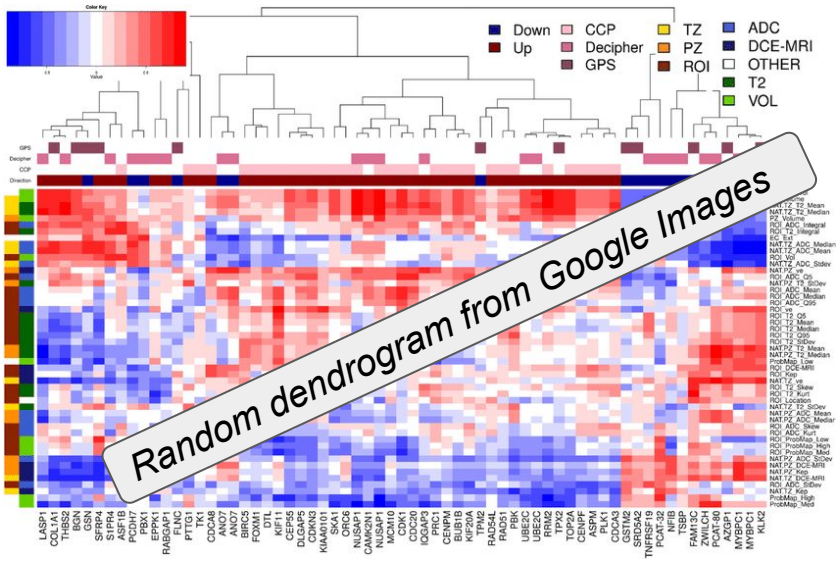
First steps towards *Deep-Learning based dosiomics*

Lucio Anderlini, Stefano Piffer, Cinzia Talamonti

Istituto Nazionale di Fisica Nucleare
SEZIONE DI FIRENZE

Introduction

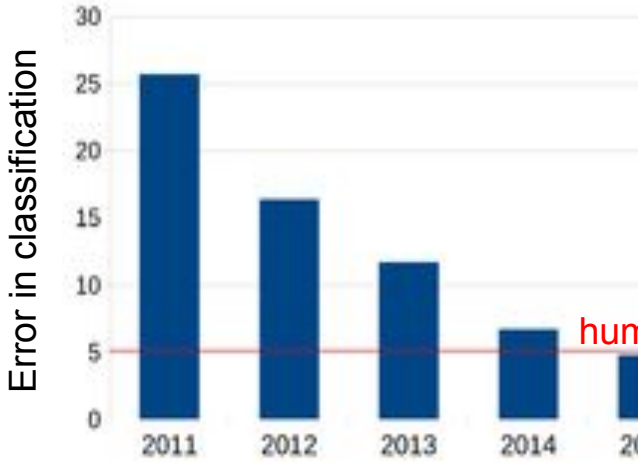
- Radiomics and dosiomics try to make predictions on the outcome of radiotherapy from the analysis of clinical images
- Several state-of-the-art techniques based on *radiomics features*: standardized formulas to extract numbers from the images. Formulas are made available for data-science for example through *pyradiomics*.
- The many extracted features are studied with *correlation diagrams* and *dendrograms* to establish the most relevant features in the forumlation of a given problem.
- Neural networks and Boosted Decision Trees are sometimes used in the literature to combine radiomics features in higher level features with higher correlation



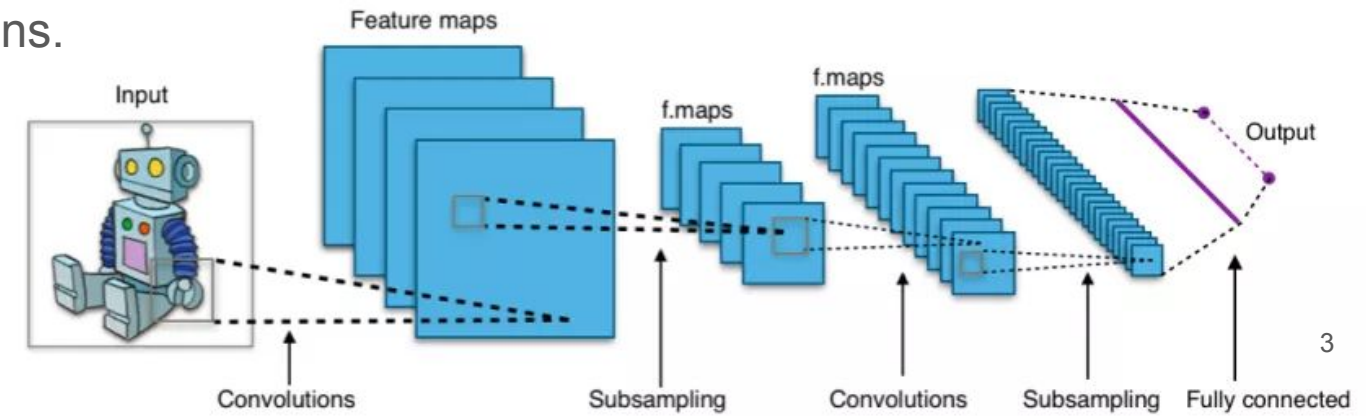
Why *Deep-Learning?*

Combining standard radiomics features into machine-learned higher level variables implies loss of information (see (J. Lao, 2017)).

This is a well-known problem in image processing, where **Convolutional Neural Networks** trained with thousands of images to distinguish objects outperform humans.



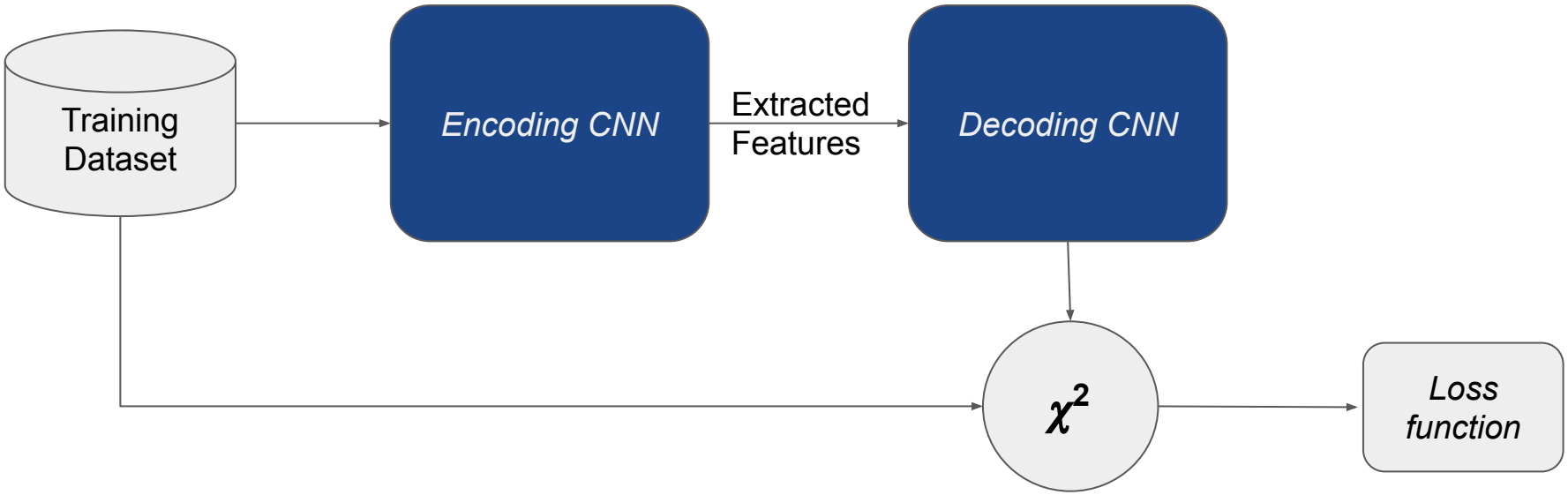
CNNs are designed to ***extract features*** with *machine-learned formulas* combining "filters" on the spatial frequencies of the images, subsampling and matrix multiplications.



Representation Learning

The big challenge with CNNs is the need of huge samples to train the feature extraction. While obtaining images is very cheap, obtain reliable labels (the ideal output of the neural network) of those feature is incredibly expensive.

The revolution for image processing was the *semisupervised* techniques to *"pre-train"* the neural networks on datasets without labels.



CNNs for Medical Images

Training requires a huge amount of images "*sharing the same conditions".*

The underlying idea is to factorize out of the features all the information which is common among the images which can be encoded in the network structure, rather than in the extracted features. If images are different for reasons unrelated to differences in the subject, the extracted feature must encode that difference too, wasting predictive power.

Such large and homogeneous samples are not easy to obtain for clinical images (different machines, different hospitals, different clinicians…)

(J. Lao, 2017) reuses the very same networks used for image recognition from the web as a starting point to train networks processing the clinical images.

Doing this they constrain the algorithm to *ignore* the correlation between two different "slices" of the clinical images (web images are 2D, TCs are 3D).

1st exercise: *learn the features from the 3D TCs*

Thanks to the large Database we own, we have the data to attempt a full *semisupervised* approach to the extraction of features from a homogeneous dataset.

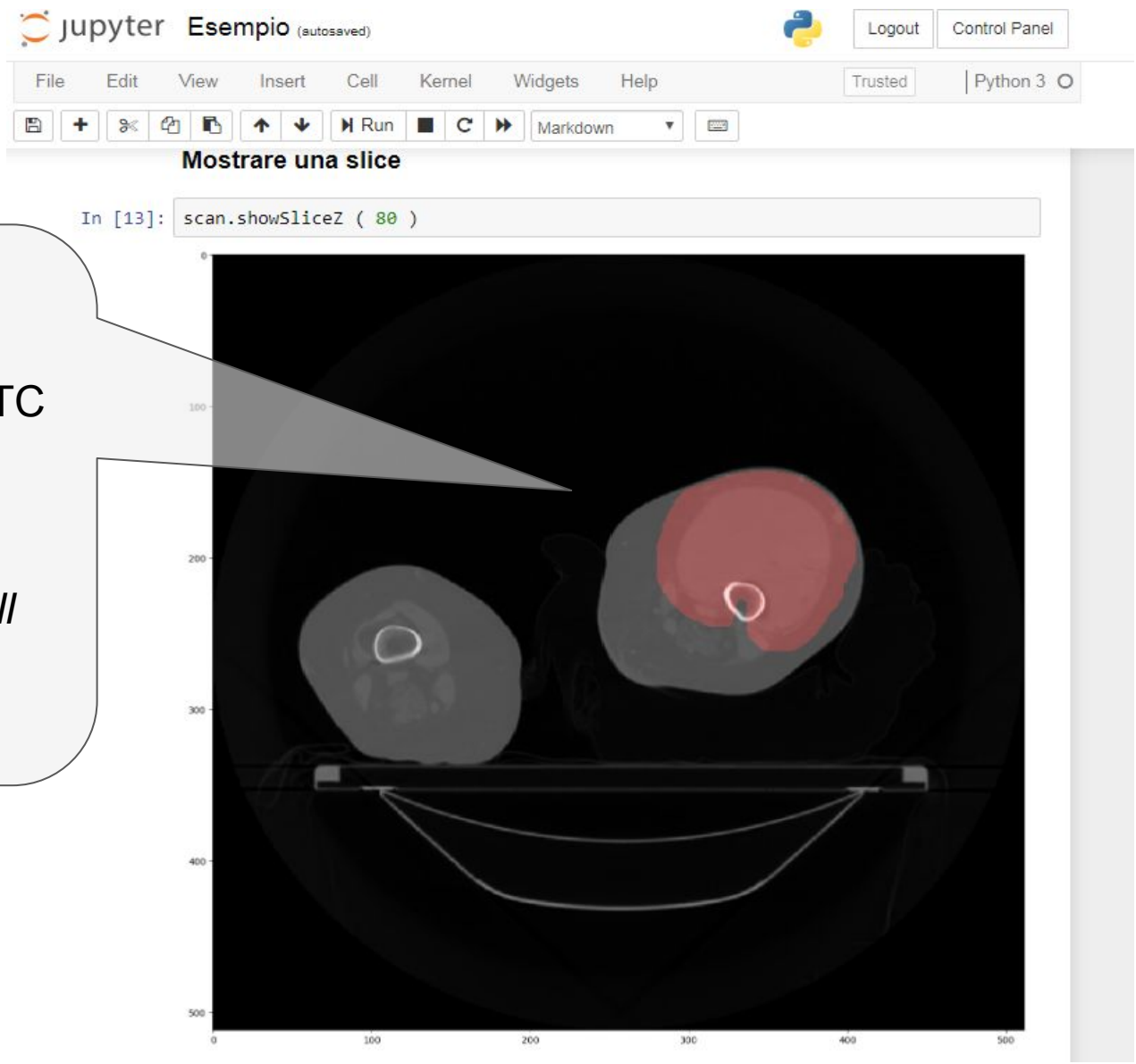
The extracted features will be then be combined to other clinical exams to assess the improvement in the prediction power obtained from the image analysis.

Python & Jupyter to analyse TCs

Visualization of images with Jupyter

Alignment of structures with the TC implemented in our custom library.

(with some bugs still to be fixed…)



Extraction of radiomics features

To ensure the correctness of the implementation, we computed the features with *pyradiomics* on the images and structures loaded with our custom library and proved they are consistent with those obtained with widely used Slice3D Software.

Features are always computed on a given structure.

In the table some features computed on some structures for a given patient.

Out[5]:

	cuore	polmoneSX	polmoneDX	esofago	midollo
InterquartileRange	5.000000e+01	2.840000e+02	2.780000e+02	5.400000e+01	4.800000e+01
Mean	2.096666e+01	-5.766873e+02	-5.919472e+02	1.587369e+01	2.209071e+01
Uniformity	1.768847e-01	4.731733e-02	4.703085e-02	1.624884e-01	1.876466e-01
Entropy	2.844653e+00	4.719556e+00	4.763380e+00	3.033391e+00	2.755337e+00
Kurtosis	5.551677e+00	3.311337e+00	3.931271e+00	5.618807e+01	1.564587e+01
RootMeanSquared	5.082254e+01	6.100237e+02	6.276299e+02	7.181131e+01	4.919388e+01
Variance	2.143330e+03	3.956069e+04	4.351778e+04	4.904890e+03	1.932039e+03
Minimum	-6.170000e+02	-1.020000e+03	-9.490000e+02	-9.780000e+02	-1.980000e+02
10Percentile	-4.100000e+01	-7.780000e+02	-7.960000e+02	-4.300000e+01	-2.600000e+01
Maximum	3.780000e+02	1.300000e+03	1.121000e+03	2.660000e+02	6.080000e+02
MeanAbsoluteDeviation	3.450775e+01	1.640590e+02	1.689574e+02	4.009981e+01	3.131847e+01
90Percentile	7.000000e+01	-2.630000e+02	-2.610000e+02	7.200000e+01	6.900000e+01
TotalEnergy	1.959795e+09	4.169745e+11	6.753427e+11	1.749365e+08	8.672123e+07
Skewness	-7.662455e-01	9.658407e-01	1.118008e+00	-5.138819e+00	1.104508e+00
RobustMeanAbsoluteDeviation	2.204044e+01	1.203734e+02	1.202565e+02	2.310854e+01	2.020143e+01
Median	2.800000e+01	-6.430000e+02	-6.630000e+02	2.300000e+01	2.200000e+01
Range	9.950000e+02	2.320000e+03	2.070000e+03	1.244000e+03	8.860000e+02
Energy	4.756932e+08	1.012105e+11	1.639232e+11	4.246162e+07	2.104949e+07

First steps towards a 3D CNN

With a teen of patients we started to build the system to create the neural network and *pretrain* it via the *semisupervised approach.*

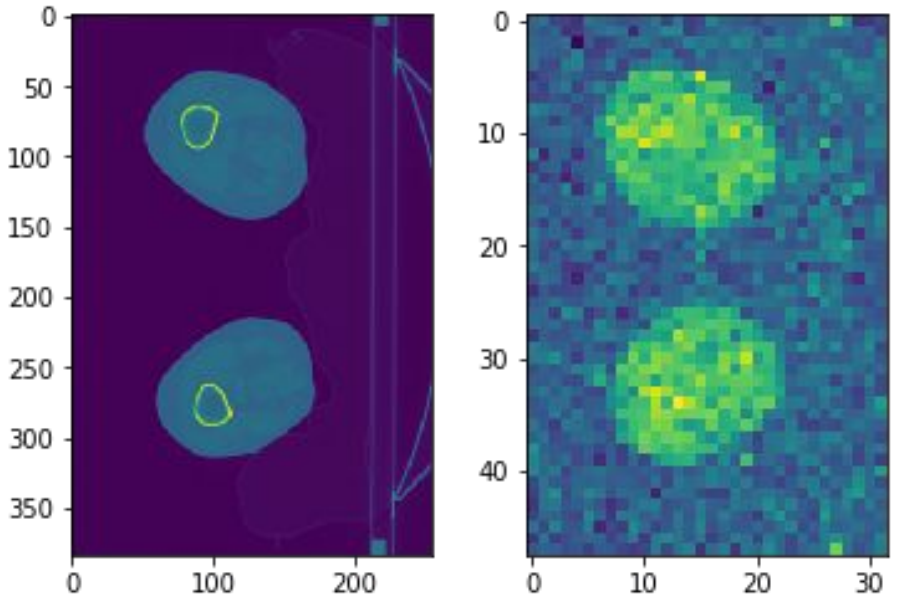
A relatively small model was implemented to:

Channels:
image, PTV, dose

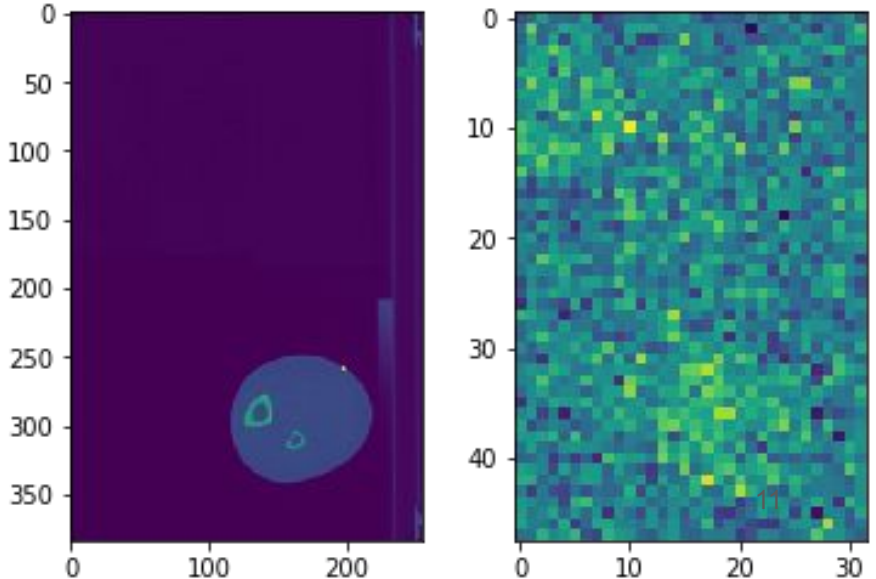
- load the images and the structures for PTV and DOSE from disk
- compile them in a unique batch of shape (N patients, Nx, Ny, N slices, 3)
- preprocess the batch (scale it into the interval (0,1))
- define up to 5 *gray scale windows* to magnify some features
- apply a set of filters, subsampling and matrix multiplications
- define the loss as a simple χ^2 of the original image vs. the output of the NN
- train with the Adam Optimizer
- store the trained model on file

Results

We compare the image obtained from the decoding of the 10 features with the original and we can recognize the *silhouette* of the legs and a slight increase in the luminosity in the region of the bones.



The network is however fooled when we present a patient with a single legs.



10 features are probably too few to spend one on the *number of legs* since most patients have 2.

On resources

A full TC image of a single patient is described by a floating point tensor (512 x 512 x 350).

When applying 5 gray scale windows this becomes (512 x 512 x 350 x 5)

Associating to each point, also the dose and the interpretation of the clinician this becomes (512 x 512 x 350 x 7)

Filtering requires to multiply the whole tensor for each weight of the *filter.* A tiny filter can consider (2 x 2 x 2) = 8 voxels, which require a tensor (512 x 512 x 350 x 7 x 8) to be allocated.

To combine different filters (*e.g. horizontal and vertical*) one needs to allocate a different tensor for each *output channel* (or filter). Using 5 filters requires (512 x 512 x 350 x 7 x 8 x 5)

Considering 32bit floating points (4 bytes), this requires 100 GB of RAM to be allocated. Anything more (additional or larger filters) results into swapping.

Conclusion

We have the infrastructure to run simple CNNs on images from TCs combined with contours from the clinicians and dose planning.

We have shown that the principle works: using 10 features we are able to encode the *silhouette* of the patient.

The limiting resource is RAM (~100 GB available on acer-1 are barely sufficient for small filters which can not deal with complex correlations in the high dimensionality problem we are defining).

We need to simplify the problem, for example cutting the borders of the image or downsampling the image to lower resolution…

A N D / O R

to rewrite the convolution algorithm to be slower, but less RAM greedy.