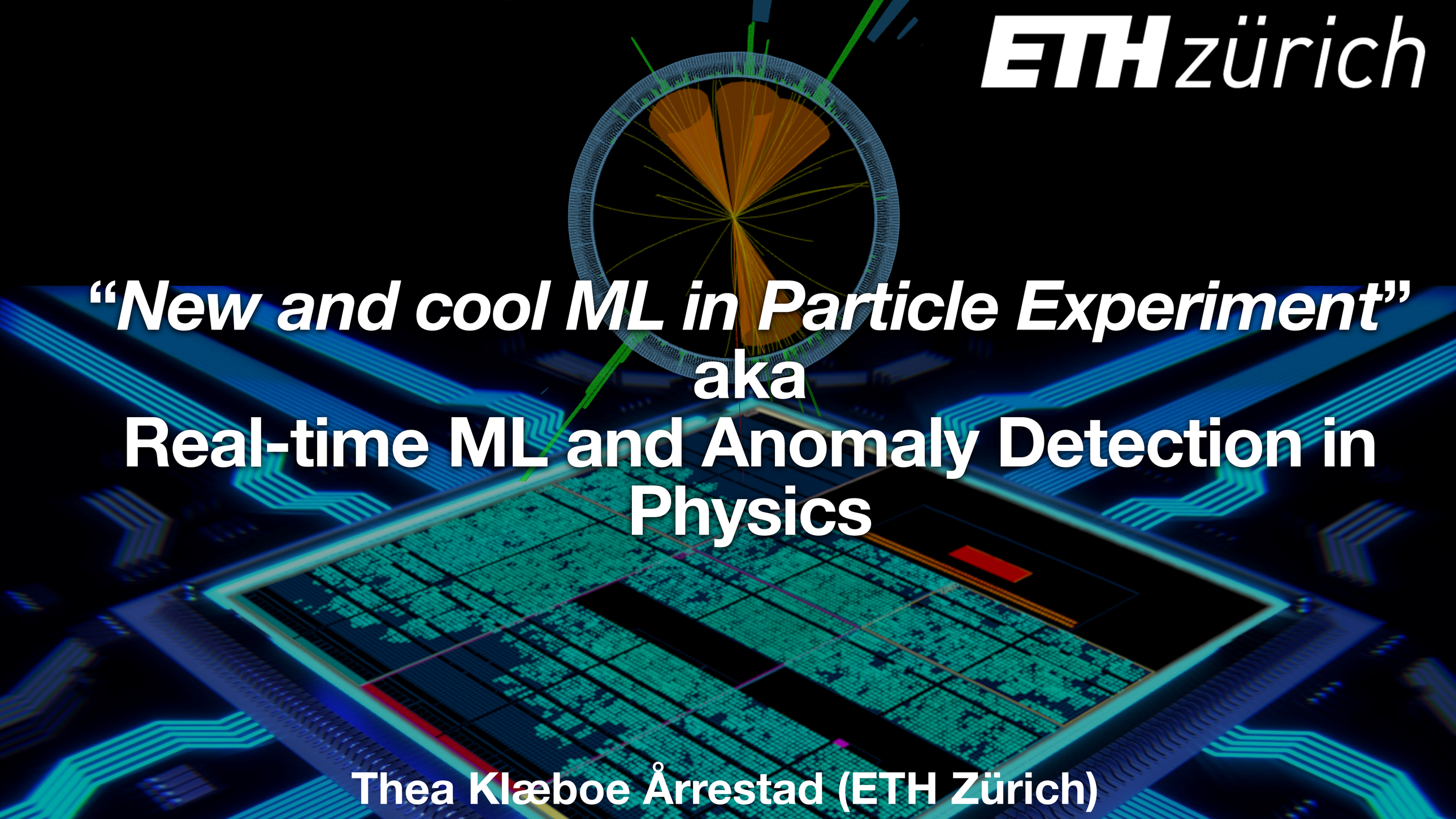


"New and cool ML in Particle Experiment"
aka
Real-time ML and Anomaly Detection in Physics

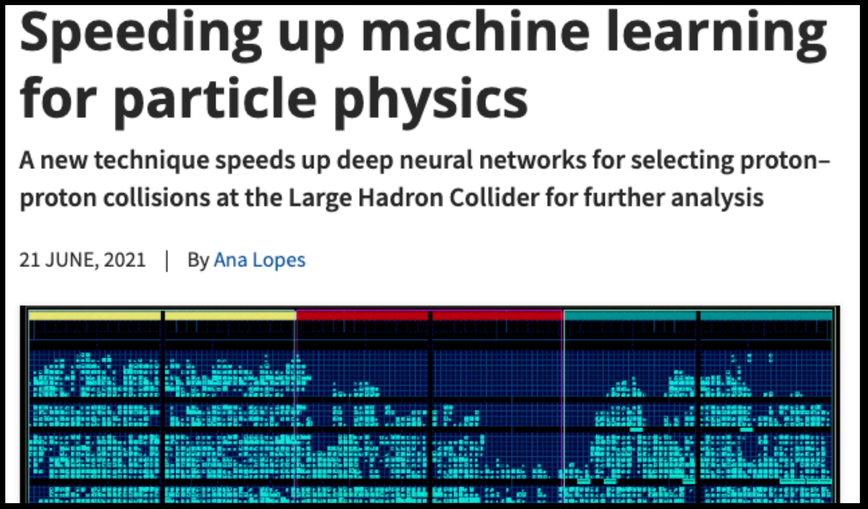
Thea Klæboe Årrestad (ETH Zürich)

Hi!

I am Thea

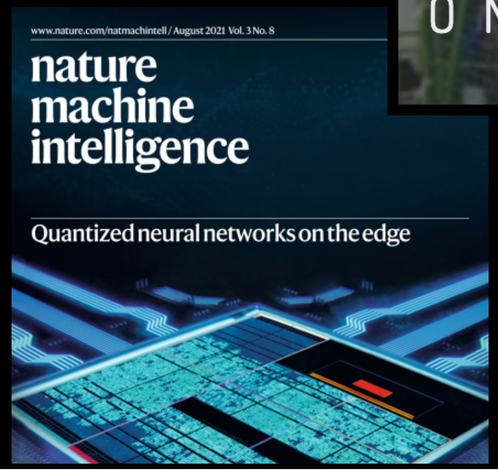
- Experimental Physicist at ETH Zürich & CERN
- Work on:
  - **real-time ML** for data processing systems for the CMS “Trigger” at LHC and elsewhere
  - harnessing ML-based **anomaly detection** for New Physics detection

I work on:

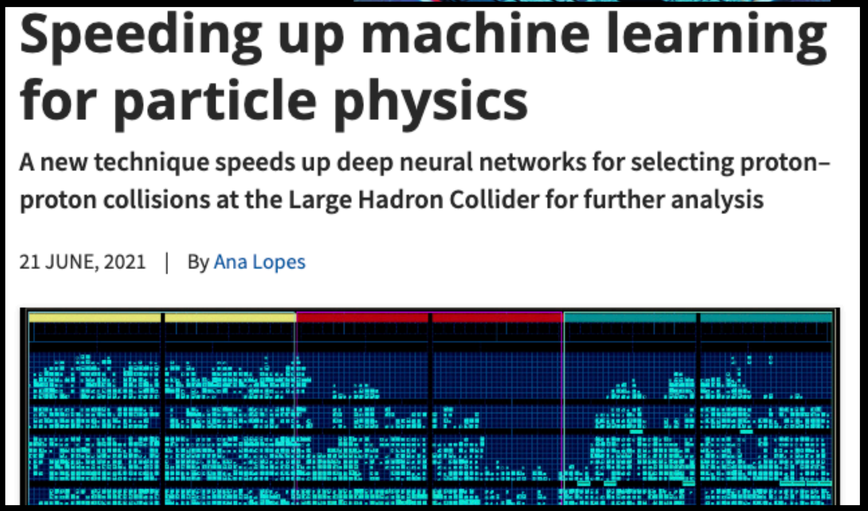


Developing the tools LHC uses for nanosecond inference

I work on:

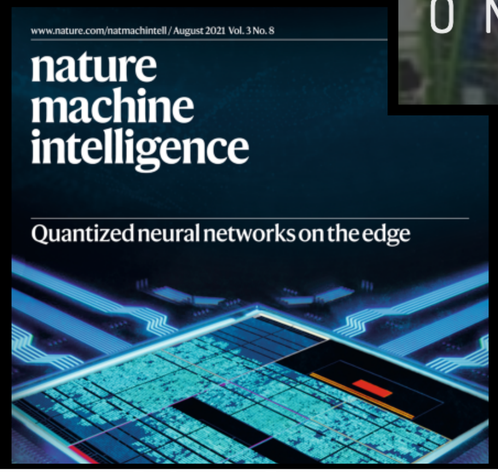


Design searches for New Physics with ML-based anomaly detection

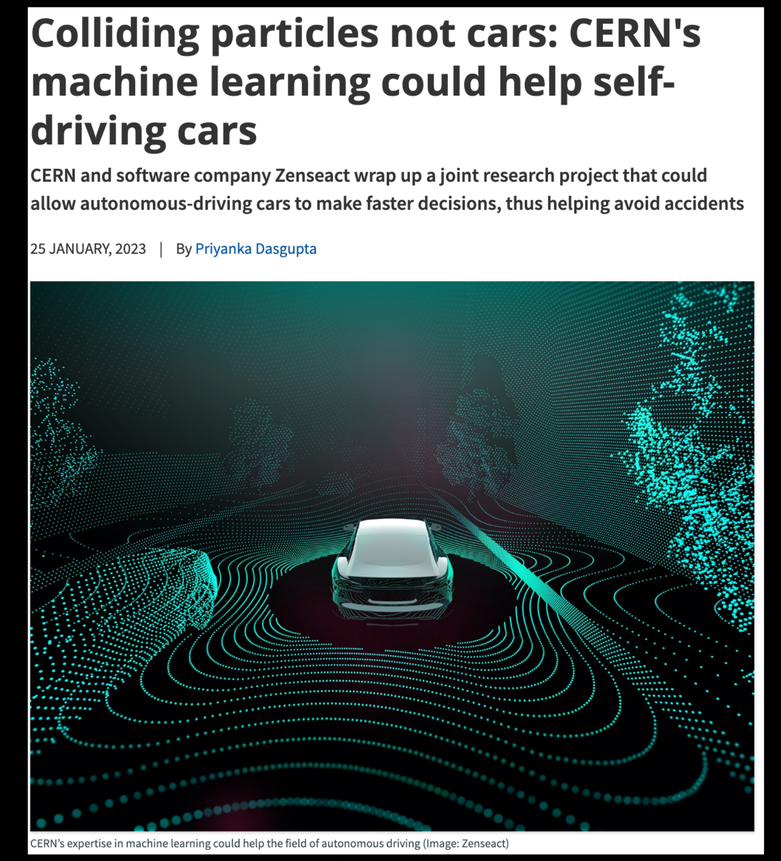


Developing the tools LHC uses for nanosecond inference

I work on:



Design searches for New Physics with ML-based anomaly detection



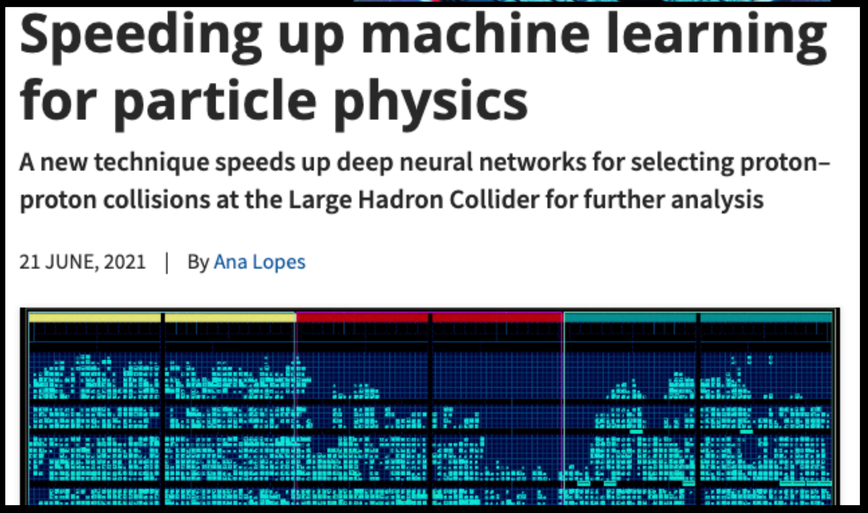
Colliding particles not cars: CERN's machine learning could help self-driving cars

CERN and software company Zenseact wrap up a joint research project that could allow autonomous-driving cars to make faster decisions, thus helping avoid accidents

25 JANUARY, 2023 | By Priyanka Dasgupta

CERN's expertise in machine learning could help the field of autonomous driving (Image: Zenseact)

And real-time inference outside of physics!



Speeding up machine learning for particle physics

A new technique speeds up deep neural networks for selecting proton–proton collisions at the Large Hadron Collider for further analysis

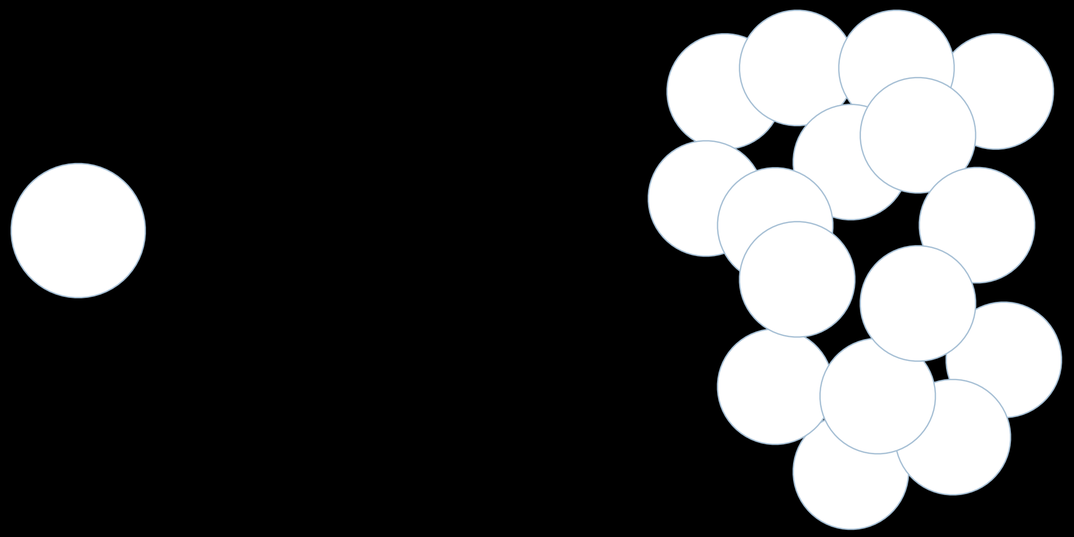
21 JUNE, 2021 | By Ana Lopes

Developing the tools LHC uses for nanosecond inference

“Today has been made possible thanks to extraordinarily large neural networks trained on large datasets”

Ilya Sutskever, co-founder OpenAI, neurIPS 2024 ToT keynote

GPT-3 GPT-4 (MoE)



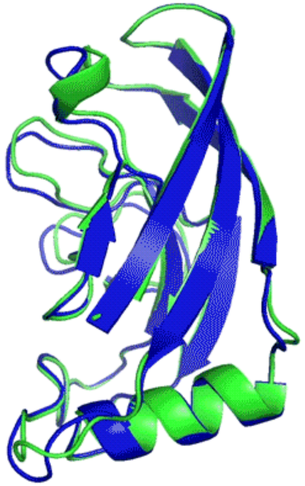
175,000,000,000
(0.16% of neurons in your brain)

800,000,000,000 and 120 layers
(1.6% of neurons in your brain)

GPT-5 rather system of models: pre-filtering, then GPT chat or reasoning

Example of AI for discovery?

T1037 / 6vr4
90.7 GDT
(RNA polymerase domain)



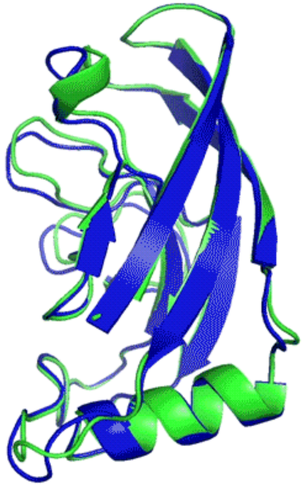
T1049 / 6y4f
93.3 GDT
(adhesin tip)

Experimental result
Computational prediction

AlphaFold nature cover

sequence—the structure prediction component of the 'protein folding problem'[8]—has been an important open research problem for more than 50 years[9]. Despite recent

T1037 / 6vr4
90.7 GDT
(RNA polymerase domain)

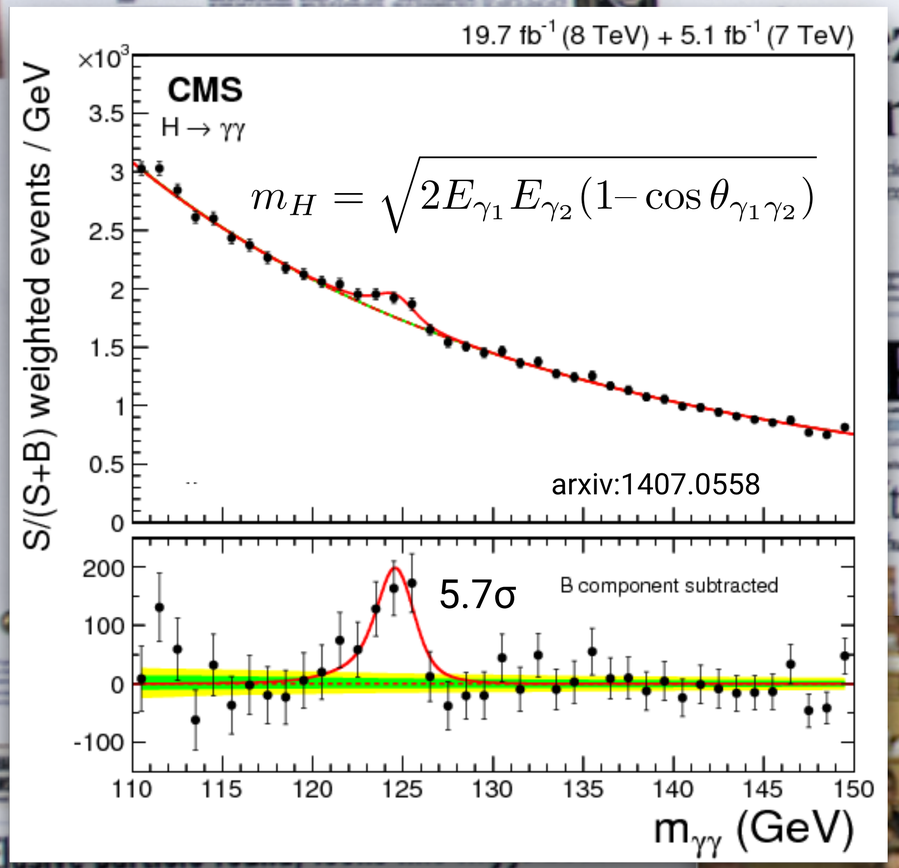


T1049 / 6y4f
93.3 GDT
(adhesin tip)

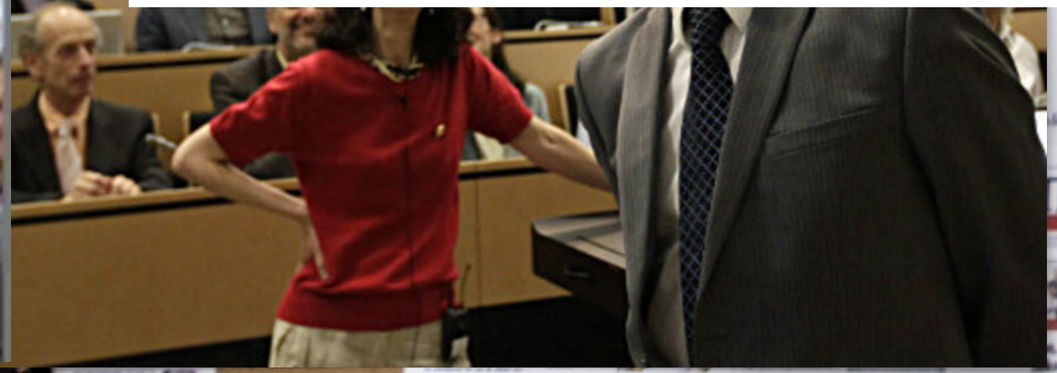
Experimental result
Computational prediction

AlphaFold nature cover

sequence—the structure prediction component of the 'protein folding problem'[8]—has been an important open research problem for more than 50 years[9]. Despite recent



Nature Review Analysis	Years of data collection	Sensitivity without machine learning	Sensitivity with machine learning	Ratio of P values	Additional data required
CMS[24] $H \to \gamma\gamma$	2011–2012	2.2σ, $P = 0.014$	2.7σ, $P = 0.0035$	4.0	51%
ATLAS[43] $H \to \tau^+\tau^-$	2011–2012	2.5σ, $P = 0.0062$	3.4σ, $P = 0.00034$	18	85%
ATLAS[99] $VH \to bb$	2011–2012	1.9σ, $P = 0.029$	2.5σ, $P = 0.0062$	4.7	73%
ATLAS[41] $VH \to bb$	2015–2016	2.8σ, $P = 0.0026$	3.0σ, $P = 0.00135$	1.9	15%
CMS[100] $VH \to bb$	2011–2012	1.4σ, $P = 0.081$	2.1σ, $P = 0.018$	4.5	125%



semianalysis 2023

Train a typical LLM:

- **2^{25-27} floating point operations**
- **>20,000 A100 GPUs**
- **90-100 days**
- **$60 million**
- **Train set: 13 trillion tokens**

semianalysis 2023

whats the typical latency of your (GPT-5) web app?

I don't have a confirmed published number from OpenAI for the **web app** version of GPT-5, so I can't give an exact "official" latency. However, there are some estimates / benchmark-reports that give a ballpark. Here are what I found +

- ~ **0.15-0.3 seconds** before you see the first token (after you submit your prompt)

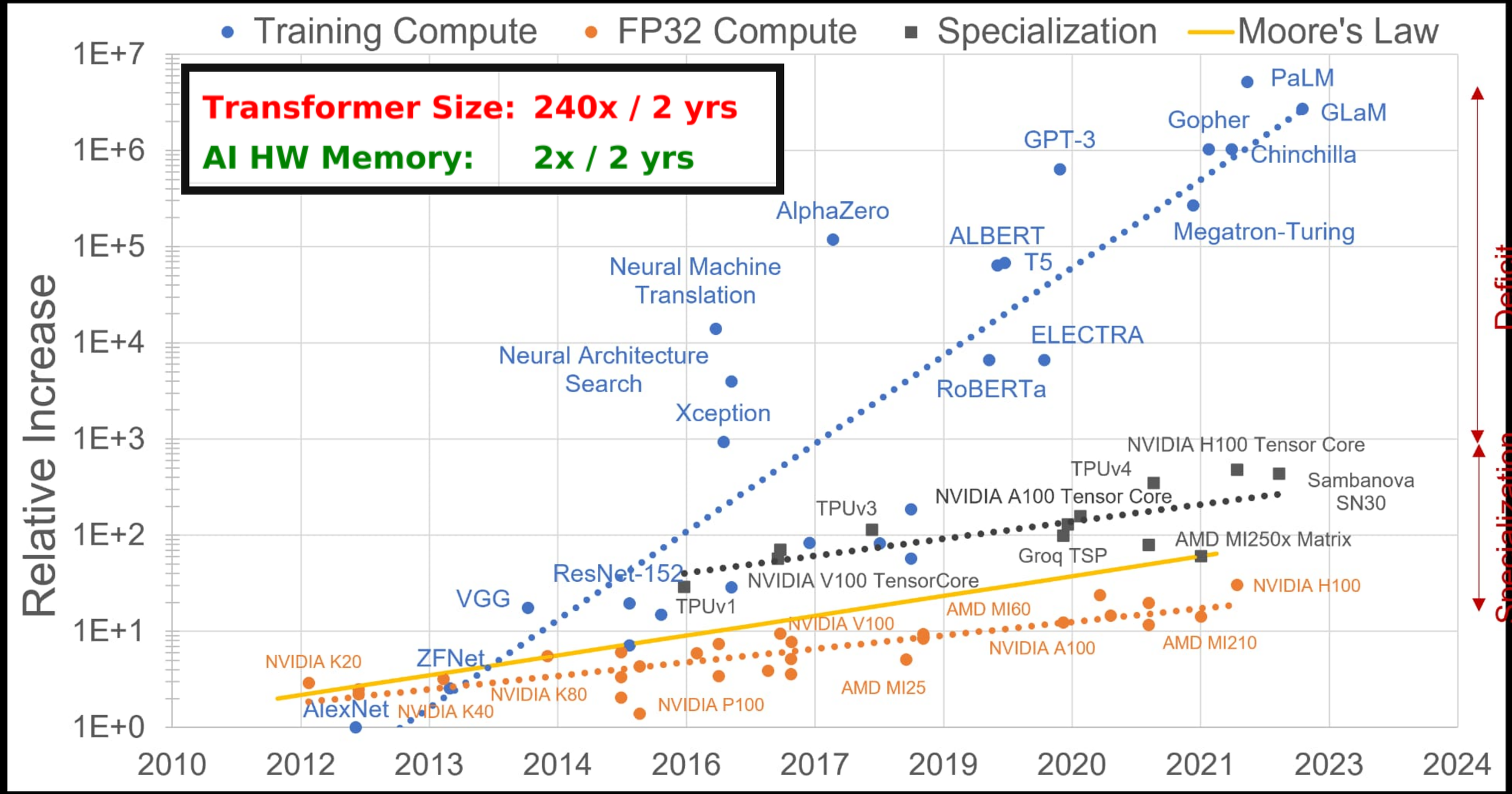
Train a typical LLM:

- **$2^{25\text{-}27}$ floating point operations**
- **>20,000 A100 GPUs**
- **90-100 days**
- **$60 million**
- **Train set: 13 trillion tokens**

Inference typical LLM:

- **Multiple clusters of >100 GPUs**
- **Model carefully mapped onto hardware**

Why can this be a problem?

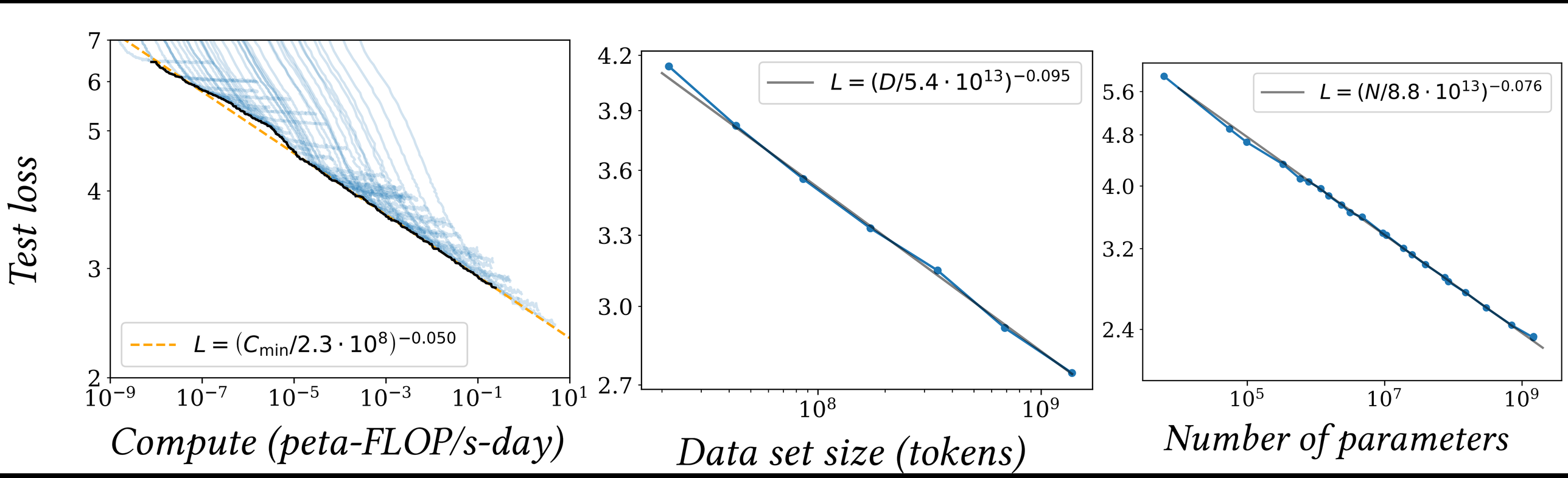


Kartik Hedge

Scaling Laws in AI

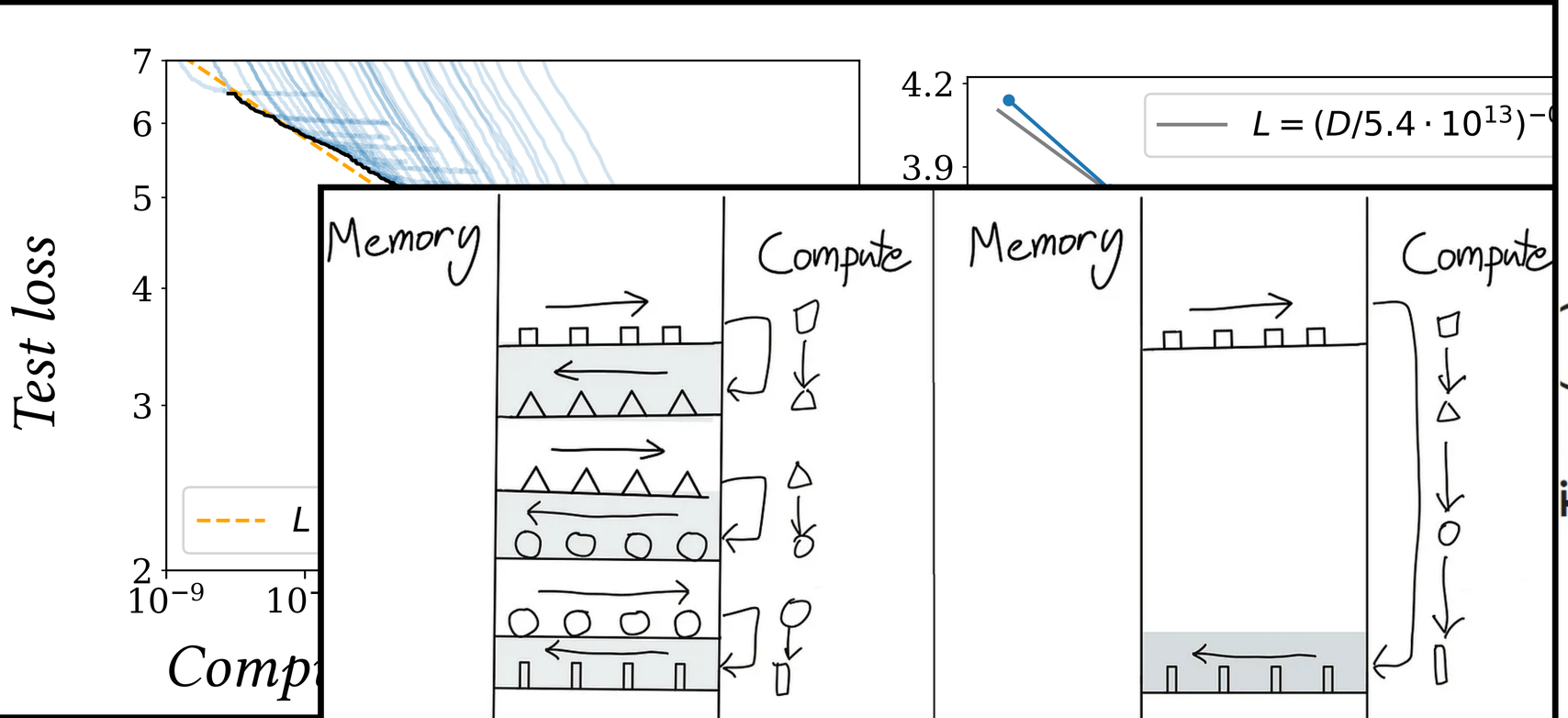
"We study empirical scaling laws for language model performance on the cross-entropy loss. ***The loss scales as a power-law with model size, dataset size, and the amount of compute used for training****, with some trends spanning more than seven orders of magnitude."*

--- OpenAI

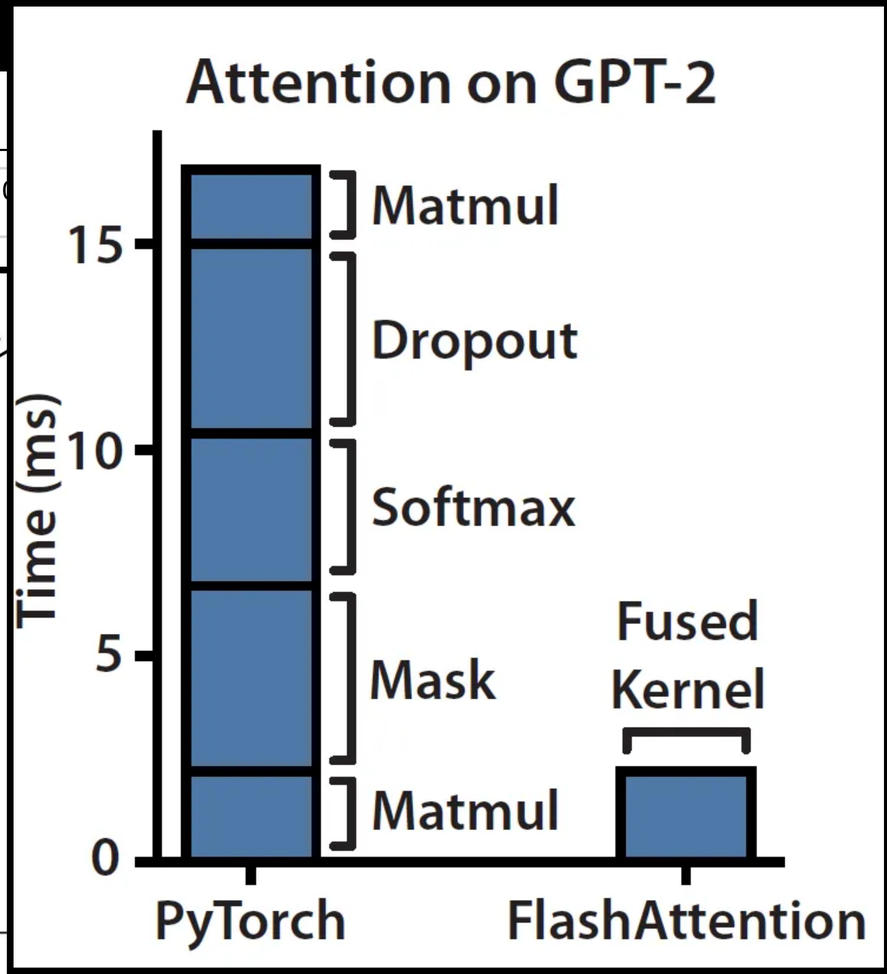


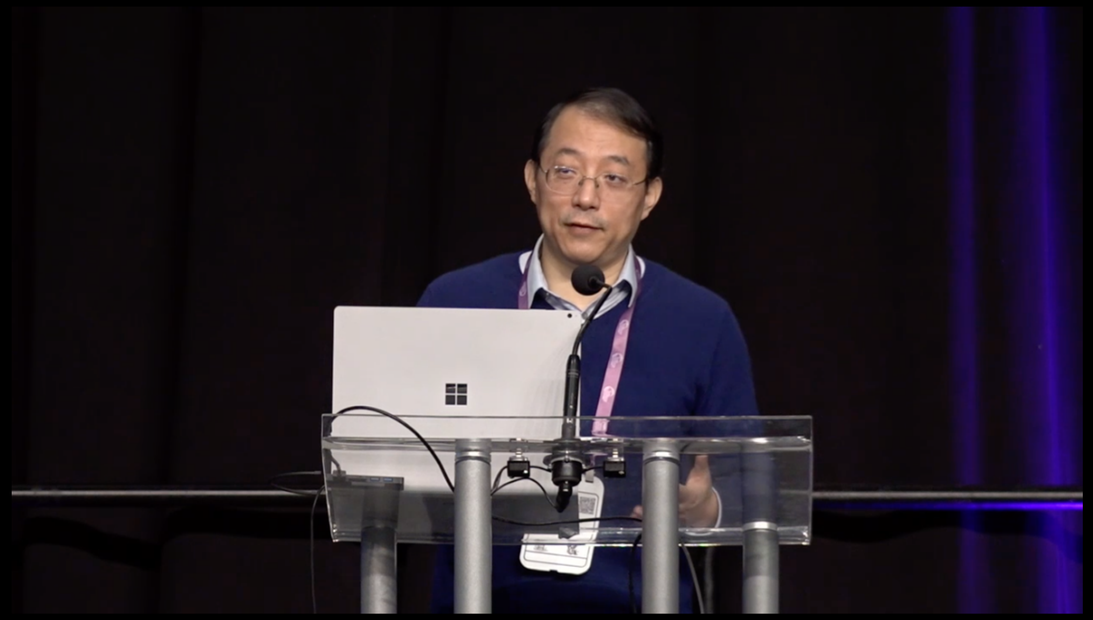
Kaplan et al. (2020)

FlashAttention, FlashAttention explained



Kaplan et al.





Scaling Folklore in Systems

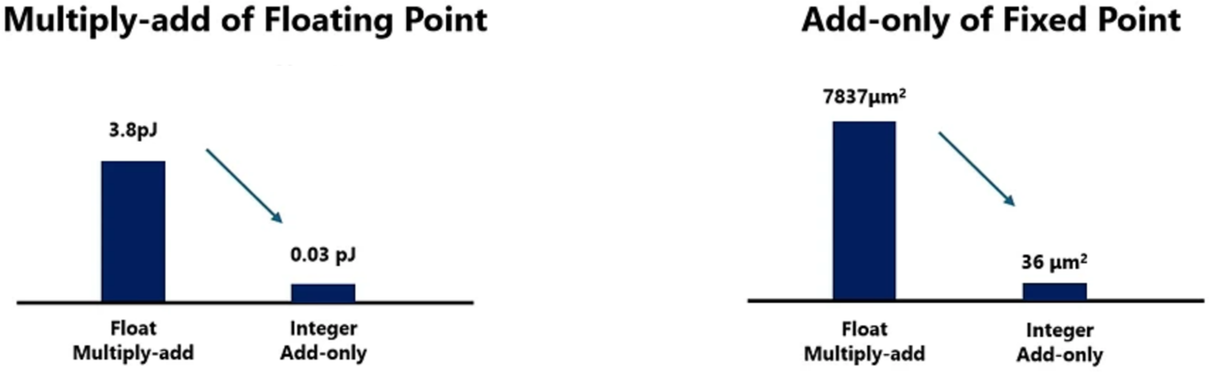
"You can't scale by throwing hardware at the problem forever"

- Scale exacerbates hidden flaws and inefficiencies
- Everything fails at scale
- Scaling is sub-linear
- Scaling requires revisiting assumptions

"We know you cant scale a system by throwing hardware at it forever."

neurIPS 2024 keynote: "A Match Made in Silicon: The Co-Evolution of Systems and AI" Lidong Zhou

A Potential Efficiency Revolution



1.1 Computing's energy problem (and what we can do about it) | IEEE Conference Publication | IEEE Xplo

The Era of 1-bit LLMs: All Large Language Models are in 1.58 Bits

Shuming Ma* Hongyu Wang* Lingxiao Ma Lei Wang Wenhui Wang
Shaohan Huang Li Dong Ruiping Wang Jilong Xue Furu Wei◇
https://aka.ms/GeneralAI

Abstract

Recent research, such as BitNet [WMD+23], is paving the way for a new era of 1-bit Large Language Models (LLMs). In this work, we introduce a 1-bit LLM variant, namely **BitNet b1.58**, in which every single parameter (or weight) of the LLM is ternary {-1, 0, 1}. It matches the full-precision (i.e., FP16 or BF16) Transformer LLM with the same model size and training tokens in terms of both perplexity and end-task performance, while being significantly more cost-effective in terms of latency, memory, throughput, and energy consumption. More profoundly, the 1.58-bit LLM defines a new scaling law and recipe for training new generations of LLMs that are both high-performance and cost-effective. Furthermore, it enables a new computation paradigm and opens the door for designing specific hardware optimized for 1-bit LLMs.

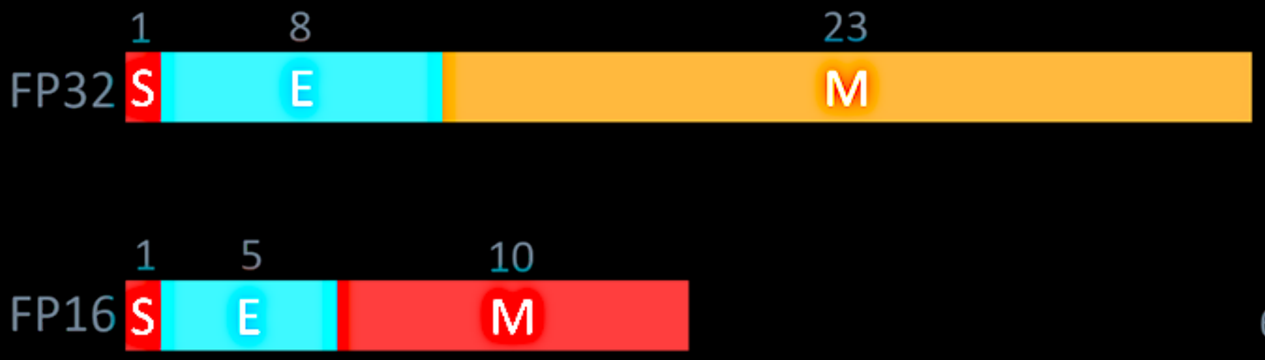
Pareto Improvement

neurIPS 2024: “A Match Made in Silicon: The Co-Evolution of Systems and AI" Lidong Zhou

GPT-3

175,000,000,000

FP16 vs FP32

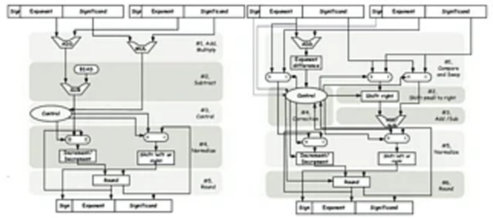


	Range	Accuracy
FP32	10^{-38} - 10^{38}	0.000006%
FP16	$6x10^{-5}$ - $6x10^{4}$	0.05%

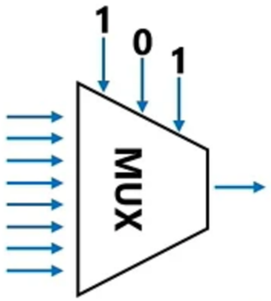
→ ~700 GB
(175B × 4B/par)
→ 9 NVIDIA A100s

→ ~350 GB
(175B × 2B/par)
→ 4.5 NVIDIA A100 ($10 000/ea)

LUT (Lookup Table) stores precomputed results for quick retrieval to replace expensive runtime computations. It is **Fast, Efficient and Simple.**



MAC (Multiply-accumulate)



LUT (Lookup Tables)

Only the Simple Can Scale, LUTs are among the simplest.

neurIPS 2024: “A Match Made in Silicon: The Co-Evolution of Systems and AI" Lidong Zhou

hls4ml

Compressing deep neural networks on FPGAs to binary and ternary precision with hls4ml

Jennifer Ngadiuba, Vladimir Loncar, Maurizio Pierini, Sioni Summers
Giuseppe Di Guglielmo
Javi
University of C
La Jolla, C
Sergo Jindariani, Mia Liu
Fermi National Acc
Batavia, IL
Z
University
Chicag

Eff

volution

hls4ml

Fast convolutional neural networks on FPGAs with hls4ml

Thea Aarrestad, Vladimir Loncar, Nicolò Ghielmetti, Maurizio Pierini, Sioni Summers
European Organization for Nuclear Research (CERN)
CH-1211 Geneva 23, Switzerland

Califor
Pas
Giu
Columl
Sergo
Fer

Automatic heterogeneous quantization of deep neural networks for low-latency inference on the edge for particle detectors

Claudionor N. Coelho Jr.
Palo Alto Networks (California, USA)

Aki Kuusela, Shan Li, and Hao Zhuang
Google LLC (California, USA)

Thea Aarrestad,* Vladimir Loncar,† Maurizio Pierini, Adrian Alan Pol, and Sioni Summers
European Organization for Nuclear Research (CERN) (Geneva, Switzerland)

Jennifer Ngadiuba
California Institute of Technology (Caltech) (California, USA)

Fast inference of deep neural networks in FPGAs for particle physics

Javier Duarte[a] , **Song Han**[b] , **Philip Harris**[b] , **Sergo Jindariani**[a] , **Edward Kreinar**[c] , **Benjamin Kreis**[a] , **Jennifer Ngadiuba**[d] , **Maurizio Pierini**[d] , **Ryan Rivera**[a] , **Nhan Tran**[a] , **Zhenbin Wu**[e]
[a] *Fermi National Accelerator Laboratory, Batavia, IL 60510, USA*

NeurIPS 2024

Low Latency Transformer Inference on FPGAs for Physics Applications with hls4ml

Zhixing Jiang, Dennis Yin, Yihui Chen,
Elham E Khoda, Scott Hauck, Shih-Chieh Hsu

University of Washington
Seattle, WA 98195, USA

Ekaterina Govorkova, Philip Harris,
Vladimir Loncar, Eric A. Moreno

Massachusetts Institute of Technology
Cambridge, MA 02139, USA

Real-time semantic segmentation on FPGAs for autonomous vehicles with hls4ml

Nicolò Ghielmetti, **Vladimir Loncar**, **Maurizio Pierini**, **Marcel Roed**, **Sioni Summers**
European Organization for Nuclear Research (CERN)
CH-1211 Geneva 23, Switzerland

Thea Aarrestad
Institute for Particle Physics and Astrophysics, ETH Zürich
8093 Zürich, Switzerland

Christoffer Petersson
Zenseact
Gothenburg, 41756, Sweden

Hampus Linander
University of Gothenburg
Gothenburg, 40530, Sweden

Jennifer Ngadiuba
Fermi National Accelerator Laboratory
Batavia, IL 60510, USA

Kelvin Lin
University of Washington
Seattle, WA 98195, USA

Philip Harris
Massachusetts Institute of Technology
Cambridge, MA 02139, USA

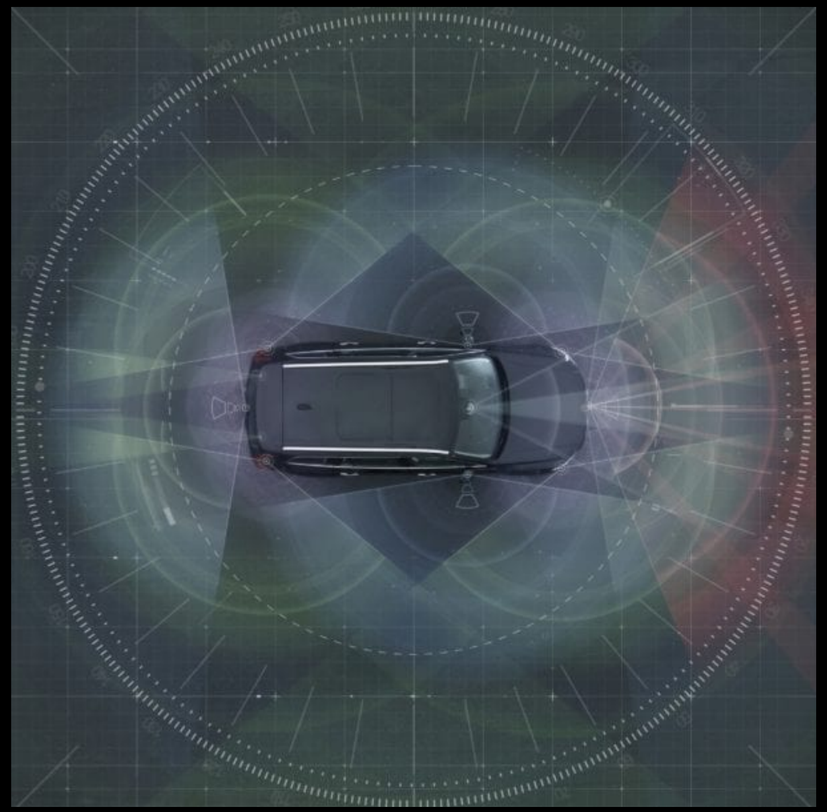
t.

neurIPS 2024: "A Match Made in Silicon: The Co-E hou

Where else might you want to use this type of “efficient” models?

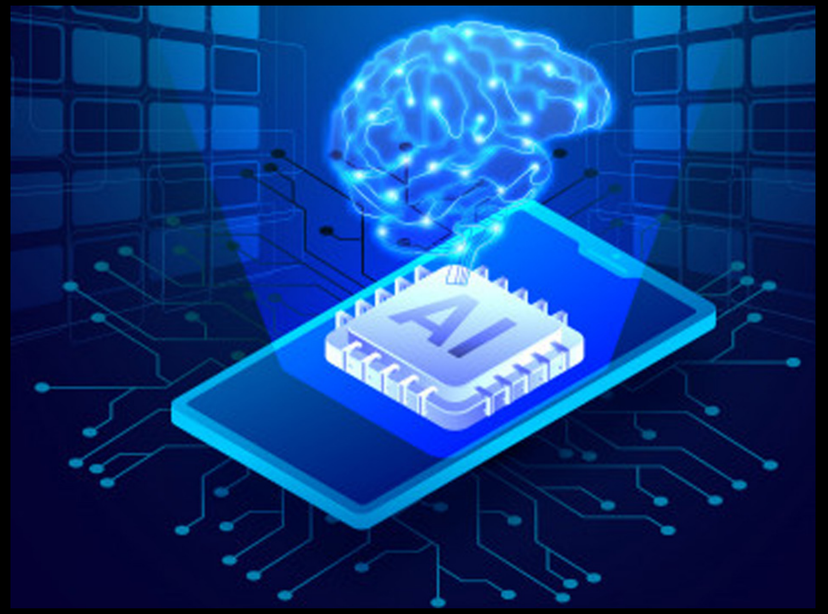
EDGE

O(1) ms



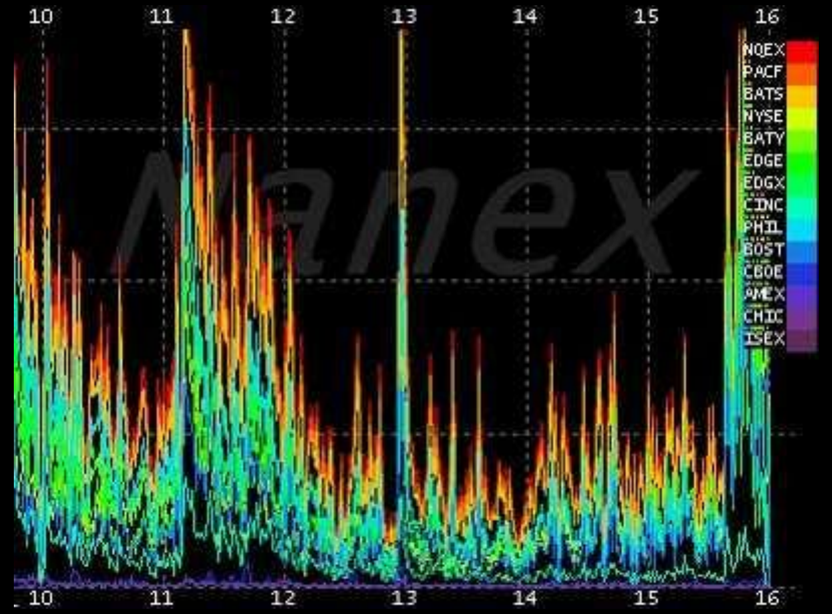
ASIC/GPU

O(1) ms



ASIC

O(1) ns



FPGA

Low power

On-device

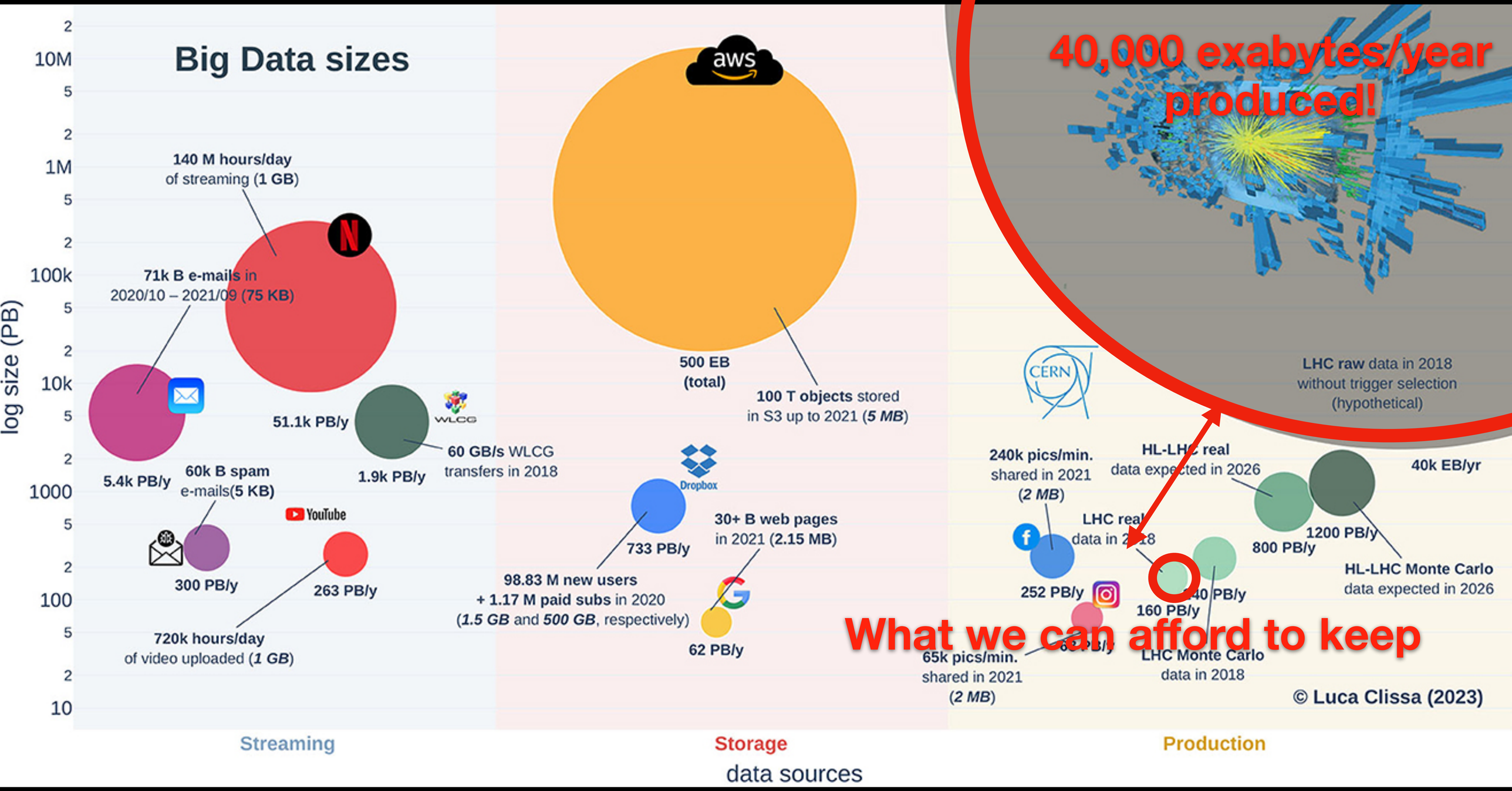
High-throughput

EFFICIENT ML ALGORITHMS

(Data-efficient)

Low-latency

Why do we need this in big science?


Big Data sizes
log size (PB)
2
10M
5
2
1M
5
2
100k
5
2
10k
5
2
1000
5
2
100
5
2
10
140 M hours/day
of streaming (1 GB)
71k B e-mails in
2020/10 – 2021/09 (75 KB)
5.4k PB/y
60k B spam
e-mails(5 KB)
300 PB/y
720k hours/day
of video uploaded (1 GB)
263 PB/y
51.1k PB/y
1.9k PB/y
60 GB/s WLCG
transfers in 2018
500 EB
(total)
100 T objects stored
in S3 up to 2021 (5 MB)
733 PB/y
98.83 M new users
+ 1.17 M paid subs in 2020
(1.5 GB and 500 GB, respectively)
30+ B web pages
in 2021 (2.15 MB)
62 PB/y
40,000 exabytes/year
produced!
LHC raw data in 2018
without trigger selection
(hypothetical)
240k pics/min.
shared in 2021
(2 MB)
252 PB/y
65k pics/min.
shared in 2021
(2 MB)
LHC real
data in 2018
160 PB/y
LHC Monte Carlo
data in 2018
HL-LHC real
data expected in 2026
800 PB/y
40k EB/yr
1200 PB/y
HL-LHC Monte Carlo
data expected in 2026
What we can afford to keep
© Luca Clissa (2023)
Streaming
Storage
Production
data sources

1958-1965 The Ferranti Mercury (vacuum tube)

Magnetic core storage 4 × 32K × 20 bits (~4*80 KB)

1963: IBM 7090 (transistor technology)
Tape storage and FORTRAN!

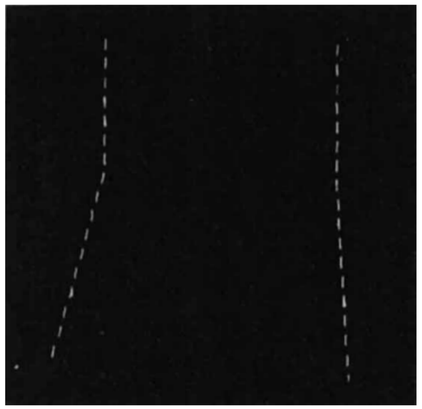


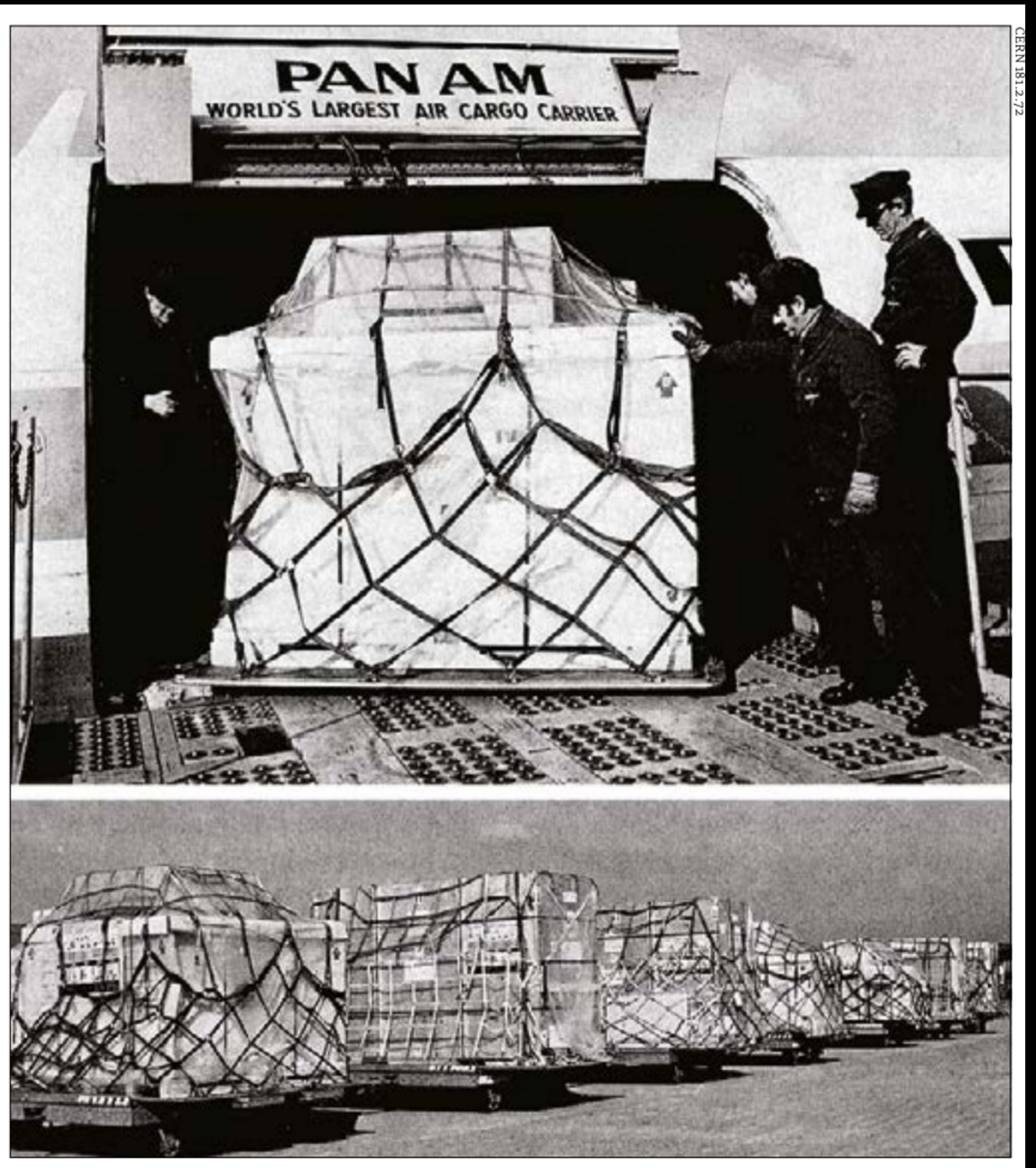
FIG. 16. Scattering of a 300 MeV proton (CRONIN and RENNINGER, 1960).

of the absence of background tracks. In a typical experiment Cork took 100,000 photographs at a rate of 10,000 per day and the analysis time was about 3 weeks.

Could analyse 300,000 frames of spark-chamber data!

everything had to go through the accumulator, the cpu was blocked during I/O. It was immediately clear that there was a big mismatch between the power of the computing engine and its input/output capability.

1972: CDC 7600
Most powerful computing system in Europe

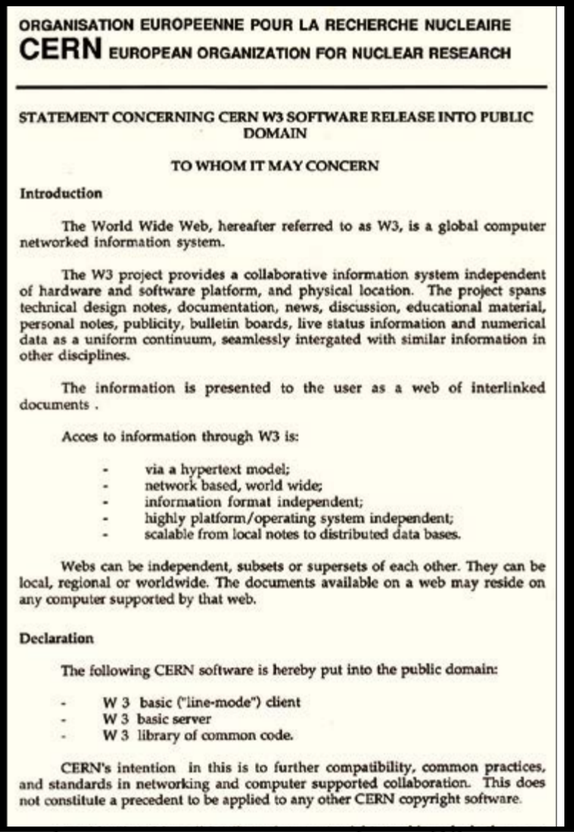


450,000 processor cores, 10 000 servers.
> 90% provided through private cloud based on OpenStack

Handling 45 Petabytes per week
Storage capacity ~3 exabytes (tape and disk) - 10 trillion times bigger than the Ferranti!
Reading rate 1 TB/s

1989: WWW

1993: Public domain, later open license



ORGANISATION EUROPEENNE POUR LA RECHERCHE NUCLEAIRE
CERN EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH

STATEMENT CONCERNING CERN W3 SOFTWARE RELEASE INTO PUBLIC DOMAIN

TO WHOM IT MAY CONCERN

Introduction

The World Wide Web, hereafter referred to as W3, is a global computer networked information system.

The W3 project provides a collaborative information system independent of hardware and software platform, and physical location. The project spans technical design notes, documentation, news, discussion, educational material, personal notes, publicity, bulletin boards, live status information and numerical data as a uniform continuum, seamlessly intergated with similar information in other disciplines.

The information is presented to the user as a web of interlinked documents .

Acces to information through W3 is:

- via a hypertext model;
- network based, world wide;
- information format independent;
- highly platform/operating system independent;
- scalable from local notes to distributed data bases.

Webs can be independent, subsets or supersets of each other. They can be local, regional or worldwide. The documents available on a web may reside on any computer supported by that web.

Declaration

The following CERN software is hereby put into the public domain:

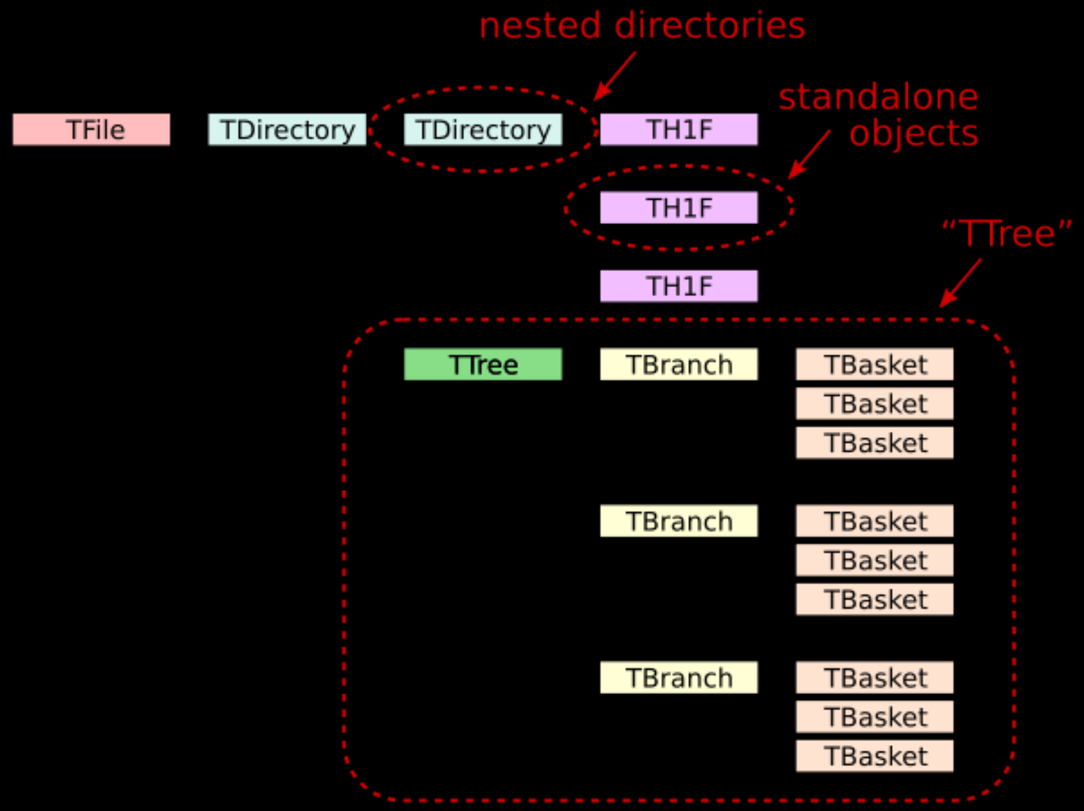
- W 3 basic ("line-mode") client
- W 3 basic server
- W 3 library of common code.

CERN's intention in this is to further compatibility, common practices, and standards in networking and computer supported collaboration. This does not constitute a precedent to be applied to any other CERN copyright software.

WWW8 1999

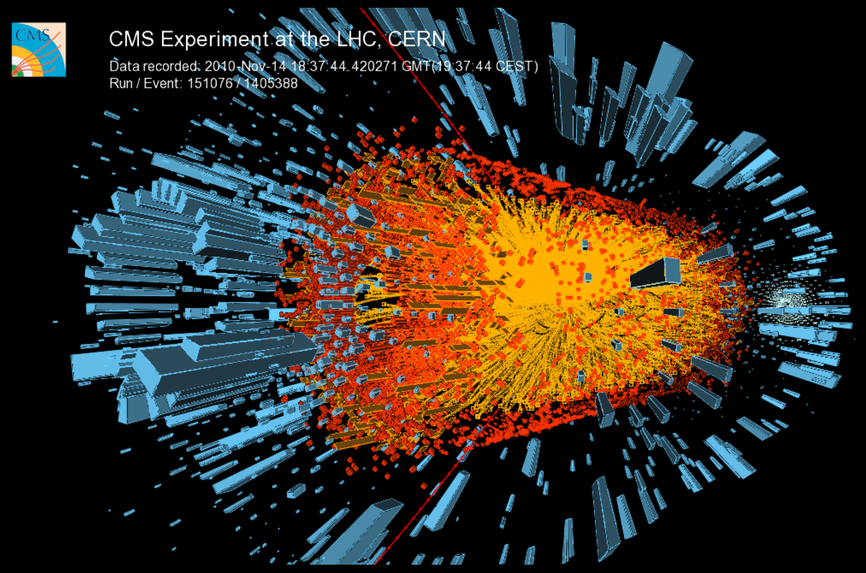
at Microsoft's conference sponsorship. There was grumbling that the delegates' pack included complementary Microsoft CD-ROMs (for Windows only). "Next year," one delegate was overheard to say, while tucking into a spring roll and sipping Chardonnay at the evening reception (courtesy of Bill Gates), "Microsoft will have bought the World Wide Web." However, his fears were not universally held. There is just too much grass-roots stuff going on out there for one company, however powerful, to take over completely.

RNtuple



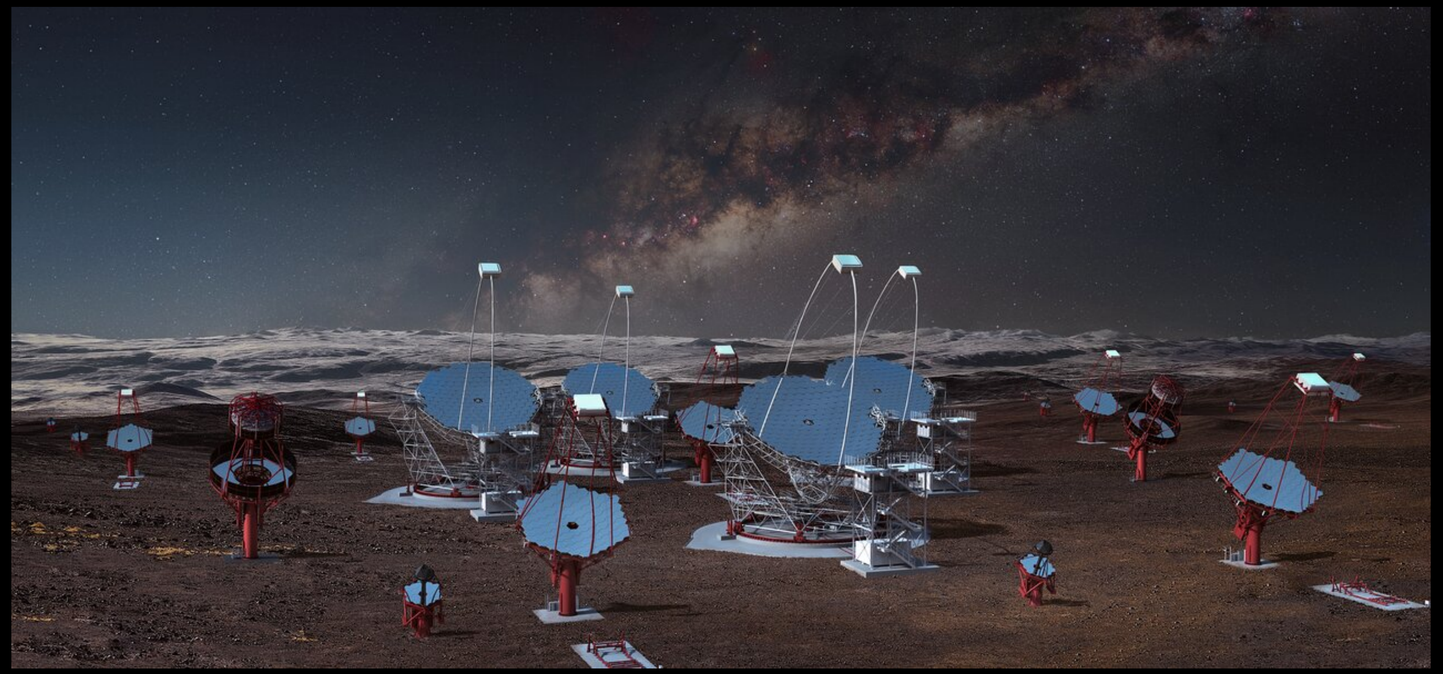
- Invented **our own binary data format**, ROOT TTree, a columnar data format developed > 25 years ago. Stores >2 exabytes of LHC data
- New format for 2031-2041 will store tens of exabytes of data
- Full data analysis stack built on top!

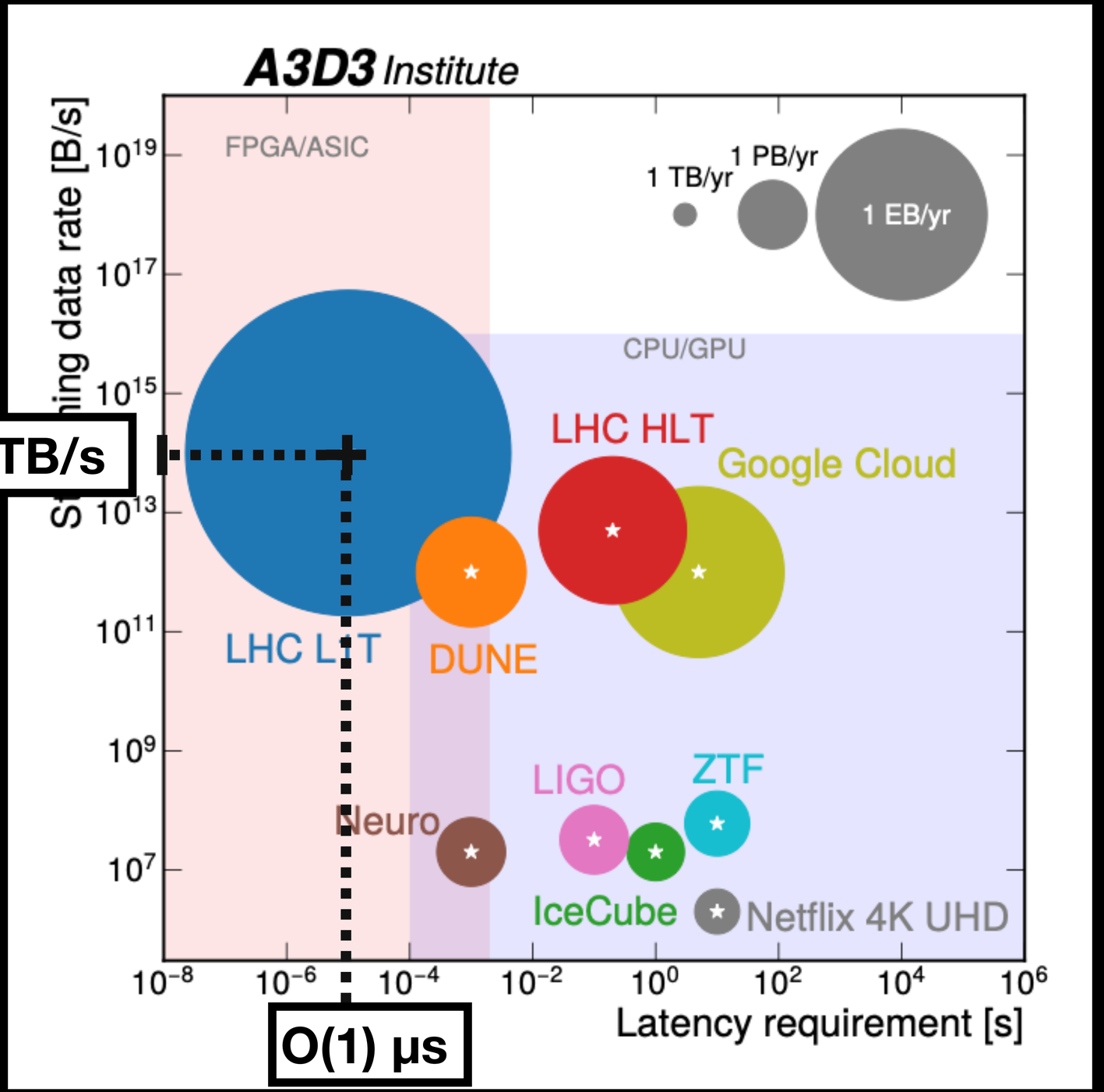
LHC: 1PB/s of proton collisions



LSST: 20 TB/night of images of night sky

CTA: 1,000 Pb/year of Cherenkov telescope data

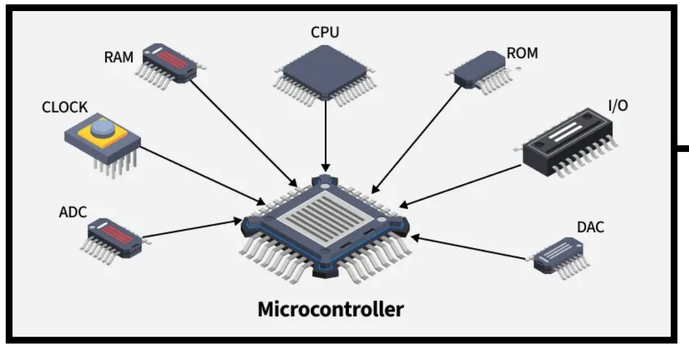




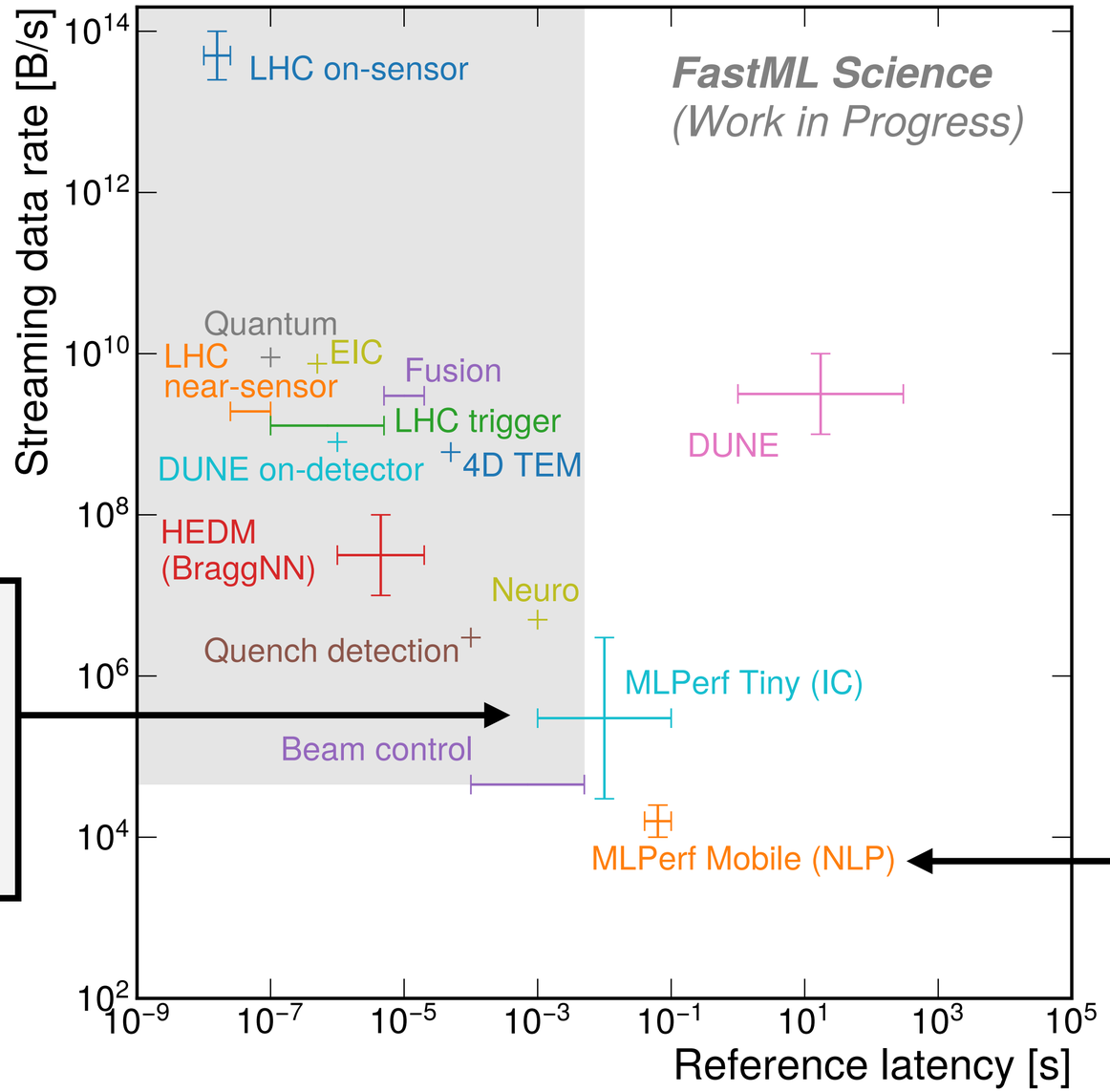
How much data goes through system

Time to process

Fast Machine Learning for HEP

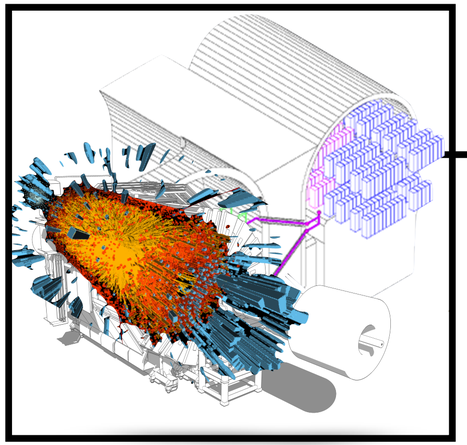


~ms latency

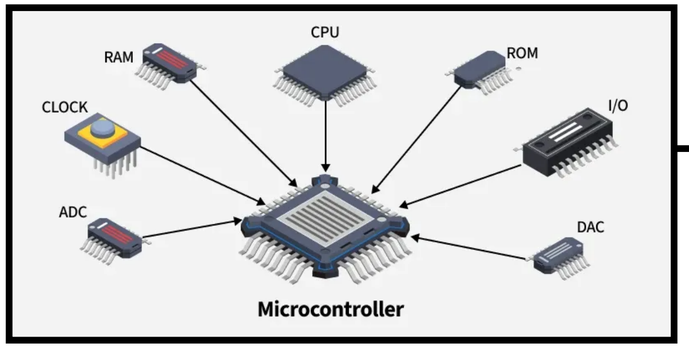


sub-second latency

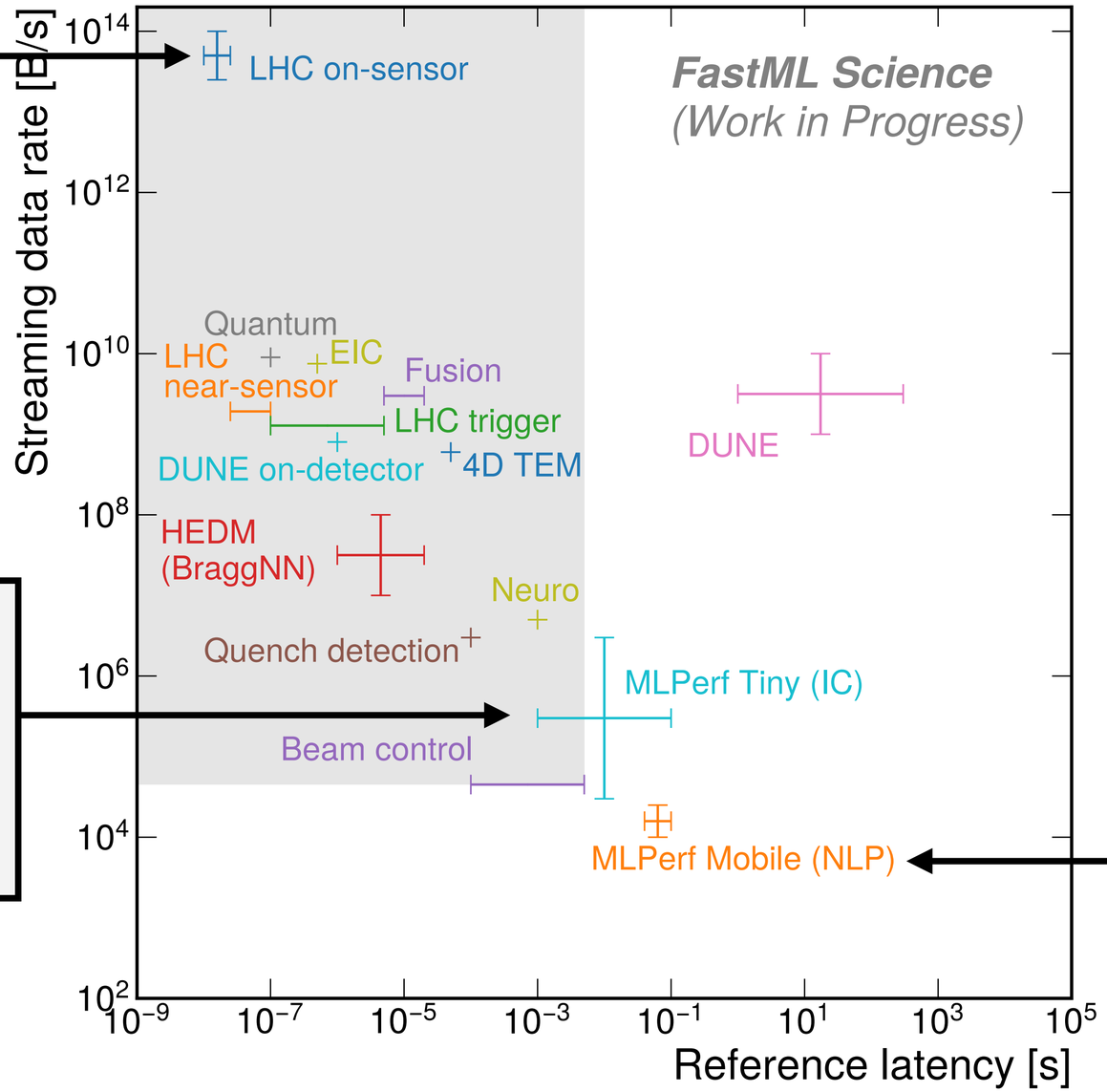
Fast Machine Learning for HEP



50,000x-10 Gbps lpGBTs

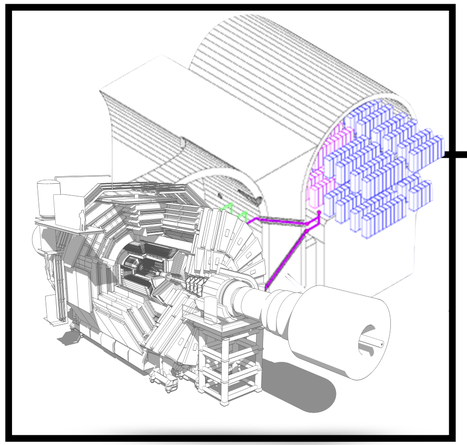


~ms latency

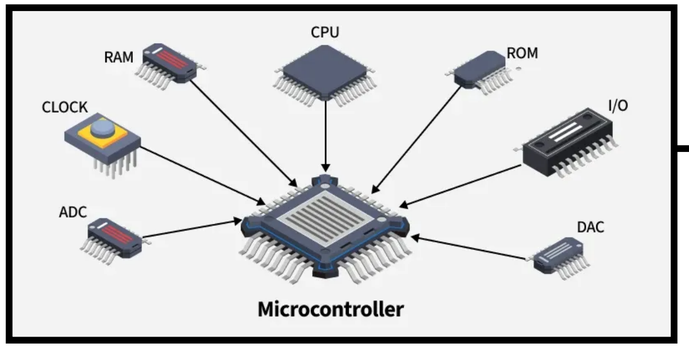


sub-second latency

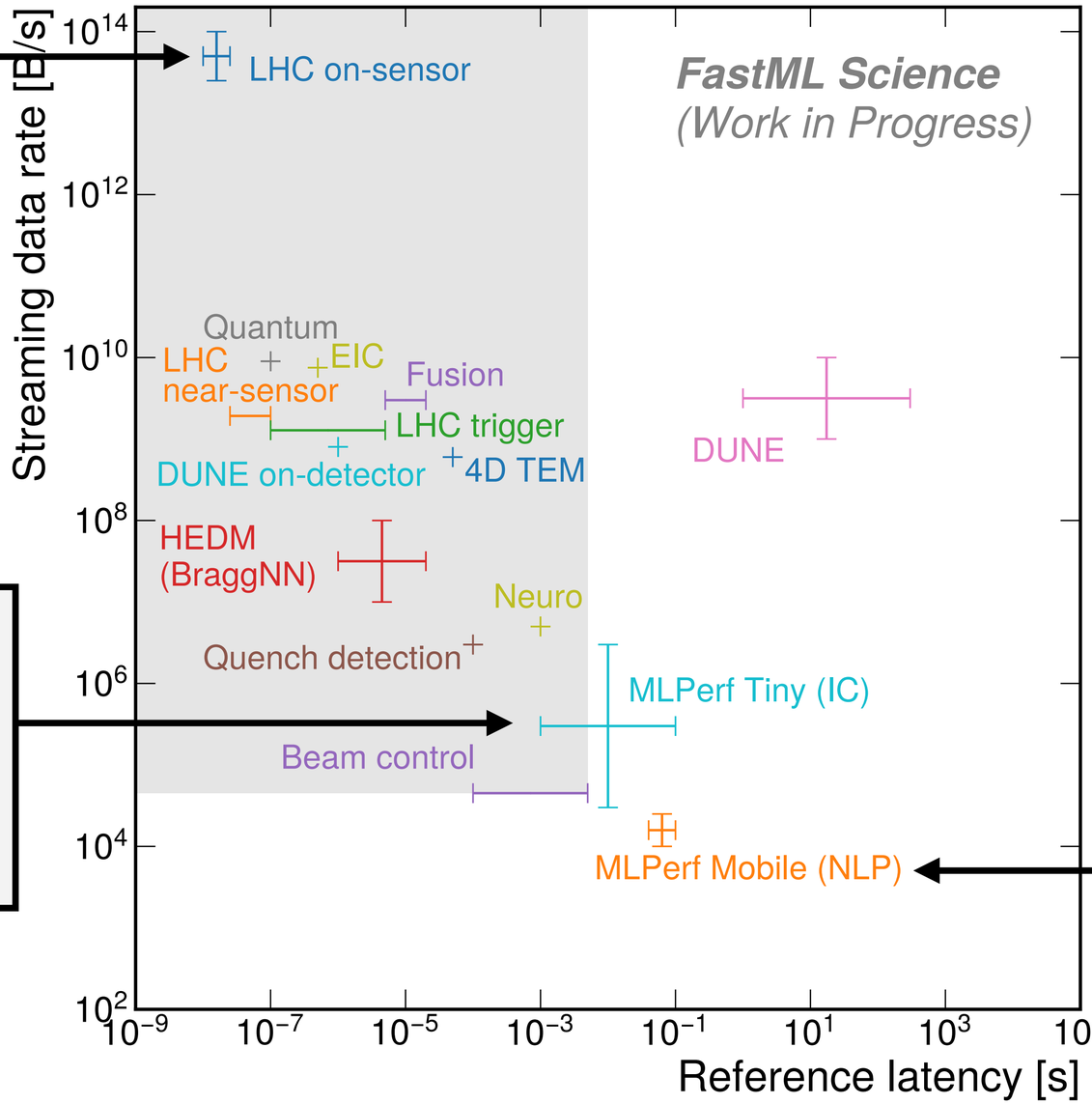
Fast Machine Learning for HEP



50,000x-10 Gbps lpGBTs



~ms latency

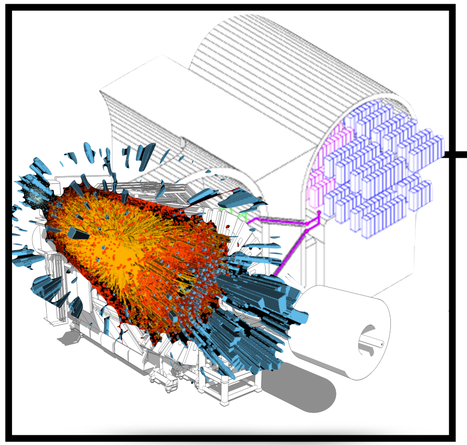


sub-second latency

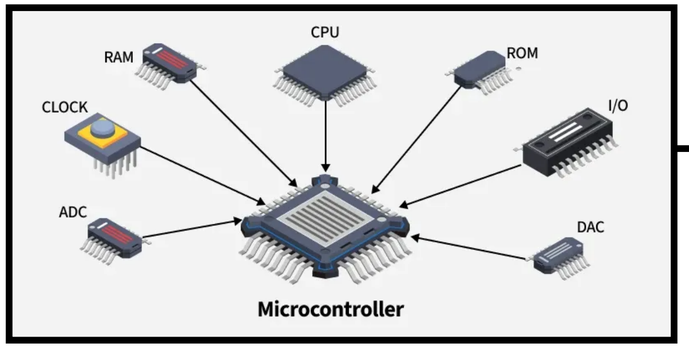
And in your future?

FCC-hh vertex detector readout ~1 PB/s

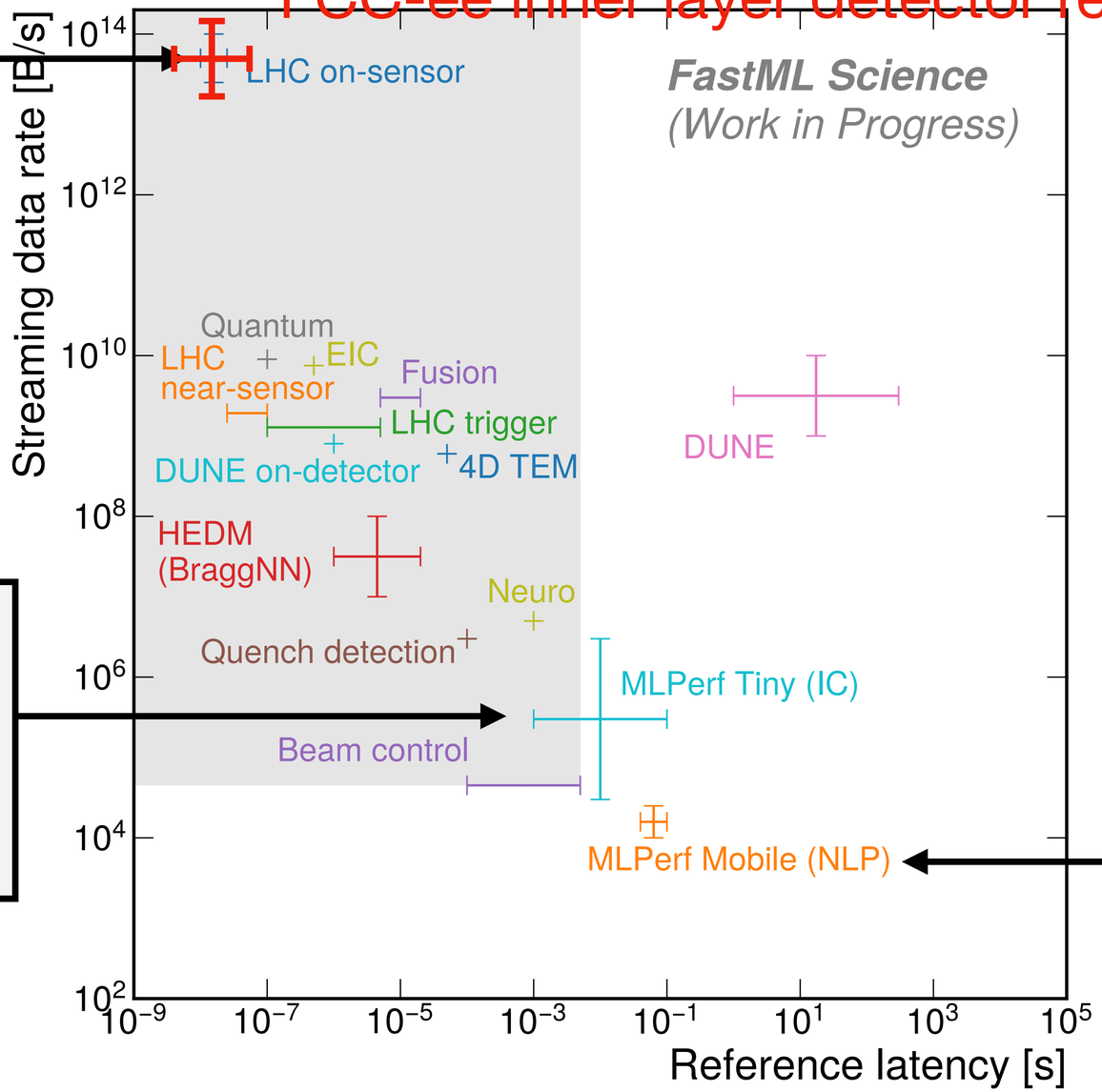
FCC-ee inner layer detector readout ~1 TB/s



50,000x10 Gbps lpGBTs



~ms latency

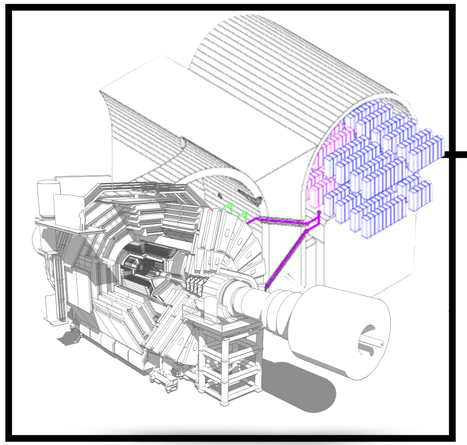


sub-second latency

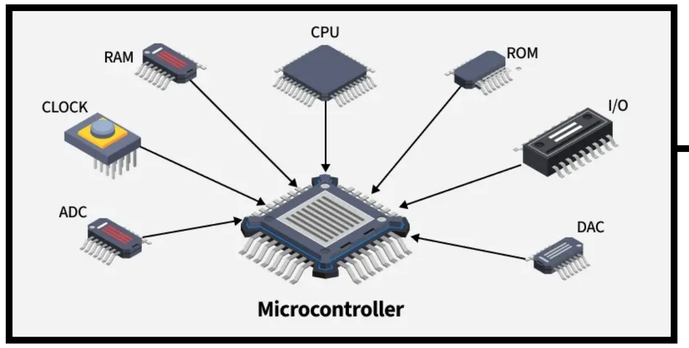
And in your future?

FCC-hh vertex detector readout ~1 PB/s

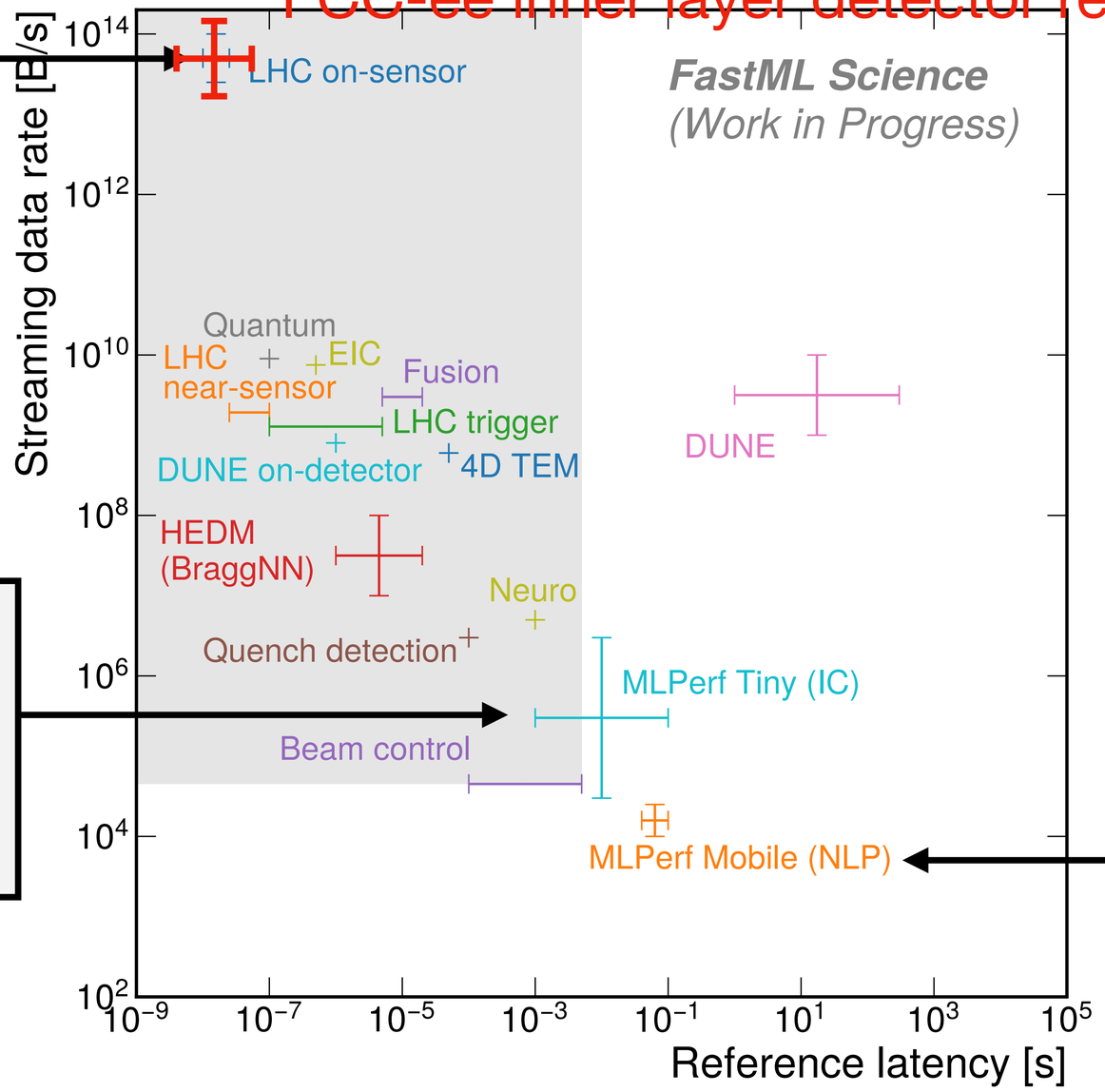
FCC-ee inner layer detector readout ~1 TB/s



50,000x10 Gbps lpGBTs

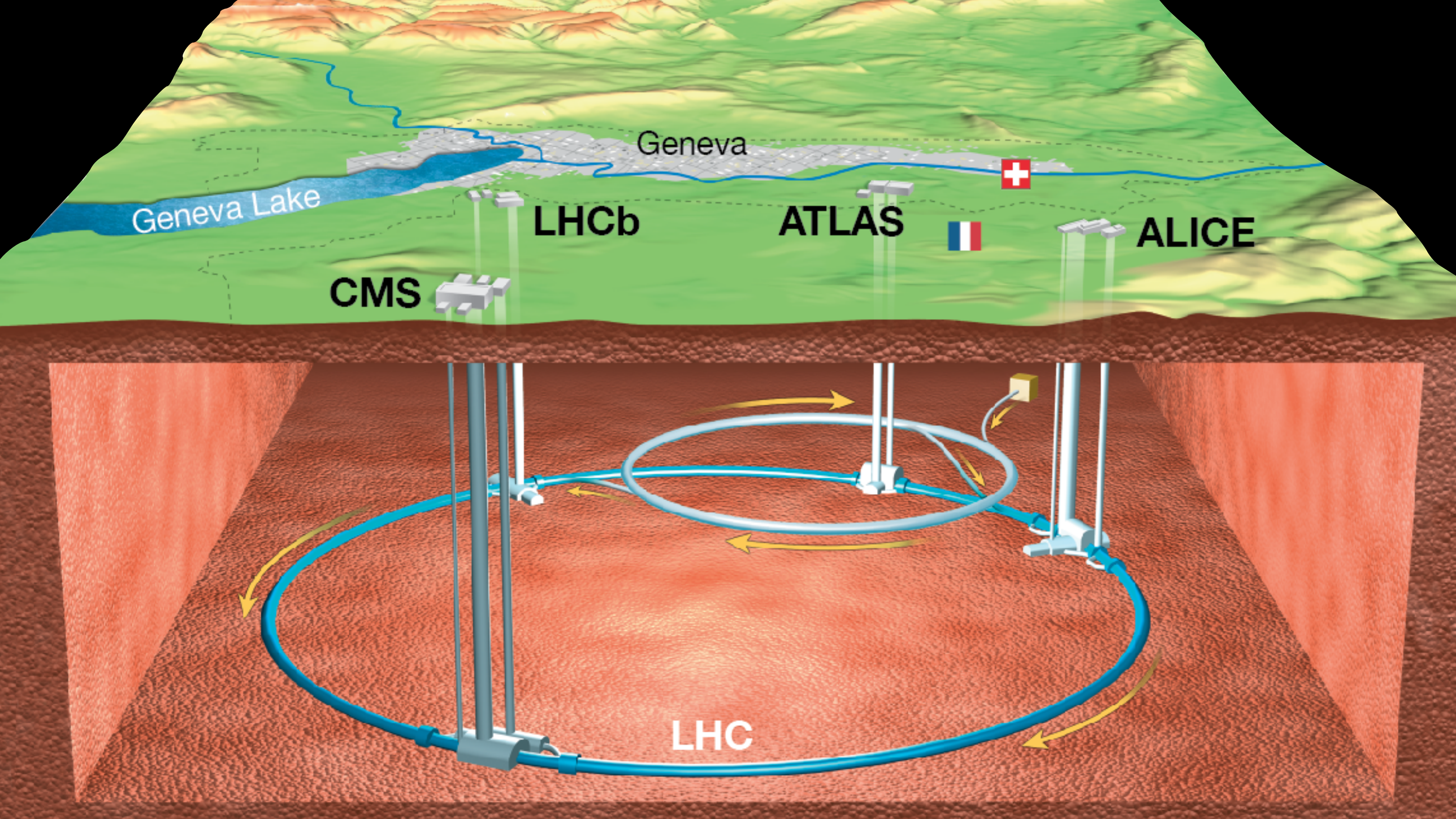


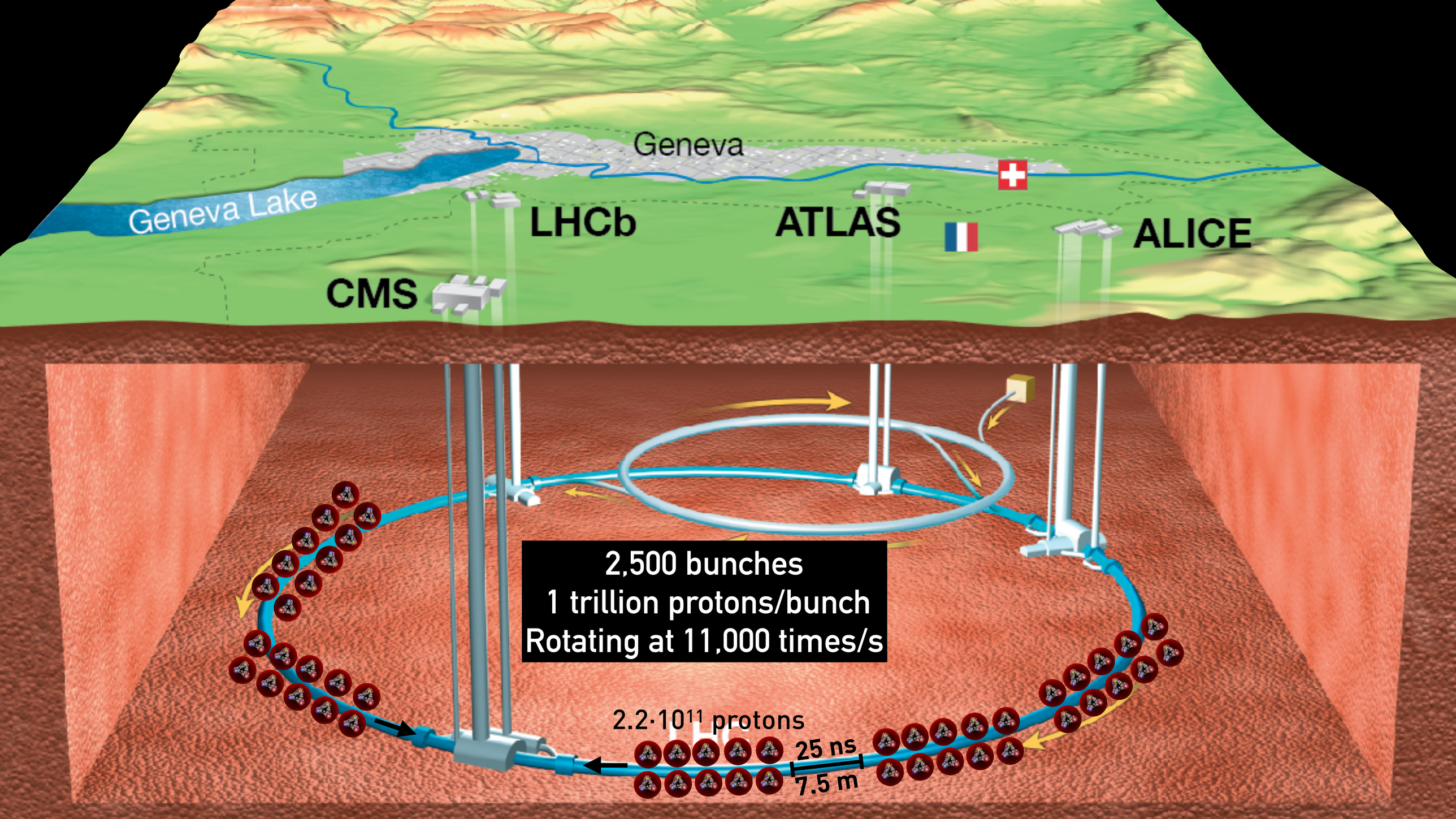
~ms latency

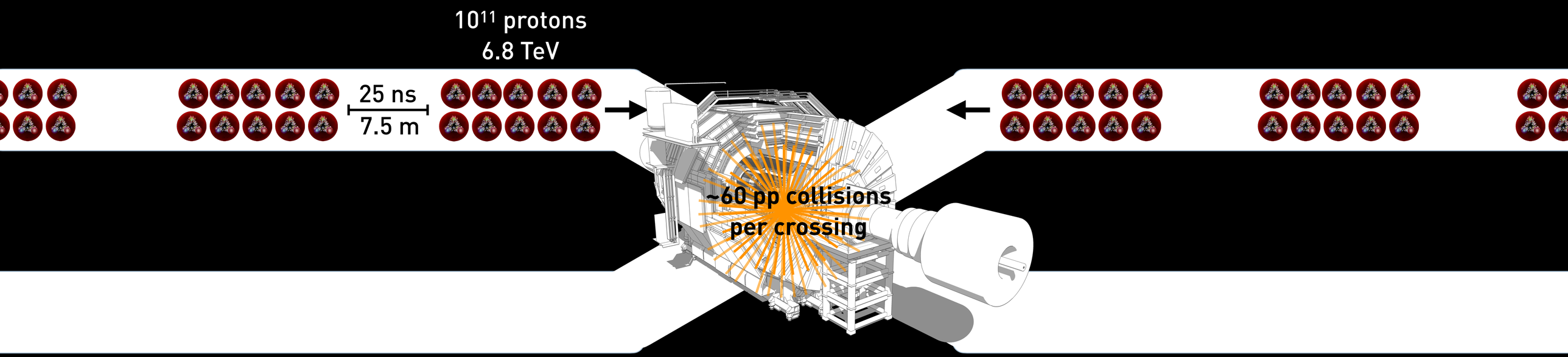


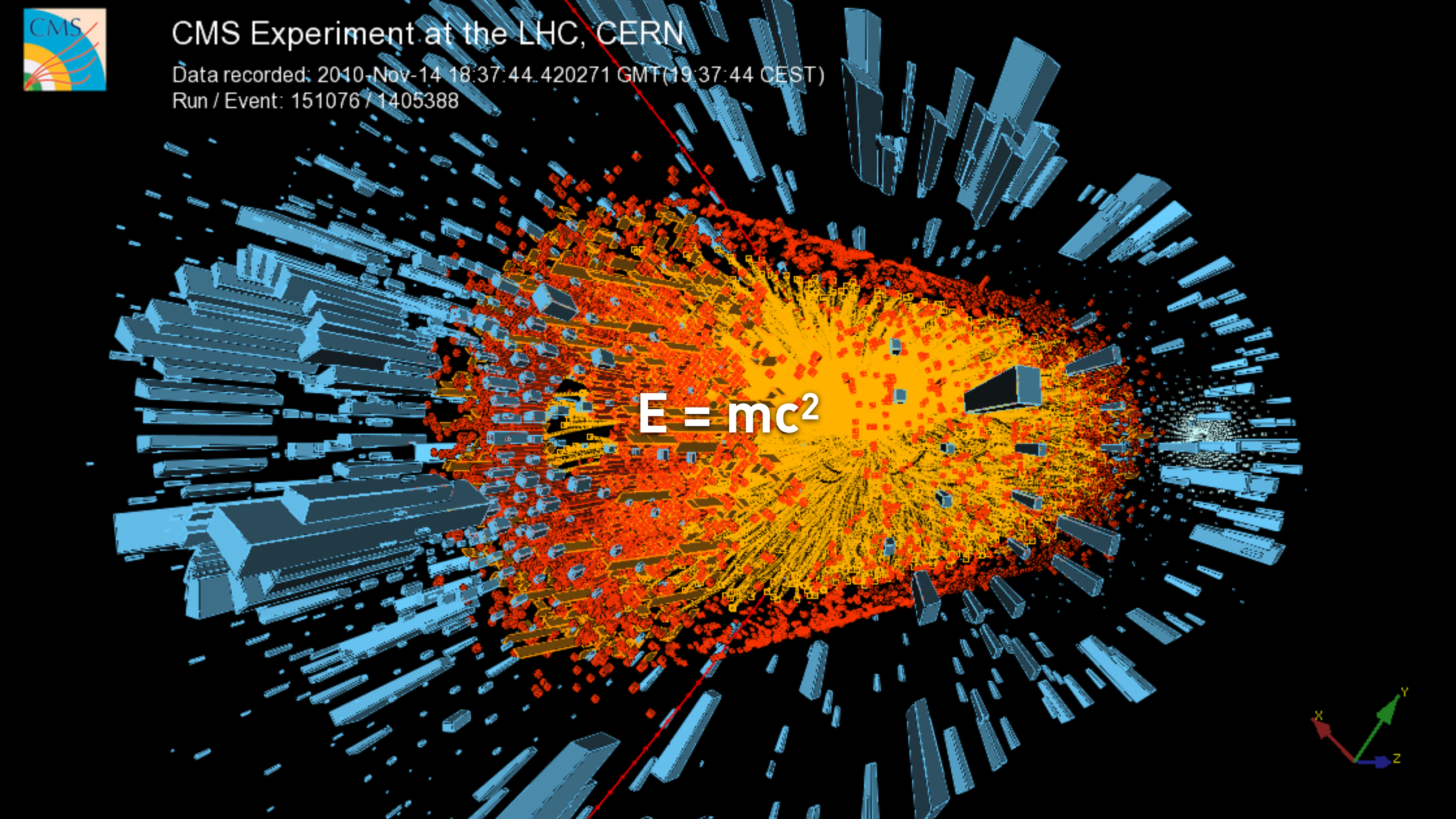
sub-second latency

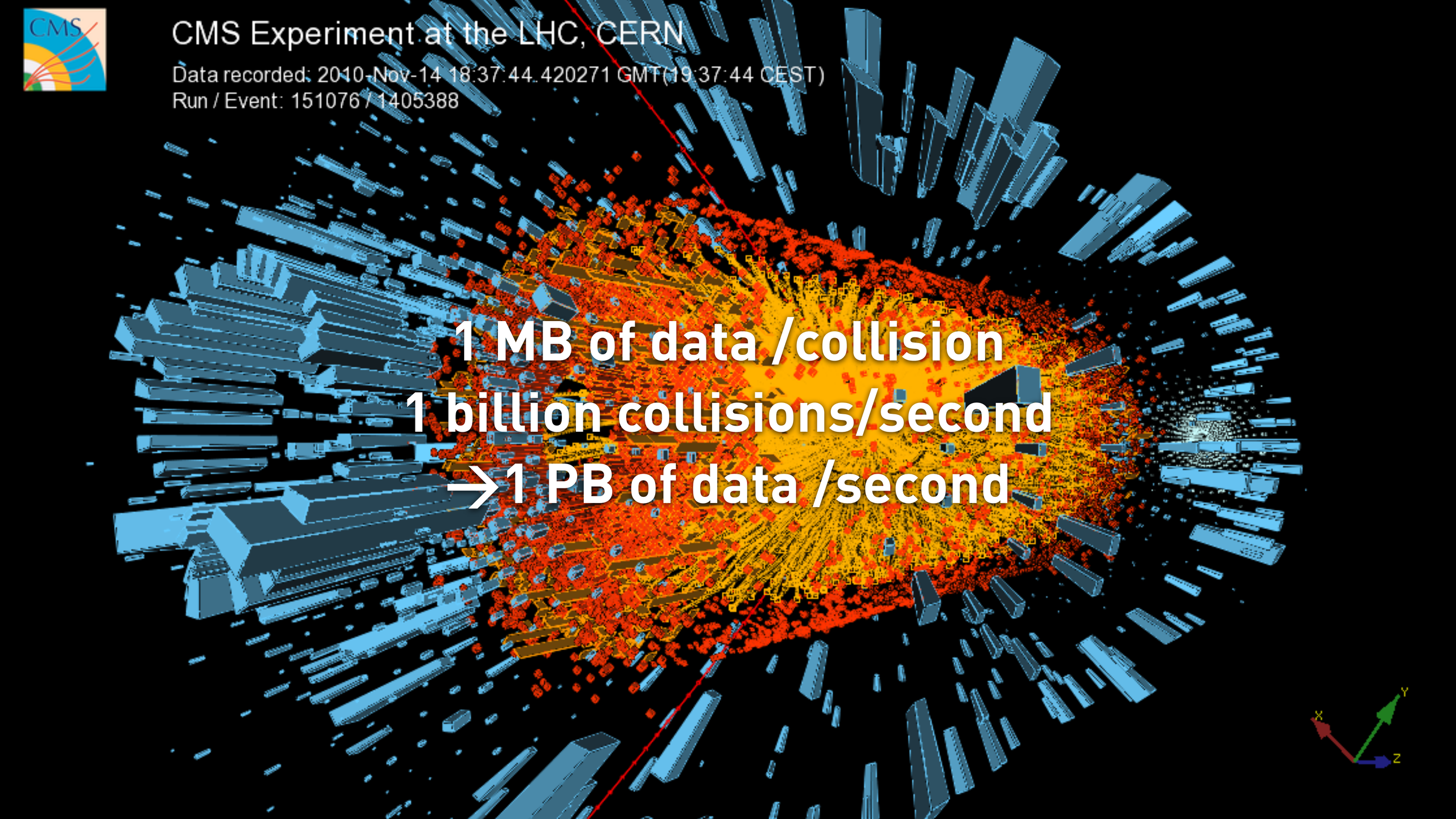
Real-time systems

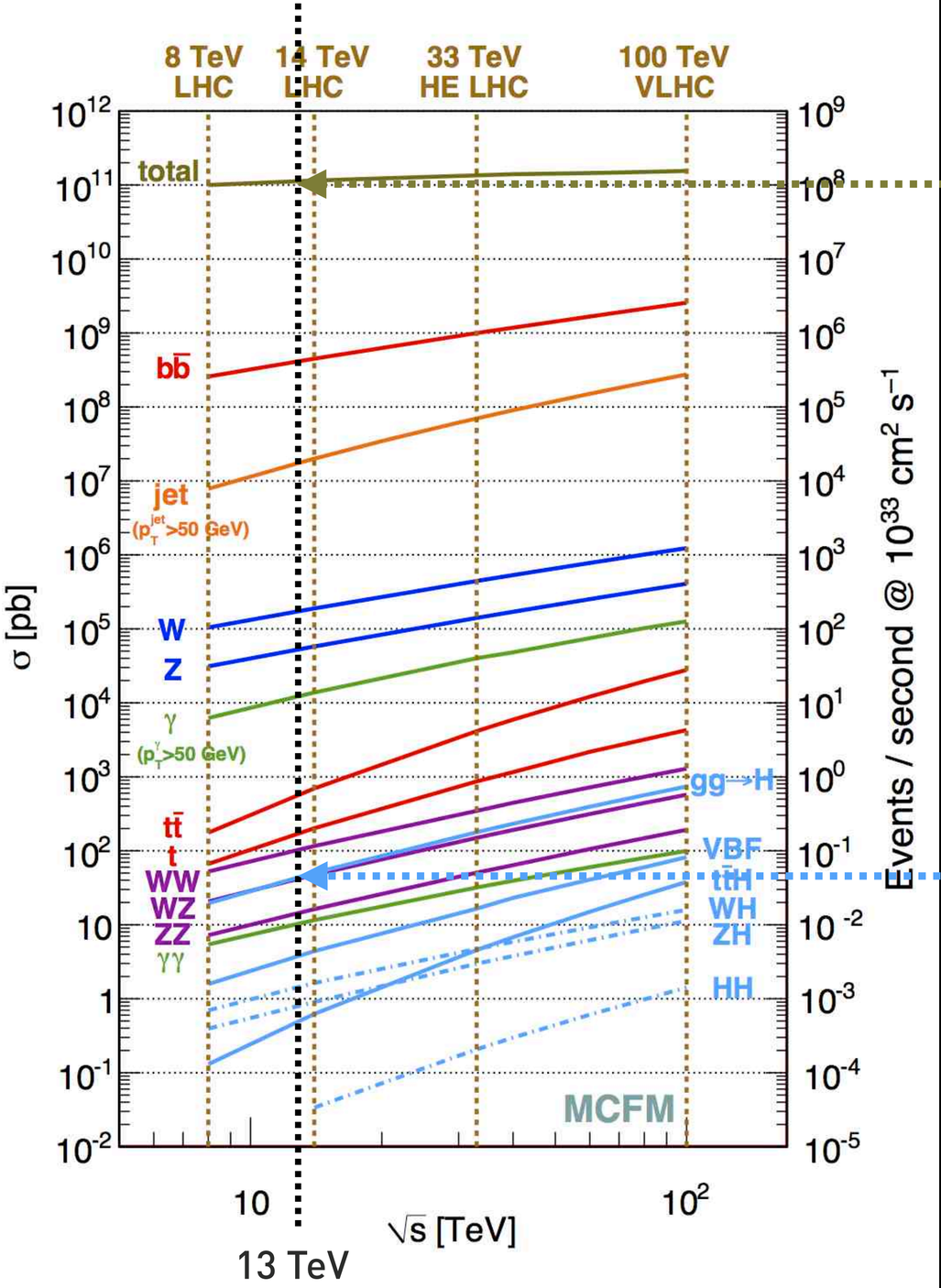

Geneva
Geneva Lake
LHCb
ATLAS
ALICE
CMS
LHC


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


10^{11} protons
6.8 TeV
25 ns
7.5 m
~60 pp collisions
per crossing


CMS Experiment at the LHC, CERN
Data recorded: 2010-Nov-14 18:37:44.420271 GMT(19:37:44 CEST)
Run / Event: 151076 / 1405388
E = mc²
Y
X
Z


CMS Experiment at the LHC, CERN
Data recorded: 2010-Nov-14 18:37:44.420271 GMT(19:37:44 CEST)
Run / Event: 151076 / 1405388
1 MB of data /collision
1 billion collisions/second
→1 PB of data /second
Y
X
Z

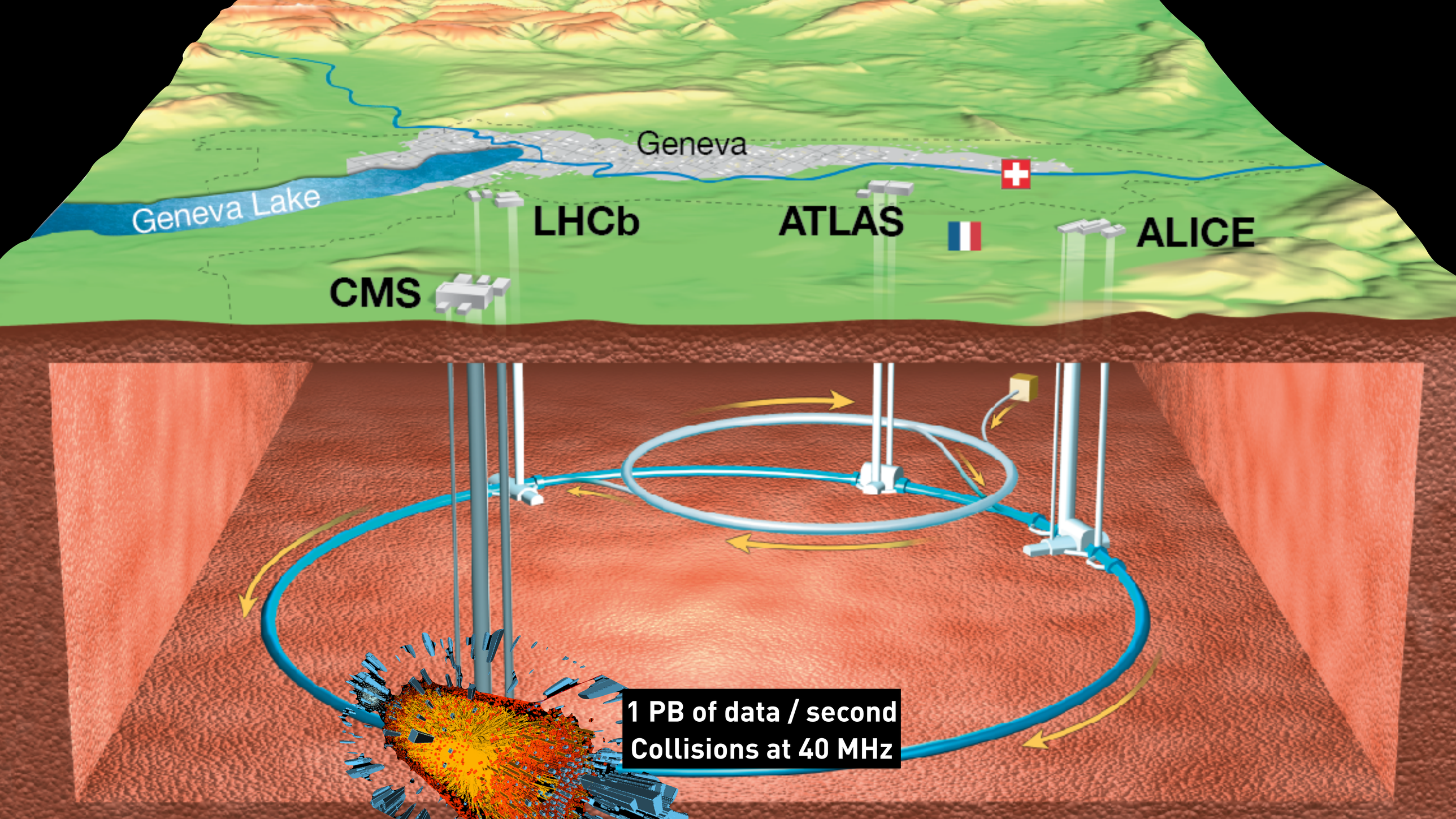


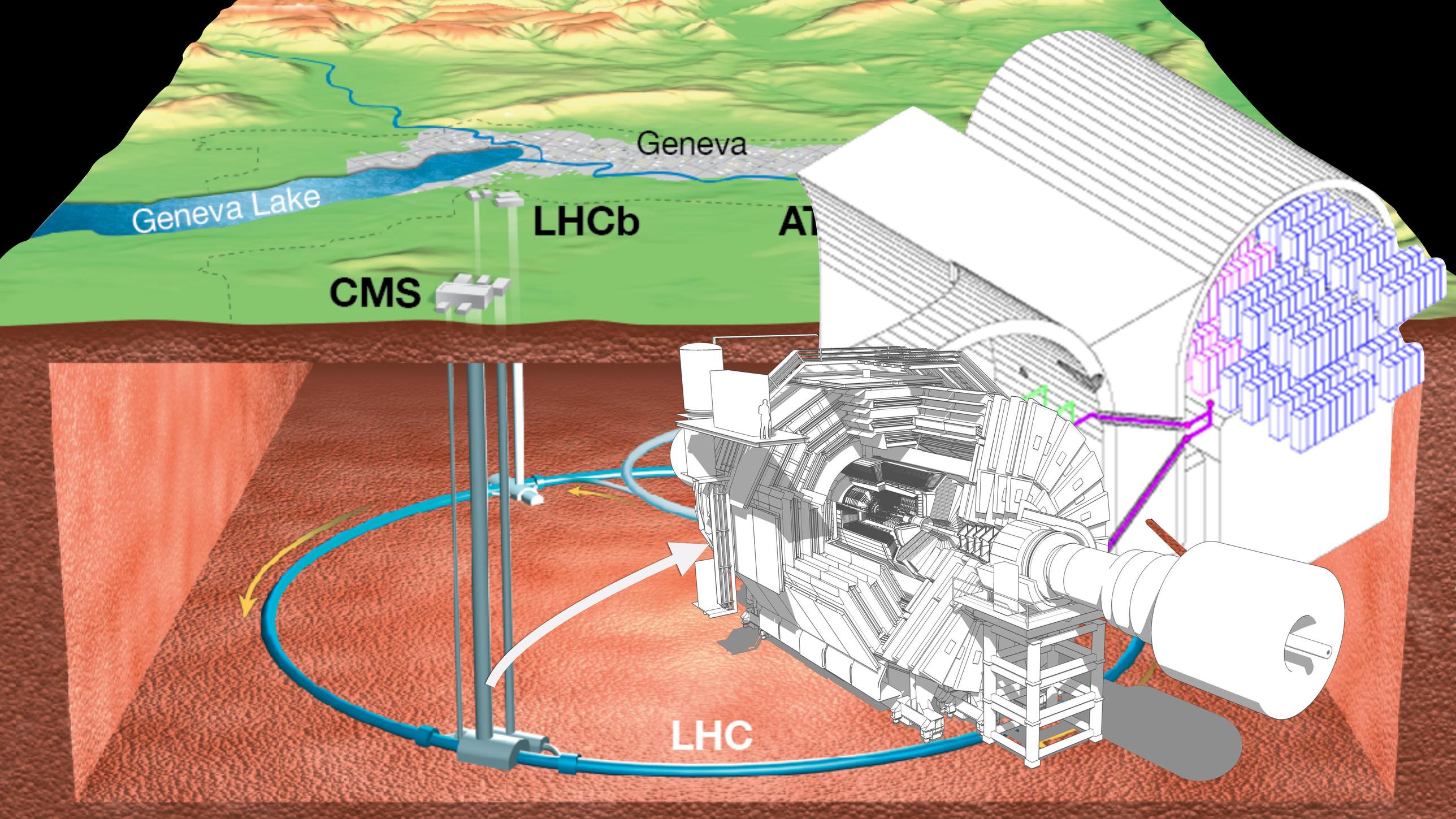
"Probability" of producing "anything"

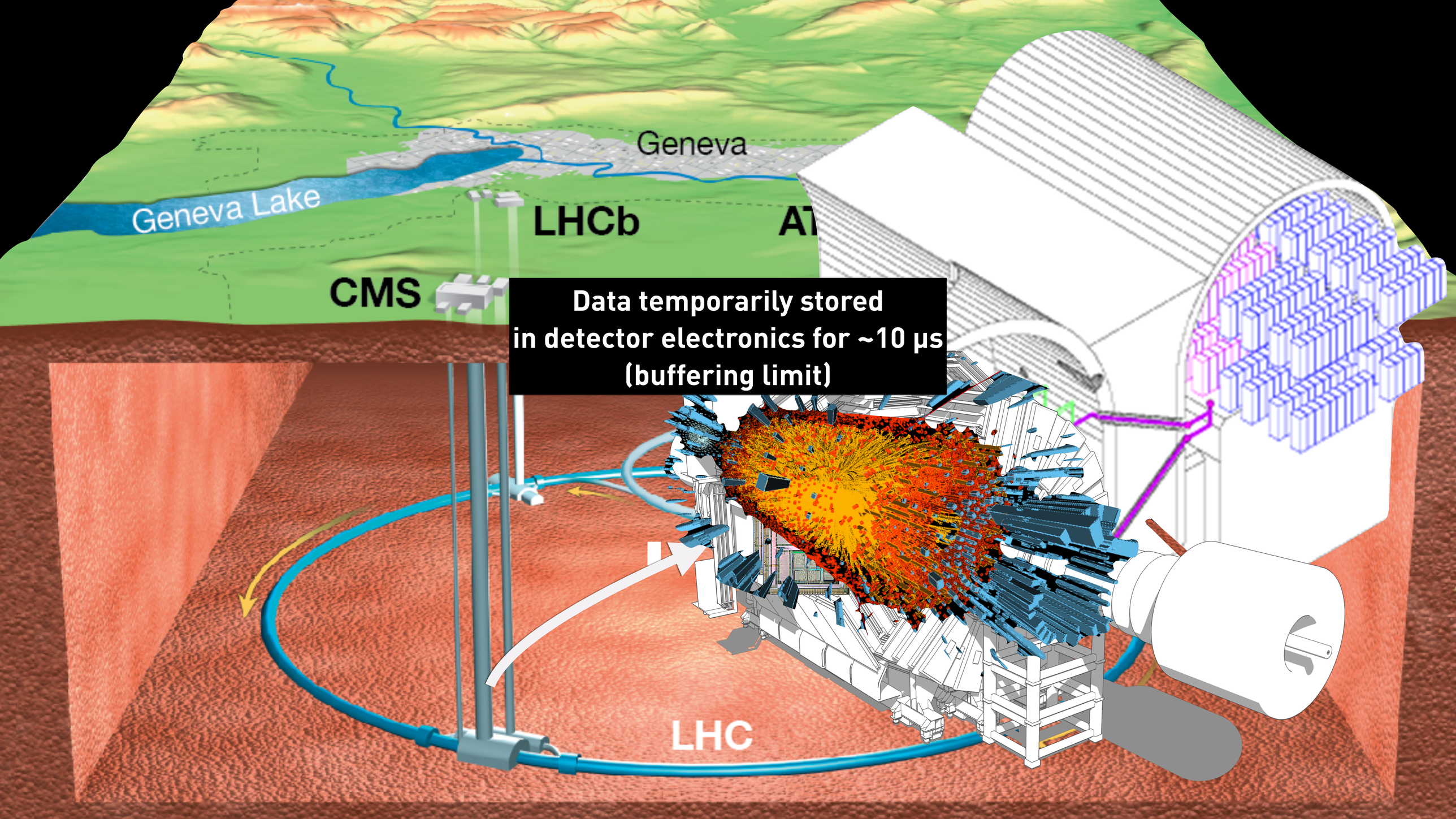
"Probability" of producing a Higgs

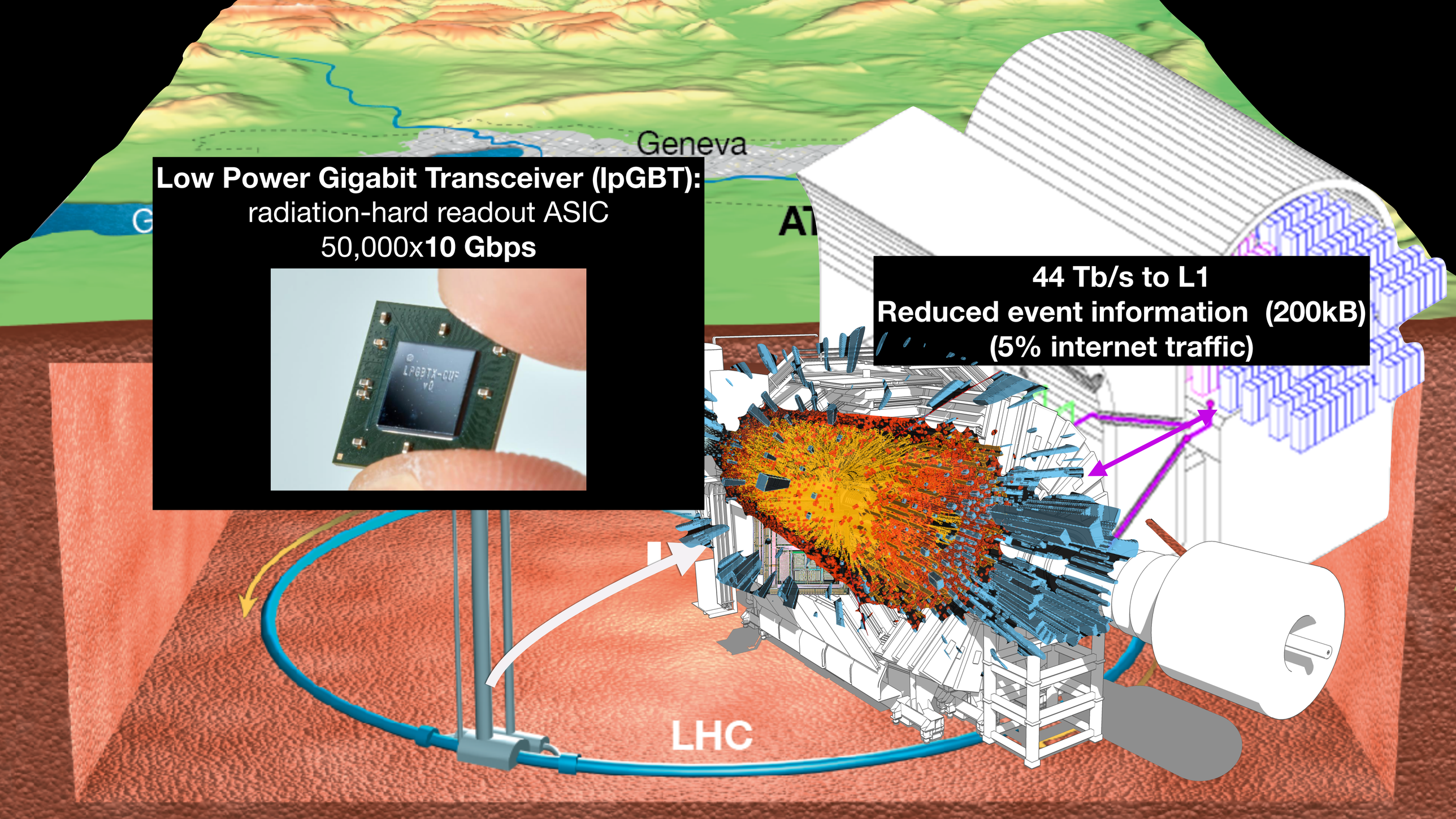
Higgs produced
~1 in a billion collisions!

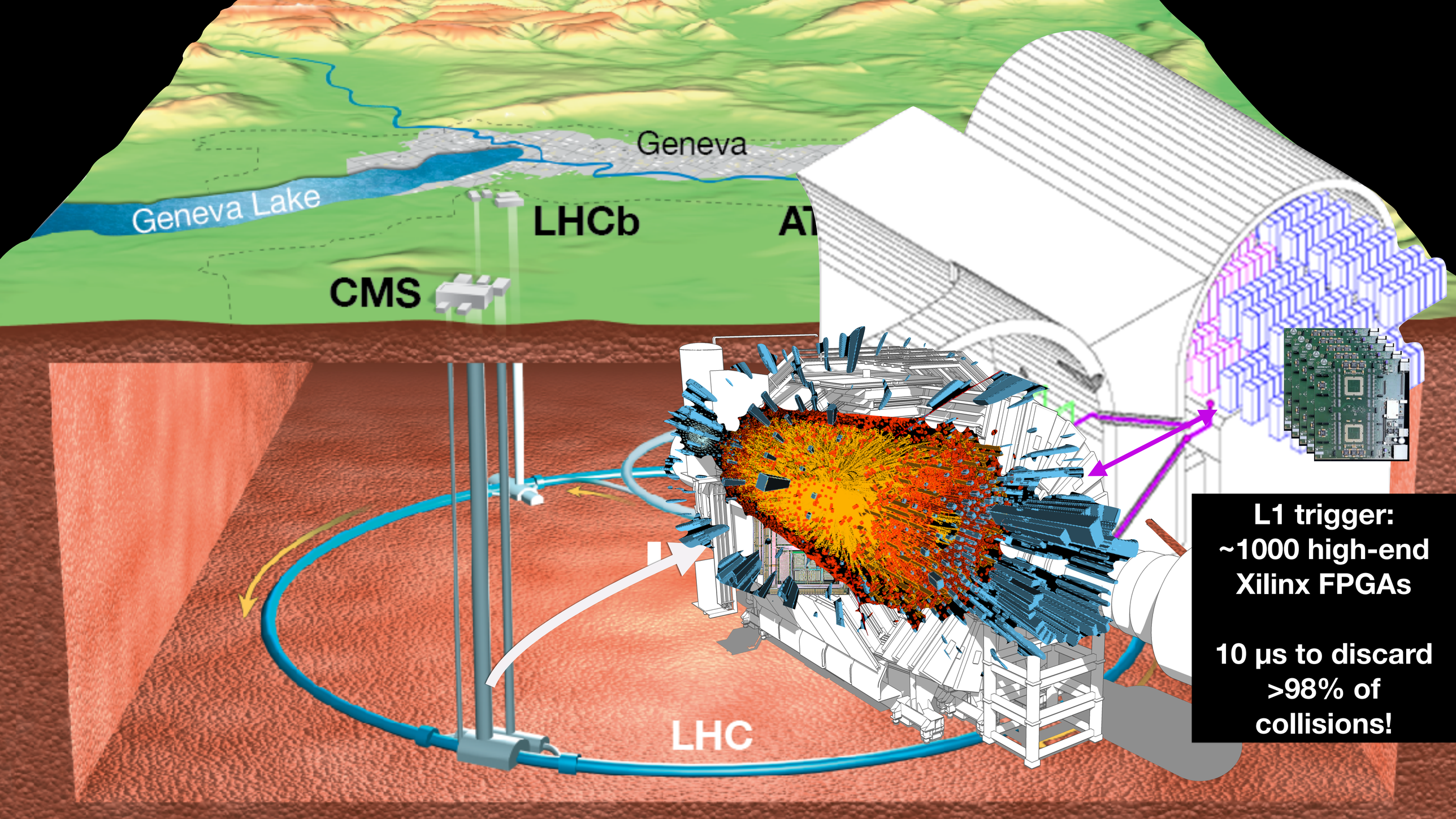
Saving all collisions not useful
(even if we could)!

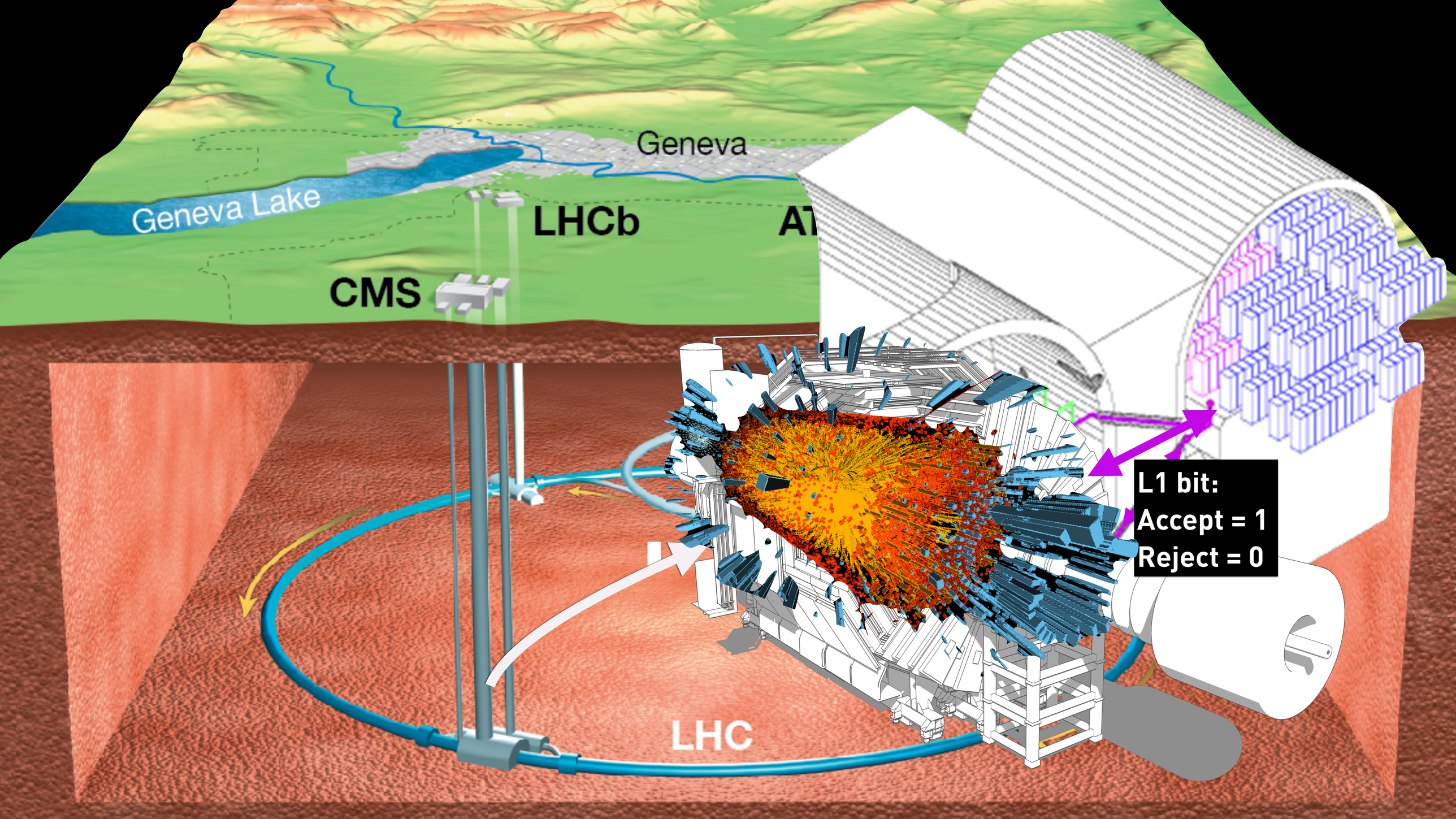

Geneva
Geneva Lake
LHCb
ATLAS
ALICE
CMS
1 PB of data / second
Collisions at 40 MHz

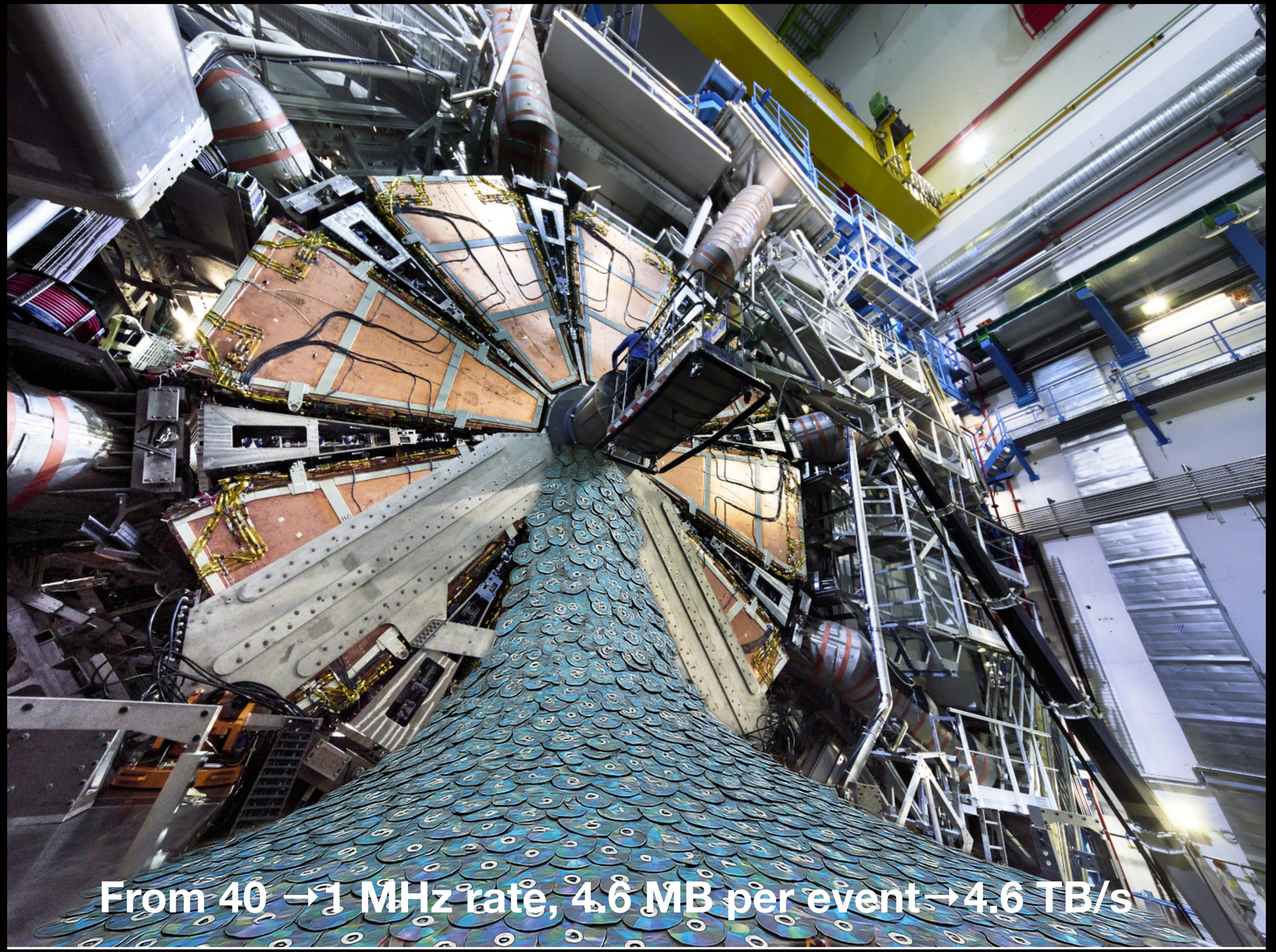

Geneva
Geneva Lake
LHCb
CMS
LHC

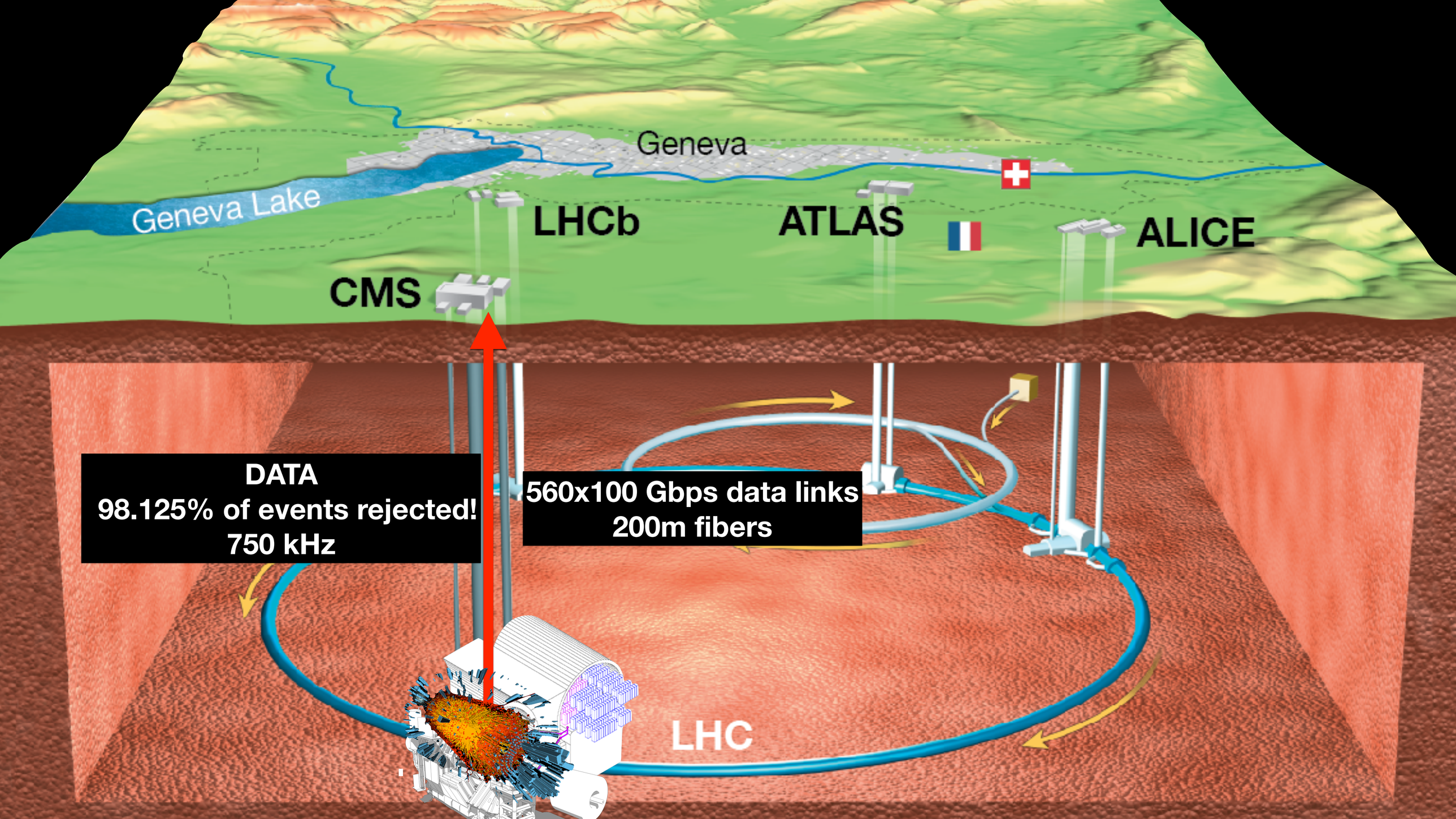

Geneva
Geneva Lake
LHCb
CMS
Data temporarily stored
in detector electronics for ~10 µs
(buffering limit)
LHC

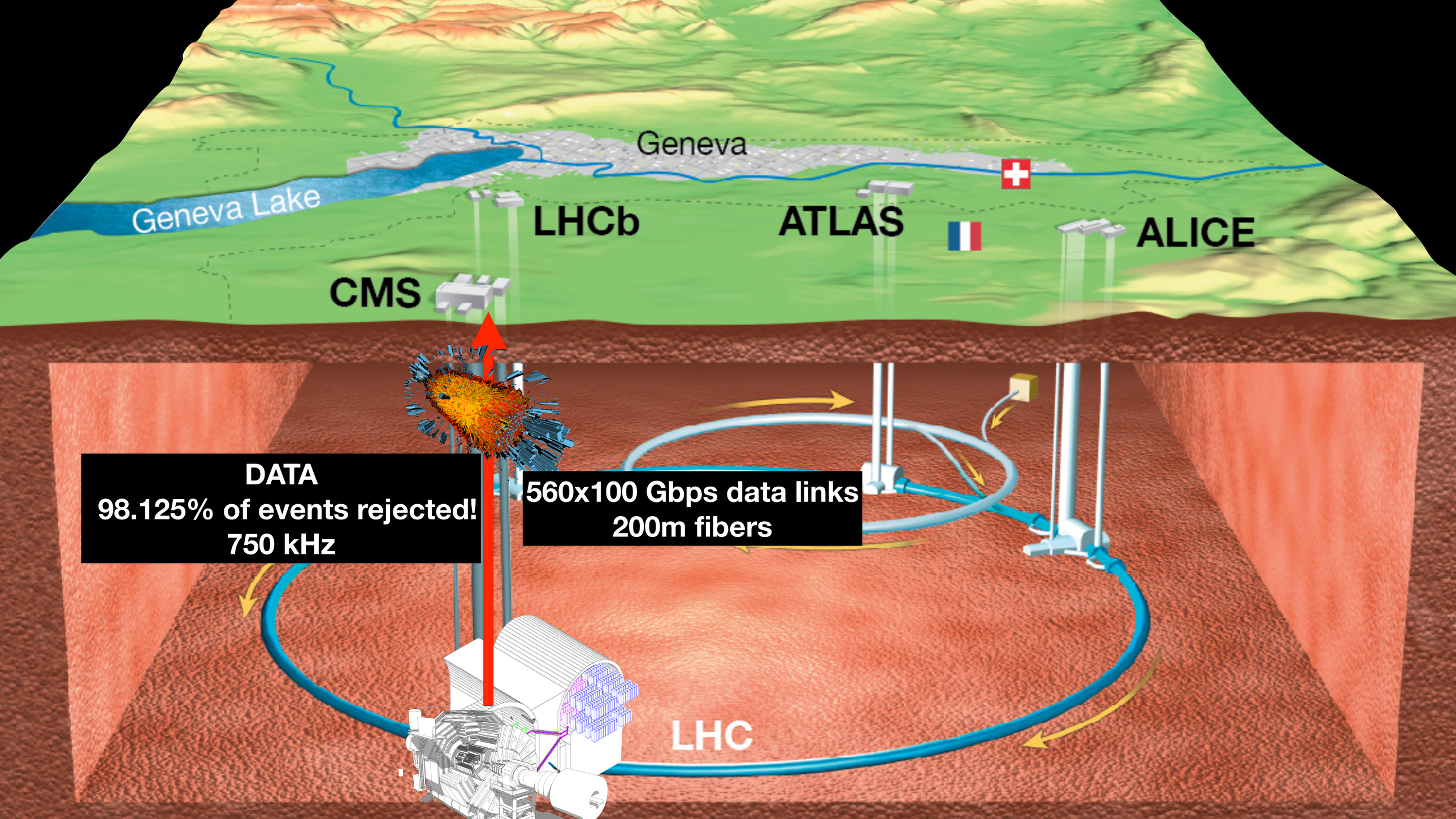

Geneva
Low Power Gigabit Transceiver (lpGBT):
radiation-hard readout ASIC
50,000x10 Gbps
LPGBTX-CUF
v0
44 Tb/s to L1
Reduced event information (200kB)
(5% internet traffic)
LHC

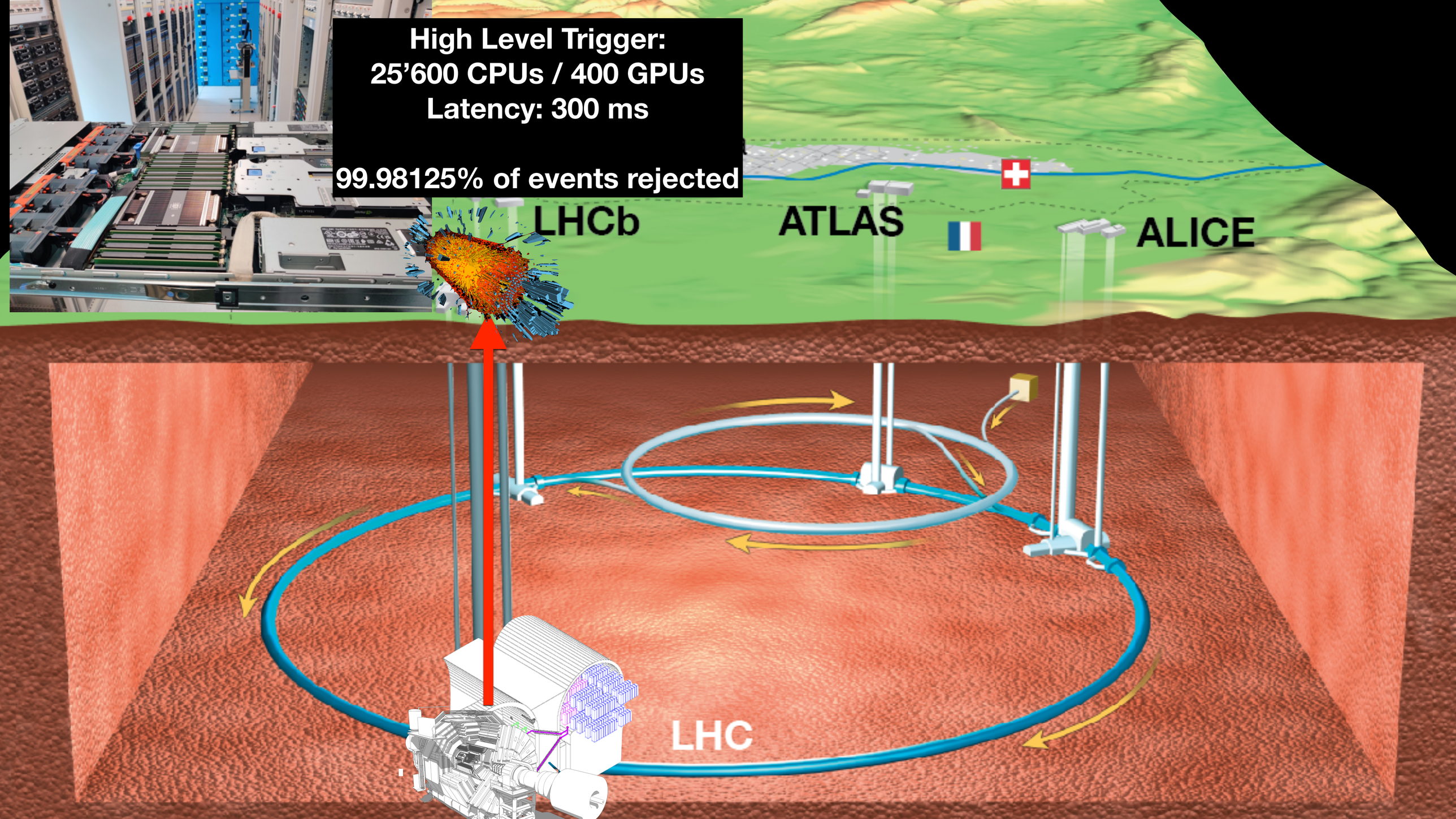

Geneva
Geneva Lake
LHCb
CMS
LHC
L1 trigger:
~1000 high-end
Xilinx FPGAs
10 µs to discard
>98% of
collisions!

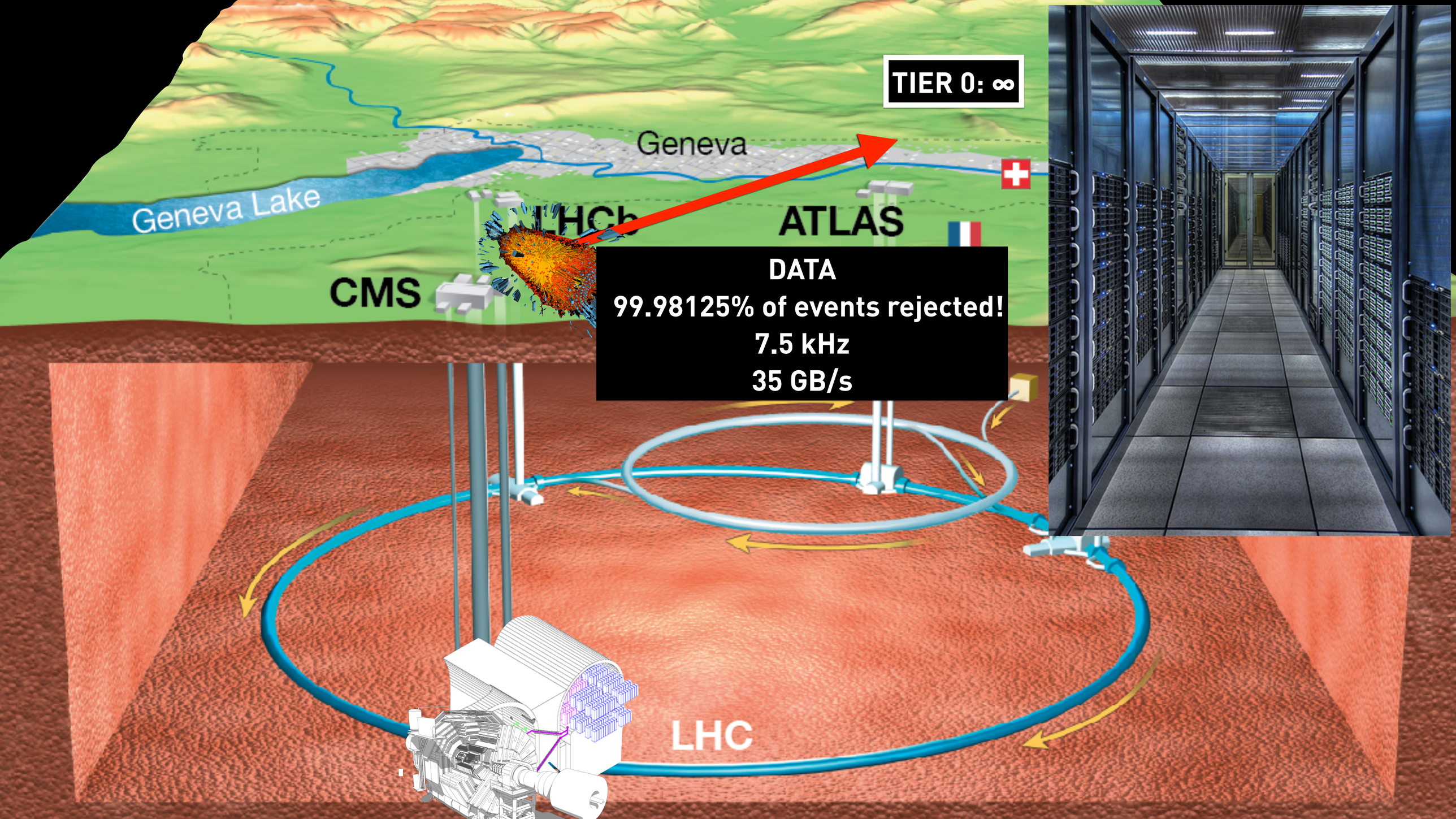

Geneva
Geneva Lake
LHCb
CMS
LHC
L1 bit:
Accept = 1
Reject = 0

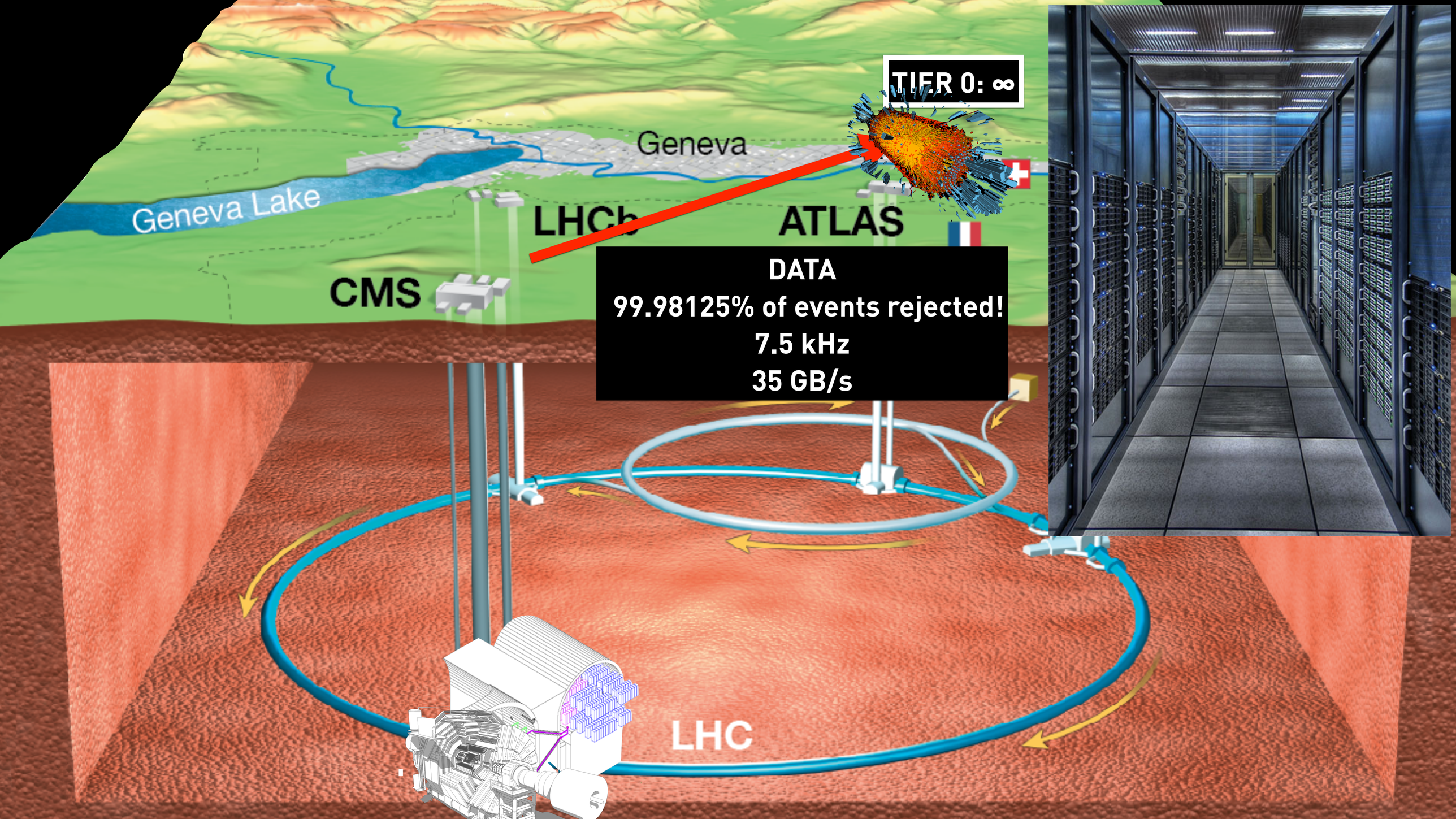

From 40 → 1 MHz rate, 4.6 MB per event → 4.6 TB/s

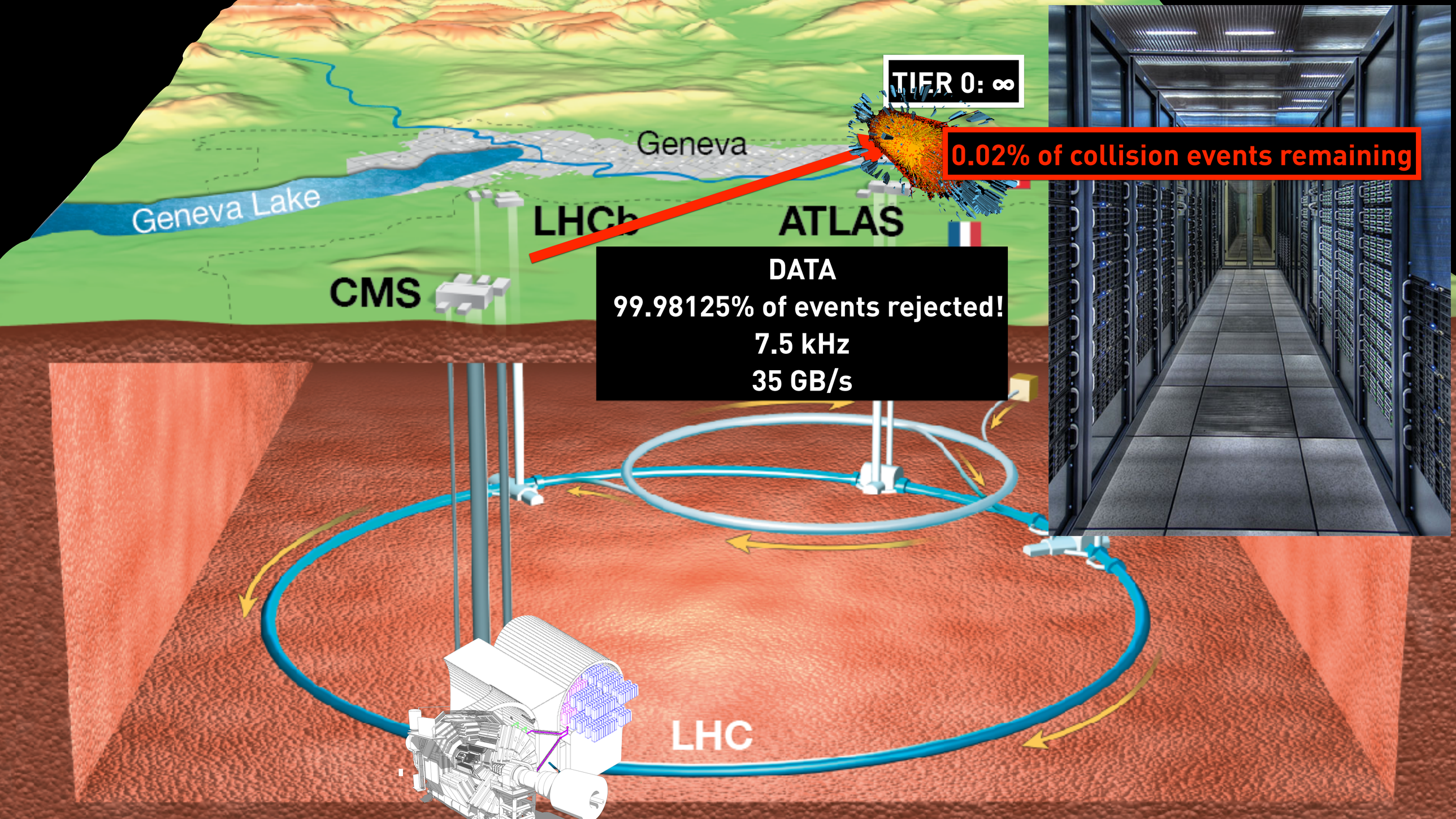

Geneva
Geneva Lake
LHCb
ATLAS
ALICE
CMS
DATA
98.125% of events rejected!
750 kHz
560x100 Gbps data links
200m fibers
LHC


Geneva
Geneva Lake
LHCb
ATLAS
ALICE
CMS
DATA
98.125% of events rejected!
750 kHz
560x100 Gbps data links
200m fibers
LHC


High Level Trigger:
25’600 CPUs / 400 GPUs
Latency: 300 ms
99.98125% of events rejected
LHCb
ATLAS
ALICE
LHC

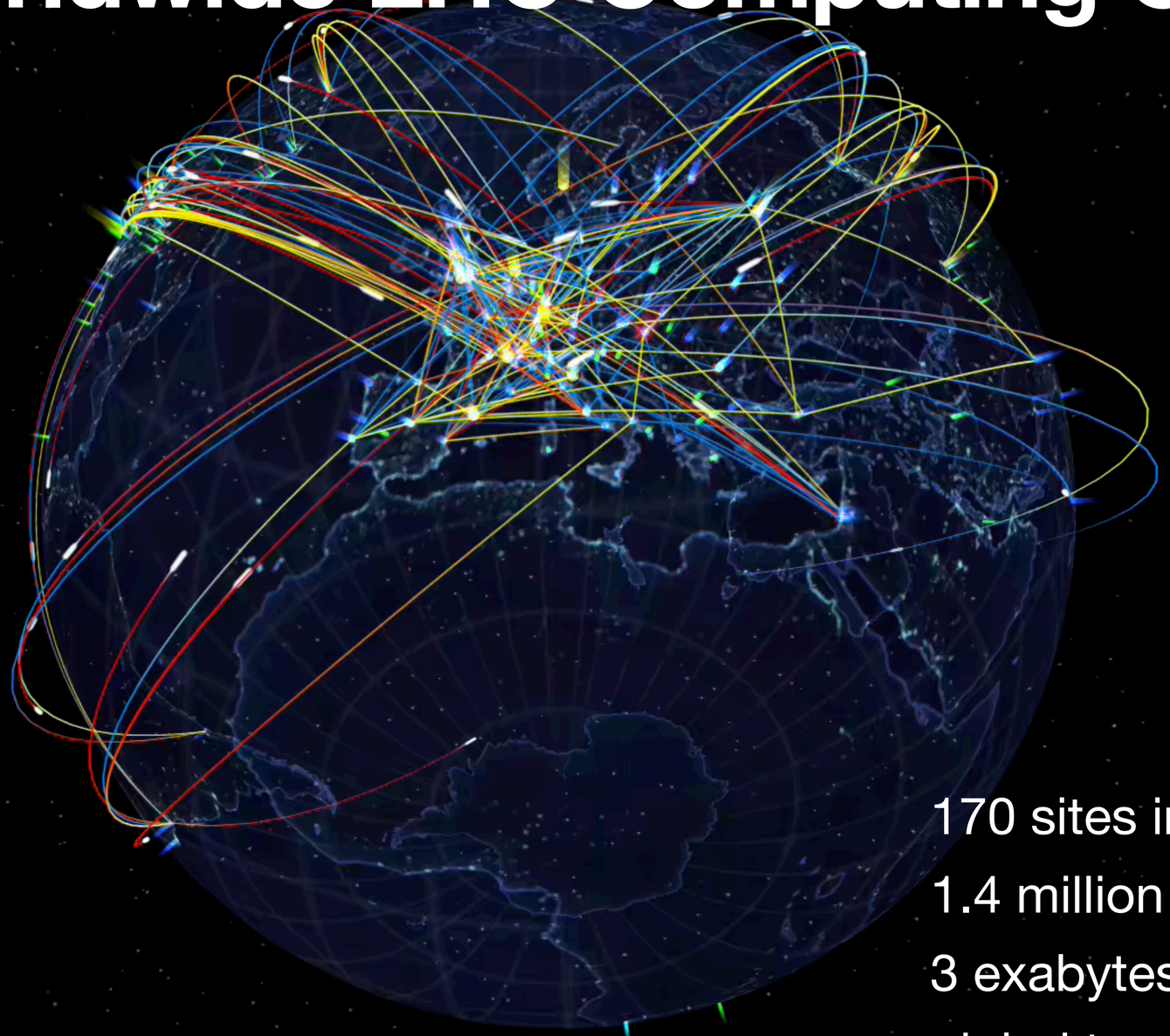

TIER 0: ∞
Geneva
Geneva Lake
LHCb
ATLAS
CMS
DATA
99.98125% of events rejected!
7.5 kHz
35 GB/s
LHC


TIER 0: ∞
Geneva
Geneva Lake
LHCb
ATLAS
CMS
DATA
99.98125% of events rejected!
7.5 kHz
35 GB/s
LHC


TIER 0: ∞
0.02% of collision events remaining
Geneva
Geneva Lake
LHCb
ATLAS
CMS
DATA
99.98125% of events rejected!
7.5 kHz
35 GB/s
LHC

And what happens to all that data once we decide to keep it?

The Worldwide LHC Computing Grid



Running jobs: 365644
Active CPU cores: 807139
Transfer rate: 21.54 GiB/sec

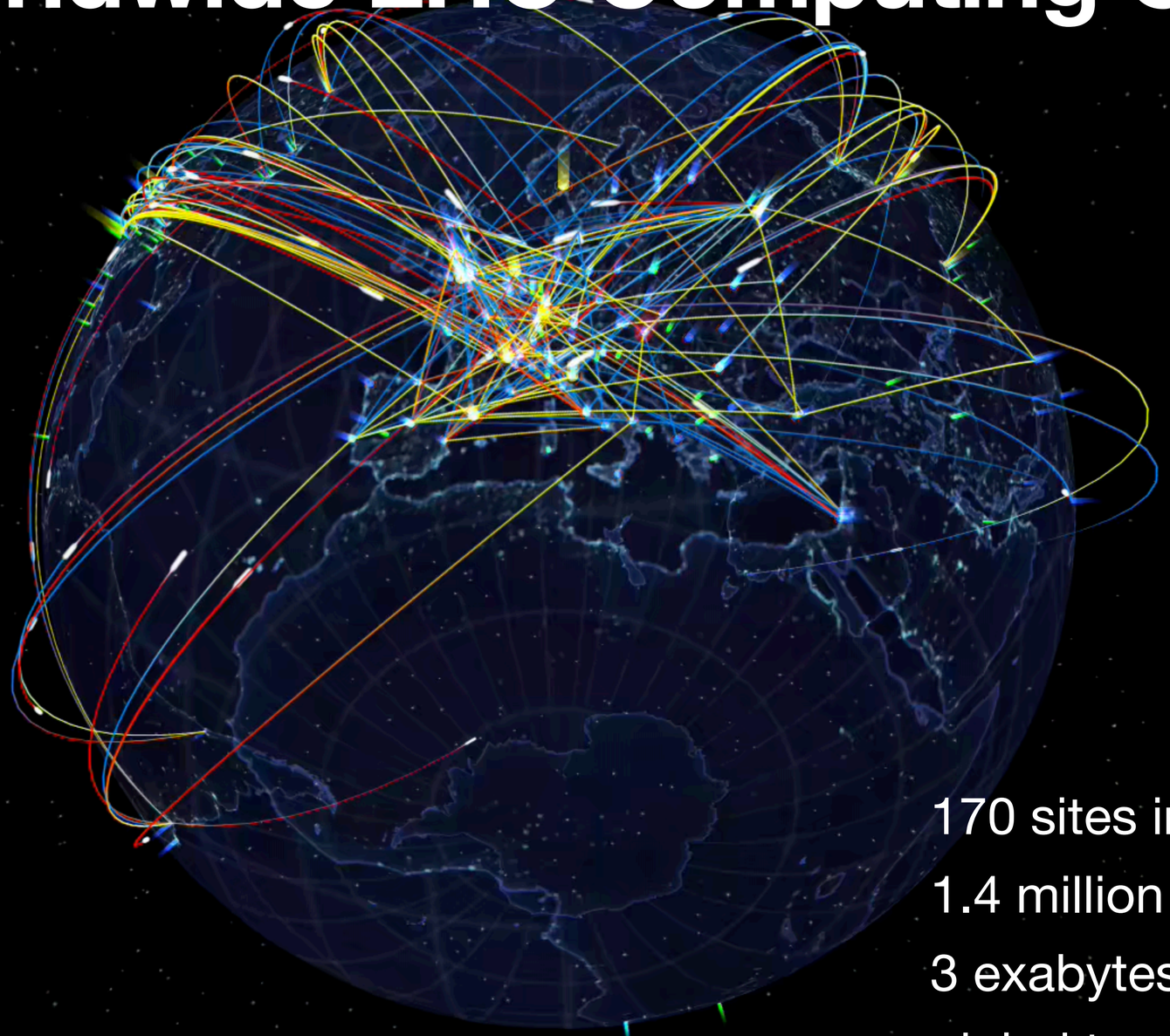
170 sites in 42 countries

1.4 million computer cores

3 exabytes of storage

global transfer rates > 260 GB/s

The Worldwide LHC Computing Grid



Running jobs: 365644
Active CPU cores: 807139
Transfer rate: 21.54 GiB/sec

170 sites in 42 countries

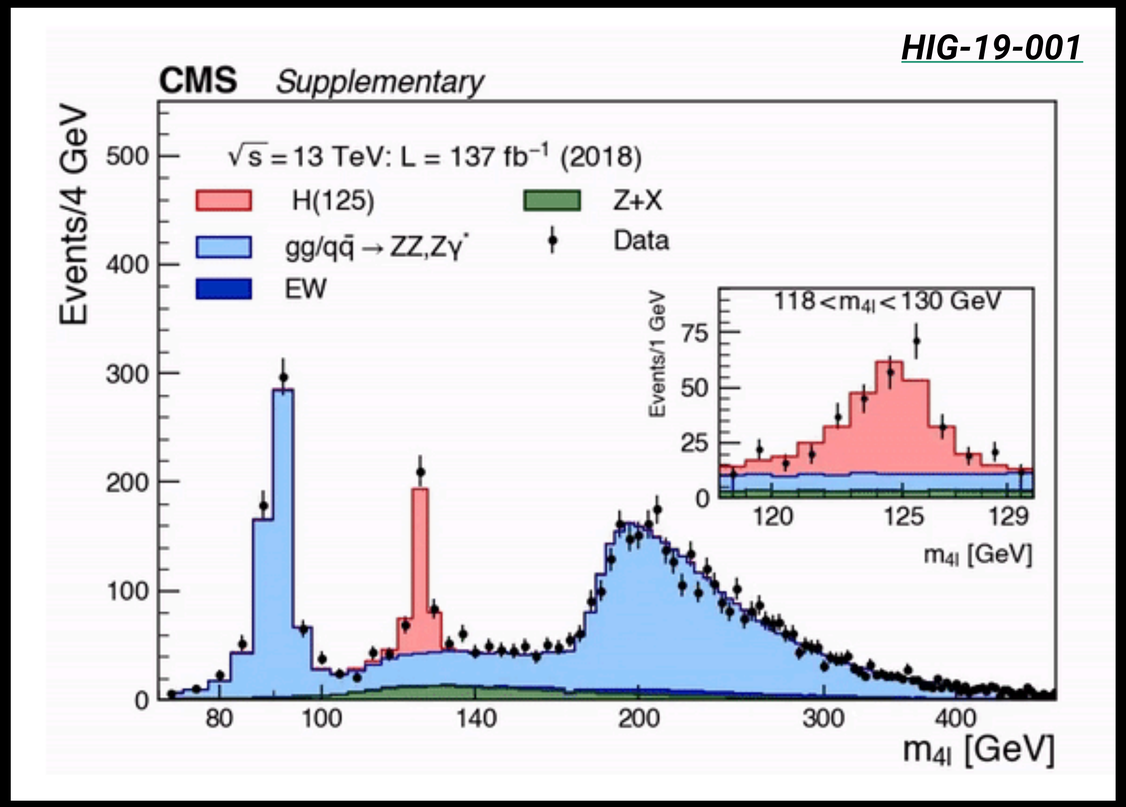
1.4 million computer cores

3 exabytes of storage

global transfer rates > 260 GB/s

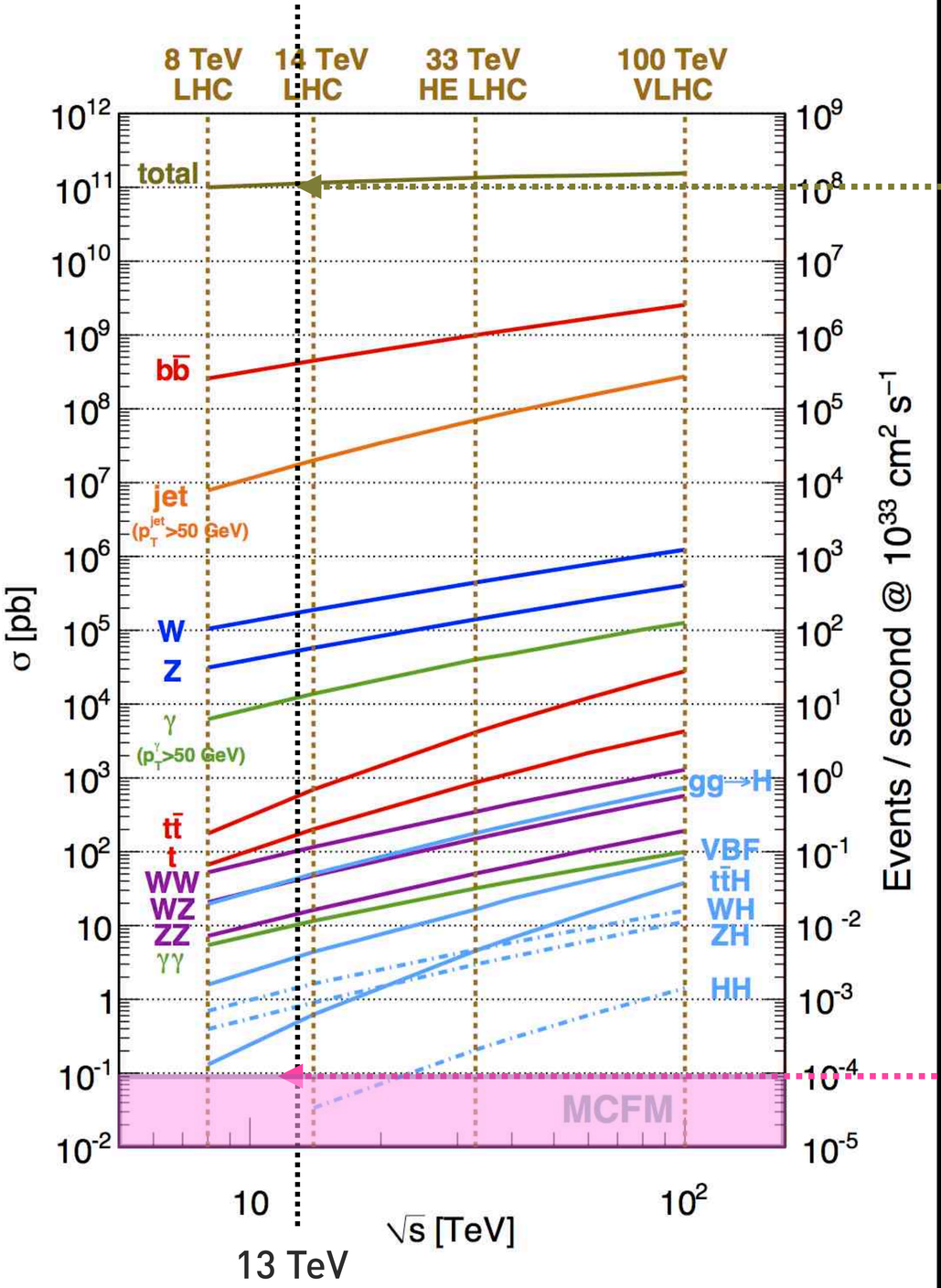
The Standard Model

$$-\tfrac{1}{2}\partial_\nu g^a_\mu \partial_\nu g^a_\mu - g_s f^{abc}\partial_\mu g^a_\nu g^b_\mu g^c_\nu - \tfrac{1}{4}g_s^2 f^{abc}f^{ade}g^b_\mu g^c_\nu g^d_\mu g^e_\nu + \tfrac{1}{2}ig_s^2(\bar{q}^\sigma_i\gamma^\mu q^\sigma_j)g^a_\mu + \bar{G}^a\partial^2 G^a + g_s f^{abc}\partial_\mu \bar{G}^a G^b g^c_\mu - \partial_\nu W^+_\mu \partial_\nu W^-_\mu - M^2 W^+_\mu W^-_\mu - \tfrac{1}{2}\partial_\nu Z^0_\mu \partial_\nu Z^0_\mu - \tfrac{1}{2c_w^2}M^2 Z^0_\mu Z^0_\mu - \tfrac{1}{2}\partial_\mu A_\nu \partial_\mu A_\nu - \tfrac{1}{2}\partial_\mu H \partial_\mu H - \tfrac{1}{2}m_h^2 H^2 - \partial_\mu \phi^+ \partial_\mu \phi^- - M^2\phi^+\phi^- - \tfrac{1}{2}\partial_\mu \phi^0 \partial_\mu \phi^0 - \tfrac{1}{2c_w^2}M\phi^0\phi^0 - \beta_h[\tfrac{2M^2}{g^2} + \tfrac{2M}{g}H + \tfrac{1}{2}(H^2+\phi^0\phi^0+2\phi^+\phi^-)] + \tfrac{2M^4}{g^2}\alpha_h - igc_w[\partial_\nu Z^0_\mu(W^+_\mu W^-_\nu - W^+_\nu W^-_\mu) - Z^0_\nu(W^+_\mu \partial_\nu W^-_\mu - W^-_\mu \partial_\nu W^+_\mu) + Z^0_\mu(W^+_\nu \partial_\nu W^-_\mu - W^-_\nu \partial_\nu W^+_\mu)] - igs_w[\partial_\nu A_\mu(W^+_\mu W^-_\nu - W^+_\nu W^-_\mu) - A_\nu(W^+_\mu \partial_\nu W^-_\mu - W^-_\mu \partial_\nu W^+_\mu) + A_\mu(W^+_\nu \partial_\nu W^-_\mu - W^-_\nu \partial_\nu W^+_\mu)] - \tfrac{1}{2}g^2 W^+_\mu W^-_\mu W^+_\nu W^-_\nu + \tfrac{1}{2}g^2 W^+_\mu W^-_\nu W^+_\mu W^-_\nu + g^2 c_w^2(Z^0_\mu W^+_\mu Z^0_\nu W^-_\nu - Z^0_\mu Z^0_\mu W^+_\nu W^-_\nu) + g^2 s_w^2(A_\mu W^+_\mu A_\nu W^-_\nu - A_\mu A_\mu W^+_\nu W^-_\nu) + g^2 s_w c_w[A_\mu Z^0_\nu(W^+_\mu W^-_\nu - W^+_\nu W^-_\mu) - 2A_\mu Z^0_\mu W^+_\nu W^-_\nu] - g\alpha[H^3 + H\phi^0\phi^0 + 2H\phi^+\phi^-] - \tfrac{1}{8}g^2\alpha_h[H^4 + (\phi^0)^4 + 4(\phi^+\phi^-)^2 + 4(\phi^0)^2\phi^+\phi^- + 4H^2\phi^+\phi^- + 2(\phi^0)^2H^2] - gMW^+_\mu W^-_\mu H - \tfrac{1}{2}g\tfrac{M}{c_w^2}Z^0_\mu Z^0_\mu H - \tfrac{1}{2}ig[W^+_\mu(\phi^0\partial_\mu\phi^- - \phi^-\partial_\mu\phi^0) - W^-_\mu(\phi^0\partial_\mu\phi^+ - \phi^+\partial_\mu\phi^0)] + \tfrac{1}{2}g[W^+_\mu(H\partial_\mu\phi^- - \phi^-\partial_\mu H) - W^-_\mu(H\partial_\mu\phi^+ - \phi^+\partial_\mu H)] + \tfrac{1}{2}g\tfrac{1}{c_w}(Z^0_\mu(H\partial_\mu\phi^0 - \phi^0\partial_\mu H) - ig\tfrac{s_w^2}{c_w}MZ^0_\mu(W^+_\mu\phi^- - W^-_\mu\phi^+) + igs_wMA_\mu(W^+_\mu\phi^- - W^-_\mu\phi^+) - ig\tfrac{1-2c_w^2}{2c_w}Z^0_\mu(\phi^+\partial_\mu\phi^- - \phi^-\partial_\mu\phi^+) + igs_wA_\mu(\phi^+\partial_\mu\phi^- - \phi^-\partial_\mu\phi^+) - \tfrac{1}{4}g^2W^+_\mu W^-_\mu[H^2 + (\phi^0)^2 + 2\phi^+\phi^-] - \tfrac{1}{4}g^2\tfrac{1}{c_w^2}Z^0_\mu Z^0_\mu[H^2 + (\phi^0)^2 + 2(2s_w^2-1)^2\phi^+\phi^-] - \tfrac{1}{2}g^2\tfrac{s_w^2}{c_w}Z^0_\mu\phi^0(W^+_\mu\phi^- + W^-_\mu\phi^+) - \tfrac{1}{2}ig^2\tfrac{s_w^2}{c_w}Z^0_\mu H(W^+_\mu\phi^- - W^-_\mu\phi^+) + \tfrac{1}{2}g^2s_wA_\mu\phi^0(W^+_\mu\phi^- + W^-_\mu\phi^+) + \tfrac{1}{2}ig^2s_wA_\mu H(W^+_\mu\phi^- - W^-_\mu\phi^+) - g^2\tfrac{s_w}{c_w}(2c_w^2-1)Z^0_\mu A_\mu\phi^+\phi^- - g^1s_w^2A_\mu A_\mu\phi^+\phi^- - \bar{e}^\lambda(\gamma\partial + m^\lambda_e)e^\lambda - \bar{\nu}^\lambda\gamma\partial\nu^\lambda - \bar{u}^\lambda_j(\gamma\partial + m^\lambda_u)u^\lambda_j - \bar{d}^\lambda_j(\gamma\partial + m^\lambda_d)d^\lambda_j + igs_wA_\mu[-(\bar{e}^\lambda\gamma^\mu e^\lambda) + \tfrac{2}{3}(\bar{u}^\lambda_j\gamma^\mu u^\lambda_j) - \tfrac{1}{3}(\bar{d}^\lambda_j\gamma^\mu d^\lambda_j)] + \tfrac{ig}{4c_w}Z^0_\mu[(\bar{\nu}^\lambda\gamma^\mu(1+\gamma^5)\nu^\lambda) + (\bar{e}^\lambda\gamma^\mu(4s_w^2 - 1 - \gamma^5)e^\lambda) + (\bar{u}^\lambda_j\gamma^\mu(\tfrac{4}{3}s_w^2 - 1 - \gamma^5)u^\lambda_j) + (\bar{d}^\lambda_j\gamma^\mu(1 - \tfrac{8}{3}s_w^2 - \gamma^5)d^\lambda_j)] + \tfrac{ig}{2\sqrt{2}}W^+_\mu[(\bar{\nu}^\lambda\gamma^\mu(1+\gamma^5)e^\lambda) + (\bar{u}^\lambda_j\gamma^\mu(1+\gamma^5)C_{\lambda\kappa}d^\kappa_j)] + \tfrac{ig}{2\sqrt{2}}W^-_\mu[(\bar{e}^\lambda\gamma^\mu(1+\gamma^5)\nu^\lambda) + (\bar{d}^\kappa_jC^\dagger_{\lambda\kappa}\gamma^\mu(1+\gamma^5)u^\lambda_j)] + \tfrac{ig}{2\sqrt{2}}\tfrac{m^\lambda_e}{M}[-\phi^+(\bar{\nu}^\lambda(1-\gamma^5)e^\lambda) + \phi^-(\bar{e}^\lambda(1+\gamma^5)\nu^\lambda)] - \tfrac{g}{2}\tfrac{m^\lambda_e}{M}[H(\bar{e}^\lambda e^\lambda) + i\phi^0(\bar{e}^\lambda\gamma^5 e^\lambda)] + \tfrac{ig}{2M\sqrt{2}}\phi^+[-m^\kappa_d(\bar{u}^\lambda_jC_{\lambda\kappa}(1-\gamma^5)d^\kappa_j) + m^\lambda_u(\bar{u}^\lambda_jC_{\lambda\kappa}(1+\gamma^5)d^\kappa_j] + \tfrac{ig}{2M\sqrt{2}}\phi^-[m^\lambda_d(\bar{d}^\lambda_jC^\dagger_{\lambda\kappa}(1+\gamma^5)u^\kappa_j) - m^\kappa_u(\bar{d}^\lambda_jC^\dagger_{\lambda\kappa}(1-\gamma^5)u^\kappa_j] - \tfrac{g}{2}\tfrac{m^\lambda_u}{M}H(\bar{u}^\lambda_ju^\lambda_j) - \tfrac{g}{2}\tfrac{m^\lambda_d}{M}H(\bar{d}^\lambda_jd^\lambda_j) + \tfrac{ig}{2}\tfrac{m^\lambda_u}{M}\phi^0(\bar{u}^\lambda_j\gamma^5u^\lambda_j) - \tfrac{ig}{2}\tfrac{m^\lambda_d}{M}\phi^0(\bar{d}^\lambda_j\gamma^5d^\lambda_j) + \bar{X}^+(\partial^2 - M^2)X^+ + \bar{X}^-(\partial^2 - M^2)X^- + \bar{X}^0(\partial^2 - \tfrac{M^2}{c_w^2})X^0 + \bar{Y}\partial^2 Y + igc_wW^+_\mu(\partial_\mu\bar{X}^0X^- - \partial_\mu\bar{X}^+X^0) + igs_wW^+_\mu(\partial_\mu\bar{Y}X^- - \partial_\mu\bar{X}^+Y) + igc_wW^-_\mu(\partial_\mu\bar{X}^-X^0 - \partial_\mu\bar{X}^0X^+) + igs_wW^-_\mu(\partial_\mu\bar{X}^-Y - \partial_\mu\bar{Y}X^+) + igc_wZ^0_\mu(\partial_\mu\bar{X}^+X^+ - \partial_\mu\bar{X}^-X^-) + igs_wA_\mu(\partial_\mu\bar{X}^+X^+ - \partial_\mu\bar{X}^-X^-) - \tfrac{1}{2}gM[\bar{X}^+X^+H + \bar{X}^-X^-H + \tfrac{1}{c_w^2}\bar{X}^0X^0H] + \tfrac{1-2c_w^2}{2c_w}igM[\bar{X}^+X^0\phi^+ - \bar{X}^-X^0\phi^-] + \tfrac{1}{2c_w}igM[\bar{X}^0X^-\phi^+ - \bar{X}^0X^+\phi^-] + igMs_w[\bar{X}^0X^-\phi^+ - \bar{X}^0X^+\phi^-] + \tfrac{1}{2}igM[\bar{X}^+X^+\phi^0 - \bar{X}^-X^-\phi^0]$$



To make sure we select "the right" 0.02%, algorithms must be

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



"Probability" of producing "anything"

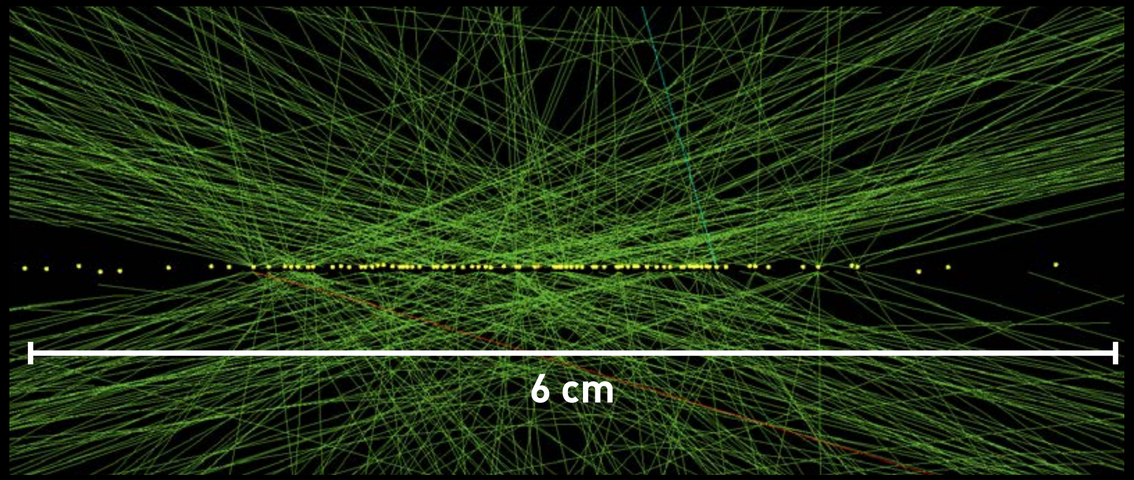
New Physics?

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

Need more data!

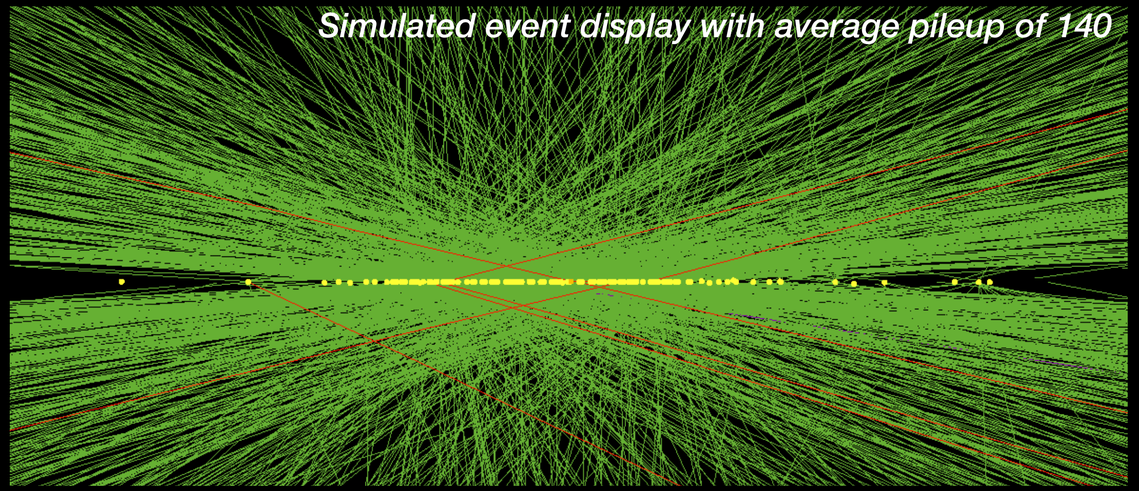
LHC (currently)

78 vertices
(average 60)



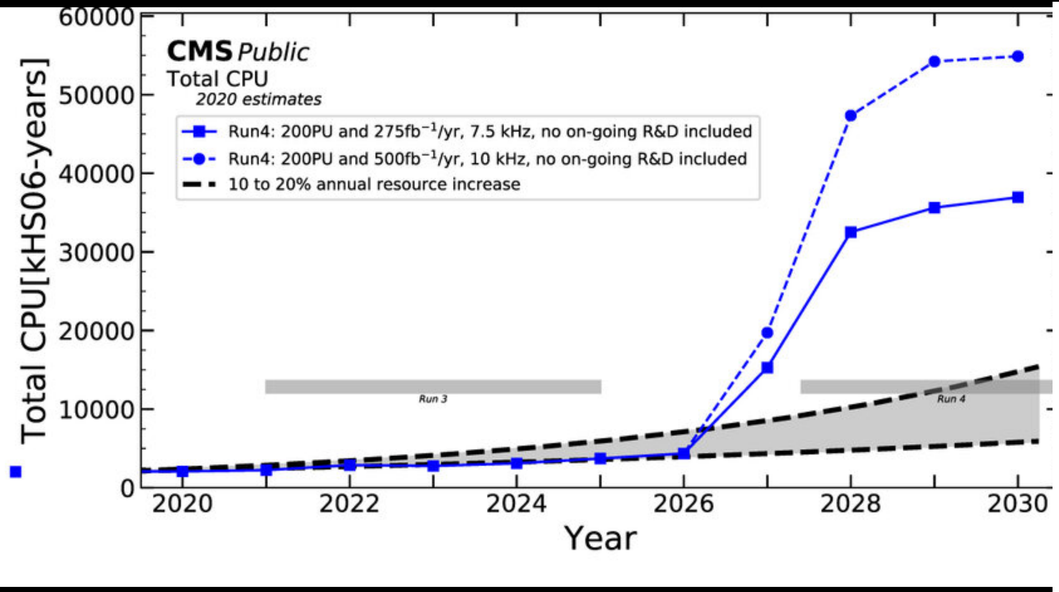
High Luminosity LHC (2030-2041)

200 vertices
(average 140)

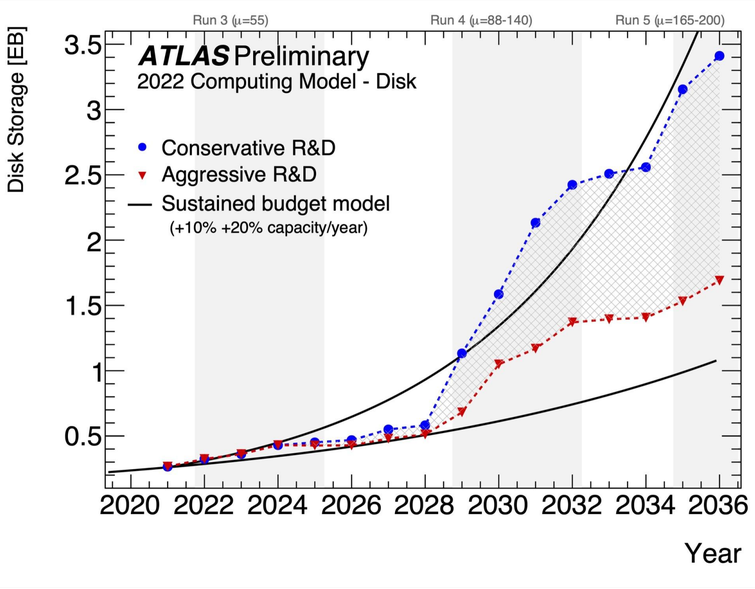


Future CPU and storage: needs vs budget

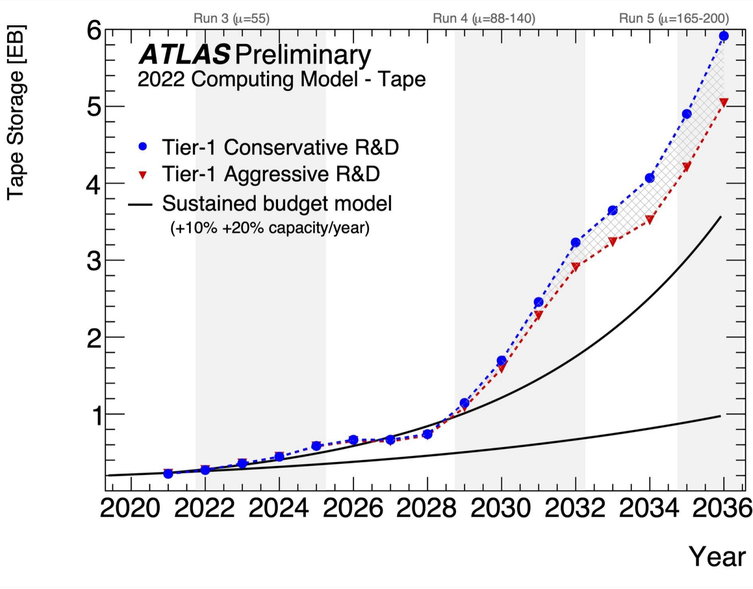
CPU



DISK

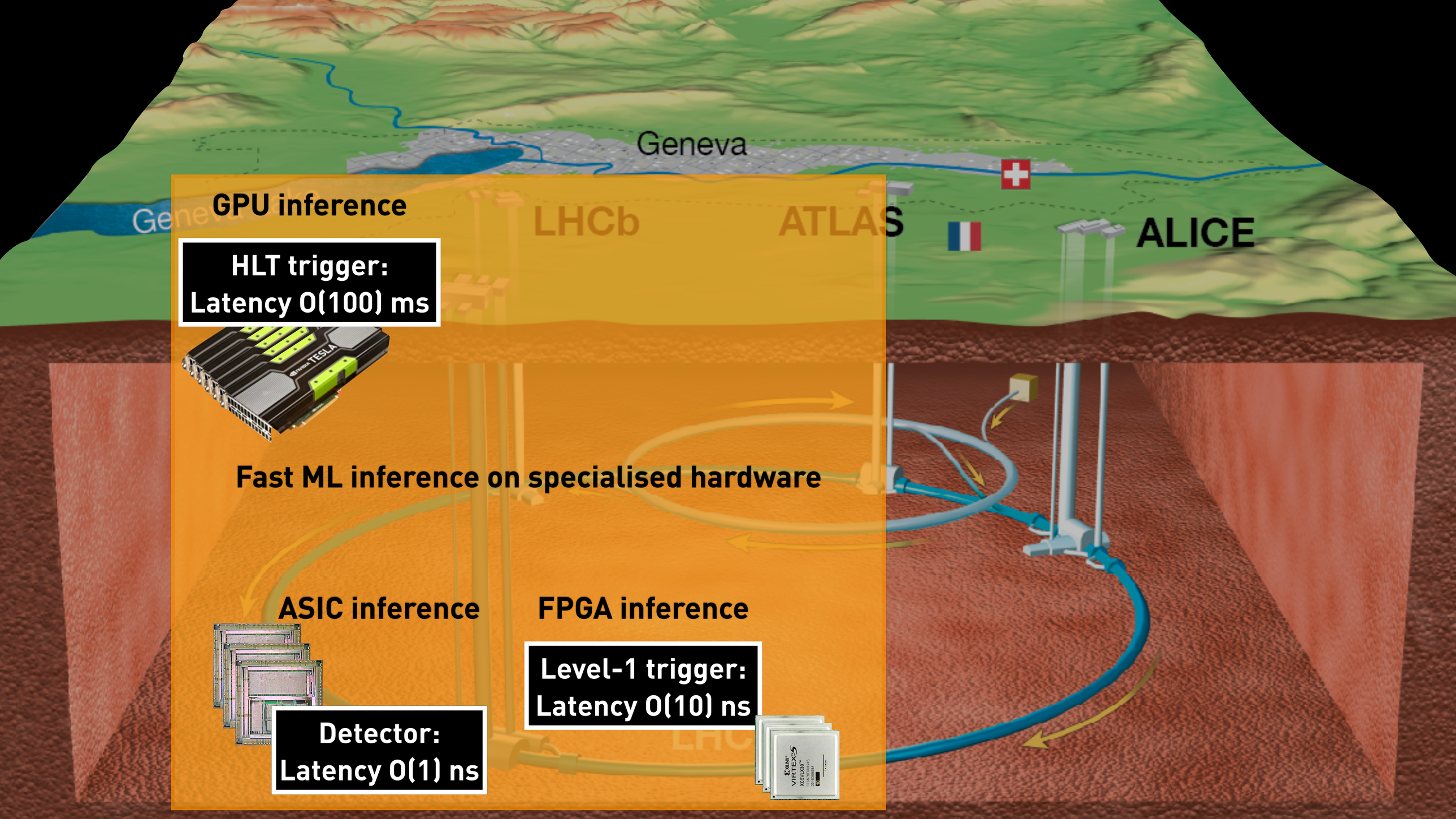


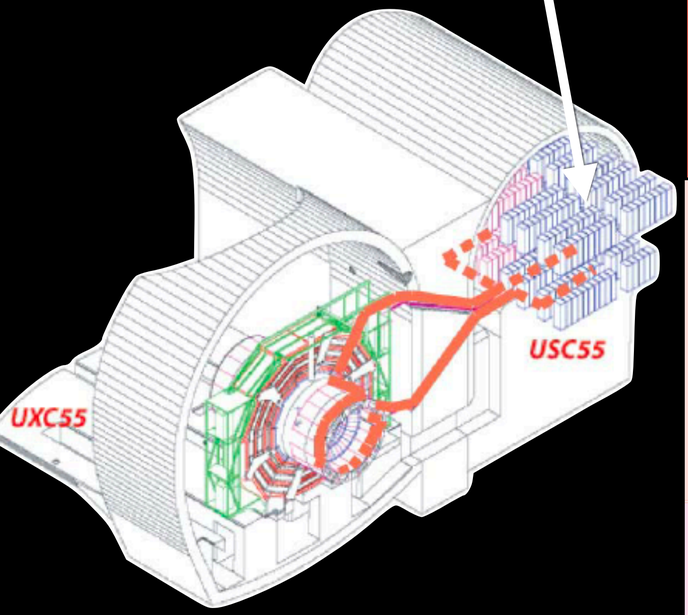
TAPE

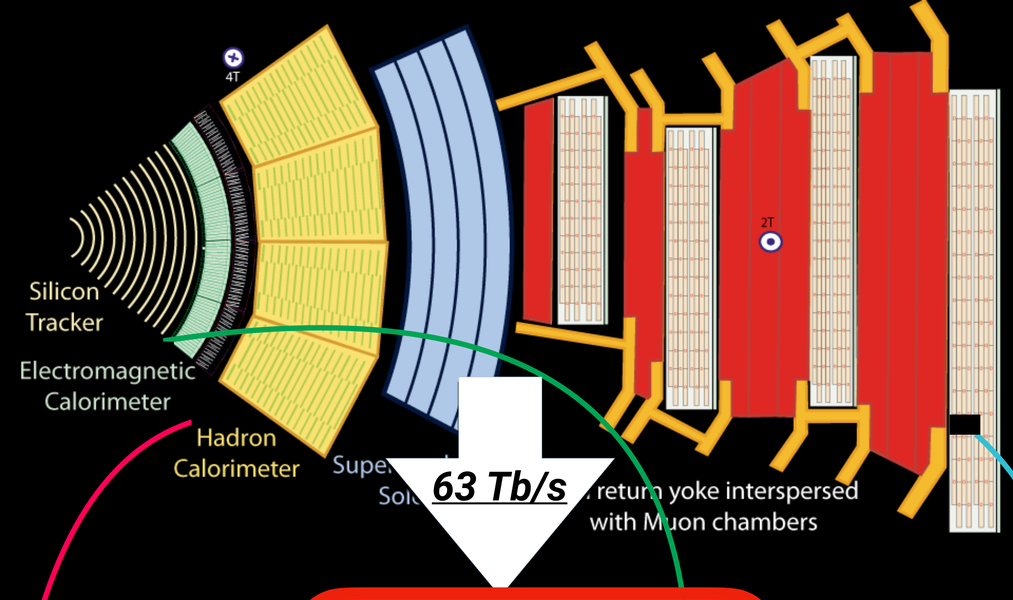


- Must get data out of detector while minimising “dead material”
- Must process all that data (20 s/event, 1B events/s)
- Must store all that data (~6-8 MB/event)

- **DATA IS OUR MAIN PRECISION BOTTLENECK**


Geneva
LHCb
ATLAS
ALICE
GPU inference
HLT trigger:
Latency O(100) ms
Fast ML inference on specialised hardware
ASIC inference
FPGA inference
Level-1 trigger:
Latency O(10) ns
Detector:
Latency O(1) ns





CALORIMETRY:
370 FPGAs

*54 for HGCAL only!

Xilinx Ultrascale+ FPGAs

TRACKING
174 FPGAs

MUONS:
96 FPGAs

5 µs

PARTICLE FLOW:
66 FPGAs

CALORIMETRY

PARTICLE FLOW

MUONS

EXTERNAL TRIGGERS

GLOBAL TRIGGER:
12 FPGAs

Trigger accept/reject

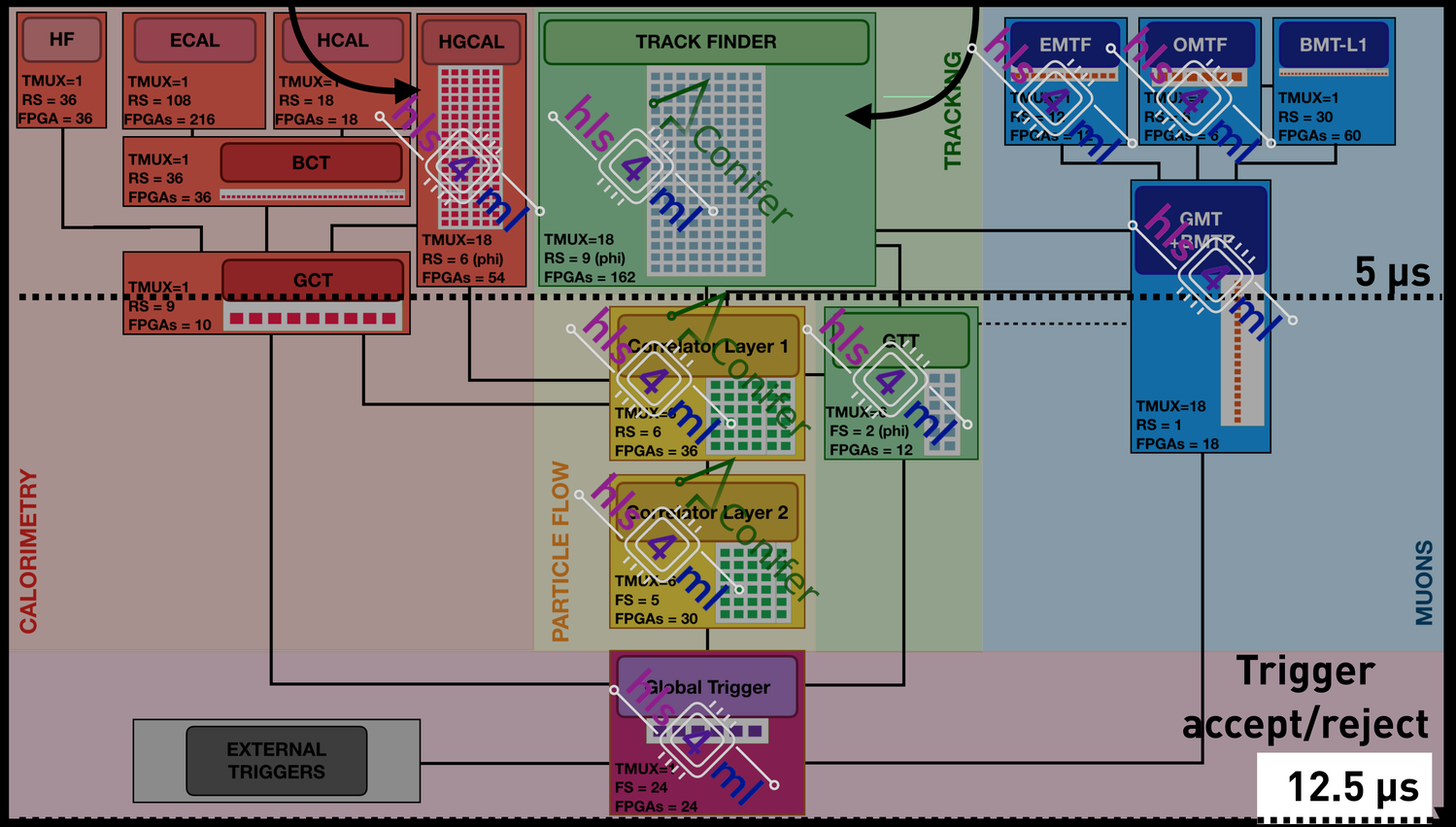
12.5 µs

Nanosecond ML inference on FPGAs!

~40 billion inferences/s during HL-LHC

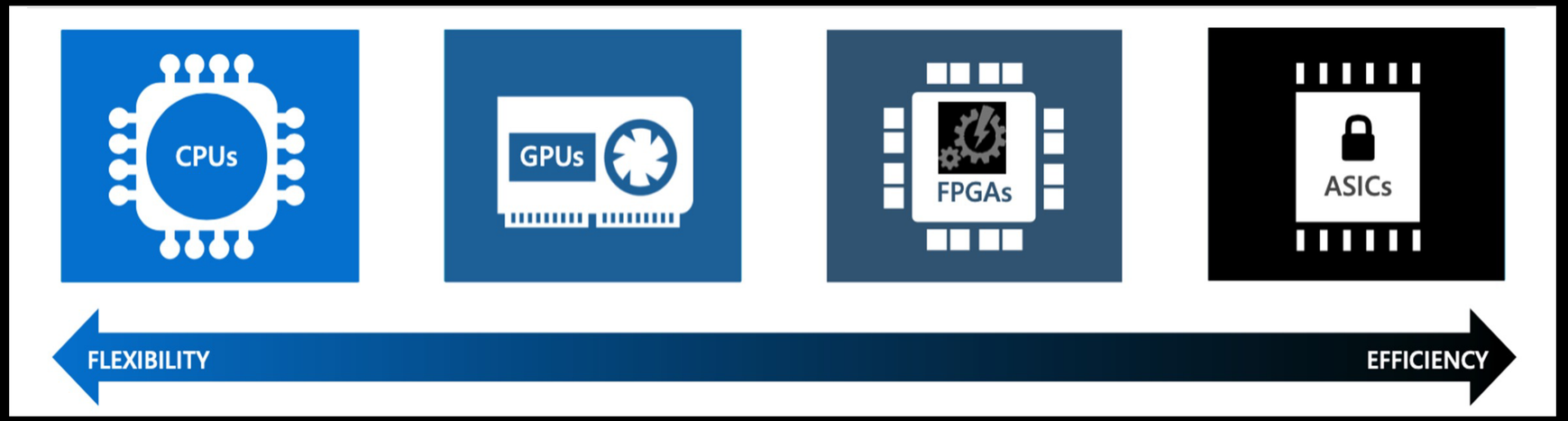
Conifer hls4ml

HEP developed libraries for fast ML on FPGAs



Why FPGAs?

Why FPGAs?



- Custom hardware acceleration, precisions and memory management
- Data-flow architecture with no scheduling or control overheads

Why FPGAs?

- Latency (resource parallelism)

Why FPGAs?

- Throughput (pipeline parallelism)

Latency (resource parallelism)

Can work on different parts of problem/different data simultaneously

High bandwidth (pipeline parallelism)

Latency deterministic

CPU/GPU processing randomness,

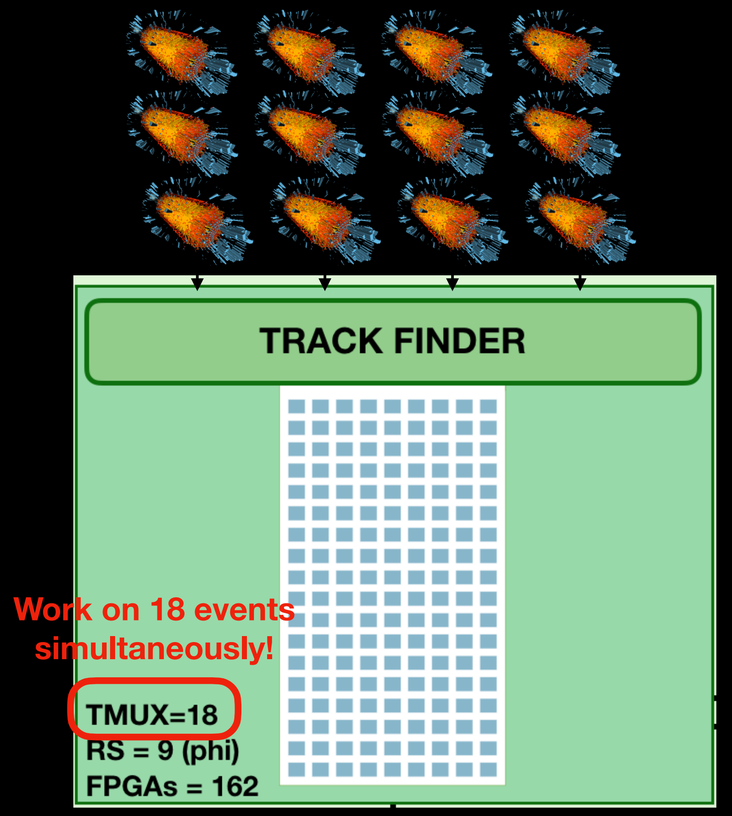
FPGAs repeatable and predictable latency

Have specialised blocks for I/O

Popular for embedded systems and HEP triggers

High speed transceivers with Tb/s total bandwidth (PCIe, 100G Ethernet, InfiniBand)

Low power per Op (relative to CPU/GPU)



Contain many different building blocks (‘resources’) which are connected together as you desire!

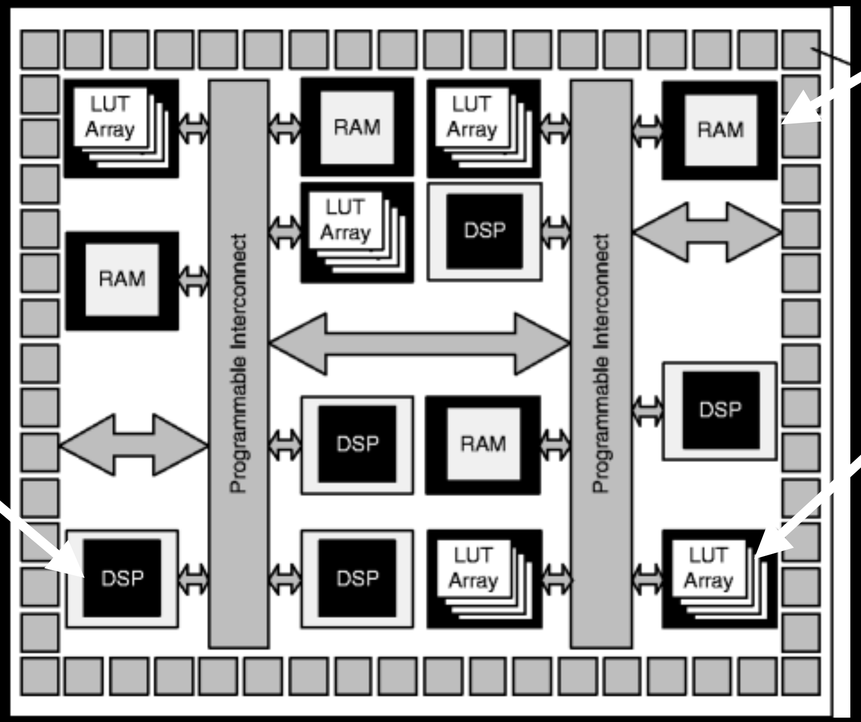
$$\mathbf{x}_n = g_n(\mathbf{W}_{n,n-1}\mathbf{x}_{n-1} + \mathbf{b}_n)$$

activation function — precomputed and stored in BRAMs

multiplication — DSPs

addition — logic cells

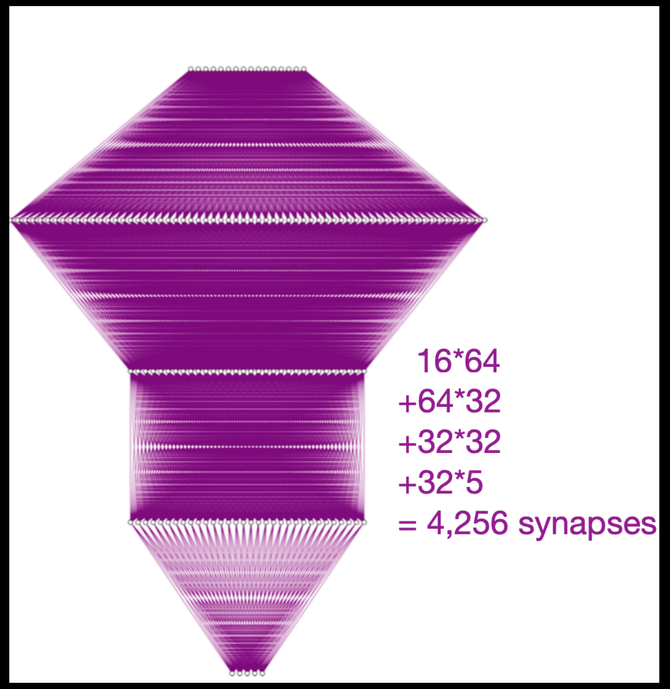
Digital signal processors (DSPs) **O(5,000) units:** specialized units for multiplication and arithmetic. Faster & more efficient than LUTs. For neural networks, DSPs are the most scarce



Memory (BRAM) **O(2000) units:** small, fast memories. Access data in one clock cycle. Big FPGA ~ 40MB BRAM chained together as needed (bandwidth)

Logic cells/lookup tables (LUTs) **O(1) million units**: perform arbitrary functions on small bit width inputs. Used for Boolean operations, arithmetic, small memories

flip-flops (FF): registers data in time with clock



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

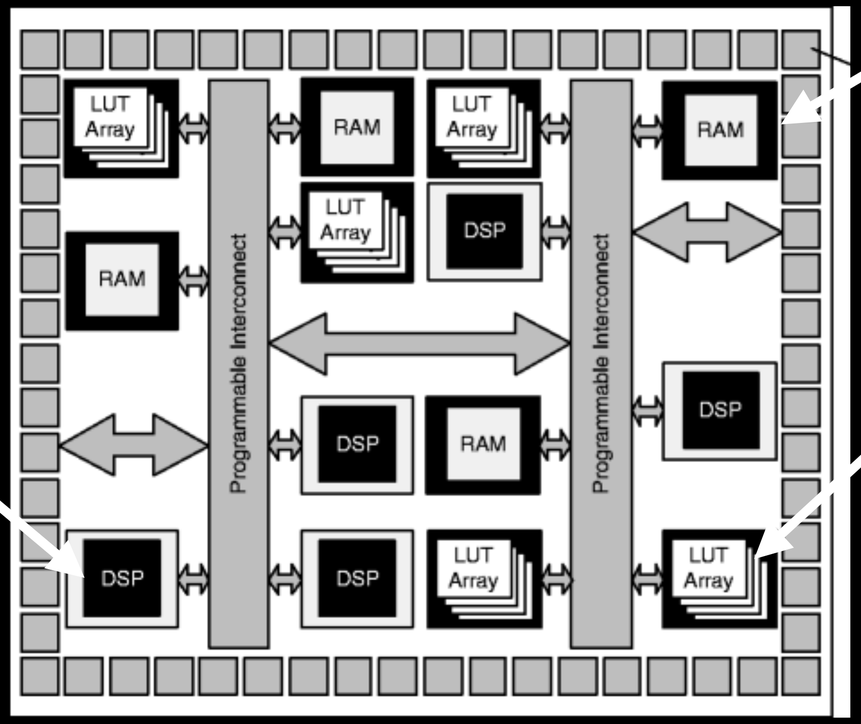
$$\mathbf{x}_n = g_n(\mathbf{W}_{n,n-1}\mathbf{x}_{n-1} + \mathbf{b}_n)$$

activation function
precomputed and stored in BRAMs

multiplication
DSPs

addition
logic cells

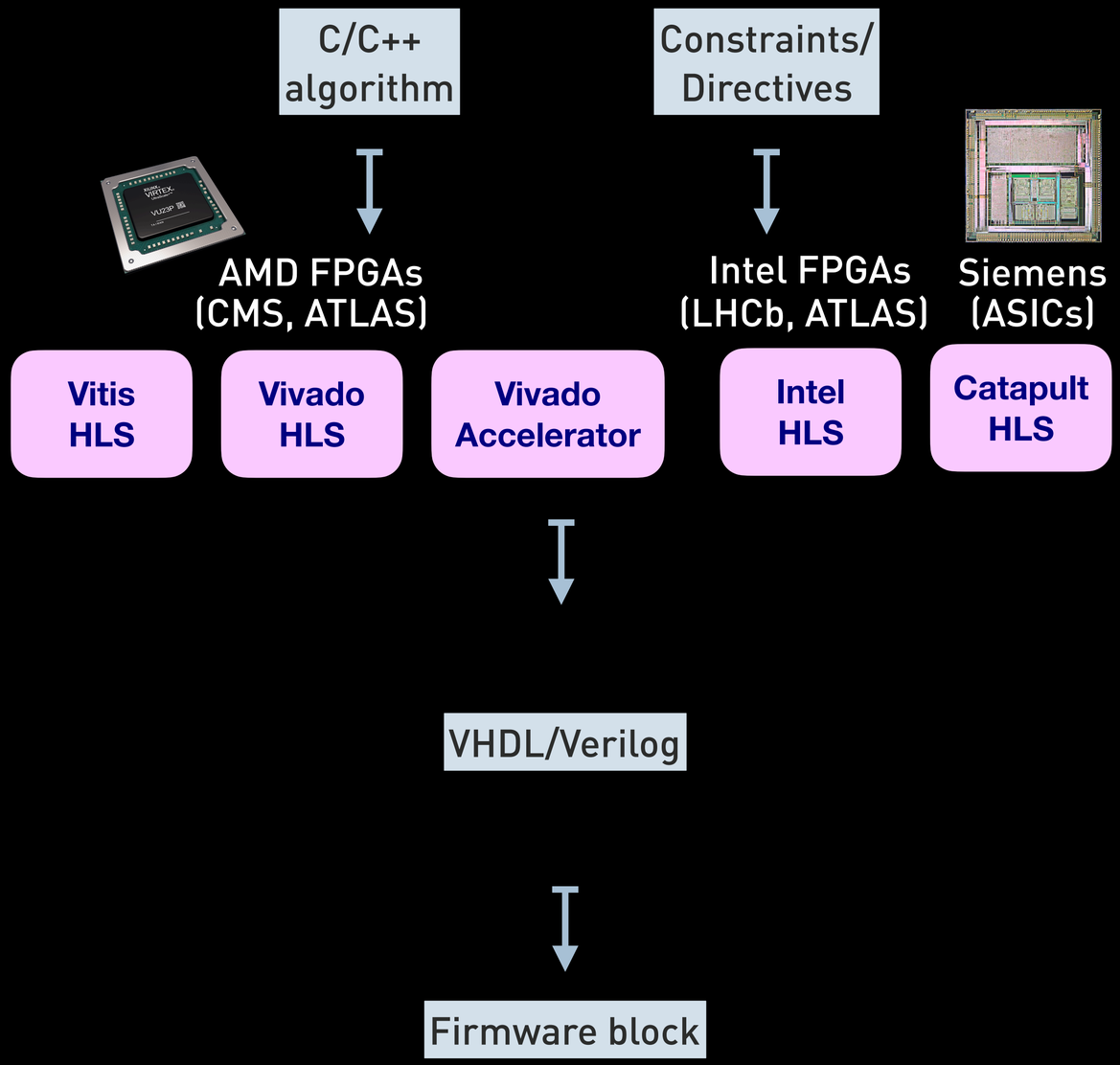
Digital signal processors (DSPs)
O(5,000) units



Memory (BRAM)
O(2000) units

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

flip-flops (FF):
registers data in time with clock

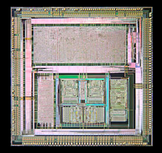

C/C++
algorithm
Constraints/
Directives
AMD FPGAs
(CMS, ATLAS)
Intel FPGAs
(LHCb, ATLAS)
Siemens
(ASICs)
Vitis
HLS
Vivado
HLS
Vivado
Accelerator
Intel
HLS
Catapult
HLS
VHDL/Verilog
Firmware block

C/C++ algorithm

Constraints/ Directives

AMD FPGAs (CMS, ATLAS)

Intel FPGAs (LHCb, ATLAS)



Siemens (ASICs)

Vitis HLS | **Vivado HLS** | **Vivado Accelerator** | **Intel HLS** | **Catapult HLS**

Hardware Description Languages:
programming languages which describe electronic circuits

VHDL/Verilog

Firmware block

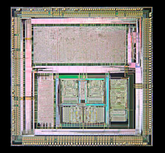
High Level Synthesis:
Compile from C/C++ to VHDL

Pre-processor directives and constraints used to optimize the design

Drastic decrease in firmware development time!

C/C++ algorithm

Constraints/ Directives



AMD FPGAs (CMS, ATLAS)

Intel FPGAs (LHCb, ATLAS)

Siemens (ASICs)

Vitis HLS

Vivado HLS

Vivado Accelerator

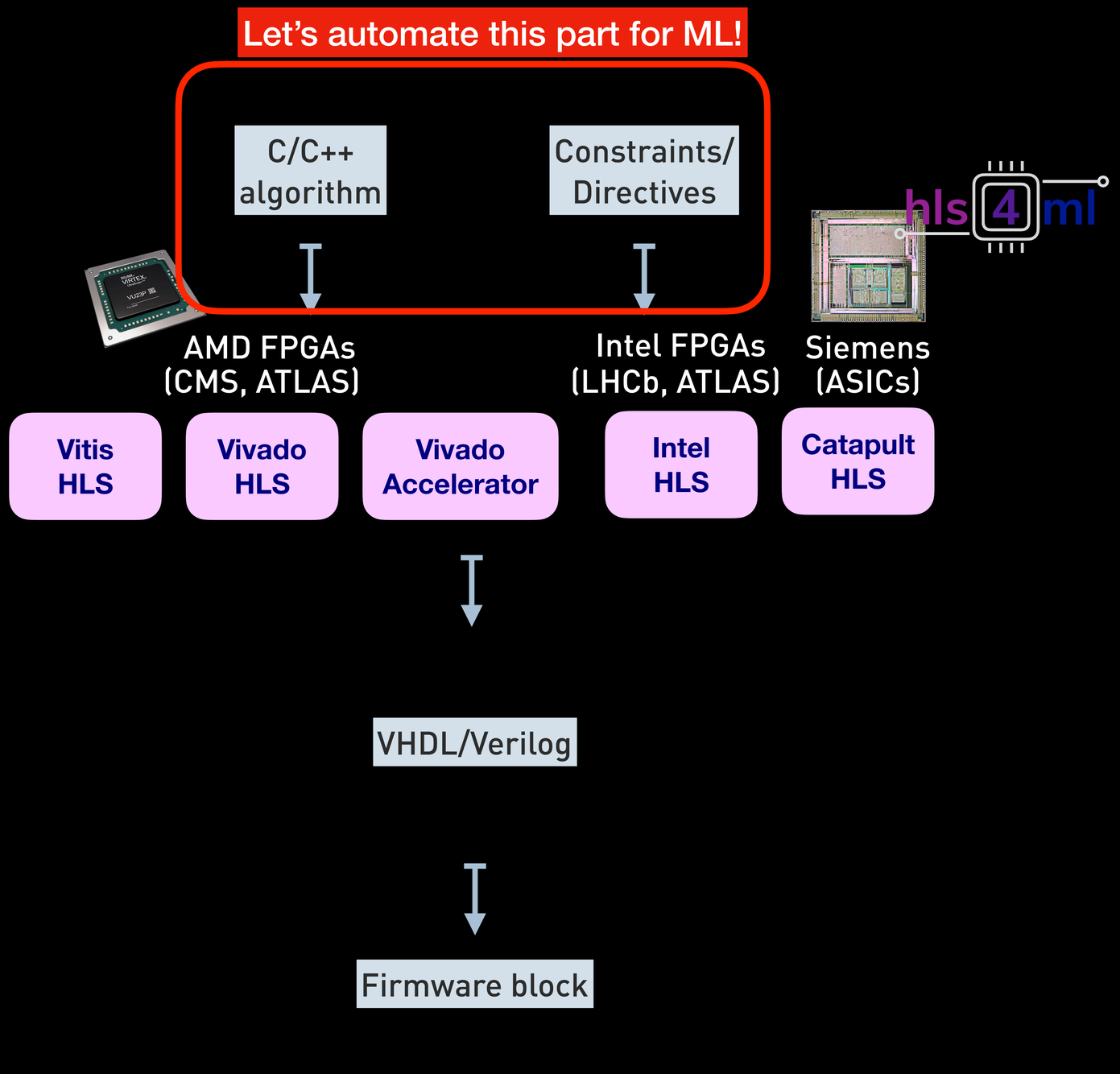
Intel HLS

Catapult HLS

Hardware Description Languages:
programming languages which describe electronic circuits

VHDL/Verilog

Firmware block


Let’s automate this part for ML!
C/C++ algorithm
Constraints/ Directives
hls 4 ml
AMD FPGAs (CMS, ATLAS)
Intel FPGAs (LHCb, ATLAS)
Siemens (ASICs)
Vitis HLS
Vivado HLS
Vivado Accelerator
Intel HLS
Catapult HLS
VHDL/Verilog
Firmware block

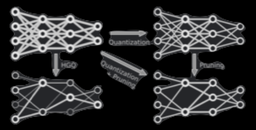
Floating point:

Model

Quantized:

Brevitas

QONNX



HQG

Floating point:

Keras + TensorFlow

ONNX

PYTORCH

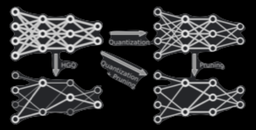
Model

Quantized:

QKeras

Brevitas

QONNX



HQG

hls4ml

HLS Model

Write HLS project
Emulate
Run synthesis

Catapult HLS SIEMENS

Vivado™ HLS

HLS COMPILER

XILINX VITIS™

Floating point:

Keras + TensorFlow

ONNX

PYTORCH

Model

Quantized:

QKeras

Brevitas

QONNX

HQG

hls4ml

HLS Model

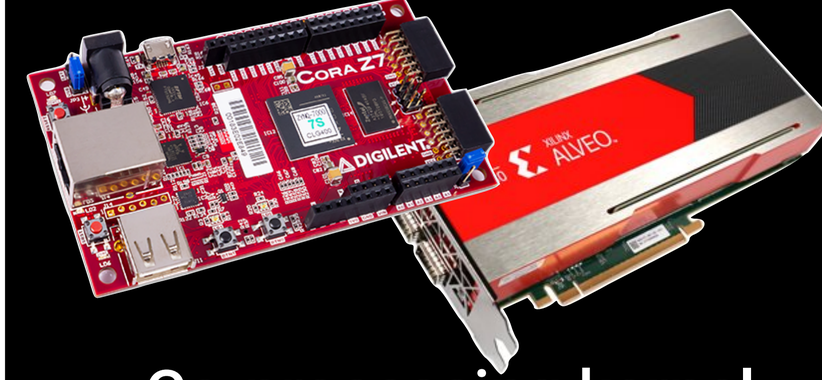
Write HLS project
Emulate
Run synthesis

Catapult HLS SIEMENS

Vivado™ HLS

HLS COMPILER

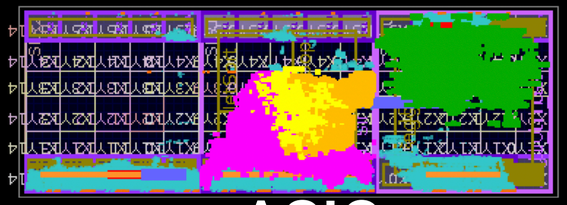
XILINX VITIS™



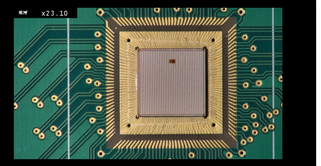
Co-processing kernel
(Accelerators/SoCs)

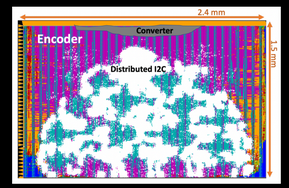
Hardware Model

FPGA custom designs

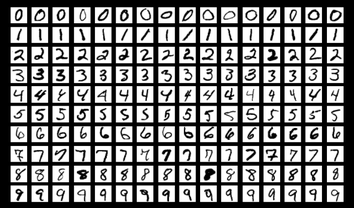


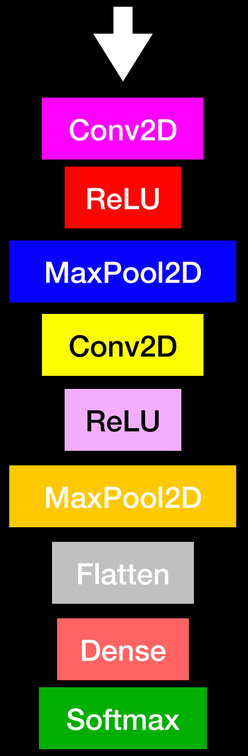
ASICs



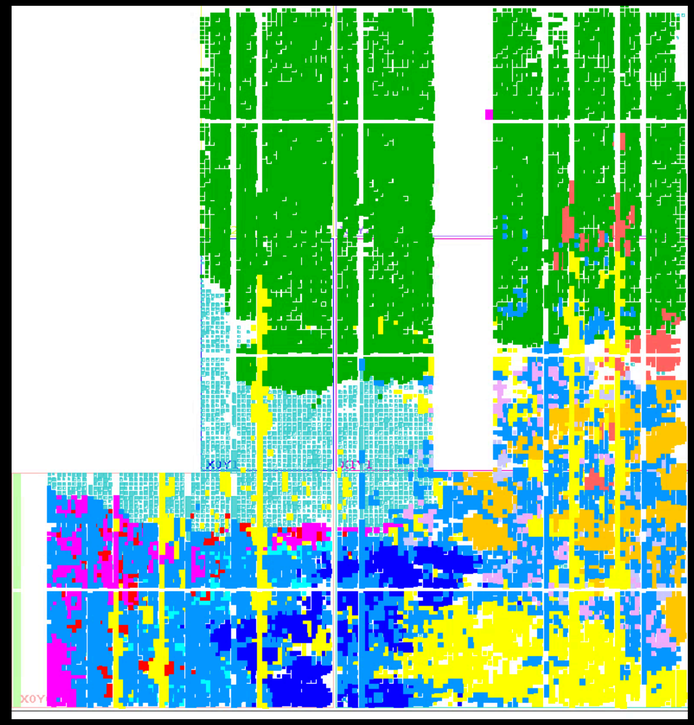


Exercise 1+2



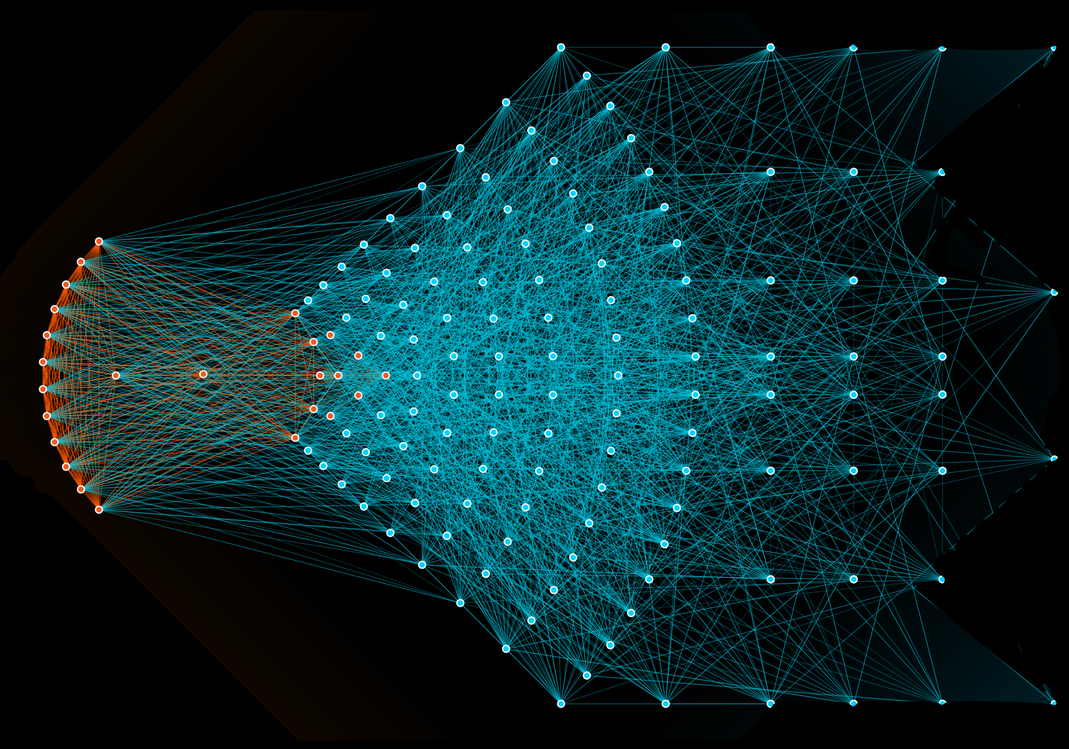


hls4ml

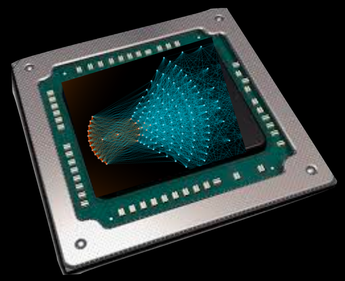


“Data flow architecture”

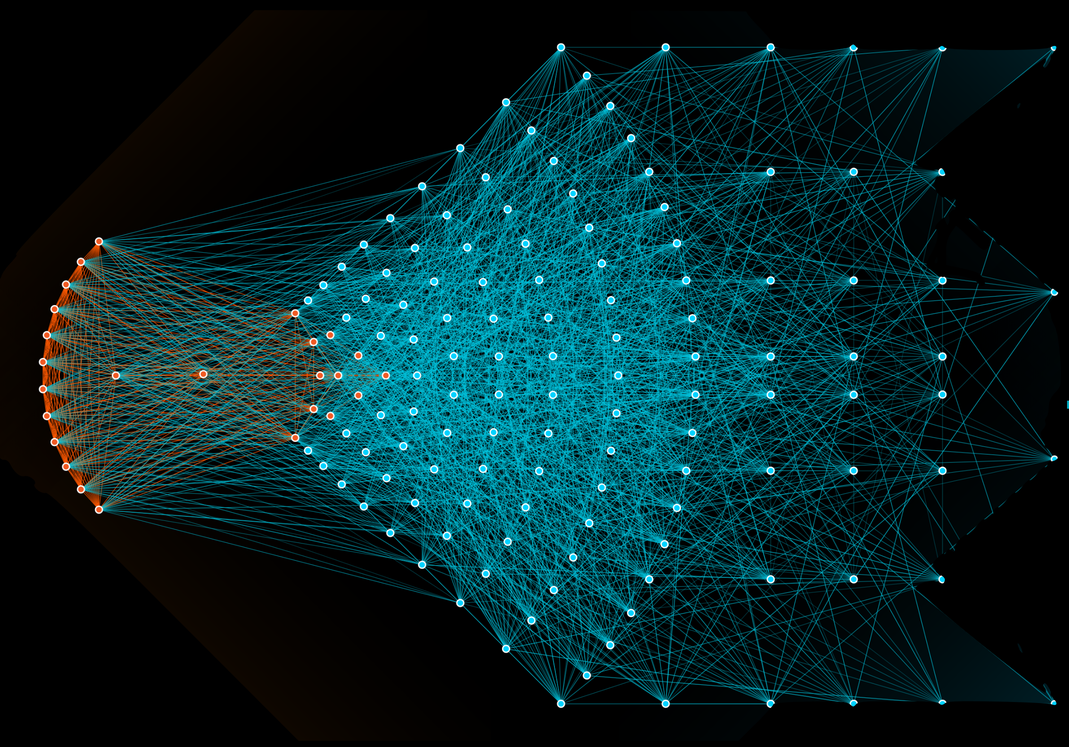
- Tailored hardware for a model
- Each layer is separate compute unit
- Stay on-chip
- “Decisions are design time”



Ideally

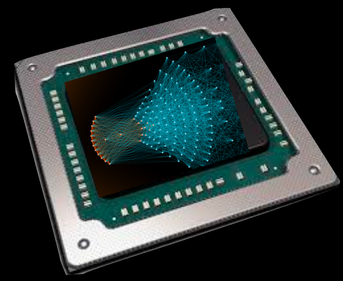


Reality



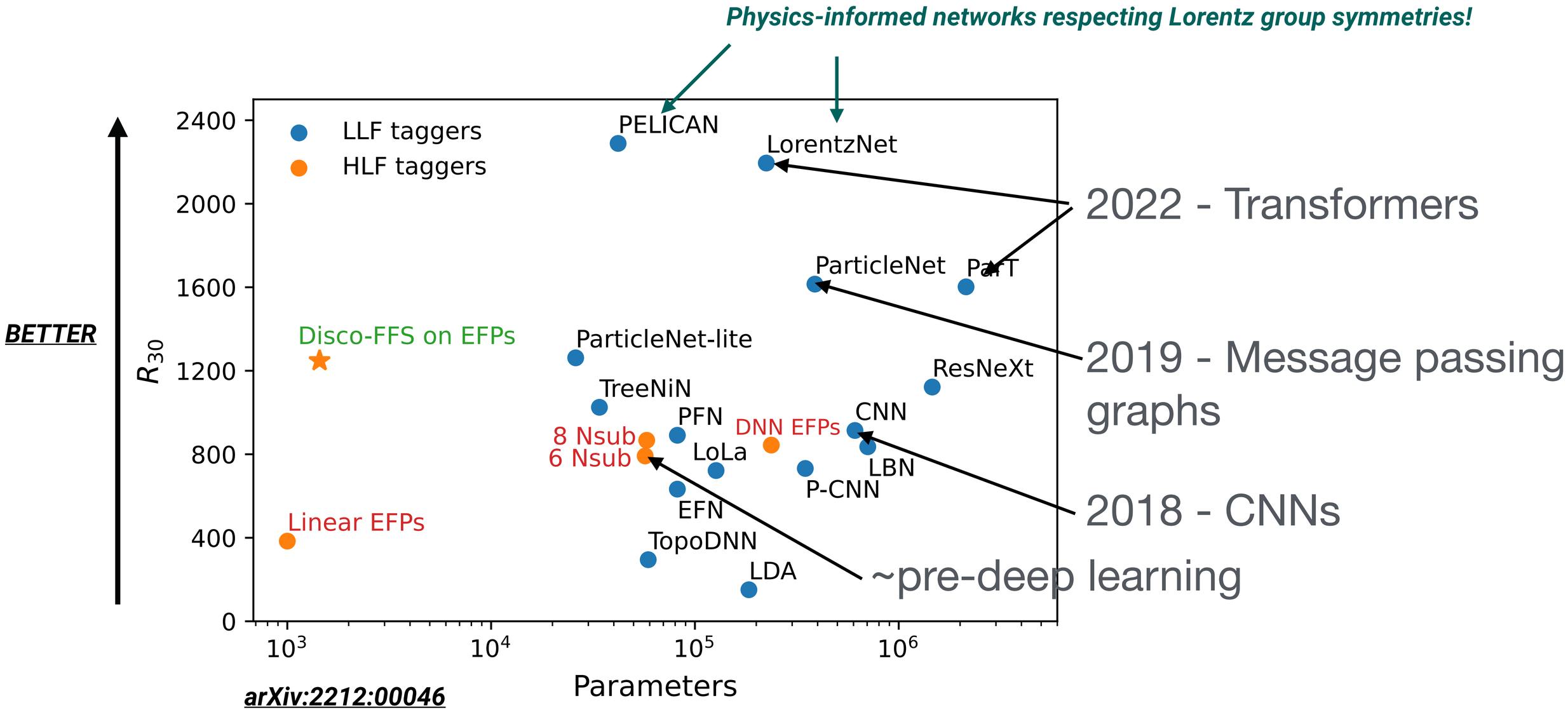
Ideally

- Quantization
- Pruning
- Parallelisation
- Knowledge distillation



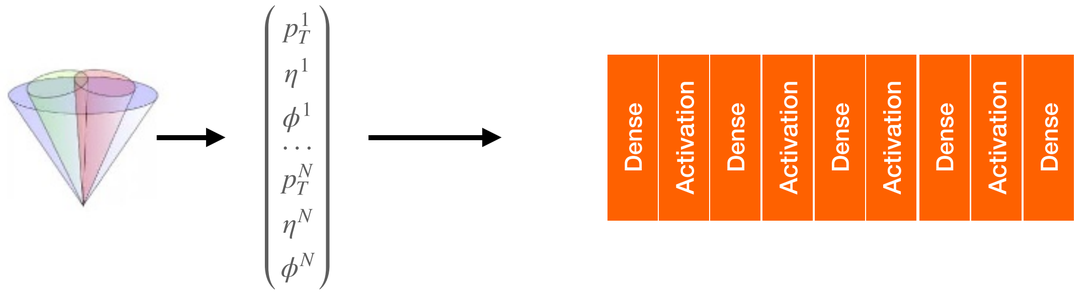
Reality

ASIDE: FLOPS versus trainable parameters!

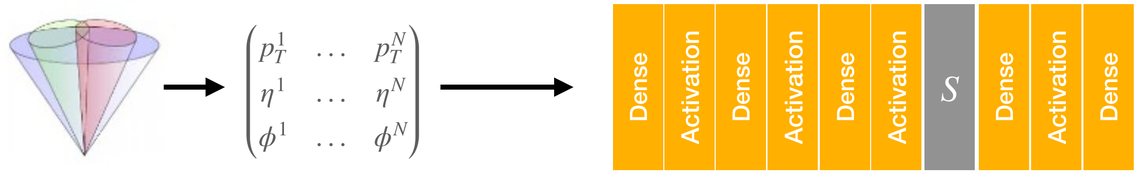


ASIDE: FLOPS versus trainable parameters!

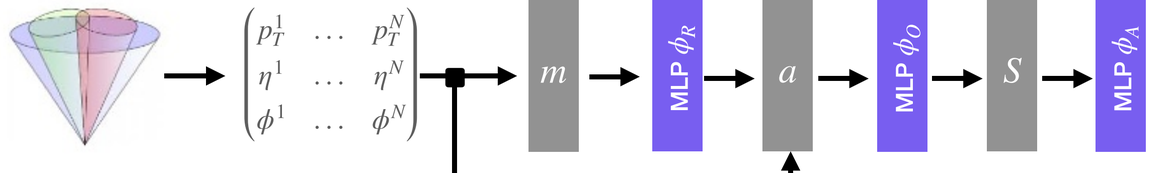
a) Multilayer Perceptron MLP



b) Deep Sets DS



c) Interaction Network IN



Architecture	Constituents	Parameters	FLOPs
MLP		26 826	53 162
DS	8	3461	36 805
IN		3347	37 232
MLP		20 245	40 485
DS	16	3461	71 109
IN		3347	140 432
MLP		24 101	48 197
DS	32	3461	139 717
IN		7400	109 556

Example FLOPs calculation

(Or use torchprofile and TensorFlow Profiler)

MLP: 16 → 64 → 32 → 32 → 5

Assumptions:

- 1 weight = 1 multiply + 1 add = **2 FLOPs**
- 1 activation ≈ **1 FLOP per neuron**

Layer-wise FLOPs:

1. Input → Hidden1 (16 → 64)

- Connections: 16 × 64 = 1024
- FLOPs: 2 × 1024 + 64 = **2112**

2. Hidden1 → Hidden2 (64 → 32)

- Connections: 64 × 32 = 2048
- FLOPs: 2 × 2048 + 32 = **4128**

3. Hidden2 → Hidden3 (32 → 32)

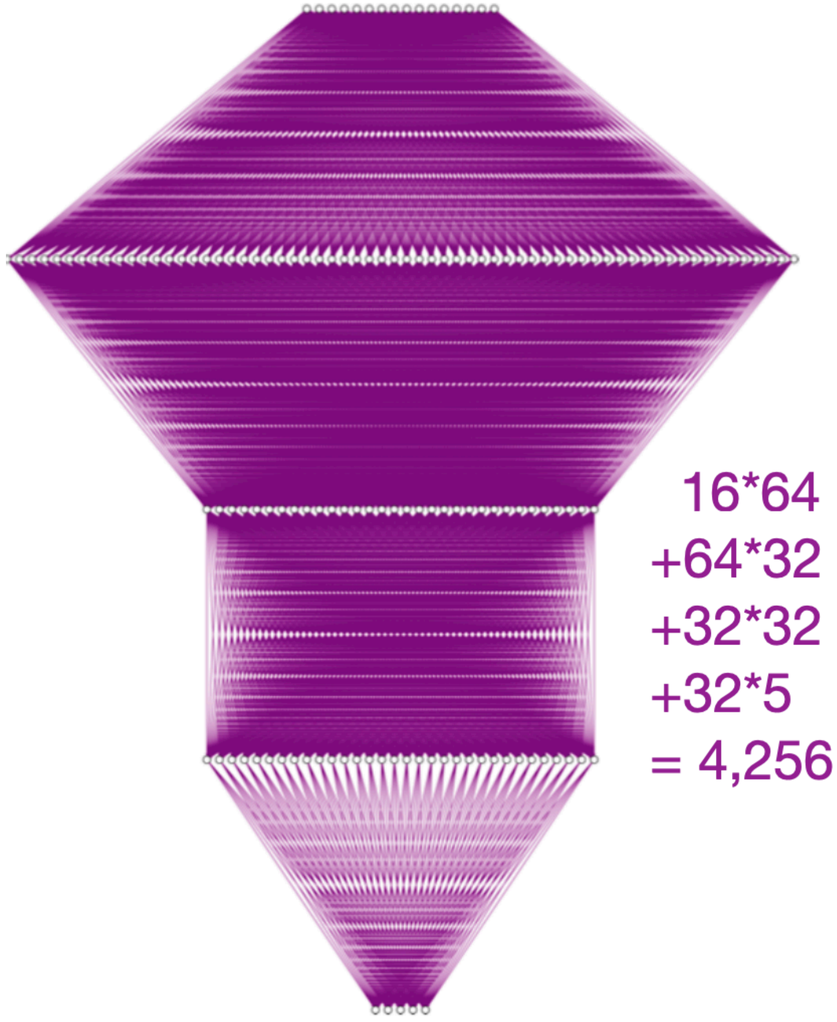
- Connections: 32 × 32 = 1024
- FLOPs: 2 × 1024 + 32 = **2080**

4. Hidden3 → Output (32 → 5)

- Connections: 32 × 5 = 160
- FLOPs: 2 × 160 + 5 = **325**

Total FLOPs (Forward Pass):

2112 + 4128 + 2080 + 325 = 8645 FLOPs



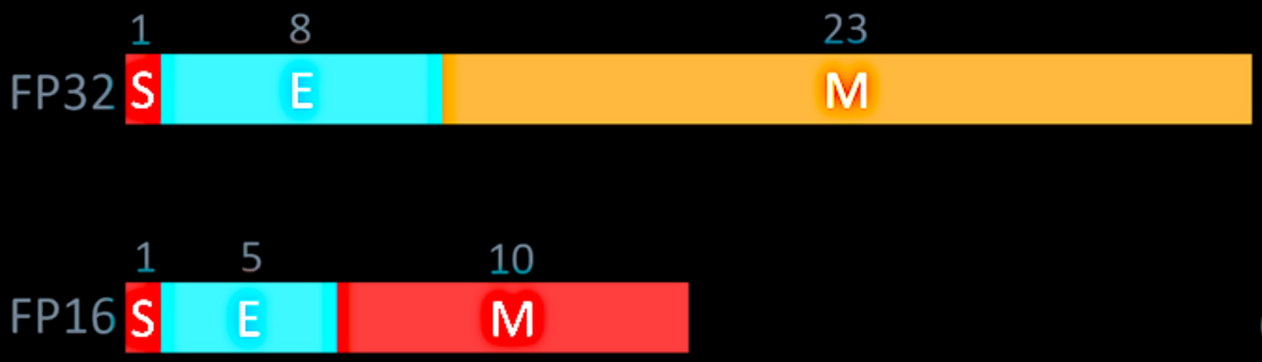
16*64
+64*32
+32*32
+32*5
= 4,256 synapses

Eg floating point 16

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

GPU & CPU

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

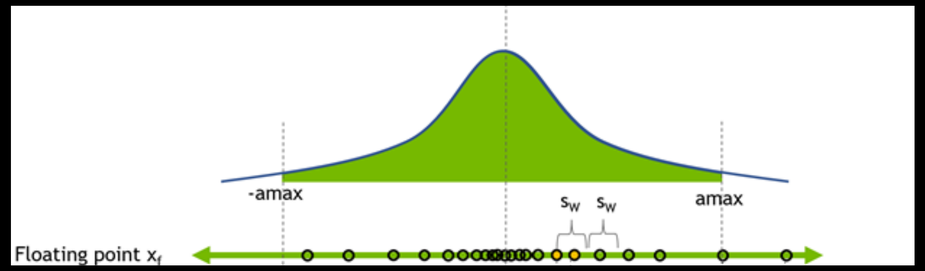
Speed:
~ FP16 x8 times faster than FP32

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

FPGA and ASIC

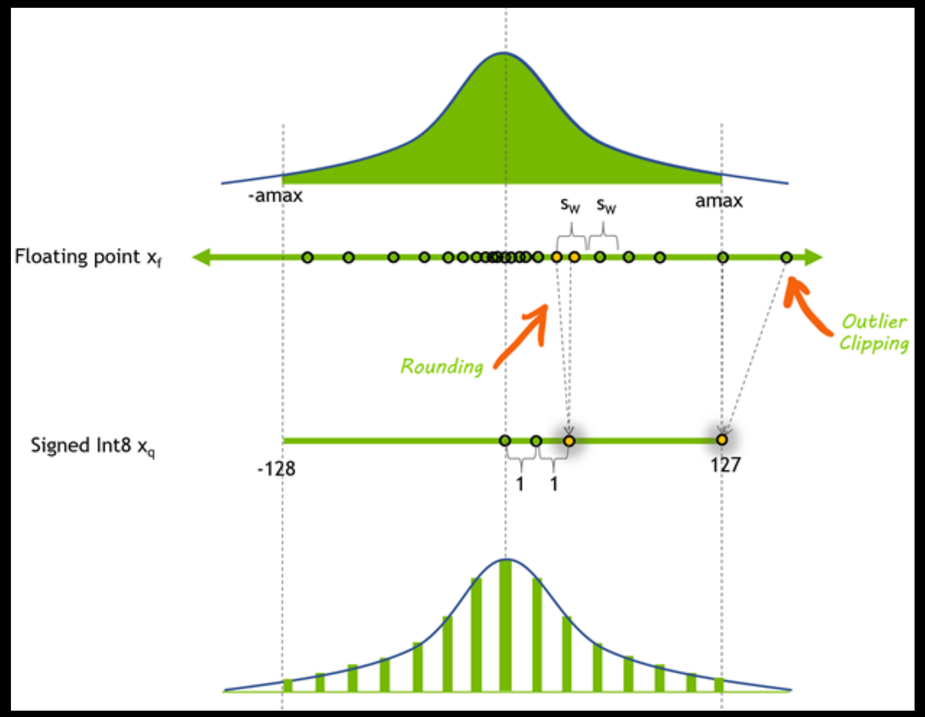
Fixed-point arithmetic:
Any precision!

Quantization



Floating point 32:
4B numbers in [-3.4e38, +3.4e38]

Quantization

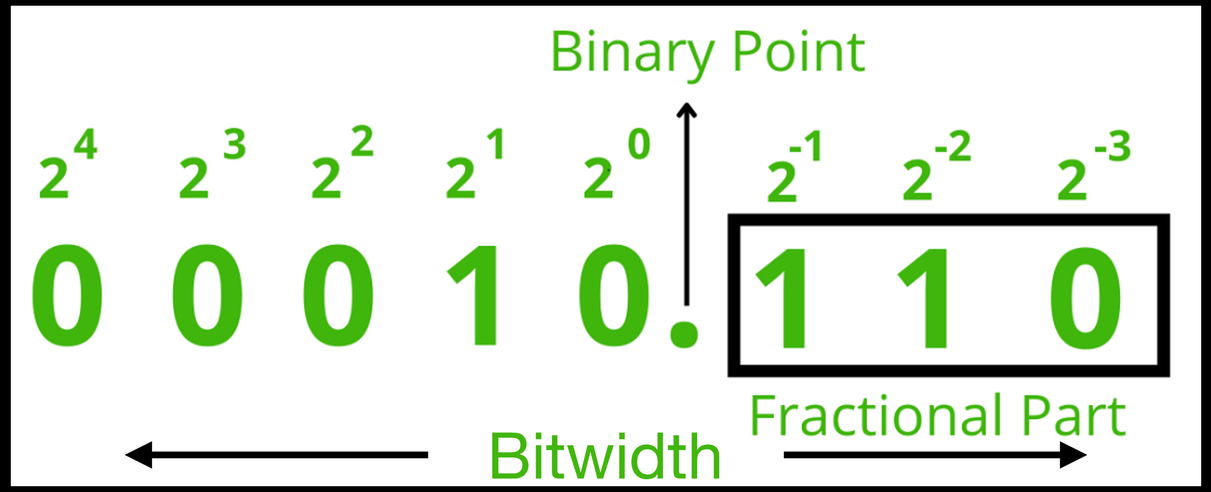


Quantising:
int8 2^8=256 numbers in [-128,127]

$$x_q = Clip(Round(\frac{x_f}{scale}))$$

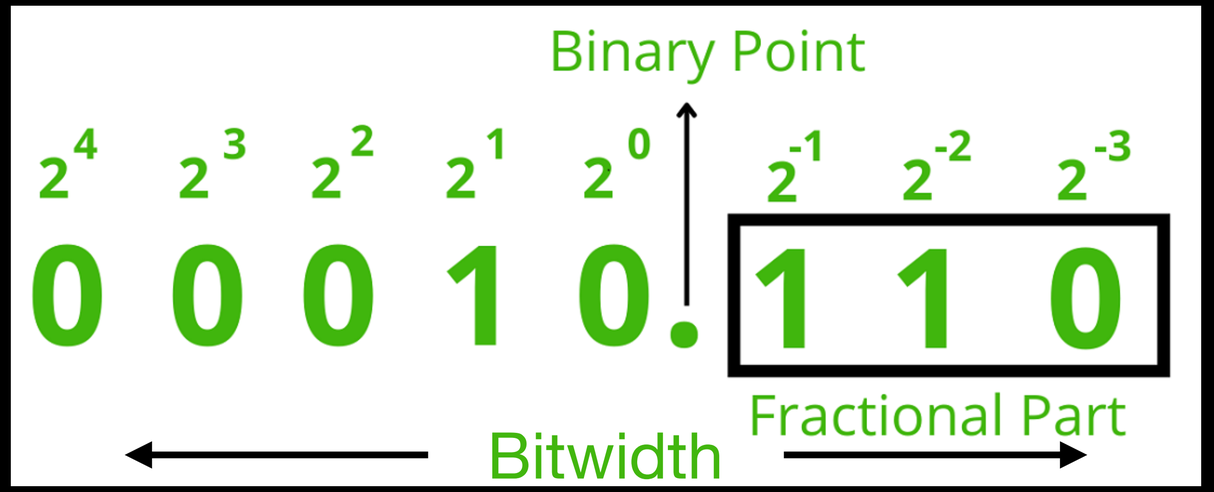
Fixed-point $< W, I >$

a way to express fractions with integers! E.g to represent 2.75:



$$= 2^4 \cdot 0 + 2^3 \cdot 0 + 2^2 \cdot 0 + 2^1 \cdot 1 + 2^0 \cdot 0 + 2^{-1} \cdot 1 + 2^{-2} \cdot 1 + 2^{-3} \cdot 0 = 2.75$$

Fixed-point $< W, I >$



Trade off: range (integer bits) and precision (fractional bits). E.g $< 8,0 >$:

$$\text{Precision} = \frac{1}{2^F} = \frac{1}{2^8} = 0.00390625$$

$$\text{Range} = [-2^0, -2^0 - 1] = [-1,0]$$

EXTRA

- On your CPU (and GPU) you are probably familiar with integer and floating point numerical formats
- Integer represents integer values in a fixed number of bits (e.g. 32, 16, 8)
- Floating point represents values with a mantissa and exponent : $m \cdot 2^e$
  - It's like scientific notation with binary
- Simplified examples, representing the number 113, ignoring sign (positive values only)

Base 10 integer
113_{10}

10^2	10^1	10^0
1	1	3

Base 10 floating point
(3 digit mantissa,
2 digit exponent)
$1.13 \times 10^{2.0}$

10^0	10^{-1}	10^{-2}		10^0	10^{-1}
1 .	1	3	· 10^	2 .	0

Base 2 integer
0b01110001

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	1	1	1	0	0	0	1

Base 2 floating point
(4 bit mantissa,
4 bit exponent)
$14 \times 2^8 = 112$
(Approximate)

2^3	2^2	2^1	2^0		2^3	2^2	2^1	2^0
1	1	1	0	· 2^	1	0	0	0

EXTRA

Fixed Point

- **Exercise**: what are the values ("in base 10") of these 8 bit unsigned fixed-point numbers?
  - What are the maximum and minimum values?

a.

Integer				Fraction			
2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}
0	1	1	1	0	0	0	1

b.

Integer						Fraction	
2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}
0	1	1	1	0	0	0	1

c.

Fraction							
2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}	2^{-9}	2^{-11}	2^{-11}
0	1	1	1	0	0	0	1

Example

Integer				Fraction			
2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}
1	0	0	1	0	1	0	1

$= 9\ .3125_{10}$

EXTRA

Fixed Point

a.

Integer				Fraction			
2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}
0	1	1	1	0	0	0	1

X = 7.0625
Max = 15.9375
Min = 0.0625

b.

Integer						Fraction	
2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}
0	1	1	1	0	0	0	1

x = 28.25
Max = 63.75
Min = 0.25

c.

Fraction							
2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}	2^{-9}	2^{-11}	2^{-11}
0	1	1	1	0	0	0	1

X = 0.05517578125
Max = 0.12451171875
Min = 0.00048828125

Example

Integer				Fraction			
2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}
1	0	0	1	0	1	0	1

= 9 .3125$_{10}$

EXTRA

Adding and subtracting

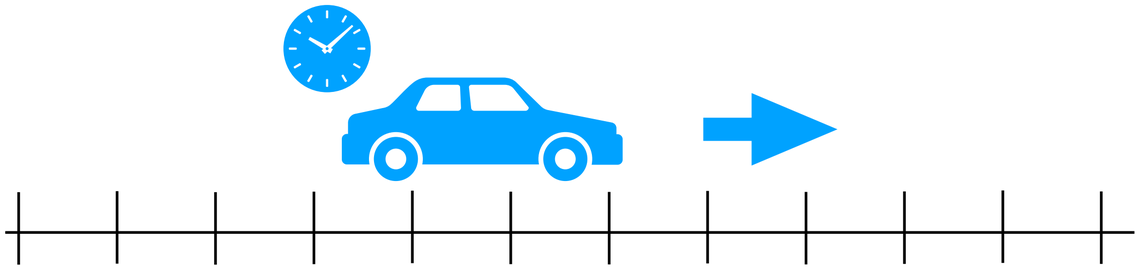
- Operand 0: F_0 fractional bits, I_0 integer bits (F_0 + I_0 width); Operand 1: F_1 fractional bits, I_1 integer bits (F_1 + I_1 width)

- Addition:
  - To safely add/subtract two fixed-point values, align the binary point first (HLS will do this automatically)
  - The result must accommodate the largest integer range and maximum fractional precision
  - Integer bits : $\max(I_0, I_1) + 1$ // +1 for carry
  - Fractional bits: $\max(F_0, F_1)$
  - Total width : $\max(I_0, I_1) + \max(F_0, F_1) + 1$
- Multiplication:
  - When multiplying two fixed-point numbers, both integer and fractional parts grow
  - The binary point is at the sum of the original positions
  - Integer bits : $I_0 + I_1$
  - Fractional bits: $F_0 + F_1$

EXTRA

Bit growth and domain knowledge

- It can be useful to use some domain knowledge about the realistic values to constrain the types
  - i.e. grow then shrink
  -



- Example: d = v · t distance: meters; time: seconds; velocity: meters per second
  - For a car, suppose v ∈ [0, 50] m/s (50 m/s ≃ 180 km/h) ; t ∈ [0, 20] s sampling device
  - Then v → `ap_ufixed<10,6>` and t → `ap_ufixed<10,6>` — values up to 63, steps of 1/16 ~ 0.06
  - Bitgrowth multiplication rule yields `ap_ufixed<20, 12>` — values up to 4095 m, steps of 1/256 ~ 0.004
  - But given our constraints the real maximum value could be 1000 m, and 0.004 m is too precise
  - Could clip to `ap_fixed<14, 10, AP_RND, AP_SAT>` for our real range and a reasonable precision, with rounding and saturation for safety — values up to 1023 m, steps of 1/16 ~ 0.06 m

Why bother?

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

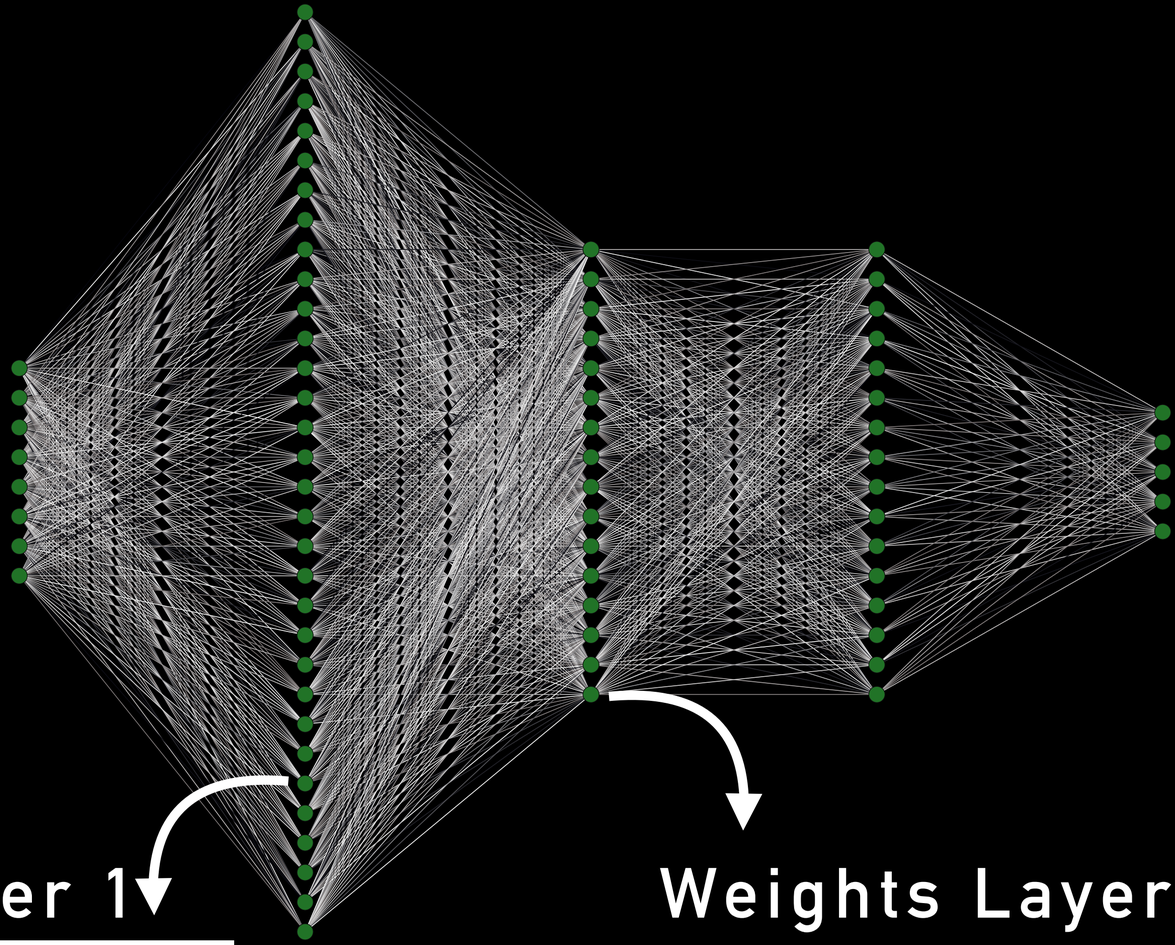
200x

30x

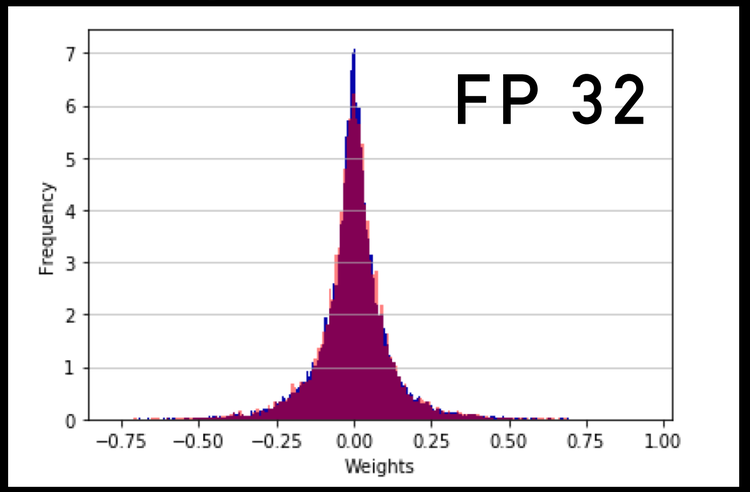
Trillions of quantized operations per second

Weights can stay **entirely on-chip**

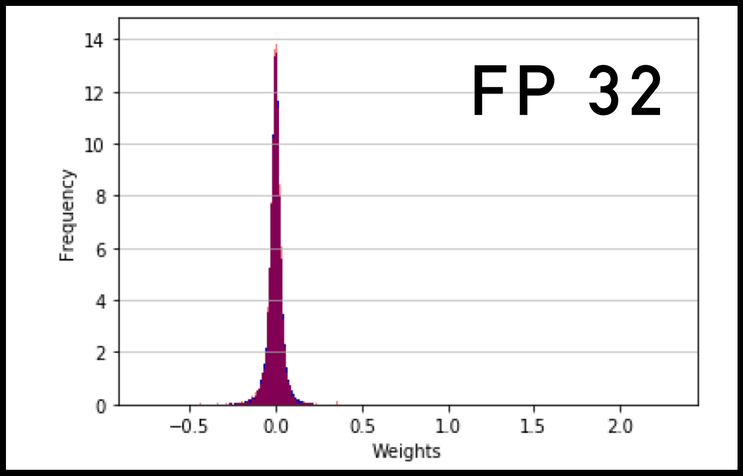
AMD UltraScale+ MPSoC ZU19EG (conservative estimates)



Weights Layer 1



Weights Layer 2



More number exercises

- What precision (bit width, integer bits) would you use for the model input and why?

- Example: I want to represent a number 0-120 to 3 decimals

  - Q1: “What power of 2 is just greater than my number?”
    log2(120) = 6.9 => 7 bits (2^7-1=127)

  - Q2: “What power of 2 is just less than 0.001?”
    -log2(10^{-3}) = 9.9 => 10 bits. (2^{-10}=0.0009766, 2^{-9}=0.00195)

  - Maximum number you can represent is: 2^7-1=127 (from integer part) + (1-2^{-10}=)for the fractional part

  - If its positive only I can keep it unsigned:
    ap_ufixed<17,7,AP_RND,AP_SAT>

Part 1: Getting started

- Can I tell hls4ml? Yes!
- Find your input layer. What is its name? (“input_1”)
- You can set the precision of that input!

```
input_precision = ‘ap_ufixed<17,7,AP_RND,AP_SAT>'
for layer in model.layers:
    layer_name = layer.__class__.__name__
    if layer_name in ["BatchNormalization", "InputLayer"]:
        config["LayerName"][layer.name]["Precision"] = input_precision
```

This is a fixed-point number with:
Total bits: 17
Integer bits: 7 (so, 7 integer + 10 fractional bits)

Rounding mode: AP_RND:
when a fixed-point operation produces a result that doesn't fit precisely into the target precision, the value is **rounded to the nearest representable number**.

Saturation mode: AP_SAT
When a value exceeds the representable range (e.g., overflows or underflows), instead of wrapping around (which would happen with AP_WRAP), the value is **saturated**:

- If it’s too **large**, it is clamped to the **maximum representable value**.
- If it’s too **small**, it is clamped to the **minimum representable value**.

input_precision = ‘ap_ufixed<17,7,AP_RND,AP_SAT>'

Saturation is expensive!
Additional logic complexity (comparisons logic (Is the result > max representable value?), output must be replaced etc.

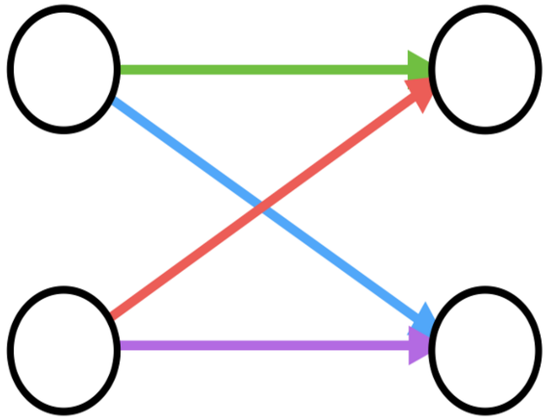
Part 1: Getting started

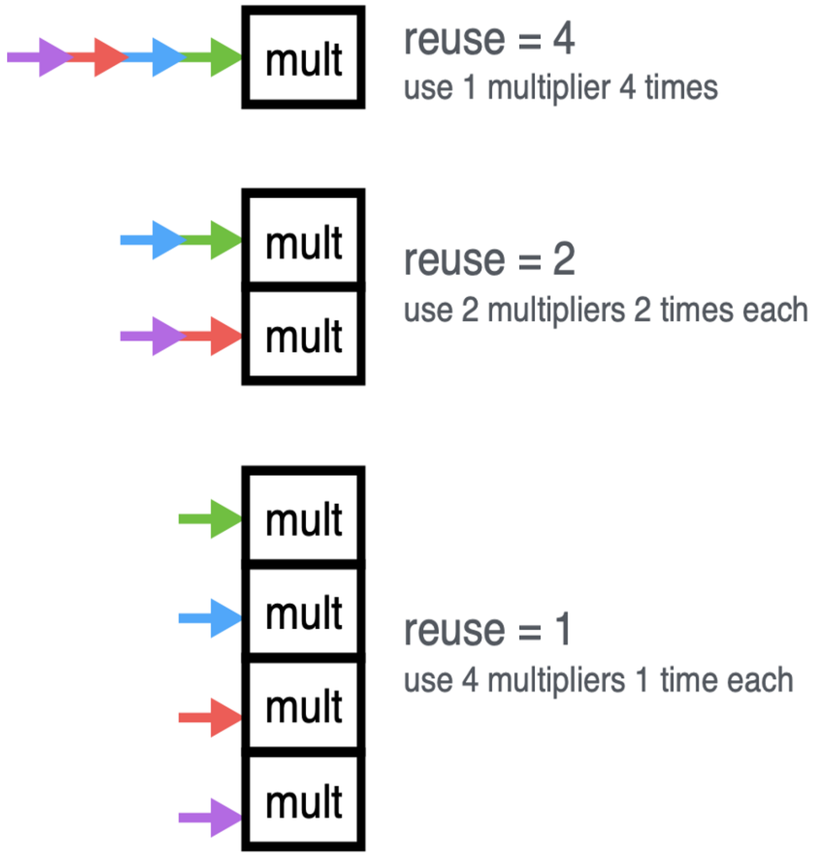
- Find your input layer. What is its name? (“input_1”)
- You can set the precision of that input!

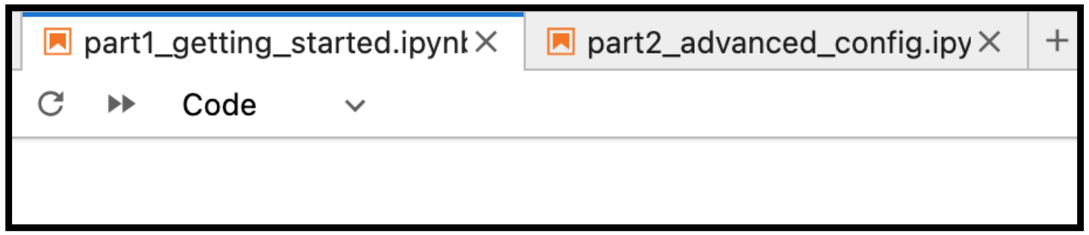
```
input_precision = 'ap_fixed<24,12,AP_RND,AP_SAT>'
for layer in model.layers:
    layer_name = layer.__class__.__name__
    if layer_name in ["BatchNormalization", "InputLayer"]:
        config["LayerName"][layer.name]["Precision"] = input_precision
```

Part 1 & 2

Parallelisation







Fully serial

Fewer resources
Lower throughput
Higher latency

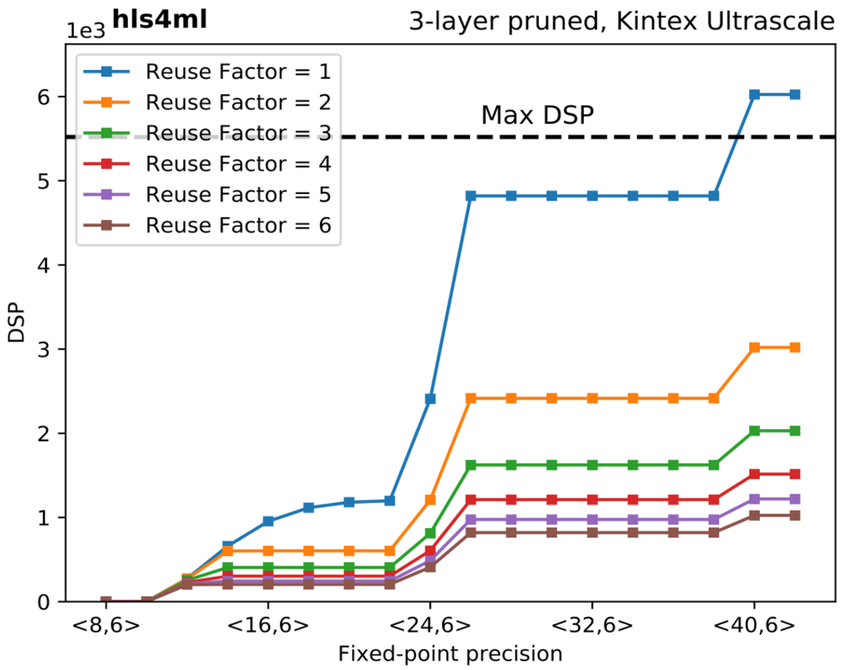
Fully parallel

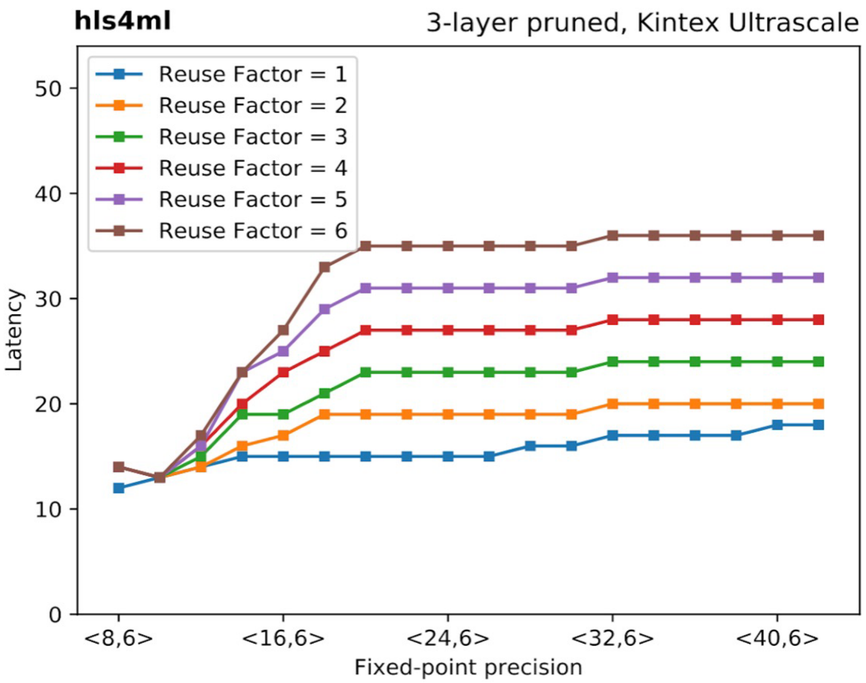
More resources
Higher throughput
Lower latency

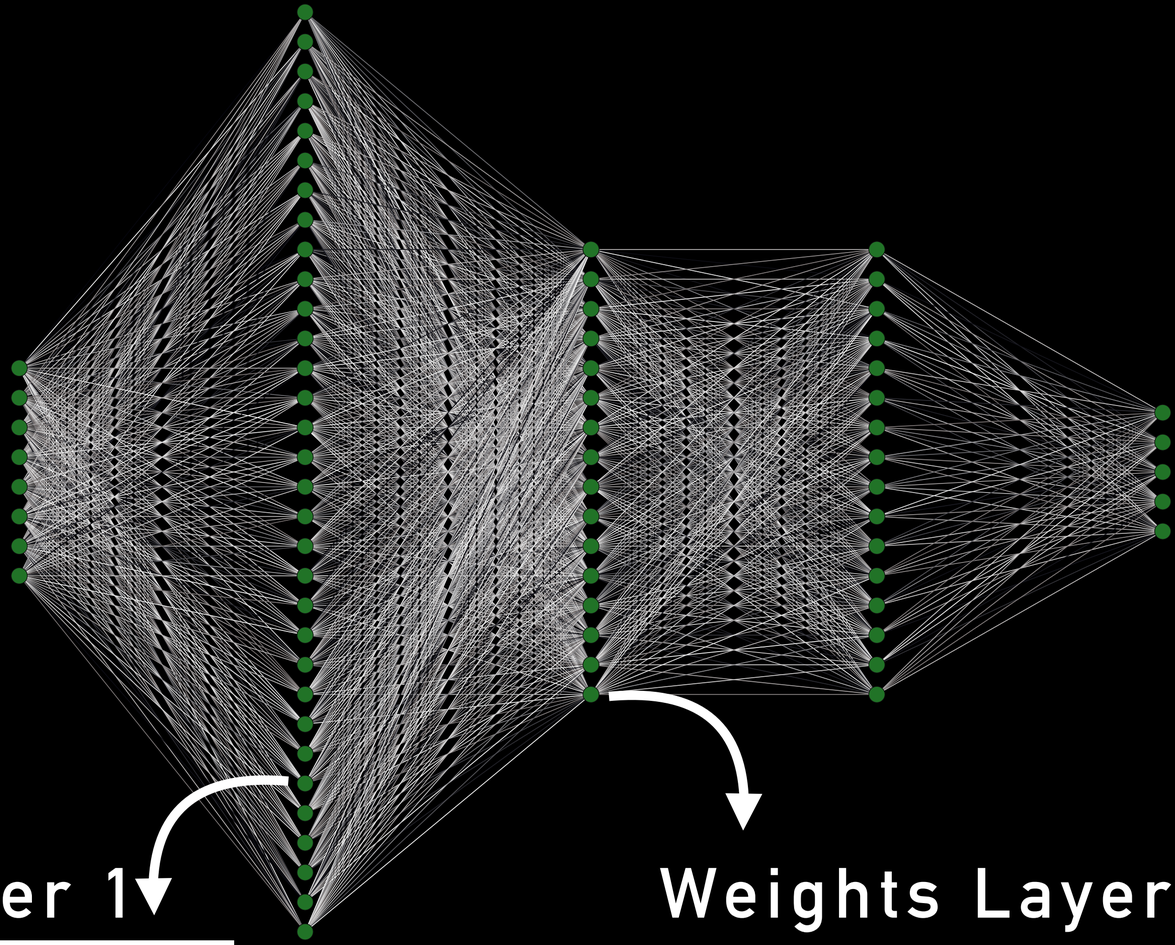
Fully parallel
Each mult. used 1x

Each mult. used 2x

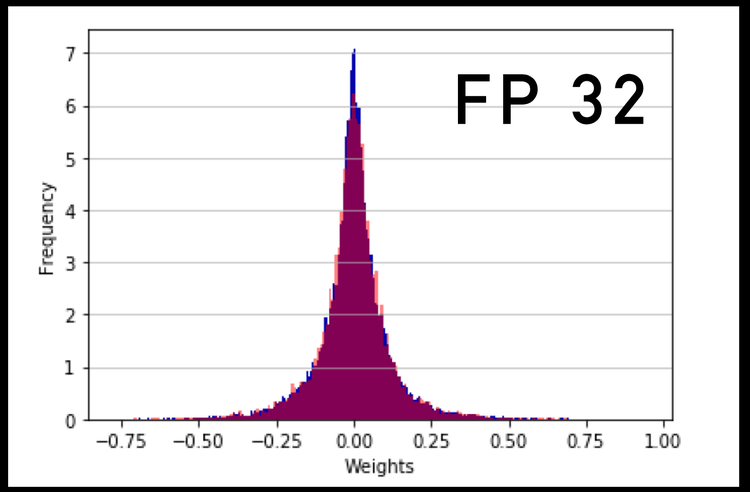
Each mult. used 3x



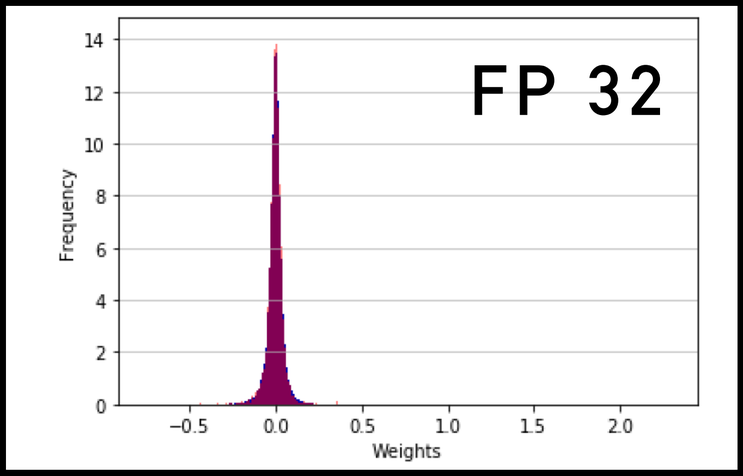




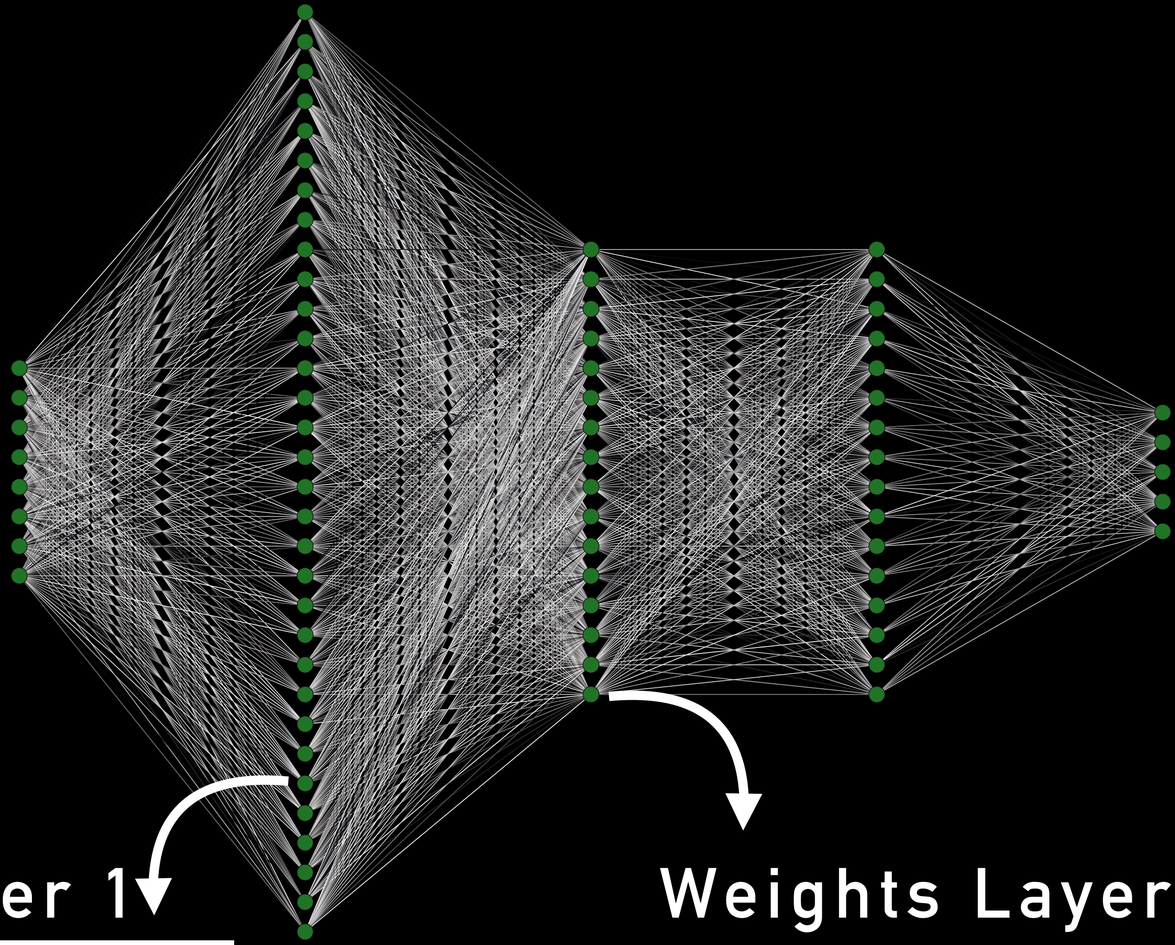
Weights Layer 1



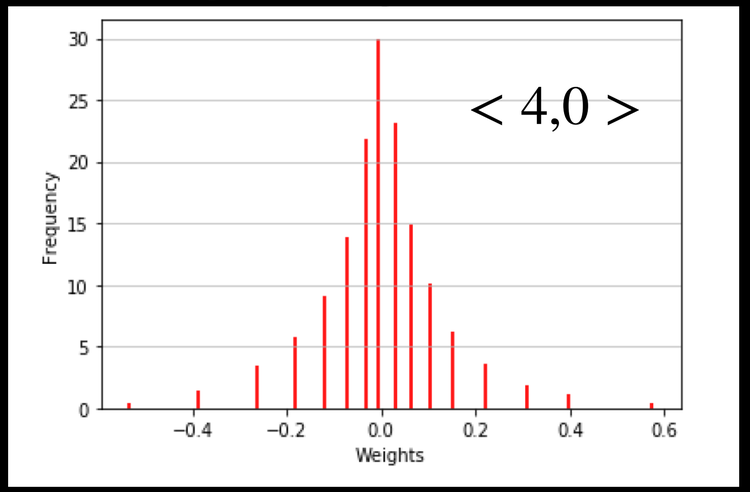
Weights Layer 2



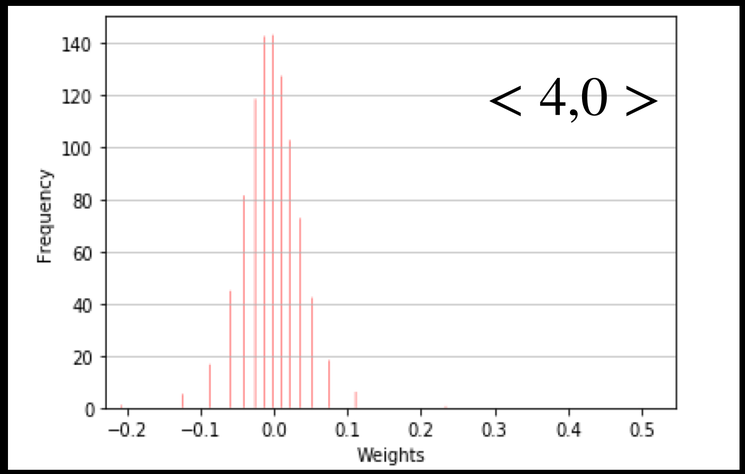
Fixed point

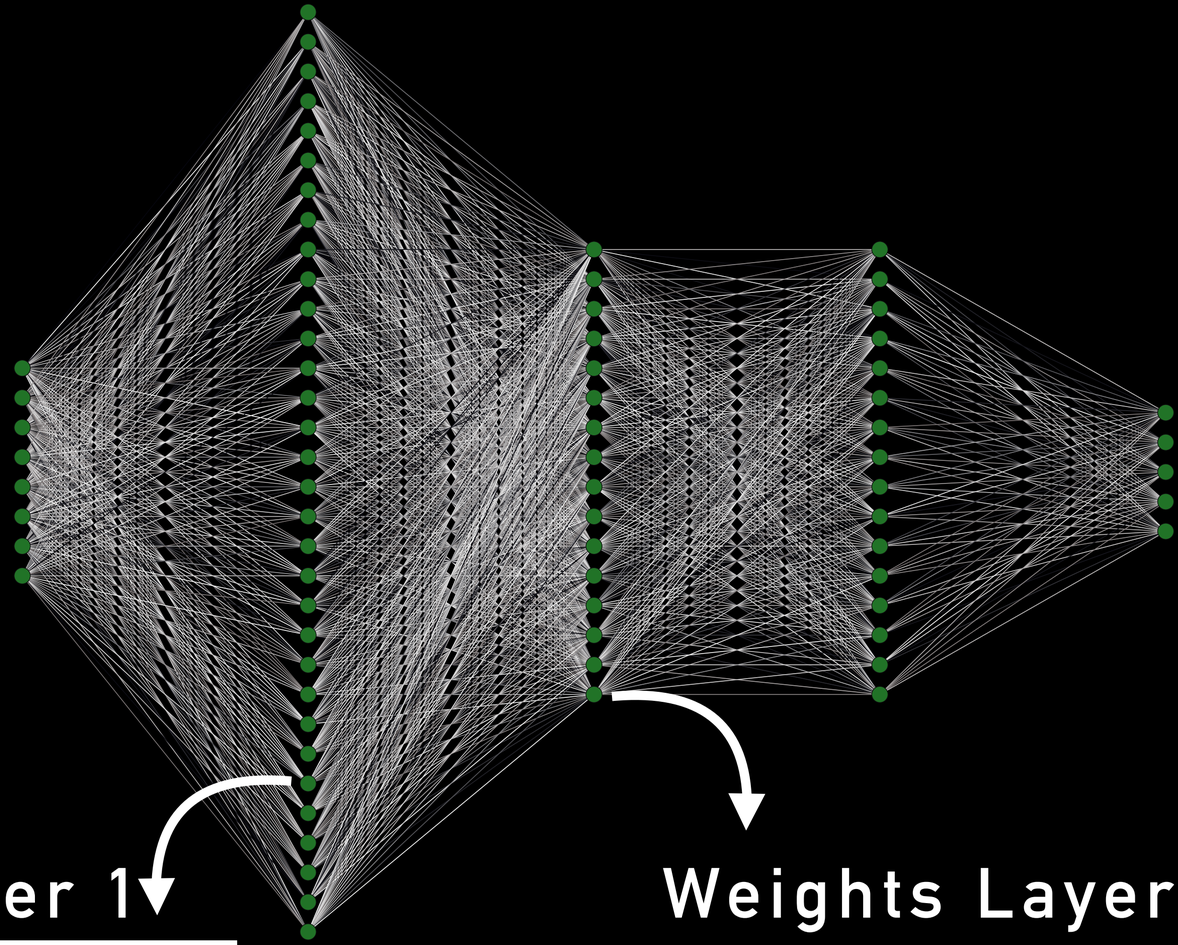


Weights Layer 1



Weights Layer 2





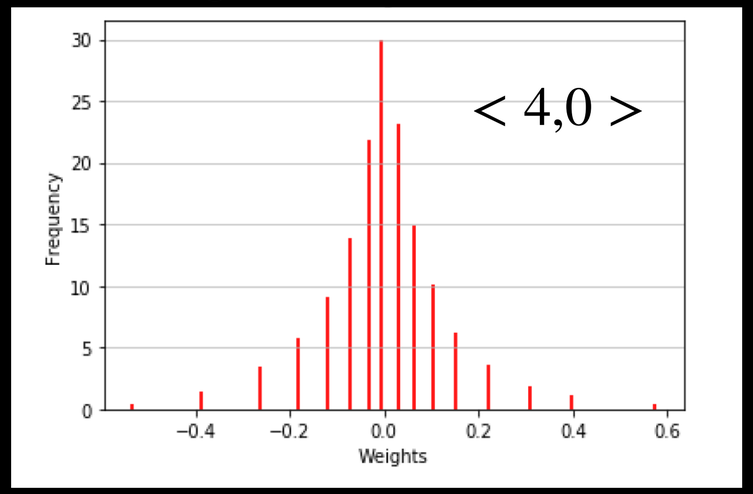
Fixed point

0101.1011101010

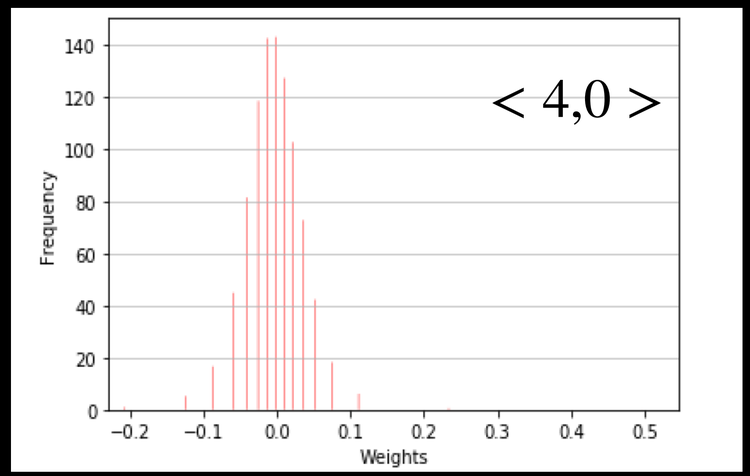
integer | fractional

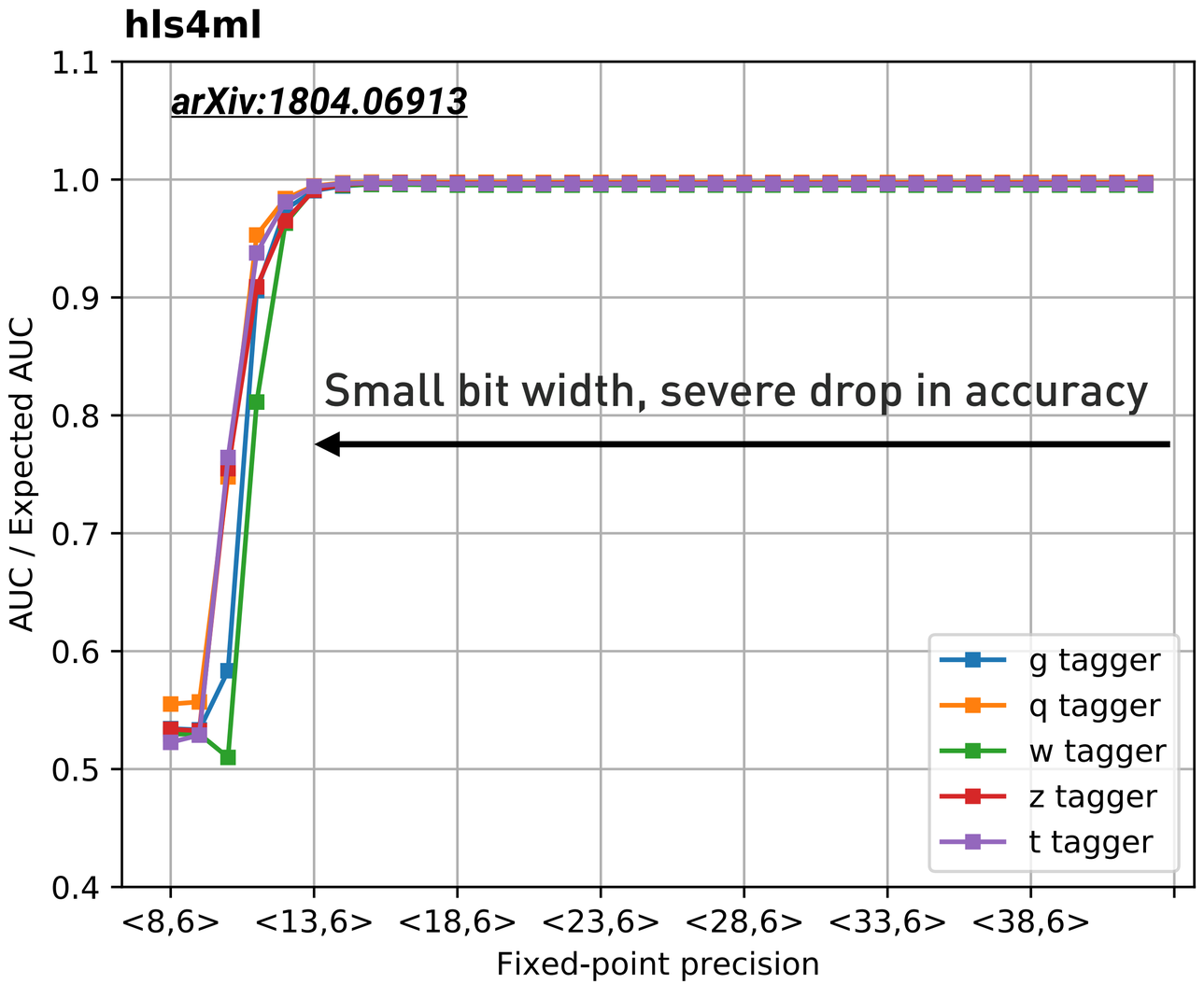
width

Weights Layer 1



Weights Layer 2

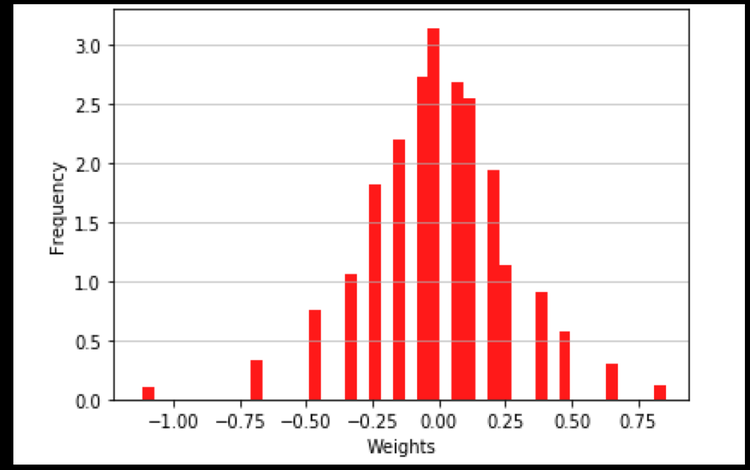


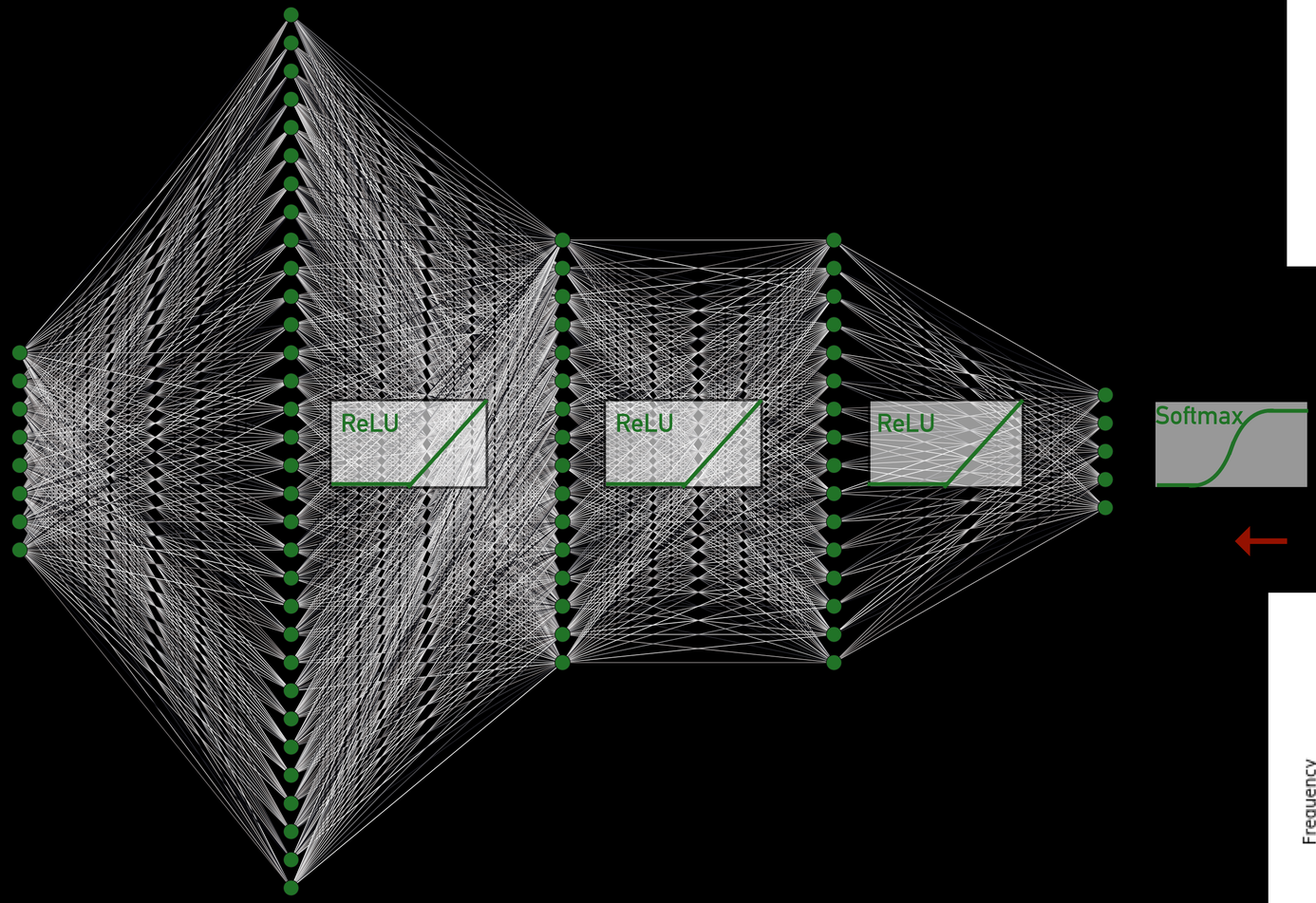

hls4ml
arXiv:1804.06913
Small bit width, severe drop in accuracy
AUC / Expected AUC
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.4
<8,6>
<13,6>
<18,6>
<23,6>
<28,6>
<33,6>
<38,6>
Fixed-point precision
g tagger
q tagger
w tagger
z tagger
t tagger

hls4ml + Google

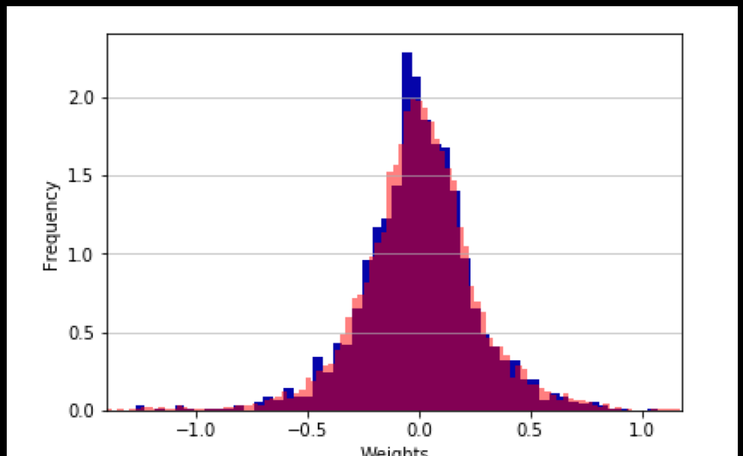
Quantization-aware training

Forward pass →





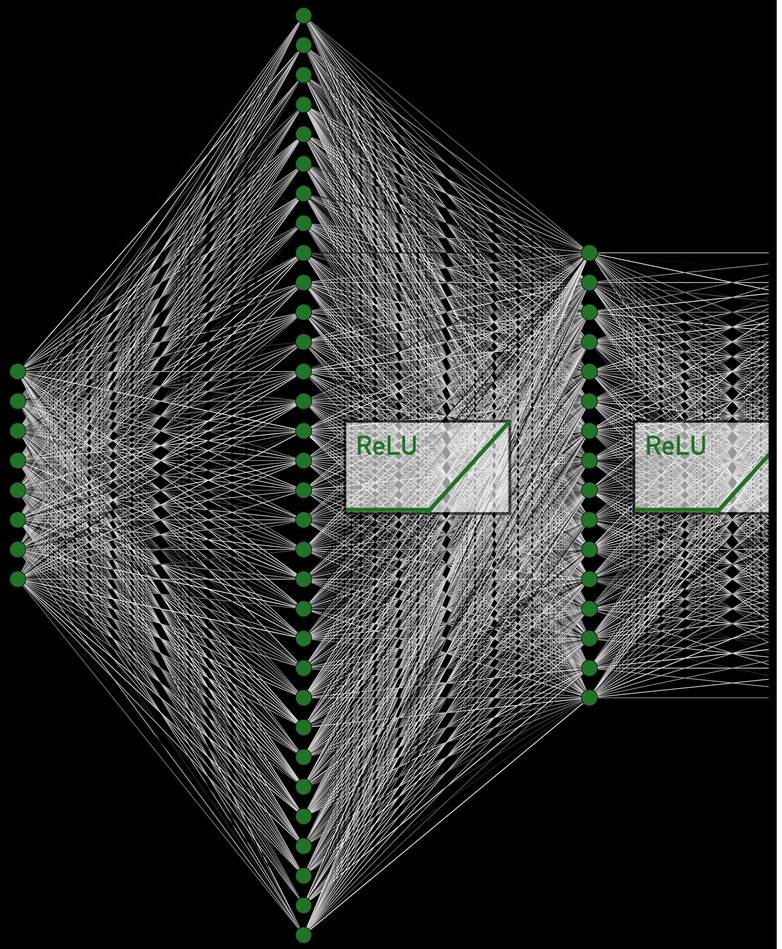
← Back propagation



Nature Machine Intelligence 3 (2021)

hls4ml + Google

Quantization-aware training



Nature Machine Intelligence 3 (2021)

Exercise 4

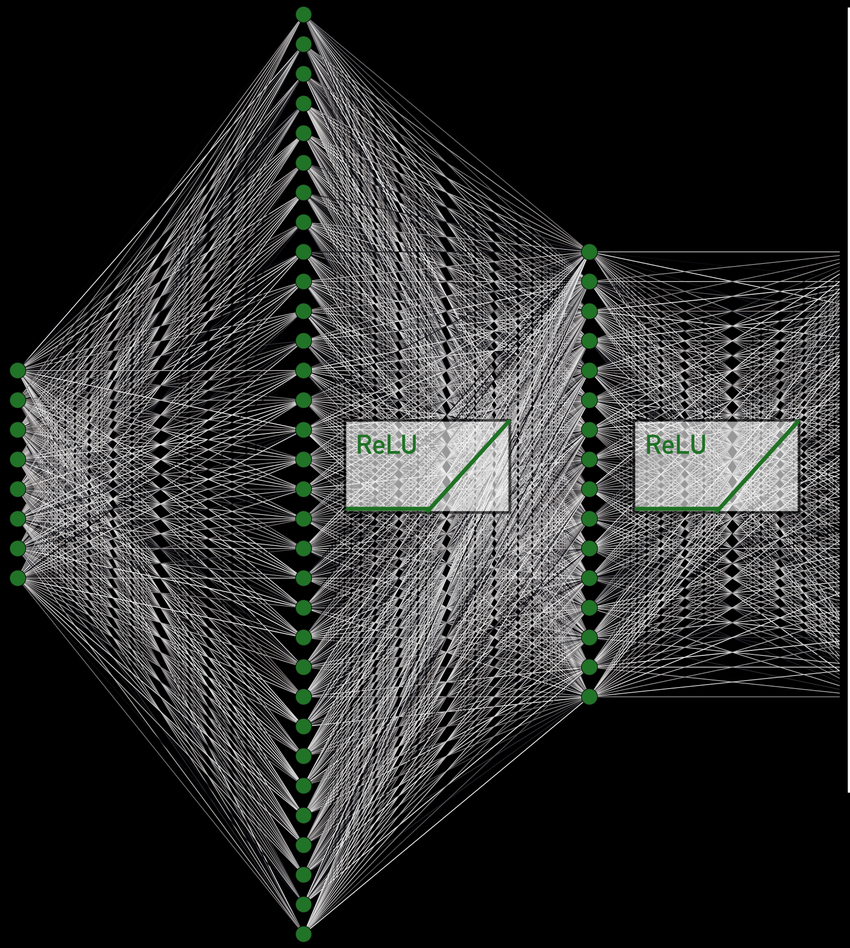
```
from tensorflow.keras.layers import ...tion
from qkeras import quantized_bits
from qkeras import QDense, QActivation
from qkeras import QBatchNormalization

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)
```

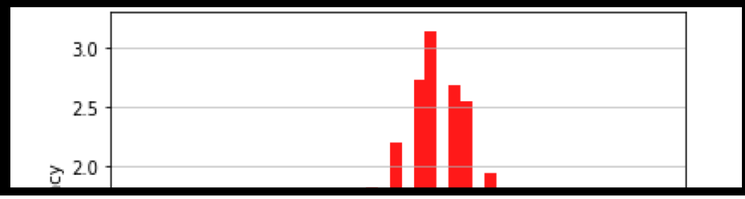
<6,0>

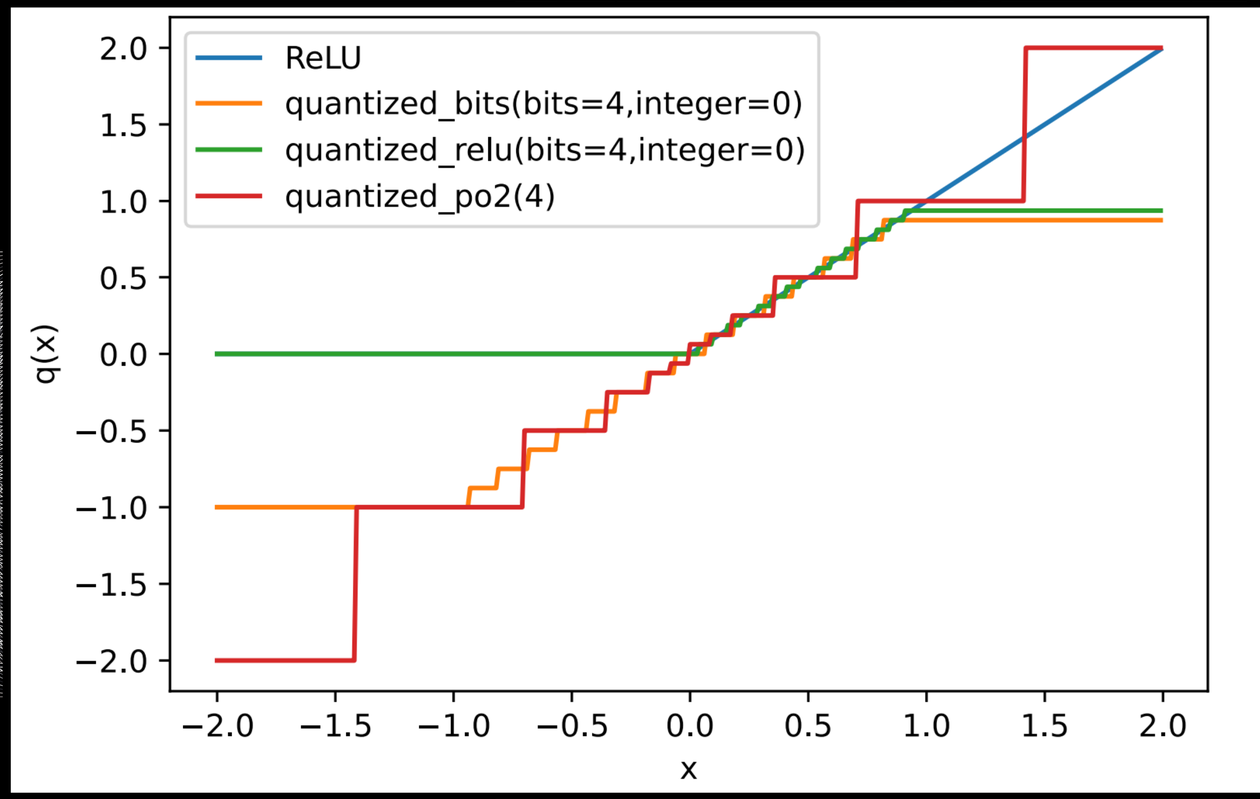
hls4ml + QKeras

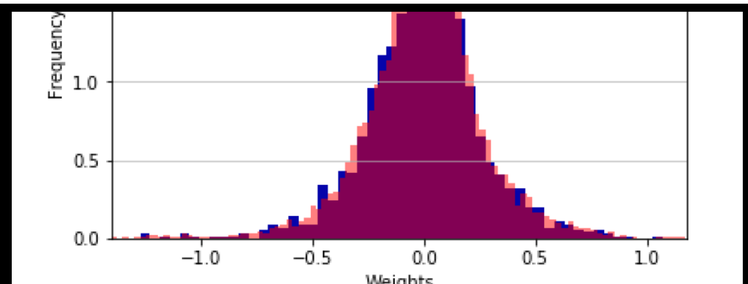
Quantization-aware training

Forward pass →



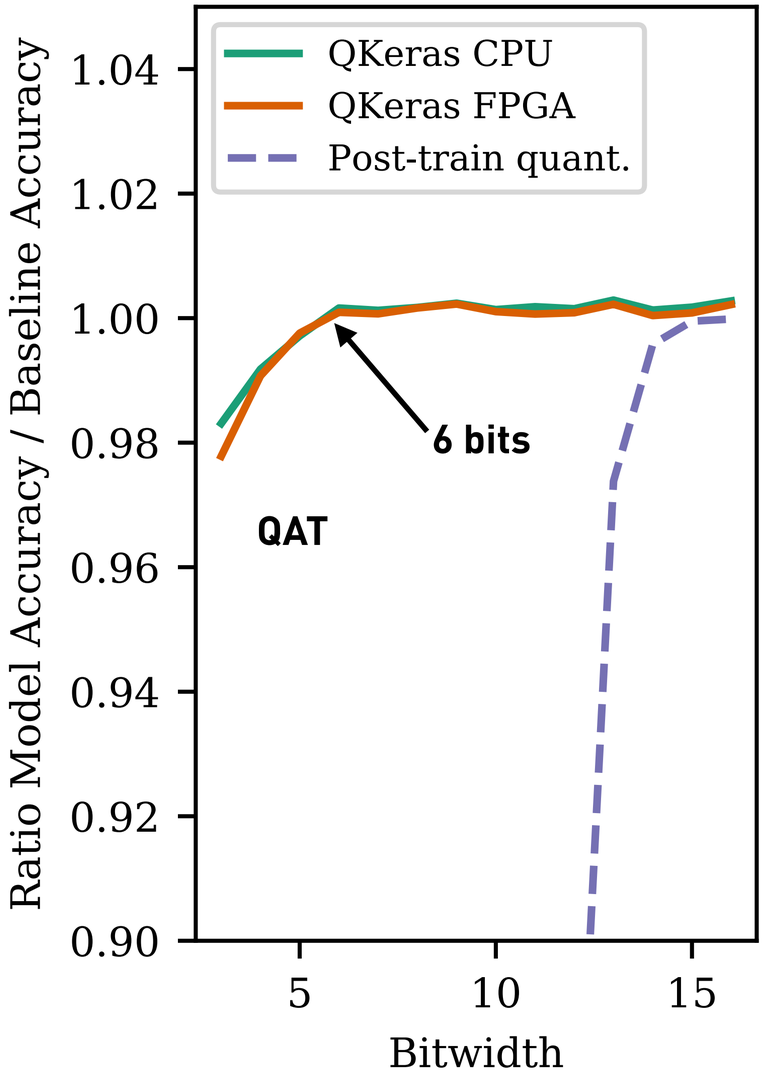


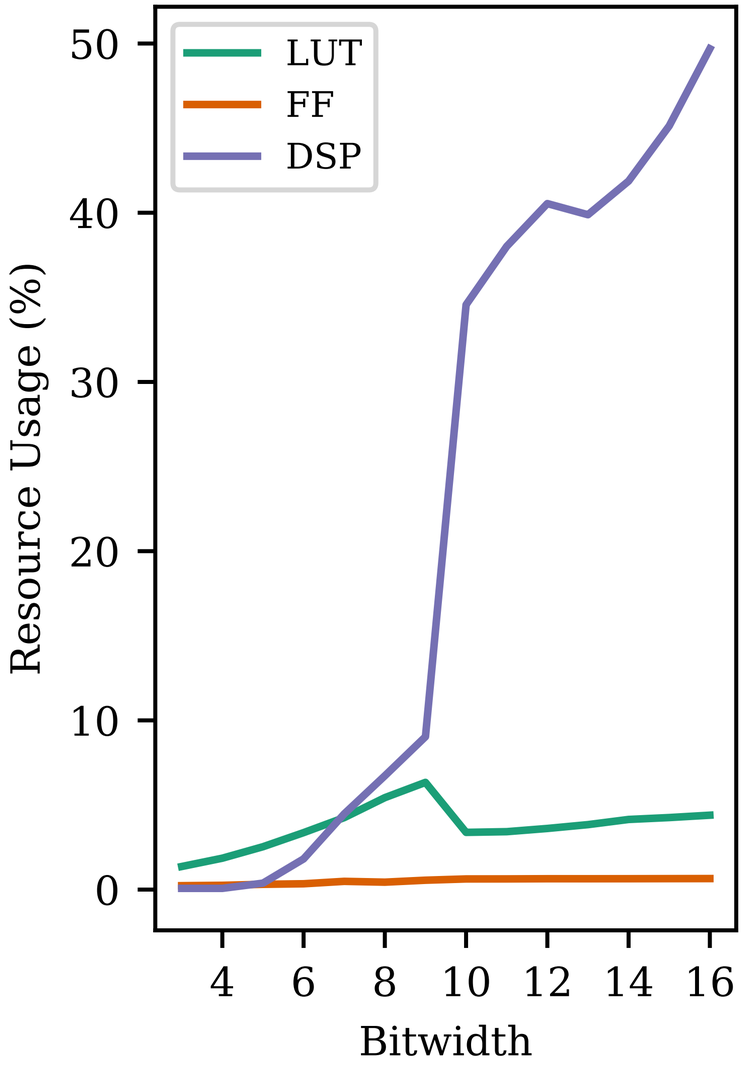




Nature Machine Intelligence 3 (2021)

Nature Machine Intelligence 3 (2021)





Heterogeneous quantization

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

- heterogeneous quantization

Heterogeneous quantization

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**

Heterogeneous quantization

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

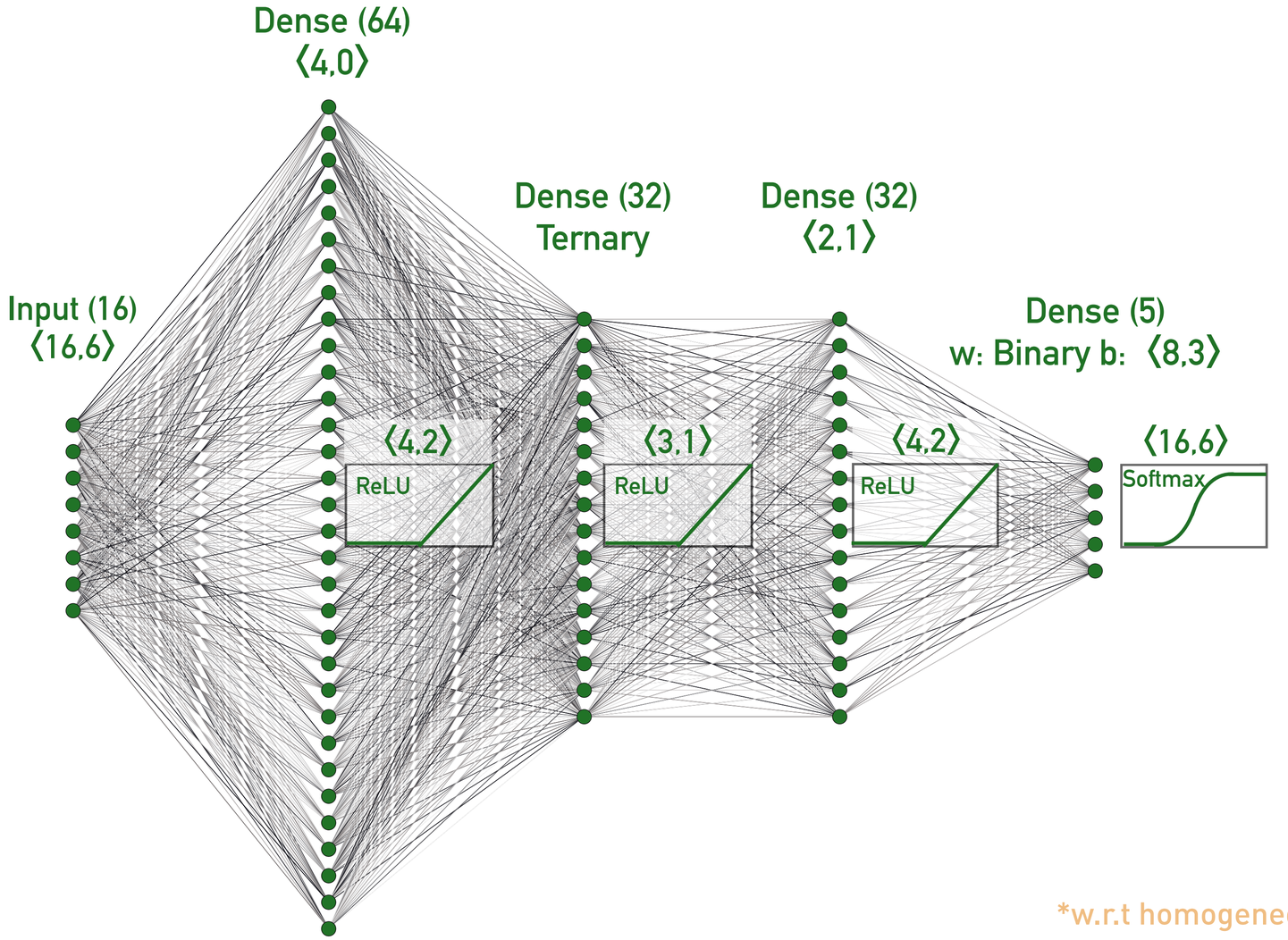
- Relative model size in bits
- Relative energy consumption in Watts

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

AutoQKeras

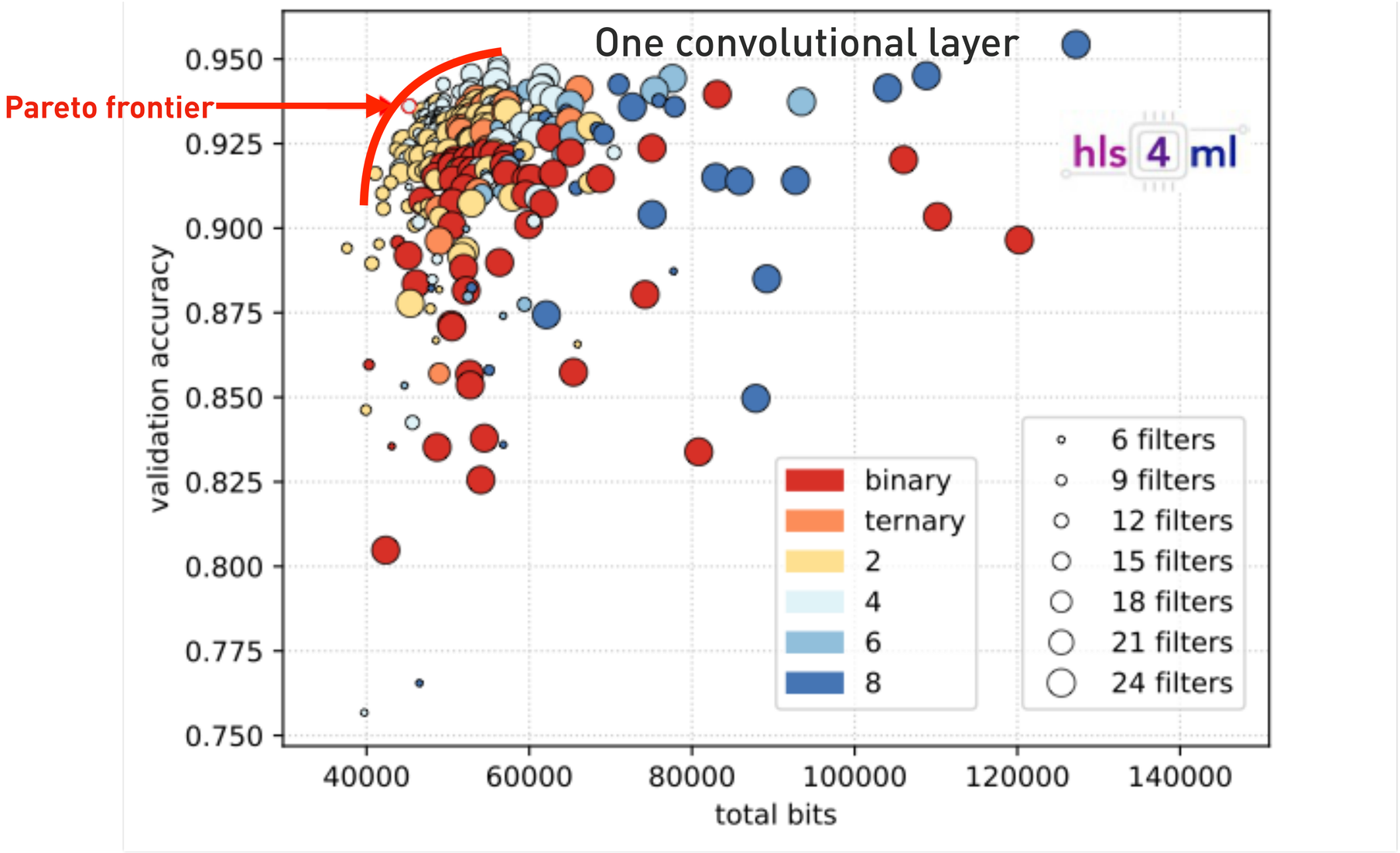


*w.r.t homogeneously quantized 6 bit model

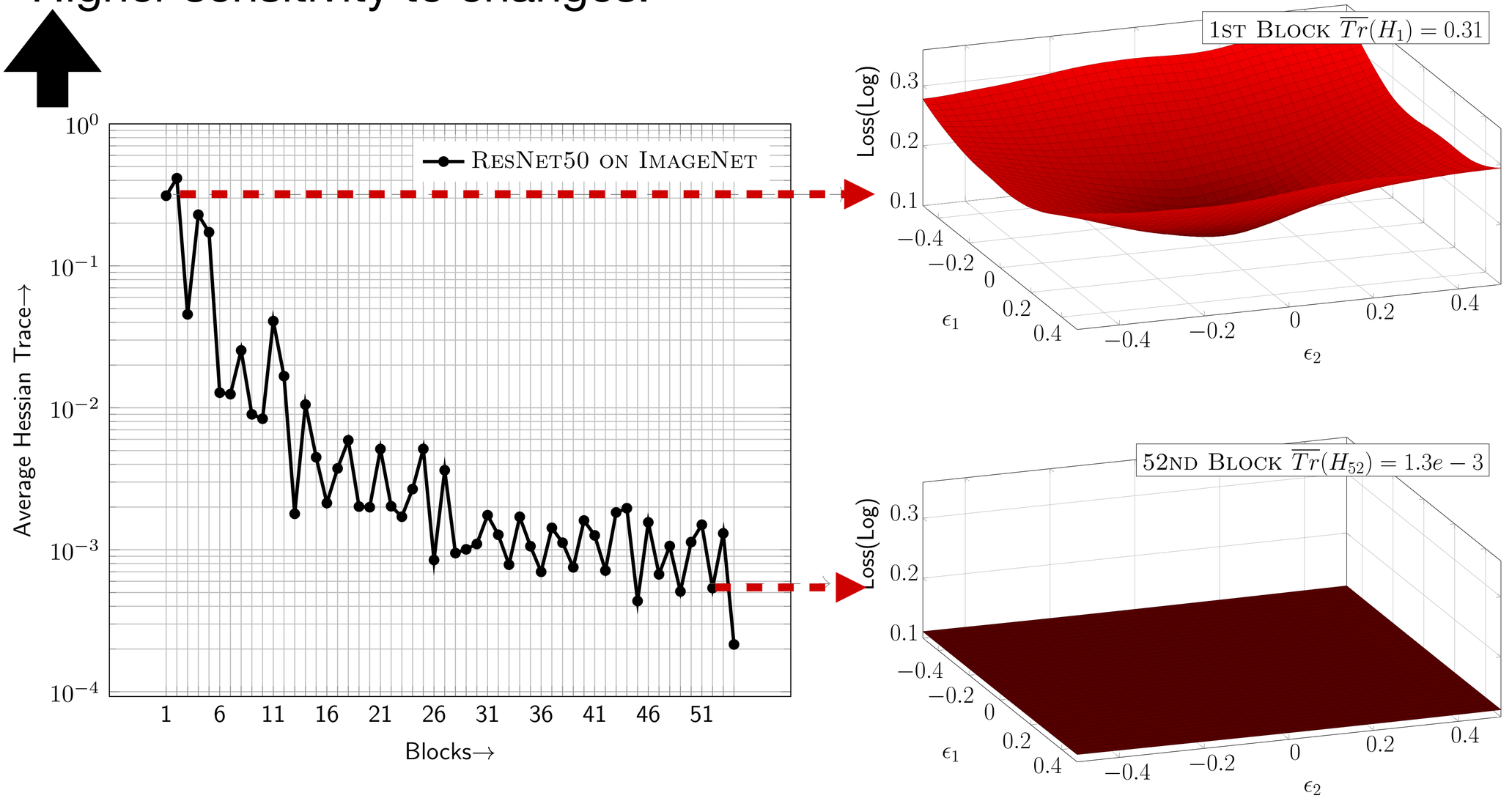
Model	Accuracy (%)	Precision								$\frac{E}{E_{Q6}}$	$\frac{\text{Bits}}{\text{Bits}_{Q6}}$
		Dense	ReLU	Dense	ReLU	Dense	ReLU	Dense	Softmax		
QE	72.3	⟨4, 0⟩	⟨4, 2⟩	Ternary	⟨3, 1⟩	⟨2, 1⟩	⟨4, 2⟩	w: Stoc. bin. b: ⟨8, 3⟩	⟨16, 6⟩	0.27	0.18*

DOI 10.1088/2632-2153/ac0ea1

AutoQKeras



Higher sensitivity to changes!



HAWQ-V2, Z. Dong et al. 2020

Exercise 4.1

HGQ: High Granularity Quantization

C.Sun et al. 2024

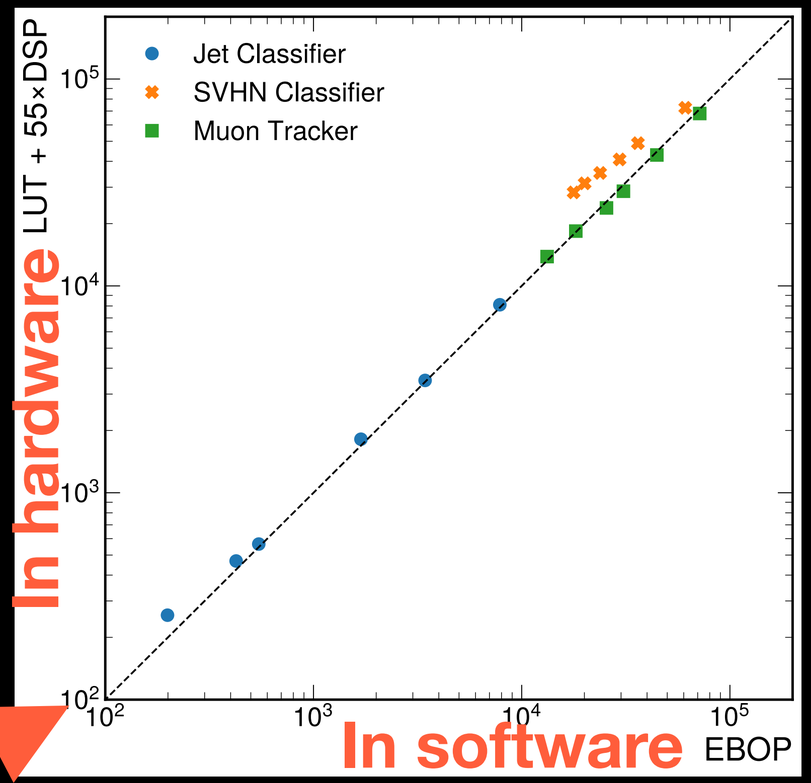
```
from tensorflow.keras.layers import Input
from HGQ import HQuantize, HDense

inp = Input((16,))
out = HQuantize(name='inp_q', beta=beta)(out)
out = HDense(64, activation='relu', beta=beta)(out)
out = HDense(32, activation='relu', beta=beta)(out)
out = HDense(32, activation='relu', beta=beta)(out)
out = HDense(5, activation='linear', beta=beta)(out)

hgq_model = Model(inp, out)
```

Exercise 4.1

HGQ: High Granularity Quantization

C.Sun et al. 2024

```
from tensorflow.keras.layers import Input
from HGQ import HQuantize, HDense

inp = Input((16,))
out = HQuantize(name='inp_q', beta=beta)(out)
out = HDense(64, activation='relu', beta=beta)(out)
out = HDense(32, activation='relu', beta=beta)(out)
out = HDense(32, activation='relu', beta=beta)(out)
out = HDense(5, activation='linear', beta=beta)(out)

hgq_model = Model(inp, out)
```



Proxy for resource cost

$$\mathcal{L} = \mathcal{L}_{\text{base}} + \beta \cdot \overline{\text{EBOPs}} + \gamma \cdot \text{L1}_{\text{norm}}$$

Making bitwidths differentiable! Fully heterogeneous quantization!

Exercise BONUS

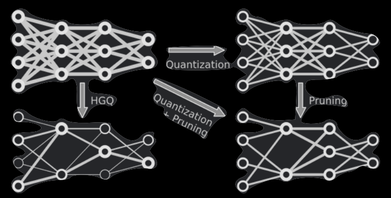
hls4ml + AMD Research Labs: QONNX

Frontend (QAT)

Backend (compiler)

Keras

PYTORCH Brevitas



HQG

QONNX

FINN AMD

hls4ml

Marius Köppel
A. Pappalardo et al. (2022)

OpenReview.net

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

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

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

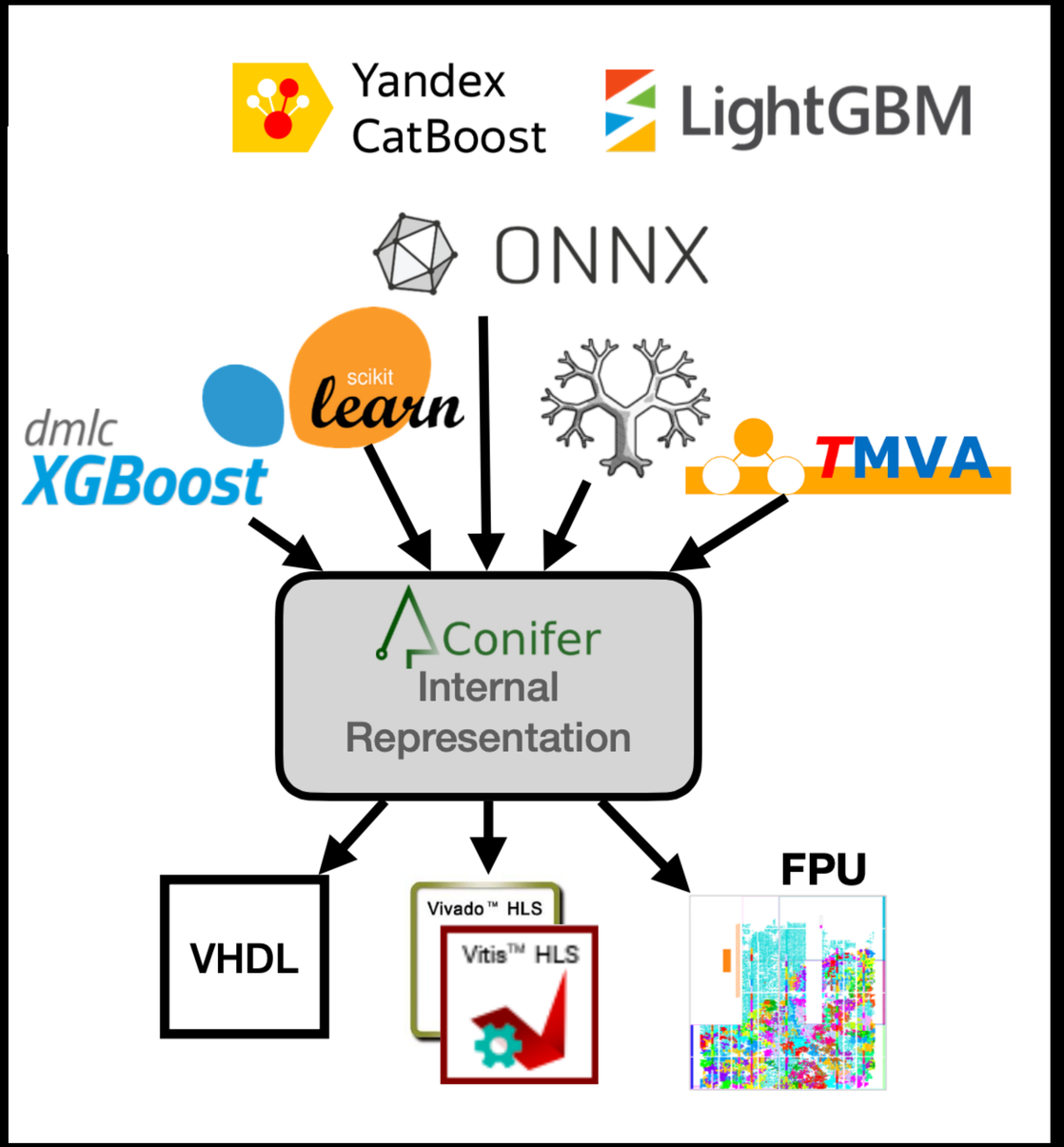
Submission history

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

Exercise BONUS

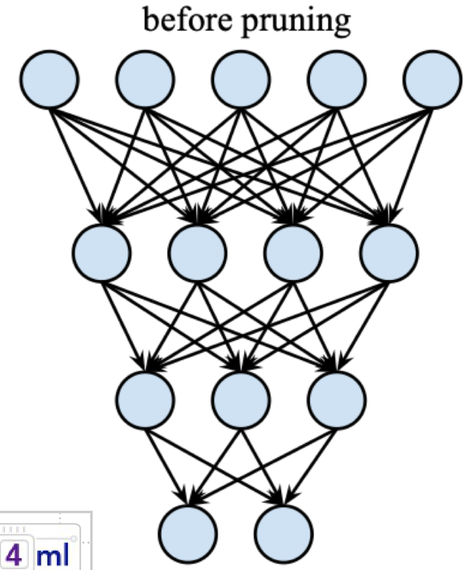
%VU9P	Accuracy	Latency	DSP	LUT
qDNN	75.6%	40 ns	22 (~0%)	1%
BDT	74.9%	5 ns	-	0.5%

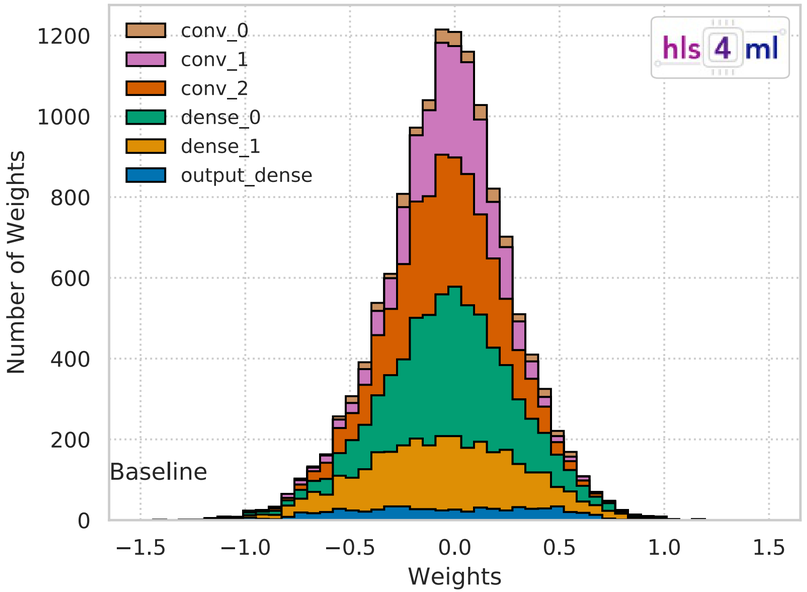


Exercise 3

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

Pruning

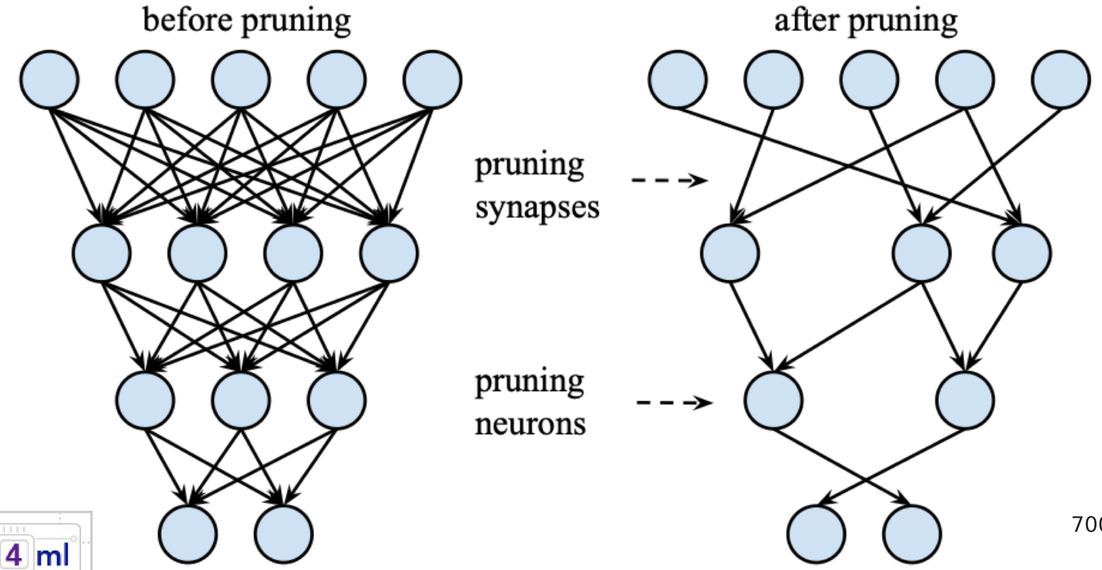


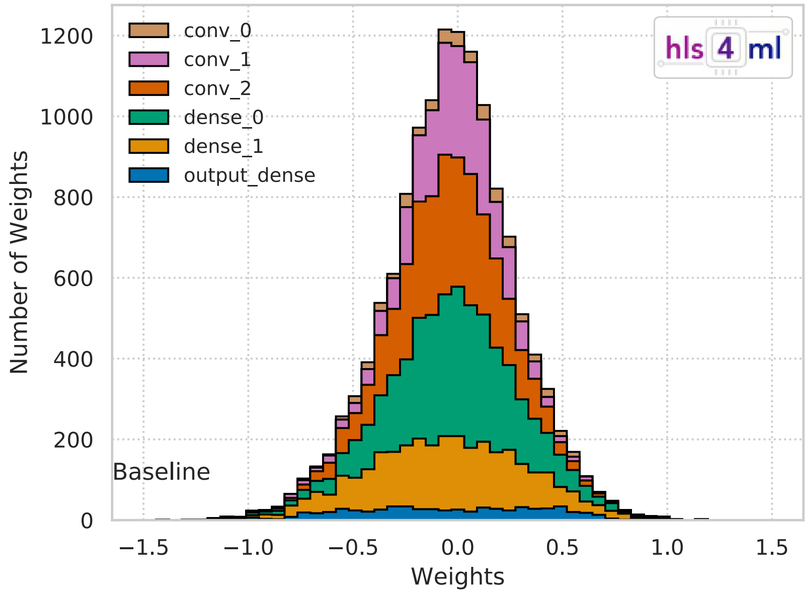


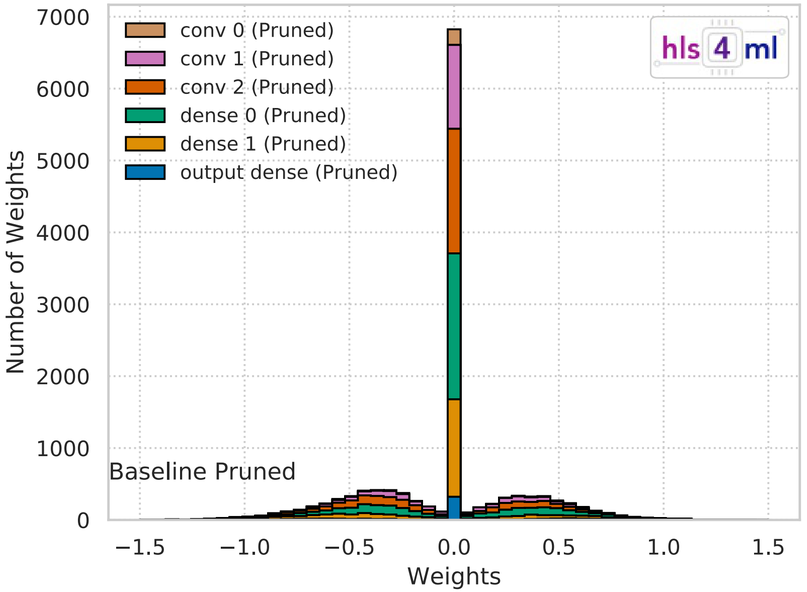
Exercise 3

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

Pruning



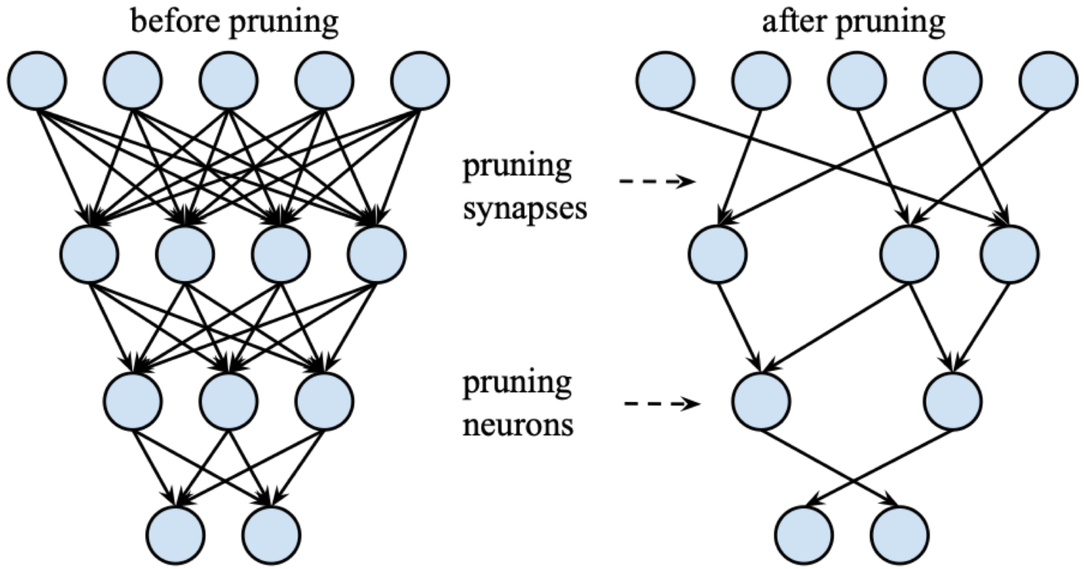


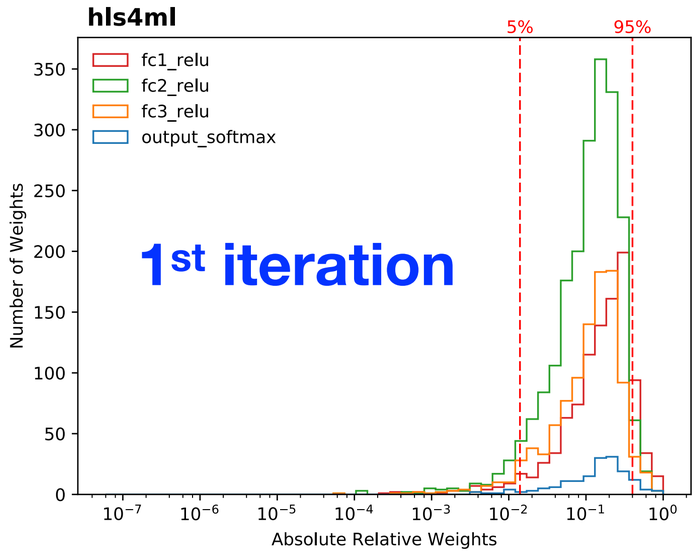


Exercise 3

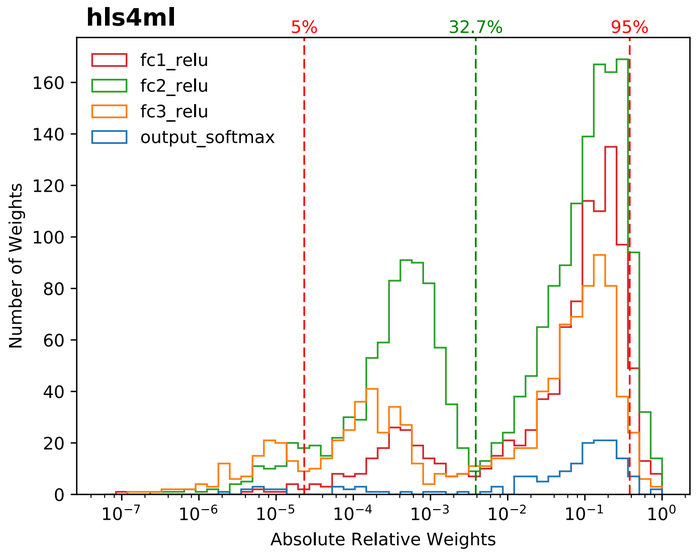
Pruning

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

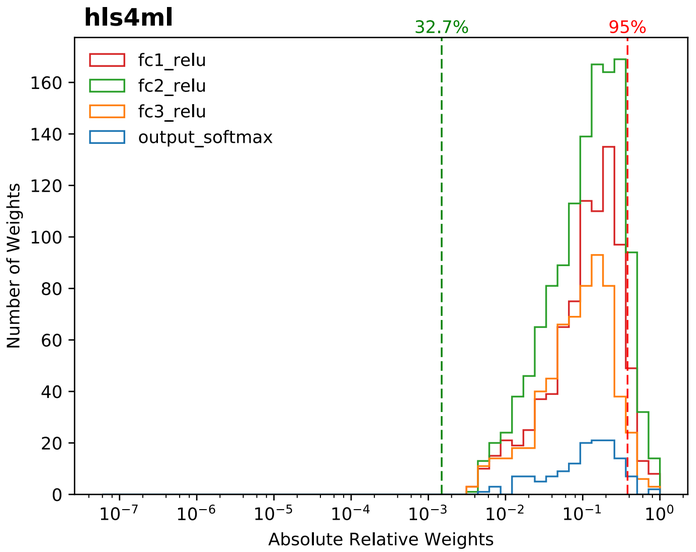




Train with L_1



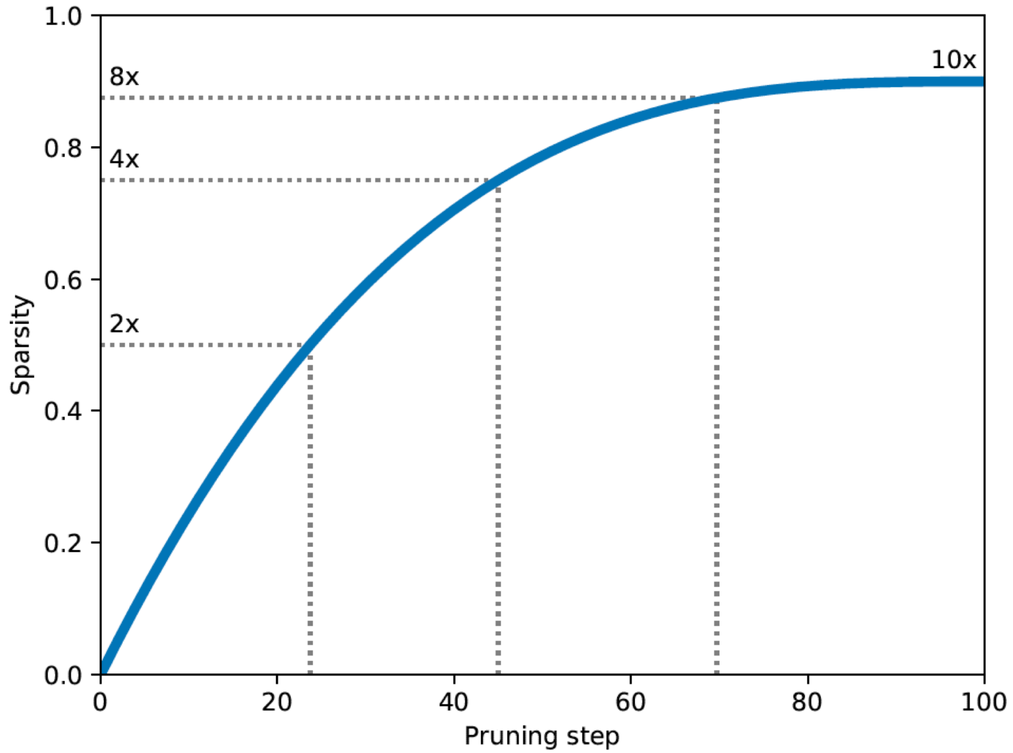
Prune

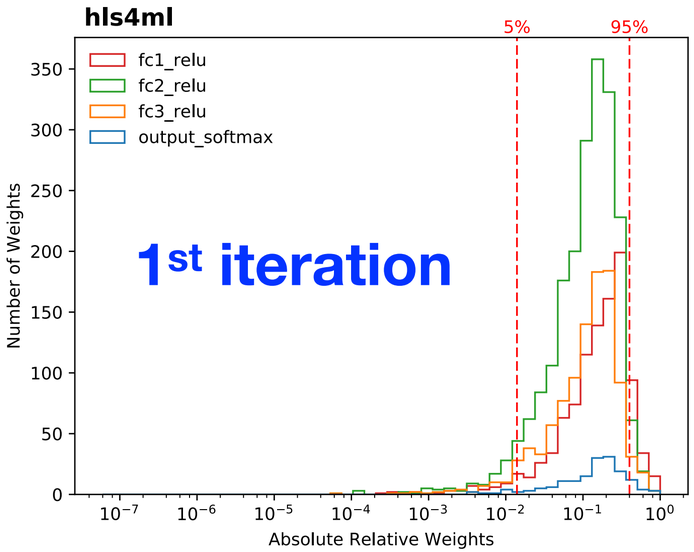


Exercise 3

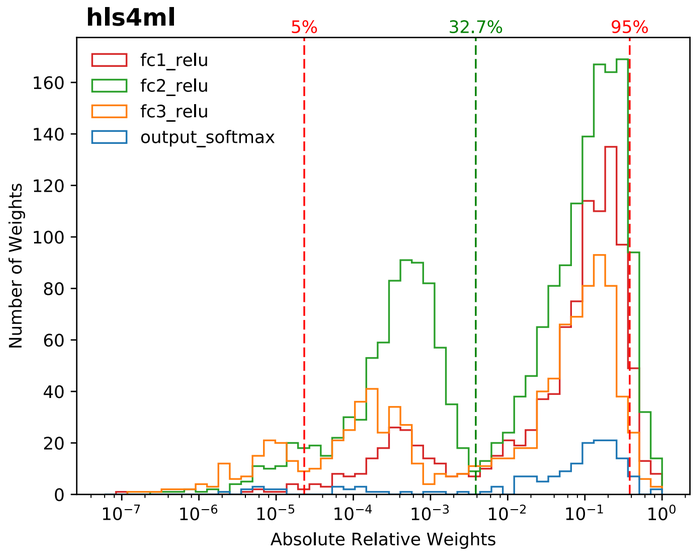
Pruning

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

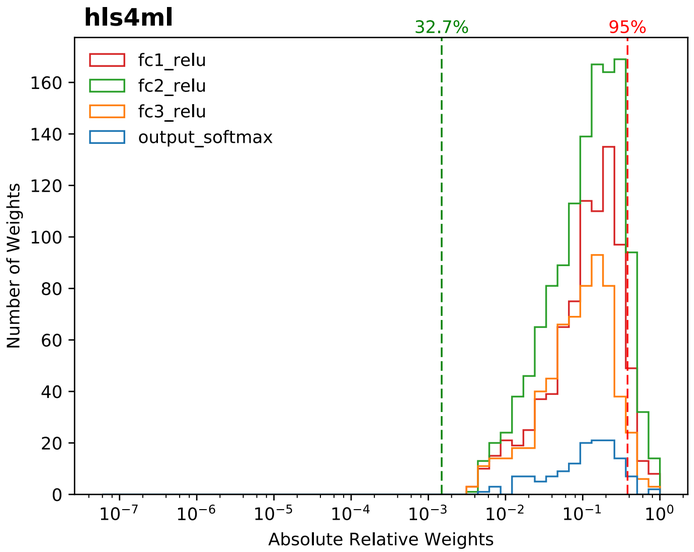




Train with L_1



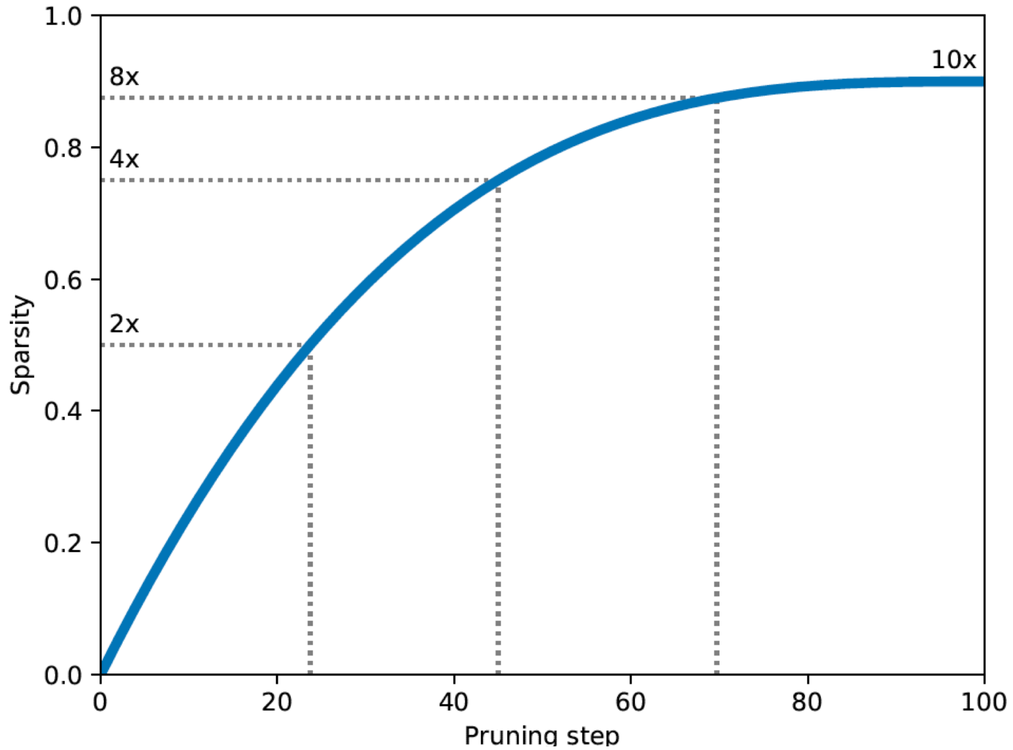
Prune



Exercise 3

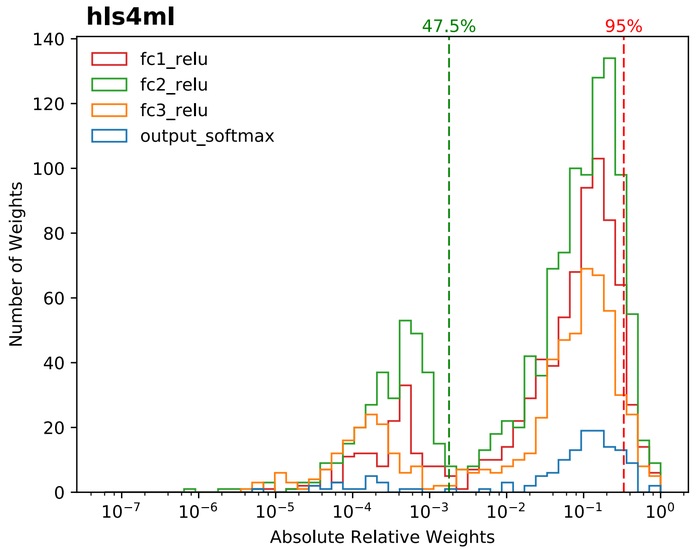
Pruning

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

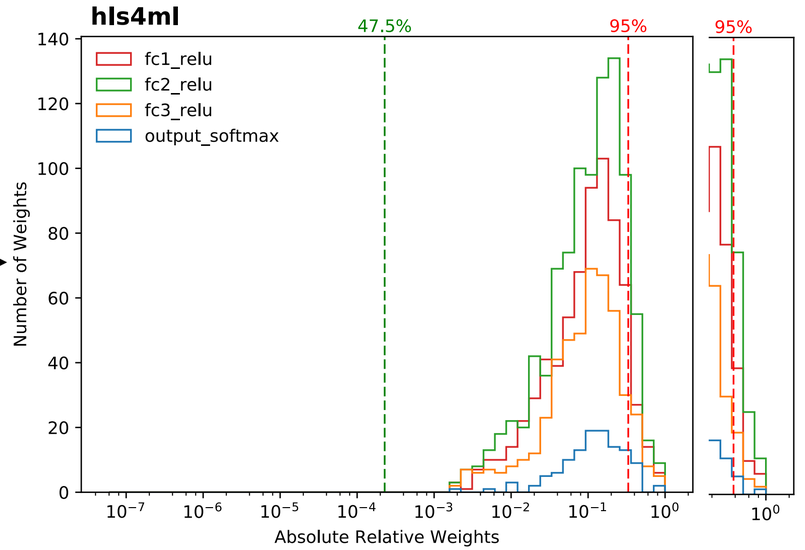


2nd iteration

Retrain with L_1



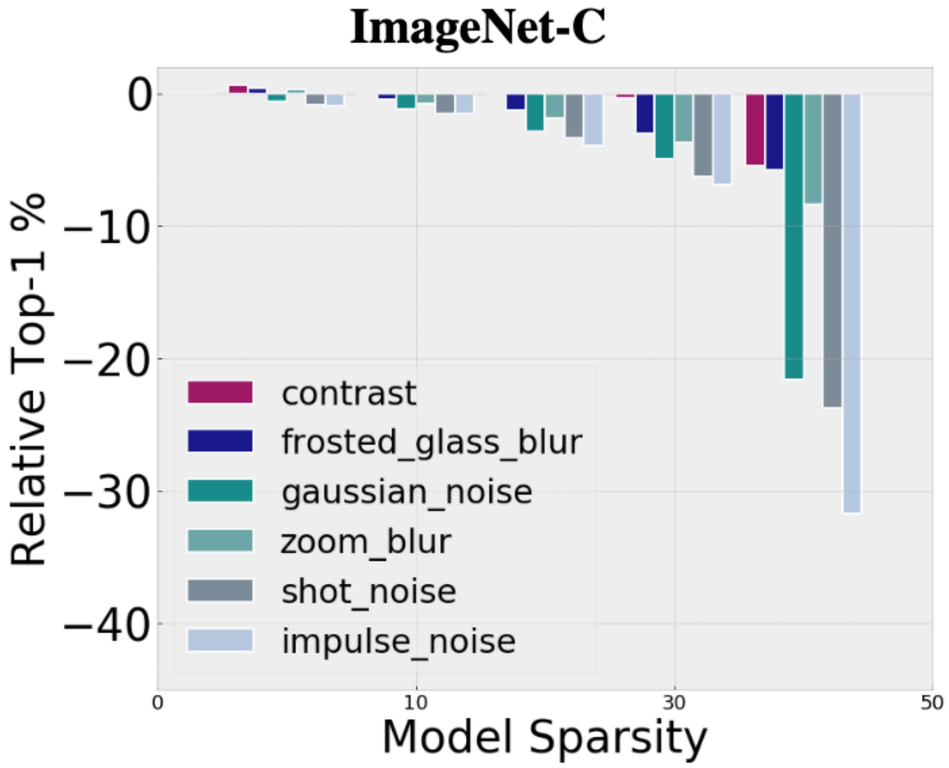
Prune

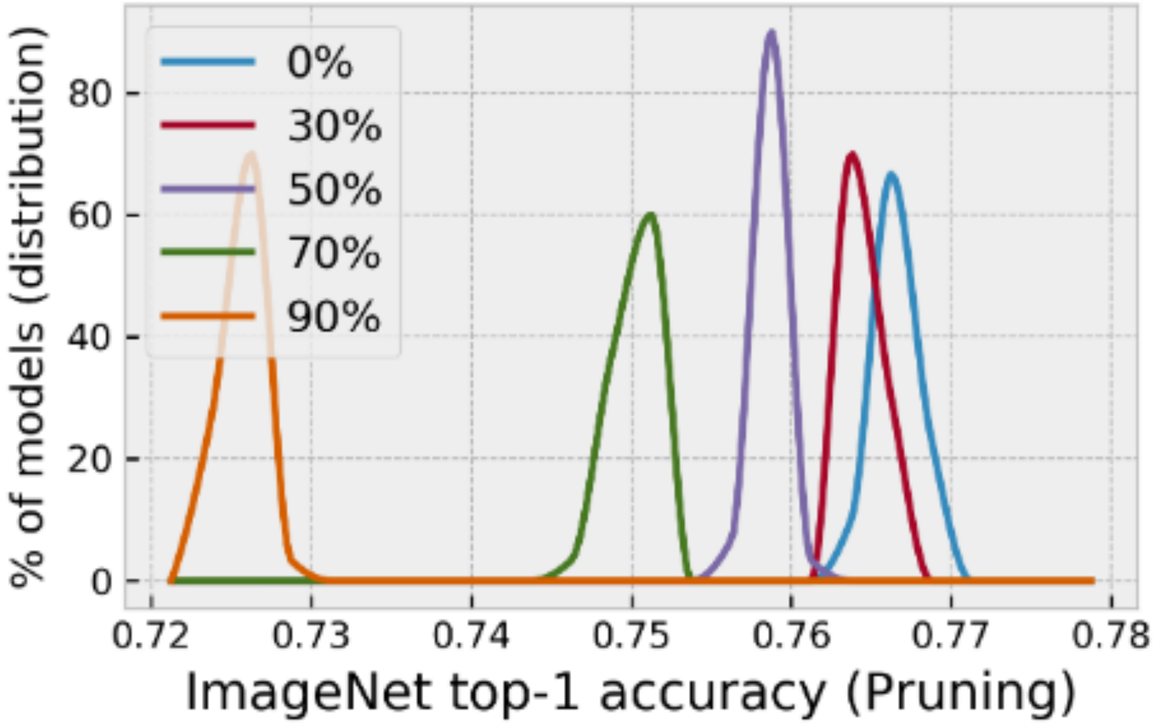


Hooker et al. (2021)

Pruning, robustness and accuracy

From Brian Bartoldson

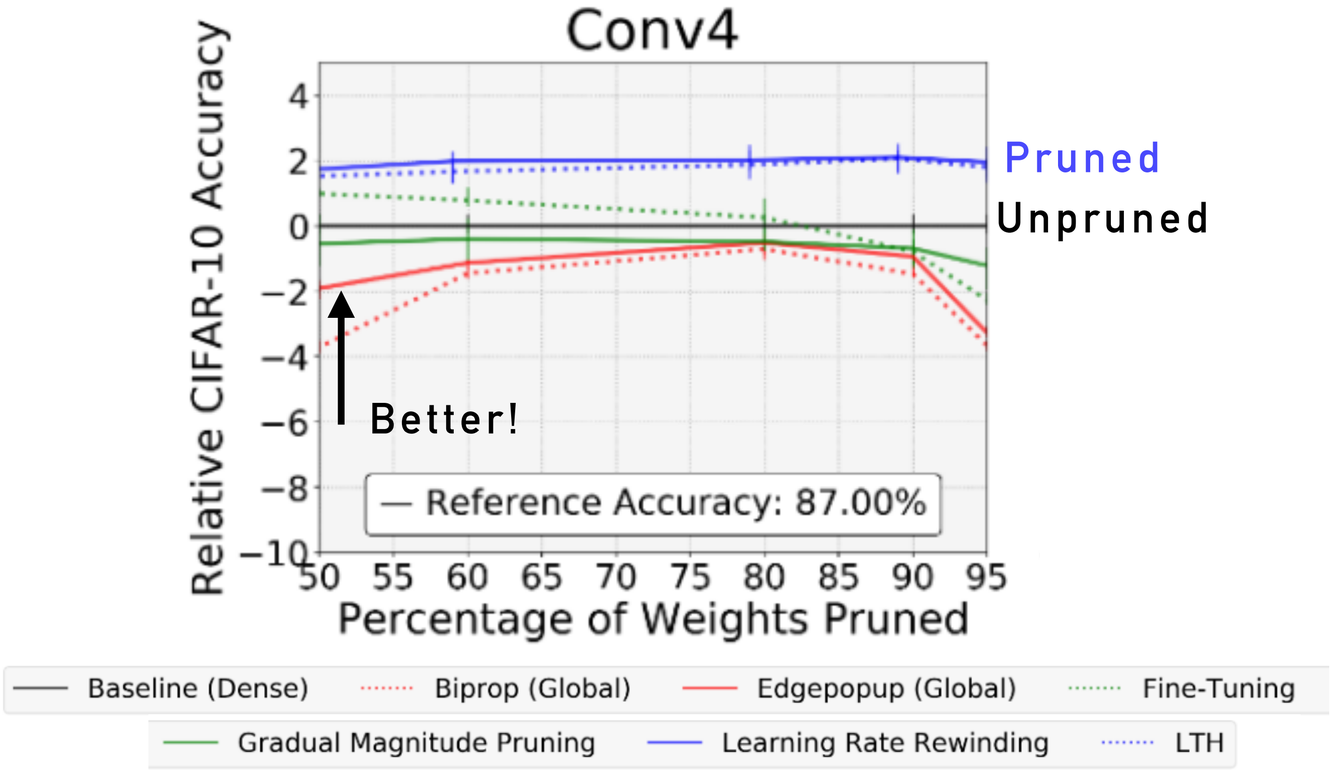




Exercise BONUS

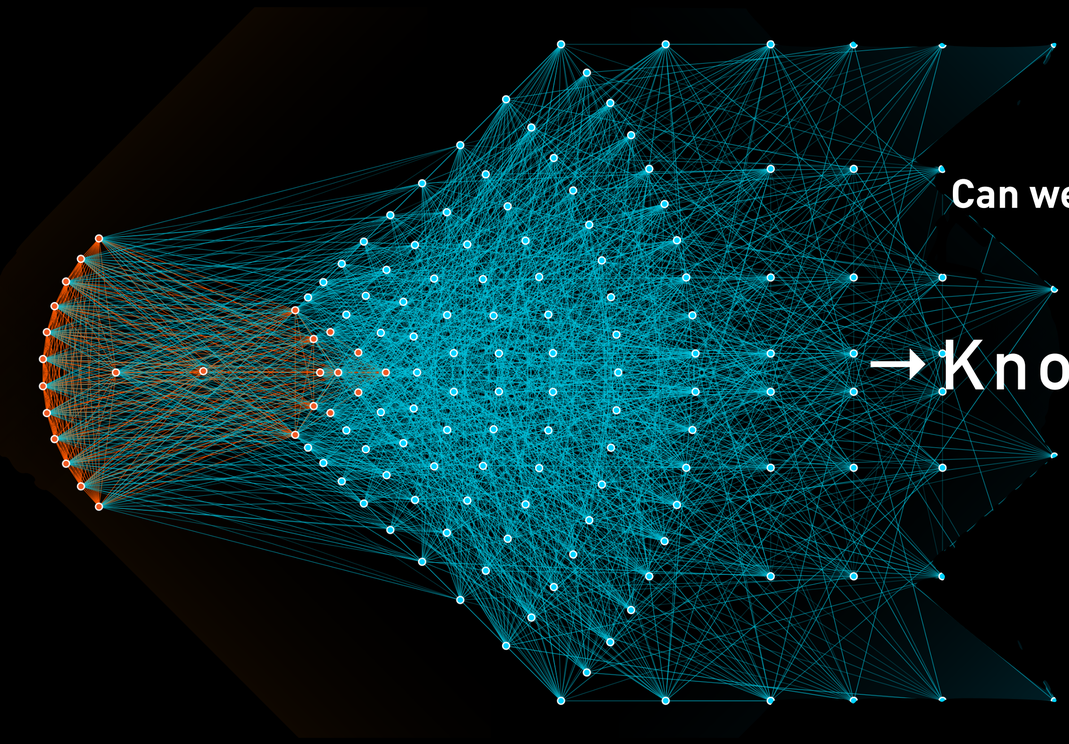
Pruning, robustness and accuracy

From Brian Bartoldson



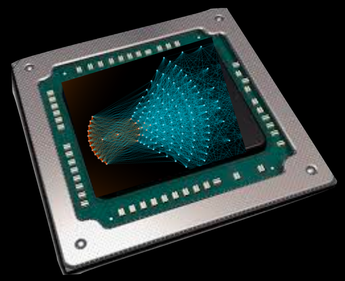
There exists a optimal network WITHIN each network (lottery ticket)
Uncover it through pruning!

Diffenderfer, Bartoldson, et al. (2021)



Can we have the best of both worlds?

→Knowledge Distillation

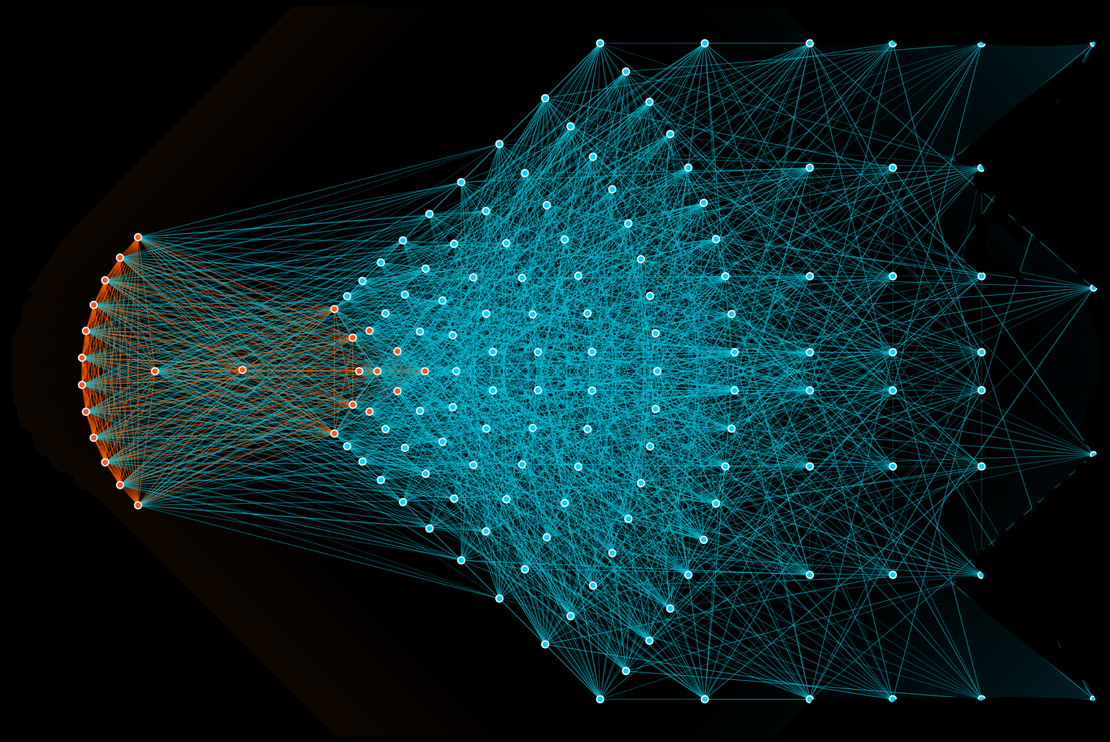


Train

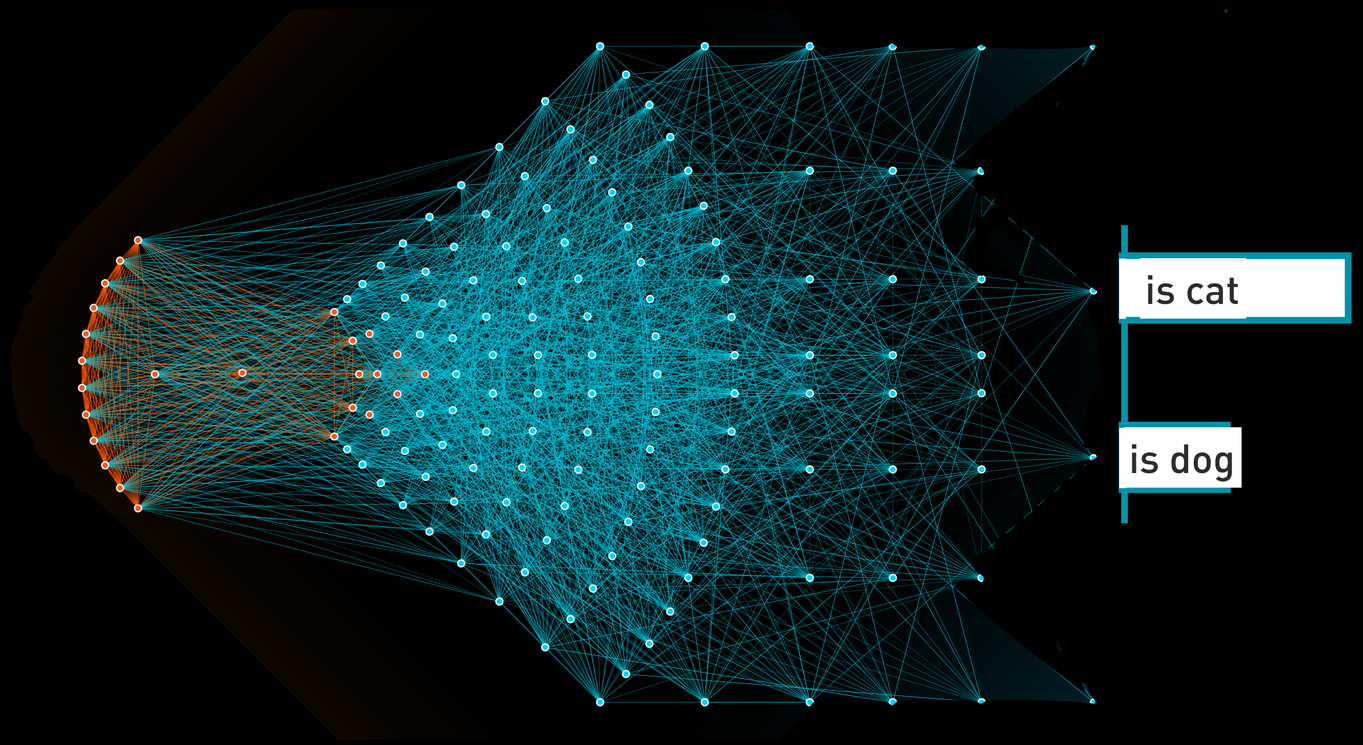
Inference

Cat

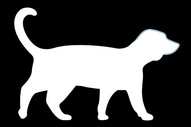
Dog

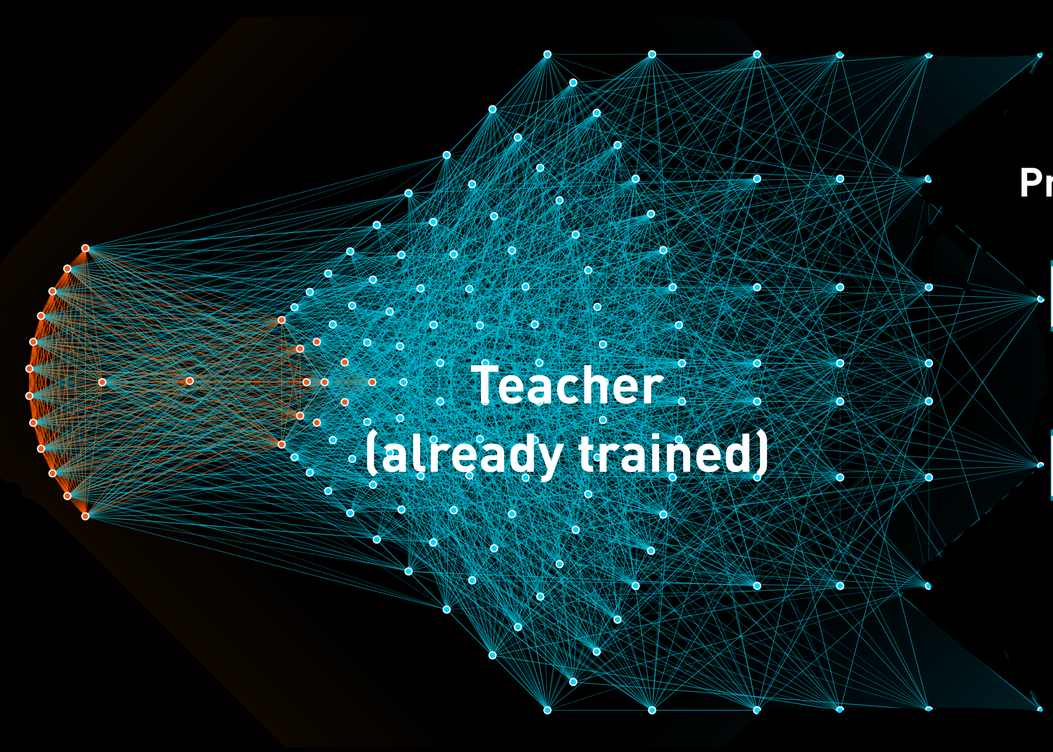


Cat



Cat



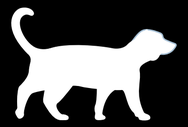


Predicted labels

is cat = 0.89

is dog = 0.11

Cat



Teacher
(already trained)

Predicted labels

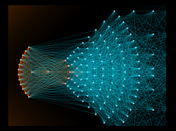
is cat = 0.89

is dog = 0.11

True labels

is cat = 1

is dog = 0

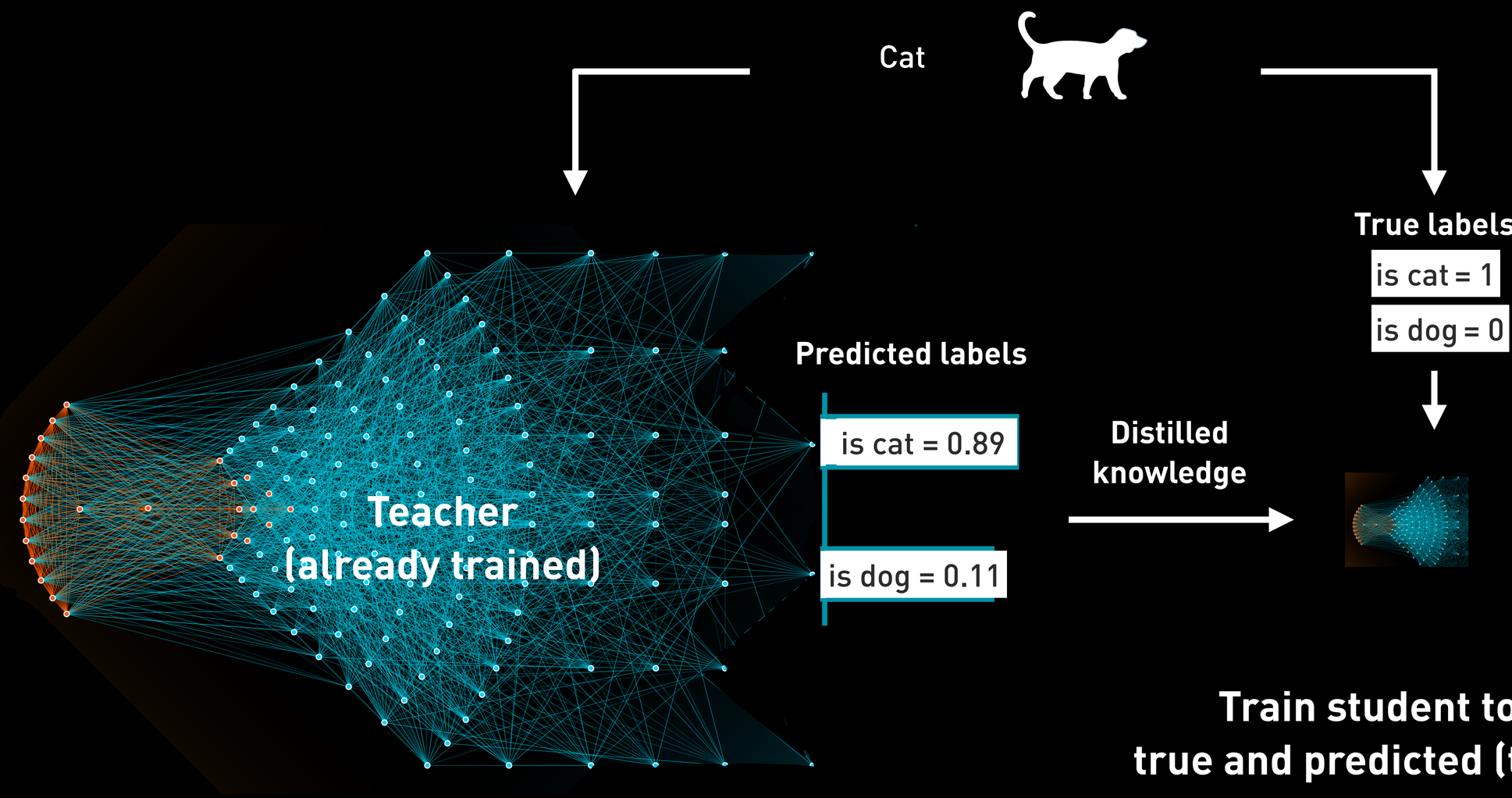


Predicted labels	True labels
is cat = 0.46	is cat = 0
is dog = 0.54	is dog = 1

Soft labels contain information!!

Predicted labels	True labels
is cat = 0.03	is cat = 0
is dog = 0.97	is dog = 1

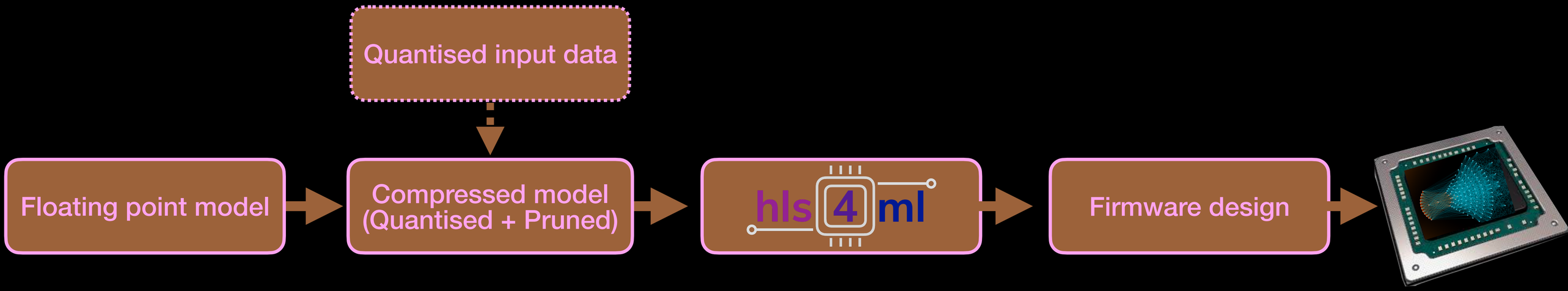
Exercise BONUS



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!



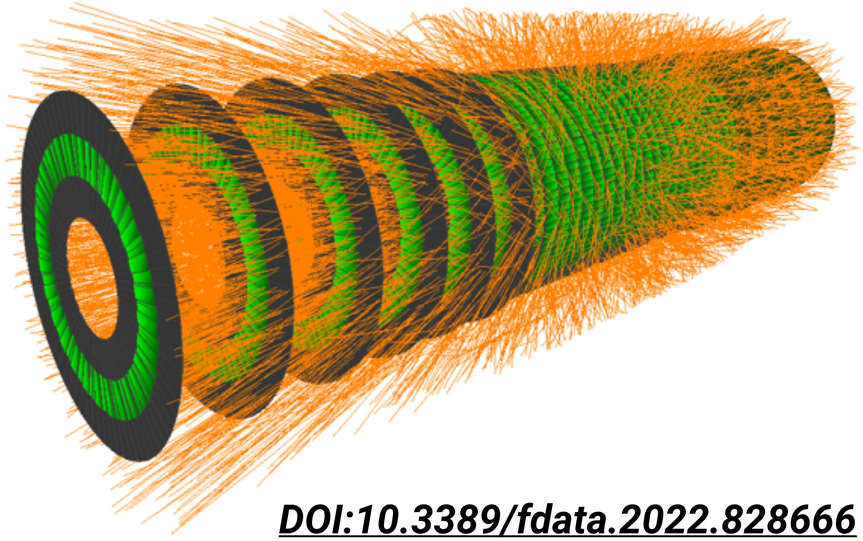
hls4ml tutorial

hls4ml Conifer

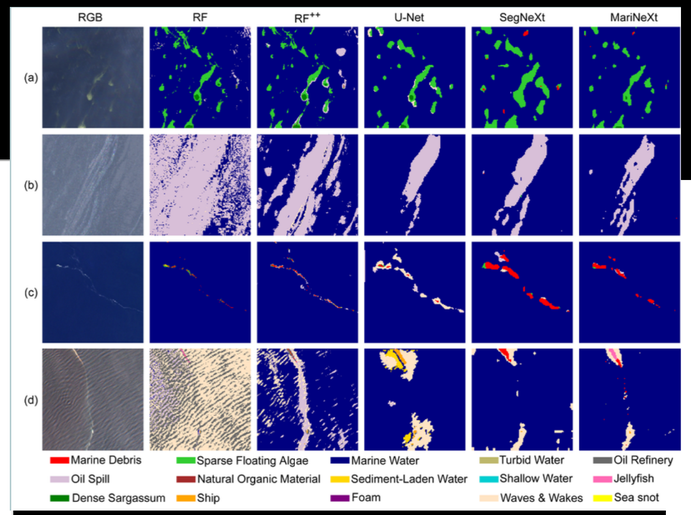
Well-established: MLPs, CNNs (classification, semantic segmentation), DeepSets, GNNs, RNNs, symbolic regression, (variational) autoencoders, BDTs, IsolationTrees

Experimental: Transformers, large distributed CNNs (ResNet, VGG)

300ns GNN for tracking

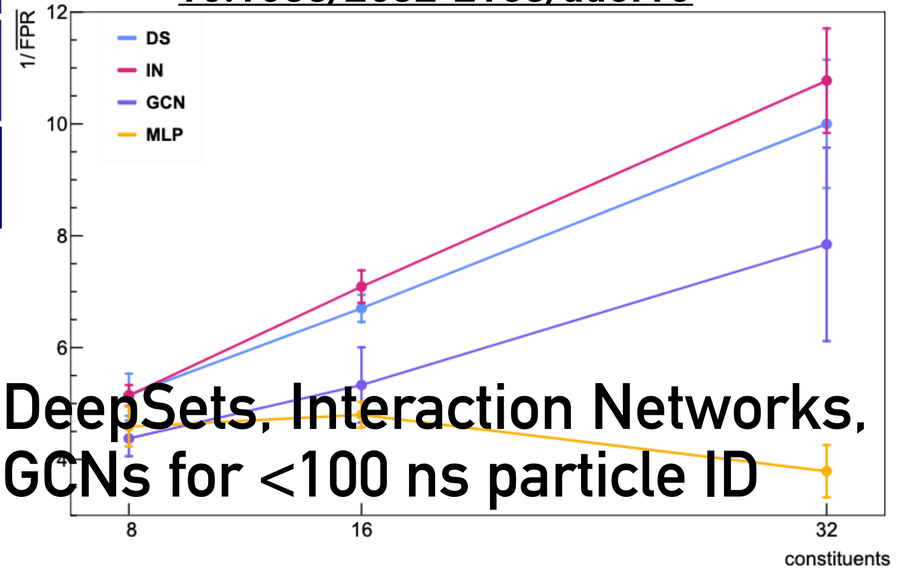


DOI:10.3389/fdata.2022.828666



UNet for semantic segmentation on satellites

10.1088/2632-2153/ad5f10



DeepSets, Interaction Networks, GCNs for <100 ns particle ID

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)

Reading exercise

1.1 Computing's Energy Problem (and what we can do about it)

Mark Horowitz

Departments of Electrical Engineering and Computer Science,
Stanford University, Stanford, CA

1. Introduction

Technology scaling has decreased the cost of computing to the point where it can be included in almost anything. As a result, we now live in a world surrounded by computing devices. They power our searches on Google, connect to our friends on Facebook, answer our questions to Siri, and serve us our entertainment on Youtube; they are in our homes everywhere, in all our appliances (I recently had to reboot my refrigerator), cars, workplaces, and even in the cards we send to each other. We have become so accustomed to computing becoming faster, cheaper, and lower power, we simply assume it will continue. Already, smartphone capabilities are being embedded in eye glasses [1] and smart watches [2].

While scaling computing performance has never been easy, a number of factors have made scaling increasingly difficult this past decade, and have caused power to become the principal constraint on performance. Section 2 quickly reviews how computing became power limited, even before Dennard constant-field scaling [3] broke down, and explains the difficulties of using a technology change to fix our problems. The rest of the paper explores different approaches to addressing this computing-energy-consumption challenge, and shows that it will take more than parallelism to get the results we need. The new key to scaling computing performance is to create applications and hardware which are better matched to the task and each other. Accomplishing this will require tools that allow application experts to create these new efficient systems. While creating these tools is challenging, they will enable a renaissance in application-optimized computing!

8. Conclusion

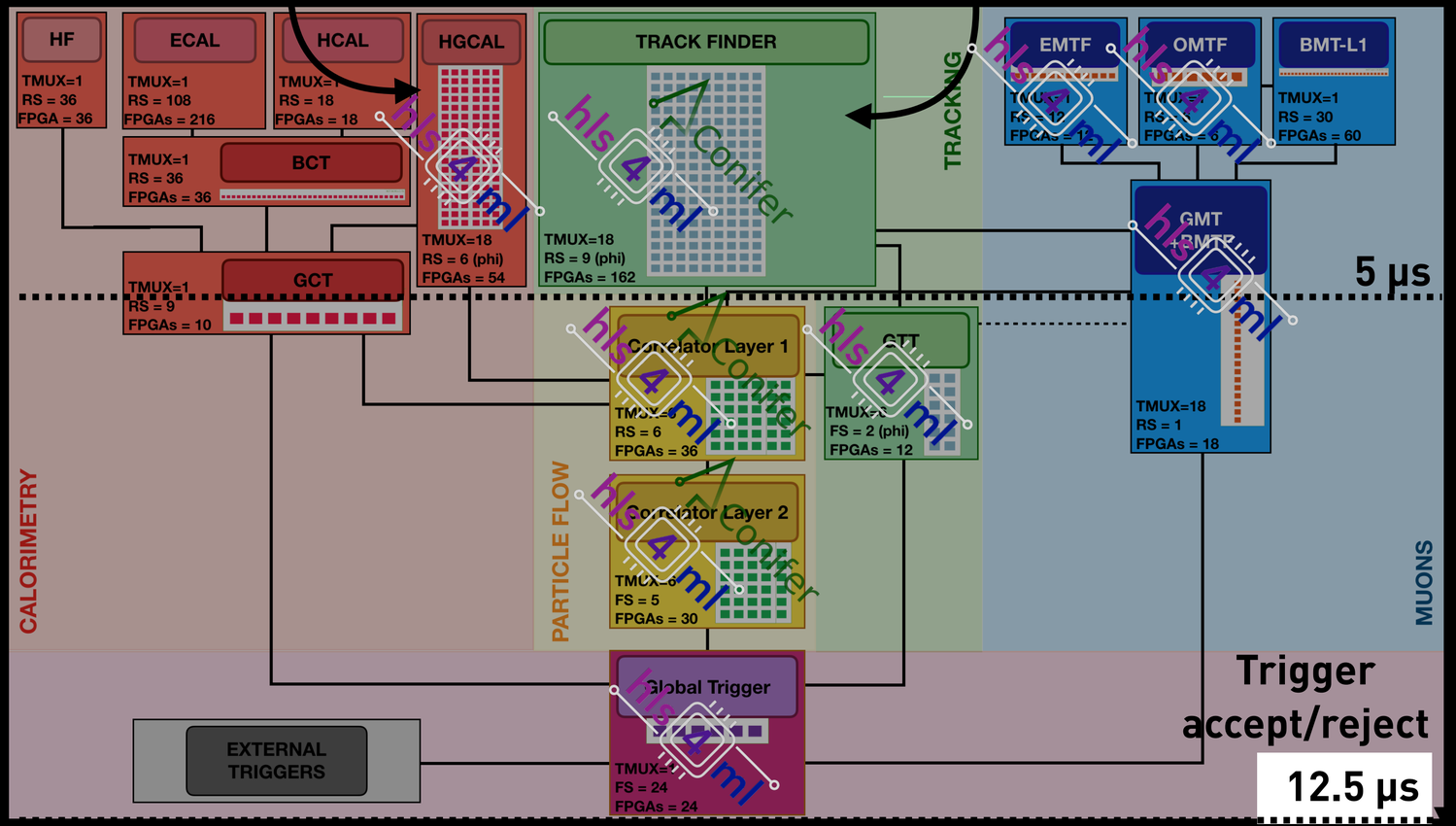
In summary, our challenge is clear: The drive for performance and the end of voltage scaling have made power, and not the number of transistors, the principal factor limiting further improvements in computing performance. Continuing to scale compute performance will require the creation and effective use of new specialized compute engines, and will require the participation of application experts to be successful. If we play our cards right, and develop the tools that allow our customers to become part of the design process, we will create a new wave of innovative and efficient computing devices.

https://ieeexplore.ieee.org/document/6757323

Nanosecond ML inference on FPGAs!

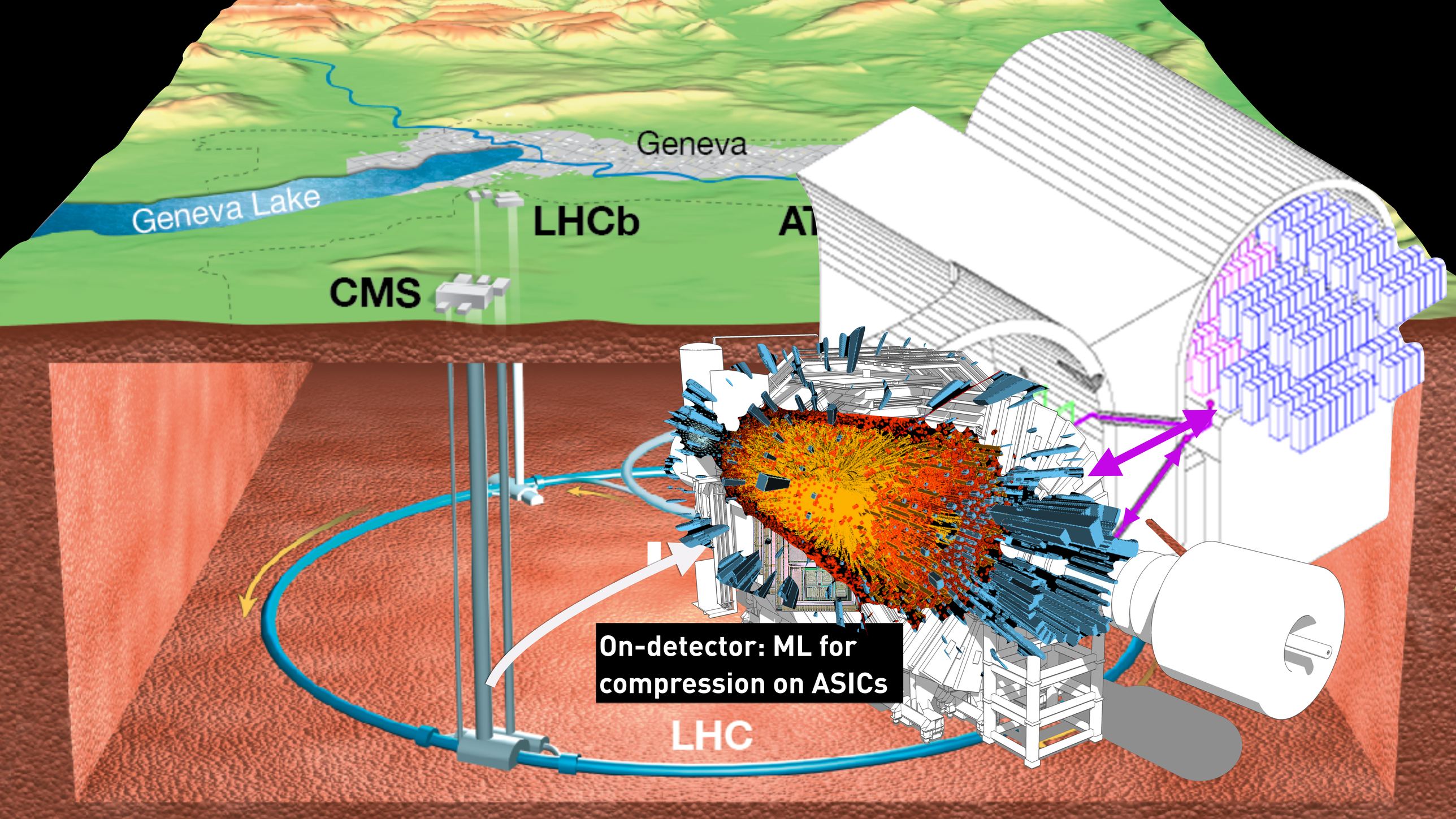
40 billion inferences/s during HL-LHC

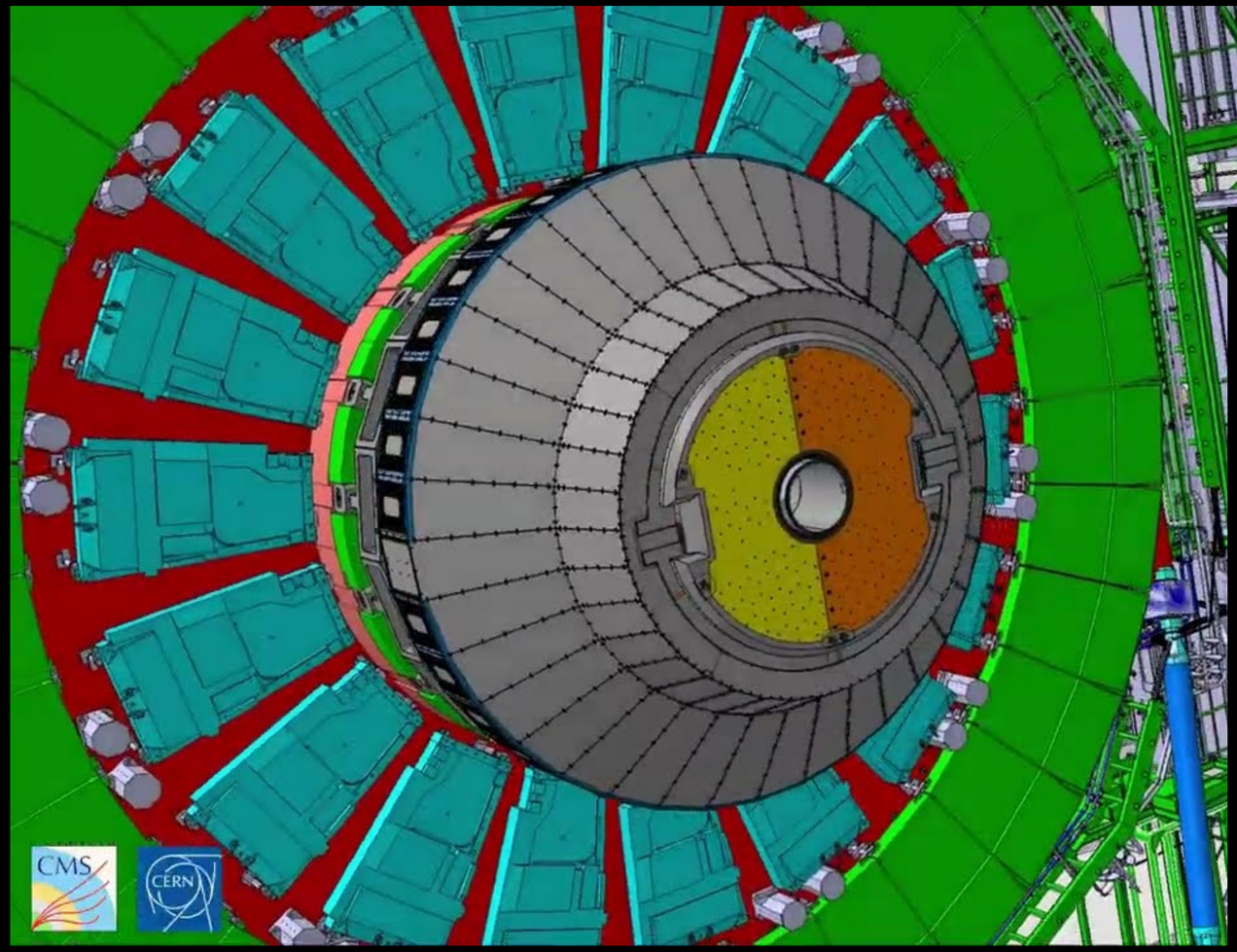
(≈ all inferences at Google)

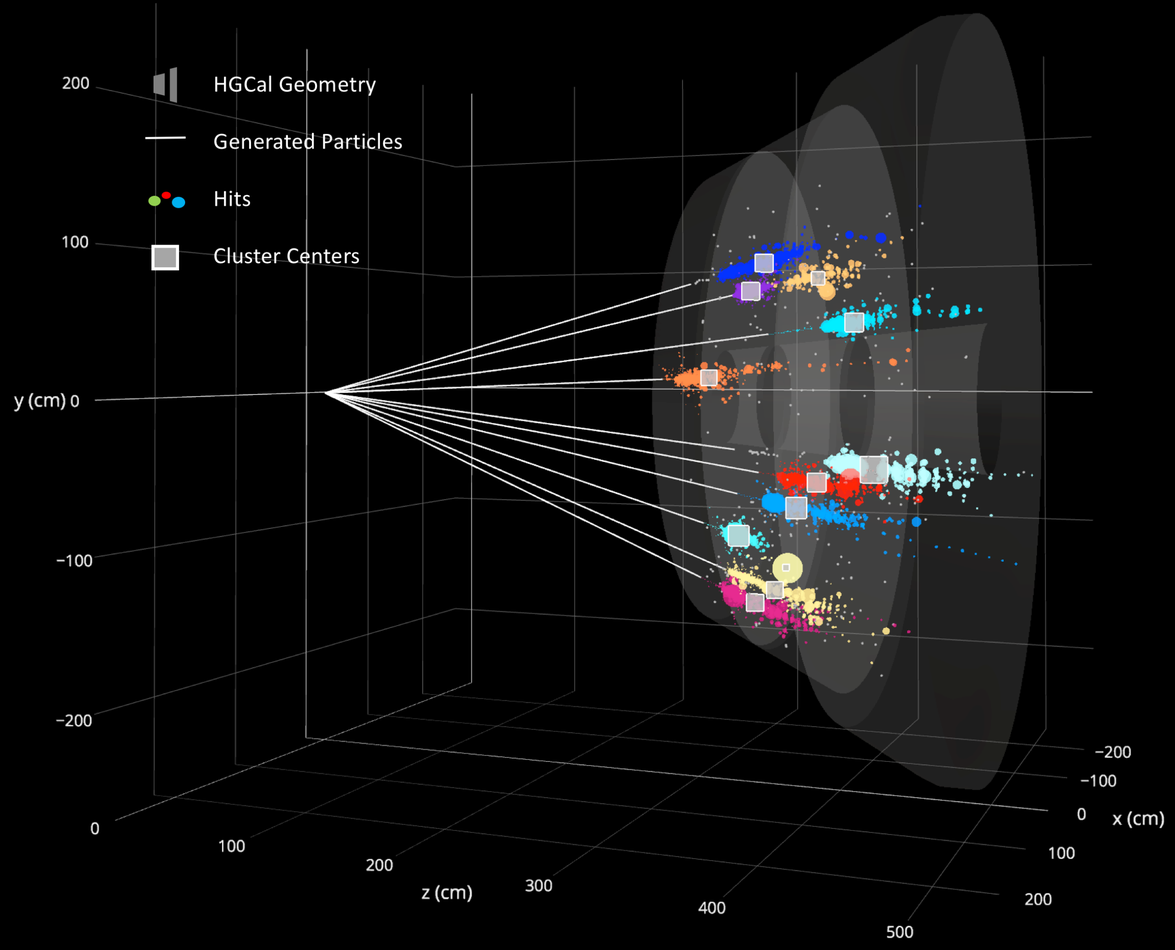


Conifer hls4ml

HEP developed libraries for fast ML on FPGAs

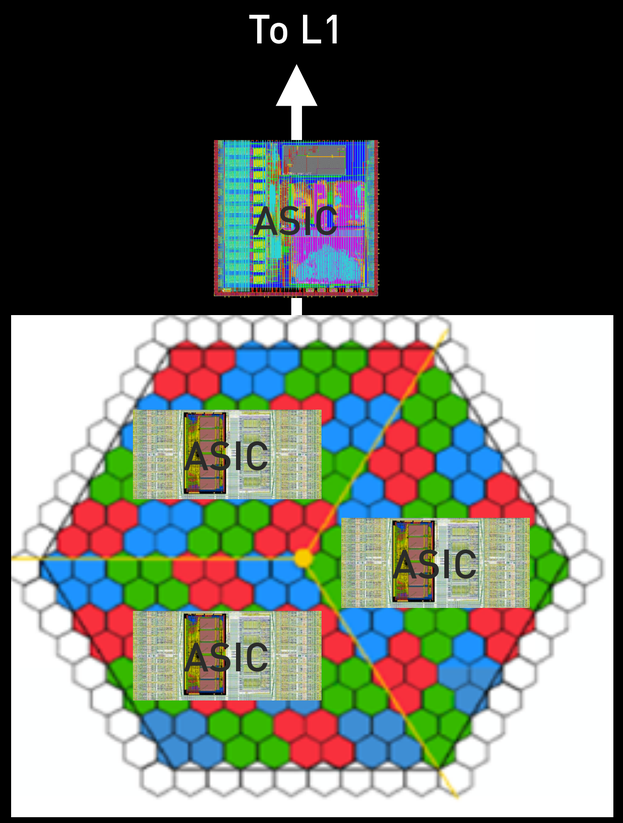

Geneva
Geneva Lake
LHCb
CMS
On-detector: ML for
compression on ASICs
LHC

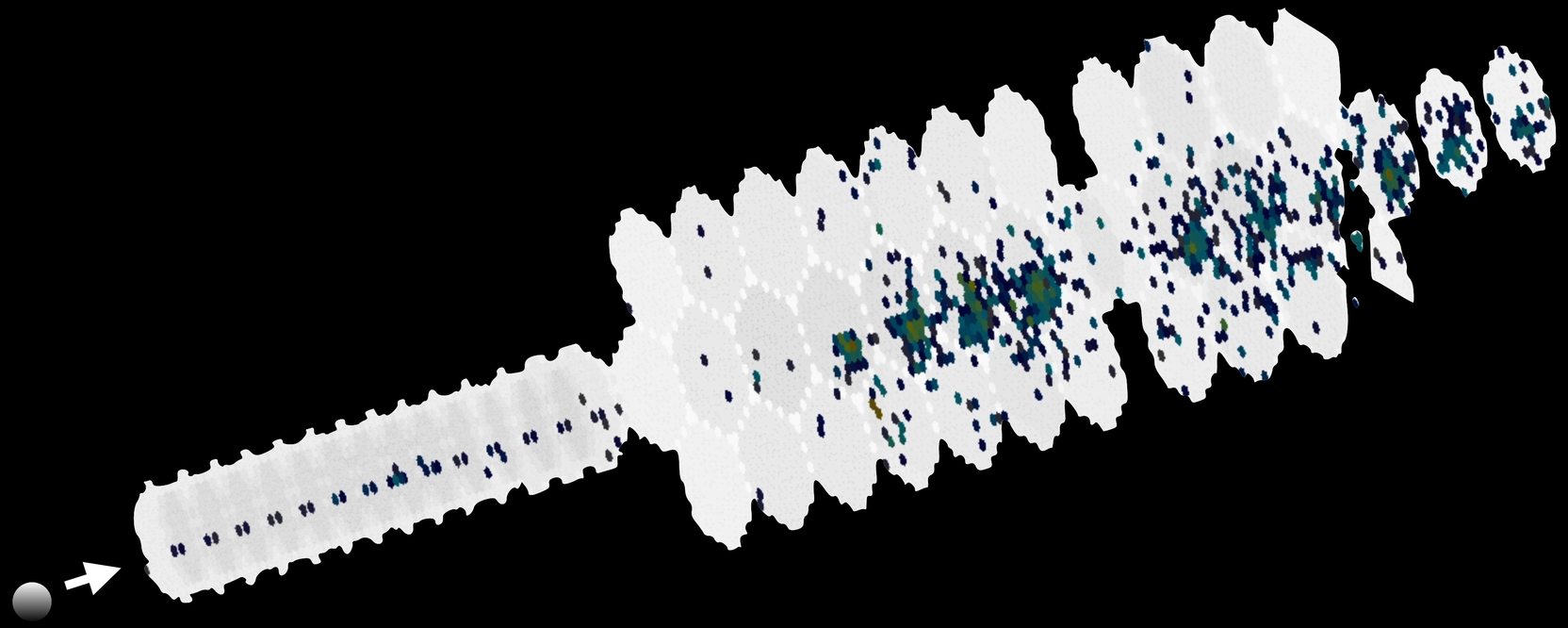




CMS High Granularity calorimeter

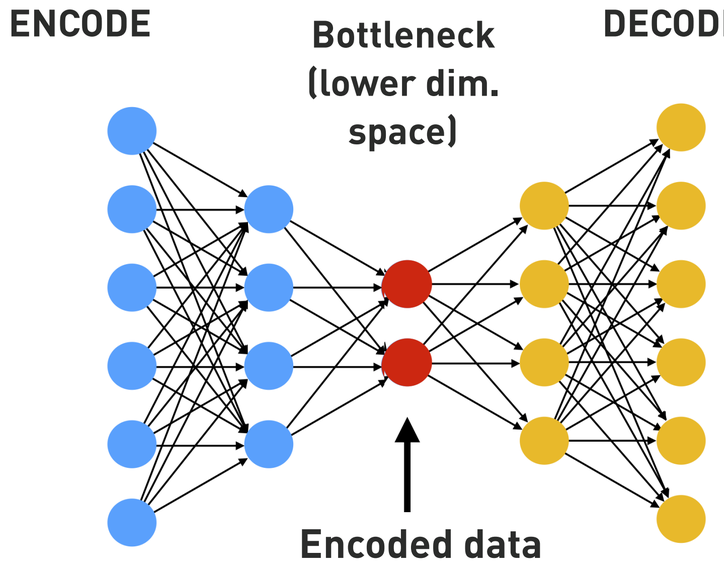
- 6.5 million readout channels, 50 layers



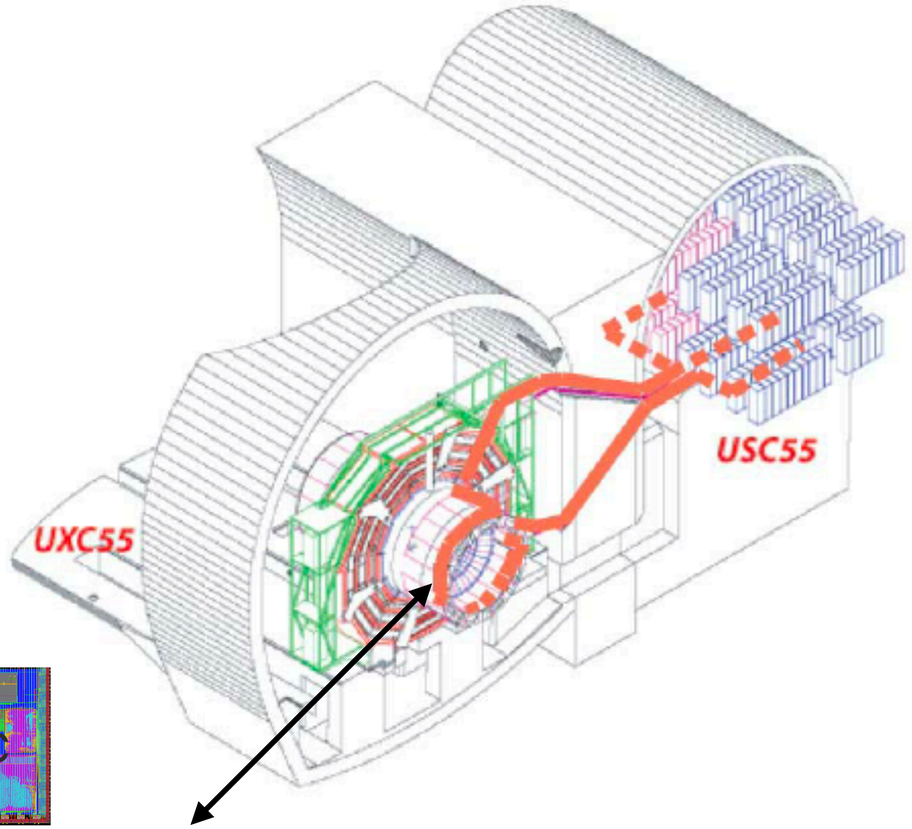


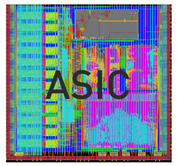
Must compress ON DETECTOR

- High radiation
- Cooled to -30 → low power
- 1.5 μs latency

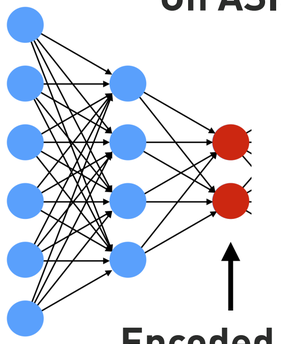


Variational Autoencoder

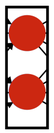




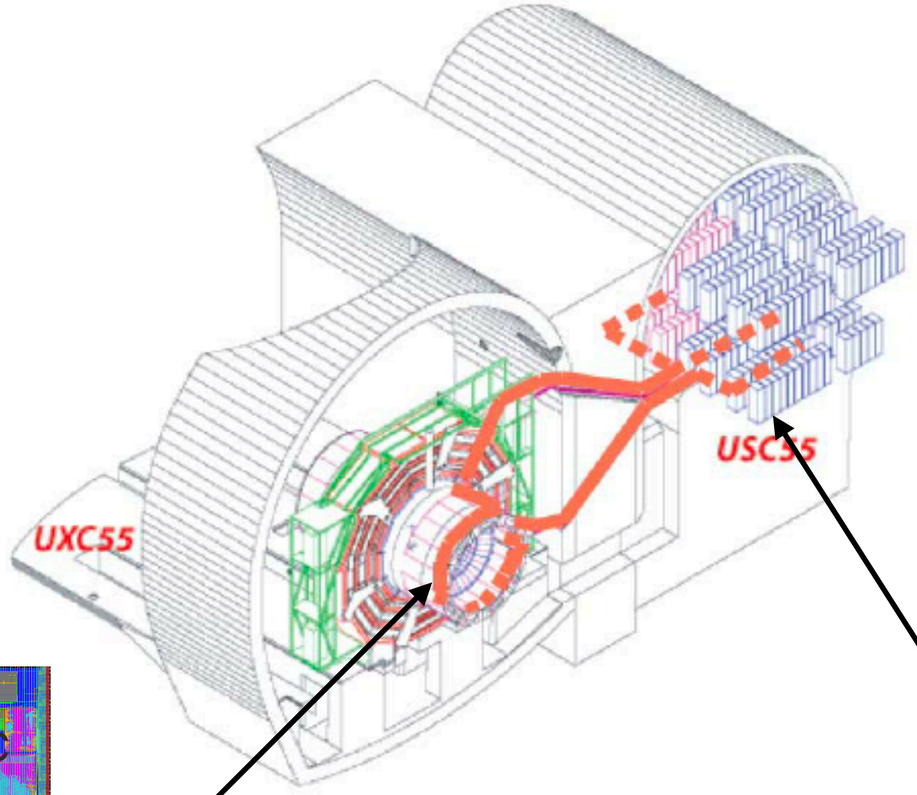
On ASIC

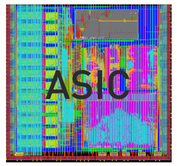


Encoded data

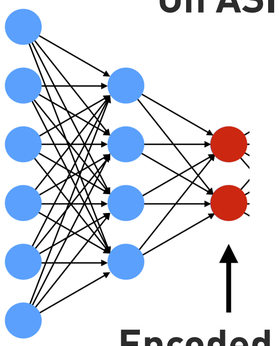


Transmit encoded data!

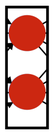




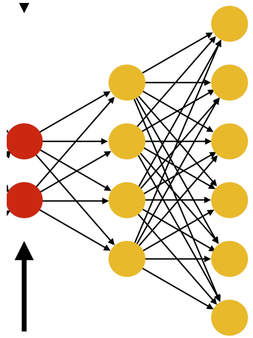
On ASIC



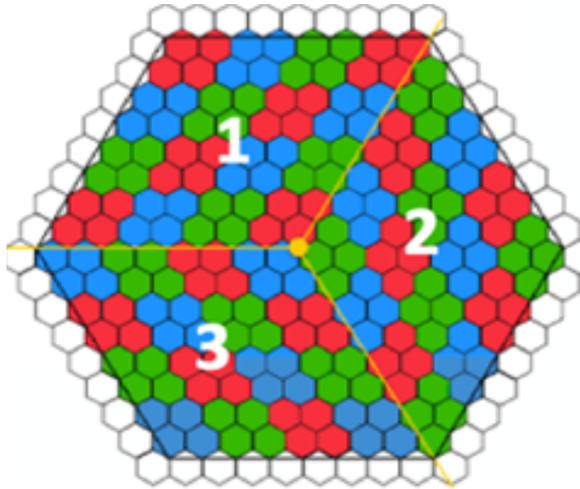
Encoded data



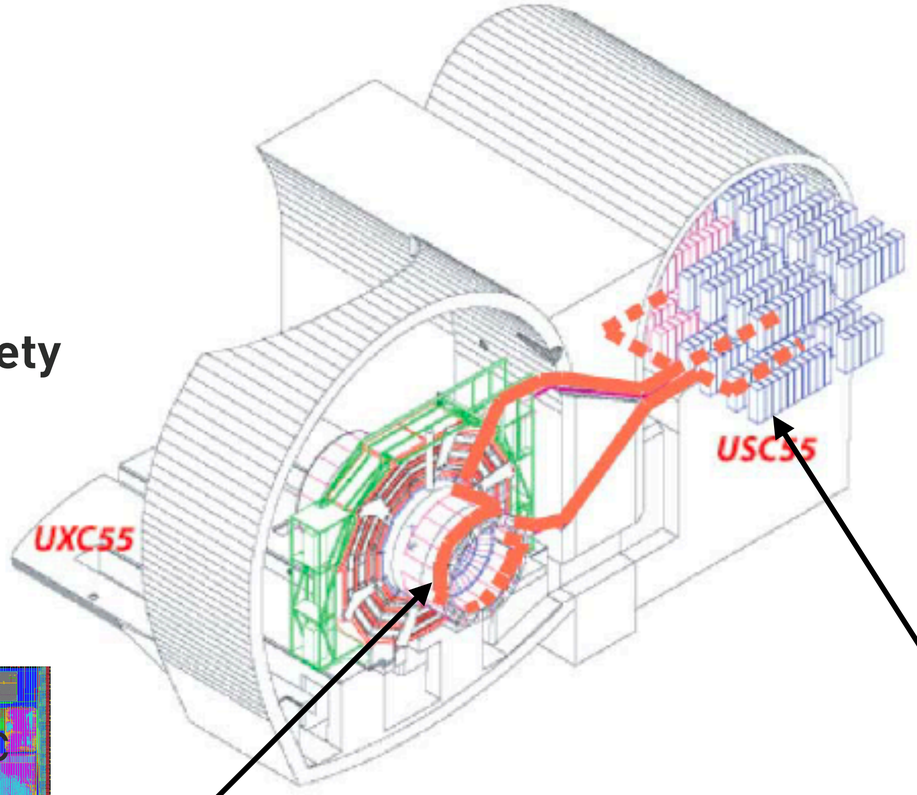
Transmit encoded data!

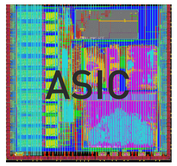


Encoded data

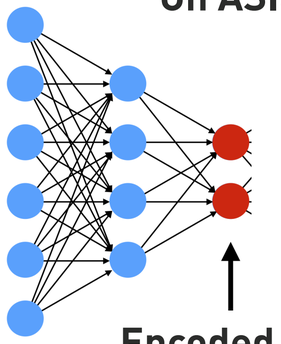


- **75-100 mW**
- **Triplicated w/b for radiation safety**
  Reprogrammable w/b over IC2!

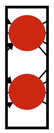




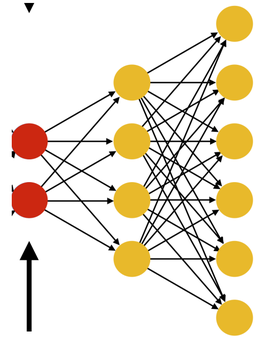
On ASIC



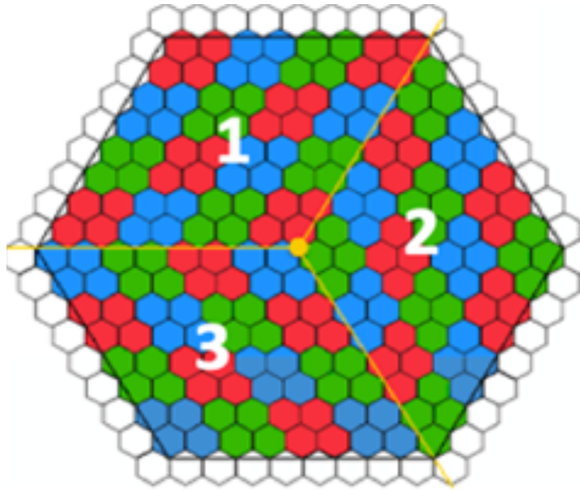
Encoded data



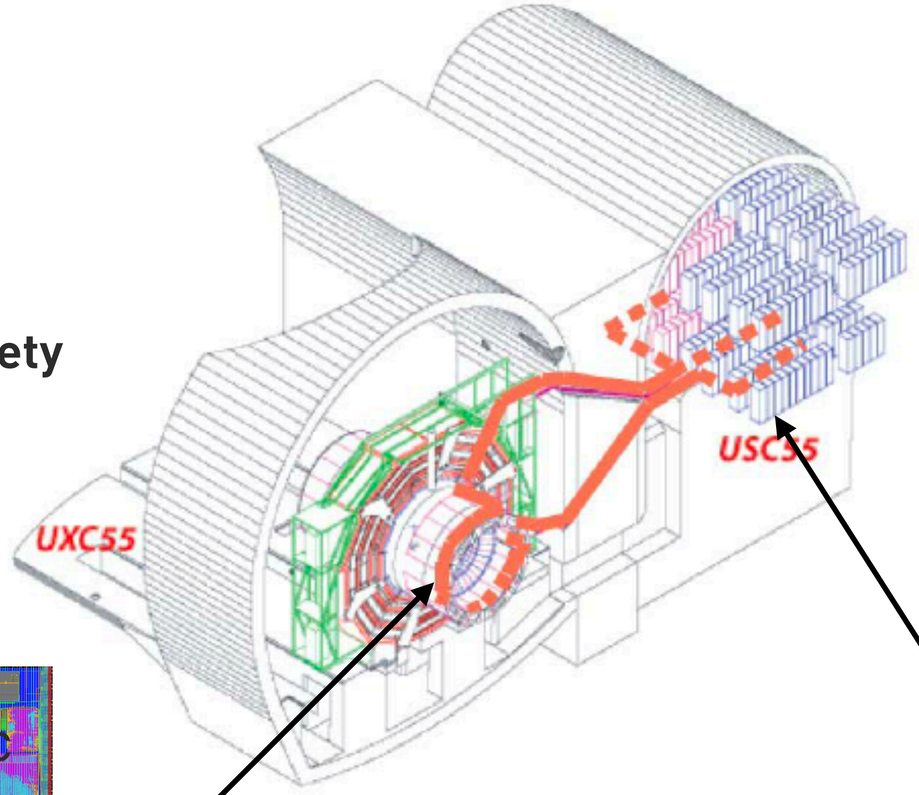
Transmit encoded data!

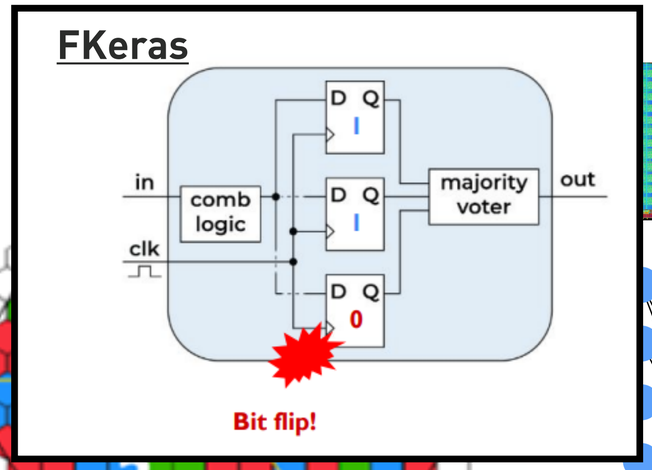


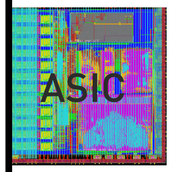
Encoded data

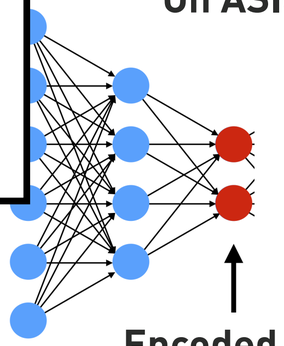


- **75-100 mW**
- **Triplicated w/b for radiation safety**
  Reprogrammable w/b over IC2!



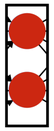




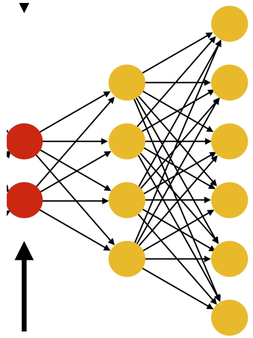


On ASIC

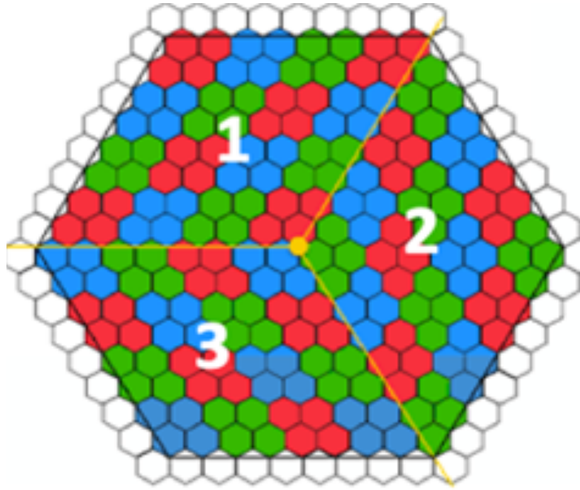
Encoded data



Transmit encoded data!

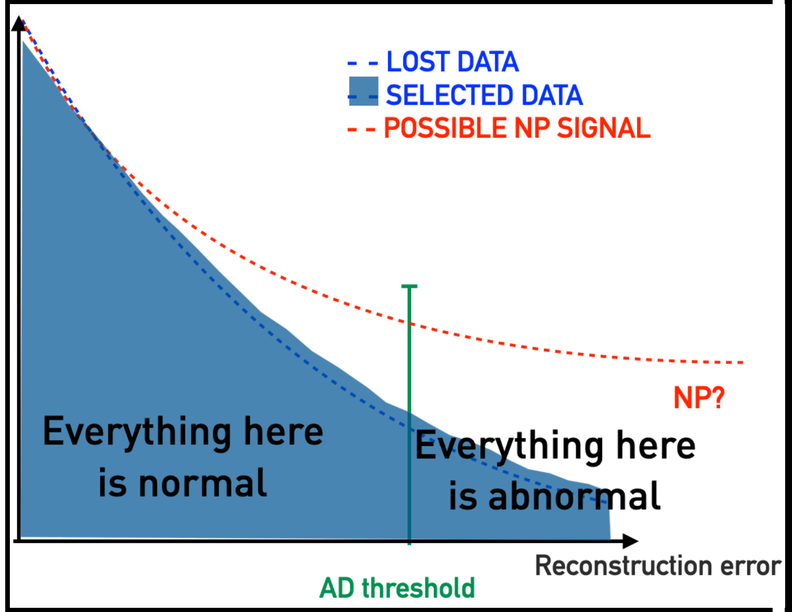


Encoded data



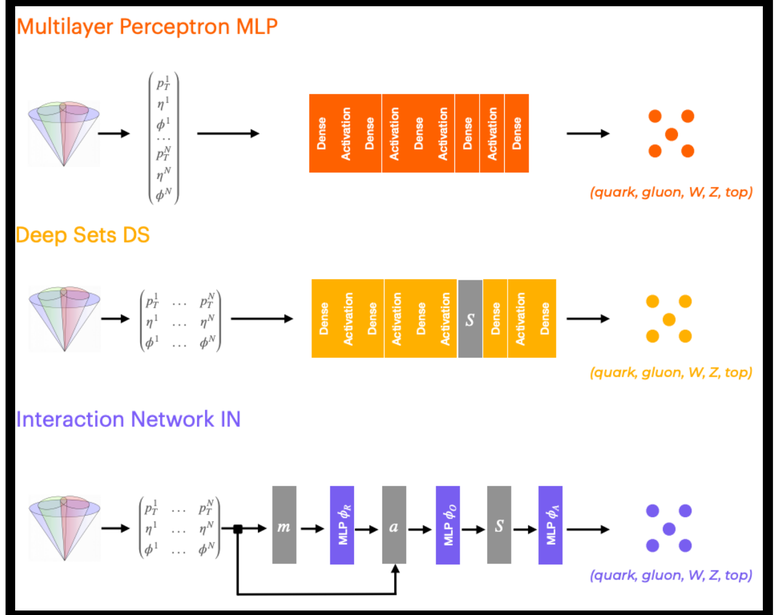
Final exercise

Anomaly detection with VAEs in 50 ns



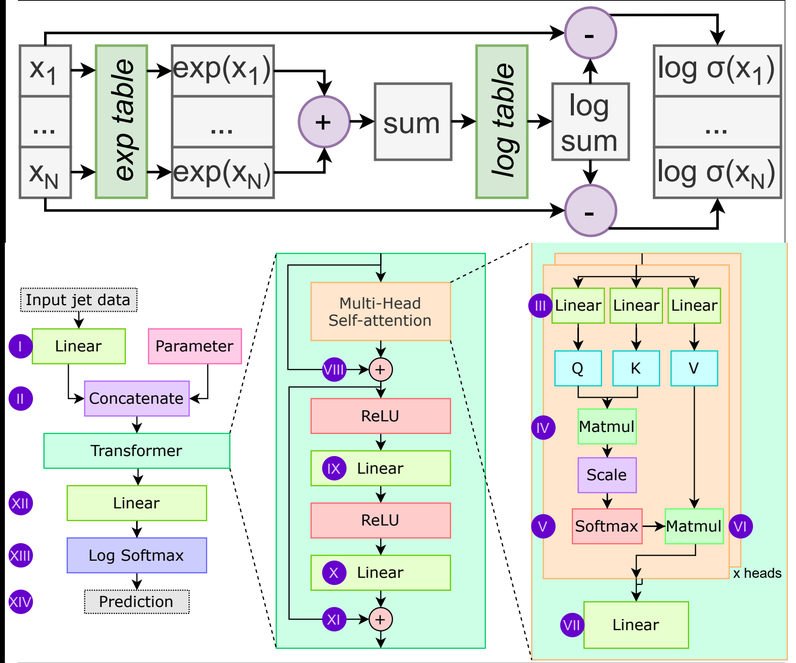
CMS DP2023_079
E. Govorkova et al (2022)

Quantised Interaction Networks and Deep Sets in <160 ns



P. Odagiu et al. 2024

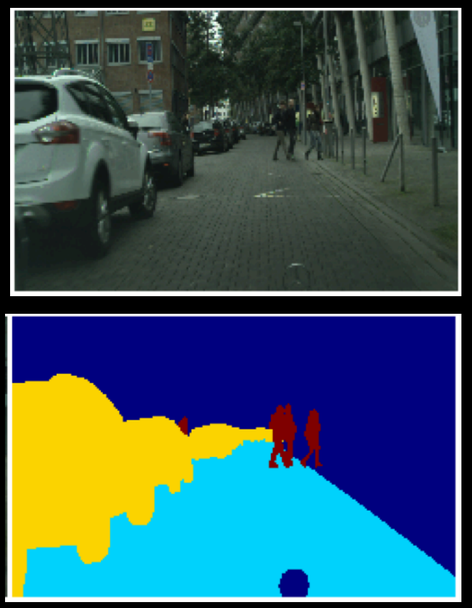
Fully on-chip transformers in 90 ns (500k flops, 9k param)



IEEE ICFTP 2022
arxiv:409.05207

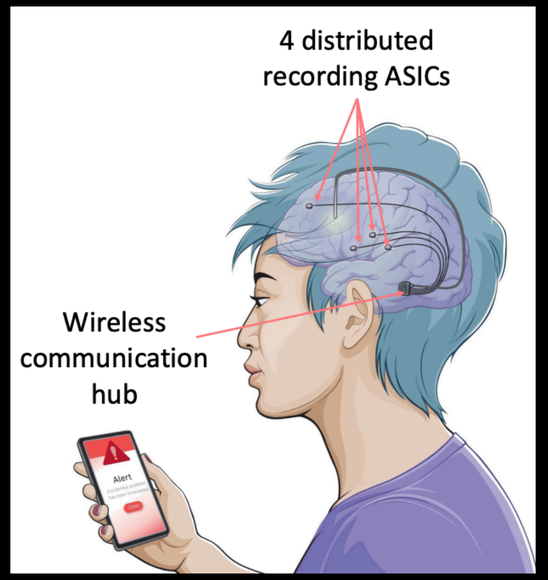
...and outside

Semantic segmentation for autonomous vehicles



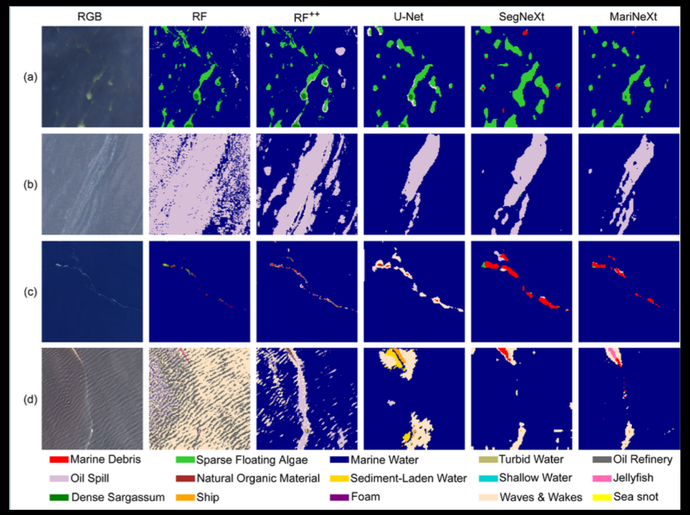
N. Ghielmetti et al. 2022

Seizure Predicting Brain Implant



W. Lemaire et al. 2022

Earth monitoring in satellites



Edge SpAIce, S. Summers

- MLPerf tinyML benchmarking
- For fusion science phase/mode monitoring
- Crystal structure detection
- Triggering in DUNE

- Accelerator control
- Magnet Quench Detection
- Food contamination detection
- Quantum control etc....

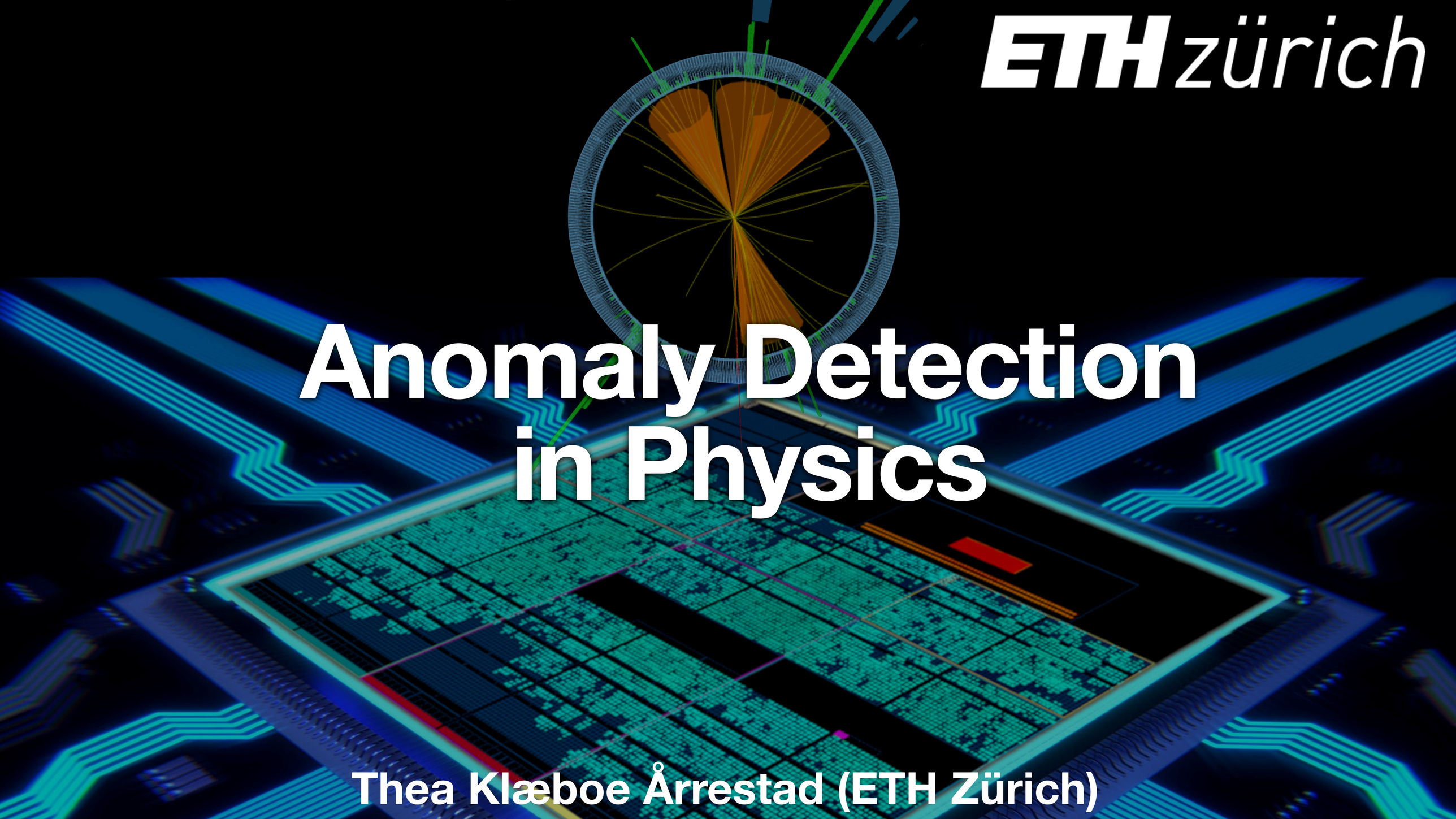
github.com/thaarres/ML_nuggets

github.com/fastmachinelearning/hls4ml-tutorial

Jargon

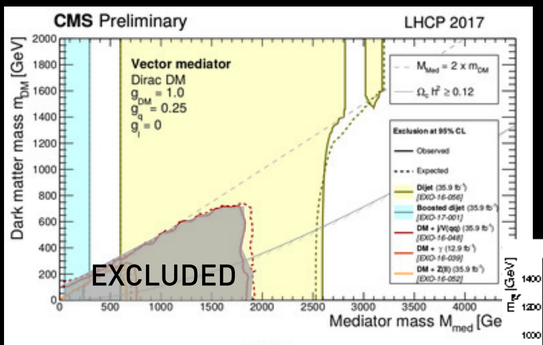
Great slide from Benjamin:

- **LUT** - Look Up Table aka 'logic' - generic boolean functions on small bitwidth inputs. Combine many to build the algorithm
- **FF** - Flip Flops - control the flow of data with the clock pulse. Used to build the pipeline and achieve high throughput
- **DSP** - Digital Signal Processor - performs multiplication and other arithmetic in the FPGA
- **BRAM** - Block RAM - hardened RAM resource. More efficient memories than using LUTs for more than a few elements
- **HLS** - High Level Synthesis - compiler for C, C++, SystemC into FPGA IP cores
- **HDL** - Hardware Description Language - low level language, such as Verilog or VHDL for describing circuits
- **Latency** - time between starting processing and receiving the result
- **II** - Initiation Interval - time from accepting first input to accepting next input (visualize, cars on a production line)

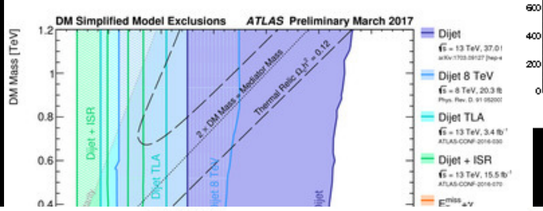


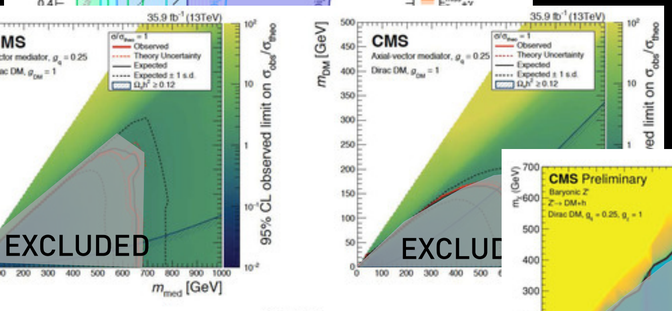
Anomaly Detection in Physics

Thea Klæboe Årrestad (ETH Zürich)

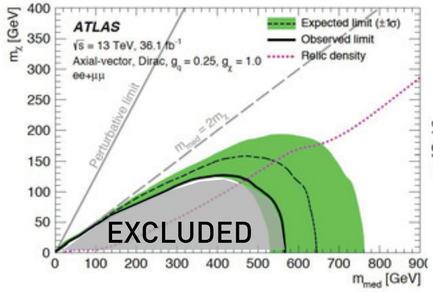


(a) CMS

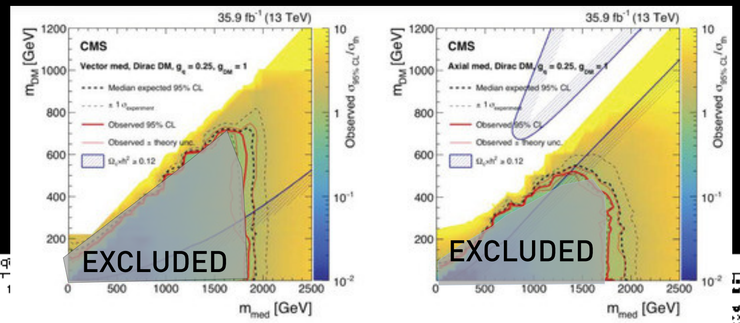




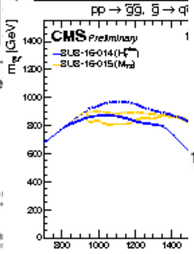
(a) CMS

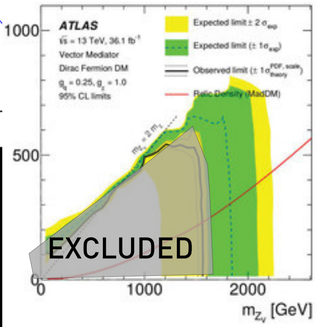


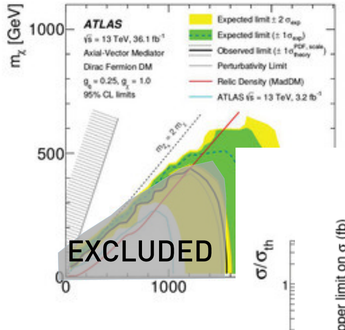
(b) ATLAS



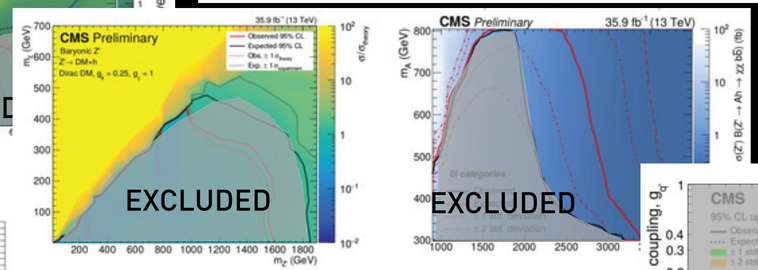
(a) CMS



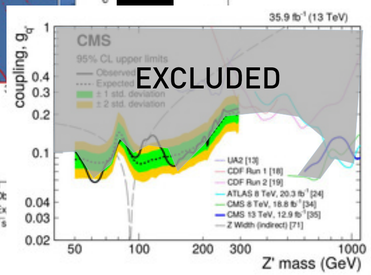


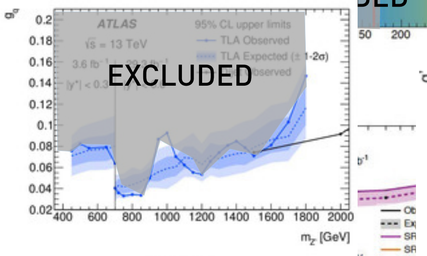


(b) ATLAS

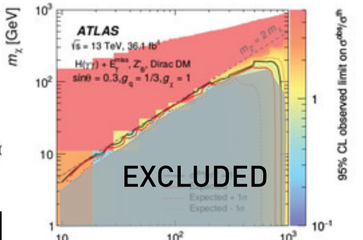


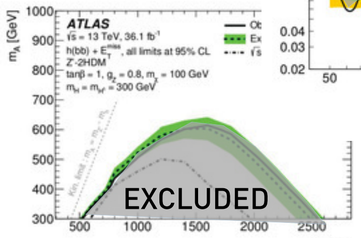
(a) CMS

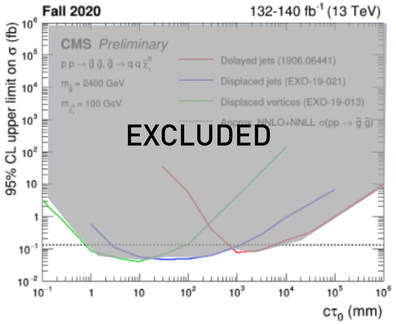


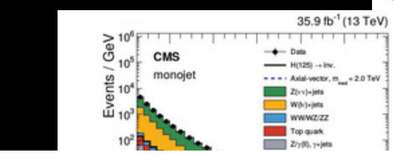


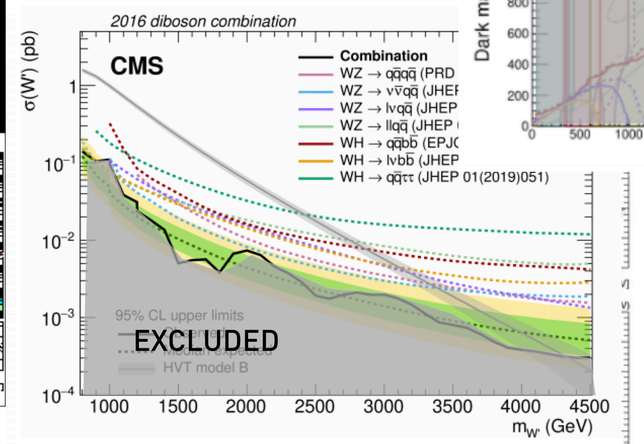
(b) ATLAS

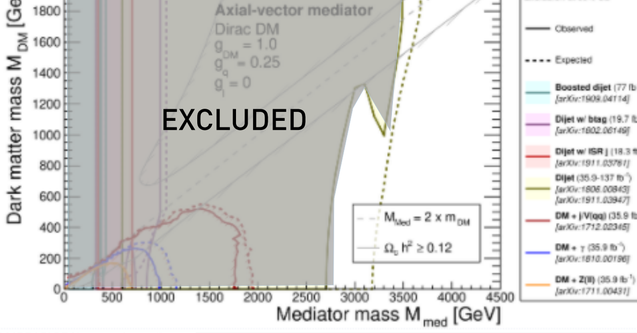


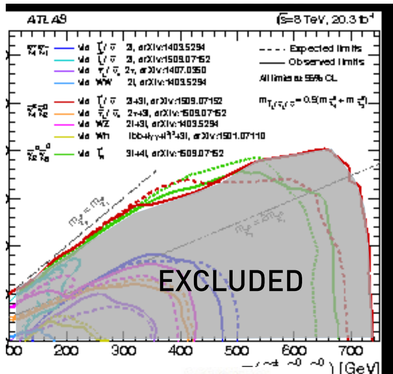


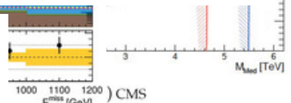


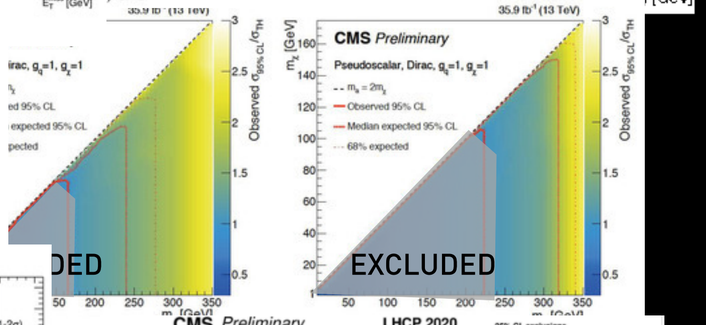


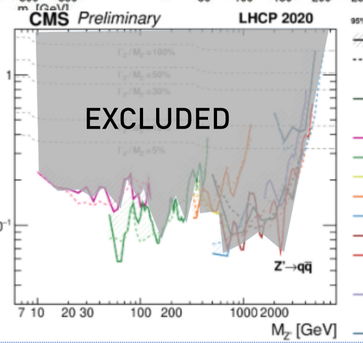




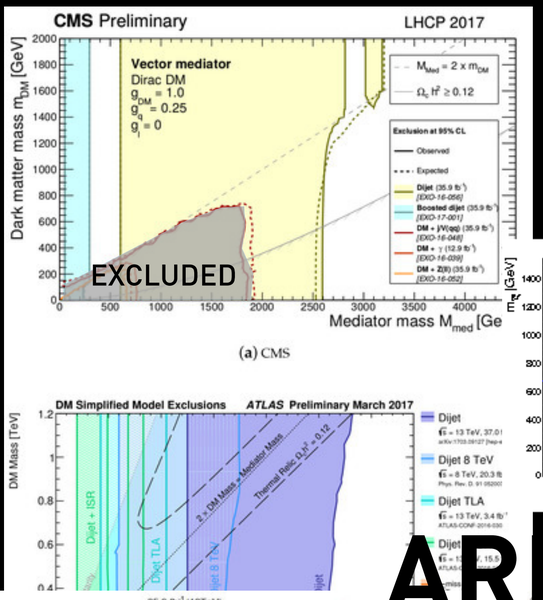




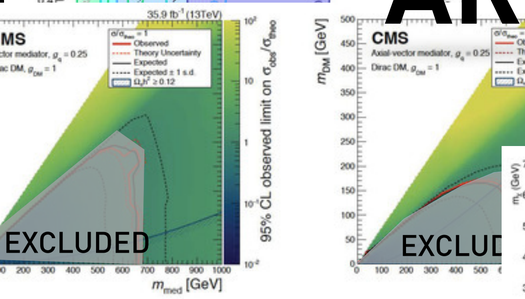


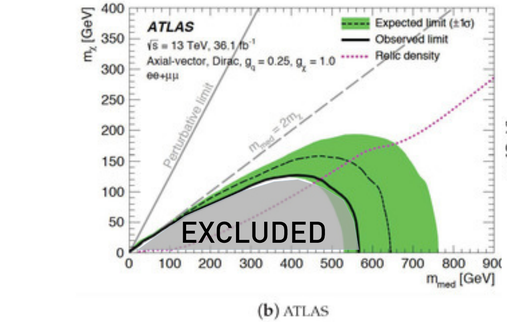


ARE WE LOOKING IN THE WRONG WAY?

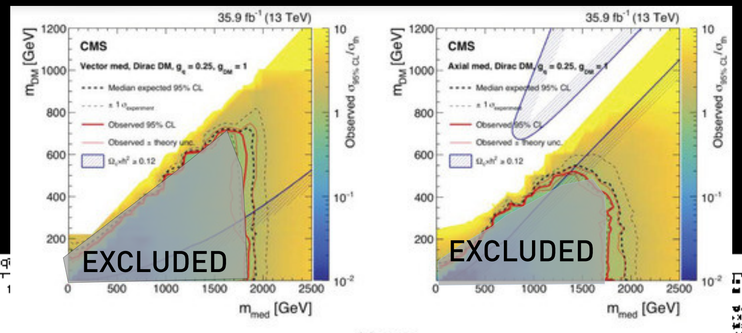


(a) CMS

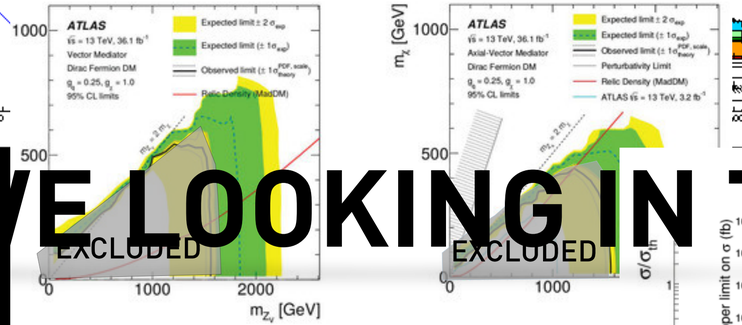




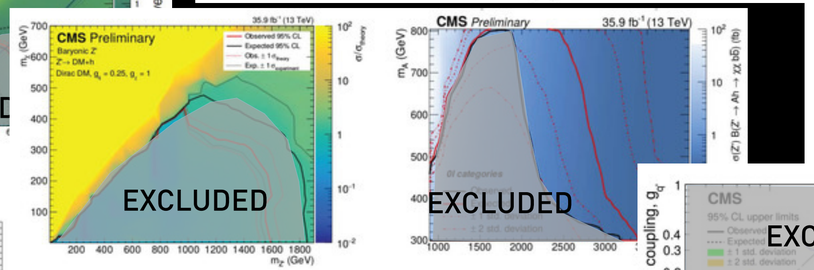
(b) ATLAS



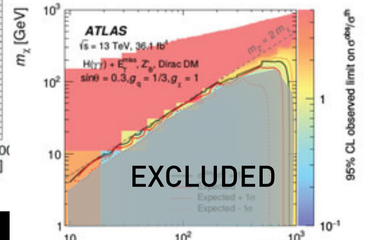
(a) CMS

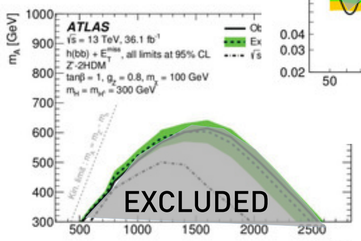


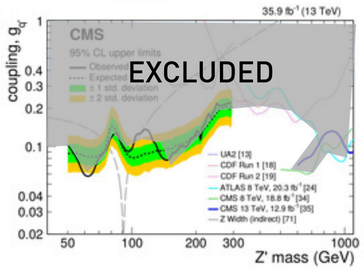
(b) ATLAS



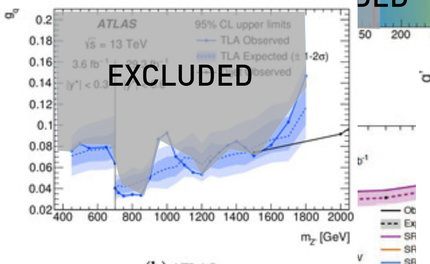
(a) CMS



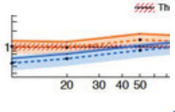


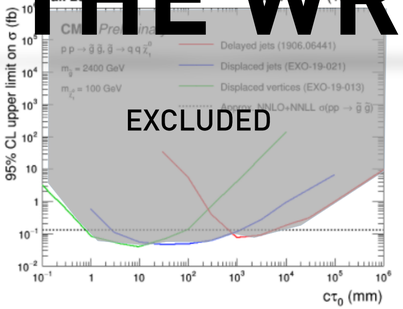


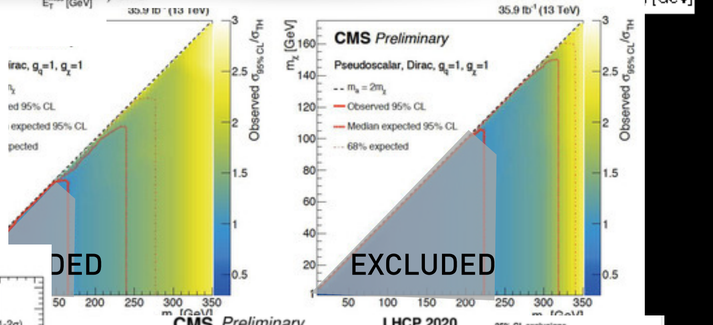
(a) CMS

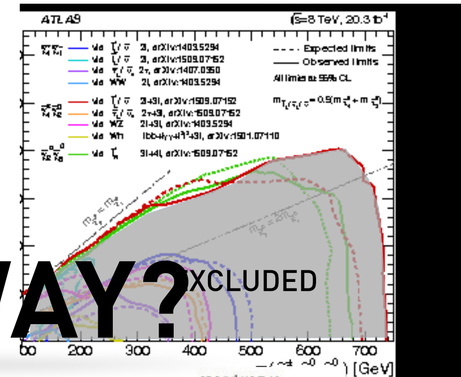


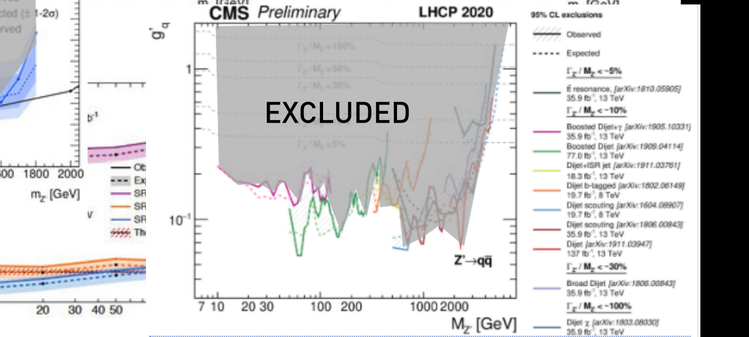
(b) ATLAS











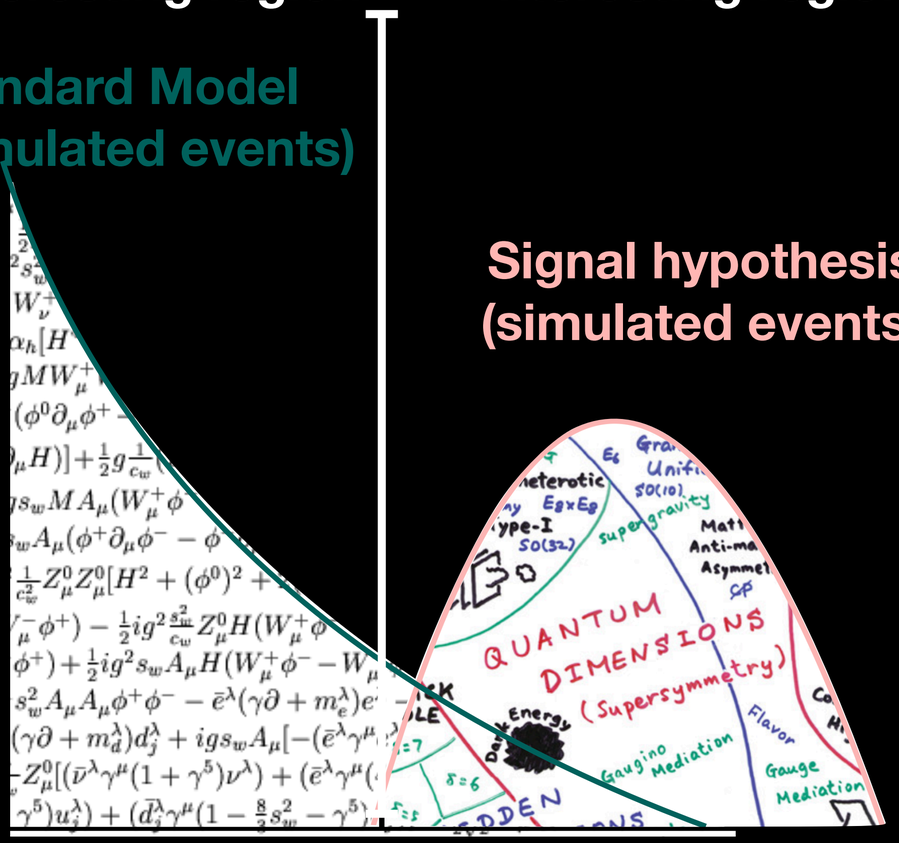
Bias in particle physics

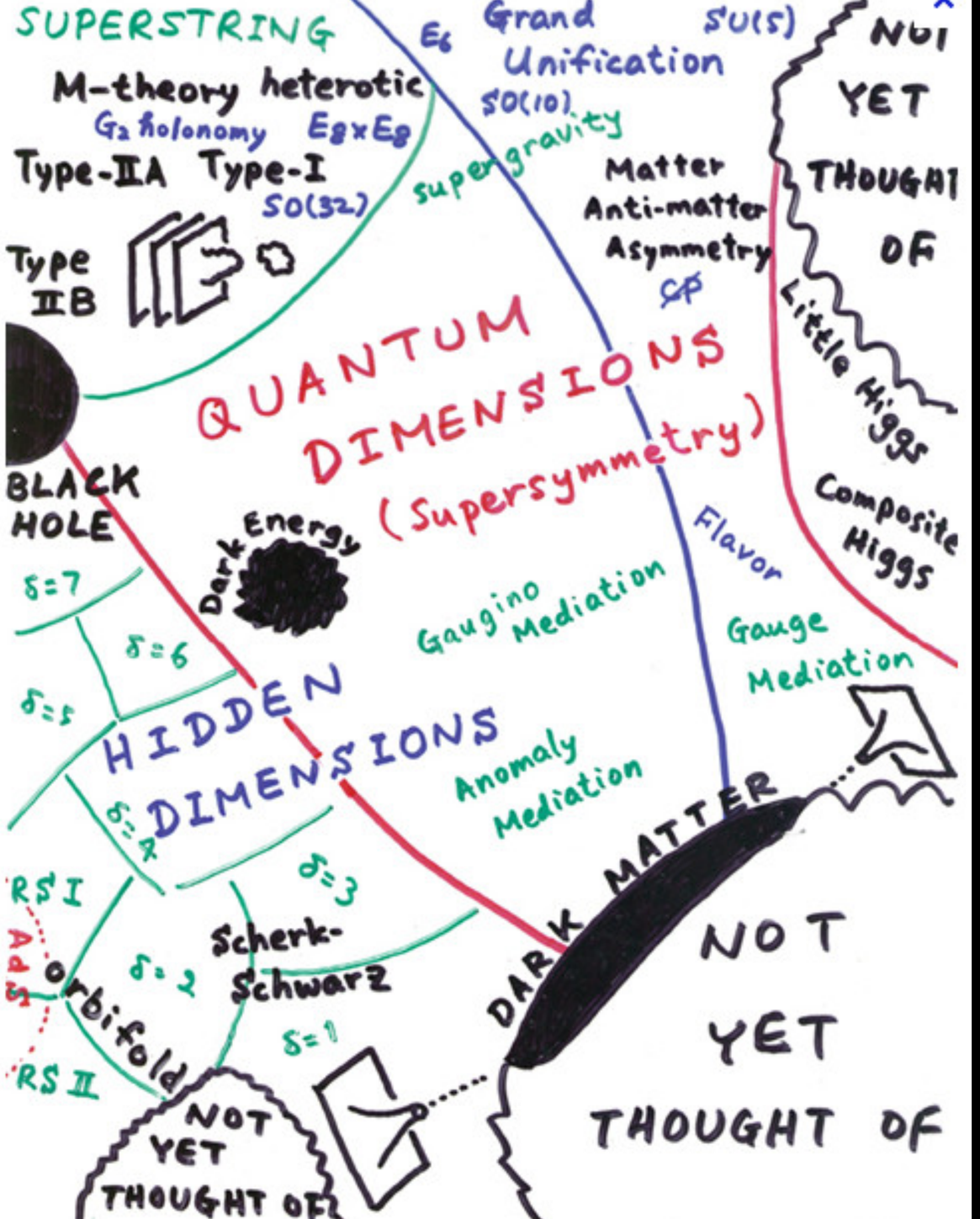
Not interesting region

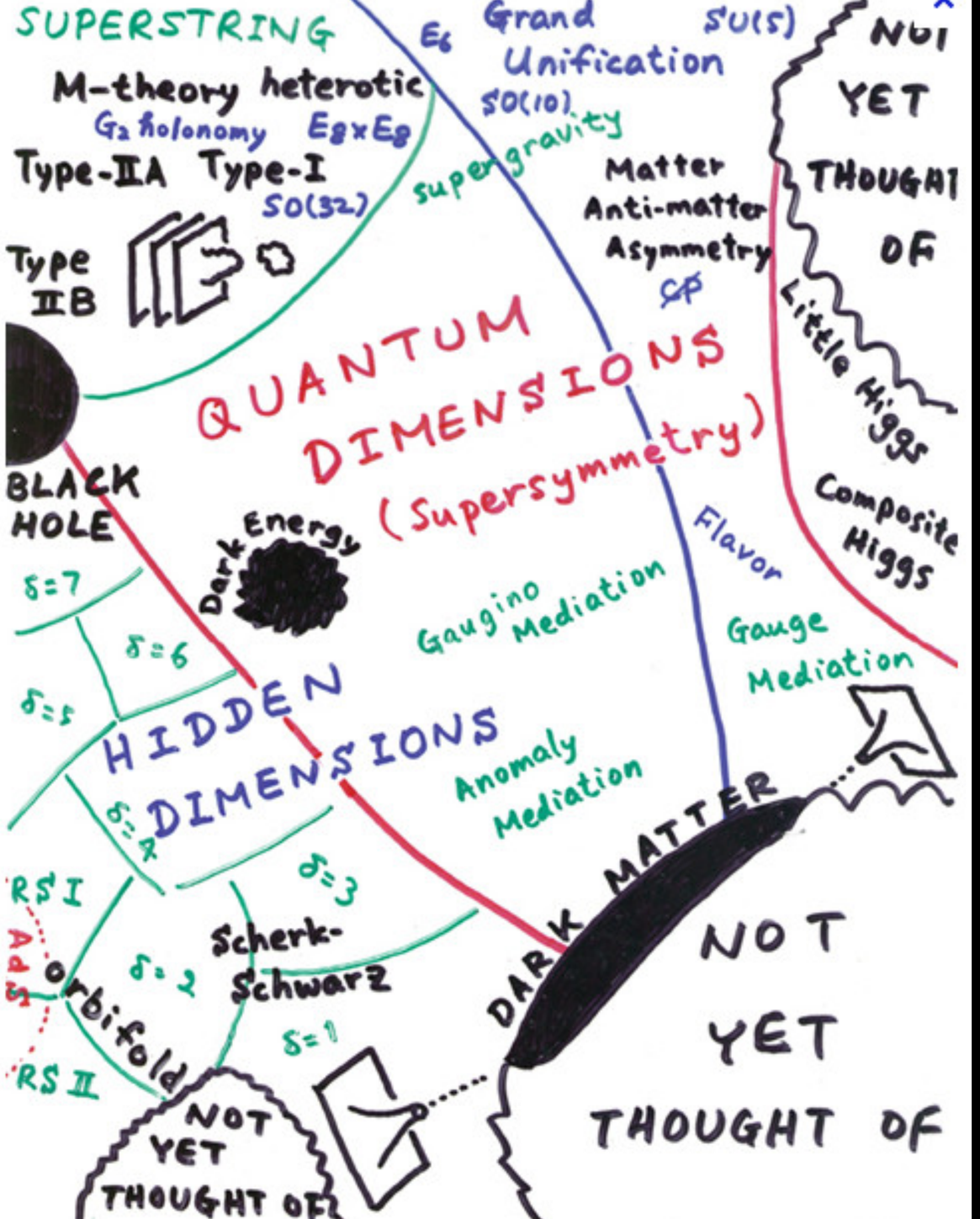
Interesting region

Standard Model
(simulated events)

Signal hypothesis
(simulated events)

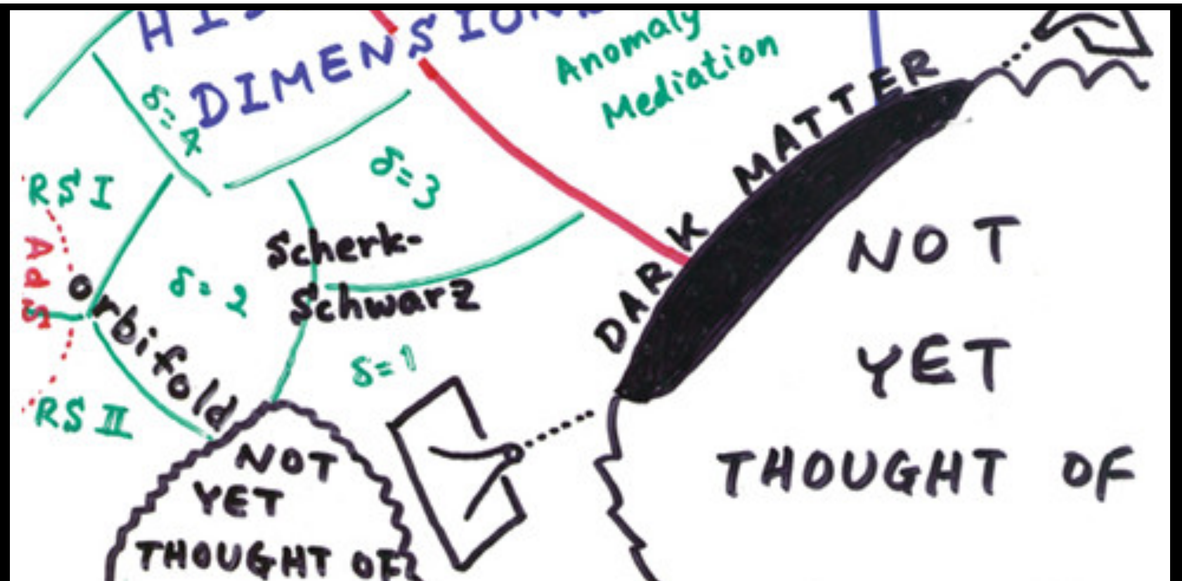



SUPERSTRING
M-theory
heterotic
G_2 holonomy
$E_8 \times E_8$
Type-IIA
Type-I
SO(32)
Type IIB
E_6
Grand Unification
SU(5)
SO(10)
supergravity
Matter Anti-matter Asymmetry
CP
NOT YET THOUGHT OF
Little Higgs
Composite Higgs
QUANTUM DIMENSIONS
(Supersymmetry)
BLACK HOLE
Dark Energy
Gaugino Mediation
Flavor
Gauge Mediation
Anomaly Mediation
HIDDEN DIMENSIONS
$\delta=7$
$\delta=6$
$\delta=5$
$\delta=4$
$\delta=3$
$\delta=2$
$\delta=1$
RSI
AdS
orbifold
RS II
Scherk-Schwarz
DARK MATTER
NOT YET THOUGHT OF
NOT YET THOUGHT OF


SUPERSTRING
M-theory
heterotic
G_2 holonomy
$E_8 \times E_8$
Type-IIA
Type-I
SO(32)
Type IIB
E_6
Grand Unification
SU(5)
SO(10)
supergravity
Matter Anti-matter Asymmetry
CP
NOT YET THOUGHT OF
Little Higgs
Composite Higgs
QUANTUM DIMENSIONS
(Supersymmetry)
BLACK HOLE
Dark Energy
Gaugino Mediation
Flavor
Gauge Mediation
Anomaly Mediation
HIDDEN DIMENSIONS
$\delta=7$
$\delta=6$
$\delta=5$
$\delta=4$
$\delta=3$
$\delta=2$
$\delta=1$
RSI
AdS
orbifold
RS II
Scherk-Schwarz
DARK MATTER
NOT YET THOUGHT OF
NOT YET THOUGHT OF

How do you search for something when

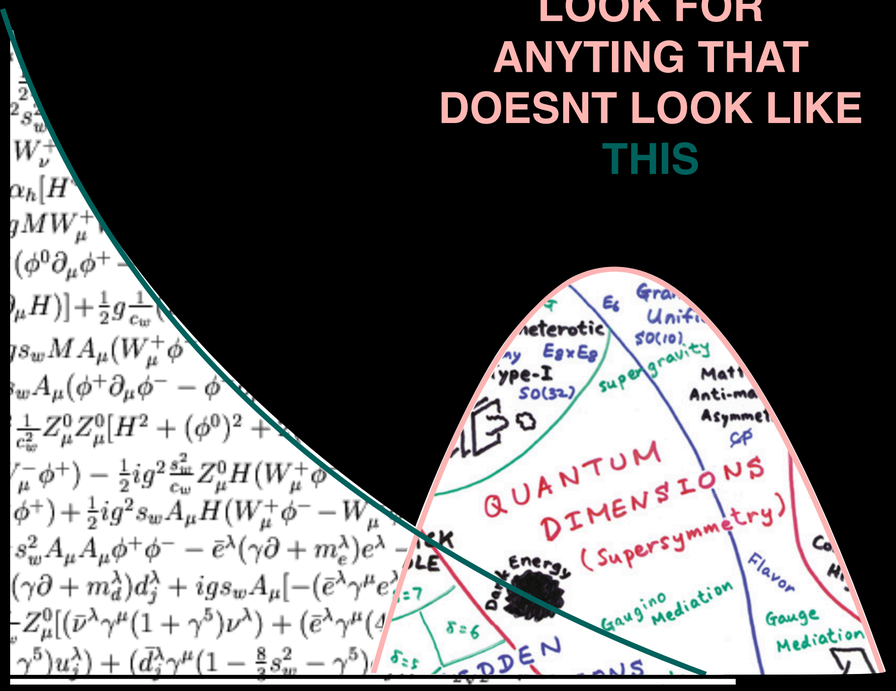
1) **You don't really know what that something is**
2) **You know vaguely what that something is, but the parameter space is huge**



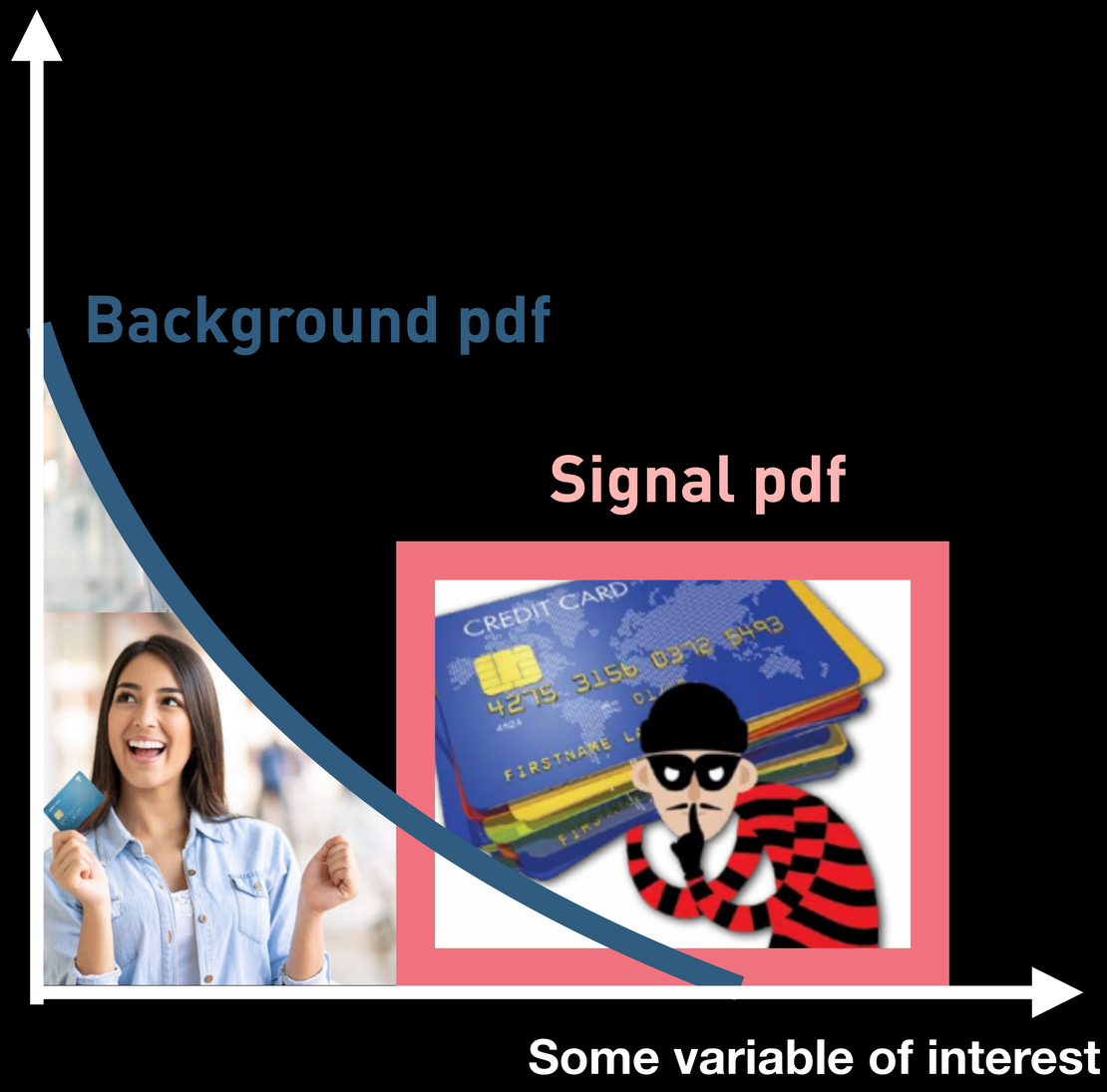
Anomaly detection for New Physics searches

LEARN THIS
FROM DATA

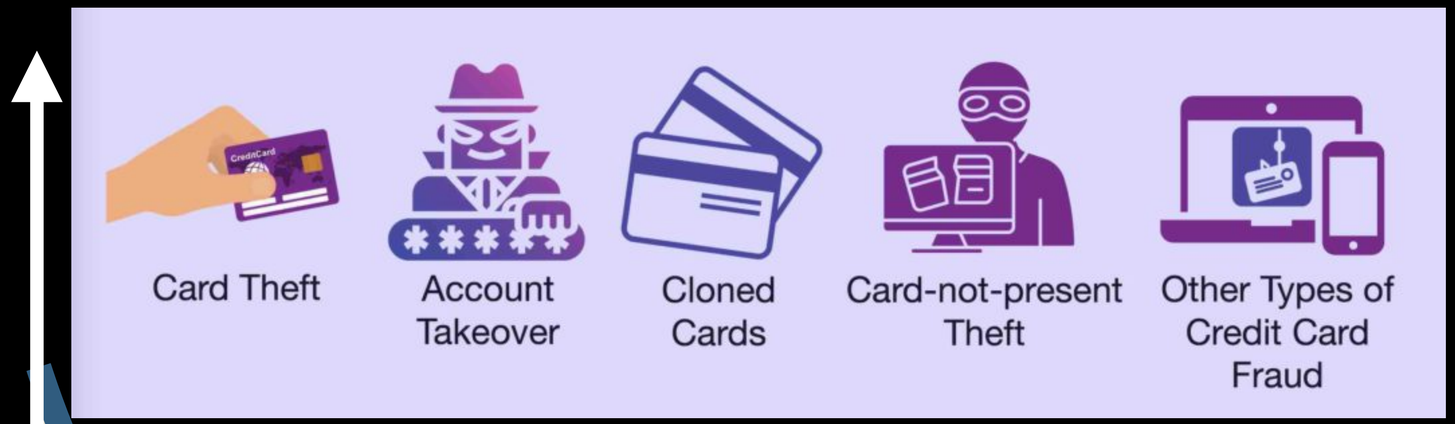
LOOK FOR
ANYTING THAT
DOESNT LOOK LIKE
THIS

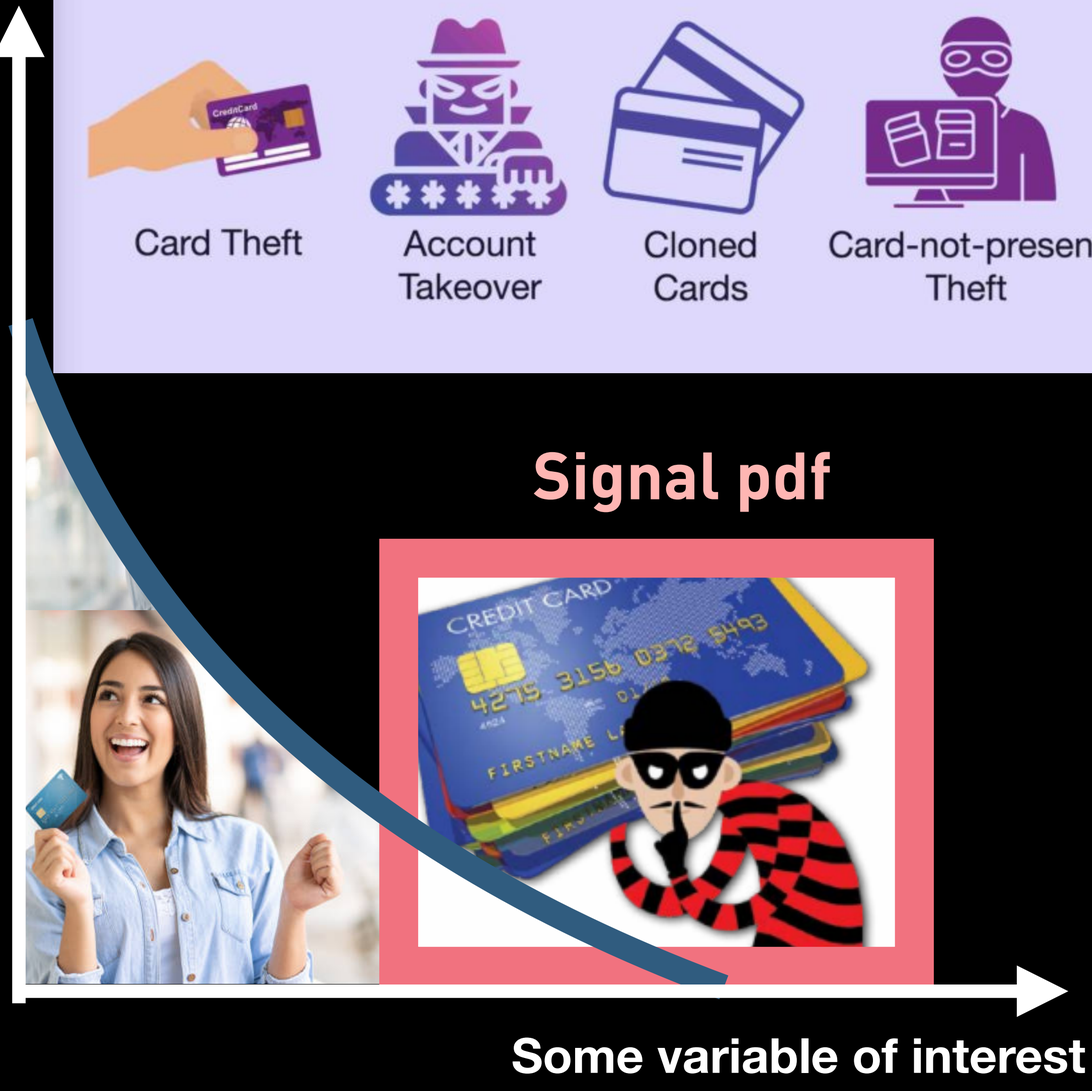


What is anomaly detection



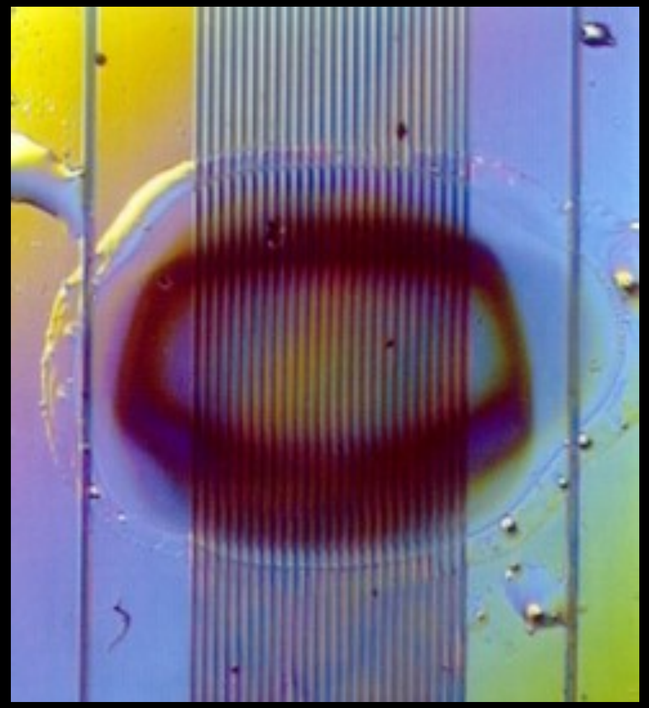
What is anomaly detection

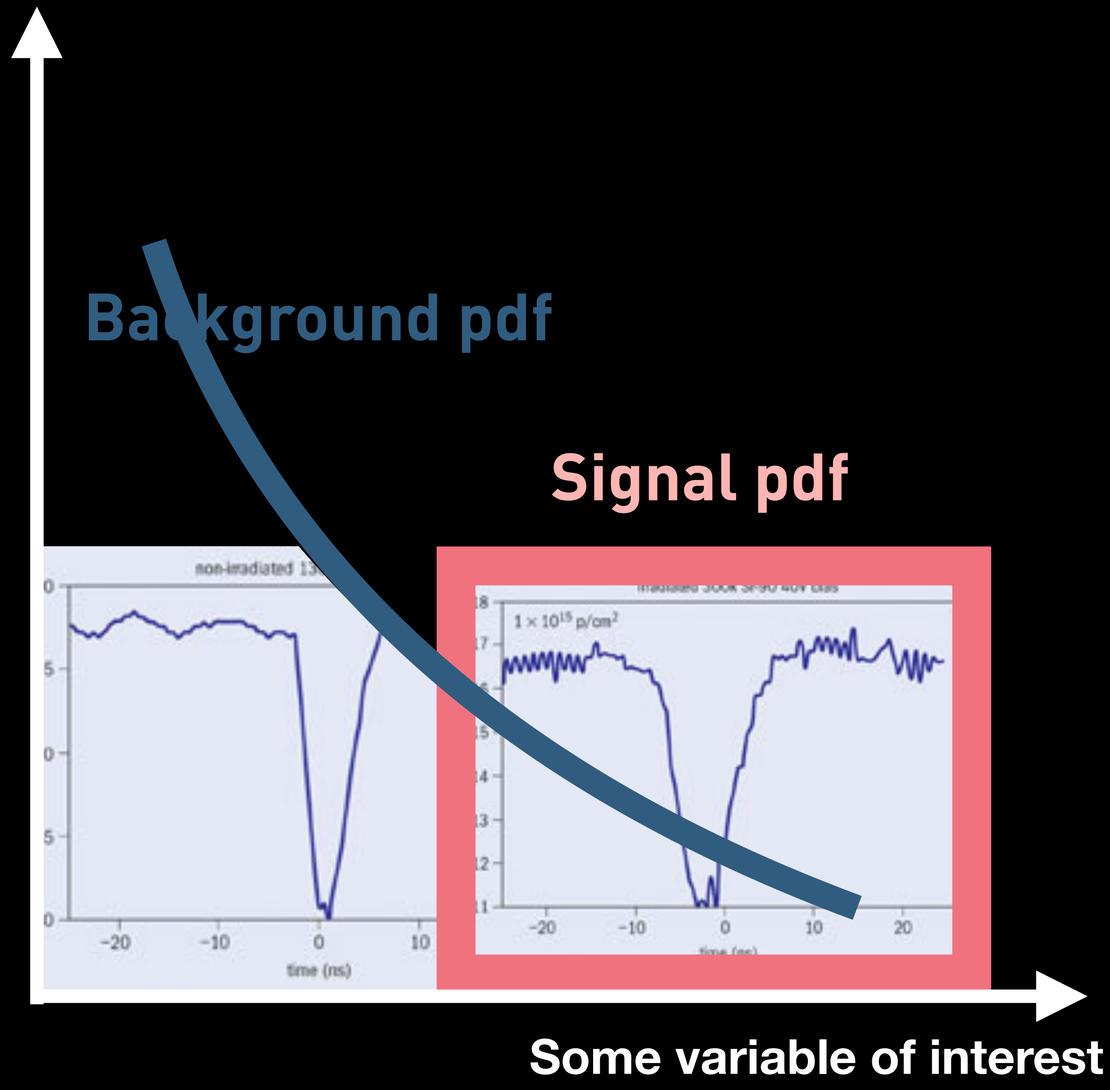




What is anomaly detection

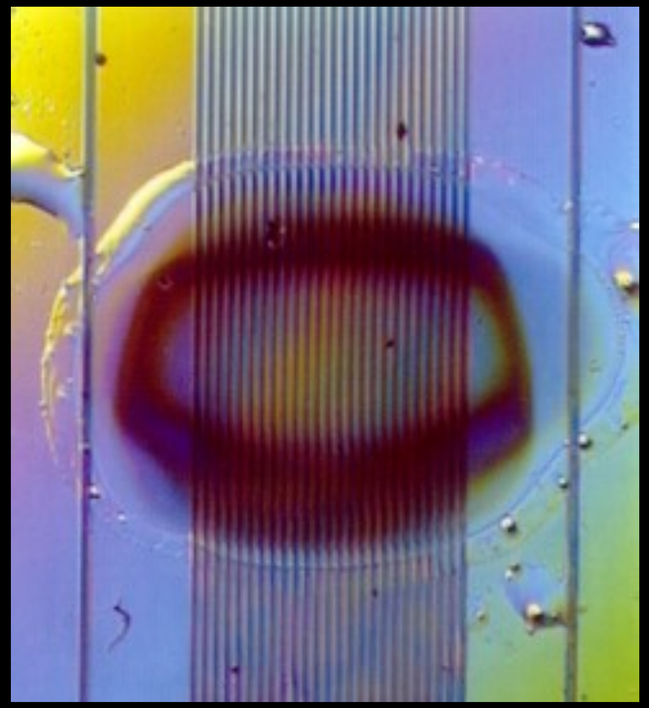
Radiation damage of silicon detector

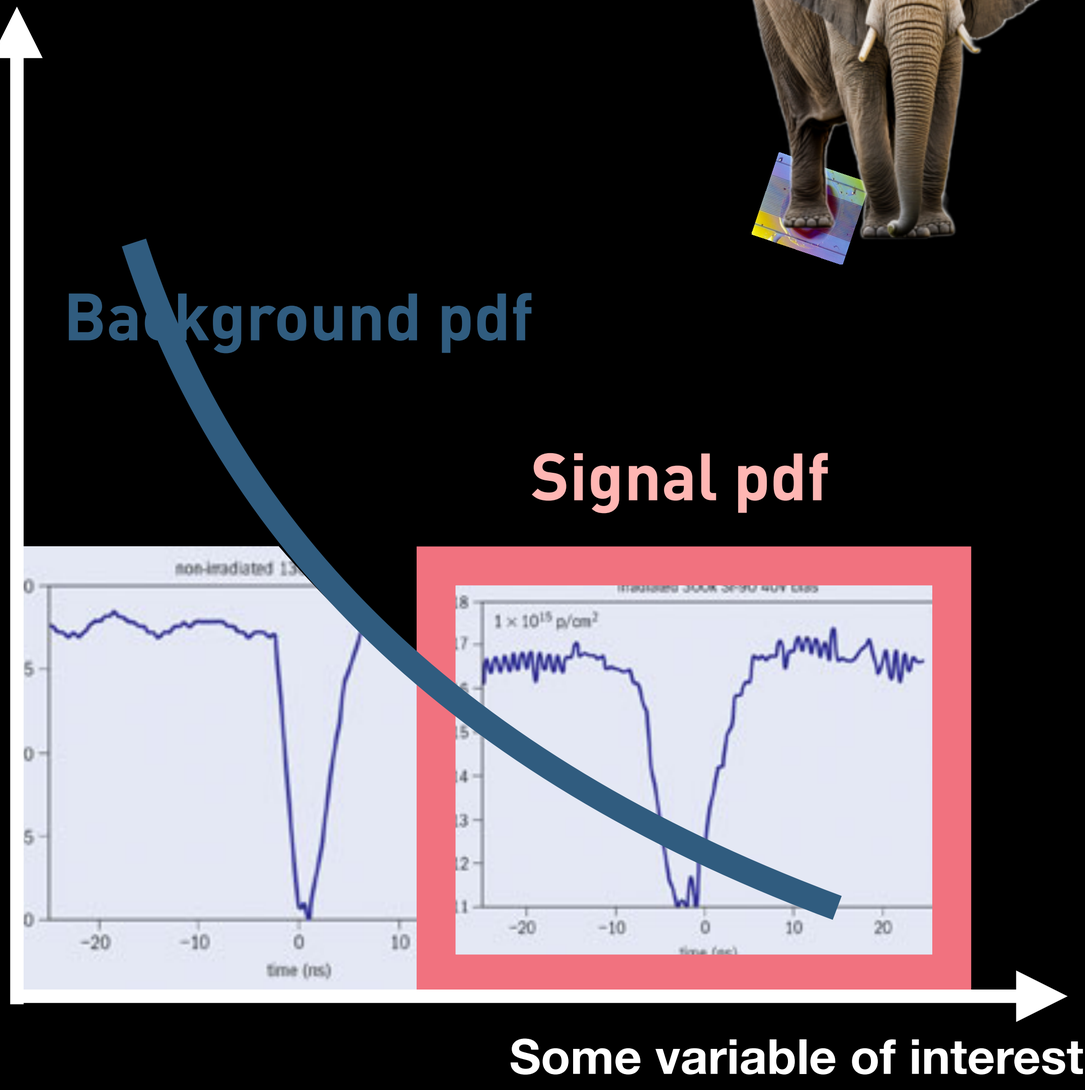




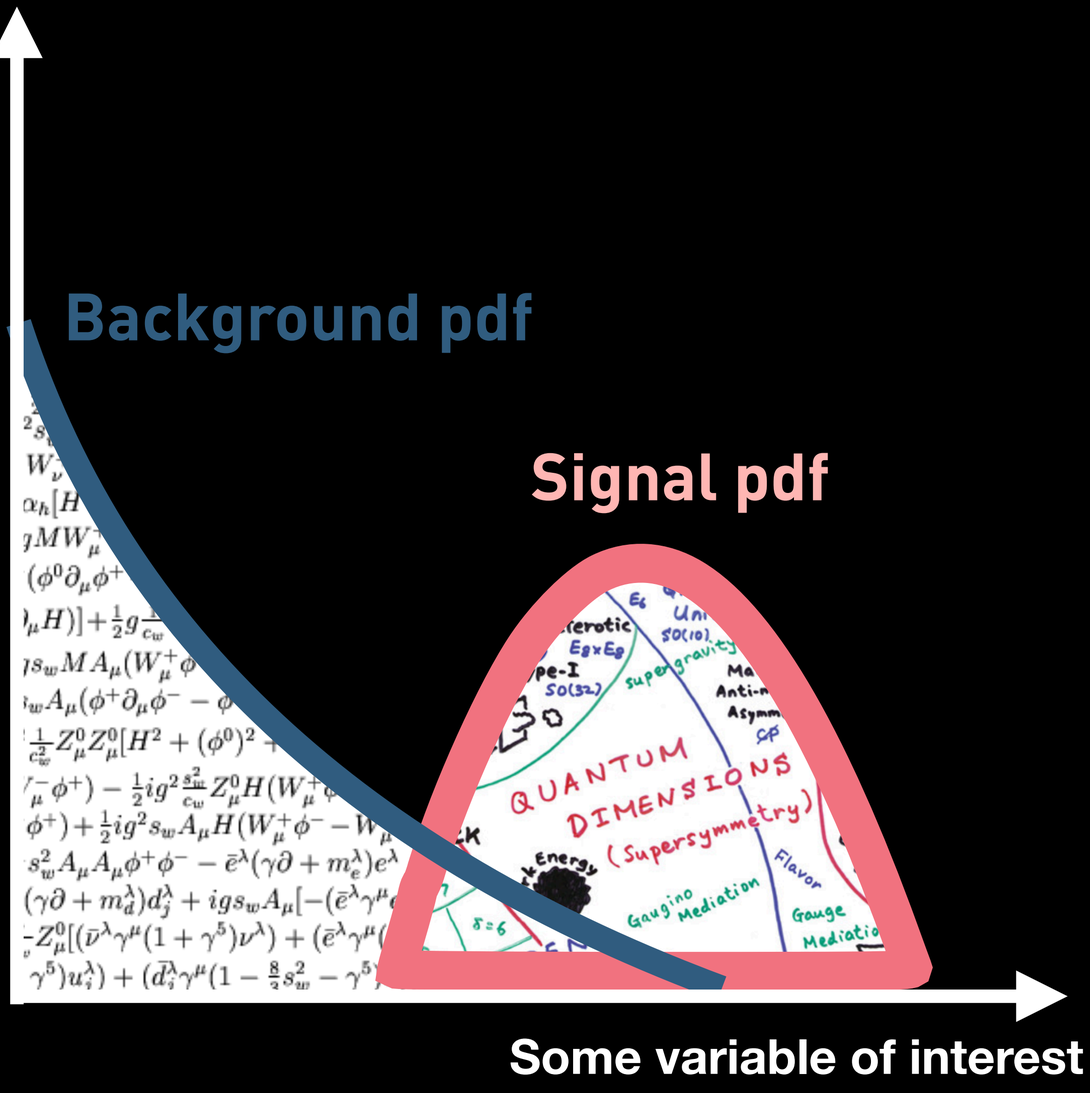
What is anomaly detection

Radiation damage of silicon detector

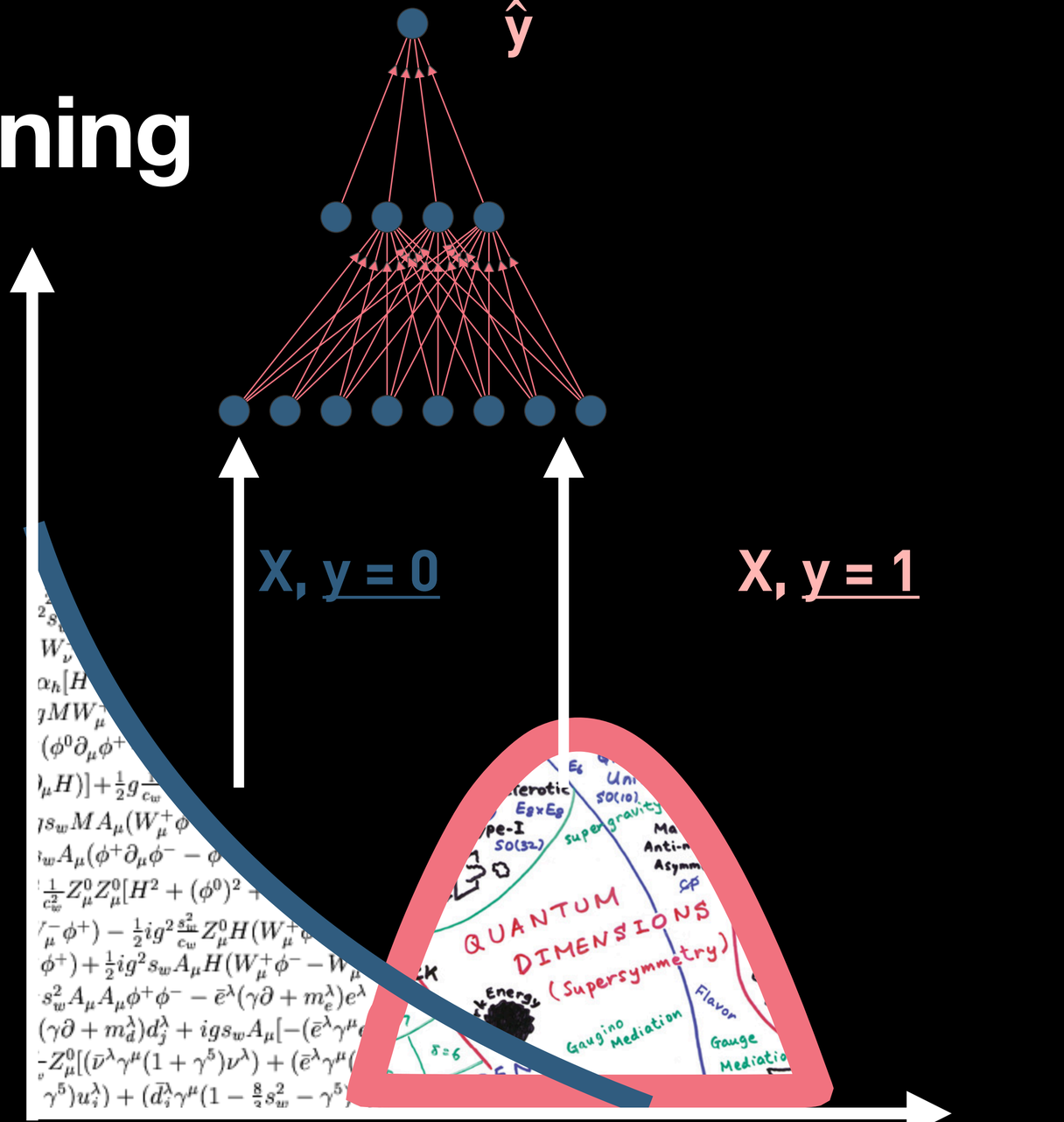




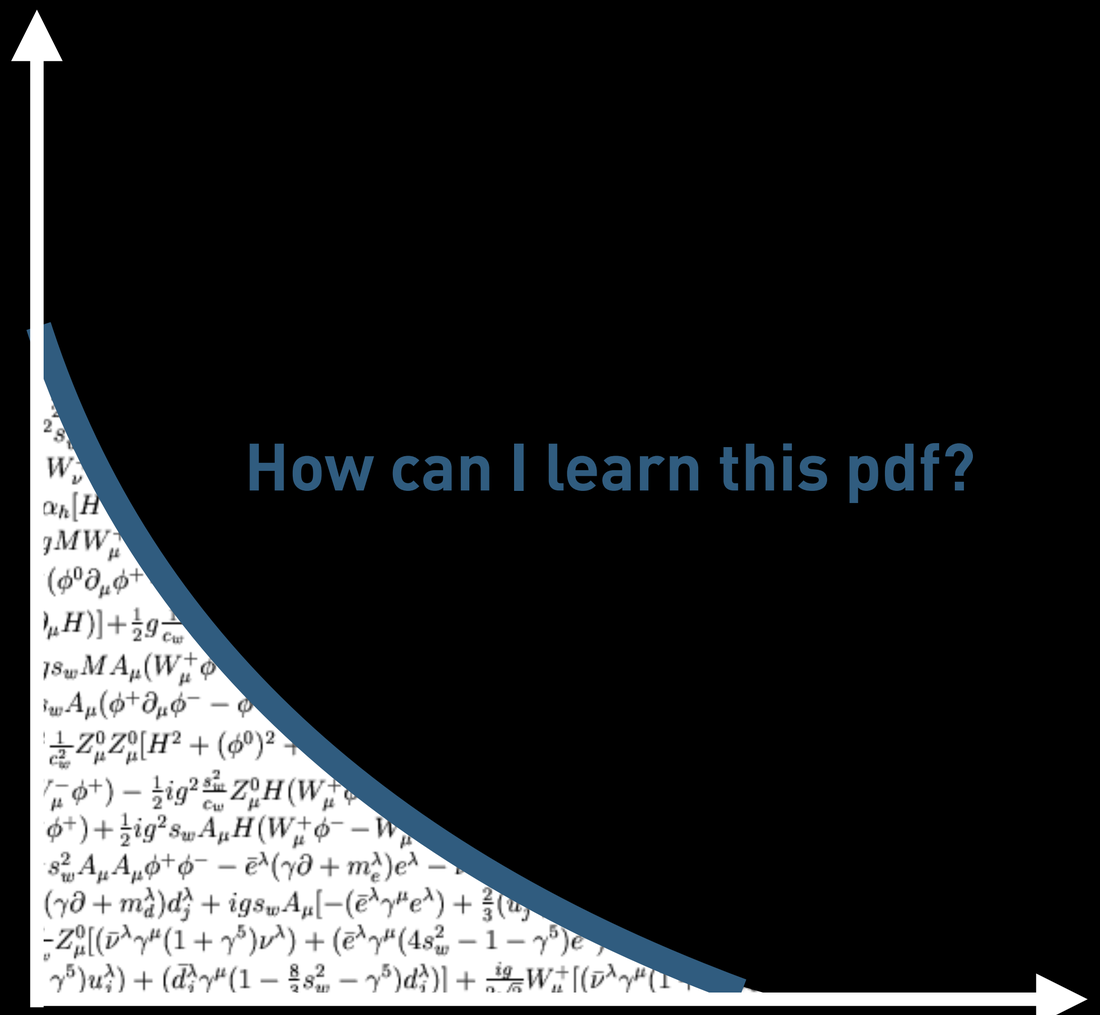
What is anomaly detection



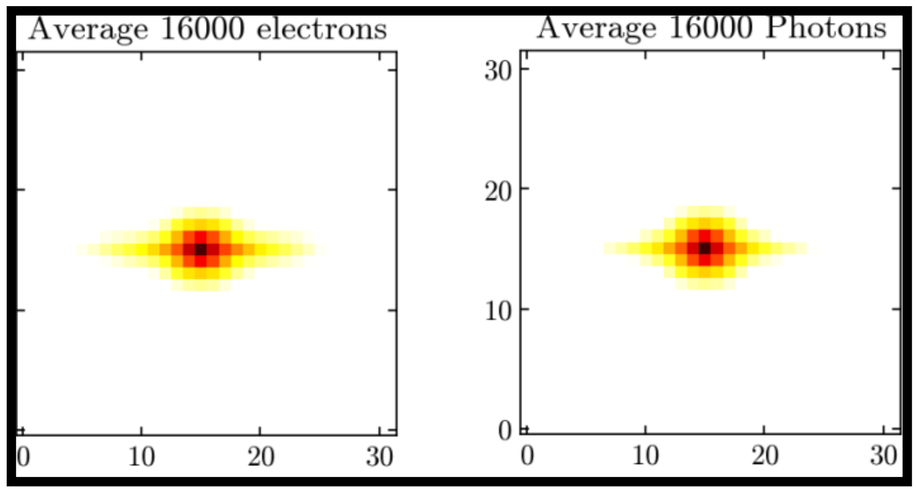
Supervised learning

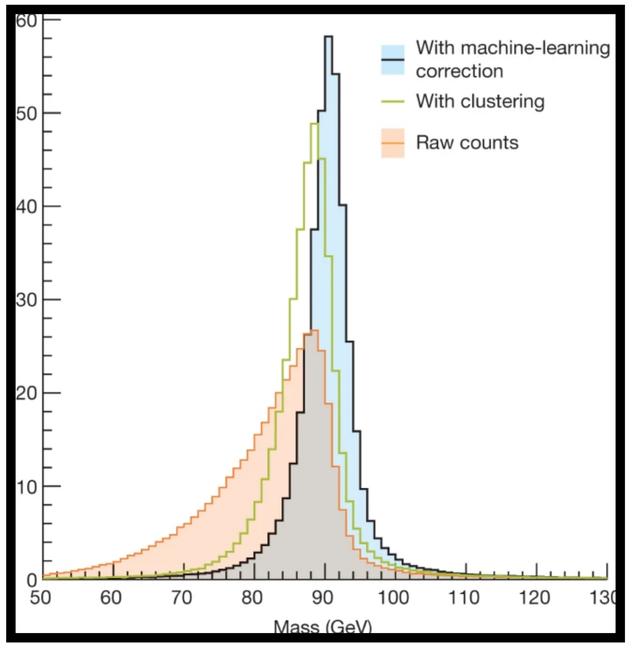


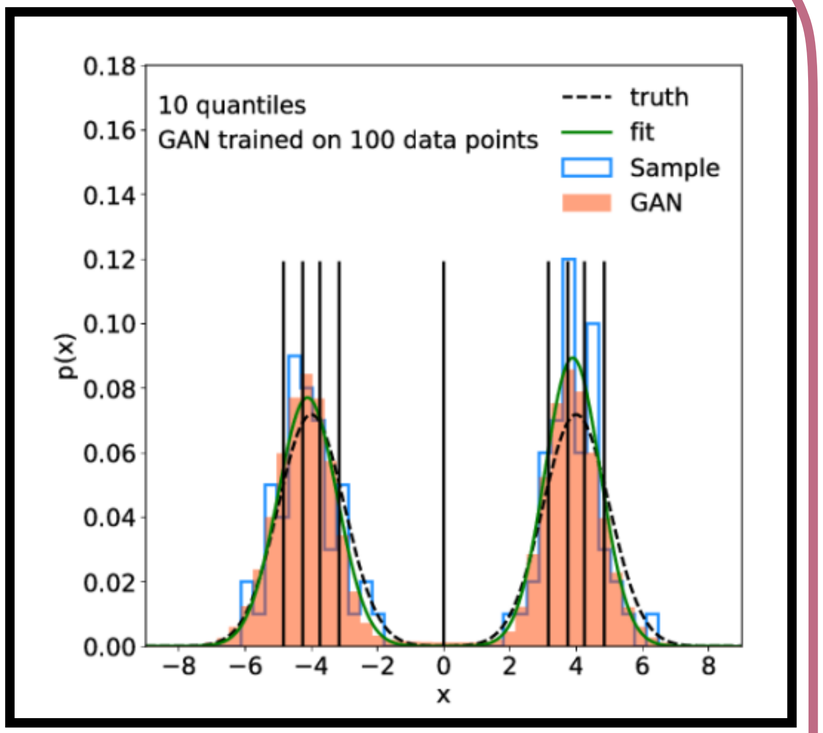
Anomaly detection


How can I learn this pdf?
Some variable(s) of interest X

Types of models







Classification

- Predict value from a finite set {1,…,C}
- Training: Pairs of input and ground-truth

Supervised Learning

Regression

- Prediction of continuous-valued vector $y \in \mathbb{R}^K$
- Training: Pairs of input and ground-truth

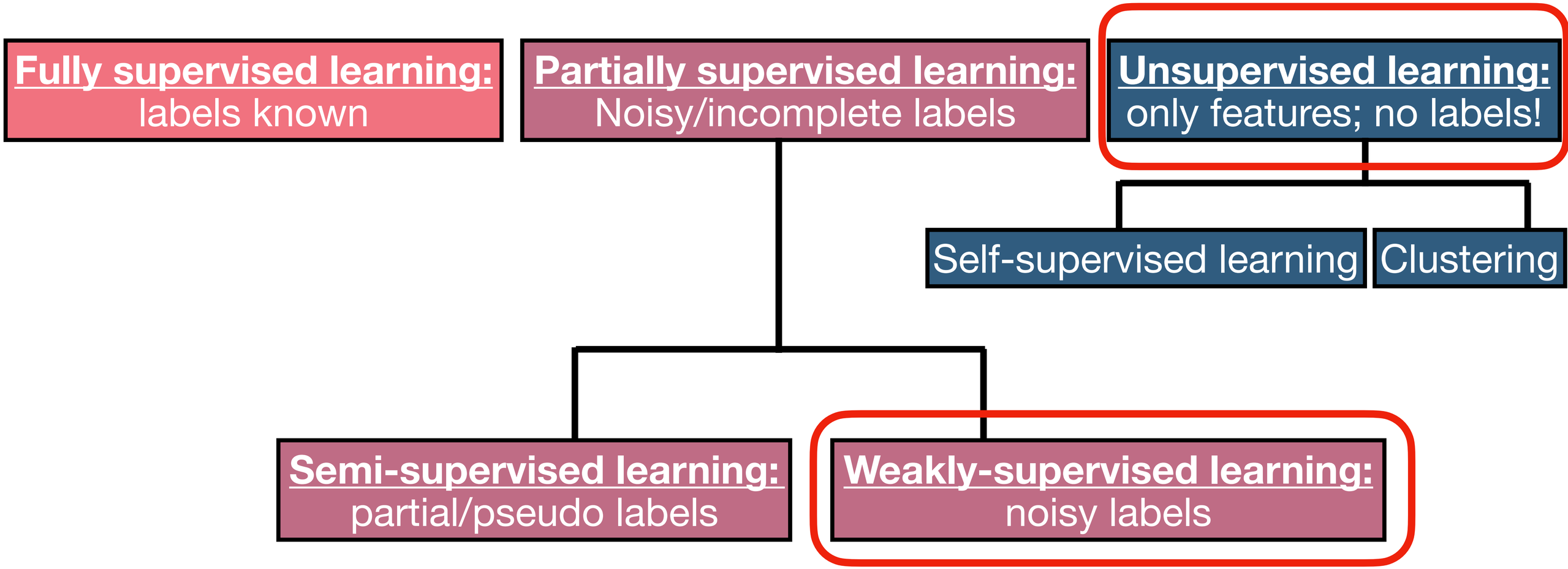
Supervised Learning

Density Modeling

- Model probability density function of data μ_x
- Training: values x_N without values to predict

Unsupervised Learning

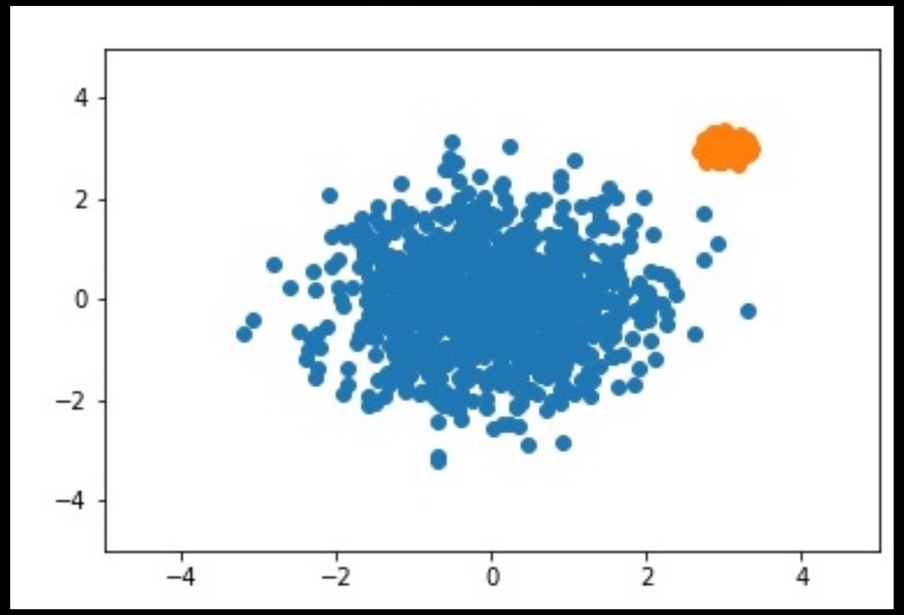
Types of learning



Types of anomaly detection

Outlier detection

Find (non-resonant) out-of-distribution datapoints
Often (variational) auto-encoders
Useful for **triggering/“selecting”**!

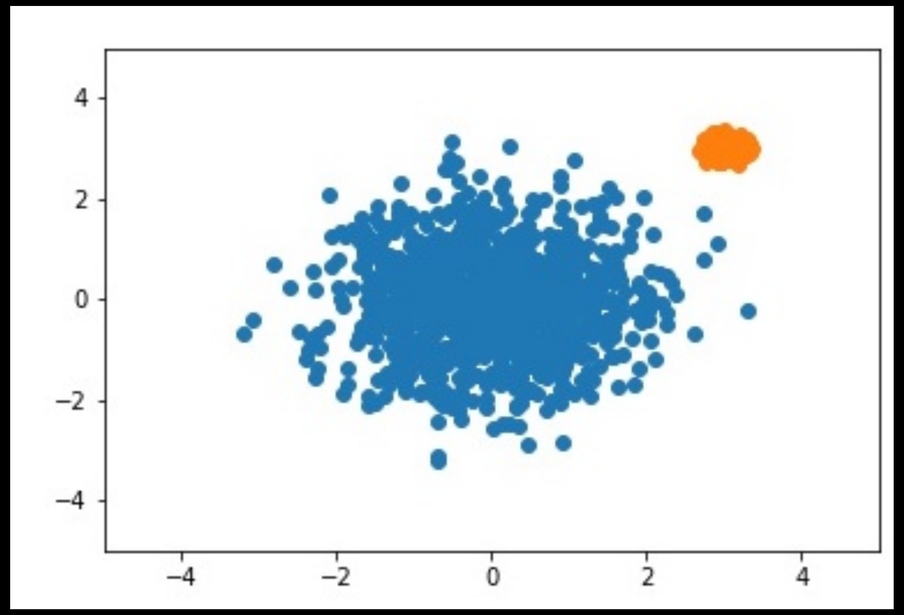


Unsupervised learning:
only features; no labels!

Types of anomaly detection

Outlier detection

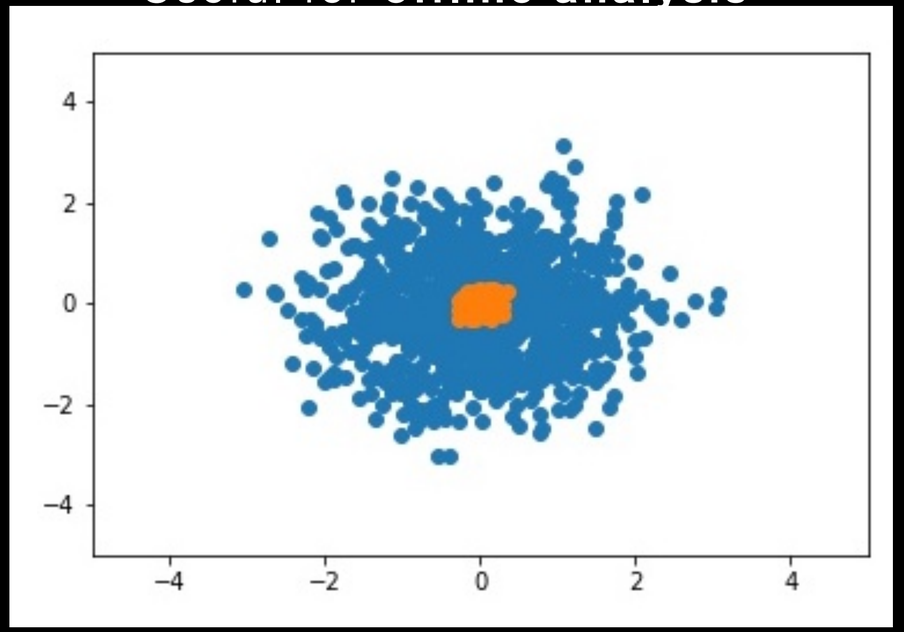
Find (non-resonant) out-of-distribution datapoints
Often (variational) auto-encoders
Useful for **triggering/“selecting”**!



Unsupervised learning:
only features; no labels!

Detecting overdensities

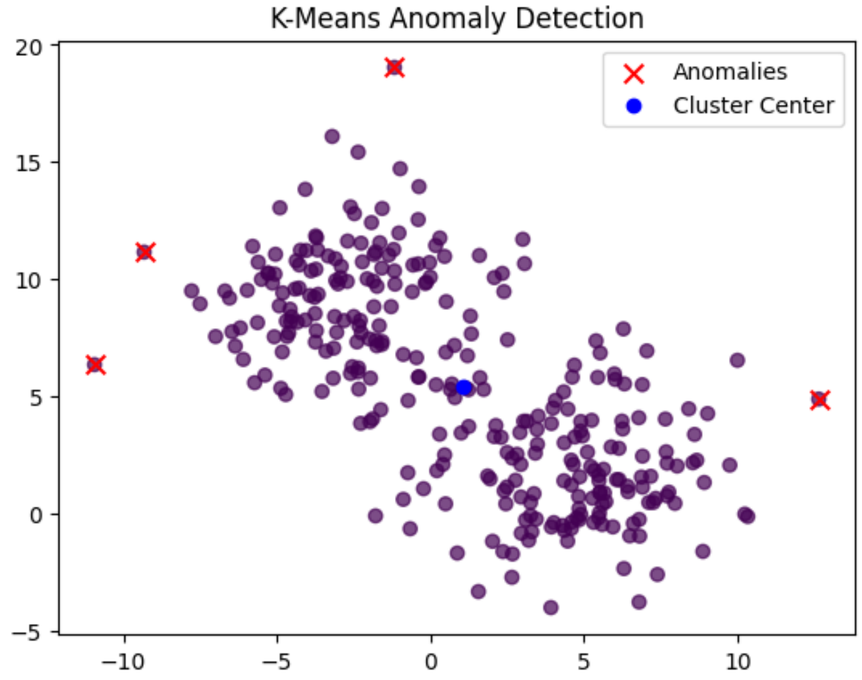
Find (resonant) overdensities in distributions
Density estimation methods
Useful for **offline analysis**

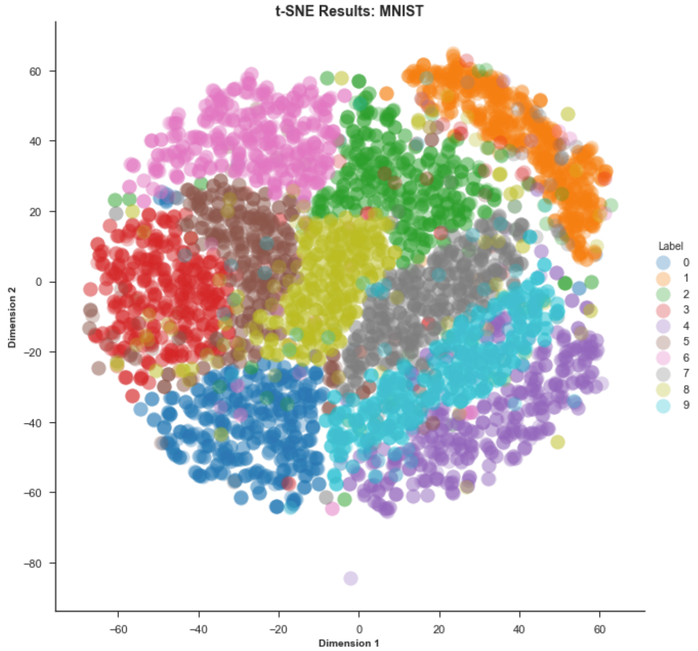


Weakly-supervised learning:
noisy labels

Unsupervised learning

- **Clustering** (K-means, hierarchical): “Summary of data” through cluster centres
- **Dimensionality reduction** (PCA, t-SNE, autoencoders): data compression



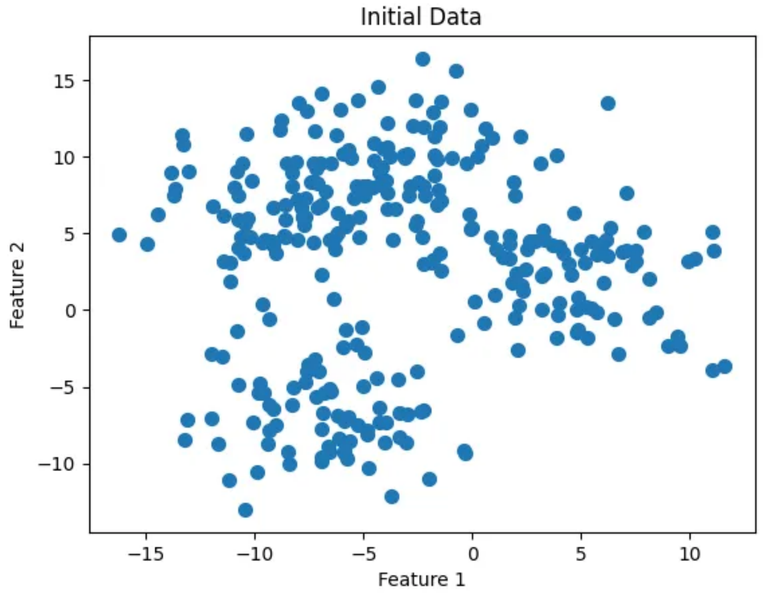


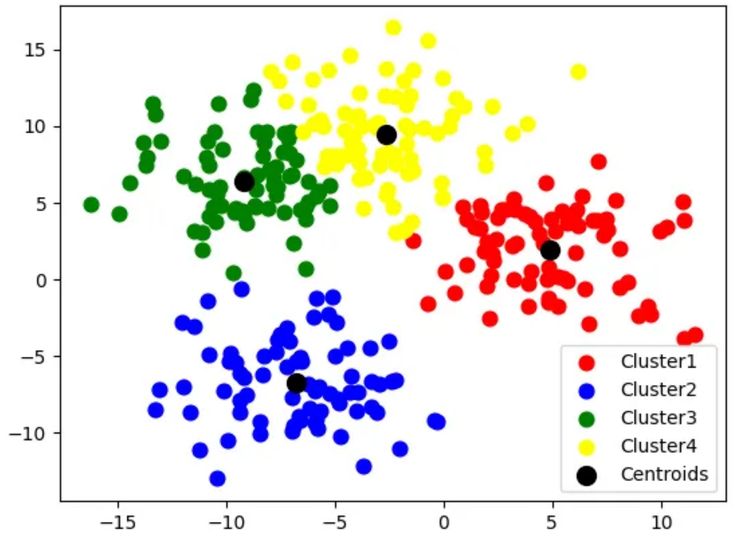
Unsupervised learning

- **Given**: unlabelled data

- **Goal**: find hidden structure/patterns
  - e.g hidden structure *is* a clustering of data
  - Generative model of data $P(x)$
  - Low dimensional summary of the data (compact preprocessing)

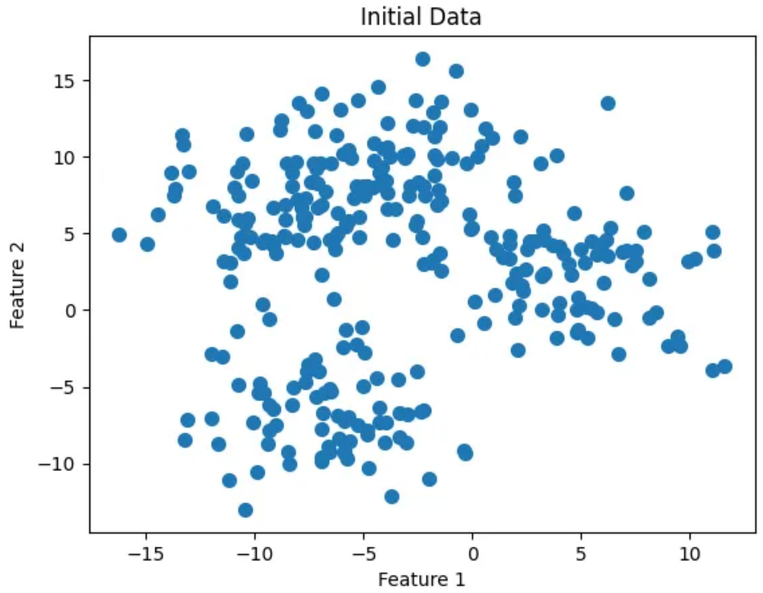
K-means (centroid) clustering

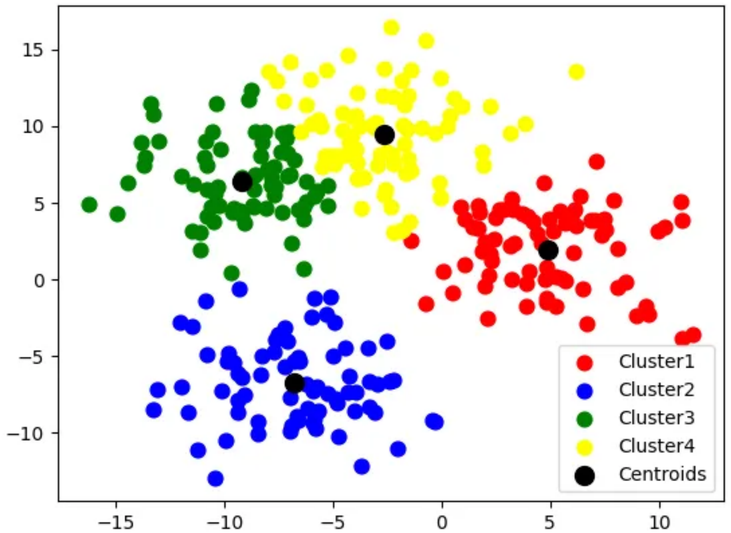




- Given $S = \{x_i\}_{i=1}^{N}$, find

K-means (centroid) clustering

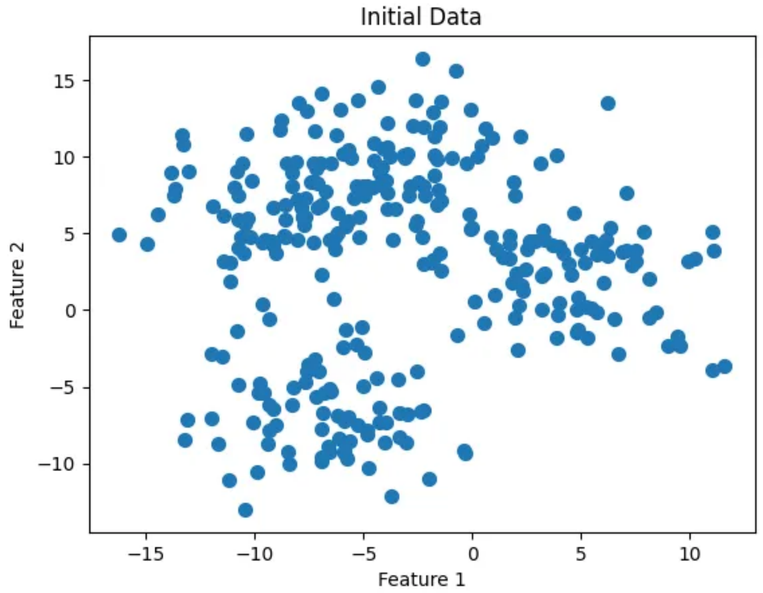


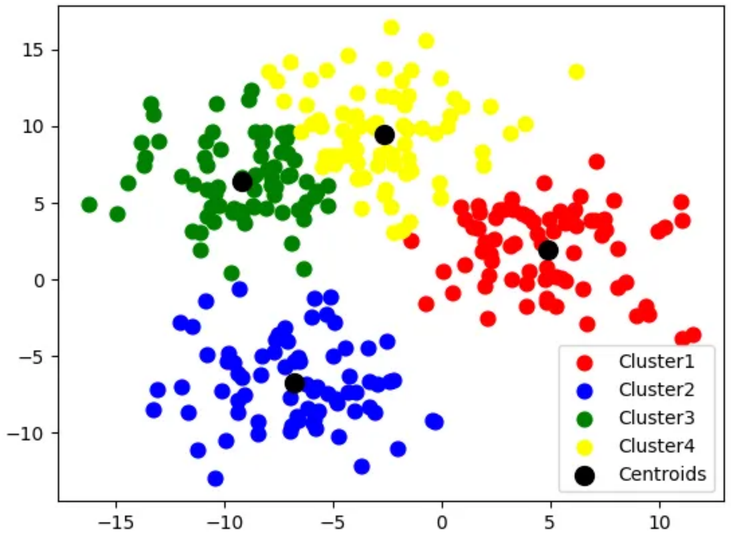


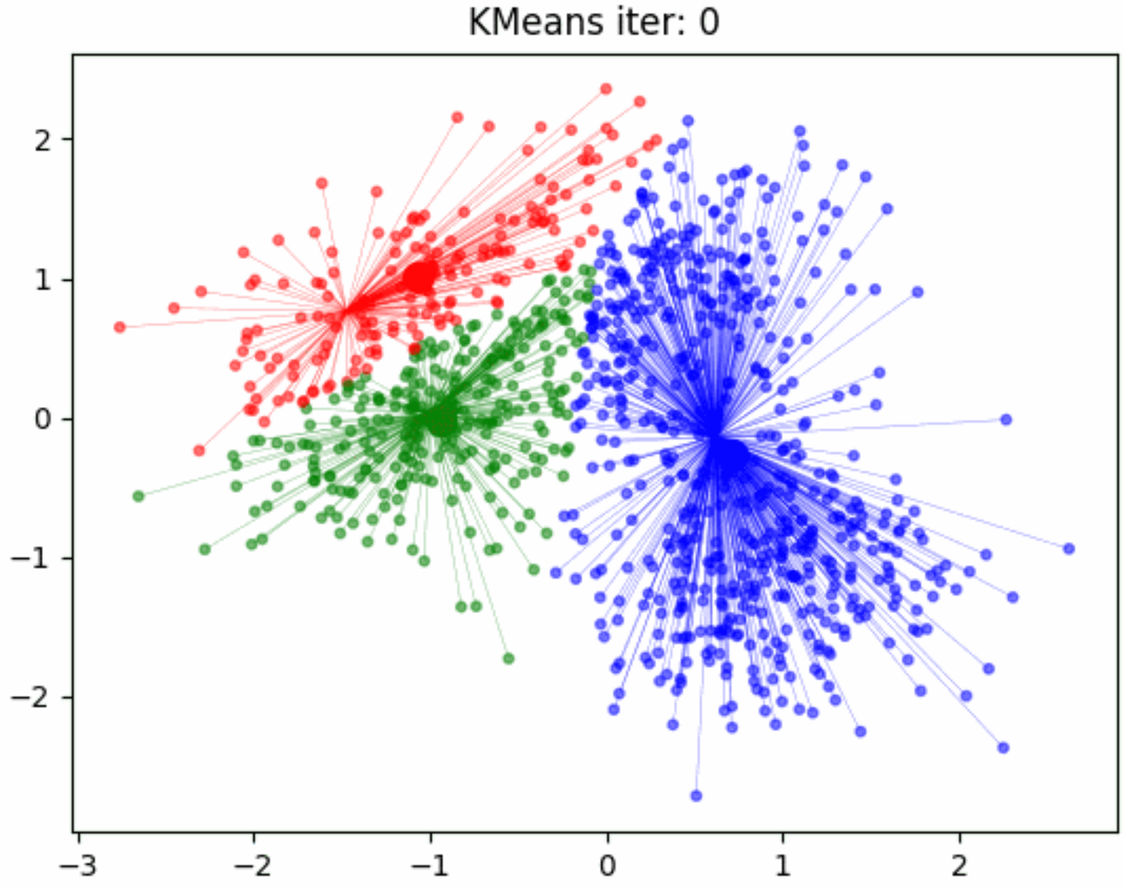
- Given $S = \{x_i\}_{i=1}^{N}$, find

$$\underset{S=C_1 \cup \cdots \cup C_k, \{c_1,\ldots,c_k\}}{\operatorname{argmin}} \sum_k \sum_{x \in C_k} \|x - c_k\|^2$$

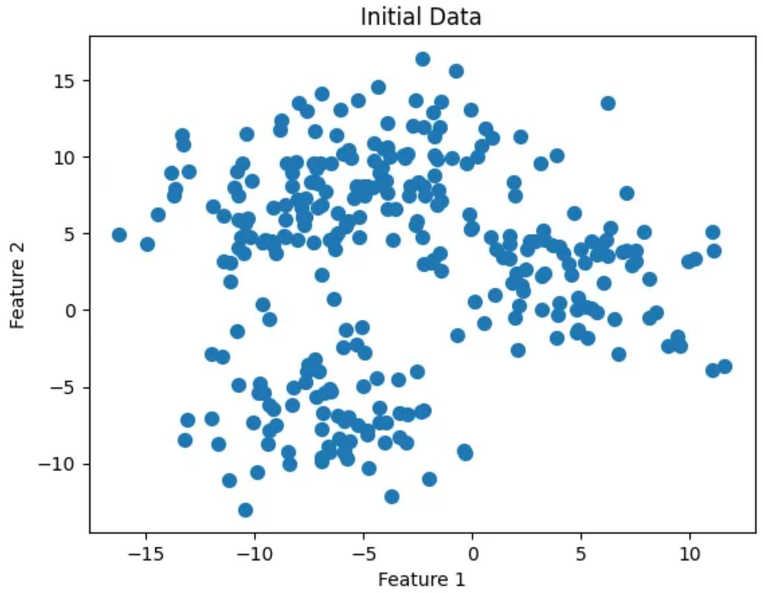
K-means (centroid) clustering

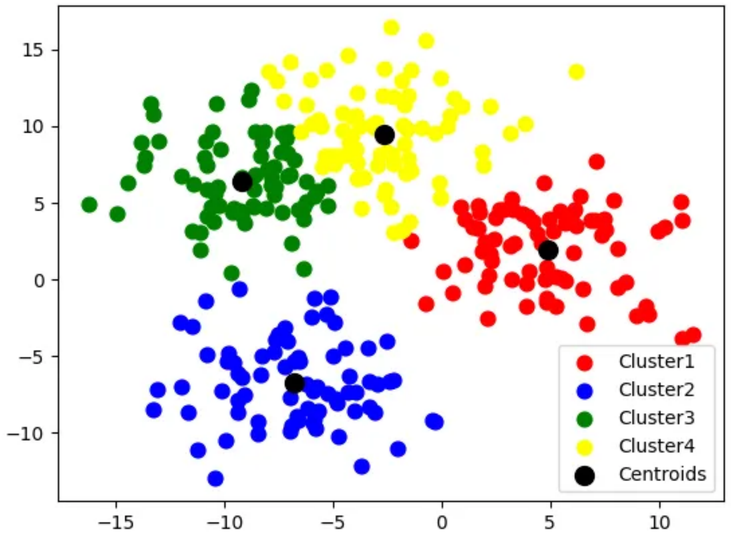


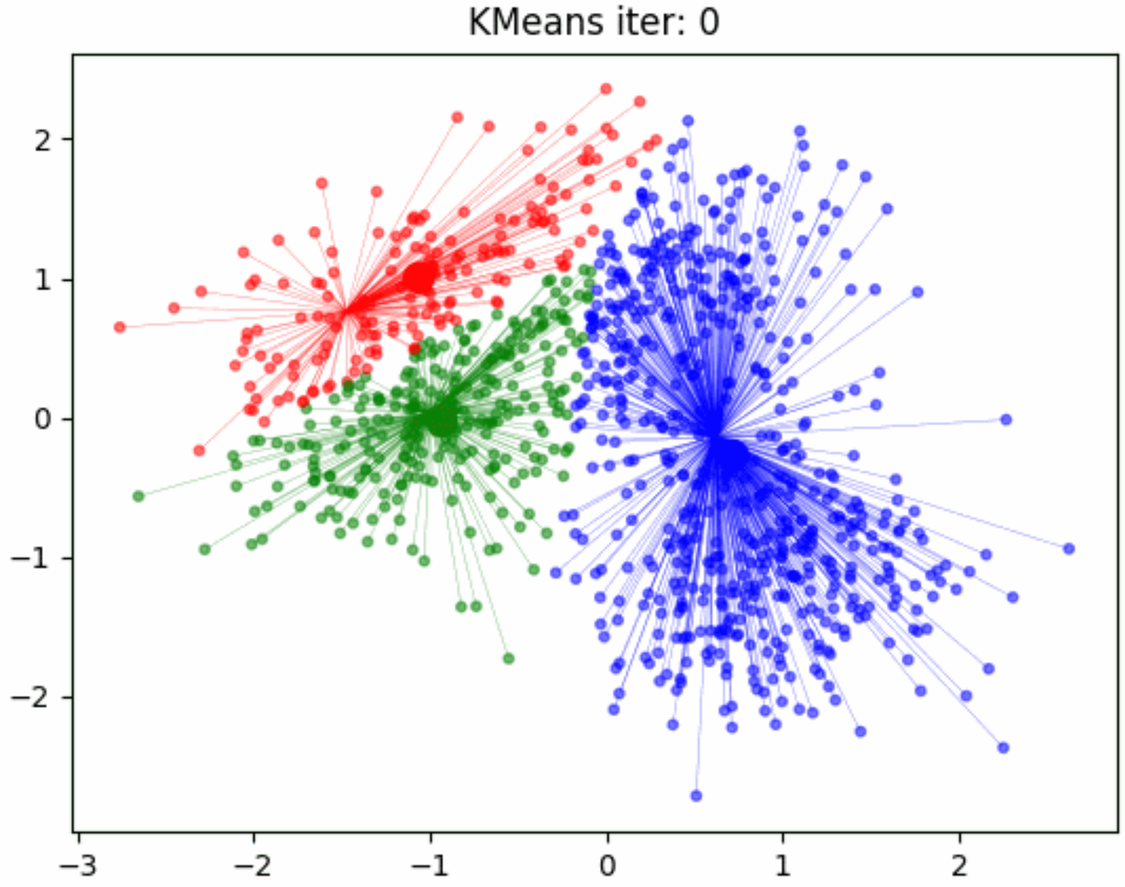




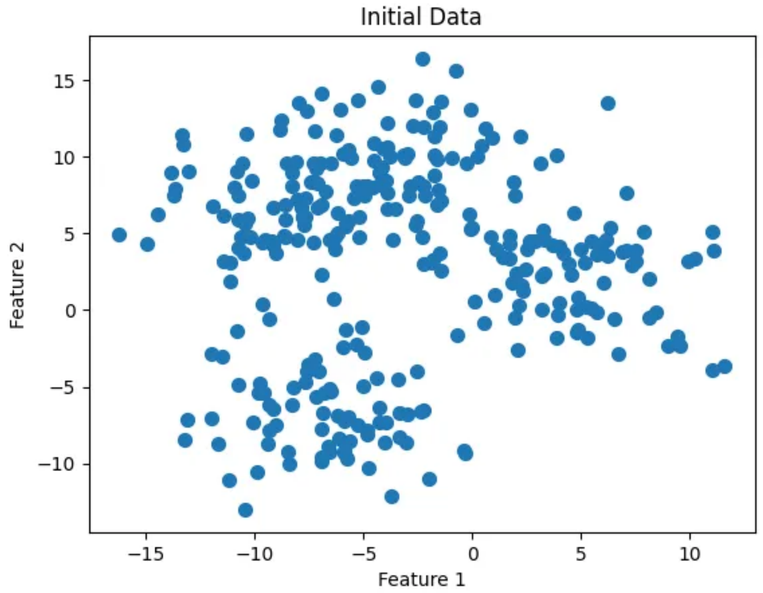
K-means (centroid) clustering

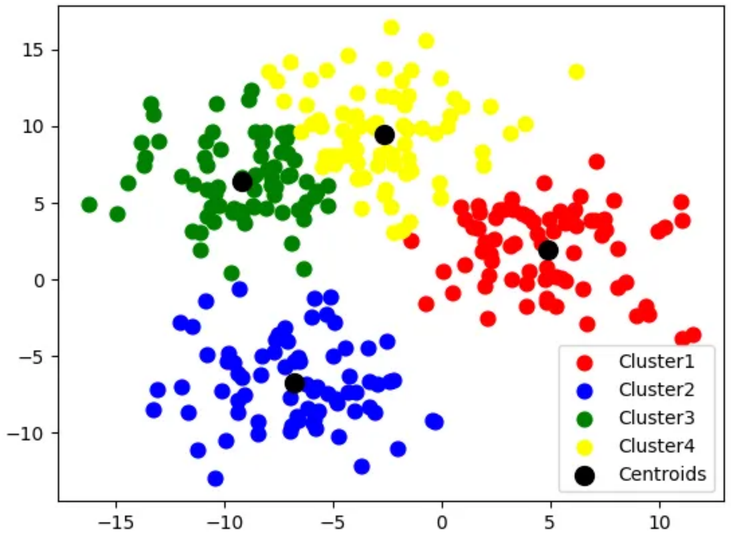


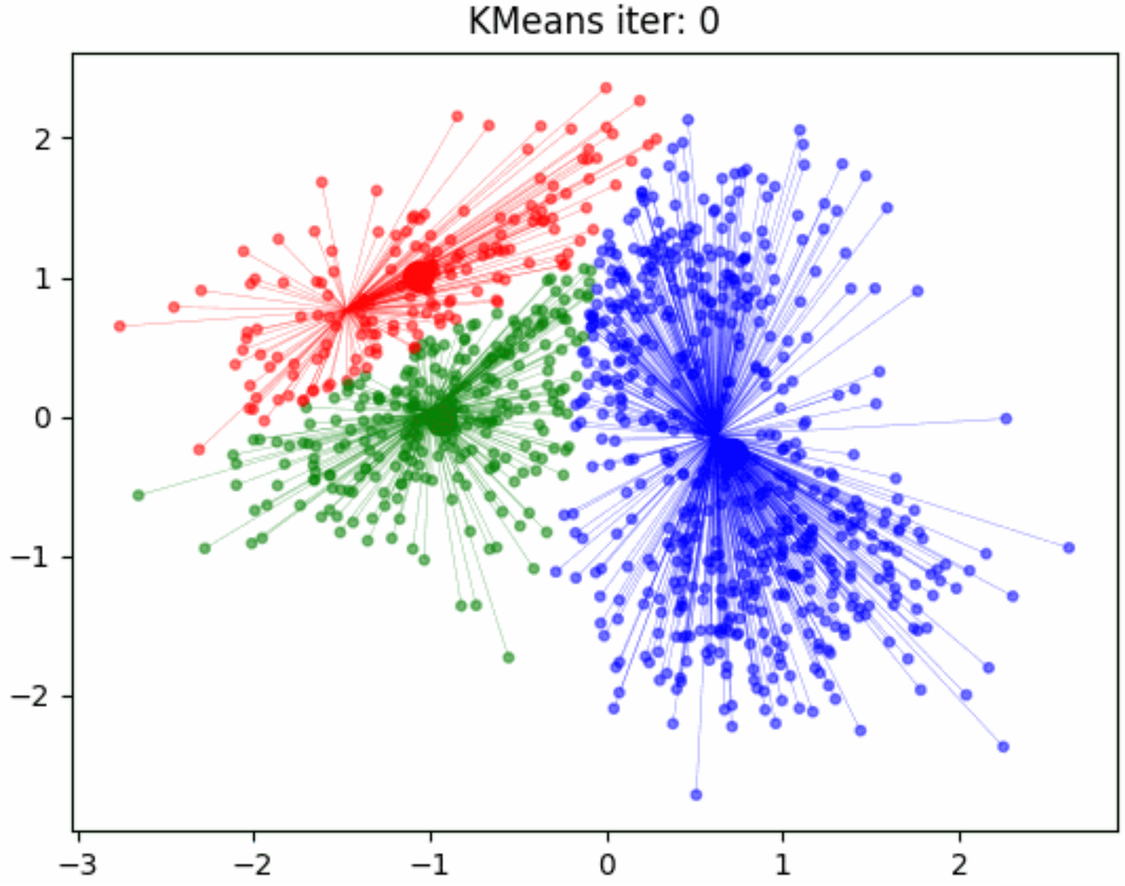




K-means (centroid) clustering

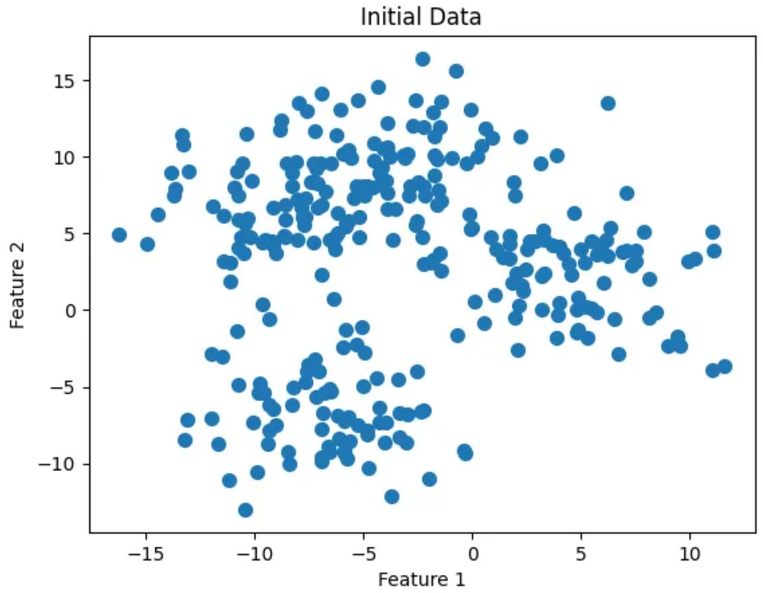


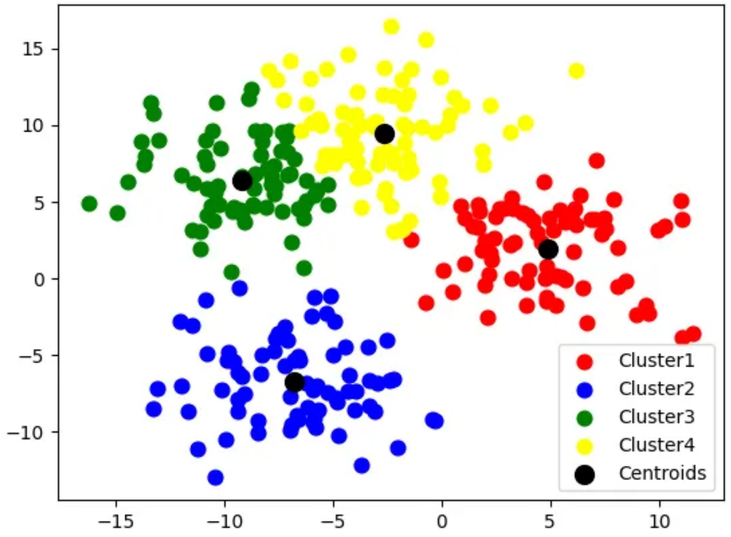


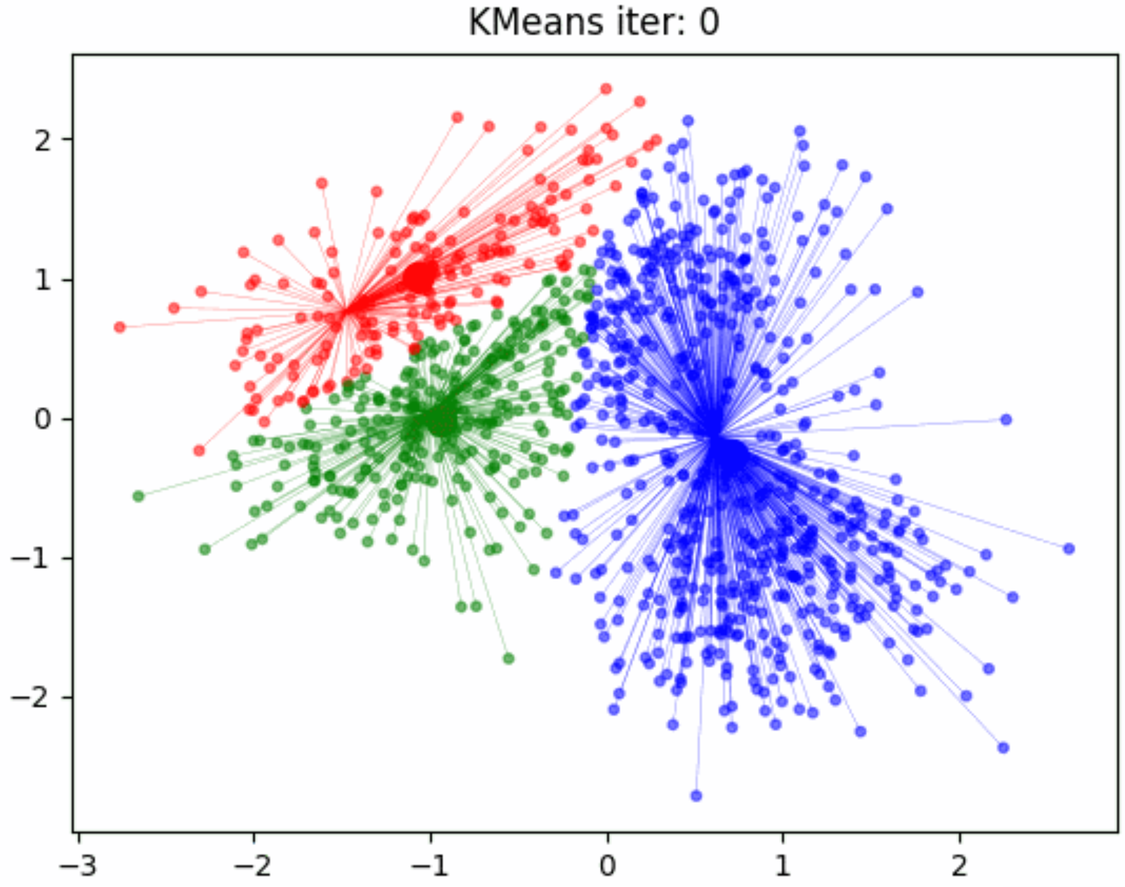


$$\underset{S=C_1\cup\cdots\cup C_k}{\operatorname{argmin}} \sum_k |C_k| \operatorname{var}(C_k)$$

K-means (centroid) clustering







- Useful **when centrality assumption is good and linear data set**

K-means compression

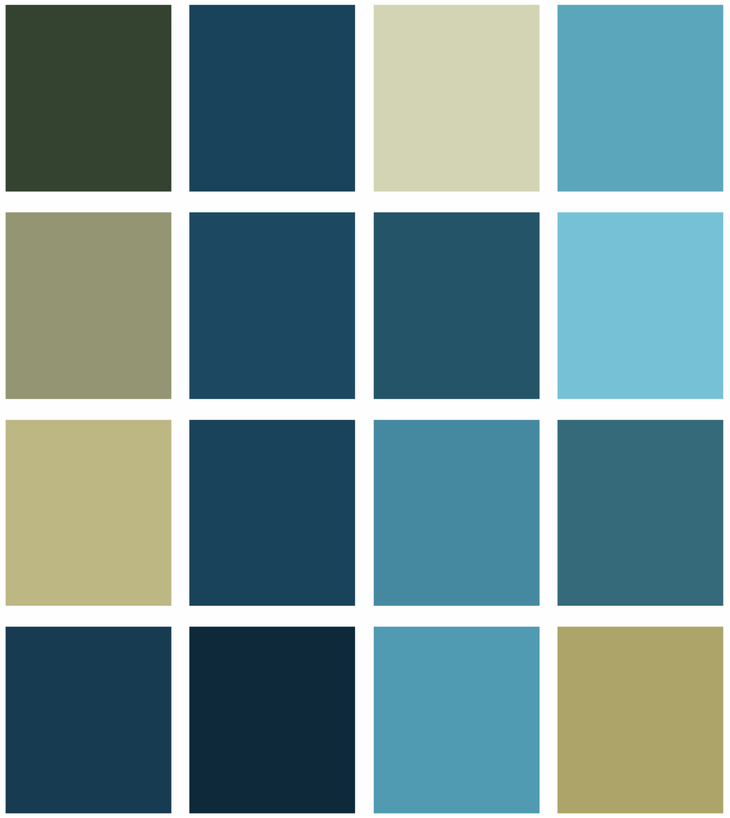
RGB: 0-255

For each pixel (256*3):
$2^8*2^8*2^8 = 2^{24} = 17M$

24 bits storage space

Use k-means to reduce to 16 colours
$\log_2(16) = 4$ bits storage space

Centroids



Iteration 1

Compressed Image

K-means compression

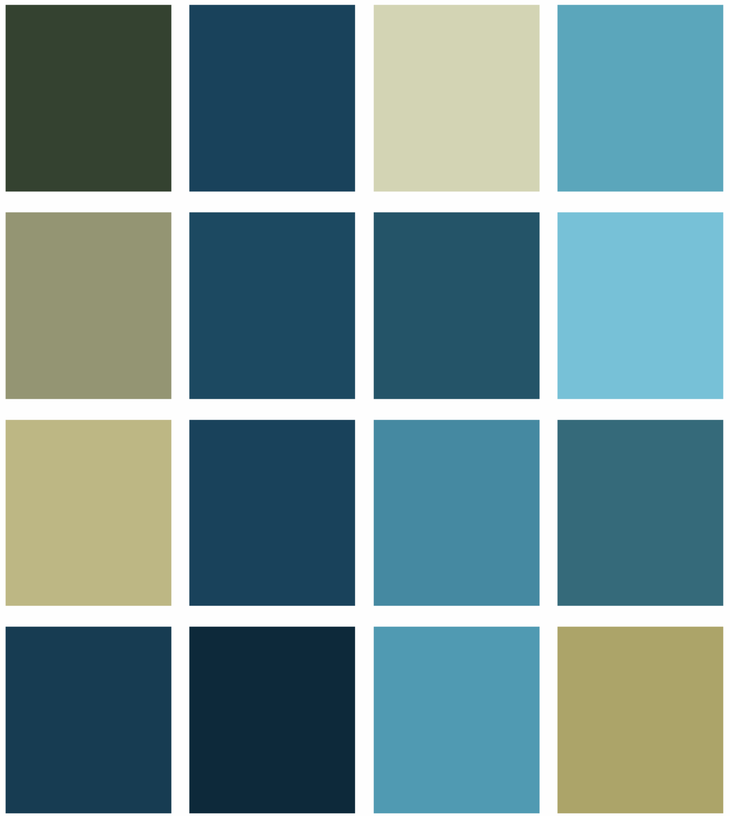
RGB: 0-255

For each pixel (256*3):
$2^8*2^8*2^8 = 2^{24} = 17M$

24 bits storage space

Use k-means to reduce to 16 colours
$\log_2(16) = 4$ bits storage space

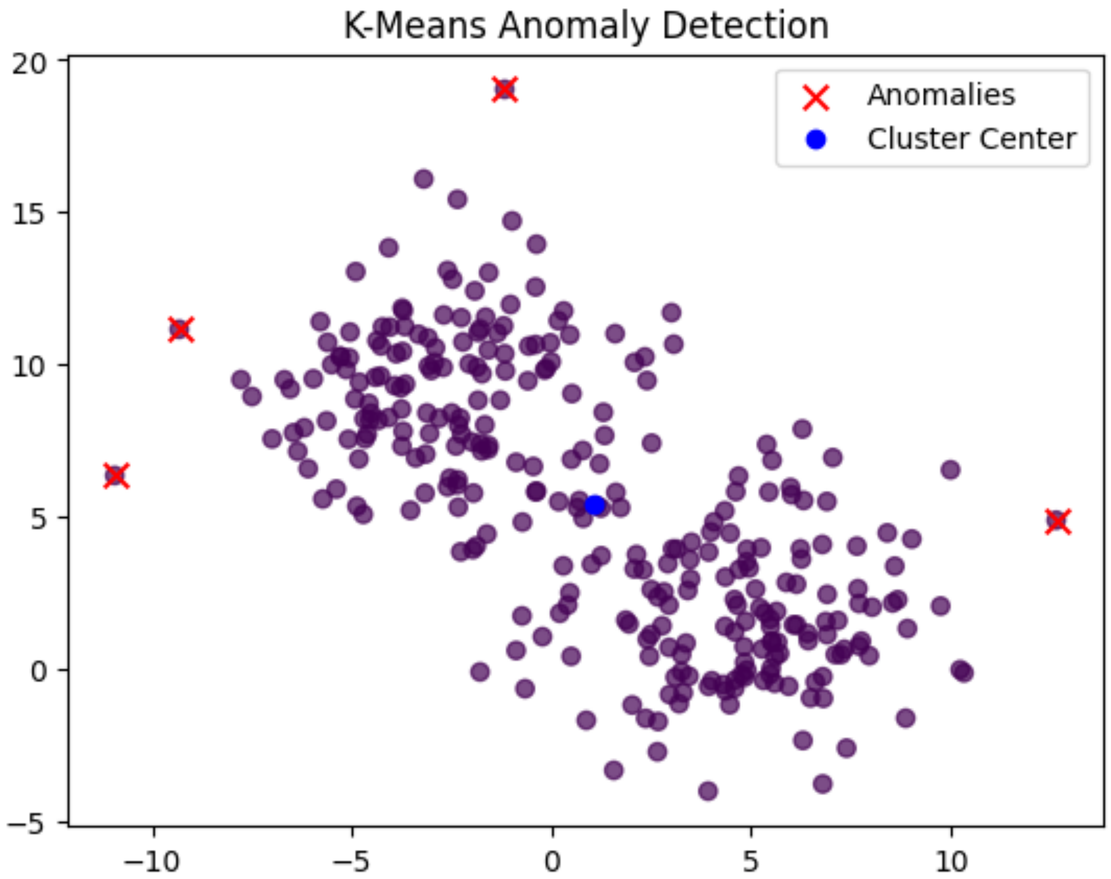
Centroids



Iteration 1

Compressed Image

Example: Anomaly Detection in Network Traffic with K-means clustering



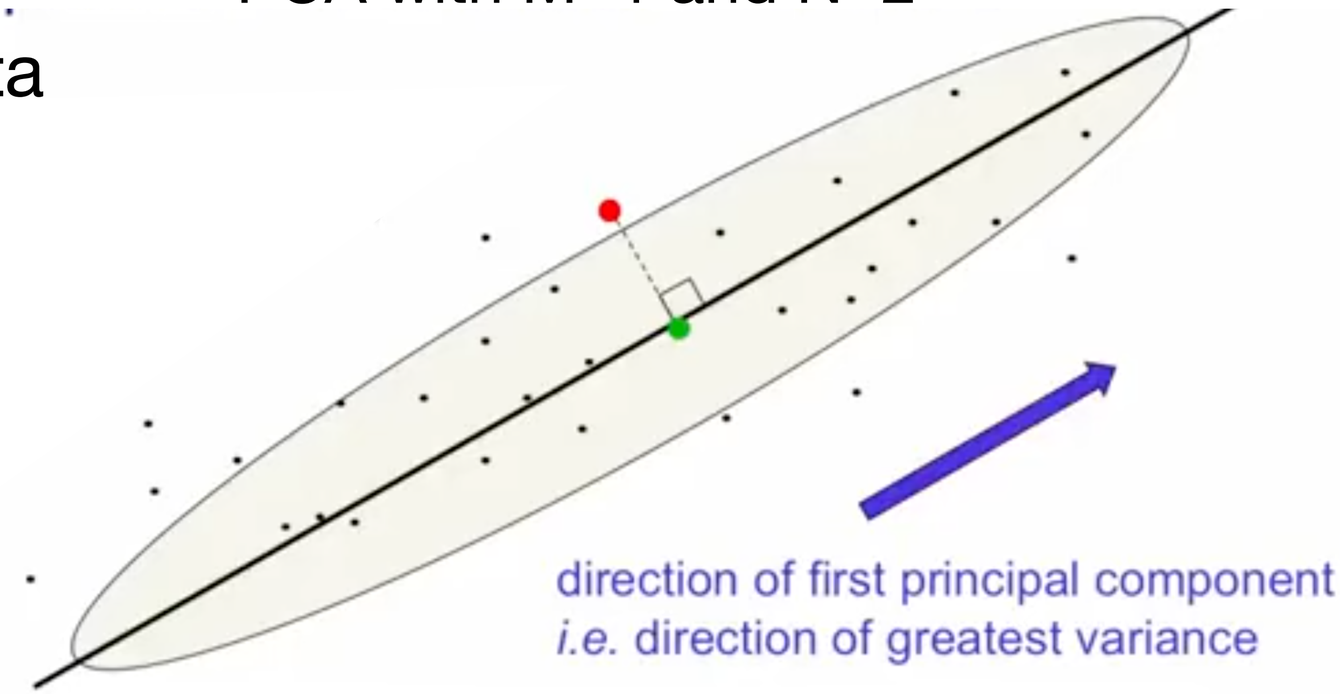
- k-means: summarize via clustering (cluster centers)

PCA (and connection to autoencoders)

PCA: For N-D data, find M orthogonal directions in which data has most variance

N-D datapoint represented by its projections onto M principal directions

PCA with M=1 and N=2



- PCA: summarize via orthogonal projections

See this video by Geoffrey Hinton!

PCA (and connection to autoencoders)

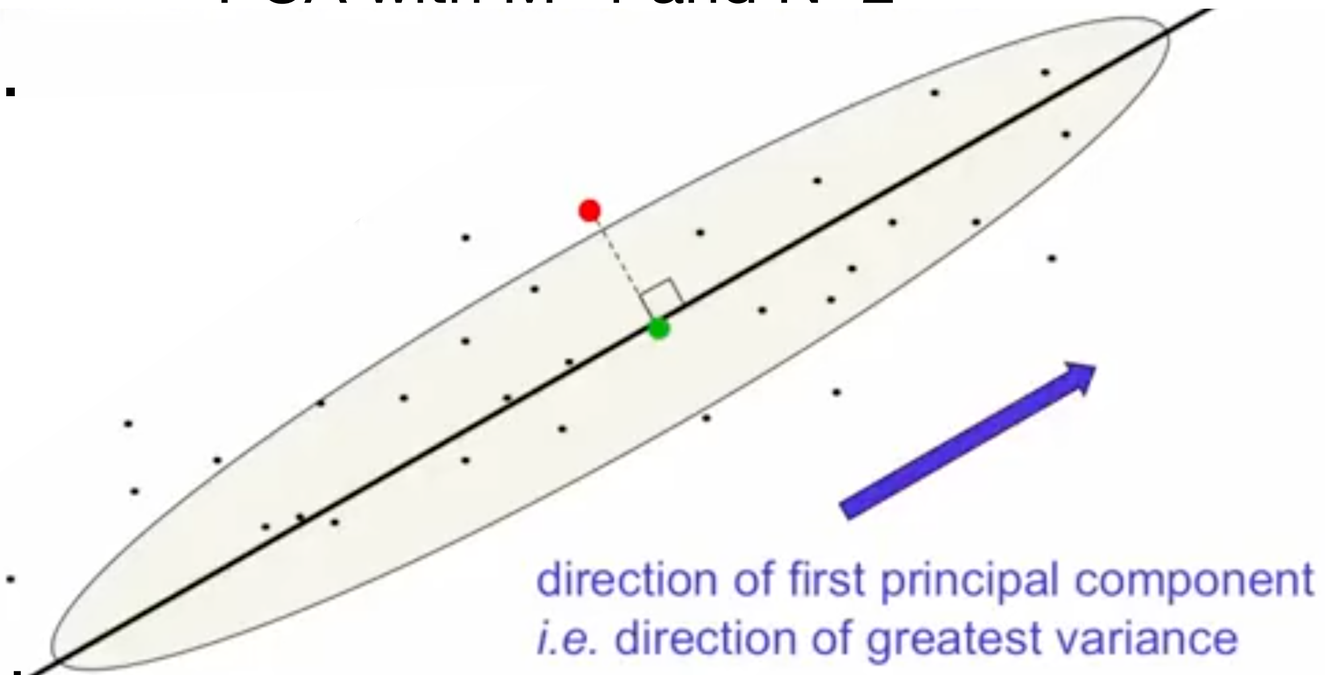
To reconstruct original data from M:
use mean value for all M-N
directions that are not represented.

Reconstruction error

$$J(\mathbf{x}, \tilde{\mathbf{x}}) = \frac{1}{N} \sum_{n=1}^{N} |\mathbf{x}_n - \tilde{\mathbf{x}}_n|^2$$

Minimise our loss by choosing to
ignore direction with least variance!

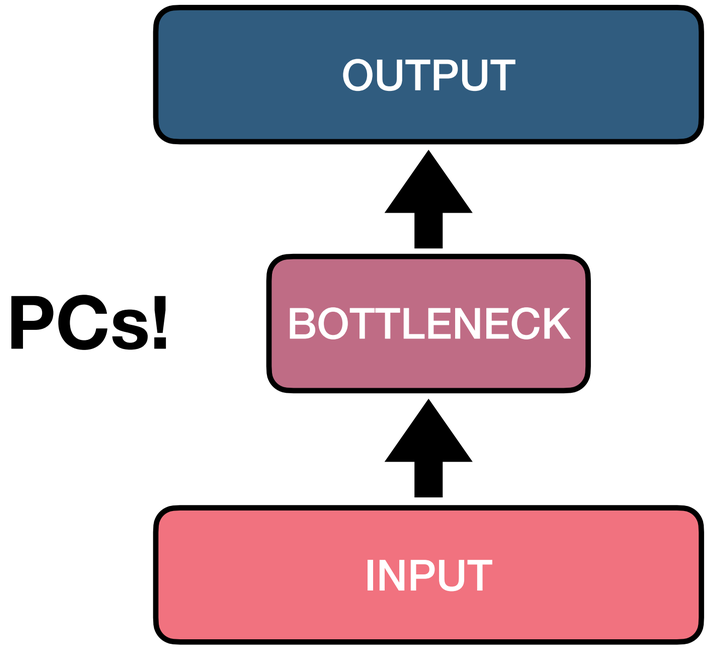
PCA with M=1 and N=2



See this video by Geoffrey Hinton!

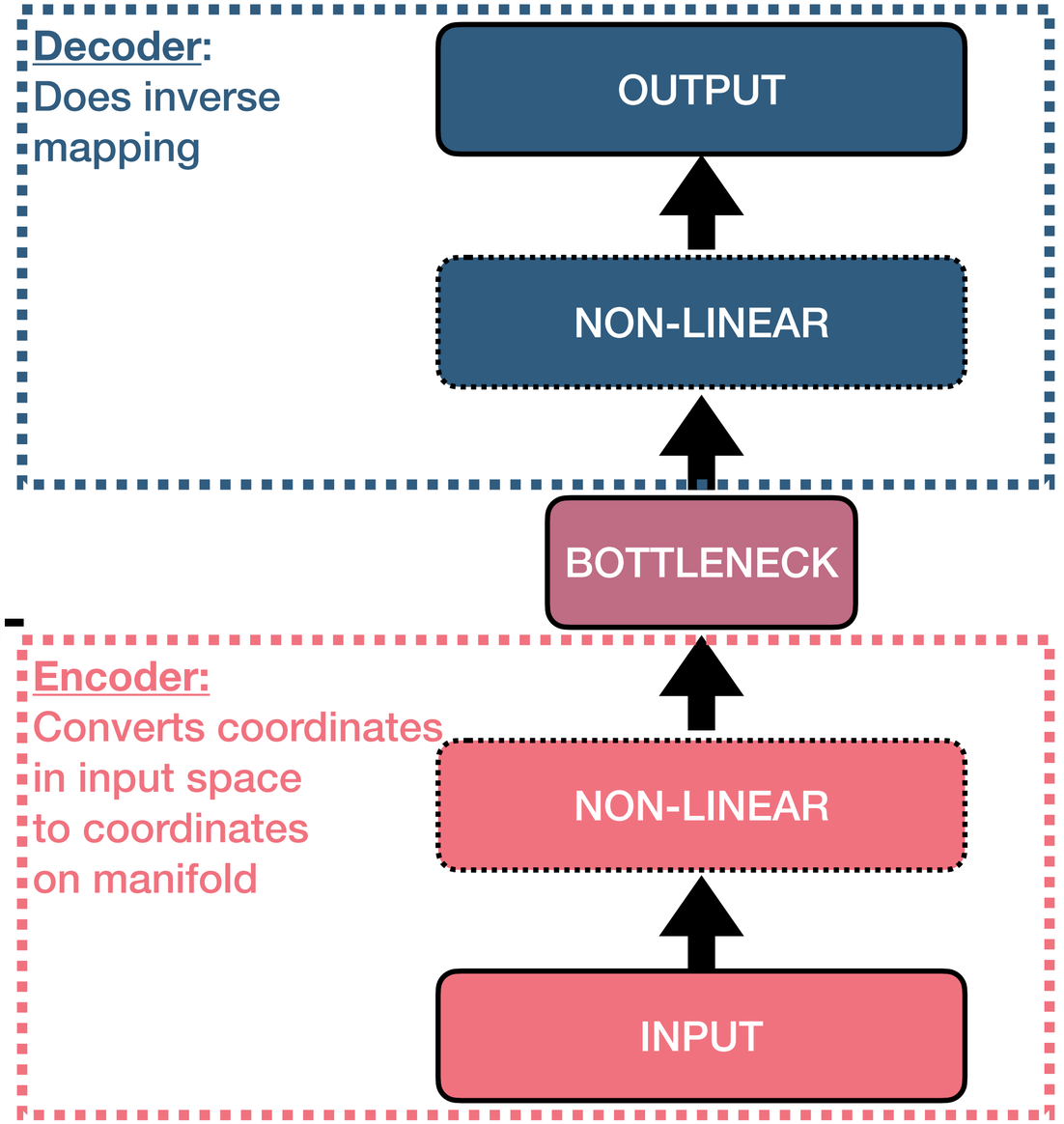
PCA with a neural network

- **Try to make output same as input in network with a bottleneck!**
- **Exactly what PCA does!**
- **M hidden units will span same space as first M components found by PCA (not same "weights", slightly rotation/skewing of axes)**
  - Weight vectors might not be orthogonal
  - Tend to have equal variance
  - But space spanned, is exactly same!
- **SGD way less efficient, why do it with NNs?**



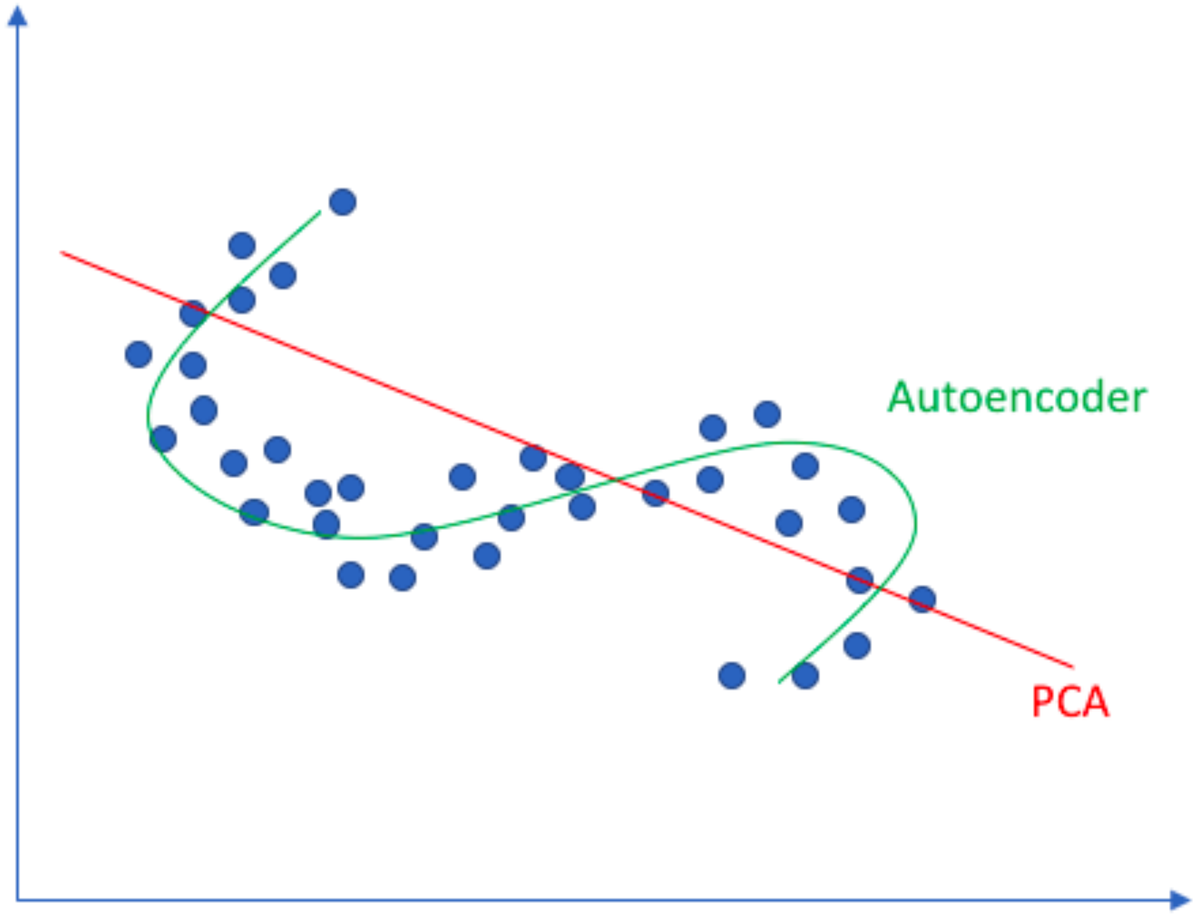
Autoencoders vs. PCA?

- **With non-linear layers before and after bottleneck, possible to efficiently represent data that lies on near/non-linear manifold!**
  - **Much more general (from linear to non-linear function of input!)**
  - **Supervised learning algorithm, doing unsupervised learning!**



See this video by Geoffrey Hinton!

Linear vs non-linear dimensionality reduction



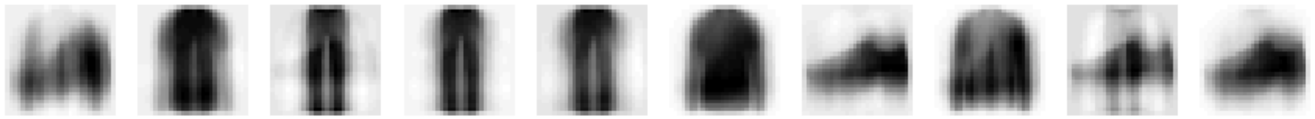
Linear vs non-linear dimensionality reduction

```
m = Sequential()
m.add(Dense(1000, activation='relu',
            input_shape=(784,)))
m.add(Dense(500, activation='relu'))
m.add(Dense(250, activation='relu'))
m.add(Dense(32, activation='relu'))
m.add(Dense(2, activation='linear',
            name="bottleneck"))
m.add(Dense(32, activation='relu'))
m.add(Dense(250, activation='relu'))
m.add(Dense(500, activation='relu'))
m.add(Dense(1000, activation='relu'))
m.add(Dense(784, activation='sigmoid'))
```

Original

Autoencoder

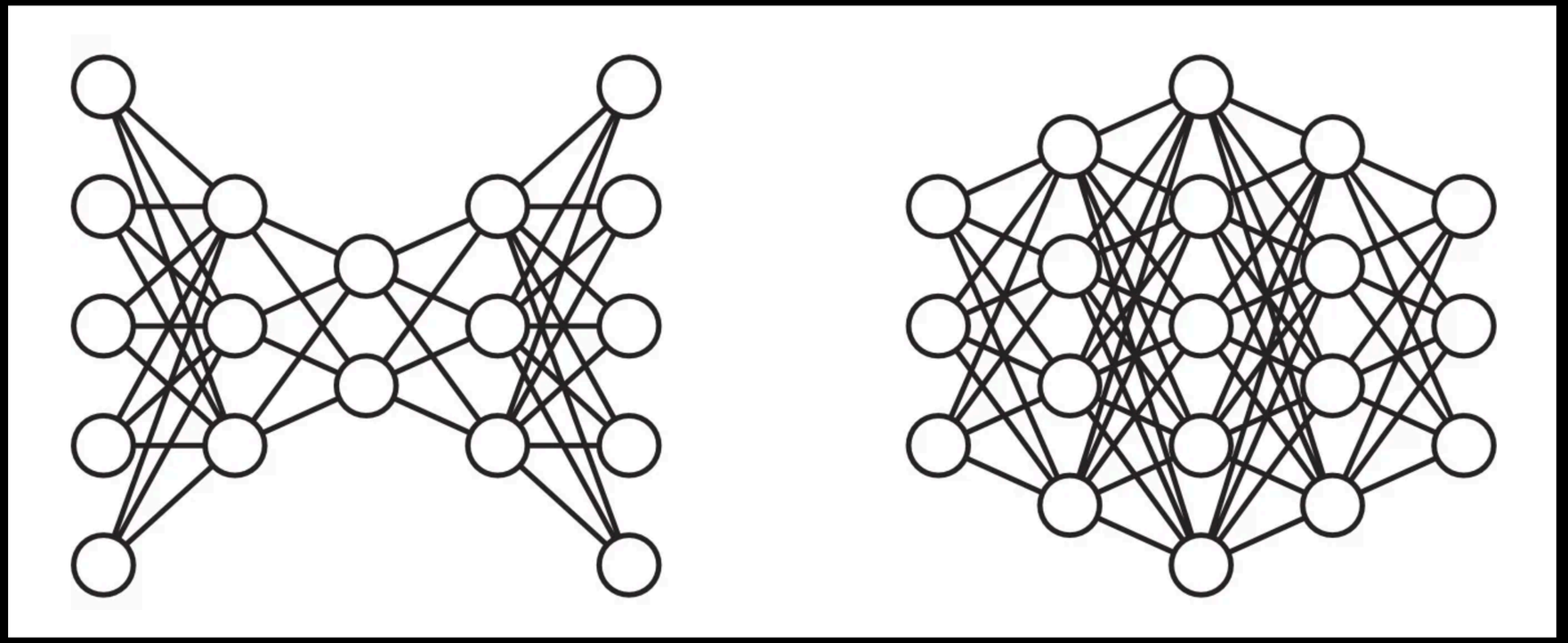
PCA



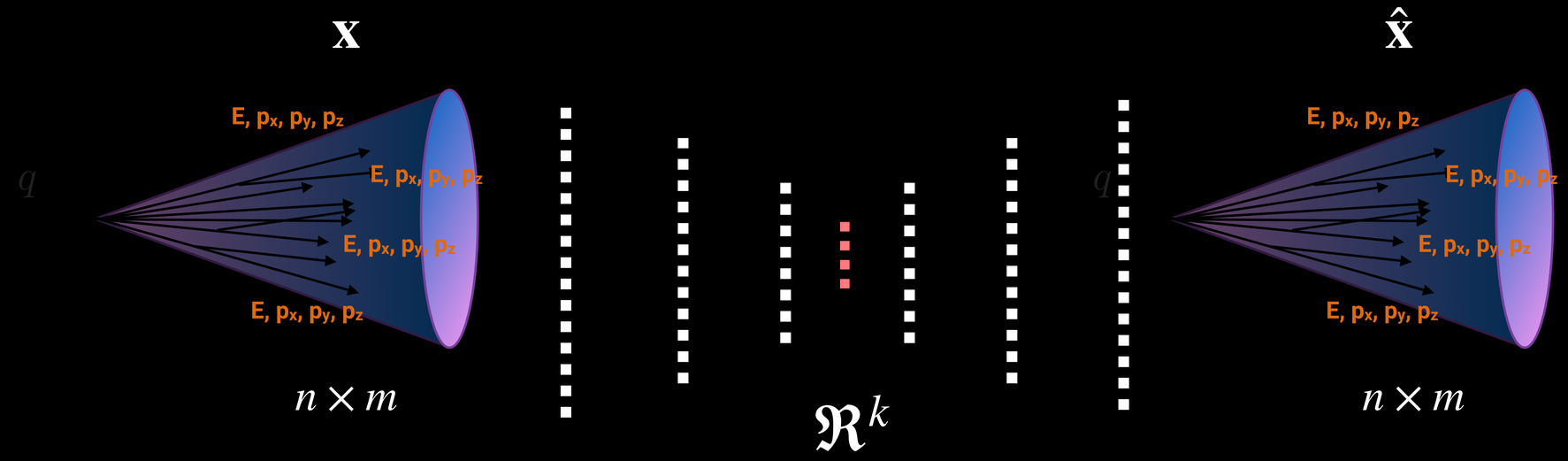
Under-complete verus over-complete?

Autoencoders usually undercomplete (bottleneck that has compression)

Overcomplete also possible, typically with some reguarlization (such as L1 or sparsity conditions)

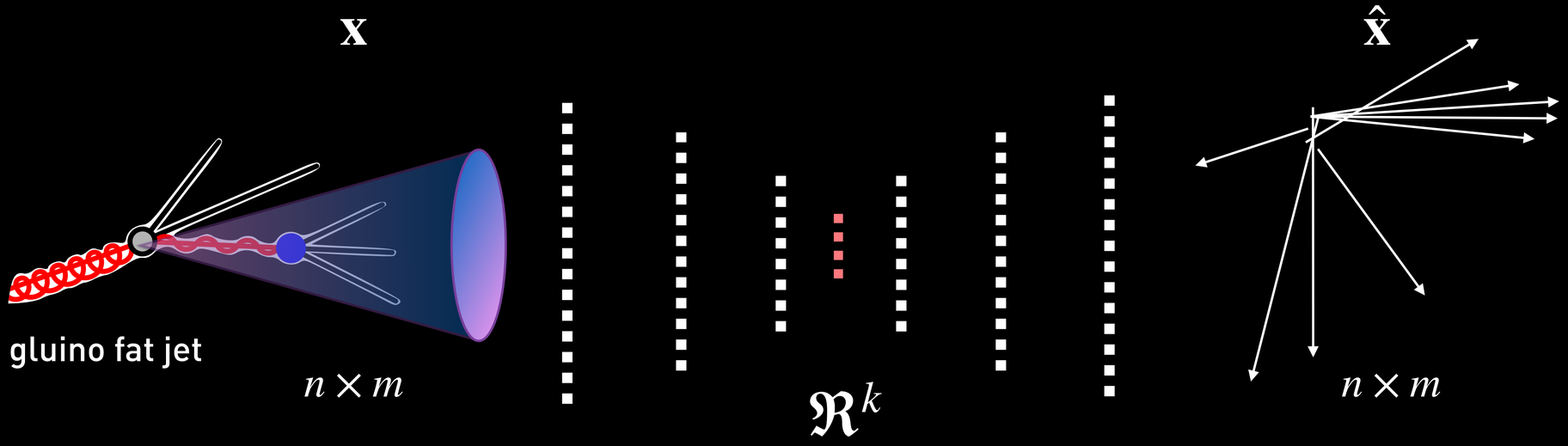


Outlier detection



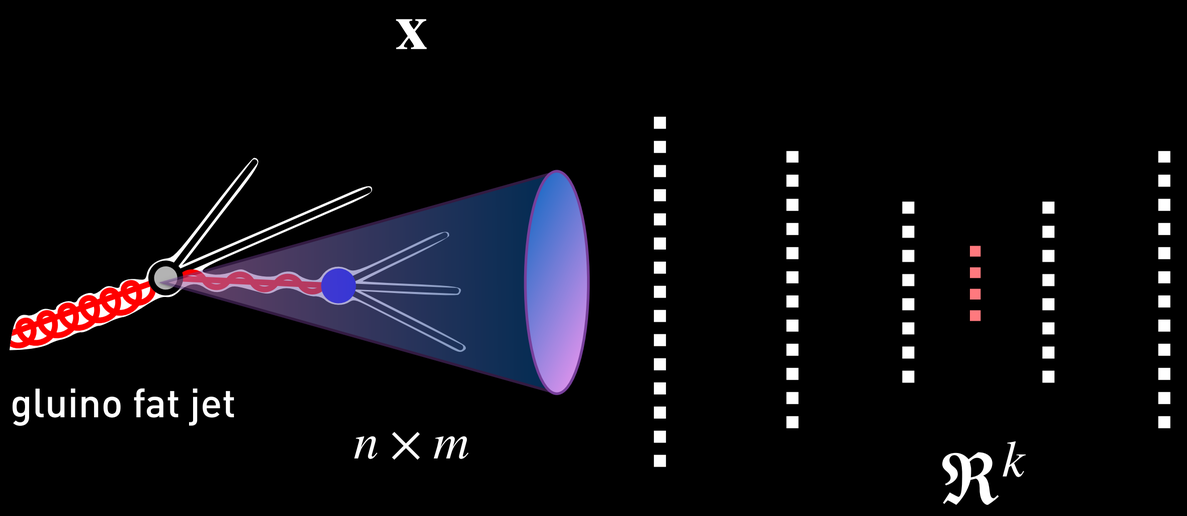
Compressed representation of x.
Latent space $\Re^k$, k < m×n
prevents memorisation of input, must learn

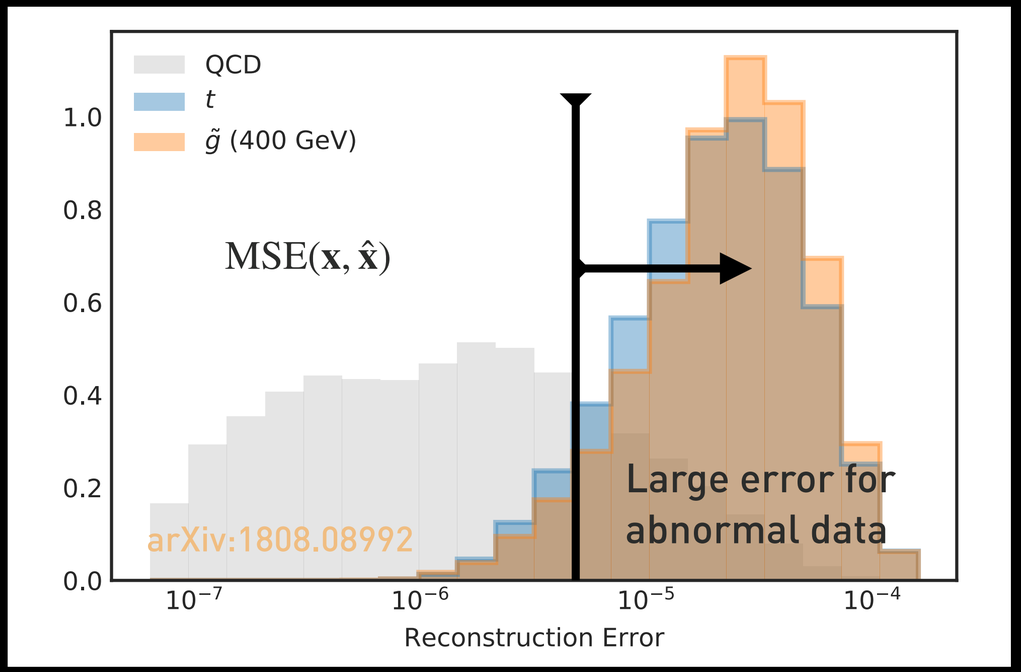
Outlier detection



$\mathscr{L}(\mathbf{x}, \hat{\mathbf{x}})$ **is Mean Squared Error**$(\mathbf{x}, \hat{\mathbf{x}})$**, "high error events" proxy for "degree of abnormality"**

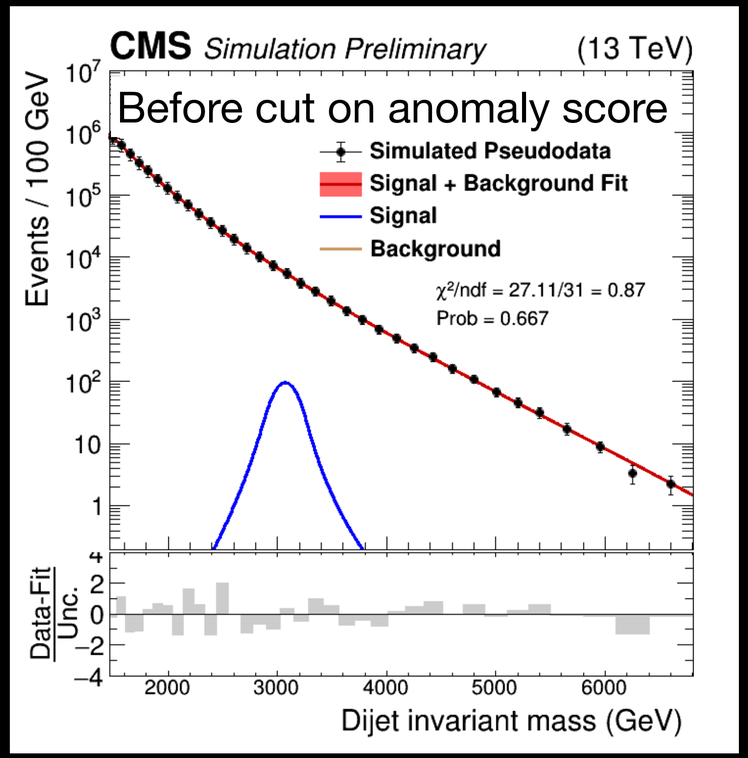
Outlier detection

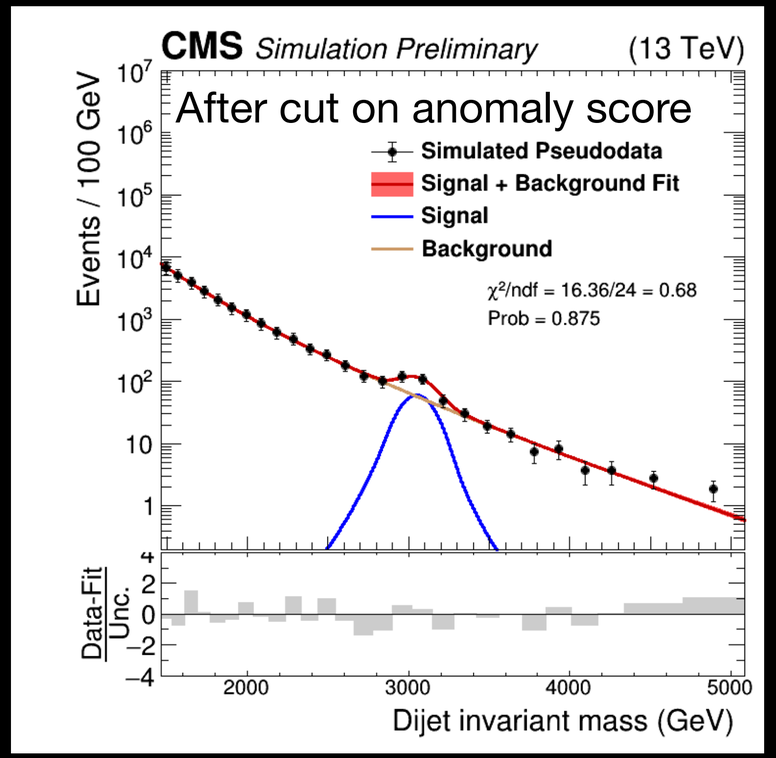




Outlier detection in analysis

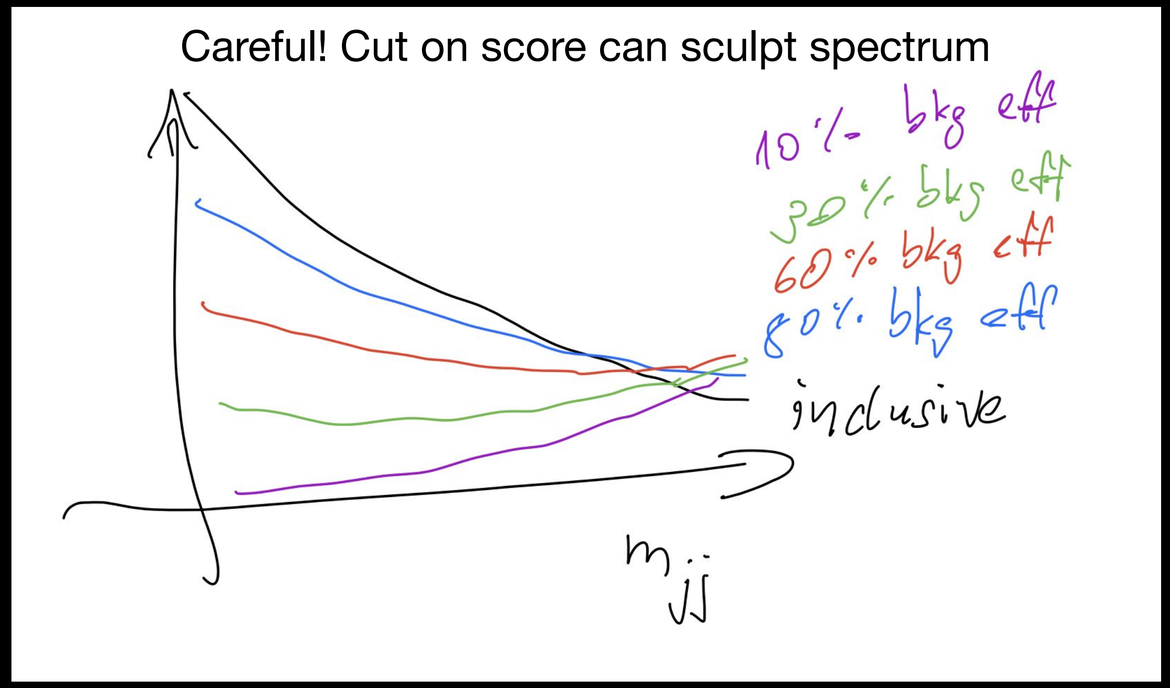
E.g CASE

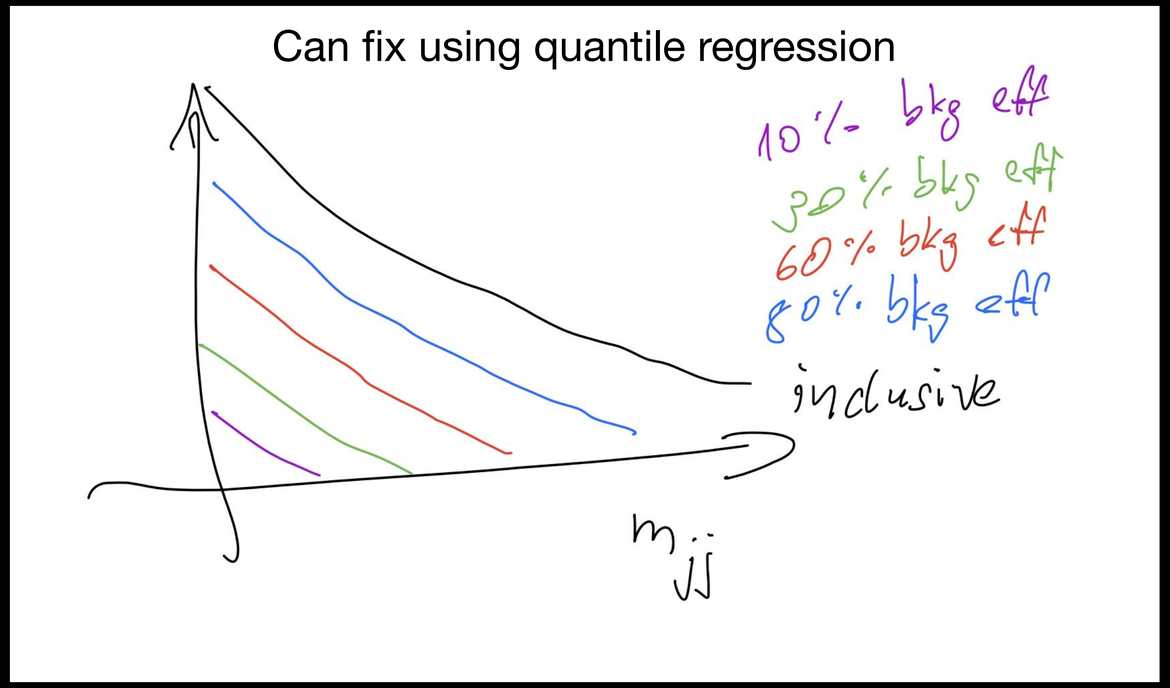




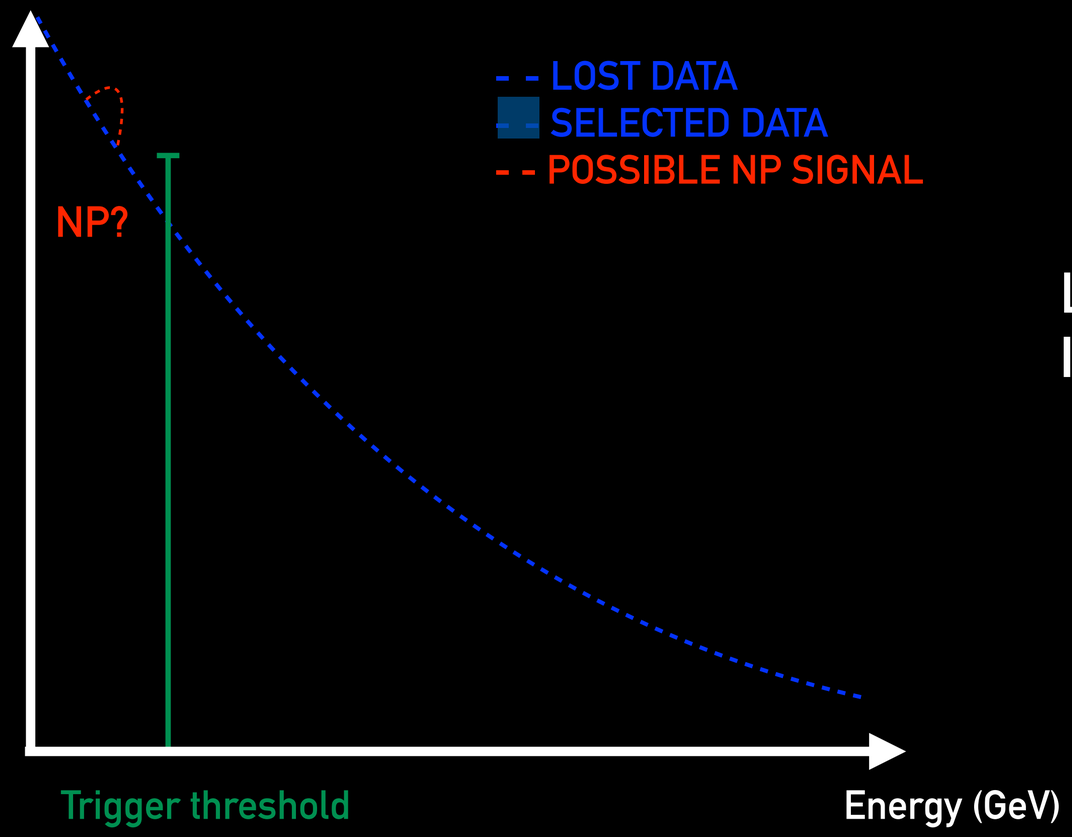
Outlier detection in analysis

E.g CASE



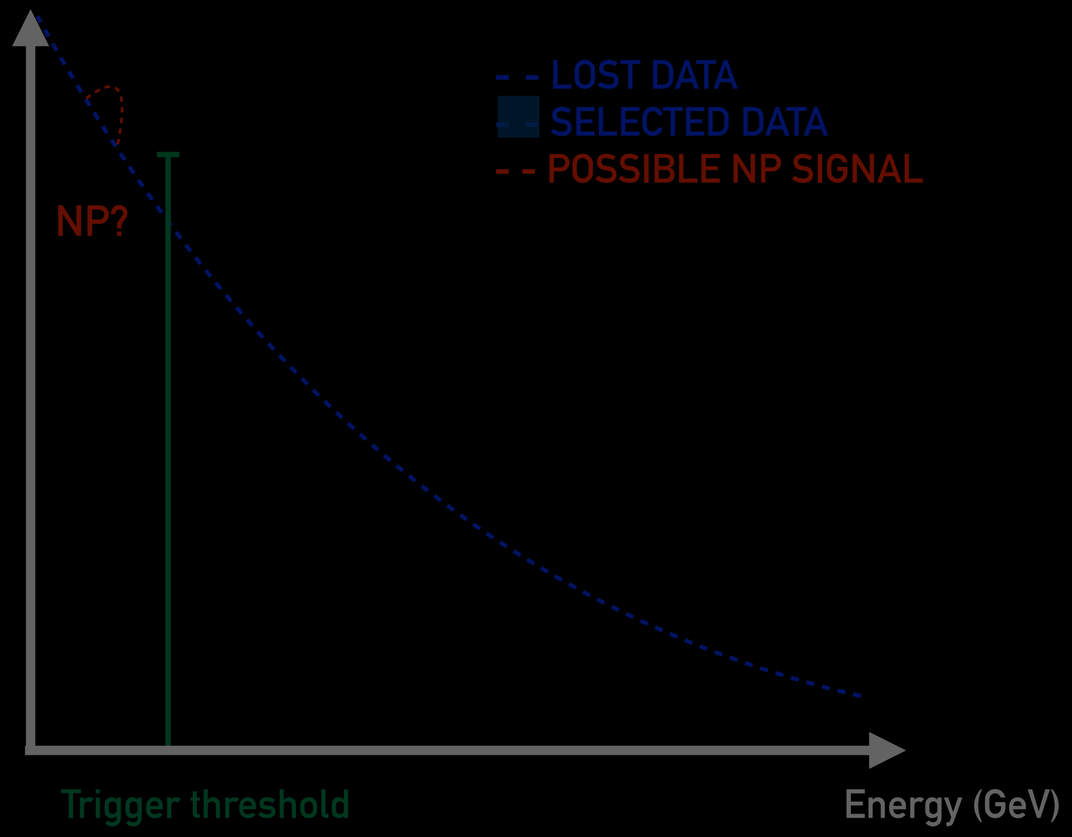


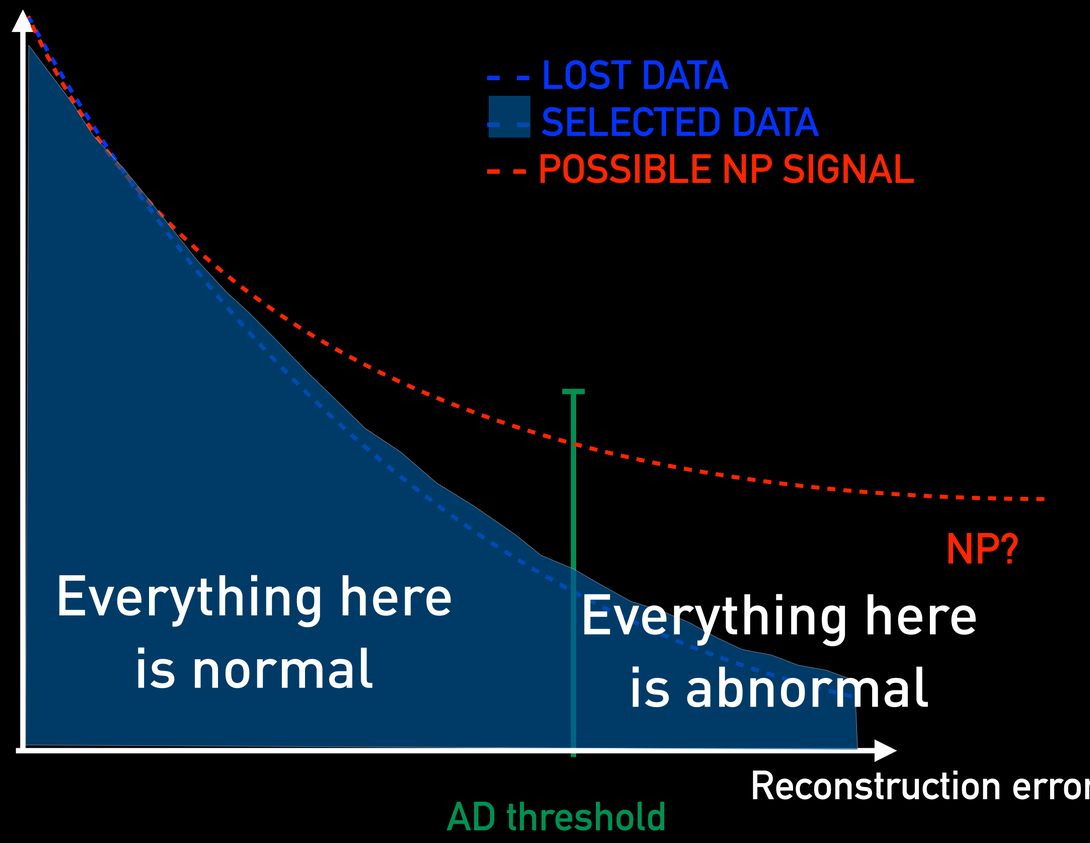
Anomaly Detection triggers



LHC “triggers” >99% of events (next lecture)!
Is there a smarter way to select?

Anomaly Detection triggers

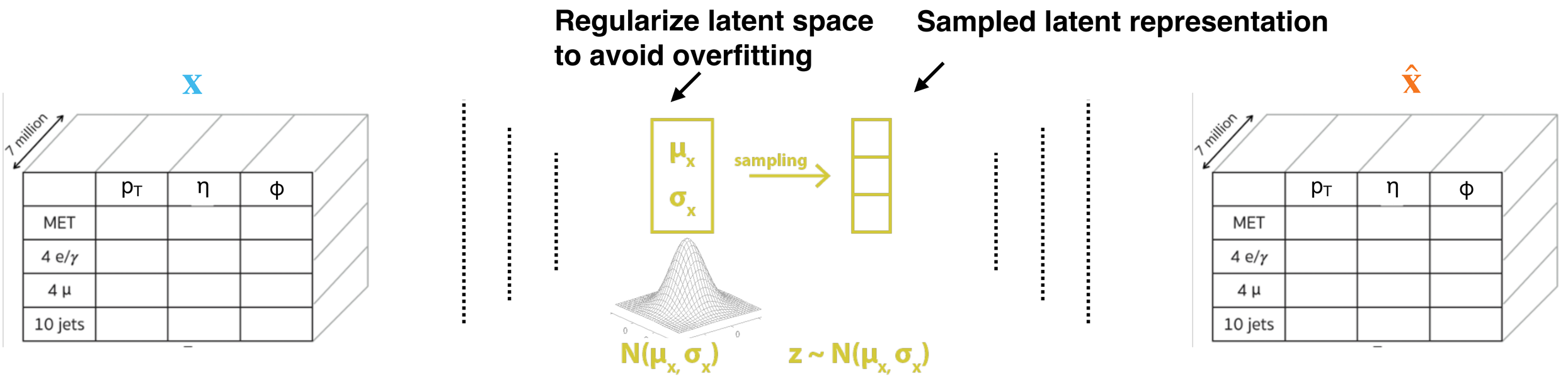




AXOL1TL

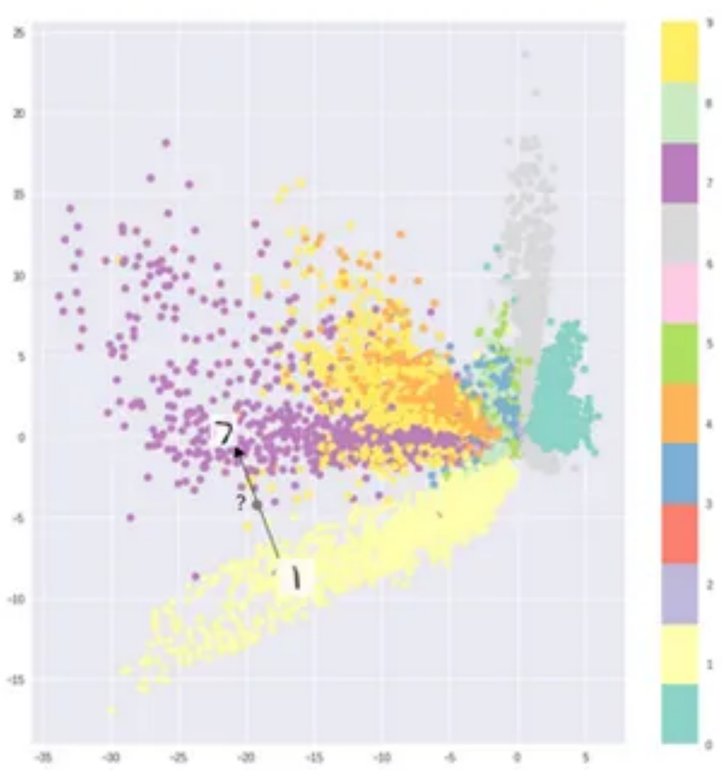
X

AXOL1TL

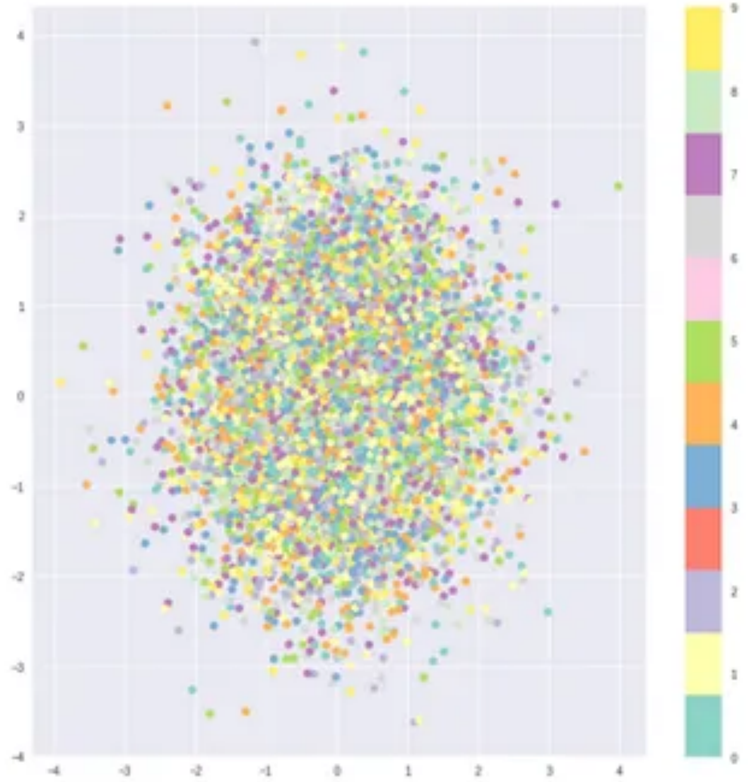


$$\text{loss} = ||x - \hat{x}||^2 + KL[N(\mu_x, \sigma_x), N(0, I)]$$

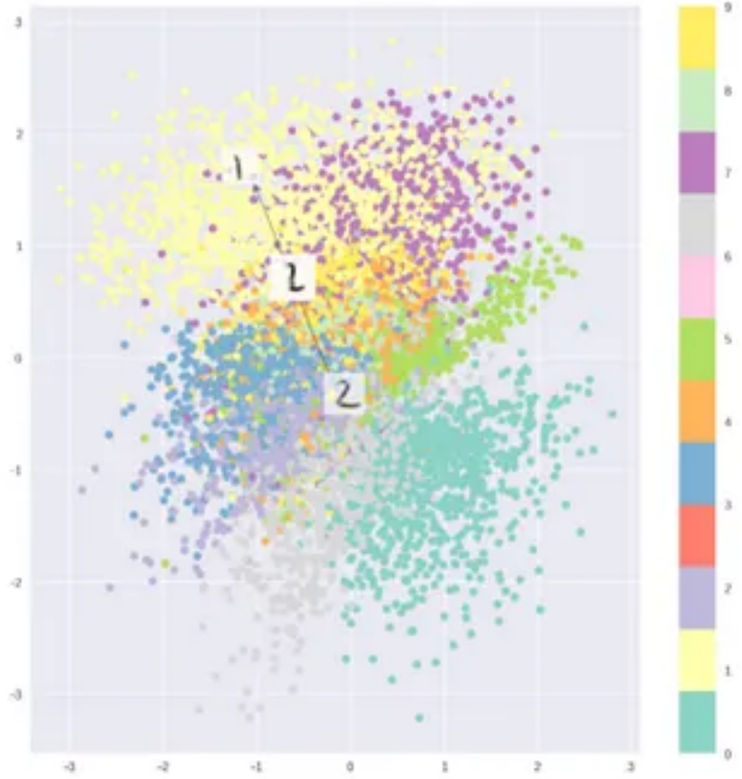
Only reconstruction loss



Only KL divergence



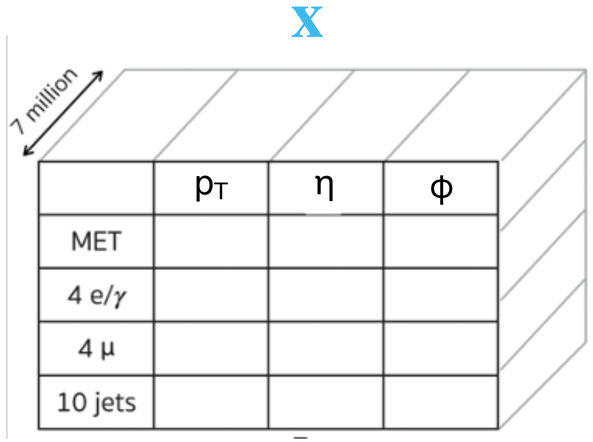
Combination

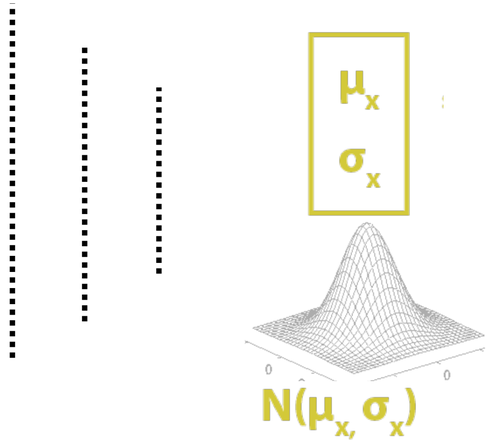


Push latent space
towards normality!

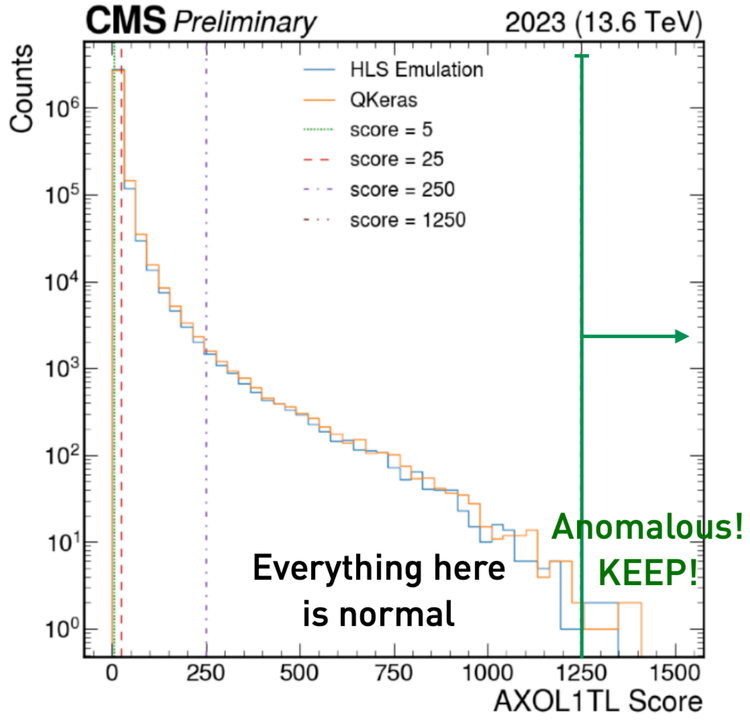
AXOL1TL

In 50 nanoseconds!!!

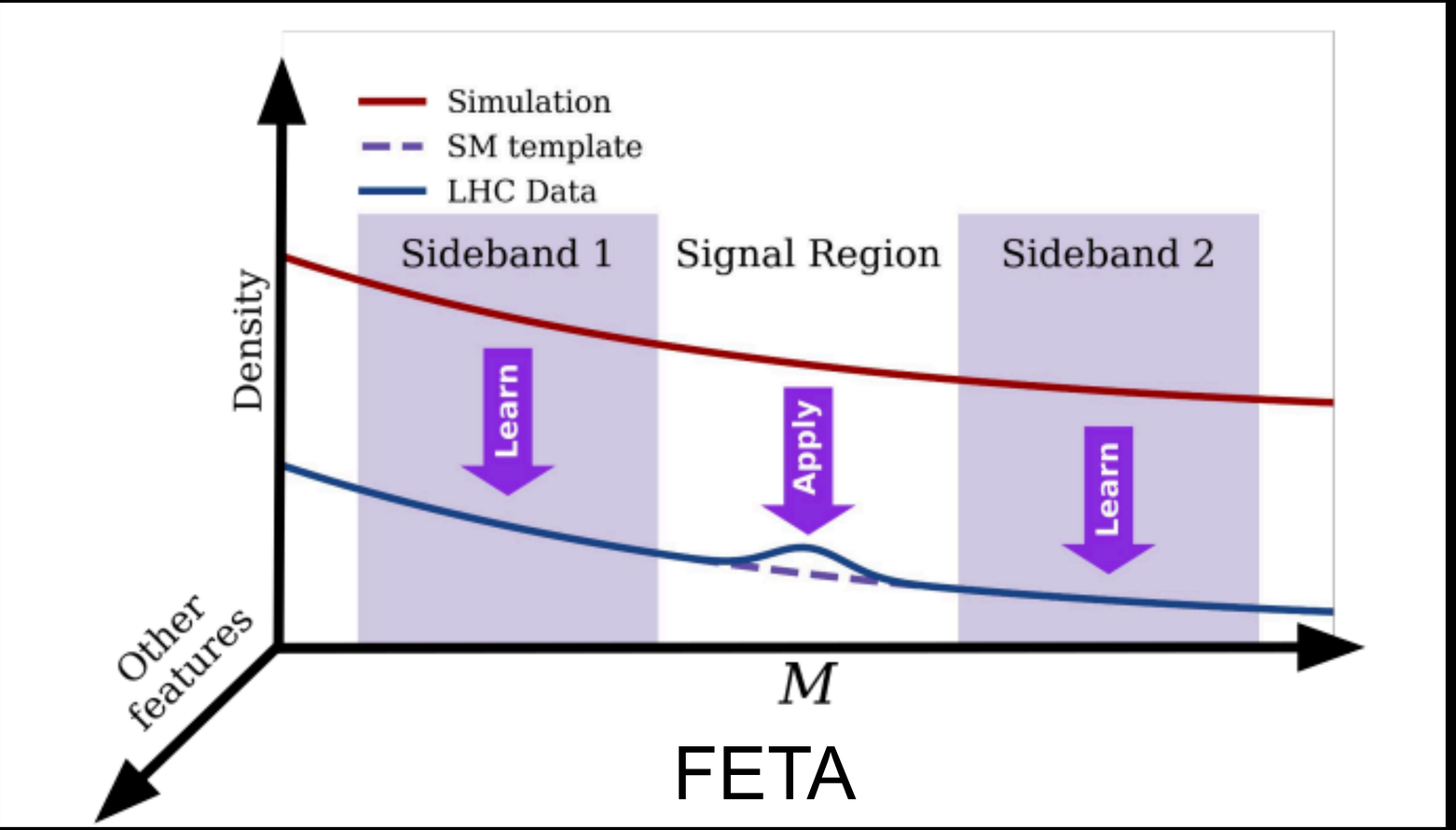




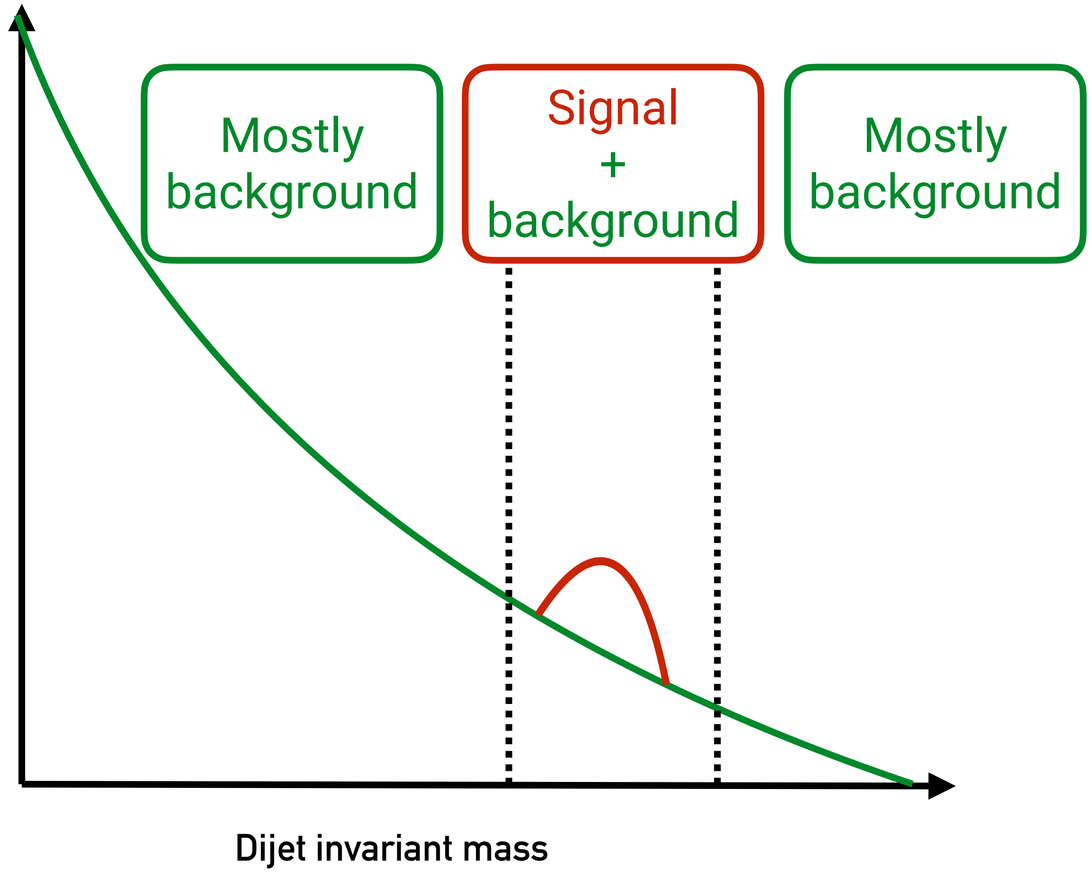
$$\text{loss} = ||\, x - \hat{x} \,||^2 + \mathrm{KL}[\, N(\mu_x, \sigma_x), N(0, I)\,]$$



Finding overdensities



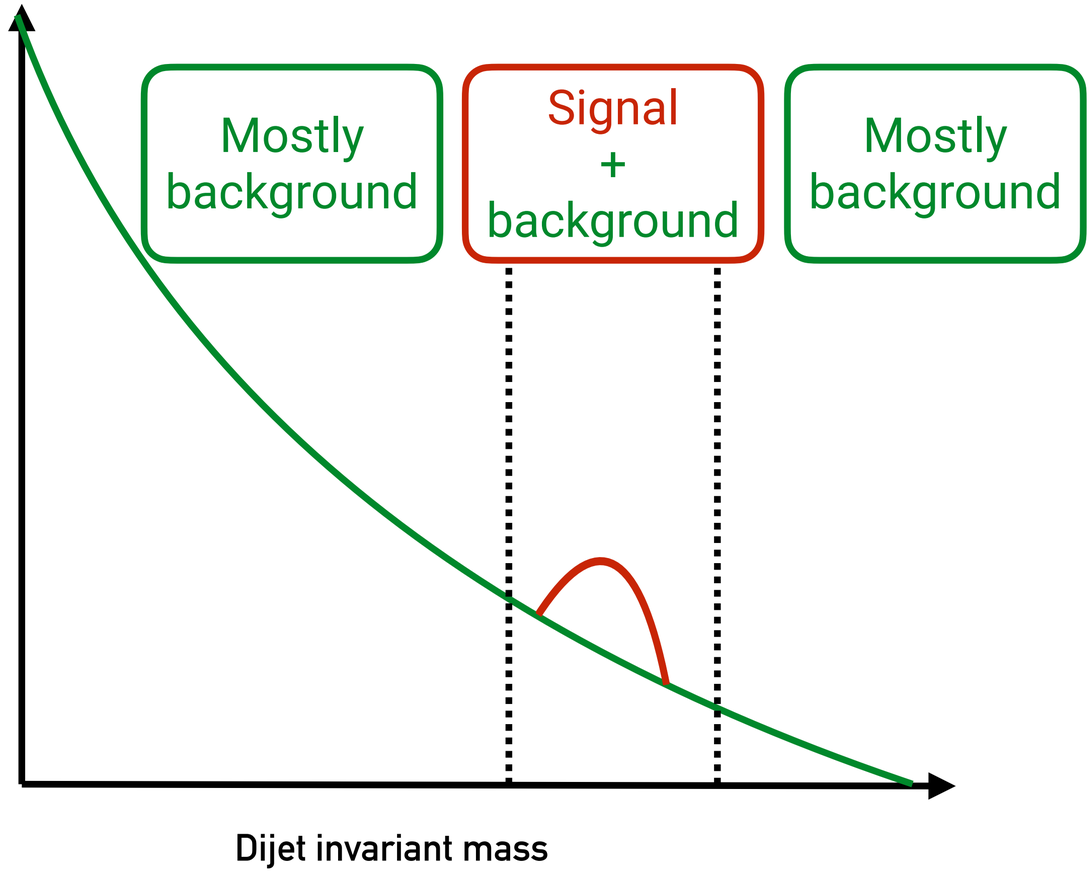
Weak classification without labels (CWoLa)



Weak classification without labels (CWoLa)

Classification Without Labels

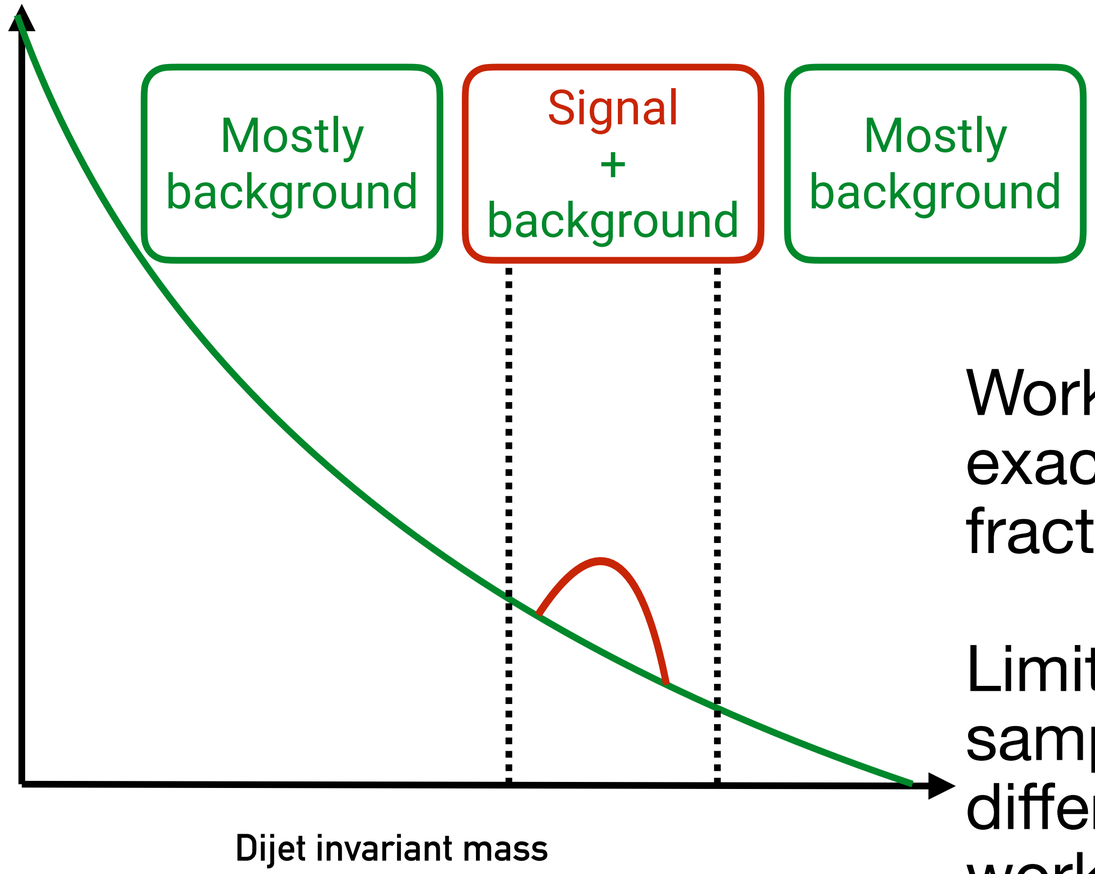
- Lemma: “Given mixed S+B samples SB and SR, optimal classifier trained to distinguish SB and SR is also optimal for distinguishing S from B”



Weak classification without labels (CWoLa)

Classification Without Labels

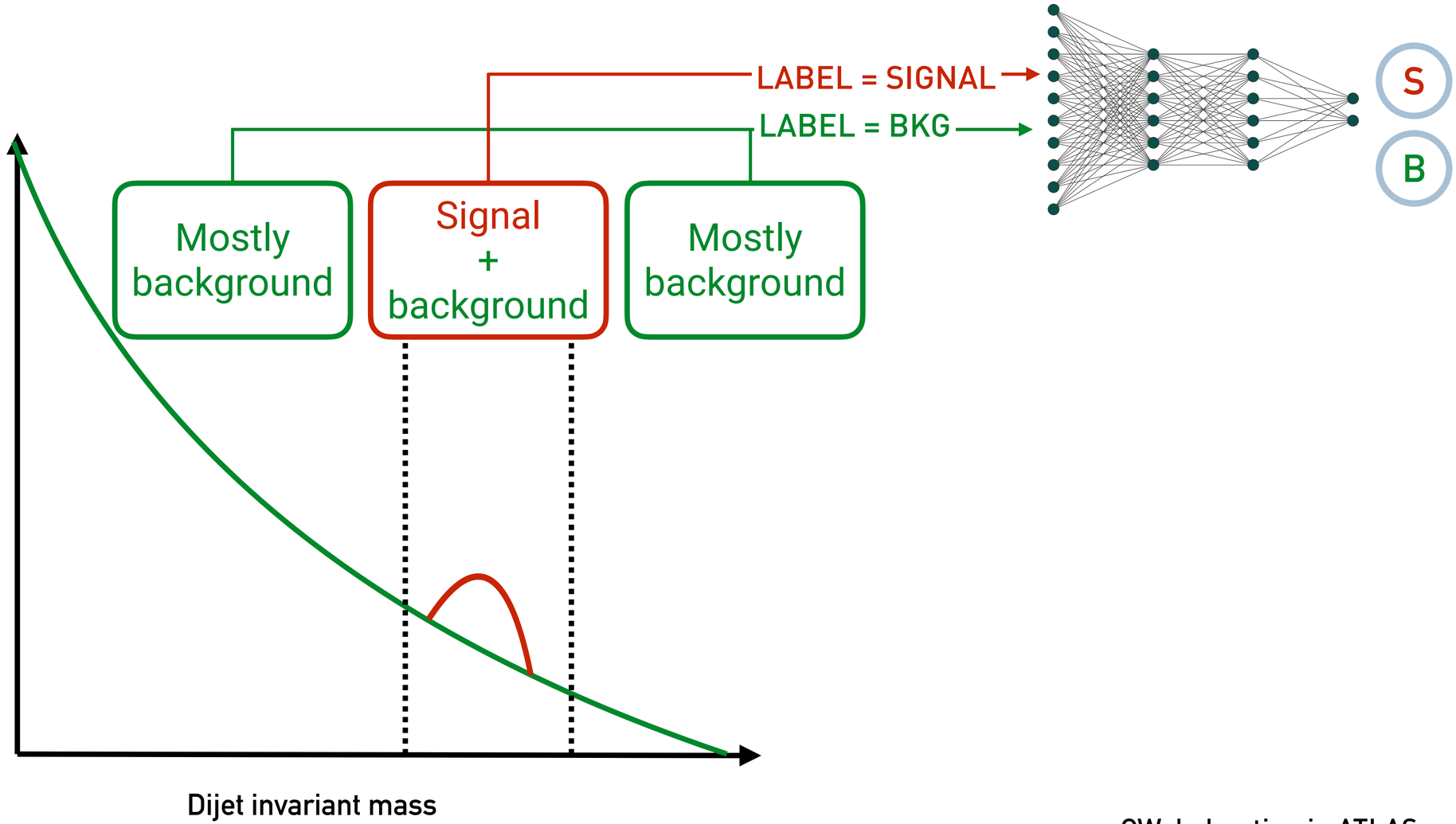
- Lemma: “Given mixed S+B samples SB and SR, optimal classifier trained to distinguish SB and SR is also optimal for distinguishing S from B”



Works even if you don’t exactly know the S/B fractions!

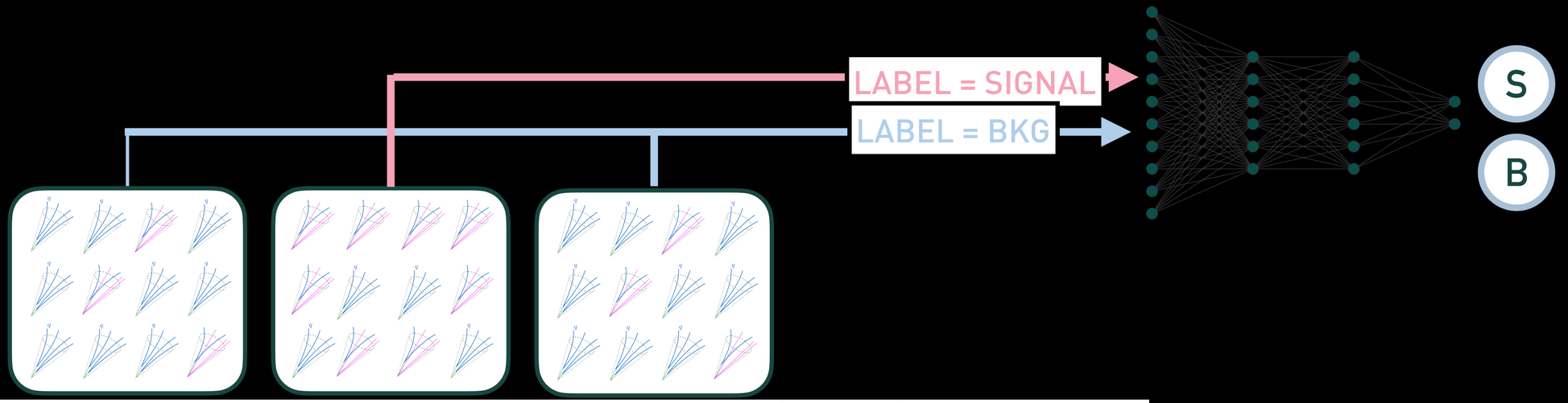
Limitation: when the two samples don’t have different fractions, doesn’t work

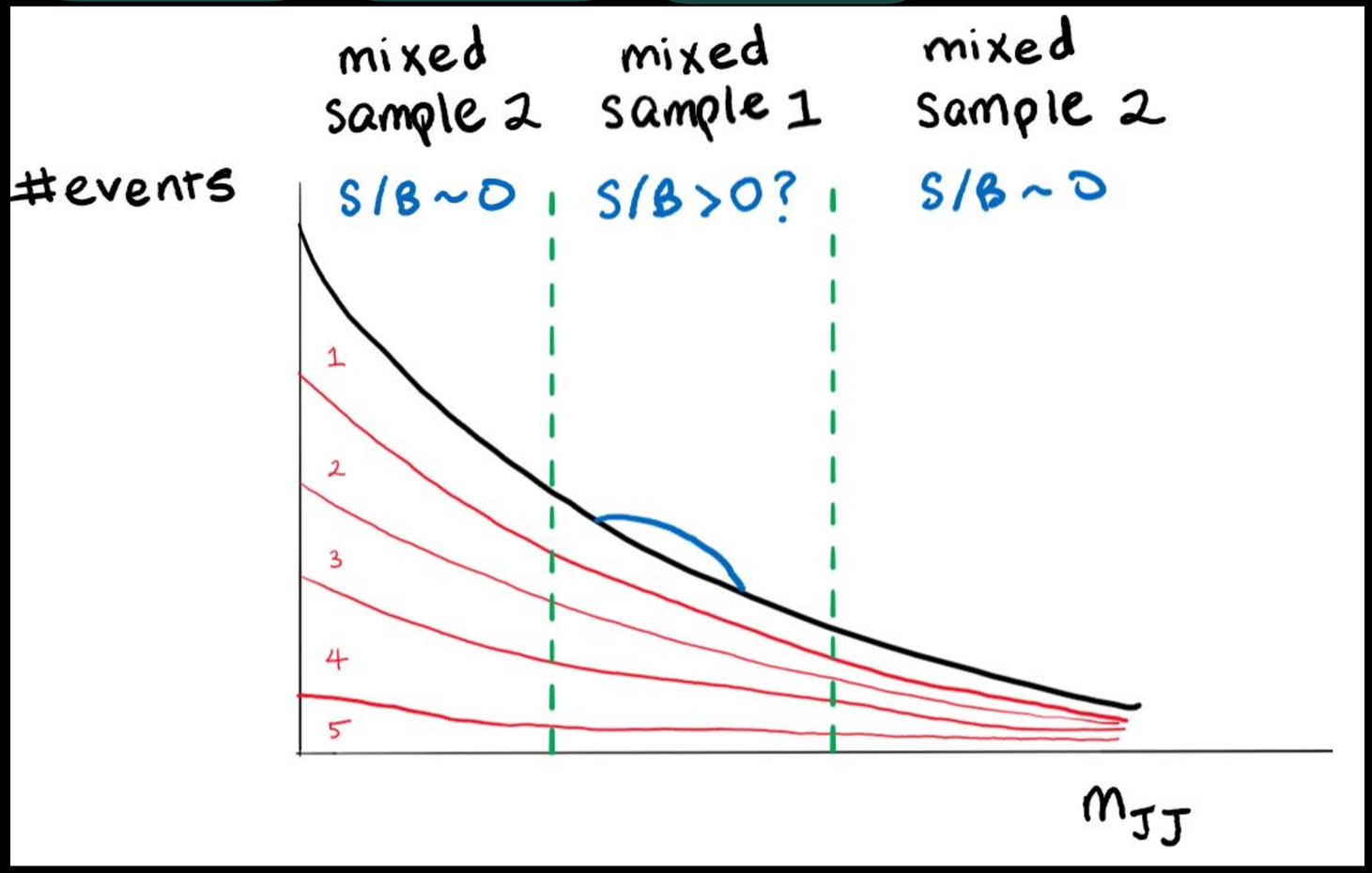
Weak classification without labels (CWoLa)



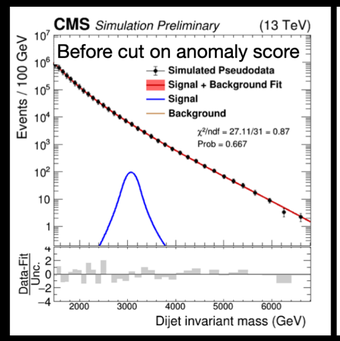
CWola hunting in ATLAS

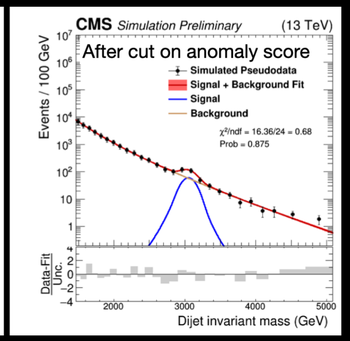
E.g CASE

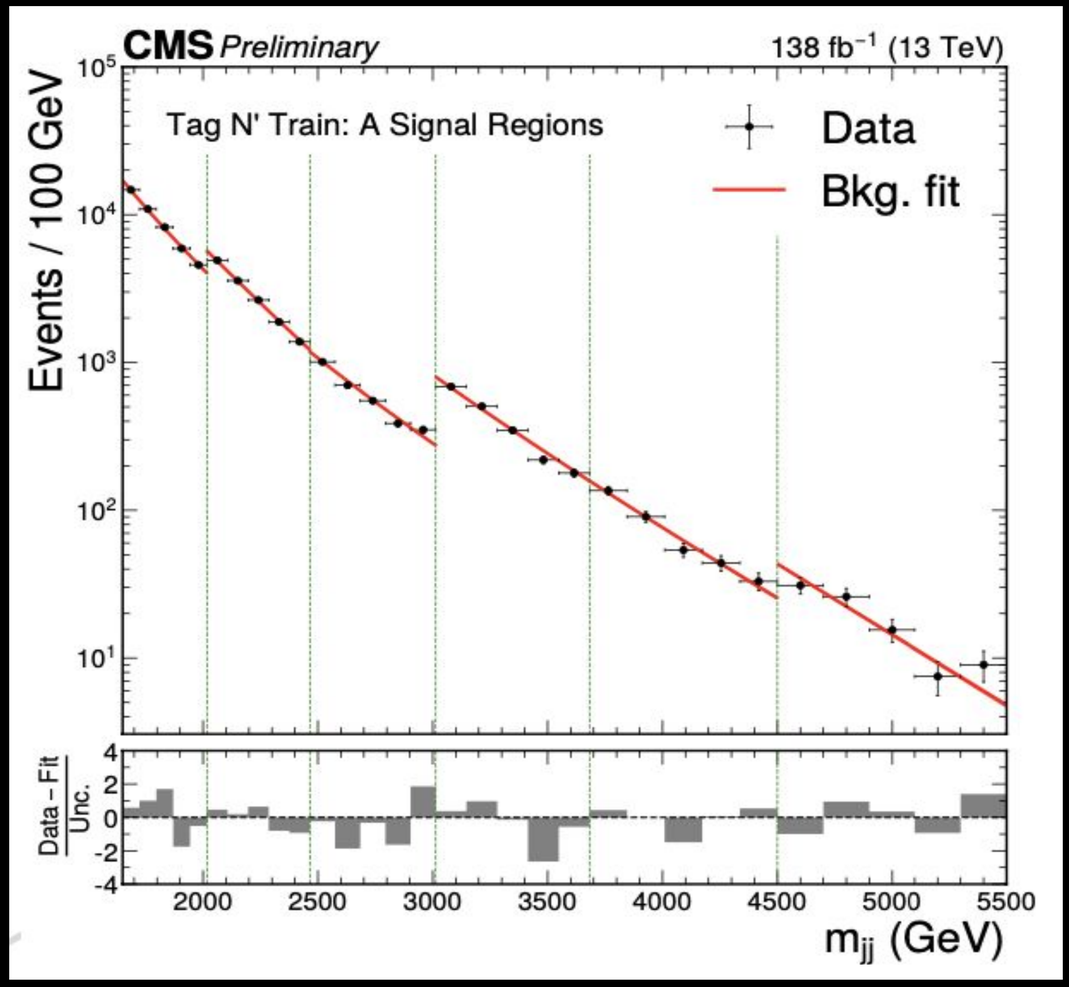

LABEL = SIGNAL
LABEL = BKG
S
B


mixed sample 2
mixed sample 1
mixed sample 2
#events
S/B ~ 0
S/B > 0 ?
S/B ~ 0
1
2
3
4
5
m_{JJ}

E.g CASE


CMS Simulation Preliminary (13 TeV)
Before cut on anomaly score
Simulated Pseudodata
Signal + Background Fit
Signal
Background
χ²/ndf = 27.11/31 = 0.87
Prob = 0.667
Events / 100 GeV
Data-Fit Unc.
Dijet invariant mass (GeV)


CMS Simulation Preliminary (13 TeV)
After cut on anomaly score
Simulated Pseudodata
Signal + Background Fit
Signal
Background
χ²/ndf = 16.36/24 = 0.68
Prob = 0.875
Events / 100 GeV
Data-Fit Unc.
Dijet invariant mass (GeV)


CMS Preliminary
138 fb⁻¹ (13 TeV)
Tag N' Train: A Signal Regions
Data
Bkg. fit
Events / 100 GeV
Data − Fit / Unc.
m_jj (GeV)

Neural Density Estimation for AD

ML-based interpolation from sidebands to signal region:

ANODE: estimates densities in data sidebands, and signal-region, interpolate sideband to signal region & constructs likelihood ratio

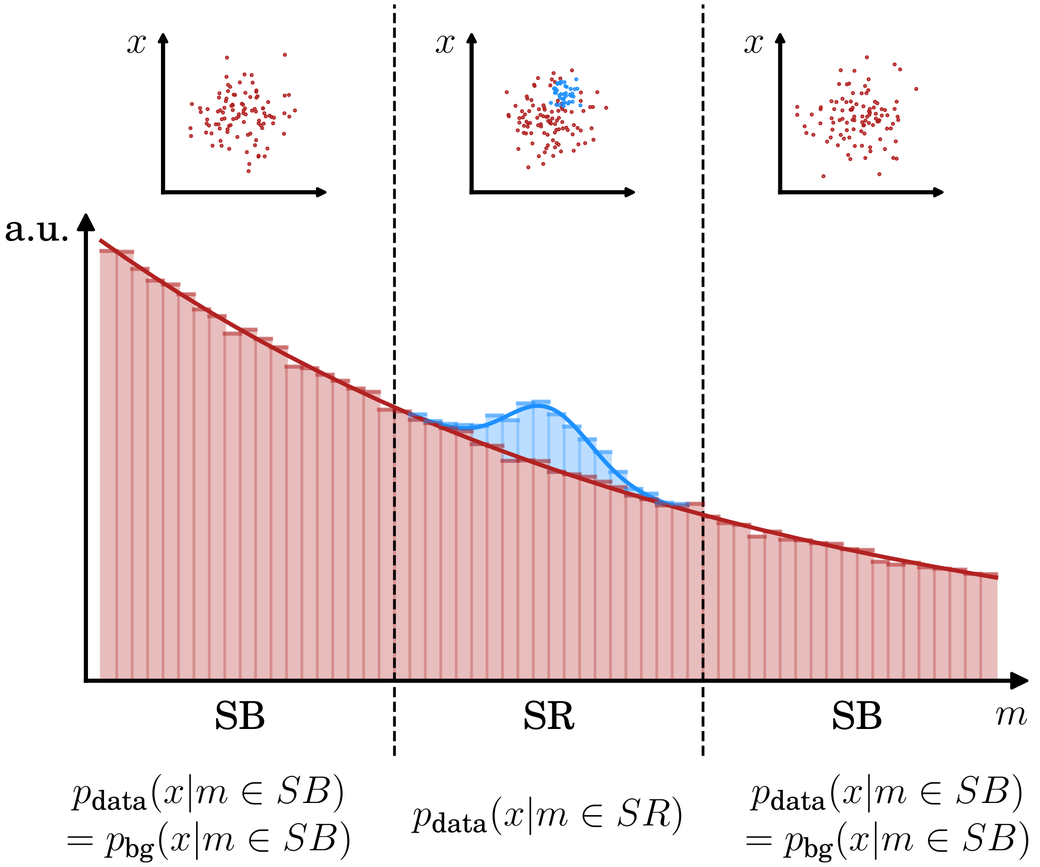
CATHODE: samples from the background model in signal region after interpolating, and estimates likelihood ratio by training classifier

CURTAINS: Train invertible NN conditioned on mass to map between sidebands

ML-based MC reweighting:

SALAD: Reweight simulation to match sideband, interpolate into the signal region and use a second classifier to get the likelihood

FETA: Map simulation to data in sidebands, then compare to SR data



End-to-end-approach: NPLM

Alternative approach: End-to-end DNN search

- How do we get around defining a signal hypothesis?
- What is alternate hypothesis to test reference?

Idea: Assume alternate model n(x|w) can be parametrised in terms of reference model n(x|R)

$$n(x \mid \vec{w}) = n(x \mid R) e^{f(x;\vec{w})}$$ ← Set of real functions

- Let DNN parametrise alternative model

$$f(x; \vec{w}) = NN$$

End-to-end-approach: NPLM

Alternative approach: End-to-end DNN search

- How do we get around defining a signal hypothesis?
- What is alternate hypothesis to test reference?

Idea: Assume alternate model n(x|w) can be parametrised in terms of reference model n(x|R)

$$n(x \mid \vec{w}) = n(x \mid R) e^{f(x;\vec{w})}$$ ← Set of real functions

- Let DNN parametrise alternative model

$$f(x; \vec{w}) = NN$$

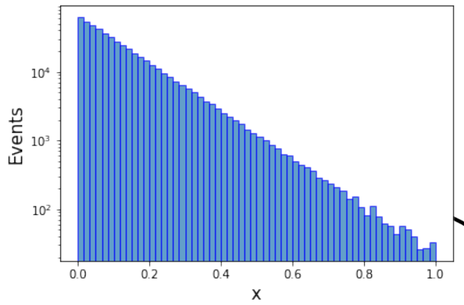
- Formulate loss as log likelihood.
  → Trained DNN **is** the maximum likelihood fit to data and reference log-ratio
  → best approximate of true data distribution

$$f(x, \hat{\mathbf{w}}) \simeq \log \left[\frac{n(x|\mathrm{T})}{n(x|\mathrm{R})} \right]$$ ← True underlying data distribution ← MC distribution

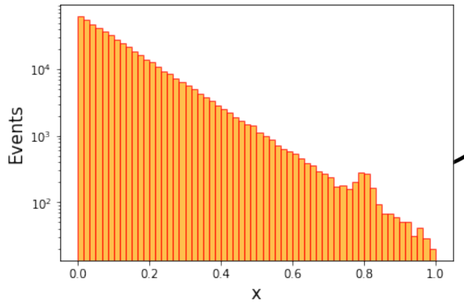
INPUTS
- any high level features

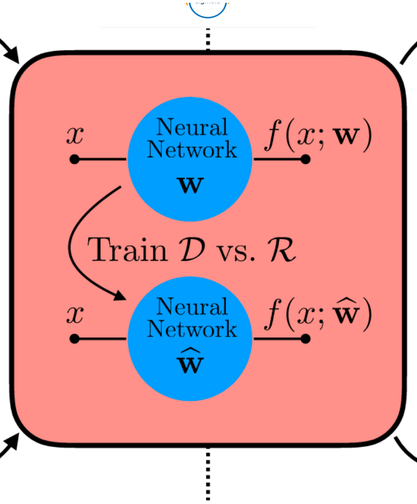
OUTPUTS
-t^{obs} and $f(x; \hat{w})$

QCD MC R

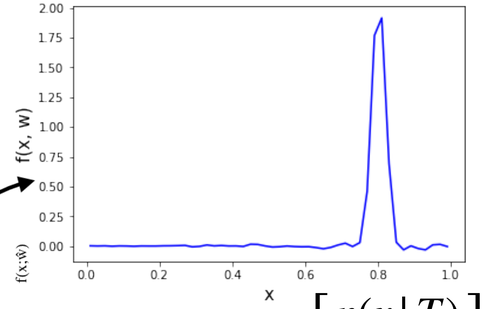


CMS DATA D





1) Best fit log ratio of data and MC PDFs



$$f(x, \hat{w}) \simeq \log \left[\frac{n(x \mid T)}{n(x \mid R)} \right]$$

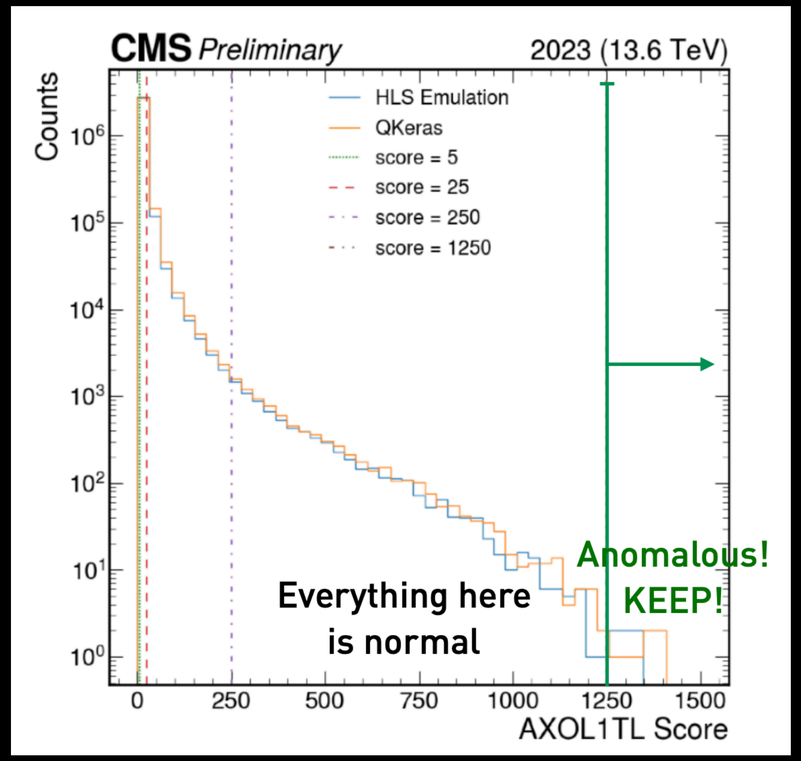
2) test-statistic on data sample t_{obs}

$$t(\mathcal{D}) = -2 \operatorname*{Min}_{\{\mathbf{w}\}} L[f]$$ ← **DNN loss function!**
Can be used to build hypothesis test + p-value
Data → toys under R, repeat

$$f(x, \hat{\mathbf{w}}) \simeq \log \left[\frac{n(x|\mathrm{T})}{n(x|\mathrm{R})} \right]$$
← True underlying data distribution
← MC distribution

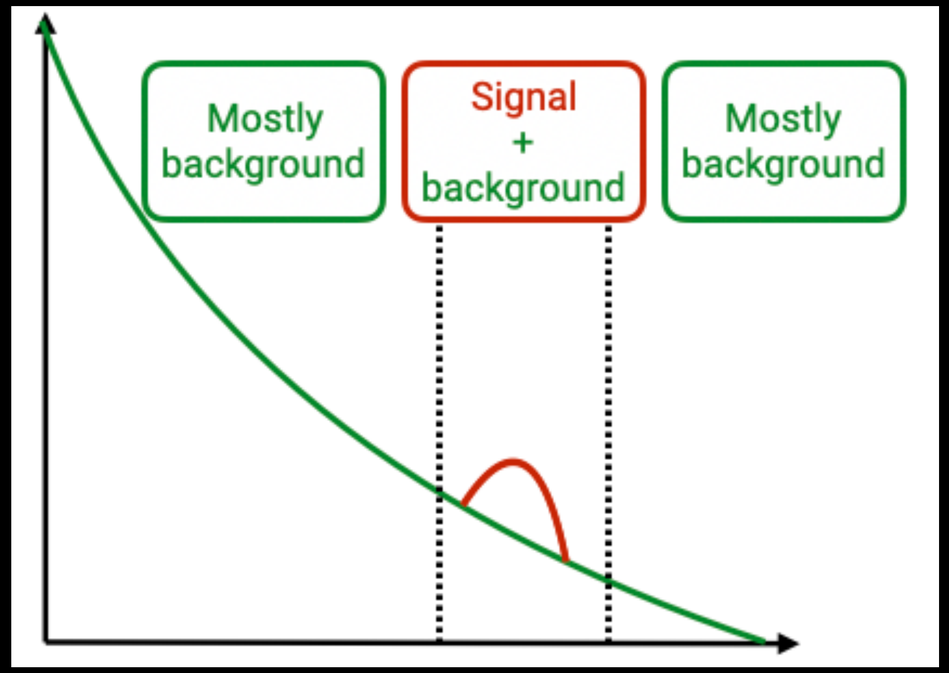
Types of anomaly detection

Outlier detection



Unsupervised learning:
only features; no labels!

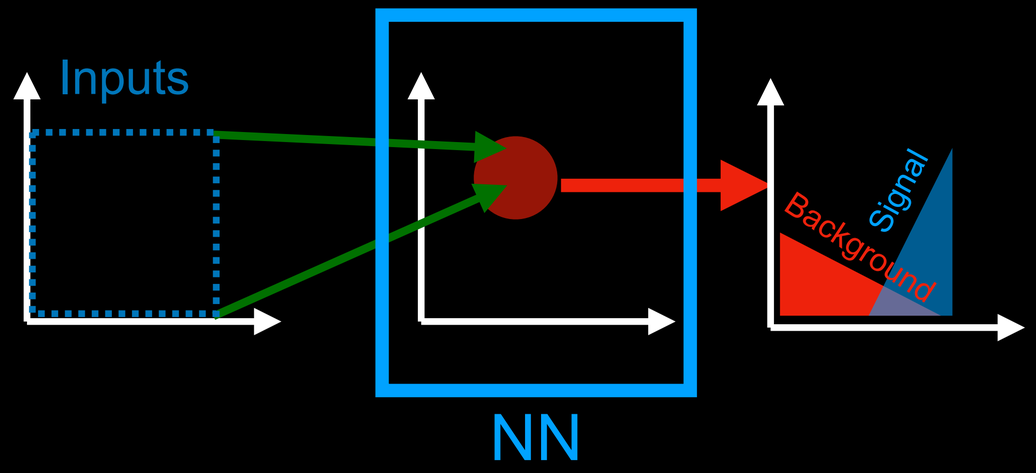
Detecting overdensities



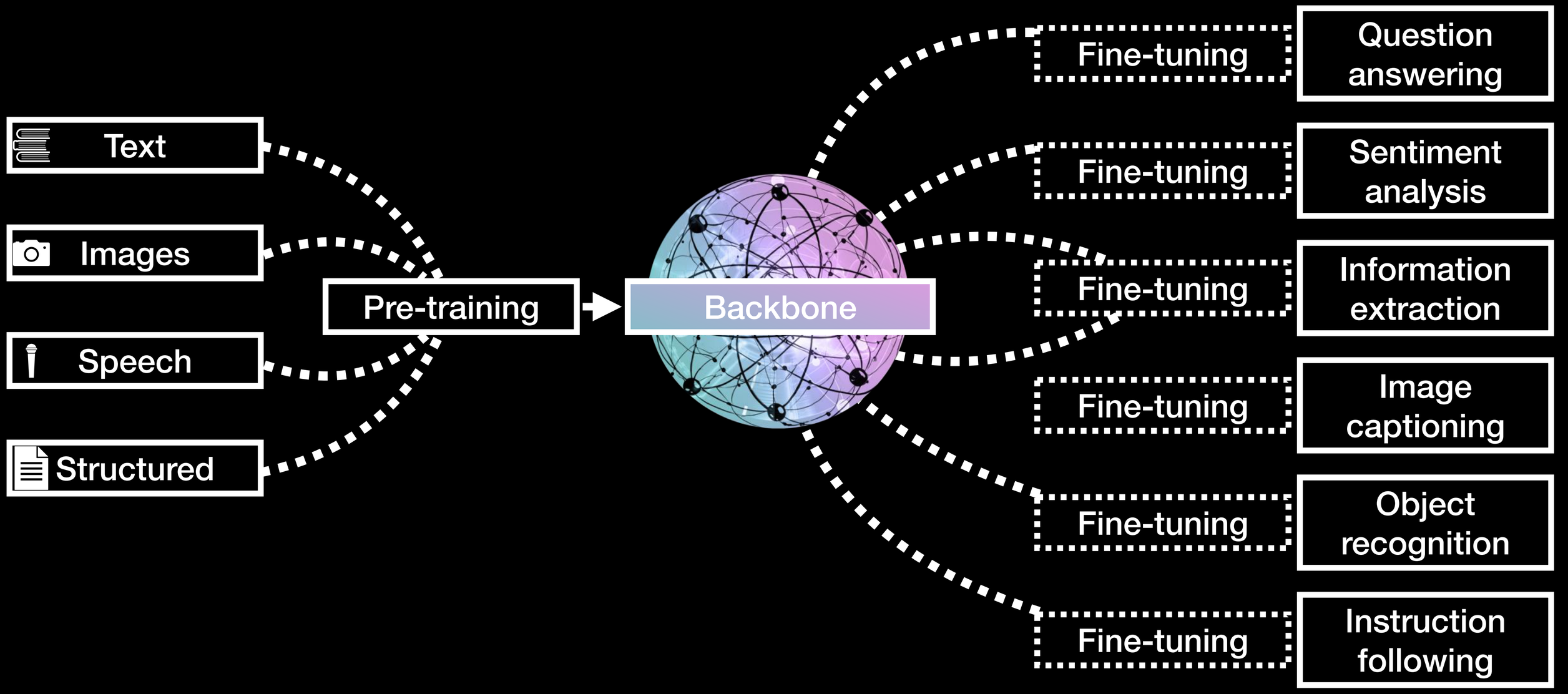
Weakly-supervised learning:
noisy labels

ETH zürich
Building better spaces
Thea Klæboe Årrestad (ETH Zürich)

Too many models, too little learning?

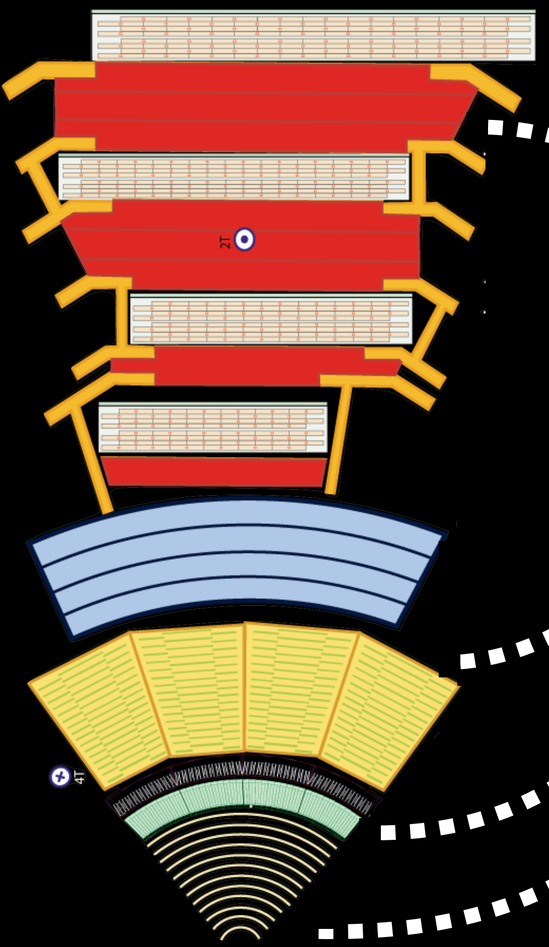


Discrimination



See mondays lecture on transformers and foundation models by Anna!

Heterogeneous detector
Multi-modal input!



Pre-training

Backbone

Fine-tuning — Jet reconstruction

Fine-tuning — Electron reconstruction

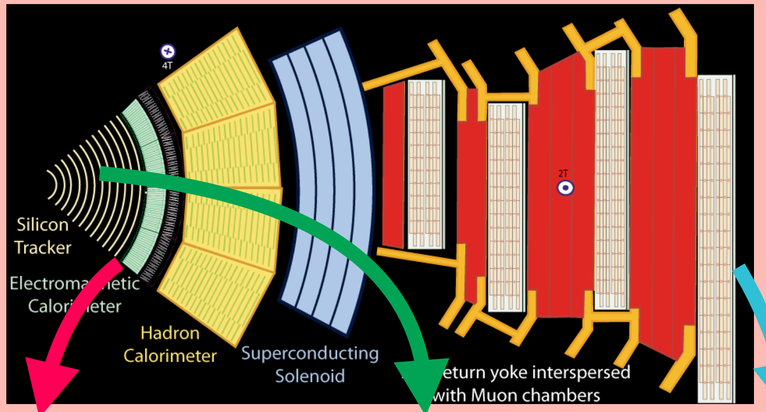
Fine-tuning — Pile-up removal

Fine-tuning — Missing energy computation

Fine-tuning — Anomaly Detection

Fine-tuning — 0/1?

(Generate simulation?)



Some tiny NN here

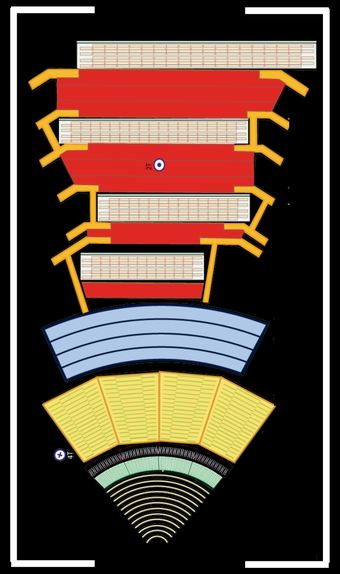
4 tiny NNs here

A few NNs and BDTs here

2 tiny NNs here

BDT here

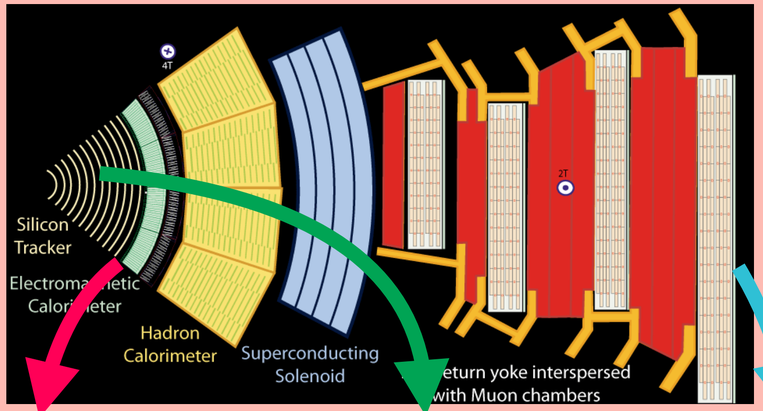
Accept / Reject



$$f(x; w^*) \rightarrow \hat{y}$$

$$x = (x_1, x_2, \ldots,)$$

Too many models, too little learning?



Some tiny NN here

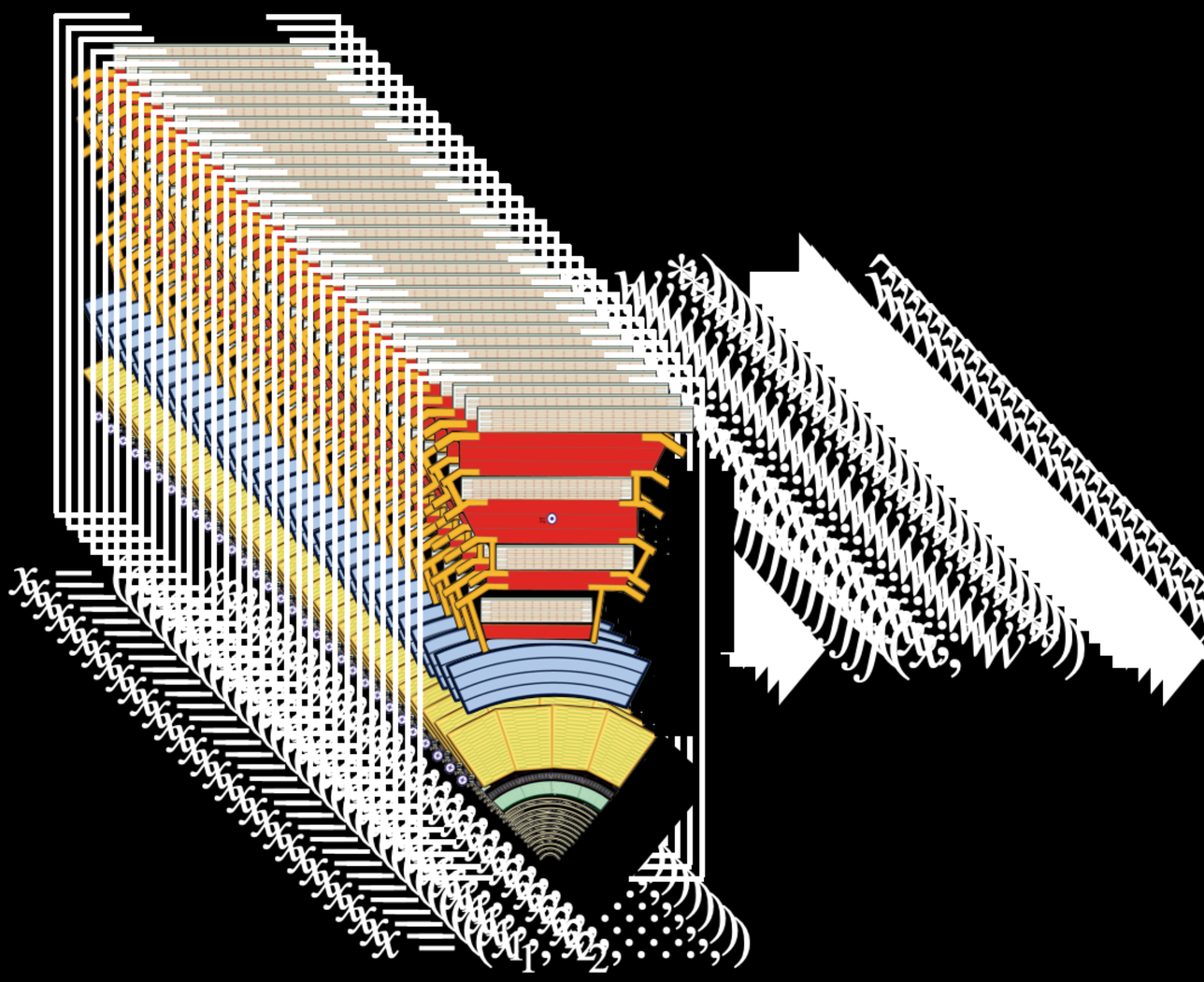
4 tiny NNs here

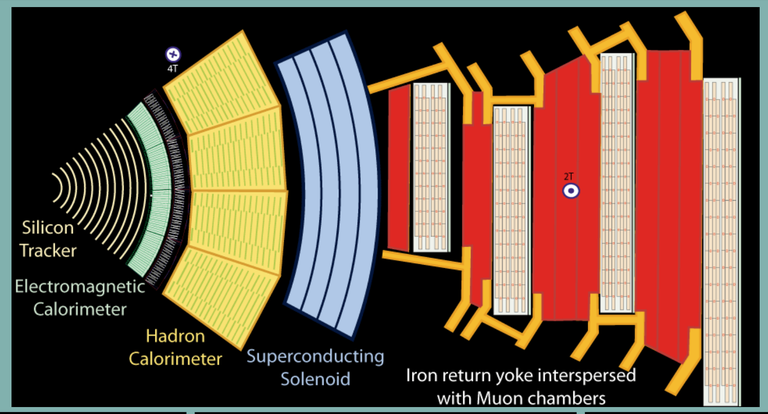
A few NNs and BDTs here

2 tiny NNs here

BDT here

Accept / Reject





CALORIMETER PRE-PROCESSING

TRACKING PRE-PROCESSING

MUON CHAMBERS PRE-PROCESSING

End-to-end reconstruction model

Latent representation

Downstream Task

Downstream Task

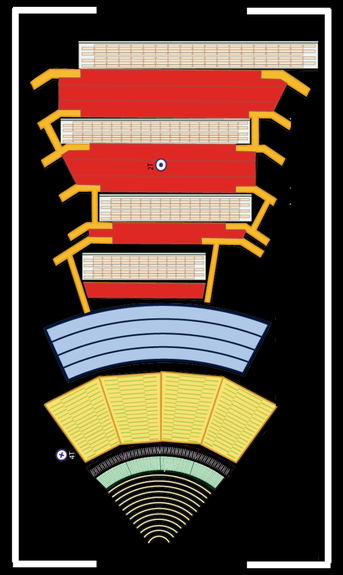
Downstream Task

Downstream Task

μs

Accept / Reject

One model, learn neural embedding?



$x = (x_1, x_2, \ldots,)$

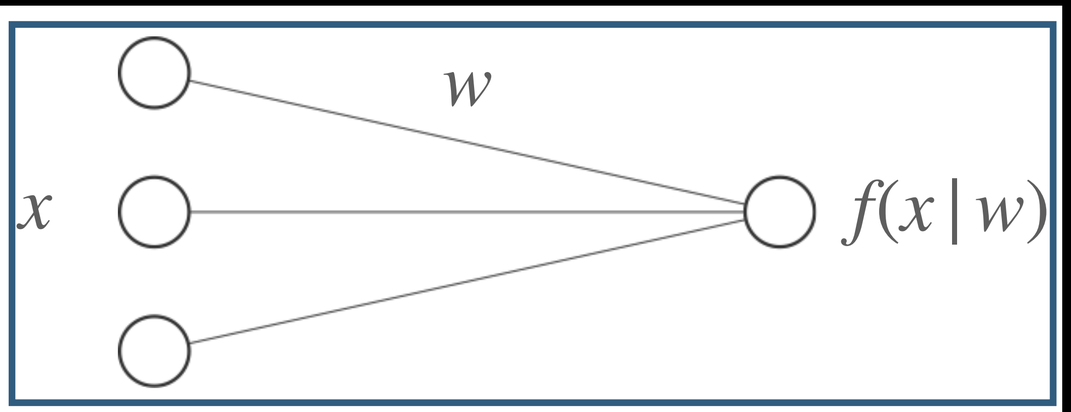
Some new space

Downstream Task

Downstream Task

Downstream Task

NNs are featurizers!



Linear model: $f(x \mid w) = w^\intercal x$

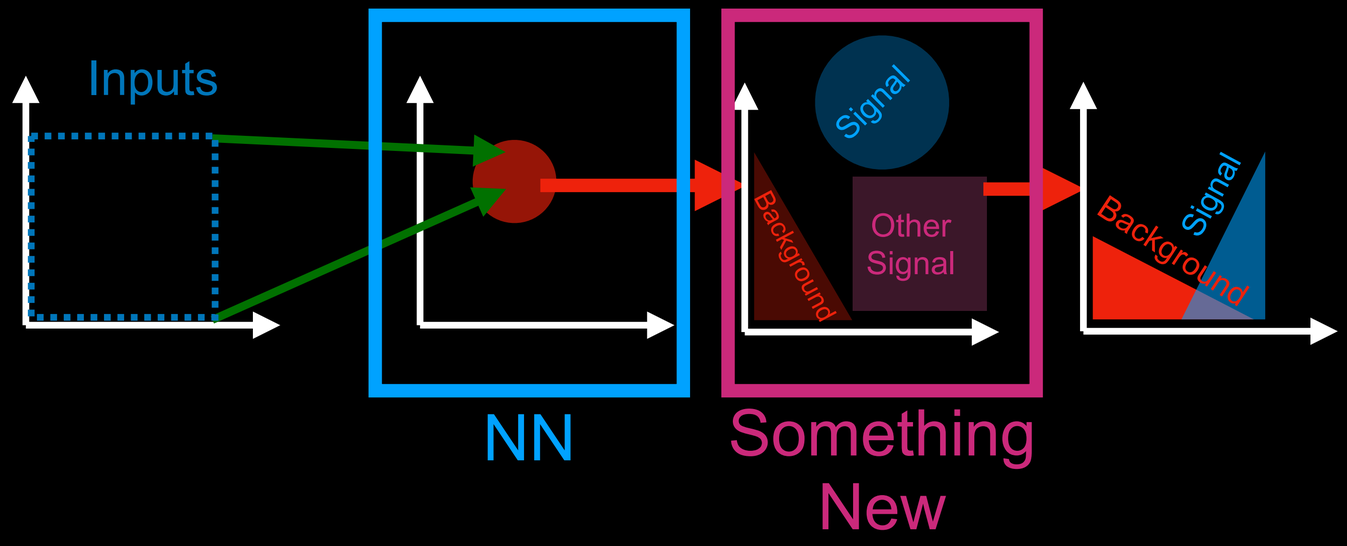
Neural network: linear model after inputs are mapped to features through a nonlinear transformation σ

$f(x \mid w_1, w_2) = w_2^T \sigma(w_1^T x)$

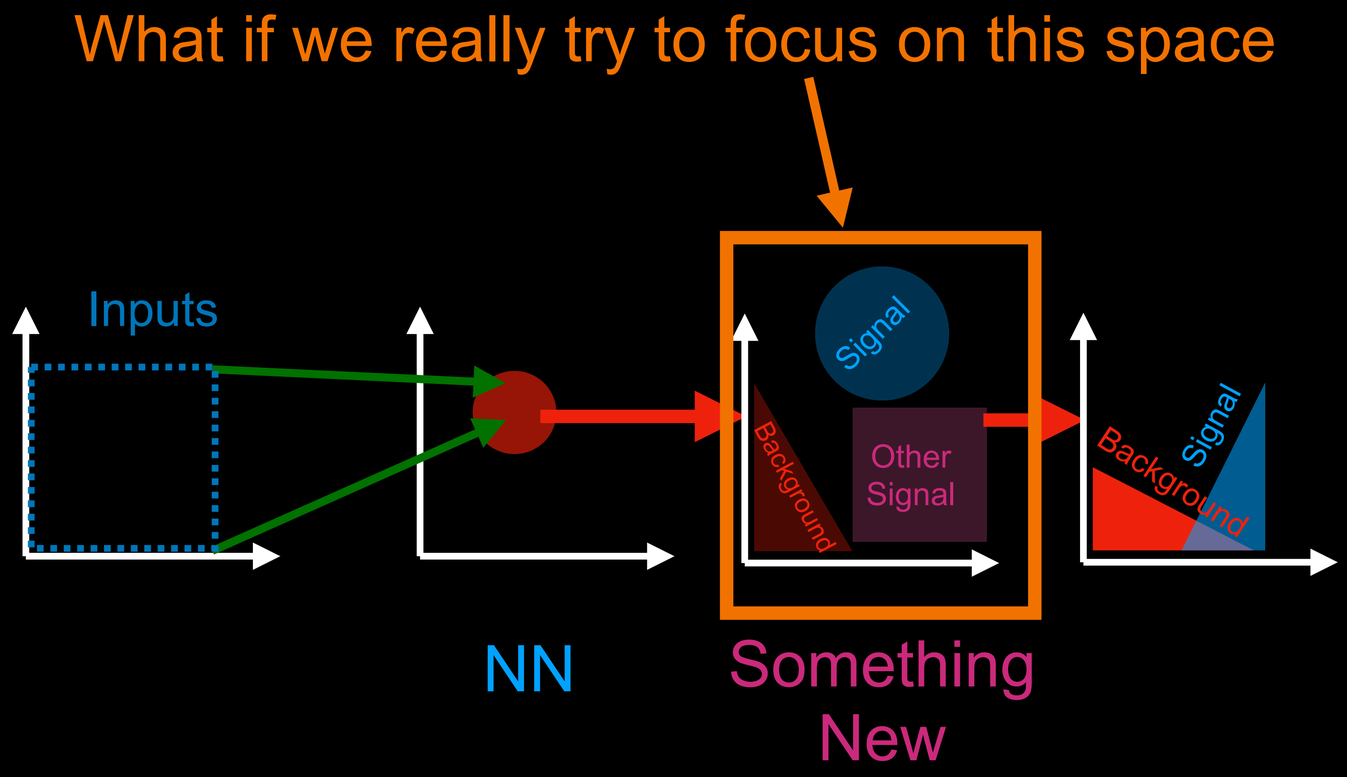
Neural networks have linear models as basic building block

Act as “automatic featurizers/ embeddings”!

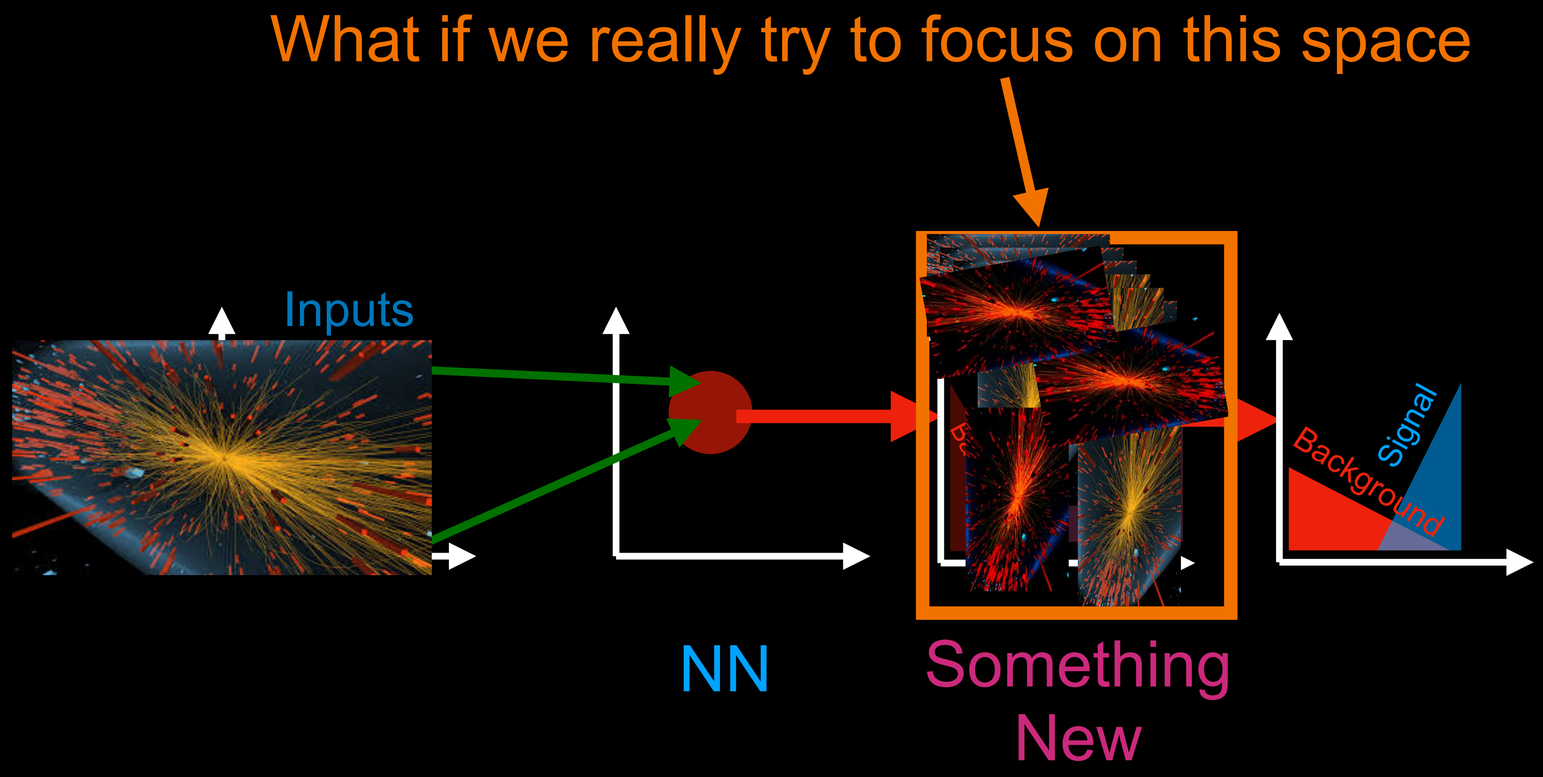
How can we make meaningful embeddings?



Metric Learning

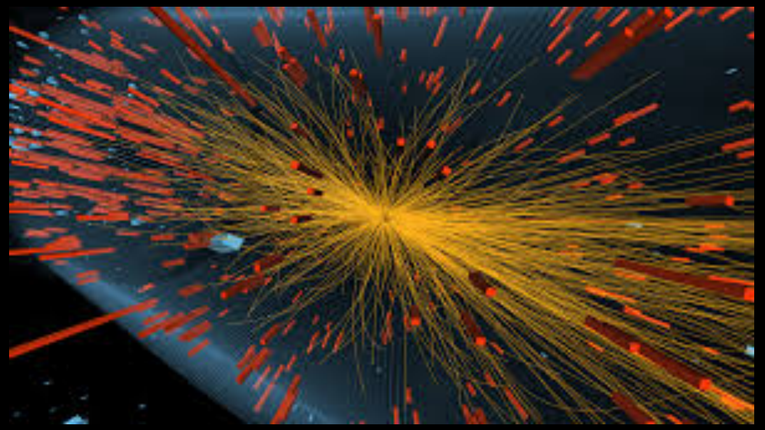


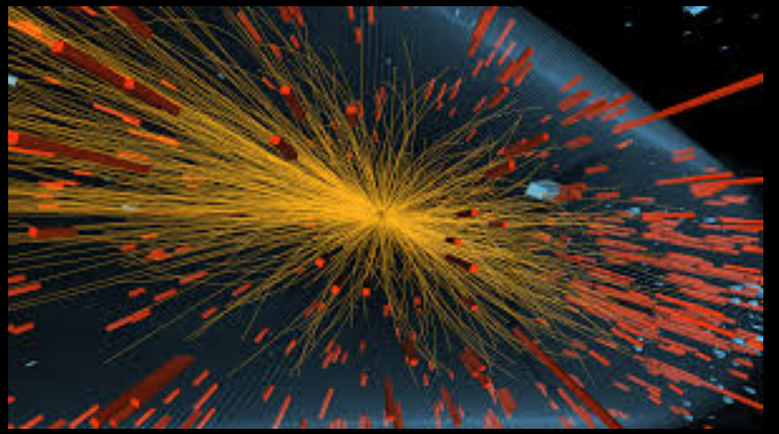
Neural embedding



Neural embedding

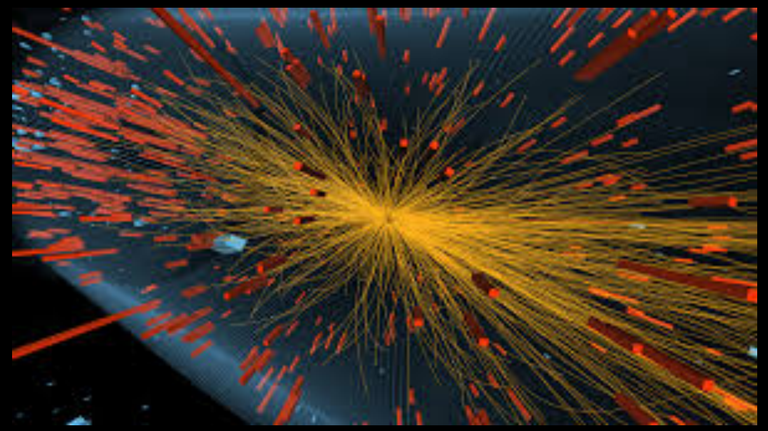
Learning the space

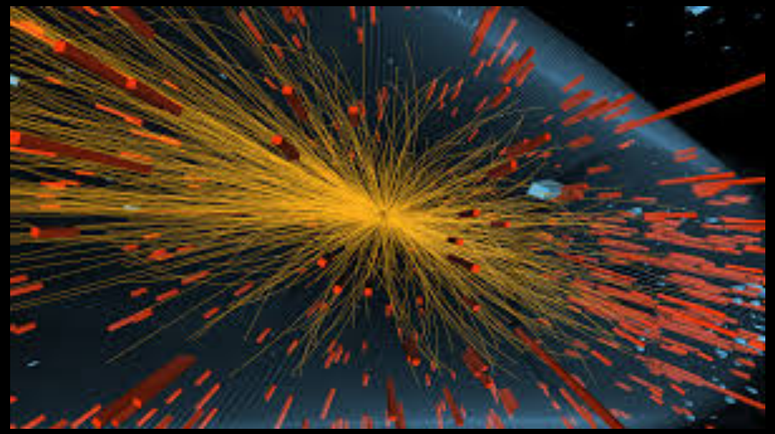




Learning the space

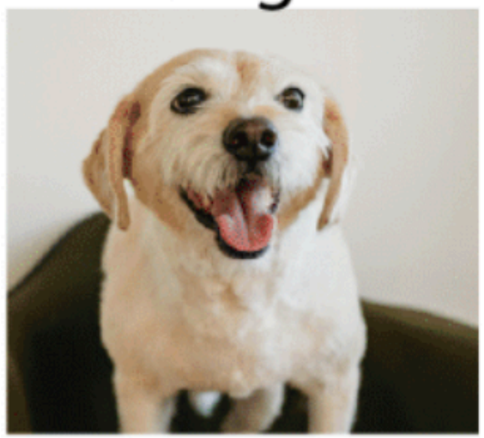
- By looking at data, we can learn a lot
  - Go over input piece by piece
  - Analyze every aspect
  - Compare every feature
- Find distinctive style of the input
  - can be done e.g by looking for a deviation





Cat A

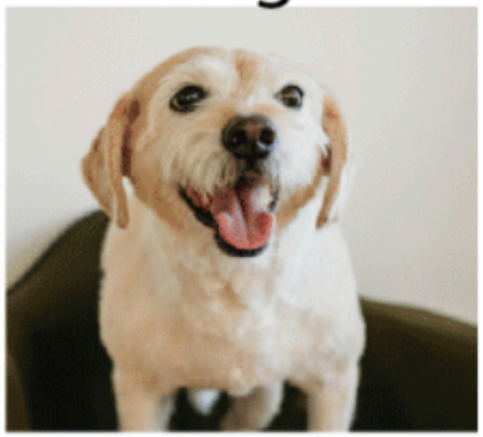
Dog A



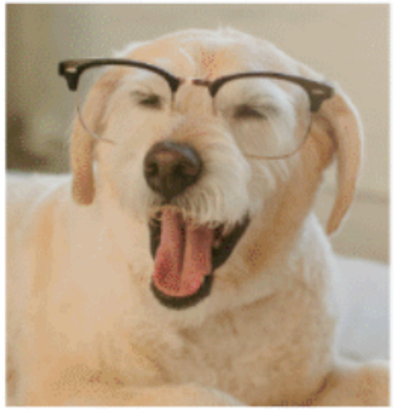
Cat A

Augmented Cat A

Dog A



Augmented Dog A

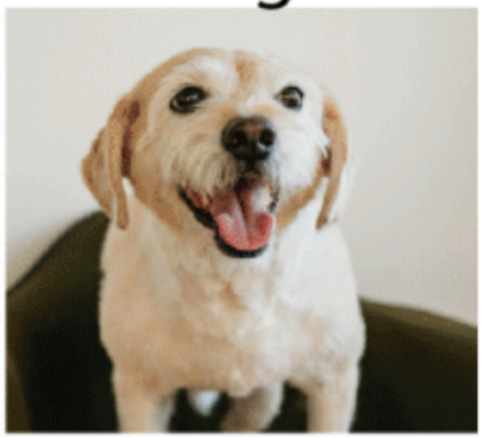


Minimize

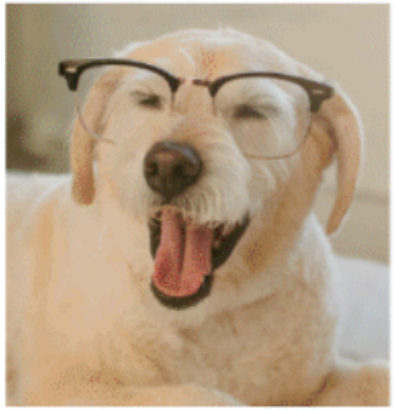
Cat A

Augmented Cat A

Dog A



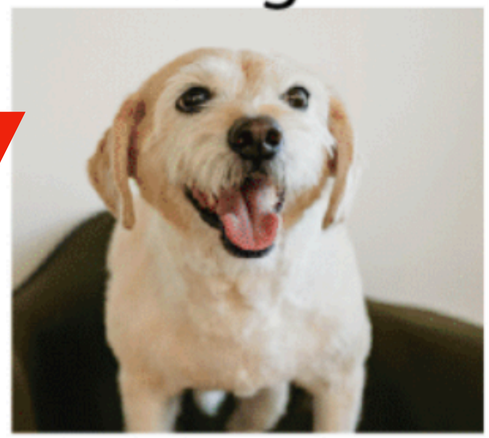
Augmented Dog A



Maximize

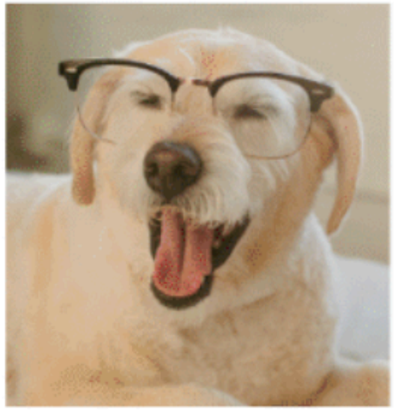
Cat A

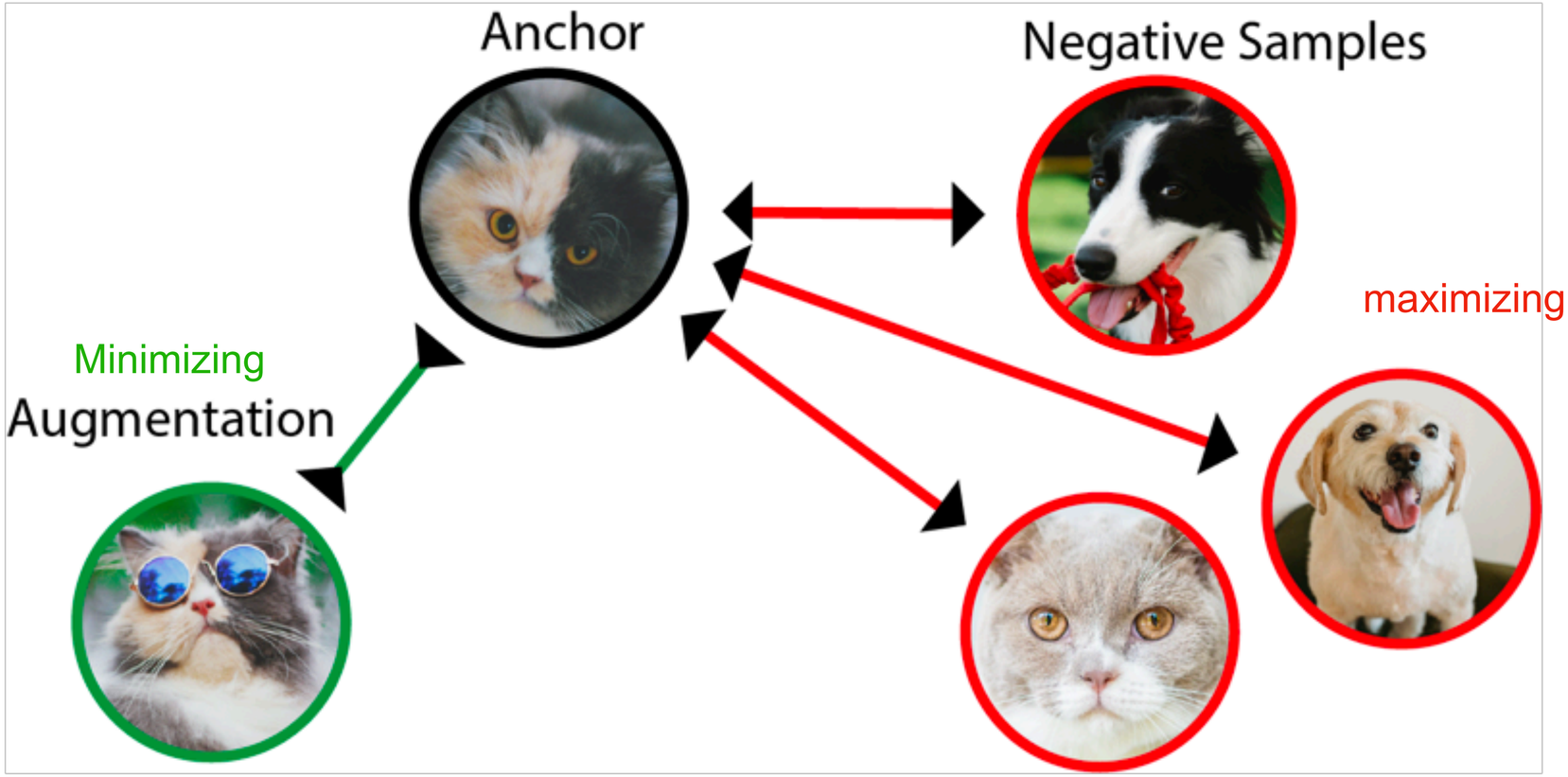
Dog A



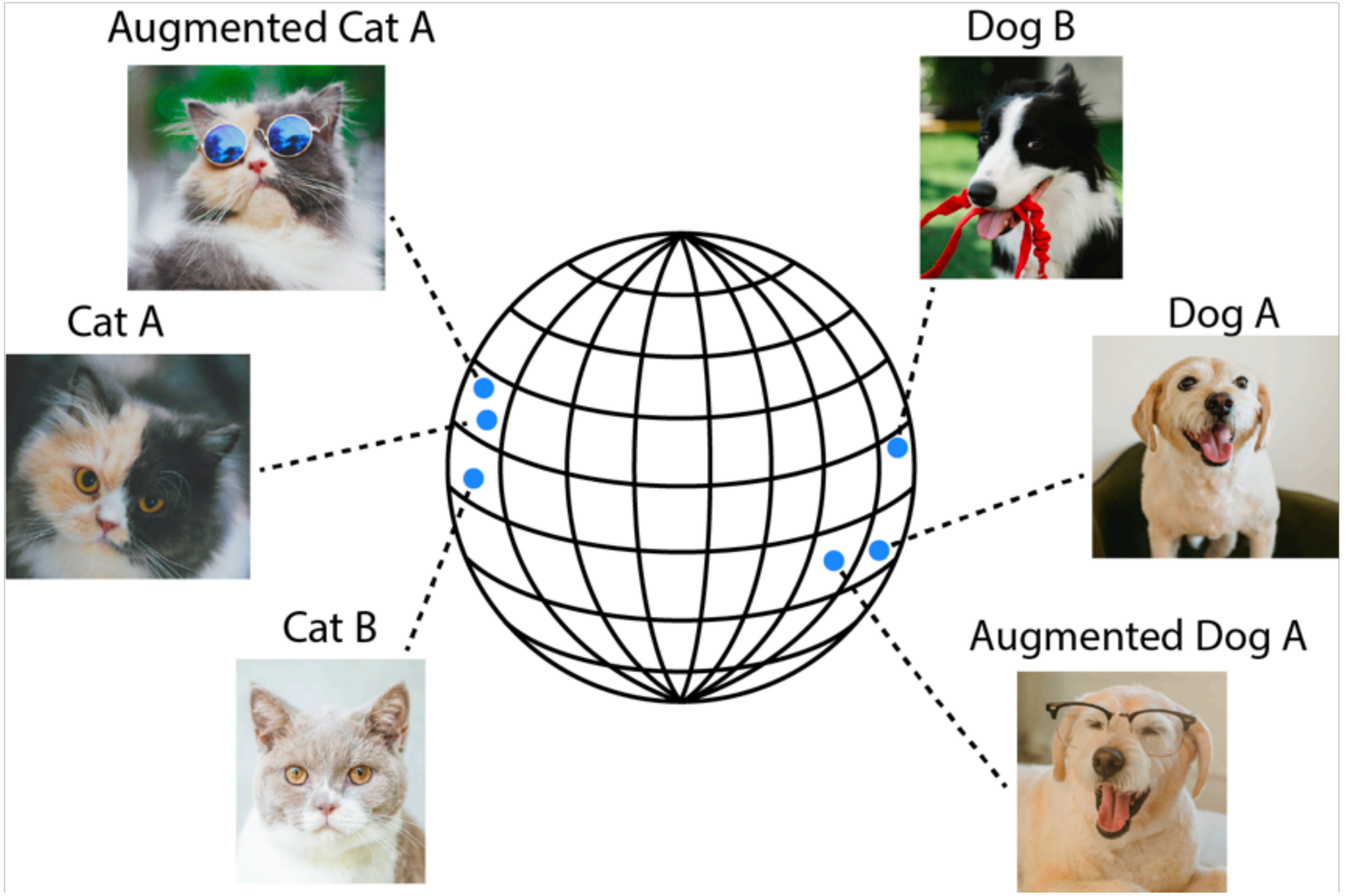
Augmented Cat A

Augmented Dog A



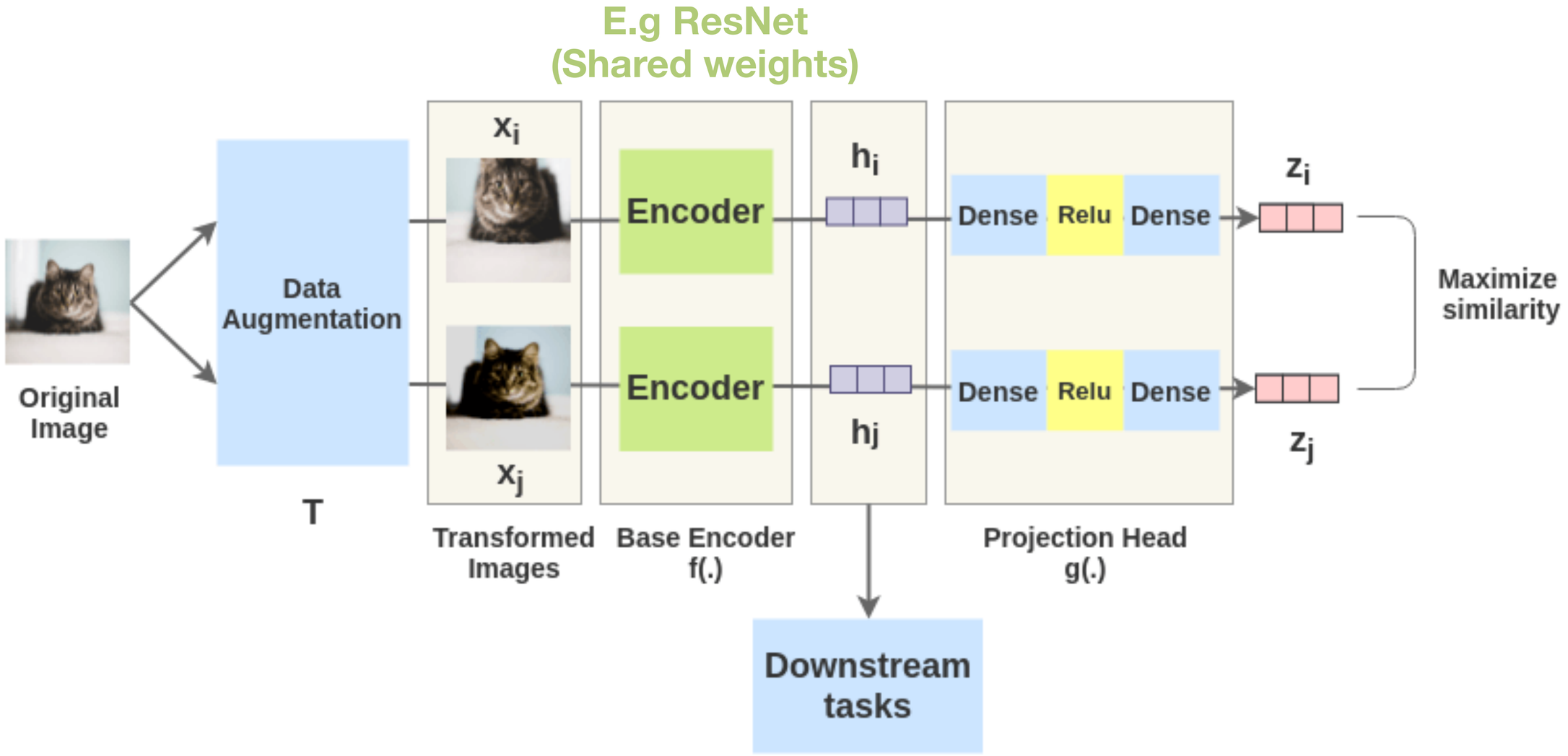


- Minimizing and maximizing distances learns a space


Augmented Cat A
Dog B
Cat A
Dog A
Cat B
Augmented Dog A

SimCLR Framework

Designing neural embeddings

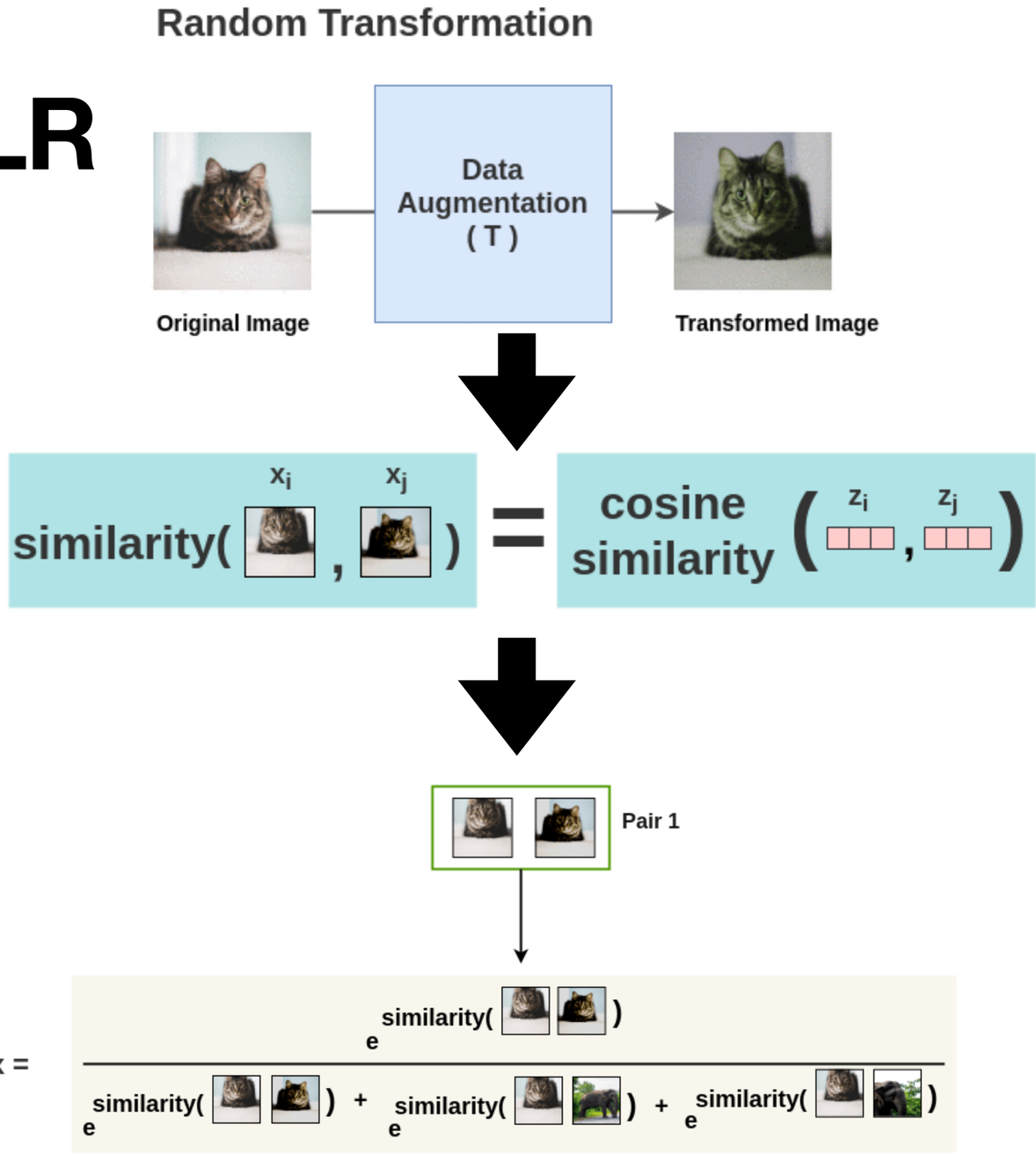


Contrastive loss - SimCLR

Designing neural embeddings

- For all methods: attractive force, repulsive force and some sort of regularization

$$\ell_{i,j} = -\log \frac{\exp(\text{sim}(\mathbf{z_i}, \mathbf{z_j})/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(\text{sim}(\mathbf{z_i}, \mathbf{z}$$

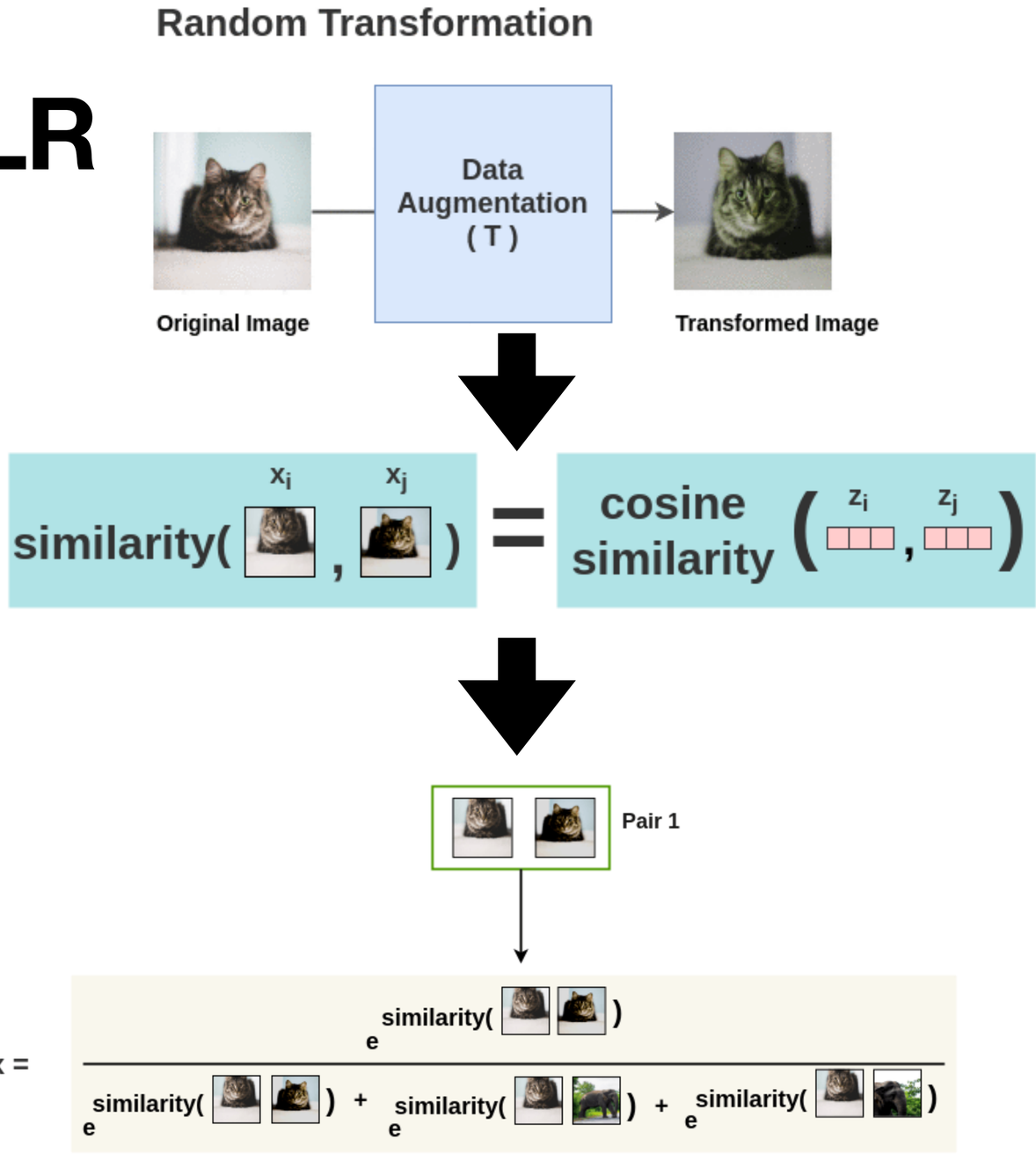


Contrastive loss - SimCLR

Designing neural embeddings

- For all methods: attractive force, repulsive force and some sort of regularization

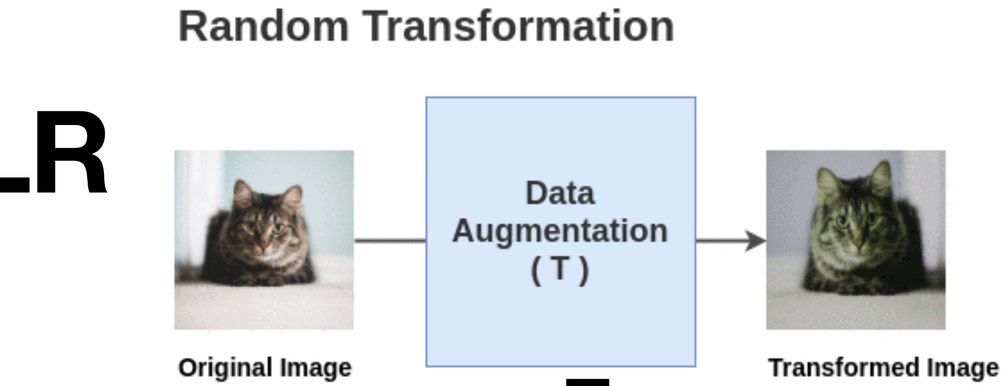
$$\ell_{i,j} = -\log \frac{\exp(\text{sim}(\mathbf{z_i}, \mathbf{z_j})/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(\text{sim}(\mathbf{z_i}, \mathbf{z}$$

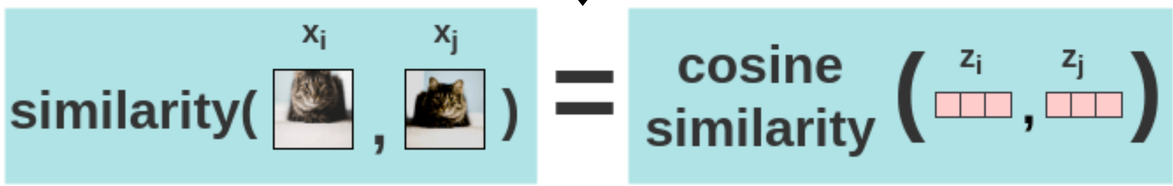


Contrastive loss - SimCLR

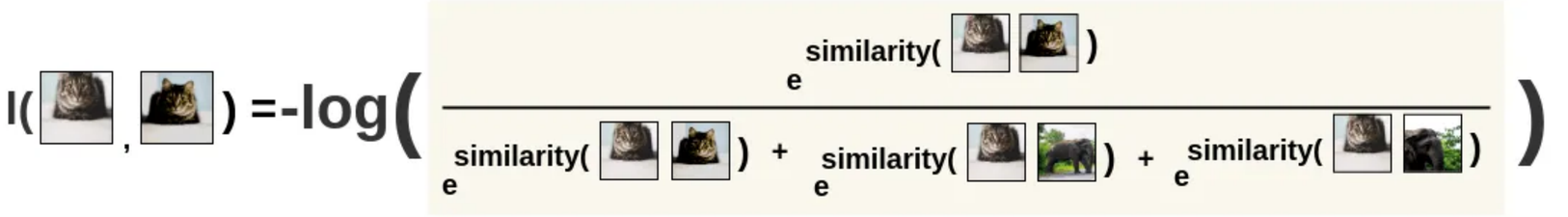
Designing neural embeddings

- For all methods: attractive force, repulsive force and some sort of regularization





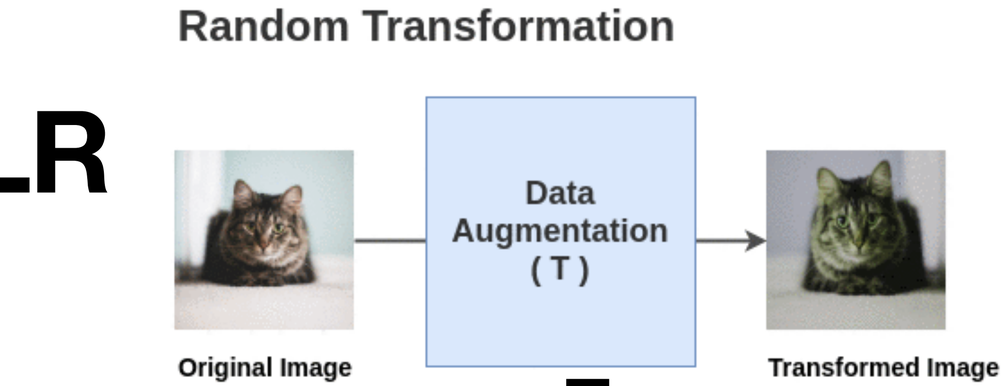
$$\ell_{i,j} = -\log \frac{\exp(\mathrm{sim}(\mathbf{z_i}, \mathbf{z_j})/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(\mathrm{sim}(\mathbf{z_i}, \mathbf{z}$$

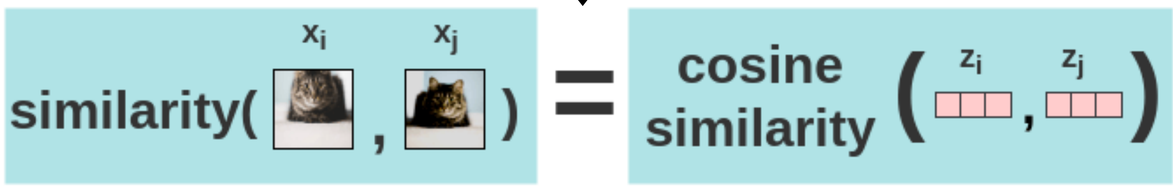


Contrastive loss - SimCLR

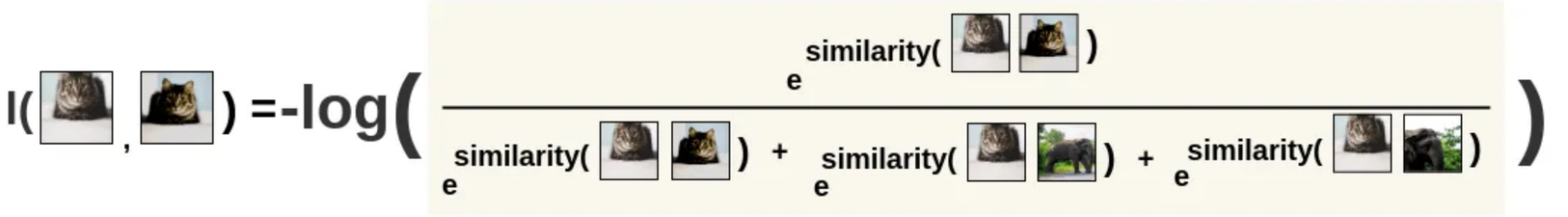
Designing neural embeddings

- For all methods: attractive force, repulsive force and some sort of regularization



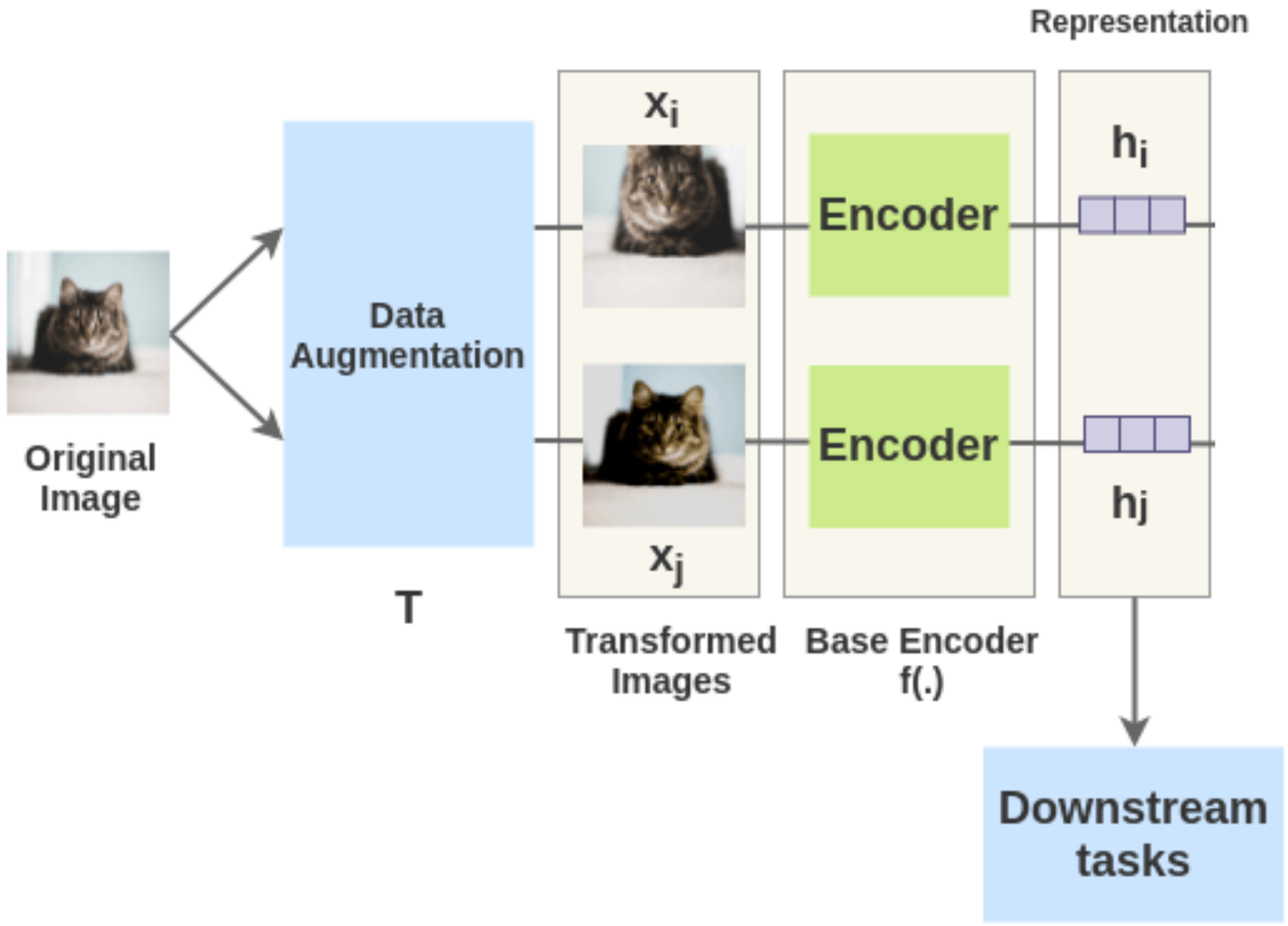


$$\ell_{i,j} = -\log \frac{\exp(\mathrm{sim}(\mathbf{z_i}, \mathbf{z_j})/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(\mathrm{sim}(\mathbf{z_i}, \mathbf{z}$$

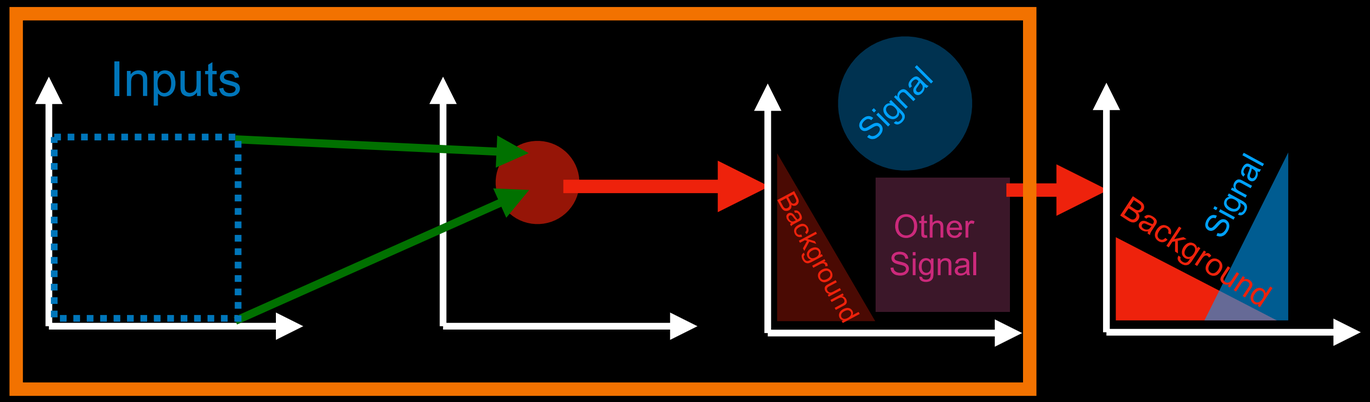


SimCLR Framework

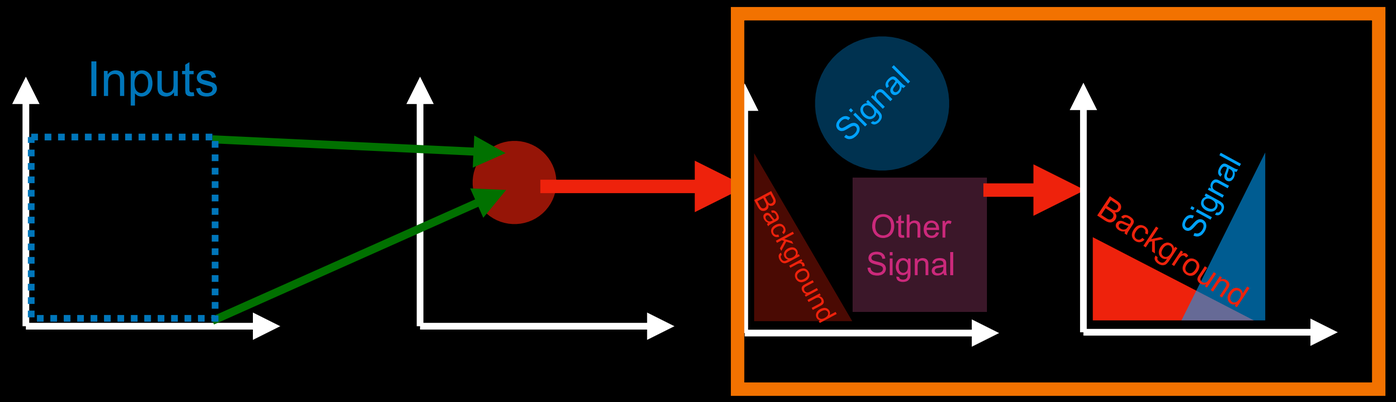
Designing neural embeddings



Training number one: learn embedding



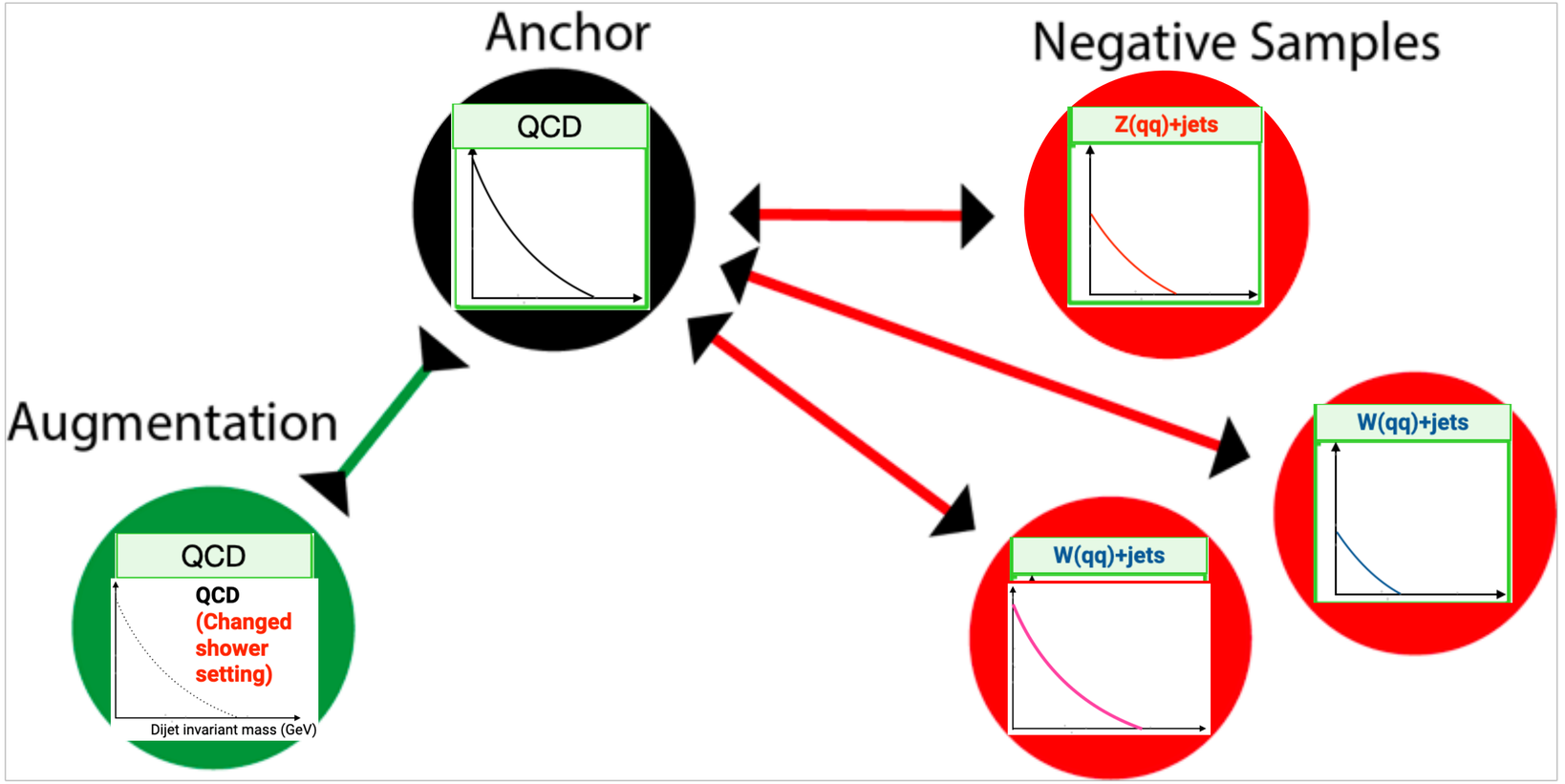
Training number two: fine tuning



NN

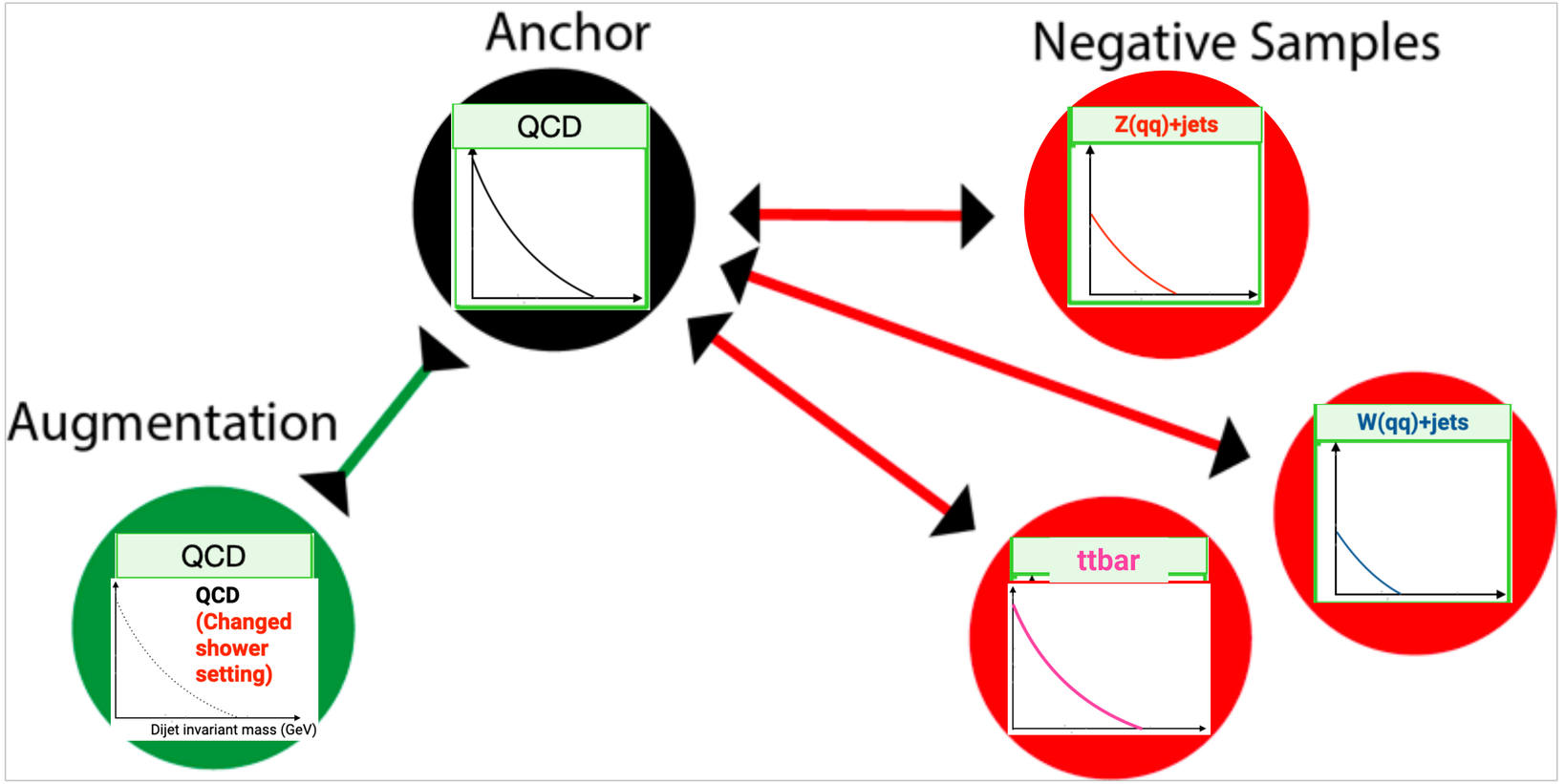
Something
New

Physically motivated augmentations?

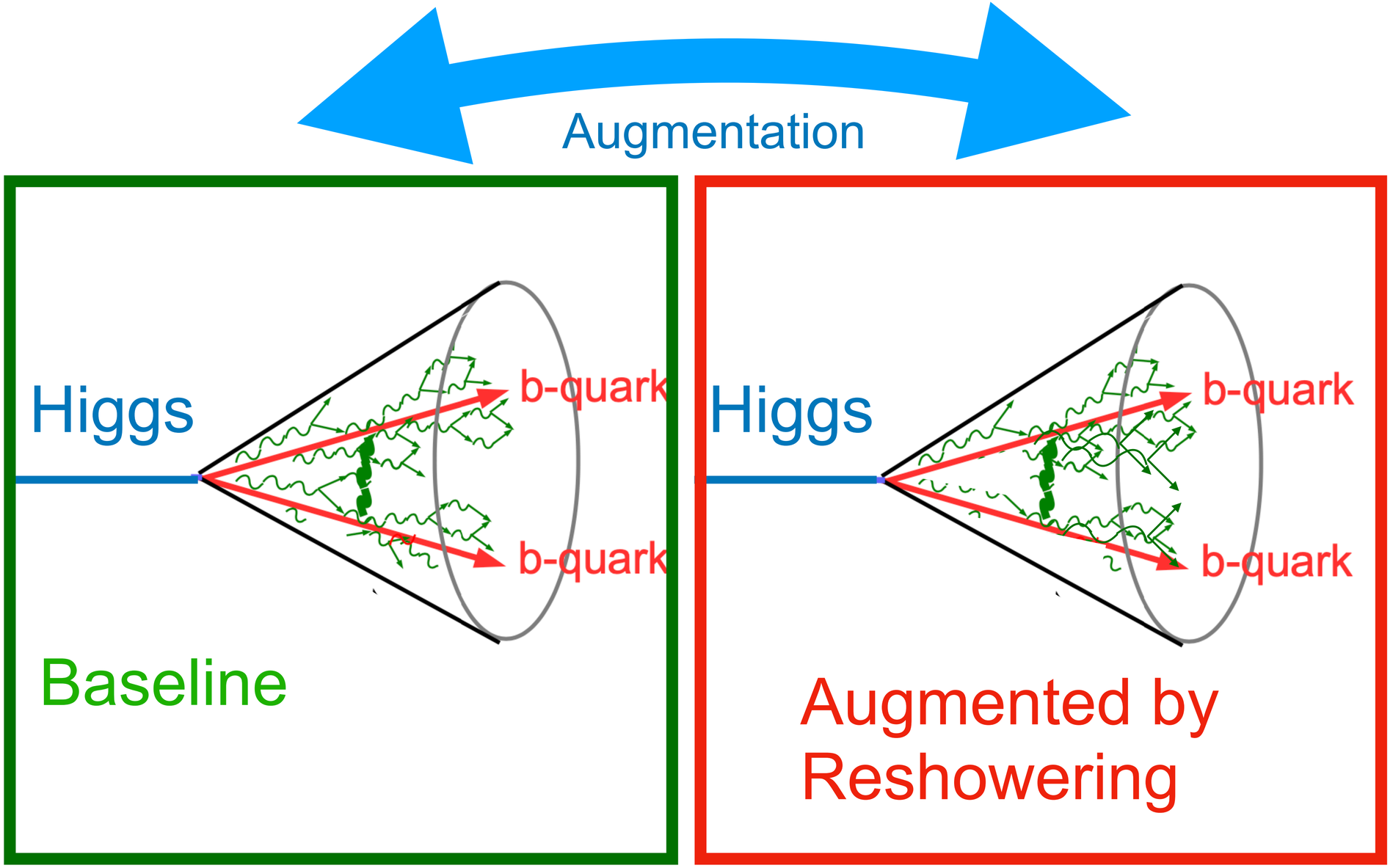


No class labels used in training! How do we augment detector data?

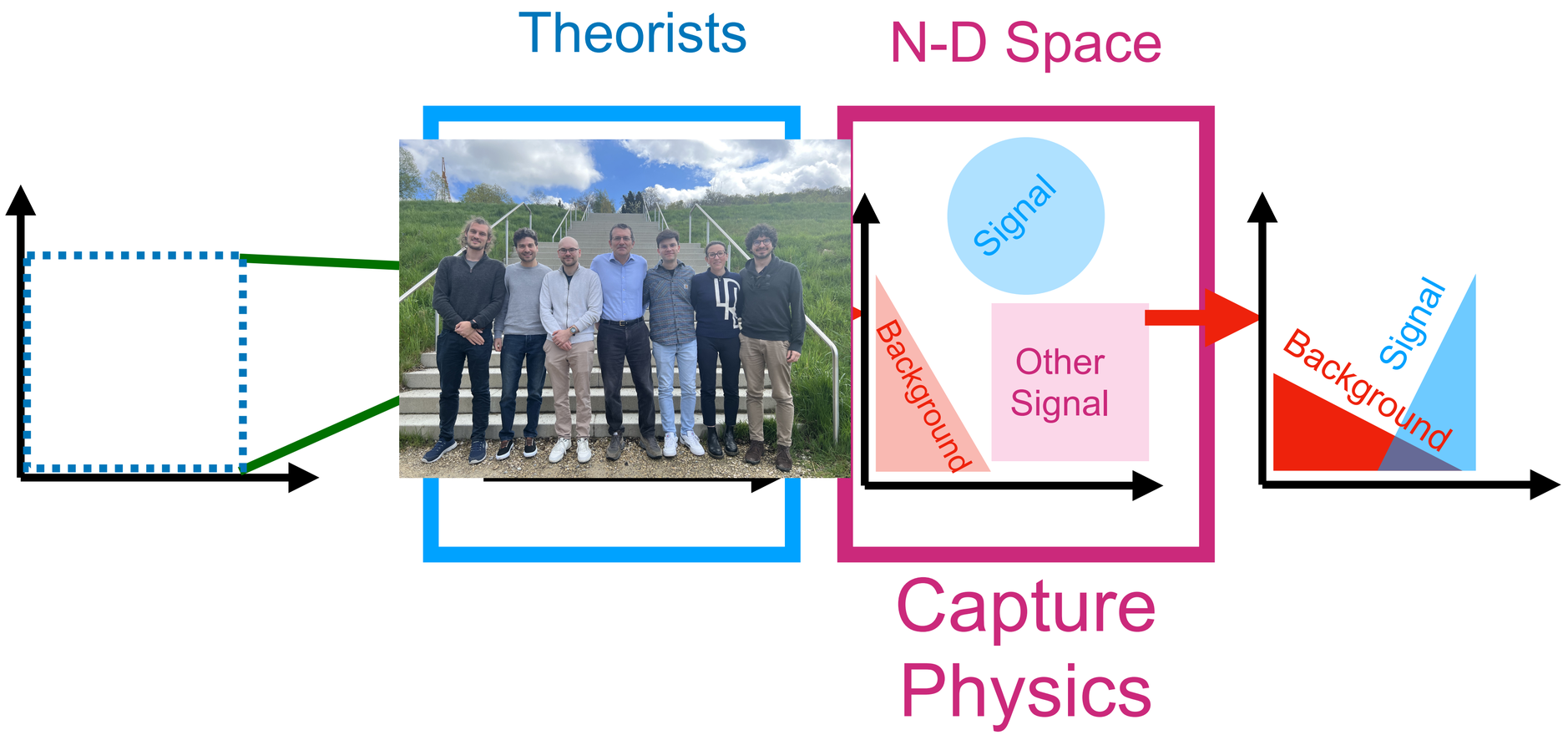
Physically motivated augmentations?



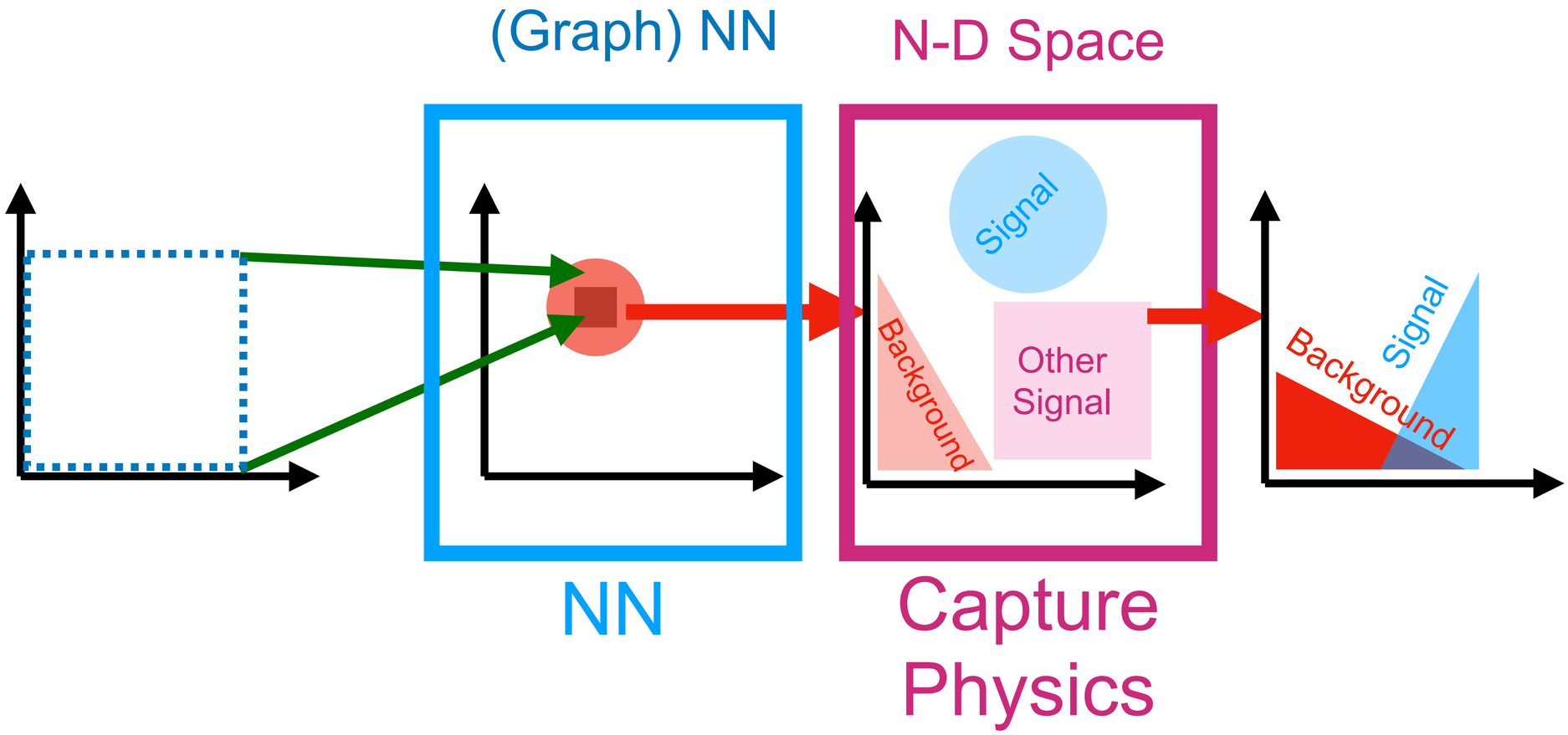
No class labels used in training! How do we augment detector data?



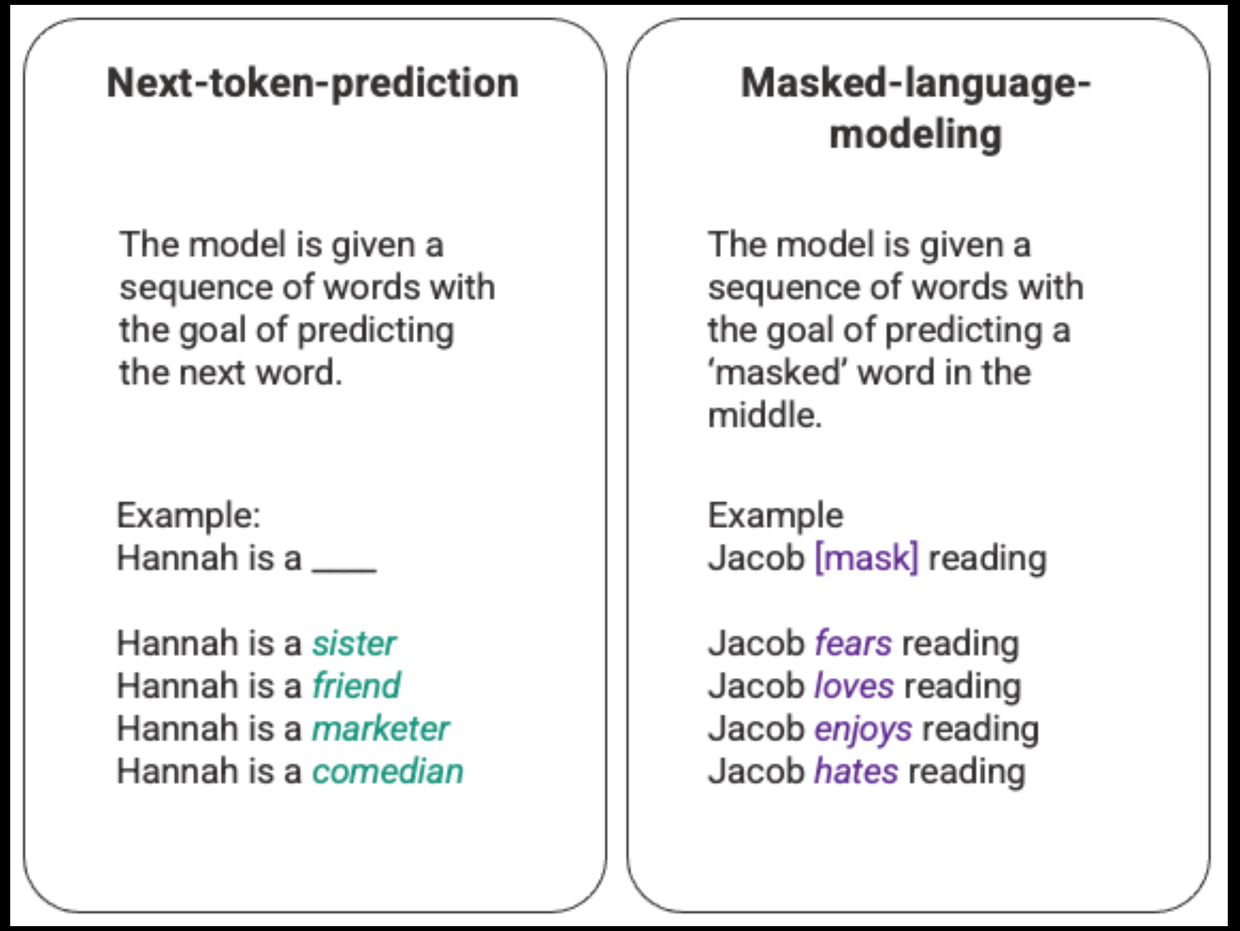
Embedded Space can use any NN to embed


Theorists
N-D Space
Signal
Background
Other
Signal
Signal
Background
Capture
Physics

We can replace the QCD theorist with a NN
(And it works better)

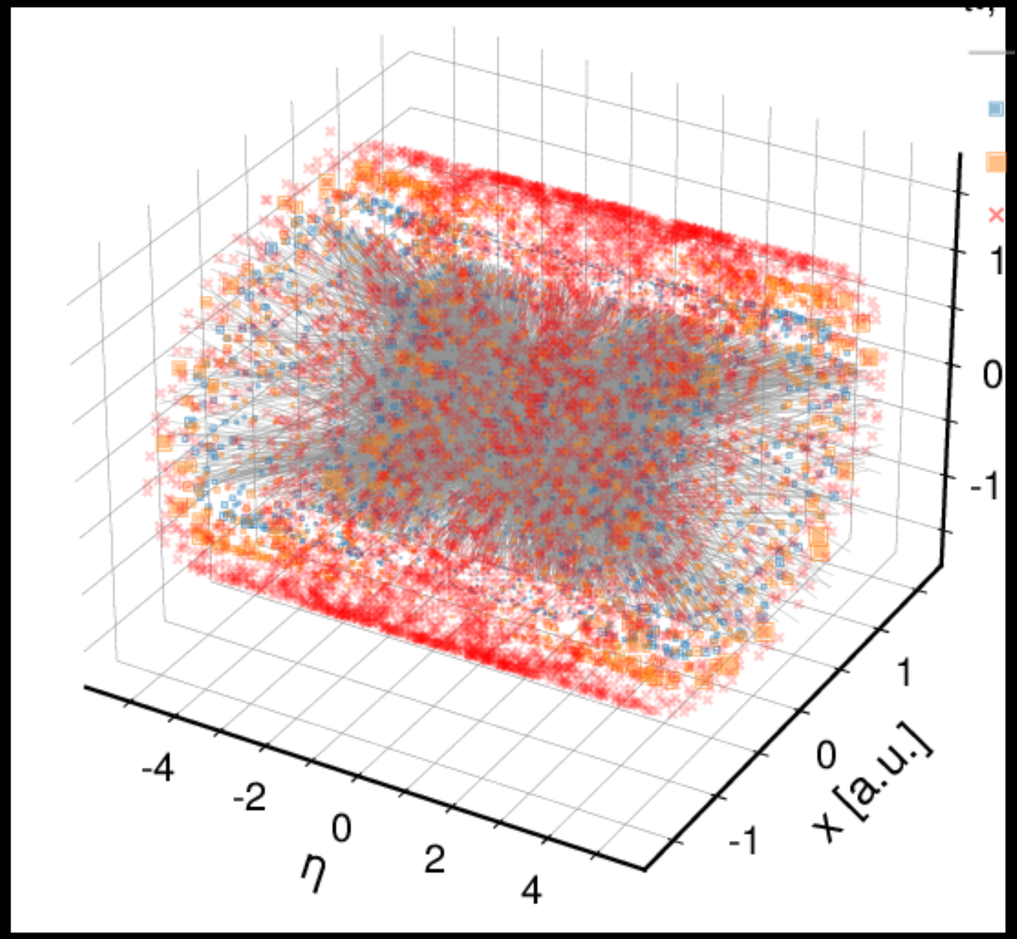


Masked language modelling

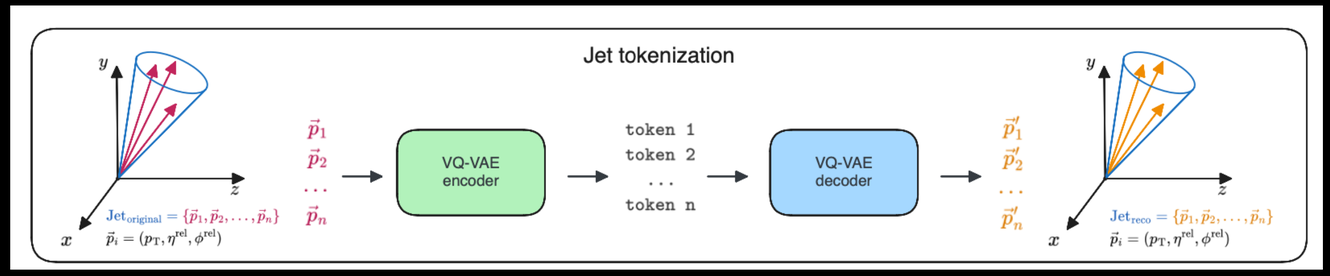


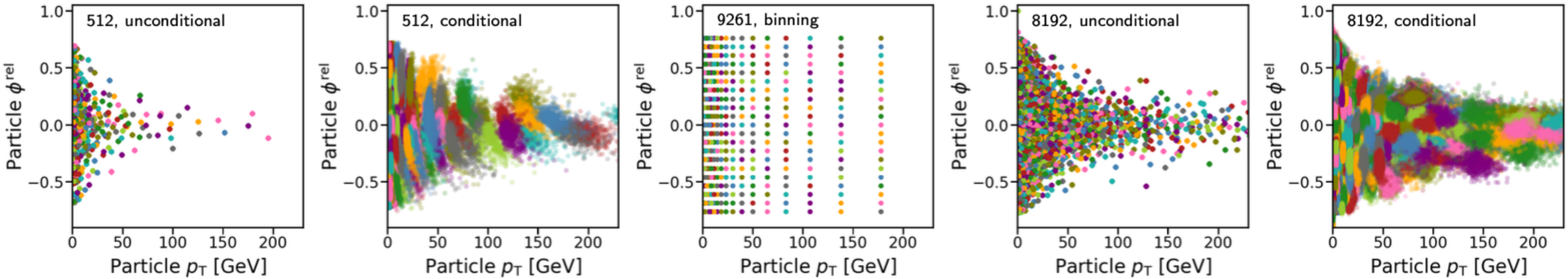
Self-supervised pre-training

Masked particle modelling



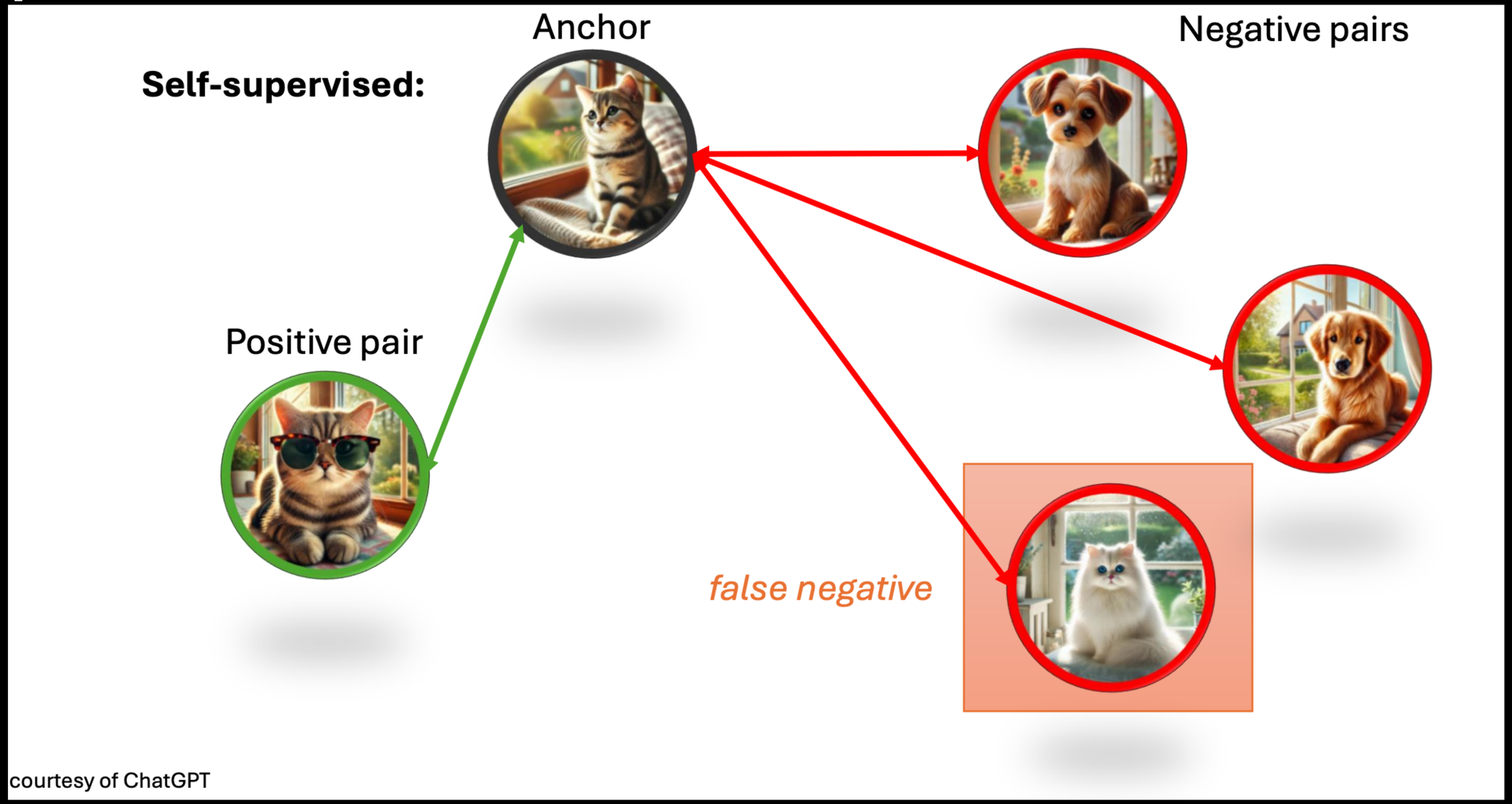
Tokenisation?





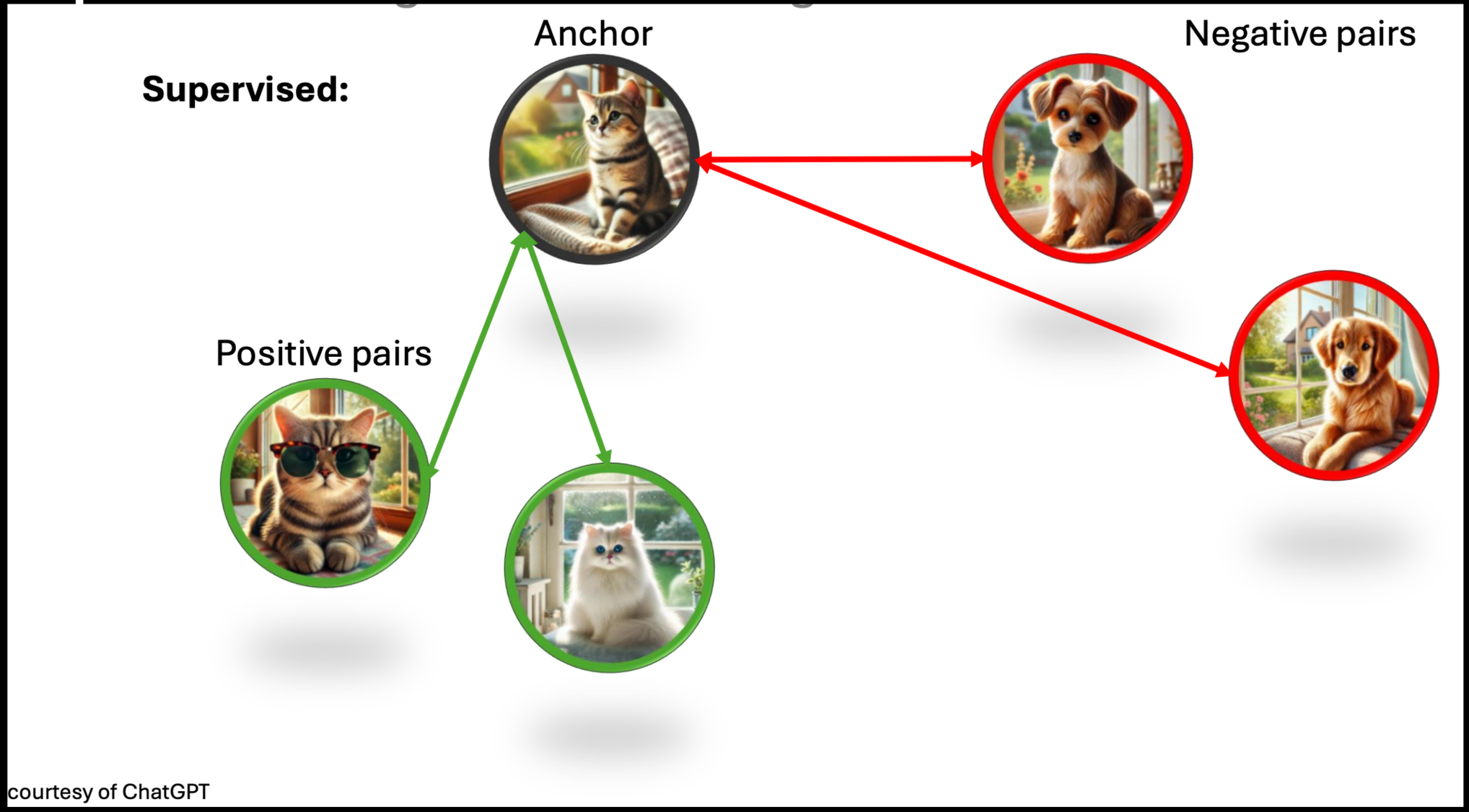
Neural embeddings for anomaly detection

Supervised extension to SimCLR



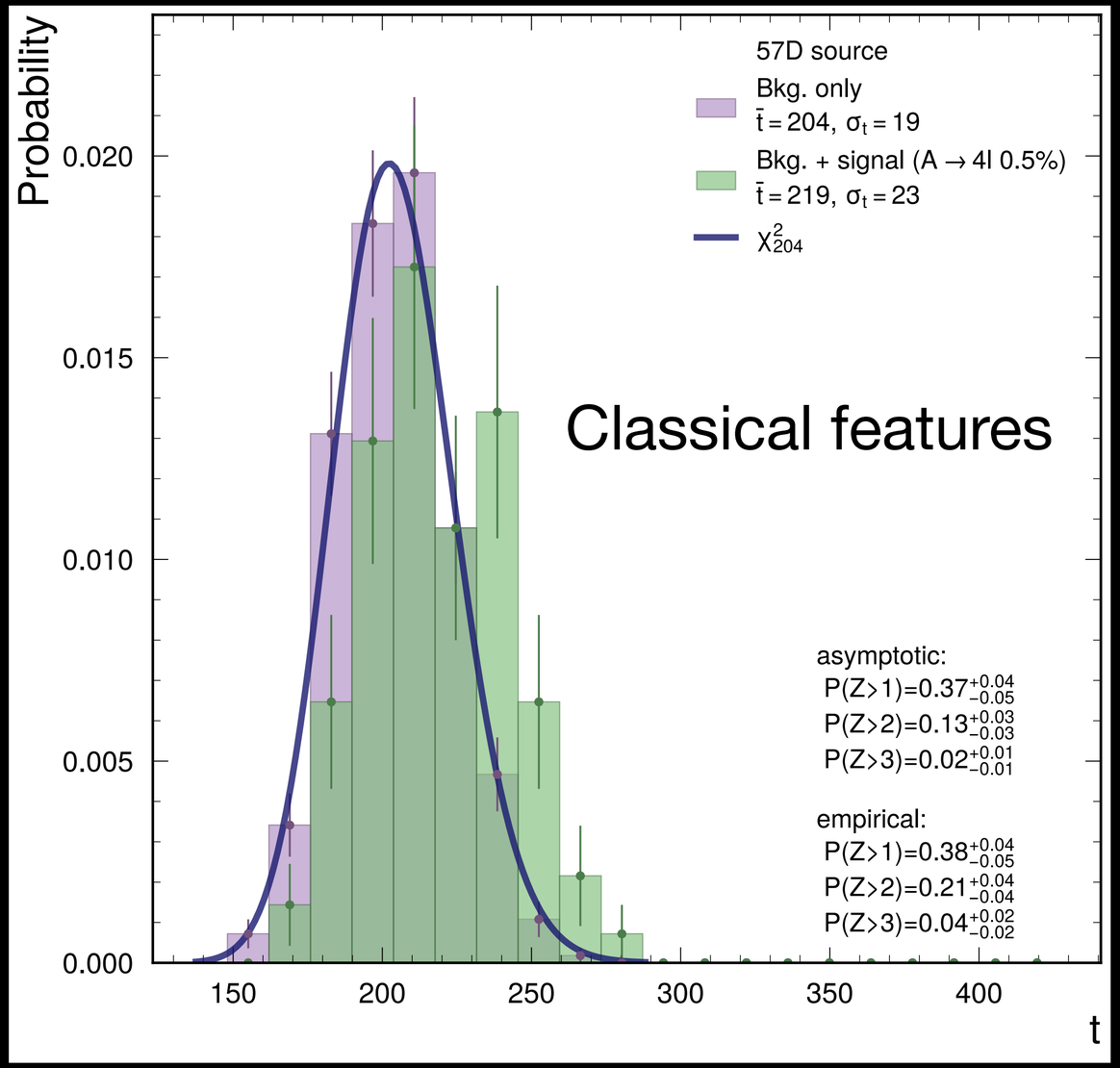
Neural embeddings for anomaly detection

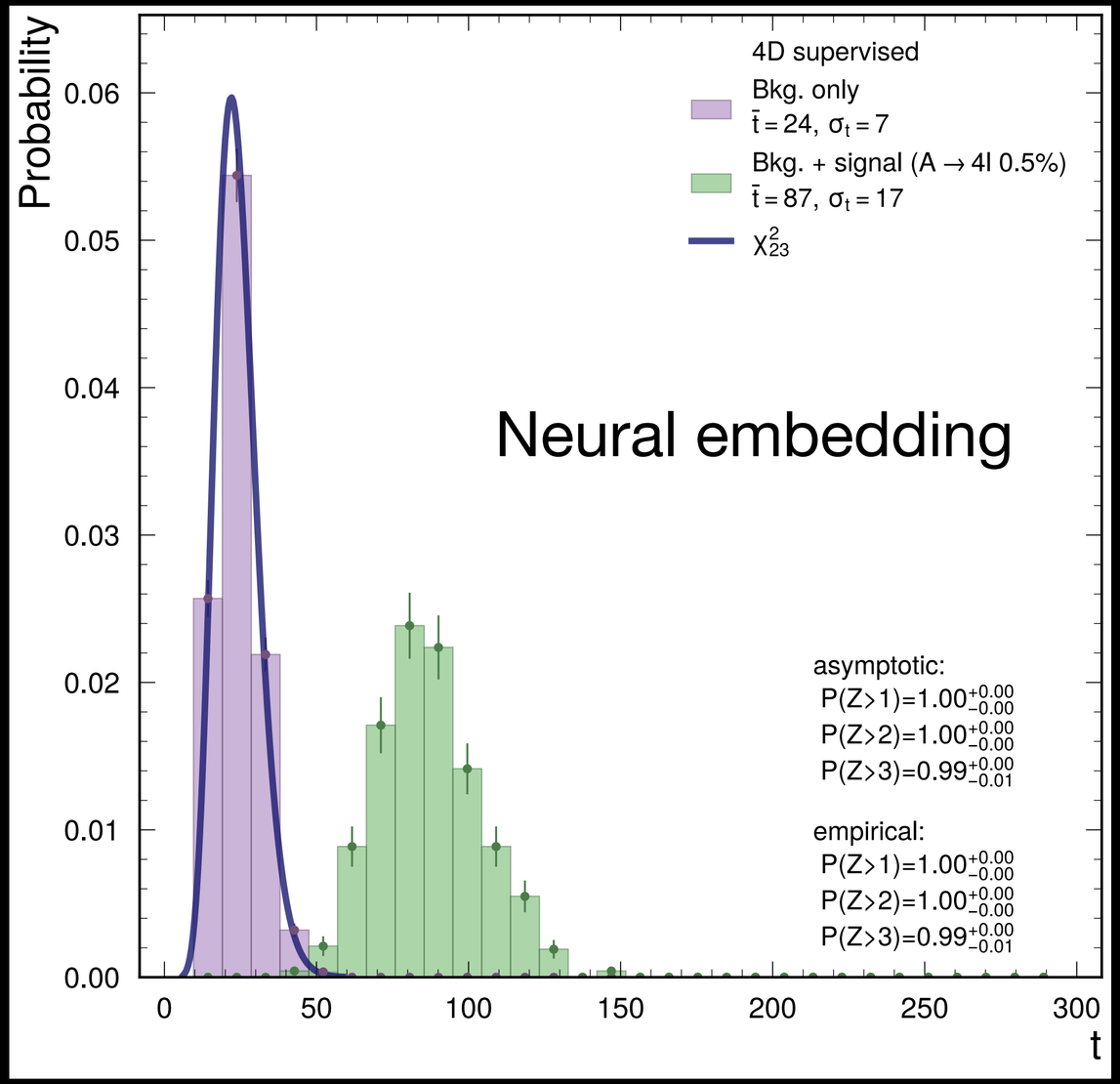
Supervised extension to SimCLR



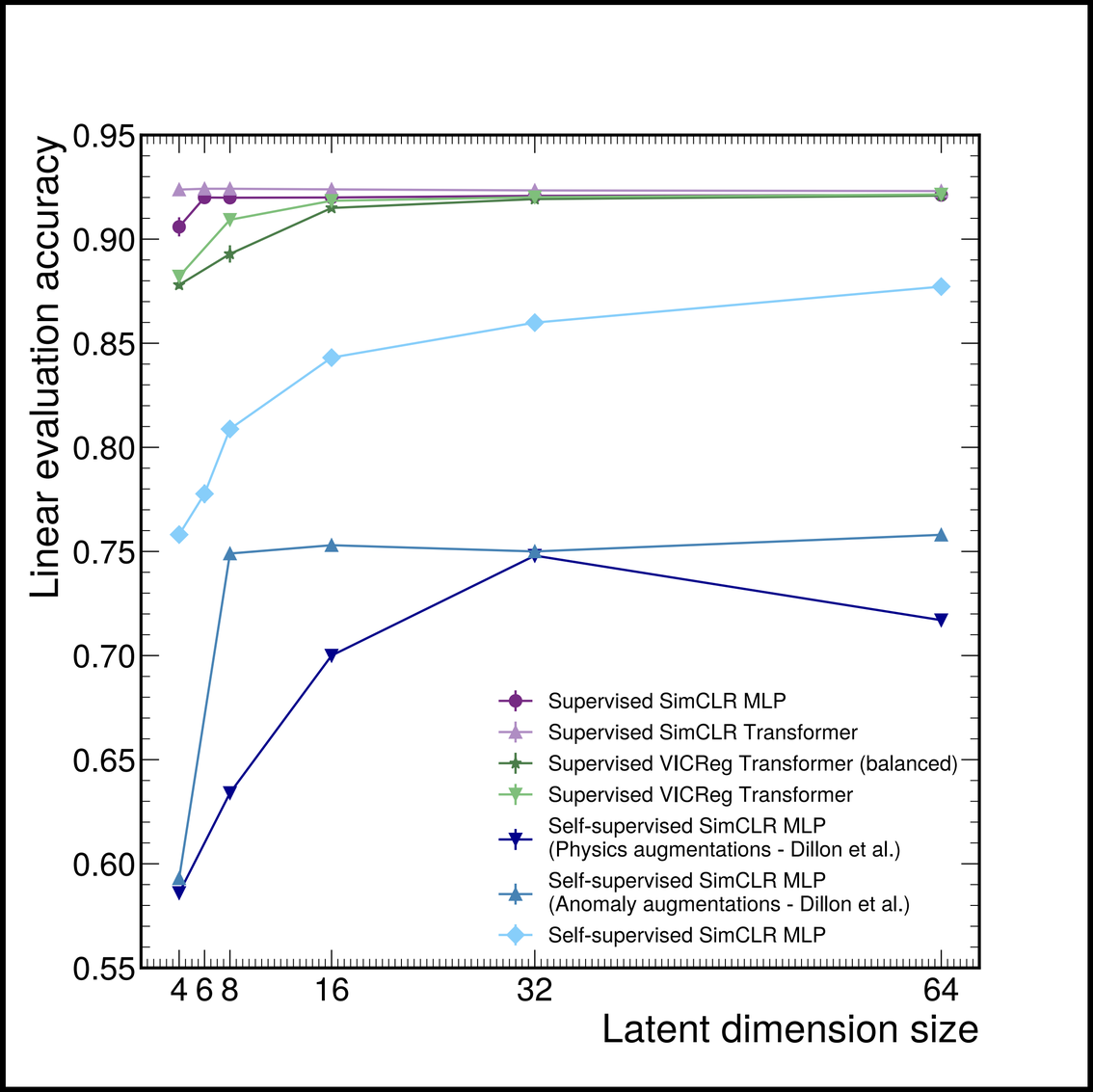
[2502.15926]

Neural embeddings for anomaly detection



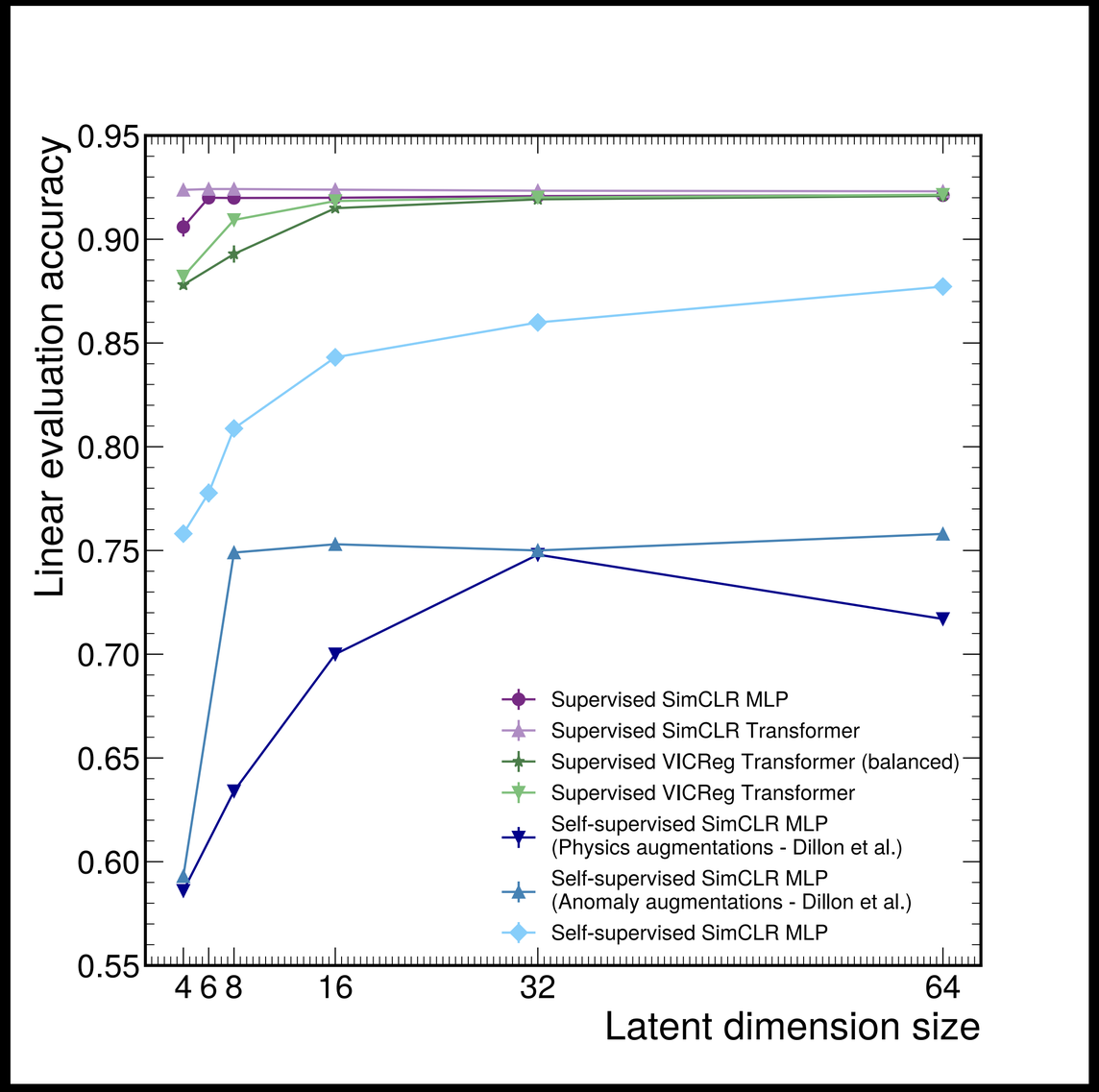


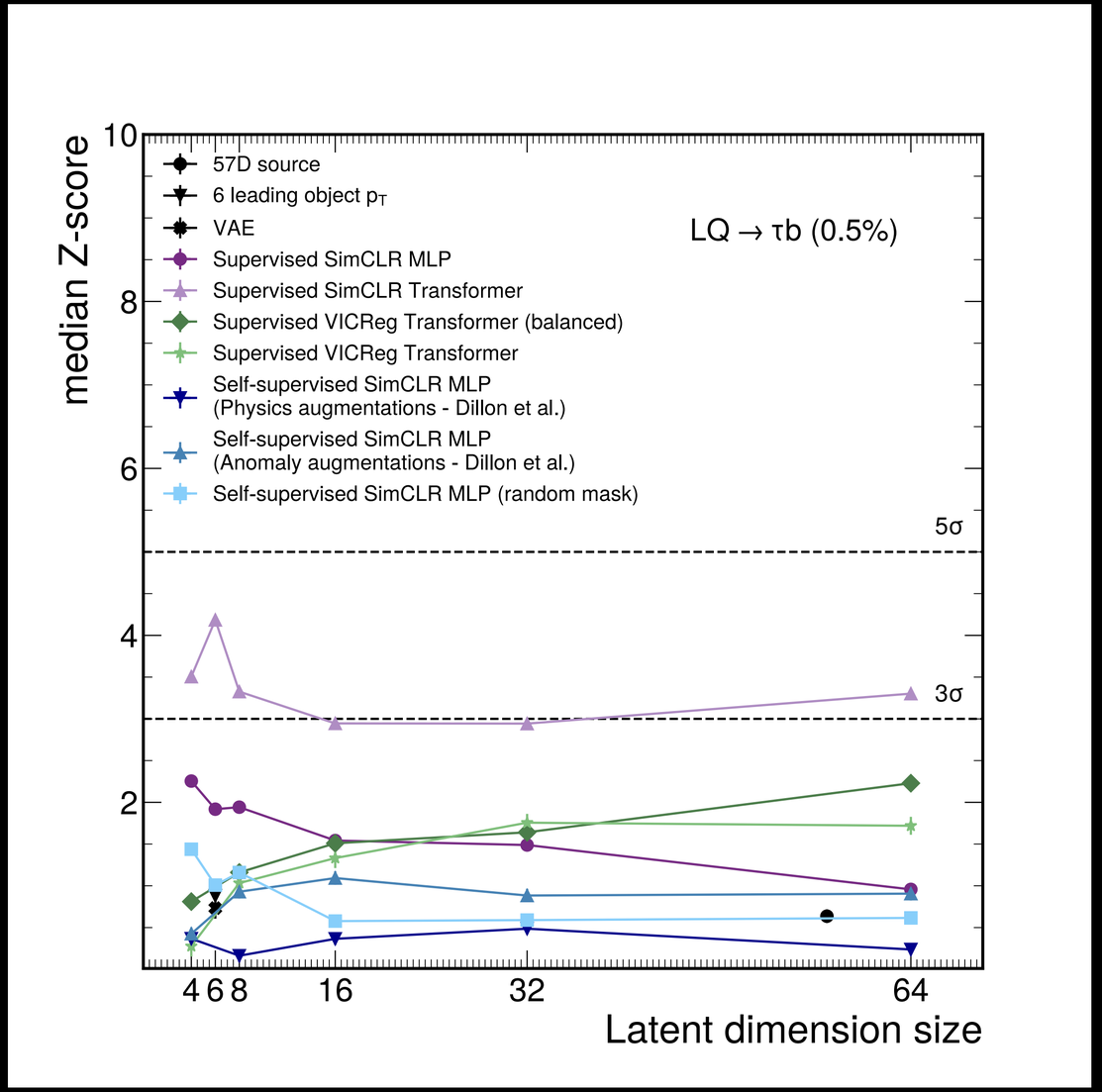
Neural embeddings for anomaly detection



[2502.15926]

Neural embeddings for anomaly detection

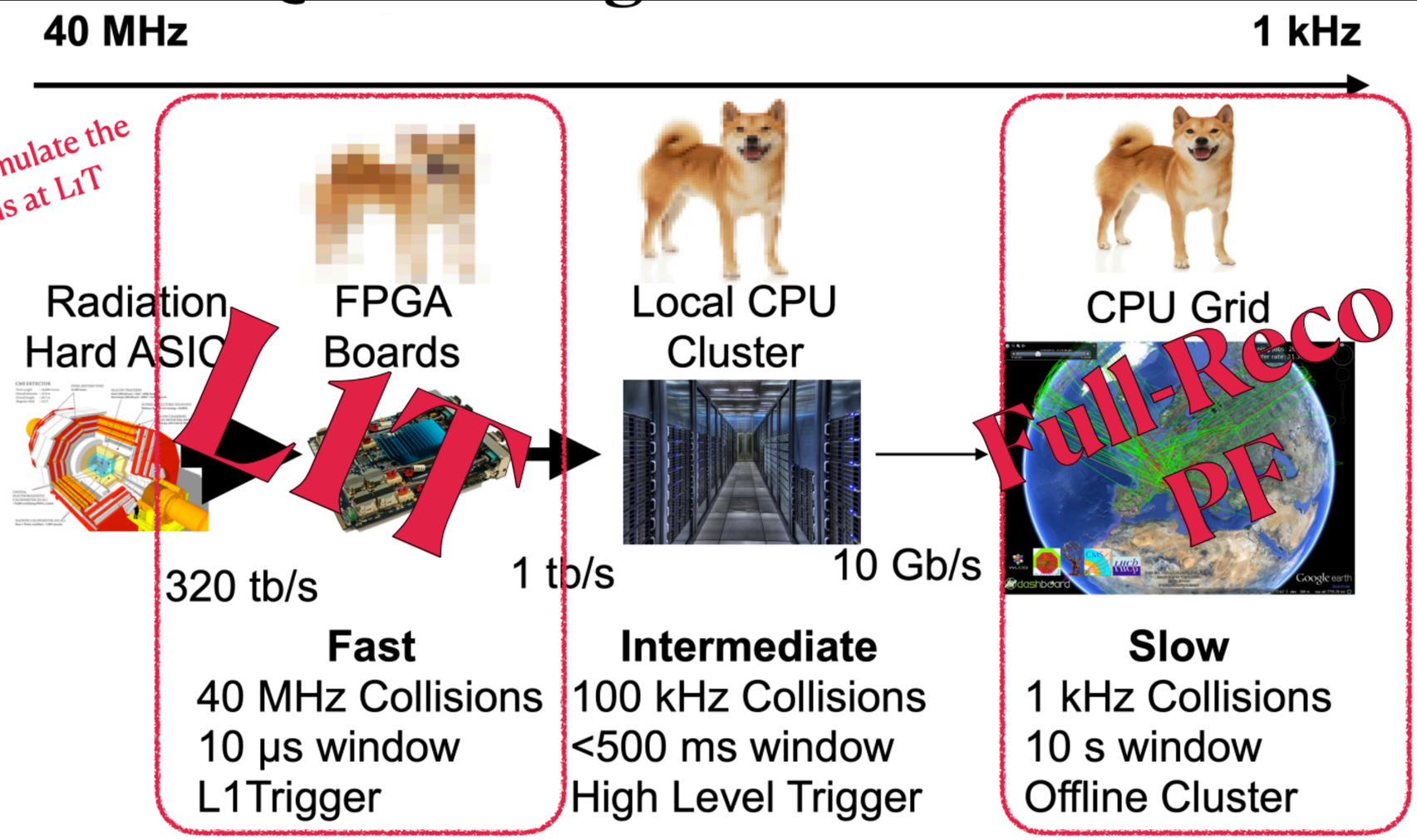




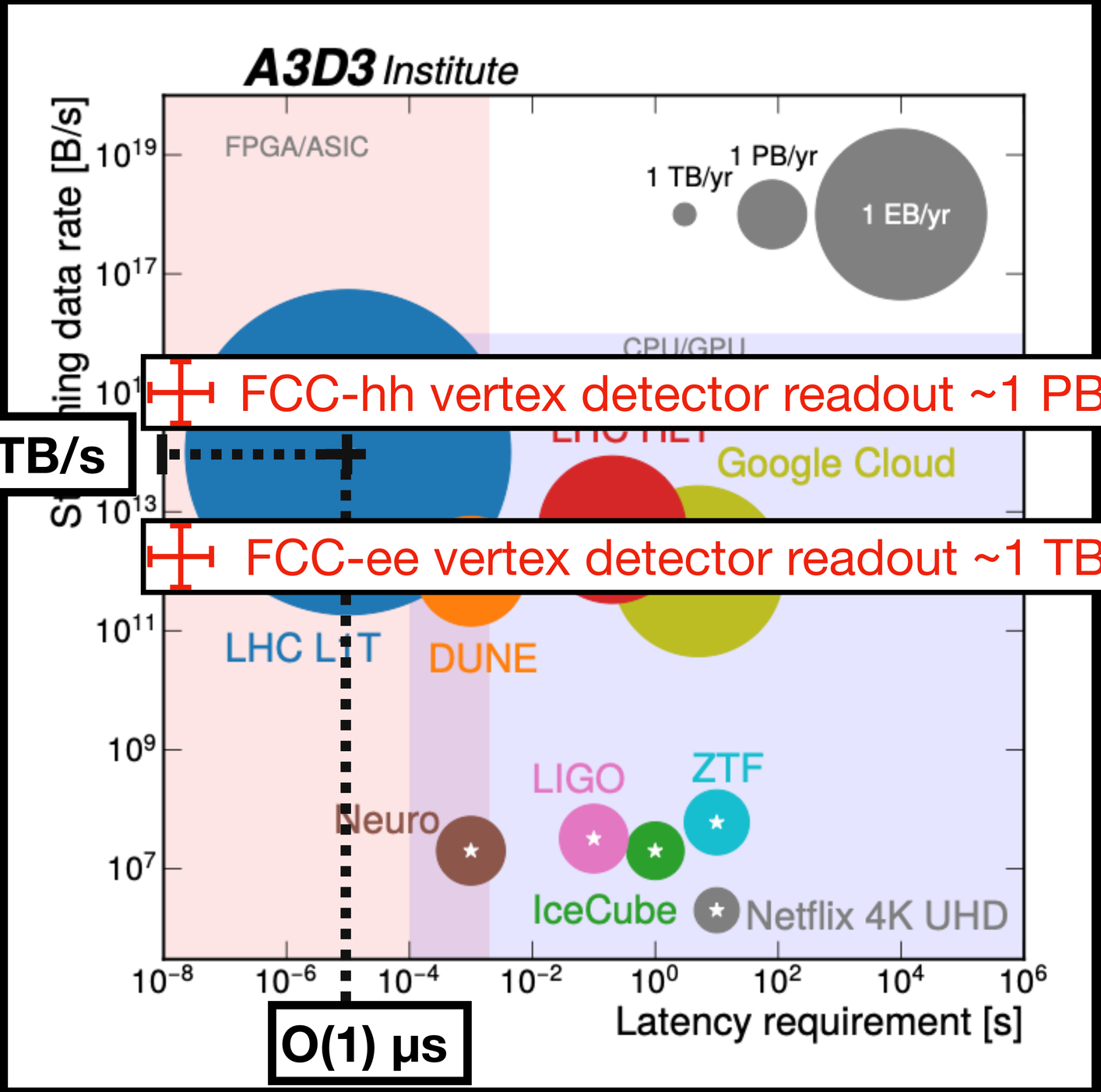
Collide-2V: A foundation model dataset

COLLIDE-2V

750 Million Dual-View LHC Event Dataset for Low-Latency ML



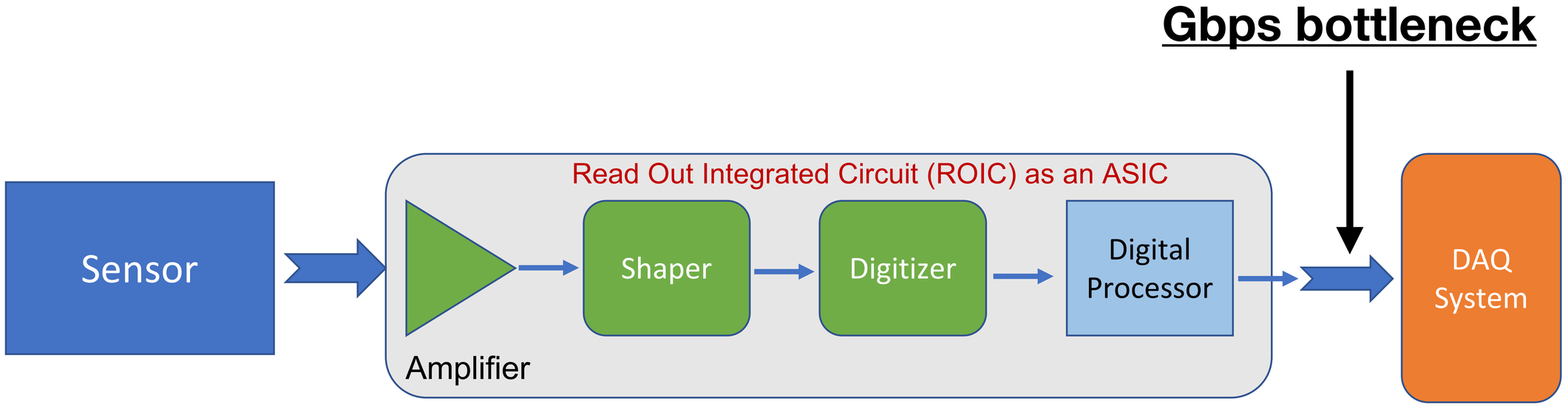
What looms on the horizon?


A3D3 Institute
FPGA/ASIC
CPU/GPU
1 TB/yr
1 PB/yr
1 EB/yr
Google Cloud
LHC L1T
DUNE
LIGO
ZTF
Neuro
IceCube
Netflix 4K UHD
FCC-hh vertex detector readout ~1 PB/s
FCC-ee vertex detector readout ~1 TB/s
O(100) TB/s
O(1) µs
How much data goes through system
Time to process
Latency requirement [s]

This readout technology doesn’t (yet) exist!

Ultimate precision at future colliders could be bottlenecked by our data acquisition systems

Data readout bottleneck



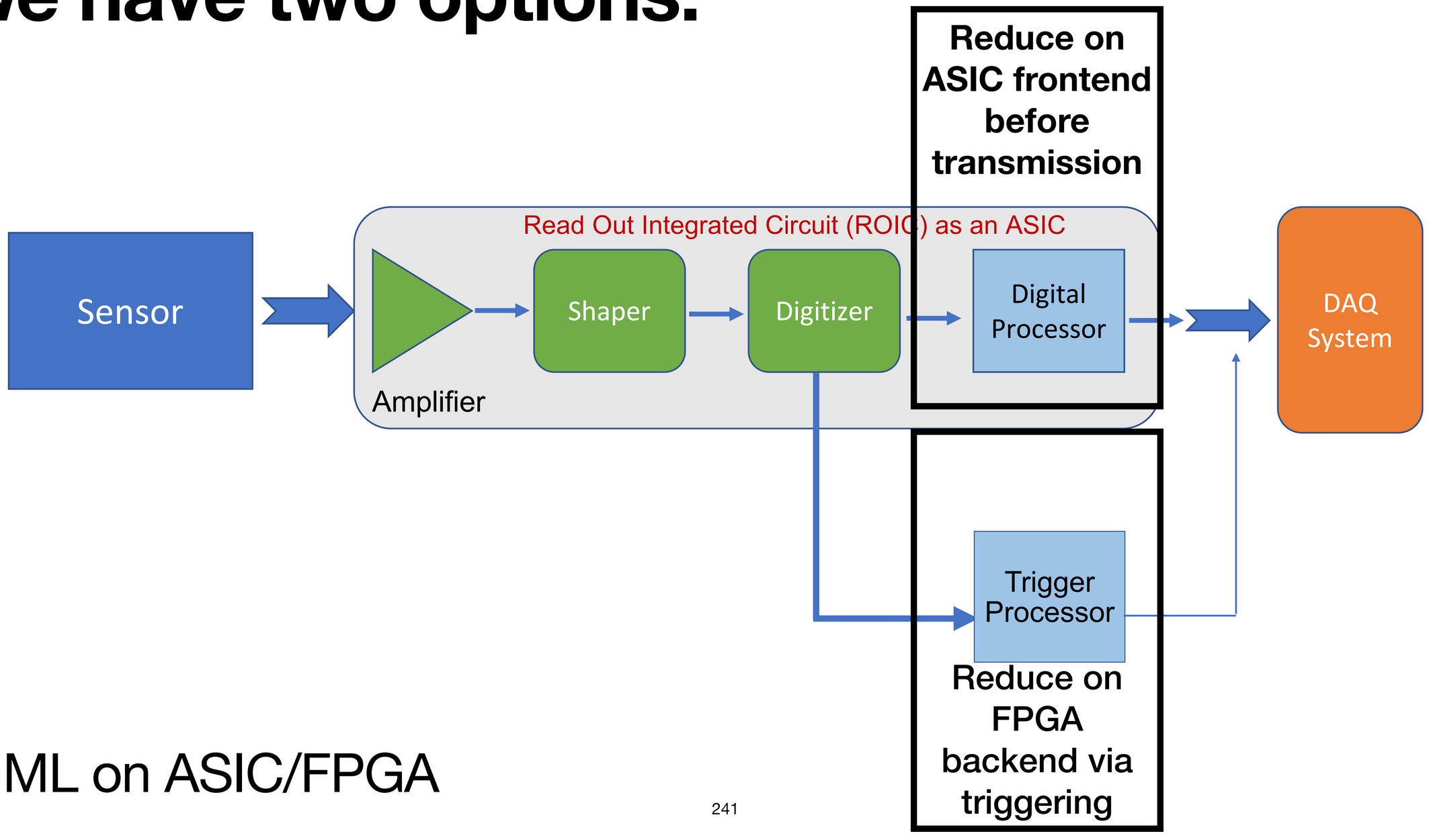
Example vertex detector inner layer from before (#241 FCC-ee):

32 bits pixel data in inner layer @ 200 MHz/cm^2

→ **24.4 Gbit/s**

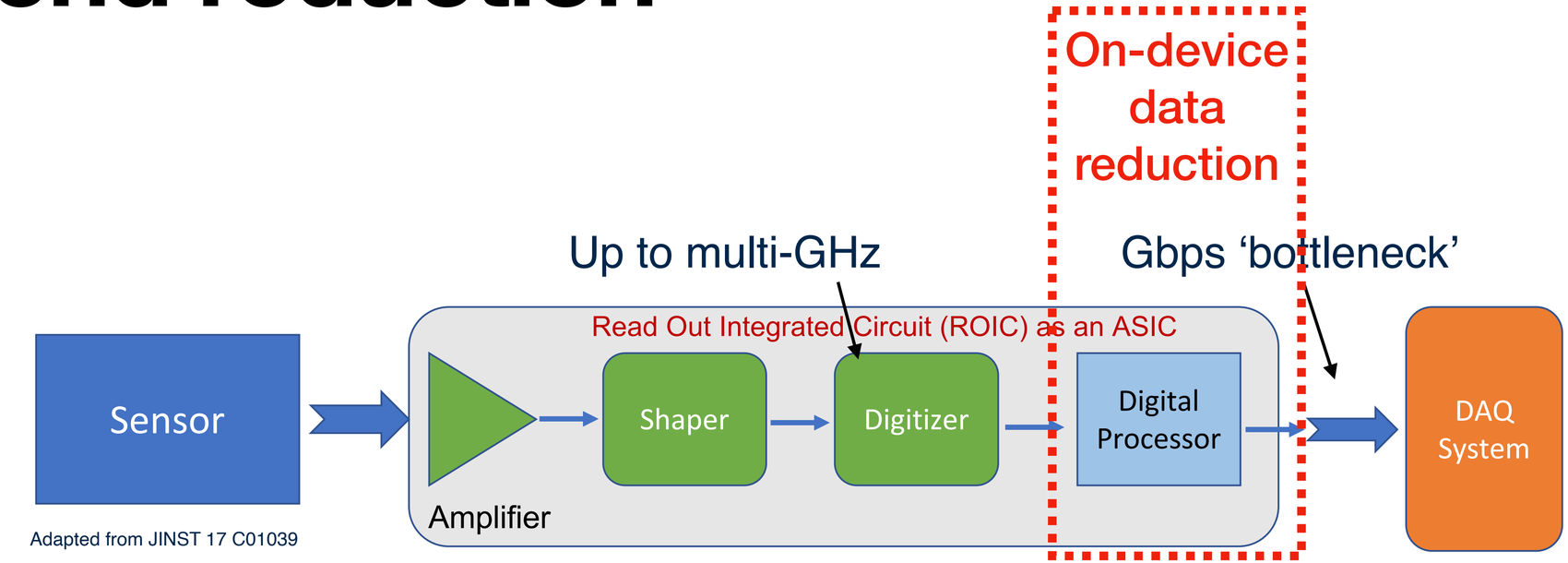
Current technology 3.2 - 6.4 Gbit/s

We have two options:



→ML on ASIC/FPGA

Frontend reduction



- highly granular detectors, background → high data transfer rate
- limited by **available bandwidth** of electrical and optical links
- 2 solutions: **frontend reduction** or **higher capacity** transfer

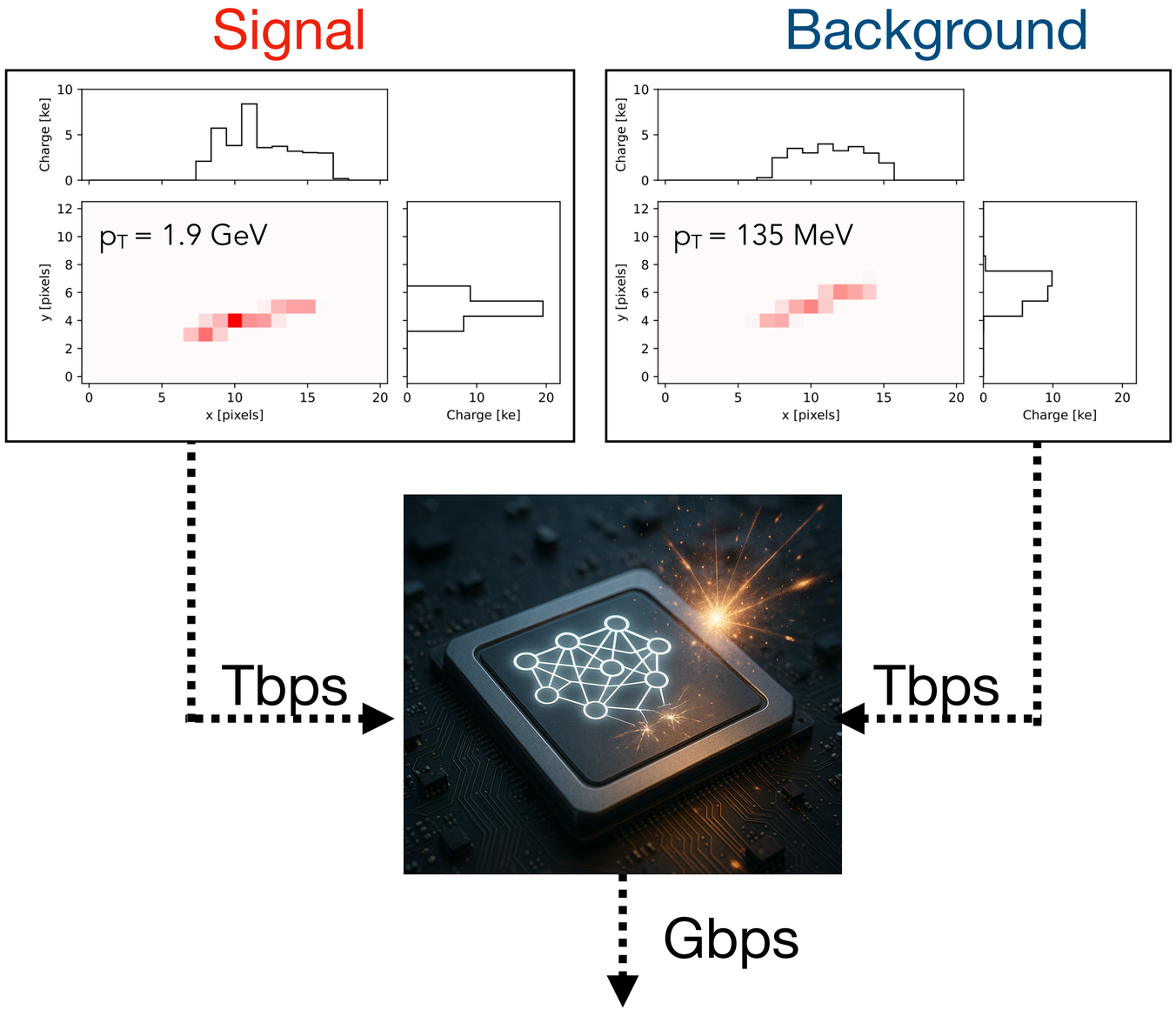
Intelligent frontends

Reduce on-detector

- **Subtle data patterns within single (pixel) layer! ML on ASIC** on-detector
  - filtering, and/or
  - featurizing
- **Challenges**

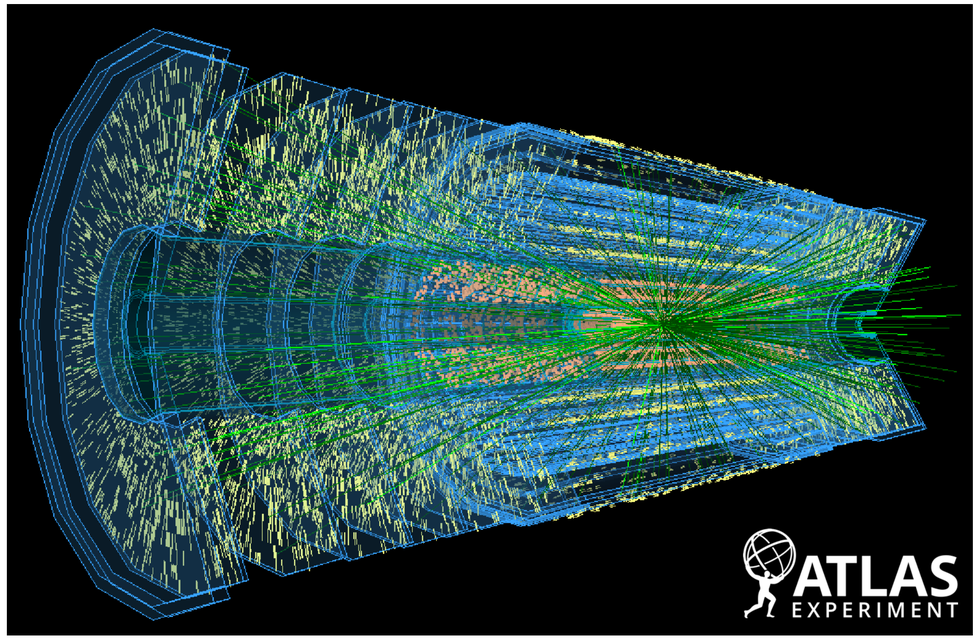
Metric	Simulation	Target
Power	48 mW	<100 mW
Energy / inference	1.2 nJ	N/A
Area	2.88 mm^2	<4 mm^2
Gates	780k	N/A
Latency	50 ns	<100 ns

  - Radiation hardness

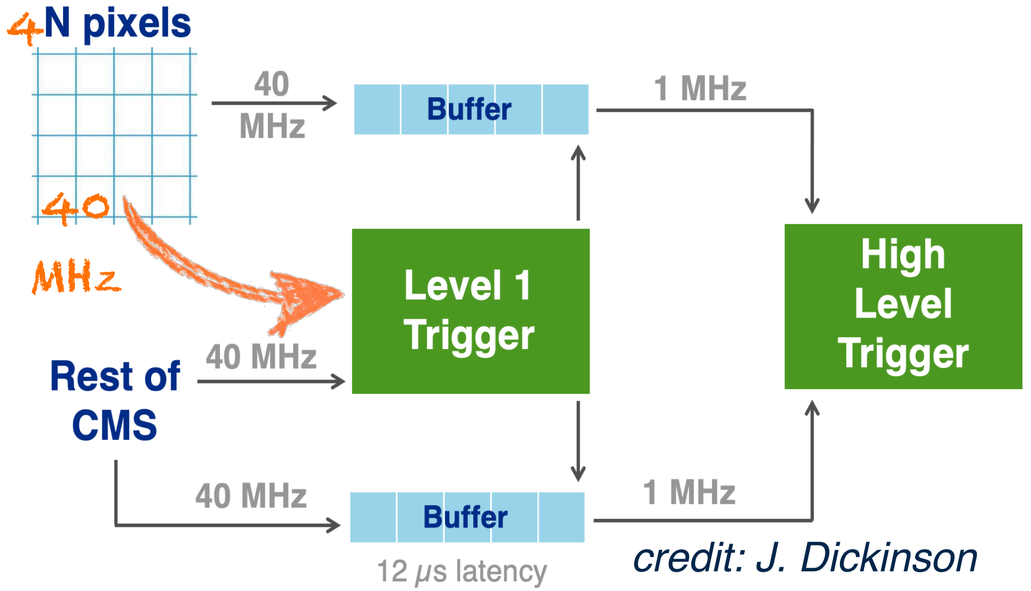


Smart Pixels

https://fastmachinelearning.org/smart-pixels/



- Reduce silicon data via **in-pixel intelligence**
  - **frontend filtering**: discard low-p_T track data (< 2 GeV)
  - **feature extraction**: Extract particle position and angle in pixel front-end ASICs from charge in single pixel layer
- Bandwidth savings of 57-75%!



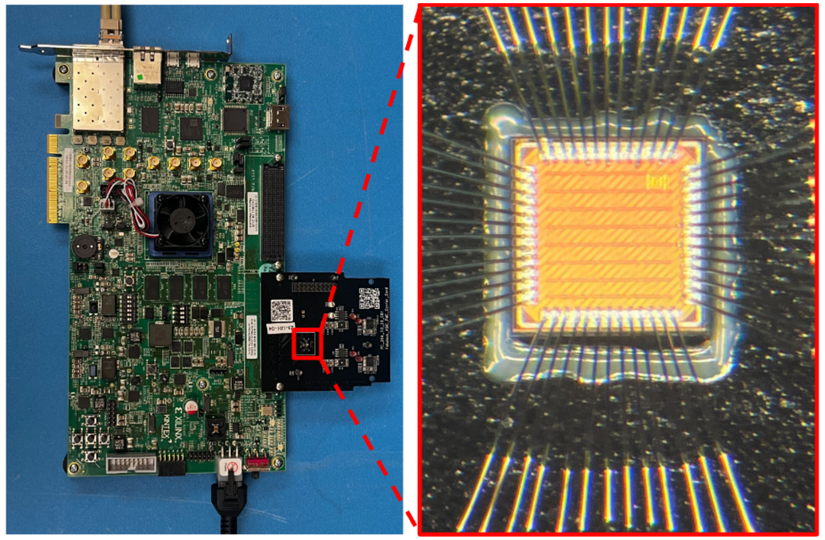
credit: J. Dickinson

Reconfigurable logic in ASIC design

The Embedded FPGA framework

- Pathway to implementing ML "at source”
  - Fully reconfigurable logic on detector frontend
- Open source (FABulous, OpenFPGA)
  - potential to apply to variety of subsystems/ fields (SuperKEKB, FCC-ee, DUNE, free-electron lasers)

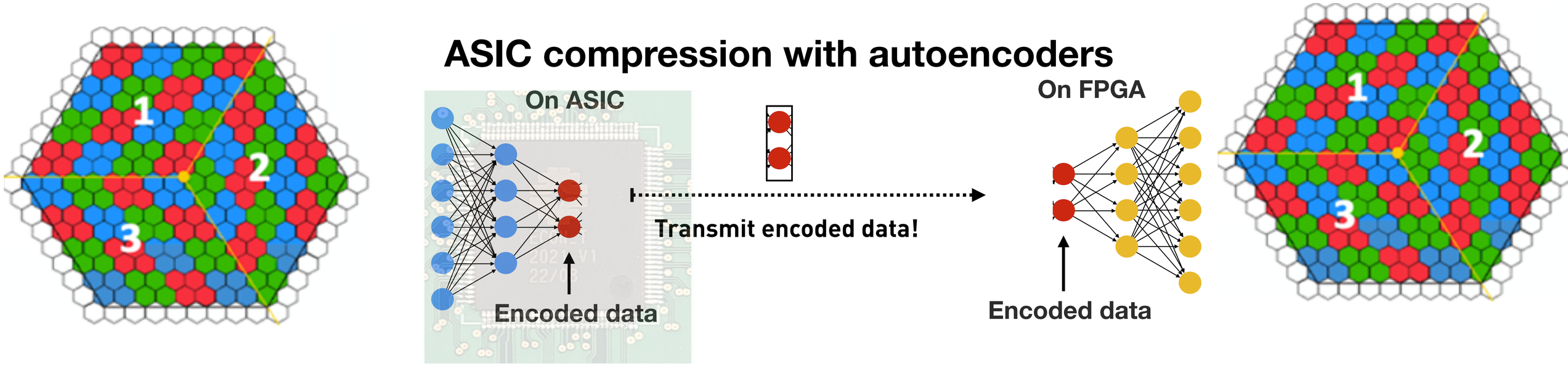
28nm CMOS ASIC (1x1mm)



Gonski et. Al

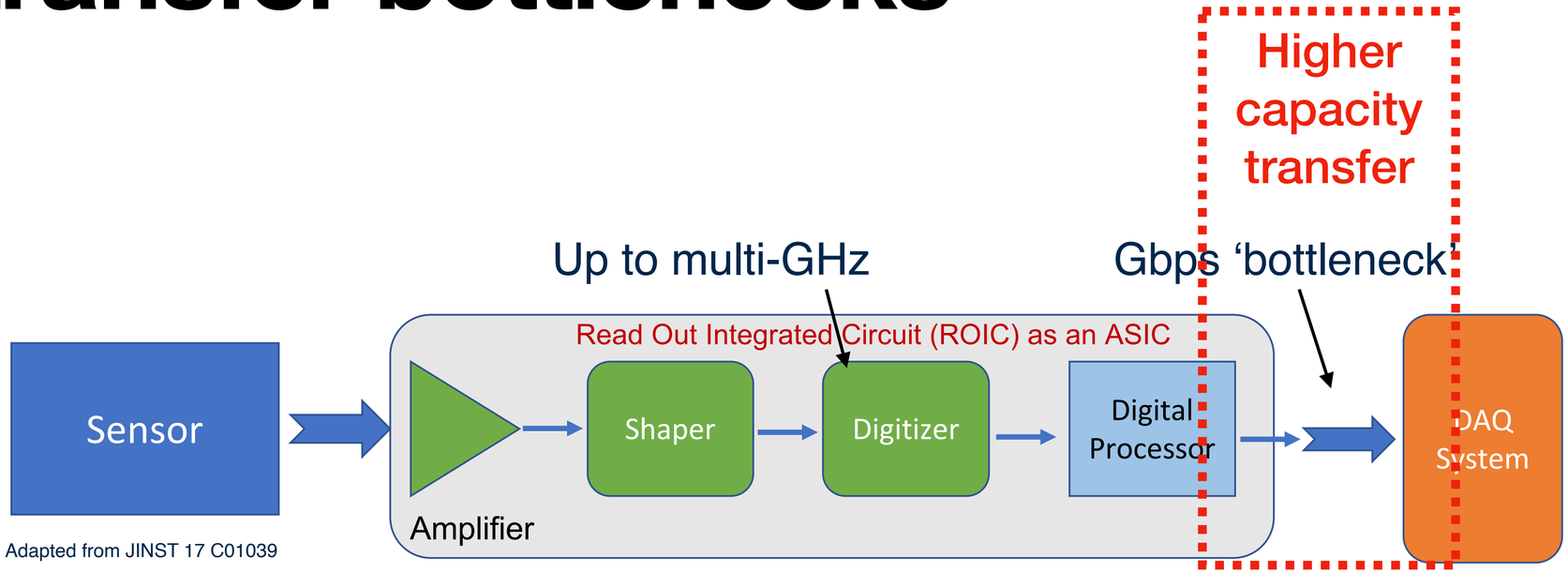
ML compression: Calorimeter data concentrator

10,000 ECONs with ML inside going live in HL-LHC



Di Guglielmo et Al, IEEE TNS 68.8 (2021)

Data transfer bottlenecks

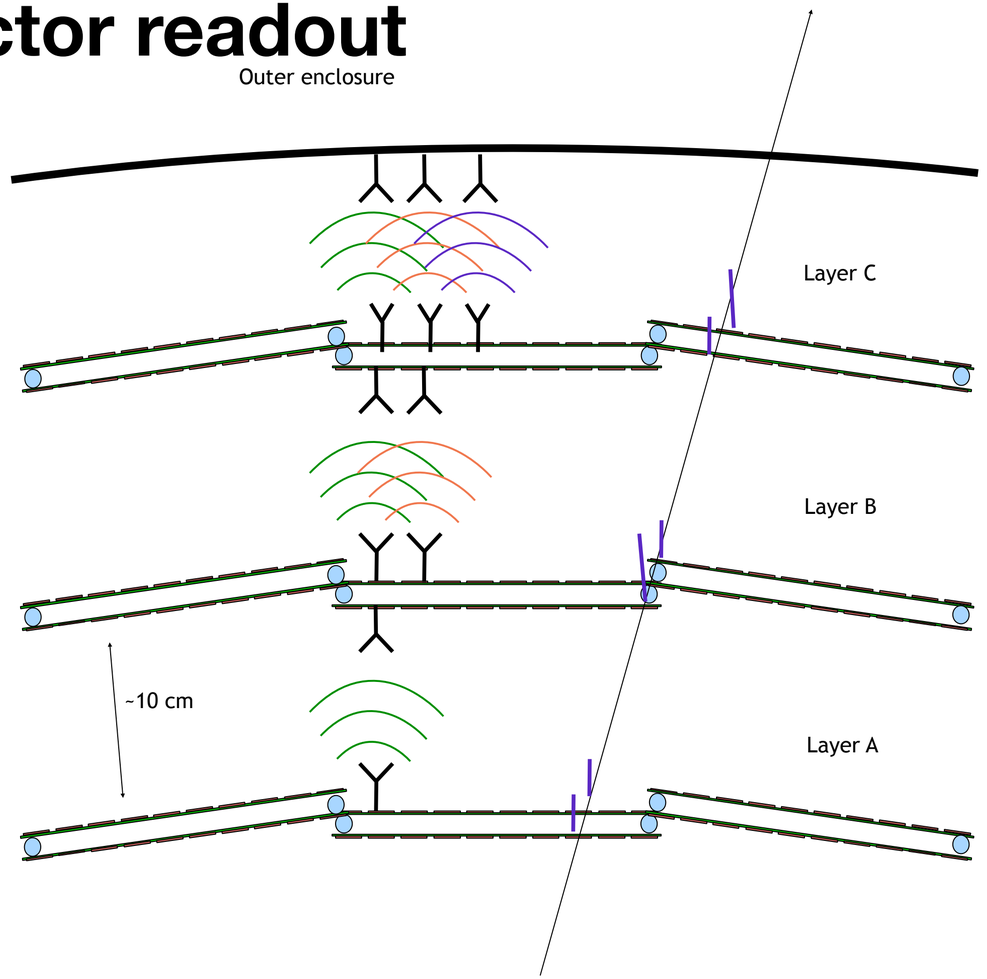


- 2 solutions: **frontend reduction** or **higher capacity** transfer

Wireless data transmission

Reducing material budget for detector readout

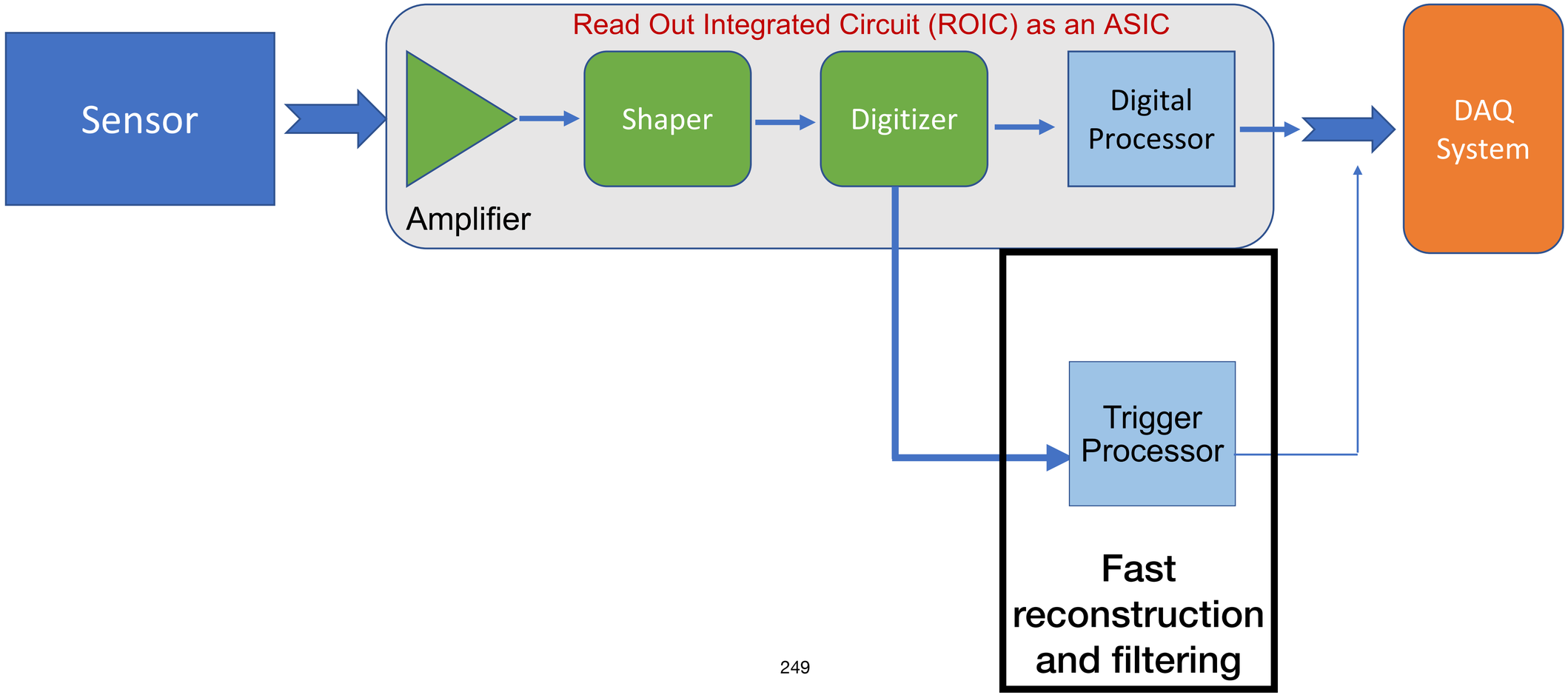
- Increase Gbps w/o increasing material
- Send single signal to several receivers, saves cabling
- Cost reduction, simplified installation/ repair, reduction in dead material
  - Especially important for future tracking detectors



Few Gbps possible with 802.11ac/ad WiFi!

Wireless Allowing Data And Power Transmission (DRD-7.1c)

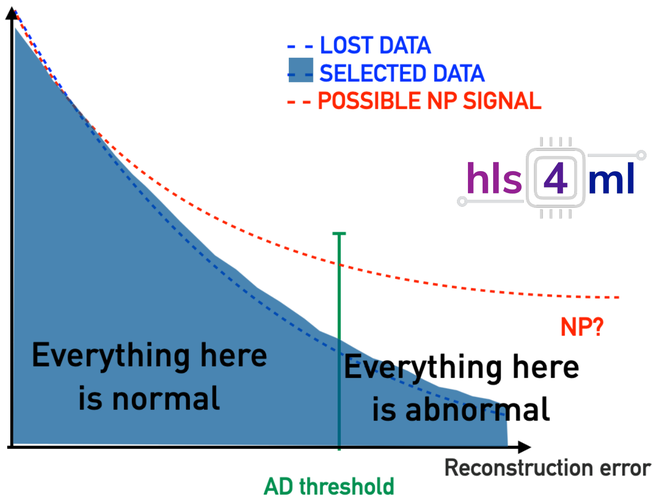
Intelligent back-ends



2024: Neural hardware triggers

CMS:

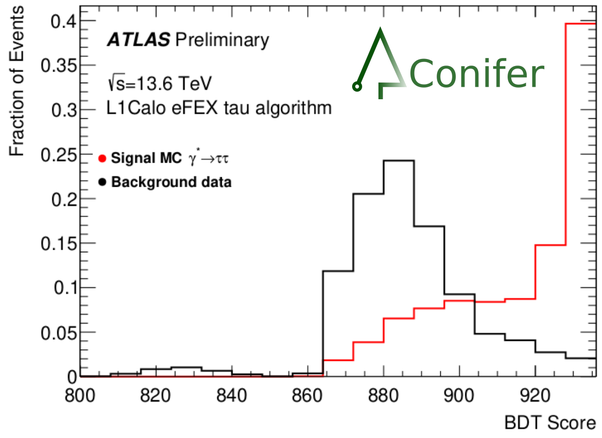
- Anomaly detection in 50 ns



CMS DP2023_079

ATLAS:

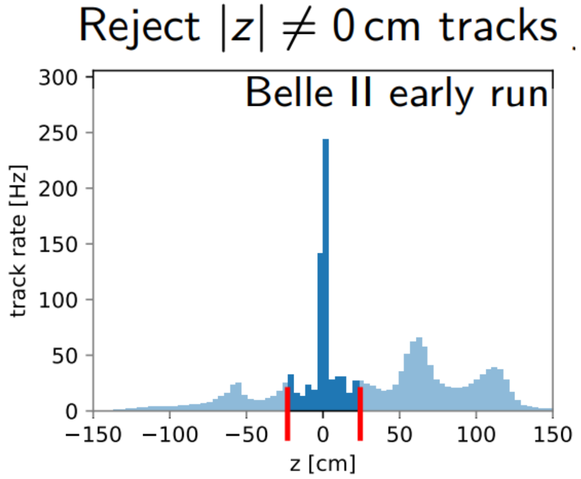
- BDT selecting candidate τ events in <100ns



L1CaloTriggerPublicResults

Belle-2:

- Neural track trigger since April 2021

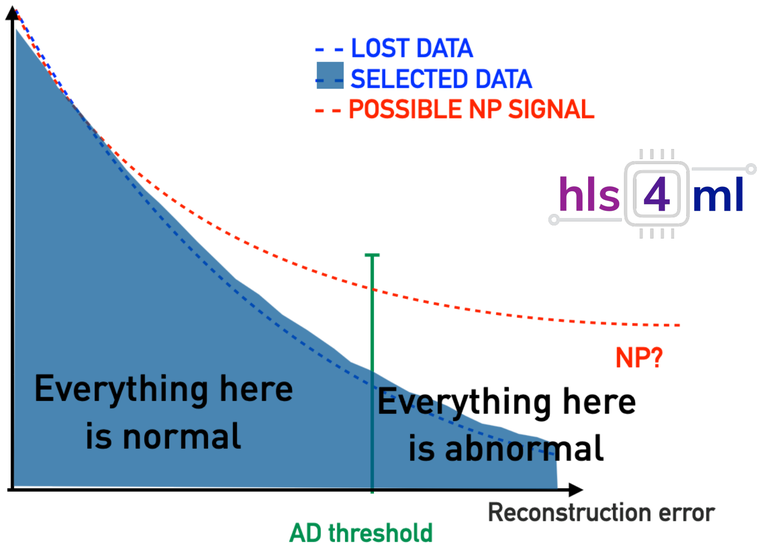


arxiv:2402.14962

2024: Neural hardware triggers making decisions in LHC experiments!

CMS Experiment:

- Anomaly detection in 50 ns
- 300 events/second

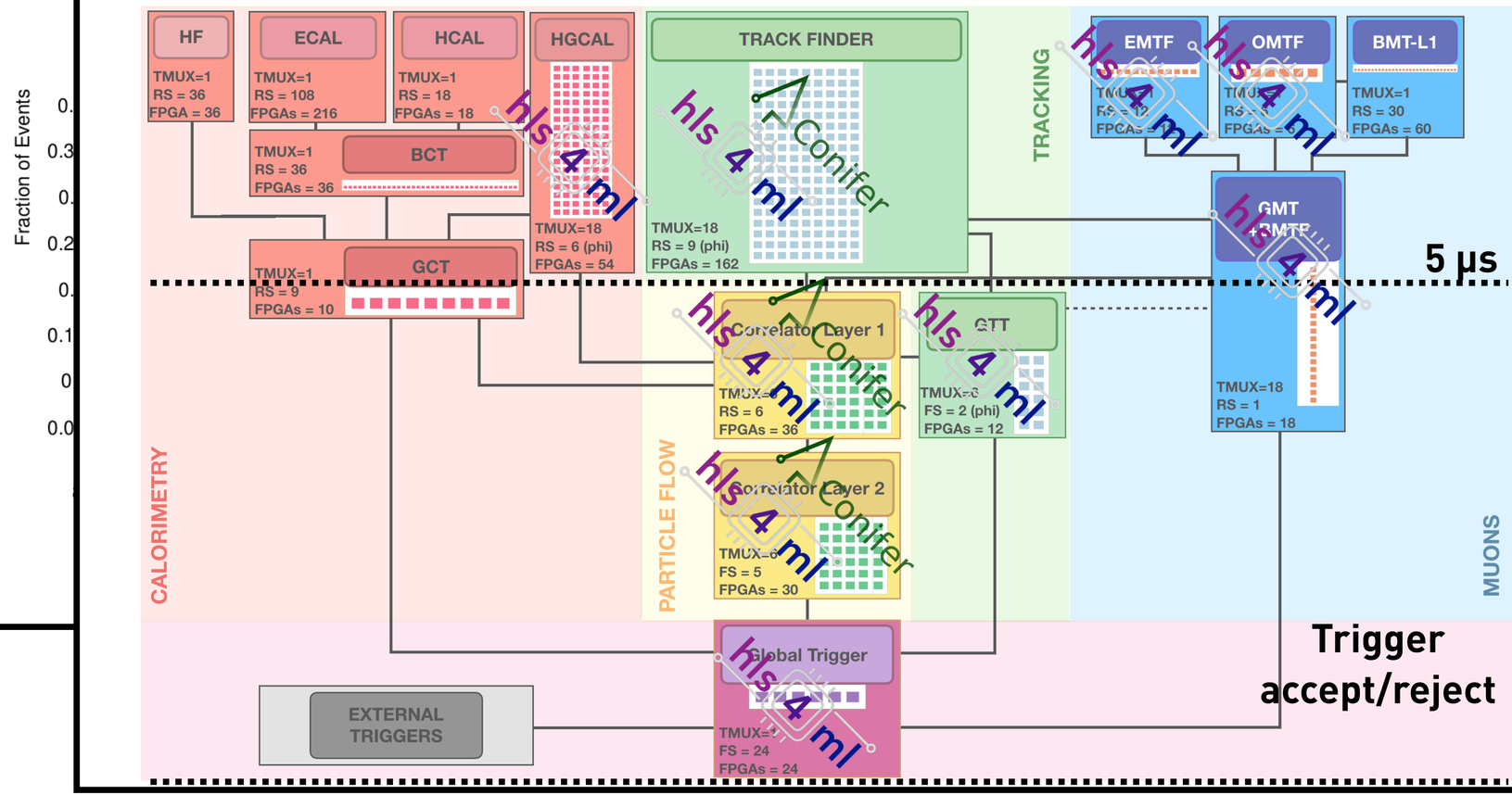


CMS DP2023_079

ATLAS Experiment:

- BDT selecting candidate τ lepton events in <100 ns

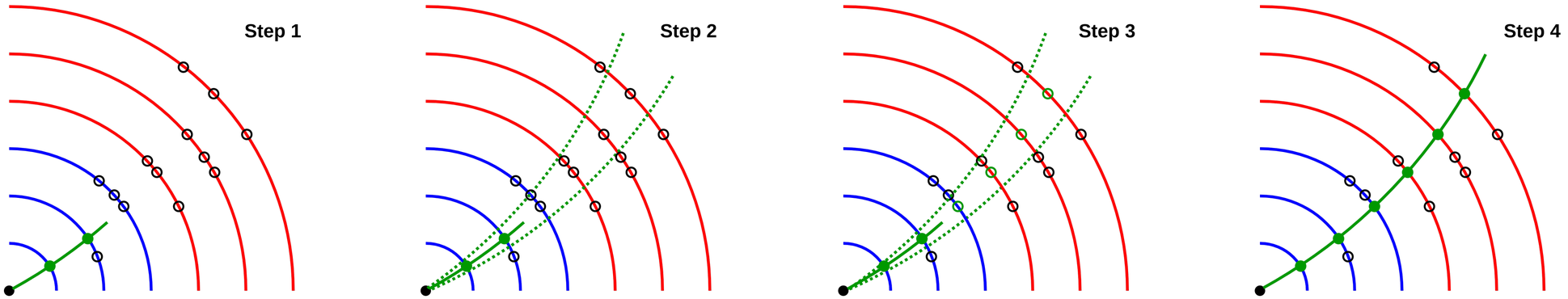
2031: ~20 billion inferences/s during HL-LHC



C. Brown, Vertex2023

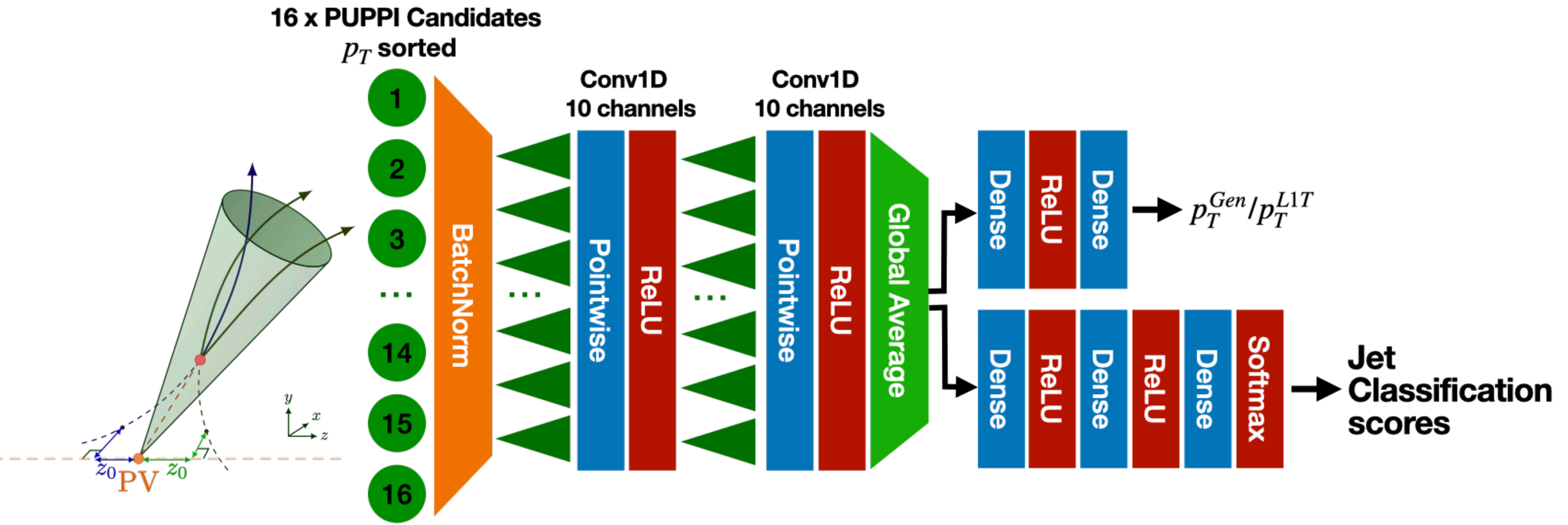
Real-time tracking

- HL-LHC CMS will read out tracks with p_T>2 GeV and run L1 track reconstruction
  - reduce rate by x10 by filtering low momentum hits
  - Offline like reconstruction: track seeding, building and Kalman Filter to identify final track candidates and determine track parameters
  - Track quality with BDT implemented with Conifer



- ….. **in 3.4 μs!**

Hardware jet building and flavour tagging



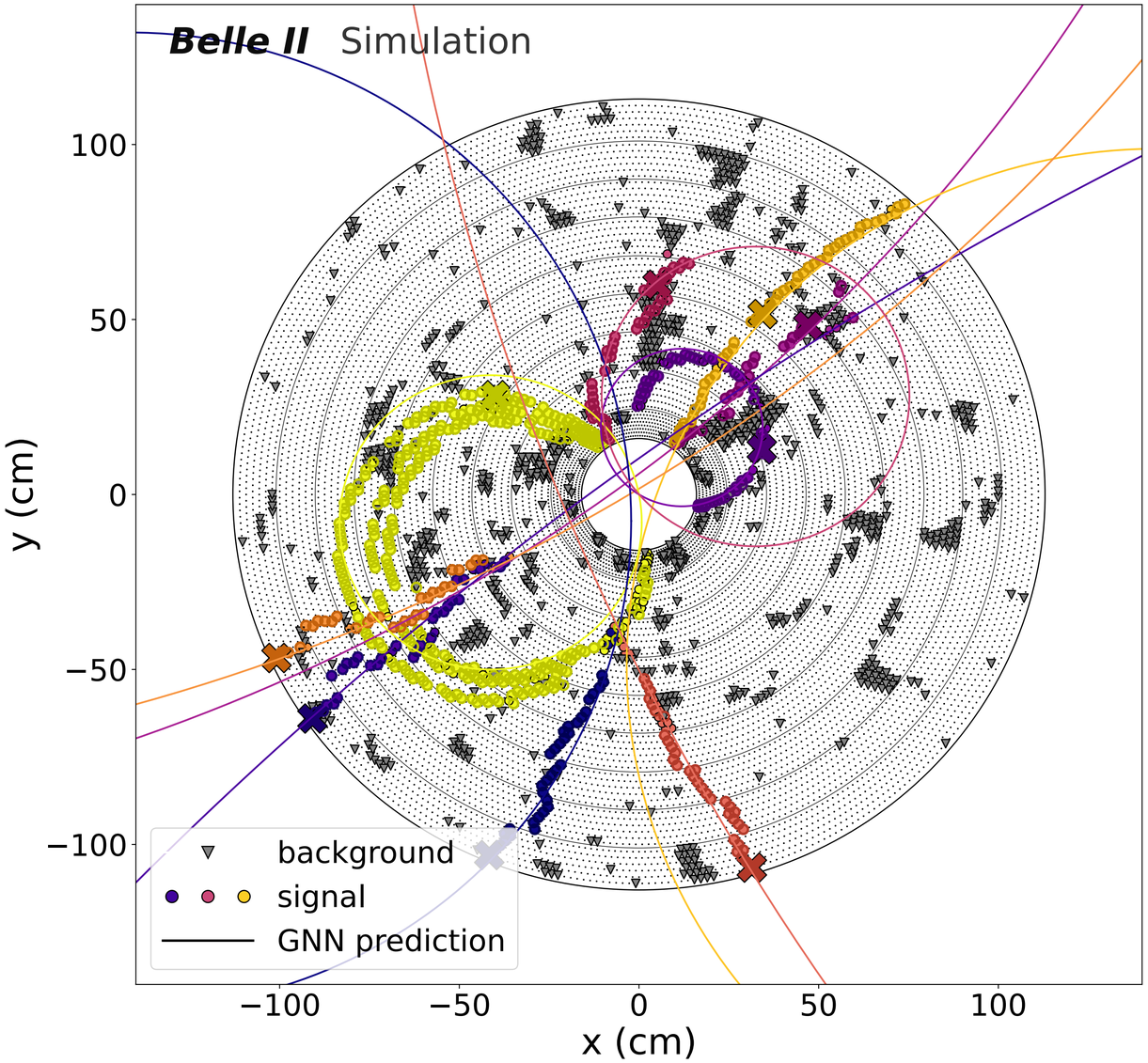
HL-LHC CMS will run with

FPGA track finding, particle flow, jet clustering and flavour tagging in < 12 μs

Real-time track triggers

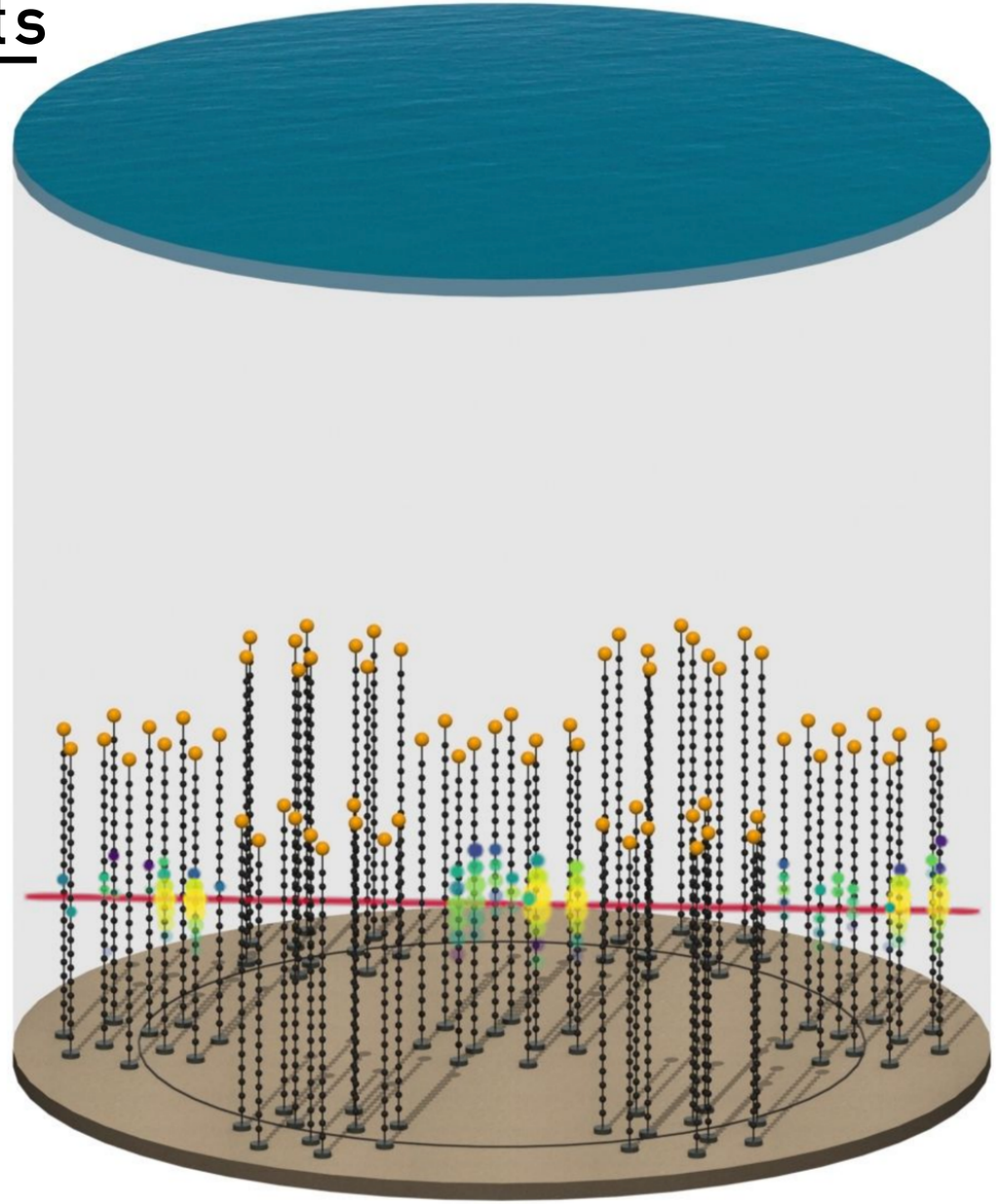
BELLE-II

- Belle-2 uses neural track hardware trigger for reconstruction of vertex, and azimuthal and polar angles of single particle track
- ML for GNN-based offline tracking
  - Work ongoing on bringing this to the hardware trigger



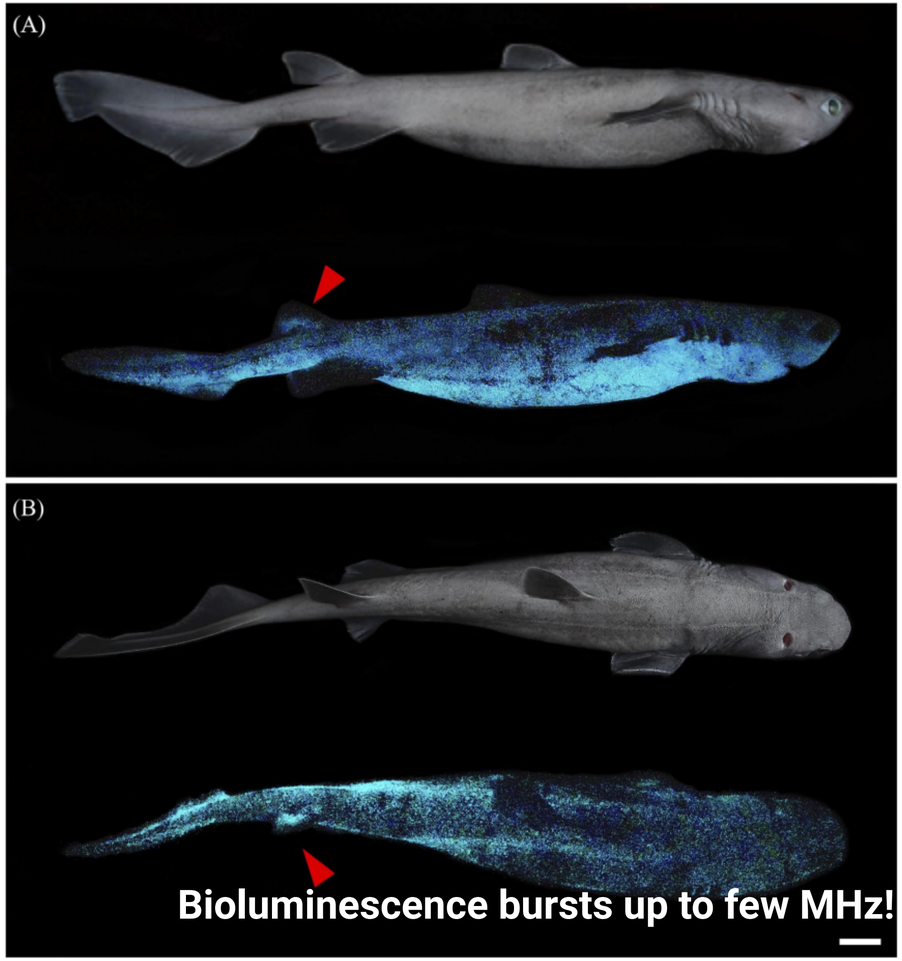
F. Capel et al.

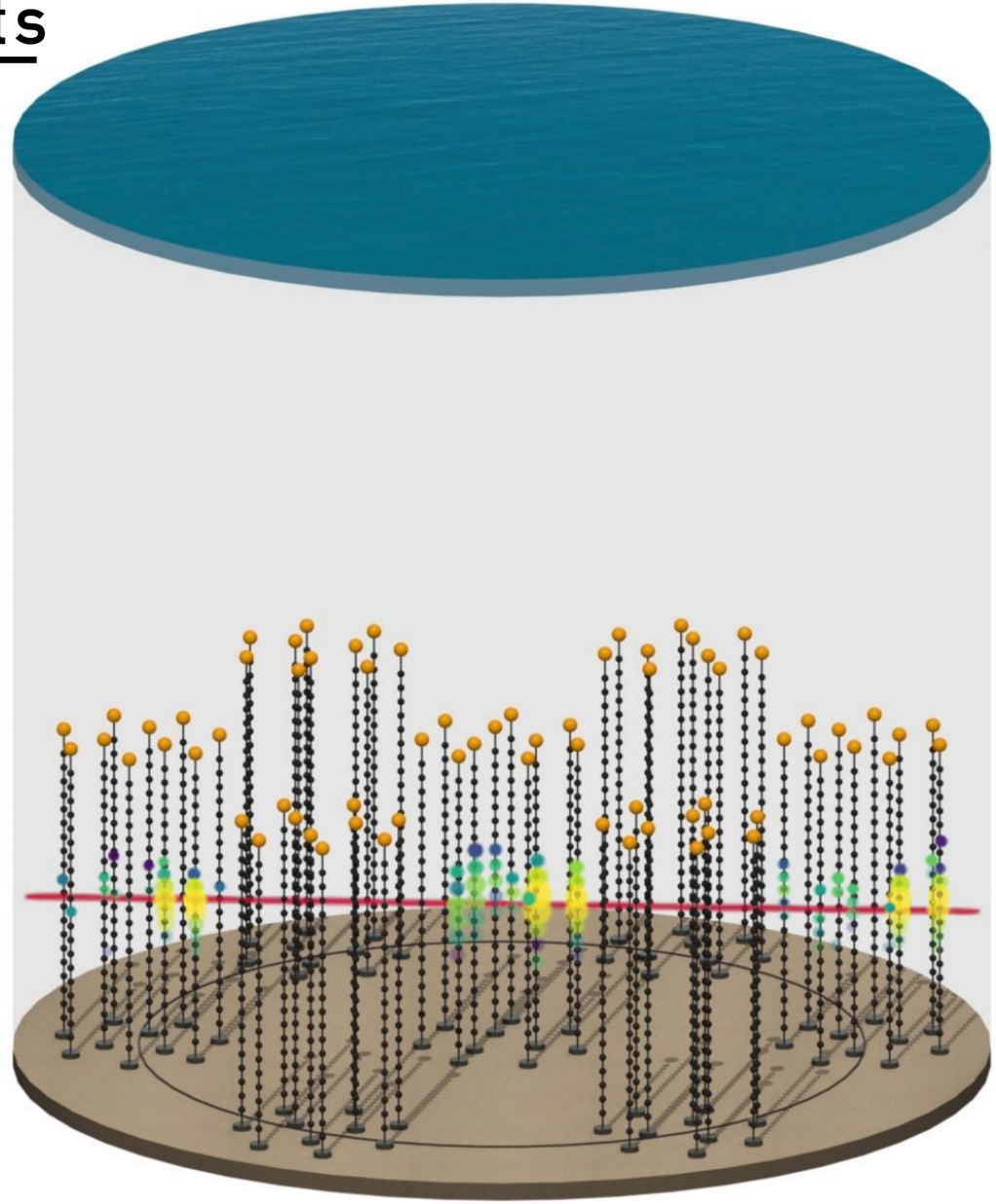
Real-time ML in other experiments



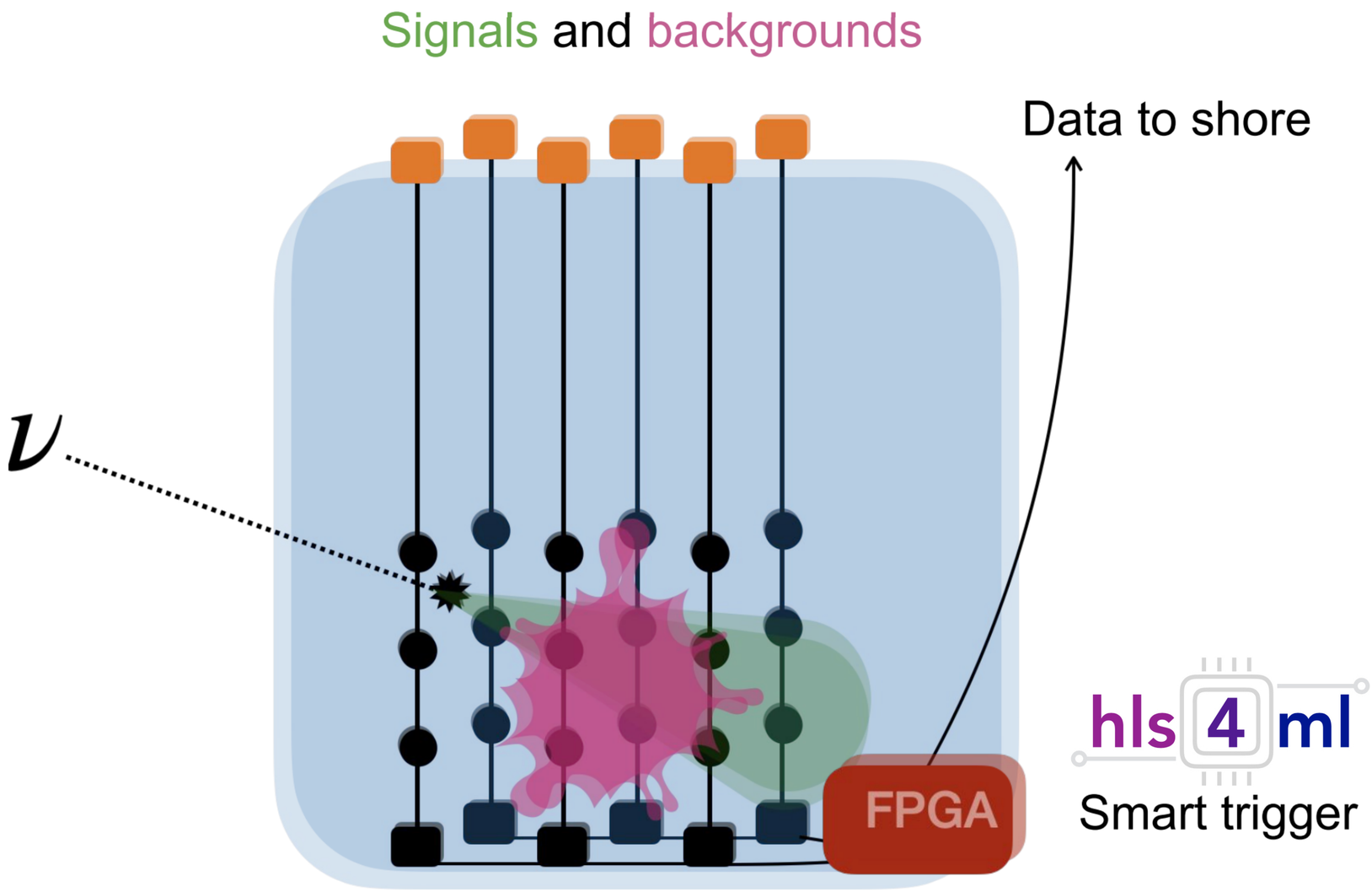
F. Capel et al.

Real-time ML in other experiments



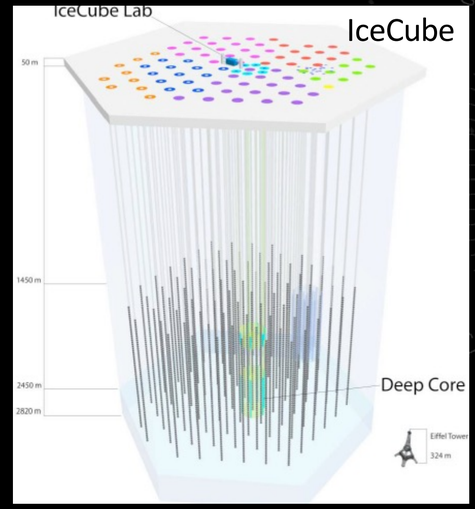


Real-time ML in other experiments



Electromagnetic waves

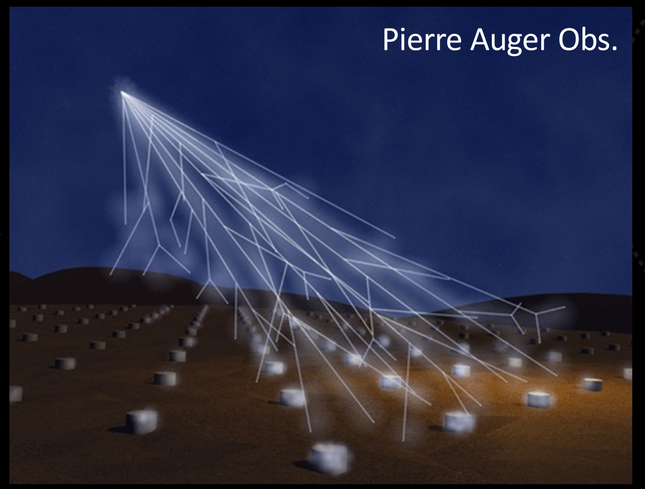
Neutrinos



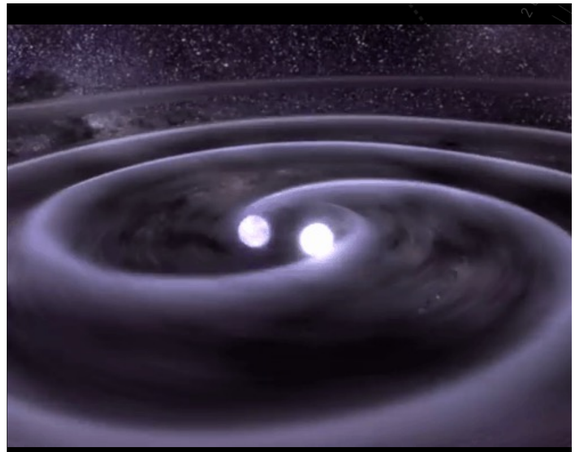
MULTI-MESSENGER ASTRONOMY (MMA)

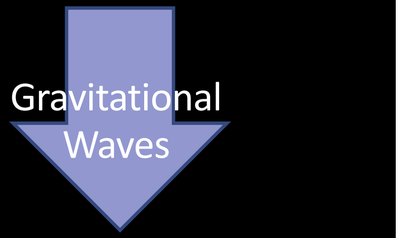
Gravitational Waves

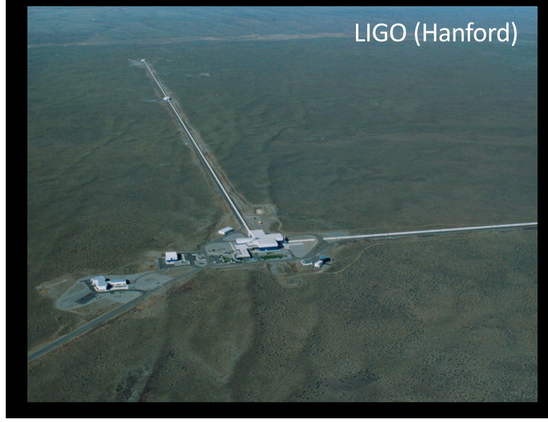
Cosmic Rays



Real-time gravitational wave detection

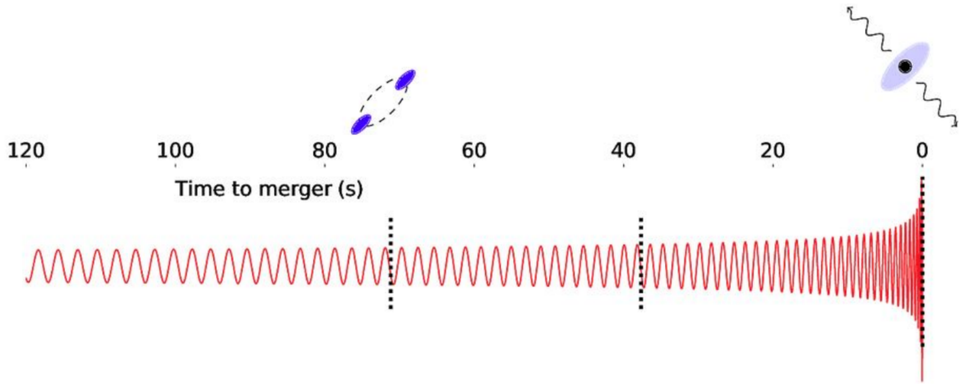






Using gravitational waves, we can detect neutron star merger (kilo novae) before it occurs!

- Use real time ML for event cleaning, parameter estimation and anomaly detection!



Rapid communication of gravitational-wave date on

- source location
- properties

Crucial to facilitate multi-messenger follow-up of sources !

LSST TAKES 20TB OF IMAGES PER NIGHT

Rubin Observatory

10 million transient alerts per night!

How can we expect to get lucky with such high volumes of data?

Transient searches have relied on human eyes for alerts ***→real-time ML!***

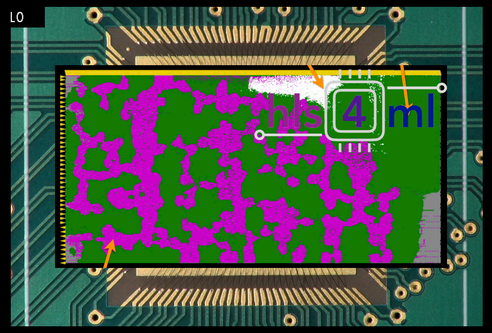
Conifer

hls4ml

Join the community:

fastmachinelearning.org

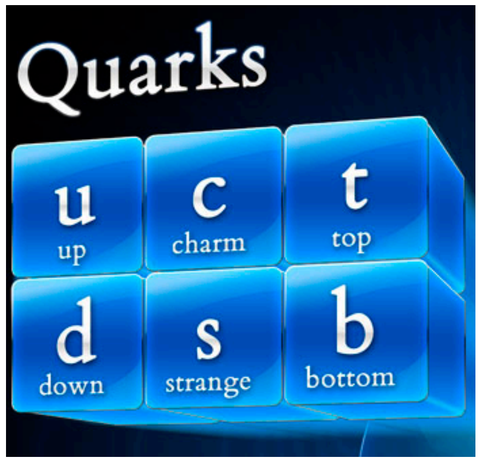
Sign up to the hls-fml group

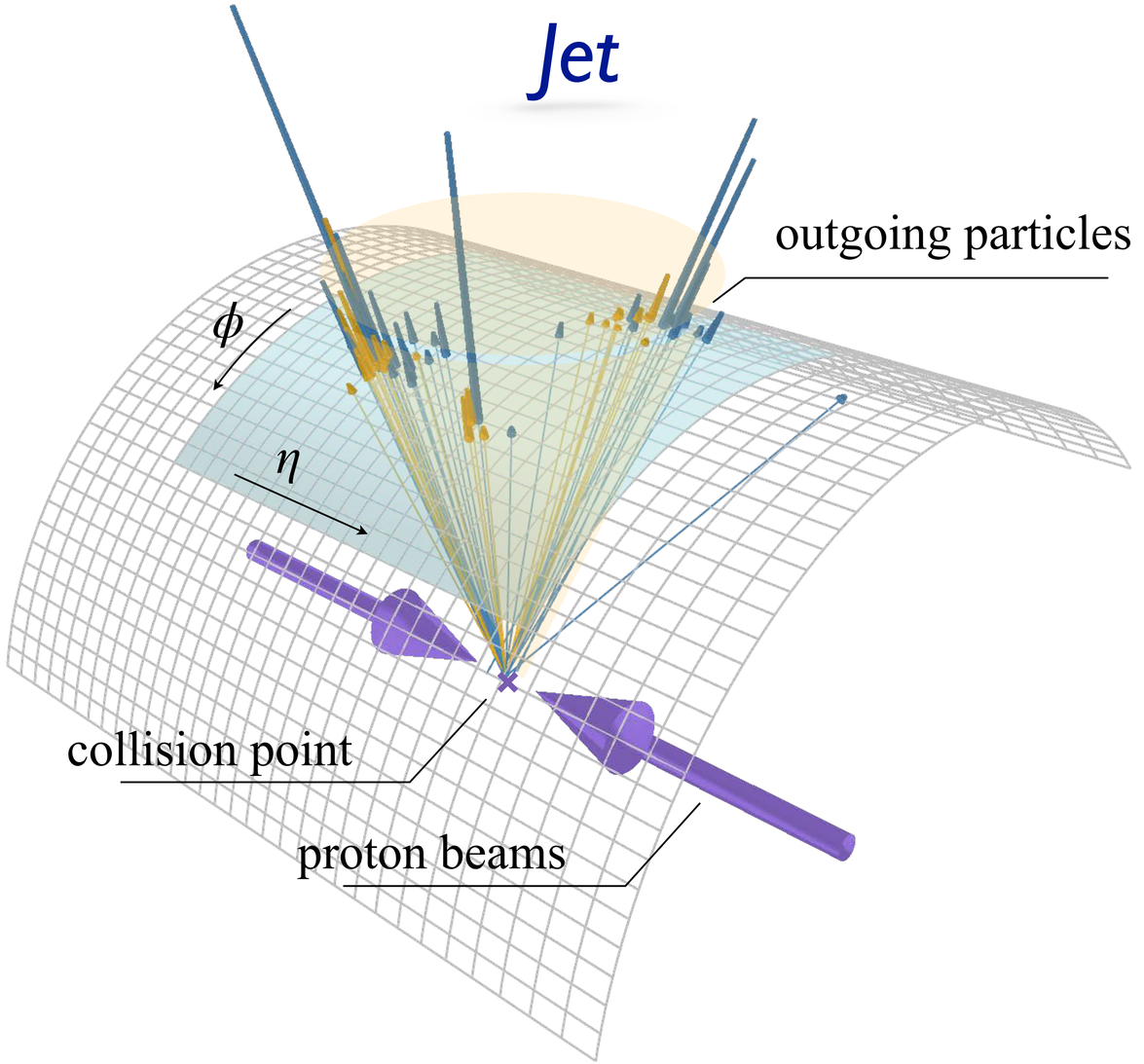


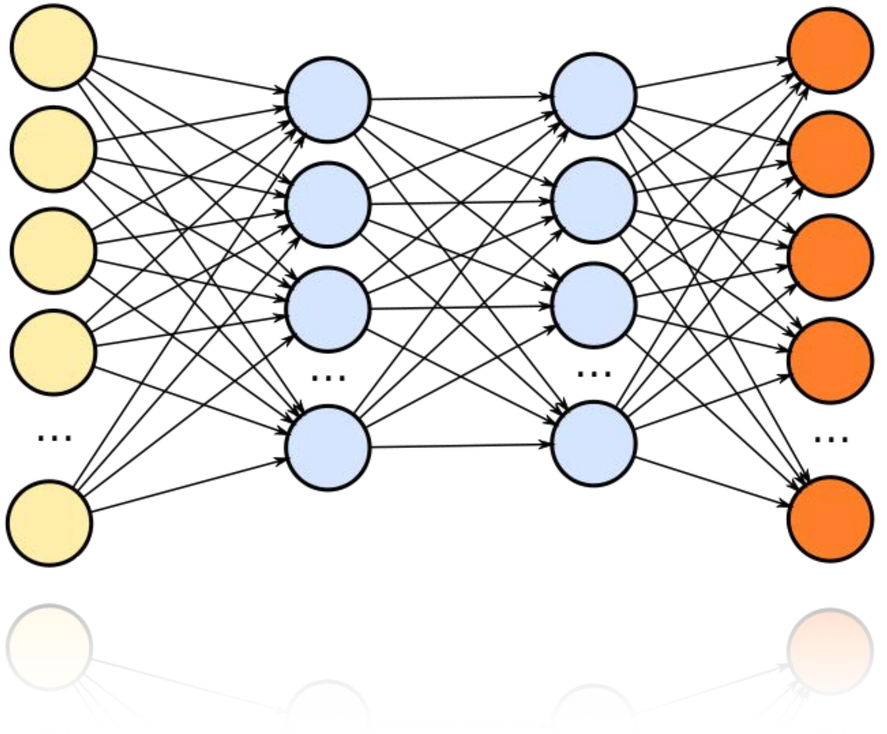
Bonus: Transformers (?)

Thea Klæboe Årrestad (ETH Zürich)

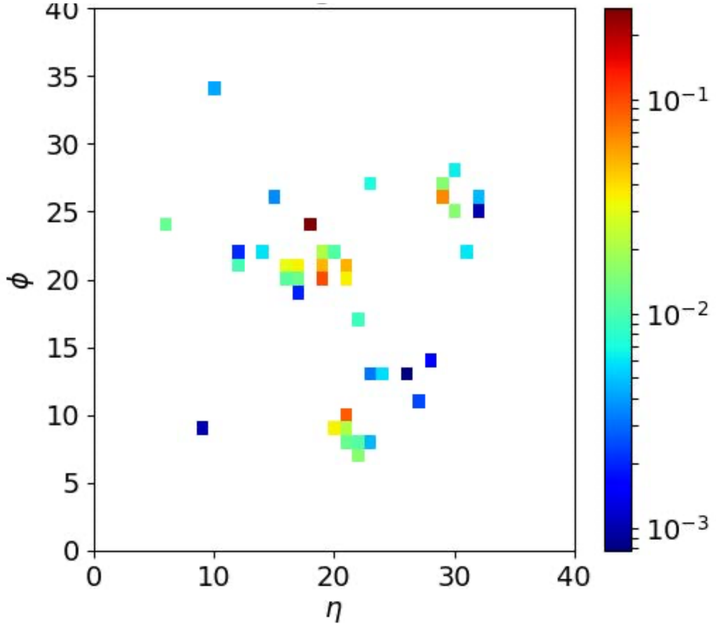
?


Quarks
u
up
c
charm
t
top
d
down
s
strange
b
bottom


Jet
outgoing particles
ϕ
η
collision point
proton beams

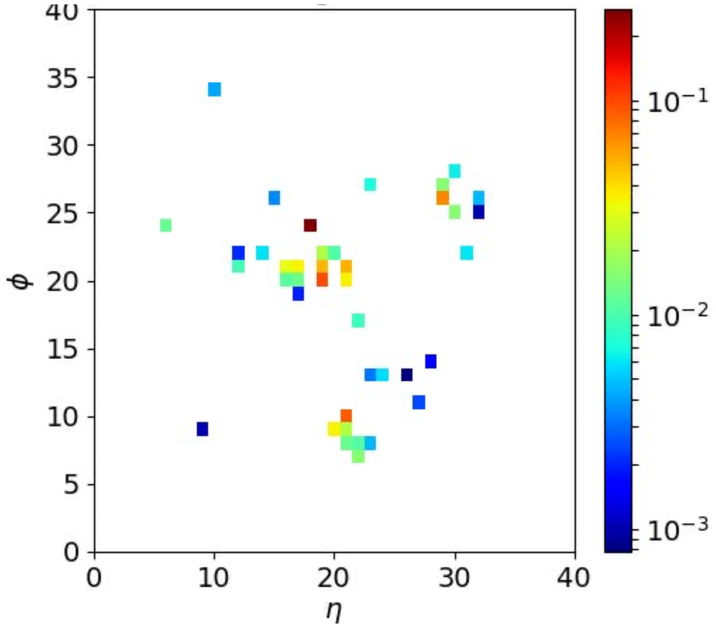


Image



arXiv:1511.05190

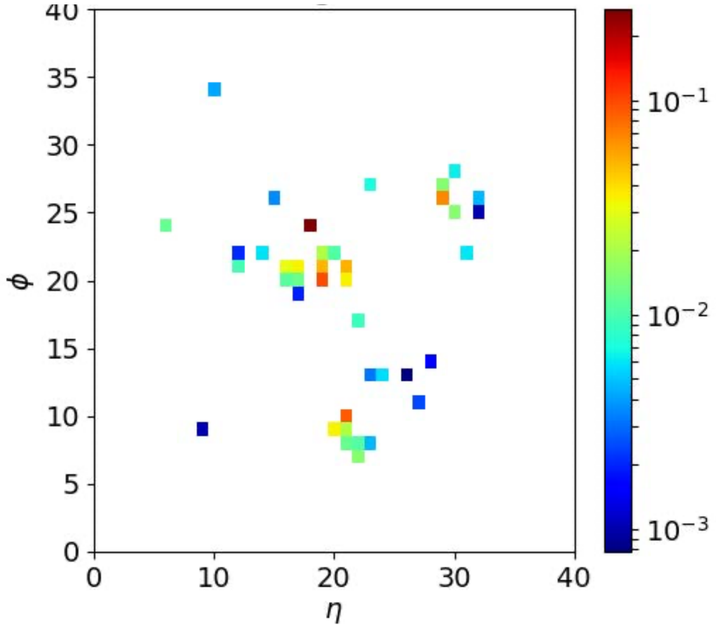
Image



arXiv:1511.05190

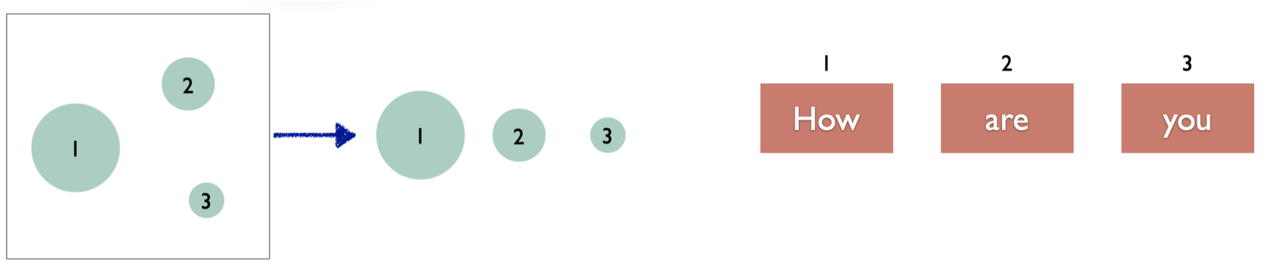
But... inhomogeneous geometry, high sparsity

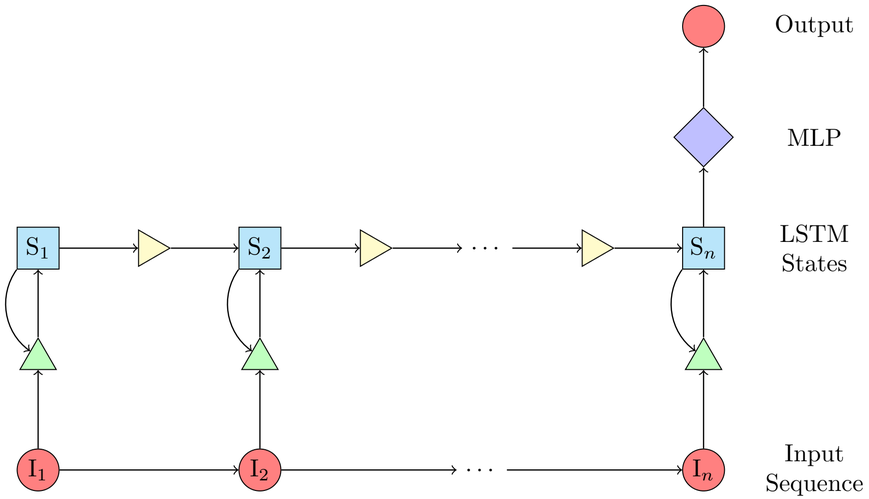
Image



arXiv:1511.05190

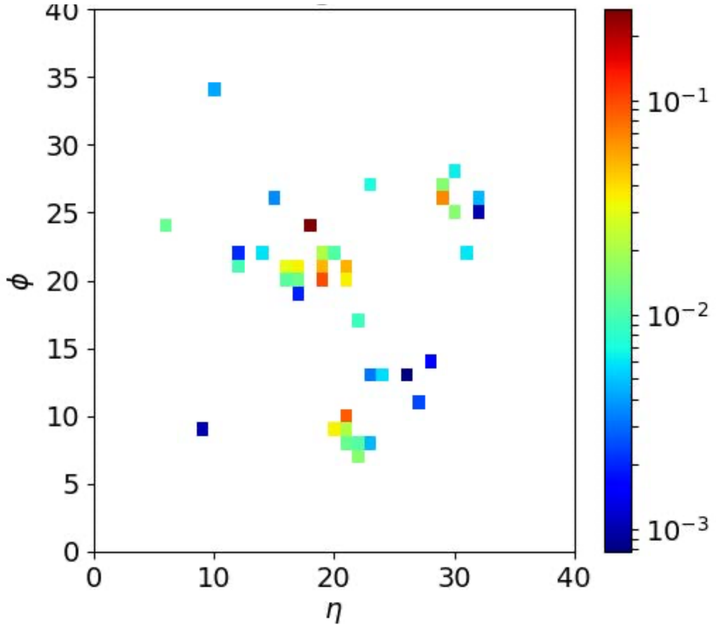
Sequence





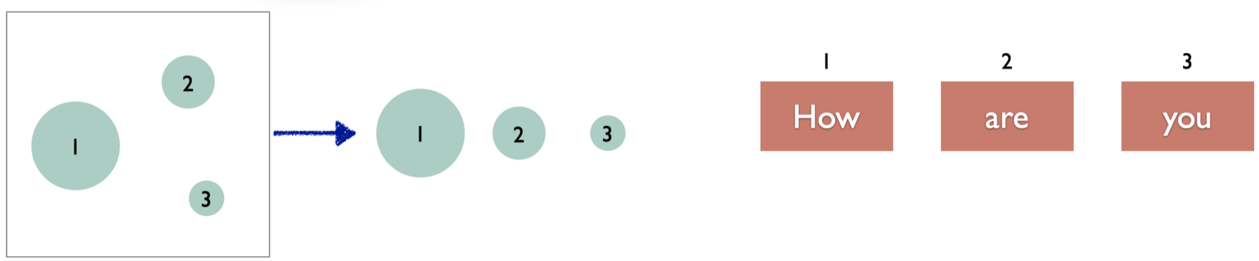
arXiv:1607.08633

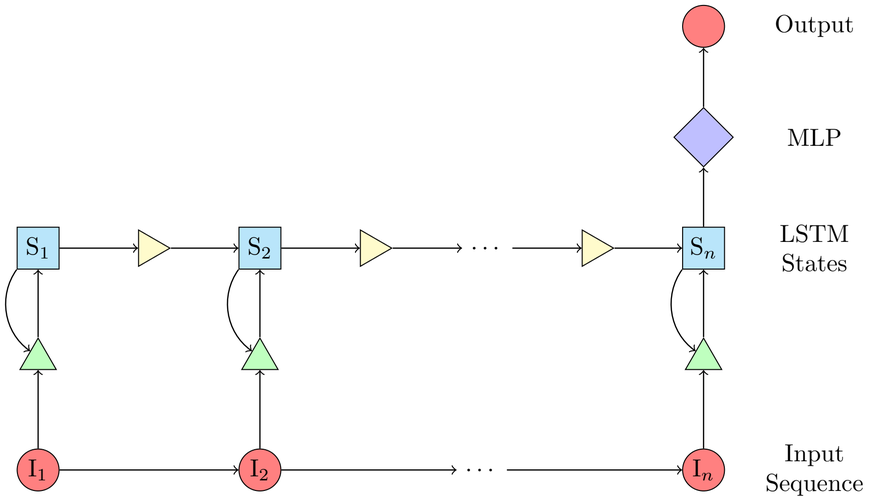
Image



arXiv:1511.05190

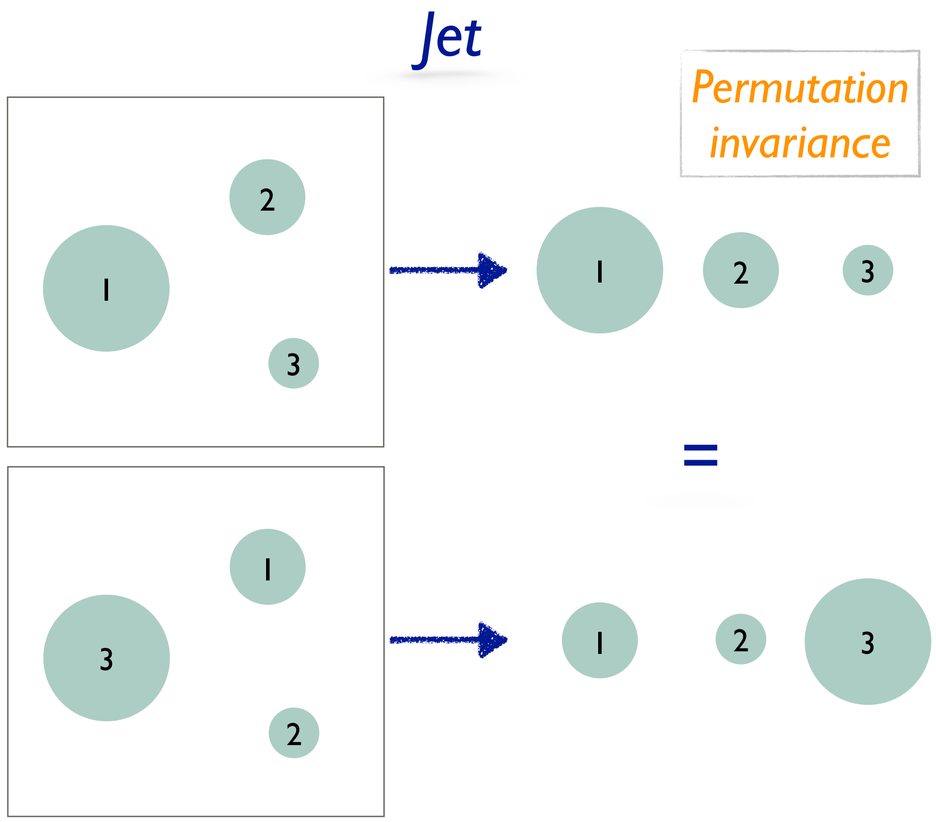
Sequence

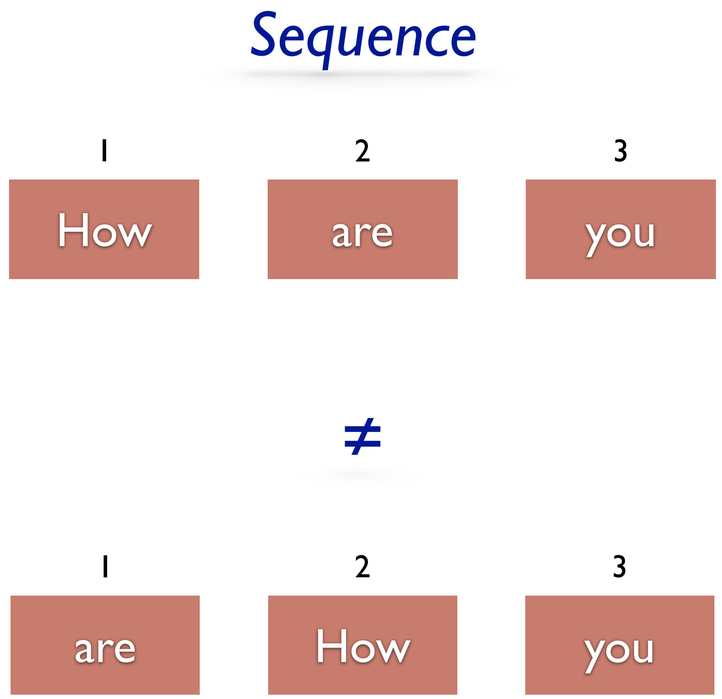




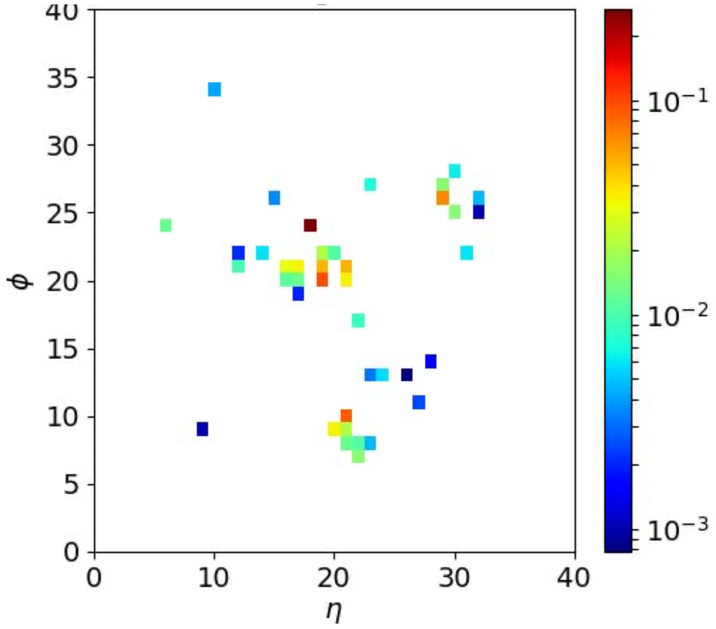
arXiv:1607.08633

But... permutation-invariance


Jet
Permutation invariance
2
1
3
1
2
3
=
1
3
2
1
2
3

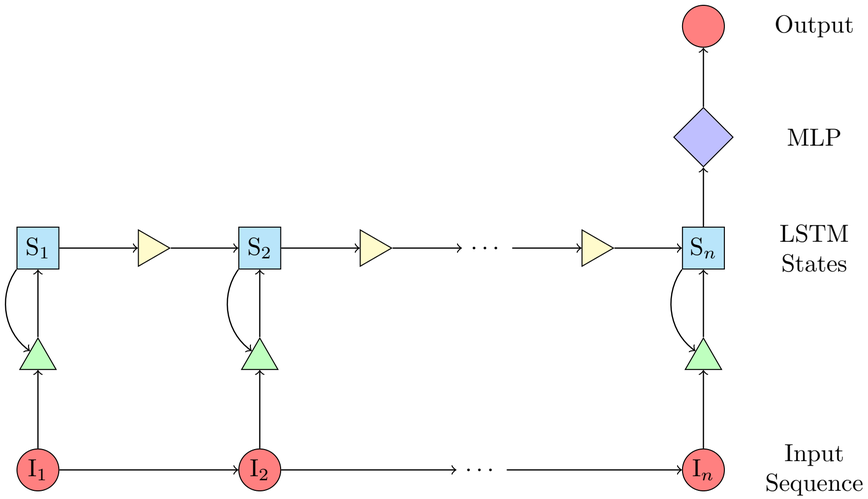

Sequence
1
2
3
How
are
you
≠
1
2
3
are
How
you

Image



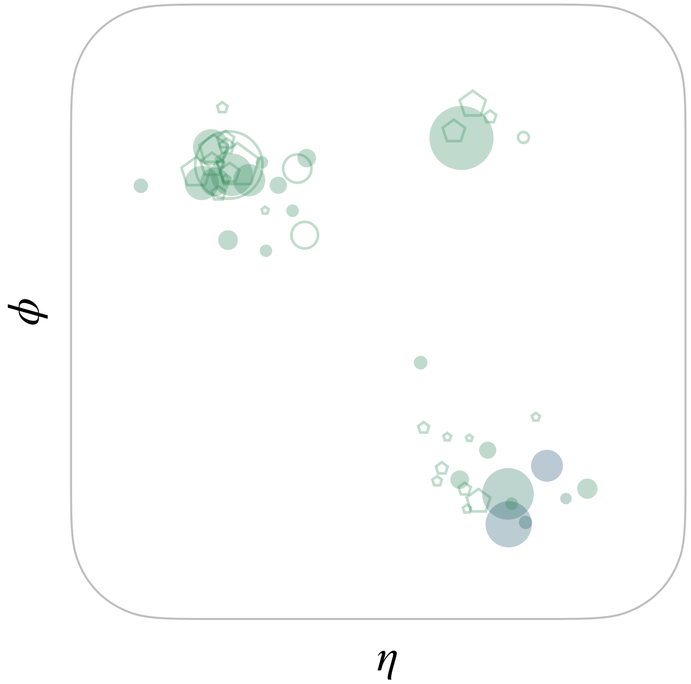
arXiv:1511.05190

Sequence

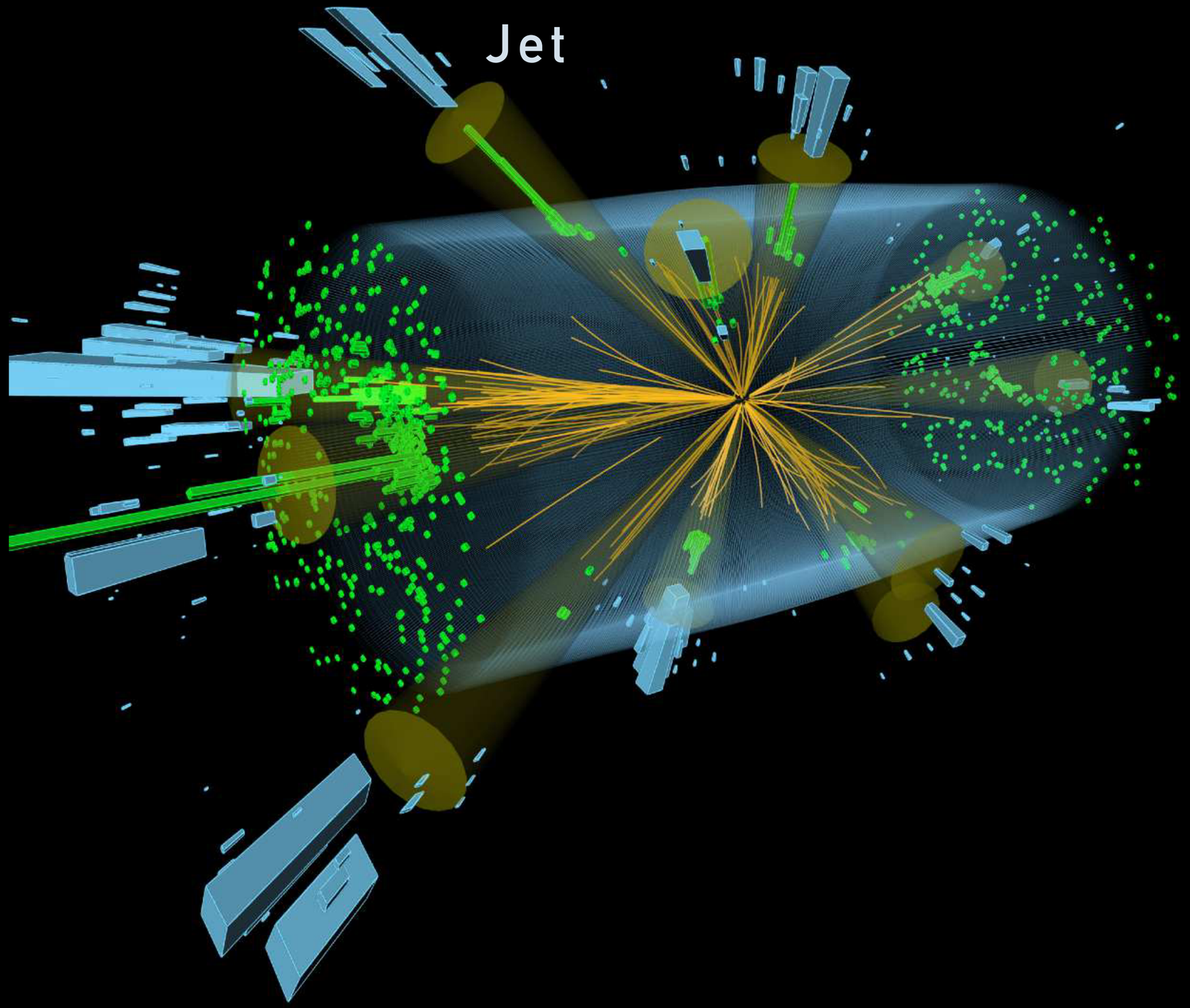


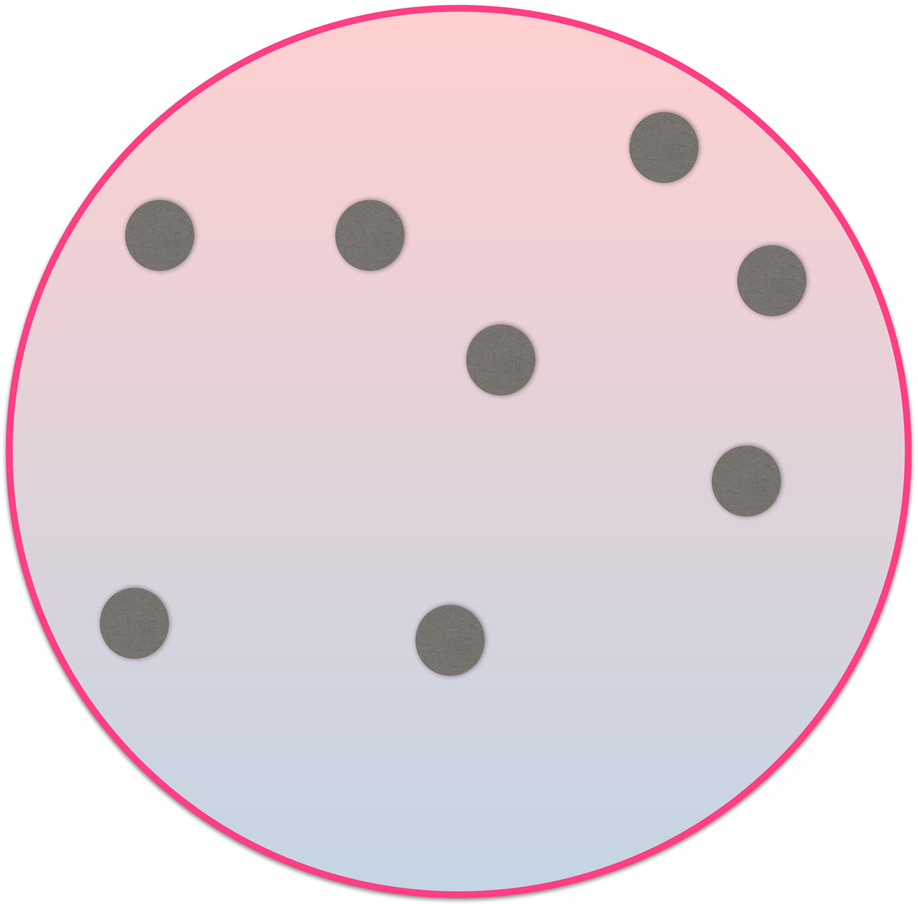
arXiv:1607.08633

Point Cloud



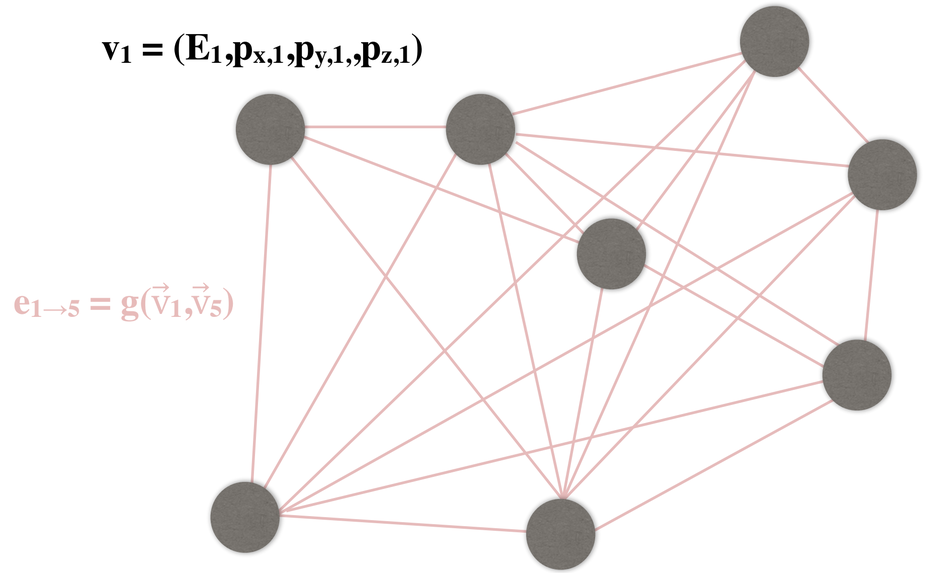
PRD:101.056019


Jet



$v_2 = (E_2, p_{x,2}, p_{y,2}, , p_{z,2})$

$v_1 = (E_1, p_{x,1}, p_{y,1}, , p_{z,1})$



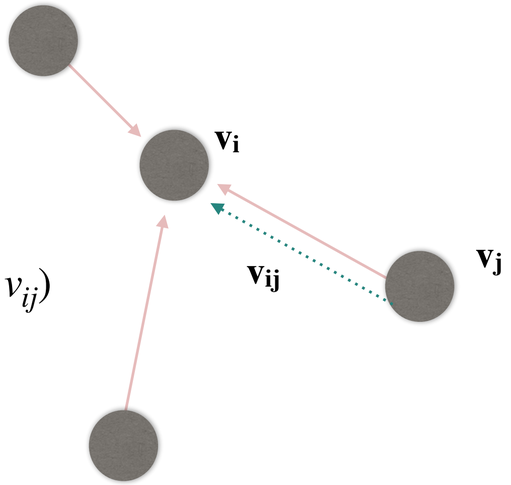
$e_{1\to5} = g(\vec{v}_1, \vec{v}_5)$

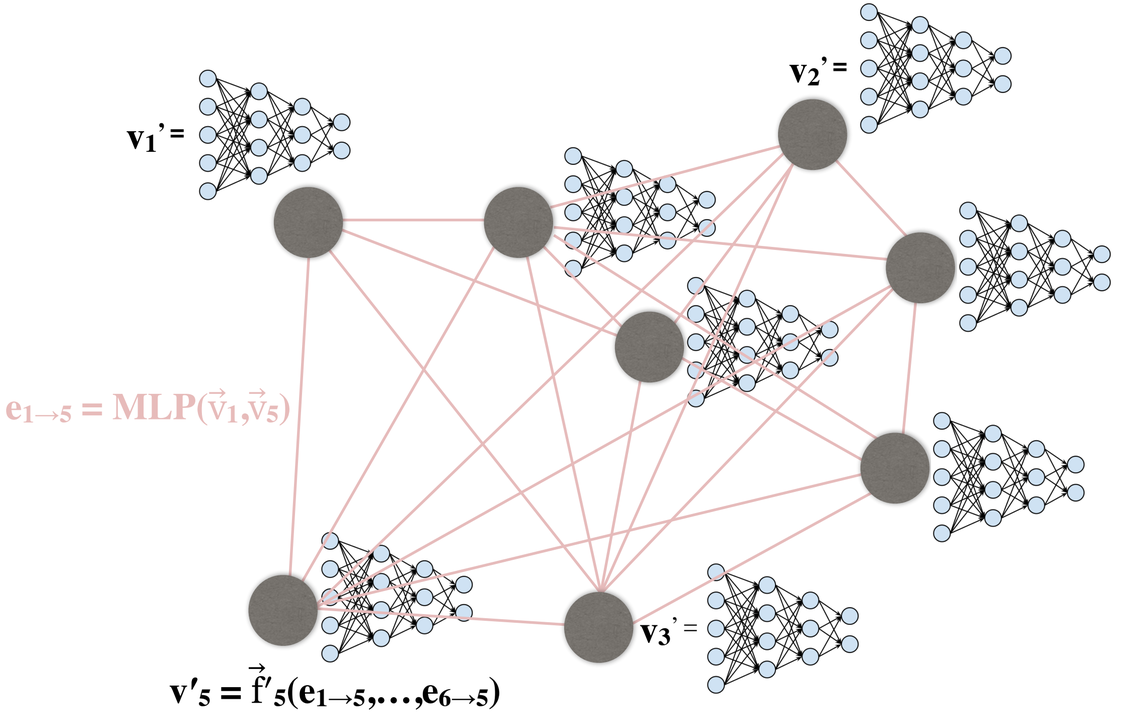
$v_5 = (E_5, p_{x,5}, p_{y,5}, , p_{z,5})$

v_3

SOTA: Graph Neural Networks acting on point cloud data

- ParticleNet (GNN on point cloud)
  LundNet (GNN,Lund plane)
  ABCNet (GNN, attention)
  Point Cloud Transformers (transformer, attention)
  ParticleNeXt (GNN, attention, Lund)
  ParT (transformer, attention)





Invariant Operation

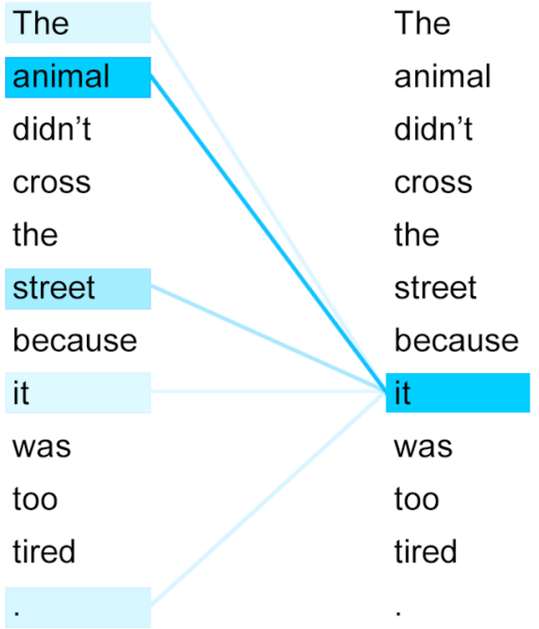
Fully connected

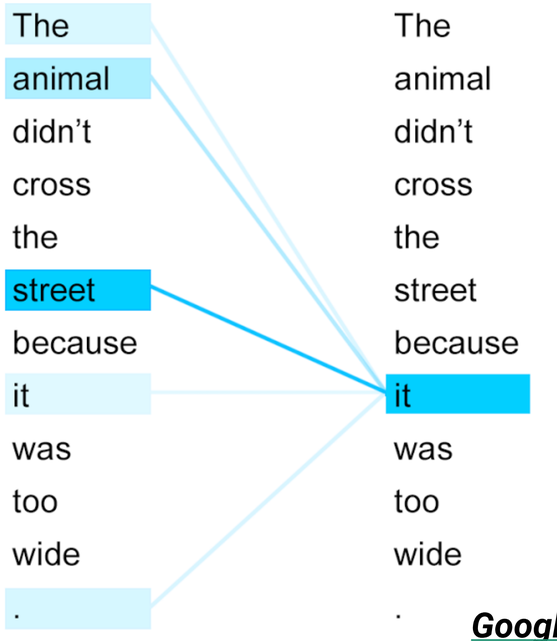
Fully connected

Transformers and (self-)attention

Adding (Self-)Attention

- Allows particles to interact with each other ("self") and find out which other particles they should pay more attention to ("attention")



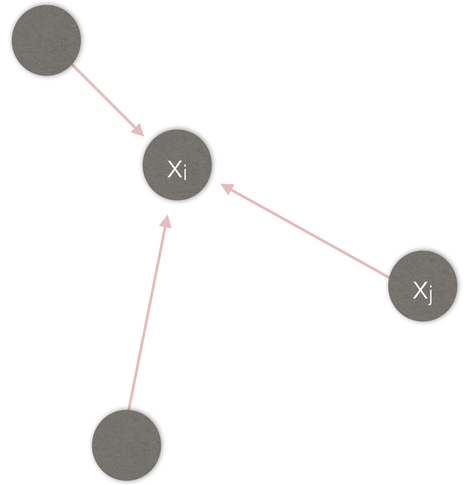


Google AI blog

Transformers and (self-)attention

Adding (Self-)Attention

- Allows particles to interact with each other ("self") and find out which other particles they should pay more attention to ("attention")



Weighted sum over all input vectors:

$$\mathbf{y}_i = \sum_j w_{ij} \mathbf{x}_j$$

Weight (how related inputs are):

$x_j \rightarrow MLP(x_J)$
$x_i \rightarrow MLP(x_i)$

$$w'_{ij} = \mathbf{x}_i{}^{\mathsf{T}} \mathbf{x}_j$$

Map to [0,1]:

$$w_{ij} = \frac{\exp w'_{ij}}{\sum_j \exp w'_{ij}}$$

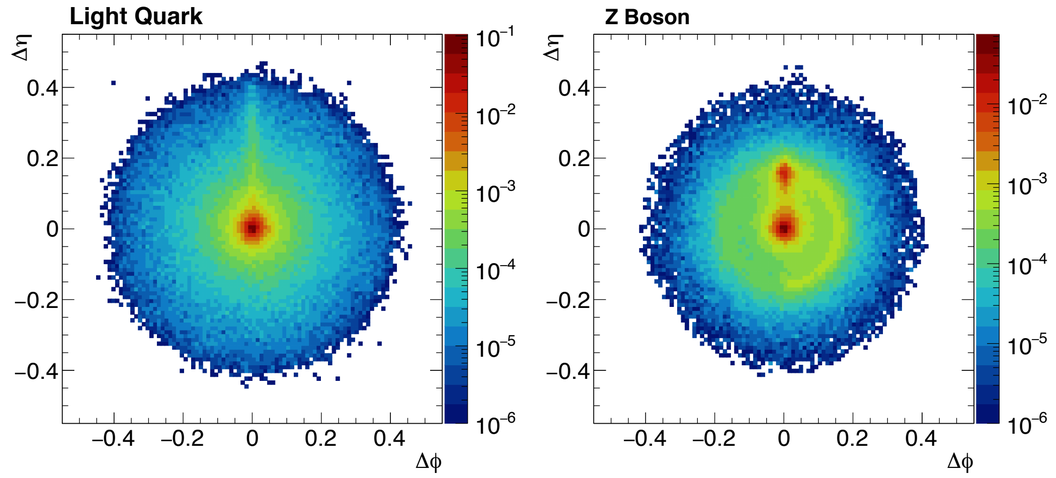
Transformers and (self-)attention

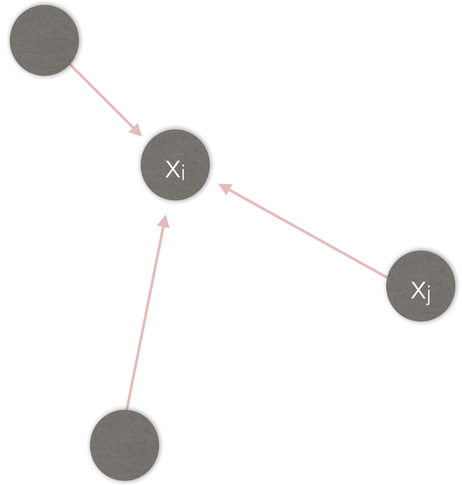
Adding (Self-)Attention

- Allows particles to interact with each other ("self") and find out which other particles they should pay more attention to ("attention")

Attention weights: weighted importance between each pair of particles

- Determine relationship between all particles of point cloud
- Several attention layers → different important features (multi-head attention)





Weighted sum over all input vectors:

$$\mathbf{y}_i = \sum_j w_{ij} \mathbf{x}_j$$

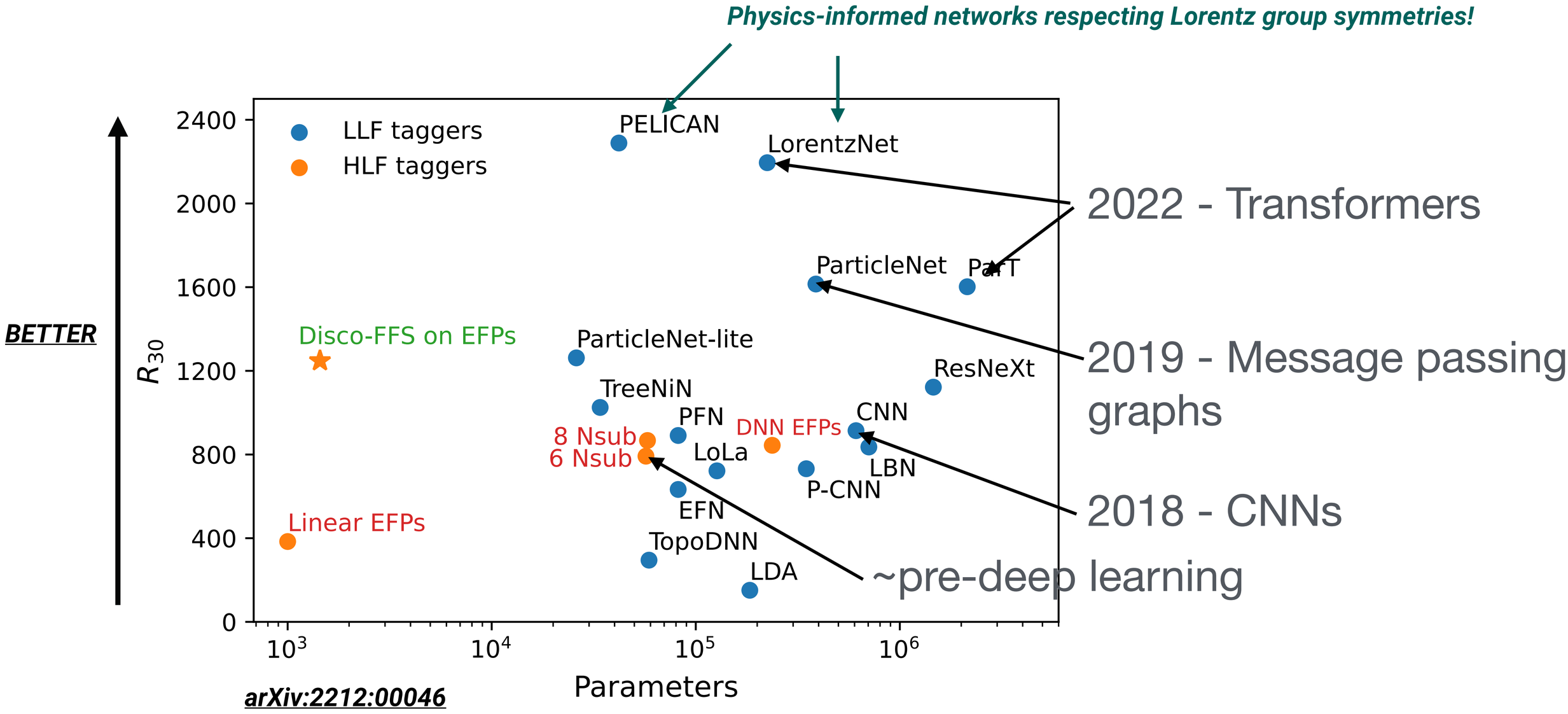
Weight (how related inputs are):

$x_j \rightarrow \text{MLP}(x_J)$
$x_i \rightarrow \text{MLP}(x_i)$

$$w'_{ij} = \mathbf{x}_i{}^{\mathsf{T}} \mathbf{x}_j$$

Map to [0,1]:

$$w_{ij} = \frac{\exp w'_{ij}}{\sum_j \exp w'_{ij}}$$


Physics-informed networks respecting Lorentz group symmetries!
BETTER
R_{30}
2400
2000
1600
1200
800
400
0
10^3
10^4
10^5
10^6
Parameters
LLF taggers
HLF taggers
PELICAN
LorentzNet
ParticleNet
ParT
ParticleNet-lite
Disco-FFS on EFPs
ResNeXt
TreeNiN
CNN
PFN
DNN EFPs
8 Nsub
6 Nsub
LoLa
LBN
P-CNN
EFN
Linear EFPs
TopoDNN
LDA
2022 - Transformers
2019 - Message passing graphs
2018 - CNNs
~pre-deep learning
arXiv:2212:00046