RECONSTRUCTION IN A 3D PLASTIC SCINTILLATOR DETECTOR USING DEEP LEARNING

Saúl Alonso-Monsalve

ETH Zurich

IPA-ML

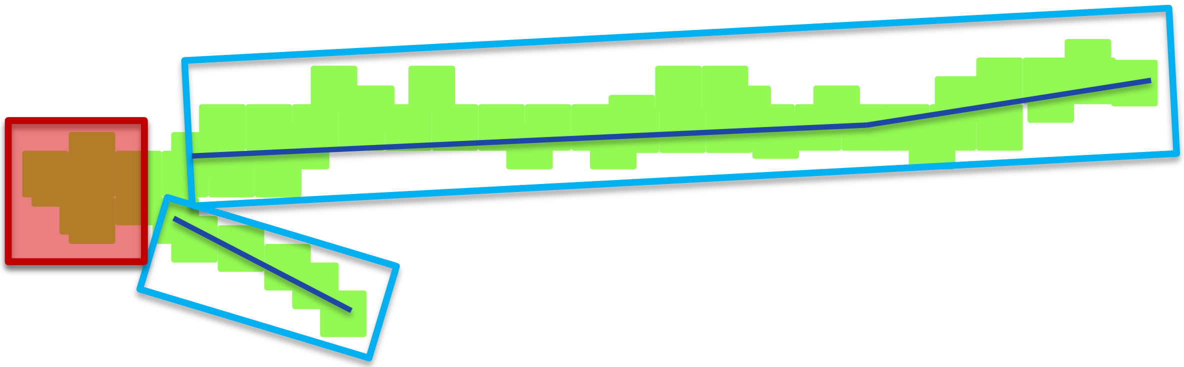
21 March 2023

Reconstruction approach

- Goal: Develop an analysis strategy that does not depend on the ν interaction model:

1. Algorithms to reject noise and identify single vs multi-primary-particle hits.
    - Graph neural networks and Submanifold sparse convolutional networks.
    - Publication: doi:10.1103/PhysRevD.103.032005.

2. Algorithm to perform fitting on single-particle objects.
    - RNN/Transformer, drastically improving track fitting performance.
    - Publication: arXiv:2211.04890.



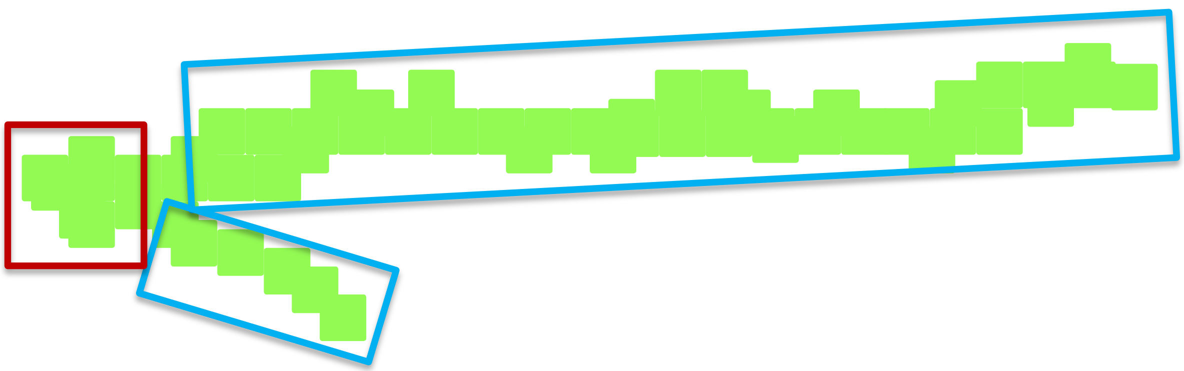
3. Algorithm to extract physics parameters from vertex activity.
    - GAN for fast simulation, used for fitting particles in the vertex activity.

Approach

1. Hit identification.
2. Particle trajectory fitting.
3. Vertex activity fitting.

Approach

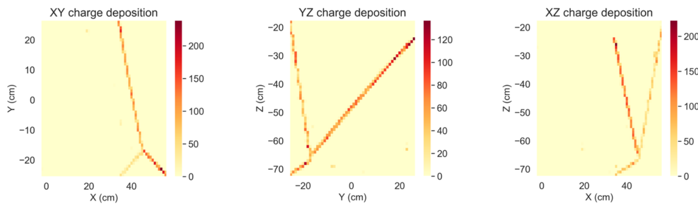
1. **Hit identification.**
2. Particle trajectory fitting.
3. Vertex activity fitting.

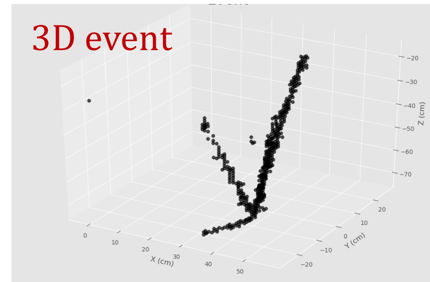


Hit identification: noise rejection

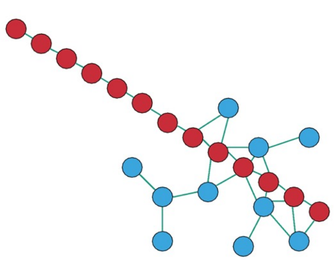
- Matching the common axis 2-to-2 in the three views XY, XZ, YZ we obtain the 3D information.

Orthogonal views.

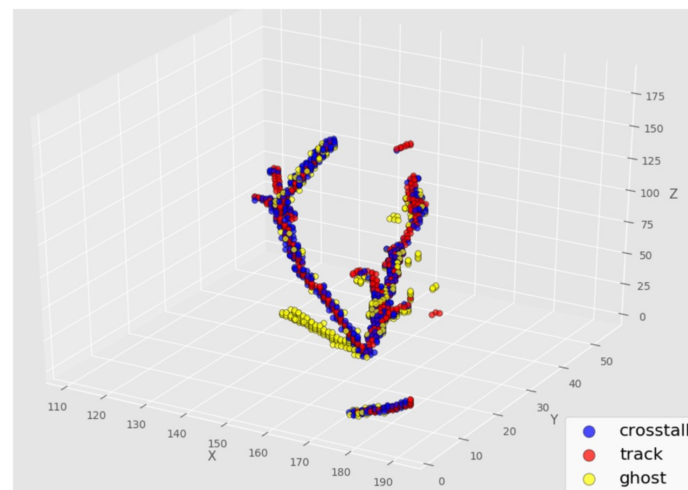




- Drawback: non-physical voxels appear due to lack of information during the 2D to 3D reconstruction algorithm, called **ghost voxels.**
- Approach: use Graph Neural Networks (GNNs) to identify and reject ghost voxels.
    - In GNNs, each node's neighbourhood defines a computation graph (in our example, each voxel is connected to all voxels within a 1.75 cm radius).
    - The algorithm chosen was GraphSAGE (*arXiv:1706.02216*).

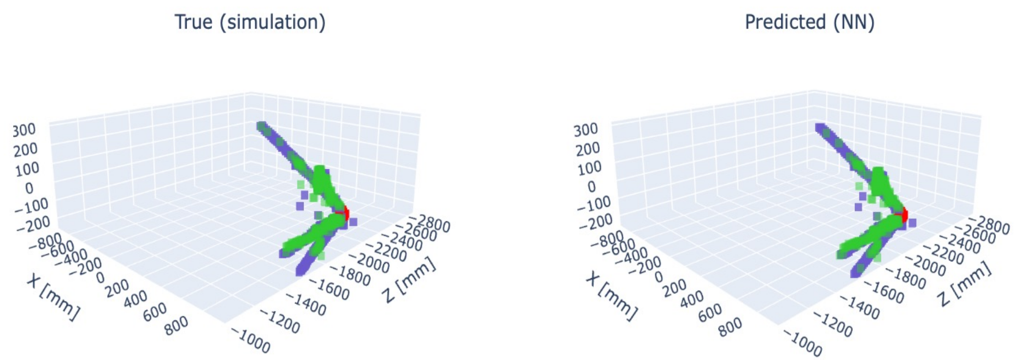


		GENIE Training		
GENIE Testing		Track	xTalk	Ghost
	Efficiency	94%	94%	88%
	Purity	96%	91%	92%
Pgun* Testing		Track	xTalk	Ghost
	Efficiency	95%	94%	85%
	Purity	98%	89%	92%

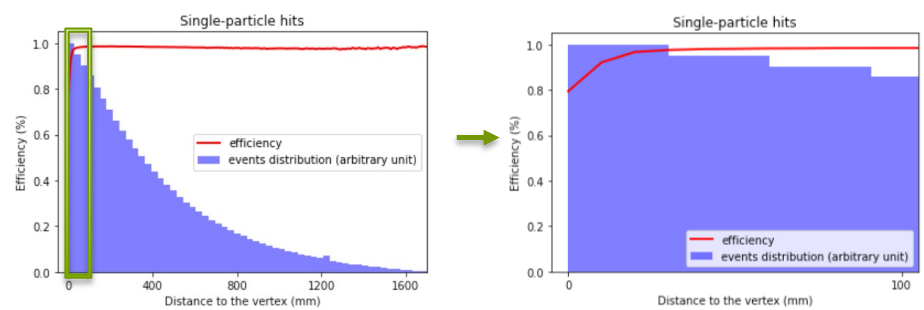


Hit identification: single vs multi-particle hits

- Classify each individual hit as:
  - **Single-particle hit**: only one particle passes through the hit cube and no other tracks pass through its adjacent cubes
  - **Multiple-particle hit**: at least two different particles pass through the hit cube and its adjacent cubes.
  - **Other**: mainly crosstalk or ghost.

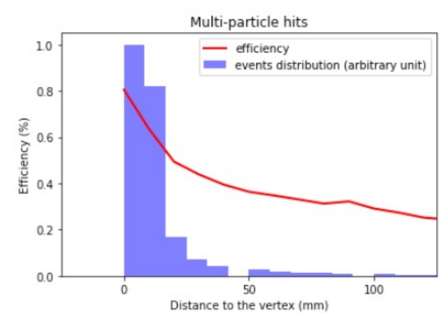


- Using a submanifold sparse U-Net-based neural network architecture (https://arxiv.org/abs/1706.01307).
  - More computationally efficient than standard CNNs.



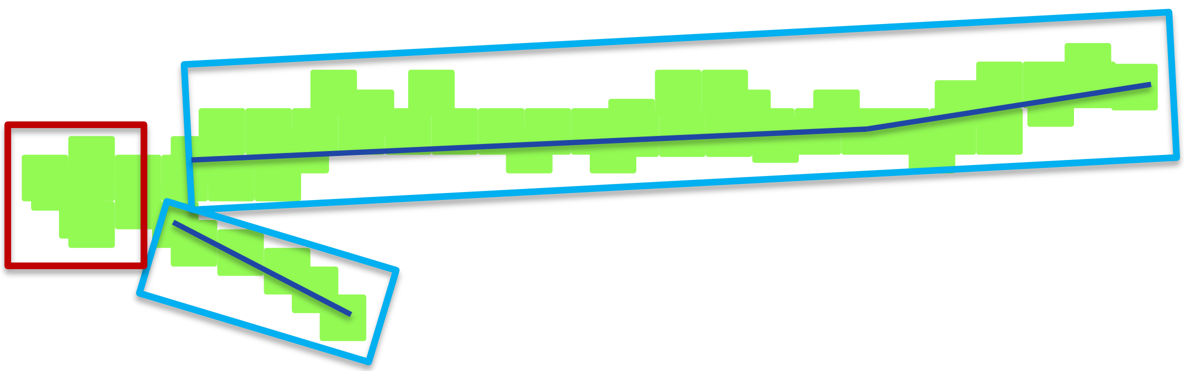
- Efficiencies:

	True multiple-particle hit	True single-particle hit	True other
Pred. multiple-particle hit	**0.7777**	0.1511	0.0711
Pred. single-particle hit	0.0055	**0.9654**	0.0291
Pred. other	0.0079	0.0479	**0.9442**



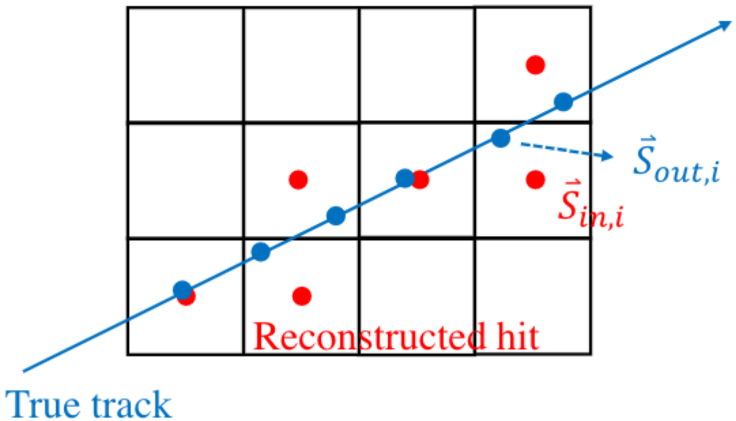
Approach

1. Hit identification.
2. **Particle trajectory fitting.**
3. Vertex activity fitting.



Fitting of the particle trajectory

- The next step is to predict the trajectory of particles based on single-particle hit information.
- For each state we consider 3D position, and energy deposition of the hit.

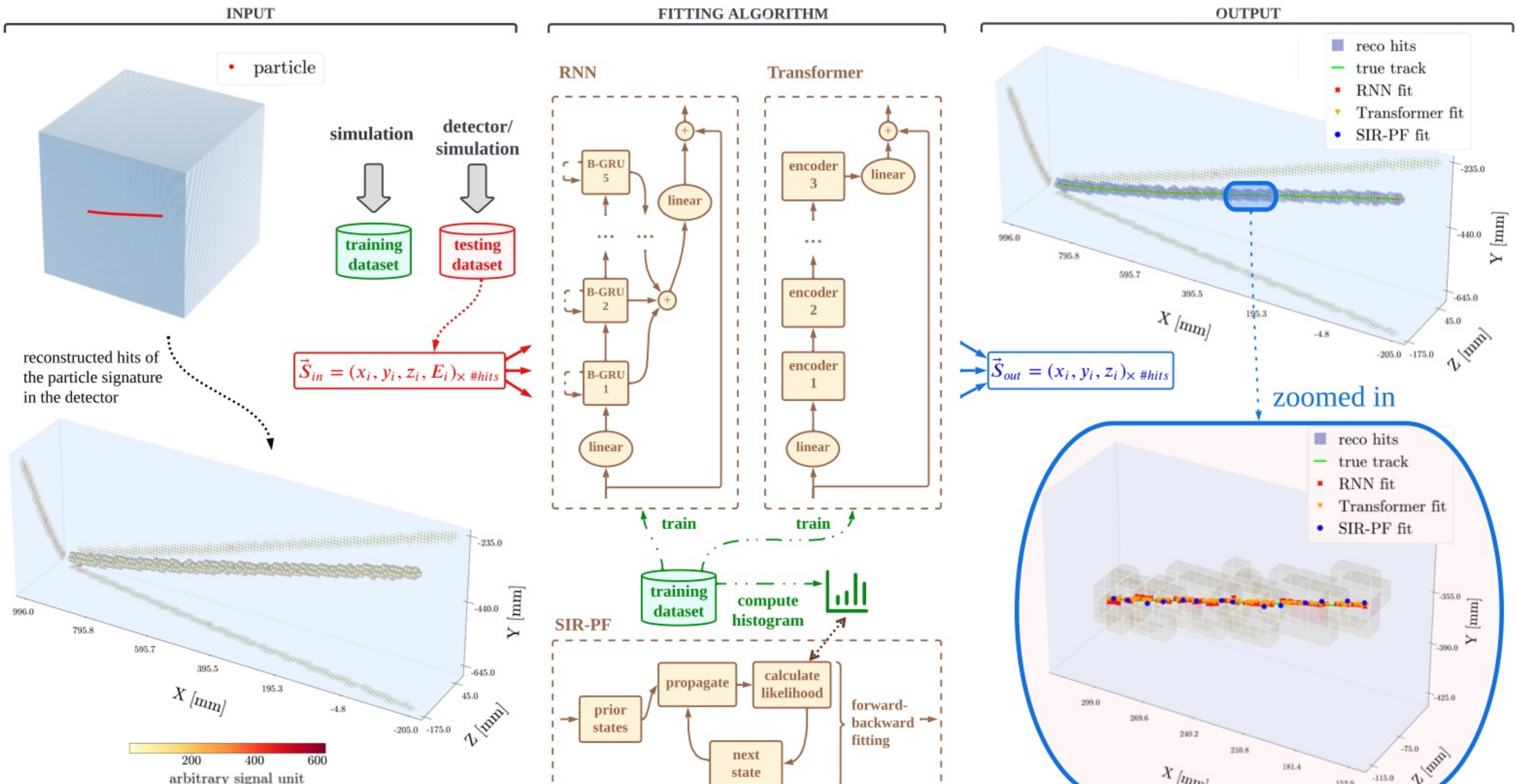


- Input hit state: $\vec{S}_{in,i} = (x_i, y_i, z_i, t_i, E_i), i = 1, \cdots, N$.
- Output node state: $\vec{S}_{out,i} = (x_i, y_i, z_i, t_i, E_i), i = 1, \cdots$
- Use neural network to construct the map:

$$\{\vec{S}_{in,i}\} \to \{\vec{S}_{out,i}\}$$

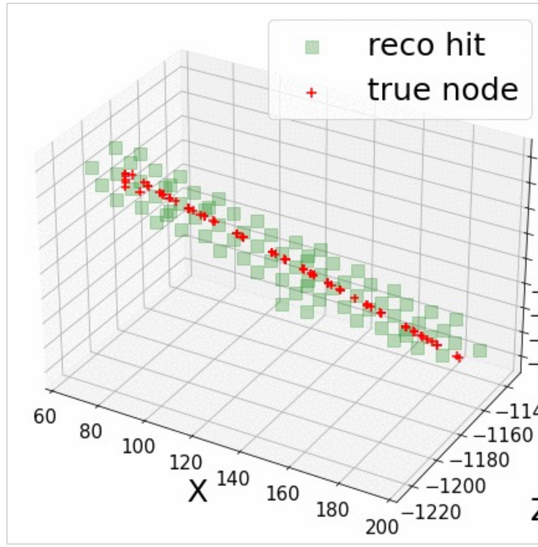
- Implemented a recurrent neural network (RNN), a Transformer (encoder), and a sequential-importance-resampling particle filter (SIR-PF).
  - We treat each particle as a sequence of hits, benefiting from the success of RNN and Transformer in Natural Language Processing (NLP).

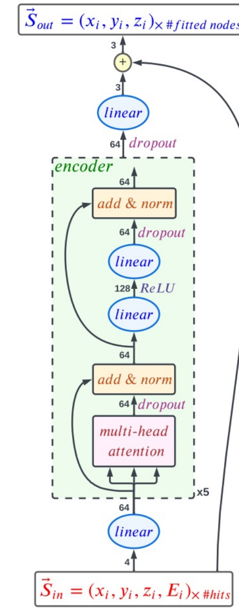
Workflow



INPUT
FITTING ALGORITHM
OUTPUT
particle
simulation
detector/ simulation
training dataset
testing dataset
reconstructed hits of the particle signature in the detector
$\vec{S}_{in} = (x_i, y_i, z_i, E_i)_{\times \#hits}$
arbitrary signal unit
RNN
Transformer
B-GRU 5
B-GRU 2
B-GRU 1
linear
encoder 3
encoder 2
encoder 1
train
training dataset
compute histogram
SIR-PF
prior states
propagate
calculate likelihood
next state
forward-backward fitting
$\vec{S}_{out} = (x_i, y_i, z_i)_{\times \#hits}$
reco hits
true track
RNN fit
Transformer fit
SIR-PF fit
zoomed in
X [mm]
Y [mm]
Z [mm]

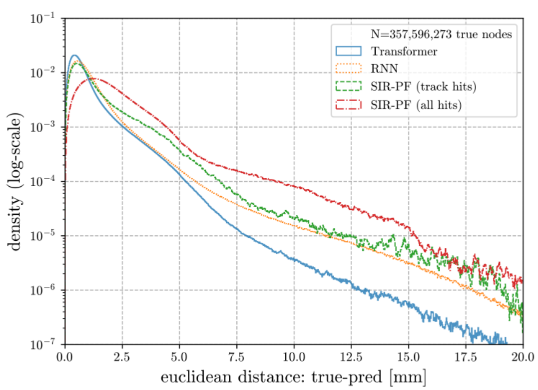
Details

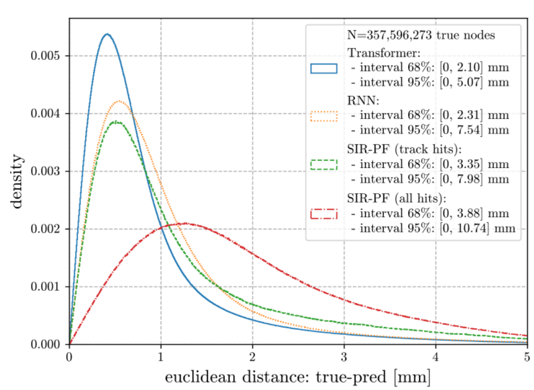


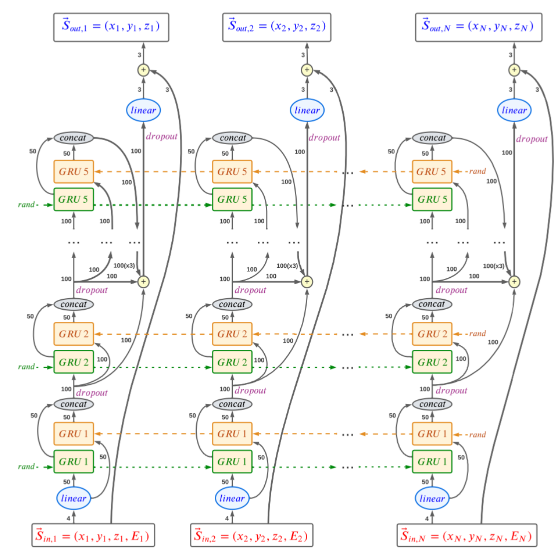


- Each algorithm outputs the fitted 3D trajectory point for each input hit.
  - SIR-PF: first reconstructed hit as prior, sample propagation through the following 15 hits. The likelikhood calculating relies on precomputed 5-dimensional histogram.
  - RNN: five bi-directional GRU layers, 50 hidden units each, .
  - Transformer: 5 transformer-encoder layers, 8 heads, and hidden size of 64.

- Main results:

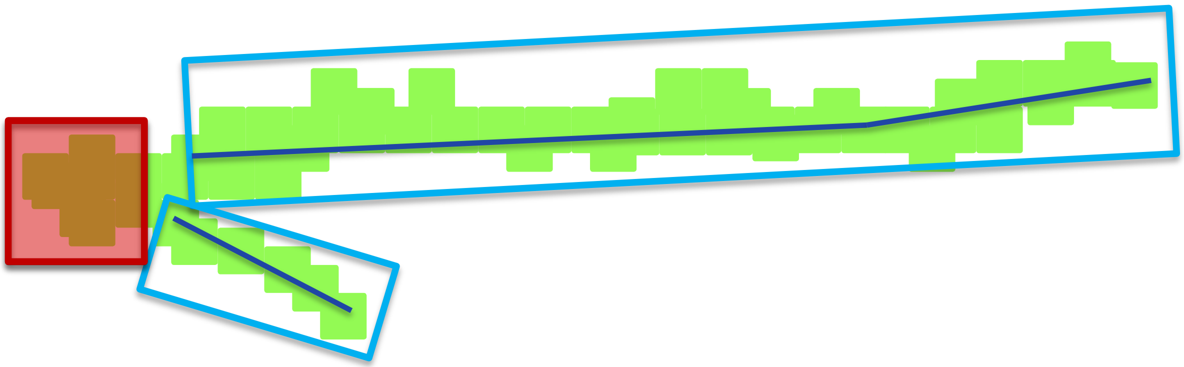






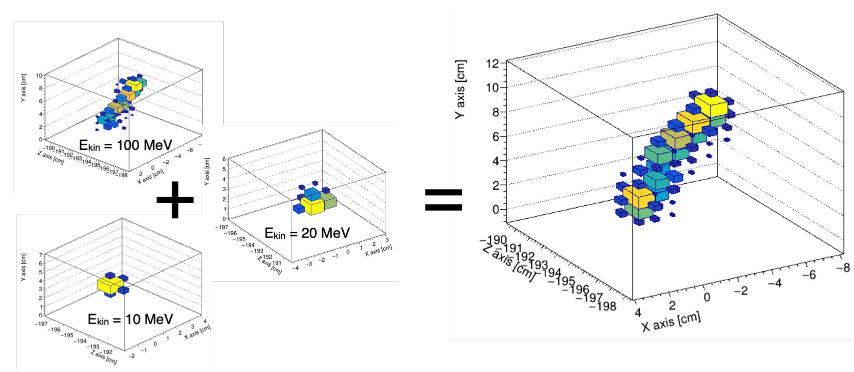
Approach

1. Hit identification.
2. Particle trajectory fitting.
3. **Vertex activity fitting.**



Vertex activity: fitting method

- Fitting method:
  - Extract the values of the parameters that define VA.
    - # of particles (mostly protons), energy, direction, vertex position.
  - Analysis Method: likelihood fitting, i.e. VA simulation is performed during the fitting.
    1. Simulate any possible combination of the VA parameters and build VA.
    2. Find the VA 3D image (e.g. SFGD hits) that "best fit" the data and find the "best-fit" parameters.



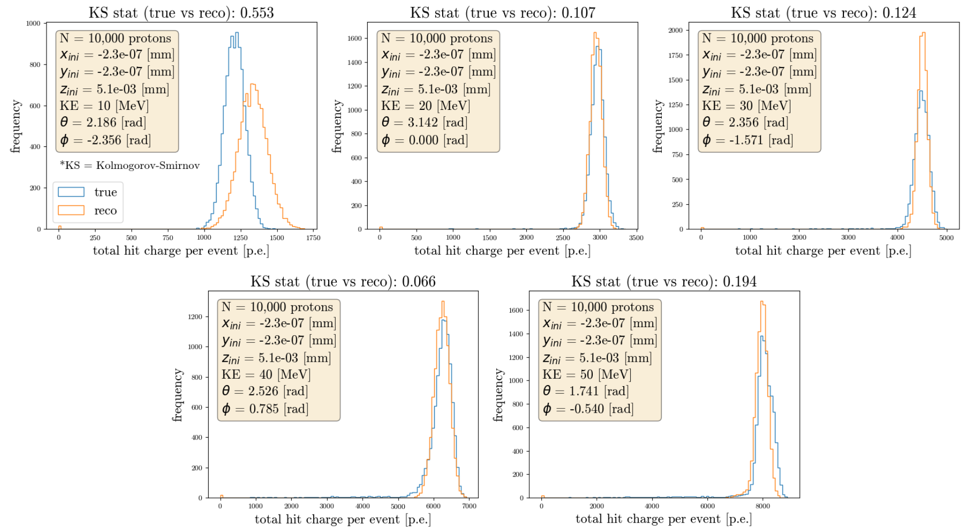
- The fitting method is highly computationally expensive.
  - Requires a large number of combinations of parameters to be simulated.
- Solution: learn a fast and accurate VA simulation using neural networks (generative models).
  - VA simulation independent of the position in SFGD, i.e. 7x7x7 cubes are enough.
  - The VA simulation becomes differentiable, meaning that physical parameters can be inferred through minimisation methods.
- We found GANs as the best trade-off between speed, generation accuracy, and ease of use.
  - Implementation of a Wasserstein GAN (WGAN) with gradient penalty.

	GANs	VAEs	Flow-based	Diffusion
Accuracy	Good	Moderate	Very Good	Excellent
Speed	Fast	Fast	Fast	Very slow
Computing Resources	Moderate/ High	Low/ Moderate	High	Low
Complexity/ Ease of Use	Low/ Moderate	Low/ Moderate	High	Low
Ecosystem Maturity	High	High	Low/ Moderate	Low/ Moderate

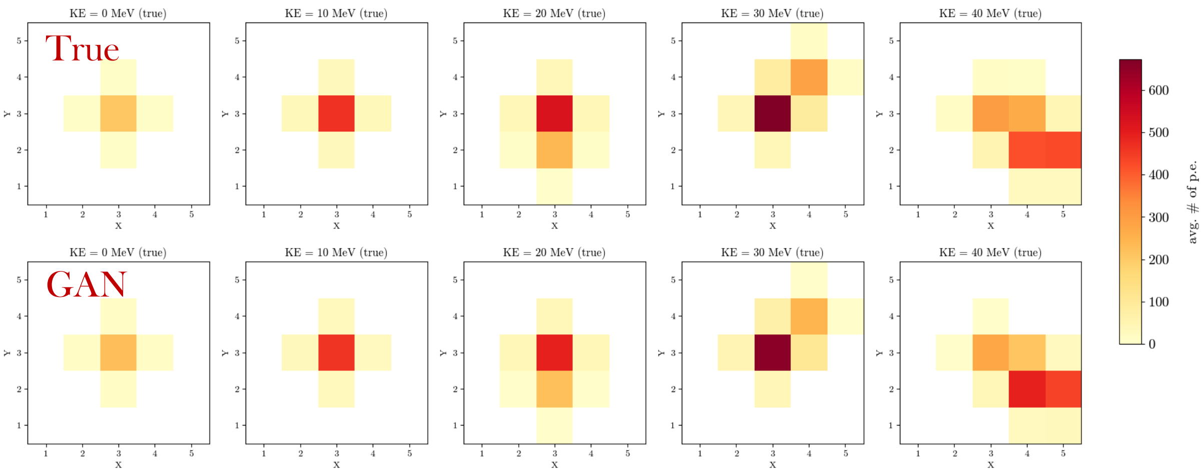
Validating the GAN generator

- Five distinct batches of samples, each with fixed initial physics parameters.
  - 10K protons each.
- We generated five batches of protons with the NN using the same physics parameters.
  - 10K protons each too.
- Allows us to understand whether the network is catching the stochasticity of the simulation.

	Number of protons	Initial KE [MeV]	Initial position [cm]	θ [rad]	φ [rad]
Testing sample 1	10K	10	$\begin{pmatrix}-2.3e^{-7},\\-2.3e^{-7},\\-5.1e^{-3}\end{pmatrix}$	2.186	-2.356
Testing sample 2	10K	20	$\begin{pmatrix}-2.3e^{-7},\\-2.3e^{-7},\\-5.1e^{-3}\end{pmatrix}$	3.142	0.000
Testing sample 3	10K	30	$\begin{pmatrix}-2.3e^{-7},\\-2.3e^{-7},\\-5.1e^{-3}\end{pmatrix}$	2.356	-1.571
Testing sample 4	10K	40	$\begin{pmatrix}-2.3e^{-7},\\-2.3e^{-7},\\-5.1e^{-3}\end{pmatrix}$	2.526	0.785
Testing sample 5	10K	50	$\begin{pmatrix}-2.3e^{-7},\\-2.3e^{-7},\\-5.1e^{-3}\end{pmatrix}$	1.741	-0.540



XY charge projection



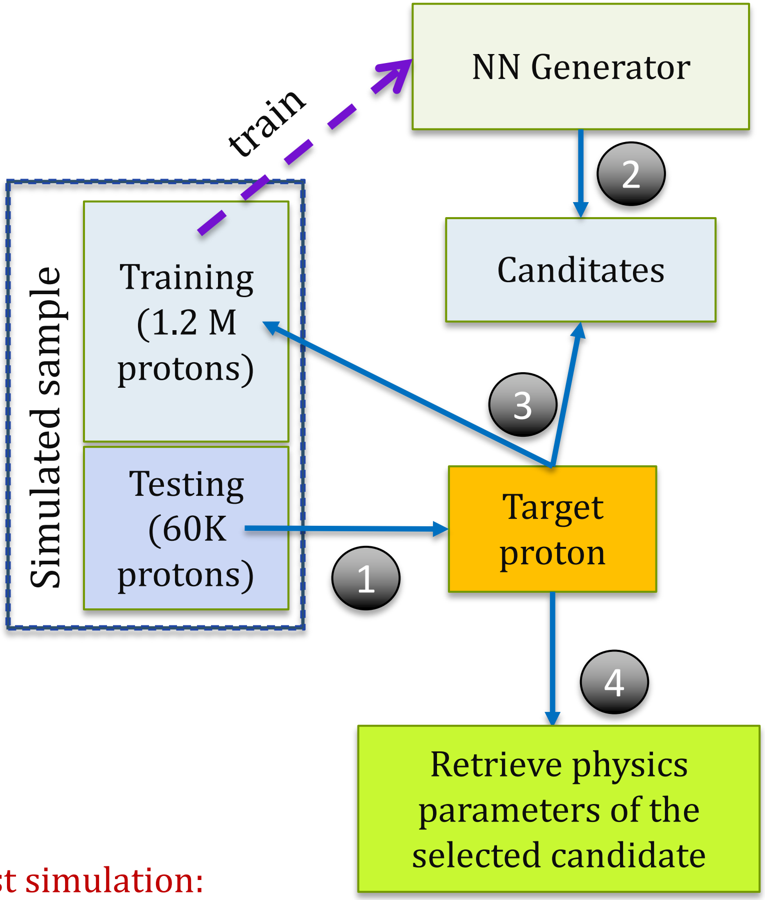
Fitting strategy

1. Select a target event not used for training.

2. Generate N random proton candidates using the NN (trained on the training sample).
   - The candidates can be stored for caching.

3. Choose a metric (e.g., chi2, MSE) to find the closest candidate to the target event.
   - Alternatively, since the NN is fully differentiable, one can run a gradient-descent approach to find the best-fit parameters.

4. Select the physical parameters of the best candidate.



* The fitting can be done for multiple events at the same time. The idea is to extend the approach to N protons.

The generator module of the GAN becomes an extremely fast simulation:

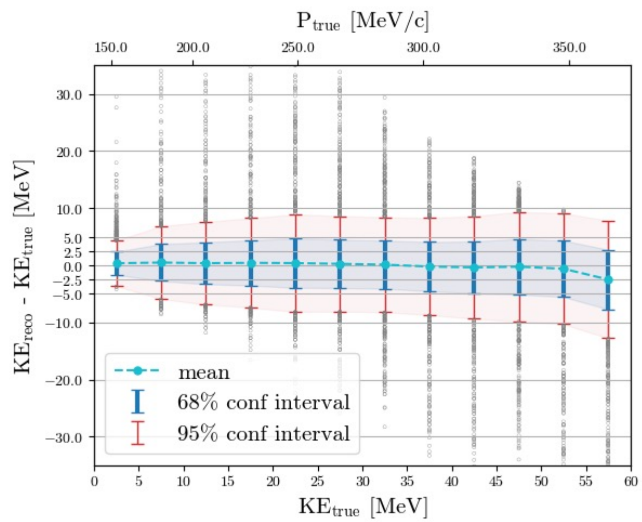
- 100K images: 3 seconds.
- 1M images: 31 seconds (>1 hour with GEANT4).
- 10 M images: ~5:30 min.
- 100 M images: <1 hour.

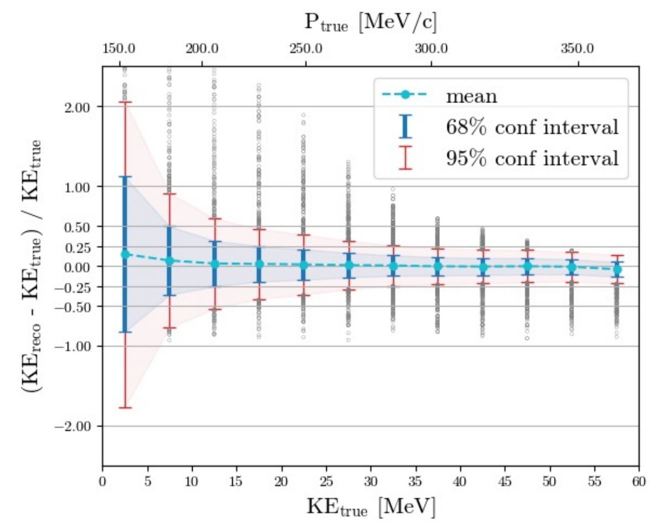
*test on an NVIDIA A100 GPU.

Fitting results (60K events, 1 proton per event)

standard
brute force

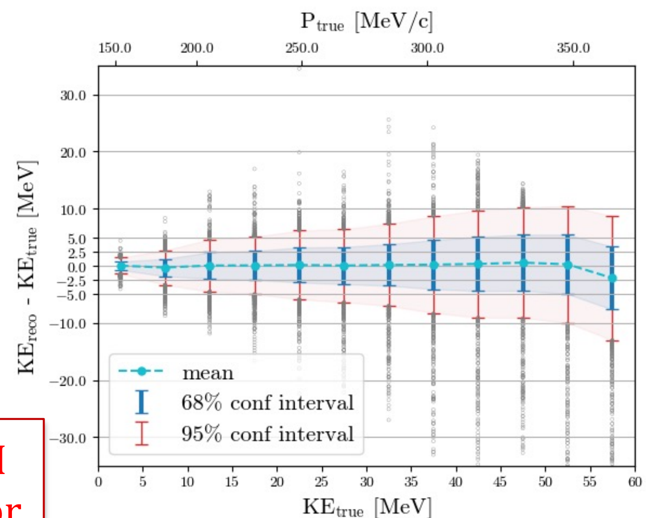
1.2M
simulated
images

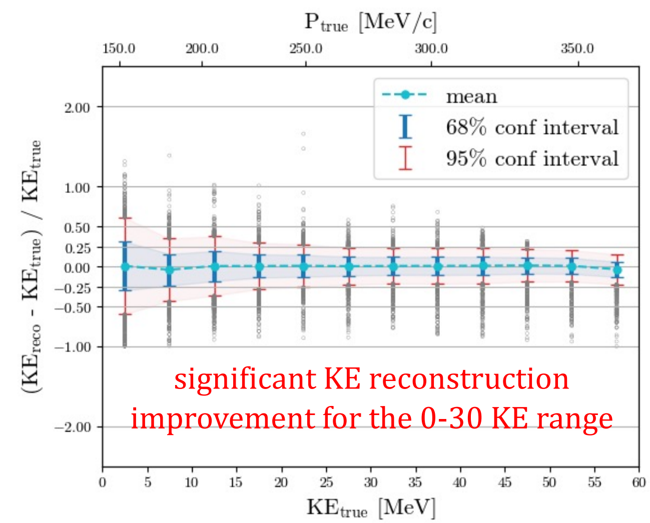




GAN
brute force

10M
generated
images





significant KE reconstruction improvement for the 0-30 KE range

GAN trained on the 1.2M simulated images used for the standard brute force!

Summary

- Deep learning is used for different complementary tasks.

- Rejecting noise and identifying single-particle hits.
  - Graph neural networks (GNN) and Sparse Convolutional Neural Networks (SCNN).

- Fitting the track trajectory.
  - Recurrent neural networks (RNN) and Transformers.
  - Better performance than Bayesian inference.

- Fast vertex activity simulation.
  - Generative adversarial networks (GANs).
  - Useful for the sampling stage of the vertex activity fitting.

- Future work:
  - Fully validate the different methods (avoid biases, test on control samples, etc.).
  - Integrate the different methods into the same analysis.

RECONSTRUCTION IN A 3D PLASTIC SCINTILLATOR DETECTOR USING DEEP LEARNING

Saúl Alonso-Monsalve

ETH Zurich

IPA-ML

21 March 2023

Training details

- Each proton is a 5x5x5 cube volume, with a proton starting uniformly in the centre cube of the volume; isotropic direction, uniform KE in the range 0-60 MeV.
- We may look at the generated images during training.
  - Fix arbitrary input parameters:
    - $\overrightarrow{X_{ini}} = (-2.3e^{-7}, -2.3e^{-7}, -5.1e^{-3})\ [mm]$.
    - $KE = 30\ MeV,\ \theta = 2.356\ [rad], \varphi = 1.571\ [rad]$.

Example of true simulated image using the selected parameters