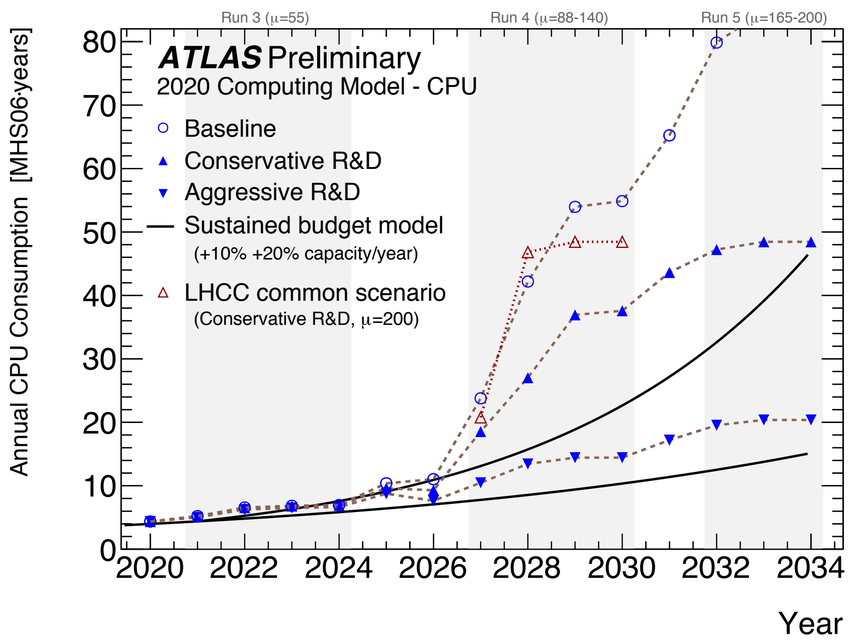
FAST.AR Simulation
- with Deep AutoRegressive Networks -

Ioana Ifrim
EP-SFT

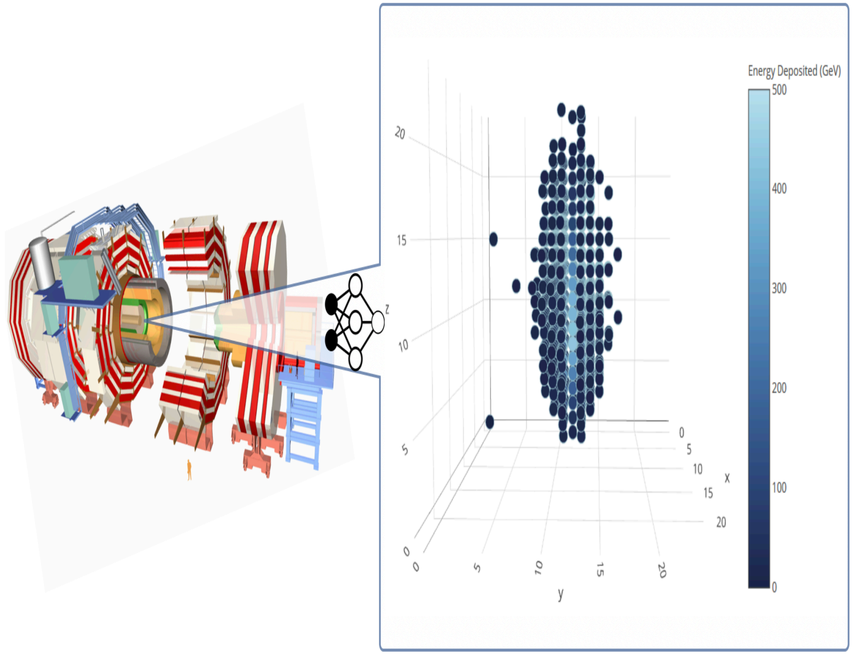
Motivation

Computational Requirements



Increasing luminosity and energy of particle accelerators pose greater challenges - large MC statistics to model experimental data - more collisions = more data = more computing resources required

Deep Learning Fast Simulation

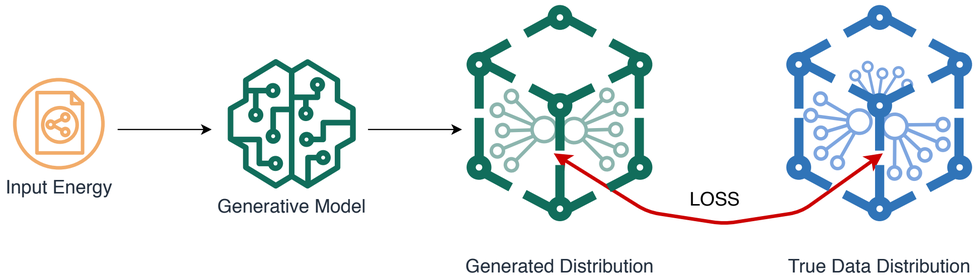


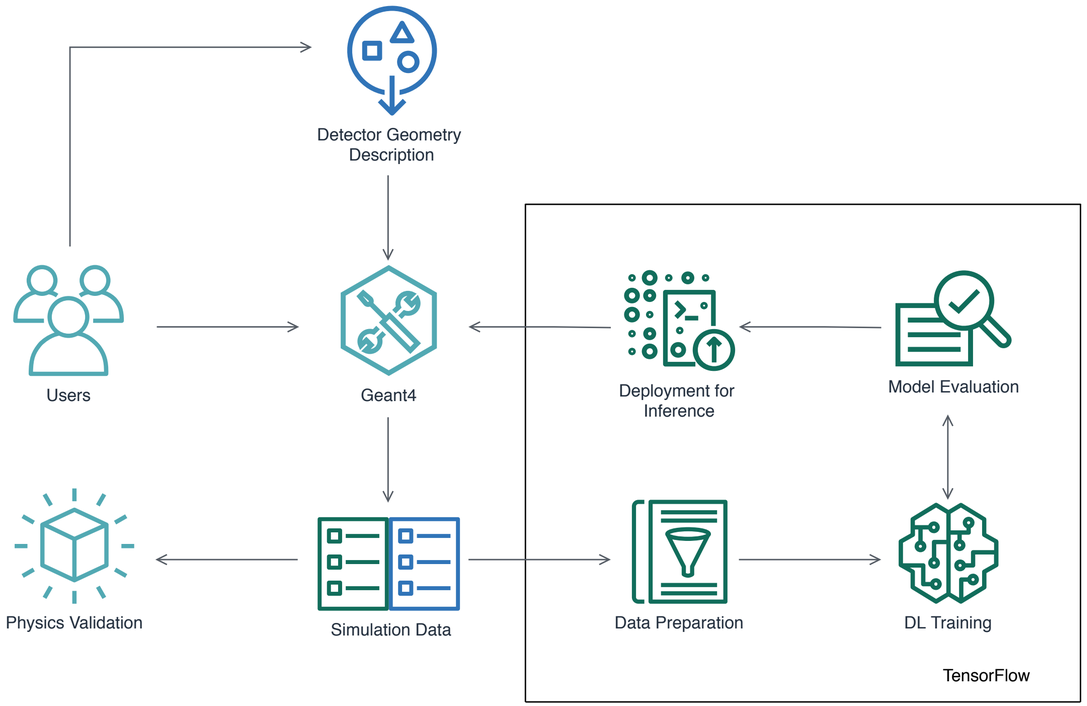
Using Deep Learning principles in treating physics data, we can generate the simulation output (energy depositions) in a fast manner - by employing a Deep Learning generative model

Retrieved from [1]https://twiki.cern.ch/twiki/bin/view/AtlasPublic/ComputingandSoftwarePublicResults [2] https://cds.cern.ch/record/1433717/files/CMSnc.jpg

Deep Learning for Fast Simulation & Streamlined DNN Fast Simulation Workflow

- Achieve both computational and statistical efficiency in estimating distributions of complex, high-dimensional simulation outputs through generative deep learning models





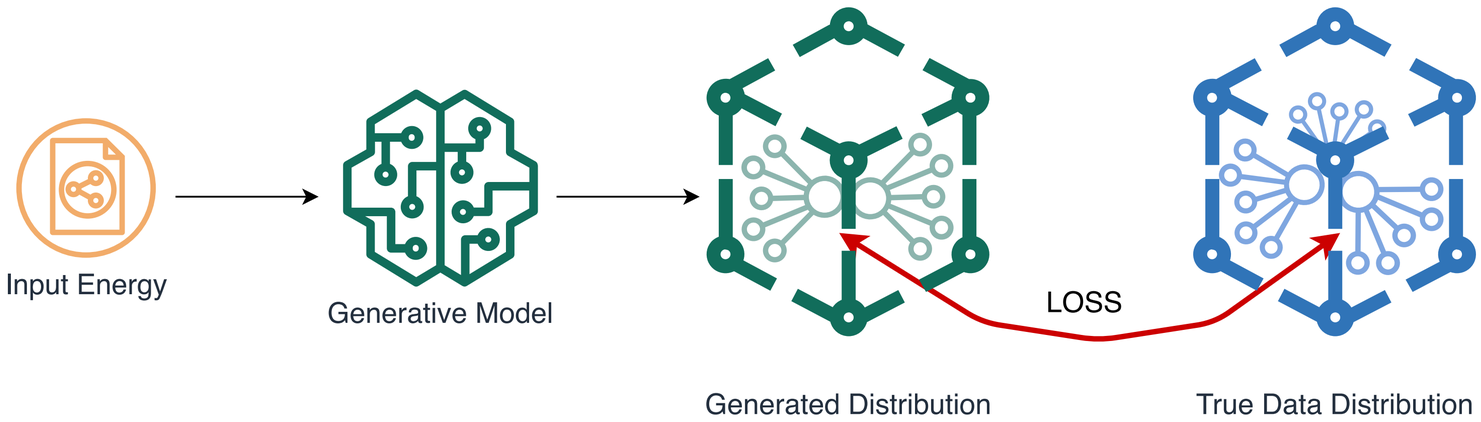
- Implement a full cycle system for Geant4 integration of Deep Learning utilities, from data production through inference integration and results validation

Deep Learning for Fast Simulation - Goals

- Estimate distributions of complex, high-dimensional data:
    - One data sample - event - lies in a ~14000 dimensional space (24x24x24)
    - Features are correlated between event cells

- Achieve both computational and statistical efficiency:
    - Efficient training through parallelisation
    - Expressiveness and generalisation
    - Event sampling quality and speed

Generative Deep Learning



- The learning aim is for the parameters within a group of model distributions to minimise the distance between the model distribution p_θ and our calorimeter showers data distribution p_{data}:

$$\min_{\theta \in \mathcal{M}} d(p_{\text{data}}, p_\theta)$$

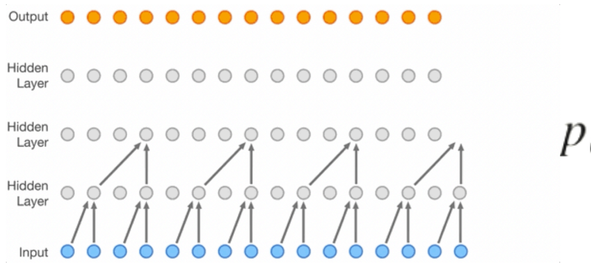
- Our choices of design revolve around:
  - which model representation is suitable?
  - what is the objective function of distance, d(·)?
  - which optimisation procedure should we use for minimising d(·)?

Generative Models

- Models learn the probability density function differently:

The choice of our study

Explicitly - tractable: **AutoRegressive Models**

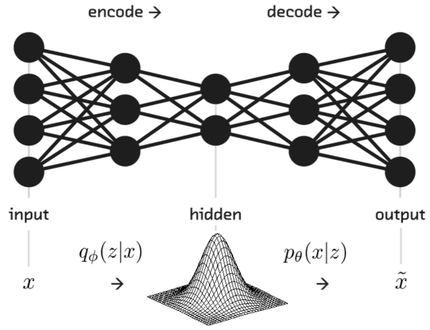


$$p_\theta(\mathbf{x}) = \prod_{i=1}^{N} p_\theta(x_i|\mathbf{x}_{<i})$$

- Complex modelling of the data distribution
- Able to encode long term dependencies between energy cells
- Training is highly parallelizable given the type of operations
- The training is stable
- Autoregressive factorisation is general: expressivity of model
- Meaningful parameter sharing has good inductive bias => good generalisation

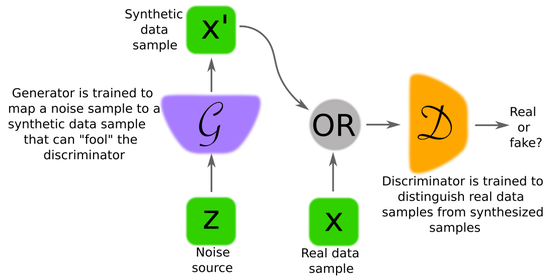
Commonly studied

Explicitly - approximation: Variational Autoencoders (**VAEs**)



$$p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}, \mathbf{z}) d\mathbf{z}$$

Implicitly: Generative Adversarial Networks (**GANs**)



$$\min_\theta \max_\phi V(G_\theta, D_\phi) = \mathbb{E}_{\mathbf{x}\sim p_{data}}[\log D_\phi(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}[\log(1 - D_\phi(G_\theta(\mathbf{z})))]$$

https://blog.fastforwardlabs.com/2016/08/22/under-the-hood-of-the-variational-autoencoder-in.html https://tinyurl.com/y2vl6ulm https://deepmind.com/blog/article/wavenet-generative-model-raw-audio

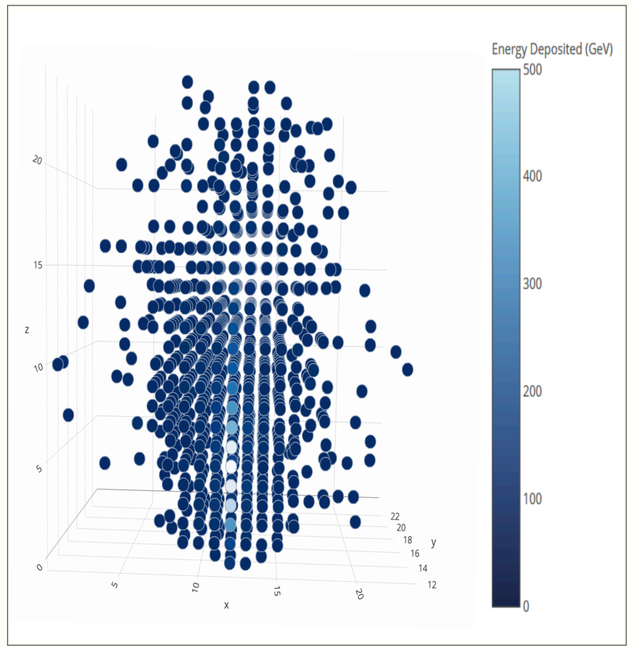
Training Data Production

- Using the Geant4 full simulation toolkit

- A single particle event for deep learning training is represented by:

    - the label = particle properties (energy, type, …)
    - energy depositions in cylindrical coordinates

- Flat incoming energy spectrum (1-100 GeV) along z axis



EM sum shower for 10, 500 GeV electrons

- For training the data used consisted of 24x24x24 cells in R x Phi x Z coordinates, $PbWO_4$

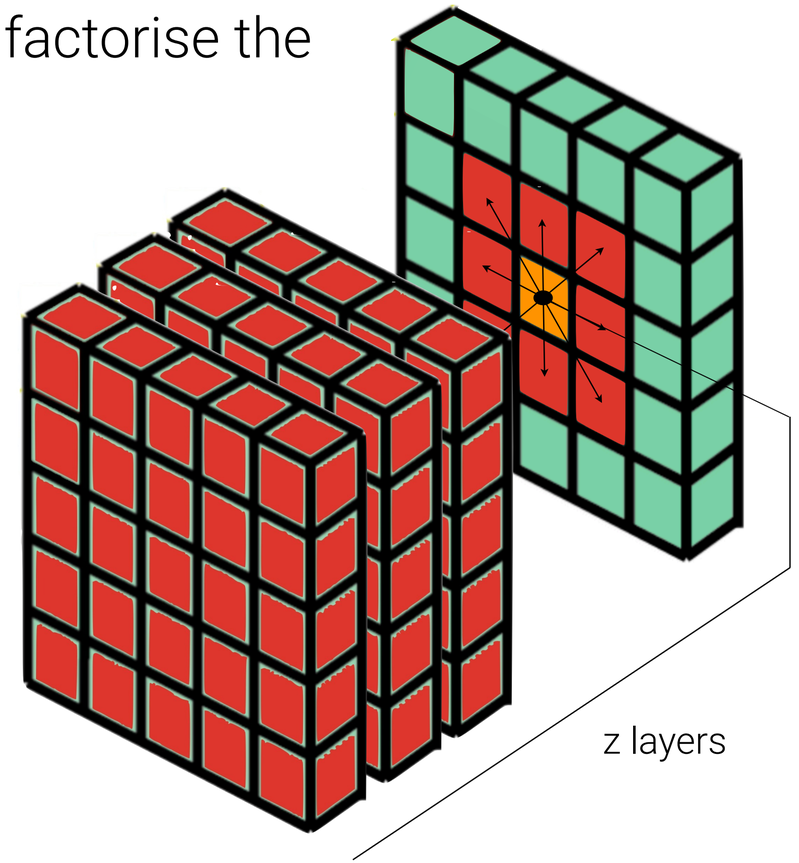
Model Representation - AutoRegressive

- Given our dataset D of n-dimensional data points x, we can factorise the joint distribution over the n-dimensions as:

$$p(x) = \prod_{i=1}^{n} p(x_i | x_1, \ldots, x_{i-1})$$

Likelihood of event x

Probability of i'th energy deposition value given all previous depositions



- Correlations between layers are kept given that cell x_i is dependant on preceding z layers cells, thus physics relevant dependencies between calorimeter cells are implemented

Modelling of Data Distribution

- How to generate energy depositions values based on incoming particle properties
    - predict the distribution of energy depositions, thus any value will have a probability to be represented (regardless if it is present or not in the training set, as per softmax)

- How to model the complex data distribution?
    - use a linear combination of distributions

$$p(x) = \pi_1 \times p_1(x) + \pi_2 \times p_2(x) + \ldots + \pi_n \times p_n(x)$$

where π_i is the probability of the distribution to be picked

Loss Functions

- To train sampling layers in the neural network, the objective is to have some output distribution parameters that maximise the likelihood of a target y to be sampled from such distribution

- Our loss is comprised of:

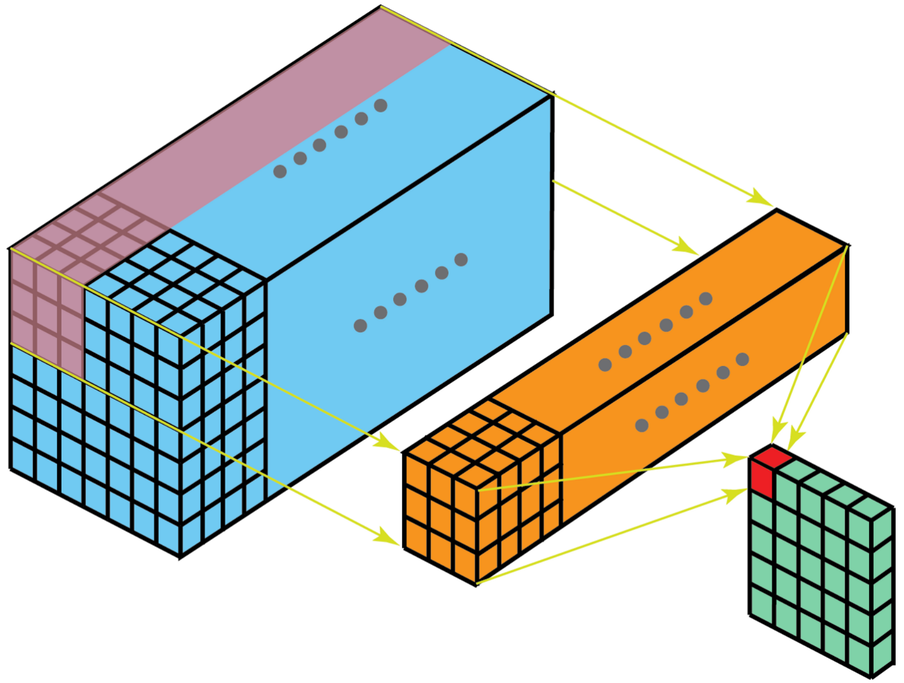
    - The loss on the distribution itself (how likely is the distribution to hold the target y)

    - The loss on the distribution selection

Network Implementation

- Cells are conditioned along the shower development axis while subsequent information is blocked

- The dependency on previous layers is modelled using a Convolutional Neural Network over a context region

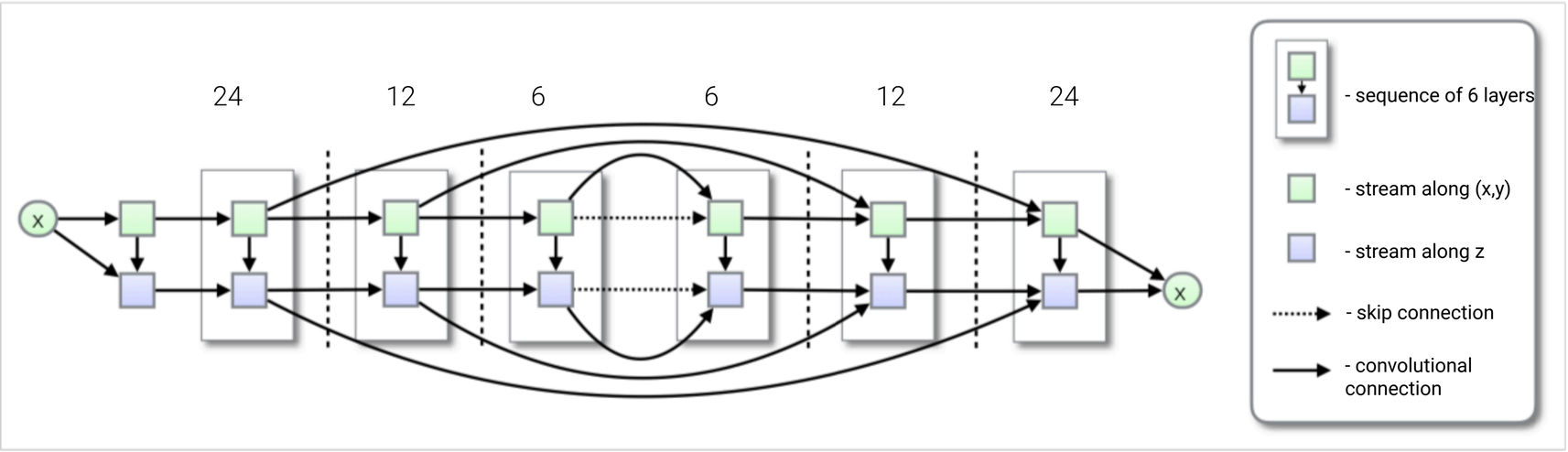


https://towardsdatascience.com/a-comprehensive-introduction-to-different-types-of-convolutions-in-deep-learning-669281e58215

Network Implementation

- The network captures the long range structure by downsampling with convolutions of stride 2 (thus improving the relative size of receptive field); the loss in information is accounted for by adding extra short-cut connections

- The behaviour of a Recurrent Neural Network is emulated with a Convolutional Neural Network in order to parallelise the computations and control the access to past information



https://openreview.net/pdf?id=BJrFC6ceg

Training Details and Experiments

- For these experiments, the “true” data distribution is represented by a total of ~ 1500 events / energy label

- Data consists of 24x24x24 cells in R x Phi x Z coordinates, $PbWO_4$

- The aim is to achieve good results on large datasets (where the number of labels differ with the energy range and granularity chosen)

- Since the network is only seeing empirical data distribution, not true data distribution it is important for the model to have the ability to generalise:

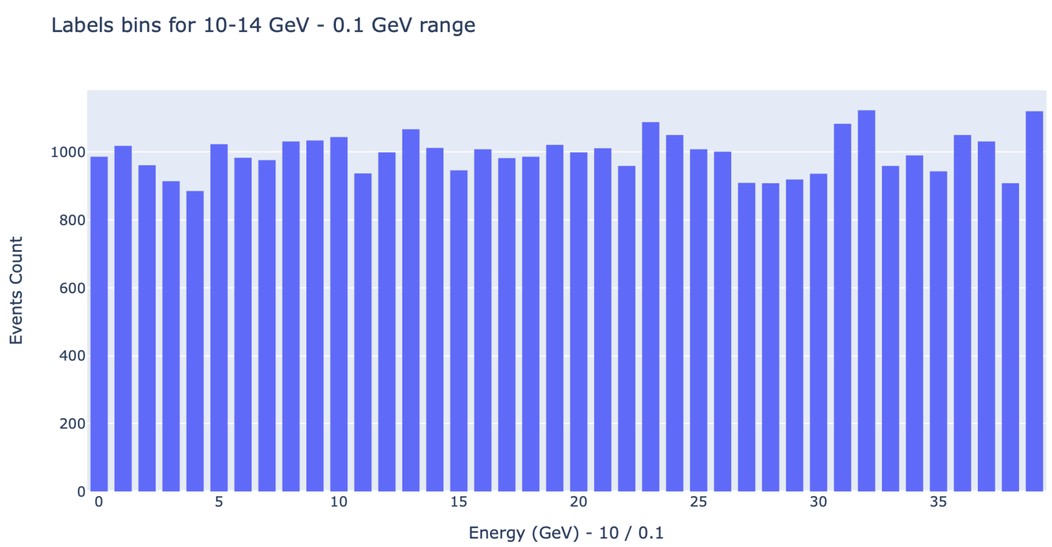
    1. capture small energy fluctuations (0.1 GeV) - through small label granularity

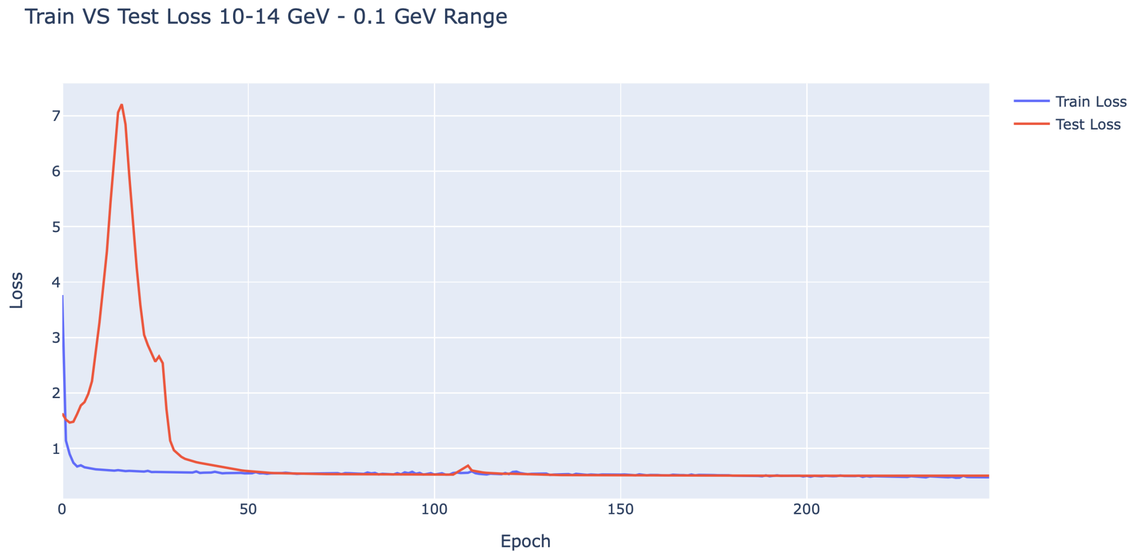
    2. capture both small and large energy responses accurately

Small Label Granularity Experiments

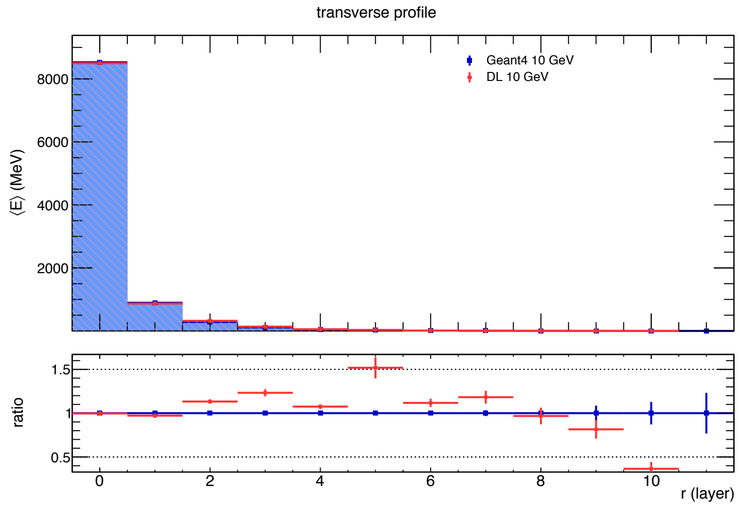
- Since the network is only seeing empirical data distribution, not true data distribution it is important for the model to have the ability to generalise

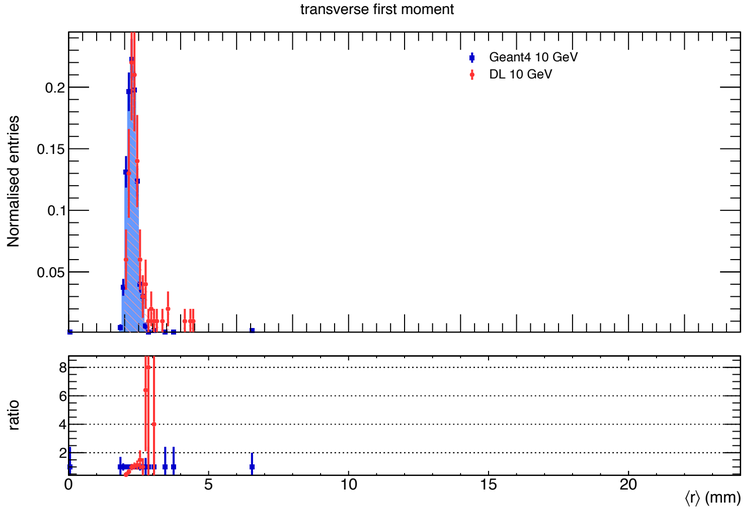
- To test the ability to capture small energy fluctuations, the network was trained on :
    - 50 energy labels
    - 10 - 14 GeV network granularity
    - 0.1 GeV label granularity

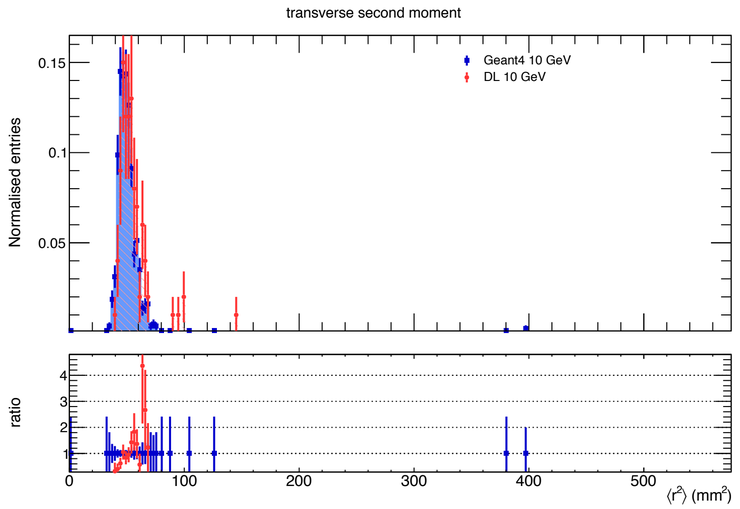


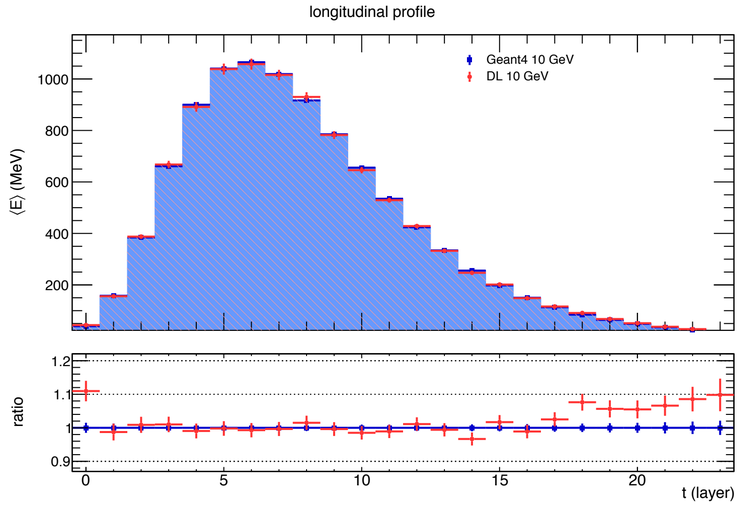


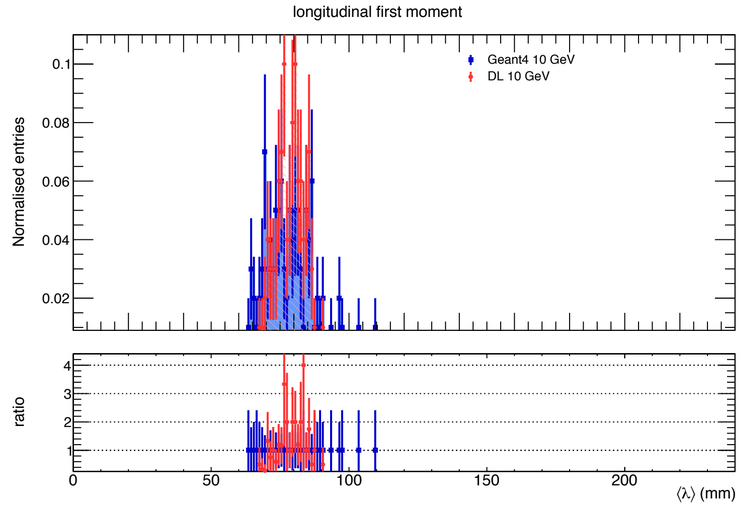
Validation Results - 10 GeV

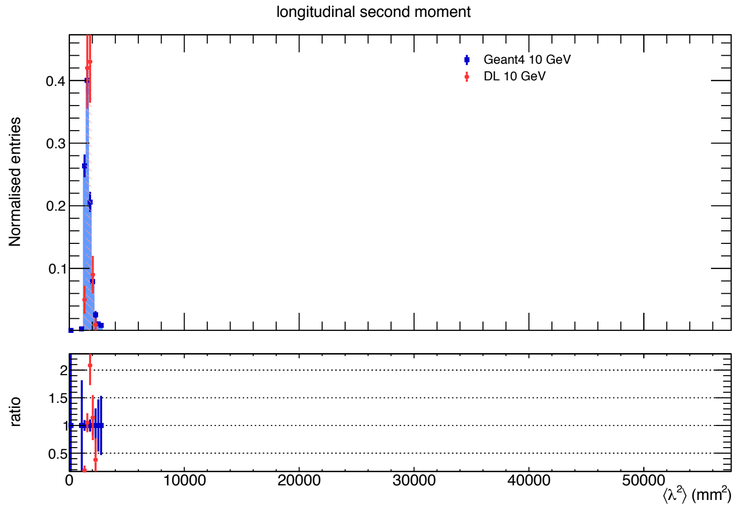




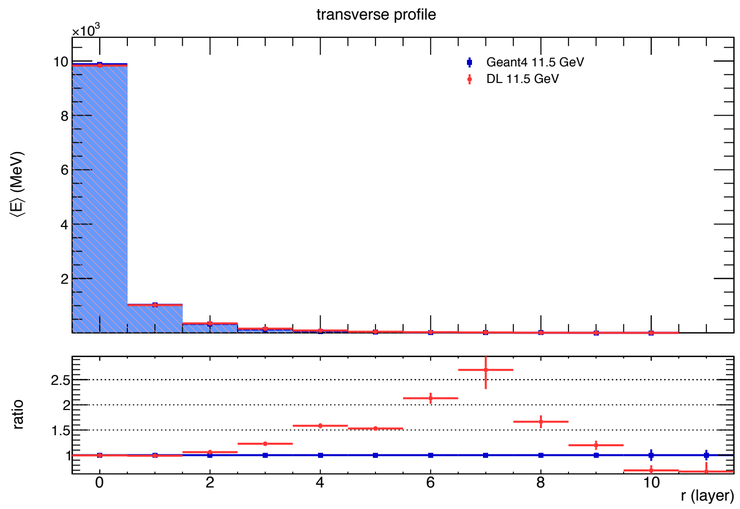


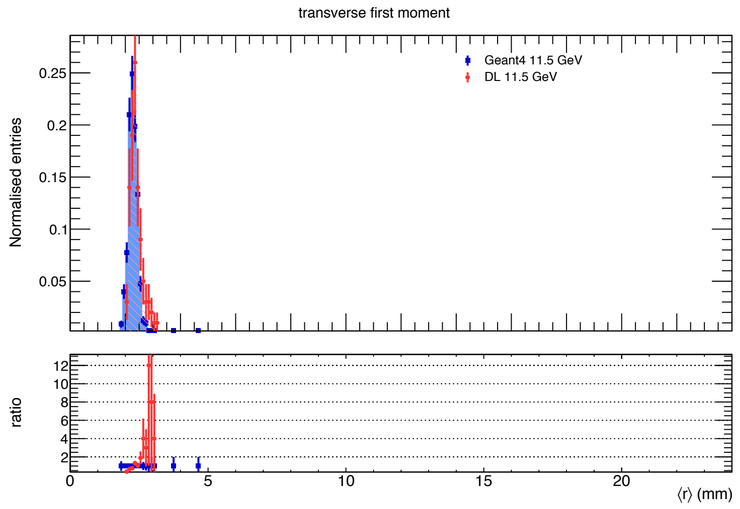


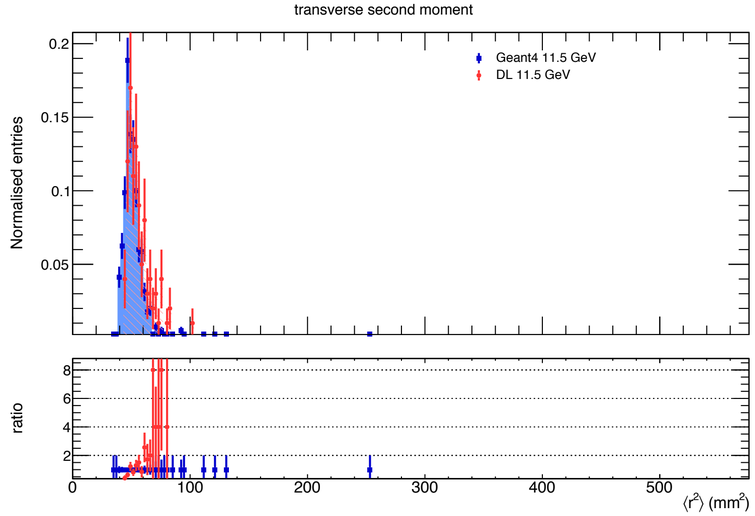


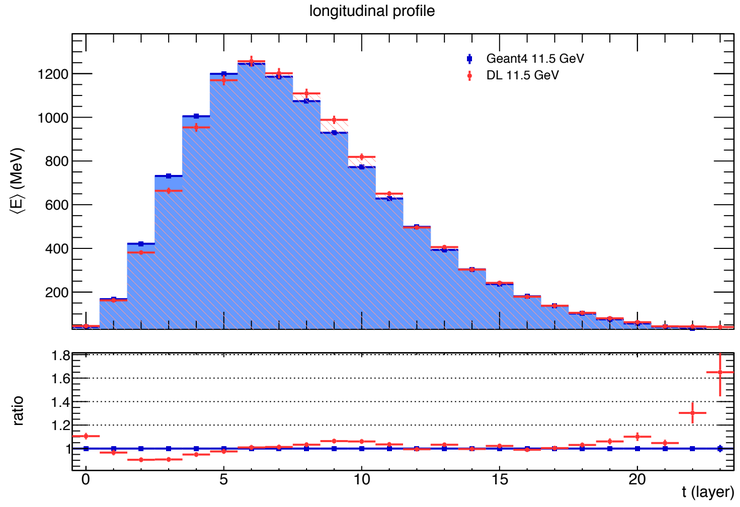


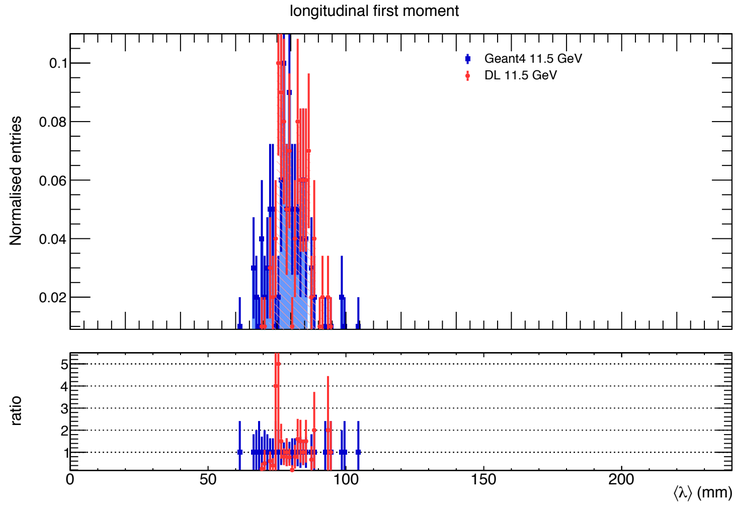
Validation Results - 11.5 GeV

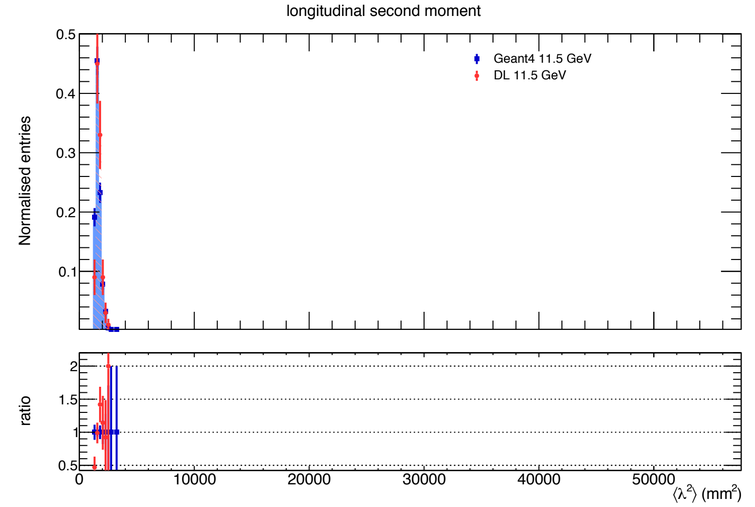




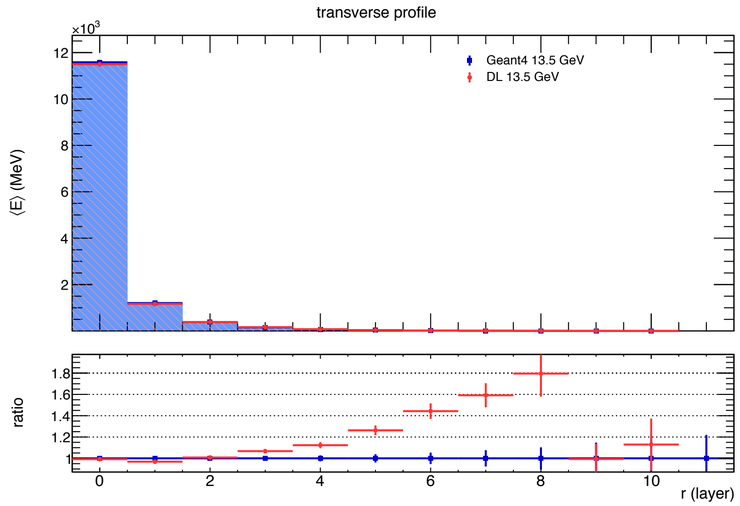


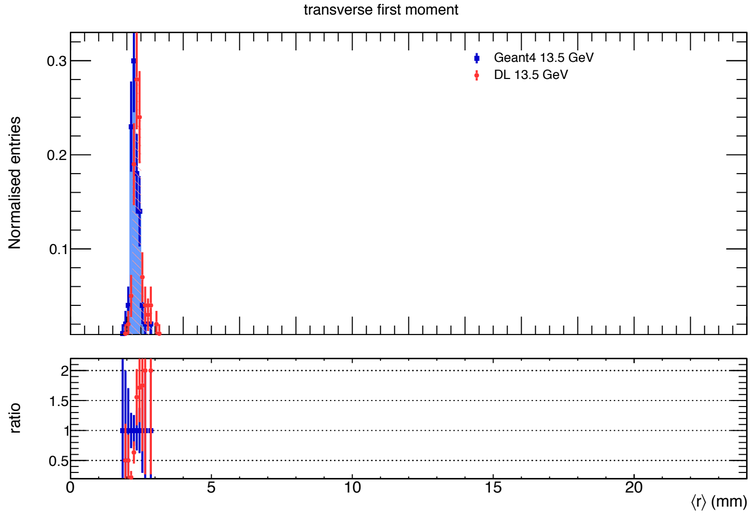


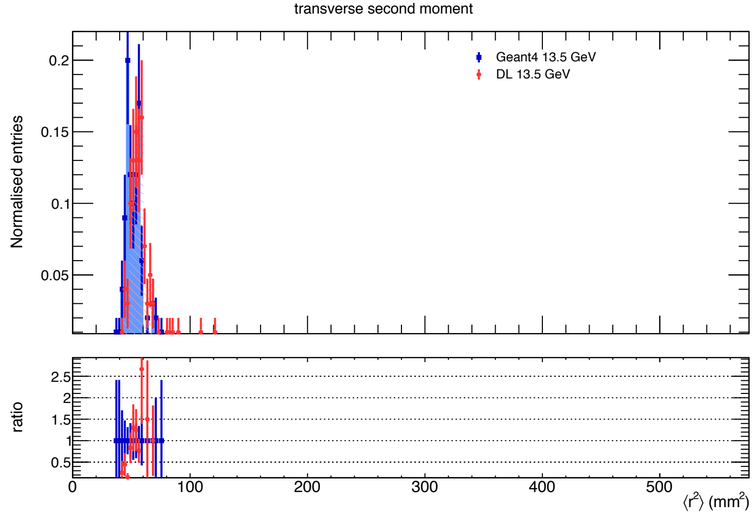


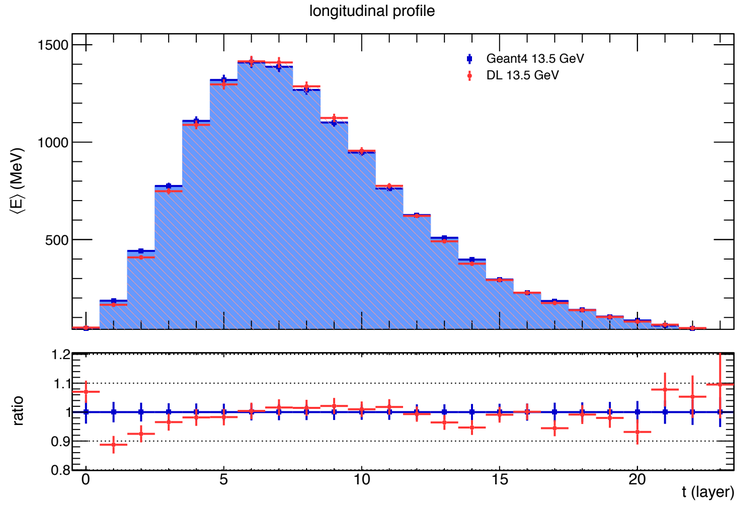


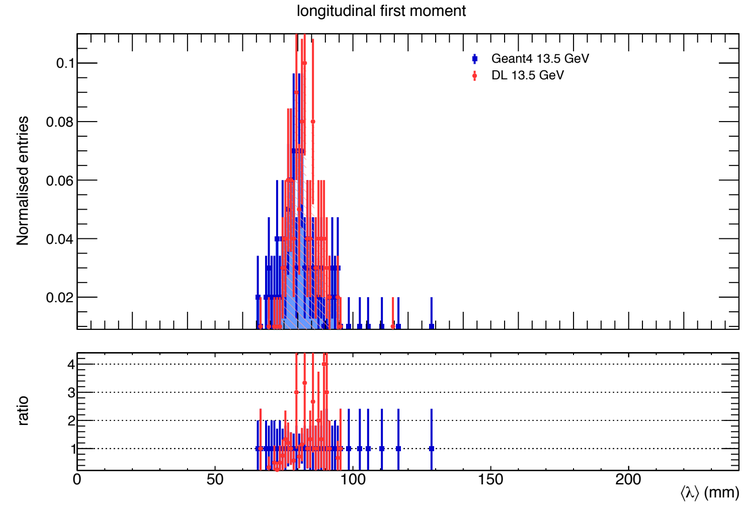
Validation Results - 13.5 GeV

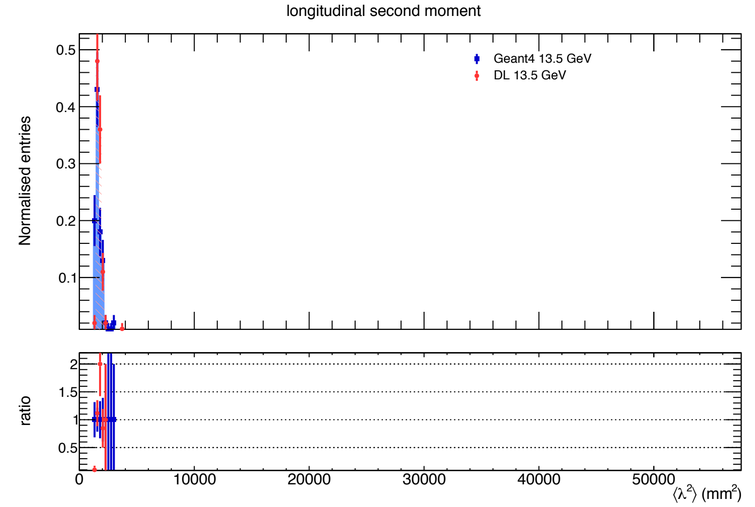








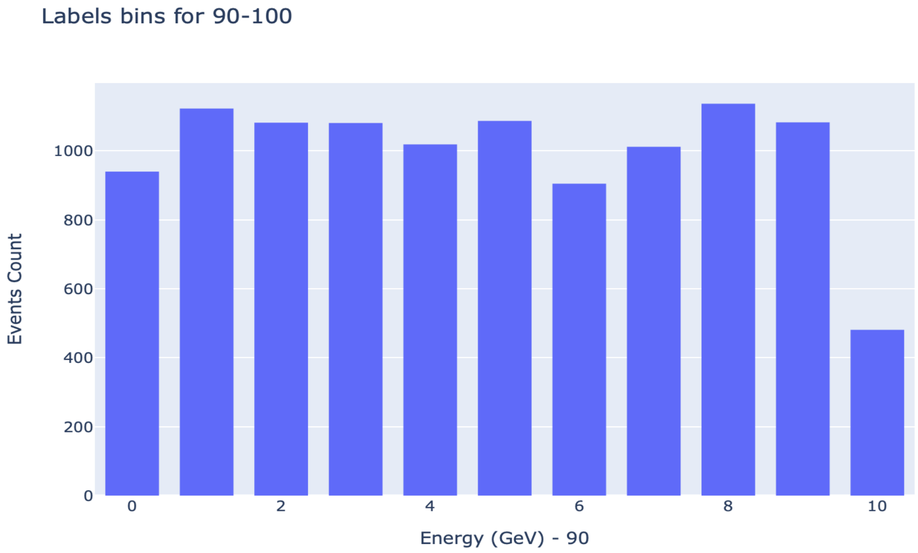


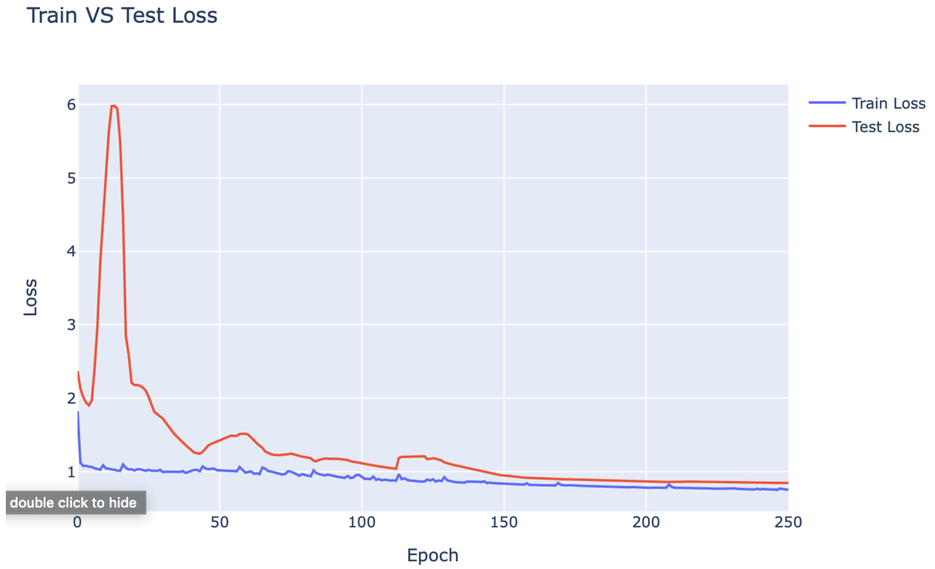


High Energy Experiments

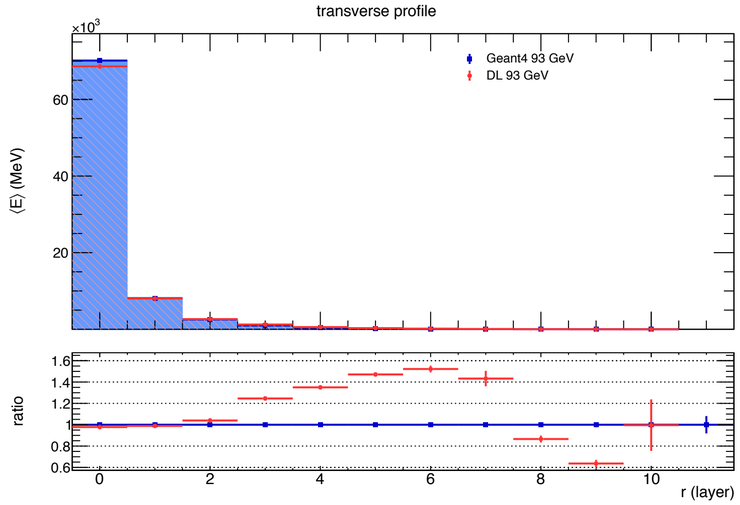
- Since the network is only seeing empirical data distribution, not true data distribution, it is important for the model to have the ability to generalise

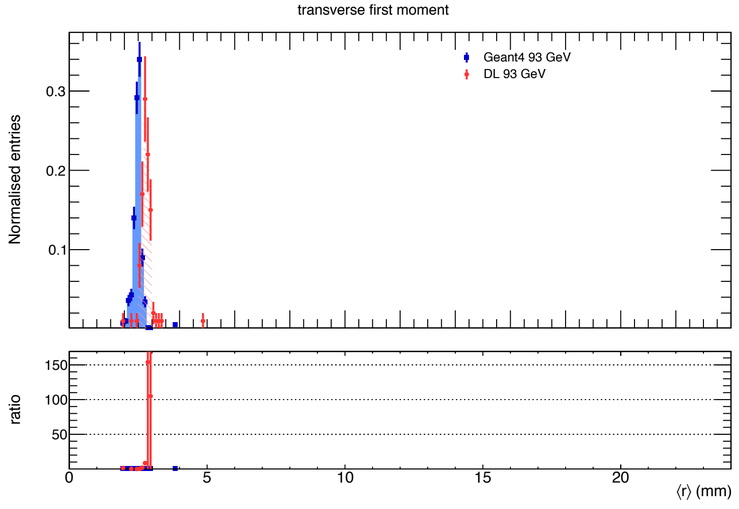
- To understand the generation quality of large energy labels, the network was trained on :
    - 10 energy labels
    - 90 - 100 GeV network granularity
    - 1 GeV label granularity

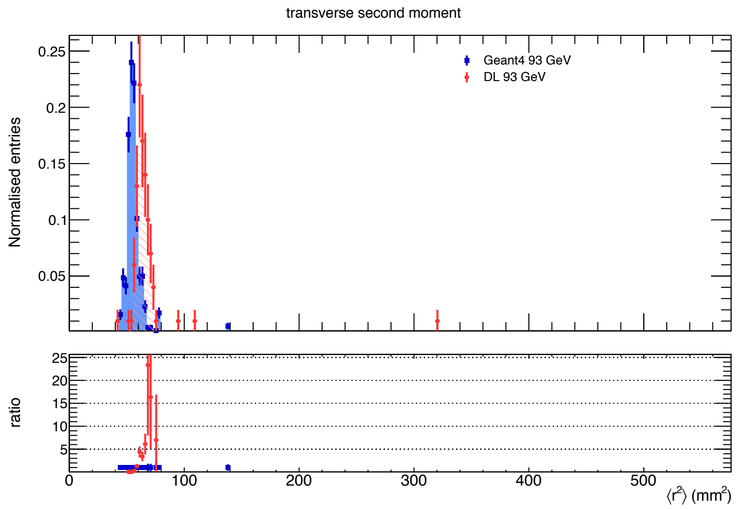


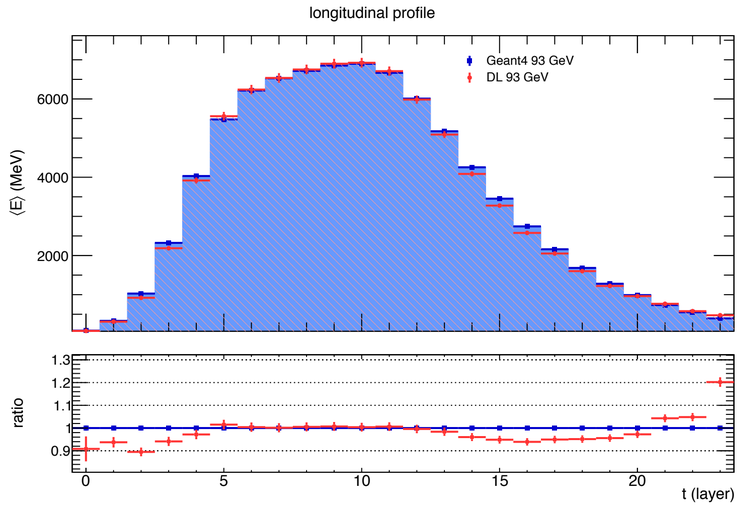


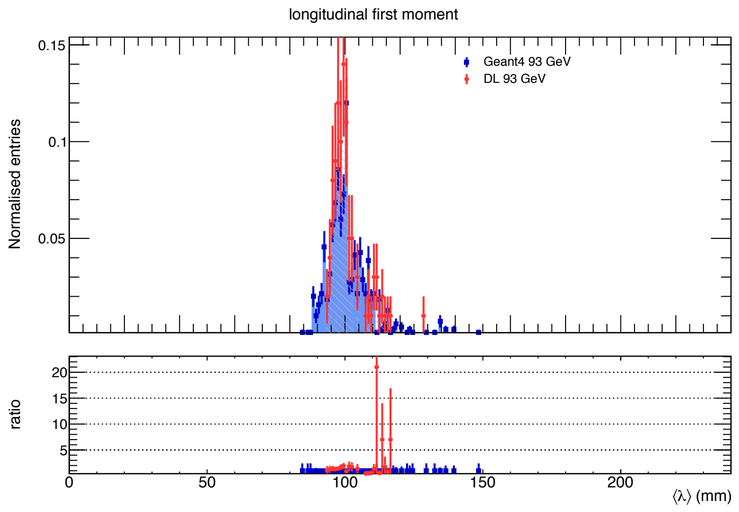
Validation Results - 93 GeV

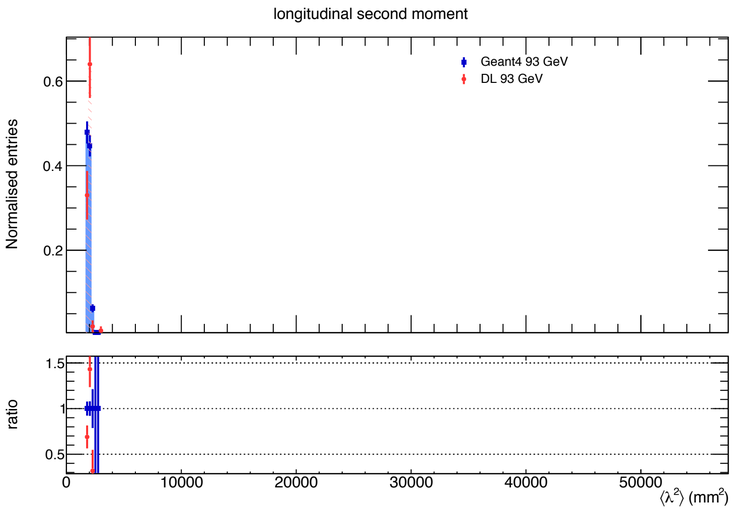




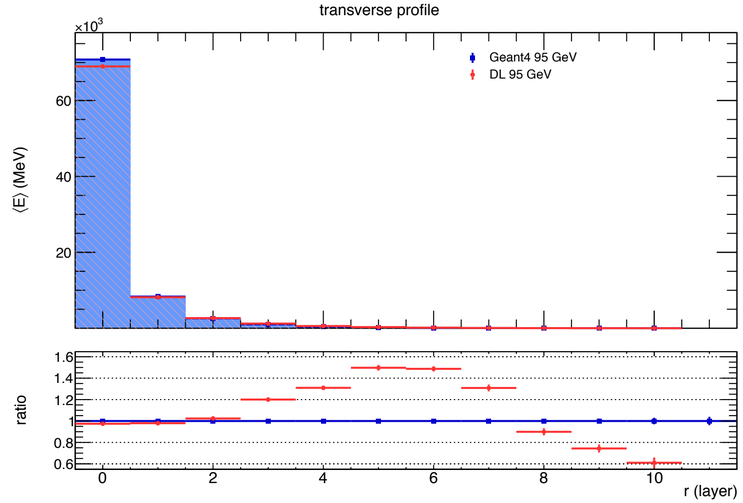


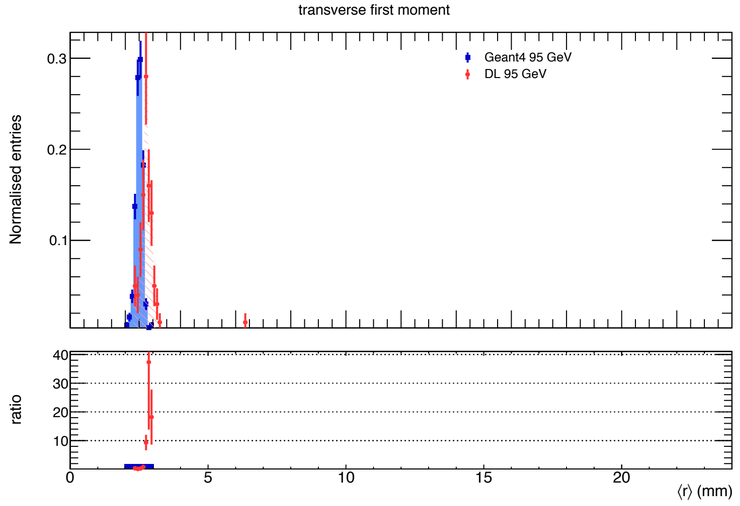


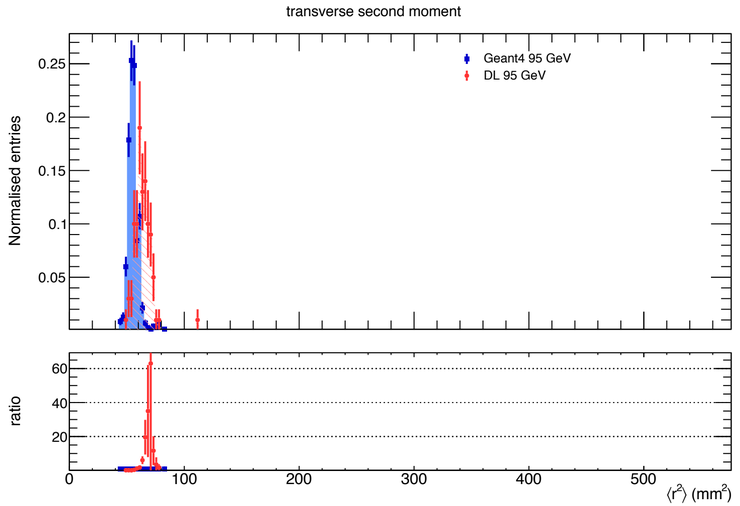


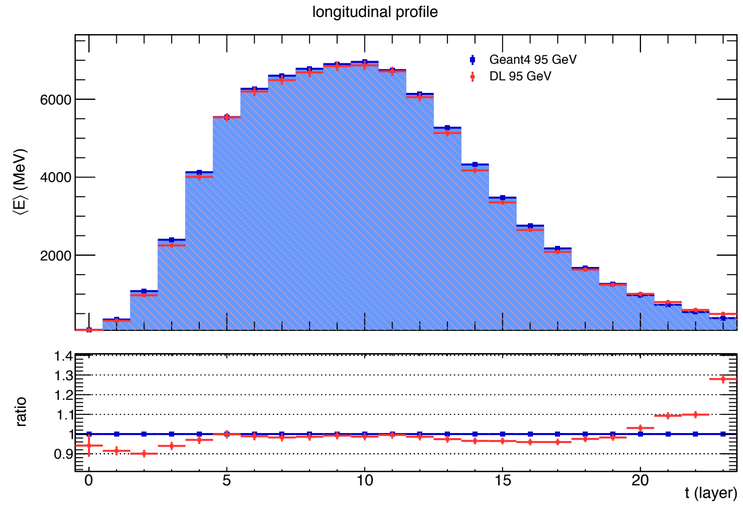


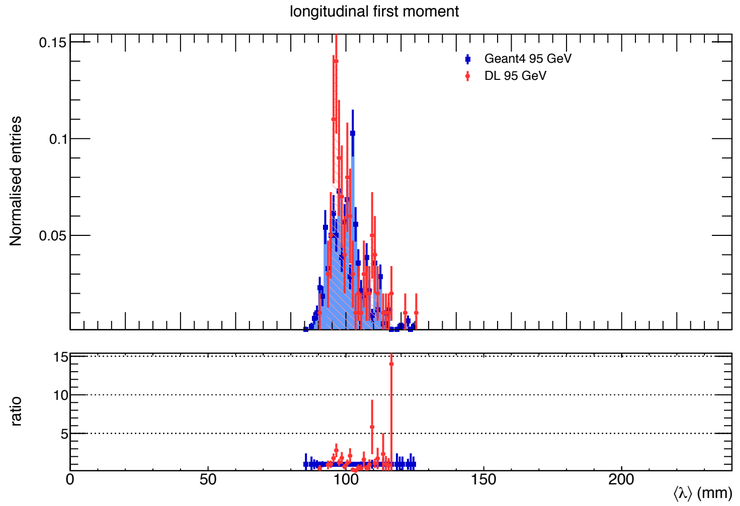
Validation Results - 95 GeV

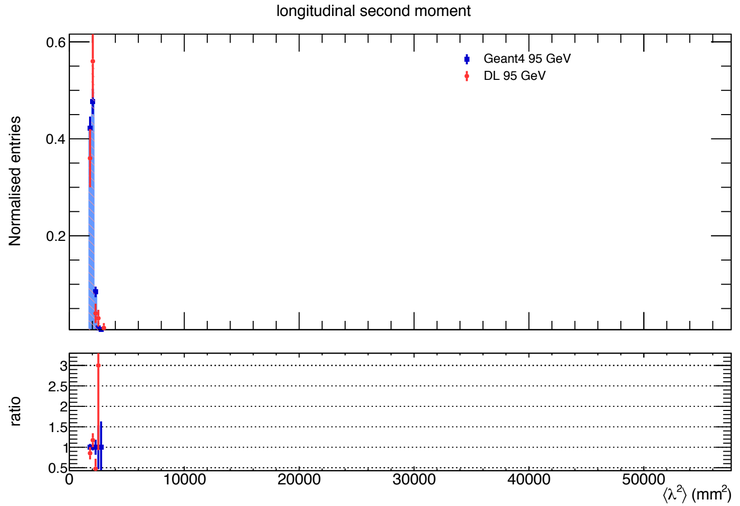




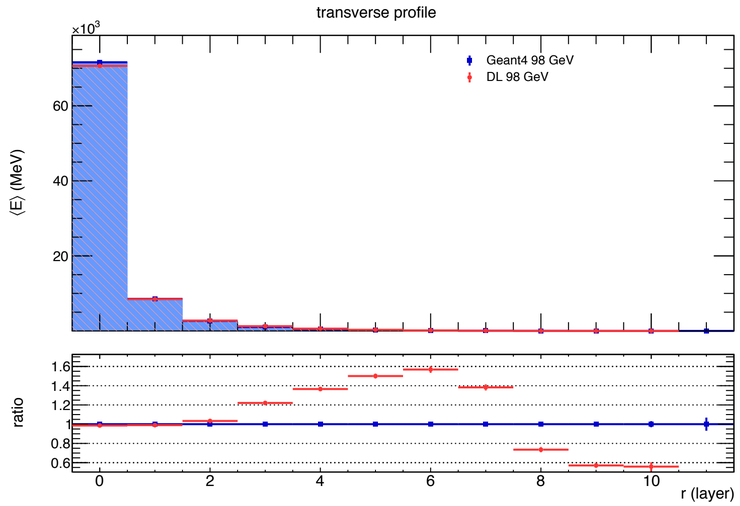


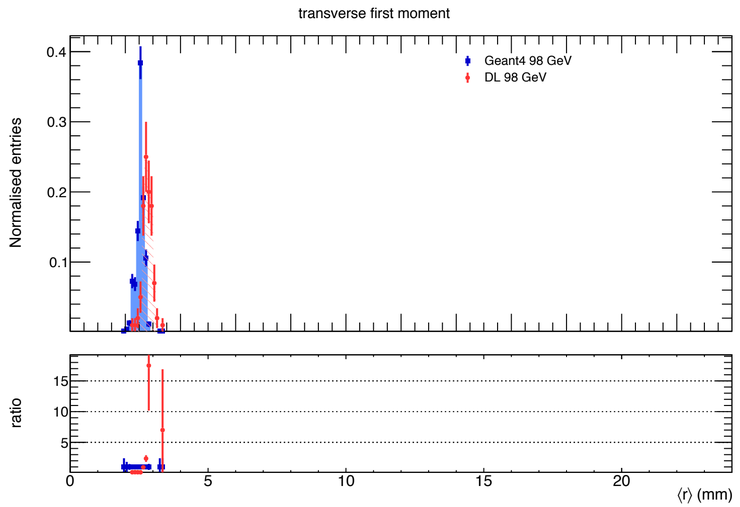


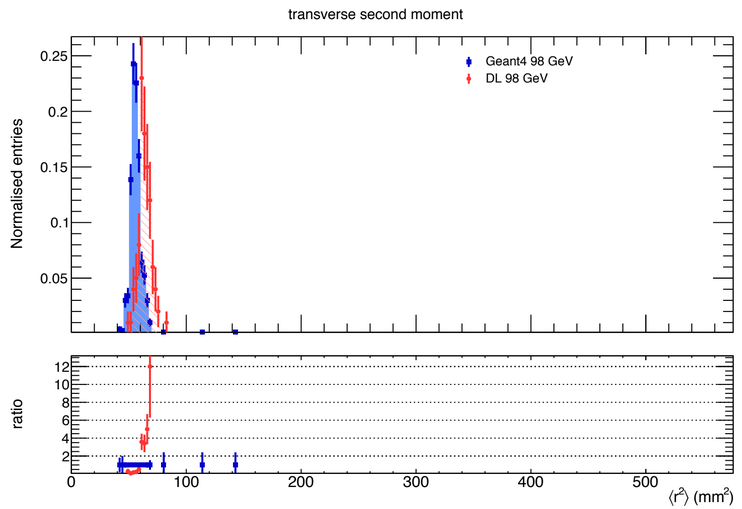


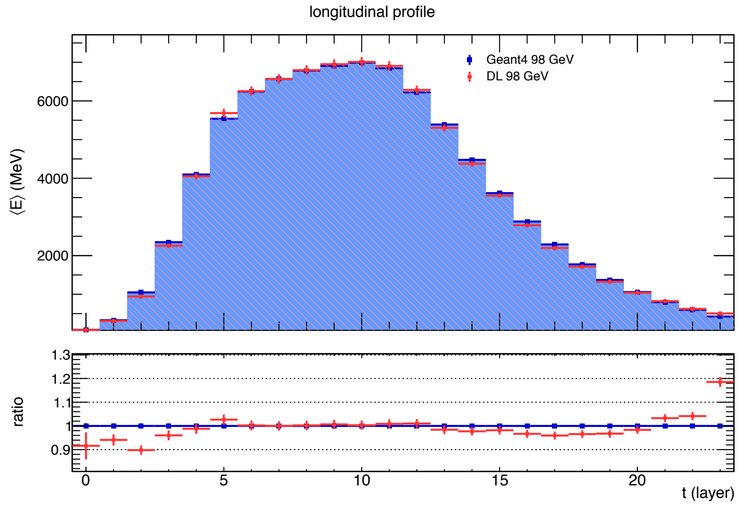


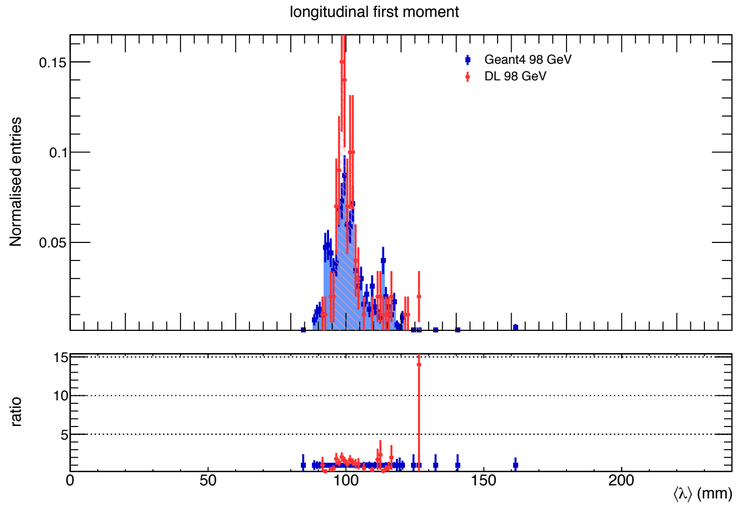
Validation Results - 98 GeV

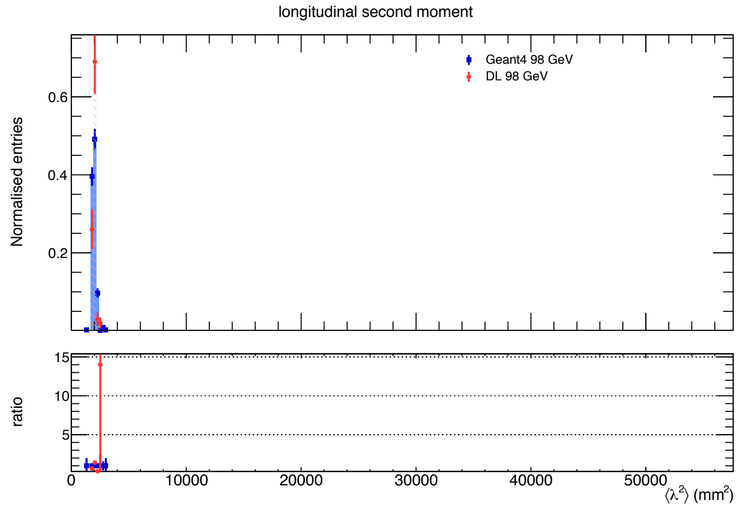












Streamlined DNN Fast Simulation Workflow - Goal

- Develop an end–to–end solution which integrates Deep Learning (DL) simulation methods with Geant4

Step 1: Data Production
– Creating matrices of energy deposits using Geant4.

Step 2: DNN Training
– Generative models HEP customised and trained.

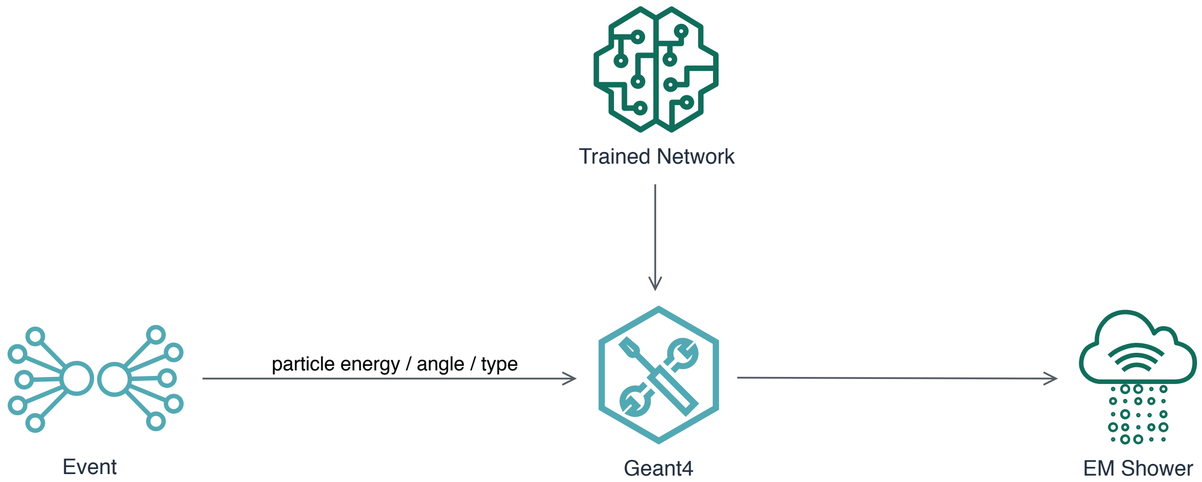
Step 3: Physics Validation
– Sequence chain of HEP performance measurements.

Step 4: DNN Inference
– Geant4 hooks for FastSim with DNN dependencies.

- Steps 1, 3 and 4 are within the Geant4 application, while step 2 is performed independently in custom designed tools

Streamlined DNN Fast Simulation Workflow - Inference Module

Inference refers to computing the posterior distribution for the given observation (particle energy/ angle/type):



- The inference Library builds on top of the TensorFlow C API for seamless integration of inference with C++ projects (using CMake)

- It does not require TensorFlow to be built from source - can be used from LCG or simply download pre-compiled headers & libraries and link against them

- Different generative models can be employed: AutoRegressive Networks, VAEs, GANs

Streamlined DNN Fast Simulation Workflow - Inference Module

- Repository : https://github.com/ioanaif/dl-inference-module

- Use the library to integrate and further validate a trained model given the following:
    - The model's input and output node names,
    - The graph definition in a .pb file as well as the latest checkpoint in .ckpt files,
    - The input data shape information (both for samples and labels),
    - The translation of label to particle energy and of inference output to cell energies

- Use the library with an integrated, pre-existing trained model:
    - Simply chose a model and pass the desired particle labels for generation of events

- Integration of this library with Geant4 will result in obtaining simulation events through any of the integrated models

Observations & Enhancement Possibilities

- This implementation exploits the causal structure of the generative process and preserves layer wise correlations which proved crucial for improving events generation quality

- The network is able to generate shower events without overfitting of implementation on use-case *

From a production-level enhancements possibilities perspective, the important aspects are:

- the energy label can be extended to include angle (extending one-hot encoding)

- *a better data transformation procedure for labels above 20GeV will help improve the 1st/ 2nd moments network results on high energies

- interpolation can be used to increase label granularity while network granularity can be a constant (https://arxiv.org/pdf/1912.05015v2.pdf)

- graph autoregressive networks could be a natural extension for this work suitable for solving geometry issues

Thank you!

Ioana Ifrim
EP-SFT