Update on the emulation of nuclear interaction models with Deep Learning

G4 Collaboration Meeting 2024 - Four Points Catania

L. Arsini[1,2], S. Burrello[3], B. Caccia[4], A. Ciardiello[1], M. Colonna[3], S. Giagu[1,2], C. Mancini Terracciano[1,2]

[1]Department of Physics,Sapienza University of Rome, Rome, Italy. [2]INFN, Section of Rome, Rome, Italy. [4]INFN, Section of LNS
[3]Istituto Superiore di Sanità, Rome, Italy.

09/10/2024

Problems in Geant4 below 100 MeV/u

No dedicated model to nuclear interaction **below 100 MeV/u** in Geant4

Many papers showed discrepancies:

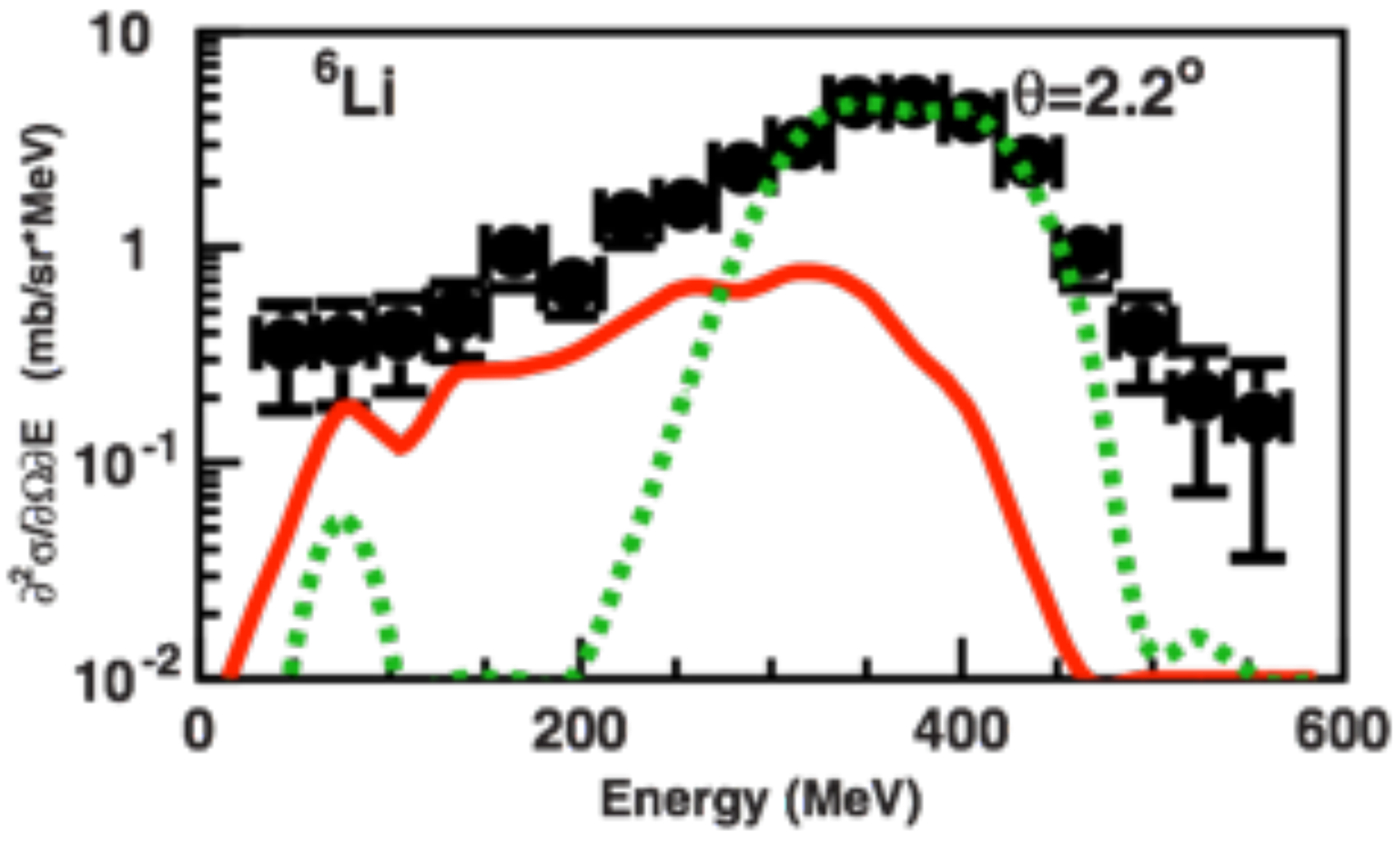
Braunn et al. : one order of magnitude in 12C fragmentation at 95 MeV/u on thick PMMA target

De Napoli et al. : angular distribution of the secondaries emitted in the interaction of 62 MeV/u 12C on thin carbon target

Dudouet et al. : similar results with a 95 MeV/u 12C beam on H, C, O, Al and Ti targets

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

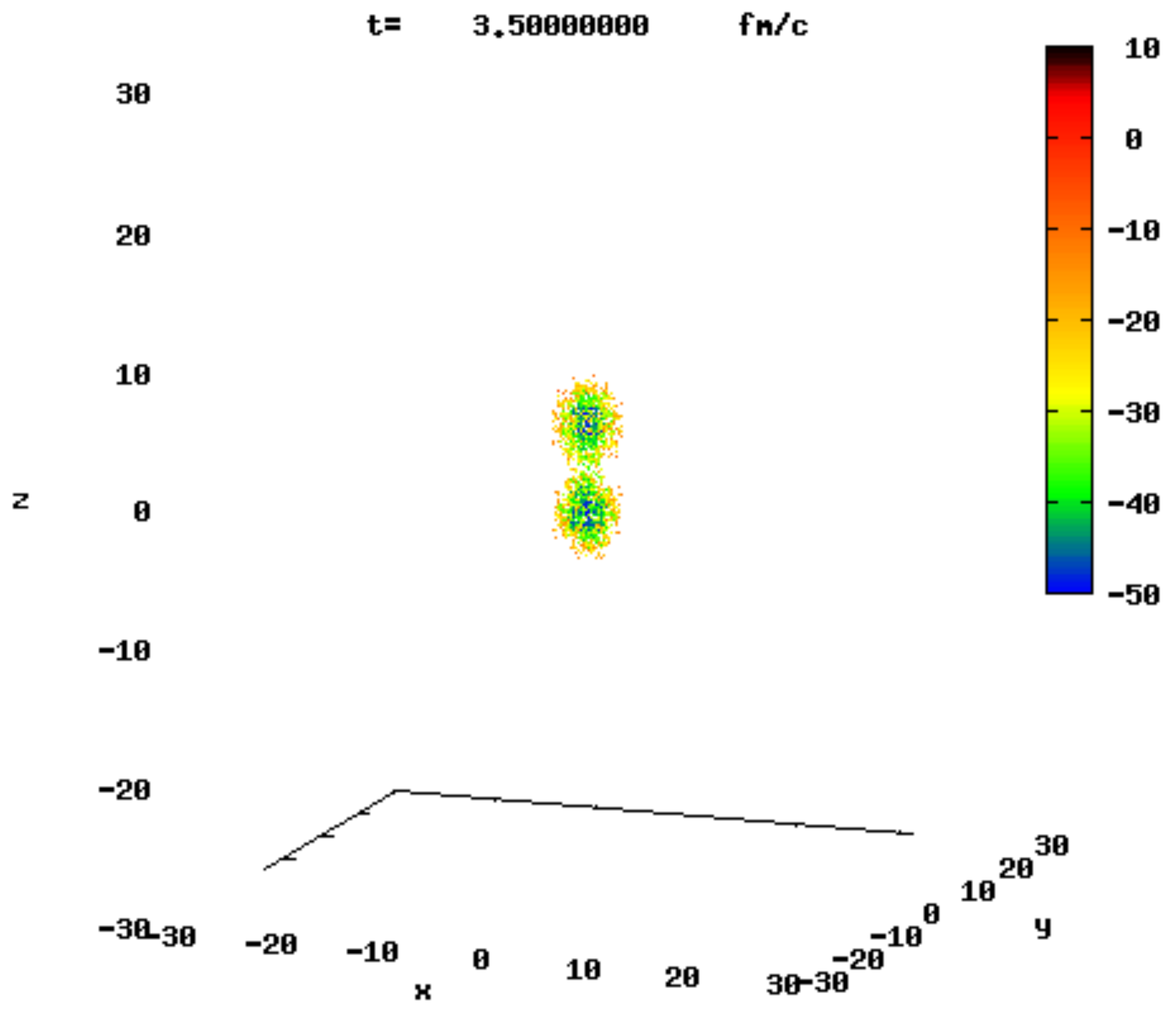
[Plot from De Napoli et al. Phys. Med. Biol., vol. 57, no. 22, pp. 7651–7671, Nov. 2012]



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

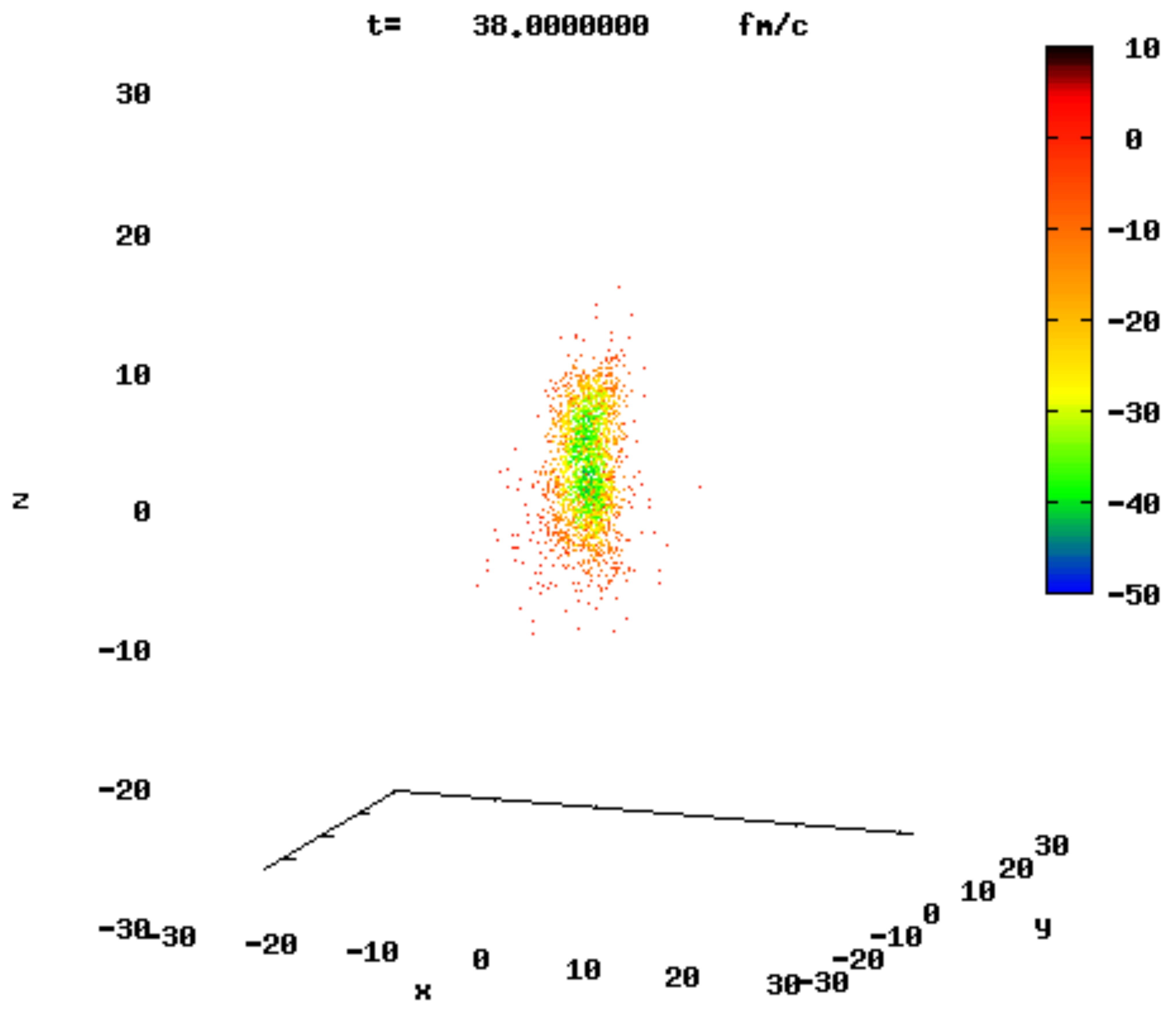
BLOB (Boltzmann-Lagevein One Body)

- Test-particle approach
- Self-consistent **mean field** + collisions
- Probability to find a nucleon in the phase space



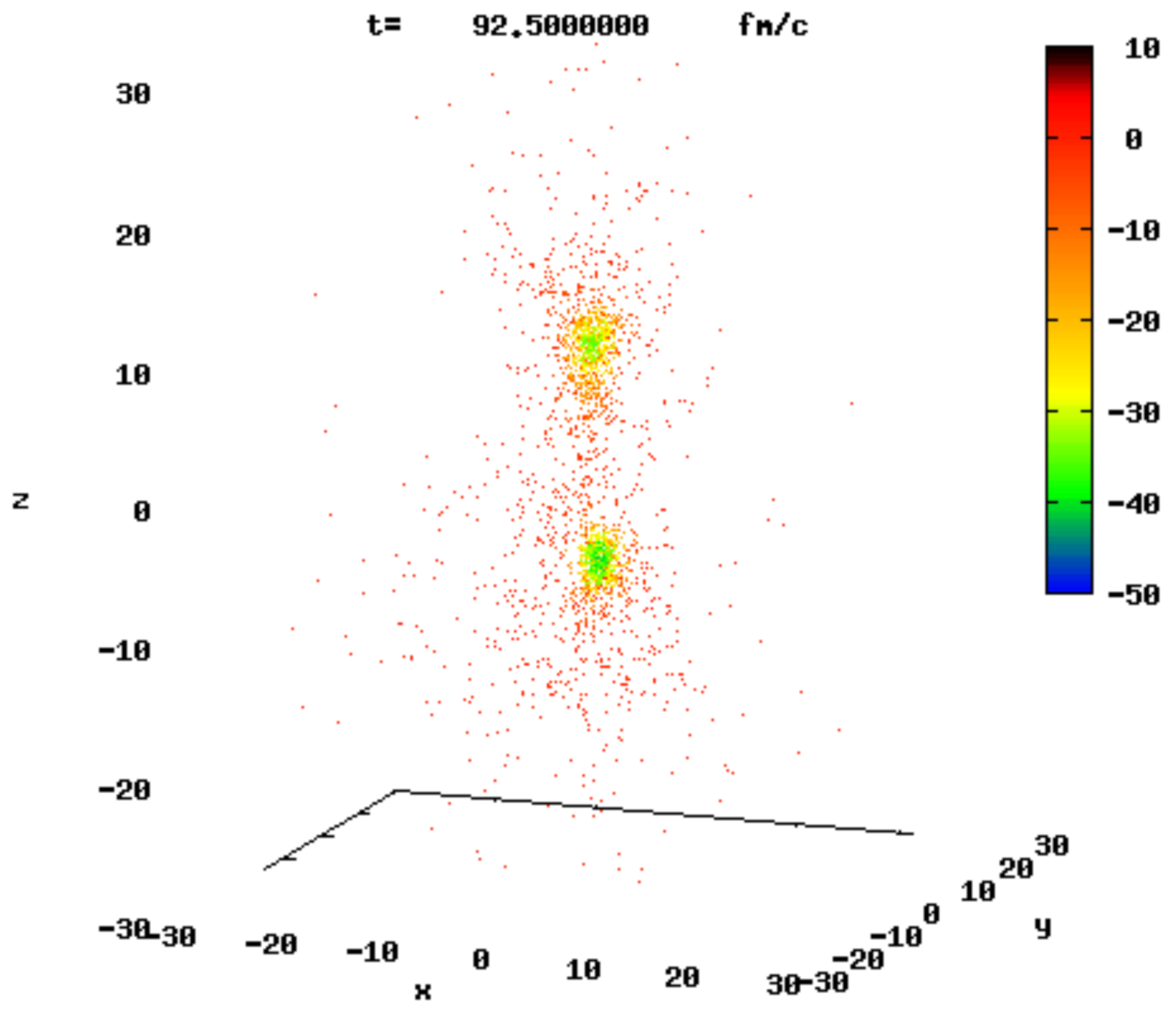
BLOB (Boltzmann-Lagevein One Body)

- Test-particle approach
- Self-consistent **mean field** + collisions
- Probability to find a nucleon in the phase space



BLOB (Boltzmann-Lagevein One Body)

- Test-particle approach
- Self-consistent **mean field** + collisions
- Probability to find a nucleon in the phase space



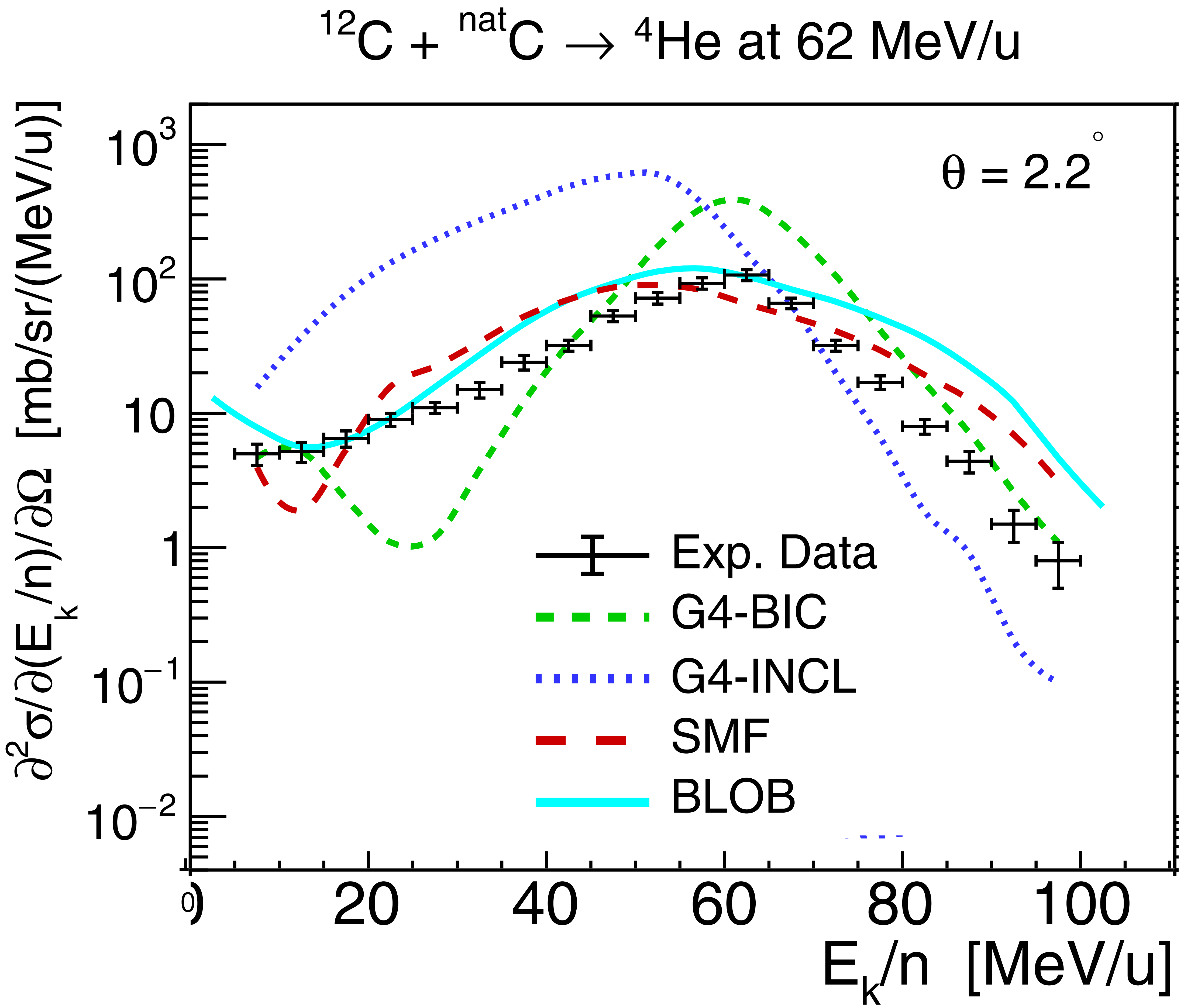
BLOB (Boltzmann-Lagevein One Body)

We interfaced BLOB with Geant4 and its de-excitation model

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

Accurate **Slow**

Order of minutes per interaction!



Deep Learning to accelerate NIMs

Why?

- Approximating complex functions with **Neural Networks**
- Leveraging **GPU acceleration** for ultra-fast execution

How?

- Building **Physics-inspired** architectures → Physics under control / **Explainability**

Starting from a proof-of-concept study on **QMD**

What to emulate?

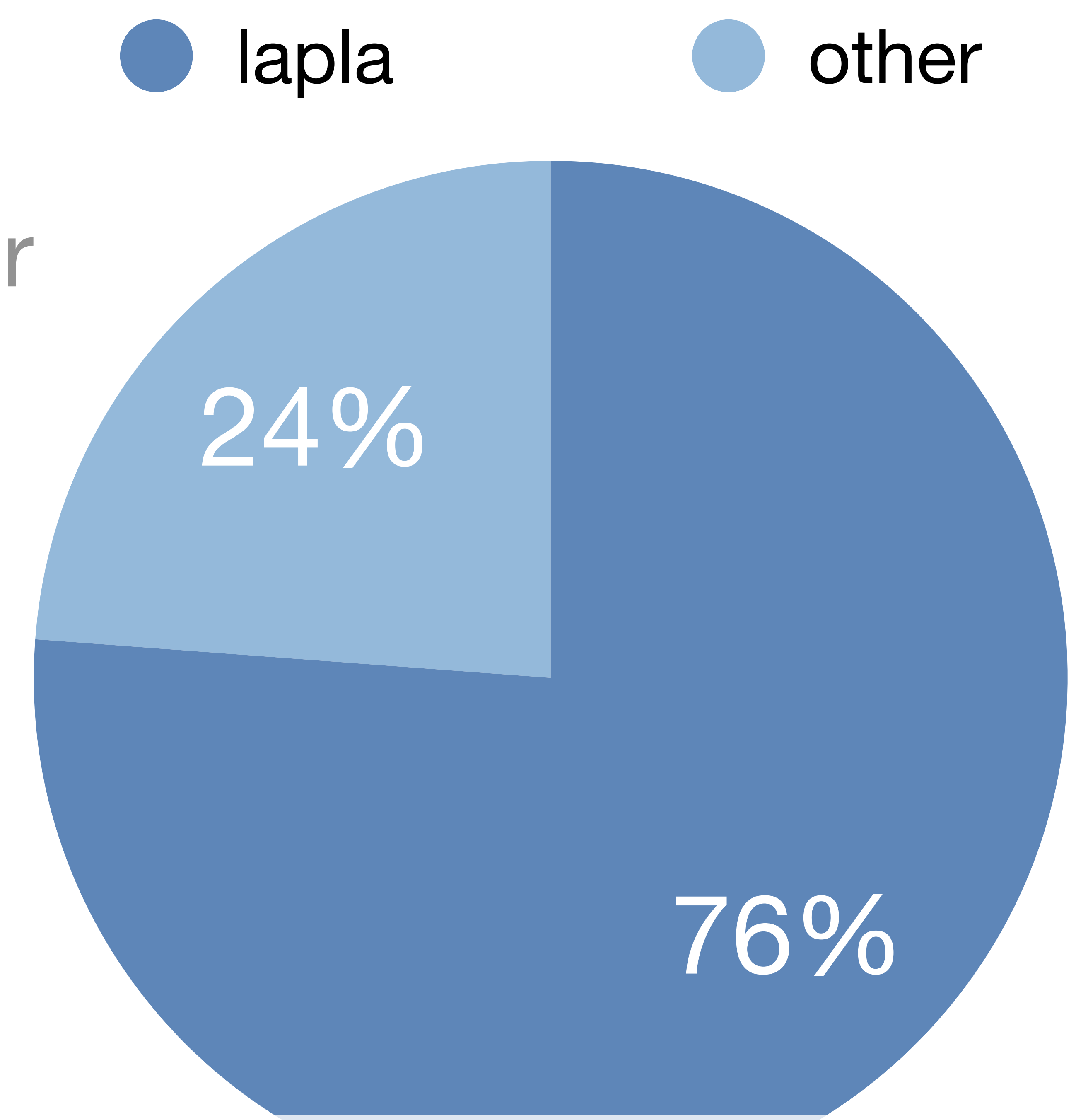
The Potential

It is the Bottleneck

Profiling BLOB

with Intel VTune Amplifier

~ 4 mins per interaction



3 mins: computing mean field laplacian

Elapsed Time: 231.966s

CPU Time: 231.938s

Total Thread Count: 1

Paused Time: 0s

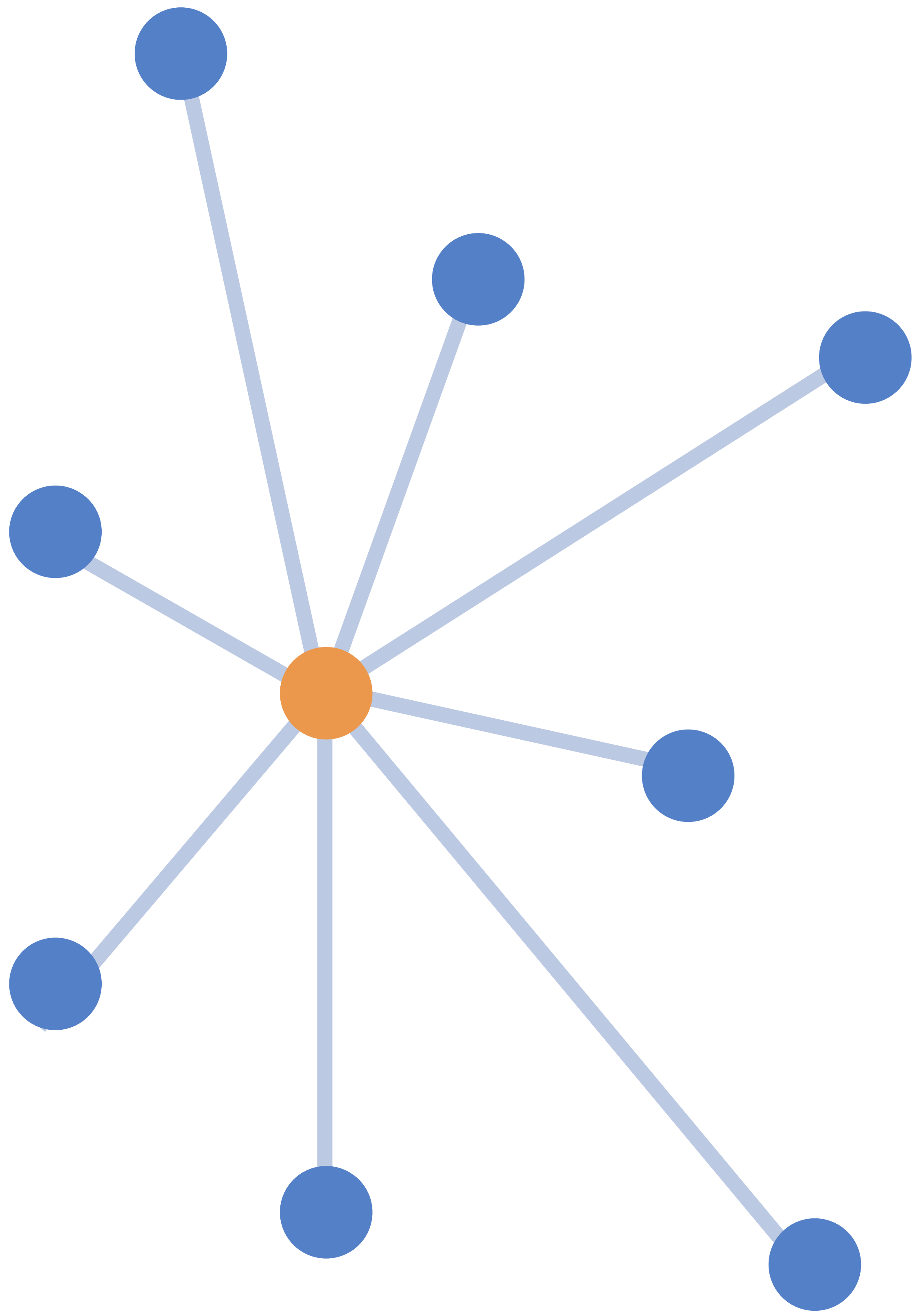
Top Hotspots

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

Learning the Potential: DL model

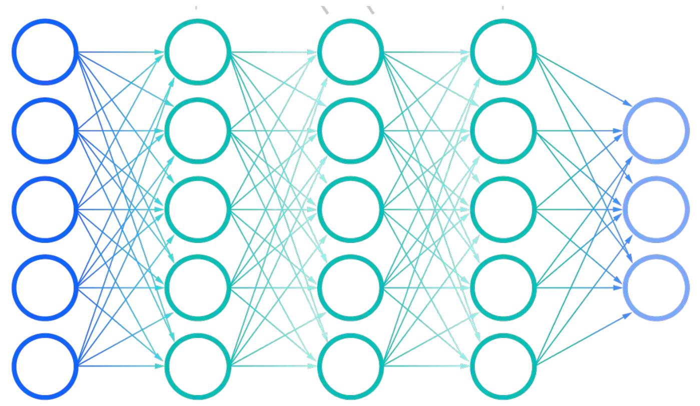
Particle-wise MLP for Potential Prediction



In QMD

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

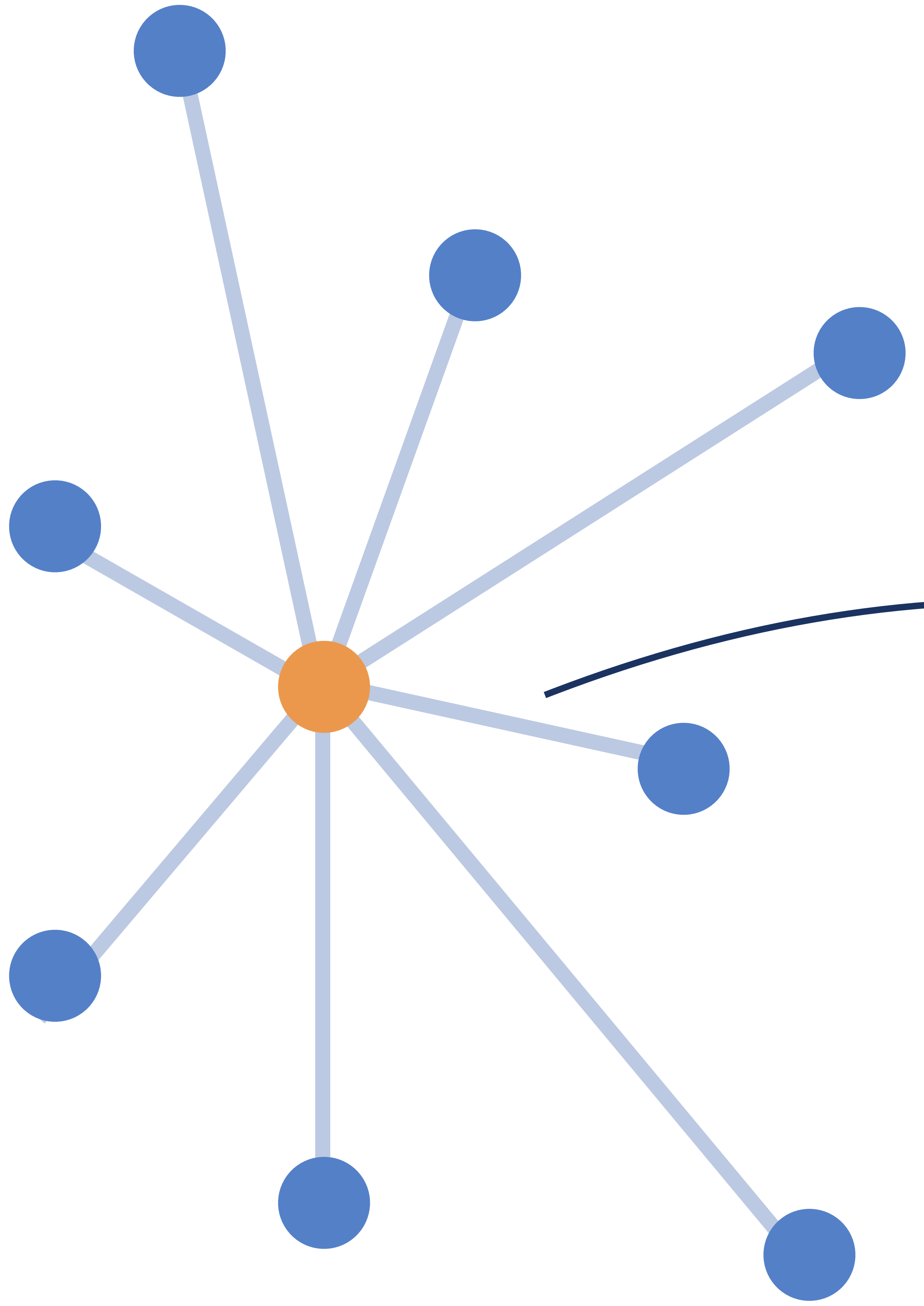
$$f\left(q_i, q_j, p_i, p_j, c_i, c_j\right) =$$



MLP

Learning the Potential: DL model

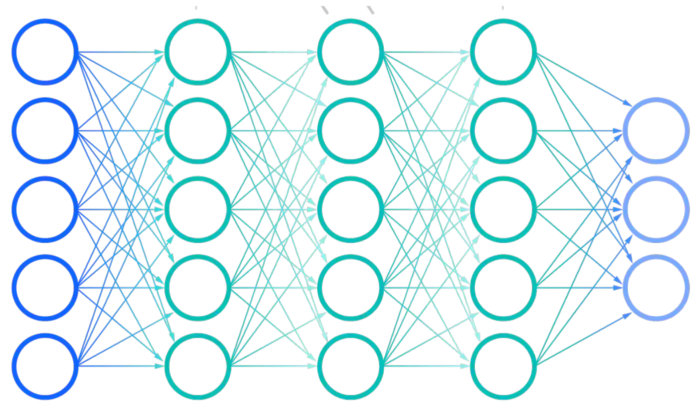
Particle-wise MLP for Potential Prediction



In QMD

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

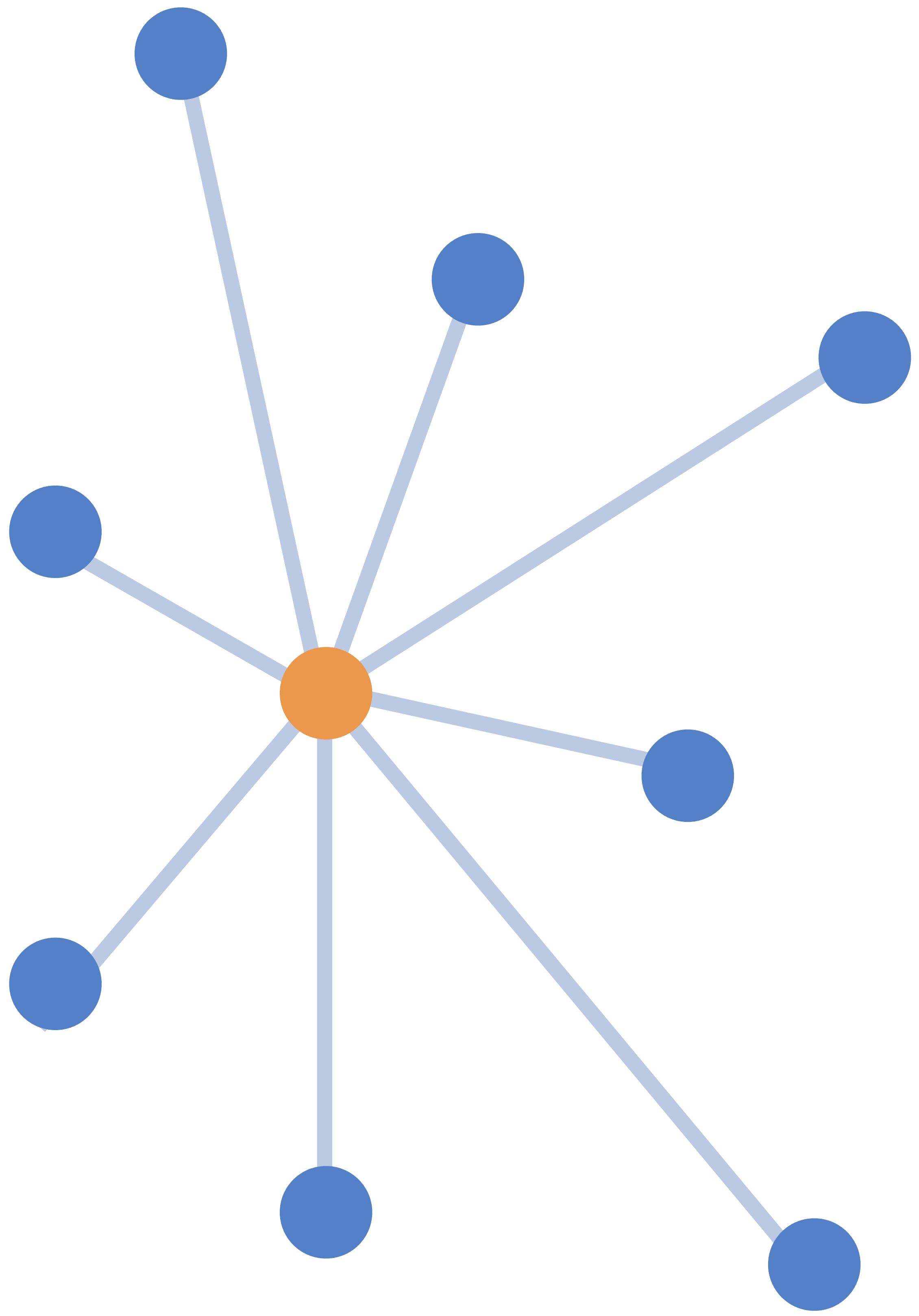
$$f\,(q_i, q_j, p_i, p_j, c_i, c_j) =$$



MLP

Learning the Potential: DL model

Particle-wise MLP for Potential Prediction



Building a DL model which:

- is **coherent** with the Physics

  Particle exchange symmetry embedded in the architecture

- works with **any** number of particles

  Particles are treated in **batch**

Potential Predictions

C12 on C12 at 62 MeV/u

Model:

5 layers MLP + ReLu + LayerNorm

Data:

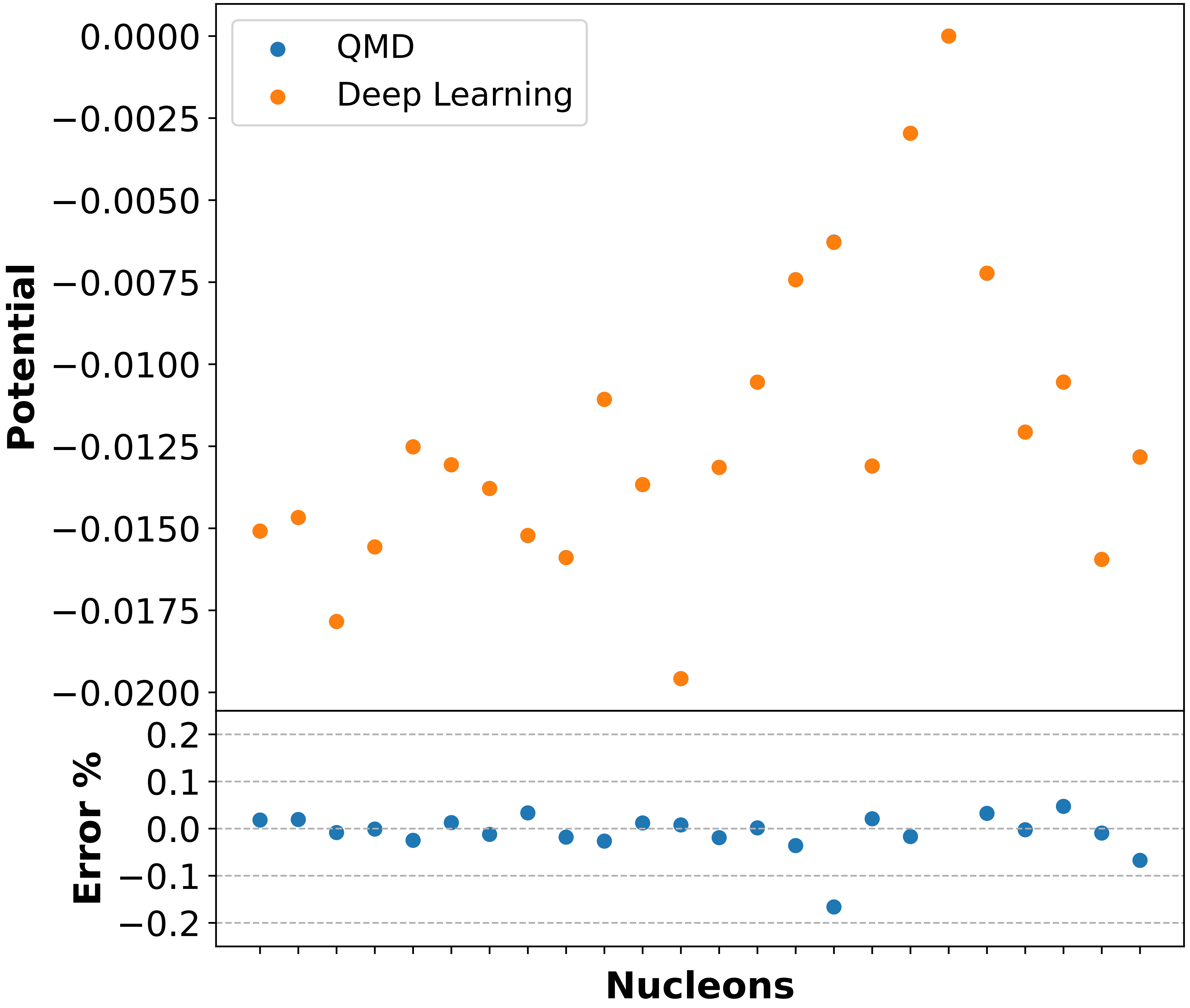
23k stories

10 events

24 particles : ~5 M examples

Training: ~3d training on Nvidia V100

Results: Median Relative Error 0,05 %



Is it useful for QMD itself?

- Recent development of **LightIonQMD** → Yoshi-hide Sato *et al* 2022 *Phys. Med. Biol.* **67** 225001

- Possibilities to improve the model → Currently bounded by **execution time** requirements

Can Deep Learning be applied to **accelerate QMD**?

Implementation in Geant4

Exporting the DL models from pytorch to **ONNX**

Using ONNX C++ API → substituting GetPotential() Method in QMD

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

Thread-safe implementation

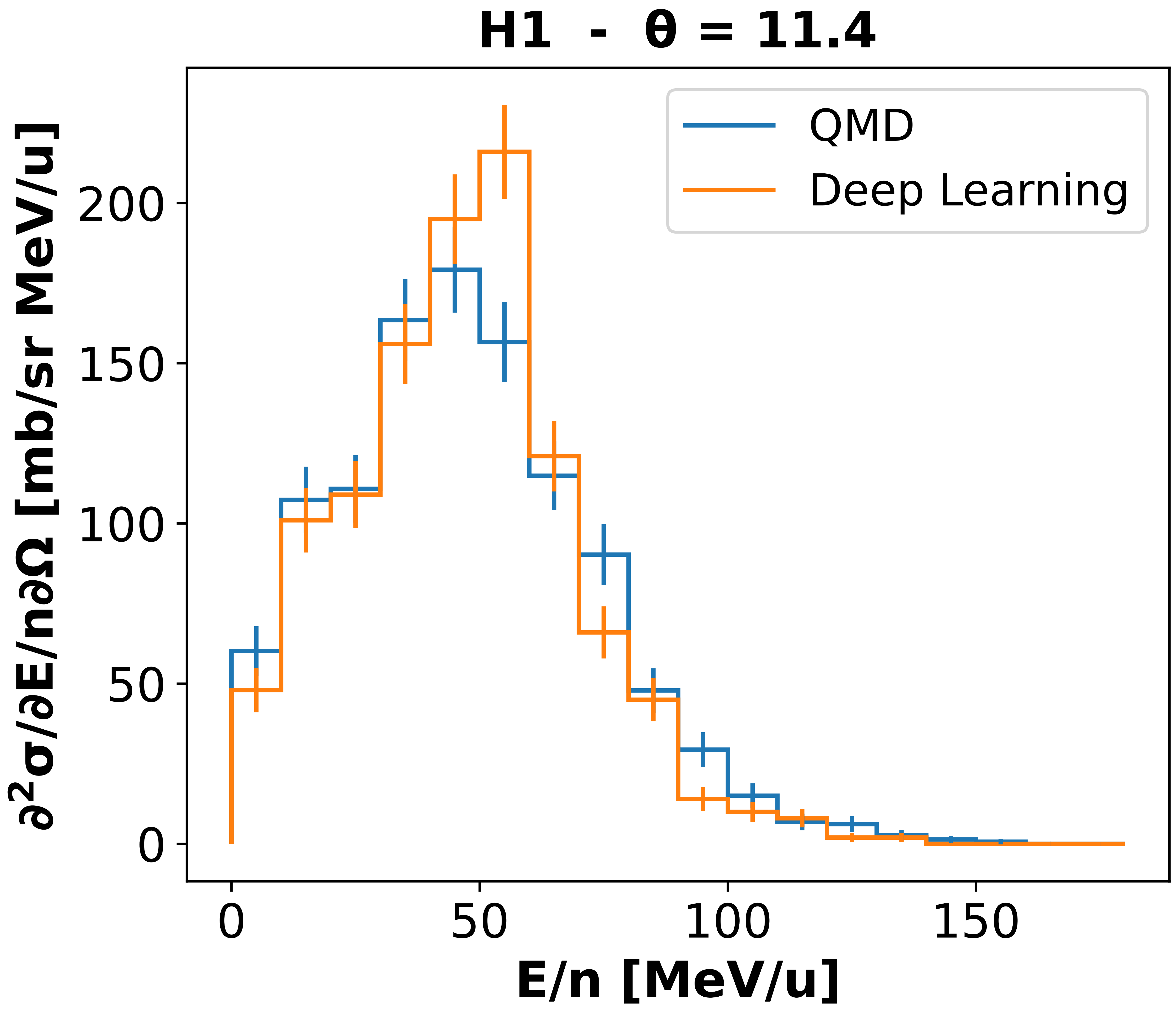
Test on the Potential

Simulating the reaction:

C12 on C_nat at 62 MeV/u

Interfacing DL model with Geant4

- Reasonable accuracy on double differential cross section of **lighter fragments**



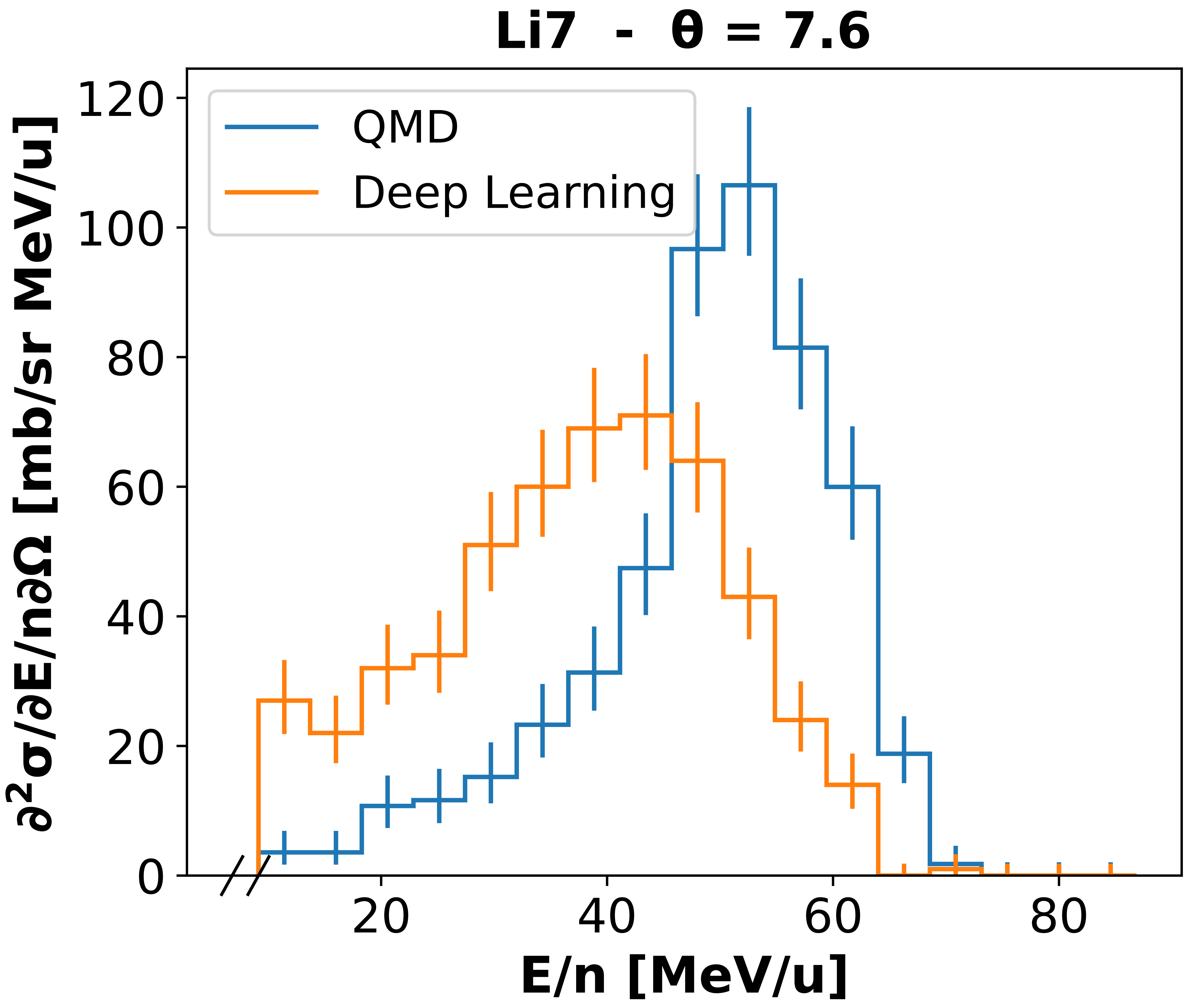
Test on the Potential

However: for heavier fragments

- Even **small errors** on the potential **propagate** badly to the double differential cross sections

- It is not the bottleneck!

Only 4% of QMD execution time



Another possibility

Derivatives of the Hamiltonian

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

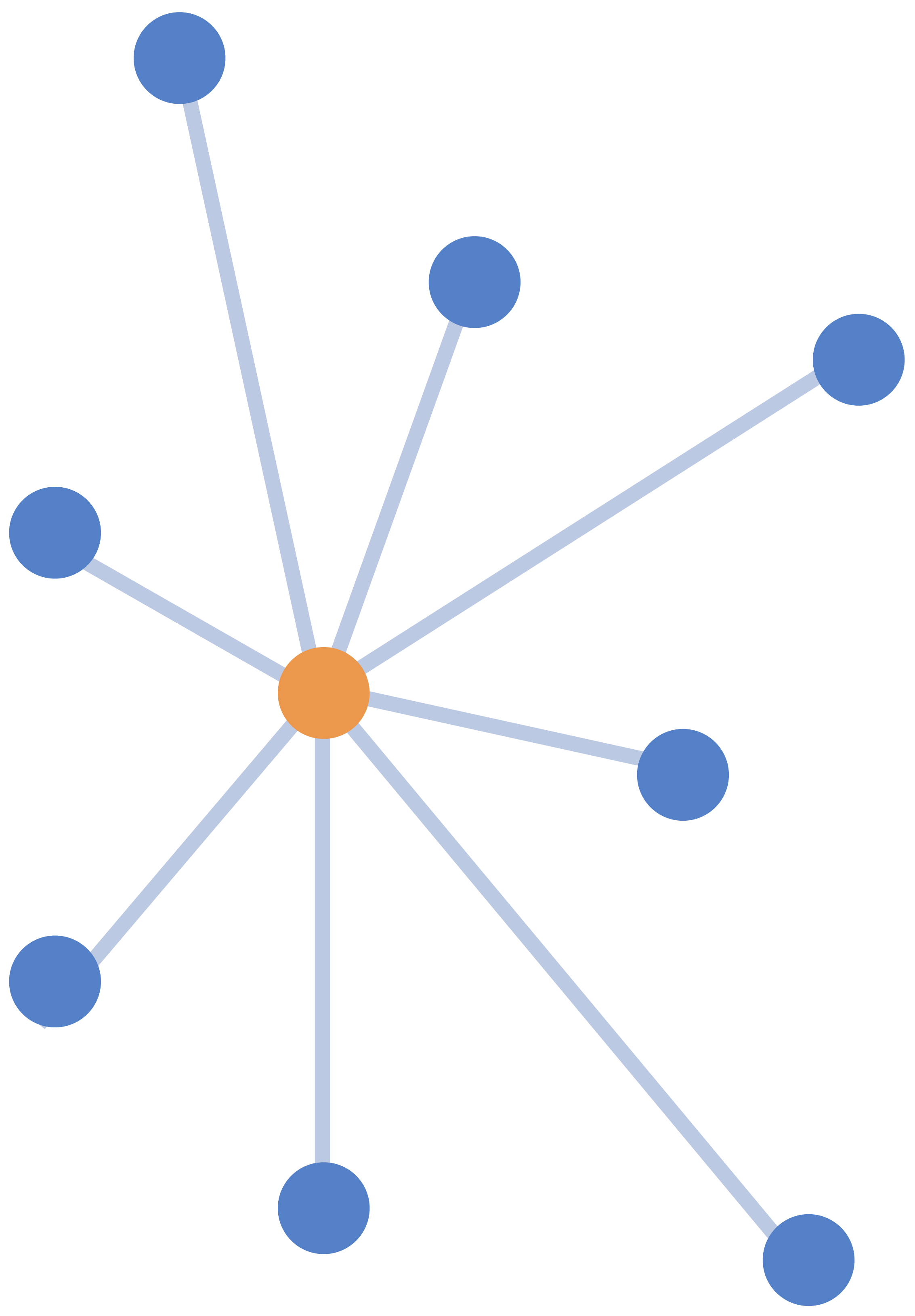
1) Cross sections are **resilient** to 1-2% errors

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

2) This is the **bottleneck!**

CalGraduate() is **50%** of QMD

Emulating the derivatives



Same architectural design of the Potential model

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

Approximating the derivatives

Hyper-parameter optimization on the number of **terms K**

Derivatives prediction

C12 on C12 at 62 MeV/u

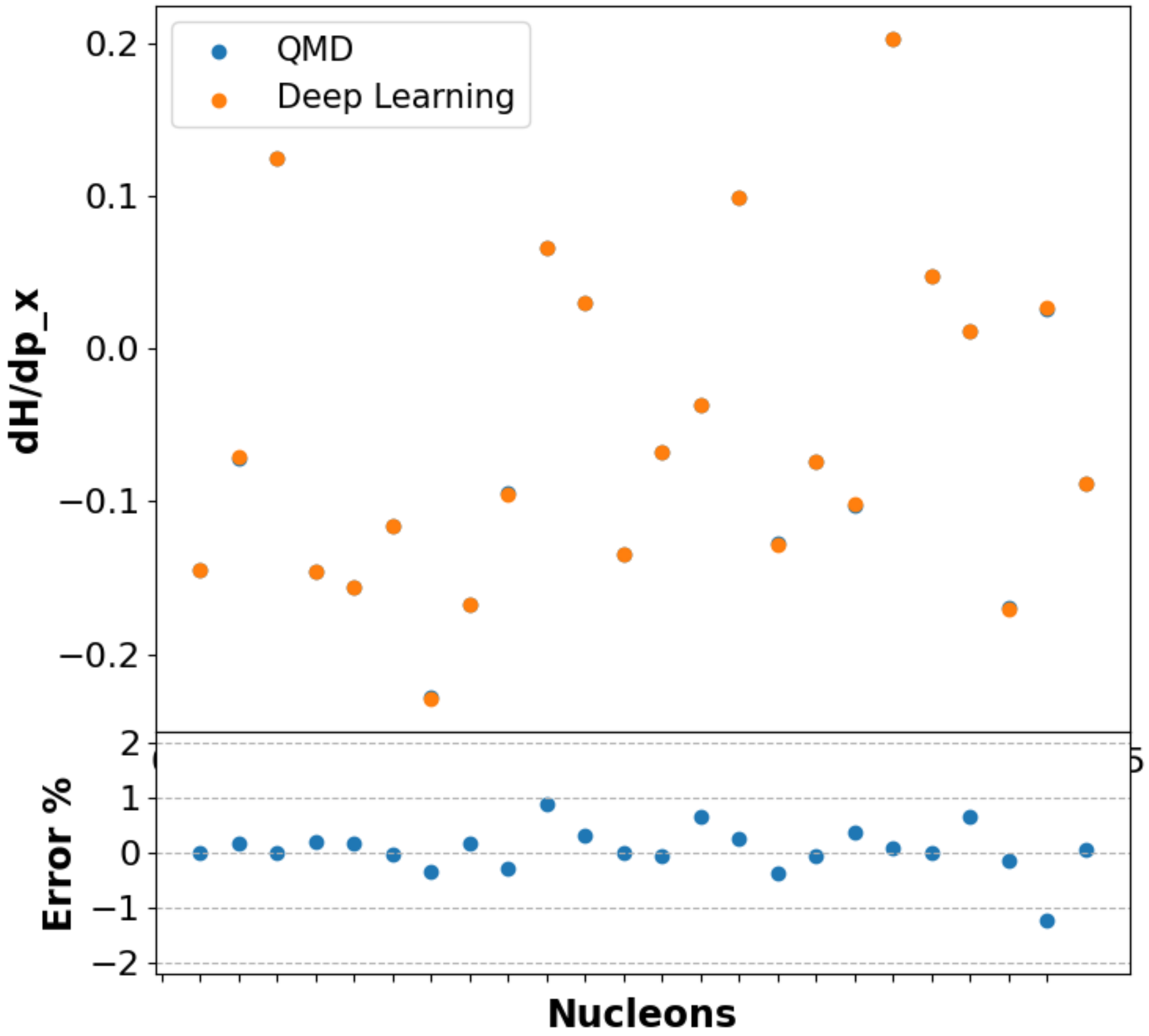
Model: 2 $\alpha^{(k)}$ terms +
5 layers MLP + ReLu + LayerNorm

Data:

12k stories
1 events
24 particles : ~300k examples

Training: ~3h training on Nvidia V100

Results: Median Relative Error 0,6 %



Implementation in Geant4

Exporting the DL models from pytorch to **ONNX**

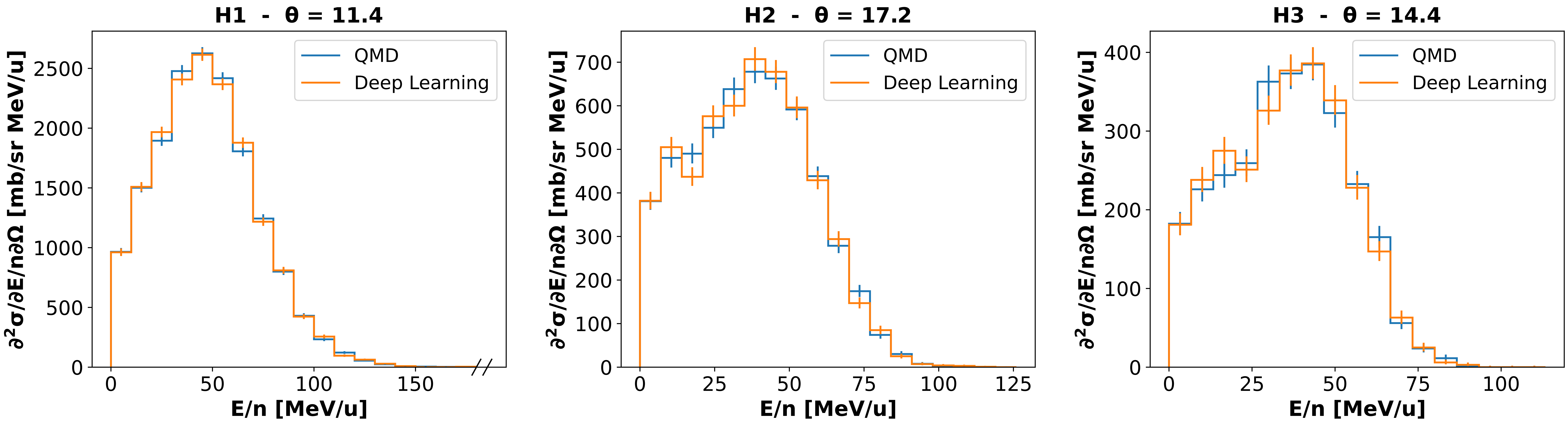
Using ONNX C++ API → substituting CalGraduate() Method in QMD

```cpp
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

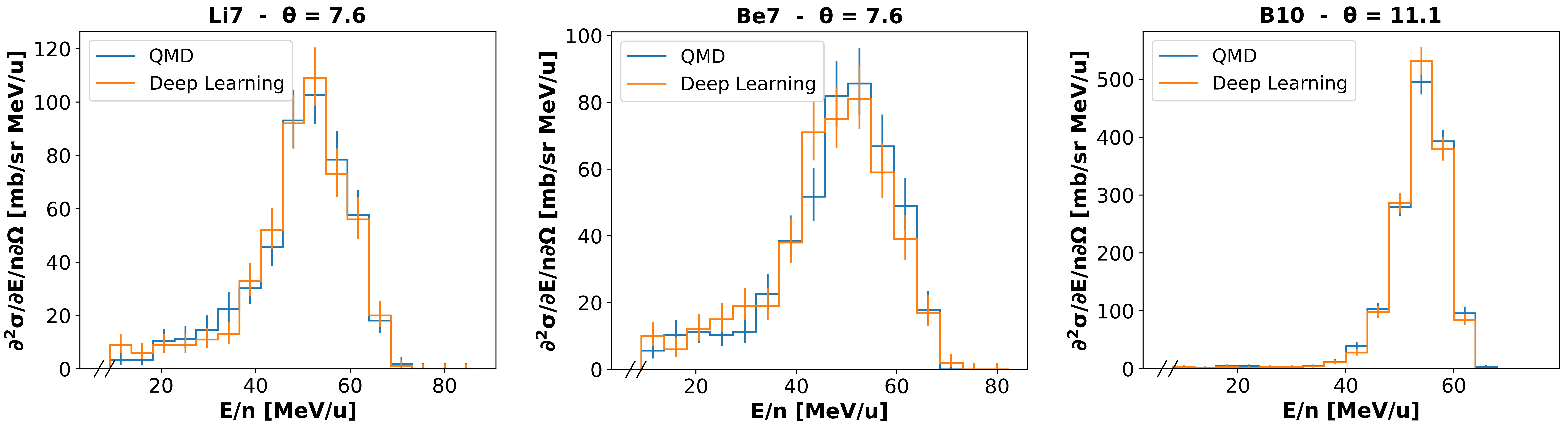
Double differential cross sections

Running LoweFrag example: C12 on C_nat at 62 MeV/u



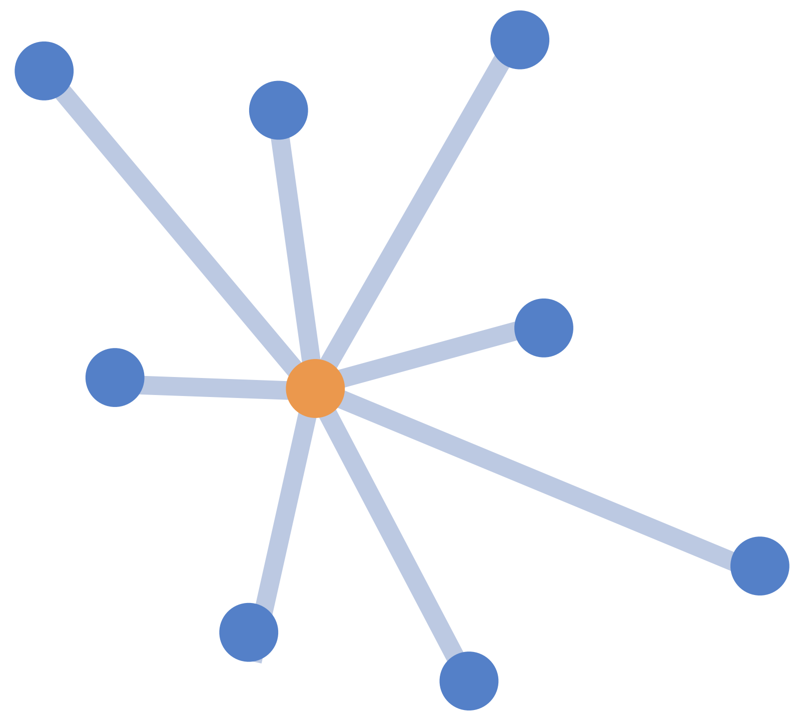
Double differential cross sections

Running LoweFrag example: C12 on C_nat at 62 MeV/u



Light Ion QMD

Emulating the derivatives

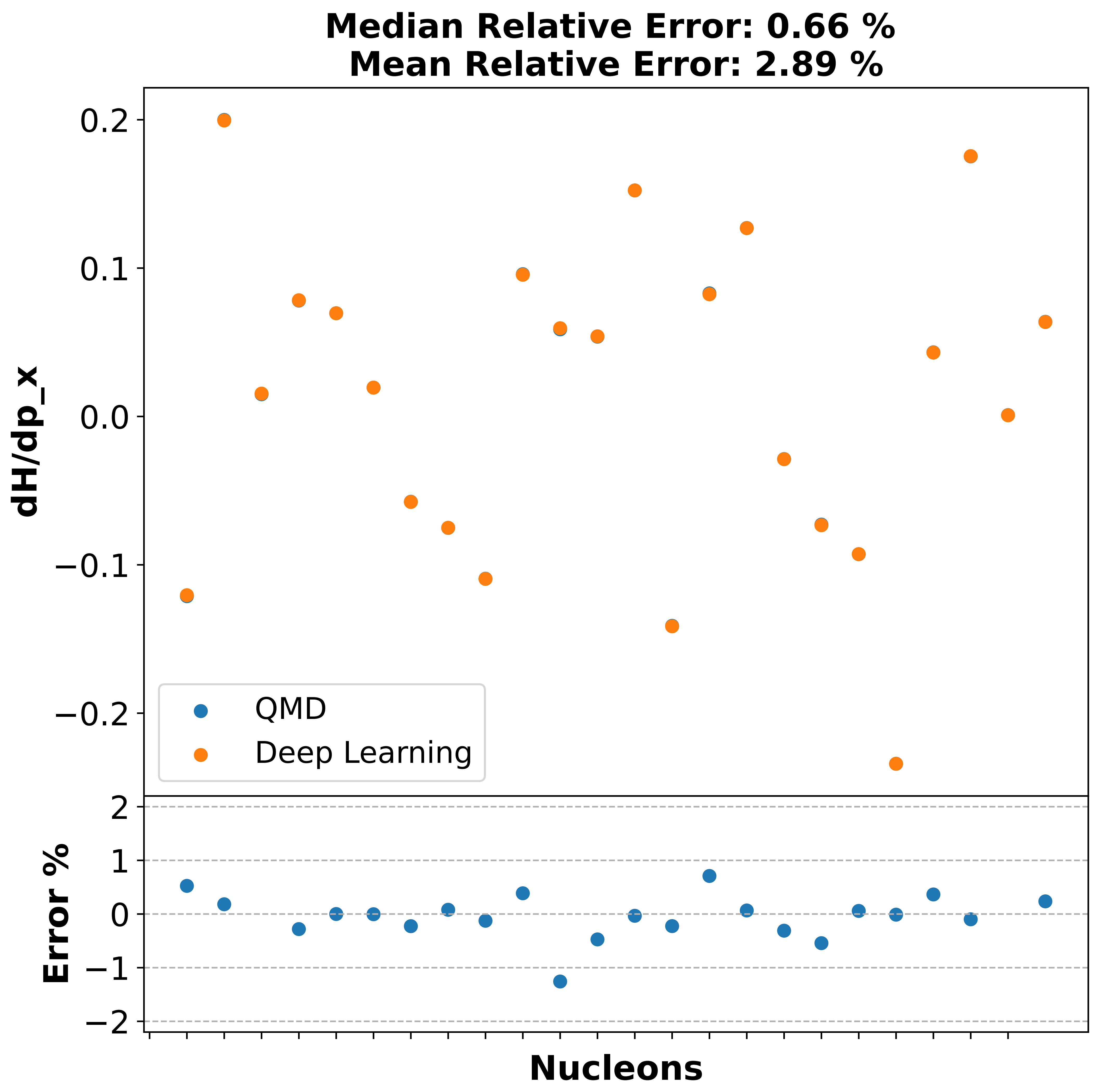


$$\frac{\partial H}{\partial q, p} \approx \sum A_{ij} + \sum_{\alpha^{(k)}} \left(\sum B_{ij}^{(k)} \right)^{\alpha^{(k)}}$$

Model: 3 $\alpha^{(k)}$ terms +
4 layers MLP + ReLu + LayerNorm

Results: Median Relative Error 0,7 %

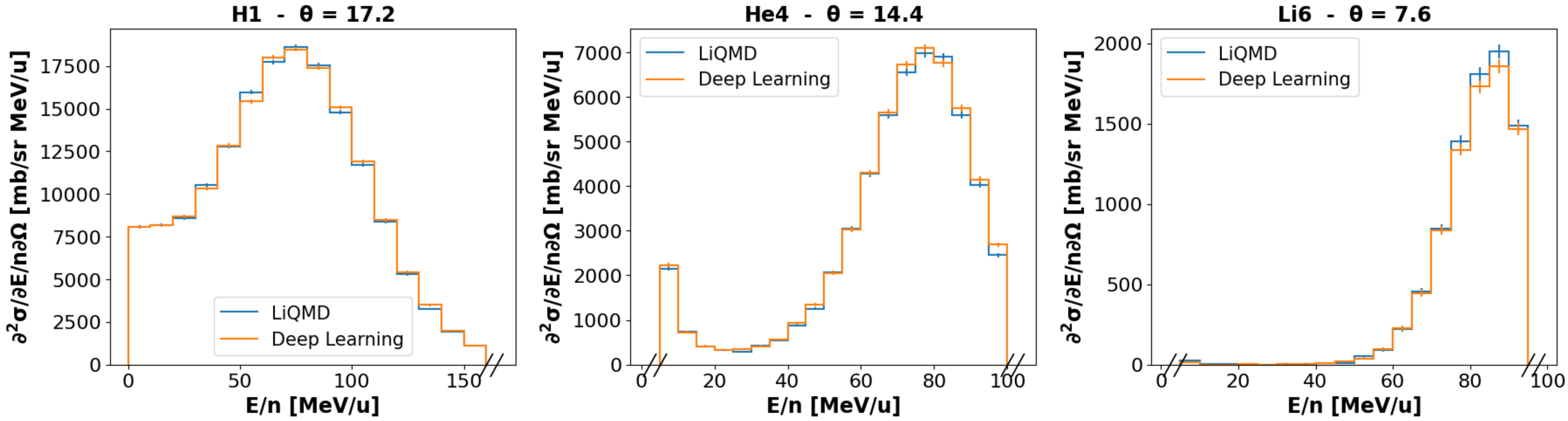
C12 on C12 at 95 MeV/u



Light Ion QMD

Double differential cross sections

Running LoweFrag example: C12 on C_nat at 95 MeV/u



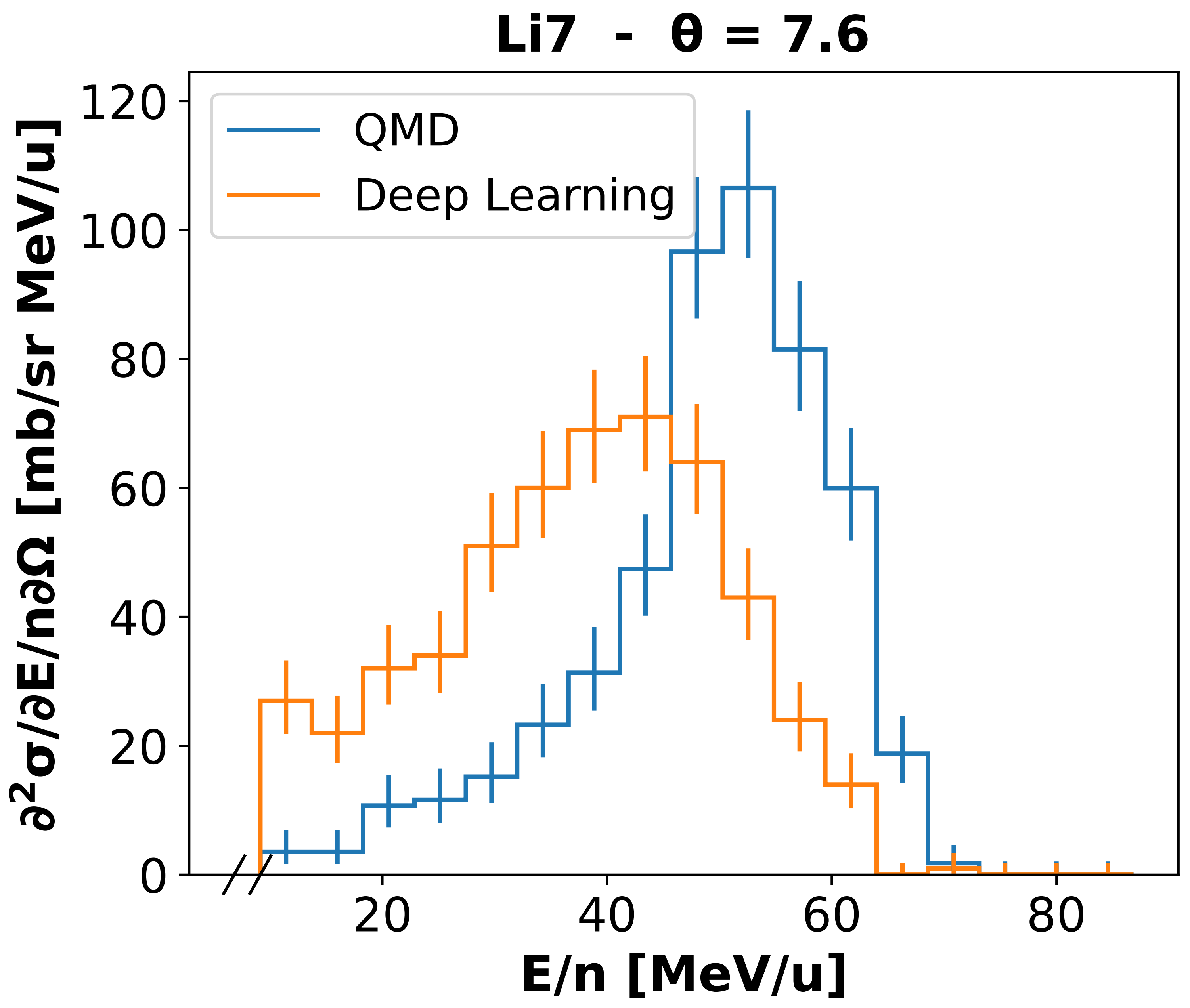
Range of applicability

Until now: Model trained and tested on the same reaction at the same energy

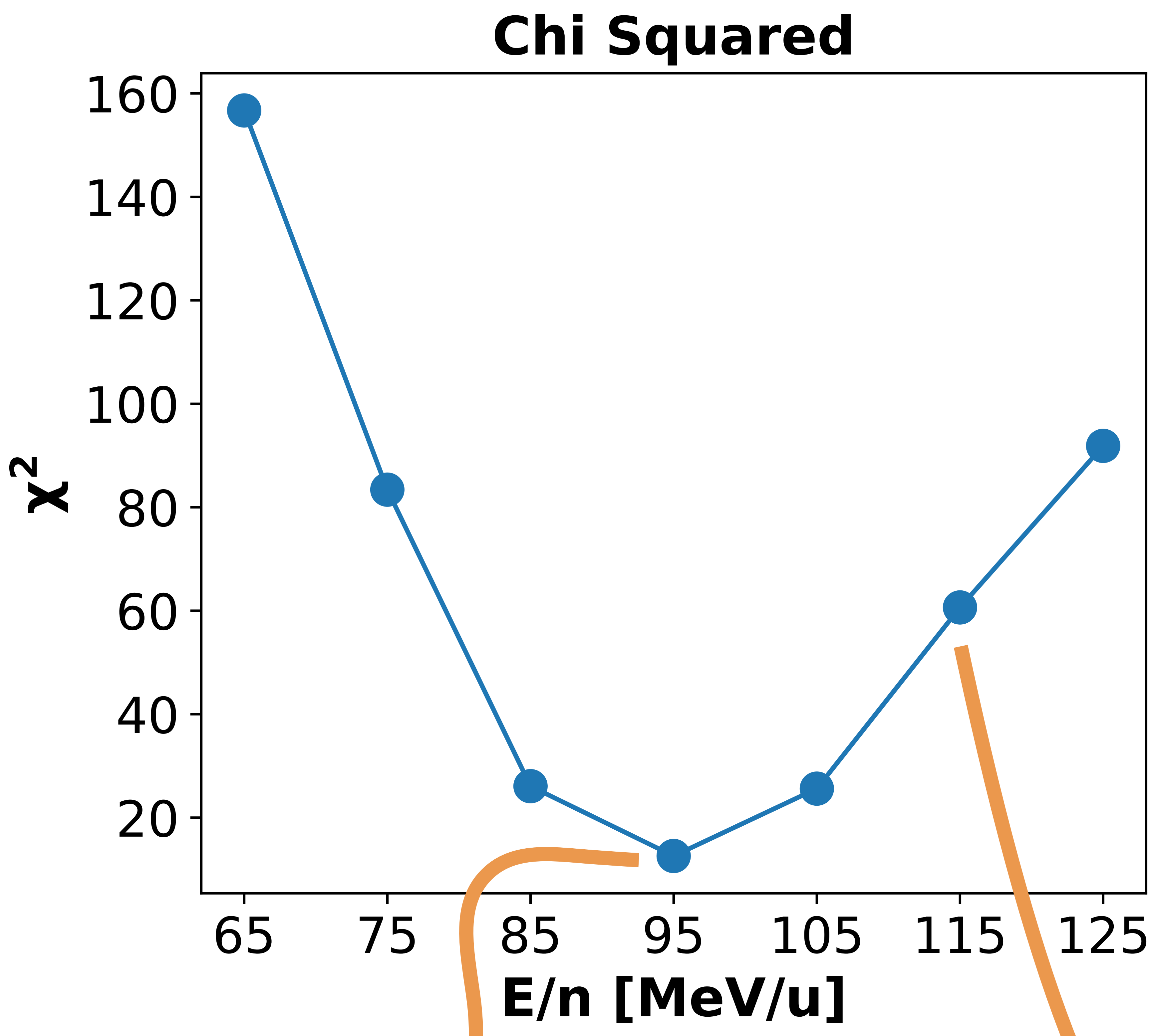
What we want: A model that works for any "reasonable" ions and energies

Metric to assess the double differential cross section consistency

$$\chi^2 = \frac{1}{N_{bins}} \sum_{i}^{N_{bins}} \frac{(N_i^{(MC)} - N_i^{(DL)})^2}{N_i^{(MC)} + N_i^{(DL)}}$$

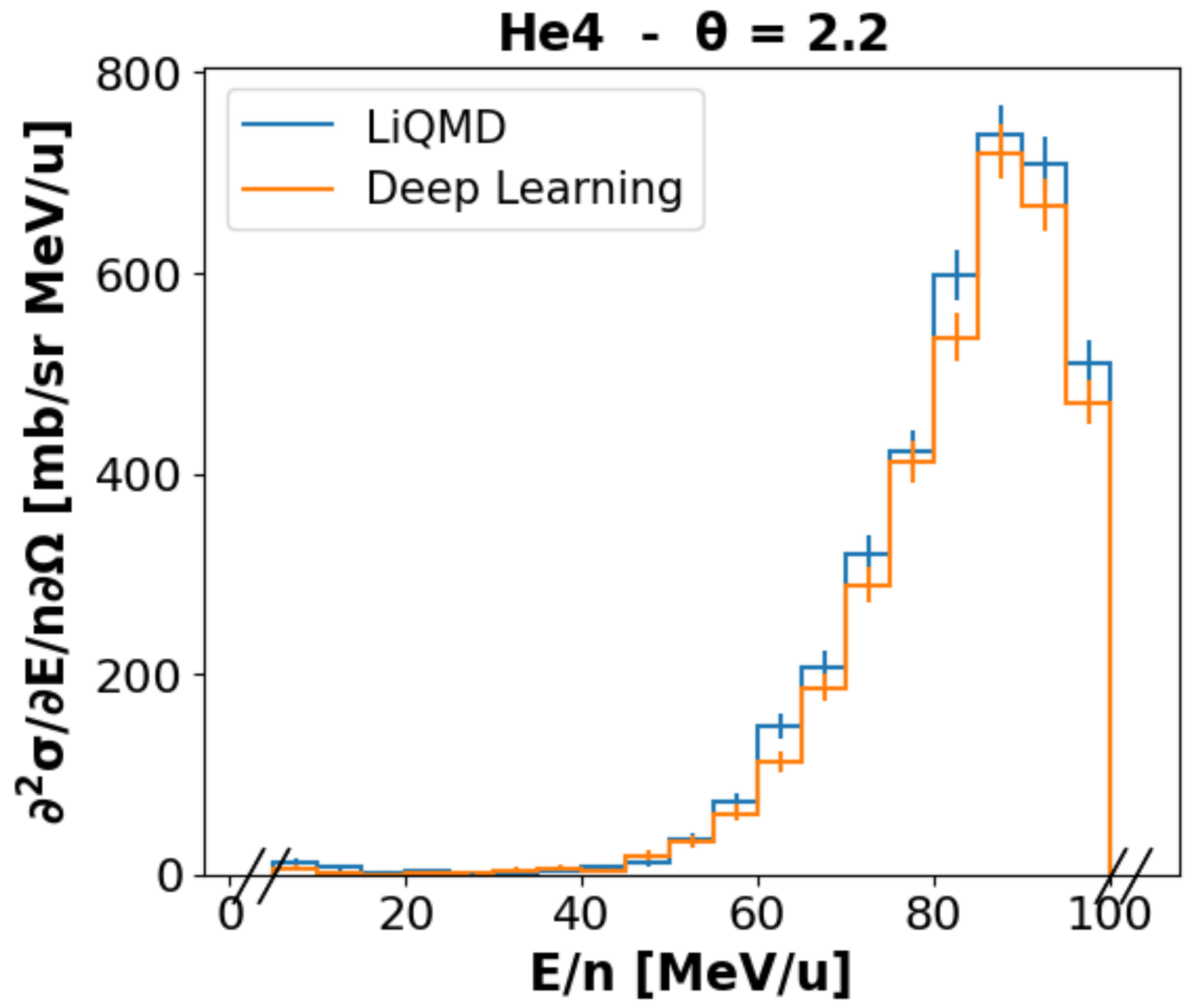


Energies



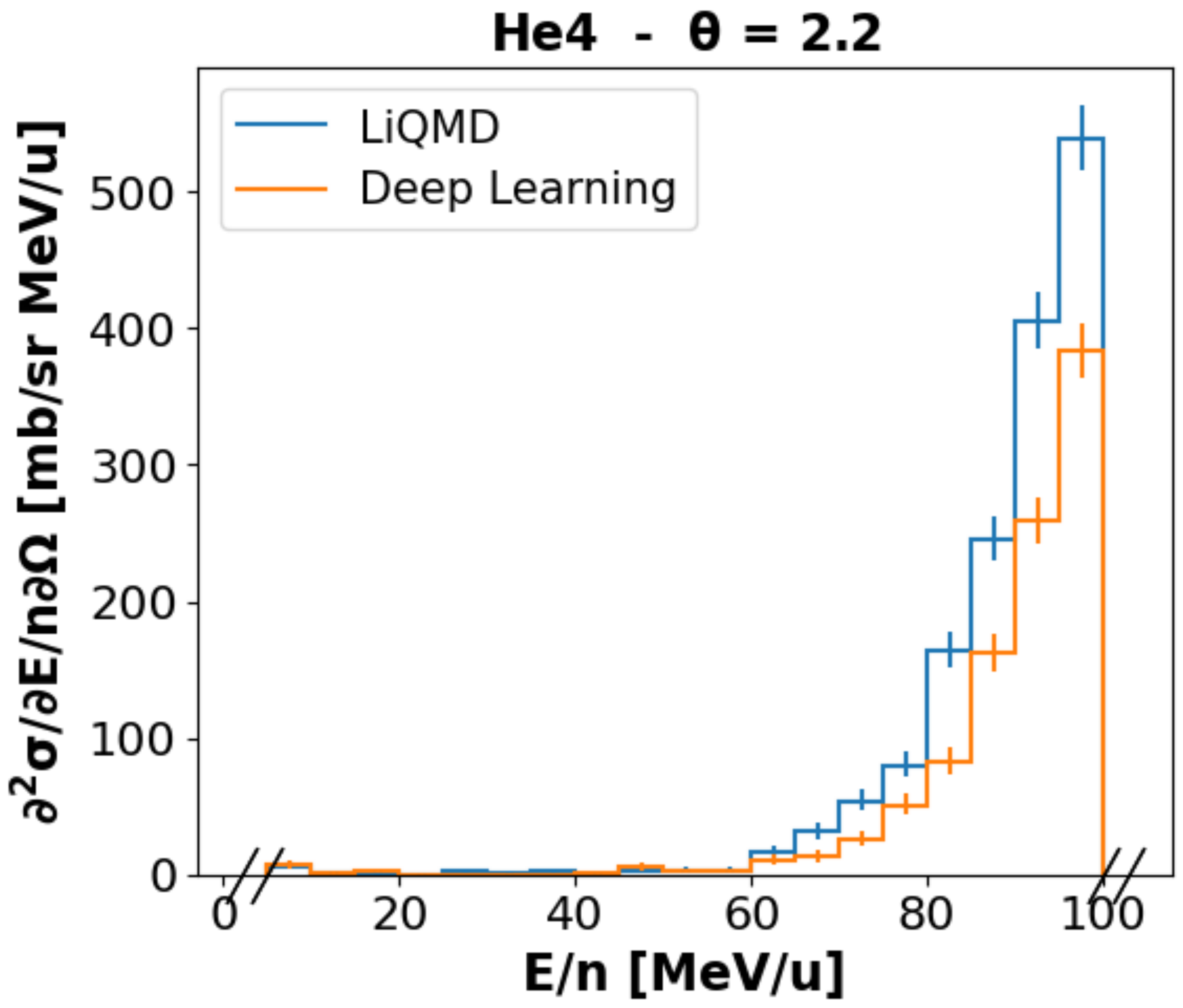
C12 on C_nat

95 MeV/u



C12 on C_nat

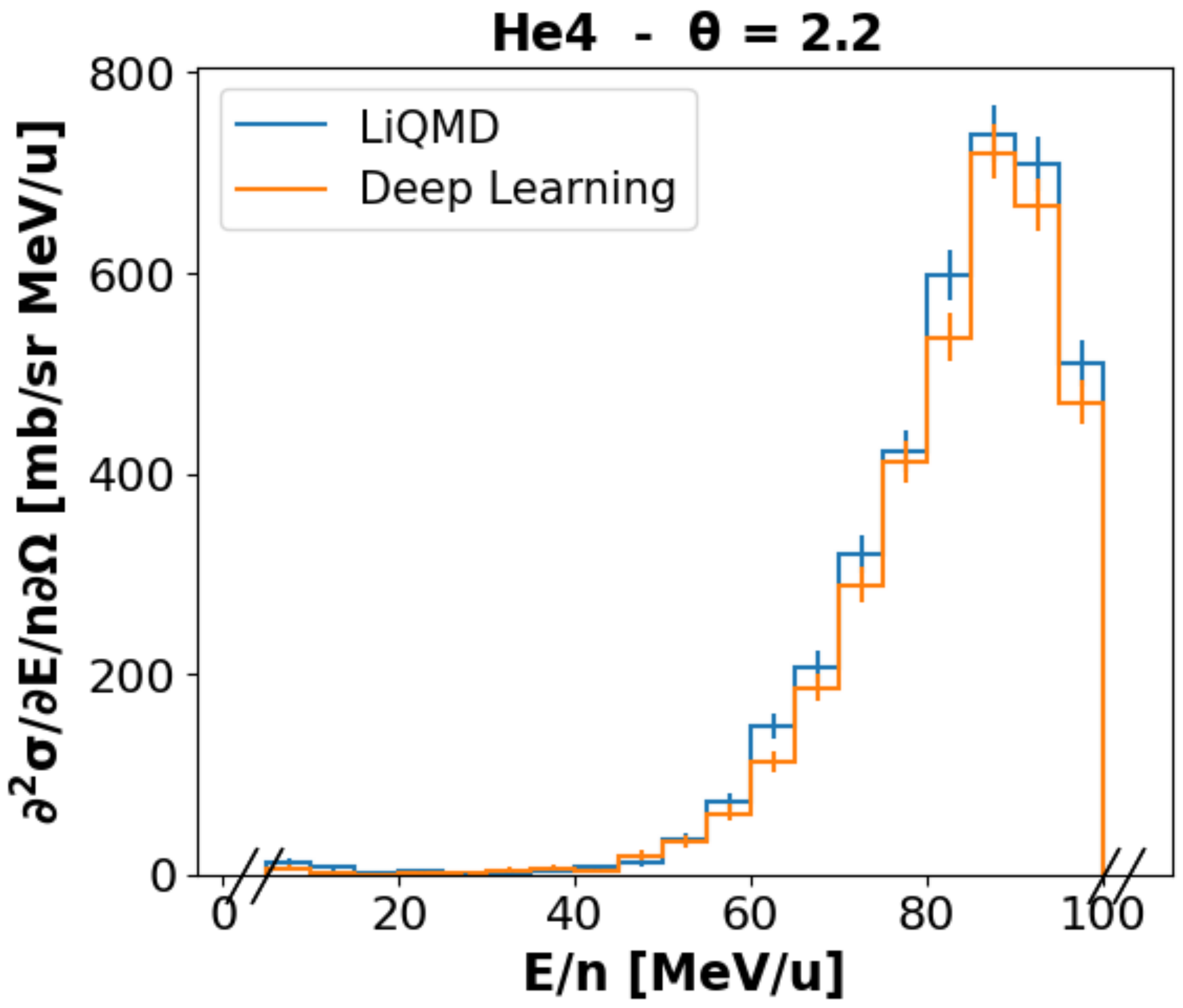
115 MeV/u

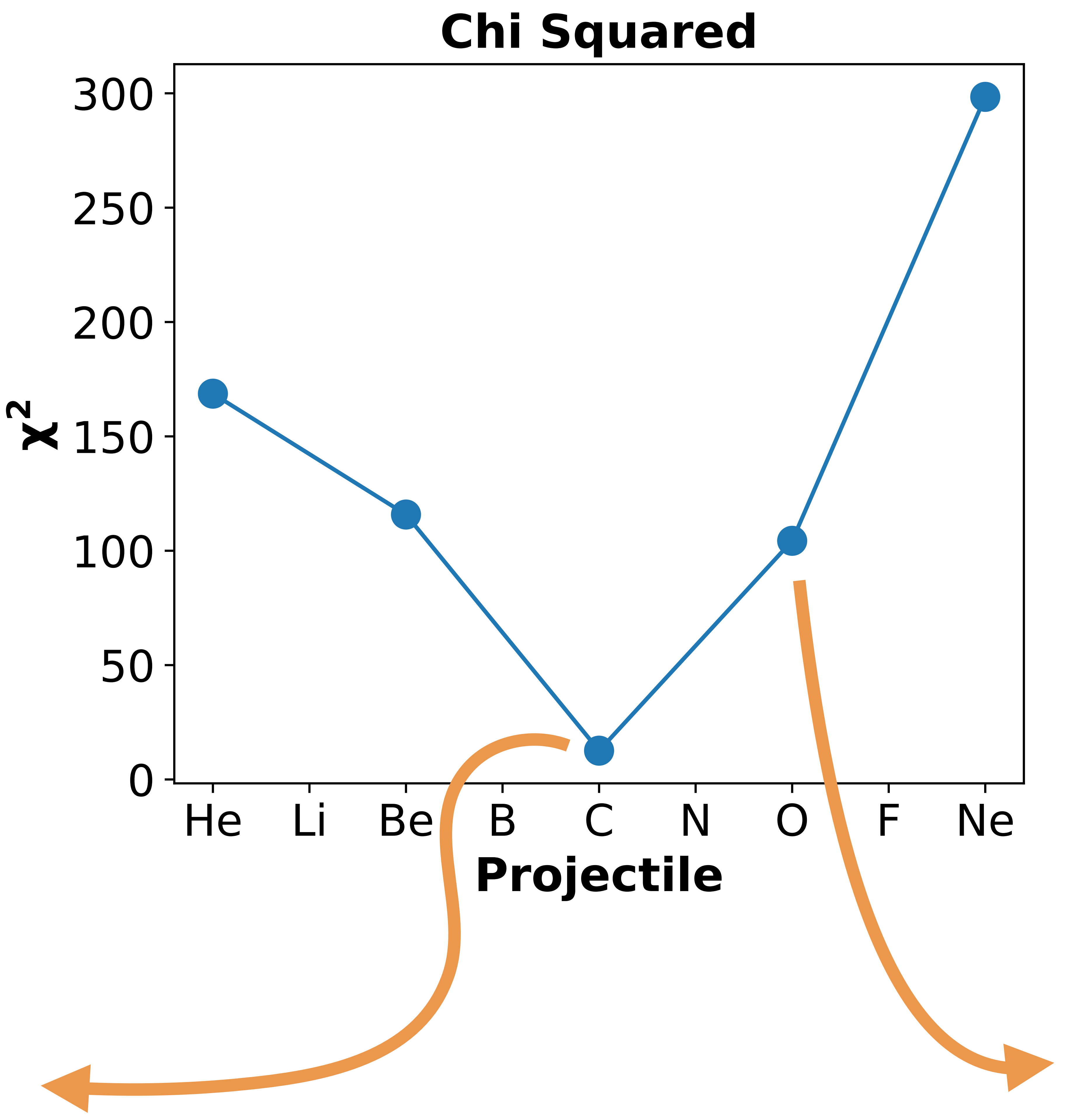


Ions

C12 on C_nat

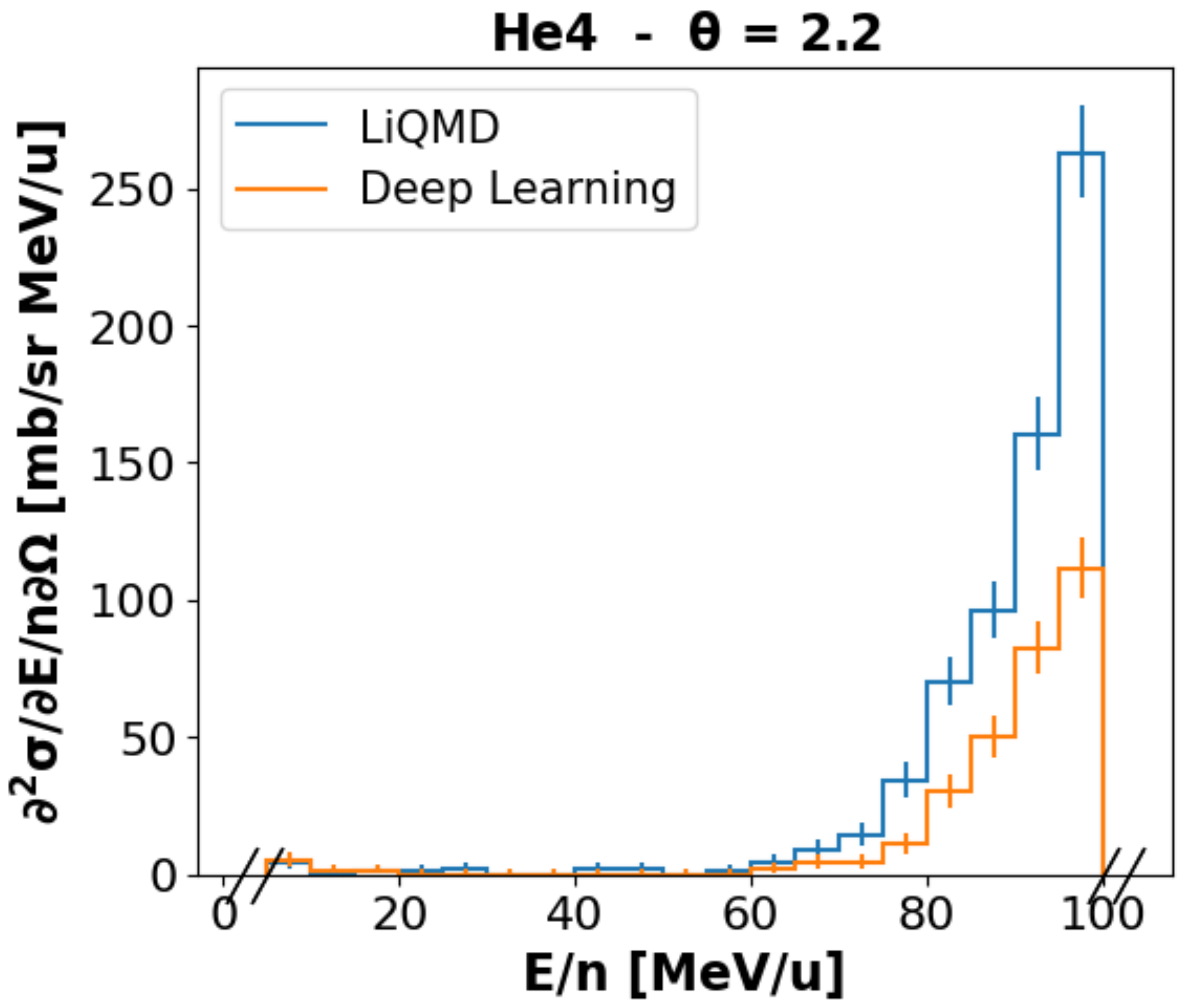
95 MeV/u



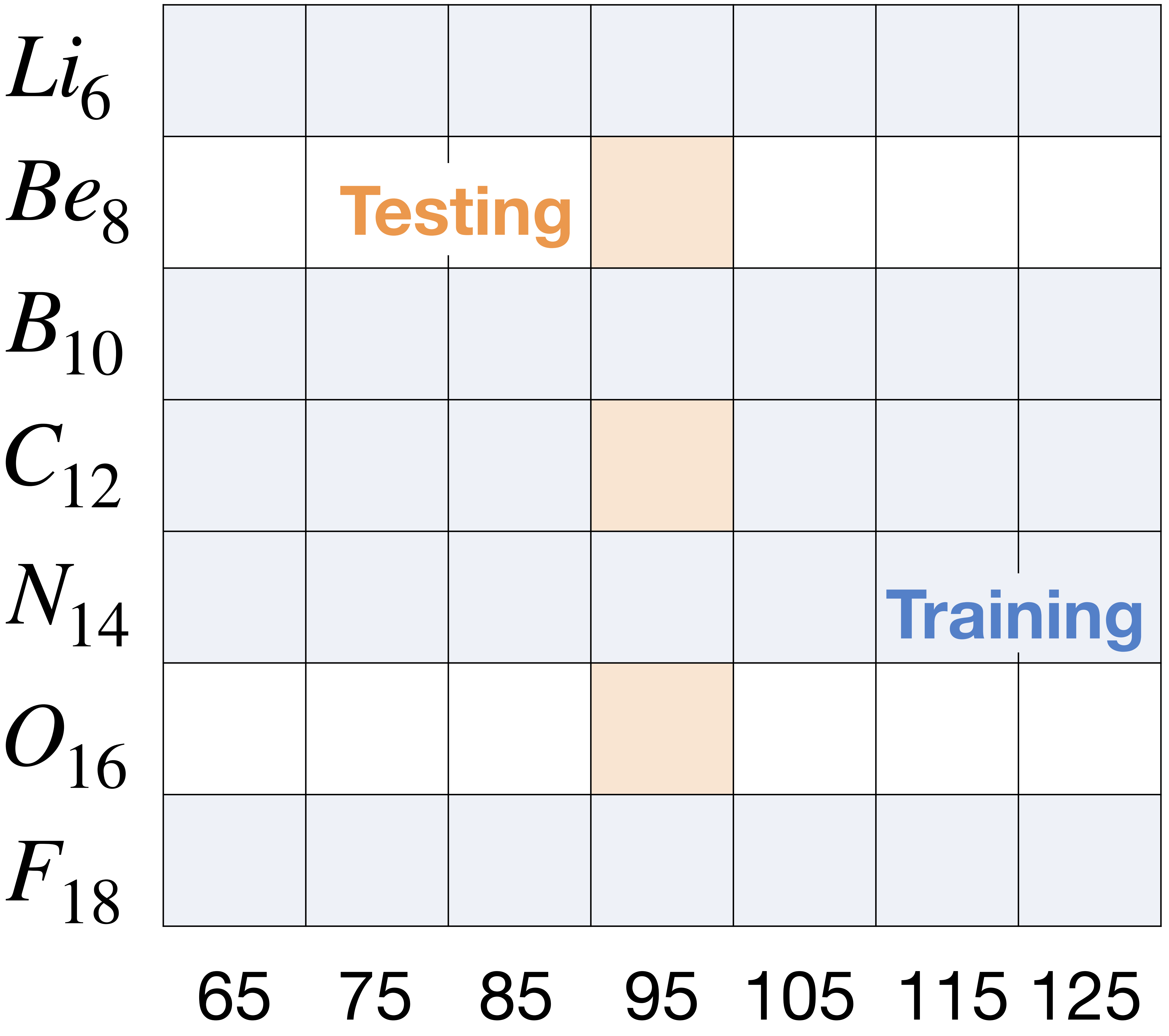


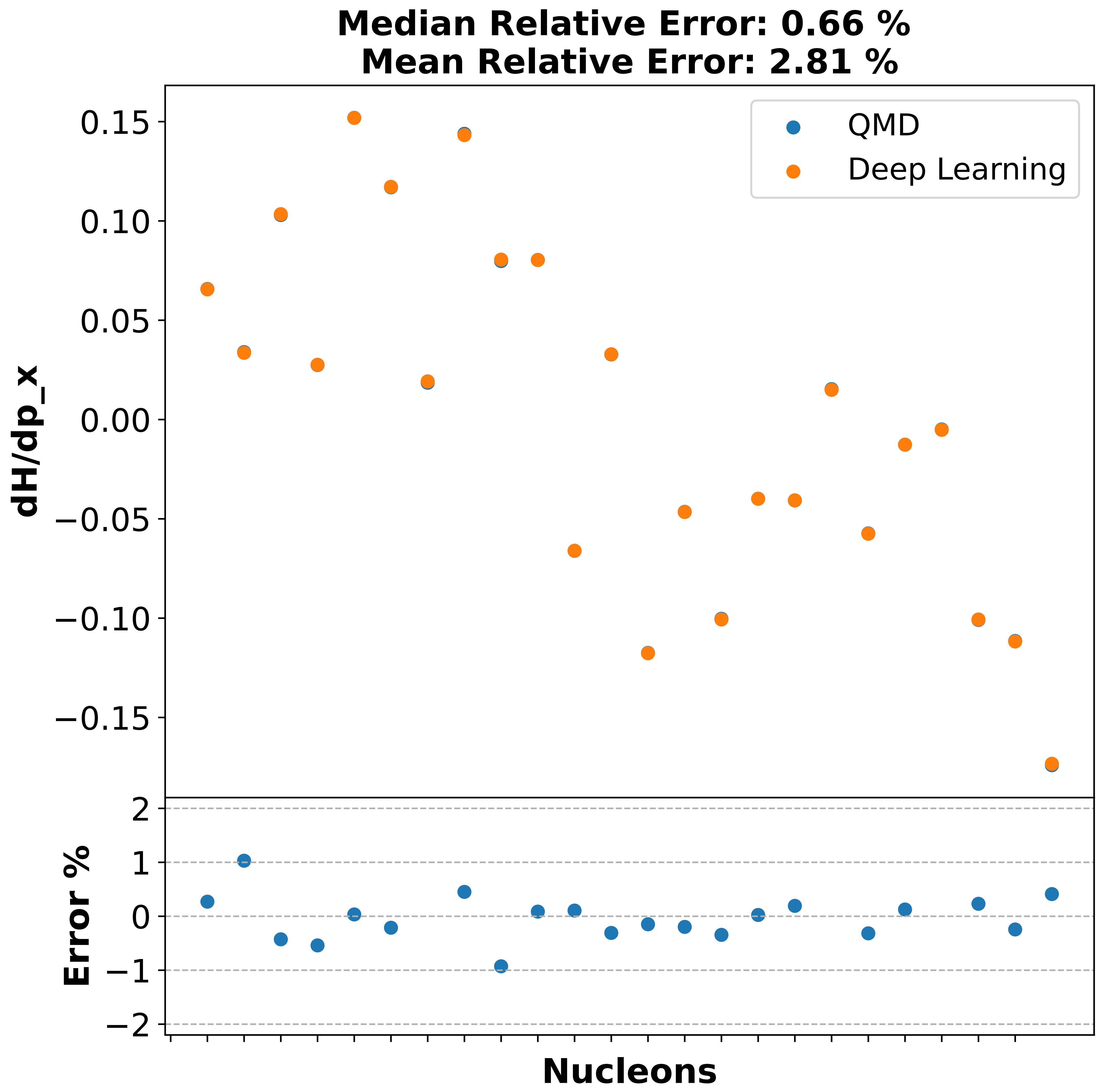
O16 on C_nat

95 MeV/u



Extending the training

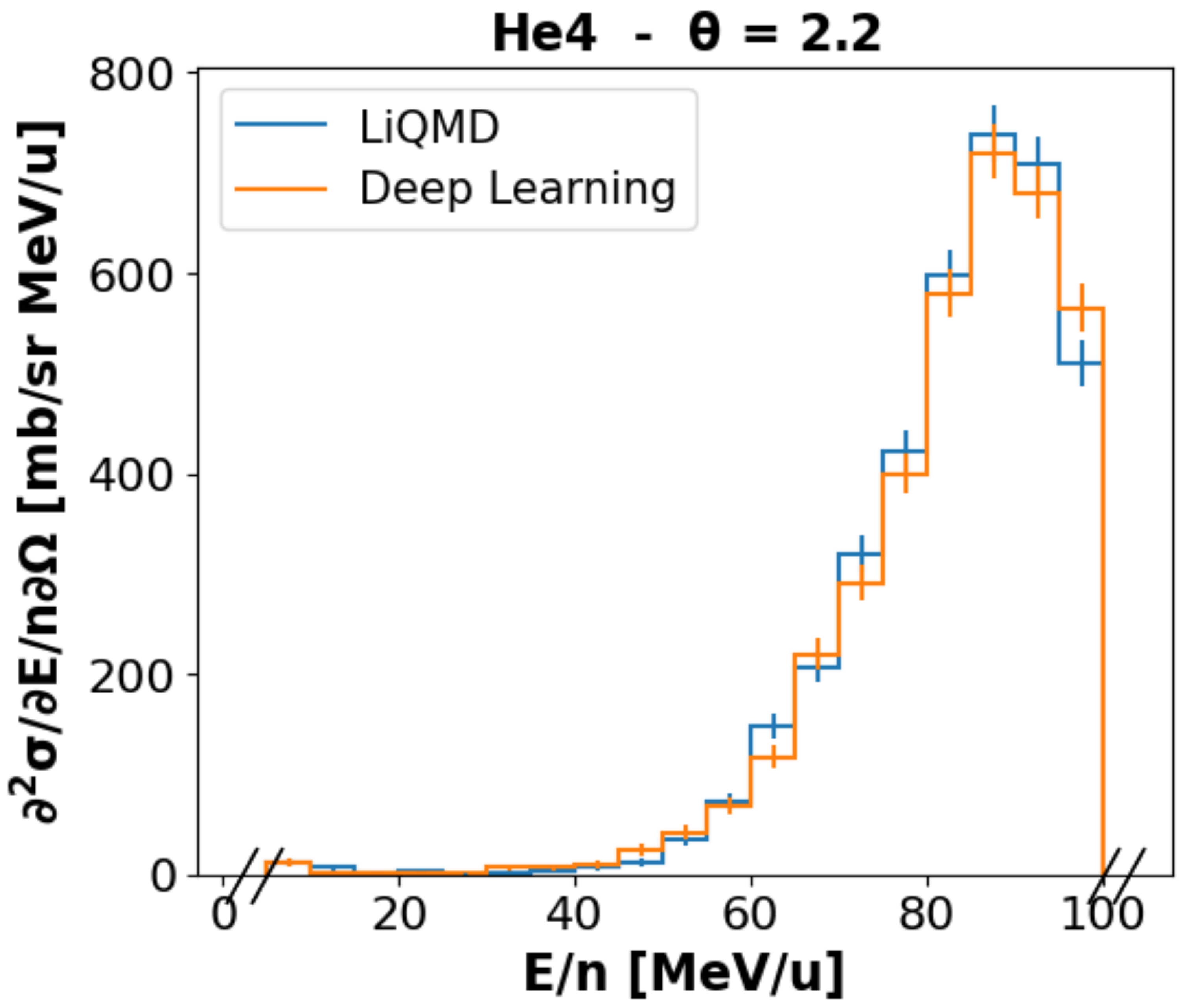


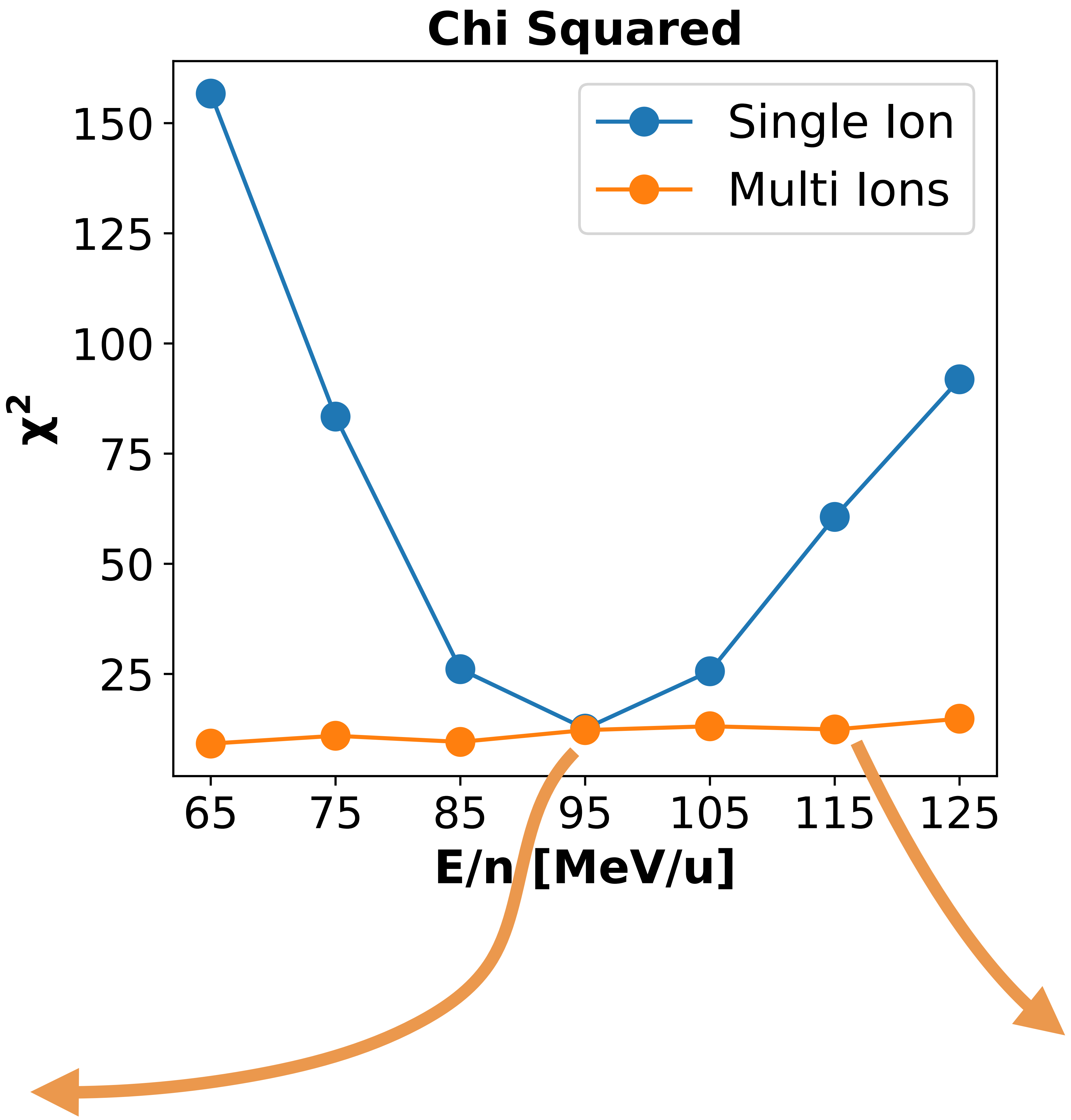


Energies

C12 on C_nat

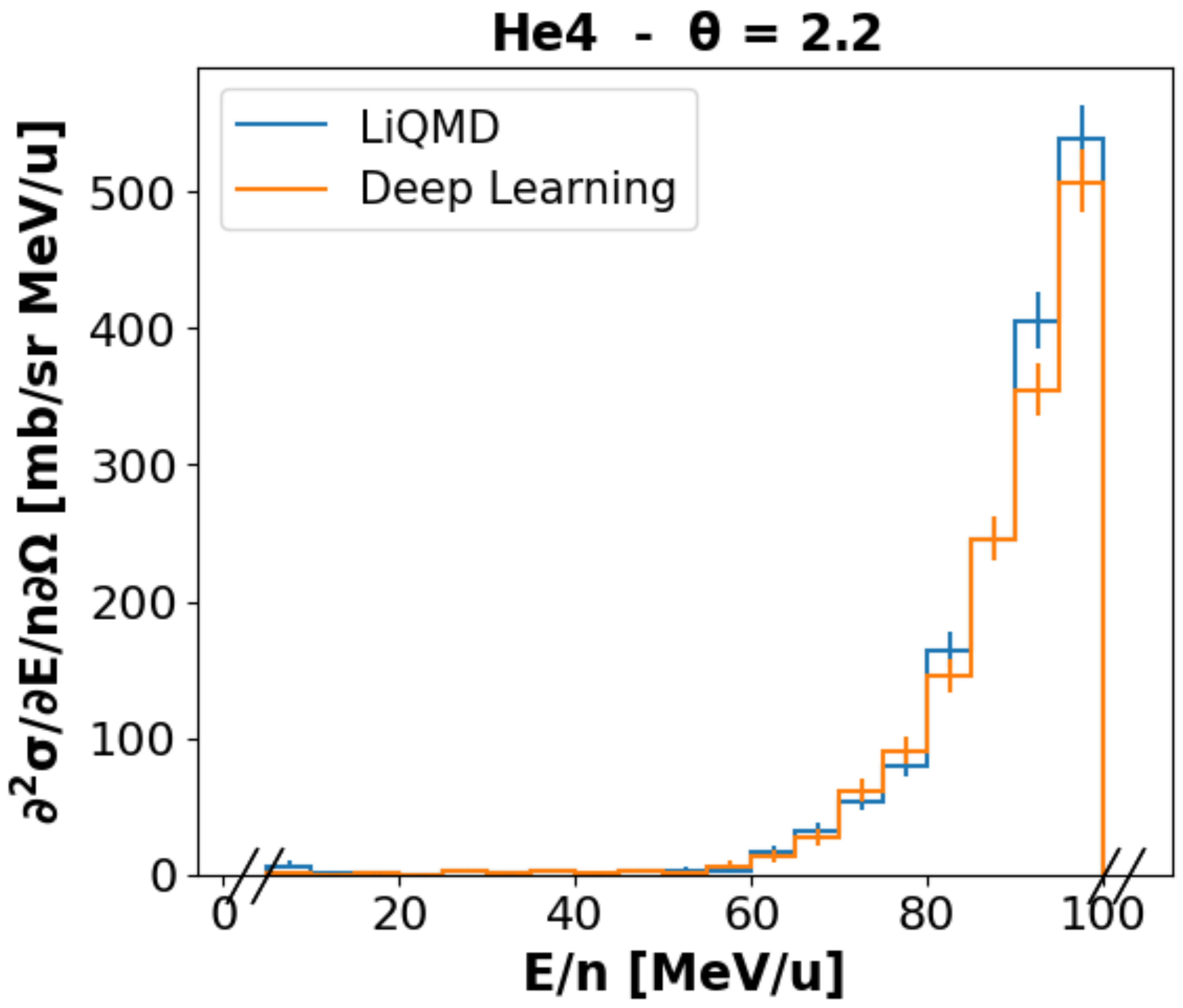
95 MeV/u





C12 on C_nat

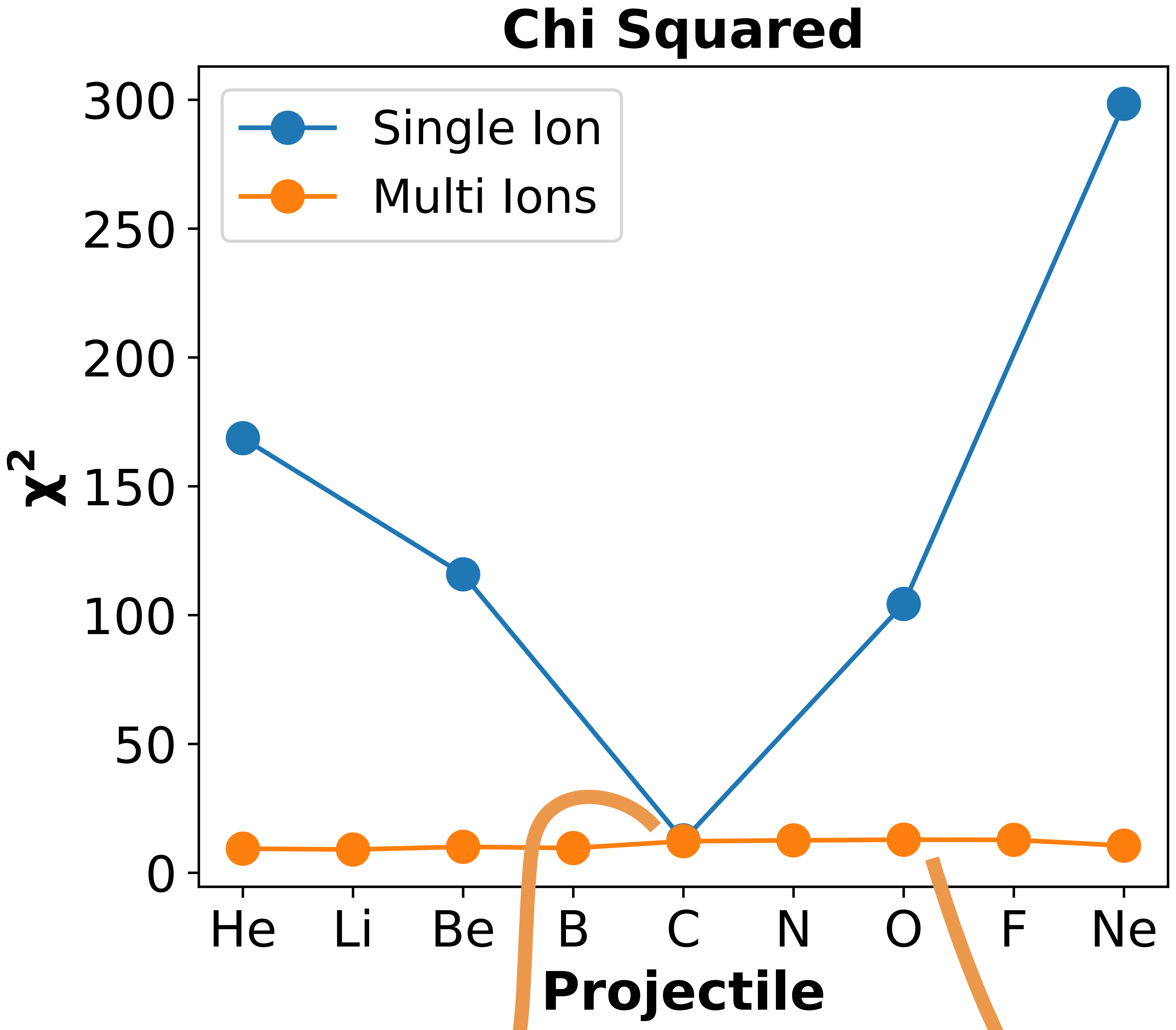
115 MeV/u



Ions

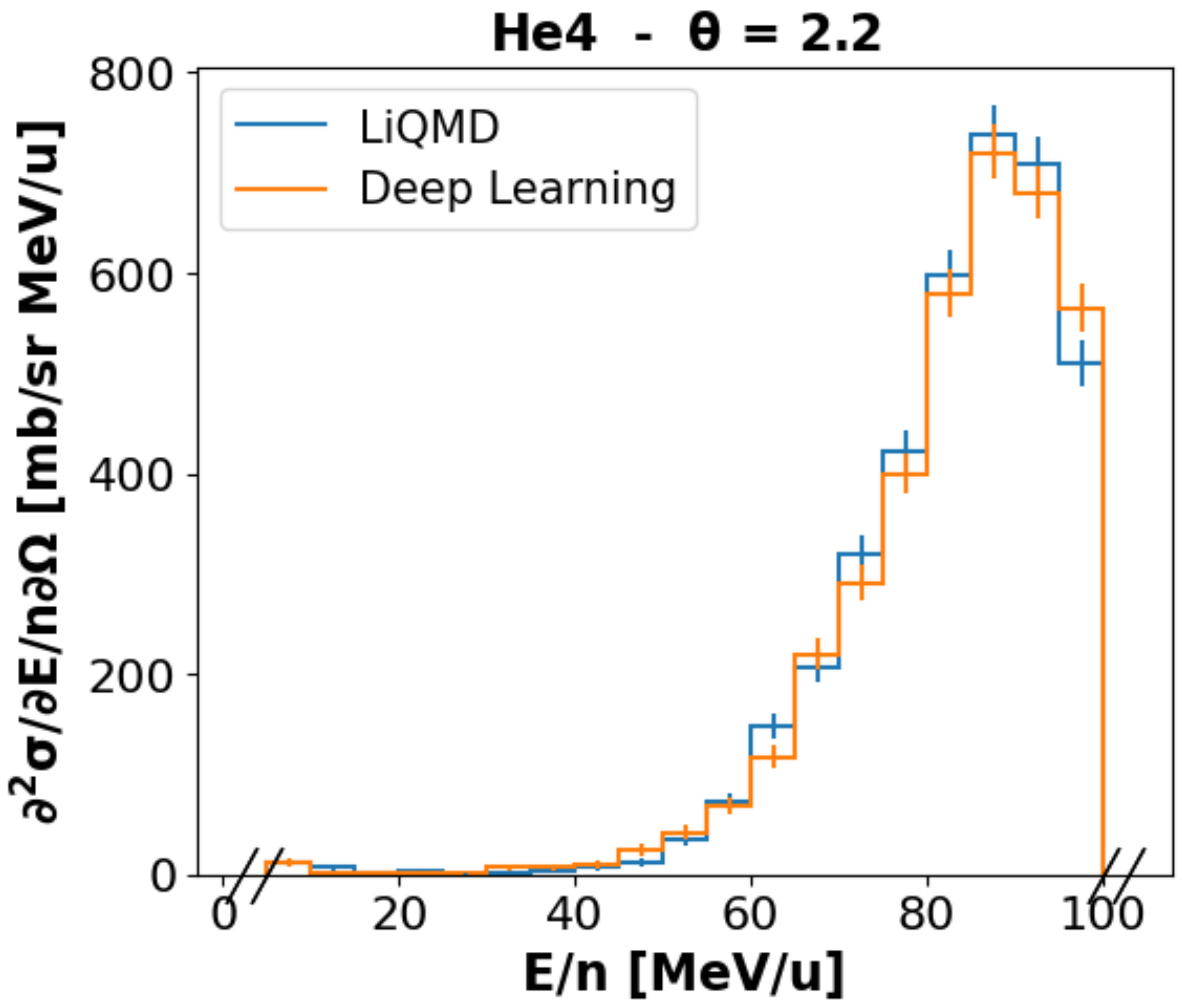
C12 on C_nat

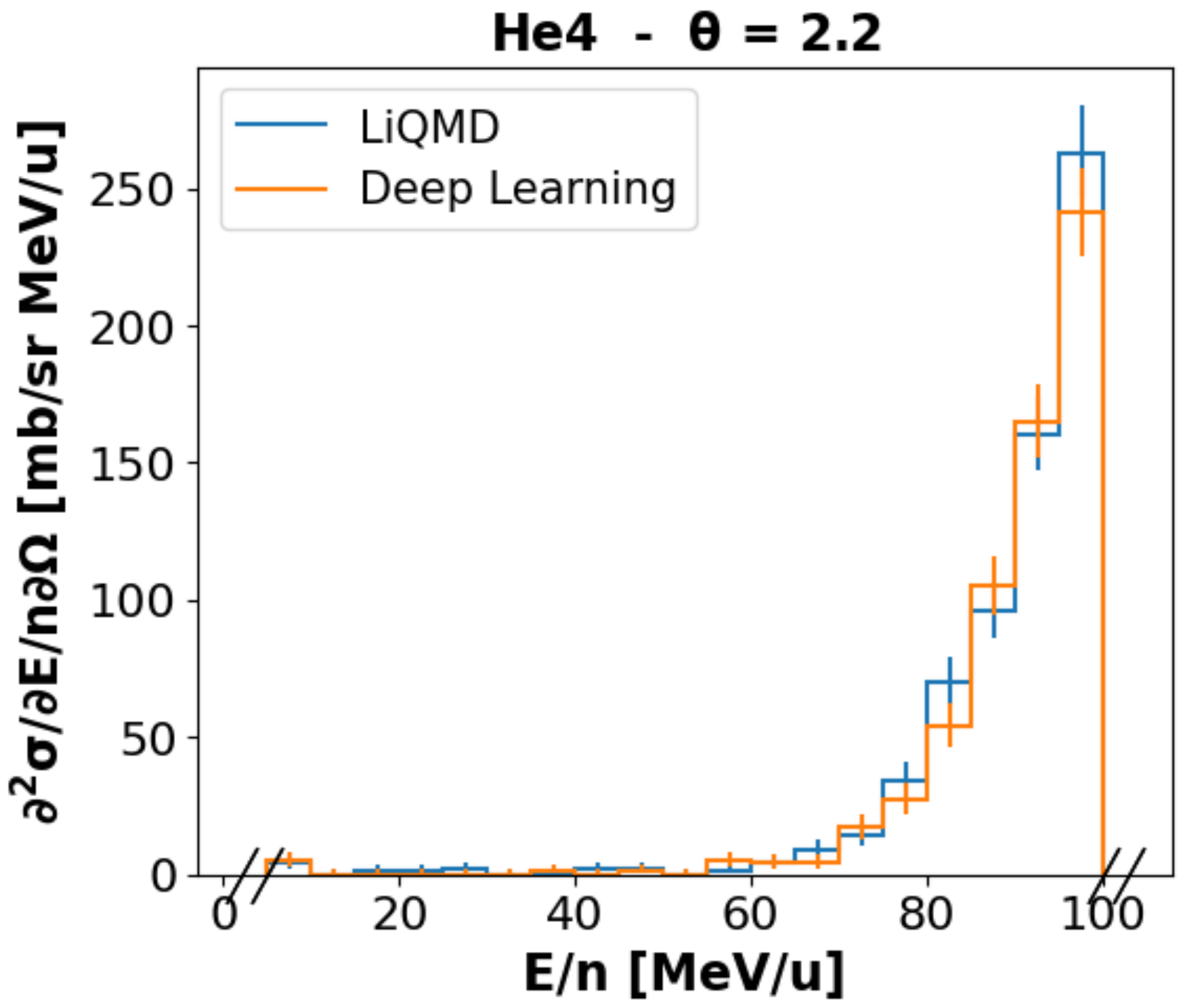
95 MeV/u

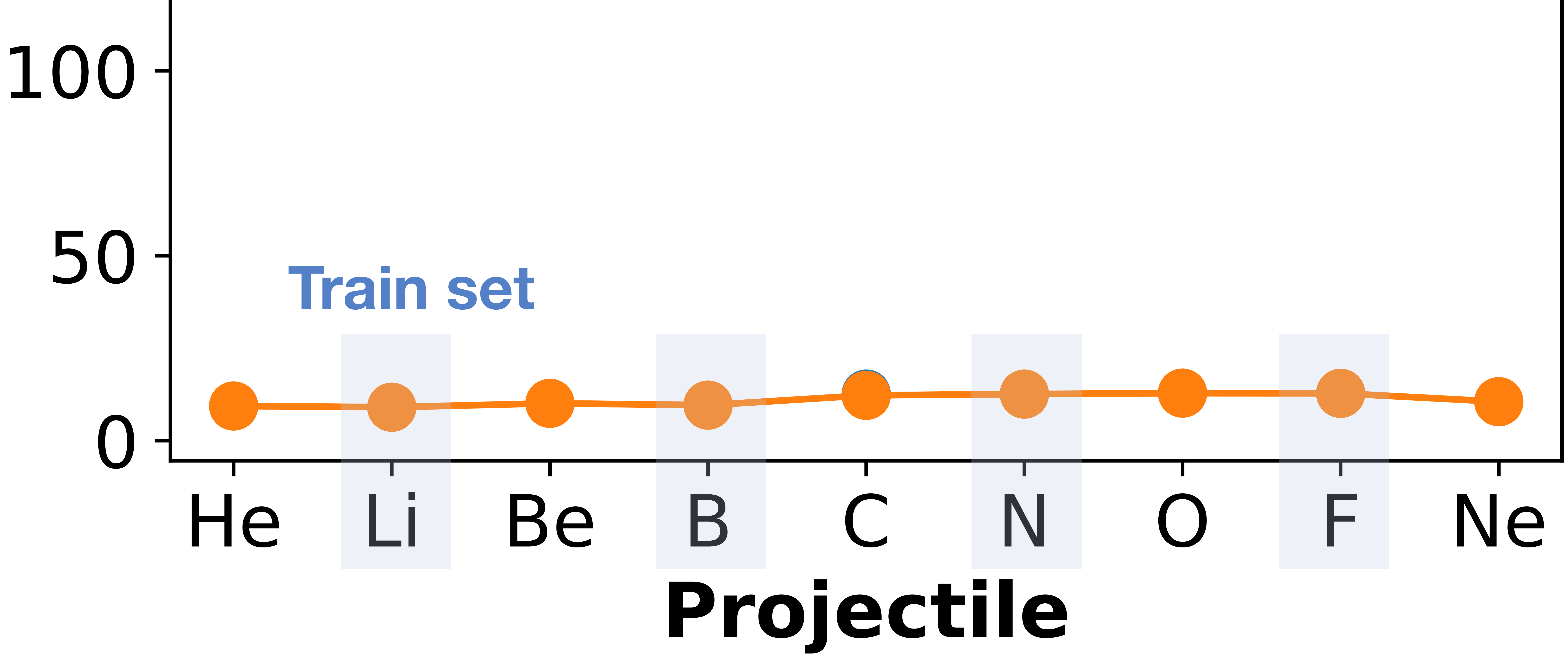


O16 on C_nat

95 MeV/u







- Training done on a subset of ions, with relatively **few example** each (~1k runs)
- Easily **extendible** to any set of ions

Next steps

Code speed-up

Leveraging GPU acceleration

Using NVIDIA TensorRT performance optimization

Current implementation (on CPU) is slower

Speed Up Inference by 36X

Optimize Inference Performance

Accelerate Every Workload

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

Possible
4X-7X speed-up

Next steps

Code speed-up

Leveraging GPU acceleration

Using NVIDIA TensorRT performance optimization

Extension to BLOB

Next steps

Code speed-up

Leveraging GPU acceleration

Using NVIDIA TensorRT performance optimization

Extension to BLOB

QMD and LiQMD Optimisation

Fully differentiable pipeline:

Model's parameters $\xrightarrow{\text{DL}}$ Cross Sections

Gradient based optimization

Emulating de-excitation model

Thank you for your attention!

- Nuclear interaction models in Geant4:
  - Sophisticated models are slow
  - No dedicated model under 100 MeV/u
- Deep Learning approach for model emulation
  - Emulation of Hamiltonian derivatives with DL for QMD
  - Multi ion training to achieve generalization
  - Possible model optimization or speed-up