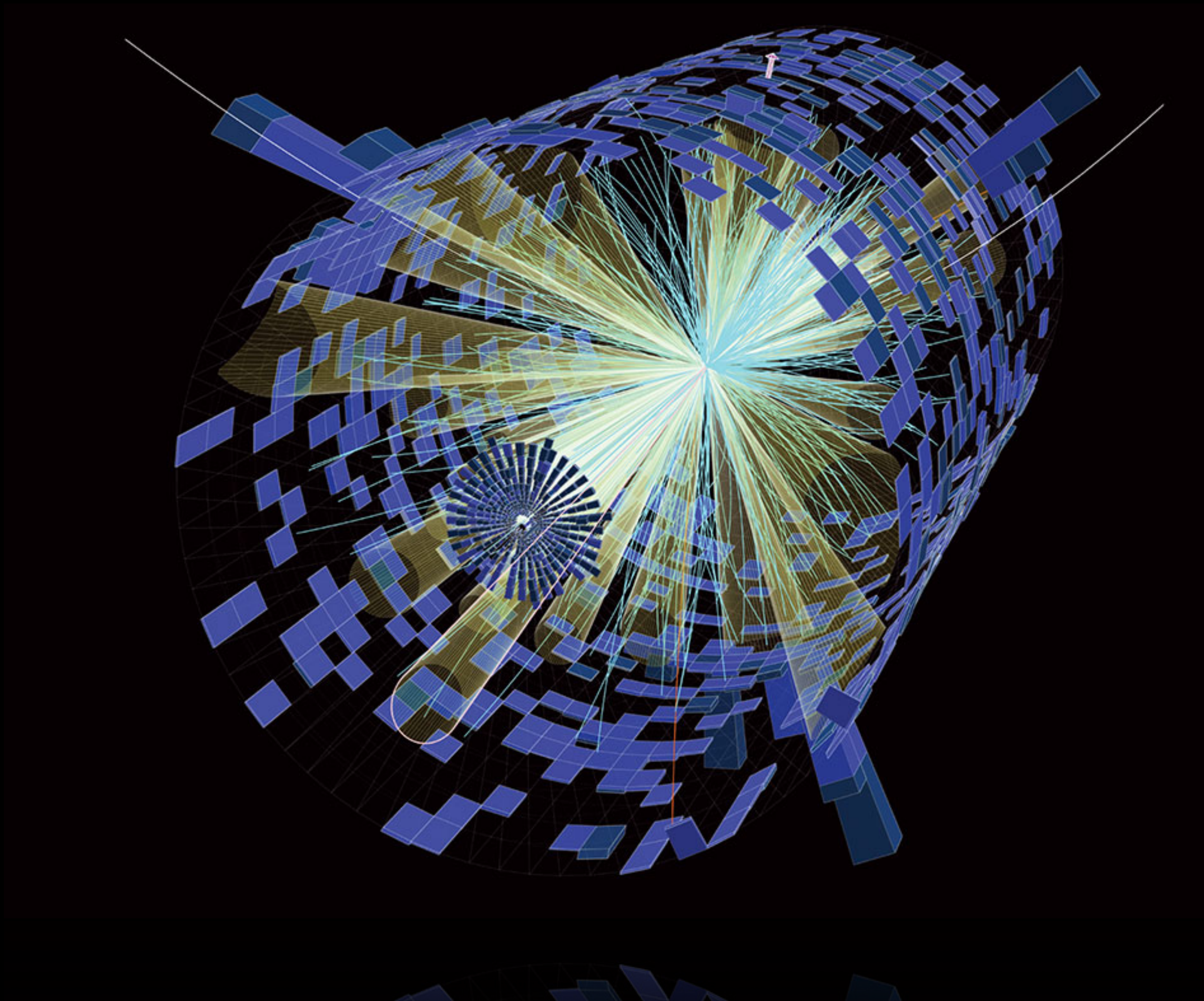
Efficient Machine Learning in High-Energy Physics

Jennifer Ngadiuba (Fermilab)

Workshop on ML on FPGAs for HEP
INFN — Sezione di Bologna
November 2, 2022



Fermilab

CERN

FastML Lab

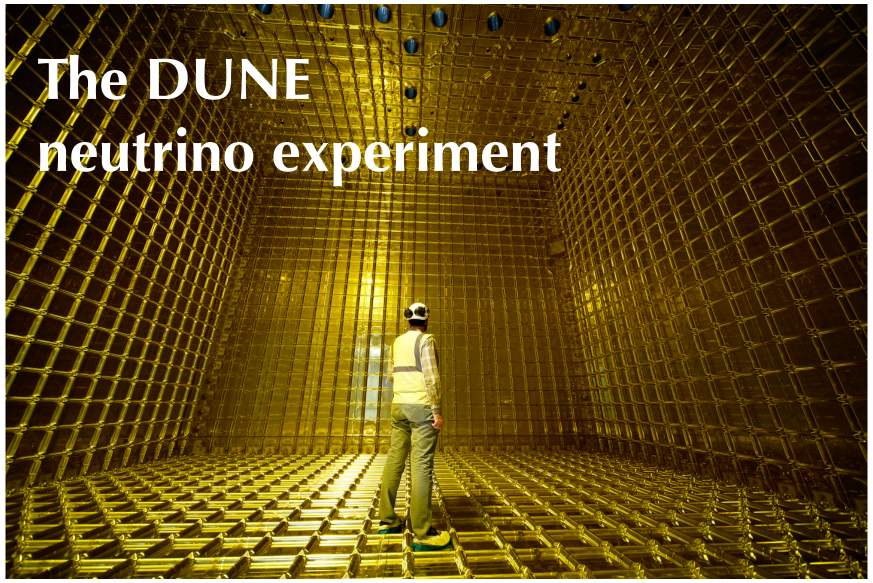
Big science in 21st century

Probing the **fundamental structure of nature** requires complex experimental devices, large infrastructures and big collaborations.

Vast amount of data are being produced by modern-day HEP experiments.

In this era of science **Machine Learning can greatly accelerate time to discovery** allowing us:

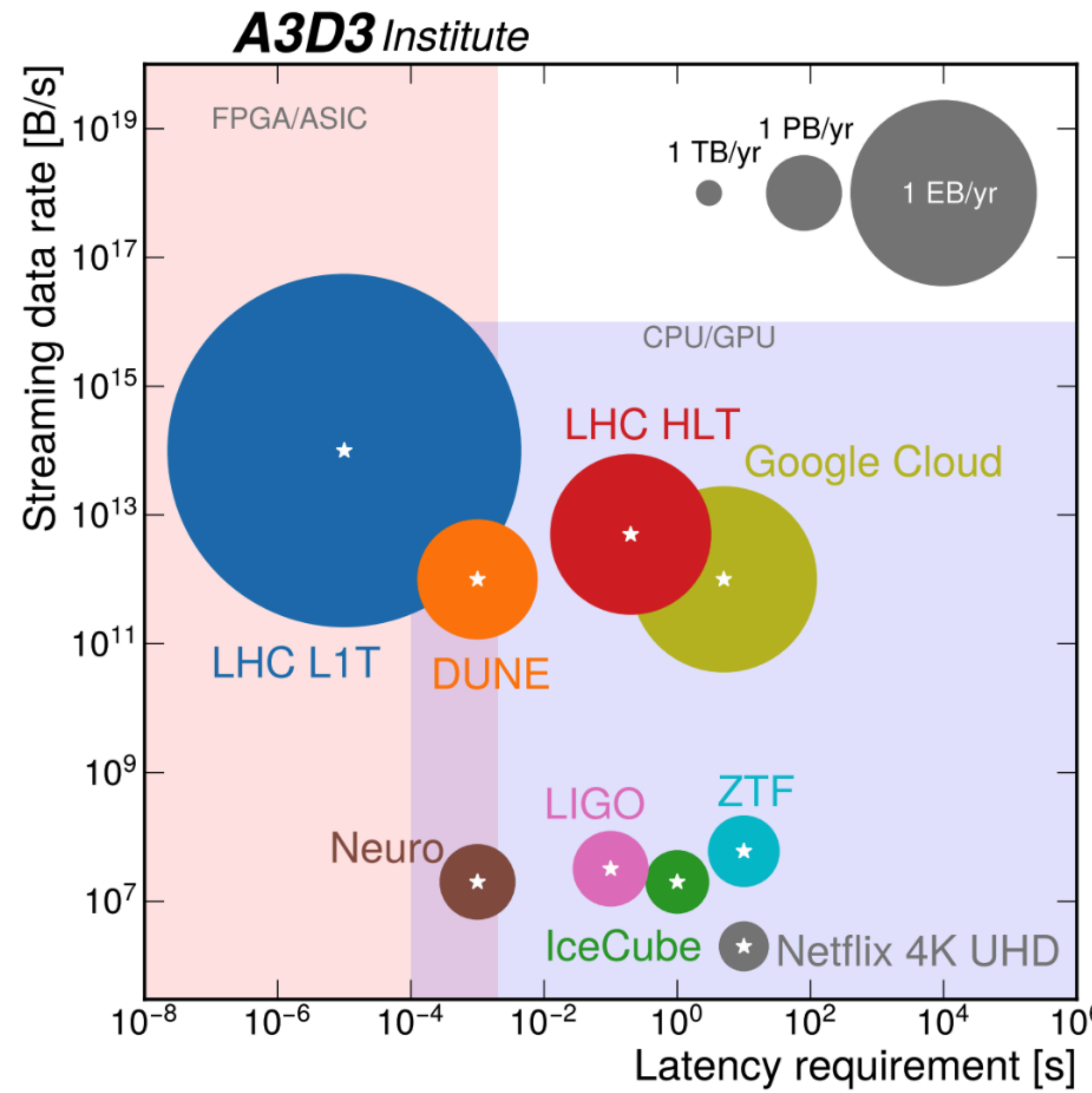
- test hypotheses significantly faster
- enhance and automate performance of detectors/ accelerators
- save and maximize potentially lost data



Big Science = Big Data

- Requirements for ML in particle physics go far beyond industrial and commercial applications because of extreme environments:
  - speed, throughput, fidelity, interpretability, and reliability

- **At the extreme edge of throughput requirements HEP experiments need efficient real time ML able to meet the most challenging latency constraint!**

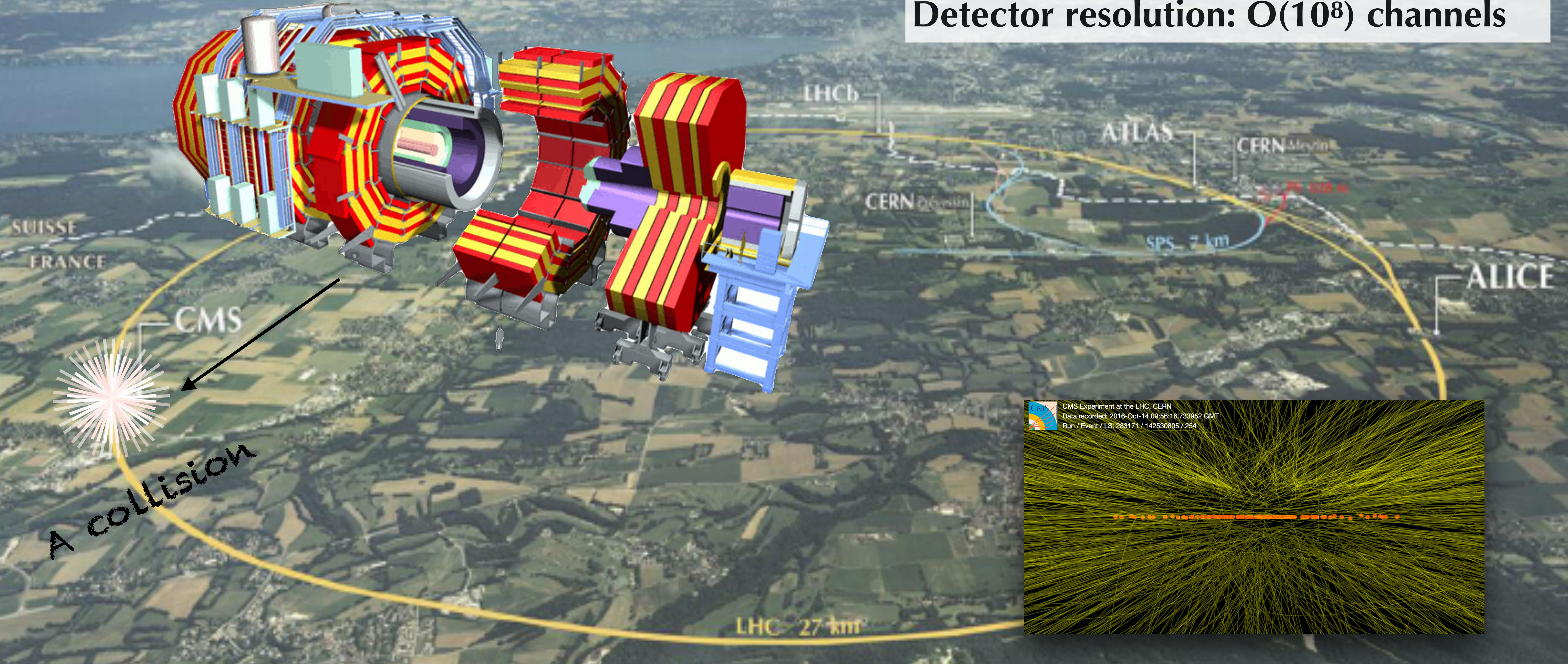


https://a3d3.ai/

Big Data @ the Energy Frontier

The Large Hadron Collider (LHC)

Collision frequency: 40 MHz
Particles per collision: $O(10^3)$
Detector resolution: $O(10^8)$ channels

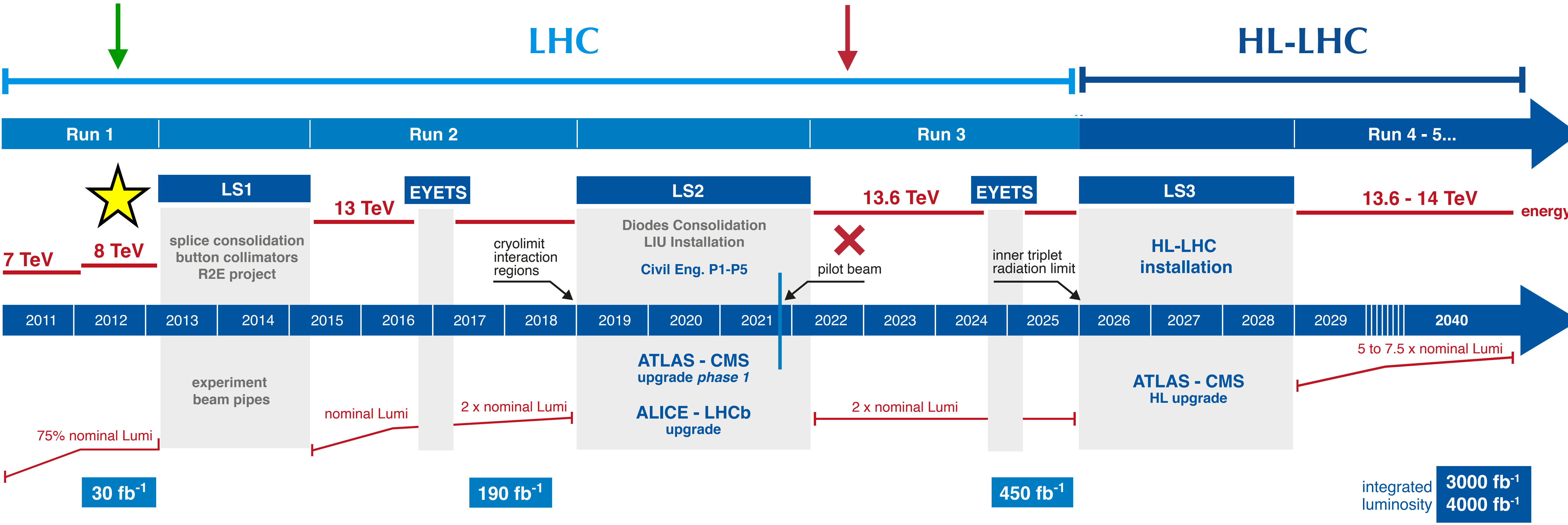


Extreme data rates of ~PB/s!

The HL-LHC challenge

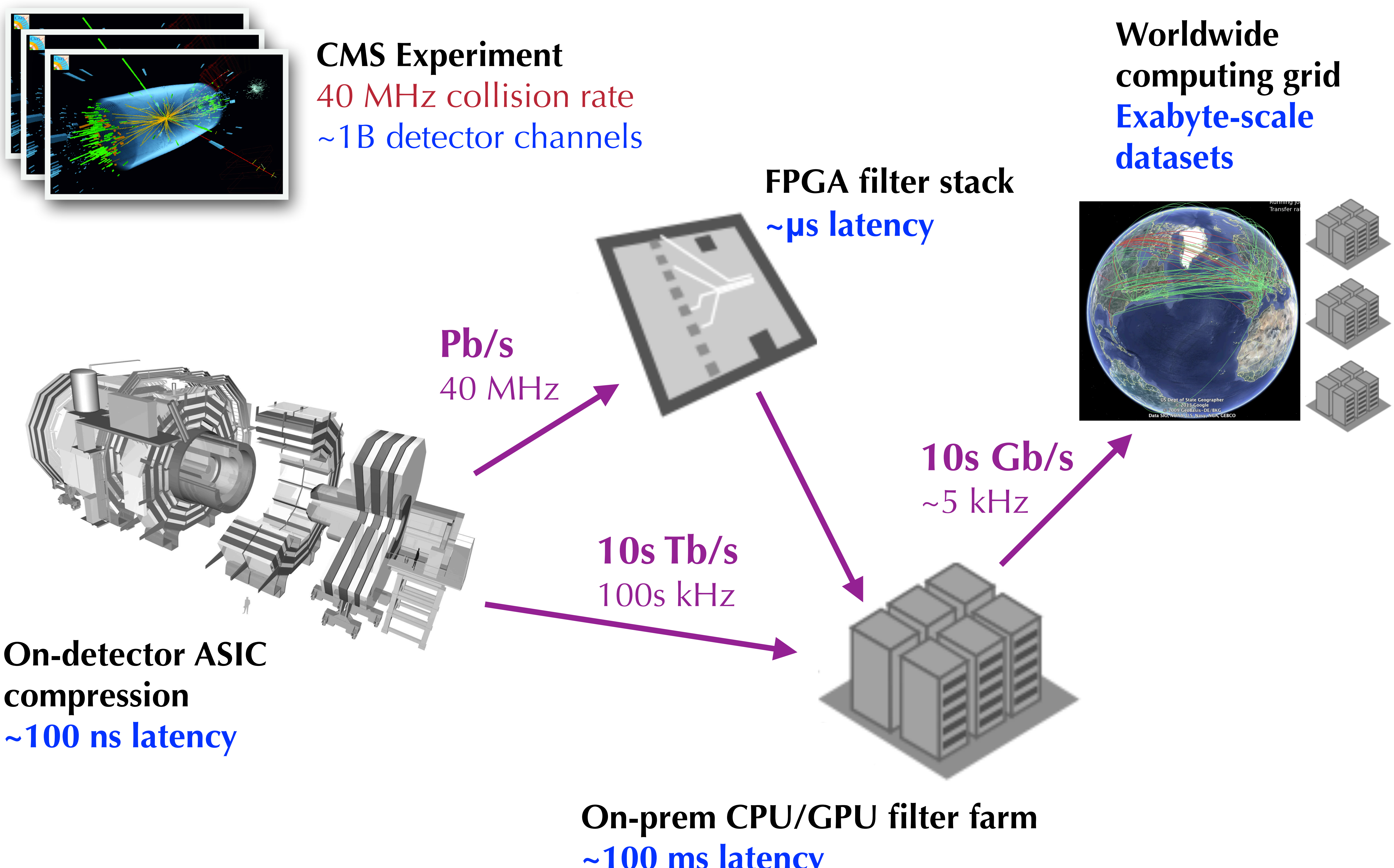
2012:
discovery of the Higgs boson

Now!

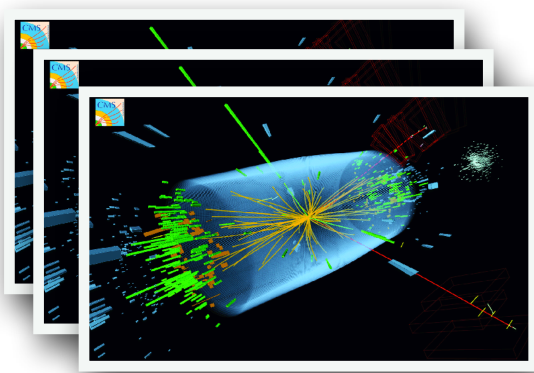


x 5	x 20	x 50	x 20
Event complexity	**Data**	**Processing time**	**Computing resources/ Disk storage**

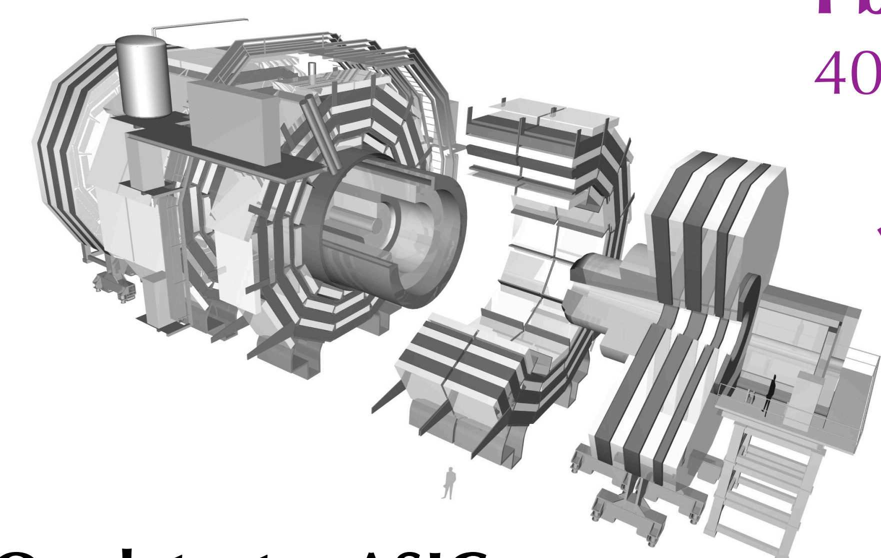
Data reduction workflow @ LHC



Data reduction workflow @ LHC



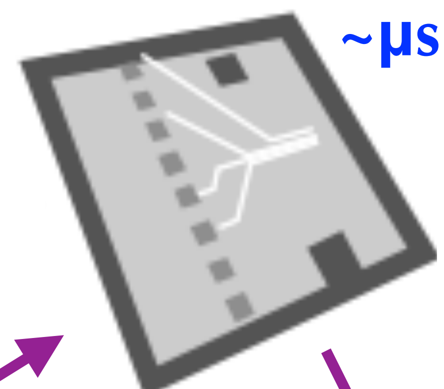
CMS Experiment
40 MHz collision rate
~1B detector channels



On-detector ASIC compression
~100 ns latency

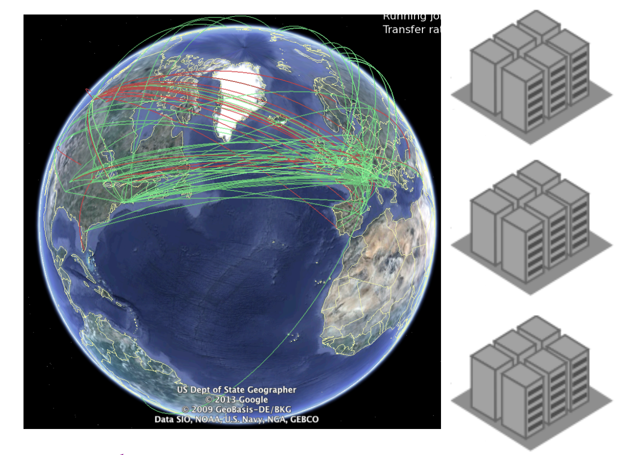
Pb/s
40 MHz

10s Tb/s
100s kHz



FPGA filter stack
~μs latency

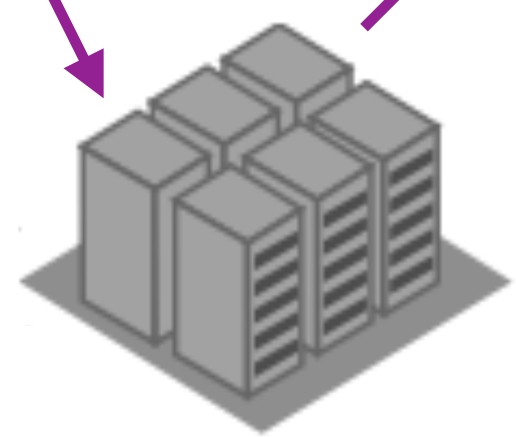
Level-1 Trigger



Worldwide computing grid
Exabyte-scale datasets

10s Gb/s
~5 kHz

Offline analysis

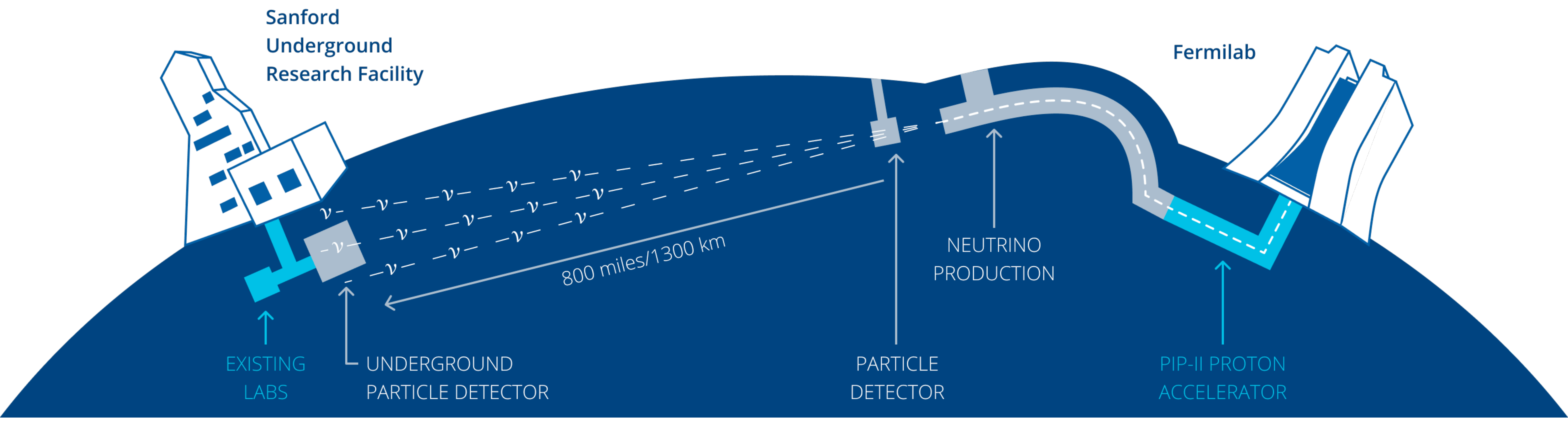


High-Level Trigger

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

Big data @ the Intensity Frontier

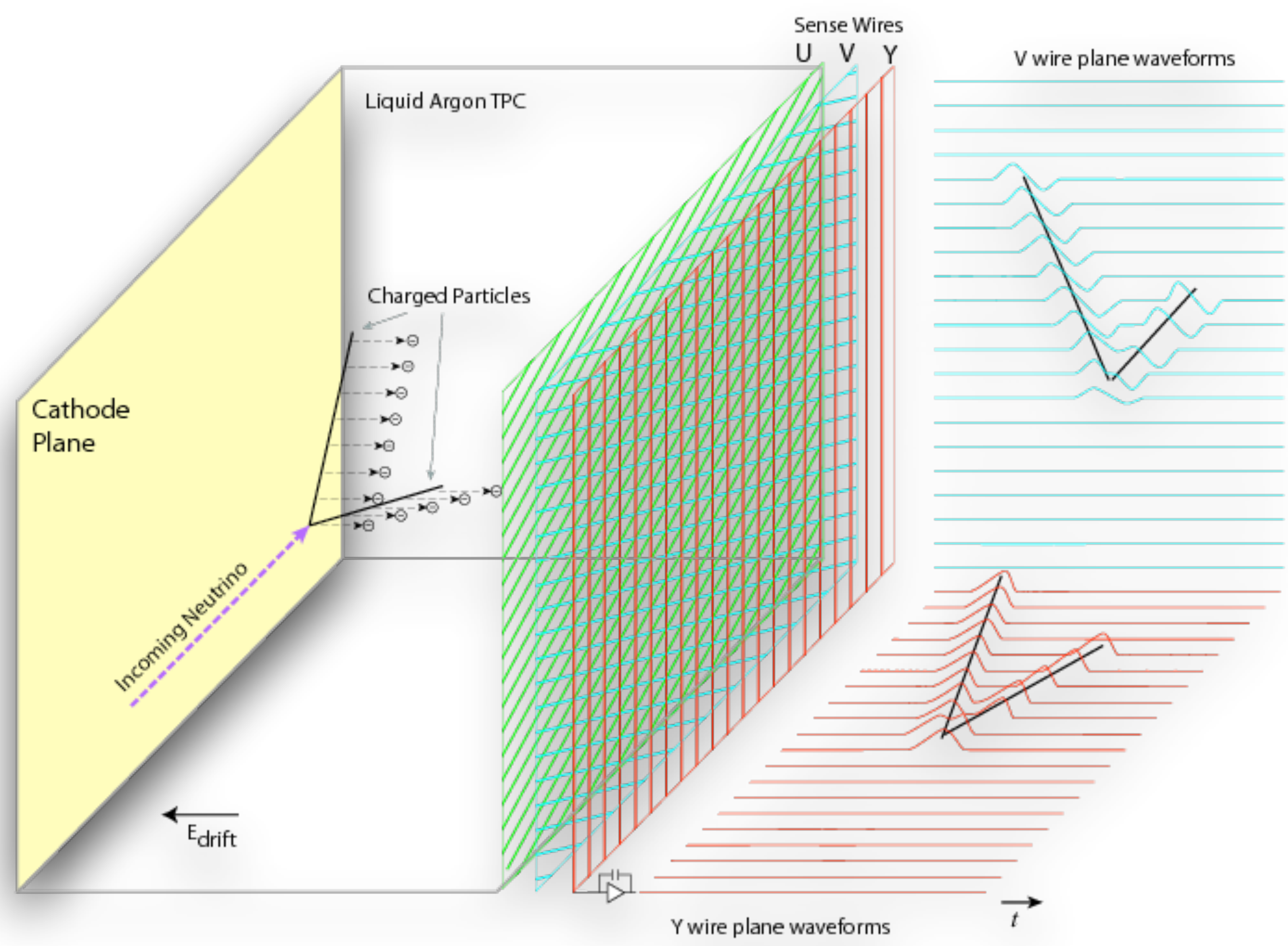
The Deep Underground Neutrino Experiment (DUNE)



- Next generation neutrinos oscillation experiment now under construction and R&D to start operations by the end of current decade
- Massive far detector 1 mile underground comprising **70k tons of LAr** and advanced technology (LAr Time Projection Chambers) to record neutrino interactions with extraordinary precision

Big data @ the Intensity Frontier

Operating principle of a LArTPC



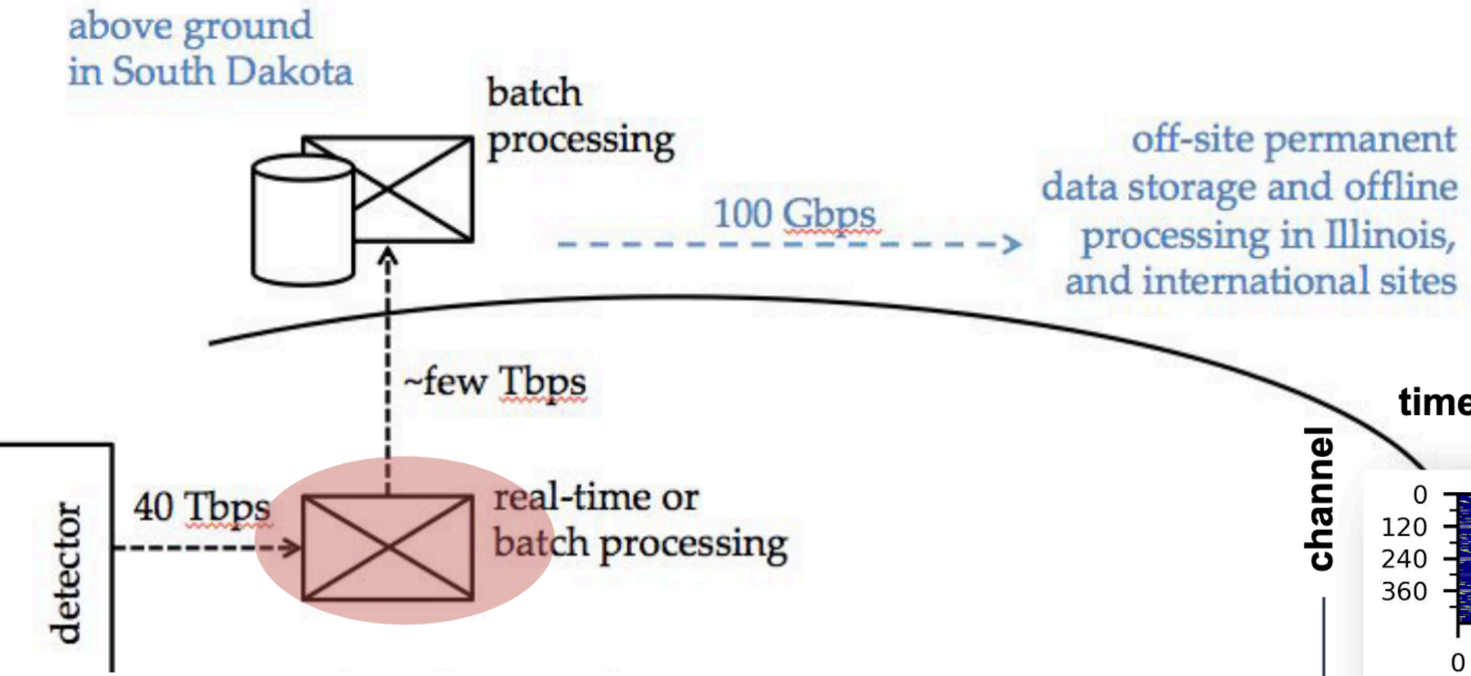
Electrons are produced by charged particles interacting with a large multiple-cubic meters volume of LAr

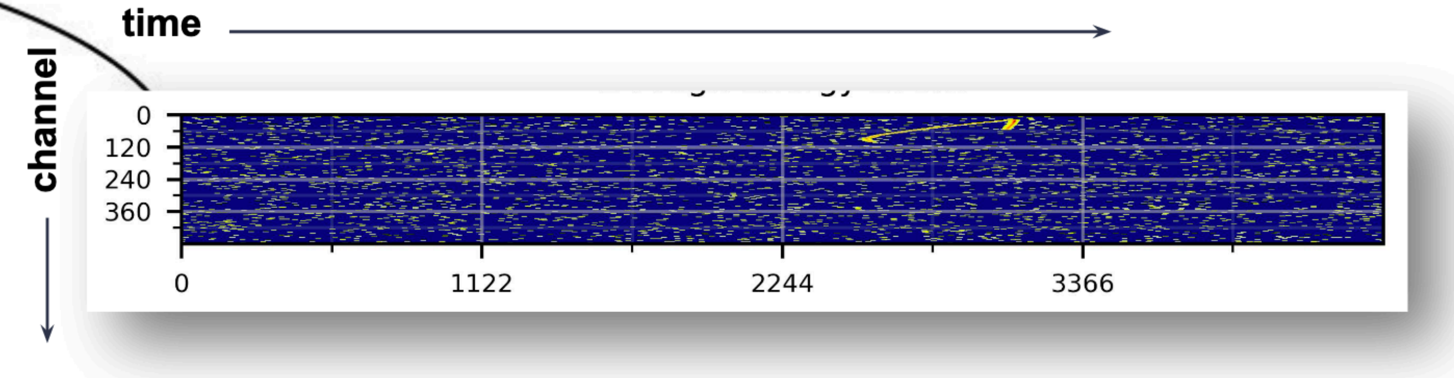
Continuous stream of 3D images of detector volume yielding a **high-resolution "video"**:

4 modules x 150 cell volumes
O(10) MB / frame
O(10^5) frames / s for 2.25 ms
a total of ~40 Tb/s

With continuous operation for more than a decade expected Zettabytes of data!

Big data @ the Intensity Frontier

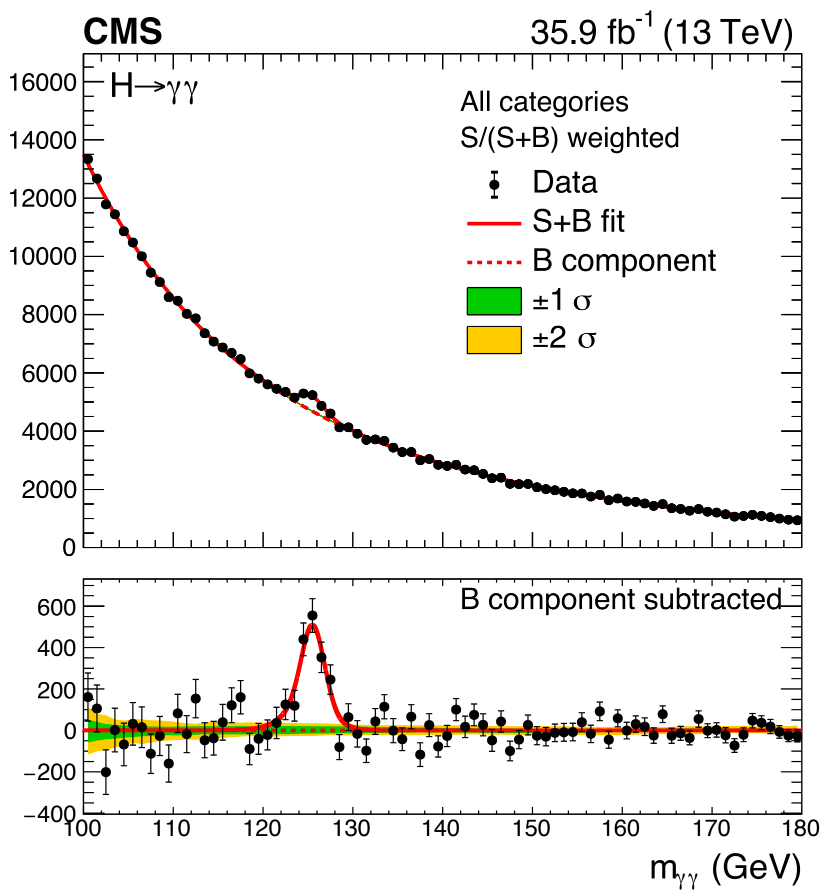




- Trigger decision made underground to achieve a **10^4 data reduction factor**
- Half of 150 cells processed in parallel in custom low power Xilinx FPGA board
- Coarse **first level of filtering on a per-cell basis**
- **Second level aggregates low-level information from all cells** in a single module to make a module-level trigger decision
  - executed on CPU resources with O(s) latency
  - positive decision initiate readout of 2.25 ms worth of continuous data from all 150 cells

Boosting ML efficiency

- **Machine Learning** has been used since long time in HEP in offline analyses and found **crucial to maximize the physics output**

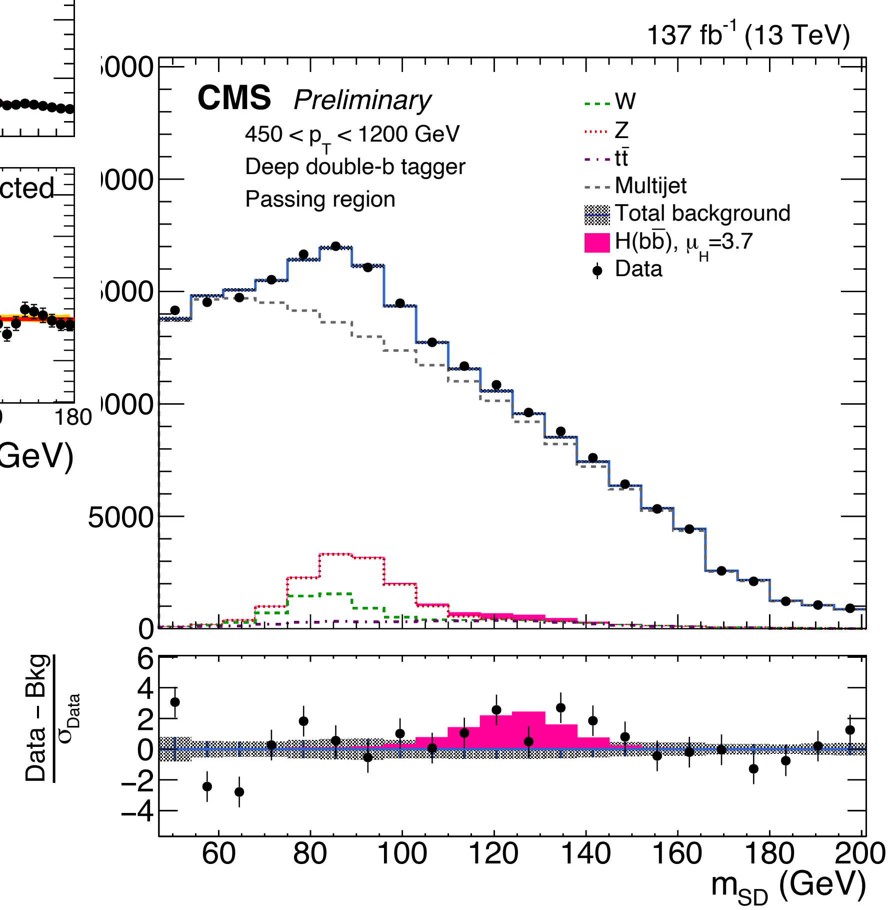


Higgs → photons

Phys. Lett. B 805 (2020) 135425

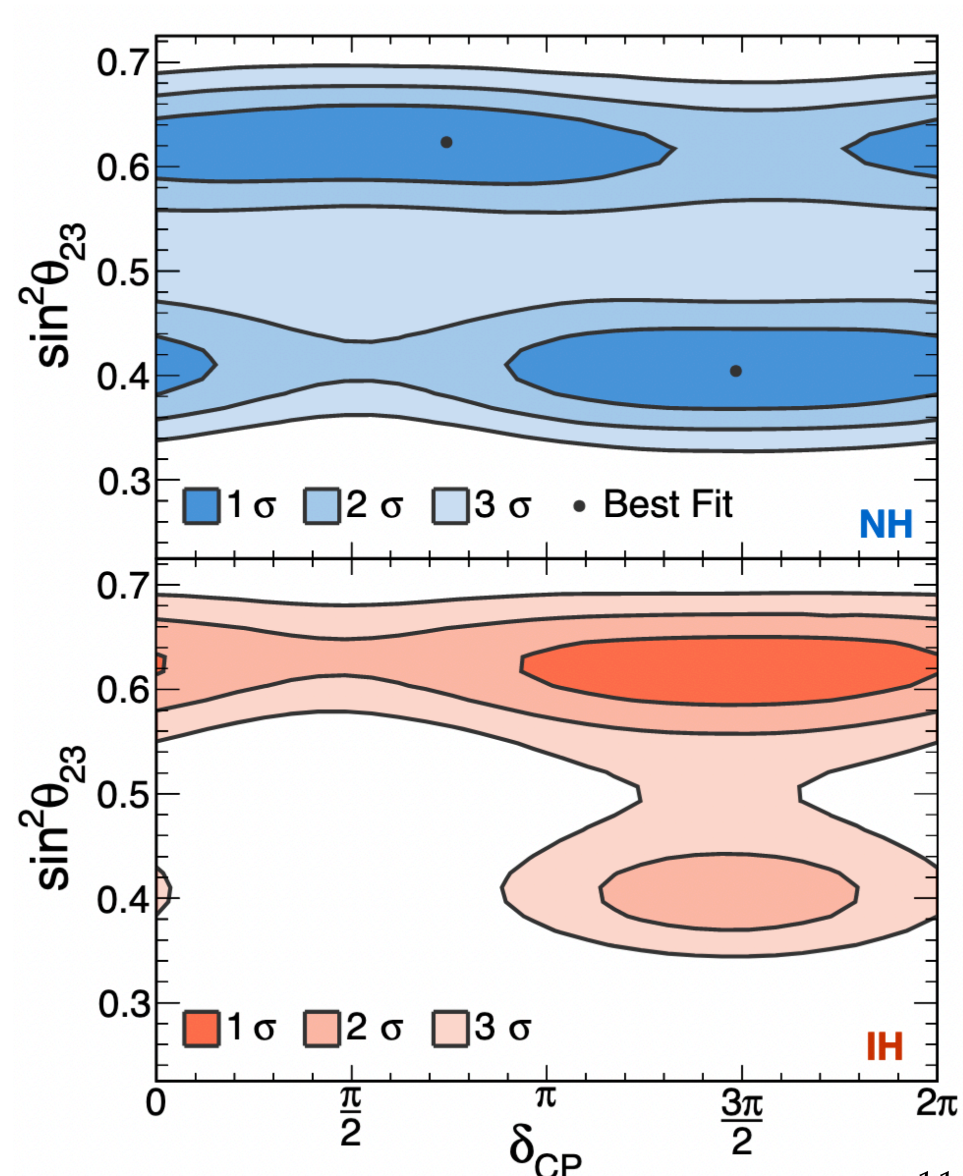
Higgs → bottom quarks

CERN-EP-2020-107



Measurement of neutrino oscillation parameters @ NovA

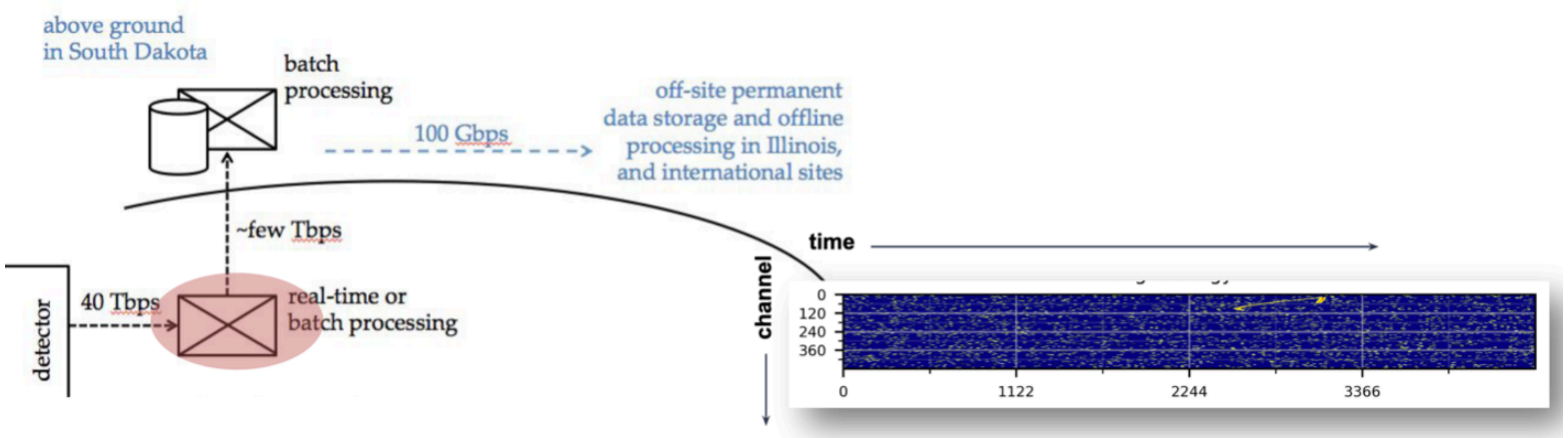
Phys. Rev. Lett. 118, 231801 (2017)

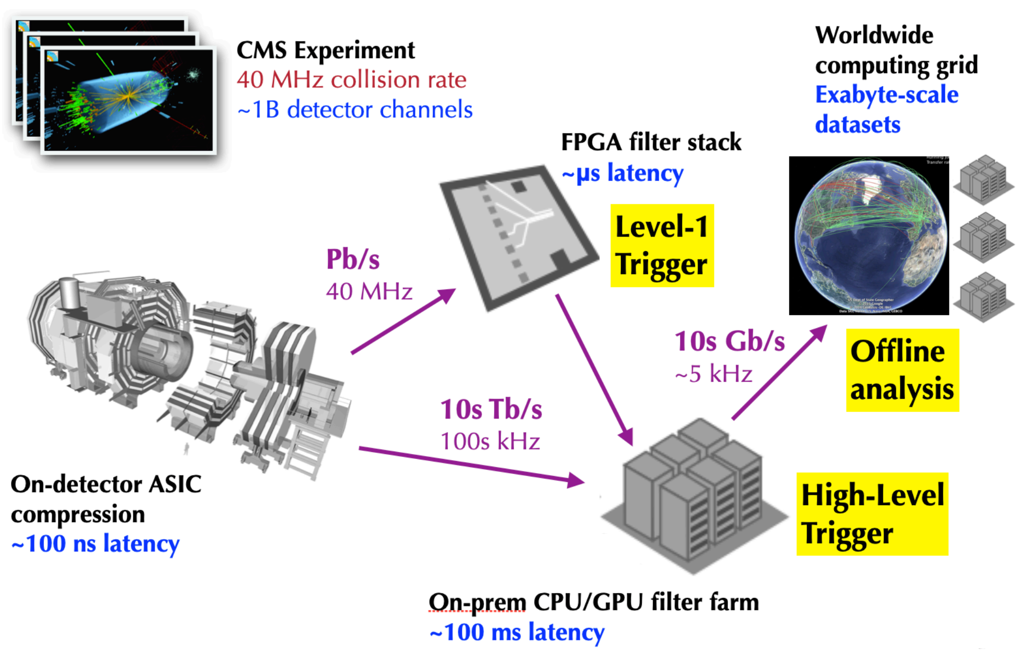


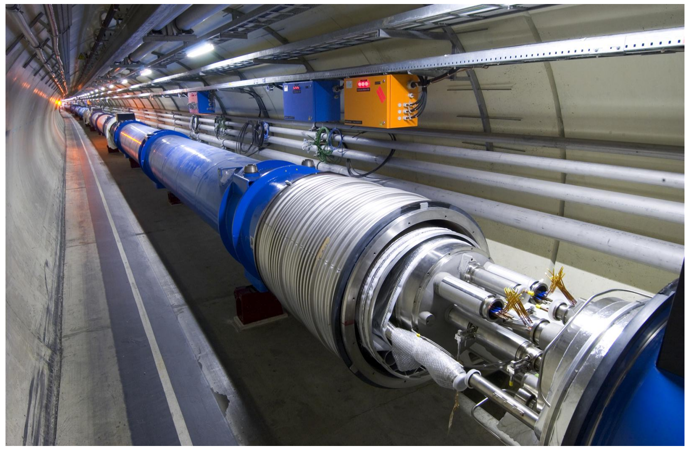
Boosting ML efficiency

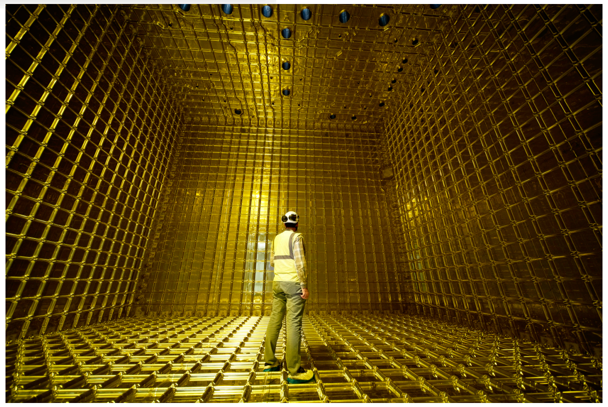
- **Machine Learning** has been used since long time in HEP in offline analyses and found **crucial to maximize the physics output**

- As experiments grow in sophistication it is crucial to **bring these powerful algorithms closer to the detector for a more efficient features extraction**









Boosting ML efficiency

- **Machine Learning** has been used since long time in HEP in offline analyses and found **crucial to maximize the physics output**

- As experiments grow in sophistication the more urgent is the need to **bring these powerful algorithms closer to the detector for a more efficient features extraction**

The wishes:

99% accurate
O(µs) fast
Minimum resources (O(1) FPGAs)
Interpretable and robust

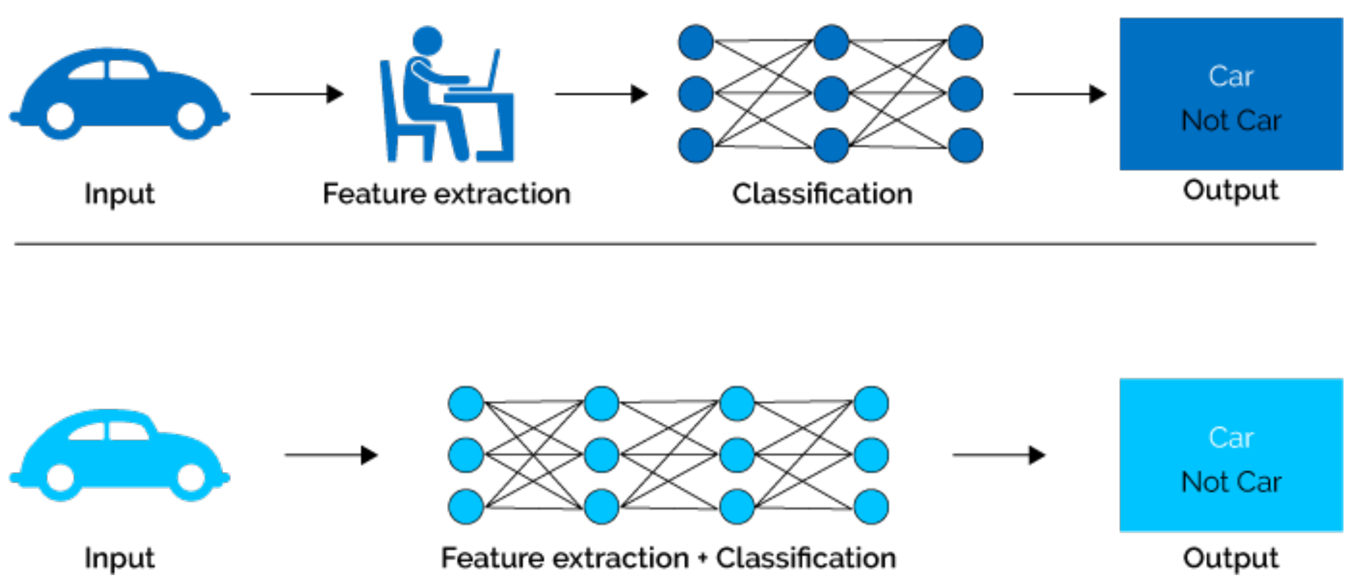
Some solutions:

Inductive bias
Hardware codesign tools

Boost efficiency with inductive bias

- **Straightforward approach:** start with expert domain features and combine them in a shallow dense neural network
  - **PROS:** interpretable input features, high NN computational efficiency
  - **CONS:** rely entirely on the informativeness of such new features, expert features computation typically not efficient (ex, full reconstruction not possible at 40 MHz)

- **Not straightforward approach:** automate the expert feature extraction process from raw features with DNNs where each new layer captures a more abstract representation of the data
  - **PROS: highest accuracy**
  - **CONS: computational efficiency does not come for free**

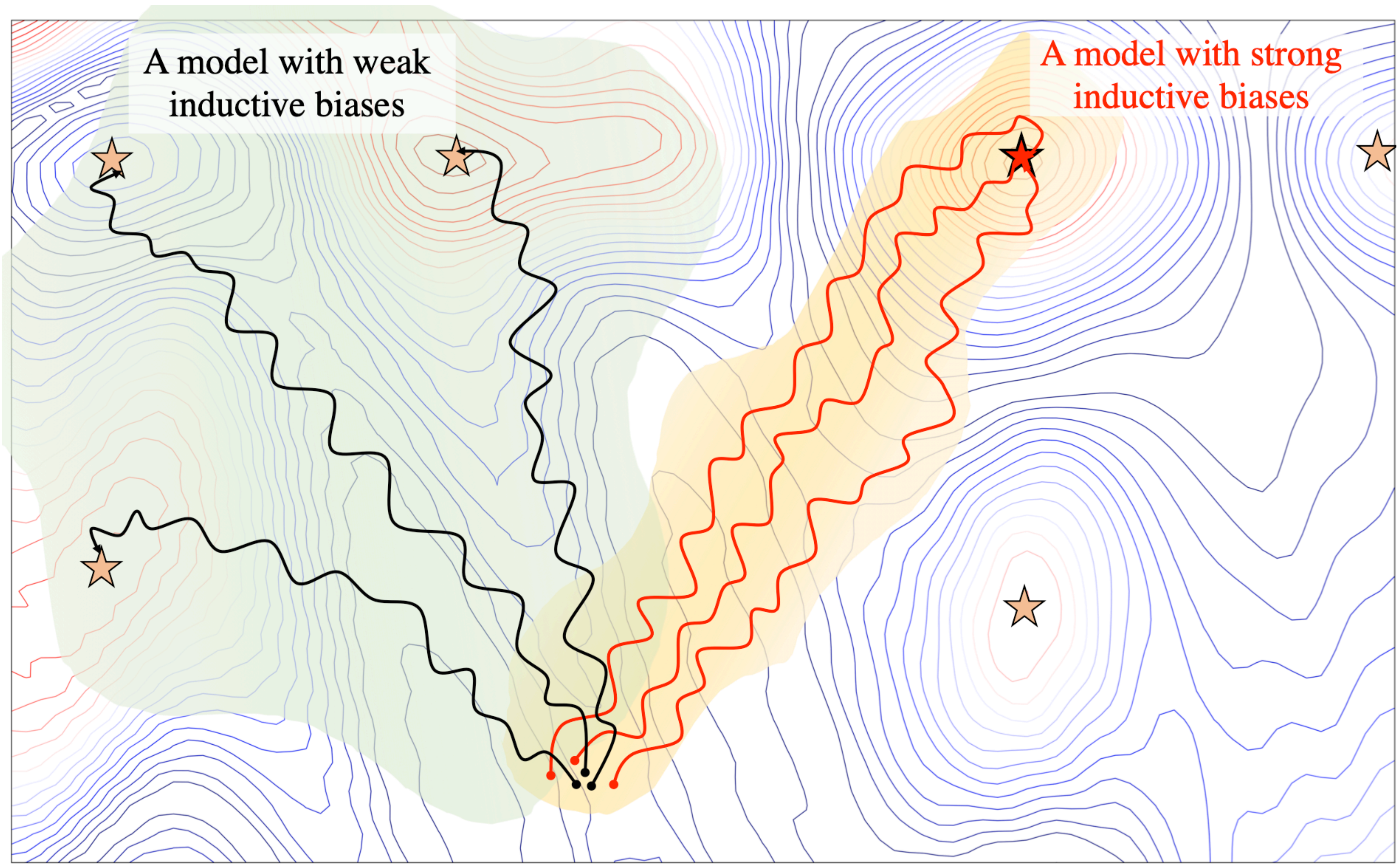


A shallow neural network

A deep neural network

Boost efficiency with inductive bias

- Incorporating **domain knowledge** into ML *(inductive bias)* can provide **better accuracy, training/inference efficiency, smaller model size, interpretability and robustness** against domain shift

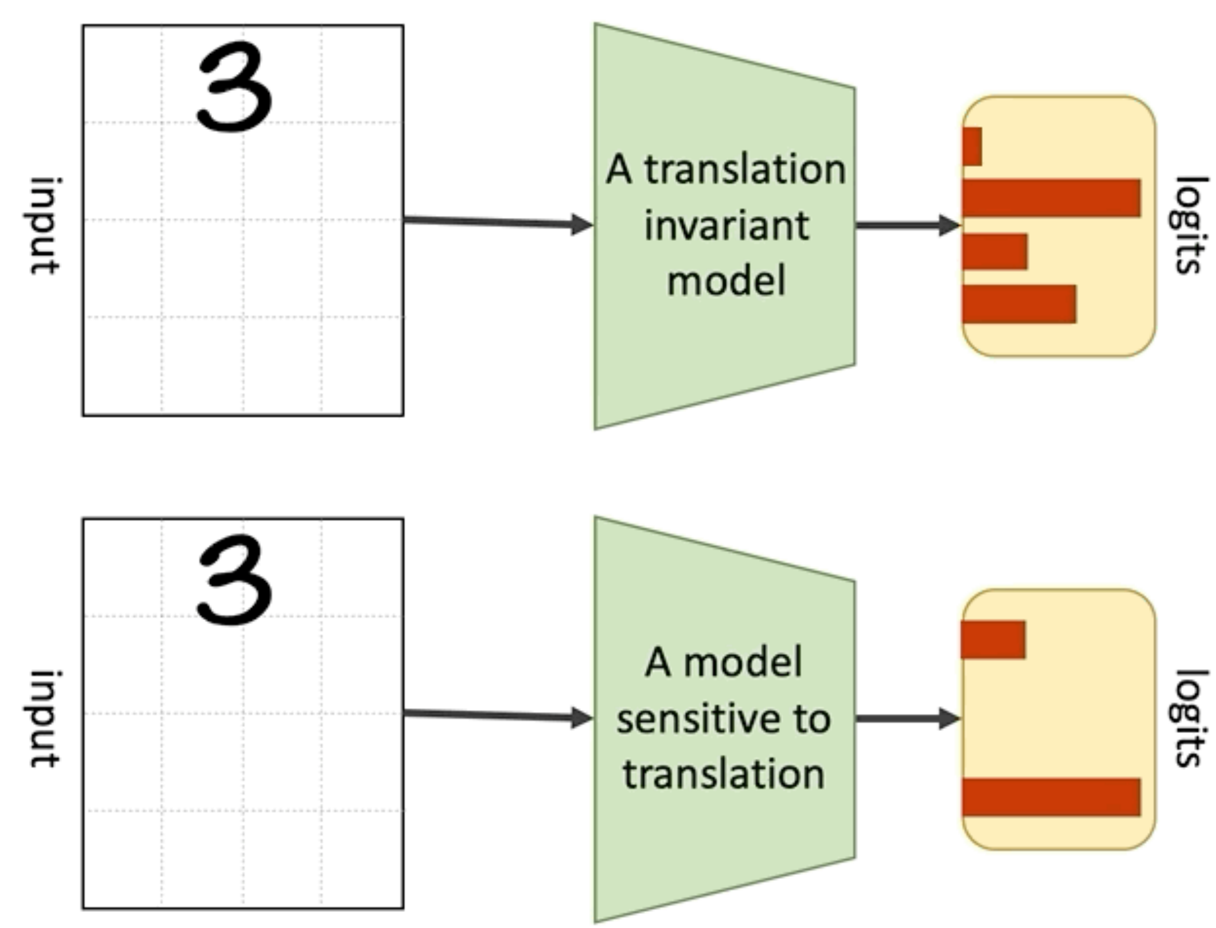


https://samiraabnar.github.io/articles/2020-05/indist

Example: Convolutional NN

- CNNs was a breakthrough: tailored algorithms to the structure (and symmetries) of the data

- **Leverage spatial symmetries** (translation invariance and equivariance) to achieve higher accuracy at lower computational cost wrt Dense NNs
    - intelligent feature extraction from raw pixel-level high-dimensional data with less parameters



Example: Convolutional NN

- CNNs was a breakthrough: tailored algorithms to the structure (and symmetries) of the data

- **Leverage spatial symmetries** (translation invariance and equivariance) to achieve higher accuracy at lower computational cost wrt Dense NNs
  - intelligent feature extraction from raw pixel-level high-dimensional data with less parameters

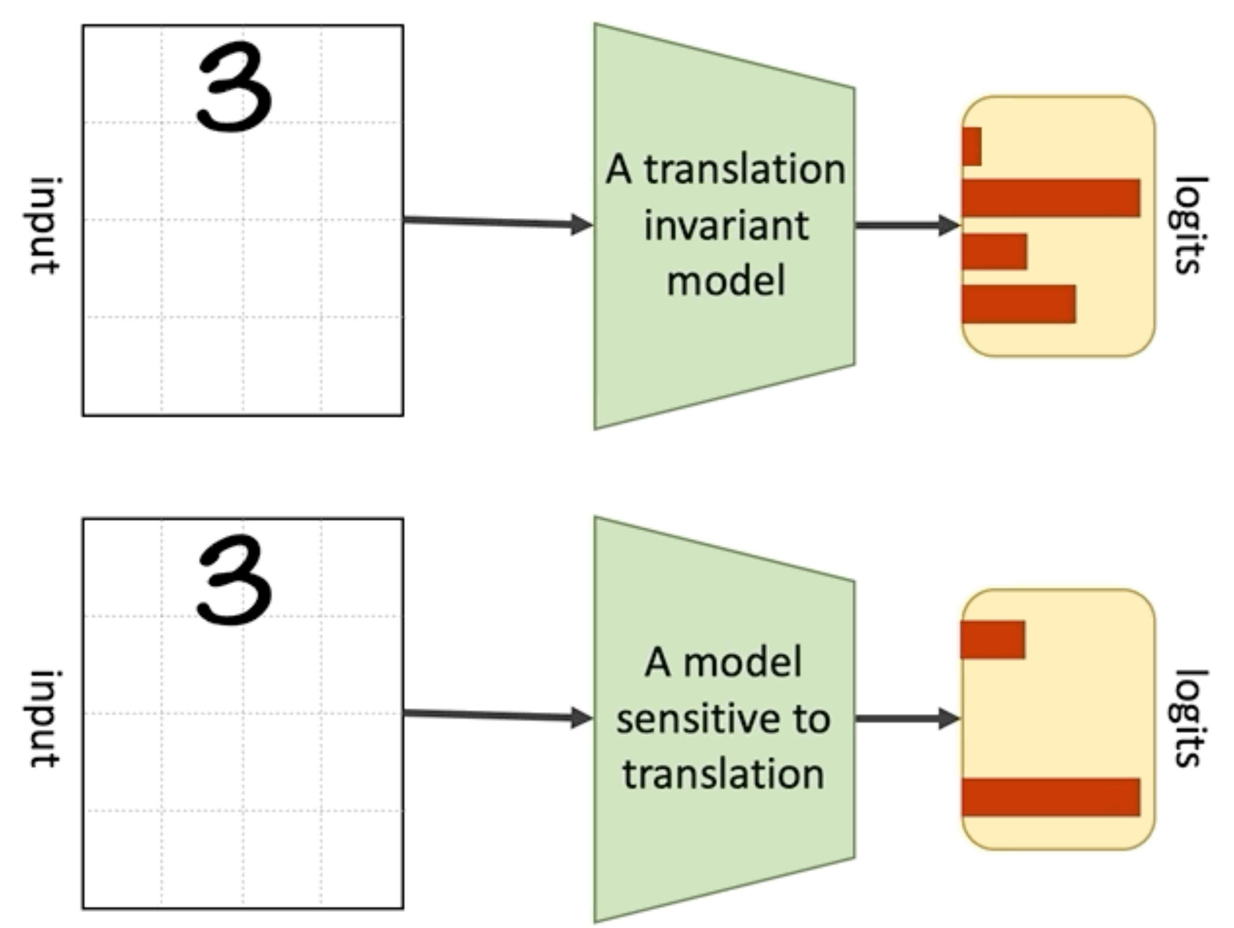


Image data vs HEP data

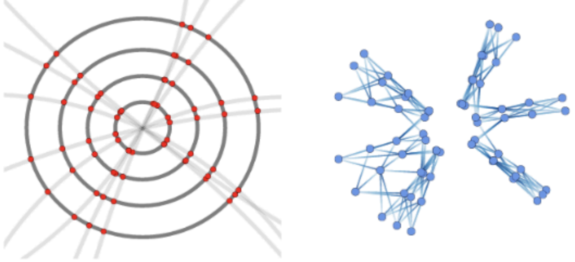
- CNNs was a breakthrough: tailored algorithms to the structure (and symmetries) of the data

- **Leverage spatial symmetries** (translation invariance and equivariance) to achieve higher accuracy at lower computational cost wrt Dense NNs
  - intelligent feature extraction from raw pixel-level high-dimensional data with less parameters

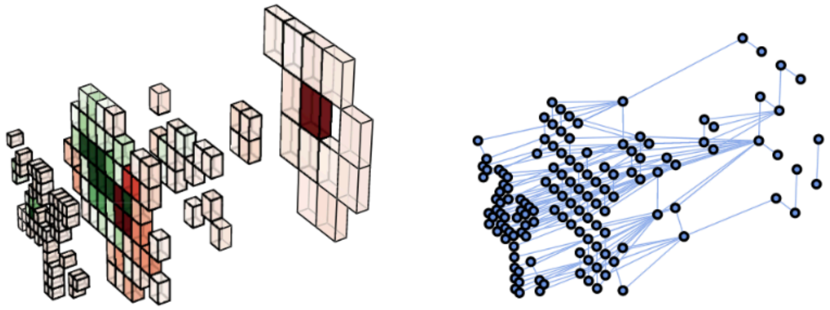
- **What about HEP data?**
  - Distributed unevenly in space
  - Sparse
  - Heterogenous
  - Variable size
  - No defined order
  - Interconnections

↓

Graph Neural Networks



(a)



(b)

arXiv.2203.12852

Graph NNs for HEP

- Represent objects as points with pairwise relationships
- Effectively capture complex relationships and dependencies between objects of many different kinds in HEP
  - energy deposits, individual physics objects, individual particles, heterogenous information
- **Applications and architectures keep successfully growing!**

Static isotropic
- E.g. GCN

Static anisotropic
- E.g. Interaction Network

Dynamic (An)isotropic
- E.g. GravNet

Node prediction
- E.g. Node regression or classification

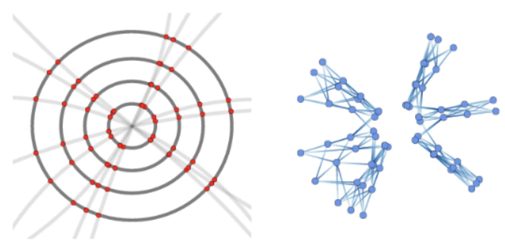
Edge prediction
- E.g. Social network link prediction

Graph prediction
- E.g. Molecular property regression

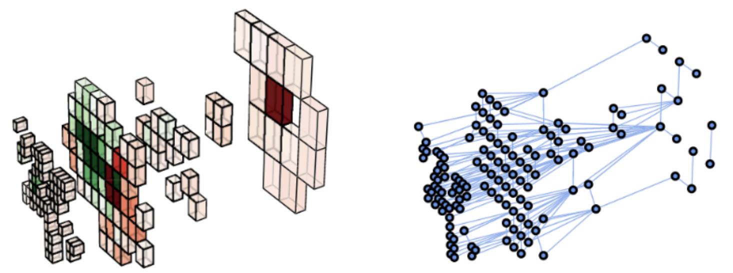
Object segmentation
- E.g. Find all hydrogen in graph

Instance segmentation
- E.g. Find *each* hydrogen in graph

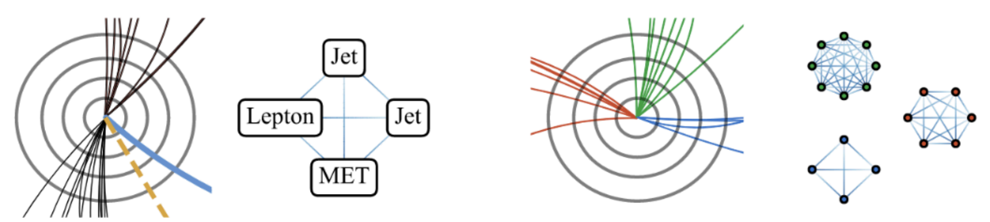
Spatio-Temporal
- E.g. STGCN (Graph conv. + temporal conv.)



(a)



(b)



arXiv.2203.12852 arXiv.2007.13681

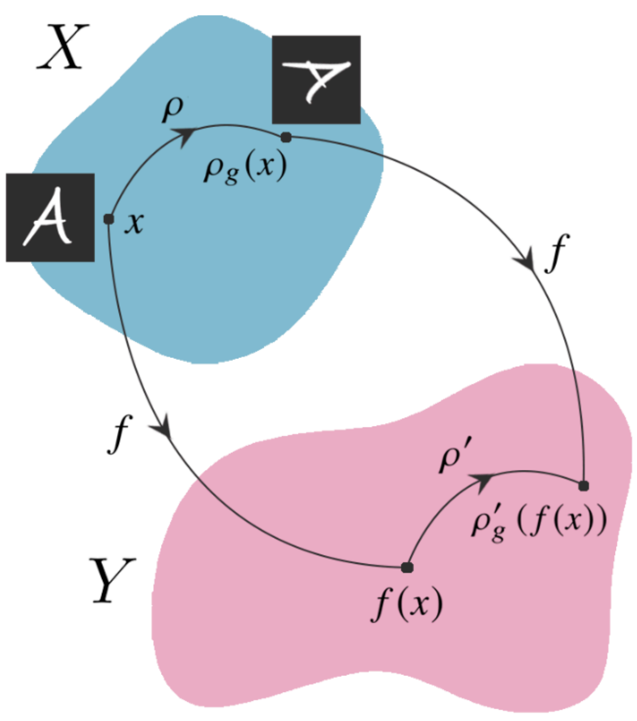
Physics-informed ML

- Embedding symmetries, e.g. Lorentz group symmetry, leads to improved efficiency

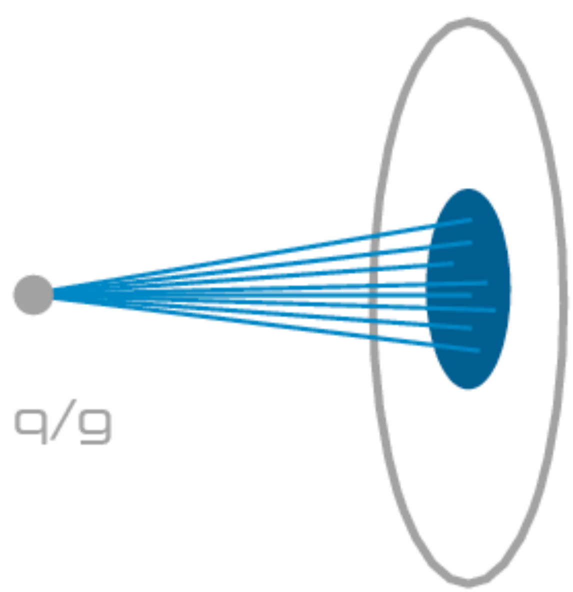
- **Exemplary application to jet tagging:**
  - Jets are spray of hadrons initiated by a fundamental particle of some kind
  - These hadrons get clustered into one object called "a jet"
  - The jet can have different properties depending on the mother particle
  - **Jet identification ("tagging") = who was the mother particle?**

Equivariance

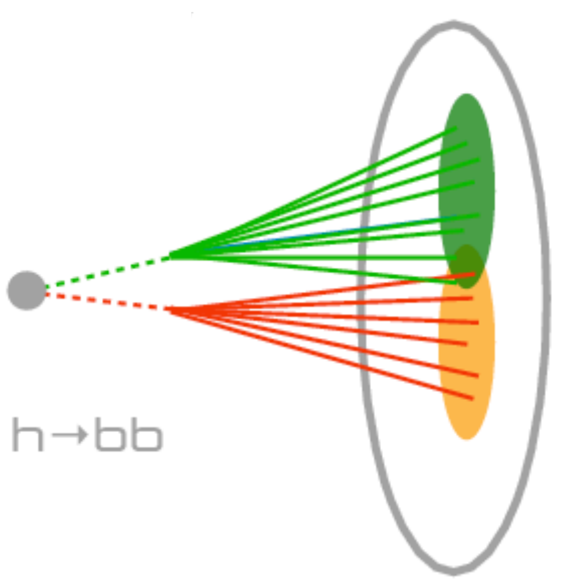
$$f(\rho_g(x)) = \rho'_g(f(x))$$



BACKGROUND JET
(single quark/gluon)

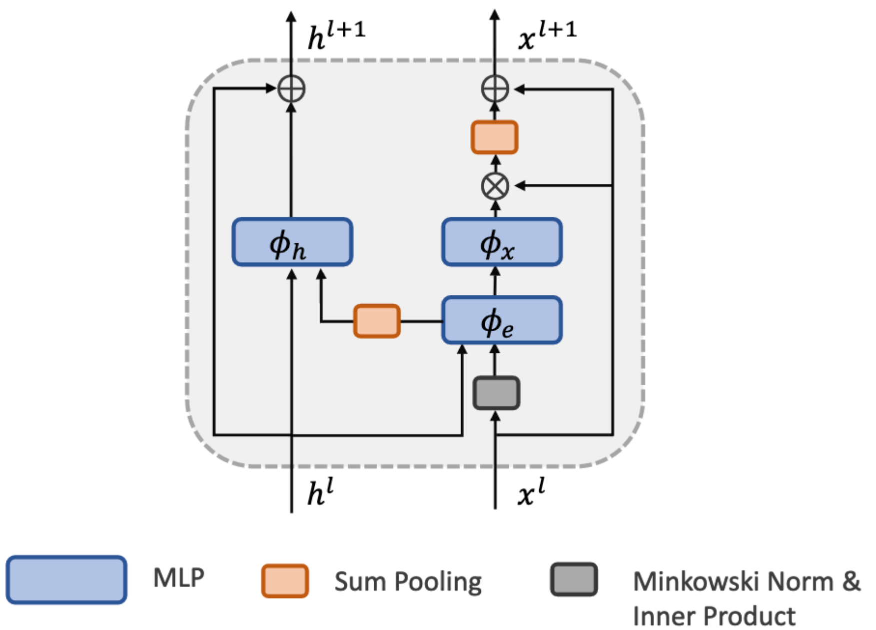


SIGNAL JET
(ex, Higgs boson to bottom quarks)



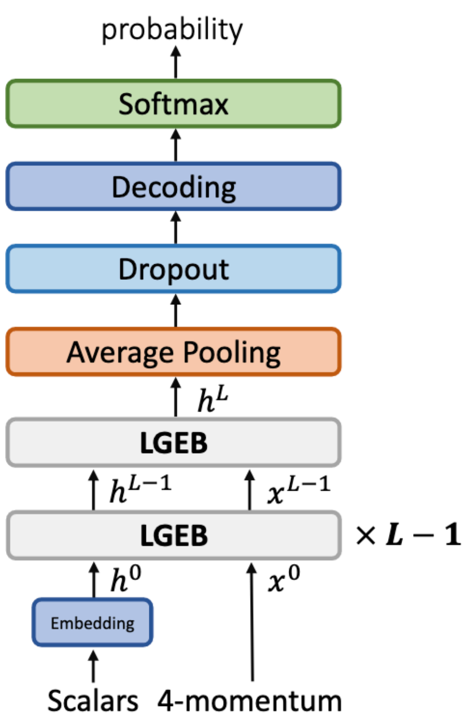
Physics-informed ML

- Embedding symmetries, e.g. Lorentz group symmetry, leads to improved efficiency
- **Exemplary application to jet tagging:** the jet tagging result should not depend on the spatial orientation of a jet → better generalization!
- Achieved symmetry through Minkowski dot product attention
- **Training efficient and reduced number of parameters!**



Lorentz Group Equivariant Block (LGEB)

Training Fraction	Model	Accuracy	AUC	$1/\varepsilon_B$ ($\varepsilon_S = 0.5$)	$1/\varepsilon_B$ ($\varepsilon_S = 0.3$)
0.5%	ParticleNet	0.913	0.9687	77 ± 4	199 ± 14
	LorentzNet	**0.929**	**0.9793**	$\mathbf{176 \pm 14}$	$\mathbf{562 \pm 72}$
1%	ParticleNet	0.919	0.9734	103 ± 5	287 ± 19
	LorentzNet	**0.932**	**0.9812**	$\mathbf{209 \pm 5}$	$\mathbf{697 \pm 58}$
5%	ParticleNet	0.931	0.9807	195 ± 4	609 ± 35
	LorentzNet	**0.937**	**0.9839**	$\mathbf{293 \pm 12}$	$\mathbf{1108 \pm 84}$



LorentzNet

JHEP 07, 30 (2022)

Physics-informed ML

- Embedding symmetries, e.g. Lorentz group symmetry, leads to improved efficiency

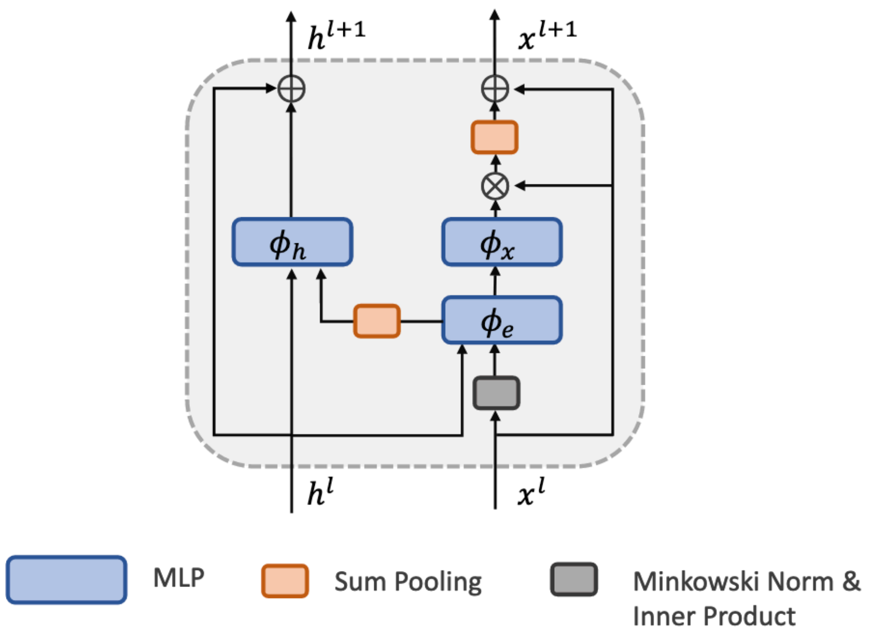
- **Exemplary application to jet tagging:** the jet tagging result should not depend on the spatial orientation of a jet → better generalization!

- Achieved symmetry through Minkowski dot product attention

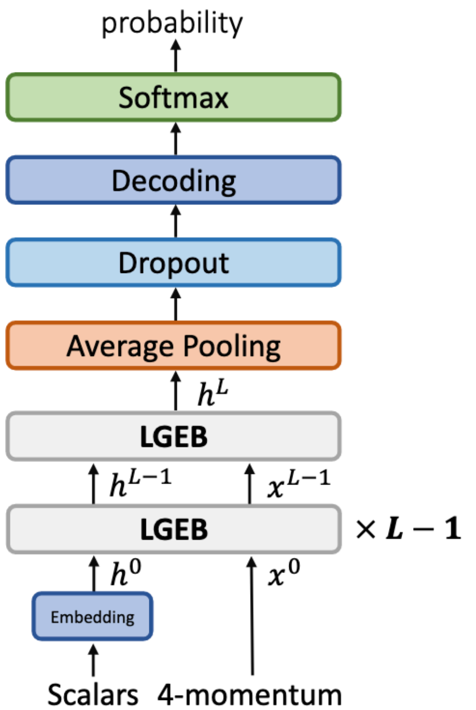
- **Training efficient and reduced number of parameters!**

  - does not necessarily translate in faster inference speed… the key is understanding trade off!

Model	Equivariance	Time on CPU (ms/batch)	Time on GPU (ms/batch)	#Params
ResNeXt	✗	5.5	0.34	1.46M
P-CNN	✗	**0.6**	**0.11**	348k
PFN	✗	**0.6**	0.12	82k
ParticleNet	✗	11.0	0.19	366k
EGNN	E(4)	30.0	0.30	222k
LGN	$SO^+(1,3)$	51.4	1.66	4.5k
LorentzNet	$SO^+(1,3)$	32.9	0.34	224k



Lorentz Group Equivariant Block (LGEB)



LorentzNet

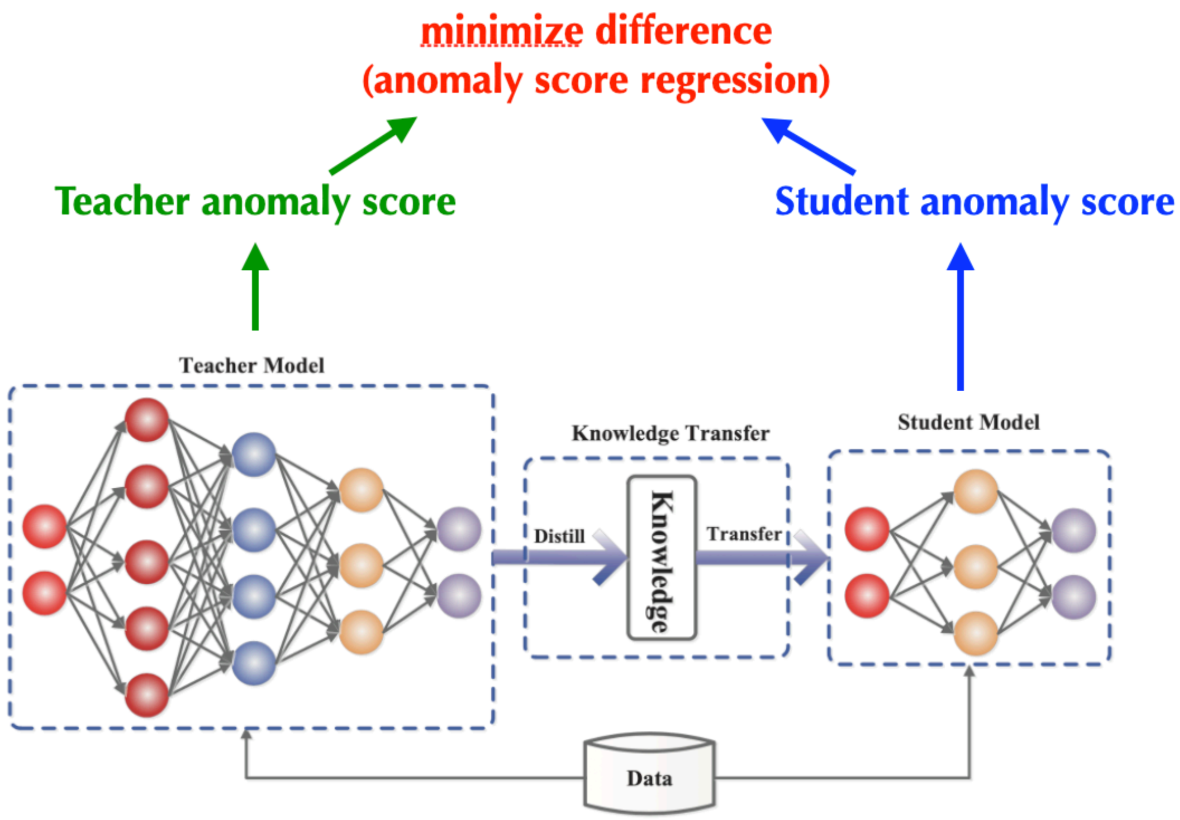
JHEP 07, 30 (2022)

Knowledge distillation to the rescue!

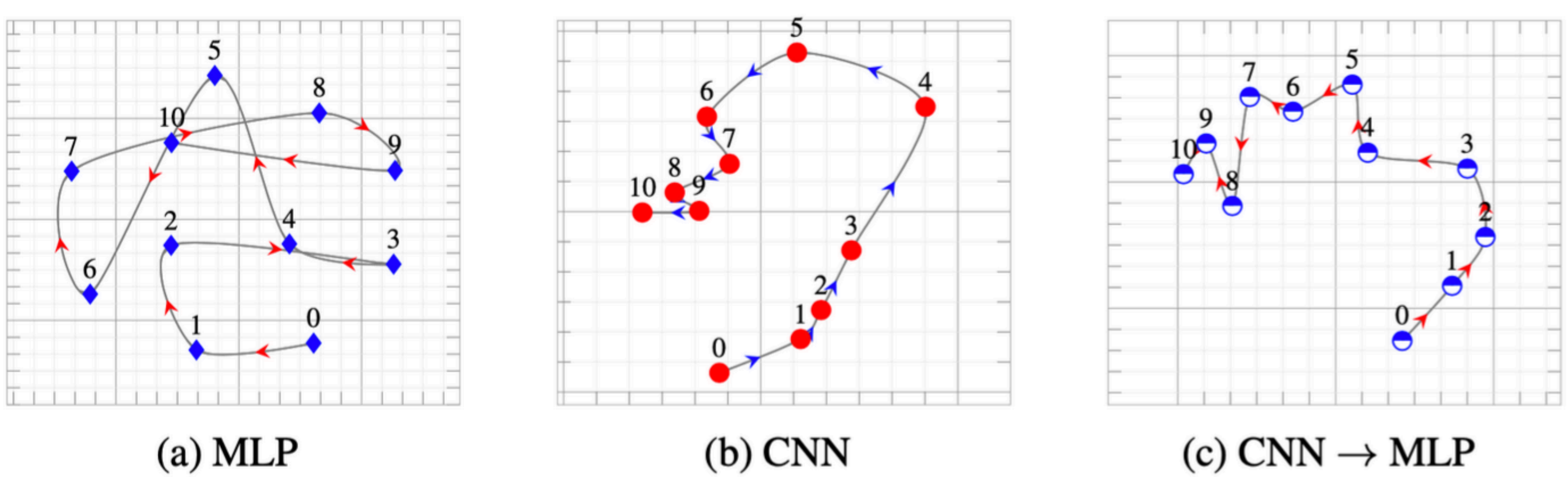
- The process of **transferring knowledge from a teacher model to a student model**, where the logits from the teacher are used to train the student

- The **student could be more computationally efficient** while taking advantage of the huge number of parameters during training!

- Through distillation, the **generalization** behaviour of the teacher that is affected by its inductive biases also transfers to the student model



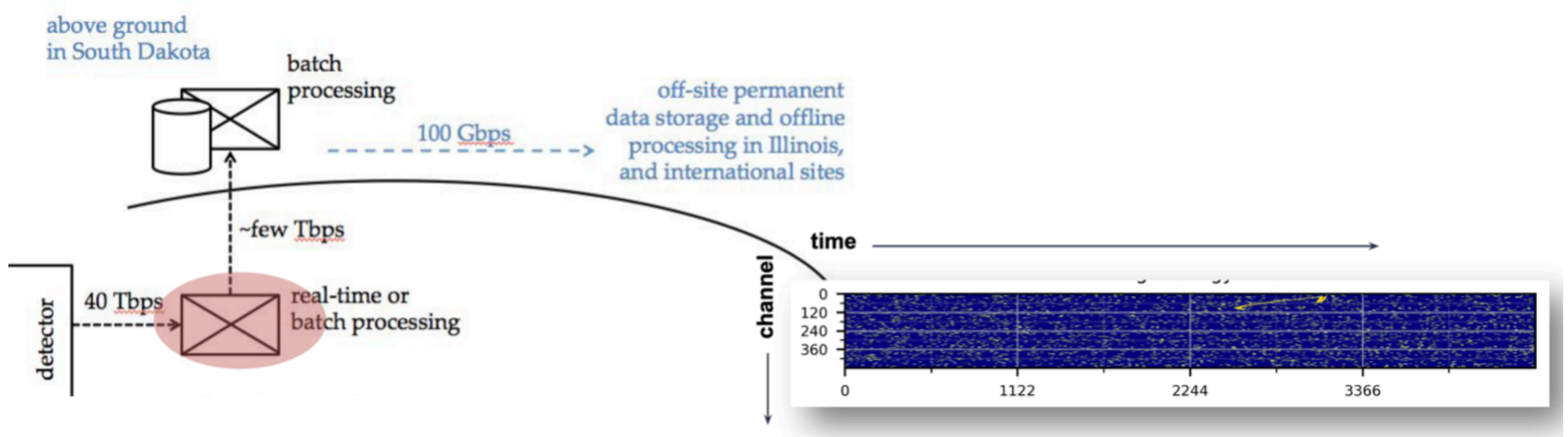
Not explored in HEP so far!

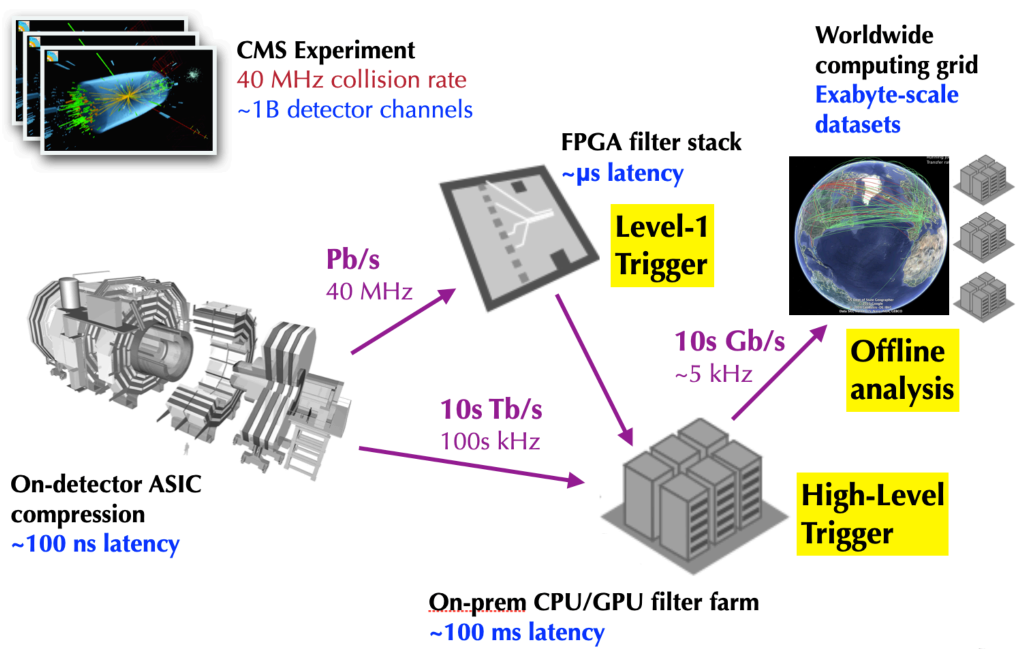


arXiv.2006.00555

Bring it to the hardware!

- Not trivial… given latency and resource constraints cannot simply reuse industry tools to port ML to hardware (FPGAs, GPUs, IPUs, …)
  - mostly optimized for standard needs and hardly customisable for low-latency, low-resources and/or sparse graph computations as needed in HEP



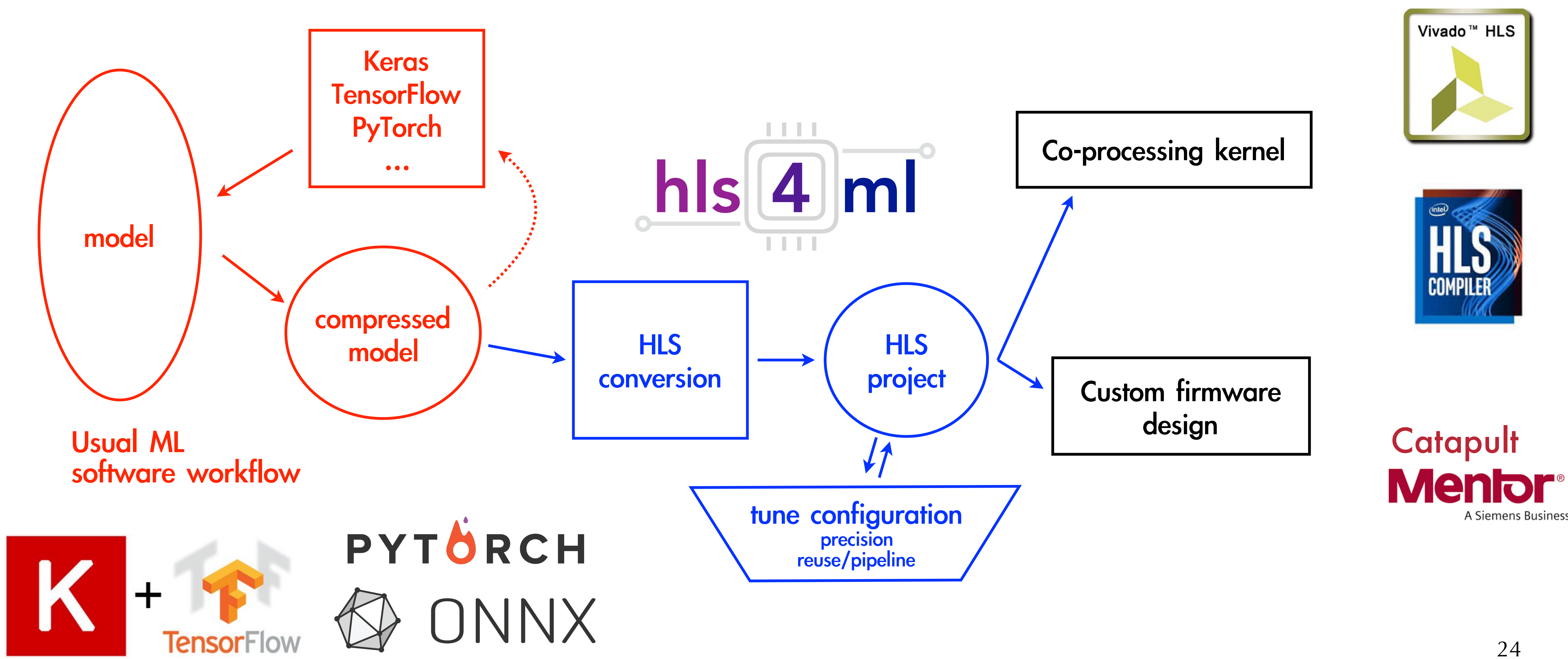


Bring DL to FPGA for real-time ML

high level synthesis for machine learning

A user-friendly, open-source tool to develop and optimize FPGA firmware for ML inference

- Input models trained with standard ML libraries like (Q)Keras, PyTorch, (Q)ONNX
- Python package for conversion, configuration and optimization
- Uses HLS software: rapid design space exploration + rapid feature development
- Comes with implementation of common ingredients - layer types, activation functions
- And novel ingredients for fast, efficient inference - low-precision NNs, network optimisations

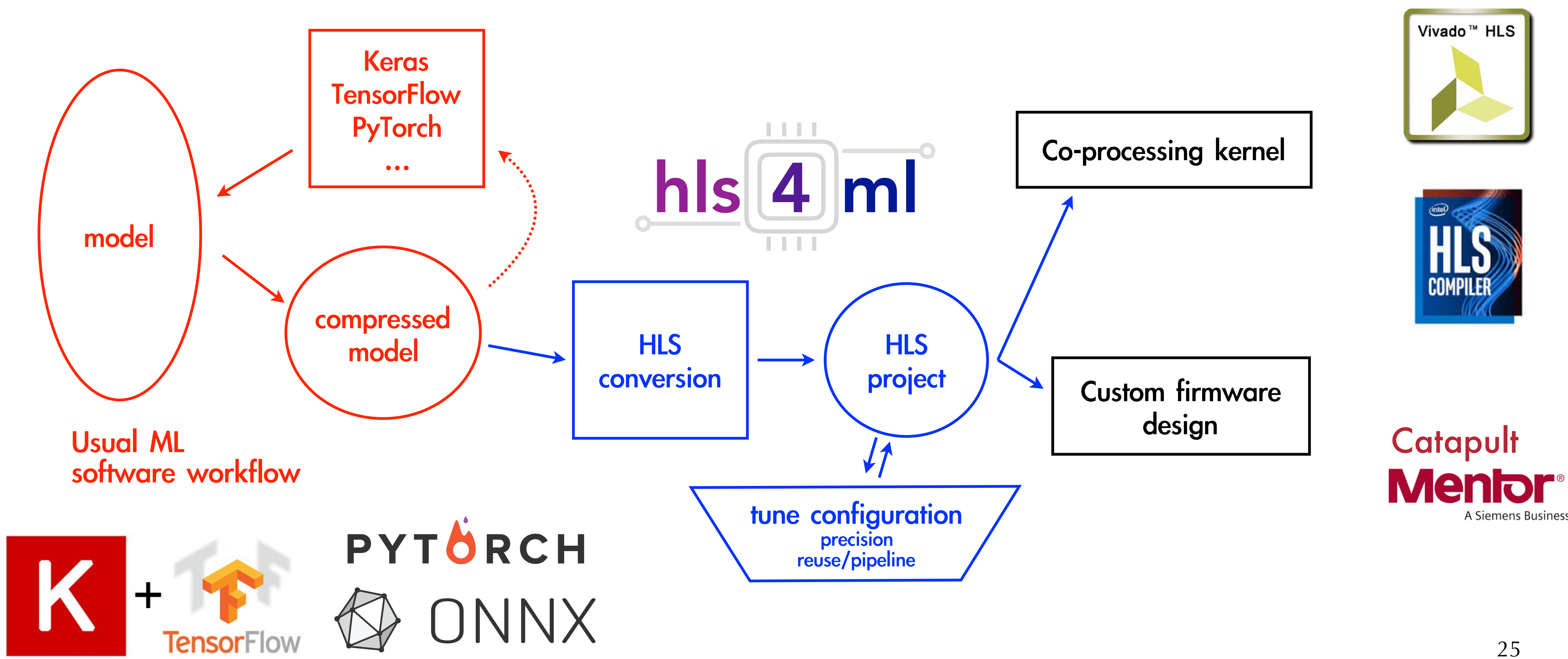


Bring DL to FPGA for real-time ML

high level synthesis for machine learning

A codesign tool to build algorithms with hardware in mind and providing efficient platforms for programming the hardware.

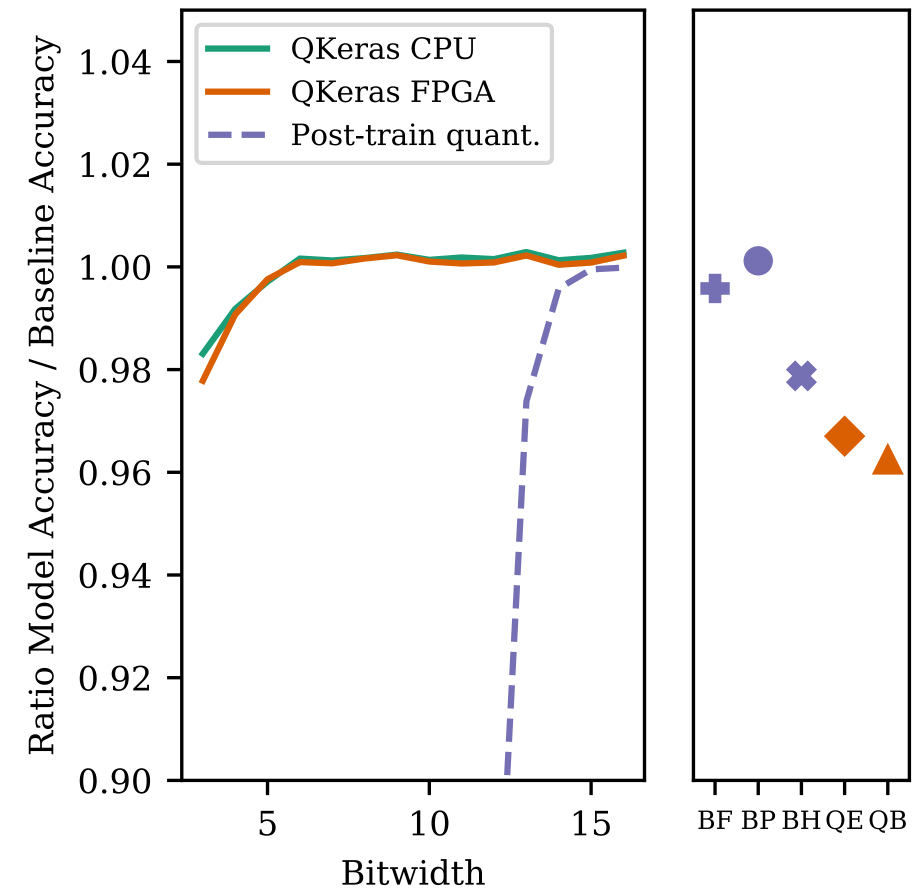
Many use cases in HEP and beyond… and still growing!
(see Fast Machine Learning For Science Workshop last month)

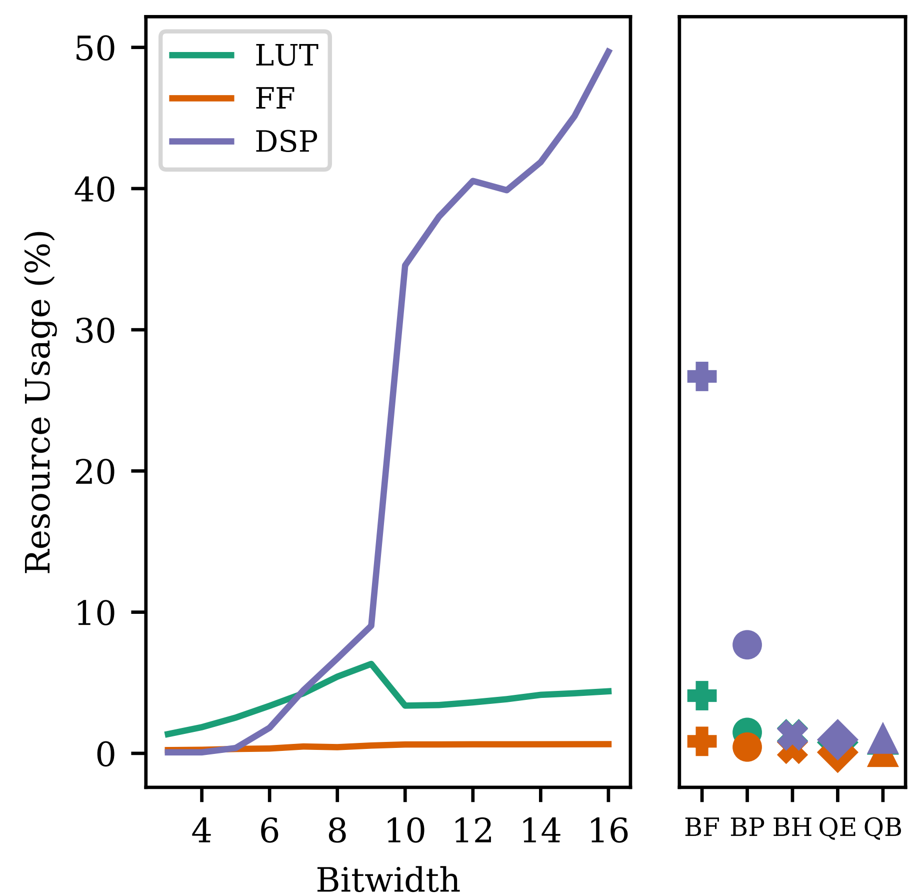


Quantization-aware training

More in Thea's talk!

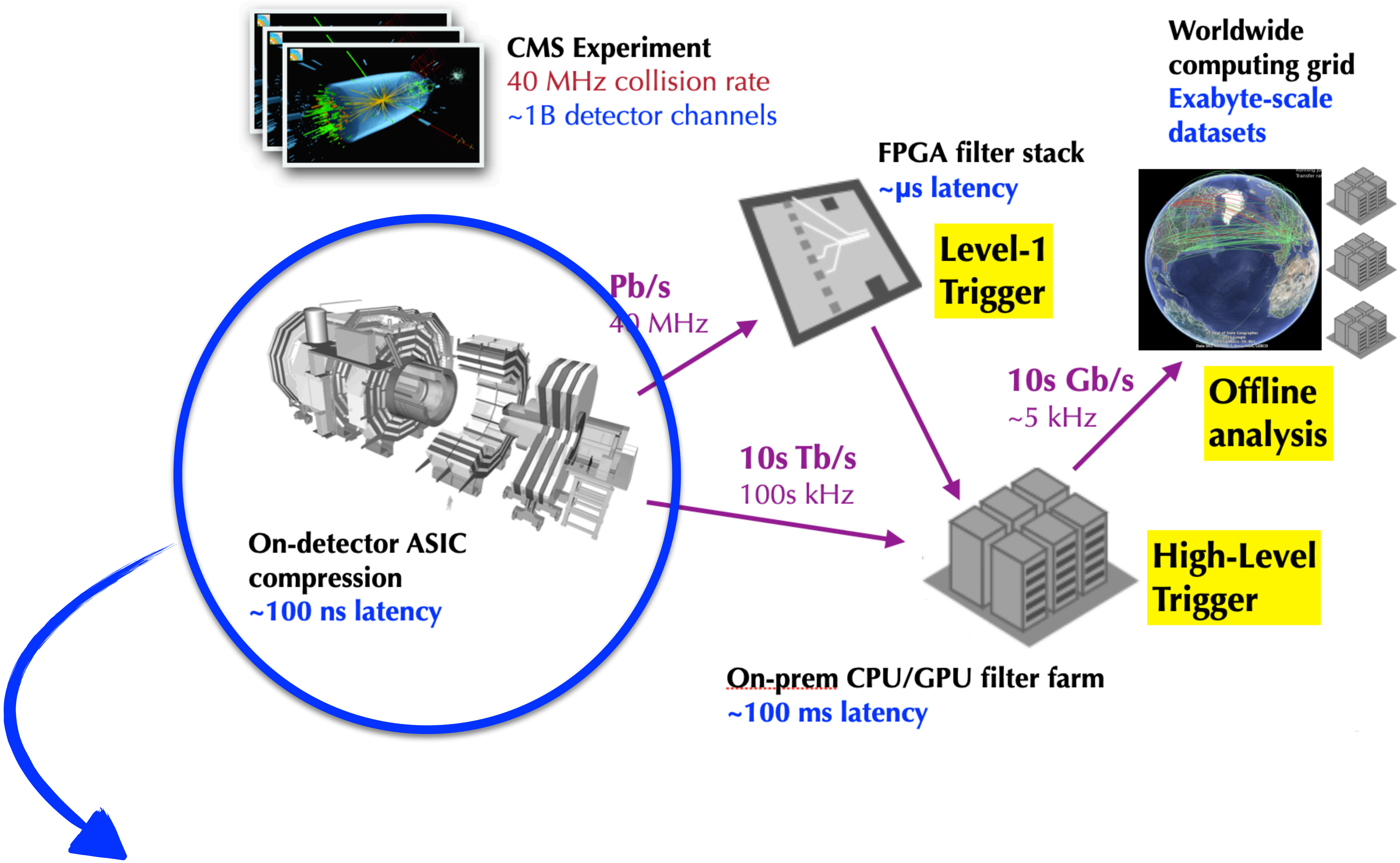
- Efficient hardware implementation uses reduced precision wrt floating point
- Post-training quantization can affect accuracy
  - for a given bit allocation, the loss minimum at floating-point precision might not be the minimum anymore
- One could specify quantization while look for the minimum
  - maximize accuracy for minimal FPGA resources
- Workflow: quantization-aware training with [Google QKeras]() and firmware design with [hls4ml]() for best NN inference on FPGA performance





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

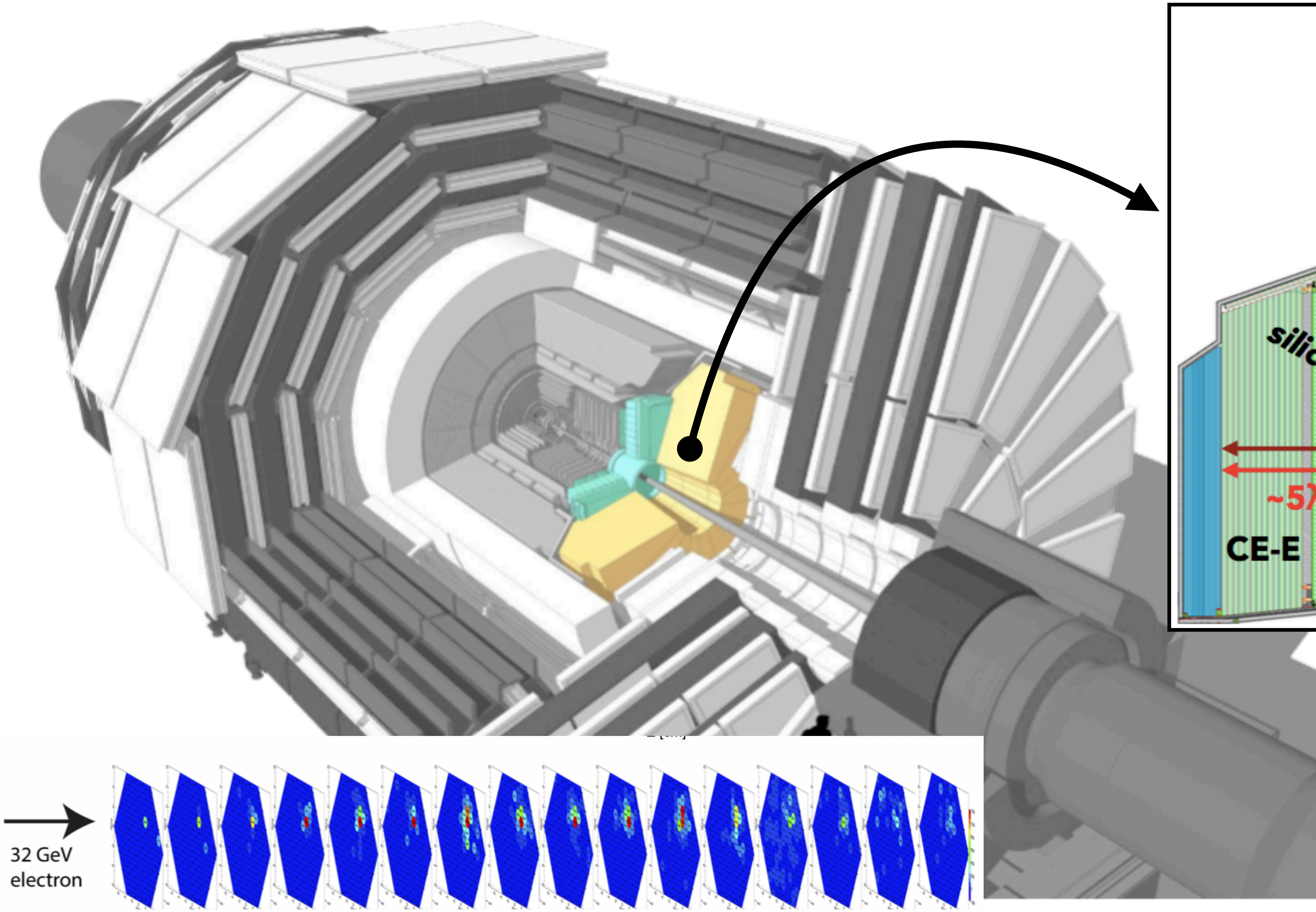
From fast to ultra fast ML

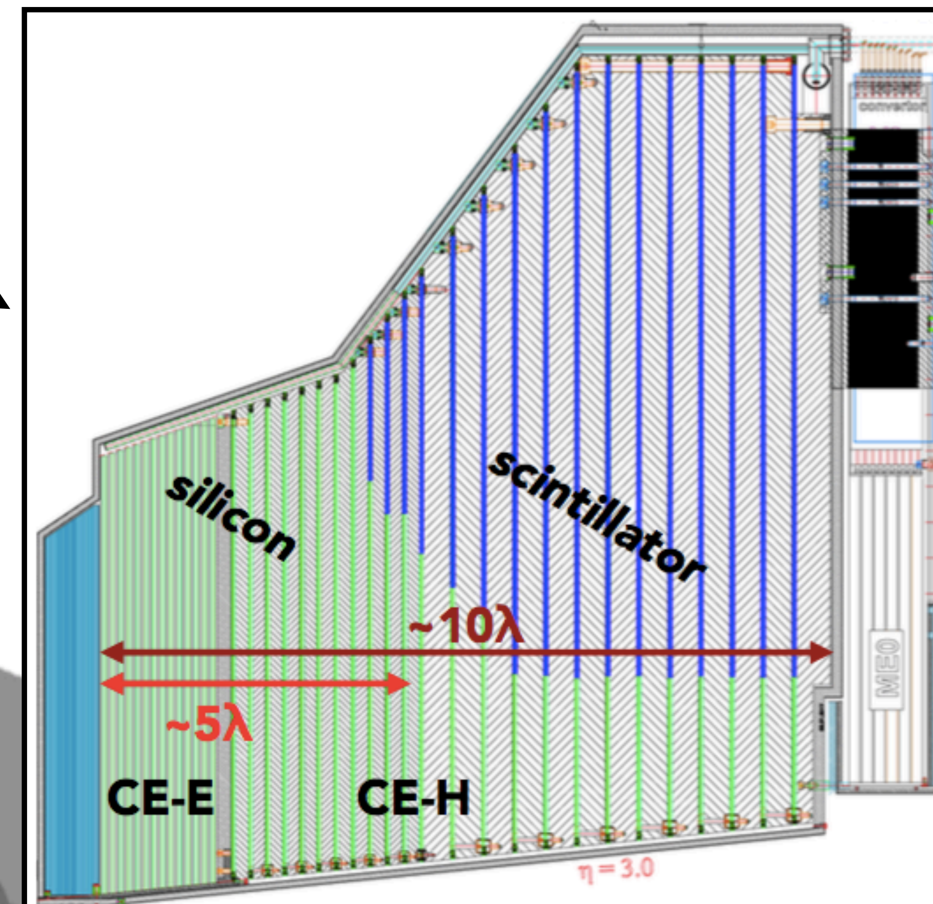


ASICs typically used at the front end for sensors read out: directly embed ML in here to allow intelligent data compression at the very edge

Example:
High-granularity calorimeter @ HL-LHC

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

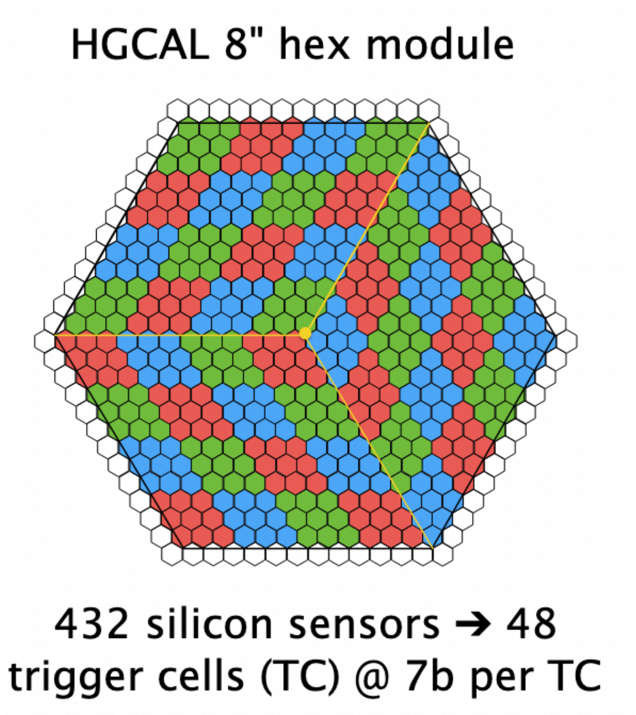




CMS HGCAL TDR

Example: CMS HG calorimeter

Input



(336b in total)

Output:

"Super trigger cell" algo

3[16 TC sum] x 16–48 bits

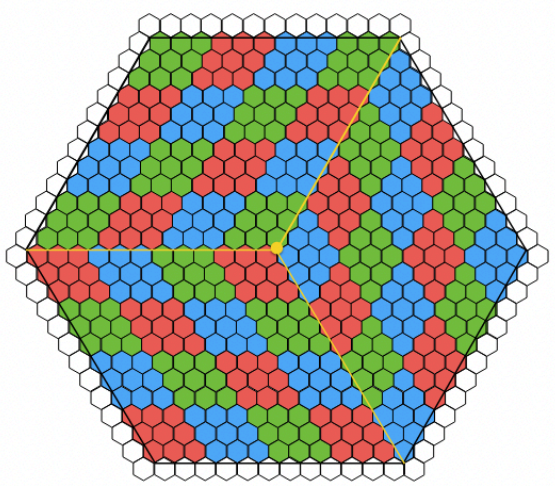
= 48—144 bits

(depending on the position)

Example: CMS HG calorimeter

Input

HGCAL 8" hex module



432 silicon sensors ➔ 48 trigger cells (TC) @ 7b per TC

(336b in total)

→ ASIC →

Output:

"Super trigger cell" algo

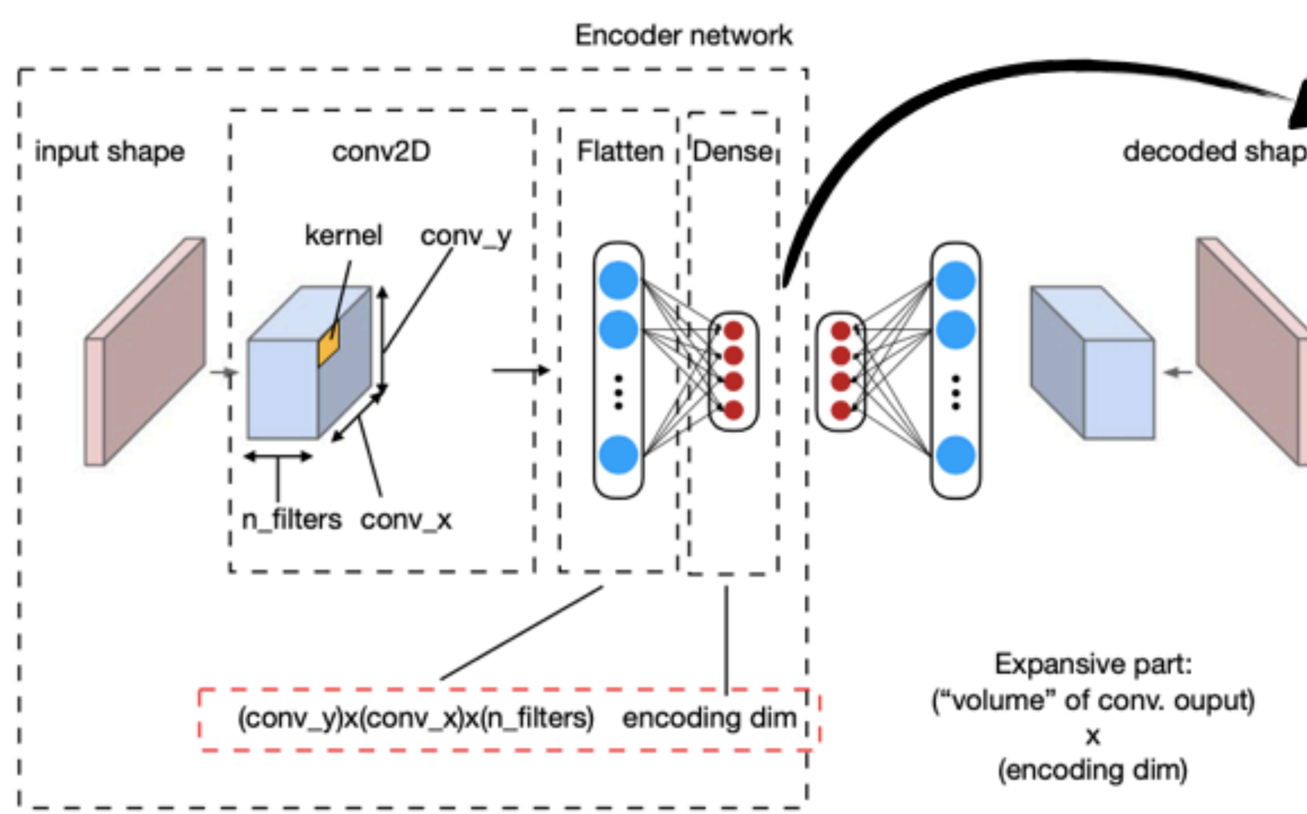
3[16 TC sum] x 16–48 bits

= 48—144 bits

(depending on the position)

Can we do a better job of encoding the info in those bits w/o so much loss in granularity?

Encoder on ASIC ⟷ **Decoder on L1 board**



Really need quantized training here to optimize information encoding

Use QKeras!

Example: CMS HG calorimeter

- Evaluate AutoEncoder performance according to image similarity

- Energy Mover's distance: quantify the cost of transforming one image into another as energy x distance (lower EMD better performance)

- Use of more outputs at lower precision outperform their counterpart

- **Use hls4ml for mapping the ML model onto reconfigurable logic:**

  - extended for the ML-to-ASIC flow to support Mentor's Catapult HLS and target the specific 65 nm LP CMOS technology

- **Downstream performance driven by physics to be fully assessed with codesign tools allowing for fast feedback loop!**

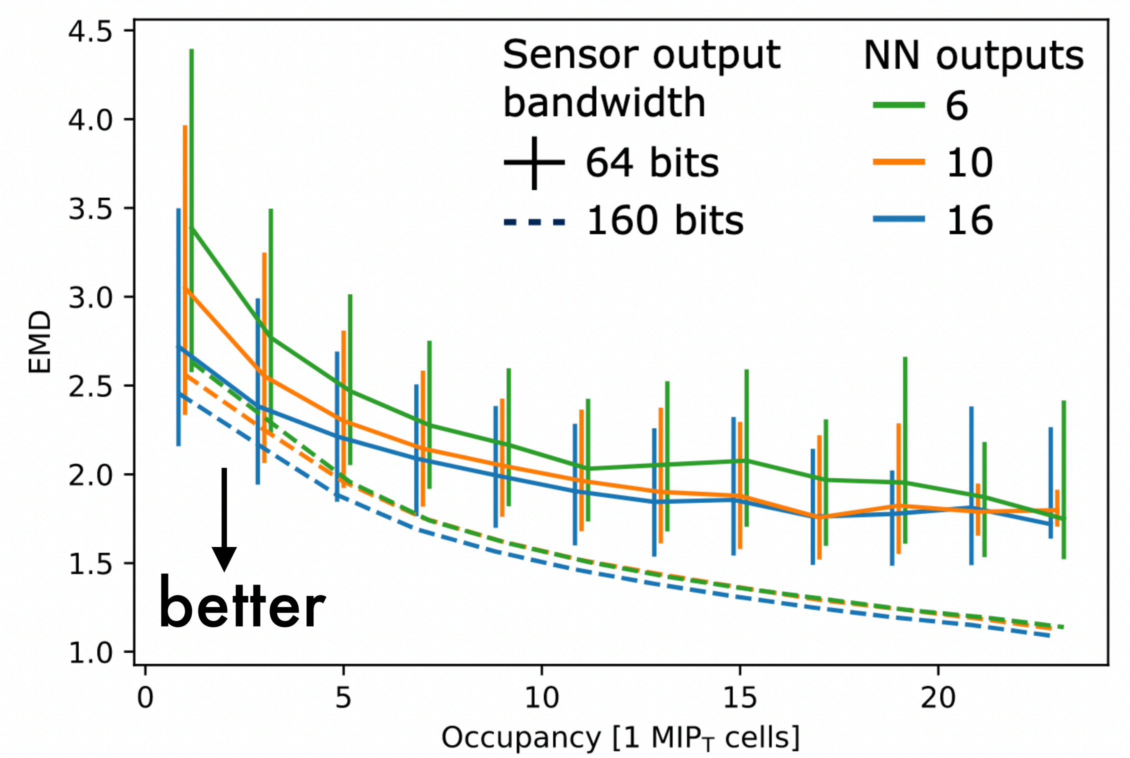
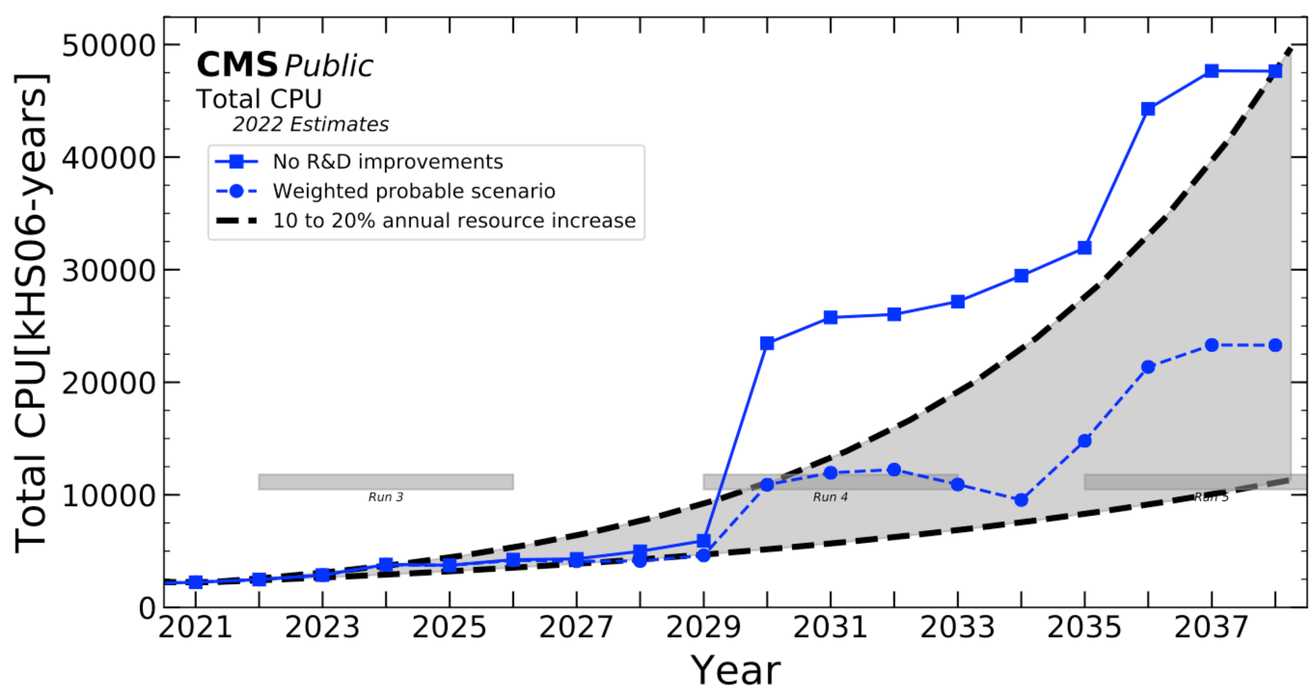


TABLE III
KEY SIMULATION PERFORMANCE PARAMETERS OF THE DESIGN.

Latency	Energy/inference	Power	Area
50 ns	2.38 nJ/inf.	95 mW	3.6 mm^2

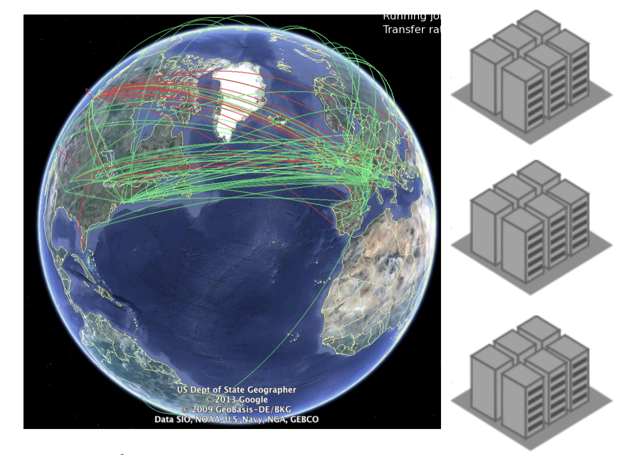
ML for high throughput

- HLT and offline: typically relaxed or no latency constraints but **high throughput** is required
  - current algorithms, workflows, and computing infrastructure do not scale



Source

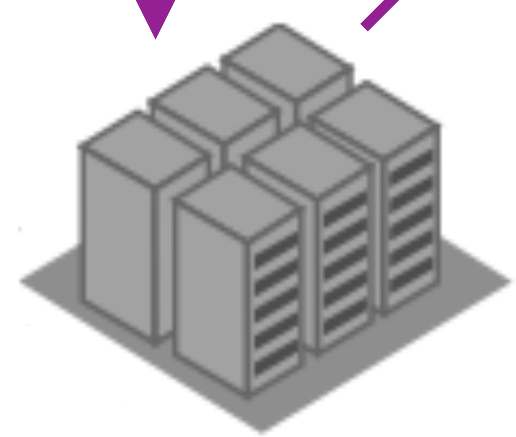
Worldwide computing grid
Exabyte-scale datasets



Input:
10s Tb/s
100s kHz

10s Gb/s
~5 kHz

Offline analysis



High-Level Trigger

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

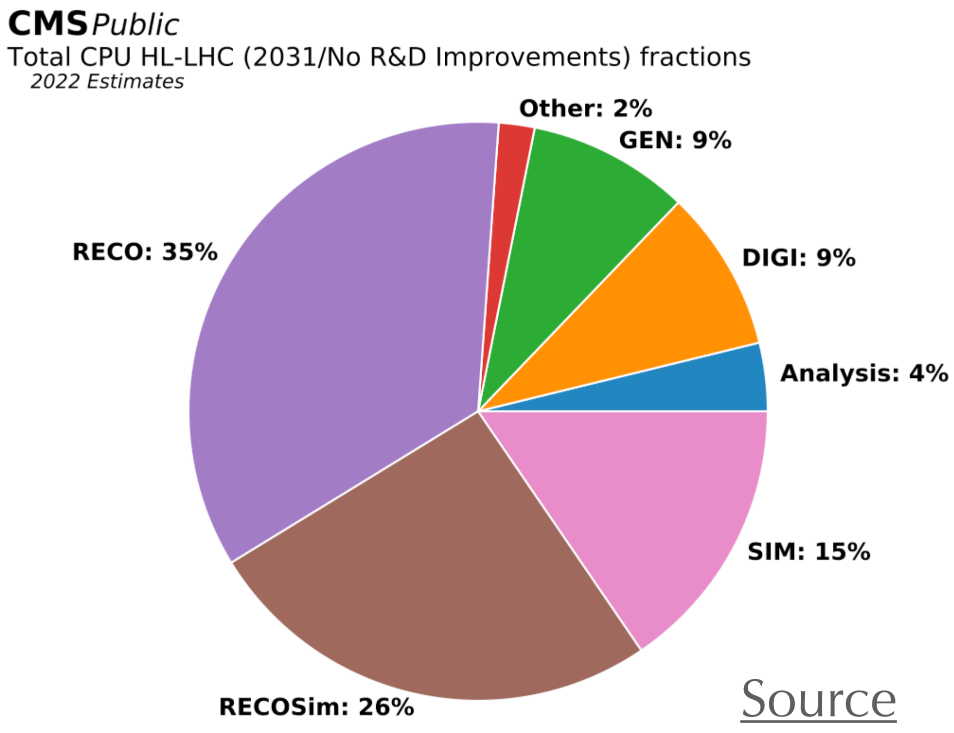
ML for high throughput

- HEP experiments rely heavily on simulations from experimental design all the way to data analysis

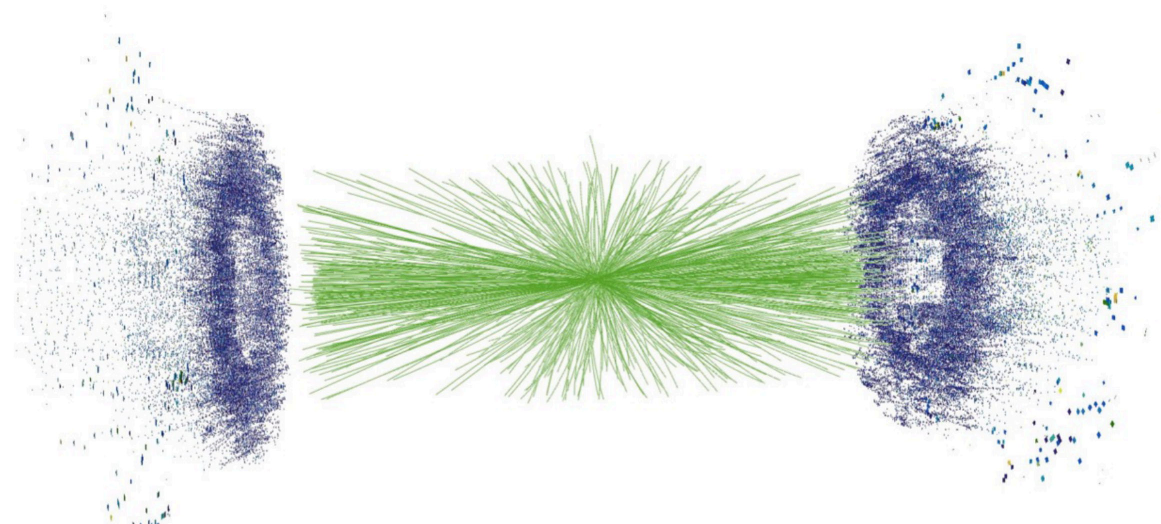
- **Detector simulation** (GEANT4) and **event generation** (MG5, Pythia, Herwig, …) are major and growing bottlenecks at LHC and other experiments

- **Event reconstruction** for both MC events and real data also computing intensive
  - ex, for track reconstruction CPU time can scale quadratically with number of particles in today's detectors

- **Effort to accelerate this workflow with ML** through end-to-end approach or by replacing single steps
  - generative models for MC simulation with calorimeter images or point cloud representation
  - for reconstruction (ex, tracking) GNNs is most promising approach



Source

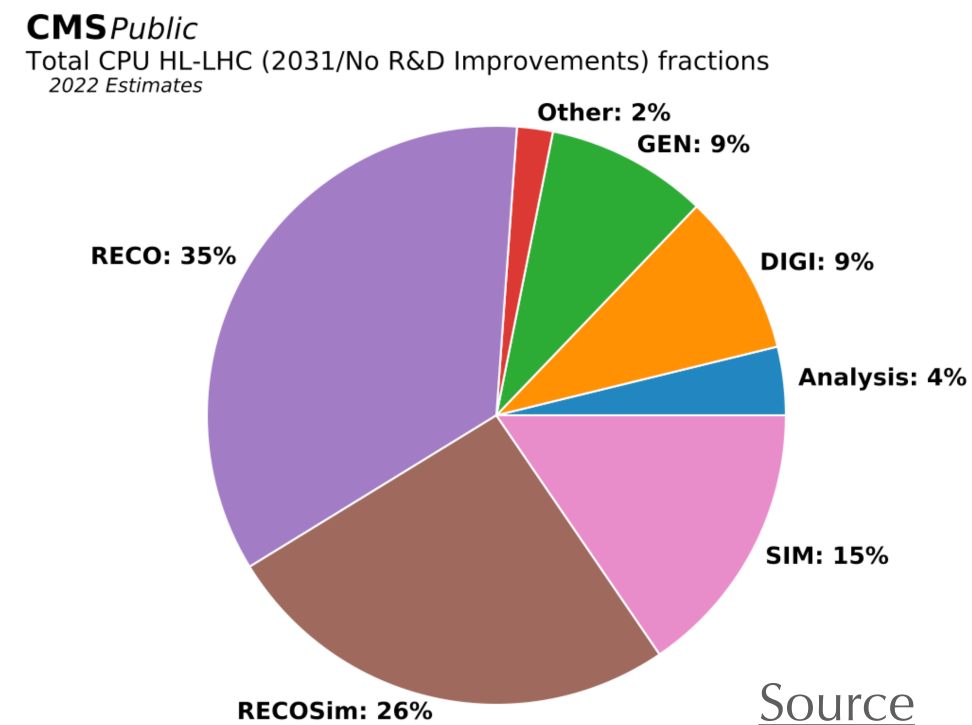


$t\bar{t}$ event with pileup 200

ML for high throughput

Effort to accelerate this workflow with ML

- Improve physics performance
- Minimize need to learn new processor-specific code
  → decrease effort, increase maintainability
- Must exploit **heterogeneous architectures** to achieve highest throughput
  → requires new computing paradigm and execution in experimental framework



Source

GEANT4 → 10^{10} events **SLOW but ACCURATE**

GEANT4 → 10^{5} events → Surrogate model → 10^{10} events

(GAN, VAE, Normalizing Flow, …)
Learn underlying distribution of GEANT4 events

FAST and ACCURATE?

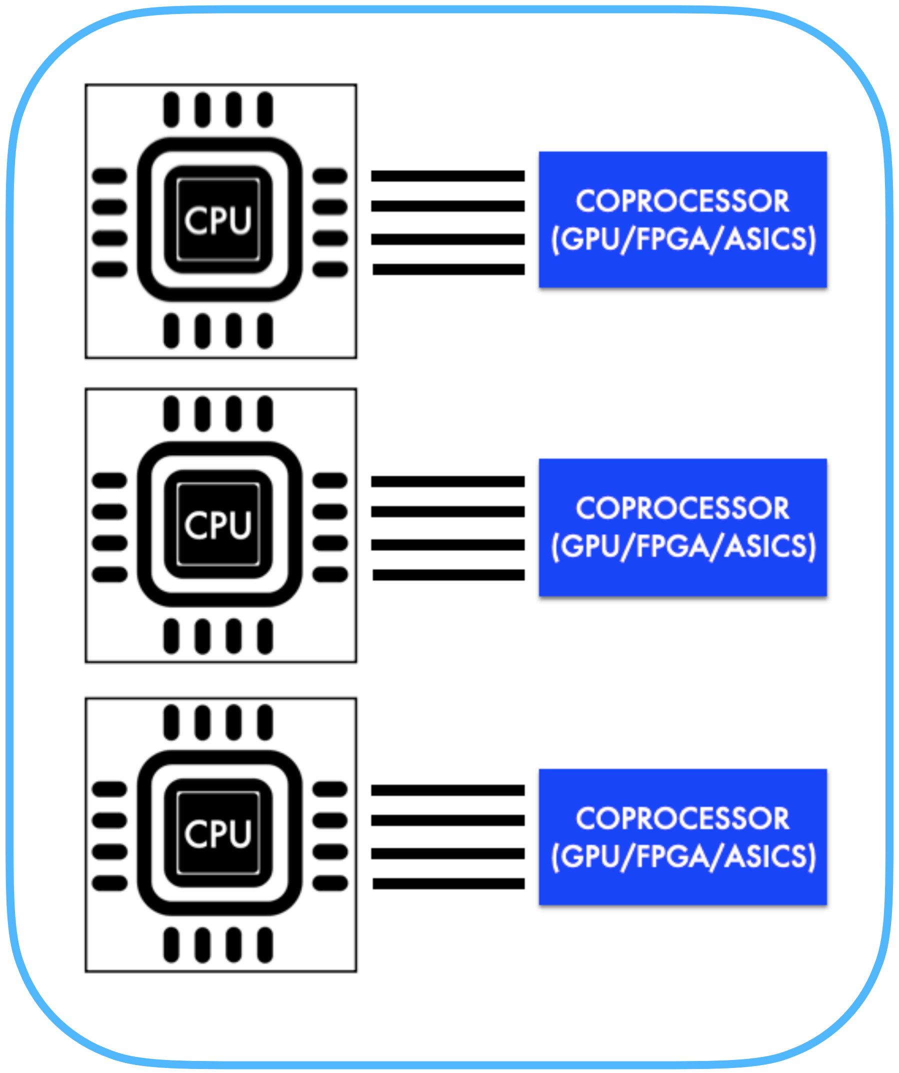
ML methods can provide fast and accurate “surrogate models” for GEANT4 etc

from D. Shih at Snowmass 2021 (Seattle)

See also plenary talks at ACAT2022: generative models, summary

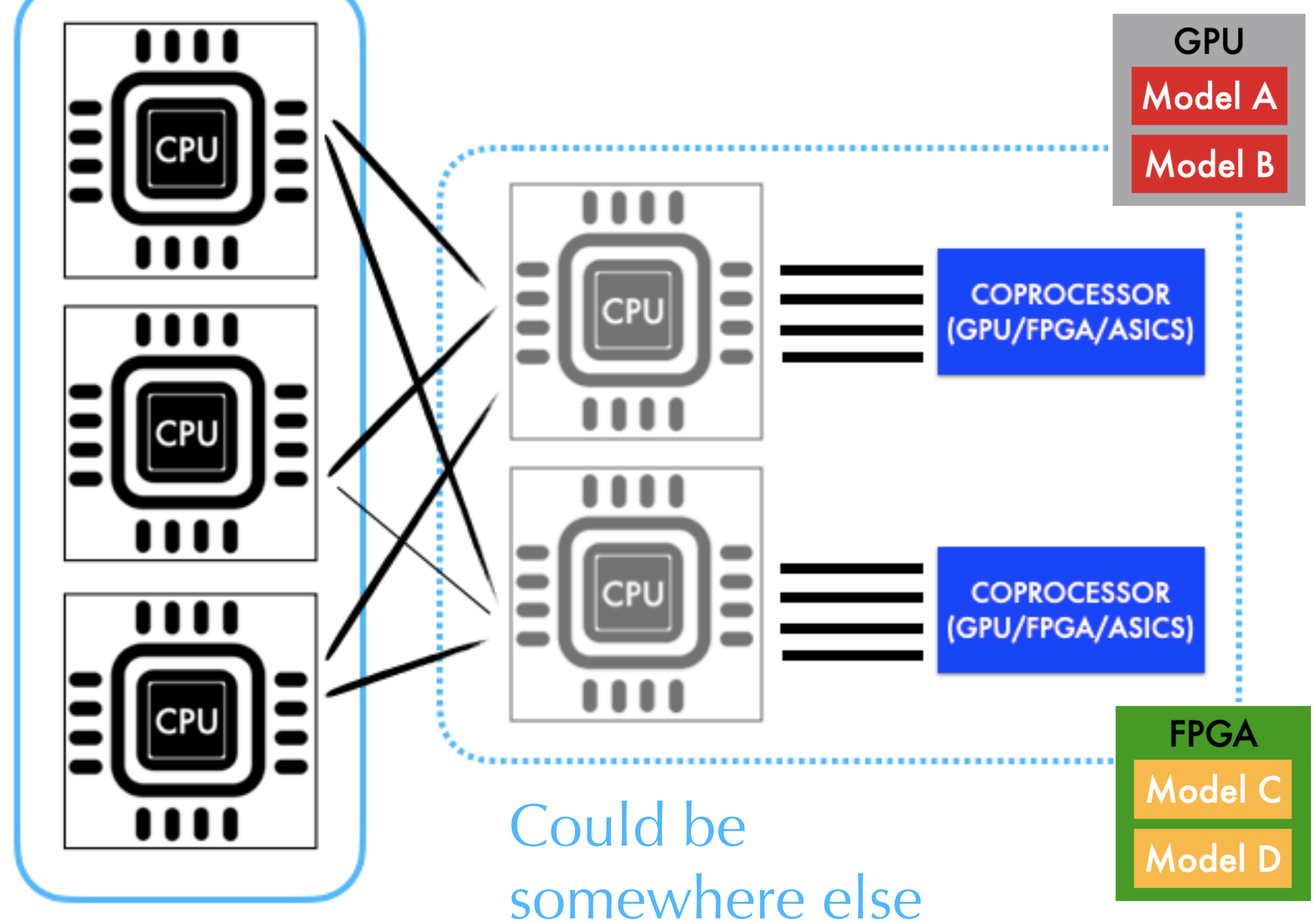
Heterogenous computing @ LHC

Option 1: direct



Data center/
experimental site

Option 2: as a service

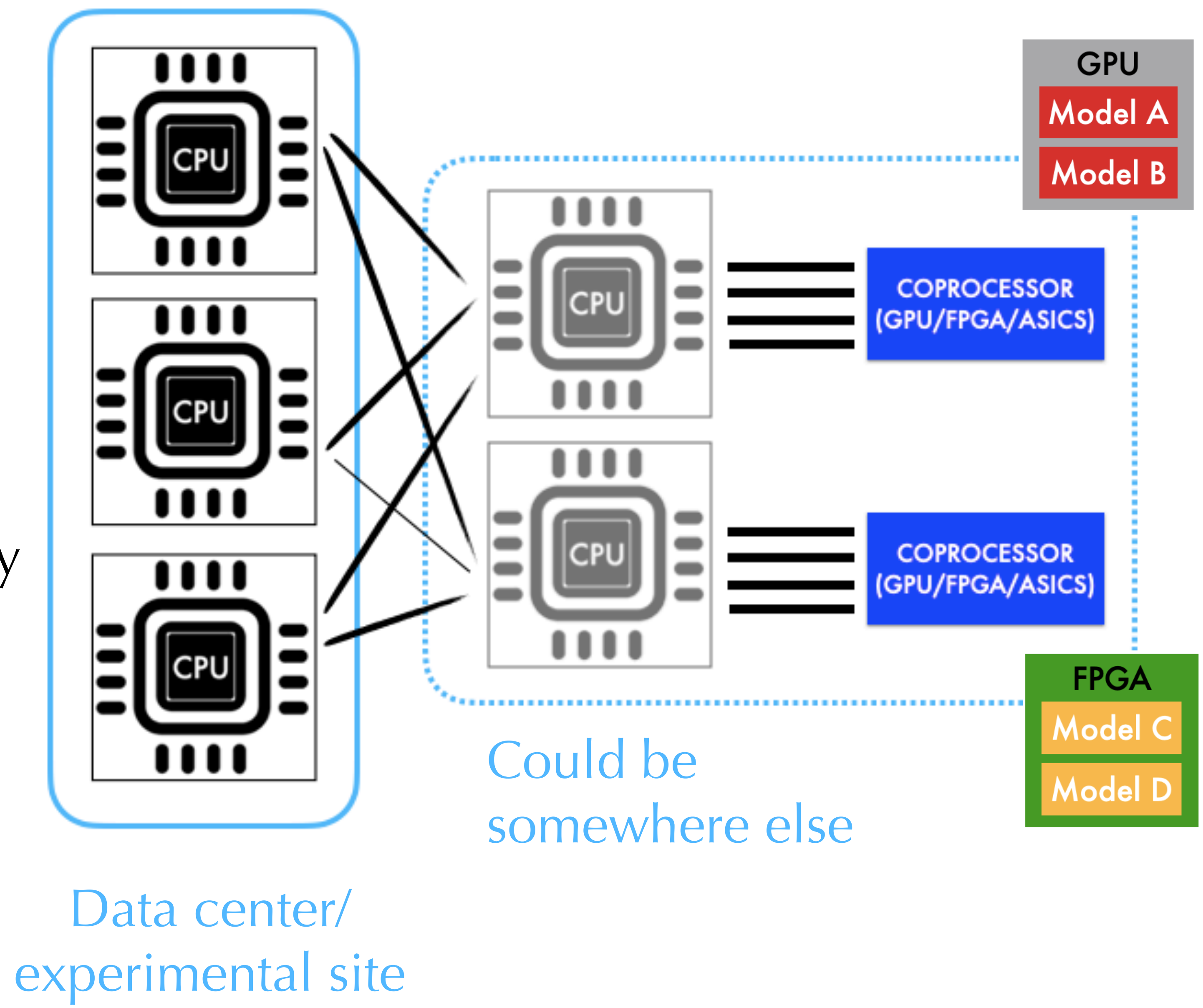


Data center/
experimental site

Heterogenous computing @ LHC

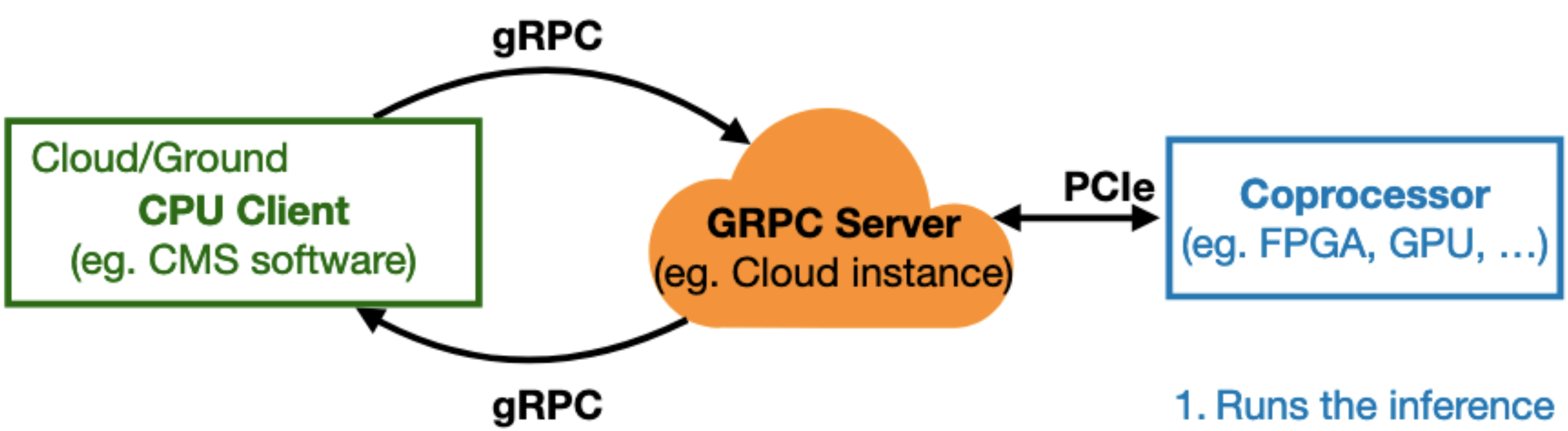
Option 2: as a service

- One coprocessor can serve many CPUs → reduce cost and increase scalability
- Increase heterogeneity: choose best device for each job
- Deploy GPUs, FPGAs, …simultaneously
- Model optimization for the processor could be obtained with available tools (ex, Intel oneAPI [*])



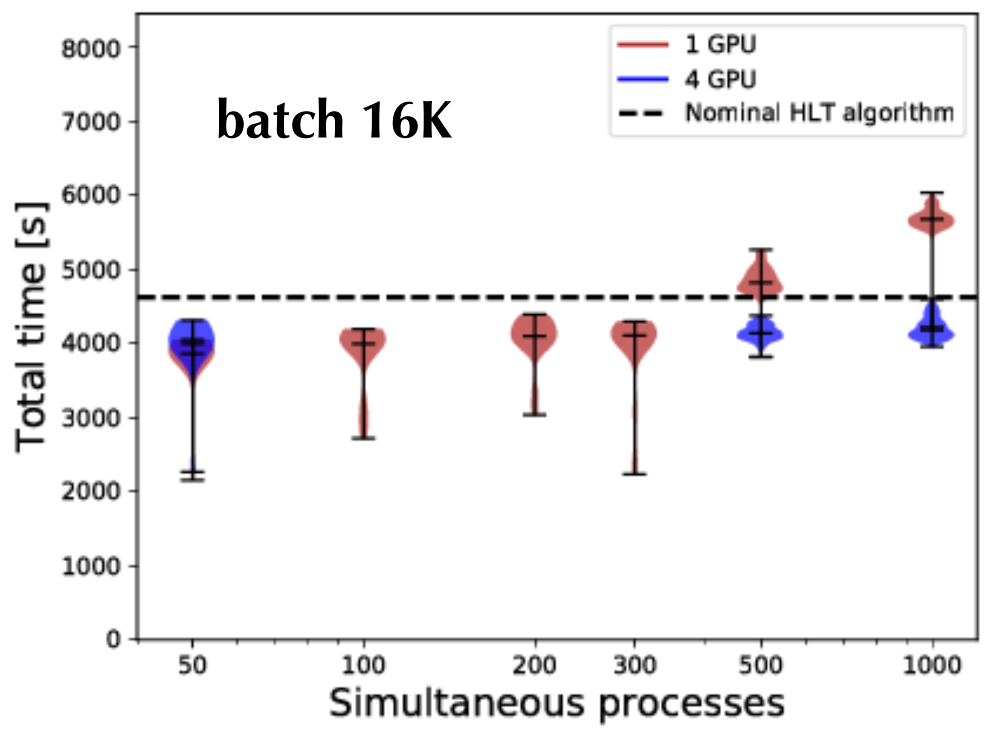
MLaaS with Sonic

- **Services for Optimized Network Inference on Coprocessors** (SONIC) enables inference as a service in experiment software frameworks
  - experiment software (C++) only has to handle converting inputs and outputs between event data format and inference server format
- Uses industry tools as gRPC communication and Nvidia Triton inference servers
- Interacts with cloud services: Azure, AWS, GCP

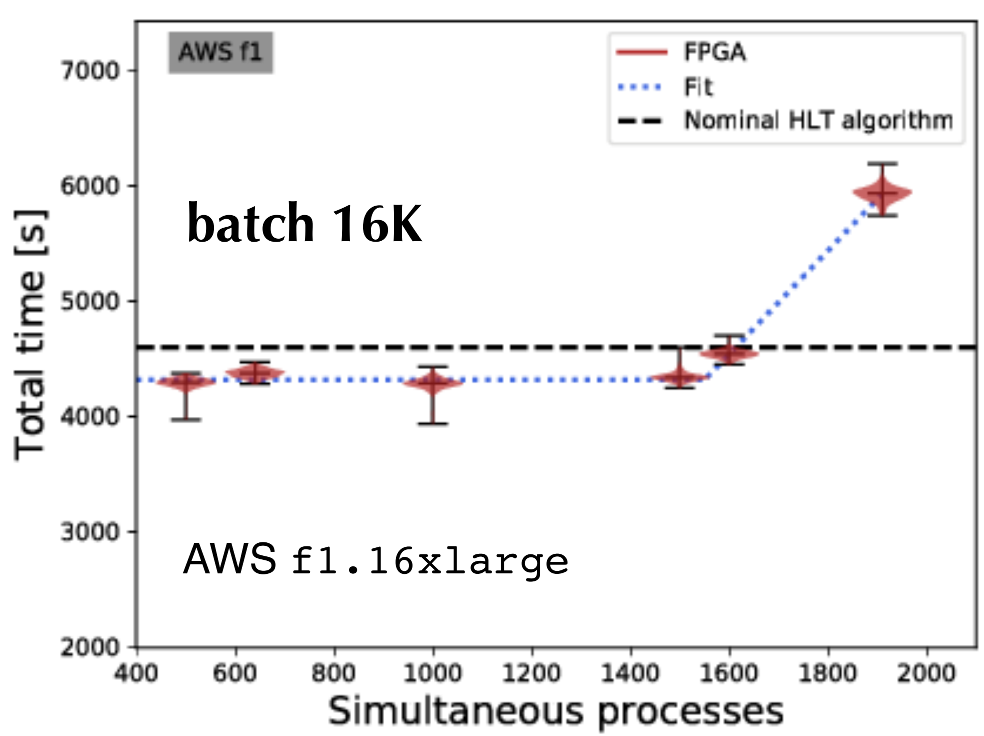


MLaaS with Sonic

Replace hadronic calorimeter reconstruction with ML (2k parameters dense NN here) and enable the model inference in the CMS software with SONIC



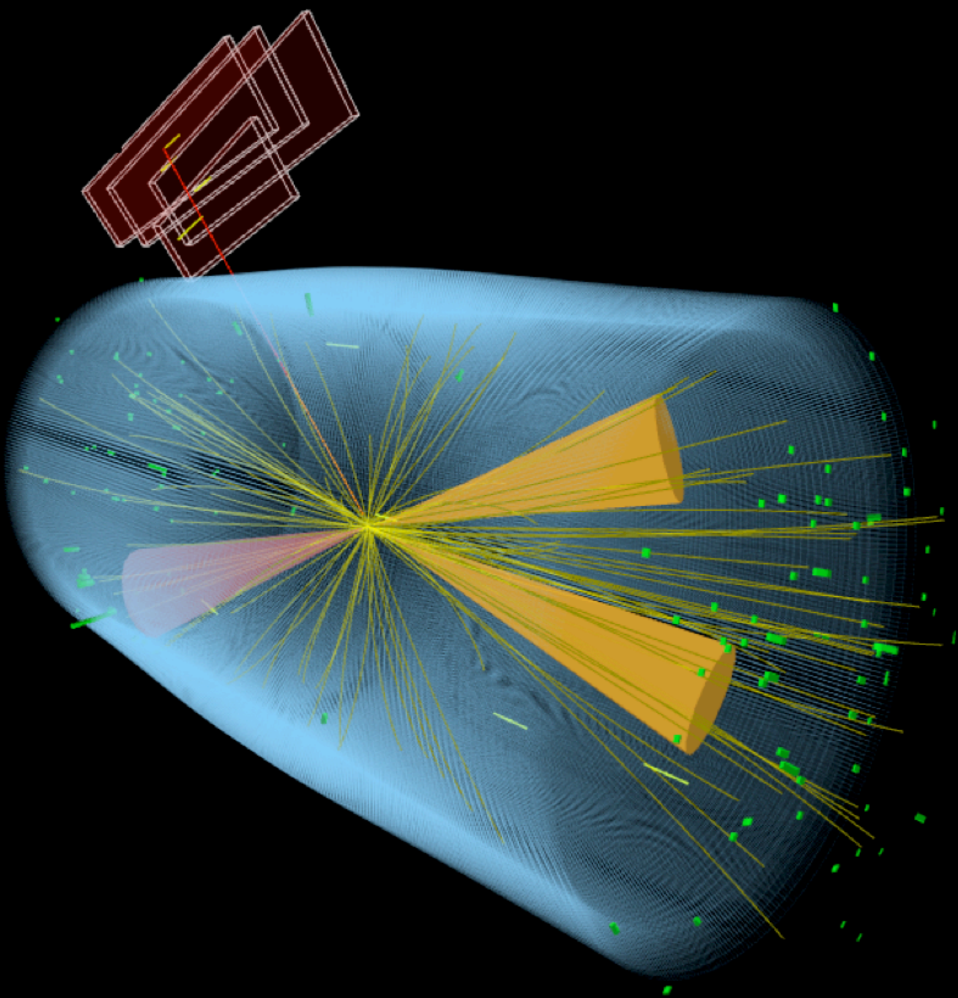
GPU as a service [arxiv.2007.10359]
Each client is given 7,000 events
A single GPU can serve up to 500 HLT nodes
with 10% increase in throughput

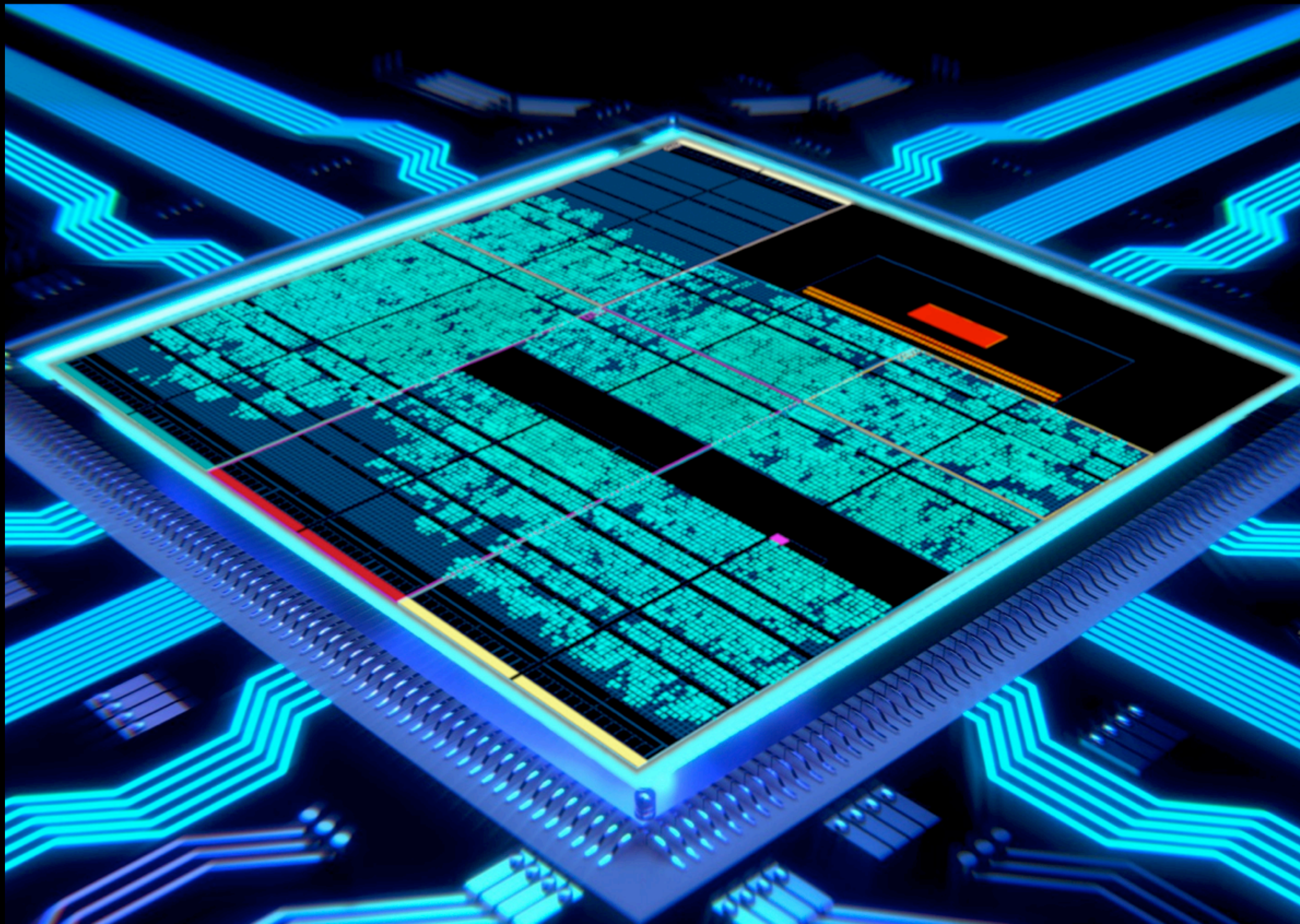


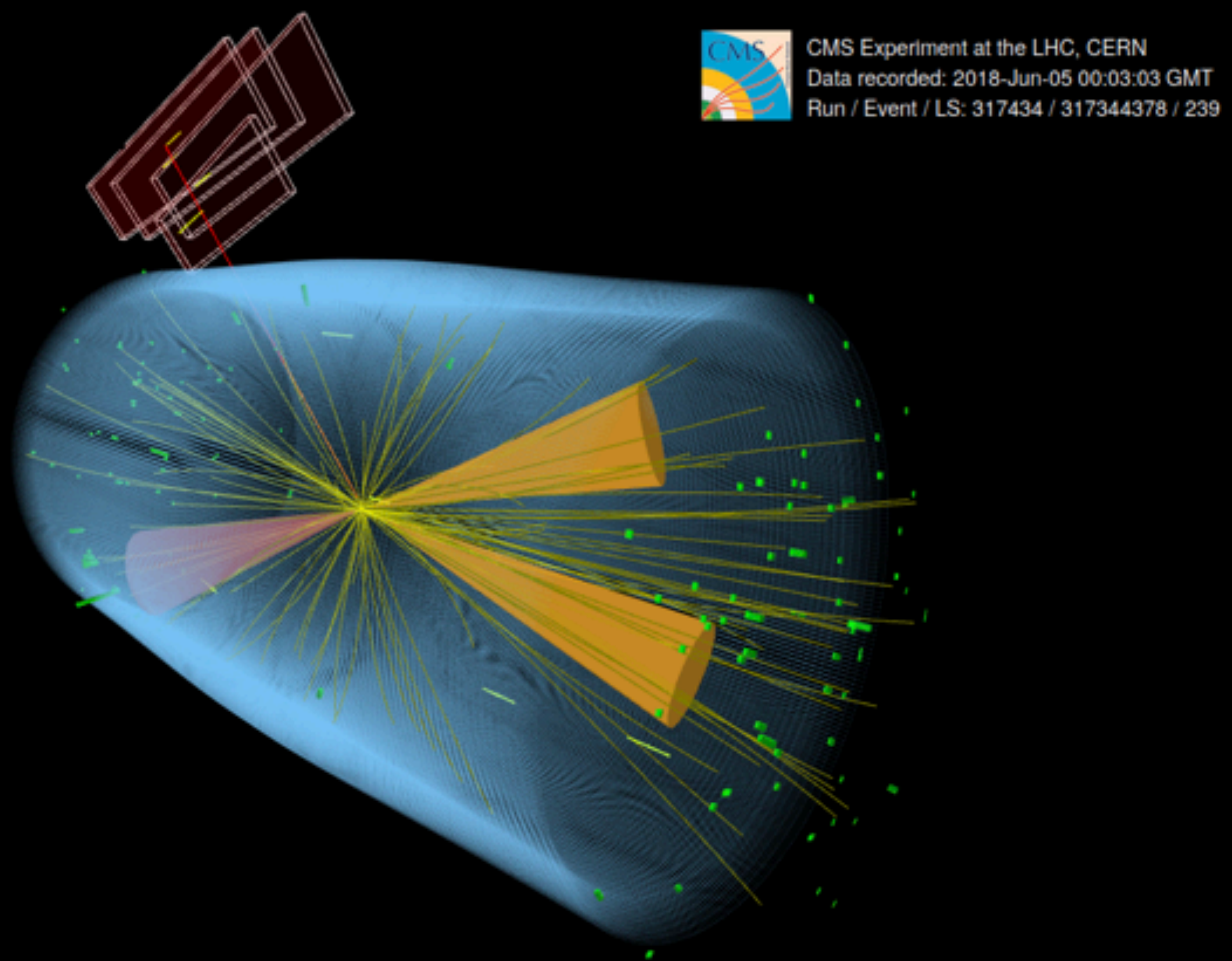
FPGA as a service [arxiv.2010.08556]
A single service server capable of serving
1500 simultaneous clients while preserving throughput
25Gbps network bandwidth limit hit above 1500

Summary

- **We hope to understand the fundamental structure of nature**
  - we expect new phenomena to answer those questions
  - but these are rare so we build large scale experimental setups
- **The challenge ahead is big**
  - more data, more complex data, not enough resources
- **This is why we need to push ML to the edge**
  - to do more with less (faster & better)
- **And hopefully discover new phenomena!**



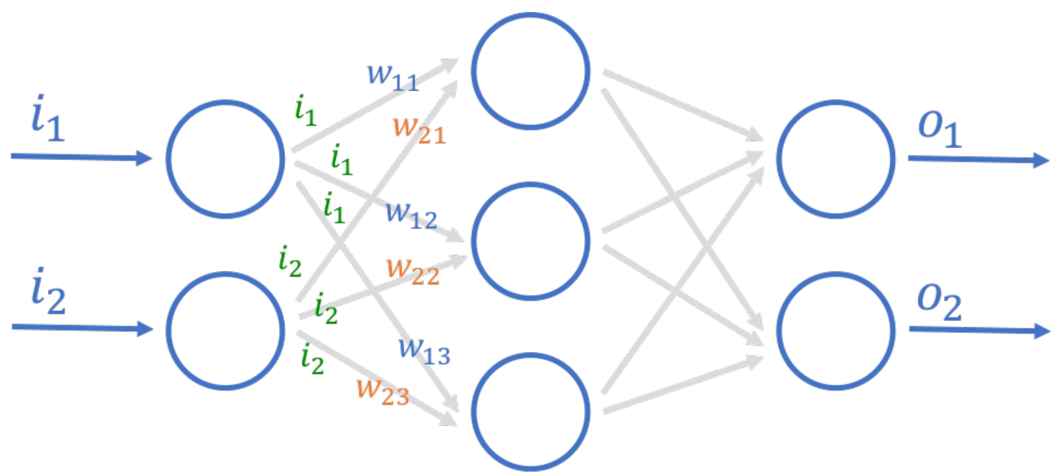



CMS Experiment at the LHC, CERN
Data recorded: 2018-Jun-05 00:03:03 GMT
Run / Event / LS: 317434 / 317344378 / 239

BACKUP

Neural Network inference on FPGA

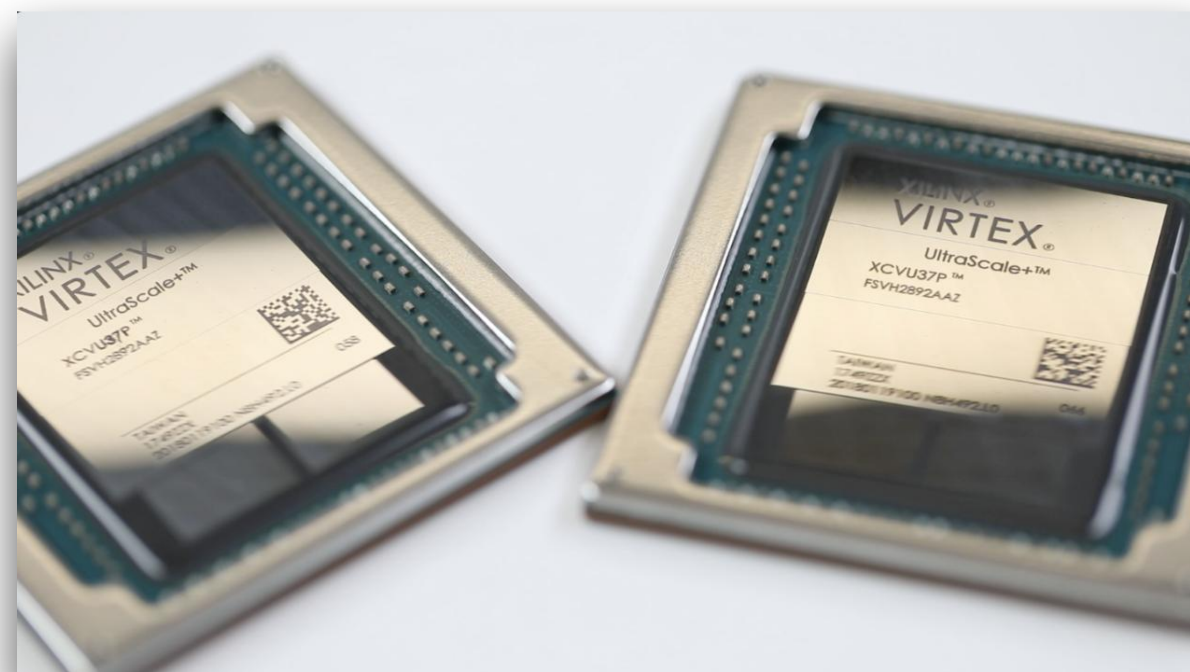
Neural network inference
=
matrix multiplication



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

Efficient implementation on FPGA uses
DIGITAL SIGNAL PROCESSORS

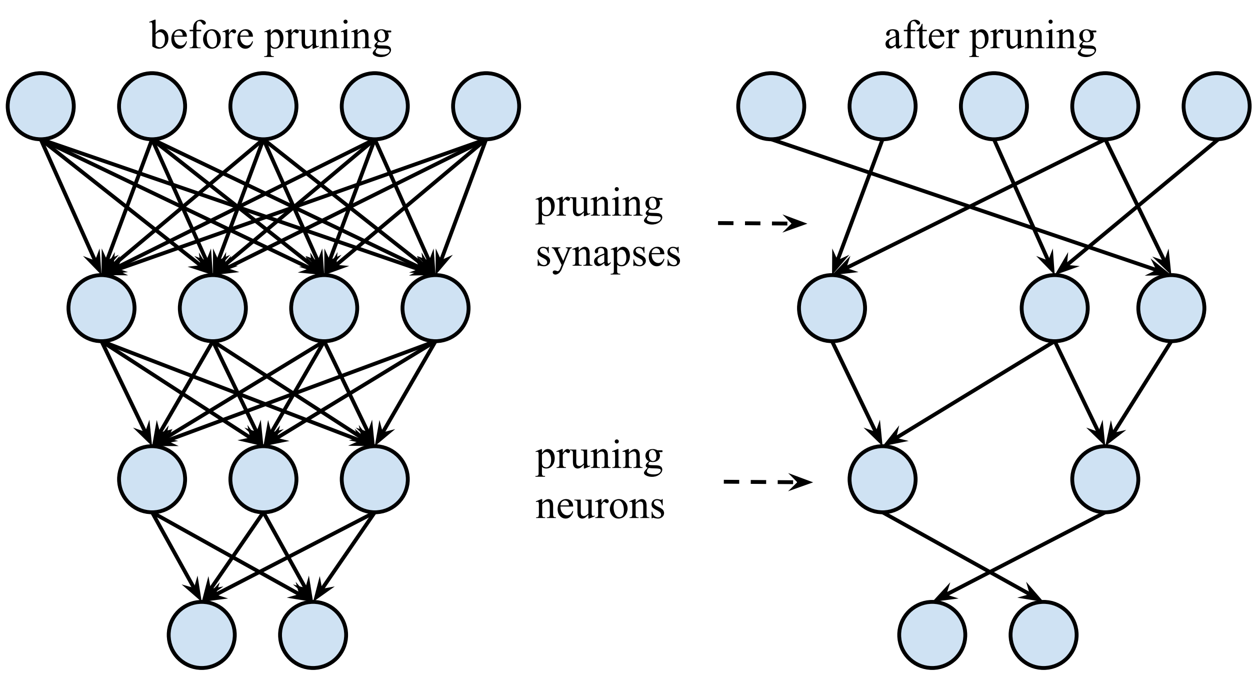
There are about 5–10k DSPs in modern FPGAs!

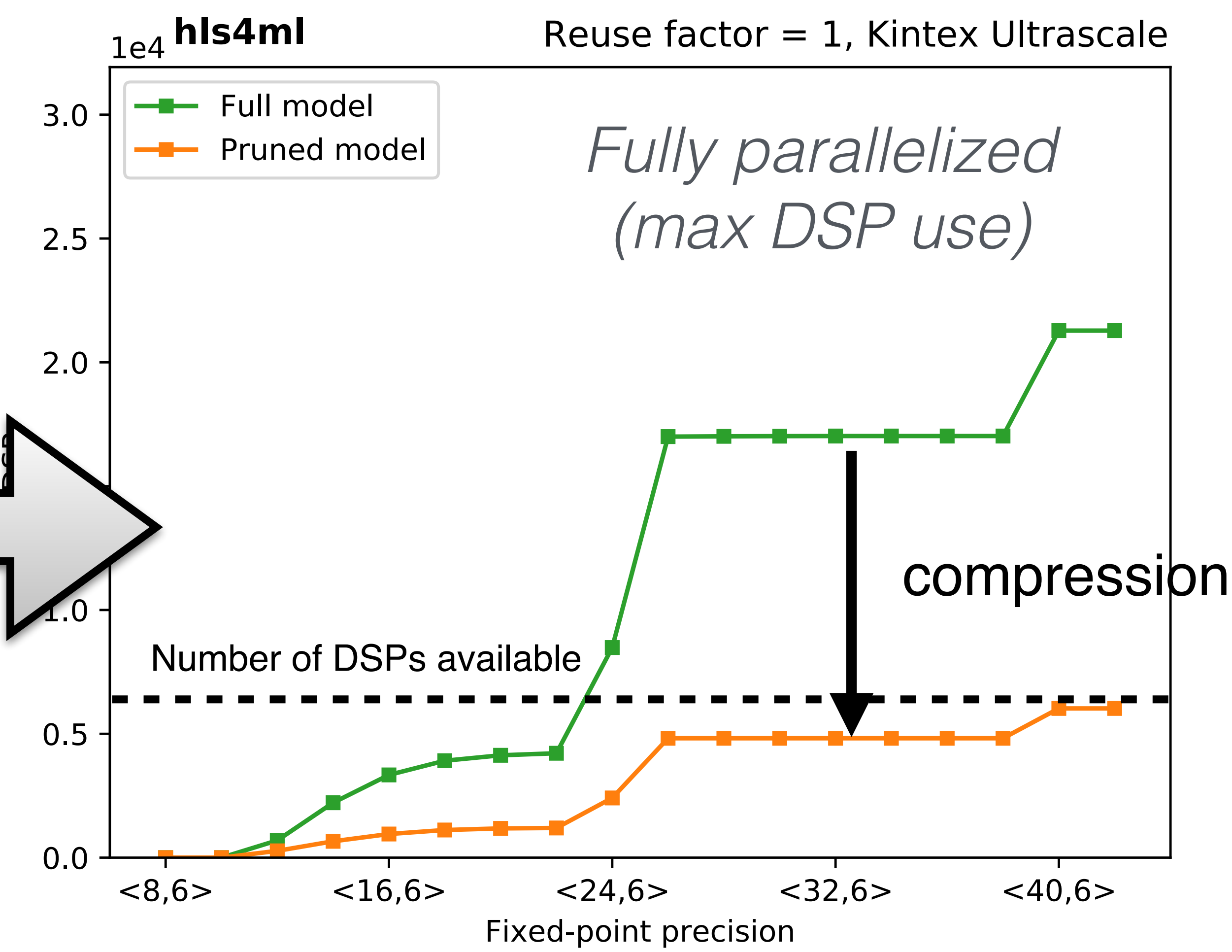


ex: Xilinx Virtex Ultrascale +

Make the model fit on one chip

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

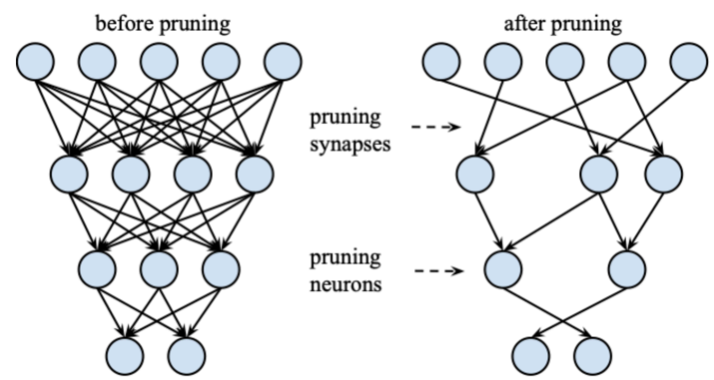


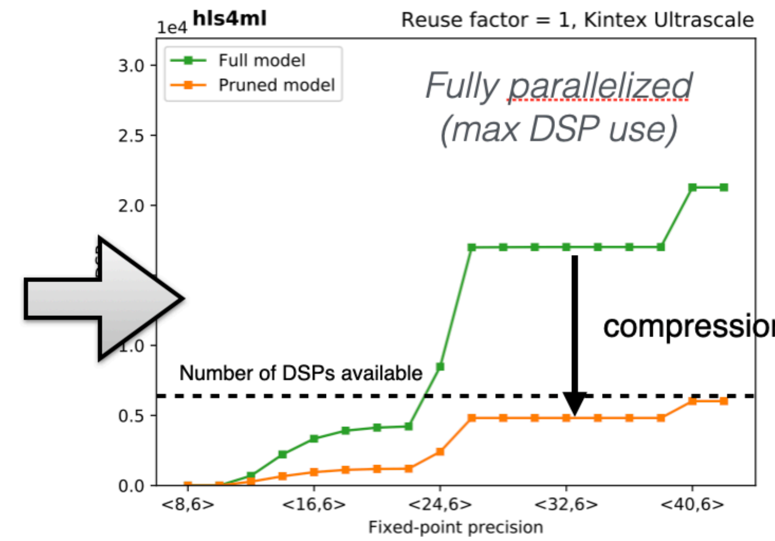


70% compression ~ 70% fewer DSPs

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

0101.1011101010

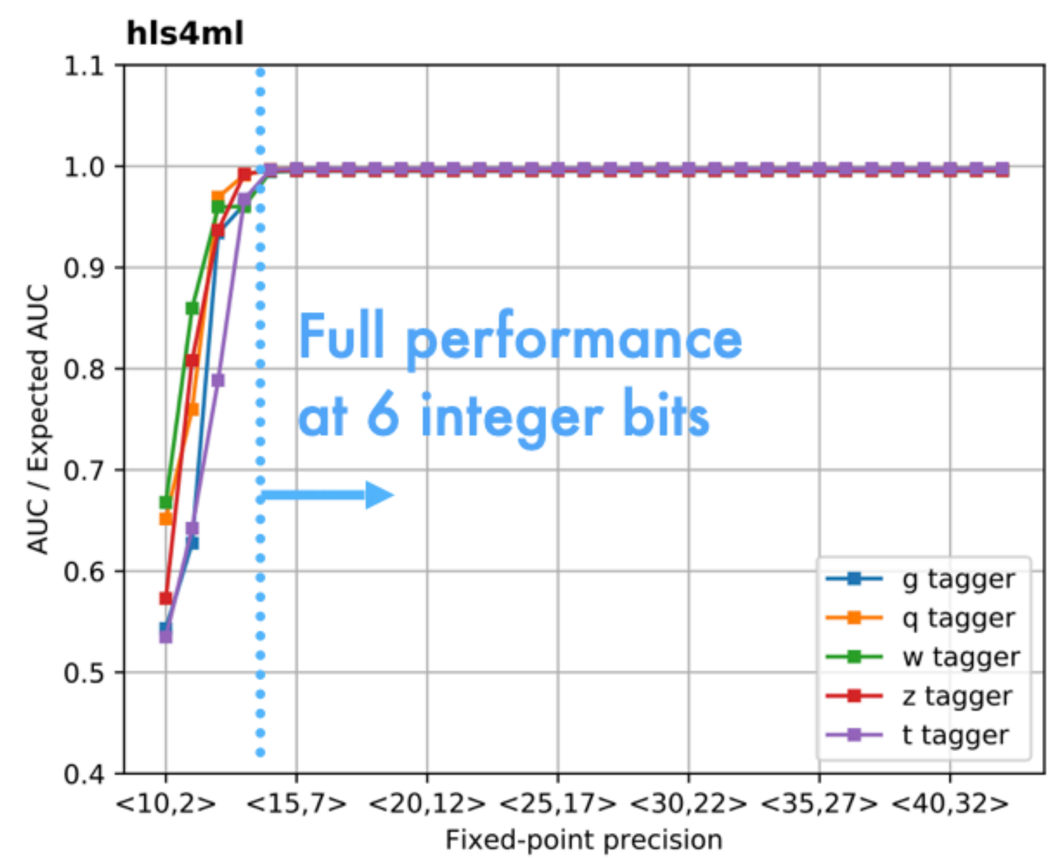
integer | fractional

width

ap_fixed<14,4>

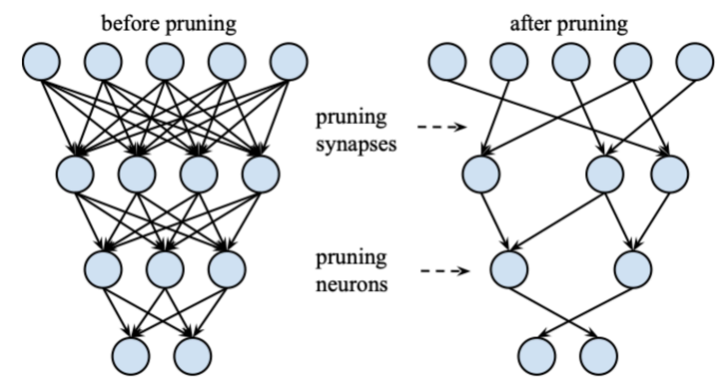
Scan integer bits

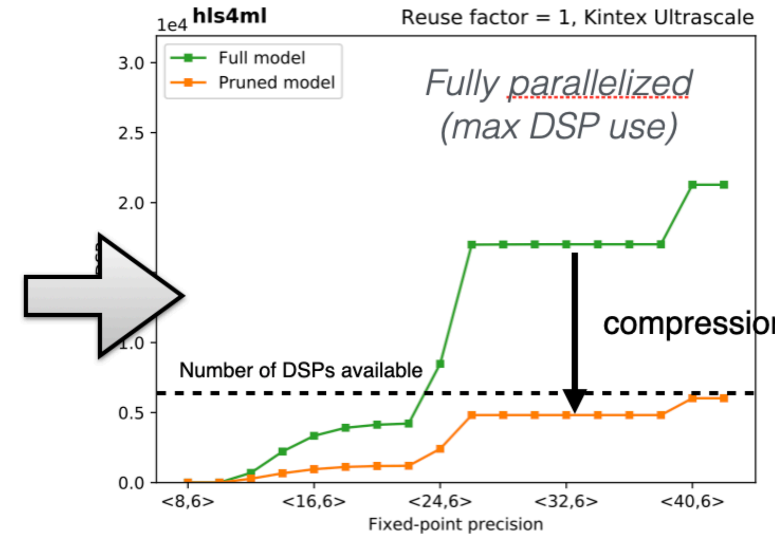
Fractional bits fixed to 8



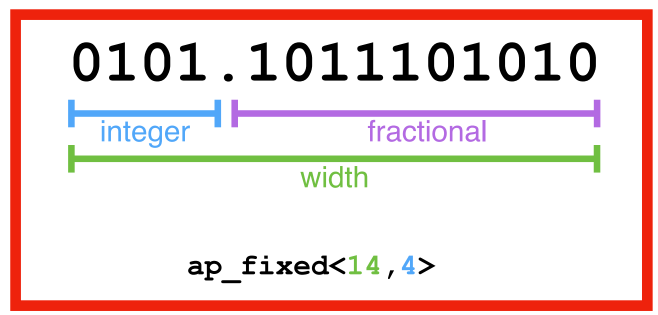
Make the model fit on one chip

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

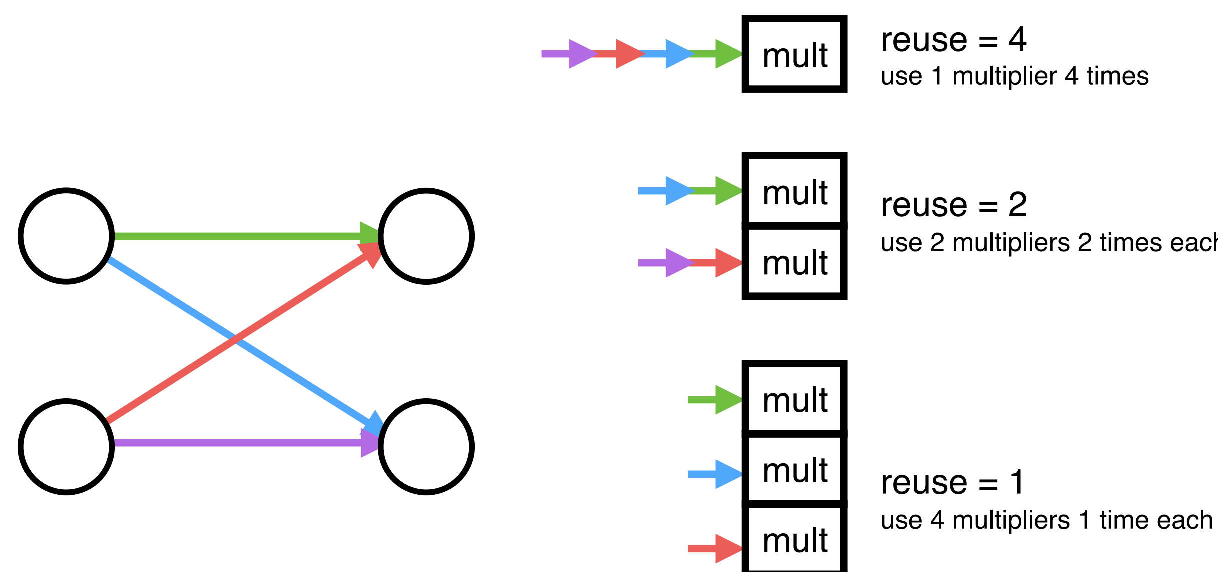




  - **Quantisation:** represents numbers with few bits reduce resources



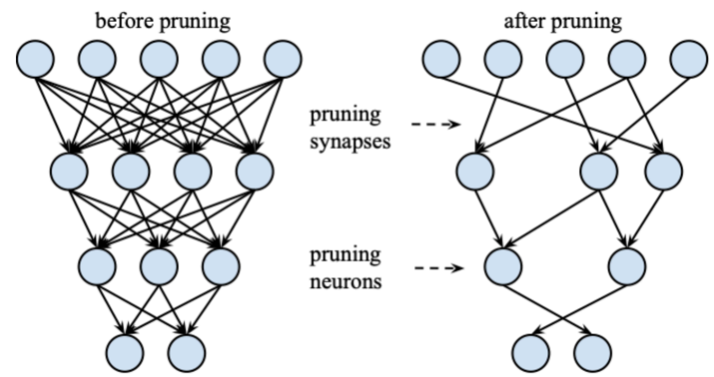
  - **Reuse:** allocate resources for each operation (run all network in one clock) vs spread calculation across several clock cycles

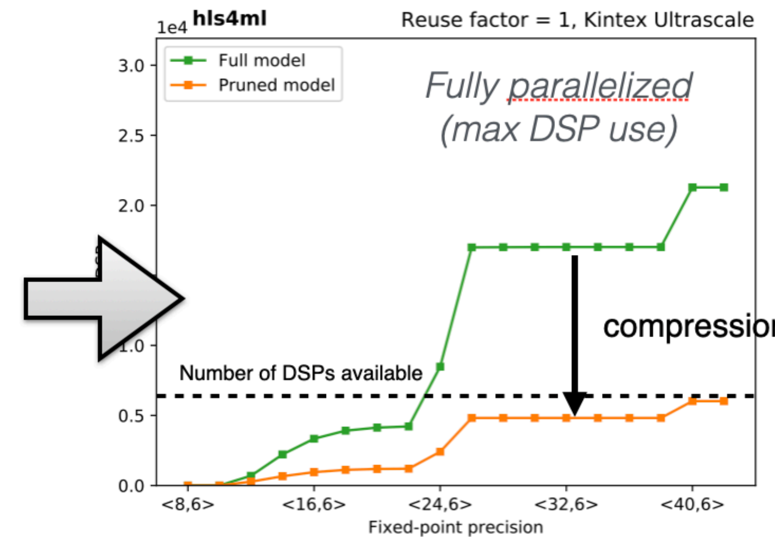


more parallelization → more resources

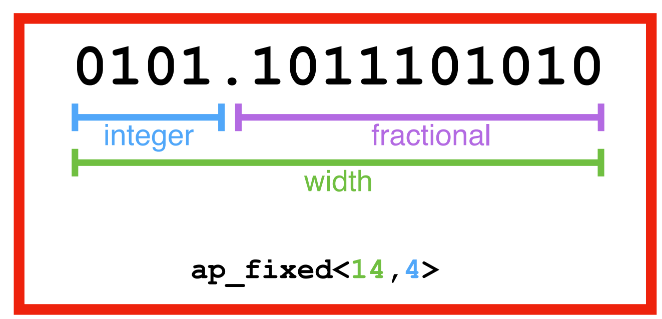
Make the model fit on one chip

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

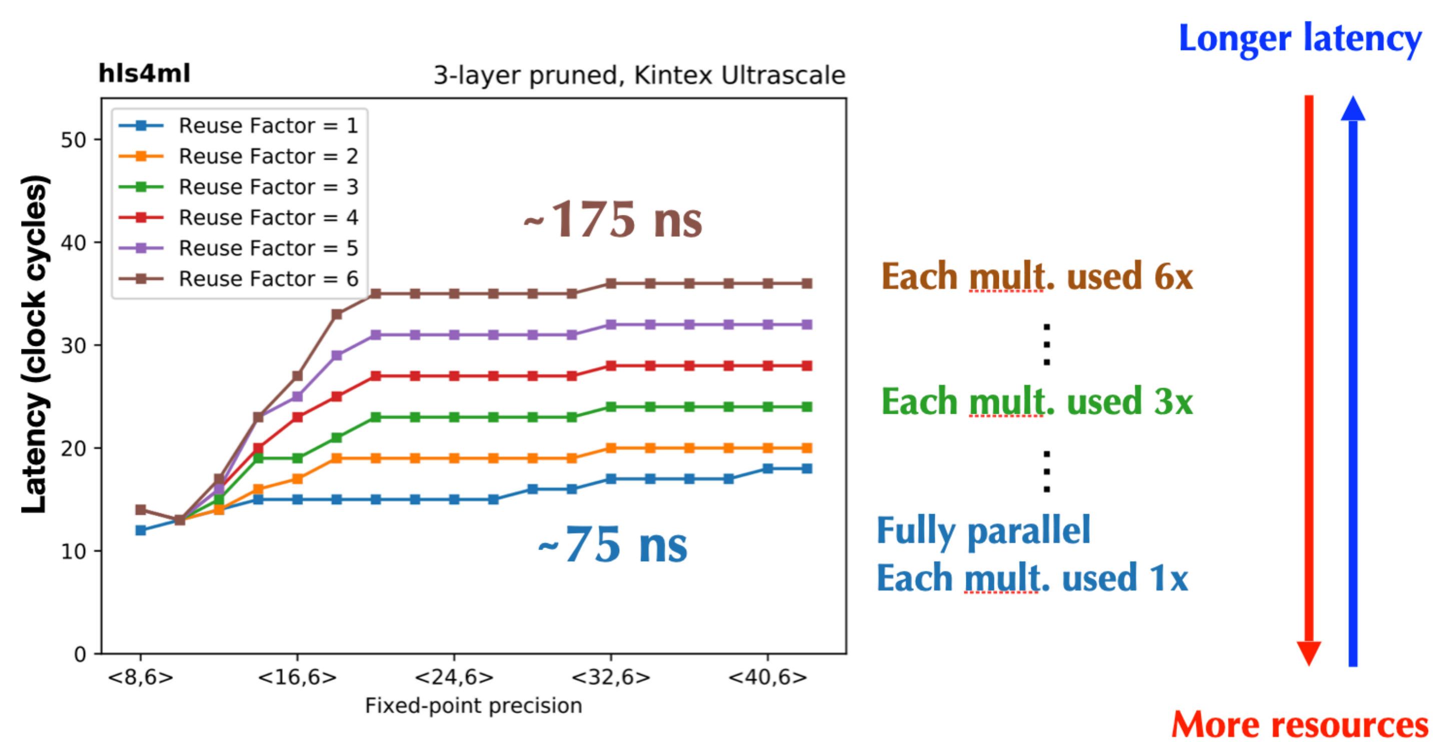




  - **Quantisation:** represents numbers with few bits reduce resources

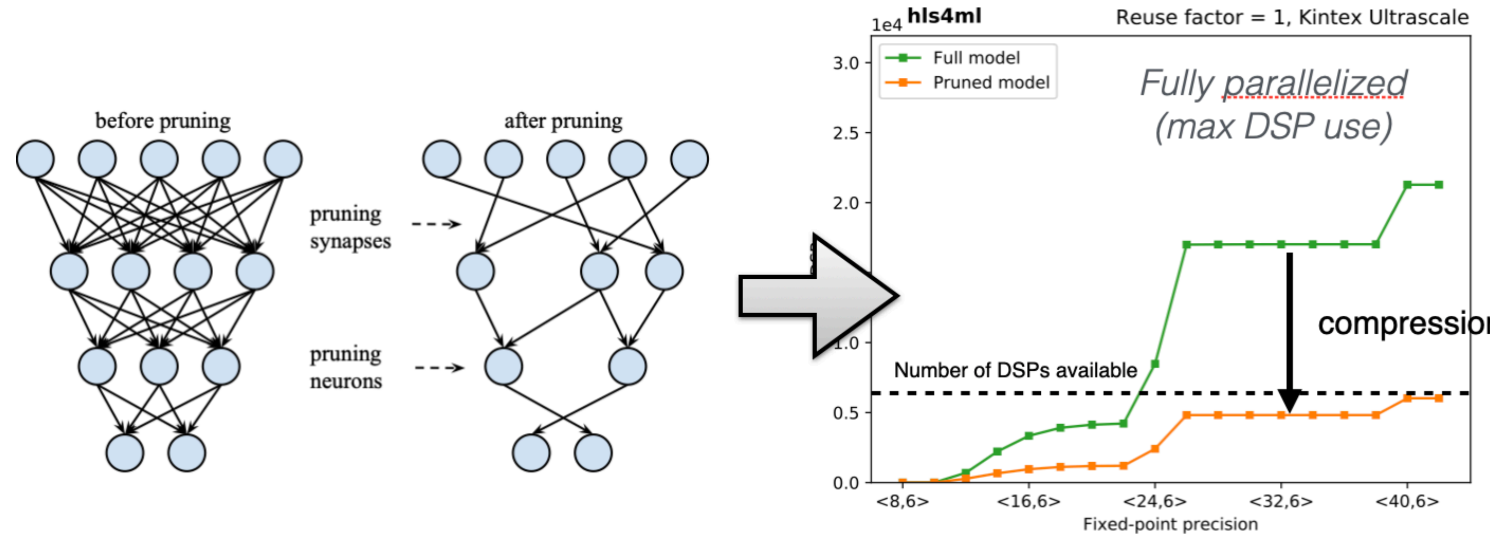


  - **Reuse:** allocate resources for each operation (run all network in one clock) vs spread calculation across several clock cycles

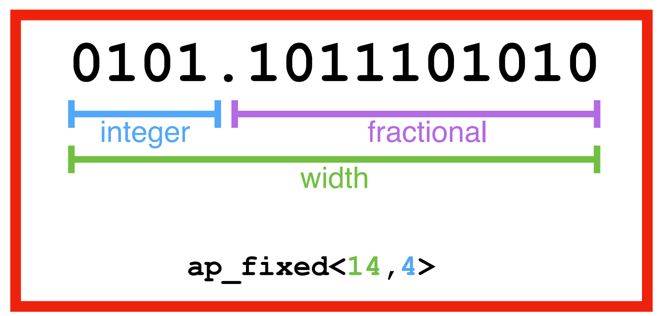


Make the model fit on one chip

- 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



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