Dose activities boosted with AI from lab LaTIM

Jing Zhang (Post-doc), Chi-Hieu Pham, Dimitris Visvikis, Julien Bert

jing.zhang@univ-brest.fr

26th April, 2023

Laboratoire de Traitement de l'Information Médicale
Laboratory of Medical Information Processing

Outline

Part 1: Monte Carlo simulation for dose estimation in radionuclide therapy

Part 2: Deep learning approach for improving the statistical quality of dose distribution

Part 1
Monte Carlo simulation for dose estimation in radionuclide therapy

Motivation

1. Radionuclide therapy (Lu177-PSMA)
2. Dose estimation in pretherapy
3. Monte Carlo simulation

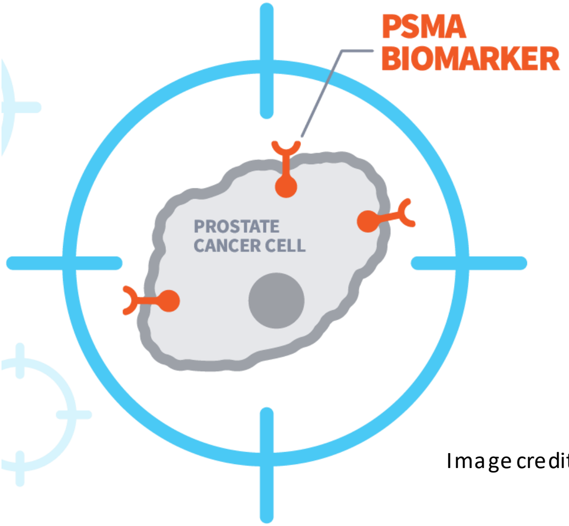
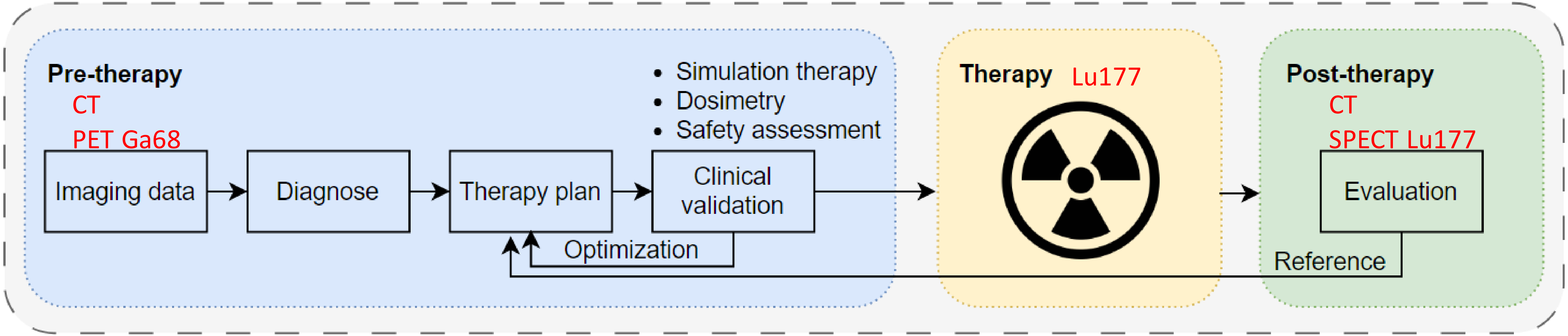


Image credit : https://www.scanforpsma.com/scan-for-psma/



Process of disease diagnose and therapy

Monte Carlo simulation for dosimetry

OpenGATE on Python

Macro parameters

- Initialization settings
- Geometry
- Physics
- Source
- Actors
- Simulation

Reference:
https://opengate-python.readthedocs.io/en/latest/index.html

Monte Carlo simulation

OpenGATE on Python

- **Import libraries**
- Initialization settings
- Geometry
- Physics
- Sources
- Actors
- Simulation

Pseudo code

```
import opengate_core as g4
import opengate as gate
import numpy as np
from scipy.spatial.transform import Rotation
import pathlib
import os
import sys
```

Monte Carlo simulation

OpenGATE on Python

- Import libraries
- **Initialization settings**
- Geometry
- Physics
- Sources
- Actors
- Simulation

Pseudo code

```
sim = gate.Simulation()
sim.add_material_database('GateMaterials.db')
ui = sim.user_info
ui.g4_verbose = False
ui.g4_verbose_level = gate.INFO
ui.running_verbose_level = gate.RUN
ui.visu = False
ui.number_of_threads = 1
ui.random_engine = 'MersenneTwister'
ui.random_seed = 123654
Bq = gate.g4_units('Bq')
m = gate.g4_units('m')
um = gate.g4_units("um")
keV = gate.g4_units('keV')
sec = gate.g4_units('second')
```

Monte Carlo simulation

OpenGATE on Python

- Import libraries
- Initialization settings
- Geometry
- Physics
- Sources
- Actors
- Simulation

Pseudo code

```
world = sim.world
world.size = [2 * m, 2 * m, 2 * m]
world.material = "G4_AIR"
patient = sim.add_volume('Image', 'patient')
patient.image = ct_filename
ct_info = gate.read_image_info(str(patient.image))
patient.mother = 'world'
f1 = str("Schneider2000MaterialsTable.txt")
f2 = str("Schneider2000DensitiesTable.txt")
tol = 0.05 * gcm3
patient.voxel_materials, materials =
gate.HounsfieldUnit_to_material(tol, f1, f2)
patient.dump_label_image = label_filename
```

Monte Carlo simulation

OpenGATE on Python

- Import libraries
- Initialization settings
- Geometry
- **Physics**
- Sources
- Actors
- Simulation

Pseudo code

```
p = sim.get_physics_user_info()
p.physics_list_name = "G4EmStandardPhysics_option3"
p.enable_decay = True
p.apply_cuts = True
cuts = p.production_cuts
cuts.patient.gamma = 100 * um
cuts.patient.electron = 100 * um
cuts.patient.positron = 100 * um
cuts.patient.proton = 100 * um
```

Monte Carlo simulation

OpenGATE on Python

- Import libraries
- Initialization settings
- Geometry
- Physics
- **Sources**
- Actors
- Simulation

Pseudo code

```
source = sim.add_source("Voxels", "ion")
source.mother = "patient"
source.particle = "ion 71 177" # Lu177
source.image = pt_filename
source.half_life = 574560 * sec
source.position.type = "sphere"
source.position.radius = 1 * mm
source.position.translation =
gate.get_translation_between_images_center
(str(patient.image), str(source.image))
source.direction.type = "iso"
source.energy.type = "mono"
source.energy.mono = 0 * keV
source.activity = 1e4 * Bq / ui.number_of_threads
```

Monte Carlo simulation

OpenGATE on Python

- Import libraries
- Initialization settings
- Geometry
- Physics
- Sources
- **Actors**
- Simulation

Pseudo code

```
dose = sim.add_actor('DoseActor', 'dose')
dose.output = dose_filename
dose.mother = 'patient'
dose.size = ct_info.size
dose.spacing = ct_info.spacing
dose.img_coord_system = True
dose.uncertainty = True
dose.gray = True
stats =
sim.add_actor('SimulationStatisticsActor','stats')
stats.mother = 'world'
stats.track_types_flag = True
```

Monte Carlo simulation

OpenGATE on Python

- Import libraries
- Initialization settings
- Geometry
- Physics
- Sources
- Actors
- **Simulation**

Pseudo code

```
sim.run_timing_intervals = [[0, 1 * sec]]
sim.initialize()
sim.start()
sim.apply_g4_command("/run/verbose 2")
sim.apply_g4_command("/tracking/verbose 2")
sim.apply_g4_command("/event/verbose 2")
stats = sim.get_actor('stats')
stats.write(stats_filename)
```

PSMA Dataset

❖ Dataset is from Bern University Hospital. In collaboration with Prof. Kuangyu Shi, Dr. Song Xue in University of Bern, Switzerland.

Dataset info

- 21 prostate cancer patients.
- Modality: CT, PET, RTDOSE
- Preprocessing:
  - Convert
  - Resample
  - Crop

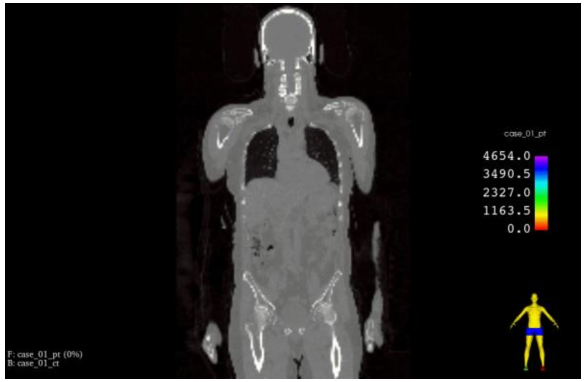
MC protocol

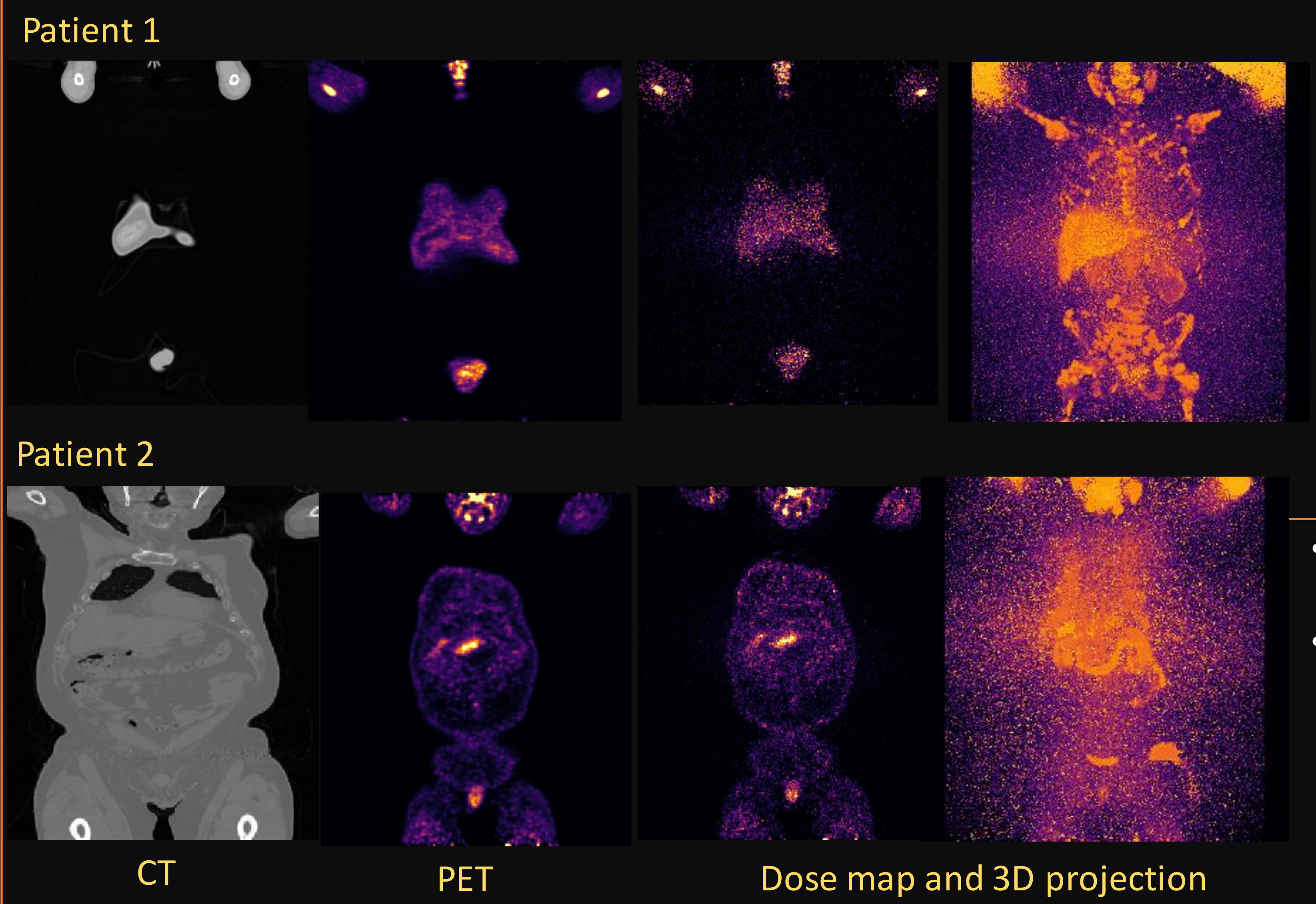
- Lu177 ion radiation
- Input geometry: CT
- Voxelized source: PET
- Radiation time: 20 minutes

Table 1: The PSMA dataset profile

No.	age(y)	height(cm)	weight(kg)	cycles	date
1	73	176	70	4	13/07/21
2	71	178	65	4	10/03/20
3	68	168	56	1	09/03/21
4	77	187	90	6	24/03/20
5	77	160	70	2	15/12/20
6	75	168	86	6	29/10/19
7	73	166	83	5	20/03/20
8	76	175	66	1	29/09/20
9	59	183	100	2	30/06/20
10	73	178	83	1	01/12/20
...					
21	73	180	86	4	13/04/21

Fused CT-PET





Simulation results

- Simulation time: ~5 hours (8-core CPU)
- Uncertainty: ~5% (local)

Summary

Subject	Radiation therapy on prostate cancer
Plan	Dosimetry estimation
Method	Monte Carlo simulation (OpenGATE)
Question	Can DL boost?

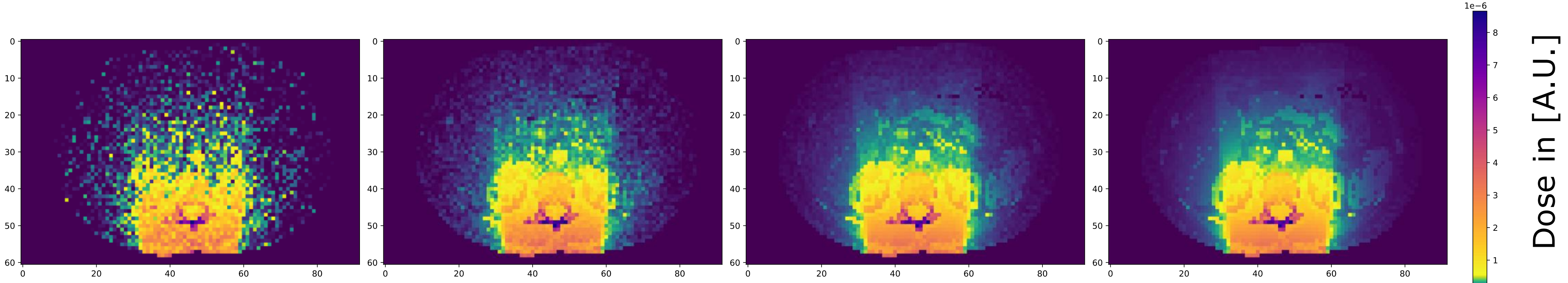
	MC	DL
Concept	Physical model based	Non-linear regression, data driven
Speed	5 h/patient	5 s/patient
Operability	MACRO	Learning; inferencing

- Dosimetry in therapy plan is vital and time critical before radiation therapy.
- Our lab is investigating the dose map prediction using both OpenGATE (Python) and DL methods.
- Technically, the DL-based method performs faster than the method of MC.
- There is space for improving the prediction results (Data scale and DL model performance).

Part 2
Improving the statistical quality of Monte Carlo dose distribution by using a deep learning approach

Motivations

MC simulations



Nb of particles	5.10^5	5.10^6	5.10^7	5.10^8
Avg uncertainty	**60**±30 %	**25**±17 %	**9**±10 %	**3**±9 %

GATE[1]*	**40** s	**413** s	**~1** h	**~10** h
GGEMS[2]**	0.7 s	0.8 s	1.7 s	10.7 s

* x1 CPU core Intel Xeon W-2223 3.6 GHz
** NVIDIA GeForce RTX 3090

[1] http://www.opengatecollaboration.org, Jan et al. GATE V6: a major enhancement of the GATE simulation platform enabling modelling of CT and radiotherapy *Phys Med Biol* **56** 881–901, 2011
[2] https://ggems.fr, Bert et al. Geant4-based Monte Carlo simulations on GPU for medical applications *Phys Med Biol* **58** 5593–611, 2013

Monte Carlo Denoising in Medical Physics (deep learning)

Denoising approach

- Filtering methods

  Naqa I E, Kawrakow I, Fippel M, Siebers J V, Lindsay P E, Wickerhauser M V, Vicic M, Zakarian K, Kauffmann N and Deasy J O
  A comparison of Monte Carlo dose calculation denoising techniques *Phys. Med. Biol.* **50** 909–22, 2005

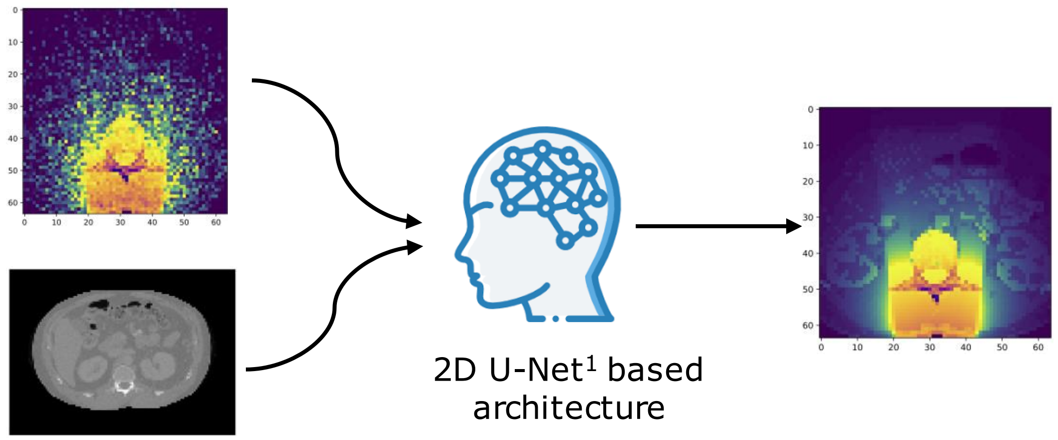
- Recent deep learning methods

  R. Neph, Y. Huang, Y. Yang, and K. Sheng, *Artificial Intelligence in Radiation Therapy*, vol. 11850, pp. 137–145, 2019

  T. Bai, B. Wang, D. Nguyen, and S. Jiang, *Mach. Learn.: Sci. Technol.*, vol. 2, no. 2, p. 025033, 2021

  S. Martinot, N. Bus, M. Vakalopoulou, C. Robert, E. Deutsch, and N. Paragios, *Medical Image Computing and Computer Assisted Intervention*, vol. 12904, pp. 499–508, 2021.

Example of simple Deep learning approach



Aims

- **Extreme case (highly undersampled)**
- **Complex distribution (heterogeneity)**
- **Inpainting / denoising**
- **Generic approach**

[1] Ronneberger O, Fischer P and Brox T *Medical Image Computing and Computer-Assisted Intervention – MICCAI*, vol 9351, pp 234–41, 2015

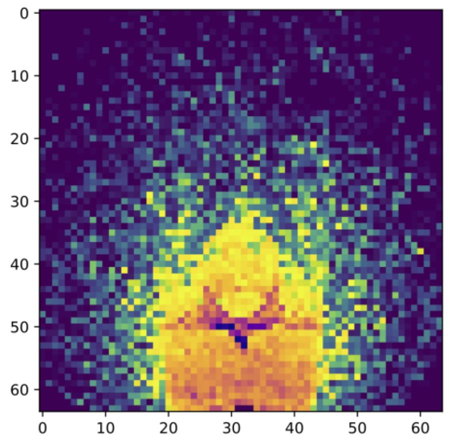
Simple deep learning approach

MC simulations (GGEMS[1])
- 82 CT scans (abdominal)
- Cone-beam photon source (single angulation, X-ray tube 120 kVp)
- 10^6 photons (undersampeld)
- $3x10^9$ photons (fully sampled)

Training dataset
- x40 2D slices for each patient (CT+dose maps)
  - Decimate (64x64 pixels), centered and padded
  - Norm intensity (0, 1),
  - Data augmentation (mirror)
- 6561 shuffled samples (2D undersampled, 2D fully sampled, 2D CT)
  - 5240 (80%) training
  - 1310 (20%) validation
  - 10 test

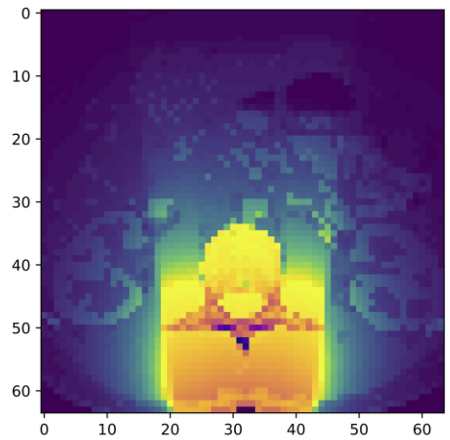
Training
- Adam optimizer, 10^{-4} learning rate, MSE loss function
- Training (200 epochs, 10 batches, 1 day):
  - 2D undersampled, 2D fully sampled
  - 2D undersampled, 2D fully sampled, 2D CT

Evaluation
- Using the 10 test samples
- MSE and abs. error.



Low sampling (input)

0.8 s*, uncer. 53±30%



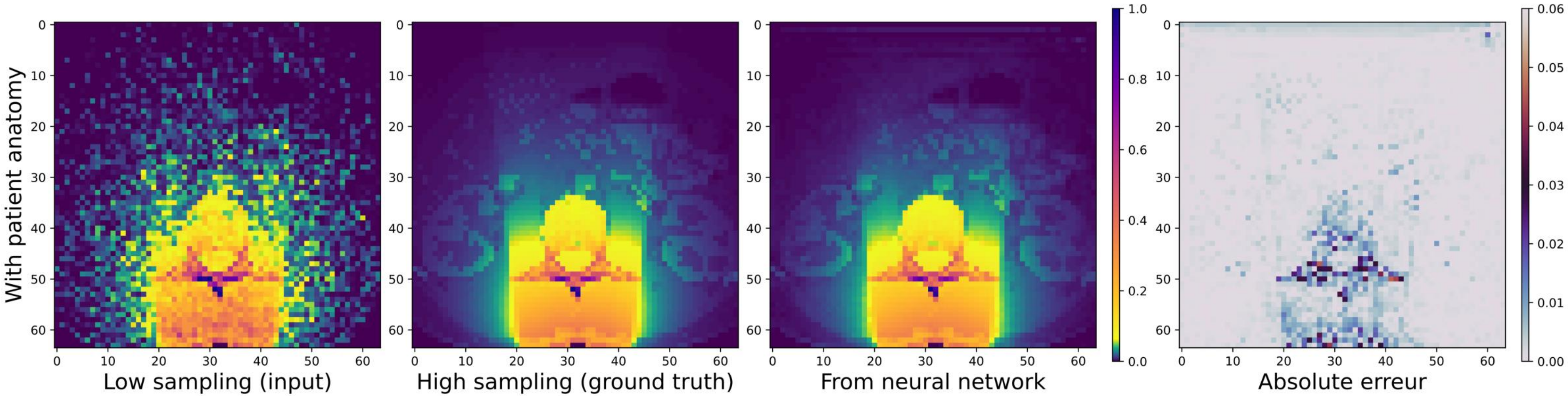
High sampling (ground truth)

642 s*, uncer. 1.2±0.7%

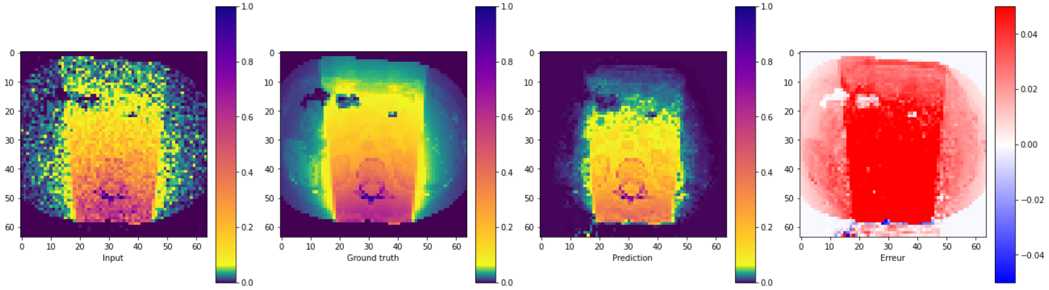
* NVIDIA RTX3090

[1] https://ggems.fr, Bert et al. Geant4-based Monte Carlo simulations on GPU for medical applications *Phys Med Biol* **58** 5593–611, 2013

Results

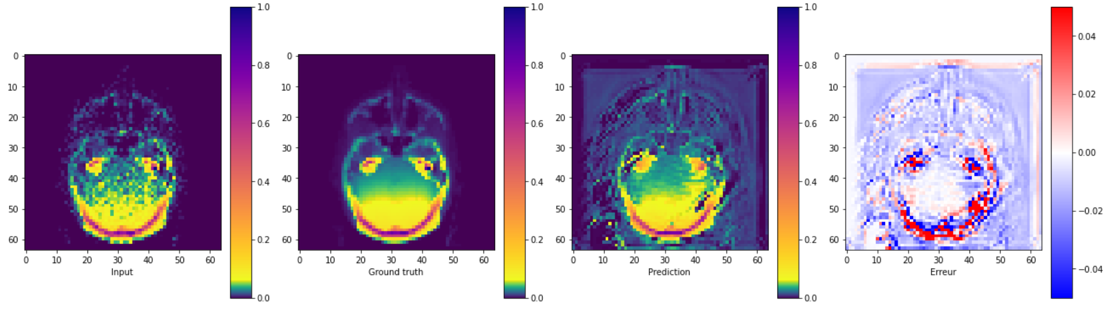


Different energy beam (from 120 kvp to single energy at 200 keV)



Low sampling (input) High sampling (ground truth) From neural network Absolute erreur

Different part of the body (abdo. to head & neck CT)

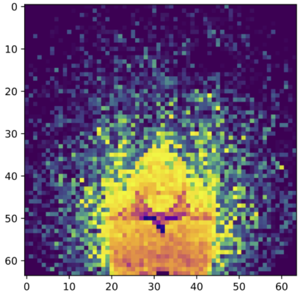


Low sampling (input) High sampling (ground truth) From neural network Absolute erreur

Aims
- Extreme case (highly undersampled)
- Complex distribution (heterogeneity)
- Inpainting / denoising
- Generic approach

Generic approach
- Training data set (dif. parts of the body)
- Including others kind of information
- Patch-based learning

Evaluations
- Different medical applications
- Compare with dif. existing VRT
- Compare with standard filtering methods



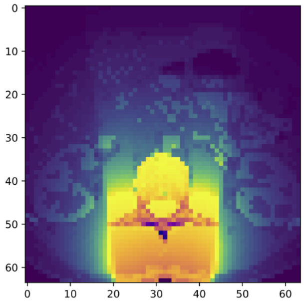
Low sampling (input)

10^6 photons

GATE **82 s**
GGEMS **0.8 s**

Processing time:
- IO images
- IO model
- GPU loading
- Inference

~ 25 s



High sampling (ground truth)

$3x10^9$ photons

GATE **~600 h**
GGEMS **642 s**

x1 CPU core Intel Xeon W-2223 3.6 GHz
NVIDIA GeForce RTX 3090

The end.

Thank you!