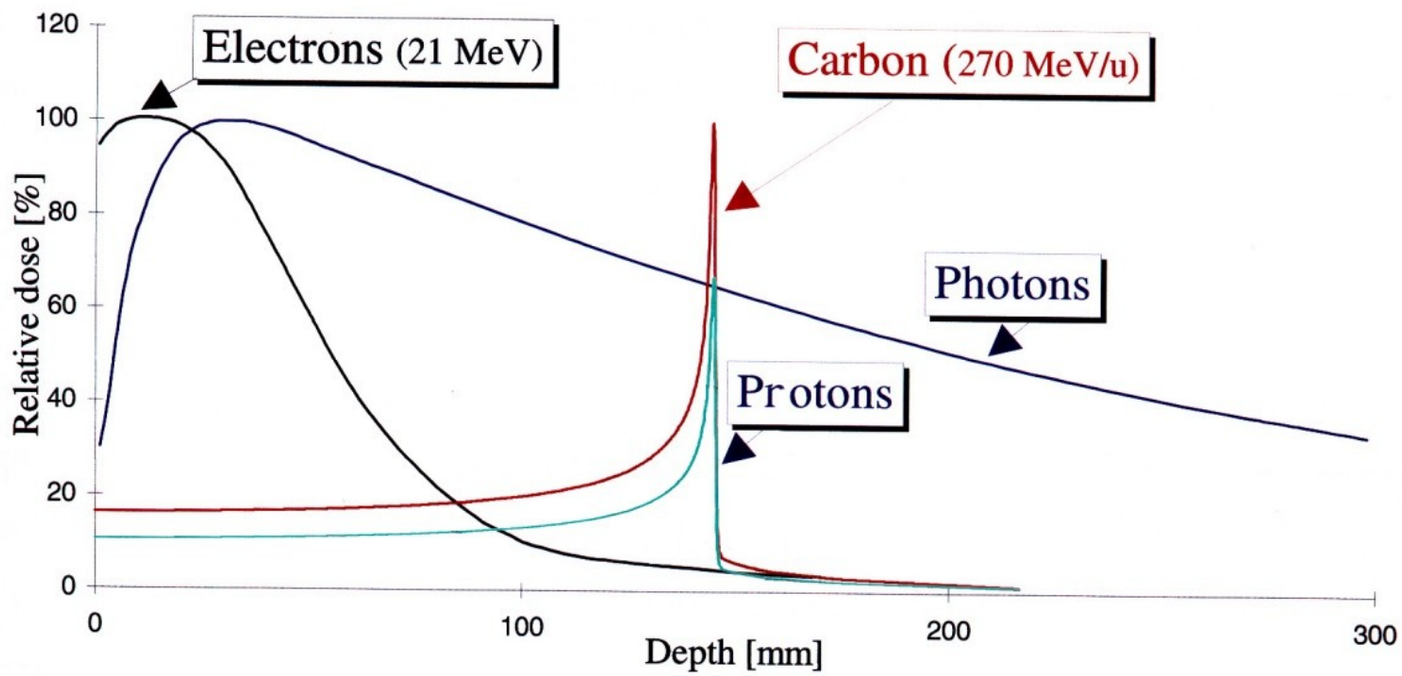
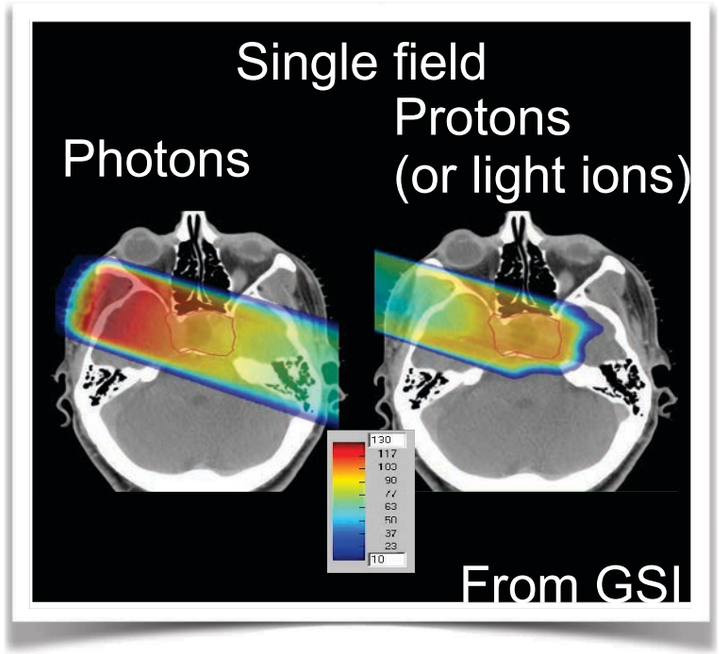
Deep Learning and Monte Carlo simulations, opportunities and challenges

carlo.mancini-terracciano@uniroma1.it

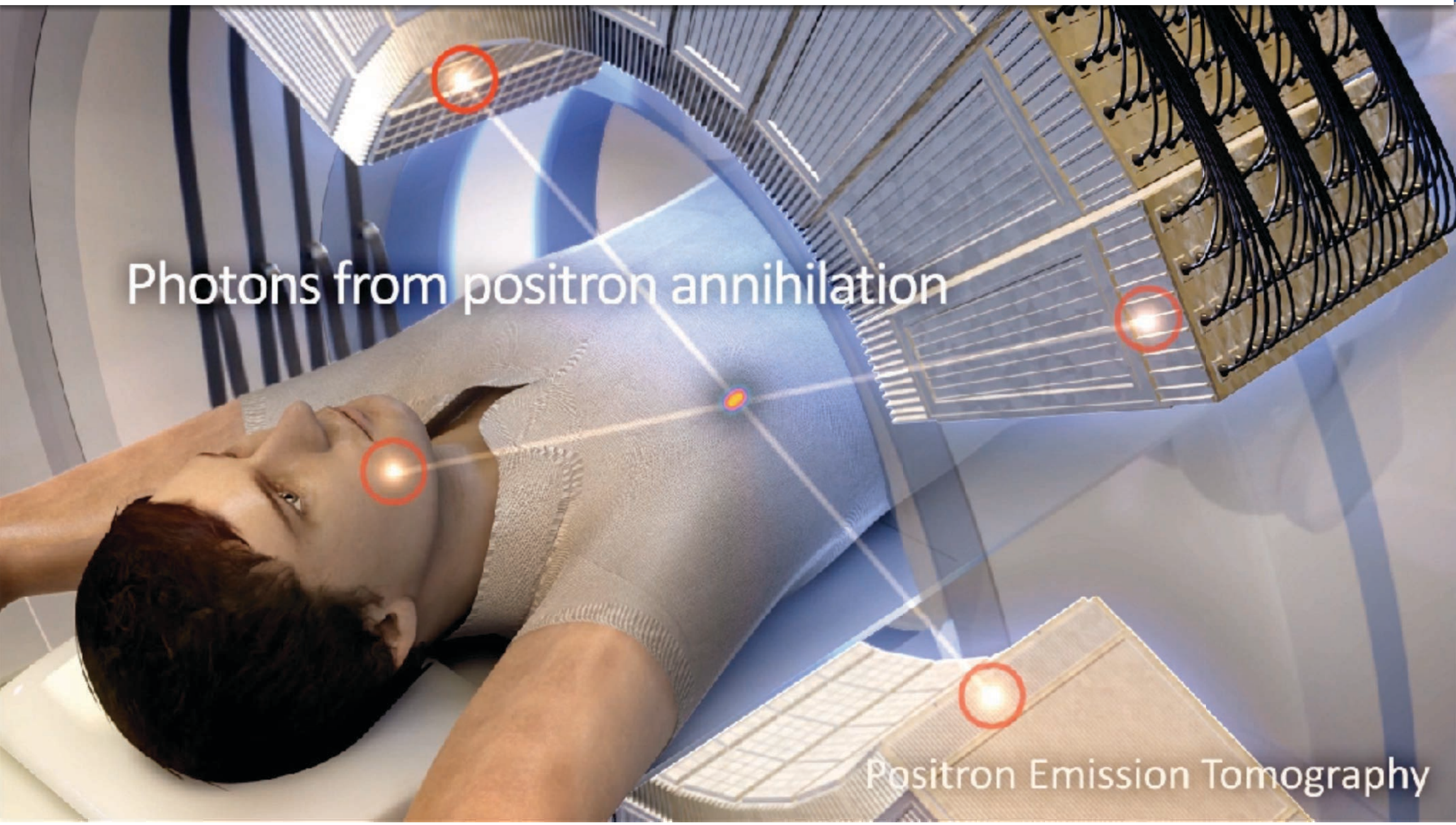
TECH-FPA PhD Retreat
LNGS - 18th February 2025

Radiotherapy





- Radiotherapy is, together with surgery, the most frequently used treatment for tumours



- The MC simulation is necessary to link the information
- The PCS is dispensable to link the information
- The PCS is necessary to link the information
- The MC simulation is dispensable to link the information

distribution with the dose distribution

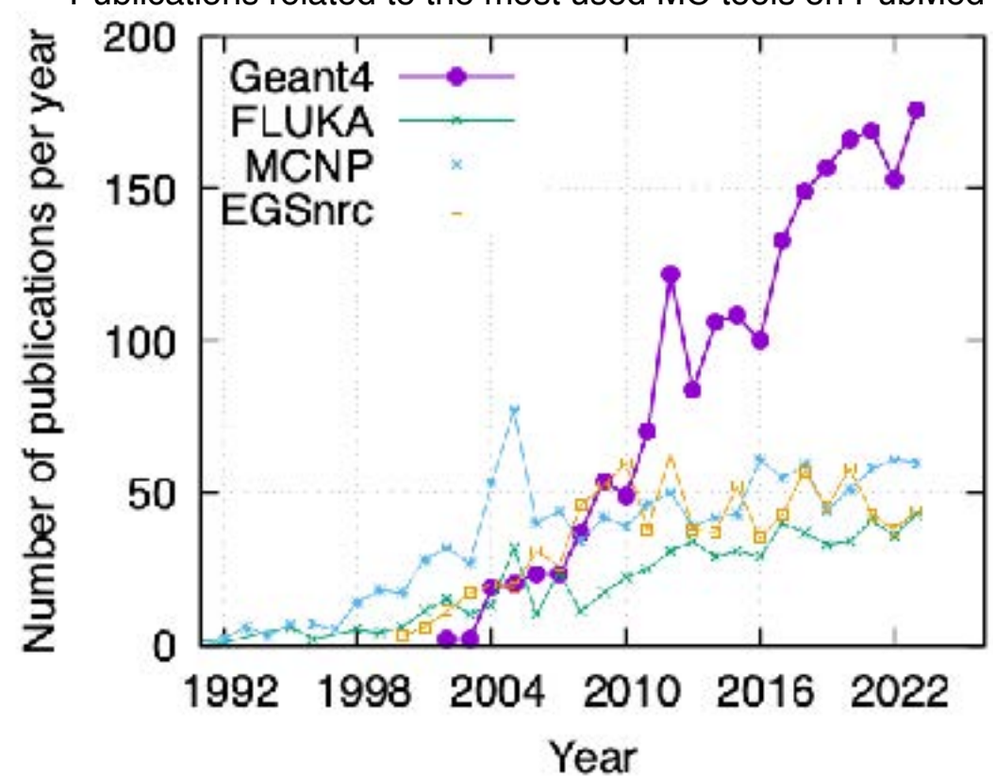
Geant4 (GEometry ANd Traking)

- Developed by an international collaboration
- Open source
- Written in C++ language
  - Takes advantage from the Object Oriented approach
  - It is multithread
- The most used MC tool for research in medical applications

[Geant4, a simulation toolkit Nucl. Inst. and Methods Phys. Res. A, 506 250-303

Geant4 developments and applications Transaction on Nuclear Science 53, 270-278]

Publications related to the most used MC tools on PubMed



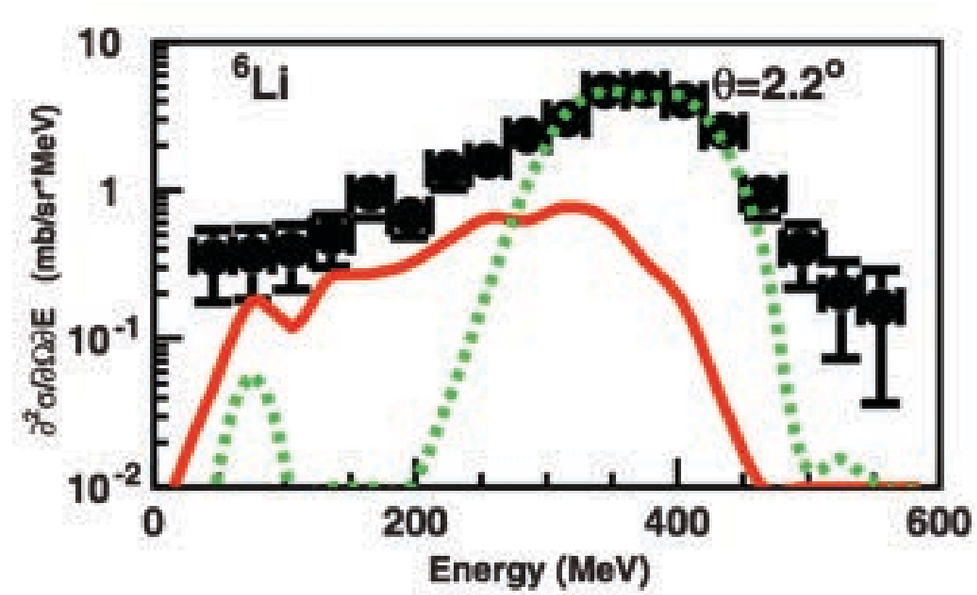
Problems in Geant4 below 100 MeV/u

- Despite the numerous and relevant application would use it, there is no dedicated model to nuclear interaction below 100 MeV/u in Geant4

- Many papers showed the difficulties of Geant4 in this energy domain:
  - Braunn et al. have shown discrepancies up to one order of magnitude in ^{12}C fragmentation at 95 MeV/u on thick PMMA target
  - De Napoli et al. showed discrepancy specially on angular distribution of the secondaries emitted in the interaction of 62 MeV/u ^{12}C on thin carbon target
  - Dudouet et al. found similar results with a 95 MeV/u ^{12}C beam on H, C, O, Al and Ti targets

- **Exp. data**
- **G4-BIC**
- **G4-QMD**

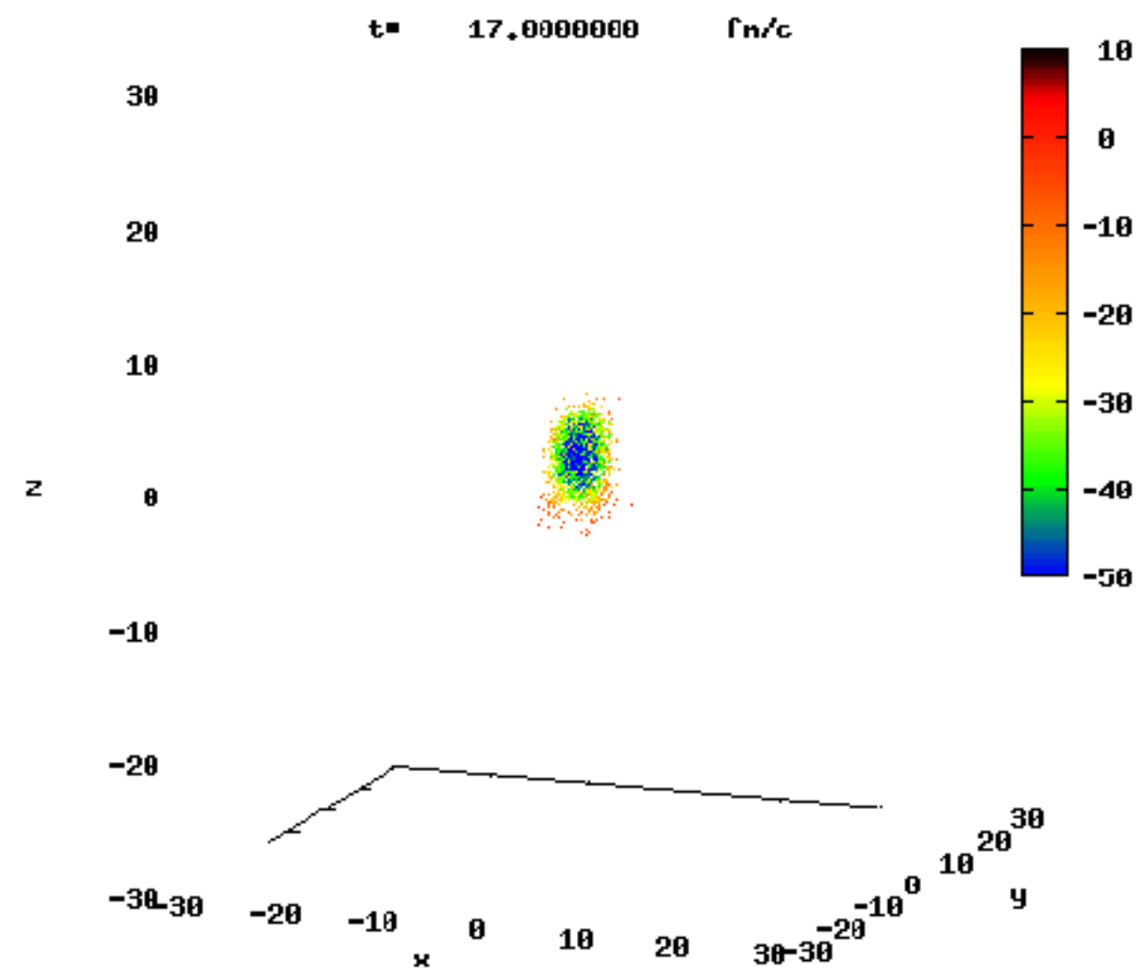
[Plot from De Napoli et al. Phys. Med. Biol., vol. 57, no. 22, pp. 7651–7671, Nov. 2012]



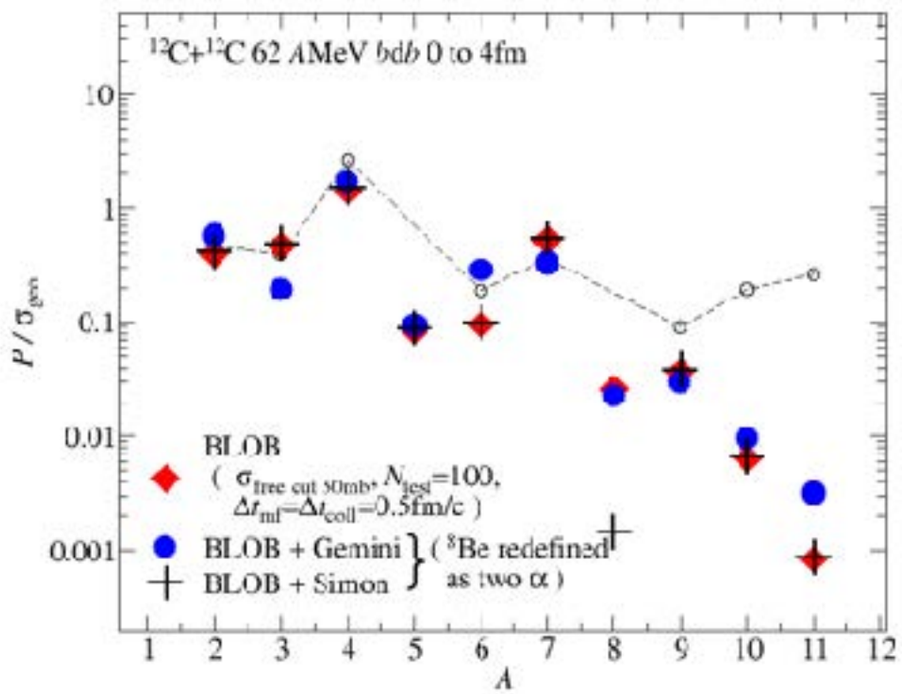
Cross section of the ^{6}Li production at 2.2 degree in a ^{12}C on ^{nat}C reaction at 62 MeV/u.

BLOB (Boltzmann-Lagevein One Body)

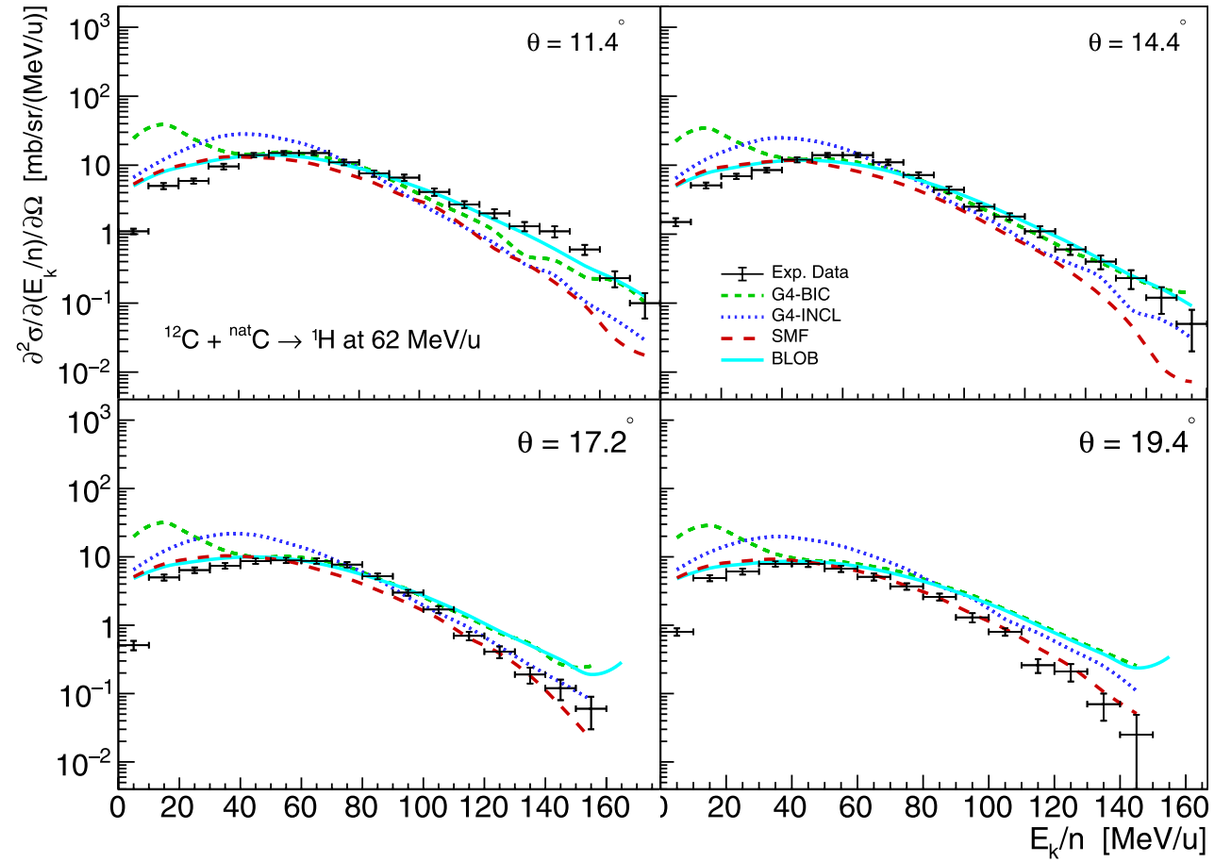
- Describes the time evolution of the density distribution
- Test-particle approach
- Self-consistent mean field + 2-body collisions
- Probability to find a nucleon in the phase space

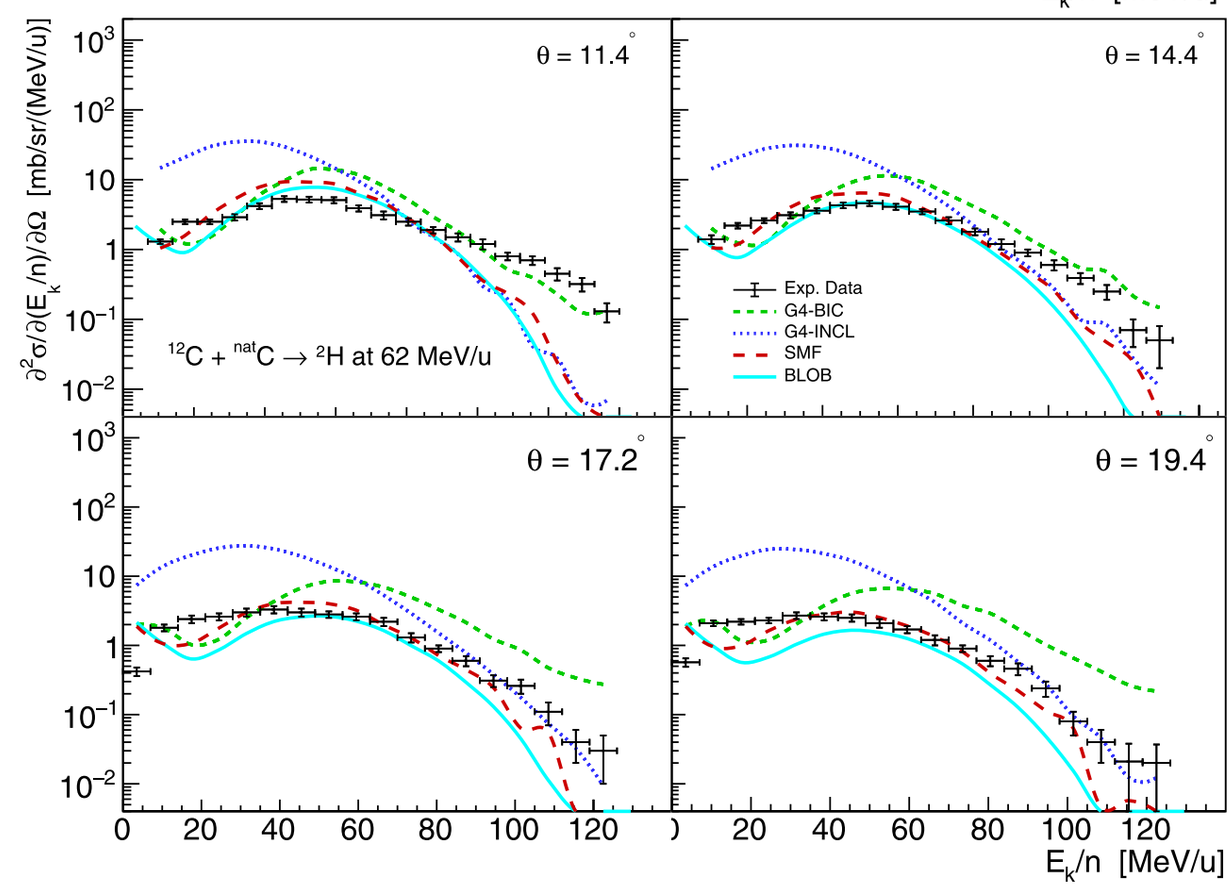


BLOB and Geant4



[C. Mancini-Terracciano et al. Preliminary results coupling "Stochastic Mean Field" and "Boltzmann-Langevin One Body" models with Geant4. Phys. Med. 67 (2019), pp. 116–122. https://doi.org/10.1016/j.ejmp.2019.10.026]





7

BLOB Code optimisations

- We optimised BLOB without changing the code structure (52% speed-up overall)
- Not enough for practical applications

Elapsed Time: 231.966s

CPU Time:	231.930s
Total Thread Count:	1
Paused Time:	0s

Top Hotspots

This section lists the most active functions in your application. Optimizing these hotspot functions typically results in improving overall application performance.

Function	Module	CPU Time
lapla	run-orig	176.281s
erff	libm.so.6	17.201s
define_two_clouds_rp	run-orig	9.658s
sortrx	run-orig	7.018s
powf	libm.so.6	5.377s
[Others]		16.403s

Elapsed Time: 110.235s

CPU Time:	110.223s
Total Thread Count:	1
Paused Time:	0s

Top Hotspots

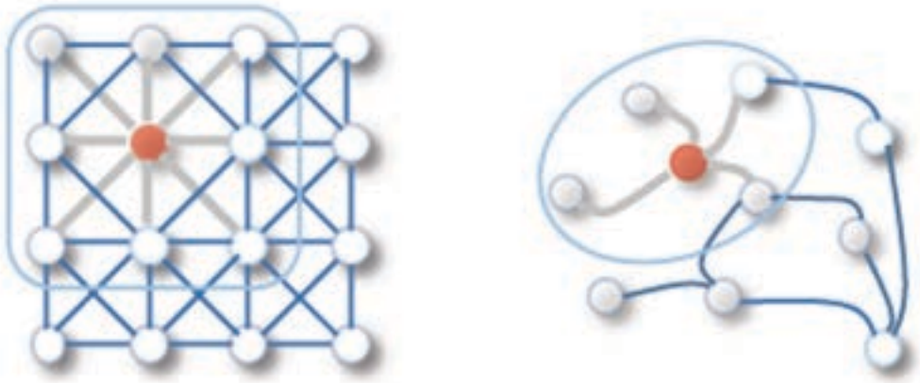
This section lists the most active functions in your application. Optimizing these hotspot functions typically results in improving overall application performance.

Function	Module	CPU Time
lapla	run	55.003s
erff	libm.so.6	17.038s
define_two_clouds_rp	run	9.051s
sortrx	run	7.450s
powf	libm.so.6	5.184s
[Others]		15.414s

Graph Network-based Simulators (GNS)

- We tested the possibility of emulating the whole interaction (i.e. producing the final state) using Graph CNN

- Encode graph-structured data

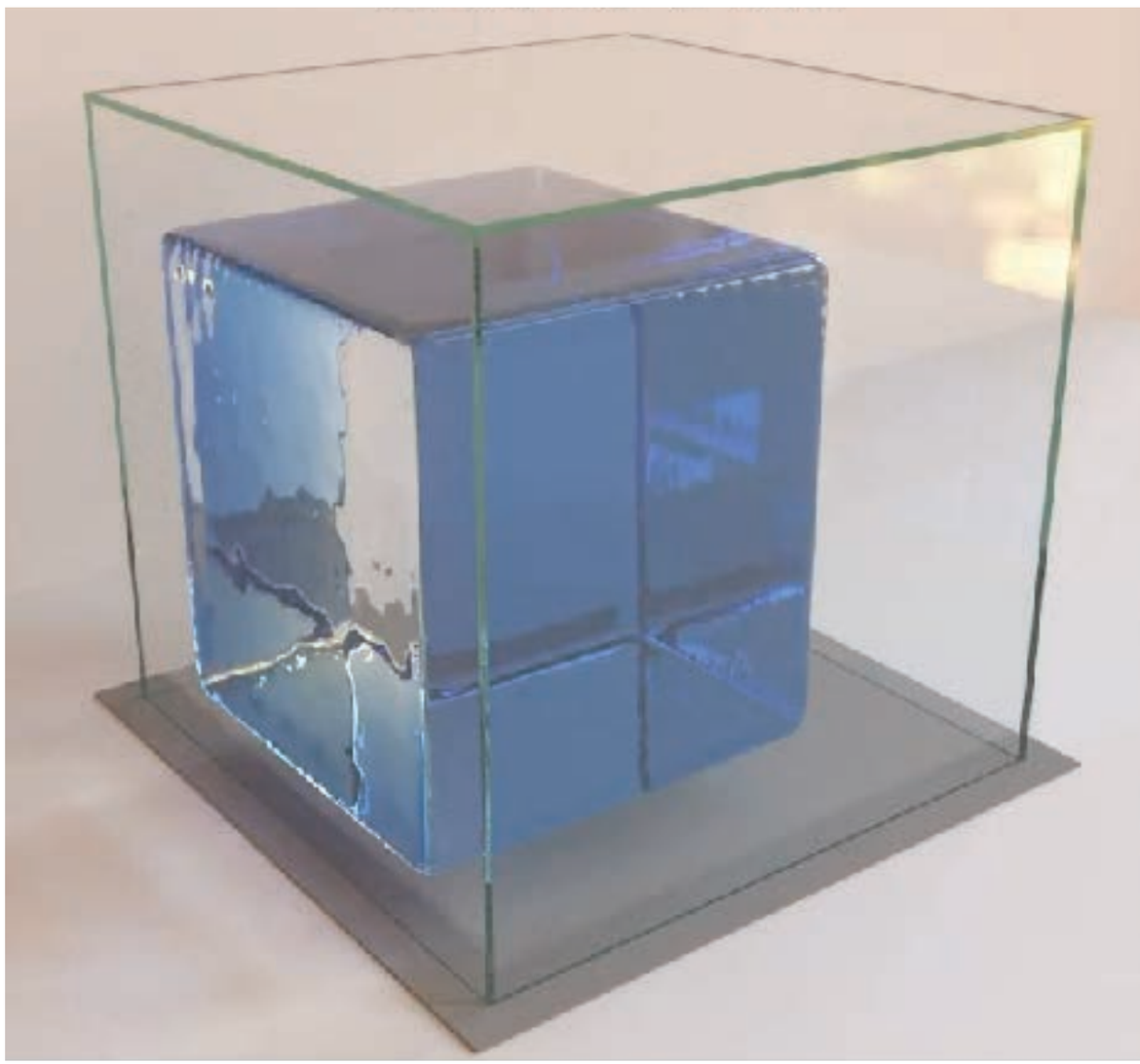


- Use of topological relationships among nodes

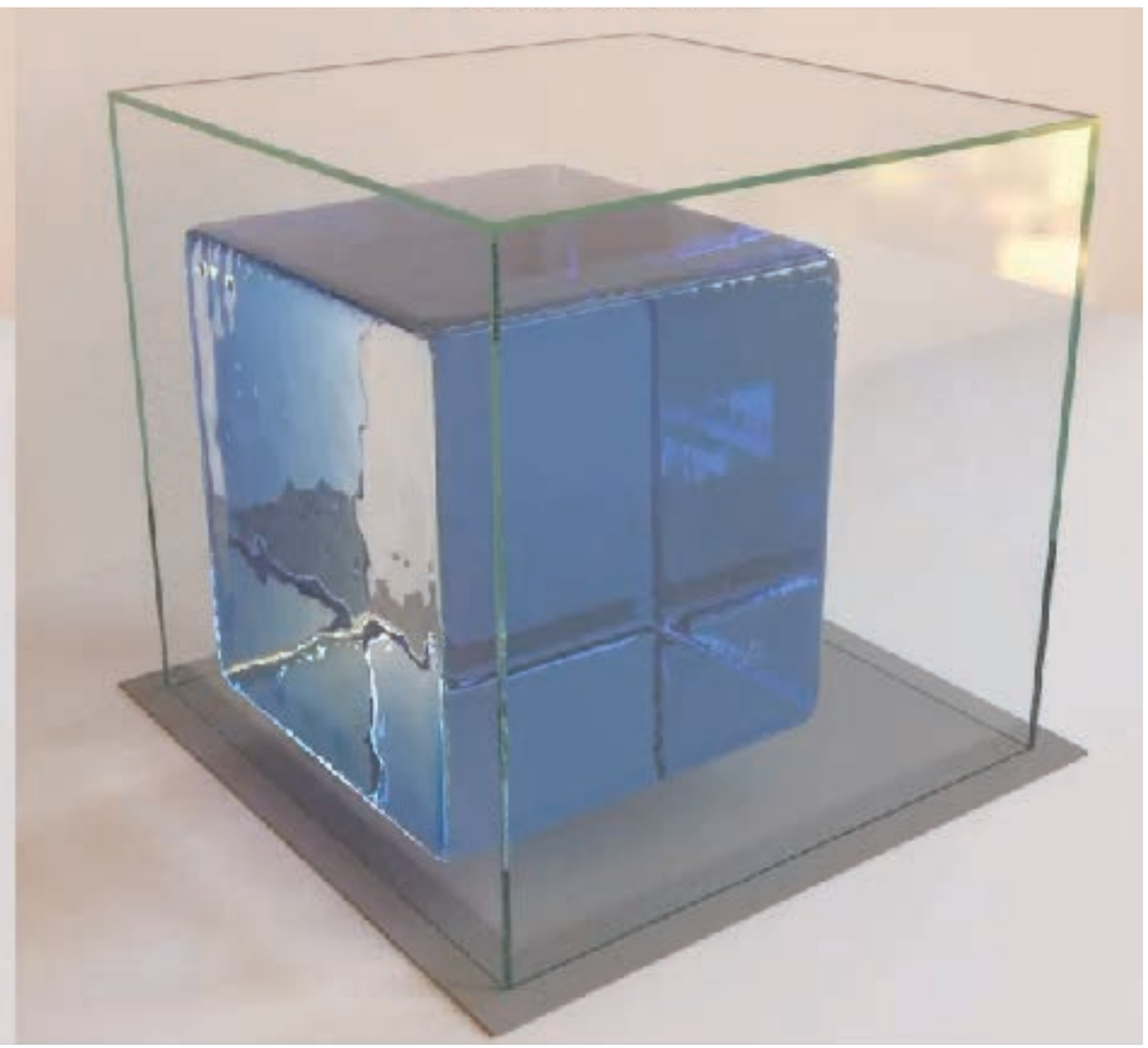
- Convolve the central node's representation with its neighbours' representations

Graph Network-based Simulators (GNS)

Ground truth



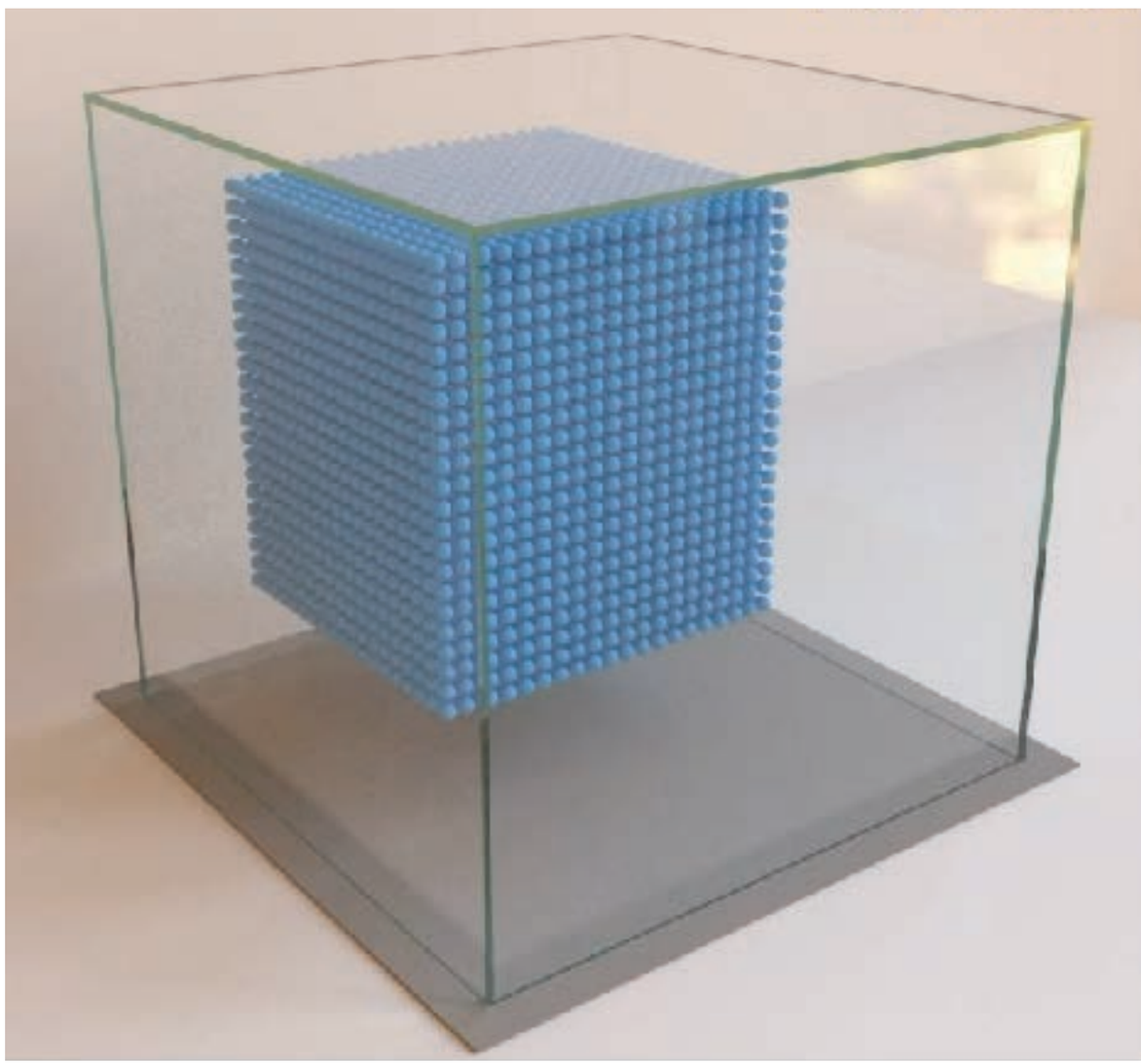
GNS



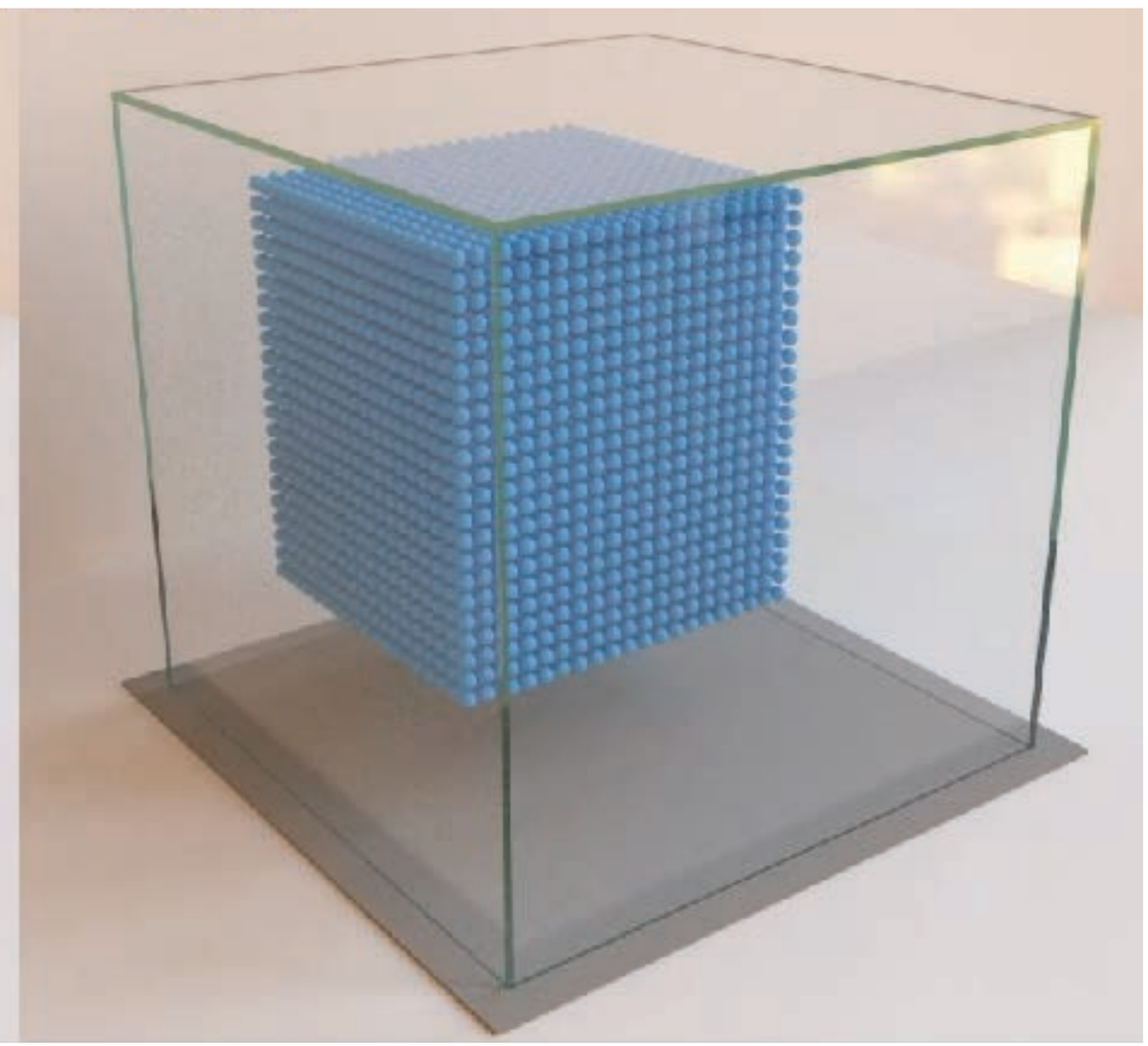
[A. Sanchez-Gonzalez et al. "Learning to simulate complex physics with graph networks." International Conference on Machine Learning. PMLR, 2020.]

Graph Network-based Simulators (GNS)

Ground truth



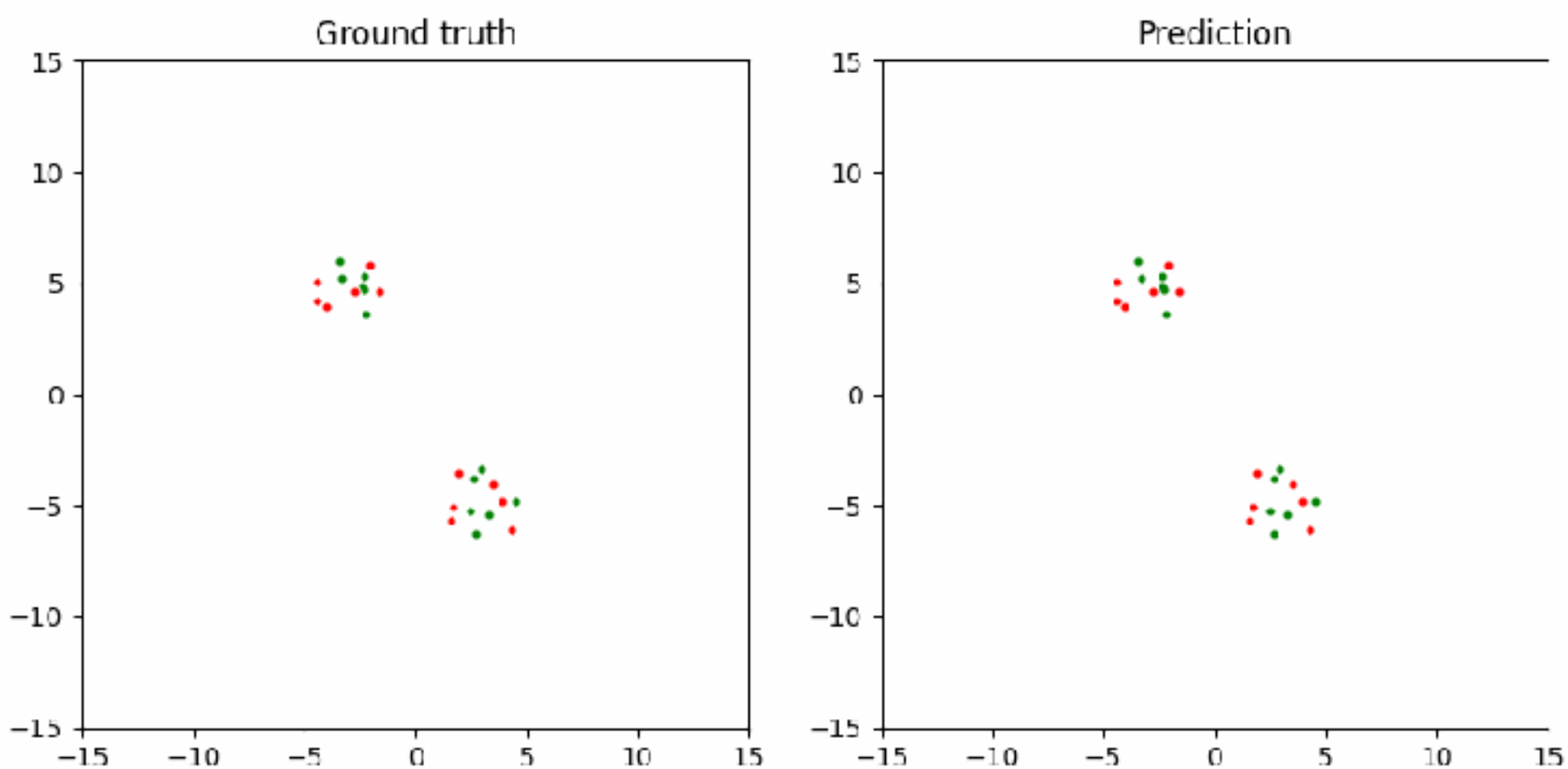
GNS



[A. Sanchez-Gonzalez et al. "Learning to simulate complex physics with graph networks." International Conference on Machine Learning. PMLR, 2020.]

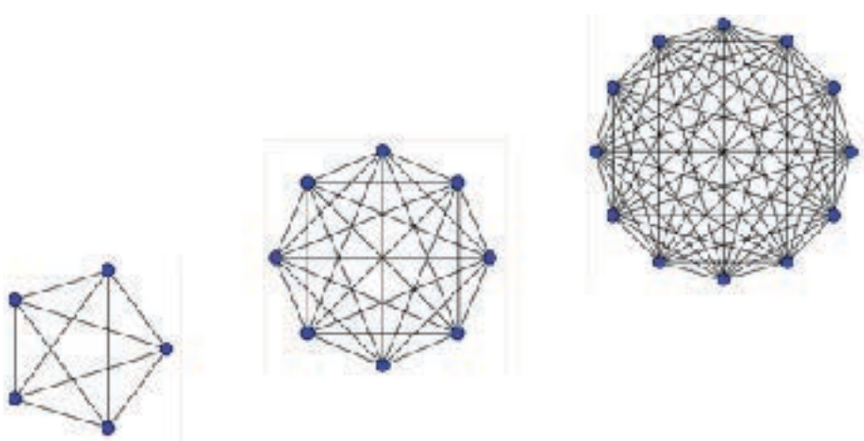
Emulating QMD with GNS

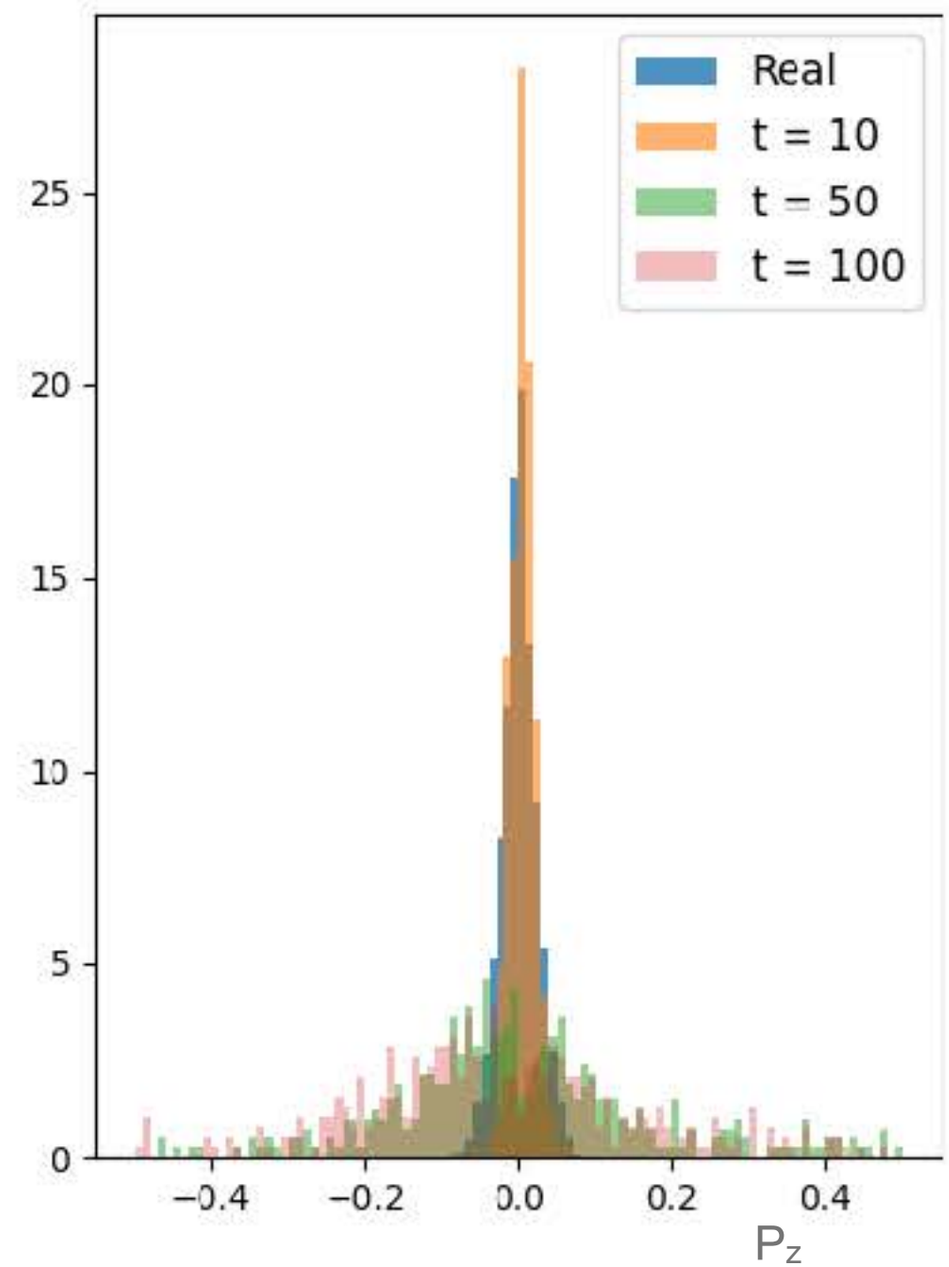
- G4-QMD to develop a demonstrator
- We trained a full connected GNN to reproduce the nuclear reaction dynamic
- Each nucleon is a node of the graph



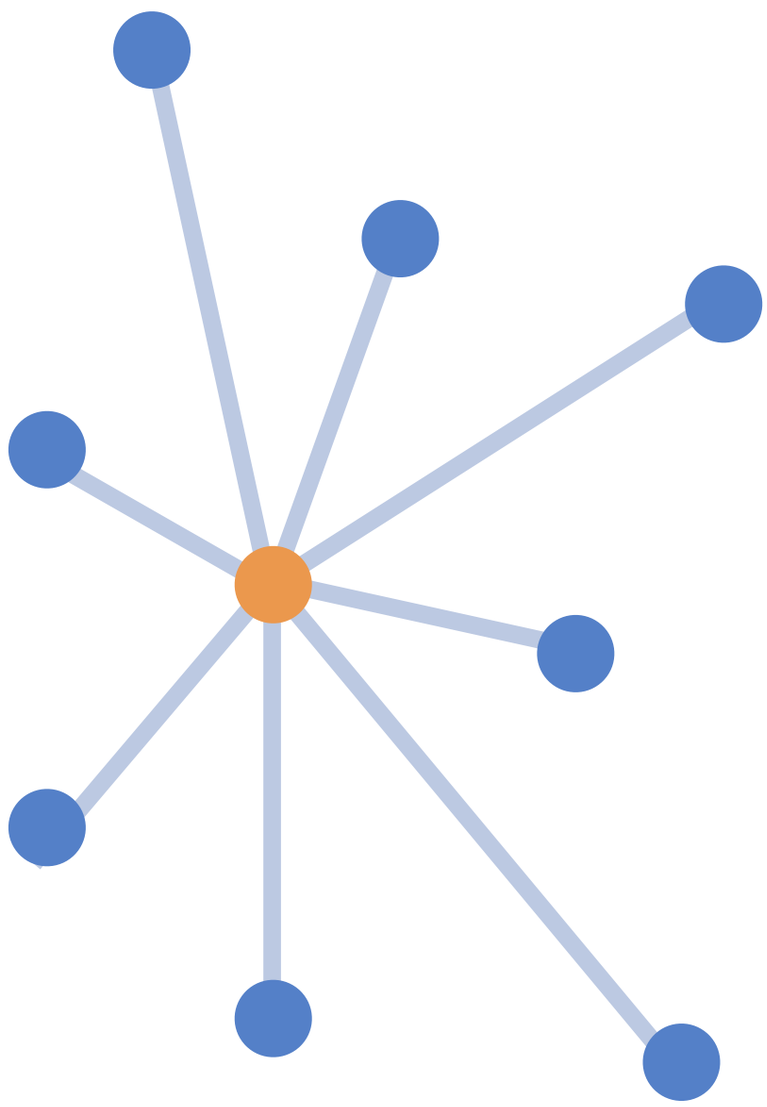
Emulating QMD with GNS

- Quantities are conserved on average
- Variance explodes increasing time steps
- $N_{edges} \propto N^2$



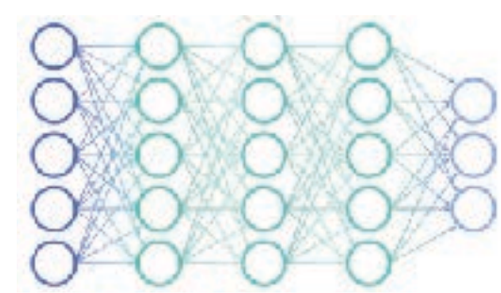


Emulating the Mean Field: DL model



$$V_i = \sum A_{ij} + \left(\sum B_{ij} \right)^{\gamma}$$

$$f\left(q_i, q_j, p_i, p_j, c_i, c_j\right) =$$



Hybrid model

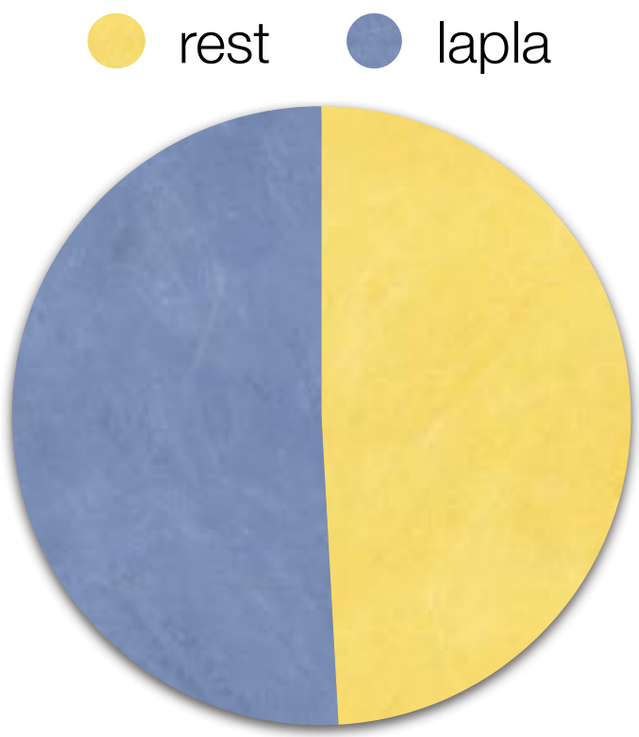
- Emulating the mean field calculation
- Keeping the model structure
- Enforcing physical conservation laws in the model
- Independent from the number of nucleons involved in the reaction

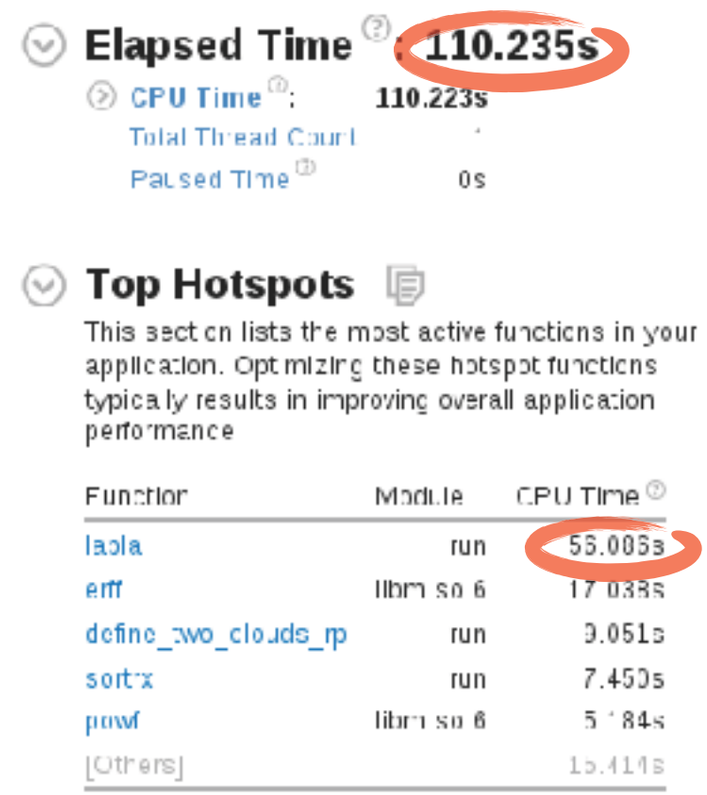
$$\{q_i, p_i, q_j, p_j\} \rightarrow$$

$$\rightarrow V_{ij}$$

Why the mean field

- It is the bottleneck of the BLOB computation





Elapsed Time 110.235s

CPU Time: 110.223s

Total Thread Count

Paused Time 0s

Top Hotspots

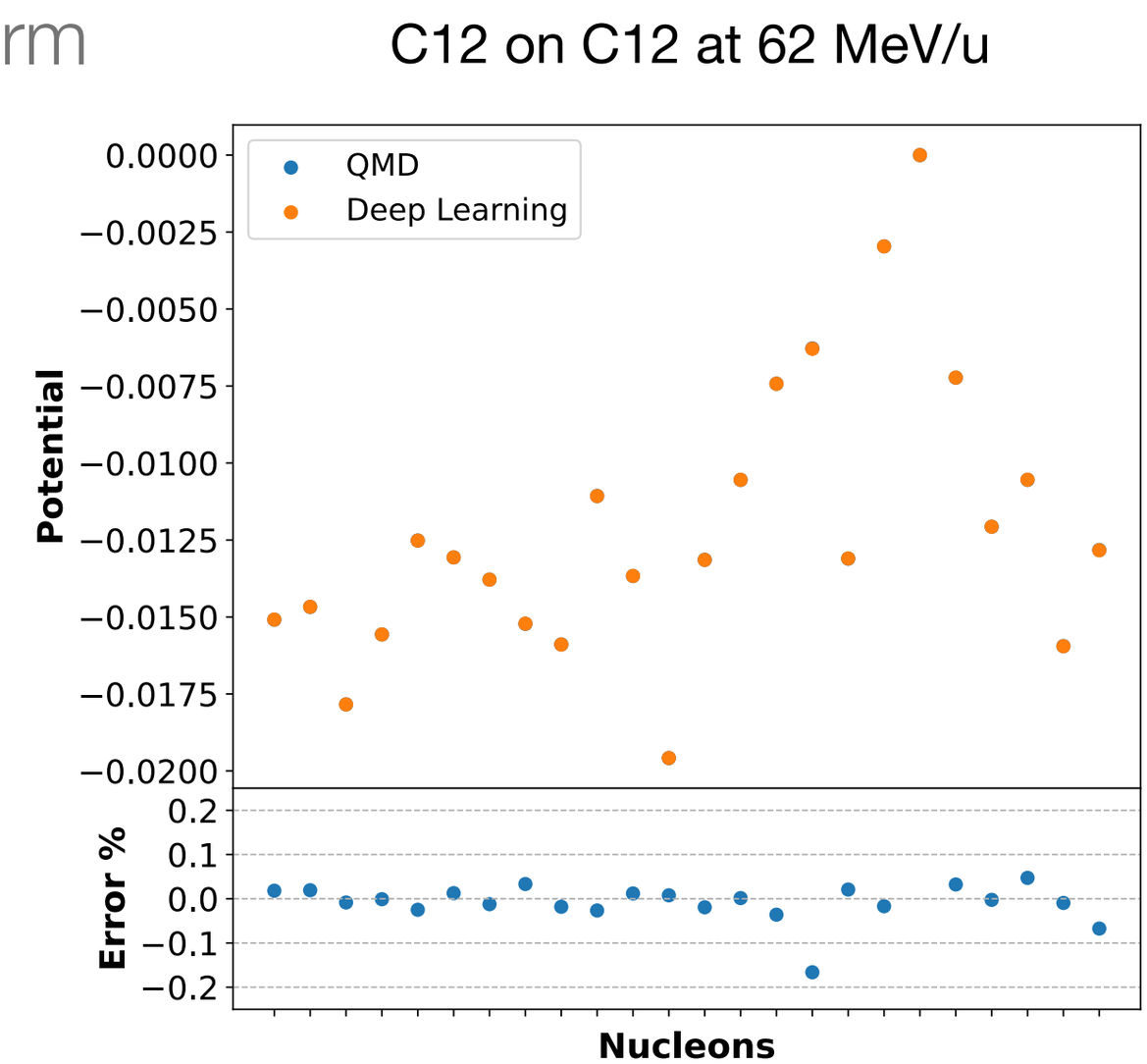
This section lists the most active functions in your application. Optimizing these hotspot functions typically results in improving overall application performance

Function	Module	CPU Time
lapla	run	53.003s
erff	libm.so.6	17.038s
define_two_clouds_rp	run	9.051s
sortx	run	7.450s
powf	libm.so.6	5.184s
[Others]		15.414s

- It also would be possible to improve the mean field approximations

Preliminary results with QMD mean field

- Model:
  5 layers MLP + ReLu + LayerNorm
- Data:
  - 23k stories
  - 10 events
  - 24 particles : ~5 M examples
- Training:
  ~3d training on Nvidia V100



Geant4 implementation

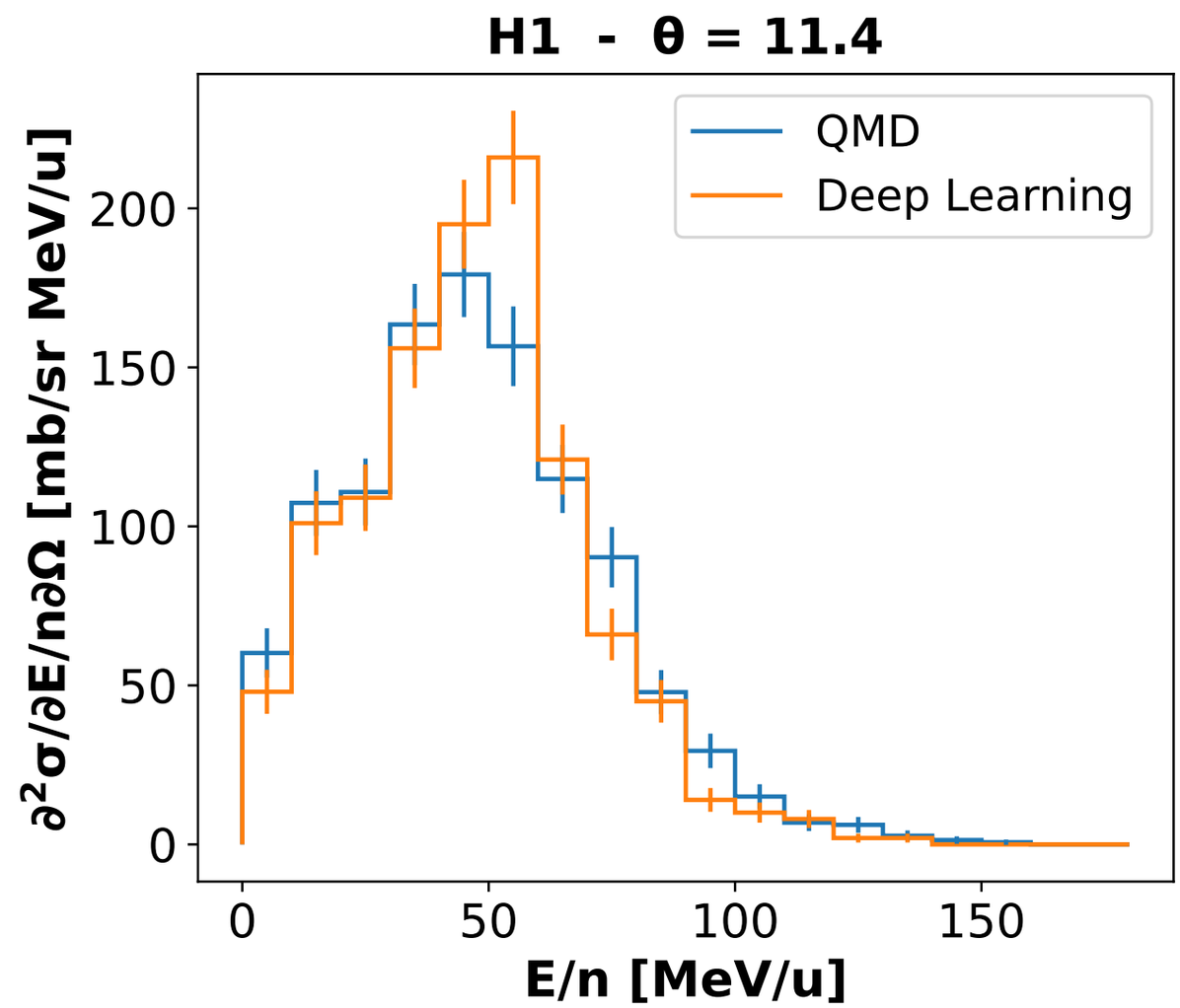
- We exported the DL models from pytorch to ONNX
- Substituting GetPotential() Method in QMD
- Using ONNX C++ API

```
G4double MyQMDMeanField::GetPotential_dl( G4int i )
{
    // ------------------------PREDICT WITH DEEP LEARNING ---------------------------
   return static_cast<G4double>( ONNXInterface::GetInstance()->Generate(i, system)[0]);
    // ------------------------------------------------------------------------------
}
```

- Thread-safe implementation

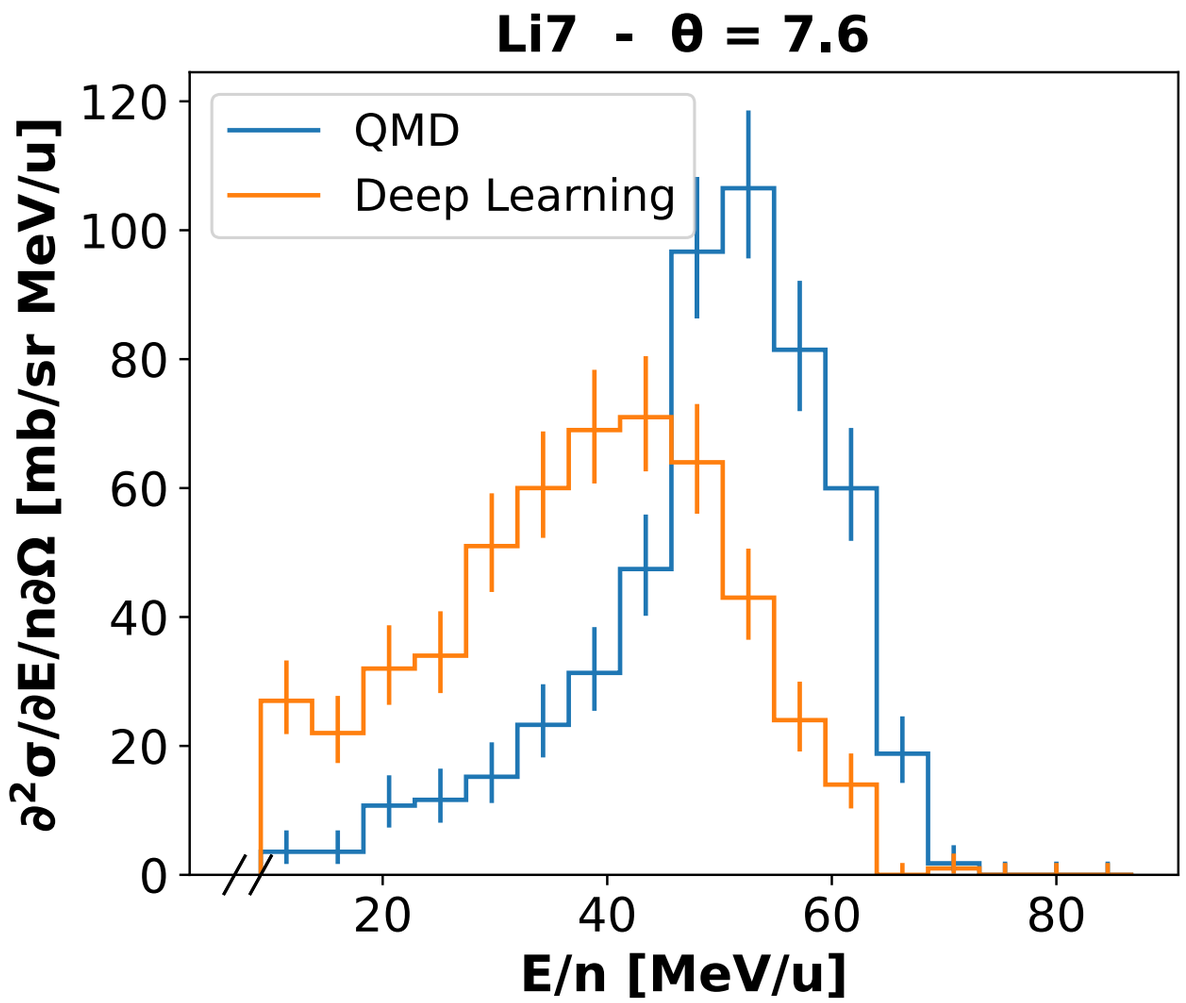
Results with mean field emulation

- C12 on C_nat at 62 MeV/u
- Reasonable accuracy on double differential cross section of lighter fragments



Results with mean field emulation

- On heavier fragments
- Even small errors on the potential propagate badly to the double differential cross sections



Emulating the Hamiltonian derivatives

- $\frac{\partial H}{\partial q}, \frac{\partial H}{\partial p}$

- It is the bottleneck of QMD

Callees	CPU Time: Total ▼
▼ MyQMDReaction::ApplyYourself	100.0%
▼ G4QMDMeanField::DoPropagation	88.7%
▶ G4QMDMeanField::CalGraduate	47.5%
▶ G4QMDMeanField::Cal2BodyQuantities	40.5%

- $$\frac{\partial H}{\partial q, p} \approx \sum A_{ij} + \sum_{\alpha^{(k)}} \left(\sum B_{ij}^{(k)} \right)^{\alpha^{(k)}}$$

- Hyper-parameter optimisation on the number of terms K

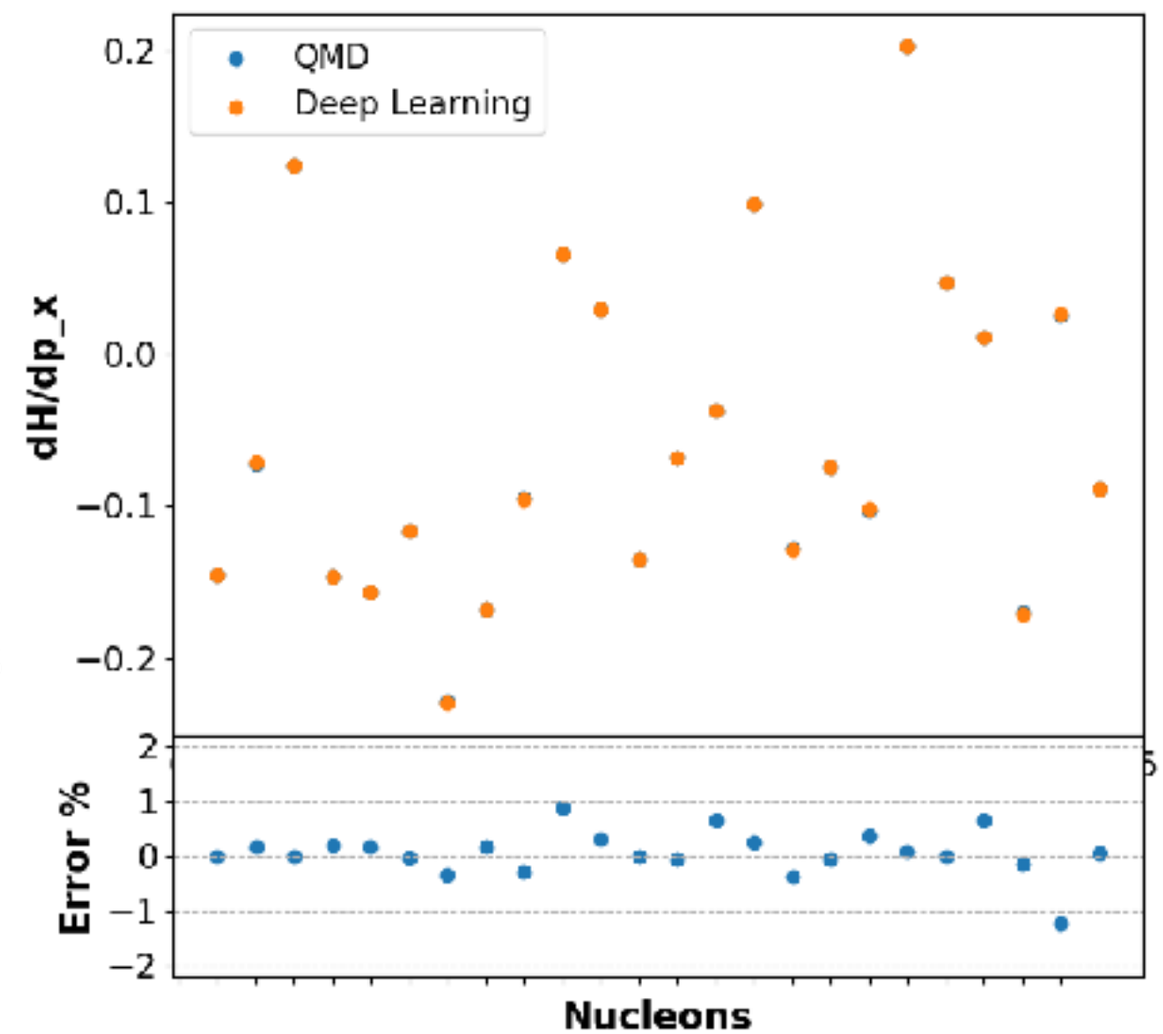
Preliminary results with QMD H derivatives

- Model:
  2 $\alpha^{(k)}$ terms + 5 layers MLP + ReLu + LayerNorm

- Data:
  - 12k stories
  - 1 event
  - 24 particles : ~300 k examples

- Training:
  ~3h training on Nvidia V100

C12 on C12 at 62 MeV/u



Geant4 implementation

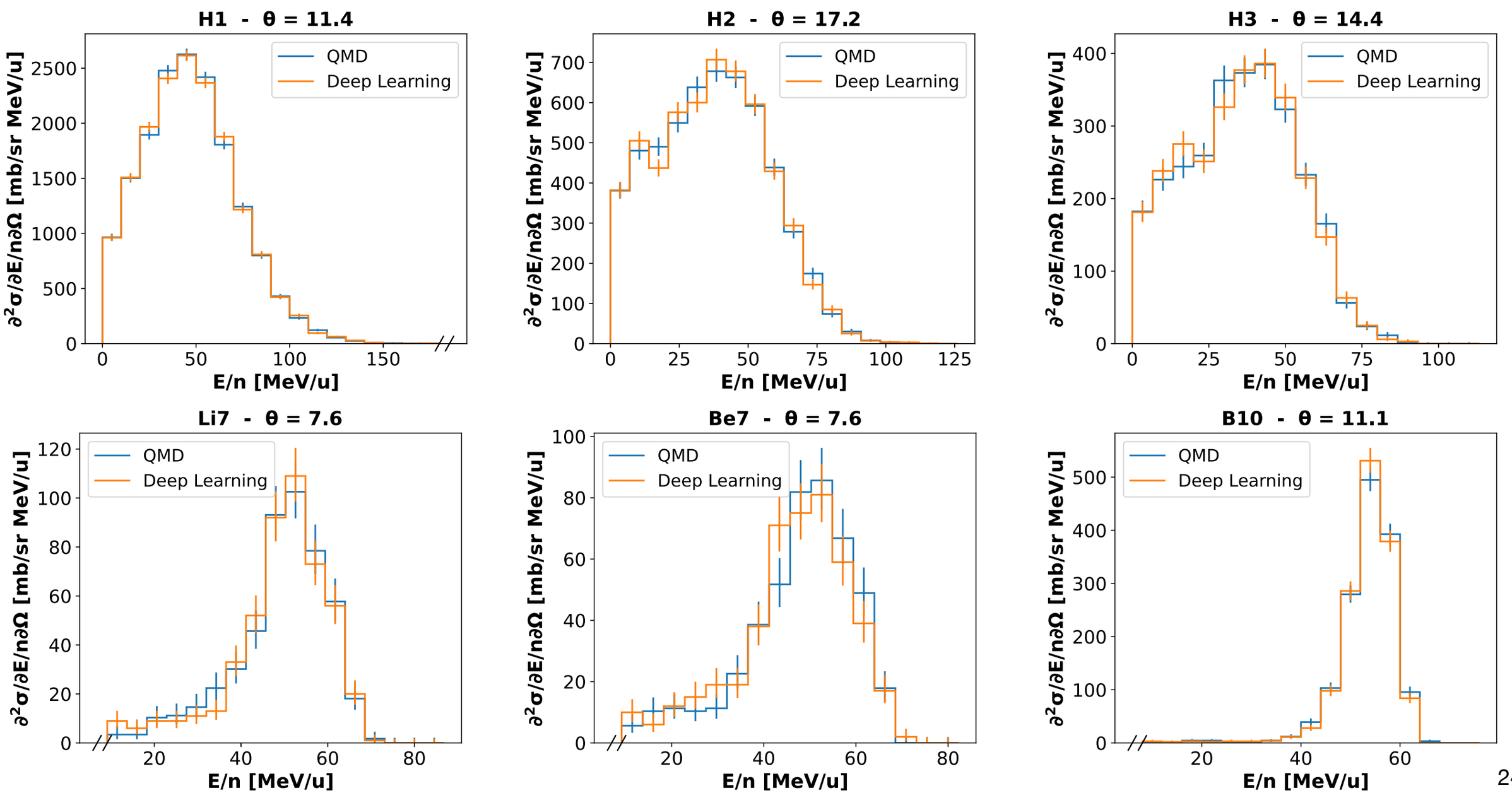
- We exported the DL models from pytorch to ONNX
- Substituting GetPotential() Method in QMD
- Using ONNX C++ API

```
void MyQMDMeanField::CalGraduate_dl()
{
   ffr.resize( system->GetTotalNumberOfParticipant() );
   ffp.resize( system->GetTotalNumberOfParticipant() );

  // ----------------- PREDICT WITH DEEP LEARNING -----------------
  auto gradients = ( ONNXInterface::GetInstance()->Generate(system));
  ffr = gradients[0];
  ffp = gradients[1];

}
```

- Thread-safe implementation

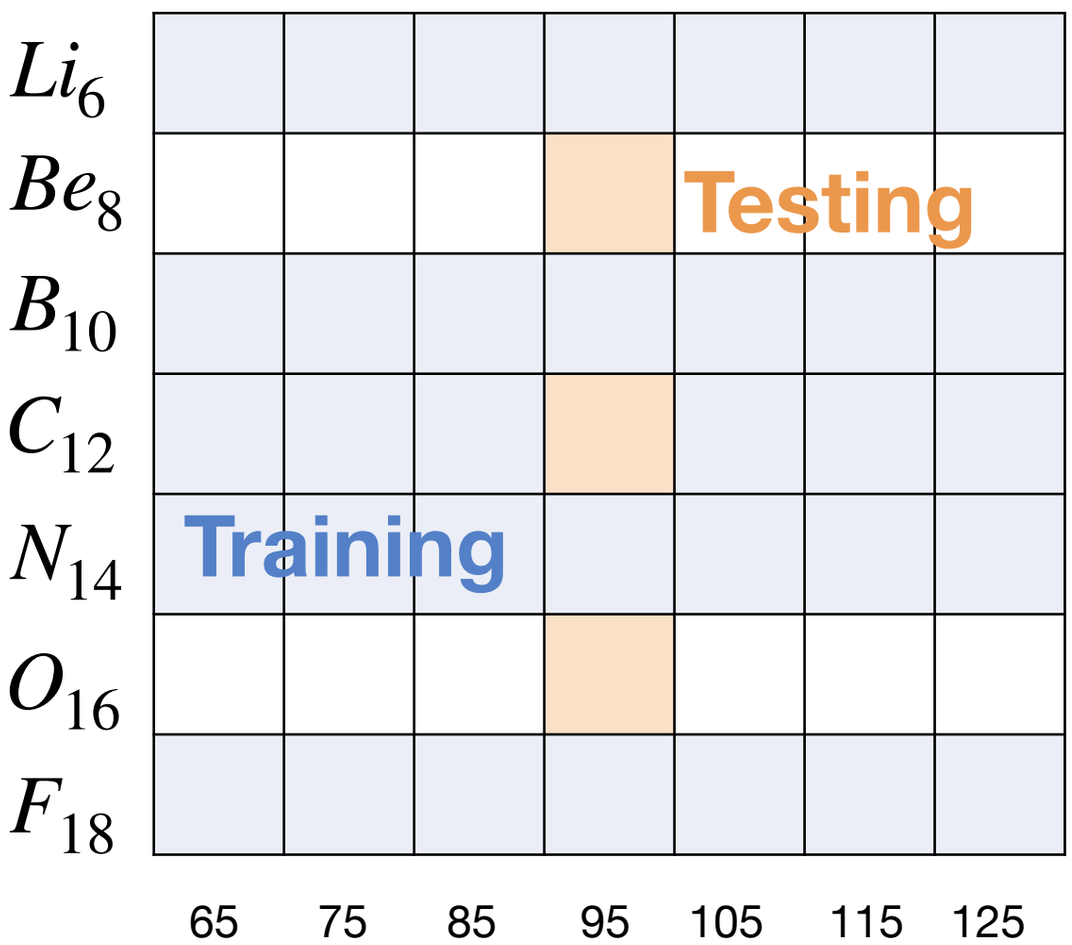
Results with H derivatives emulation

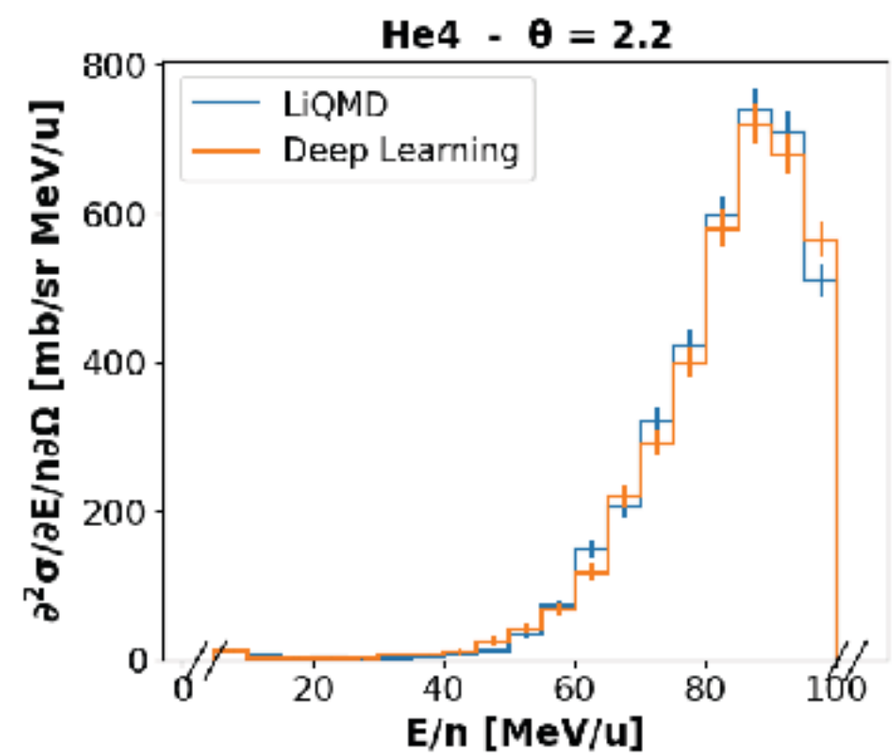
- Excellent agreement also on large fragments



Extending the training

- Training done on a subset of ions, with relatively few example each (~1k runs)





Next steps

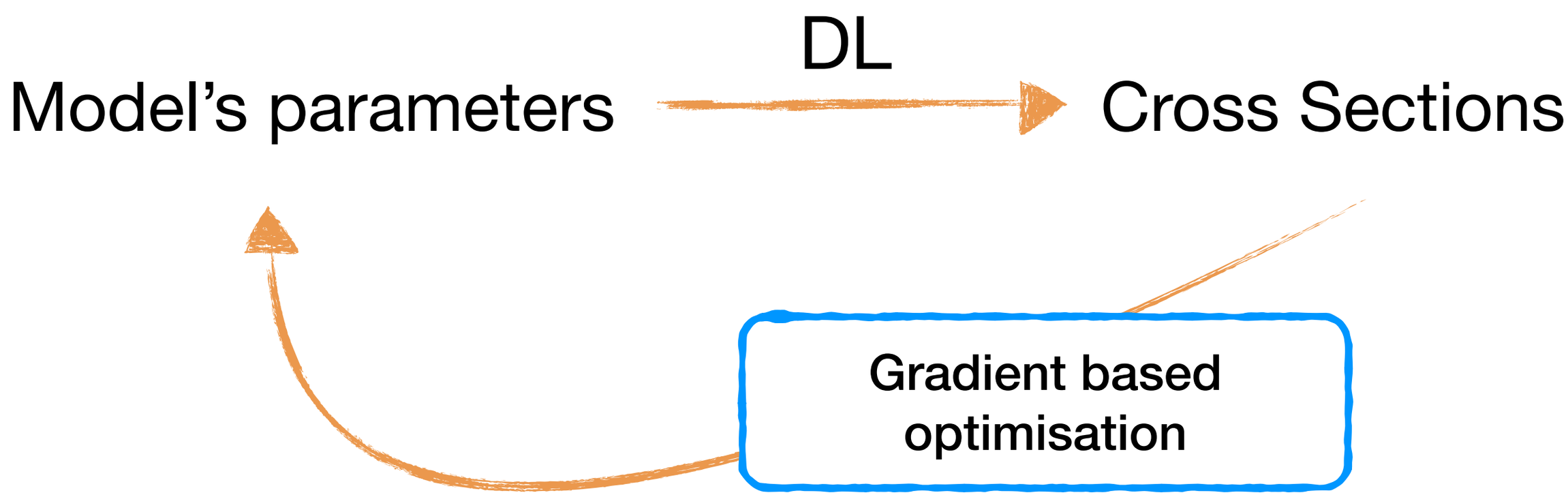
- Current implementation (on CPU) is slower than the original QMD

  "NVIDIA TensorRT-based applications perform up to 36X faster than CPU-only platforms during inference"

- Using NVIDIA TensorRT it could be possible a 4x-7x speed-up

Next steps

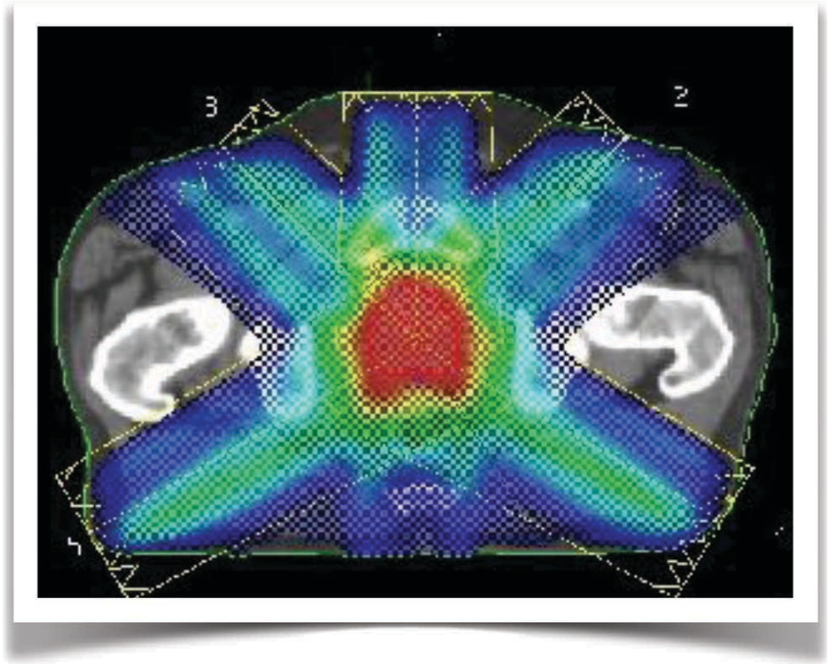
- Develop a fully differentiable pipeline to optimise the model free parameters

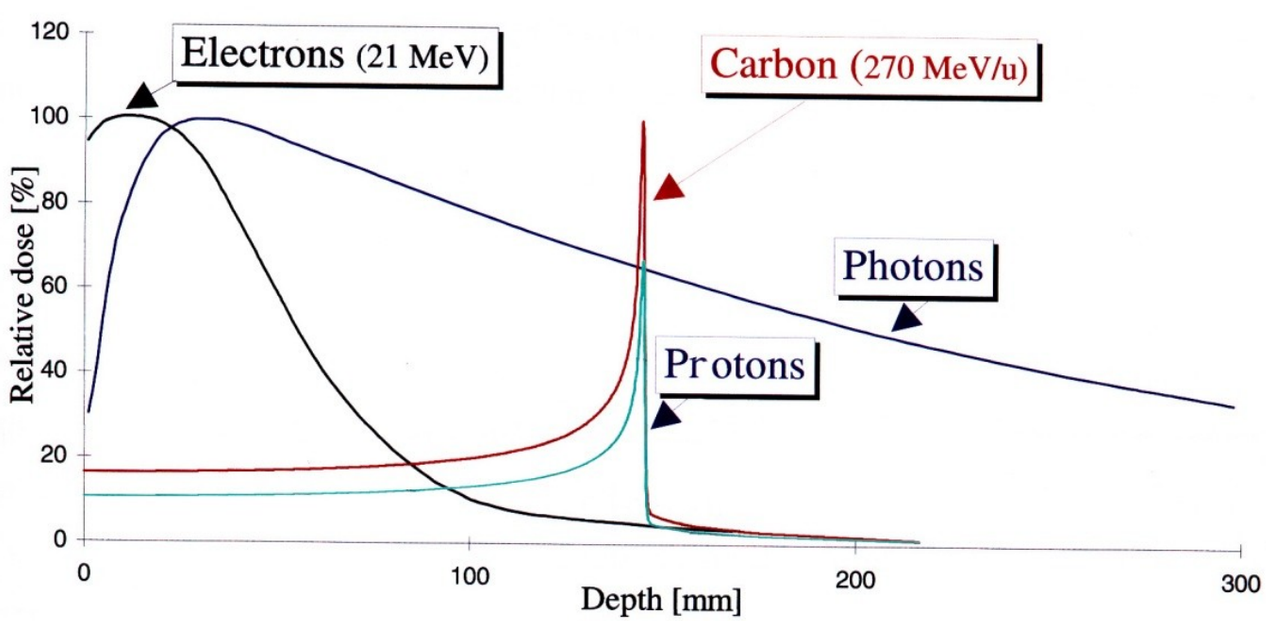


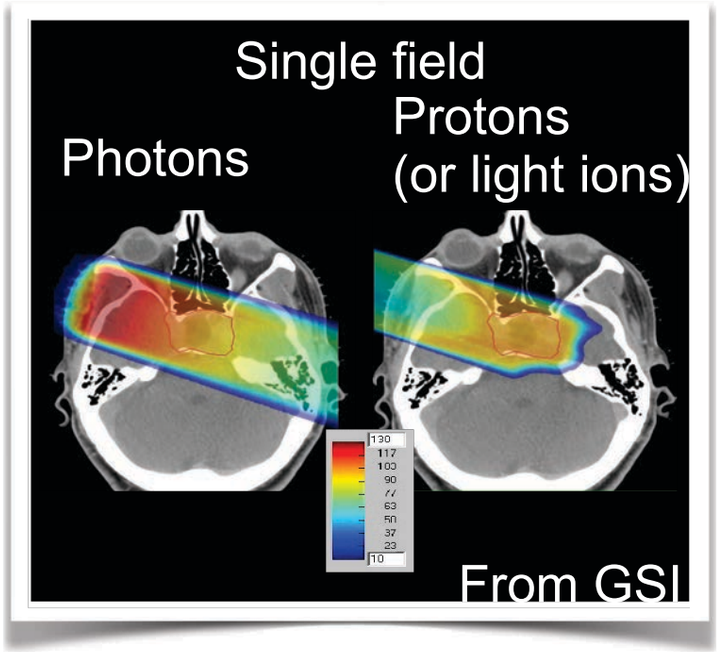
Emulating de-excitation model

Treatment plan optimisation

- The goal is to maximise tumour dose while minimising exposure to healthy tissues

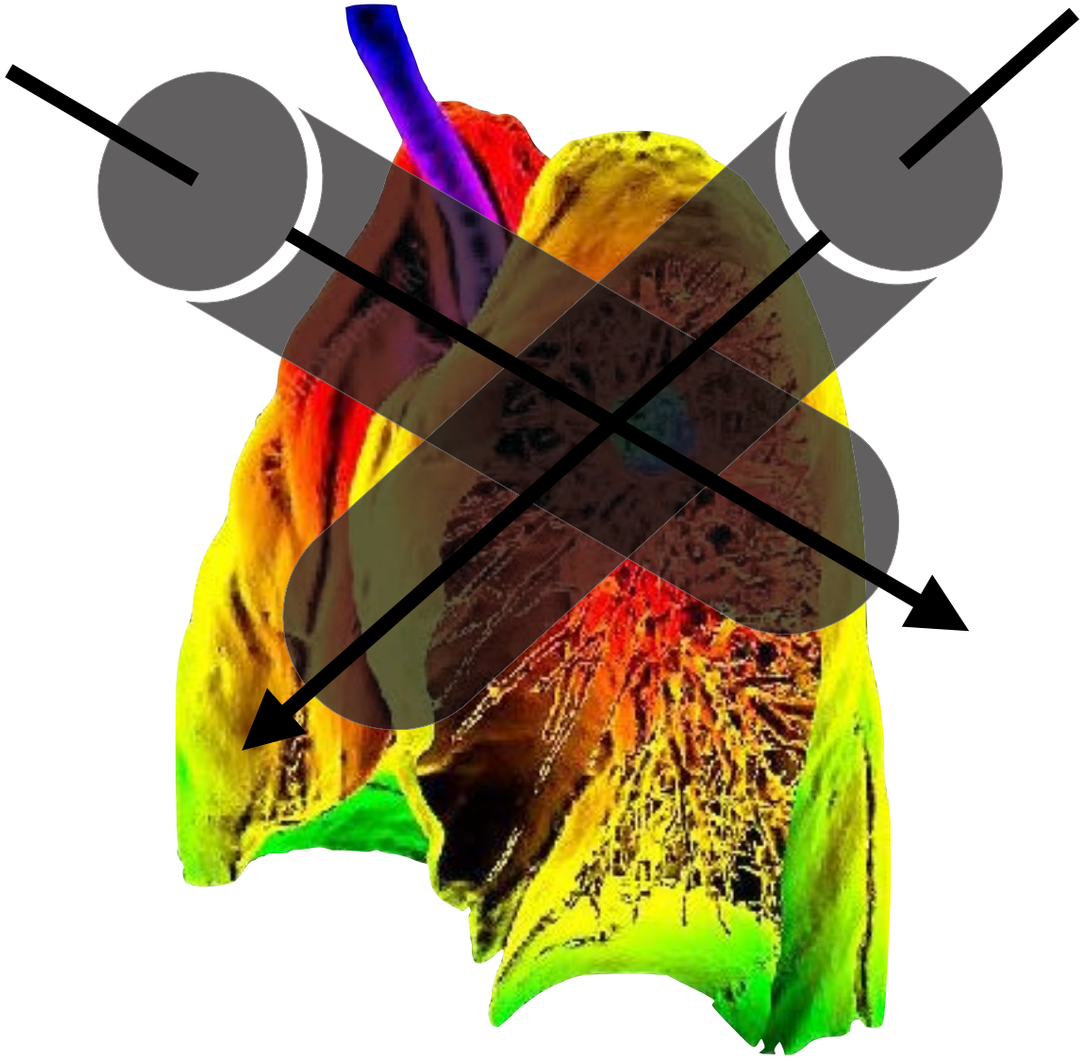




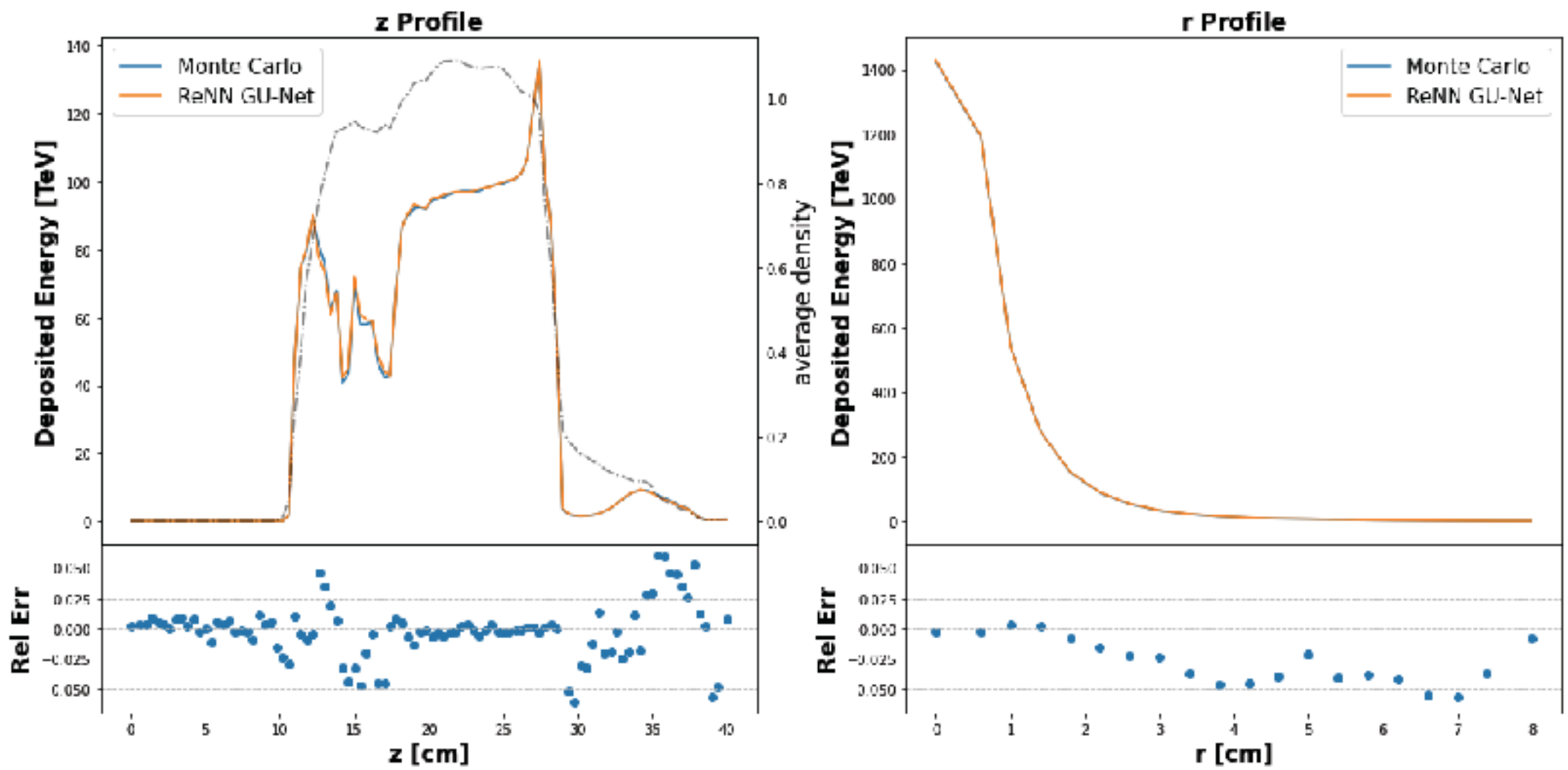


Cylindrical scorers

- We used cylinders around the beam
- Two main advantages:
  - Reduce complexity without loss of generalisation: the cylinder follows the beam
  - More resolution near the beam-line

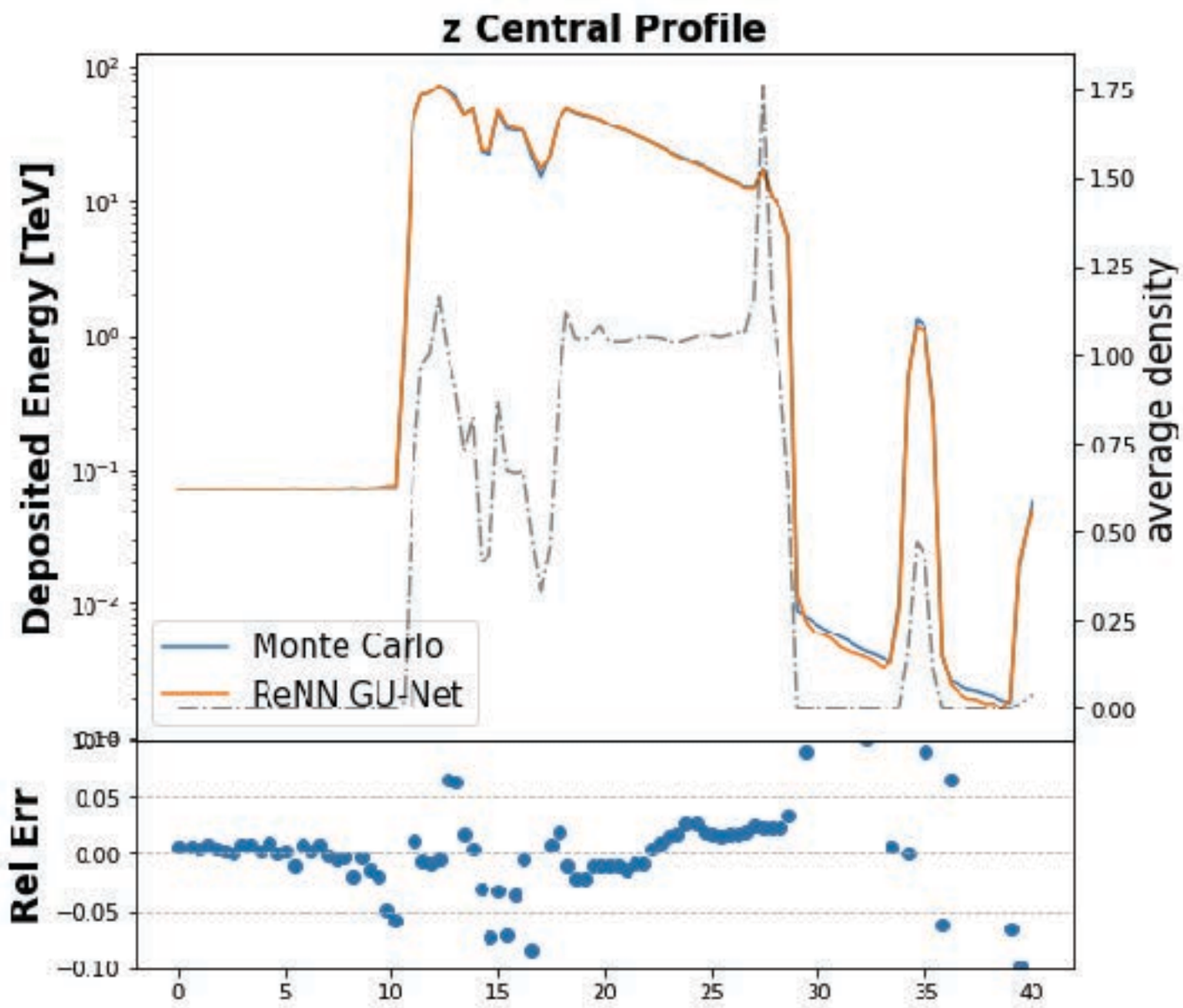


Integrated Energy Profiles on CT scan



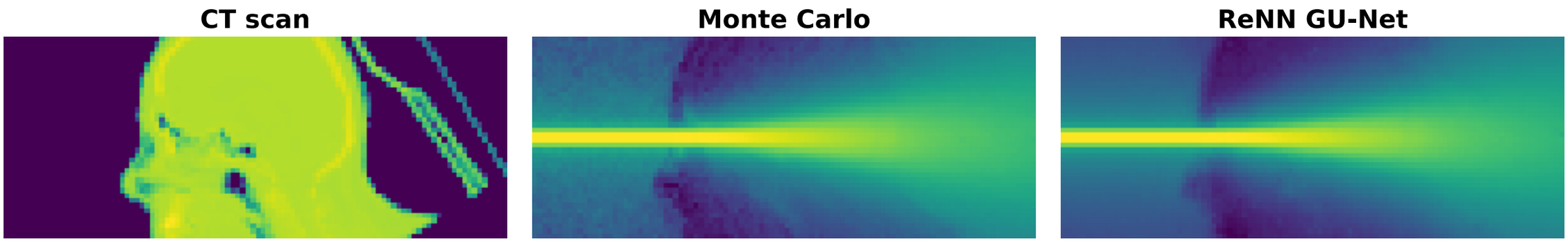
- Excellent agreement with MC

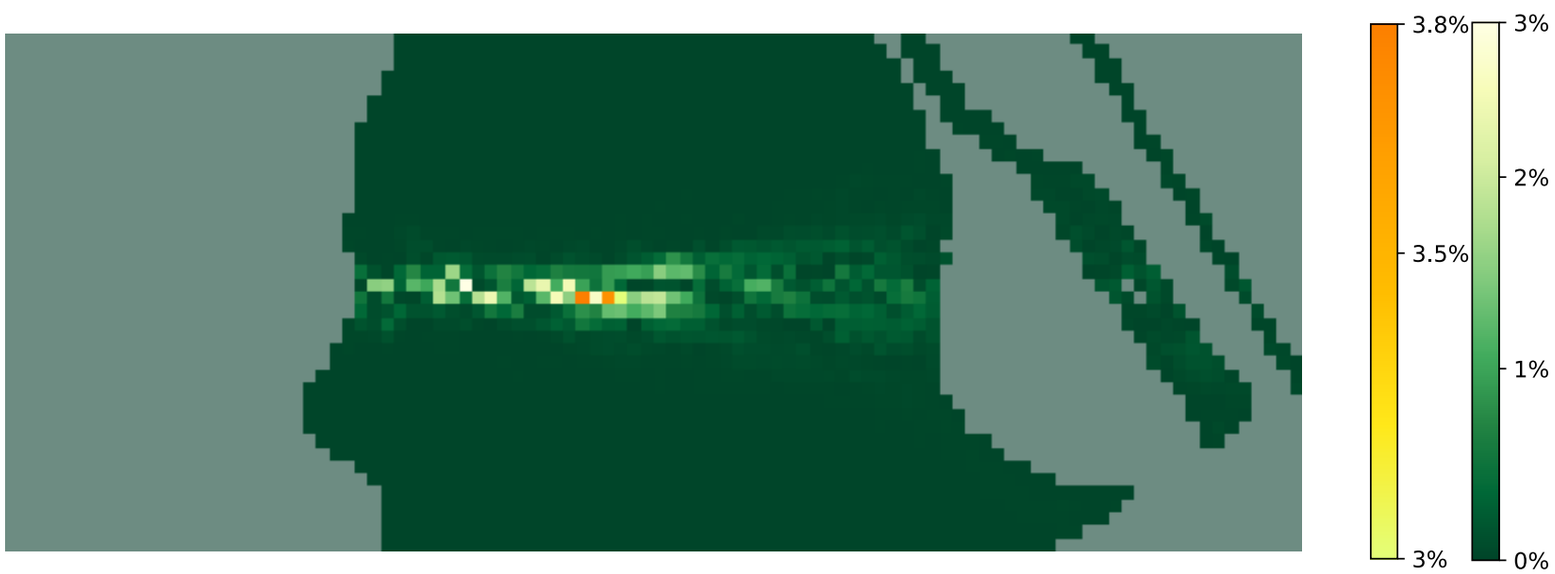
Precise also locally… …and fast!



	Generation time
Deep Learning	~ 0.01 - 0.1 s
Monte Carlo (48 cores) 1M primaries	~ 15 - 20 min

Dose prediction



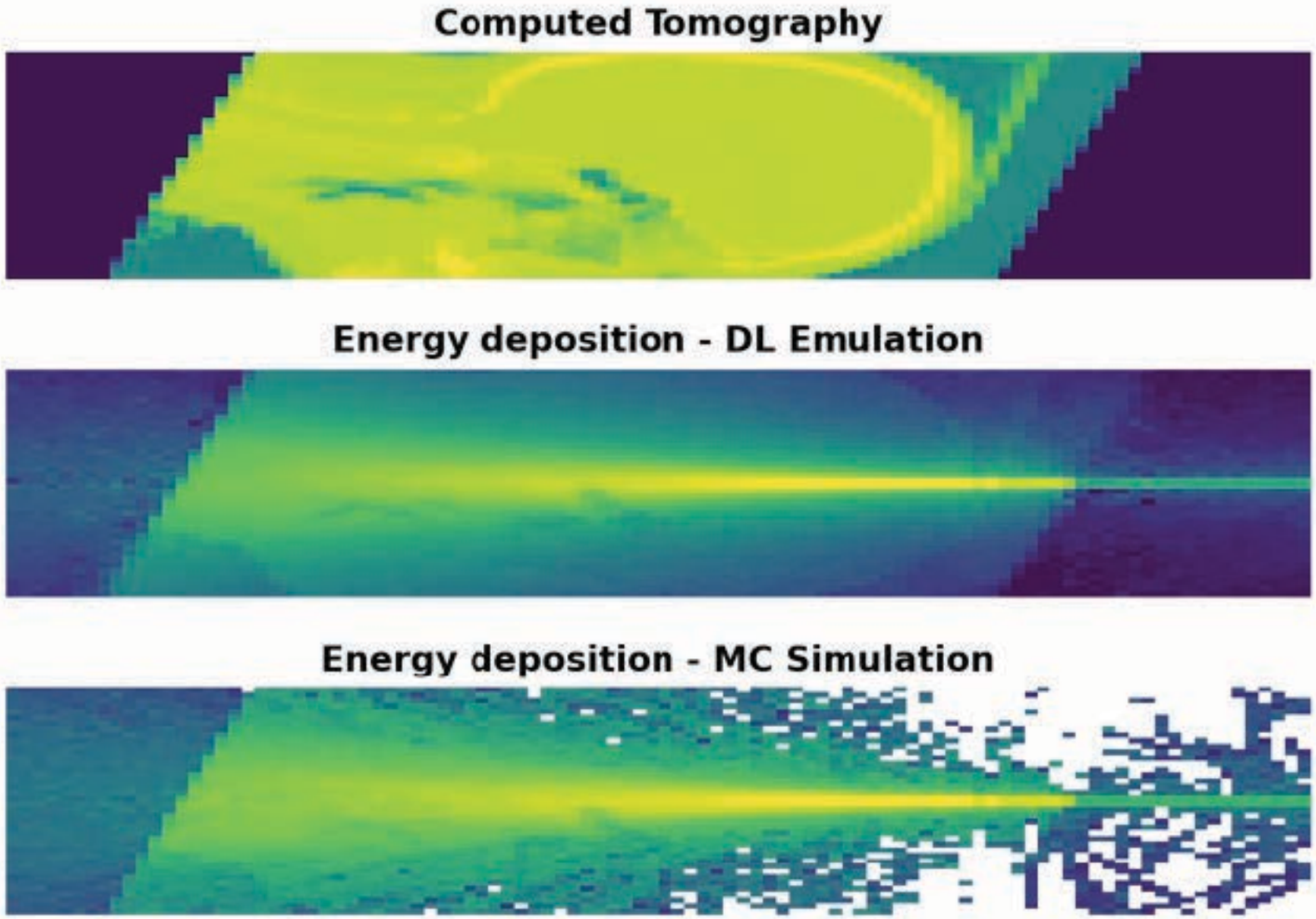


99.8 ± 0.2 %

of the voxels have

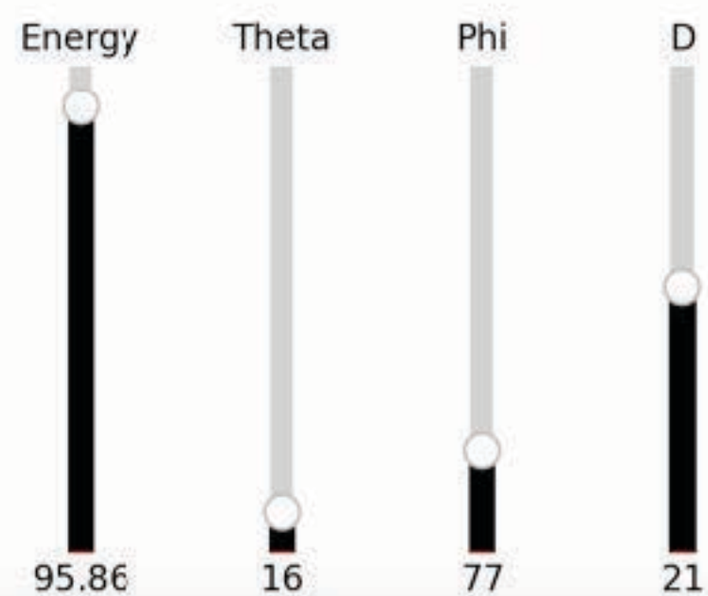
$$\delta = \frac{|D_{MC} - D_{DL}|}{max(D_{MC})} < 3\,\%$$

Generating dose distributions from beam's energy and density map

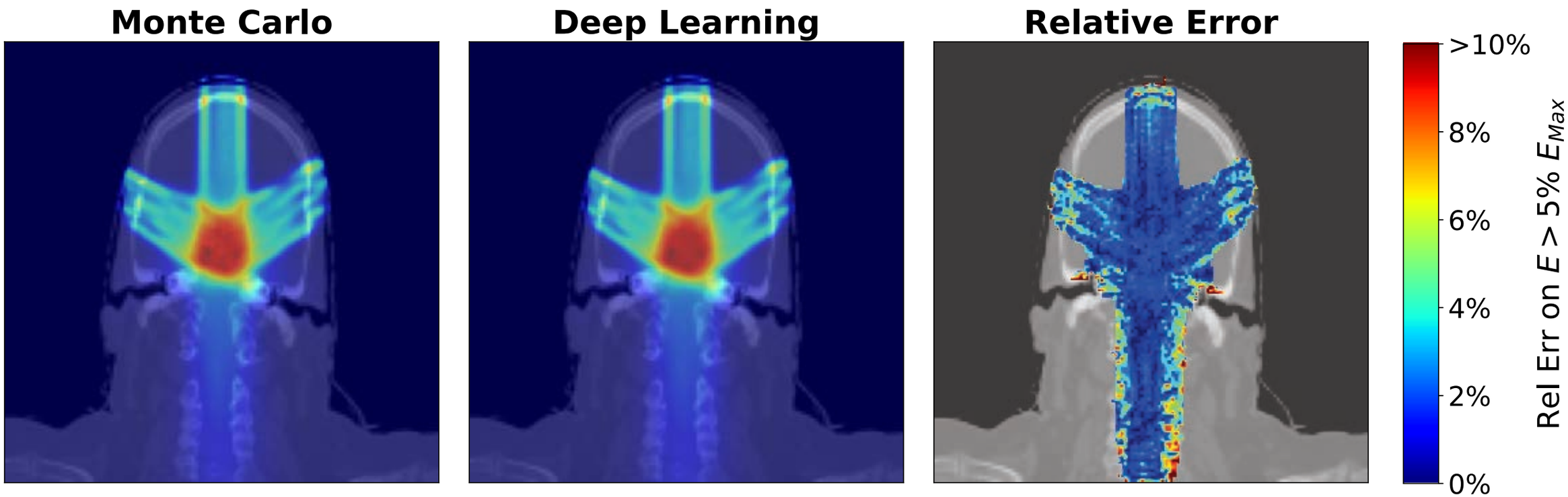


	Energy	Dose
γ-index(1%) :	90.70 %	99.01 %
γ-index(3%) :	98.49 %	99.60 %
γ-index(5%) :	99.63 %	99.81 %

Next Datapoint



Full Treatment Plan



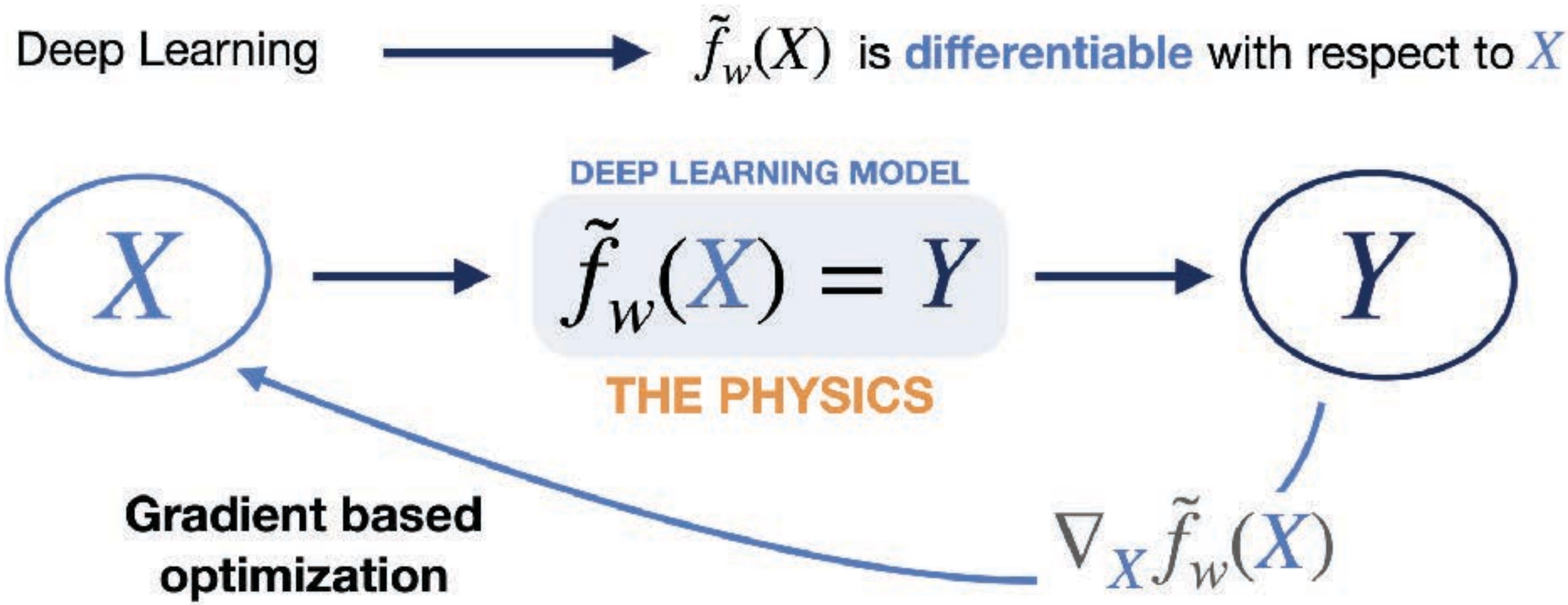
350 pencil beams in 6 seconds

On a single Nvidia Tesla V100 GPU

Local γ-index 3%/3mm: 98.4%

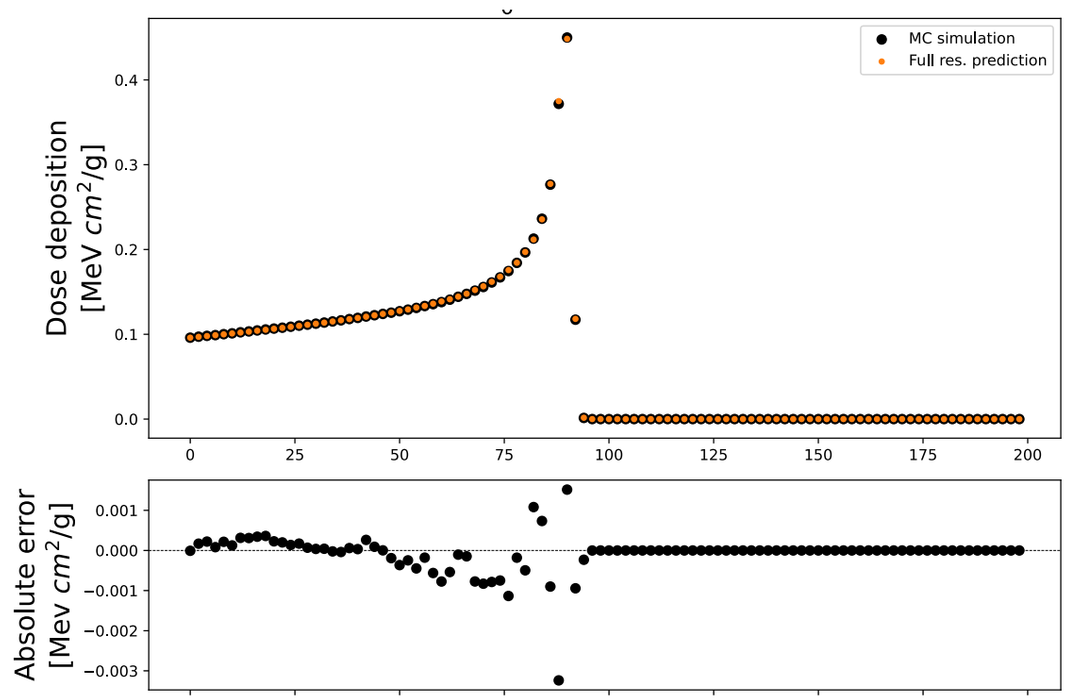
Gradient based optimisation

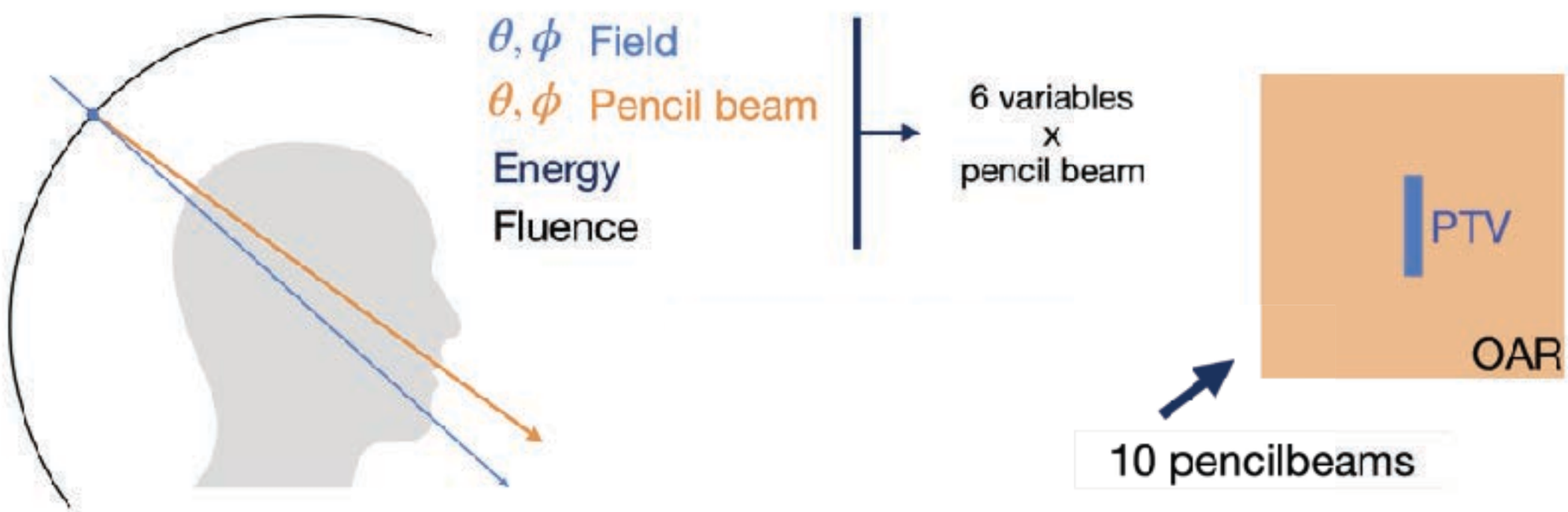
- It is possible to optimise the treatment plan with a differentiable programming approach



Test on proton therapy optimisation

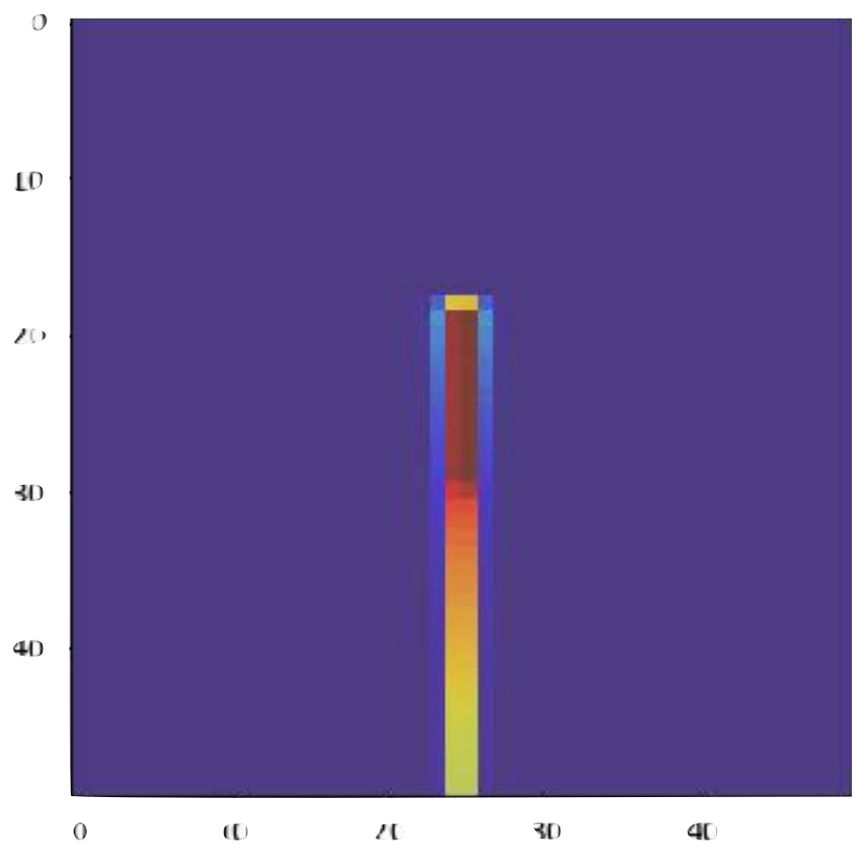
- Optimising:
  - Field Orientation
  - Pencil-beam Energy
  - Pencil-beam Fluence

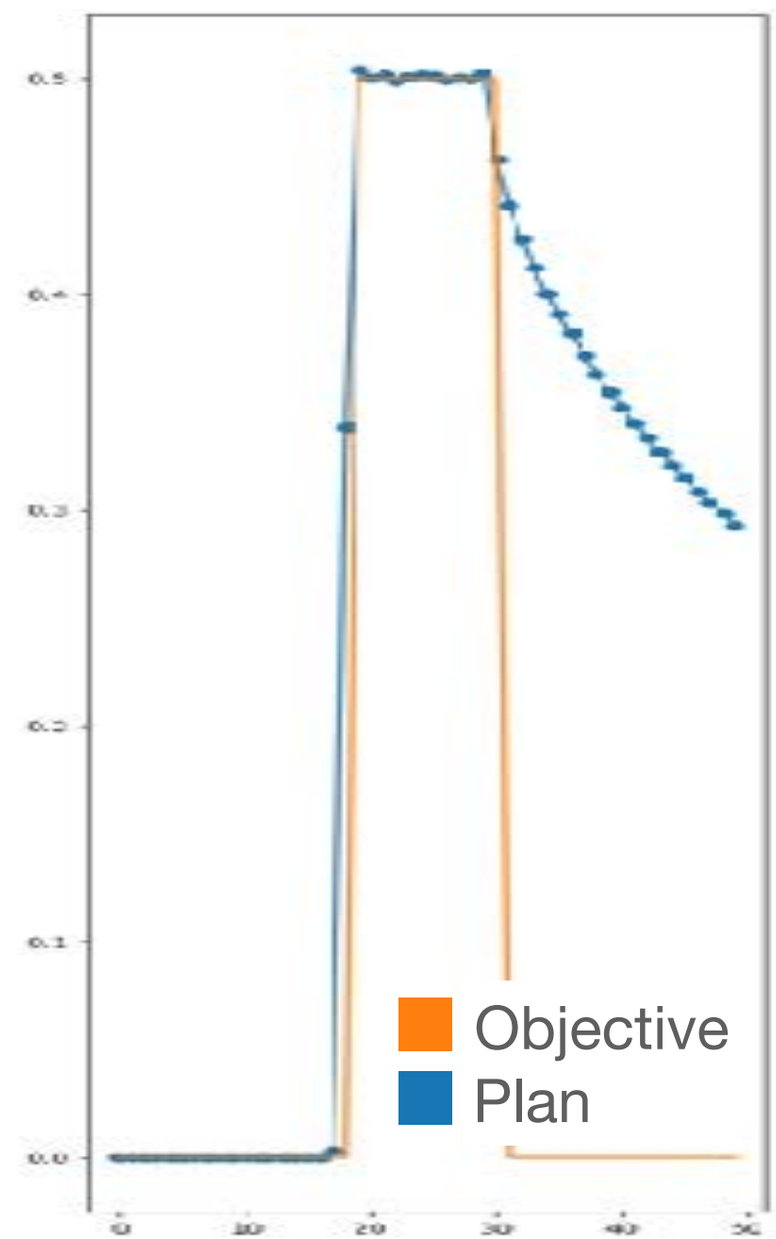




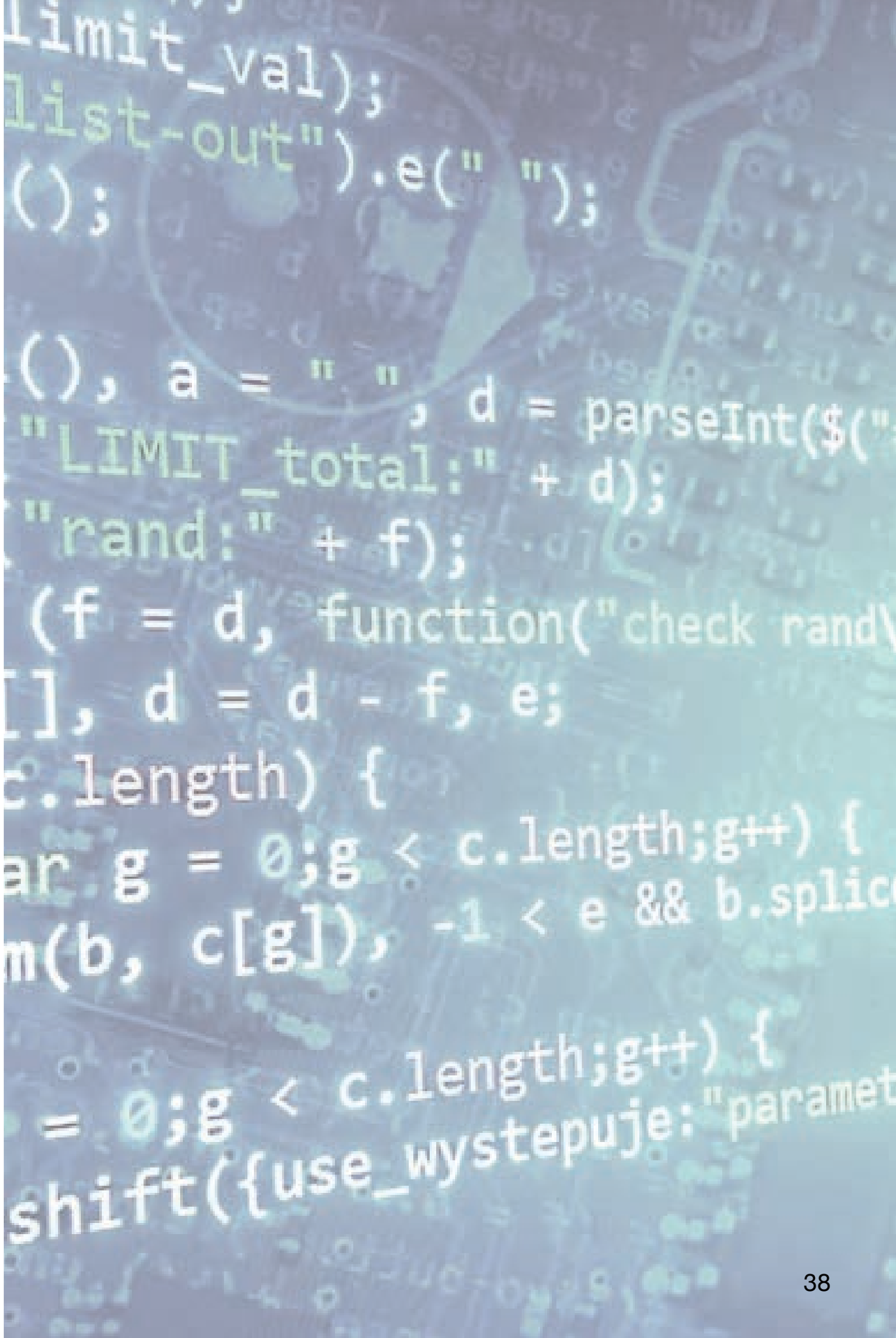
Optimisation results

- A gradient descent algorithm found the configuration to obtain a Spread Out Bragg Peak
- Optimising also the entrance direction





Technical details



Virtualisation!

- If you create a virtual machine with Geant4 and ONNX compiled in the way needed by our code it would work everywhere
- It’s heavy and slow!
- Containers overcome all the shortcomings of Virtual Machines

Virtualisation!

- Containers don’t require the installation of a separate guest operating system.
- They directly run and use the host operating system
- Containers only need the dependent file system and binaries for their functioning
- lightweight than Virtual Machines

What is a container?

- Containers (such as Docker) are
  - a standard for cloud computing and clusters
  - a good way to run an application in the same environment on different machines
  - a fast way to distribute code for multiple architectures
  - integrated in all the CI/CD platforms
  - light and efficient

Security issue and Apptainer

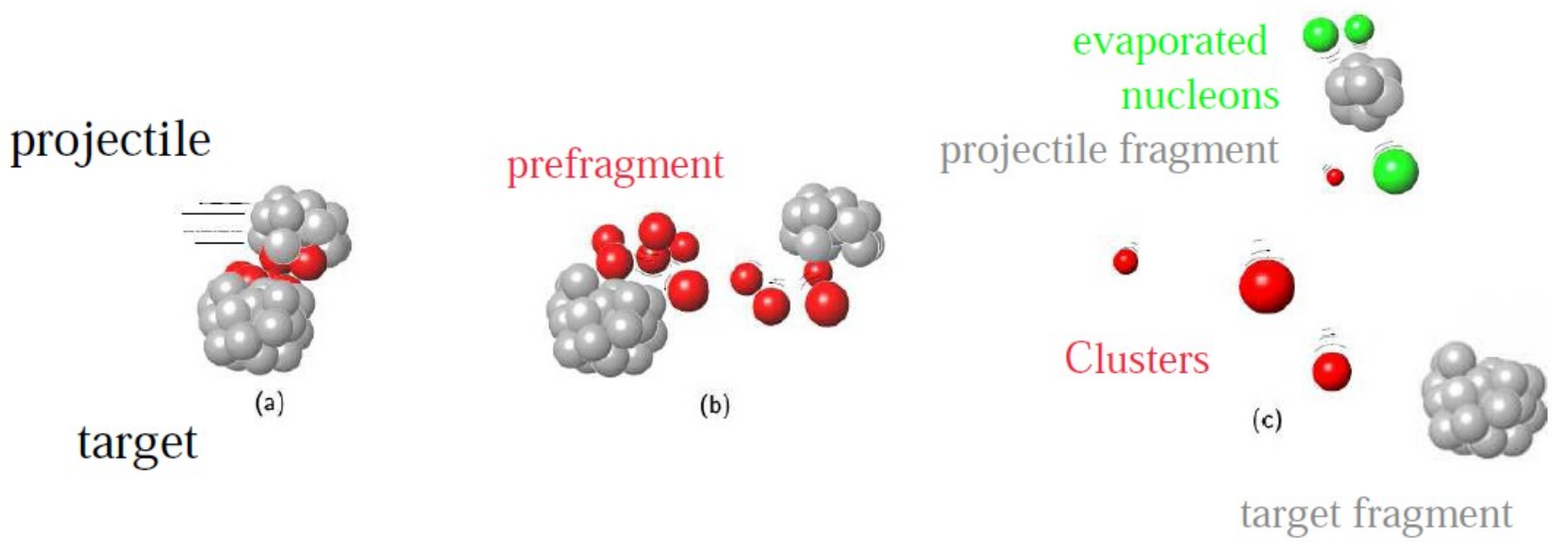
- Depending on how the Docker demon is installed, you could be root of the container (and if the host is Linux on the volumes mounted)

- Apptainer (formerly Singularity) is a container system (compatible with Docker images) which doesn't have this security issue

- https://apptainer.org/

- Largely available on scientific computing clusters
  eg: https://confluence.infn.it/display/TD/Singularity+in+batch+jobs

Nuclear interactions



- Hadronic interactions are simulated in two different stages:
  - The first one describes the interaction from the collision until the excited nuclear species produced in the collision are in equilibrium
  - The second one, such as the Fermi break-up, models the emission of such excited, but equilibrated, nuclei

Nuclear interactions

- **The entrance channel model characteristics have a larger effect** on particles and fragments production as compared to the choice of the exit channel

[Conclusions from: J. Dudouet et al. Phys. Rev. C, vol. 89, no. 5, p. 054616, May 2014]

- Hadronic interactions are simulated in two different stages:
  - The first one describes the interaction from the collision until the excited nuclear species produced in the collision are in equilibrium
  - BLOB is one of the entrance channel models

Geant4 interface to SMF and BLOB

- Dummy G4-model, loads the output from SMF/BLOB
- Sample the final state
  - Fragments mass and charge
  - Gas particles emitted
- Applies Geant4 de-excitation to excited fragments