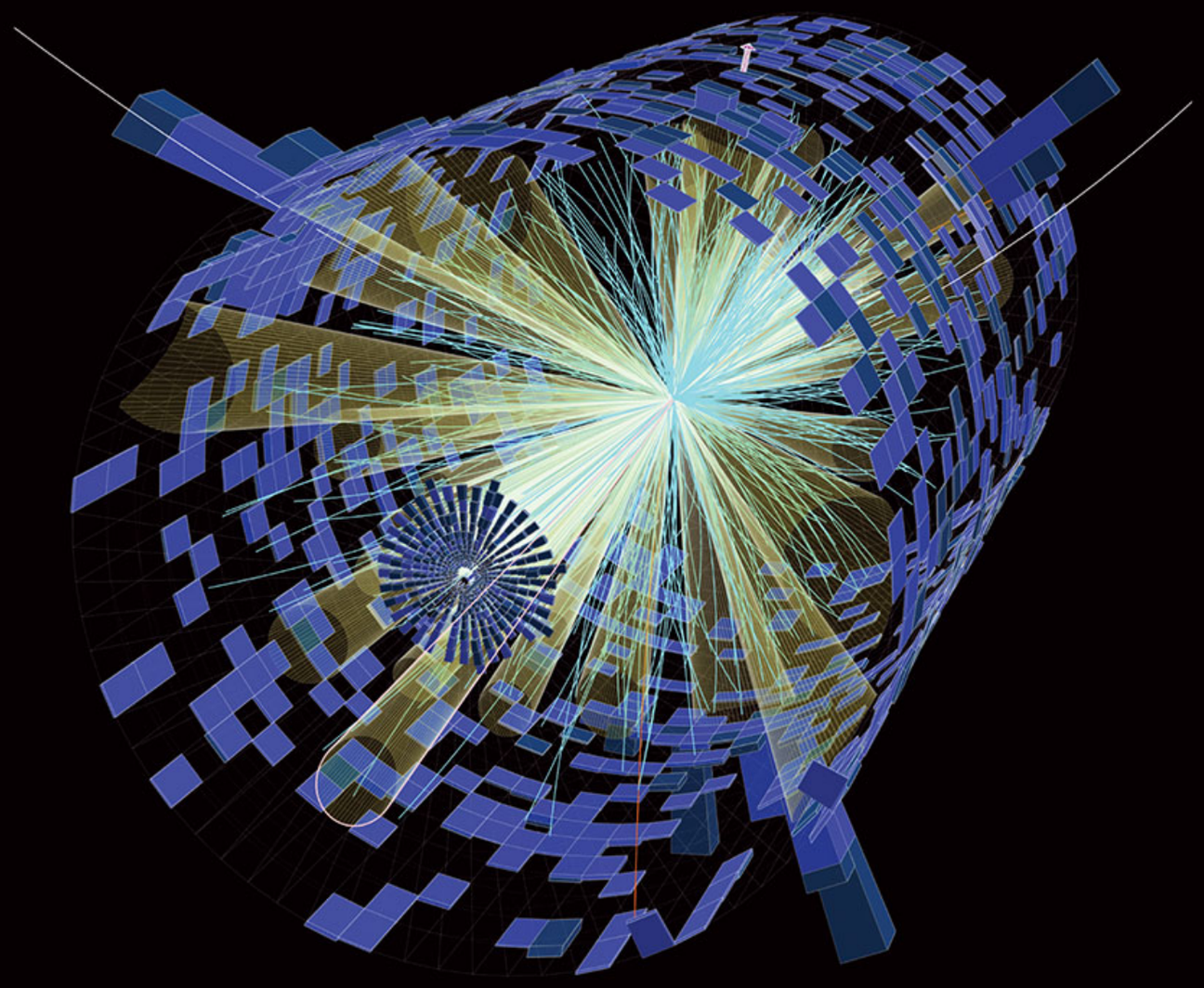
Real-time AI for LHC triggers

Jennifer Ngadiuba (Fermilab)

TREASURE workshop
BNL
April 27—29, 2026



Fermilab

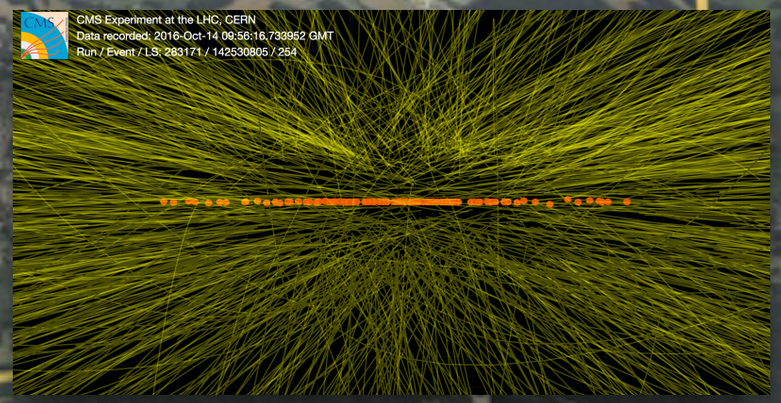
CERN

FastML Lab

Big Data @ the Energy Frontier

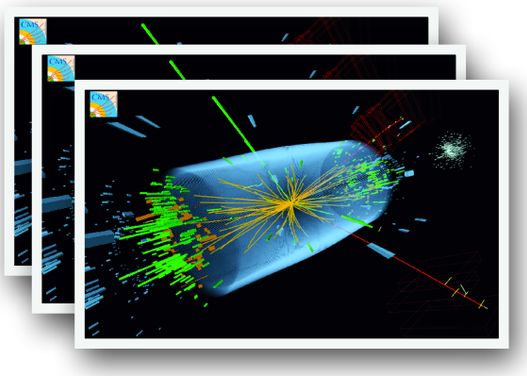
The Large Hadron Collider (LHC)

Collision frequency: 40 MHz
Particles per collision: O(10^3)
Detector resolution: ~ 1B channels



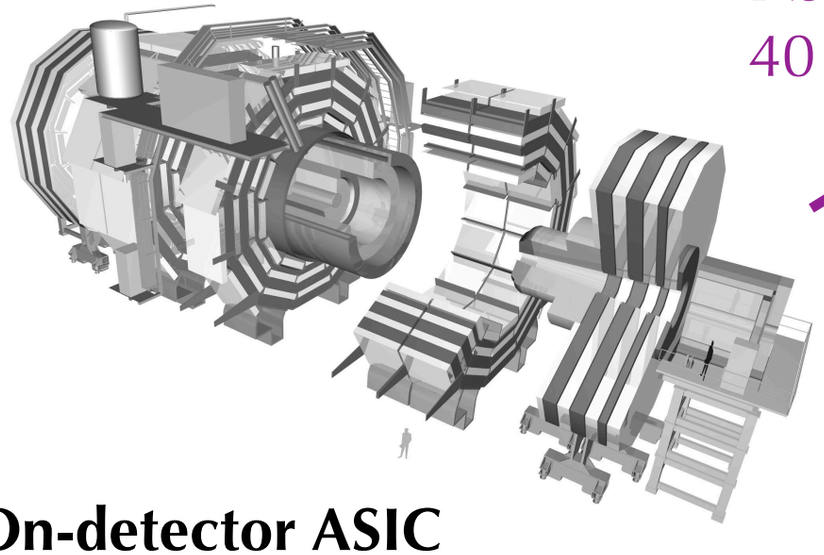
Online selection (trigger) necessary to reduce data rates & volume:
1-2 MB/event @ 40 MHz → 500 Tbps , 300 EB/year

Data reduction workflow @ LHC



CMS Experiment
40 MHz collision rate
~1B detector channels

Pb/s
40 MHz



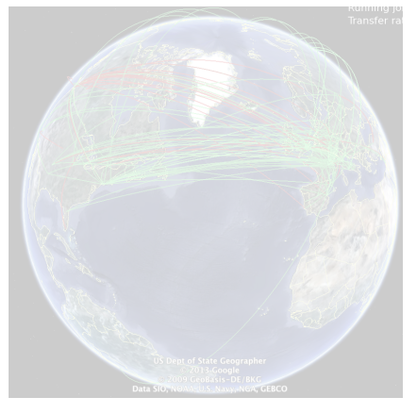
On-detector ASIC compression
~100 ns latency

FPGA filter stack
~µs latency

Level-1 Trigger

10s Tb/s
100s kHz

10s Gb/s
~5 kHz



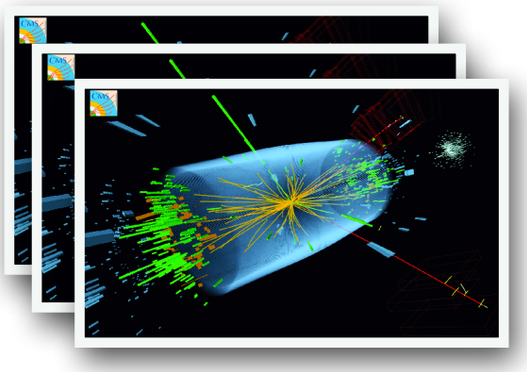
Worldwide computing grid
Exabyte-scale datasets

Offline analysis

High-Level Trigger

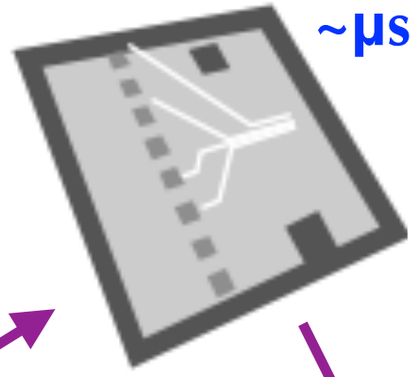
On-prem CPU/GPU filter farm
~100 ms latency

Data reduction workflow @ LHC



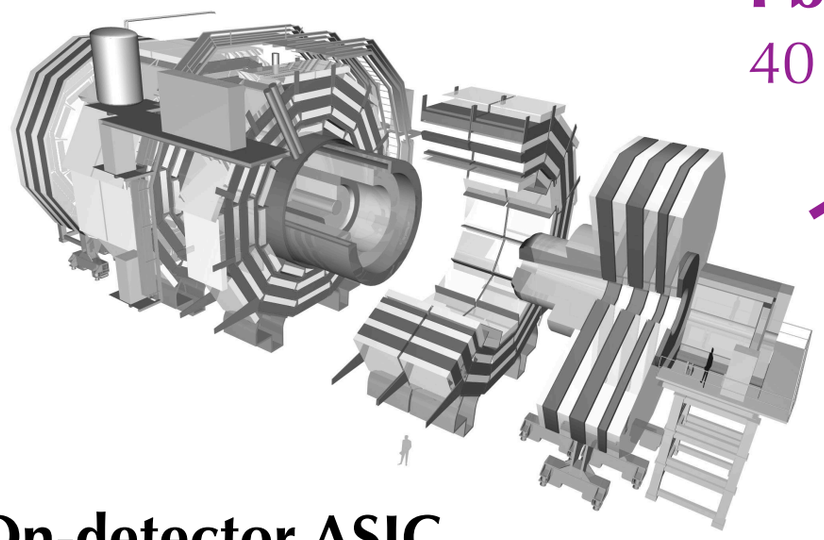
CMS Experiment
40 MHz collision rate
~1B detector channels

FPGA filter stack
~µs latency



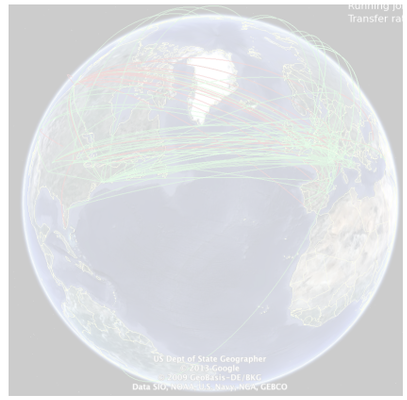
Level-1 Trigger

Pb/s
40 MHz



On-detector ASIC compression
~100 ns latency

10s Tb/s
100s kHz



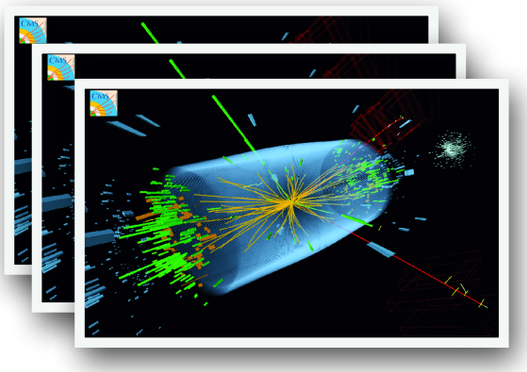
10s Gb/s
~5 kHz

Offline analysis

High-Level Trigger

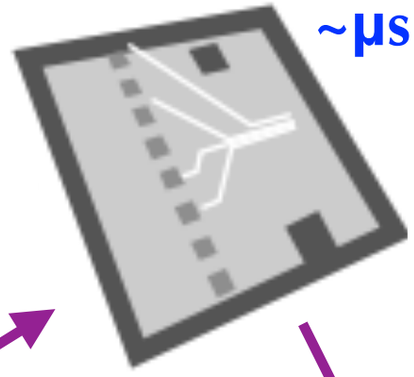
On-prem CPU/GPU filter farm
~100 ms latency

Data reduction workflow @ LHC



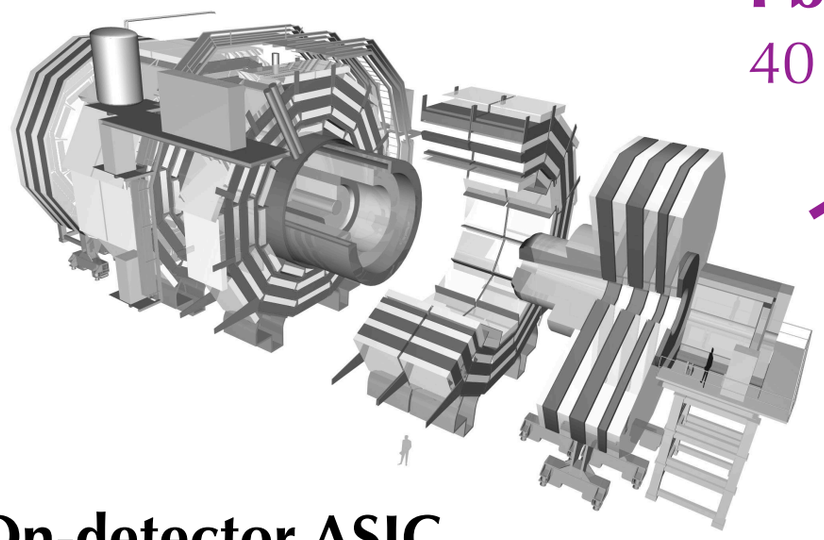
CMS Experiment
40 MHz collision rate
~1B detector channels

FPGA filter stack
~µs latency



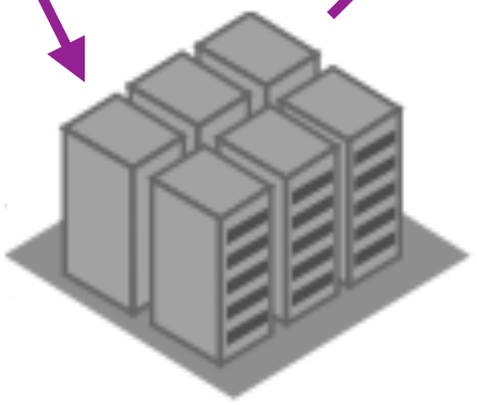
Level-1 Trigger

Pb/s
40 MHz



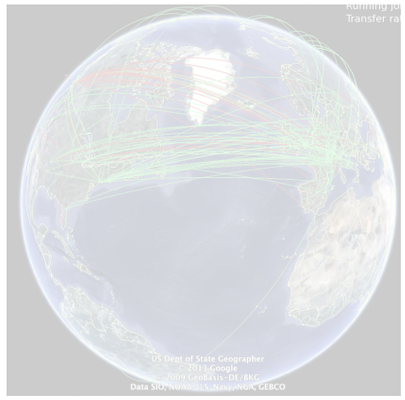
On-detector ASIC compression
~100 ns latency

10s Tb/s
100s kHz



High-Level Trigger

10s Gb/s
~5 kHz

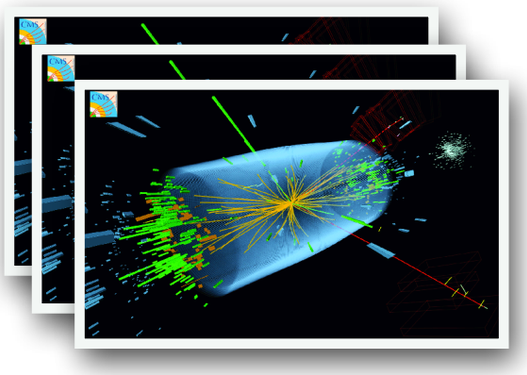


Worldwide computing grid
Exabyte-scale datasets

Offline analysis

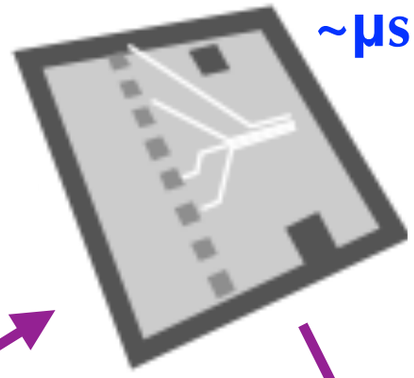
On-prem CPU/GPU filter farm
~100 ms latency

Data reduction workflow @ LHC



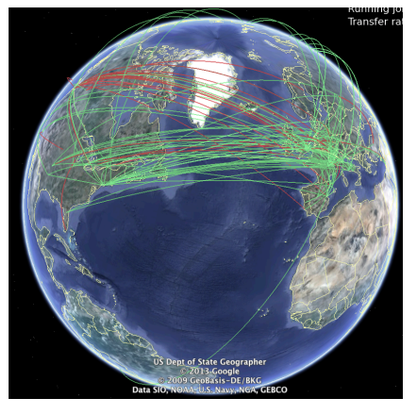
CMS Experiment
40 MHz collision rate
~1B detector channels

FPGA filter stack
~µs latency

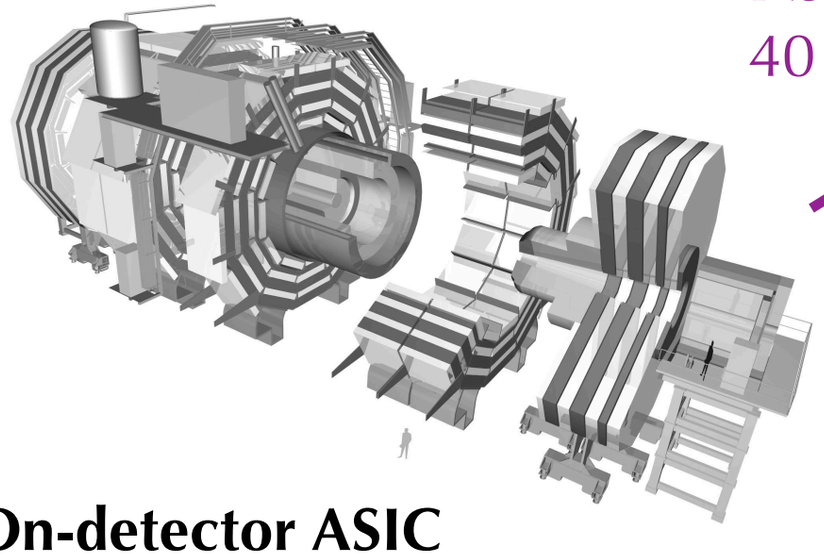


Level-1 Trigger

Worldwide computing grid
Exabyte-scale datasets



Pb/s
40 MHz

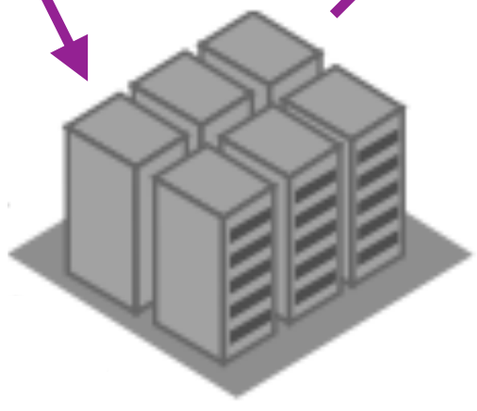


On-detector ASIC compression
~100 ns latency

10s Tb/s
100s kHz

10s Gb/s
~5 kHz

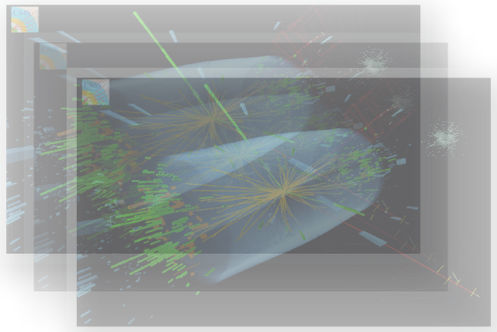
Offline analysis



High-Level Trigger

On-prem CPU/GPU filter farm
~100 ms latency

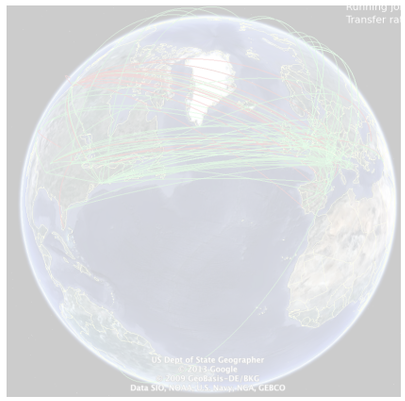
Data reduction workflow @ LHC



CMS Experiment
40 MHz collision rate
~1B detector channels

Worldwide computing grid
Exabyte-scale datasets

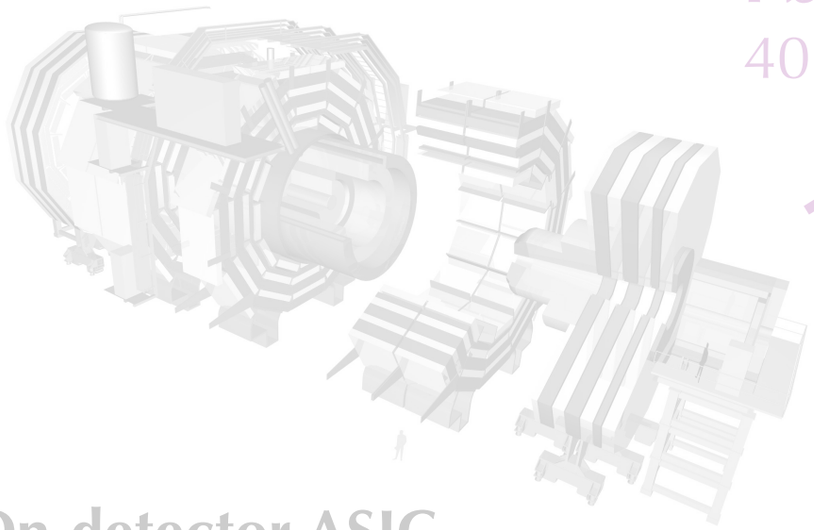
FPGA filter stack
~µs latency



99.75% events rejected!

Level-1 Trigger

Pb/s
40 MHz



10s Gb/s
~5 kHz

10s Tb/s
100s kHz

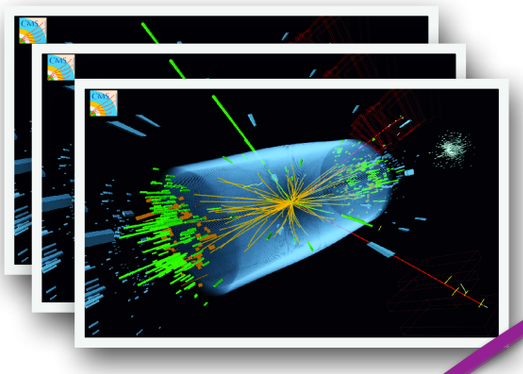
On-detector ASIC compression
~100 ns latency

99% events rejected!

High-Level Trigger

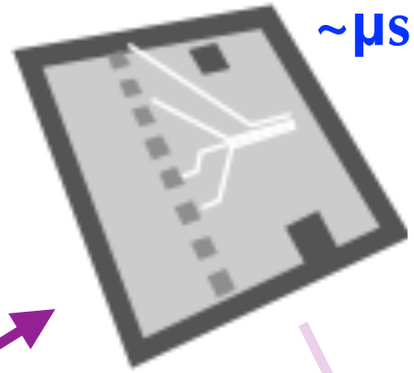
On-prem CPU/GPU filter farm
~100 ms latency

Data reduction workflow @ LHC



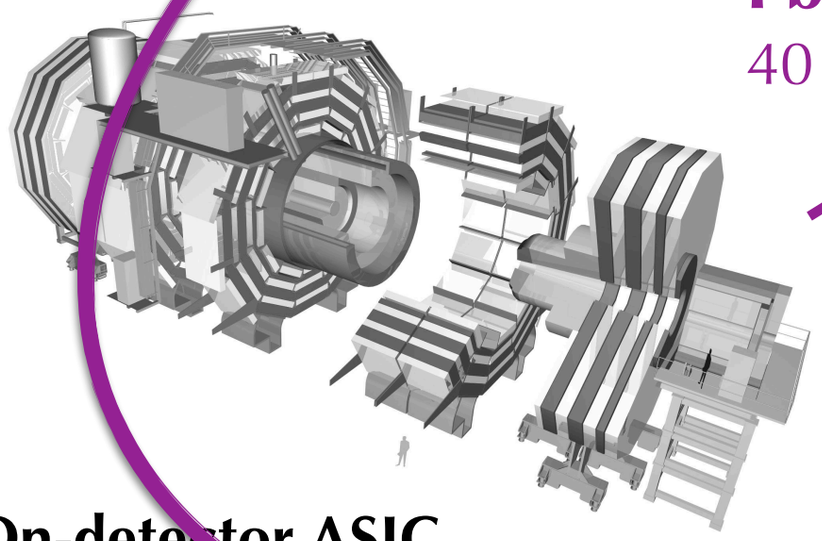
CMS Experiment
40 MHz collision rate
~1B detector channels

FPGA filter stack
~µs latency



Level-1 Trigger

Pb/s
40 MHz

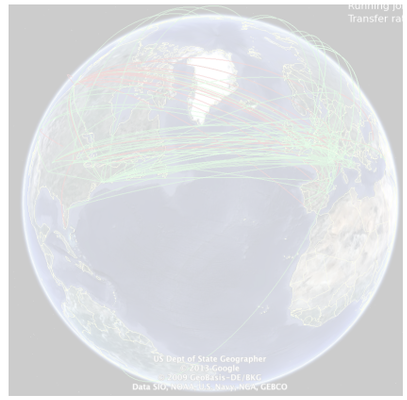


On-detector ASIC compression
~100 ns latency

10s TB/s
100s kHz

One of the challenges today is to bring complex AI models to the extreme environments closer to the data source!

Worldwide computing grid
Exabyte-scale datasets



High-Level Trigger

On-prem CPU/GPU filter farm
~100 ms latency

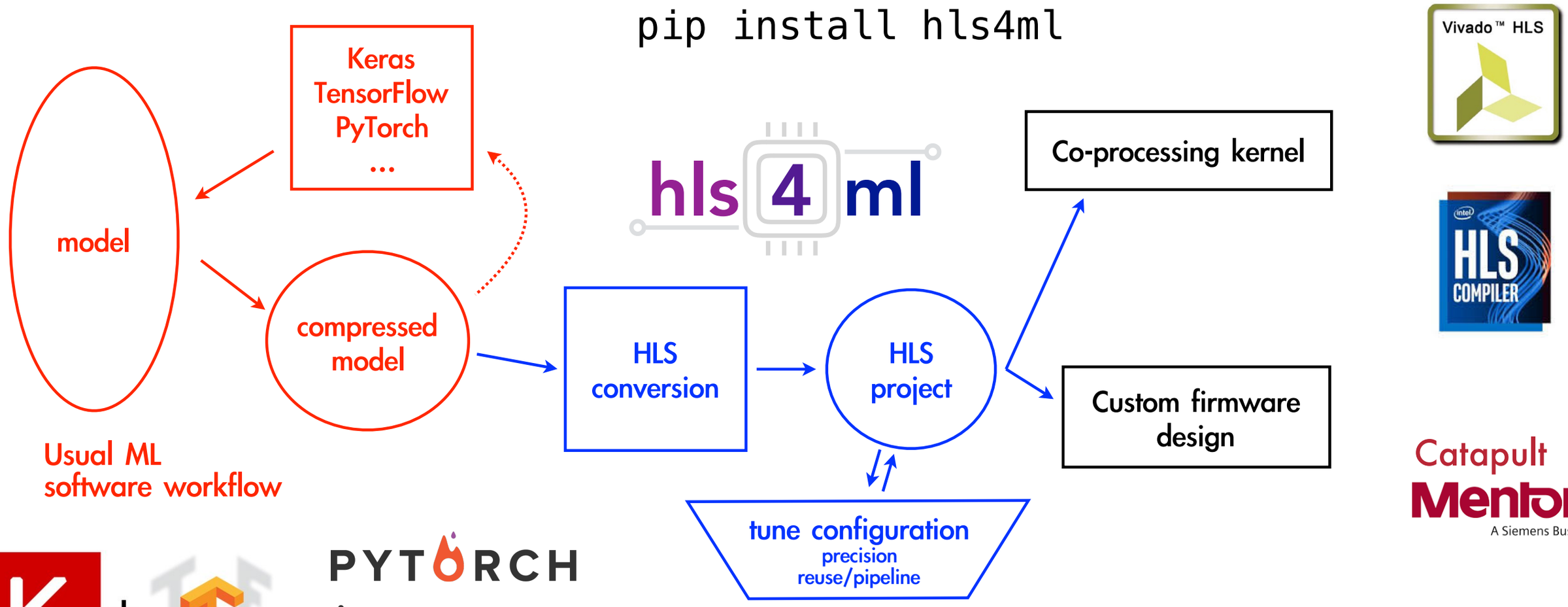
Bring ML models to hardware for real-time AI

high level synthesis for machine learning

A tool to efficiently program the FPGA hardware for Neural Networks with experimental constraints in mind!

Many use cases in HEP and beyond… and still growing!
(see Fast Machine Learning For Science Workshop Sept '25)

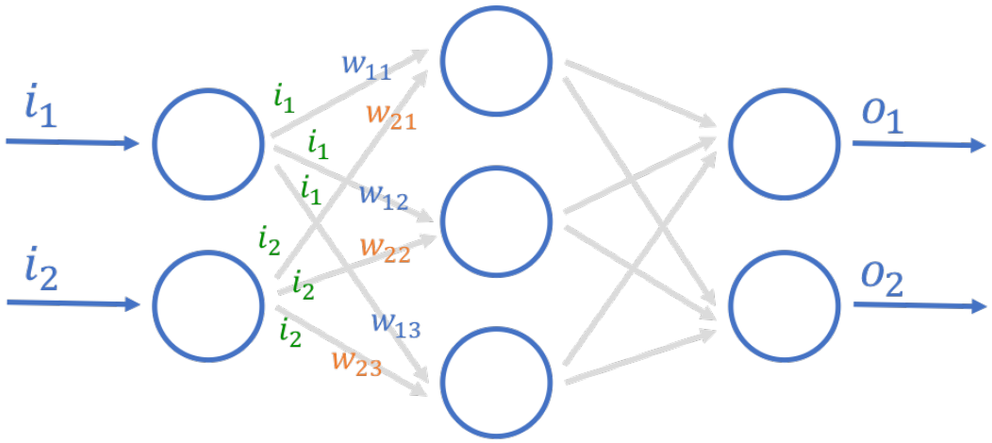
```
pip install hls4ml
```



https://fastmachinelearning.org/hls4ml/

Neural Network inference on FPGA

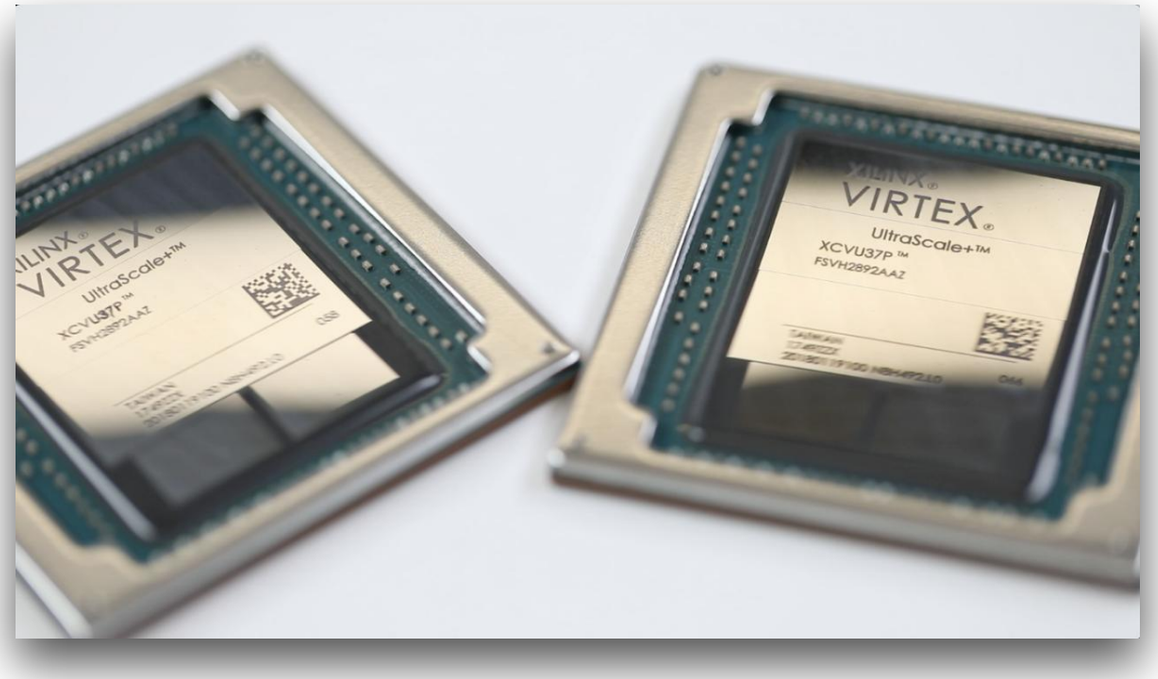
Neural network inference
=
matrix multiplication



$$\begin{bmatrix} w_{11} & w_{21} \\ w_{12} & w_{22} \\ w_{13} & w_{23} \end{bmatrix} \cdot \begin{bmatrix} i_1 \\ i_2 \end{bmatrix} = \begin{bmatrix} (w_{11} \times i_1) + (w_{21} \times i_2) \\ (w_{12} \times i_1) + (w_{22} \times i_2) \\ (w_{13} \times i_1) + (w_{23} \times i_2) \end{bmatrix}$$

Efficient implementation on FPGA uses
DIGITAL SIGNAL PROCESSORS

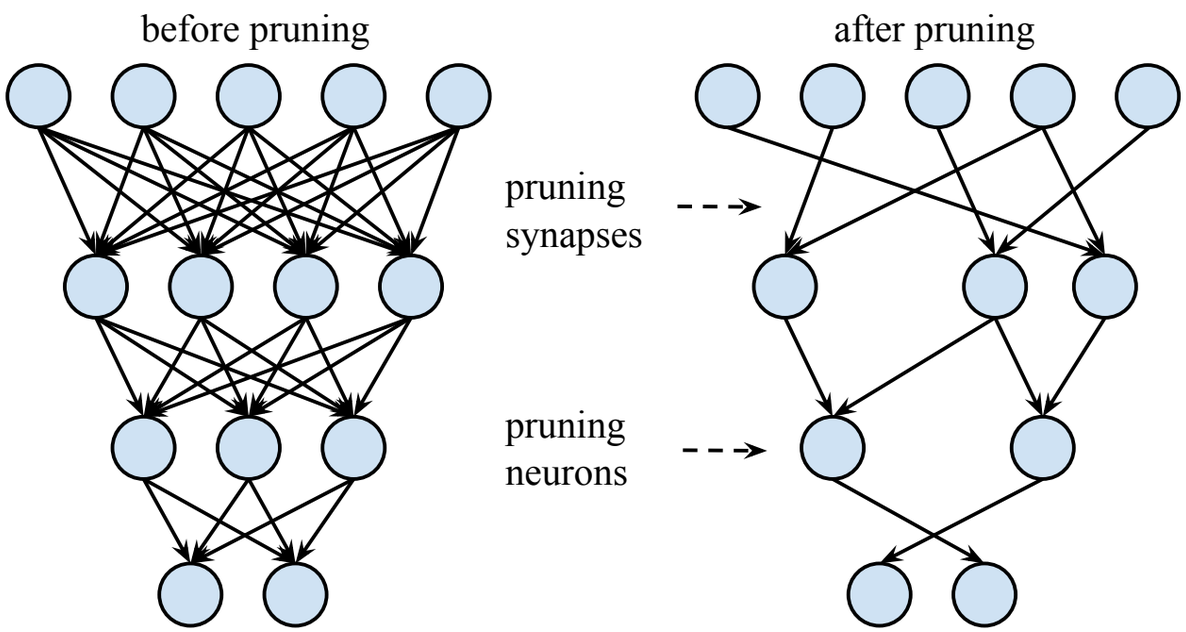
There are about 5–10k DSPs in modern FPGAs!

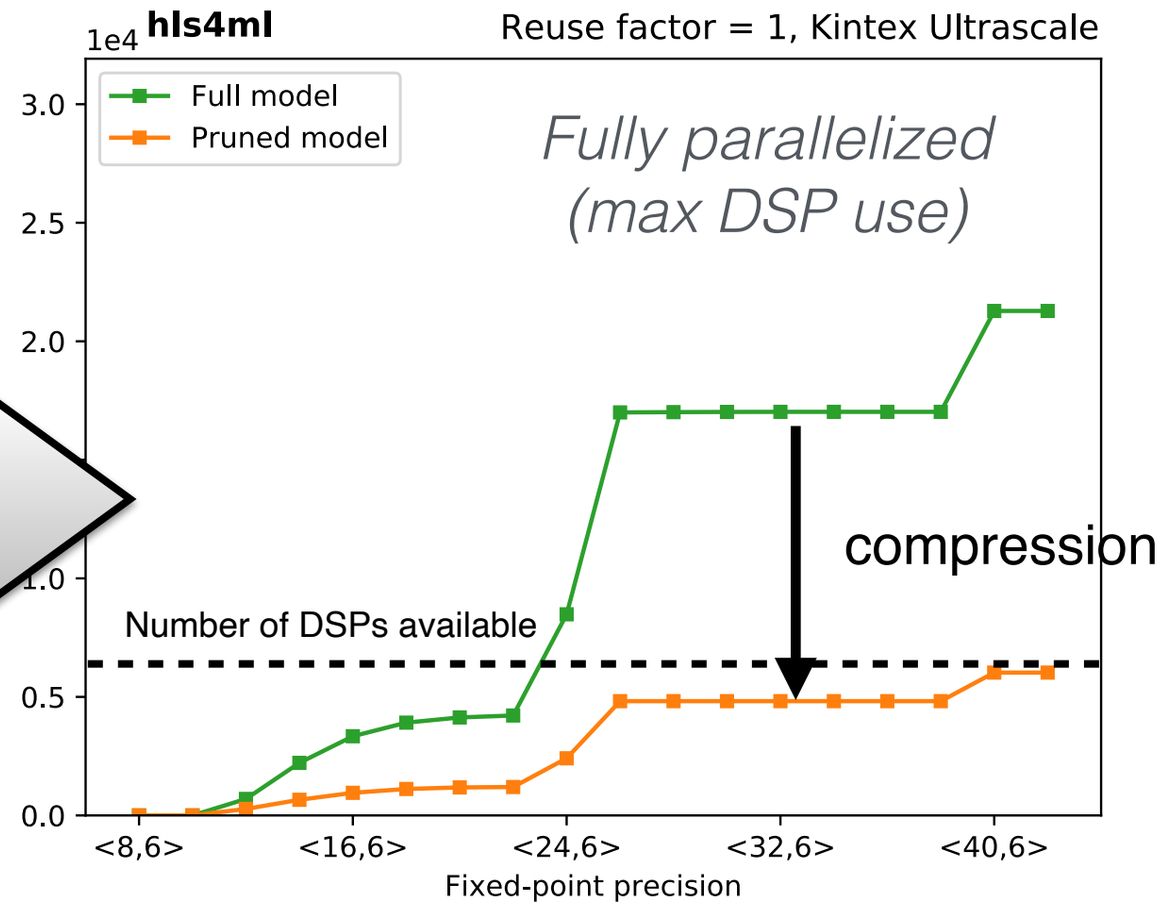


ex: Xilinx Virtex Ultrascale +

Make the model fit on one chip

- Some tricks are needed here:
  - **Compression/pruning:** remove the connections that play little role for final decision

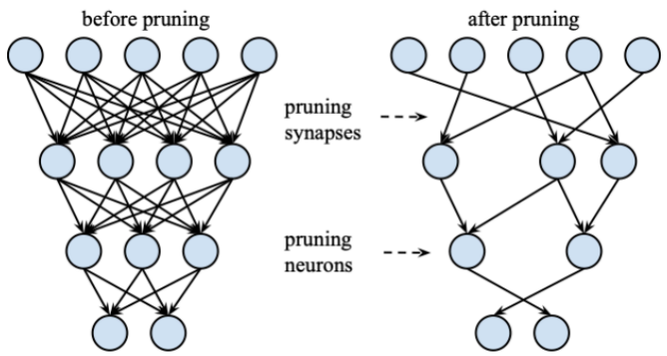


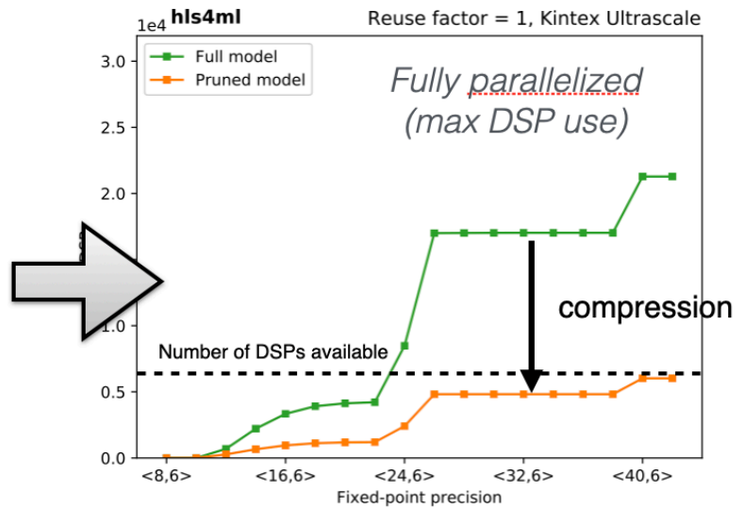


70% compression ~ 70% fewer DSPs

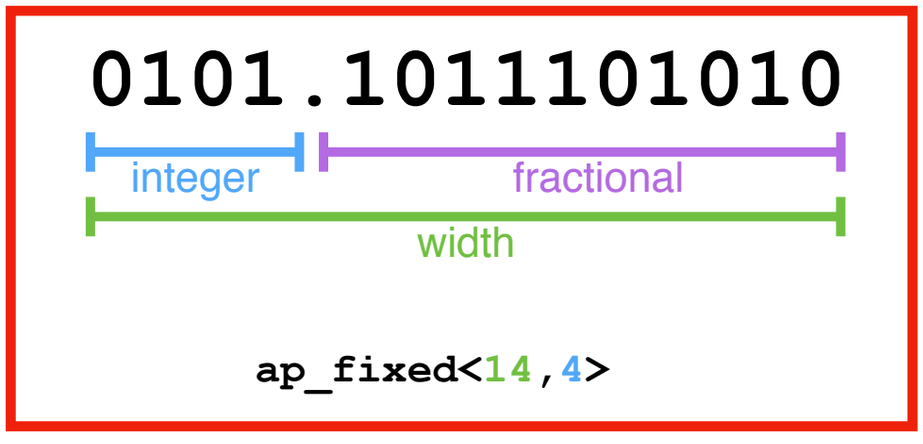
Make the model fit on one chip

- Some tricks are needed here:
  - **Compression/pruning:** remove the connections that play little role for final decision



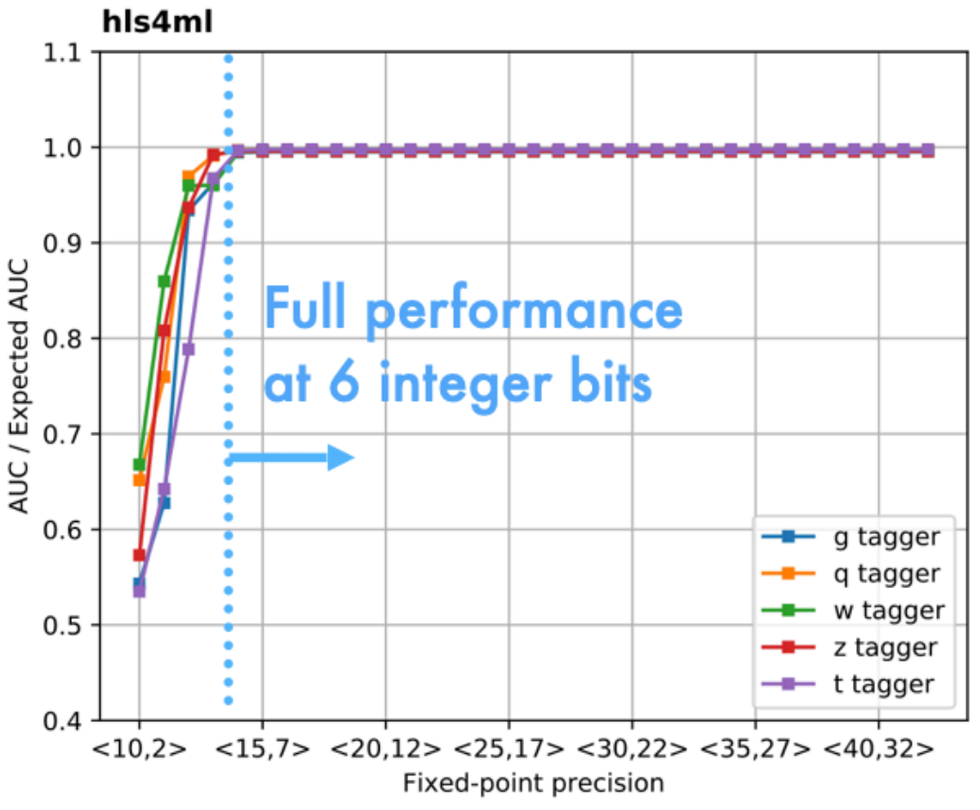


- **Quantisation:** represents numbers with few bits reduce resources



Scan integer bits

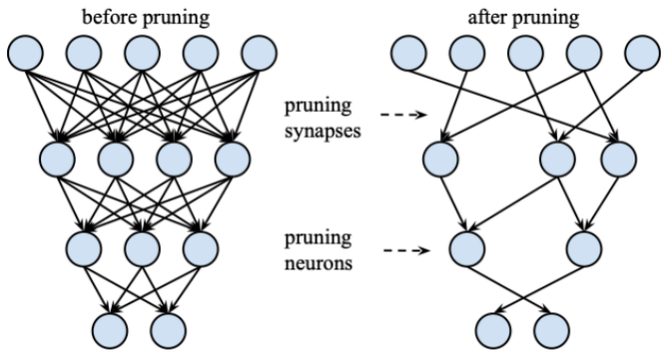
Fractional bits fixed to 8

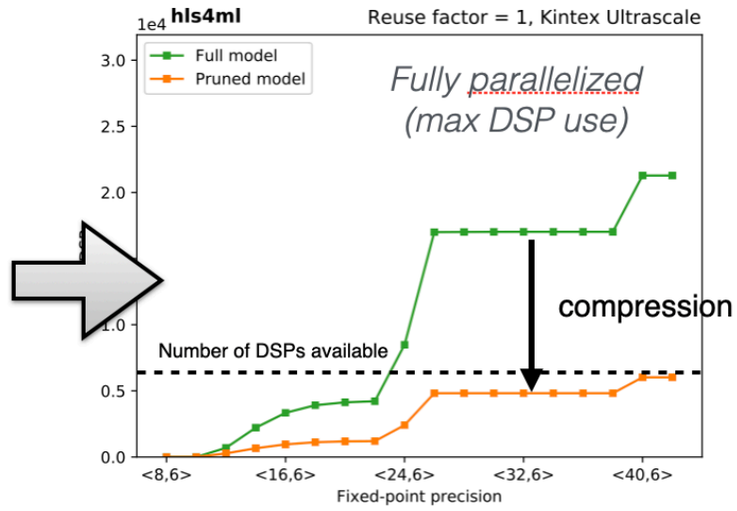


Make the model fit on one chip

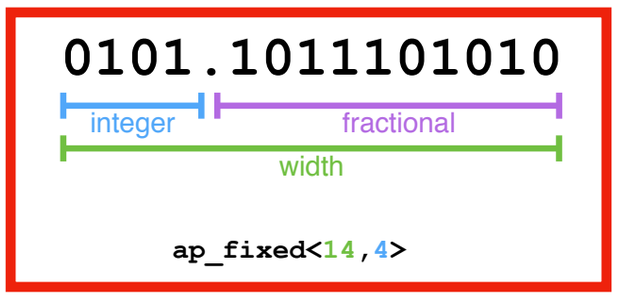
- Some tricks are needed here:

  - **Compression/pruning:** remove the connections that play little role for final decision

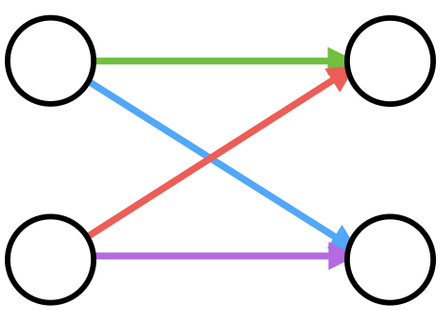


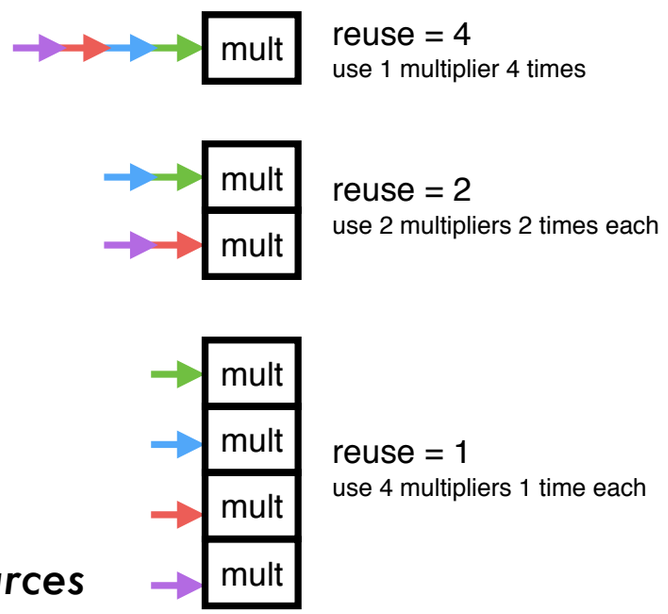


  - **Quantisation:** represents numbers with few bits reduce resources



  - **Parallelization:** allocate resources for each operation (run all network in one clock) vs spread calculation across several clock cycles



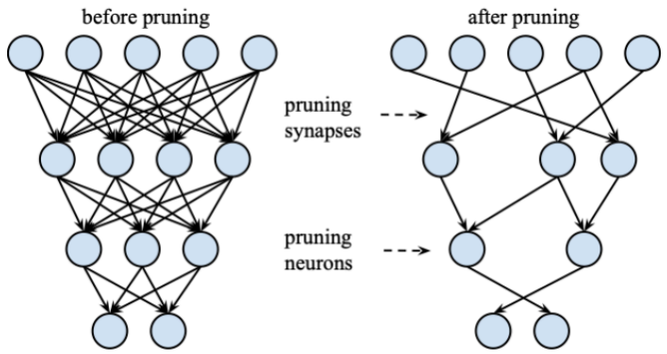


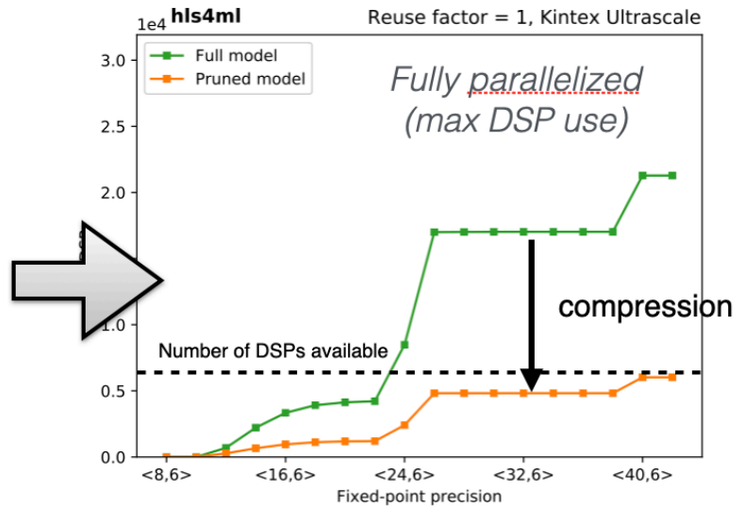
more parallelization → more resources

Make the model fit on one chip

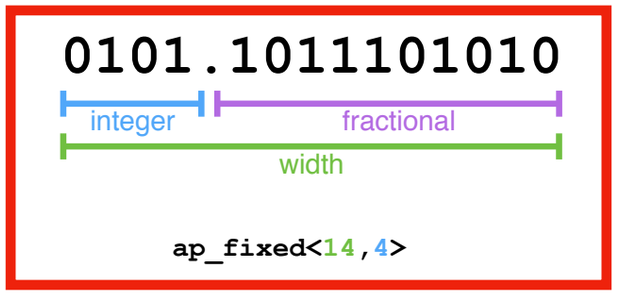
- Some tricks are needed here:

  - **Compression/pruning:** remove the connections that play little role for final decision

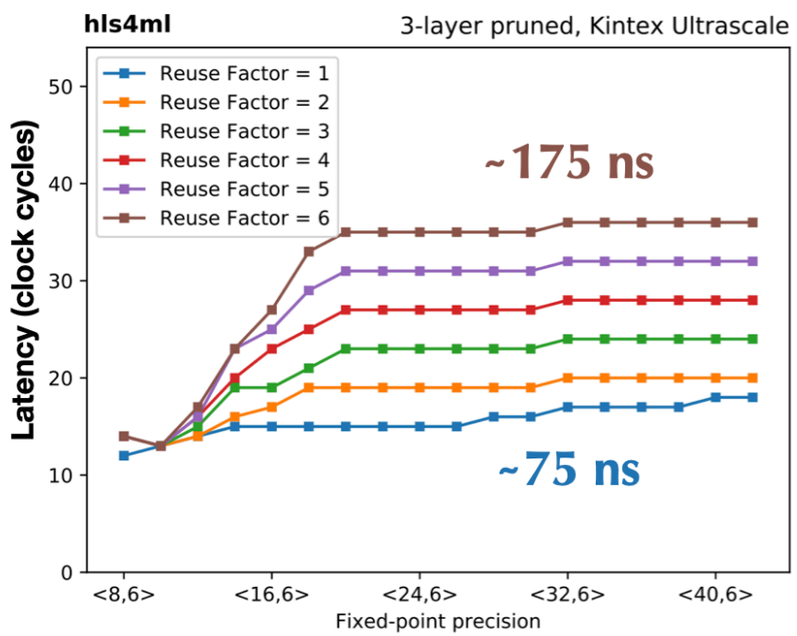


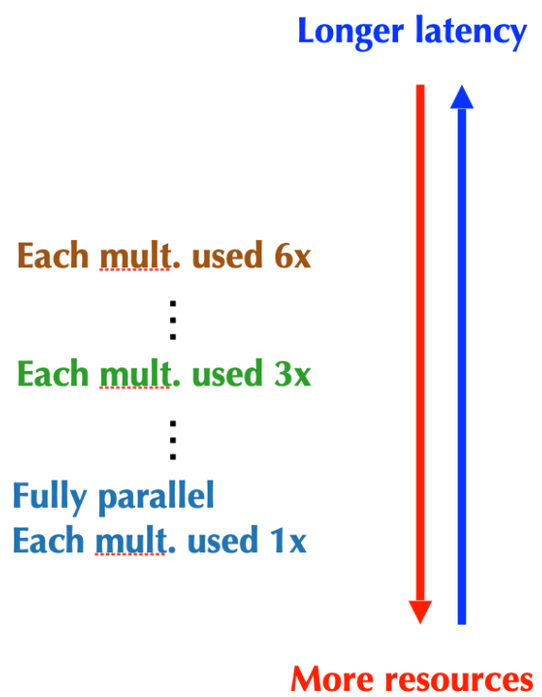


  - **Quantisation:** represents numbers with few bits reduce resources



  - **Parallelization:** allocate resources for each operation (run all network in one clock) vs spread calculation across several clock cycles

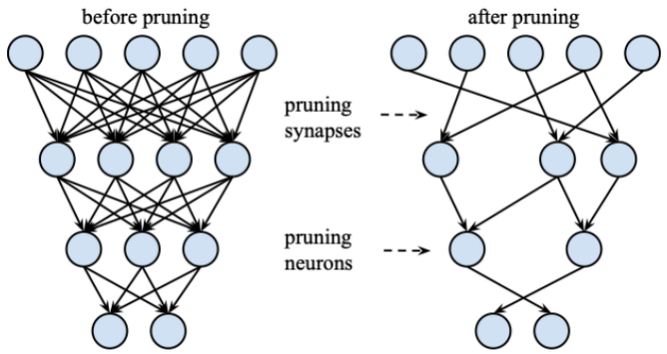


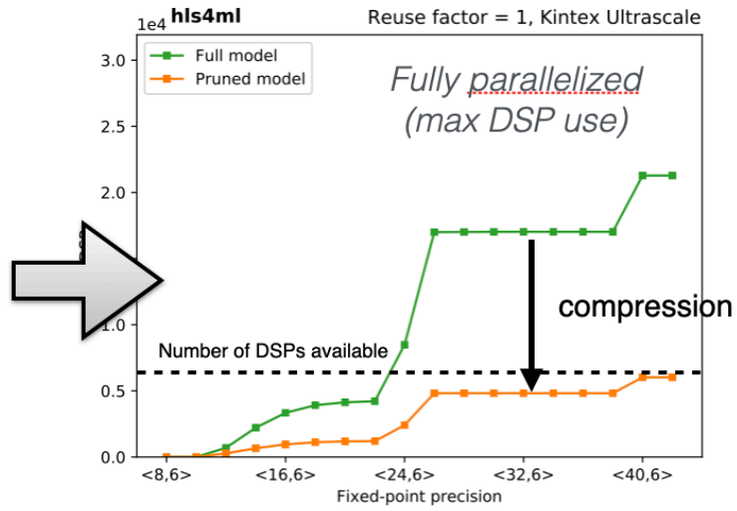


Make the model fit on one chip

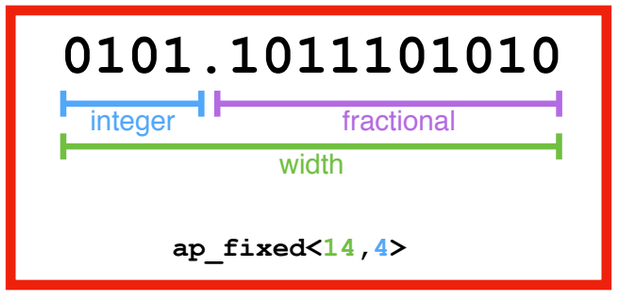
- Some tricks are needed here:

  - **Compression/pruning:** remove the connections that play little role for final decision

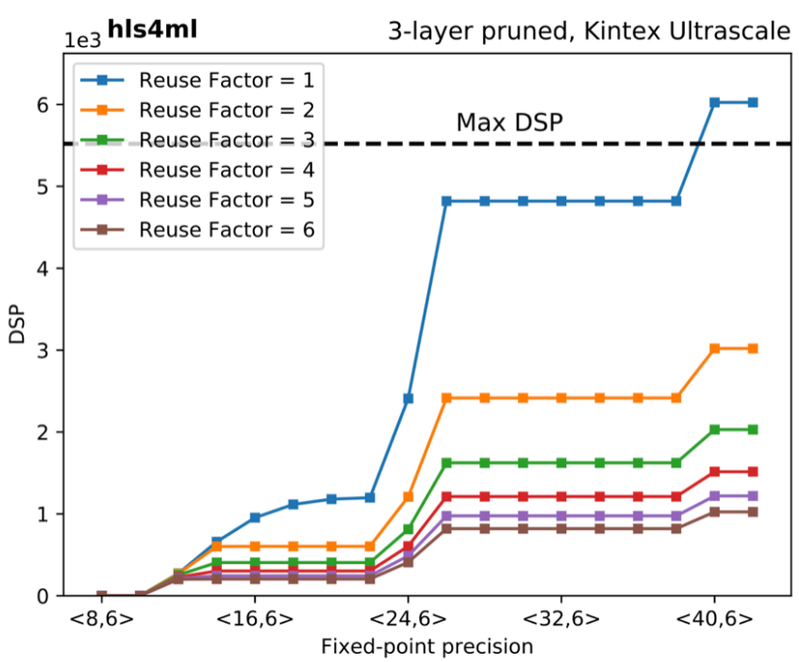


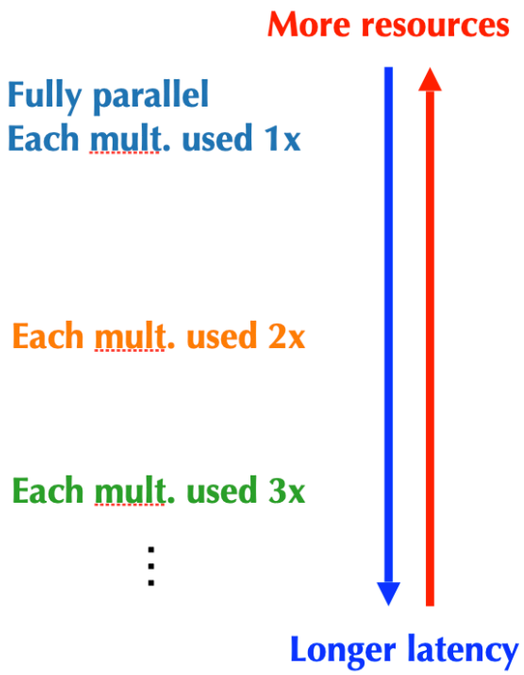


  - **Quantisation:** represents numbers with few bits reduce resources



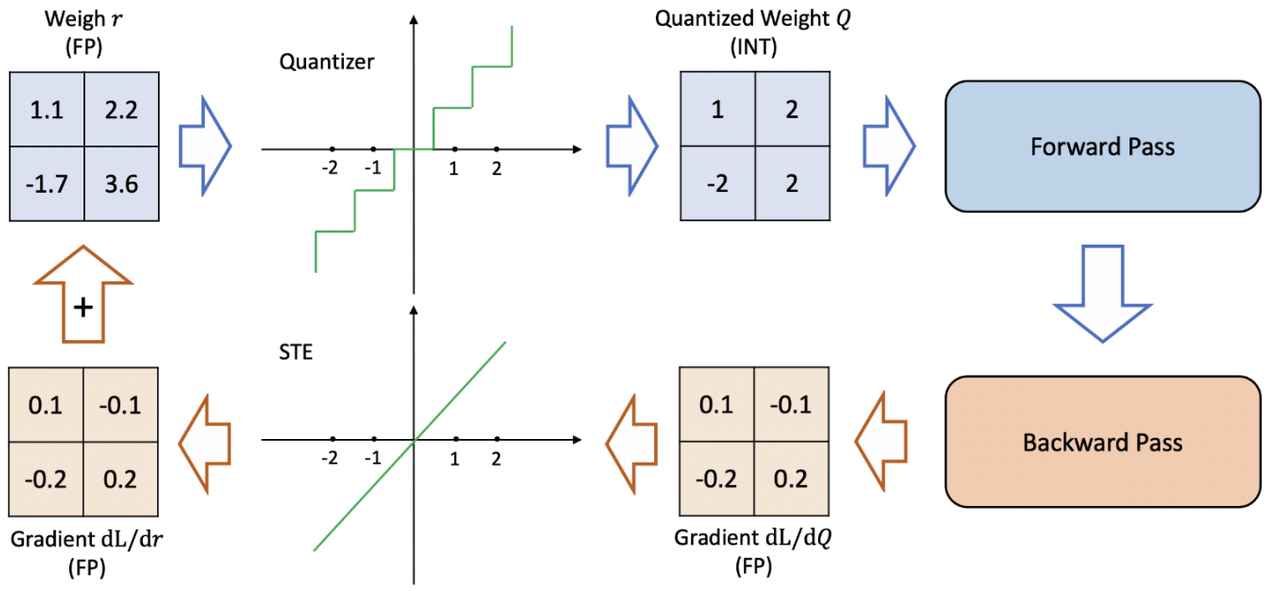
  - **Parallelization:** allocate resources for each operation (run all network in one clock) vs spread calculation across several clock cycles





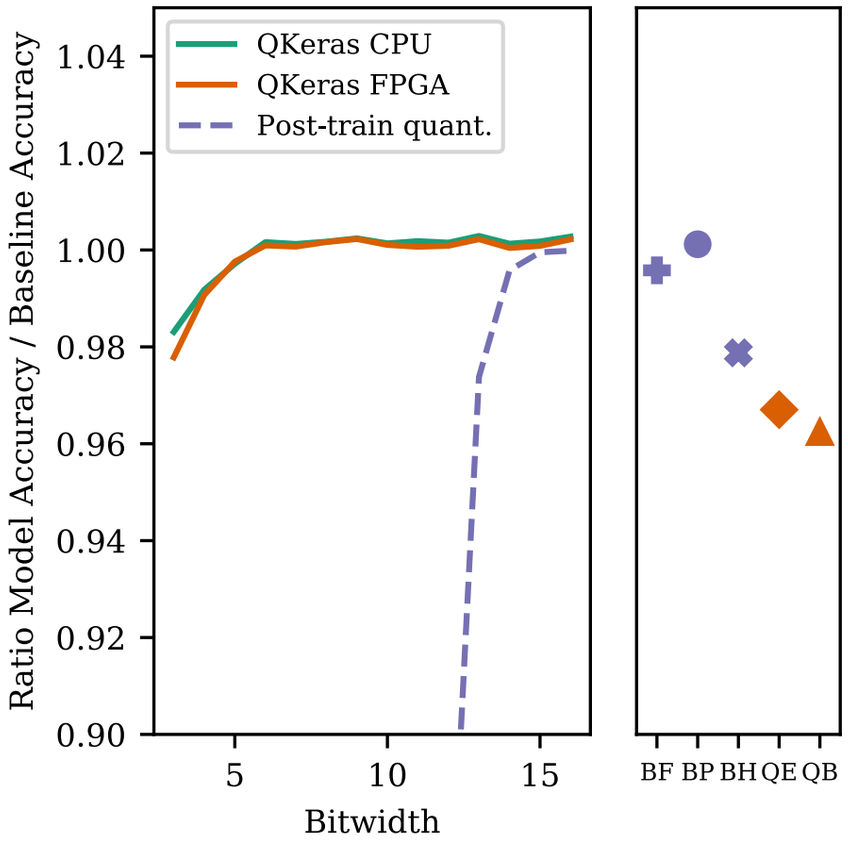
Quantization-aware training

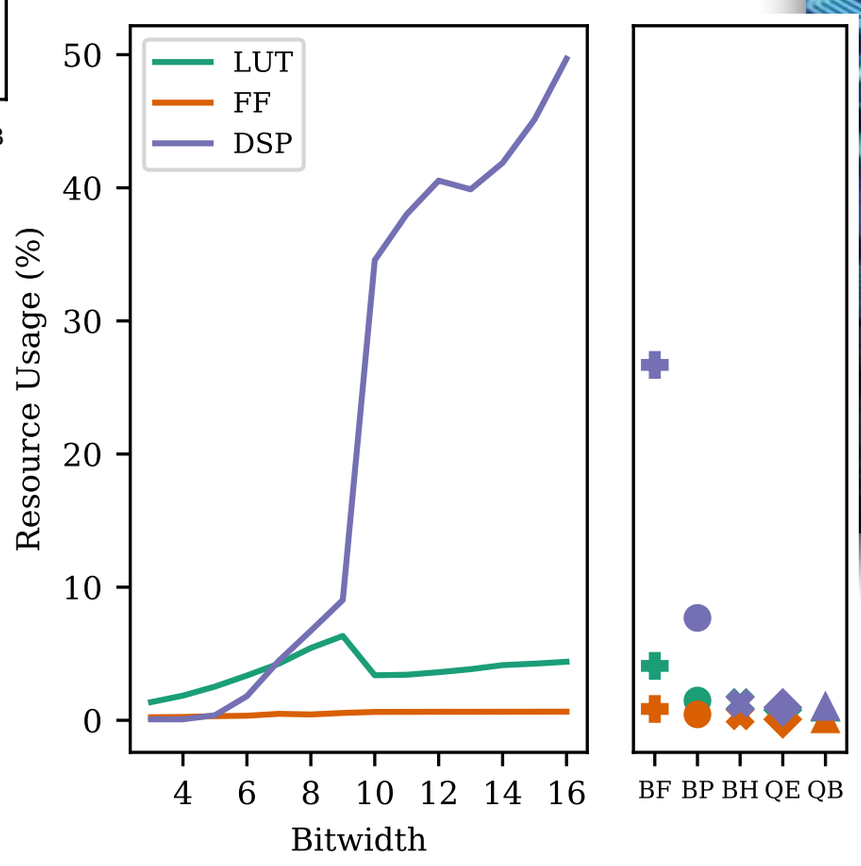
- **Post-training quantization can affect accuracy**
  - for a given bit allocation, the loss minimum at floating-point precision might not be the minimum anymore
- One could **specify quantization while look for the minimum during training**
  - quantization functions applied to weights and activations only in the forward pass
    - use Straight Through Estimator for back propagation step
- **Our workflow:** quantization-aware training with `Google QKeras` and firmware design with `hls4ml` for most efficient NN inference on chip!

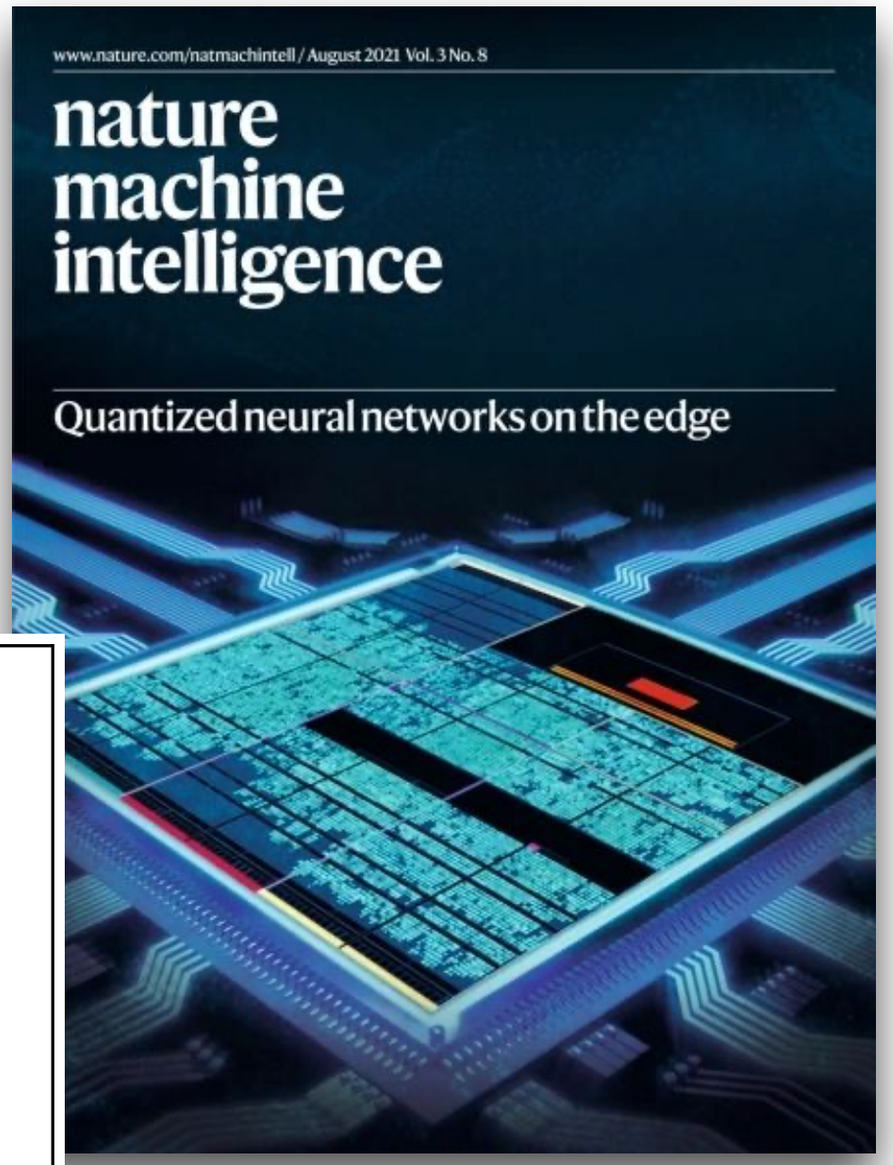


A. Gholami et al, arxiv.2103.13630

QKeras & hls4ml







Matches ap_fixed exactly!
- same granularity as hls4ml
- same precision at training and inference

C. N. Coelho et al.: Nature Machine Intelligence, Volume 3 (2021)

High-Granularity Quantization

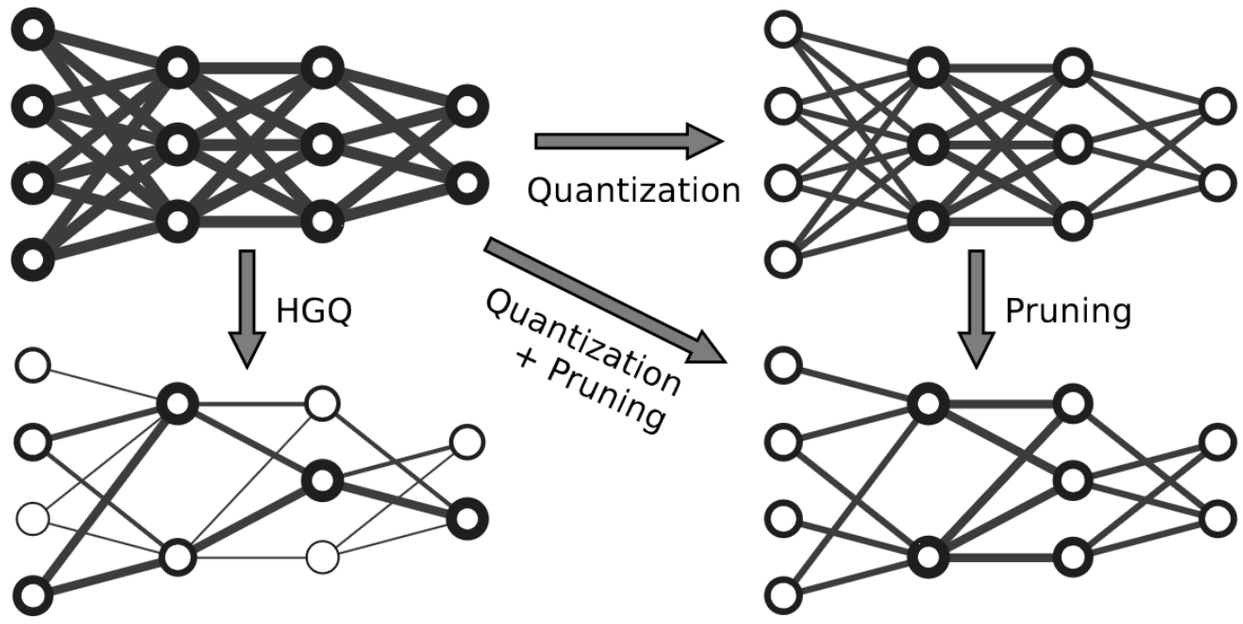
- **The wish:** squeeze even more NN inference performance when each parameter in the network may have its unique bitwidth

- **Limitations of QKeras:**

  - bitwidths for NN parameters are optimized in predefined, structured block (e.g., per layer)

  - bitdiwdth is not part of optimization
    → need to run your own hyperparameter scan

- **Solution: optimize the individual bitwidths alongside the NN accuracy using gradient descent**



High-Granularity Quantization

- **Solution: optimize the individual bitwidths alongside the NN accuracy using gradient descent**

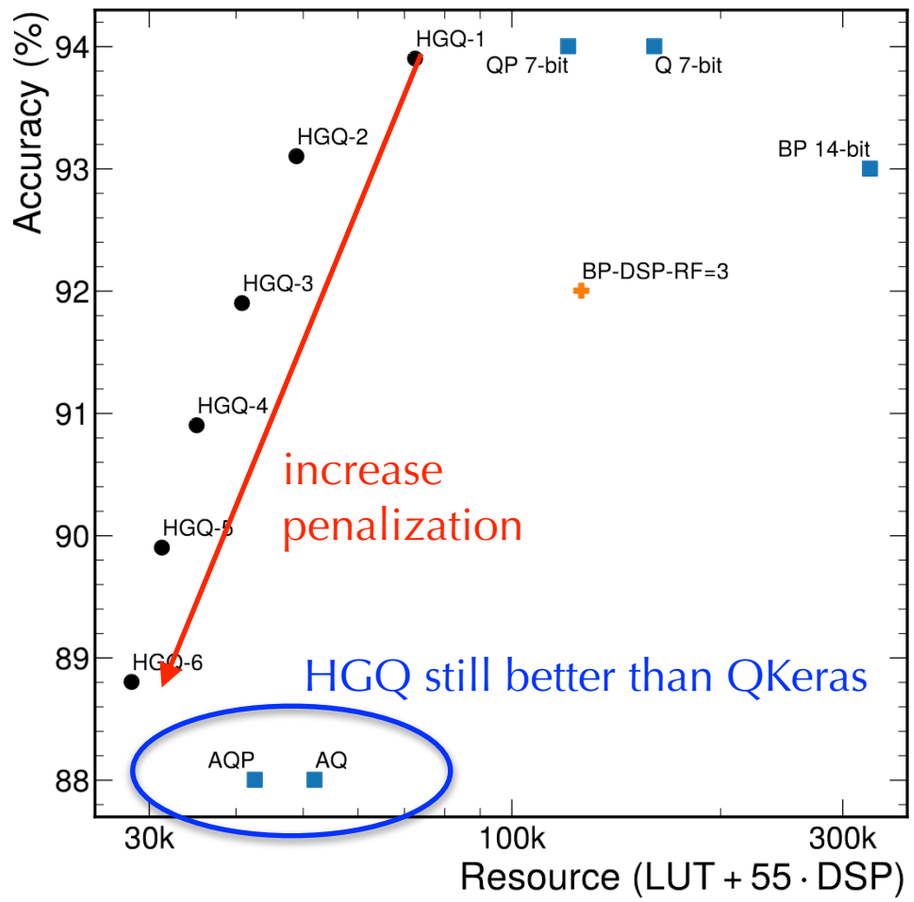
- **How:**

  - treat the bitwidths as continuous variables

  - introduce surrogate gradients for discrete variables such as bitwidths

  - introduce a novel on-chip resource consumption metric that when incorporated into the loss function penalizes larger bitwidths efficiently

  - pruning integrated naturally in the optimization step (gradient descent reduces certain bitwidths to zero)



arXiv.2405.00645 (FPGA '26)

Fully supported in hls4ml!

Ultra-fast anomaly detection @ CMS

Learn typicality: by training on unbiased dataset

CMS established in 2024 a new trigger paradigm with sub-μs autoencoders for anomaly detection!

AXOL1TL

	p_T	η	φ
MET		N/A	
4 e/γ			
4 μ			
10 jets			

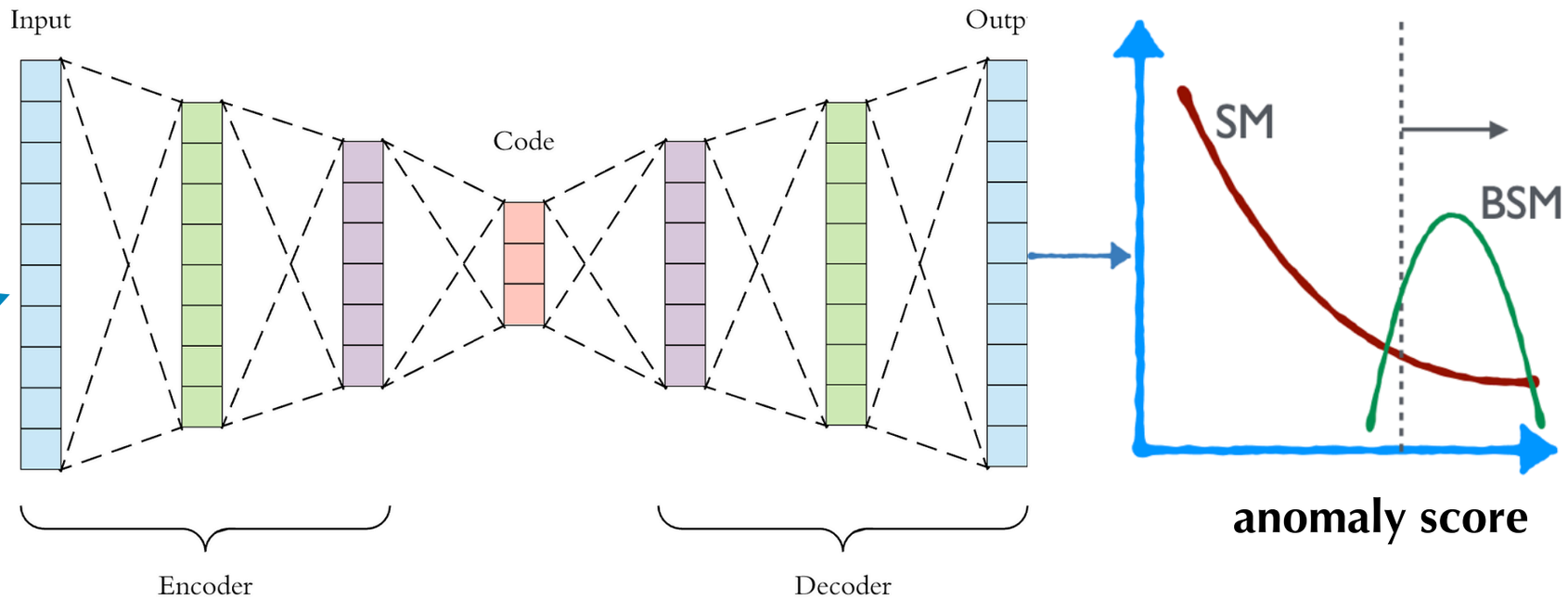
From calorimeter and muon trigger system:

CMS-DP-2023-079
CMS-DP-2024-059

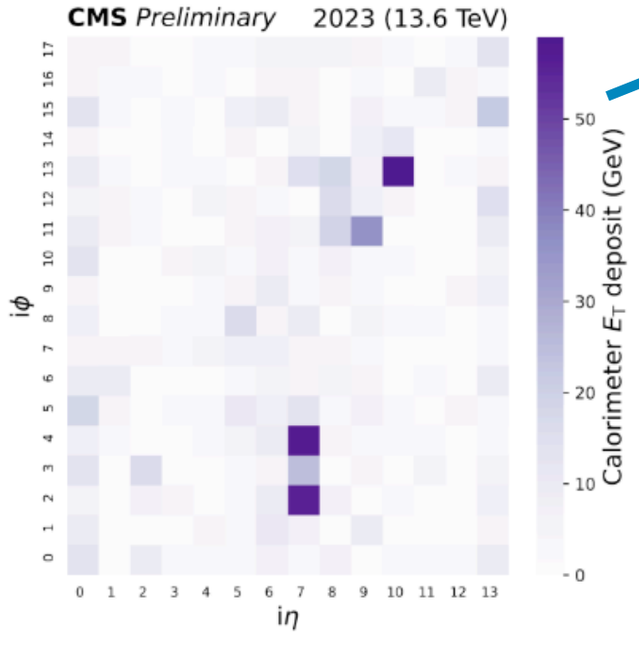
Objects: 10 jets, 4 muons, e/**γ**, MET

Features: p_T, η, ϕ (in raw integer values)

Architecture: MLP



CICADA



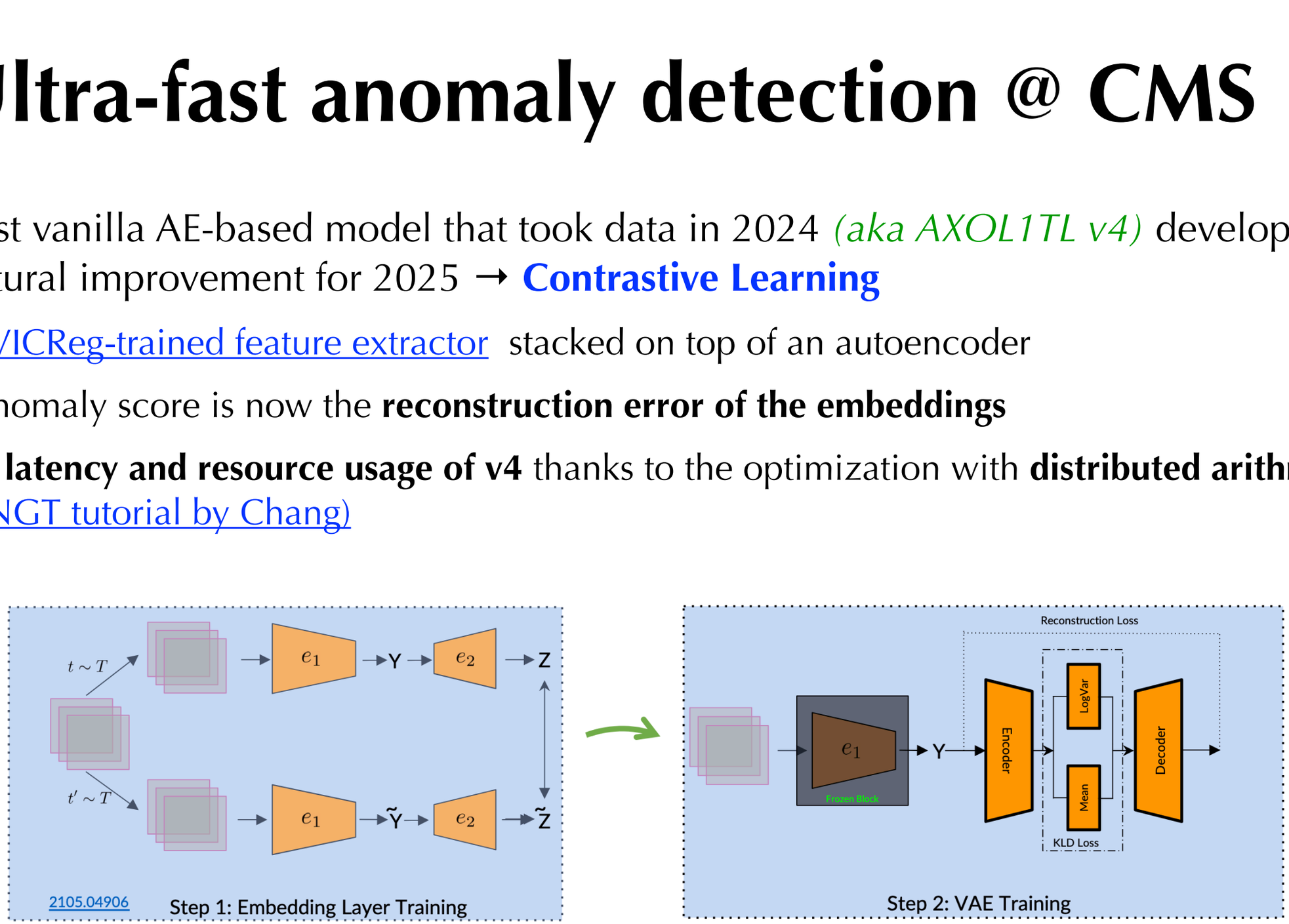
Low-level inputs: aggregated calorimeter towers

Architecture: 2D CNN w/ knowledge distillation

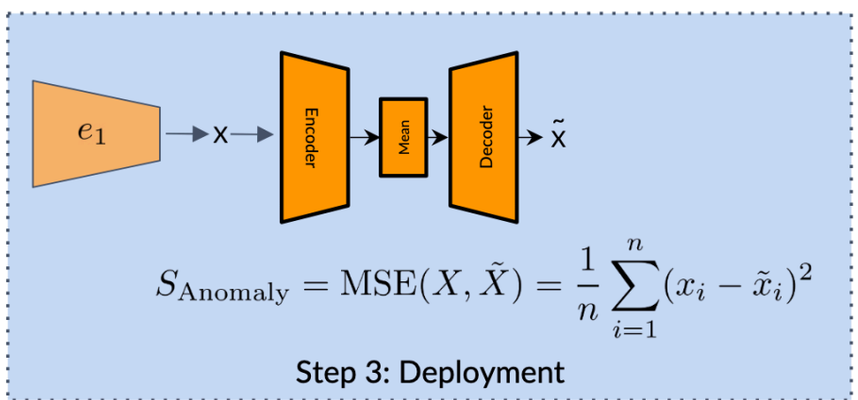
[CMS-DP-2023-086]

Ultra-fast anomaly detection @ CMS

- Since first vanilla AE-based model that took data in 2024 *(aka AXOL1TL v4)* developed architectural improvement for 2025 → **Contrastive Learning**
  - new VICReg-trained feature extractor stacked on top of an autoencoder
  - the anomaly score is now the **reconstruction error of the embeddings**
  - **same latency and resource usage of v4** thanks to the optimization with **distributed arithmetic** (see NGT tutorial by Chang)

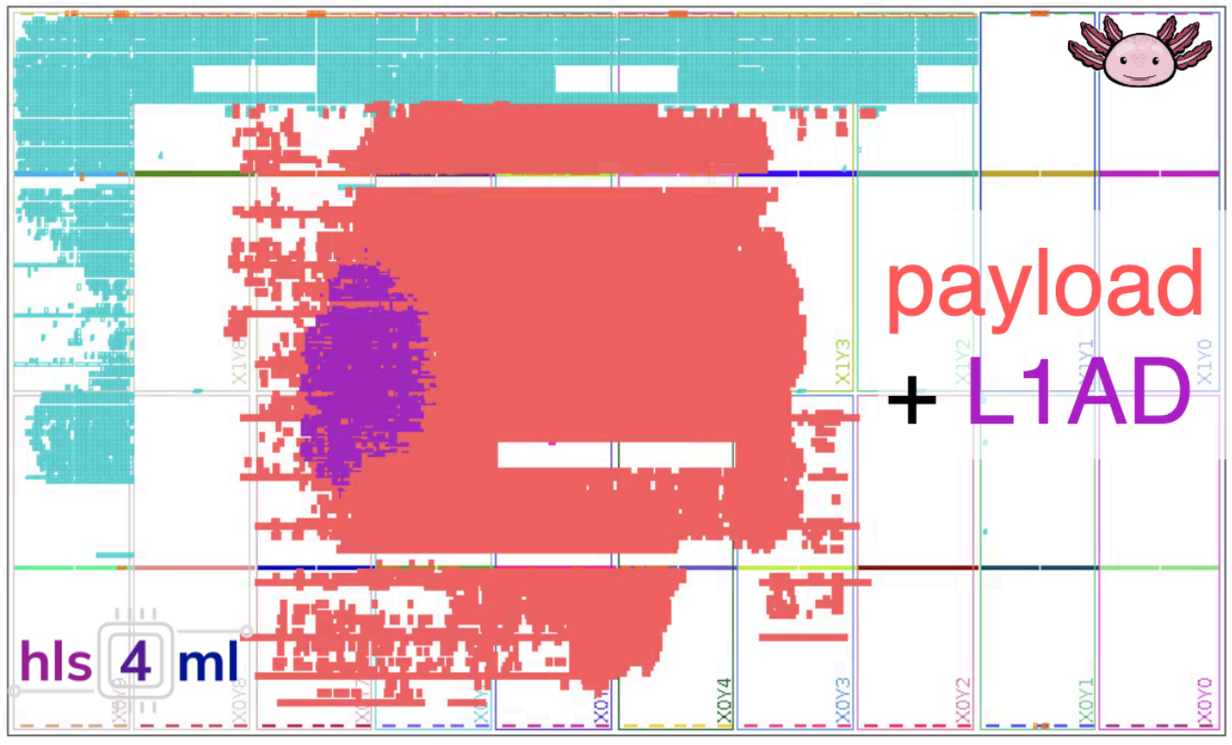


Augmentations considered:
gaussian smearing in within reconstruction resolutions and objects masking



CMS-DP-2025-061

Ultra-fast anomaly detection @ CMS



Model	Latency	LUTs (% w.r.t GT)	FFs (% w.r.t GT)	DSPs
AXOL1TL V4	2 clocks, 50 ns	18961 (4.3%)	424 (0.05%)	0
AXOL1TL V5 (w/ da4ml)	2 clocks, 50 ns	22023 (5.0%)	627 (0.07%)	0
AXOL1TL V5 (w/o da4ml)	2 clocks, 50 ns	30583 (7.1%)	965 (0.11%)	0
GT	160 clocks, 4 us	433,200	866,400	-

CMS Experiment at the LHC, CERN

Data recorded: 2023-May-24 01:42:17.826112 GMT

Run / Event / LS: 367883 / 374187302 / 159

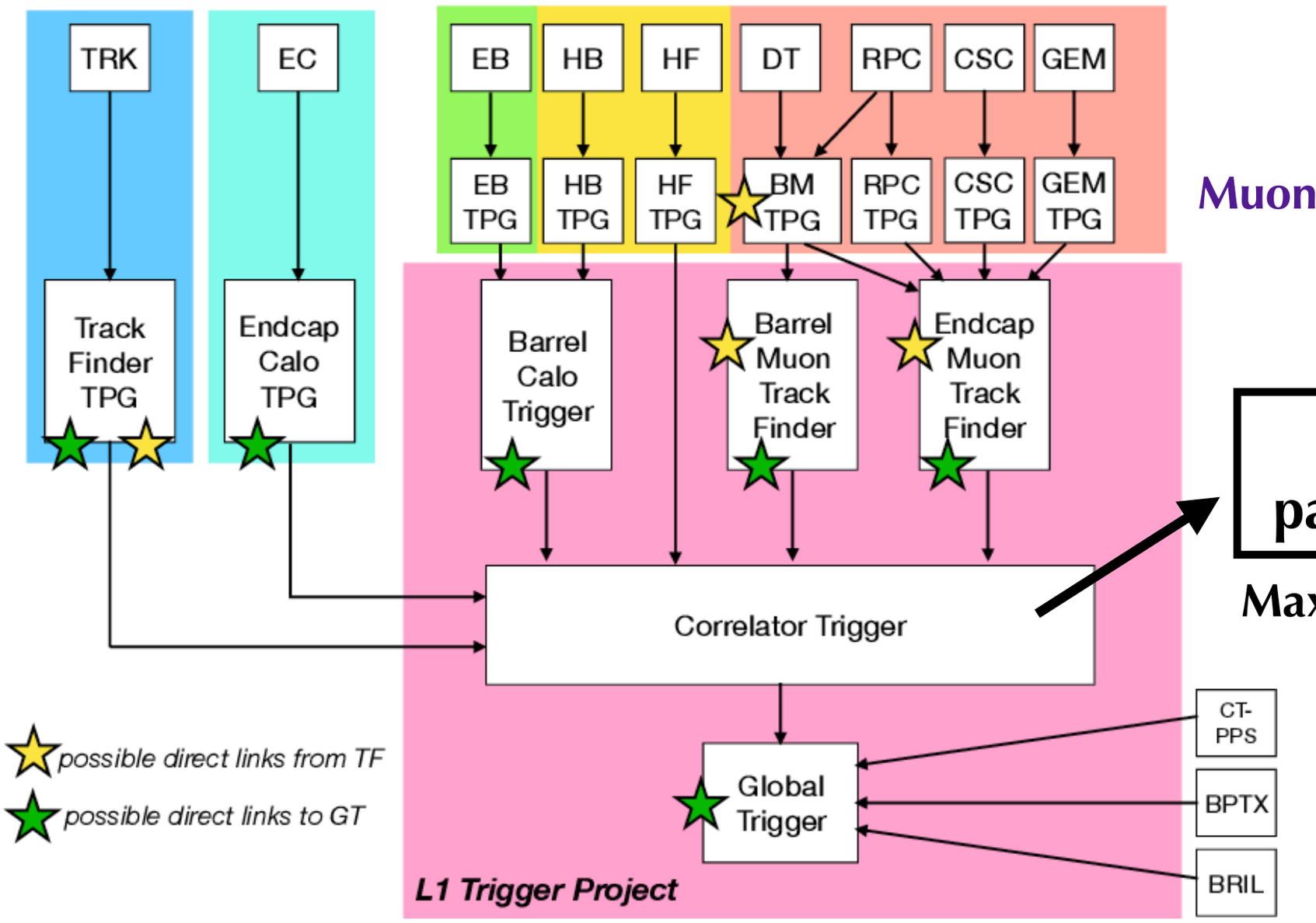
An otherwise untriggered high-multiplicity event!

The HL-LHC challenge: CMS Phase 2

At HL-LHC, up to 200 pile-up interactions: *CMS is upgrading the L1T and HLT to enable the same physics program we are doing now (at @60 PU)*

40 MHz tracking!

*** input data from 2 Tb/s to 63 Tb/s**
*** latency of 12.5μs to take decision**



40 MHz particle flow!

Max of 128 particles available

CMS-TDR-021

The HL-LHC challenge: CMS Phase 2

At HL-LHC, up to 200 pile-up interactions: *CMS is upgrading the L1T and HLT to enable the same physics program we are doing now (at @60 PU)*

With significantly more powerful compute we expect ML to be well embedded into L1T to exploit higher information granularity:

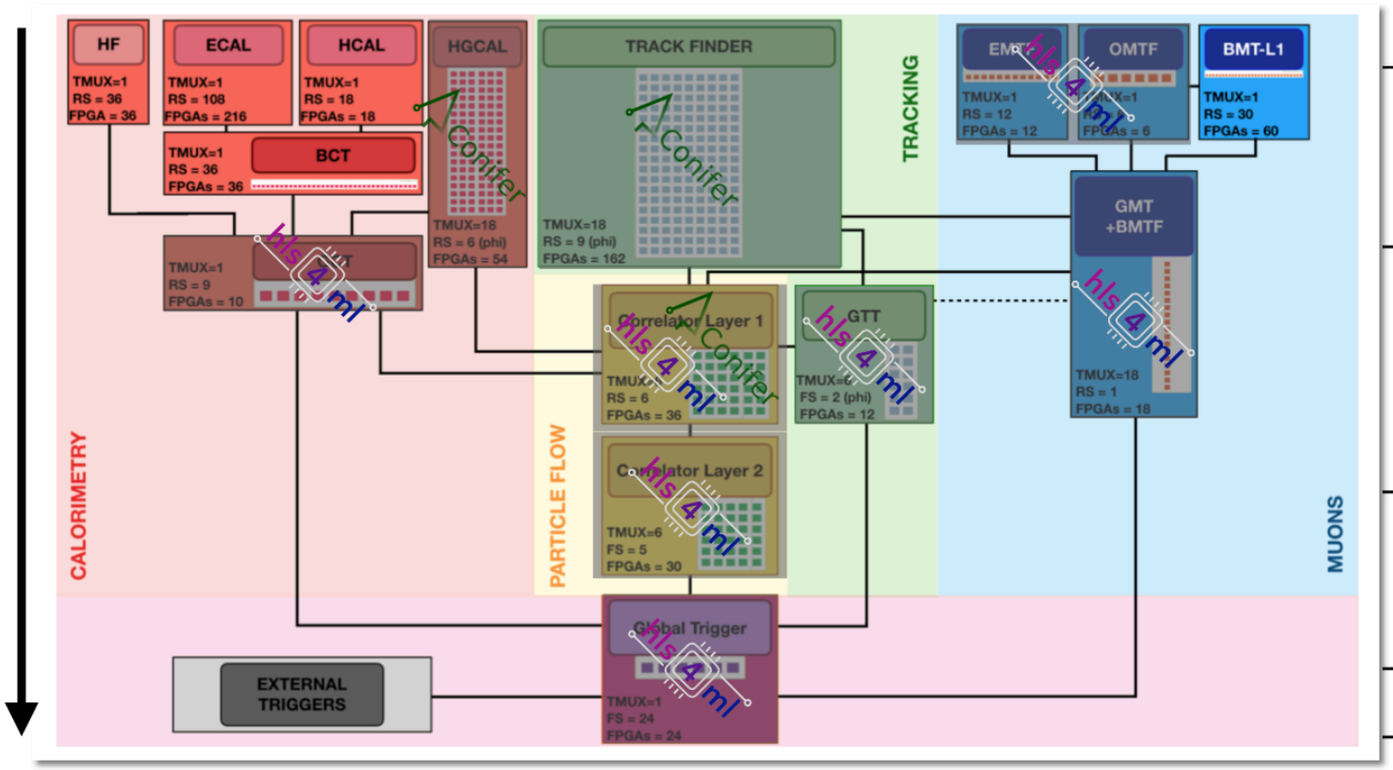
Around 20 projects (NNs, BDTs) in development accounting for 25 billion ML inferences per second

t = 0

Each small box is one Xilinx Ultrascale+ FPGA

These have NNs and/or BDTs inside

hls4ml Conifer



t = 12.5 µs

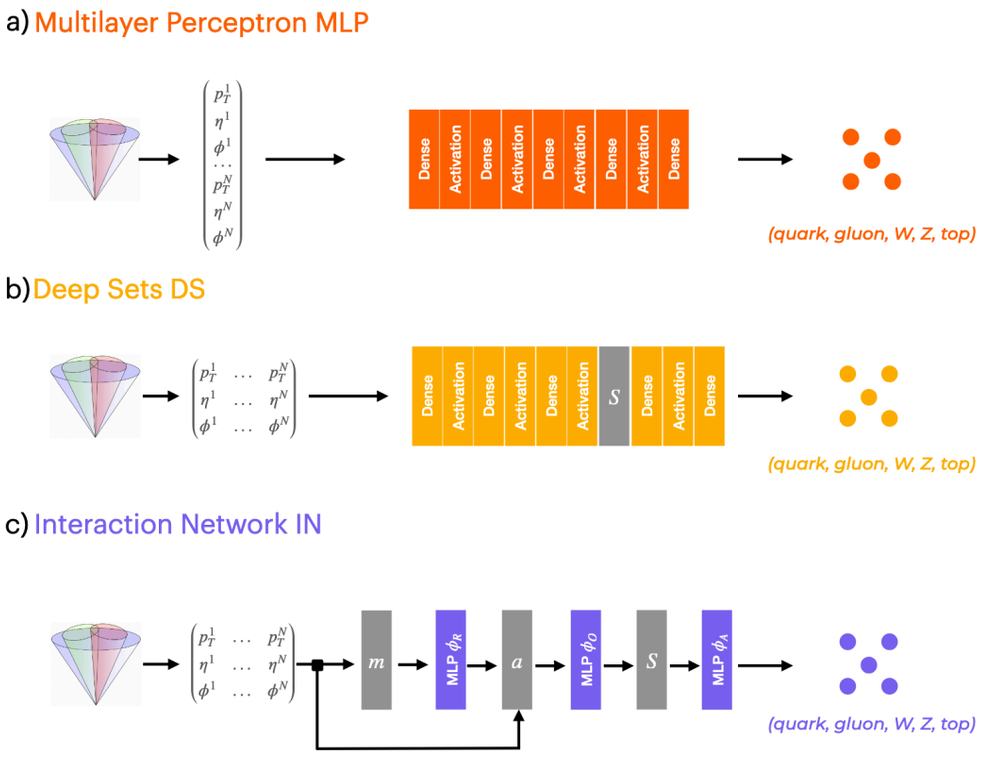
- Detector hits
- Clusters & Tracks
- Particles
- Event Categorisation
- 1 bit: keep / discard

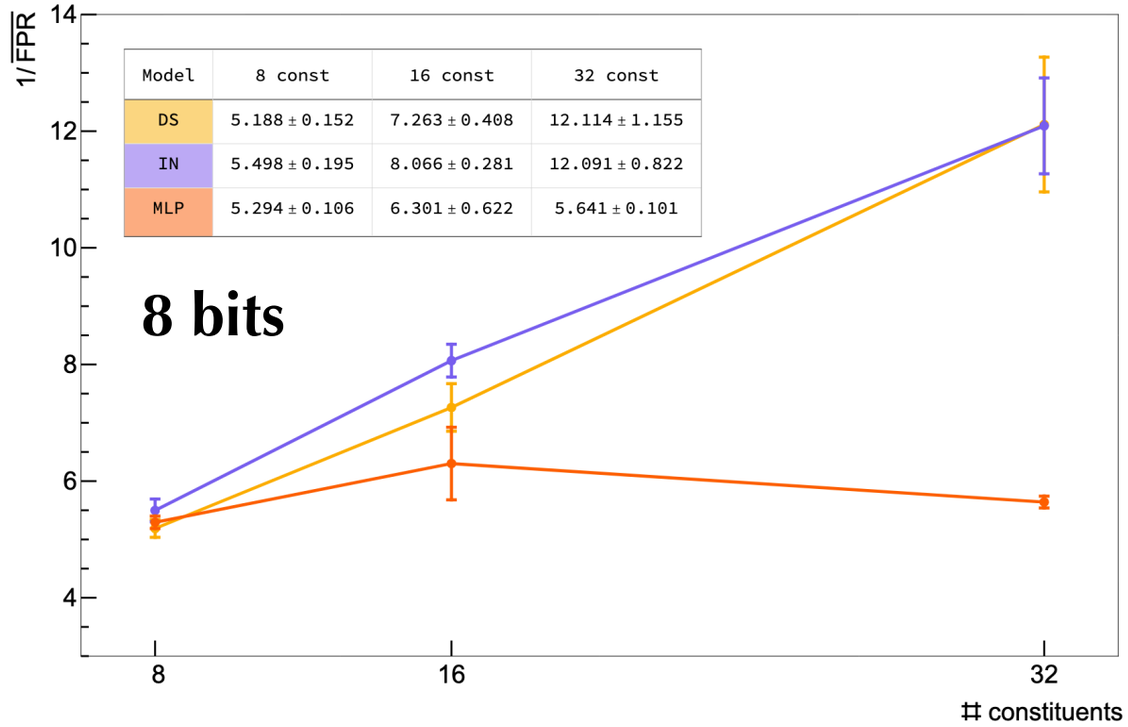
Credits: Sioni Summers (CERN)

Finding the best NN architecture

- **At offline level: chose the architecture with highest accuracy even if not efficient…**
- For edge applications this is not an option: crucial to **co-design the architecture with the application and its constraints**

arXiv.2402.01876





Model	8 const	16 const	32 const
DS	5.188 ± 0.152	7.263 ± 0.408	12.114 ± 1.155
IN	5.498 ± 0.195	8.066 ± 0.281	12.091 ± 0.822
MLP	5.294 ± 0.106	6.301 ± 0.622	5.641 ± 0.101

Graph NNs at O(100) ns latency!

hls4ml

FPGA: Xilinx Virtex UltraScale+ VU13P

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%)

Finding the best NN architecture

arXiv.2503.03103

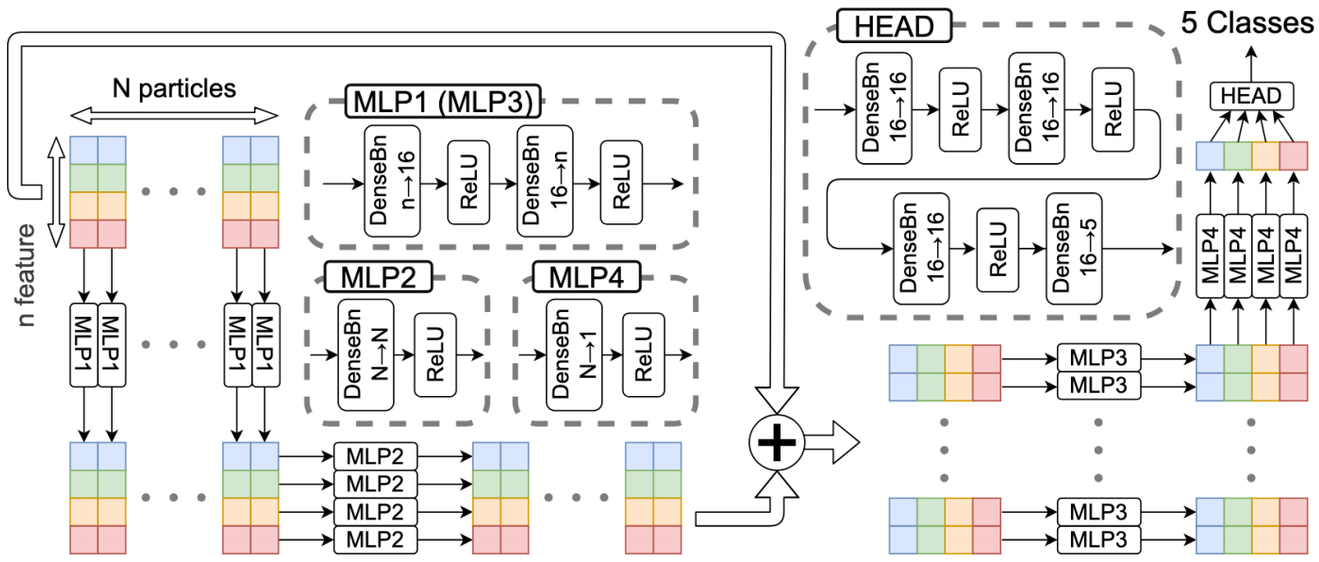
- Also found more recently that **MLP-mixer architecture** can further enhance performance

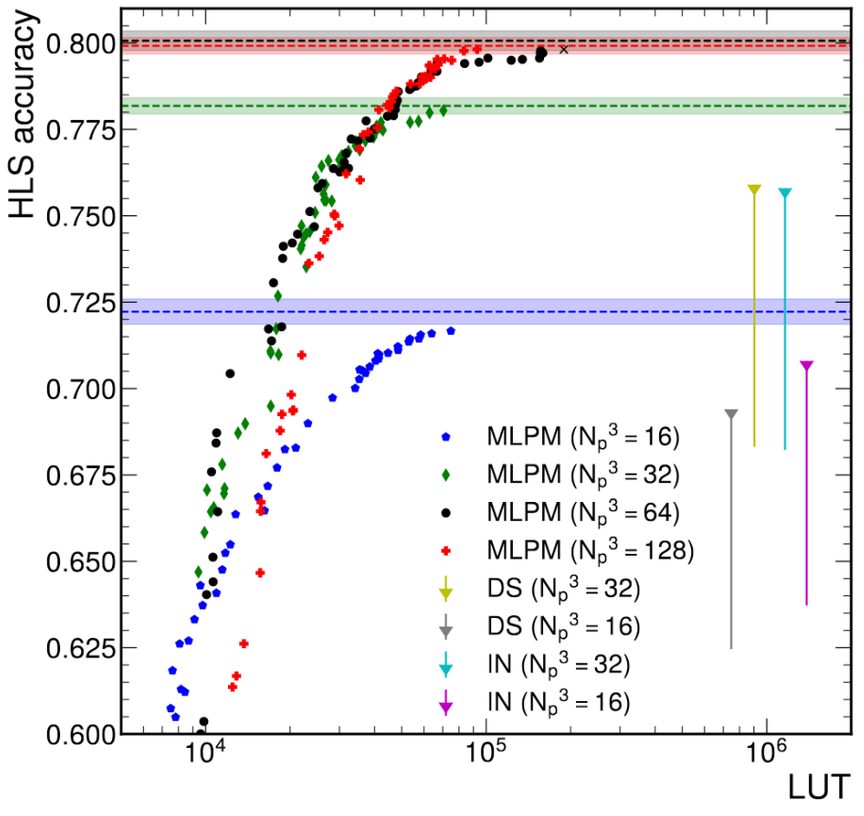
- Processes sets (like particle clouds or jet constituents) using only MLPs and alternates between two key components:

  - Token-mixing MLP
    – mixes information across particles (rows),

  - Channel-mixing MLP
    – mixes features within each particle (columns)

- **Not naturally permutation-invariant**
  → not necessarily needed for ordered sets (as in the trigger)
  → enables it to learn which features to retain and which to discard, facilitating HGQ





Finding the best NN architecture

arXiv.2503.03103

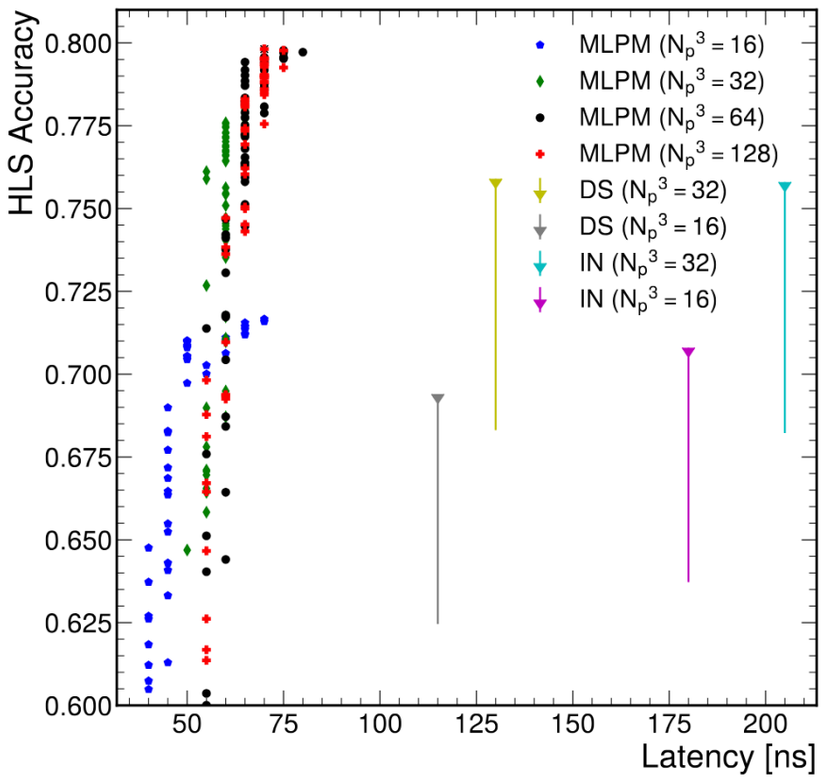
- Also found more recently that **MLP-mixer architecture** can further enhance performance

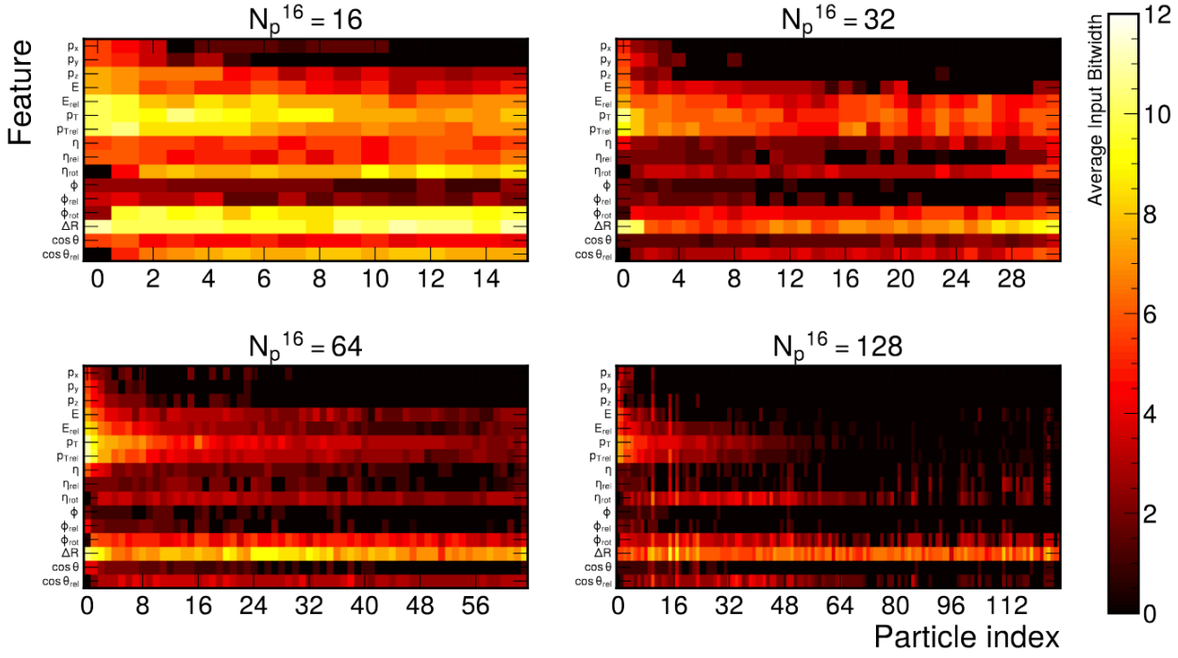
- Processes sets (like particle clouds or jet constituents) using only MLPs and alternates between two key components:

  - Token-mixing MLP
    – mixes information across particles (rows),

  - Channel-mixing MLP
    – mixes features within each particle (columns)

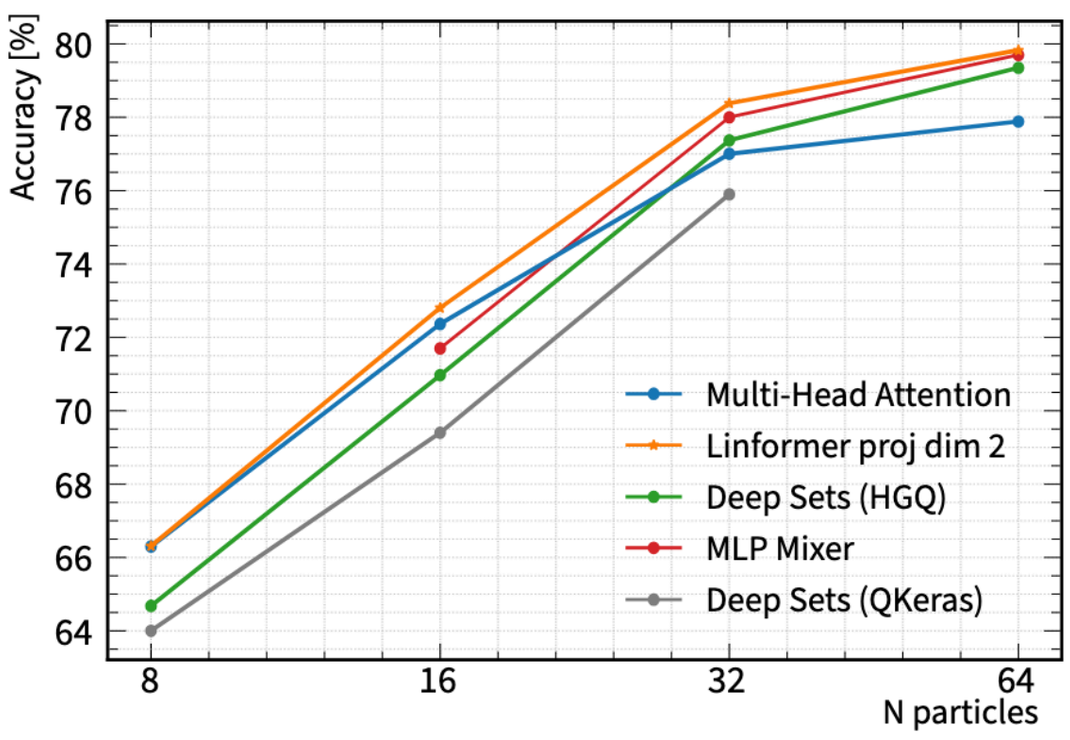
- **Not naturally permutation-invariant**
  → not necessarily needed for ordered sets (as in the trigger)
  → enables it to learn which features to retain and which to discard, facilitating HGQ





Towards ultra-fast transformers

- Implemented in HLS a simple encoder-only transformer architecture with vanilla MHA, a single attention head and no positional encoding (**Set Transformer**)
- Compare to the more efficient **Linformer** with O(*k* x *n*) complexity instead of O(n^2)
- Use **HGQ** and **da4ml** for most efficient HLS implementation



Model	Particles	Acc. (%)	Latn. (ns)	LUT (k)	II (clk)	DSP
Multi-Head Attention	8	66.3	104	246	1	0
Multi-Head Attention	16	72.3	98	279	1	0
Multi-Head Attention	32	77.0	83	180	1	0
Multi-Head Attention	64	77.9	44	47	1	0
Linformer	8	66.3	110	230	1	0
Linformer	16	72.8	103	246	1	0
Linformer	32	78.4	140	267	1	0
Linformer	64	79.8	78	202	1	0
Deep Sets (HGQ)	8	64.7	49	177	1	0
Deep Sets (HGQ)	16	70.1	53	205	1	0
Deep Sets (HGQ)	32	77.4	53	256	1	0
Deep Sets (HGQ)	64	79.4	44	191	1	0
MLP Mixer [18]	16	71.7	68	75	1	0
MLP Mixer [18]	32	78.0	62	63	1	0
MLP Mixer [18]	64	79.7	72	159	1	0
Deep Sets (QKeras) [36]	8	64.0	95	386	3	626
Deep Sets (QKeras) [36]	16	69.4	115	747	3	555
Deep Sets (QKeras) [36]	32	75.9	130	903	2	434
Deep Sets M (QKeras) [37]	8	65.1	110	130	3	548
Deep Sets L (QKeras) [37]	8	66.6	135	337	3	2,458

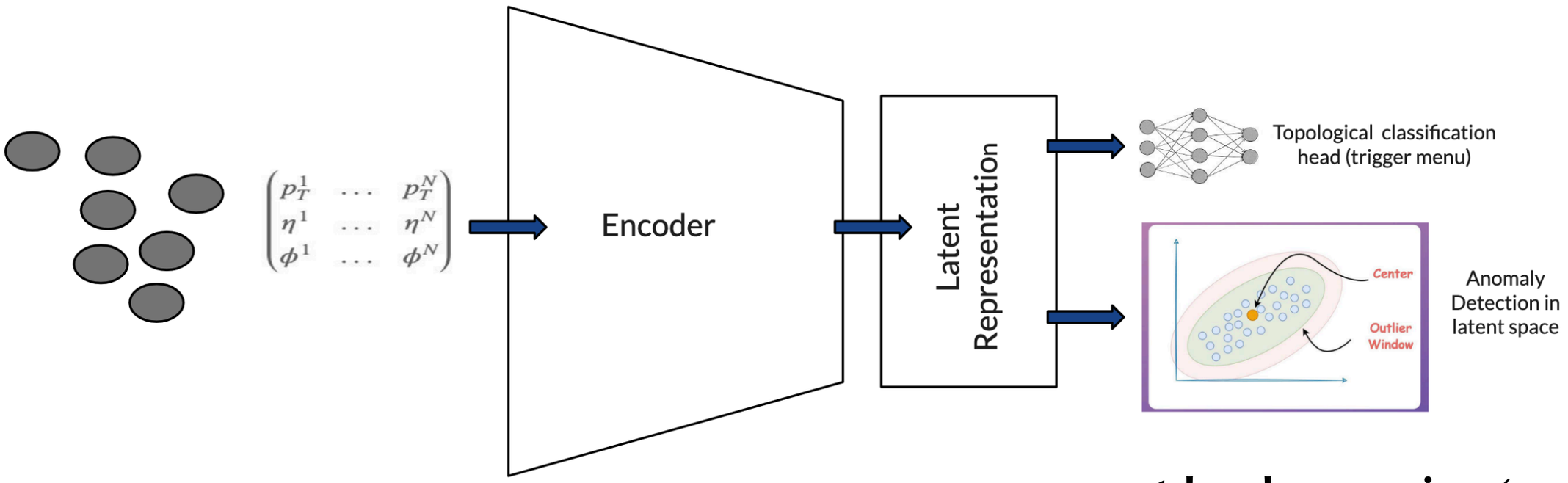
arXiv.2510.24784 (NeurIPs ML4PS '25)

Towards a (ultra-fast) foundation model

- With **more granular particle-level information and more compute in the Phase 2 L1T** we can think bigger

- In particular, **an expressive latent representation** can be re-used for multiple downstream tasks

  - could potentially reduce redundancy in hardware and deploy just one pre-trained model
    → can serve as a trigger for known and anomalous processes



+ event-level regression (e.g., MET)

Towards a (ultra-fast) foundation model

- With **more granular particle-level information and more compute in the Phase 2 L1T** we can think bigger

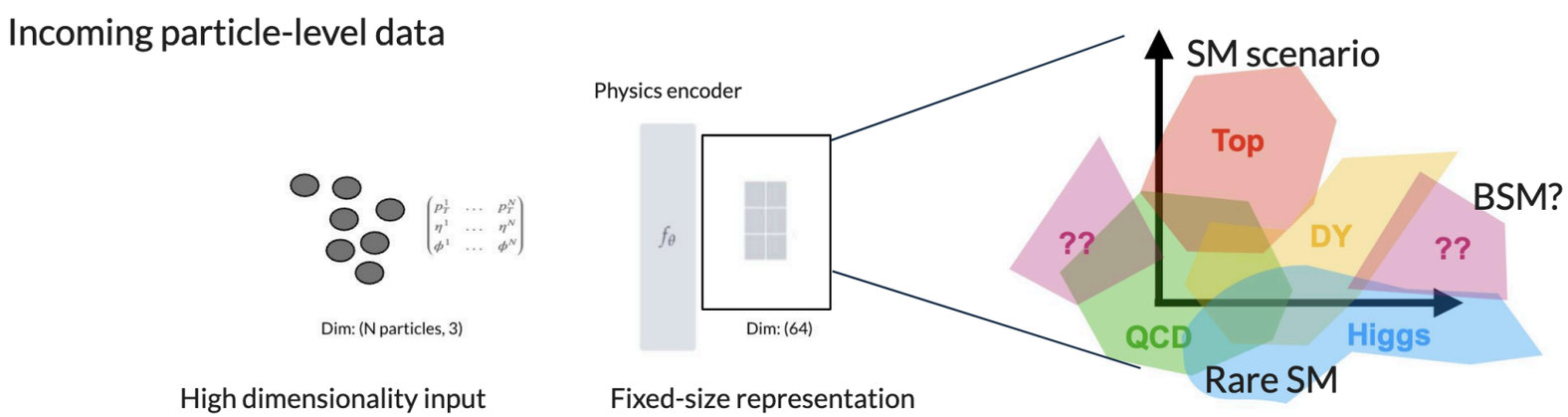
- In particular, **an expressive latent representation** can be re-used for multiple downstream tasks

  - could potentially reduce redundancy in hardware and deploy just one pre-trained model
    → can serve as a trigger for known and anomalous processes

- **Main idea: a physics informed latent space** that makes use of "all" known SM processes:

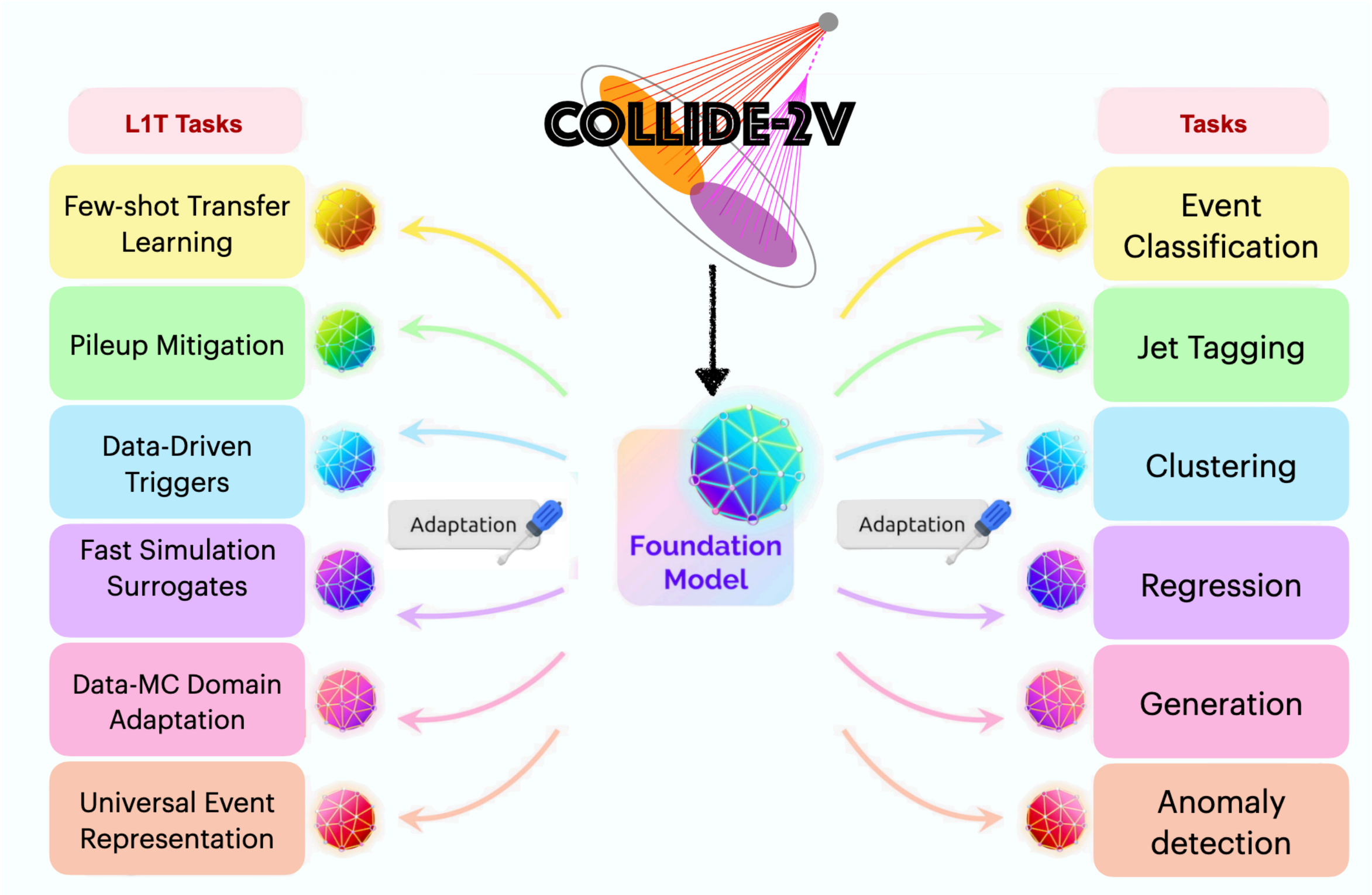
  - use **simulation** of diverse processes during training
  - allow the model to learn physical features across energy scales
  - create an event grouping based rather than pure outlier detection
  - use this compressed representation in the trigger decision

Offline data today first playground for this concept!



Credits: P. Harris

Towards a (ultra-fast) foundation model



Processes

Legend: Process | Size (Millions)

Soft & QCD Multijet

Minbias / Soft QCD | 100
QCD Inclusive | 100
QCD bb (heavy-flavor enriched) | 25

Quarkonium Control Sample

$\Upsilon \to leptons$ | 25

Single-Higgs

Gluon Fusion ($ggH \to X$) | 10

$ggHbb, ggHcc, ggH\tau\tau, ggH\gamma\gamma, ggHgg, ggHZZ, ggHWW$

Vector-Boson Fusion (VBFH → X) | 10

$VBFHbb, VBFHcc, VBFH\tau\tau, VBFH\gamma\gamma, VBFHgg, VBFHZZ, VBFHWW$

Associated Higgs

VH (incl) (W/Z + H; decay-agnostic) | 10

Single Bosons, V+jets & DY

$W \to \ell\nu, W \to qq$ | 25
$DY \to \ell\ell$ | 25
$Z \to \nu\nu + jet$ | 25
$Z \to qq\ (uds), Z \to bb, Z \to cc$ | 25

Top-quark family

$t\bar{t}$ (all-had, semi-lep, all-lep) | 25
$t\bar{t}t\bar{t}$ (4-top) | 2
$t\bar{t}W$ (incl), $t\bar{t}Z$ (incl) | 5
$t\bar{t}H$ (incl) | 10

Photon Processes

γ (prompt photon + jets) | 25
$\gamma + V$ ($W\gamma/Z\gamma$-like) | 10
Tri-γ(3γ) | 2

Diboson & Triboson

WW (all-had, semi-lep, all-lep) | 3
WZ (all-had, semi-lep, all-lep) | 3
ZZ (all-had, semi-lep, all-lep) | 3
VVV (triboson) | 2

Di-Higgs (HH→final states)

$HH \to b\bar{b}b\bar{b}$ | 2
$HH \to b\bar{b}\tau\tau$ | 2
$HH \to b\bar{b}WW, HH \to b\bar{b}ZZ$ | 2
$HH \to b\bar{b}\gamma\gamma$ | 2

BSM Processes (planned)

Dark shower prompt (including SUEPs)
Dark shower semi-visible
Dark shower long lived
RPV multijets
$Z' \to qq/\tau\tau$
$H \to 2a \to 4b/4\tau/2b2\tau/4\gamma/2b2\mu$
Light dark photon to di-lepton/di-pion
Heavy neutral leptons

A large suite of SM processes to build a physics-informed latent representation that can be used for many different event-level tasks downstream!

Made available for testing ~ 10% of it on HuggingFace & public CERNBox with example notebook for data loader and preprocessing… give it a try and let us know!

More info in Eric's talk at FastML2025

Adaptability: AXOL1TL as case study

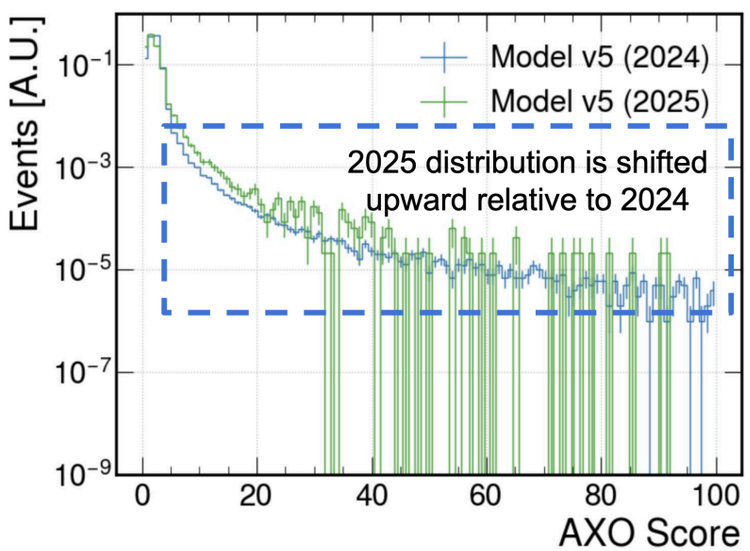
- Maintaining **an ML algorithm in a changing environment** could be challenging
  → it is first crucial to strengthen **MLOps practices but not only…**

  - particularly in view of Phase 2 when we expect a cascade of ML models and dependencies

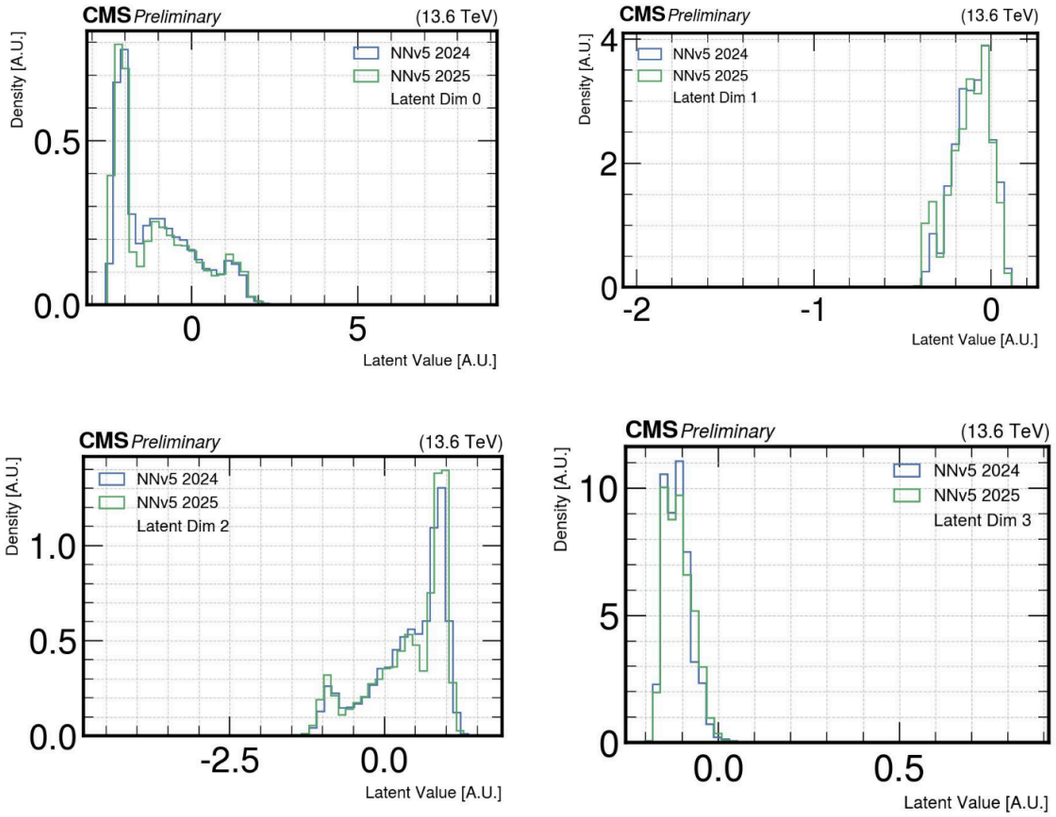
- **AXOL1TL provided a test bench TODAY**:

  - Two data taking period studied:
    2024 (used for training) and 2025

  - Between these periods, several electromagnetic calorimeter settings and object calibrations were updated



Output distribution of AXOL1TL NNv5 for 2024 data and 2025 data[5].



Latent distribution of the VAE[5]. Since the data semantics are unchanged, the probability density is expected to remain constant.

Adaptability: Re-training vs re-calibrating

- **RE-TRAINING: Collect a new dataset and retrain the model** (CMS initial approach with AXOL1TL)

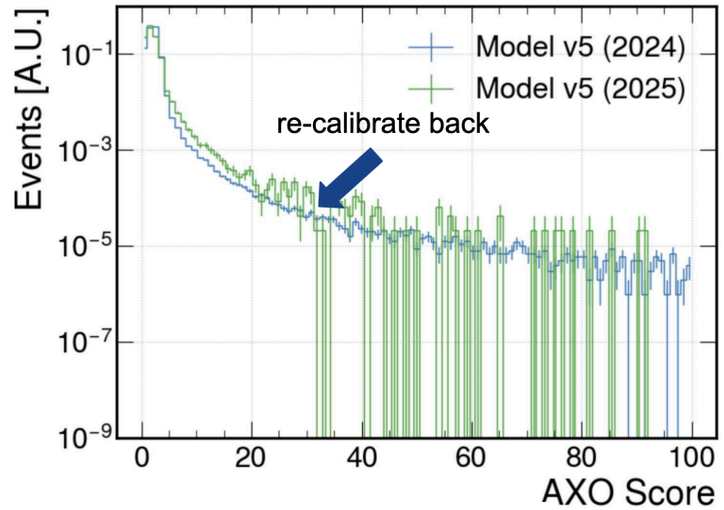
  - each retraining changes the algorithm's characteristics, effectively selecting a new phase space

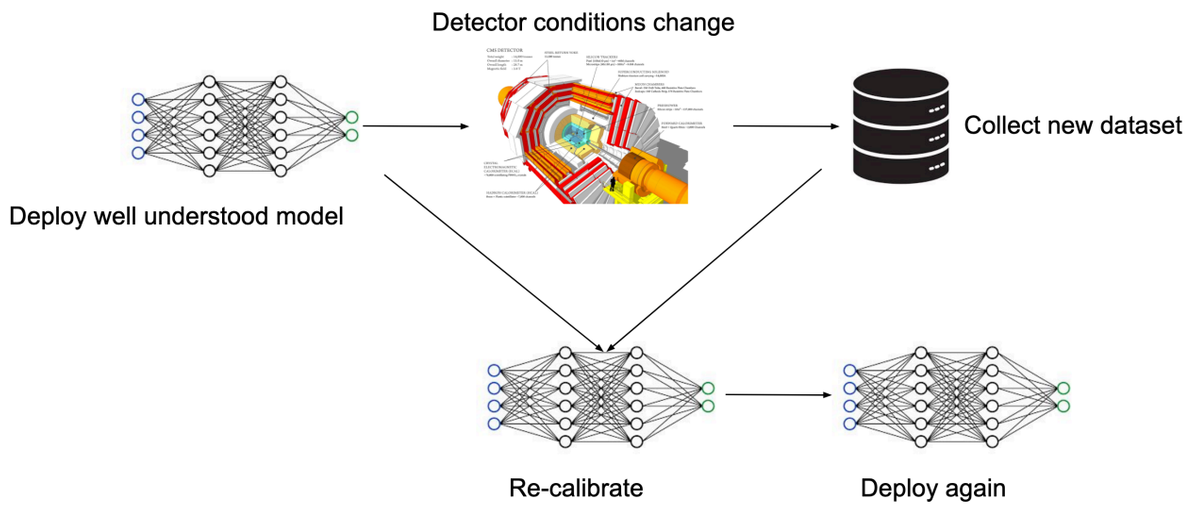
  - the new behaviour must then be tested, validated, presented: very lengthy process.

- **RE-CALIBRATING: Collect new dataset and re-calibrate the model to a reference distribution**

  - the reference is already well understood in terms of trigger characteristics

  - this can be done quickly to respond to sudden drifts in the data





Adaptability: Re-training vs re-calibrating

- **RE-TRAINING: Collect a new dataset and retrain the model** (CMS initial approach with AXOL1TL)

  - each retraining changes the algorithm's characteristics, effectively selecting a new phase space

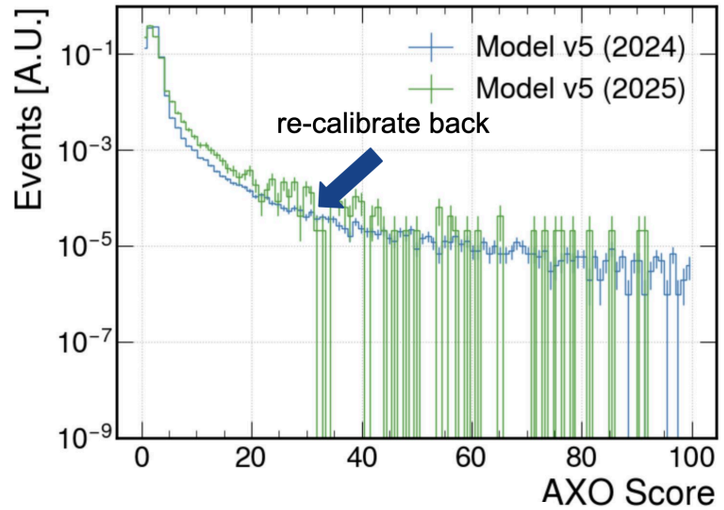
  - the new behaviour must then be tested, validated, presented: very lengthy process

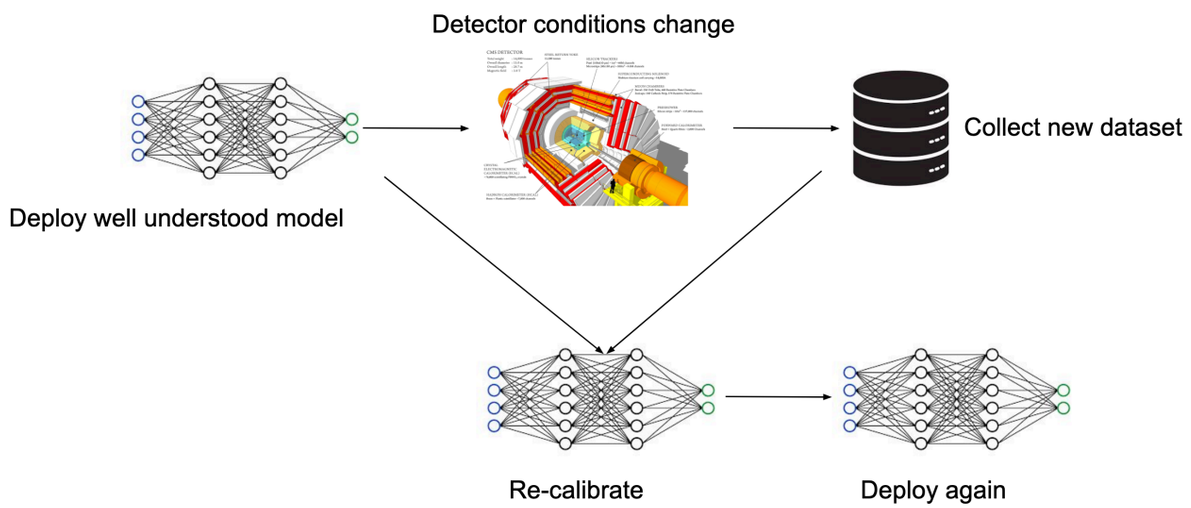
Topical session at FastML2025 by M. Glowacki
Talk at ACAT25 by M. Glowacki
ATLAS+CMS workshop on MLOps

- **RE-CALIBRATING: Collect new dataset and re-calibrate the model to a reference distribution**

  - the reference is already well understood in terms of trigger characteristics

  - this can be done quickly to respond to sudden drifts in the data

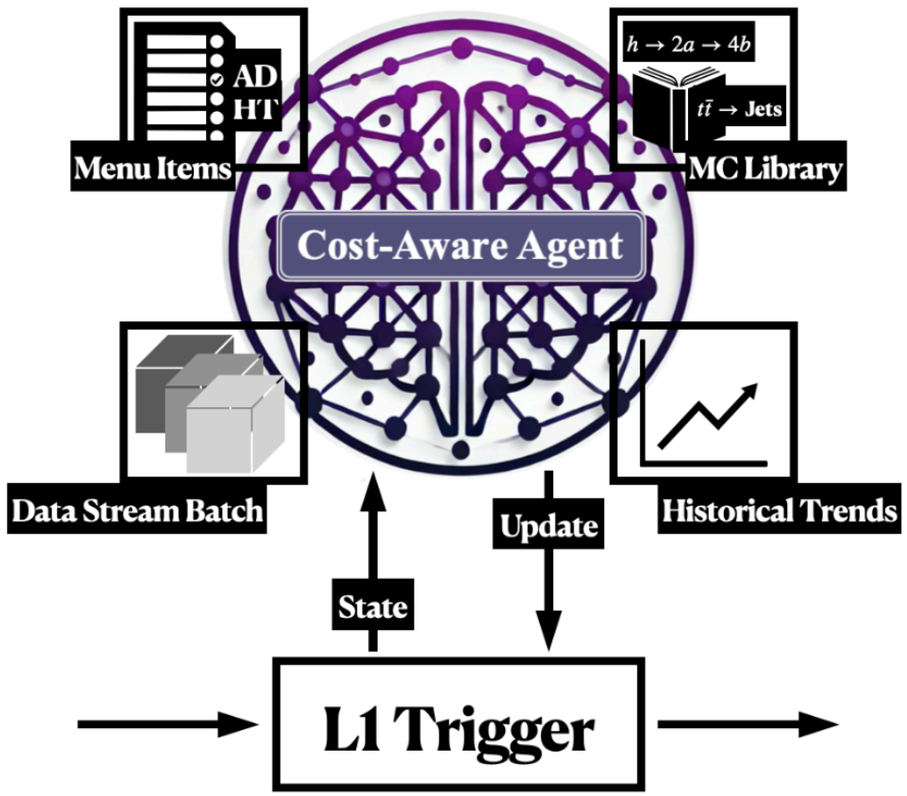




Studies ongoing on methods inspired by reinforcement learning strategies

Towards a self-driving trigger

- **Early-on studies** on a self-adaptable trigger system:
  - a feedback-control strategy based on a “simple” **PD controller** that takes into account background rate, signal efficiency, downstream bandwidth, and computational cost over a batch of data → **quasi real time**



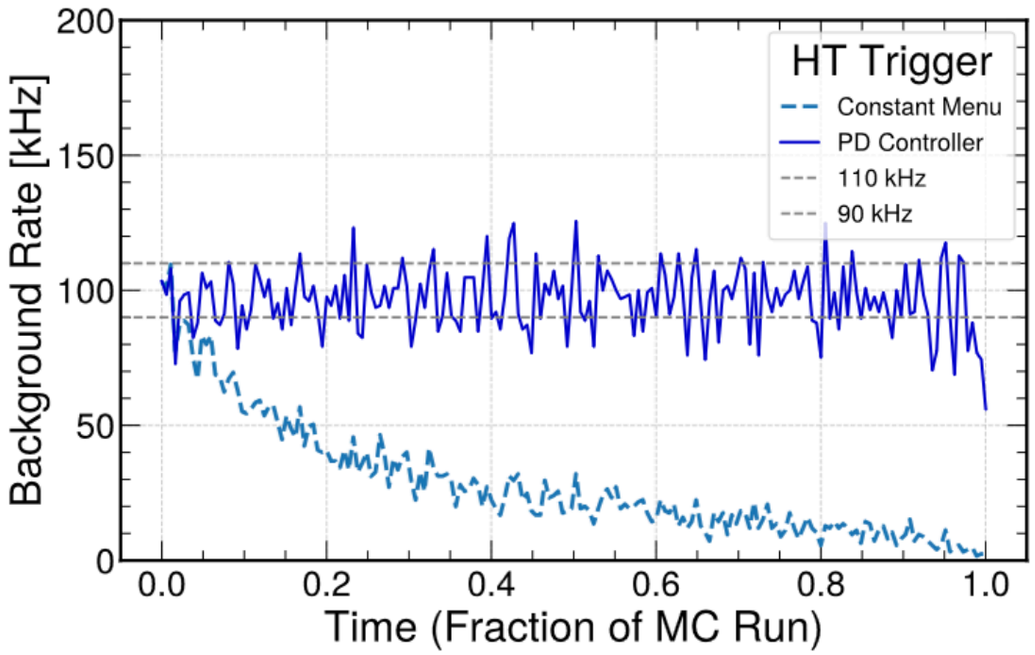
arXiv.2601.08910

FIG. 1: Schematic overview of the adaptive trigger control system. A cost-aware agent receives batches of collision data, evaluates trigger performance and computational costs, and updates L1 trigger thresholds and bandwidth allocations in real time through a feedback-driven control loop.

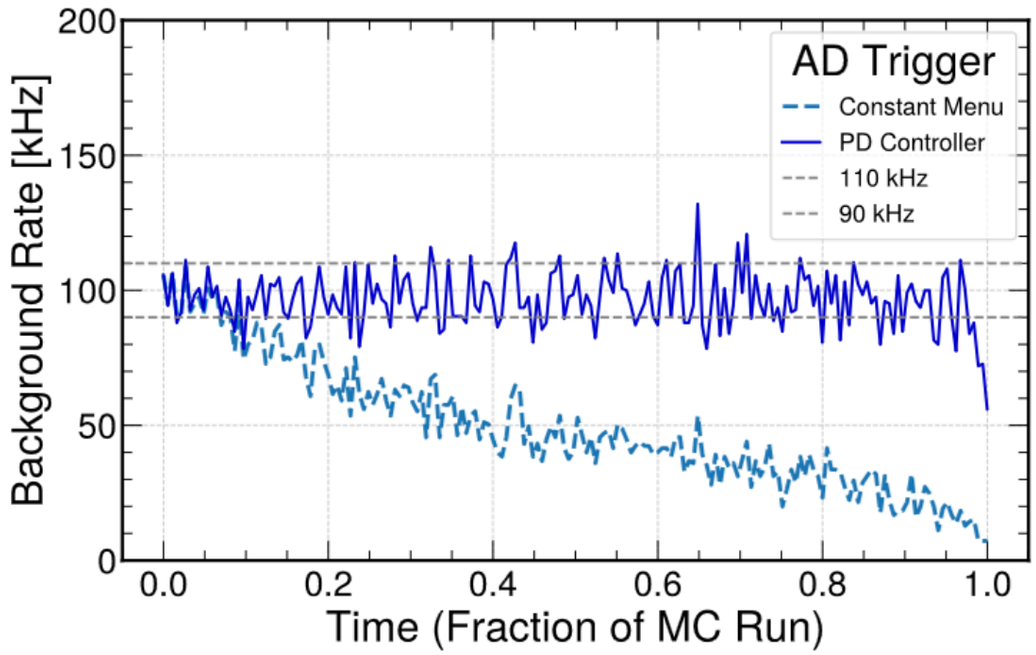
Towards a self-driving trigger

arXiv.2601.08910

- **Early-on studies** on a self-adaptable trigger system:
  - a feedback-control strategy based on a “simple” **PD controller** that takes into account background rate, signal efficiency, downstream bandwidth, and computational cost over a batch of data → **quasi real time**
- **Key message:** condition-aware AI should evolve from fixed ML scores to adaptive, hardware-aware control policies that learn how to operate the trigger in real time



(a) H_{T} trigger.

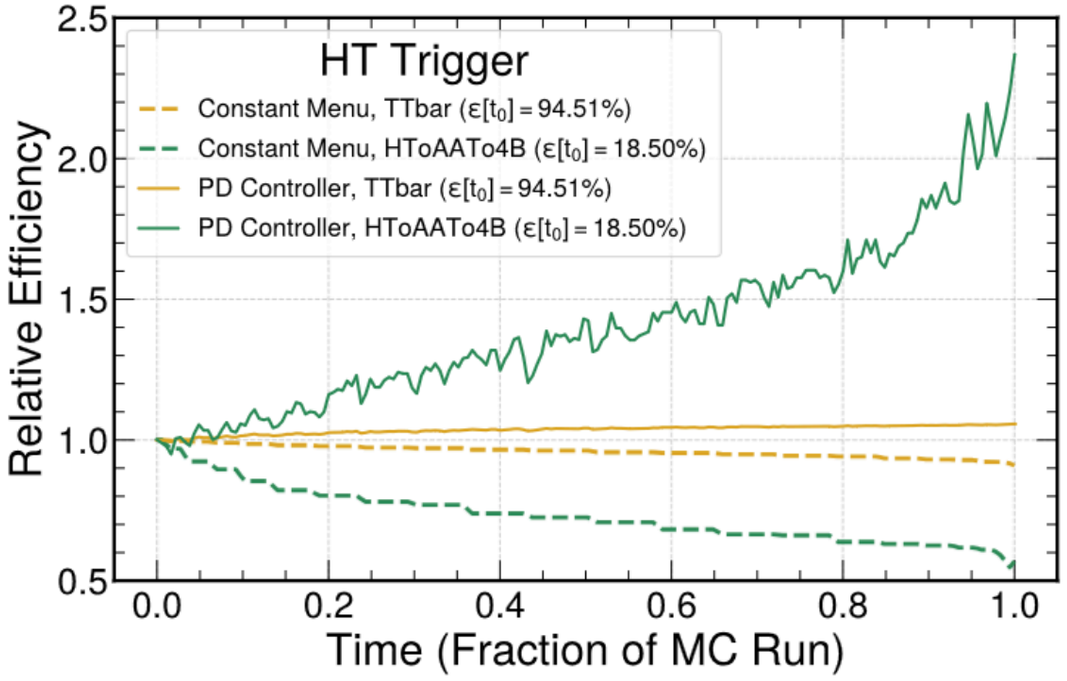


(b) AD trigger.

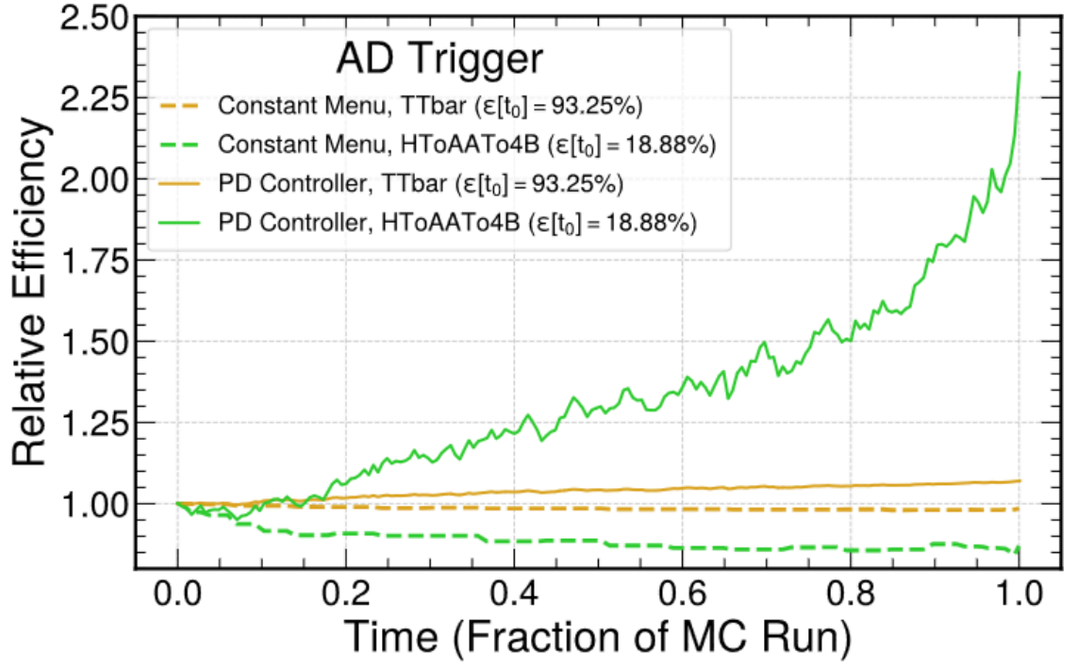
Towards a self-driving trigger

arXiv.2601.08910

- **Early-on studies** on a self-adaptable trigger system:
  - a feedback-control strategy based on a "simple" **PD controller** that takes into account background rate, signal efficiency, downstream bandwidth, and computational cost over a batch of data → **quasi real time**
- **Key message:** condition-aware AI should evolve from fixed ML scores to adaptive, hardware-aware control policies that learn how to operate the trigger in real time



(a) H_{T} trigger.



(b) AD trigger.

Towards a self-driving trigger

arXiv.2601.08910

- **Early-on studies** on a self-adaptable trigger system:

  - a feedback-control strategy based on a “simple” **PD controller** that takes into account background rate, signal efficiency, downstream bandwidth, and computational cost over a batch of data → **quasi real time**

- **Key message:** condition-aware AI should evolve from fixed ML scores to adaptive, hardware-aware control policies that learn how to operate the trigger in real time

- **Toward reinforcement learning:** reinterpret the trigger menu as an RL environment where the agent observes recent rates, conditions, correlations, and costs

  - actions could be threshold/bandwidth/corrections updates to reward rate stability, physics efficiency, and resource constraints

  - could be even faster and more performant if using fast ML tools

Fast Machine Learning for Science

- These unique needs brought together a community of scientists from different domains, computer and data scientists, electronic engineers

- Scientists created the **Fast Machine Learning Lab** holding interdisciplinary yearly workshops

  - last year edition at Zurich ETH

  - this year edition at UCSD

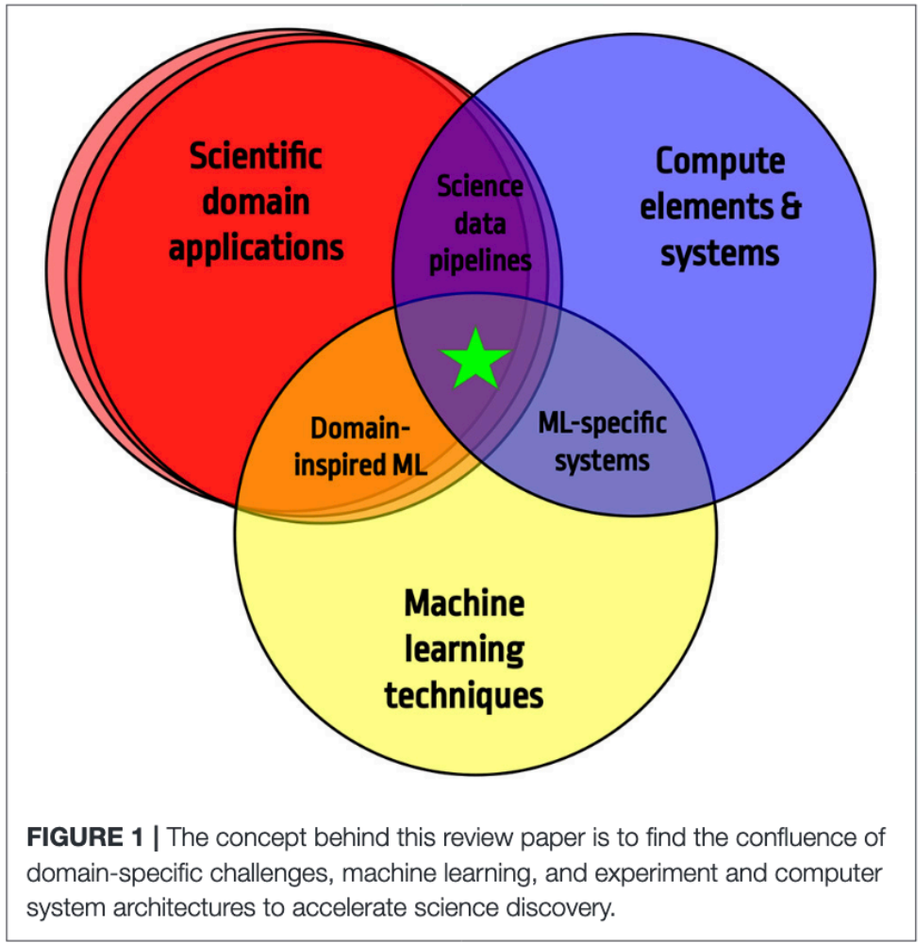
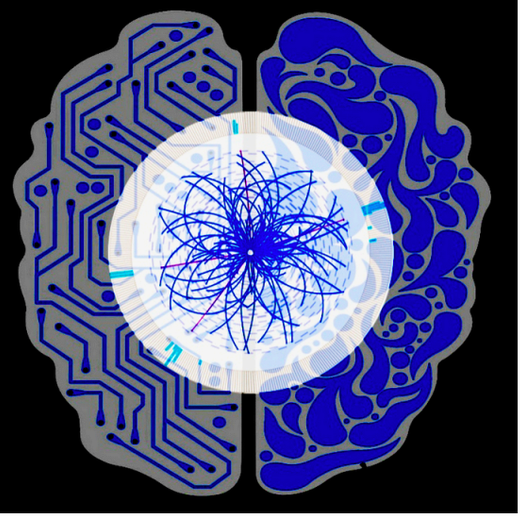


FIGURE 1 | The concept behind this review paper is to find the confluence of domain-specific challenges, machine learning, and experiment and computer system architectures to accelerate science discovery.

FastML Lab

Real-time and accelerated ML for fundamental sciences

Fast ML Lab is a research collective of physicists, engineers, and computer scientists interested in deploying machine learning algorithms for unique and challenging scientific applications. Our projects range from real-time, on-detector and low latency machine learning applications to high-throughput heterogeneous computing big data challenges. We are interested in deploying sophisticated machine learning algorithms to advance the exploration of fundamental physics from the world's biggest colliders to the most intense particle beams to the cosmos.



frontiers | Frontiers in Big Data

REVIEW
published: 12 April 2022
doi: 10.3389/fdata.2022.787421

Applications and Techniques for Fast Machine Learning in Science

Allison McCarn Deiana[1], Nhan Tran[2,3]*, Joshua Agar[4], Michaela Blott[5], Giuseppe Di Guglielmo[6], Javier Duarte[7], Philip Harris[8], Scott Hauck[9], Mia Liu[10], Mark S. Neubauer[11], Jennifer Ngadiuba[2], Seda Ogrenci-Memik[3], Maurizio Pierini[12], Thea Aarrestad[12], Steffen Bähr[13], Jürgen Becker[13], Anne-Sophie Berthold[14], Richard J. Bonventre[15], Tomás E. Müller Bravo[16], Markus Diefenthaler[17], Zhen Dong[18], Nick Fritzsche[19], Amir Gholami[18], Ekaterina Govorkova[12], Dongning Guo[3], Kyle J. Hazelwood[2], Christian Herwig[2], Babar Khan[20], Sehoon Kim[18], Thomas Klijnsma[2], Yaling Liu[21], Kin Ho Lo[22], Tri Nguyen[8], Gianantonio Pezzullo[23], Seyedramin Rasoulinezhad[24], Ryan A. Rivera[2], Kate Scholberg[25], Justin Selig[14], Sougata Sen[26], Dmitri Strukov[27], William Tang[28], Savannah Thais[28], Kai Lukas Unger[13], Ricardo Vilalta[29], Belina von Krosigk[13,30], Shen Wang[21] and Thomas K. Warburton[31]*

OPEN ACCESS

Edited by:
Elena Cuoco,
European Gravitational Observatory, Italy

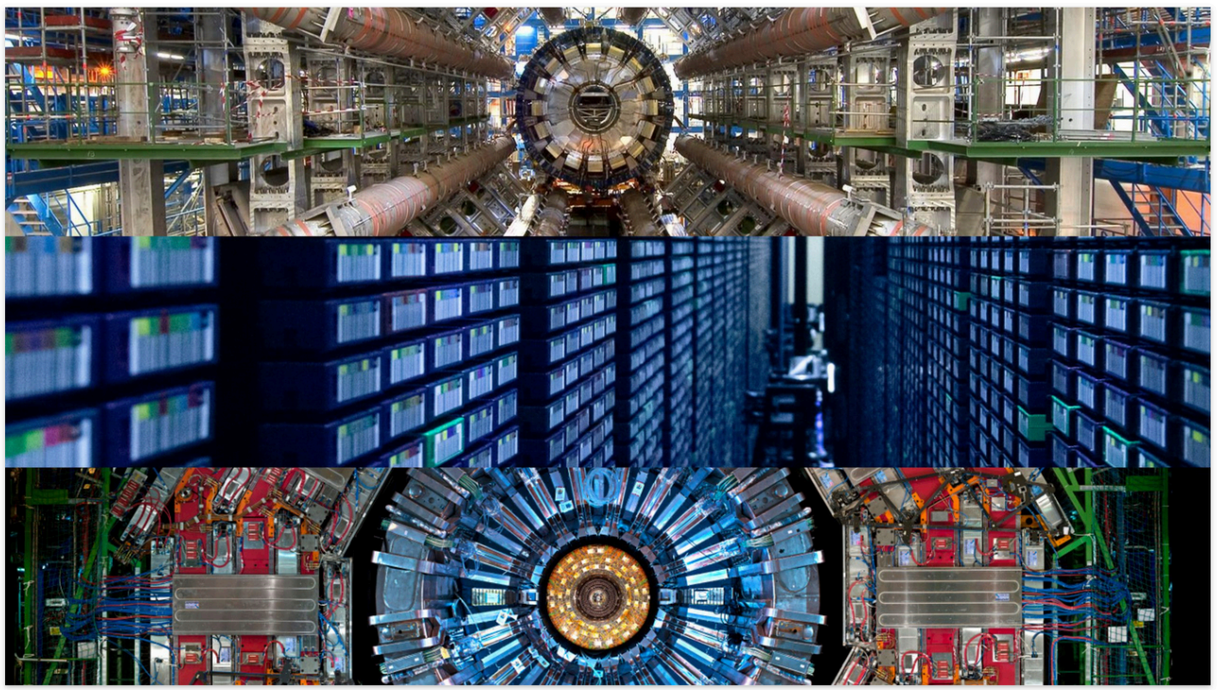
The Next Generation Triggers

[CERN news](#)

The next-generation triggers for CERN detectors

The recently launched Next-Generation Triggers project is set to remarkably increase the efficiency, sensitivity and modelling of CERN experiments

11 APRIL, 2024 | By Antonella Del Rosso



From top to bottom: ATLAS, CERN Data Centre and CMS (Image: CERN)

- Eric & Wendy Schmidt foundation fund a CERN project that will *enhance the physics reach of the ATLAS and CMS experiments at HL-LHC and beyond* using novel technologies:
  - neural network optimization
  - quantum-inspired algorithms
  - high-performance computing and FPGAs
  - theoretical modelling

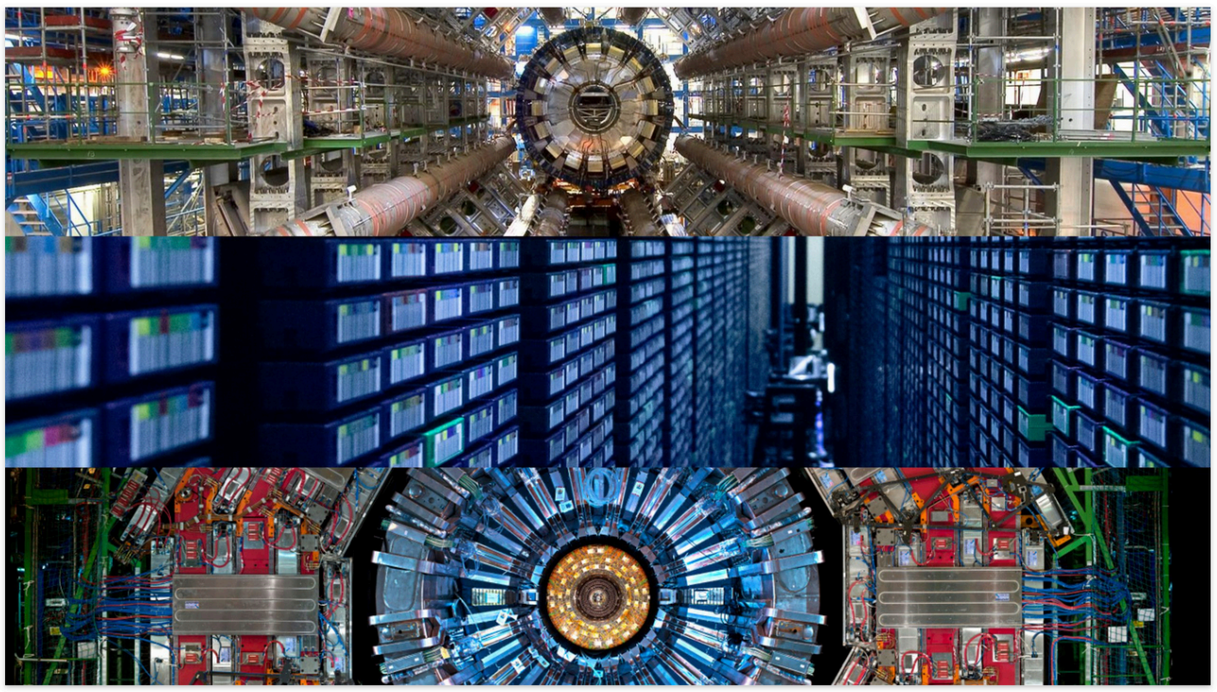
The Next Generation Triggers

[CERN news]

The next-generation triggers for CERN detectors

The recently launched Next-Generation Triggers project is set to remarkably increase the efficiency, sensitivity and modelling of CERN experiments

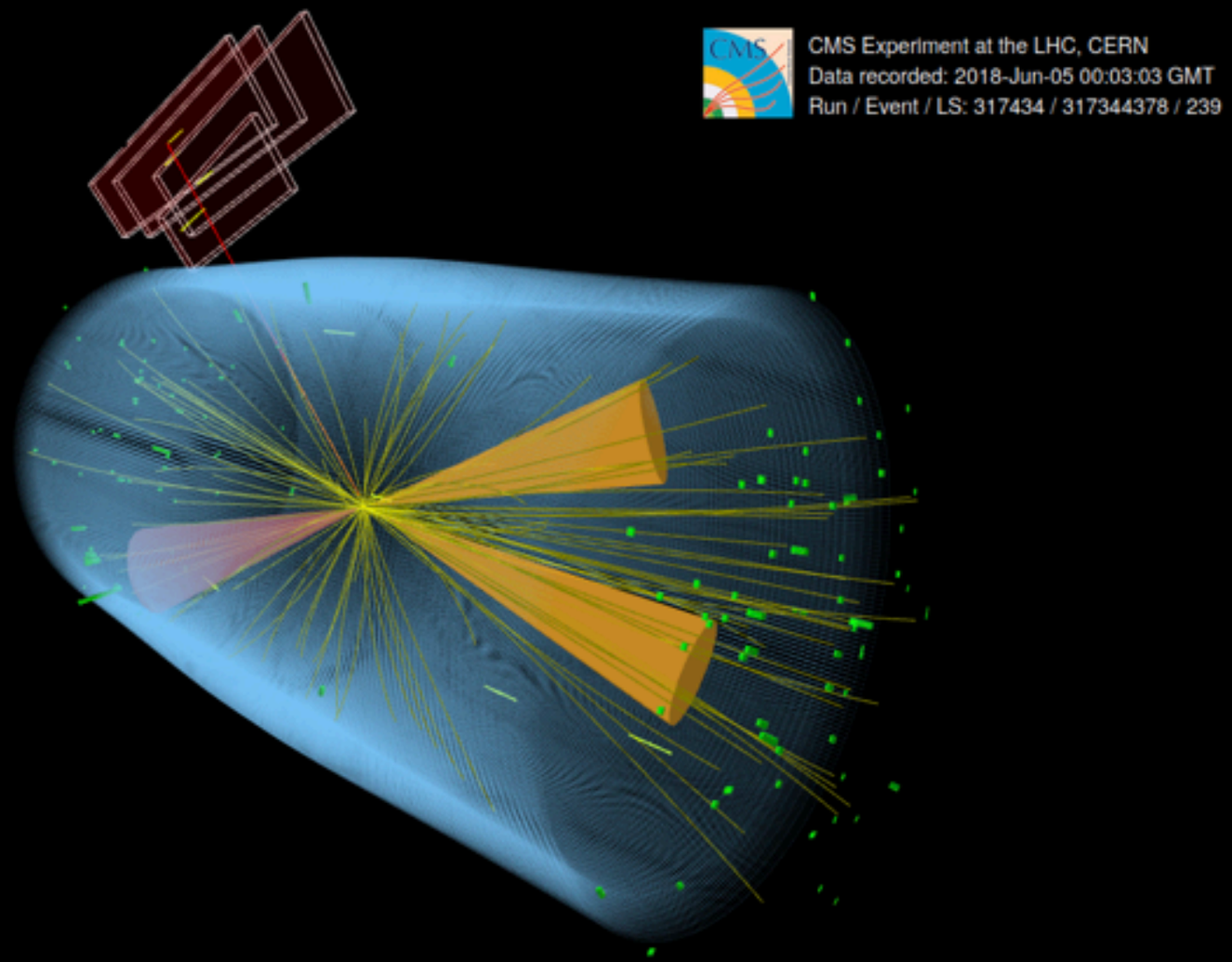
11 APRIL, 2024 | By Antonella Del Rosso



From top to bottom: ATLAS, CERN Data Centre and CMS (Image: CERN)

- Eric & Wendy Schmidt foundation fund a CERN project that will *enhance the physics reach of the ATLAS and CMS experiments at HL-LHC and beyond* using novel technologies:
  - neural network optimization
  - quantum-inspired algorithms
  - high-performance computing and FPGAs
  - theoretical modelling

The NextGen Triggers project will mark a new chapter in high-energy physics, leveraging upgraded event-selection systems and data-processing techniques to unlock a realm of discoveries.


CMS Experiment at the LHC, CERN
Data recorded: 2018-Jun-05 00:03:03 GMT
Run / Event / LS: 317434 / 317344378 / 239

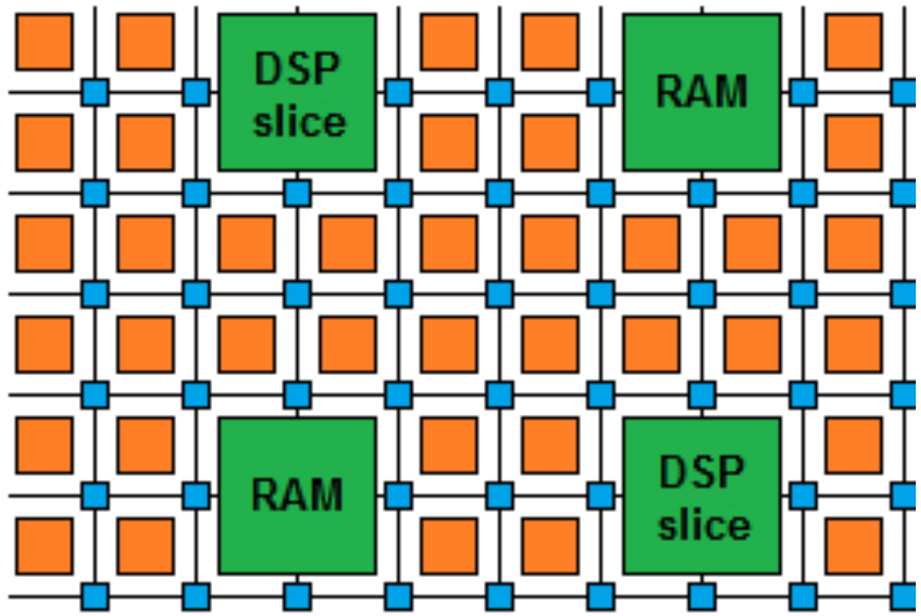
BACKUP

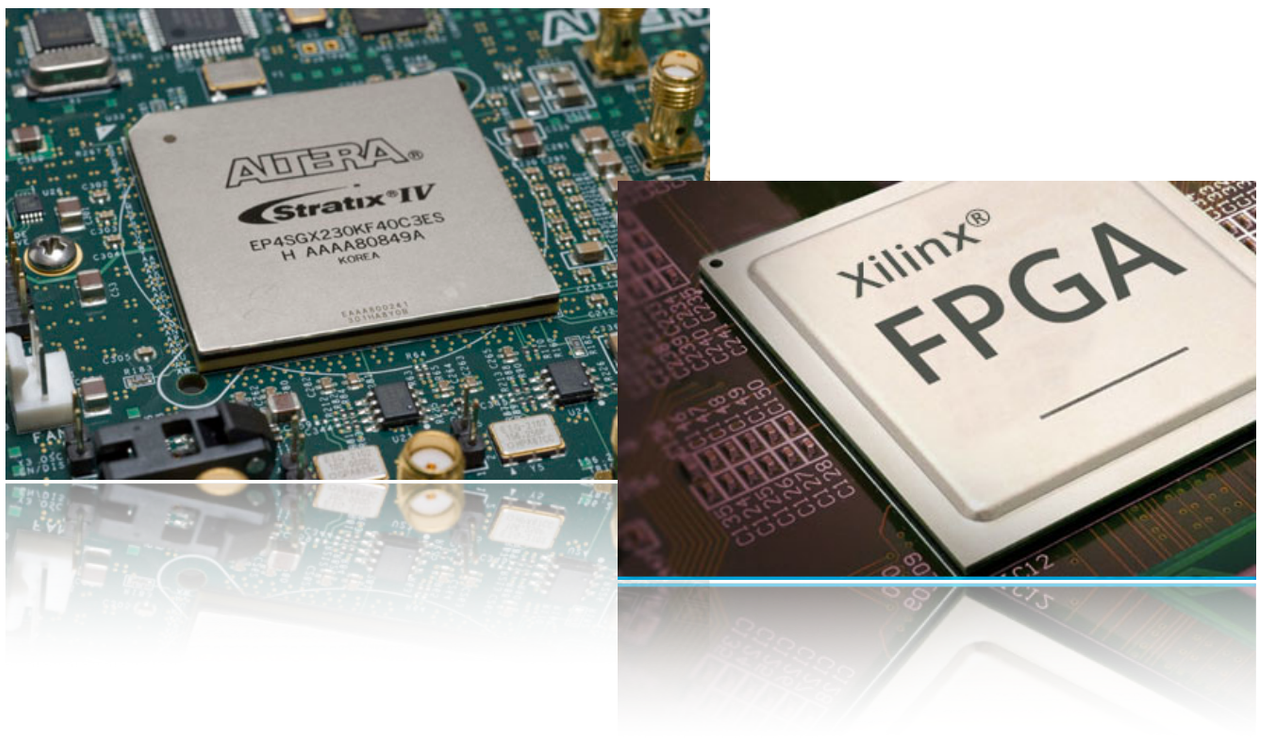
What are FPGAs?

Field Programmable Gate Arrays
are reprogrammable integrated circuits

Contain many different building blocks ('resources') which are connected together as you desire

FPGA diagram



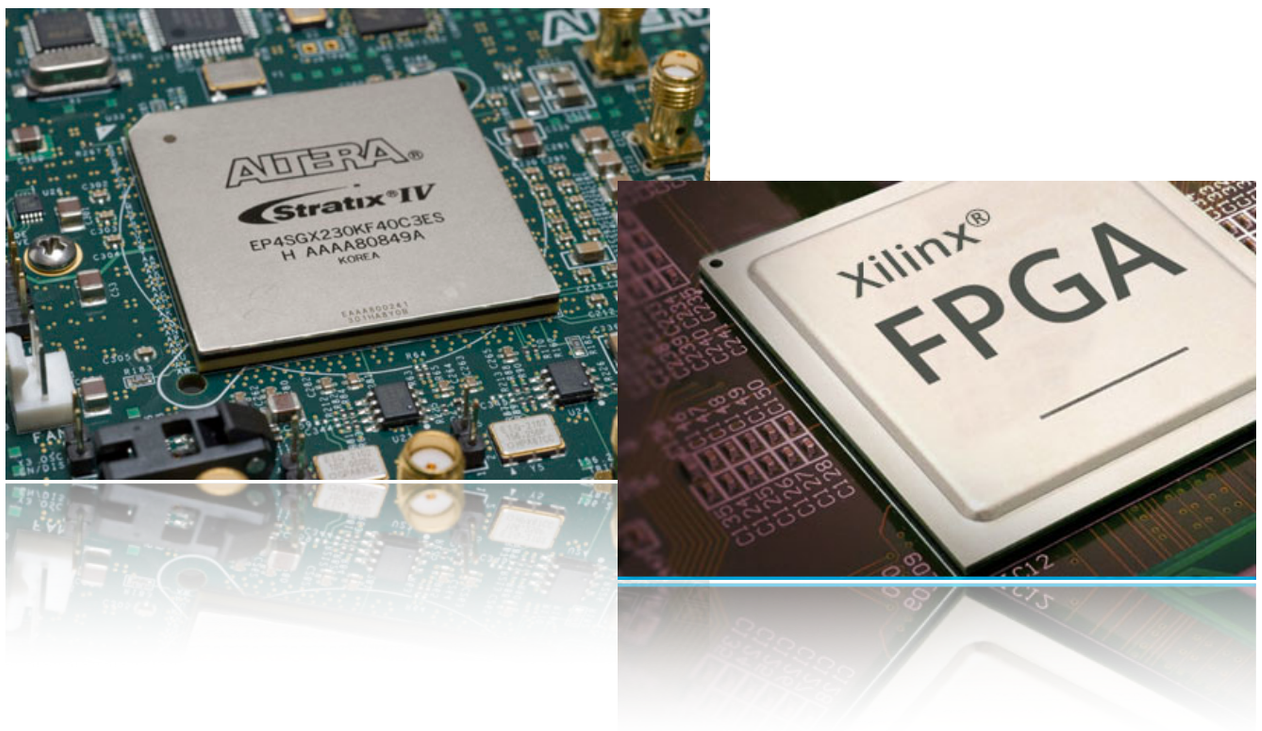


What are FPGAs?

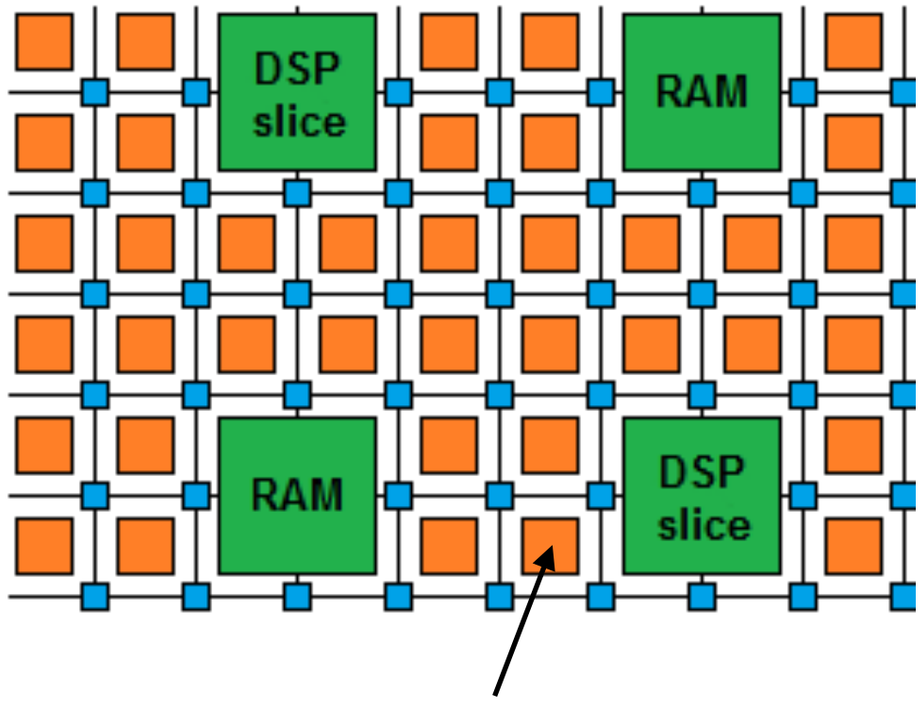
Field Programmable Gate Arrays
are reprogrammable integrated circuits

Look Up Tables (LUTs) perform arbitrary functions on small bitwidth inputs (2-6 bits)
→ used for boolean operations, arithmetics, memory

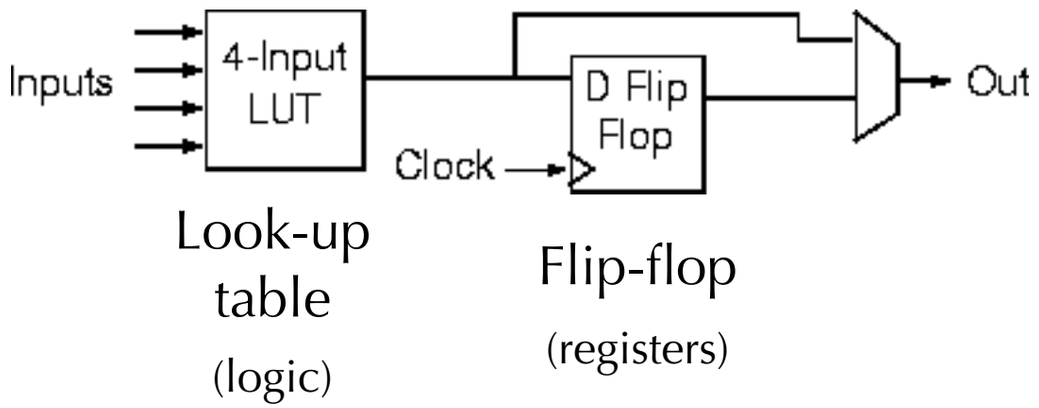
Flip-flops register data in time with the clock pulse



FPGA diagram



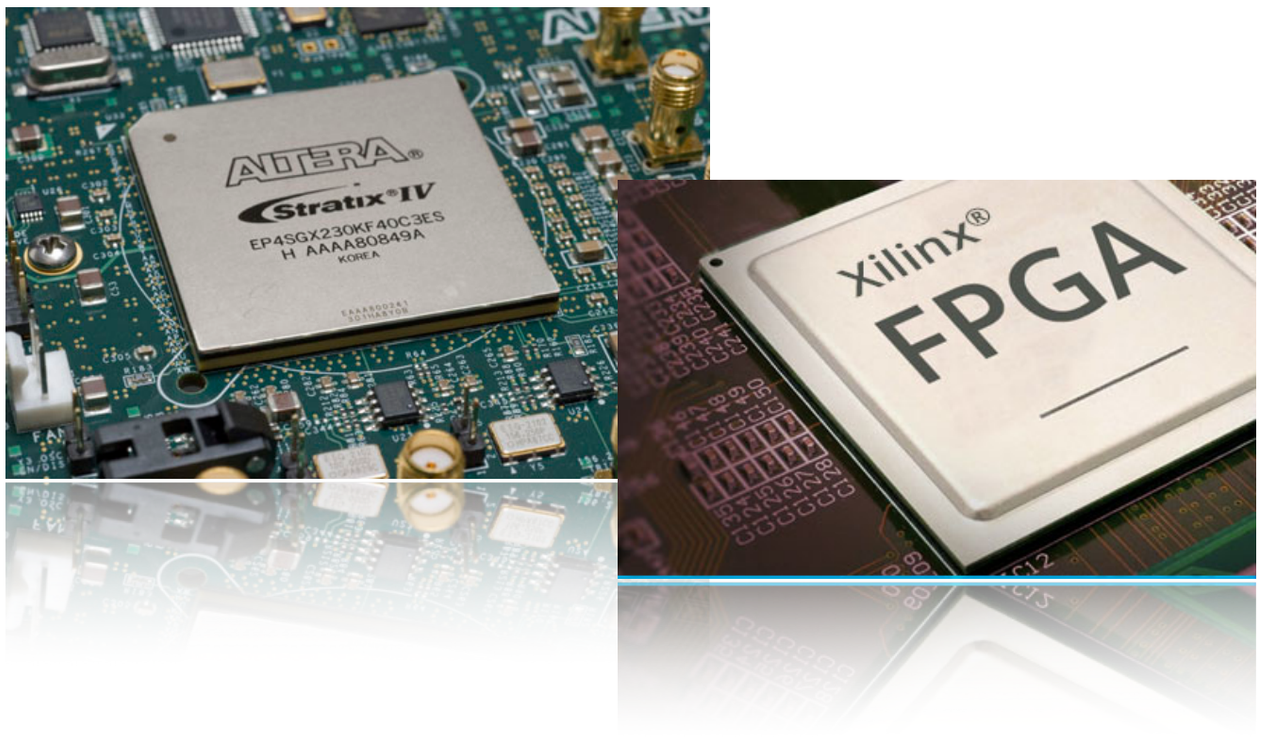
Logic cell



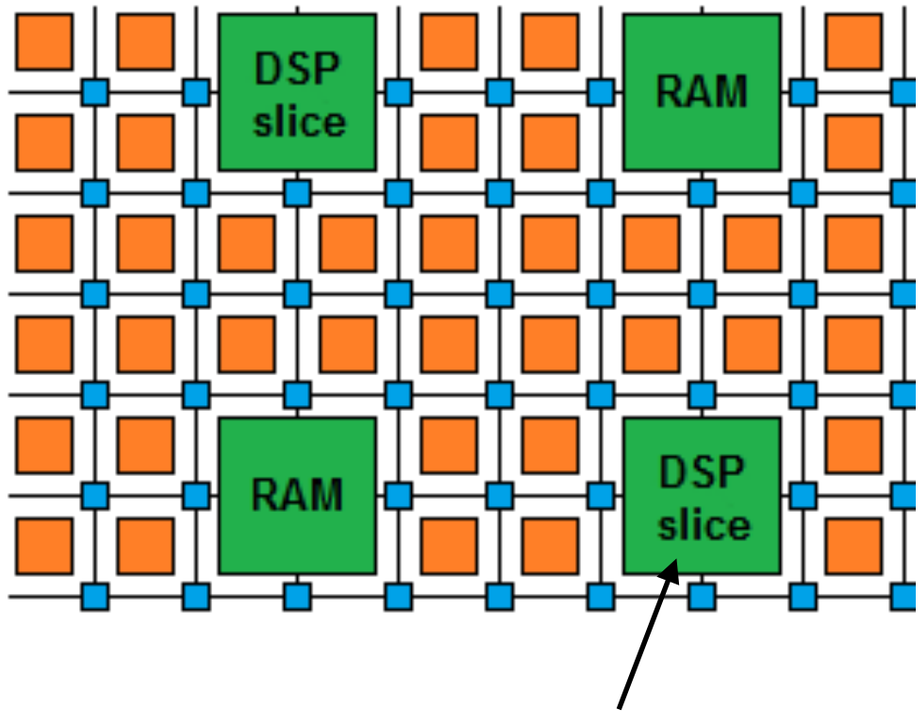
What are FPGAs?

Field Programmable Gate Arrays
are reprogrammable integrated circuits

DSPs are specialized units for multiplication and arithmetic
→ faster and more efficient than LUTs for these type of operations
→ for deep learning, they are often the most precious resource



FPGA diagram



Also contain embedded components:

Digital Signal Processors (DSPs): logic units used for multiplications

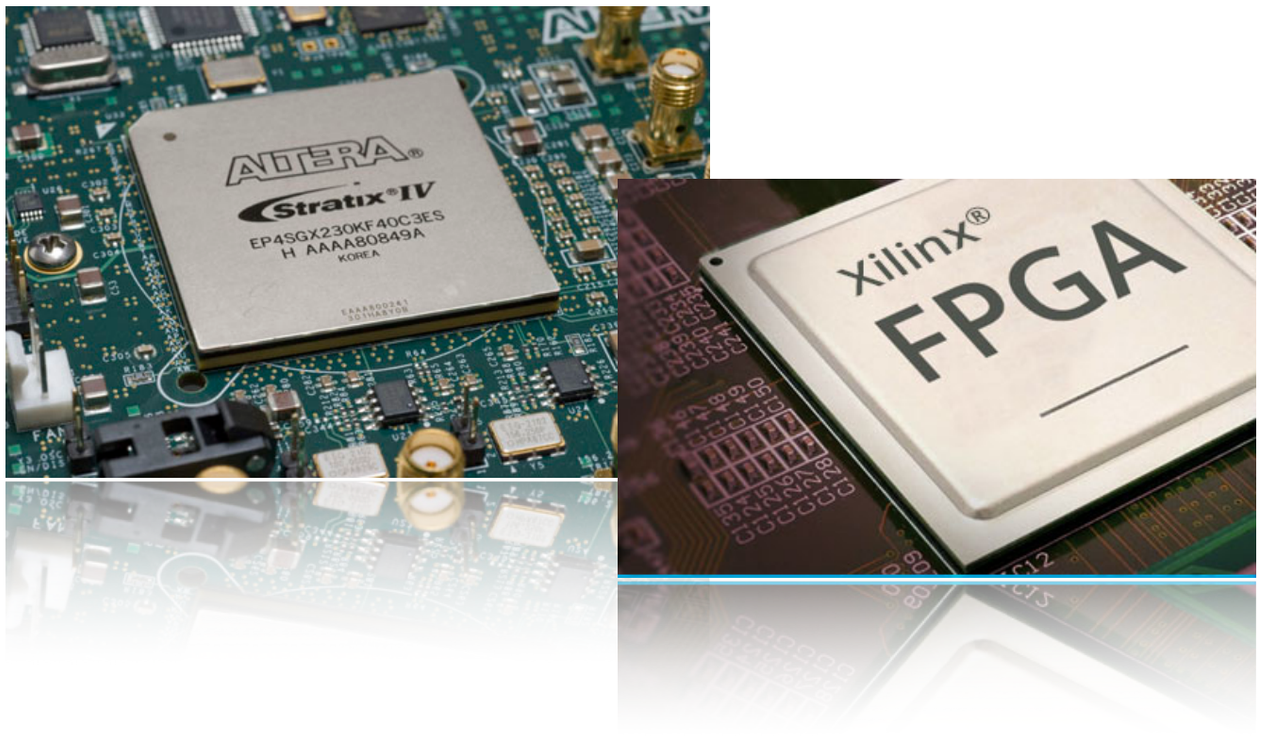
What are FPGAs?

Field Programmable Gate Arrays
are reprogrammable integrated circuits

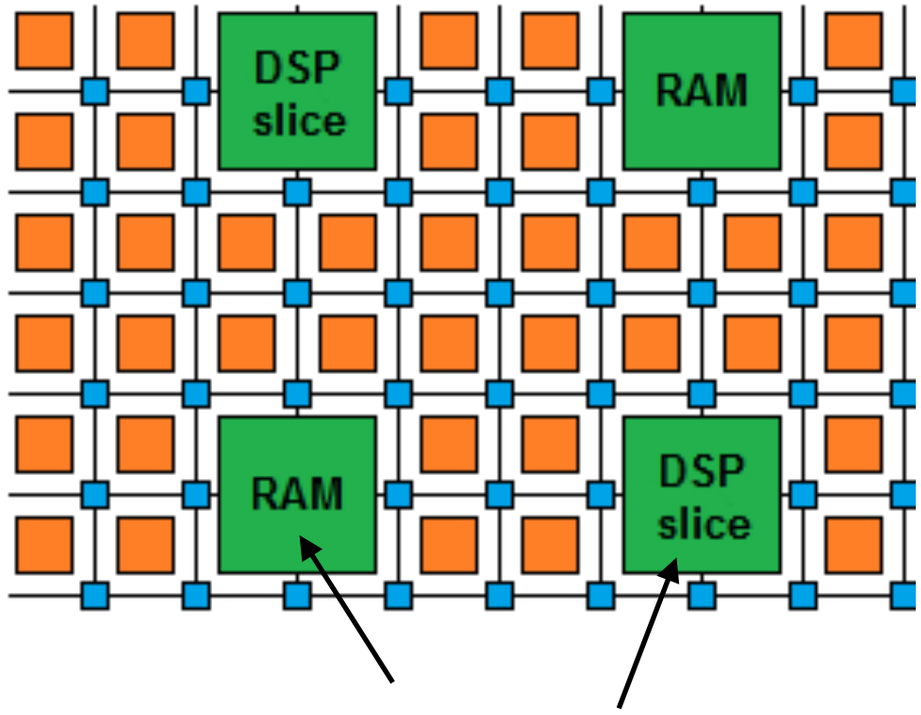
BRAMs are small, fast memories (ex, 18 Kb each)

→ more efficient than LUTs when large memory is required

Modern FPGAs have ~100 Mb of BRAMs, chained together as needed



FPGA diagram



Also contain embedded components:

Digital Signal Processors (DSPs): logic units used for multiplications

Random-access memories (RAMs): embedded memory elements

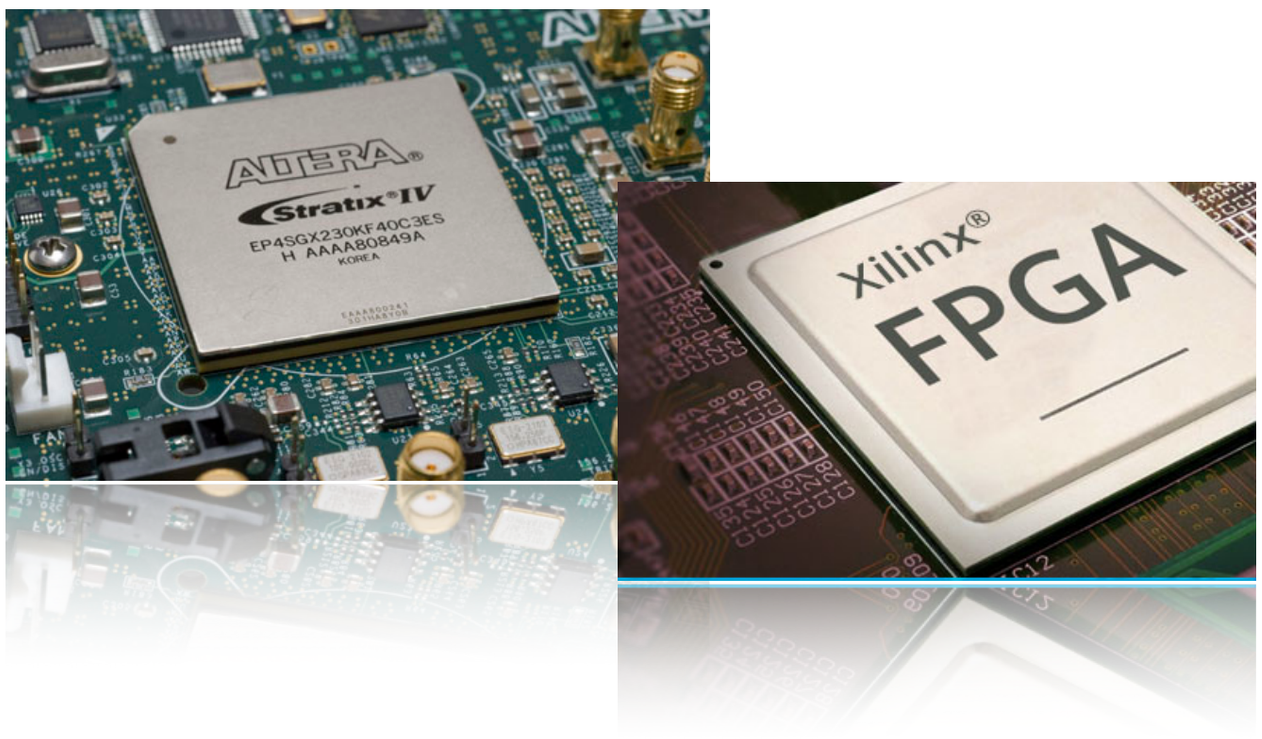
What are FPGAs?

Field Programmable Gate Arrays
are reprogrammable integrated circuits

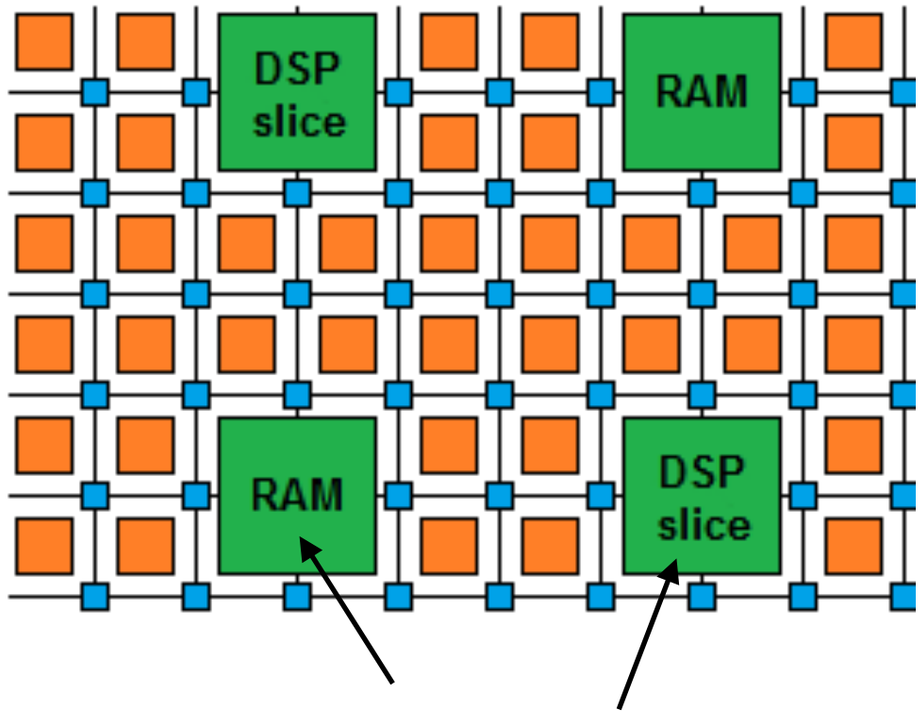
Contain array of **logic cells** embedded with **DSPs**, **BRAMs**, etc.

Support **highly parallel** algorithm implementation

Low power per Op (relative to CPU/GPU)



FPGA diagram



Also contain embedded components:

Digital Signal Processors (DSPs): logic units used for multiplications

Random-access memories (RAMs): embedded memory elements

Why are FPGAs fast?

- Fine-grained / resource parallelism
  - use the many resources to work on different parts of the problem simultaneously
  - allows us to achieve **low latency**
- Most problems have at least some sequential aspect, limiting how low latency we can go
  - but we can still take advantage of it with…
- **Pipeline parallelism**
  - instruct the FPGA to work on different data simultaneously
  - allows us to achieve **high throughput**

Like a production line for data…

How are FPGAs programmed?

Hardware Description Languages

HDLs are programming languages which describe electronic circuits

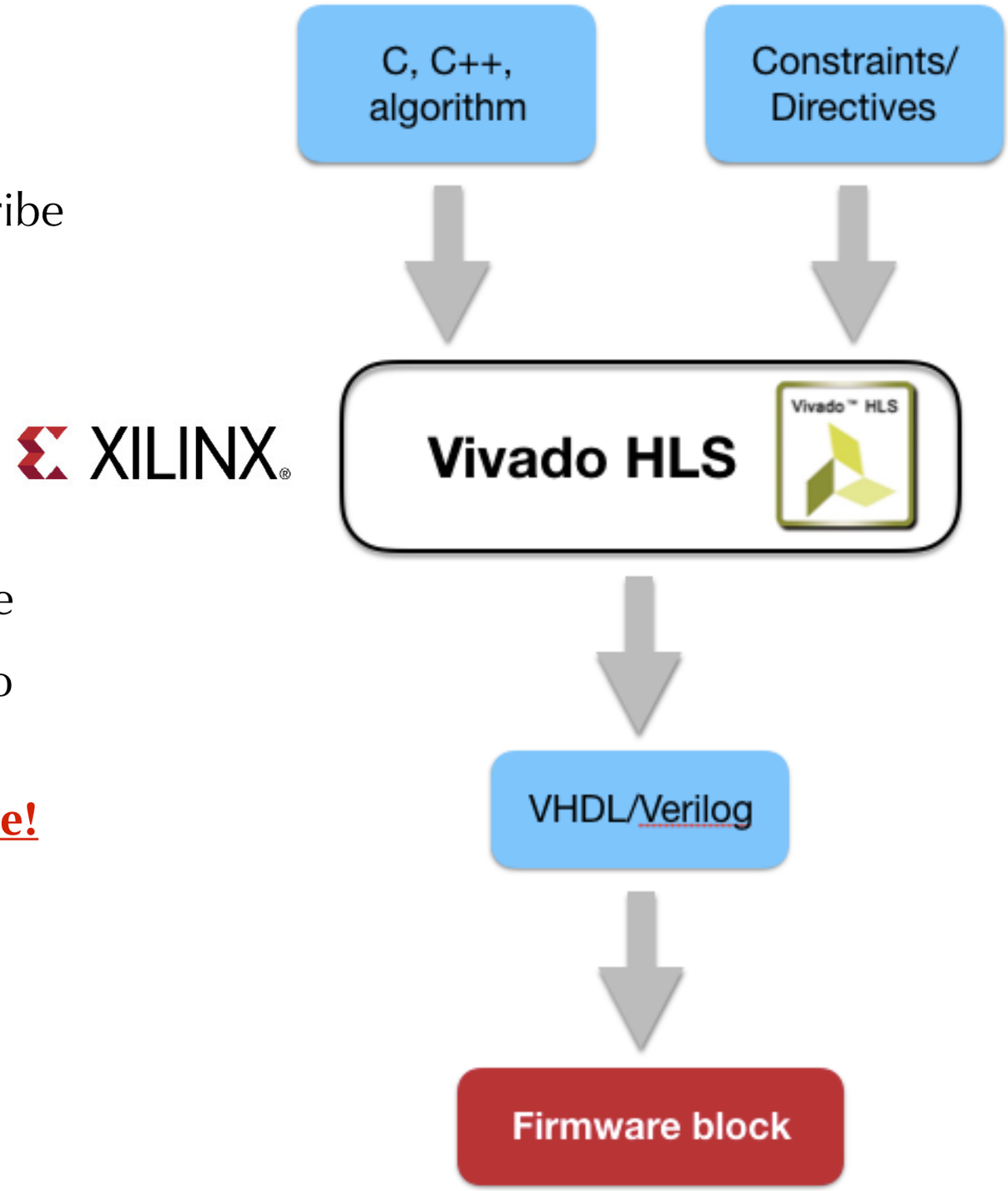
High Level Synthesis

generate HDL from more common C/C++ code

pre-processor directives and constraints used to optimize the timing

drastic decrease in firmware development time!

See Xilinx Vivado HLS, Intel HLS, Catapult HLS

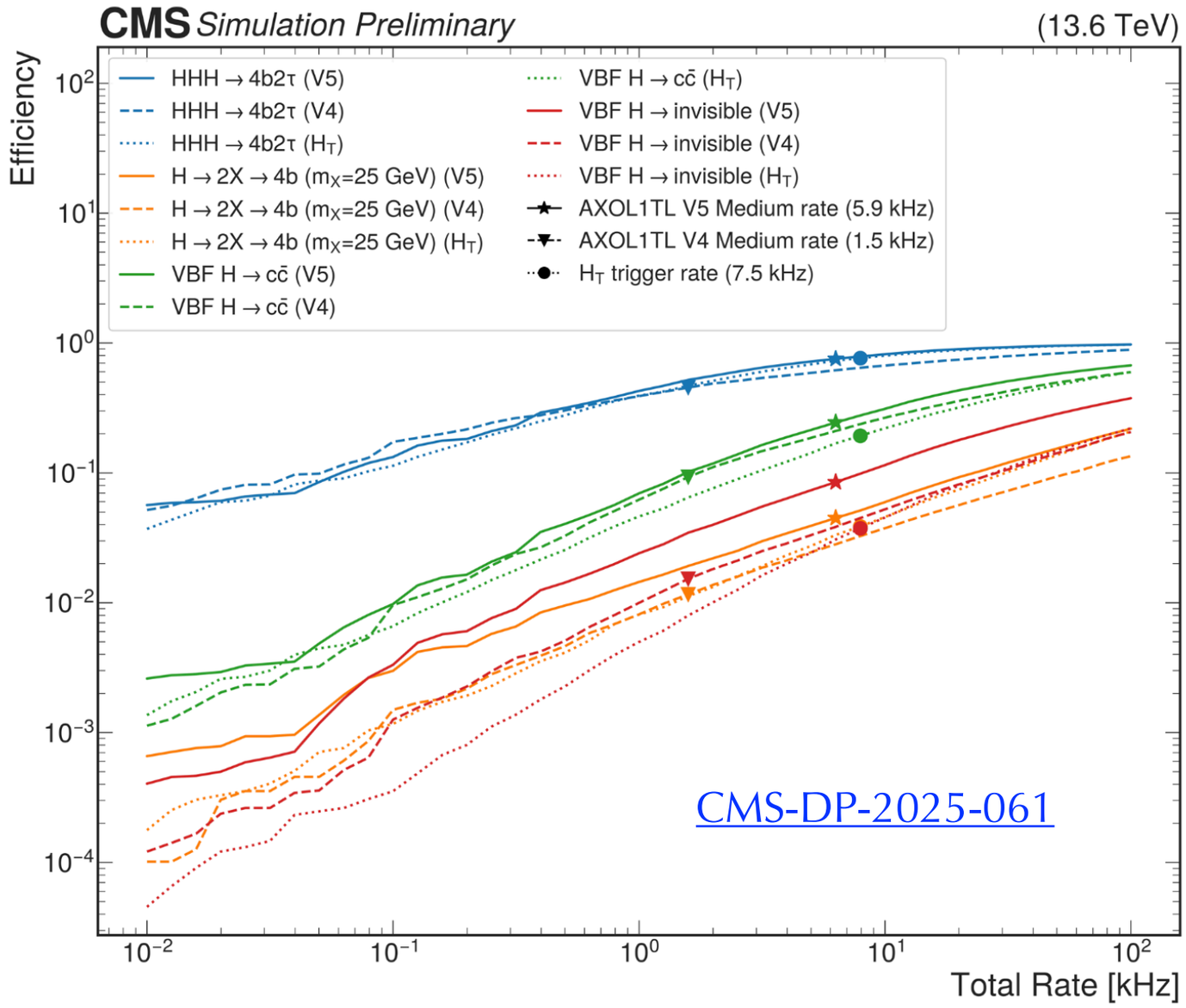


Physics performance

- Scan of AXOL1TL anomaly score thresholds, corresponding to a total range, for SM and BSM simulated samples and calculate the efficiency

- Broad sensitivity of the unsupervised AXOL1TL approach to a wide range of new physics scenarios.

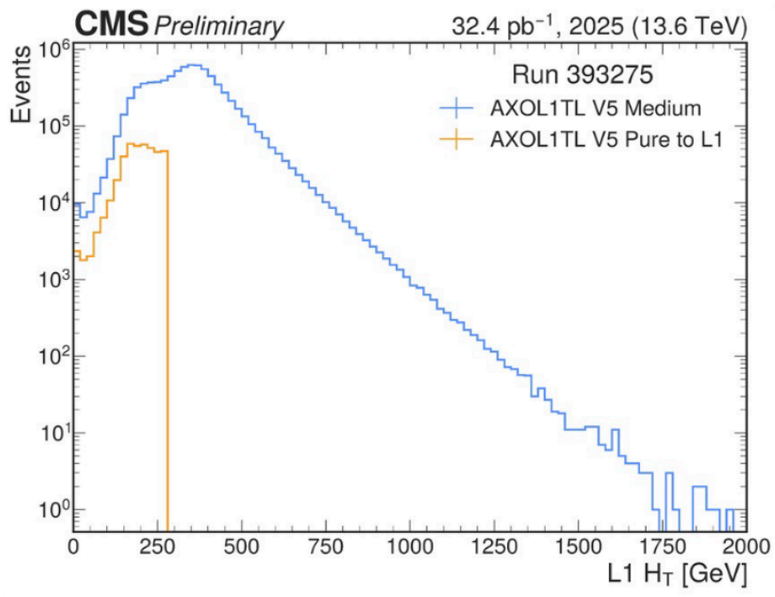
- V5 efficiency improved compared to V4 and competitive with a standard trigger like H_T (scalar sum of jets p_T)

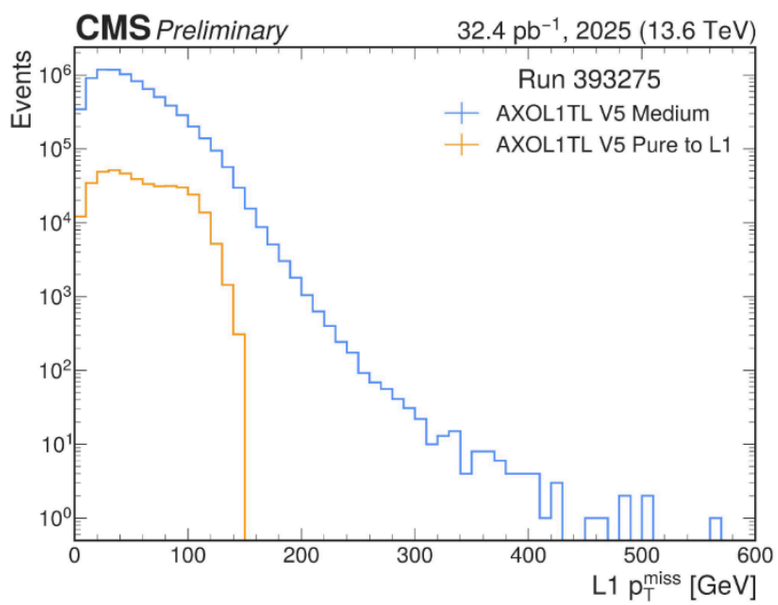


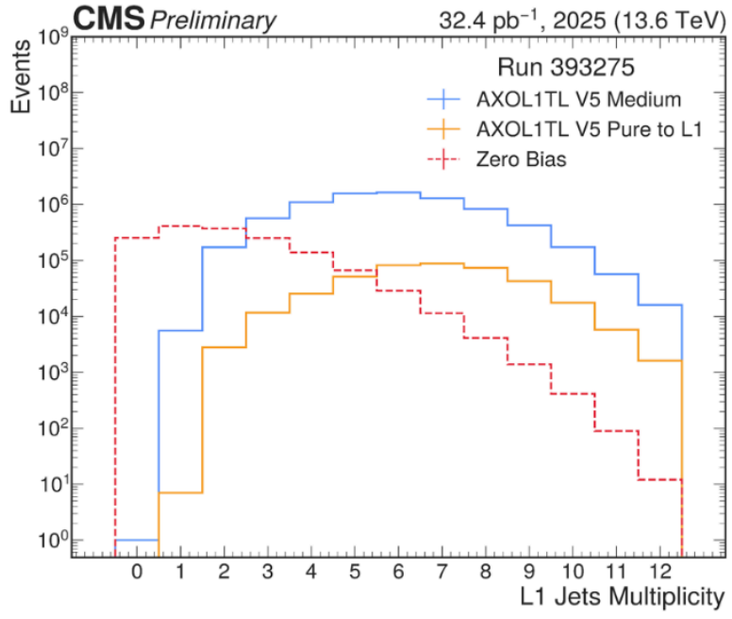
How anomalous events look like?

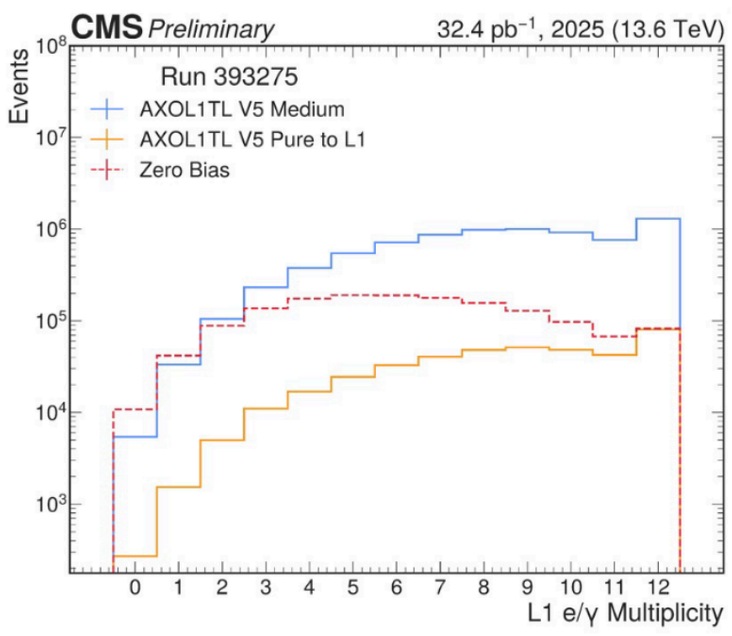
- Pure AXOL1TL events in the energy sums spectrum: enhanced sensitivity to lower L1 H_T and p_T^{miss} values

- AXOL1TL selects events with higher object multiplicity, particularly for jets and e/γ.



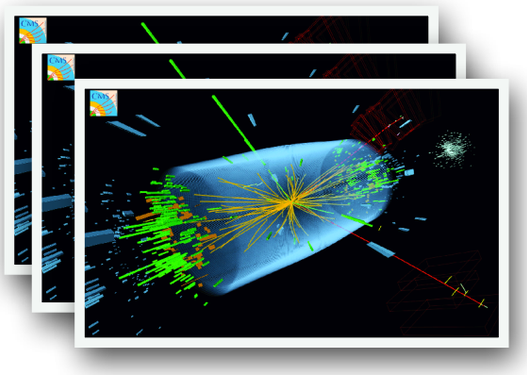






CMS-DP-2025-061

AI @ Extreme Edge

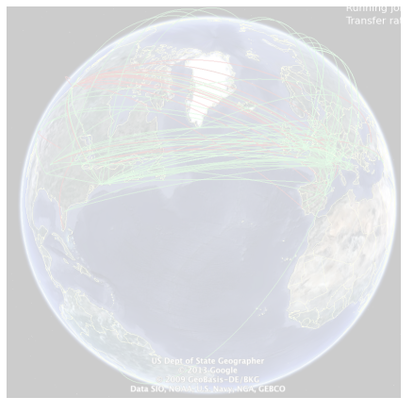


CMS Experiment
40 MHz collision rate
~1B detector channels

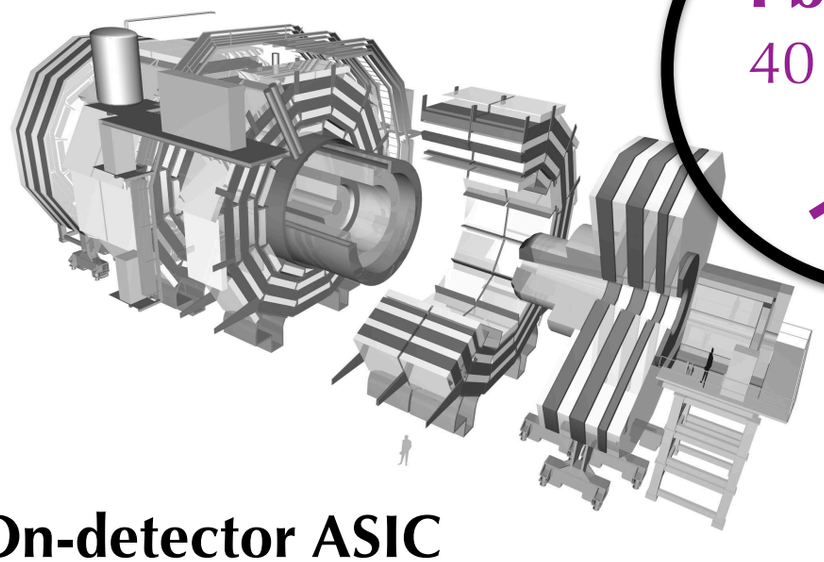
Worldwide computing grid
Exabyte-scale datasets

FPGA filter stack
~µs latency

Level-1 Trigger



Pb/s
40 MHz



ASICs typically used at the front end for sensors read out: directly embed ML in here to allow intelligent data compression at the very edge

On-detector ASIC compression
~100 ns latency

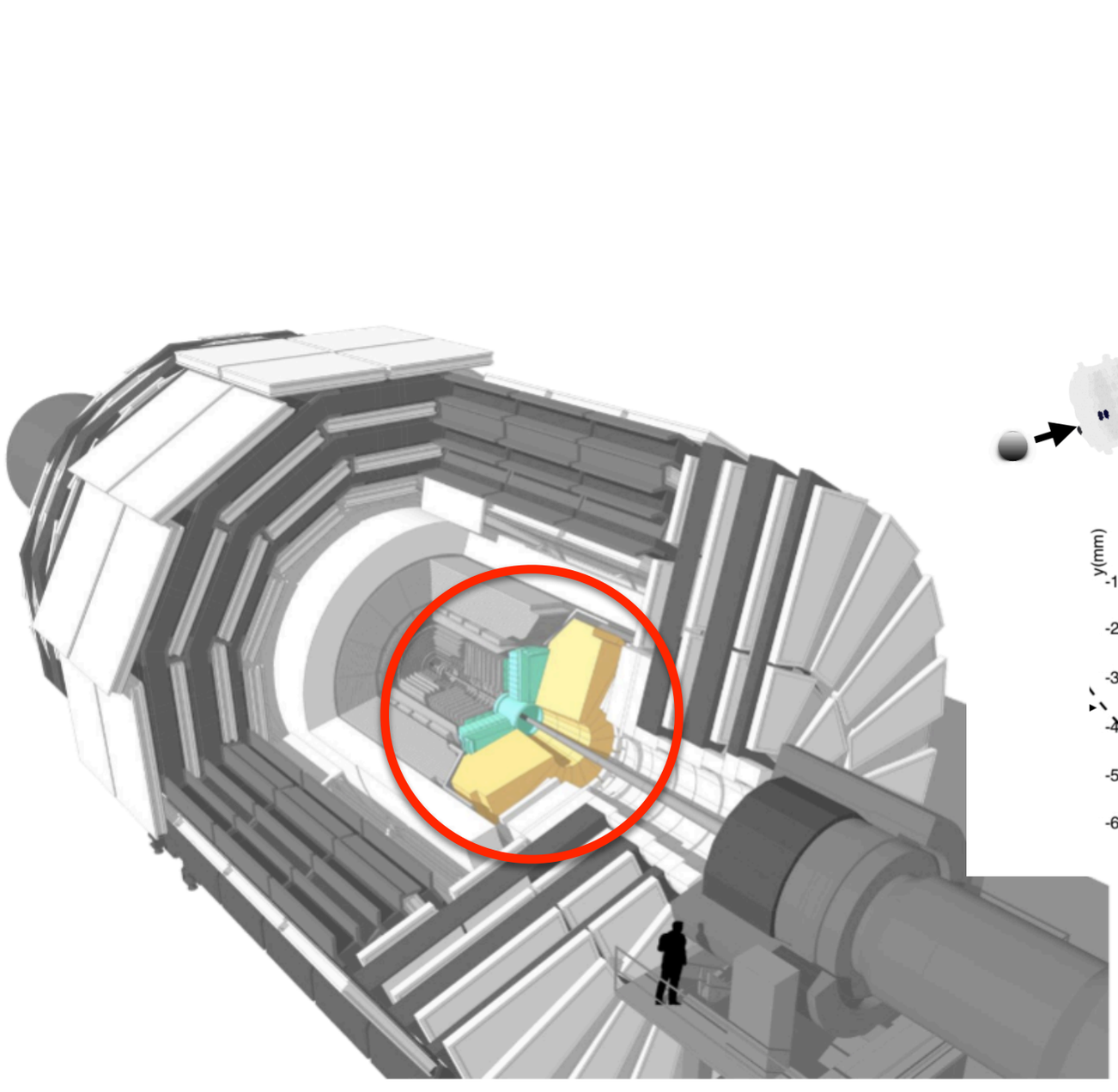
100s kHz

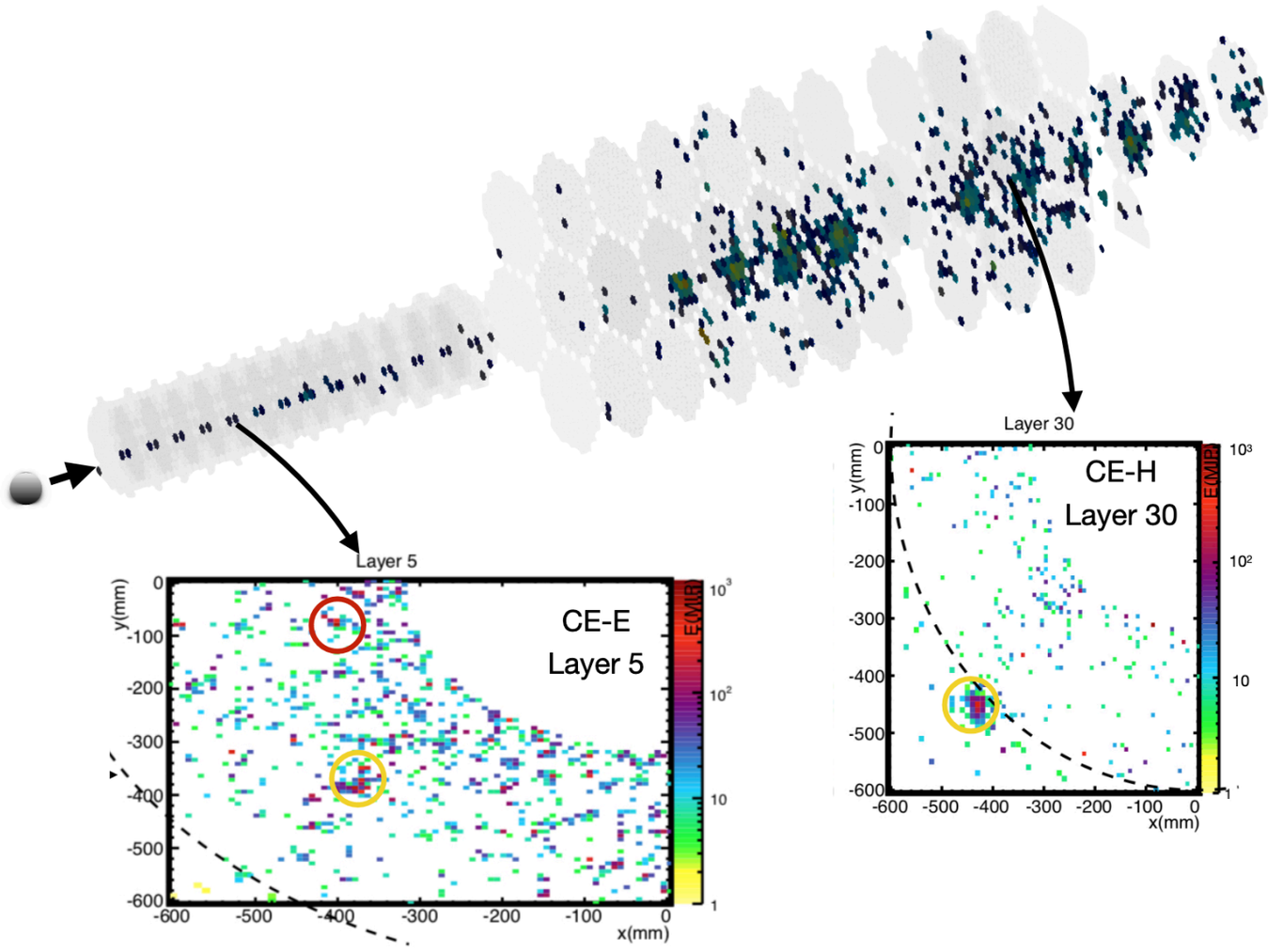
High-Level Trigger

On-prem CPU/GPU filter farm
~100 ms latency

Example:
High-granularity calorimeter @ HL-LHC

Novel technology for future CMS endcap calorimeter:
50 layers with unprecedented number of readout channels (6M)!



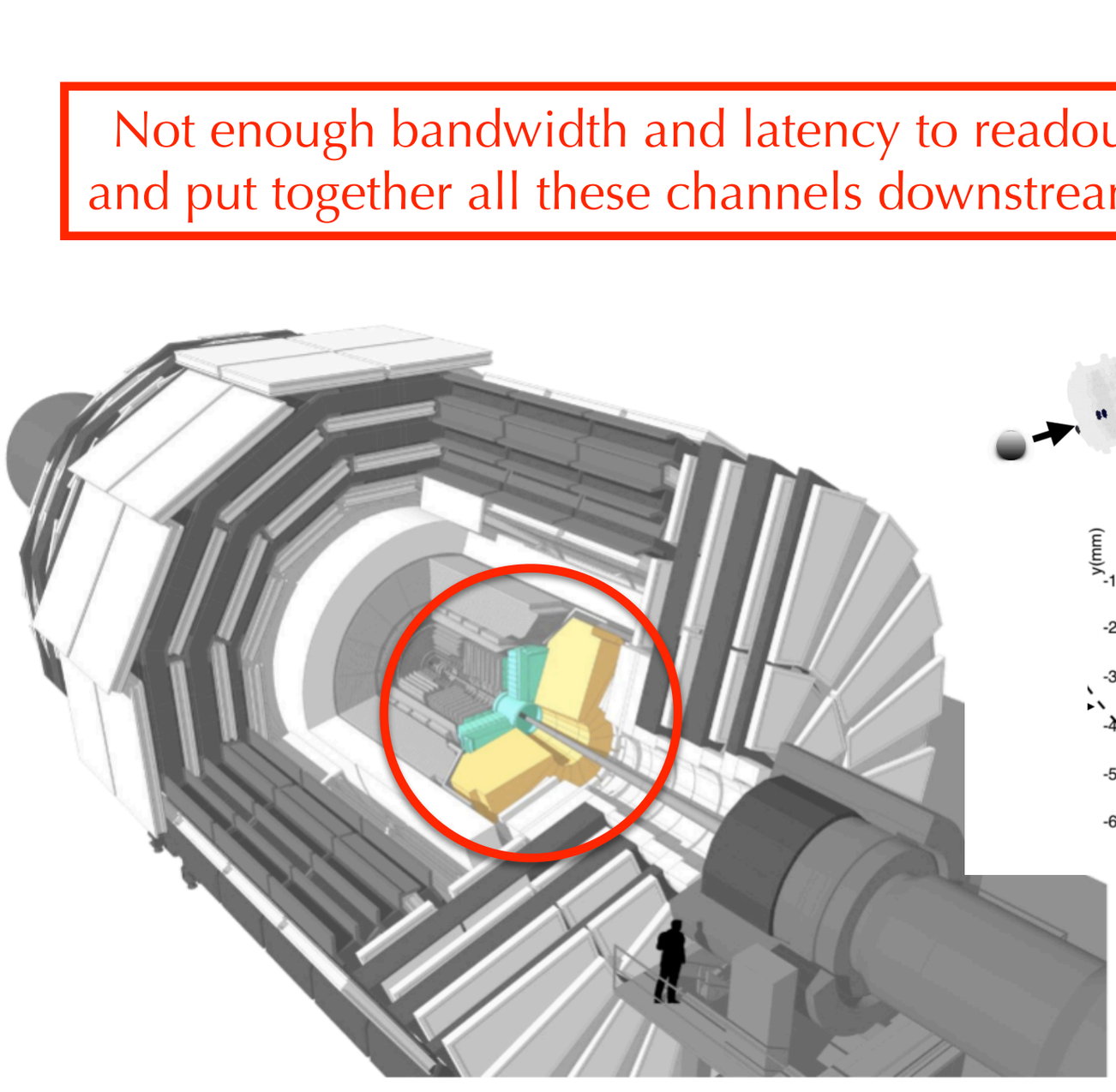


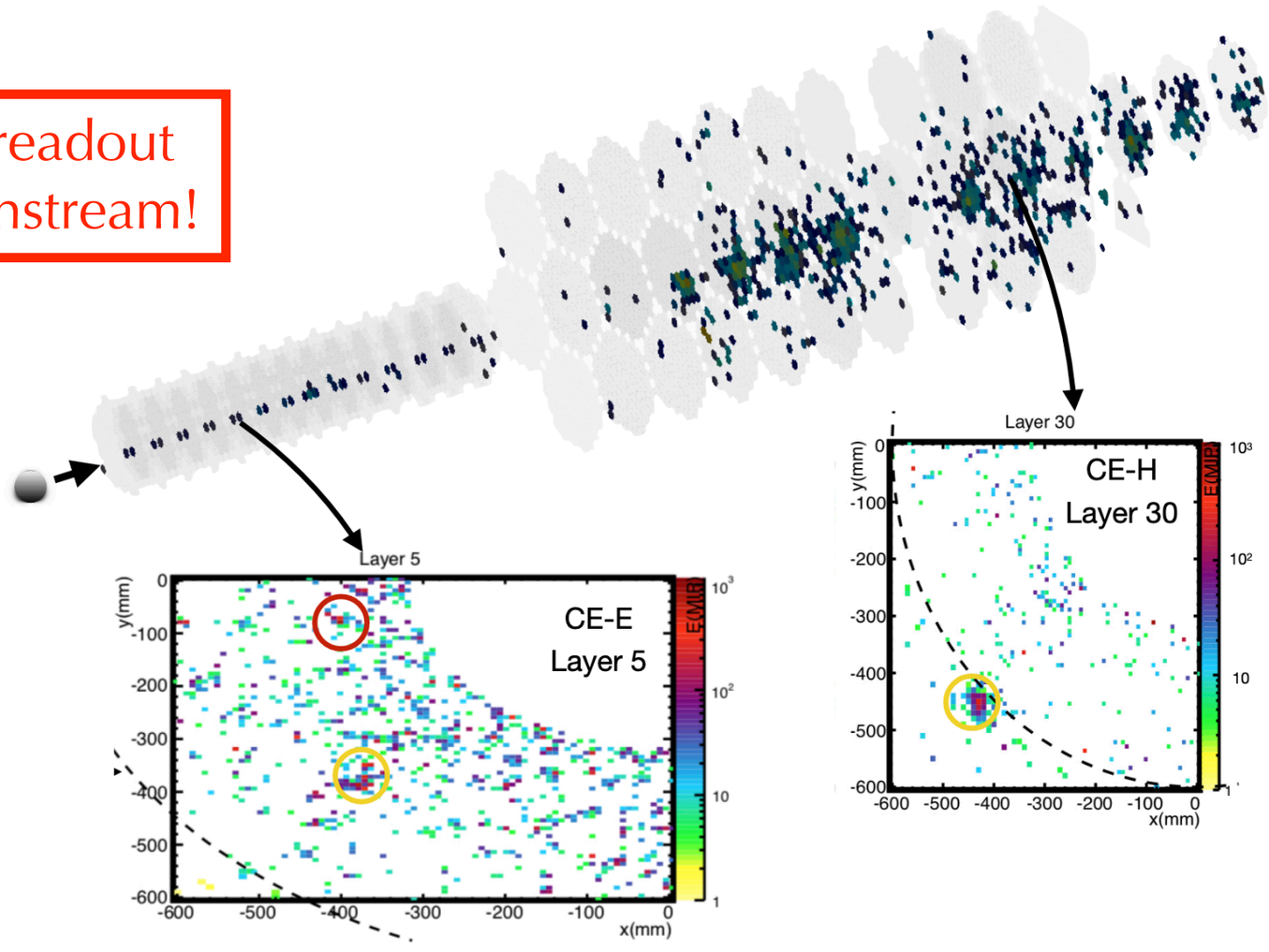
CMS HGCAL TDR

Example:
High-granularity calorimeter @ HL-LHC

Novel technology for future CMS endcap calorimeter:
50 layers with unprecedented number of readout channels (6M)!

Not enough bandwidth and latency to readout and put together all these channels downstream!

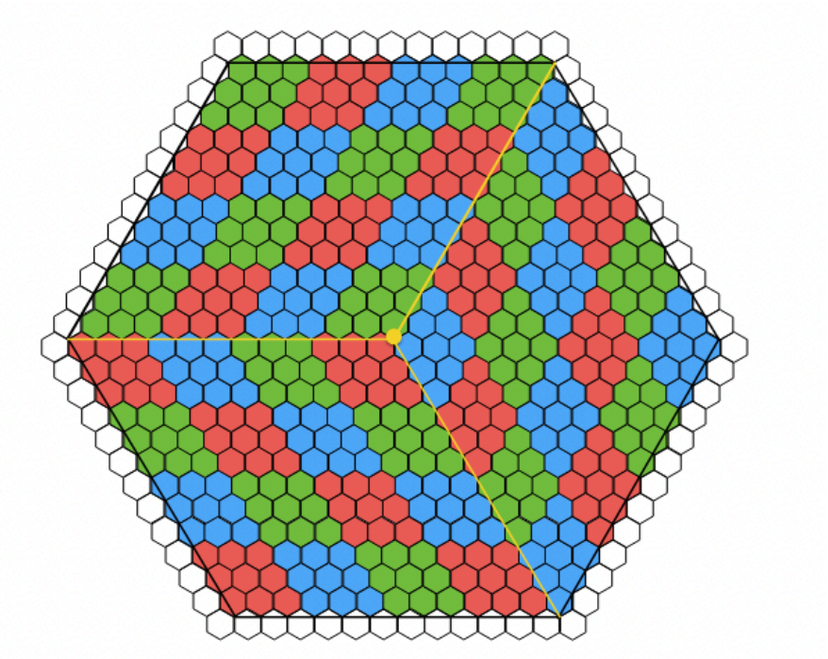




CMS HGCAL TDR

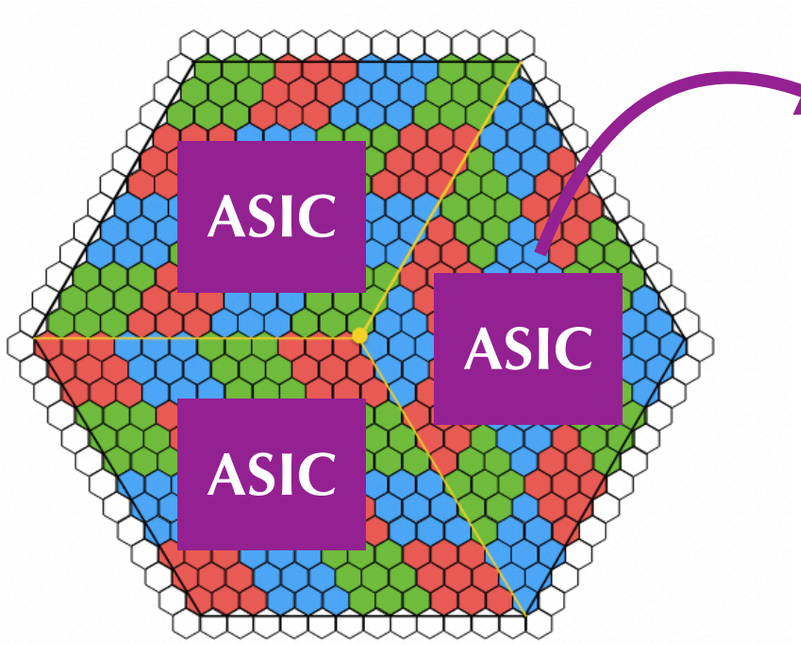
Example: CMS HG calorimeter

One module = 432 sensors

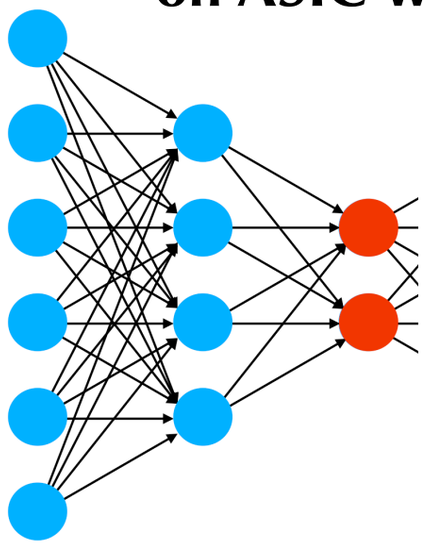


Example: CMS HG calorimeter

One module = 432 sensors



Encode to N bits on ASIC with NN

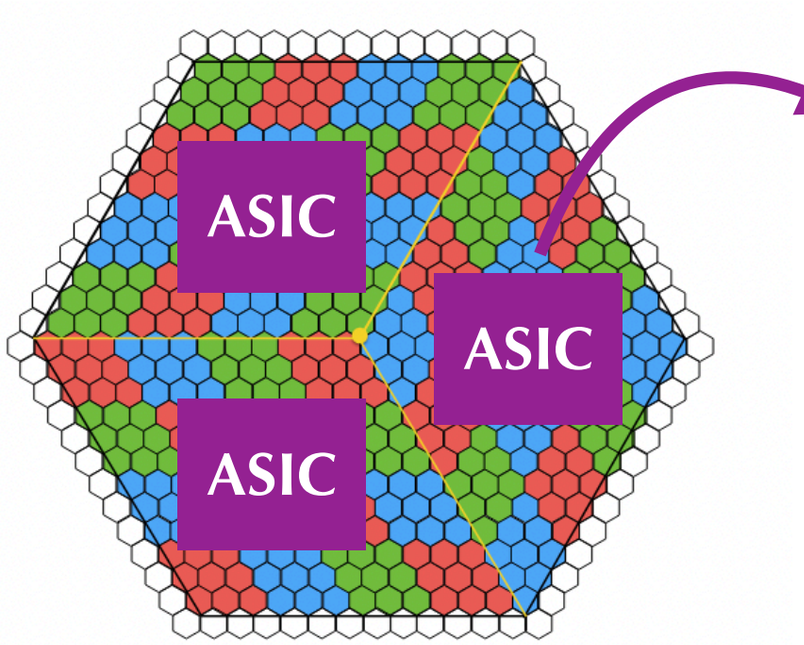


Compress data on sensor in ASIC:

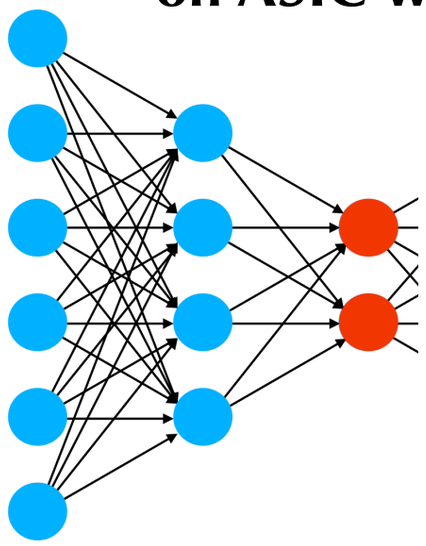
High radiation
Cooled to -30 C → low power
O(100) ns latency

Example: CMS HG calorimeter

One module = 432 sensors

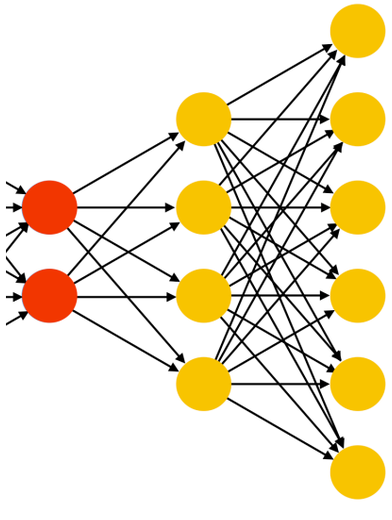


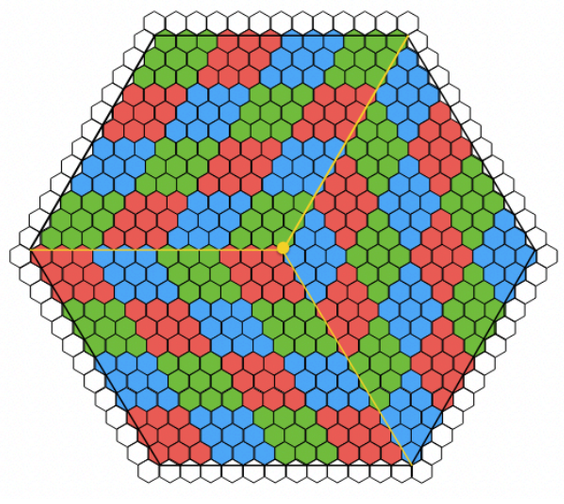
Encode to N bits on ASIC with NN



Transmit encoded data

Reconstruct or do latent space analysis on downstream processors (FPGAs)





Compress data on sensor in ASIC:

High radiation
Cooled to -30 C → low power
O(100) ns latency

AI @ Extreme Edge

- **Tiny and heavily quantized NN**
- NN IP block created for the ASIC with **Catapult HLS (Mentor/Siemens) and hls4ml**
  - NN architecture is fixed, weights can be reprogrammed over I2C
  - NN parameters (weights and biases) triplicated for radiation tolerance → 200% overhead
- Developed in parallel a tool — FKeras — that performs **bit-level sensitivity study of each weight in the NN**
  - allows to prioritize which bits need protection and which may be safely disregarded, reducing resource overhead