Fast Machine Learning at the Edge for HEP Experiments

Sioni Summers

CERN Detector Seminar

8th March 2024

Introduction

- Fast Machine Learning is becoming prevalent at the level of hardware triggering in FPGAs and even at the detector frontend
- Data deluge from increasingly granular detectors, and searches for rare phenomena require precise and fast selections
- In this presentation I'll present:
- Tools: **hls4ml** and **conifer**
- Techniques for high performance
  - Quantization aware training, pruning, hardware-aware training
- Applications
  - Developments using **hls4ml** and **conifer**
  - Many from CMS, plus some others
  - Some non-HEP use cases
  - Roughly in direction of furthest → closest to detector

About me

- PhD High Energy Physics Imperial College London
  - Thesis: "Applications of FPGAs to triggering in particle physics"
  - Designing physics algorithms with high level languages for FPGAs
- Applied Physicist Staff at CERN (previously Senior Fellow) working on Level 1 Trigger Upgrade for CMS experiment, EP-CMG-OS group
  - Mostly designing and implementing detector reconstruction algorithms for Level 1 Trigger
  - Track reconstruction, vertexing, particle flow, jets
- Also deploying Machine Learning into FPGAs for low latency
  - **hls4ml** coordinator 2020-2022, creator and maintainer of **conifer**
  - Developing and deploying ML algorithms to the trigger

sioni@cern.ch
sioni.web.cern.ch
@thesps
@ssummers

The challenge: triggering at LHC

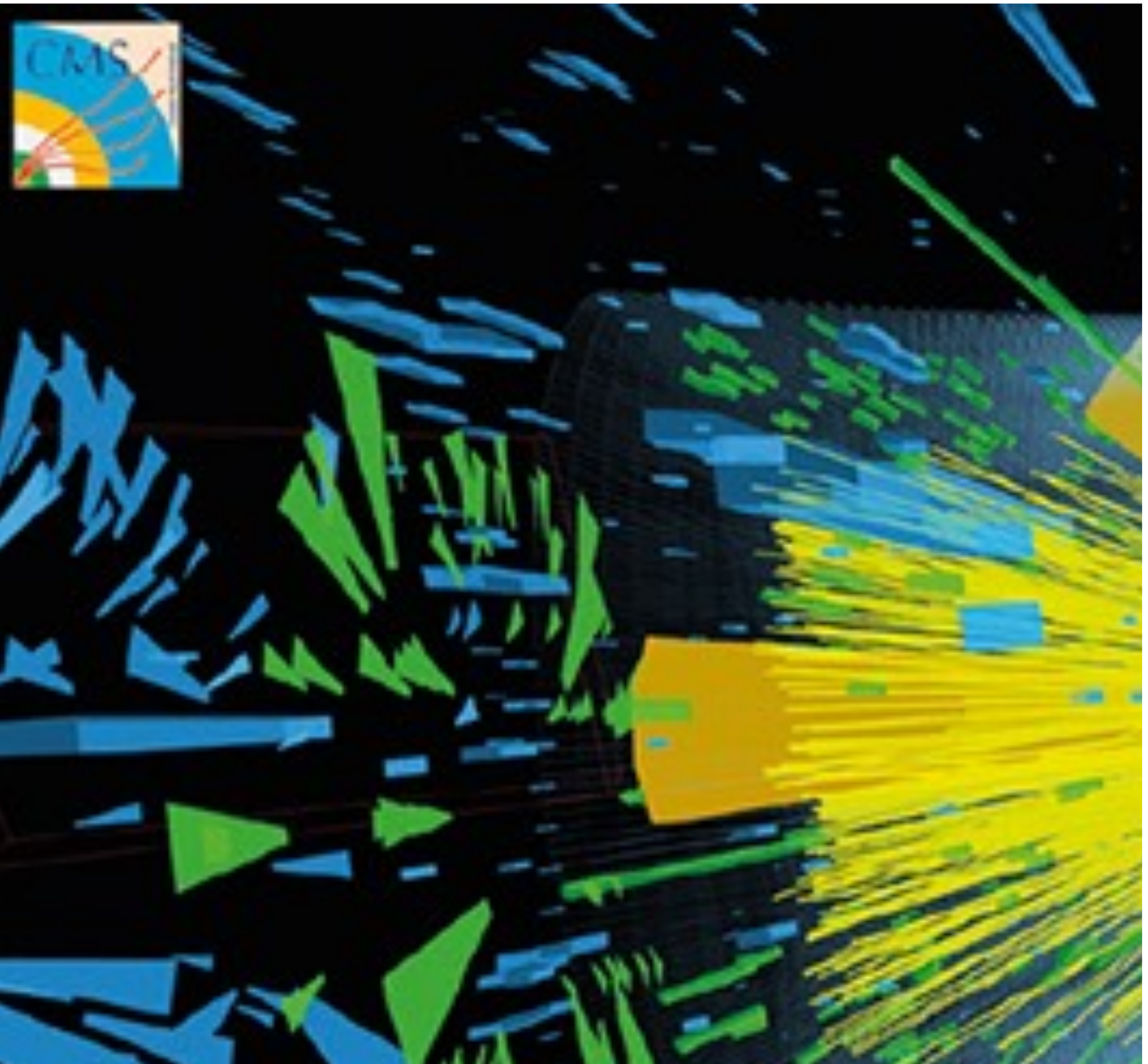
At LHC protons collide at 40 MHz → extreme data rates O(100 Tb/s)
Most collisions don't produce exciting new particles
"Triggering" = filtering events to reduce data rates to manageable levels

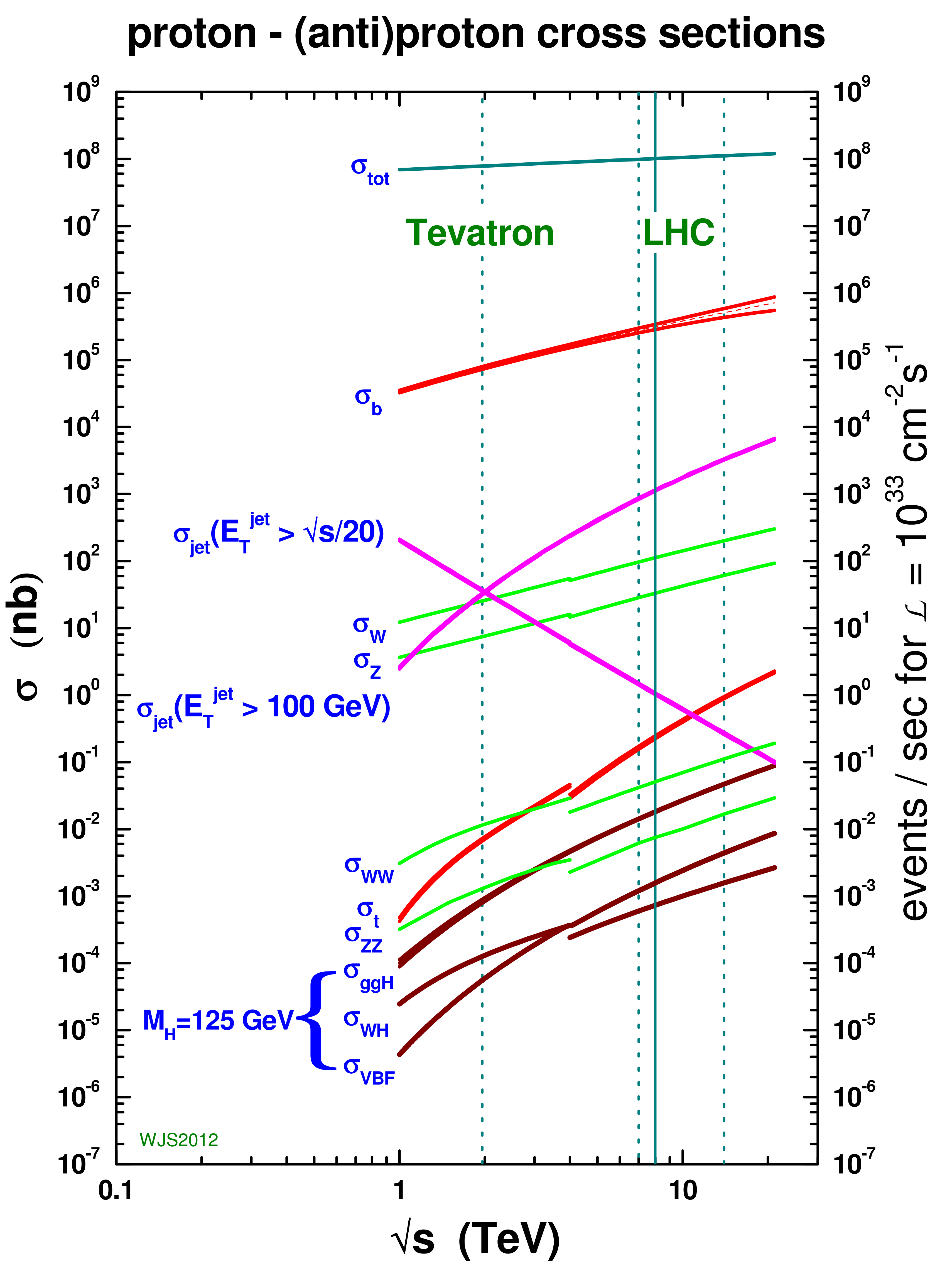
The challenge: triggering at LHC

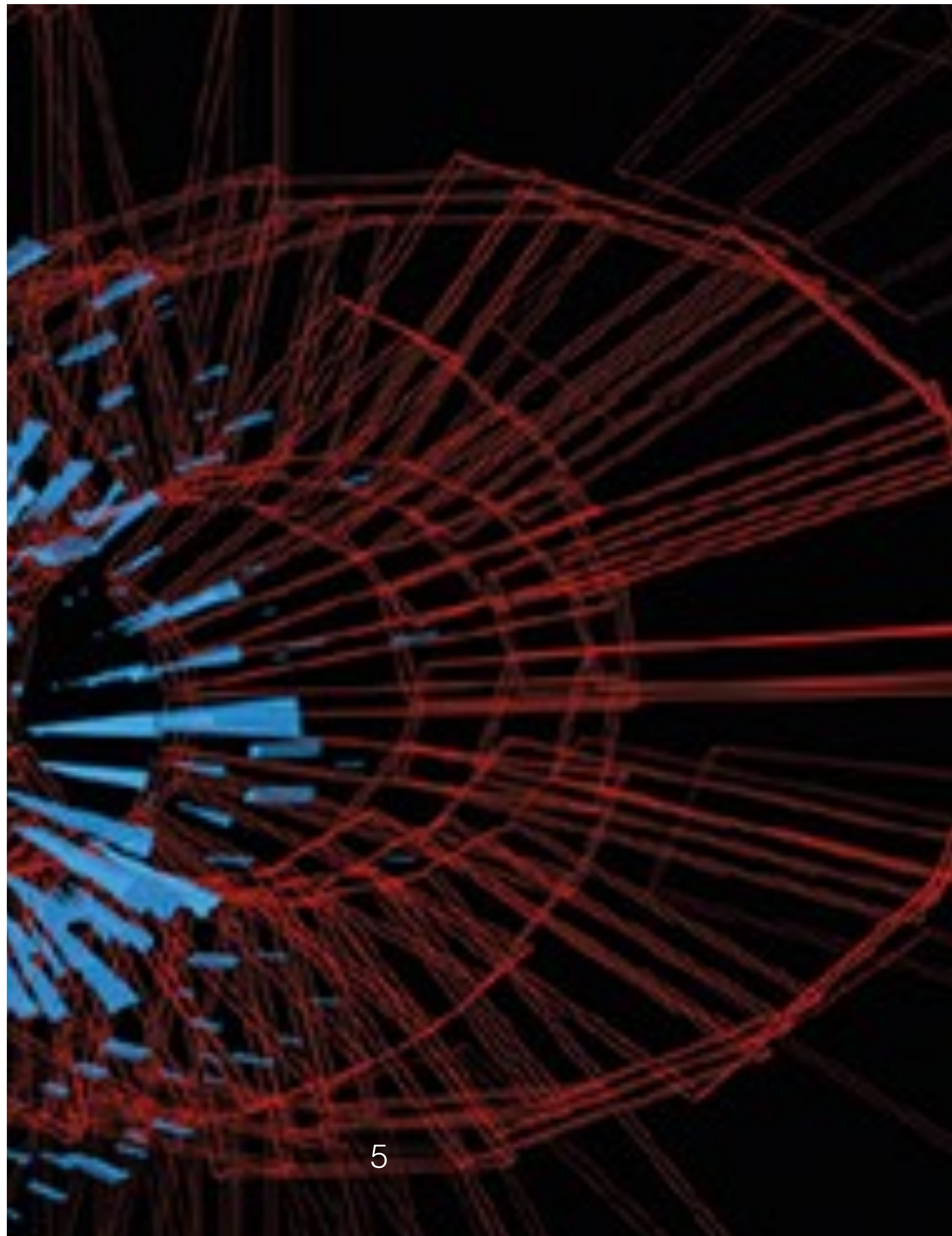
At LHC protons collid … **0 Tb/s)**

Most collisior

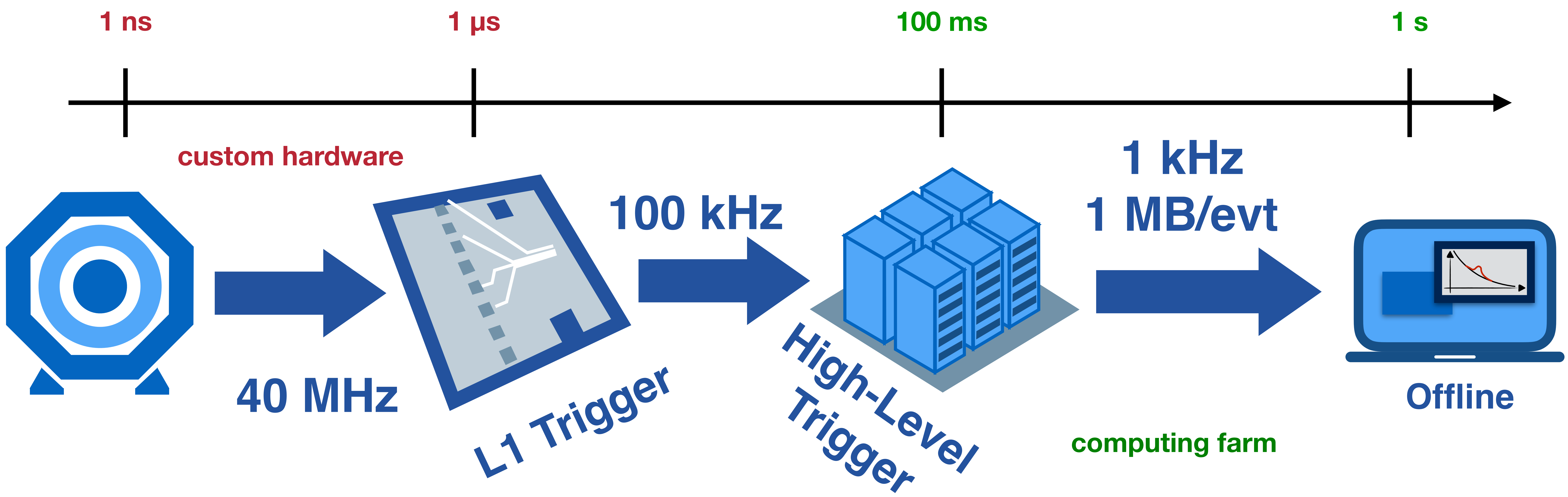
“Triggering” = filtering e … **ble levels**







Trigger at LHC



Triggering performed in multiple stages @ ATLAS and CMS

Reduce data rate in stages

Process 100s Tb/s

Trigger decision to be made in latency O(µs)

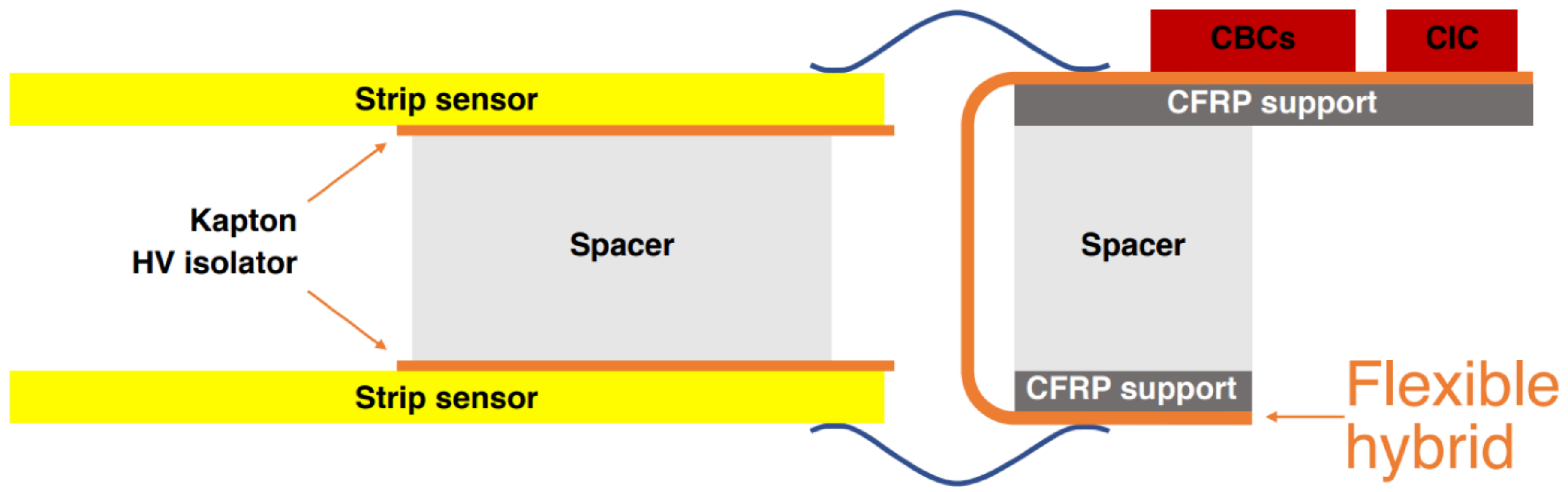
Frontends in rad. hard ASICs, processing in FPGAs

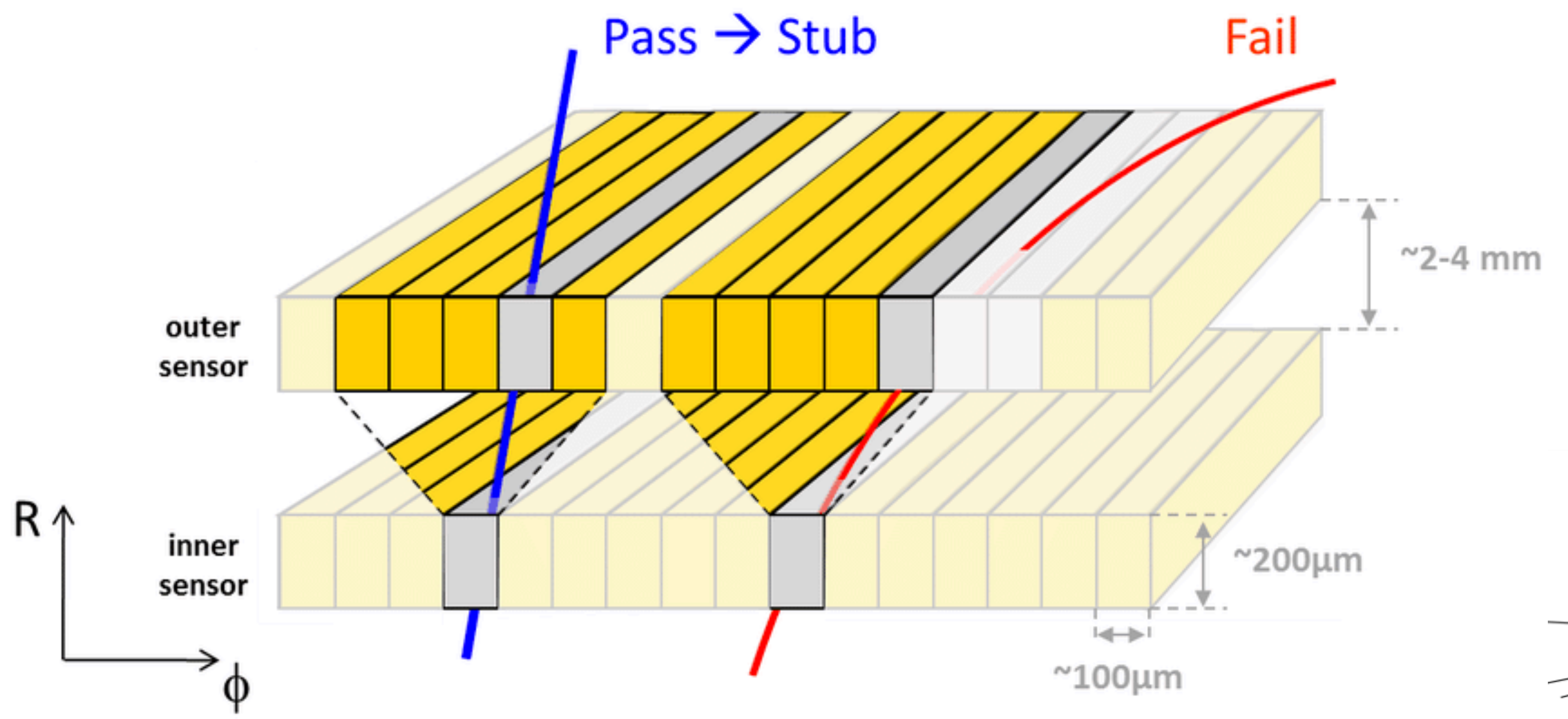
Computing farm for detailed analysis of the full event

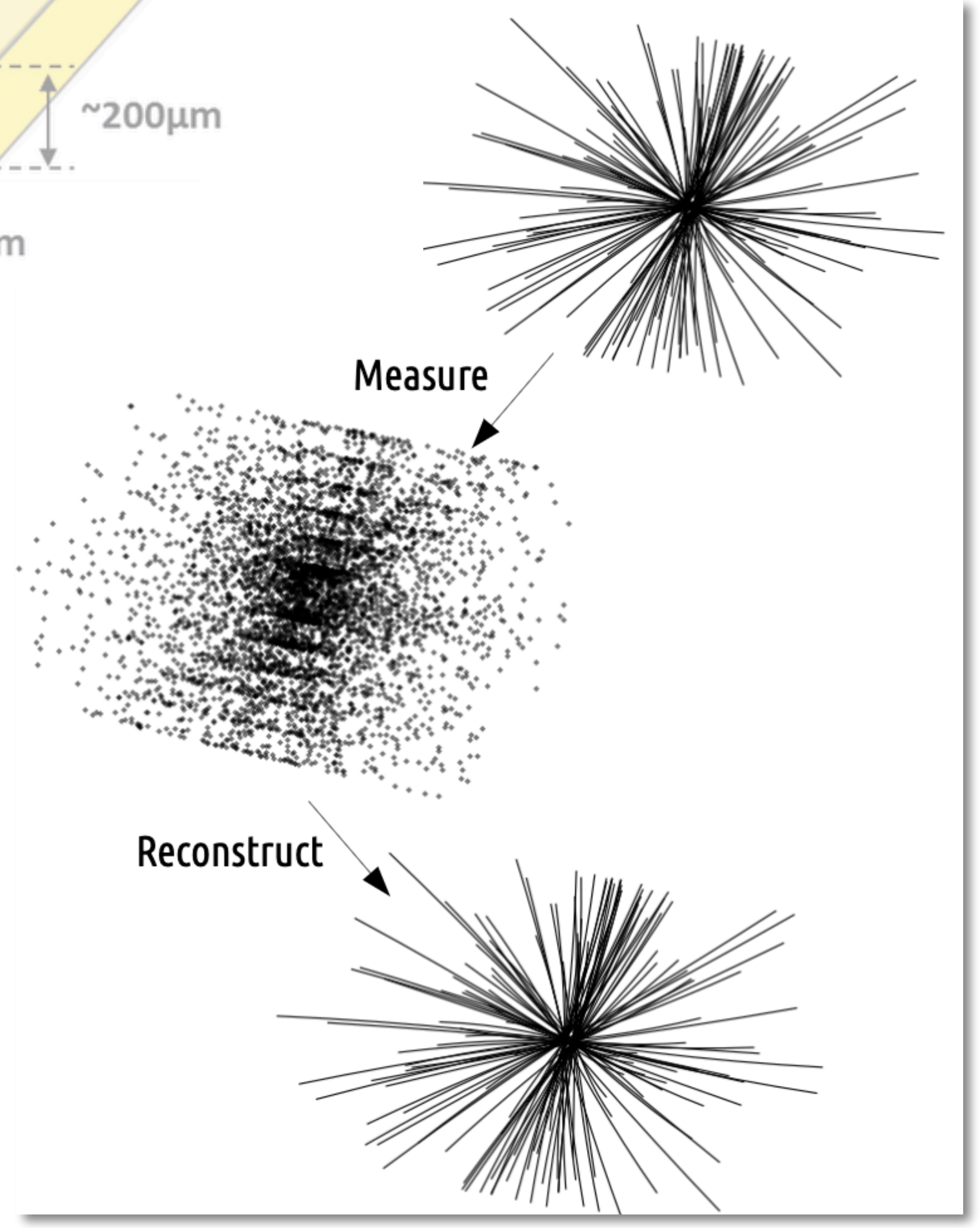
Latency O(100 ms)

CMS Detector Upgrade 1: Tracker

- At CMS the Phase 2 Upgrade brings data from new detectors to the trigger for the first time
- New Outer Tracker implements on-detector p_T cut
- Two silicon strip sensors separated by few mm
  - Far enough to measure bending of charged particle in B field
  - Close enough to be read out on one device
  - “Stubs” passing 2 GeV p_T cut → Level 1 Trigger
- Level 1 Track Finder system reconstructs tracks from stubs
  - Around 200 FPGAs, with “classical” tracking algorithms: data organisation, seed building, road following, track fitting
- Tracks in the Level 1 Trigger essential for 200 PU conditions
  - Reconstruct primary vertex for suppression of 200 PU background
  - Better measurements of properties of electrons, muons, jets
  - 6.4 Tb/s of reconstructed tracks sent downstream

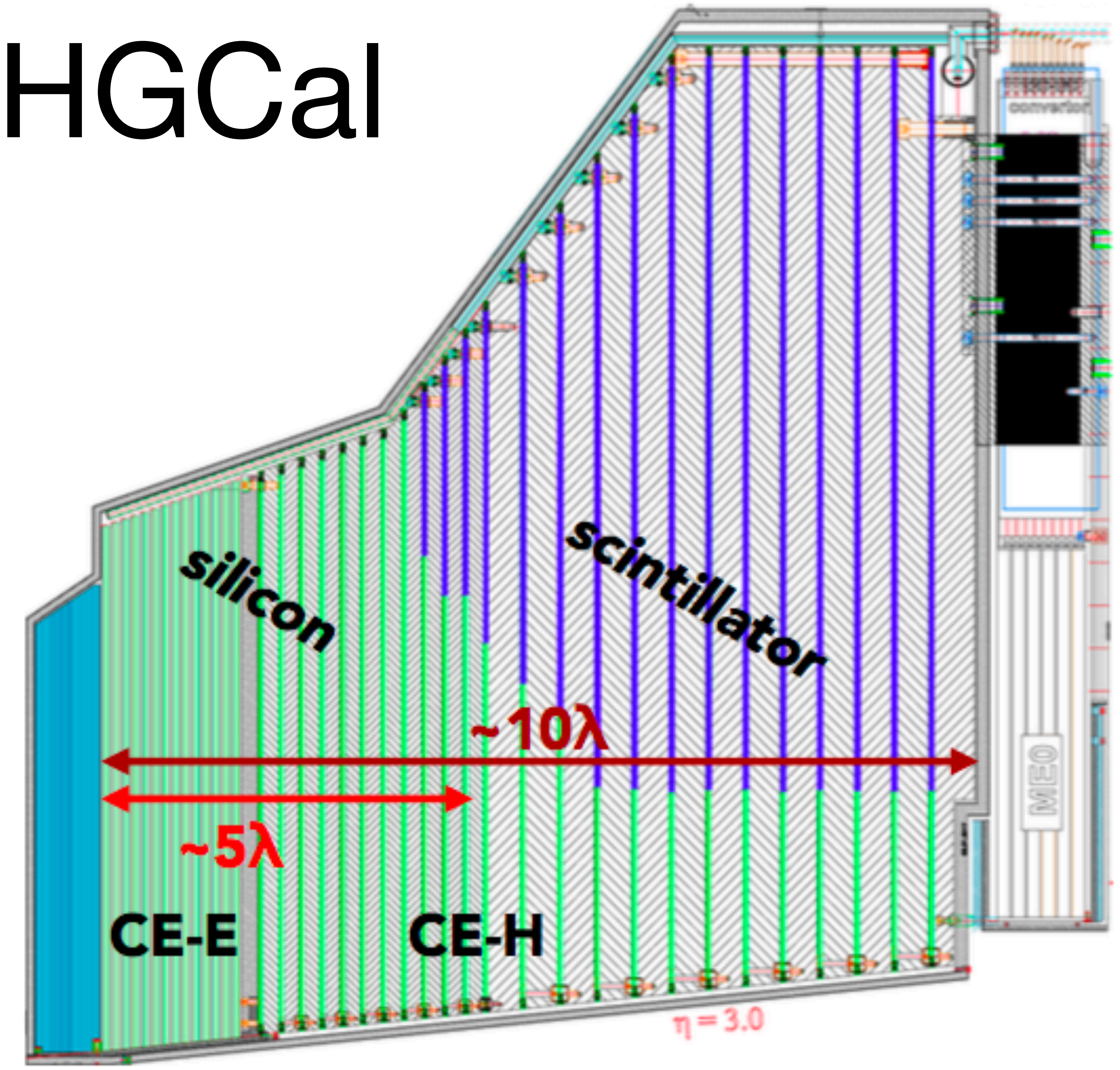




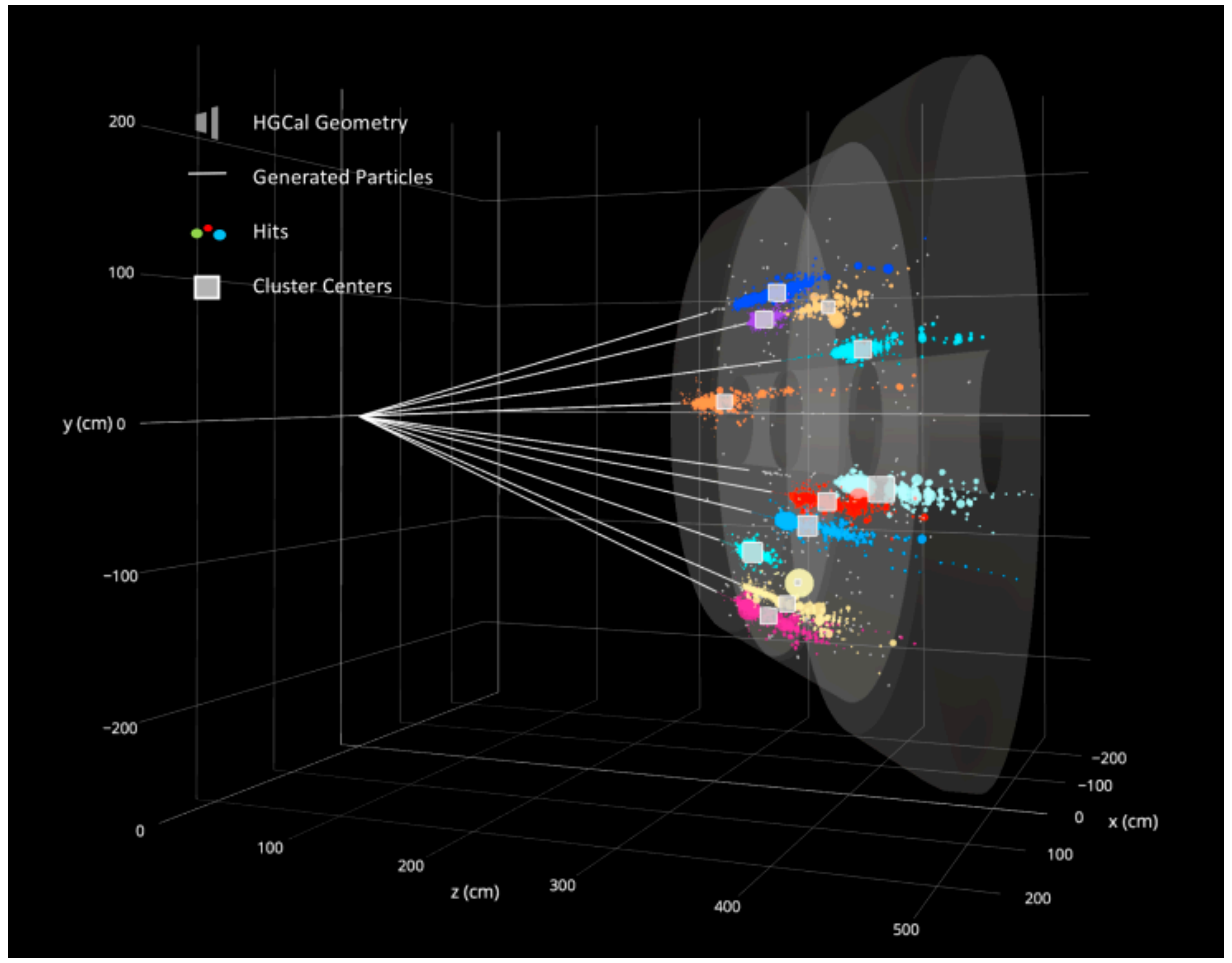


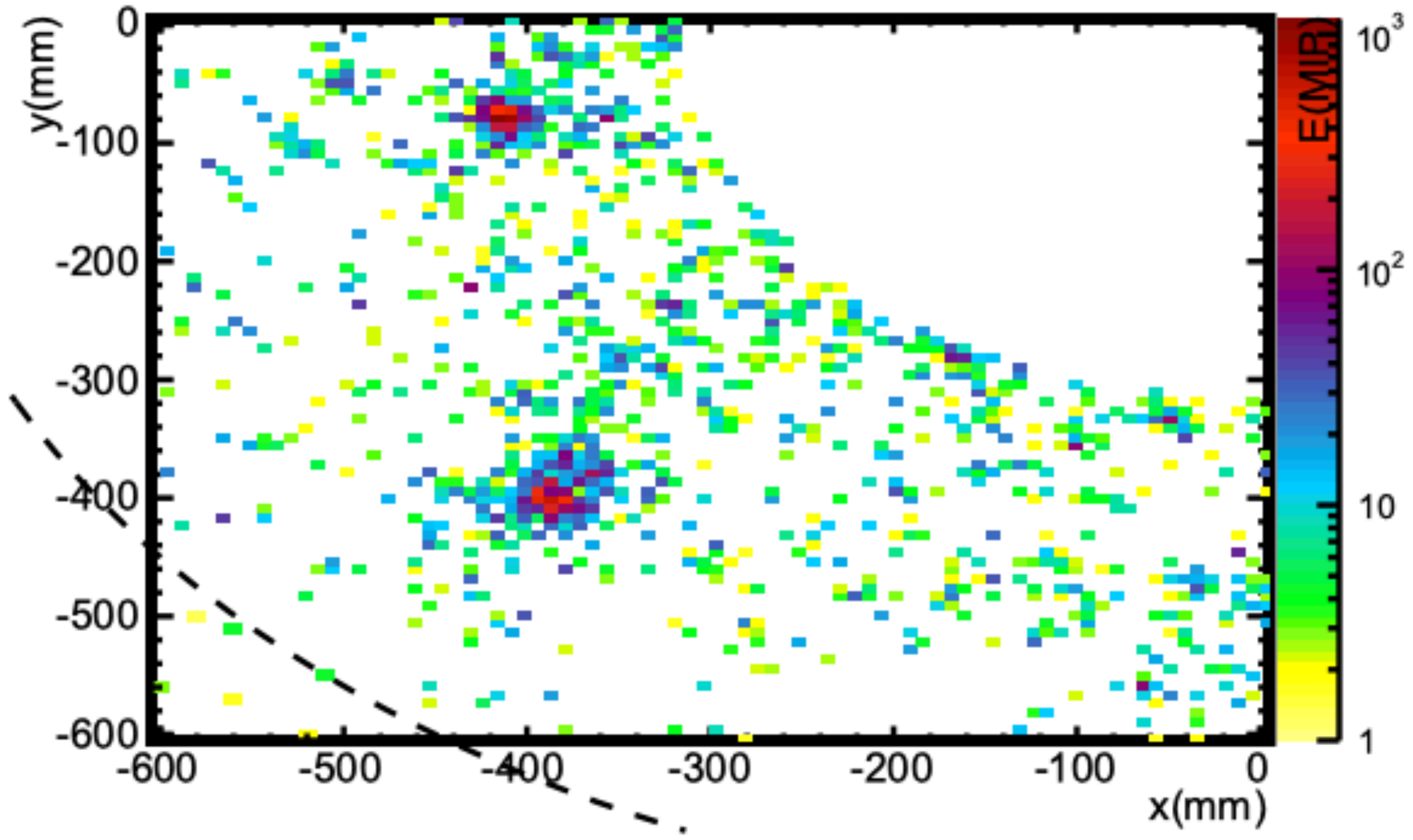
CMS Detector Upgrade 2: HGCal

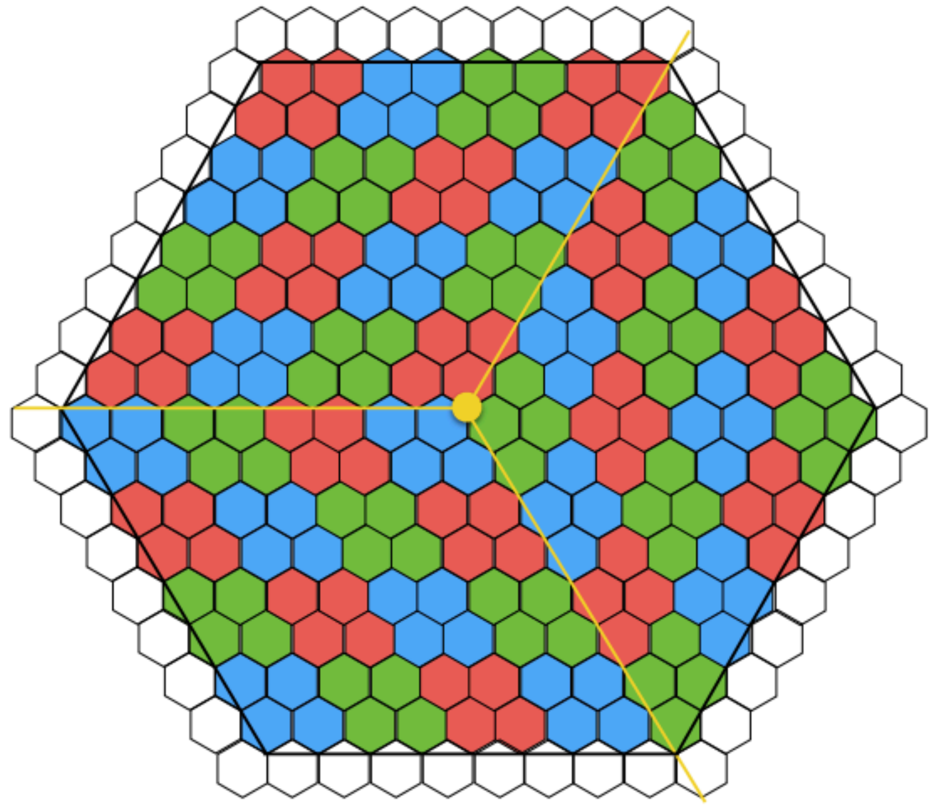
- High granularity calorimeter: silicon sampling calorimeter for the endcaps
- 6.5 million channels in 50 layers
  - Very fine transverse and longitudinal segmentation
- Dedicated ASIC to prepare data for trigger reconstruction - more later
- Trigger backend comprises around 200 FPGAs
  - Reconstructing 3D clusters: 4 Tb/s clusters sent downstream



CMS-TDR-019 **arXiv:1708.08234**

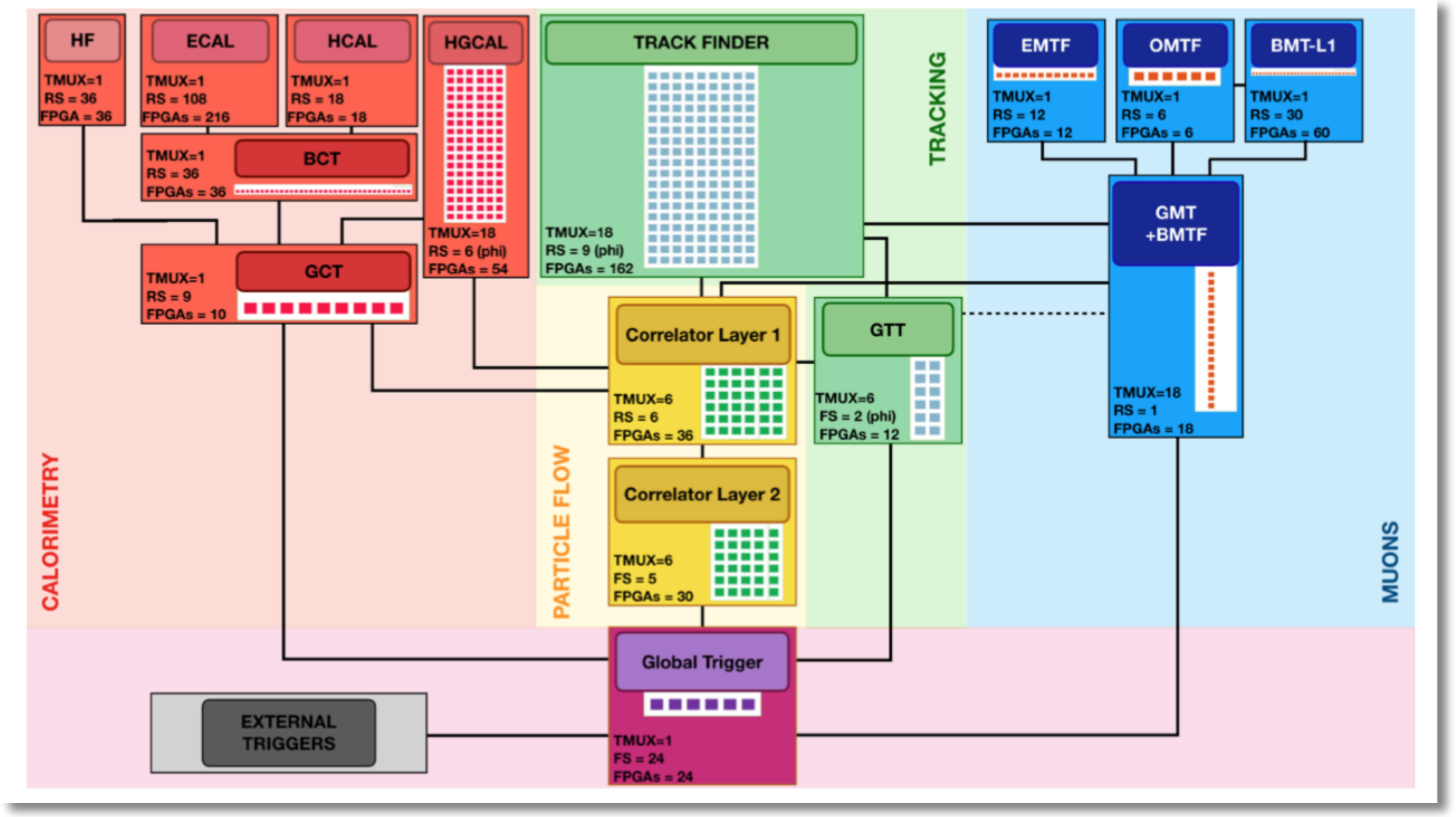


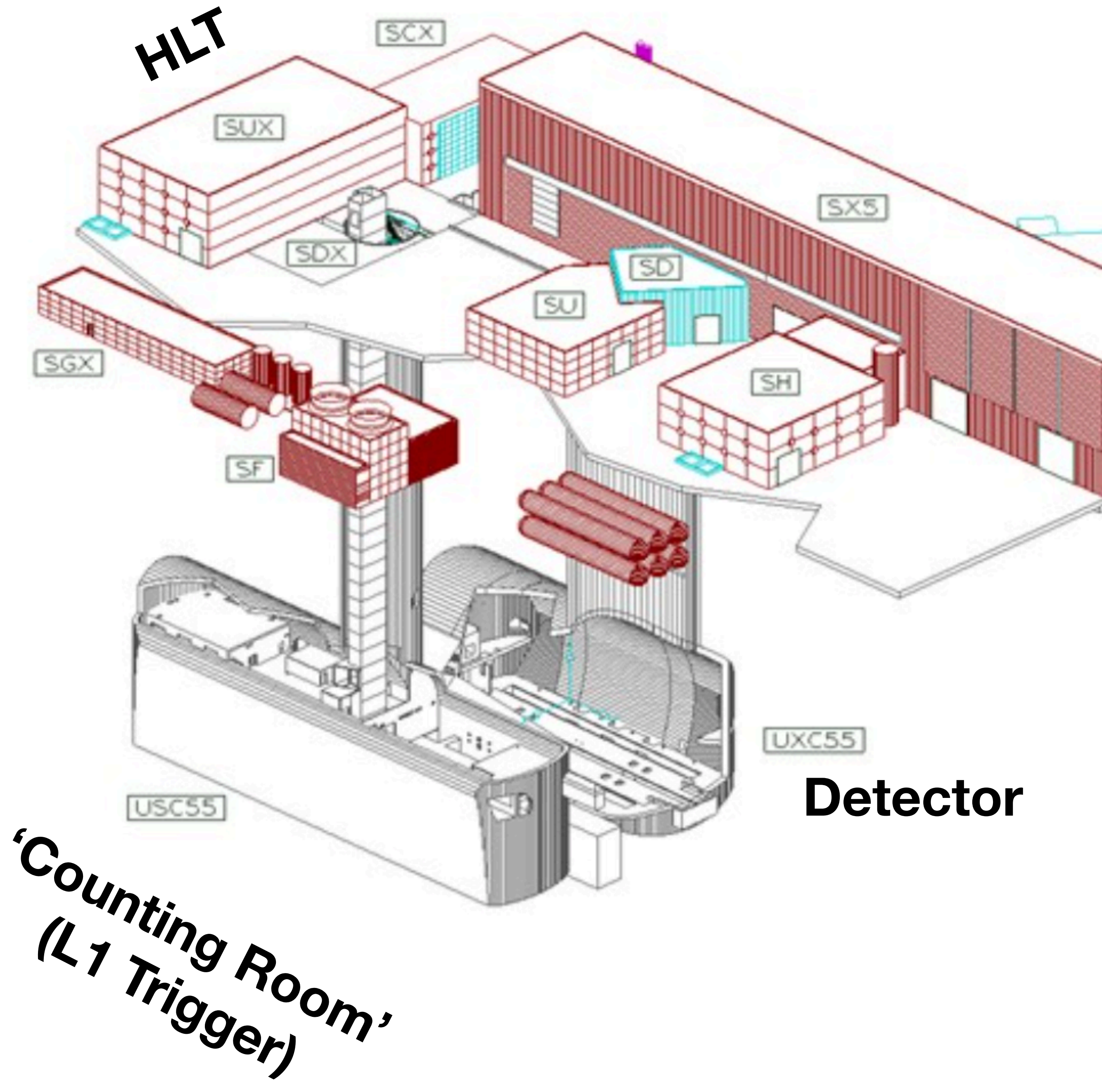




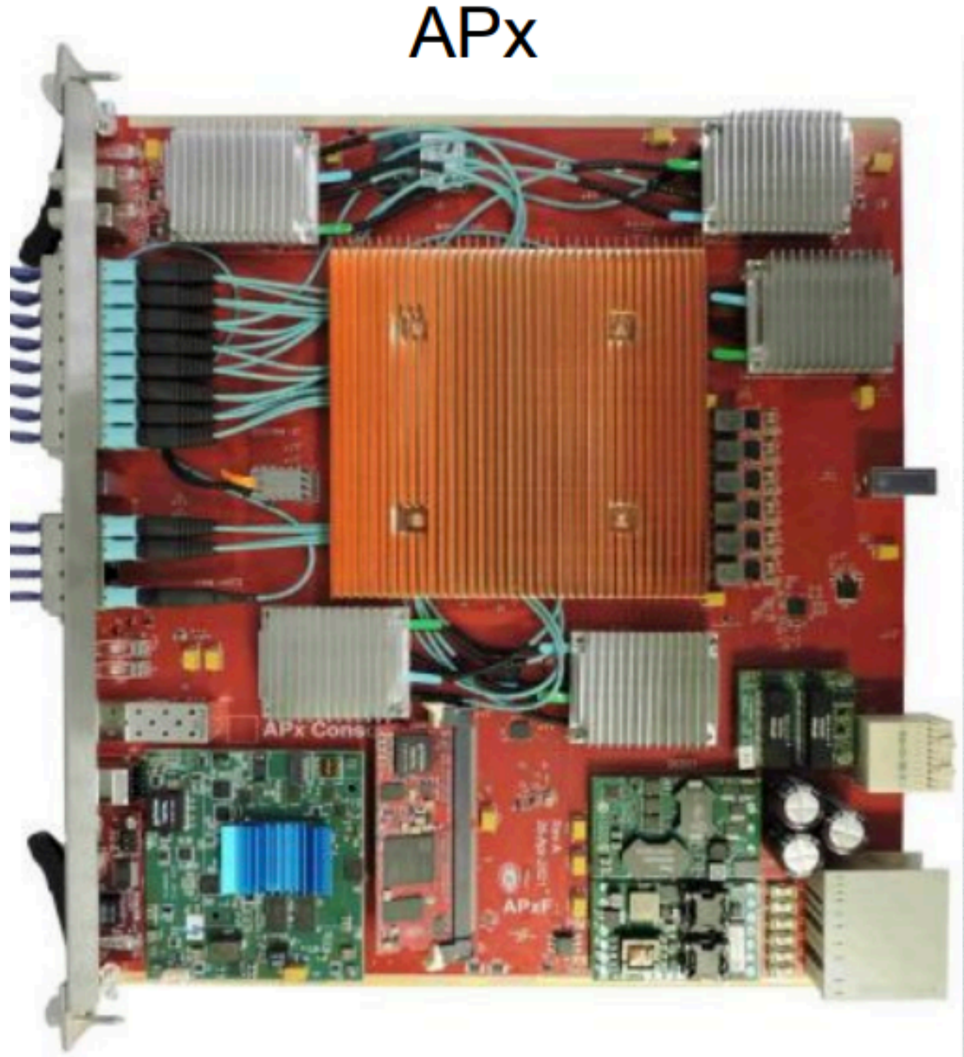
CMS Level 1 Trigger

- Phase 2 Upgrade of CMS L1T will have hundreds of boards with FPGAs like those shown below - AMD/Xilinx Ultrascale+ FPGAs
- Data rate of multiple terabits per second into / out of each board on optical fibres
- System organised in layers with normally ~ 1-2 μs per step
  - Reducing raw detector data into physics objects (e.g. track finding: hits to tracks)
  - New event every 25 ns, latency for trigger decision for one event 12.5 μs
- Final output is one bit: keep or discard event



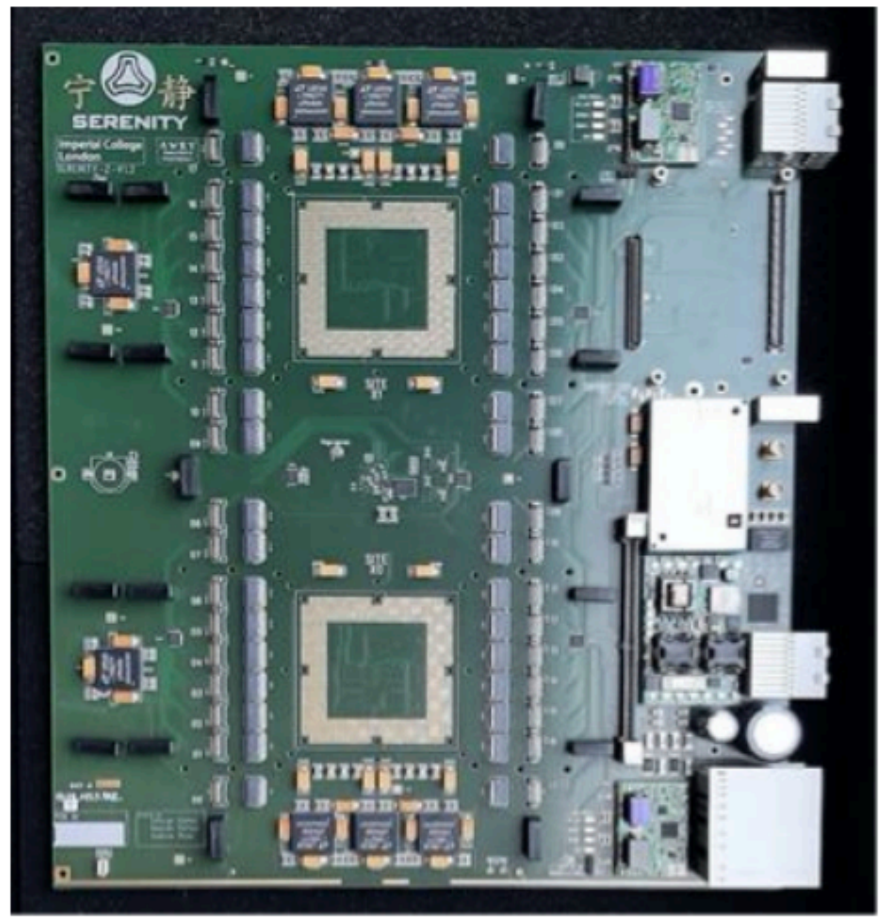


APx

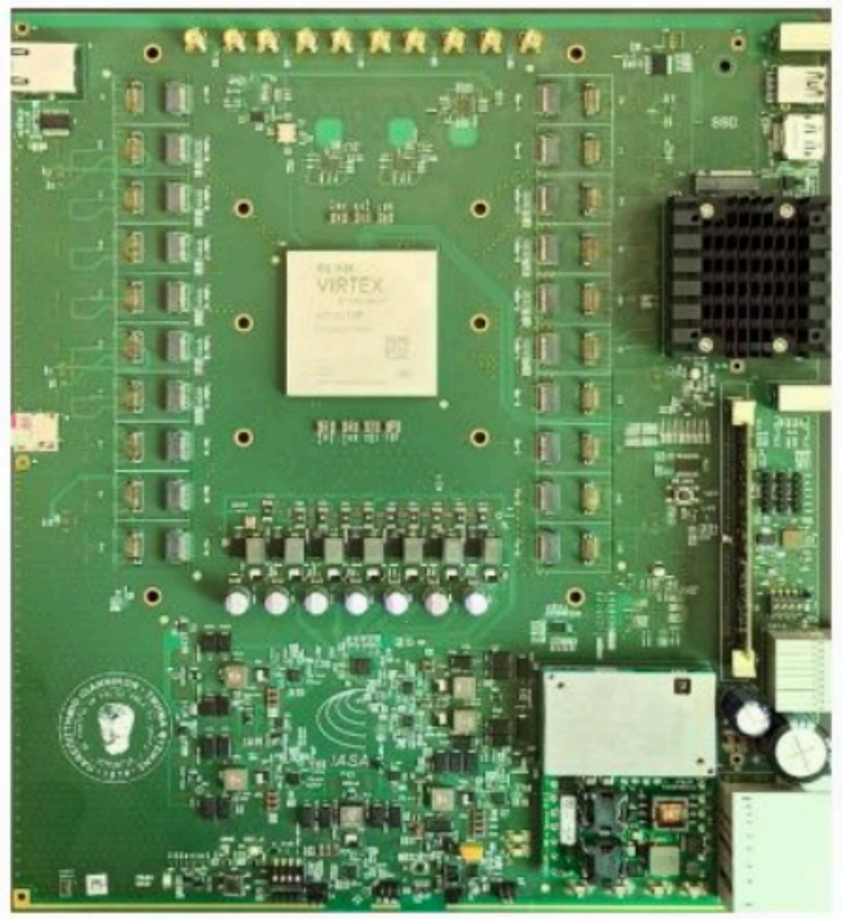


X2O

Serenity



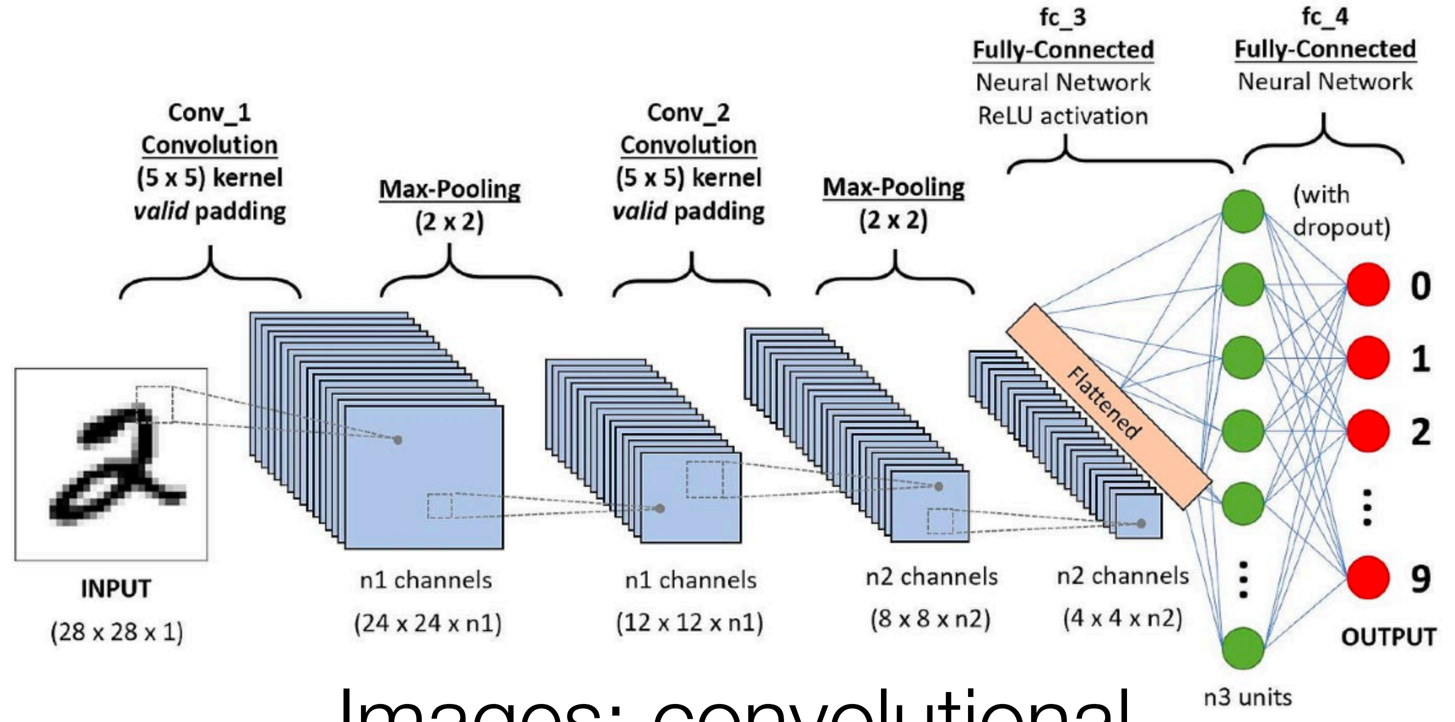
BMT



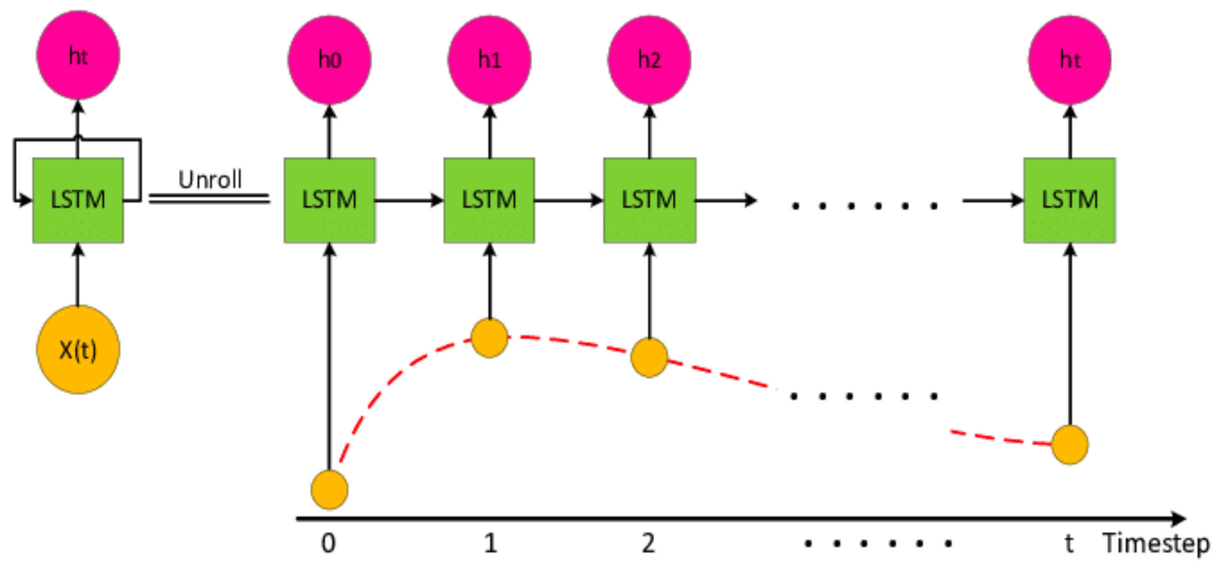
Machine Learning

- Build models that learn from data in order to make predictions on new, unseen data
- "*Models*" can be Neural Networks, Decision Forests, or anything else "trainable"
- "*Training*" is the process of fitting the model parameters that best describe the data
- "*Inference*" is the process of using a fitted model to make new predictions
  - For Fast ML at experiments we are mostly concerned with making fast inference
- ML used in HEP since the first ML wave in the 80s, and nowadays extremely prevalent
- Different model types for different data representations

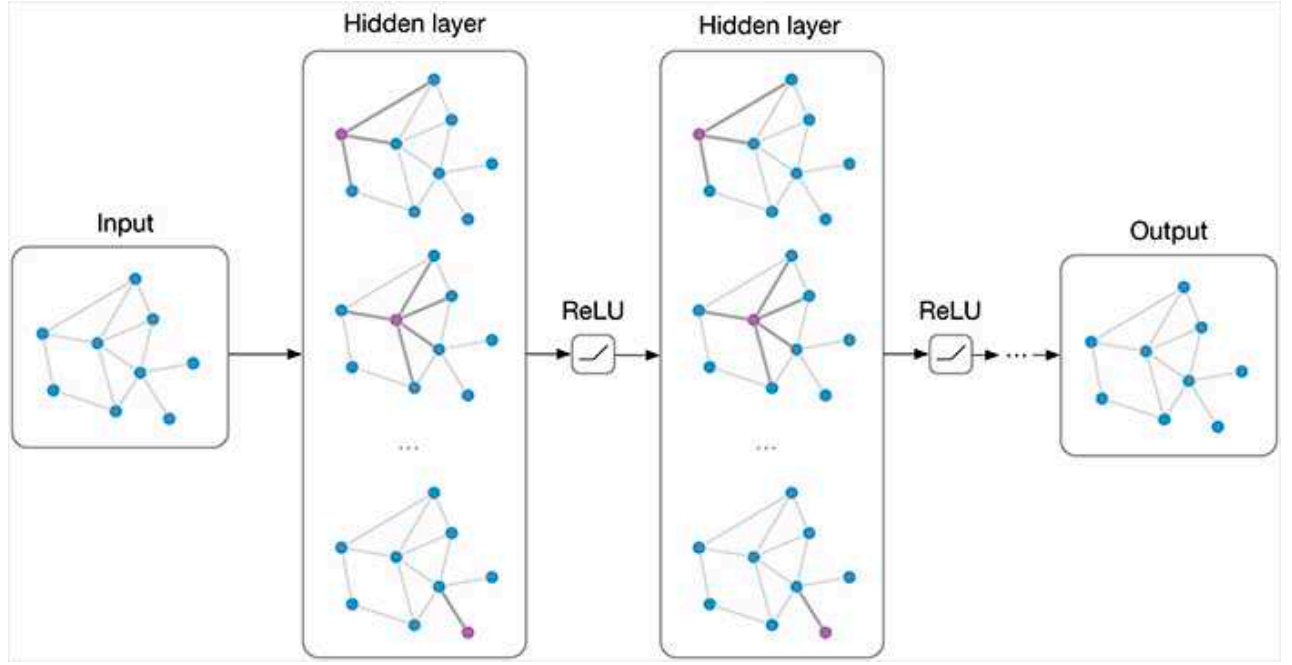
Tabular: fully connected or Decision Forest



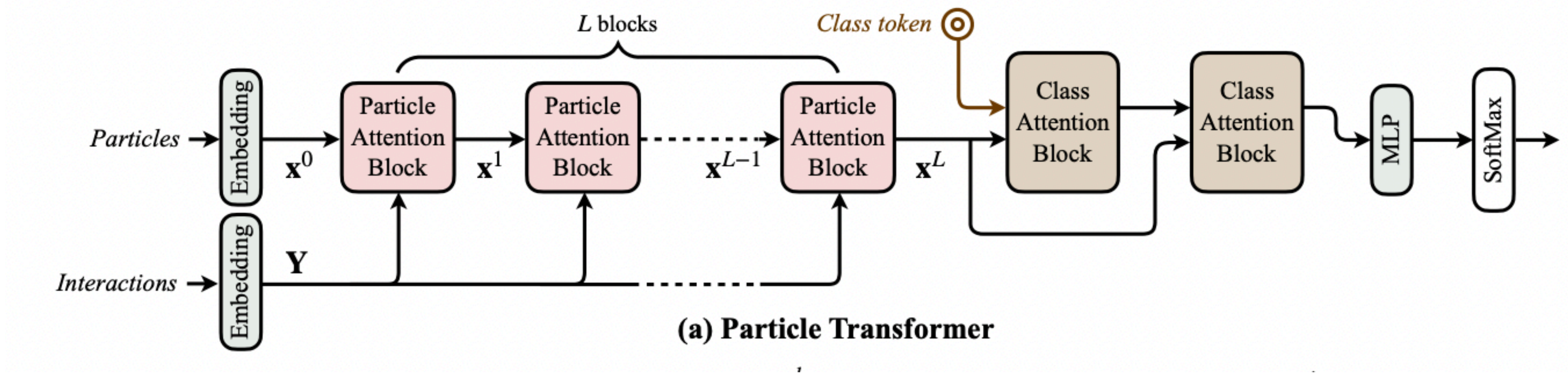
Images: convolutional



Time series: recurrent



Point cloud: graph

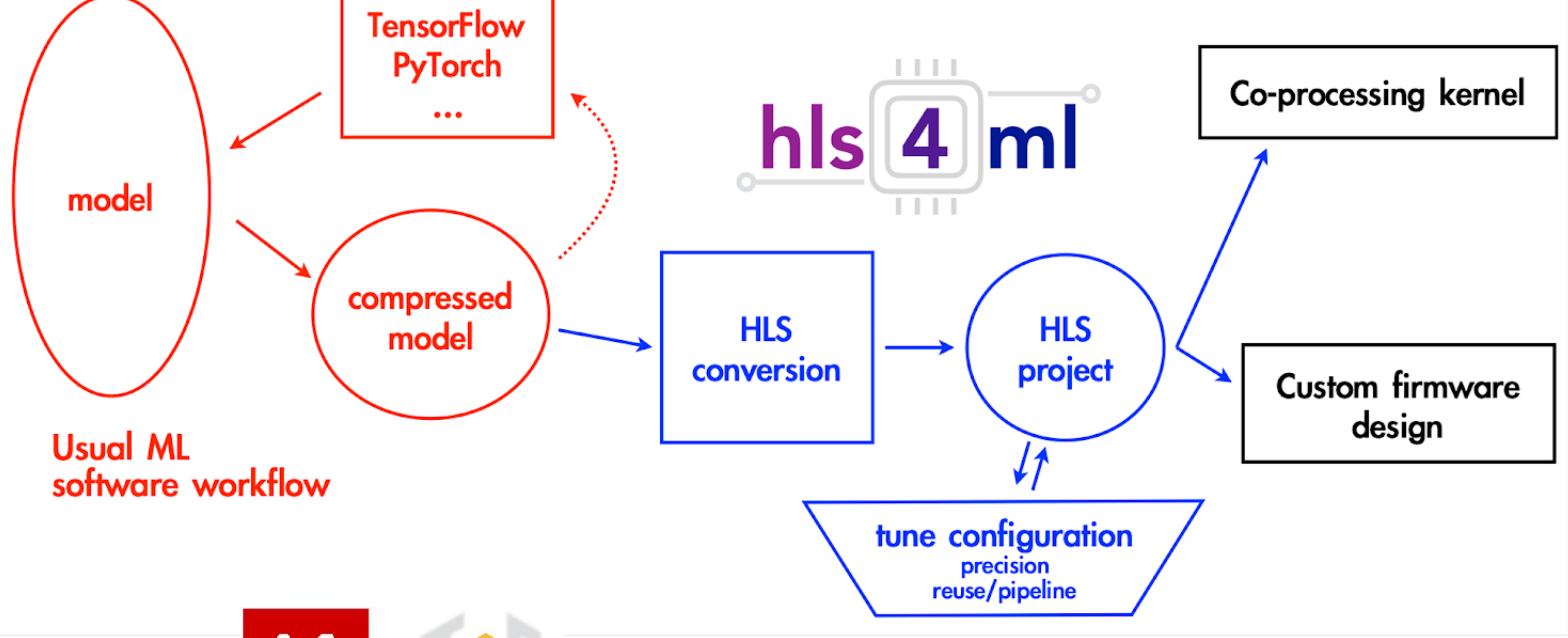


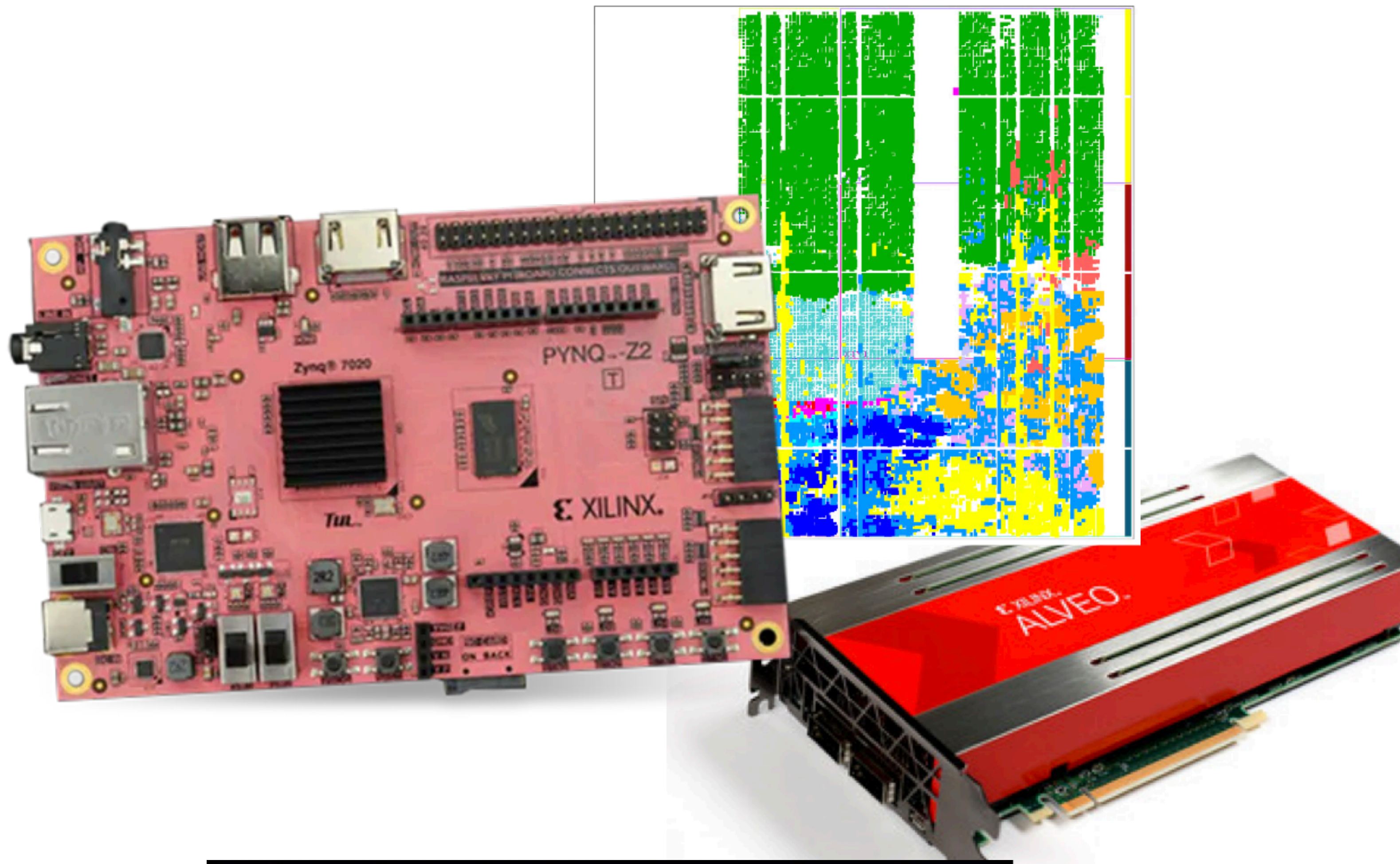
Sequence-to-sequence: transformer

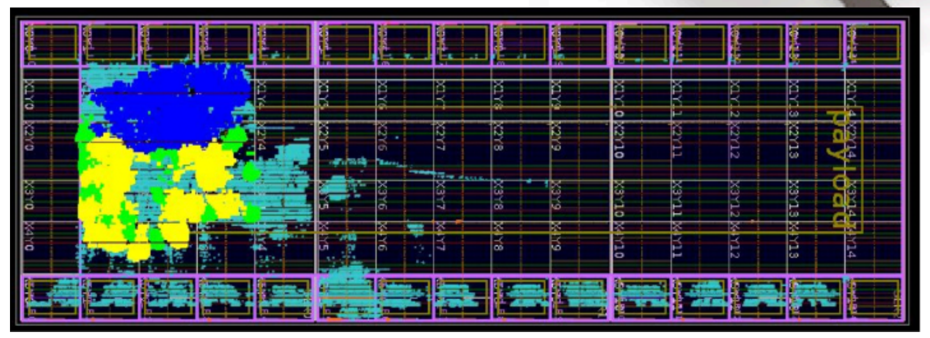
hls4ml high level synthesis for machine learning

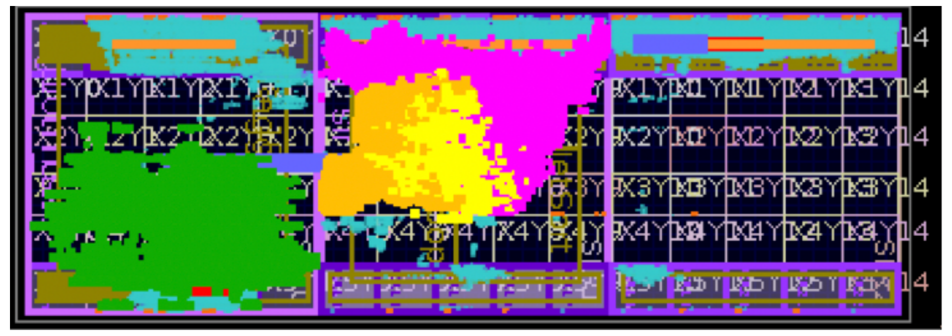
fastmachinelearning.org/hls4ml/

`pip install hls4ml`









(Q) K + TensorFlow

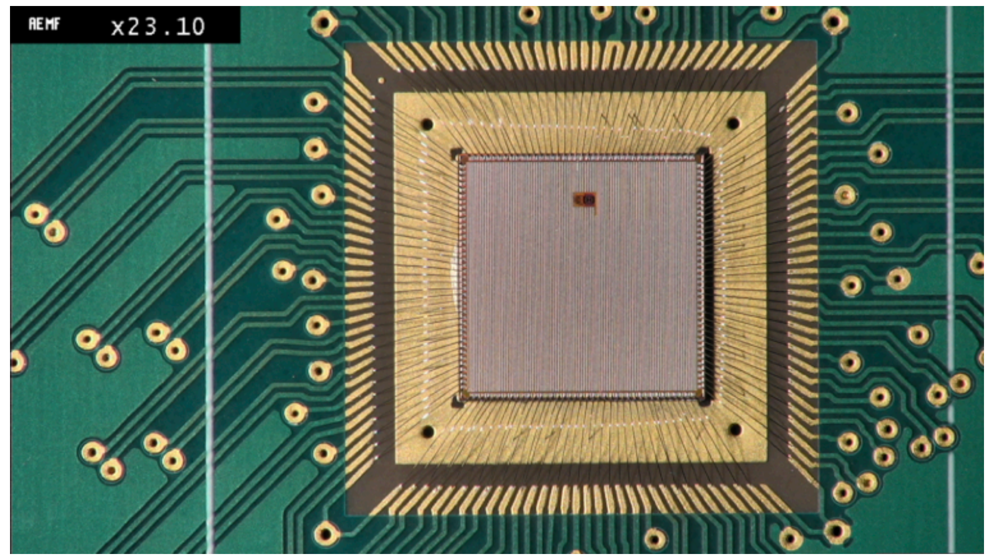
(Q) ONNX

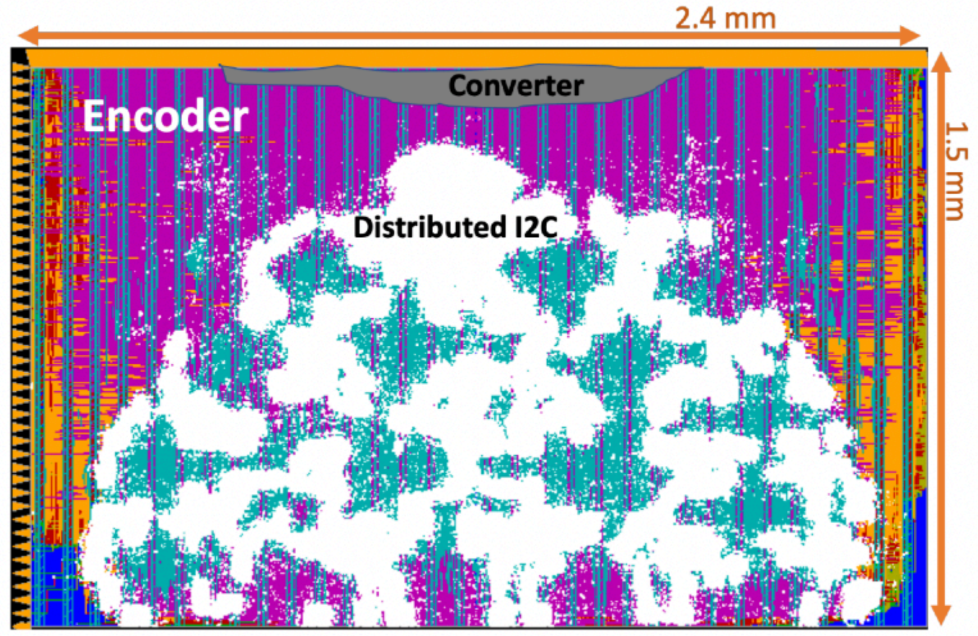
PYTORCH

Vivado™ HLS

HLS COMPILER

Mentor A Siemens Business
Catapult
Coming soon





hls4ml - Dataflow Architecture

- Dataflow architecture: each layer is an independent compute unit
  - With tunable parallelism and quantization
- Fully on-chip: NN must fit within available FPGA resources (pynq-z2 floorplan shown)
  - Example: small CNN trained on MNIST

Conv2D

ReLU

MaxPool2D

Conv2D

ReLU

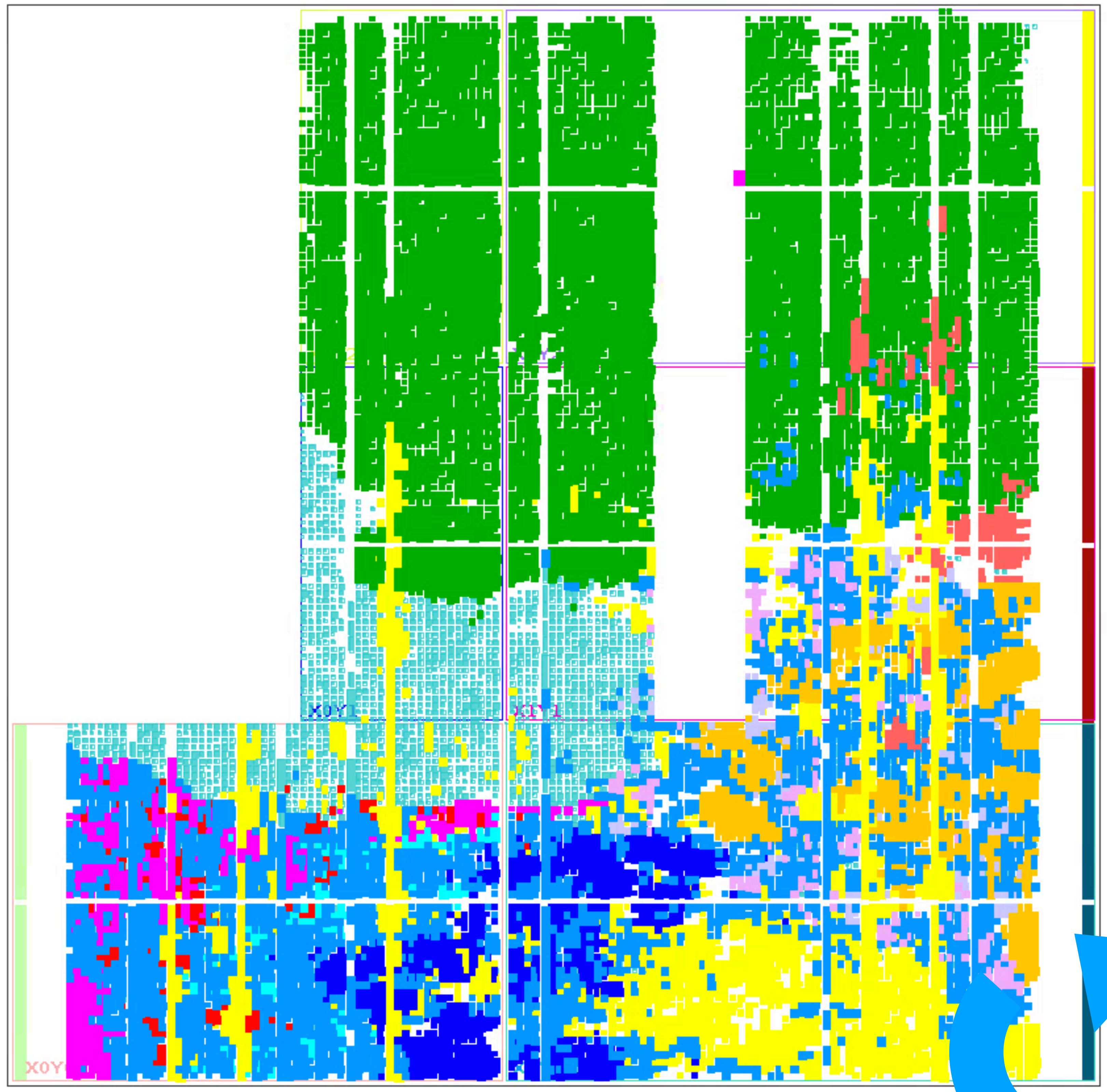
MaxPool2D

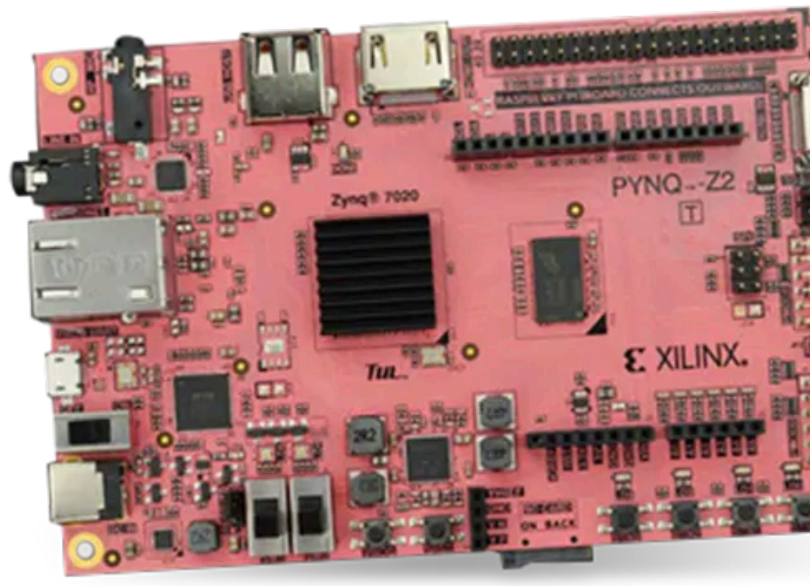
Flatten

Dense

Softmax

Prediction

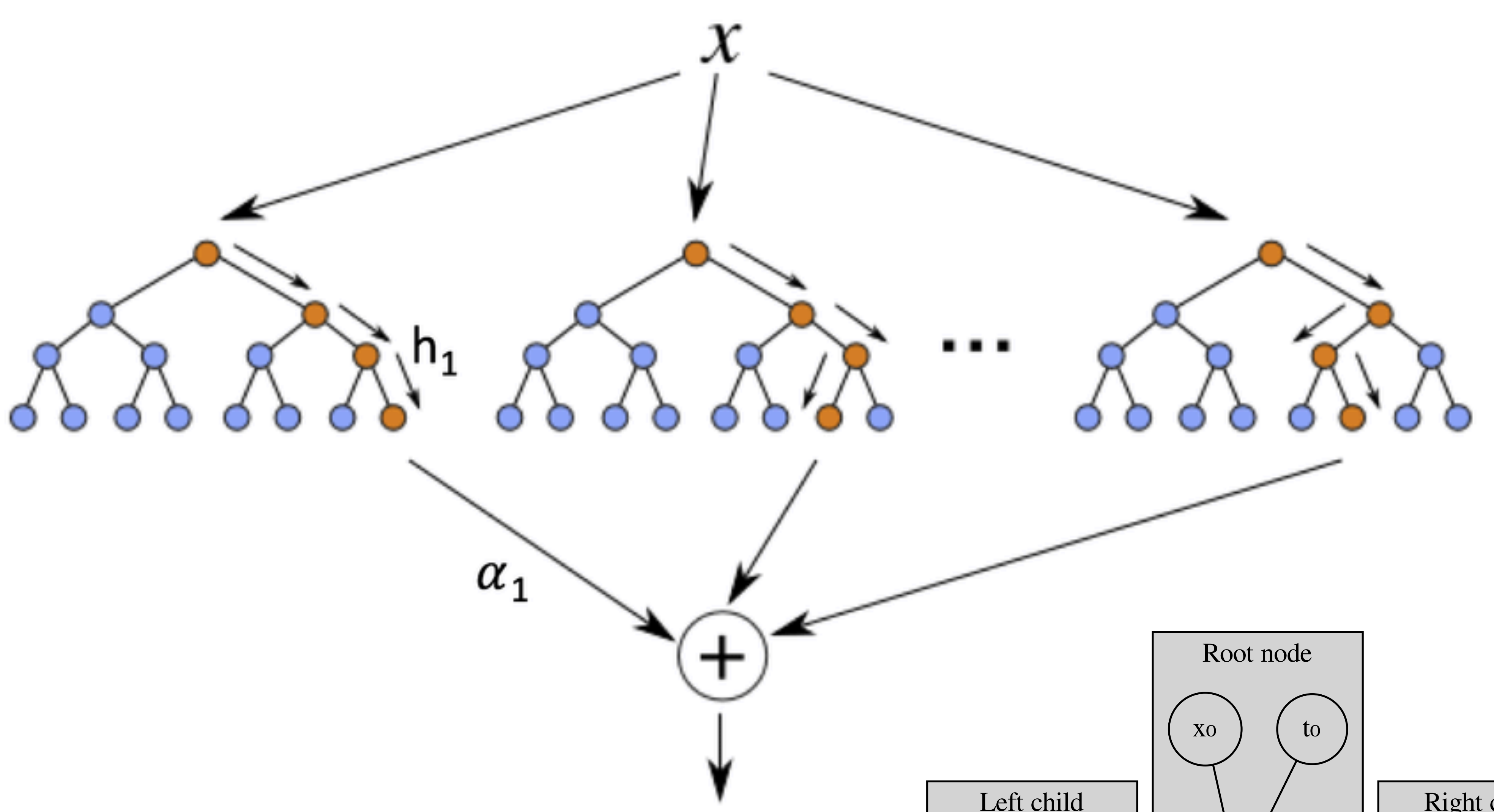


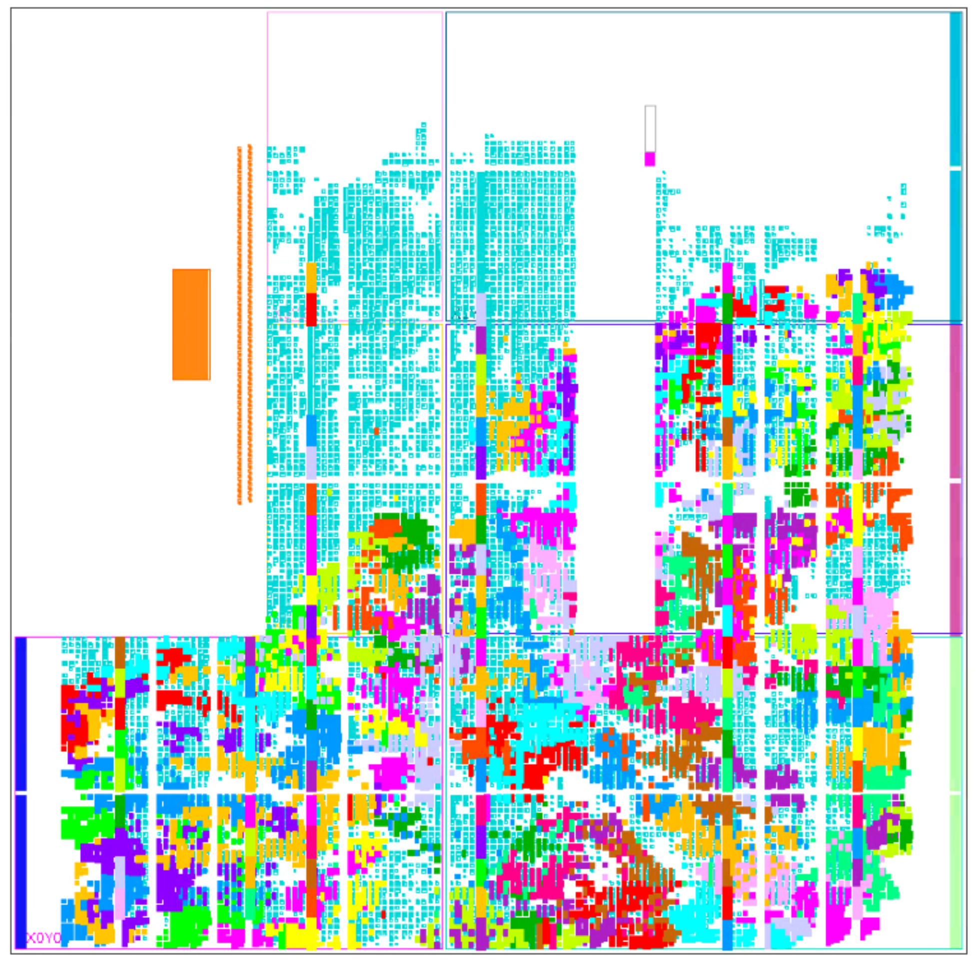


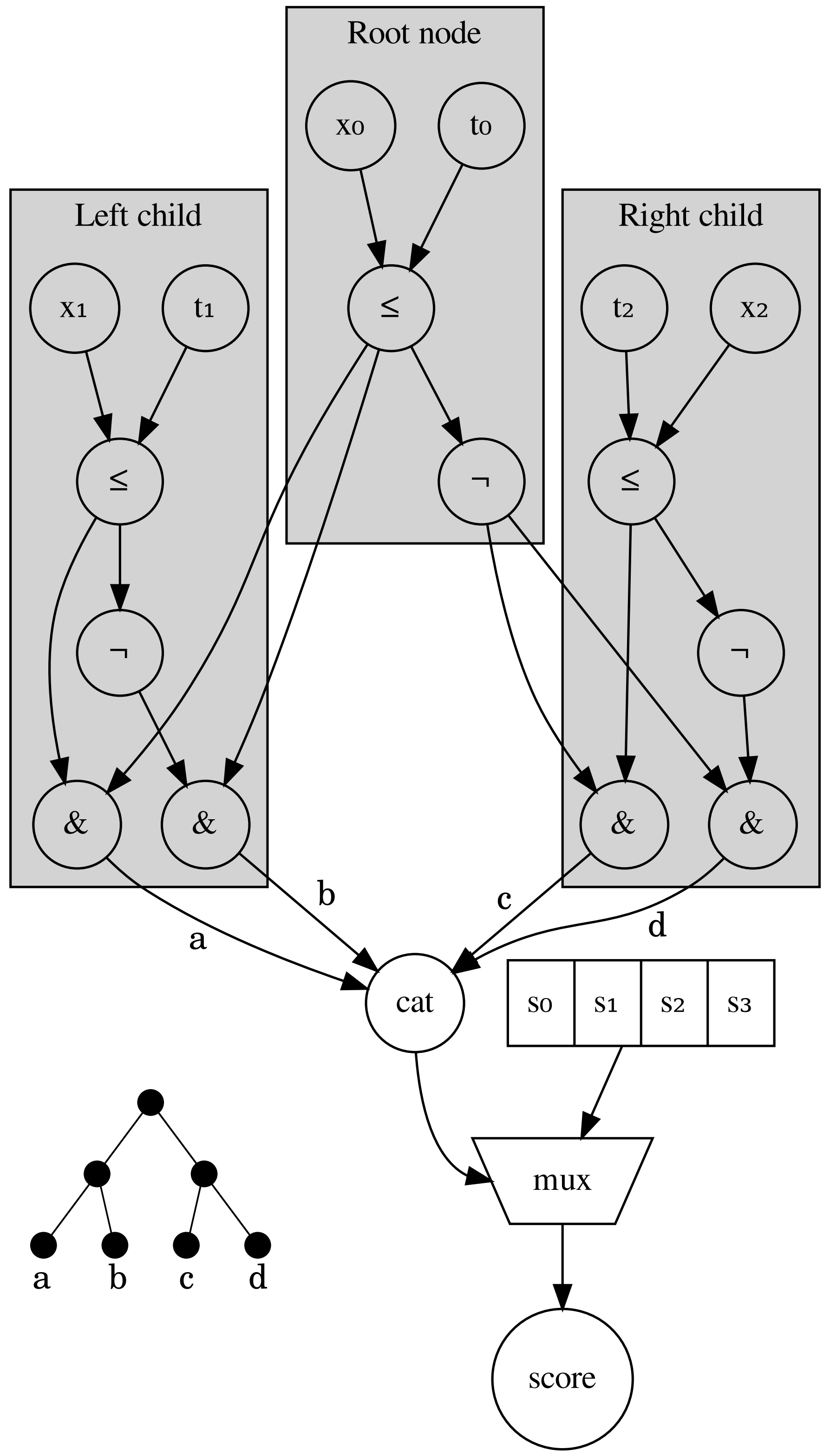
FIFOs

conifer for Decision Forests

- Neural Networks like Transformers for Large Language Models dominate the ML discourse
- But the old ways are still relevant: Decision Forests ("MVAs")
  - Fast, lightweight, robust (arXiv:2207.08815, IML keynote)
- **conifer** is to DFs as hls4ml is to NNs
- A Decision Tree *splits* on data variables until reaching a *leaf*
  - Leaves associate a *score* corresponding to prediction probability
- A Decision Forest is an ensemble of Decision Trees
  - Randomisation of each DT as a form of regularisation
  - Ensemble score is an aggregation over trees e.g. sum
- **conifer** maps DFs onto FPGA logic
  - Implemented with high parallelism for low latency and high throughput



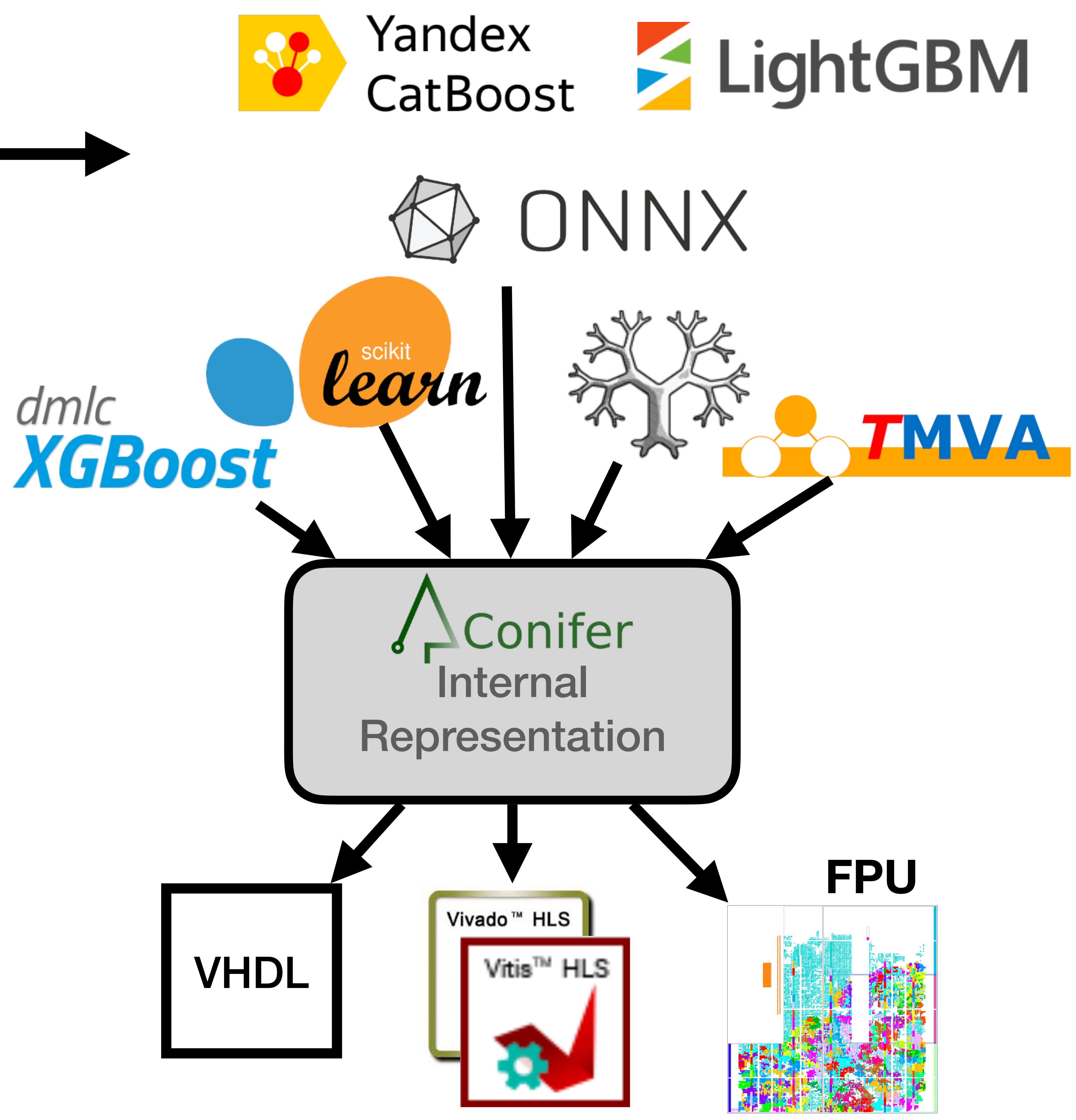




Conifer

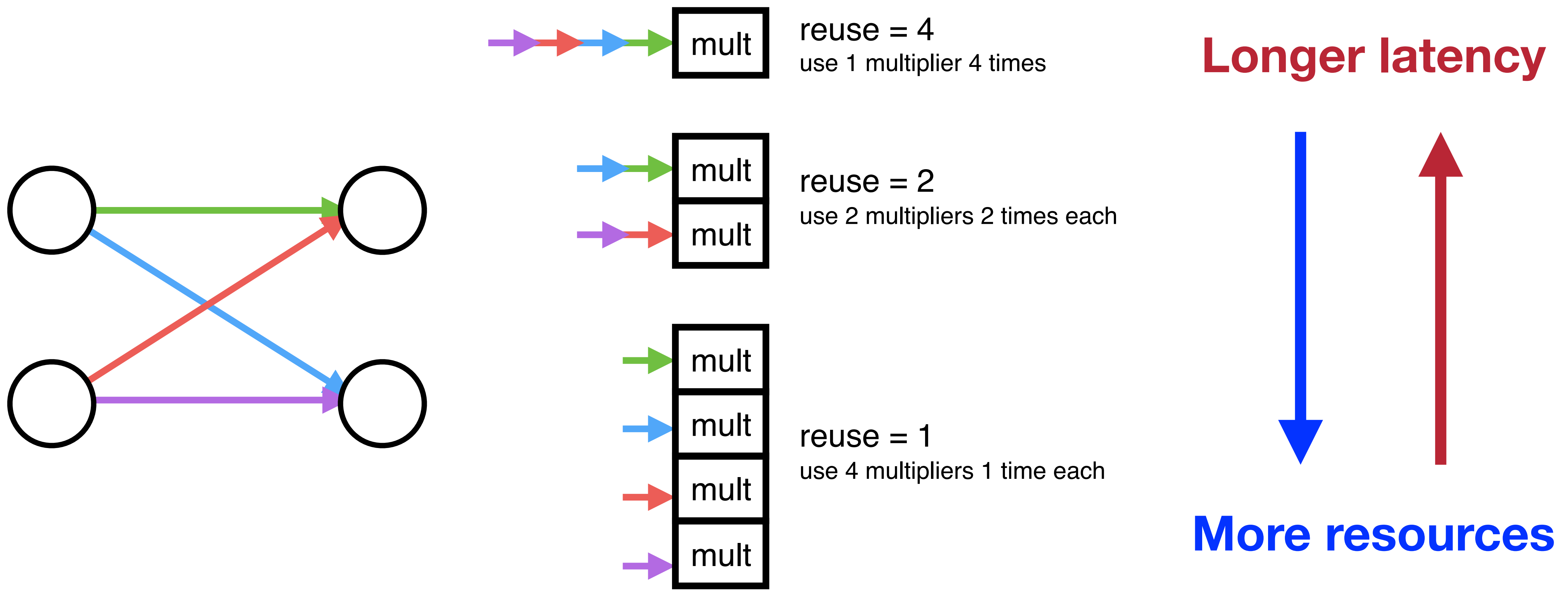
conifer implementations

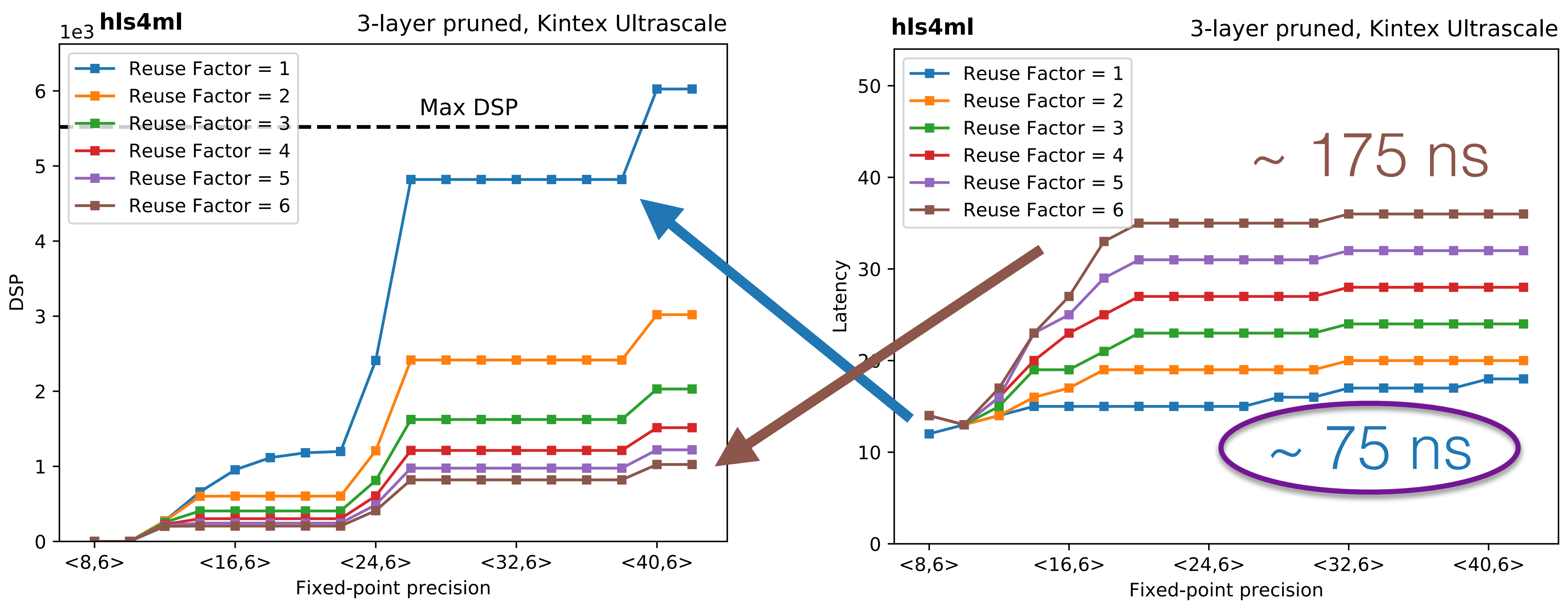
- Very much like **hls4ml**, **conifer** has frontends, an Internal Representation, and backends
- Frontend support for popular BDT training libraries
- Backends: HLS, (hand-written) VHDL, Forest Processing Unit (FPU)
- HLS and VHDL backends map one DF to one hardware implementation
  - Capable of inference at O(10) ns latency, O(100) MHz throughput
- FPU is a reconfigurable module that new models can be loaded onto
  - Binaries for some AMD devices for download
  - Implemented with HLS
- GitHub, website, paper, `pip install conifer`



hls4ml and conifer key features

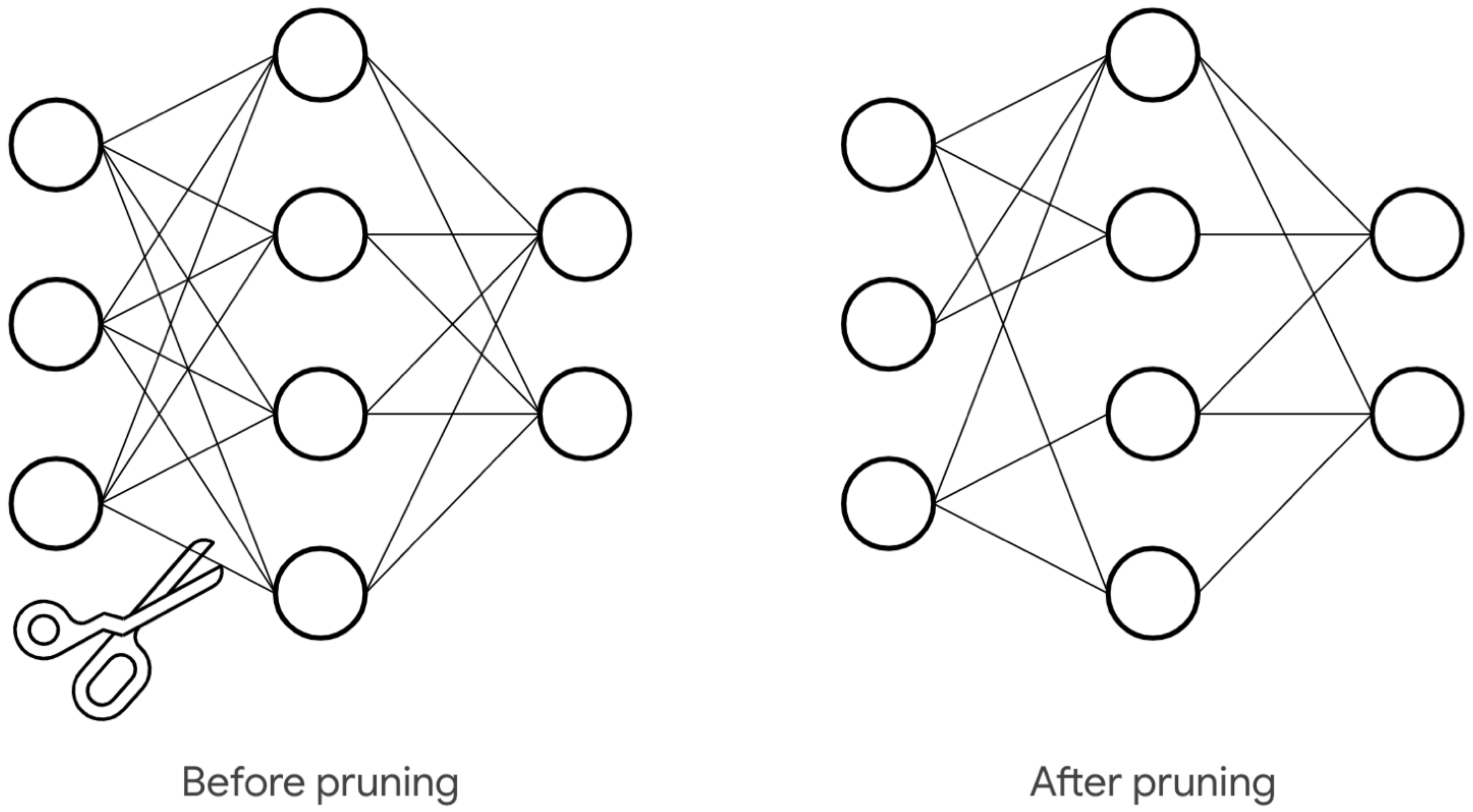
- Easy to use
  - Reduce the barrier to entry for non hardware experts
  - Python packages with nice interfaces to EDA tools
  - `pip install hls4ml conifer`
  - Configuration interface for fine-grained control
  - Tutorial and documentation for getting started
- High Level Synthesis implementations (C++)
  - More accessible, and powerful Design Space Exploration
- Open source software, open communities
  - fastmachinelearning.org
- Massively parallel for low latency and high throughput
  - 'Unrolled' implementations
- Common interfaces
  - Make it easier to integrate generated IPs into designs

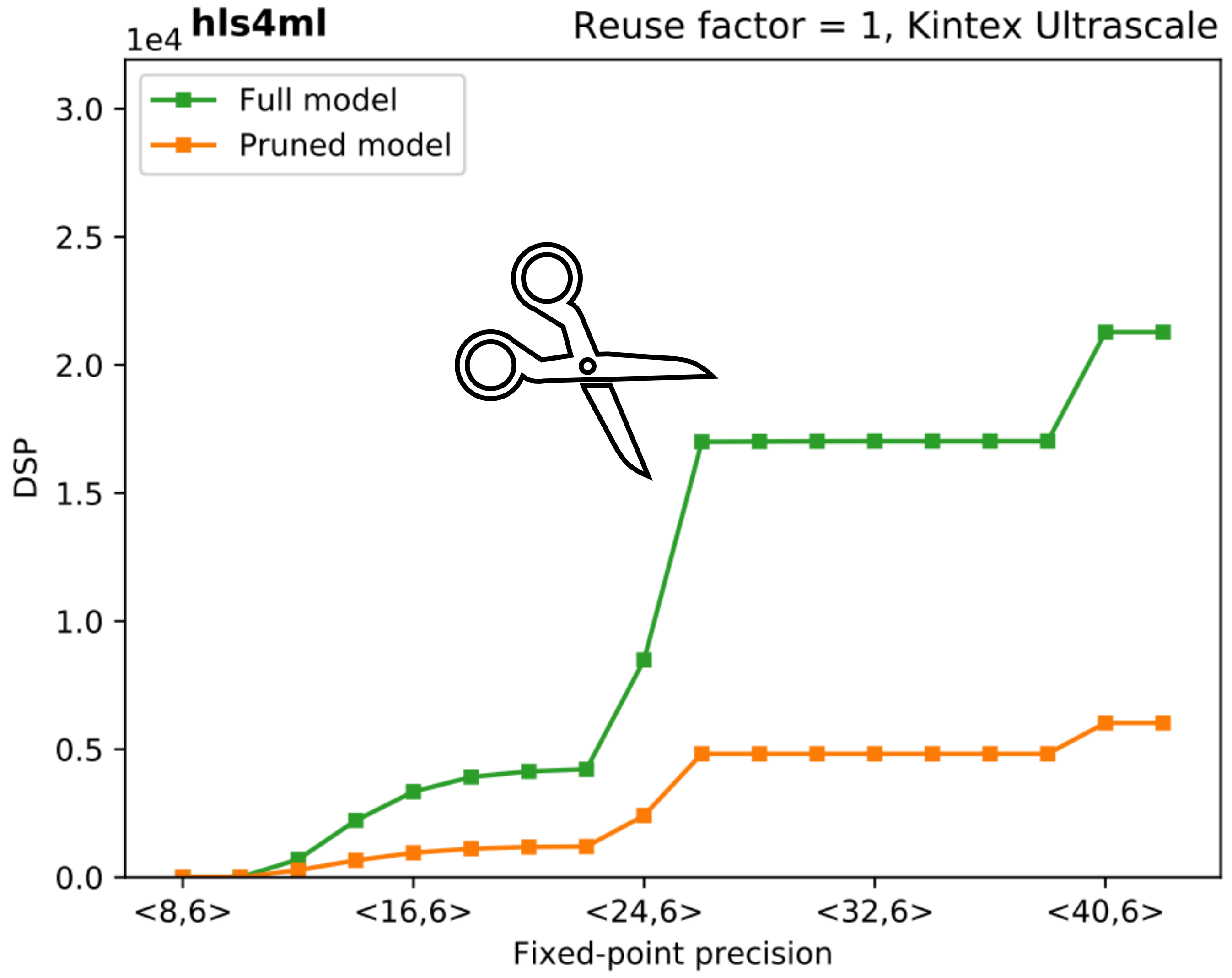


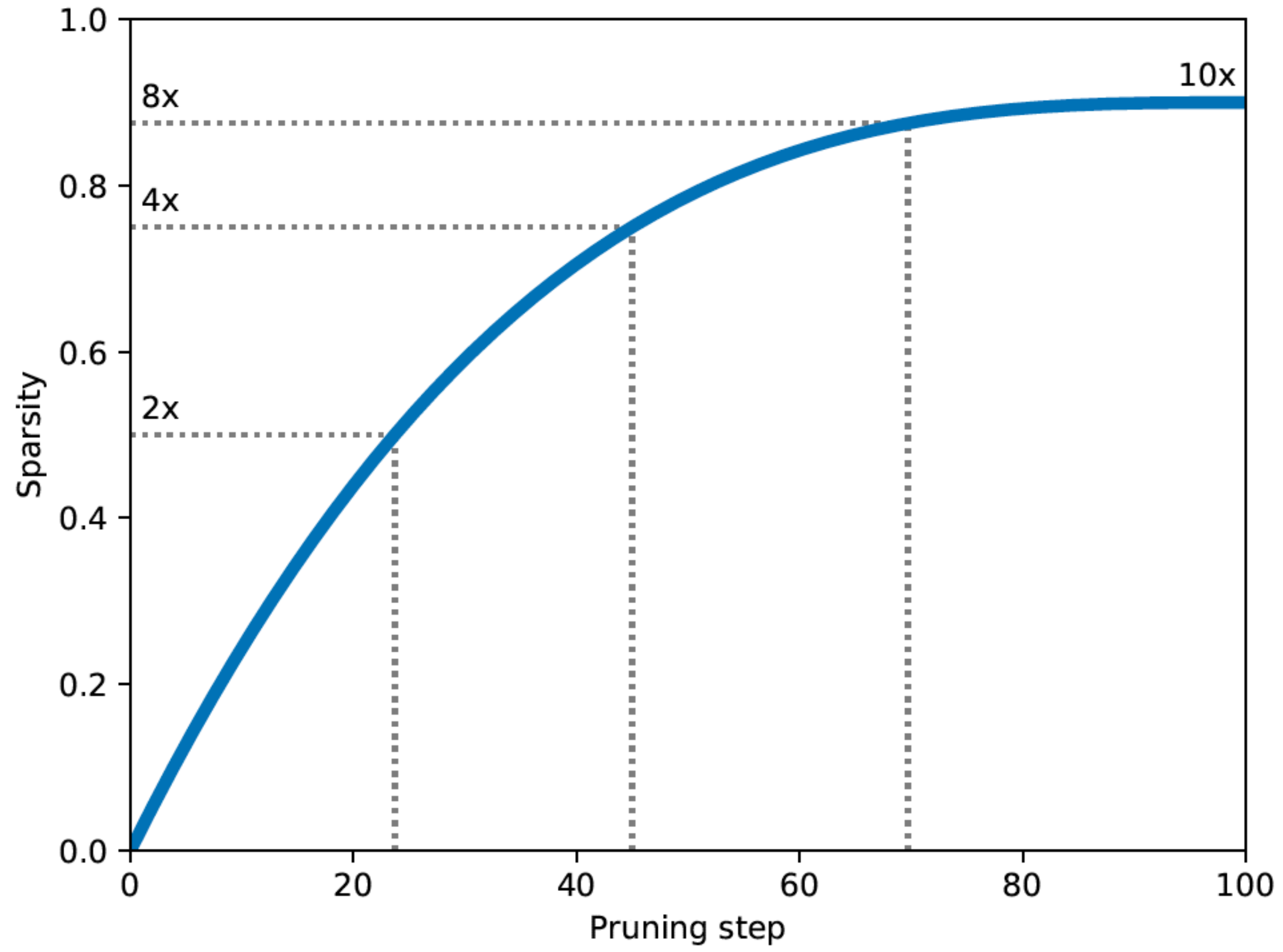


Efficient training: pruning

- A Neural Network or Decision Forest may contain many redundant connections
- Pruning methods generally remove some connections from the final model
  - Can improve generalisability also
- **hls4ml** and **conifer**'s fully unrolled implementations can avoid unnecessary logic for pruned connections and save resources (lower left image)
- Different methods:
  - Regularisation (penalise low value weights, then make them 0)
  - Target sparsity, e.g. sparsity ramp up with TFMOT (lower right image)
  - Structured pruning - remove continuous blocks of weights; Filter pruning - entire filters of CNN



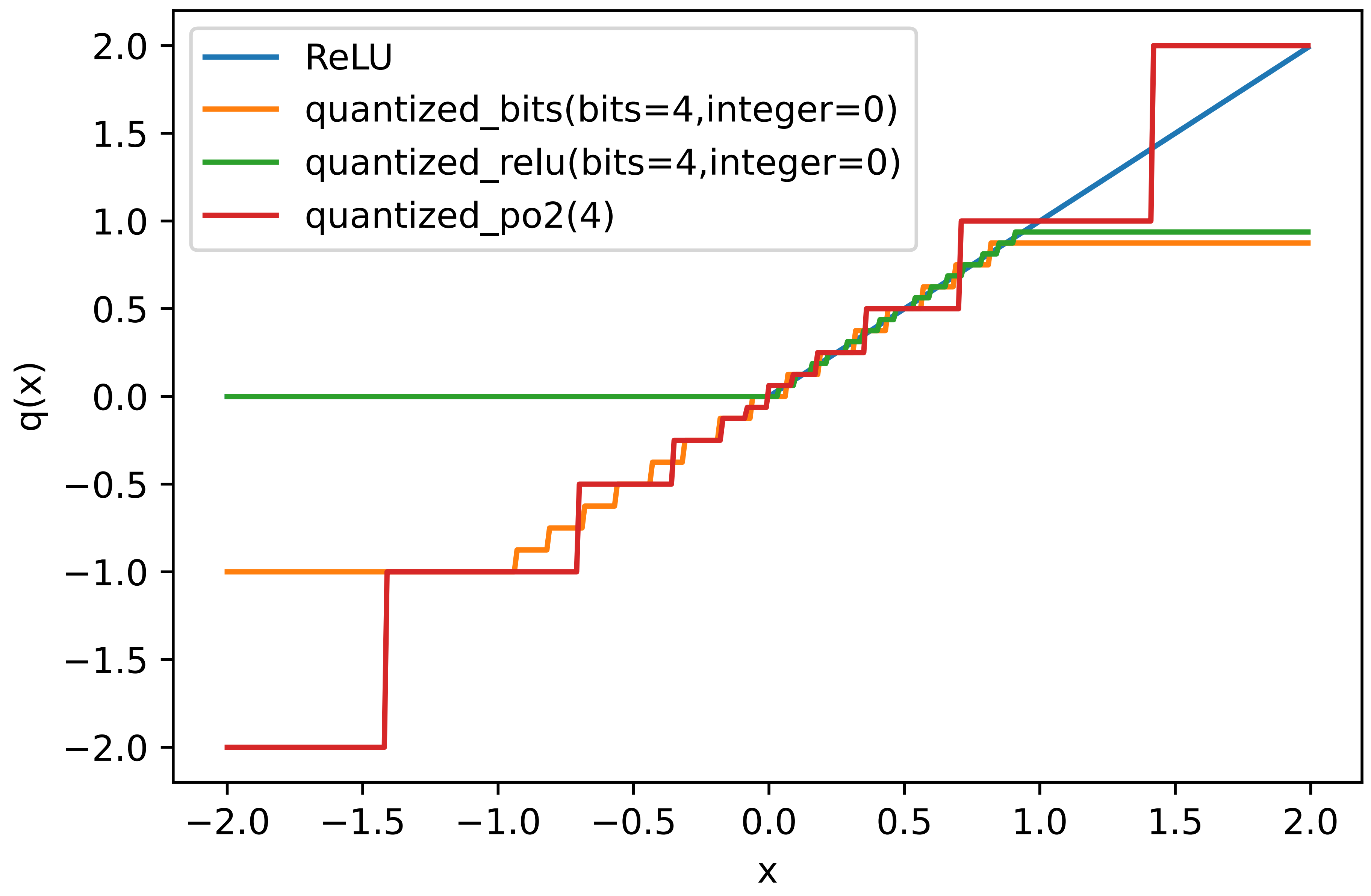


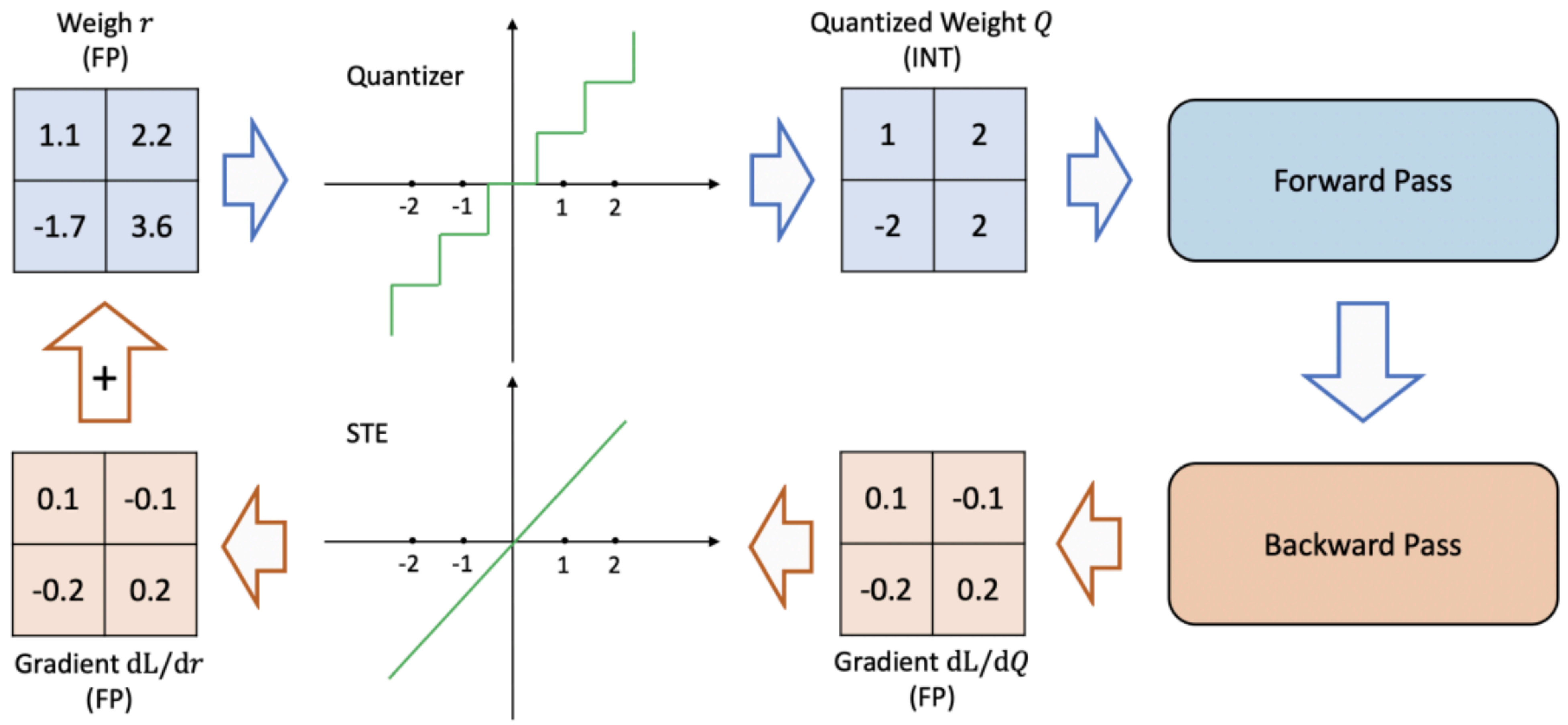


Images from Tensorflow blog

Efficient Training: Quantization

- Possibly the main technique for making NNs cheaper in FPGAs!
- Using regular TensorFlow Keras or PyTorch, you typically train with floating point
  - We like to avoid *floating point* in FPGAs - much more costly in resources & latency than *fixed point*
  - You can do P*ost-Training Quantisation* (PTQ): represent the float values with some fixed point
- With *Quantization Aware Training* (QAT), you constrain weights/biases/activations to fewer values during training
  - Superior to PTQ for lower bitwidths - can go all the way down to 1 bit (representing ±1)
  - Use quantizations with efficient hardware operators: integer, fixed point, power of 2
  - Use 'Straight Through Estimator' for back propagation step



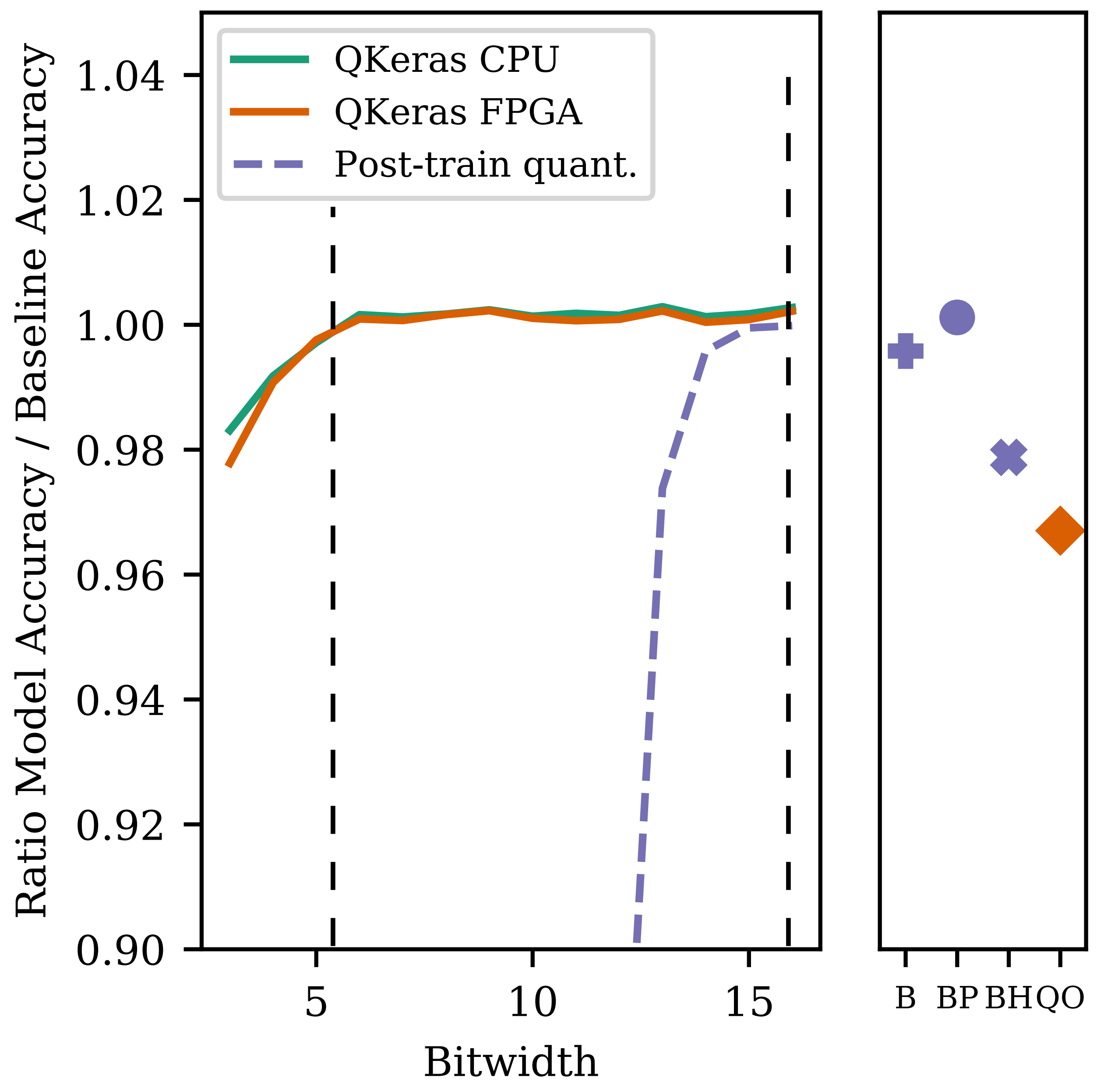


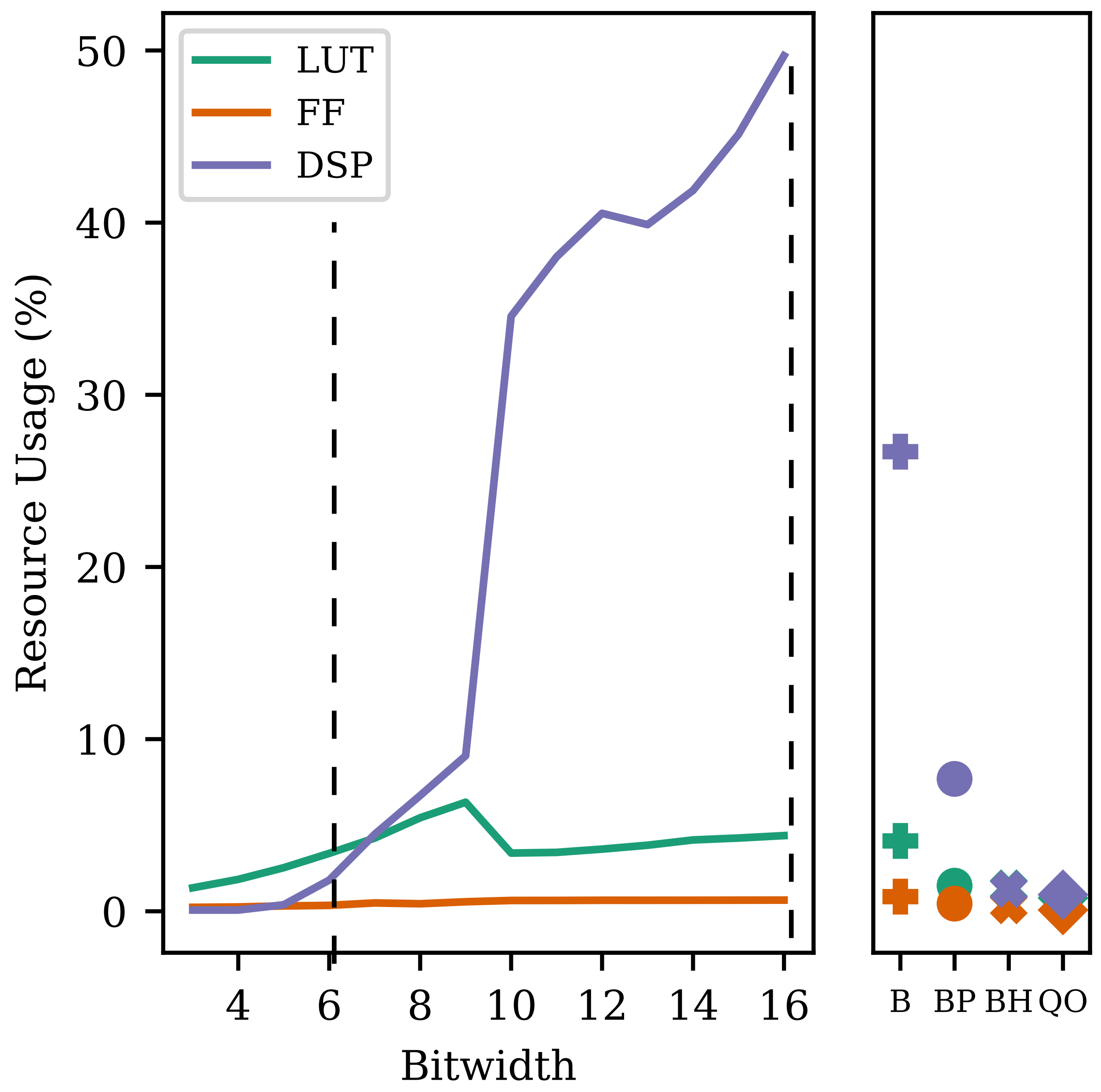
arXiv:2103.13630

QKeras

doi: 10.1038/s42256-021-00356-5

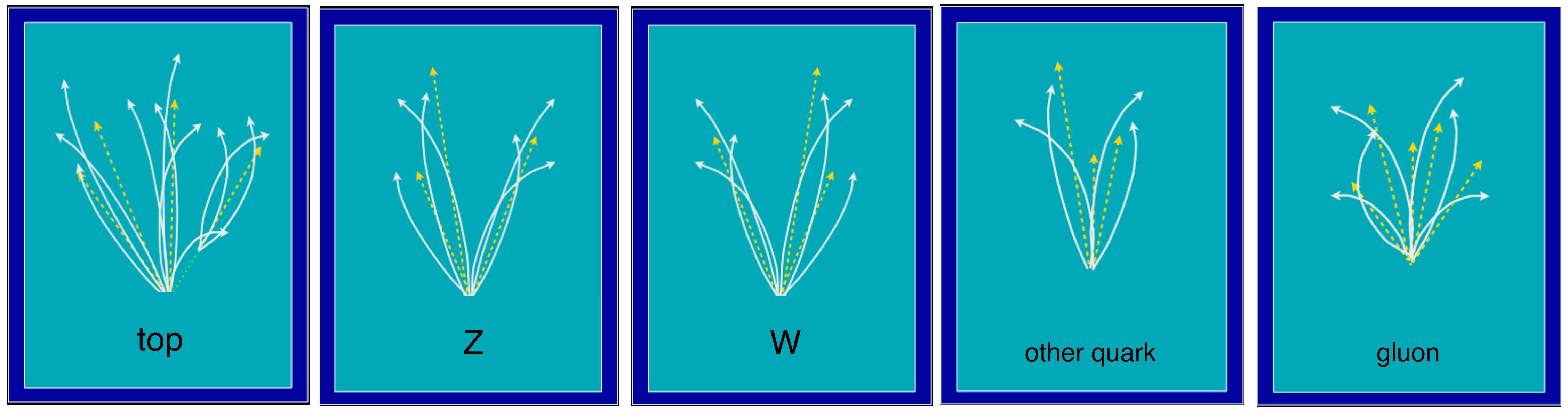
- QKeras is the Quantization Aware Training extension of Keras
- We trained ‘normal’ floating point NNs with Keras and low-precision NNs with QKeras on a benchmark jet tagging problem
- (Top plot) accuracy with QKeras training down to 6-bits is lossless wrt floating-point Keras
  - Big improvement over ‘post-training quantization’
  - Dashed line → solid lines
- As we reduce bitwidth, resource usage goes down
  - At small bitwidths LUTs are preferred over DSPs
  - The ‘critical resource’ usage decreases from **56%** (DSPs) for the Baseline (B) to **3.4%** (LUTs) for the 6-bit QKeras model (no performance loss)
  - QO model is tiny (1% DSPs/critical), 2% lost accuracy
- Right panel ‘QO’ shows some automatic optimisation of the bitwidth trading accuracy vs resource cost (AutoQ)

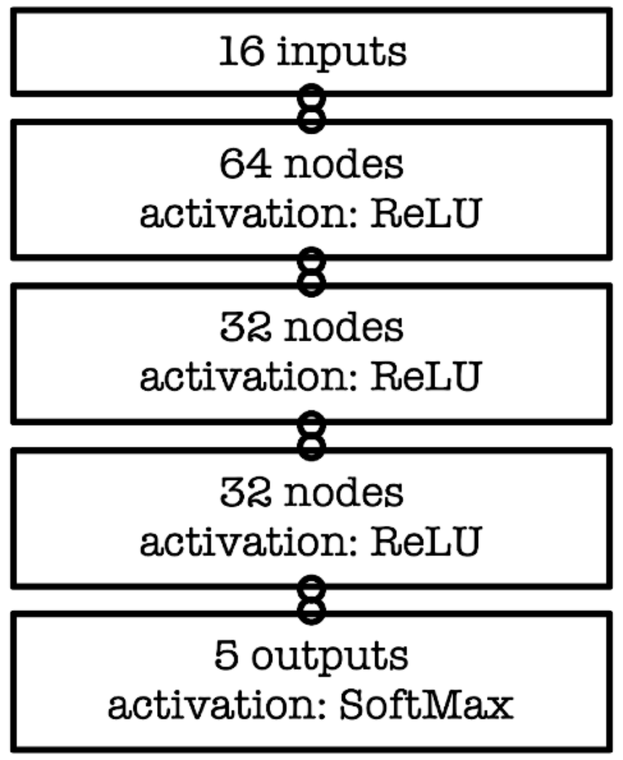


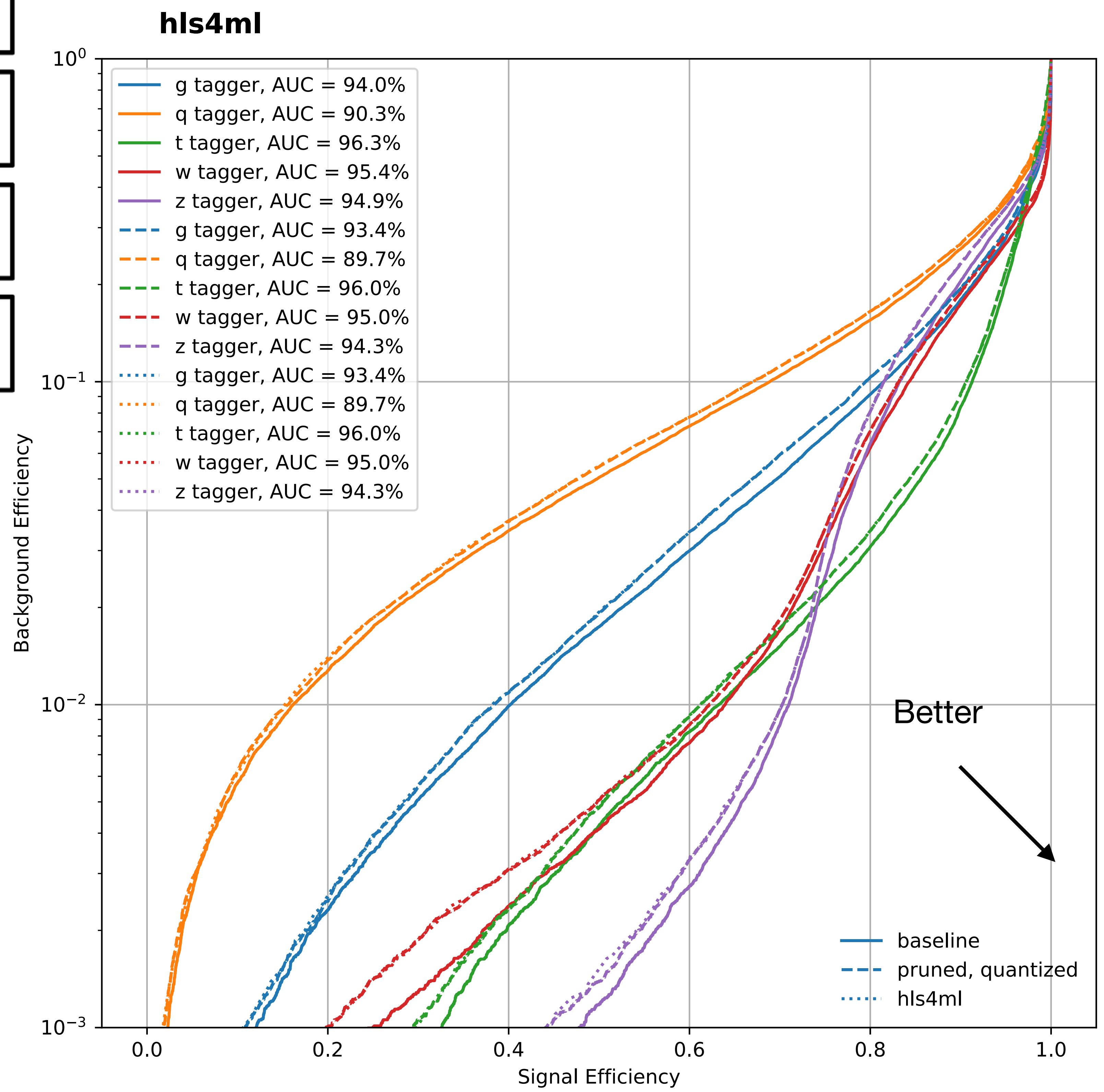


Example of QAT & Pruning

- From the hls4ml tutorial
  - Tagging jets (5 classes q/g/t/W/Z, 16 input variables)
- 3 hidden layer MLP (Dense layers):
  - 1) Keras floating point training, 16b inference
  - 2) QKeras with 6 bits for weights, biases, activations & 75% sparsity target with TFMOT
  - Minimal code changes required to go from 1) to 2)



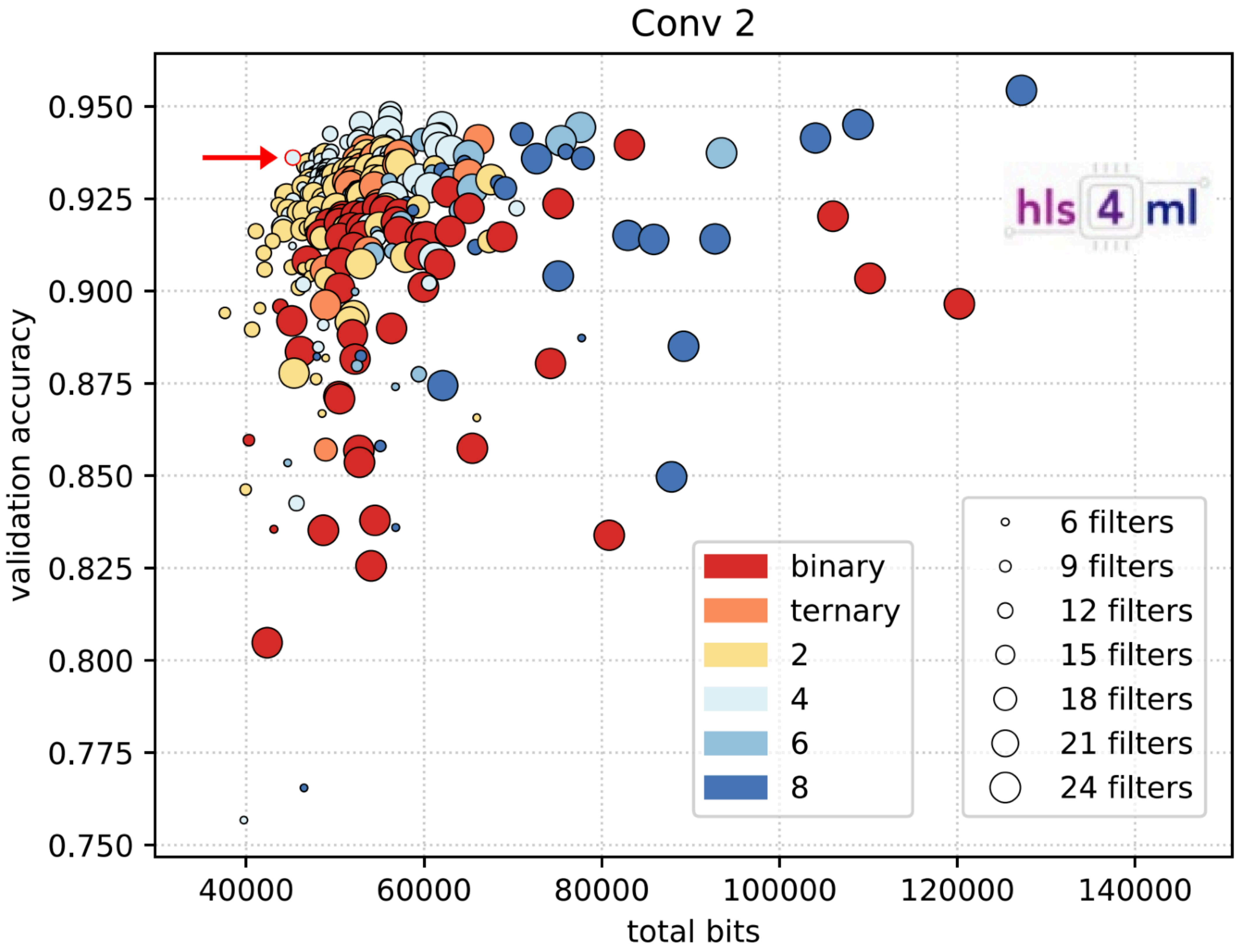




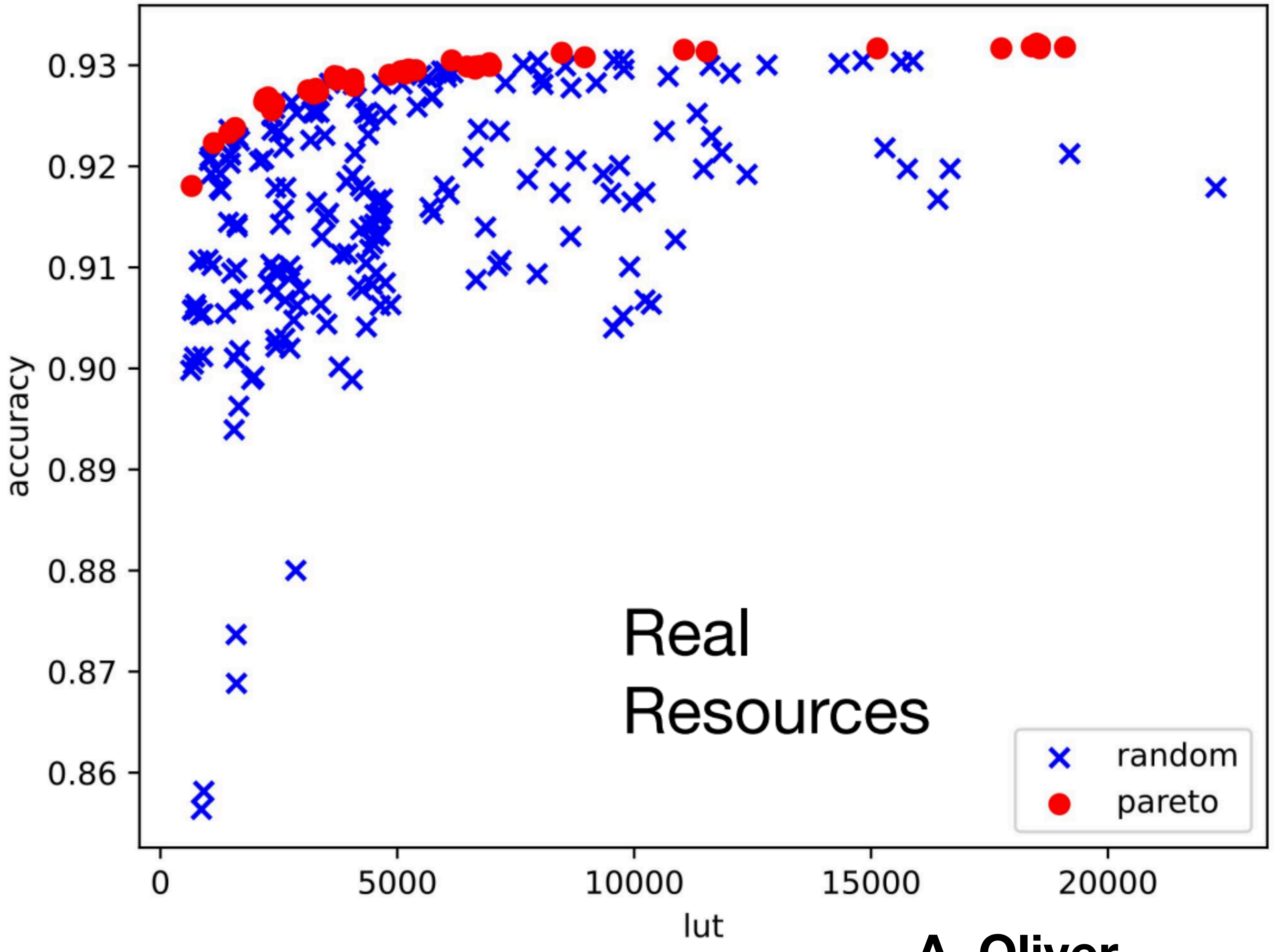
%VU9P	Latency	DSP	LUT
Keras 16b	50 ns	1890 (15%)	5%
QKeras 6b	40 ns	22 (~0%)	1%

Optimised training for hardware

- With some heuristic for runtime cost, models can be optimised at training time to tradeoff accuracy and cost
- Left example: total bits of model parameters as proxy for hardware cost vs model accuracy
- Right example: finding Pareto optimal Decision Forests for accuracy and resources using fast estimation



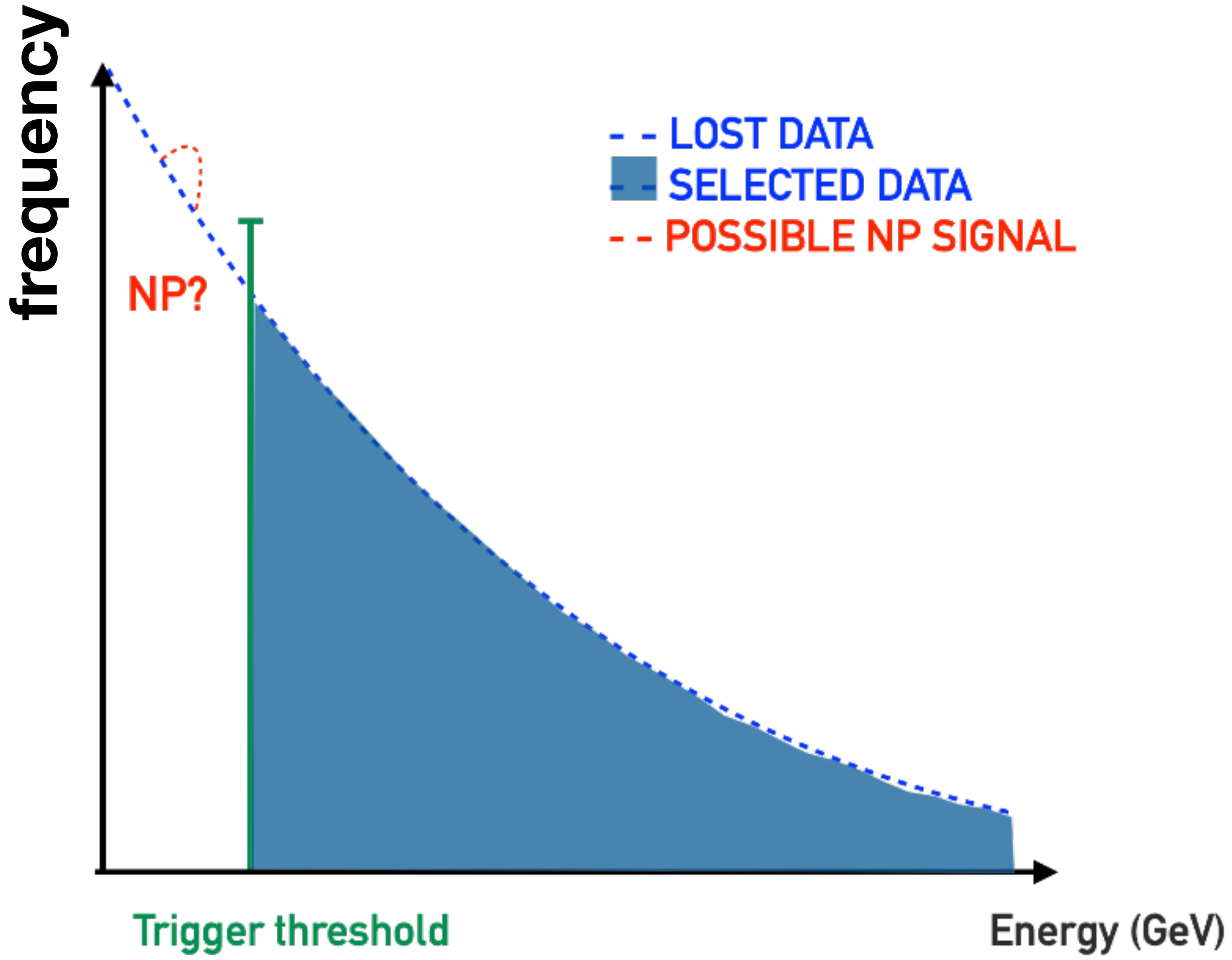
doi: 10.1088/2632-2153/ac0ea1

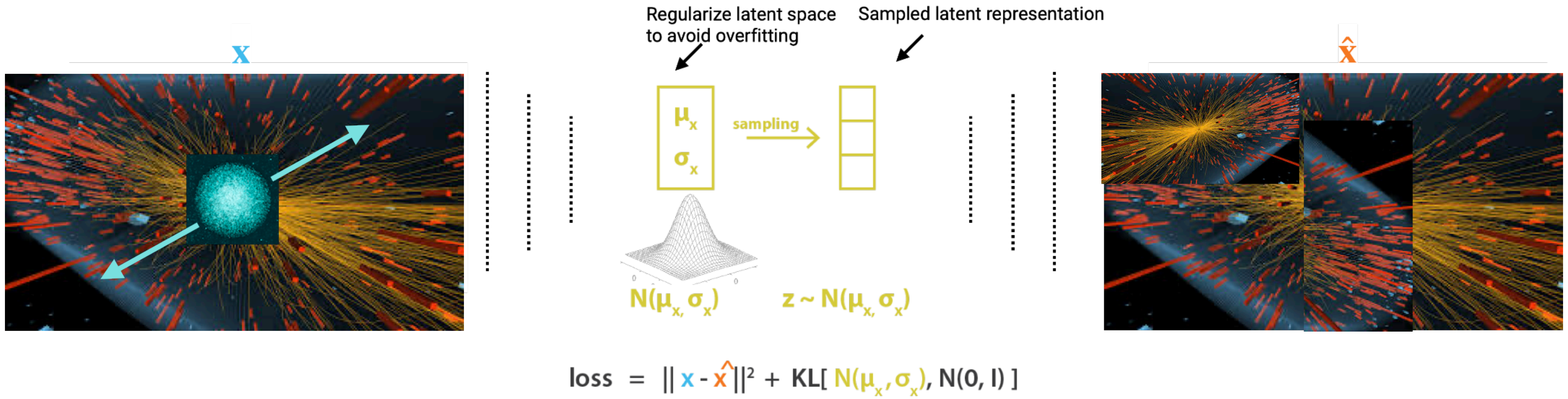


A. Oliver

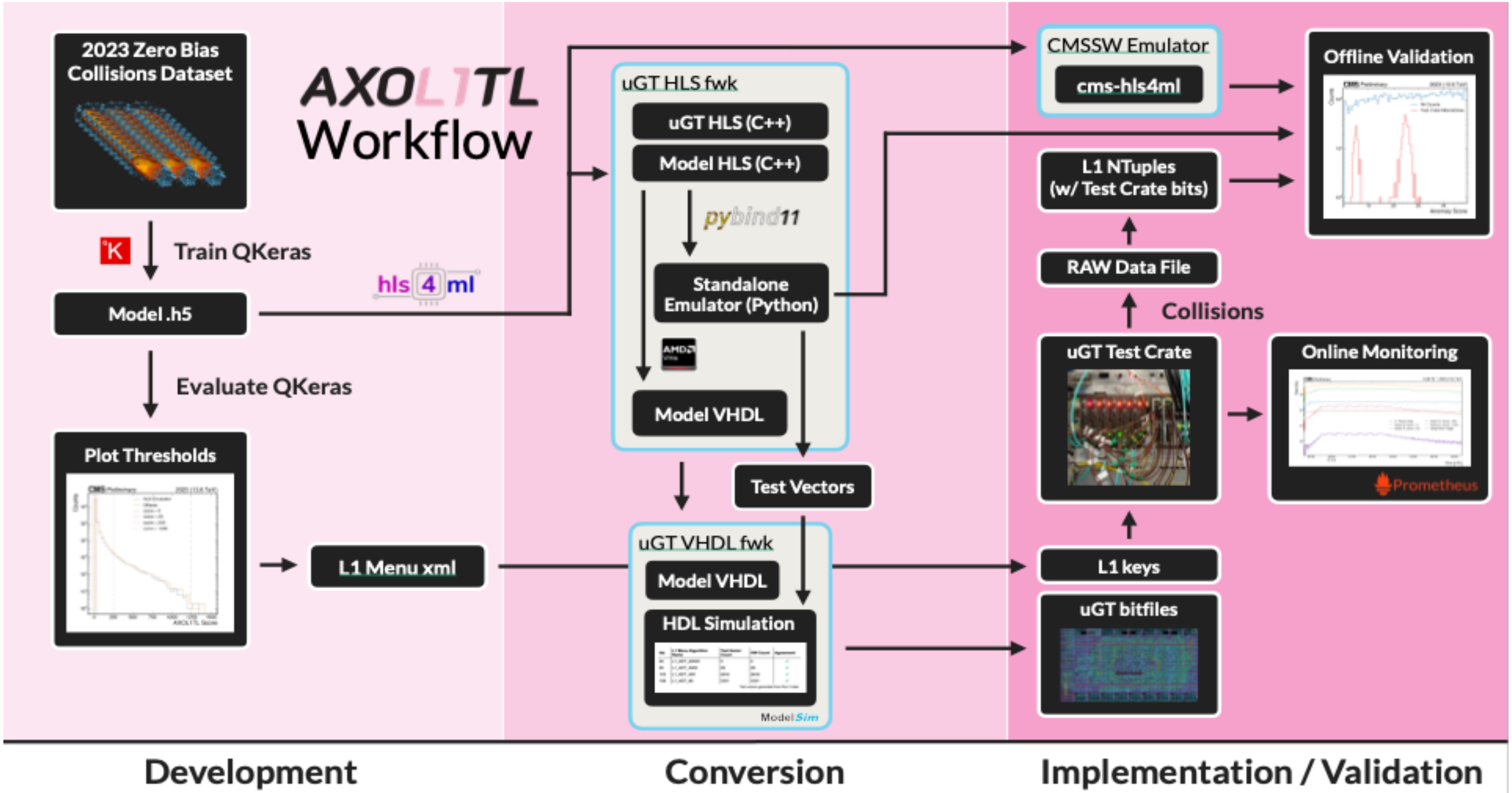
New Trigger strategies: anomaly detection

- What if the selections we've been making in the trigger are wrong? We could be missing the New Physics
- Anomaly Detection method proposed to search for New Physics in a model agnostic / unbiased way
  - Train a Variational AutoEncoder on unbiased data (background + ε new physics), trigger events with a high loss: anomalies
- Tiny VAE translated to FPGA with **hls4ml** requiring 50 ns latency for integration with Run 3 system
- Tested in 'safe mode' without triggering CMS in 2023, aiming to take data in 2024



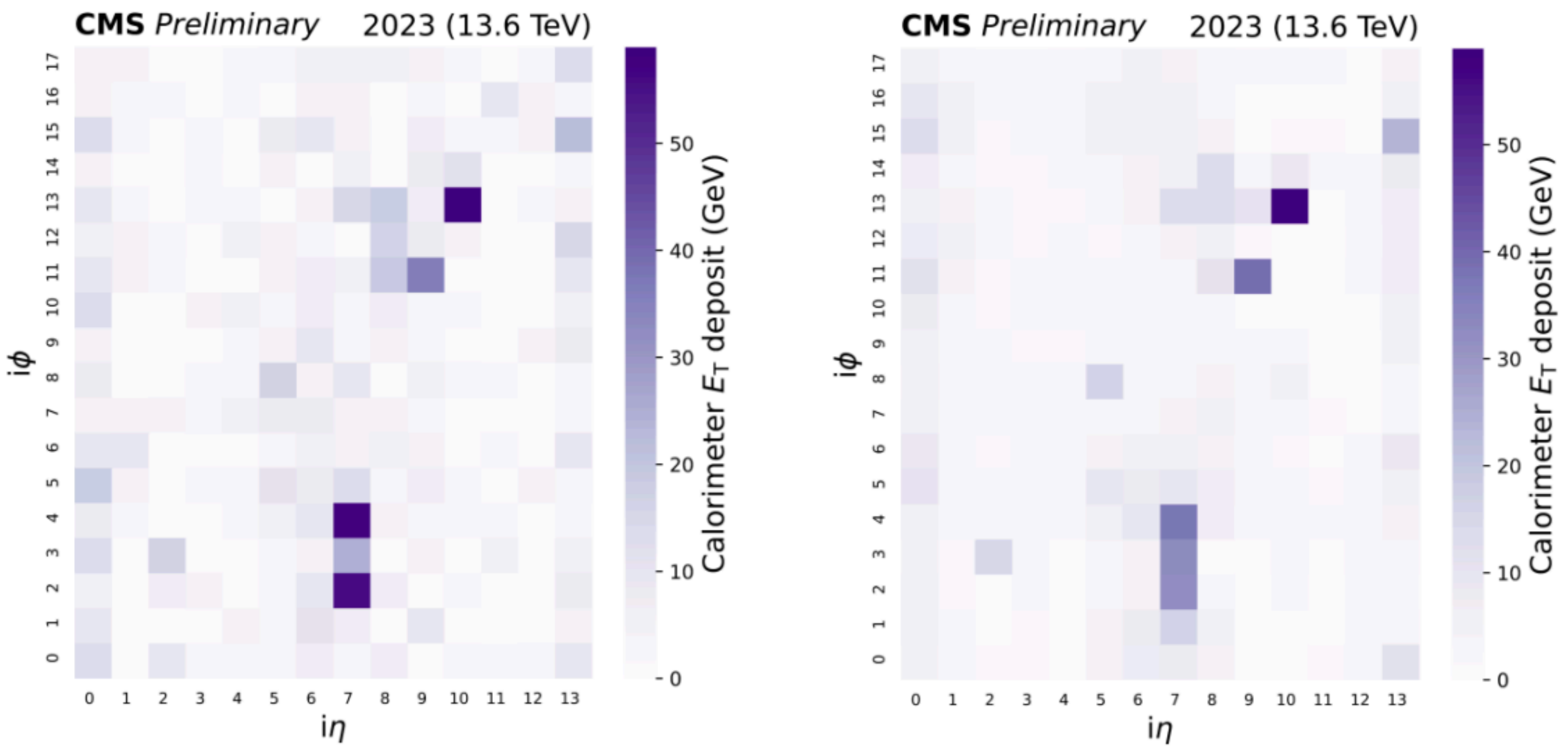


Anomaly Detection Tech Workflow

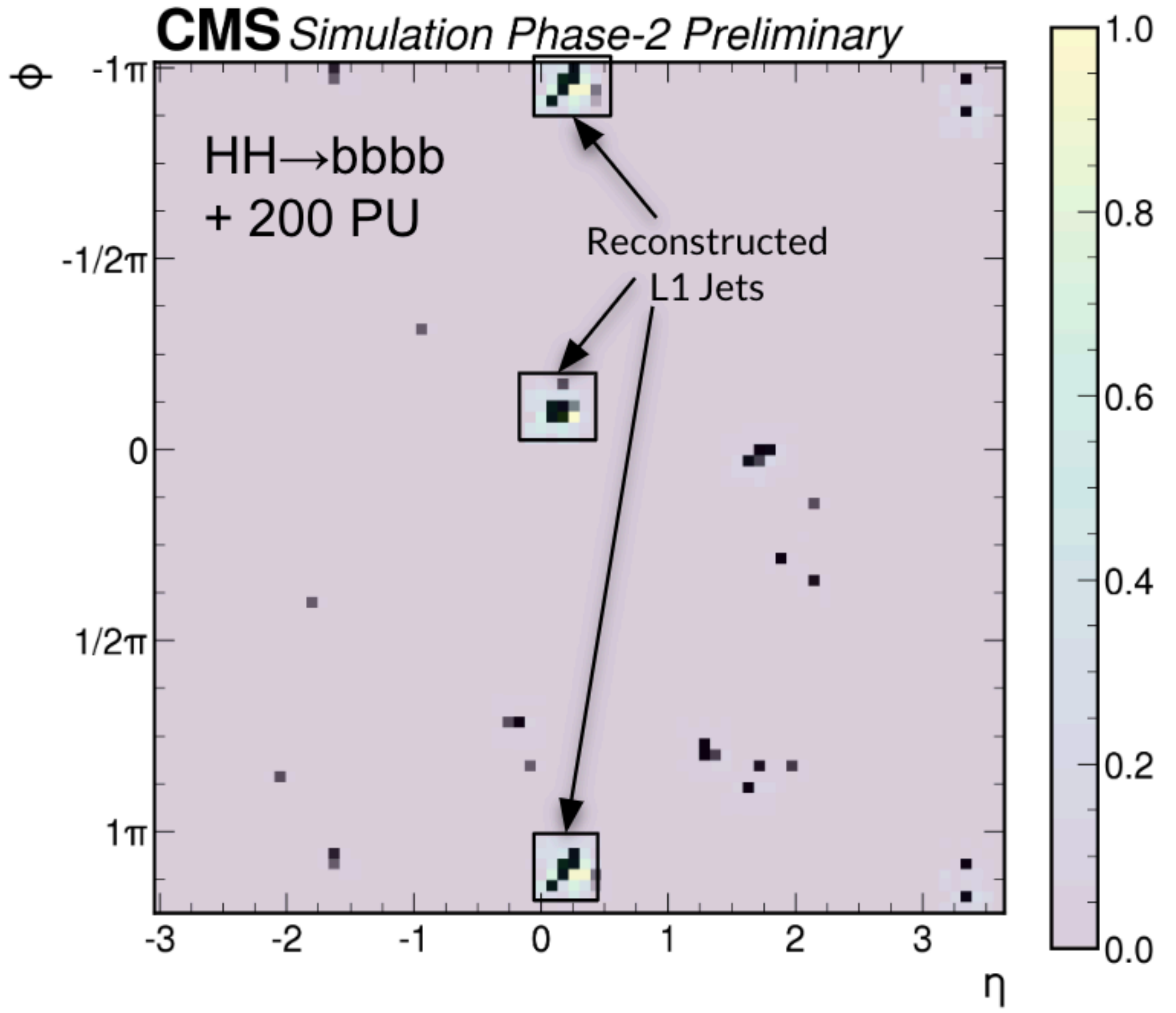


ML at CMS L1T: from Run 3 to Phase 2

- In addition to axol1tl there are three other projects using ML for triggering CMS in Run 3:
  - In development: CICADA, complementary anomaly detection technique using low level data from the calorimeter (top image)
  - Tested in 'safe mode': topology trigger (topol1tl): classifier models for better triggering of specific final states
  - In production: p_T regression of muons in the endcap
- For Phase 2 we expect ML to be well embedded into L1T
  - Significantly more powerful compute, 3x latency budget
  - Around 20 projects (NNs, BDTs) in development
  - Accounting for **25 billion ML inferences per second**



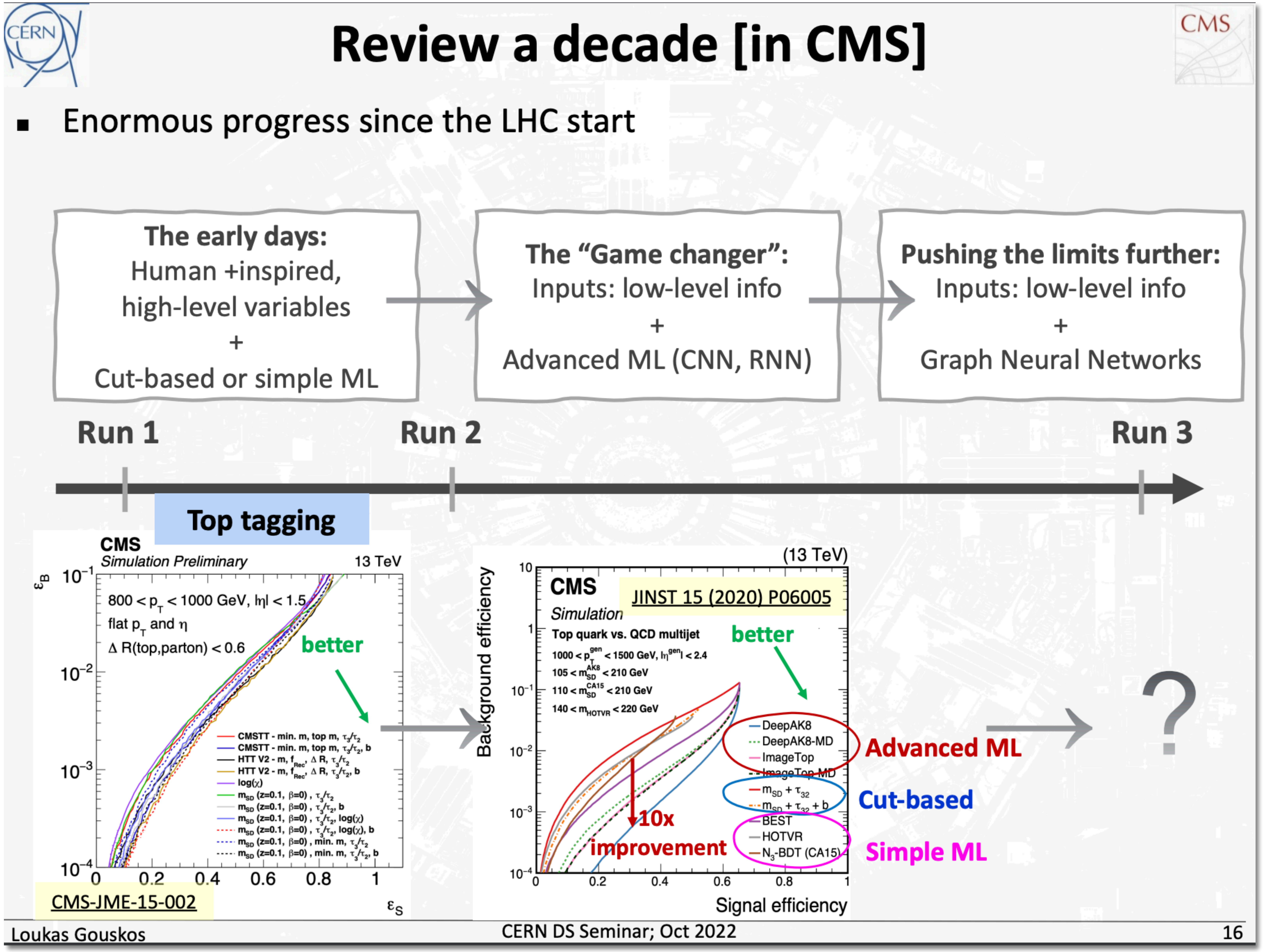
CMS-DP-2023-086



Jet Tagging

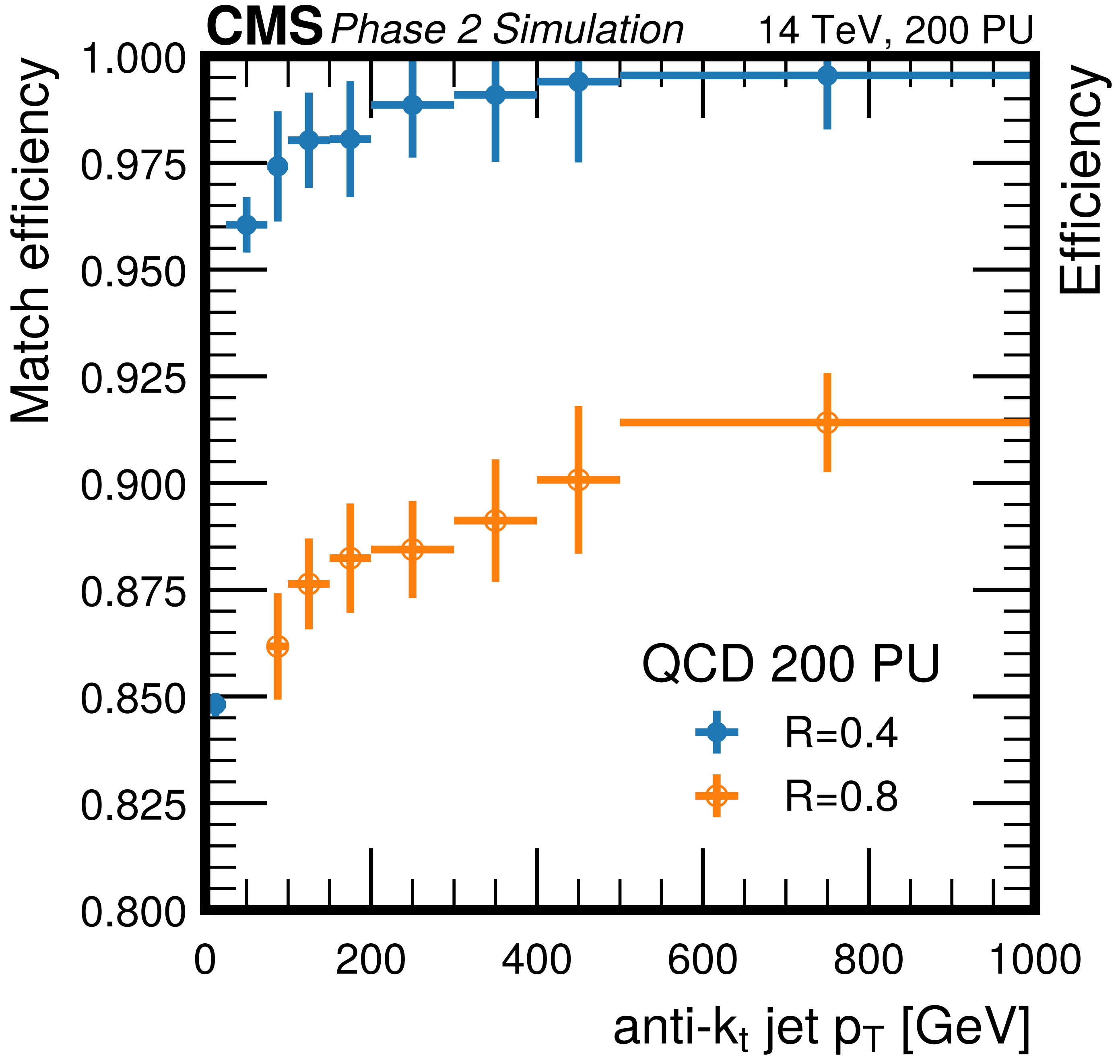
Loukas Gouskos, CERN DS Seminar

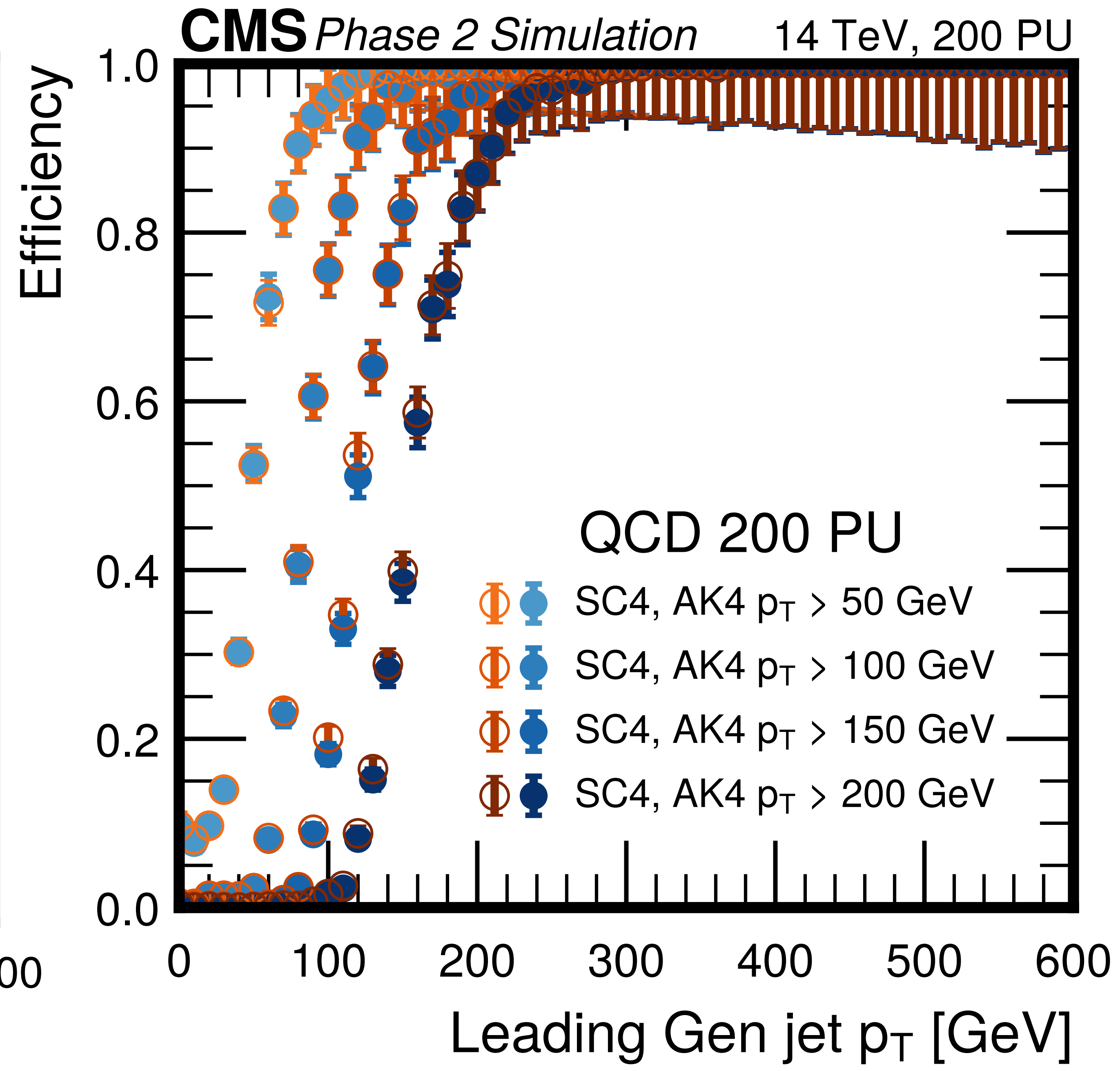
- Jet tagging: classifying the particle flavour that initiated a jet
- Developments in tagging at L1T following huge progress in tagging for offline reconstruction
- For Phase 2 will have similar “low-level” information available: particles and their properties
- Developing support for the same kinds of cutting edge ML models
  - Graph NNs, DeepSets
  - Using the best practice techniques previously described: tiny models, quantized, pruned

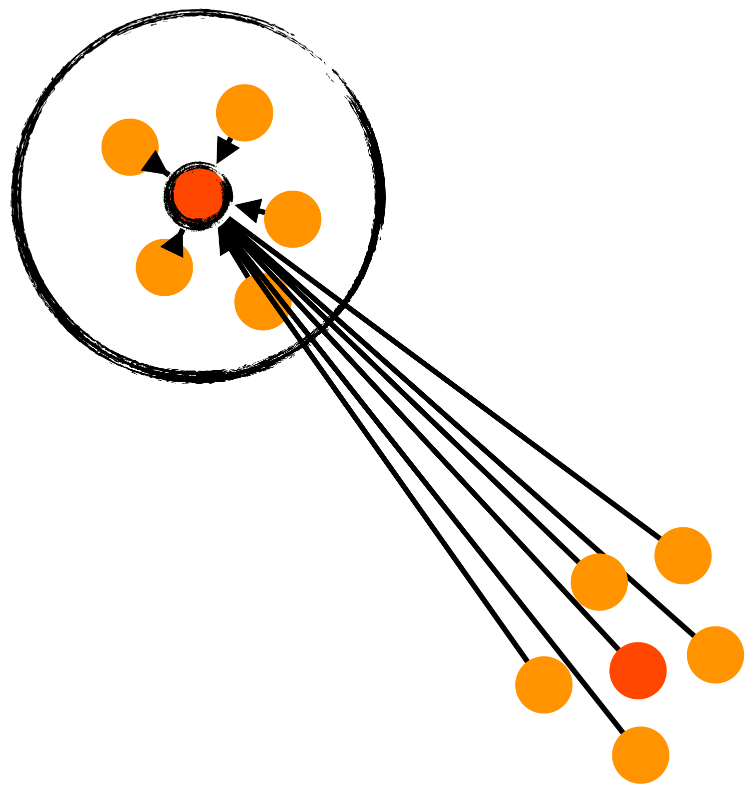


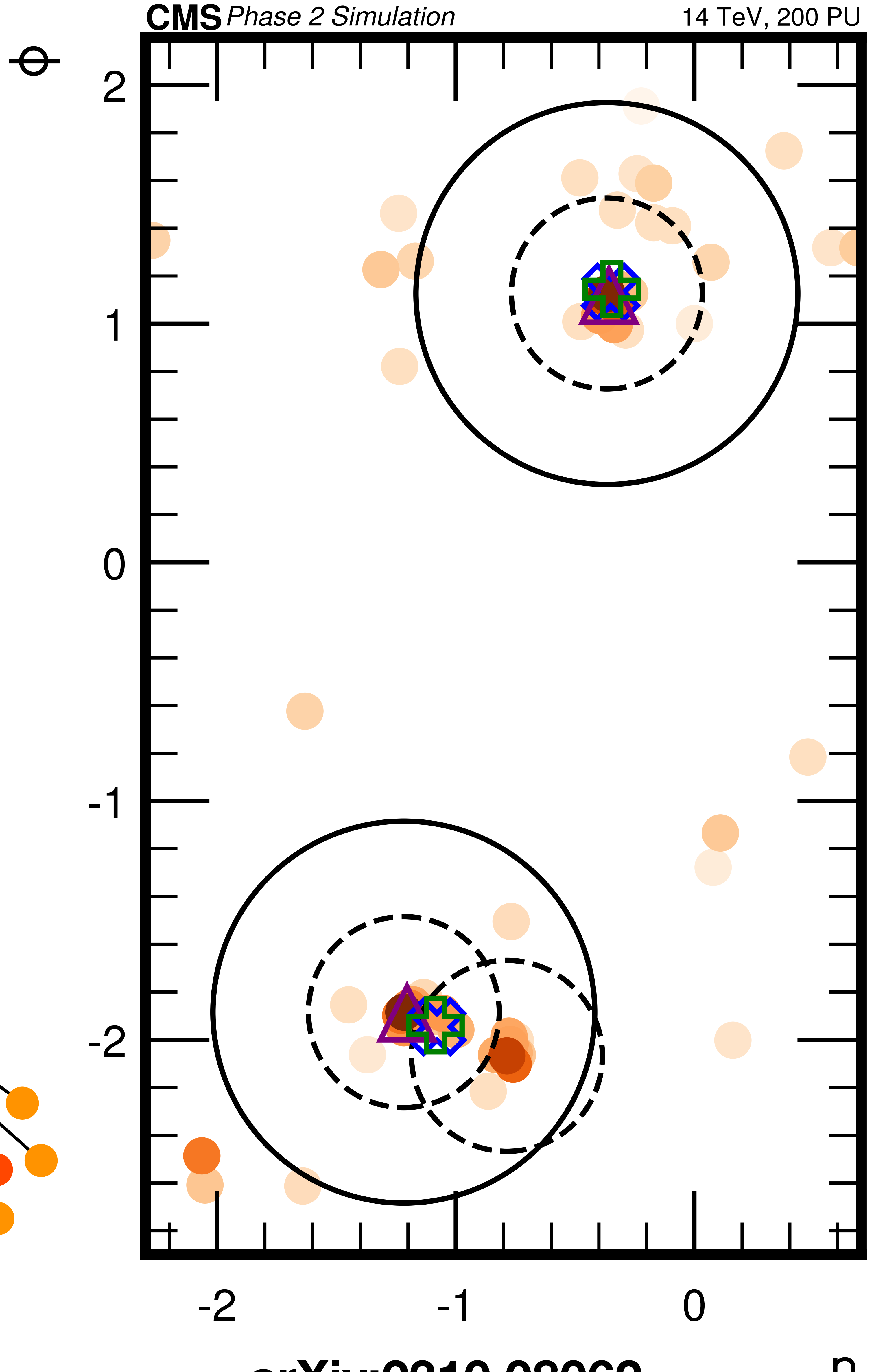
Jet Reconstruction at CMS L1T

- CMS Phase 2 will perform Particle Flow in the Level 1 Trigger for the first time
- Now we can cluster particles into jets and tag the flavour of those jets
- First we develop a fast and performant jet reconstruction for FPGA
  - Latency 750 ns for 12 jets, performance close to anti-k_T on the same particles
  - Both small & large radius reconstructions for unmerged & merged jets
- Jet constituents are buffered and provided for downstream processing: taggers









arXiv:2310.08062

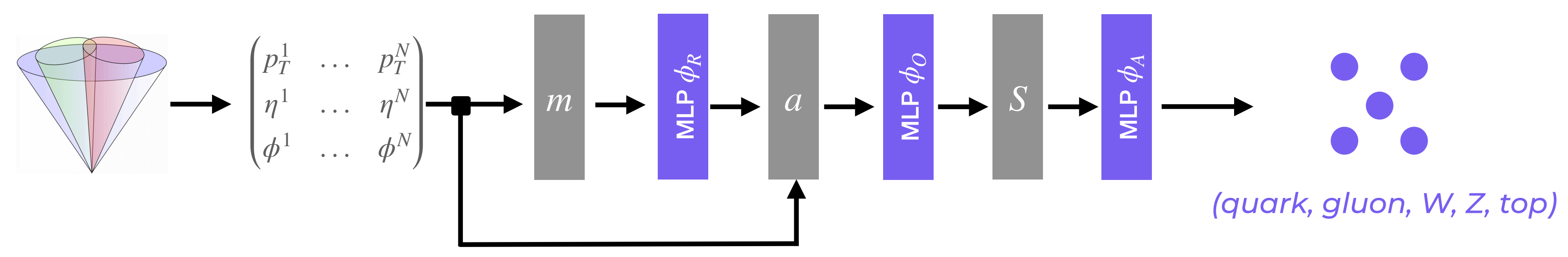
Jet Tagging Architectures for L1T

- Now we have the clustered particles in one place in the FPGA, we can send them to a Neural Network for tagging
- Which Neural Network model architectures performs well for jet tagging, and can we deploy it in an FPGA with O(100) ns latency and O(100) MHz throughput?

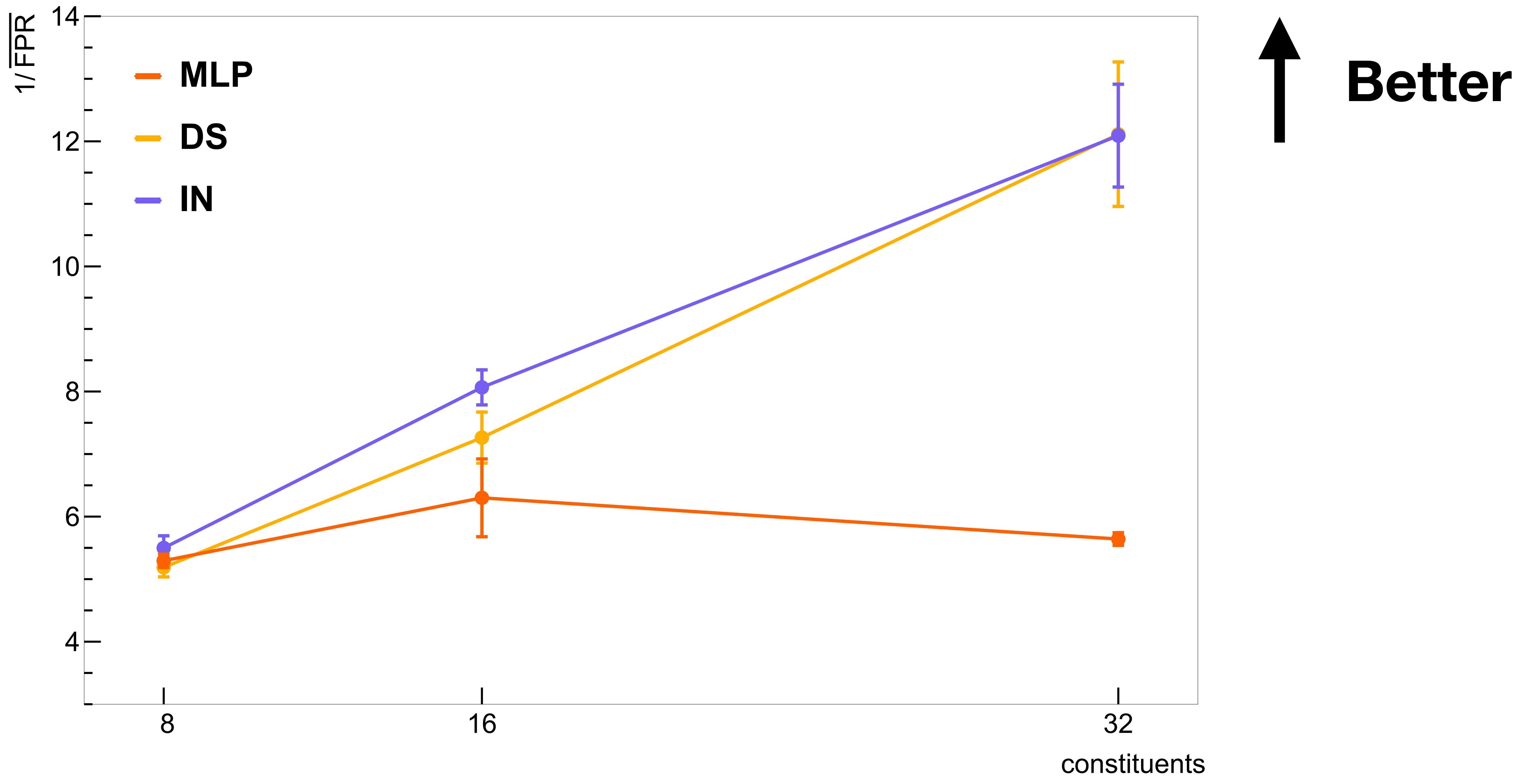
a) Multilayer Perceptron MLP

b) Deep Sets DS

c) Interaction Network IN



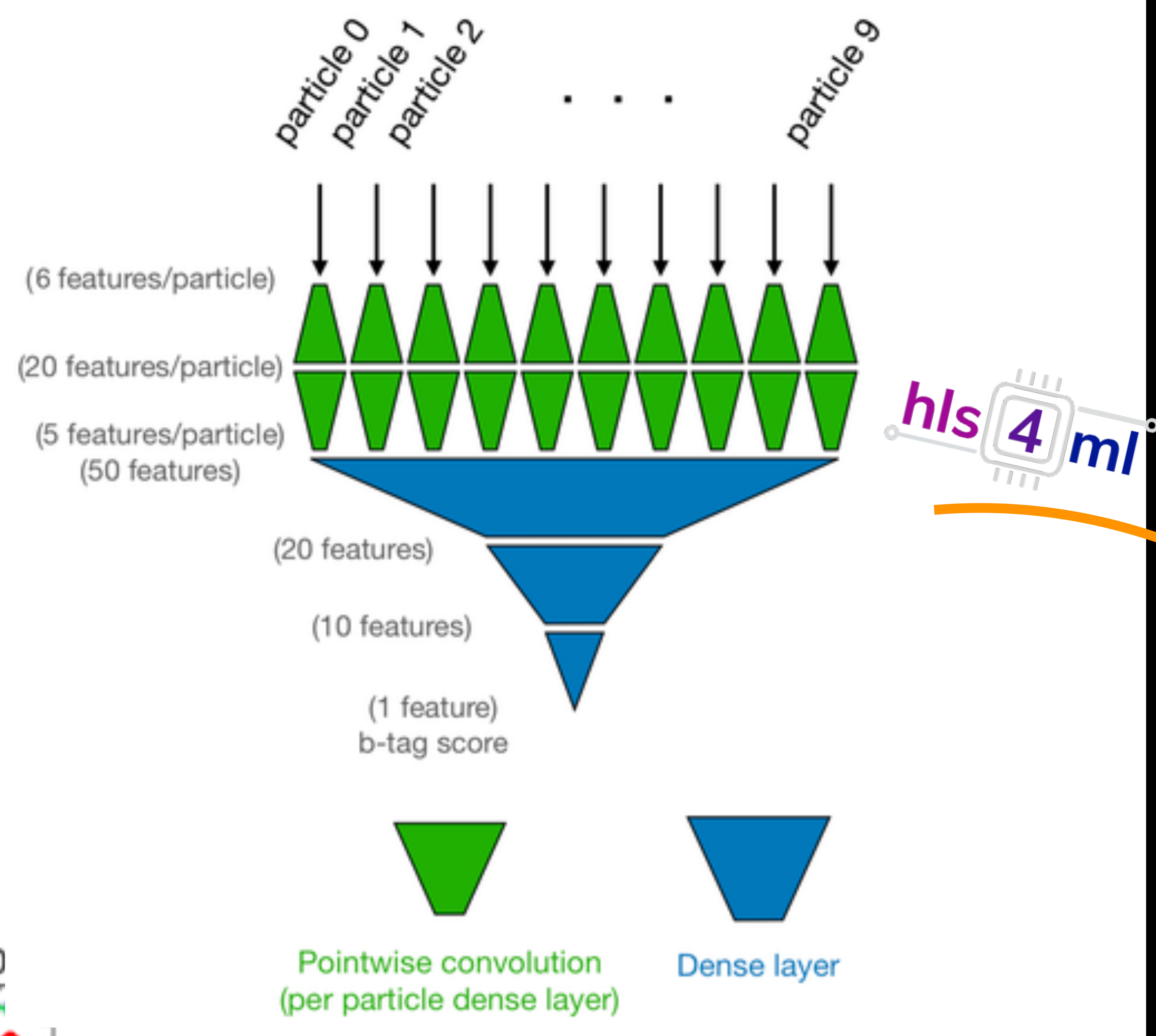
arXiv:2402.01876

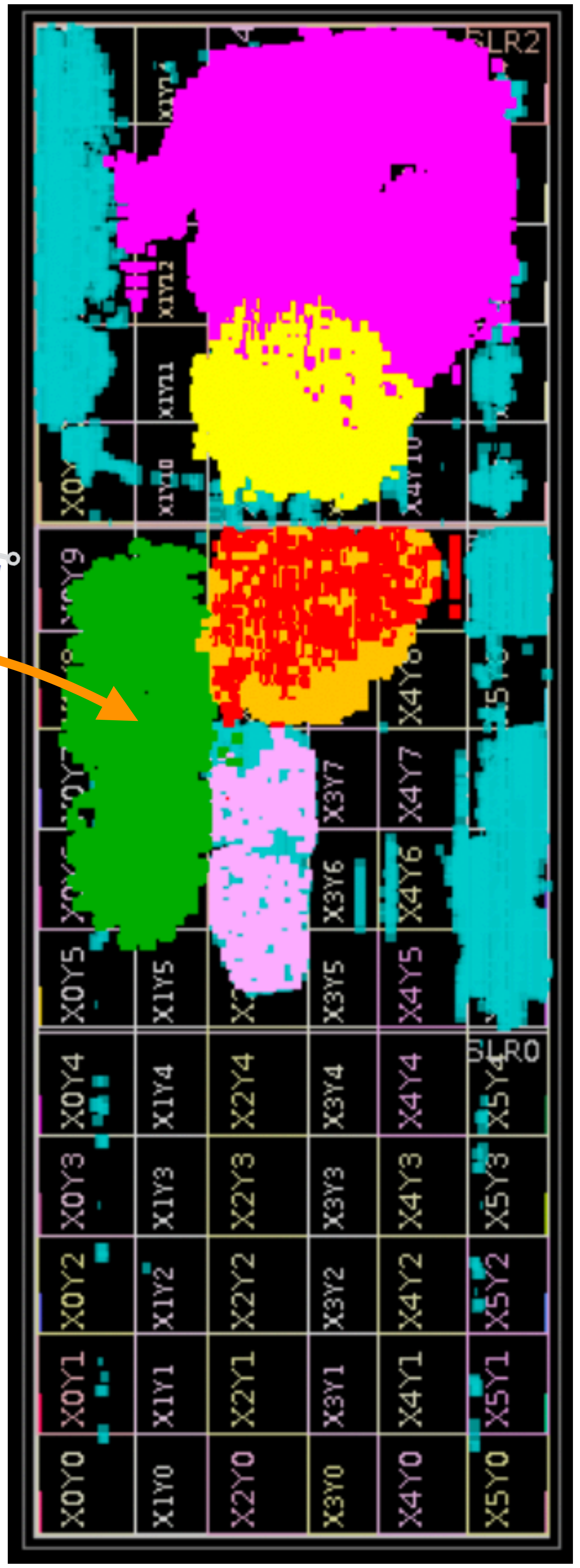


Architecture	Constituents	RF	Latency [ns] (cc)	II [ns] (cc)	DSP	LUT	FF	BRAM18
MLP	8	1	105 (21)	5 (1)	262 (2.1%)	155,080 (9.0%)	25,714 (0.7%)	4 (0.1%)
	16	1	100 (20)	5 (1)	226 (1.8%)	146,515 (8.5%)	31,426 (0.9%)	4 (0.1%)
	32[a]	1	105 (21)	5 (1)	262 (2.1%)	155,080 (7.2%)	25,714 (0.7%)	4 (0.1%)
DS	8	2	95 (19)	15 (3)	626 (5.1%)	386,294 (22.3%)	121,424 (3.5%)	4 (0.1%)
	16	4	115 (23)	15 (3)	555 (4.5%)	747,374 (43.2%)	238,798 (6.9%)	4 (0.1%)
	32[a]	8	130 (26)	10 (2)	434 (3.5%)	903,284 (52.3%)	358,754 (10.4%)	4 (0.1%)
IN	8	2	160 (32)	15 (3)	2,191 (17.8%)	472,140 (27.3%)	191,802 (5.5%)	12 (0.2%)
	16	4	180 (36)	15 (3)	5,362 (43.6%)	1,387,923 (80.3%)	594,039 (17.2%)	52 (1.9%)
	32[a]	8	205 (41)	15 (3)	2,120 (17.3%)	1,162,104 (67.3%)	761,061 (22.0%)	132 (2.5%)

Jet Tagging at CMS L1T

- Jets are first reconstructed with the described algorithm
- Using DeepSets-inspired architecture for b tagging
  - Relies on track displacement measurement from L1 track finder
- Tiny model improves trigger reach to important final states (HH→bbbb shown)
- Fits in FPGA (right) and total latency (jet reco + tagging) less than 1 μs

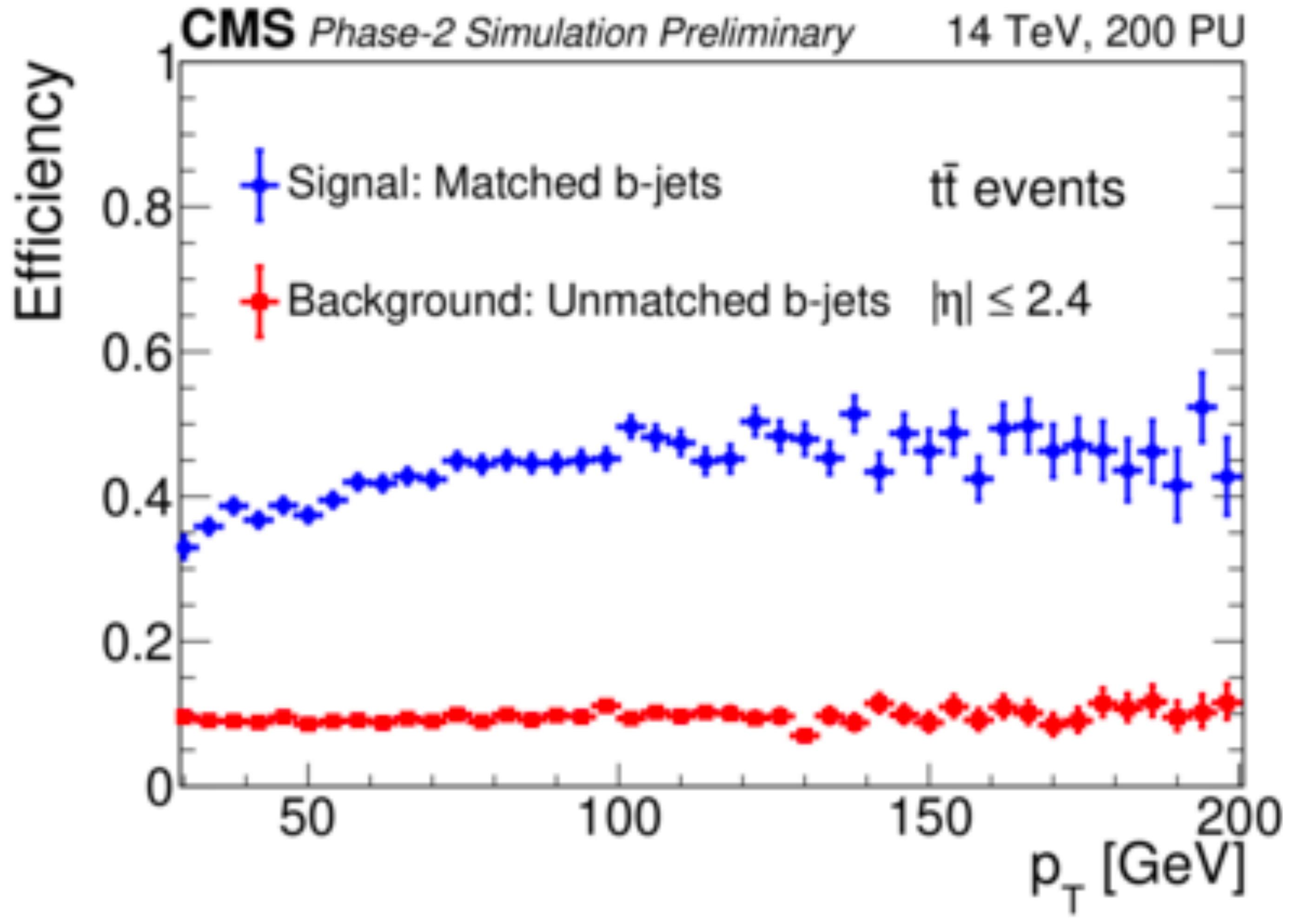


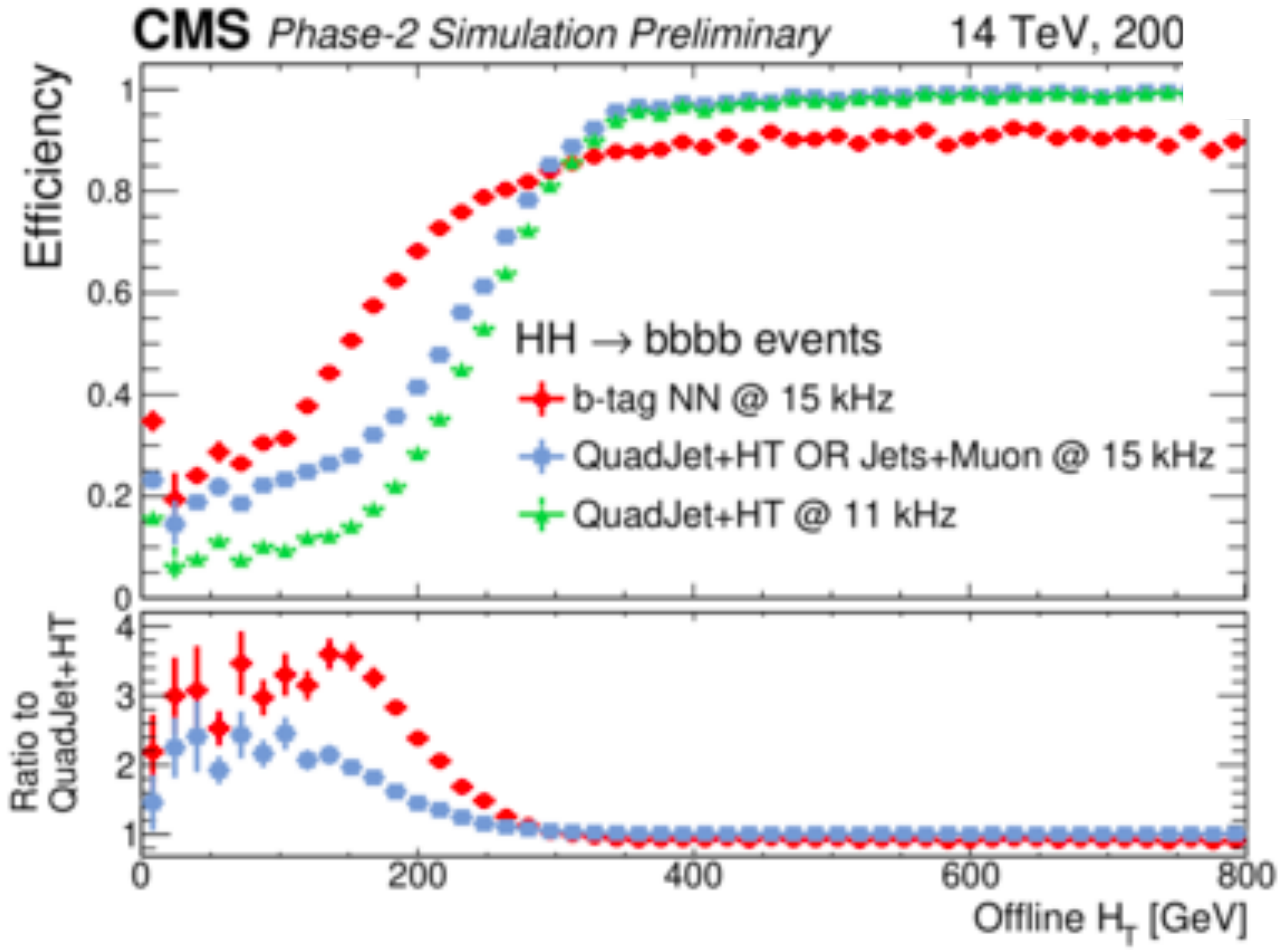


Particle Receiving
Jet Constituent Finding
Jet Axis Computation
Sorting, Buffering
B tagging Neural Network

AMD VU9P

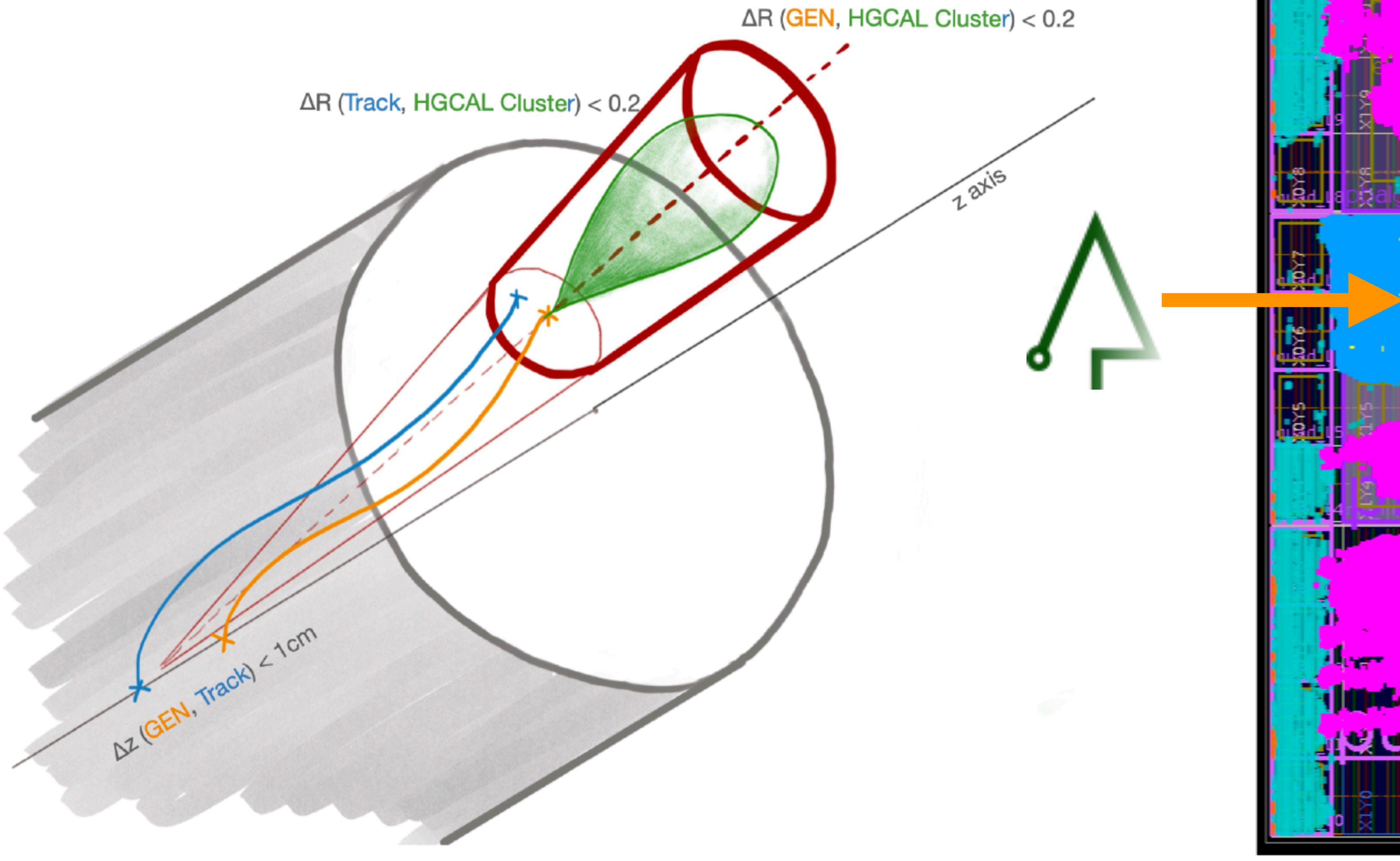
CMS-DP-2022-021

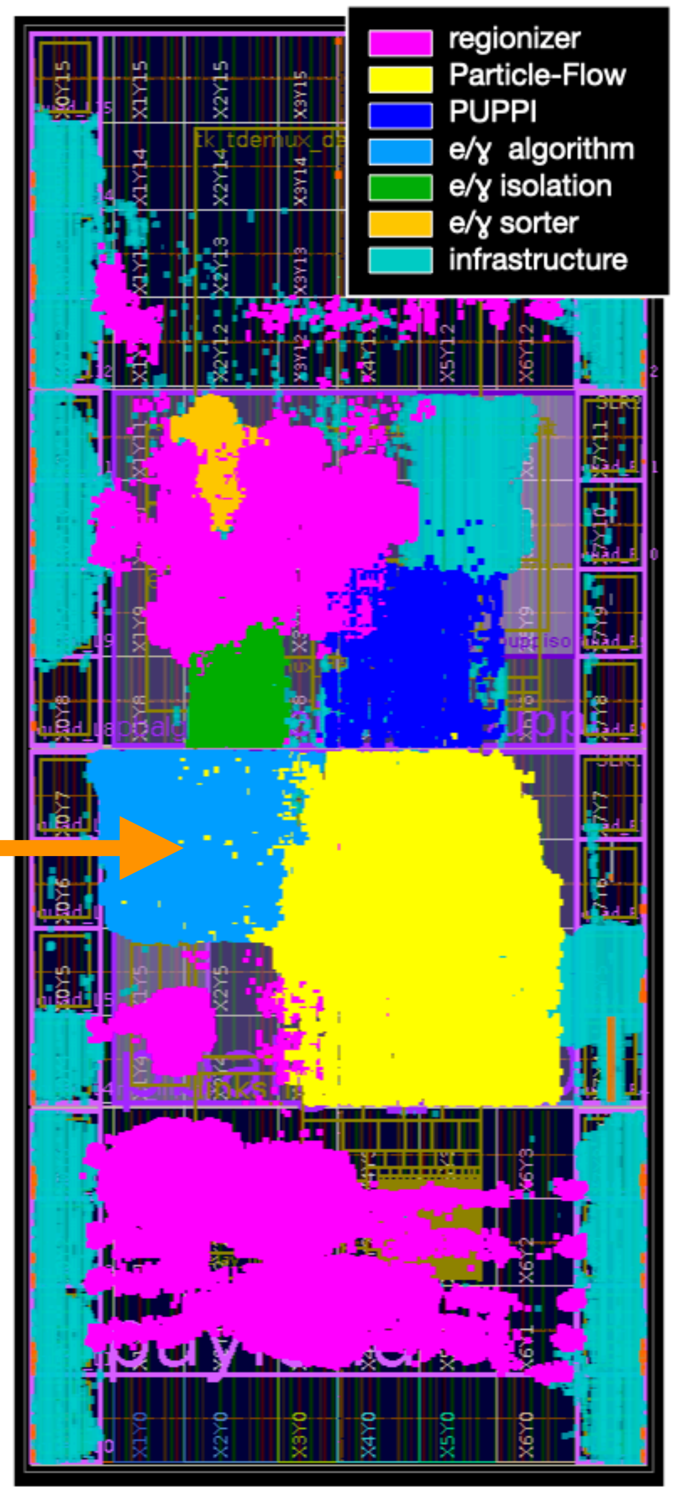




CMS Phase 2 L1T electron ID

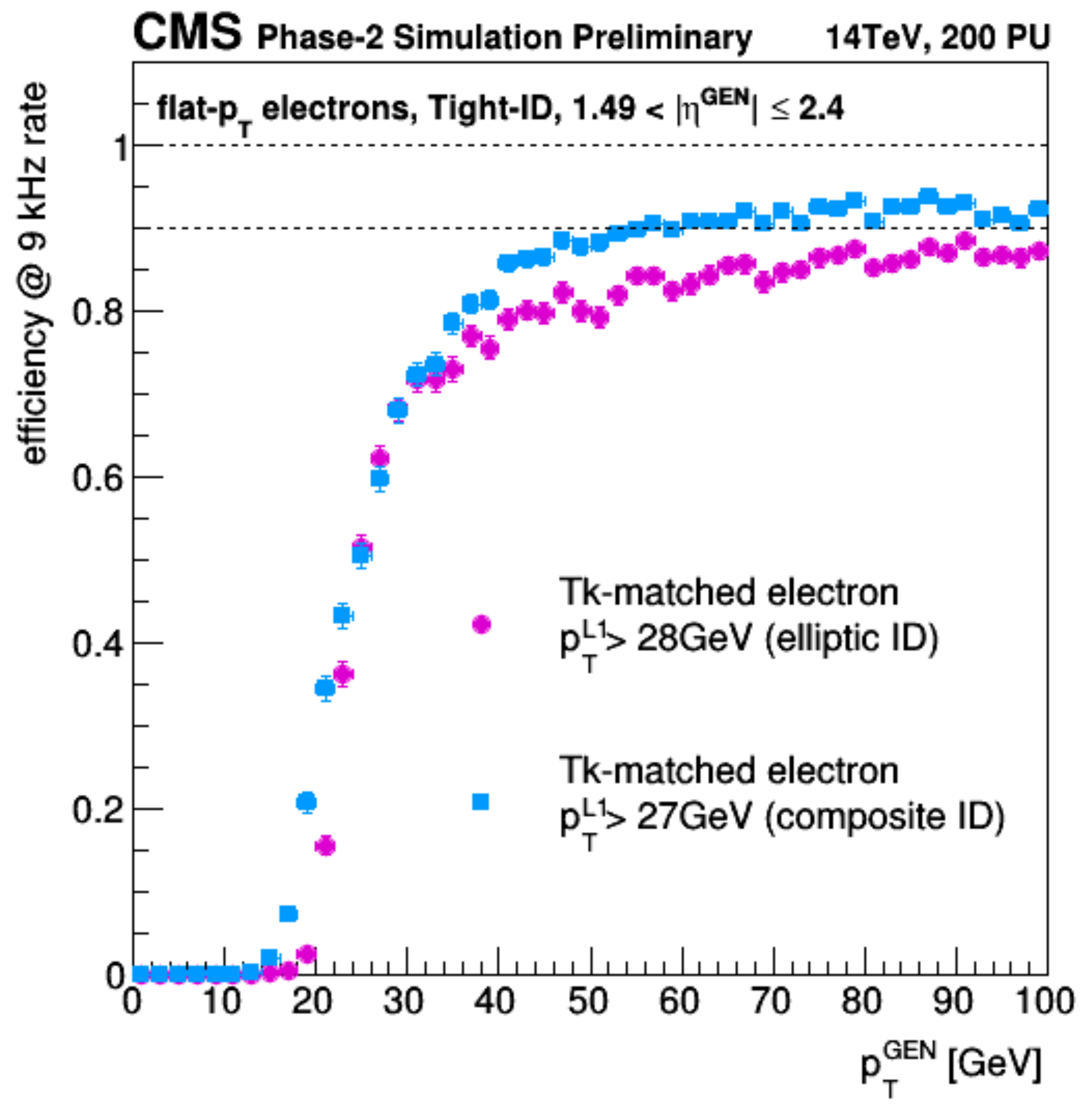
- Electrons will be reconstructed by linking a track with a calorimeter cluster
- Neither reconstruction is perfect, and electrons bremsstrahlung
- **Baseline kinematic approach** used distance and p_T compatibility to make a link
- **New BDT approach** first makes a loose kinematic selection, then uses ML to predict probability that the track & cluster both originated from an electron
- Improved electron reconstruction efficiency with new method (bottom left)
- xgboost for model training, **conifer** for inference
  - Tiny model with 10 trees & maximum depth 4
  - 10 parallel model copies to maintain electron reco rate
  - Well within system resource and latency envelope

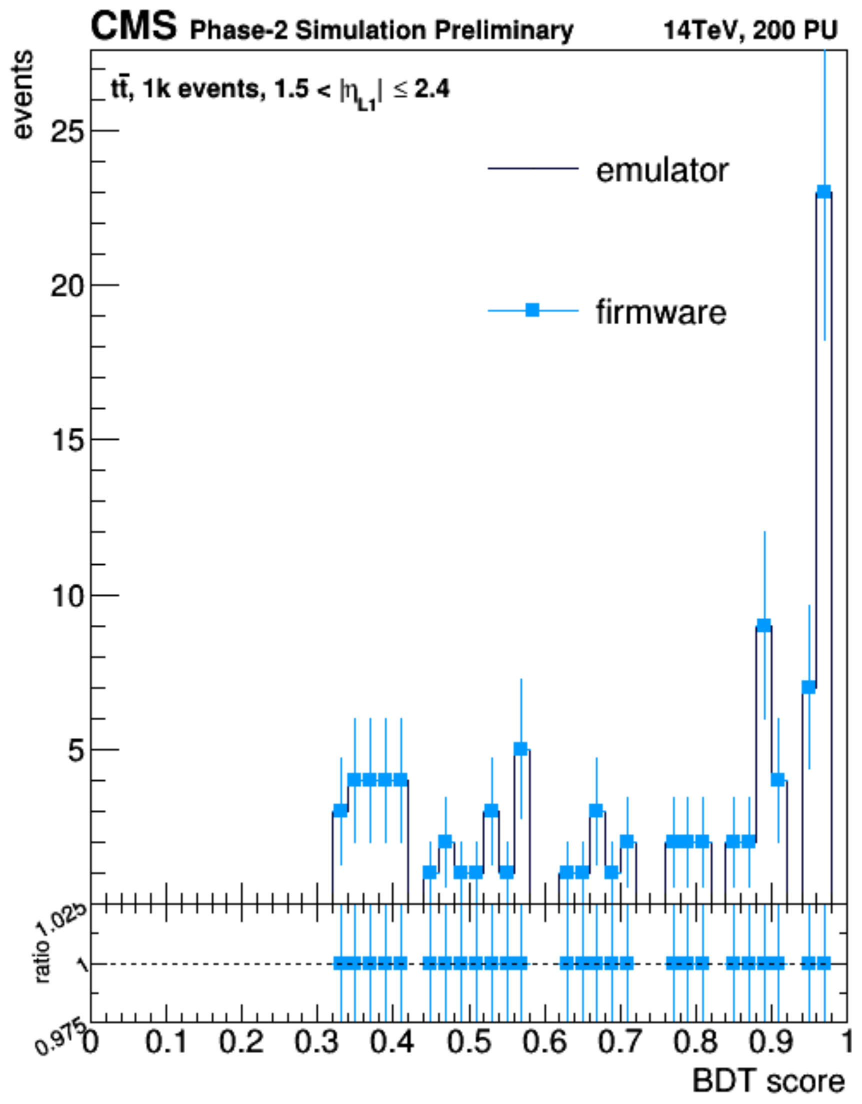




AMD VU13P

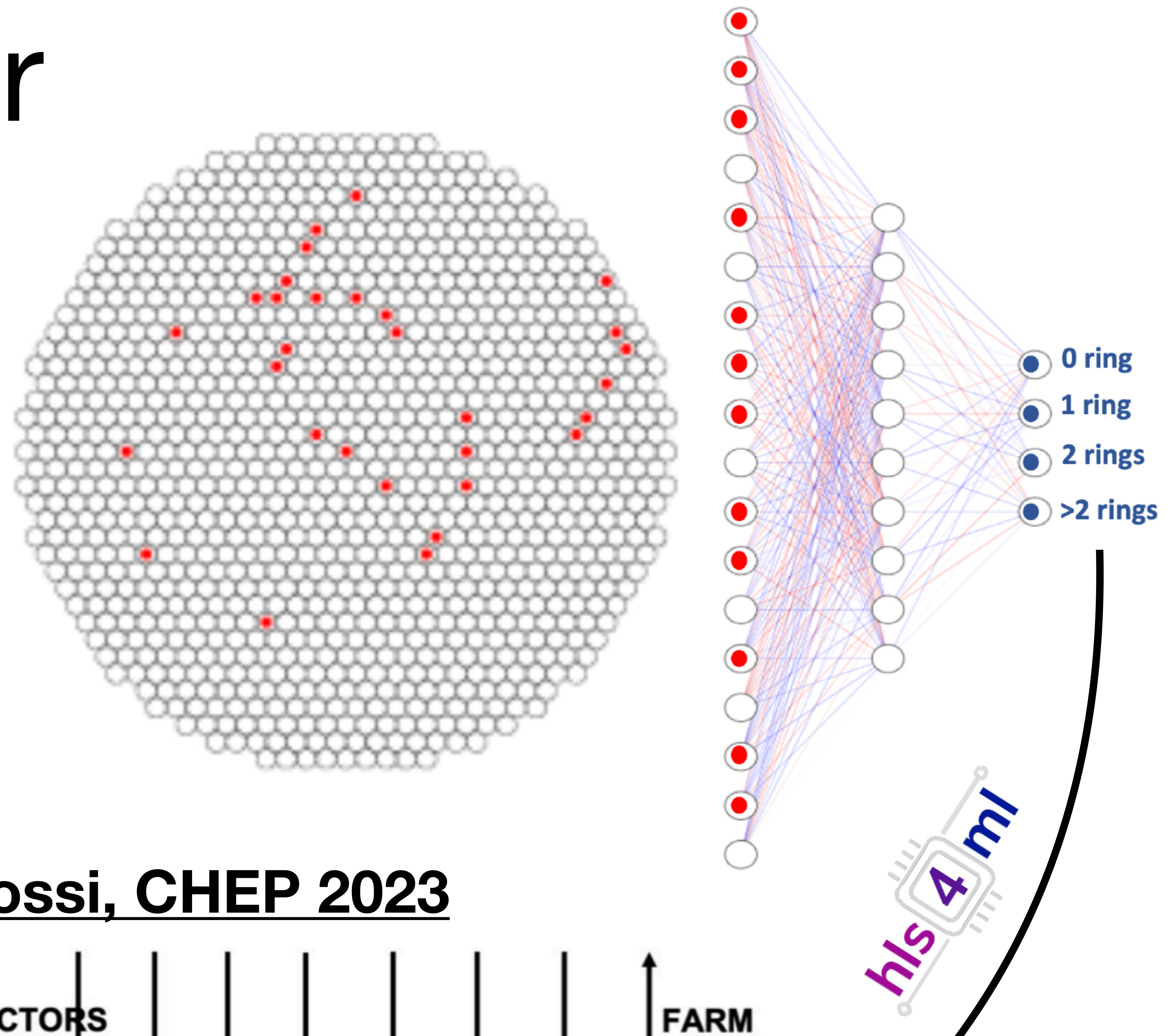
CMS-DP-2023-047



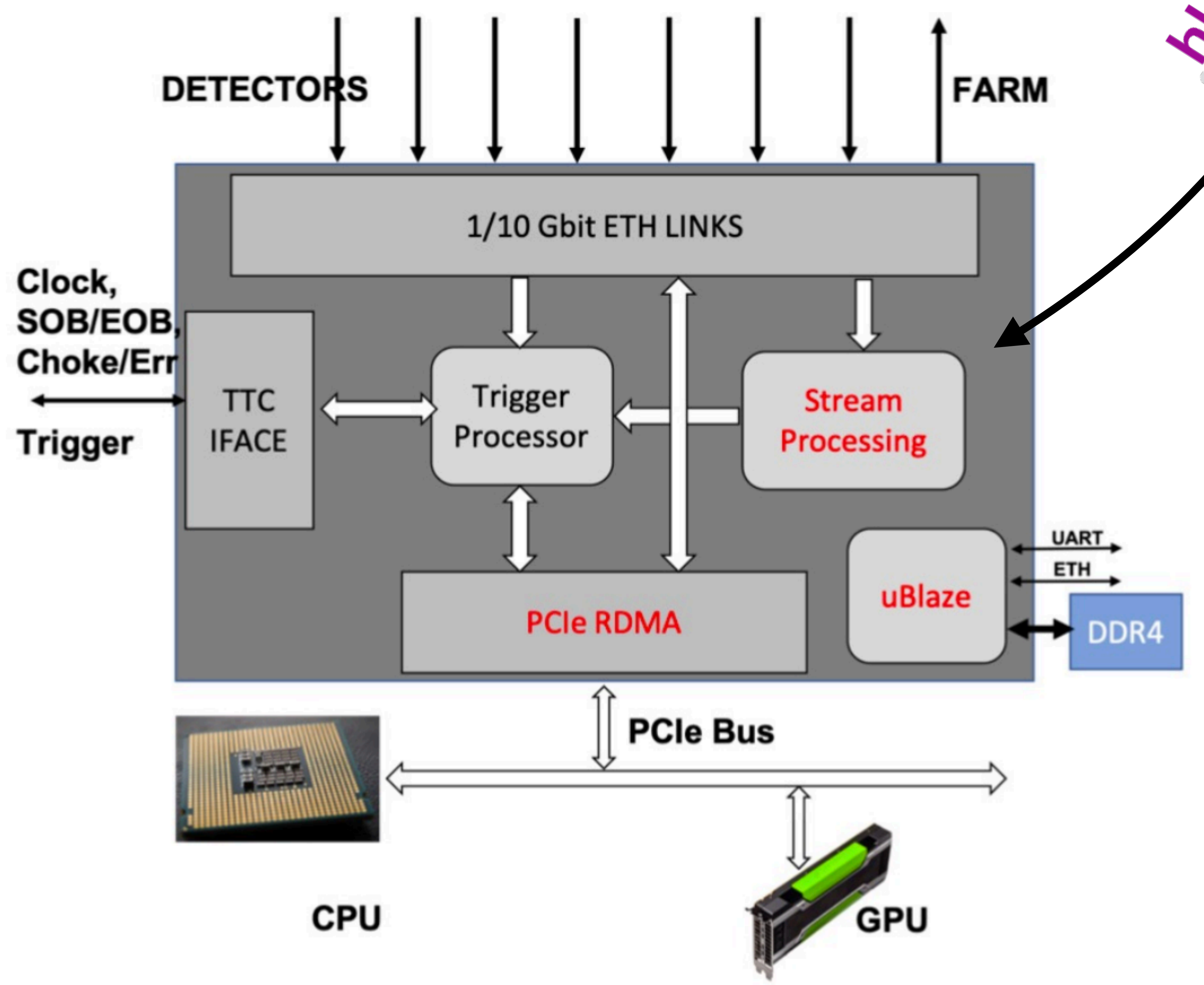


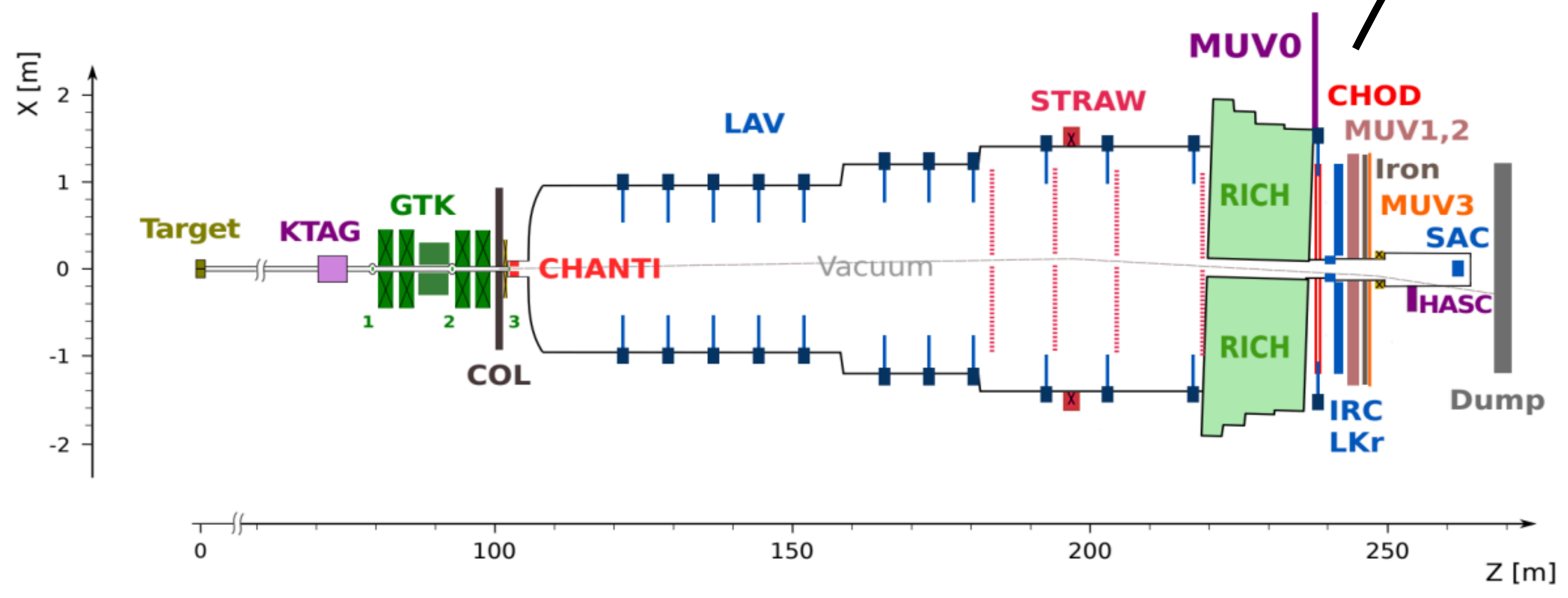
NA62 RICH Trigger

- NA62: fixed target experiment measuring ultra rare kaon decays
  - BR($K^+ \rightarrow \pi^+ \nu \bar{\nu}$) = (8.4±1.0) x 10^{-11}
- Hardware trigger reduces from 750 MHz beam to 1 MHz
- L0TP+ upgrade includes a DNN to classify number of rings in RICH detector
  - Using QKeras and hls4ml to deploy to FPGA
- Tested in parasitic mode (alongside classical triggers)



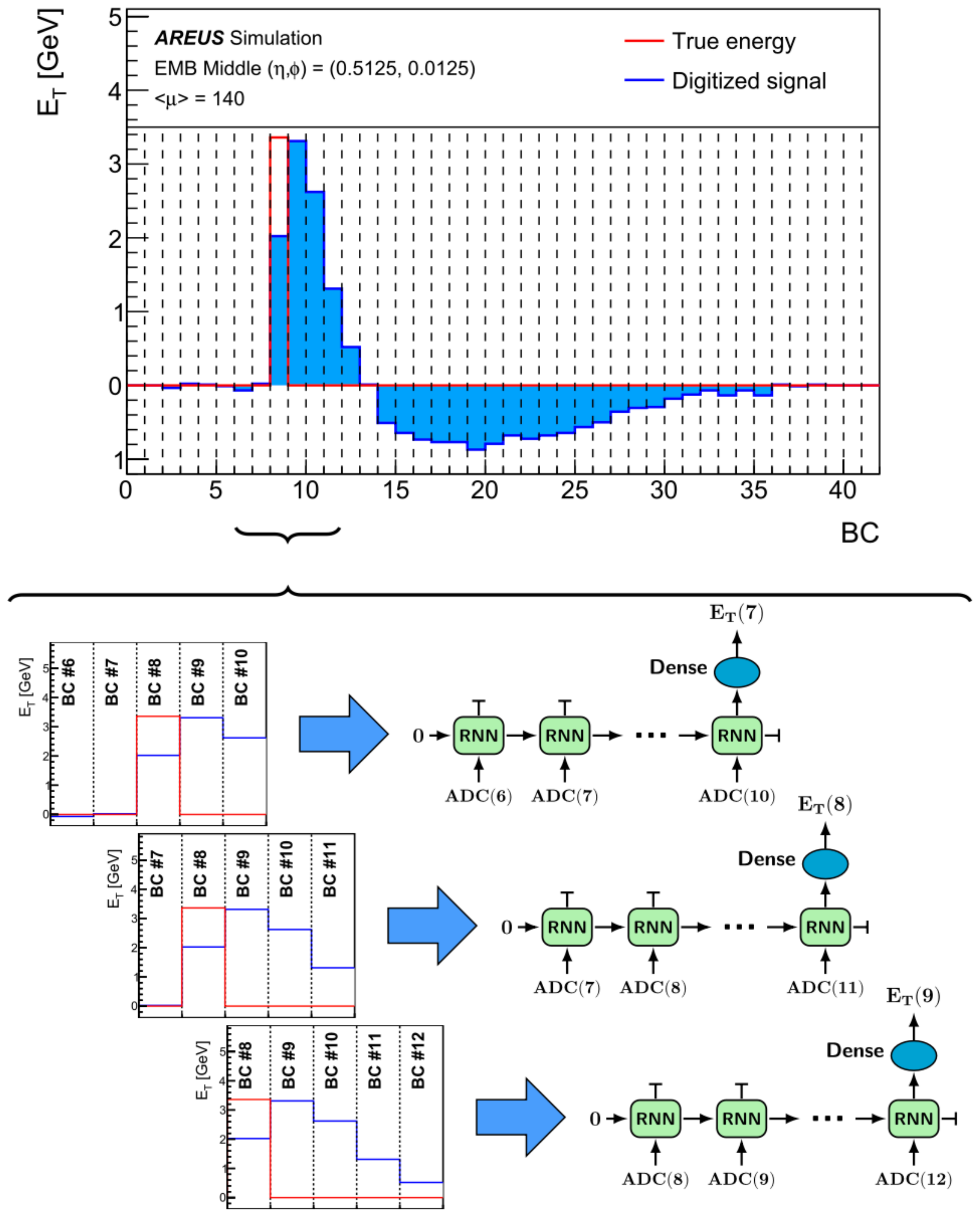
Cristian Rossi, CHEP 2023

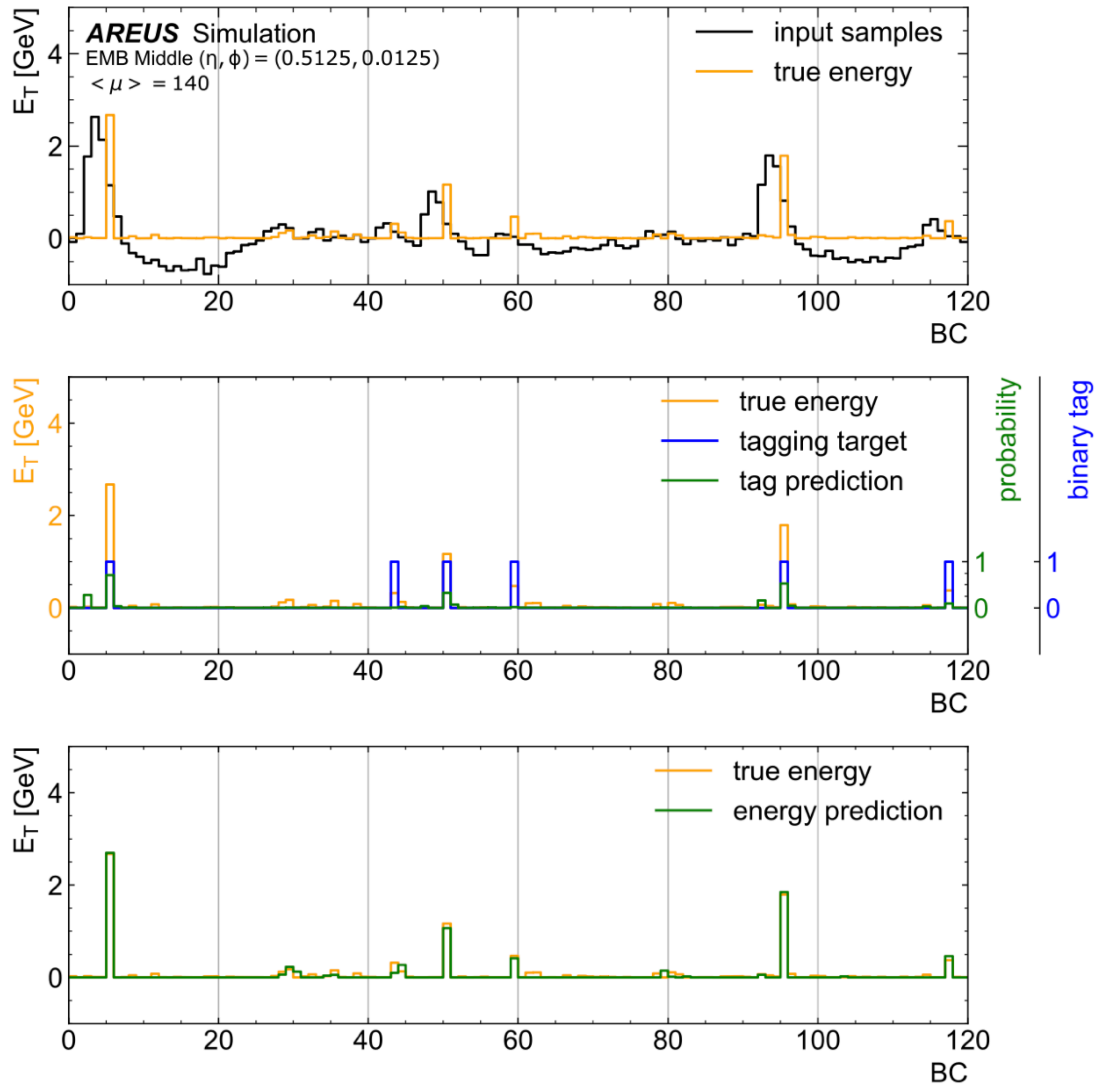




ATLAS LAr Calorimeter

- Convolutional and Recurrent Neural Networks for real-time energy reconstruction of ATLAS LAr Calorimeter for Phase 2
- Up to around 600 calorimeter channels processed by on device
- 200 ns latency of predictions
- Implemented on Intel FPGAs (previous examples all AMD)
    - Team contributed majorly to RNN and Intel implementations of hls4ml





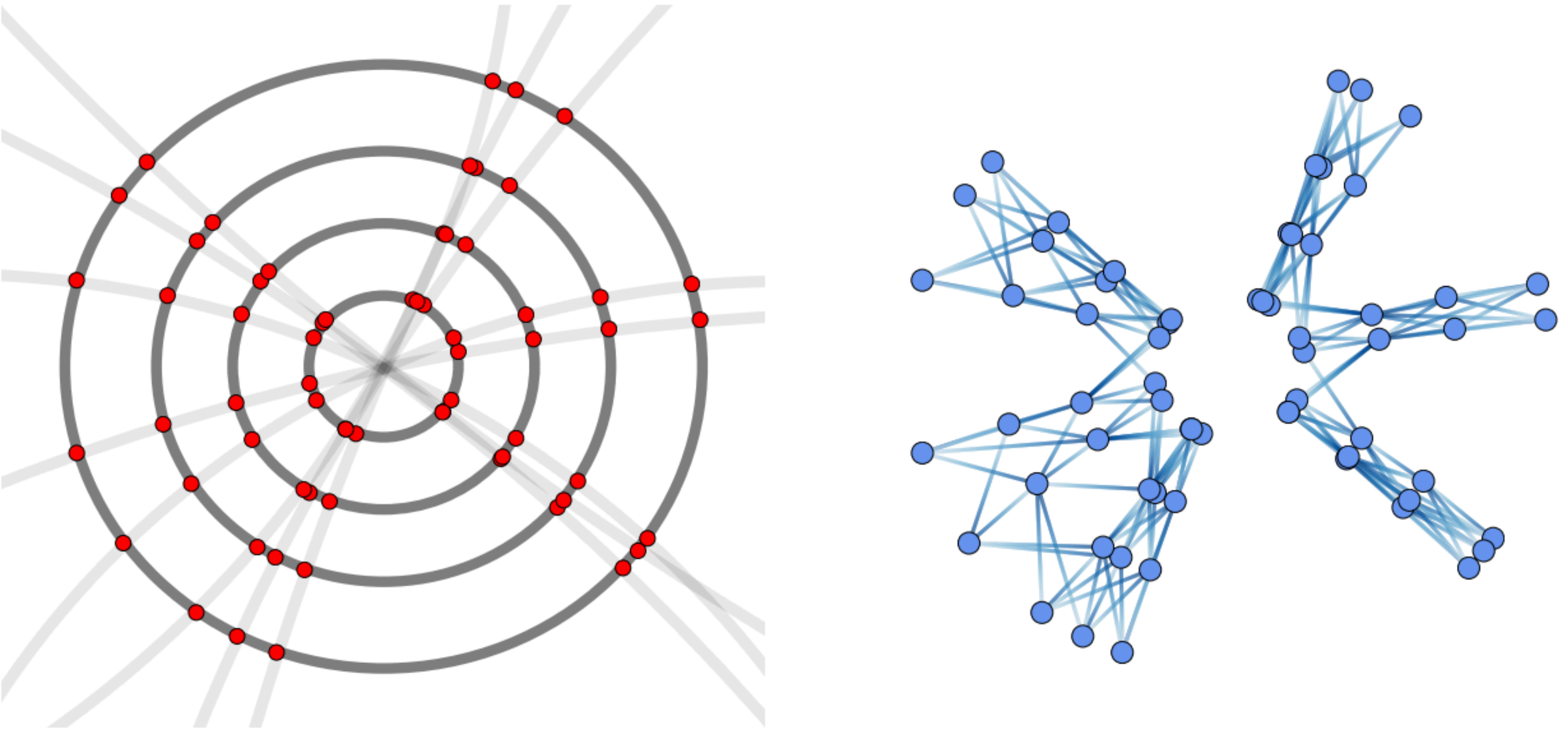
doi: 10.1007/s41781-021-00066-y

GraphNN Track Reconstruction

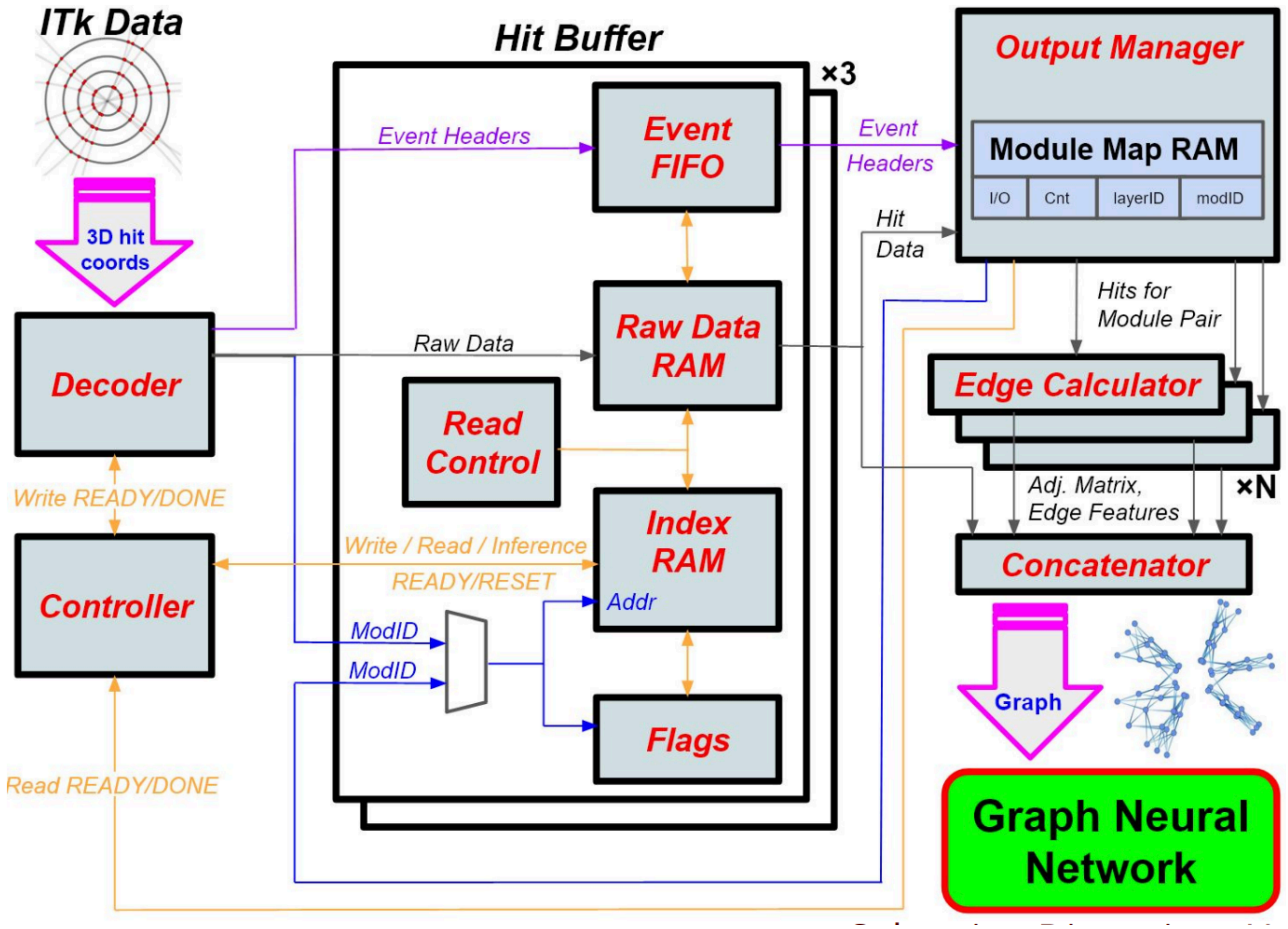
arXiv:2012.01249

doi: 10.3389/fdata.2022.828666

- Graph representation suits HEP data well: point-cloud data with edges and vertices

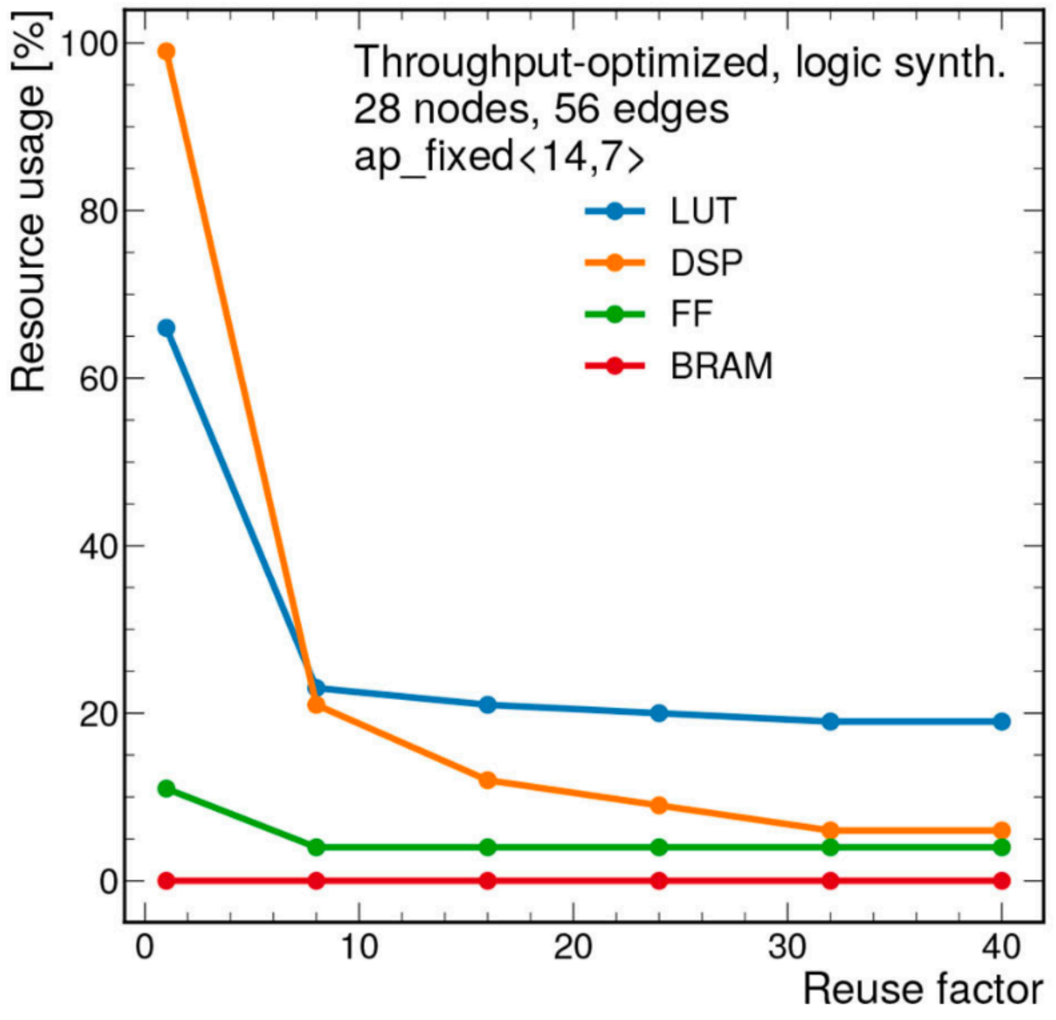


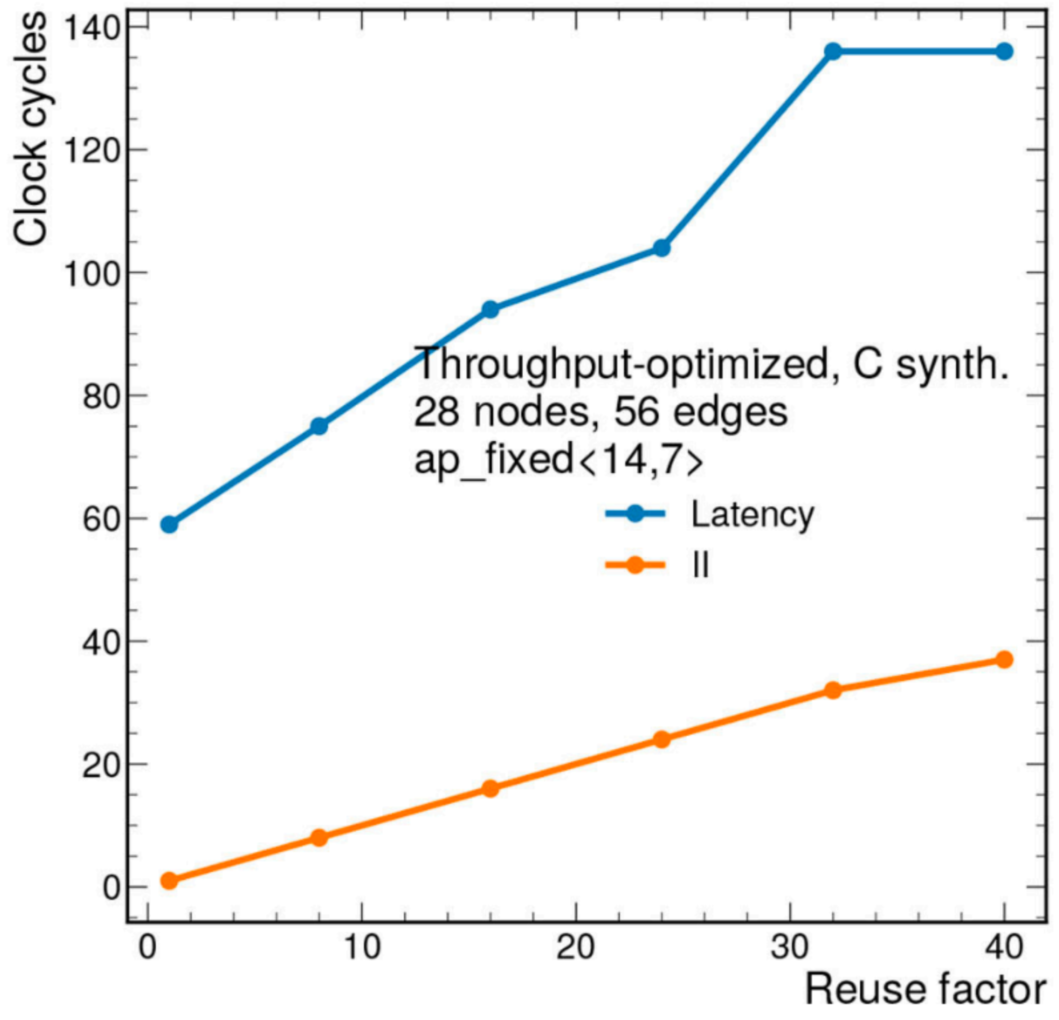
- Graph Neural Networks make predictions on graphs: track parameters from graph of hits
- Below: hardware architecture for ATLAS Inner Tracker reconstruction with GNN



S. Dittmeier

- Implementations for FPGA with hls4ml have been developed
- Graph NNs are expensive, scaling larger graphs is WIP

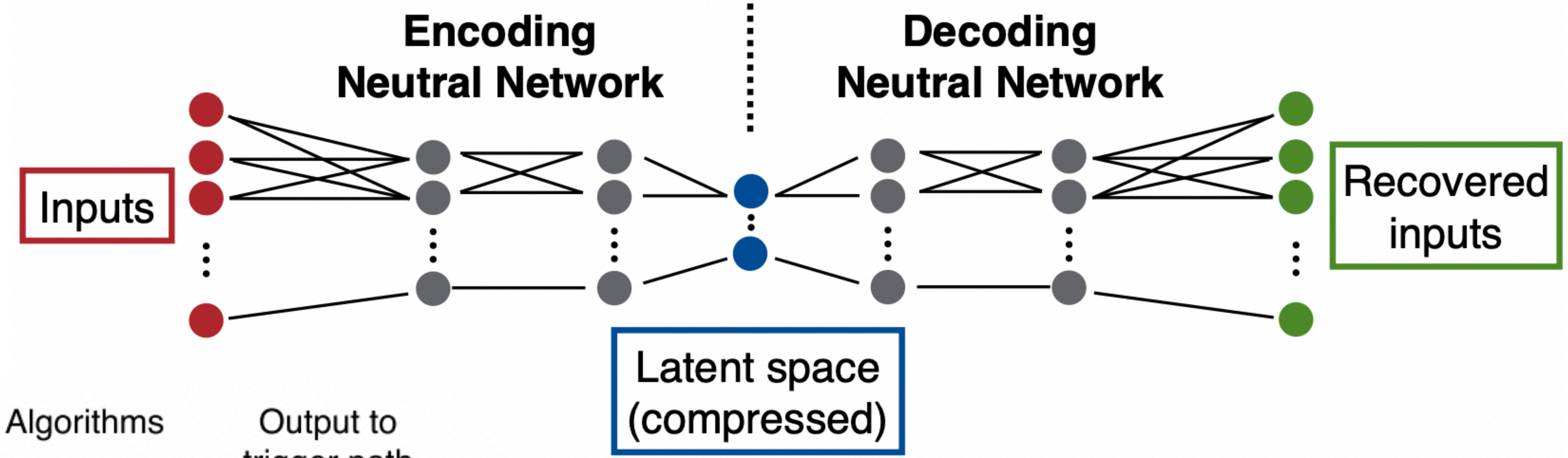


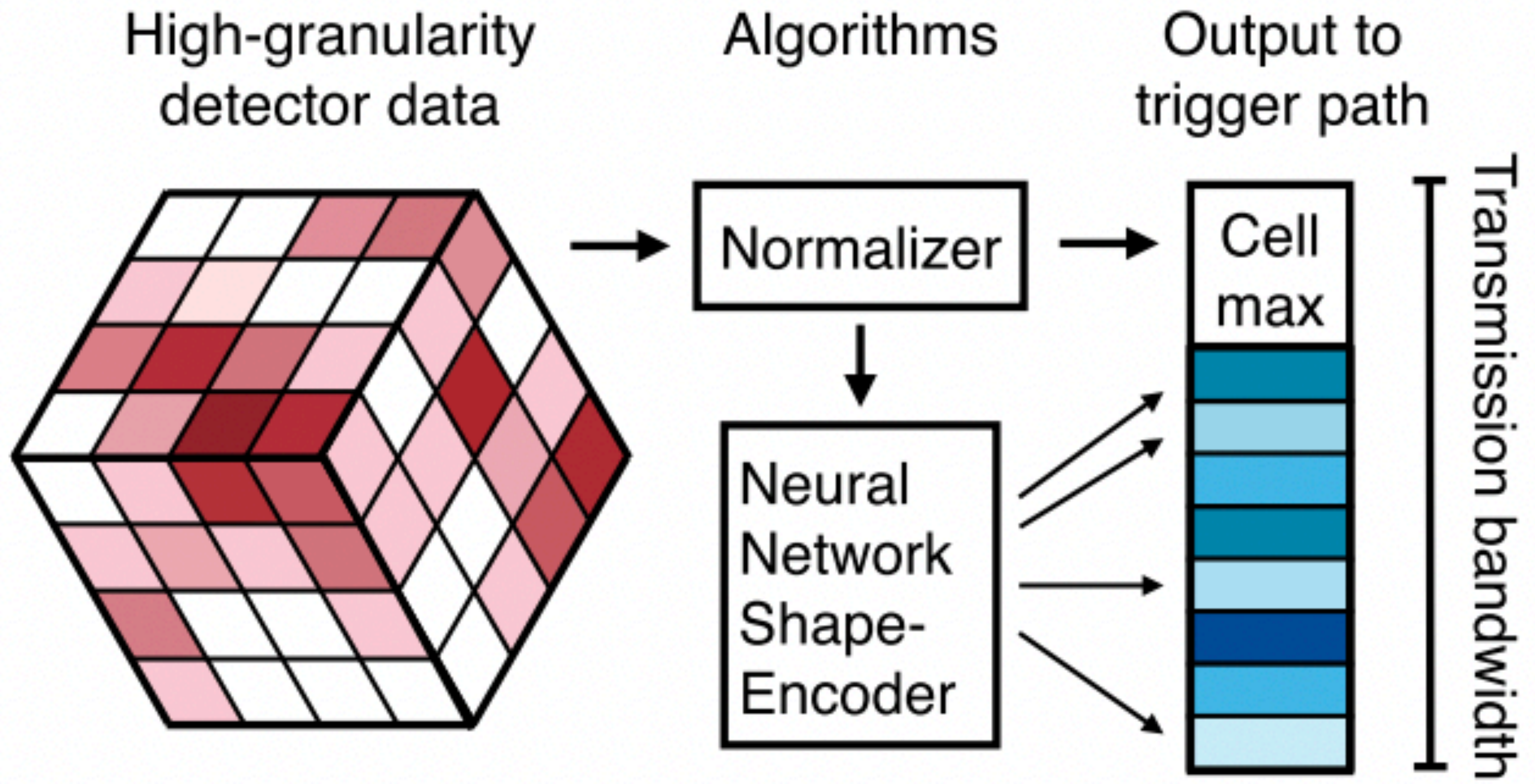


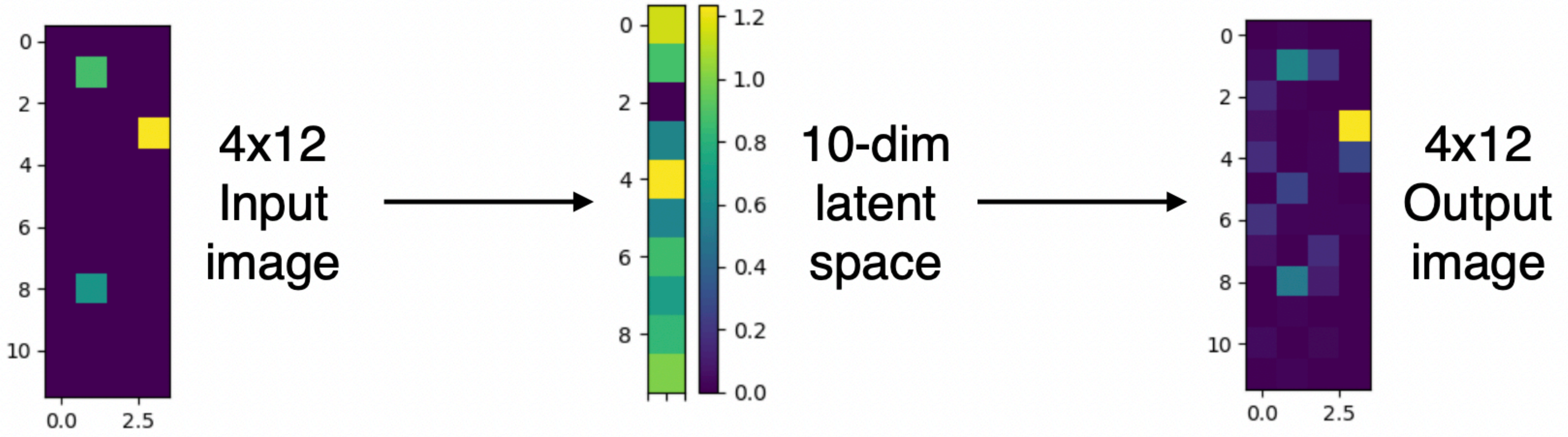
Applications: on-detector ML

- ECON-T ASIC for CMS High Granularity Calorimeter
    - Compress data to be sent to trigger FPGAs with an AutoEncoder in frontend, decode off detector
    - Includes “classical” algorithms (e.g. summing neighbouring cells) and an AutoEncoder

On detector → lpGBT → Off detector (trigger)



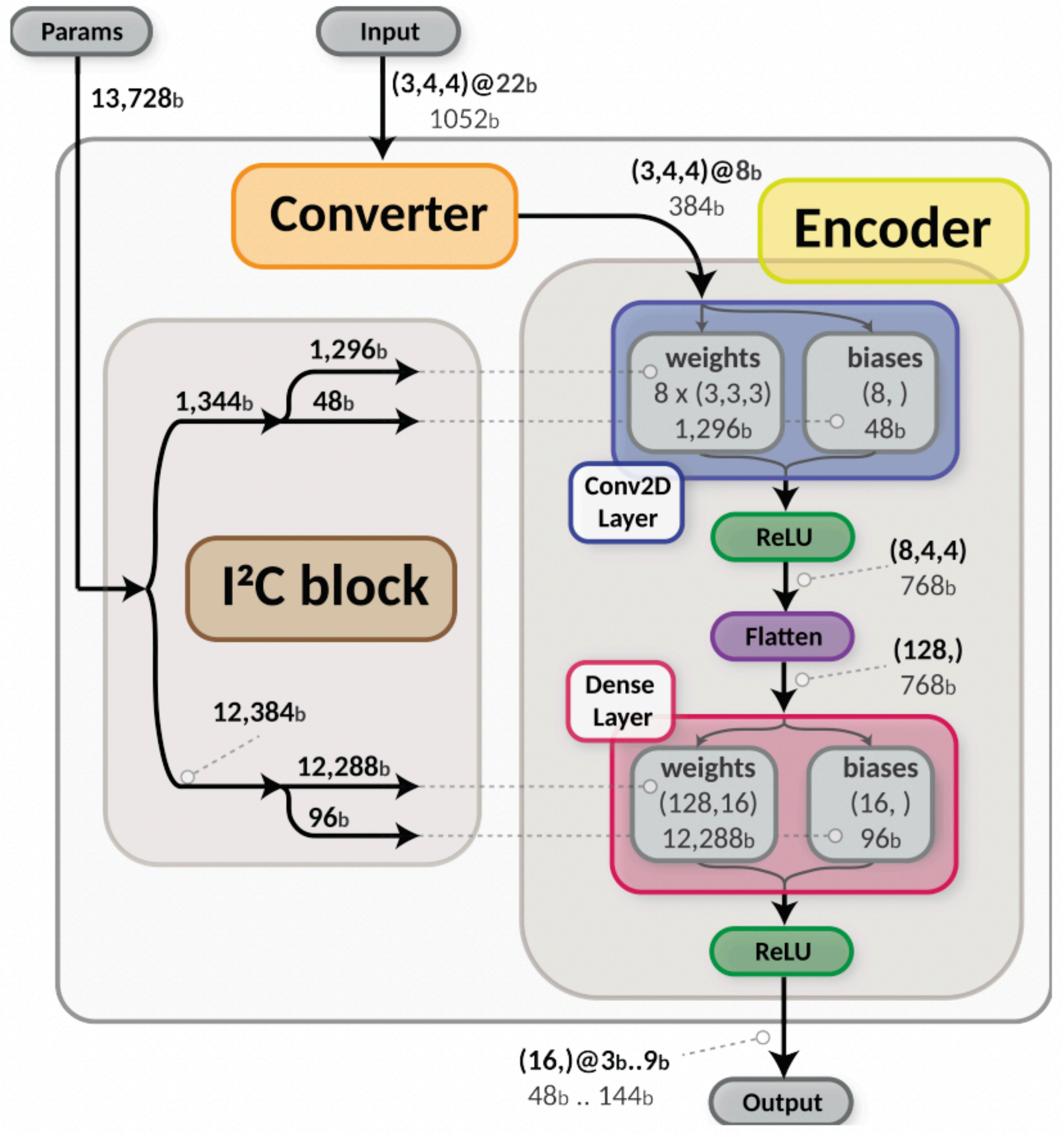


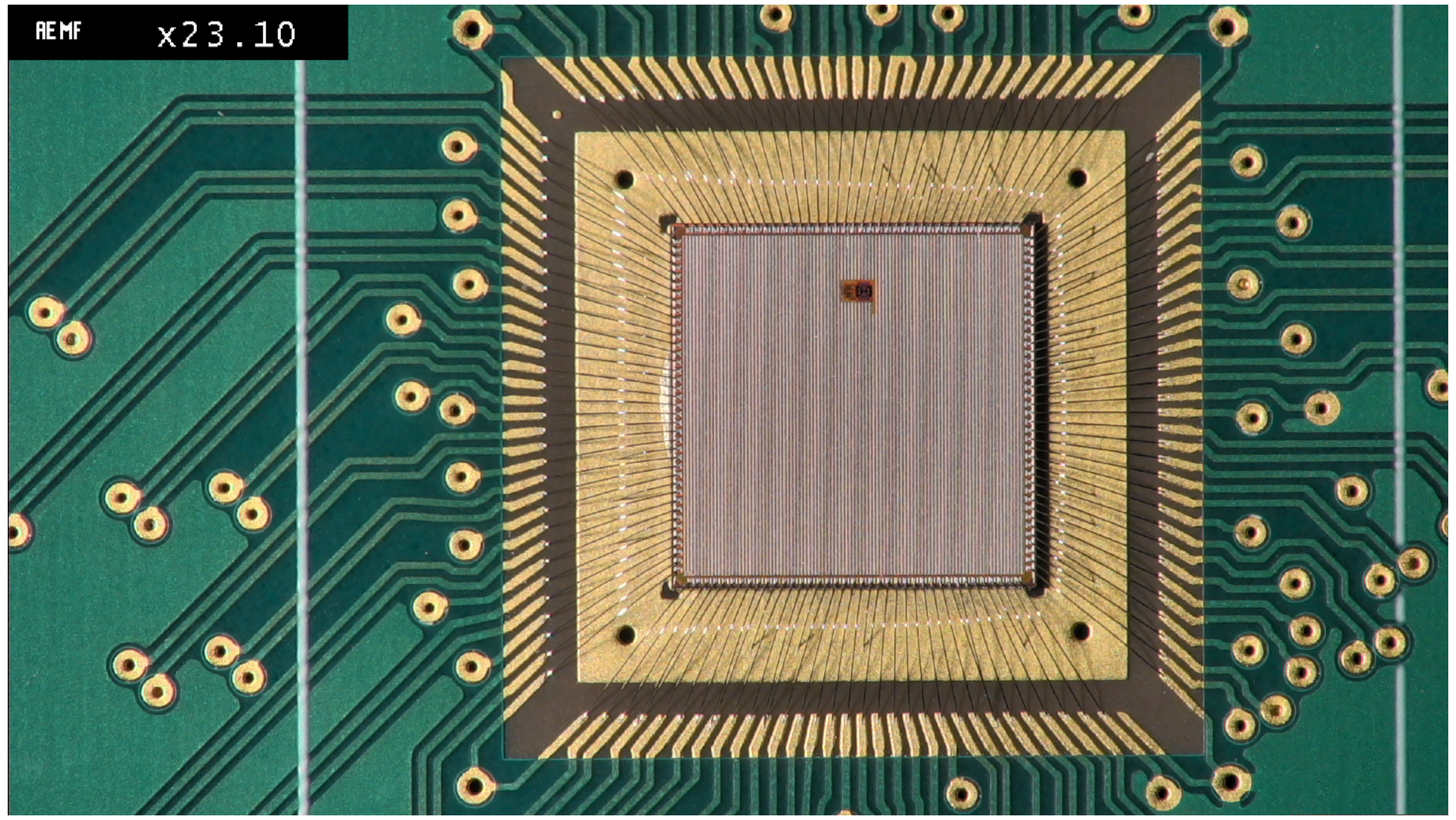


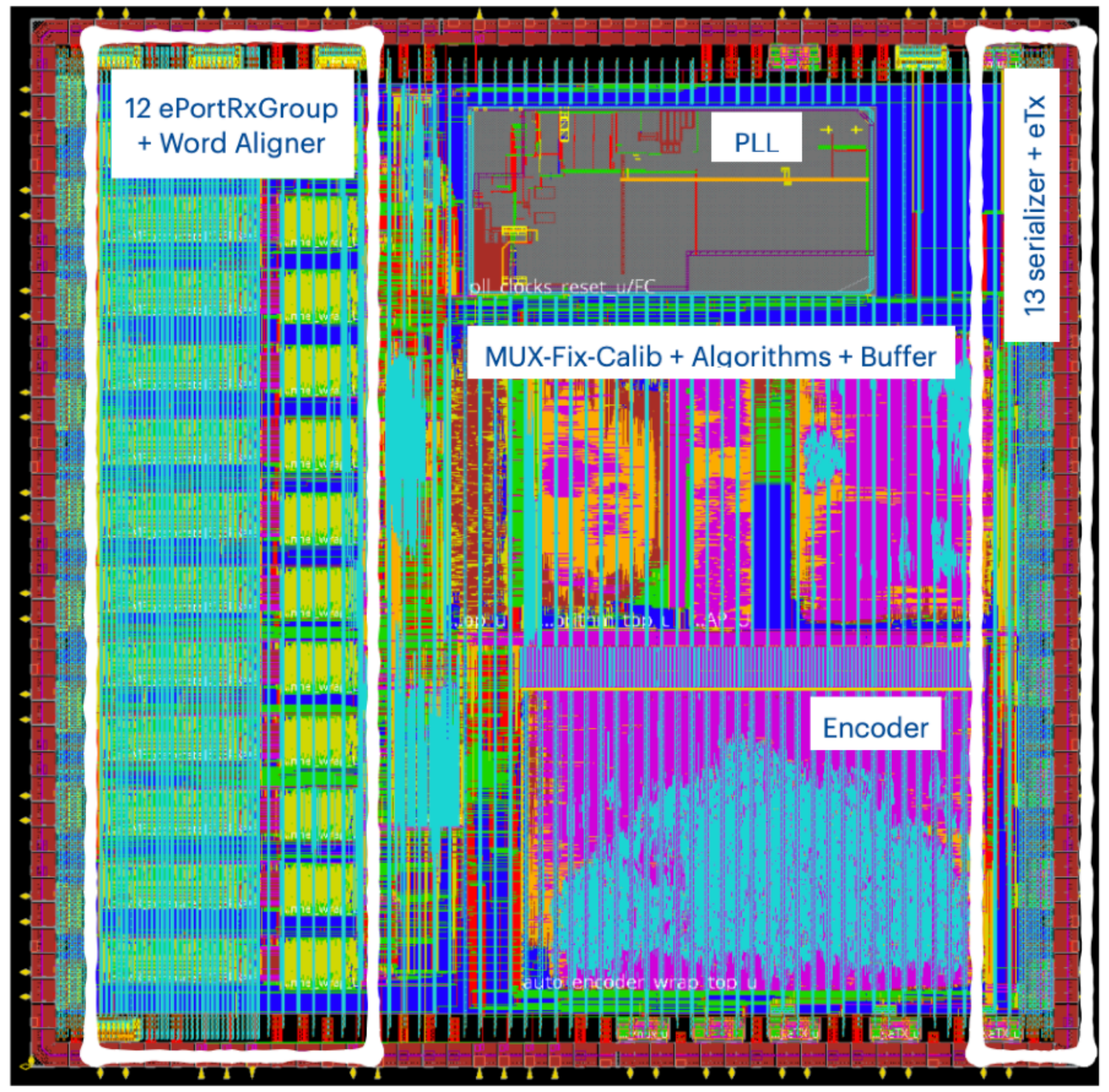
doi: 10.1109/TNS.2021.3087100

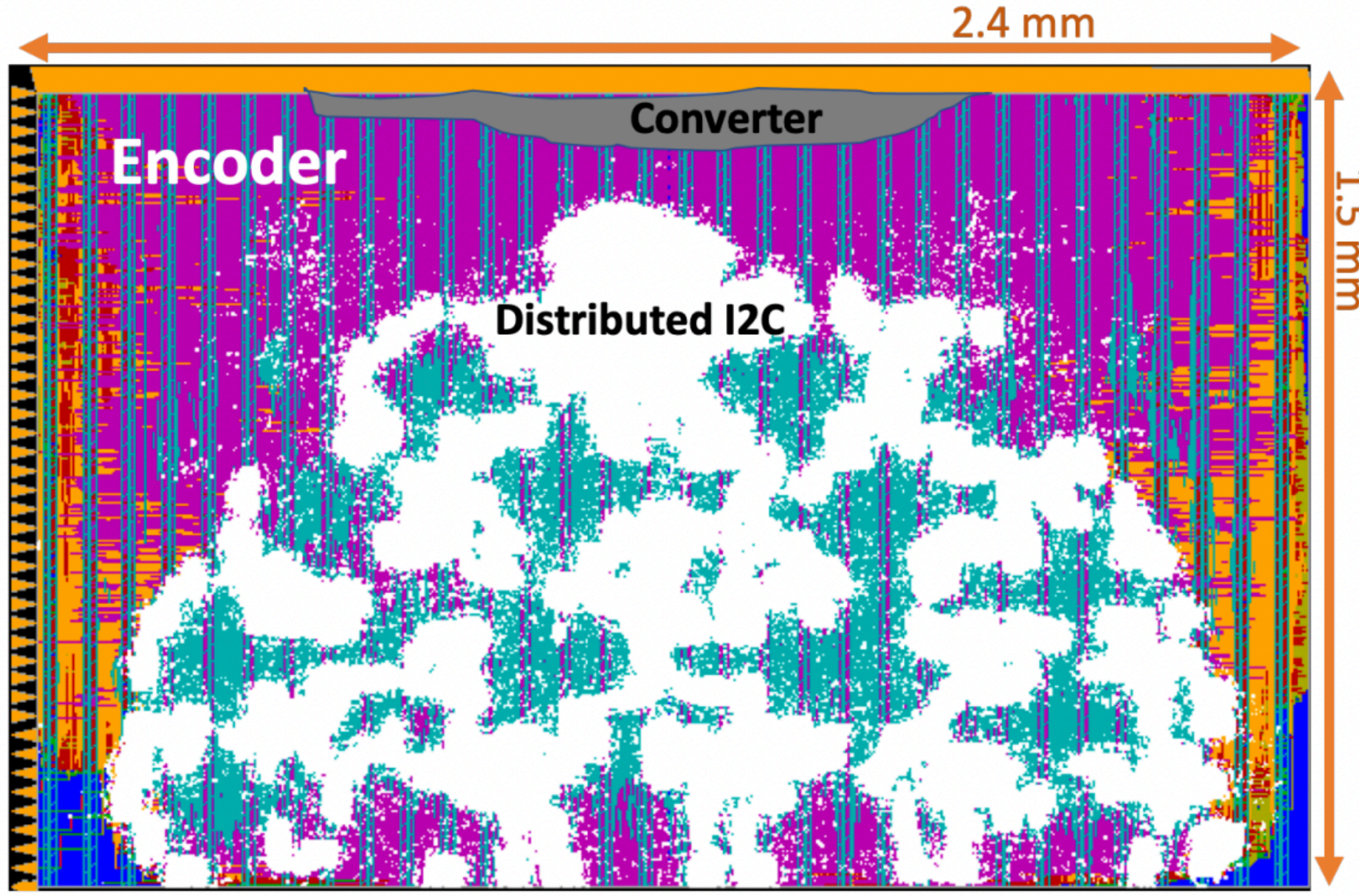
Applications: on-detector ML

- Neural Net encoder IP block created for ECON-T ASIC with Catapult HLS (Mentor/Siemens) and **hls4ml**
  - NN architecture is fixed, weights can be reprogrammed over I2C e.g. after NN retraining
  - NN parameters (weights and biases) triplicated for radiation tolerance
- Decoder block would run in trigger FPGAs
- Device manufactured and validated
- Can do Fault analysis at ML model level: FKeras (towards RadHard training)







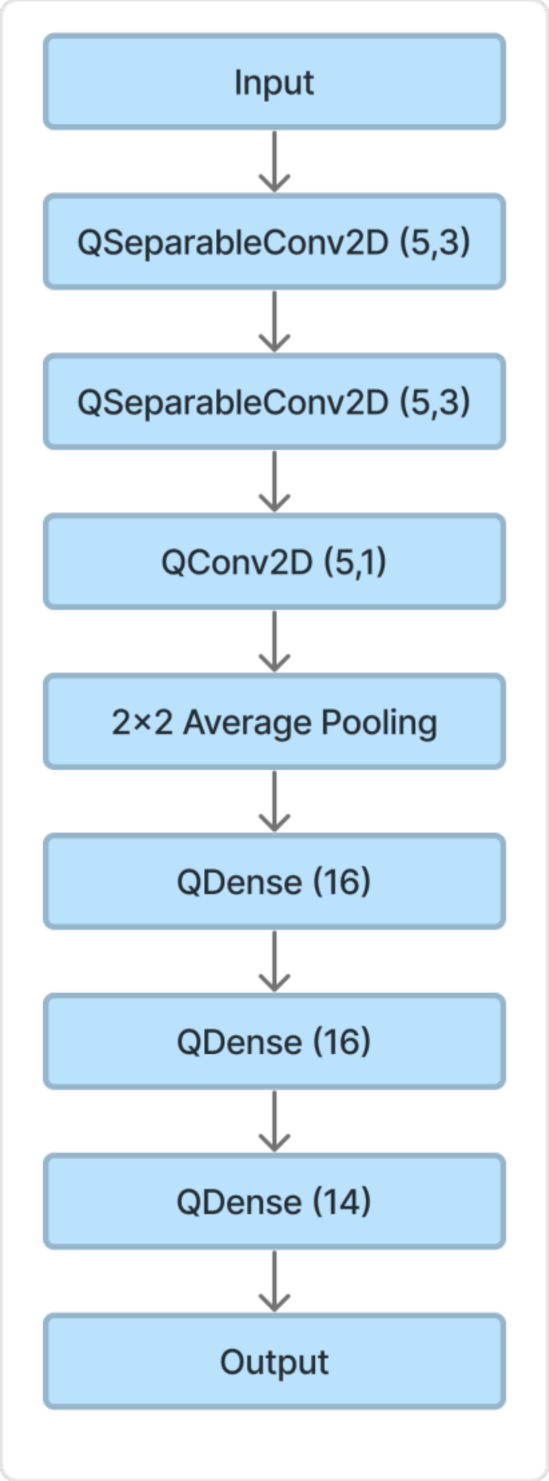


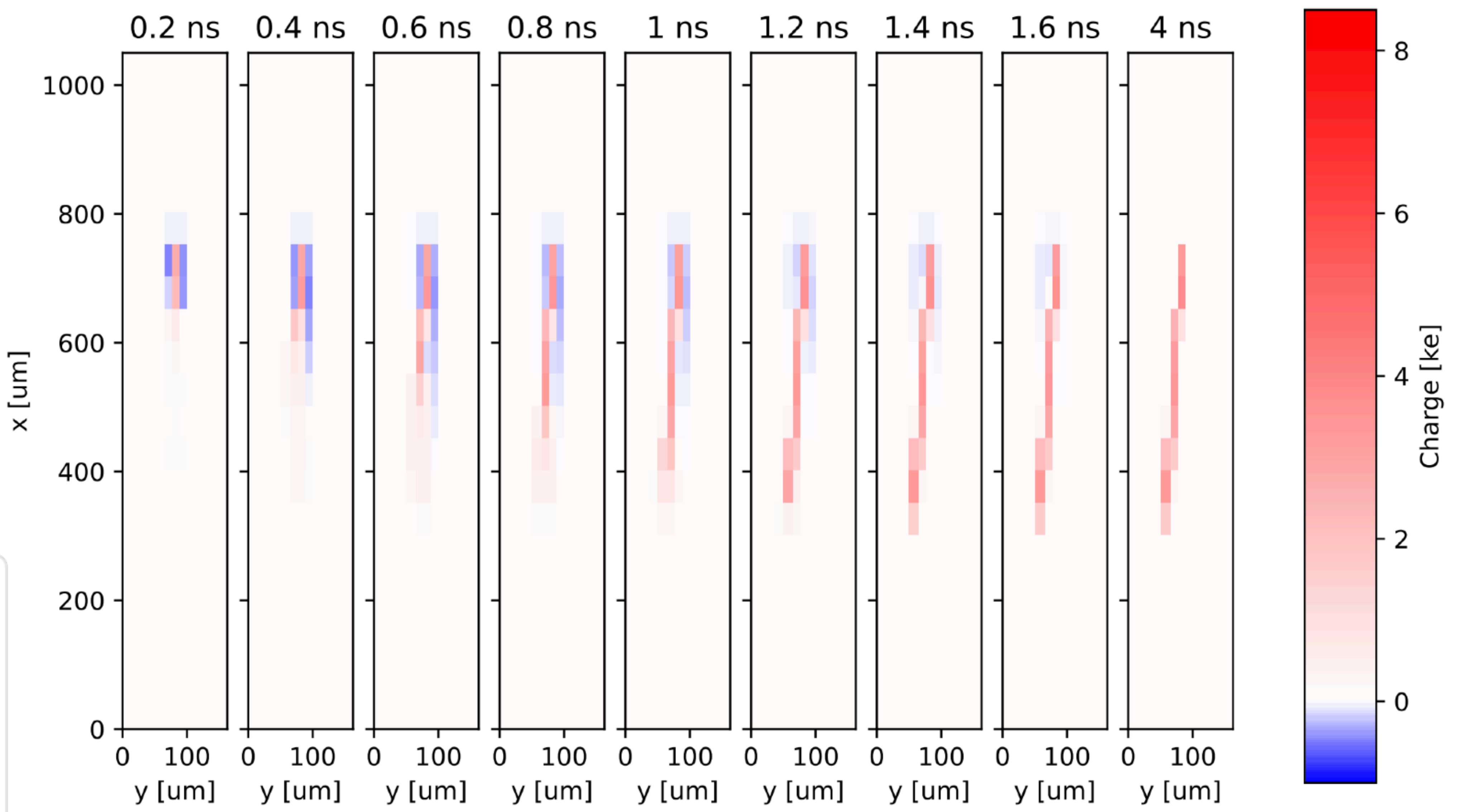
C. Suarez

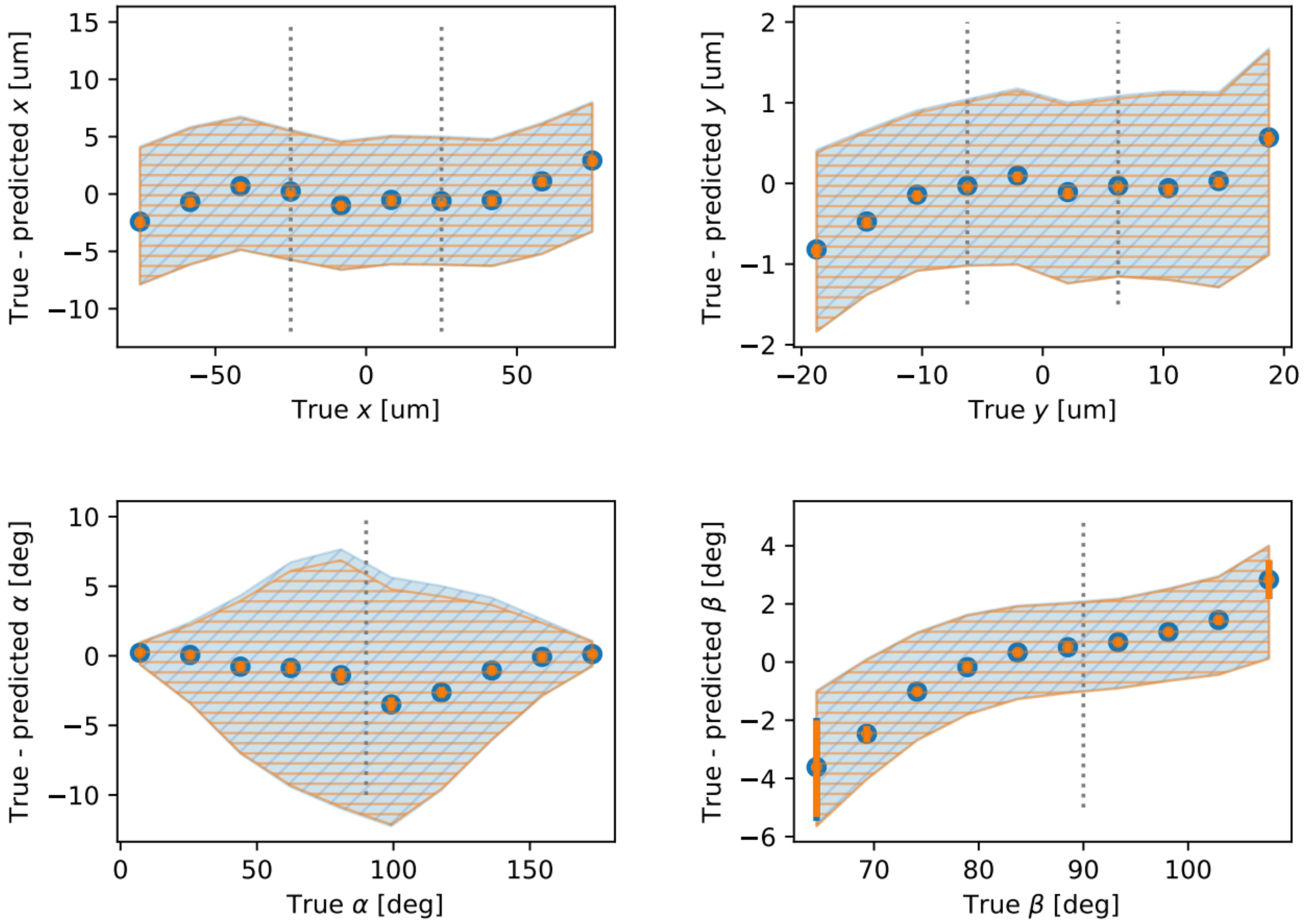
Smart Pixels

[arXiv:2312.11676](https://arxiv.org/abs/2312.11676)

- Data reduction and reconstruction on sensor for silicon pixel detectors
- Pixel detectors not read into hardware trigger systems
  - High data rate, expensive reconstruction
- Predict charged particle crossing position (x,y) and angles (α,β) from sensor charge measurements
  - And their covariance matrix
  - Could be for data reduction or early processing
- Tiny Convolutional Neural Network using **hls4ml** and Quantization Aware Training (QKeras)
- Opportunity to massively reduce search space for next hit along a track trajectory with on-sensor reco.
- Towards pixel reconstruction in trigger & simplified offline reconstruction
- Open dataset





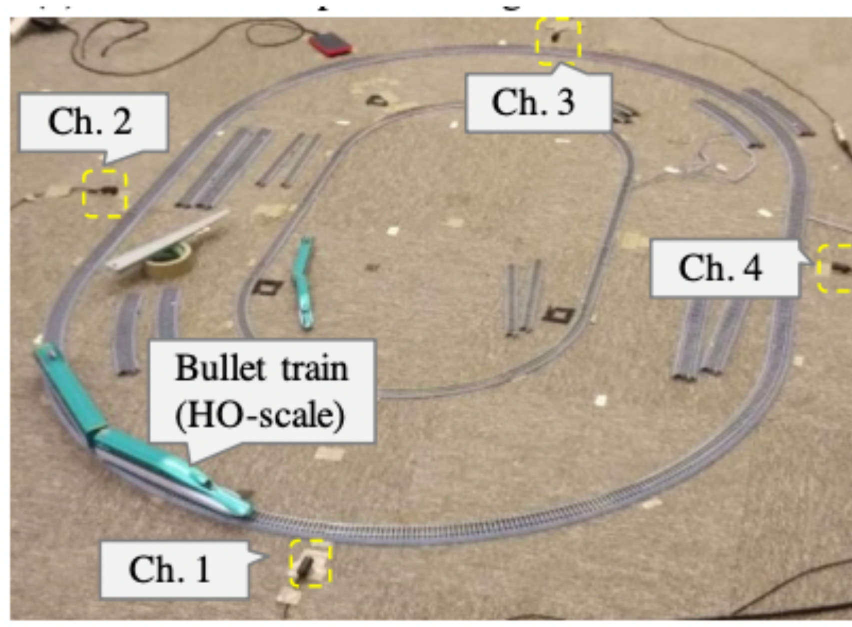


TinyML - MLPerf Tiny ™

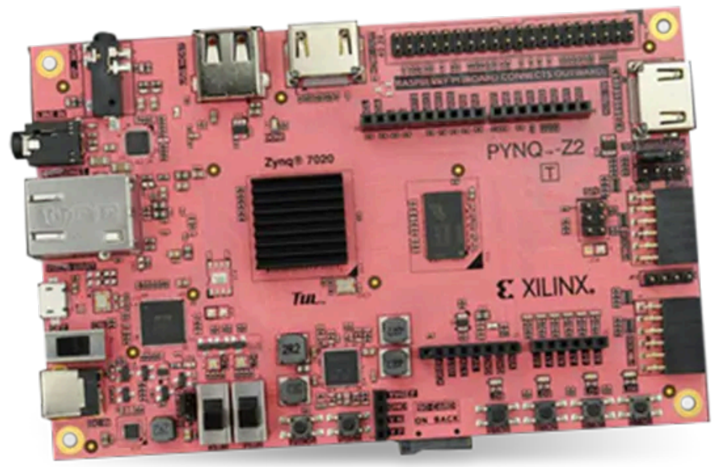
- MLCommons group organise benchmarks of Machine Learning (MLPerf) - now also for low power devices (MLPerf Tiny)
- 4 benchmark datasets, open/closed division allowing/disallowing model retraining
- **hls4ml** in open category (for Quantisation Aware Training) achieves competitive performance
- In collaboration with AMD / Xilinx Research Labs developers of FINN project

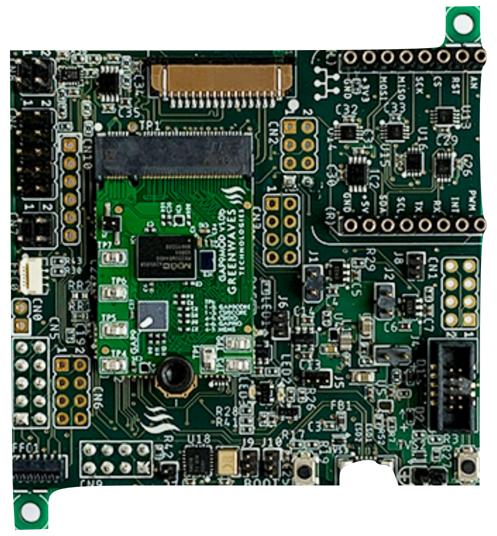
Benchmark		CIFAR-10			ToyADMOS		
Team	Device	Accuracy	Latency (ms)	Energy (uJ)	AUC	Latency (ms)	Energy (uJ)
hls4ml	Pynq-z2[1]	83.5%	7.64	12266	0.83	0.019	30.1
GreenWaves	GAP9 EVK[2]	85%	0.62	40.4	0.85	0.18	7.3
STMicro	Nucleo-L4R5ZI[3]	85%	54.3	8707	0.85	1.82	266.5
OctoML	Nucleo-L4R5ZI[3]	85%	389.2	21342	0.85	11.7	633.7

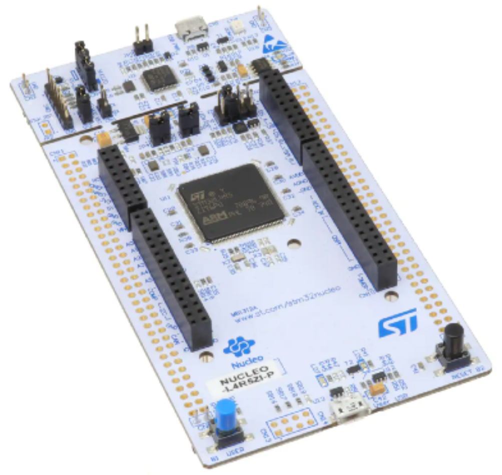
CIFAR-10



ToyADMOS

1


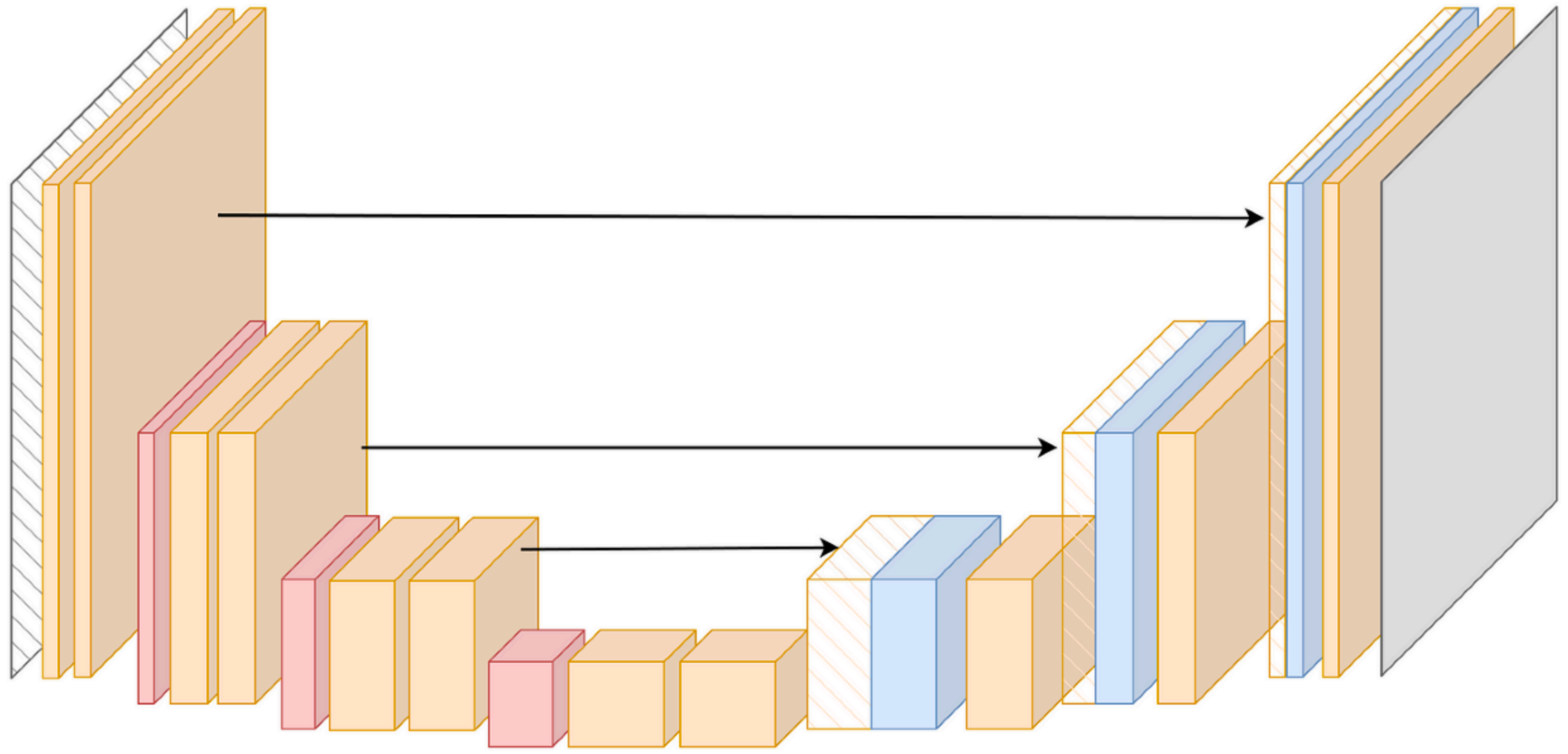
2


3


Autonomous Vehicles

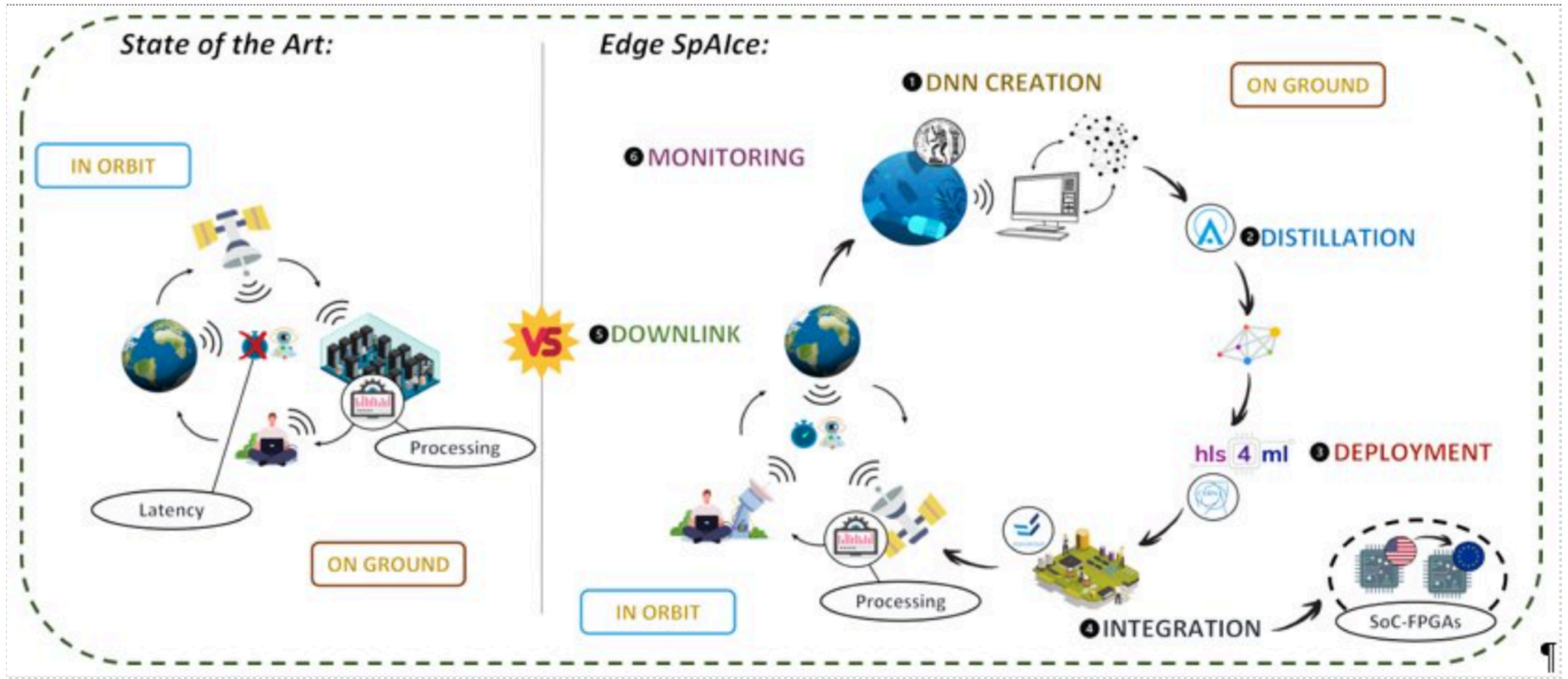
- Image segmentation is labelling the class of each pixel of an image
  - e.g. road, vehicle, pedestrian, other for a self driving car
- Project undertaken in partnership with Zenseact - Swedish autonomous vehicle ML solutions company - and CERN Knowledge Transfer (Paper, web)
- Developed new image-streaming CNN implementations for **hls4ml**
- Trained Quantized NNs with AutoQKeras on Cityscapes dataset
  - Label pixels as road/pedestrian/car/background
  - Lowest latency model has around 10k parameters, 8 bit quantisation
- Deployed on ZCU102 Zynq SoC kit with **hls4ml**
  - 5 ms latency with batch size = 1, 30 ms with batch size = 10

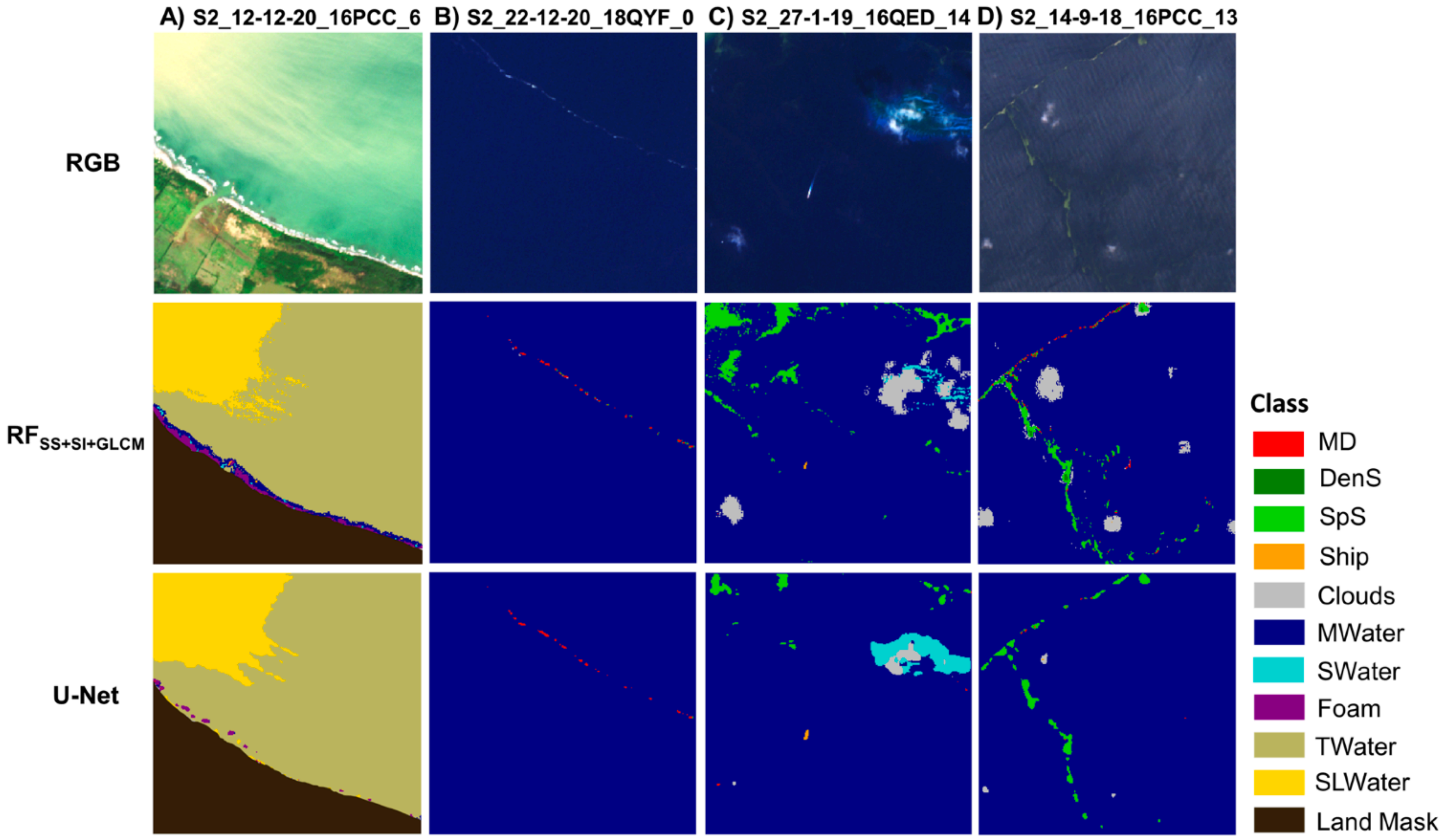
Model	Acc.	mIoU	Latency [ms]		BRAM	LUT	FF	DSP
			b=1	b=10				
EnetHQ	81.1%	36.8 %	4.9	30.6	224.5 (25%)	76,718 (30%)	87,059 (16%)	450 (18%)
Enet8Q4	77.6%	33.9 %	4.8	30.2	342.0 (37%)	166,741 (61%)	90,536 (16%)	0
Enet8Q8	77.1%	33.4 %	4.8	30.0	508.5 (56%)	126,458 (46%)	134,385 (25%)	1,502 (60%)
ENet [23]	-	63.1%	30.38 (720)[a]	-	257	62,599	192,212	689



hls4ml - Earth Observation

- Using **hls4ml** to monitor plastics pollution in the ocean onboard Earth Observation satellites
- Satellites use space-grade FPGAs that are resistant to errors caused by bit-flips from radiation
- Downlink bandwidth is limited, and missions may only look for certain objects (plastics pollution, land use)
- Hyper-spectral imaging and Deep Neural Networks effective at detecting surface objects from satellites
  - Image segmentation - label each pixel
- Deploy DNN onboard satellite to identify debris, avoid downlink of useless data, decreasing mission cost and notification time
- Using **hls4ml** to reach sweet spot of performance, area, latency/throughput and power usage
- New project since 2 months, first results soon!





MARIDA

NextGen Triggers

- If this sounds exciting to you, there are opportunities to get involved!
- New 5 year project at CERN to advance use of Artificial Intelligence for LHC experiment's trigger selections
- Opportunities for students (bachelors, masters, doctoral) and postdocs opening ~now and throughout the next 5 years
- For CMS L1T projects contact me and Cristina Botta
- For ATLAS projects contact Markus Elsing, Stefano Veneziano
- For **hls4ml** and **conifer** development projects contact Maurizio Pierini
- **hls4ml** and **conifer** are also open source software, and contributions can come from anywhere

Summary

- Fast Machine Learning is changing the way we process detector data and make trigger decisions
- Embedding ML closer to the detector requires sophisticated techniques
    - Strict constraints (low latency, high throughput, low area, low power) and often highly custom compute platforms
    - Projects like hls4ml and conifer aim to lower the barrier to entry for deployment of ML models optimised for the needs of HEP
- I presented the tools, and techniques like Quantization Aware Training, pruning, and hardware aware training
- We reviewed applications, from final trigger decision selections to frontend data processing