Grant Agreement No: 101004761

AIDAinnova

Advancement and Innovation for Detectors at Accelerators

Horizon 2020 Research Infrastructures project AIDAINNOVA

MILESTONE REPORT

PROTOTYPE OF ML BASED SHOWER SIMULATION

MILESTONE: MS48

Document identifier:	AIDAinnova-M48
Due date of milestone:	End of Month 22 (January 2023)
Report release date:	24/01/2023
Work package:	WP12: Software for Future Detectors
Lead beneficiary:	CERN
Document status:	Final

Abstract:

Detailed simulation of showers in calorimeters is often the most time-consuming part of computing for high energy physics (HEP) experiments. Instead of the expensive multi-step particle tracking computation, one can develop models that generate energy deposited in the calorimeters according to some parameterisations. Machine learning (ML) techniques are employed to reproduce particle showers. A prototype of the example application for fast shower simulation has been developed and integrated within the standard toolkit for HEP simulation, Geant4. It demonstrates how to produce data for training of the ML model, how to train the model, and use it in the simulation alongside the detailed simulation.

The Advancement and Innovation for Detectors at Accelerators (AIDAinnova) project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement no. 101004761. AIDAinnova began in April 2021 and will run for 4 years.

Delivery Slip

	Name	Partner	Date
Authored by	A. Zaborowska [Task coordinator]	CERN	06/01/2023
Edited by	Paolo Giacomelli [Scientific coordinator]	INFN	23/01/2023
Reviewed by	F. Gaede [WP coordinator] G. Stewart [WP coordinator]	DESY CERN	13/01/2023
Approved by	Paolo Giacomelli [Scientific coordinator]		24/01/2023

TABLE OF CONTENTS

Executive summary

This report documents the prototype of fast calorimeter shower simulation based on machine learning (ML) techniques, which are introduced in the first section.

The second section describes a full workflow that has been developed, with intermediate steps published in a form that can facilitate studies of other researchers, including the relevant simulation data. A pre-trained ML model to generate showers is distributed with the example, alongside the code used for training. The validation of fast simulation can be performed easily thanks to the integration of the inference libraries in the Geant4 toolkit. This is a key step necessary to incorporate ML fast shower simulations in production frameworks at HEP experiments. Three inference libraries are incorporated, to present a choice to users and allow for benchmarking.

The third section describes how to install and run the example. Code of this example, Par04, is distributed in Geant4 starting from release 11.0, ensuring its accessibility and continuous testing.

1. INTRODUCTION

Detailed simulation of showers can be replaced by a faster generation of the resulting energy deposits in the calorimeters. Machine learning (ML) techniques, and in particular generative models, can be employed for this task. They require training on the detailed simulation outputs in order to produce meaningful results, and they need to be integrated within the simulation frameworks to perform inference and generate showers (in the form of energy deposits).

The scope of this milestone is to provide the example application demonstrating how to employ machine learning (ML) techniques for shower simulation. The full workflow is documented and published, starting with the production of the simulation training data, then design of the ML model, training, and its integration back into the simulation toolkit. The latter allows a fair comparison between detailed and fast simulation. The example incorporates several inference libraries to fit the needs of different users, as well as to benchmark their performance. This example has been released in Geant4 [1] 11.0, with further improvements introduced in version 11.1 [2]. Integration of this simulation application within the standard toolkit for detailed simulation ensures its accessibility to physics community.

This report will first detail the fast simulation workflow in Section 2, before giving a brief summary of how to use the published example and the dataset in Section 3. More details on the work that has been carried out in order to complete this milestone, as well as the work around that topic is regularly updated on the website [3].

2. FAST SIMULATION WORKFLOW

2.1 WORKFLOW

Fast simulation differs from the full (detailed) simulation as it does not act on a microscopic scale, it instead replaces the final output of the detailed simulation. Hence, it strongly depends on the detector and every shower model must be tuned to the output of the full simulation for this specific detector. Once machine learning models are trained, they can be used to generate new showers in the inference step that is integrated in the same application that is used to produce initial data. This allows the validation and benchmarking of the fast simulation, as well as, ultimately, to use those generated showers in the production code of HEP experiments.

Elements of the fast simulation workflow are presented in Fig. 1 and described below:

- **Geant4 application:** The application used to produce full simulation data. It is released under the name of Par04 within the extended examples of Geant4 [2]. It demonstrates the use of fast simulation on a sampling calorimeter composed of silicone and tungsten layers, with the possibility to easily change the materials and their thicknesses in the configuration file.
- **Pre-processing**: Produced full simulation data is pre-processed to prepare for ML training. This includes scaling of the data and encoding all condition information (detector identifier, initial particle energy, the angle at which it enters the detector).
- **Training**: Training uses the pre-processed data, applying the **optimisation** at training to search for the best set of hyperparameters of the model, with the **initial validation** performed to compare to full simulation. Both pre-processing and training steps are performed in Python and the necessary code is distributed in the Par04 example.
- **Fast sim model**: The trained and validated model is exported to a format that can be used in the inference library and stored to disk. This can be JSON for the LWTNN library, ONNX for ONNXruntime, or serialized Torch Script format for the LibTorch library.
- **Inference**: The same Geant4 application Par04 can be used both to produce the training data, and to run fast simulation. Different inference libraries can be chosen, depending on the input model format. If the model was trained for an incident particle (energy, angle), then a shower it causes can be generated with the ML model. All other particles will be simulated in detail. The ultimate performance **validation** and time speed-up can be assessed. Optimisation at inference such as quantisation and graph optimisations are used to further reduce the memory footprint of the model at the inference time. First results of optimisation were published in [4].

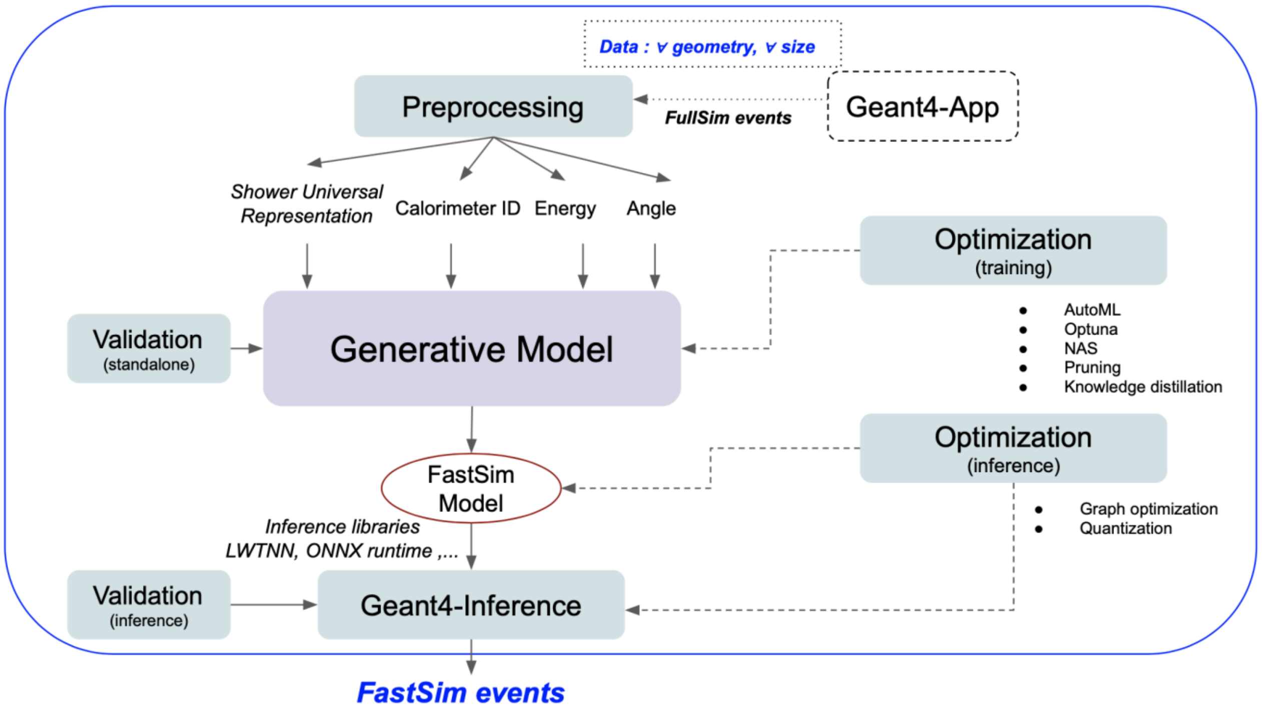


Fig. 1 All components of the fast simulation workflow.

2.2 SIMULATION DATA

Example Par04 includes a calorimeter that is composed of concentric cylinders, with interchanging active and passive layers. Materials and their thicknesses are defined in the configuration file, set in the example to 0.3 mm silicon and 1.4 mm tungsten. Incident electrons are simulated in the calorimeter, with the energy deposits scored in the output file. The example can be used to produce full simulation data, but a sample has been already produced and published at Zenodo [5] to facilitate studies by different researchers. This publication also includes a different calorimeter, built with 1.2 mm scintillator and 1.4 mm lead layers. The full simulation samples are showers of electrons that are generated with an energy range from 1 GeV to 1 TeV (stepped in powers of 2) and angles from 50° to 90° (in steps of 10°). Ten thousand particle showers are simulated for each primary particle energy and angle. The first calorimeter (silicon/tungsten) is also used in the first fast shower simulation ML challenge [6] and used as a benchmark for development of different ML models.

2.3 GENERATIVE MODEL

Data produced using full simulation is used as input to train the ML model and to validate it. A variational autoencoder (VAE) architecture is employed in this example, presented in Fig. 2. It comprises 4 hidden layers with width of 100, 50, 20, 14 and 14, 20, 50, 100 for the encoder and decoder respectively. The decoder part can be used to generate showers, using the 10-dimensional latent space and initial conditions (energy, angle). In this example only one calorimeter (geometry) is used. Training was performed on the silicon/tungsten calorimeter, so each change to the calorimeter's materials or dimensions will require a repetition of the data production, as well as re-training. The published example, Par04, is distributed with a pre-trained VAE model, stored in 3 different formats, suitable for all inference libraries (LWTNN, ONNXruntime, LibTorch).

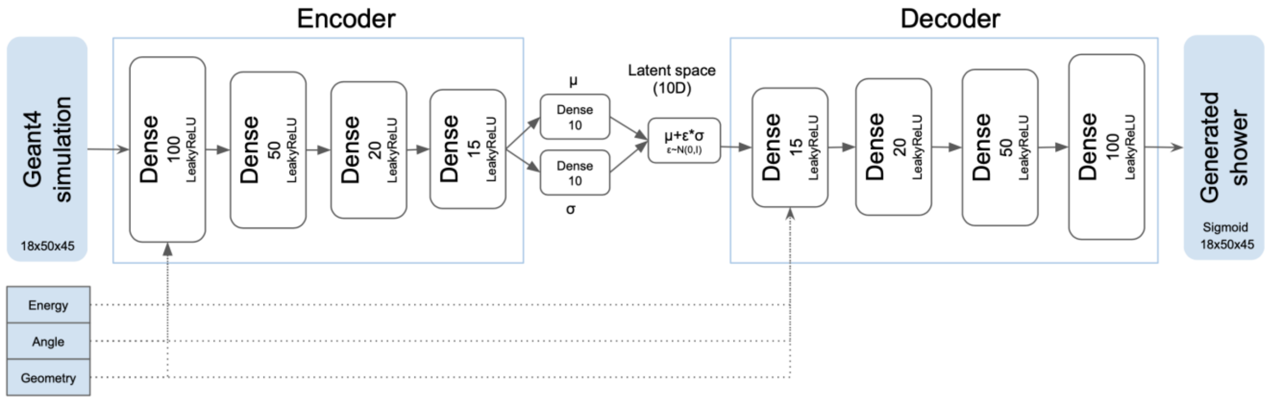


Fig. 2 The VAE architecture used in Par04 example.

2.4 INFERENCE

Model inference can be performed in C++ code, using the same Geant4 application that was used to produce the full simulation data. This demonstrates how to perform fast simulation within a production framework, but also, thanks to the integration with Geant4, how to run a combined full and fast simulation.

Thanks to the application of the same tools to score energy, the output of both simulation modes is identical, which facilitates validation. Necessary changes to Geant4 core code were introduced to make this possible. Fig. 3. shows a validation plot of one of the compared variables, a longitudinal profile (energy distribution along the shower development). The accuracy of this model is satisfactory for the example but may need to be improved before being applied in an experiment, depending on its needs.

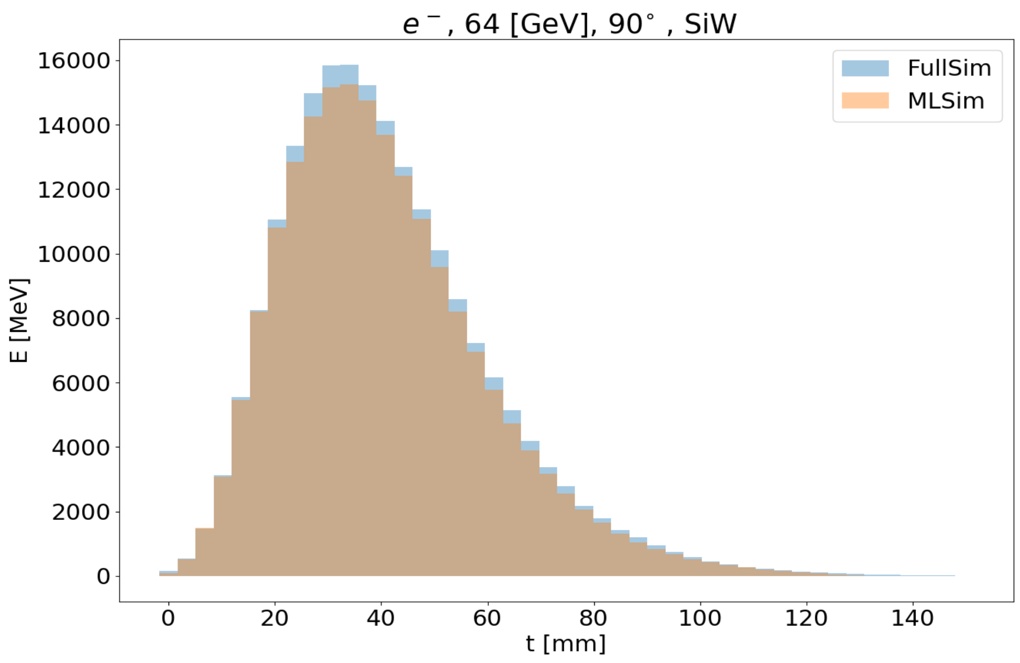


Fig. 3 Validation of ML fast simulation (MLSim) against full simulation (FullSim). The longitudinal profile is shown (energy distribution along the shower development axis) for electrons with energy of 64 GeV. The inference library used in this test is ONNXRuntime.

3. RELEASED EXAMPLE

The example Par04 is released in Geant4 11.0 and is further improved in version 11.1. This includes integration with LibTorch library, a result of a successful AIDAInnova hackathon of WP12 held at CERN in June 2022.

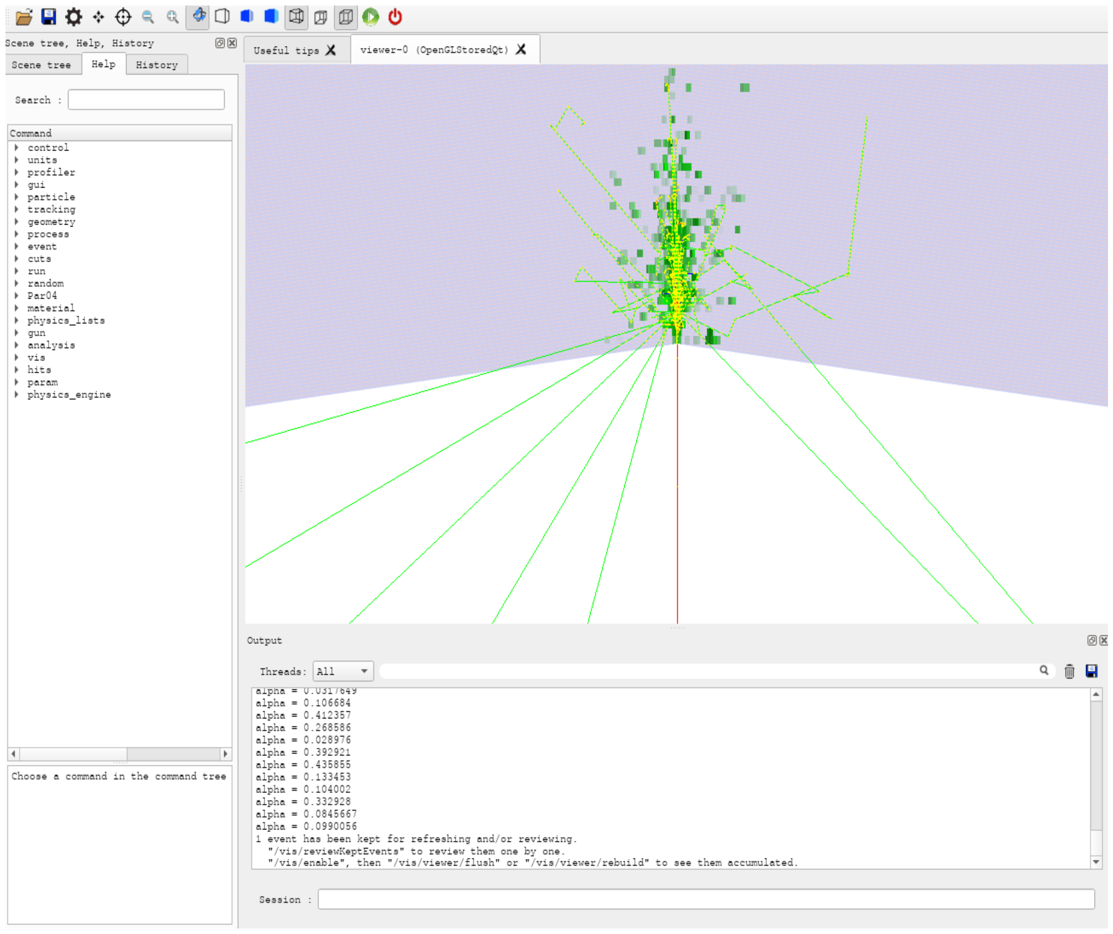
3.1 INSTALLATION

The source code of the example Par04 is distributed with Geant4. It can be installed with any of the inference libraries (LWTNN, ONNX Runtime, LibTorch), or without to run only full simulation. Installation of the inference library will also download a pre-trained ML model that can be used to run fast simulation.

3.2 VISUALIZATION

A first inspection of the example can be done using the visualization. It will show a detector together with the energy deposits. Fig. 4. shows the window that users can see executing:

```
./examplePar04 -i -m vis.mac
/vis/viewer/zoomTo 20
/run/beamOn 1
```



```
alpha = 0.0317649
alpha = 0.106684
alpha = 0.412357
alpha = 0.268586
alpha = 0.028976
alpha = 0.392921
alpha = 0.435855
alpha = 0.133453
alpha = 0.104002
alpha = 0.332928
alpha = 0.0845667
alpha = 0.0990056
1 event has been kept for refreshing and/or reviewing.
  "/vis/reviewKeptEvents" to review them one by one.
  "/vis/enable", then "/vis/viewer/flush" or "/vis/viewer/rebuild" to see them accumulated.
```

Fig. 4 Visualization of full simulation of 10 GeV electron in Par04 example.

3.3 SIMULATION

High statistics simulations can be run using one of the provided configuration files:

- **examplePar04.mac:** Runs full simulation. It will run 100 events with single electrons, 10 GeV and along the y axis.
- **examplePar04_onnx.mac**: Available only if ONNX Runtime is found by cmake at installation. Runs fast simulation with an ML model stored in onnx file.
- **examplePar04_lwtnn.mac:** Available only if LWTNN is found by cmake at installation. Runs fast simulation with an ML model stored in json file.
- **examplePar04_torch.mac**: Available only if LibTorch is found by cmake at installation. Runs fast simulation with an ML model stored in pt file.

3.4 VALIDATION

The output files produced in full and fast simulation have the same format because the energy deposits were placed in the detector and handled by Geant4 in the same manner. This facilitates the validation process and allows visualisation validation plots of the longitudinal shower profile, as presented in Fig. 3.

4. REFERENCES

JOURNAL ARTICLES:

[1] Allison, J., et al. (2016). Recent developments in Geant4. *Nuclear Instruments and Methods in Physics Research Section A (* 835), 186–225. https://doi.org/10.1016/j.nima.2016.06.125

REPORTS:

[4] Aglieri Rinella, G., et al. (2022) *Strategic R&D Programme on Technologies for Future Experiments – Annual Report 2021*, CERN, CERN-EP-RDET-2022-006, p. 110 - 111 https://cdsweb.cern.ch/record/2808204

WORLD WIDE WEB DOCUMENTS:

[2] Salamani, D. and Zaborowska A. (2022) *Geant4 extended example Par04* [online]. Available from: https://gitlab.cern.ch/geant4/geant4/-/tree/geant4-11.1-release/examples/extended/parameterisations/Par04/ [Accessed 6 January 2023].

[3] Salamani, D. and Zaborowska A. (2022) *Machine learning for Fast Shower Simulation in High Energy Physics, Geant4 example Par04* [online]. Available from: https://g4fastsim.web.cern.ch/docs/G4_Inference/G4_examples[Accessed 6 January 2023].

[5] Salamani, D. and Zaborowska, A. (2022). *High Granularity Electromagnetic Calorimeter Shower Images* [Data set]. Zenodo. https://doi.org/10.5281/zenodo.6082201 [Accessed 6 January 2023].

[6] *Faucci Giannelli, M. (2022). First ML FastSim Calo Challenge [online]. Available from:* https://calochallenge.github.io/homepage/ [Accessed 6 January 2023].

ANNEX: GLOSSARY

Acronym	**Definition**
ML	Machine learning
HEP	High energy physics