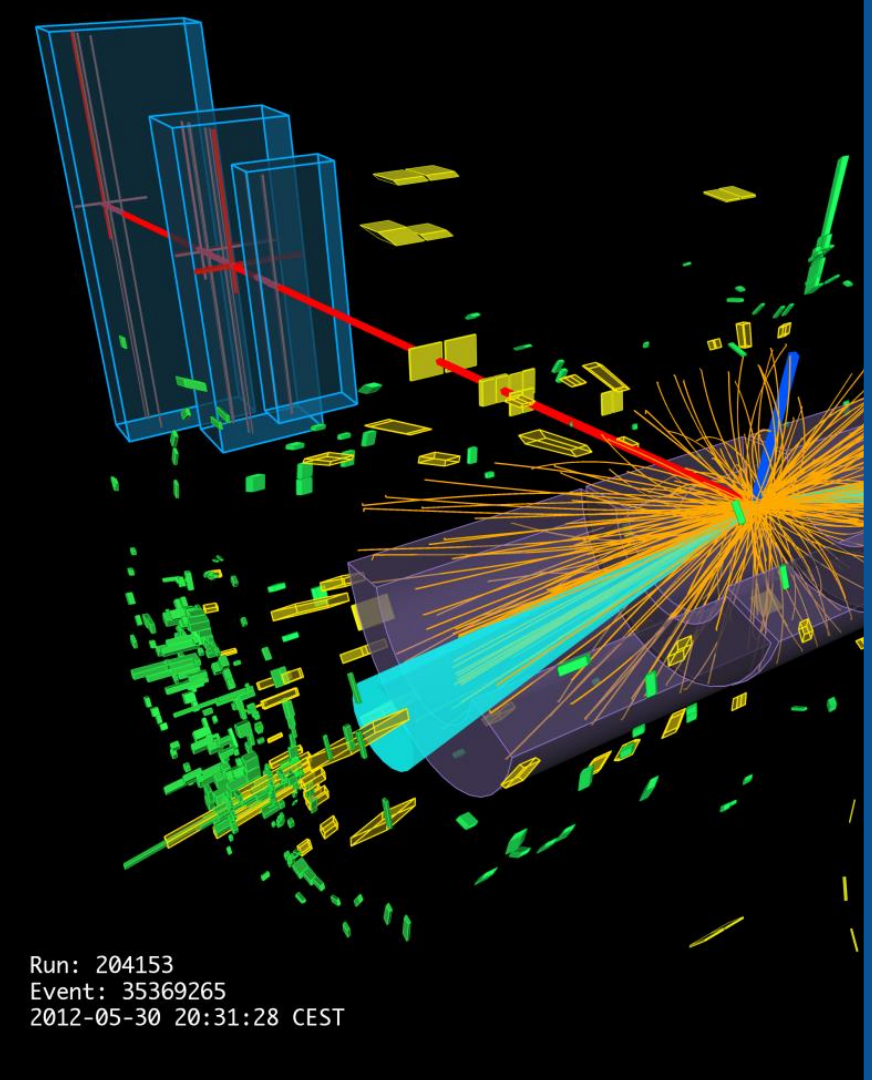


Fast Deep Learning on FPGAs for the Phase-II L0 Muon Barrel Trigger of the ATLAS Experiment

IML2019
April 17th 2019

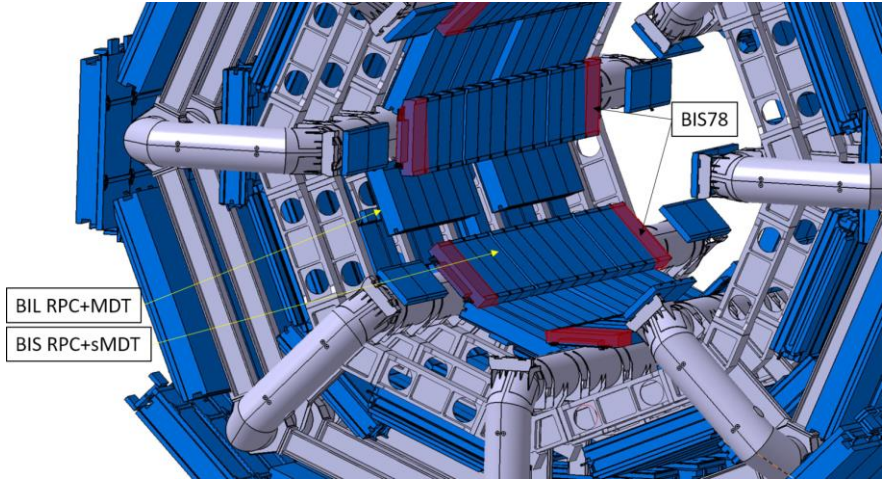
Luigi Sabetta

Motivations

The ATLAS Level-0 muon trigger will face a complete upgrade HiLumi-LHC

Detector parameters:
- Pile up: 30-40 → 200
- Luminosity: (2 → 7.5) × 10^{34} $cm^{-2}s^{-1}$

Greater muon hit rate in the spectrometer:
Up to 600 Hz/cm^2



ATLAS improvements:
New trigger processor
- New FPGA based system
  - Virtex UltraScale+ XCVU13P
  - Logic cells (K): 3780
  - Memory (Mb): 455
  - GTY Transreceivers (32.75 GB/s): 128
  - I/O Pins: 832

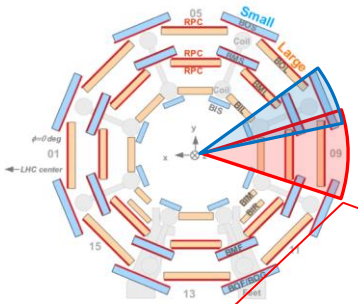
New trigger station
- New RPC layer

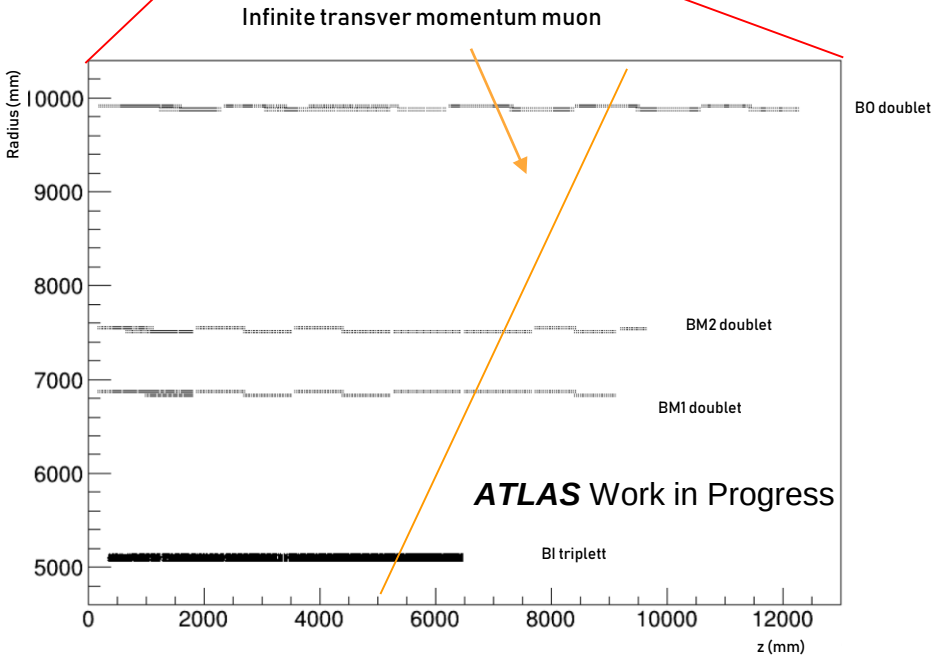
Trigger algorithms improvement
- Need to be very fast and flexible

Neural networks → valid candidate

ATLAS-TDR-026

Available information for trigger



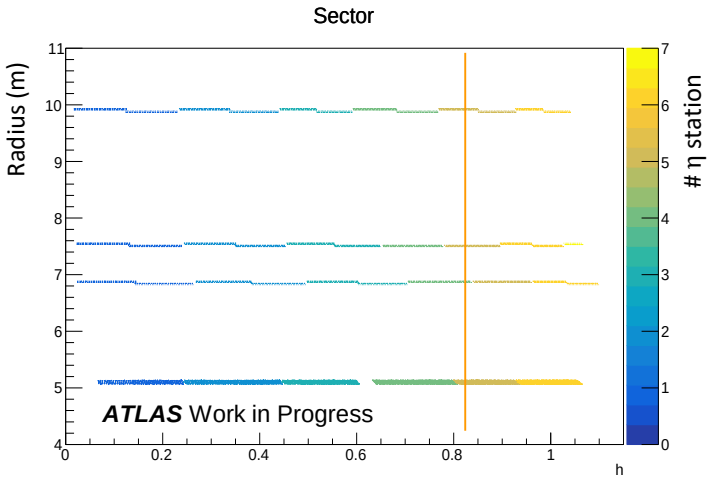


Resistive Plate Chamber(RPC):

- Fast gas detector
- Trigger sources in ATLAS

Sector:

- Divide ATLAS in 2 sides ($z \gtrless 0$)
- Divide φ coordinate in 8 sectors, each one divided in 2 (Small and Large)
- Take all the strips from the obtained sector



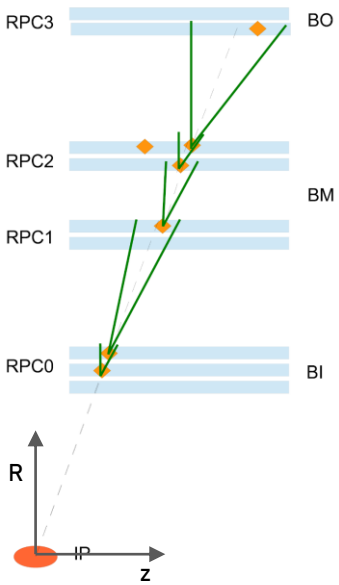
Atlas Trigger Requirements

Design Trigger Requirements→ up to three candidates

Candidate =muon with p_T>threshold, + extimate of the position (η^{muon})

Processing time < $6\mu s$

Standard trigger Algorithm



ATLAS-TDR-026

Standard algorithm:

- Check the presence of coincidences inside windows sequentially opened based over the previous layers's hits

👍 Stable and reliable

👍 Good performances

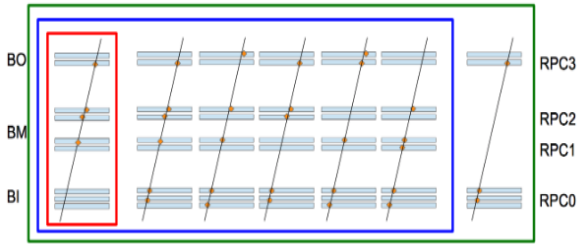
👎 The coincidence windows need to be tuned "by hand"

- Windows dimensions depend over the p_T trigger threshold
- Strong dependency over the trigger choices and local detector geometry

👎 Assumes pointing tracks

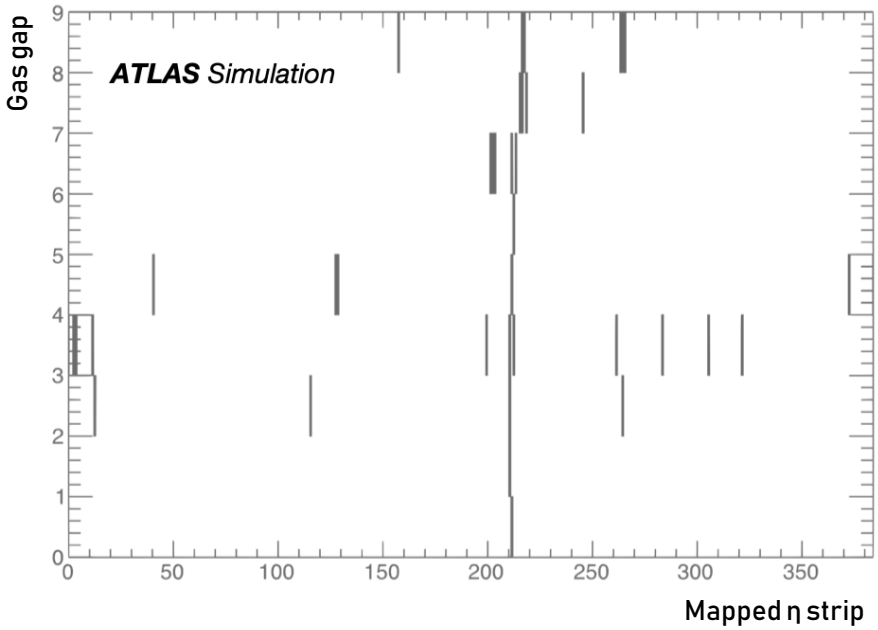
- Decay vertex far from the IP?

👎 No p_T extimate



From strip map to images

https://twiki.cern.ch/twiki/bin/view/AtlasPublic/L1MuonTriggerPublicResults



Example of a muon with p_T=19 GeV + noise

Mapping → $$i^{strip} = \frac{\eta_{hit} - \eta^{min}}{\eta^{max} - \eta^{min}} \times 384$$

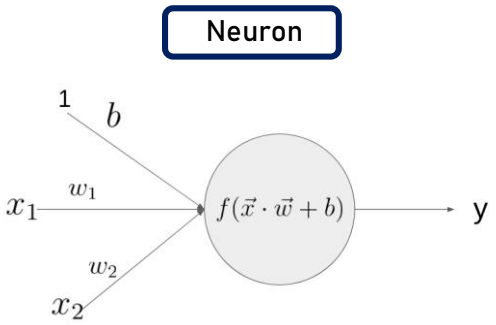
In this way, Images can be built:

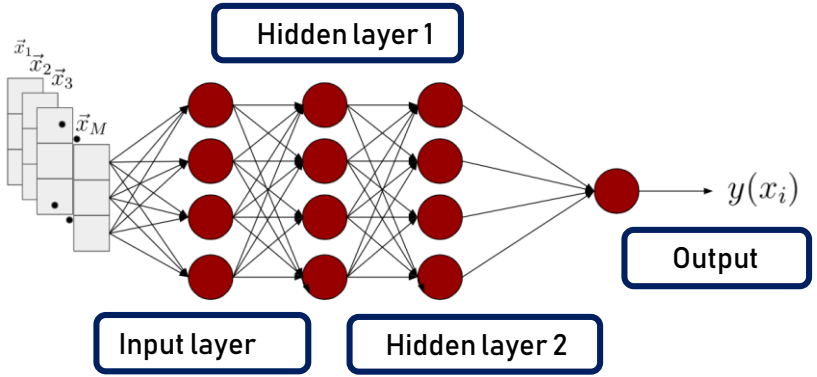
- Minimal case 384 x 9 pixel
- Muon → straight line

Problem:

- Images are highly sparsified

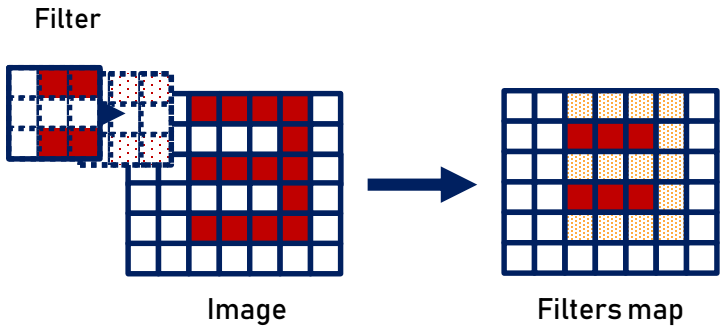
Convolutional Neural Networks





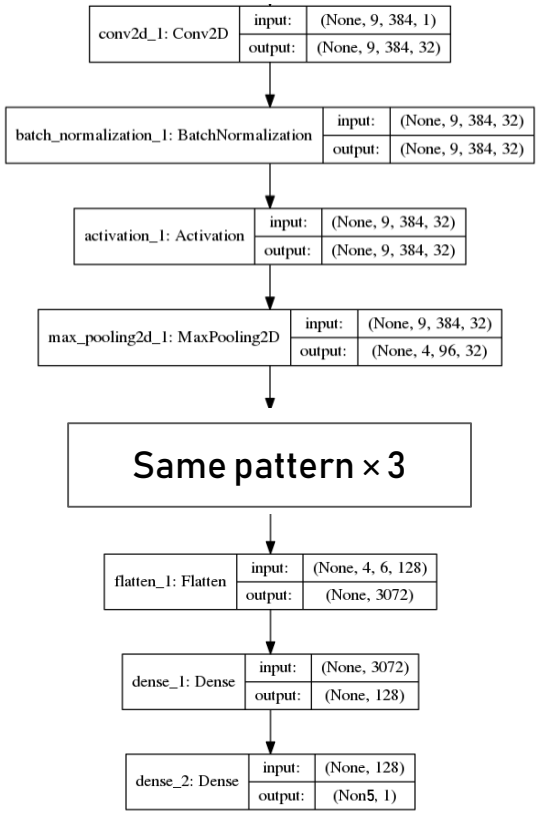
Convolutional Neural Network (CNN): Deep Neural networks optimized for image recognition.

Highly effective for problems with rotational or translational symmetries



In this way the number of the parameters can be highly reduced

CNN Floating Point structure



Filters [3 × 3]

Pool. [2 × 4]

Architecture:

- (Conv2D + Batch Norm. + Max Pooling) x 3
- (Dense) x 2 layers to get to the output

Total number of parameters: 500k

Decaying Learning rate: $10^{-2} \rightarrow 10^{-5}$

Activation= ReLu

BatchNormalization:

- $\epsilon = 10^{-6}$
- Momentum= 0.9

Input DataSet

Sample:

- 1 muon + Background (random hits ~uniformely distributed)
- 2 muons + Background
- 3 muons + Background
- Just noise (10% of the total number of images)

$$0 < p_T < 20 \text{ GeV}$$

$$0 < |\eta| < 1.05$$

Parametrization of full phase-2 events in the ATLAS detector

Grand total of ~ 900k images

Output

The CNN is trained to give a 5D output:

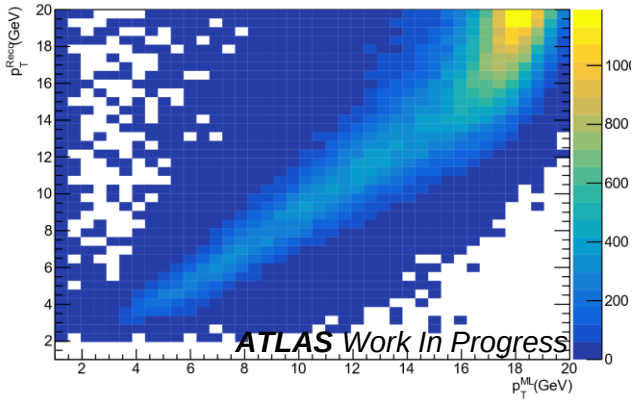
0 → 0/1/2 muons
-1 → more than 2 muons

$$(p_T^{lead},\quad \eta^{lead},\quad p_T^{sublead},\quad \eta^{sublead},\quad N^{muons})$$

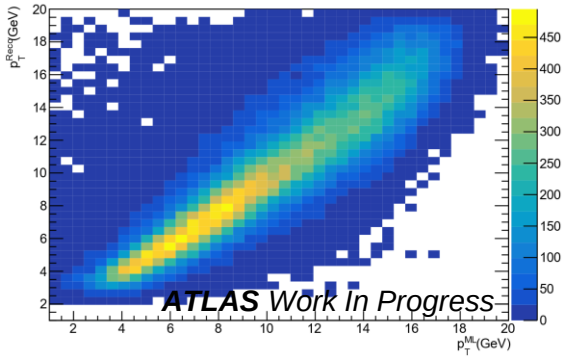
Standard algorithm
↓
"Is there a muon over threshold?"

NN
↓
"Is there a muon? What are its p_T and η?"

CNN FP performances– Physics quantities



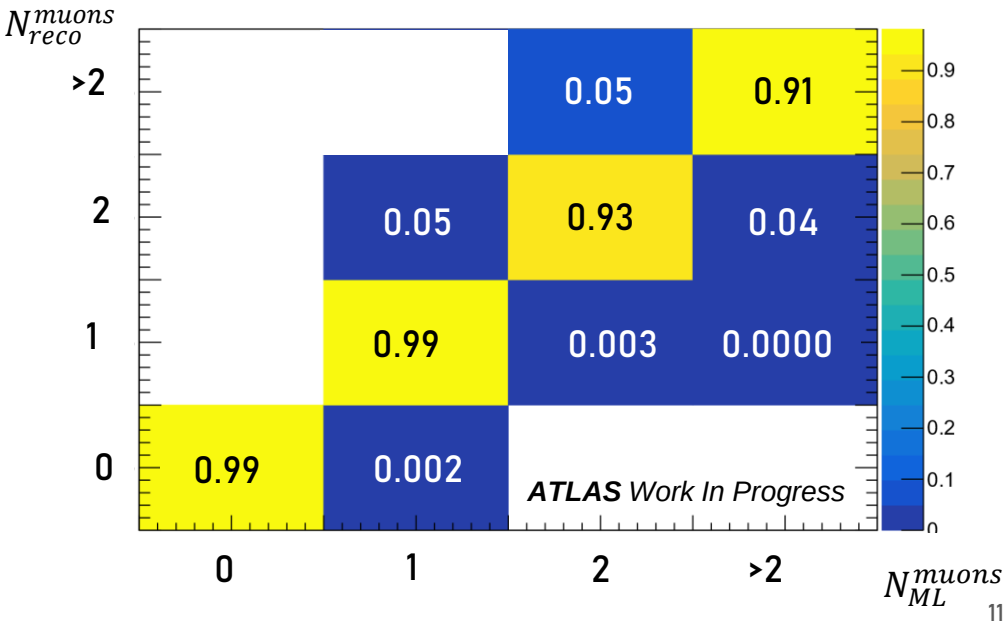
Leading (↑) and subleading (↓) muon



MAE= 1.9 GeV

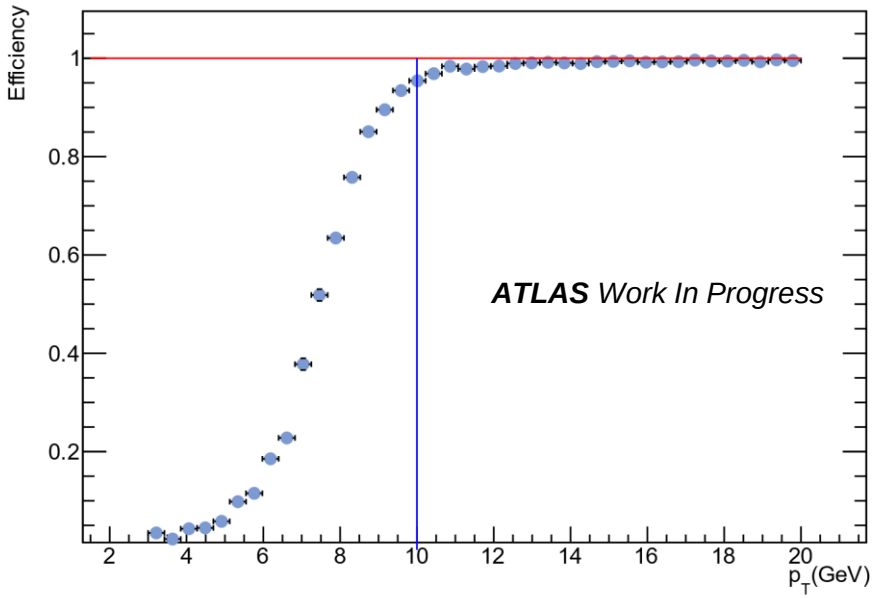
Interesting physics quantities are well represented

Events are well classified in n^{muons}

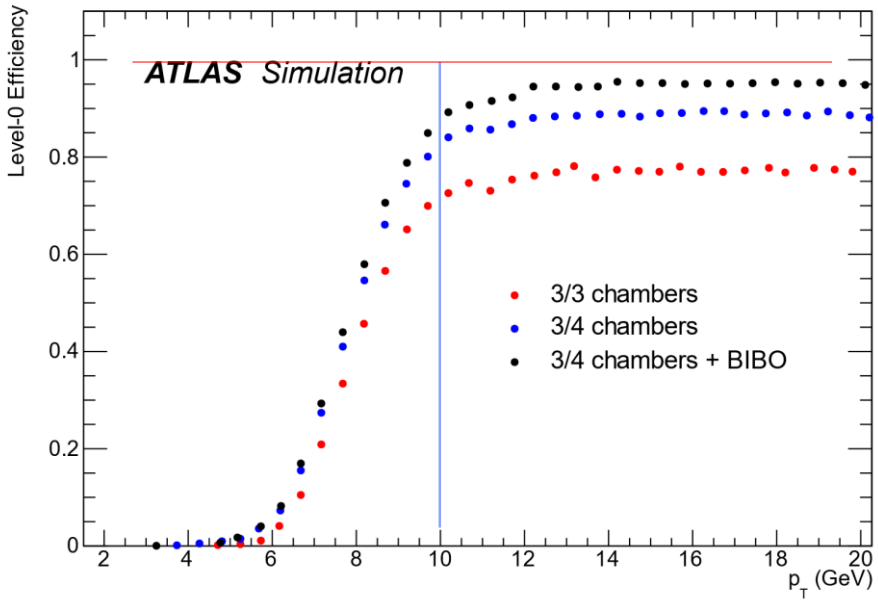


CNN FP performances- Trigger

CNN



Standard Algorithm



Efficiency curves are comparable to the ones obtained with the standard algorithm

Ternary CNN – implementation

Level-0 trigger sector logic will be implemented in an FPGA

- Standard NNs work with weight described by 32 bit floating Point precison numbers
- FP weights aren't the optimal choice
  Great logic resources consumption

Ternary CNN:

Weights = 

(only two bits)

A Ternary CNN may imply a loss in performances

- Few percentage points in respect to an FP32 NN with the same structure

Smaller size:

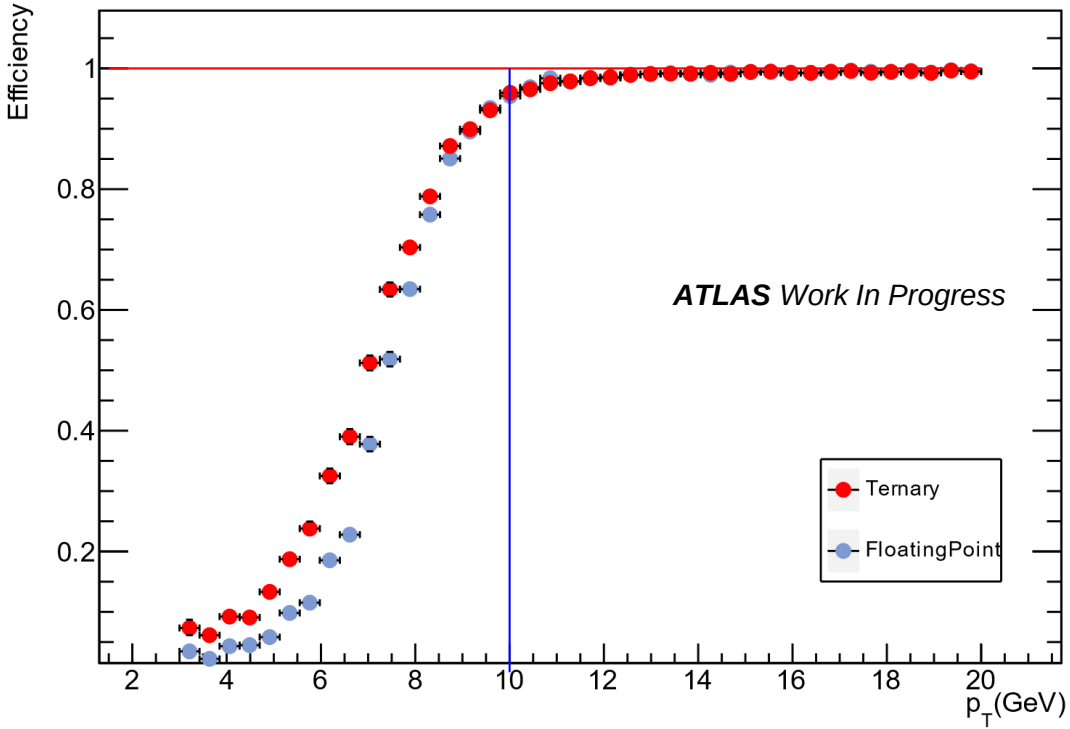
- Smaller logic resources consumption
- Up to 16 times smaller

In principle it can be made deeper

- More layer recover the loss in precision

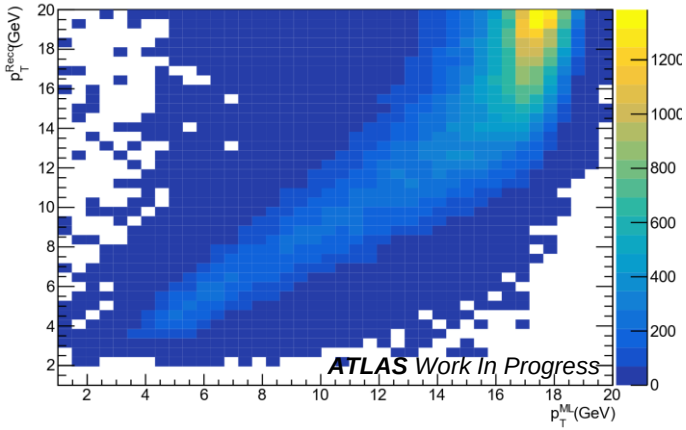
Ternary CNN- Performances

MAE= 1.9 GeV

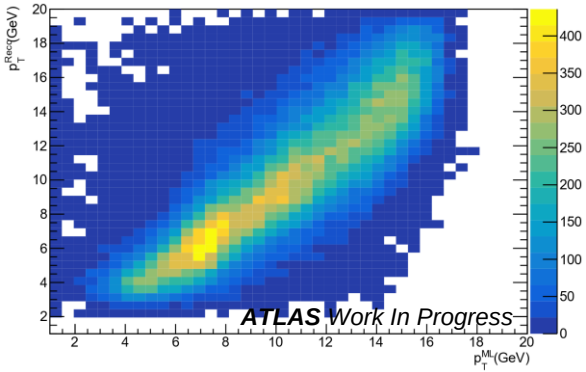


$\mathbf{eff_{FP} = 99.1\%}$ $\mathbf{fake_{FP} = 32\%}$

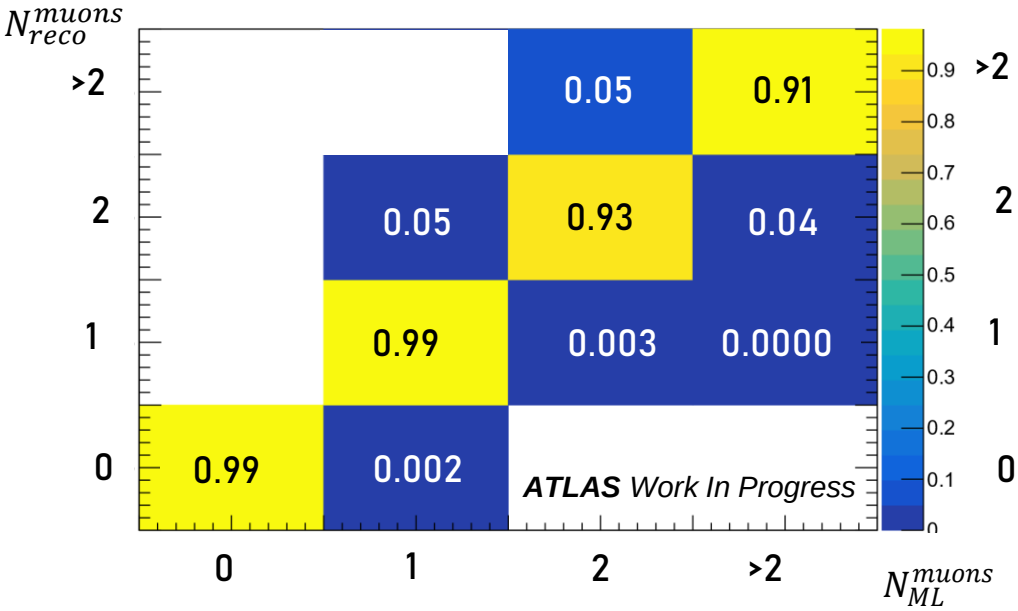
$\mathbf{eff_{T} = 99.0\%}$ $\mathbf{fake_{T} = 37\%}$

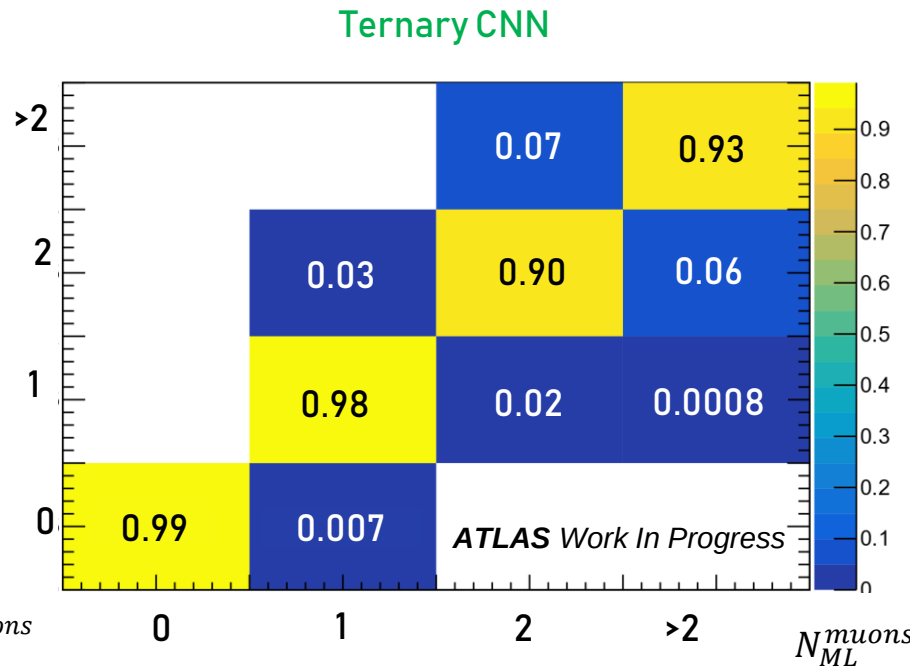


Leading (↑) and subleading (↓) muon



Ternary CNN performances– Number of muons





NN on FPGAs – How to Implement

Python
Architechture

Long programming
operations

VHDL/Verilog → Synthesis and Implementation

NN on FPGAs – How to Implement

Python
Architechture

We are using HLS4ML to implement firmware algorithm in FPGAs

C/C++ → Vivado HLS → VHDL/Verilog → Synthesis and Implementation

Current Status

Dense NN → Succesfully implemented

- Clock cycles = 660 → Latency = 1.2µs (Frequency=500MHz)
- DPS = 207

Ternary Dense NN → Succesfully implemented

- Clock cycles = 485 → Latency = 0.95µs (Frequency=500MHz)
- DPS = 102

Next Steps

Current Goal → Synthesize our CNN in the Ultrascale+ FPGA

Work in progress

↓

Fine tuning of the CNN

↓

Test the algorithm at the trigger slice test
(planned to be in the next future)

Thank you for the attention