Neural-Network-based Surrogate Simulator for Particle Accelerator with High Dimensional Control Settings

Emmanuel Goutierre / Hayg Guler

H. Guler[1], C. Bruni[1]
J. Cohen[2], M. Sebag[2]

[1]Laboratoire de Physique des 2 Infinis Irène Joliot-Curie (IJCLab)

[2]Laboratoire Interdisciplinaire des Sciences du Numérique (LISN)

Tuesday 28th November 2023

Motivations

Surrogate models : from aggregate to 6D

- Models are good for reproducing beam aggregate properties (size, emittance, Edep, ...)
- aggregates could be insufficient to get all properties of a beam
- by definition, full beam reconstruction gets all the beam properties
- 6D beam is perfect as input for a new simulation
- Full 6D beam helps to better understand disturbed beam
- help to better understand the beam halo
- better understand collective effects

Table of Contents

Table of Contents

How does a surrogate model work?

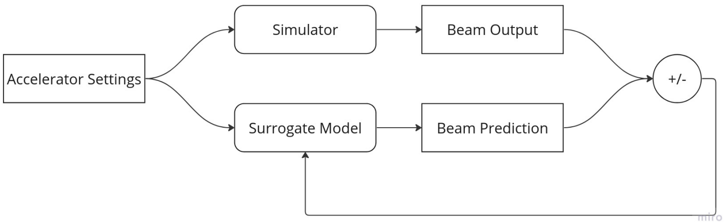


Figure: Training of a Surrogate Model

Why Surrogate Models of Particle Accelerator Simulator?

General motivation concerning the need for surrogate models for particle accelerators.

Fast Execution

- ms vs. several minutes

Optimization

- Offline & Online

Real-time Feedback

- Runnable in a control room during operations

Table of Contents

ThomX: A Compact Compton Source

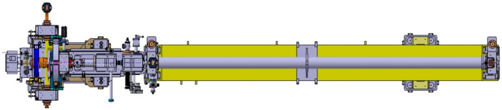


Figure: Linac of ThomX.

ThomX

- X-ray source by Compton backscattering
- Compact Accelerator $(70\mathrm{m}^2)$
- In commissioning at the IJCLab since May 2021

Linac

- Accelerate the electron beam up to 50 MeV

Goal

Use machine learning to tackle the problem of adjusting the Linac parameters to fulfill the beam requirements for the transfer line.

Accelerator Tuning

$\mathcal{A}$: Controllable Parameters

- 15 controllable parameters
  - Laser position and size
  - Gun and Cavity phase and field
  - Solenoid Fields
  - Steerer Fields
  - Quadrupoles Fields

Accelerator Tuning

$\mathcal{A}$: Controllable Parameters

- 15 controllable parameters
  - Laser position and size
  - Gun and Cavity phase and field
  - Solenoid Fields
  - Steerer Fields
  - Quadrupoles Fields

$\mathcal{B}$: Hidden Parameters

- Mechanical Misalignment
- Unknown initial particle distribution
- Slow drift of electromagnetic elements

Accelerator Tuning

$\mathcal{A}$: Controllable Parameters

- 15 controllable parameters
  - Laser position and size
  - Gun and Cavity phase and field
  - Solenoid Fields
  - Steerer Fields
  - Quadrupoles Fields

$\mathcal{O}$: Observables

- 17 Observables
  - Position and Charge at BPMs
  - Charge at ICTs
  - Position and Size at Screen
  - Charge at Faraday Cup

$\mathcal{B}$: Hidden Parameters

- Mechanical Misalignment
- Unknown initial particle distribution
- Slow drift of electromagnetic elements

Accelerator Tuning

$\mathcal{A}$: Controllable Parameters

- 15 controllable parameters
  - Laser position and size
  - Gun and Cavity phase and field
  - Solenoid Fields
  - Steerer Fields
  - Quadrupoles Fields

$\mathcal{B}$: Hidden Parameters

- Mechanical Misalignment
- Unknown initial particle distribution
- Slow drift of electromagnetic elements

$\mathcal{O}$: Observables

- 17 Observables
  - Position and Charge at BPMs
  - Charge at ICTs
  - Position and Size at Screen
  - Charge at Faraday Cup

F: Objective function

- Quality of the beam
- Function of (A, B)

Accelerator Tuning

$\mathcal{A}$: Controllable Parameters

- 15 controllable parameters
  - Laser position and size
  - Gun and Cavity phase and field
  - Solenoid Fields
  - Steerer Fields
  - Quadrupoles Fields

$\mathcal{B}$: Hidden Parameters

- Mechanical Misalignment
- Unknown initial particle distribution
- Slow drift of electromagnetic elements

$\mathcal{O}$: Observables

- 17 Observables
  - Position and Charge at BPMs
  - Charge at ICTs
  - Position and Size at Screen
  - Charge at Faraday Cup

F: Objective function

- Quality of the beam
- Function of (A, B)

Goal

- Optimize : find set of **parameters** (A) depending on **hidden parameters** (B) to get minimal **objective function** (F) with the aid of **observable**$(\mathcal{O})$
- Classical way : manual tuning, heavy load on expert

Table of Contents

Table of Contents

Multi-Layer Perceptron: A First Model

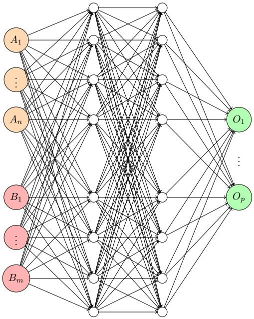


Figure: MLP as a surrogate model of a Linac

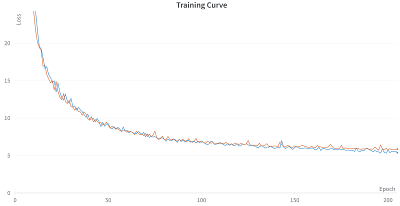


Figure: Training Curve

Multi Layer Perceptron

- Stack all inputs and outputs
- 10k simulations sampling $\mathcal{A}$ and $\mathcal{B}$
- Minimization of the L2 loss

Physics-aware: Cutting the non-causal links

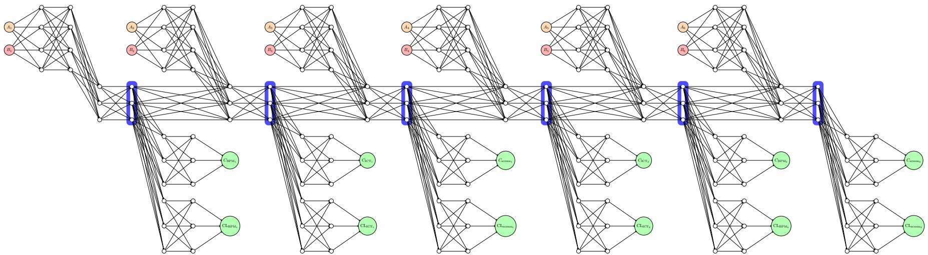


Figure: LinacNet with 6 modules corresponding to 6 diagnostic stations on the Linac

LinacNet

- Split input and output according to their position in the Linac
- Neural Network Architecture reflecting a Linac architecture
- Each Module models one Diagnostic (could be real or virtual)

Table of Contents

PointNet as a Beam Representation Network

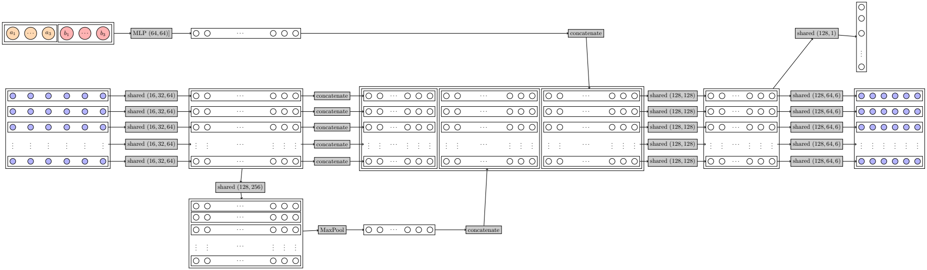


Figure: One module of ThomNet

ThomNet

- Track the full distribution of particles
- Inspired by Qi et al., "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation" (CVPR 2017)

PointNet as a Beam Representation Network

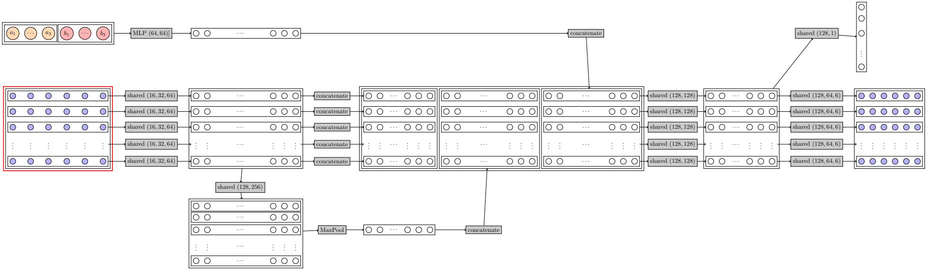


Figure: One module of ThomNet

ThomNet

- Track the full distribution of particles
- Inspired by Qi et al., "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation" (CVPR 2017)

PointNet as a Beam Representation Network

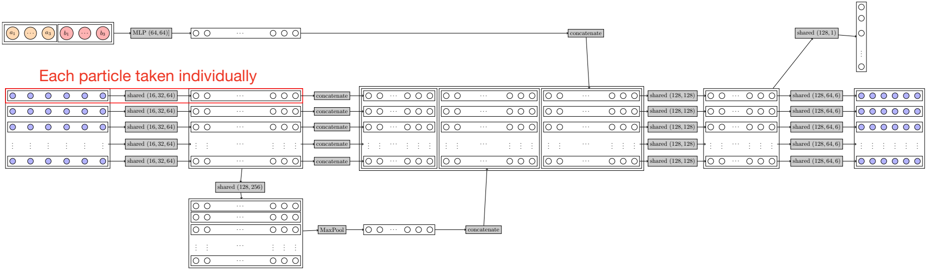


Figure: One module of ThomNet

ThomNet

- Track the full distribution of particles
- Inspired by Qi et al., "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation" (CVPR 2017)

PointNet as a Beam Representation Network

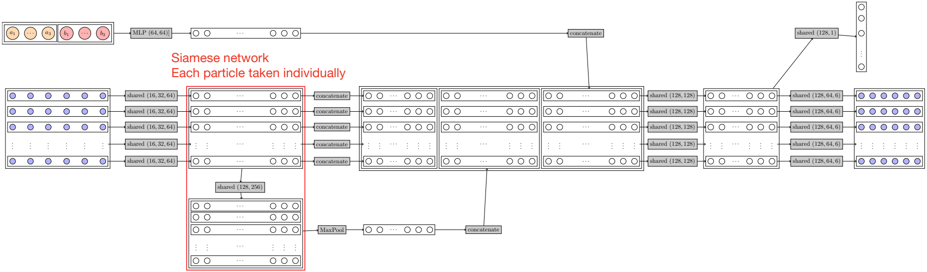


Figure: One module of ThomNet

ThomNet

- Track the full distribution of particles
- Inspired by Qi et al., "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation" (CVPR 2017)

PointNet as a Beam Representation Network

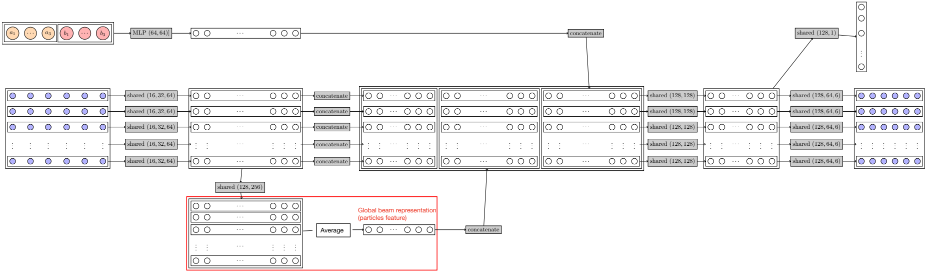


Figure: One module of ThomNet

ThomNet

- Track the full distribution of particles
- Inspired by Qi et al., "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation" (CVPR 2017)

PointNet as a Beam Representation Network

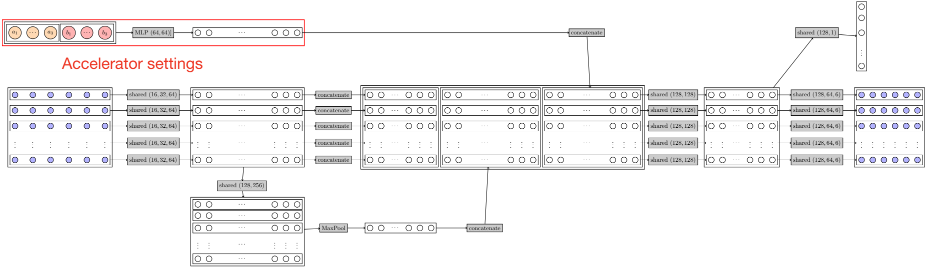


Figure: One module of ThomNet

ThomNet

- Track the full distribution of particles
- Inspired by Qi et al., "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation" (CVPR 2017)

PointNet as a Beam Representation Network

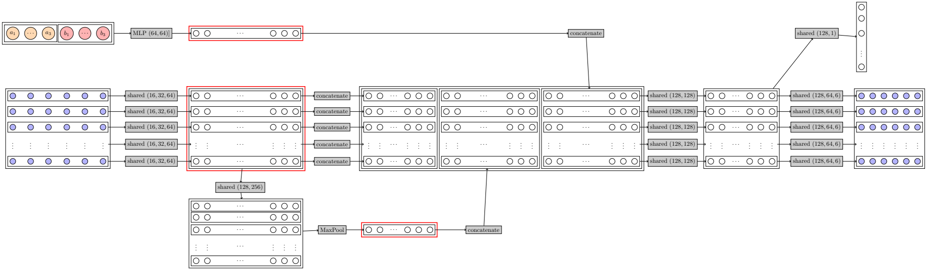


Figure: One module of ThomNet

ThomNet

- Track the full distribution of particles
- Inspired by Qi et al., "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation" (CVPR 2017)

PointNet as a Beam Representation Network

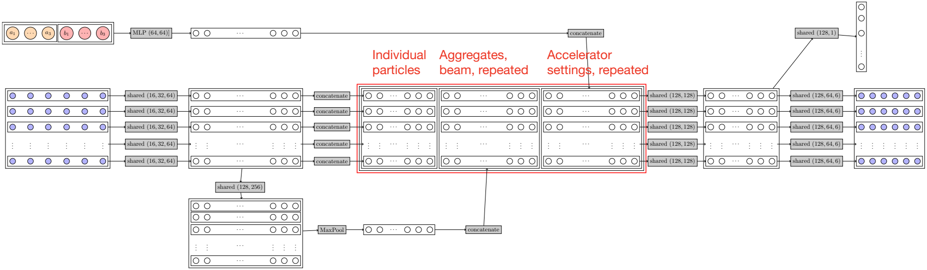


Figure: One module of ThomNet

ThomNet

- Track the full distribution of particles
- Inspired by Qi et al., "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation" (CVPR 2017)

PointNet as a Beam Representation Network

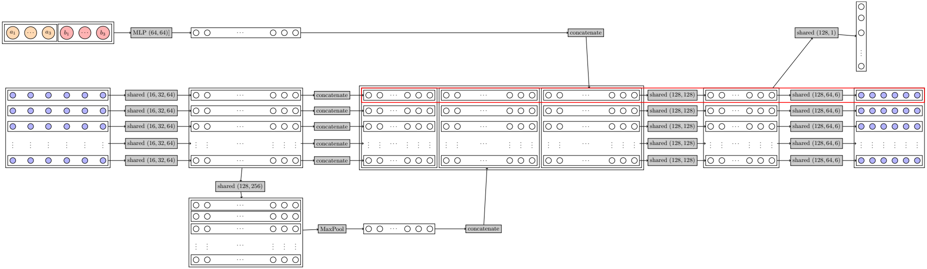


Figure: One module of ThomNet

ThomNet

- Track the full distribution of particles
- Inspired by Qi et al., "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation" (CVPR 2017)

PointNet as a Beam Representation Network

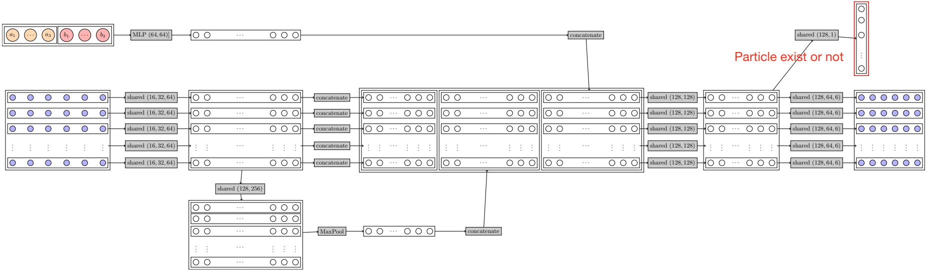


Figure: One module of ThomNet

ThomNet

- Track the full distribution of particles
- Inspired by Qi et al., "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation" (CVPR 2017)

PointNet as a Beam Representation Network

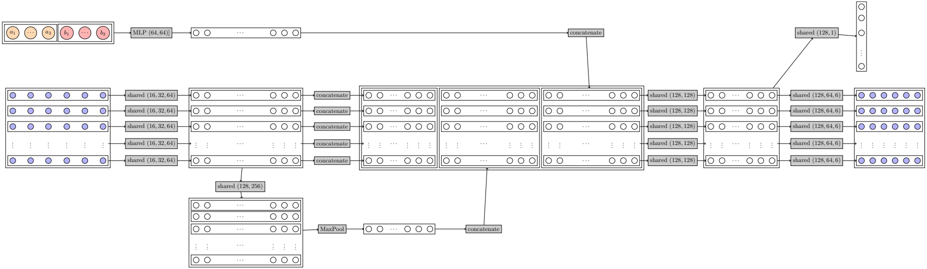


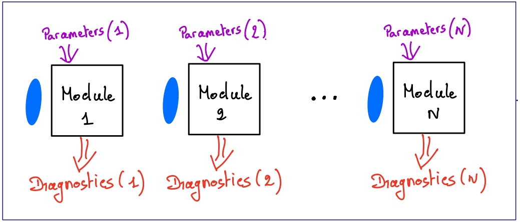
Figure: One module of ThomNet

ThomNet

- Track the full distribution of particles
- Inspired by Qi et al., "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation" (CVPR 2017)

Table of Contents

Accelerator as a sequence of modules



- We divide our accelerator in a sequence of sub-parts
- Each part could contain controls / measurements (real or virtual)
- Learning a full machine could be complicated, costly

Good for

- transfer to a real machine
- optimize : could be done by part / module
- Retrain locally due to drift in the data
- Address larger machines

Sequential Network as a Multi-Objective Optimization

- General question in machine learning
- how to learn a sequence of models, tasks ?
- Could be heterogeneous : classification, regression, etc
- Conflicting between modules could deteriorate the global loss

- Independent Errors : $Err_{i,i+1}\left(d_i, d_{i+1}, a; \theta\right) = l\left(f_{i,i+1}\left(d_i, a; \theta\right), d_{i+1}\right)$
- End-to-End Errors : $Err_{0,i}\left(d_0, d_i, a; \theta\right) = l\left(f_{0,i}\left(d_0, a; \theta\right), d_i\right)$

Scalarization of the Multi-Objective Loss

$$\mathcal{L}_w\left(d, a; \theta\right) = \sum_{i=1}^{N} w_{i-1,i} Err_{i-1,i}\left(d_{i-1}, d_i, a; \theta\right) + w_{0,i} Err_{0,i}\left(d_0, d_i, a; \theta\right)$$

One example of learning a sequence : MGDA[1]

- Dynamic weighting of the module that moderates conflicting loss between modules

$$w^* = \arg\min_w \mathcal{L}_w, \qquad w > 0, \qquad \sum_{i=1}^{N} w_{i-1,i} + w_{0,i} = 1$$

Properties

- Common descent direction to all objectives
- Stop when encountering a Pareto-invariant point

[1] Sener and Koltun, "Multi-Task Learning as Multi-Objective Optimization".

Table of Contents

Numerical results

Memory Constraints

- GPU A100 80 GB
- Max Available Batch size
  - One segment: 200
  - All segments: 16

Training Time

Limitation to:

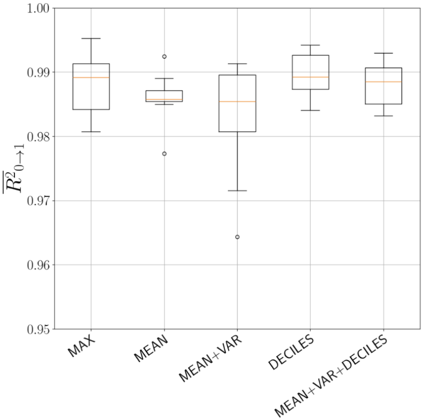
- 20 h for one segment
- 100 h for all segments

Method

- Extensive hyperparameter search on the first segment.
- Apply these parameters to all segments.
- Only Multi-Objective Strategy Study on the whole accelerator.

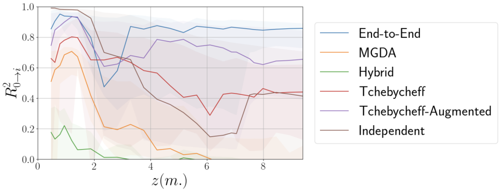
Training settings

- Aggregation methods compared
- Max works a bit better than mean

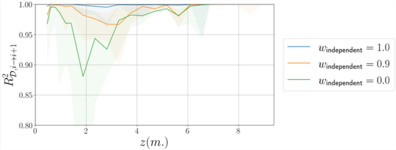


Multi task learning

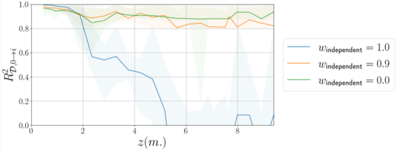
- Different methods compared
- End2End with dedicated weights works the best



Numerical results



Particle Independent Errors for Each Segment



Particle End-to-End Modeling Performance

Independent Modeling: Cumulative Errors explode.

End-to-end Modeling: Decrease Independent Performance.

Combined Modeling: Small value of end-to-end loss ($w_{\text{end-to-end}} = 0.1$) regularizes the end-to-end error.

Numerical Results

The best model achieves results comparable with the diagnostic station accuracy.

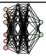
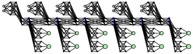
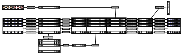
Architecture	BPM	ICT	YAG	ICT	BPM	YAG
FeedForward 	$776\mu m$	$1084\mu m$	$1692\mu m$	$1106\mu m$	$1261\mu m$	$1554\mu m$
LinacNet 	$198\mu m$	$254\mu m$	$541\mu m$	$618\mu m$	$719\mu m$	$913\mu m$
ThomNet 	$\mathbf{178\mu m}$	$\mathbf{134\mu m}$	$\mathbf{247\mu m}$	$\mathbf{224\mu m}$	$\mathbf{258\mu m}$	$\mathbf{336\mu m}$

Table: MAE of the position. The accuracy of the BPM is $\sim 100\mu m$

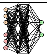
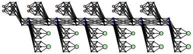
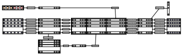
Architecture	BPM	ICT	YAG	ICT	BPM	YAG
FeedForward 	$176pC$	$177pC$	$167pC$	$91pC$	$91pC$	$91pC$
LinacNet 	$28pC$	$28pC$	$29pC$	$34pC$	$34pC$	$35pC$
ThomNet 	$\mathbf{8pC}$	$\mathbf{9pC}$	$\mathbf{9pC}$	$\mathbf{8pC}$	$\mathbf{8pC}$	$\mathbf{8pC}$

Table: MAE of the charge. The accuracy of the ICT is $\sim 10pC$

Distributions

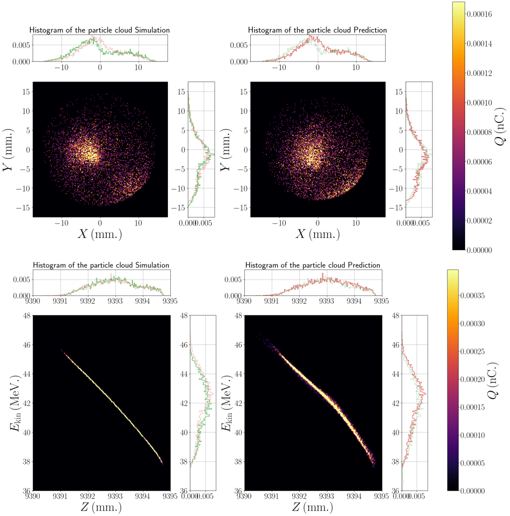


Figure: Comparison between the projection of the simulated beam (left) and predicted beam (right) on the transverse and longitudinal space.

Table of Contents

Perspectives

Results

- Reflecting the physical constraints in the neural architecture speed up the training and gives better results
- Precision of the same orders as the diagnostics installed on ThomX

Challenges

- Training of a modular model
- Performance for the optimization task to be tested

Questions?