^{s_2} \rho \log \frac{\rho}{\hat{\rho}_j} + (1-\rho) \log \frac{1-\rho}{1-\hat{\rho}_j}.$$

This penalty function can be written as:

$$\sum_{j=1}^{s_2} \mathrm{KL}(\rho || \hat{\rho}_j),$$

$$\mathrm{KL}(\rho || \hat{\rho}_j) = 0 \text{ if } \hat{\rho}_j = \rho,$$

and otherwise it increases monotonically as $\hat{\rho}_j$ diverges form ρ.