hls4ml: deploying deep learning on FPGAs for L1 trigger and Data Acquisition

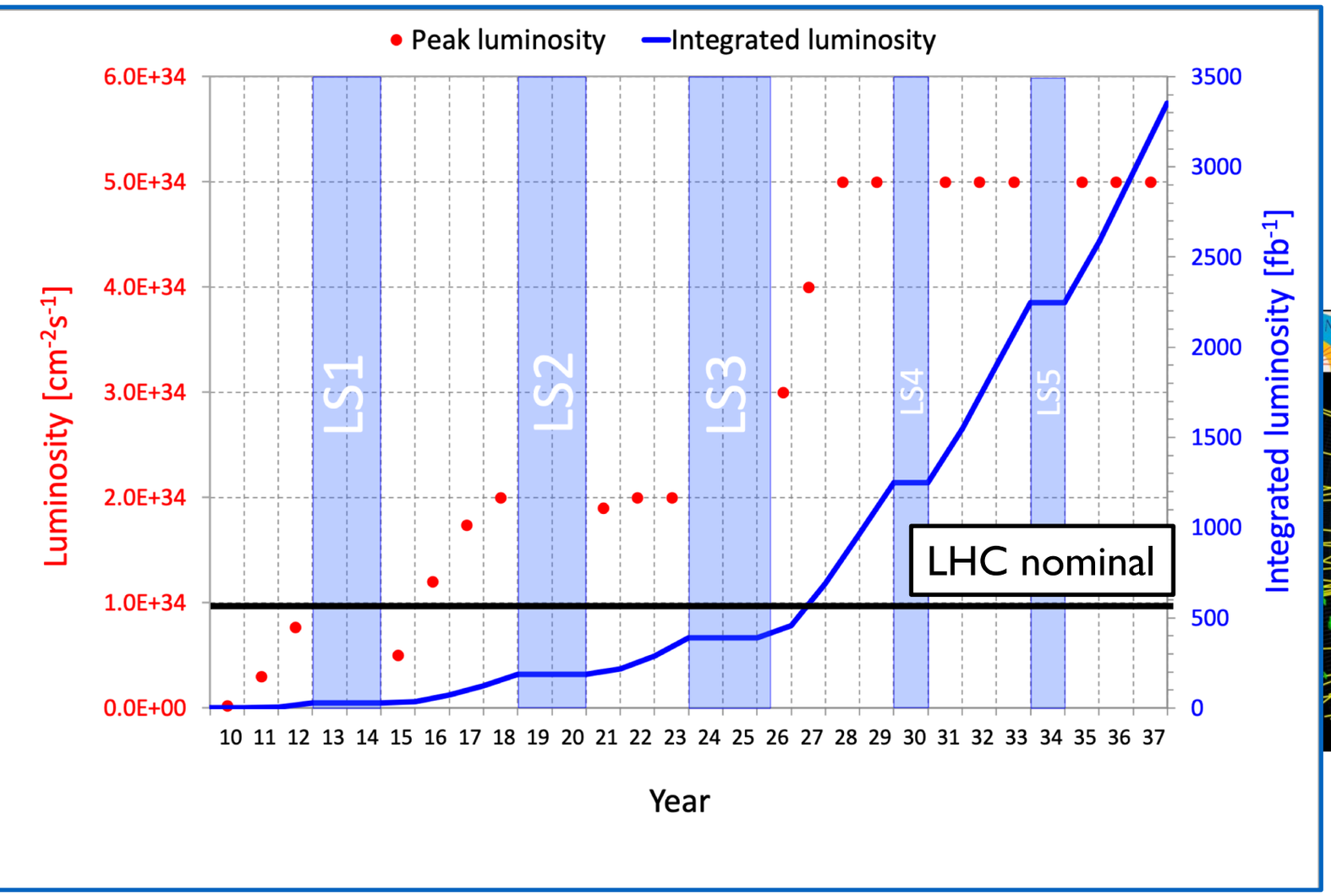
Sergo Jindariani (Fermilab)

On behalf of the hls4ml team

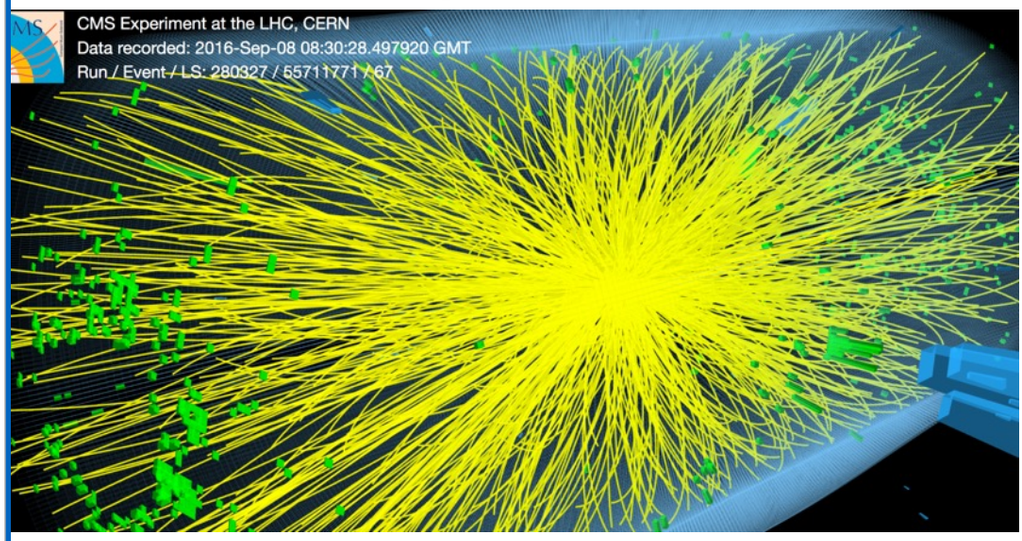
https://fastmachinelearning.org/

CPAD Instrumentation Workshop, Madison WI, 2019

The Challenge



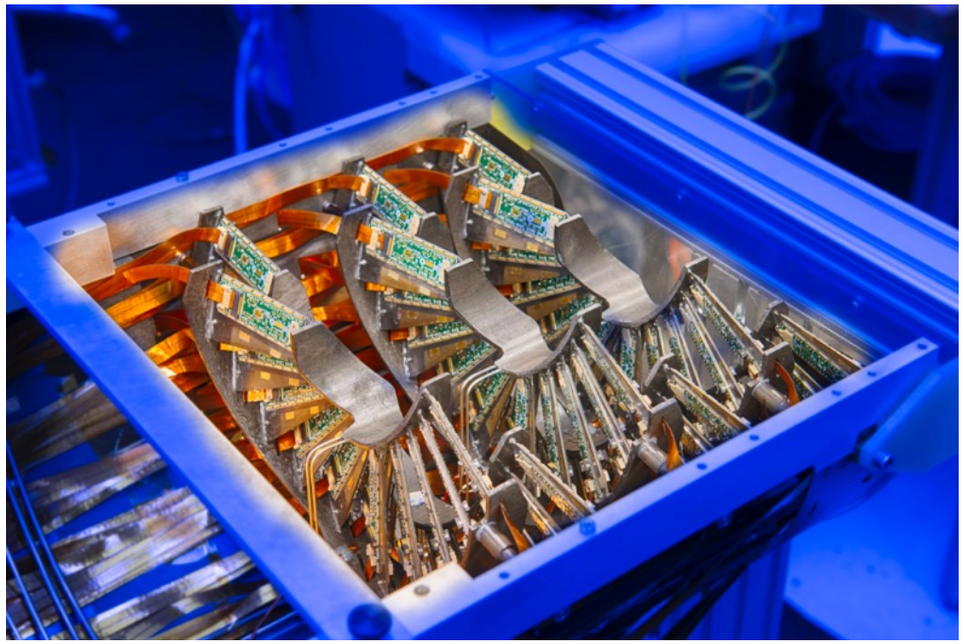
Interaction region is ~10 cm in Z

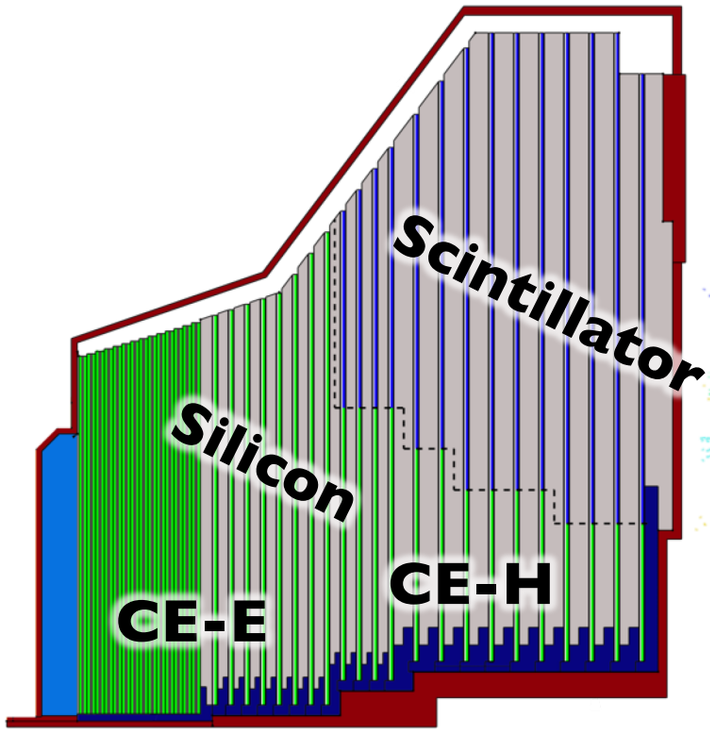


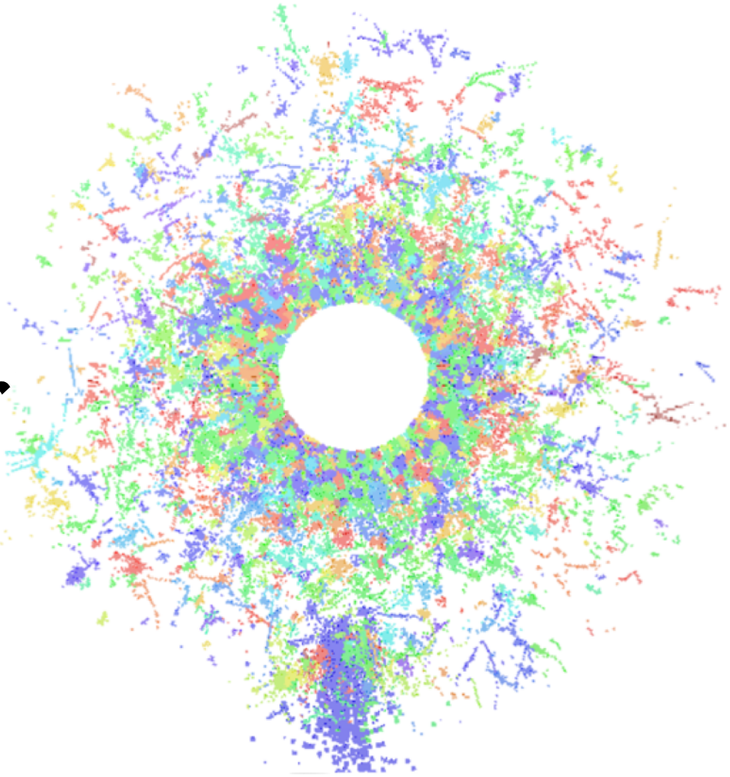
- The LHC will be running in the high luminosity mode
- More data => more physics, but also more PileUp.
- Currently up to 70 collisions per event.
- In HL-LHC the average number will go up to <PU>=200

Detector Complexity

Reconstruct thousands of clusters from 100k's of hits



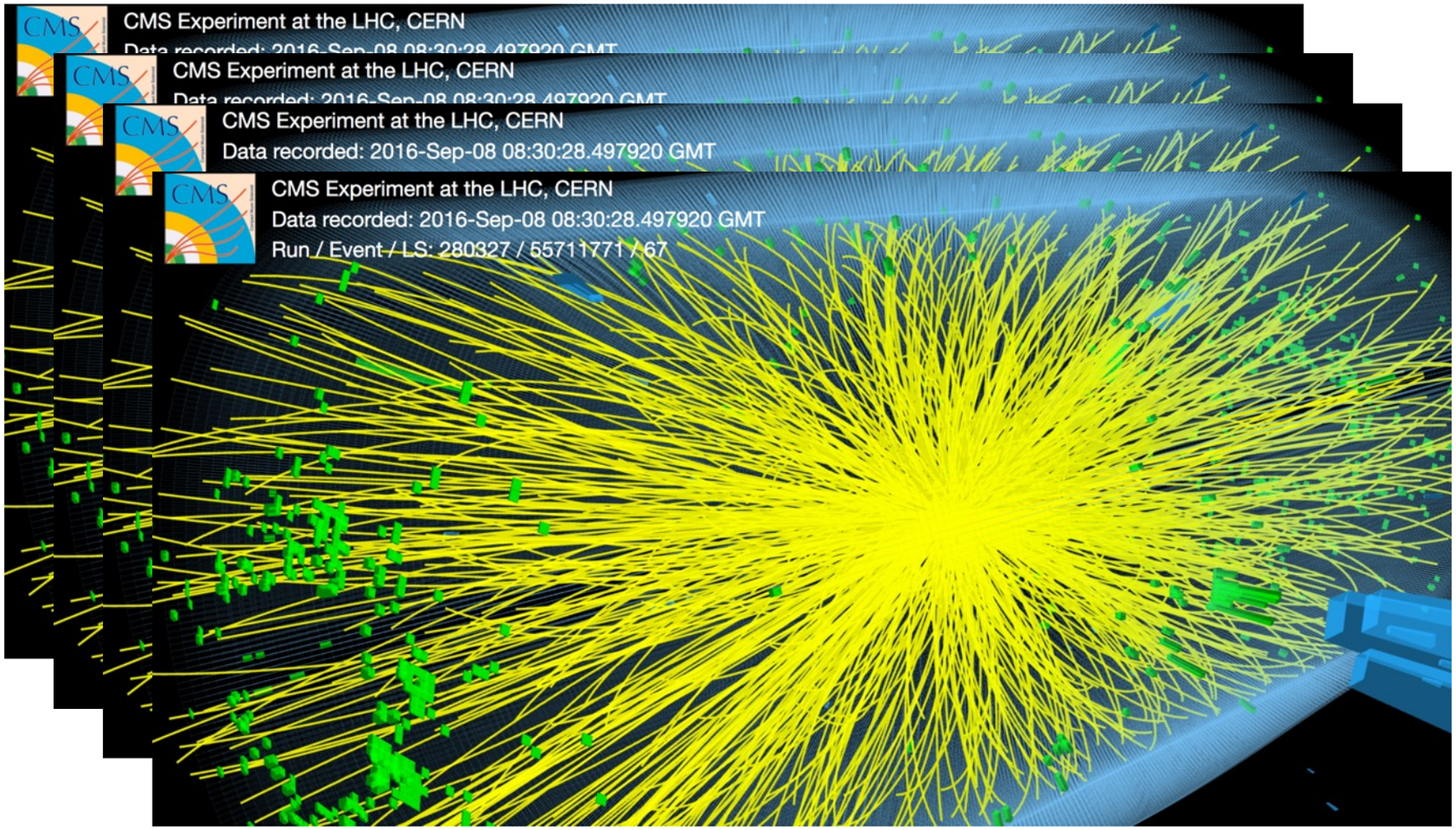




CMS pixel	Number of channels
'Phase-0'	66 M
'Phase-1'	123 M
'Phase-2'	2B

CMS ECAL+HCAL vs HGCAL	Number of channels
'Phase-0'	~80k
'Phase-1'	~90k
'Phase-2'	6.5M

Challenges for Triggering

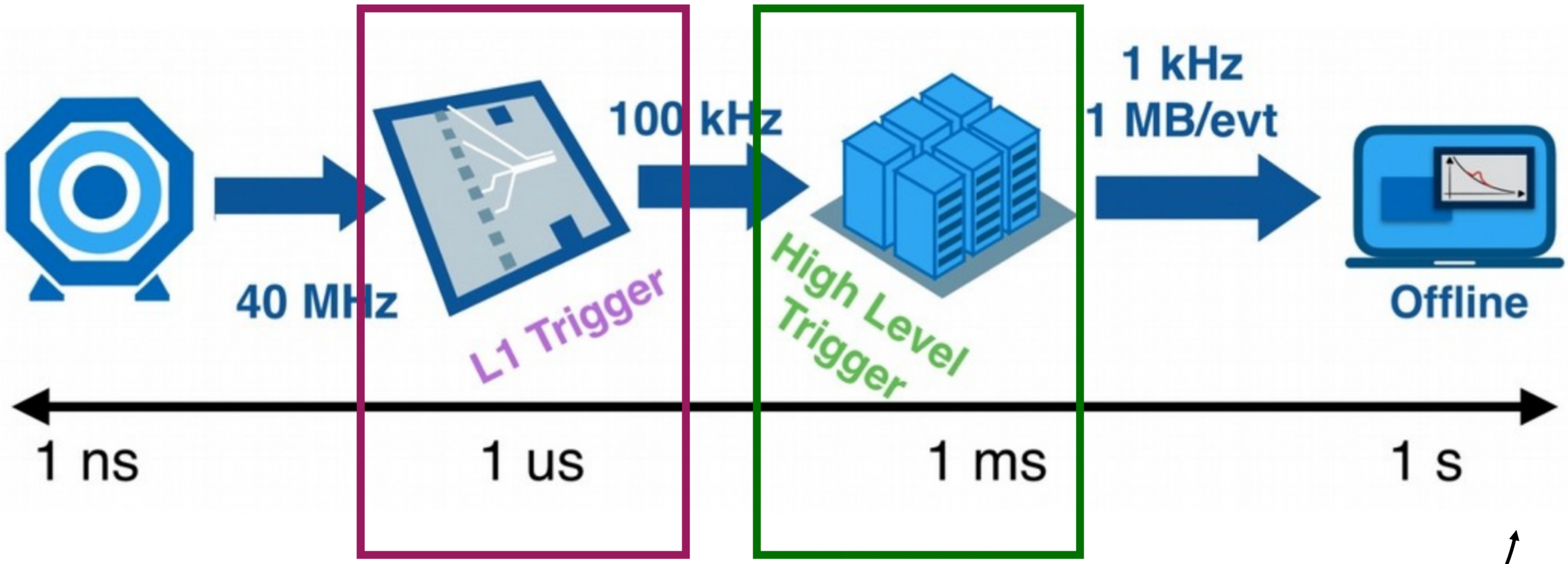


40 Million times per second

Down to 1kHz level to storage

Cannot make a mistake. Discarded data is lost forever.
Have at MOST 1μs to run an algorithm
We aim for algorithms that are in the 100ns range

ML in Data Processing



Employ ML algorithms in Level-1 hardware

I will mostly focus on this

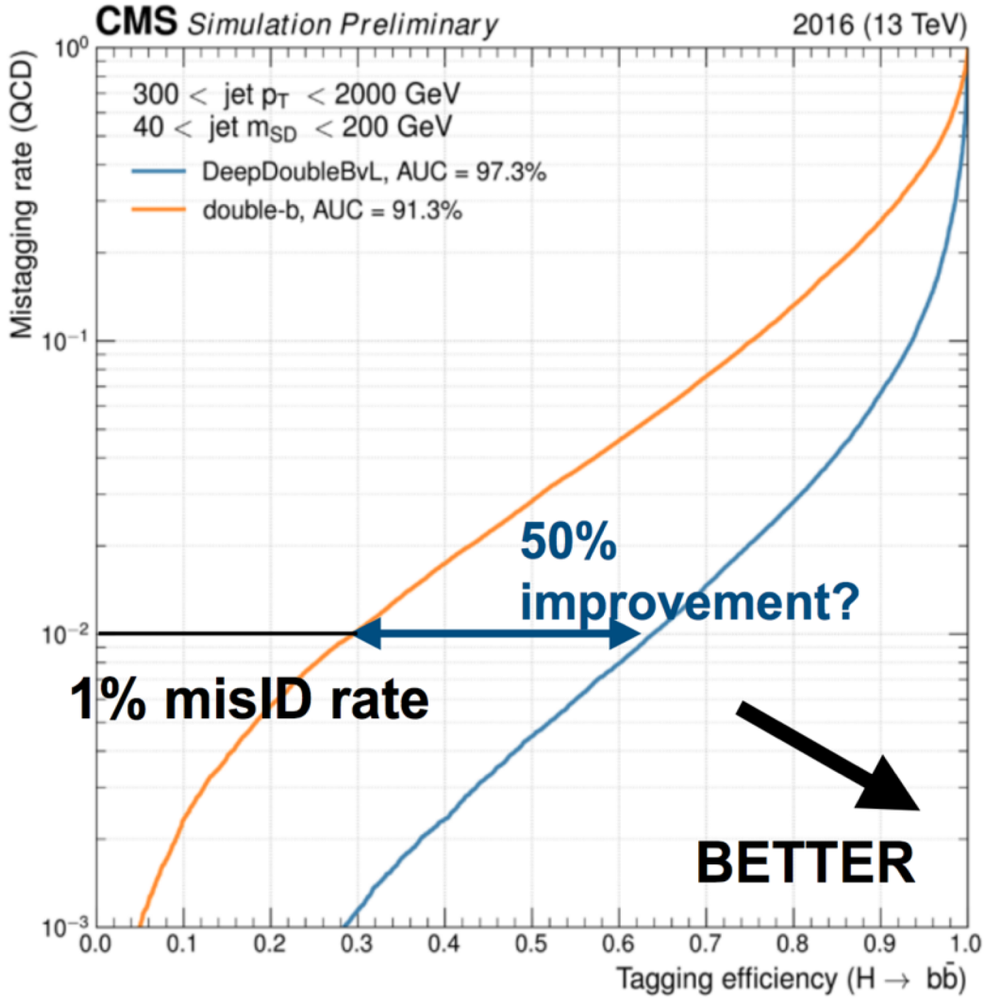
Rewrite CPU-intensive HLT/Offline tasks using ML

Accelerate using co-processors (e.g. FPGAs)

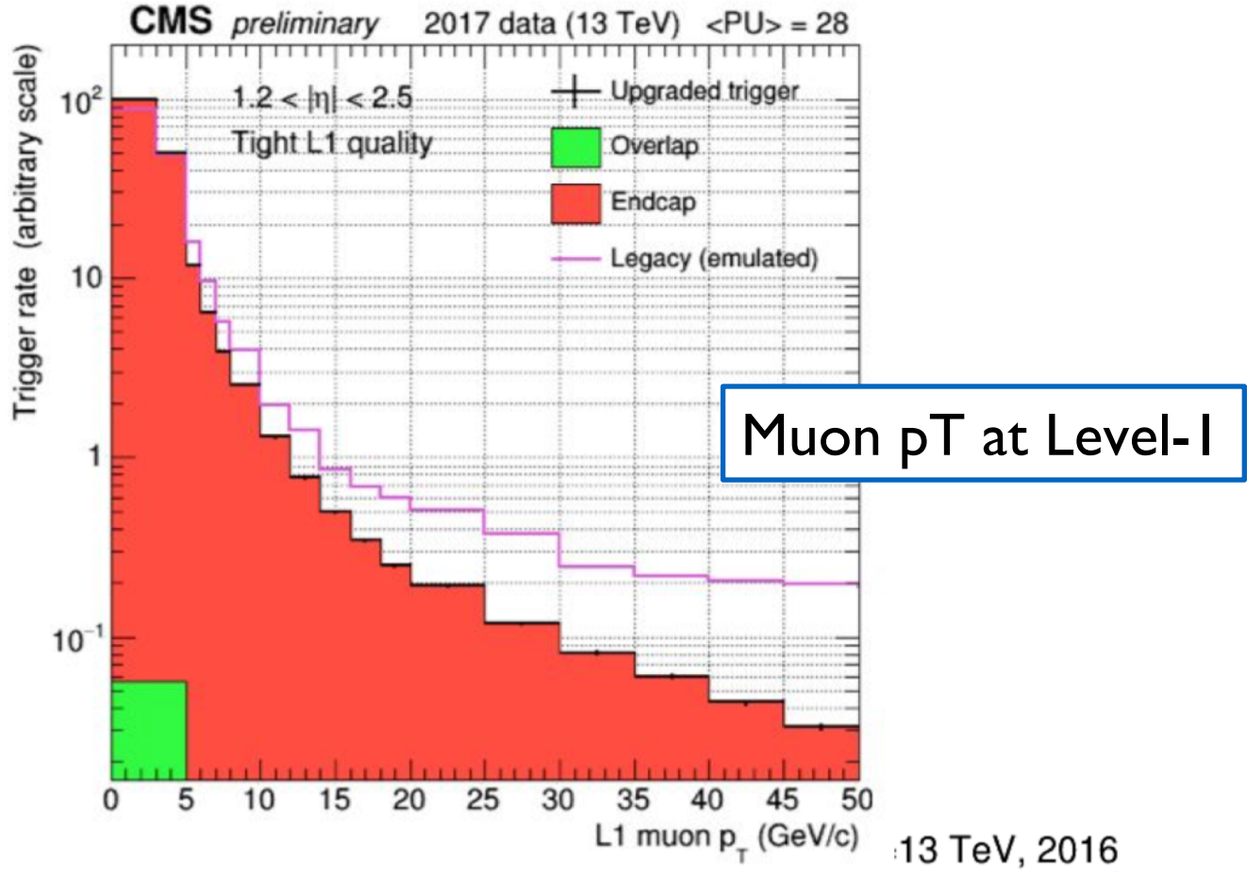
Talk by Nhan Tran

Can Machine Learning help?

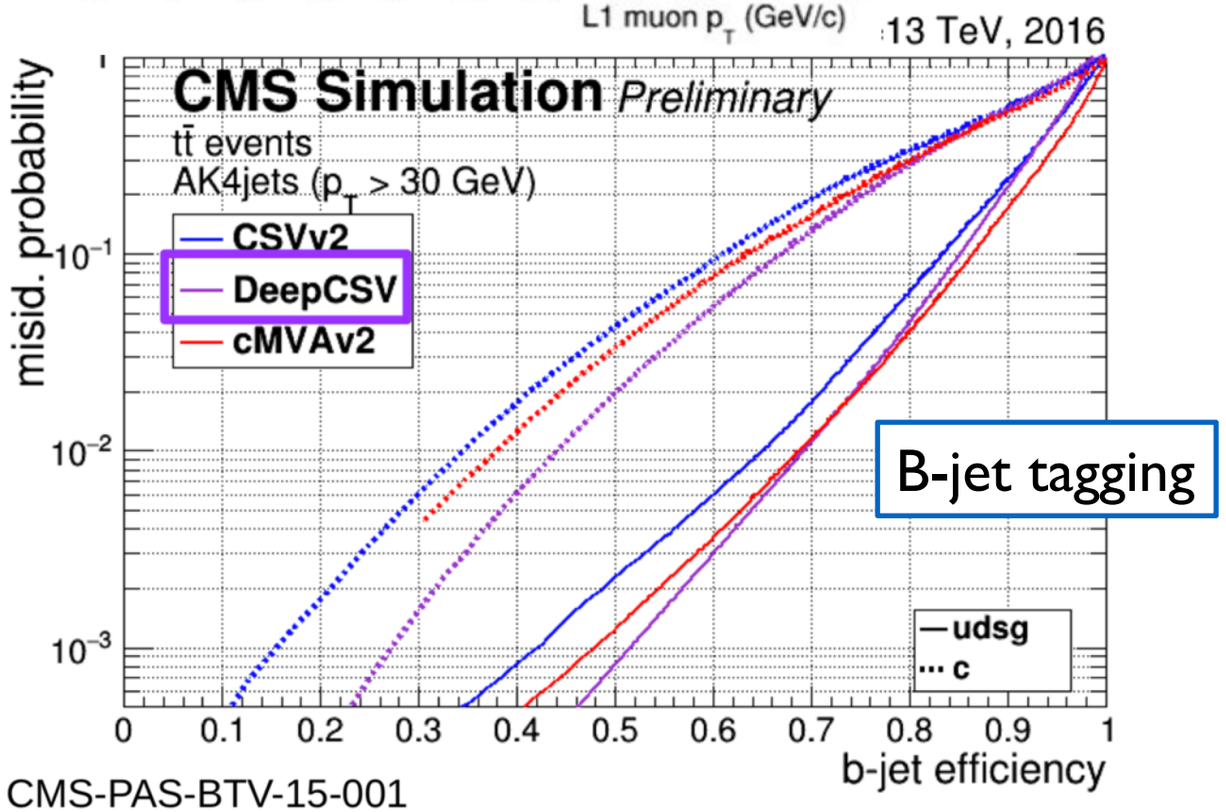
Higgs Tagging



Muon pT at Level-1



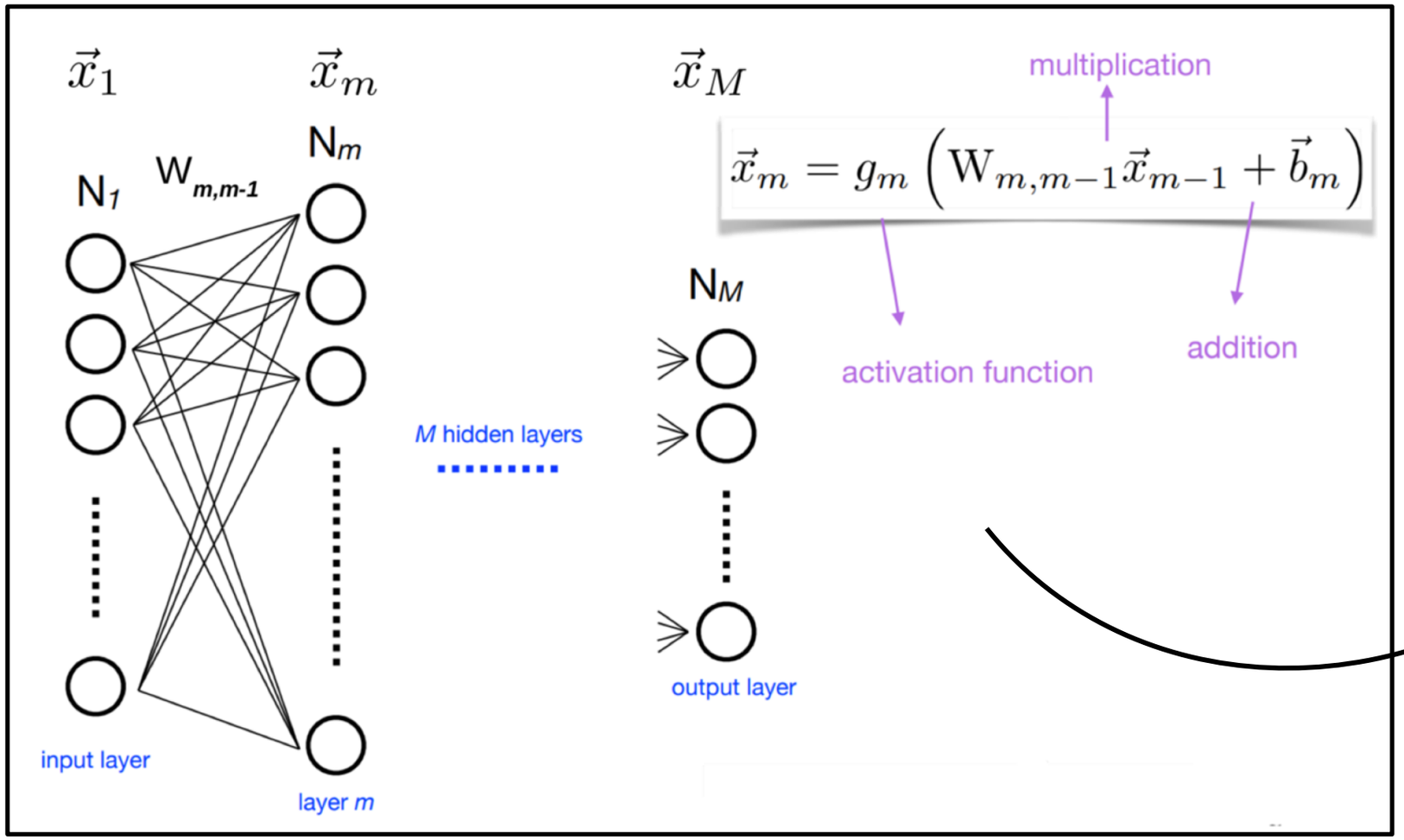
B-jet tagging



CMS-PAS-BTV-15-001

Mapping NN into FPGA

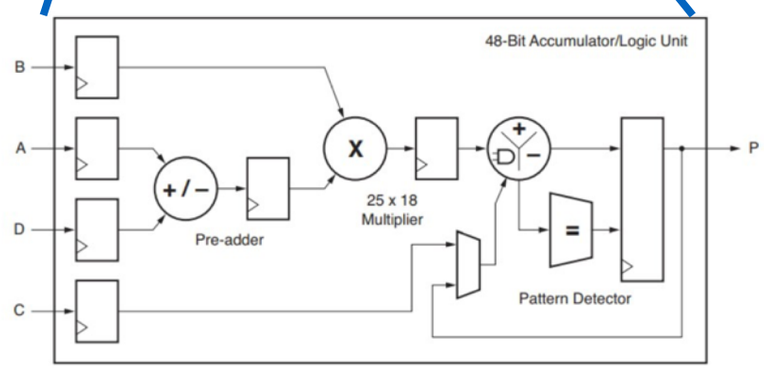
Typical NN operations:



Maps naturally into the functionality of DSP slices available in modern FPGAs

Device	# of DSPs
Kintex-7 325T	840
Virtex-7 690T	3600
Kintex UltraScale KU115	5500
Virtex UltraScale+ VU9P	6800

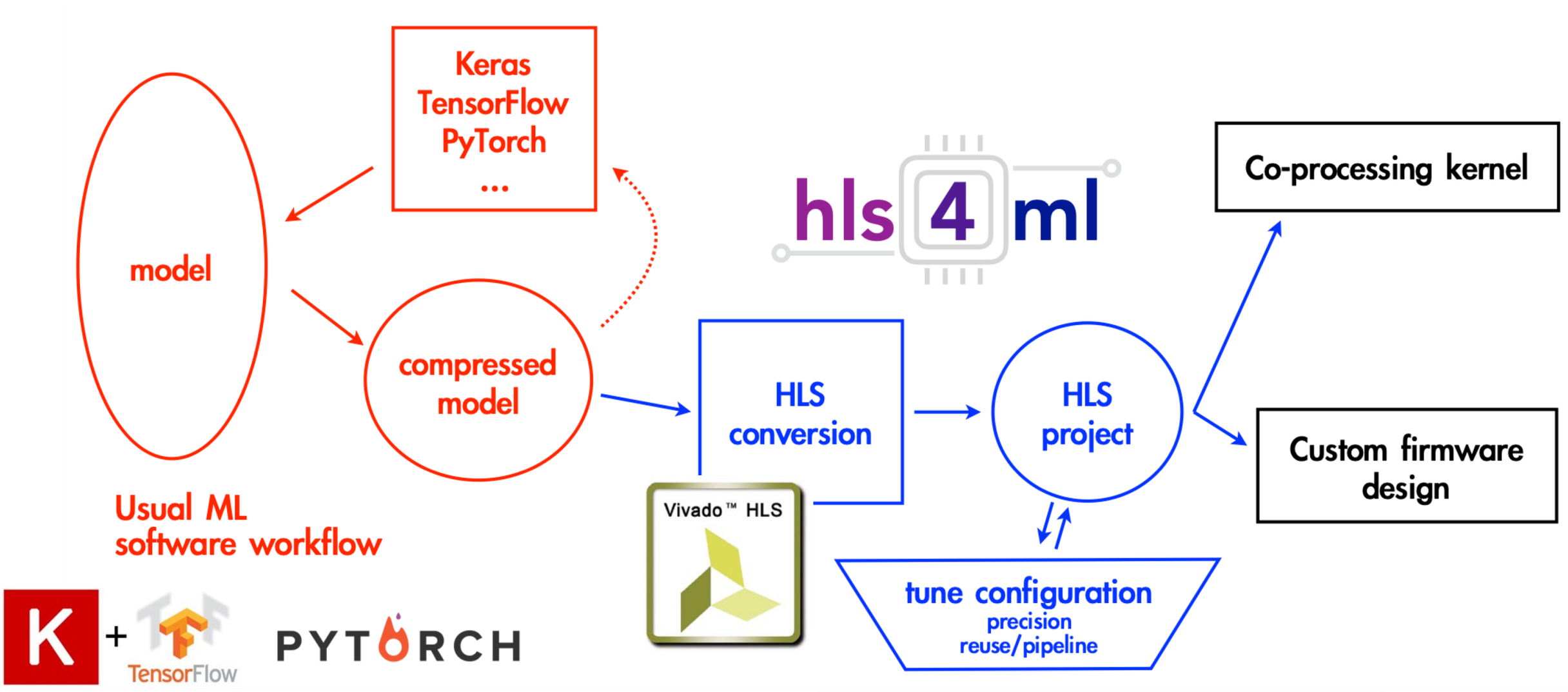
Basic DSP48E1 Slice functionality



What is hls4ml

User-friendly tool to build and optimize ML models for FPGAs:

- Reads as input models trained with standard ML libraries
- Uses Xilinx HLS software
- Comes with implementation of common ingredients (layers, activation functions, binary NN ...)



https://fastmachinelearning.org/hls4ml/

Architectures Supported

- **Quickly expanding:**

- DNN
  - Including support for large layers

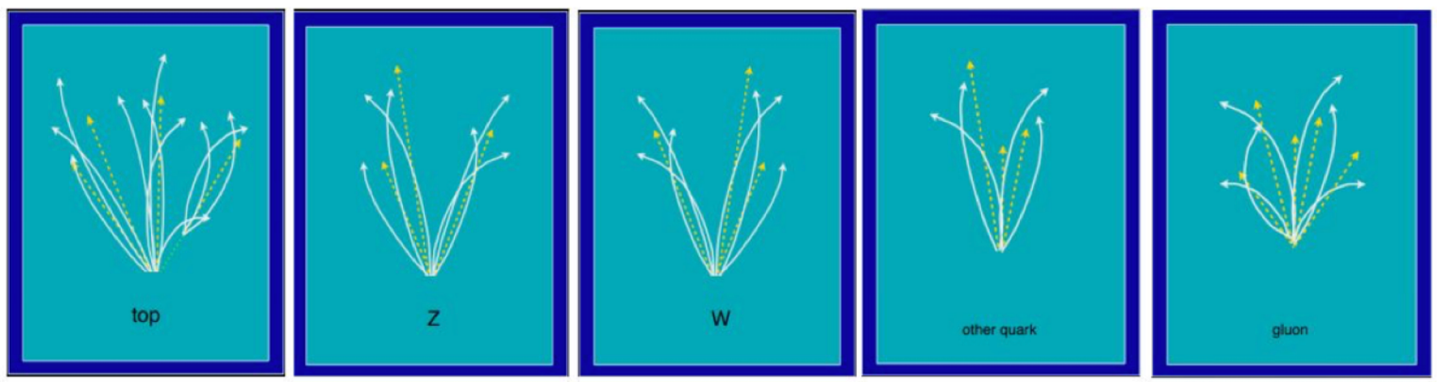
- Binary and Ternary DNN
  - Low precision (1 or 2 bit) weights => lower resource usage with very small loss in performance
  - Implemented in LUTs

- Conv1D and Conv2D (small)
  - Large Convs and Binary/Ternary coming soon

- Other features
  - Batch normalization
  - Various activation functions
  - Tools for comparing C and RTL simulation results

Benchmark Model

Multi-classification task:
Discrimination between highly energetic (boosted) **q**, **g**, **W**, **Z**, **top** initiated jets



t→bW→bqq

3-prong jet

Z→qq

2-prong jet

W→qq

2-prong jet

q/g background

no substructure

and/or mass ~ 0

Reconstructed as one massive jet with substructure

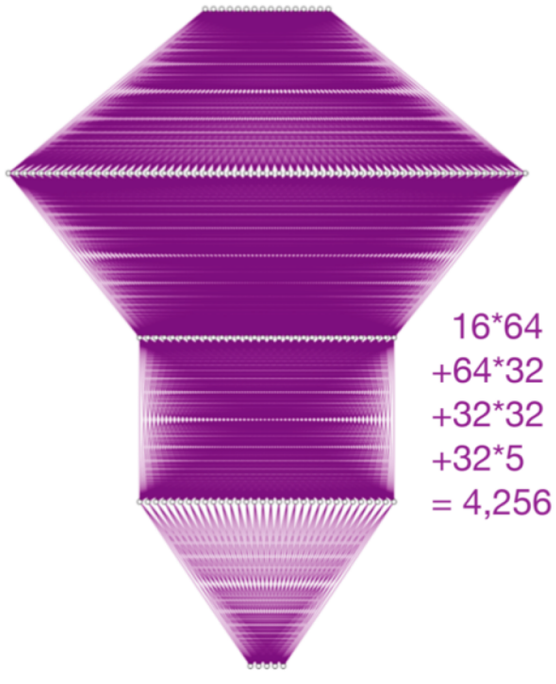
16 inputs

64 nodes
activation: ReLU

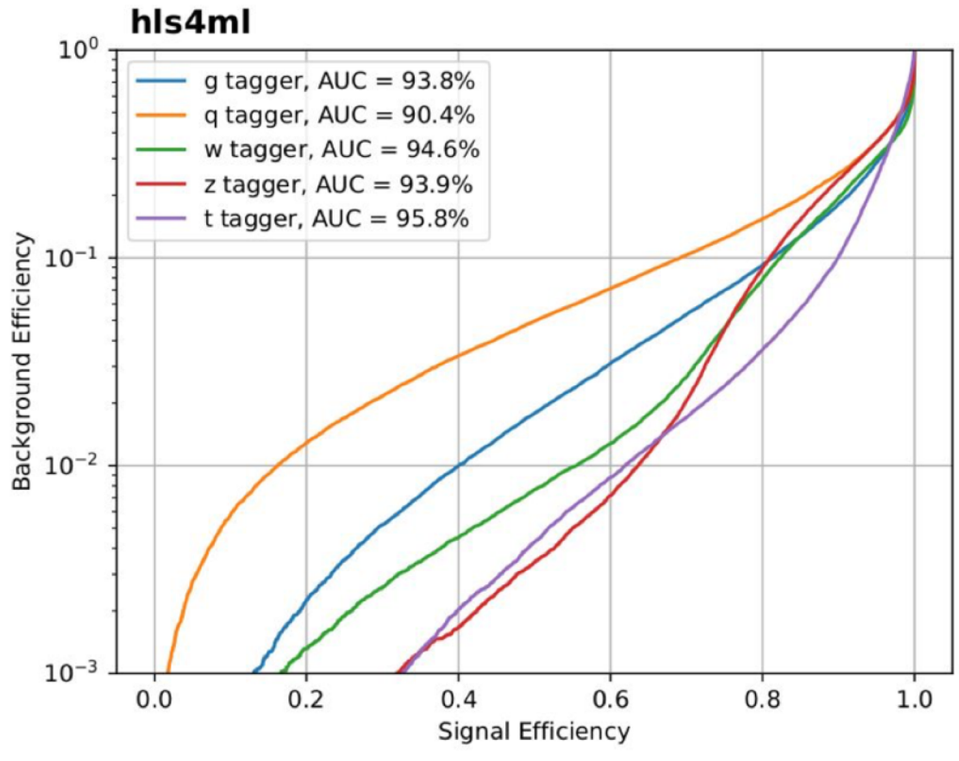
32 nodes
activation: ReLU

32 nodes
activation: ReLU

5 outputs
activation: SoftMax



16*64
+64*32
+32*32
+32*5
= 4,256 synapses

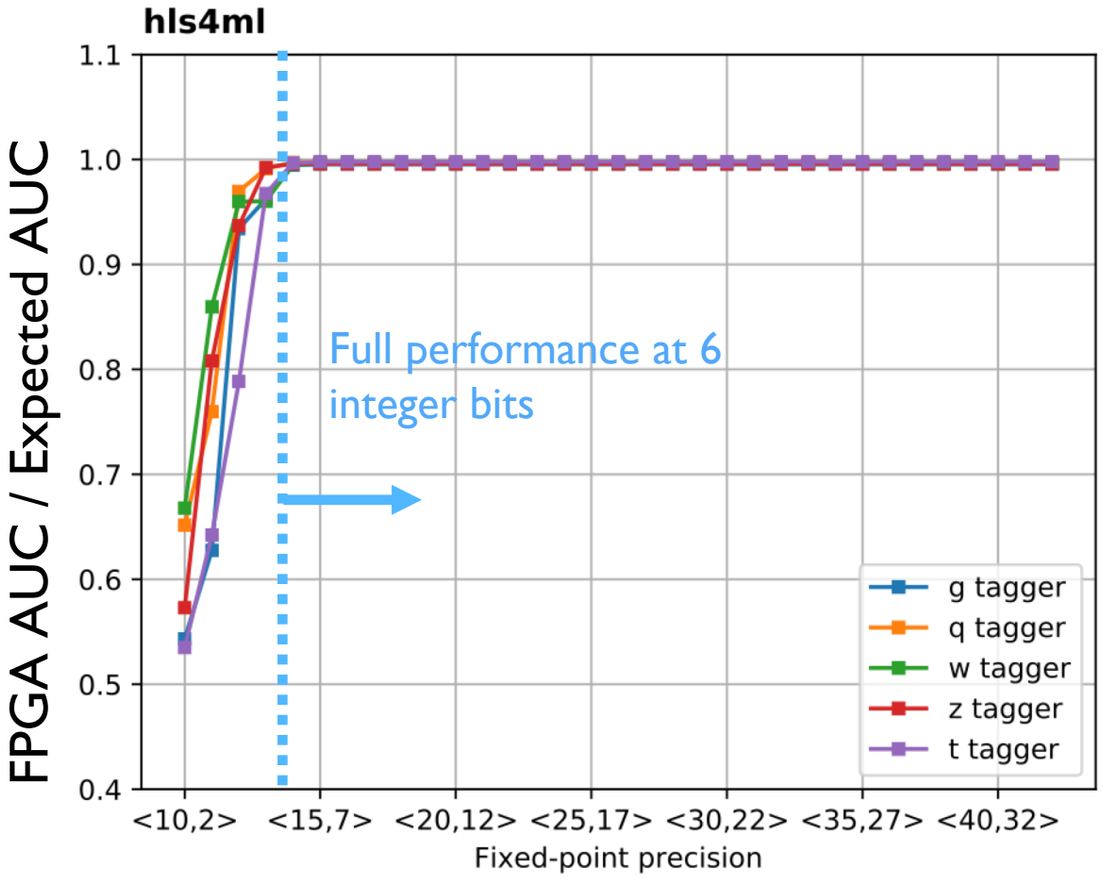


Quantization

Study performance of the NN in FPGA as a function of the number of bits

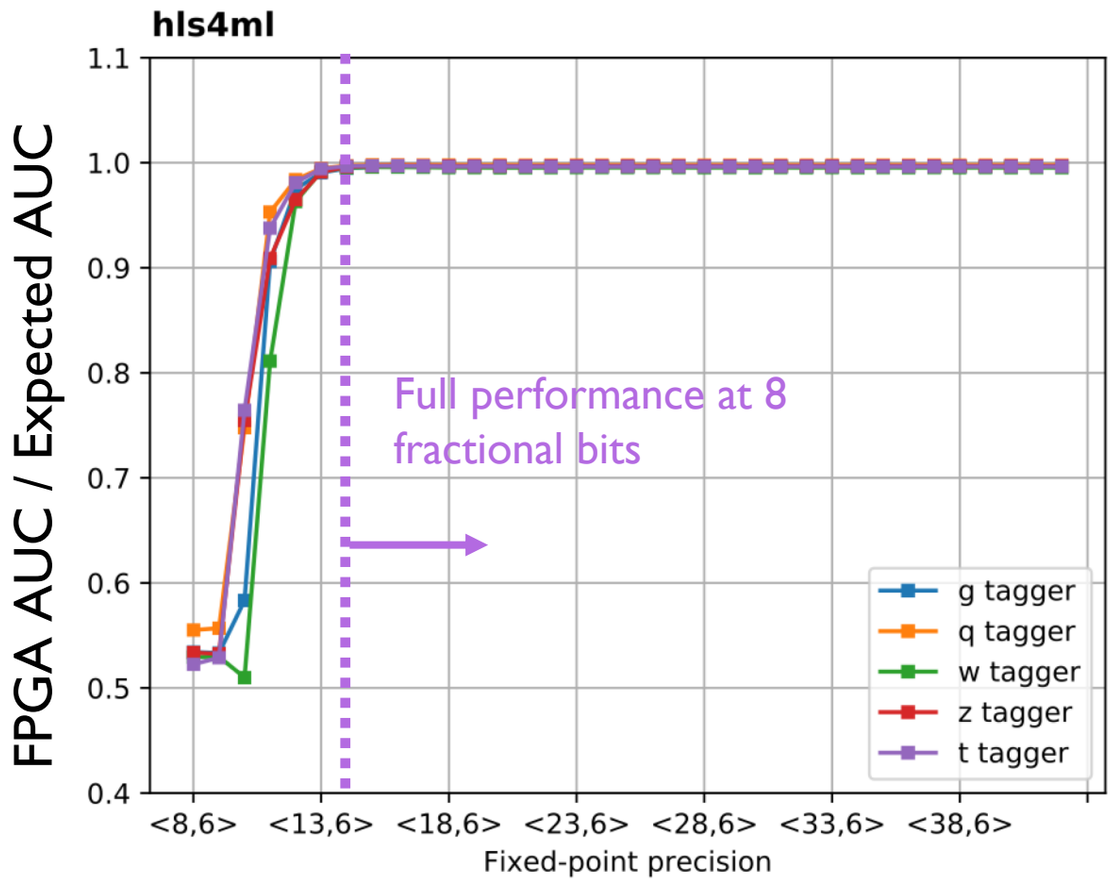
Scan integer bits

Fractional bits fixed to 8



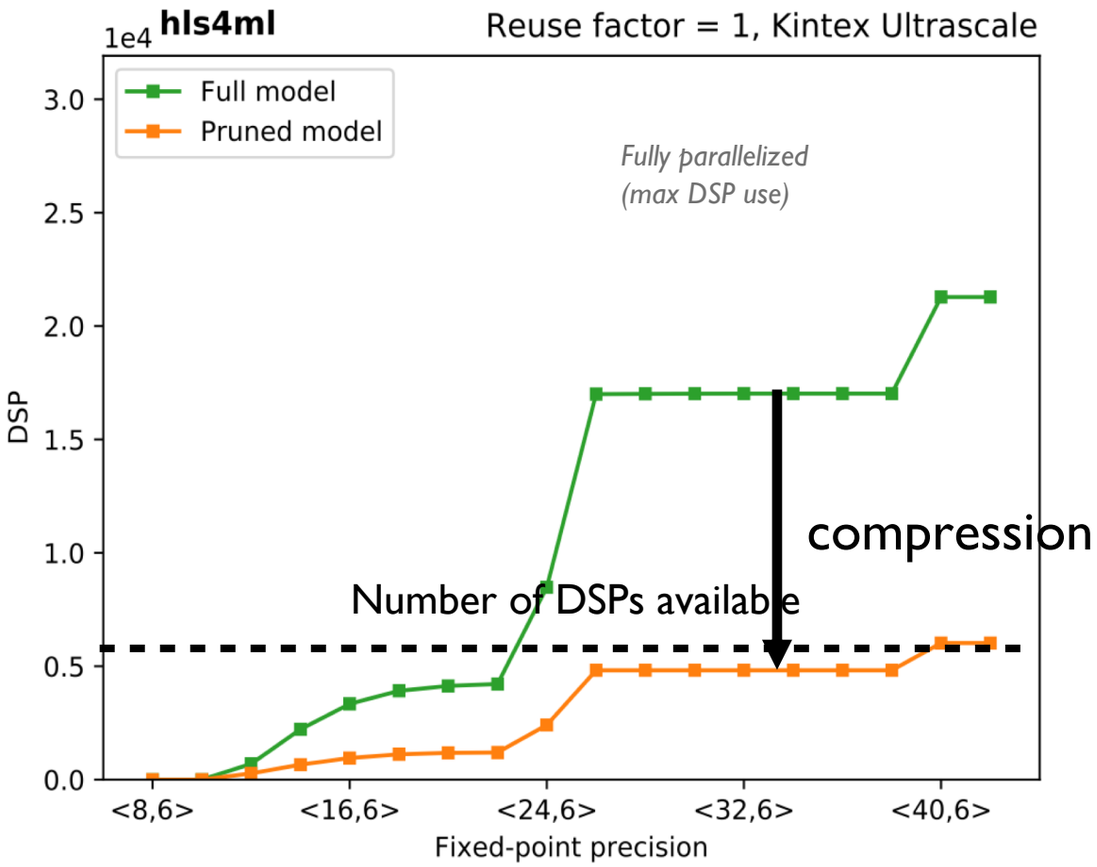
Scan fractional bits

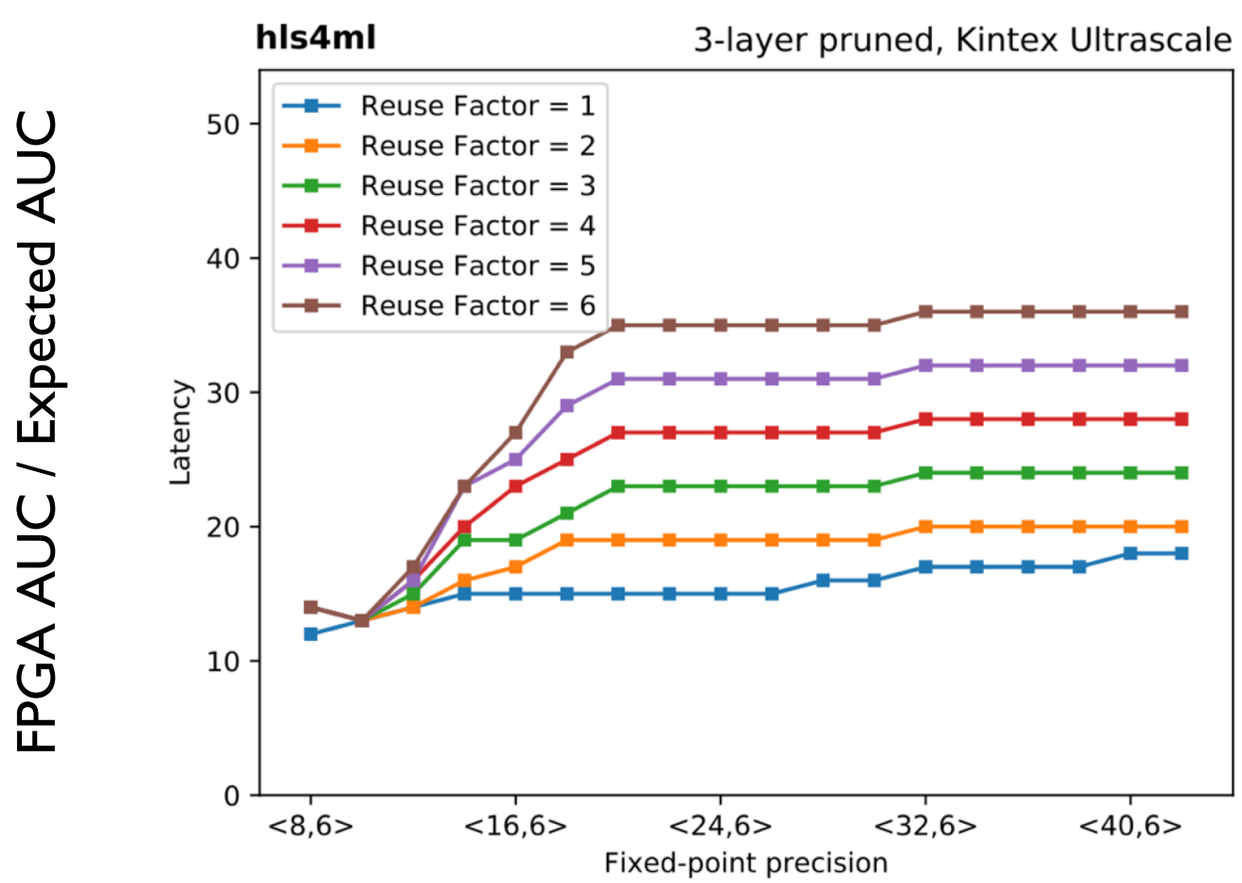
Integer bits fixed to 6



Pruning and Parallelization

Prune model to reduce resource consumption
Study model execution latency and throughput





FPGA AUC / Expected AUC

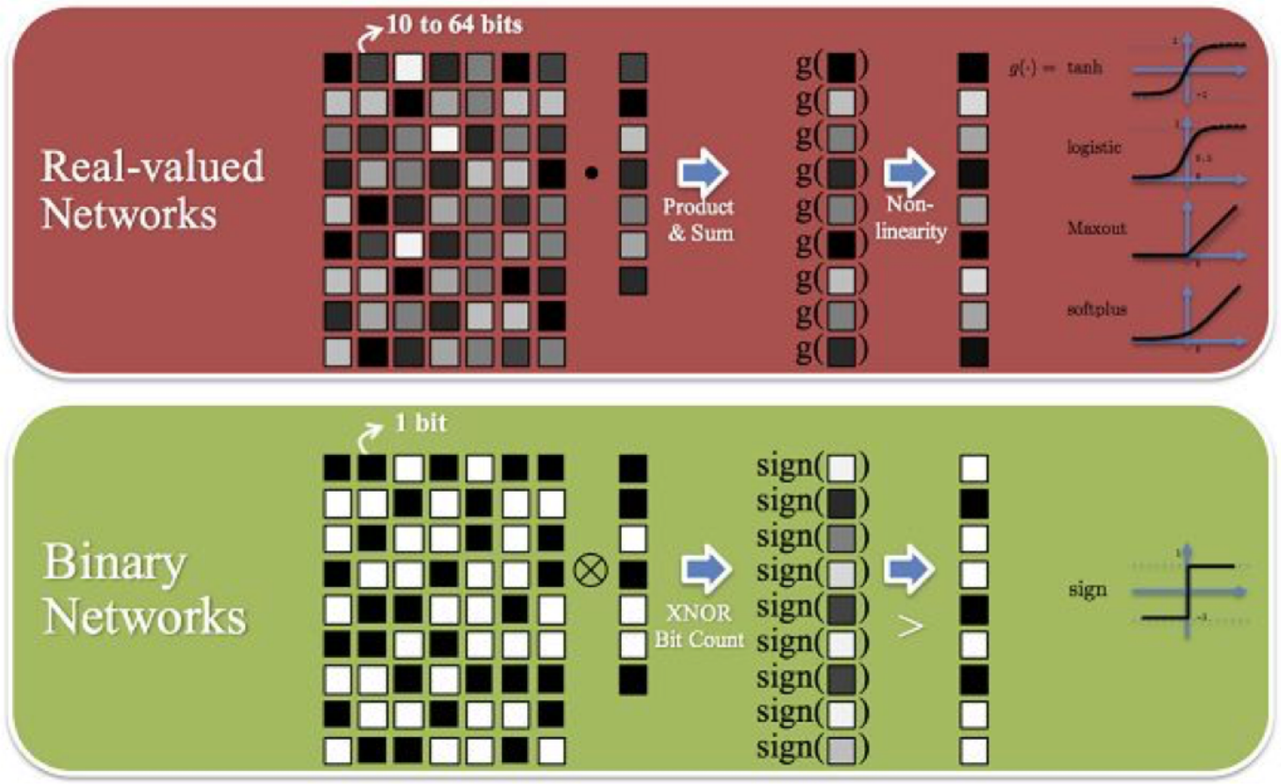
75ns latency with new input every 5ns
Fits a conventional UltraScale FPGA

Binary/Ternary Networks

New

Concept: Replace floating/fixed-point with 1/2-bit arithmetic:

- Binary 1-bit (arXiv:1602.02830)
- Ternary 2-bit (arXiv:1605.04711)
- Binary/ternary in both dense and activation layers

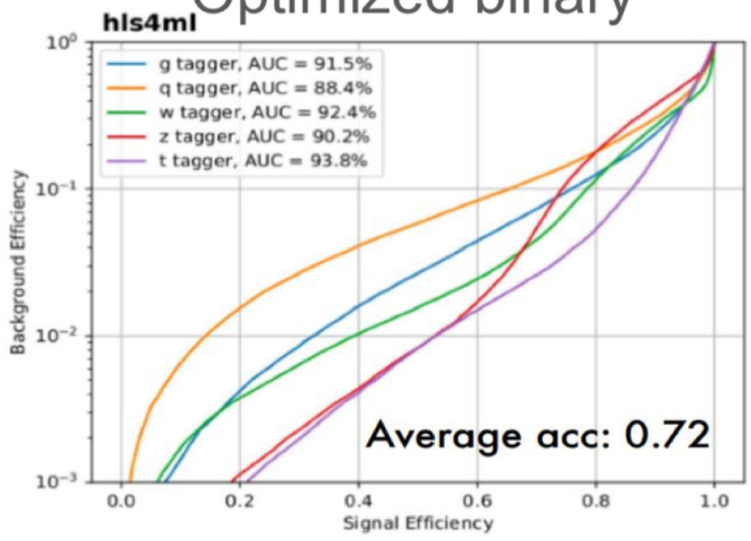


For the Basline model we Recover performance with larger models

- Binary: **16x448x224x224x5** (7x more neurons)
- Ternary: **16x128x64x64x64x5** (2x more neurons + one more layer)

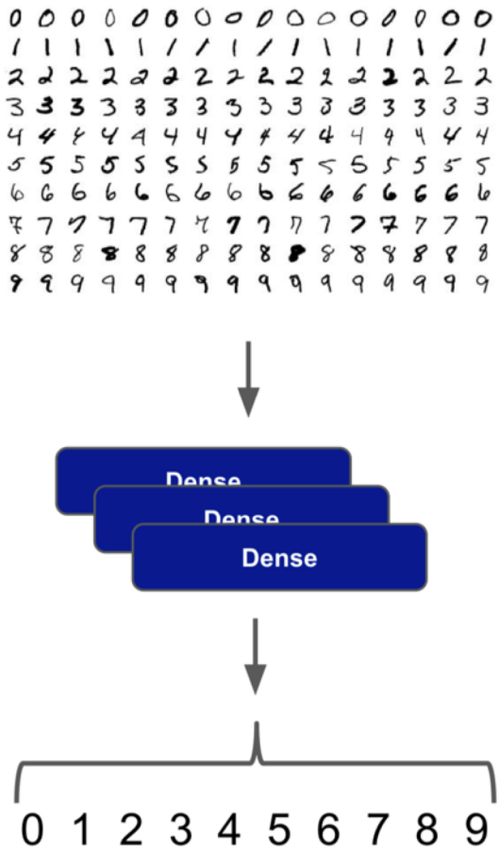
Model	Accuracy	Latency	DSP	BRAM	FF	LUT
Base model	0.75	0.06 *μs*	60%	0%	1%	7%
Optimized Binary	0.72	0.21 *μs*	0%	0%	7%	15%
Optimized Ternary	0.72	0.11 *μs*	0%	0%	1%	6%

Optimized binary



Binary/Ternary with MNIST

Can we do bigger Networks for Co-Processing applications?



Dense networks trained with the MNIST dataset
- 784 inputs (28x28 grayscale image), 10 outputs (digits)

Base model:
- 3 hidden layers with 128 neurons and ReLU activation

Binary/Ternary model:
- 3 hidden layers with batch normalization and binary/ternary tanh

Xilinx VU9P FPGA at 200 MHz, reuse factor 128

Model	Accuracy	Latency	DSP	BRAM	FF	LUT
Dense model	0.97	2.6 *μs*	21%	45%	12%	33%
Binary dense model	0.93	2.6 *μs*	0%	33%	7%	39%
Ternary dense model	0.95	2.6 *μs*	0%	33%	7%	40%

What is coming?

More ML architectures:

- Large Convolutional Layers in 1D and 2D
  - interest in Conv3D?
- BDTs
- GRAPHs
- Recursive NN
- Autoencoders

The toolkit is quickly expanding:

Multi-FPGA Inference

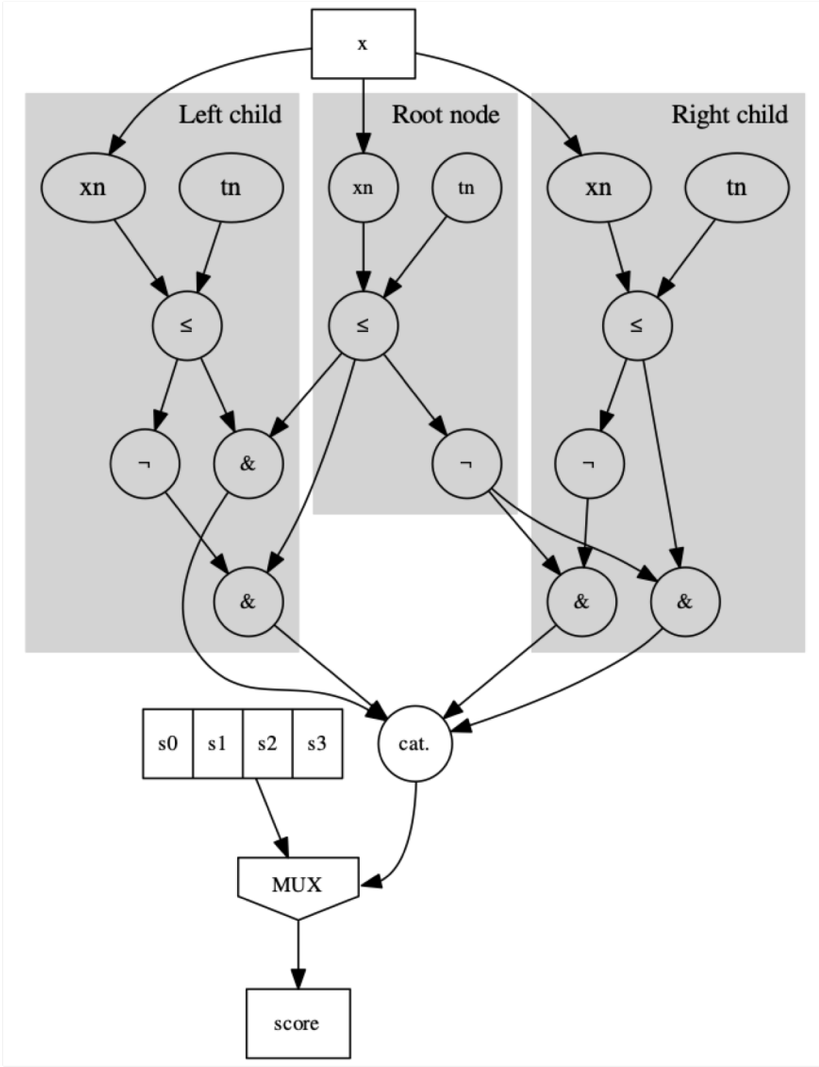
- Place layers in multiple FPGAs and pipeline execution

Beyond Xilinx

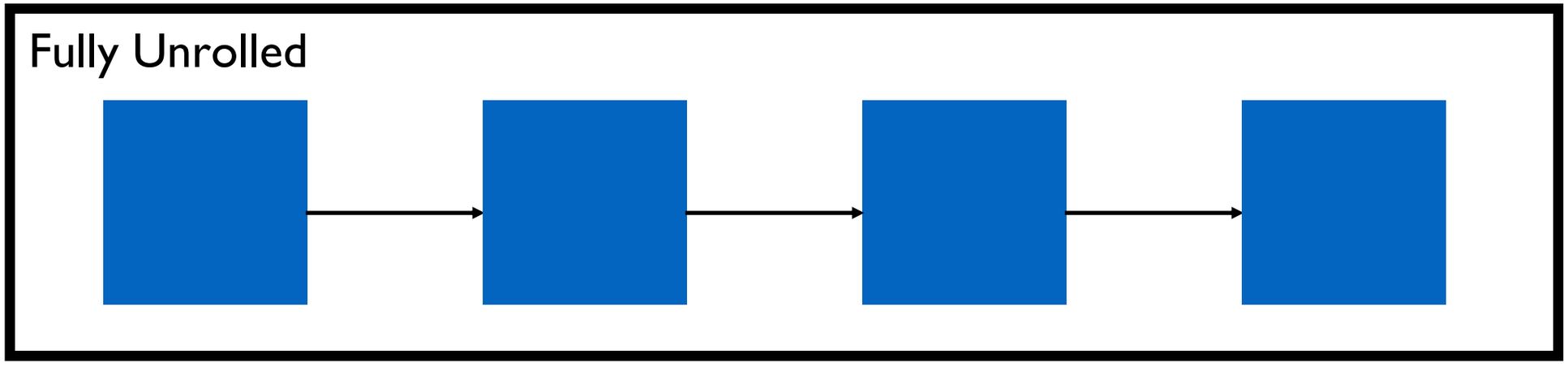
- Quartus HLS for Intel/Altera
- Mentor Catapult HLS

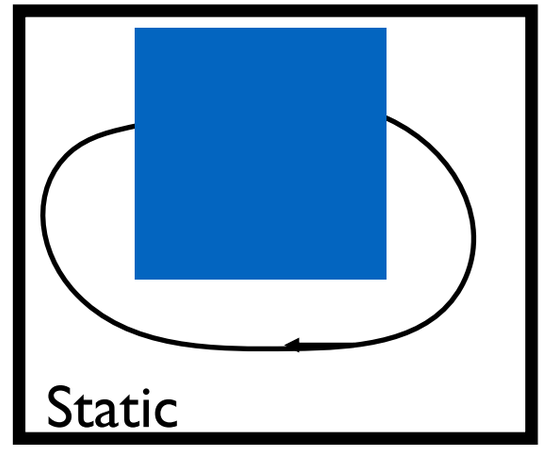
Boosted Decision Trees

- BDTs have been popular for a long time in HEP reconstruction and analysis
- Suitable for highly parallel implementation in FPGAs
- No ‘if/else’ statement in FPGAs → evaluate all options and select the right outcome
  - compare all features against thresholds, chain together outcomes to make the ‘tree’
- Benchmark model with 16 inputs, 5 classes, 100 trees, depth 3 on VU9P FPGA:
  - 4% LUTs, 1% FFs (0 DSPs, 0 BRAMs)
  - 25 ns latency with II=1



RNN





- Two implementations are of RNN available
  - Fully unrolled: each recursion goes on FPGA

    Yields latency optimized with II=1 possible
  - Static: same resources used for weights and multiplications

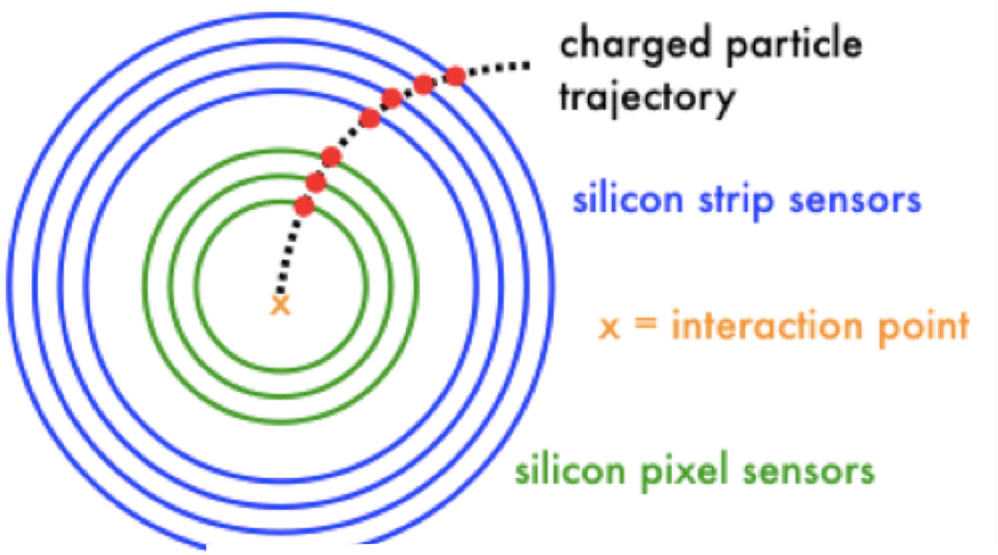
    Latency is slower and II limited to clock time for each layer (small network its 10 clocks)

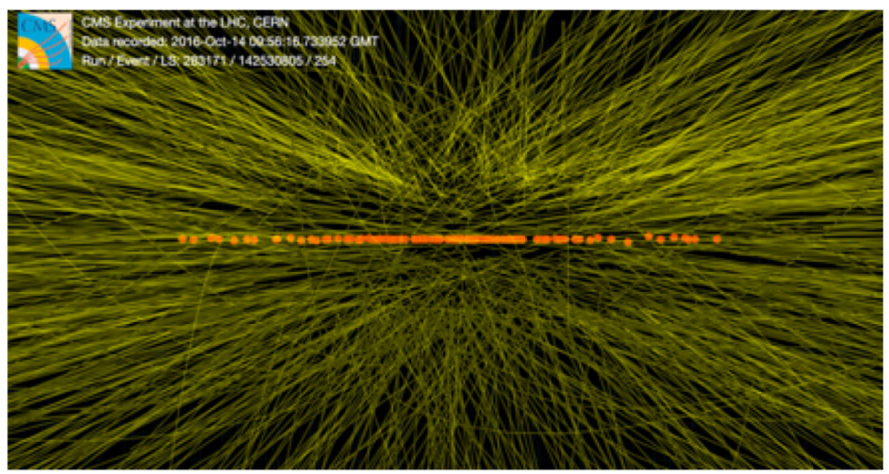
    However N (N=latency of layer) copies can go through at the same time
- Network Implementations available:
  - Simple RNN, GRU, LSTM
- Status
  - Code needs to be merged with large layer extension of HLS4ML to allow for full flexibility

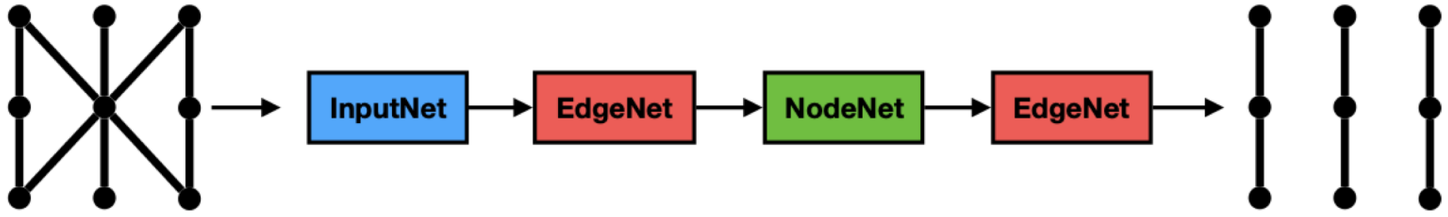
Graphs

based on HEP.TrkX GNN v1 architecture [arXiv:1810.06111]

This real collision: thousands of tracks!







Preliminary implementation in hls4ml under test:

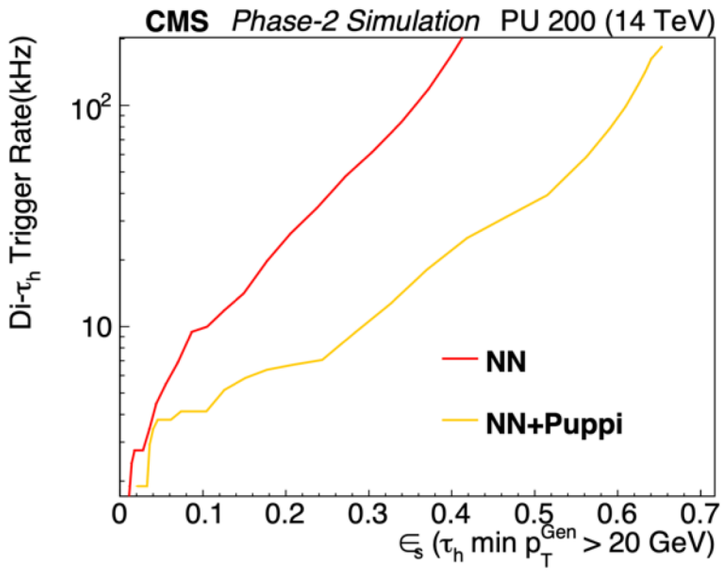
- use default MLP and activation function implementations but applied to each row of the input matrix
- develop new functions to do concatenations and special (binary/sparse) matrix multiplications for edge-node association matrices

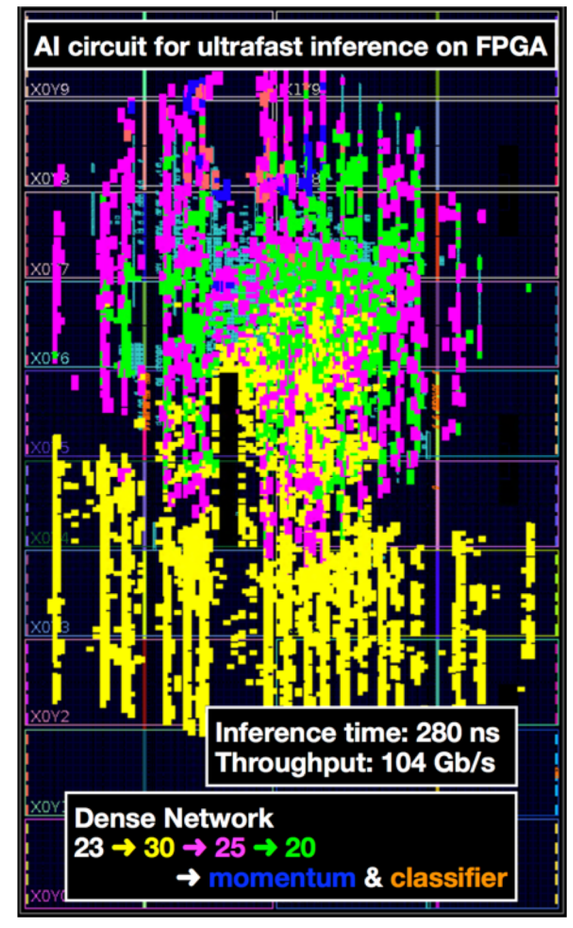
Successfully tested a small example with 4 tracks, 4 layers, no iteration → major effort now to scale this up

To be automatized in hls4ml keras-to-hls conversion tool with custom model

Example Level-1 use cases

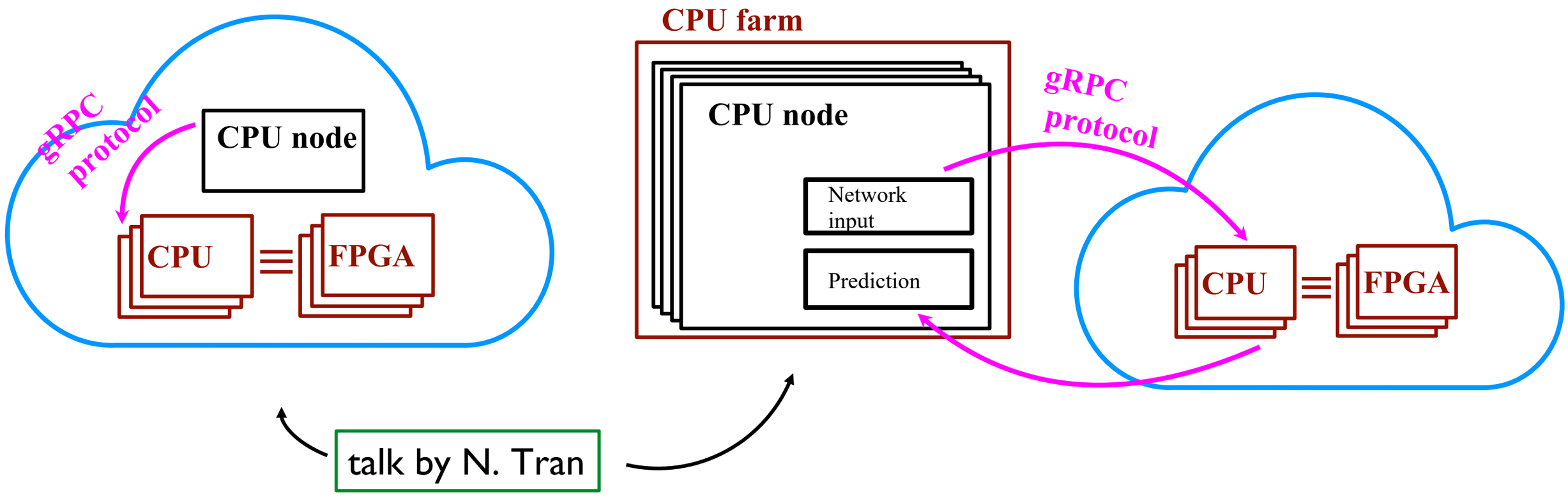
- Prompt and Displaced Muons with DNN — talk by JF. Low
- Calorimeter cluster classification or energy calibration
- Neural Net based Tau identification — talk by C.Herwig
- Anomaly detection with Autoencoders — talk by Z.Wu
- Jet substructure/tagging in L1





What about HLT/Offline?

- Previous systems have been CPU only
  - New systems will likely be heterogeneous (FPGAs/GPUs…)
- timescales at the level of milliseconds or seconds
- hls4ml facilitates ways to accelerate ML algorithms using CPU+FPGA co-processor systems (ex, Galapagos/SDAccel)



Beyond the LHC

- This effort has started to address challenge of data reconstruction at the LHC

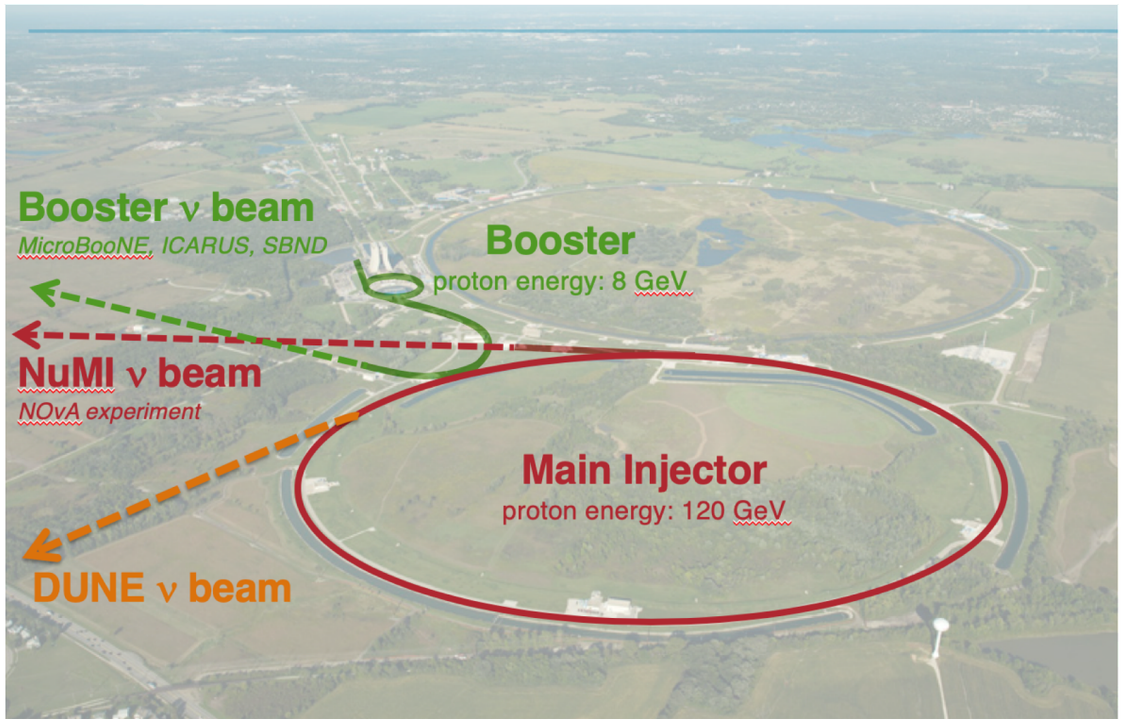
Fast inference of deep neural networks in FPGAs for particle physics

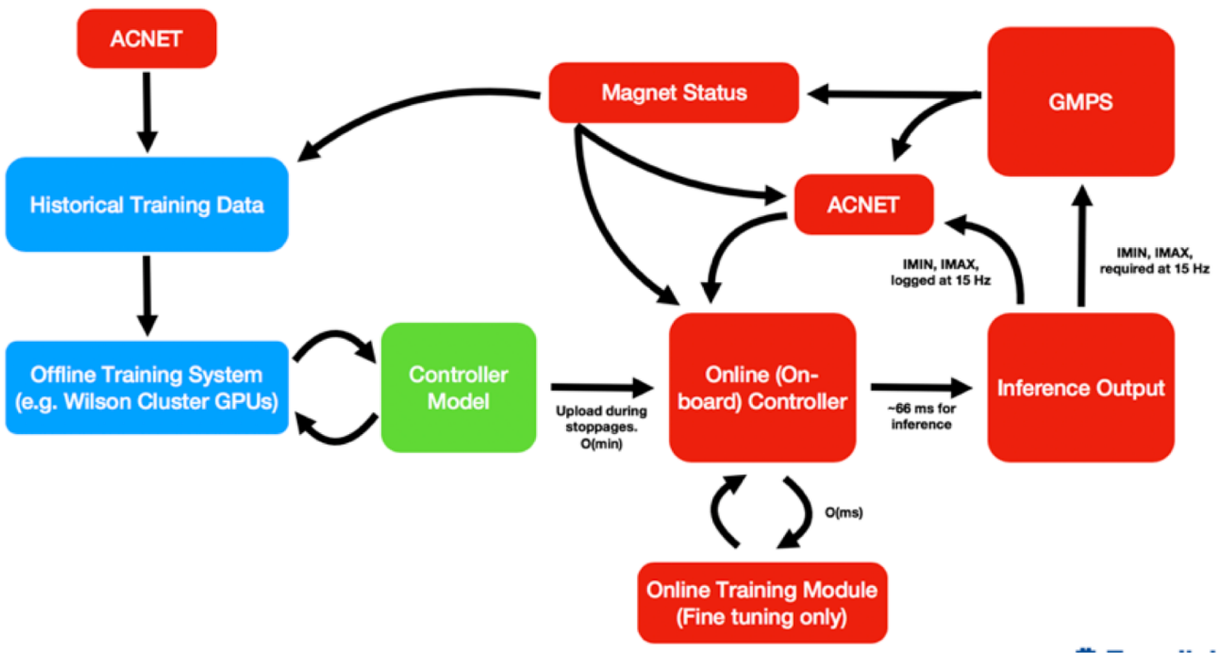
J. Duarte,[a] S. Han,[b] P. Harris,[b] S. Jindariani,[a] E. Kreinar,[c] B. Kreis,[a] J. Ngadiuba,[d] M. Pierini,[d] R. Rivera,[a] N. Tran[a,1] and Z. Wu[e]

[a] *Fermi National Accelerator Laboratory, Batavia, IL 60510, U.S.A.*
[b] *Massachusetts Institute of Technology, Cambridge, MA 02139, U.S.A.*
[c] *HawkEye360, Herndon, VA 20170, U.S.A.*
[d] *CERN, CH-1211 Geneva 23, Switzerland*
[e] *University of Illinois at Chicago, Chicago, IL 60607, U.S.A.*

- Since then, we are quickly identifying other cases with the same issues
  - Neutrino Event Reconstruction
  - Fixed Target Experiments
  - Observational Cosmology
  - GW detection
  - Accelerators
- Have extended our collaboration to incorporate everybody
  - Inaugural workshop can be found here https://indico.cern.ch/event/822126
  - You too can join our Fast Machine Learning effort

Accelerator Controls





- Goal to reduce beam losses in Booster

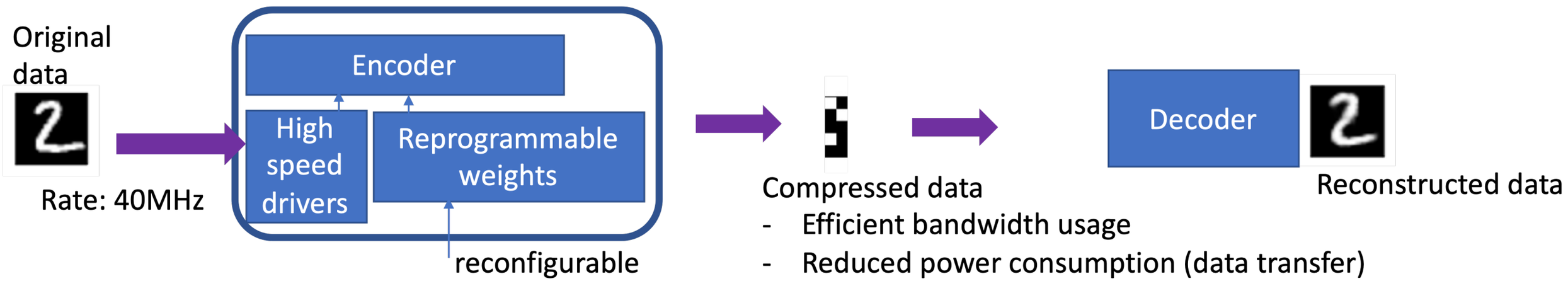
- Solve problem using ML on a custom FPGA board to control the magnet power supplies (GMPS) — deploy the hls4ml tool

- Single crate control system; project lays the foundation for a more ambitious future program.

hls4ml in 4ASIC

Hardware acceleration with an emphasis on co-design and fast turnaround time

First project: Autoencoder with MNIST benchmark (28 x 28 x 8-bits @ 40 MHz)



Enable edge compute : e.g. data compression
Programmable and Reconfigurable: reprogrammable weights
Hardware – Software codesign: algorithm-driven architectural approach
Optimized Mixed signal / Analog techniques: Low power and low latency for extreme environment (ionizing radiation, deep cryogenic)

First tests of 1-layer design
Latency: 9ns
Power (FPGA, 28nm) ~ 2.5 W
Power (ASIC, 65nm) ~ 40 mW
Area = 0.5mm x 0.5mm

Summary

- Hls4ml- a bridge from neural network tools to synthesizable FPGA firmware and more
  - Many architectures and features supported
- Already several use cases being pursued at the LHC
  - Expanding beyond the LHC
  - And beyond FPGAs
  - While maintaining strong connections with industry
- Many more features to come next year – stay tuned!
  - https://www.fastmachinelearning.org
  - https://arxiv.org/abs/1804.06913

Extras.

A quick how-to

- Easy to **install** via `pip`: `git clone … && cd hls4ml && pip install .`
- Easy to **configure** through yaml config file

  `Inputs:` your trained model
  `Precision:` inputs, weights, biases, …
  `ReuseFactor:` how much to parallelize
  `Strategy:`
  `Resource` for large NN
  `Latency` for pipelined-based code for small NN

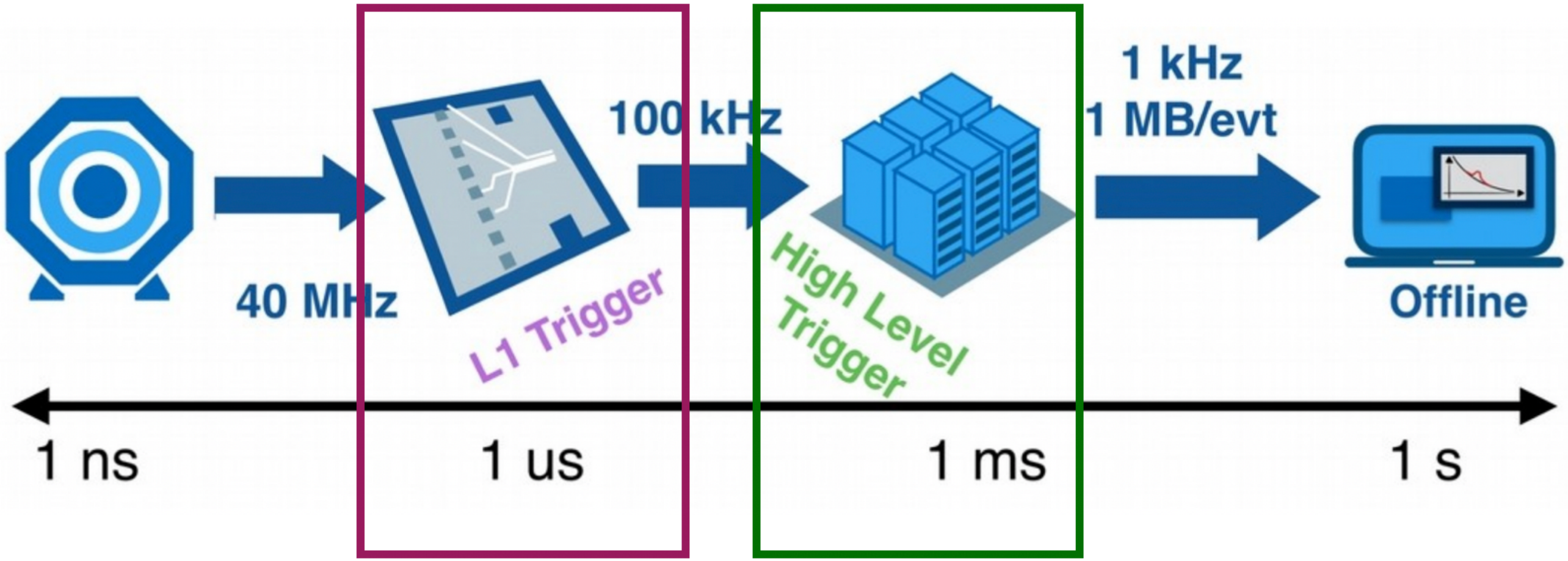
```
KerasJson: keras/KERAS_3layer.json
KerasH5:   keras/KERAS_3layer_weights.h5
OutputDir: my-hls-test
ProjectName: myproject
XilinxPart: xcku115-flvb2104-2-i
ClockPeriod: 5

HLSConfig:
  Model:
    Precision: ap_fixed<16,6>
    ReuseFactor: 1
    Strategy: Latency #Resource
  LayerName:
    dense1:
      ReuseFactor: 2
      Strategy: Latency #Resource
      Compression: True
```

keras-config.yml

- Easy to **run**:

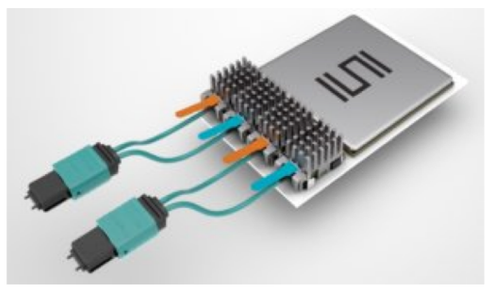
  Conversion: `hls4ml convert -c keras-config.yml`
  Build: `hls4ml build -p my-hls-test -c -s -r`
  Help: `hls4ml -h / hls4ml command -h`

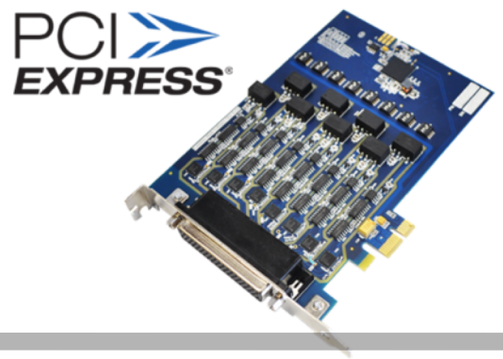
Data Processing



Optical to electrical to fpga pins

Stream through PCIexpress





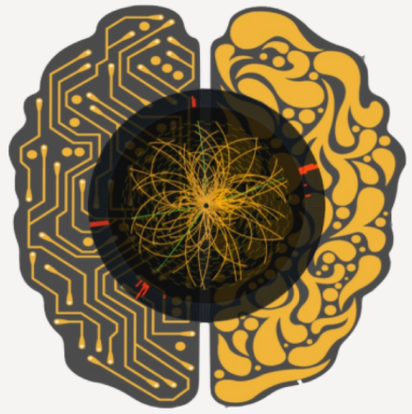
<1μs

1ms-1s

The Team

MEET THE COLLABORATORS

(click on name for more info)



CERN
Vladimir Loncar (PhD, Computer Science); Jennifer Ngadiuba (PhD, Physics); Maurizio Pierini (PhD, Physics); Sioni Summers (PhD, Physics);

Columbia University
Giuseppe Di Guglielmo (PhD, Computer Science)

Fermilab
Christian Herwig (PhD,Physics); Burt Holzman (PhD,Physics); Sergo Jindariani (PhD,Physics); Thomas Klijnsma (PhD,Physics); Ben Kreis (PhD,Physics); Mia Liu (PhD,Physics); Kevin Pedro (PhD,Physics); Ryan Rivera (PhD,EE); Nhan Tran (PhD,Physics)

Hawkeye 360
EJ Kreinar (Computer Science)

MIT
Jack Dinsmore (Undergraduate, Physics); Song Han (PhD, EECS); Phil Harris (PhD, Physics); Sang Eon Park (Graduate, Physics); Dylan Rankin (PhD, Physics);

UC San Diego
Javier Duarte: PhD, Physics, Caltech

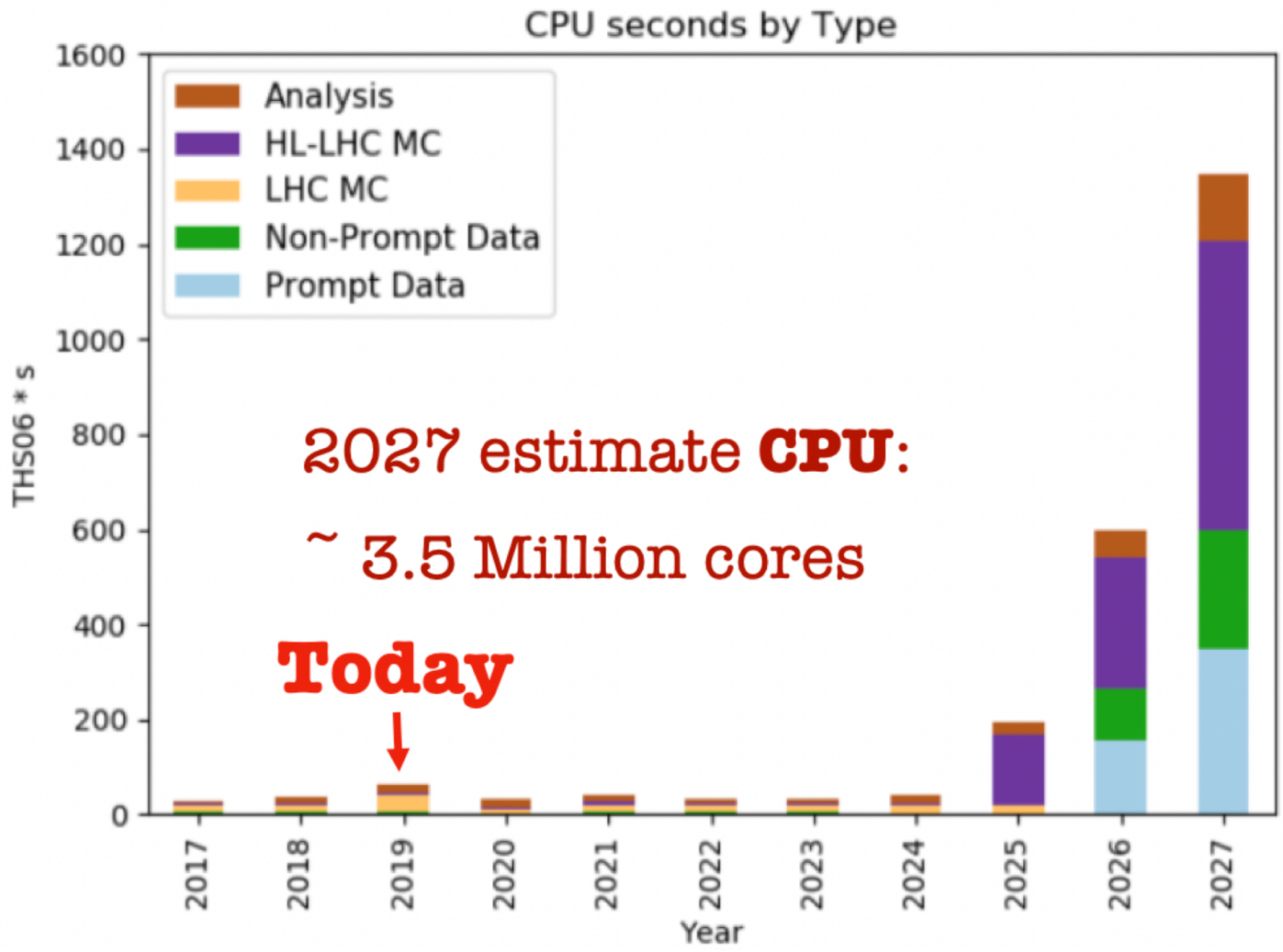
University of Illinois Chicago
Zhenbin Wu (PhD, Physics);

University of Illinois Urbana-Champaign
Markus Atkinson (PhD, Physics); Mark Neubauer (PhD, Physics);

University of Washington
Scott Hauck (PhD, EECS); Shih-Chieh Hsu (PhD, Physics);

Challenges for HLT/Offline

10x larger events * 5x the rate * 10 years of data-taking



And what if we need to expand the physics program?