Implémentation d'algorithmes de type IA : exemples d'applications en imagerie biomédicale

Yannick Boursier
Centre de Physique des Particules de Marseille

Outline

1 - Biomedical imaging context : Computerized tomography (CT Scan)

2 - Convolutional Neural Networks (CNNs) for imaging

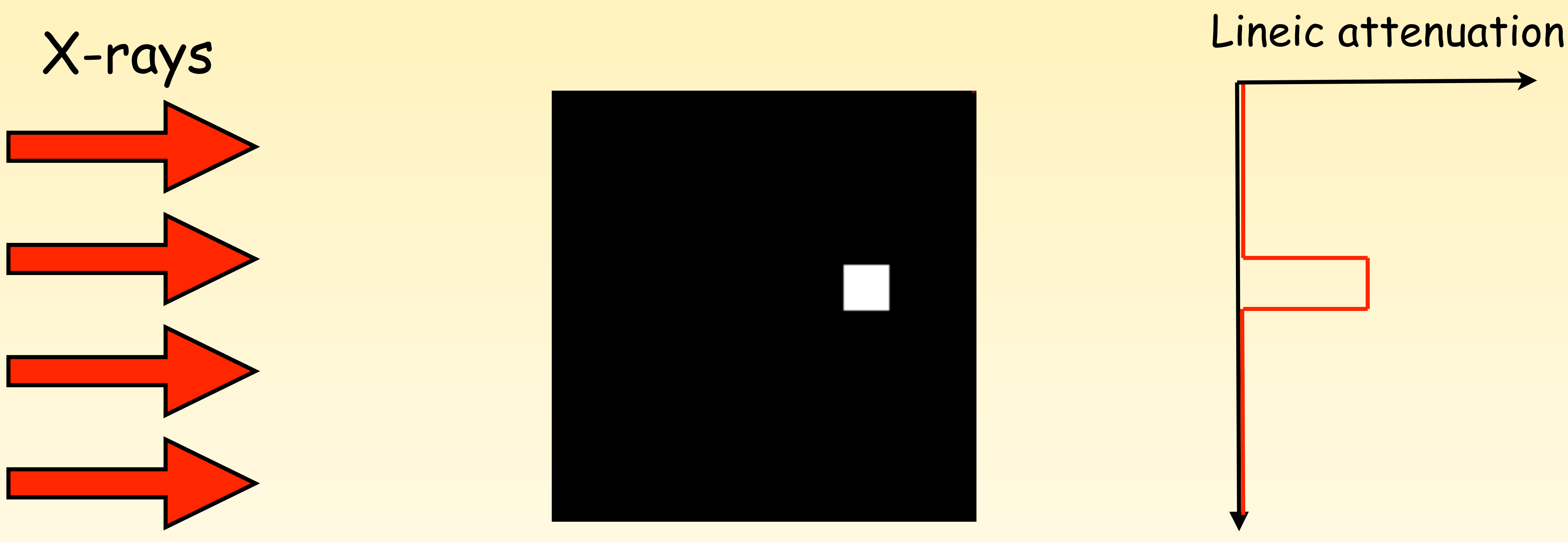
3 - Denoising issues

4 - Segmentation issues

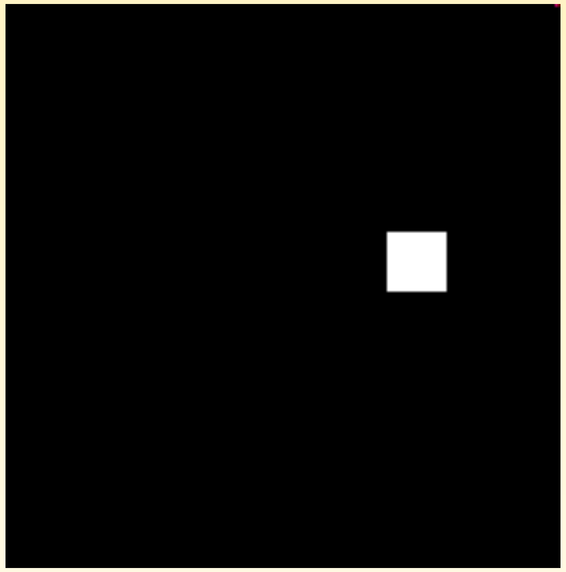
5 - Other issues

6 - Conclusion

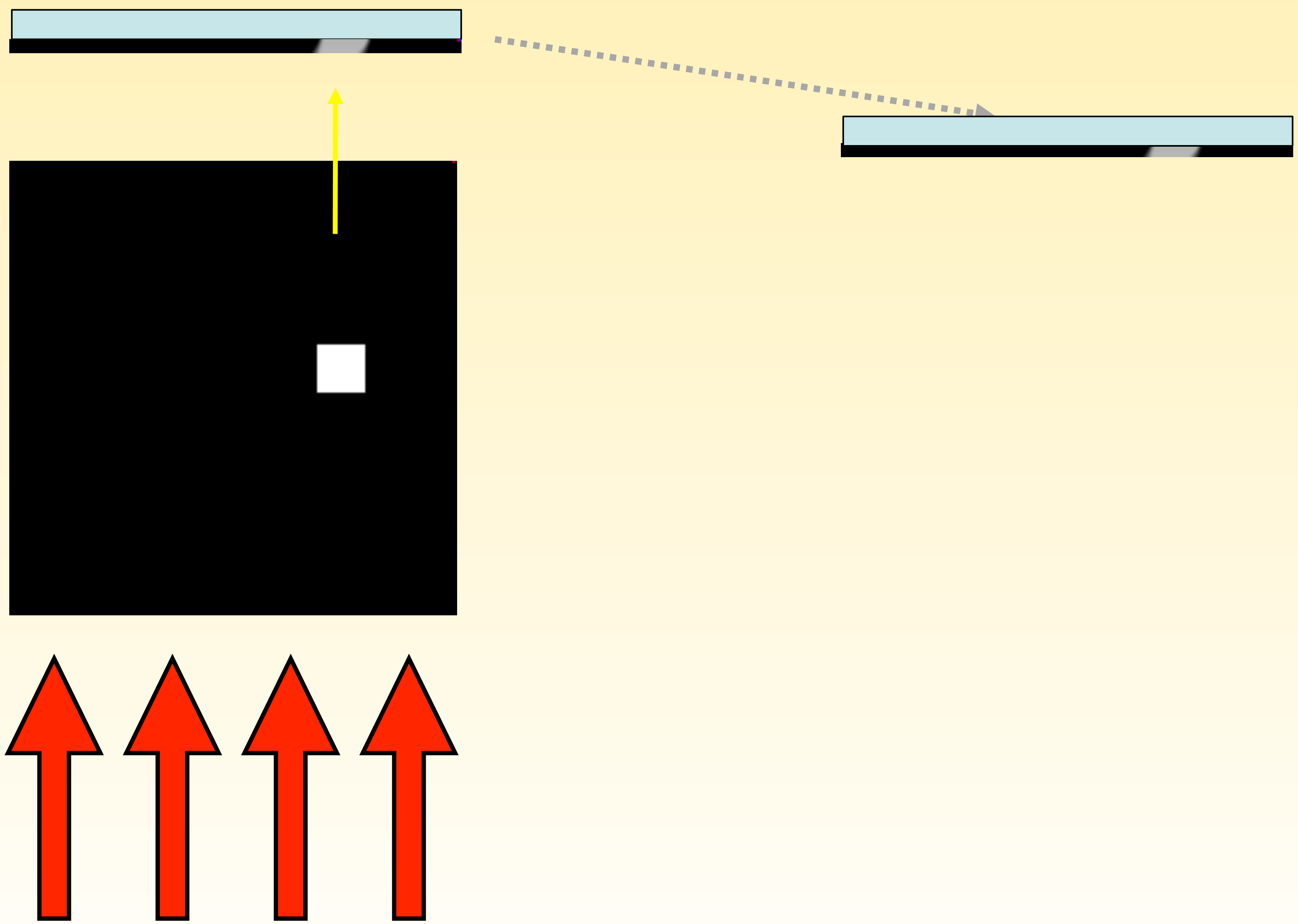
Snapshot of standard tomography (CT)



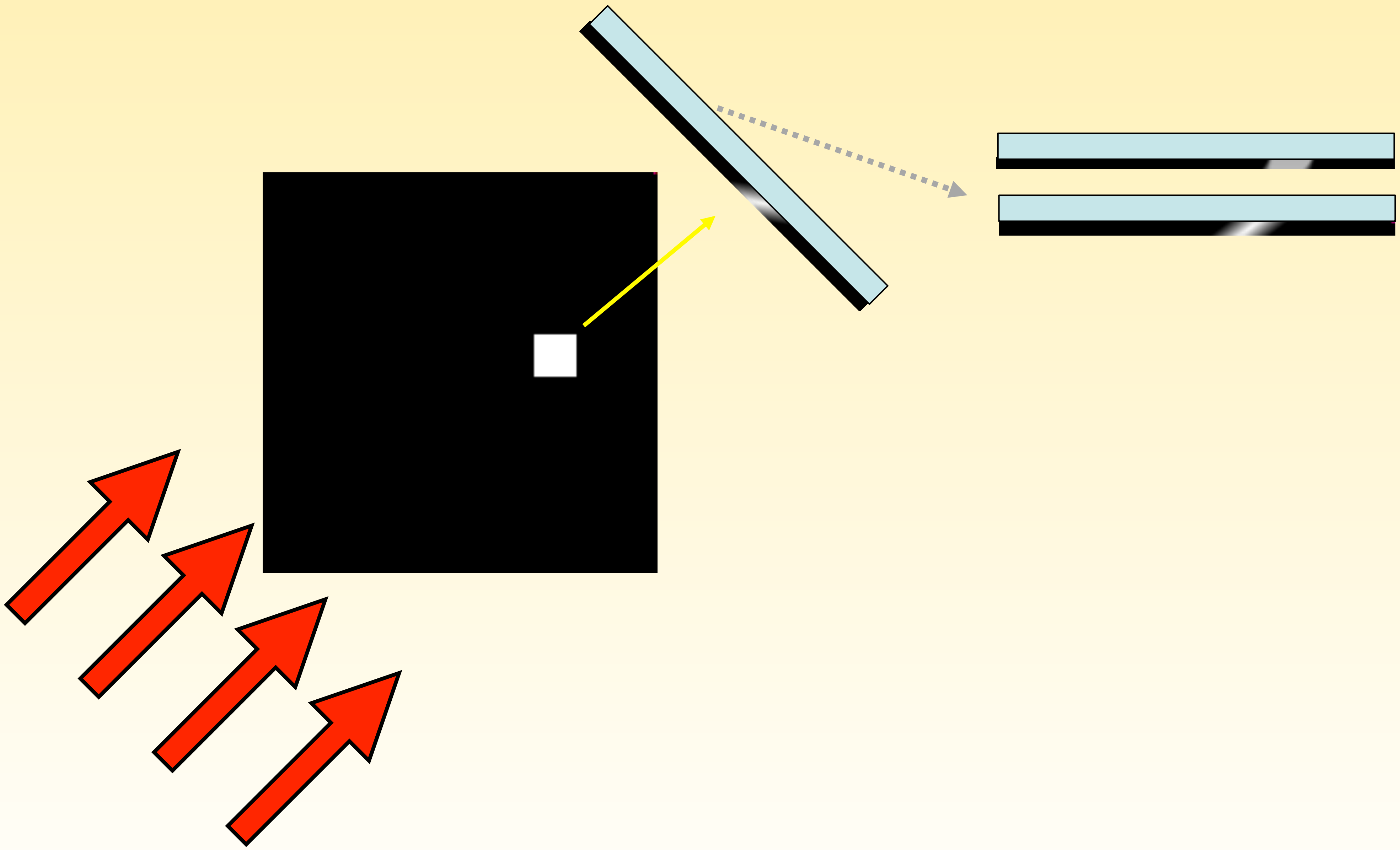
Snapshot of standard tomography (CT)
2D sinogram



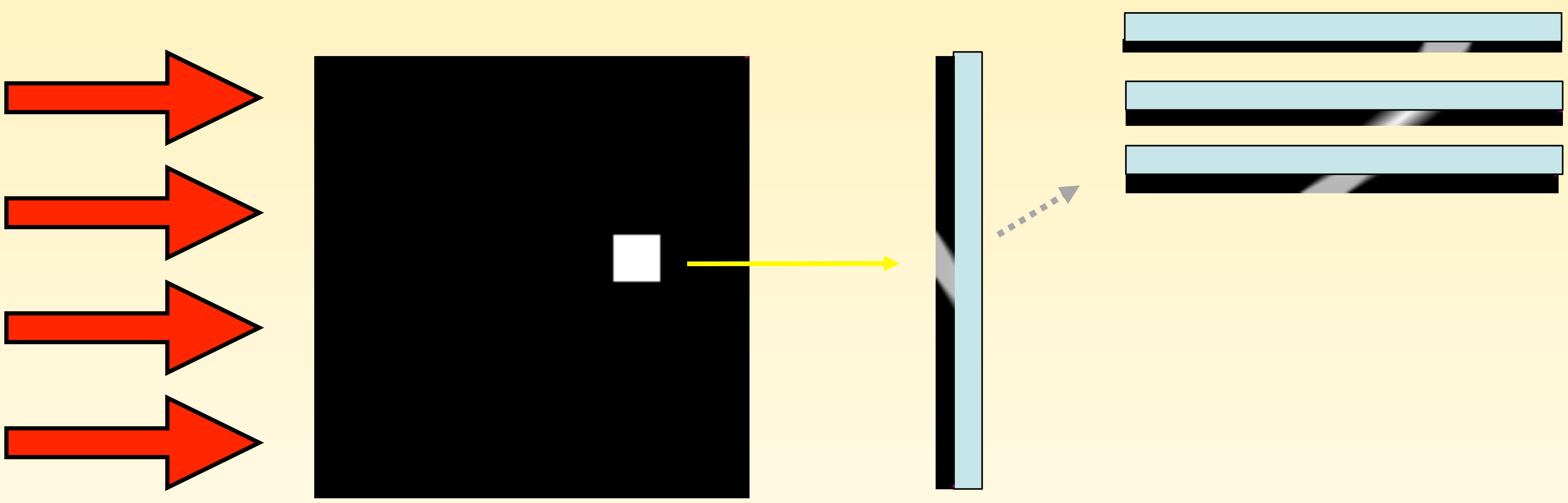
Snapshot of standard tomography (CT)
2D sinogram



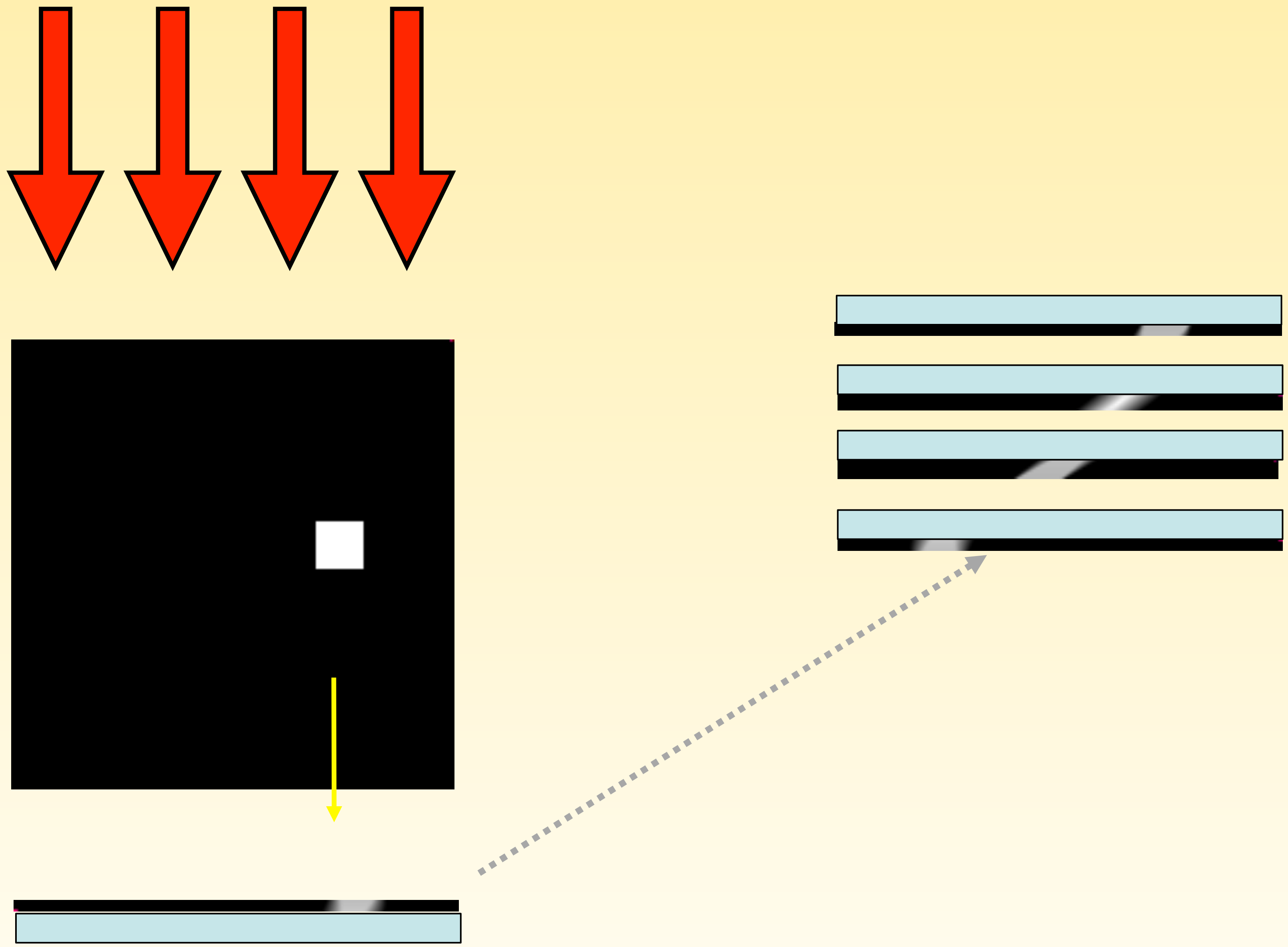
Snapshot of standard tomography (CT)
2D sinogram



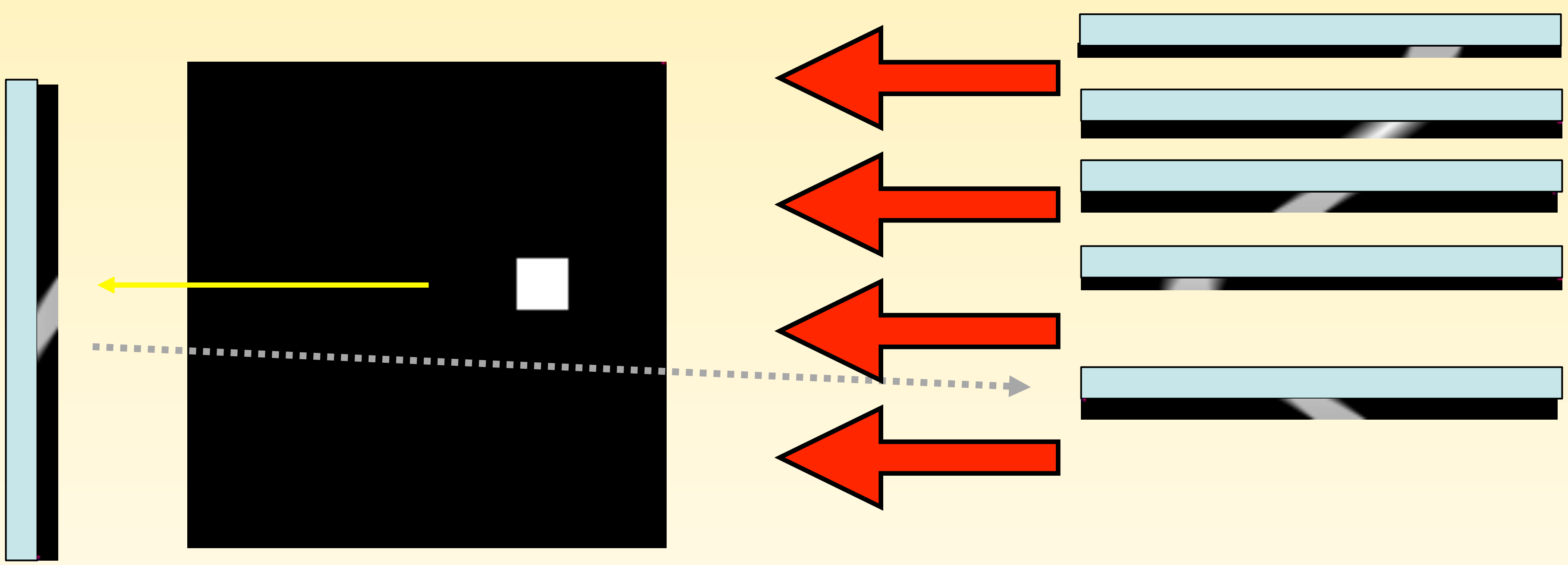
Snapshot of standard tomography (CT)
2D sinogram



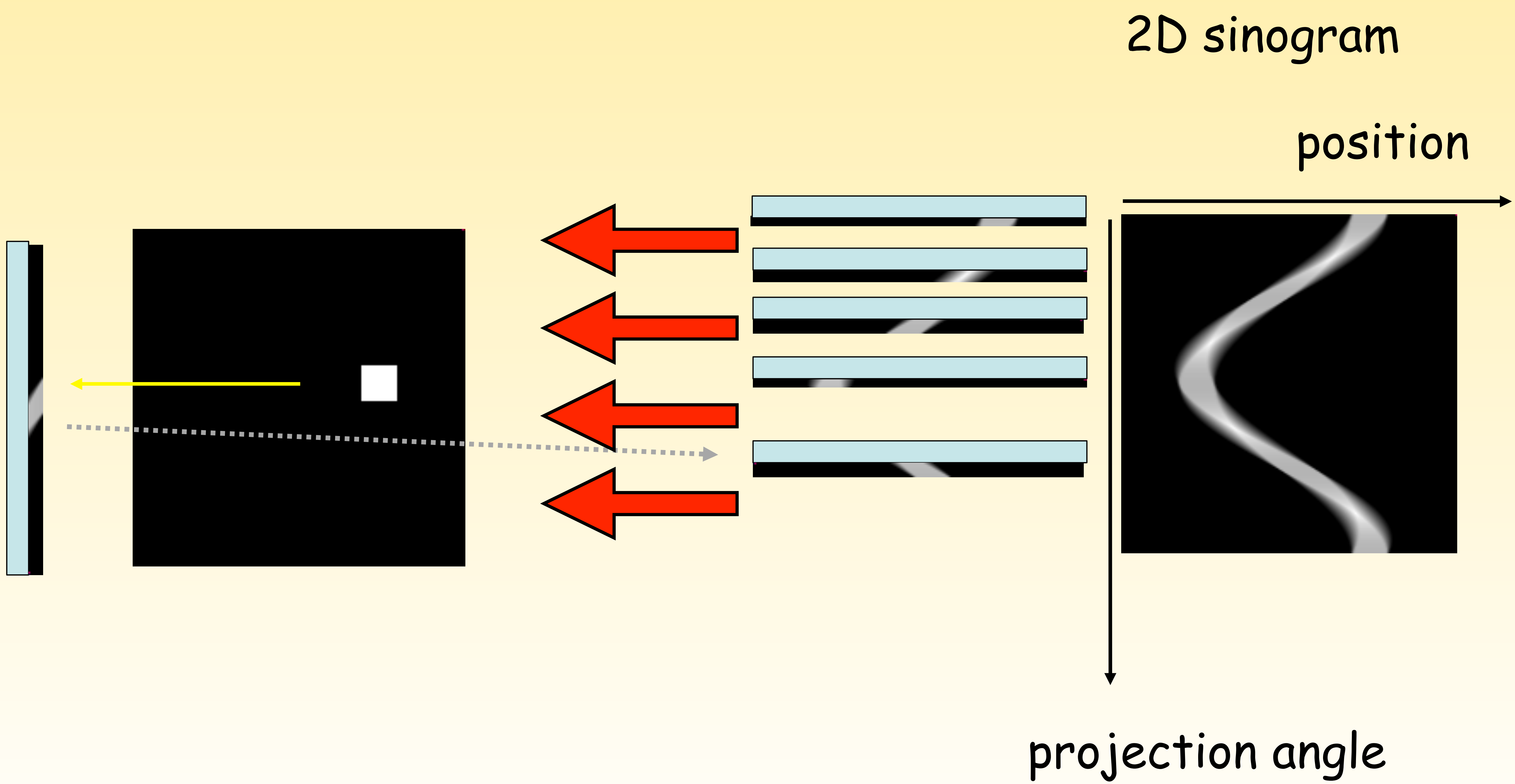
Snapshot of standard tomography (CT)
2D sinogram



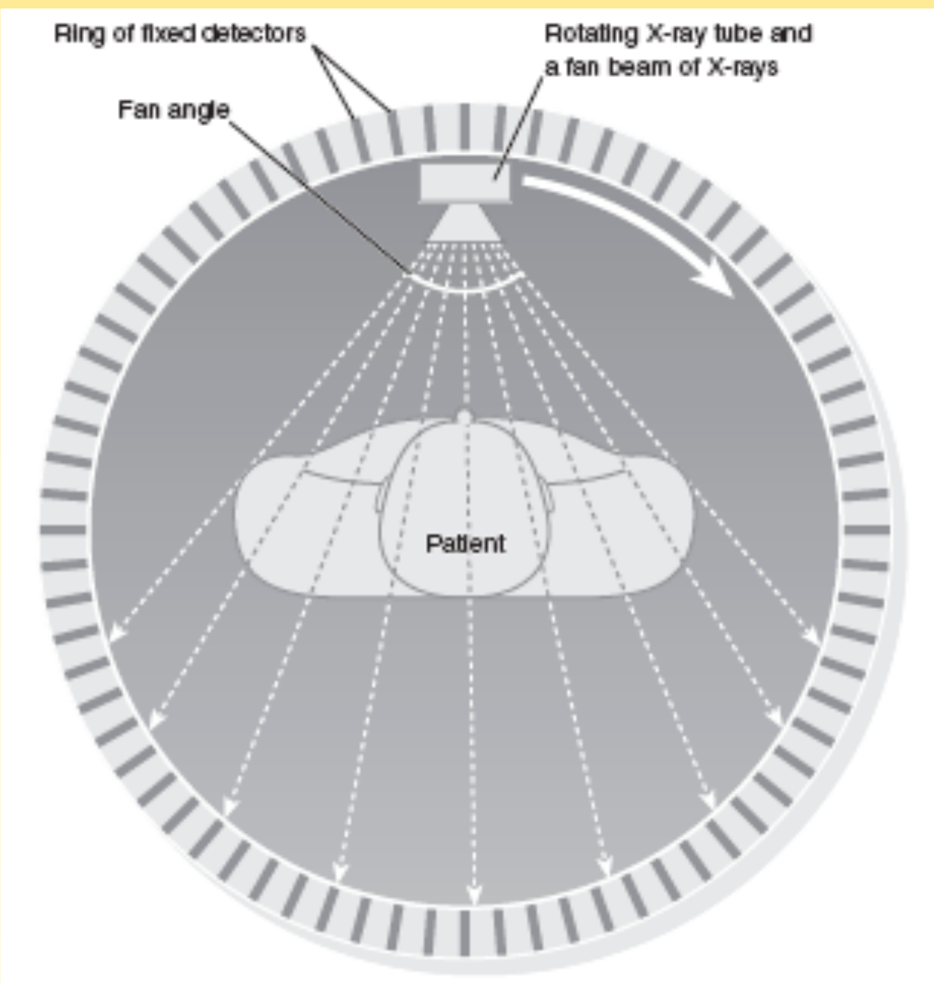
Snapshot of standard tomography (CT)
2D sinogram

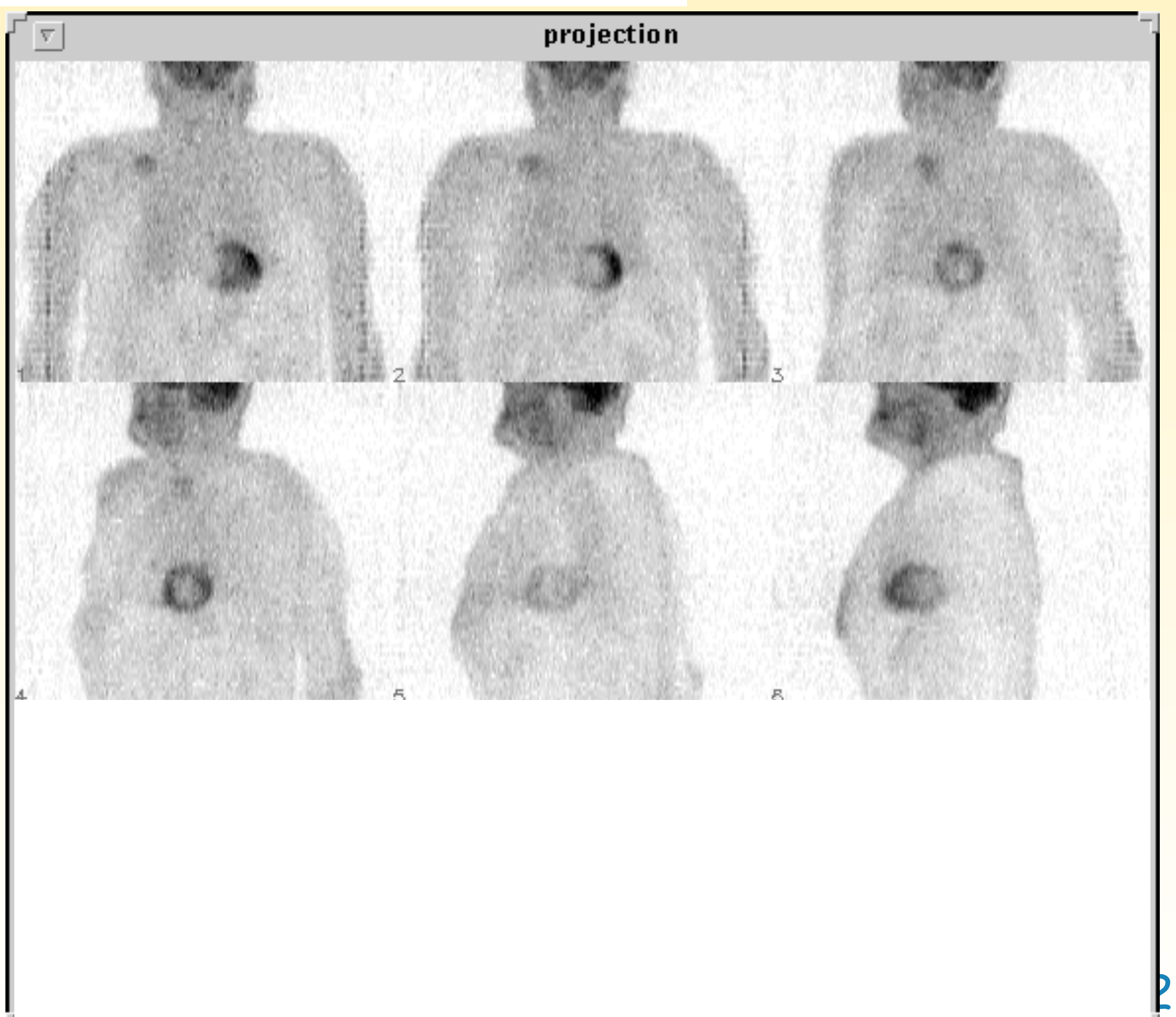


Snapshot of standard tomography (CT)
2D sinogram



Snapshot of standard tomography (CT)
3D sinogram



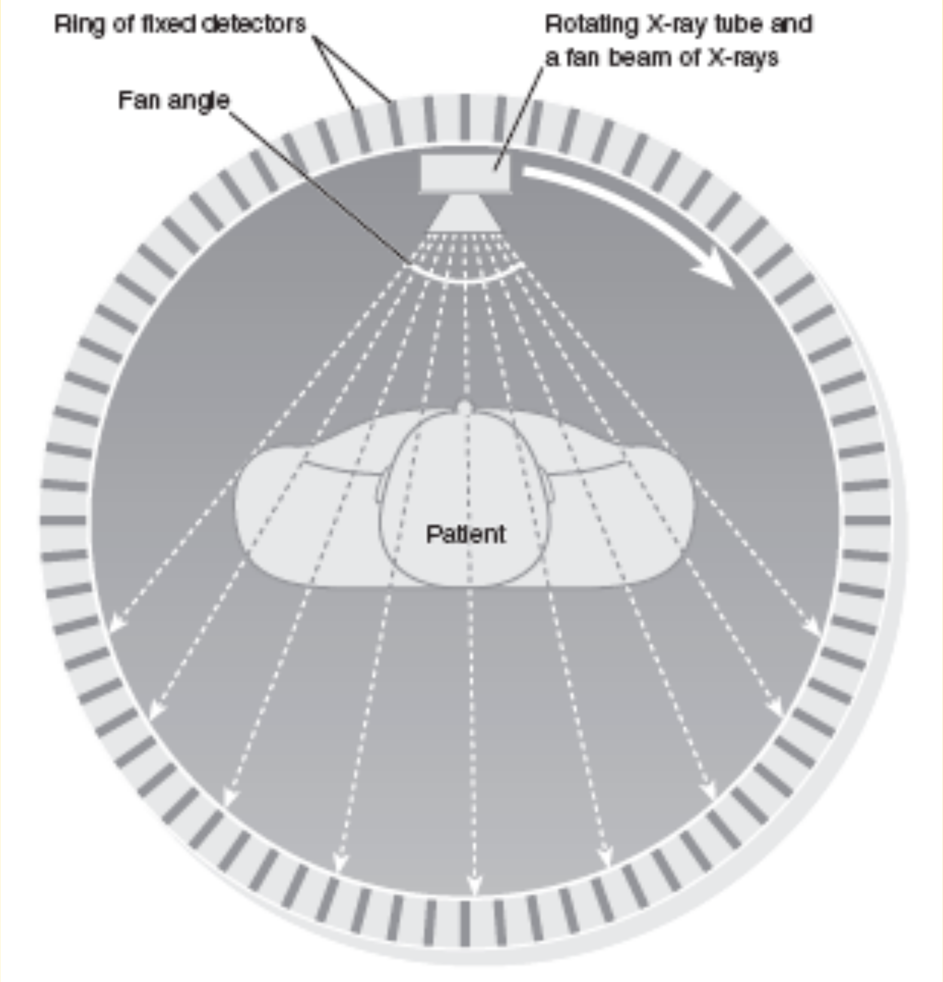


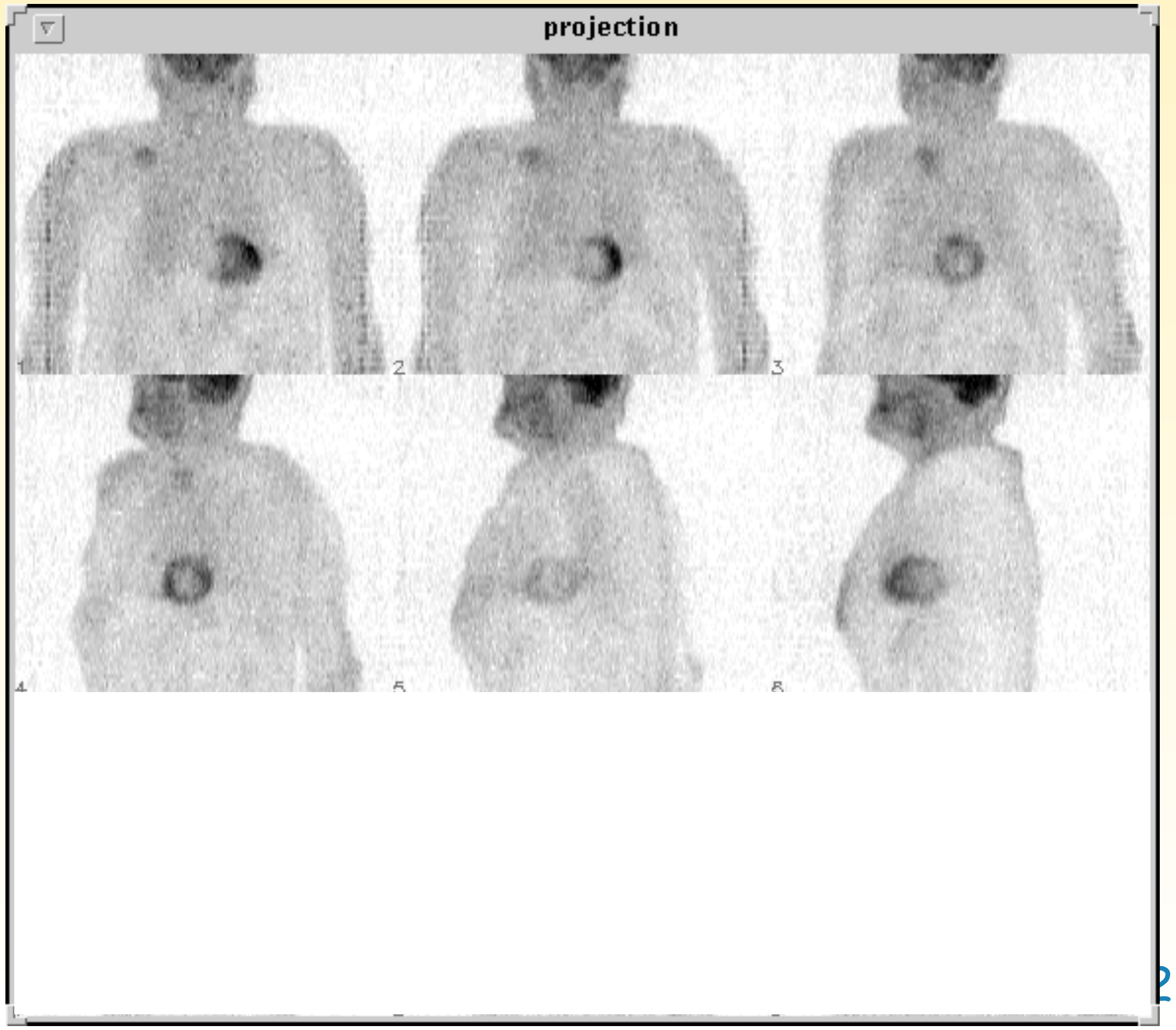
CENTRE DE PHYSIQUE DES PARTICULES DE MARSEILLE
CPPM

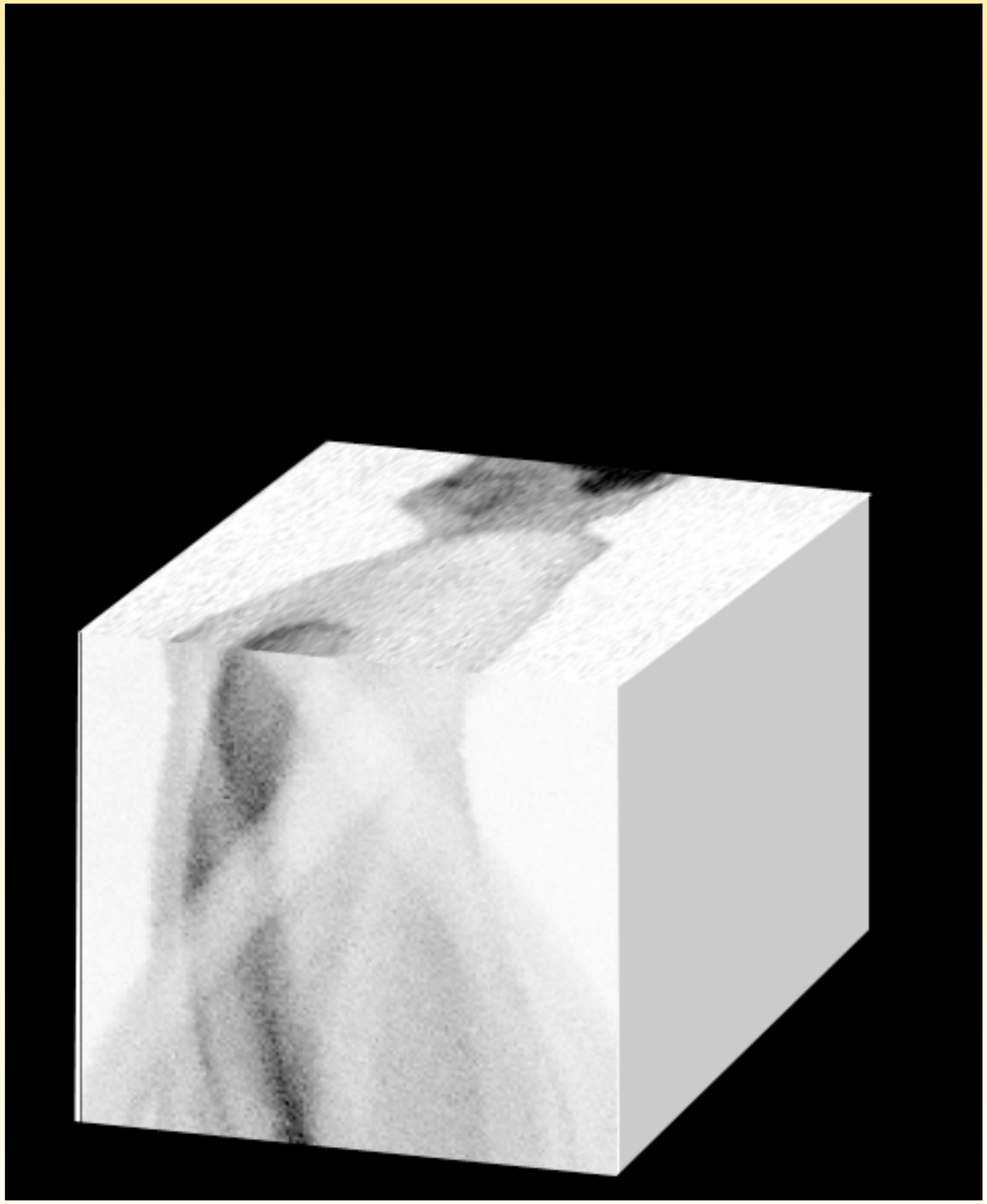
Aix*Marseille université
Initiative d'excellence

cnrs
IN2P3
Les deux infinis

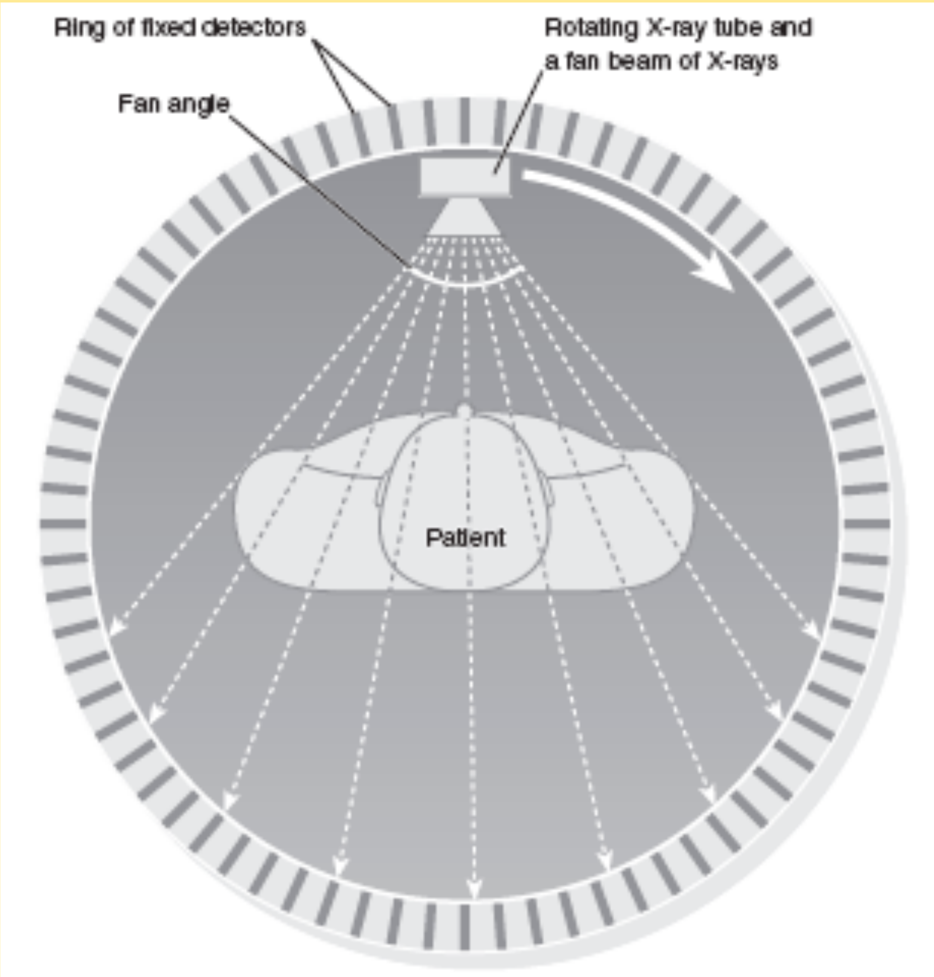
Snapshot of standard tomography (CT)
3D sinogram

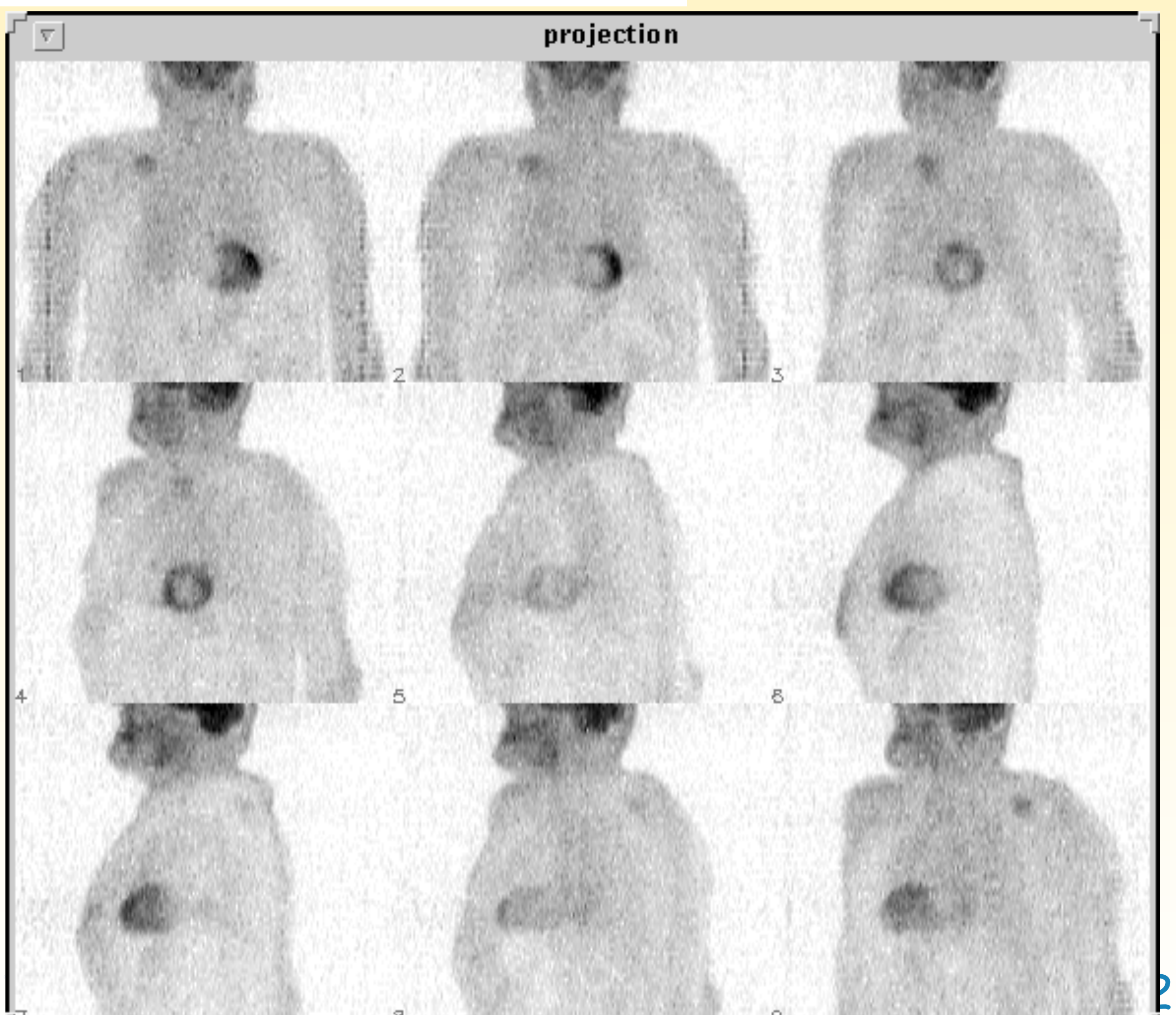


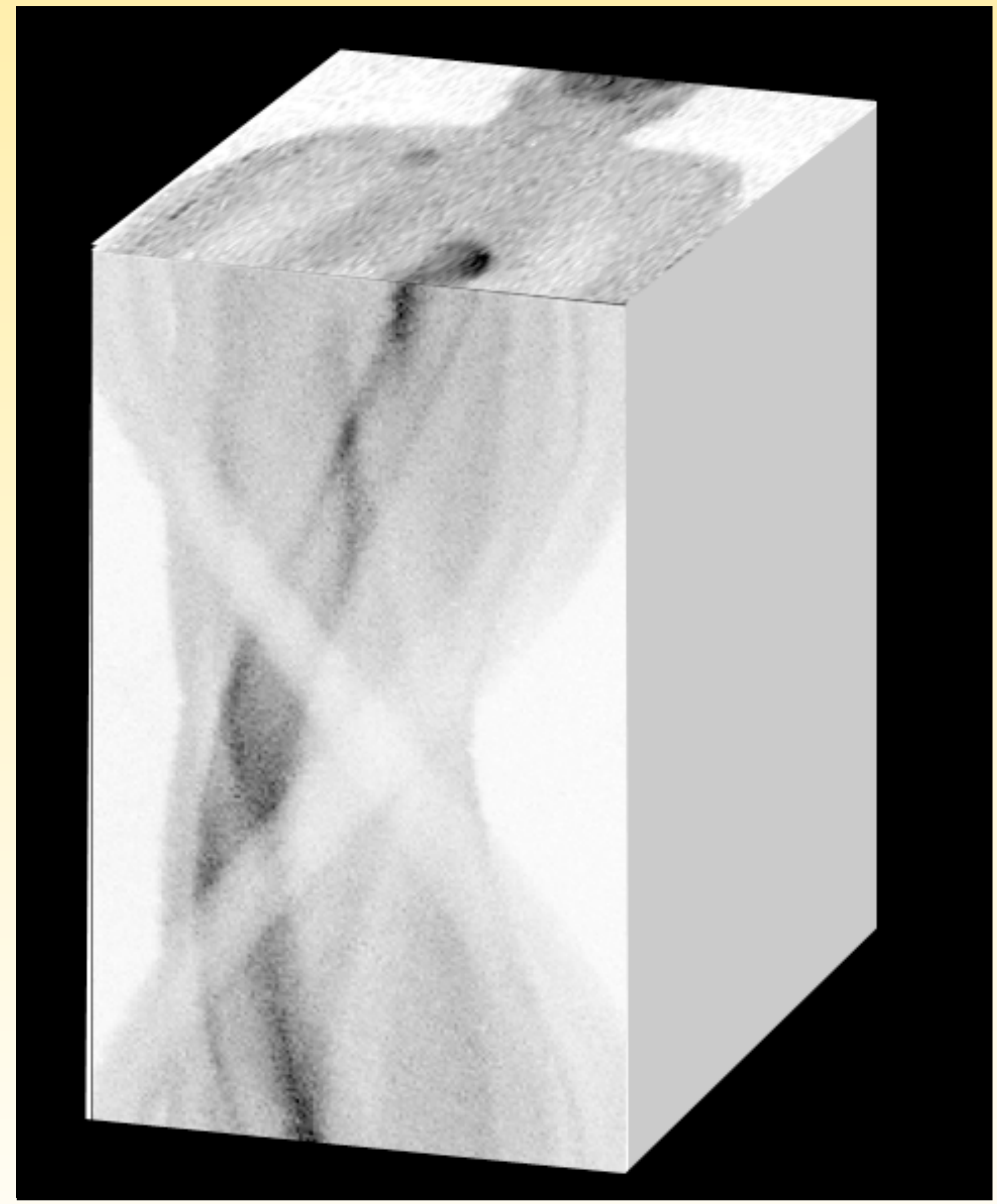




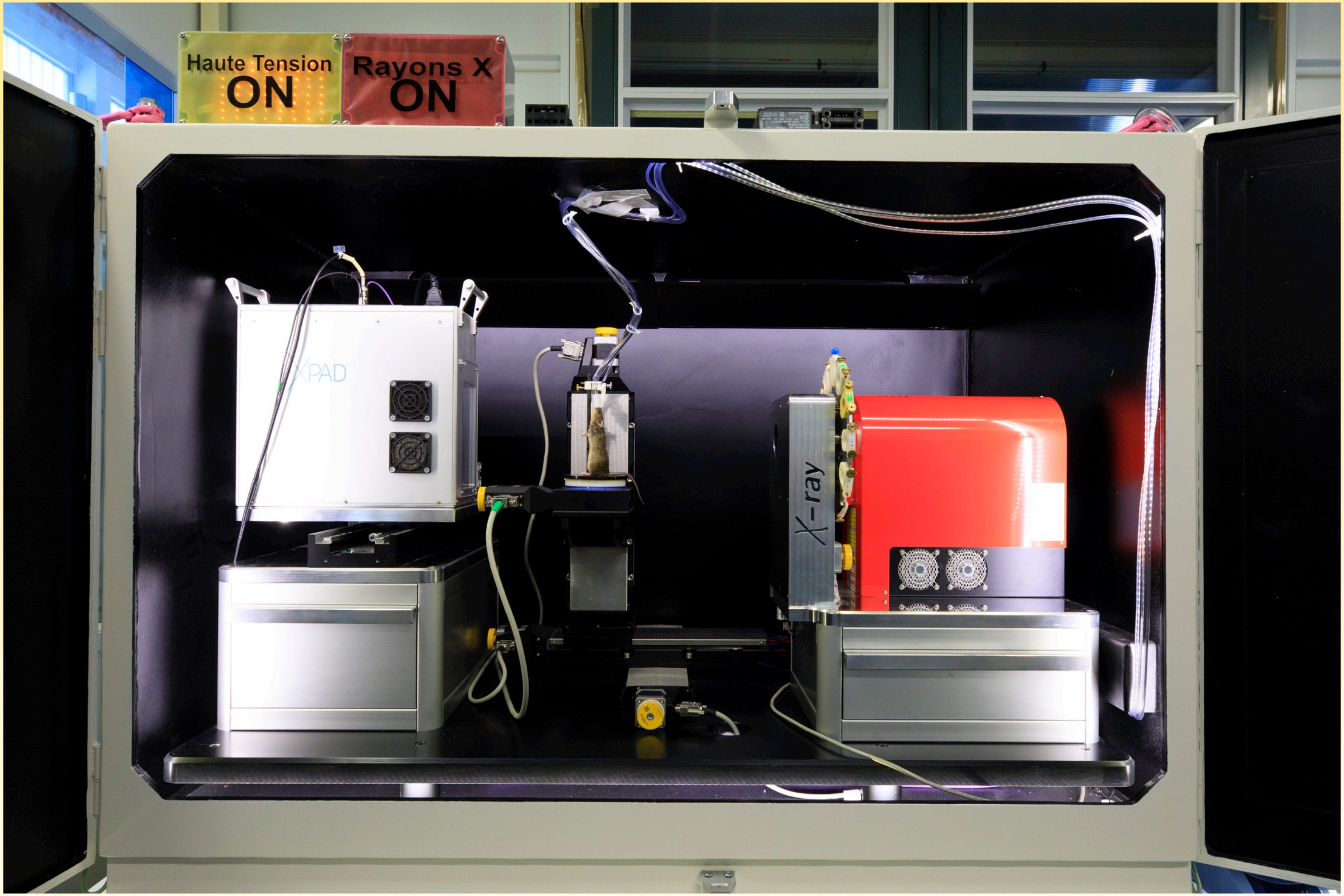
Snapshot of standard tomography (CT)
3D sinogram



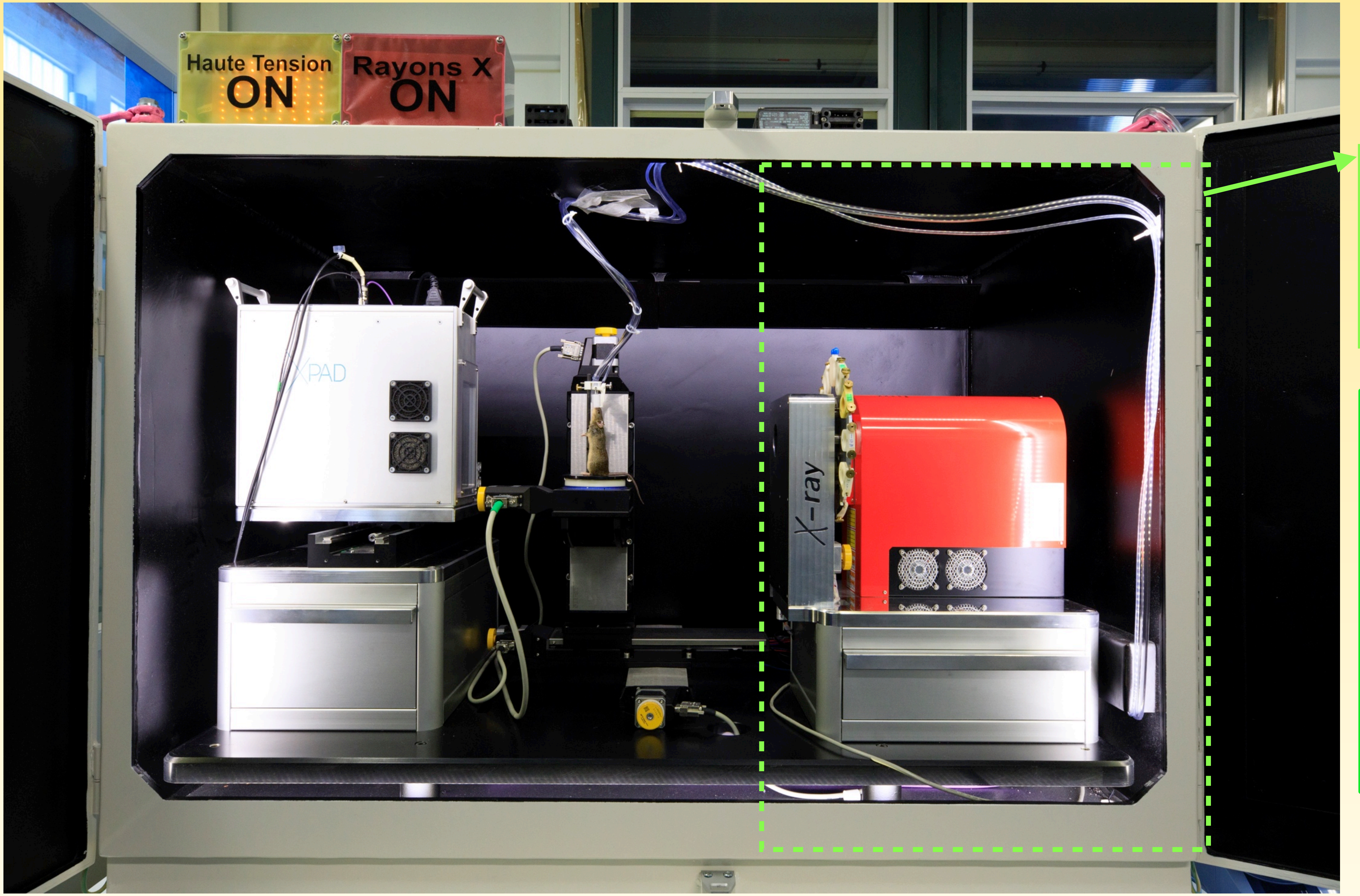




PIXSCAN-FLI

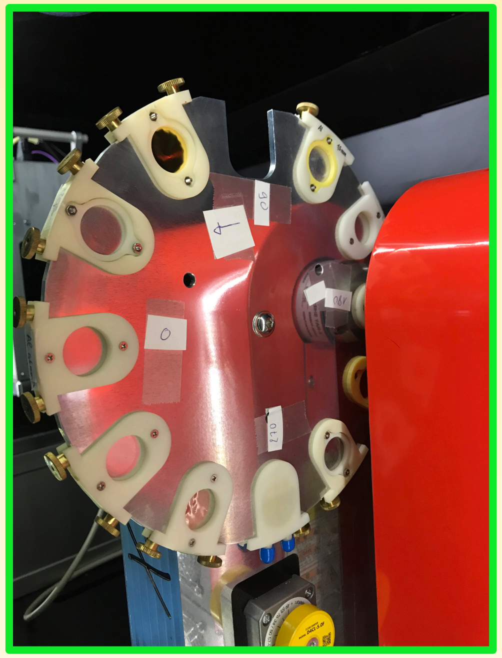


PIXSCAN-FLI

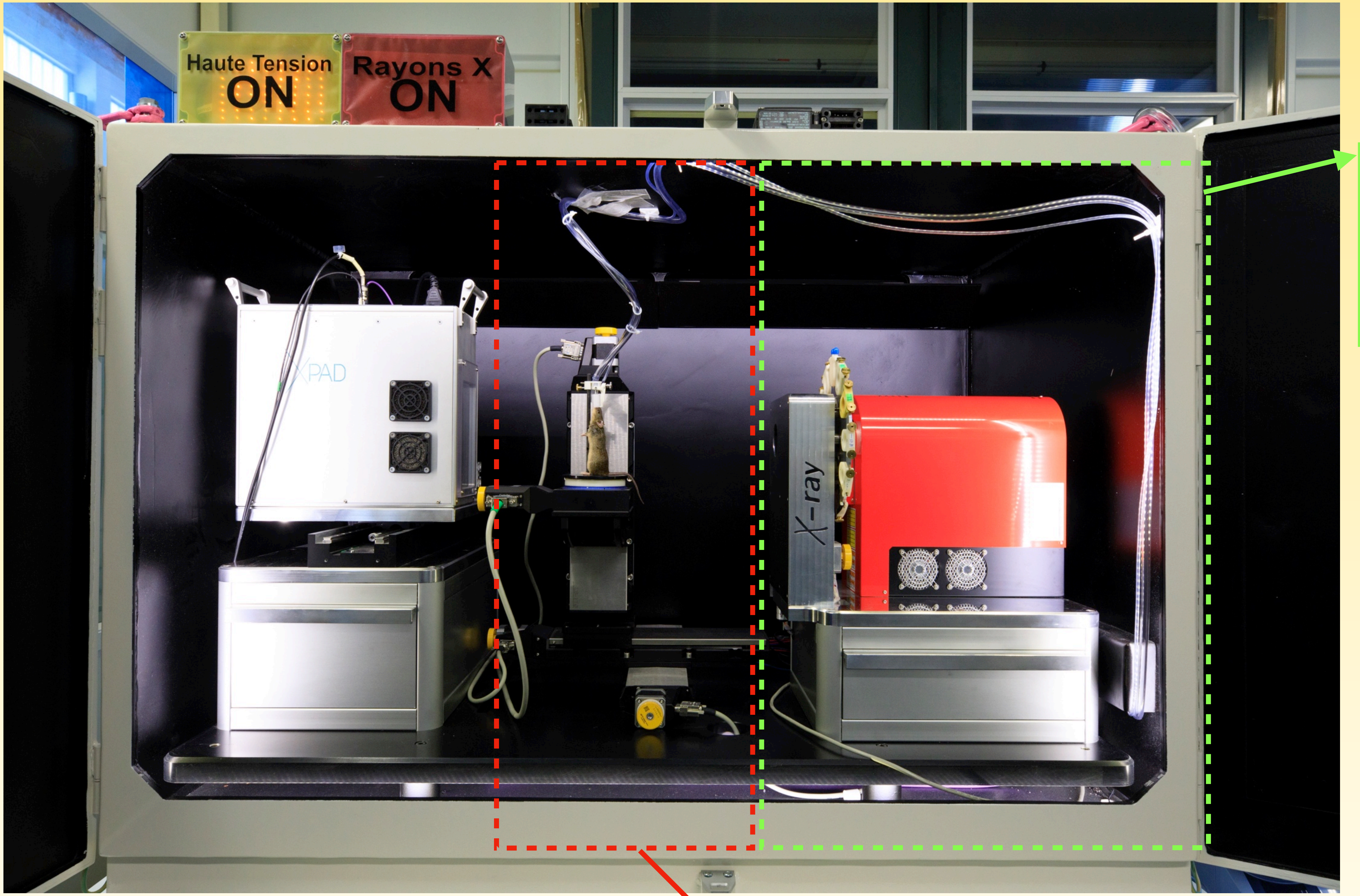


Source block

W anode (150 kVp)
Wheel filters



PIXSCAN-FLI

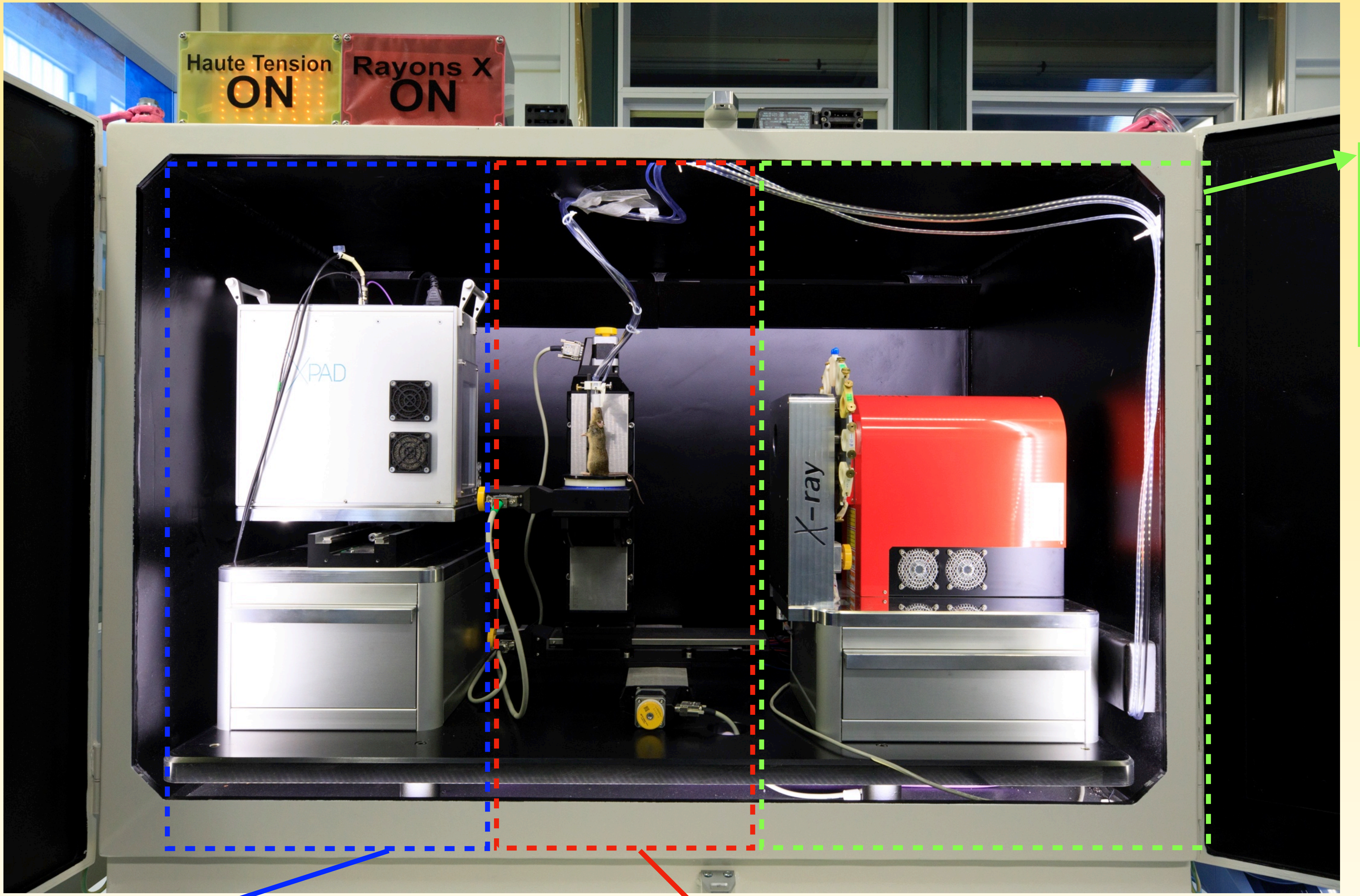


Source block

W anode (150 kVp)
Wheel filters

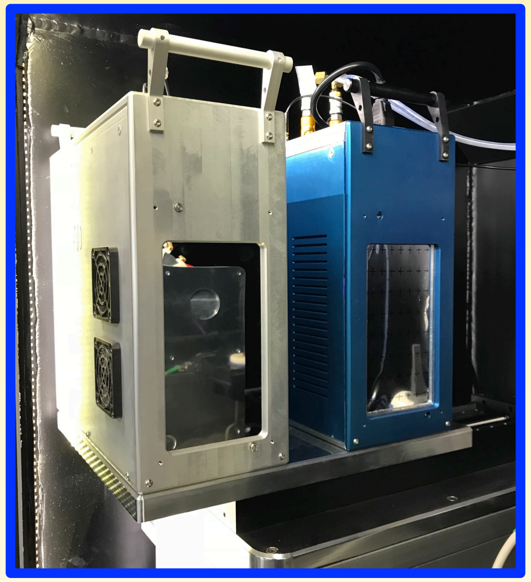
Animal block
Vertical rotation axis
Acquisition mode: shoot and step or continuous rotation
Gas anesthesia: isoflurane

PIXSCAN-FLI



Source block

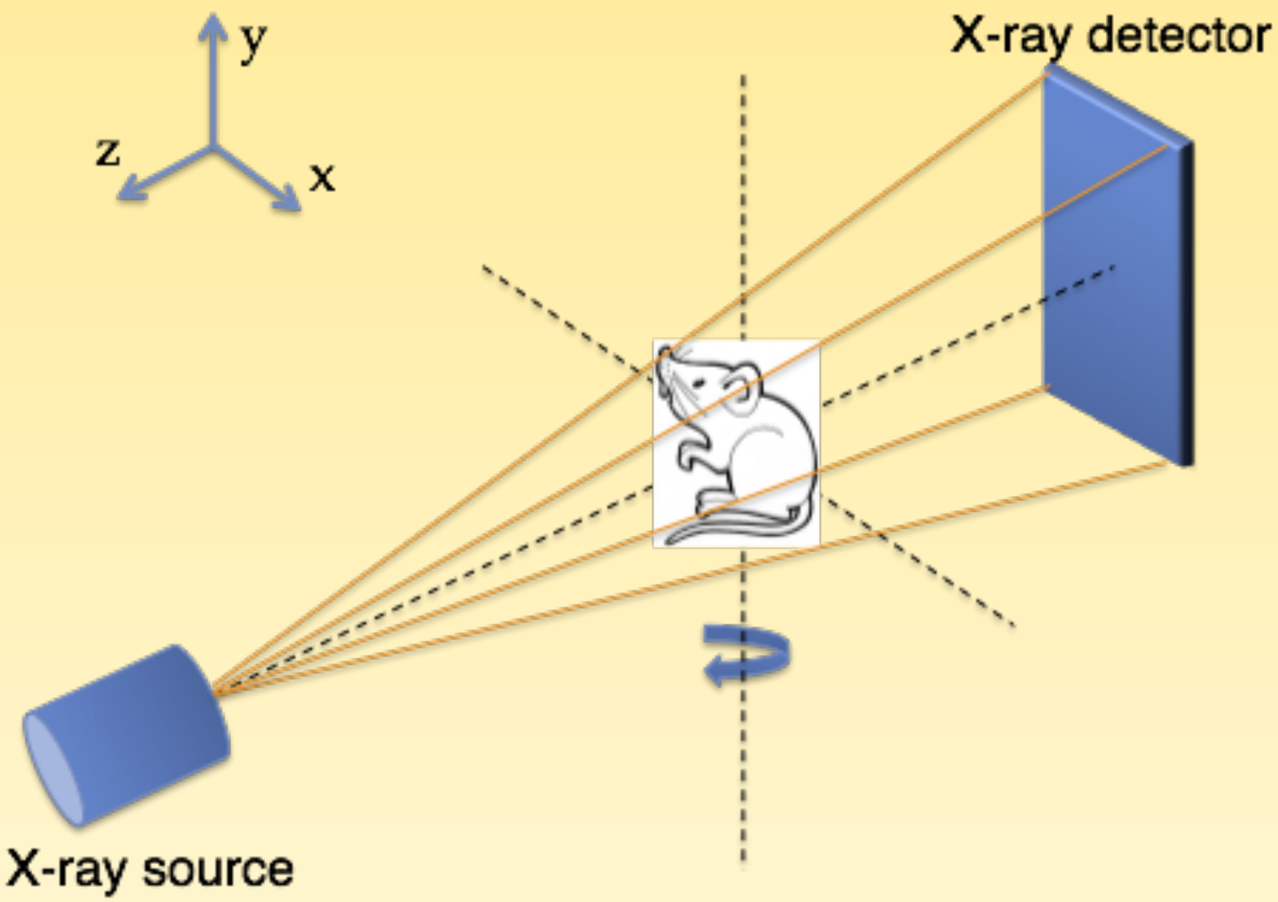
W anode (150 kVp)
Wheel filters



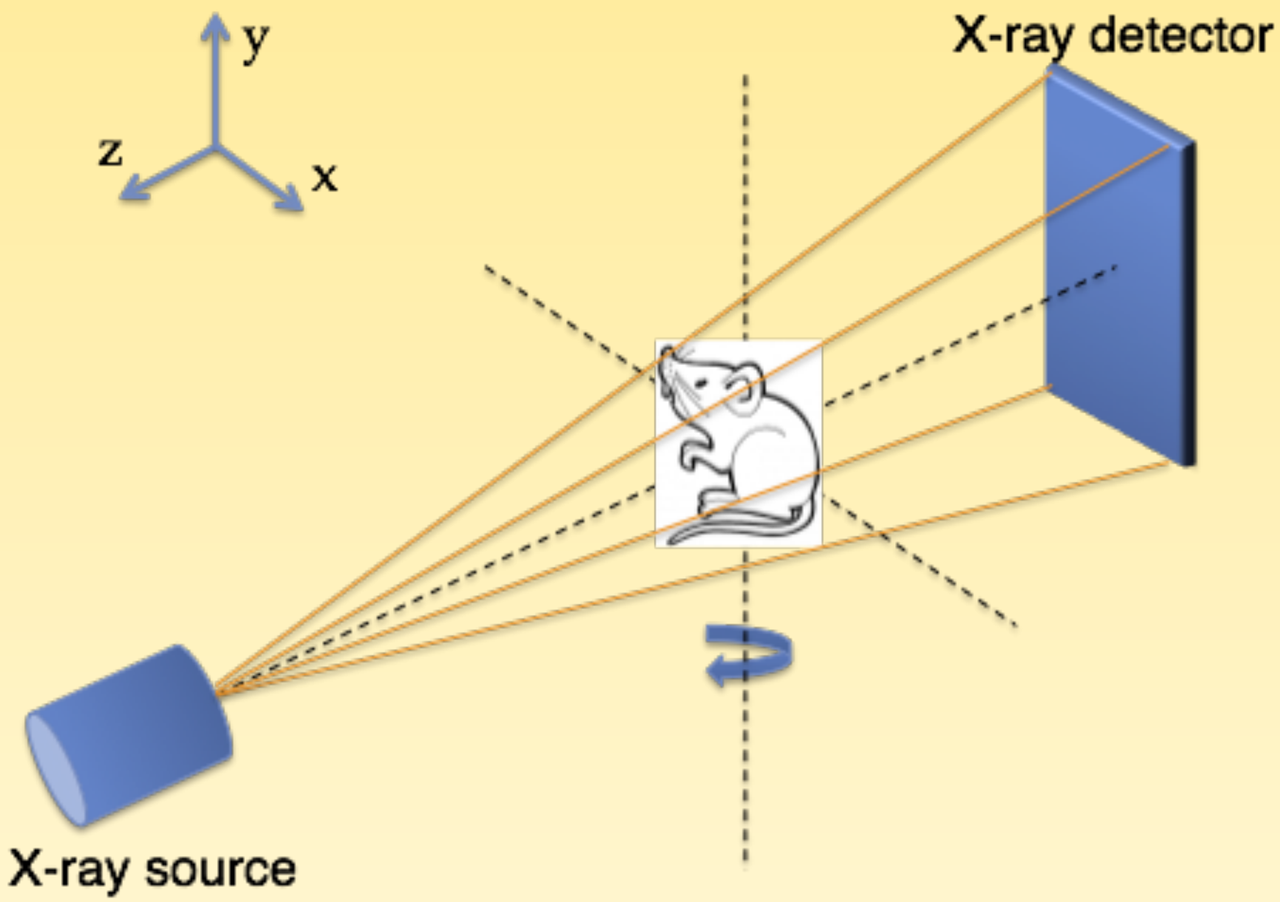
Detector block
Two cameras
Resolution of 85 µm/voxel

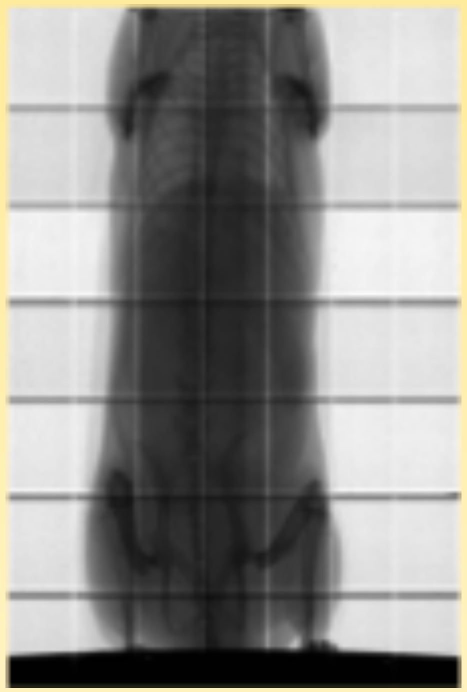
Animal block
Vertical rotation axis
Acquisition mode: shoot and step or continuous rotation
Gas anesthesia: isoflurane

In vivo X-ray imaging at CPPM

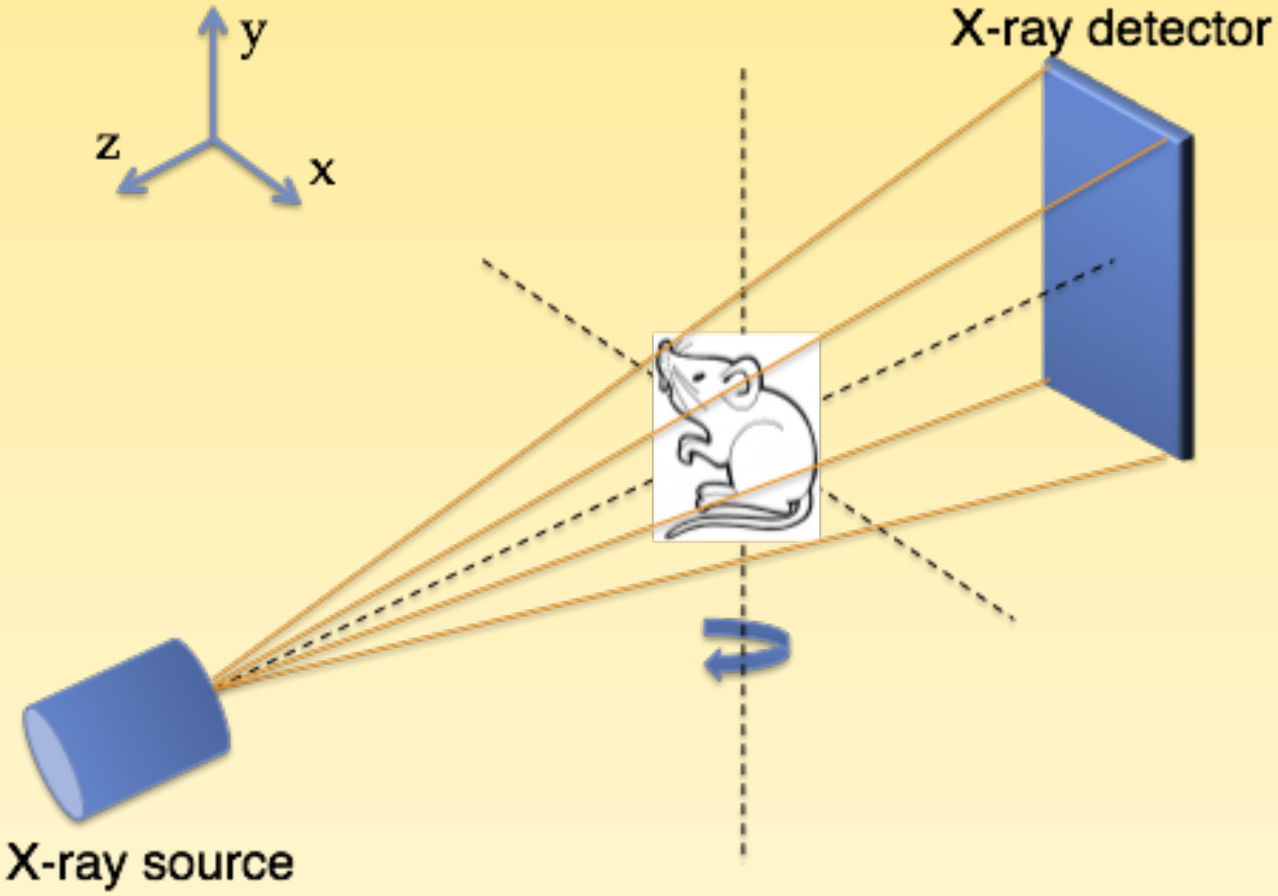


In vivo X-ray imaging at CPPM

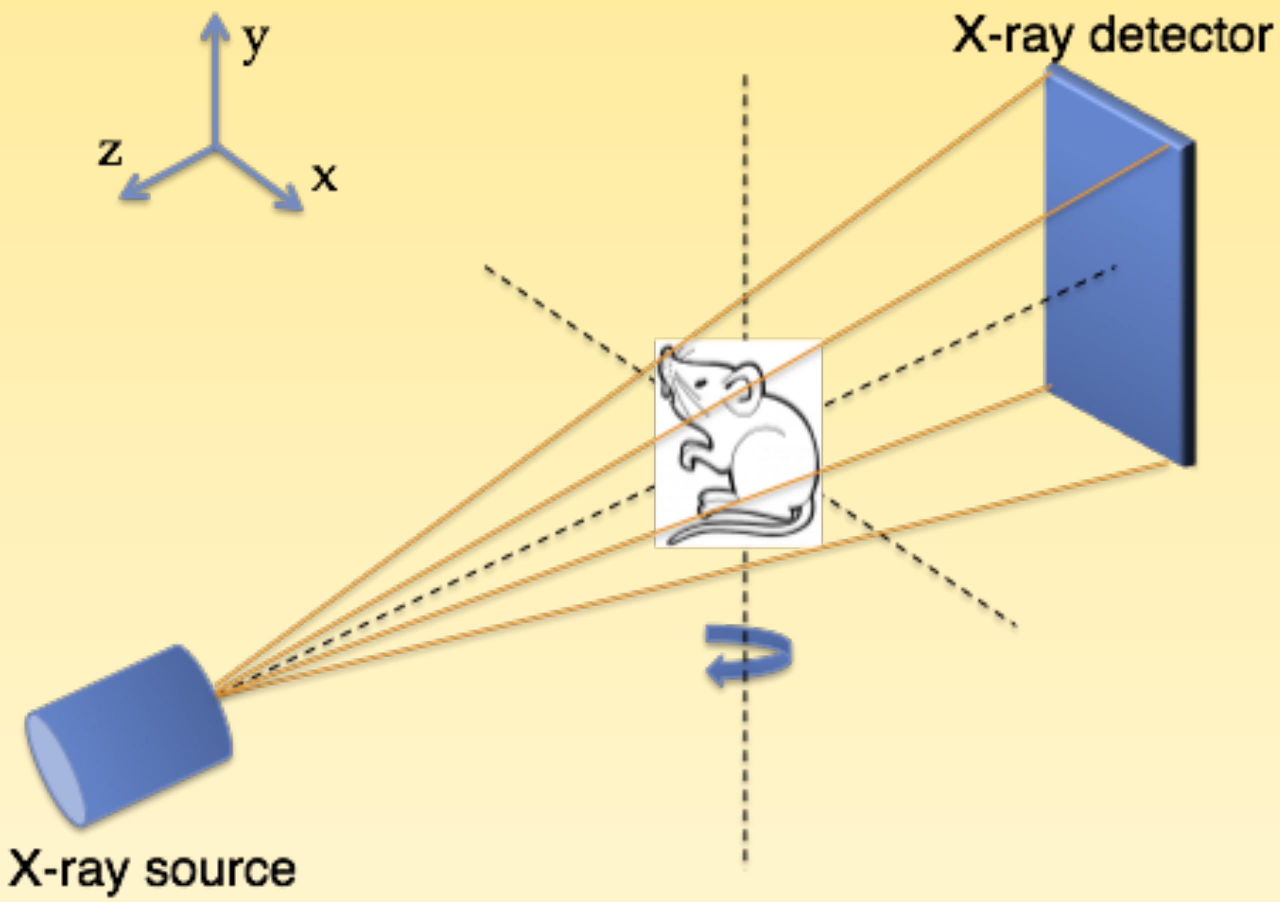


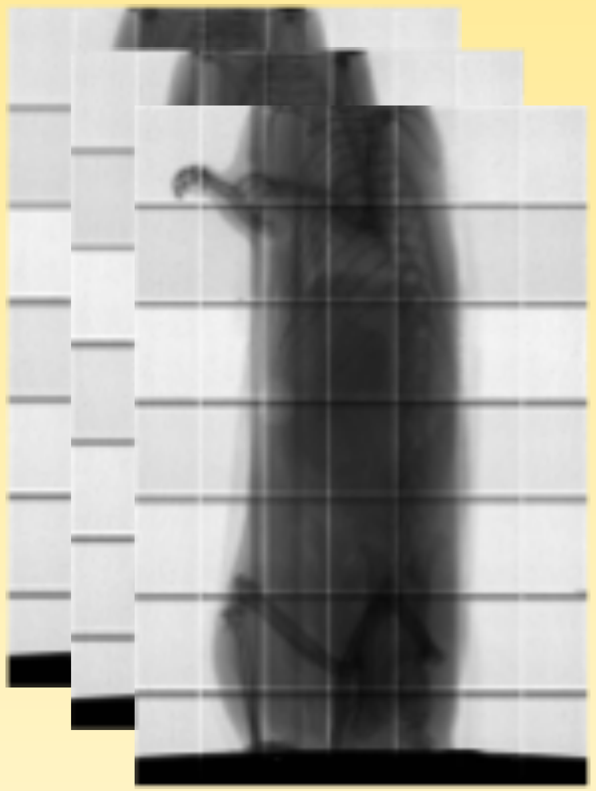


In vivo X-ray imaging at CPPM

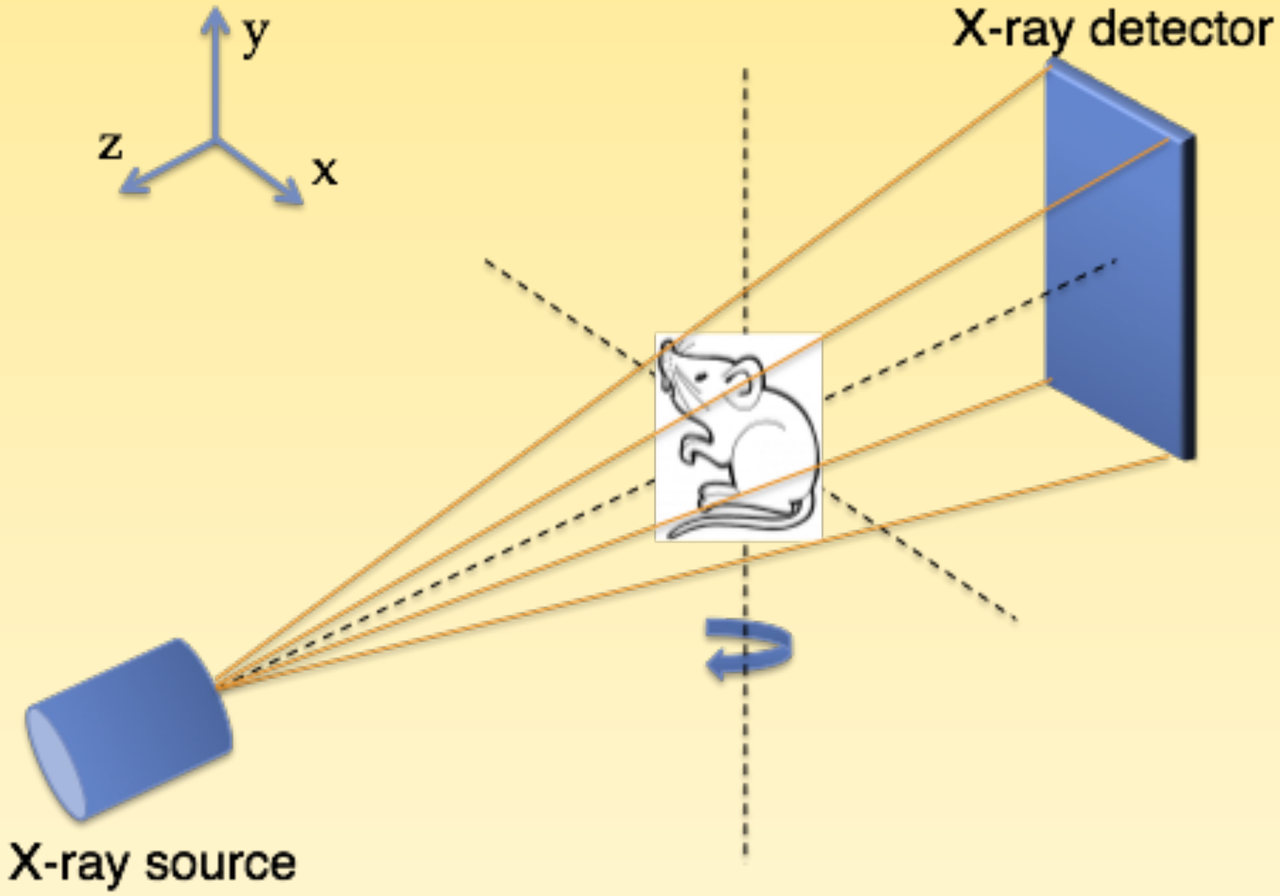


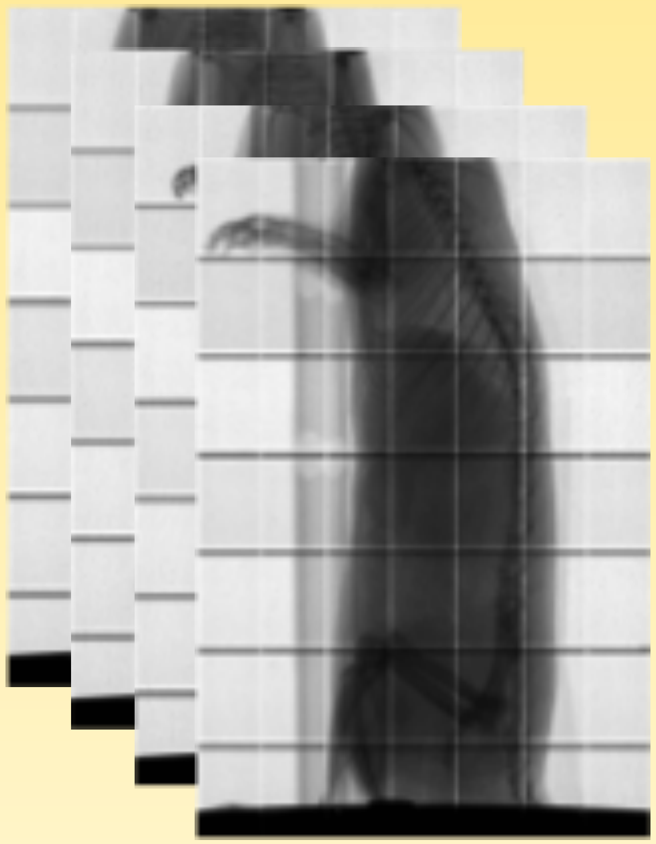
In vivo X-ray imaging at CPPM



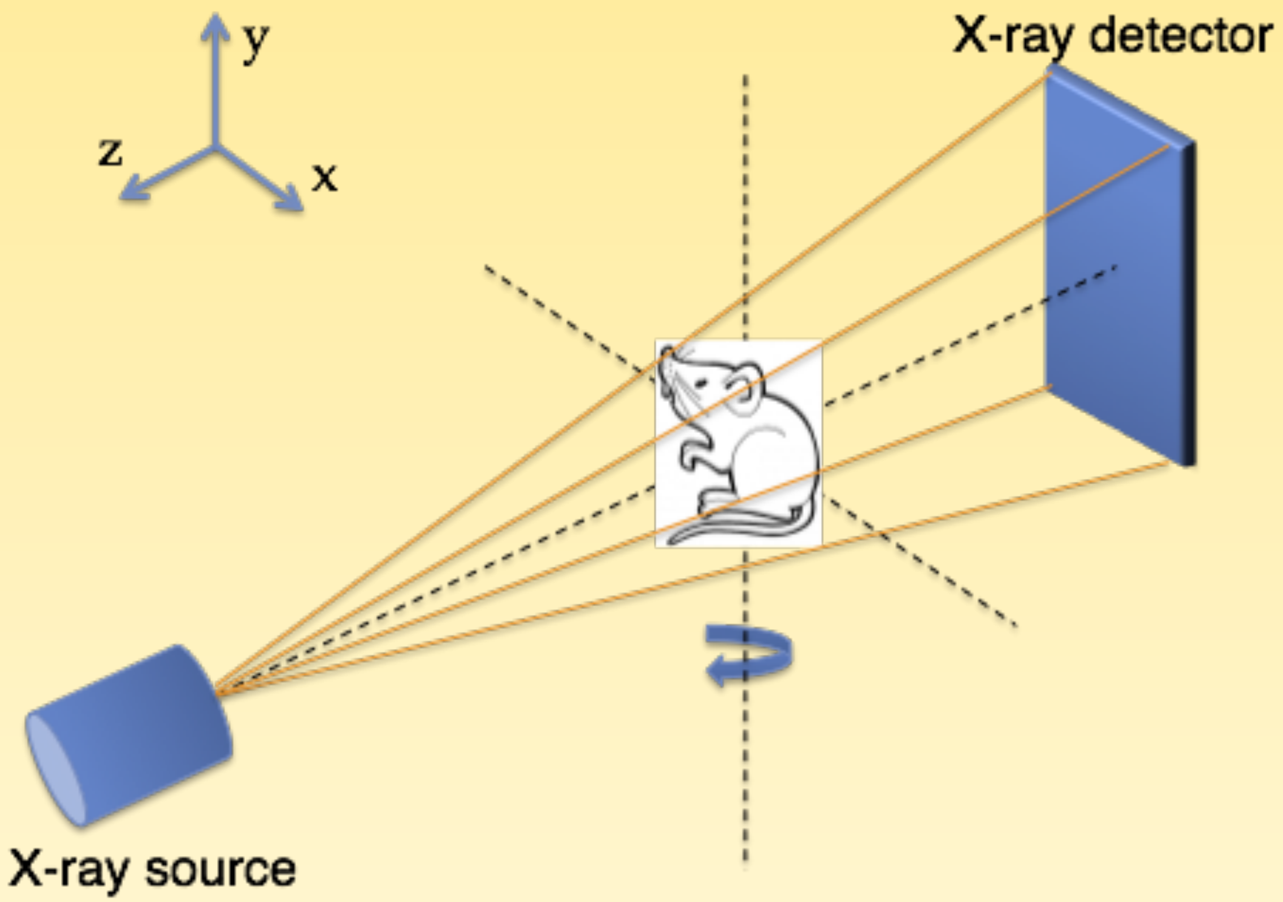


In vivo X-ray imaging at CPPM

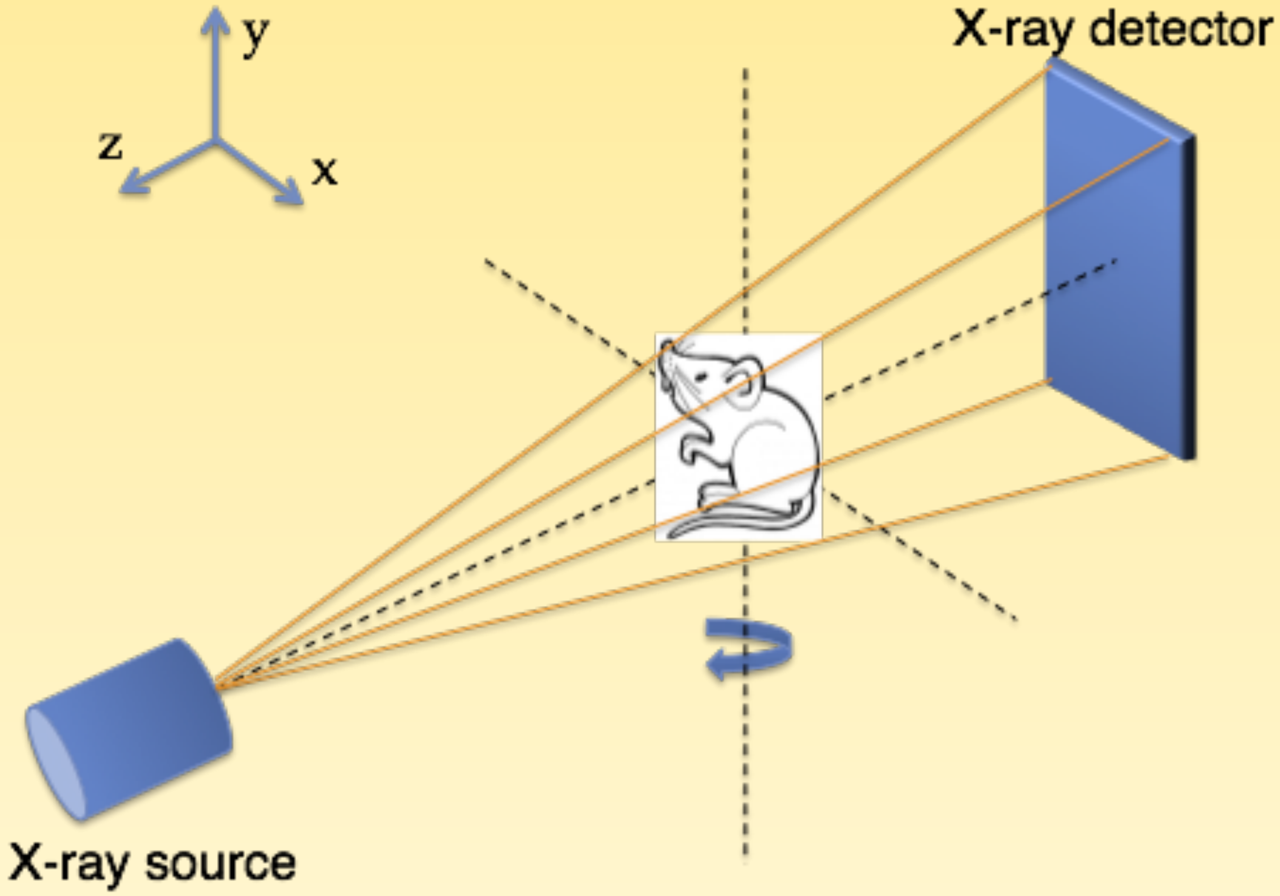




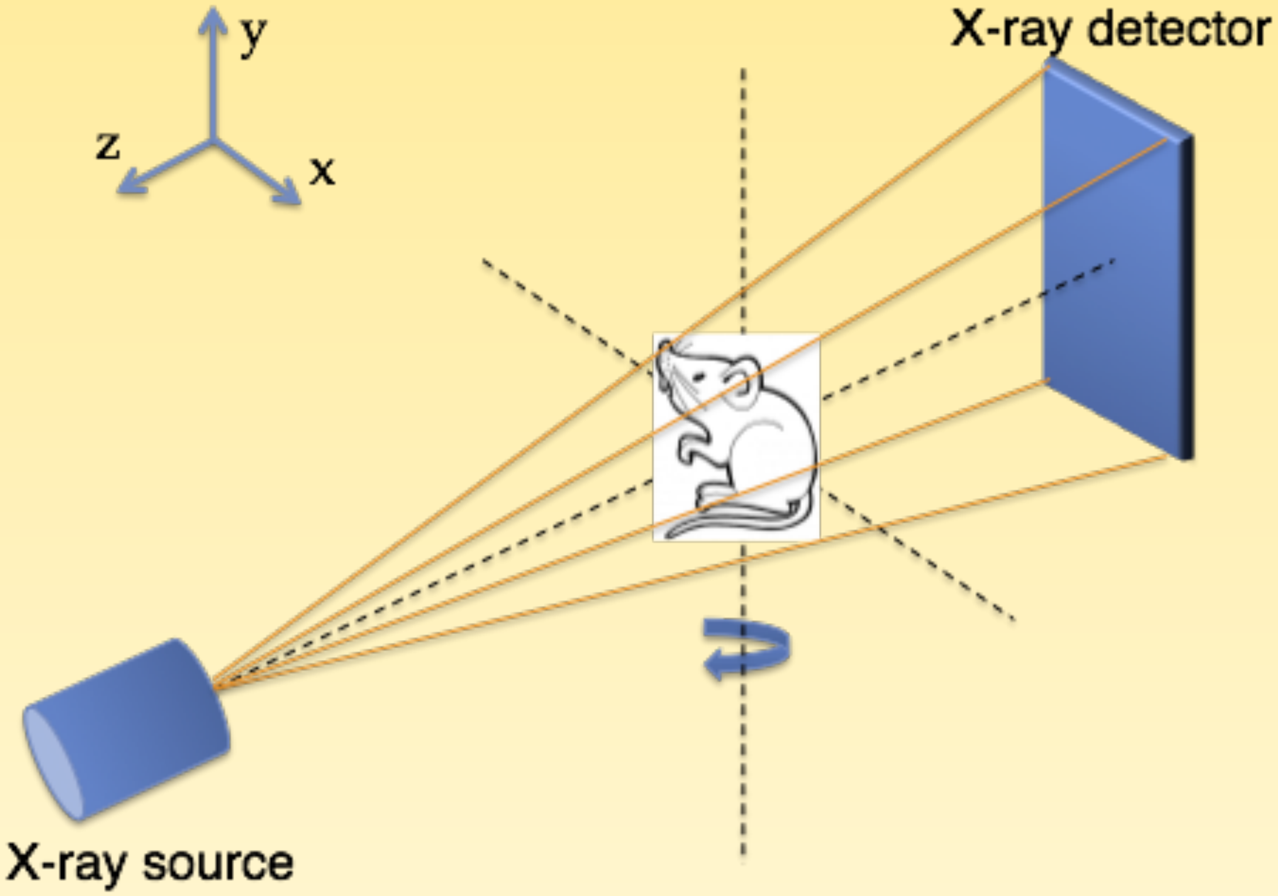
In vivo X-ray imaging at CPPM



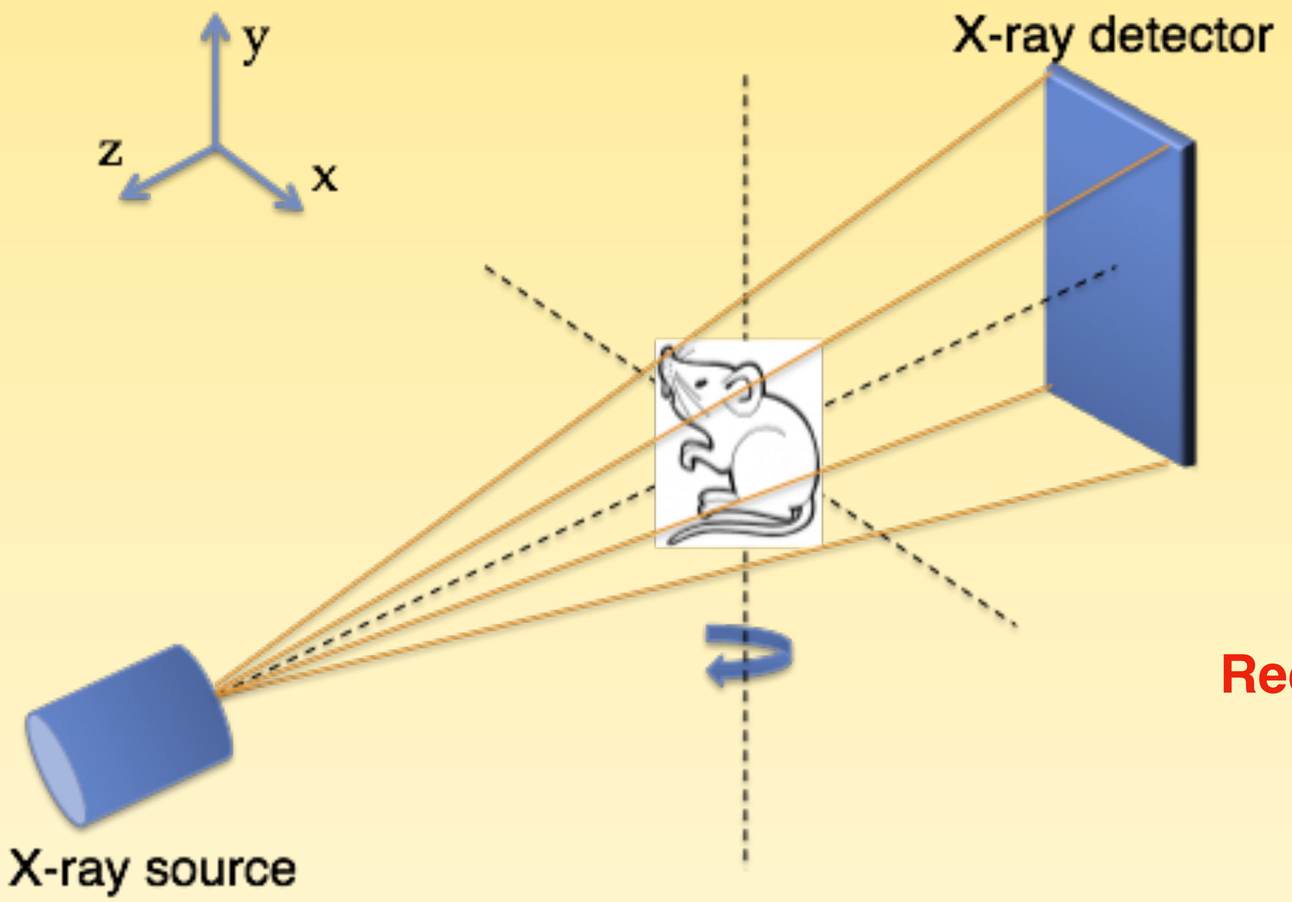
In vivo X-ray imaging at CPPM



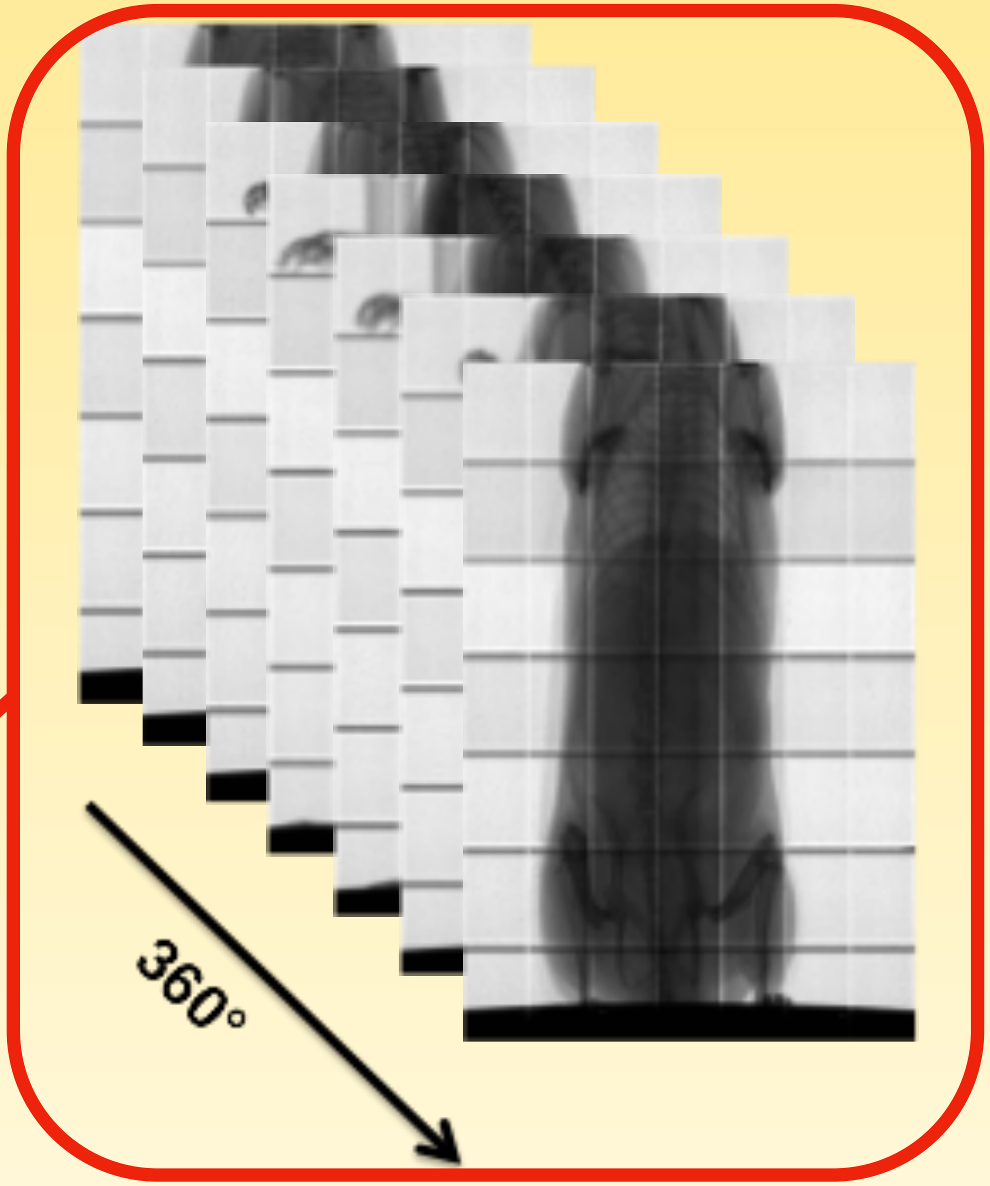
In vivo X-ray imaging at CPPM



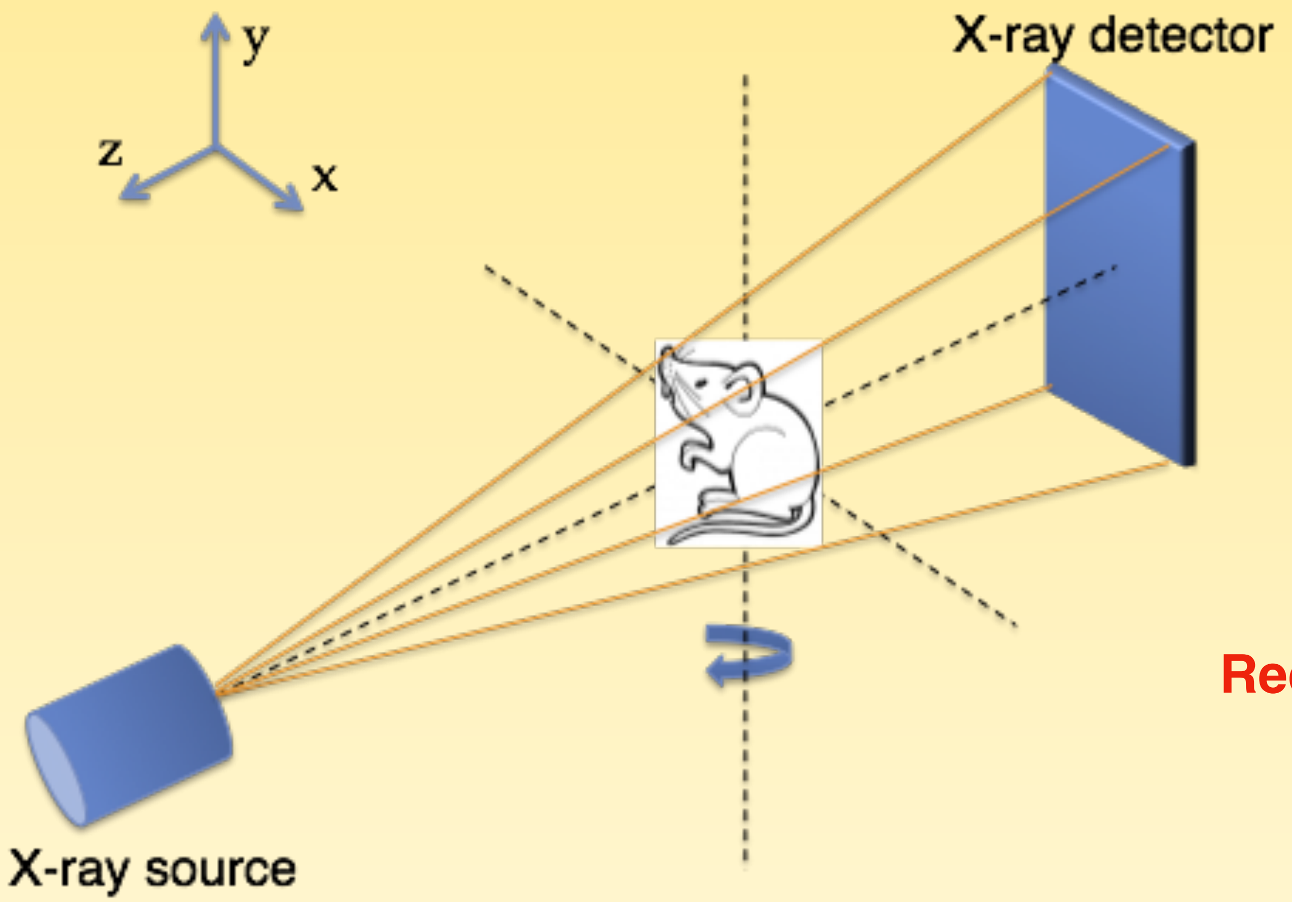
In vivo X-ray imaging at CPPM



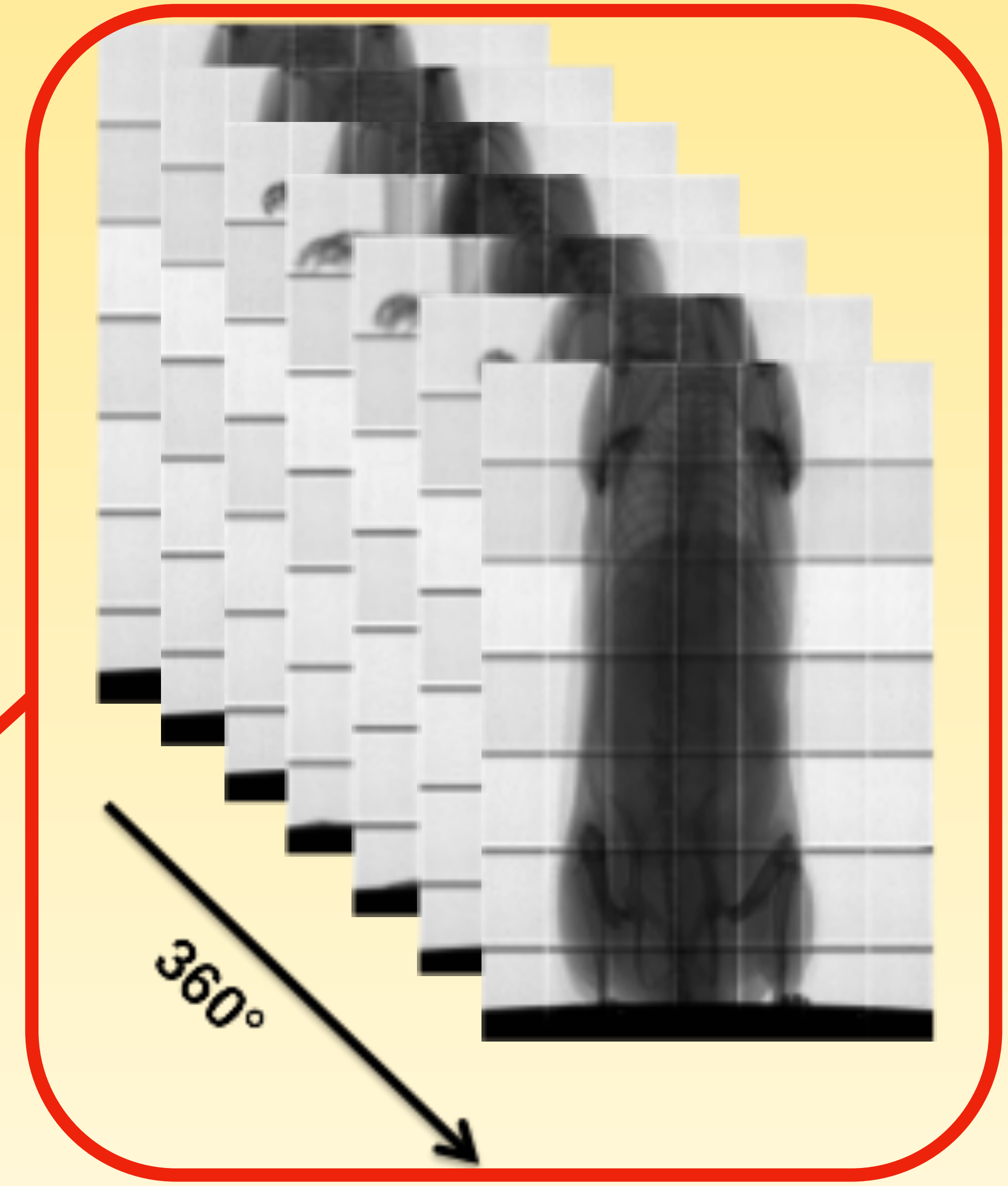
Reconstruction algorithm

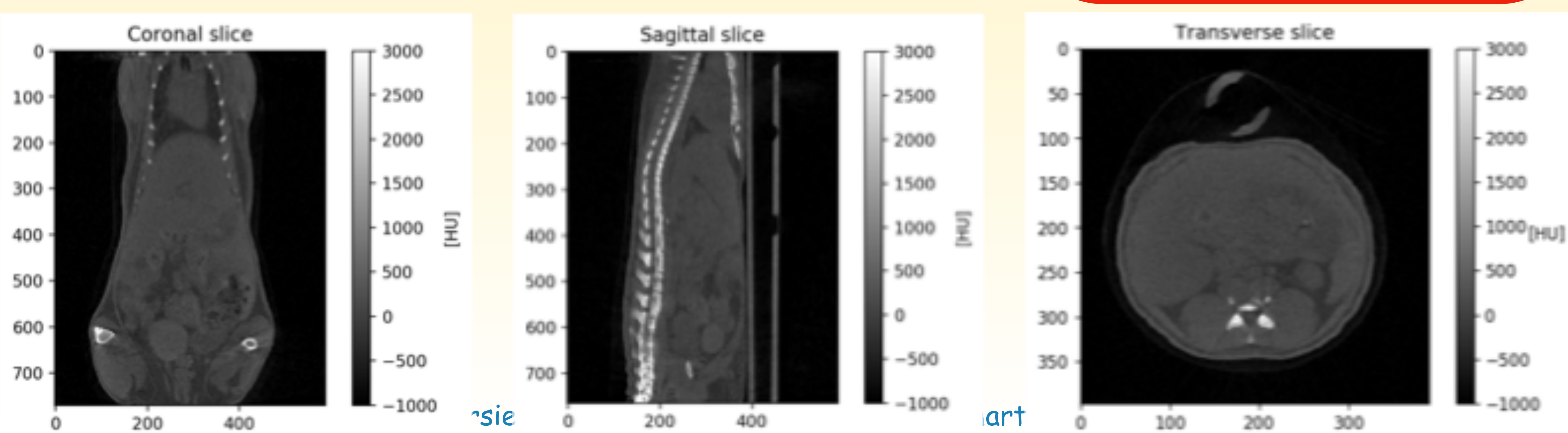


In vivo X-ray imaging at CPPM



Reconstruction algorithm





Outline

1 - Biomedical imaging context : Computerized tomography (CT Scan)

2 - Convolutional Neural Networks (CNNs) for imaging

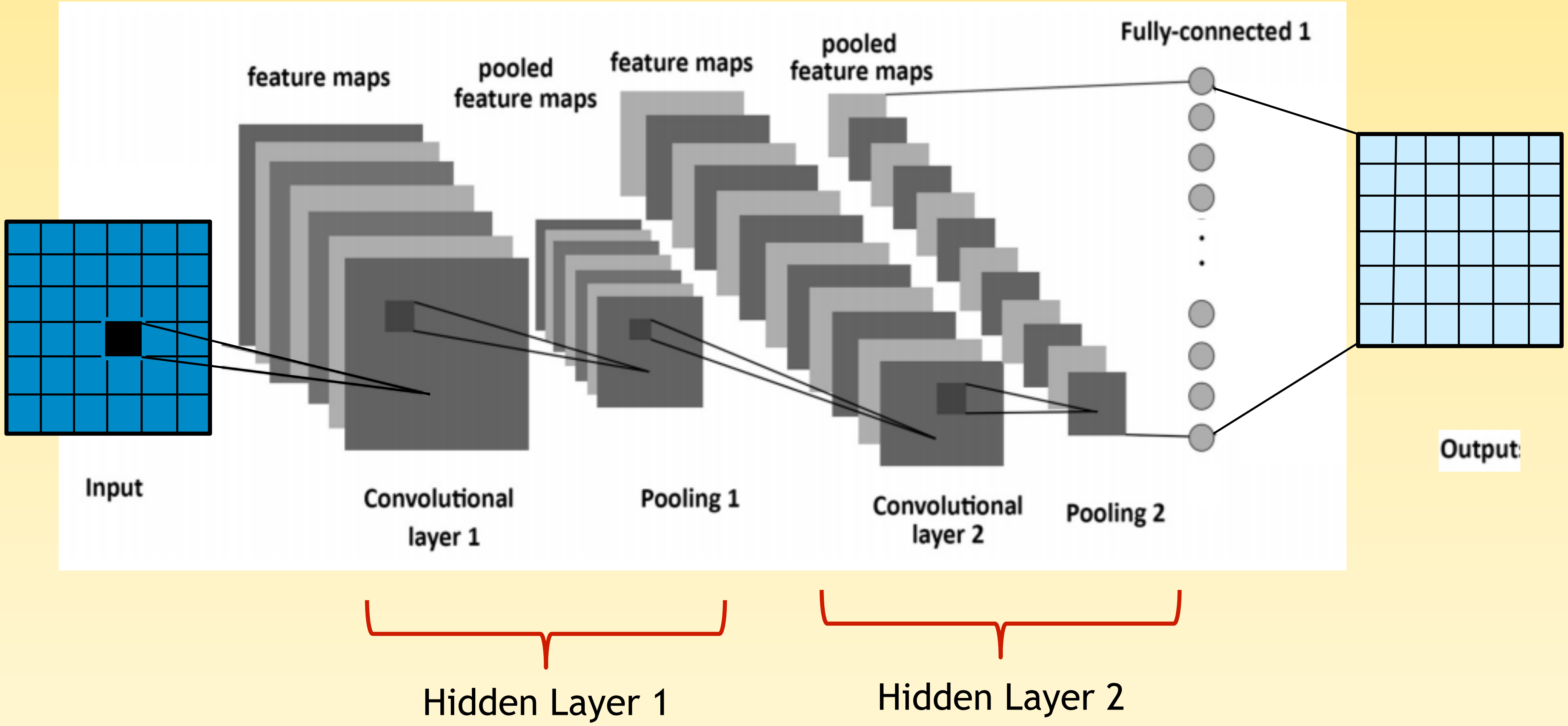
3 - Denoising issues

4 - Segmentation issues

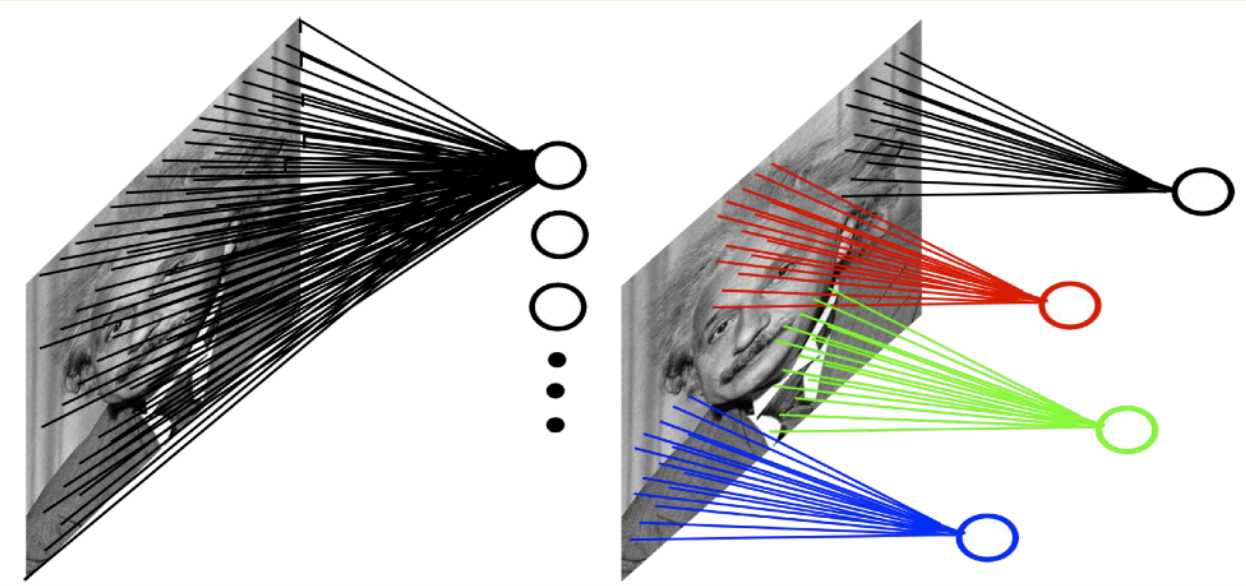
5 - Other issues

6 - Conclusion

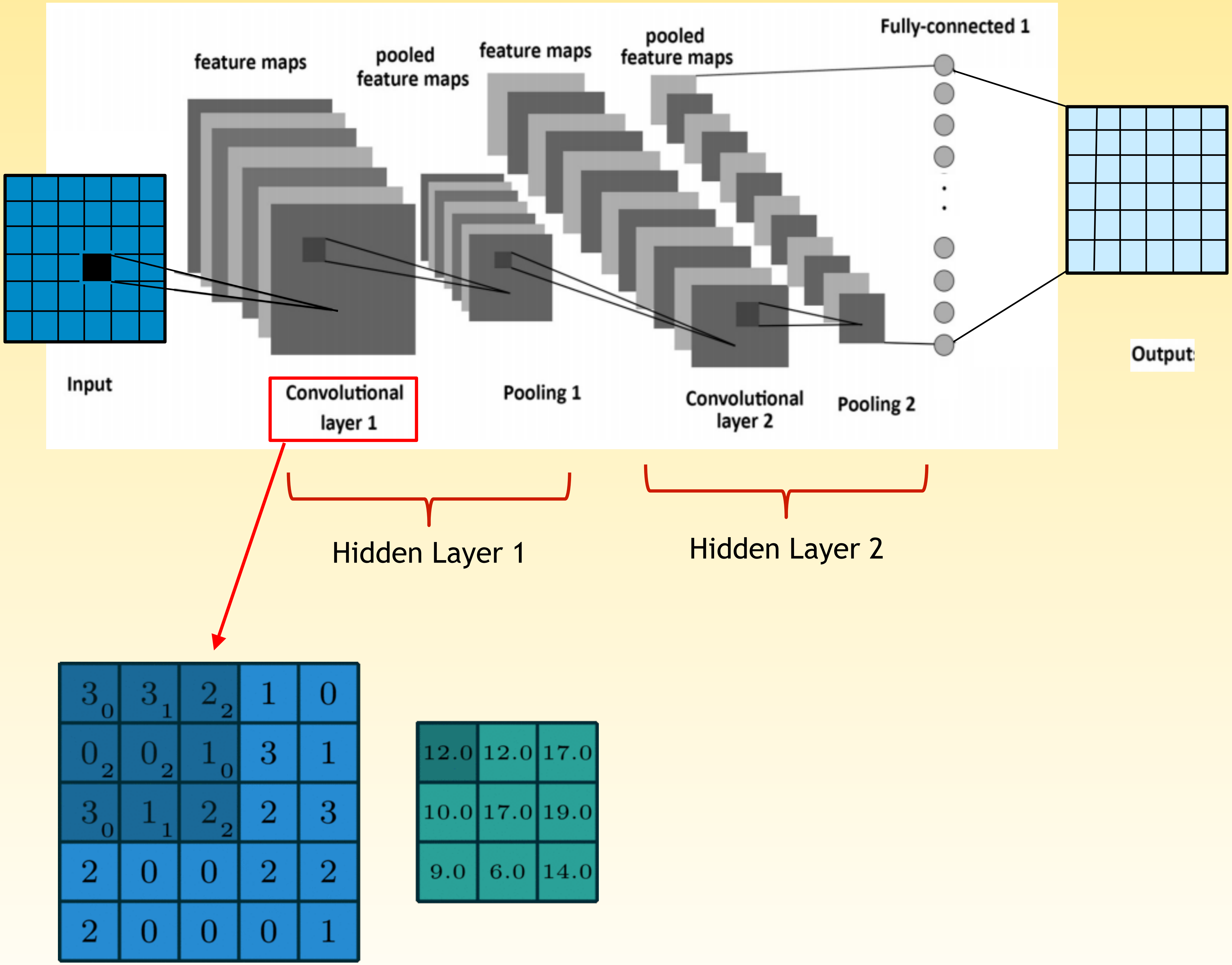
Convolutional Neural Networks (CNNs)



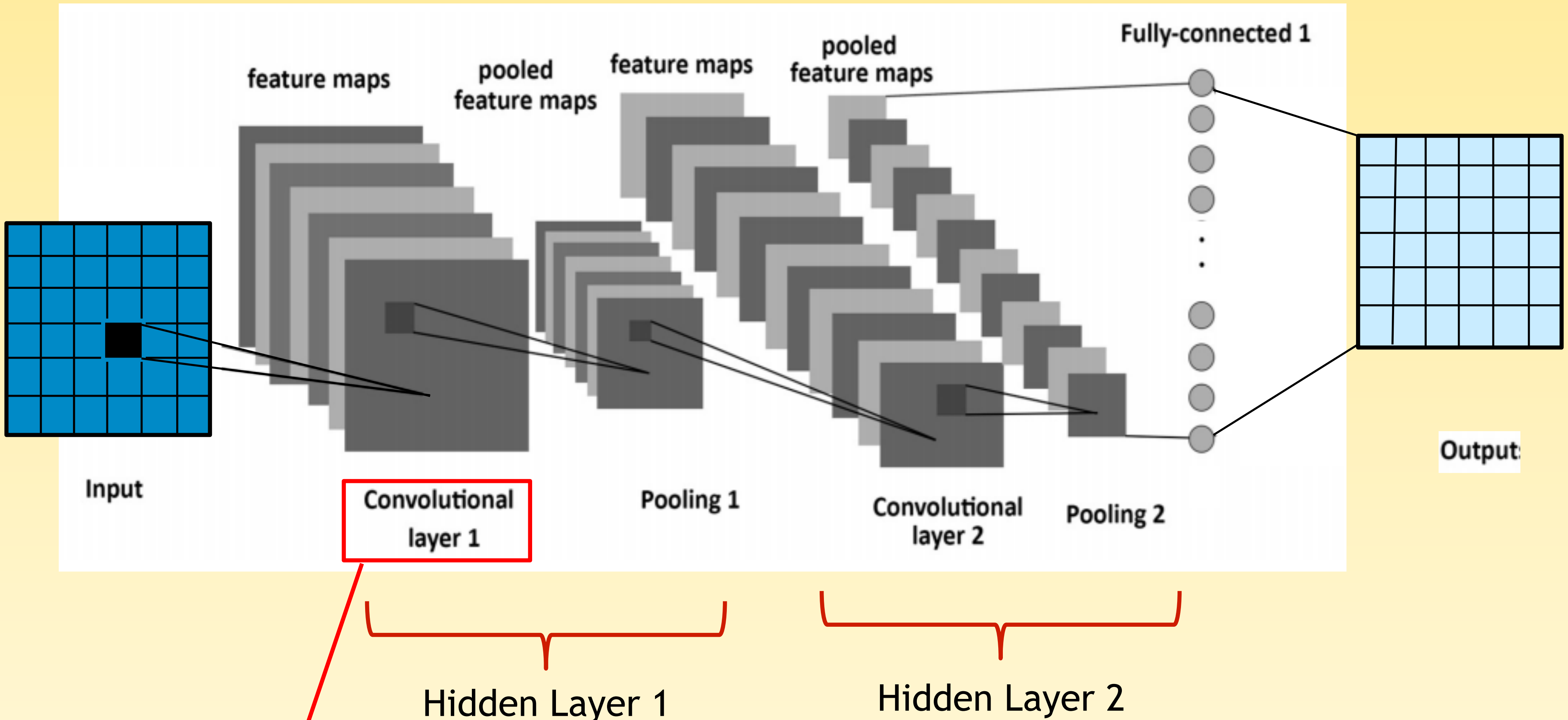
- Motivation: exploit the spatial structure of data
- Layers locally connected



Convolutional Neural Networks (CNNs)



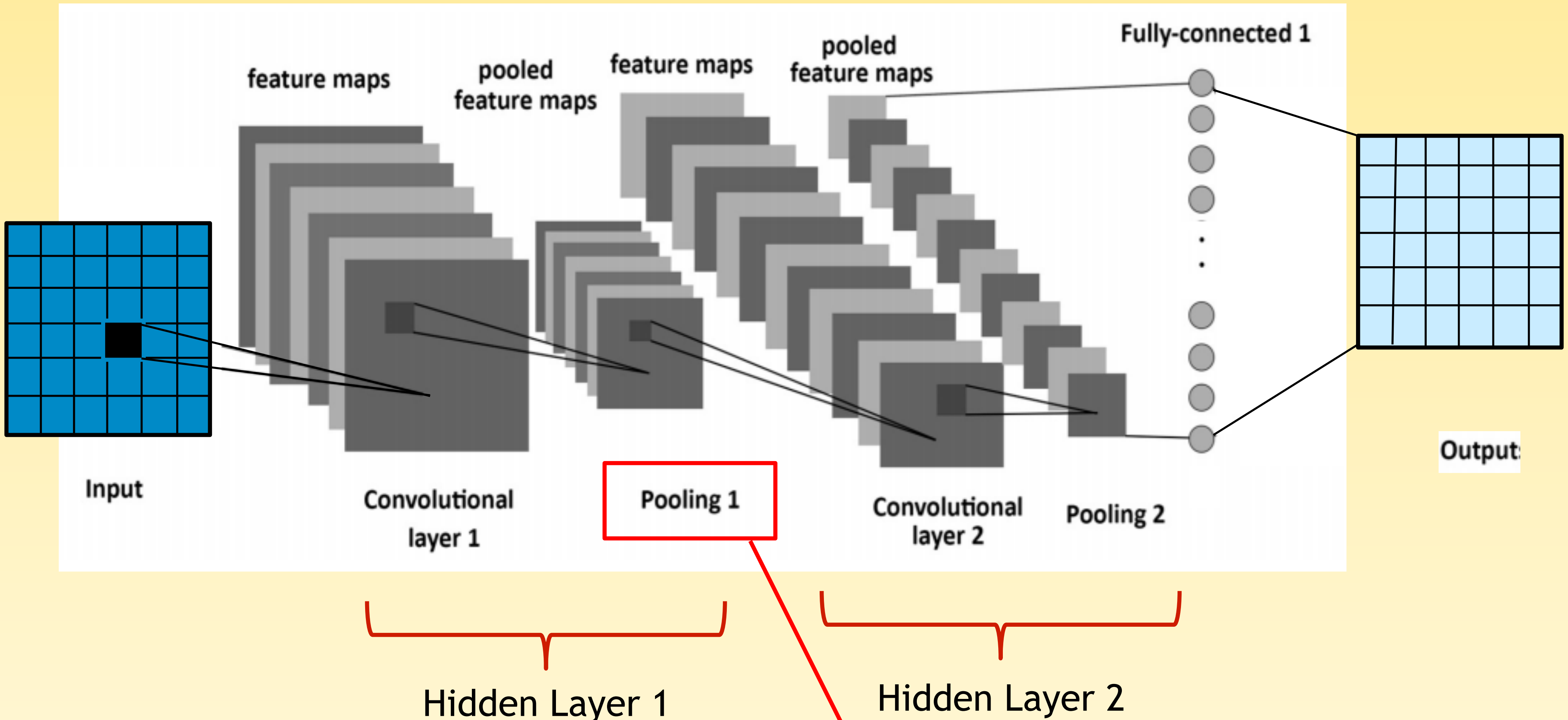
Convolutional Neural Networks (CNNs)

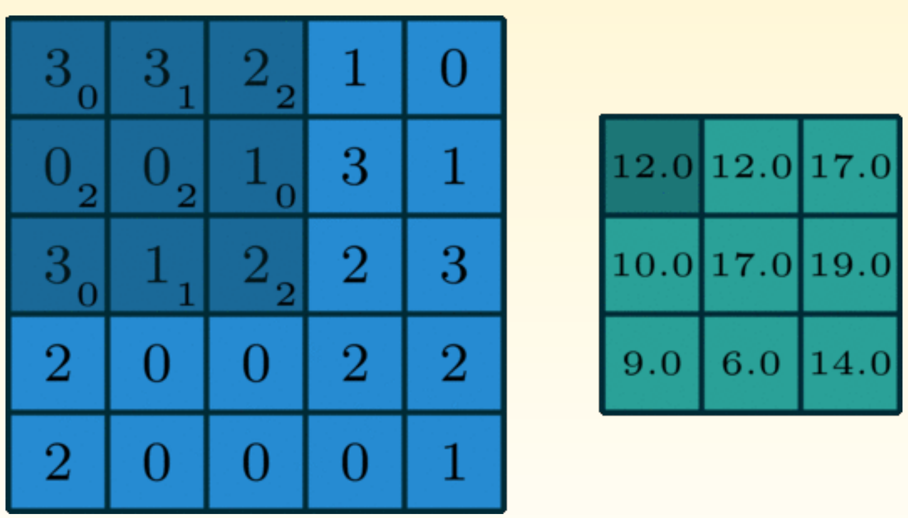


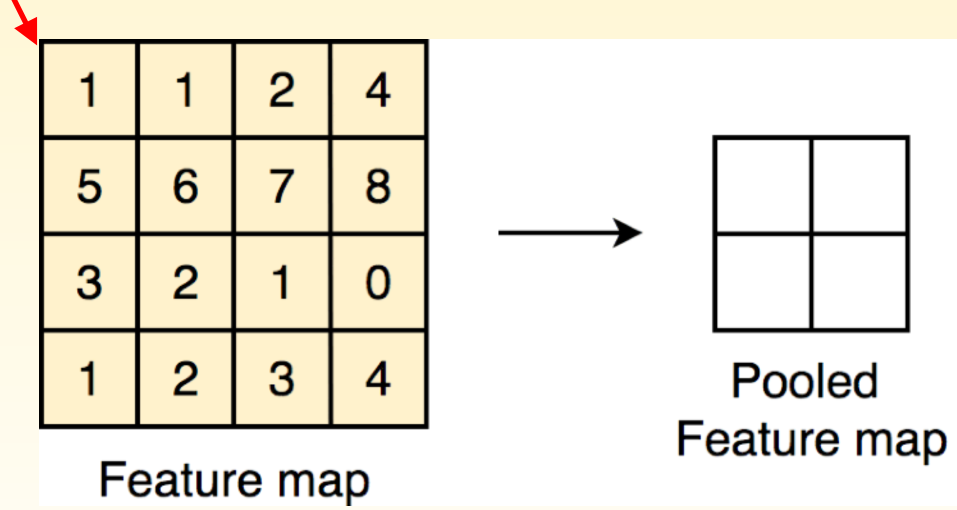
3_0	3_1	2_2	1	0
0_2	0_2	1_0	3	1
3_0	1_1	2_2	2	3
2	0	0	2	2
2	0	0	0	1

12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0

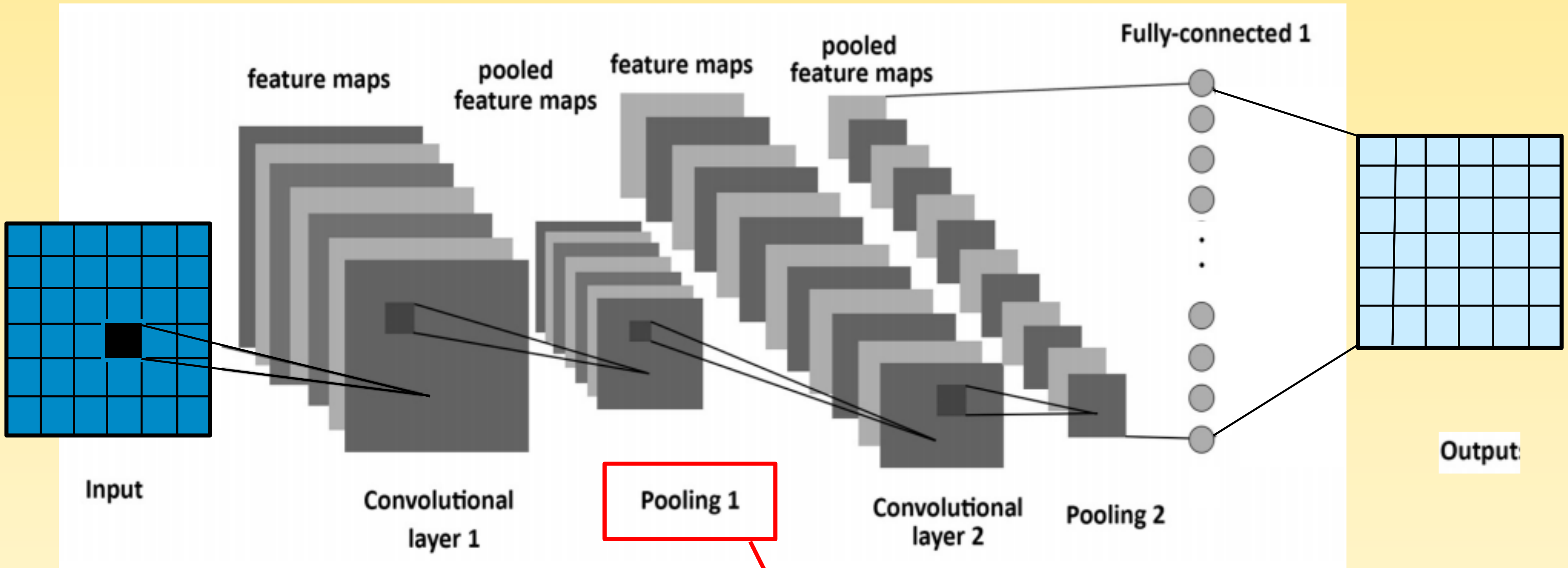
Convolutional Neural Networks (CNNs)





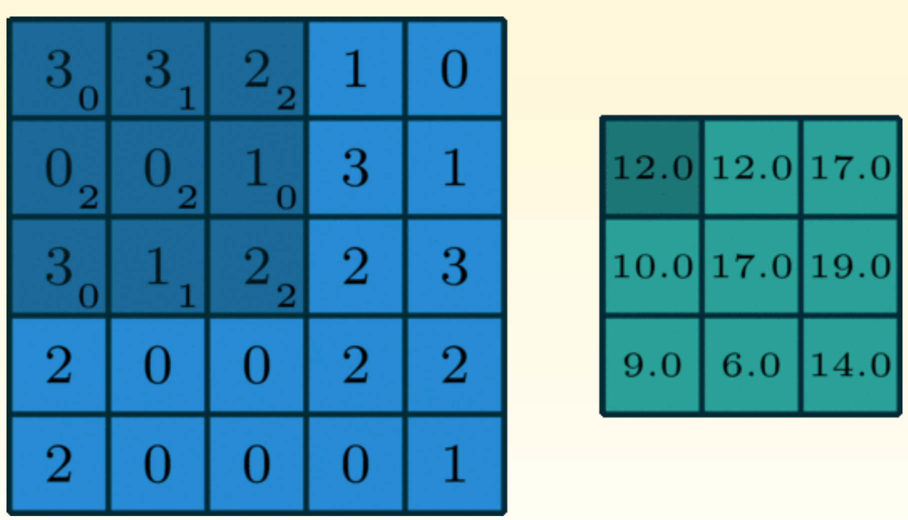


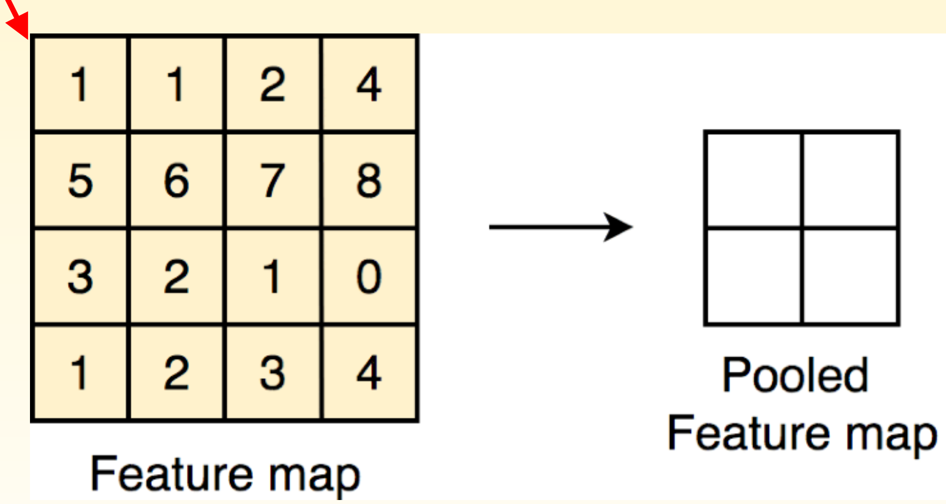
Convolutional Neural Networks (CNNs)



Hidden Layer 1

Hidden Layer 2





Outline

1 - Biomedical imaging context : Computerized tomography (CT Scan)

2 - Convolutional Neural Networks (CNNs) for imaging

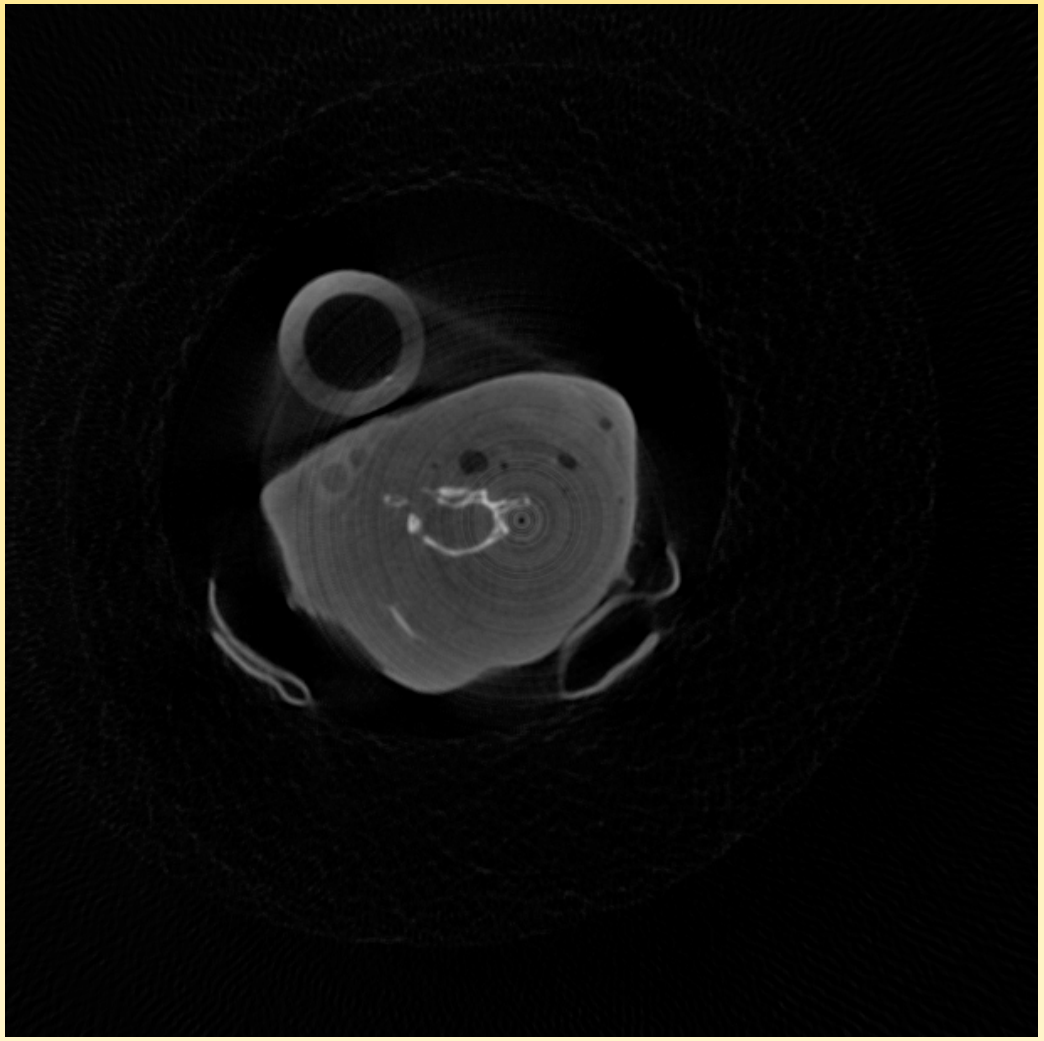
3 - Denoising issues

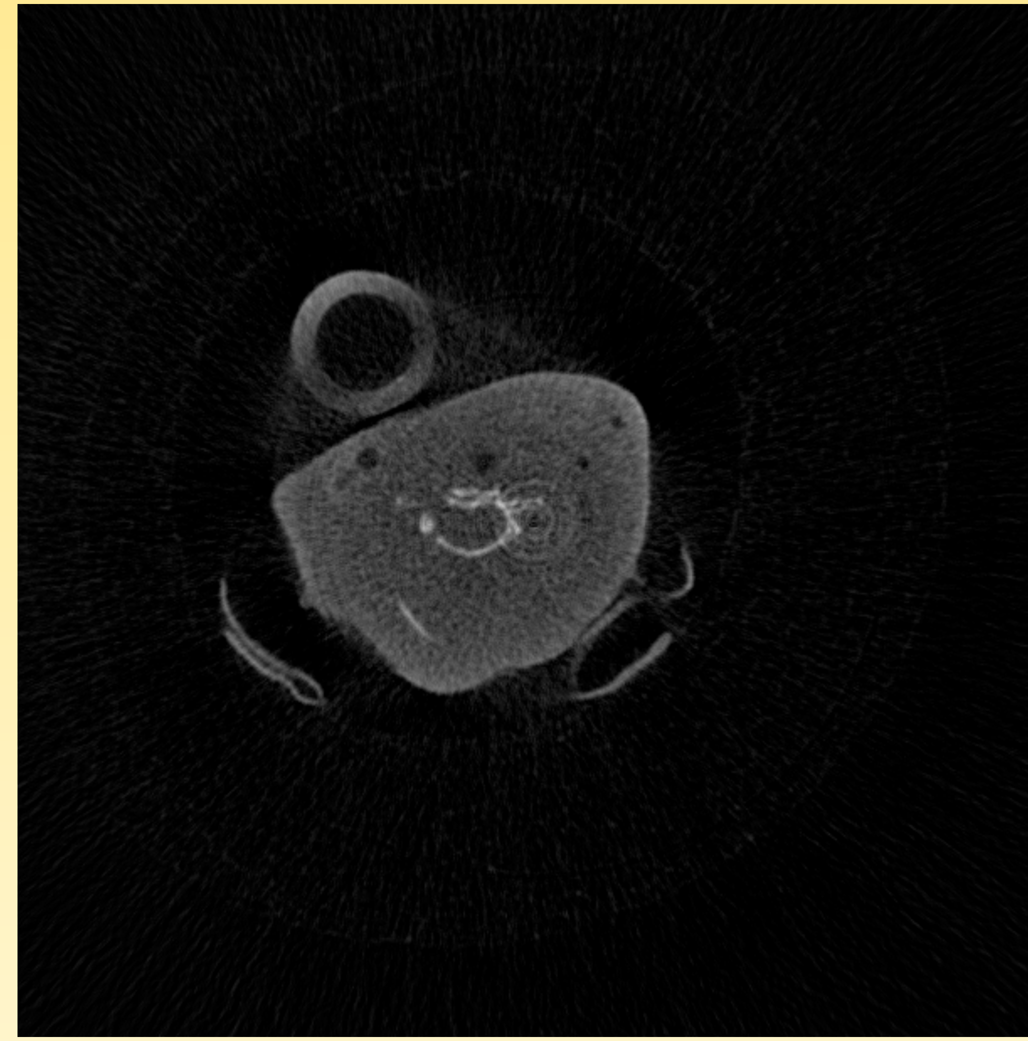
4 - Segmentation issues

5 - Other issues

6 - Conclusion

Noise in CT

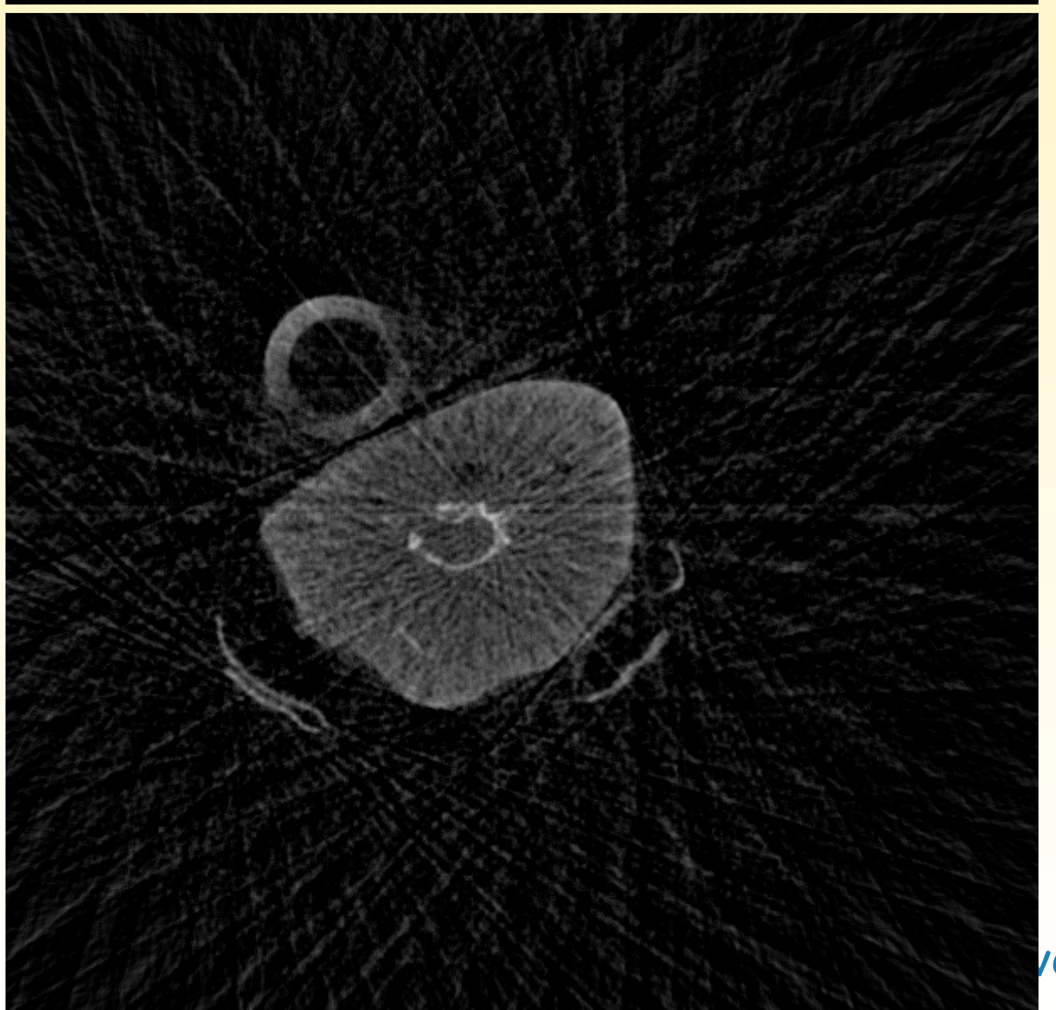




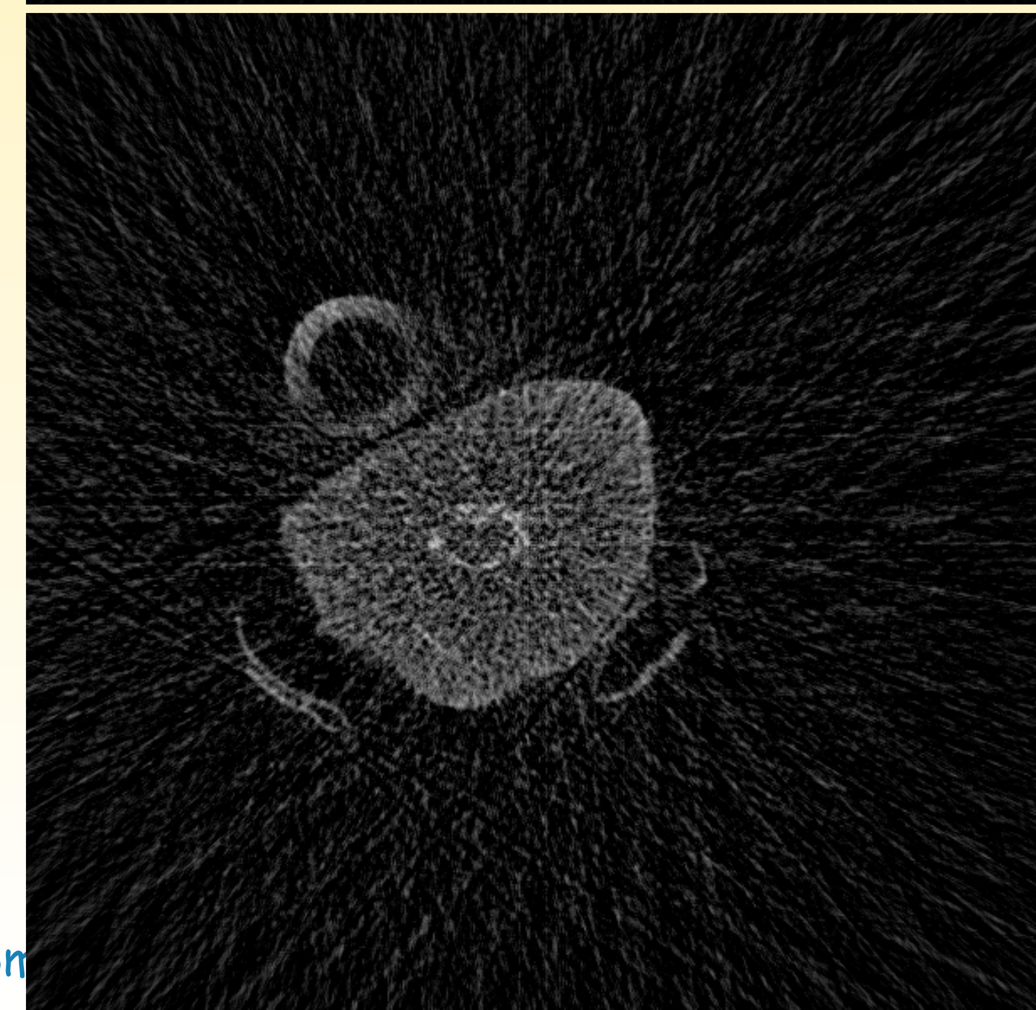
Transverse slice

FBP algo.
588 x 588 pix.

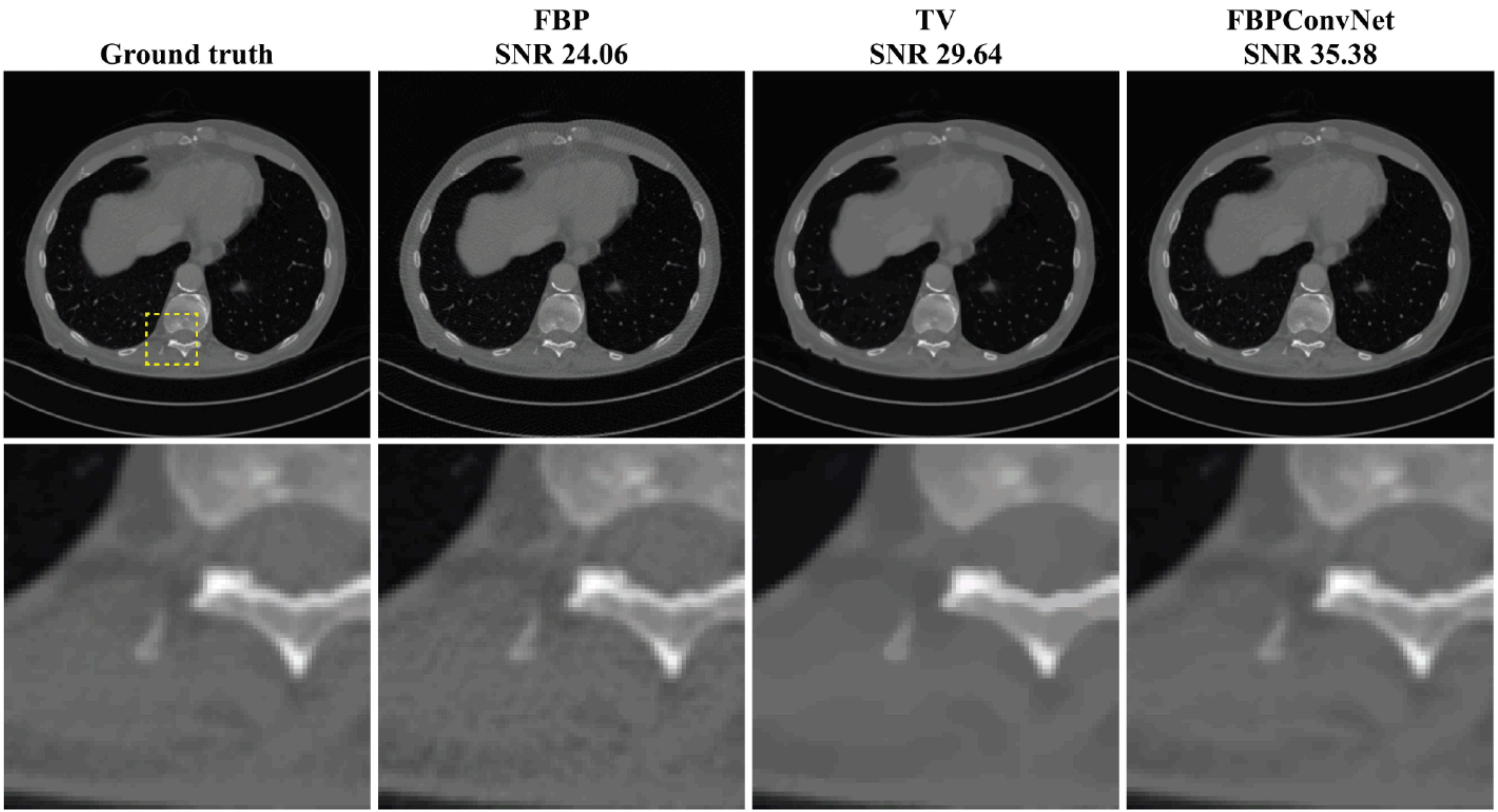
Angles/10
Radon noise



Dose/25
Gaussian noise



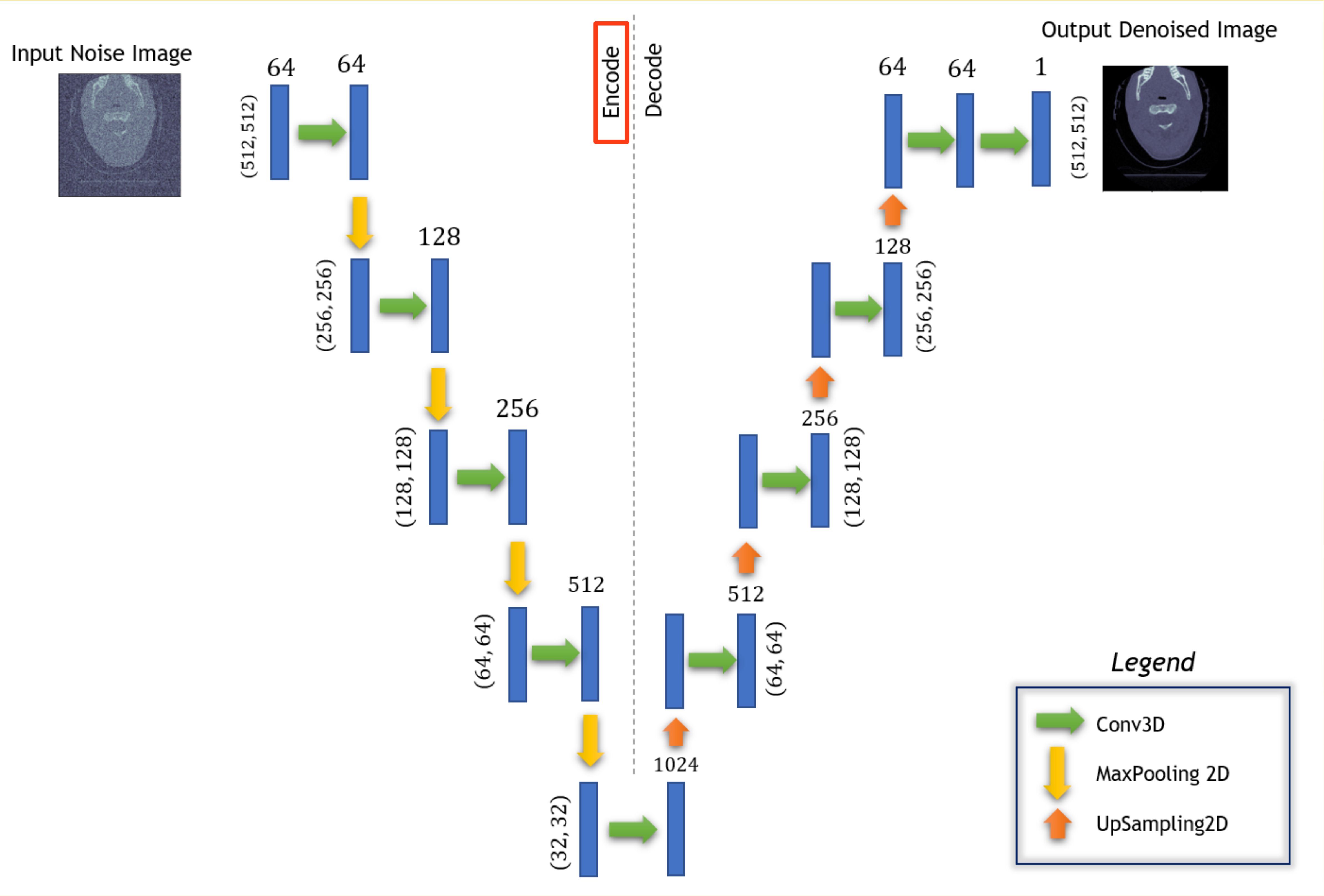
Why Deep Learning in CT Reconstruction ?



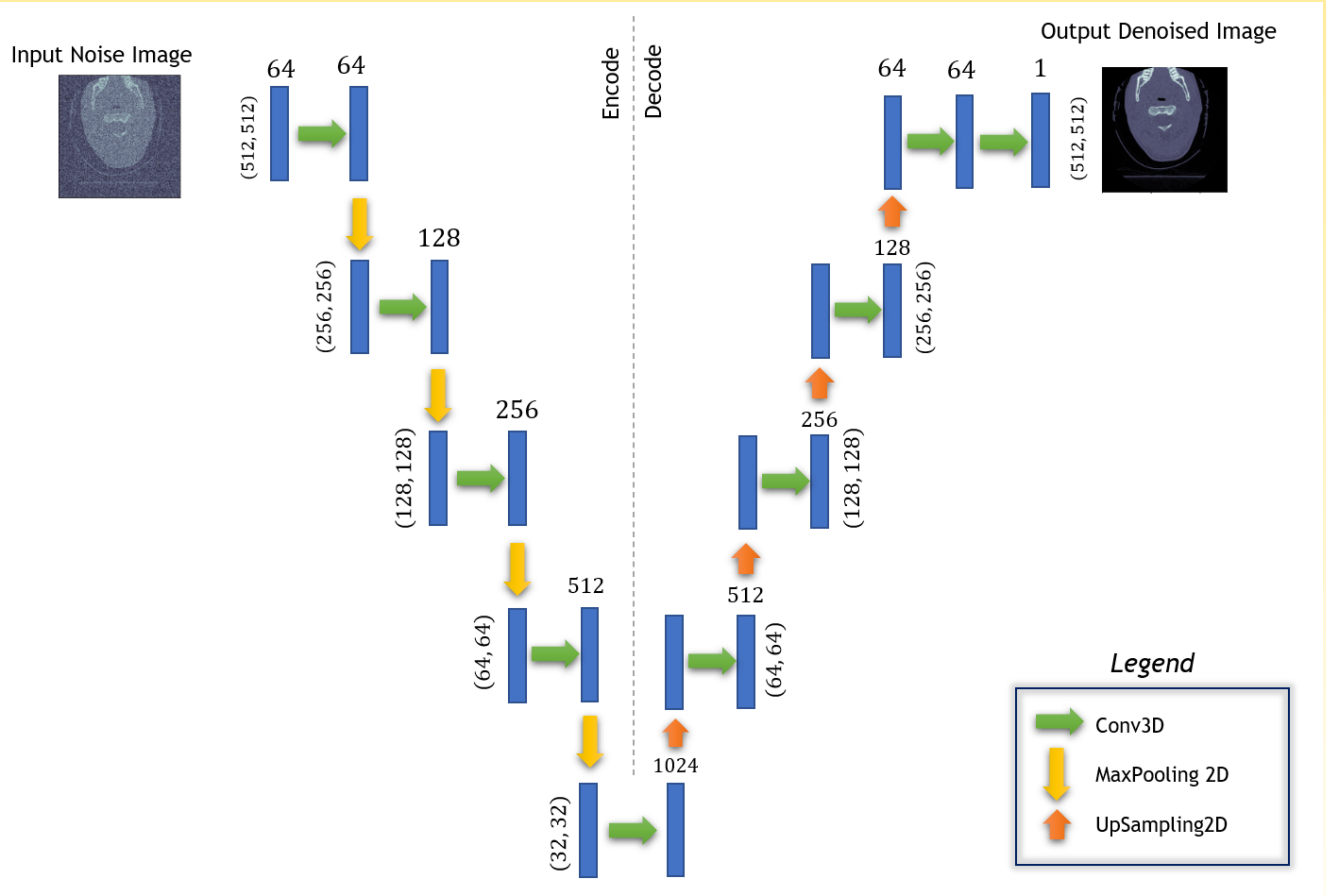
Reconstructed images of biomedical dataset from 143 views using FBP, TV regularized convex optimization [14], and the FBPConvNet.

[Jin et al., June 2017]

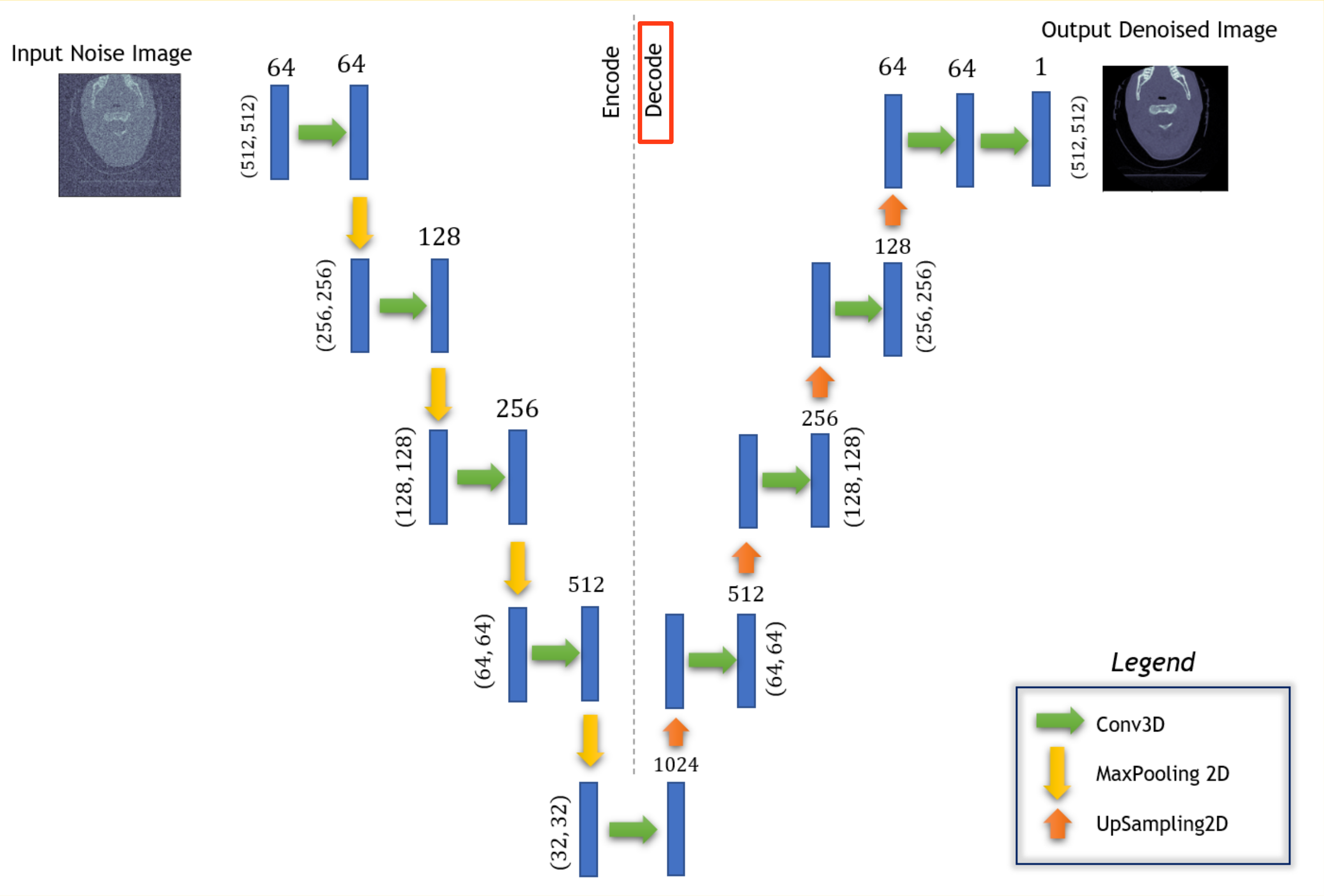
CNNs Autoencoder



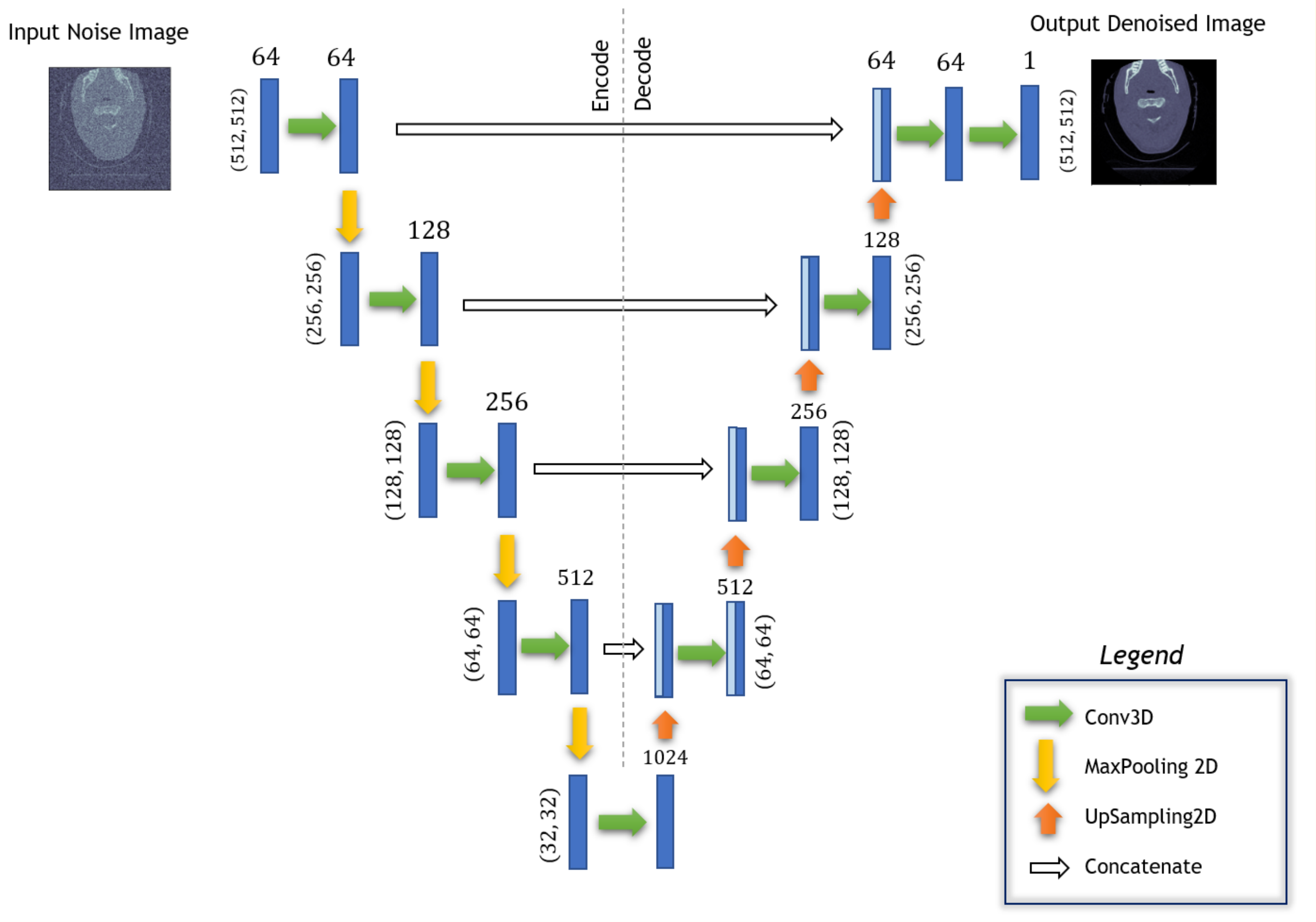
CNNs Autoencoder



CNNs Autoencoder



CNNs : The U-net network for denoising



CNNs : Implementation in Keras/Tensorflow

```python
from tensorflow.keras.activations import relu
from tensorflow.keras.layers import Input, Activation, Convolution2D, BatchNormalization, MaxPooling2D, \
    Conv2DTranspose, Concatenate, AveragePooling2D, Dense, Conv2D, SpatialDropout2D, Dropout, Flatten, concatenate, Reshape, UpSampling2D
from tensorflow.keras.models import Model
from tensorflow.keras import backend as K


def define_model (n,m):

    input_img = Input(shape=(n, m, 1))

    conv0 = Convolution2D(32, 3, padding='same', input_shape=[512,512,1], activation='tanh')(input_img) #(None, 512,512,32)
    conv0 = Convolution2D(32, 3, padding='same', activation='tanh')(conv0) #(None, 512,512,32)
    M0 = MaxPooling2D()(conv0) #(None, 256,256,32)


    conv1 = Conv2D(64, 3, activation = 'tanh', padding = 'same', kernel_initializer = 'he_normal')(M0)
    conv1 = Conv2D(64, 3, activation = 'tanh', padding = 'same', kernel_initializer = 'he_normal')(conv1)
    SD1 = SpatialDropout2D(30/64)(conv1)
    P1 = MaxPooling2D(pool_size=(2, 2))(SD1)#(None, 128,128,64)

    conv2 = Conv2D(128, 3, activation = 'tanh', padding = 'same', kernel_initializer = 'he_normal')(P1)
    conv2 = Conv2D(128, 3, activation = 'tanh', padding = 'same', kernel_initializer = 'he_normal')(conv2)
    P2 = MaxPooling2D(pool_size=(2, 2))(conv2)#(None, 64,64,128)
```

CNNs : Implementation in Keras/Tensorflow

```
conv4 = Conv2D(512, 3, activation = 'tanh', padding = 'same', kernel_initializer = 'he_normal')(P3)
conv4 = Conv2D(512, 3, activation = 'tanh', padding = 'same', kernel_initializer = 'he_normal')(conv4)#(64,64,512)
D4 = Dropout(0.5)(conv4)
P4 = MaxPooling2D(pool_size=(2, 2))(D4)#(32,32,512)#(None, 16, 16,512)

flatten = Flatten()(P4)
dense = Dense(512, activation='tanh')(flatten) #(None, 1024)

dense = Dense(256, input_shape=(512,), activation='tanh')(dense)#(None, 64)
reshape = Reshape((16, 16, 1))(dense) #(None, 8,8,1)

# up6 = Conv2D(512, 2, activation = 'tanh', padding = 'same', kernel_initializer = 'he_normal')(UpSampling2D(size = (2,2))(reshape))#(64,64,1024")
up6 = Conv2DTranspose(512, 2, strides = 2, activation = 'tanh', padding = 'same', kernel_initializer = 'he_normal')(reshape)
merge6 = concatenate([conv4,up6], axis = 3)
conv6 = Conv2D(512, 3, activation = 'tanh', padding = 'same', kernel_initializer = 'he_normal')(merge6)
SD6 = SpatialDropout2D(150/512)(conv6)
conv6 = Conv2D(512, 3, activation = 'tanh', padding = 'same', kernel_initializer = 'he_normal')(conv6)

# up7 = Conv2D(256, 2, activation = 'tanh', padding = 'same', kernel_initializer = 'he_normal')(UpSampling2D(size = (2,2))(SD6))
up7 = Conv2DTranspose(256, 2, strides=2, activation = 'tanh', padding = 'same', kernel_initializer = 'he_normal')(SD6)
merge7 = concatenate([conv3,up7], axis = 3)
conv7 = Conv2D(256, 3, activation = 'tanh', padding = 'same', kernel_initializer = 'he_normal')(merge7)
#SD7 = SpatialDropout2D(25/256)(conv7)
conv7 = Conv2D(256, 3, activation = 'tanh', padding = 'same', kernel_initializer = 'he_normal')(conv7)
```

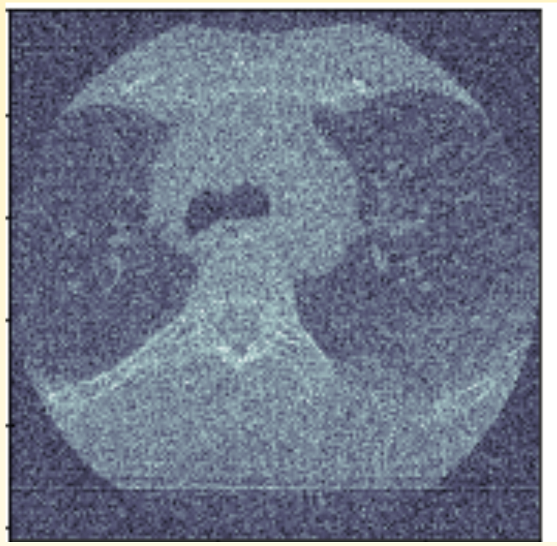
CNNs : The U-net network for denoising

Coding with Keras/Tensorflow, activation function for denoising: **tanh**

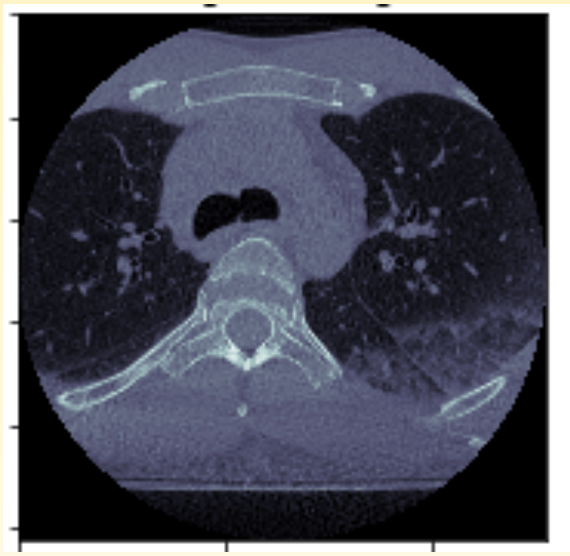
Hardware: GPU, **NVIDIA GeForce RTX 2080,** 11Go RAM, 4352 cores, 420 GFlops (double)

1 - Train the network with an as huge as possible database of couple of noisy/ground truth images.

Database : 600 couples of images of 512x512. Training time: 1h30 for 70 epochs.



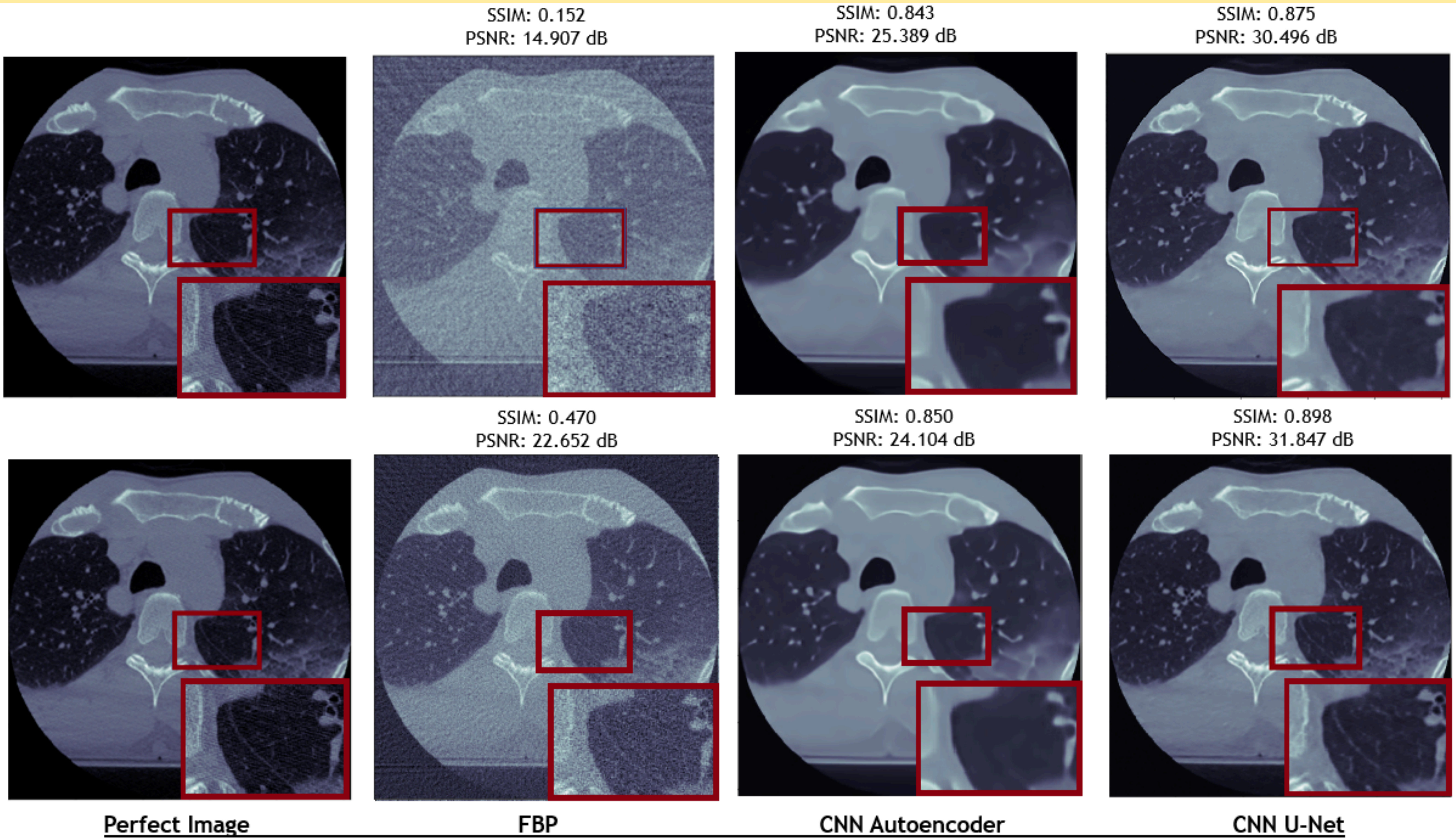
Input : noisy image



Output : perfect image

2 - Test performances of the network on unseen noisy images.
Prediction time : about 50ms

CNNs : The U-net network for denoising

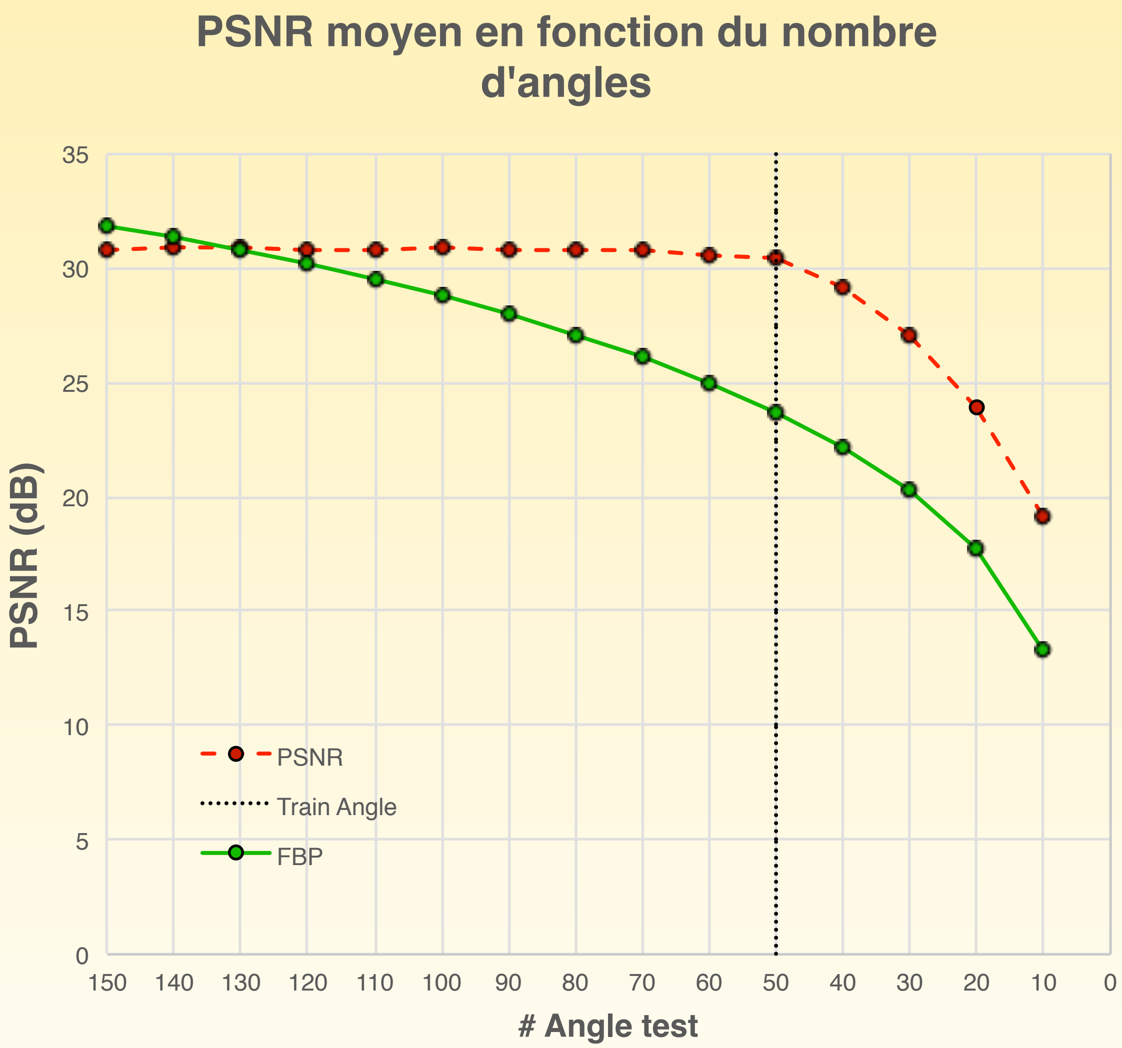


U-net Denoising approaches

Method

1 - Train the U-net with **very** noisy images (with **only** 50 projection angles).

2 - Test performances with images of any level of noise (results on 50 images).

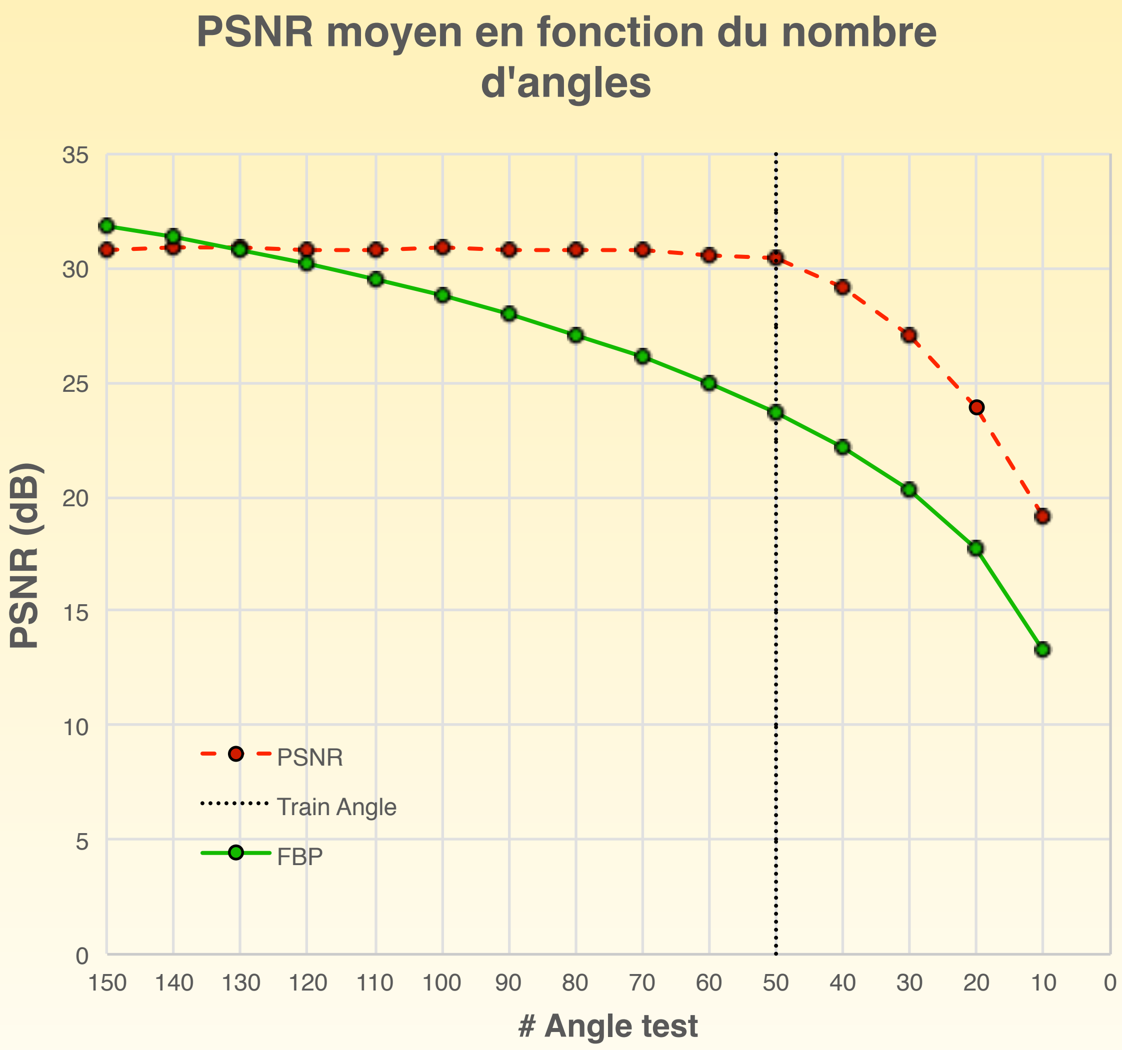


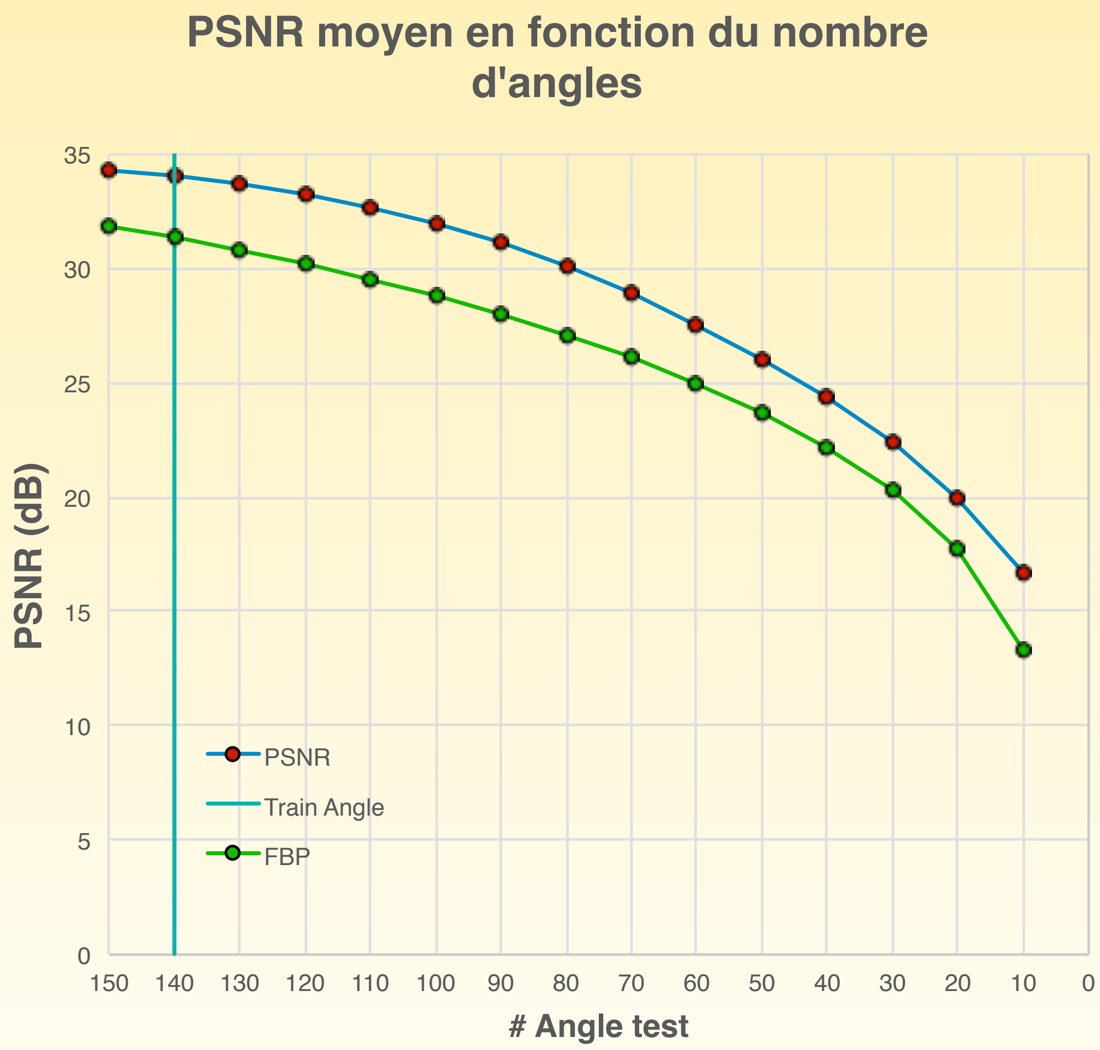
U-net Denoising approaches

Method

1 - Train the U-net with **hardly** noisy images (with 140 projection angles).

2 - Test performances with images of any level of noise.





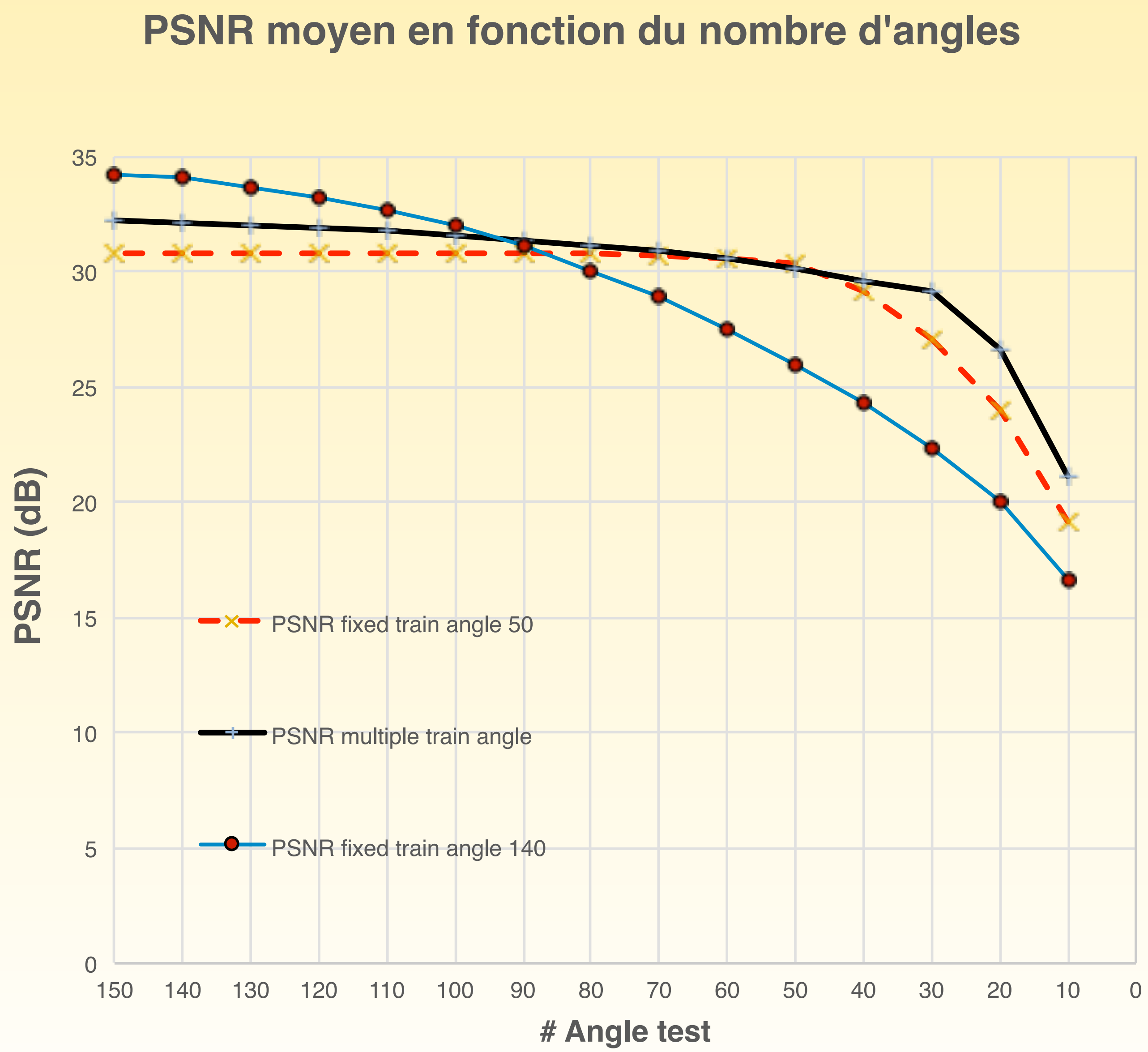
U-net Denoising approaches

Method

1 - Train the U-net with images of **any level of noise** :

20% of 30, 20% of 60, 20% of 90 , 20% of 120, 20% of 150 angles.

2 - Test performances with images of any level of noise.



Outline

1 - Biomedical imaging context : Computerized tomography (CT Scan)

2 - Convolutional Neural Networks (CNNs) for imaging

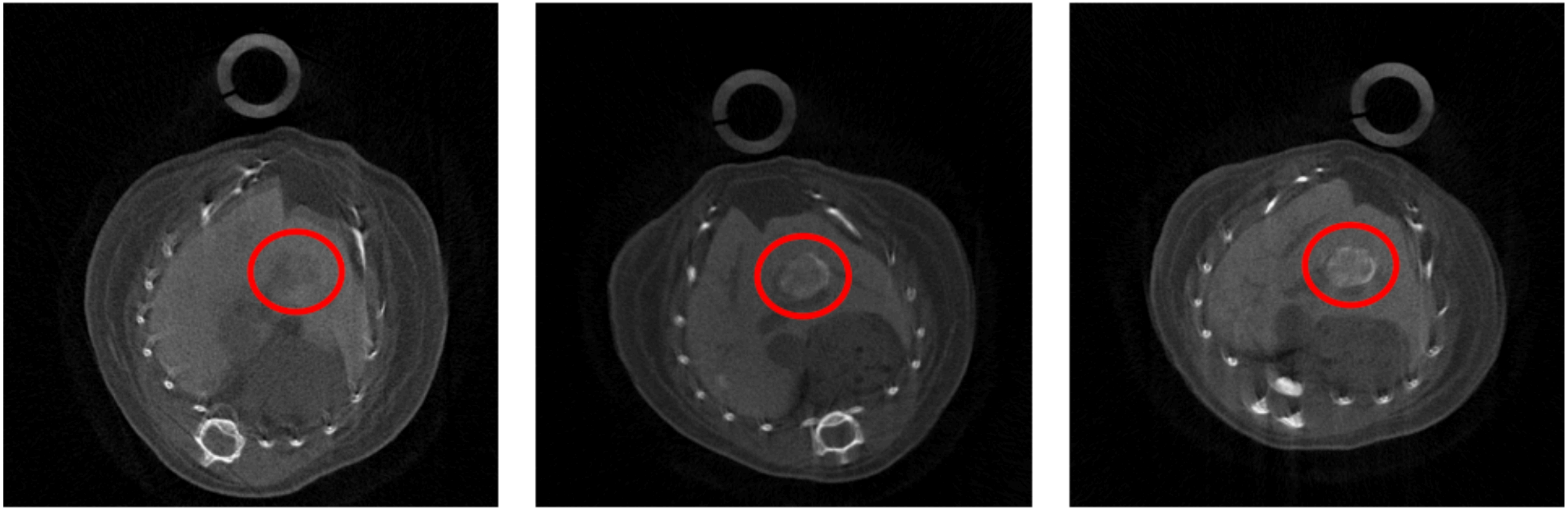
3 - Denoising issues

4 - Segmentation issues

5 - Other issues

6 - Conclusion

Why Deep Learning in CT segmentation ?

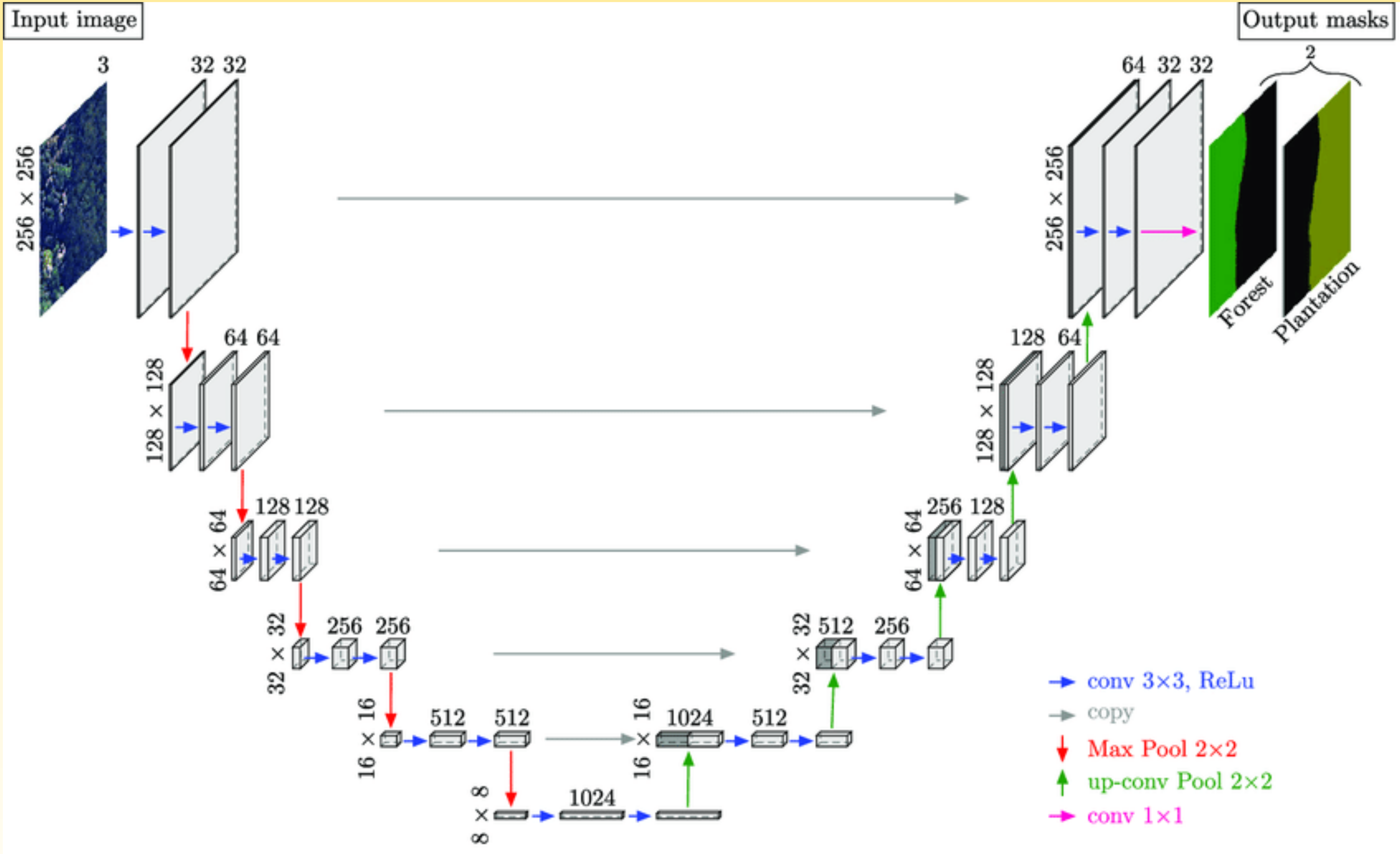


Scan 1 : 25/11/2020 Scan 2 : 03/12/2020 Scan 3 : 09/12/2020

Three transverse slices of the same mouse imaged at weekly intervals.

Different morphological classes/shapes of tumors

CNNs : The U-net network for segmentation



CNNs : Supervised segmentation

Coding with Keras/Tensorflow, activation function for segmentation:

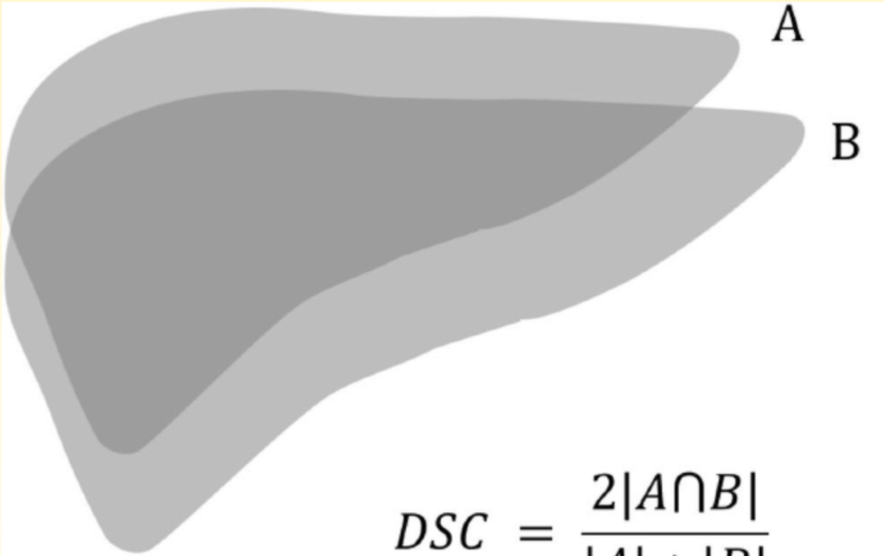
- **ReLu** then **softmax** or **sigmoid** (1 label) on the last layer.

Hardware: GPU, **NVIDIA GeForce RTX 2080,** 11Go RAM, 4352 cores, 420 GFlops (double)

1 - Train the network with a as huge as possible database of couple of images/manually segmentation.

Database : more than 10,000 couples of images of 512x512.
Training time: 1h30 for 50 epochs.

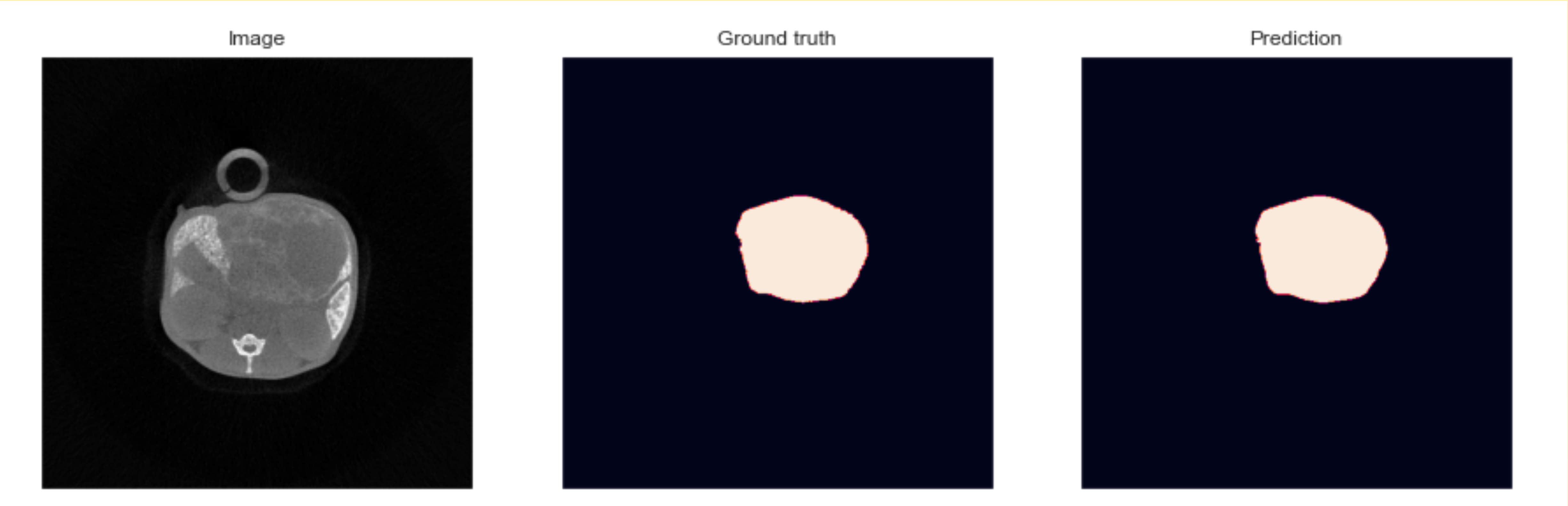
2 - Test performances of the network on unseen images with tumor.
Prediction time: about 50-100ms
Metric: DICE score in [0, 1].



$$DSC = \frac{2|A \cap B|}{|A| + |B|}$$

DSC: Dice similarity coefficient

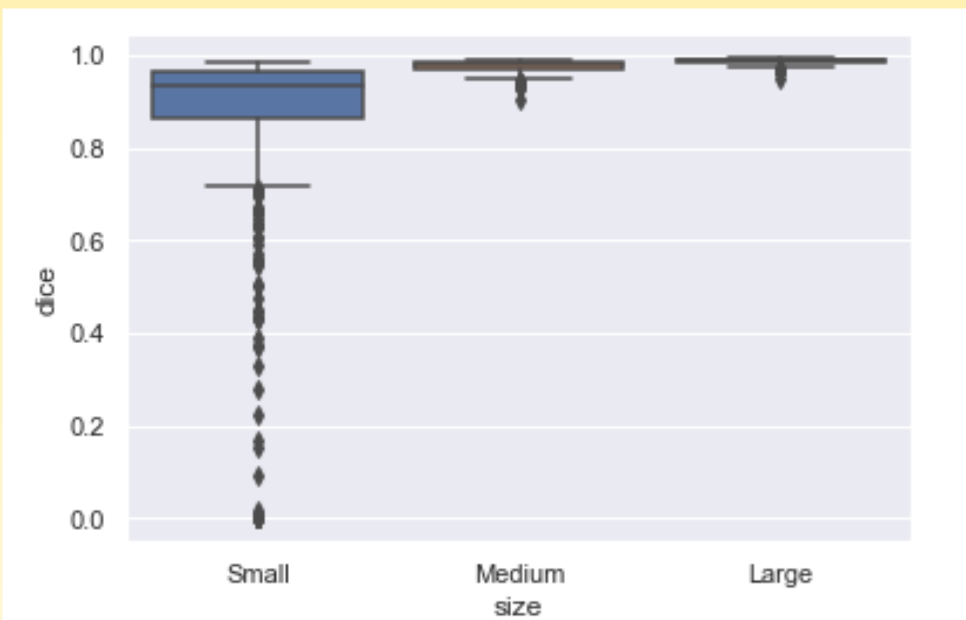
Deep Learning in CT Segmentation Liver / Tumor



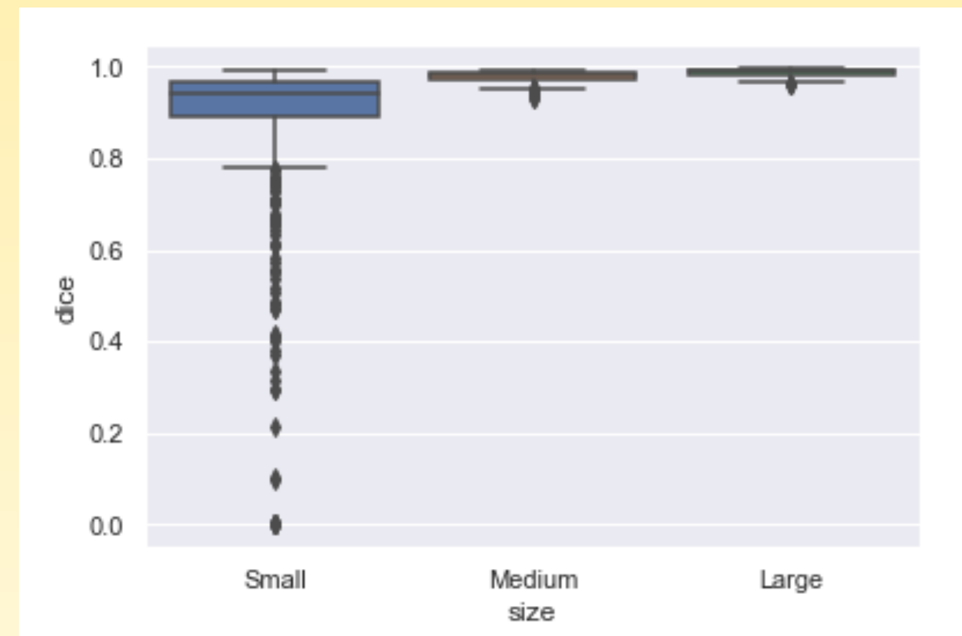
Deep Learning in CT Segmentation Liver / Tumor

DICE score as a function of tumor size:

Small: < 50mm²
Medium: 50 mm² < size < 100mm²
Large : > 100 mm²



Database with all tumors



Database with Stratification

	Réseau de référence	Stratification par taille
DICE moyen	0,916 +/- 0,157	0,929 +/- 0,132
DICE small	0,866 +/- 0,196	0,888 +/- 0,165
DICE medium	0,974 +/- 0,013	0,977 +/- 0,011
DICE large	0,987 +/- 0,008	0,987 +/- 0,006

Outline

1 - Biomedical imaging context : Computerized tomography (CT Scan)

2 - Convolutional Neural Networks (CNNs) for imaging

3 - Denoising issues

4 - Segmentation issues

5 - Other issues

6 - Conclusion

Numerous issues (only) in the context of CT

Not possible today to read everything.

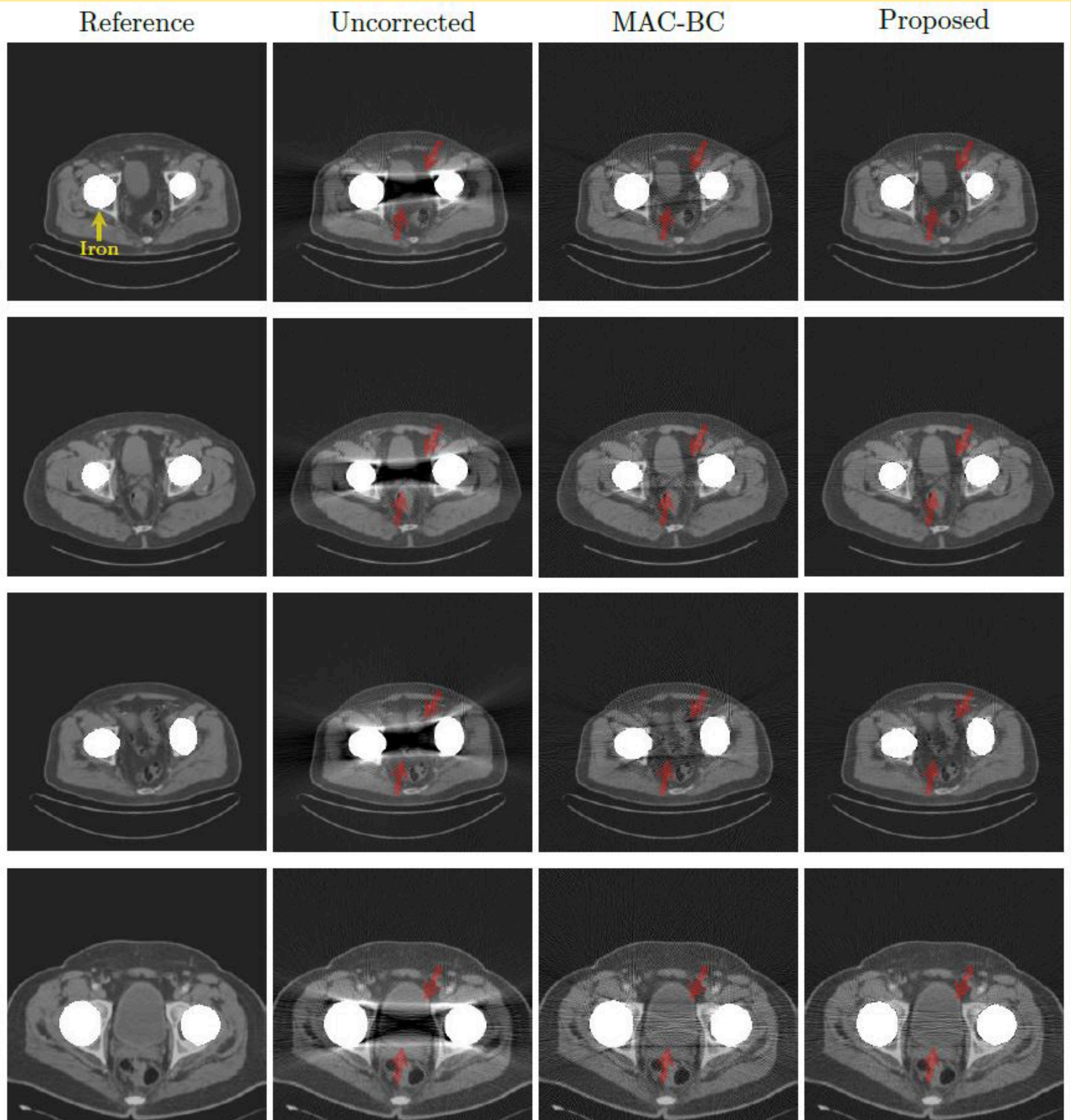
All the proposed methods were at least at the level of the gold standard in:

- Metal artifact reduction, [Park et al., Aug. 2017]
- Low Dose CT reconstruction, [Hammernik et al., Nov 2017]
- Scatter correction for spectral CT, [Xu et al., Jan. 2018]
- Monochromatic CT reconstruction, [Jin et al., June 2017]
- Material decomposition in spectral CT, [Lu et al., May 2018]
- Spectral distorsion corrections in spectral CT, [Touch et al., July 2016]
- CT segmentation with/without contrast agent
- CT registration

and…

- COVID-19 based detection in lung CT images
- …

Metal artifacts reduction



- [Park et al., Aug. 2017]

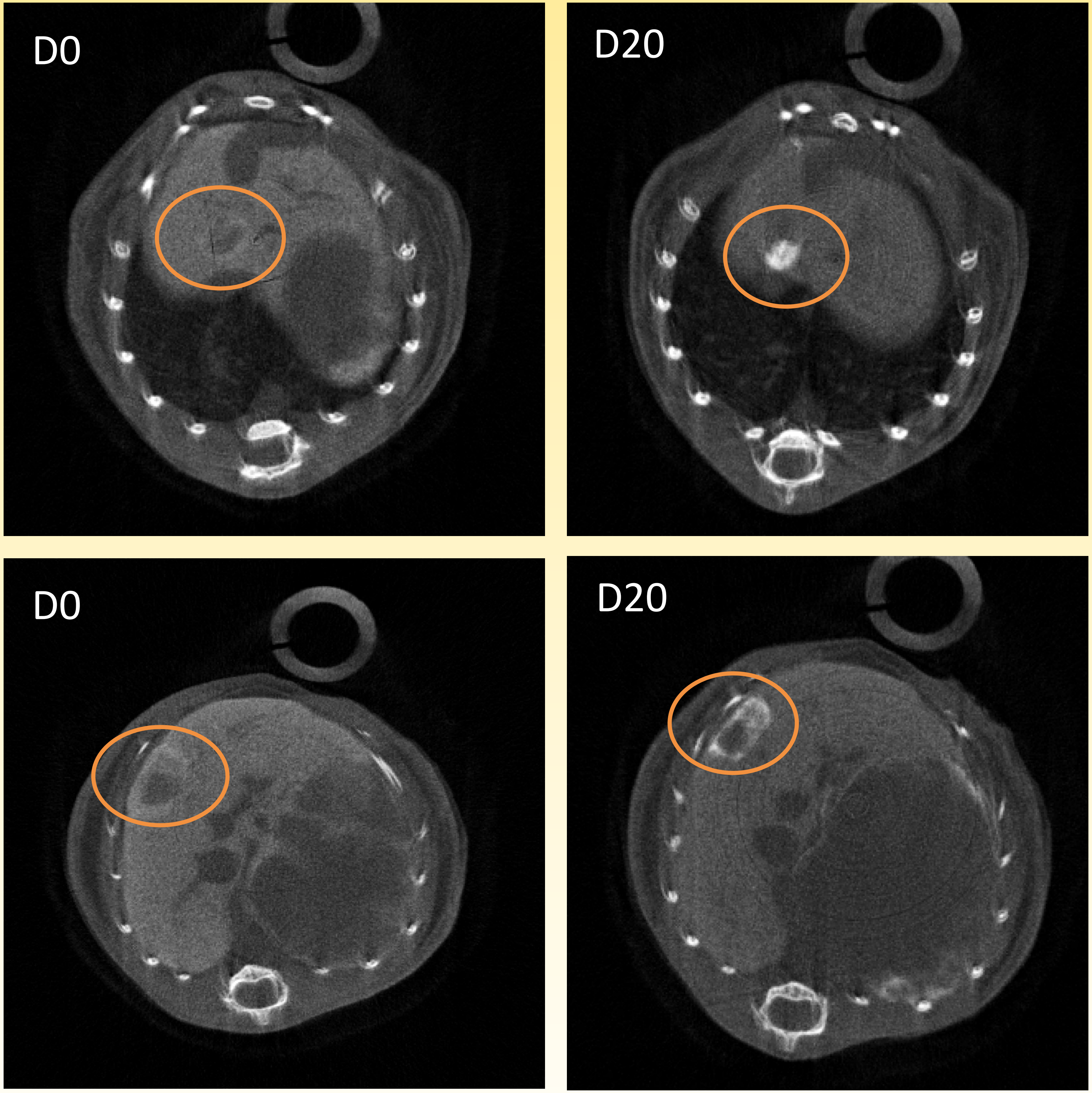
Data

Conclusions

CNNs based on U-net architectures appears to be efficient on numerous problems in biomedical imaging. Their use is everywhere.

Compare performances of GPUs vs. other hardwares for CNNs: are GPUs optimal ?

Can these U-net based CNNs be embedded closer to the sensor ?



Tumor evolution examples (at D0 in the left and D20 in the right)