# Recent developments of machine learning in experimental particle physics

AIRA Seminar, 9.12.2021

Hardware
Acceleration
Lab

Bartosz Soból

# Presentation plan

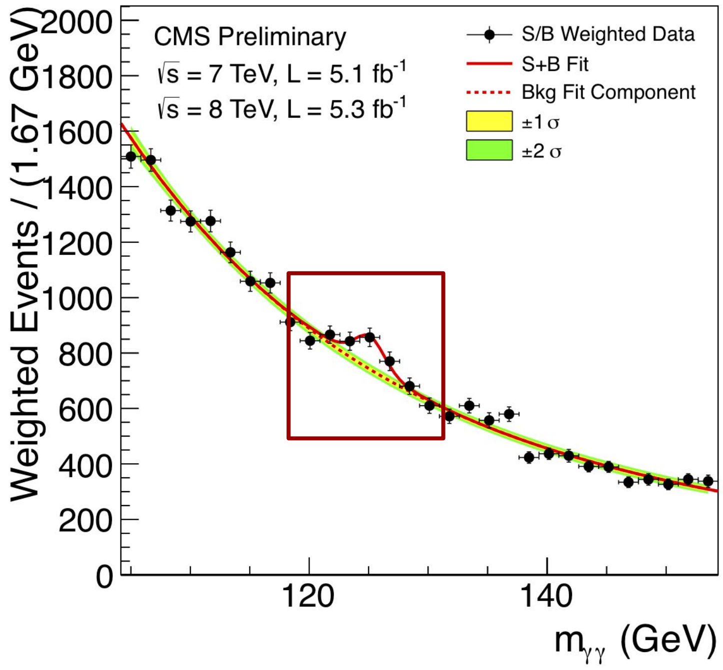
- Particle physics experiment *workflow*
- ML in particle track reconstruction
- ML in detector event simulation
- Hardware accelerated neural networks

# Particle physics experiment *workflow*

- We have a physics problem that needs to be studied
  - e.g. specific decay predicted by a new theory
- And the experiment (detector) that would be able to find effects of such prediction
- Using Monte Carlo methods it's possible to simulate the experiment outcome assuming known and confirmed physics
- Than statistical analysis of simulated and experimental data can be conducted
- When they differ - it may be a hint for more experiments
  Or, if the difference is significant, a new physics discovery

# Example: Higgs boson discovery in LHC (2012)

- Theory (Standard Model) predicts the existence of a heavy particle that decays into two photons
- From other experiments it was known that we should look for it in the mass region between 116 and 127 GeV
- Experiment: collide two protons with very high energy (7-8 TeV) and hope it will produce a new particle
- From simulations we know what the outcome should be if there's no new particle produced
- With this information, we can extract the new signal
- Which differs with more than 5σ from the expected (background)
- In particle physics 5σ confidence means a discovery



# Particle track reconstruction

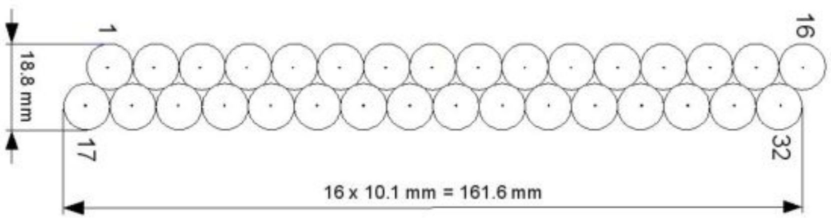
- Particle detectors generate a vast amount of multidimensional (up to over 100 million channels) readout data
  - Every channel (dimension) corresponds to detector section
- Collaborations at LHC predict they will generate 1 - 3 TB/s in the 2 years (ALICE)
  - In smaller experiments it's about ~200 - 300 GB/s
- In each detector event (timeframe, microseconds) particles pass through
  - Path, momentum, charge, etc. of each particle has to be known for physics analysis - - **track reconstruction**
- Data is often sparse
  - Each particle interacts only with a small part of the detector
- Mainly classic (and difficult to parallelize) algorithms were used for this task

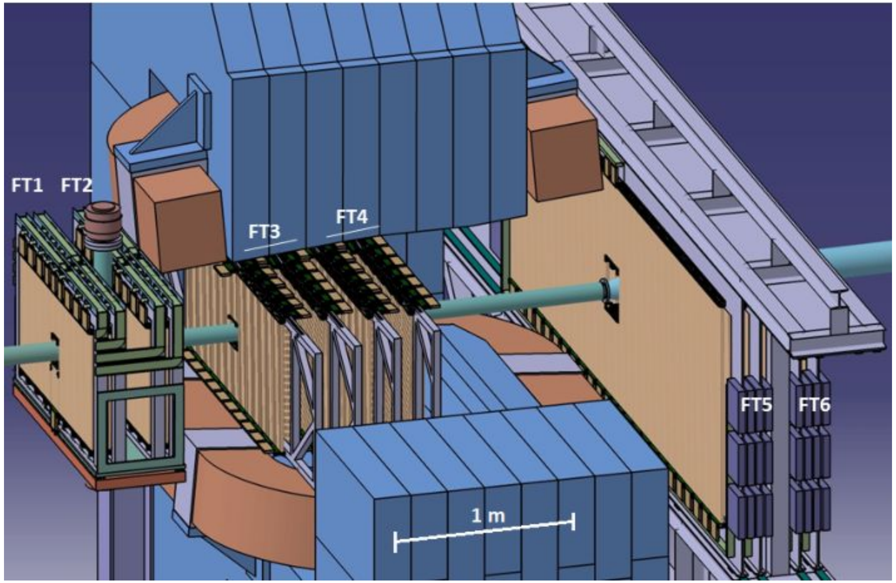
# Machine learning for track reconstruction

- In 2018 and 2019 *TrackML Challenge* on Kaggle was organised by CERN
- Results were mixed, but graph neural networks (GNNs) turned out to be the most promising approach
- Since then collaborations at CERN and other facilities evaluate and improve GNN-based solutions for their tasks
- Main difficulties include
    - Efficient transformation of readout data into graphs
    - Complexity of detectors (size and existence of multiple subsystems of different characteristics)

# Example: PANDA Forward Tracker

- PANDA is an experiment under construction at FAIR Facility (Darmstadt, Germany)
- FT is a relatively small (sub)detector
  - ~12k *straws* with which particles interact
  - Grouped into 6 *stations* and 48 layers
  - Each straw is an additional input channel
  - Easy to model in a graph structure

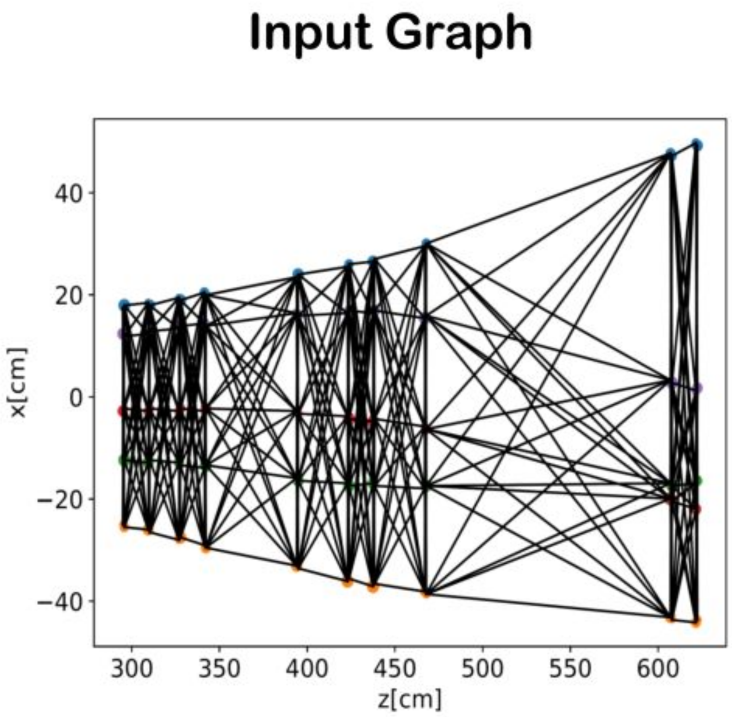


# GNNs in PANDA FT

**Input**

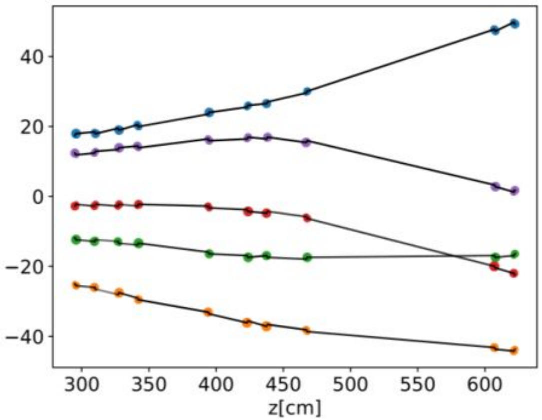
- Interactions of particles with straw in one detector event
- Transformed into graph structure:
  - Connect every hitted straw with all hitted straws in adjacent layers

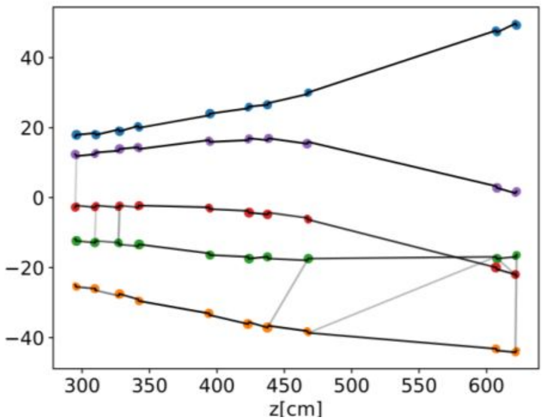


# GNNs in PANDA FT

**Output:**

- Ideally output should be the set of separate graphs representing particle tracks
- In real world it contains additional edges that may lower accuracy



# GNNs in PANDA FT: Results and remarks

- GNN-based approach was tested with simulated data
  - Synthetic case, homogeneous dataset
- Meets accuracy requirements
- Performance needs some improvement
  - Especially the step of graph generation from raw data
- Number of edges in input graph can be lowered by eliminating physically impossible connections
- Similar approaches studied by other experiments (CERN), often more advanced

# Detector event simulation

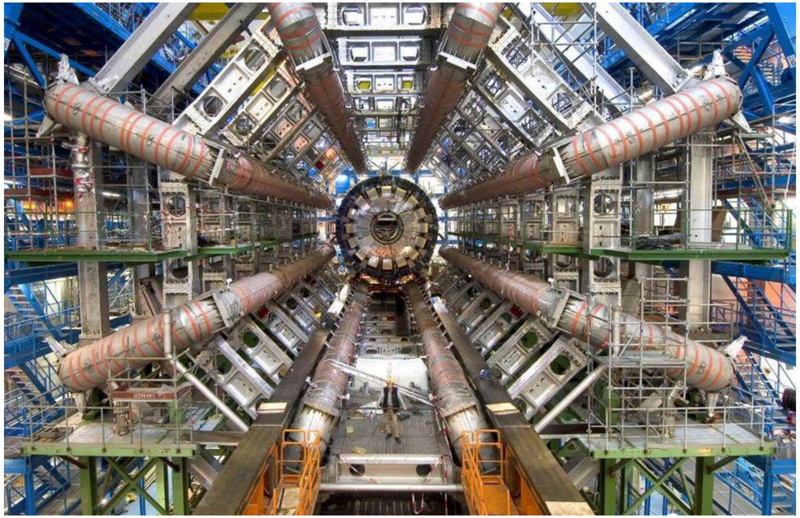
- Critical for conducting experiments
- But also very important
  - During design phase of new devices
  - For evaluation/maintenance of detectors and algorithms
- Traditionally conducted using Monte Carlo methods
  - Software: Geant3, Geant4, PYTHIA
- Consume a lot of computational resources
- More data collected in larger new experiments result in need for more simulations for adequate statistics

# Machine learning for detector simulation

- Variations of generative adversarial networks (GANs) proposed
- Challenges
  - Detector response vary a lot depending on type of particle and its physical properties
  - Generated (simulated) data has to be accurate to a certain level

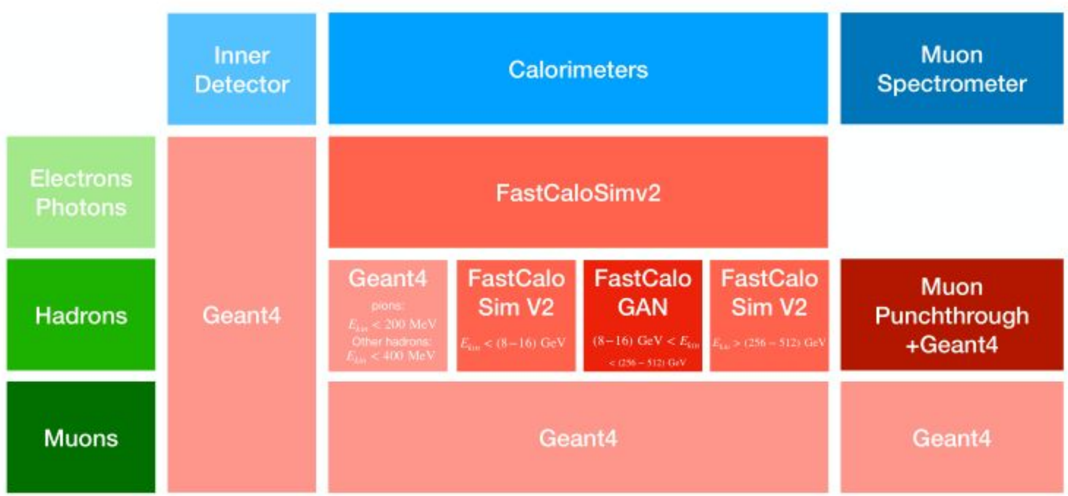
# Example: ATLAS calorimeters simulation

- ATLAS is located at LHC and is the largest particle physics experiment worldwide
- It’s expected to be the one to discover new physics
- As a result it needs cary large sets of simulation data for statistics
  - 40% of ATLAS’ CPU computation resources is consumed for simulations
  - Computing  infrastructure won’t fulfill the needs with current simulation software
- Classic Monte Carlo simulation methods are CPU-bound and vary hard or impossible to parallelize for GPUs
- Machine learning and neural networks are explored as one of alternatives



# ATLAS experiment and AtlFast3 framework

- The AtlFast3 framework was proposed for ATLAS
  - Combines current Monte Carlo tool (Geant4) with simplified simulation (FastrCaloSim) and GAN-based simulation (FastCaloGan)
  - Depending on subsystem of the detector and simulated particle



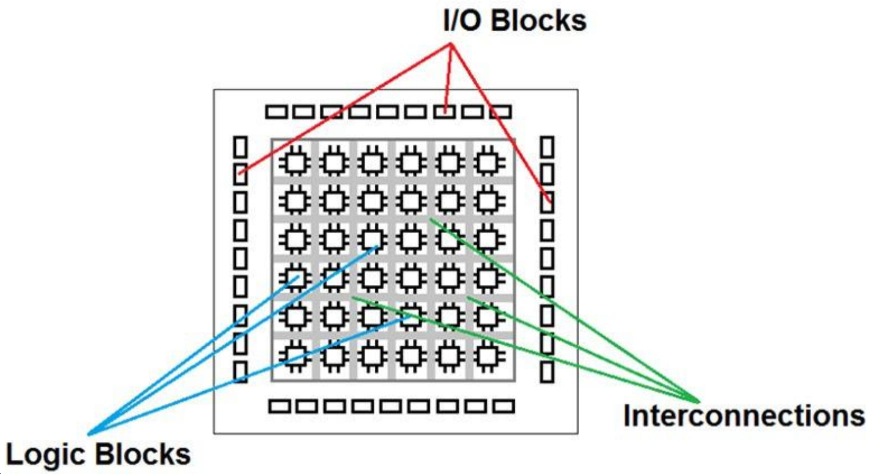
# ATLAS experiment and AtlFast3 framework

- 5x overall performance improvement (CPU-time)
  - 500x in calorimeter subsystem!
  - with ‘2%’ accuracy drop
- Usage of GANs is limited to one type of particles in one subsystem
  - Other areas will probably require different, or at least, differently trained, models
- Research on broader usage of GANs as well as improved performance and accuracy continues

# Neural networks accelerated on FPGA

**What is FPGA?**

- FPGA - Field Programmable Gate Array
- A set (array) of logic building blocks that can behave as any kind of logic gate each
- Accelerated algorithm is mapped directly to the hardware (like in custom chip)
- Can be programmed with high-level languages (C++-based)
- Are now available as accelerator cards similar to GPUs used for NN training (Xilinx Alveo)




# Neural networks accelerated on FPGA

- Processing of live data in experiments (and other applications) is often constrained in terms of computational resources and latency
- FPGA-based accelerators have unique capabilities
  - Upper bound on processing time can be strictly defined in clock cycles
  - % of chip resources used by each accelerated procedure is well-known
  - Many low level optimisation are possible and supported by hardware and programming tools
  - e.g. loop unrolling or usage of arbitrary precision fixed-point arithmetic types
- There are tools available (hls4ml) that enable compilation of Keras, PyTorch and TensorFlow code for FPGAs

# Neural networks accelerated on FPGA

- Research at CERN, Caltech and Google
- Extension of Keras and hls4ml
  - Enables fixed-point arithmetic for network parameters
  - For each network layer separately
- Results in significant reduction in on-chip resource usage for inference (4-layer dense NN)
  - With small impact on accuracy
- May be beneficial for many high-throughput, low-latency applications
- As well as resource constrained ones (IoT, robotics?)

| Model | Accuracy [%] | Latency [ns] | Latency [clock cycles] | DSP [%] | LUT [%] | FF [%] |
|---|---|---|---|---|---|---|
| **BF** | 74.4 | 45 | 9 | 56.0 (1,826) | 5.2 (48,321) | 0.8 (20,132) |
| **BP** | 74.8 | 70 | 14 | 7.7 (526) | 1.5 (17,577) | 0.4 (10,548) |
| **BH** | 73.2 | 70 | 14 | 1.3 (88) | 1.3 (15,802) | 0.3 (8,108) |
| **Q6** | 74.8 | 55 | 11 | 1.8 (124) | 3.4 (39,782) | 0.3 (8,128) |
| **QE** | 72.3 | 55 | 11 | **1.0 (66)** | **0.8 (9,149)** | 0.1 (1,781) |
| **QB** | 71.9 | 70 | 14 | 1.0 (69) | 0.9 (11,193) | 0.1 **(1,771)** |
| **LogicNets JSC-M** [47] | 70.6 | N/A[a] | N/A | 0 (0) | 1.2 (14,428) | 0.02 (440) |
| **LogicNets JSC-L** [47] | 71.8 | 13[b] | 5 | 0 (0) | 3.2 (37,931) | 0.03 (810) |

[a] Not evaluated.
[b] Using a clock frequency of 384 MHz.

# Bibliography


1. *Towards a realistic track reconstruction algorithm based on graph neural networks for the HL-LHC*
   C. Biscarat, S. Caillou, C. Rougier, J. Stark, J. Zahreddine
   *https://arxiv.org/abs/2103.00916*
2. *Implementing Graph Neural Network for Track Finding*
   W. Esmail, T. Stockmanns, J. Ritman
   *https://indico.gsi.de/event/12231/contributions/52060/attachments/35053/46054/PandaMeeting.pdf*
3. *AtlFast3: the next generation of fast simulation in ATLAS,*
   *The ATLAS Collaboration*
   https://arxiv.org/abs/2109.02551
4. *Automatic heterogeneous quantization of deep neural networks for low-latency inference on the edge for particle detectors*
   C. N. Coelho Jr., A. Kuusela, S. Li, H. Zhuang, T. Aarrestad, V. Loncar, M. Pierini, A. A. Pol, S. Summers
   https://arxiv.org/abs/2006.10159