

# Fast Simulation : from Classical to Machine Learning Models

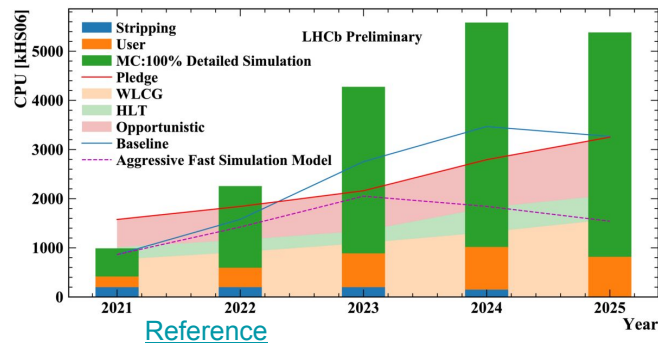
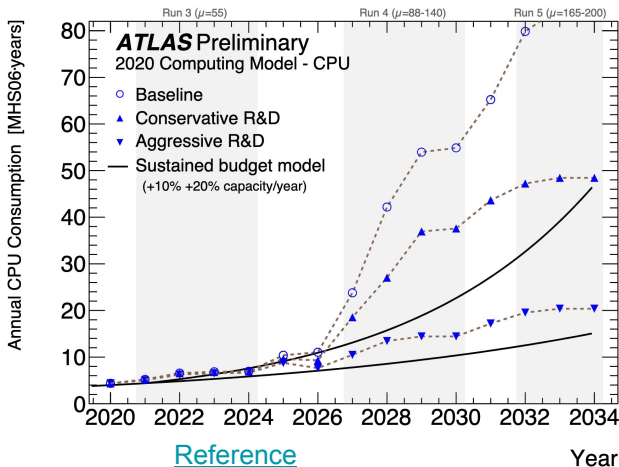
Anna Zaborowska, Witold Pokorski & Dalila Salamani  
CERN EP-SFT

*This work benefited from support by the CERN Strategic R&D Programme on Technologies for Future Experiments ([CERN-OPEN-2018-006](#))*

***Geant4 Collaboration Meeting 16/09/2021***

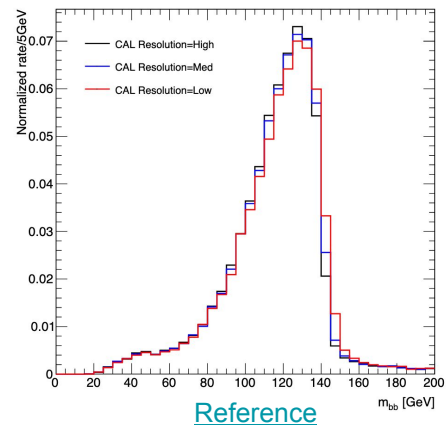
# The need for fast simulation methods

Speed-up simulation to generate more data within the same CPU time

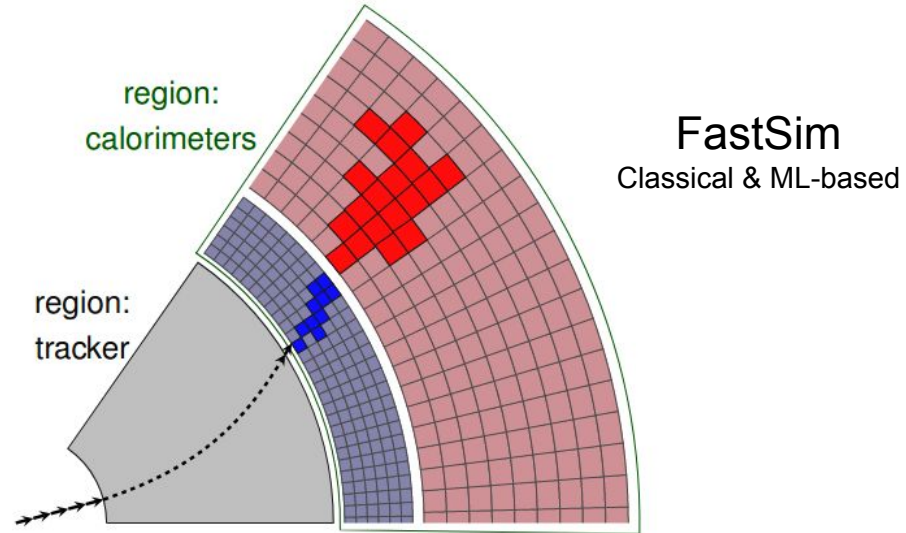
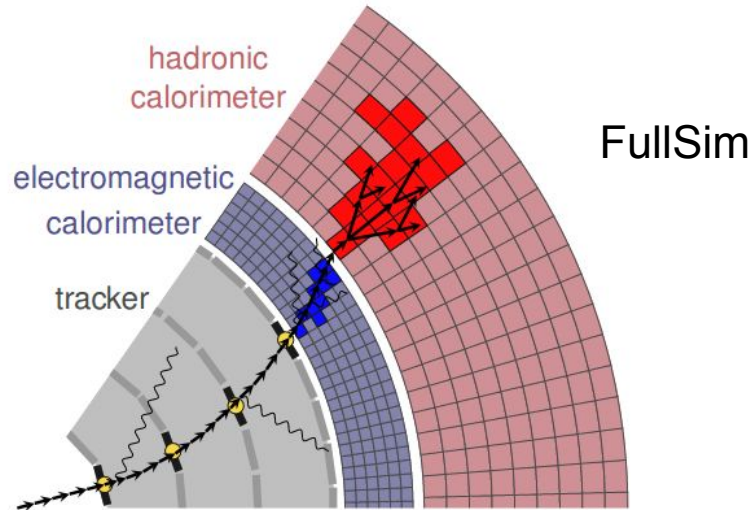


Physics studies & detector performance benchmarks

HL-LHC : simulation of more complex events



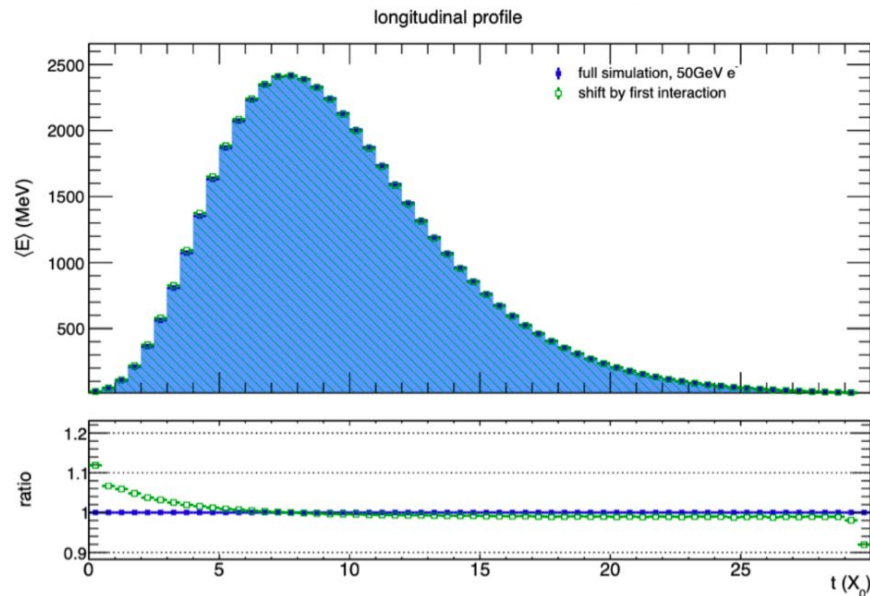
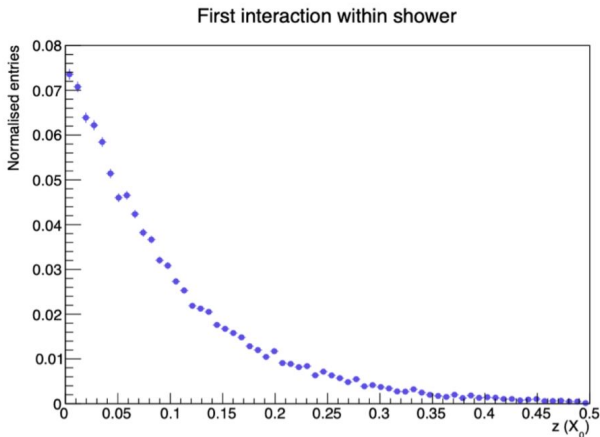
# How to fast simulate particles in Geant4?



Shortcut standard tracking & detailed simulation

# FastSim: Classical Parametrization

- Continuous integration of tuning tools and generalization procedures
- Tuning procedures of parameters of [GFlash](#)-based models
  - Start of shower tuning
  - Transverse shower profile tuning



# FastSim: Classical Parametrization

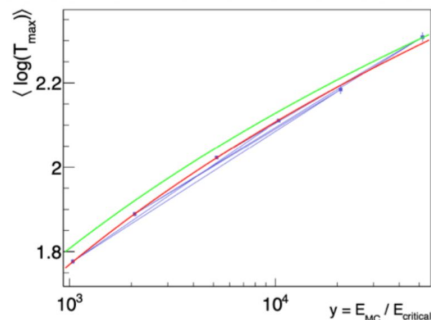
$$f(t) = \frac{((\alpha - 1) t)^{\alpha-1} (\alpha - 1) \exp^{-\beta t}}{T \cdot \Gamma(\alpha)}$$

Original GFlash parameters

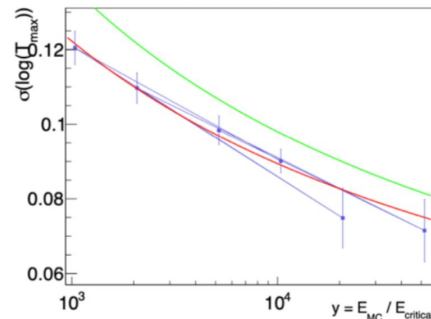
Fitted function

log T, log  $\alpha$  extracted from single particle  
longitudinal profiles for PBWO<sub>4</sub>

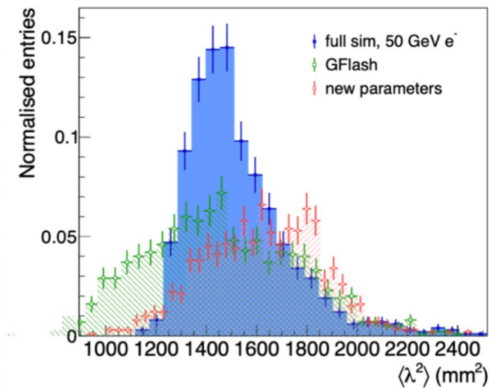
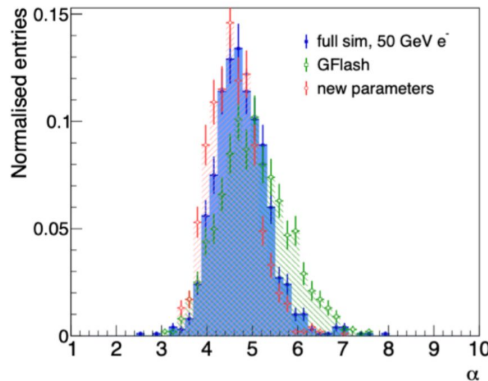
Longitudinal profile not improved immensely  
but extracted parameters ( $T, \alpha$ ), first and  
second moments are closer to full sim than  
GFlash



longitudinal profile fit - alpha parameter

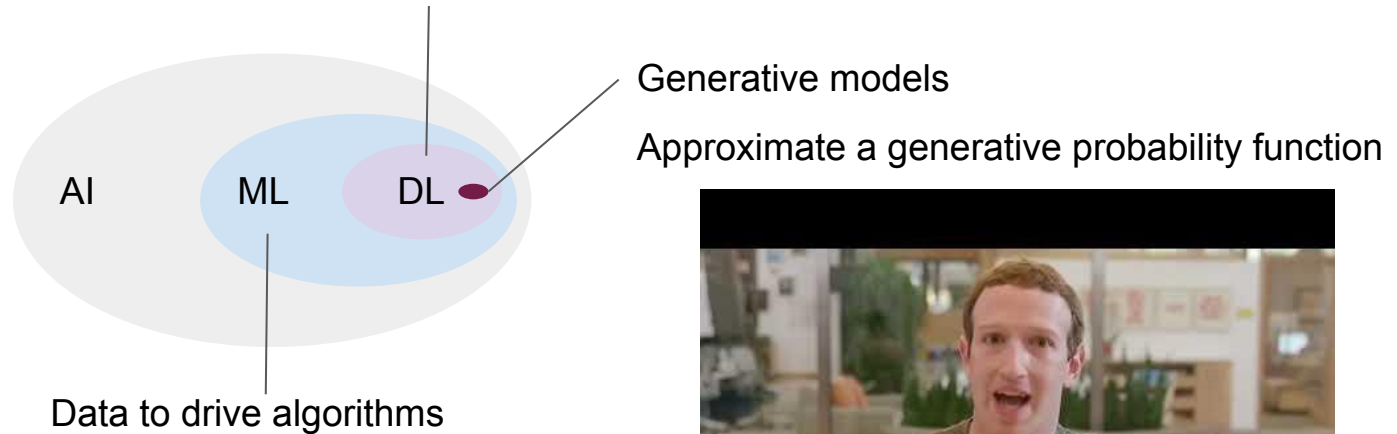


longitudinal second moment



# From AI to Generative Models

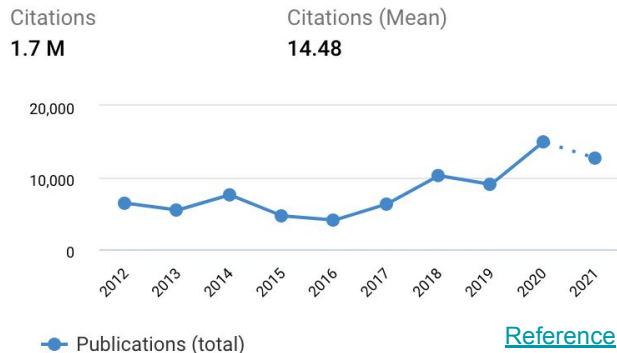
Leveraging cutting edge ML algorithms inspired by artificial  
Neural Networks



*AI : Artificial Intelligence*  
*ML: Machine Learning*  
*DL: Deep Learning*



# Fast Simulation & Generative Models



PUBLICATIONS 117,468

DATASETS 7

GRANTS 10

PATENTS 5,001

CLINICAL TRIALS 0

POLICY DOCUMENTS 216

(1/7)

## Data Augmentation at the LHC through Analysis-specific Fast Simulation with Deep Learning: W+jet training/test dataset

Pierini, Maurizio, Chen, Cheng

2020 - Zenodo

W+jet events at generator and reconstruction level, used to train analysis-specific generative models. Events are represented as an array of relevant high-level features. Reco objects are matched to Gen objects and a minimal selection is applied to define the generator support in the N-dim space identified by the input features. About 2M events, used for training/validation/testing Details in <https://arxiv.org/abs/2010.01835> [less](#)

## Three Dimensional Energy Parametrized Generative Adversarial Networks for Electromagnetic Shower Simulation

Gul Rukhkhata, Sofia Vallecorsa, Federico Carminati

2018, 2018 25th IEEE International Conference on Image Processing (ICIP) - Proceeding

High Energy Physics (HEP) simulations are traditionally based on the Monte Carlo approach and generally rely on time consuming calculations. The present work investigates the use of Generative Adversarial Networks (GANs) for electromagnetic shower simulation. [more](#)

Citations 11 Add to Library

## Generative models for fast cluster simulations in the TPC for the ALICE experiment

Kamil Deja, Tomasz Trzcinski, Lukasz Graczykowski

2019, EPJ Web of Conferences - Article

Simulating the detector response is a key component of every highenergy physics experiment. The methods used currently for this purpose provide high-fidelity results. However, this is often computationally expensive. [more](#)

Citations 8 View PDF Add to Library

## Deep Generative Models for Fast Shower Simulation in ATLAS

Dallia Salamani, Stefan Gadatsch, Tobias Golling, Graeme Andrew Stewart, Aishik Ghosh, David Rousseau, Ahmed Hasib, Jana S...

2018, 2018 IEEE 14th International Conference on e-Science (e-Science) - Proceeding

Detectors of High Energy Physics experiments, such as the ATLAS detector [1] at the Large Hadron Collider [2], serve as cameras that take pictures of the particles produced in the collision events. [more](#)

Citations 10 Open Access Add to Library

## Fast and Accurate Simulation of Particle Detectors Using Generative Adversarial Networks

Pasquale Musella, Francesco Pandolfi

2018, Computing and Software for Big Science - Article

Deep generative models parametrised by neural networks have recently started to provide accurate results in modeling natural images. In particular, generative adversarial networks provide an unsupervised way to generate realistic images. [more](#)

Citations 33 Altmetric 10 View PDF Add to Library

## Generative Adversarial Networks for LHCb Fast Simulation

Fedor Ratnikov

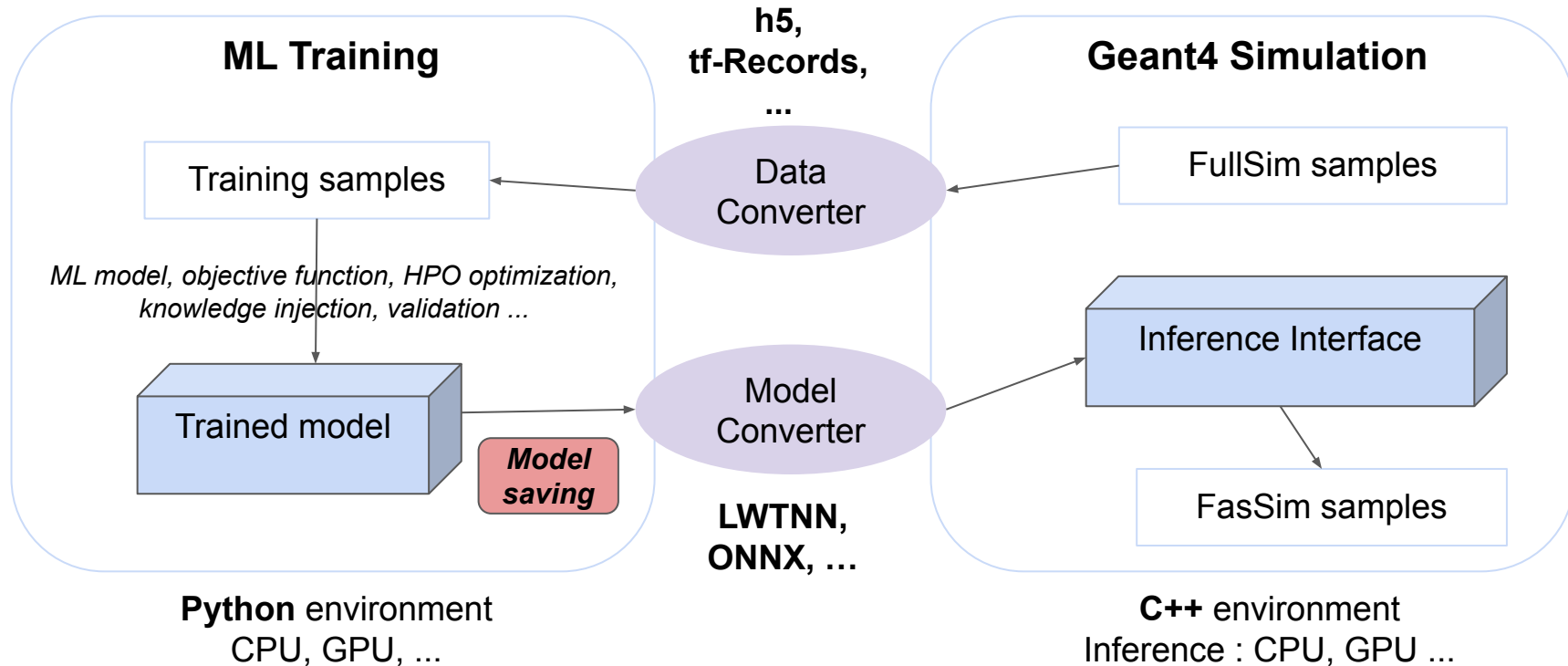
2020, arXiv - Preprint

LHCb is one of the major experiments operating at the Large Hadron Collider at CERN. The richness of the physics program and the increasing precision of the measurements in LHCb lead to the need of even more efficient simulation tools. [more](#)

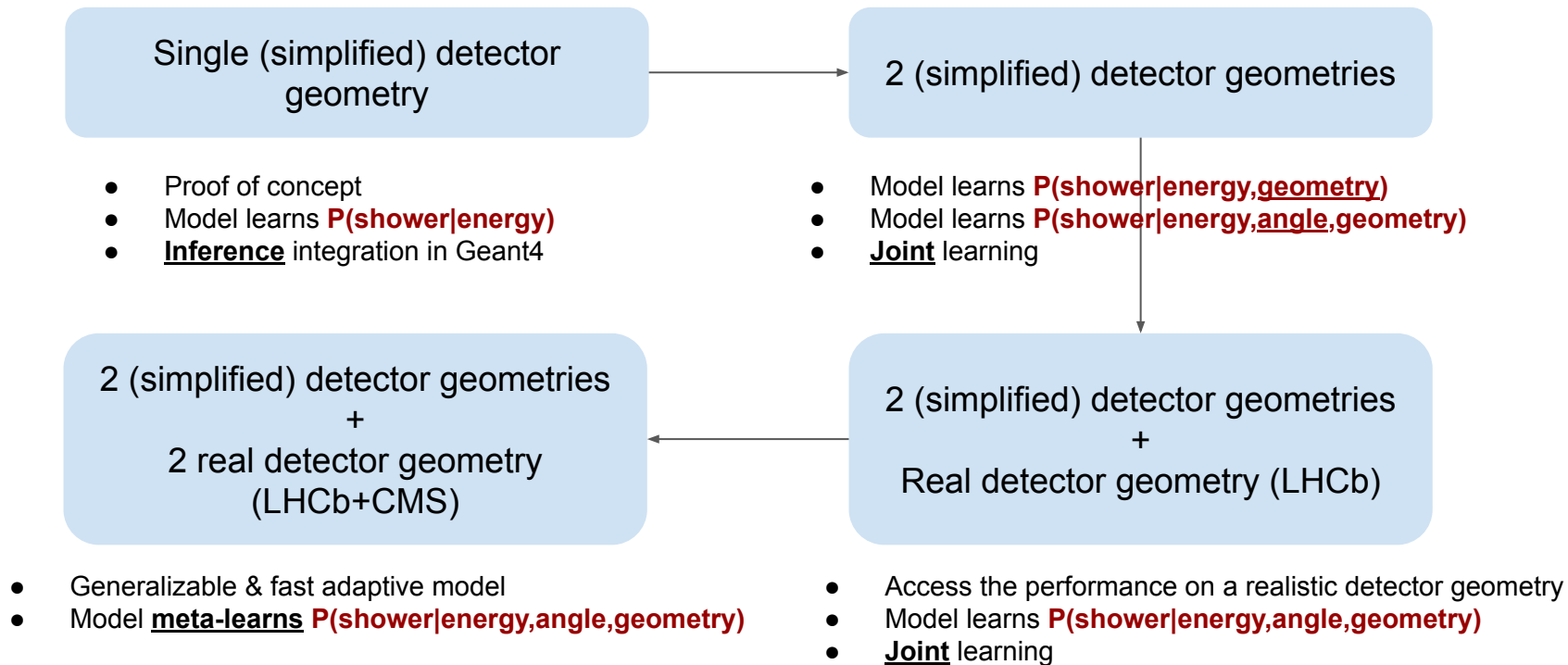
Altmetric 1 Add to Library

and many more ....

# From ML training to Geant4 fast simulation



# Towards a generalizable and a fast adaptive ML simulator



# One model to learn from different tasks/domains ?

## One Model To Learn Them All

**Lukasz Kaiser**

Google Brain

lukaszkaier@google.com

**Aidan N. Gomez\***

University of Toronto

aidan@cs.toronto.edu

**Noam Shazeer**

Google Brain

noam@google.com

**Ashish Vaswani**

Google Brain

avaswani@google.com

**Niki Parmar**

Google Research

nikip@google.com

**Llion Jones**

Google Research

llion@google.com

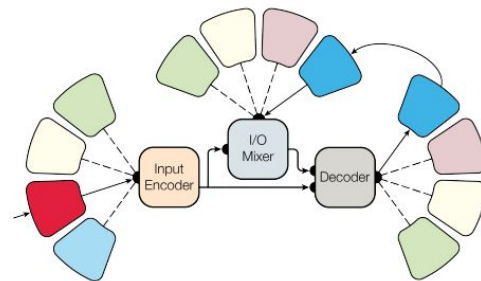
**Jakob Uszkoreit**

Google Research

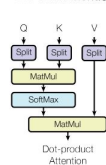
usz@google.com

### Abstract

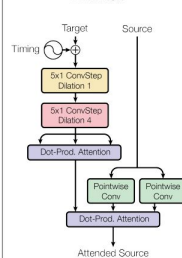
Deep learning yields great results across many fields, from speech recognition, image classification, to translation. But for each problem, getting a deep model to work well involves research into the architecture and a long period of tuning. We present a single model that yields good results on a number of problems spanning multiple domains. In particular, this single model is trained concurrently on ImageNet, multiple translation tasks, image captioning (COCO dataset), a speech recognition corpus, and an English parsing task. Our model architecture incorporates building blocks from multiple domains. It contains convolutional layers, an attention mechanism, and sparsely-gated layers. Each of these computational blocks is crucial for a subset of the tasks we train on. Interestingly, even if a block is not crucial for a task, we observe that adding it never hurts performance and in most cases improves it on all tasks. We also show that tasks with less data benefit largely from joint training with other tasks, while performance on large tasks degrades only slightly if at all.



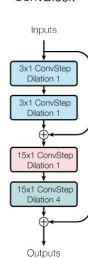
Dot-Prod. Attention



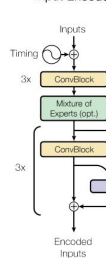
Attention



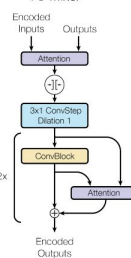
ConvBlock



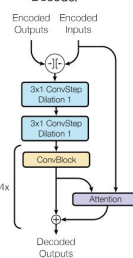
Input Encoder



I/O Mixer



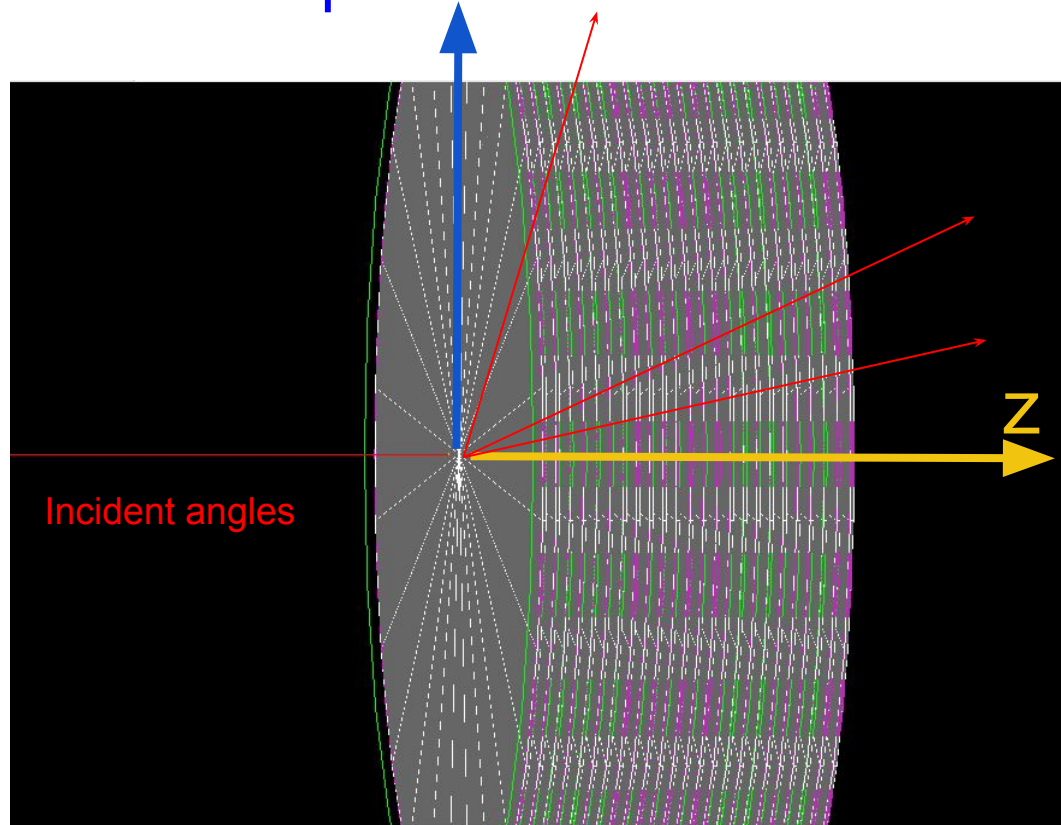
Decoder



## 2-detector geometries : Geant4 samples

- 3D readout geometry, electromagnetic calorimeters
  - Lead tungstate ( $\text{PbWO}_4$ )
  - Silicon-tungsten (SiW)
- Flat energy samples 1-500 GeV
- Incident angle from  $0^\circ$  to  $90^\circ$

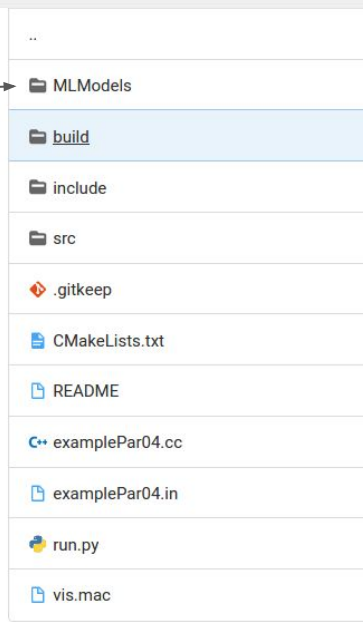
$P(\text{shower} \mid \text{energy}, \text{angle}, \text{geometry})$



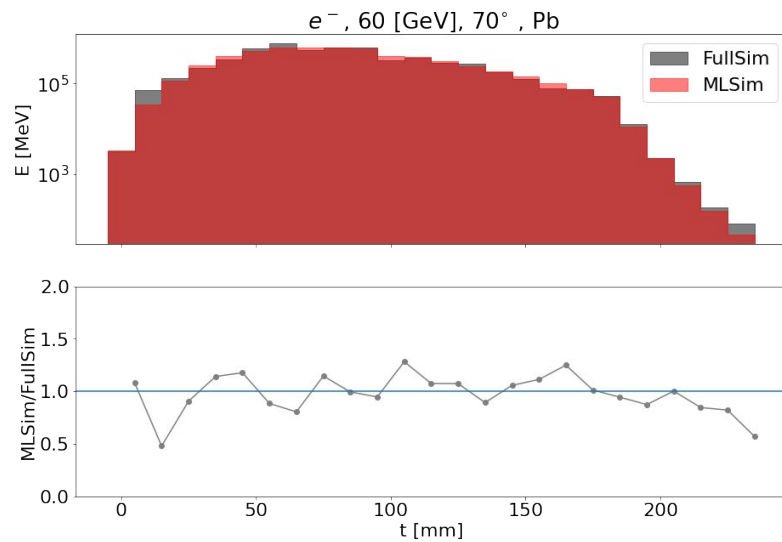
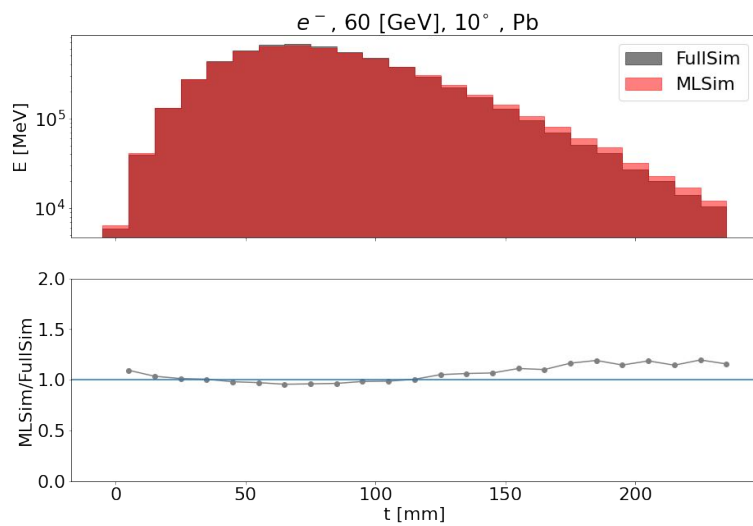
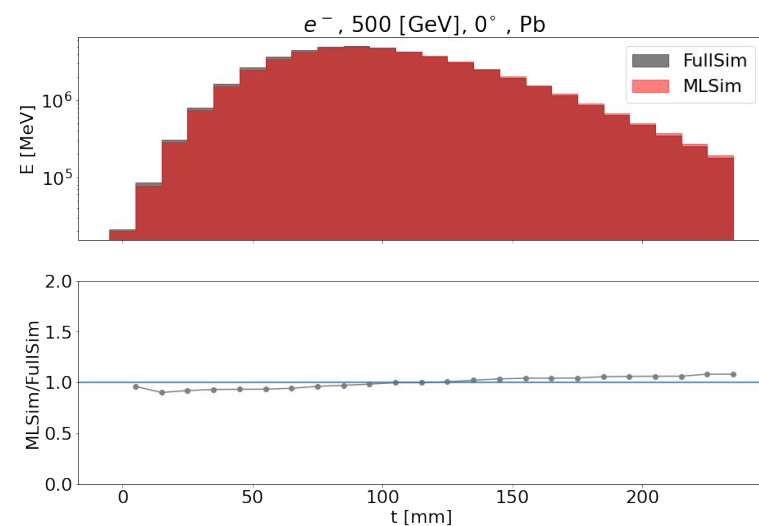
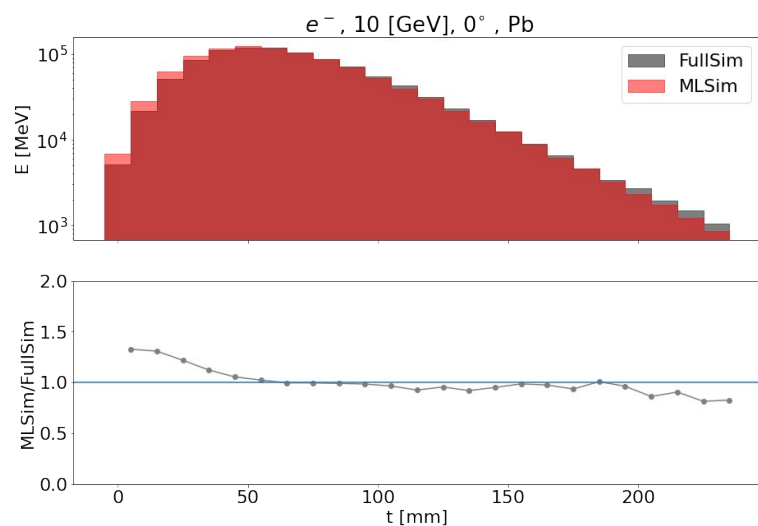
# How to use the trained ML model for inference in Geant4?

- Demo example:  
[https://gitlab.cern.ch/dasalama/ag\\_ml\\_sim/-/tree/agmlsim\\_test\\_Pb\\_Geo/Geant4MLFastSim\\_Par04](https://gitlab.cern.ch/dasalama/ag_ml_sim/-/tree/agmlsim_test_Pb_Geo/Geant4MLFastSim_Par04)
- **src/Par04DetectorConstruction.cc**
  - Cylindrical detector with r,phi,z segmentation
  - Region for the detector is created as an envelope of the fast simulation
- **src/Par04InferenceInterface.cc**
  - Load the model [Generator.onnx](#) from
  - Configure and run inference
- Root files full/fastSim are available [here](#)
- Validation plots: longitudinal and transverse profiles (+time) for different energies, angles are available in this [notebook](#)

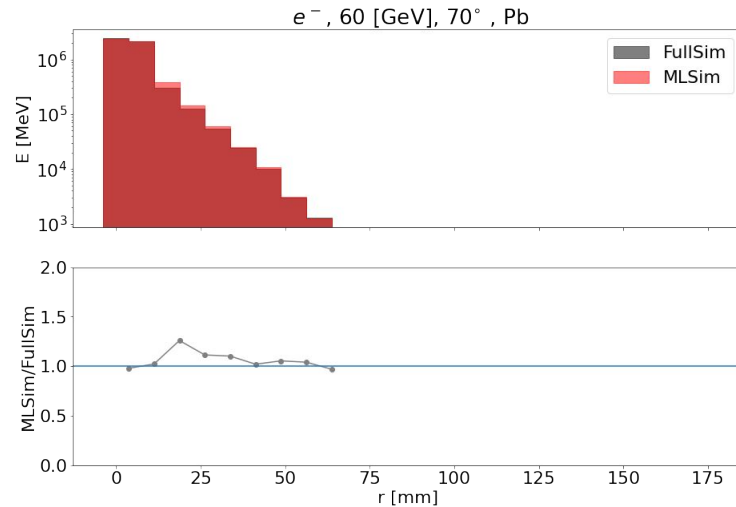
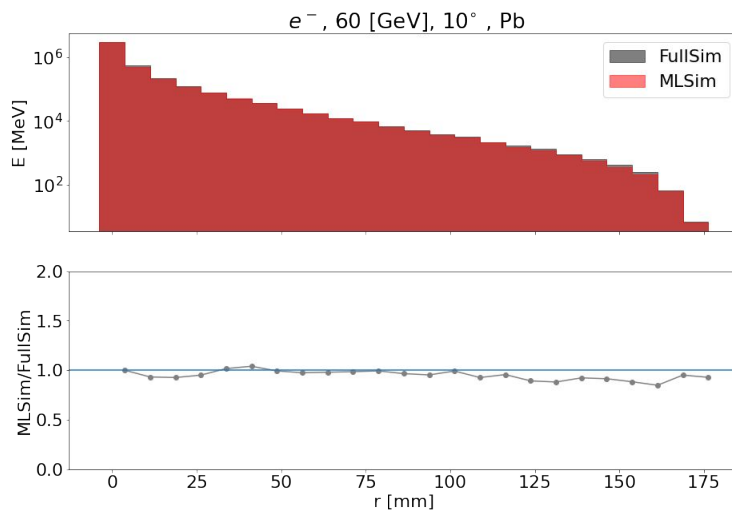
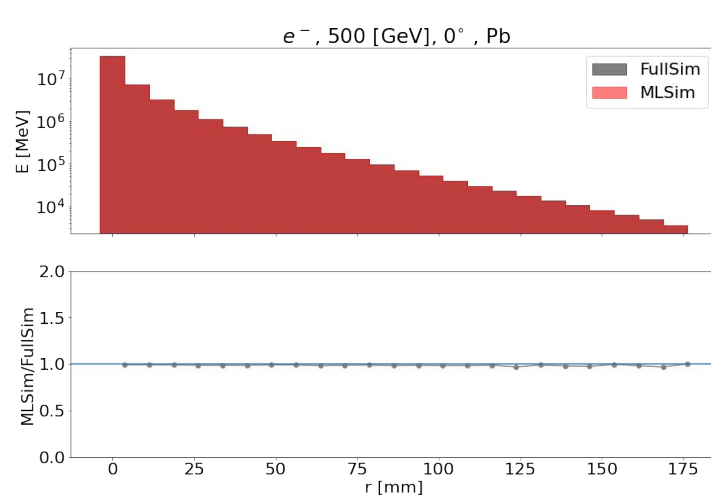
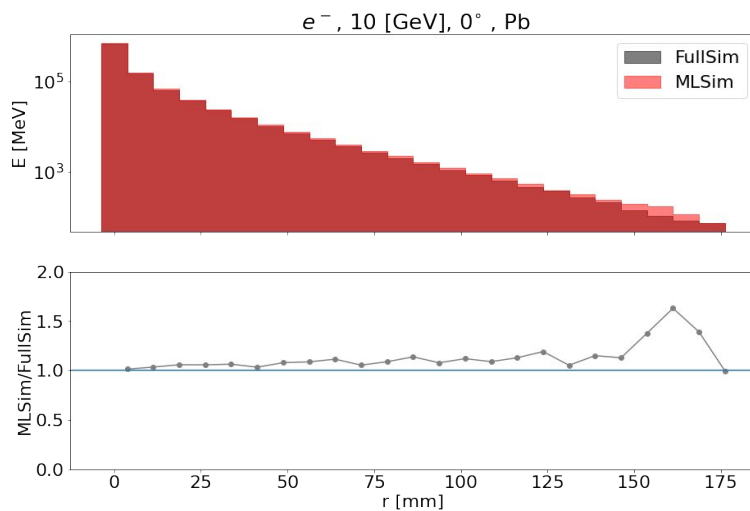
```
# Abbreviations: PV = Physical Volume, LV = Logical Volume,
# SD = Sensitive Detector, RO = Read Out Geometry.
"World":0 / "World"
"Detector":80 / "Detector"
"Layer":0-23 (24 replicas) / "Layer"
"G4_PbWO4":0 / "G4_PbWO4"
"PhiCell":0-23 (24 replicas) / "PhiCell"
"Cell":0-23 (24 replicas) / "Cell" (SD="sensitiveDetector")
G4ASCIITreeSceneHandler::EndModeling
Reverting to viewer-0 (OpenGLStoredQt)
```



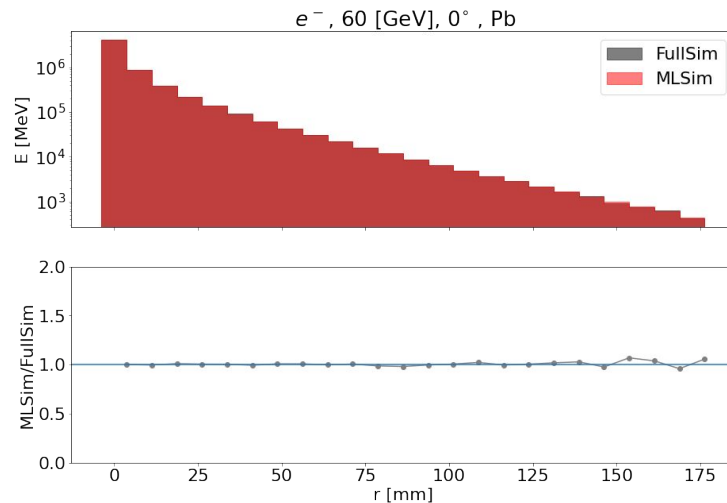
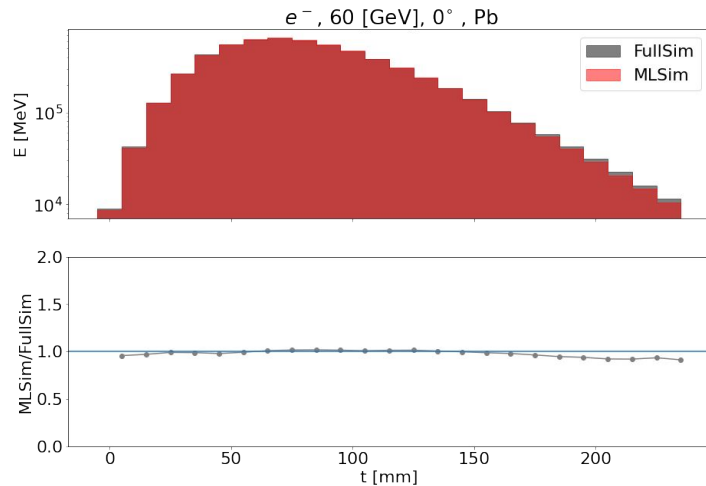
# Longitudinal profiles PBWO<sub>4</sub>



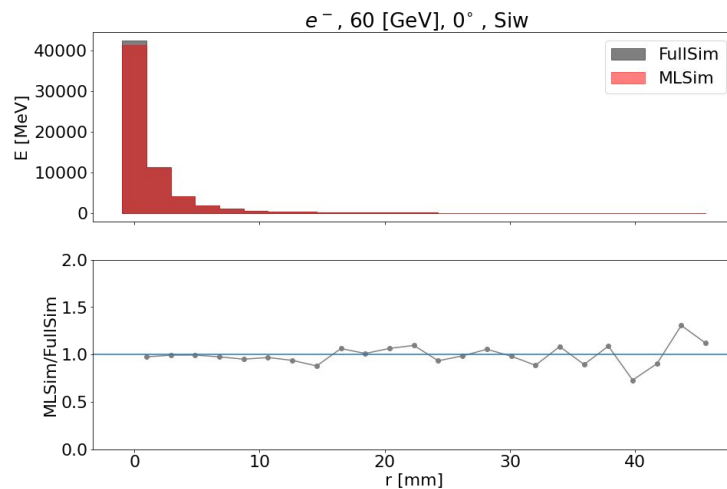
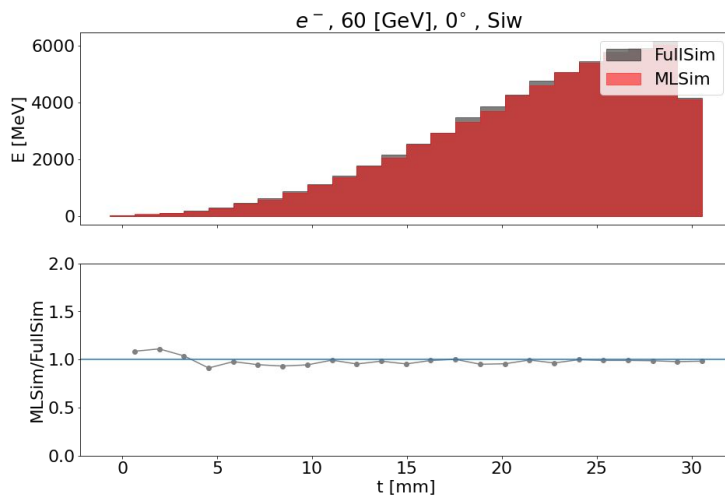
# Transverse profiles PBW<sub>4</sub>



PBWO<sub>4</sub>



SiW



# Geant4 Inference Interface

Interface that allows to read in ML models, configure, and execute inference.

Two main functions :

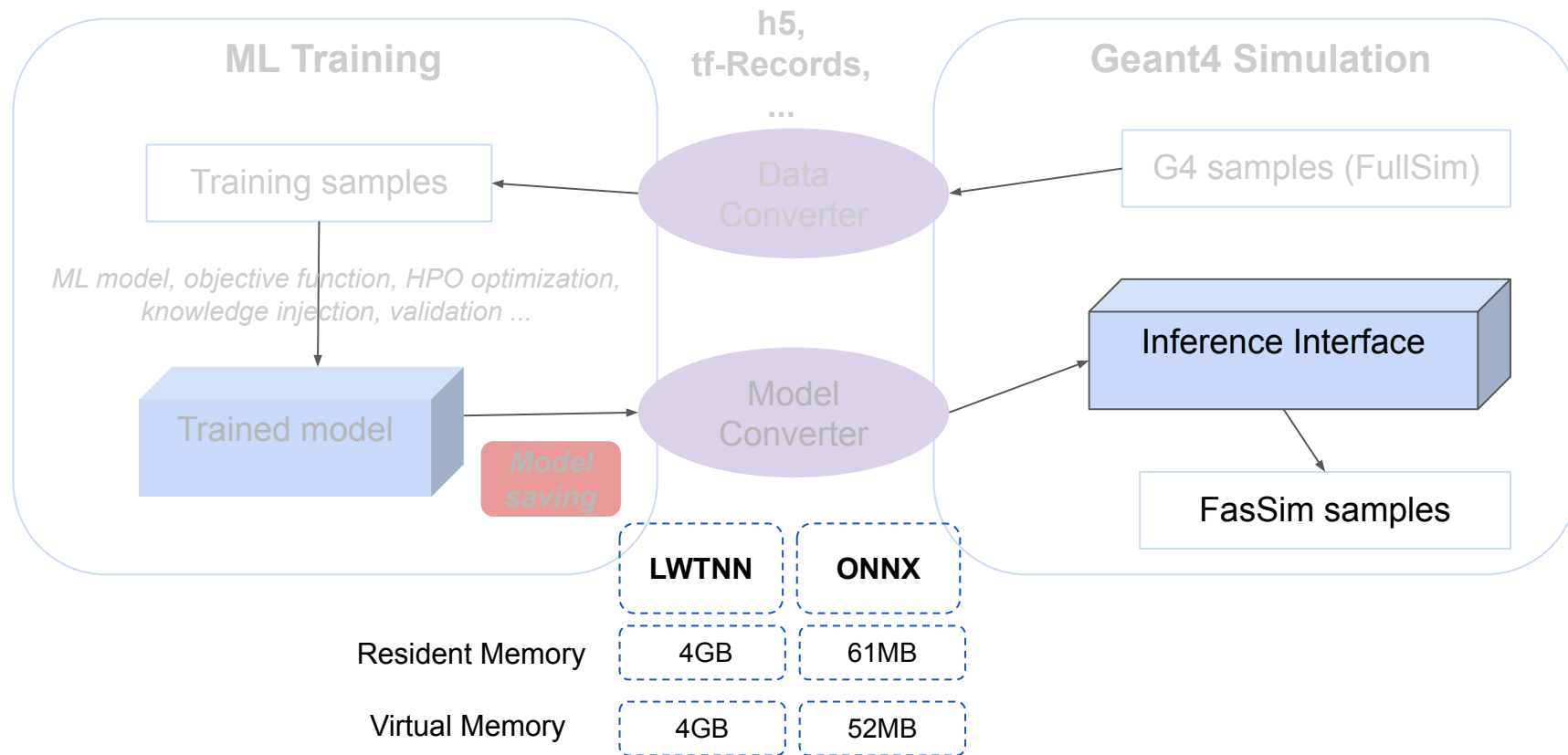
```
void GetEnergies(std::vector<G4double>& aDepositsEnergies,  
                 G4double aParticleEnergy);
```

*Infer energies deposited in the  
detector*

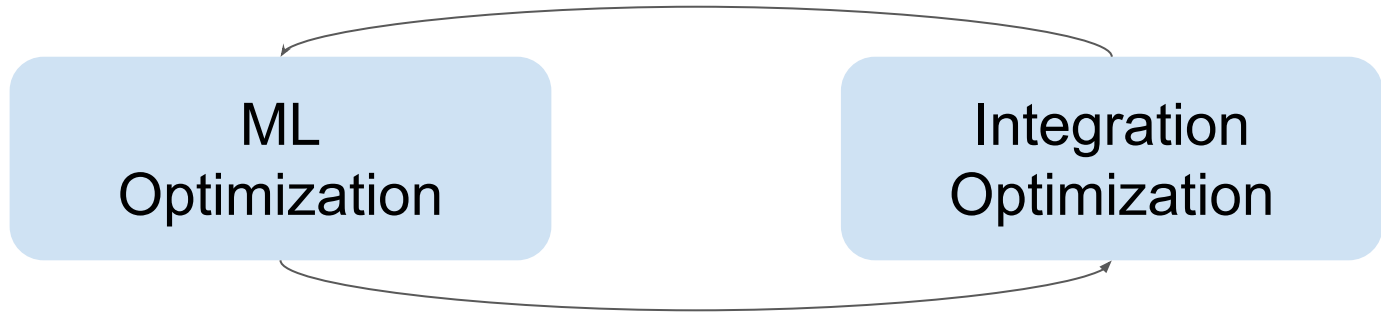
```
void GetPositions(std::vector<G4ThreeVector>& aDepositsPositions,  
                  G4ThreeVector aParticlePosition,  
                  G4ThreeVector aParticleDirection);
```

*Calculate positions to  
corresponding energies in the  
detector*

# From ML training to Geant4 fast simulation



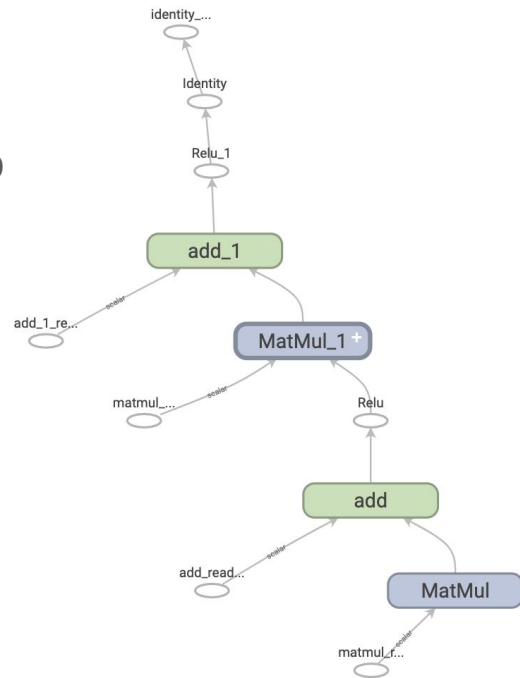
# Optimizing the memory footprint



- Using a highly granular calorimeter -> more inputs to the model -> larger network -> more parameters -> larger memory footprint
- The memory footprint can be optimized
  - ML optimization : to reduce the number of trainable parameters
  - Integration optimization : to reduce the complexity of the model representation

# Integration optimization : Graph optimization

- Graphs : as data structures
- ONNX Runtime provides various graph optimizations to improve model performance.
- Graph optimizations graph-level transformations
  - **Basic Graph Optimizations** : remove redundant nodes and redundant computation
  - **Extended Graph Optimizations**: fuse nodes



# Integration optimization : Quantization & Graph Optimization

- Quantization in ONNX Runtime refers to 8 bit linear quantization
- Floating point real values are mapped to an 8 bit quantization space

		Raw Model (without optimization)	Quantization	Quantization + Graph Optimization
Loading + Inference	Resident memory (MB)	2265.34	650.414	555.828
	Virtual memory(MB)	3205.26	1339.22	1073.21



Very large network

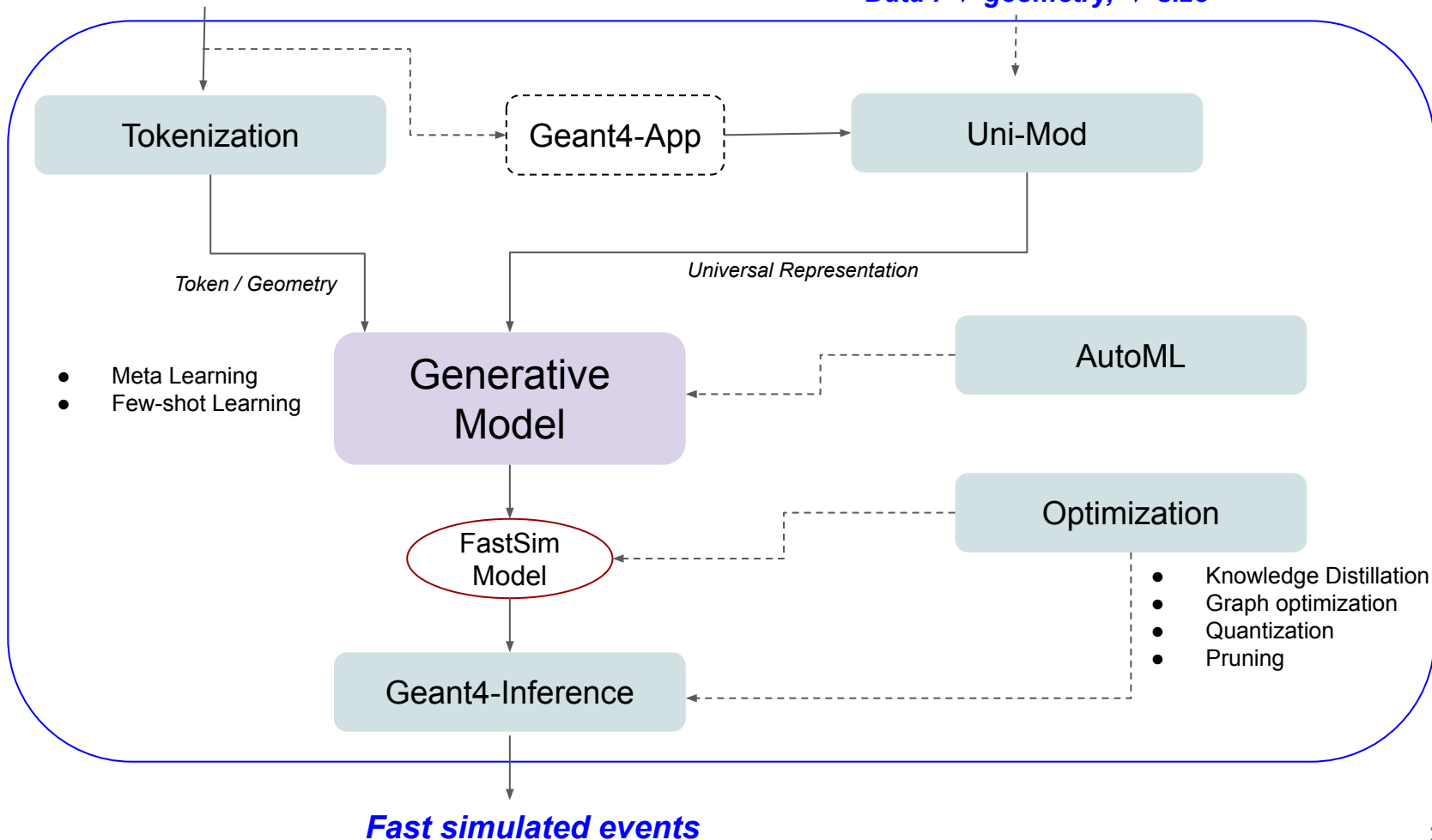
# Towards a generalizable generative modeling

- Previously...

- ML model learns  $P(\text{shower} \mid \text{energy, angle, geometry})$  (joint training) -> can't adapt quickly to a new geometry
  - Geometry encoded as one hot vector -> adding a new geometry would require changing the encoding and retraining

- Work in progress

- Adding the **CMS** (ECAL) to the set of geometries
- Re-design of the key components for generalizability and adaptability
- ML model meta-learns  $P(\text{shower} \mid \text{energy, angle, geometry})$



# Meta Learning : learning-to-learn “Fast”

## On First-Order Meta-Learning Algorithms

Alex Nichol and Joshua Achiam and John Schulman

OpenAI

{alex, jachiam, joschu}@openai.com

[arXiv:1803.02999](https://arxiv.org/abs/1803.02999)

### Abstract

This paper considers meta-learning problems, where there is a distribution of tasks, and we would like to obtain an agent that performs well (i.e., learns quickly) when presented with a previously unseen task sampled from this distribution. We analyze a family of algorithms for **learning a parameter initialization** that can be fine-tuned quickly on a new task, using only first-order derivatives for the meta-learning updates. This family includes and generalizes first-order MAML, an approximation to MAML obtained by ignoring second-order derivatives. It also includes Reptile, a new algorithm that we introduce here, which **works by repeatedly sampling a task, training on it, and moving the initialization towards the trained weights on that task.** We expand on the results from Finn et al. showing that first-order meta-learning algorithms perform well on some well-established benchmarks for few-shot classification, and we provide theoretical analysis aimed at understanding why these algorithms work.

MAML

### Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks

Chelsea Finn, Pieter Abbeel, Sergey Levine

[arXiv:1703.03400](https://arxiv.org/abs/1703.03400)

We propose an algorithm for meta-learning that is model-agnostic, in the sense that it is compatible with any model trained with gradient descent and applicable to a variety of different learning problems, including classification, regression, and reinforcement learning. The goal of meta-learning is to train a model on a variety of learning tasks, such that it can solve new learning tasks using only a small number of training samples. In our approach, the parameters of the model are explicitly trained such that a small number of gradient steps with a small amount of training data from a new task will produce good generalization performance on that task. In effect, our method trains the model to be easy to fine-tune. We demonstrate that this approach leads to state-of-the-art performance on two few-shot image classification benchmarks, produces good results on few-shot regression, and accelerates fine-tuning for policy gradient reinforcement learning with neural network policies.

---

### Algorithm 1 Reptile (serial version)

---

Initialize  $\phi$ , the vector of initial parameters

**for** iteration = 1, 2, ... **do**

    Sample task  $\tau$ , corresponding to loss  $L_\tau$  on weight vectors  $\tilde{\phi}$

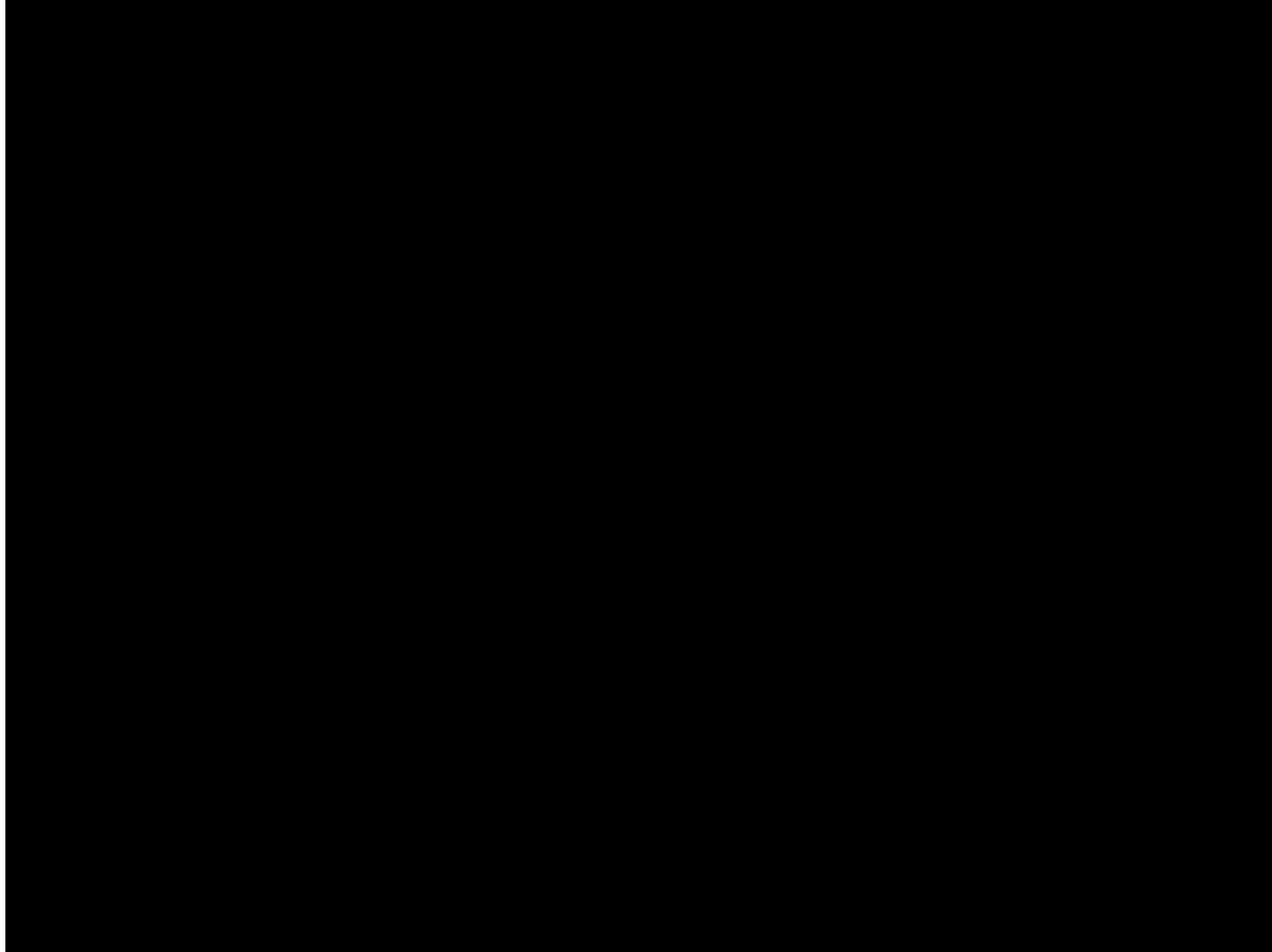
    Compute  $\tilde{\phi} = U_\tau^k(\phi)$ , denoting  $k$  steps of SGD or Adam

    Update  $\phi \leftarrow \phi + \epsilon(\tilde{\phi} - \phi)$

**end for**

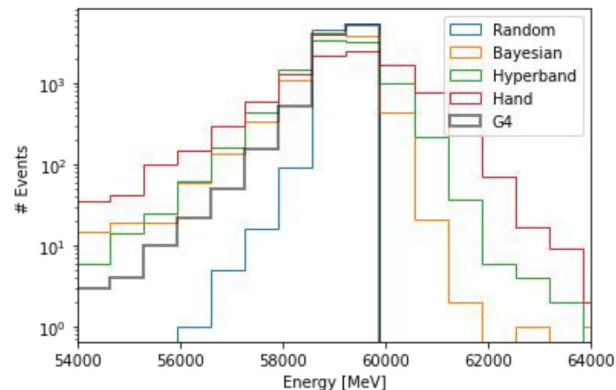
---

## Learning from few examples



# Automatic Machine Learning (AutoML)

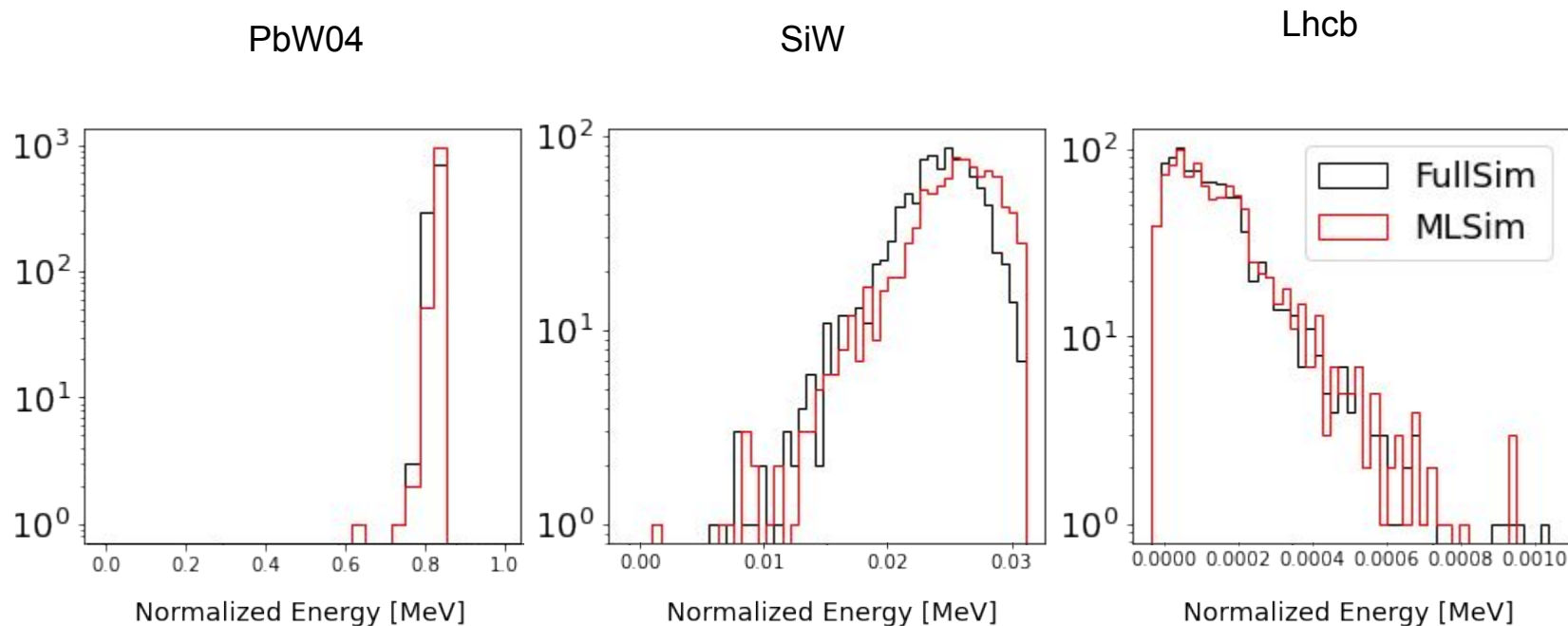
- Summer project
  - Report by Poliana Nascimento Ferreira:  
<https://indico.cern.ch/event/1060366/>
- Automatically search the best hyperparameters according to a metric
  - Combined ML-Physics metric (image reconstruction, total energy, energy per layer)
- Compared to a grid search AutoML has the advantage of changing more than one hyperparameter at the same time
- AutoML approaches:



	Random Search	Bayesian Optimization	Hyperband
+	Simple	Tracking of the tuning process Approximate objective function for fast scoring	Fast Explore larger space Tracking of the tuning process
-	No control of the tuning process	Too long to approximate a good enough function	Discard some trials too fast

# Meta-learning model performance : total energy

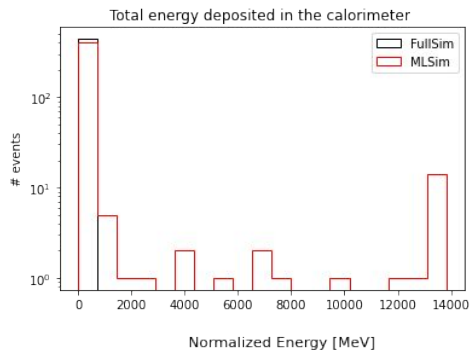
Energy 60 GeV  
Angle  $20^\circ$



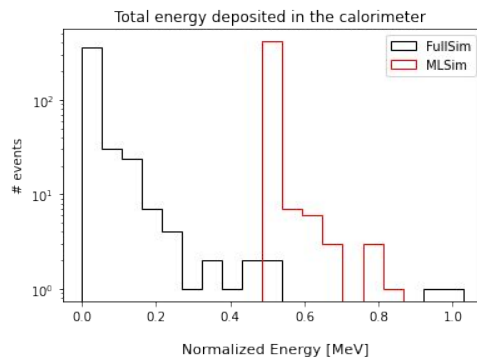
# Test on new geometry : CMS

Energy 60 GeV  
Angle  $20^\circ$

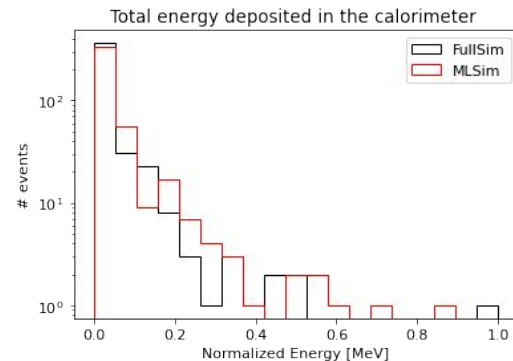
0 iteration



After 100 iterations



After 1000 iterations



3mn on CPU, local machine (4 cors, 16Gb memory)

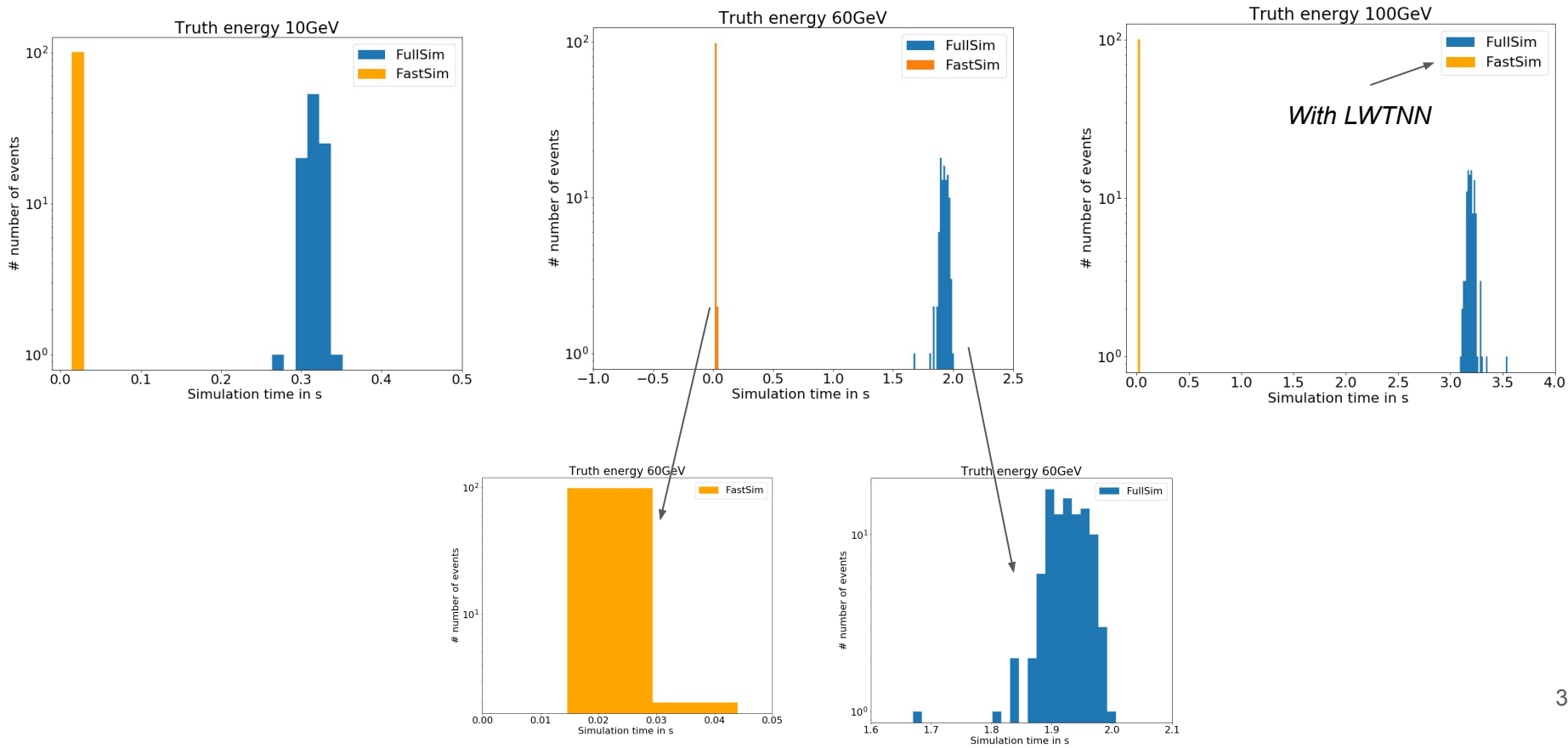
# Summary & Outlook

- Continuation of tuning of parameters of GFlash-inspired models
- ML-Multi-detector geometry model
  - Conditioned on the geometry, energy and incident angle of the particle
  - First tested on 2 simplified geometries
  - Currently testing on a real LHC experiment detectors
    - Meta-learning with few shot learning is a very promising approach towards a generalizable simulator
    - Many improvements are expected
    - More geometries will be tested and evaluated
- Geant4 inference integration
  - Provide G4 examples extending its simulation facilities to Classical & ML-based methods
  - For ML-based fast simulation
    - Compare inference libraries such as LWTNN, ONNX
    - To better optimize the memory footprint with ONNX
      - Graph optimizations
      - Quantization

*Thank you for your attention!* 28

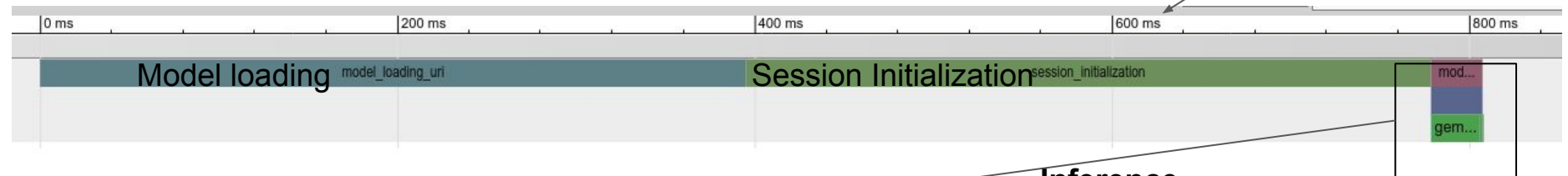
# Backup

# Geant4 Inference Interface : simulation time

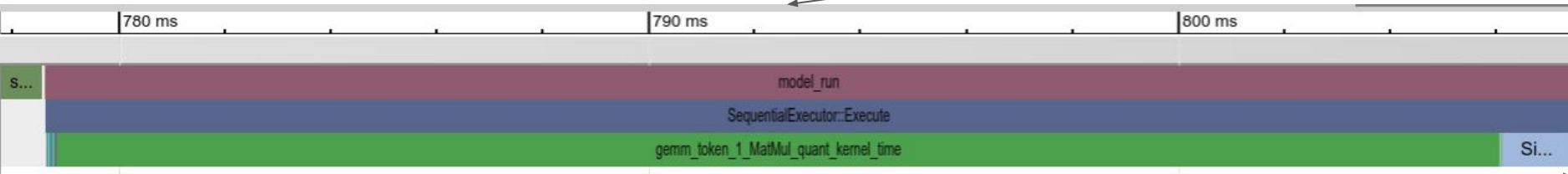


# Model profiling : inference on a single event

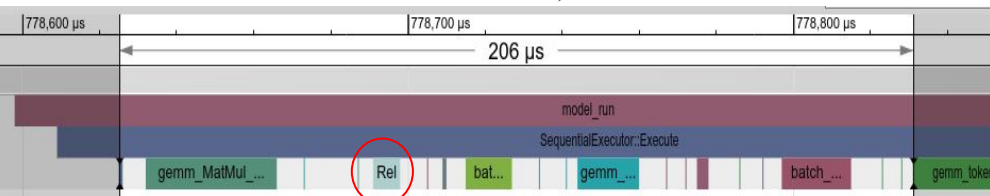
Time



Inference

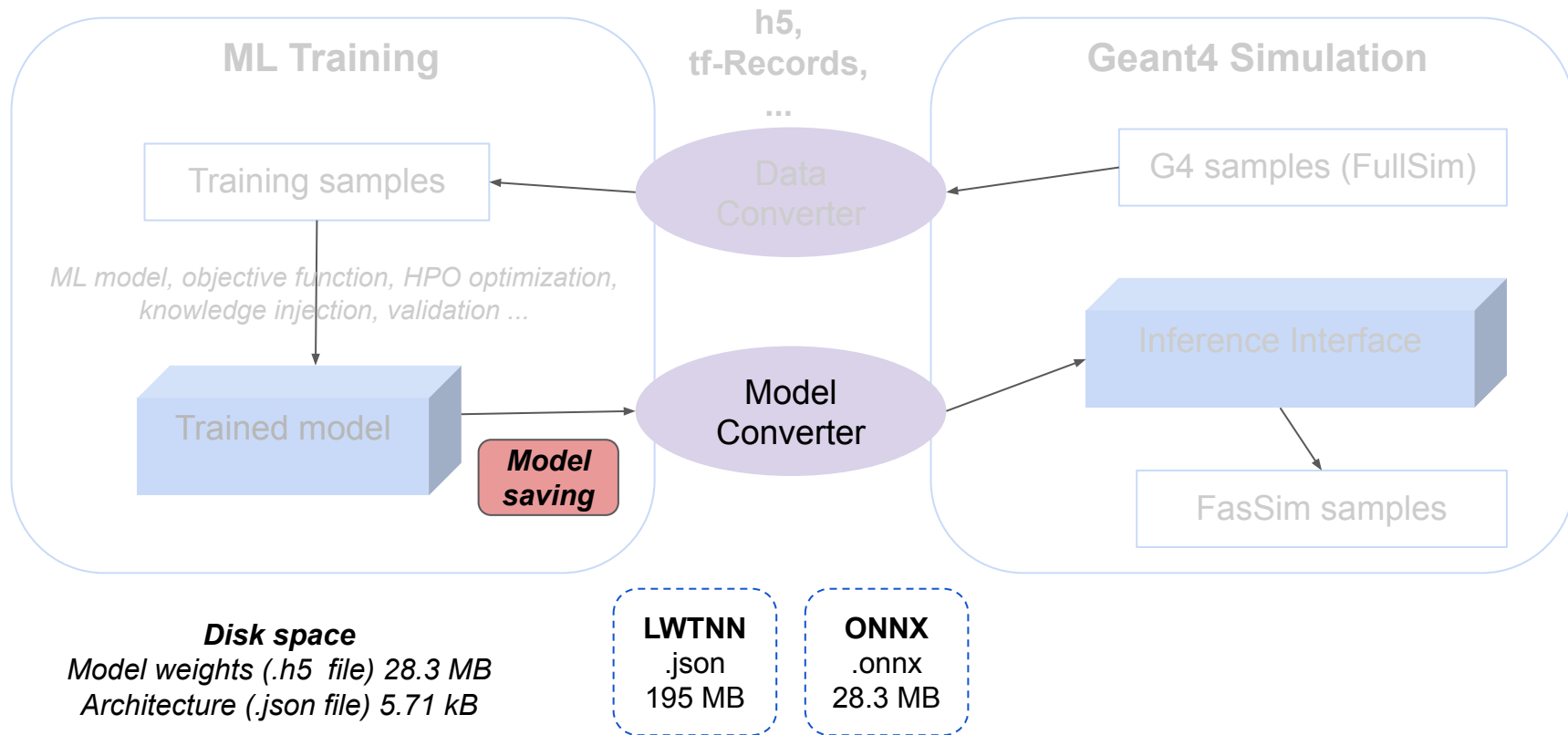


28 ms

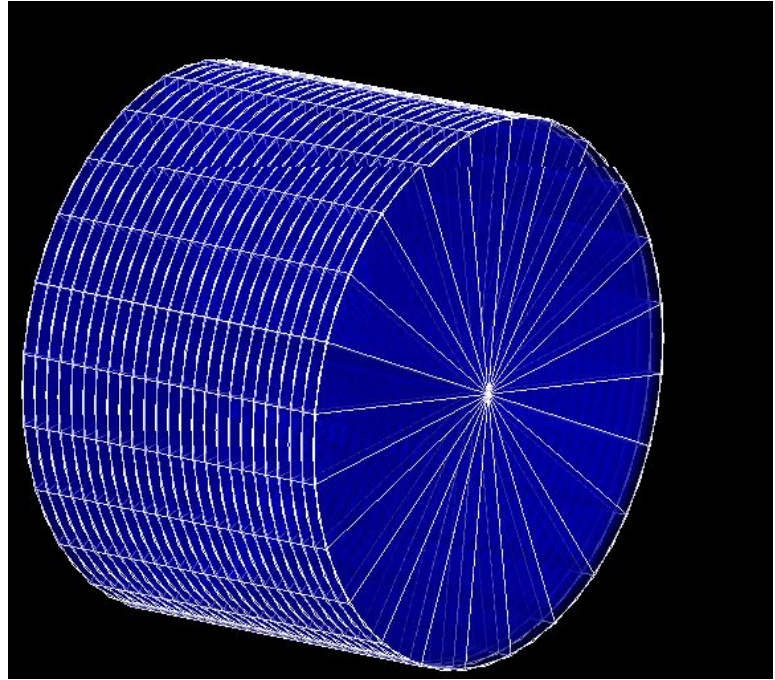
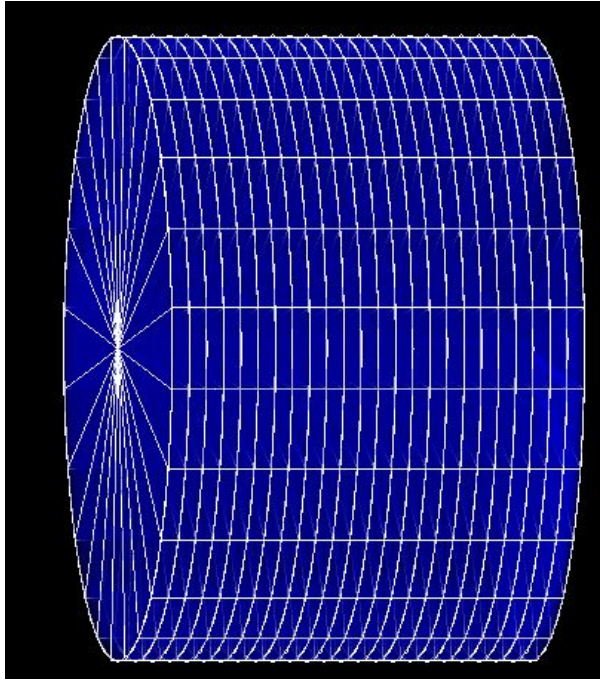


1 item selected.		Slice (1)
Title	Relu1_kernel_time	
Category	Node	
User Friendly Category	other	
Start	778.691 ms	
Wall Duration	0.007 ms	
▼ Args		
output_size	"400"	
parameter_size	"0"	
activation_size	"400"	
graph_index	"6"	
exec_plan_index	"6"	
provider	"CPUExecutionProvider"	
op_name	"Relu"	

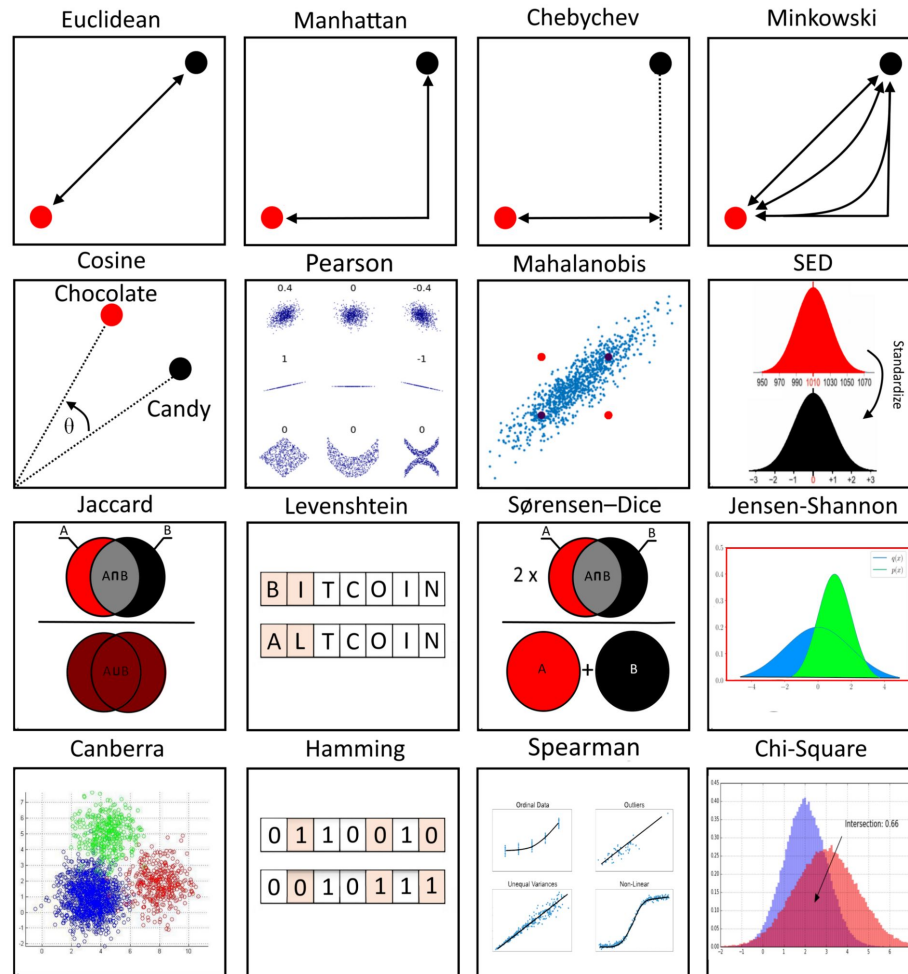
# From ML training to Geant4 fast simulation



## PBWO4 Geometry with 24x24x24 cell segmentation

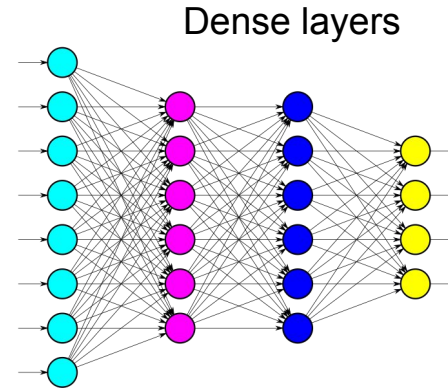


# Validation metrics



# ML optimization : from dense to convolutional layers

- With smaller input size 24x24x24 dense layers are easy to train , number of trainable parameters depends on the width and length of the NN
  - Test 1: (50x48x120) with dense layers
  - Test 2: (50,48,120) with Conv layers
- + Reduce the number of trainable parameters

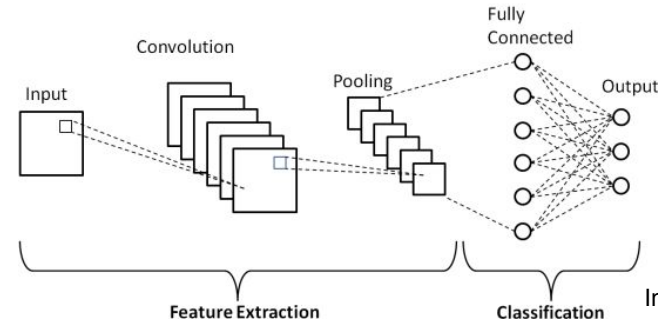


Dense layers

Memory footprint (CPU)

**600 Mb** (with integration optimization)

Convolutional layers



**500 Mb** (no integration optimization)

In CNN only last layer is fully connected

File Camera

Style Guides Clipping Extras

Name \_\_\_\_\_

GLViewer::TGLSAViewer

Camera center: \_\_\_\_\_

☐ Show

☐ External

X: 0.000

Y: -8.102

Z: 951.748

Pick center

Annotation

☐ Pick annotation

Reference marker

☐ Show

X: 0.000

Y: -8.102

Z: 951.748

Axes

☒ None

☐ Edge

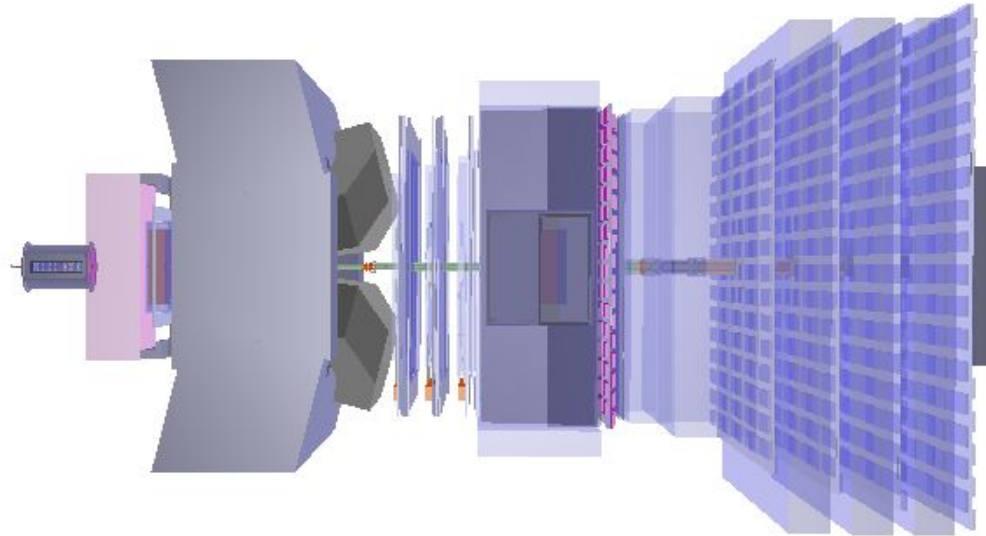
☐ Origin

☒ DepthTest

Camera overlay

☐ Show

## Multi-detector geometry modeling using an LHC experiment calorimeter

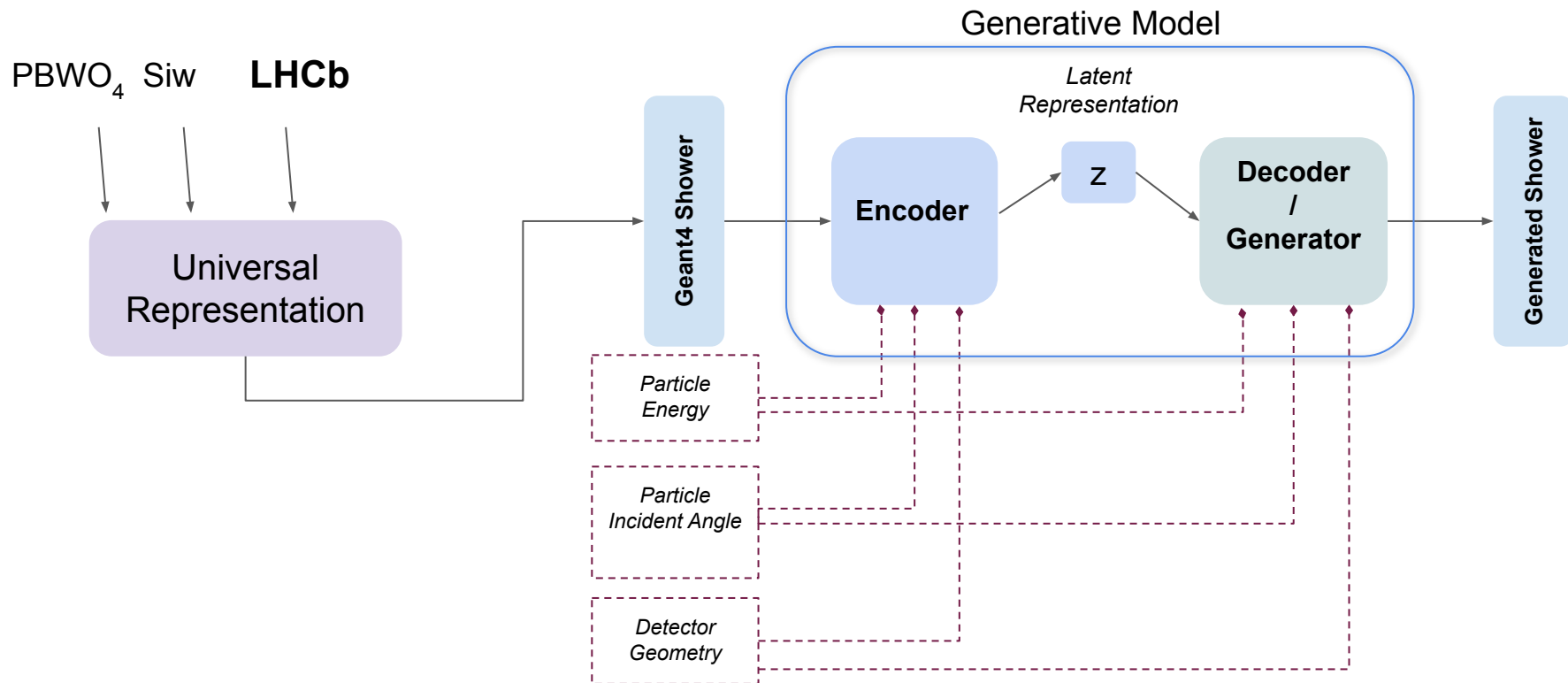


LHCb geometry loaded from a [GDML](#) file

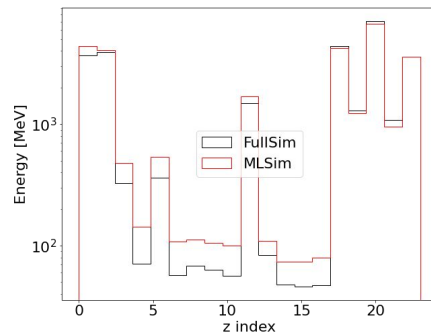
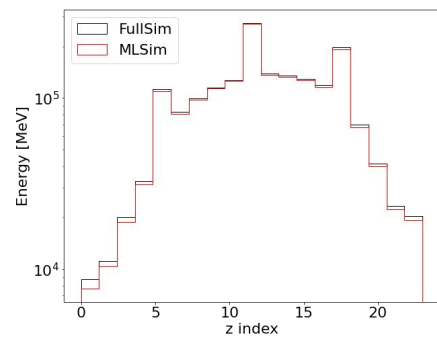
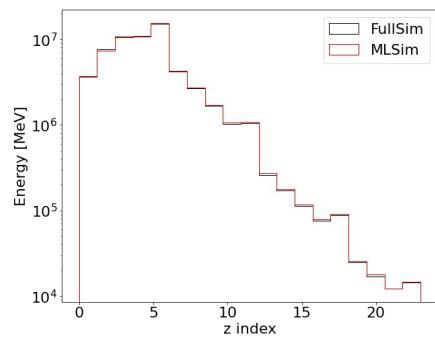
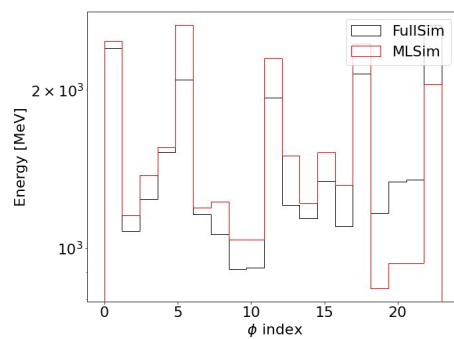
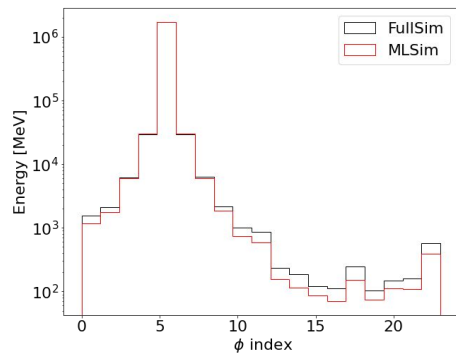
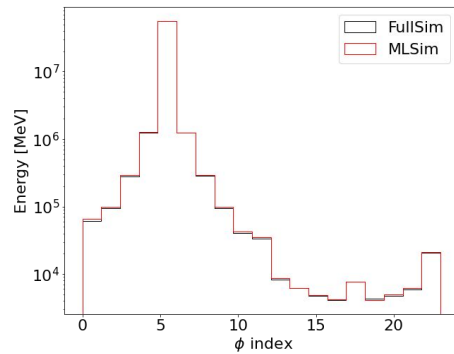
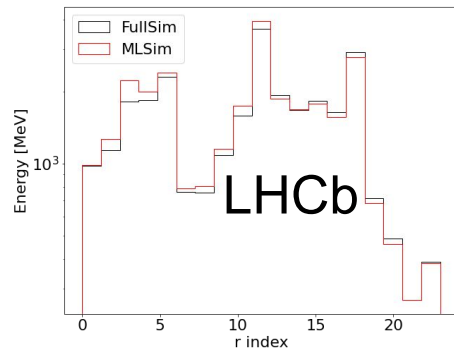
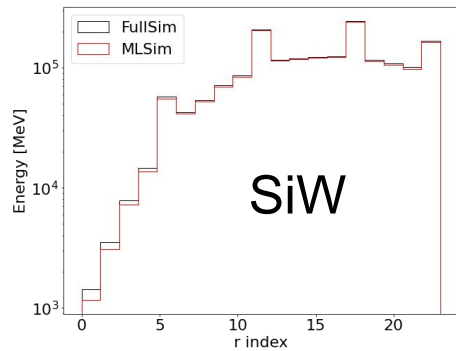
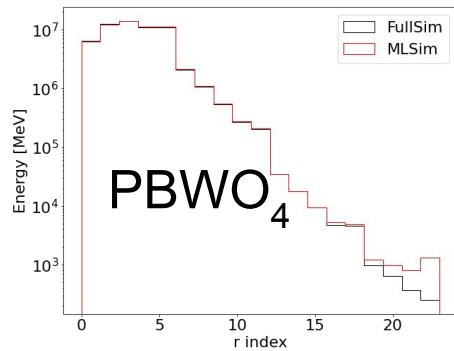
# Inference libraries : LWTNN vs ONNX

	<b>LightWeight Trained Neural Network (LWTNN)</b> <a href="#">Github</a>	<b>Open Neural Network Exchange (ONXX)</b> <a href="#">Github</a>
Description	C++ library to apply NN Minimal dependencies : Eigen, Boost	Open format to represent ML models ONNX Runtime: a cross-platform framework for ML model's inference and deployment
Supported ML libraries	Sklearn and Keras models (it is possible to convert a Tensorflow model to a keras model)	Saves models from (almost) all libraries
Supported layers	All except: CNN, Repeat Vector, Reshape.	All
Supported Activation functions	All except: Selu, PRelu	All
File format	JSON	ProtoBuf

# Multi-detector geometry modeling



Electron particles  
 Energy : 60 GeV  
 Incident angle :  $40^\circ$



# Meta Learning : learning-to-learn “Fast”

## Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks

Chelsea Finn, Pieter Abbeel, Sergey Levine

[arXiv:1703.03460](https://arxiv.org/abs/1703.03460)

We propose an algorithm for meta-learning that is model-agnostic, in the sense that it is compatible with any model trained with gradient descent and applicable to a variety of different learning problems, including classification, regression, and reinforcement learning. The goal of meta-learning is to train a model on a variety of learning tasks, such that it can solve new learning tasks using only a small number of training samples. In our approach, the parameters of the model are explicitly trained such that a small number of gradient steps with a small amount of training data from a new task will produce good generalization performance on that task. In effect, our method trains the model to be easy to fine-tune. We demonstrate that this approach leads to state-of-the-art performance on two few-shot image classification benchmarks, produces good results on few-shot regression, and accelerates fine-tuning for policy gradient reinforcement learning with neural network policies.

## On First-Order Meta-Learning Algorithms

Alex Nichol and Joshua Achiam and John Schulman

OpenAI

{alex, jachiam, joschu}@openai.com

[arXiv:1803.02999](https://arxiv.org/abs/1803.02999)

### Abstract

This paper considers meta-learning problems, where there is a distribution of tasks, and we would like to obtain an agent that performs well (i.e., learns quickly) when presented with a previously unseen task sampled from this distribution. We analyze a family of algorithms for **learning a parameter initialization** that can be fine-tuned quickly on a new task, using only first-order derivatives for the meta-learning updates. This family includes and generalizes first-order MAML, an approximation to MAML obtained by ignoring second-order derivatives. It also includes Reptile, a new algorithm that we introduce here, which **works by repeatedly sampling a task, training on it, and moving the initialization towards the trained weights on that task**. We expand on the results from Finn et al. showing that first-order meta-learning algorithms perform well on some well-established benchmarks for few-shot classification, and we provide theoretical analysis aimed at understanding why these algorithms work.

## Algorithm 1 Reptile (serial version)

Initialize  $\phi$ , the vector of initial parameters

**for** iteration = 1, 2, ... **do**

Sample task  $\tau$ , corresponding to loss  $L_\tau$  on weight vectors  $\tilde{\phi}$

Compute  $\tilde{\phi} = U_\tau^k(\phi)$ , denoting  $k$  steps of SGD or Adam

Update  $\phi \leftarrow \phi + \epsilon(\tilde{\phi} - \phi)$

**end for**

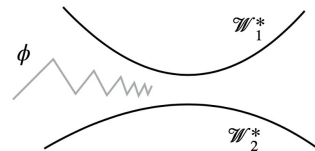


Figure 2: The above illustration shows the sequence of iterates obtained by moving alternately towards two optimal solution manifolds  $\mathcal{W}_1$  and  $\mathcal{W}_2$  and converging to the point that minimizes the average squared distance. One might object to this picture on the grounds that we converge to the same point regardless of whether we perform one step or multiple steps of gradient descent. That statement is true, however, note that minimizing the expected distance objective  $\mathbb{E}_\tau [D(\phi, \mathcal{W}_\tau)]$  is different than minimizing the expected loss objective  $\mathbb{E}_\tau [L_\tau(f_\phi)]$ . In particular, there is a high-dimensional manifold of minimizers of the expected loss  $L_\tau$  (e.g., in the sine wave case, many neural network parameters give the zero function  $f(\phi) = 0$ ), but the minimizer of the expected distance objective is typically a single point.