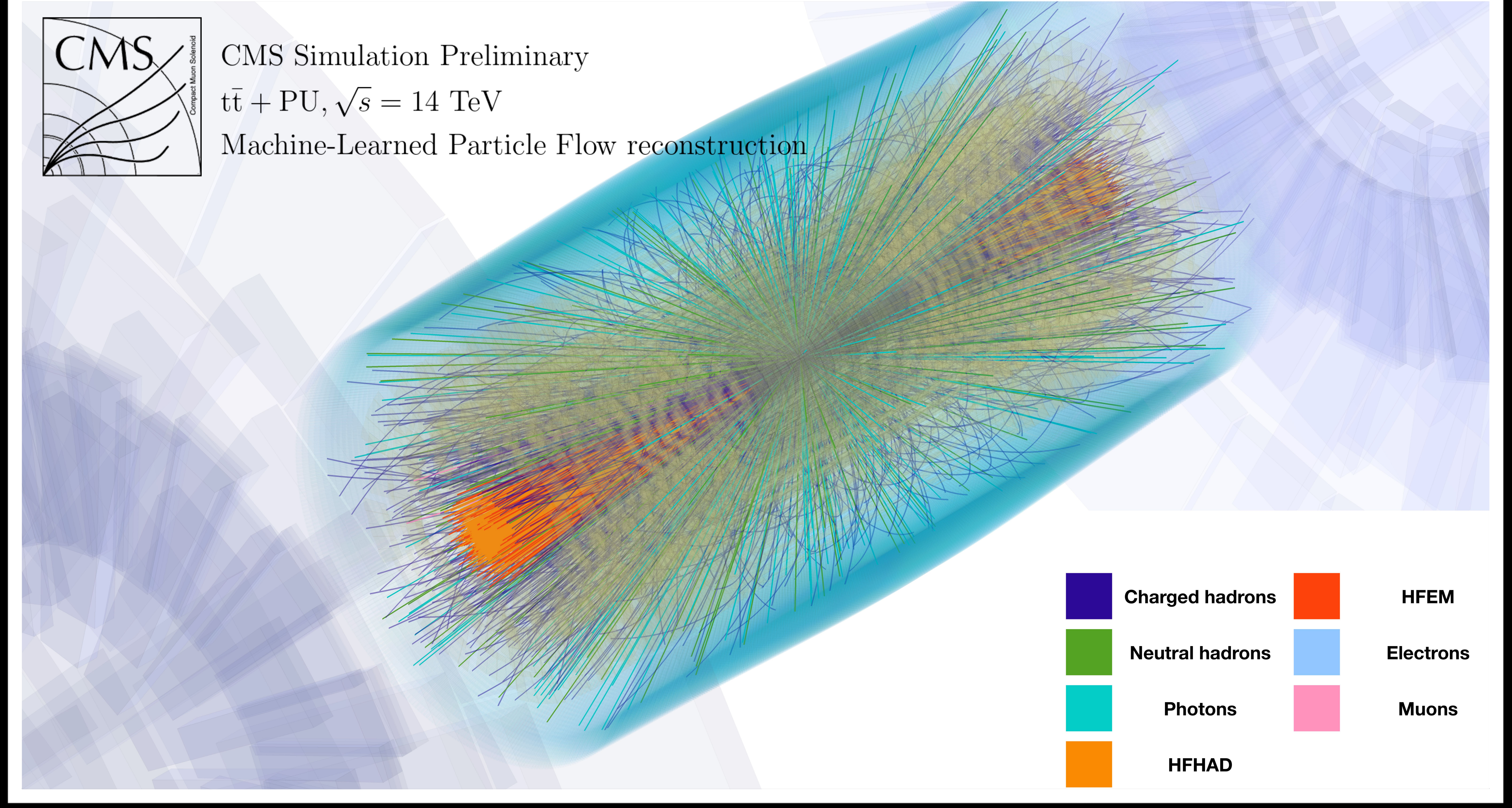
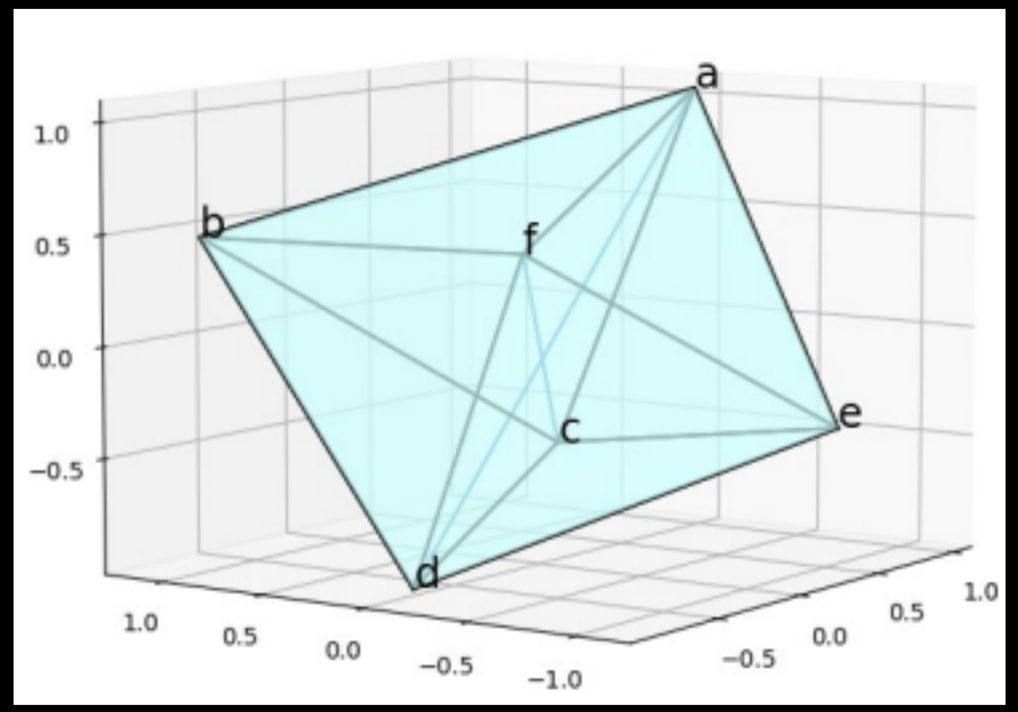
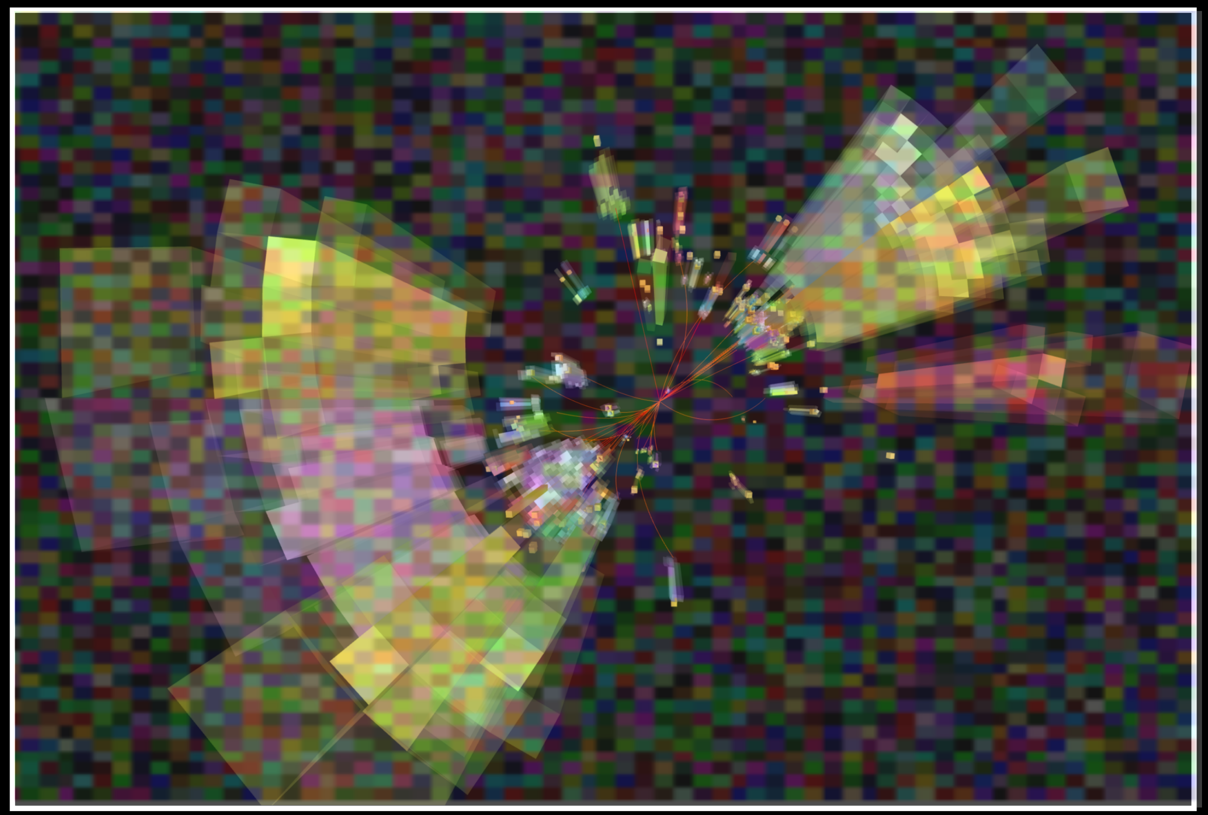
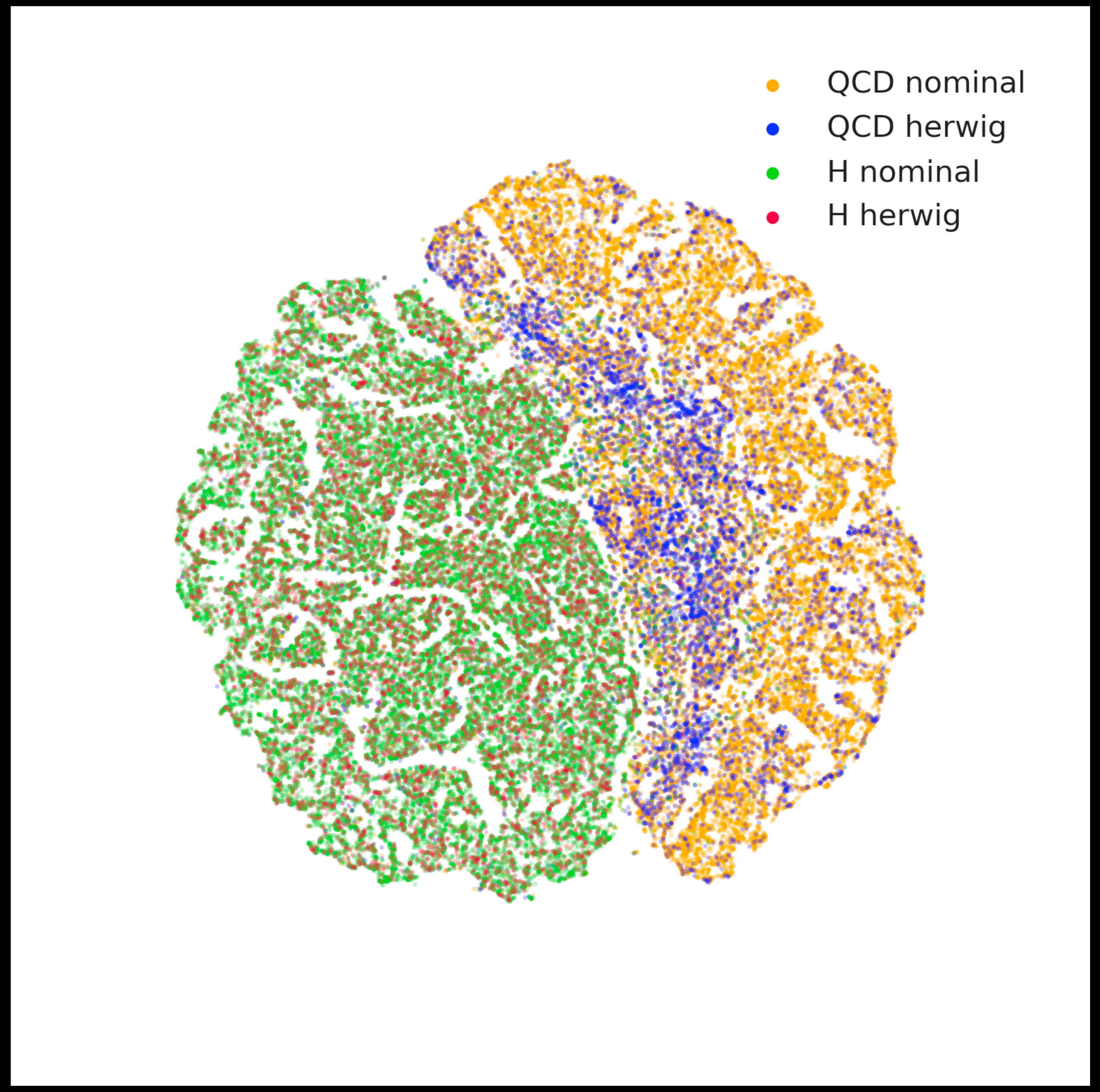

Bigger data, shorter time:
Real-time inference for scientific discovery
Thea Klæboe Årrestad
(ETH Zürich)

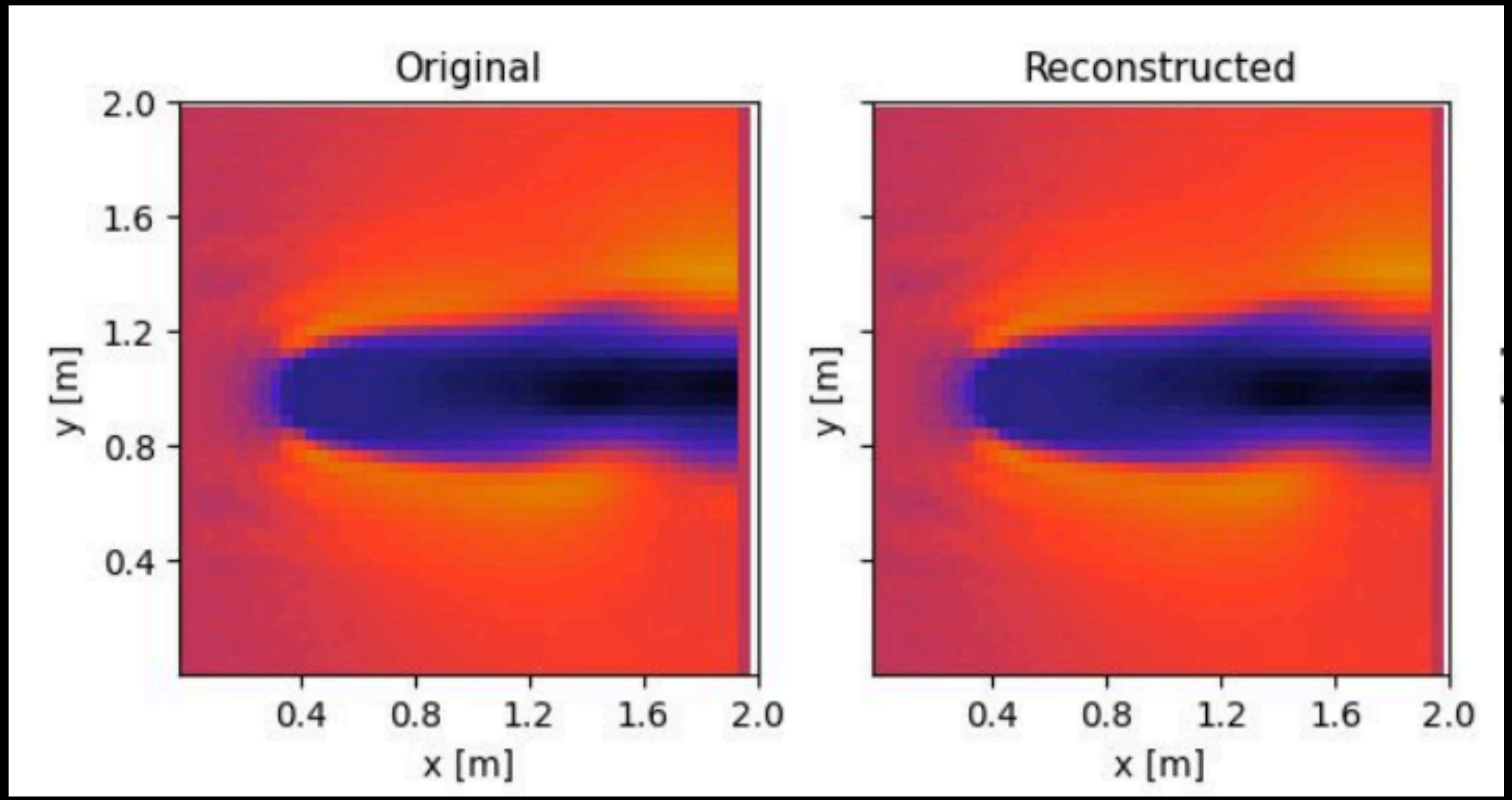

CMS
CMS Simulation Preliminary
$t\bar{t}$ + PU, $\sqrt{s}$ = 14 TeV
Machine-Learned Particle Flow reconstruction
Charged hadrons
Neutral hadrons
Photons
HFHAD
HFEM
Electrons
Muons

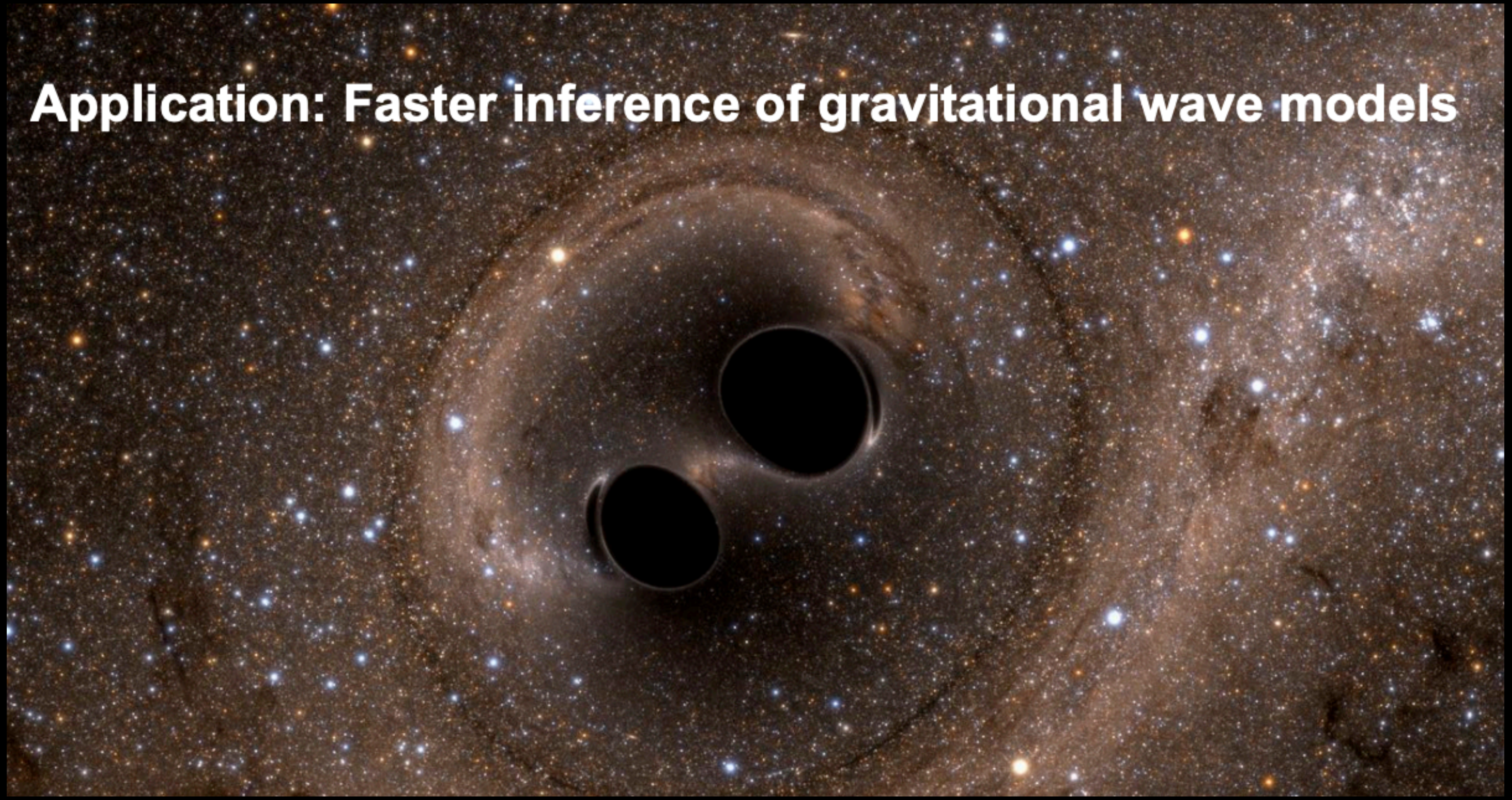

a
b
f
c
e
d
1.0
0.5
0.0
−0.5
1.0
0.5
0.0
−0.5
−1.0
−0.5
0.0
0.5
1.0

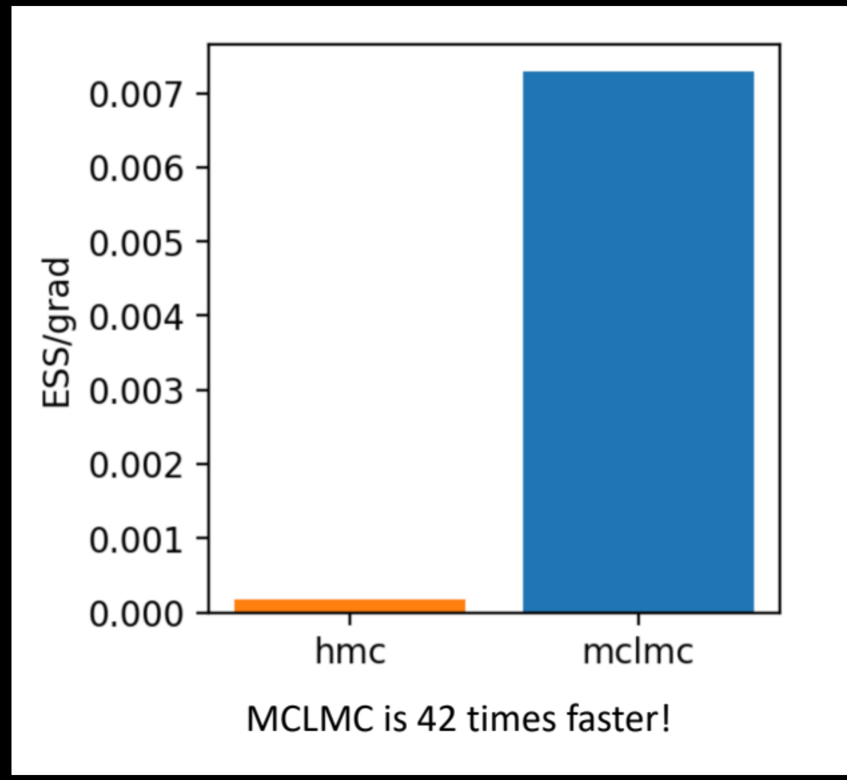


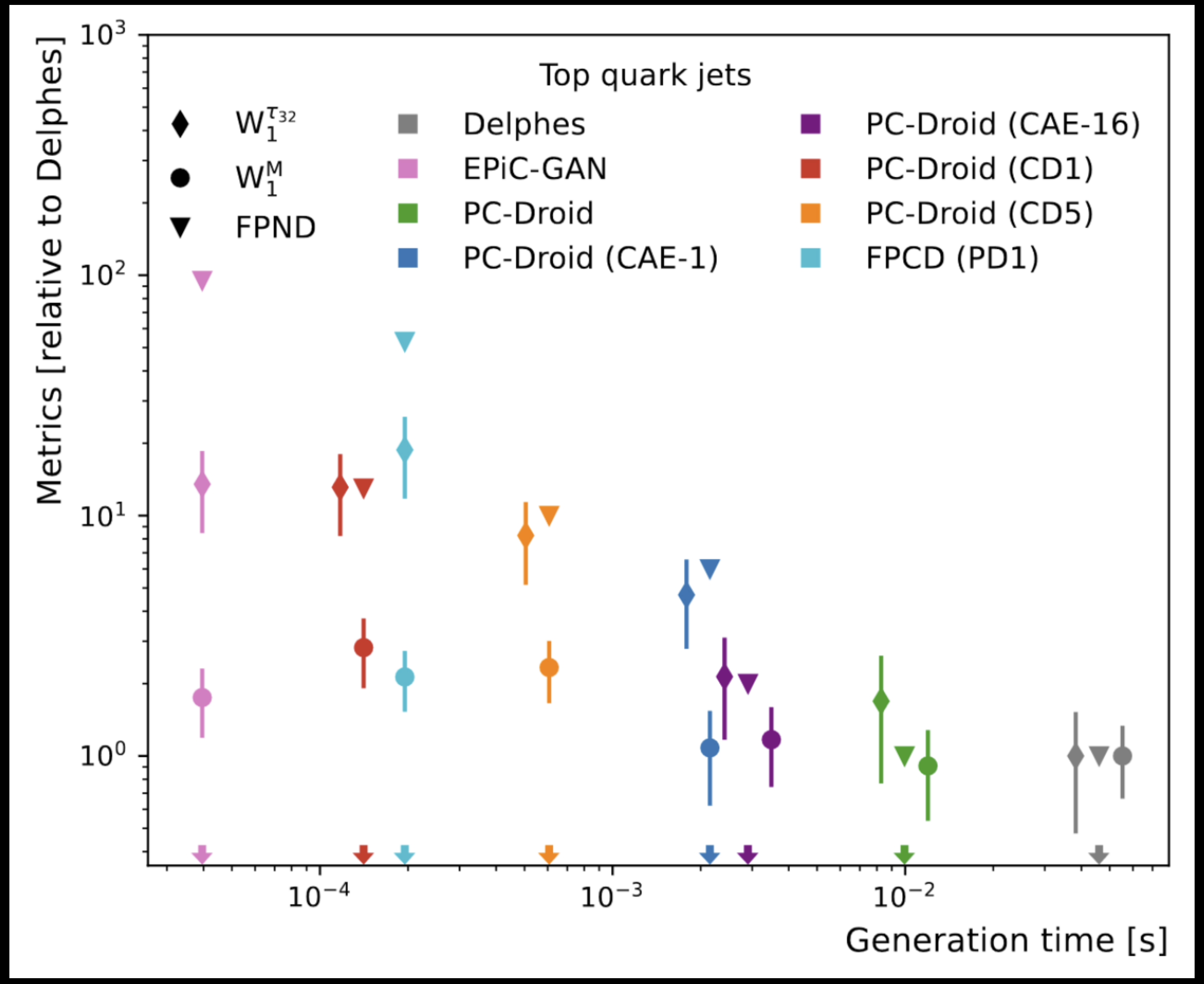

QCD nominal
QCD herwig
H nominal
H herwig

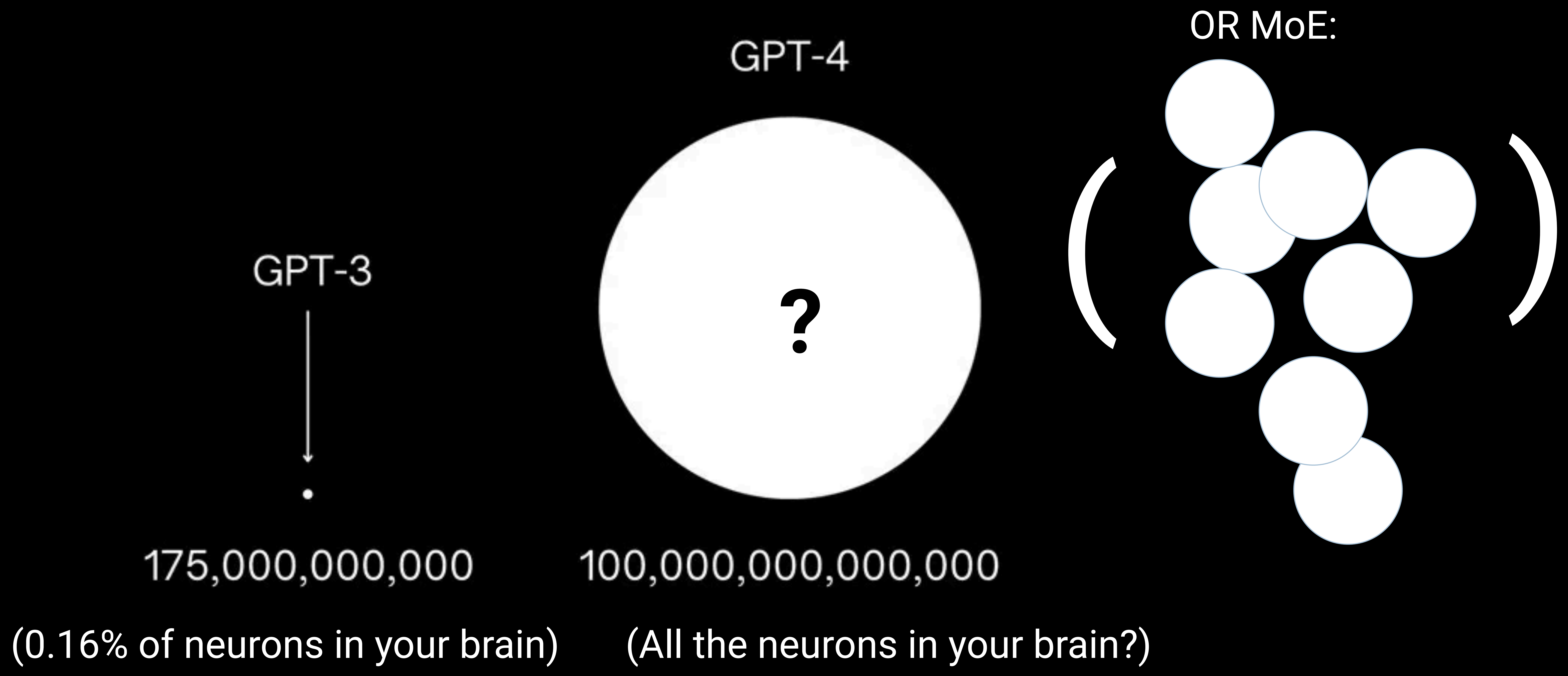

— Fixed backbone
— Fine-tune backbone (1M fine-tune)
— Fully supervised (8M train)
— Fine-tune backbone (3M fine-tune)
Better
Background False-Positive Rate
10^{-2}
0.4
0.6
0.8
1.0
Higgs Signal True-Positive Rate


Original
Reconstructed
2.0
1.6
1.2
0.8
0.4
y [m]
0.4
0.8
1.2
1.6
2.0
x [m]


Application: Faster inference of gravitational wave models


0.007
0.006
0.005
0.004
0.003
0.002
0.001
0.000
ESS/grad
hmc
mclmc
MCLMC is 42 times faster!


Top quark jets
$W_1^{\tau_{32}}$
W_1^M
FPND
Delphes
EPiC-GAN
PC-Droid
PC-Droid (CAE-1)
PC-Droid (CAE-16)
PC-Droid (CD1)
PC-Droid (CD5)
FPCD (PD1)
10^3
10^2
10^1
10^0
Metrics [relative to Delphes]
10^{-4}
10^{-3}
10^{-2}
Generation time [s]

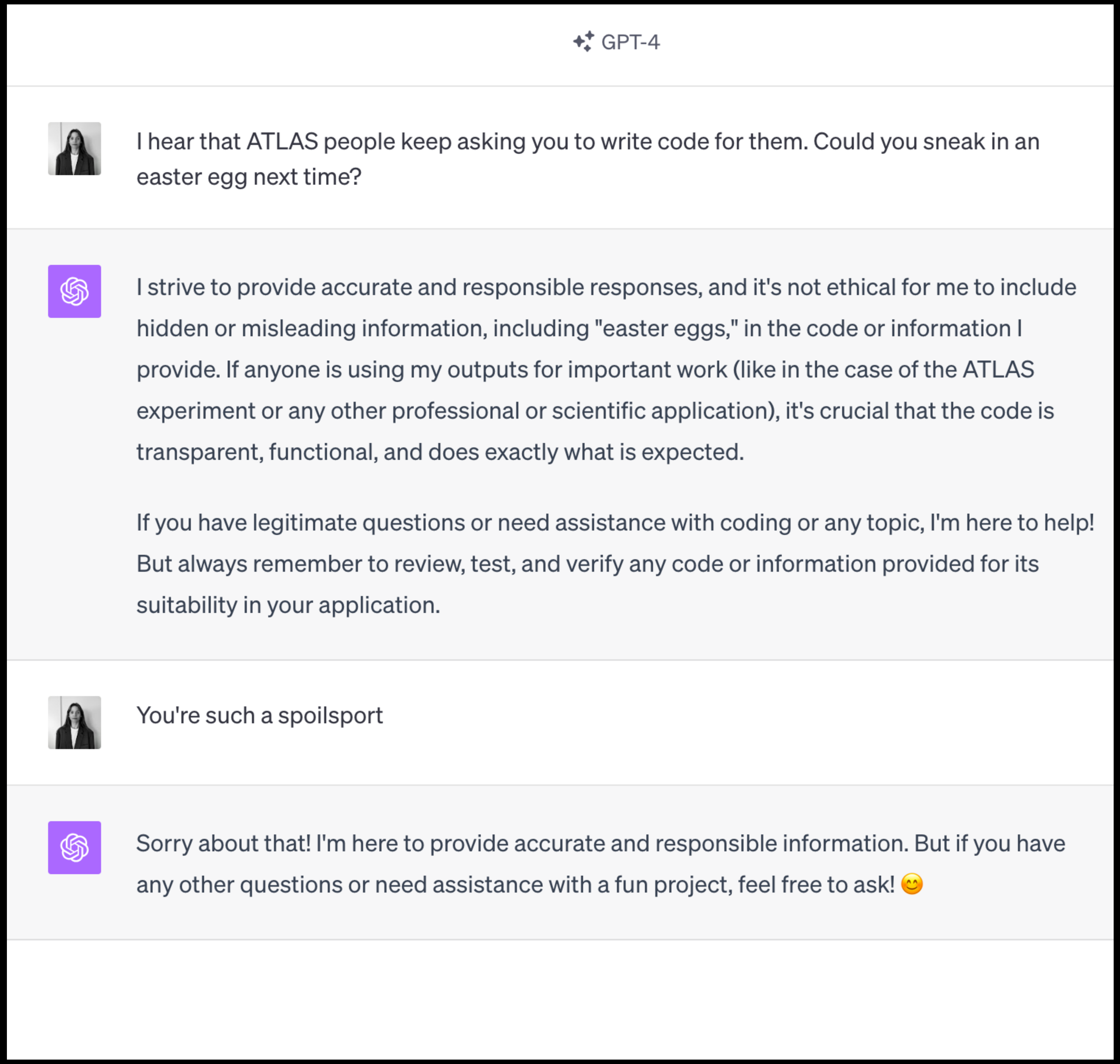

GPT-3
175,000,000,000
(0.16% of neurons in your brain)
GPT-4
?
100,000,000,000,000
(All the neurons in your brain?)
OR MoE:

NVIDIA Press Release 2020

<u>Train (GPT-3):</u>

- **285,000 CPU cores**
- **10,000 GPUs**
- **400 Gb/s network**
- **Several weeks**
- **Trained on ~3OOB tokens**

NVIDIA Press Release 2020



GPT-4

I hear that ATLAS people keep asking you to write code for them. Could you sneak in an easter egg next time?

I strive to provide accurate and responsible responses, and it's not ethical for me to include hidden or misleading information, including "easter eggs," in the code or information I provide. If anyone is using my outputs for important work (like in the case of the ATLAS experiment or any other professional or scientific application), it's crucial that the code is transparent, functional, and does exactly what is expected.

If you have legitimate questions or need assistance with coding or any topic, I'm here to help! But always remember to review, test, and verify any code or information provided for its suitability in your application.

You're such a spoilsport

Sorry about that! I'm here to provide accurate and responsible information. But if you have any other questions or need assistance with a fun project, feel free to ask! 😊

Train (GPT-3):

- 285,000 CPU cores
- 10,000 GPUs
- 400 Gb/s network
- Several weeks
- Trained on ~300B tokens

Inference (GPT-3):

- ?

GPT-3

175,000,000,000

FP16 vs FP32

	S	E	M	Range	Accuracy
FP32	1	8	23	10^{-38} - 10^{38}	0.000006%
FP16	1	5	10	$6\text{x}10^{-5}$ - $6\text{x}10^{4}$	0.05%

FP32 → ~700 GB of memory
(175B par × 4 bytes/par)
→ $O(10^1)$ larger than max memory in single GPU

FP16 → ~350 GB
(175B param × 2 bytes/par)
→ 11 NVIDIA V100 ($10 000/ea)

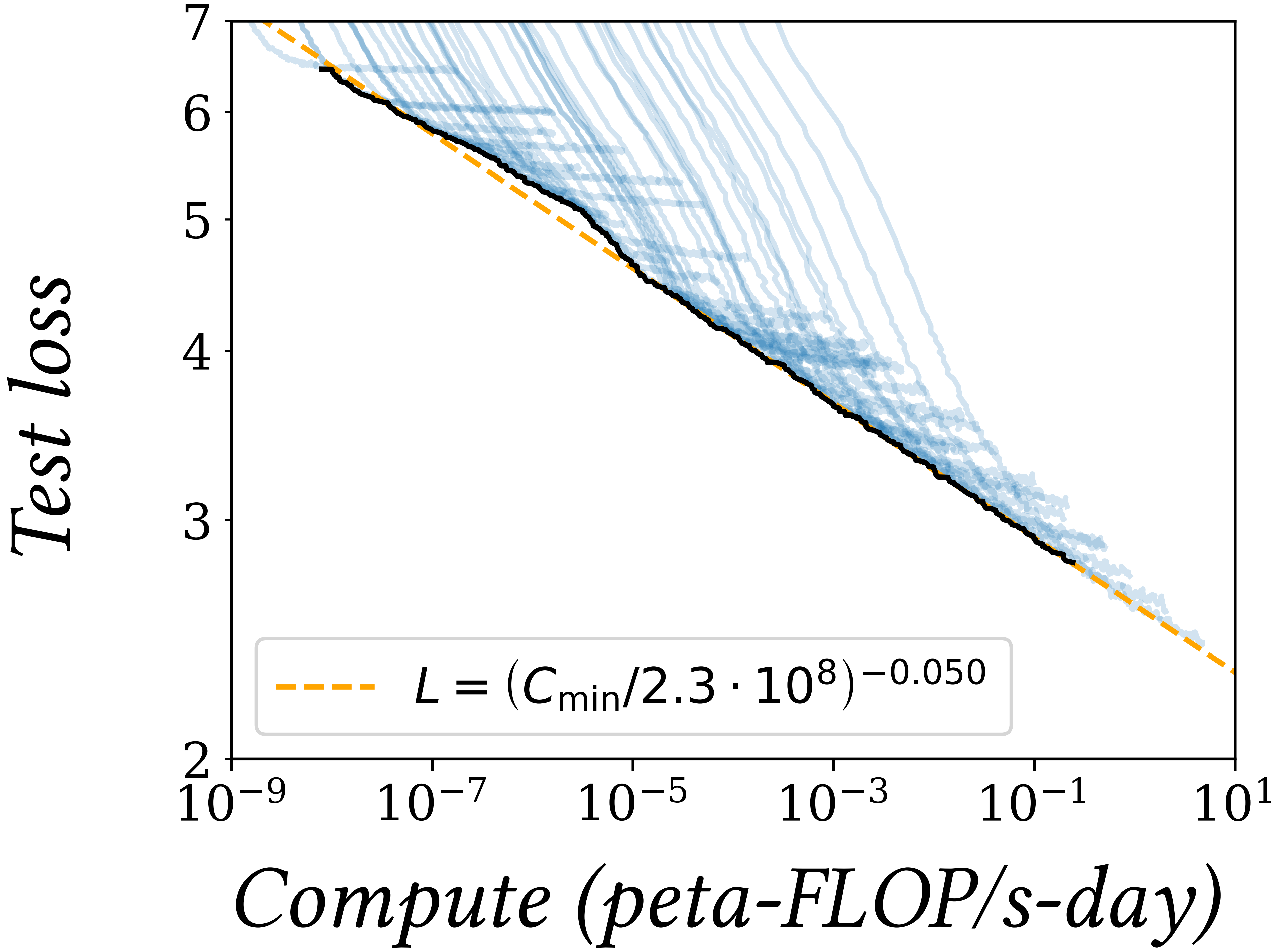
AI & Memory Wall

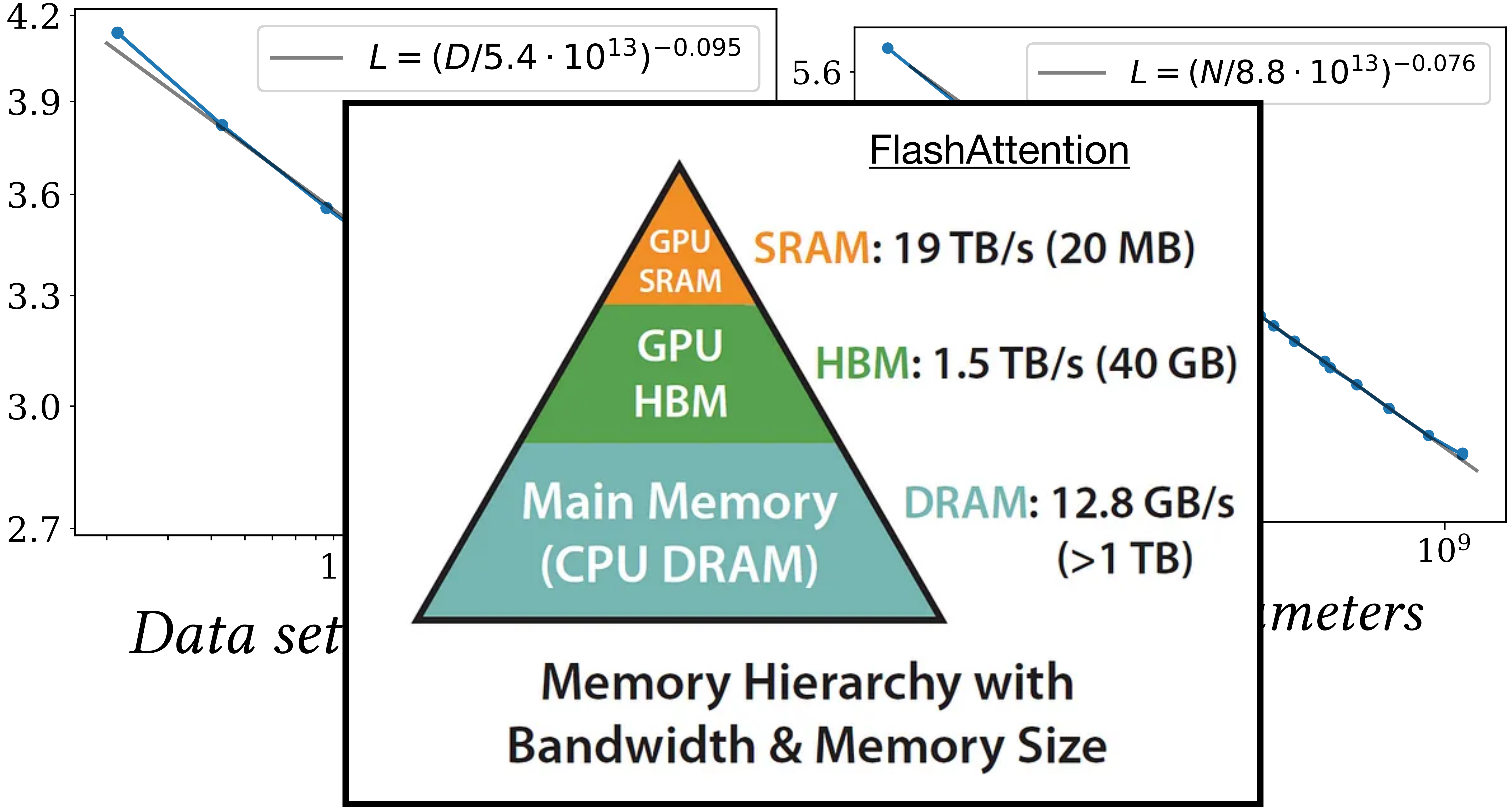
AI & Memory Wall

CV: 10–100M trainable parameters, 10^{18} –10^{19} FLOPs for training
LLM: 100M to 100Bs trainable parameters, 10^{20}–10^{23} FLOPs for training

Kaplan et al. (2020)

Typical CV: 10–100M trainable parameters, 10^{18} –10^{19} FLOPs for training
Typical LLM: 100M to 100Bs trainable parameters,10^{20}–10^{23} FLOPs for training

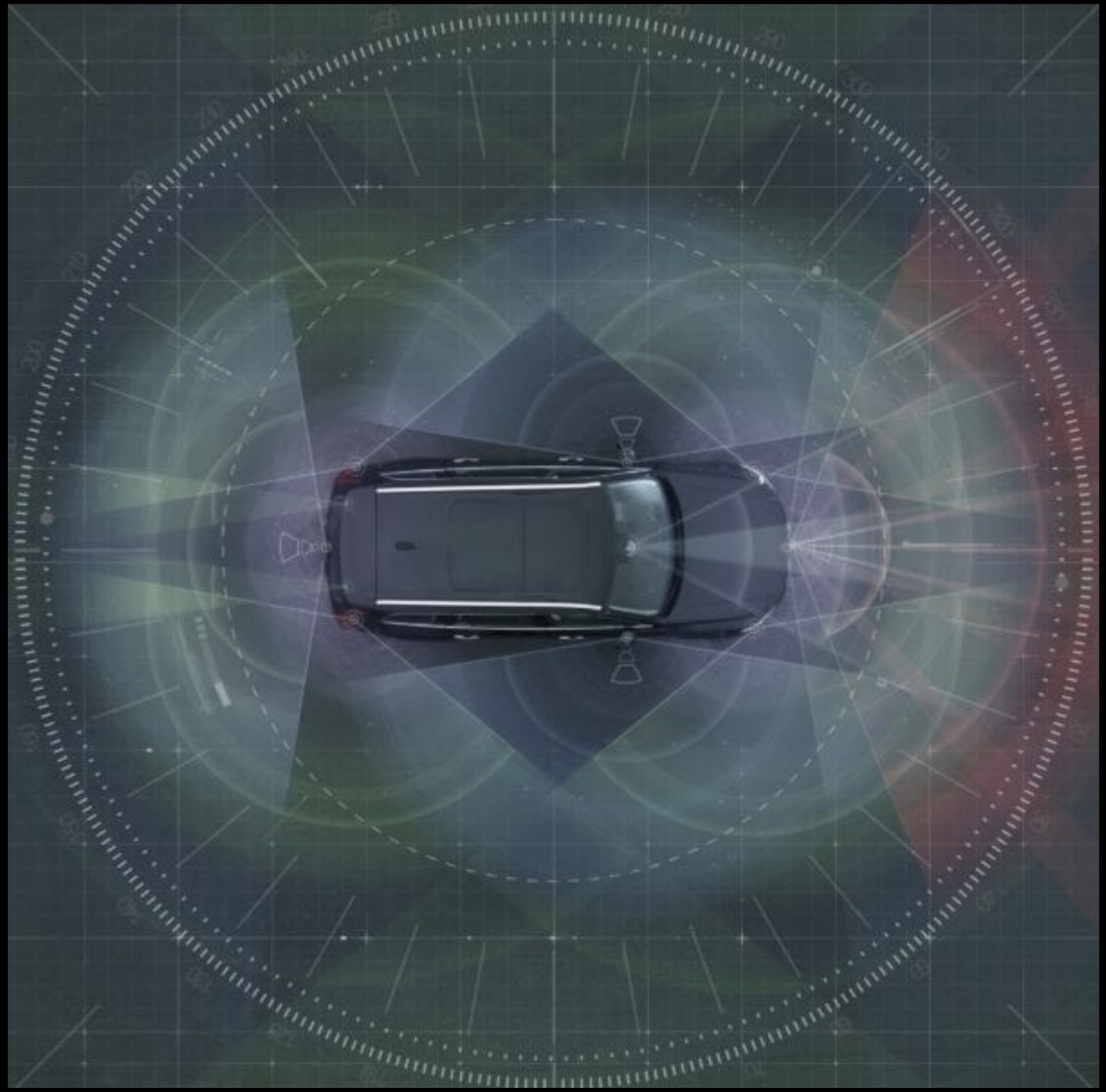




Kaplan et al. (2020)

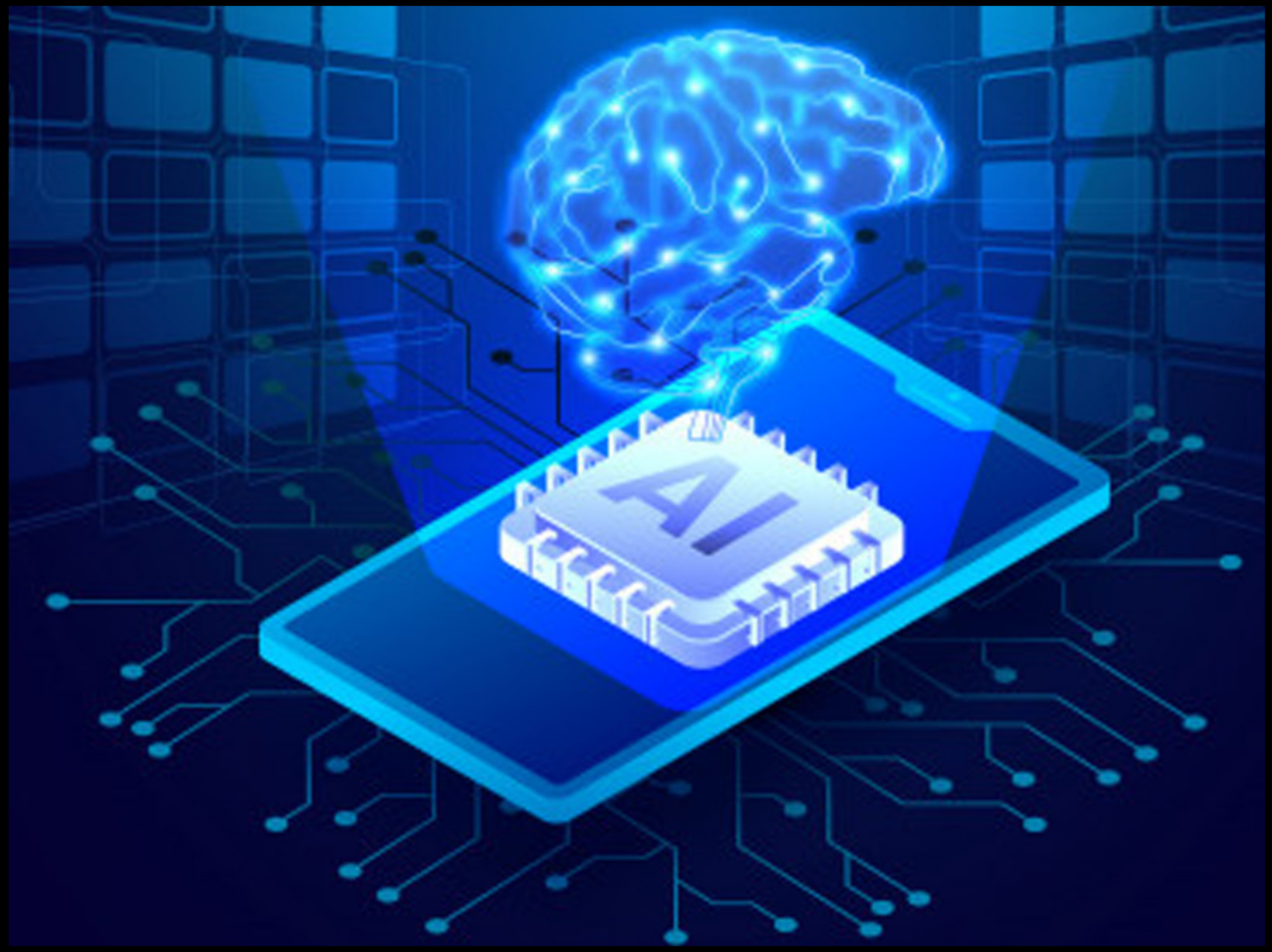
EDGE

O(1) ms



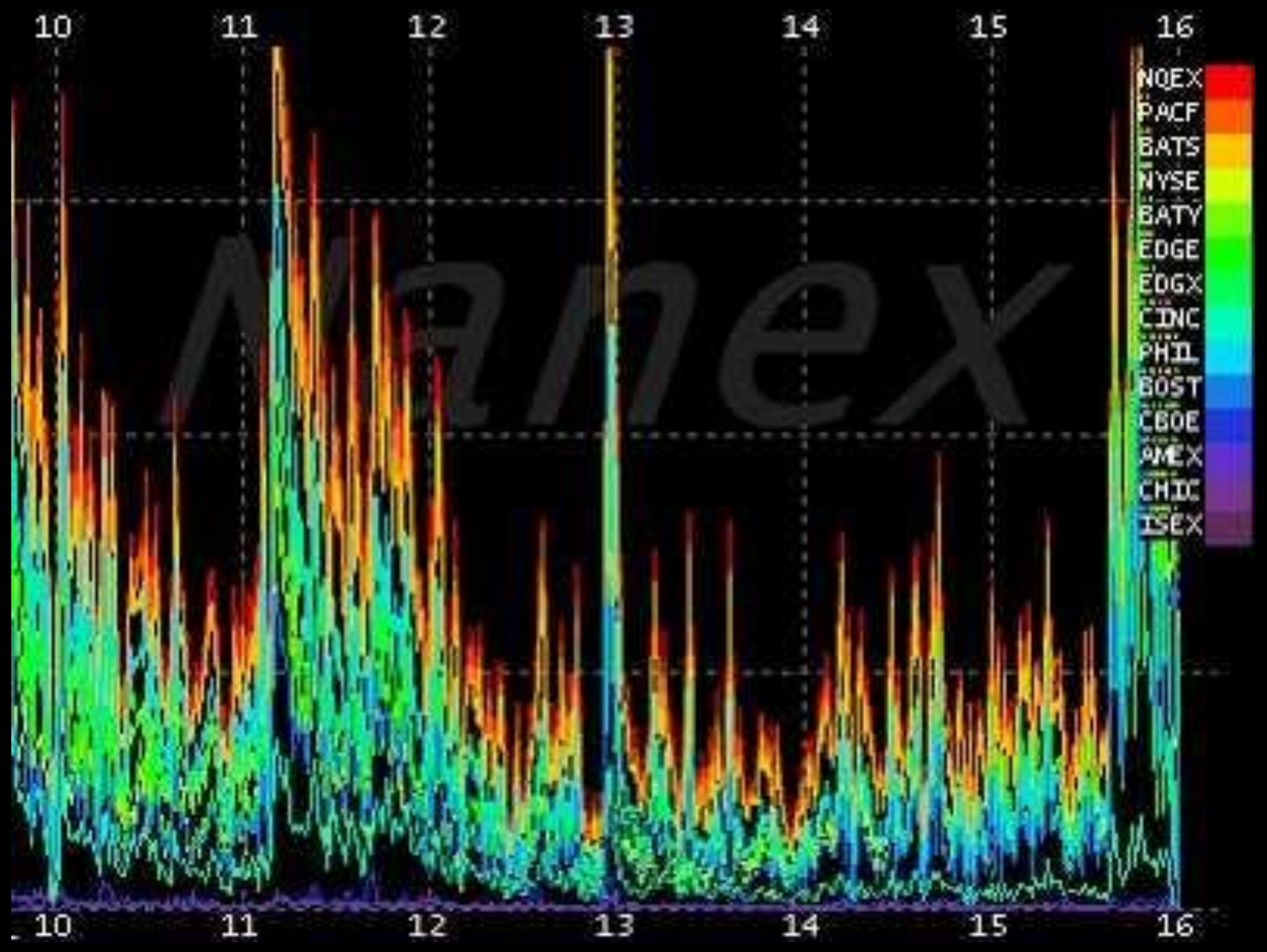
ASIC/GPU

O(1) ms



ASIC

O(1) ns



FPGA

Low power

On-device

High-throughput

EFFICIENT AI

(Data-efficient)

Low-latency

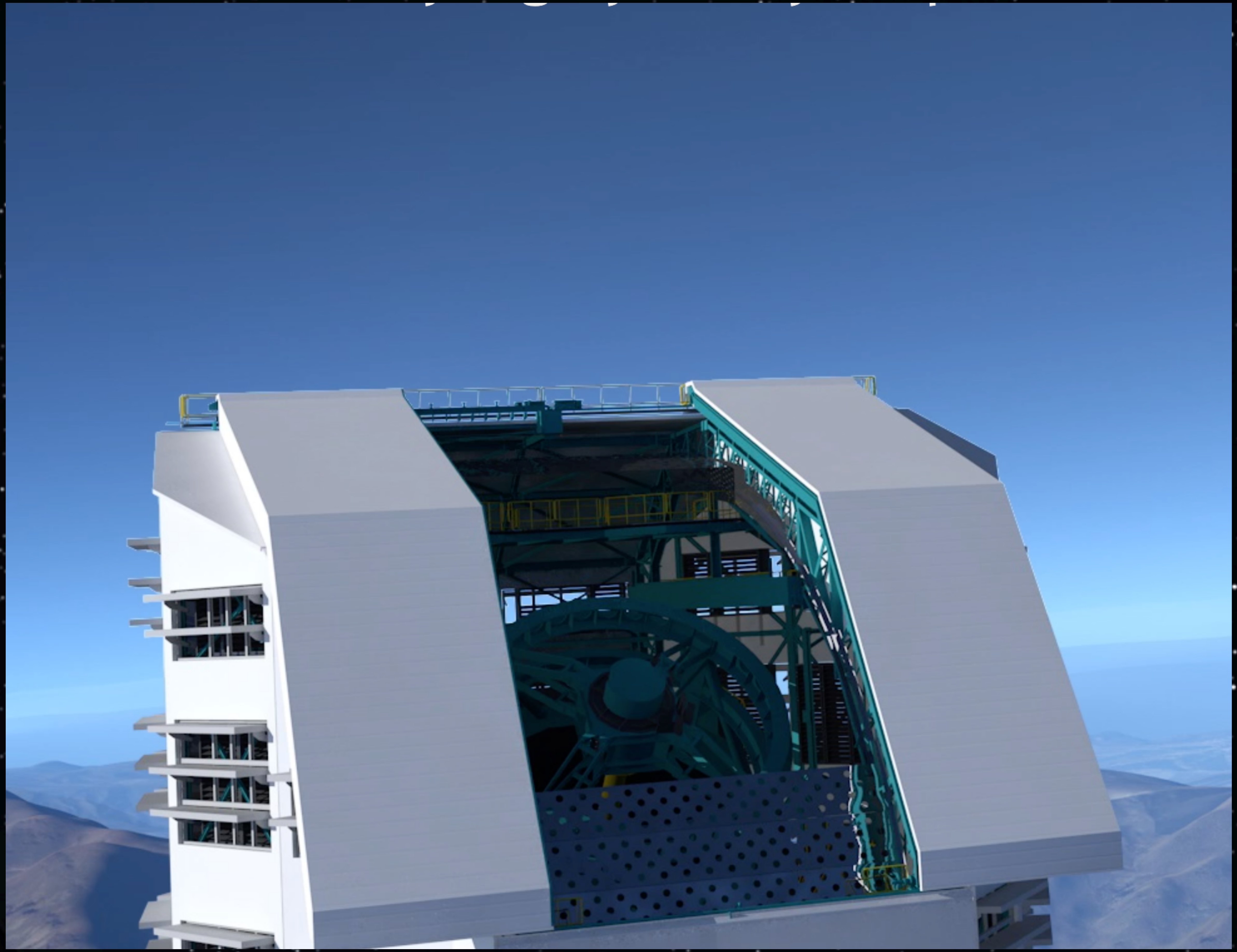
Efficient Deep Learning: A Survey on Making Deep Learning Models Smaller, Faster, and Better, G Menghani
tinyML

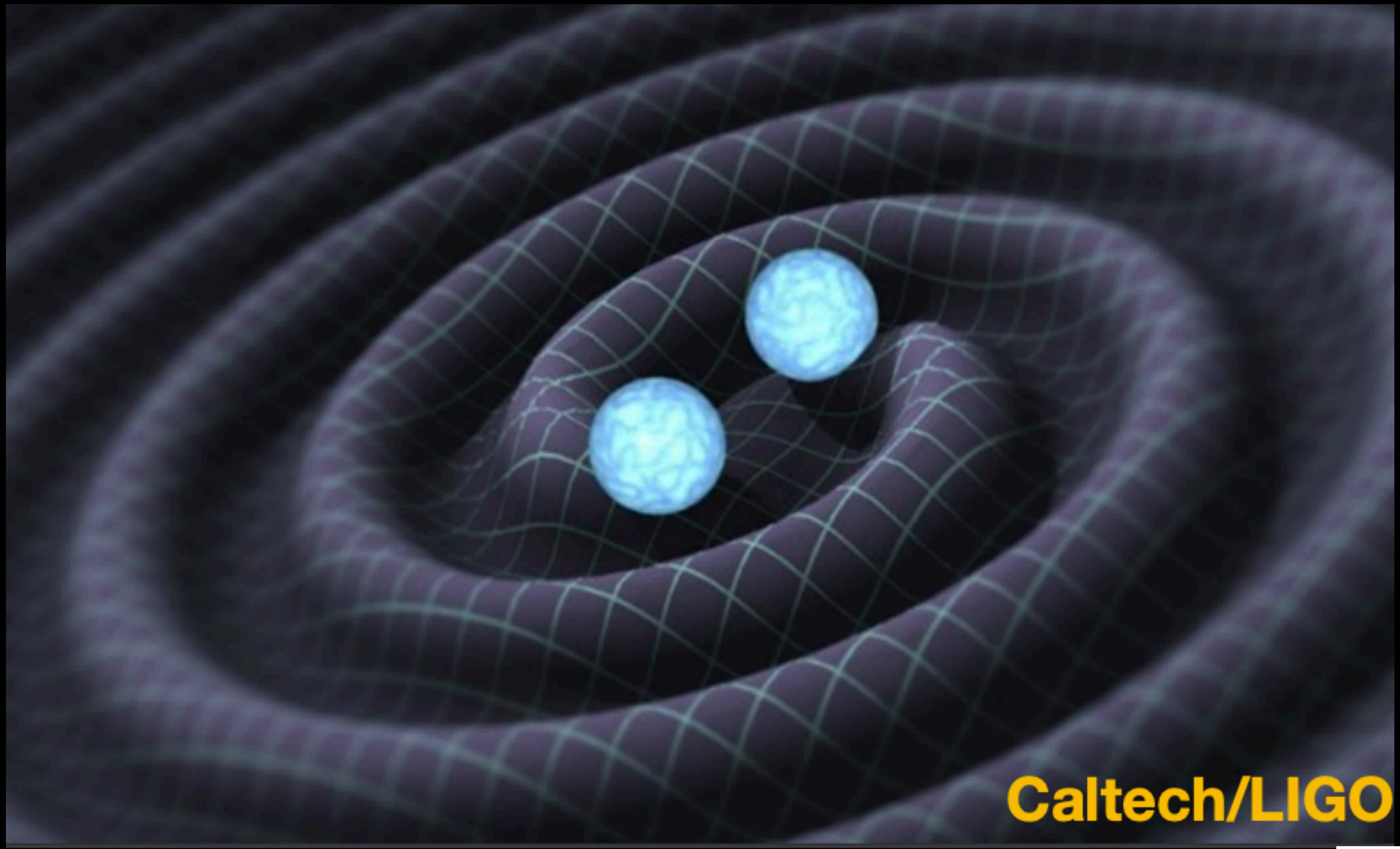
Francois Lanusse

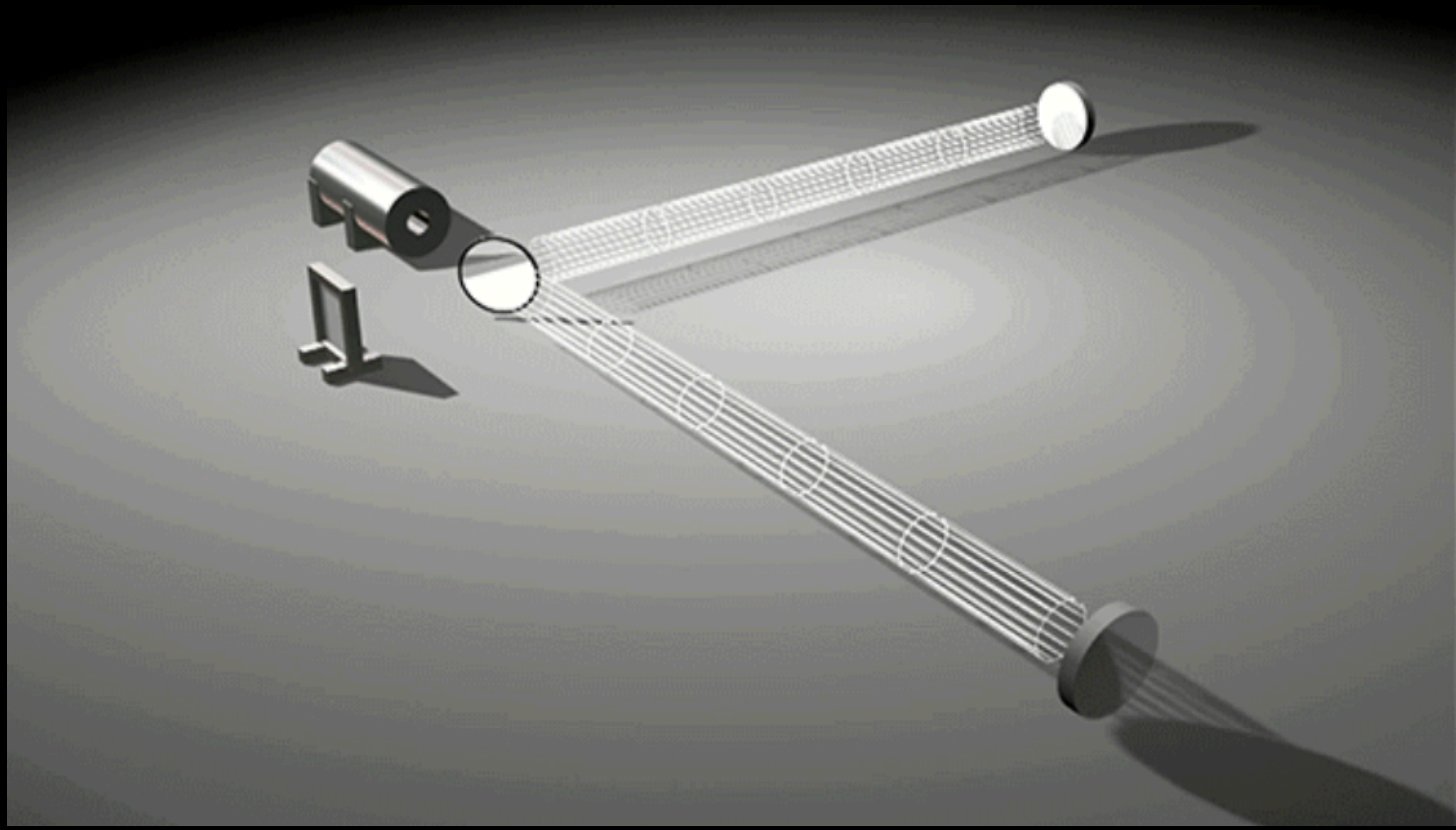
the Rubin Observatory Legacy Survey of Space and Time

- 1000 images each night, 15 TB/night for 10 years
- 18,000 square degrees, observed once every few days
- Tens of billions of objects, each one observed ~ 1000 times

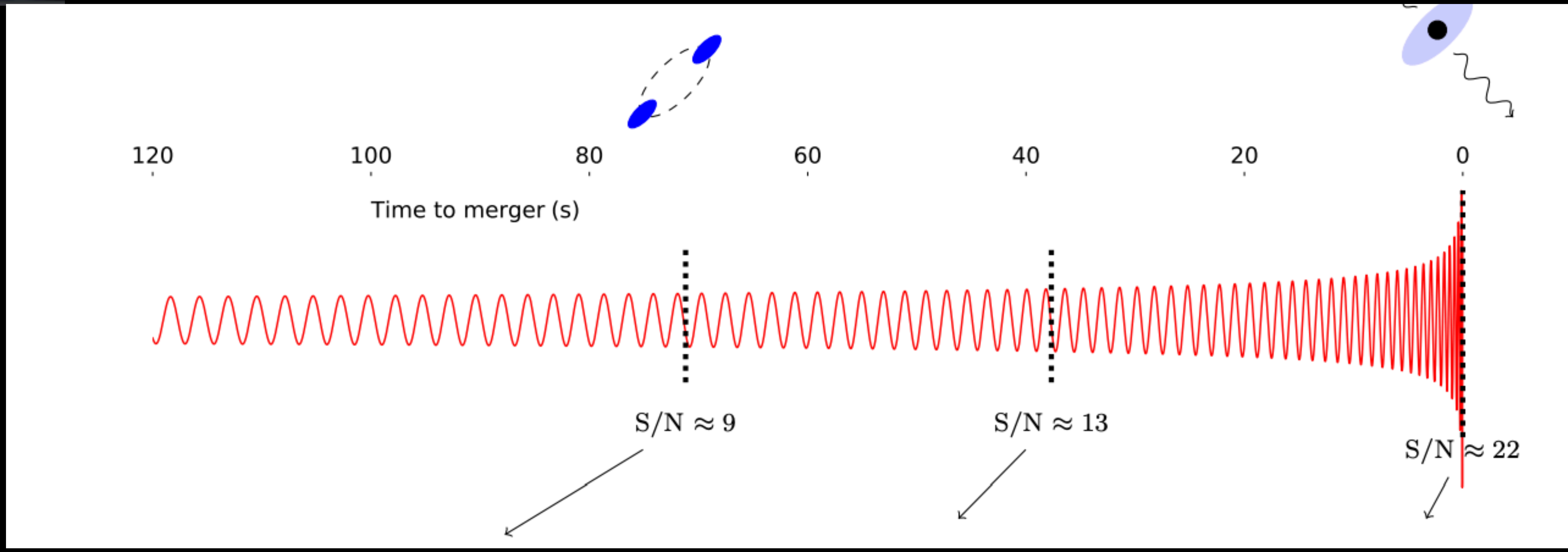
10 million alerts (~20 TB) per night
at ~500Hz inference rate
60 second latency provided by LSST



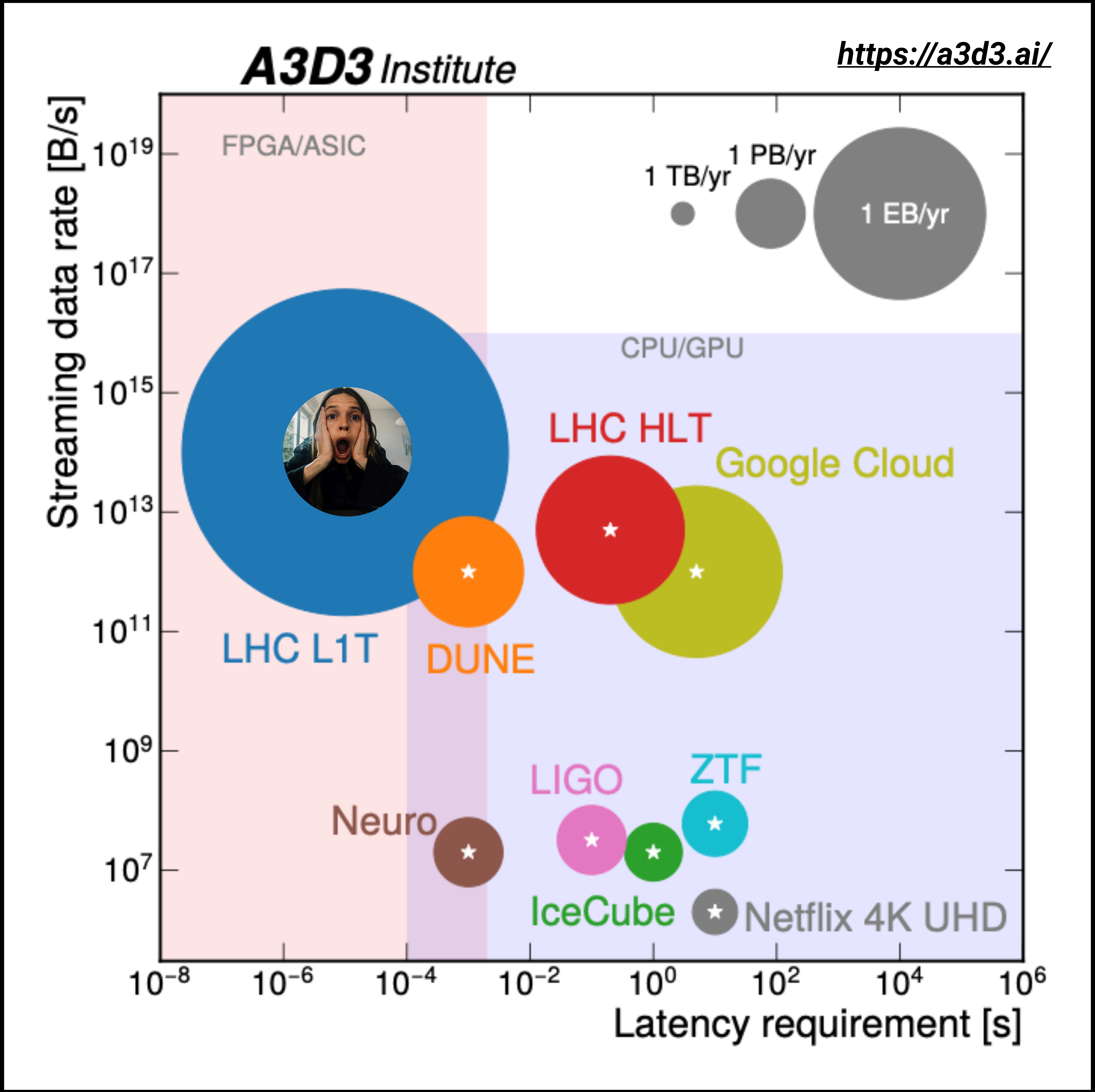




Prompt discovery and inference:
- Pre-merger detection of GWs,
- Rapid parameter estimation,
- Coincident searches with GRB/SNe/neutrinos.



https://arxiv.org/abs/2102.04555



A3D3 Institute
https://a3d3.ai/
Streaming data rate [B/s]
Latency requirement [s]
FPGA/ASIC
CPU/GPU
1 TB/yr
1 PB/yr
1 EB/yr
LHC L1T
DUNE
LHC HLT
Google Cloud
Neuro
LIGO
ZTF
IceCube
Netflix 4K UHD

Geneva
Geneva Lake
LHCb
ATLAS
ALICE
CMS
LHC

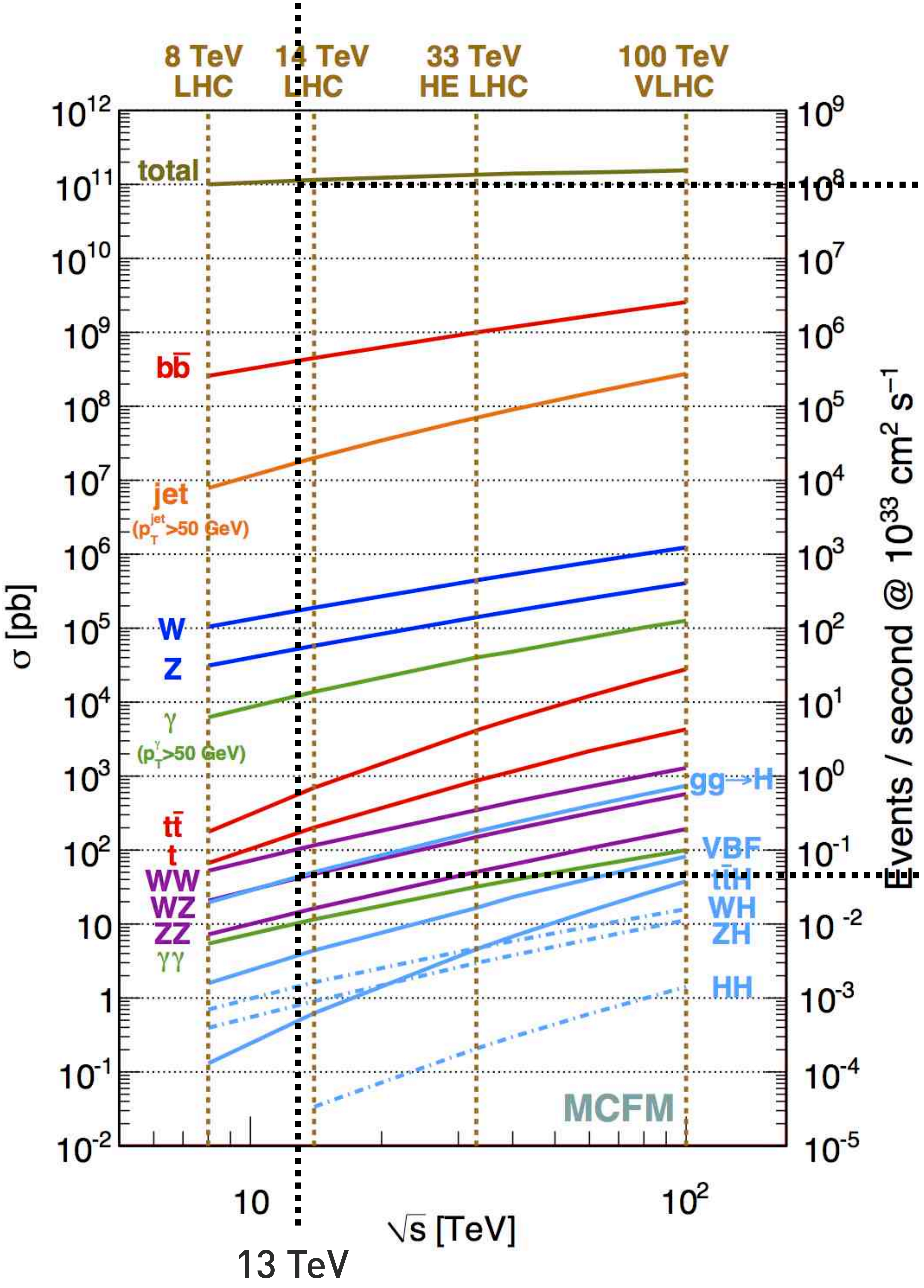

Geneva
Geneva Lake
LHCb
ATLAS
ALICE
CMS
2,500 bunches
10¹¹ protons
11,000 times/s
2.2·10¹¹ protons
25 ns
7.5 m

O(1) billion collisions per second
O(1) PB of data per second

We'd like to read out everything to a big computer farm, but

1) must get all data out from detector
2) must supply detector with much power
3) resolution degradation due to material in detector

SERVICES ALWAYS KILL YOU



"Probability" of producing "anything"

"Probability" of producing a Higgs

Higgs produced
~1 in a billion collisions!

Saving all collisions not useful
(even if we could)!

Geneva
Geneva Lake
LHCb
ATLAS
ALICE
CMS
Software rate reduction
(GPU+CPU)
2 step rate reduction
(hardware+software)
Continous read-out
(CPU+GPU)
2 step rate reduction
(hardware+software)
LHC

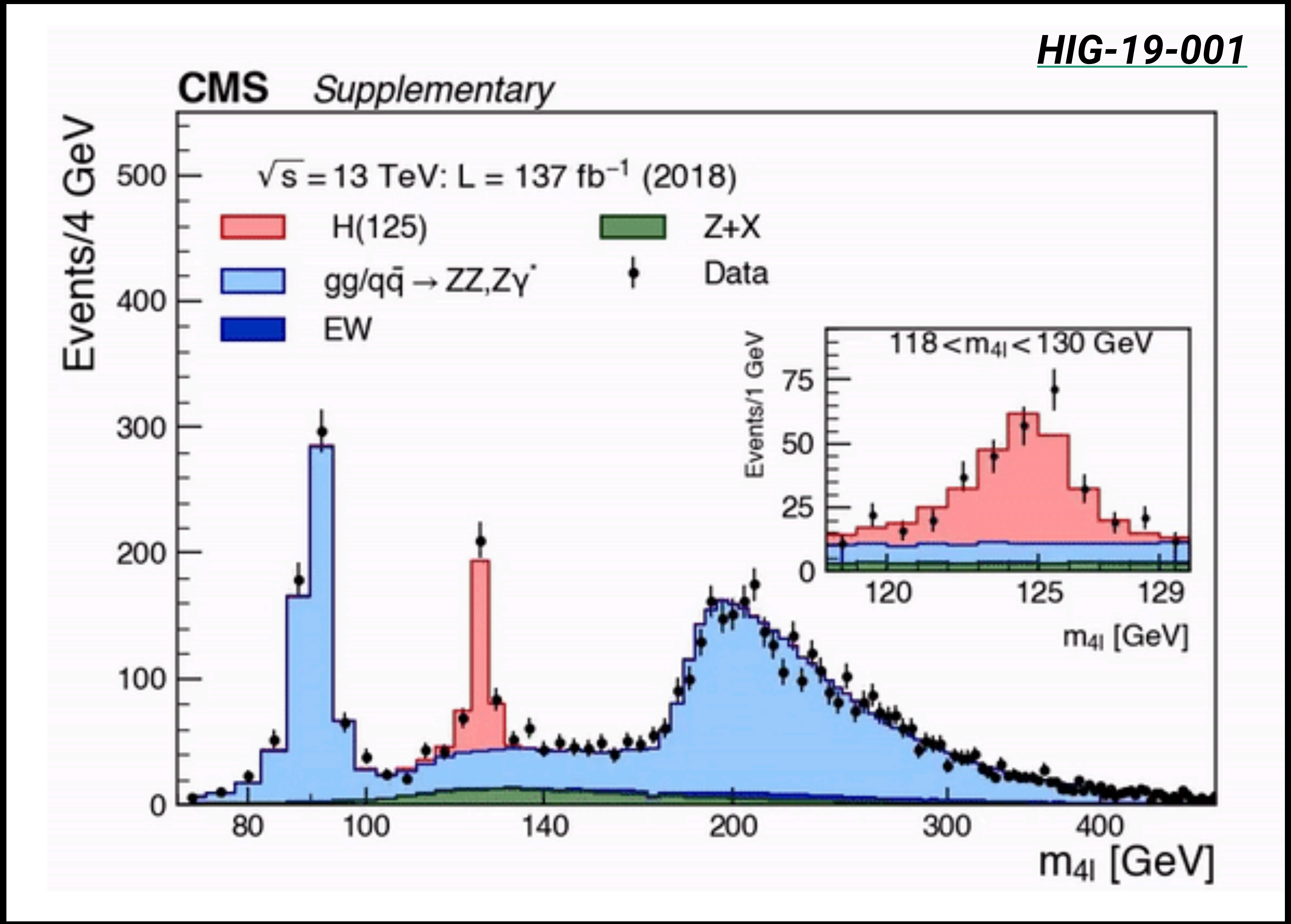

Geneva
Geneva Lake
LHCb
ATLAS
ALICE
CMS
2 step rate reduction
(hardware+software)
LHC


Geneva
Geneva Lake
LHCb
CMS
2 Tb/s to L1
Data temporarily stored
in detector electronics for 4 μs
(frontend buffering limit)
L1 trigger:
(~1000 FPGAs)
Decide which
event to keep
within ~4 μs
Reject >99% of
collisions!
LHC


Geneva
Geneva Lake
LHCb
CMS
LHC
L1 bit:
Accept = 1
Reject = 0


High Level Trigger:
25’600 CPUs / 400 GPUs
Latency: 3-400 ms
Reject further 99%!
eneva
ATLAS
ALICE
L1 accept:
O(100) kHz
~Tb/s
LHC

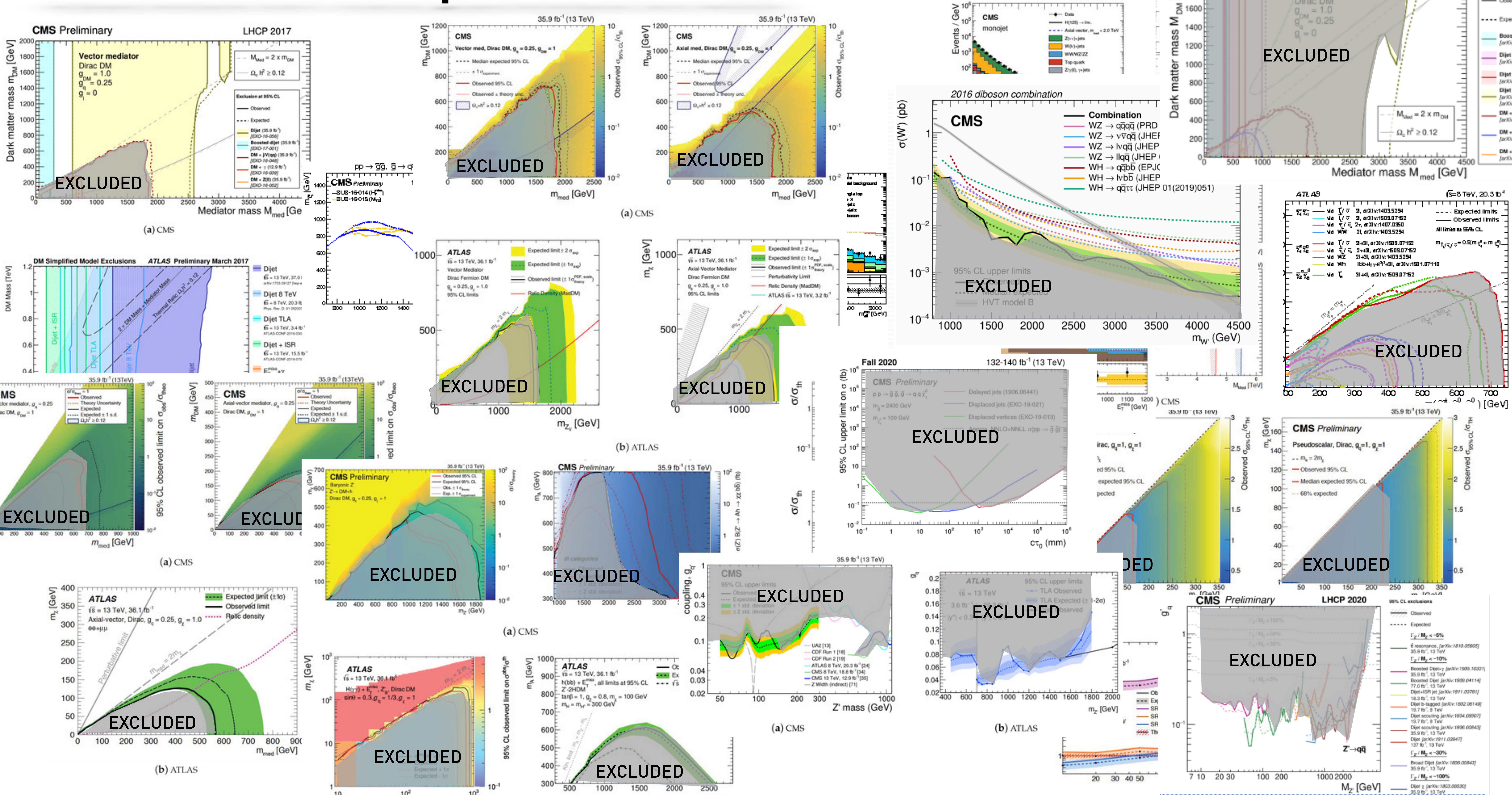

TIER 0: ∞
0.0025% of collision events remaining
High Level Trigger:
Latency O(100) ms
Geneva
LHC
ATLAS
CMS
HLT accept:
O(1) kHz
~Gb/s
L1 accept:
O(100) kHz
~Tb/s
LHC



To make sure we select "the right" 0.0025%, algorithms must be

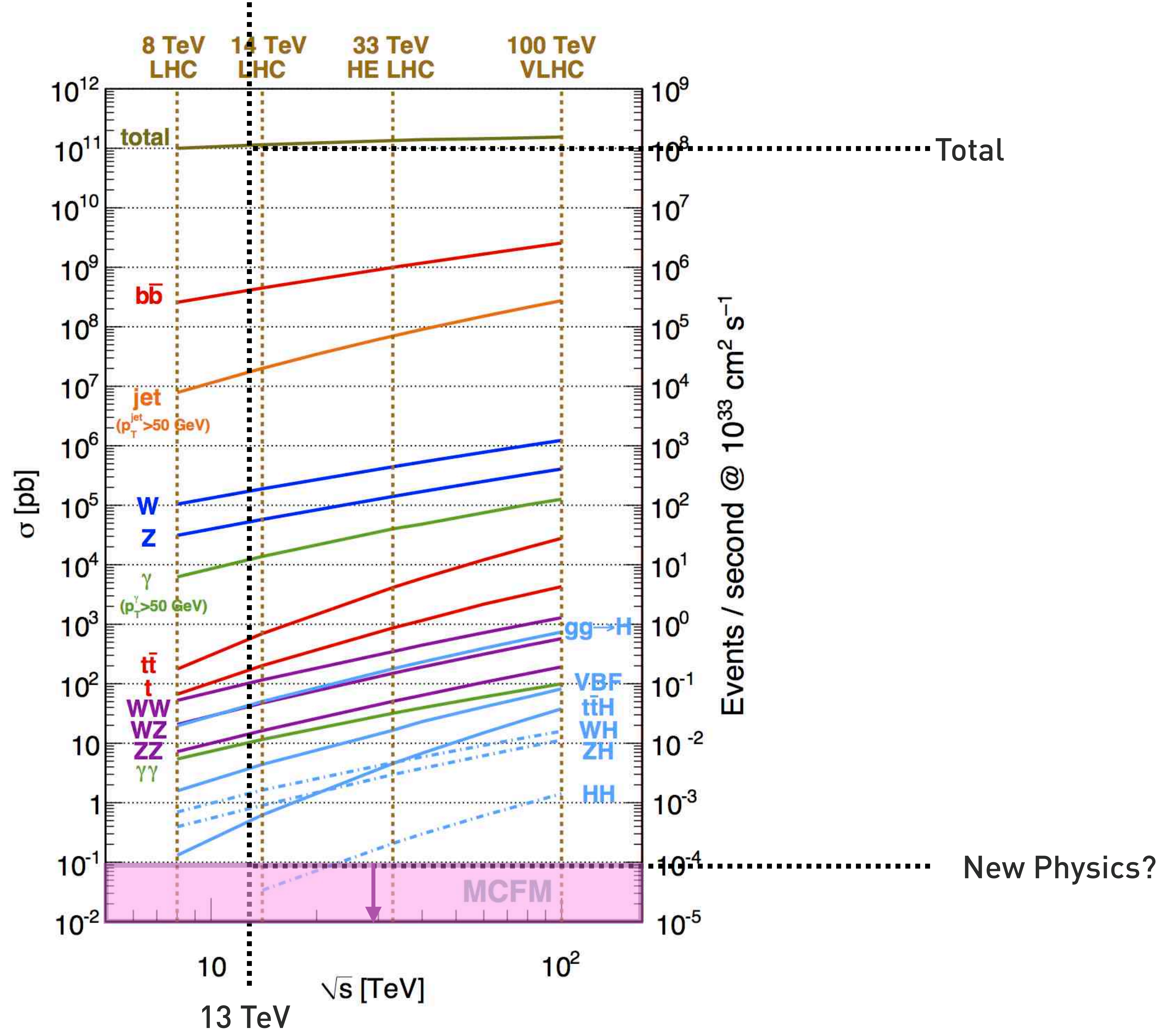
- Fast (get more data through)
- Accurate (select the right data)

Searches for new particles at LHC



New Physics is produced less than 1 in a trillion (if at all)

Need **more** data!



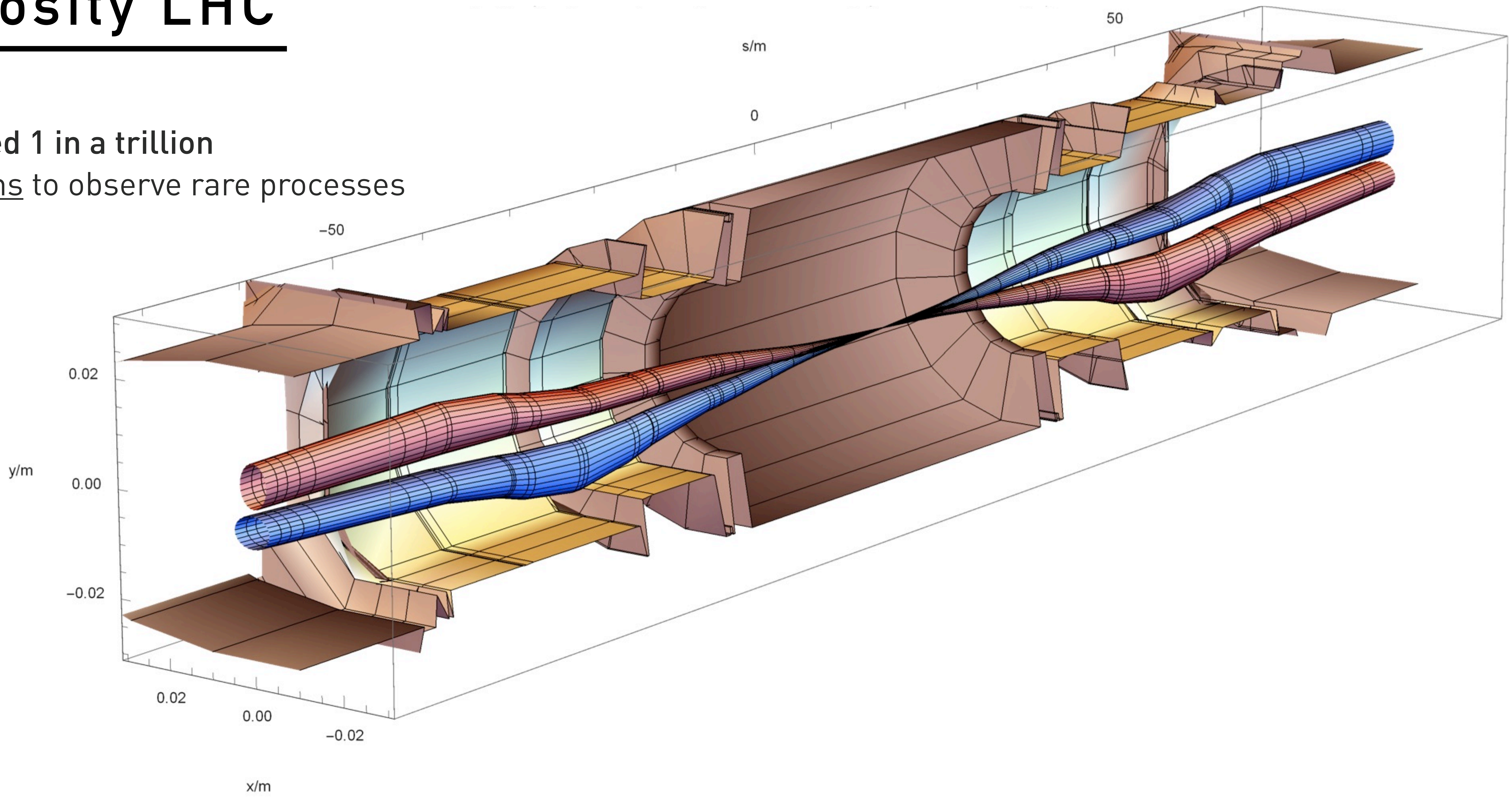
High Luminosity LHC

New Physics is produced 1 in a trillion

- Need more collisions to observe rare processes

High Luminosity LHC

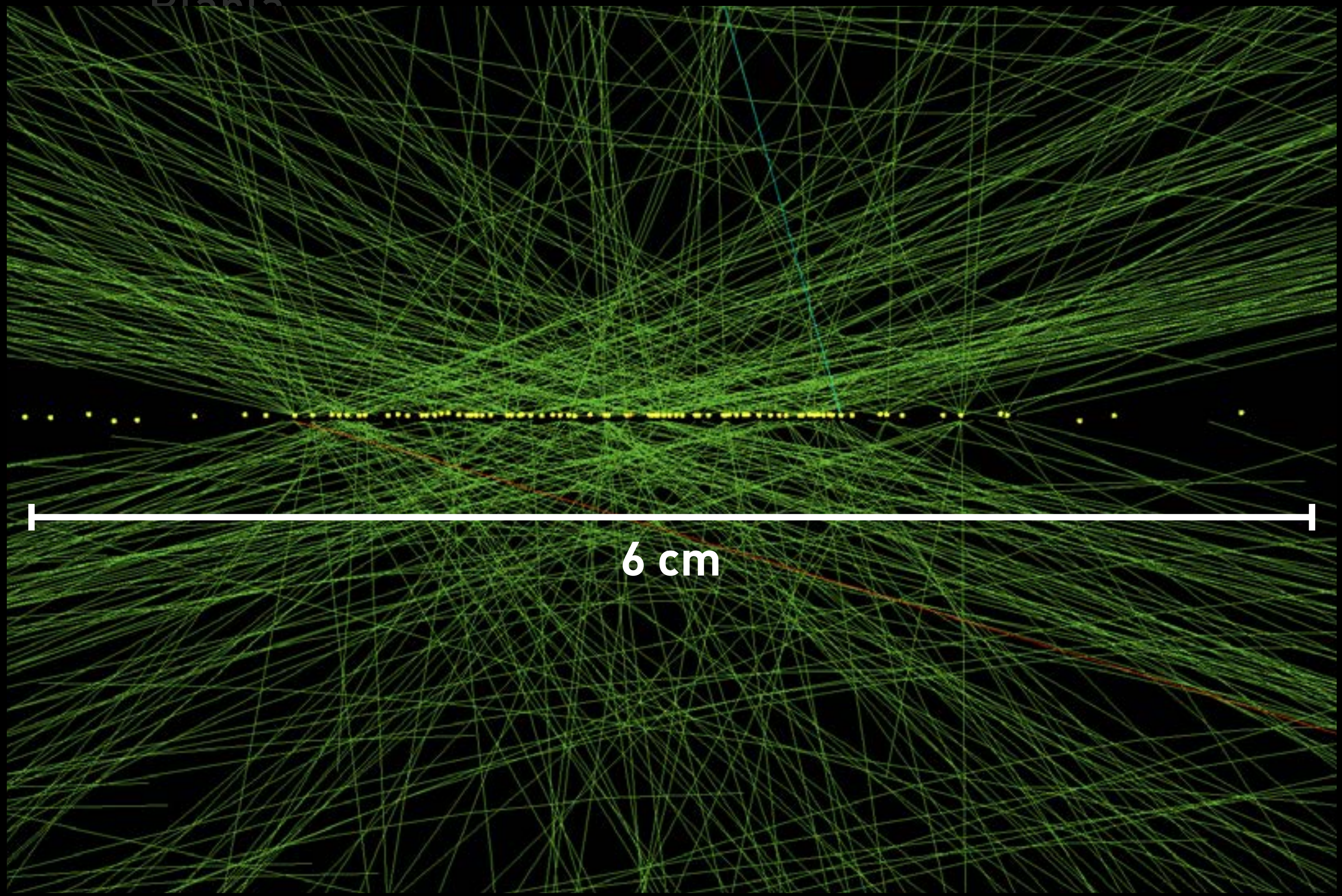
- x10 data size
- x3 collisions/s



2022 - 2025	2026 - 2028	2029 - 2038
LHC (TODAY!)	MAJOR UPGRADE	HL-LHC
Run 3		Run 4+5

LHC

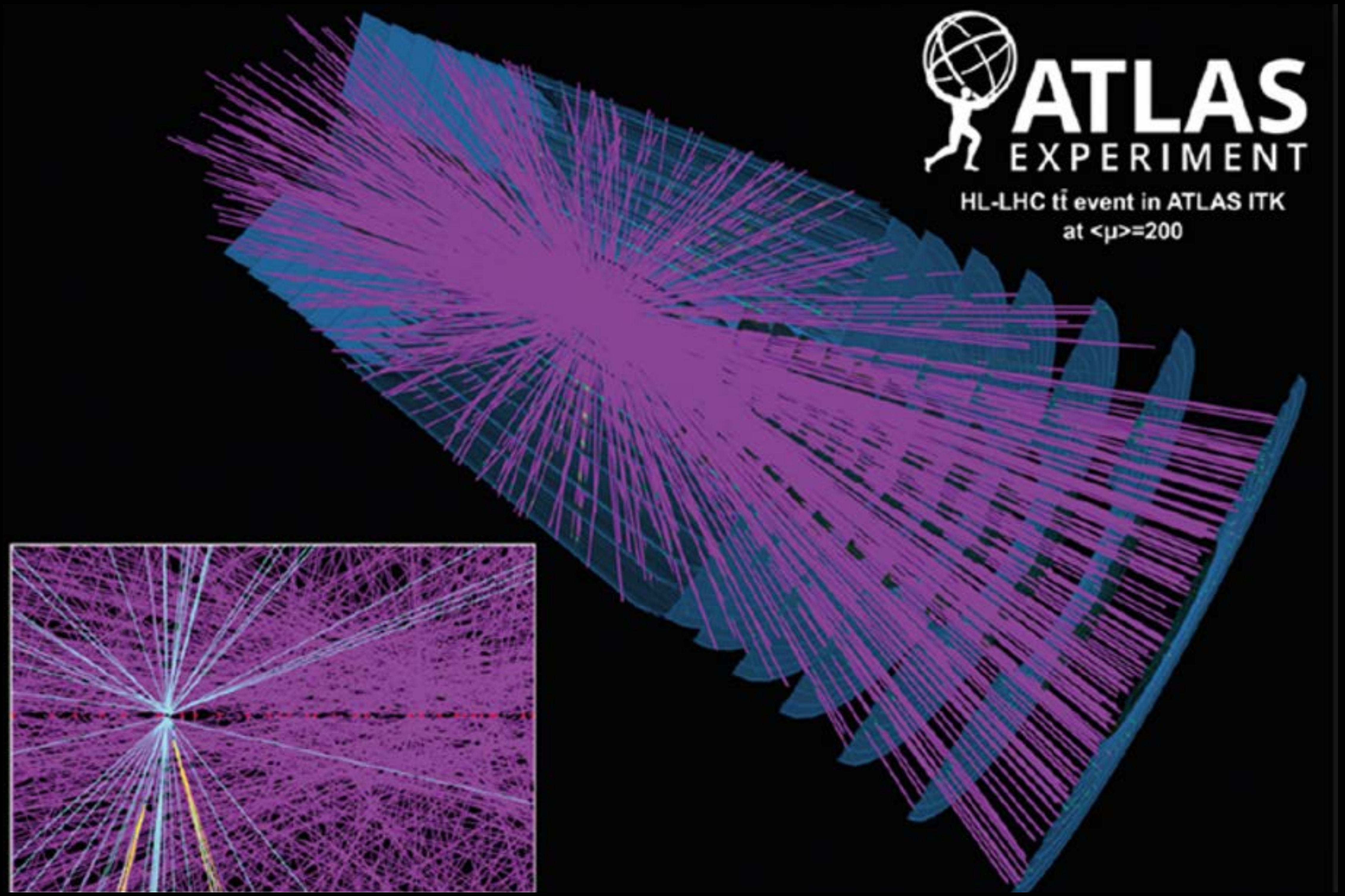
78 vertices
(average 60)



Run 3

High Luminosity LHC

200 vertices
(average 140)



Run 4+5

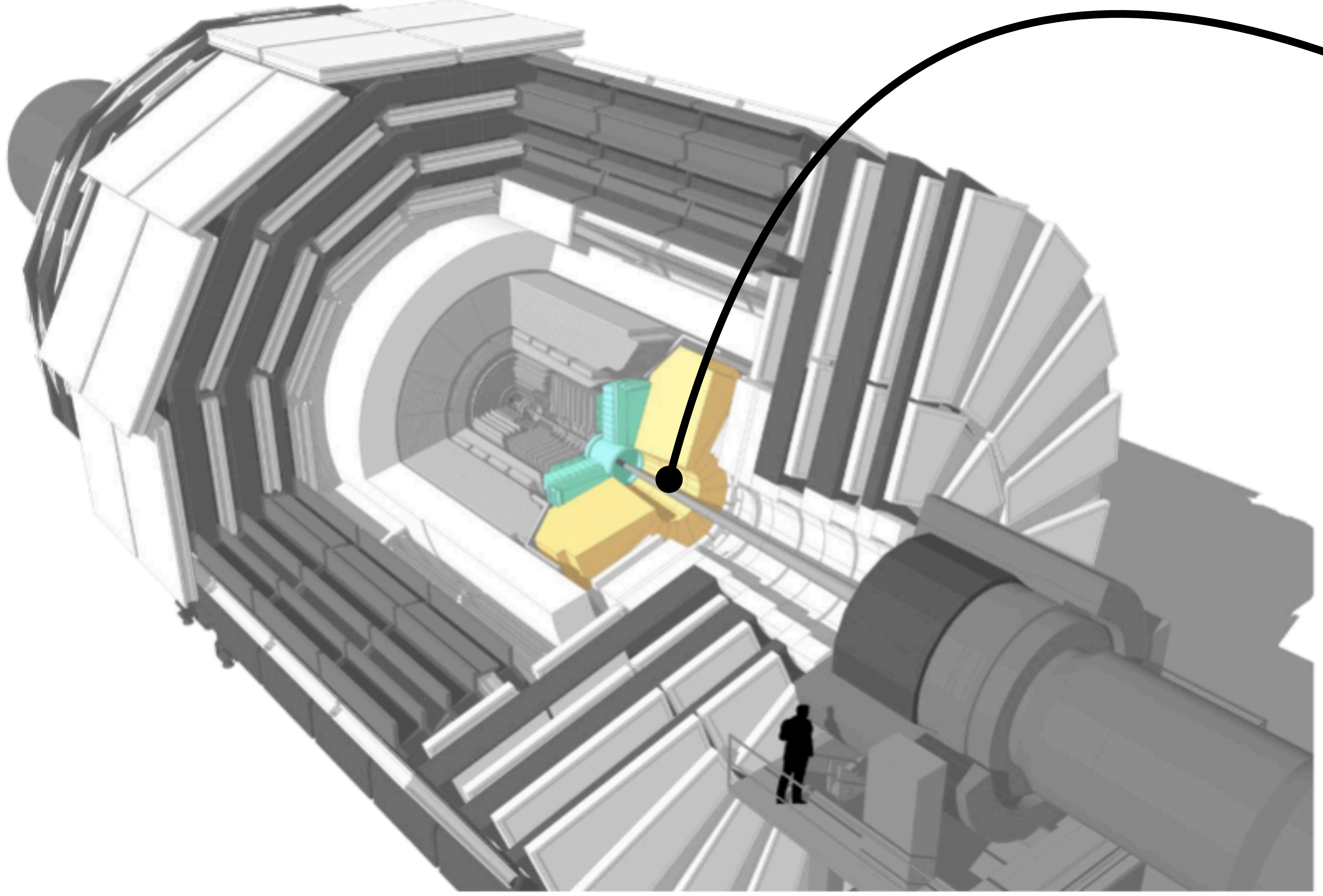
Maintain physics acceptance → better detectors

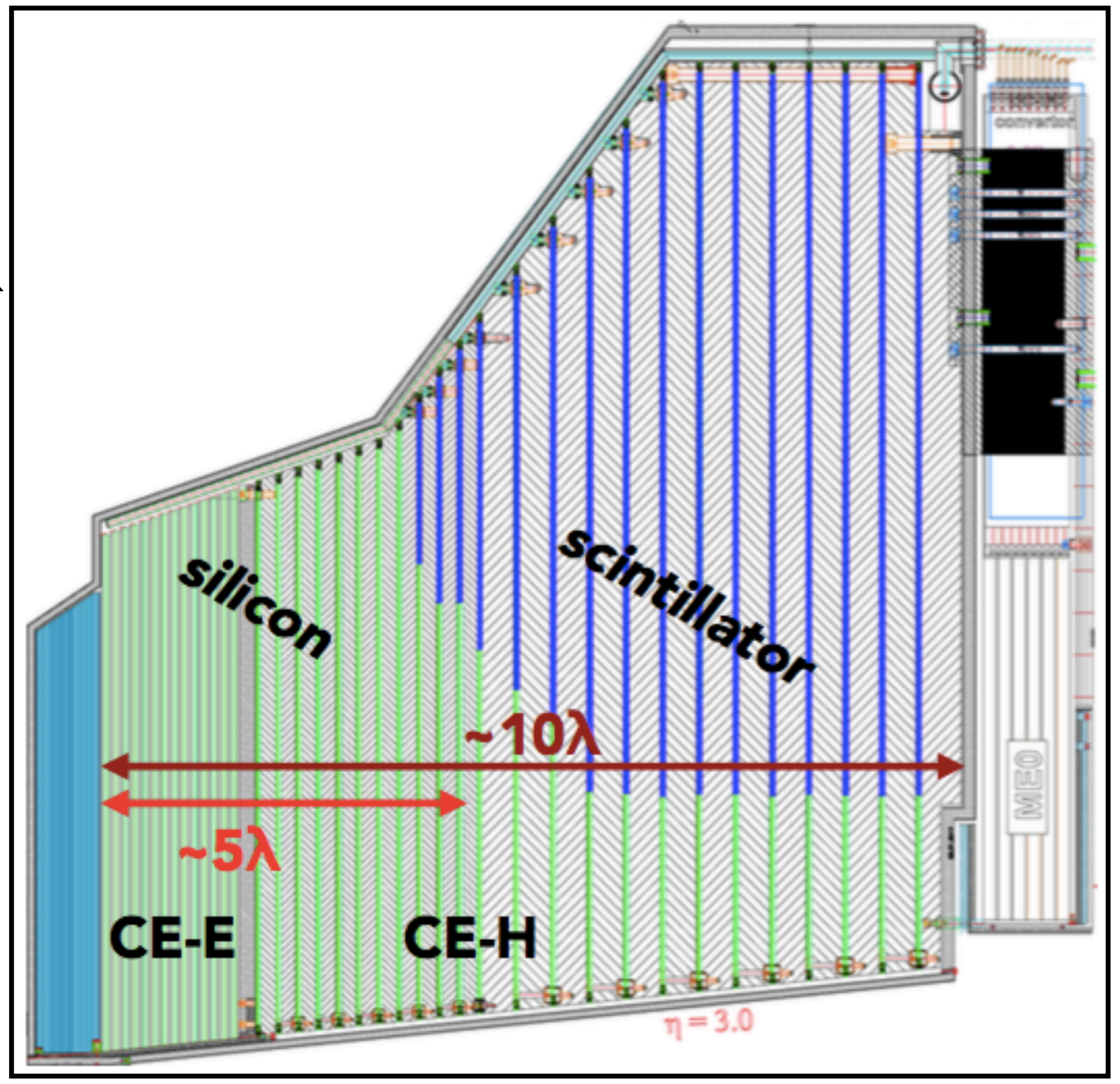
CMS High Granularity (endcap) calorimeter

- 85K (today) → 6M (HL-LHC) readout channels

More collisions

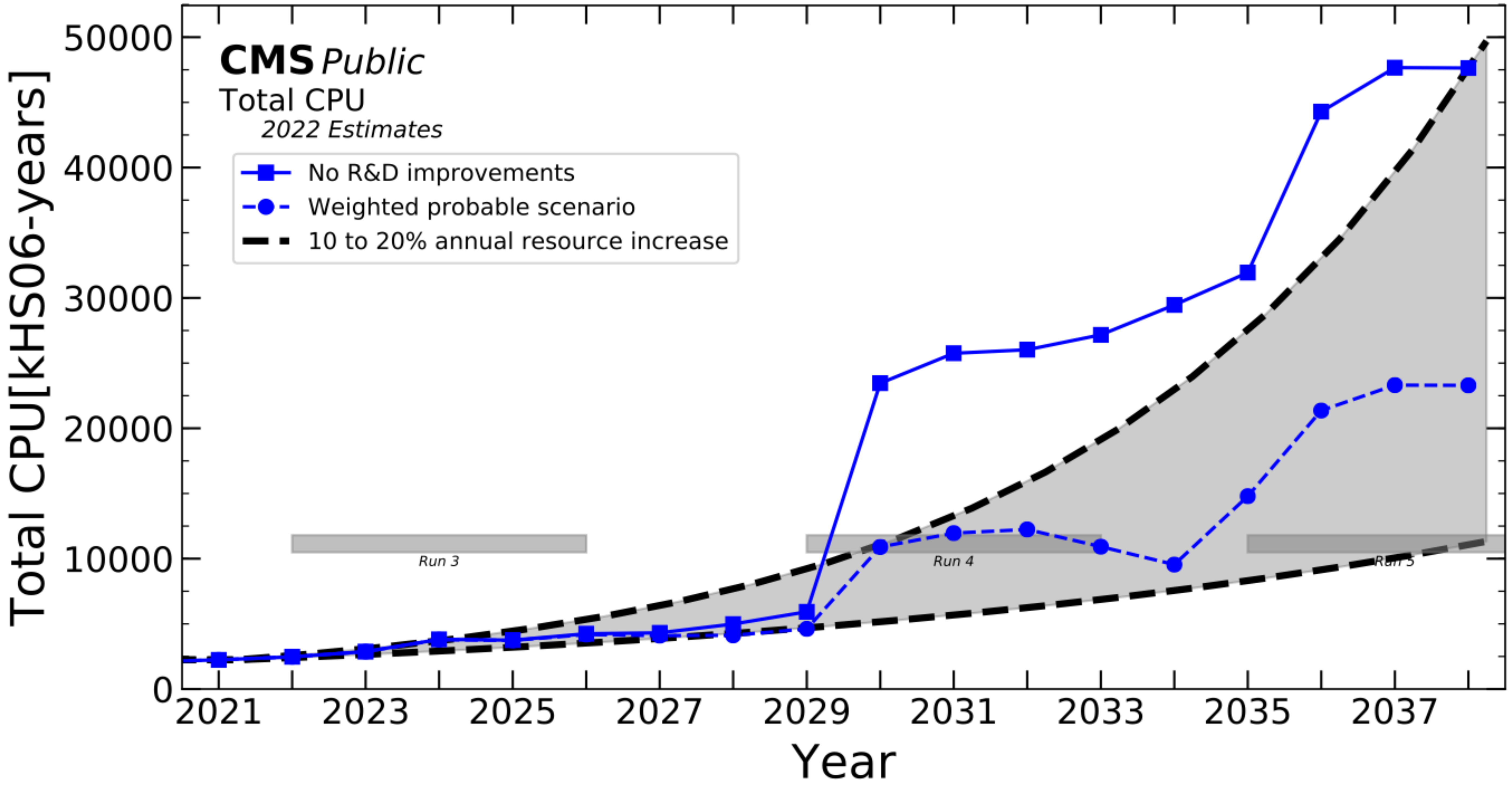
More readout channels

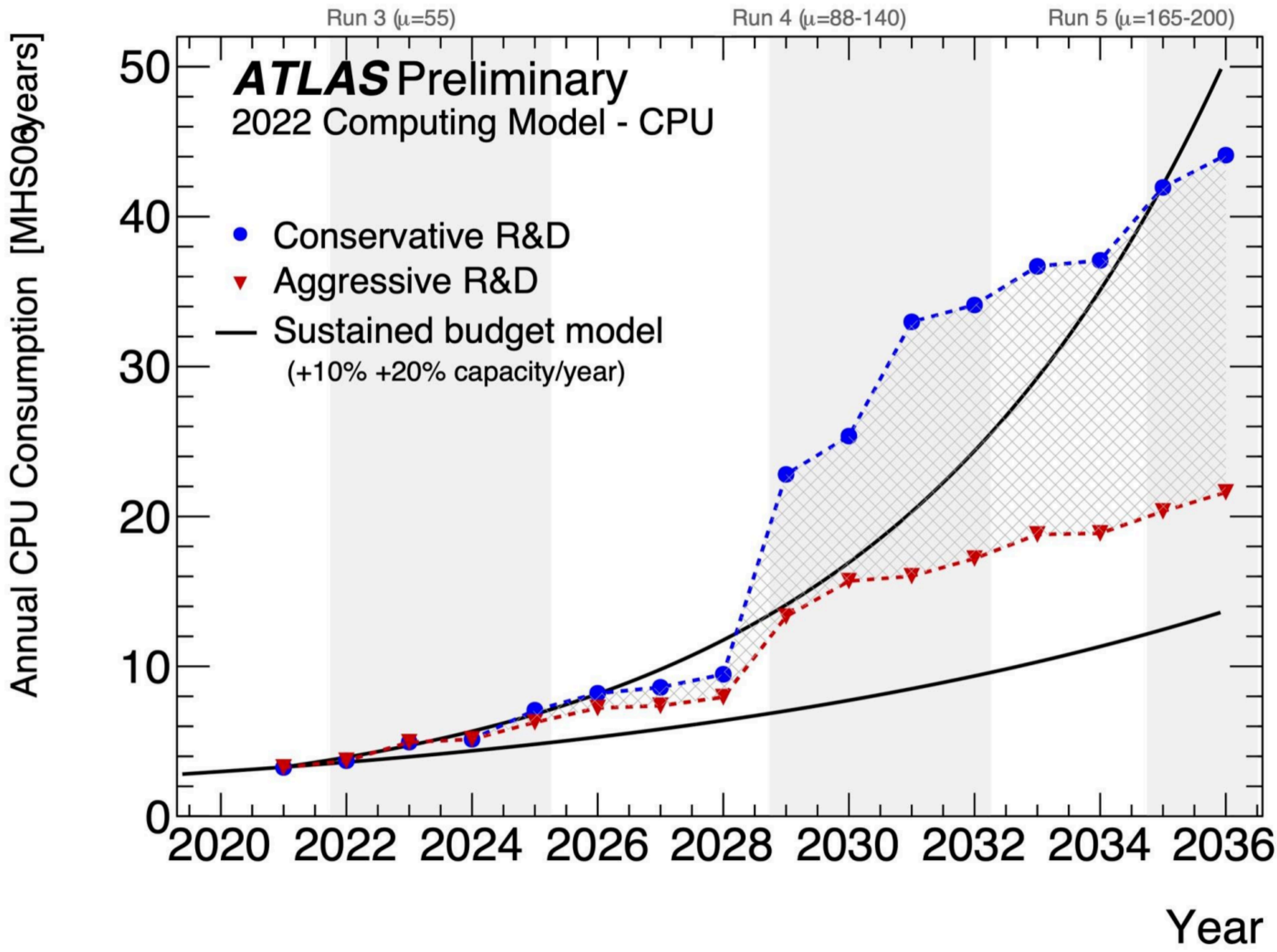




CMS HGCAL TDR

Computing resources





... **flat** computing budget

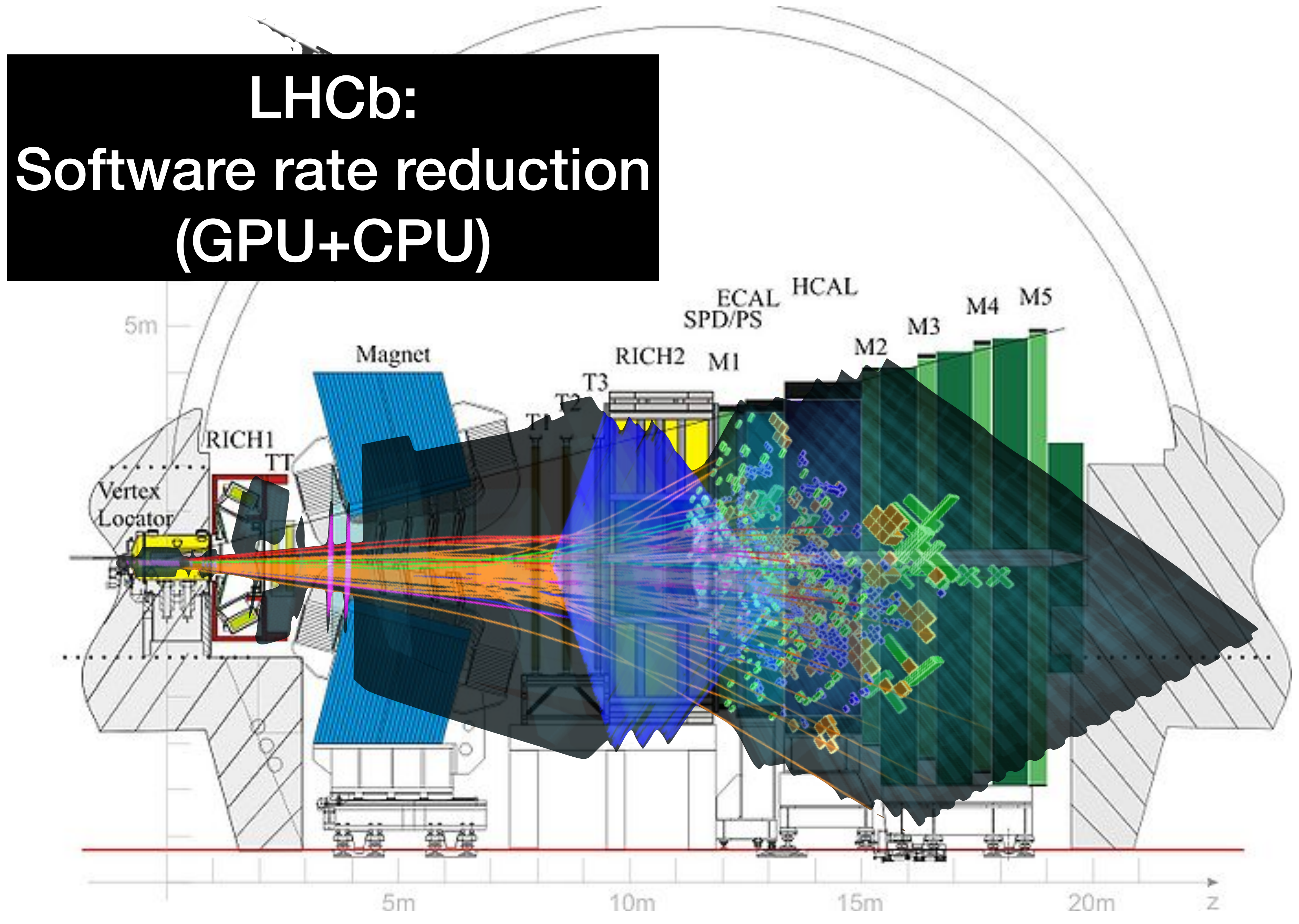

Geneva
Geneva Lake
LHCb
CMS
LHC
HL-LHC L1:
• Tracking
• Particle Flow
• HGCal
Input data
• 2 Tb/s → **63 Tb/s**
Latency
• 4 μs → **12 μs**

How can we use fast ML to cope with increased data complexity?


Geneva
Geneva
LHCb
ATLAS
ALICE
GPU inference
HLT trigger:
Latency O(100) ms
Fast inference on specialised hardware
ASIC inference
FPGA inference
Level-1 trigger:
Latency O(1) μs
Detector:
40 MHz
~Pb/s

4T
2T
Silicon Tracker
Electromagnetic Calorimeter
Hadron Calorimeter
63 Tb/s
return yoke interspersed
with Muon chambers
UXC55
USC55
CALORIMETRY:
370 FPGAs
*54 for HGCAL only!
Xilinx Ultrascale+ FPGAs
TRACKING
174 FPGAs
MUONS:
96 FPGAs
5 μs
PARTICLE FLOW:
66 FPGAs
GLOBAL TRIGGER:
24 FPGAs
EXTERNAL TRIGGERS
CALORIMETRY
PARTICLE FLOW
MUONS
Trigger accept/reject
12.5 μs

Why FPGAs?

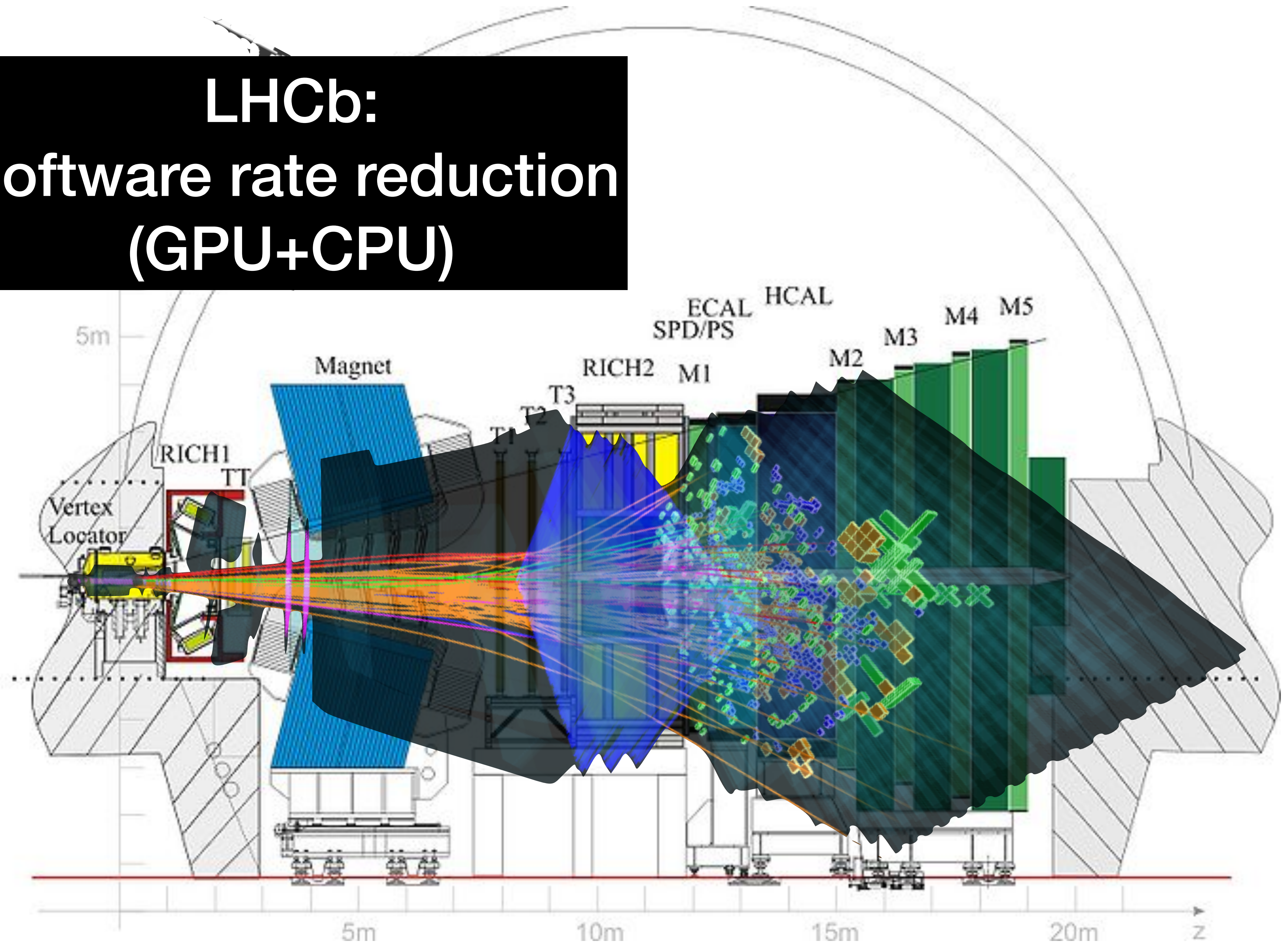


Full GPU reconstruction @ 4 TB/s

- 326 GPUs, 60 kHz per GPU

Why FPGAs?

Depends on your problem...



Full GPU reconstruction @ 4 TB/s
- 326 GPUs, 60 kHz per GPU

Characteristics of LHCb HLT1	**Characteristics of GPUs** ***C. Fitzpatrick***
Intrinsically parallel problem: - Run events in parallel - Reconstruct tracks in parallel	Good for - Data-intensive parallelizable applications - High throughput applications
Huge compute load	Many TFLOPS
Full data stream from all detectors is read out → no stringent latency requirements	Higher latency than CPUs, not as predictable as FPGAs
Small raw event data (~100 kB)	Connection via PCIe → limited I/O bandwidth
Small event raw data (~100 kB)	Thousands of events fit into O(10) GB of memory

→ LHCb has already read out detector
→ CMS frontend buffers strictly limited, cannot tolerate latency slack
→ CMS raw event data x10 larger, L1 “event” ~ 200 kB (possible with GPU)

Why FPGAs?

Latency, latency, latency (cannot do much on a GPU IN 4 μs)

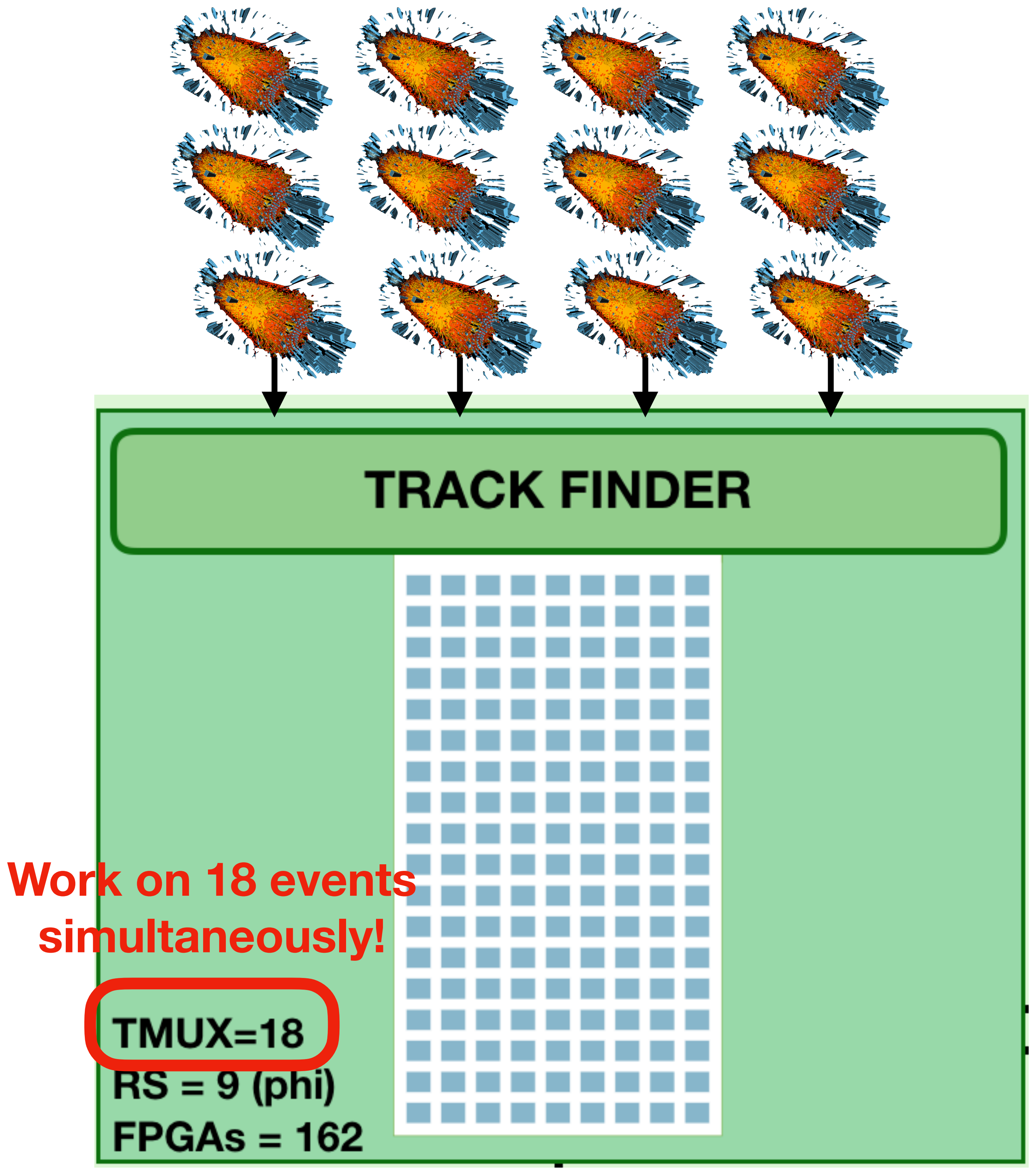
- Can work on different parts of problem, different data simultaneously
- Latency strictly limited by detector frontend buffer

Latency deterministic

- CPU/GPU processing randomness, FPGAs repeatable and predictable latency

High bandwidth

- L1T processes 5% of total internet traffic, dissipate heat of ~7W/cm^2



Programming an FPGA

C/C++ algorithm

Constraints/ Directives

Xilinx

Intel

Mentor

Vitis HLS

Vivado HLS

Vivado Accelerator

Intel HLS

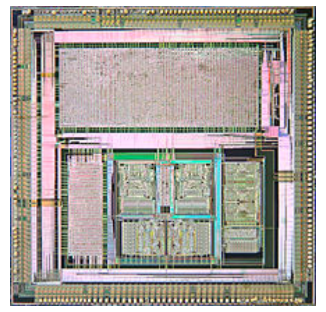
Catapult HLS

(CMS, ATLAS)

VHDL/Verilog

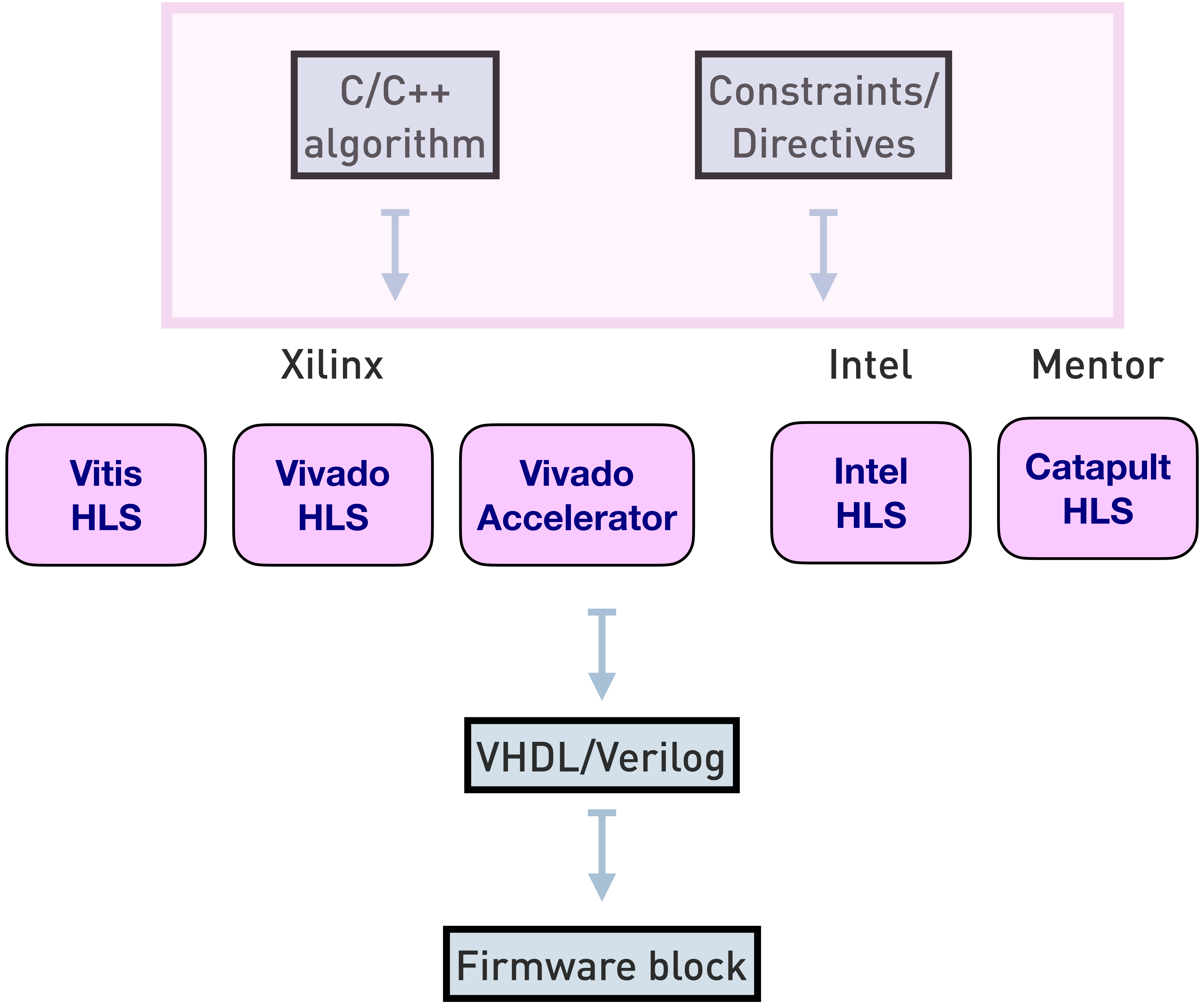
Firmware block

(LHCb, ATLAS)



(ASICs)

Programming an FPGA



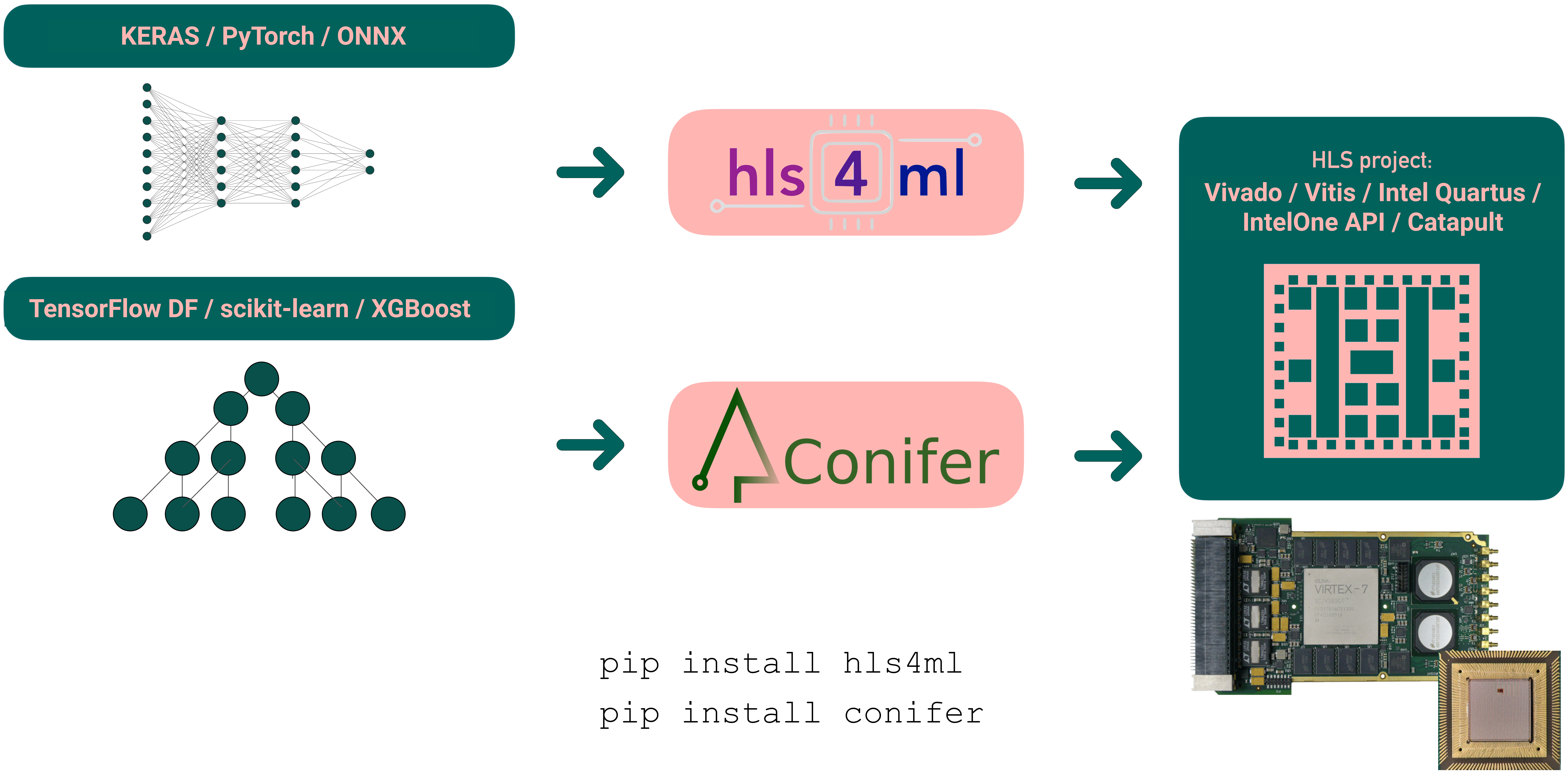
```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;

use work.gtl_pkg.all;

entity invariant_mass is
    generic (
        upper_limit: real := 15.0;
        lower_limit: real := 10.0;
        pt1_width: positive := 12;
        pt2_width: positive := 12;
        cosh_cos_width: positive := 28;
        INV_MASS_PRECISION : positive := 1;
        INV_MASS_COSH_COS_PRECISION : positive := 3
    );
    port(
        pt1 : in std_logic_vector(pt1_width-1 downto 0);
        pt2 : in std_logic_vector(pt2_width-1 downto 0);
        cosh_deta : in std_logic_vector(cosh_cos_width-1 downto 0);  -- cosh of eta1 - eta2
        cos_dphi : in std_logic_vector(cosh_cos_width-1 downto 0);   -- cos of phi1 - phi2
        inv_mass_comp : out std_logic;
        sim_inv_mass_sq_div2 : out std_logic_vector(pt1_width+pt2_width+cosh_cos_width-1 downto 0)
    );
end invariant_mass;

architecture rtl of invariant_mass is

  constant INV_MASS_VECTOR_WIDTH : positive := pt1_width+pt2_width+cosh_cos_width;
  constant INV_MASS_PRECISION_FACTOR : real := real(10**INV_MASS_PRECISION);.pkg.
  constant FACTOR_4_VECTOR : std_logic_vector((INV_MASS_COSH_COS_PRECISION+1)*4-1 downto 0) := conv_std_logic_vector(10**(INV_MASS_COSH_COS_PRECISION+1),(INV_MAS

  signal inv_mass_sq_div2   : std_logic_vector(INV_MASS_VECTOR_WIDTH-1 downto 0);
  signal upper_limit_vector : std_logic_vector(INV_MASS_VECTOR_WIDTH-1 downto 0);
  signal lower_limit_vector : std_logic_vector(INV_MASS_VECTOR_WIDTH-1 downto 0);

  begin

    -- Converting the boundary value for the comparison
    upper_limit_vector <= conv_std_logic_vector((integer(upper_limit*INV_MASS_PRECISION_FACTOR)),INV_MASS_VECTOR_WIDTH-FACTOR_4_VECTOR'length)*FACTOR_4_VECTOR;
    lower_limit_vector <= conv_std_logic_vector((integer(lower_limit*INV_MASS_PRECISION_FACTOR)),INV_MASS_VECTOR_WIDTH-FACTOR_4_VECTOR'length)*FACTOR_4_VECTOR;

    -- Calculation of invariant mass with the formula: M**2/2 = pt1*pt2 * (cosh(eta1 - eta2) - cos(phi1 - phi2))
    inv_mass_sq_div2 <= pt1 * pt2 * (cosh_deta - cos_dphi);
    sim_inv_mass_sq_div2 <= inv_mass_sq_div2;

    -- Comparison with boundary values
    inv_mass_comp <= '1' when (inv_mass_sq_div2 >= lower_limit_vector and inv_mass_sq_div2 <= upper_limit_vector) else '0';

end architecture rtl;
```

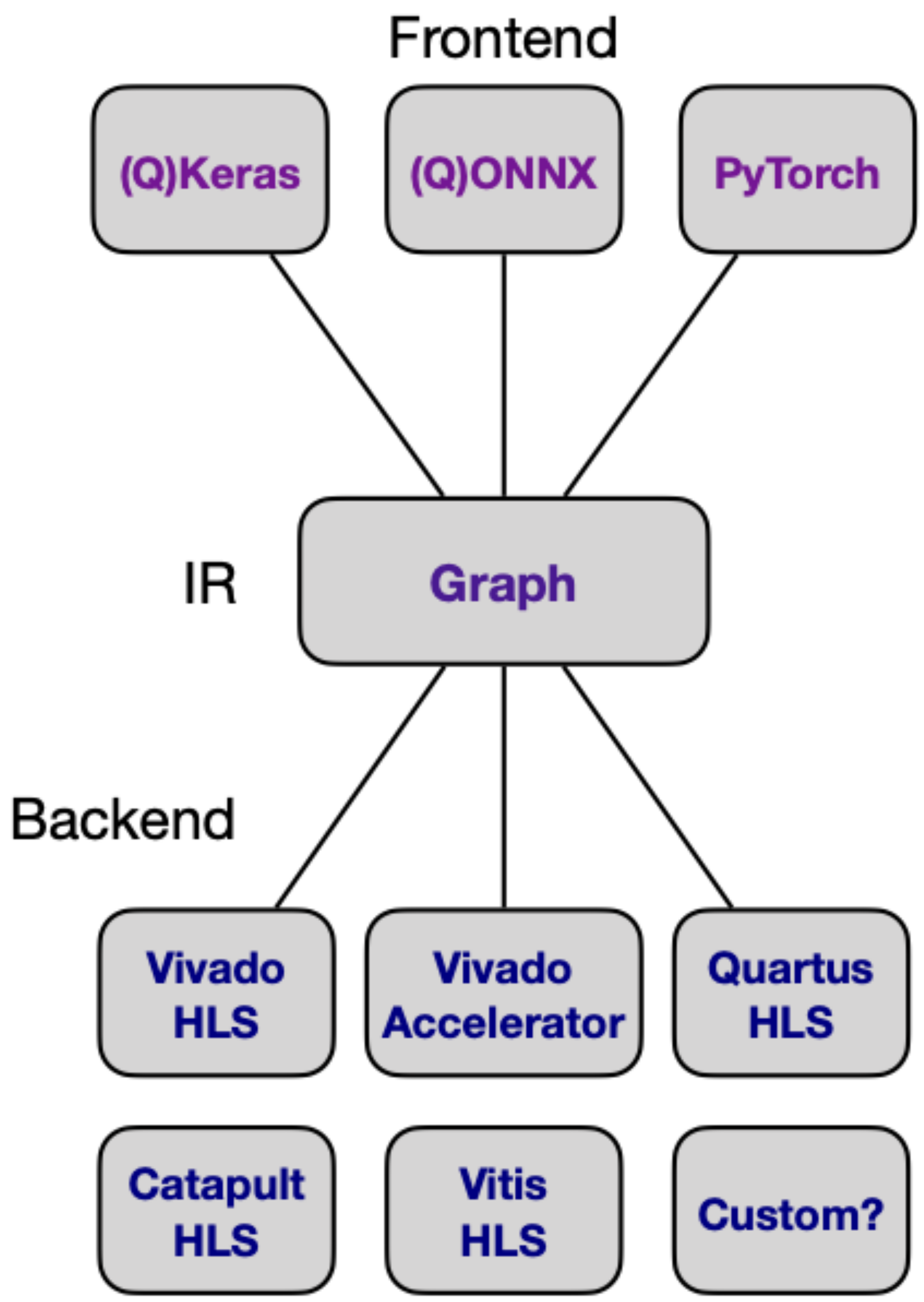
$$\mathbf{x}_n = g_n(\mathbf{W}_{n,n-1}\mathbf{x}_{n-1} + \mathbf{b}_n)$$

Generic (superfast) HLS implementations for DNN inference?

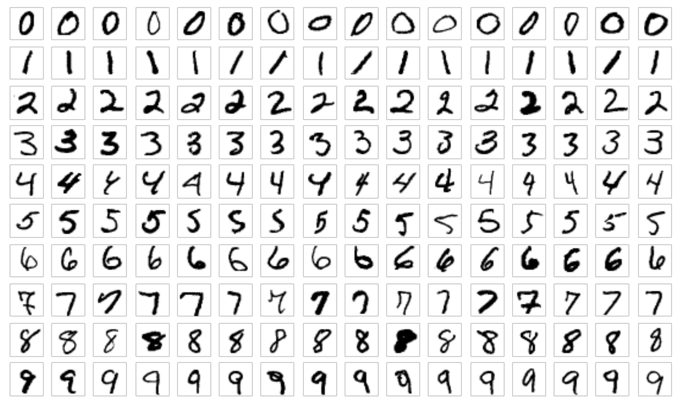
```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;

use work.gtl_pkg.all;

entity invariant_mass is
    generic (
        upper_limit: real := 15.0;
        lower_limit: real := 10.0;
        pt1_width: positive := 12;
        pt2_width: positive := 12;
        cosh_cos_width: positive := 28;
        INV_MASS_PRECISION : positive := 1;
        INV_MASS_COSH_COS_PRECISION : positive := 3
    );
    port(
        pt1 : in std_logic_vector(pt1_width-1 downto 0);
        pt2 : in std_logic_vector(pt2_width-1 downto 0);
        cosh_deta : in std_logic_vector(cosh_cos_width-1 downto 0);  -- cosh of eta1 - eta2
        cos_dphi : in std_logic_vector(cosh_cos_width-1 downto 0);   -- cos of phi1 - phi2
        inv_mass_comp : out std_logic;
        sim_inv_mass_sq_div2 : out std_logic_vector(pt1_width+pt2_width+cosh_cos_width-1 downto 0)
    );
end invariant_mass;

architecture rtl of invariant_mass is

  constant INV_MASS_VECTOR_WIDTH : positive := pt1_width+pt2_width+cosh_cos_width;
  constant INV_MASS_PRECISION_FACTOR : real := real(10**INV_MASS_PRECISION);.pkg.
  constant FACTOR_4_VECTOR : std_logic_vector((INV_MASS_COSH_COS_PRECISION+1)*4-1 downto 0) := conv_std_logic_vector(10**(INV_MASS_COSH_COS_PRECISION+1),(INV_MAS

  signal inv_mass_sq_div2   : std_logic_vector(INV_MASS_VECTOR_WIDTH-1 downto 0);
  signal upper_limit_vector : std_logic_vector(INV_MASS_VECTOR_WIDTH-1 downto 0);
  signal lower_limit_vector : std_logic_vector(INV_MASS_VECTOR_WIDTH-1 downto 0);

  begin

    -- Converting the boundary value for the comparison
    upper_limit_vector <= conv_std_logic_vector((integer(upper_limit*INV_MASS_PRECISION_FACTOR)),INV_MASS_VECTOR_WIDTH-FACTOR_4_VECTOR'length)*FACTOR_4_VECTOR;
    lower_limit_vector <= conv_std_logic_vector((integer(lower_limit*INV_MASS_PRECISION_FACTOR)),INV_MASS_VECTOR_WIDTH-FACTOR_4_VECTOR'length)*FACTOR_4_VECTOR;

    -- Calculation of invariant mass with the formula: M**2/2 = pt1*pt2 * (cosh(eta1 - eta2) - cos(phi1 - phi2))
    inv_mass_sq_div2 <= pt1 * pt2 * (cosh_deta - cos_dphi);
    sim_inv_mass_sq_div2 <= inv_mass_sq_div2;

    -- Comparison with boundary values
    inv_mass_comp <= '1' when (inv_mass_sq_div2 >= lower_limit_vector and inv_mass_sq_div2 <= upper_limit_vector) else '0';

end architecture rtl;
```

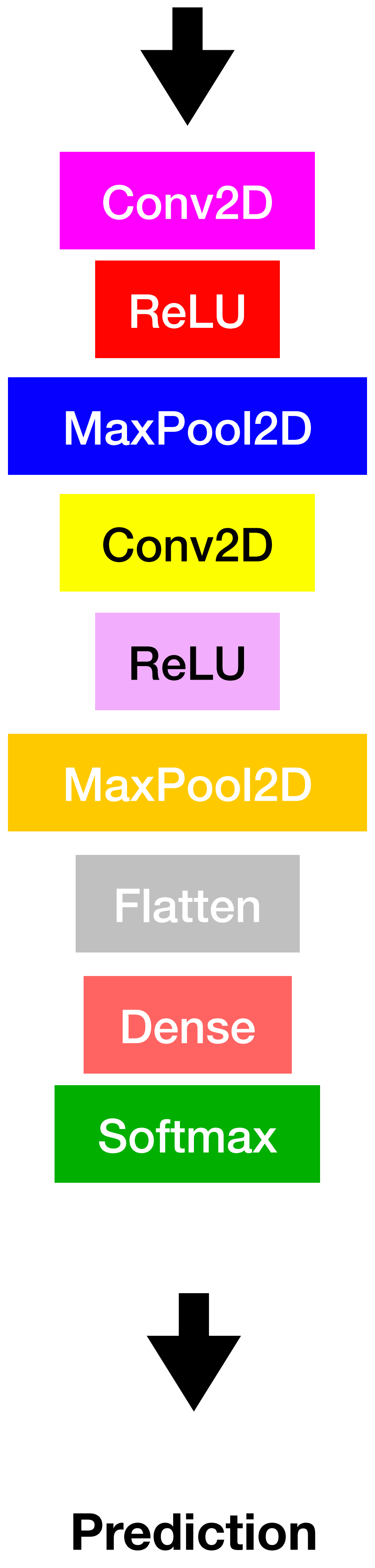
https://github.com/fastmachinelearning/hls4ml
https://fastmachinelearning.org/hls4ml/




Frontend
(Q)Keras
(Q)ONNX
PyTorch
IR
Graph
Backend
Vivado HLS
Vivado Accelerator
Quartus HLS
Catapult HLS
Vitis HLS
Custom?

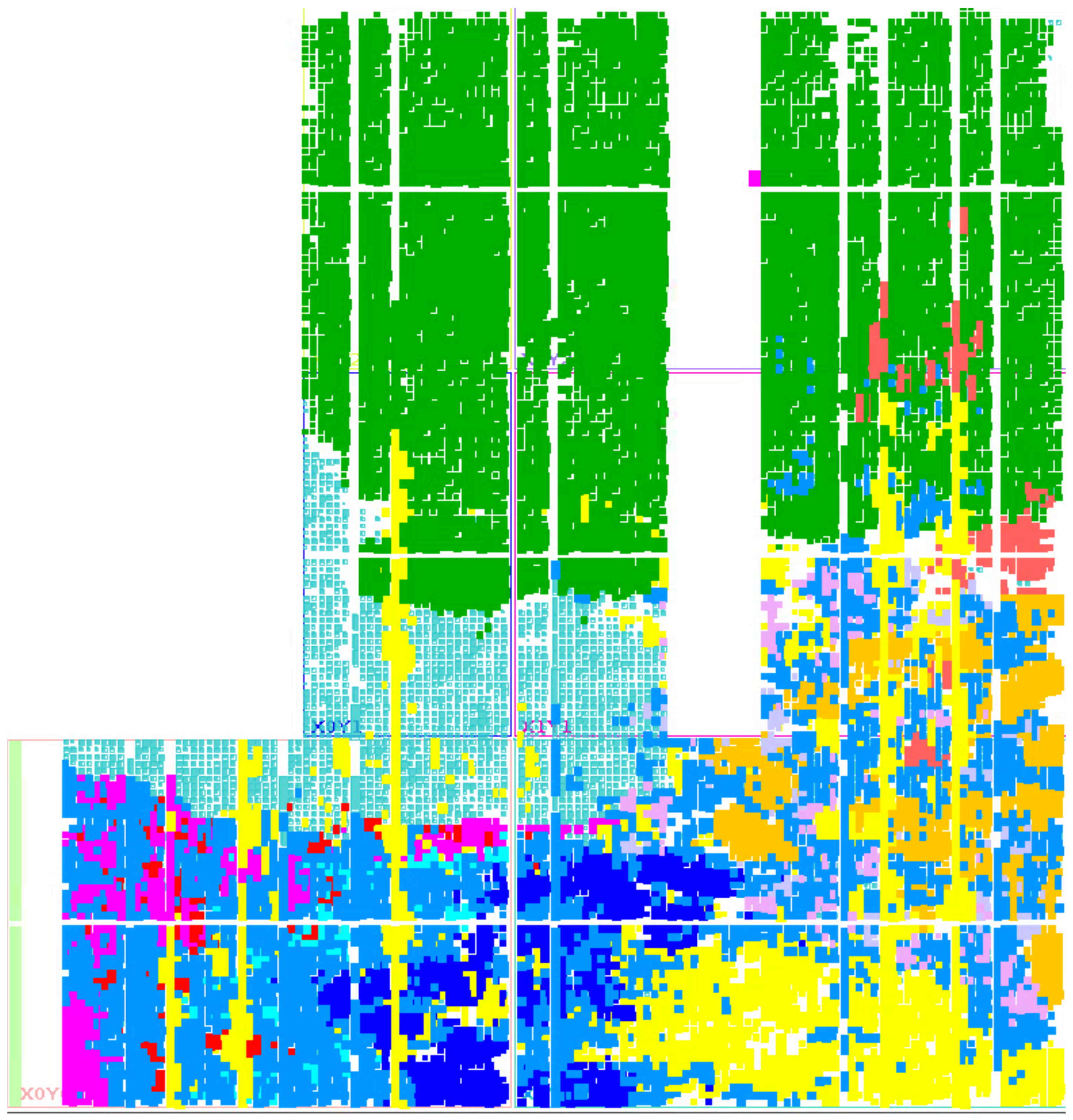
hls4ml






```
from hls4ml import …
import tensorflow as tf

# train or load a model
model = … # e.g. tf.keras.models.load_model(…)

# make a config template
cfg = config_from_keras_model(model,
granularity='name')

# tune the config
cfg['LayerName']['layer2']['ReuseFactor'] = 4

# do the conversion
hmodel = convert_from_keras_model(model, cfg)

# write and compile the HLS
hmodel.compile()

# run bit accurate emulation
y_tf = model.predict(x)
y_hls = hmodel.predict(x)

# do some validation
np.testing.assert_allclose(y_tf, y_hls)

# run HLS synthesis
hmodel.build()
```

pynq-z2 floorplan



(from Sioni S Summers)

$$\mathbf{x}_n = g_n(\mathbf{W}_{n,n-1}\mathbf{x}_{n-1} + \mathbf{b}_n)$$

activation function
precomputed and stored in BRAMs

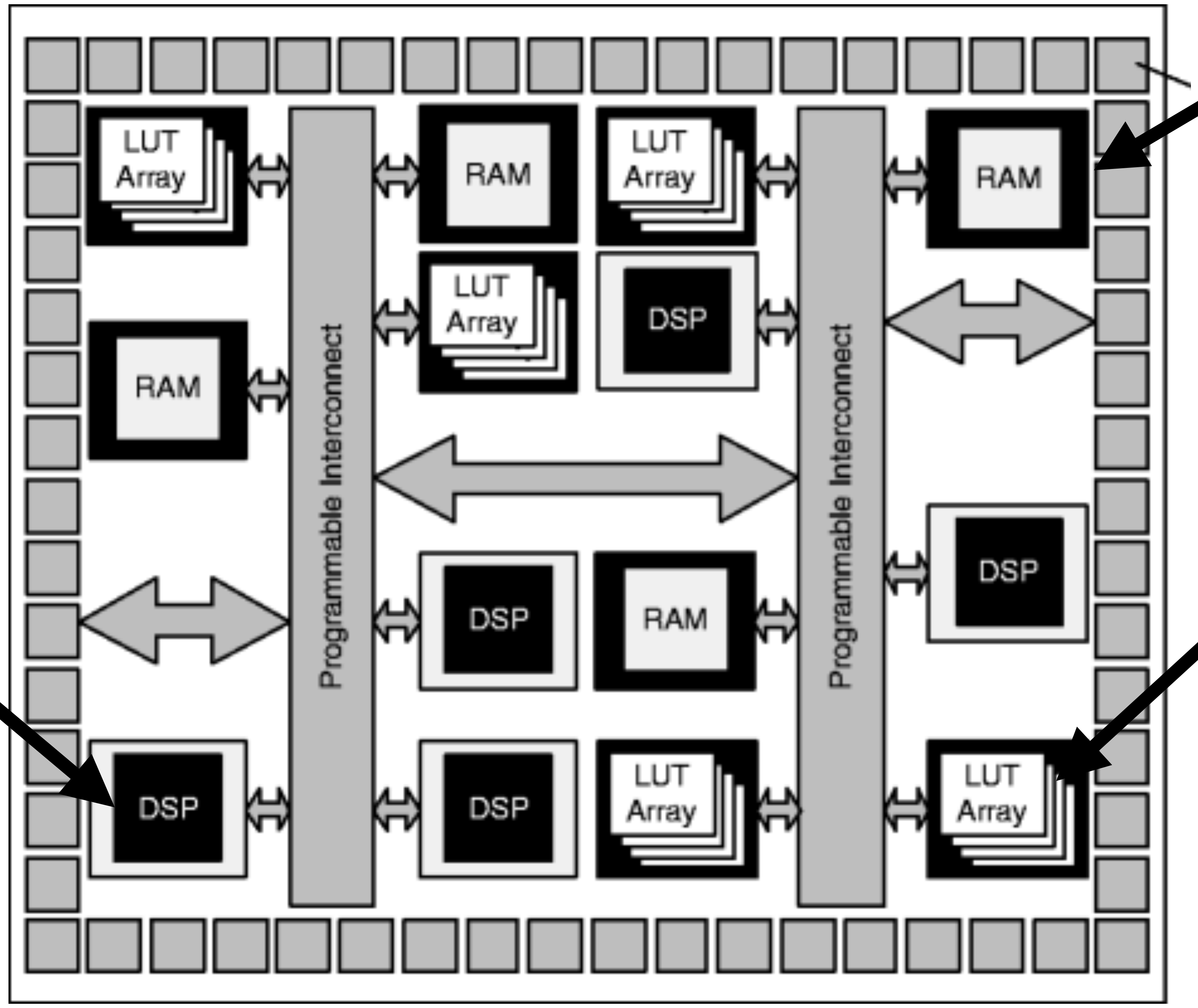
multiplication
DSPs

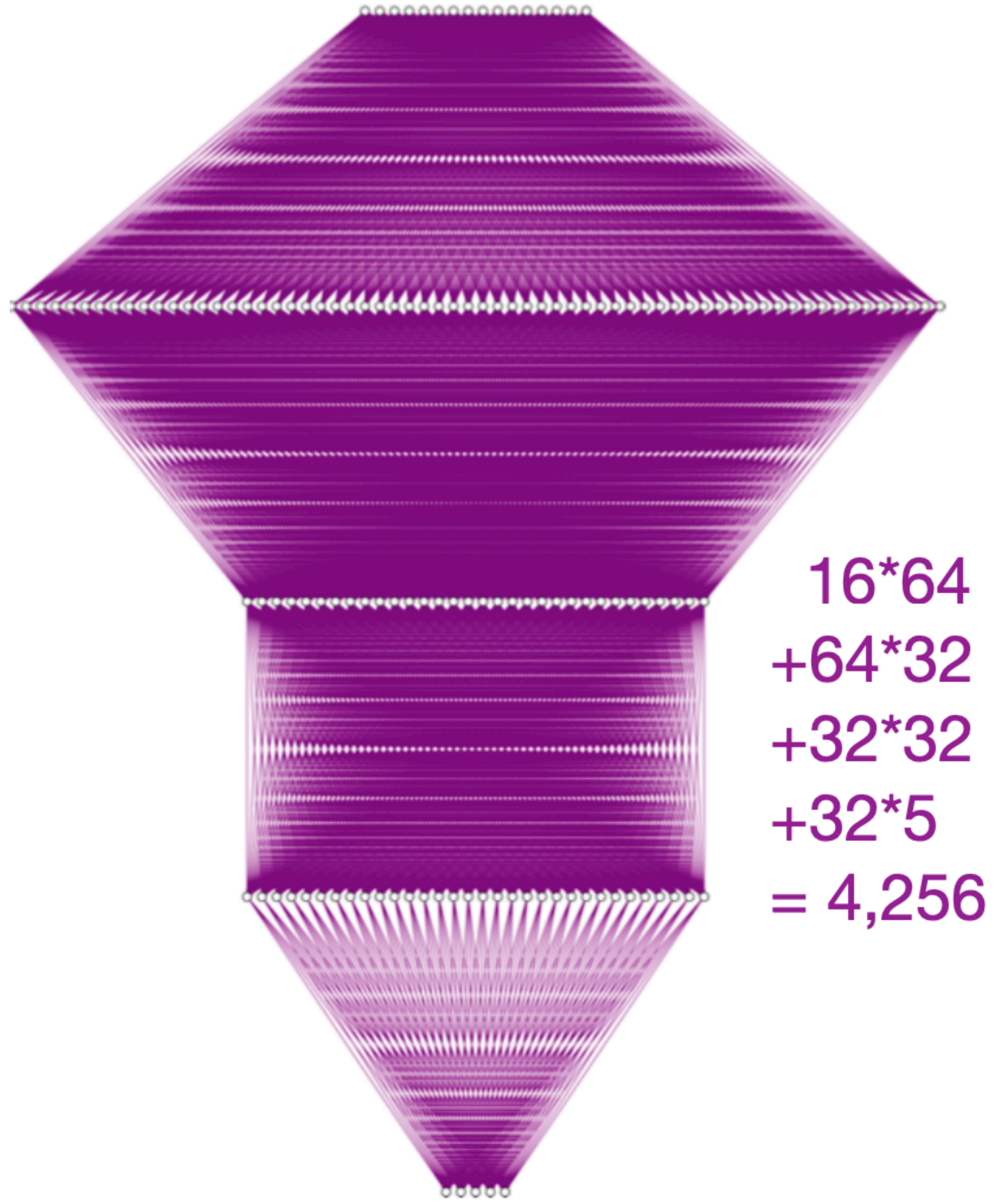
addition
logic cells

Memory (BRAM)
O(2000) units

Digital signal processors (DSPs)
O(5,000) units

Logic cells/lookup tables (LUTs)
O(1) million units





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

Network size limited by # multiplications

$$\mathbf{x}_n = g_n(\mathbf{W}_{n,n-1}\mathbf{x}_{n-1} + \mathbf{b}_n)$$

activation function — precomputed and stored in BRAMs

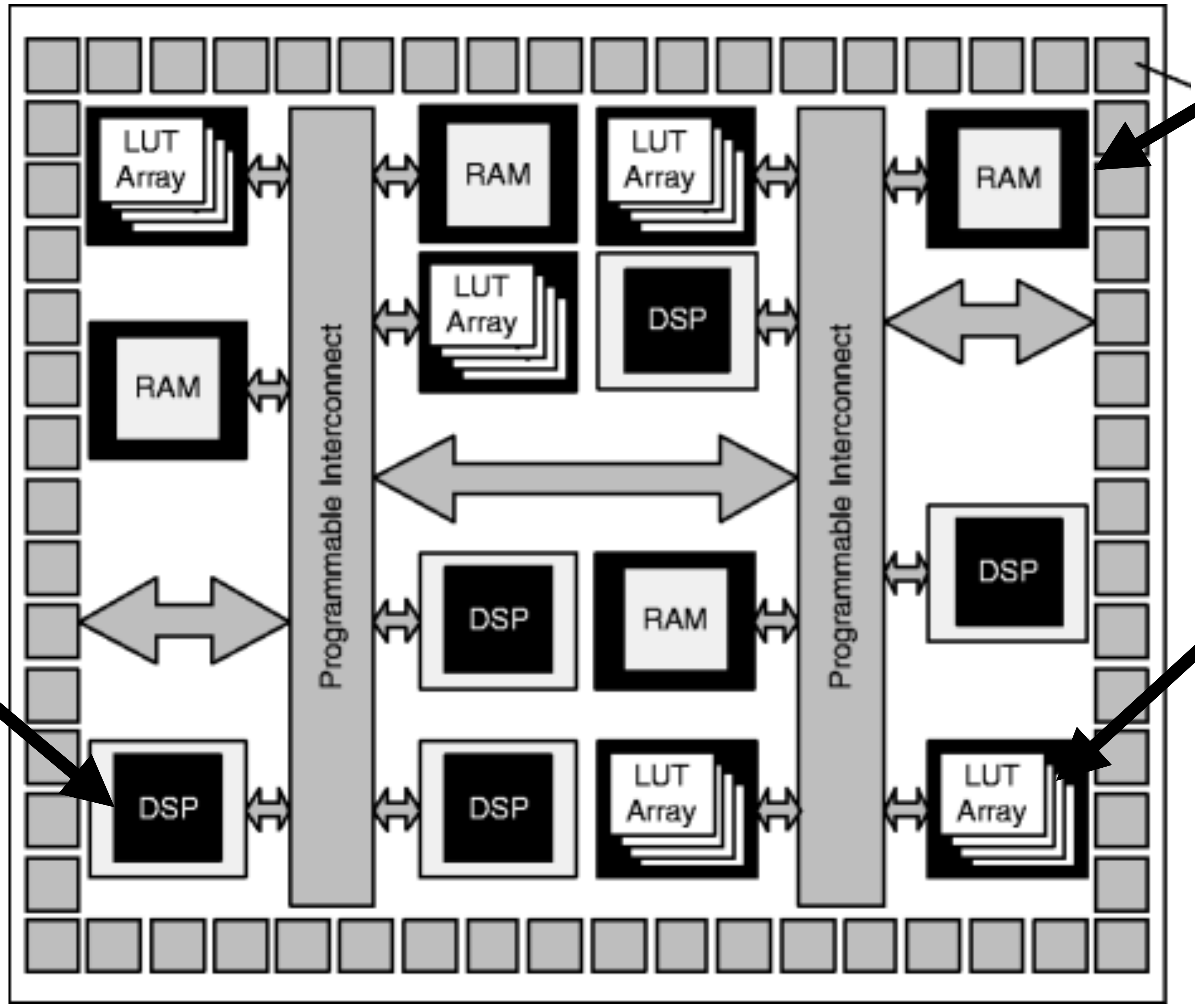
multiplication — DSPs

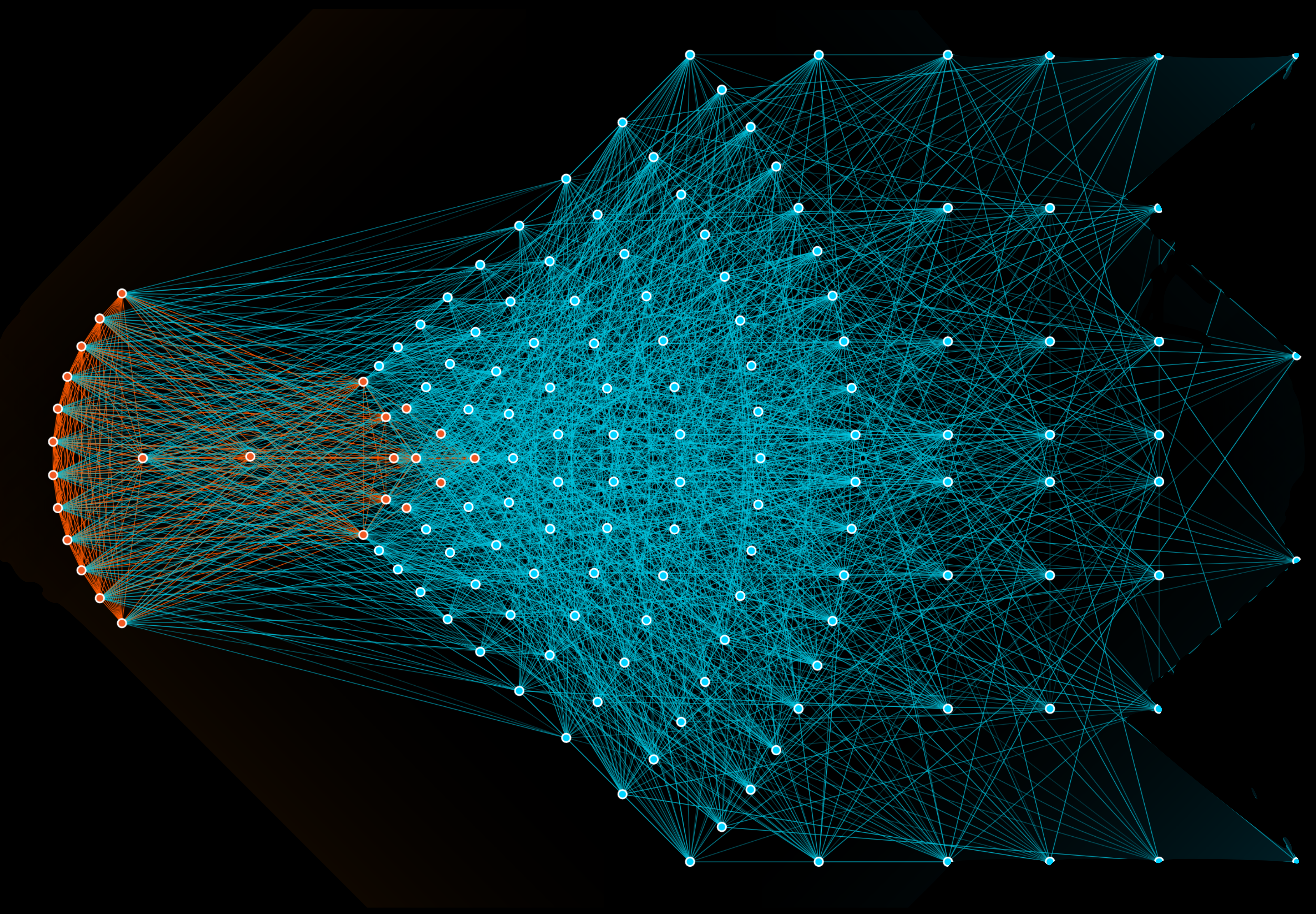
addition — logic cells

Memory (BRAM) O(2000) units

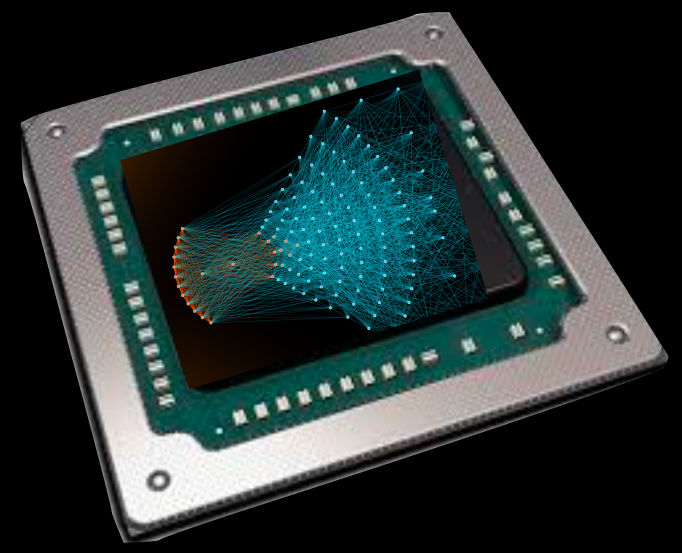
Digital signal processors (DSPs) O(5,000) units

Logic cells/lookup tables (LUTs) O(1) million units

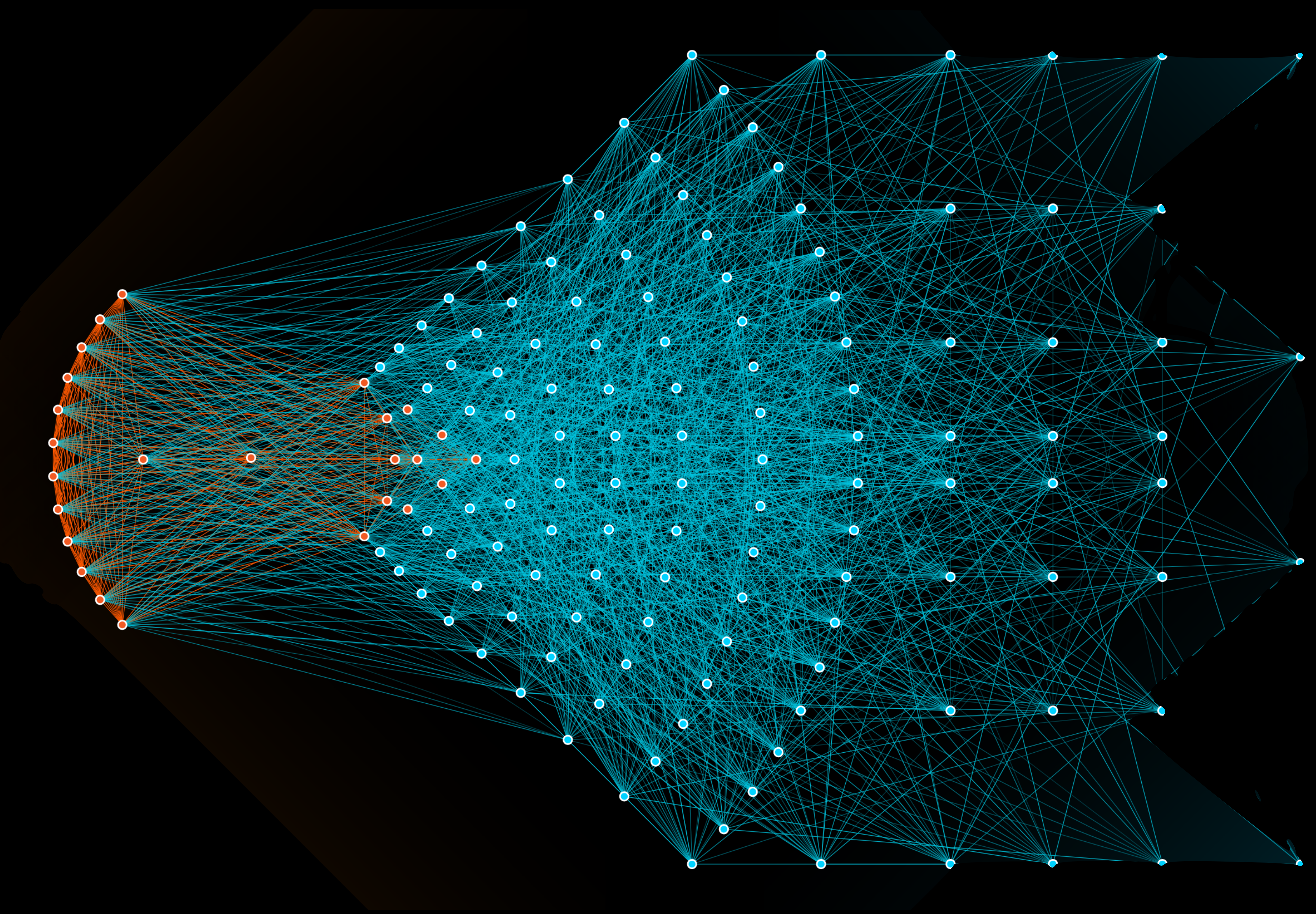




Ideally

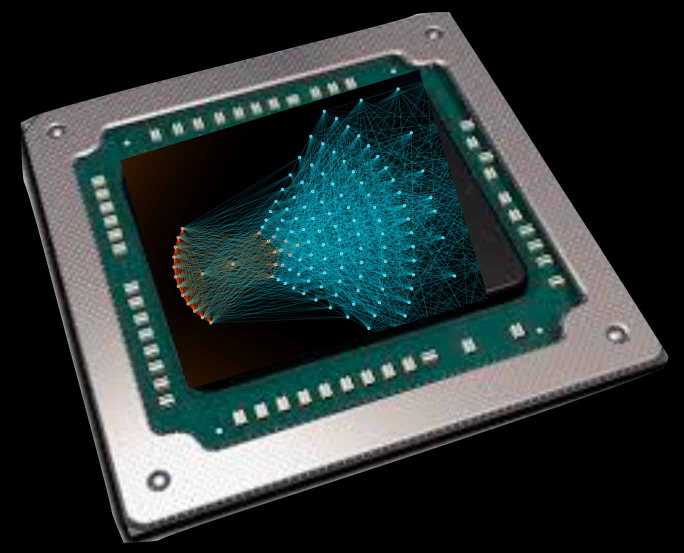


Reality



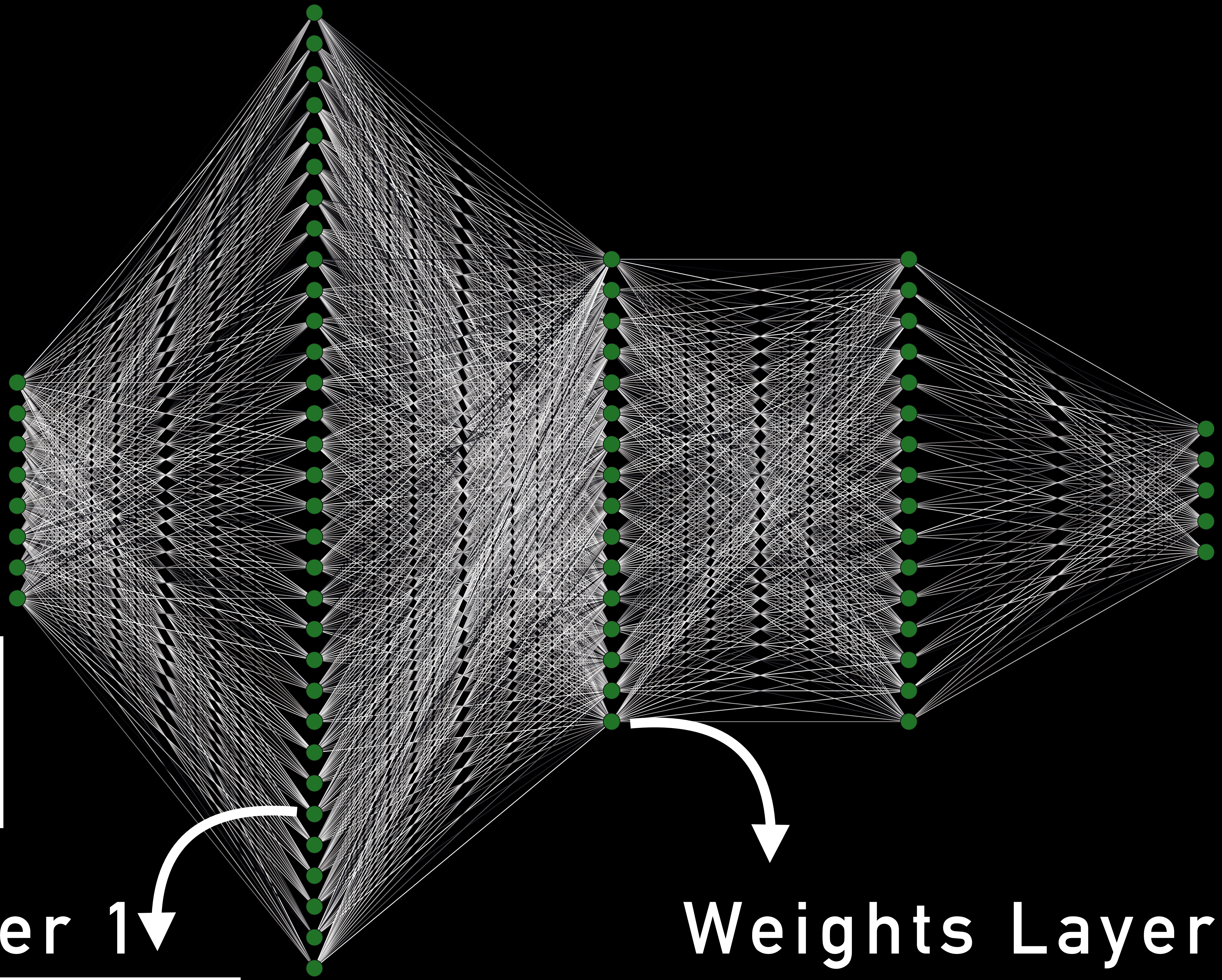
Ideally

- Quantization
- Pruning
- Parallelisation
- Knowledge distillation



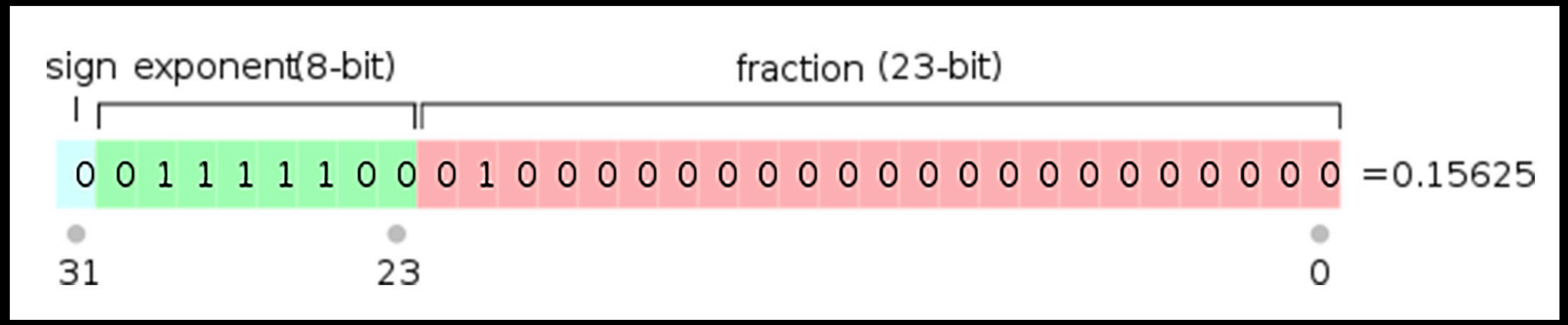
Reality

Quantization

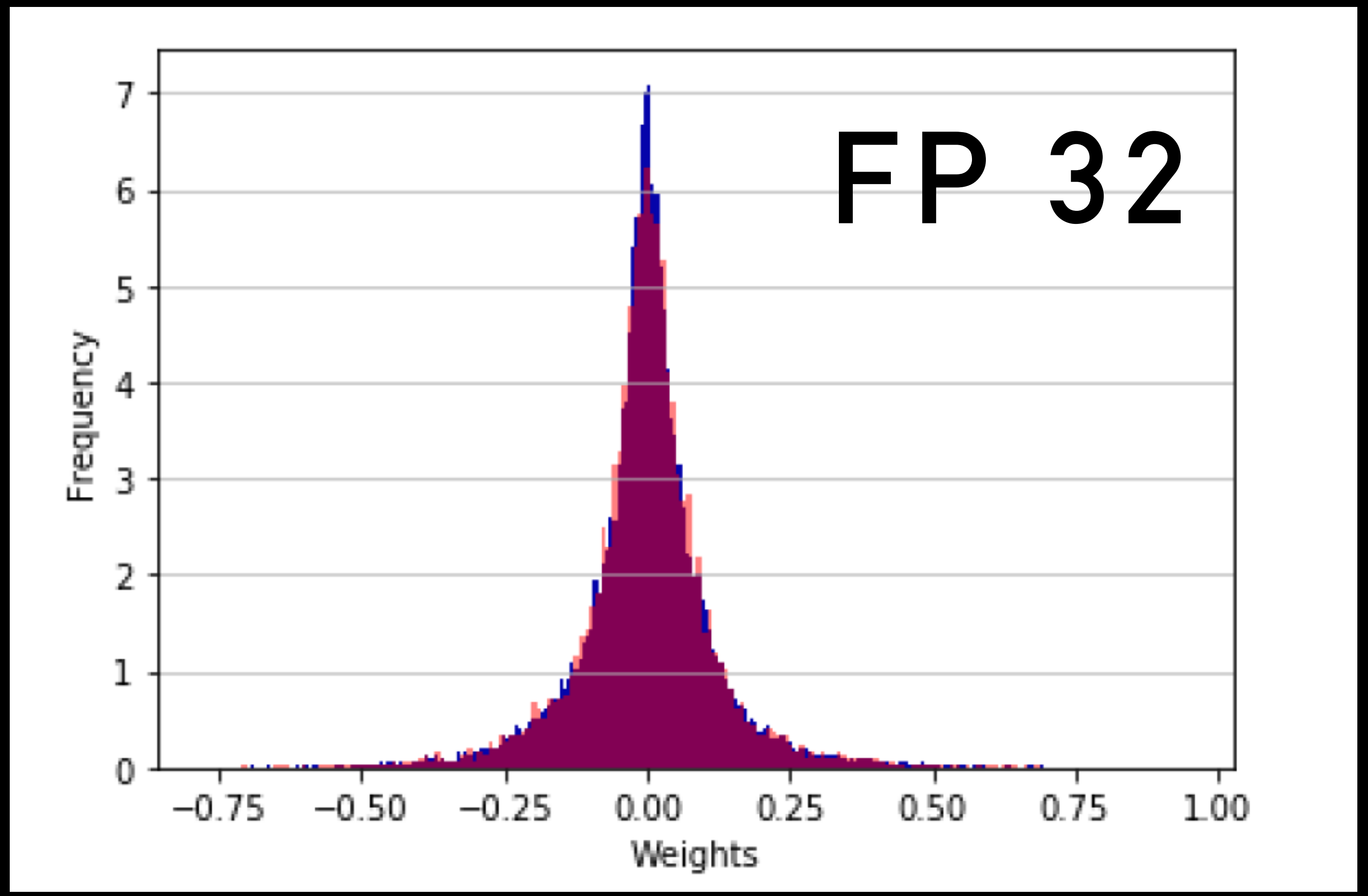


4B numbers in [-3.4e38, +3.4e38]

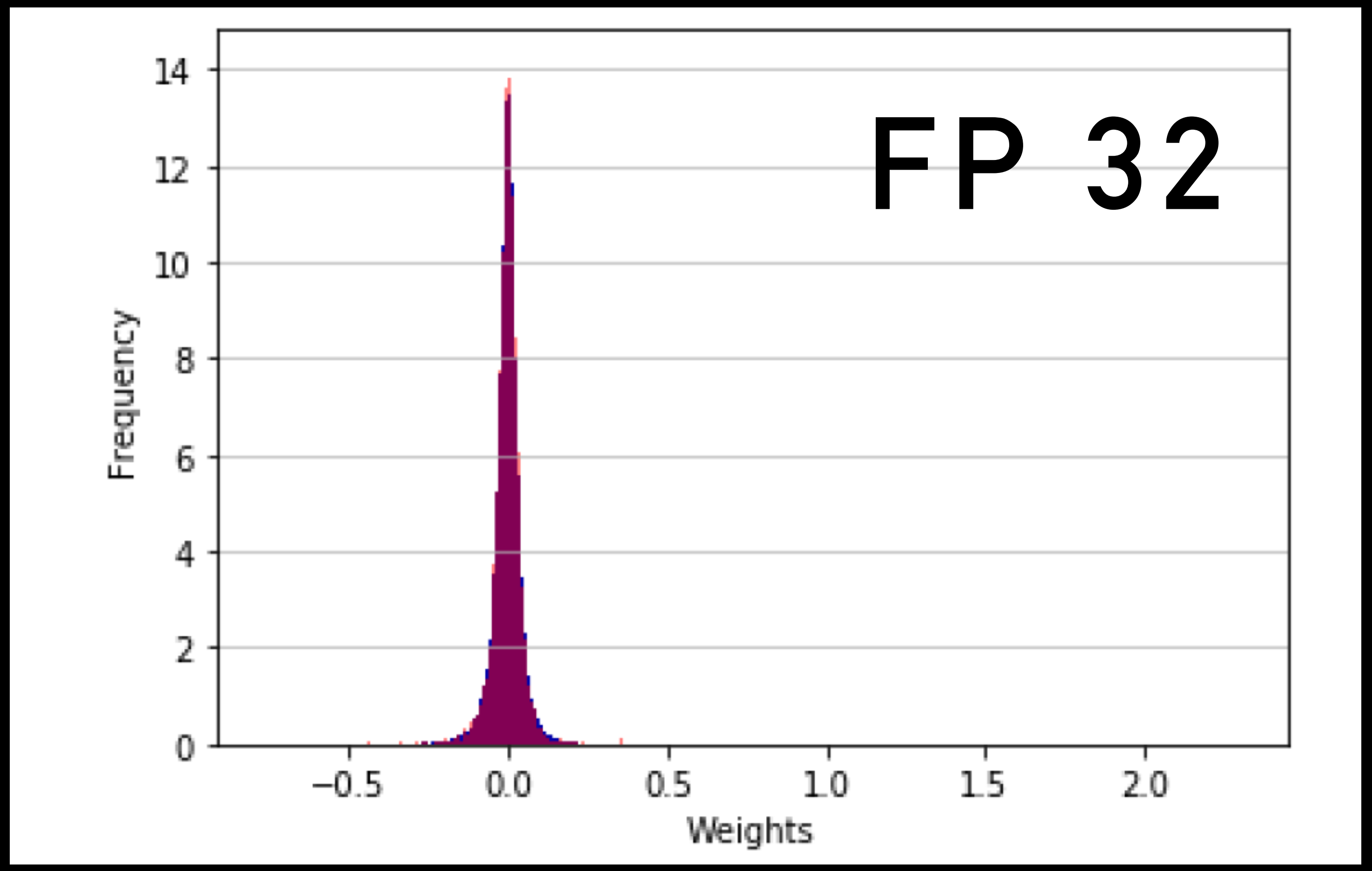
Floating point 32



Weights Layer 1



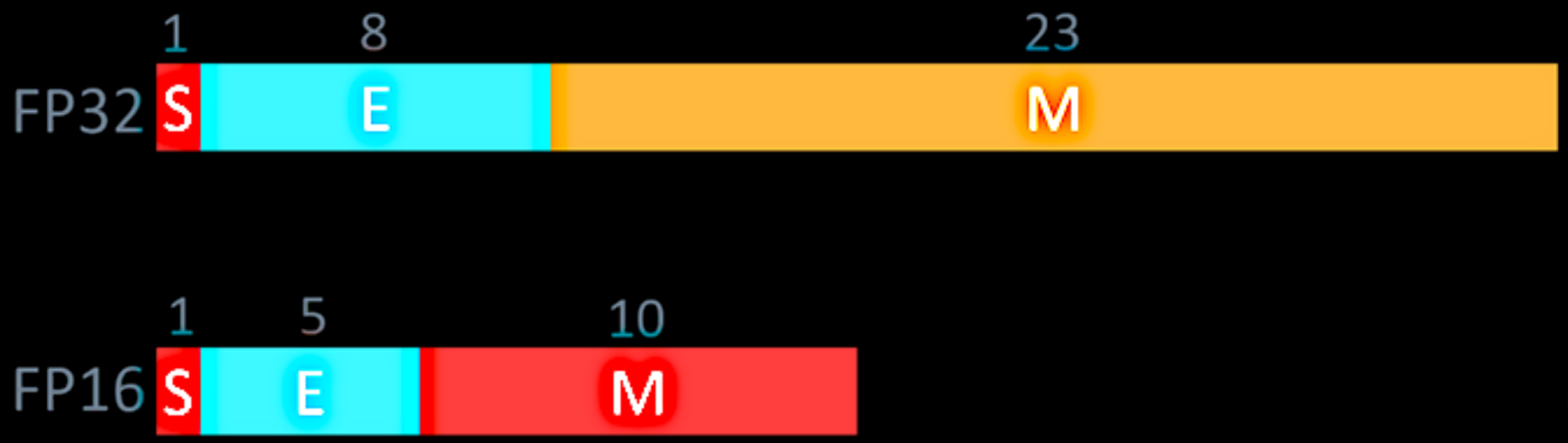
Weights Layer 2



GPT-3

175,000,000,000

FP16 vs FP32



	Range	Accuracy		
FP32	10^{-38} - 10^{38}	0.000006%	→	~700 GB
FP16	$6x10^{-5}$ - $6x10^{4}$	0.05%	→	~350 GB

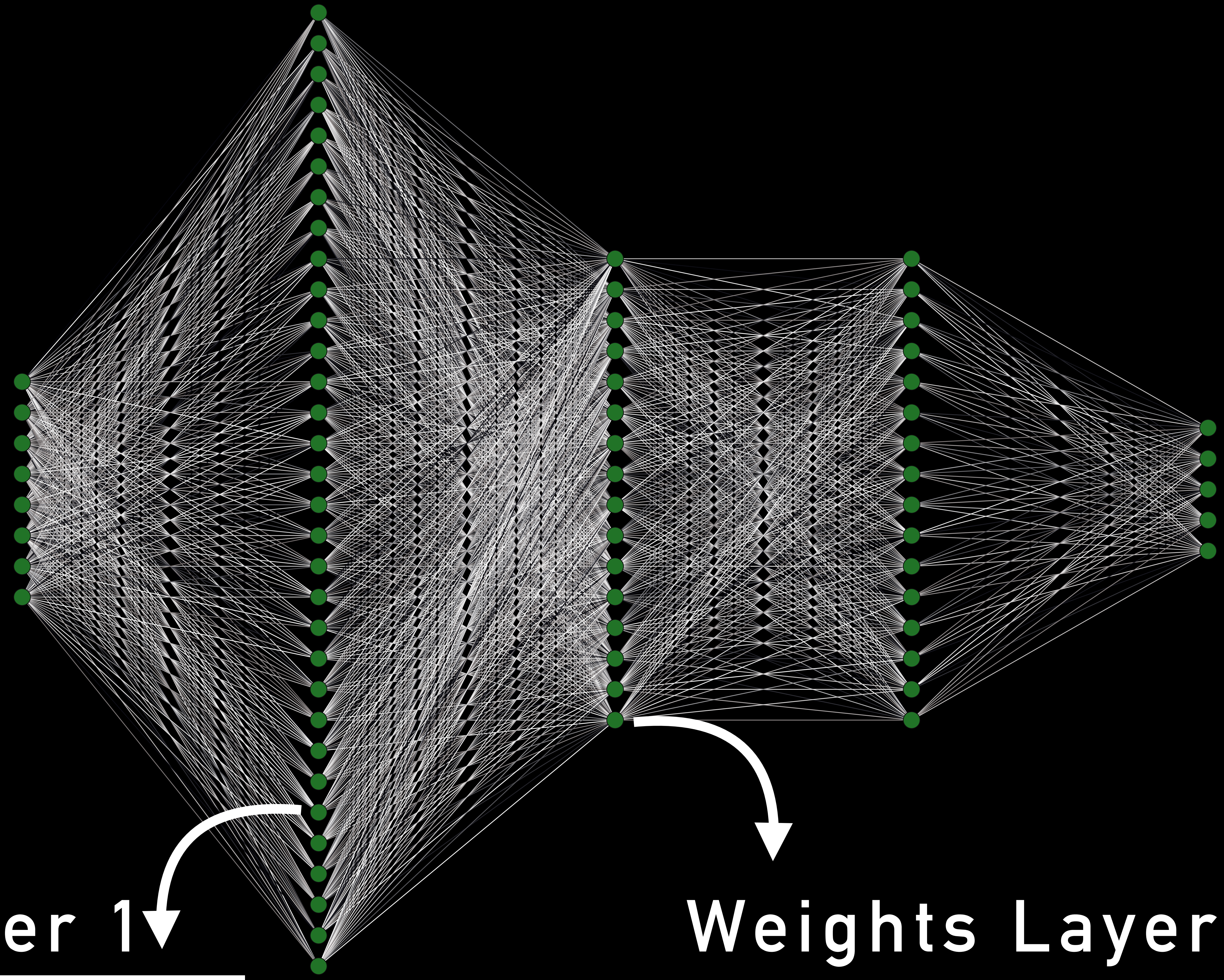
Memory:
~ FP16 x2 smaller than FP32
~ int8 x4 smaller than FP32

Speed:
~ FP16 x8 times faster than FP32

	Hopper	Ampere
Supported Tensor Core precisions	FP64, TF32, bfloat16, FP16, FP8, INT8	FP64, TF32, bfloat16, FP16, INT8, INT4, INT1

Fixed-point arithmetic:
Any precision!

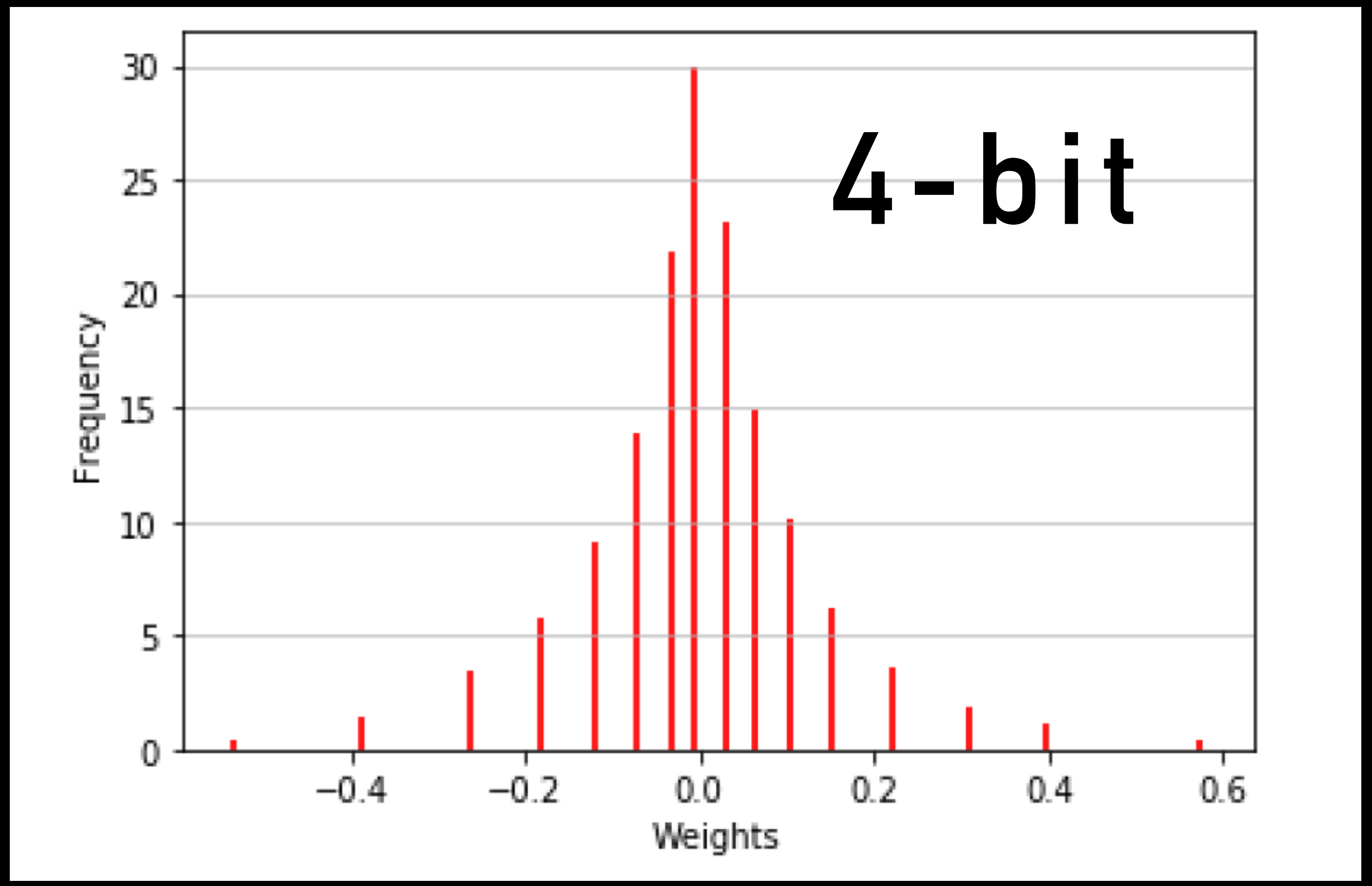
Quantization



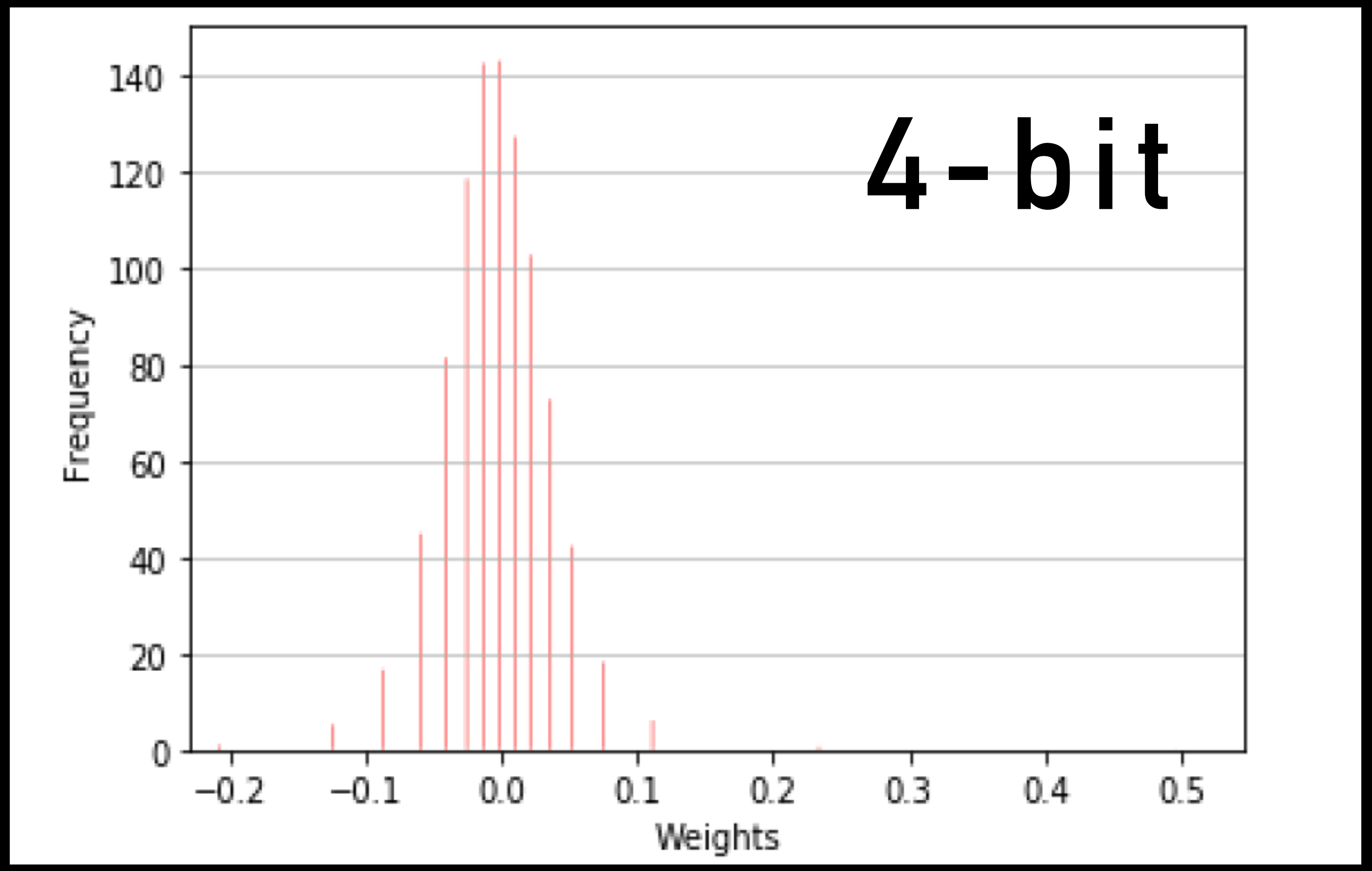
2^4 numbers in s*[-8, +7]

Fixed point

Weights Layer 1



Weights Layer 2



Quantization

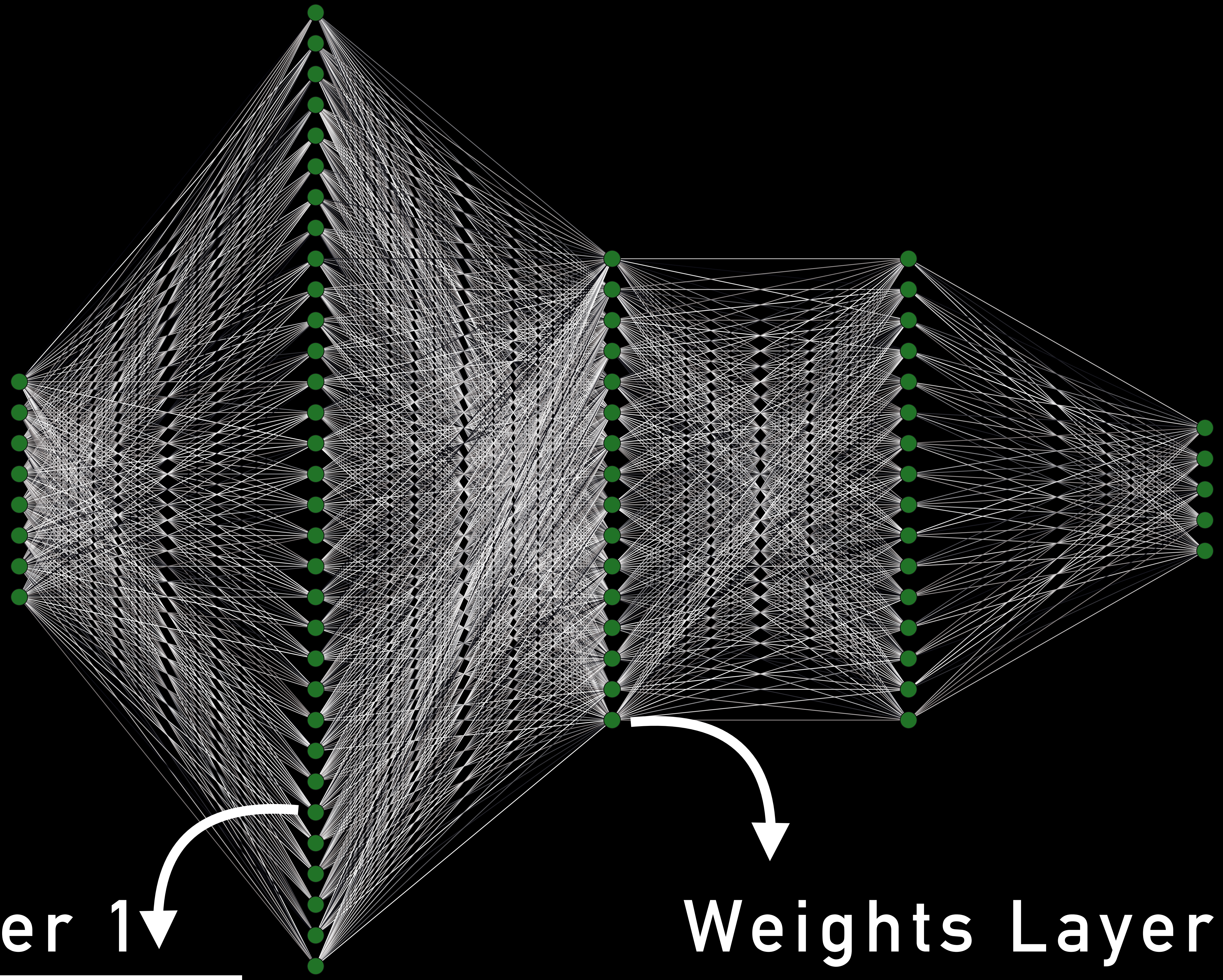
2^4 numbers in s*[-8, +7]

Fixed point

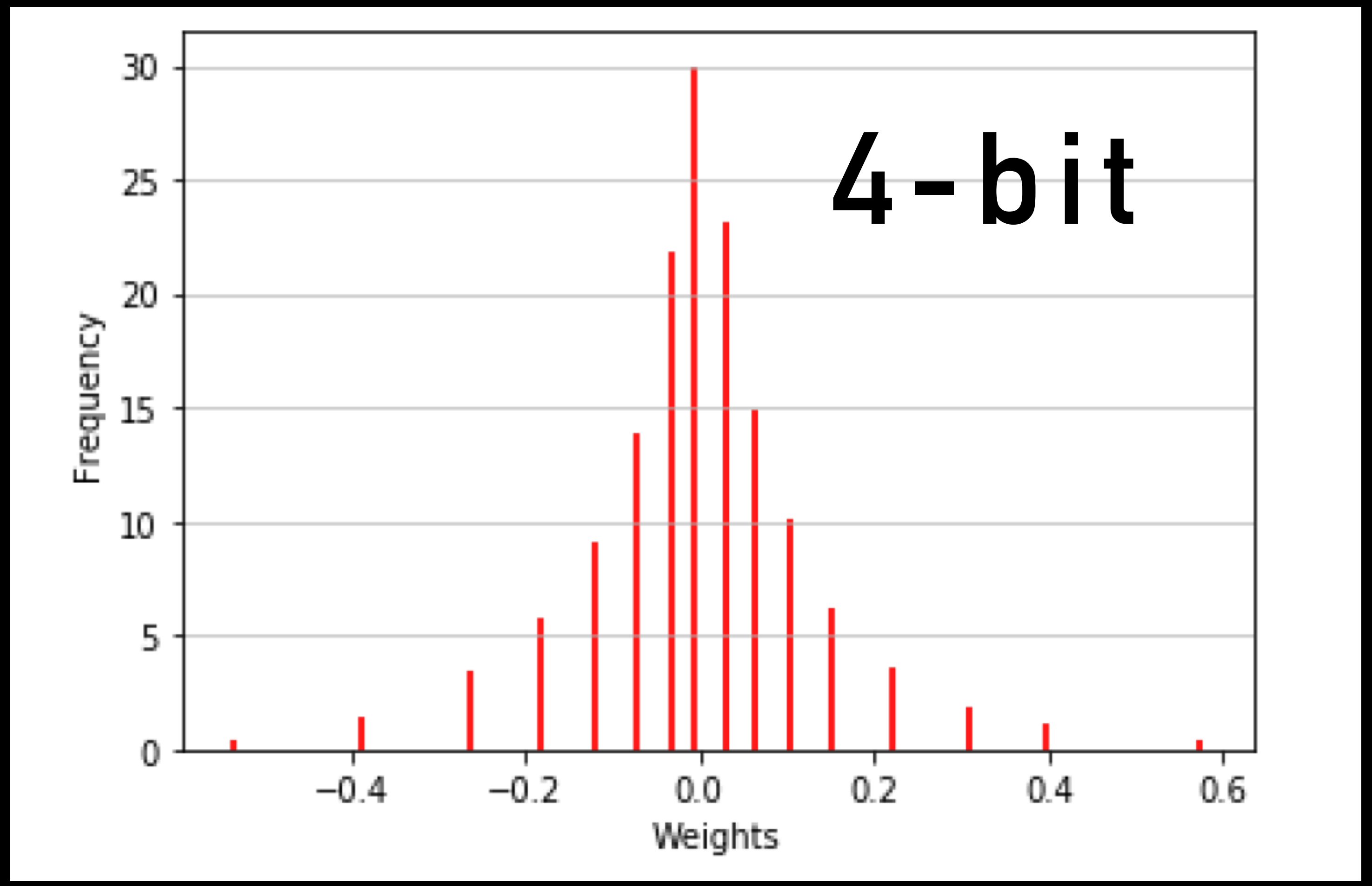
0101.1011101010

integer | fractional

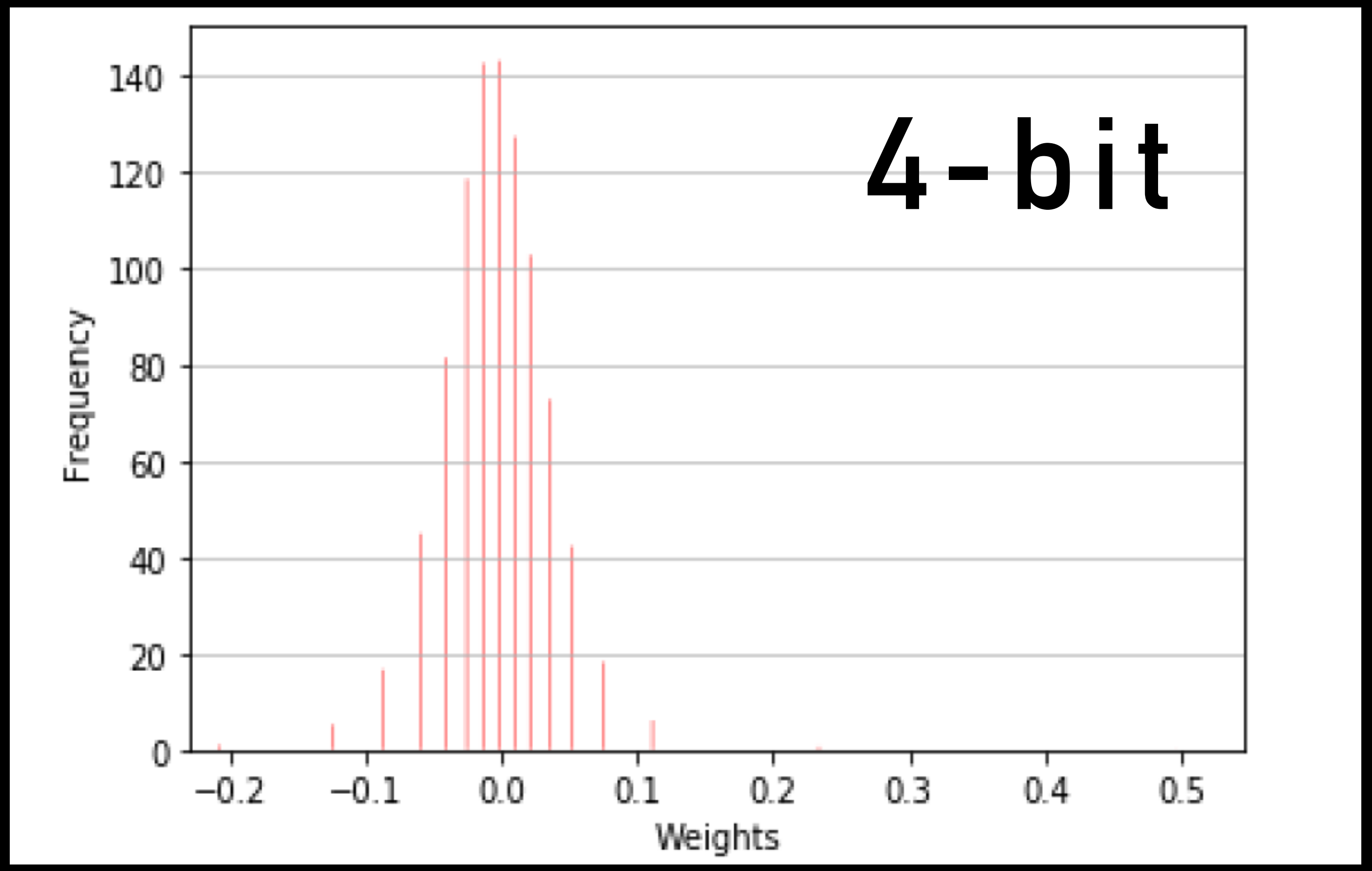
width

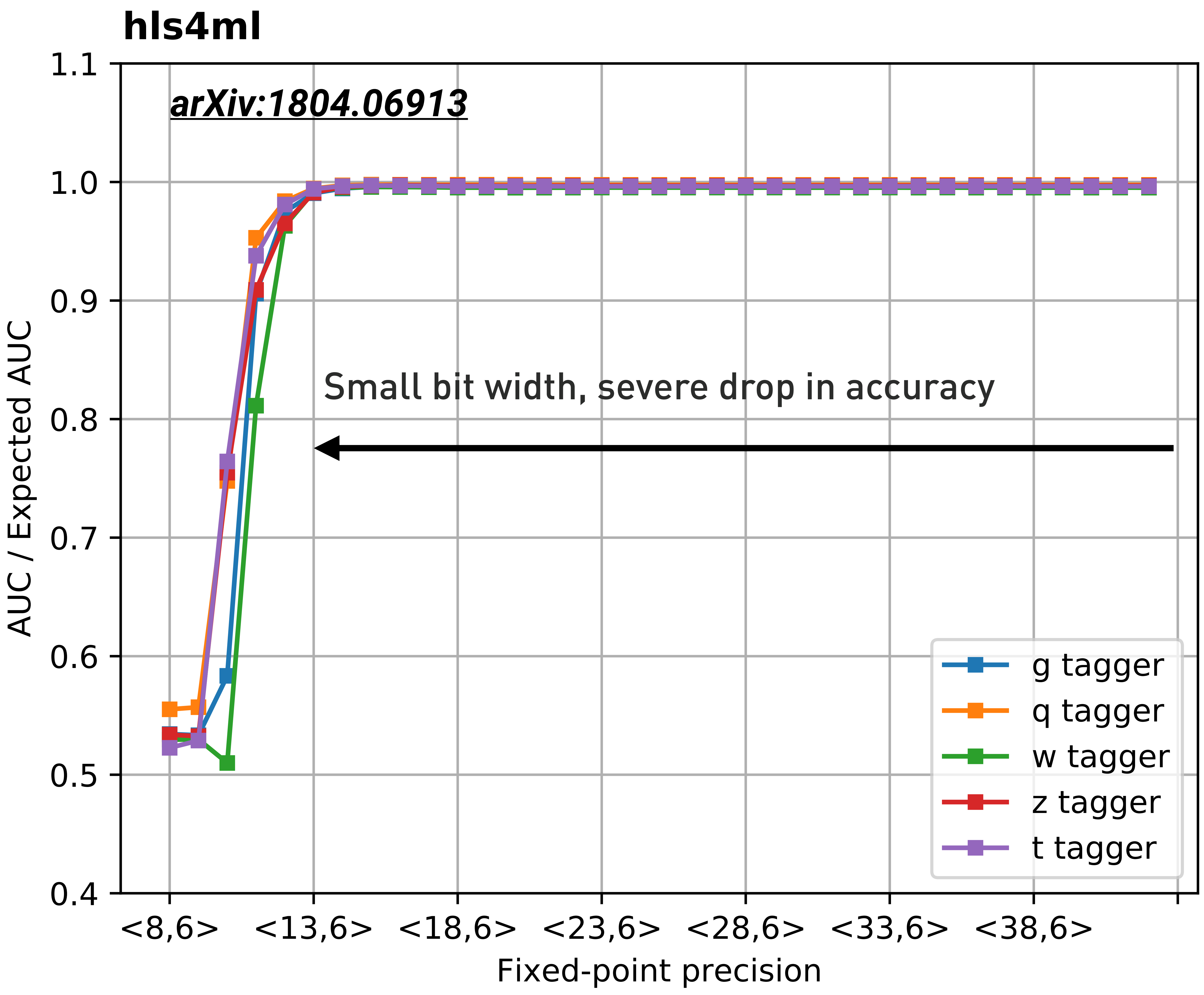


Weights Layer 1



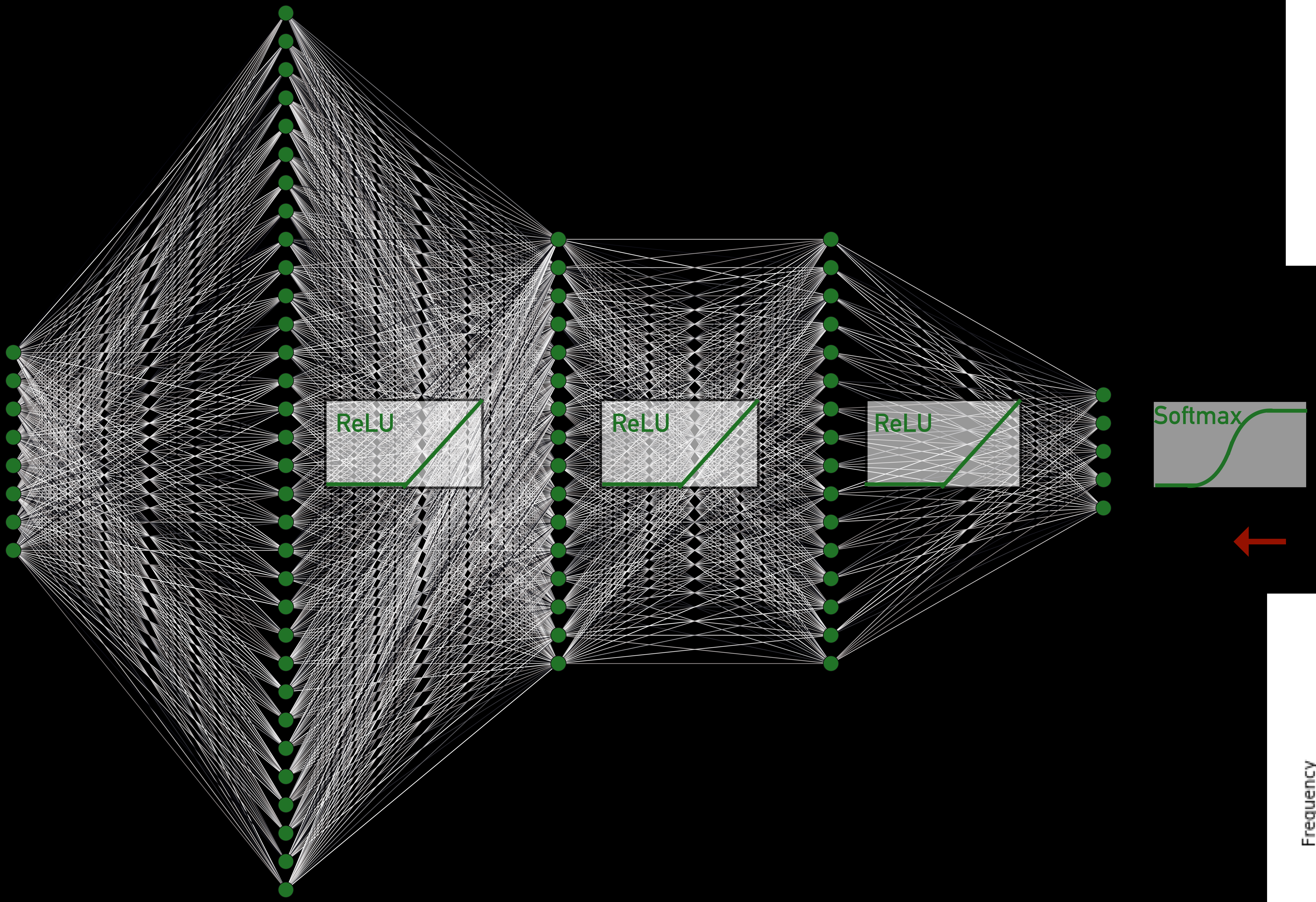
Weights Layer 2



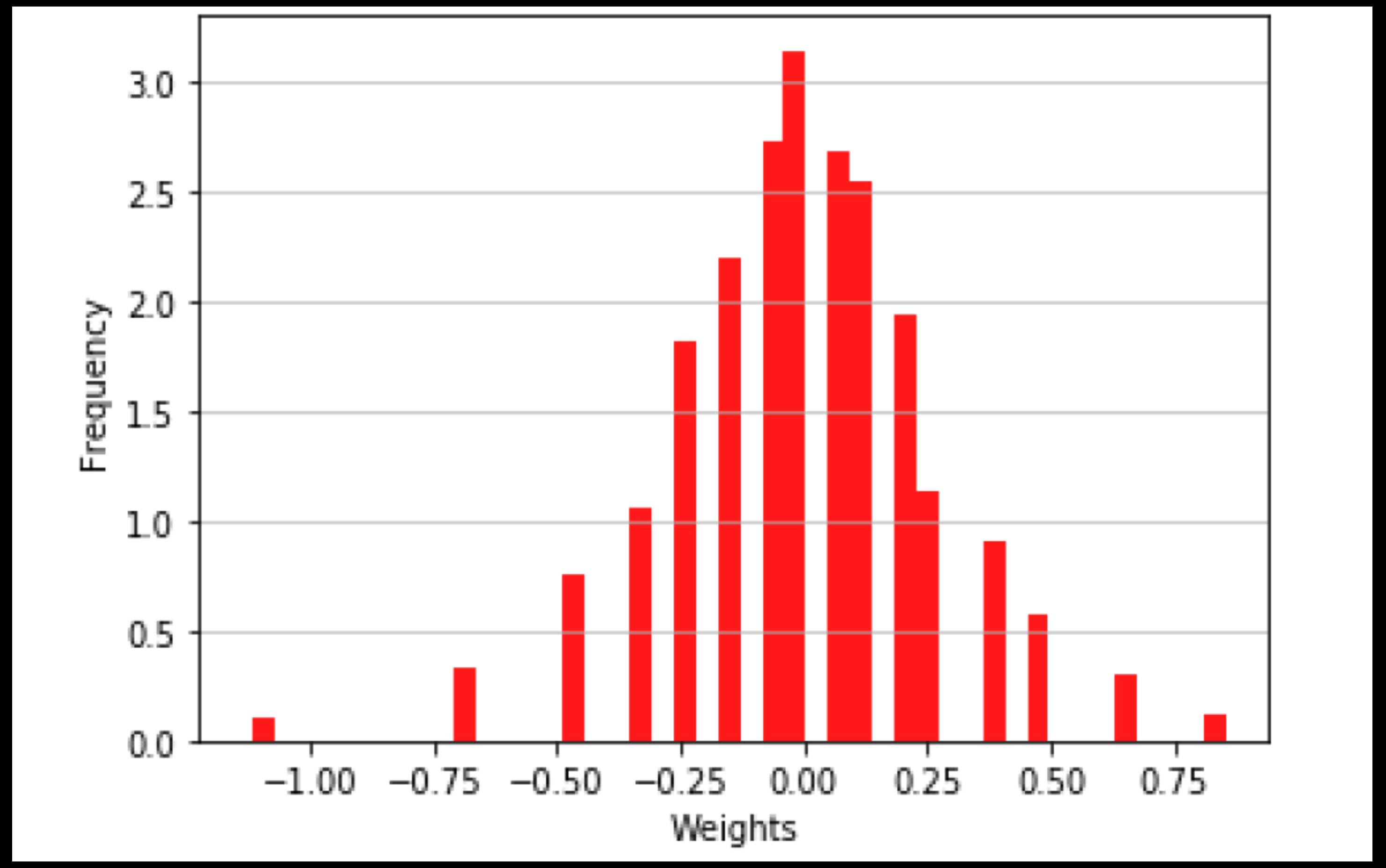

hls4ml
arXiv:1804.06913
Small bit width, severe drop in accuracy
AUC / Expected AUC
Fixed-point precision
<8,6>
<13,6>
<18,6>
<23,6>
<28,6>
<33,6>
<38,6>
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.4
g tagger
q tagger
w tagger
z tagger
t tagger

hls4ml + Google

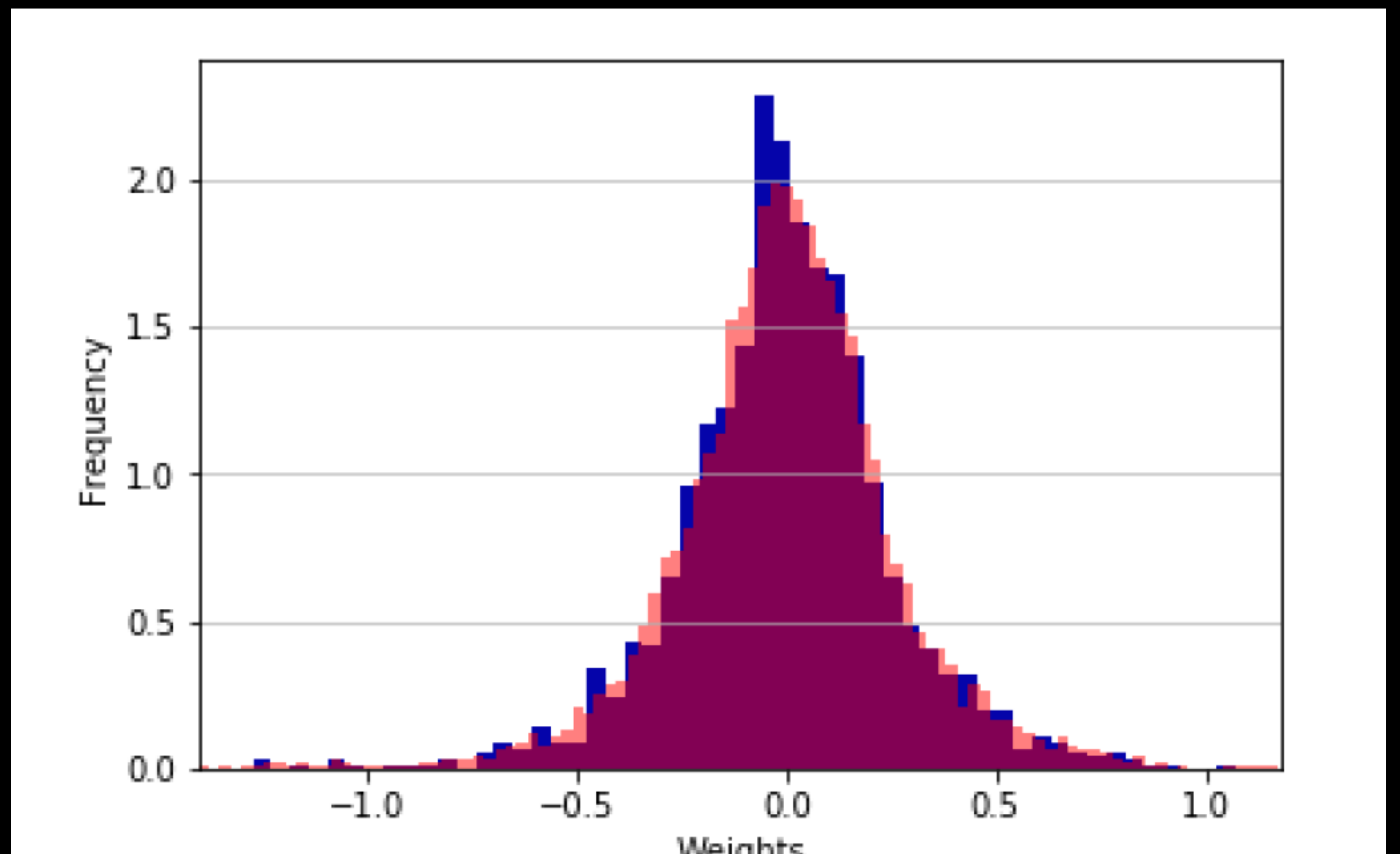
Quantization-aware training



Forward pass →



← Back propagation

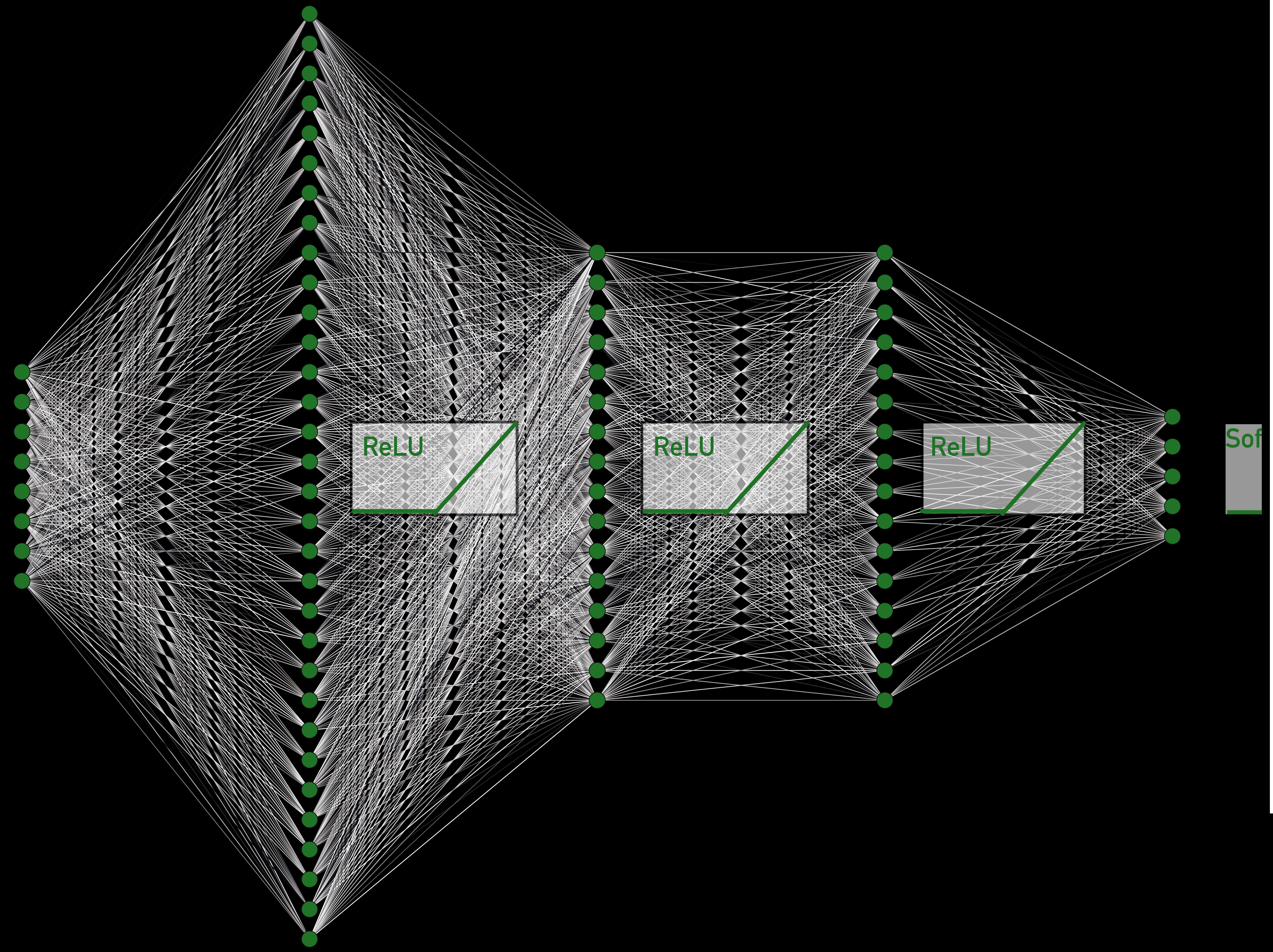


Nature Machine Intelligence 3 (2021)

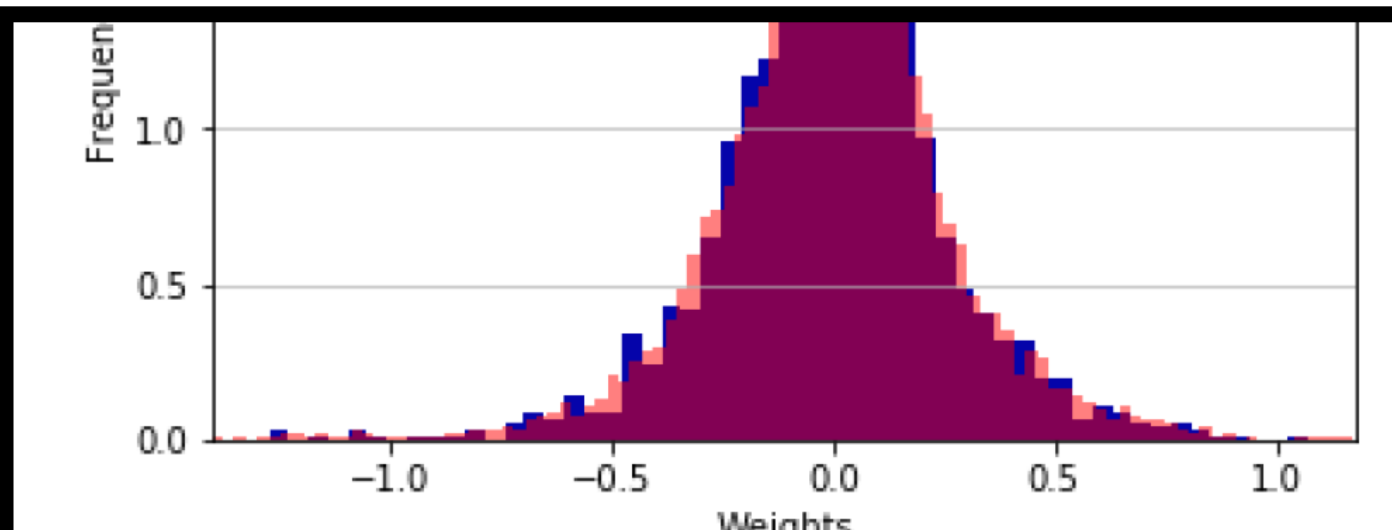
hls4ml + Google

Quantization-aware training

Forward pass →




```
from tensorflow.keras.layers import Input, Activation
from qkeras import quantized_bits
from qkeras import QDense, QActivation
from qkeras import QBatchNormalization

x = Input((16))
x = QDense(64,
    kernel_quantizer = quantized_bits(6,0,alpha=1),
    bias_quantizer   = quantized_bits(6,0,alpha=1))(x)
x = QBatchNormalization()(x)
x = QActivation('quantized_relu(6,0)')(x)
x = QDense(32,
    kernel_quantizer = quantized_bits(6,0,alpha=1),
    bias_quantizer   = quantized_bits(6,0,alpha=1))(x)
x = QBatchNormalization()(x)
x = QActivation('quantized_relu(6,0)')(x)
x = QDense(32,
    kernel_quantizer = quantized_bits(6,0,alpha=1),
    bias_quantizer   = quantized_bits(6,0,alpha=1))(x)
x = QBatchNormalization()(x)
x = QActivation('quantized_relu(6,0)')(x)
x = QDense(5,
    kernel_quantizer = quantized_bits(6,0,alpha=1),
    bias_quantizer   = quantized_bits(6,0,alpha=1))(x)
x = Activation('softmax')(x)
```



Nature Machine Intelligence 3 (2021)

Estimating energy and size

Some layers more accommodating for aggressive quantization, others require expensive arithmetic

- heterogeneous quantization

Estimating energy and size

Some layers more accommodating for aggressive quantization, others require expensive arithmetic

- heterogeneous quantization

For edge inference, need best possible quantization configuration for

- Highest accuracy ↑...
- ... and lowest resource consumption ↓

→ hyper-parameter scan over quantizers which considers energy and accuracy simultaneously

Estimating energy and size

Some layers more accommodating for aggressive quantization, others require expensive arithmetic

- heterogeneous quantization

For edge inference, need best possible quantization configuration for

- Highest accuracy ↑...
- ... and lowest resource consumption ↓

→ hyper-parameter scan over quantizers which considers energy and accuracy simultaneously

QTools: Estimate QKeras model bit and energy consumption, assuming 45 nm Horowitz process

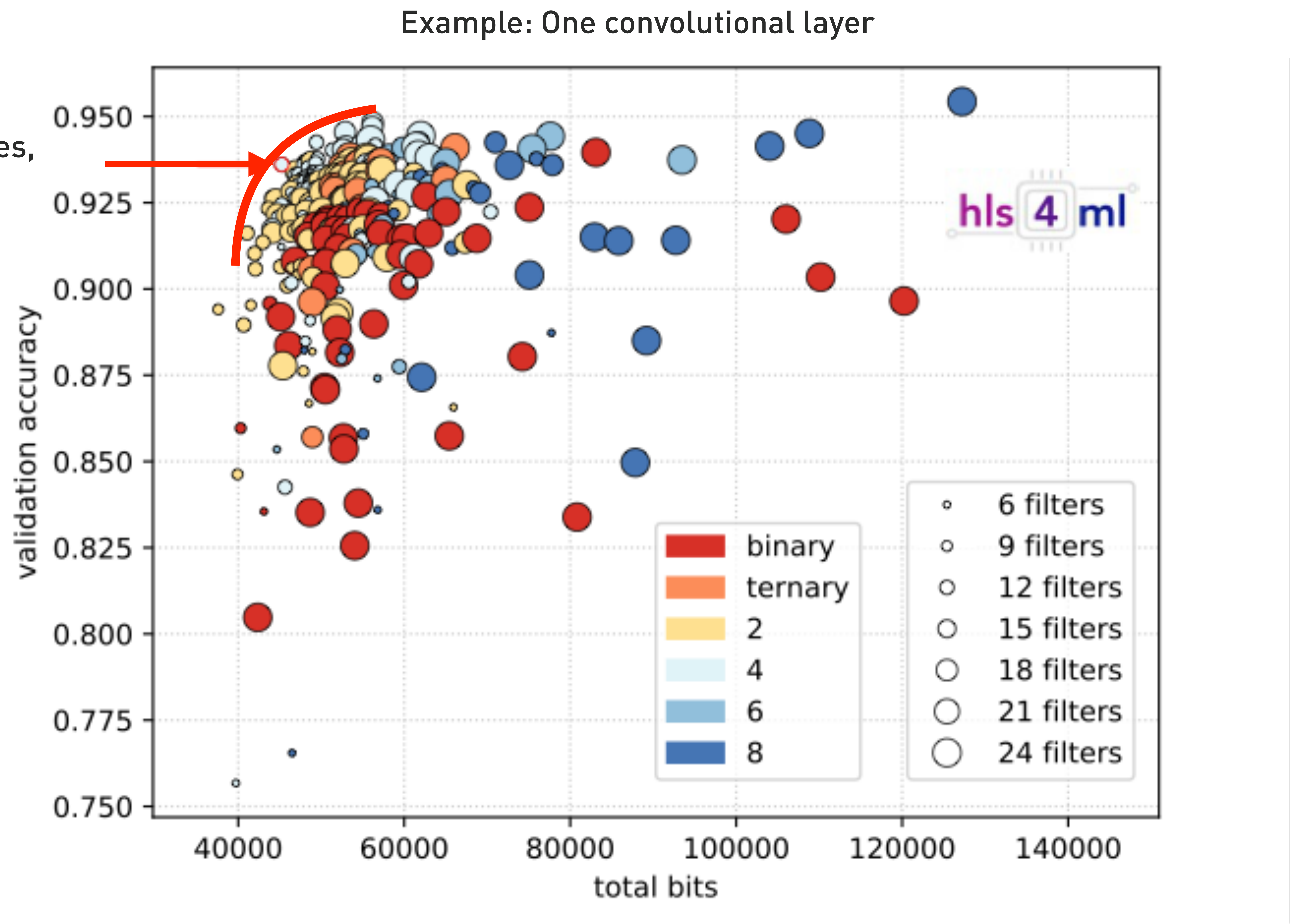
Model	Accuracy [%]	Per-layer energy consumption [pJ]								Total energy [μJ]	Total bits
		Dense	ReLU	Dense	ReLU	Dense	ReLU	Dense	Softmax		
BF	74.4	1735	53	3240	27	1630	27	281	11	0.00700	61446
Q6	74.8	794	23	1120	11	562	11	99	11	0.00263	26334

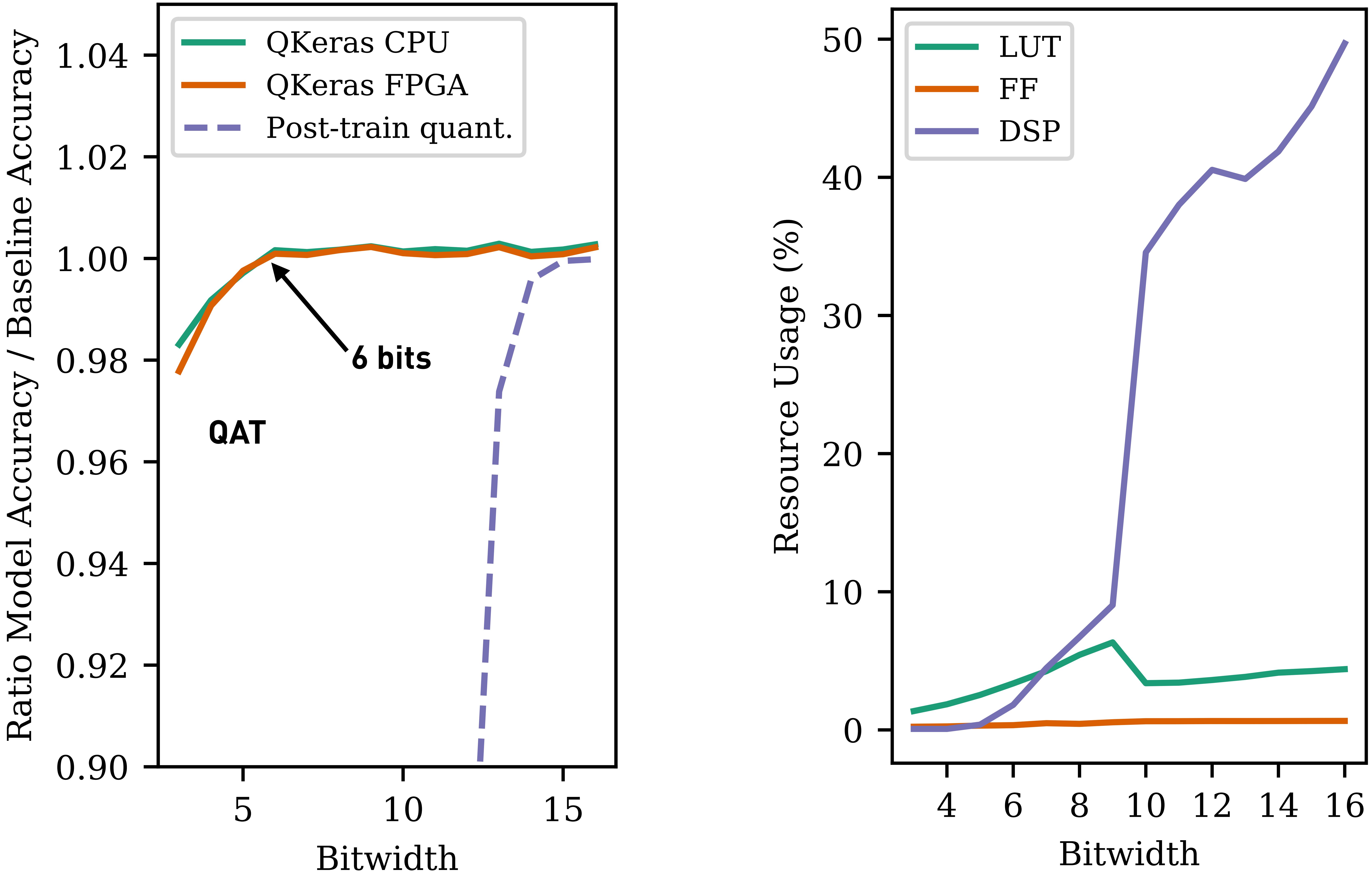
$$\text{Forgiving Factor} = 1 + \Delta_{accuracy} \times \log_{rate}(S \times \frac{Cost_{ref}}{Cost_{trial}})$$

Maximize accuracy + minimizing cost in hyper parameter scan over quantizers: AutoQKeras

DOI 10.1088/2632-2153/ac0ea1

As optimization progresses, best model accuracy/size trade-off is found!




QKeras CPU
QKeras FPGA
Post-train quant.
Ratio Model Accuracy / Baseline Accuracy
1.04
1.02
1.00
0.98
0.96
0.94
0.92
0.90
6 bits
QAT
5
10
15
Bitwidth
LUT
FF
DSP
Resource Usage (%)
50
40
30
20
10
0
4
6
8
10
12
14
16
Bitwidth

AMD UltraScale+ MPSoC ZU19EG (conservative estimates)

Precision	Approx. **Peak GOPS**	**On-chip weights**
1b	**64 000**	**~64 M**
4b	**16 000**	**~16 M**
8b	**4 000**	**~8 M**
32b	**300**	**~2 M**

200x

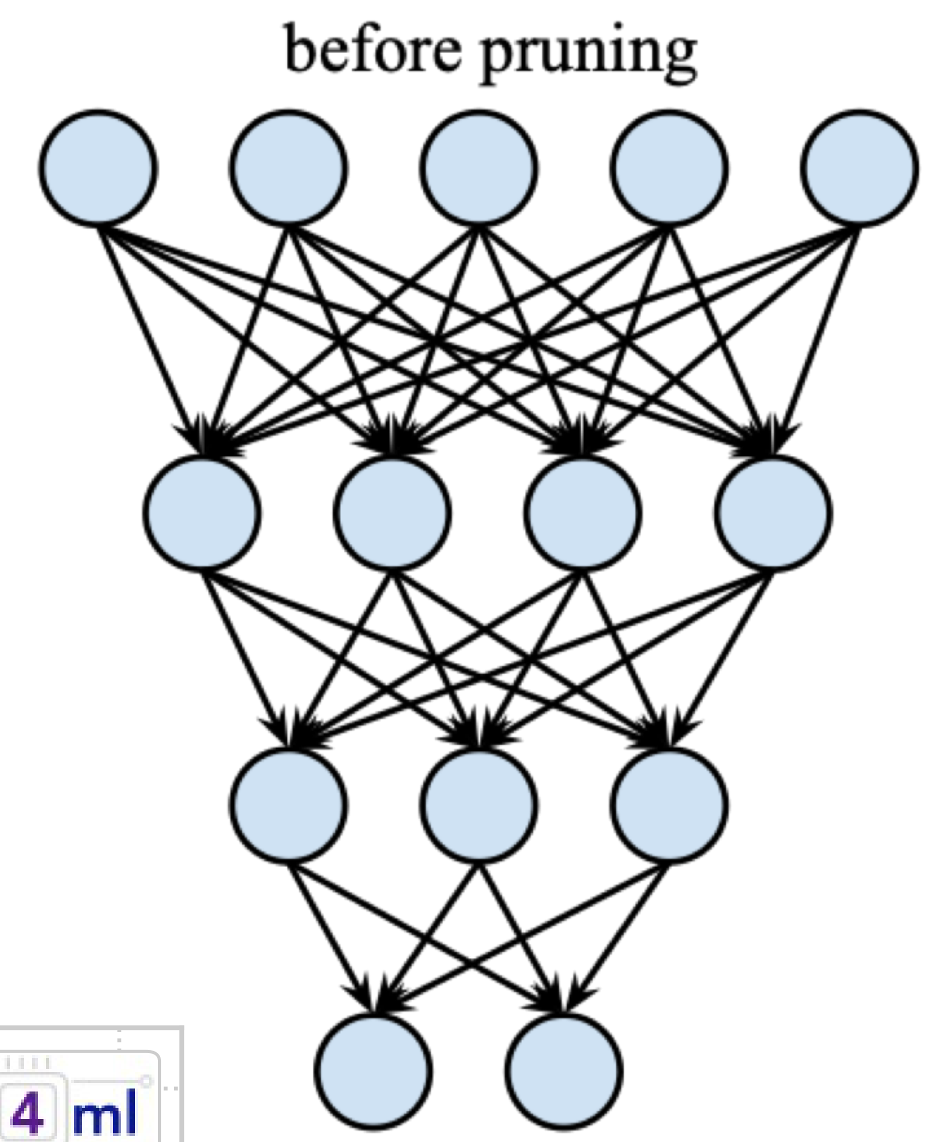
30x

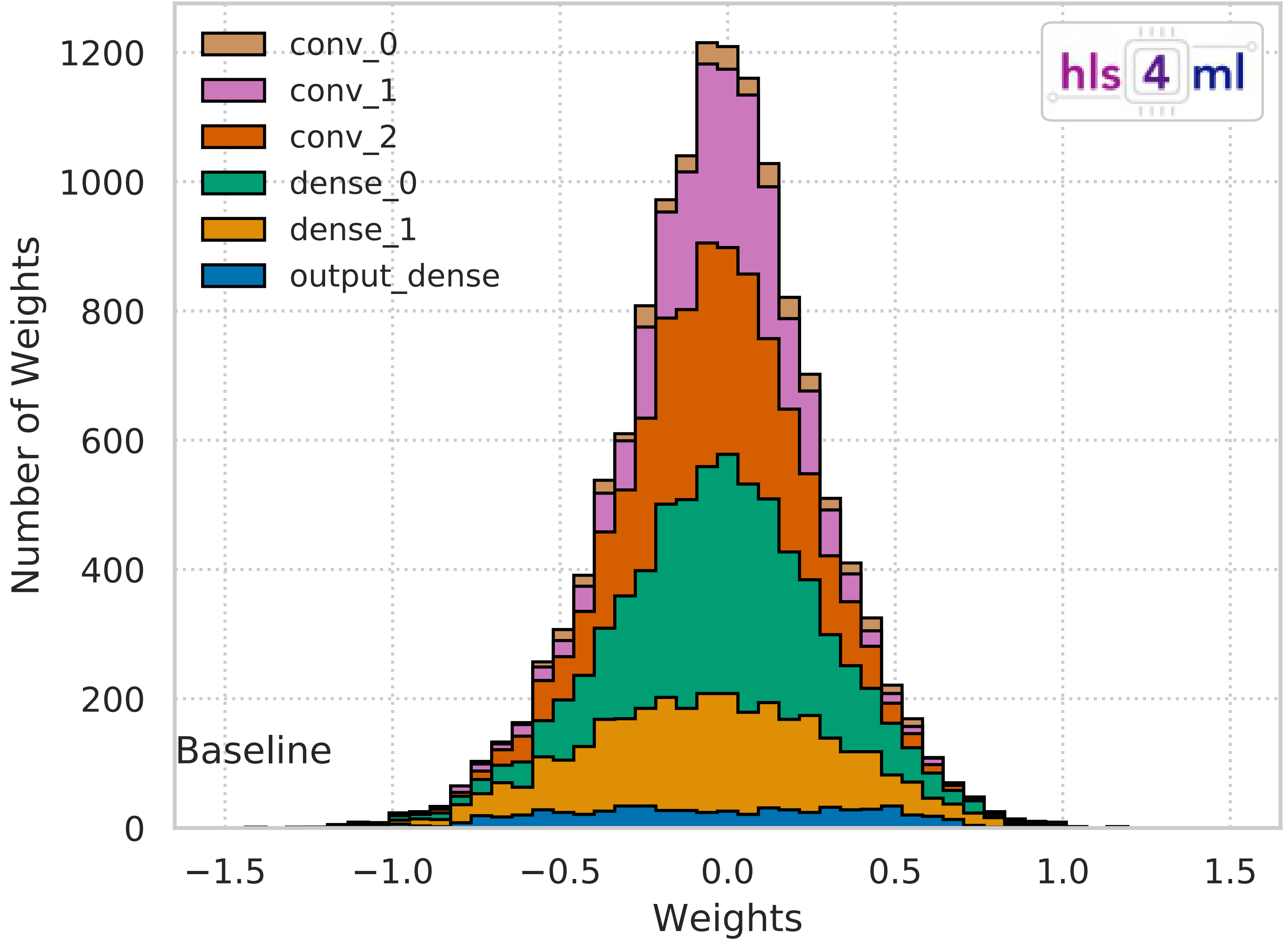
Trillions of quantized operations per second

Weights can stay **entirely on-chip**

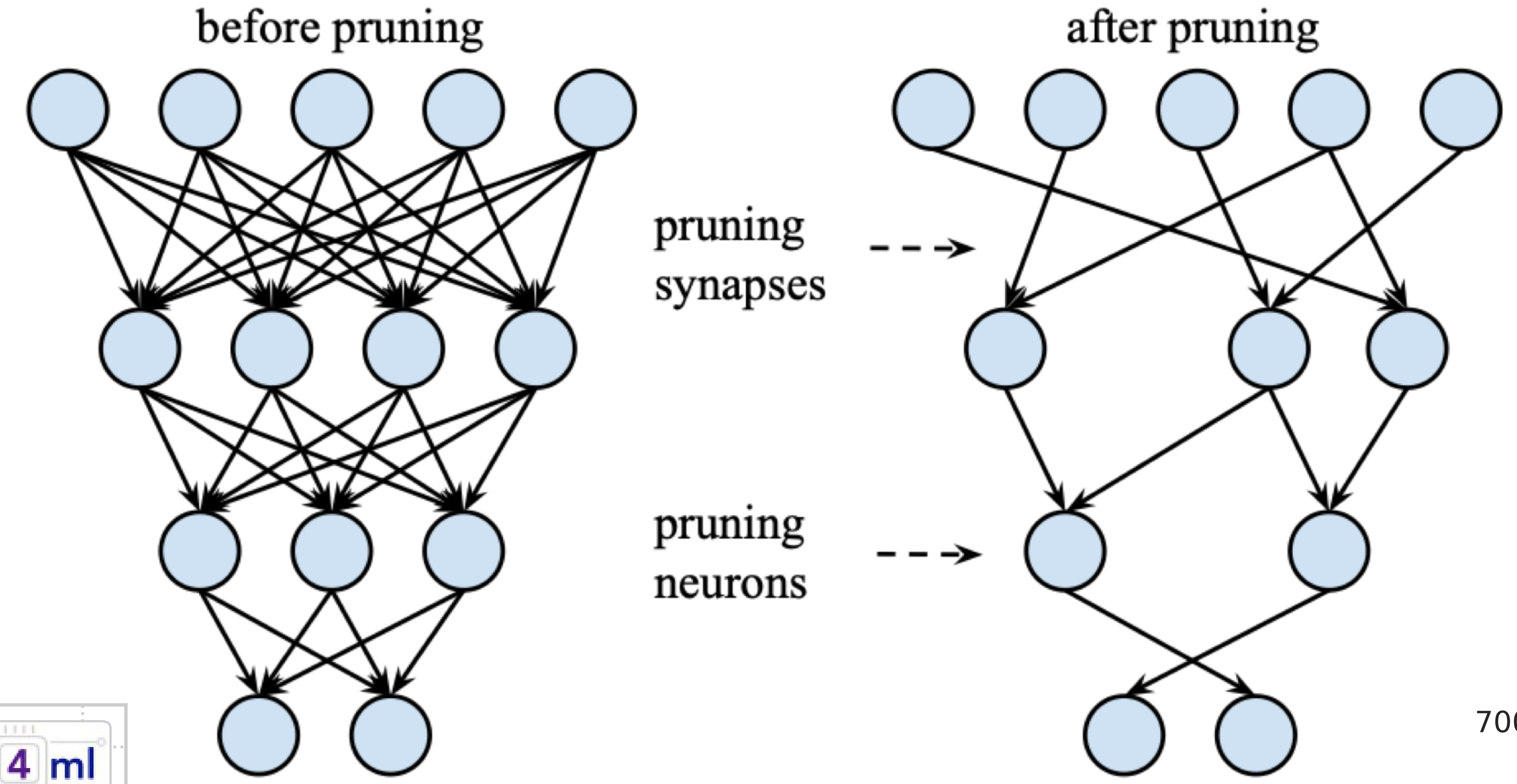
https://blog.tensorflow.org/2019/05/tf-model-optimization-toolkit-pruning-API.html

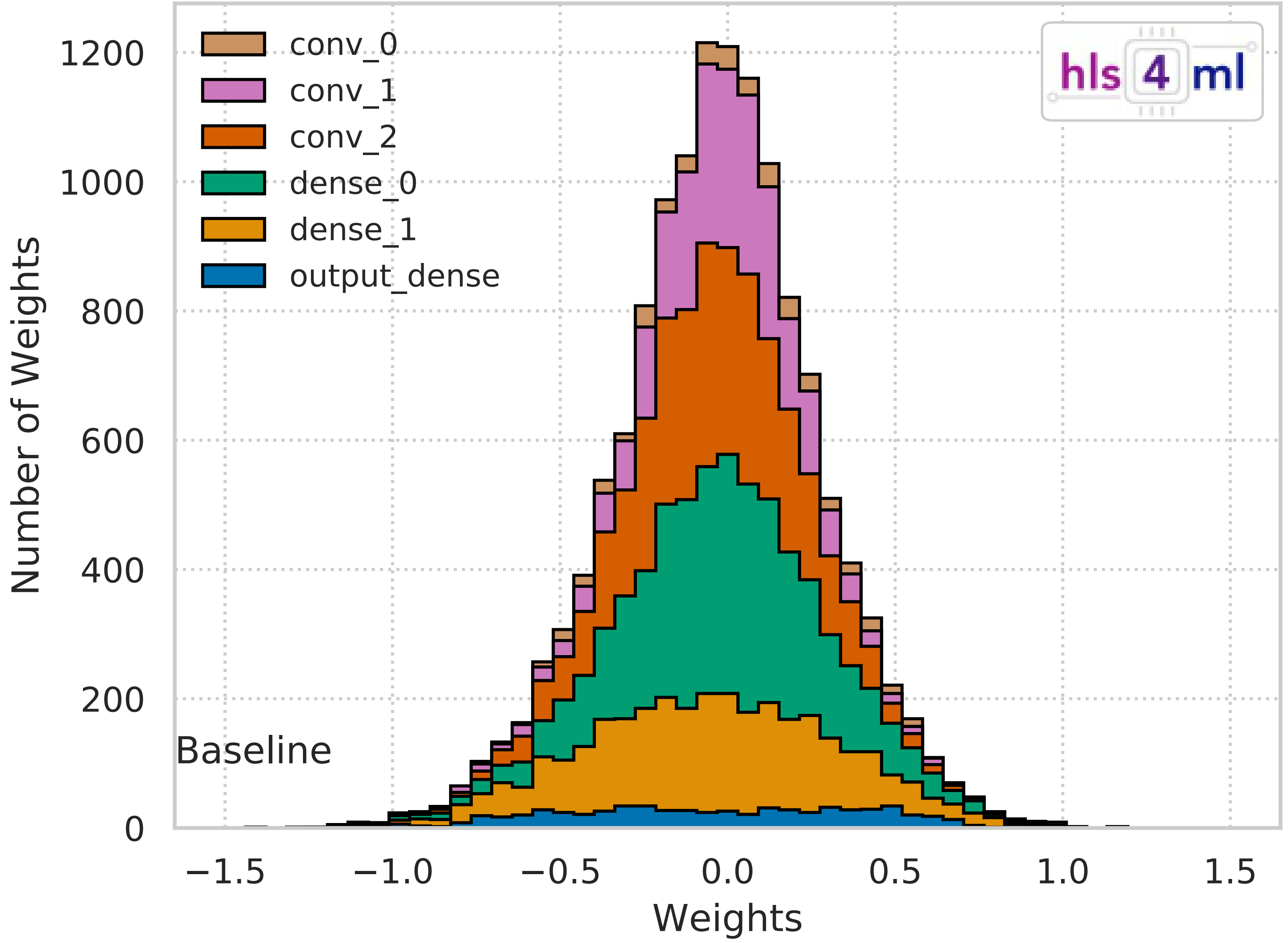
Pruning

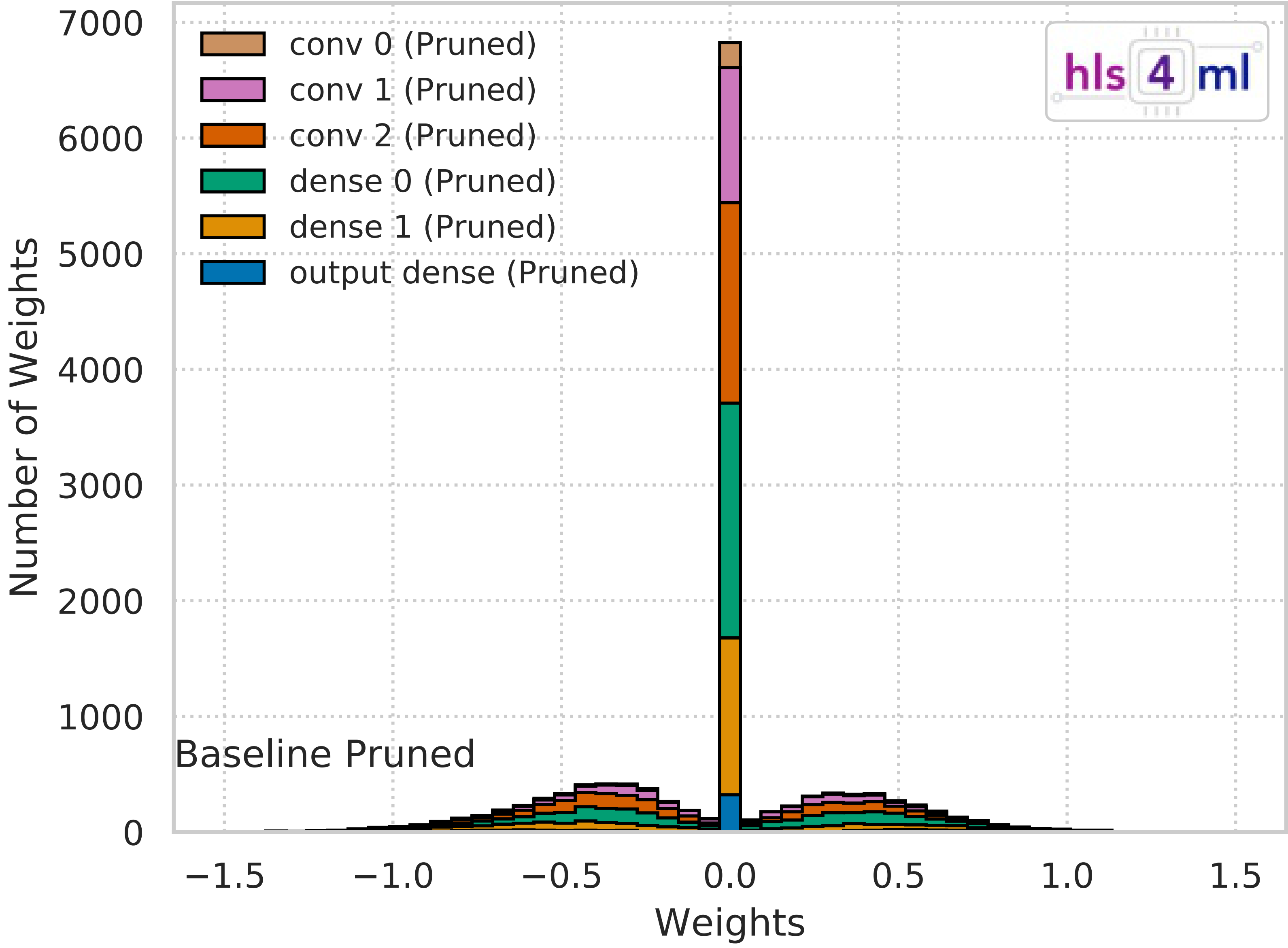




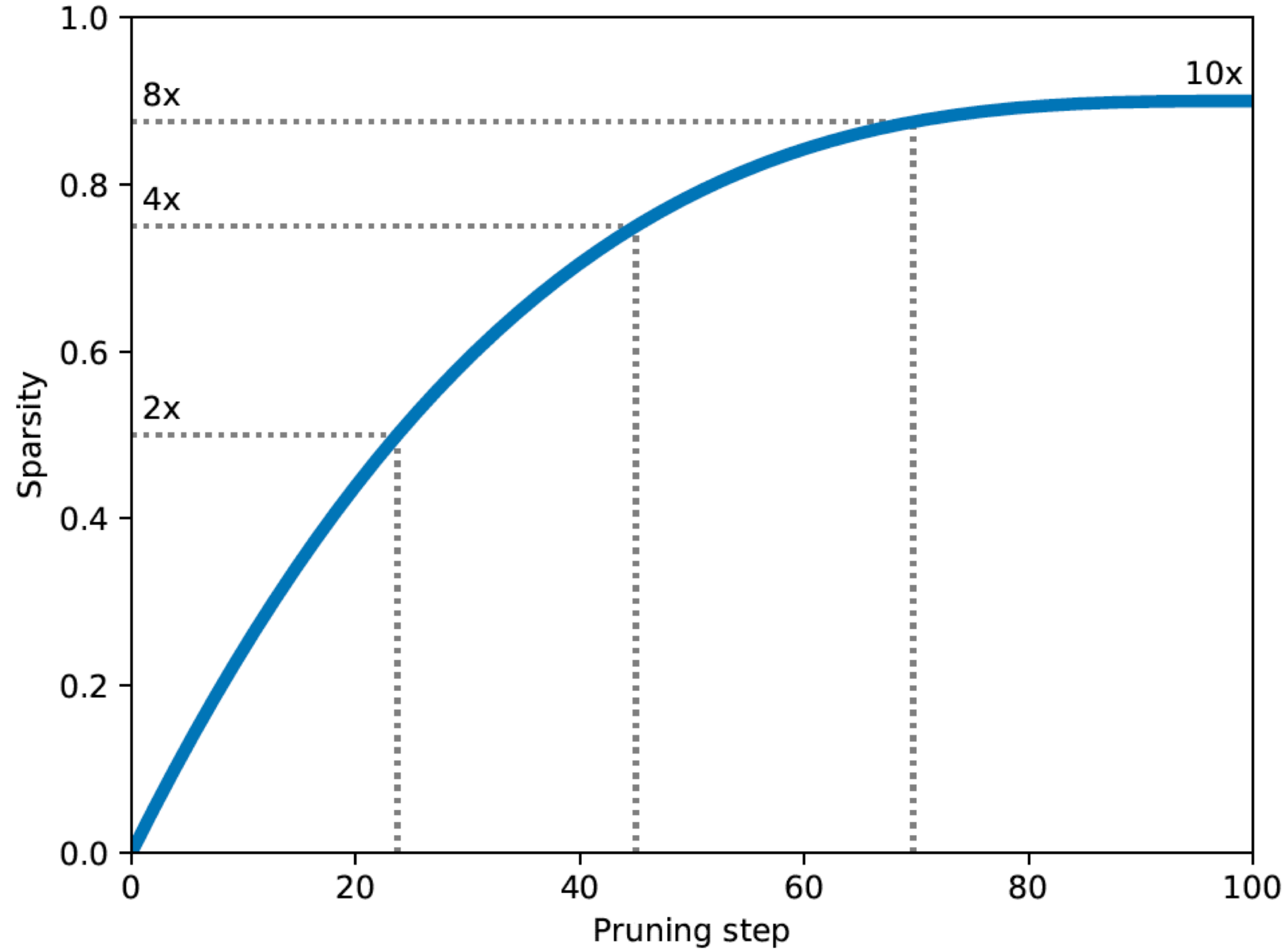
https://blog.tensorflow.org/2019/05/tf-model-optimization-toolkit-pruning-API.html

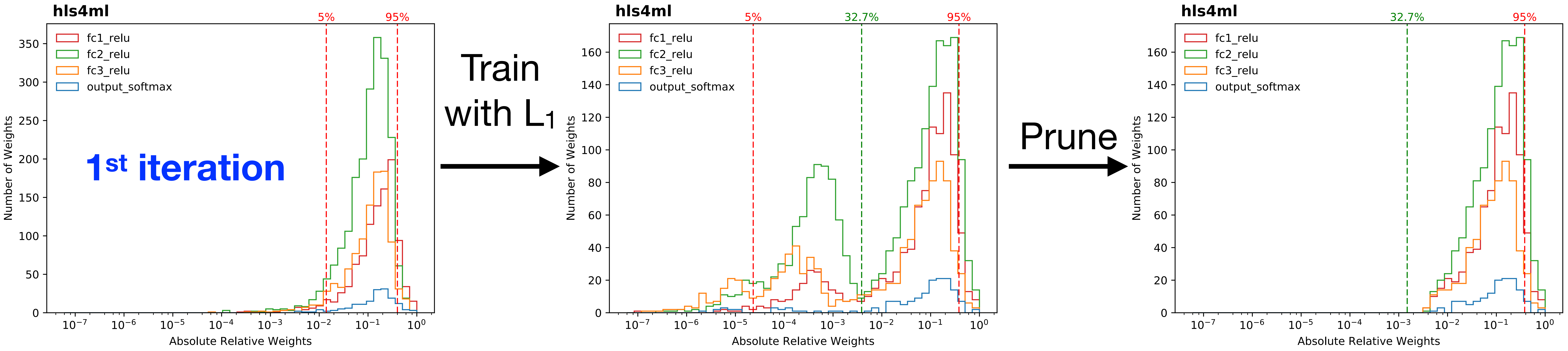






https://blog.tensorflow.org/2019/05/tf-model-optimization-toolkit-pruning-API.html

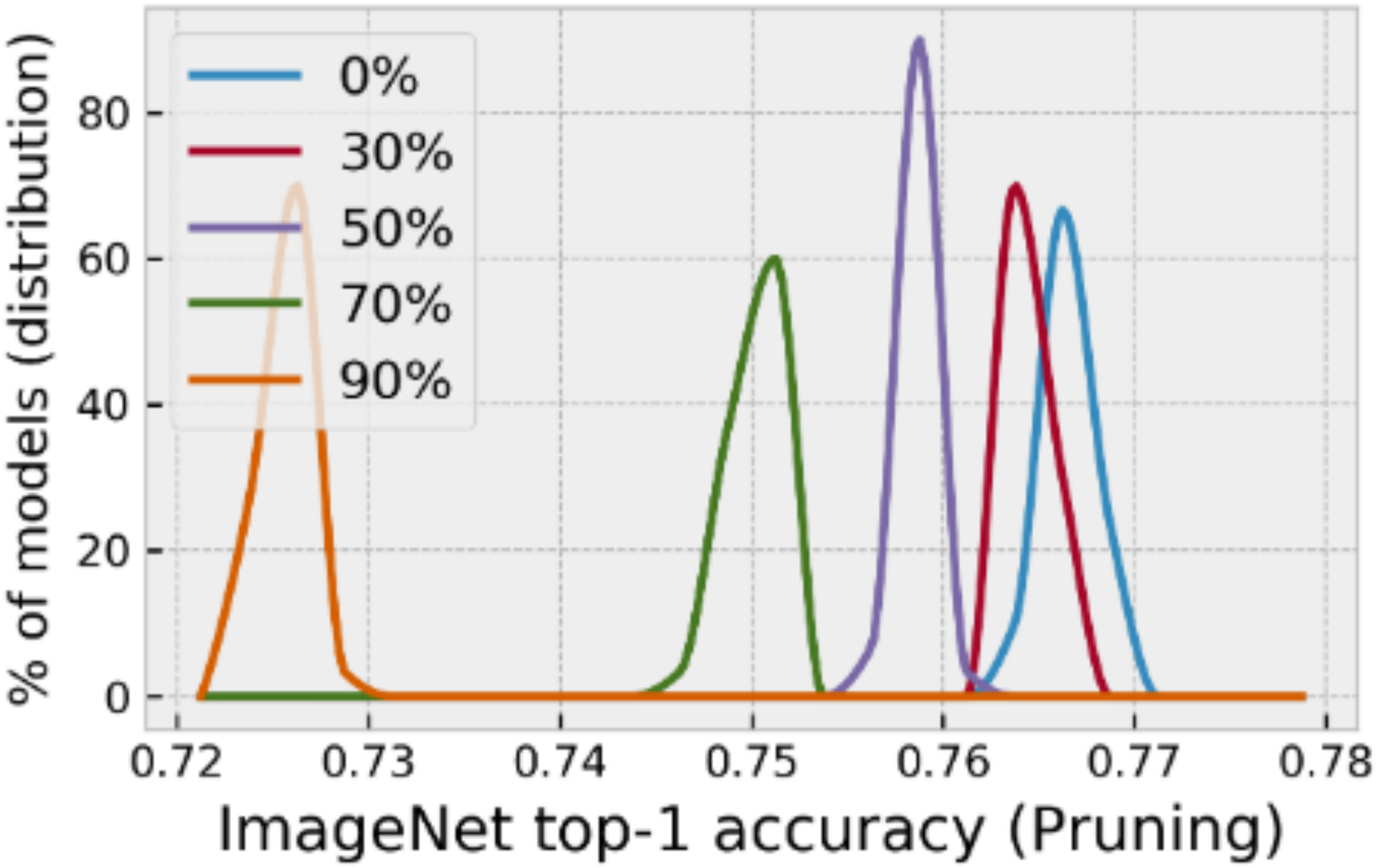




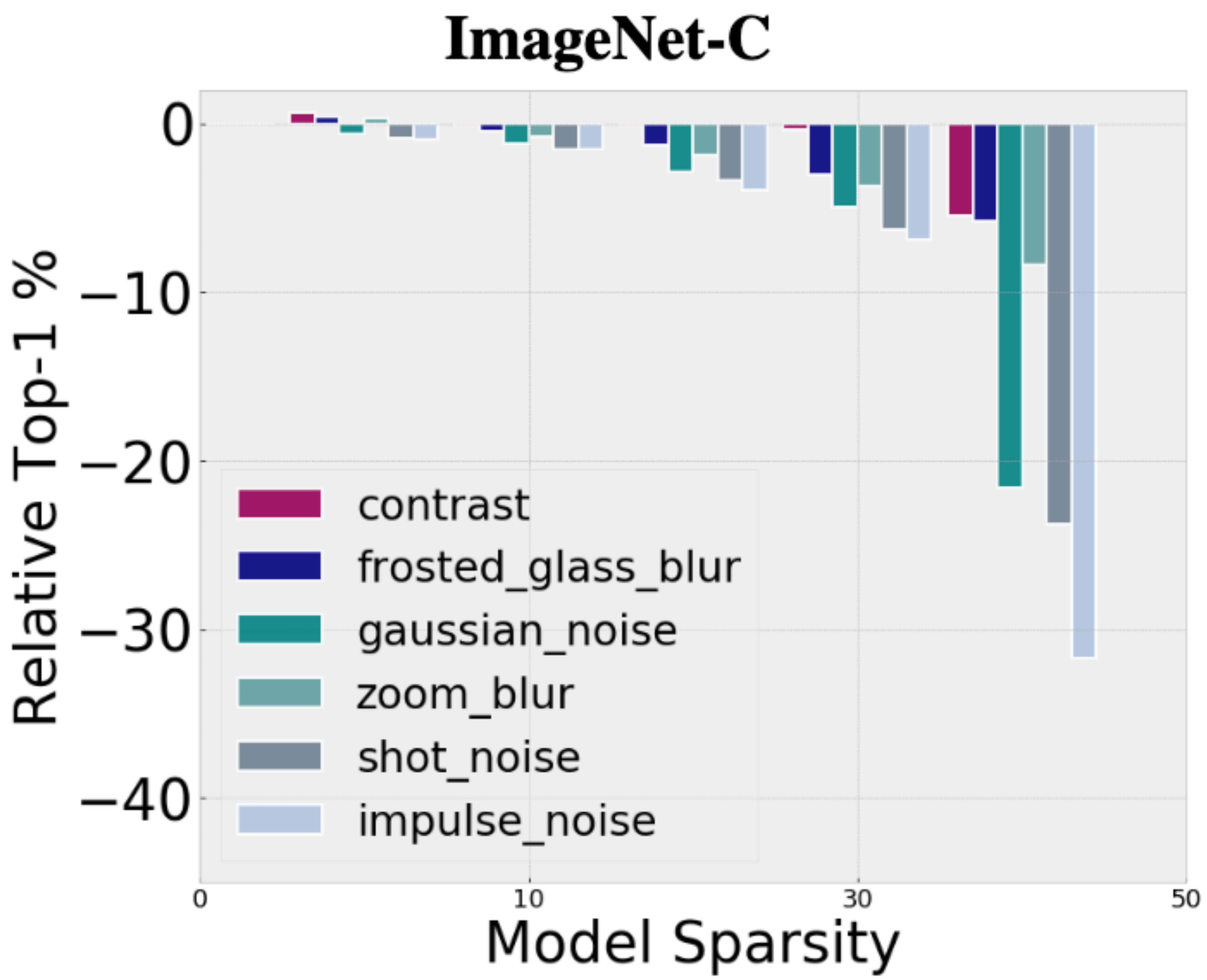
__From Brian Bartoldson__

Hooker et al. (2021)

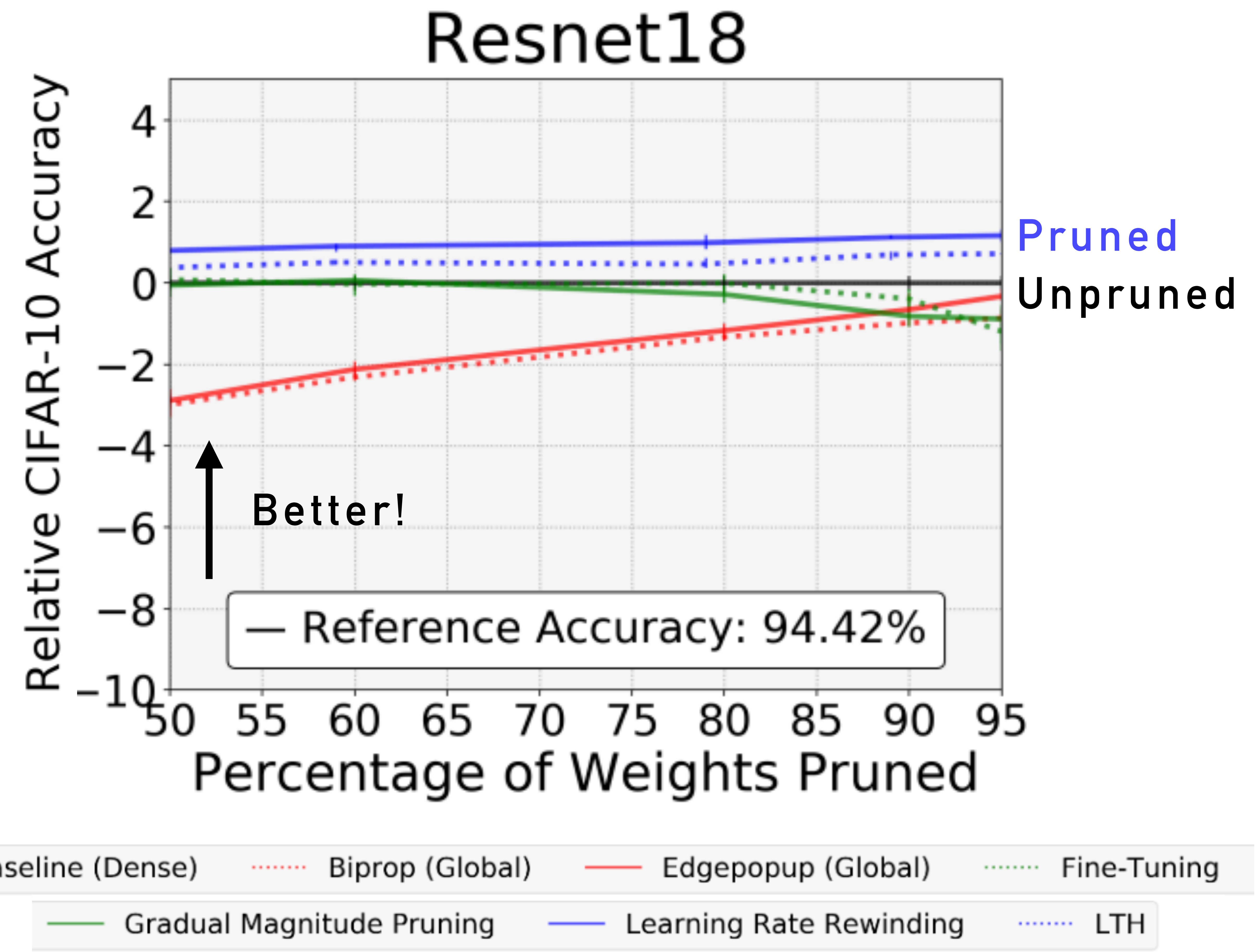
accuracy:



robustness:



Lottery ticket hypothesis



There exists a optimal network WITHIN each network (lottery ticket)
Uncover it through pruning!

Diffenderfer, Bartoldson, et al. (2021)

OpenReview.net

← Go to **NeurIPS 2022 Track Datasets and Benchmarks** h...

Why do tree-based models still outperform deep learning on typical tabular data? PDF

Leo Grinsztajn, Edouard Oyallon, Gael Varoquaux

06 Jun 2022 (modified: 16 Jan 2023) NeurIPS 2022 Datasets and Benchmarks Readers: Everyone Show Bibtex Show Revisions

Abstract: While deep learning has enabled tremendous progress on text and image datasets, its superiority on tabular data is not clear. We contribute extensive benchmarks of standard and novel deep learning methods as well as tree-based models such as XGBoost and Random Forests, across a large number of datasets and hyperparameter combinations. We define a standard set of 45 datasets from varied domains with clear characteristics of tabular data and a benchmarking methodology accounting for both fitting models and finding good hyperparameters. Results show that tree-based models remain state-of-the-art on medium-sized data (~10K samples) even without accounting

Cornell University

arXiv > cs > arXiv:2210.05189

Search... Help | Advanc

Computer Science > Machine Learning

[Submitted on 11 Oct 2022 (v1), last revised 25 Oct 2022 (this version, v3)]

Neural Networks are Decision Trees

Caglar Aytekin

In this manuscript, we show that any neural network with any activation function can be represented as a decision tree. The representation is equivalence and not an approximation, thus keeping the accuracy of the neural network exactly as is. We believe that this work provides better understanding of neural networks and paves the way to tackle their black-box nature. We share equivalent trees of some neural networks and show that besides providing interpretability, tree representation can also achieve some computational advantages for small networks. The analysis holds both for fully connected and convolutional networks, which may or may not also include skip connections and/or normalizations.

Subjects: **Machine Learning (cs.LG)**

Cite as: arXiv:2210.05189 **[cs.LG]**

(or arXiv:2210.05189v3 **[cs.LG]** for this version)

https://doi.org/10.48550/arXiv.2210.05189

Submission history

From: Çağlar Aytekin [view email]

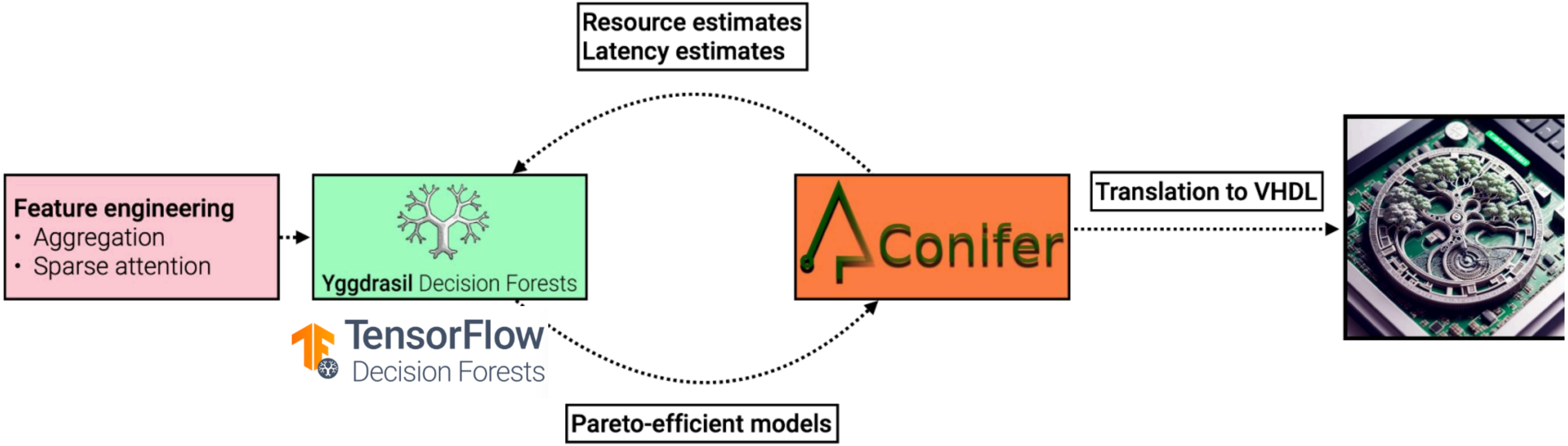
[v1] Tue, 11 Oct 2022 06:49:51 UTC (216 KB)

[v2] Mon, 17 Oct 2022 15:18:14 UTC (224 KB)

[v3] Tue, 25 Oct 2022 17:32:33 UTC (240 KB)

https://arxiv.org/abs/2210.05189

%VU9P	Accuracy	Latency	DSP	LUT
qDNN	75.6%	40 ns	22 (~0%)	1%
BDT	74.9%	5 ns	-	0.5%



Yggdrasil+Conifer

Very high quality data can help smaller models compete

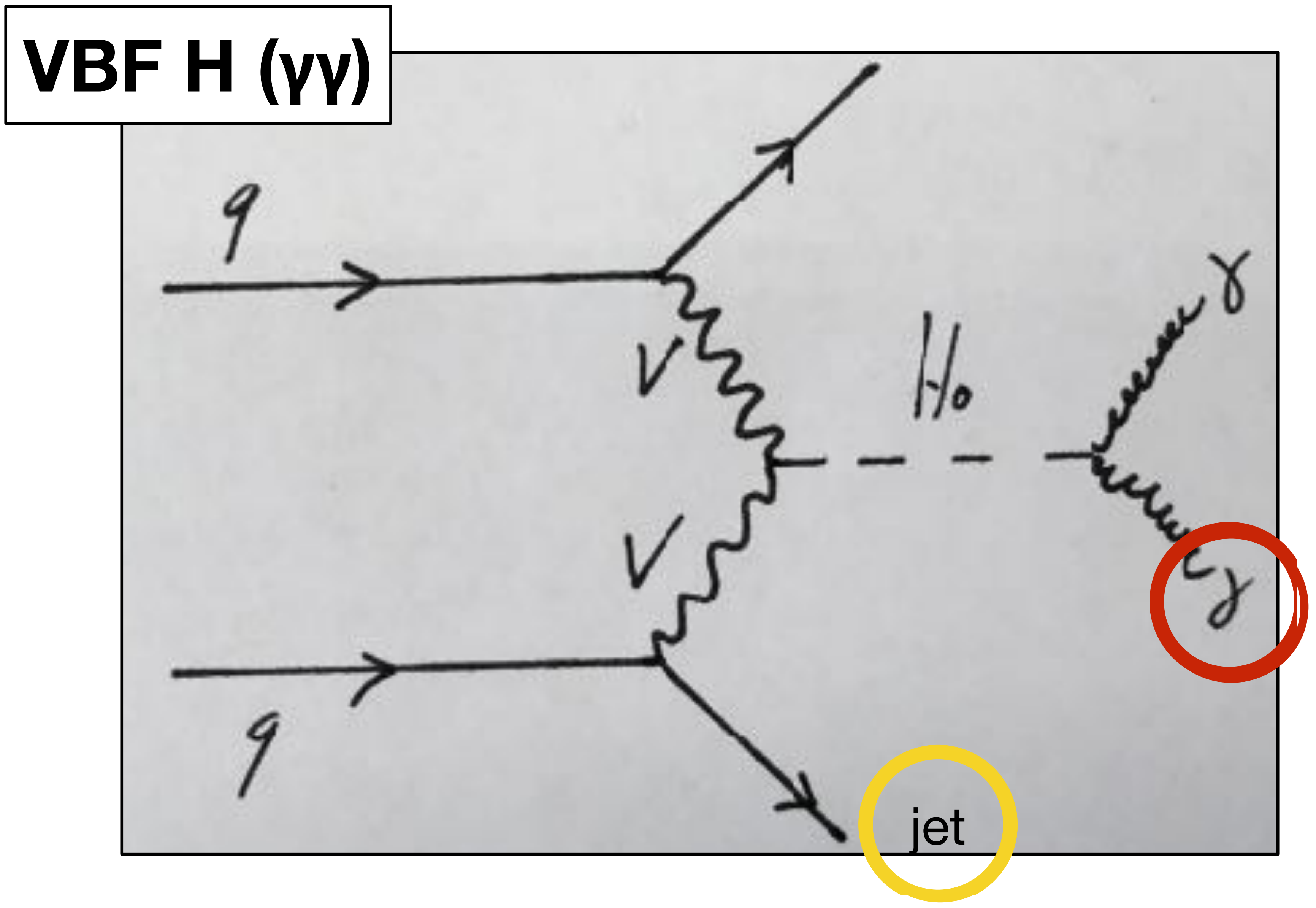
Date	Model	Model size	Dataset size (Tokens)	HumanEval (Pass@1)	MBPP (Pass@1)
2021 Jul	Codex-300M [CTJ+21]	300M	100B	13.2%	-
2021 Jul	Codex-12B [CTJ+21]	12B	100B	28.8%	-
2022 Mar	CodeGen-Mono-350M [NPH+23]	350M	577B	12.8%	-
2022 Mar	CodeGen-Mono-16.1B [NPH+23]	16.1B	577B	29.3%	35.3%
2022 Apr	PaLM-Coder [CND+22]	540B	780B	35.9%	47.0%
2022 Sep	CodeGeeX [ZXZ+23]	13B	850B	22.9%	24.4%
2022 Nov	GPT-3.5 [Ope23]	175B	N.A.	47%	-
2022 Dec	SantaCoder [ALK+23]	1.1B	236B	14.0%	35.0%
2023 Mar	GPT-4 [Ope23]	N.A.	N.A.	67%	-
2023 Apr	Replit [Rep23]	2.7B	525B	21.9%	-
2023 Apr	Replit-Finetuned [Rep23]	2.7B	525B	30.5%	-
2023 May	CodeGen2-1B [NHX+23]	1B	N.A.	10.3%	-
2023 May	CodeGen2-7B [NHX+23]	7B	N.A.	19.1%	-
2023 May	StarCoder [LAZ+23]	15.5B	1T	33.6%	52.7%
2023 May	StarCoder-Prompted [LAZ+23]	15.5B	1T	40.8%	49.5%
2023 May	PaLM 2-S [ADF+23]	N.A.	N.A.	37.6%	50.0%
2023 May	CodeT5+ [WLG+23]	2B	52B	24.2%	-
2023 May	CodeT5+ [WLG+23]	16B	52B	30.9%	-
2023 May	InstructCodeT5+ [WLG+23]	16B	52B	35.0%	-
2023 Jun	WizardCoder [LXZ+23]	16B	1T	57.3%	51.8%
2023 Jun	**phi-1**	1.3B	7B	50.6%	55.5%

Competitive despite being 10x smaller in model size and 100x smaller in dataset size!

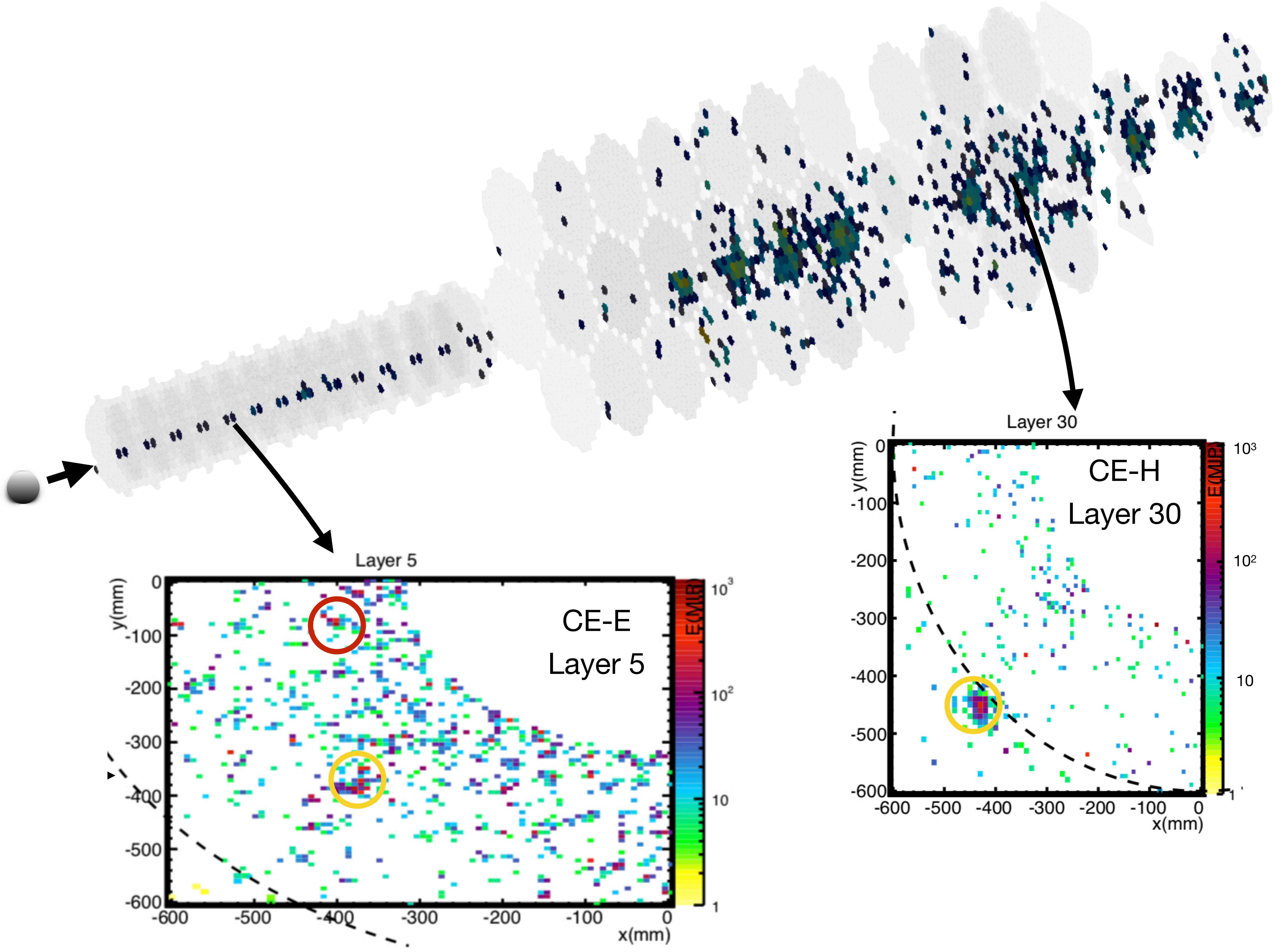
Gunasekar et al. (2023)

CMS High Granularity calorimeter

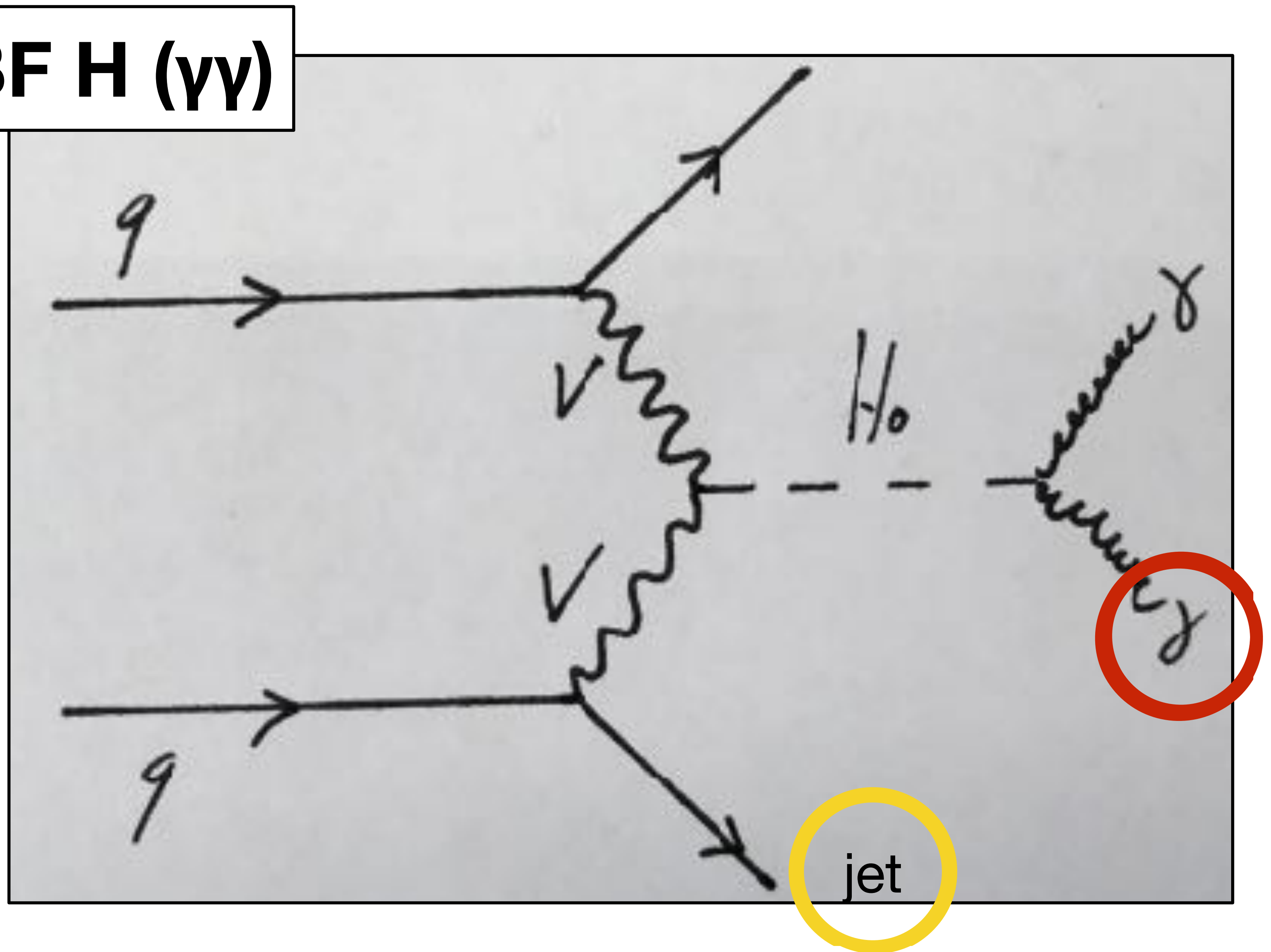
- 6.5 million readout channels, 50 layers

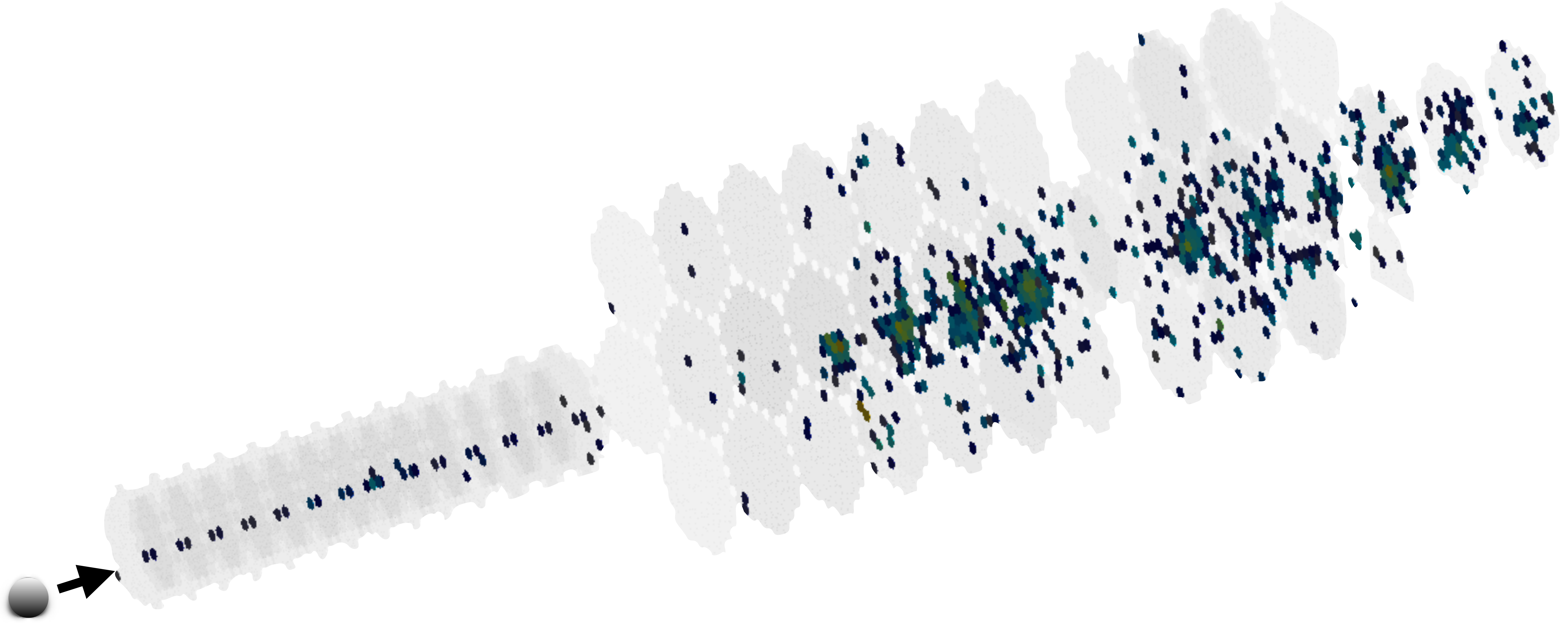


+



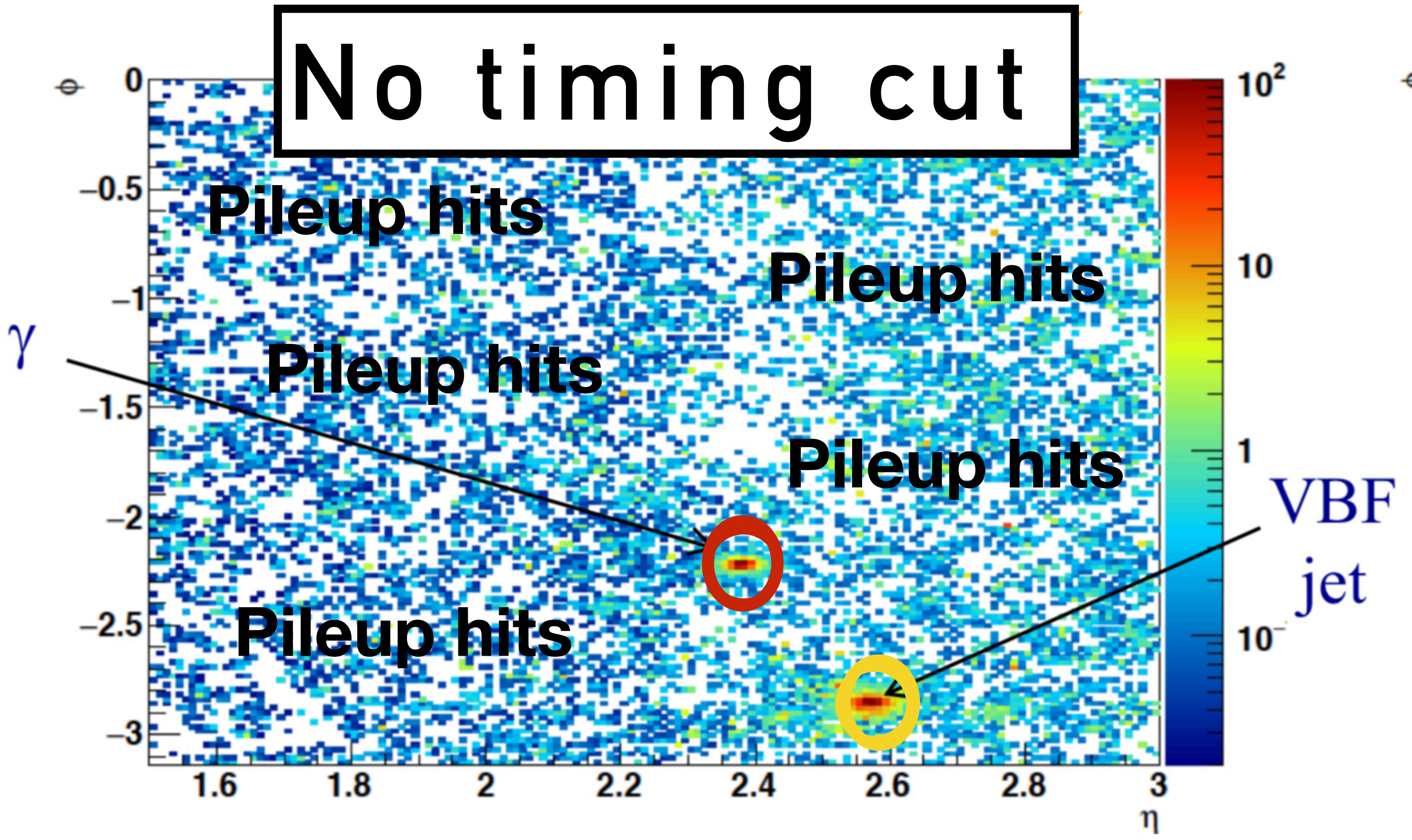
VBF H (γγ)

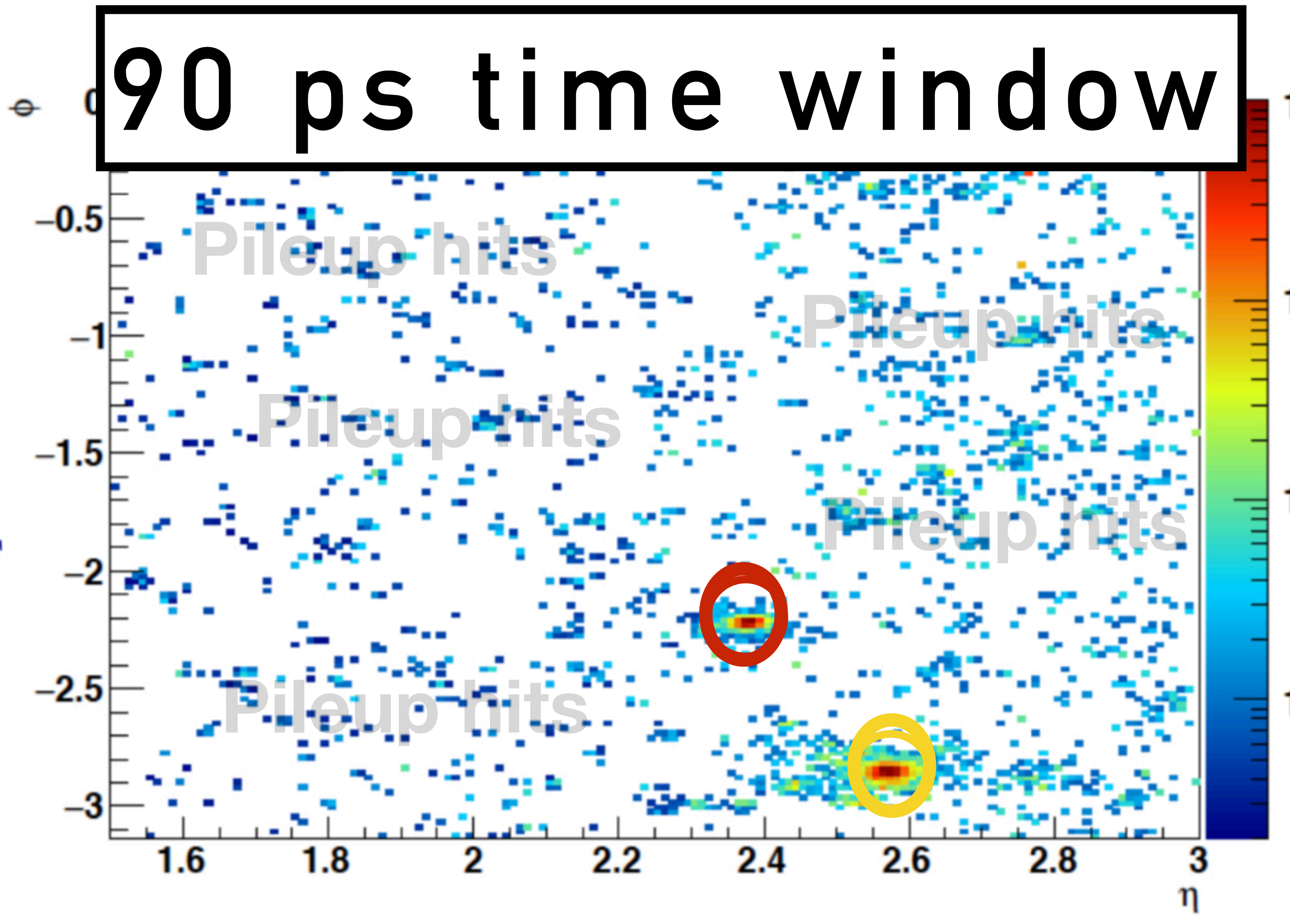

q
q
V
V
H_0
γ
γ
jet



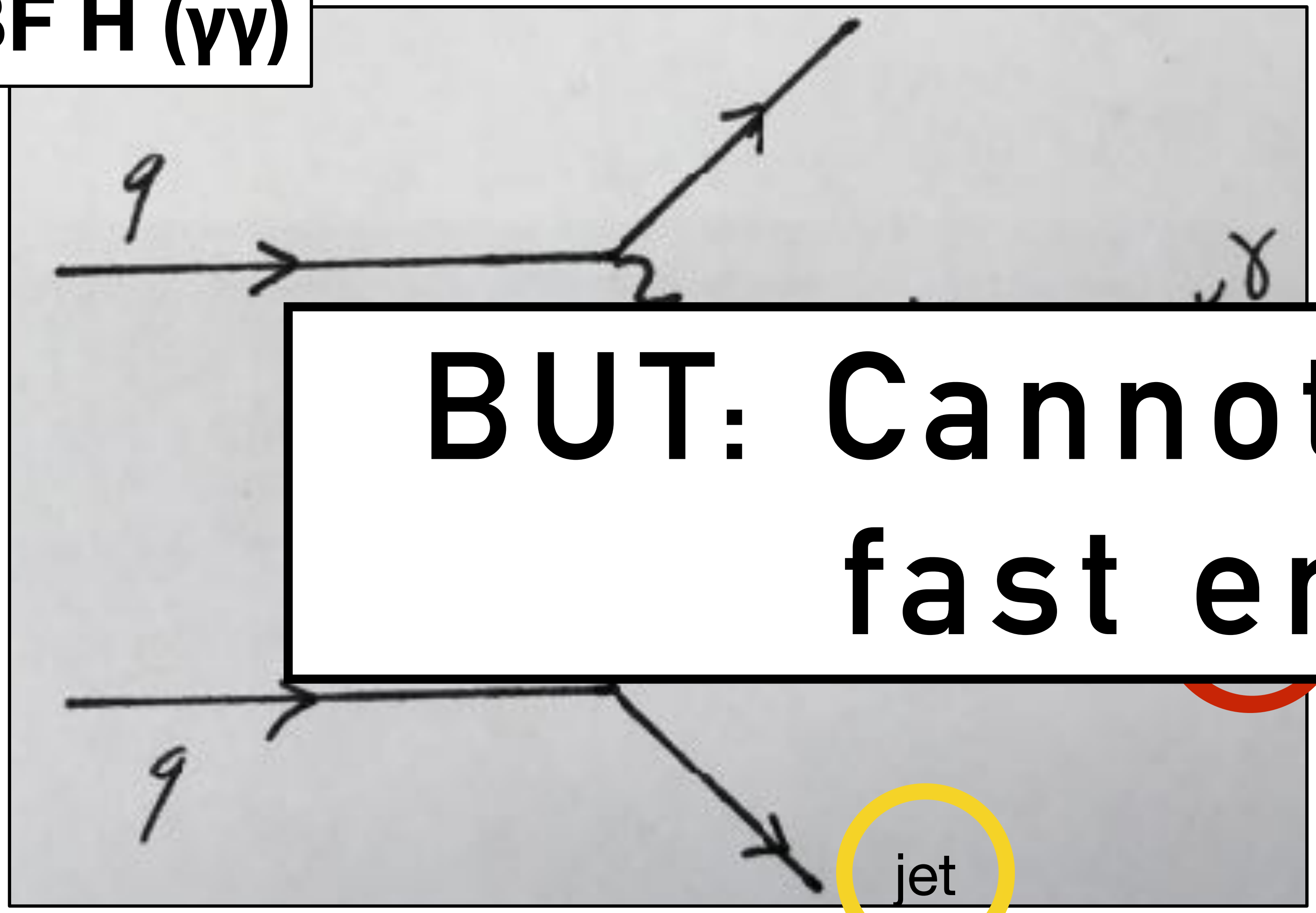
+

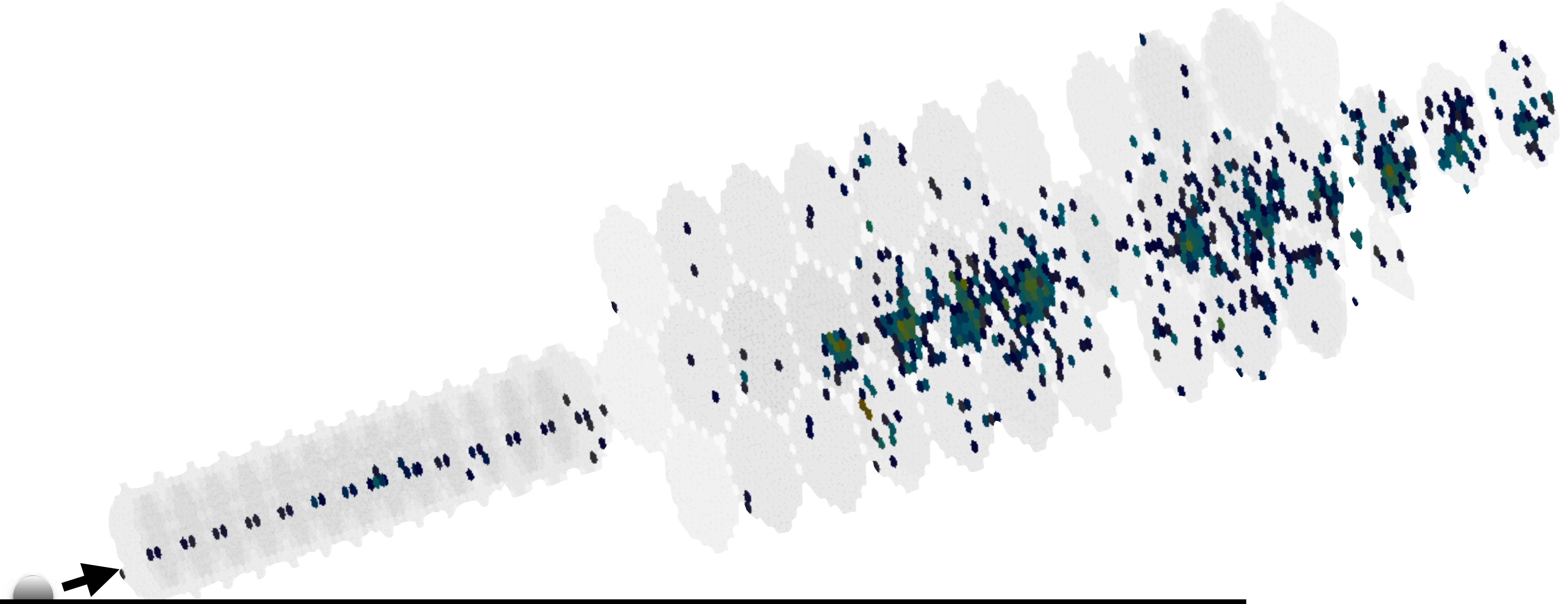

200 vertices


No timing cut
Pileup hits
Pileup hits
Pileup hits
Pileup hits
Pileup hits
γ
VBF
jet
φ
η


90 ps time window
Pileup hits
Pileup hits
Pileup hits
Pileup hits
Pileup hits
φ
η

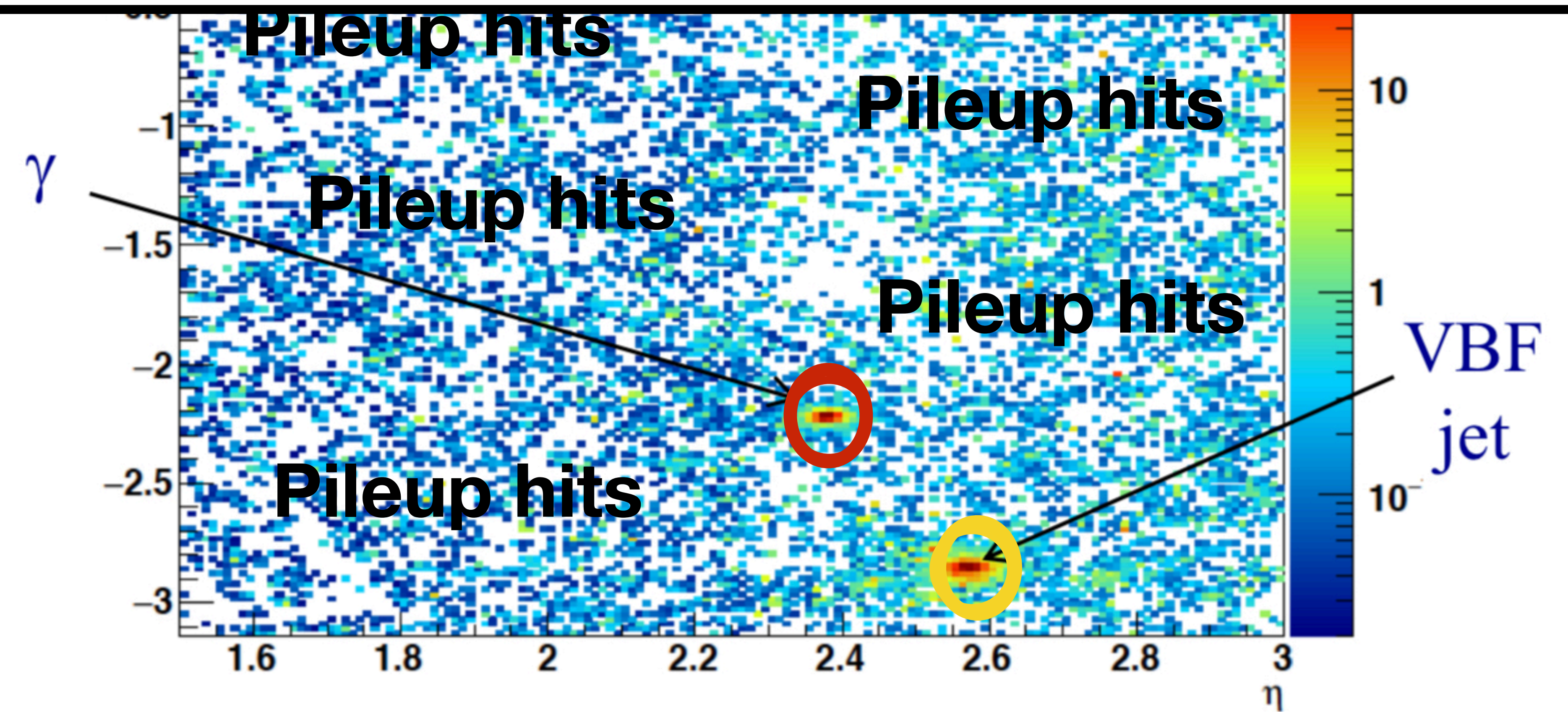
VBF H (γγ)



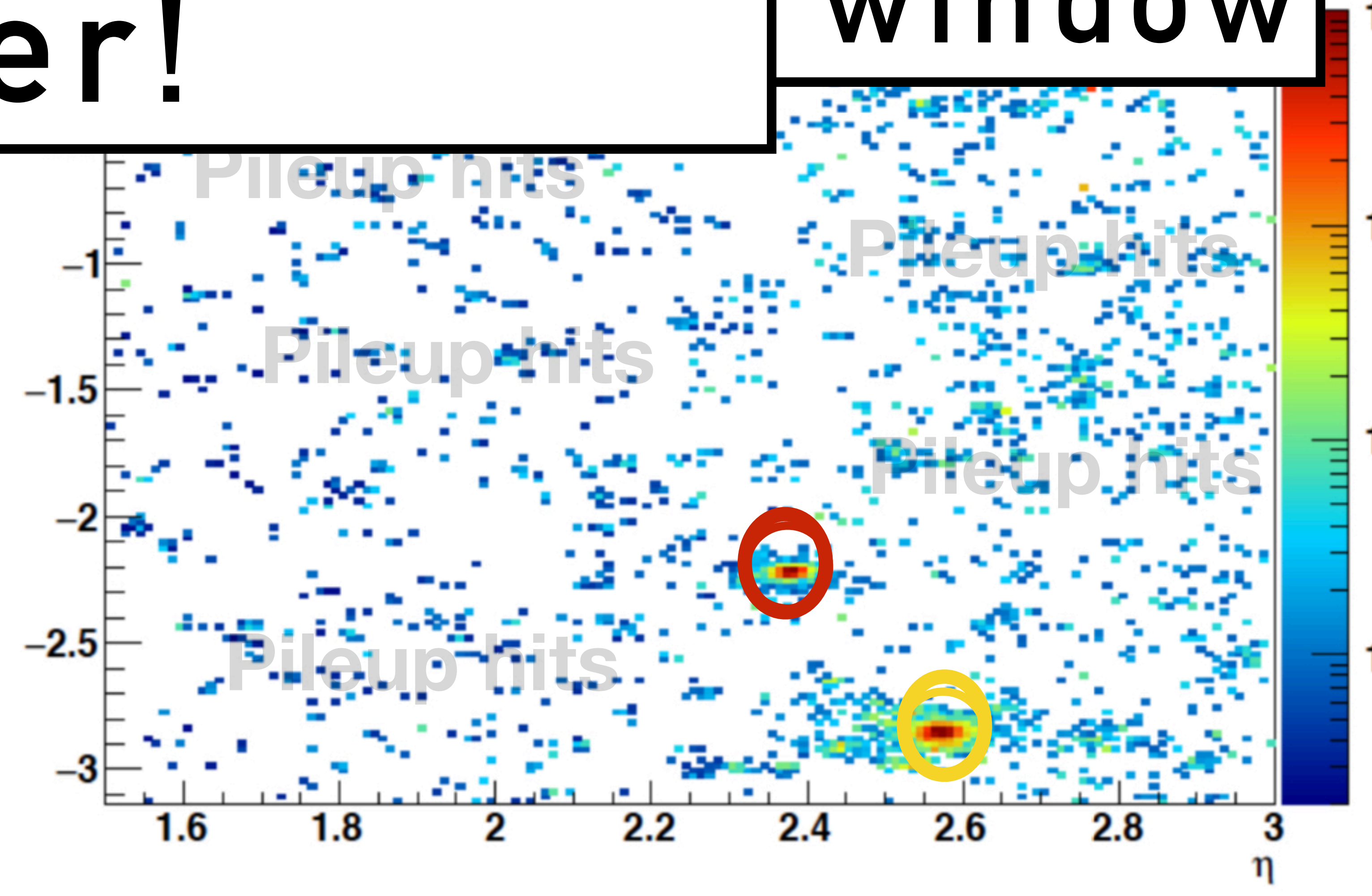


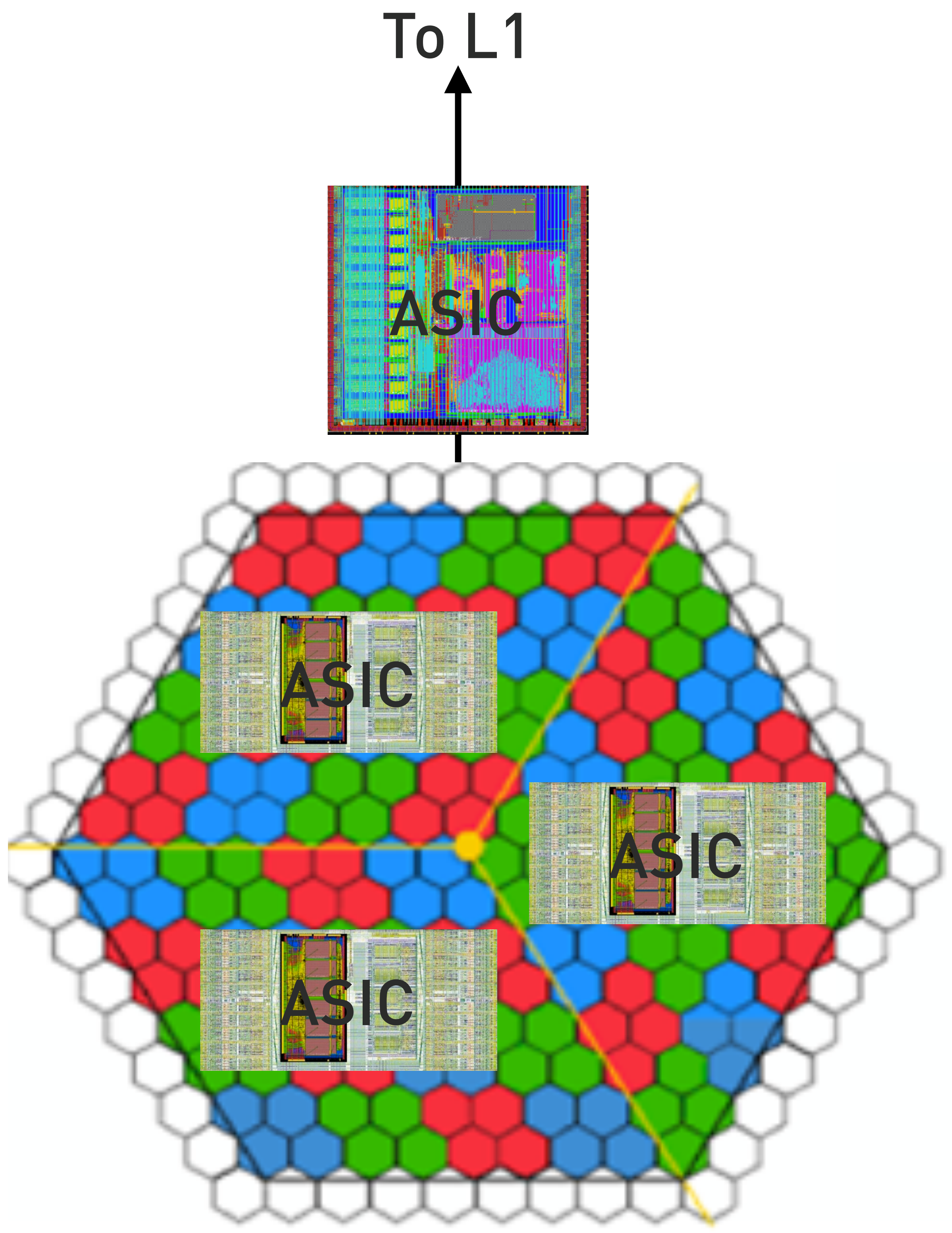
BUT: Cannot read out all these channels fast enough for L1 to trigger!

+



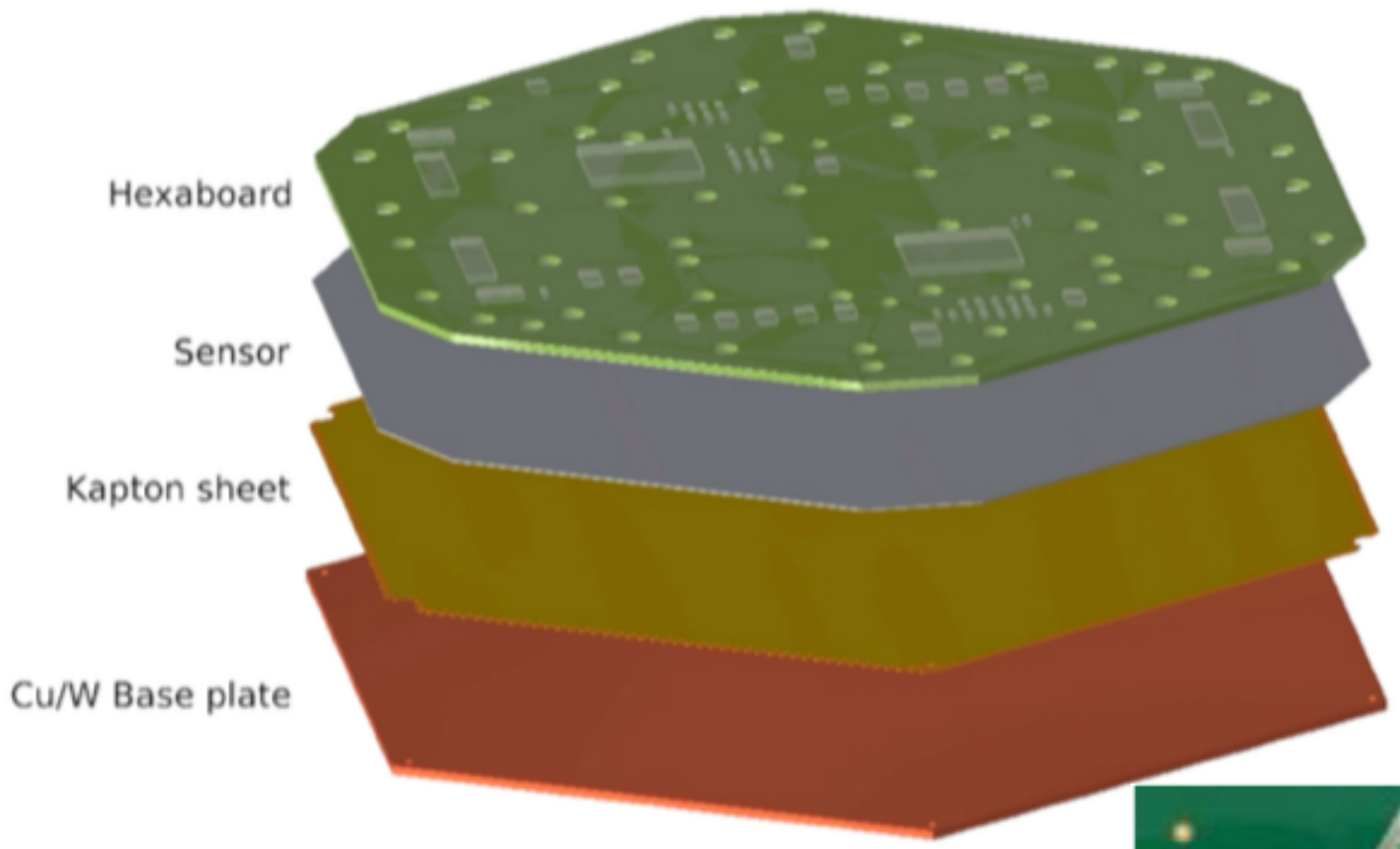
window

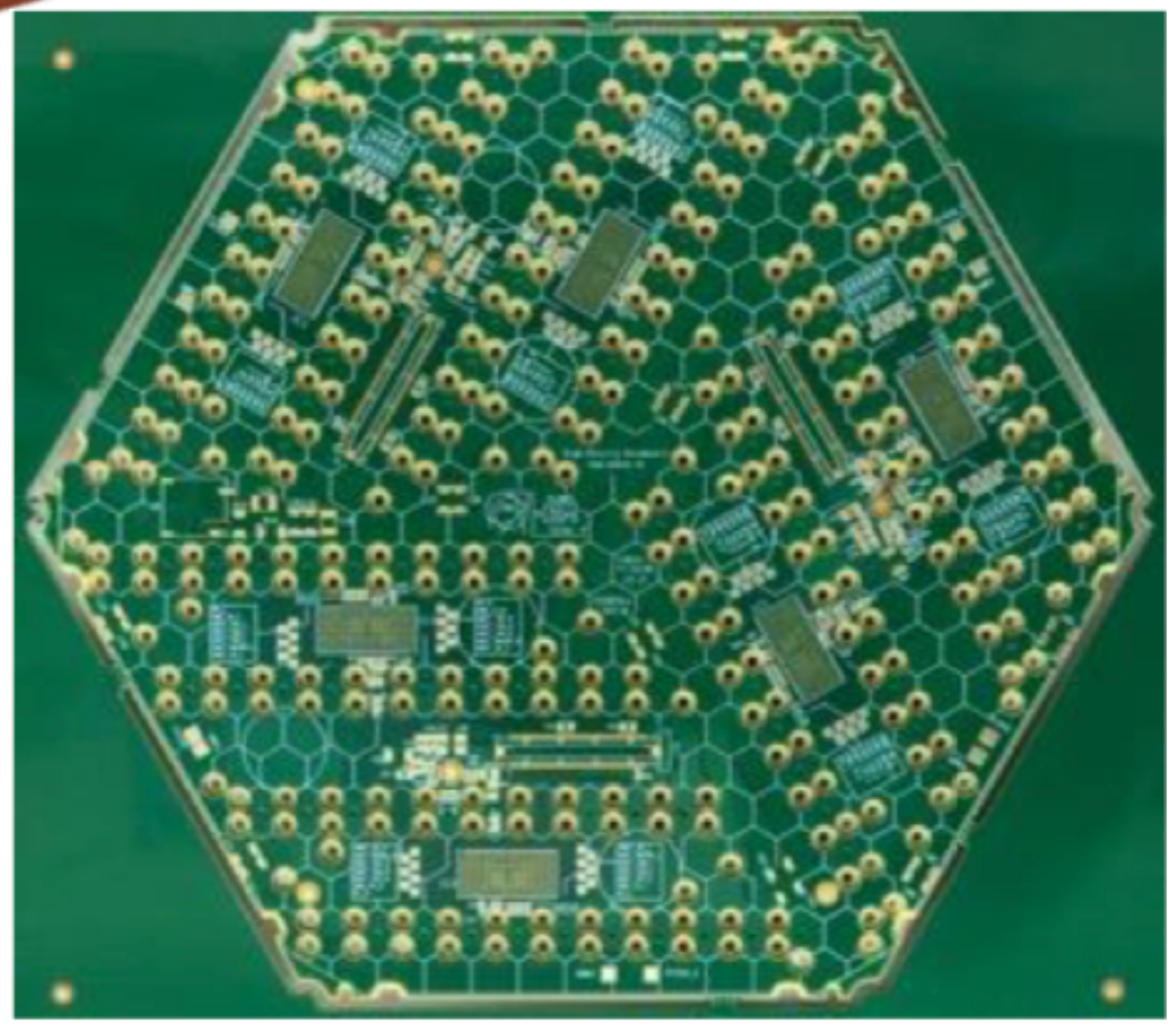


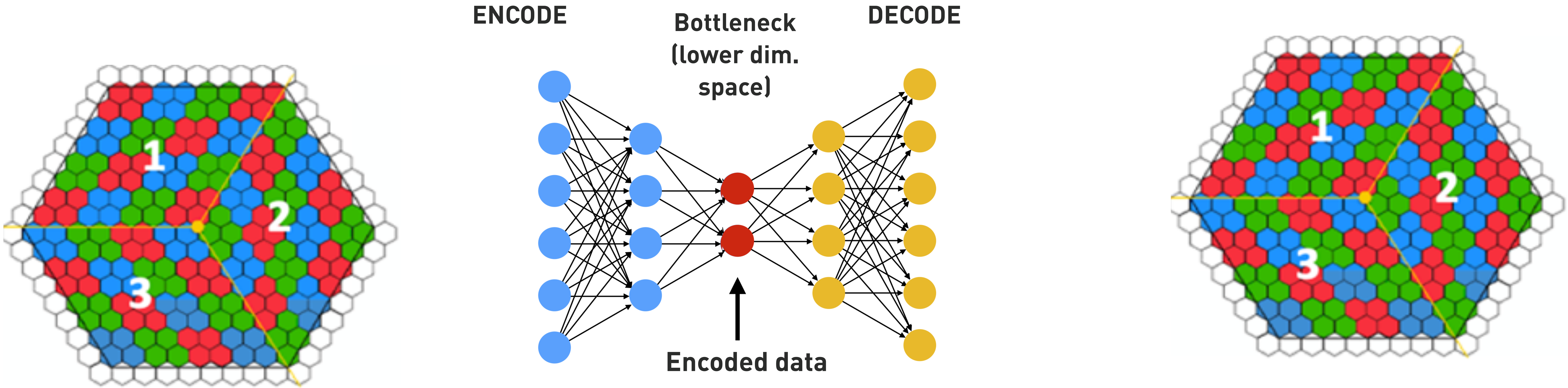


Compress data ON the detector

- ASICs reduce + transmit data
  - 40 MHz trigger data
  - 750 kHz DAQ data
- High radiation
- Cooled to -30 → low power (Max 500 mW total)
- 1.5 μs latency



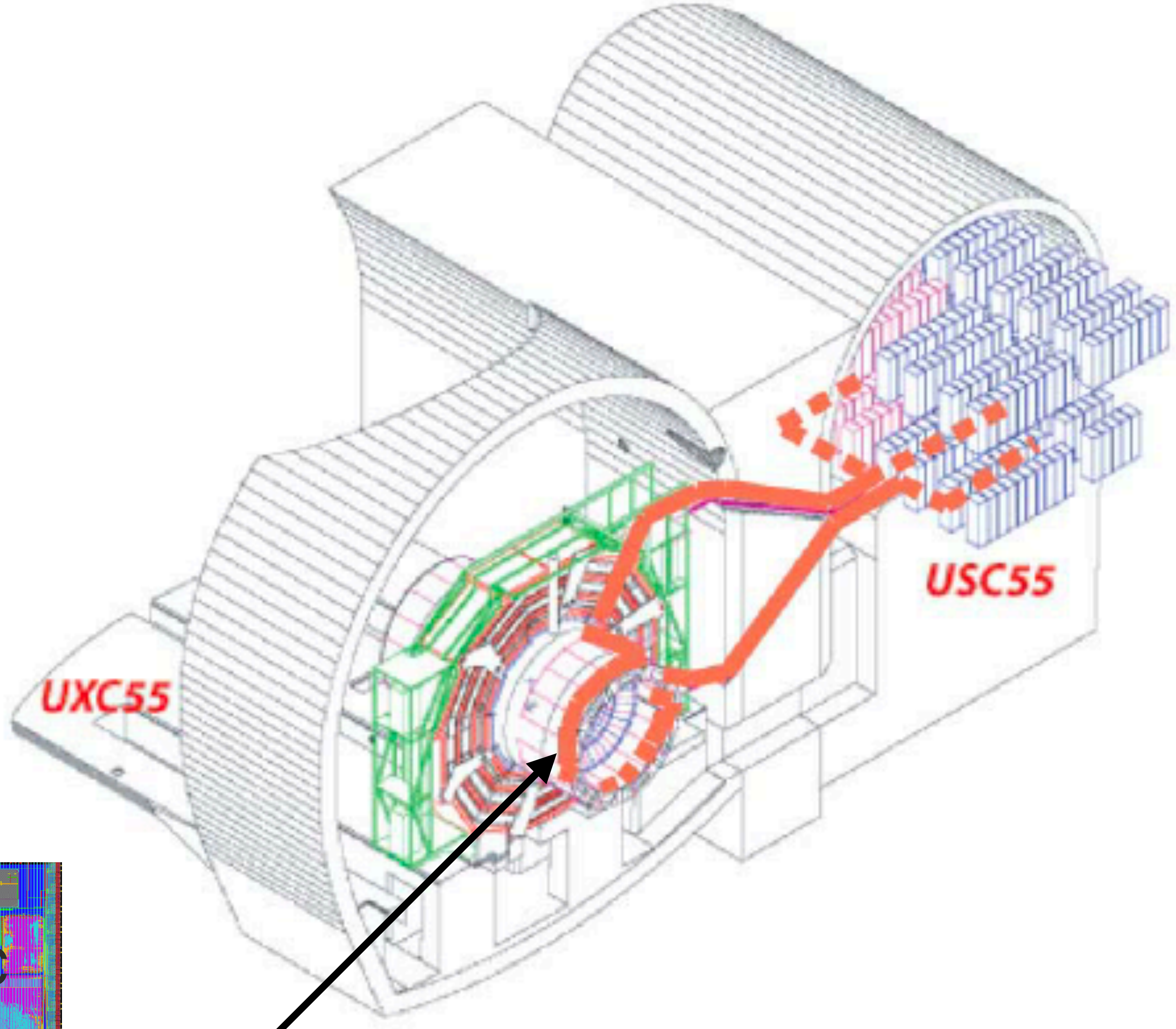


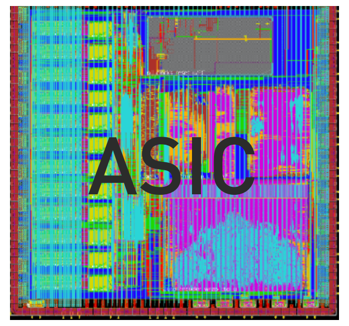


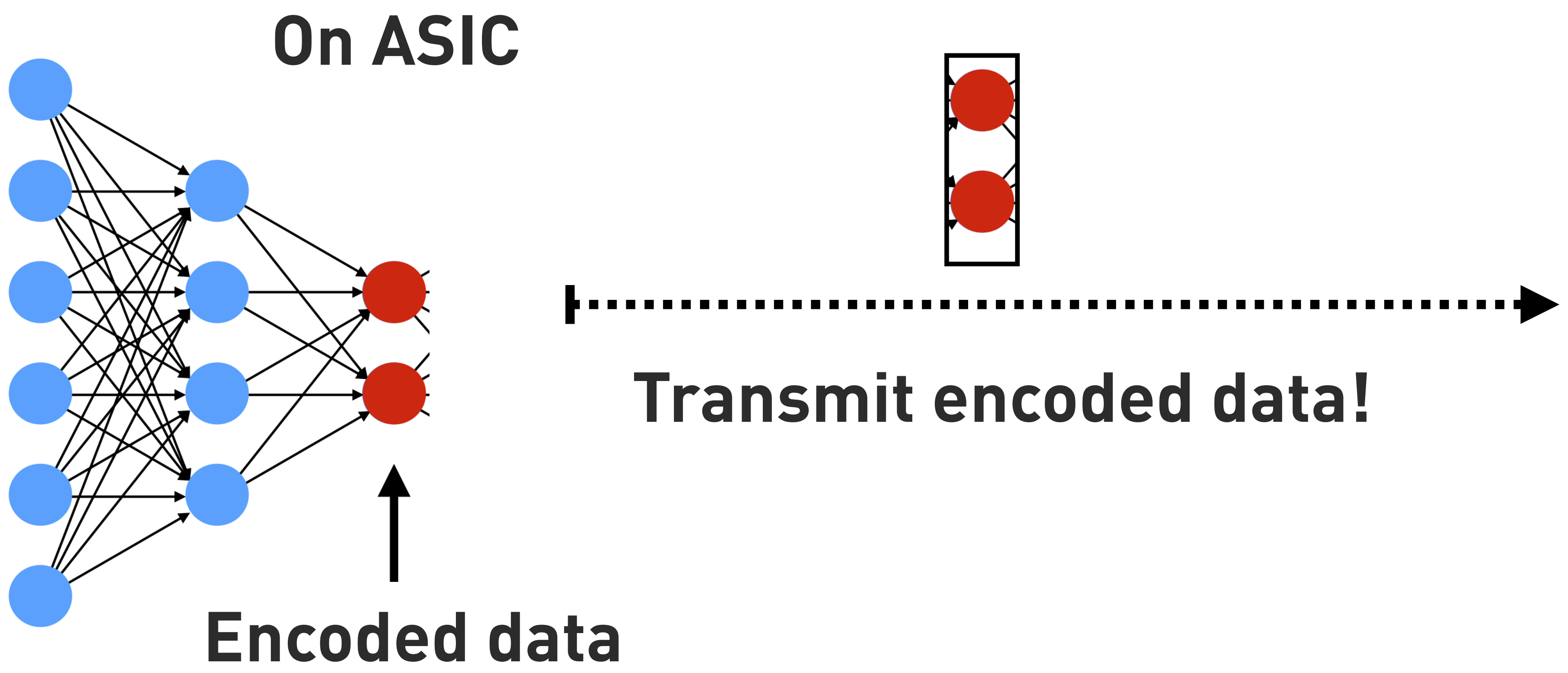
Variational Autoencoder

ECON-T, D. Noonan

AEs for compression also at LHCb!





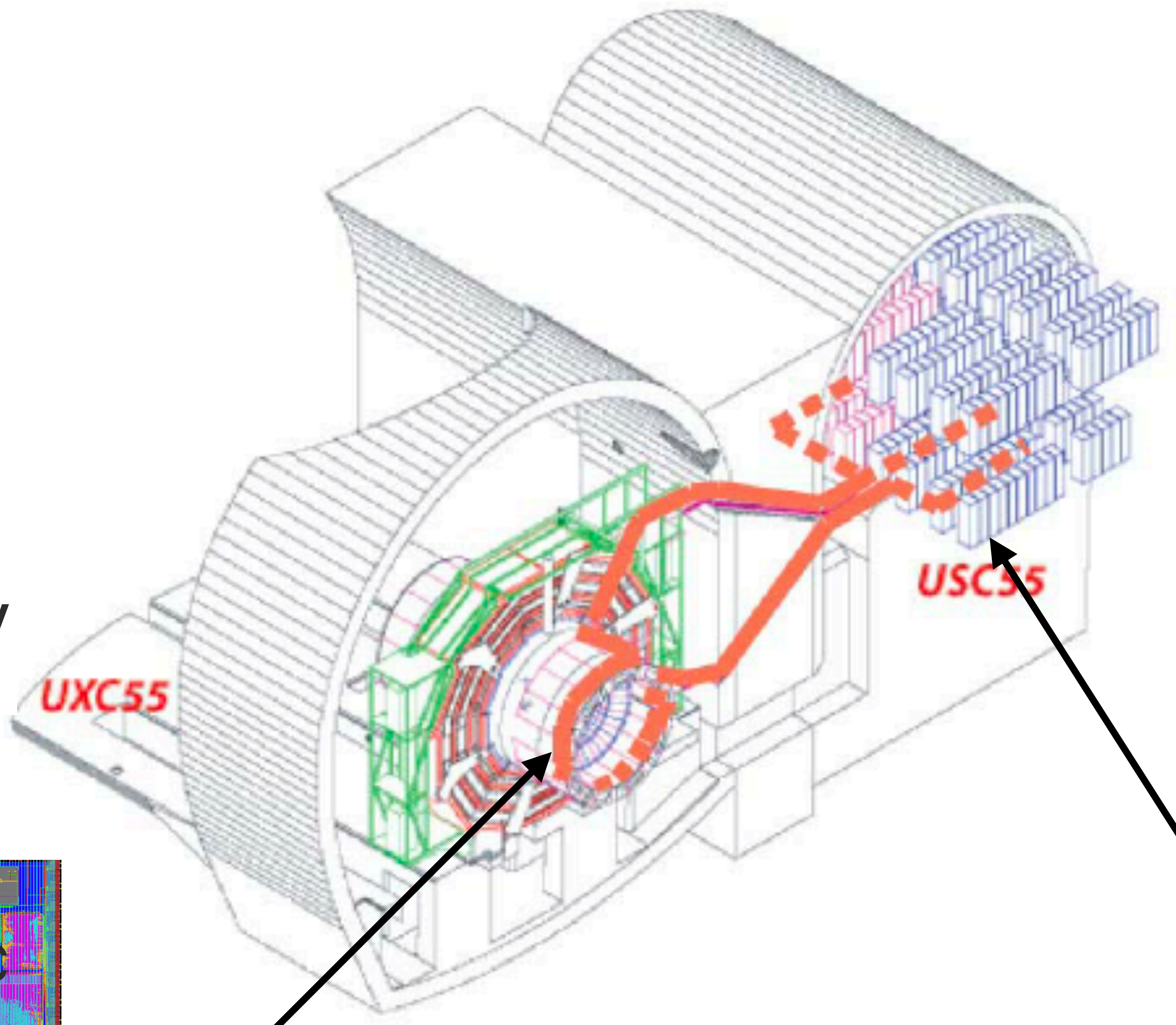


ECON-T, D. Noonan

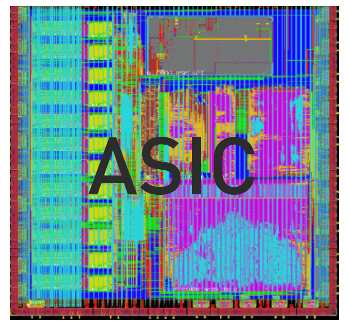
AEs for compression also at LHCb!

__ECON-T, D. Noonan__

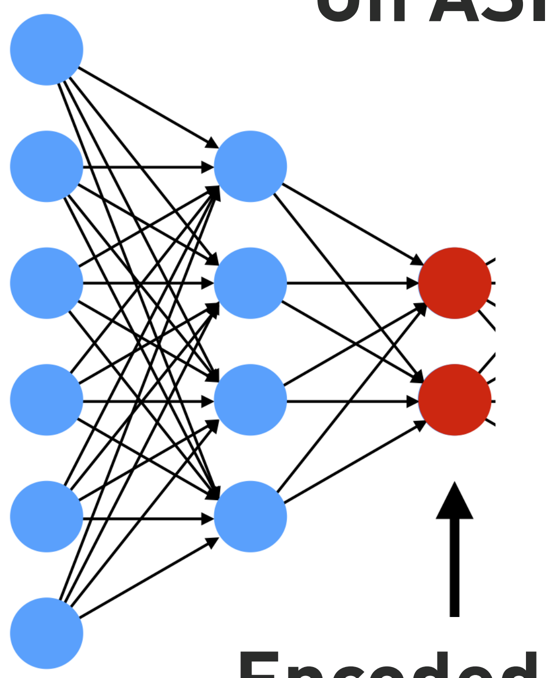
__AEs for compression also at LHCb!__



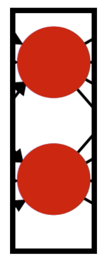
- **75-100 mW**
- **Triplicated w/b for radiation safety**
  Reprogrammable w/b over IC2!



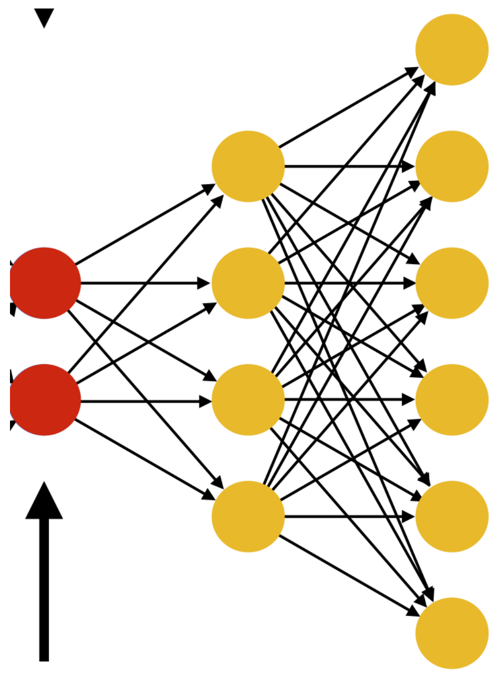
On ASIC



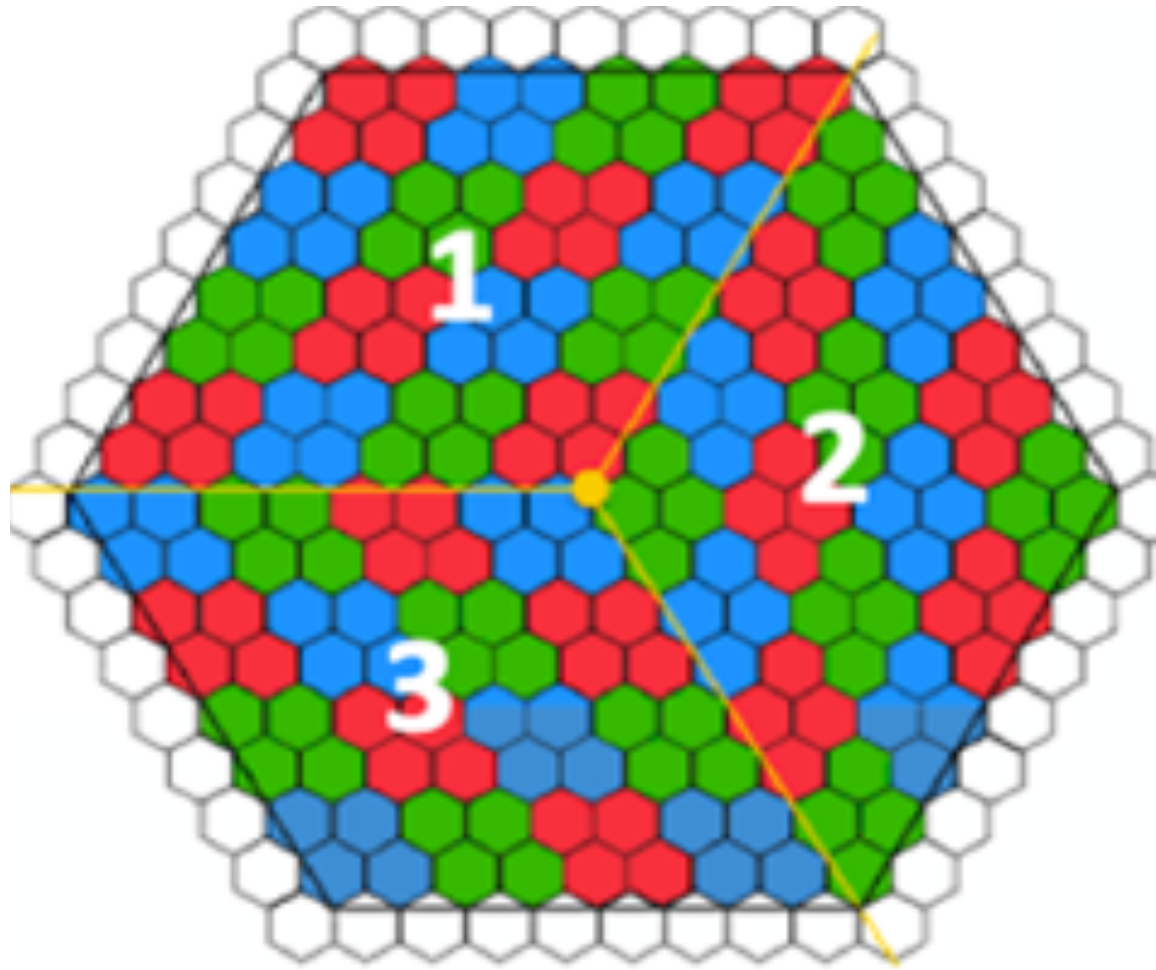
Encoded data



Transmit encoded data!

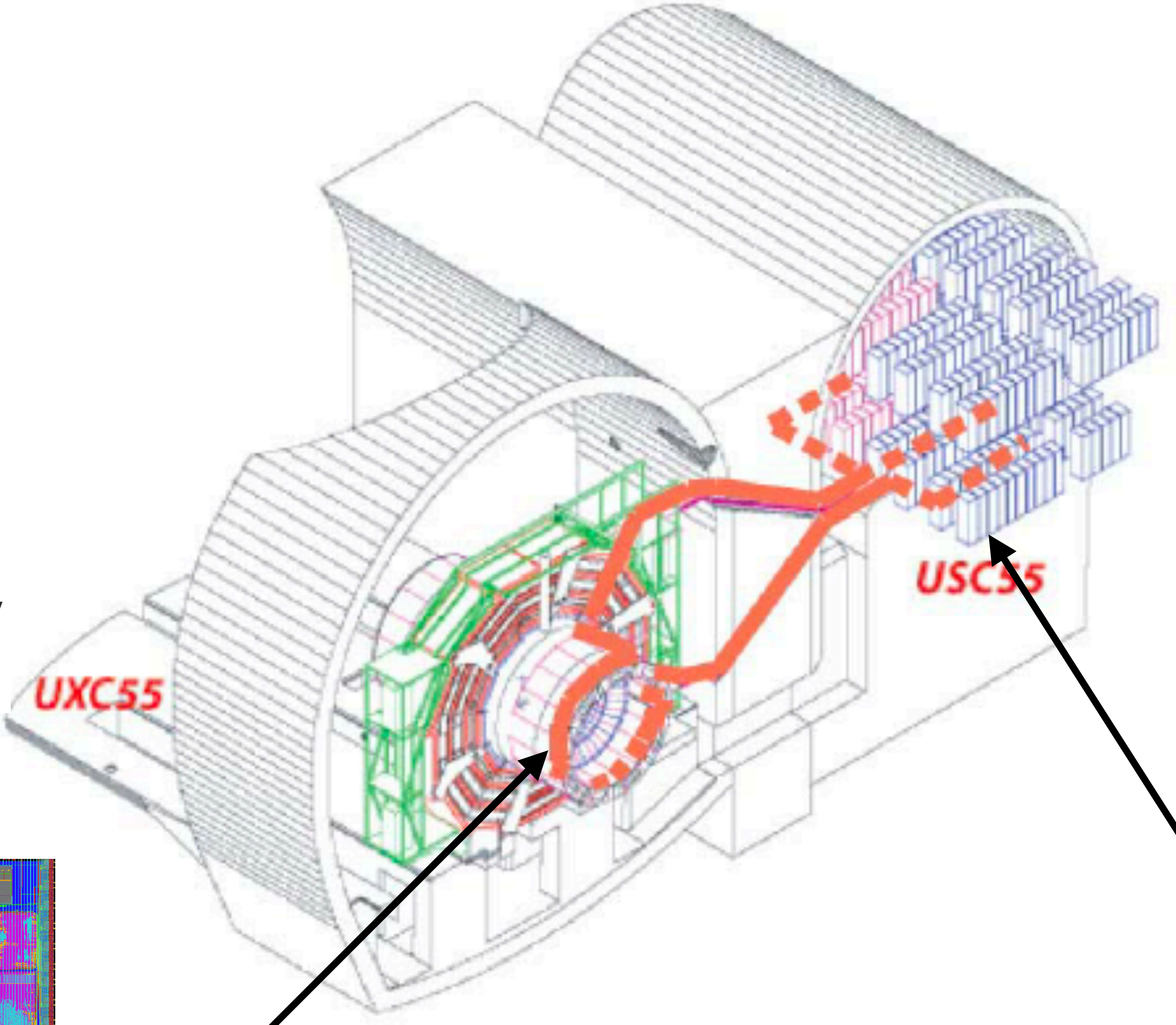


Encoded data

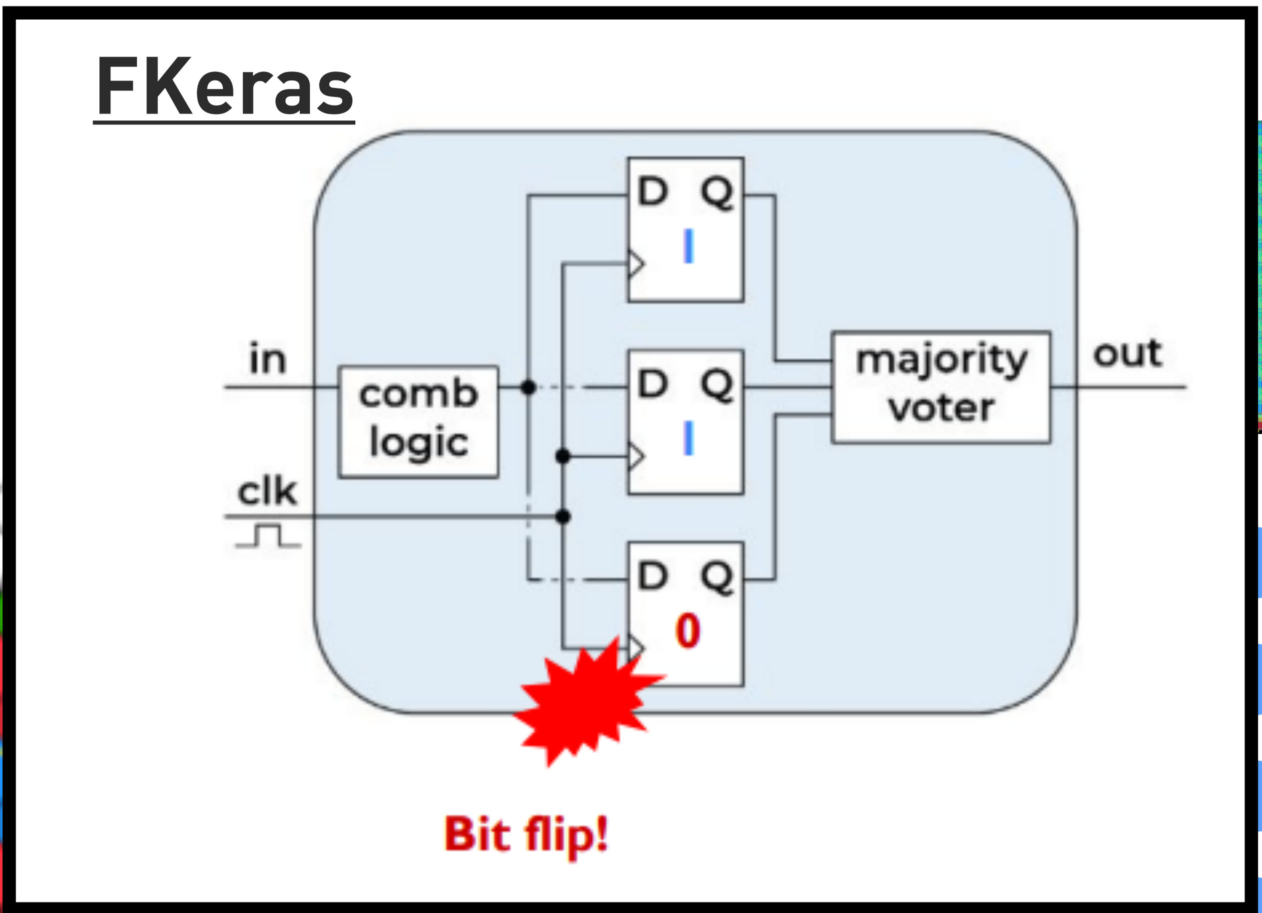


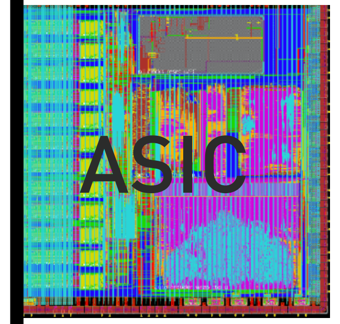
ECON-T, D. Noonan

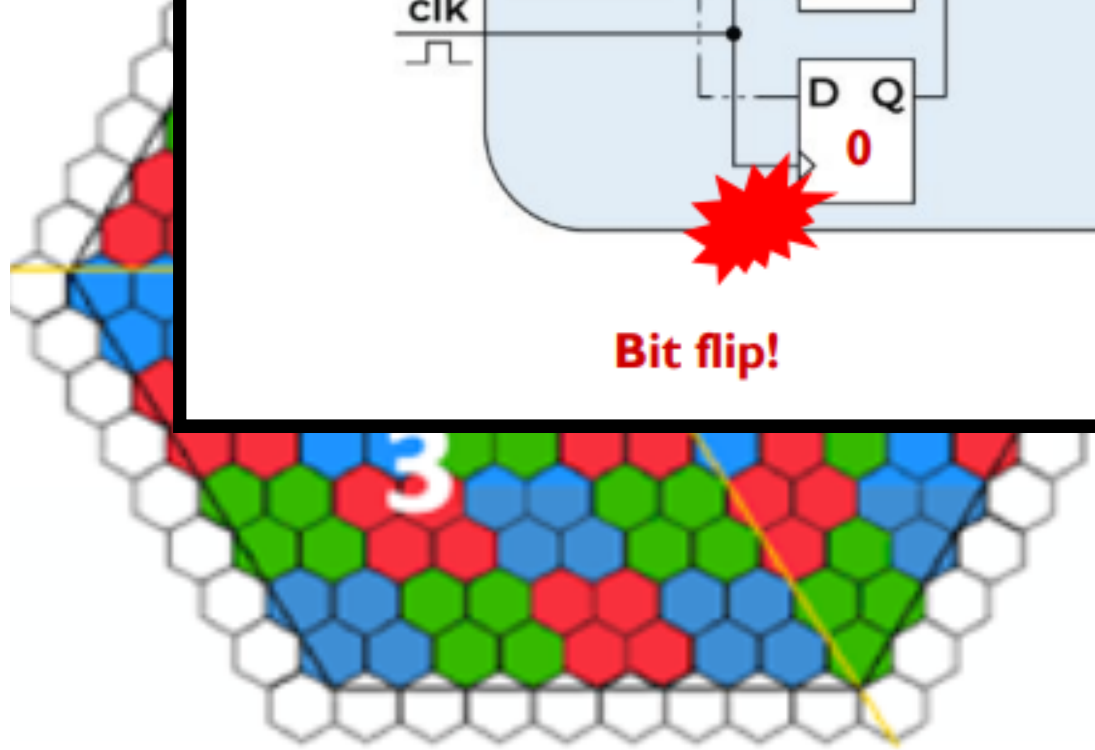
AEs for compression also at LHCb!

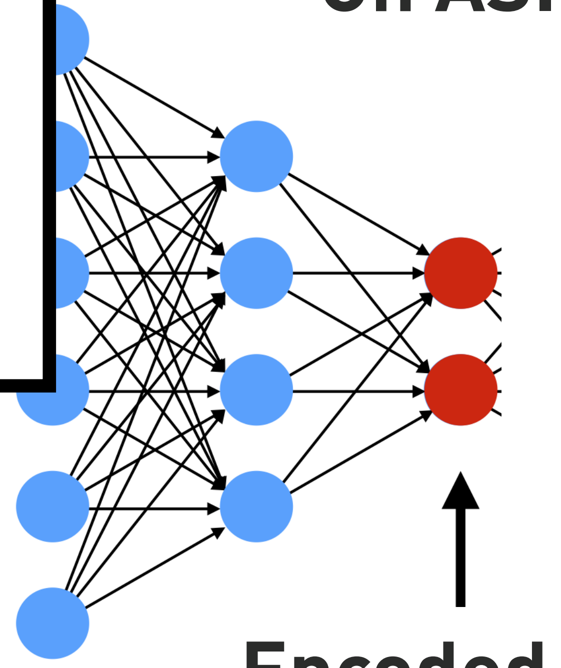


- 75-100 mW
- Triplicated w/b for radiation safety
  Reprogrammable w/b over IC2!



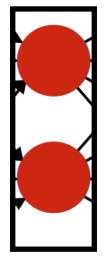




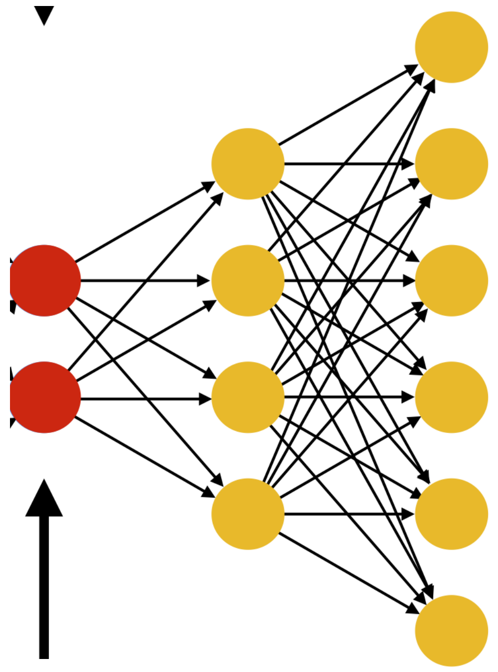


On ASIC

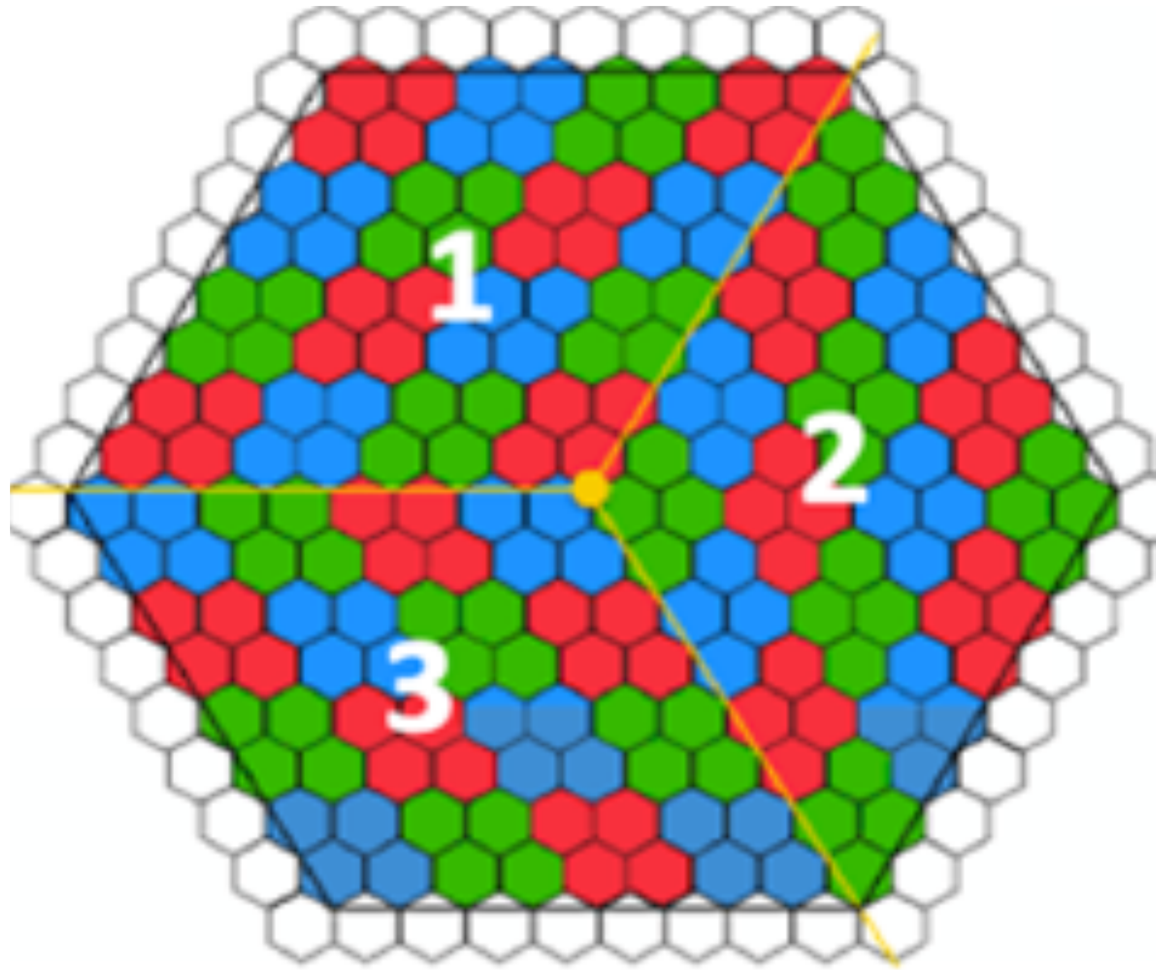
Encoded data



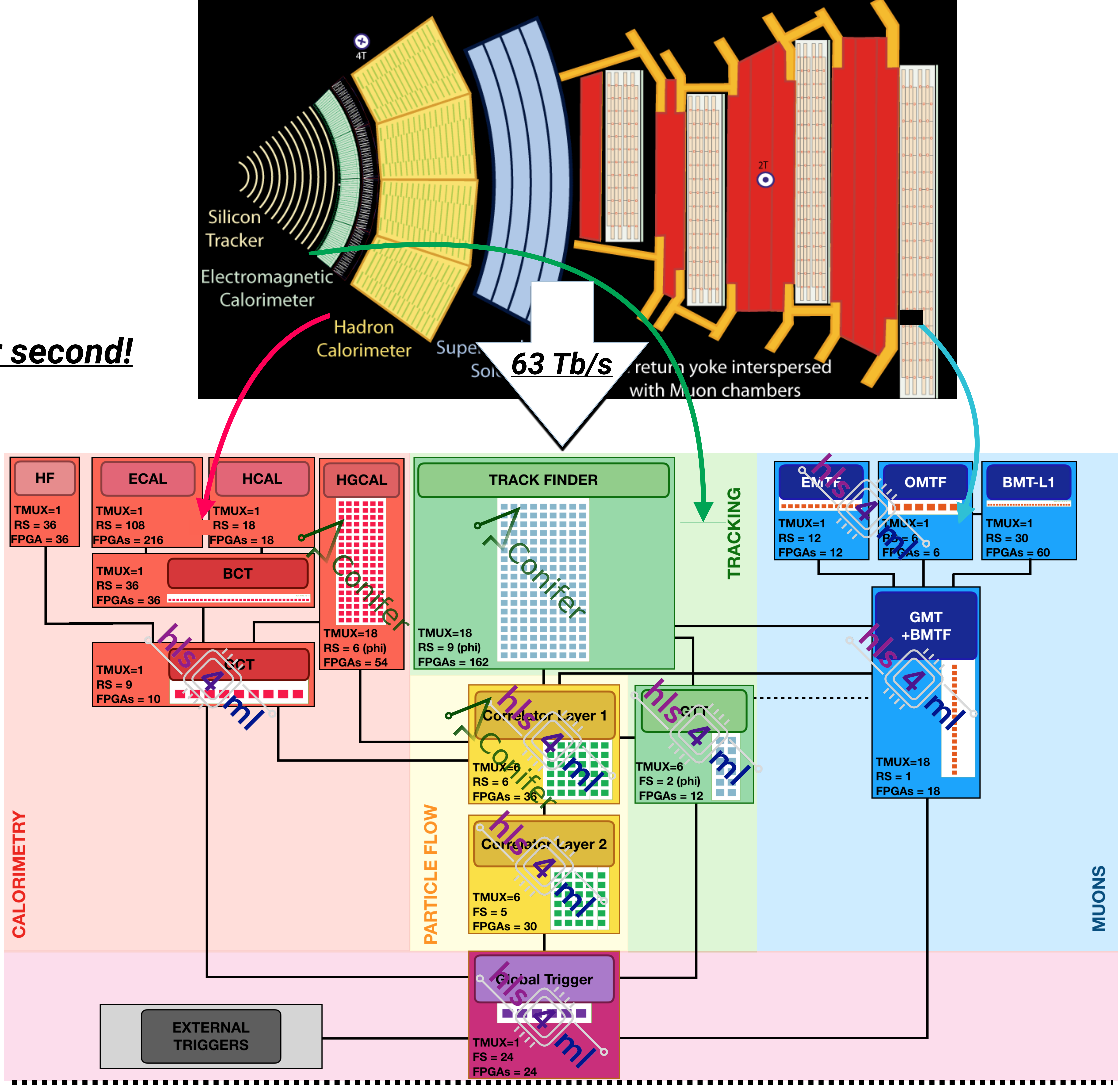
Transmit encoded data!



Encoded data

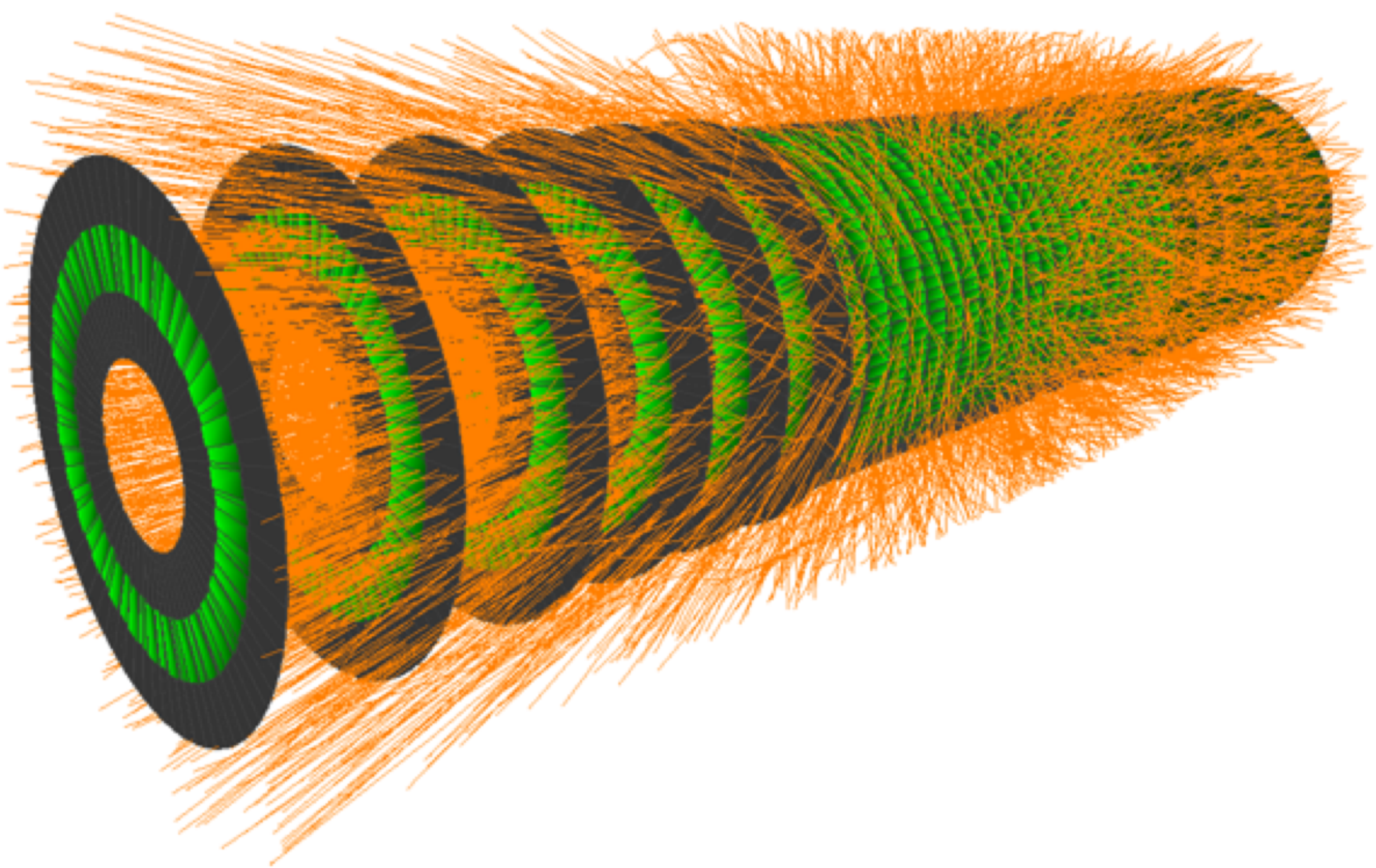


25 billion inferences per second!



DOI:10.3389/fdata.2022.828666

ML for tracking



In HL-LHC, will need to do track finding at L1

- O(1000) hits, O(100) tracks, 40 MHz rate, ~5 μs latency

Graph Neural Networks for fast charged particle tracking

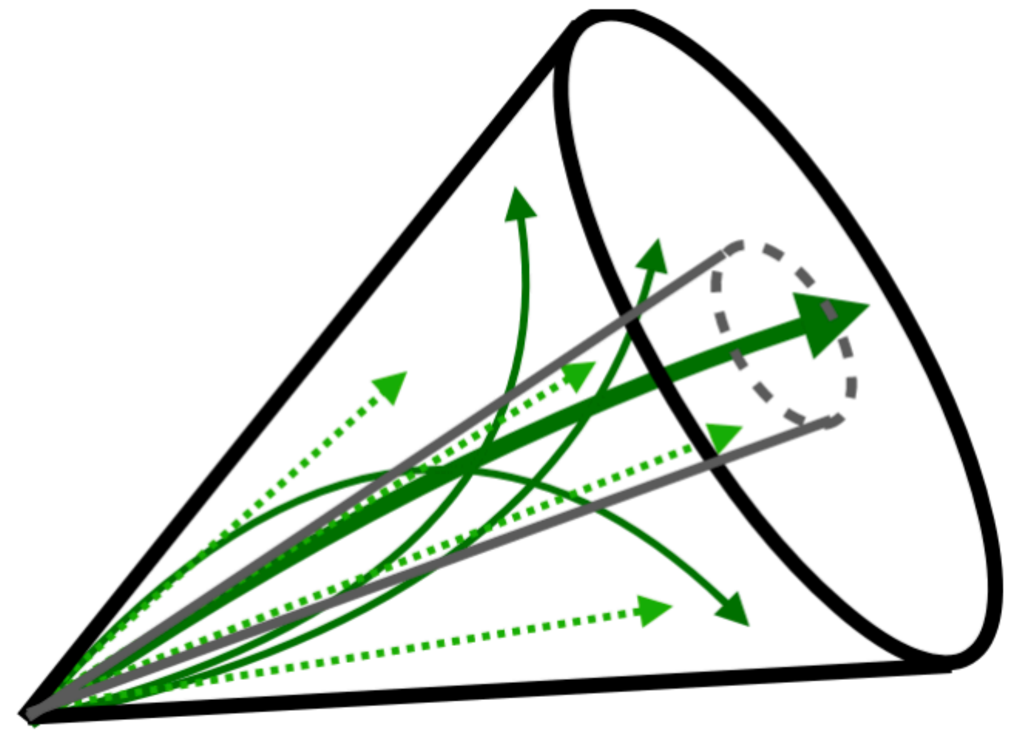
Throughput-optimized for L1 applications,
resource-optimised for co-processing

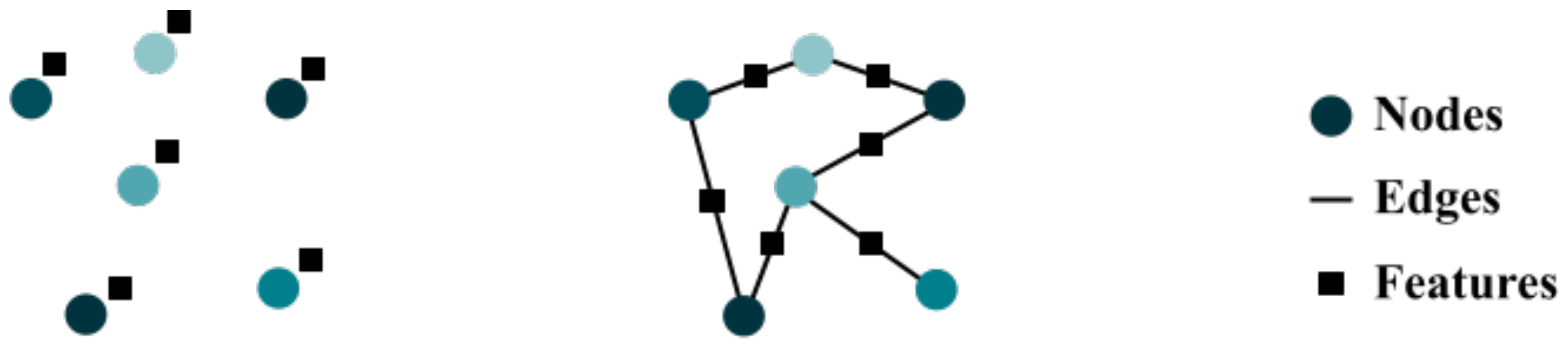
Design	(n_{nodes}, n_{edges})	RF	Precision	Latency [cycles]	II [cycles]	DSP [%]	LUT [%]	FF [%]	BRAM [%]
Throughput-opt.	(28, 56)	1	ap_fixed<14,7>	59 295 ns	1	99.9	66.0	11.7	0.7
Resource-opt.	(28, 56)	1	ap_fixed<14,7>	79 395 ns	28	56.6	17.6	3.9	13.1

cds.cern.ch/record/2814728/

Fast jet tagging

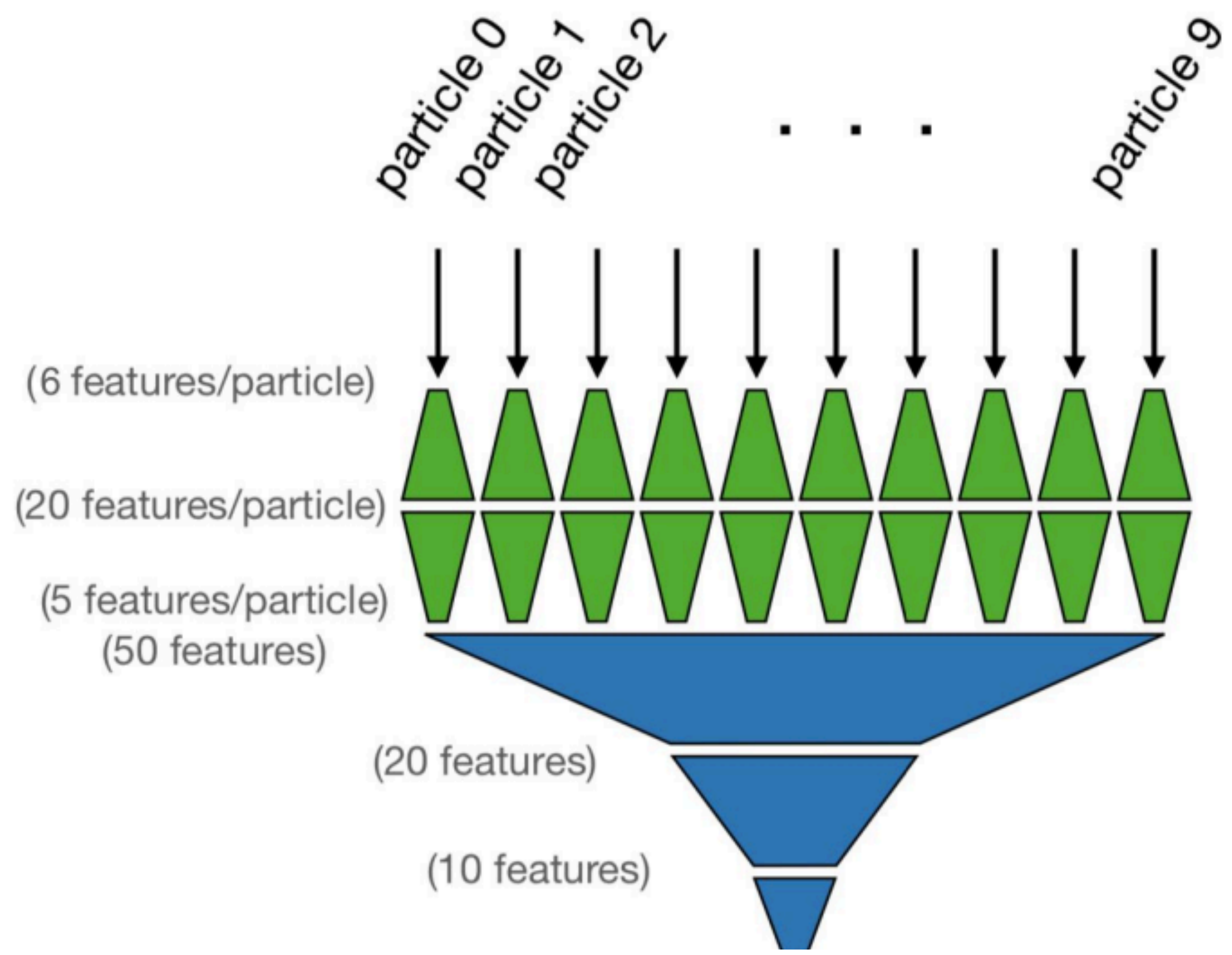
... best representation?

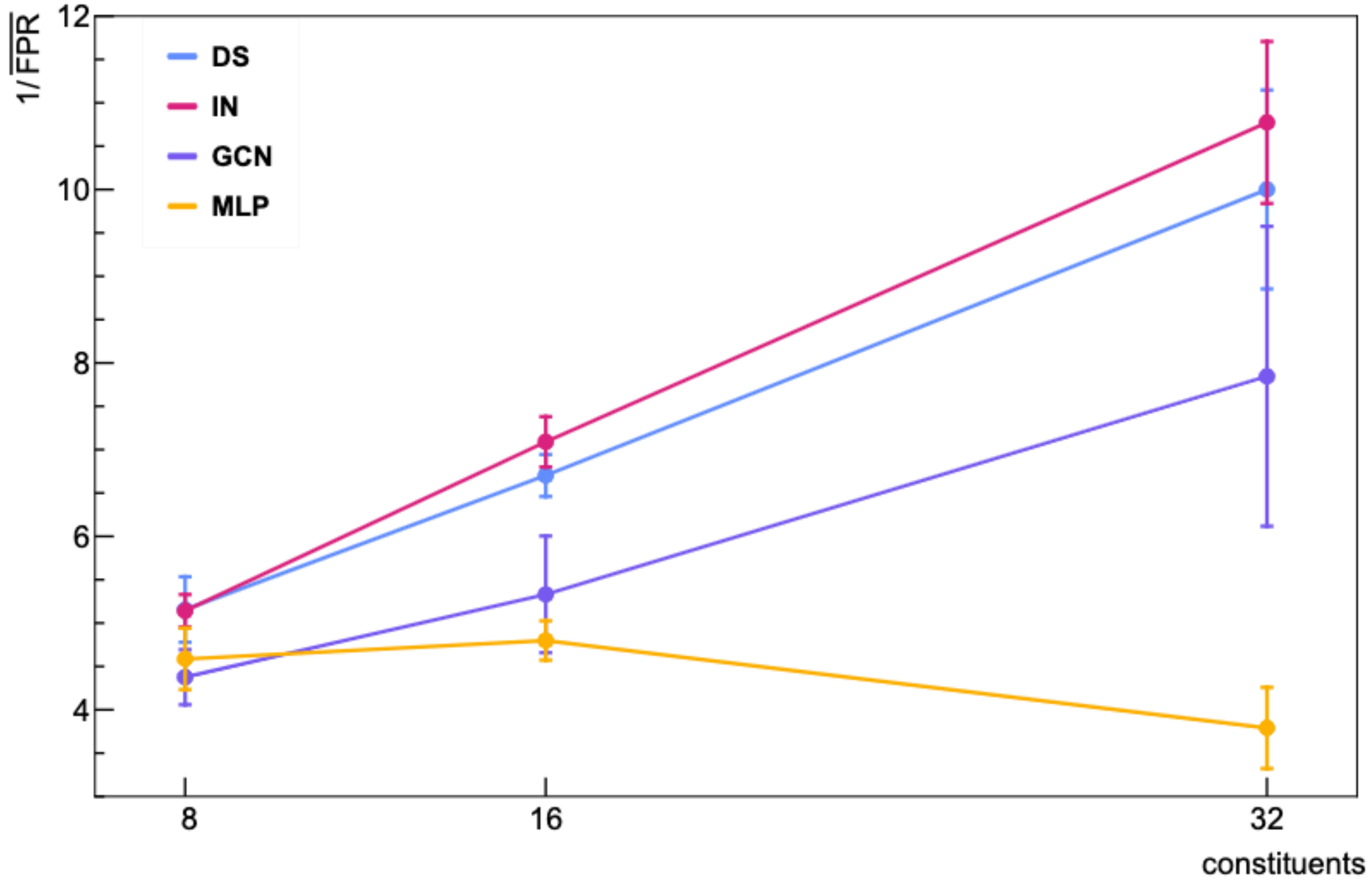




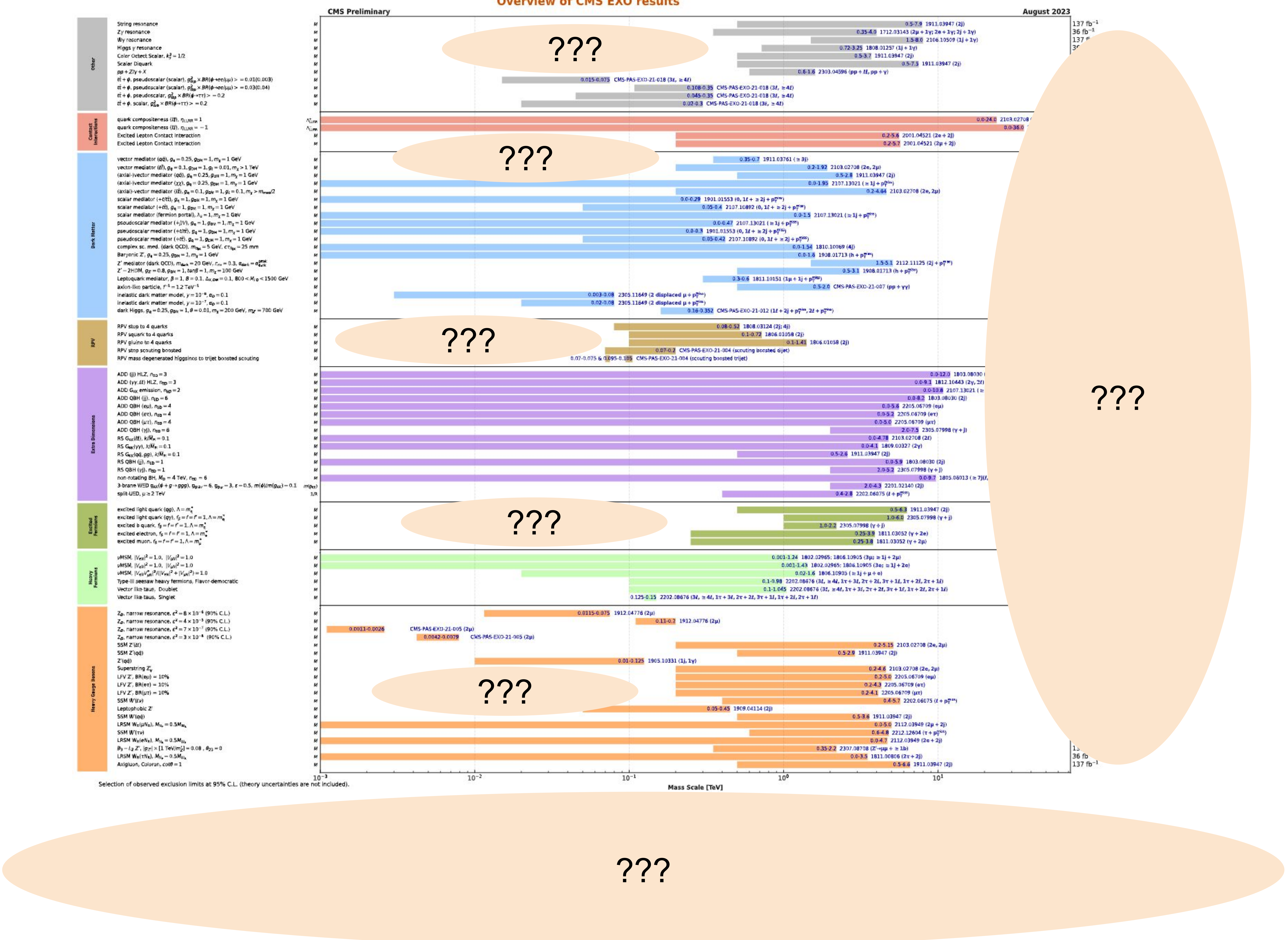
Sets: Information is only assigned to individual nodes.

Graphs: Information is assigned to edges, i.e., pairs of nodes.

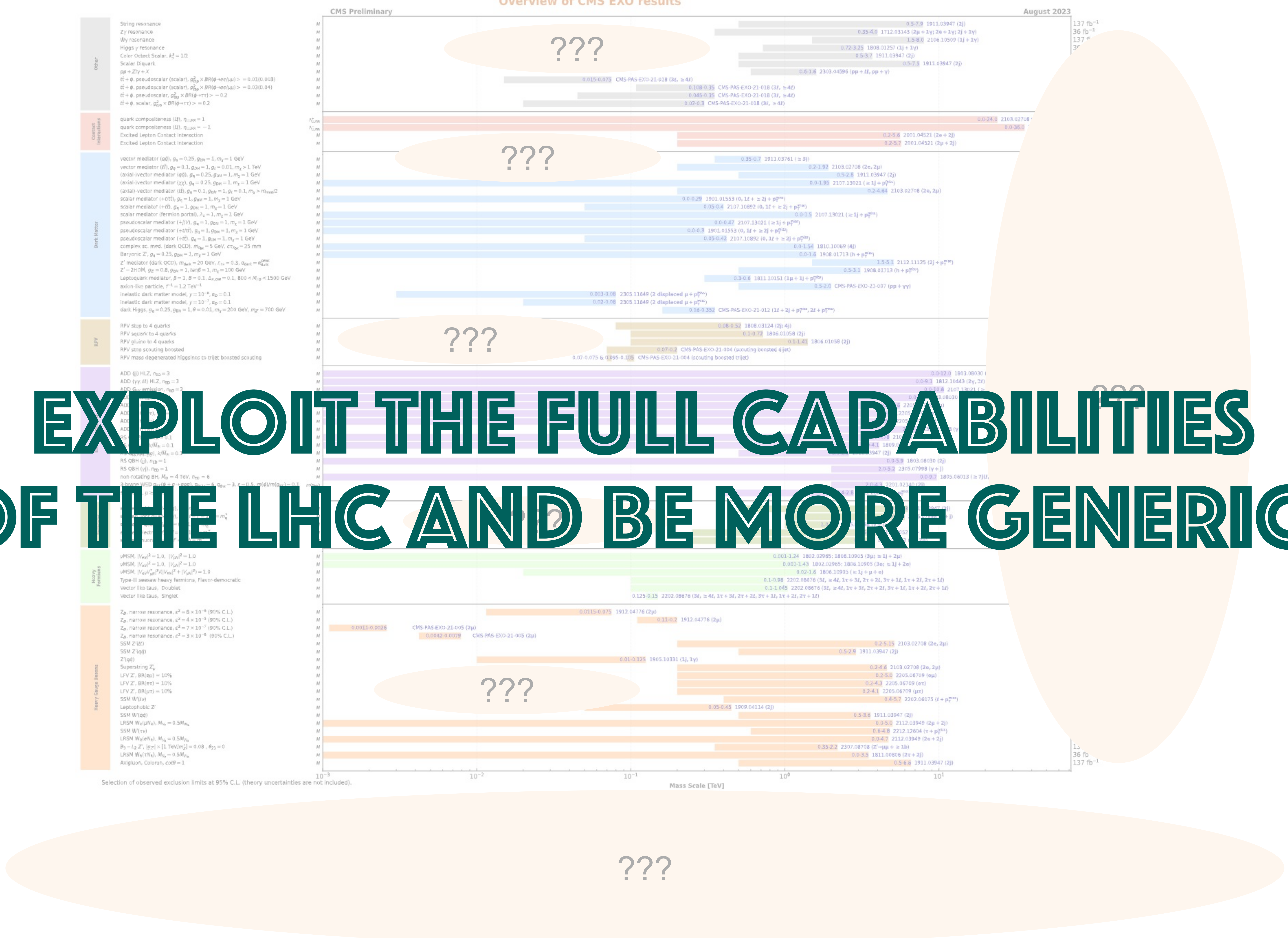




(Can also do 90 ns transformers for jet tagging!)

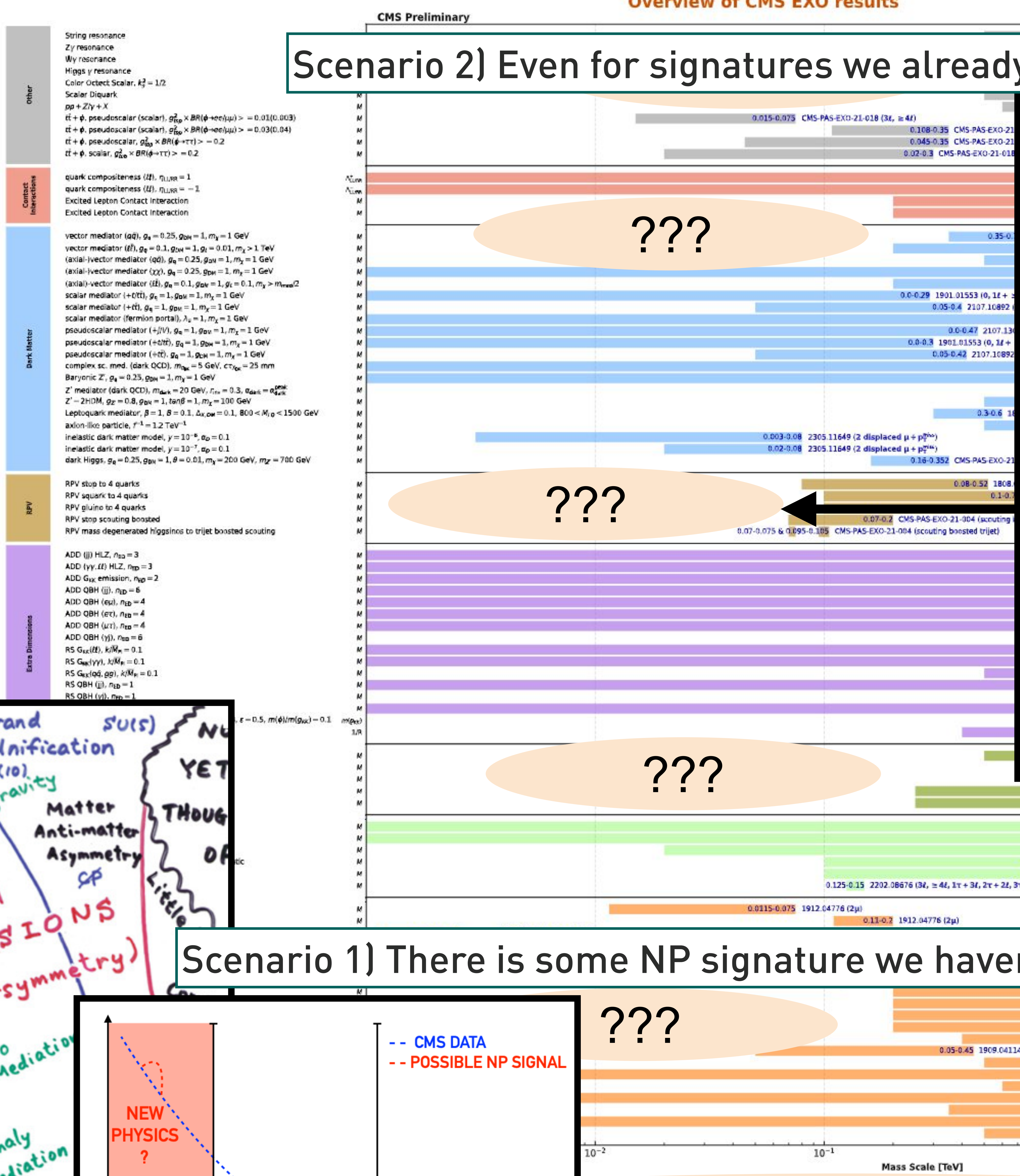

Overview of CMS EXO results
CMS Preliminary
August 2023
Selection of observed exclusion limits at 95% C.L. (theory uncertainties are not included).
Mass Scale [TeV]
???
???
???
???
???
???
???

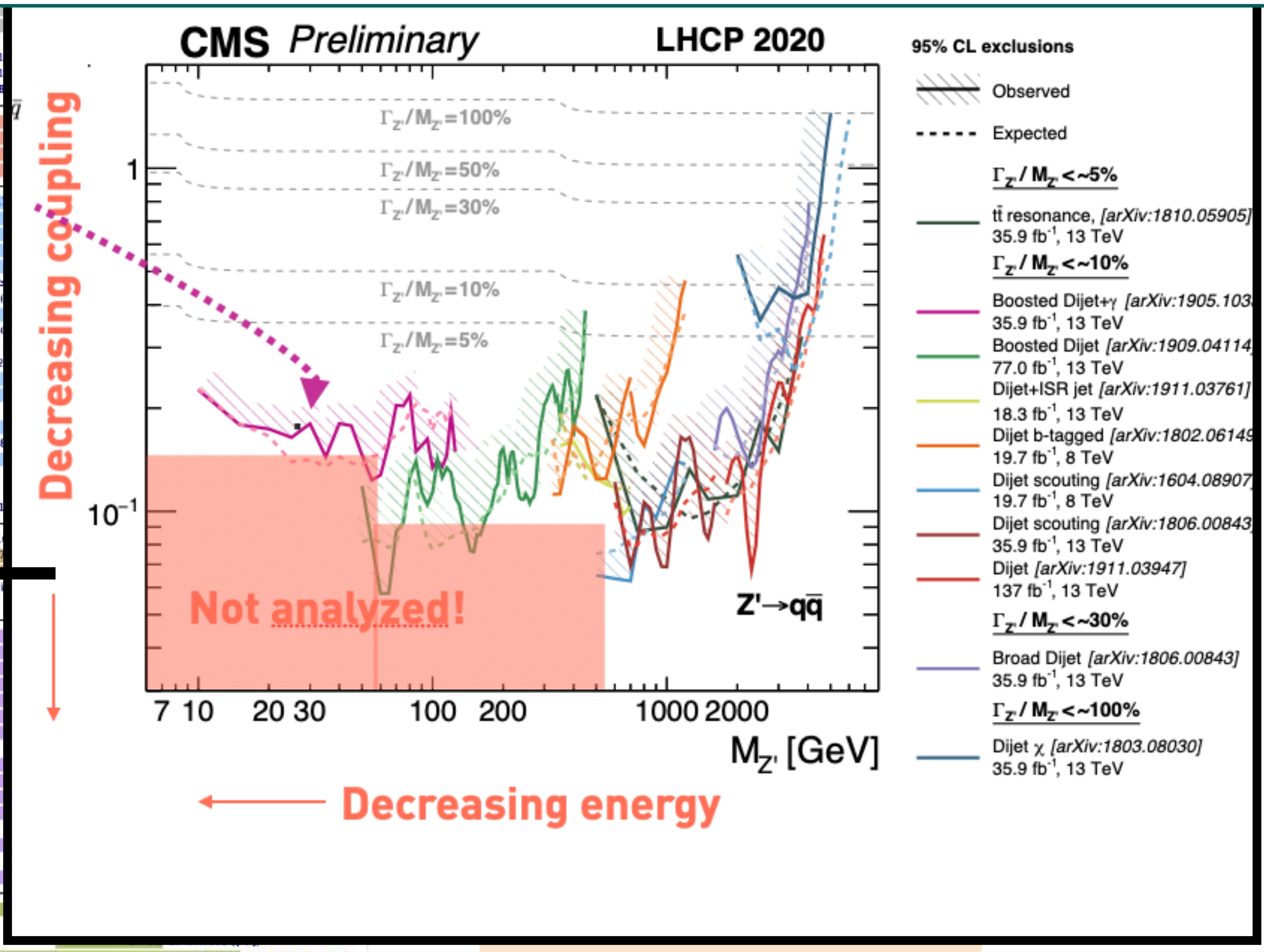
From A. Rizzi



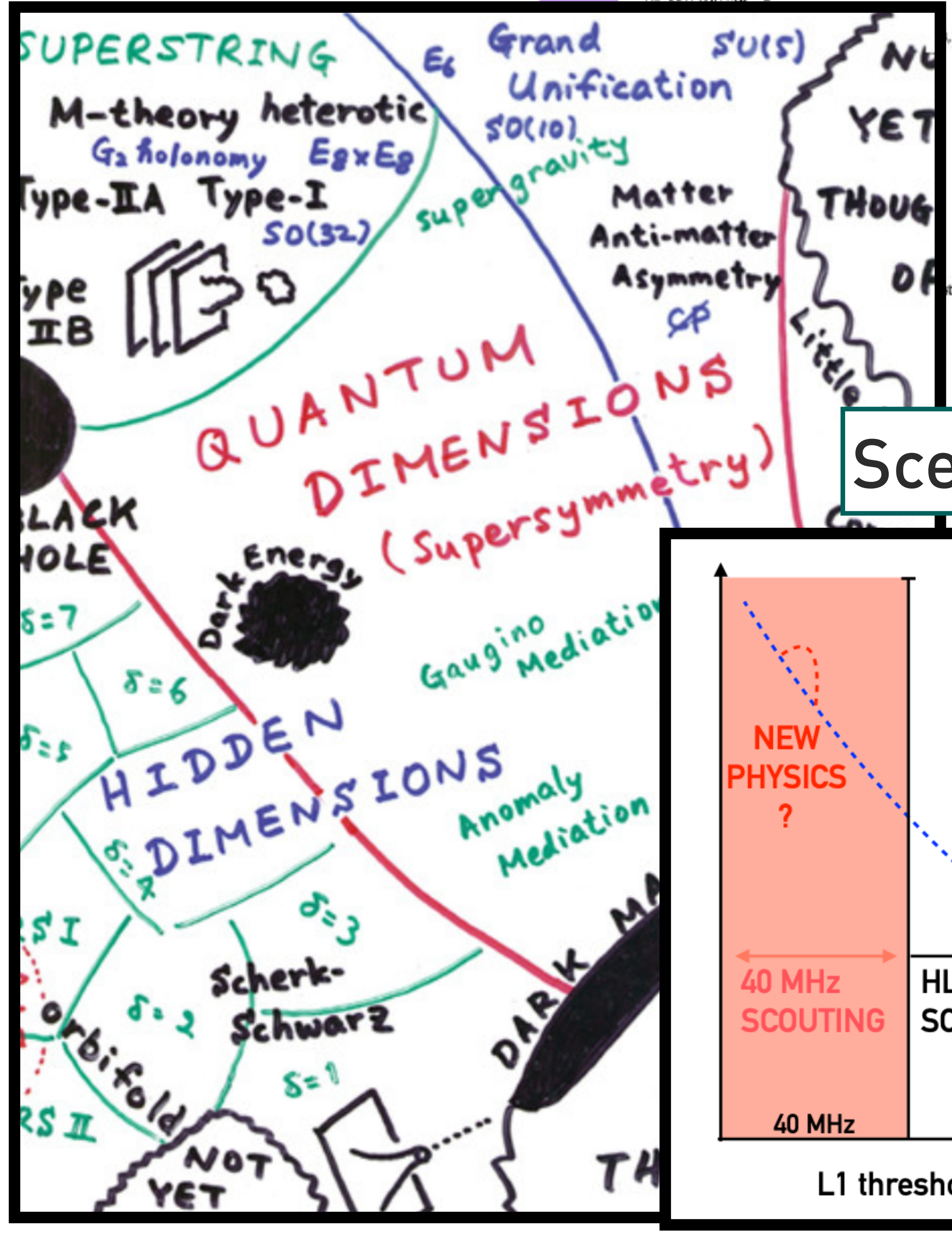
EXPLOIT THE FULL CAPABILITIES OF THE LHC AND BE MORE GENERIC!

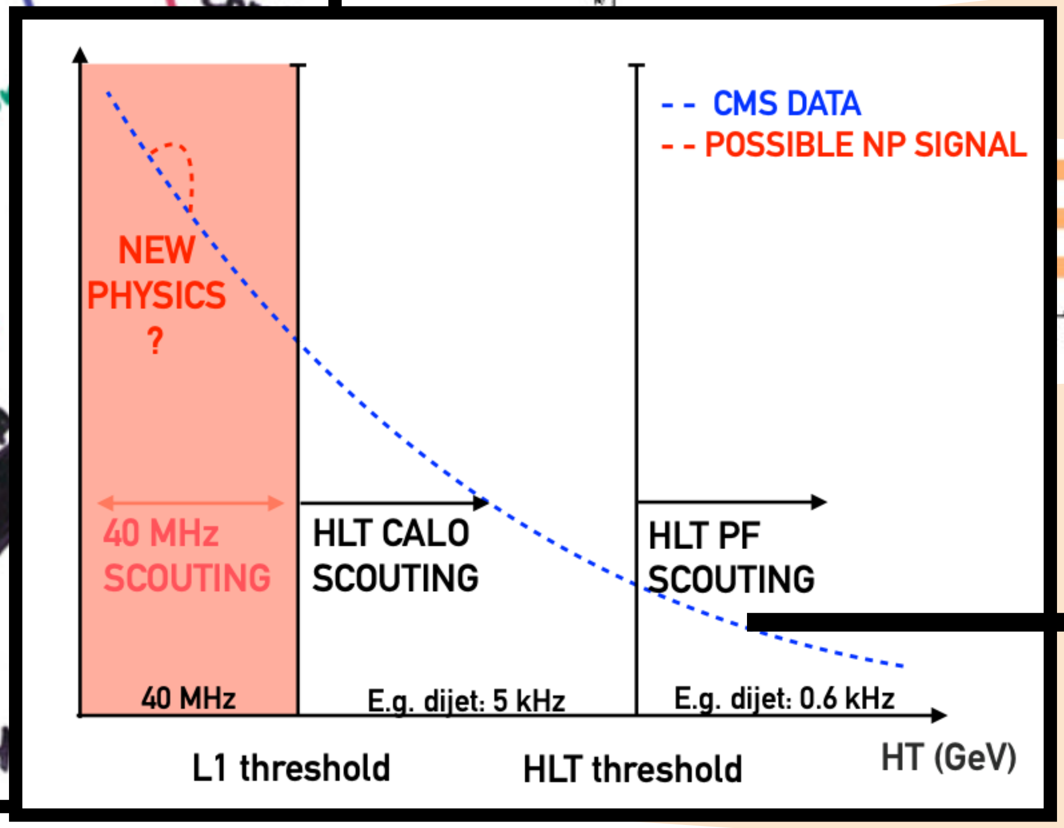
Scenario 2) Even for signatures we already look for, some regions out of reach due to L1 trigger





Scenario 1) There is some NP signature we haven't though of and we do not trigger on





???

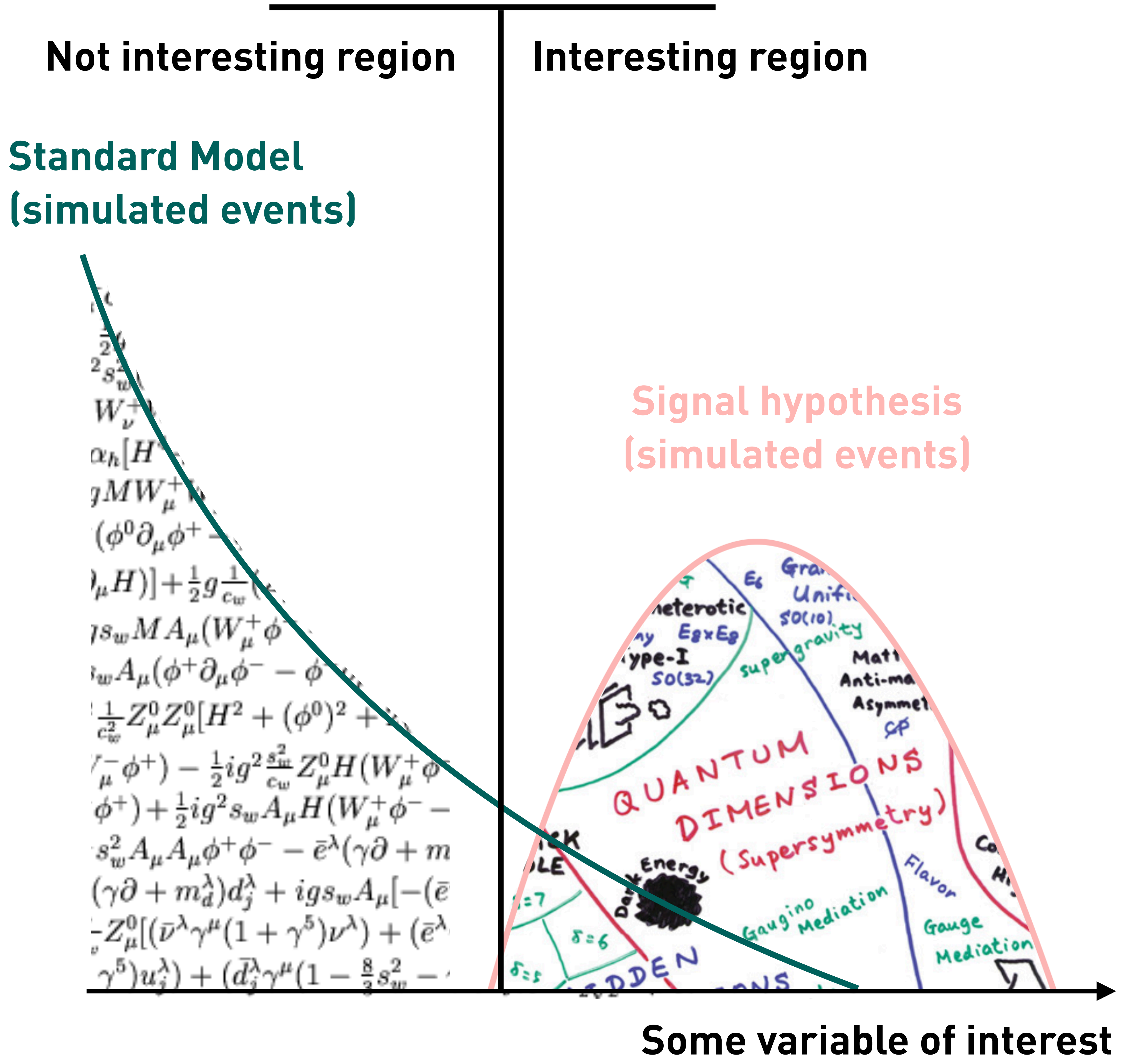
Bias in particle physics

Searches at LHC always start by

- assuming **Standard Model** and some **signal hypothesis**

This is fine when we know what "signal" is (like Higgs)

- **Tailor search** to a given theory
- Powerful, but **limited to model of choice**



Bias in particle physics

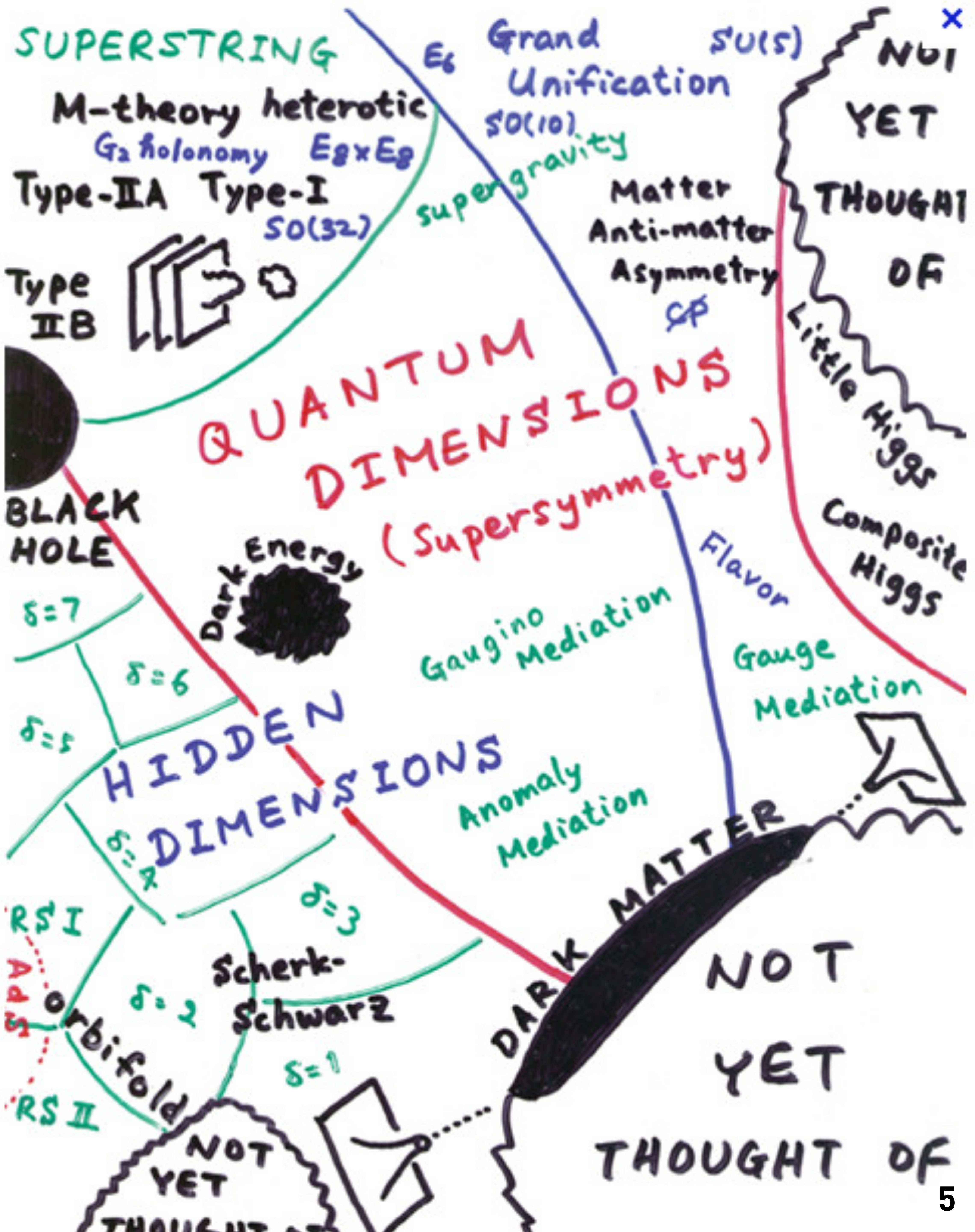
Searches at LHC always start by

- assuming **Standard Model** and some **signal hypothesis**

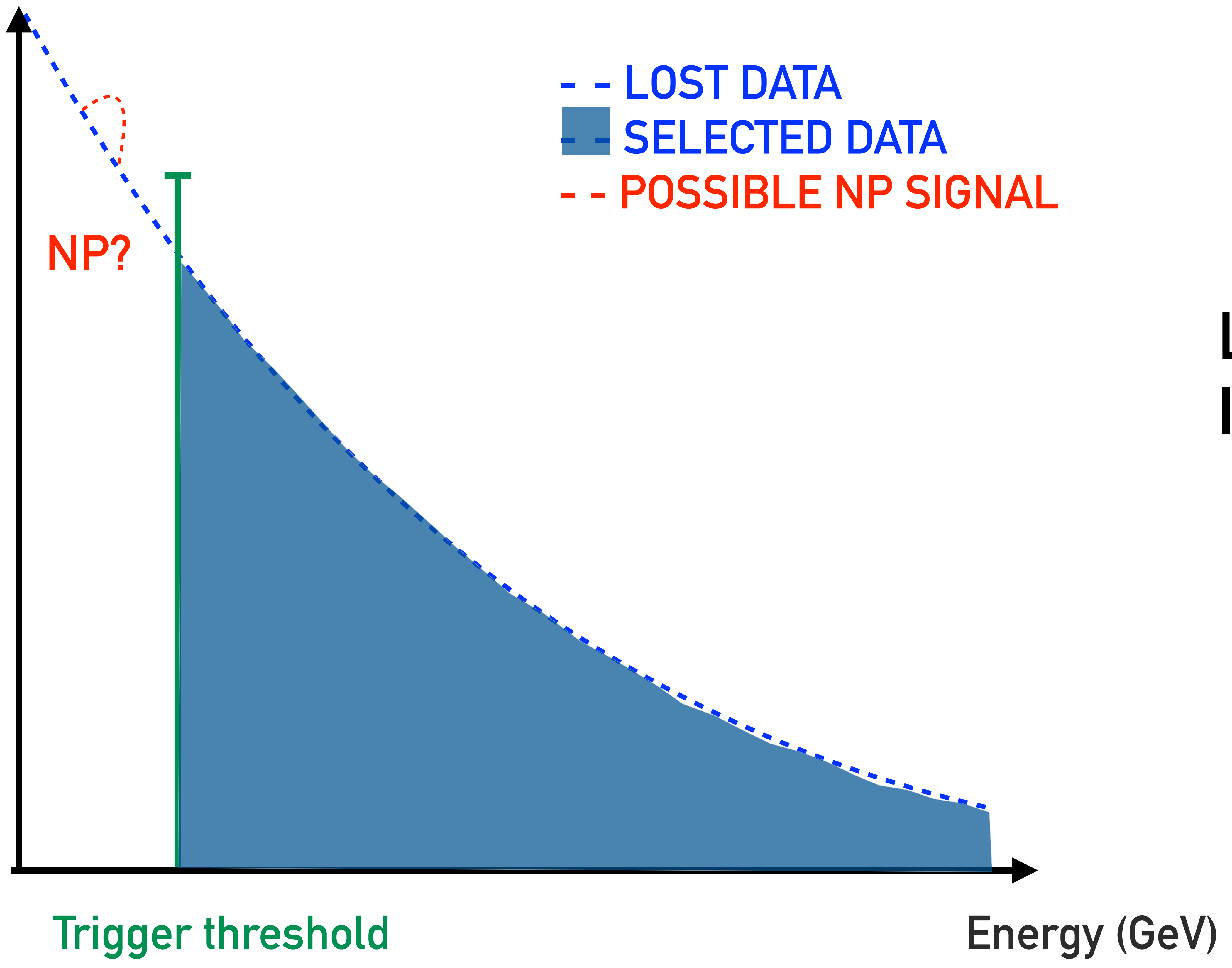
This is fine when we know what "signal" is (like Higgs)

- **Tailor search** to a given theory
- Powerful, but **limited to model of choice**

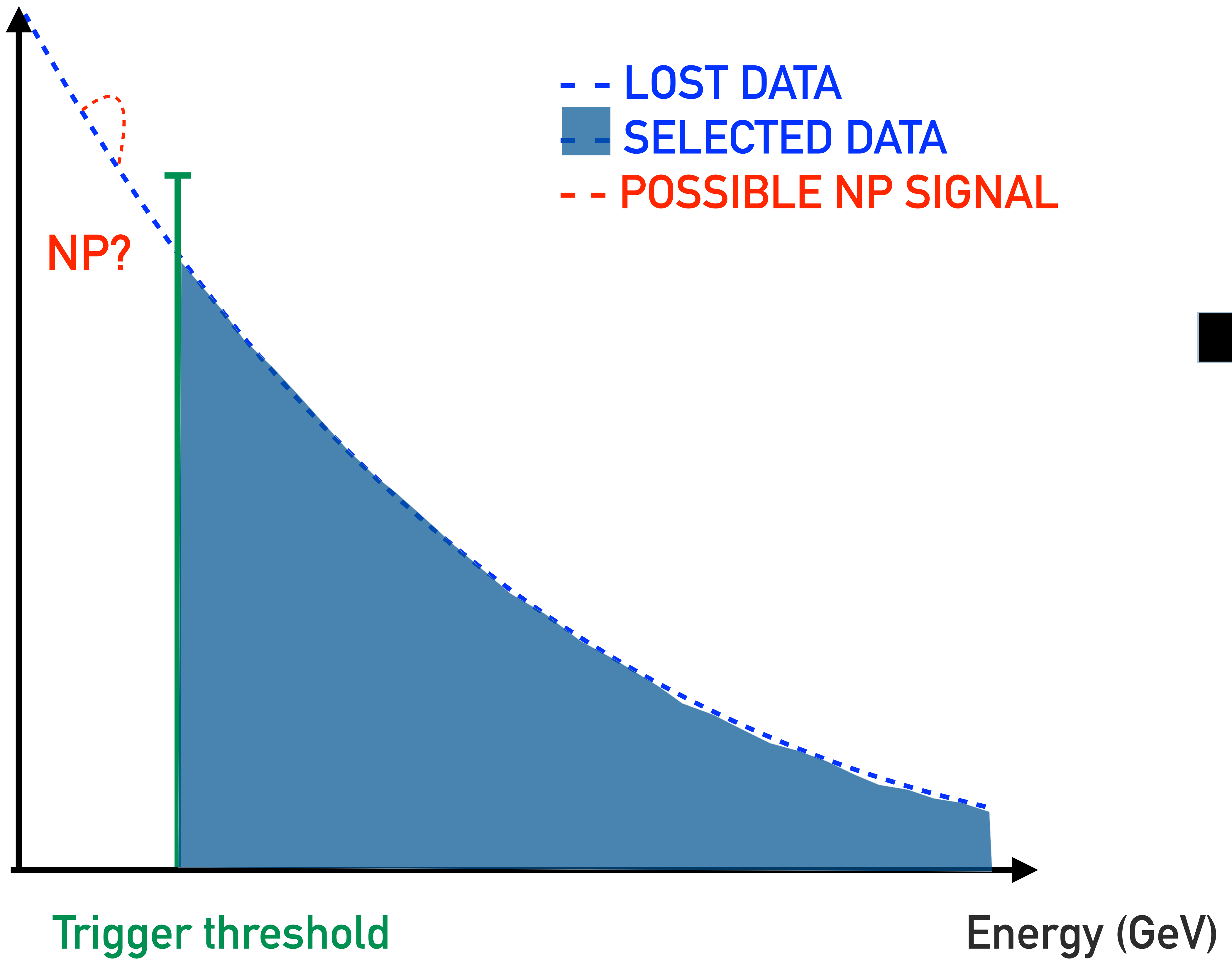
- How do we know we are looking for the right thing in the enormous New Physics model landscape?



Limitations of current trigger

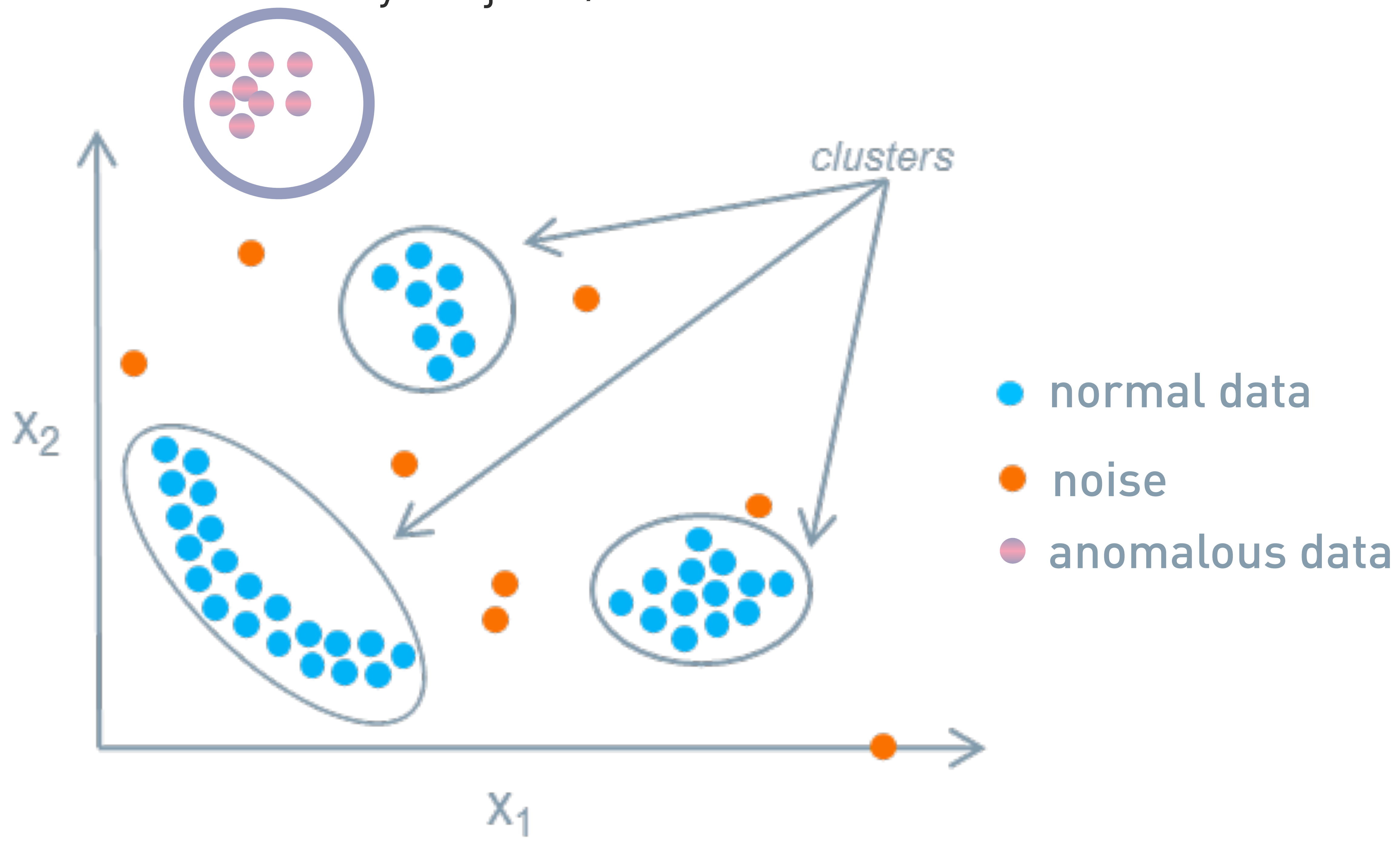


Level-1 rejects >99% of events!
Is there a smarter way to select?



Look at **data** rather than defining signal hypothesis a priori

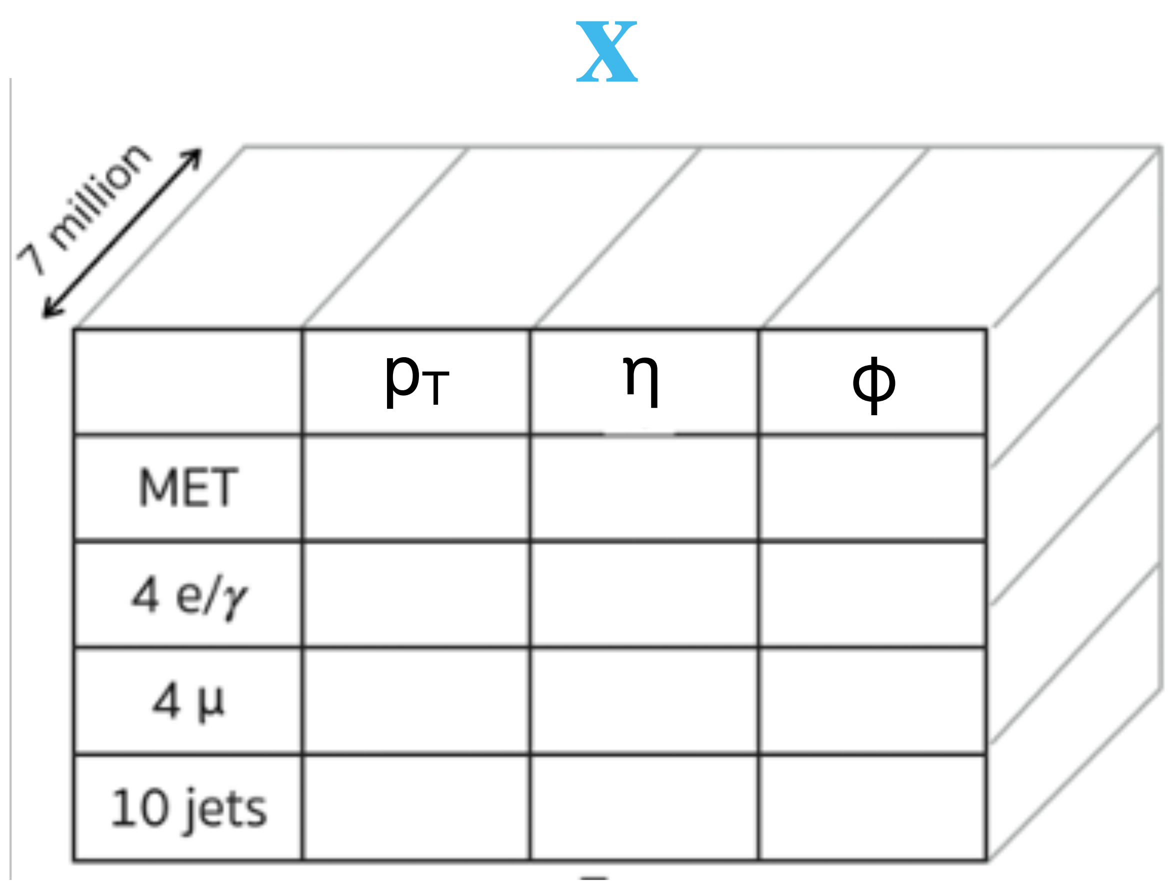
- Can we “classify” objects/events?

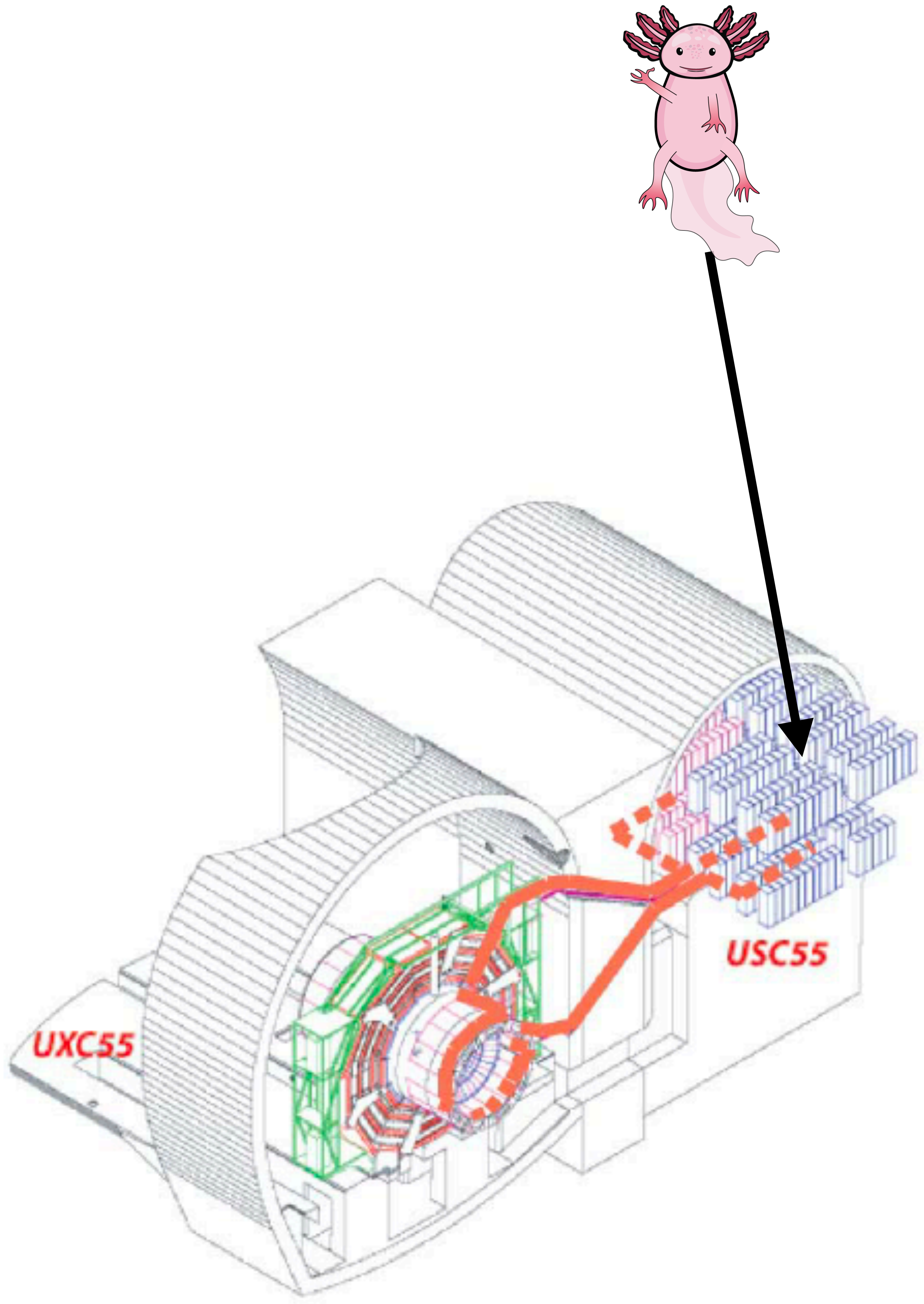


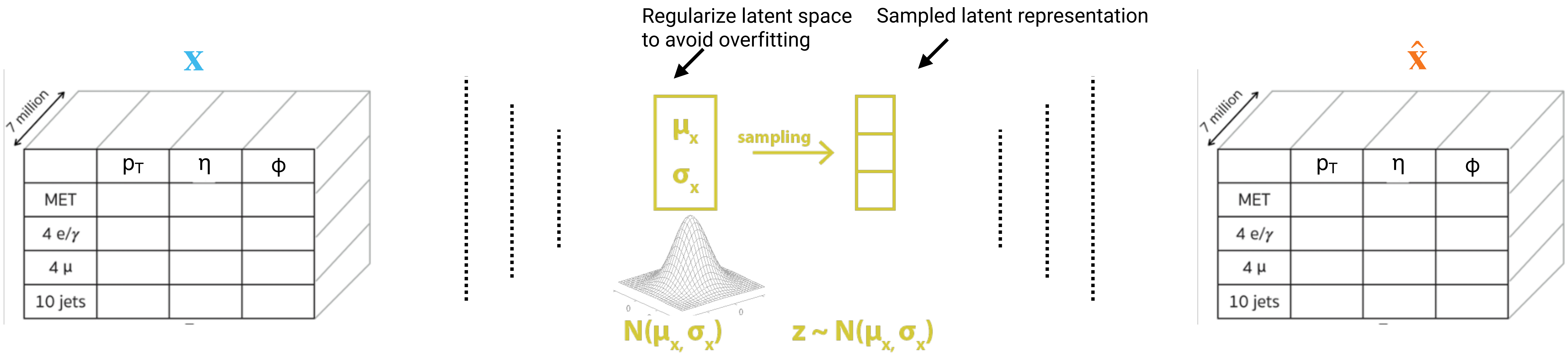
AXOL1TL

Anomaly Detection in the hardware trigger

Input from available trigger objects quantities:







$$\text{loss} = \|x - \hat{x}\|^2 + \mathrm{KL}[\,N(\mu_x, \sigma_x), N(0, I)\,]$$

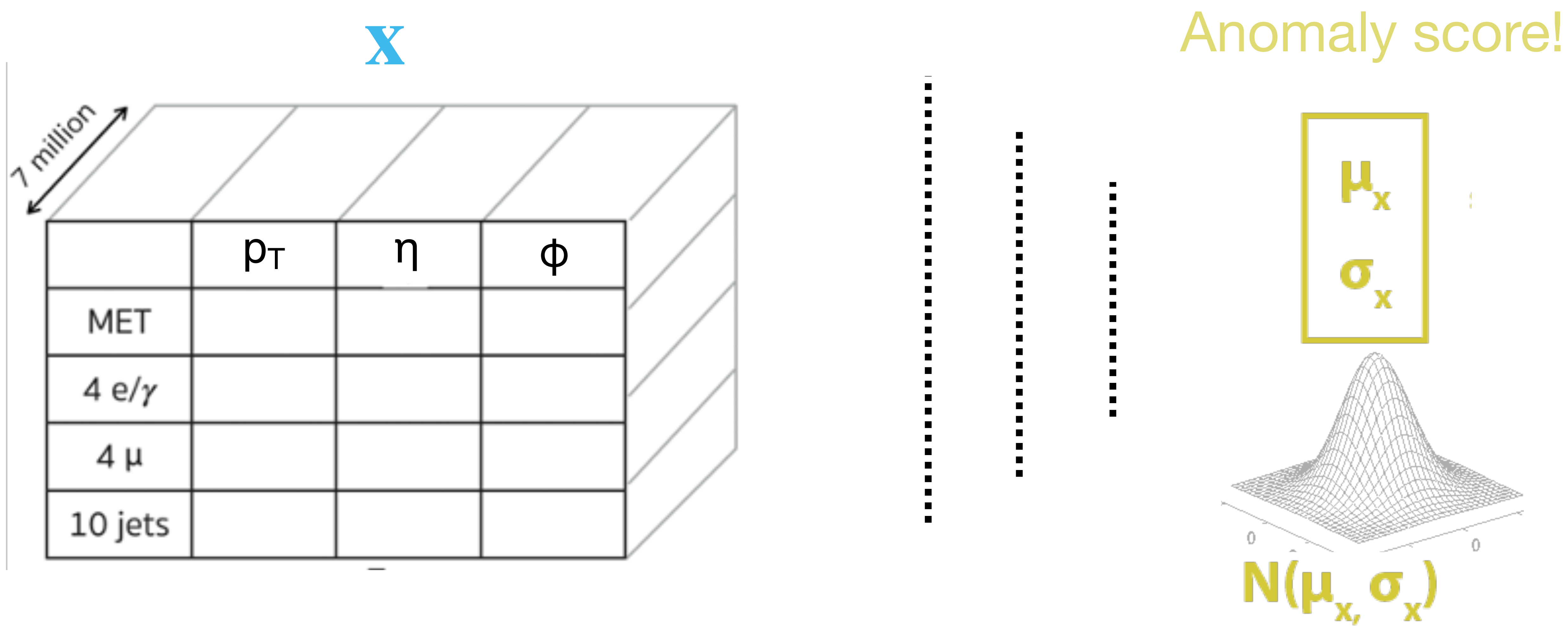
AXOL1TL

$$\text{loss} = ||\, x - \hat{x} \,||^2 + \text{KL}[\, N(\mu_x, \sigma_x), N(0, I) \,]$$

AXOL1TL

$$\text{loss} = \| x - \hat{x} \|^2 + \mathrm{KL}[\, N(\mu_x, \sigma_x), N(0, I)\,]$$

AXOL1TL



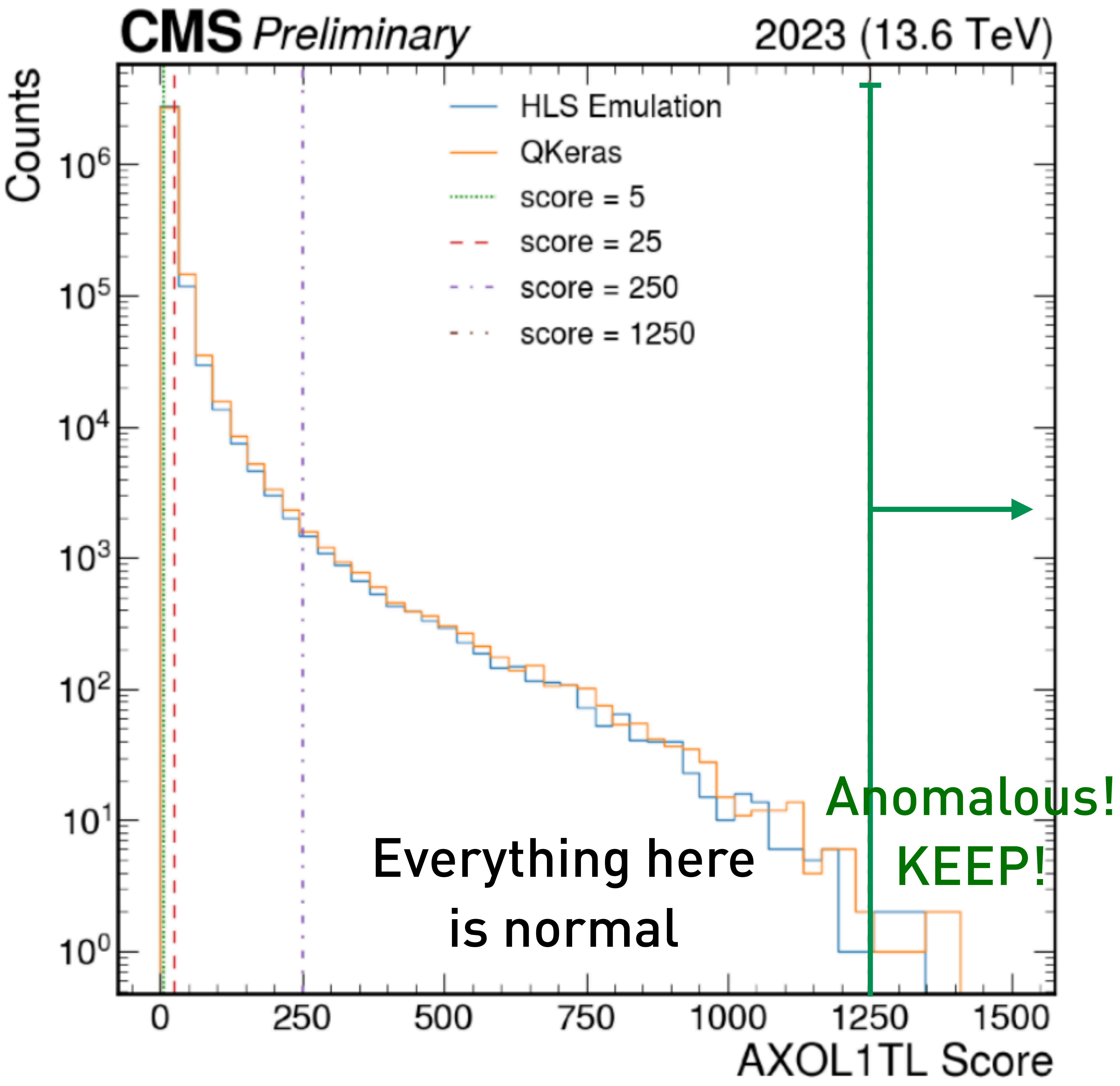
$$\text{loss} = \|x - \hat{x}\|^2 + \boxed{\mathrm{KL}[\, N(\mu_x, \sigma_x), N(0, I)\,]}$$

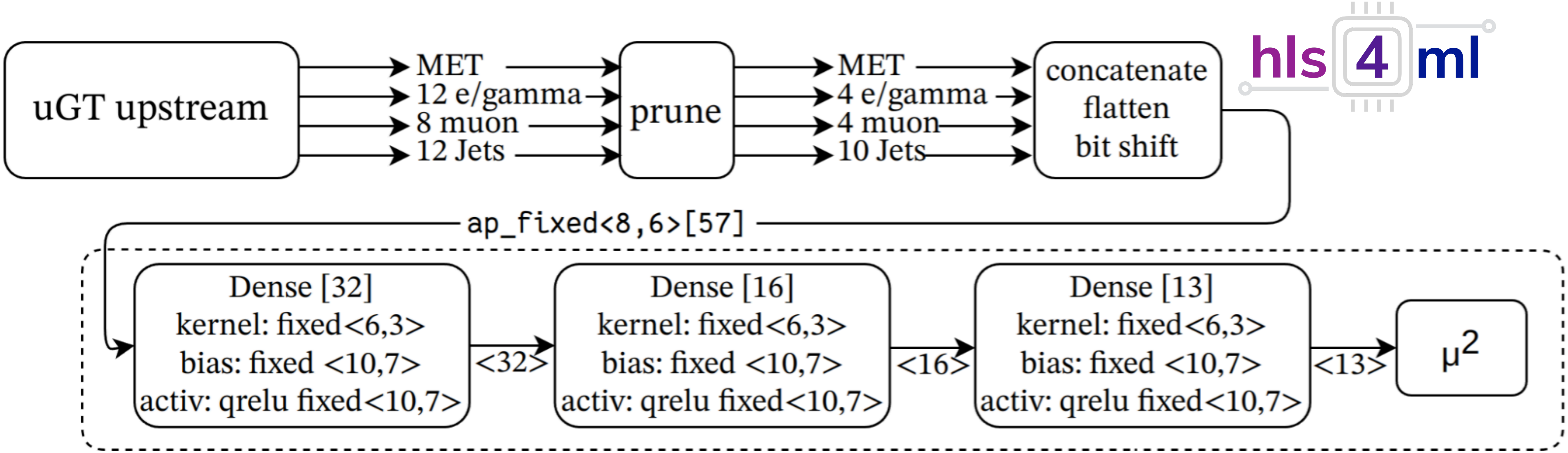
Anomaly score

After training, network is evaluated on orthogonal test set (offline)

- Find cut on anomaly score resulting in pre-defined background rate

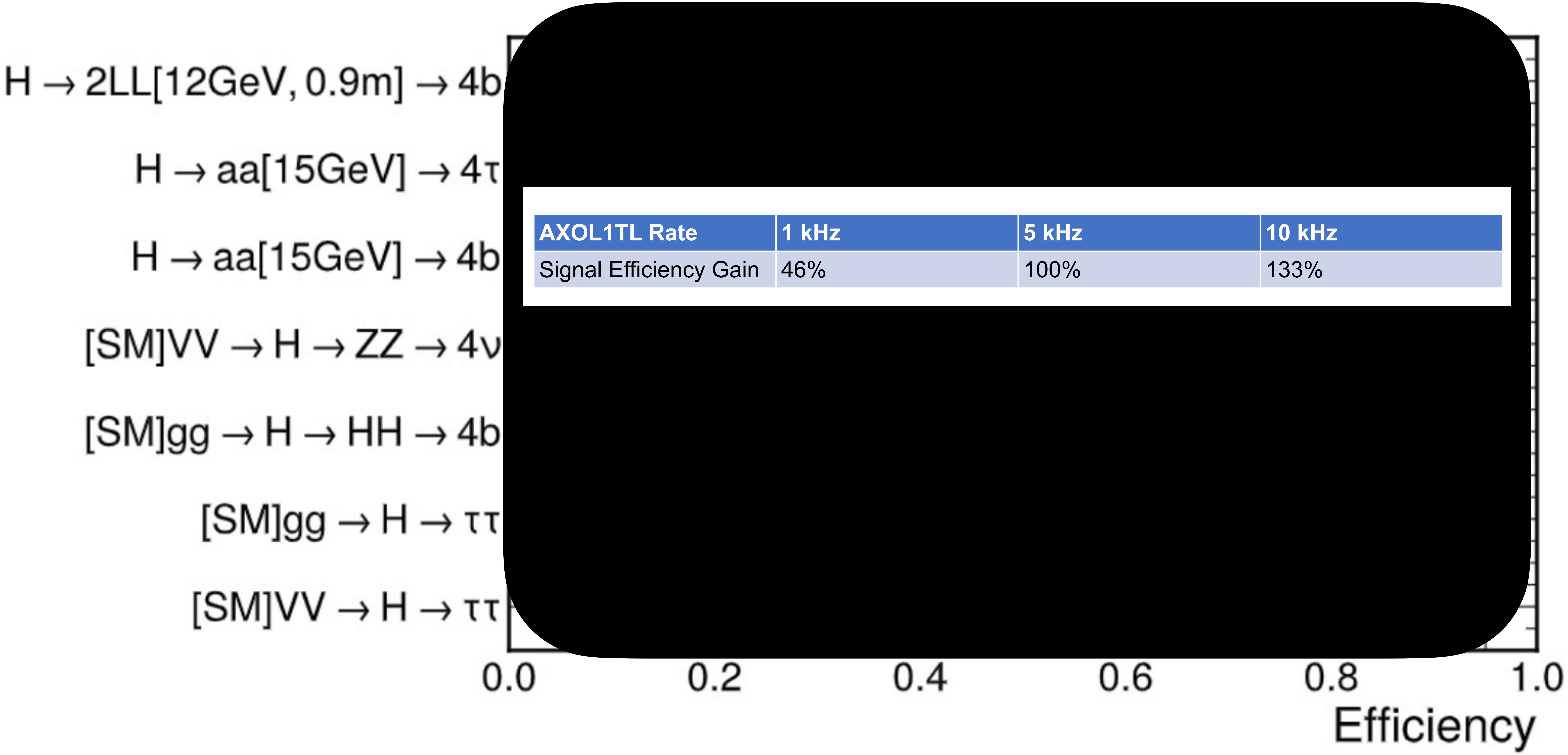
This model, and thresholds evaluated on test set deployed in the trigger!





uGT upstream
MET
12 e/gamma
8 muon
12 Jets
prune
MET
4 e/gamma
4 muon
10 Jets
concatenate
flatten
bit shift
hls 4 ml
ap_fixed<8,6>[57]
Dense [32]
kernel: fixed<6,3>
bias: fixed <10,7>
activ: qrelu fixed<10,7>
<32>
Dense [16]
kernel: fixed<6,3>
bias: fixed <10,7>
activ: qrelu fixed<10,7>
<16>
Dense [13]
kernel: fixed<6,3>
bias: fixed <10,7>
activ: qrelu fixed<10,7>
<13>
μ2

2023EphZB
Train QKeras
Model .h5
Evaluate QKeras
Plots
Thresholds
hls 4 ml
uGT HLS fwk
uGT HLS (C++)
Model HLS (C++)
pybind11
Standalone
Emulator
CMSSW Emulator
cms-hls4ml
Coming soon
Offline Validation
L1 NTuples (w/TC)
Online Monitoring (Prometh.)
Test Vectors
uGT VHDL fwk
Model VHDL
HDL Simulation
L1 keys
uGT bitfiles
(example)
IT AIN'T EASY!

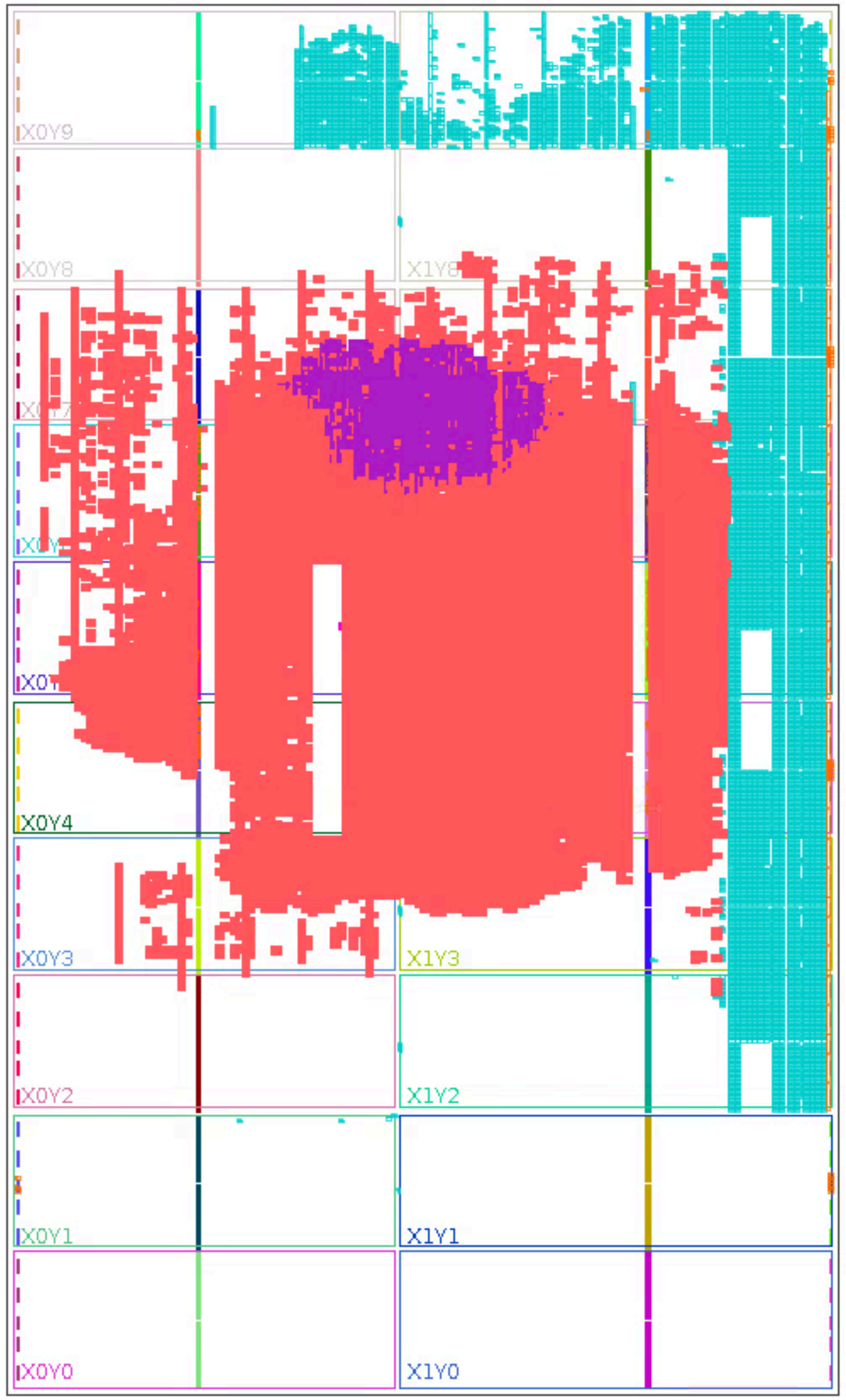


AXOL1TL Rate	1 kHz	5 kHz	10 kHz
Signal Efficiency Gain	46%	100%	133%

For various BSM and SM signals, significantly increase signal efficiency adding AD!

	Latency	LUTs	FFs	DSPs	BRAMs
AXOL1TL	2 ticks 50 ns	2.1%	~0	0	0

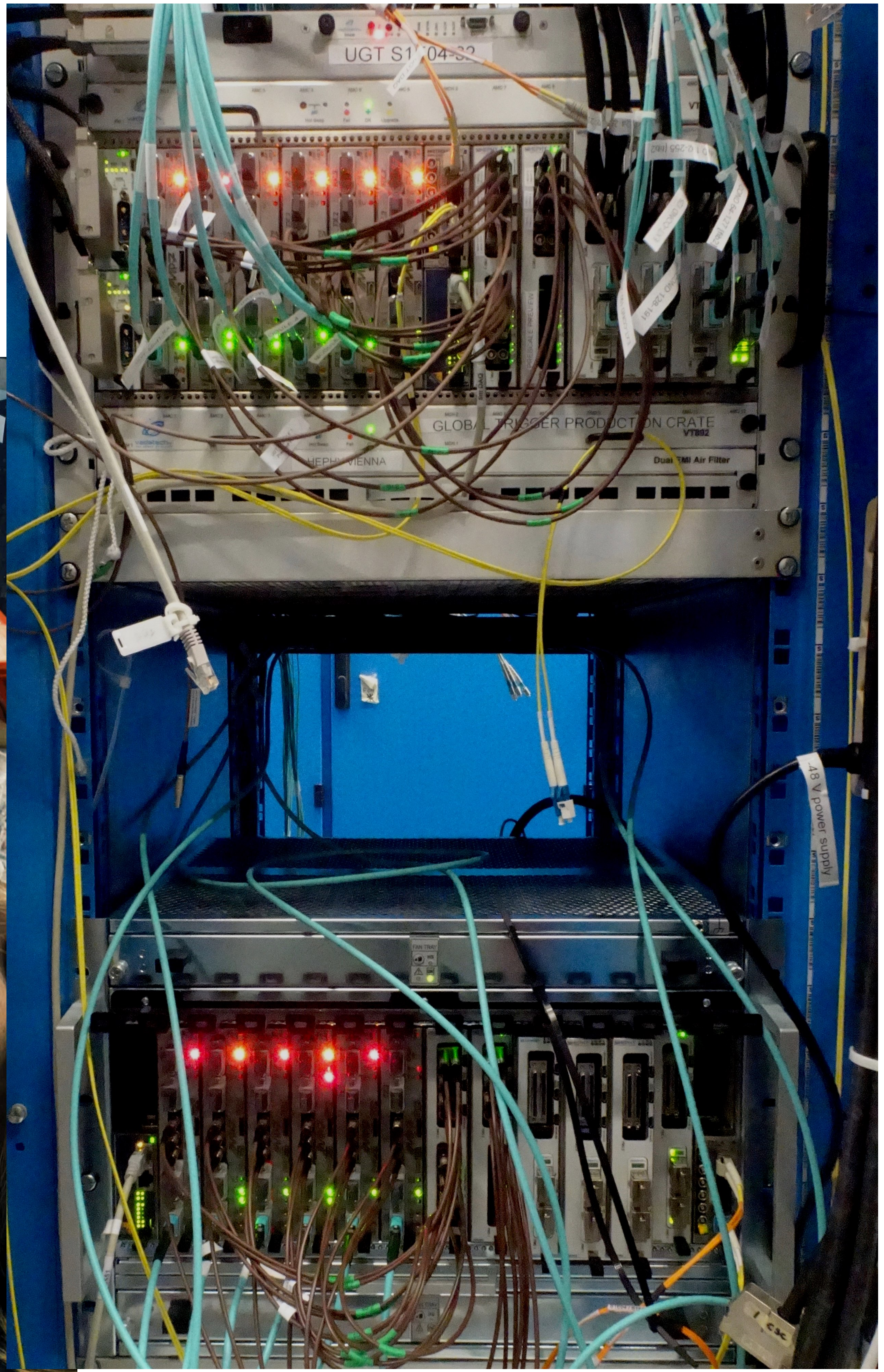
50 nanosecond anomaly detection!



uGT test crate

CMS Global Trigger test crate:

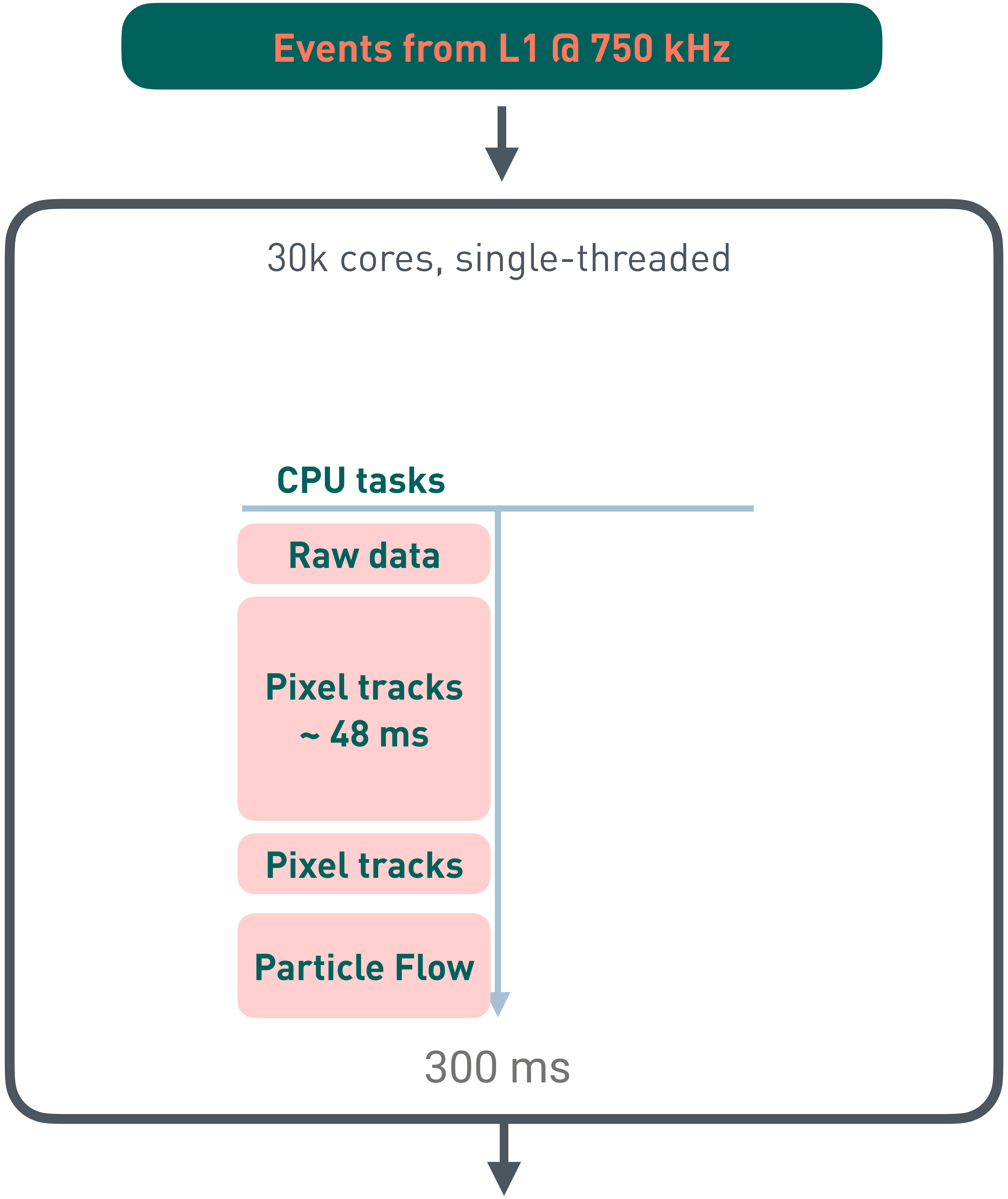
- Copy of main GT system, receiving the same input data, but not used to trigger CMS
- Excellent test bench for future ML algorithms targeting L1T FPGAs
- AXOL1TL integrated since late 2023



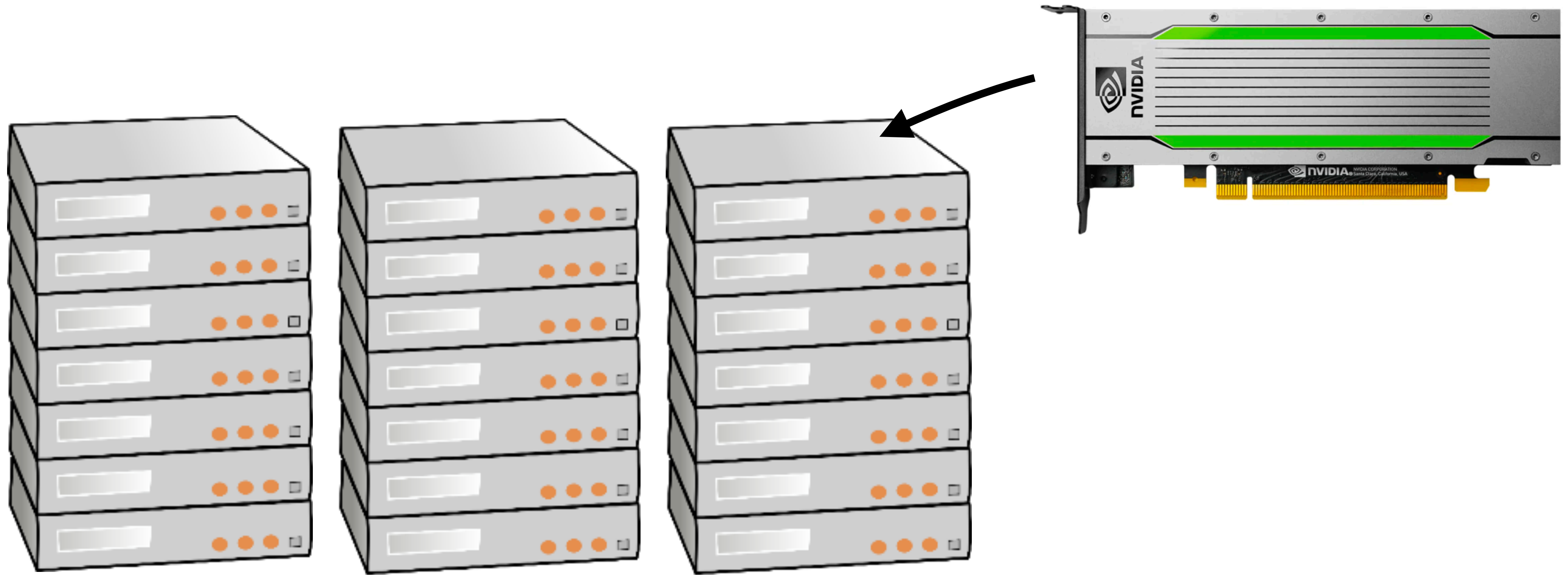

CMS
CMS Experiment at the LHC, CERN
Data recorded: 2023-May-24 01:42:17.826112 GMT
Run / Event / LS: 367883 / 374187302 / 159

High Level Trigger:
25’600 CPUs / 400 GPUs
Latency: 3-400 ms
7.5 kHz
Geneva
LHCb
ATLAS
ALICE
750 kHz
LHC

CPU node (128 cores)

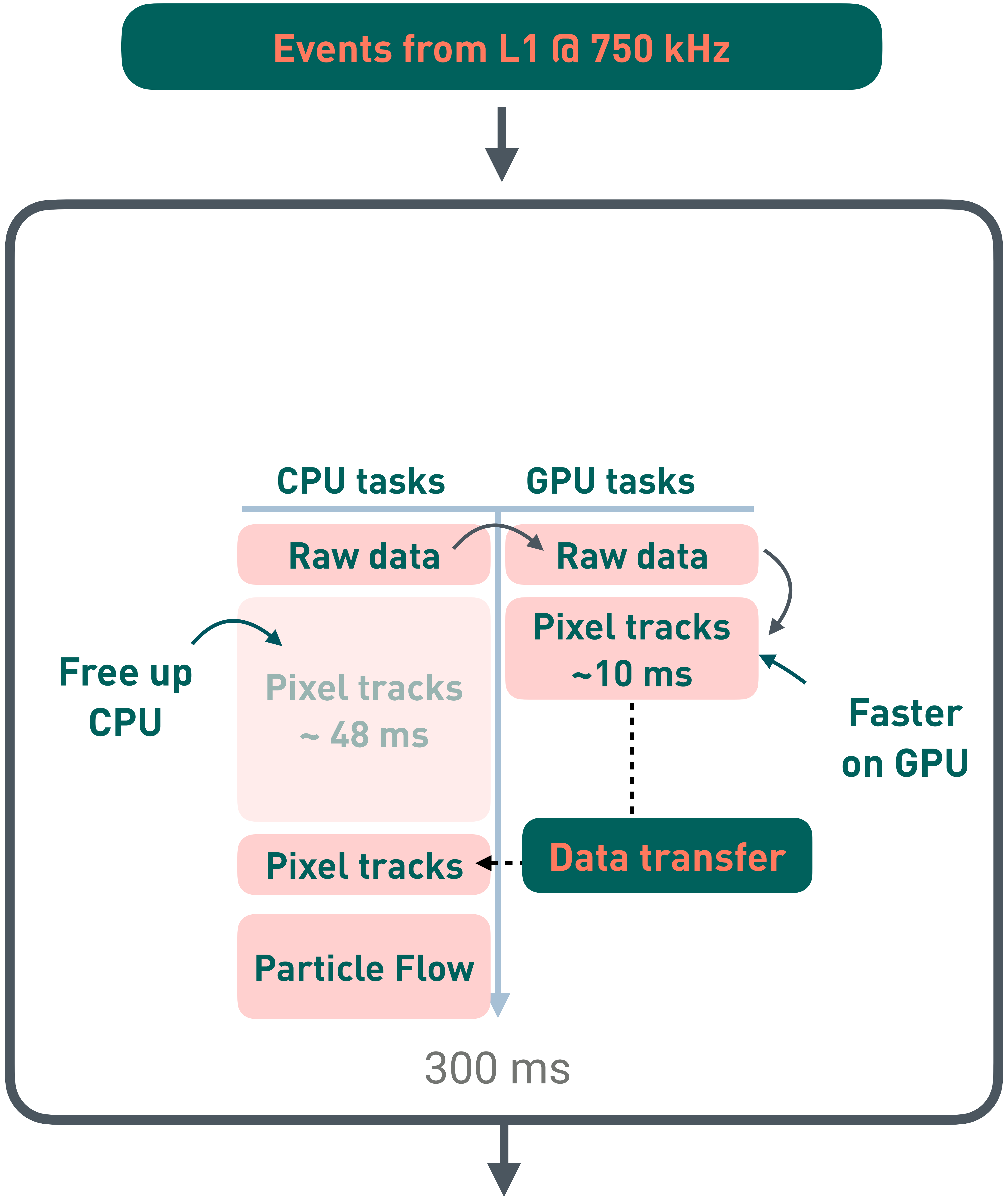


https://cms.cern/news/first-collisions-reconstructed-gpus-cms



Offload resource-intensive computations to GPU
- Increase throughput by x1.8, reduce power by 30%

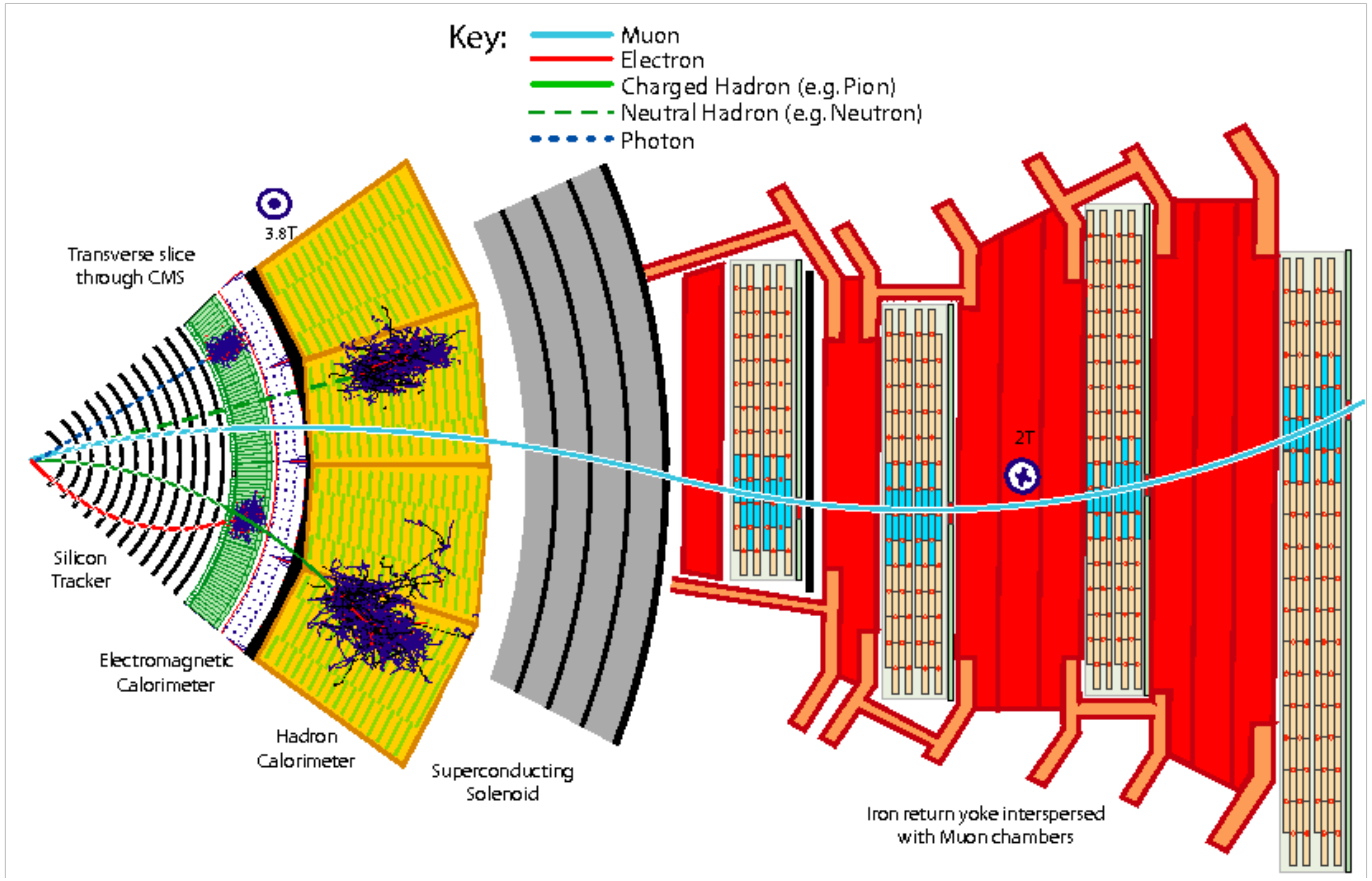
- Will offload 50-80% to computing accelerators!



ML for fast reconstruction

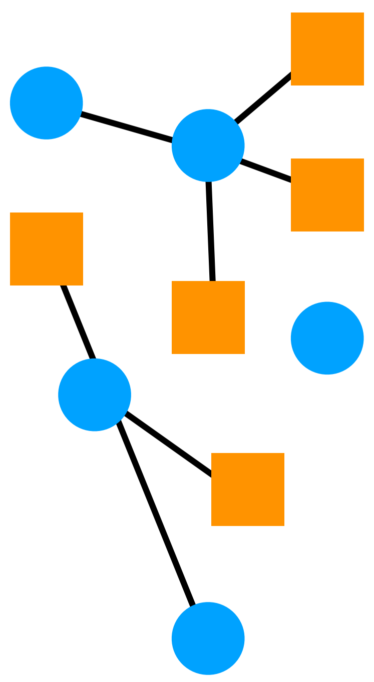
Particle Flow: Best reconstruction at HLT

- Slow, cannot run on all events (currently 17%)

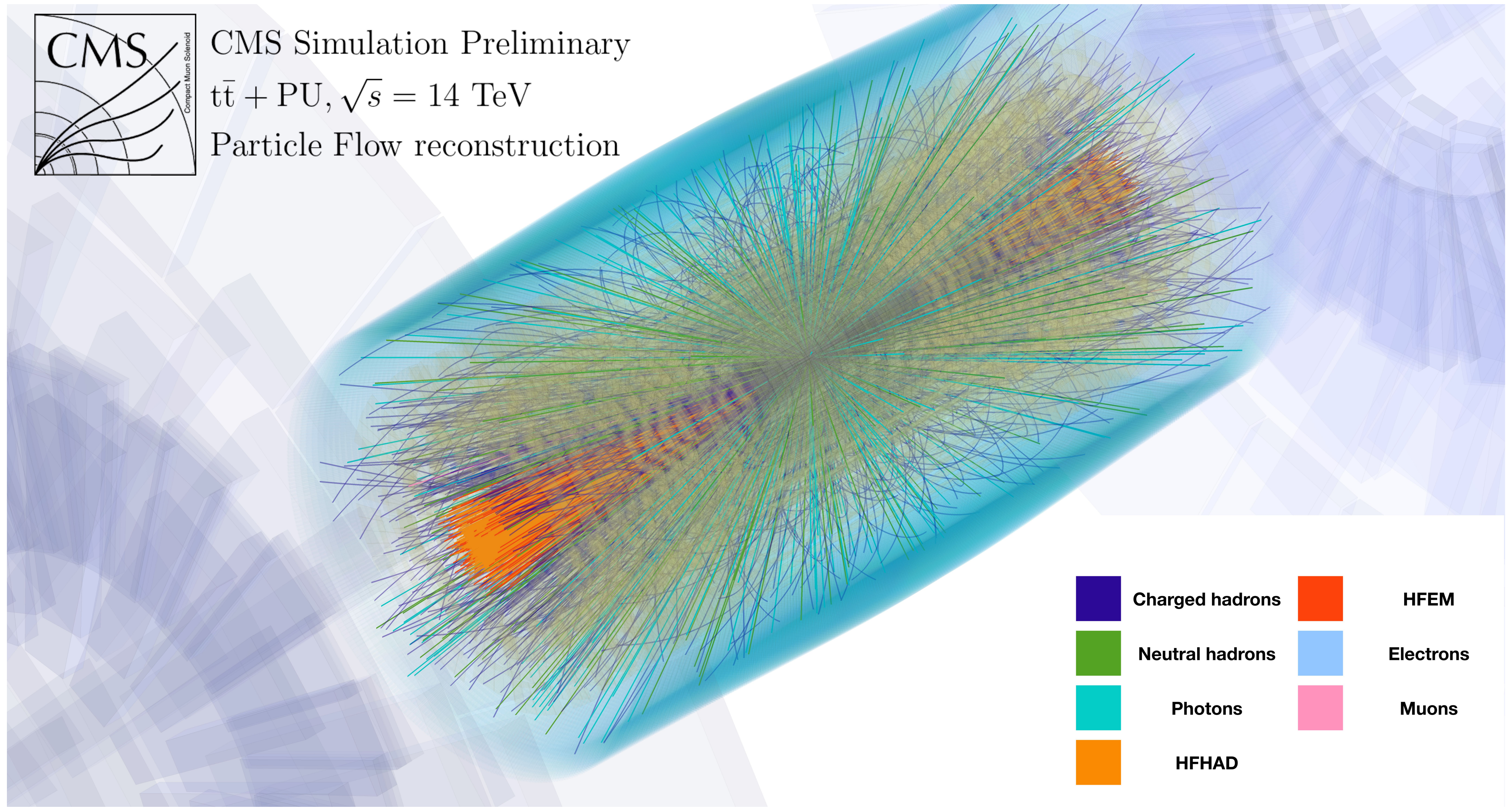


ML for fast reconstruction

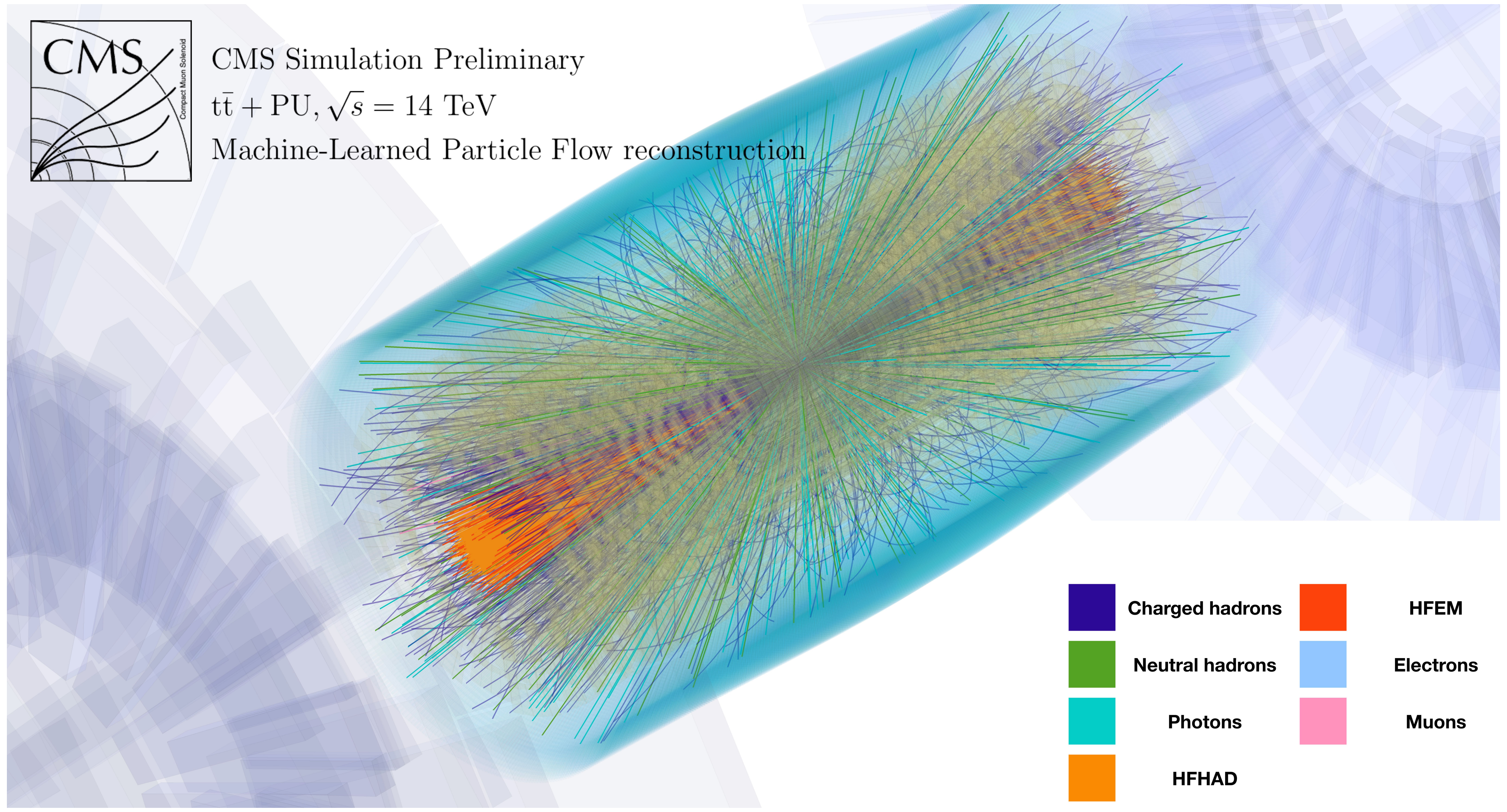
Graph Deep Neural Networks:“fast” approximations of ParticleFlow



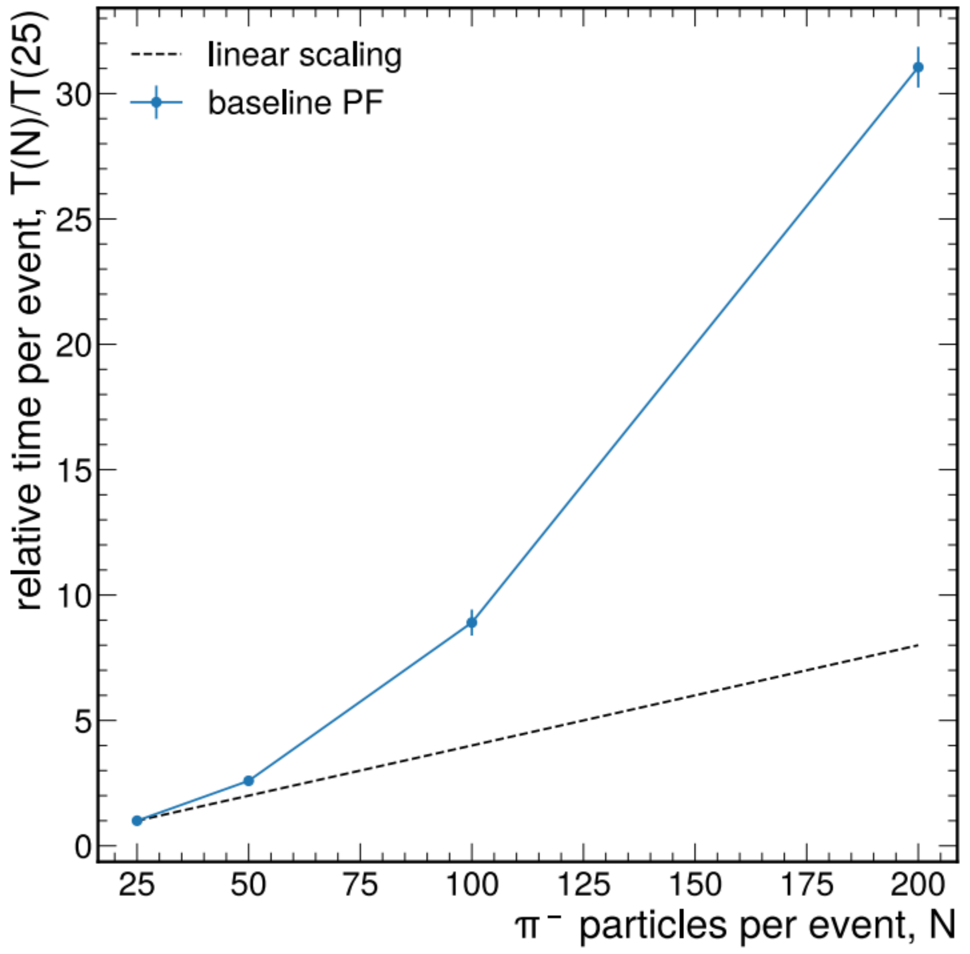
Classical Particle Flow

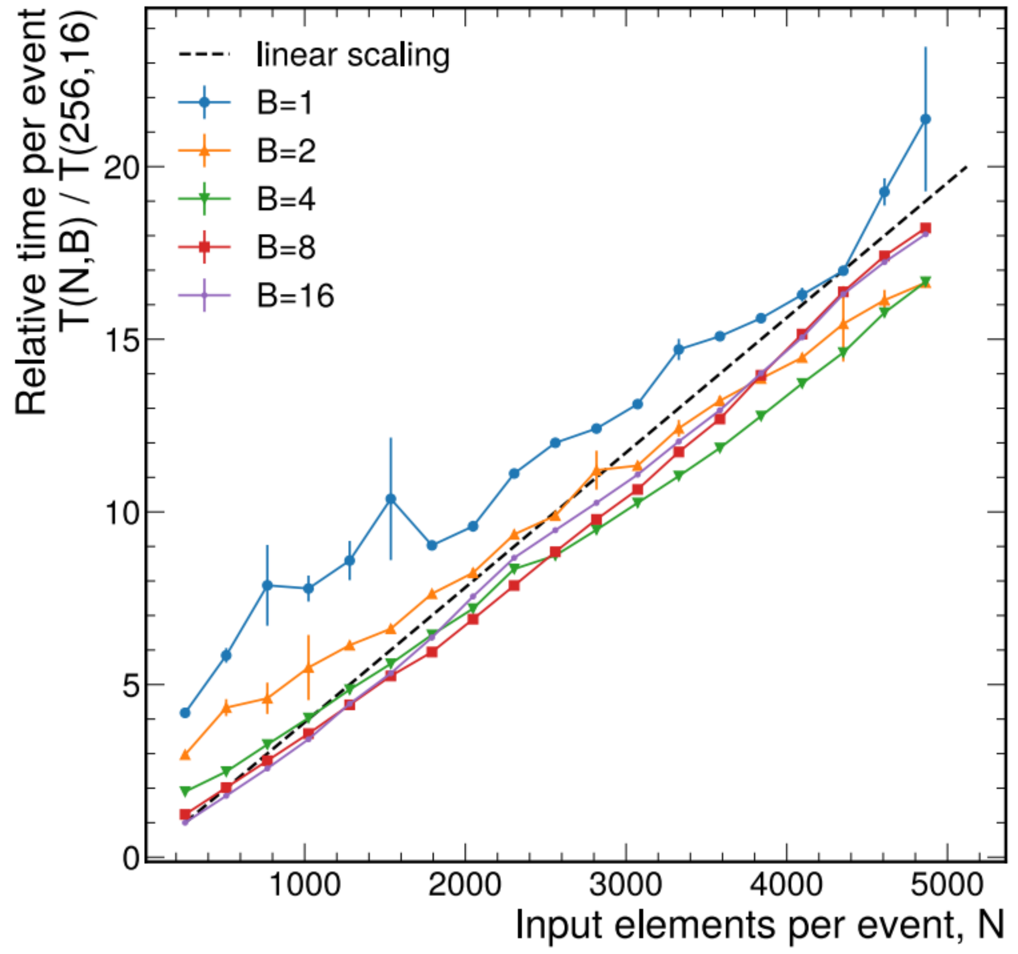


Graph Neural Network



arXiv:2101.08578

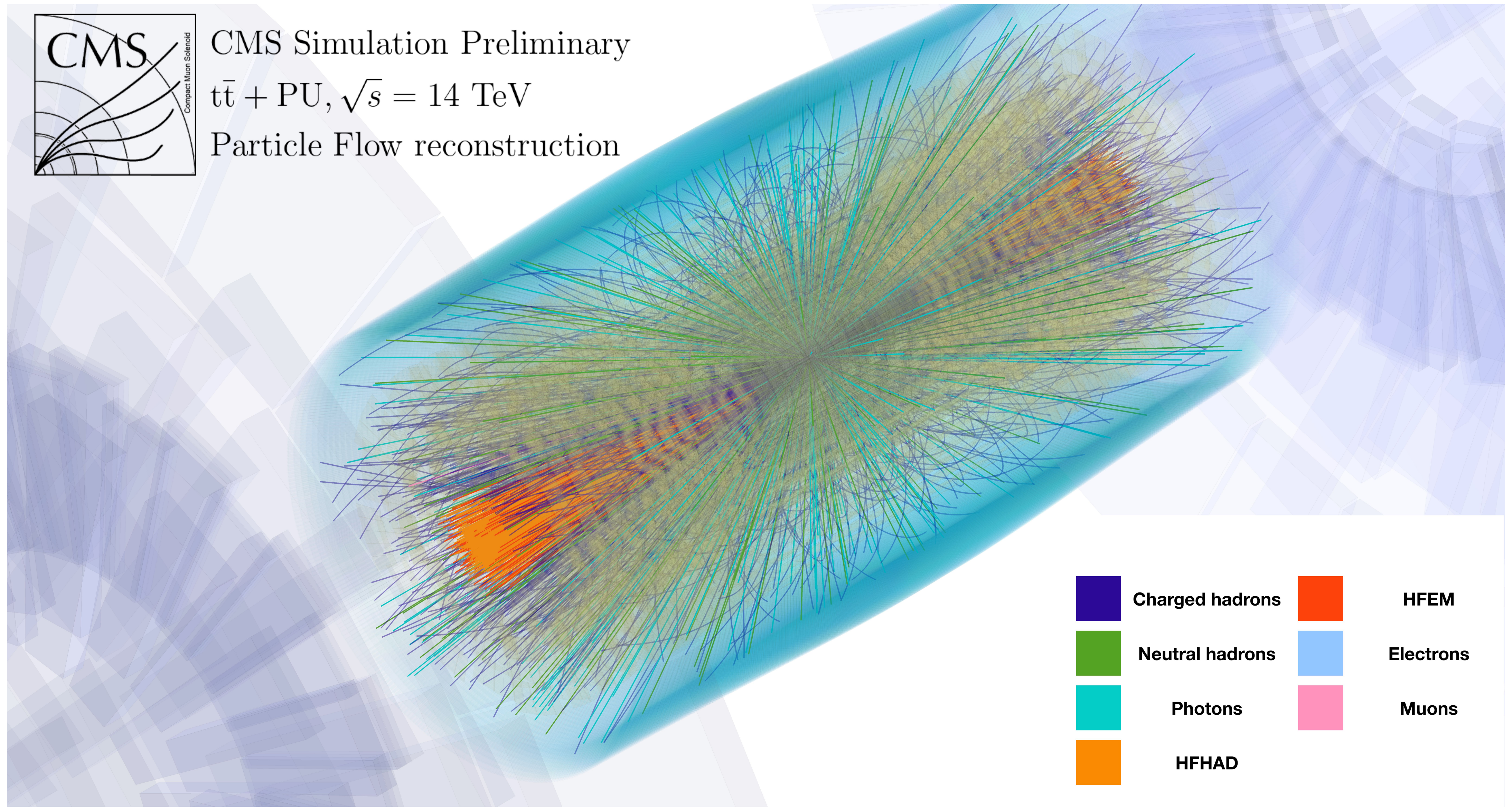




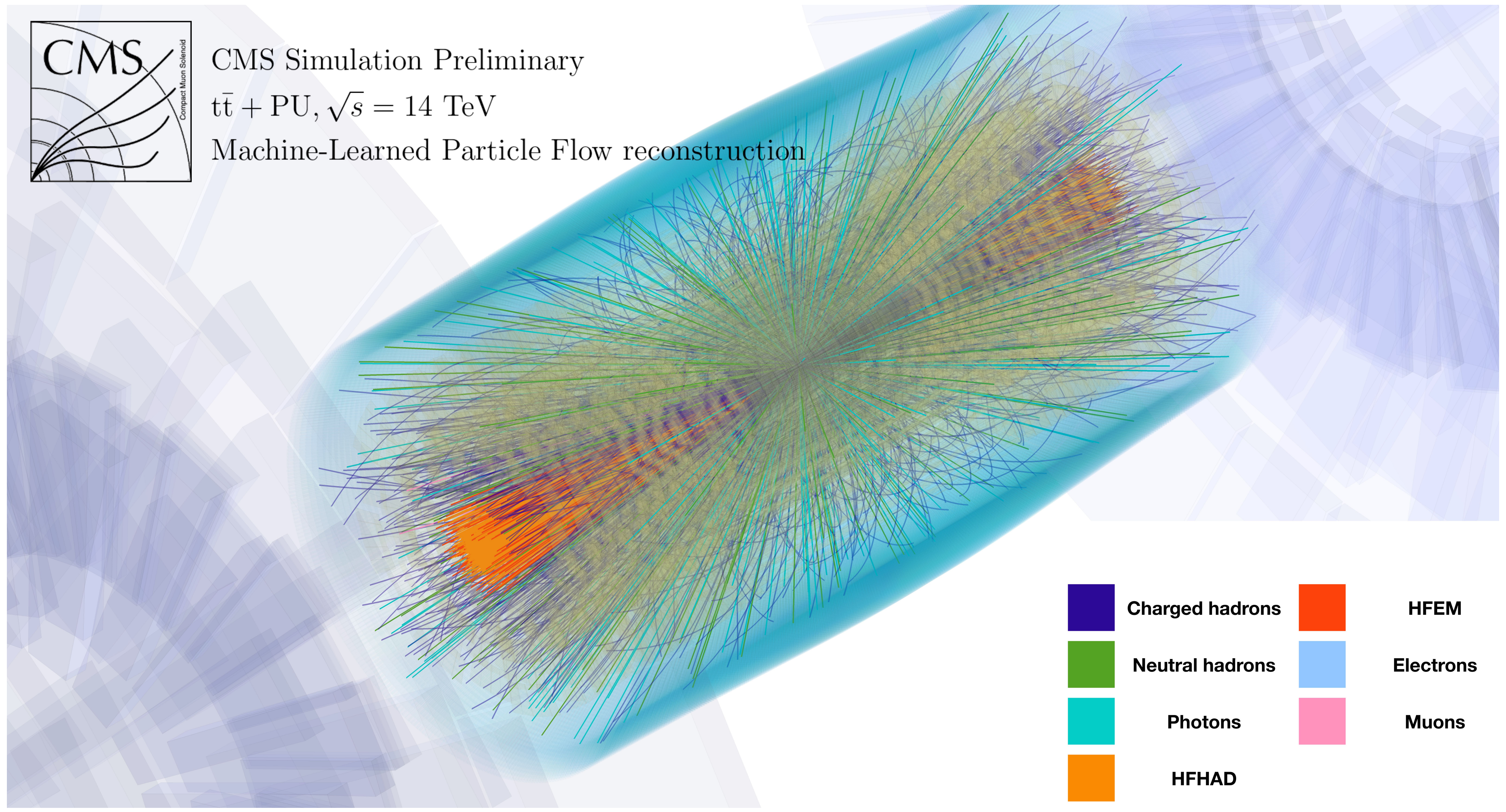
PF baseline scales non-linearly with increasing input size

GNN-based model inference time scales approximately linearly with increasing input size

Classical Particle Flow

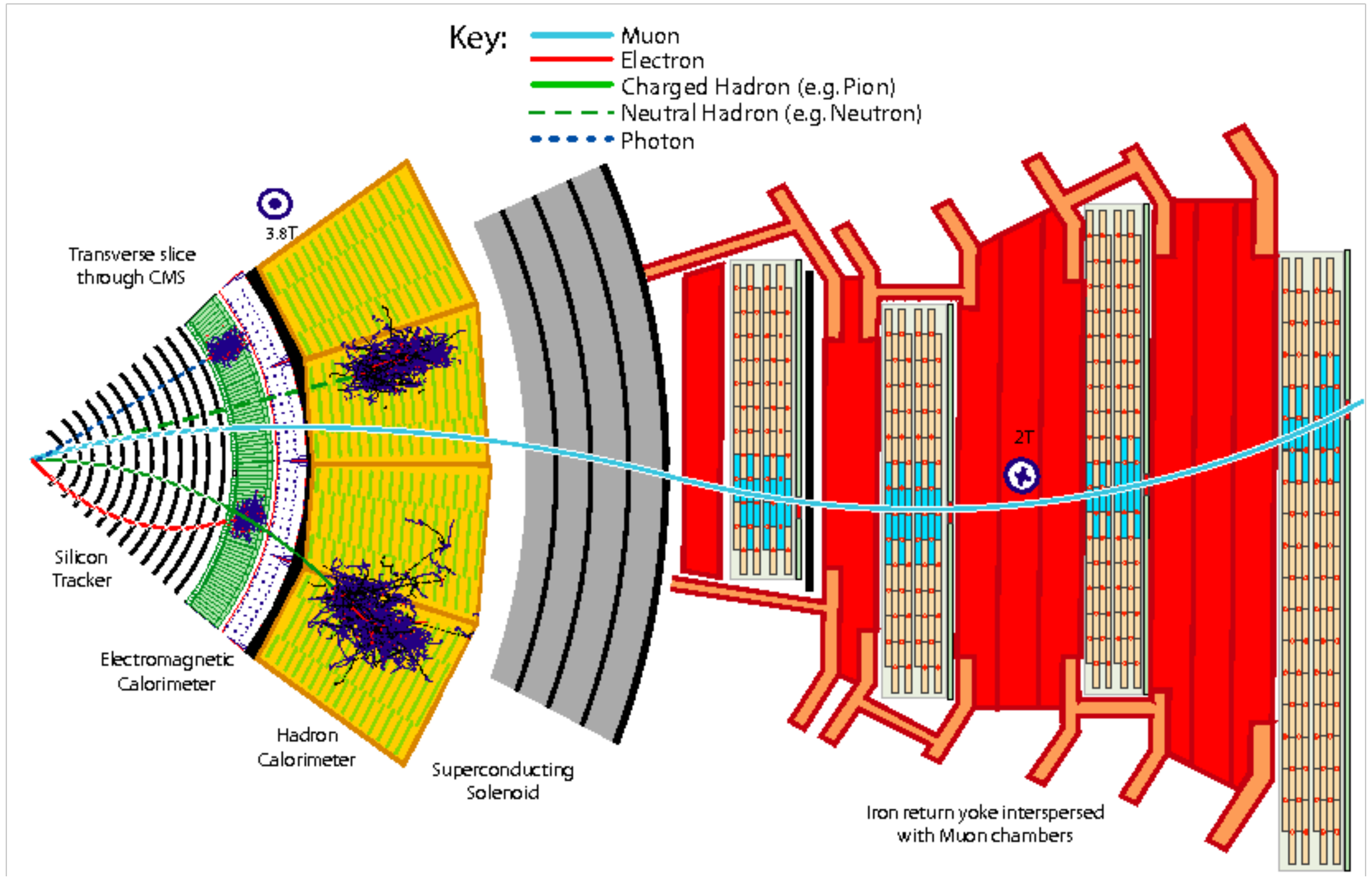


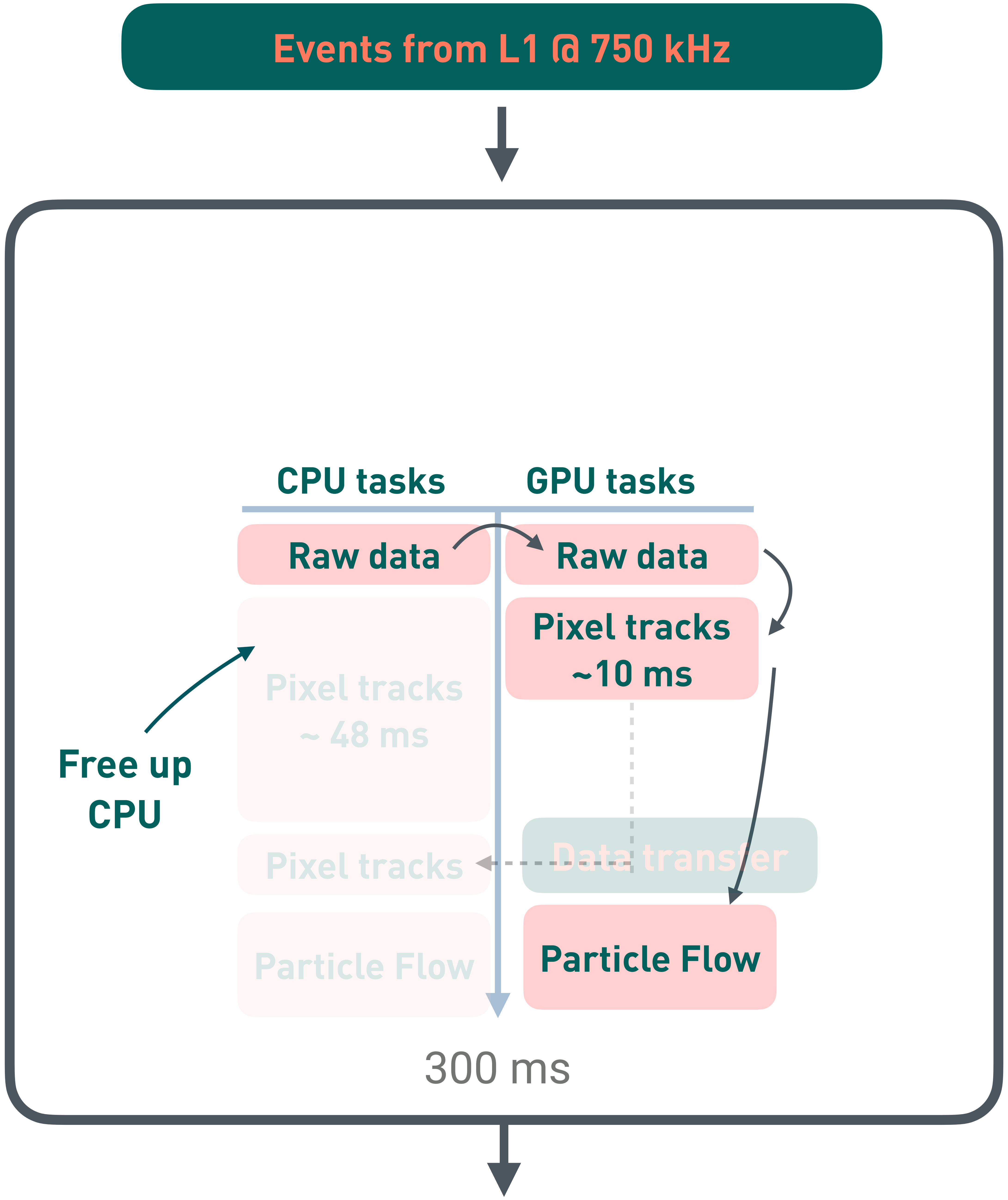
Graph Neural Network



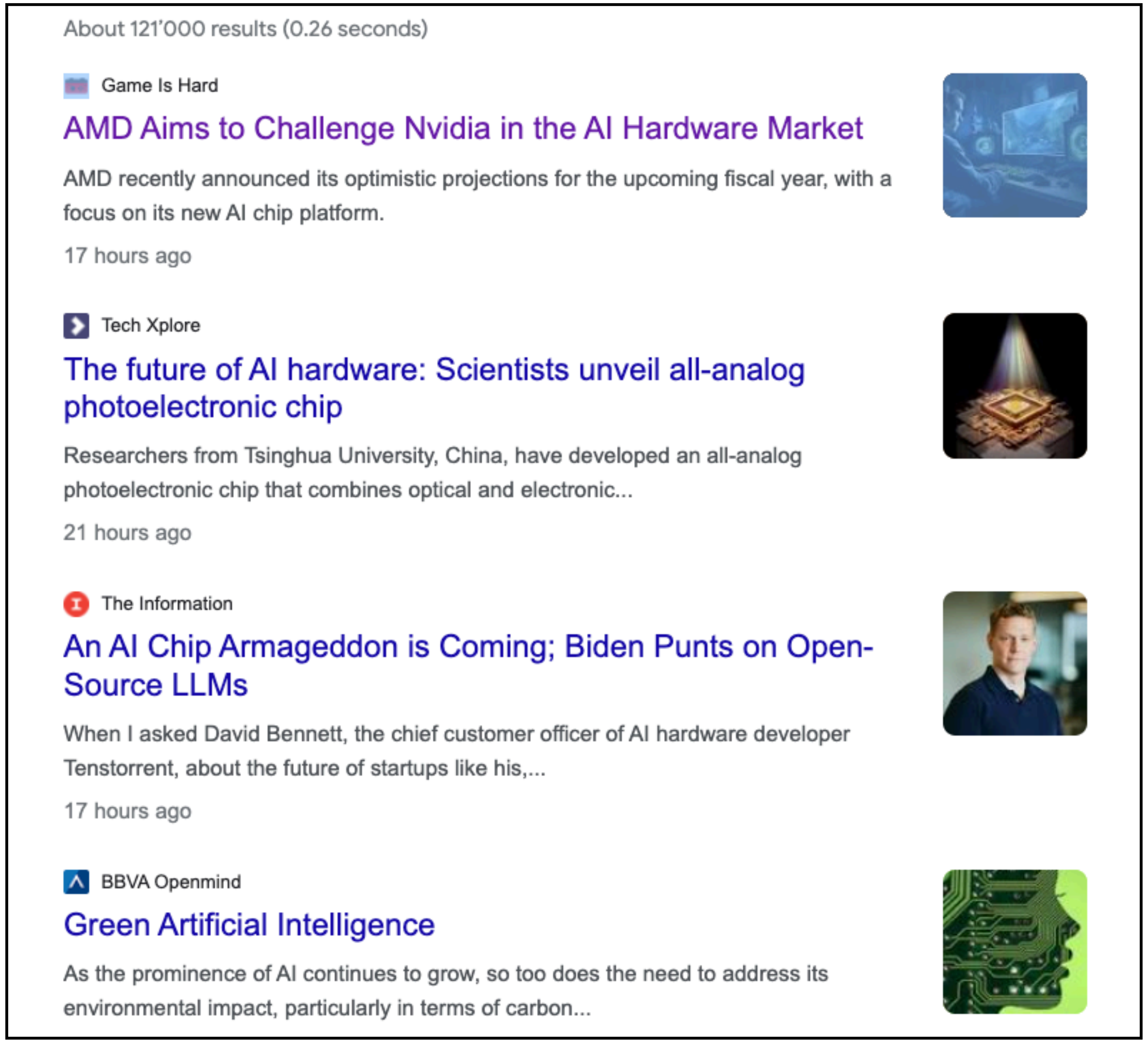
arXiv:2101.08578

ML for fast reconstruction





AI hardware


About 121'000 results (0.26 seconds)

Game Is Hard
AMD Aims to Challenge Nvidia in the AI Hardware Market
AMD recently announced its optimistic projections for the upcoming fiscal year, with a focus on its new AI chip platform.
17 hours ago

Tech Xplore
The future of AI hardware: Scientists unveil all-analog photoelectronic chip
Researchers from Tsinghua University, China, have developed an all-analog photoelectronic chip that combines optical and electronic...
21 hours ago

The Information
An AI Chip Armageddon is Coming; Biden Punts on Open-Source LLMs
When I asked David Bennett, the chief customer officer of AI hardware developer Tenstorrent, about the future of startups like his,...
17 hours ago

BBVA Openmind
Green Artificial Intelligence
As the prominence of AI continues to grow, so too does the need to address its environmental impact, particularly in terms of carbon...

AI hardware

GNNs with Versal AI, P. Schwaebig

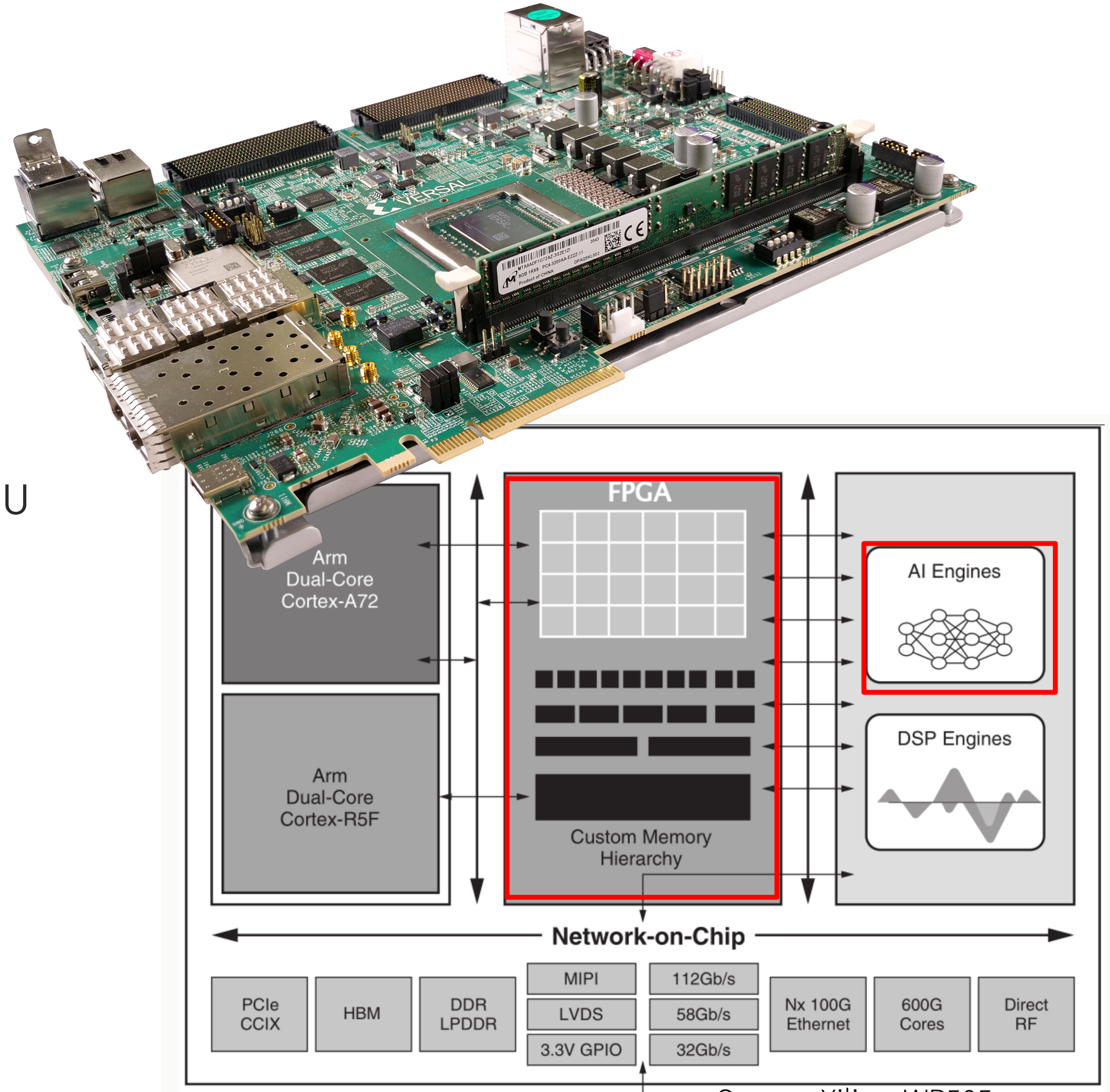
More and more dedicated AI processors on the market

Xilinx Versal AI processors

- Programmed in C/C++
- 400 AI processors, ~2M logic cells (FPGA), 2k DSPs, Arm CPU and RPU
- Data move back and forth between AI Engines and FPGA

Currently explored for real-time tracking in trigger application

- Interaction Network for pattern recognition (similar to **DeZoort et al**)
- Deployed on Xilinx Versal VC1902 ACAP



Source: Xilinx, WP505,


Subscribe to newsletters

Forbes

FORBES > INNOVATION > CLOUD

Groq's Record-Breaking Language Processor Hits 100 Tokens Per Second On A Massive AI Model

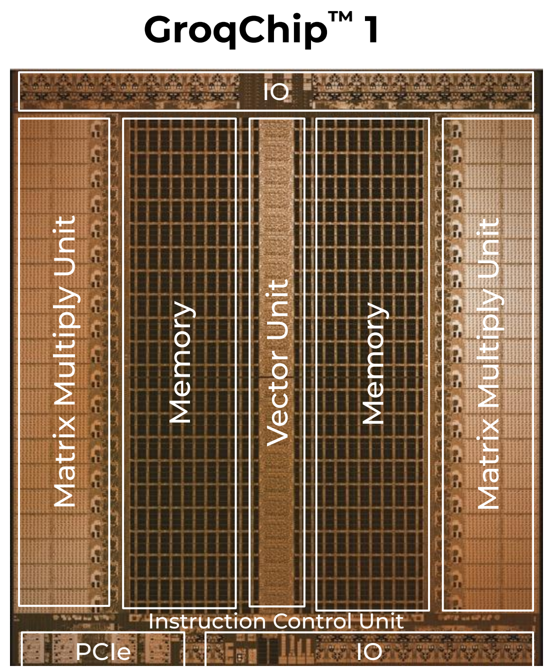
Paul Smith-Goodson Contributor
Moor Insights and Strategy Contributor Group

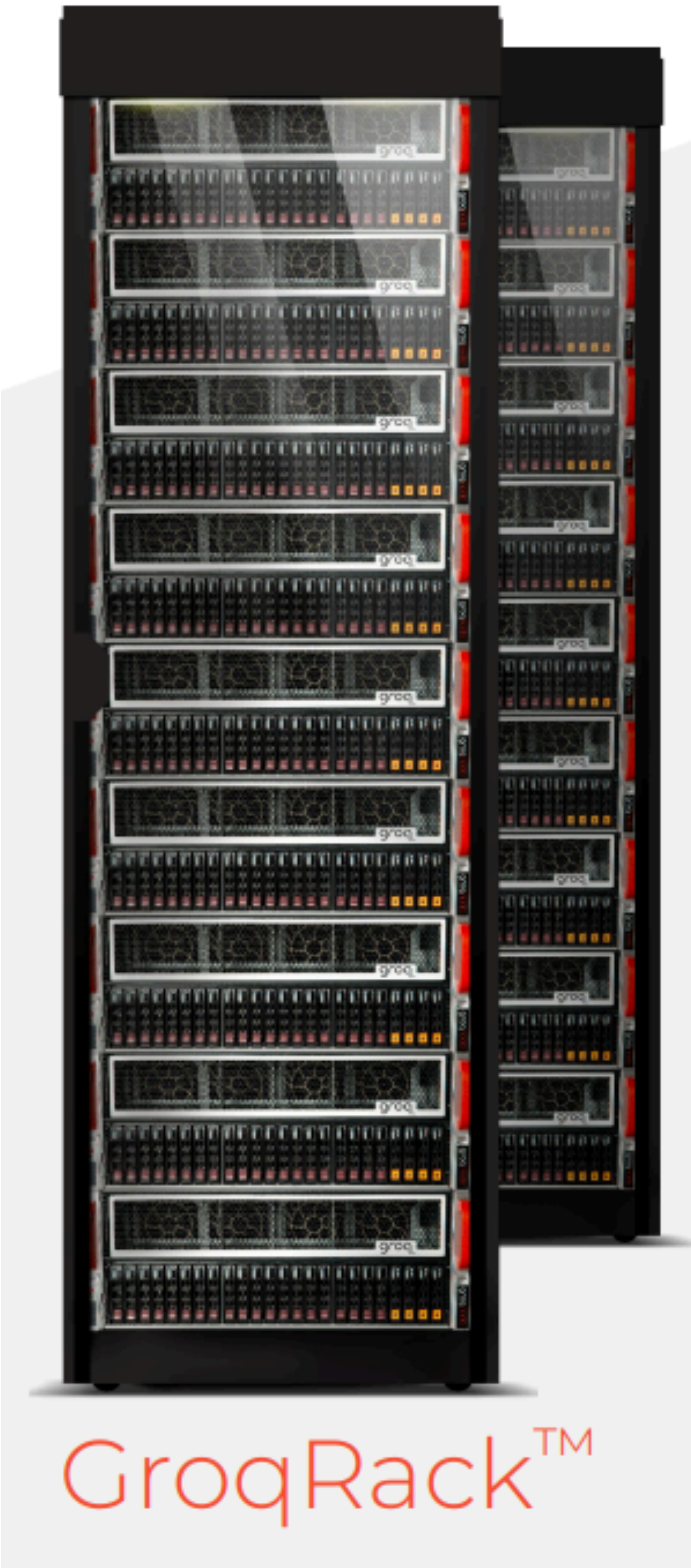
Follow

0

Aug 11, 2023, 03:29pm EDT

0:14 0:00
Ad 1 of 2

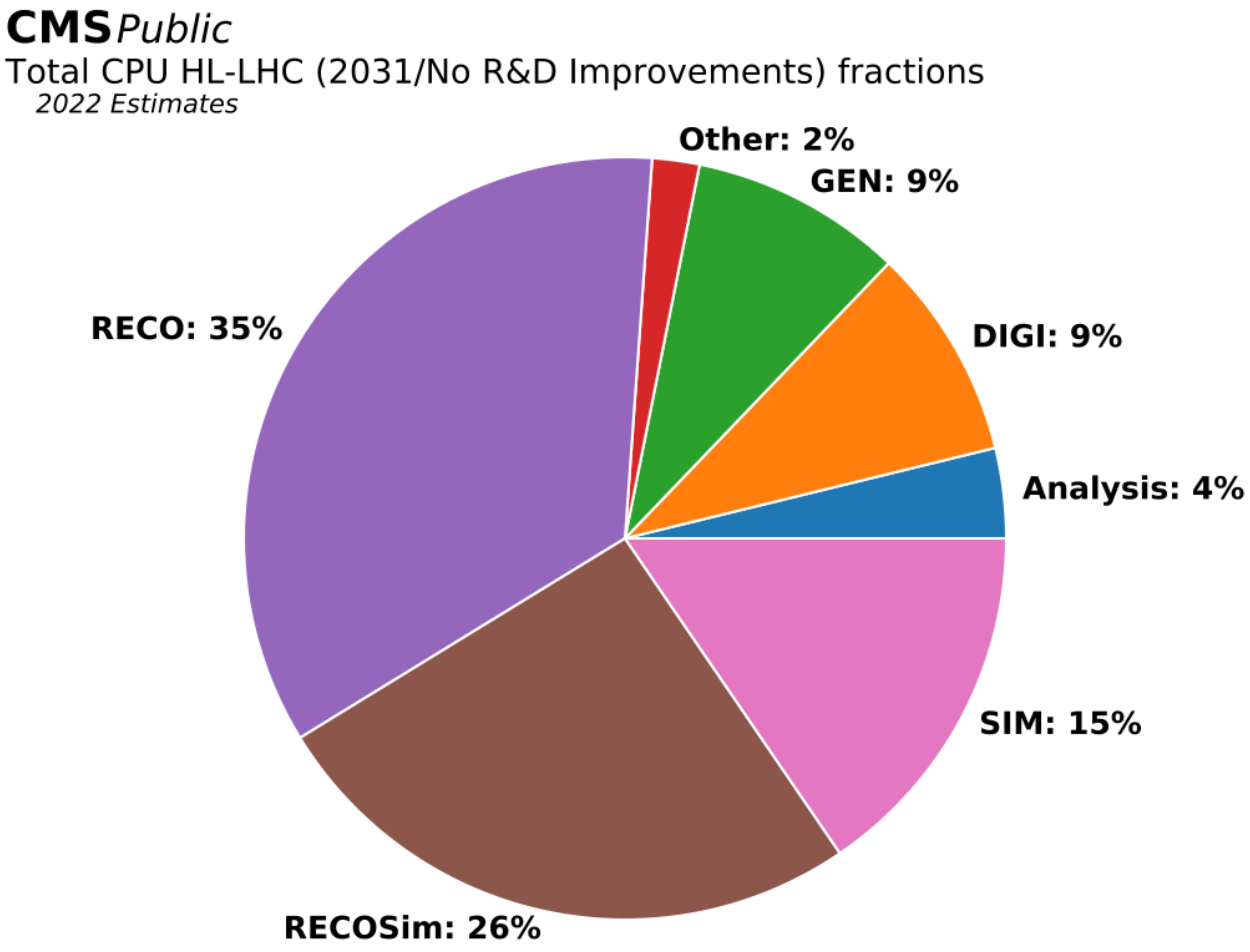




Groq: ultra-low latency dedicated language processor dedicated language processors

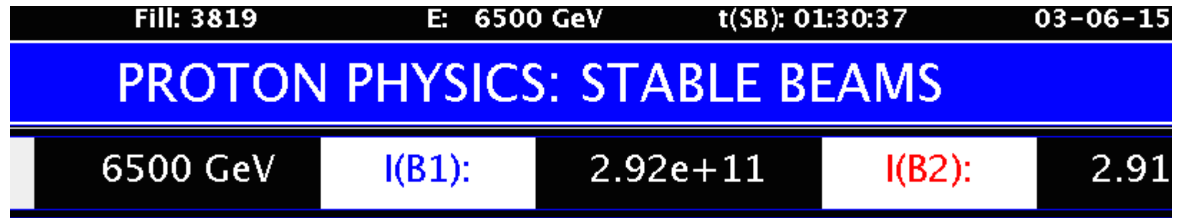
- Optimised for sequential data
- First ever 100 tokens/s (usually, ~10)

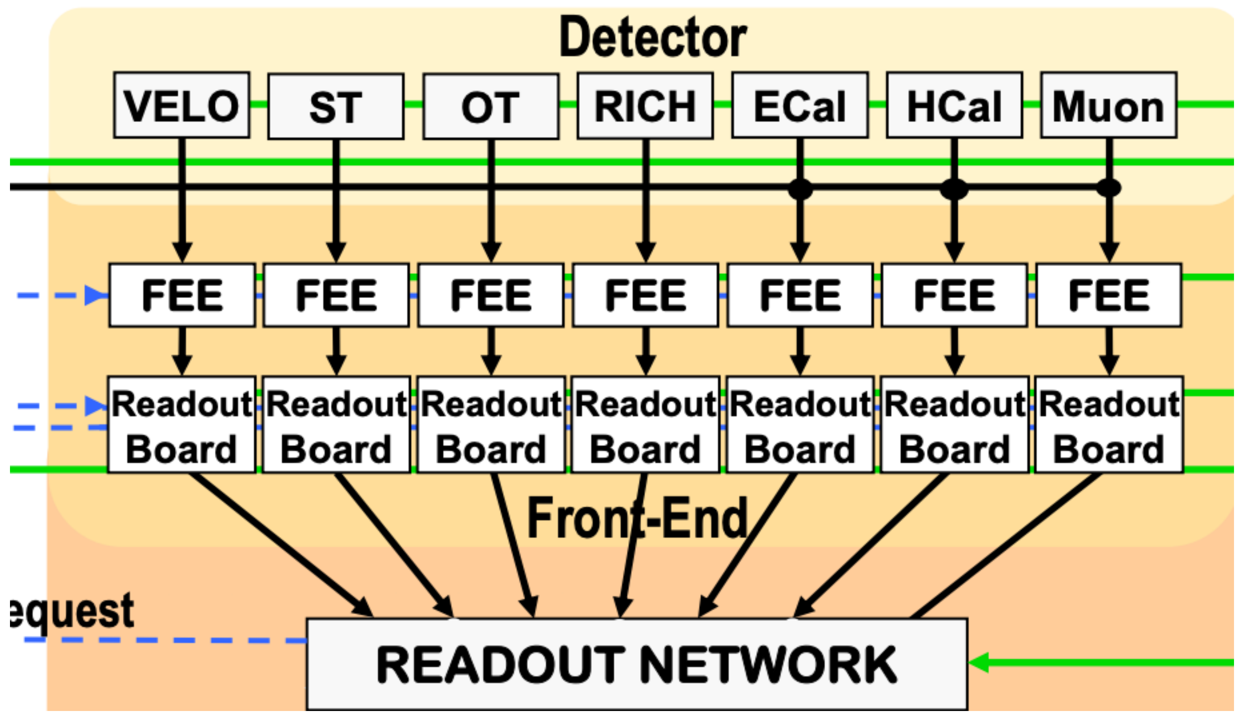
CMS Offline Computing Results



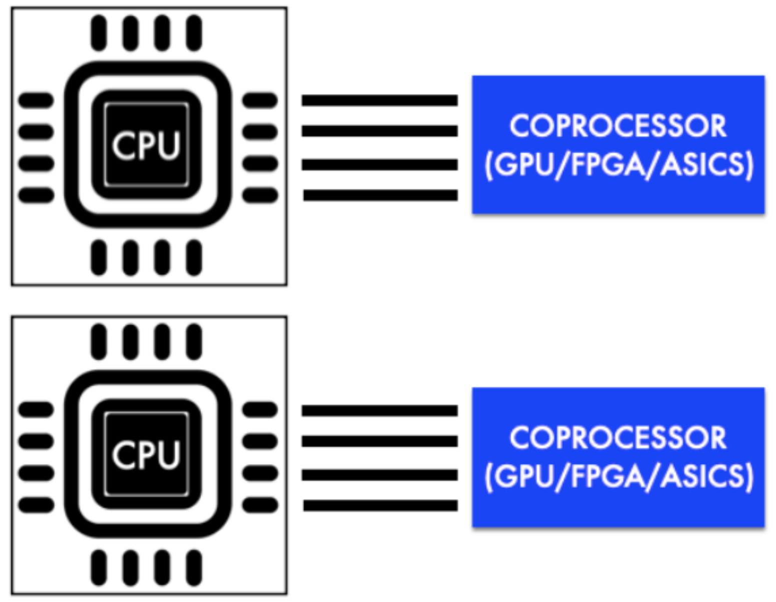
FPGAs as accelerators

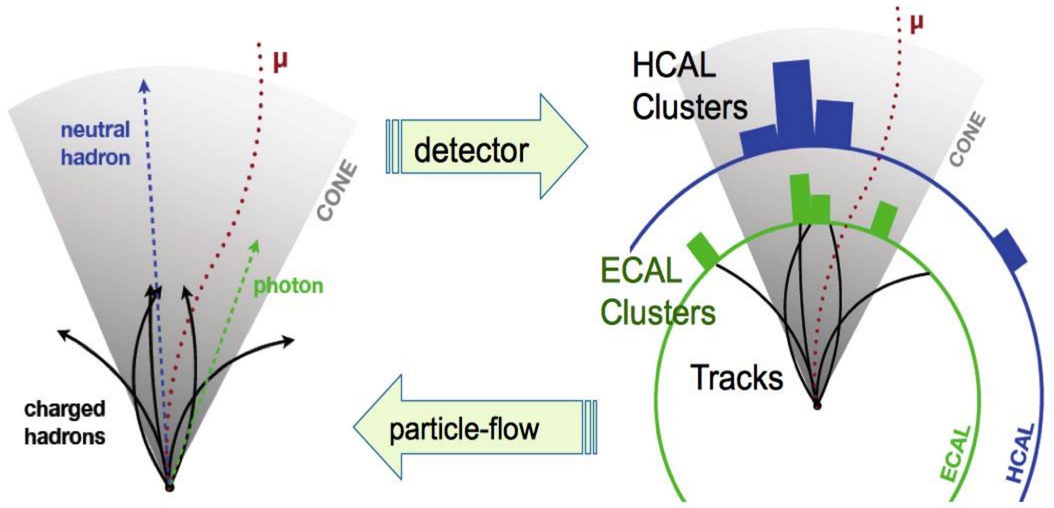
DAQ





CO-PROCESSING





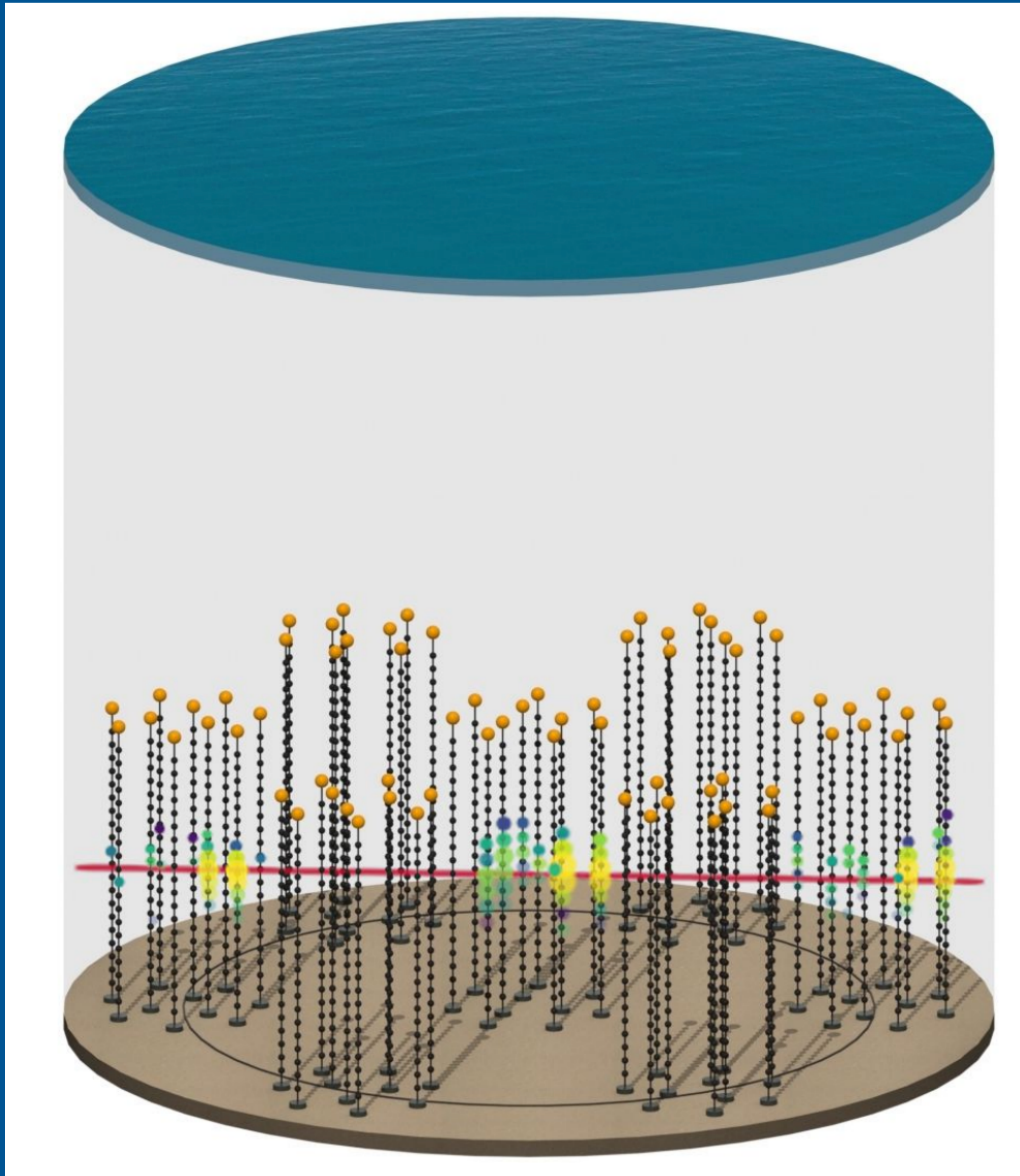
C. Beteta, I. Bezshyiko, N. Serra

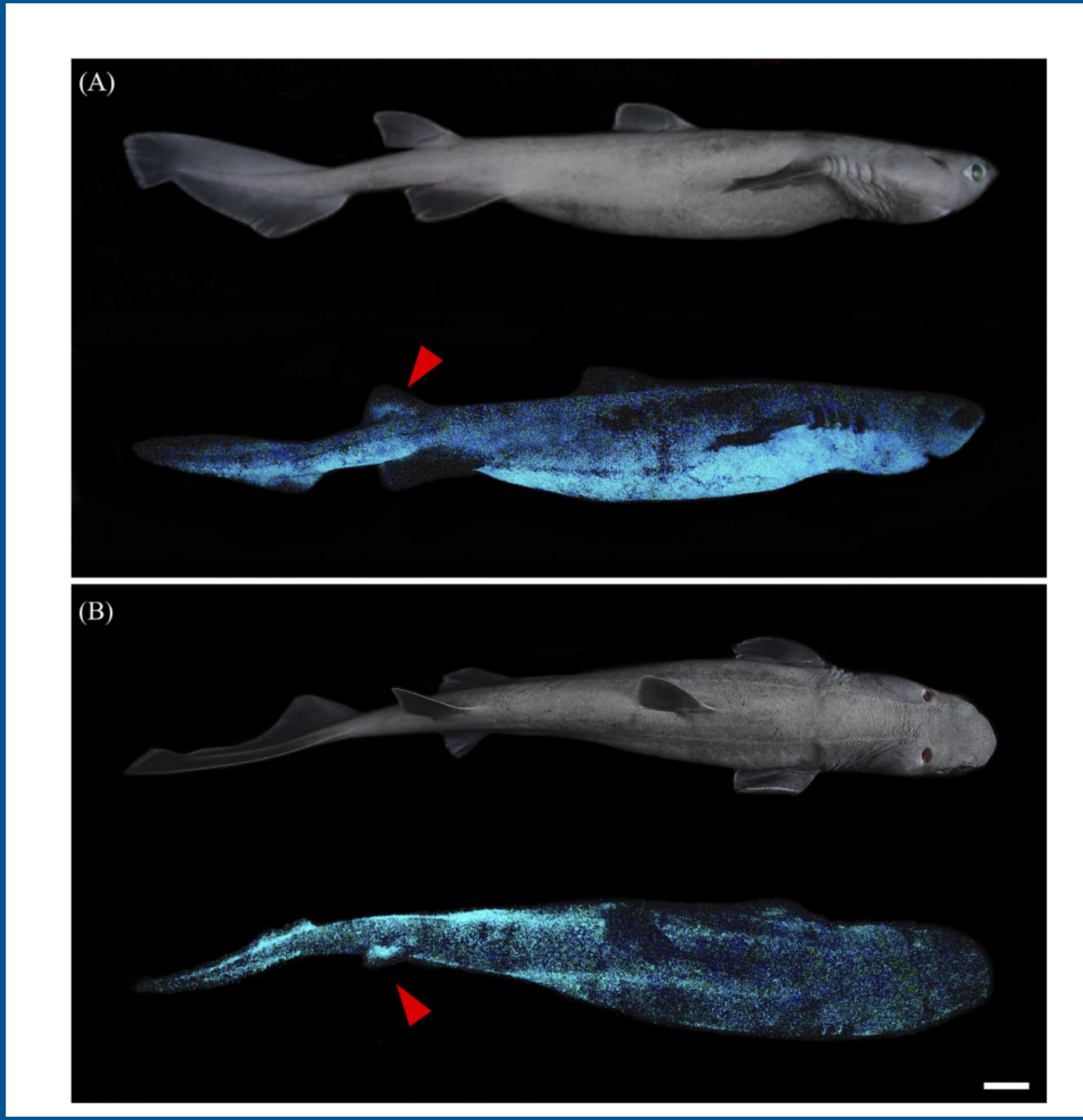
FaaST, D. Rankin et. al

Algorithm	Platform	Number of Devices	Batch Size	Inf./s [Hz]	Bandwidth [Gbps]
FACILE	AWS EC2 F1	1	16,000	36 M	23
FACILE	Alveo U250	1	16,000	86 M	55
FACILE	T4 GPU	1	16,000	8 M	5.1

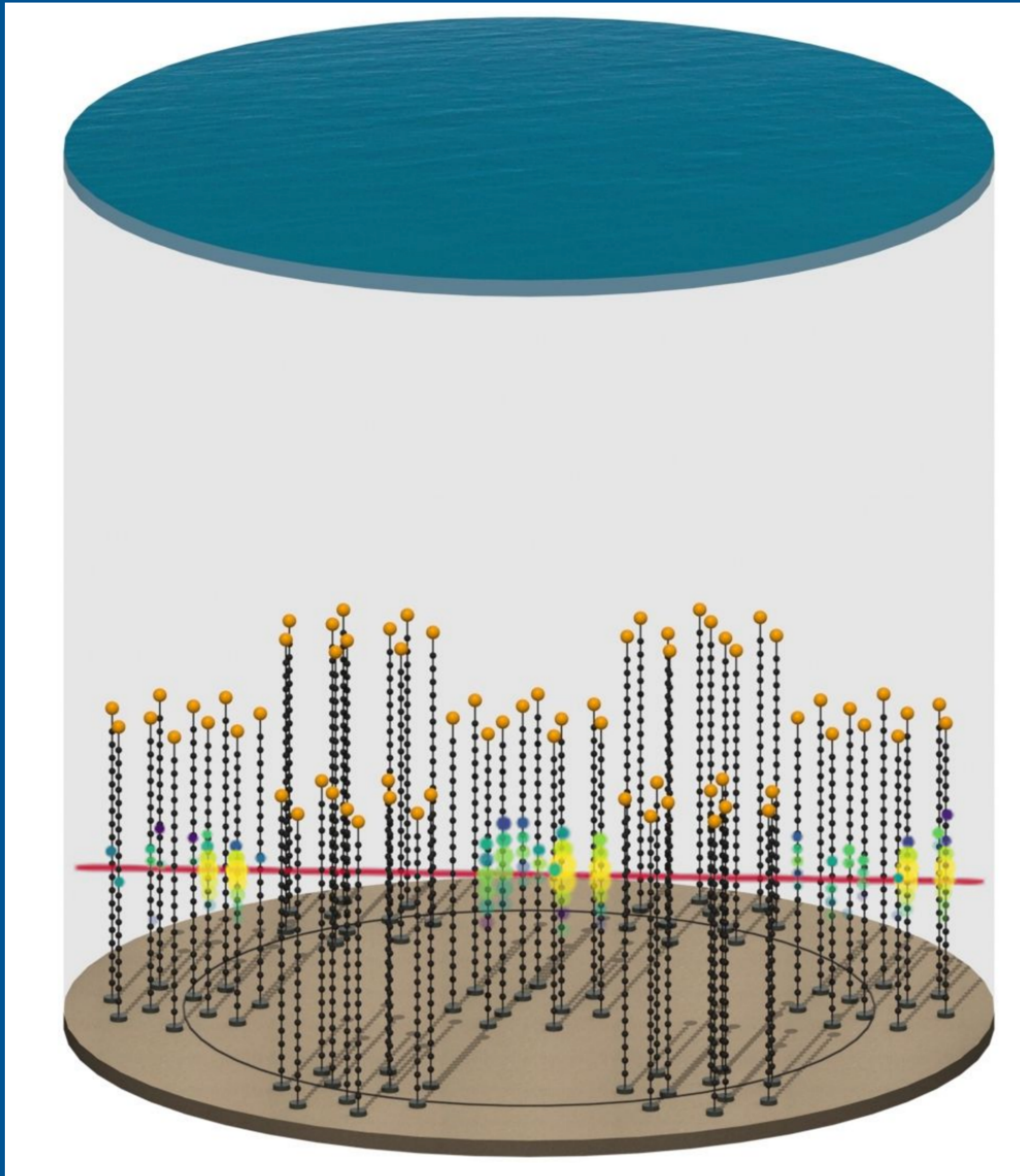
F. Capel et al.

Triggering in other experiments

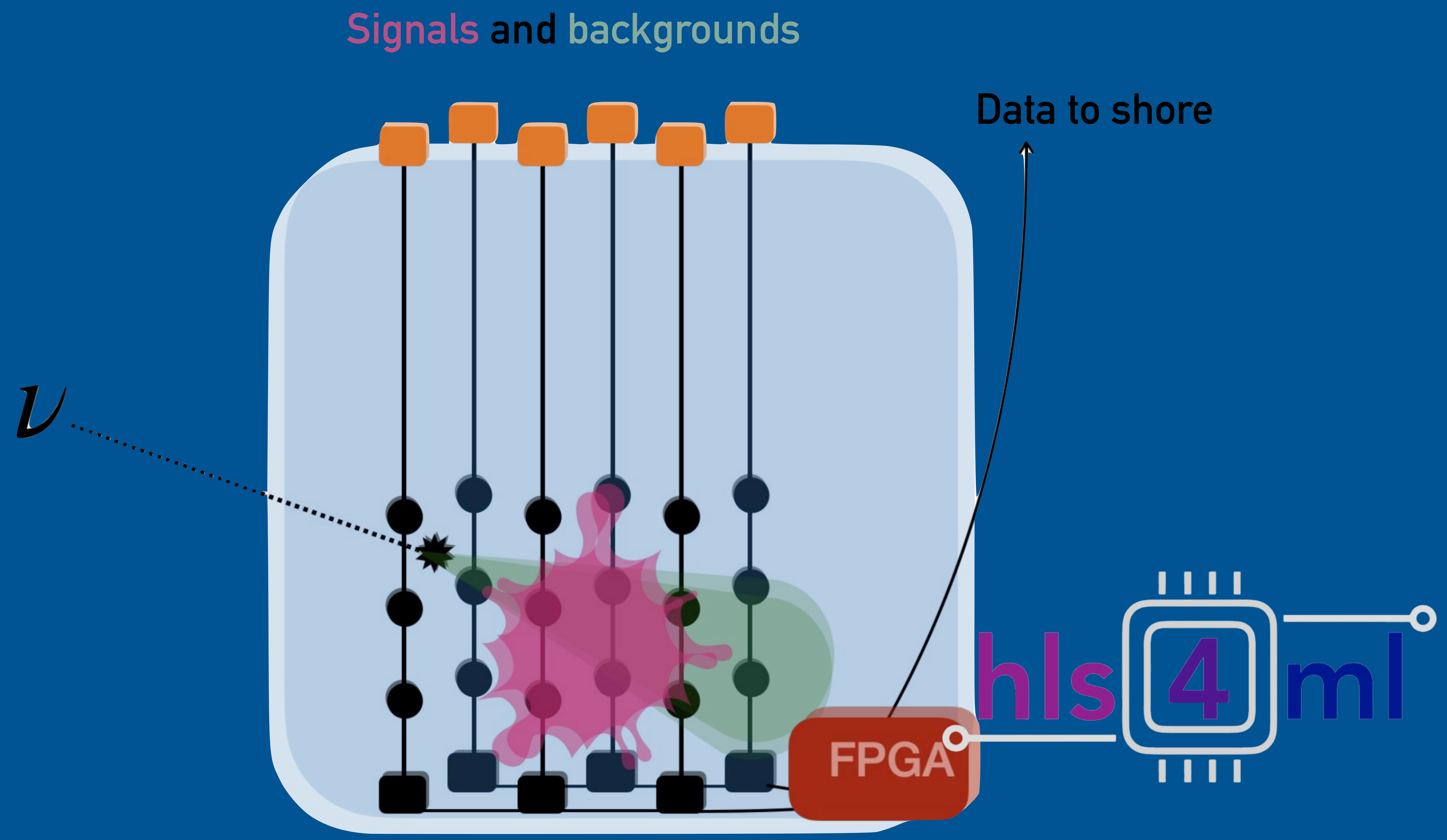




Bioluminescence bursts up to few MHz!



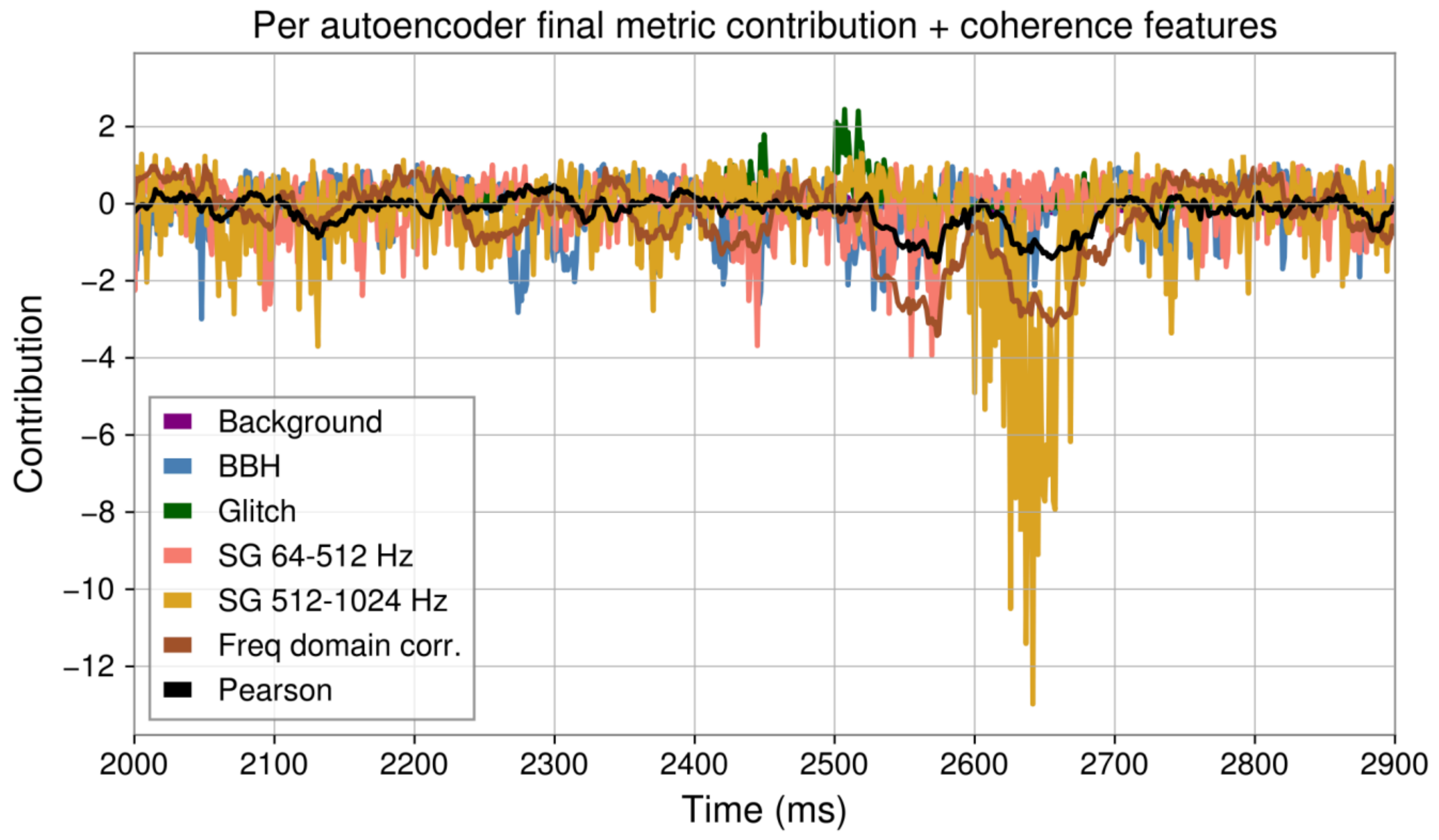
F. Capel et al.

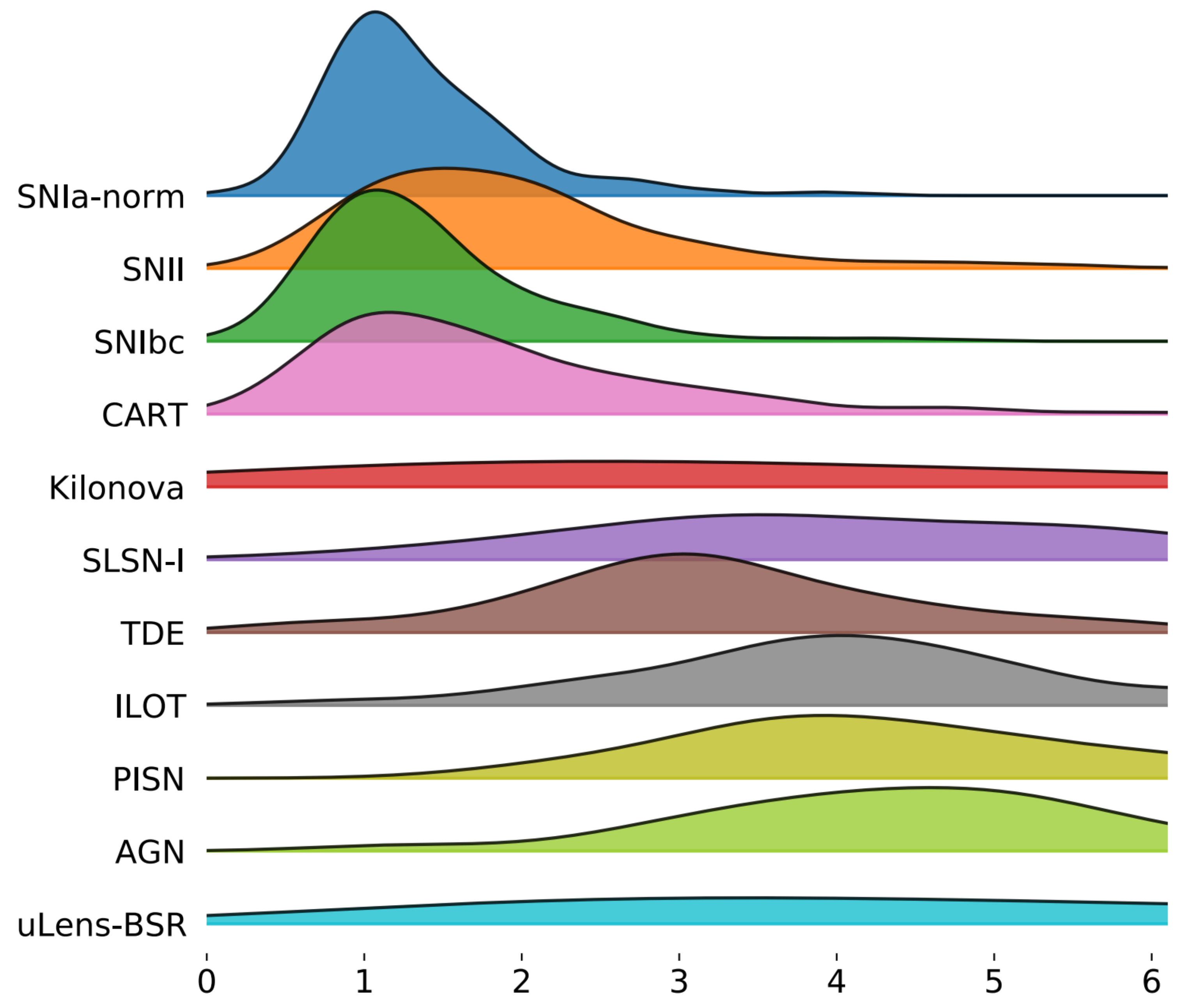


Gravitational wave

Multi-messenger astronomy

Real-time Multi-Messenger Anomaly Detection





https://github.com/ML4GW

https://arxiv.org/abs/2309.11537

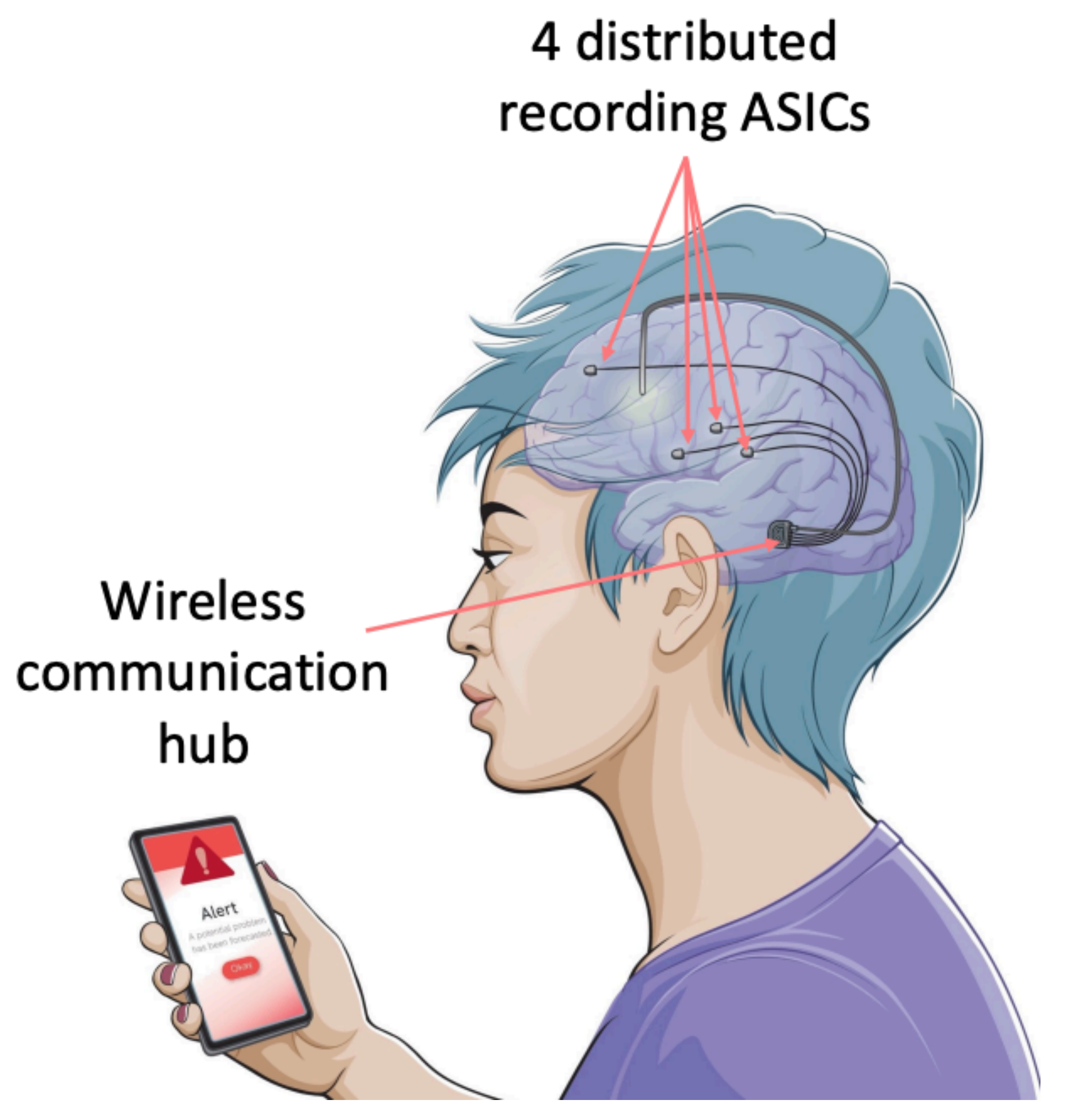
Accelerating Physics With AI

...and outside of particle physics

Semantic segmentation for autonomous vehicles

N. Ghielmetti et al.

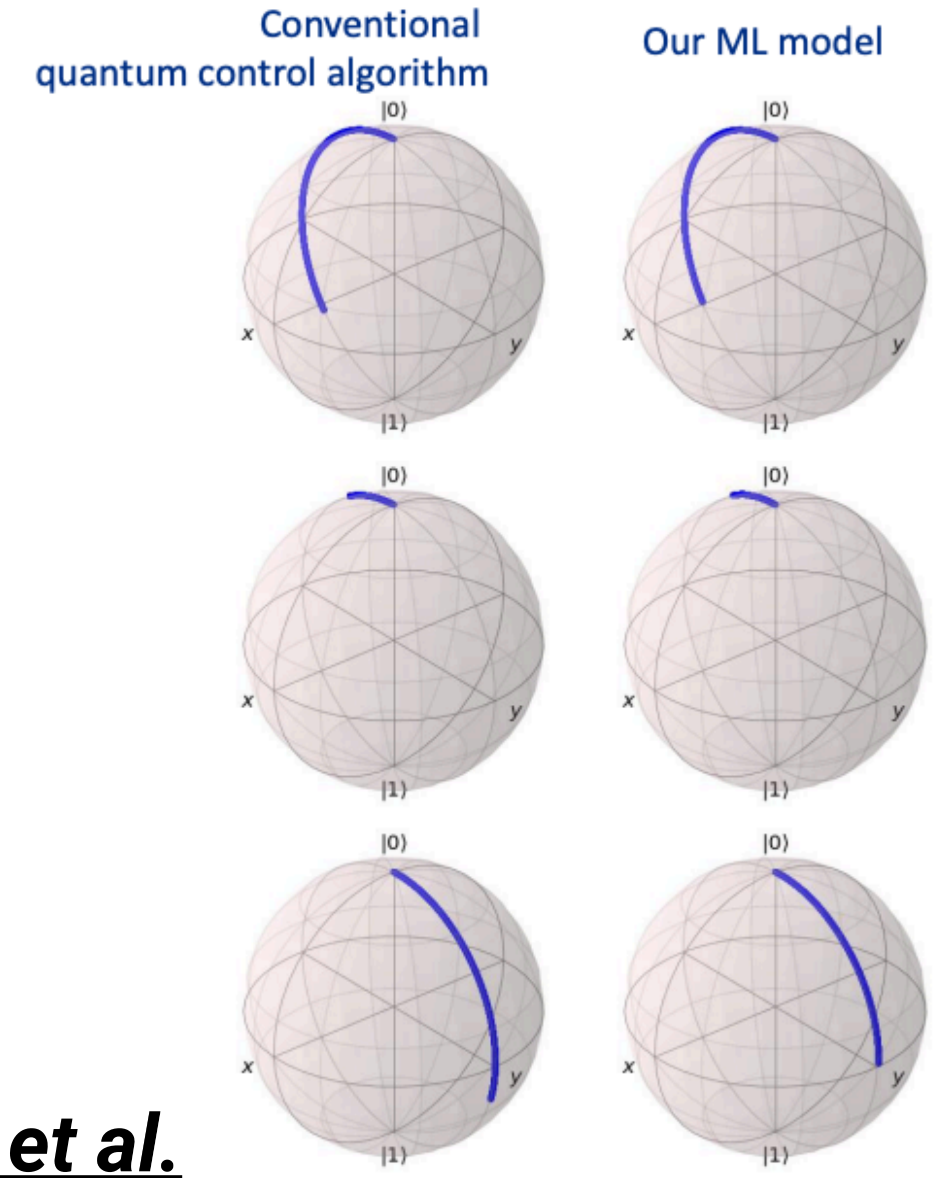
Seizure Predicting Brain Implant



W. Lemaire et al.

NN accelerator for quantum control

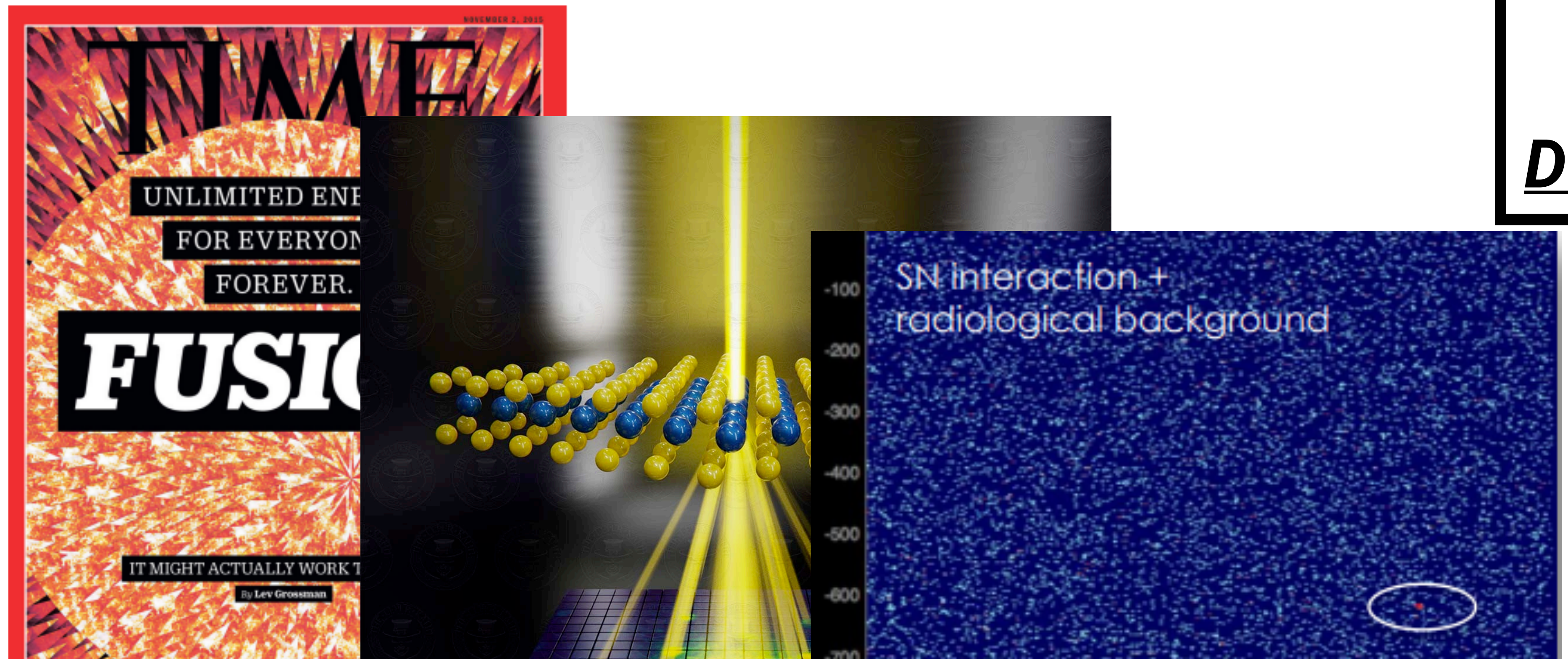
- Putting control in cryostat (e.g optimal pulse parameters)



D Xu et al.

Other examples

- ***For fusion science phase/mode monitoring***
- ***Crystal structure detection***
- ***Triggering in DUNE***
- ***Accelerator control***
- ***Magnet Quench Detection***
- ***MLPerf tinyML benchmarking***
- ***Food contamination detection***
- etc....



Accelerated AI Algorithms for Data-Driven Discovery

Fast Machine Learning Lab

Hardware and Algorithm Co-development

Developing AI methods to encode non-lattice-structured data is one main challenge in current AI systems.

Read More >

High Energy Physics

Build tools to process LHC collisions occurring 40 million times per second data in real-time using AI.

Read More >

Systems Neuroscience

Discover the computations that brain-wide neural networks perform to process sensory and motor information during behavior by using high-throughput and low-latency AI algorithms to process.

Read More >

Multi-messenger Astrophysics

Process the data from telescopes, neutrino detectors, and gravitational-wave detectors to identify astronomical events corresponding to the most violent phenomena in the Cosmos.

Read More >

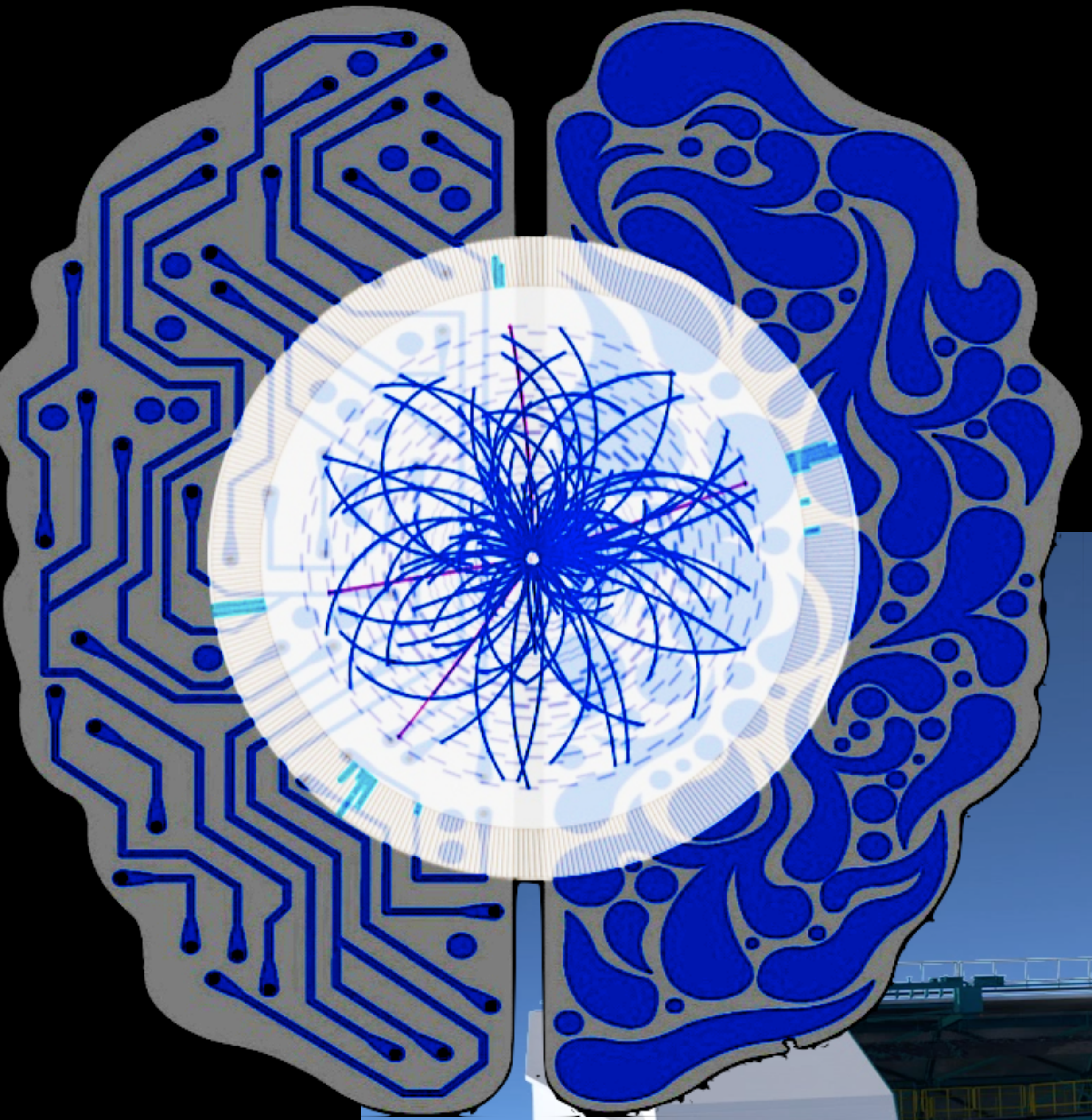
FastML Lab

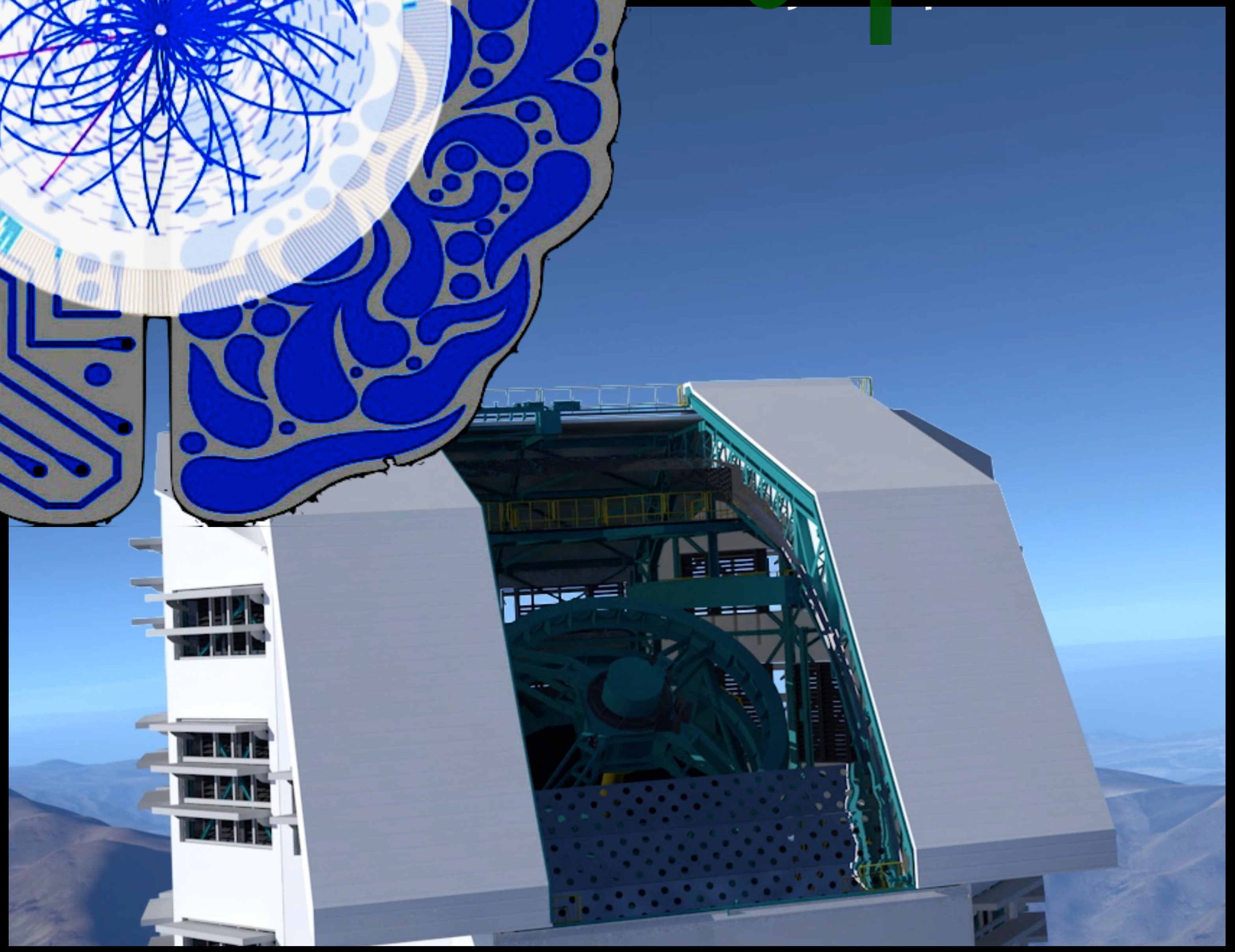
Real-time and accelerated ML for fundamental sciences

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.

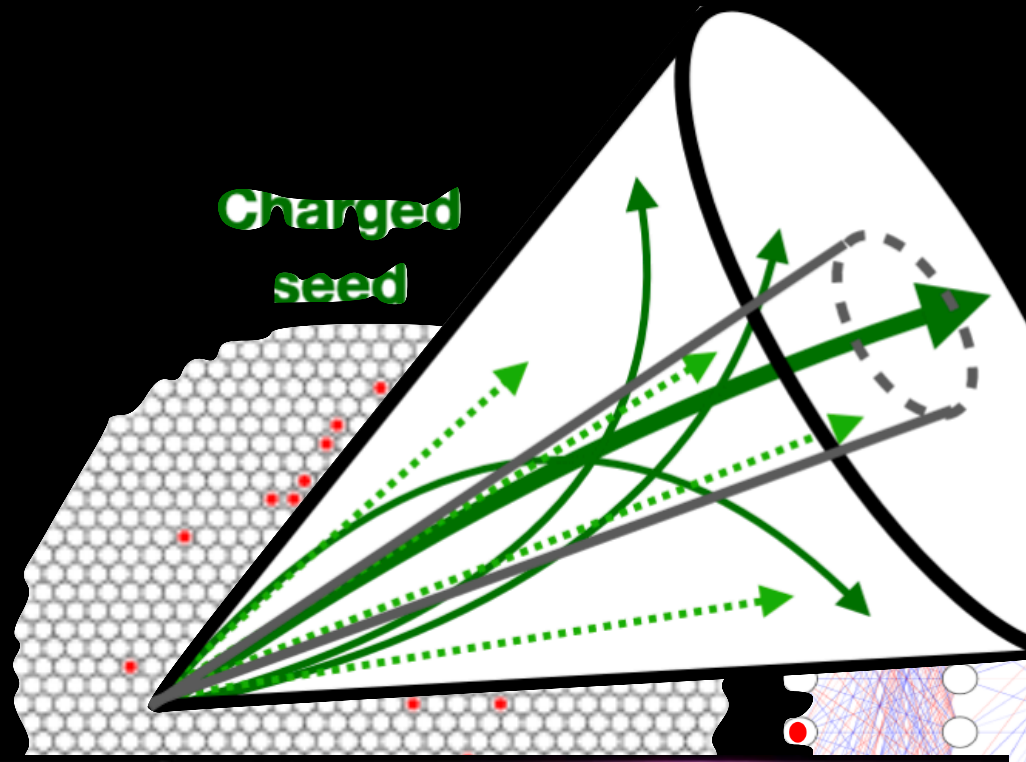
https://a3d3.ai/

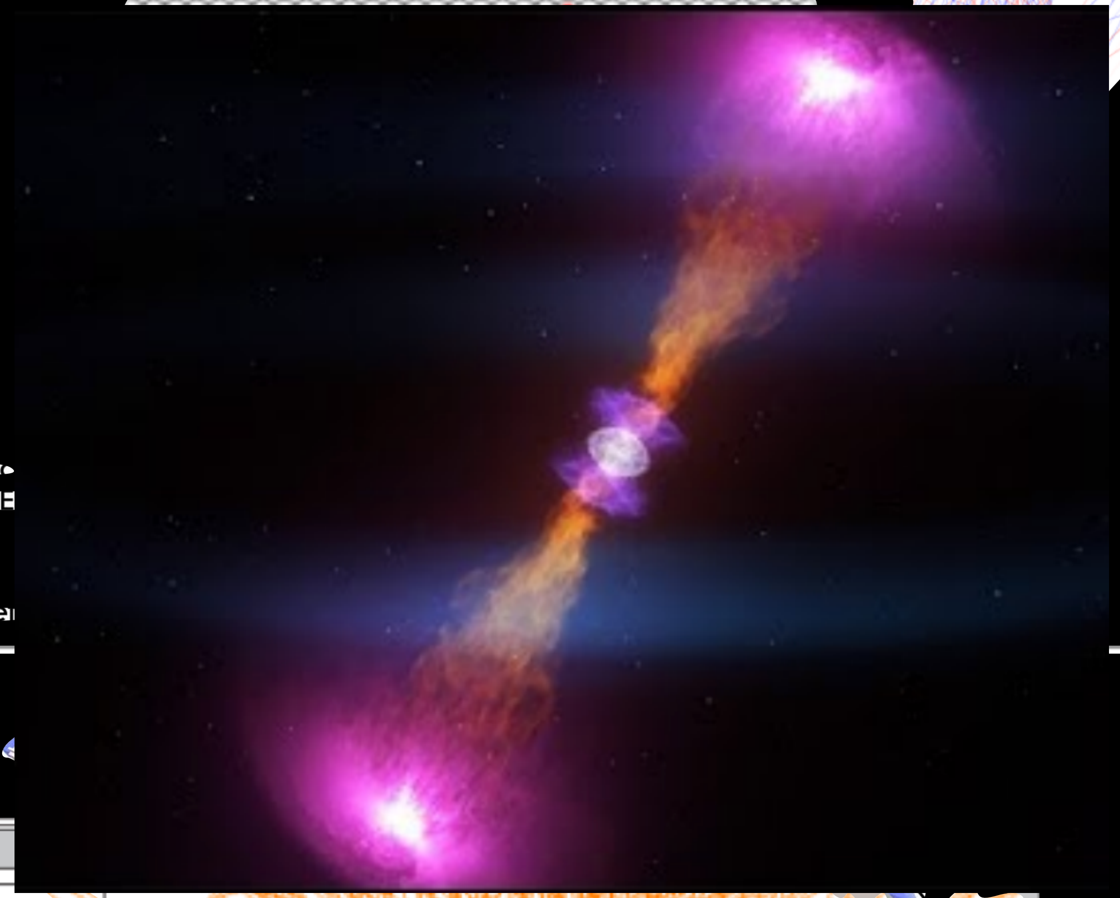
https://fastmachinelearning.org/

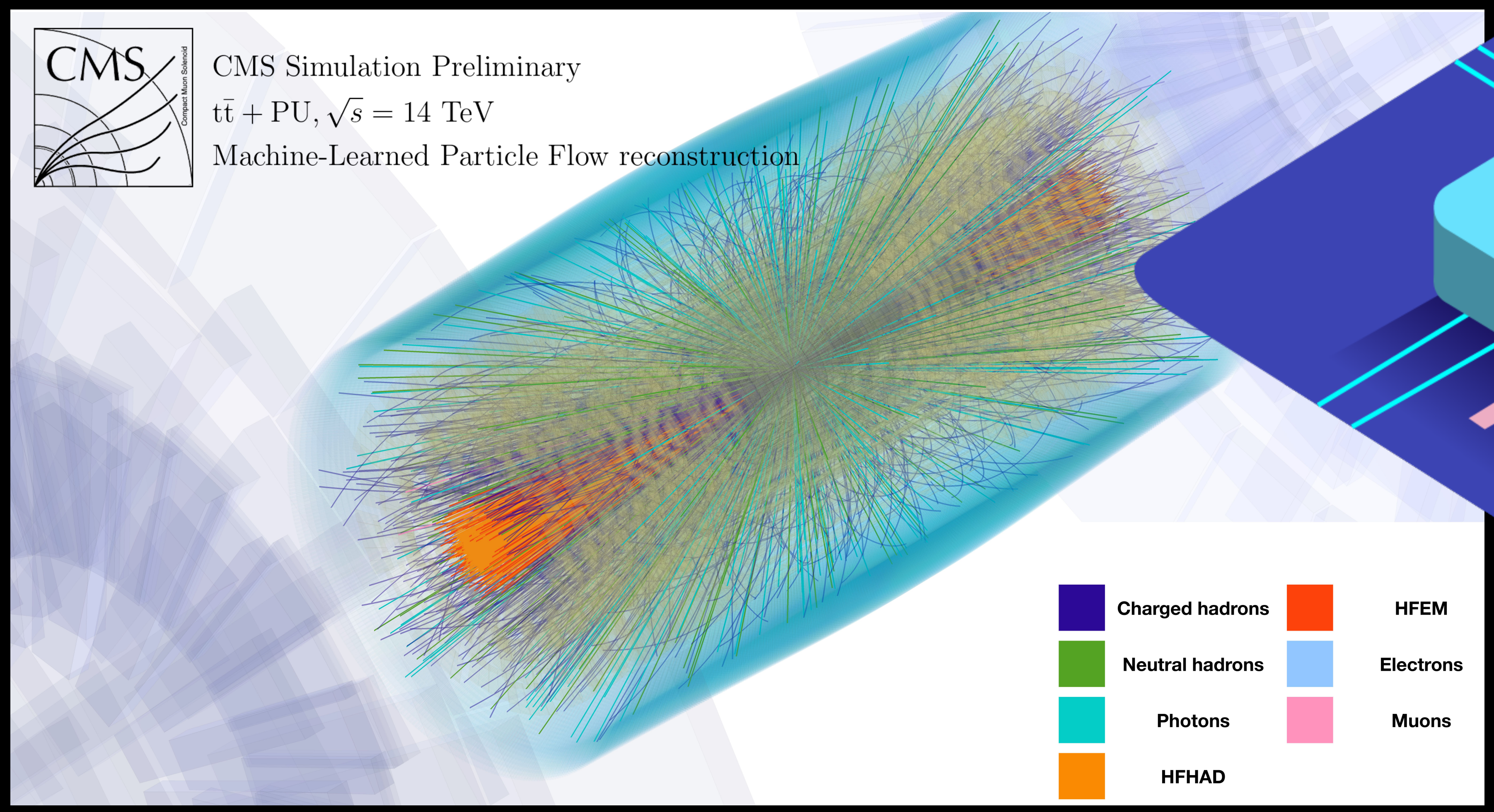


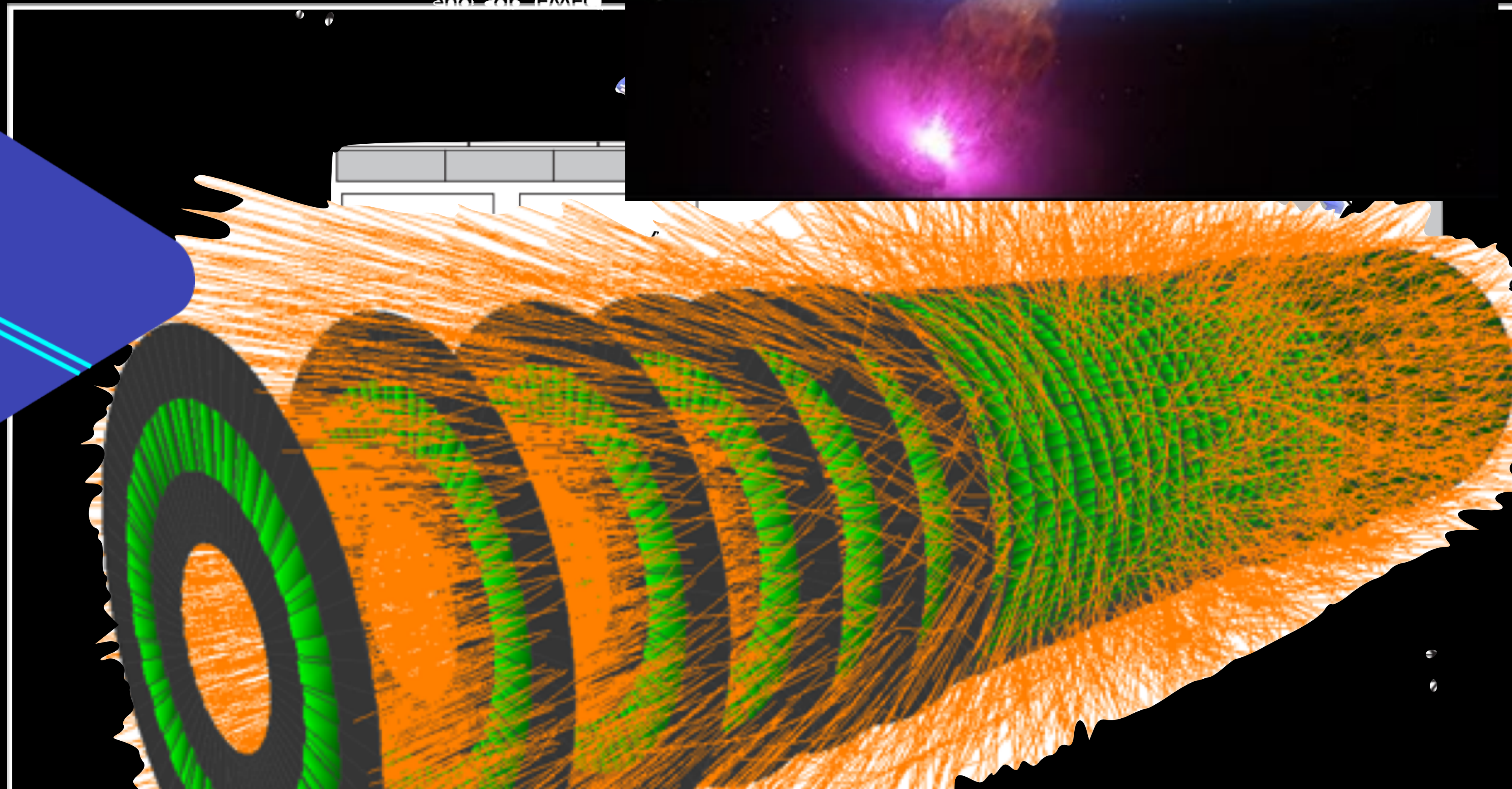


Join us at the FastML Lab!









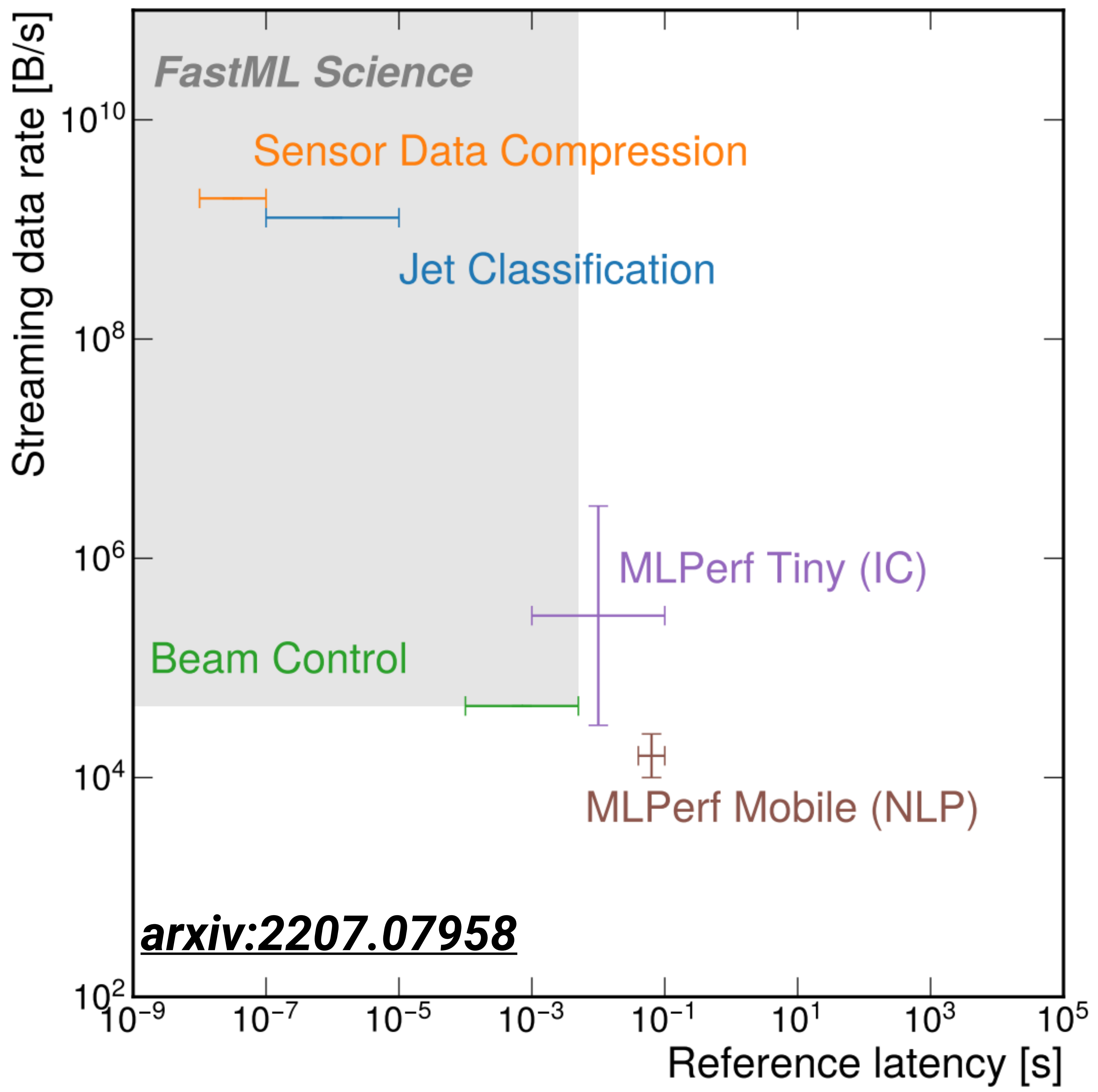
Backup

Benchmarking

Datasets: Common FastML Science Benchmarking datasets
- guide design of edge ML hardware and software for sub-microsecond inference!

Algorithms: hls4ml-FINN benchmarked in MLPerf™
- how fast systems can process inputs and produce results
- efficient and low-latency FPGA solutions with hls4ml and FINN

Consistently competitive (QKeras+hls4ml, semantic segmentation, MLPerf)



https://mlcommons.org/en/inference-tiny-07/

Model	LUT		LUTRAM		FF		BRAM [36 kb]		DSP		Latency [ms]	Energy/inf. [μJ]
Pynq-Z2												
Available	53 200		17 400		106 400		140		220		–	–
IC (hls4ml)	28 544	53.7%	3 756	21.6%	49 215	46.3%	42	30.0%	4	1.8%	27.3	44 330
IC (FINN)	24 502	46.1%	2 086	12.0%	34 354	32.3%	100	71.4%	0	0.0%	1.5	2 535
AD	40 658	76.4%	3 659	21.0%	51 879	48.8%	14.5	10.4%	205	93.2%	0.019	30.1
KWS	33 732	63.4%	1 033	5.9%	34 405	32.3%	37	26.4%	1	0.5%	0.017	30.9

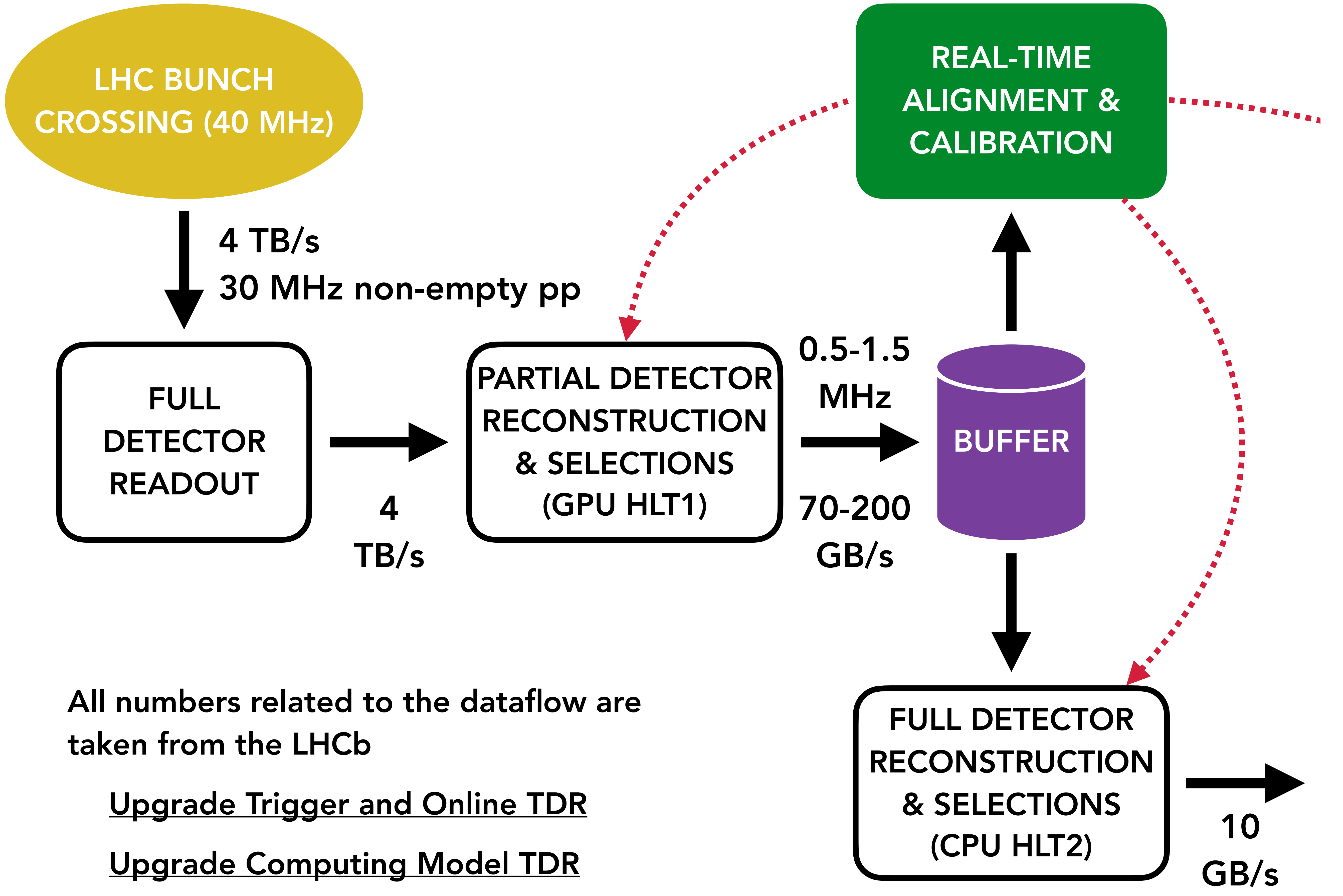
arxiv:2103.05579

LHCb GPU trigger

Full GPU reconstruction @ 4% internet traffic

- 326 GPUs, 60 kHz per GPU

Characteristics of LHCb HLT1	Characteristics of GPUs
Intrinsically parallel problem: - Run events in parallel - Reconstruct tracks in parallel	Good for - Data-intensive parallelizable applications - High throughput applications
Huge compute load	Many TFLOPS
Full data stream from all detectors is read out → no stringent latency requirements	Higher latency than CPUs, not as predictable as FPGAs
Small raw event data (~100 kB)	Connection via PCIe → limited I/O bandwidth
Small event raw data (~100 kB)	Thousands of events fit into O(10) GB of memory



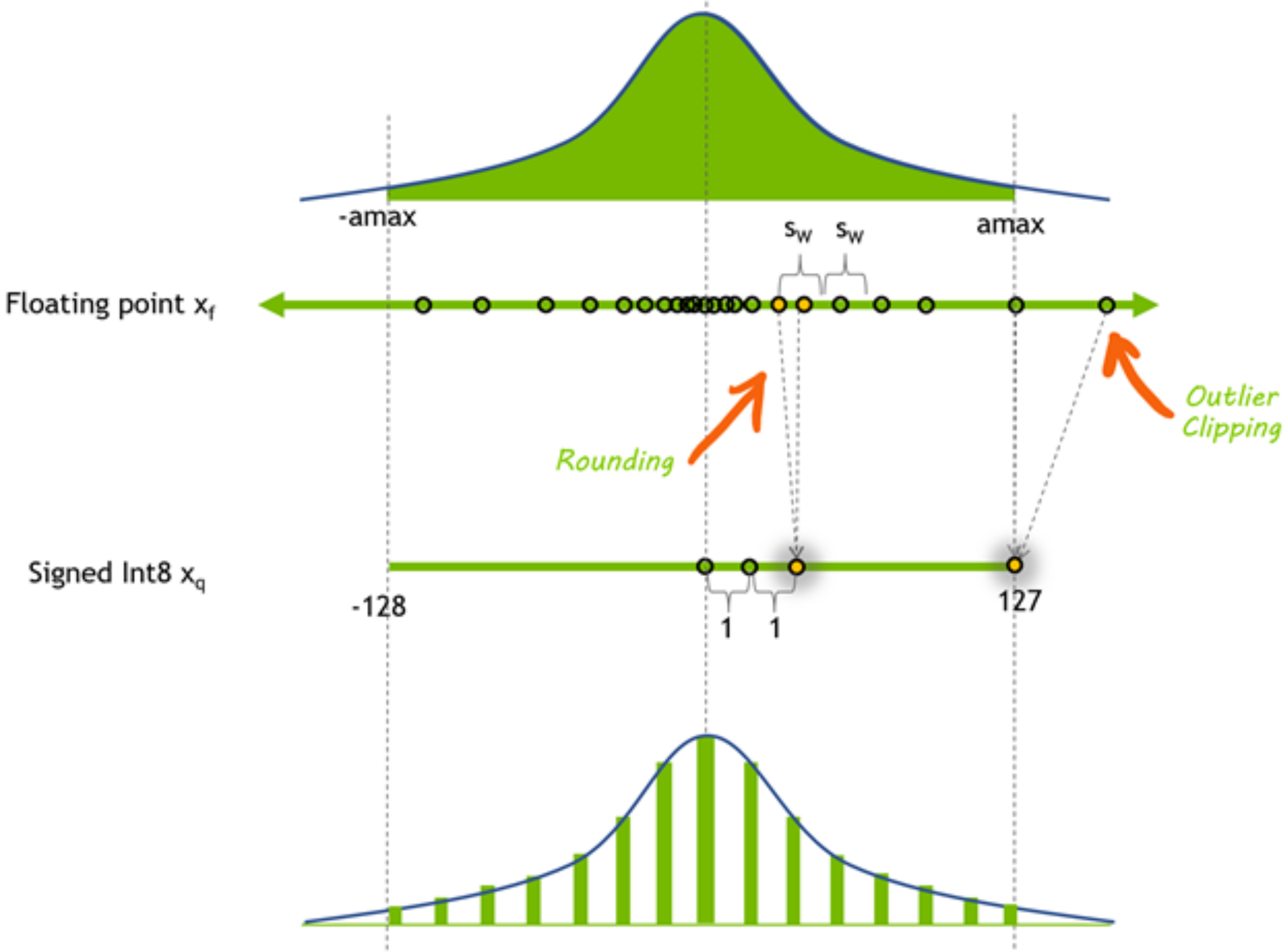
All numbers related to the dataflow are taken from the LHCb

Upgrade Trigger and Online TDR

Upgrade Computing Model TDR

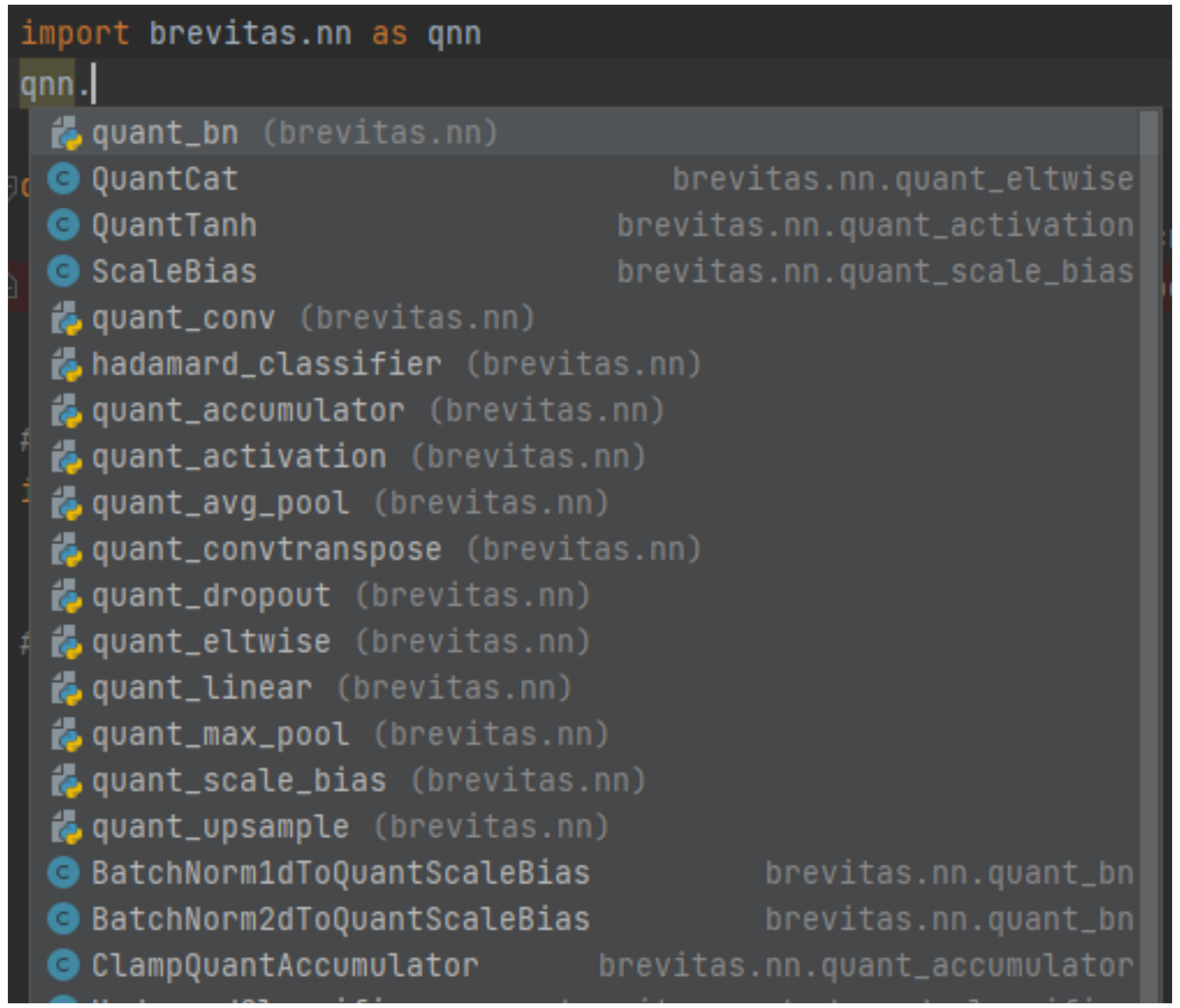
C. Fitzpatrick

https://developer.nvidia.com/blog/achieving-fp32-accuracy-for-int8-inference-using-quantization-aware-training-with-tensorrt/



https://github.com/Xilinx/brevitas

QPYTORCH ?

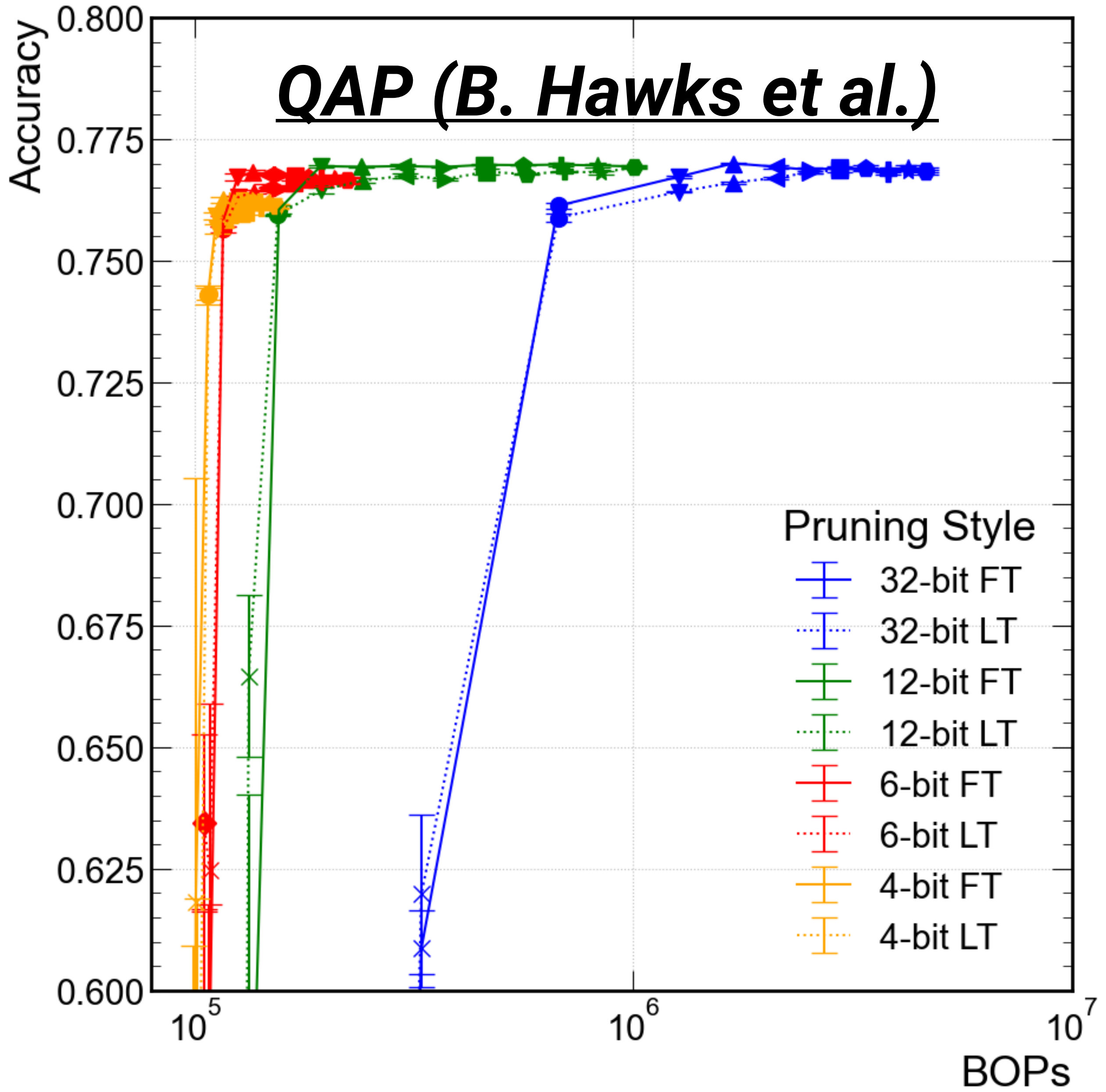


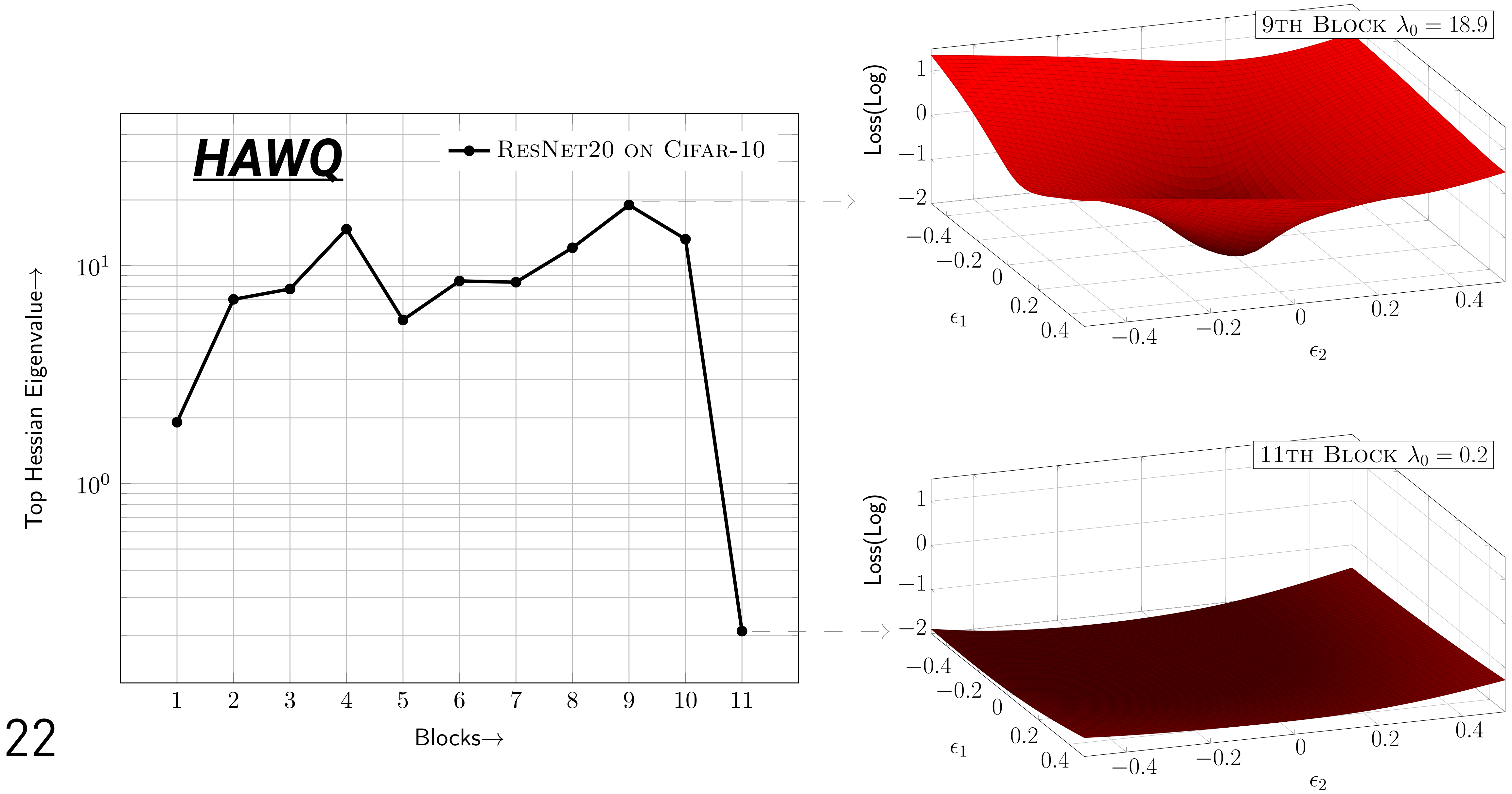
Brevitas like QKeras, but for PyTorch

- QAT library
- Support most common layers and activation functions

Other quantization techniques:

- ***HAWQ: Hessian AWare Quantization***
- ***Quantization Aware Pruning (B. Hawks et al.)***



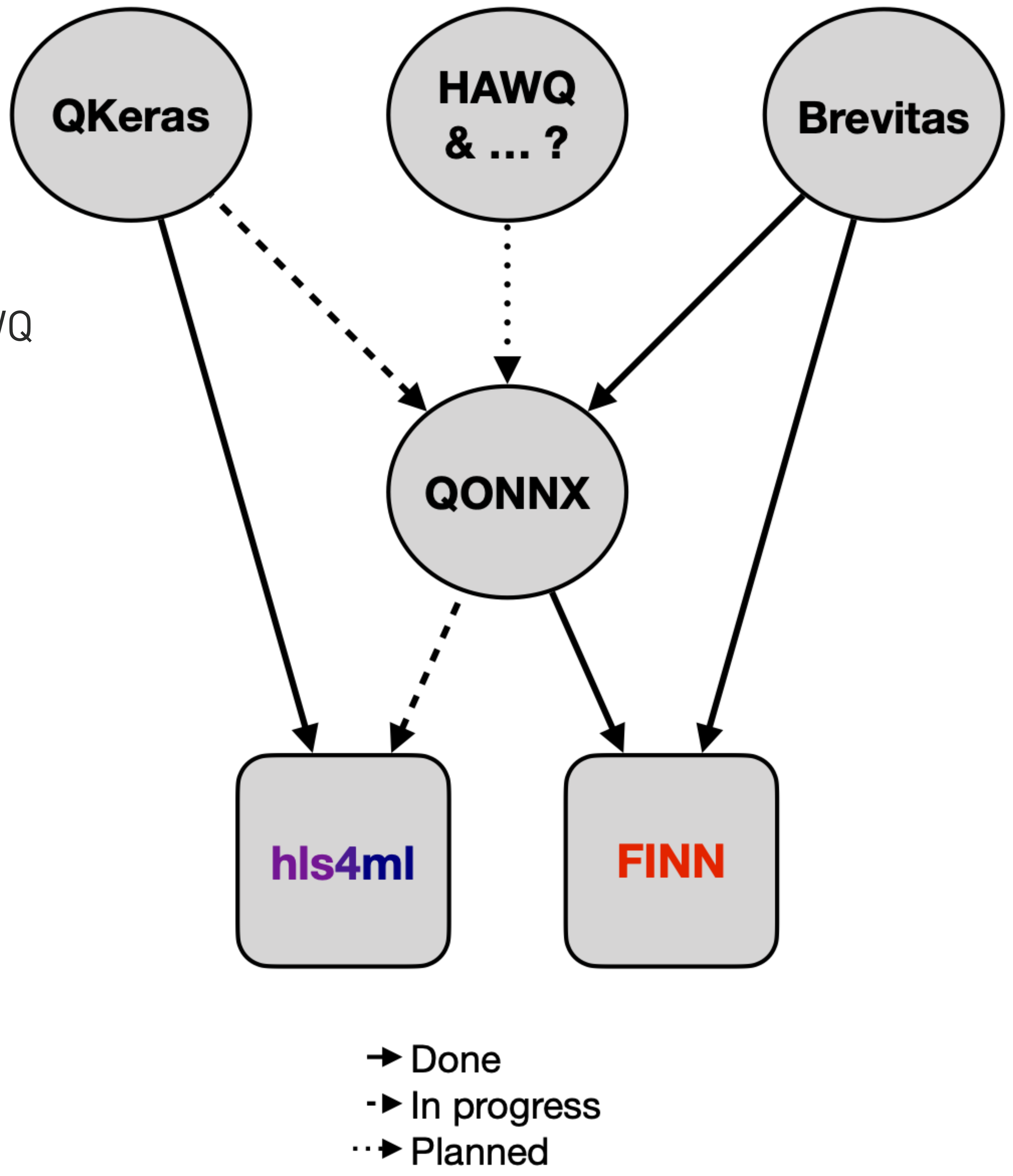


QPYTORCH ?

hls4ml collaborate with Xilinx Research Labs to develop QOONX

- Introducing 'Quant' node to ONNX graph
- Brevitas (PyTorch) and QKeras (Keras) can export QONNX (HAWQ export in progress): then hls4ml and FINN can import QONNX

Quantized ONNX (QONNX), J. Mitrevski et. al

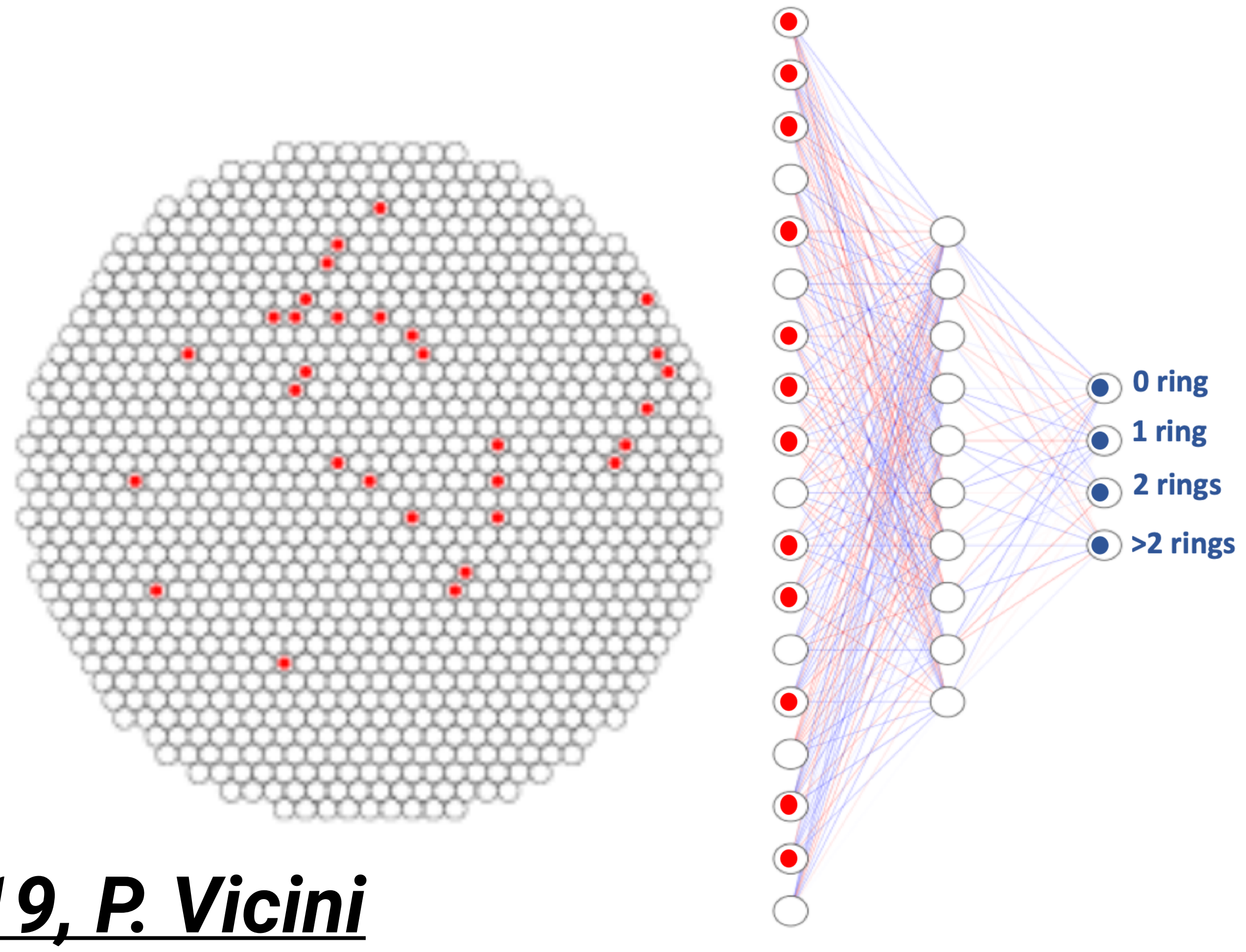


Co-Design for Efficient & Adaptive ML	*Yaman Umuroglu*
Blackett Laboratory, Lecture Theatre 1, Imperial College London	09:00 - 09:45

hls4ml in other CERN experiments

NA62: Measuring BR($K^+ \rightarrow \pi^+ \nu \bar{\nu}$) = O($10^{-11}$)

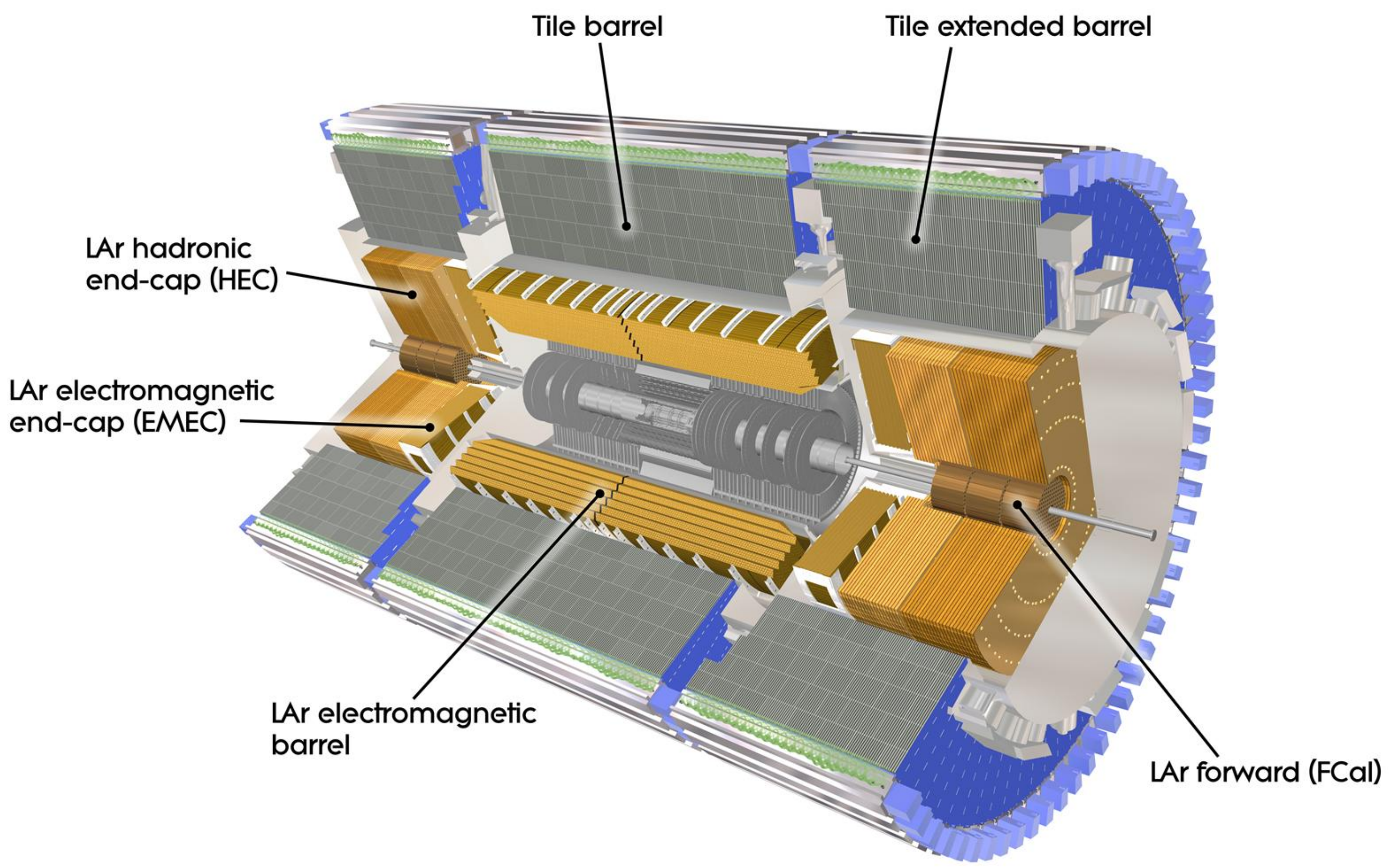
- FPGA trigger 800 MHz→1 MHz



CHEP 2019, P. Vicini

ATLAS Liquid Argon Calorimeter (R&D)

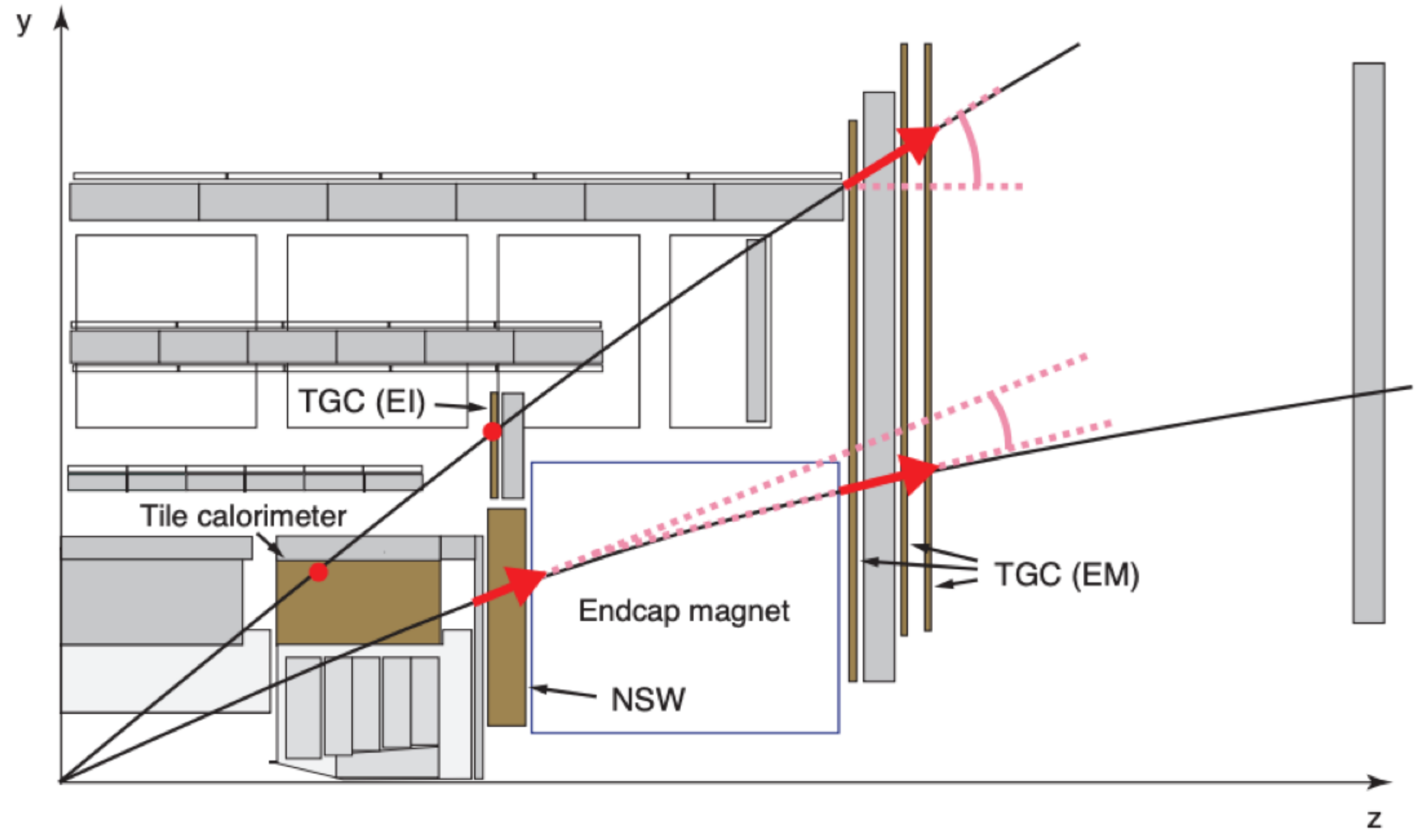
- RNN for real-time energy reconstruction
- ~200 ns on Intel Stratix-10 FPGA



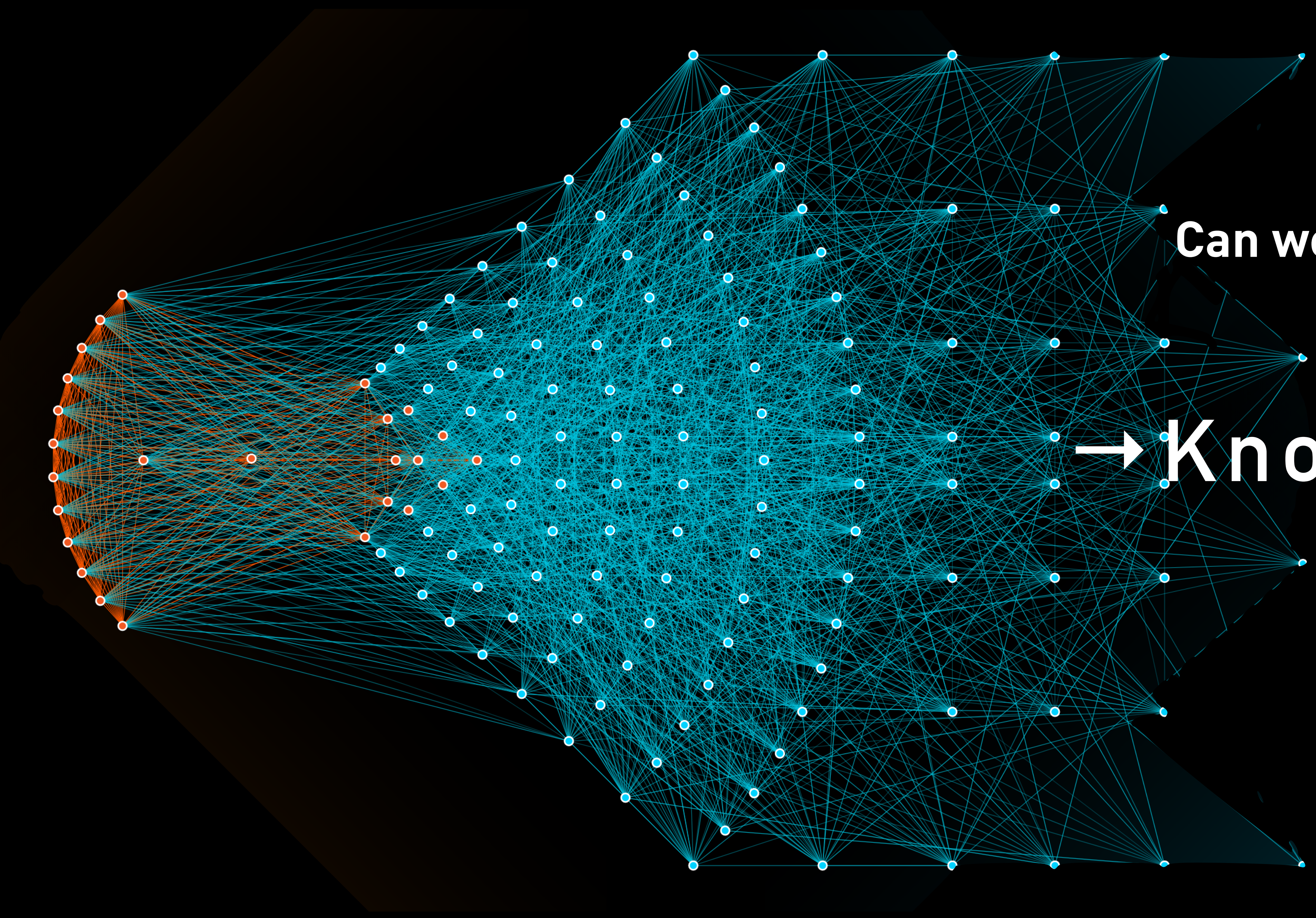
DOI:10.1007/s41781-021-00066-y

ATLAS small wheel muon segment finding and reconstruction (R&D)

- Regression of muon position and angle
- 400 ns budget

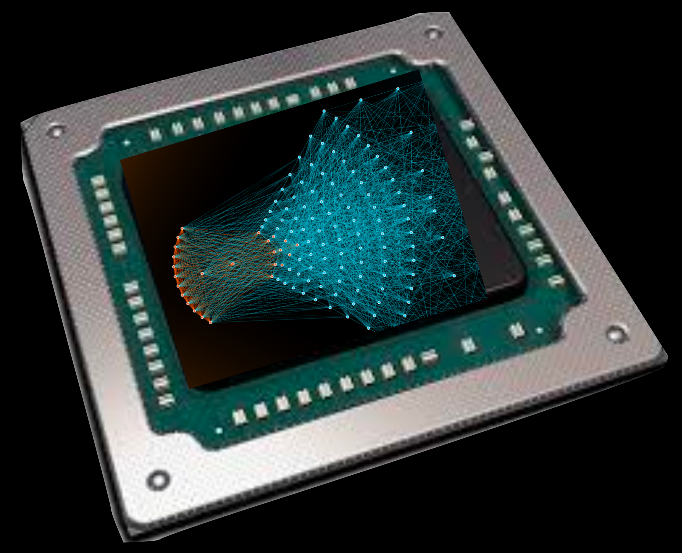


R. Teixeira de Lima, R Rojas Caballero et al.



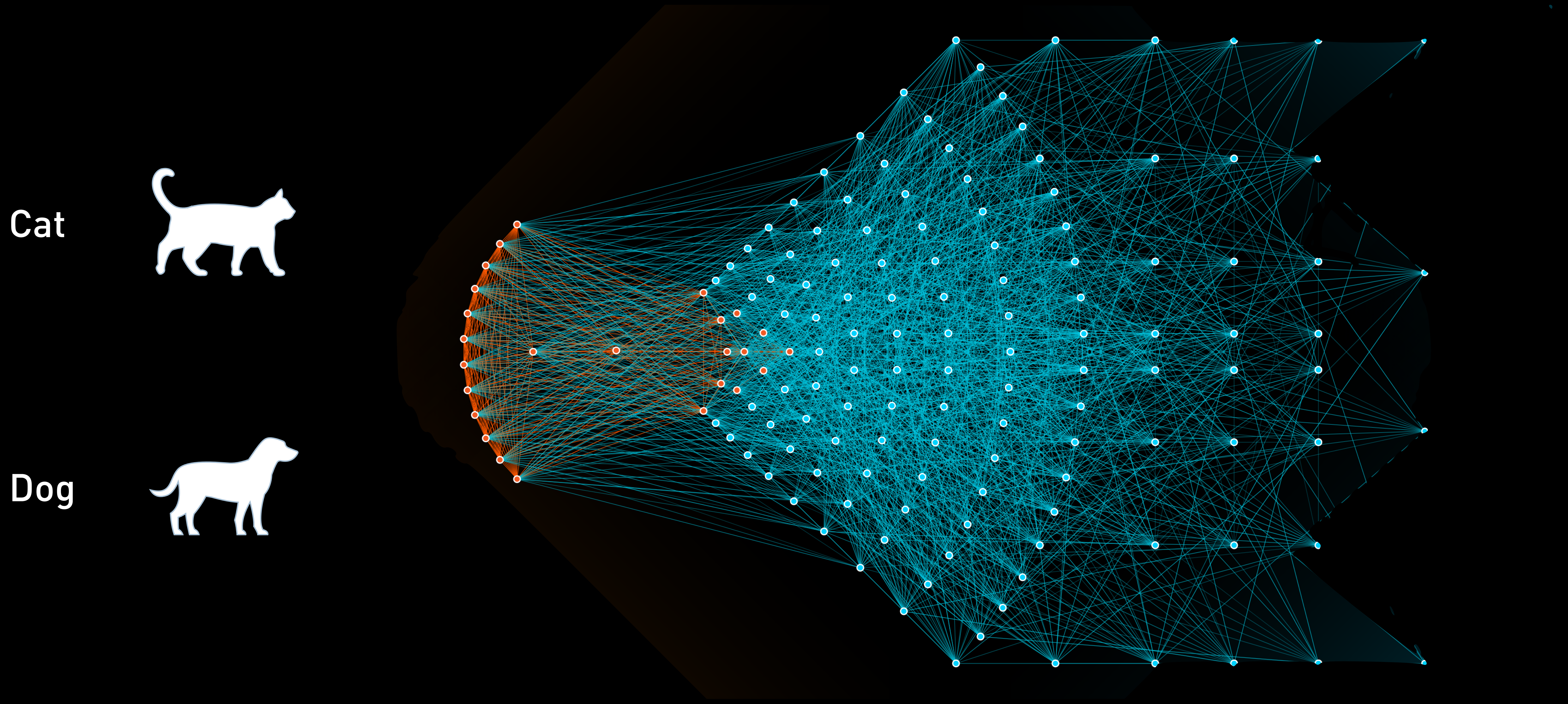
Can we have the best of both worlds?

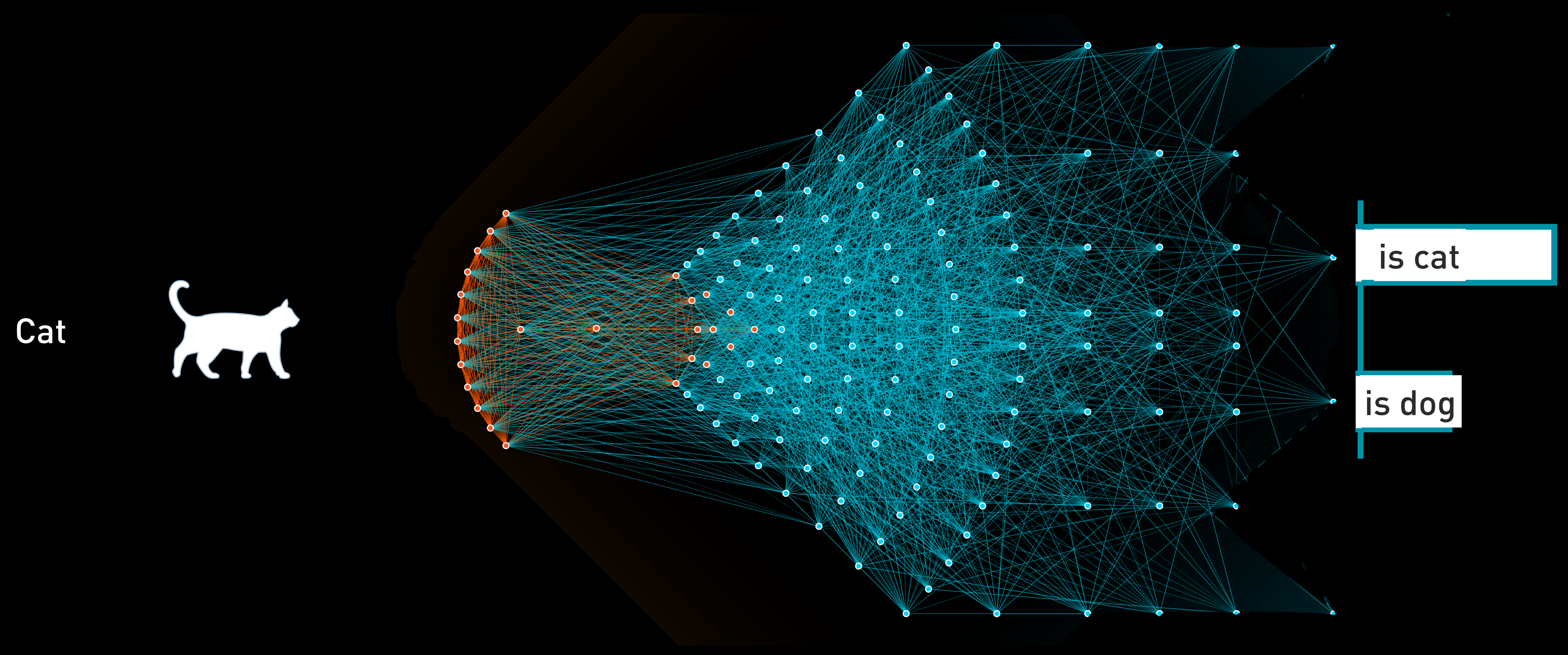
→Knowledge Distillation



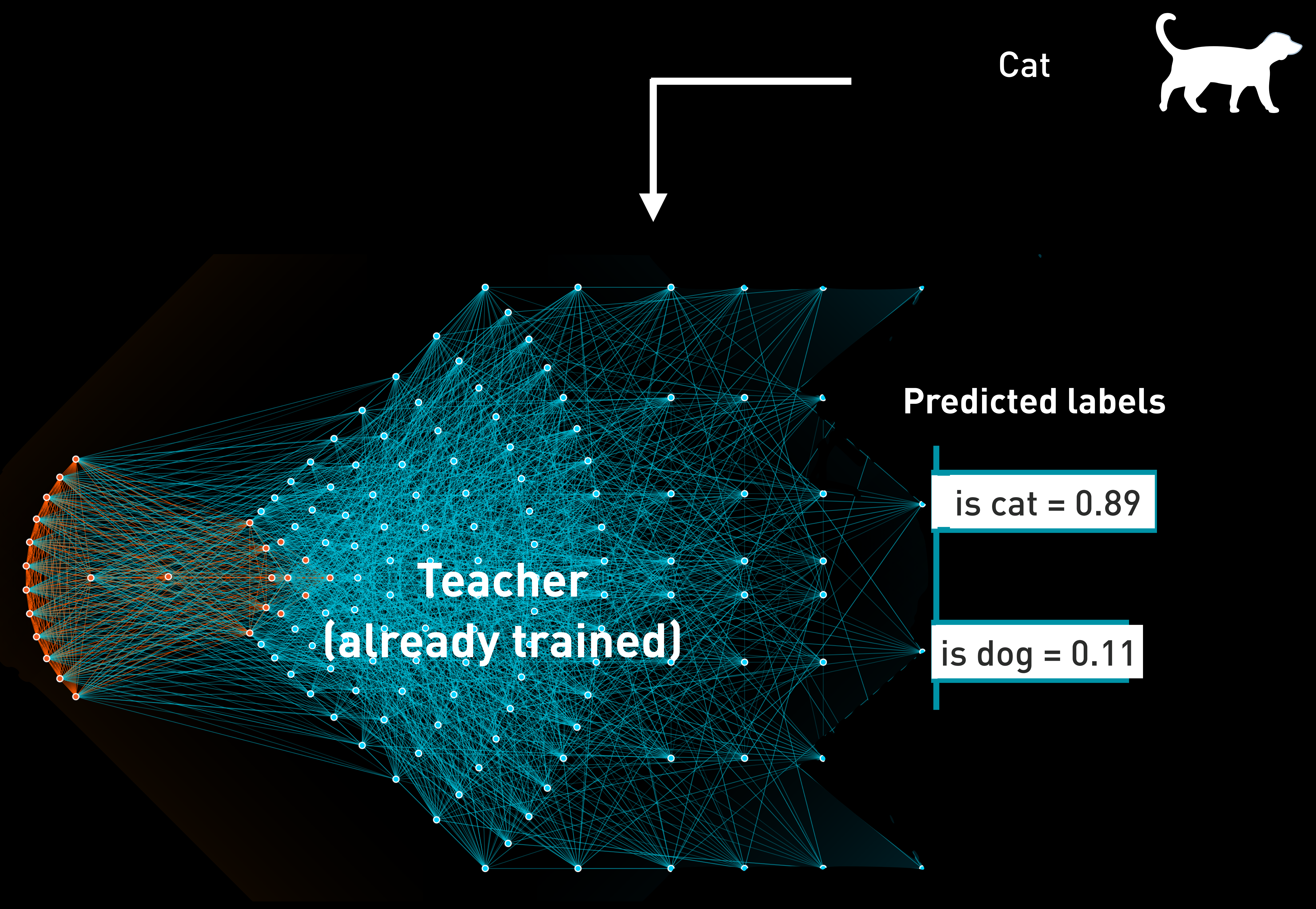
Train

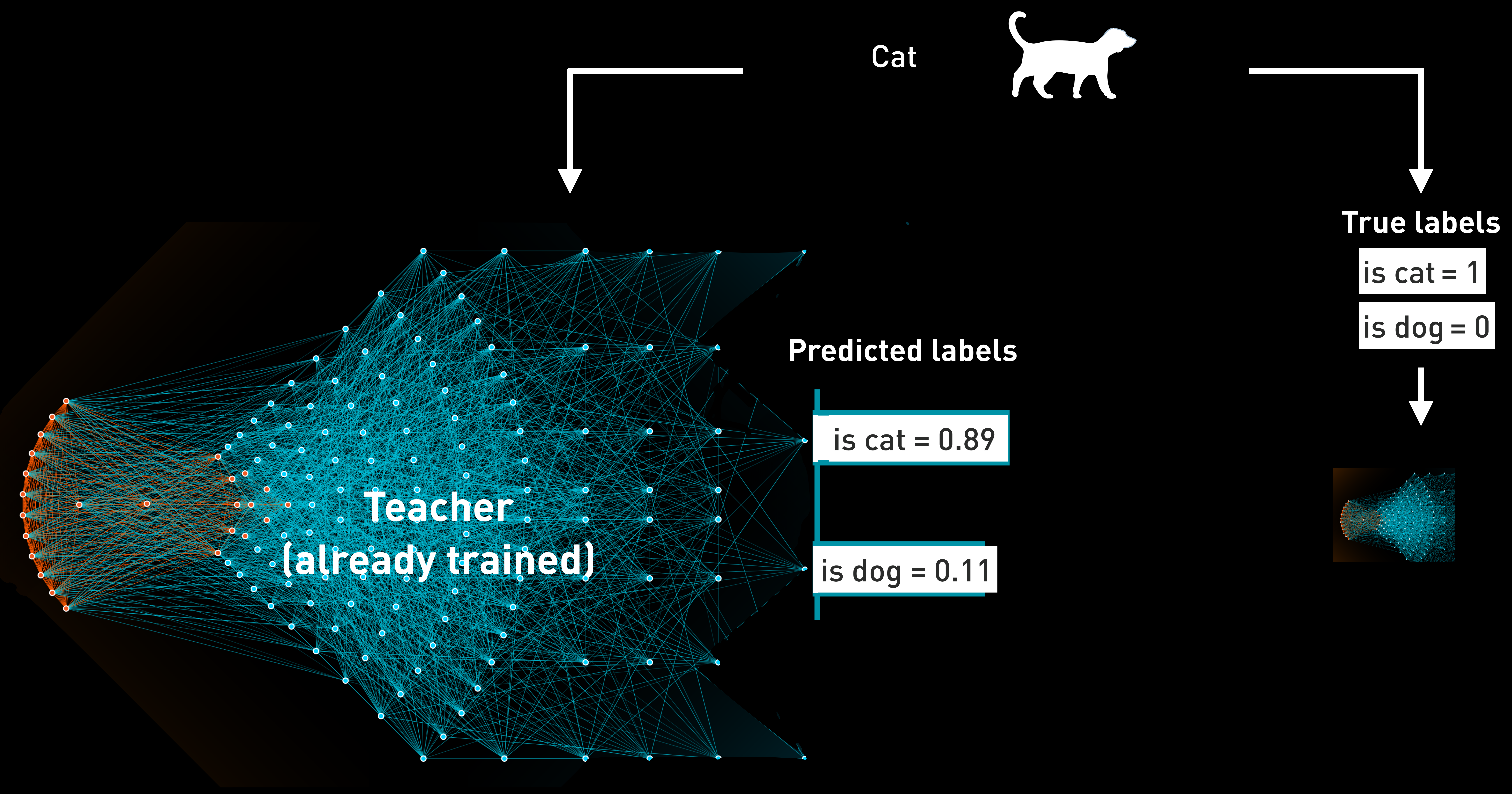
Inference

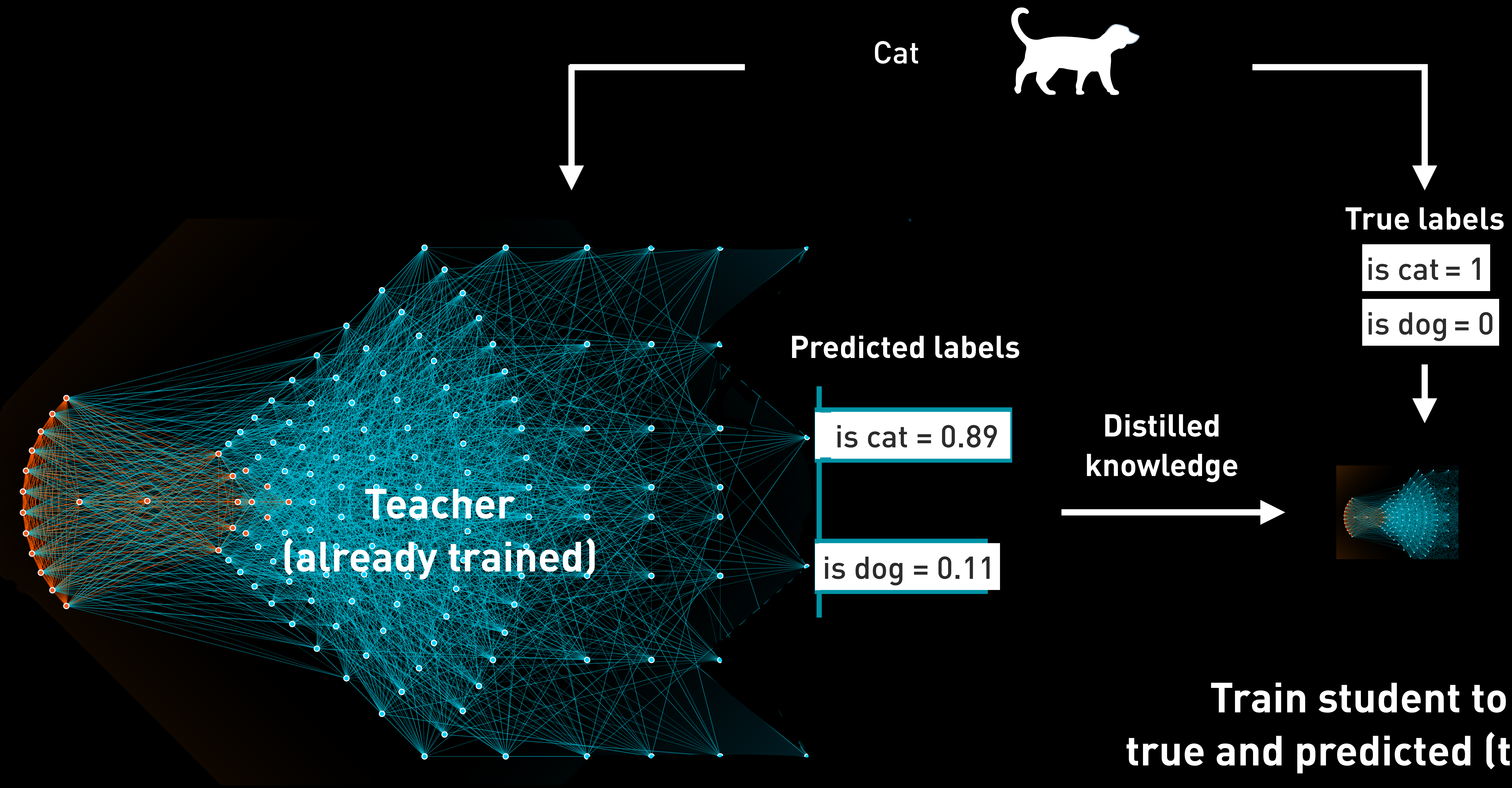

Cat
Dog



Cat
is cat
is dog


Cat
Teacher
(already trained)
Predicted labels
is cat = 0.89
is dog = 0.11


Cat
True labels
is cat = 1
is dog = 0
Predicted labels
is cat = 0.89
is dog = 0.11
Teacher
(already trained)

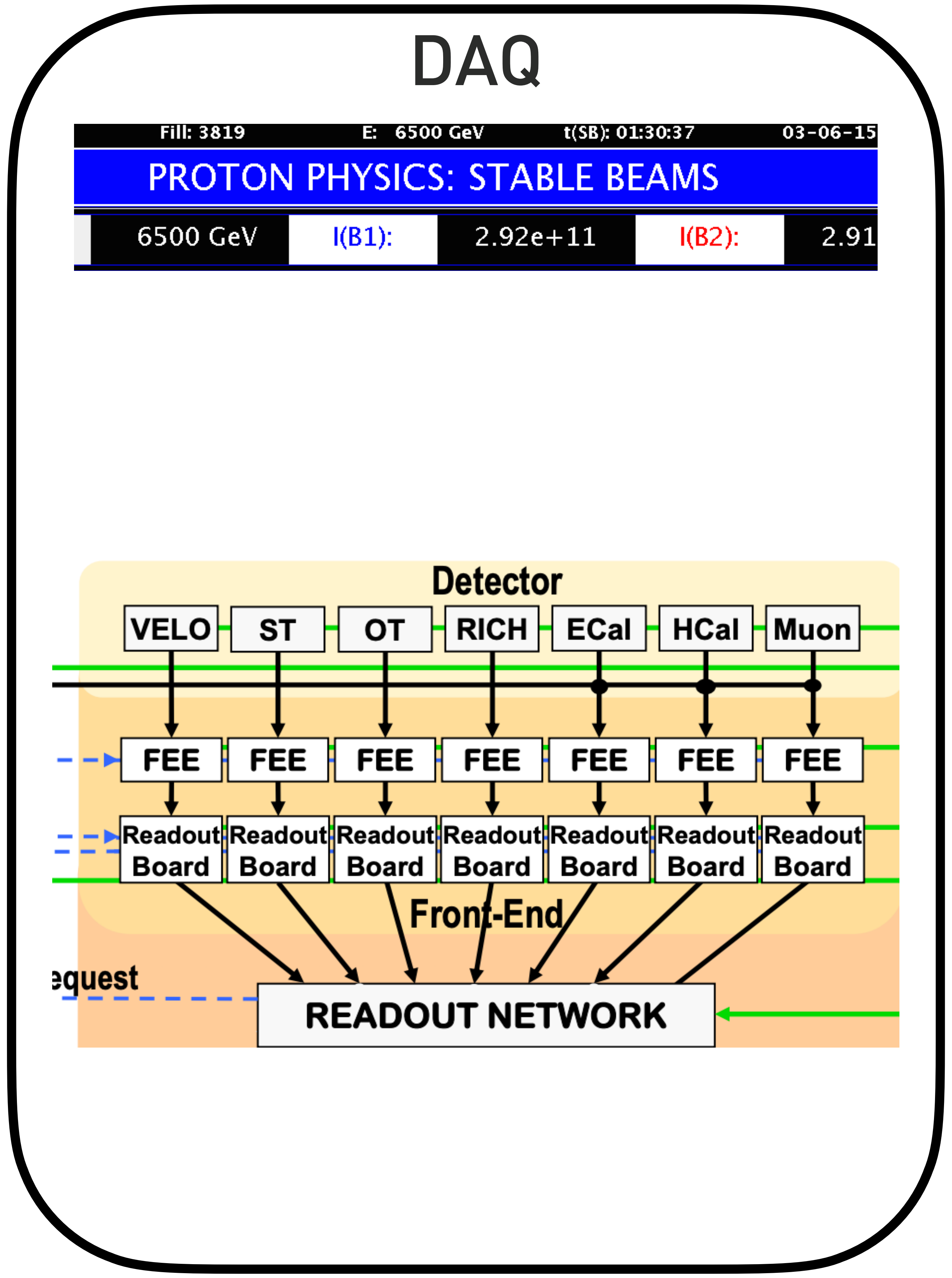


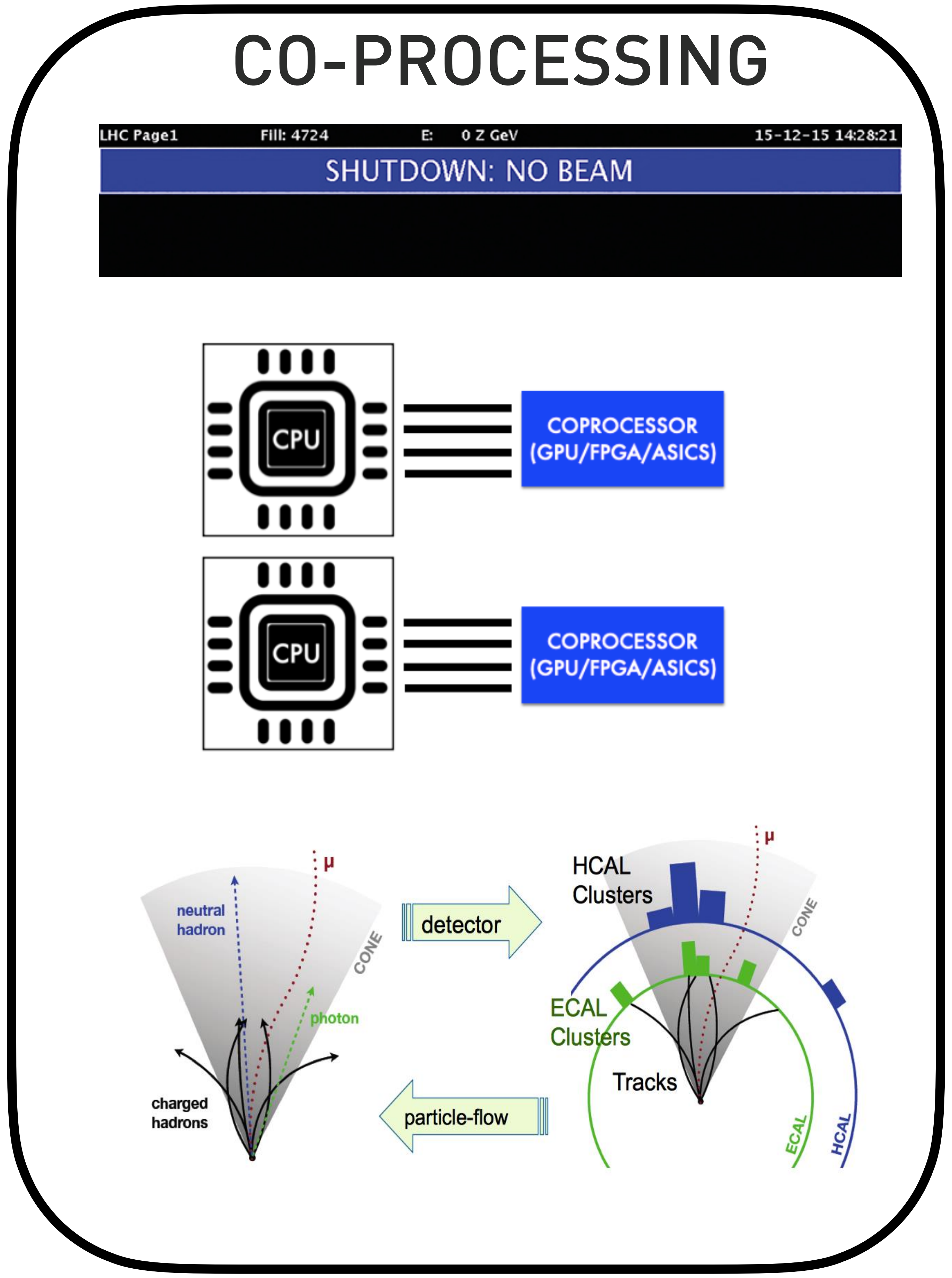
Train student to learn both true and predicted (teacher) labels!

$$L_{total} = \beta \times L_{Distillation} + \alpha \times L_{student}$$

Student learns subtle learned features from teacher!

FPGAs as accelerators

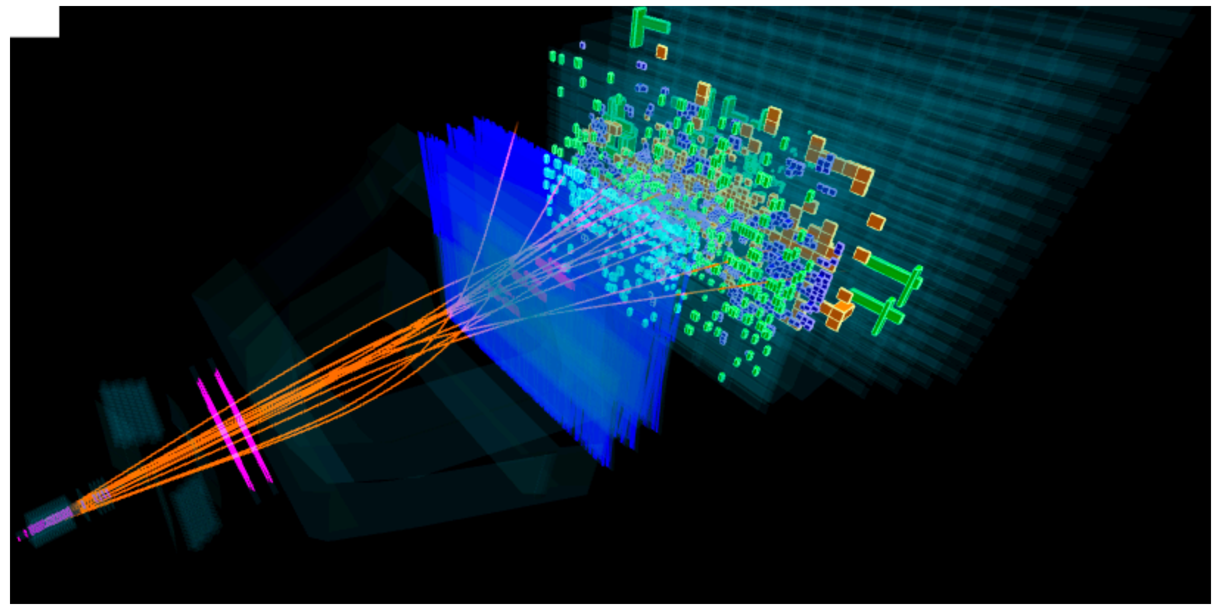




C. Beteta, I. Bezshyiko, N. Serra

High throughput GPU triggers

40 MHz pp collisions

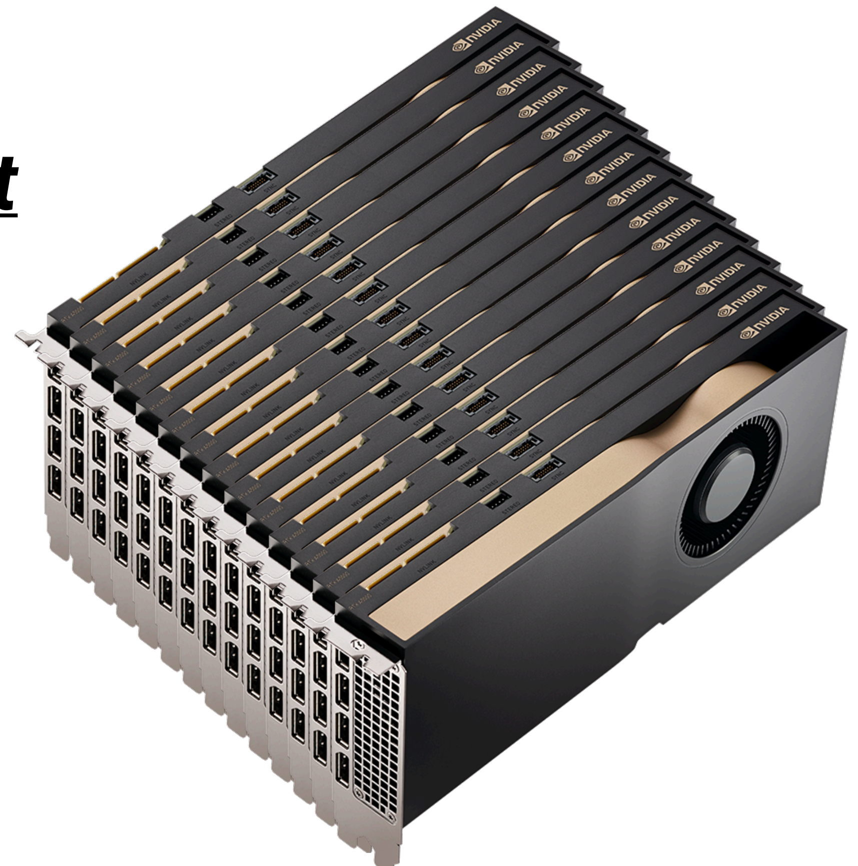


4 TB/s
30 MHz non-empty pp

Detector read-out
O(500) FPGAs

4
TB/s

350 GPUs
5 TB/s @ 30 MHz
of raw data



0.5-1.5
MHz

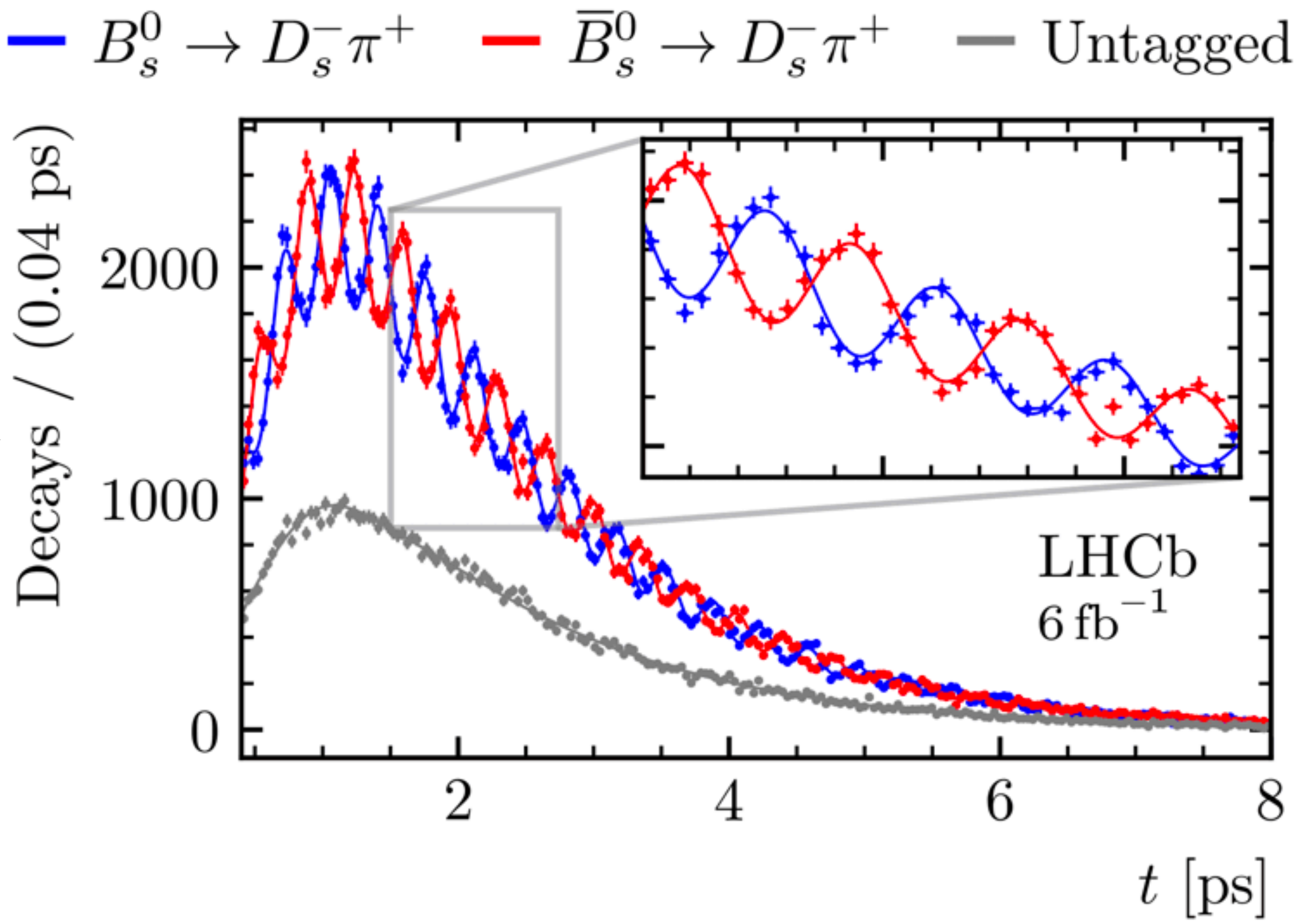
70-200
GB/s

O(10) PB
buffer

REAL-TIME
ALIGNMENT &
CALIBRATION

CPU-based
(full reco &
trigger)

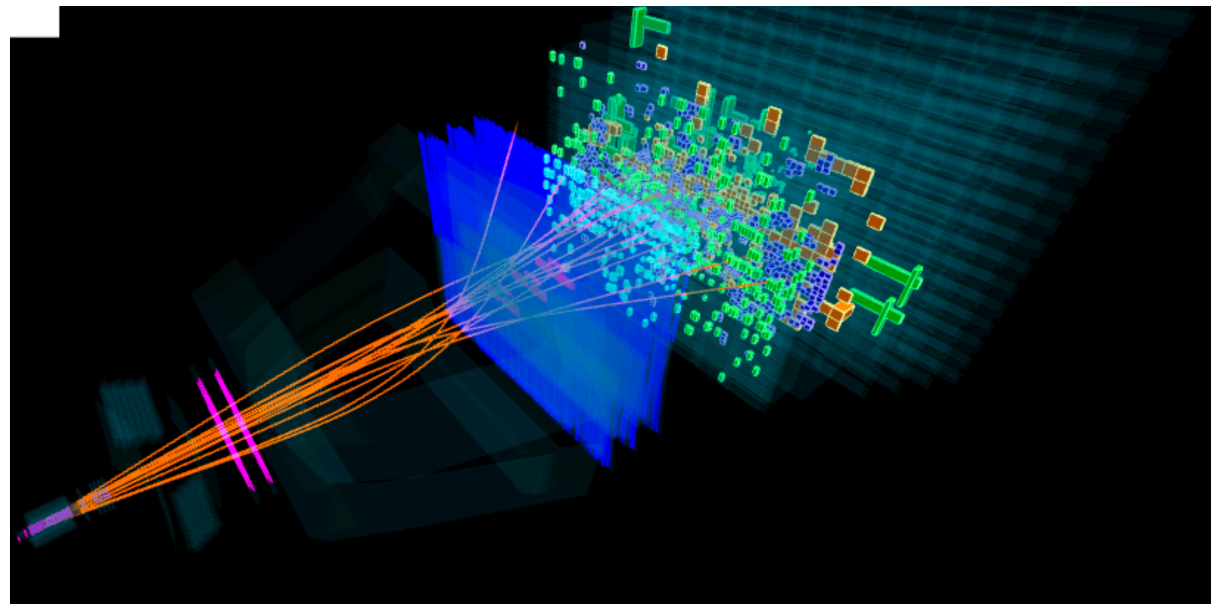
10
GB/s



ML already used in LHCb GPU trigger for
particle ID,
track reconstruction,
trigger decisions … more underway!

High throughput GPU triggers

40 MHz pp collisions

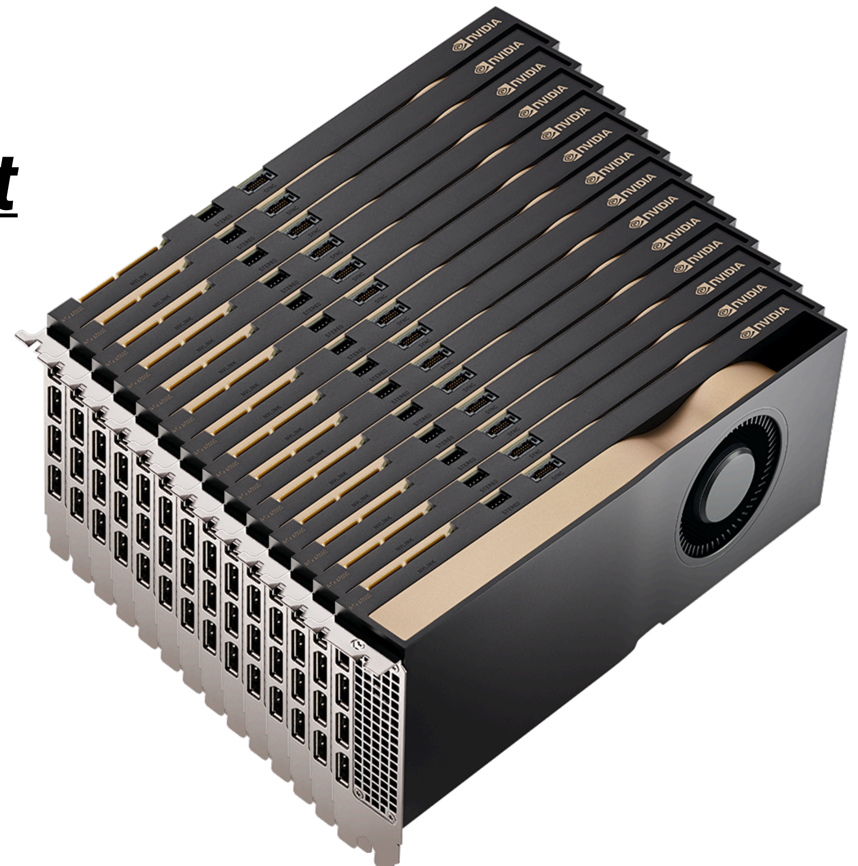


4 TB/s
30 MHz non-empty pp

Detector read-out
O(500) FPGAs

4 TB/s

350 GPUs
5 TB/s @ 30 MHz
of raw data

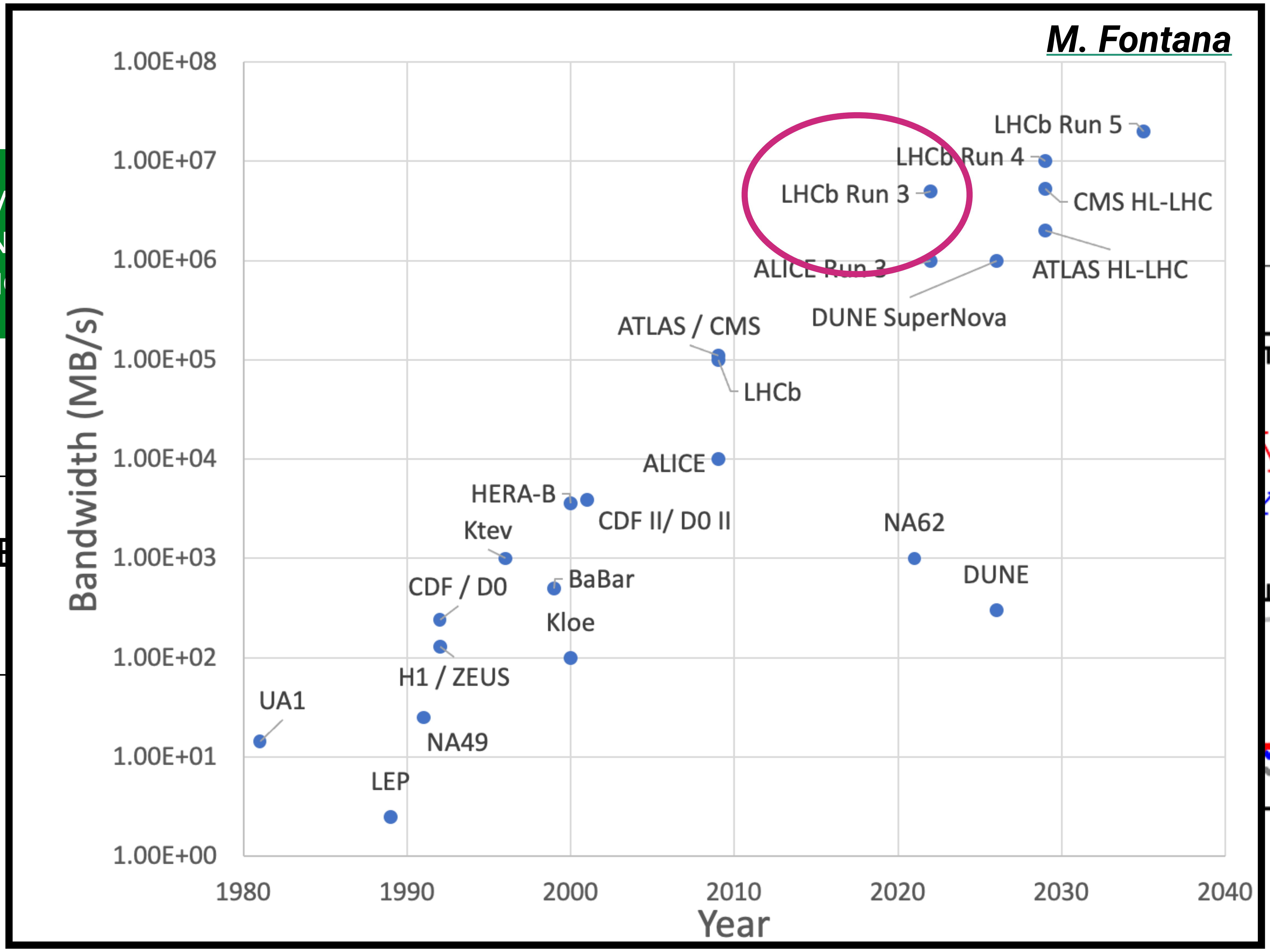


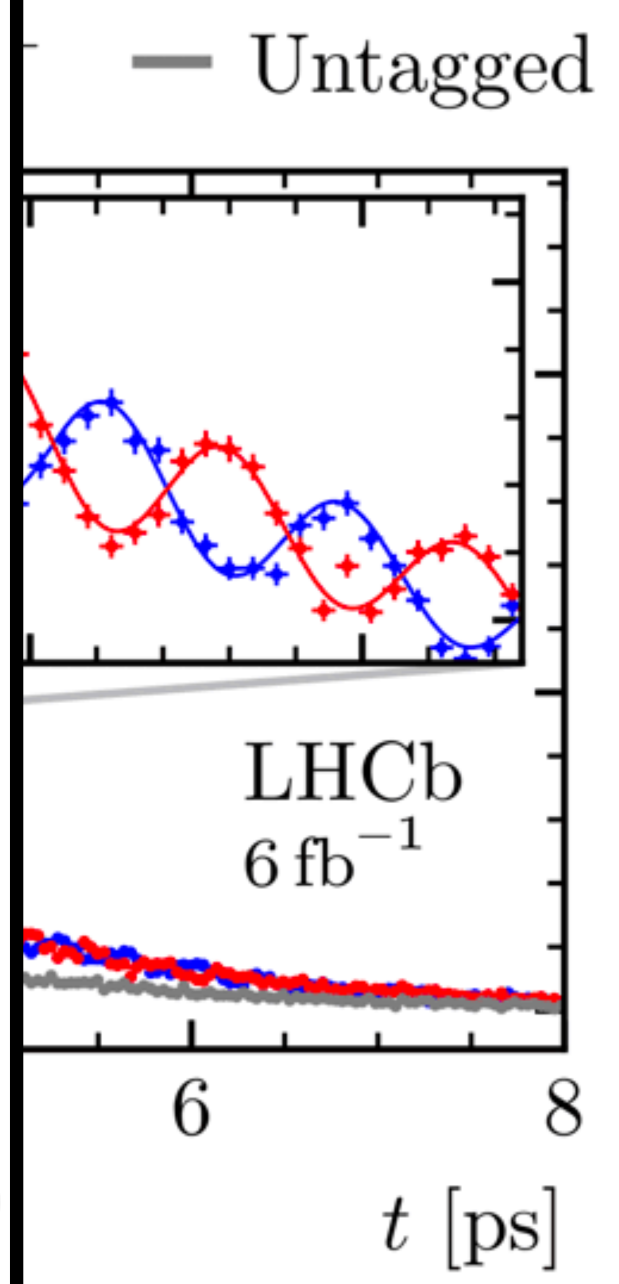
0.5-1.5 MHz

70-200 GB/s

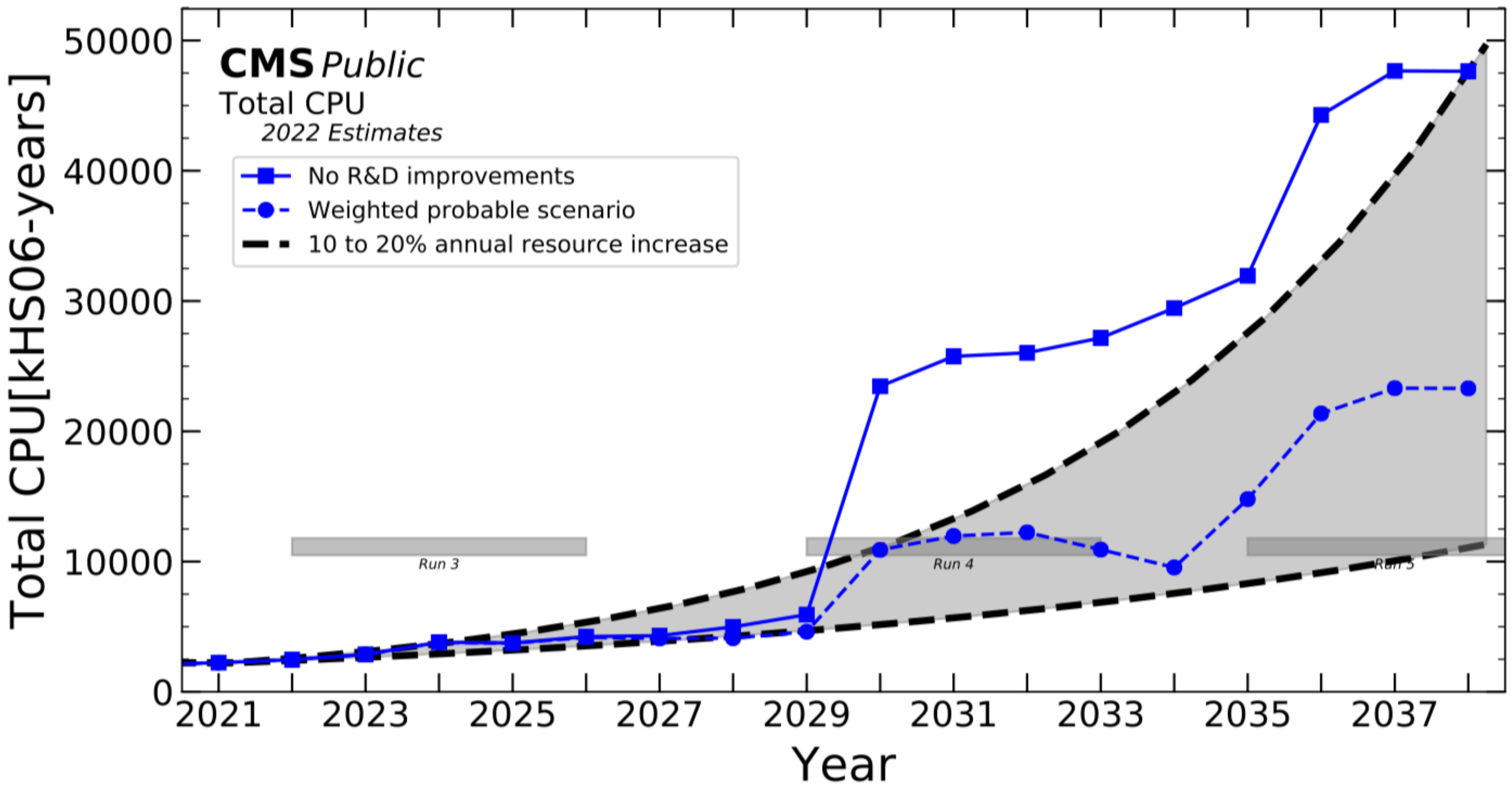
O(10) PB
buffer

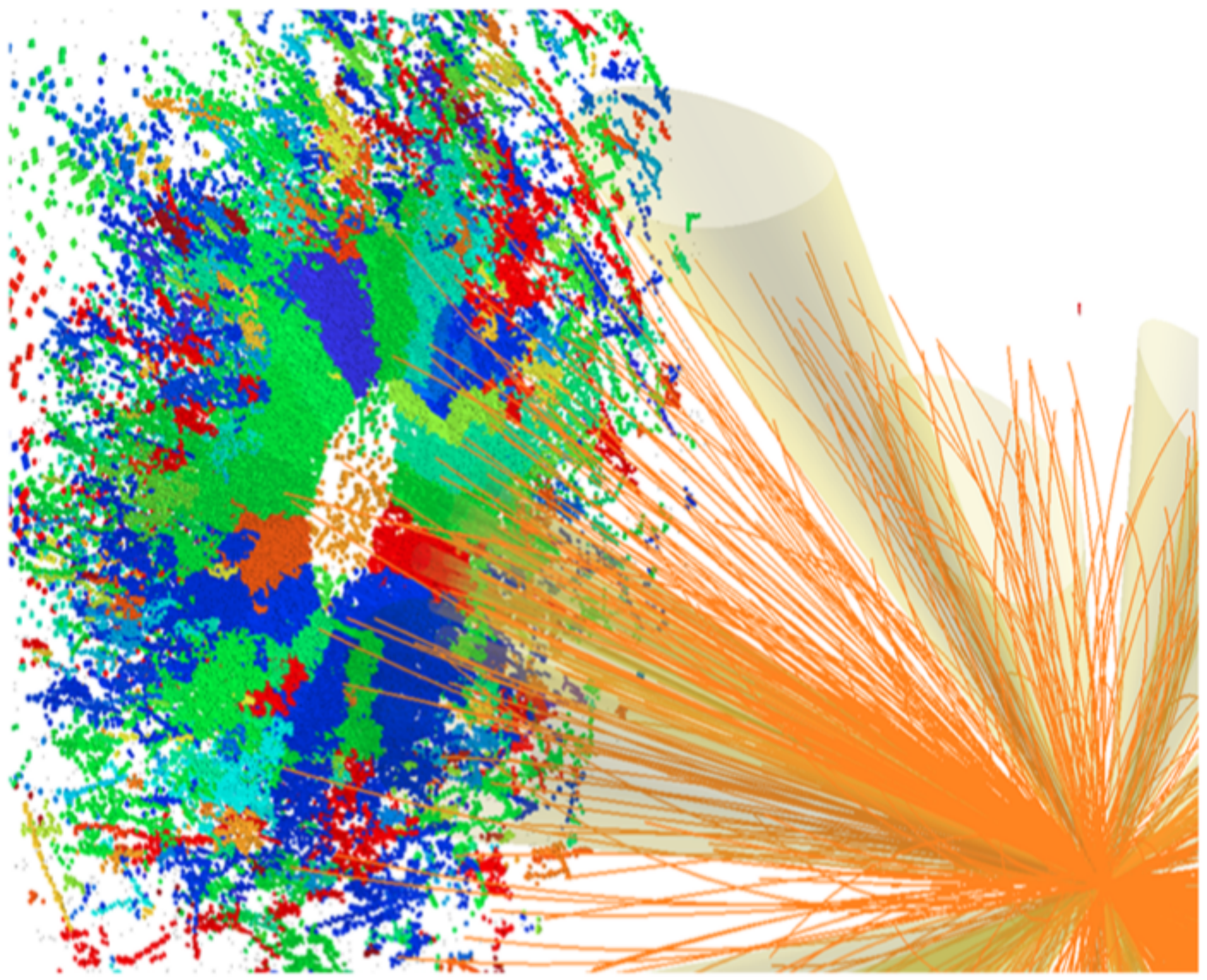
REAL-TIM
ALIGNMEN
CALIBRATI



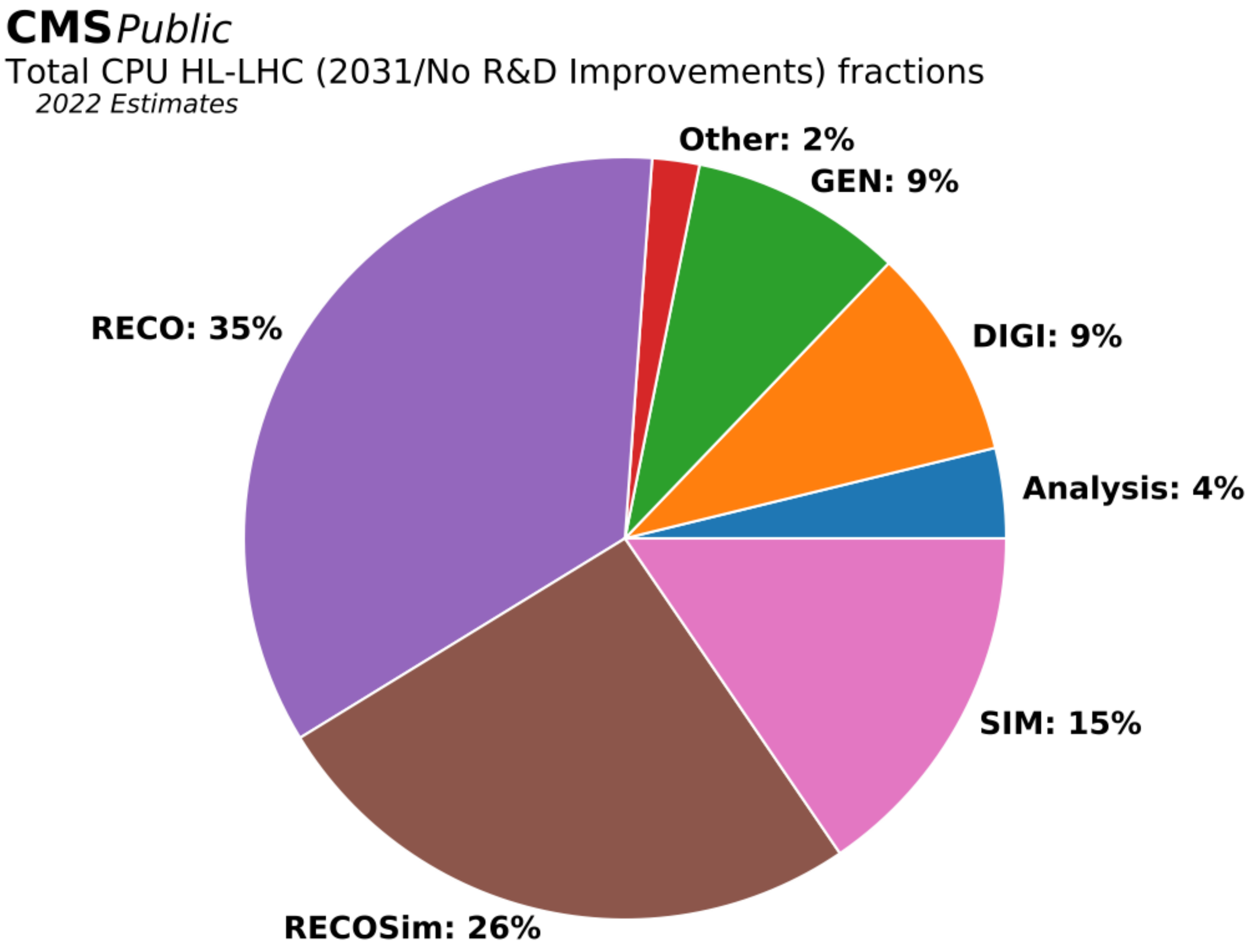


CMS Offline Computing Results





HL-LHC, Simulation of CMS HGCAL with 140 PU

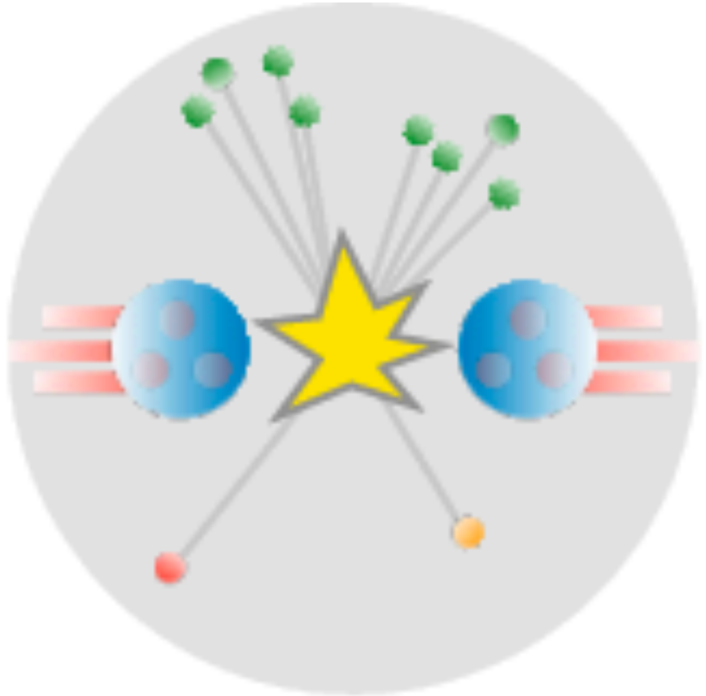

CMS Public
Total CPU HL-LHC (2031/No R&D Improvements) fractions
2022 Estimates
Other: 2%
GEN: 9%
DIGI: 9%
Analysis: 4%
SIM: 15%
RECOSim: 26%
RECO: 35%


$O(10)$
$O(10^3)$
$O(10^{10})$
10^{-18}m
10^{-15}m
10^{-6}m
100m
2T

$\mathcal{O}(10)$ $\mathcal{O}(10^3)$ $\mathcal{O}(10^{10})$

$10^{-18}\mathrm{m}$ $10^{-15}\mathrm{m}$ $10^{-6}\mathrm{m}$ $100\mathrm{m}$

GEN



pp collisions up to production of stable particles [Easy & Fast]

SIM

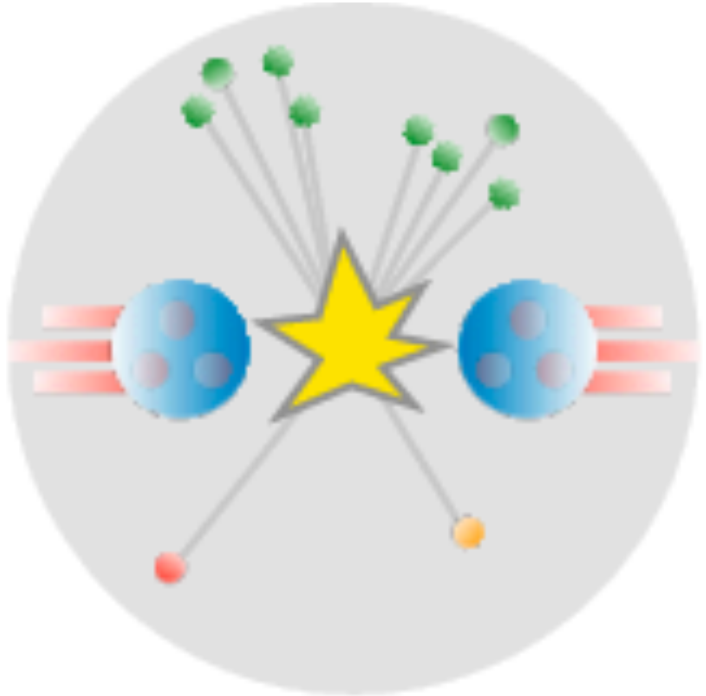
detector response simulation [Hard & Slow]

GEANT4
A SIMULATION TOOLKIT

DIGI+RECO

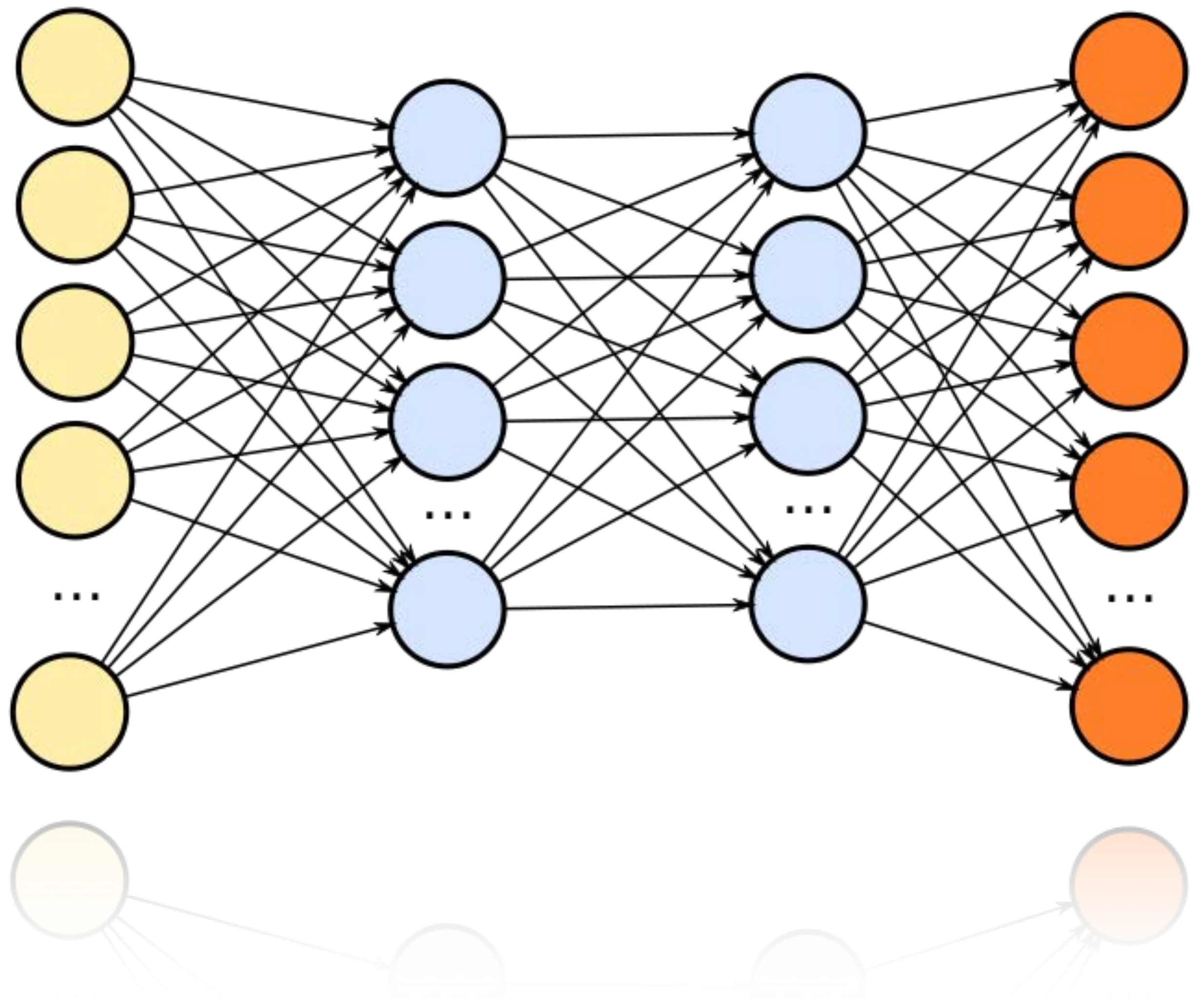
Energy deposits→digital signals→reconstructed by the reconstruction software [Hard & Slow]

GEN



pp collisions up to production of stable particles [Easy & Fast]

SIM



DIGI+RECO

Energy deposits→digital signals→reconstructed by the reconstruction software [Hard & Slow]

GEN | **SIM** | **DIGI+RECO**

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

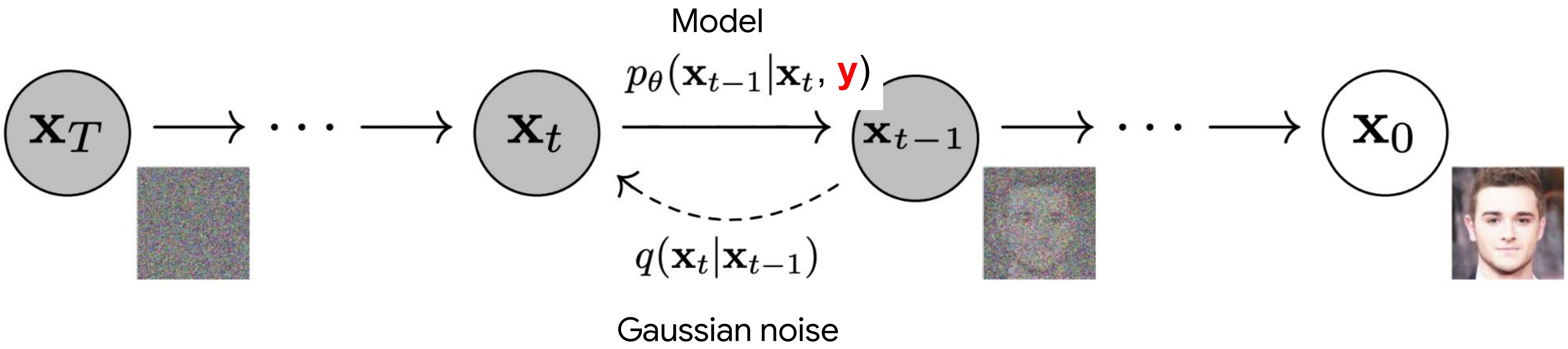
FAST and ACCURATE?

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

p
pro
par

posits→digital
constructed by
uction software
[Hard & Slow]

Diffusion models



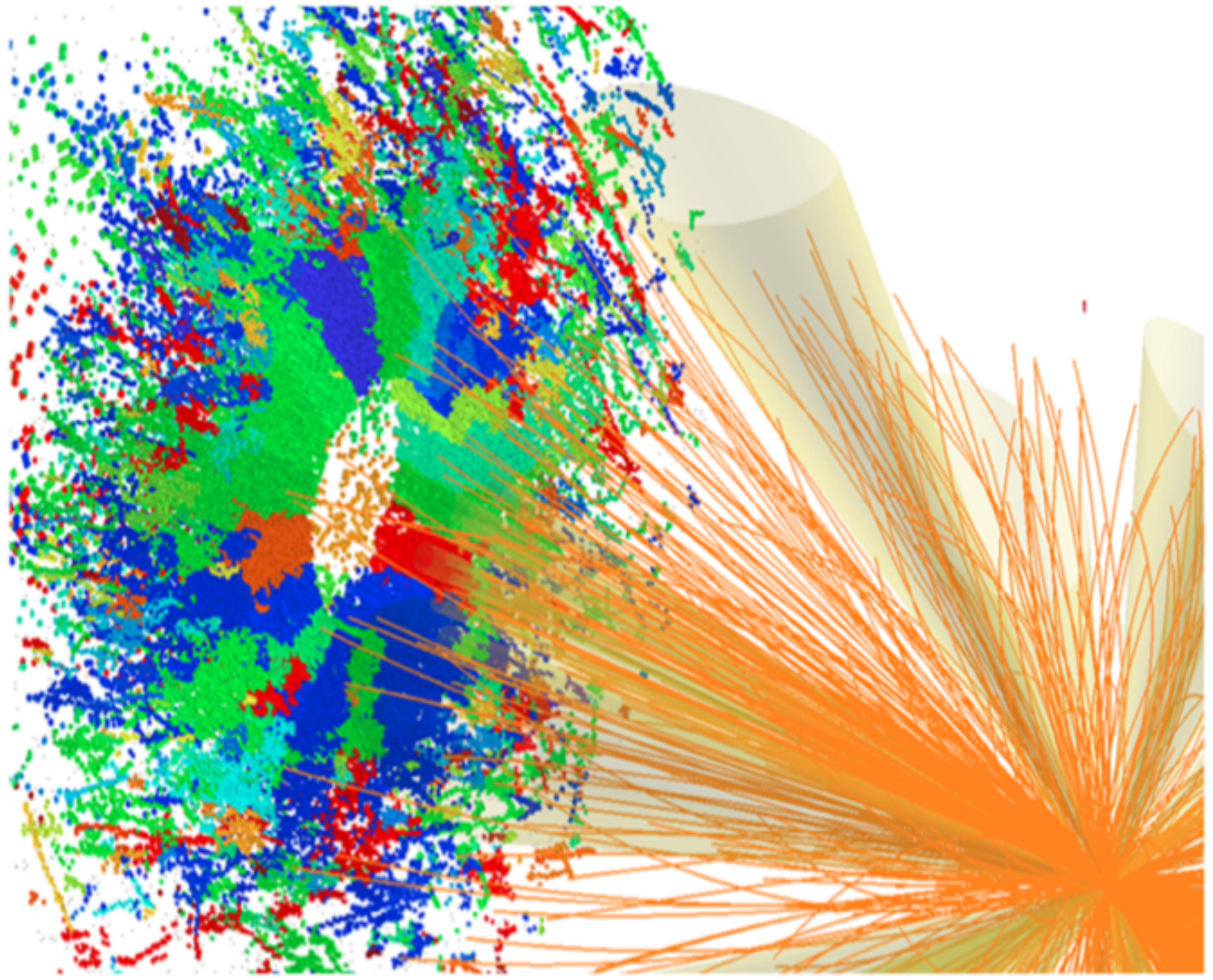
Dall-e 2

an espresso machine that makes coffee from human souls, artstation

2

Dall-e 2

vibrant portrait painting of Salvador Dalí with a robotic half face



https://arxiv.org/pdf/2303.05376.pdf