UNIVERSITY OF LIVERPOOL

Machine Learning for tracking in HEP experiments

HEP annual meeting 21st May 2026

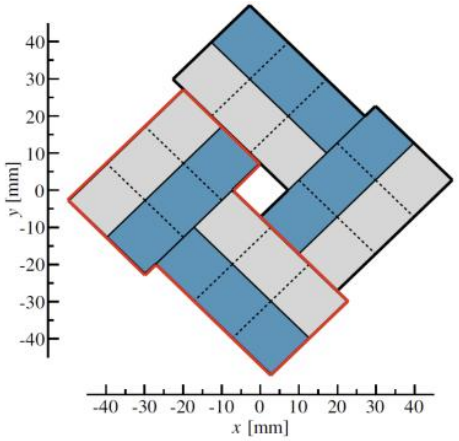
Giuseppe Costantino

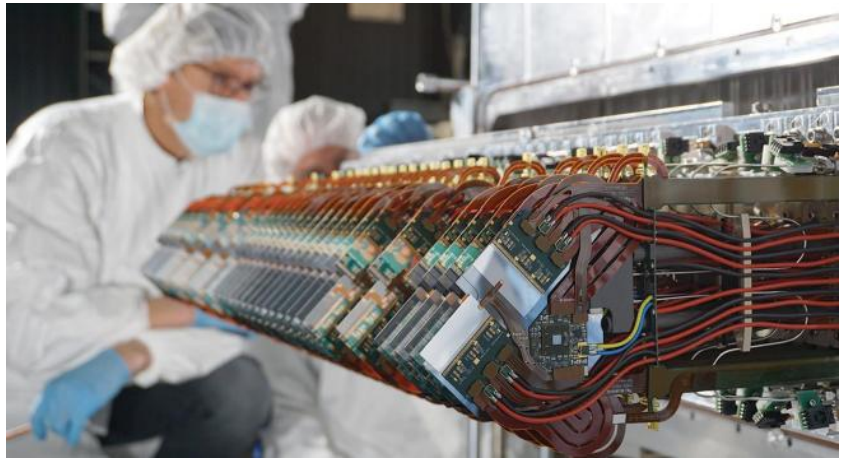
Supervisors: Karol Hennessy, David Hutchcroft, Tara Shears

w/ Kurt Rinnert

PhD Project

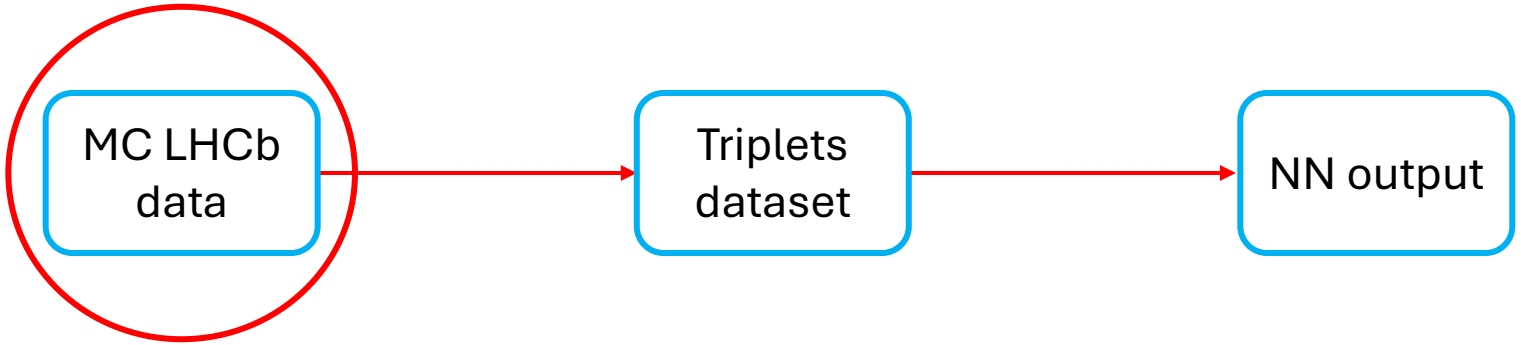
- Explore Machine Learning techniques for particle tracking in HEP experiments and evaluate performances on accelerators
- Model learns from physics – general approach and exploration of adaptability on other experiments
- LHCb velo used as case study





Workflow

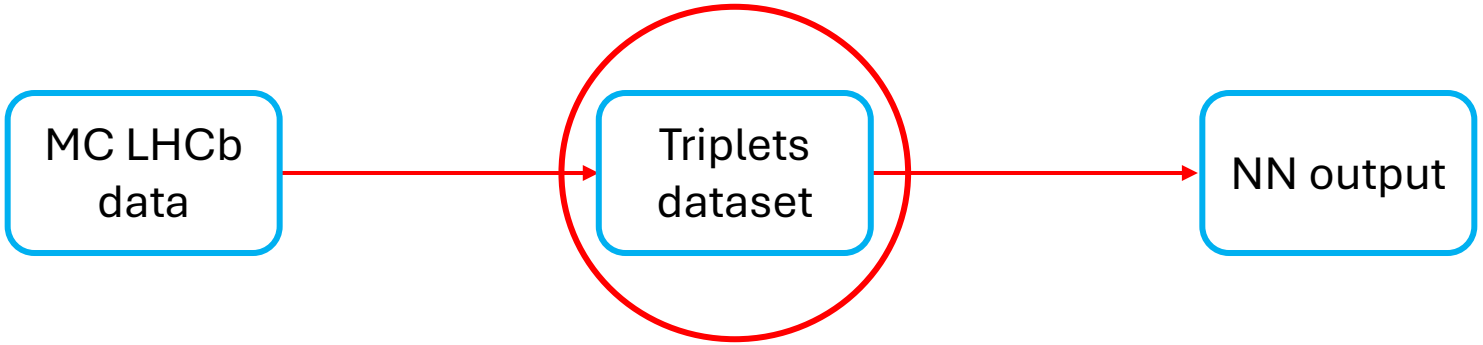
Workflow

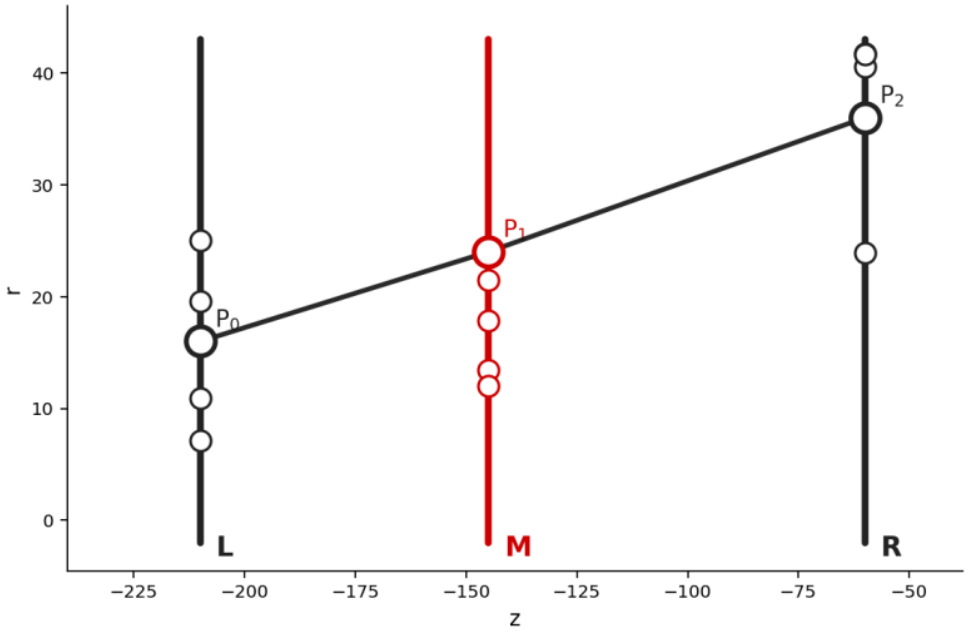


- 11k events
- Analysed with ROOT
- Collect useful information for labelling and debugging using MC truth

ROOT
Data Analysis Framework

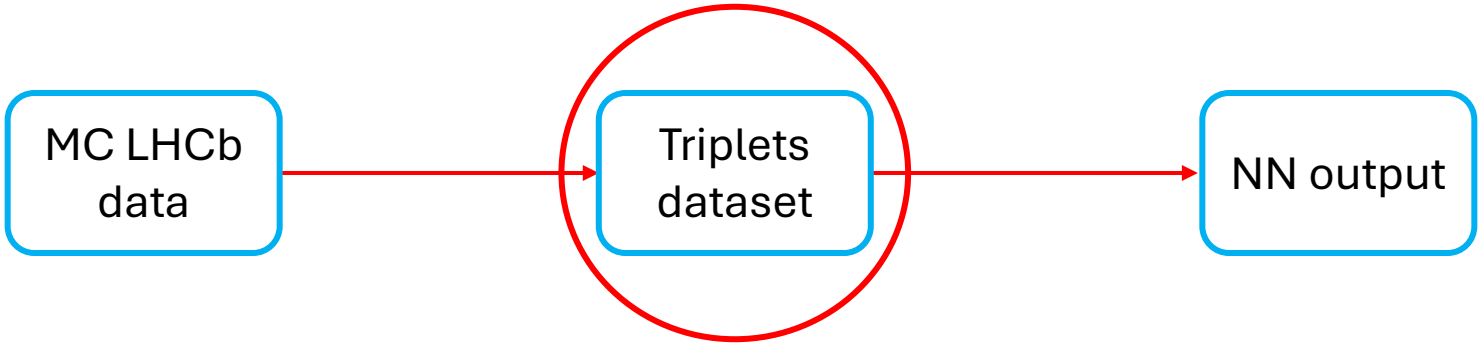
Workflow

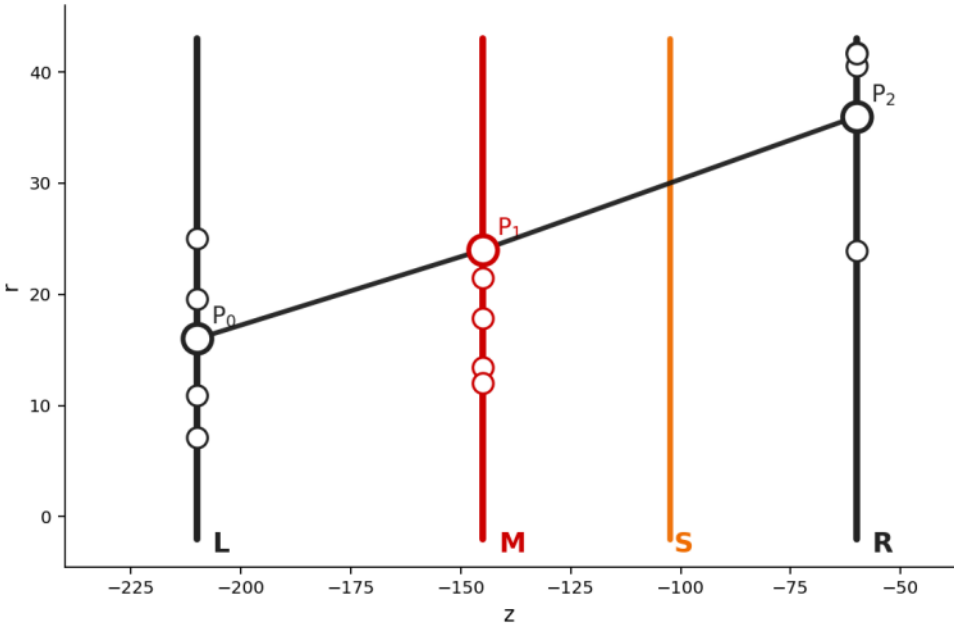




- Defined as 3 consecutive hits consecutive planes
- Cut on ϕ window

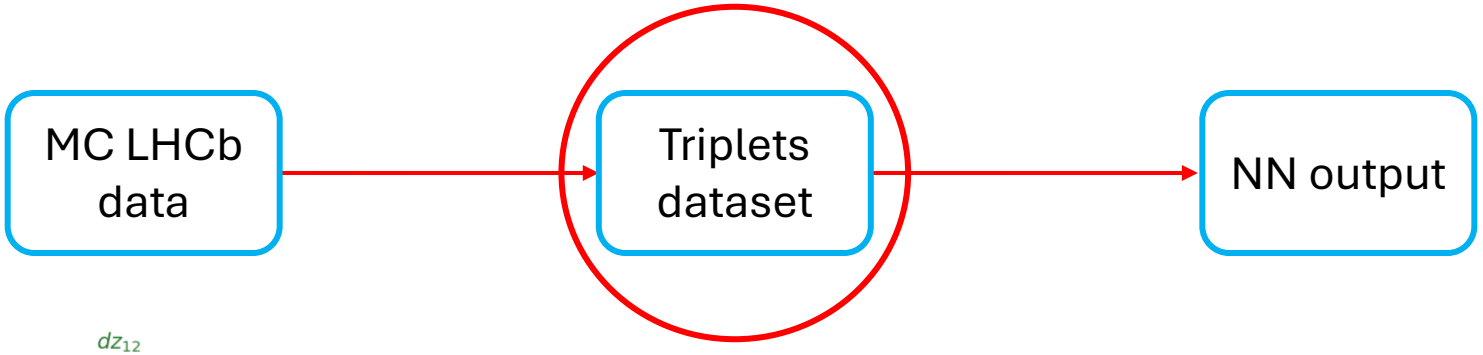
Workflow

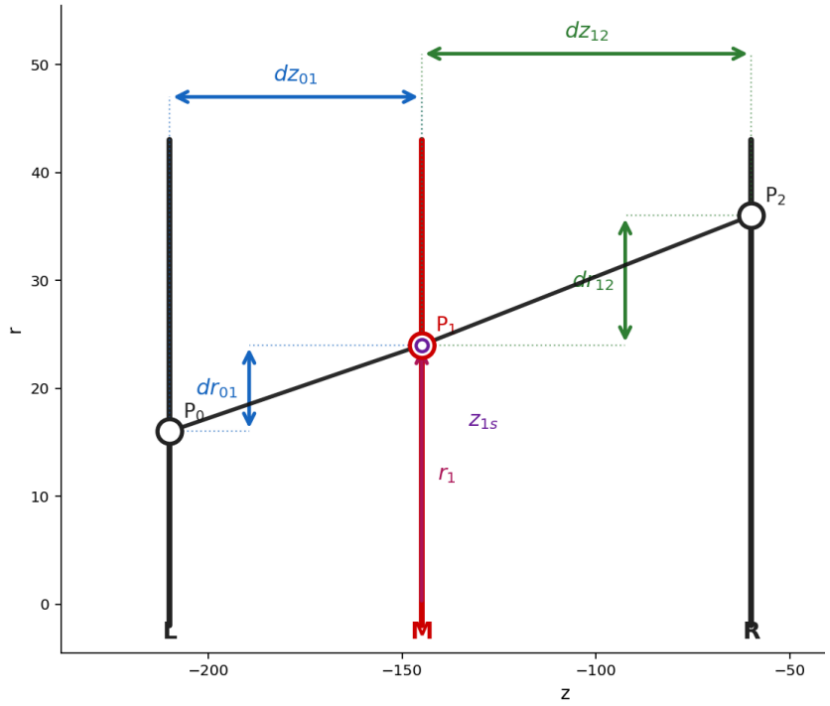




- Defined as 3 consecutive hits consecutive planes
- Cut on ϕ window
- Cut on one skipped module

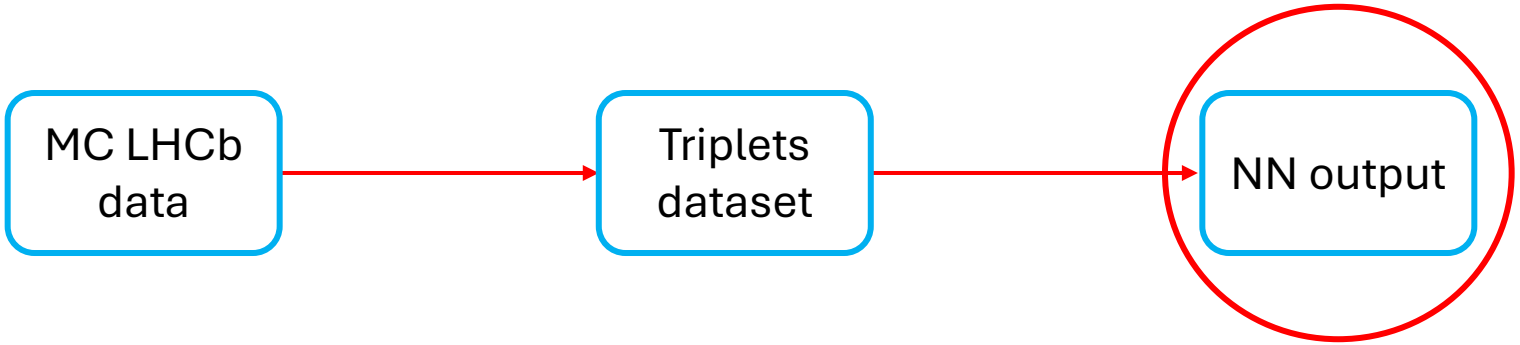
Workflow



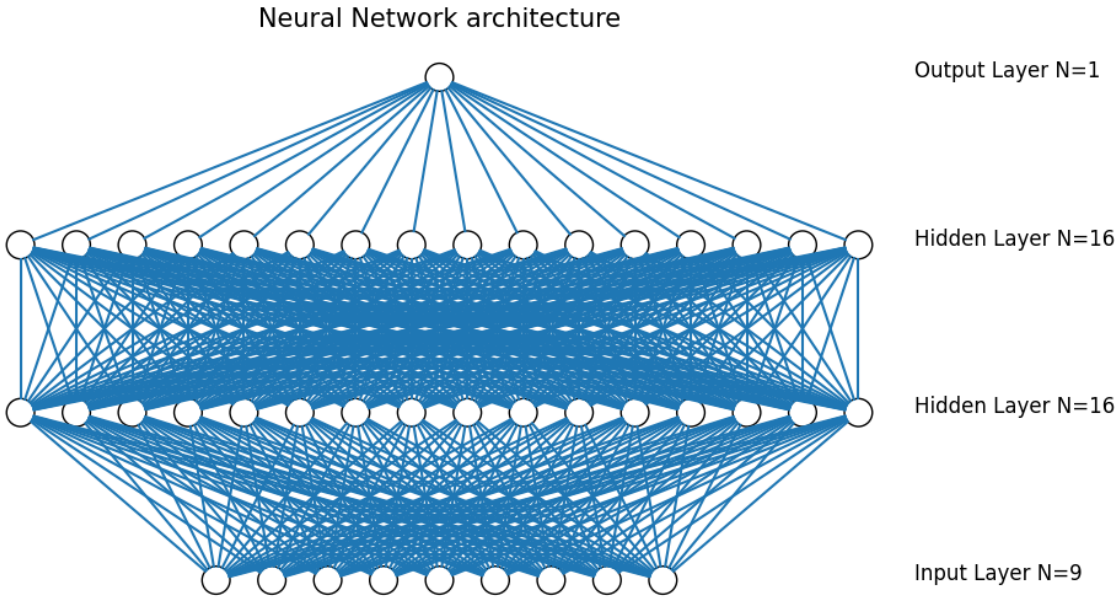


- 9 features
- Labelled balanced dataset
  - 36M samples training set
  - 7.7M validation set
  - 7.7 test set
  - 40 events per run / 130k triplets per event

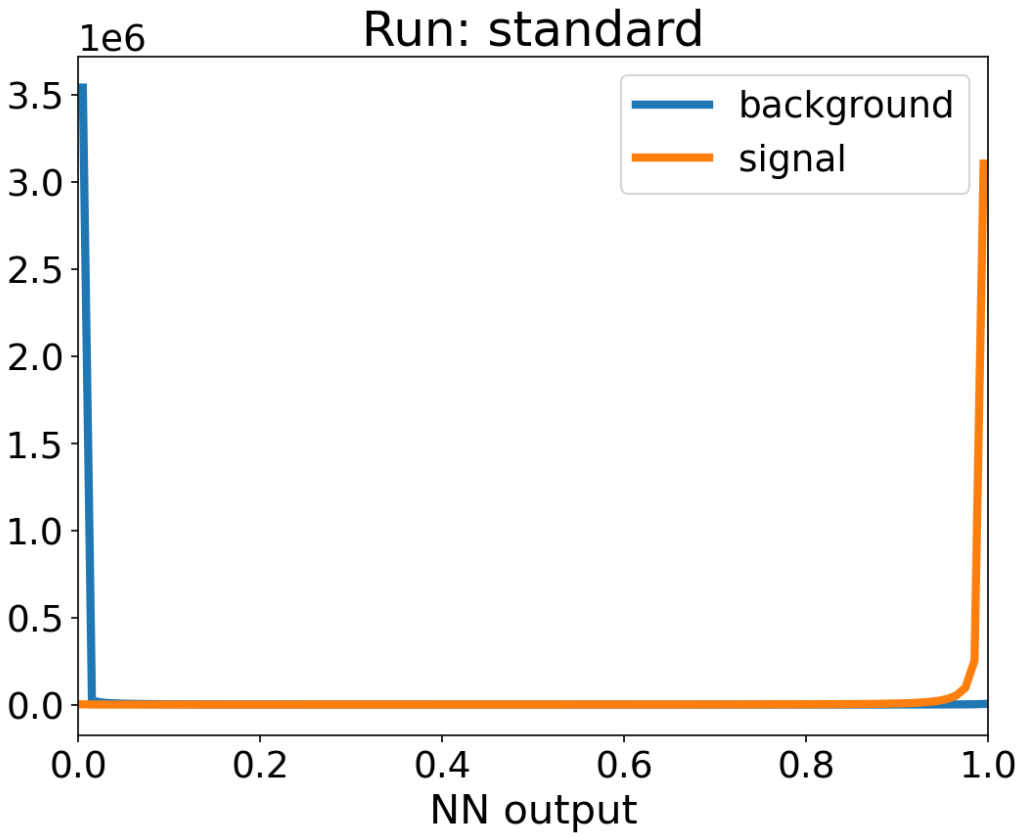
Workflow

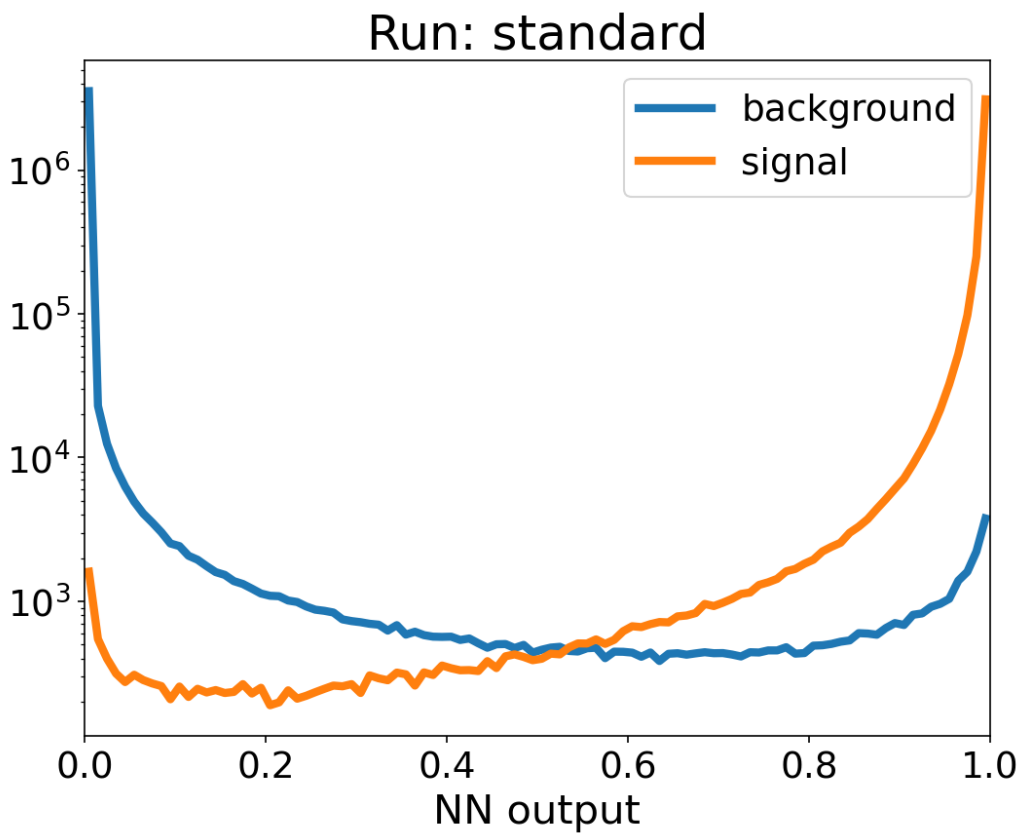


- PyTorch: framework for building and training learning models
- MLP fully connected: multi layer perceptron in which every neuron in a layer is connected to every single neuron in the next layer
- 9,16,16,1

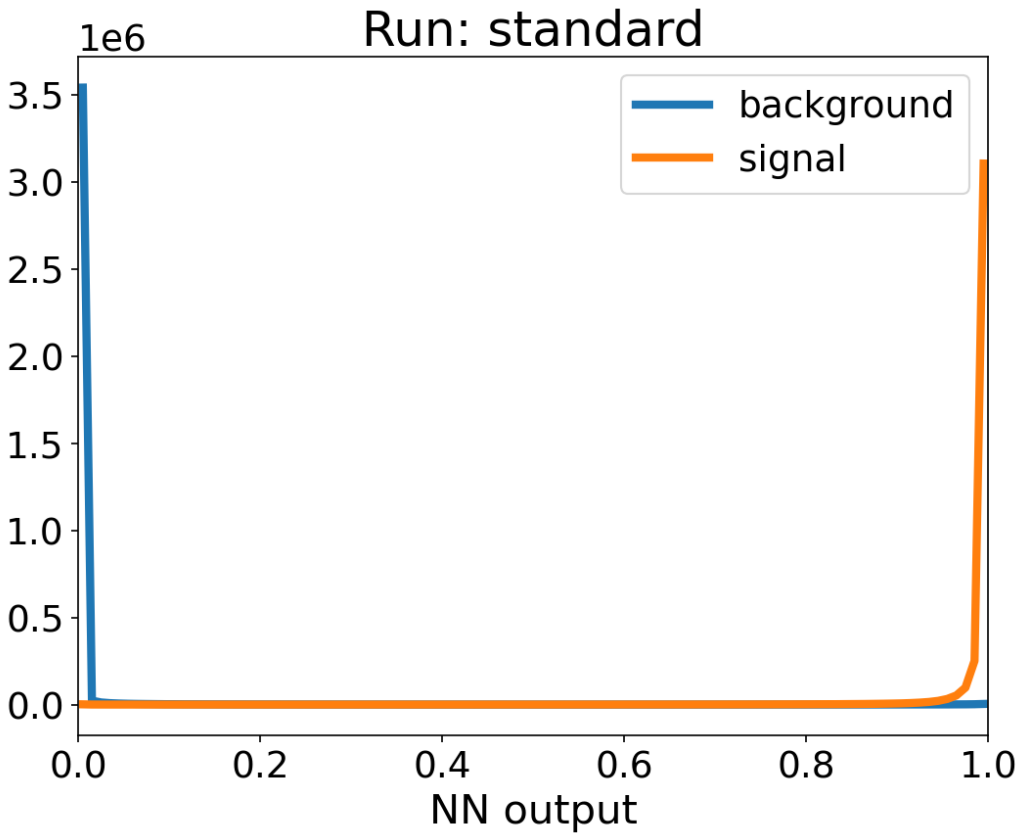


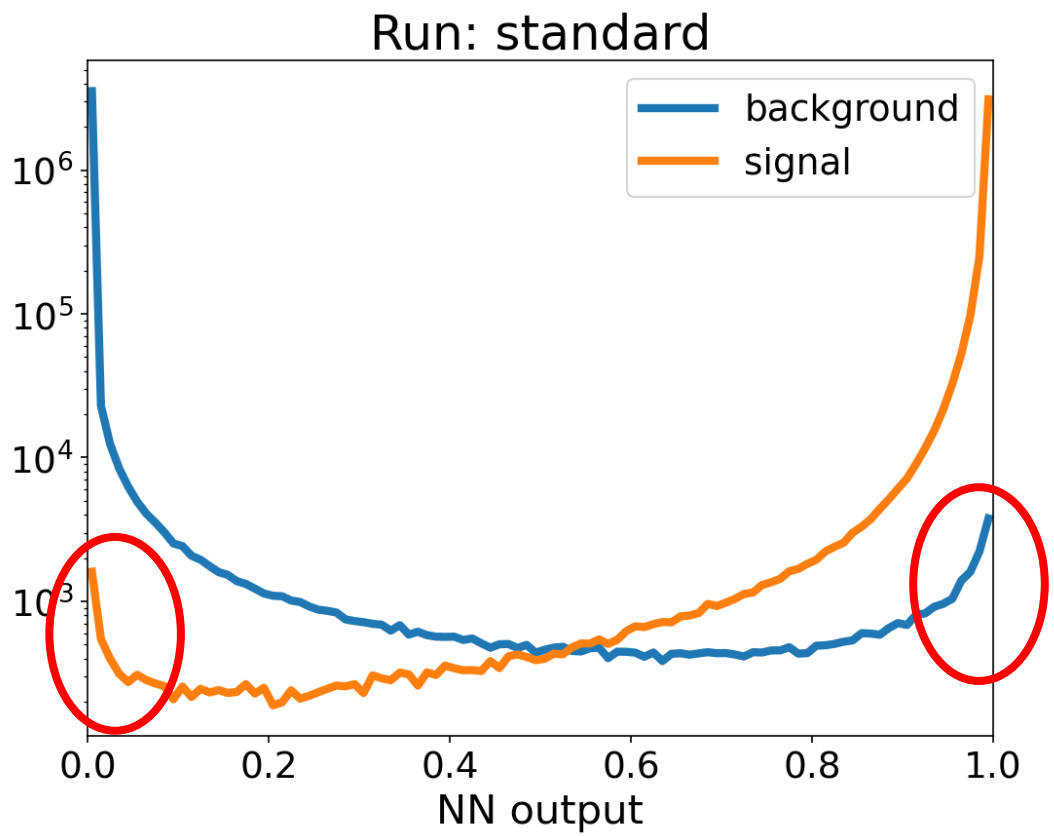
Signal - background separation





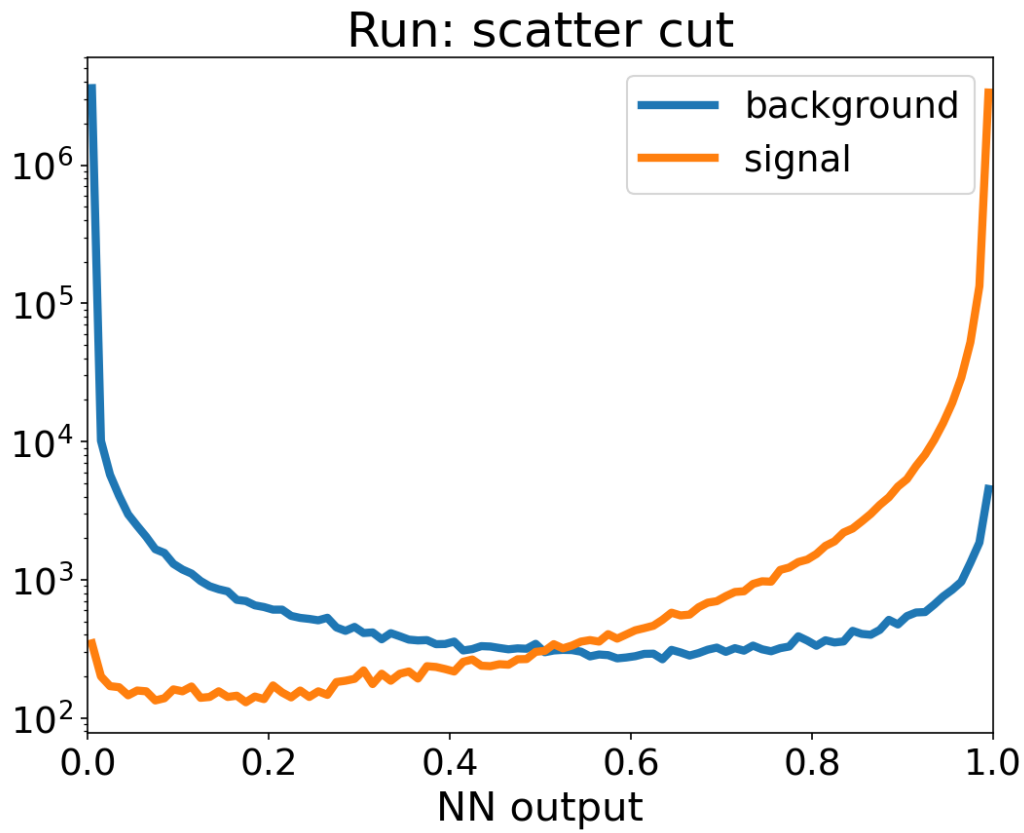
Signal - background separation





Signal - background separation

- Most of the misclassified triplets came from low momentum tracks, electrons and multiple scattering
- Applied a max scatter cut:



Integration in the LHCb workflow

- Standard algorithm performs real-time reconstruction of particle tracks in the VELO detector
- Implemented in HLT1 it has two steps:
  - Seeding: triplet construction, first step of tracking
  - Forwarding: extending found triplets to the whole detector
- It runs on GPU and it's designed to exploit CUDA-based parallelism

Integration in the LHCb workflow

- Using the weights of my trained model, I replaced the seeding procedure with my own implementation:
  - Fake rate 5.31% vs 8.57%
  - Efficiency 16% lower

Future works

- Understand how to get closer to standard implementation
  - Measure times / optimization
- Compute acceleration implementation
  - CUDA
  - Intel oneAPI
  - FPGA
  - Cost benefit analysis
- Training on other simulated geometries

Thank you ☺

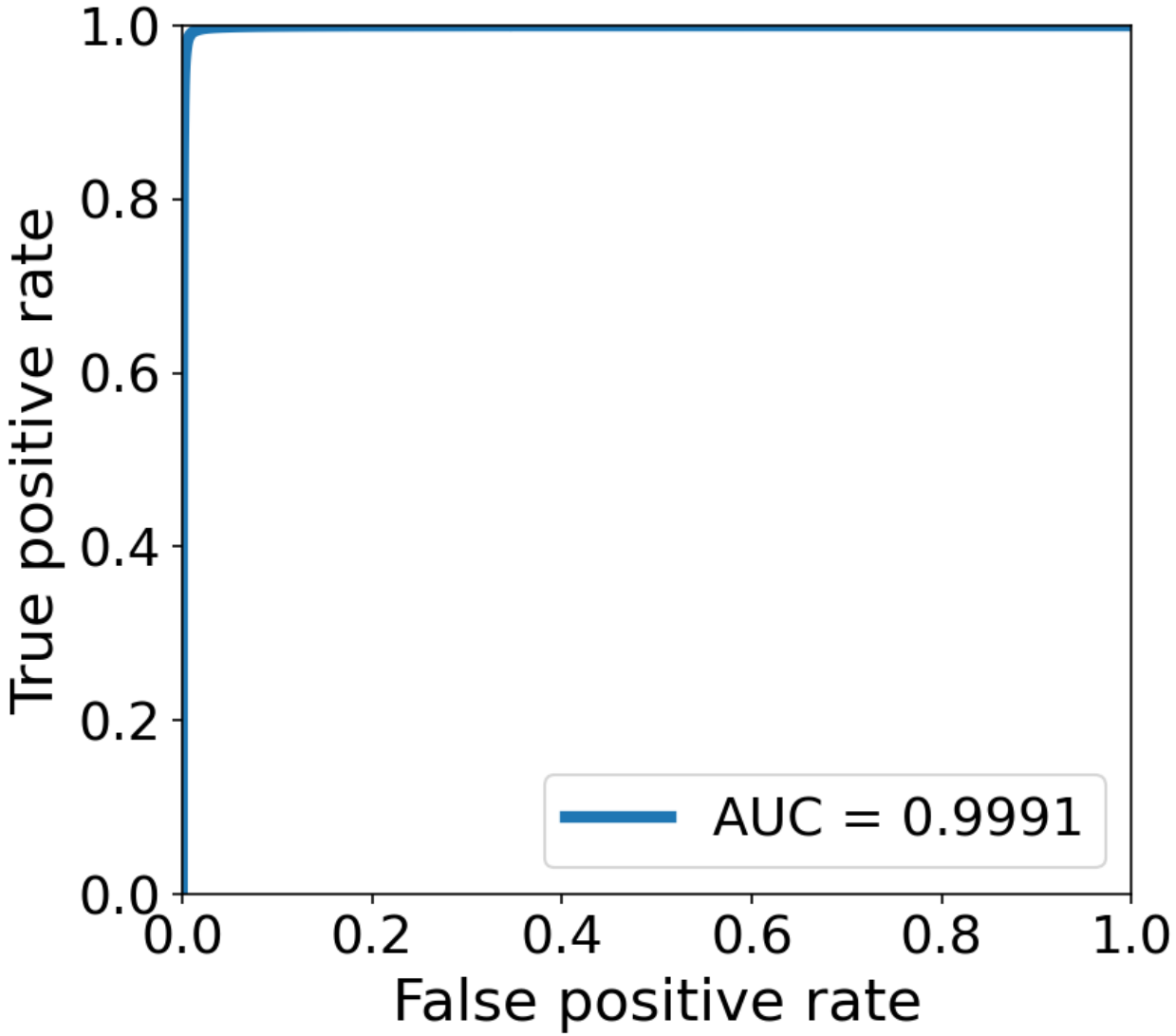
Backup

Training features

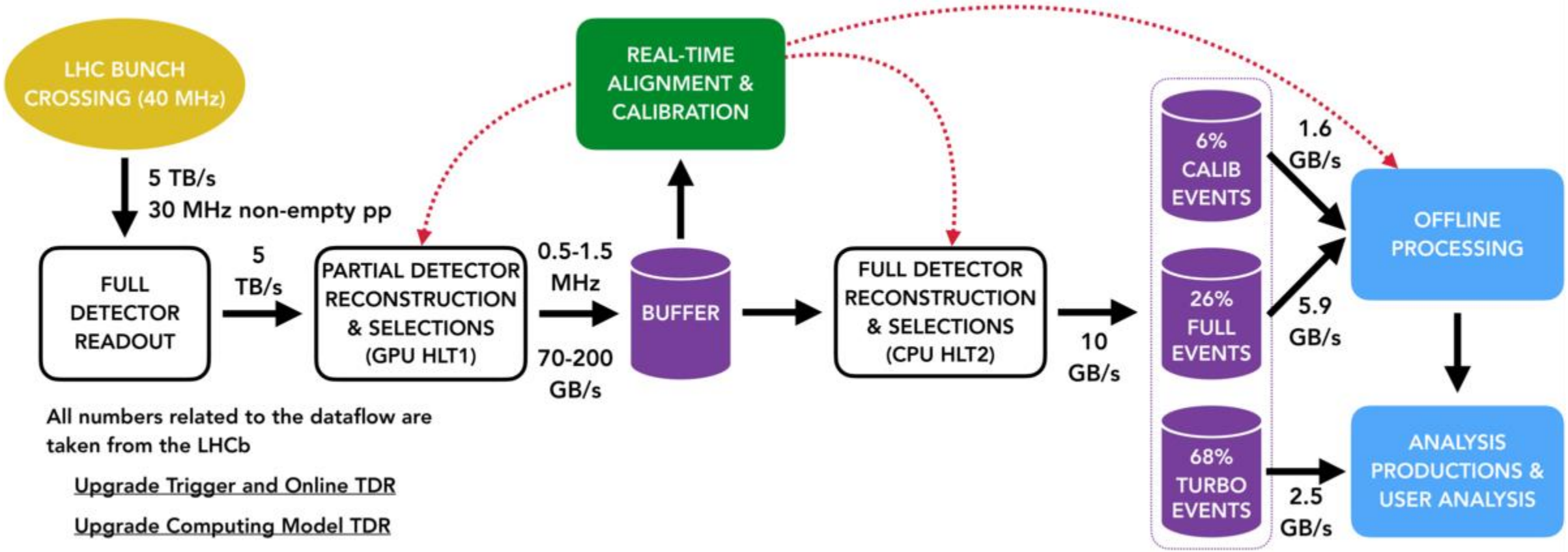
- dr01(12): radial distance between hit 0 and hit 1
- dphi01(12/02): azimuthal angular difference wrapped in (-pi,pi] and abs
- dz01(12): difference on z axis
- r1: $\sqrt{\{x\_1\ \hat{}2 + y\_1\ \hat{}2}$
- z1s: z coordinate of the middle hit

- Max scatter: $(x\_p - x\_2)\hat{}2 + (y\_p - y\_2)\hat{}2$

ROC


1.0
0.8
0.6
0.4
0.2
0.0
True positive rate
0.0
0.2
0.4
0.6
0.8
1.0
False positive rate
AUC = 0.9991

LHCb Workflow



All numbers related to the dataflow are taken from the LHCb

Upgrade Trigger and Online TDR

Upgrade Computing Model TDR