Fast inference of Deep Learning Applications with FPGAs

Maurizio Pierini

Jennifer Ngadiuba Vladimir Loncar, Maurizio Pierini

Javier Duarte, Burt Holzman, Sergo Jindariani ,Ben Kreis, Mia Liu, Kevin Pedro, Ryan Rivera, Nhan Tran, Aris Tsaris

Edward Kreinar

Sioni Summer

Song Han, Phil Harris, Dylan Rankin

Zhenbin Wu

https://github.com/hls-fpga-machine-learning/hls4ml

https://github.com/hls-fpga-machine-learning/SonicCMS

https://hls-fpga-machine-learning.github.io/hls4ml/

arXiv.org https://arxiv.org/pdf/1804.06913.pdf

hls4ml

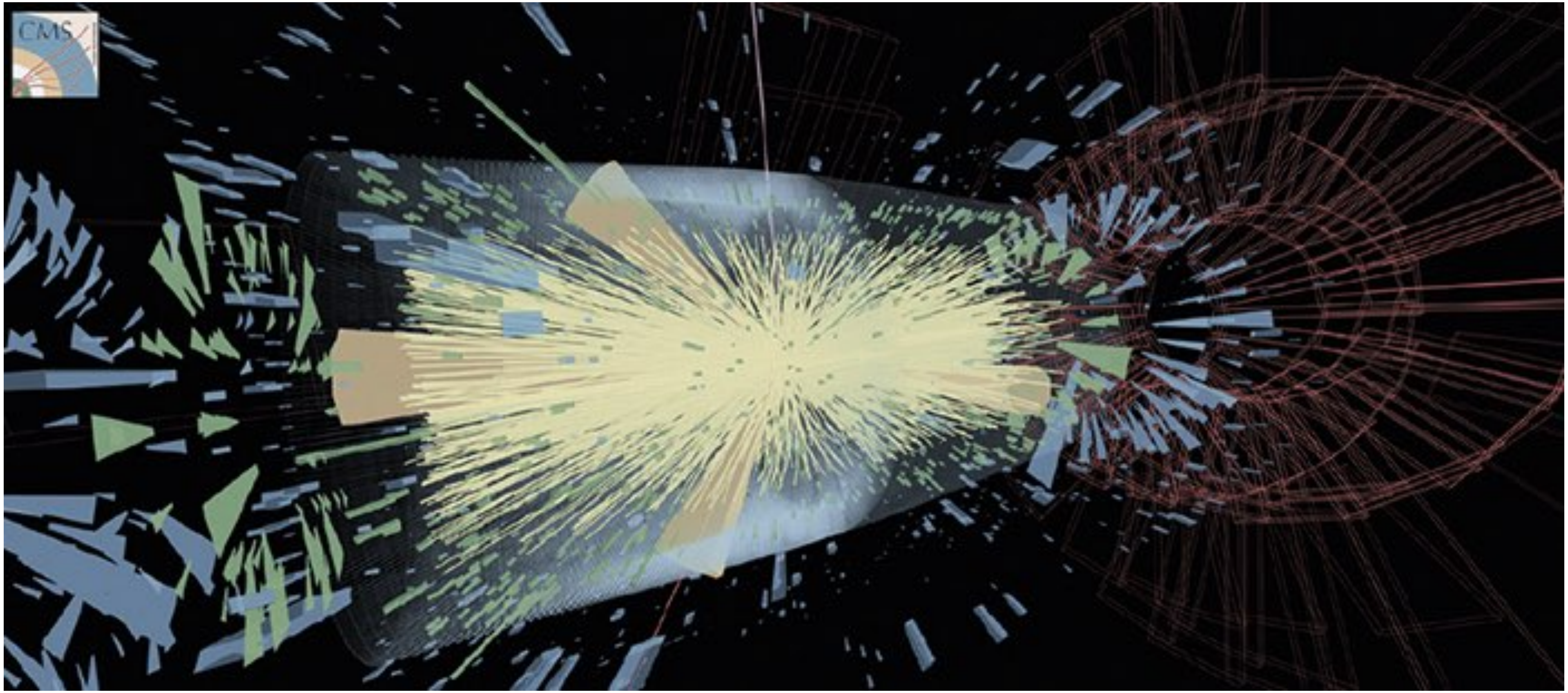
Recent talks at conferences:

CERN Data Science Seminar
FNAL Seminar
Connecting The Dots 2018
TWEPP 2018

erc European Research Council

Outline

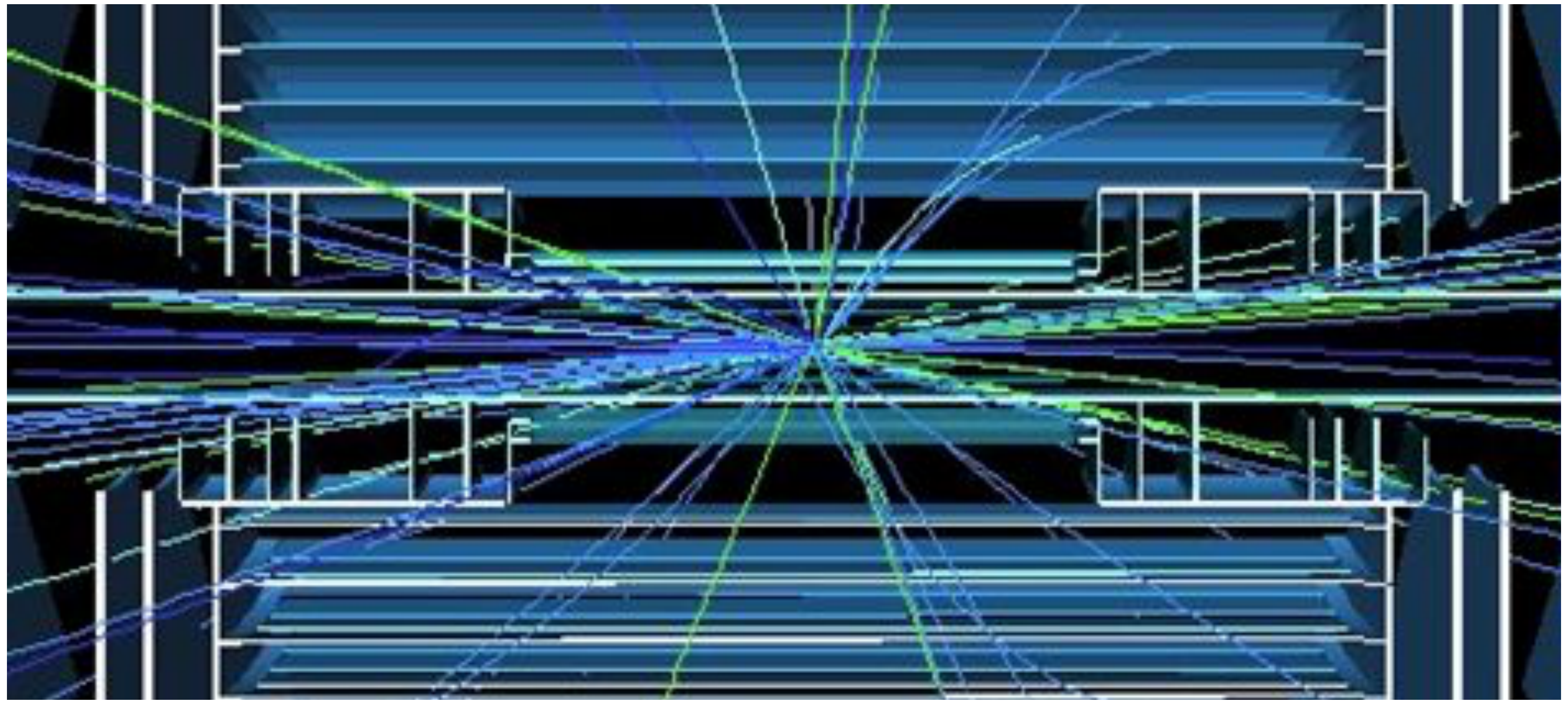
- *The problem: High-Luminosity LHC*
- *Why Deep Learning: a few application examples*
- *Deploying Deep NNs online*
  - *HLT accelerated inference*
  - *L1 NN on FPGAs with HLS*
- *Conclusions*



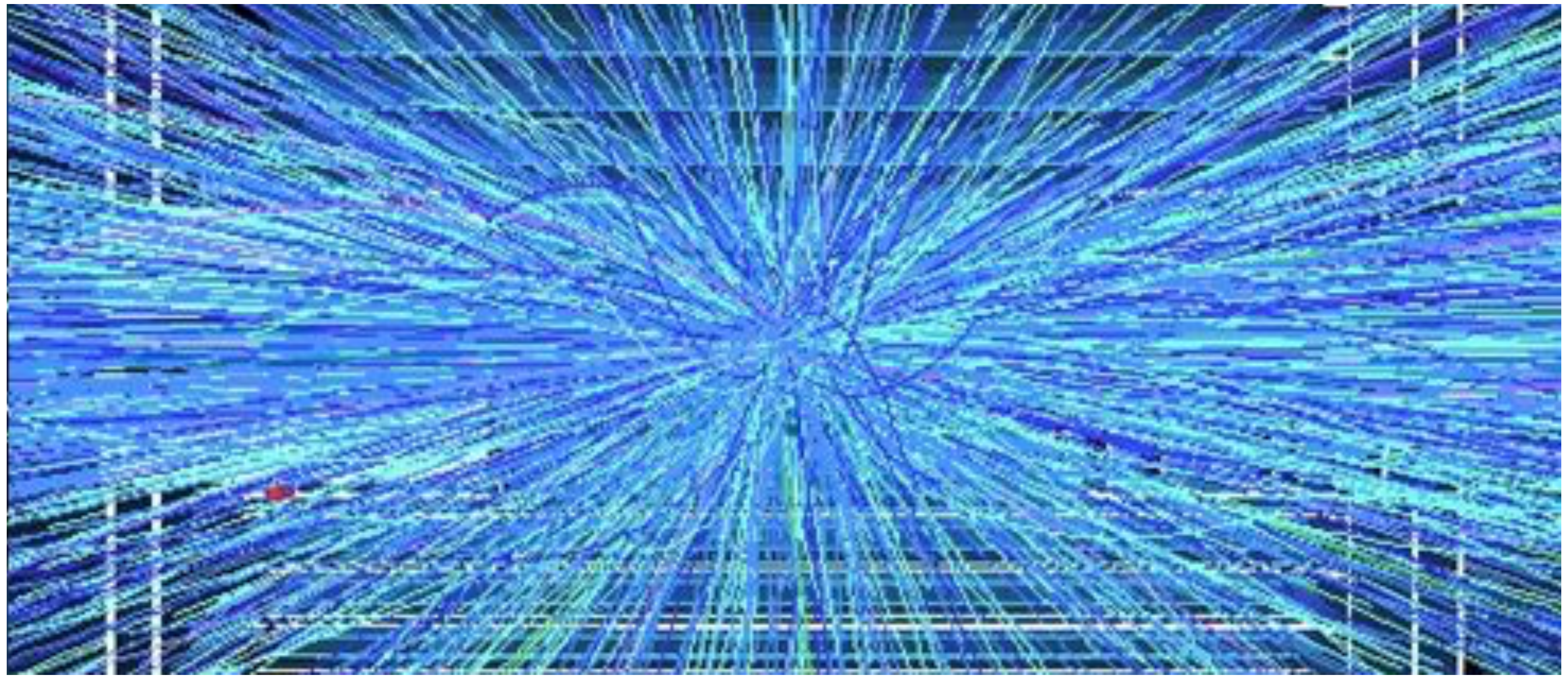
ML and HEP future challenges

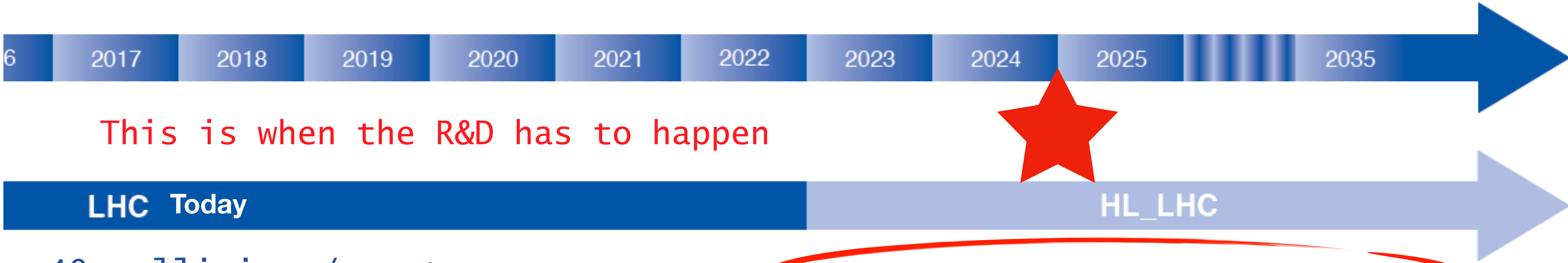
HL-LHC: elephant in the room

5 interactions/beam cross



400 interactions/beam cross



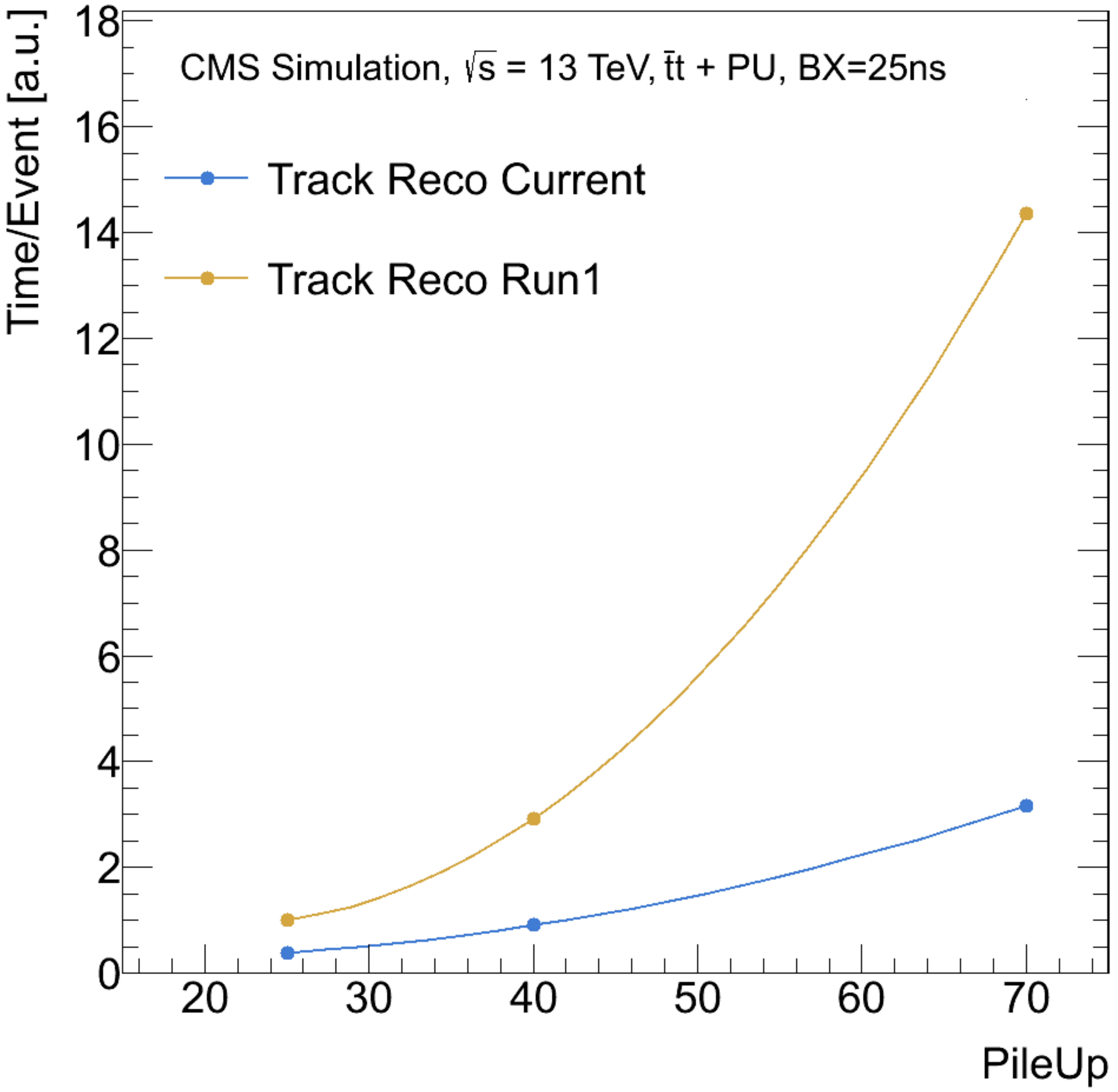


- ~40 collisions/event
- ~10 sec/event processing time
- (at best)Same computing resources as today

- ~200 collisions/event
- ~minute/event processing time
- (at best)Same computing resources as today

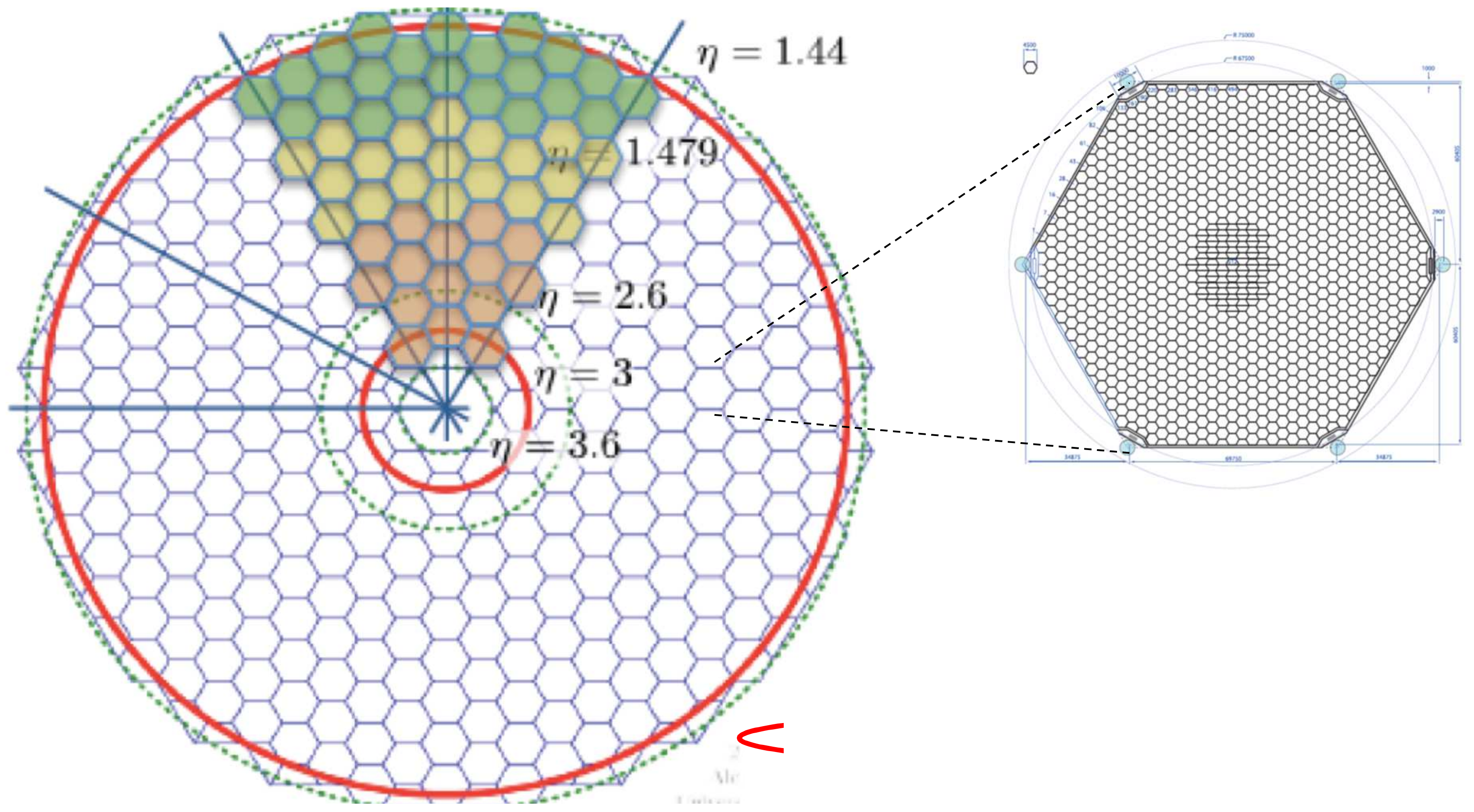
HL-LHC: elephant in the room

- *Flat budget vs. more needs = **<u>current rule-based reconstruction algorithms will not be sustainable</u>***
- ***<u>Adopted solution:</u>** more granular and complex detectors → more computing resources needed → more problems*
- ***Modern Machine Learning might be the way out***

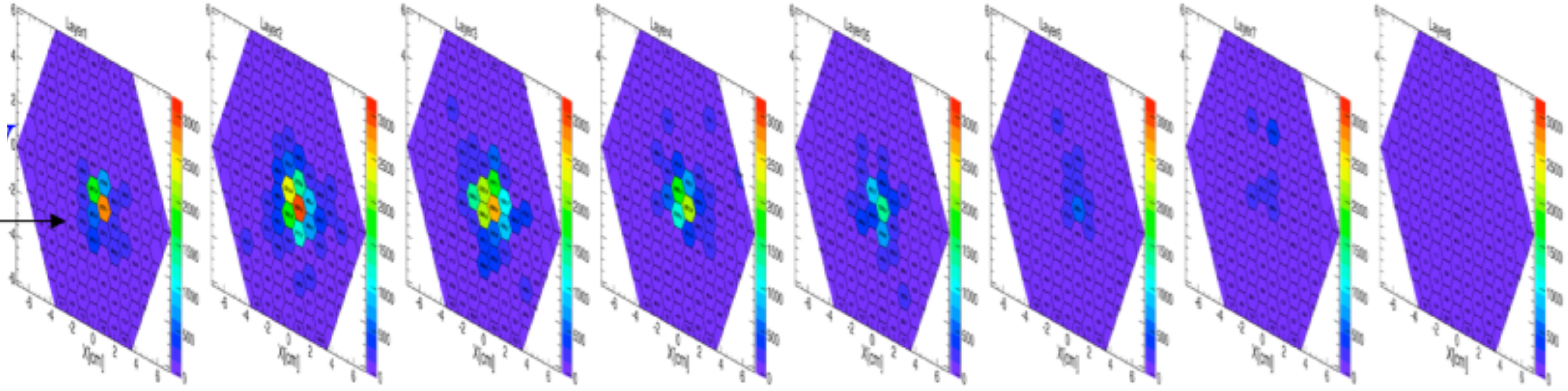


HL-LHC: elephant in the room

- *Flat budget vs. more needs = **current rule-based reconstruction algorithms will not be sustainable***
- ***Adopted solution:** more granular and complex detectors → more computing resources needed → more problems*
- ***Modern Machine Learning might be the way out***

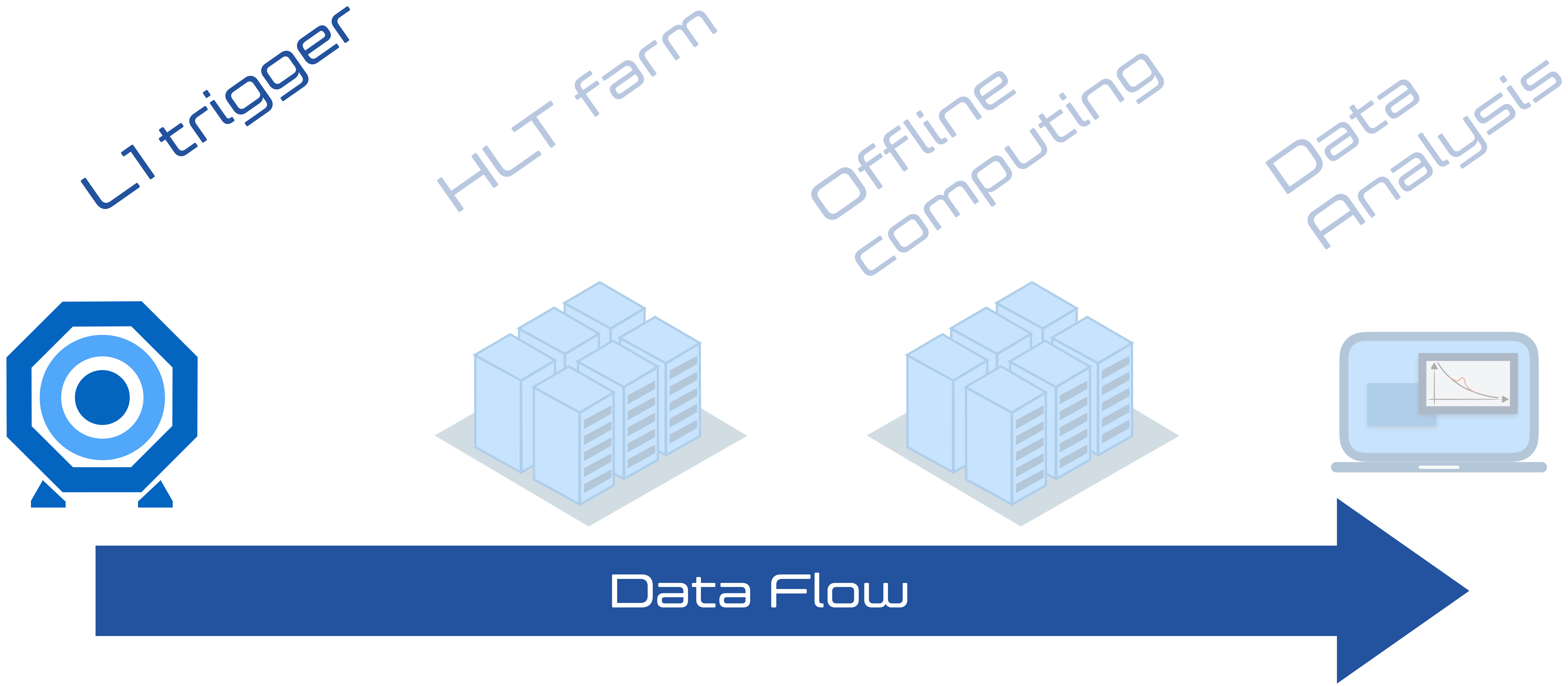


250 GeV electron passing through 8 layers (27 X_0)



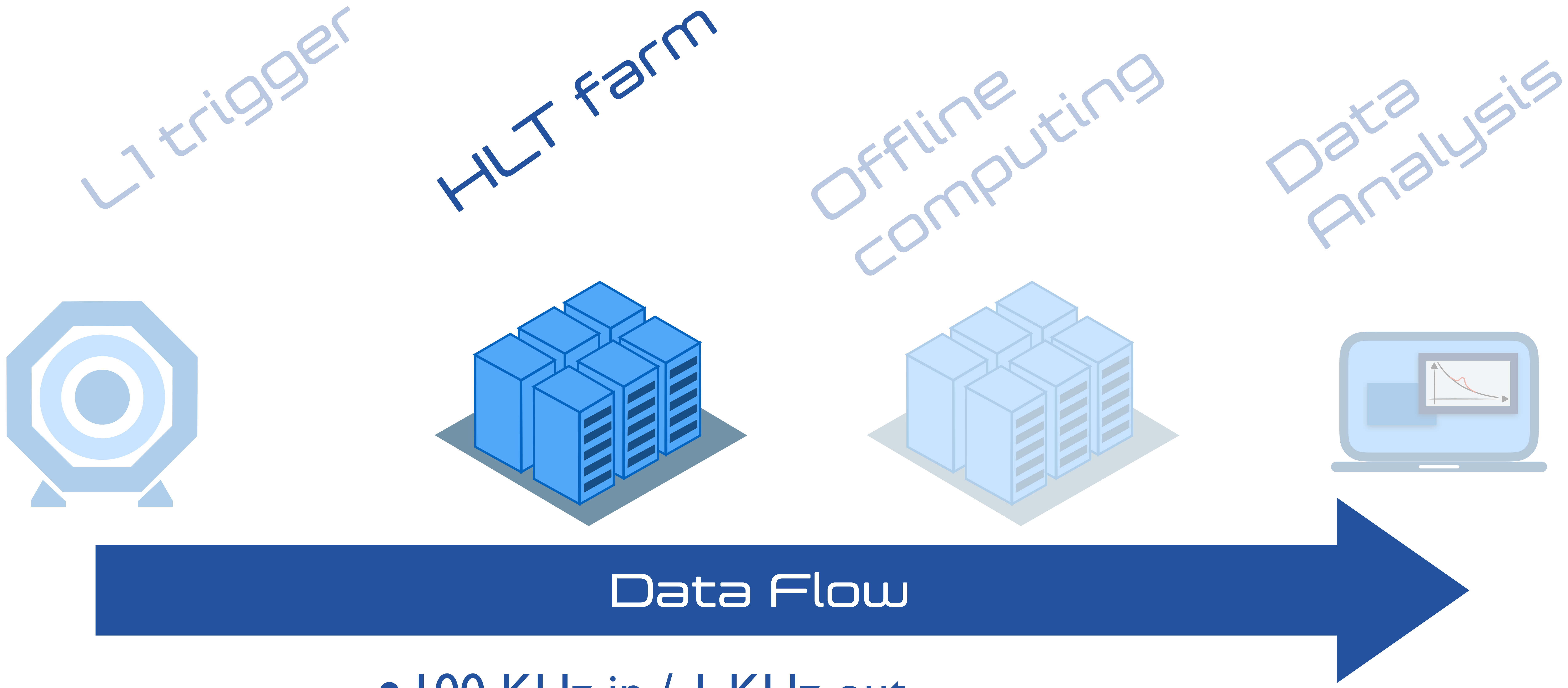
$5X_0$ $8.5X_0$ $12X_0$ $15X_0$ $17X_0$ $19X_0$ $21X_0$ $27X_0$

The LHC Big Data problem



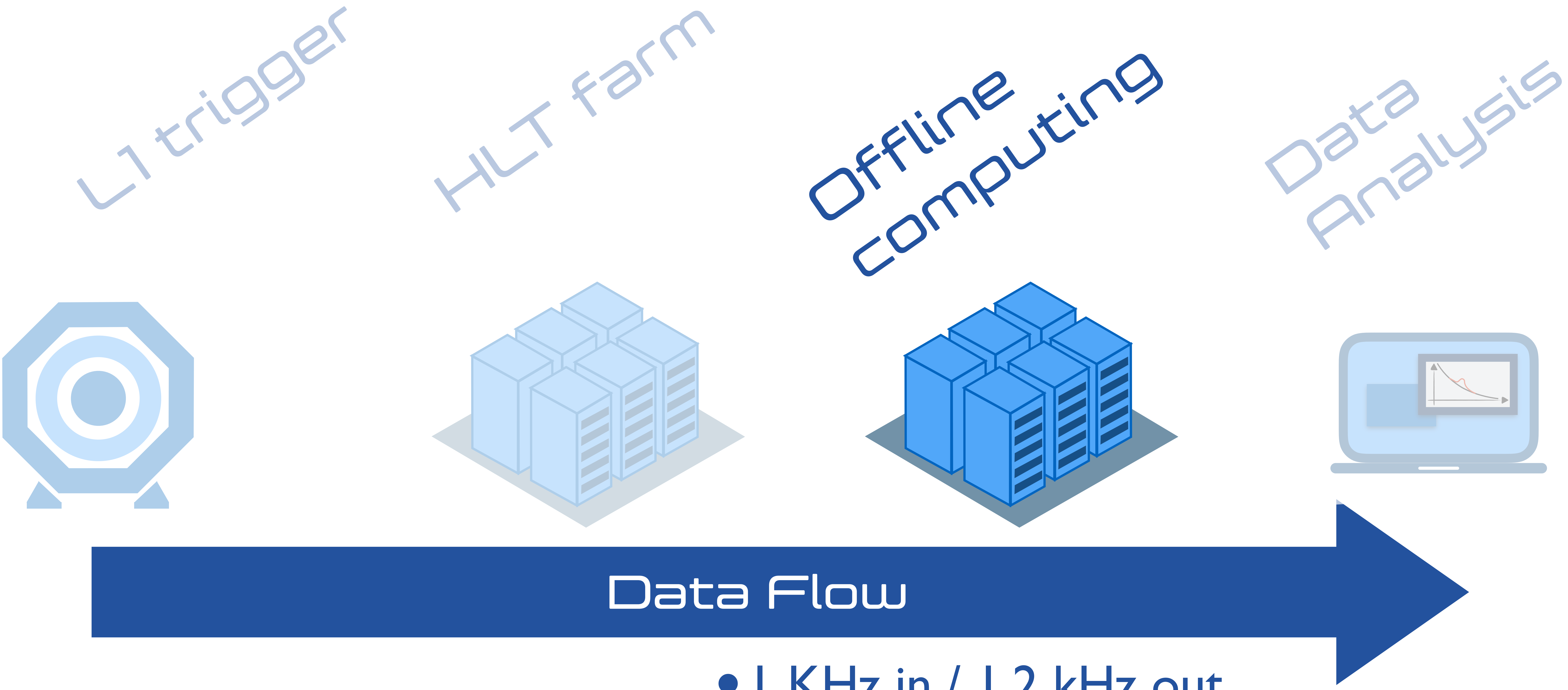
- 40 MHz in / 100 KHz out
- ~ 500 KB / event
- Processing time: ~10 µs
- Based on coarse local reconstructions
- FPGAs / Hardware implemented

The LHC Big Data problem



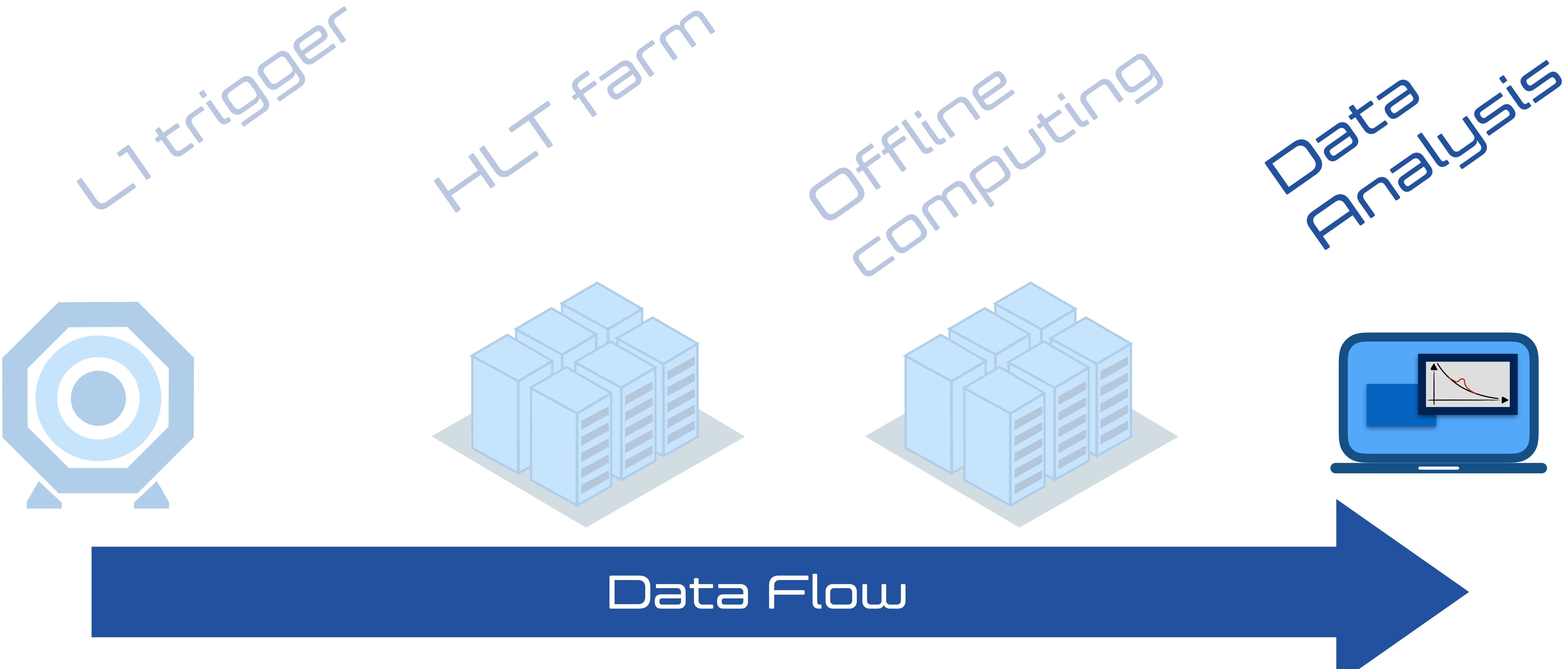
- 100 KHz in / 1 KHz out
- ~ 500 KB / event
- Processing time: ~30 ms
- Based on simplified global reconstructions
- Software implemented on CPUs

The LHC Big Data problem



- 1 KHz in / 1.2 kHz out
- ~ 1 MB / 200 kB / 30 kB per event
- Processing time: ~20 s
- Based on accurate global reconstructions
- Software implemented on CPUs

The LHC Big Data problem



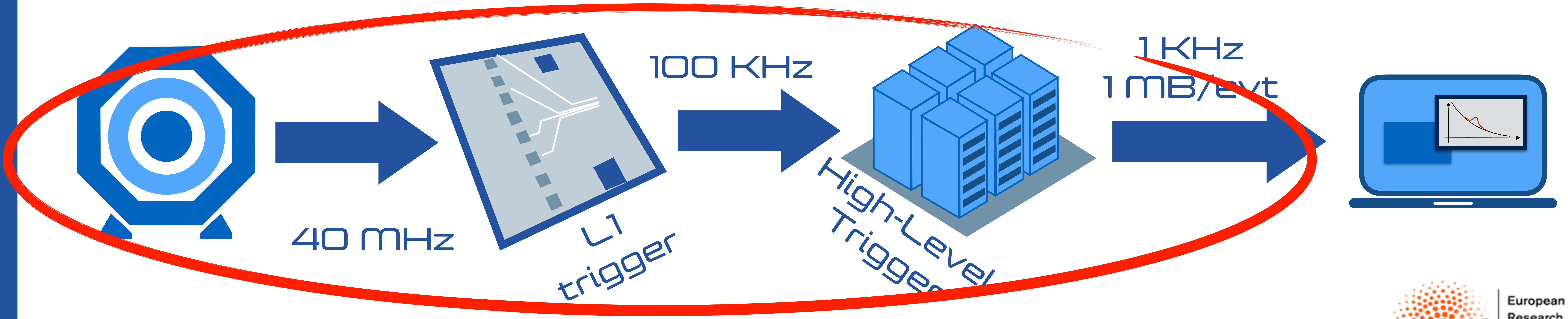
- Up to ~ 500 Hz In / 100-1000 events out

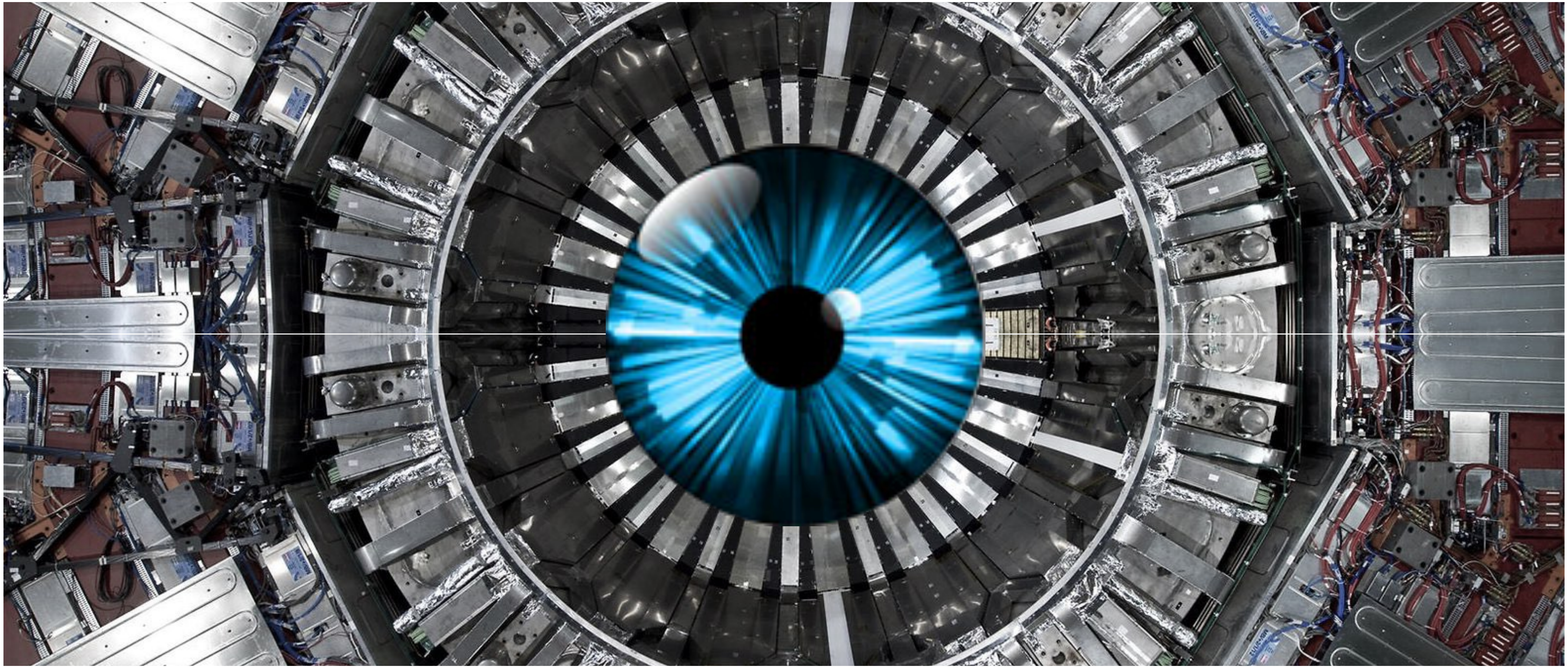
- <30 KB per event
- Processing time irrelevant
- User-written code + centrally produced selection algorithms

Deep Learning and LHC Big Data

- *Possible solution to the HL-LHC problem: modern Machine Learning to be <u>faster</u> and <u>better</u> in what we do today, freeing resources for new ideas*

- *This ML deployment need to happen in between collisions and data analysis (trigger, reconstruction, …), where freeing resources will make a difference*

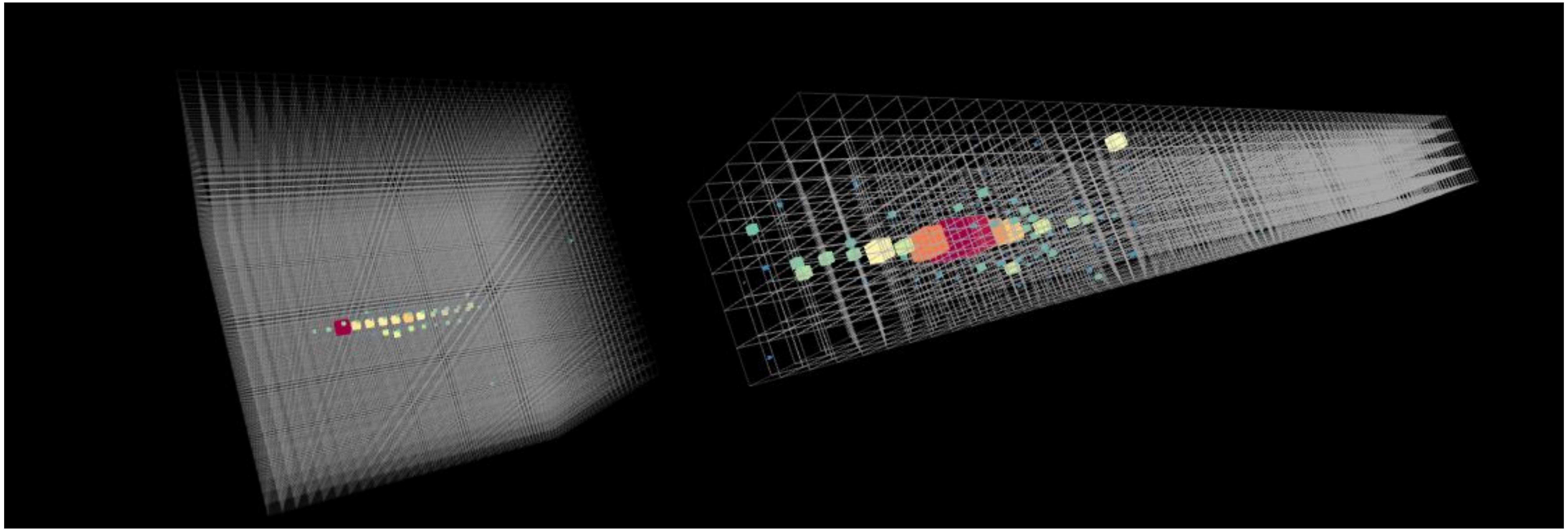




Faster Particle Reconstruction With Computer Vision

Calorimetry & Computer Vision

- *(next generation) digital calorimeters: 3D arrays of sensors with more regular geometry*
- *Ideal configuration to apply Convolutional Neural Network*
  - *speed up reconstruction at similar performances*
  - *and possibly improve performances*

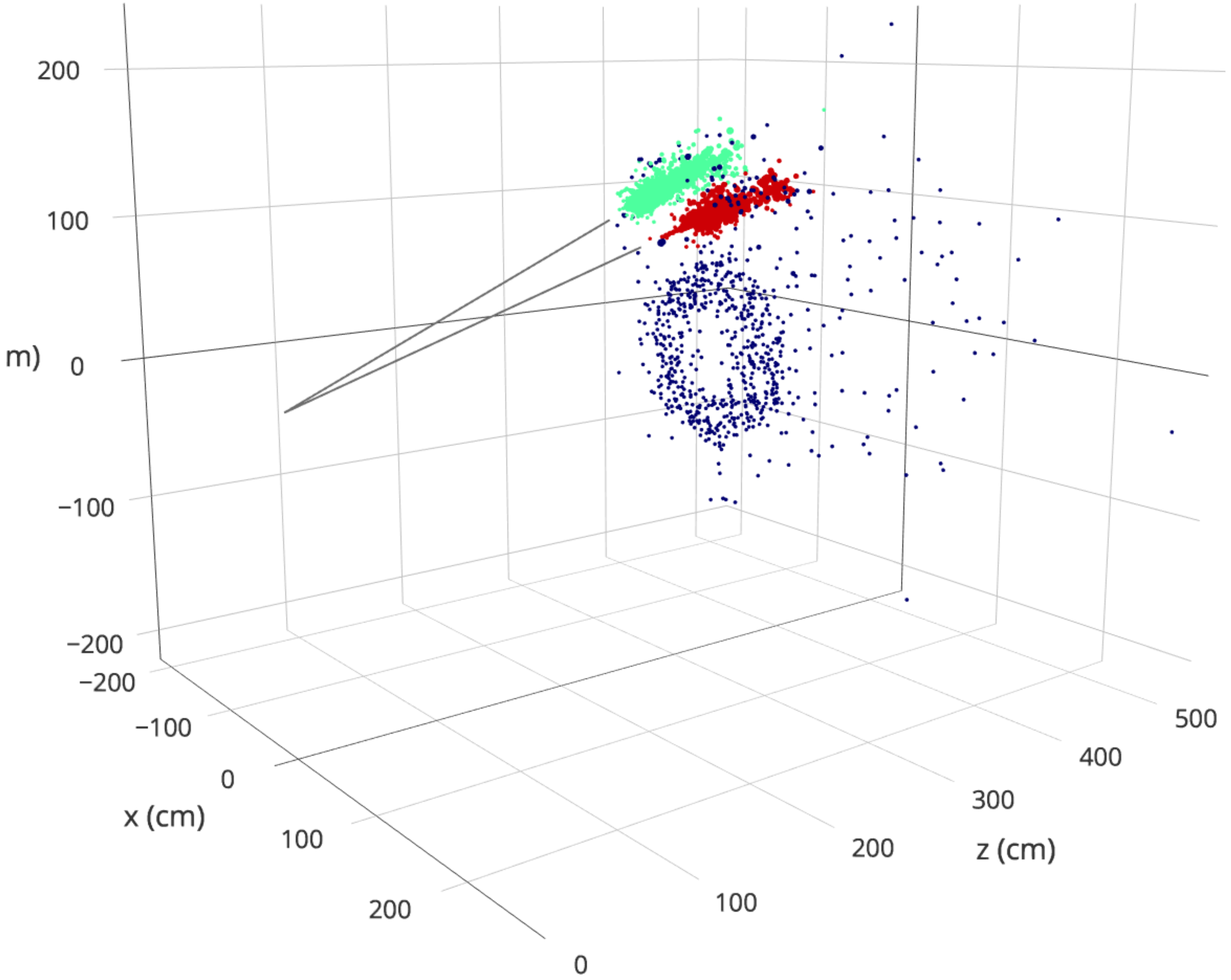


See contribution to NIPS workshop

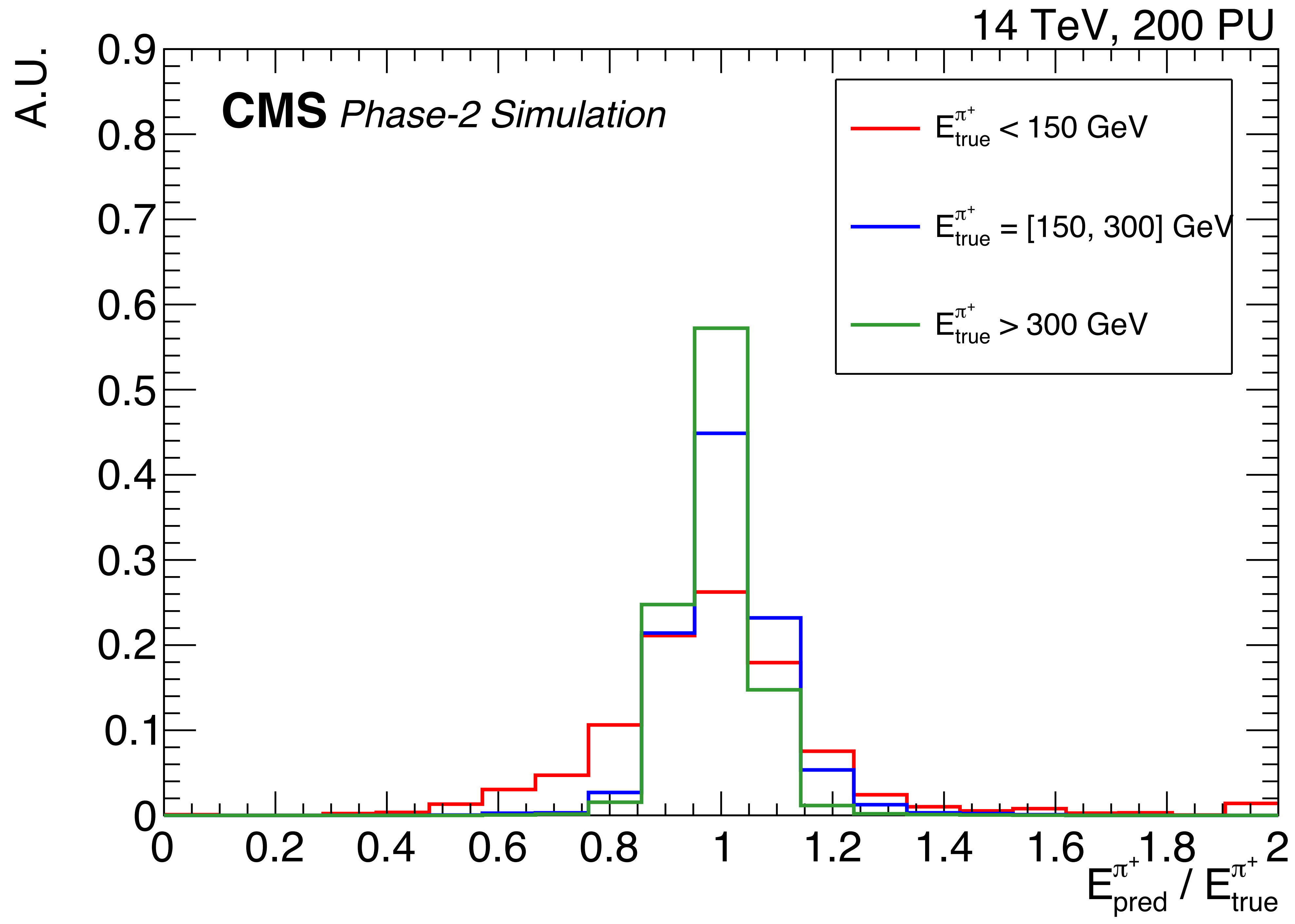
European Research Council

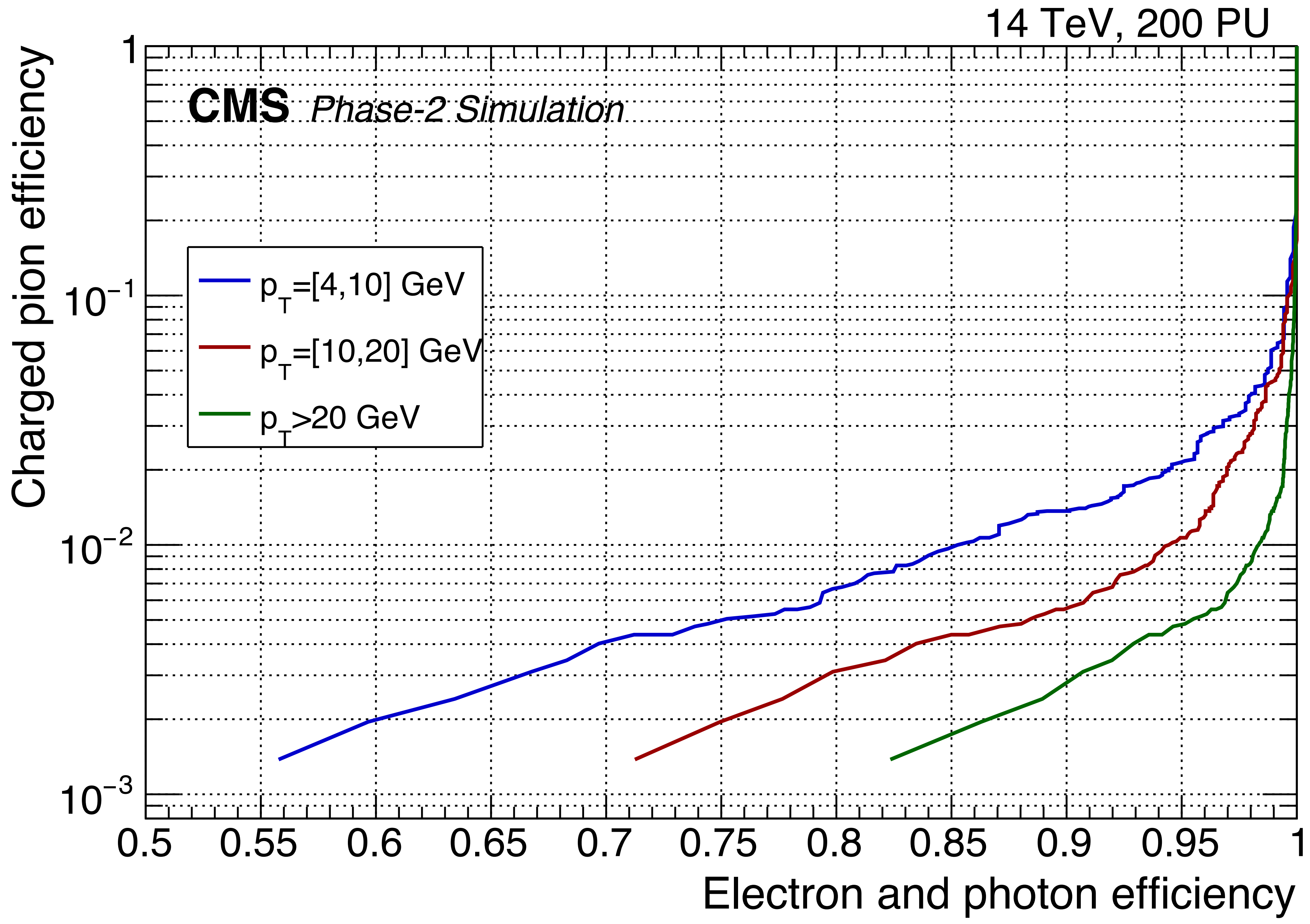
HGCAL: Why Deep Learning

- *High granularity to distinguish individual particles even with many simultaneous collisions*
- *Standard algorithms slowed down by combinatorial*
- *3D Convolutional Neural Networks much faster in going from raw data to answer*
- *Need to develop models to guarantee same performances, possibly better*



HGCAL: DL reconstruction

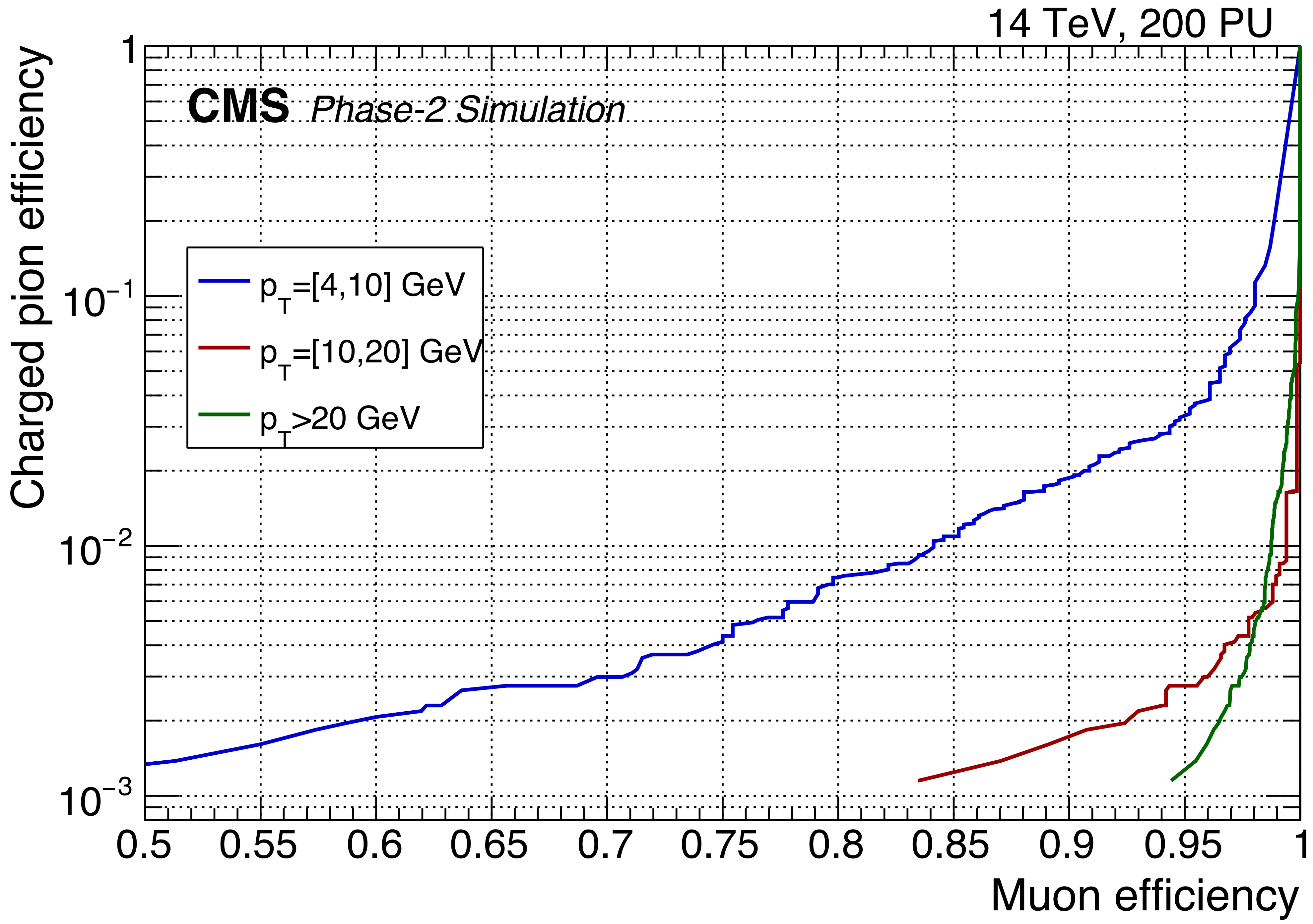


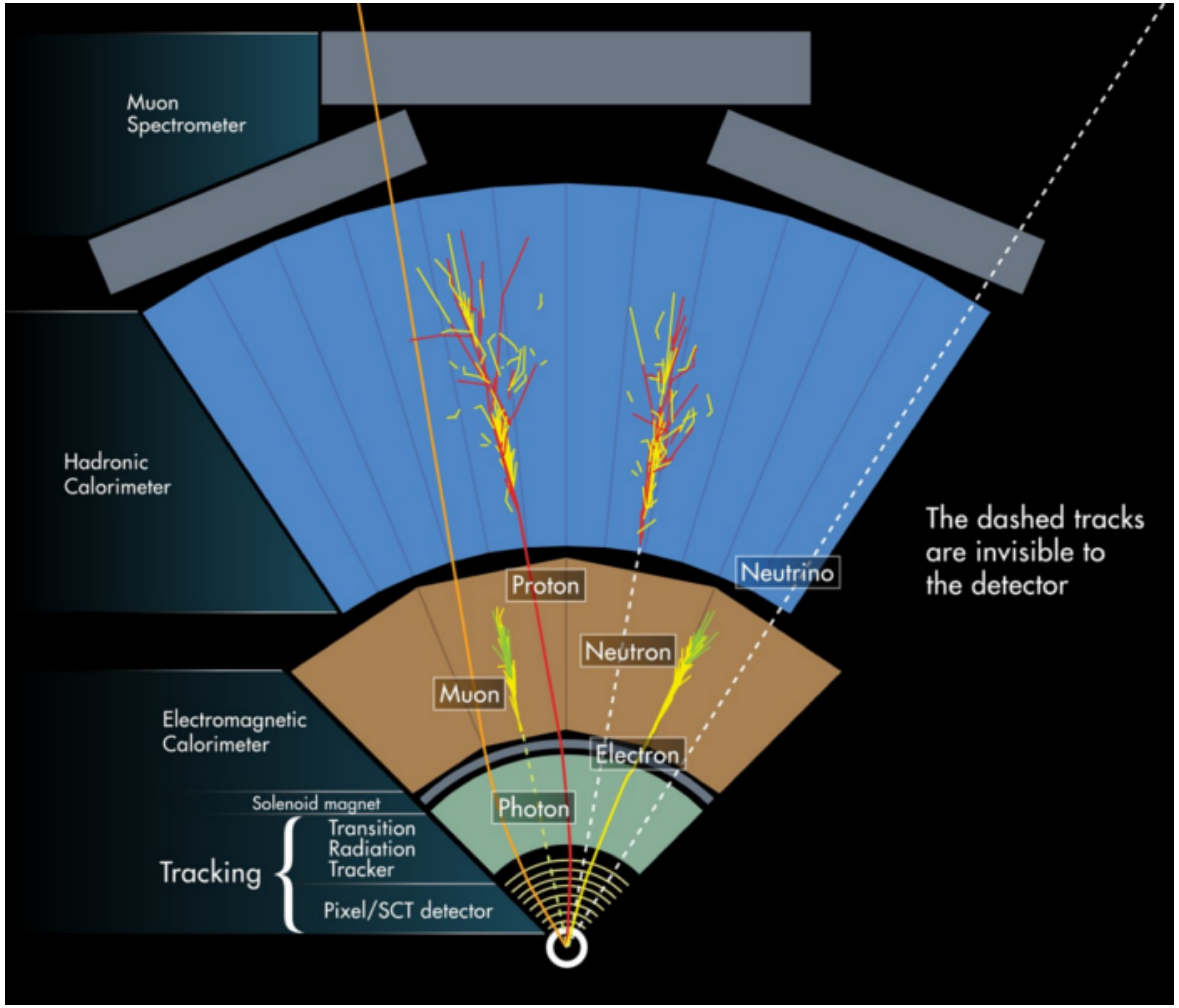


- State-of-the-art performances in terms of particle identification & energy measurement
- Sizeable speed-up at reconstruction time
- Can get even better performances with model optimization

HGCAL: Opportunities

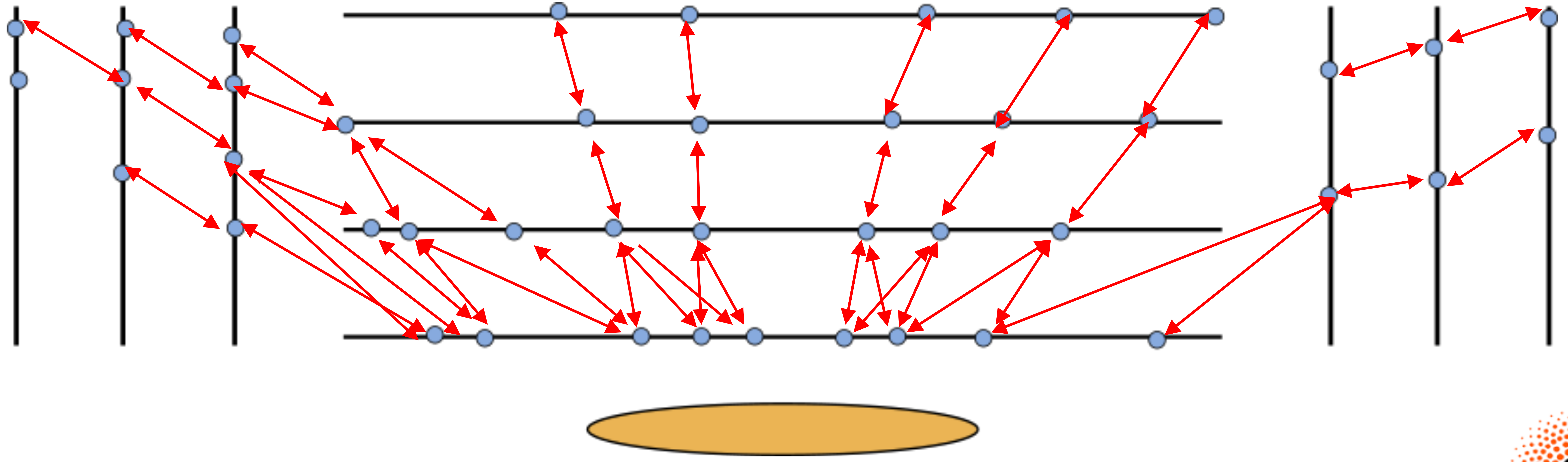
- *New hardware + new techniques = new opportunities & paradigm breaking*
- *Muon reconstruction with calorimeters*





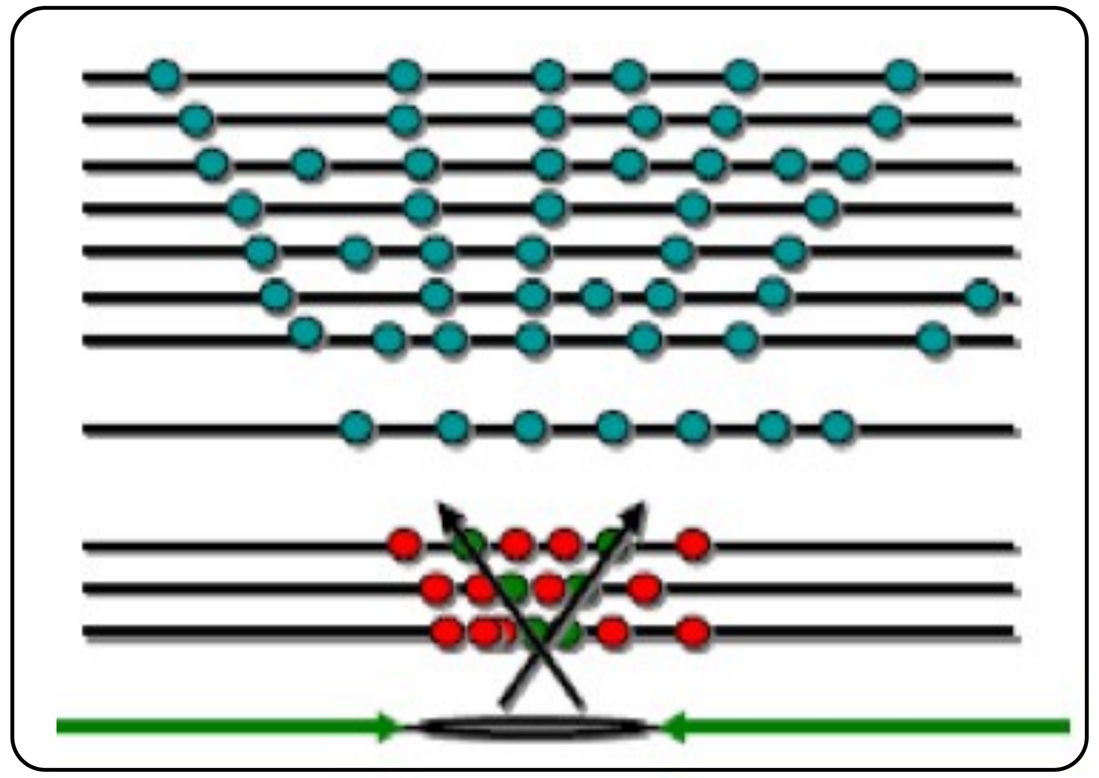
Tracking

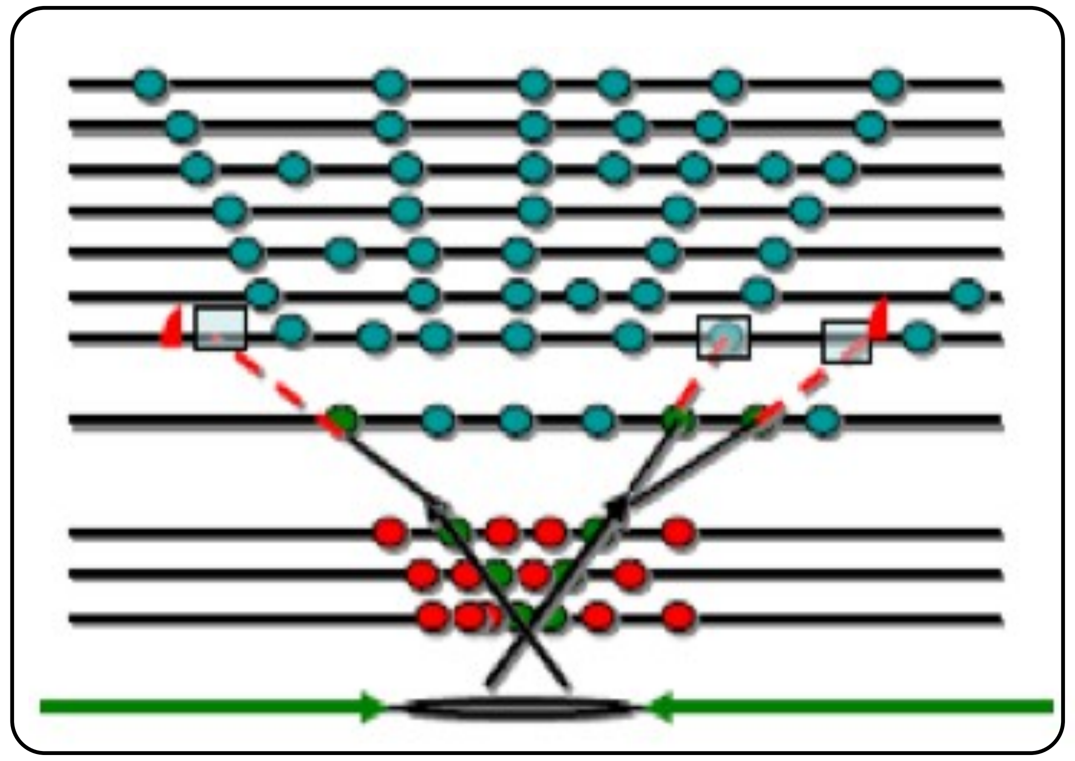
- Tracking is the pattern-recognition task that builds particle trajectories from a set of recorded hits
- One of the slowest tasks we perform to reconstruct particles in LHC collisions
- Non-linear slow-down with number of simultaneous collisions, due to combinatoric effects

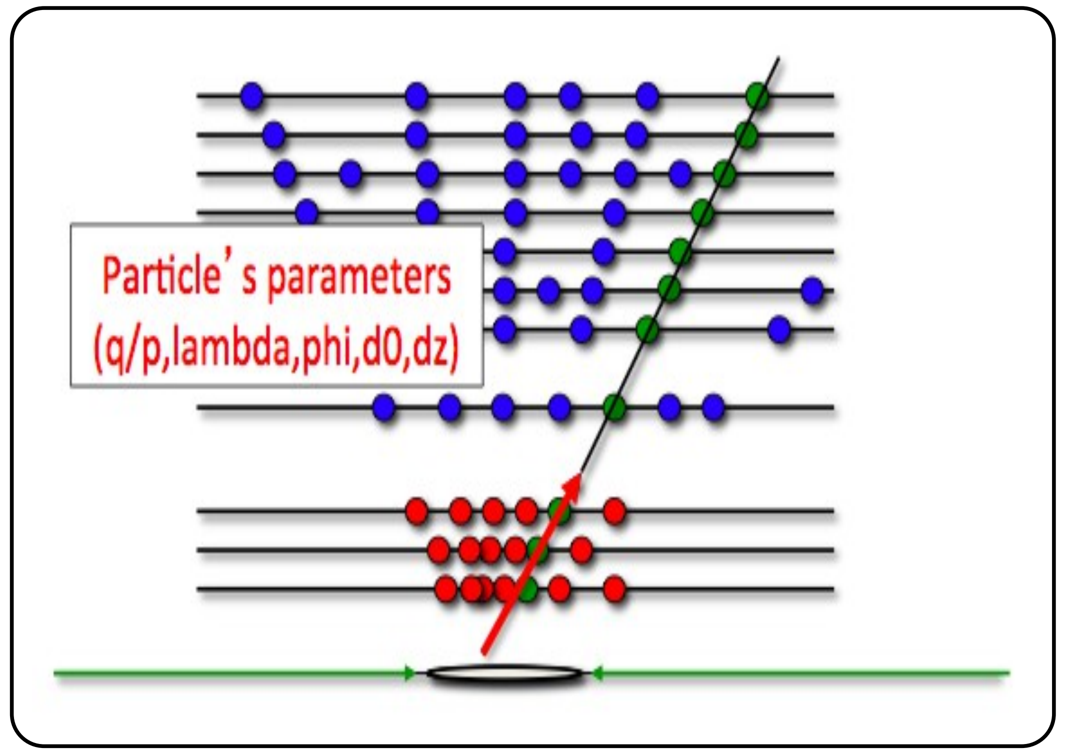


Tracking

- Works in three steps
  - seeding: start from pair of hits in the inner detector
  - hit-to-track association: propagate the seed and look for hits close ton the predicted trajectory
  - Track fitting: measure the track parameters (particle energy) from a fit of the points to an helix trajectory

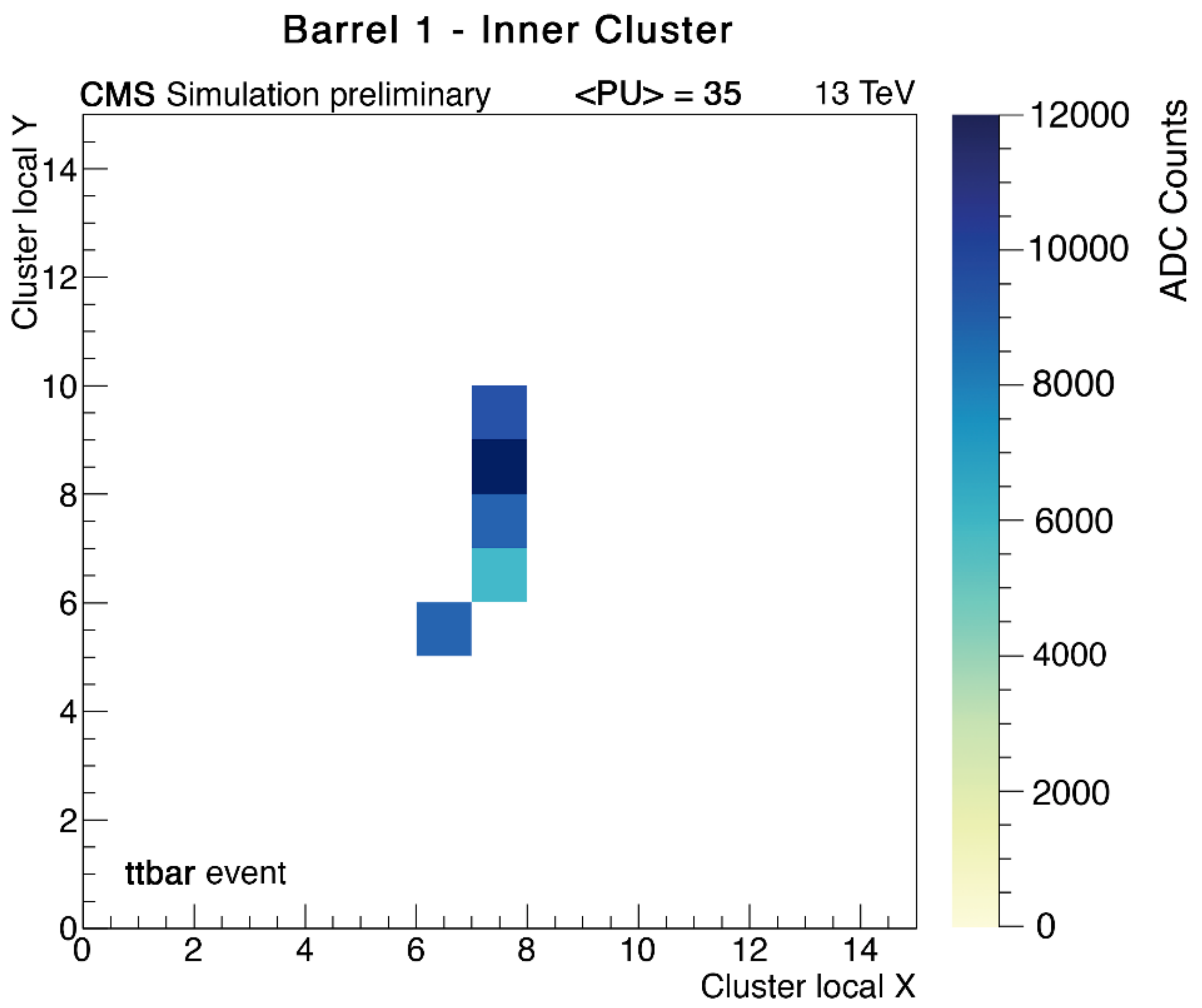


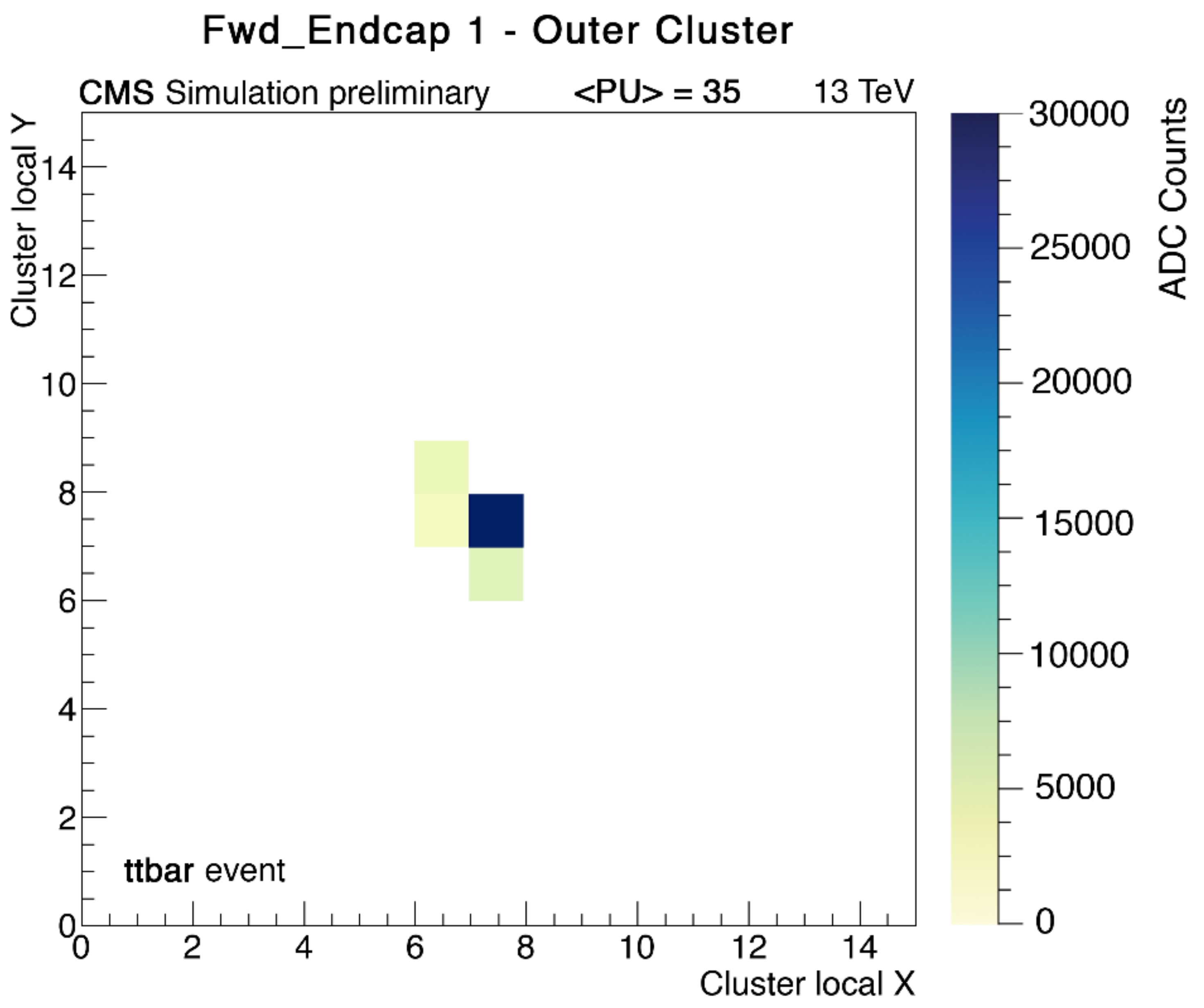




Deep Learning to the Rescue

- *The detector sees the charge deposited by the crossing particle: a hit*
- *A hit is a window of sensors (16x16 here) with its deposited charge. This can be seen as a sparse digital image.*
- *Given two images, one can train a network to decide if a pair of hits is a good or bad match*

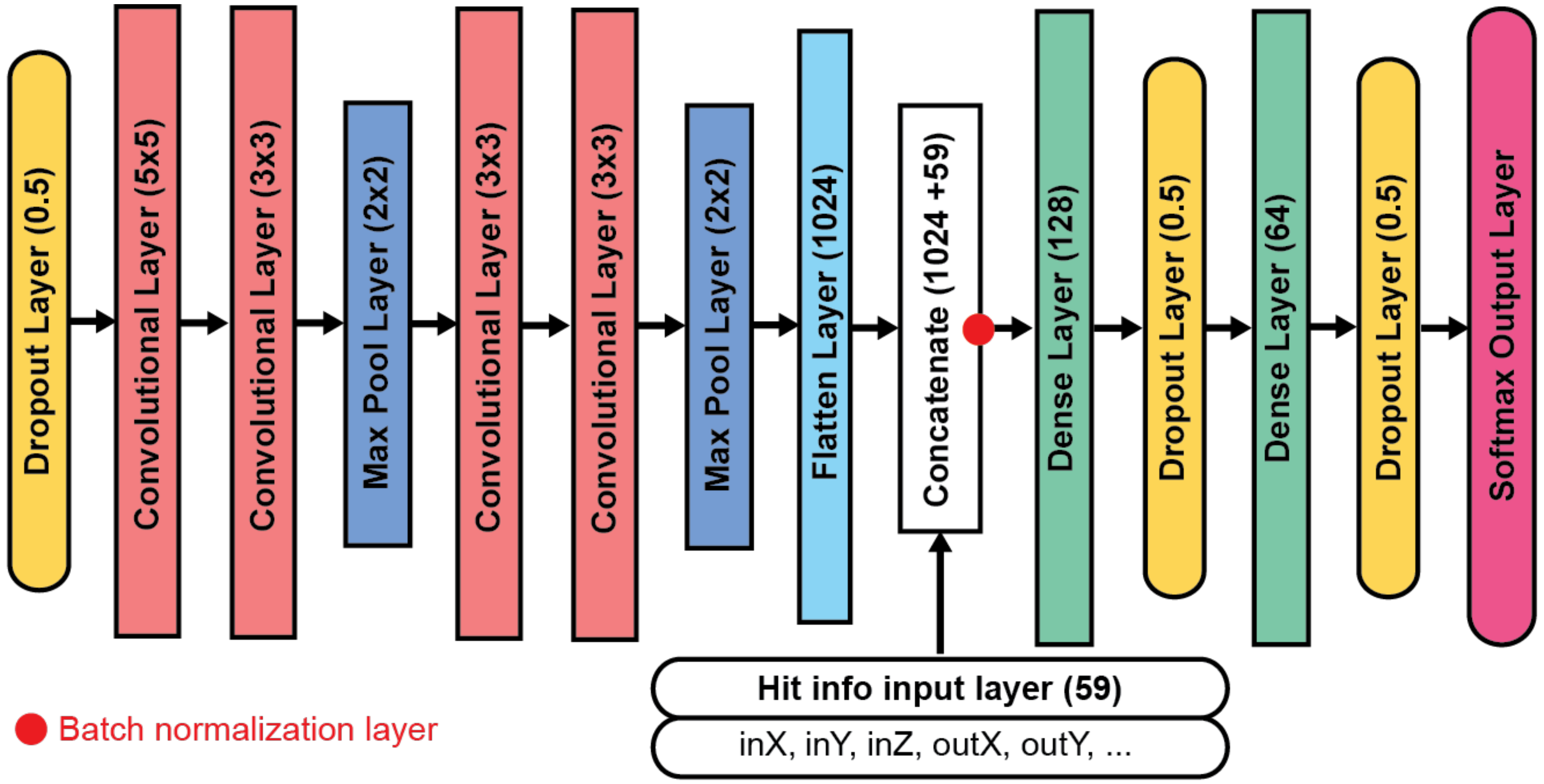




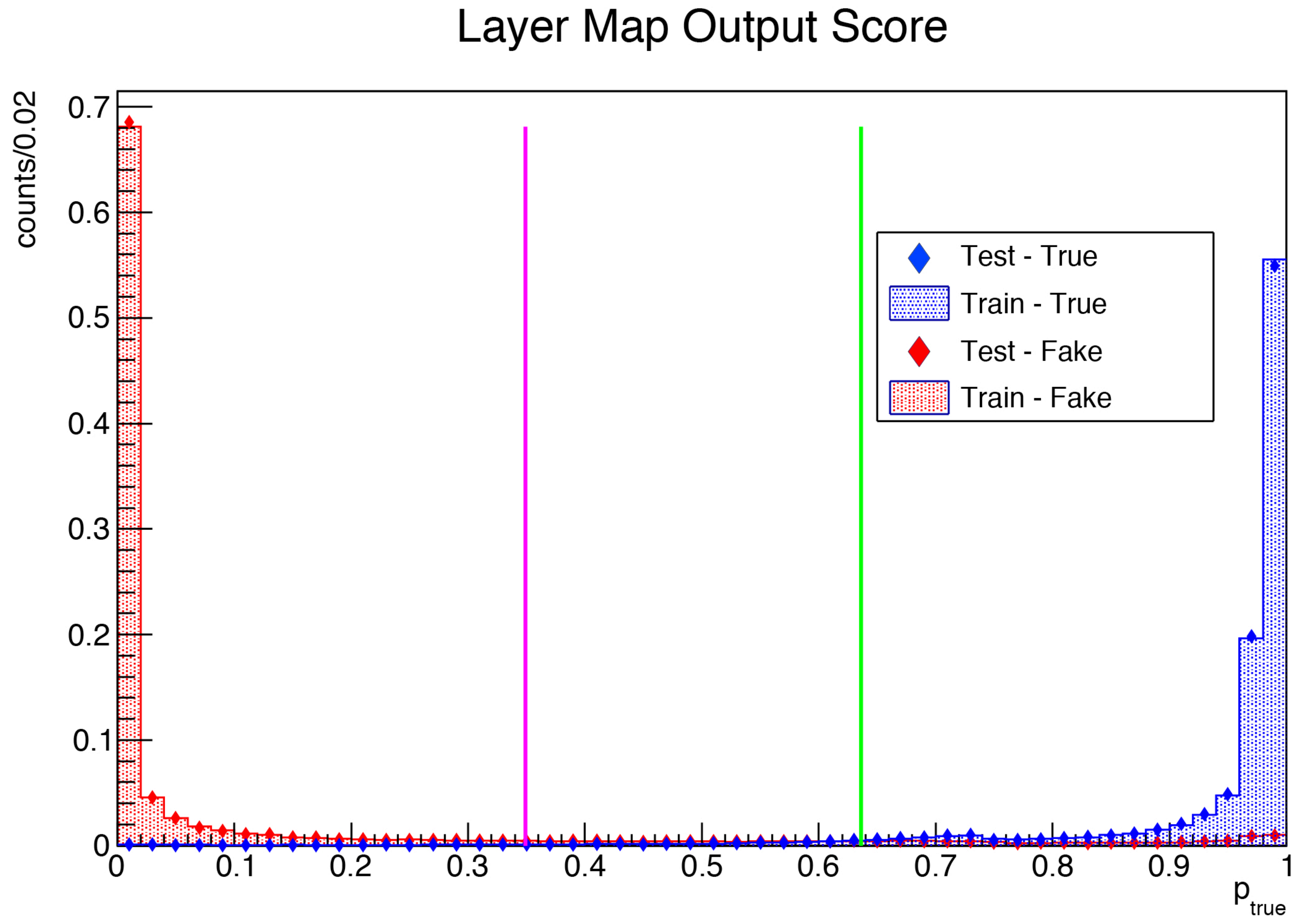
PixelSeed ConvNN

- *The final model uses two sets of inputs:*
  - *the hit images*

  - *a set of expert features (e.g., position of the hits in the detector) to help the learning process*

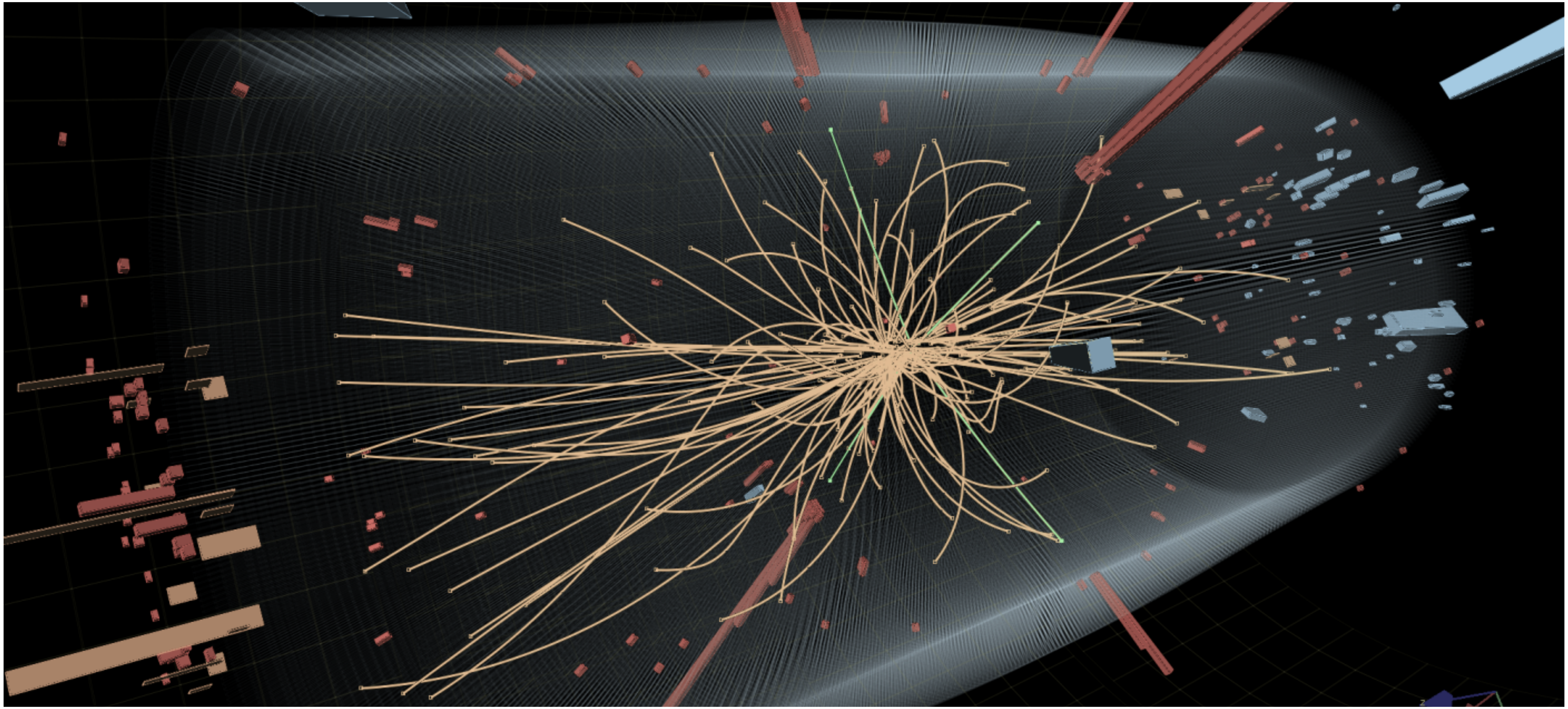


- *The trained model shows a good separation of true vs fake seeds*



- ***One can reduce the fake rate by one order of magnitude with a few % loss in efficiency***

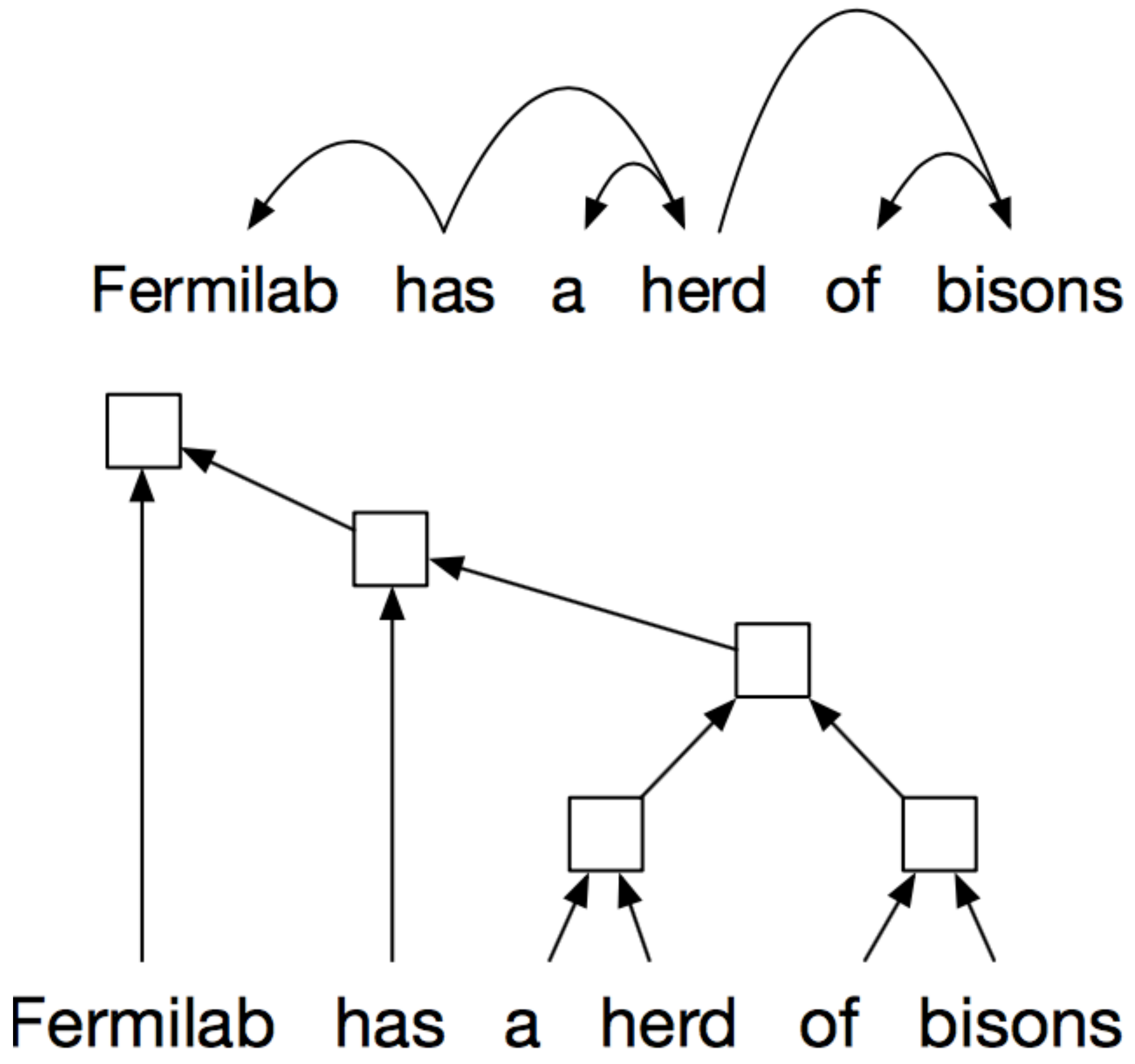
```
Efficiency (tpr) @ fake rejection
tpr @ rej 50%: 0.998996700259
tpr @ rej 75%: 0.990524391331
tpr @ rej 90%: 0.922210826719
tpr @ rej 99%: 0.338669401587
```

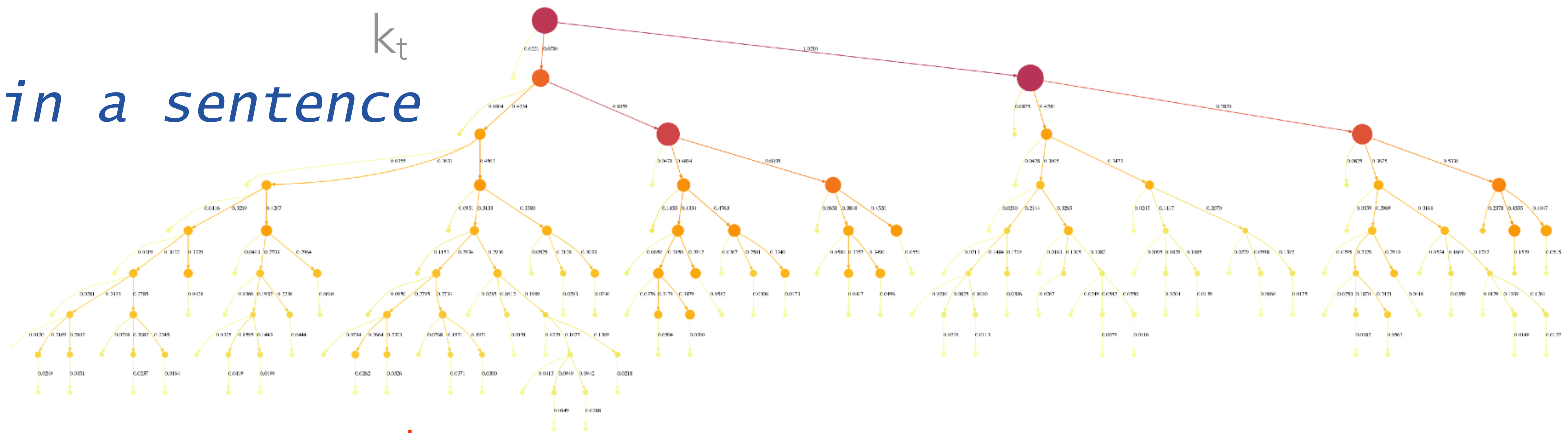


HEP & Language processing networks

Particle (language) processing

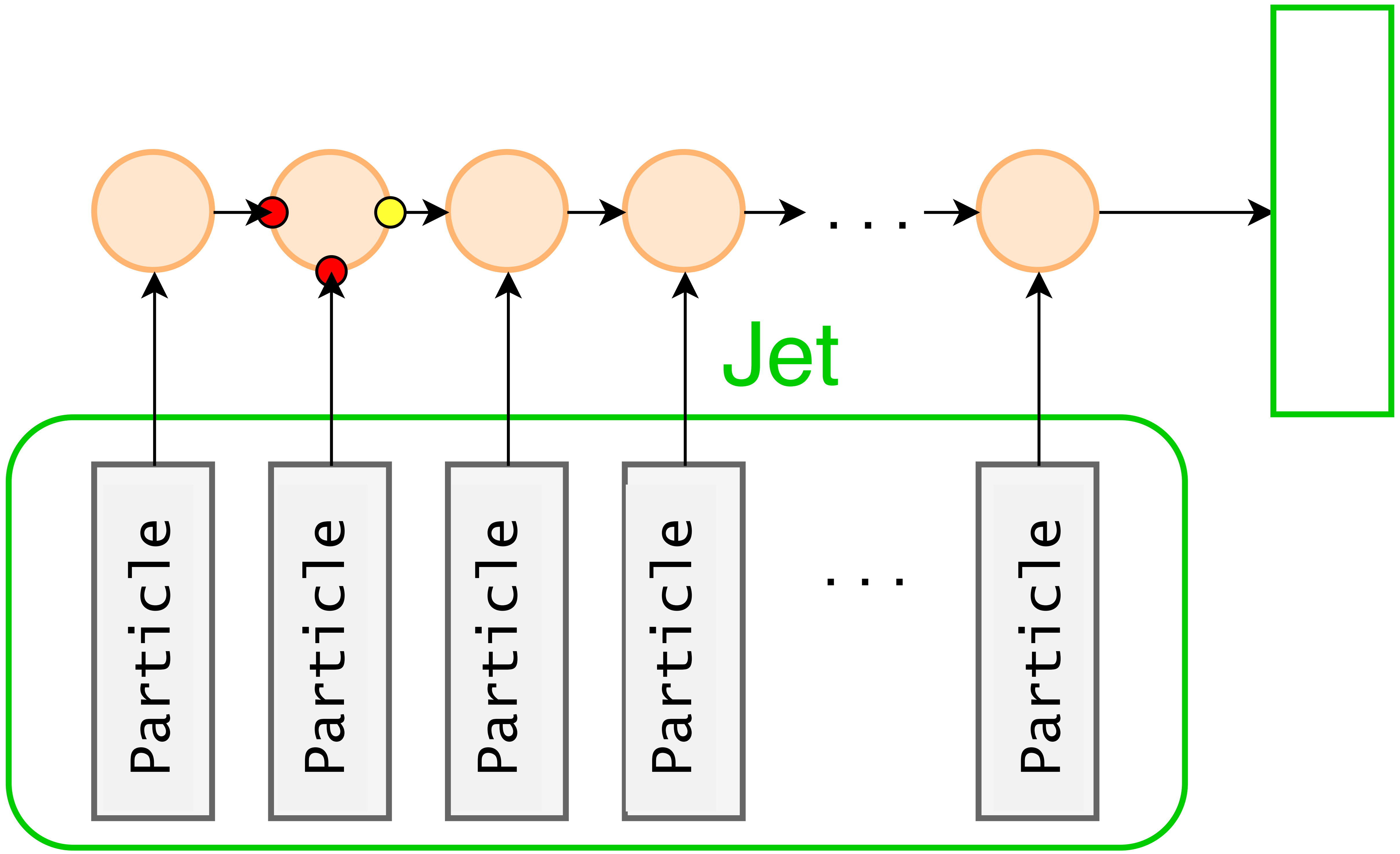
- *CMS uses particle flow for event reconstruction:*
  - *At some point in the central processing, collision images are turned into a list of particles.*
  - *From these particles, complex objects (e.g., jets) are formed*
- *In this framework, Computing vision approaches are not necessarily ideal*
- *One can instead use language-processing approaches (e.g., recurrent neural networks*
  - *particles are words in a sentence*
  - *QCD is the grammar*





Recurrent Neural Networks

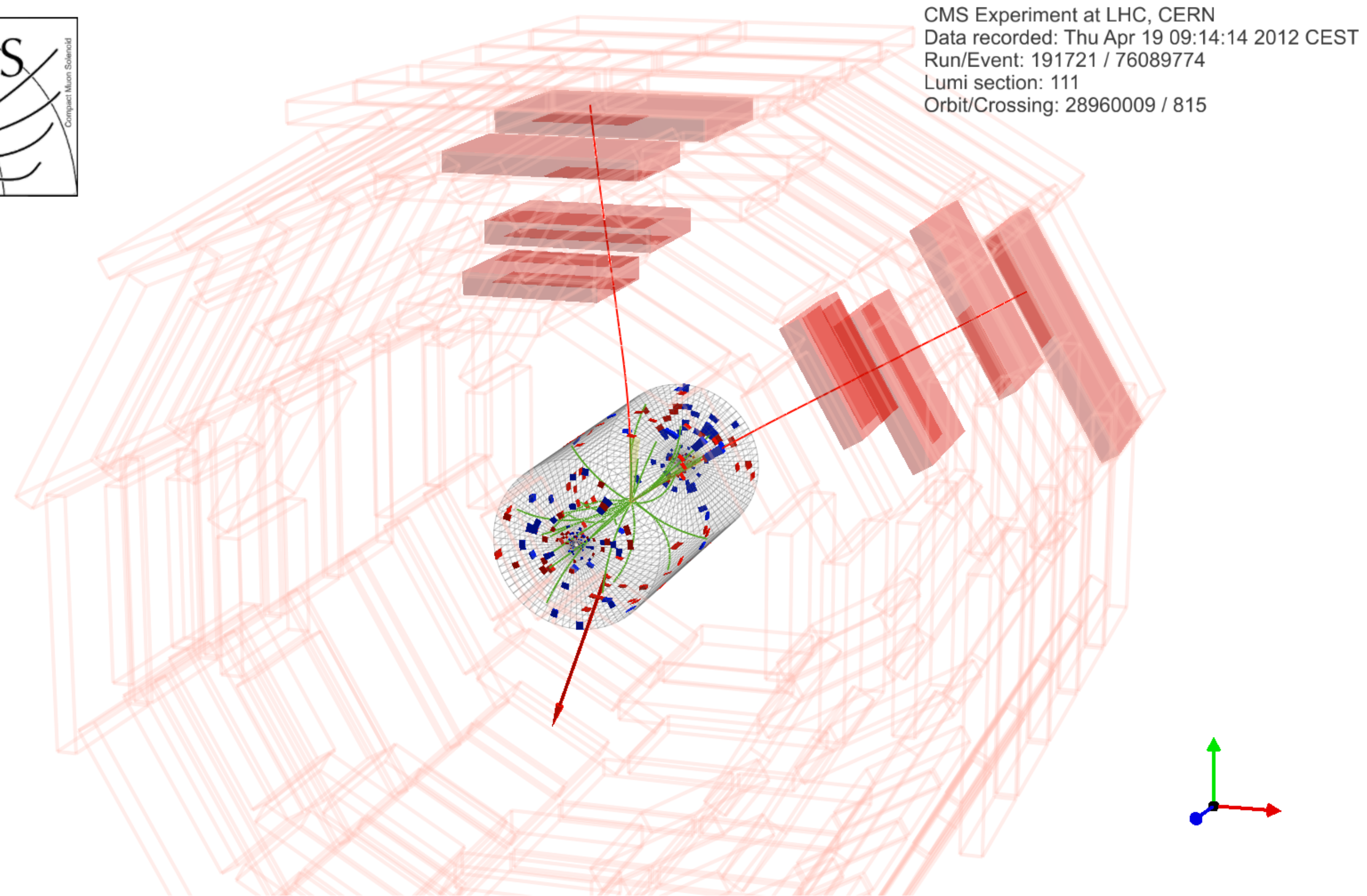
- *A network architecture suitable to process an ordered sequence of inputs*
  - *words in text processing*
  - *a time series*
  - *particles in a list*
- *Could be used for a single jet or the full event*
- *Next step: graph networks (active research direction)*

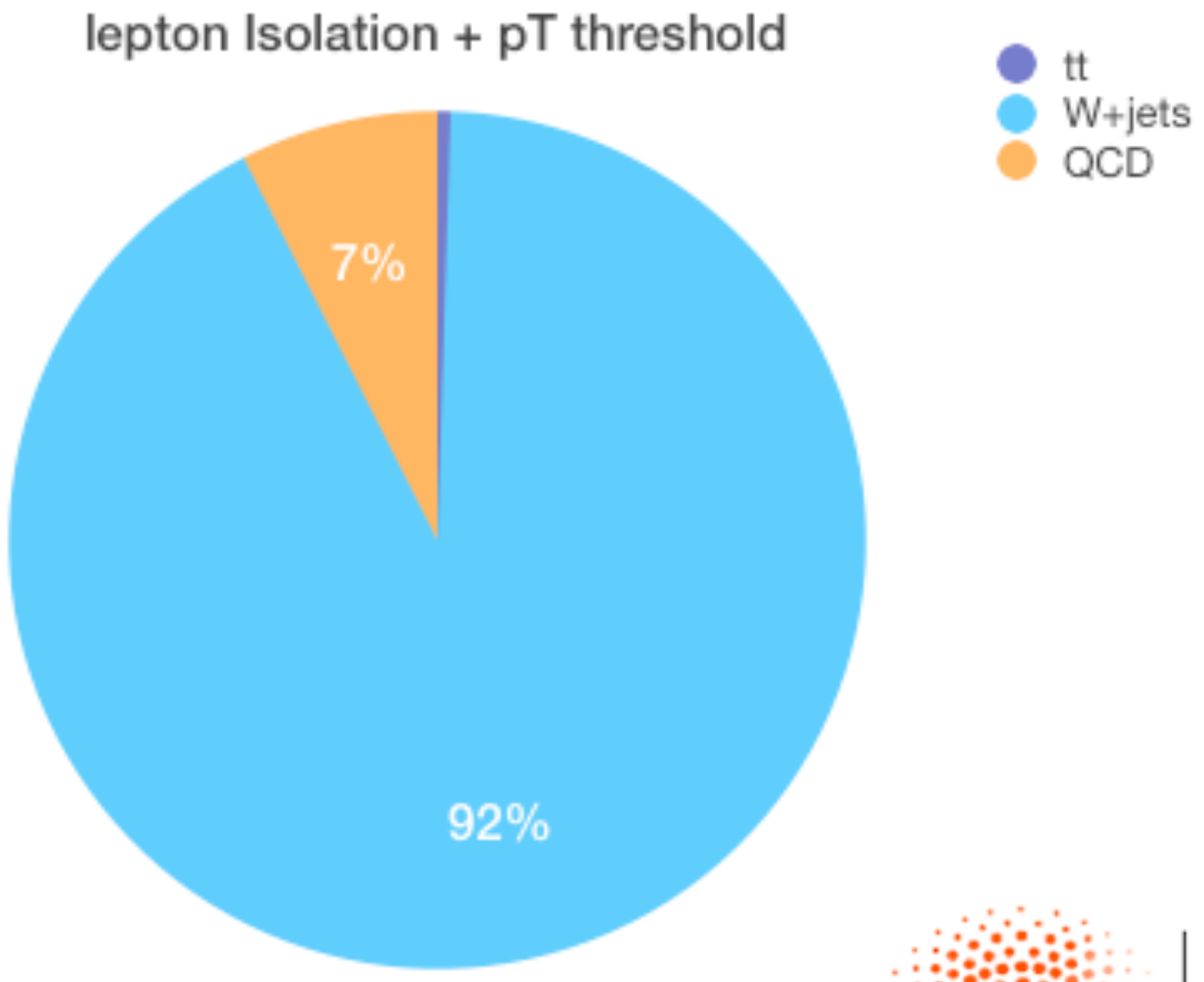


A Topology Classifier

A typical example: leptonic triggers

- *at the LHC, producing an isolated electron or muon is very rare. Typical smoking gun that something interesting happened (Z,W,top,H production)-> **TAKE THEM!***
- *Triggers like those are very central to ATLAS/CMS physics*
- ***The sample selected is enriched in interesting events, but still contaminated by non-interesting ones***
- *Can we clean this up w/o biasing the physics? yes, with ML*

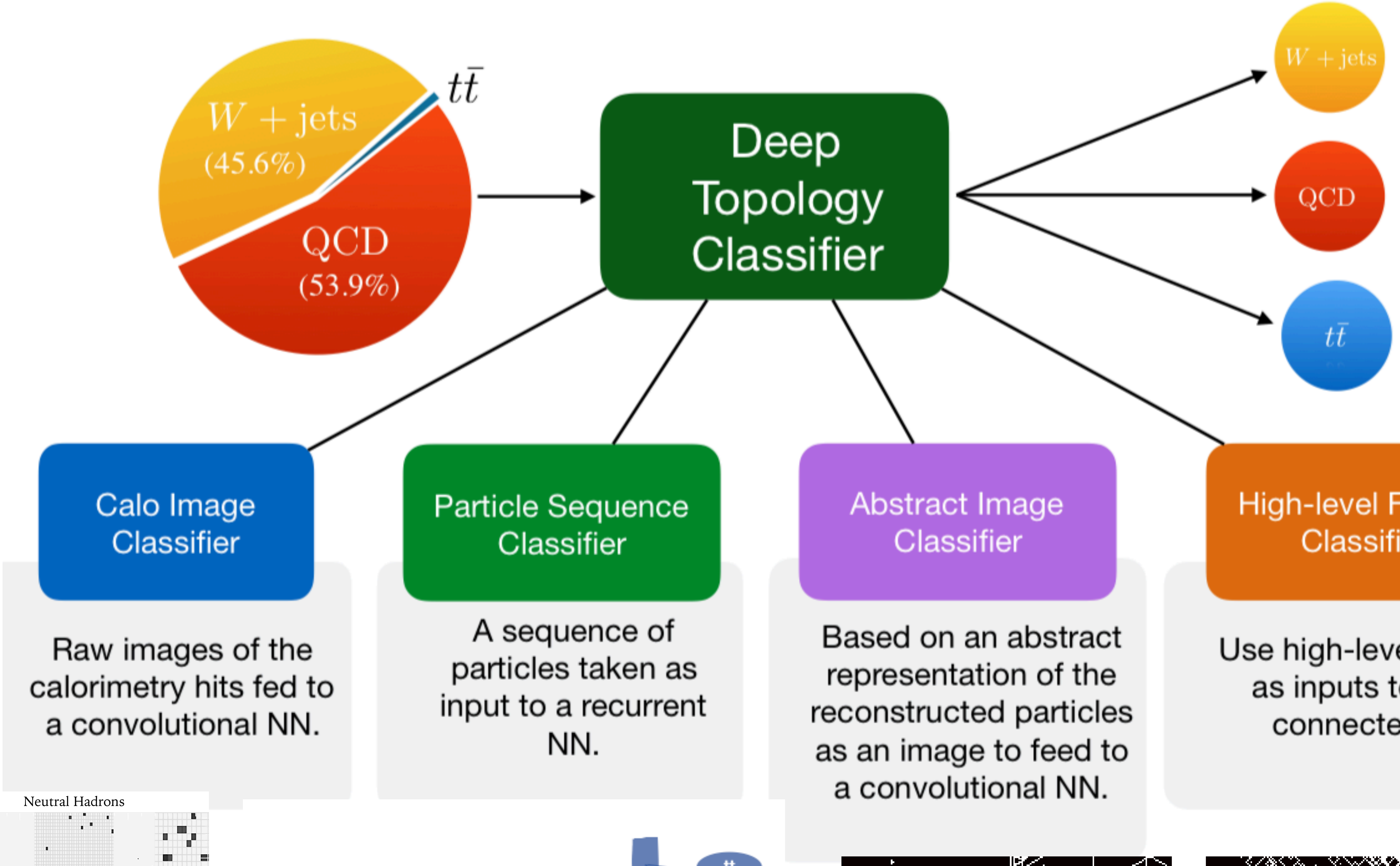


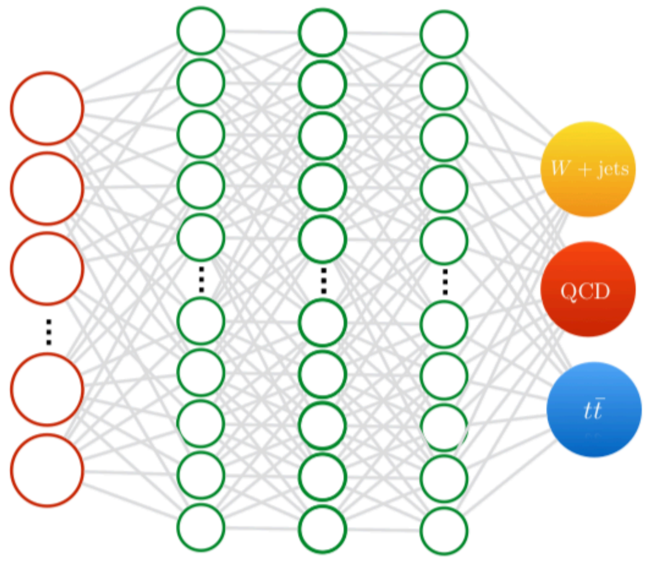


See contribution to NIPS workshop

European Research Council

A Topology Classifier





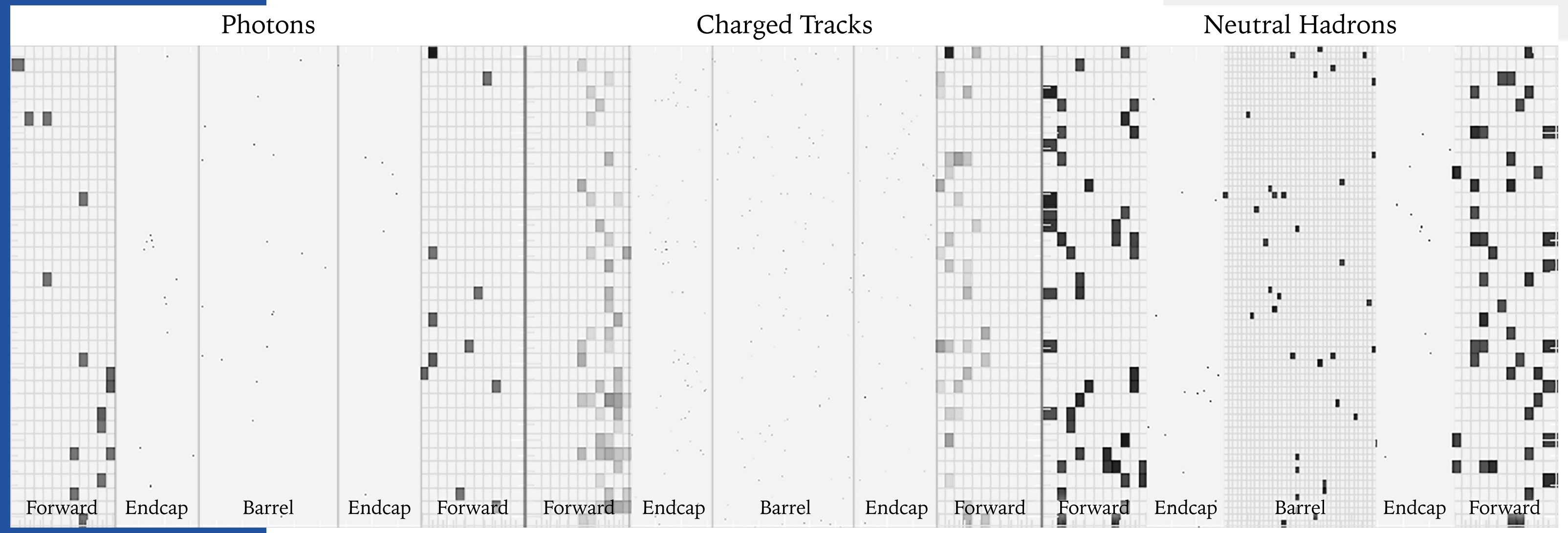
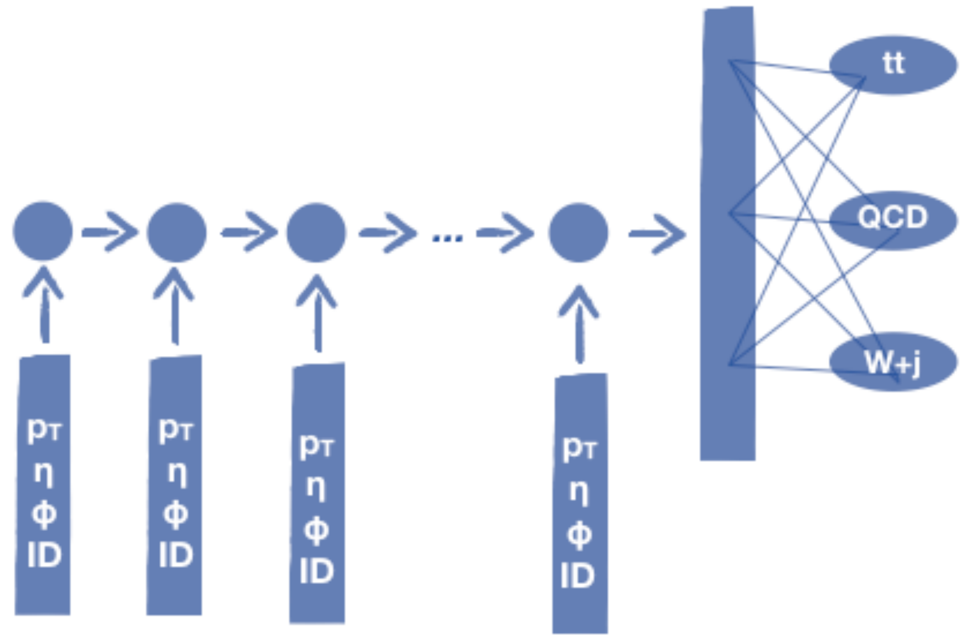
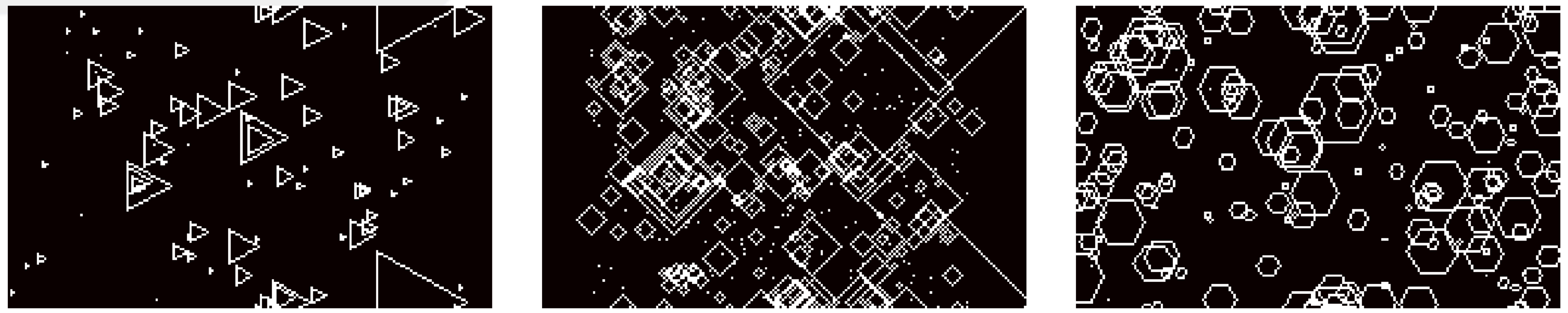


Figure 2: An example of a $t\bar{t}$ event as the input of the raw-image classifier.

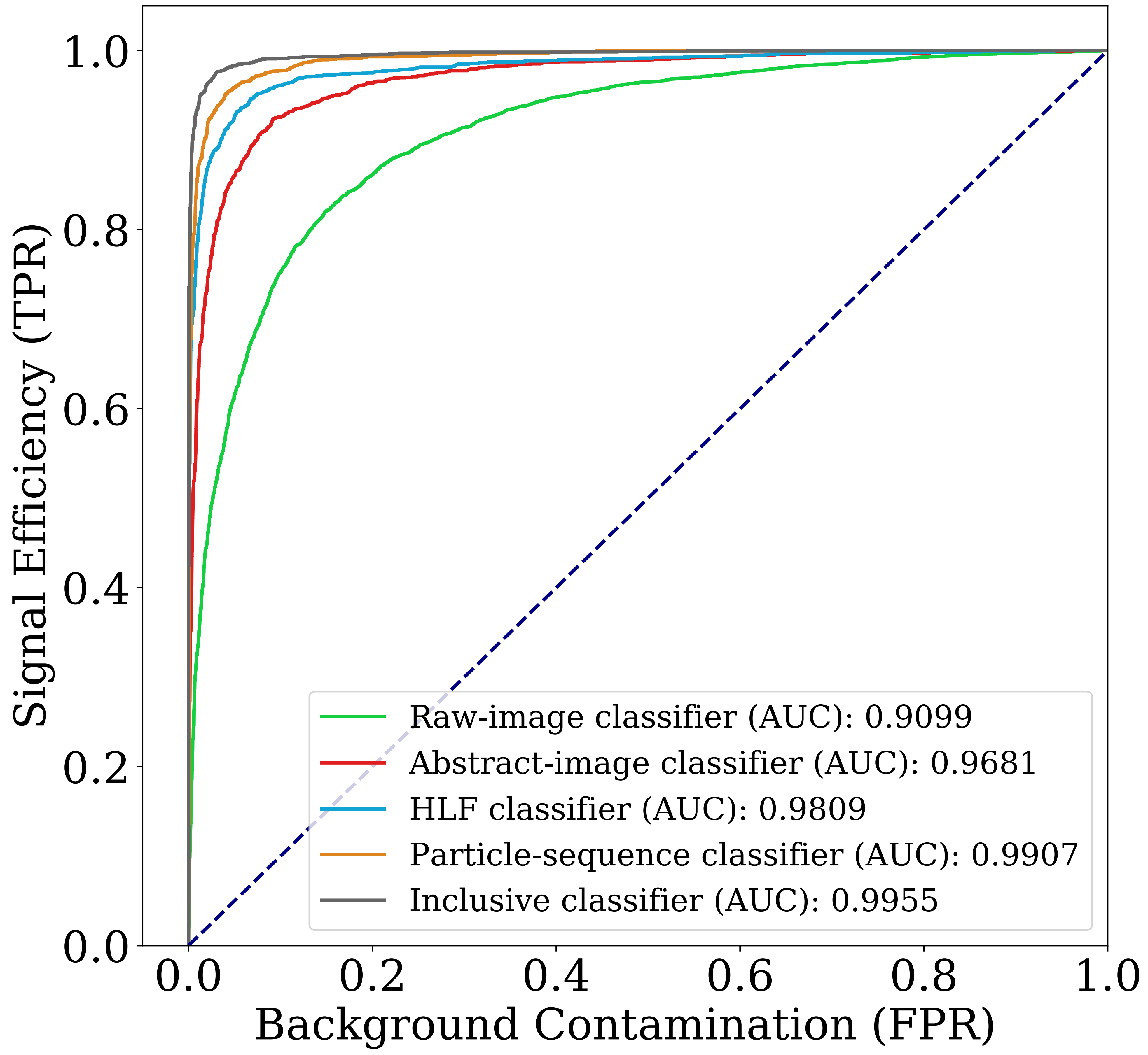


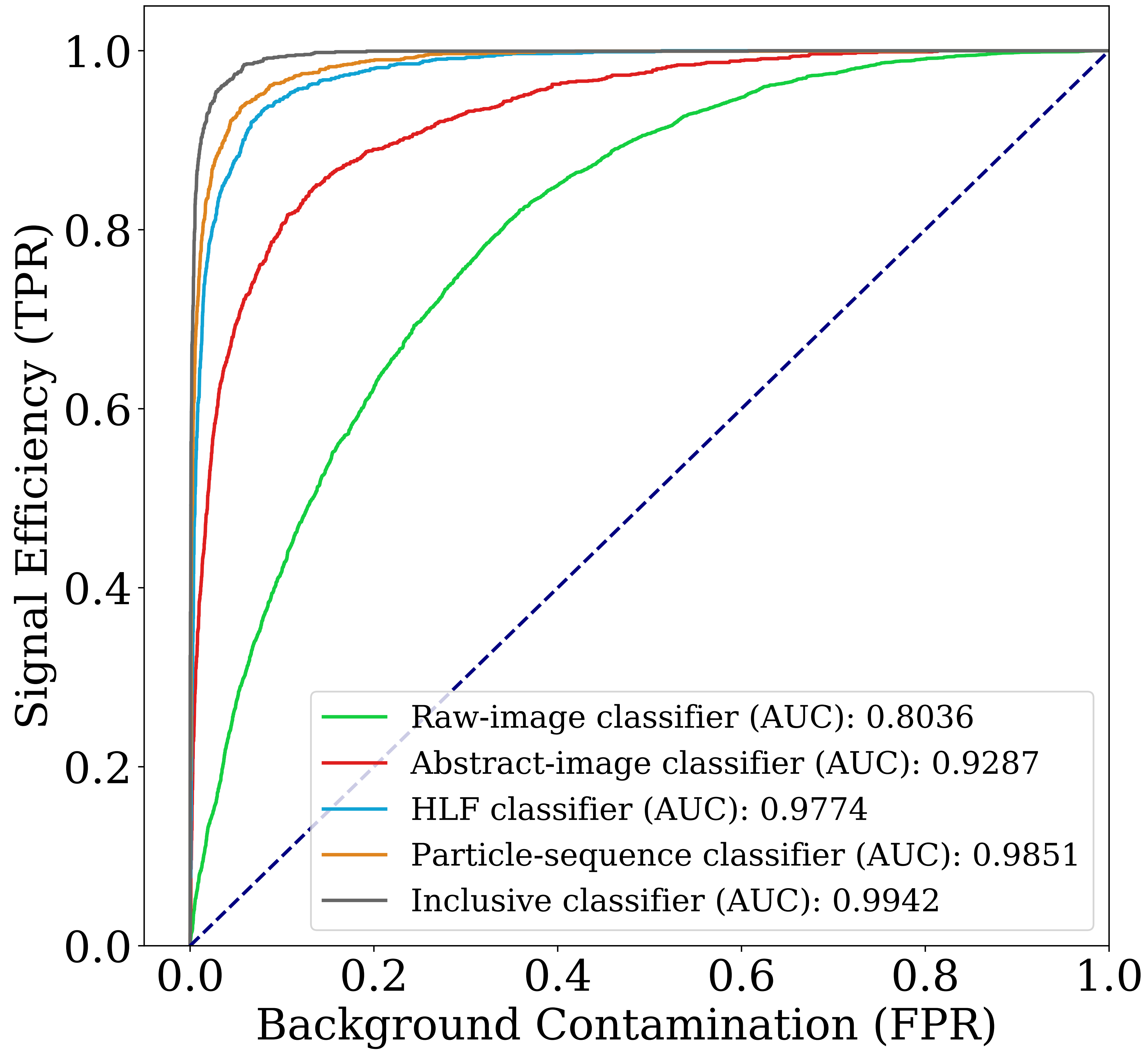


(a) Photons (b) Charged Particles (c) Neutral Hadrons

European Research Council

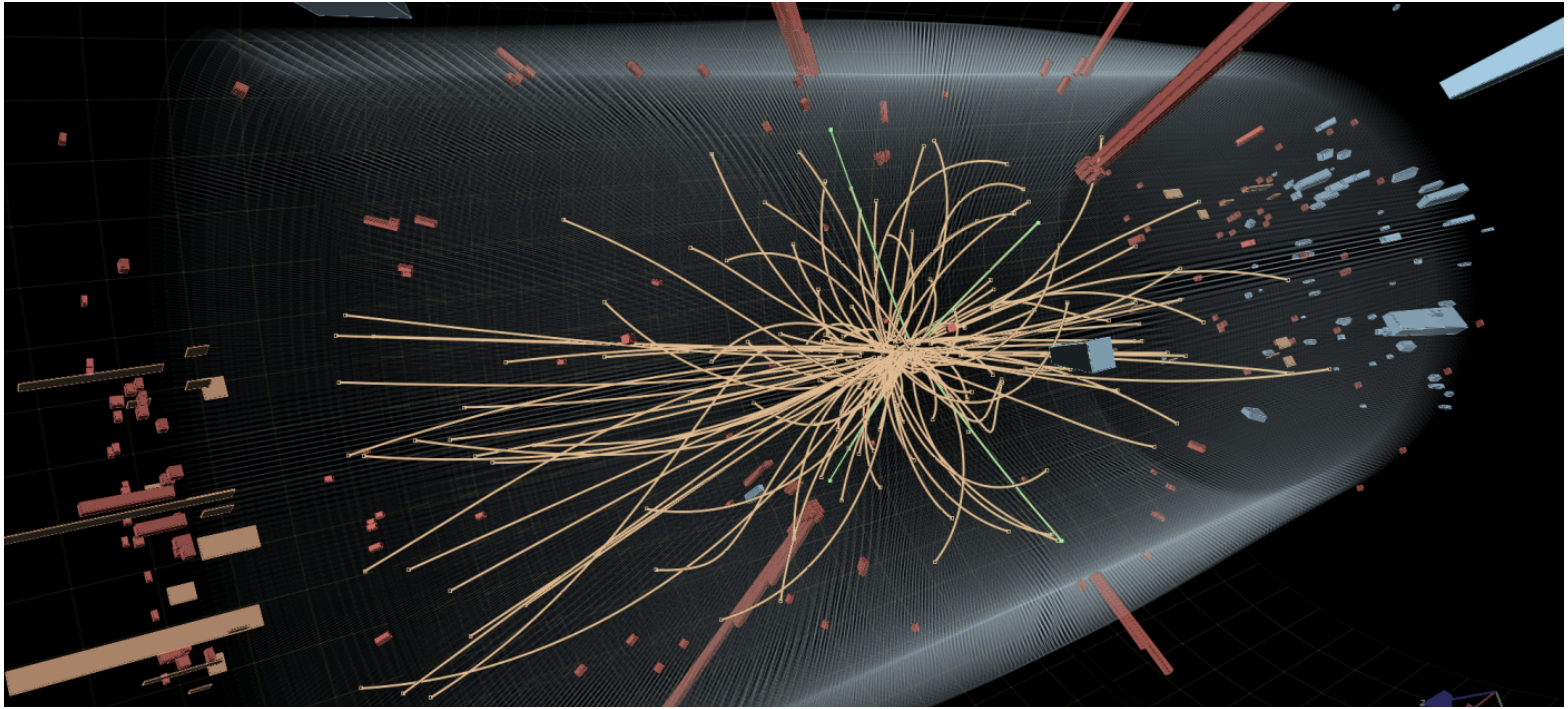
Selection performances





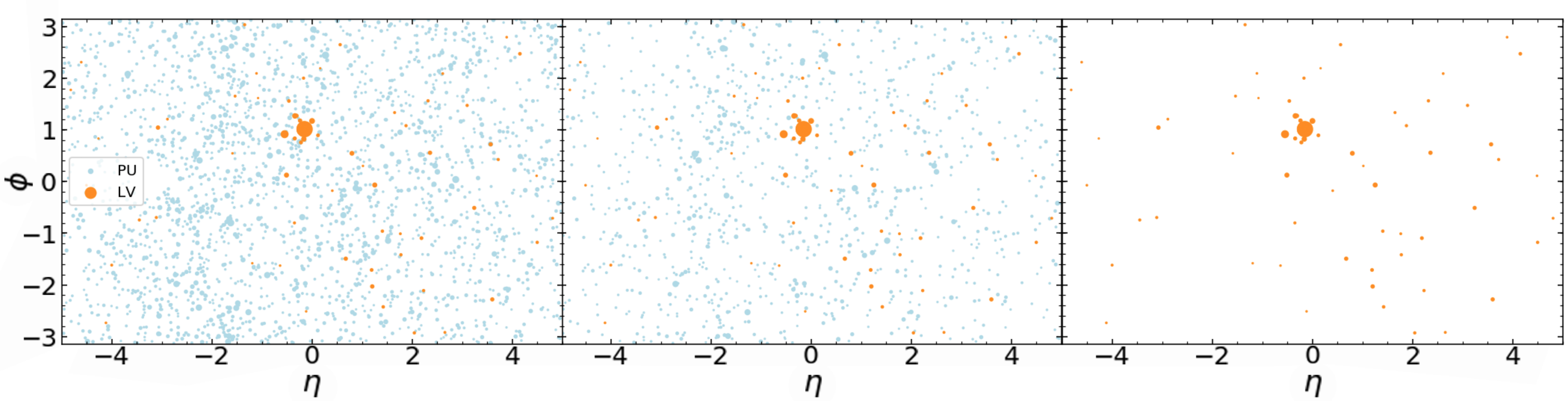
Can select 99% of the top events and reduce the fraction of written events by a factor ~ 7

European Research Council



Graph Networks for particle physics

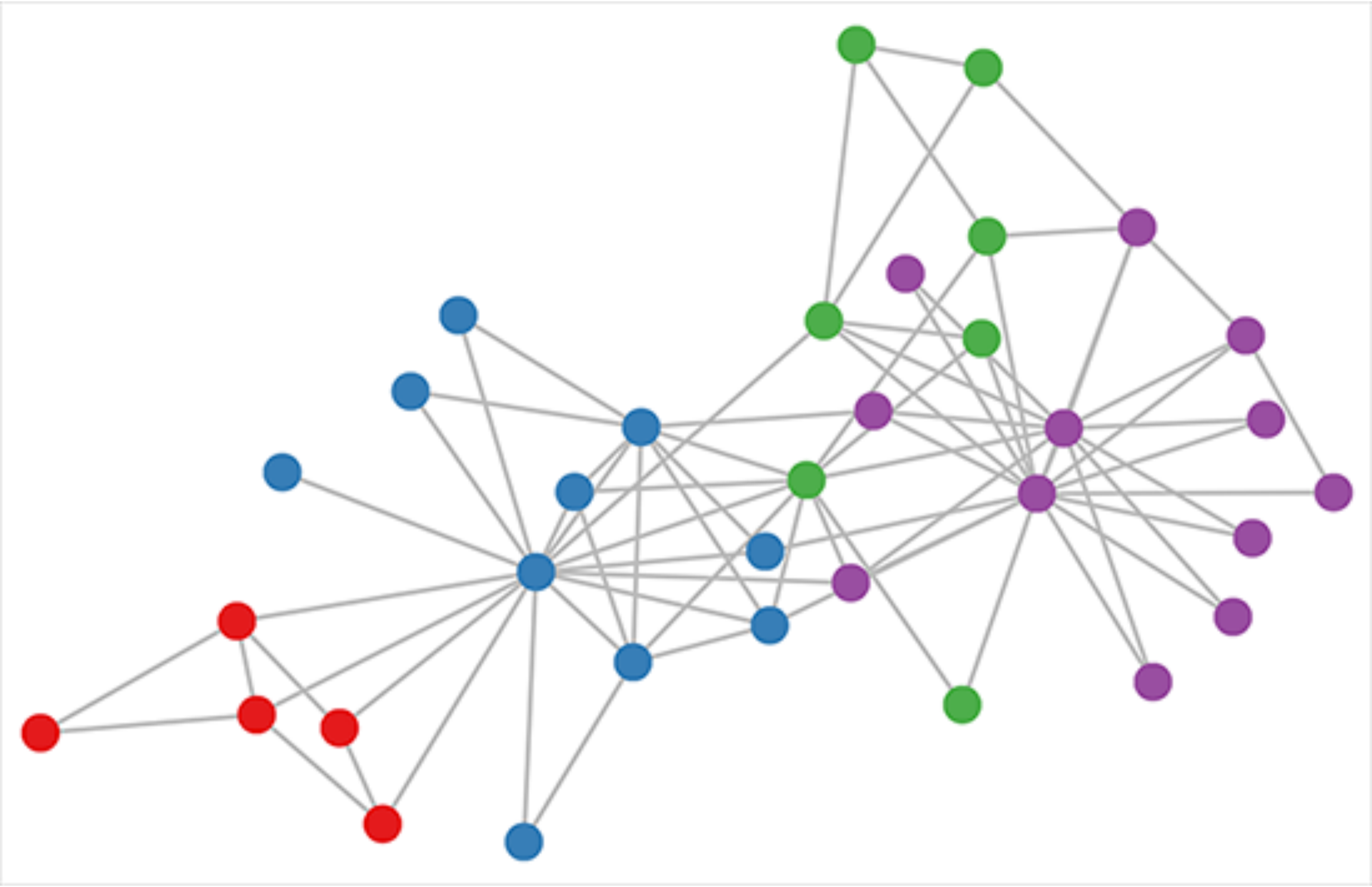
The pileup problem

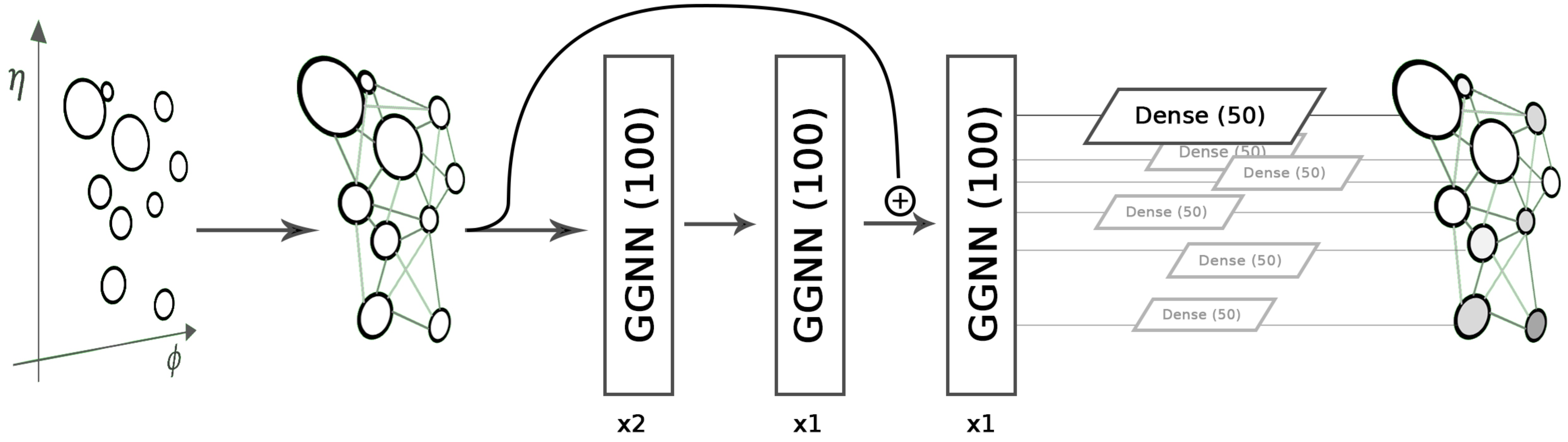


- Pileup introduces noise to any reconstruction algorithm
- Usually, one runs a PU subtraction algorithms first
  - Usually based on global information of the event (average occupancy vs observed local occupancy)
  - OK offline, sort of OK @HLT, complicated @ L1
- State-of-the-art algorithms (Softkiller, PUPPI) improved situation dramatically wrt Run I
  - Now ATLAS and CMS deal with ~40 collisions/bunch crossing with no problems

Graph Networks

- Graph networks can be seen as generalization of Conv NN
  - Network learns from single "pixel" (graph node) and its neighbours
  - The concept of neighbour is not driven by geometrical proximity
  - Instead, what is "close" and what is not depends on connections (graphs) which are learned in the training
- We used a Gated Graph NN to decide if a given particle is from PU or not, based on its neighbours charged particles (which can be tracked to a vtx) are pileup or not

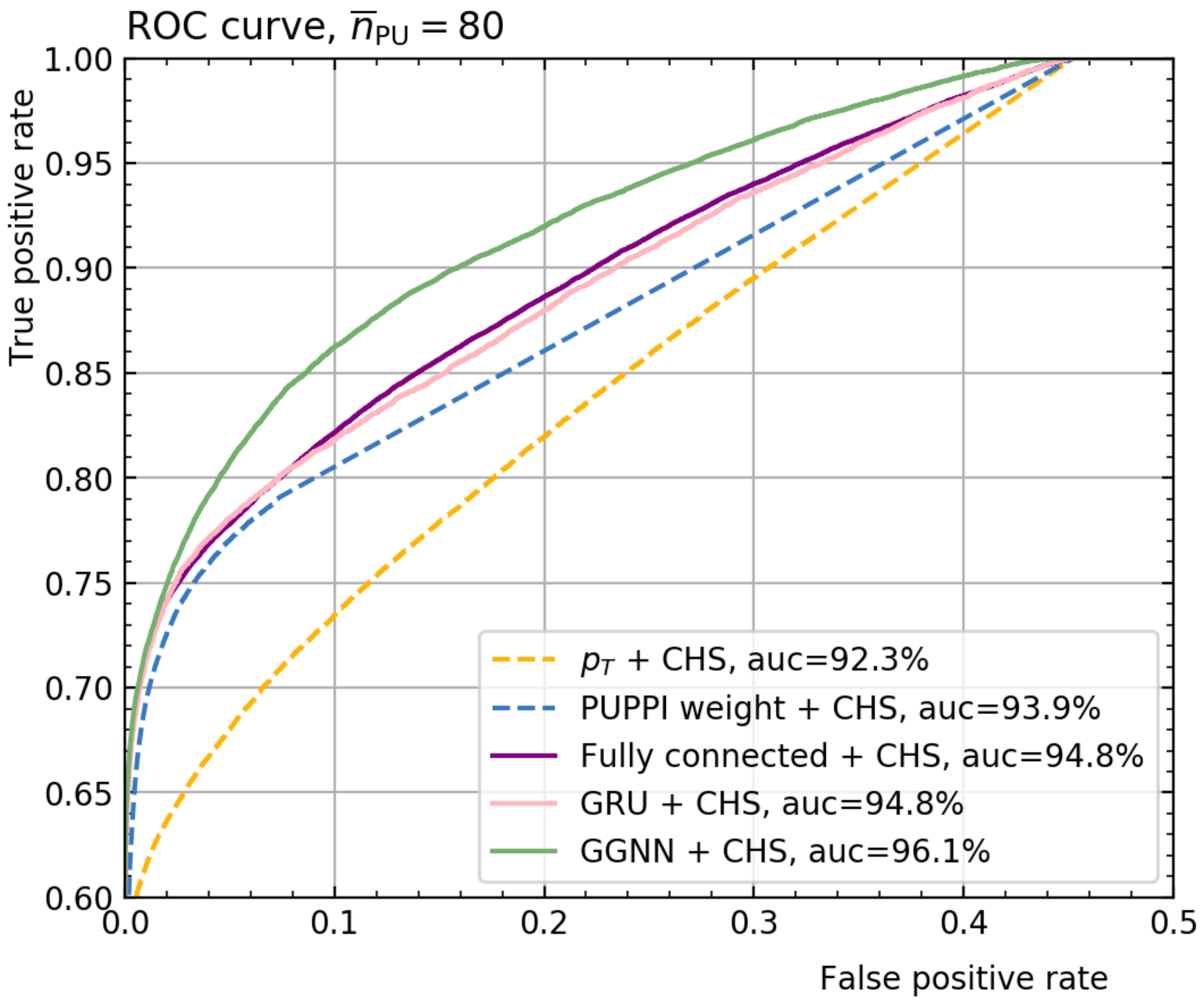




PUPPIML: Graph Nets for PU subtraction

- *Improve state-of-the-art algorithms substantially*
- *Little dependence of algorithm tuning on pileup conditions*
- *Small/No performance loss with average number of PU collisions*

$\overline{n}_{\mathrm{PU}}$	20 (CHS)	80 (CHS)	140 (CHS)	80 (No CHS)
p_T	92.3%	92.3%	92.5%	64.9%
PUPPI weight	94.1%	93.9%	94.4%	65.1%
Fully-connected	95.0%	94.8%	94.8%	68.5%
GRU	94.8%	94.8%	94.7%	68.8%
GGNN	**96.1%**	**96.1%**	**96.0%**	**70.1%**



Pileup mitigation at the Large Hadron Collider with Graph Neural Networks

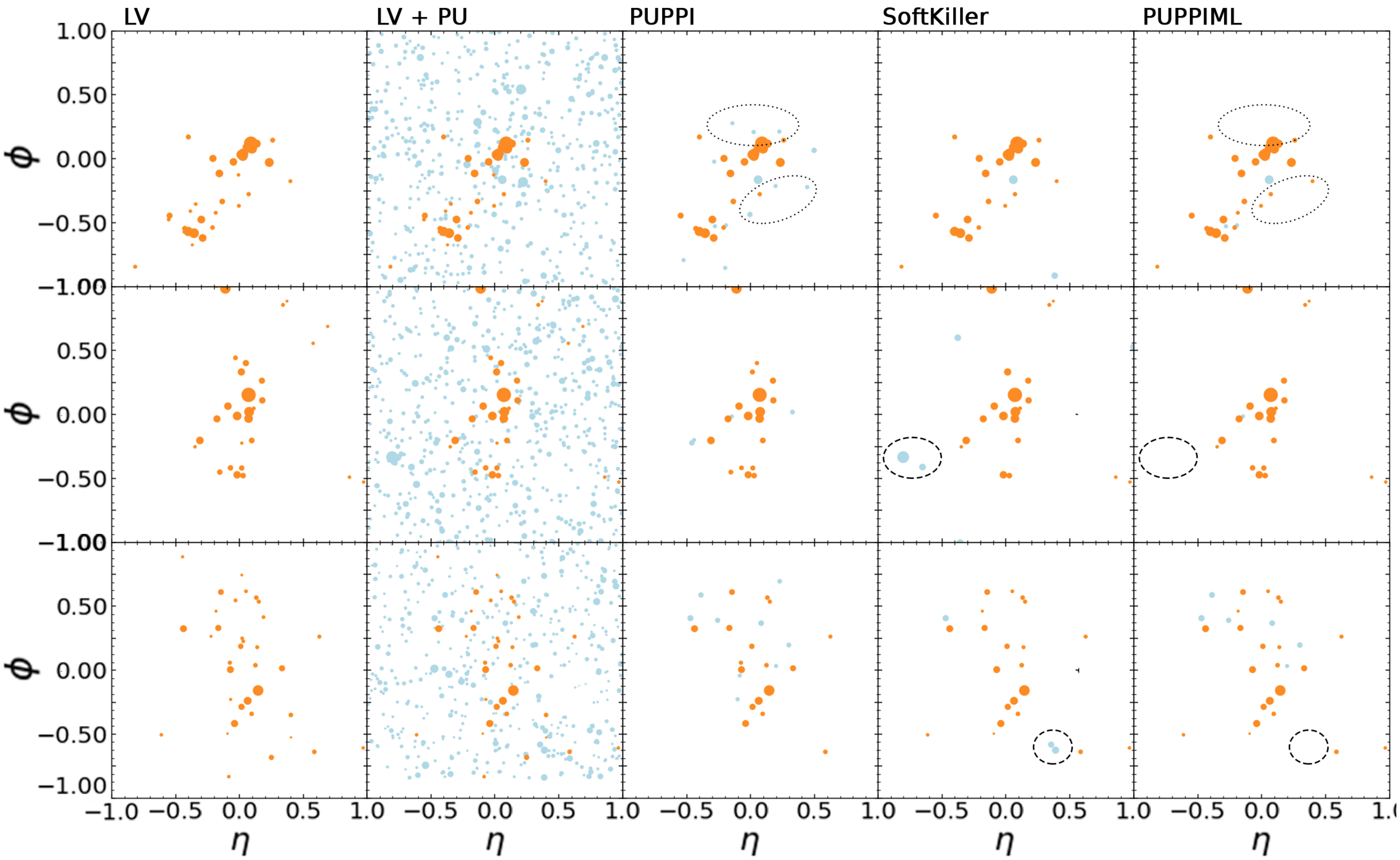
J. Arjona Martínez,[a,b,c] O. Cerri,[b] M. Pierini,[c] M. Spiropulu[b] and JR. Vlimant[b]

[a] *University of Cambridge, Trinity Ln, Cambridge CB2 1TN, UK*
[b] *California Institute of Technology, 1200 E. California Blvd, Pasadena, CA 91125*
[c] *CERN, CH-1211 Geneva, Switzerland*

E-mail: ja618@cam.ac.uk, olmo.cerri@cern.ch, maurizio.pierini@cern.ch, smaria@caltech.edu, jvlimant@caltech.edu

European Research Council

PUPPIML: Graph Nets for PU subtraction



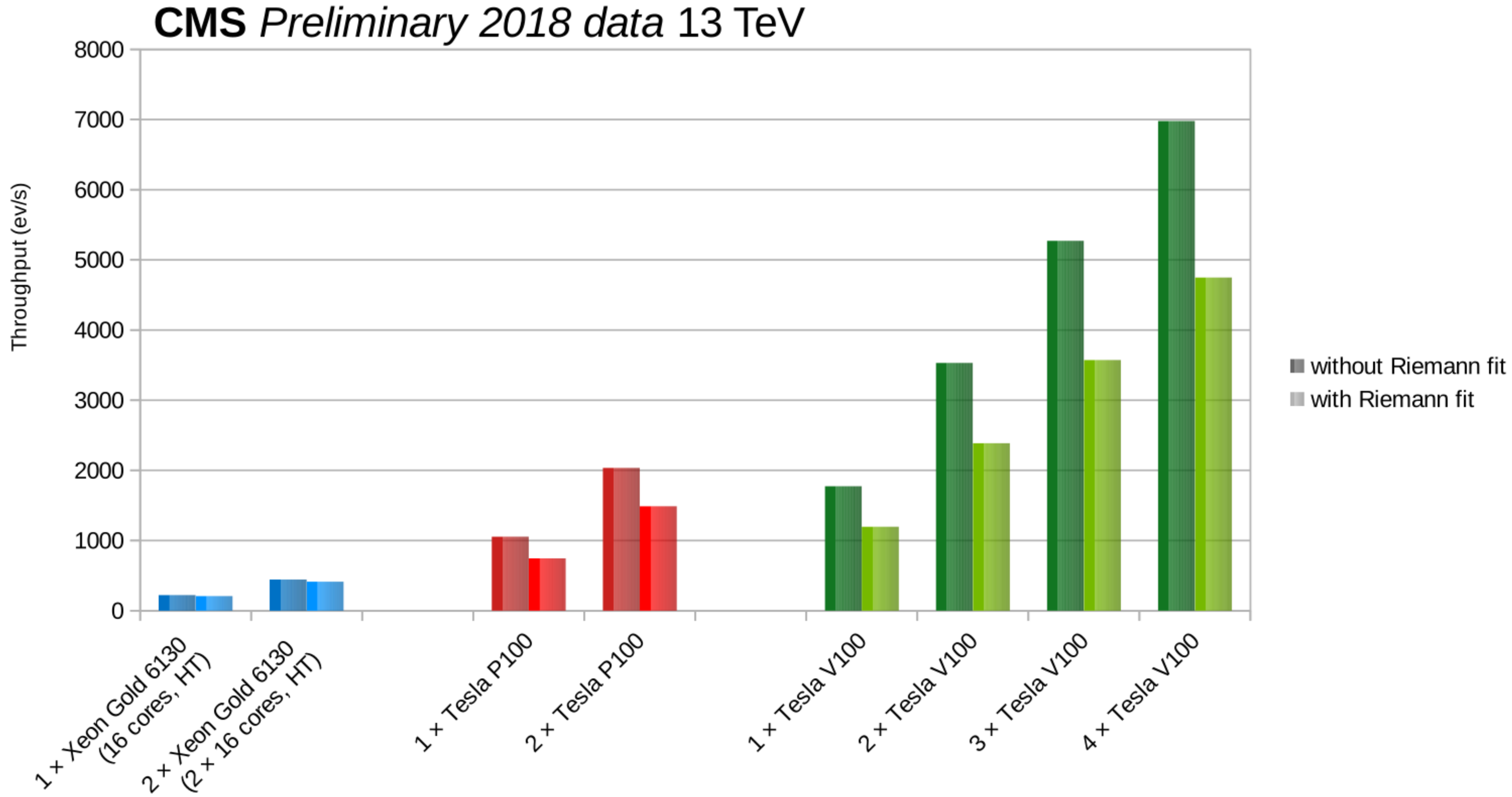
Porting Deep Learning to Trigger/DAQ system

European Research Council

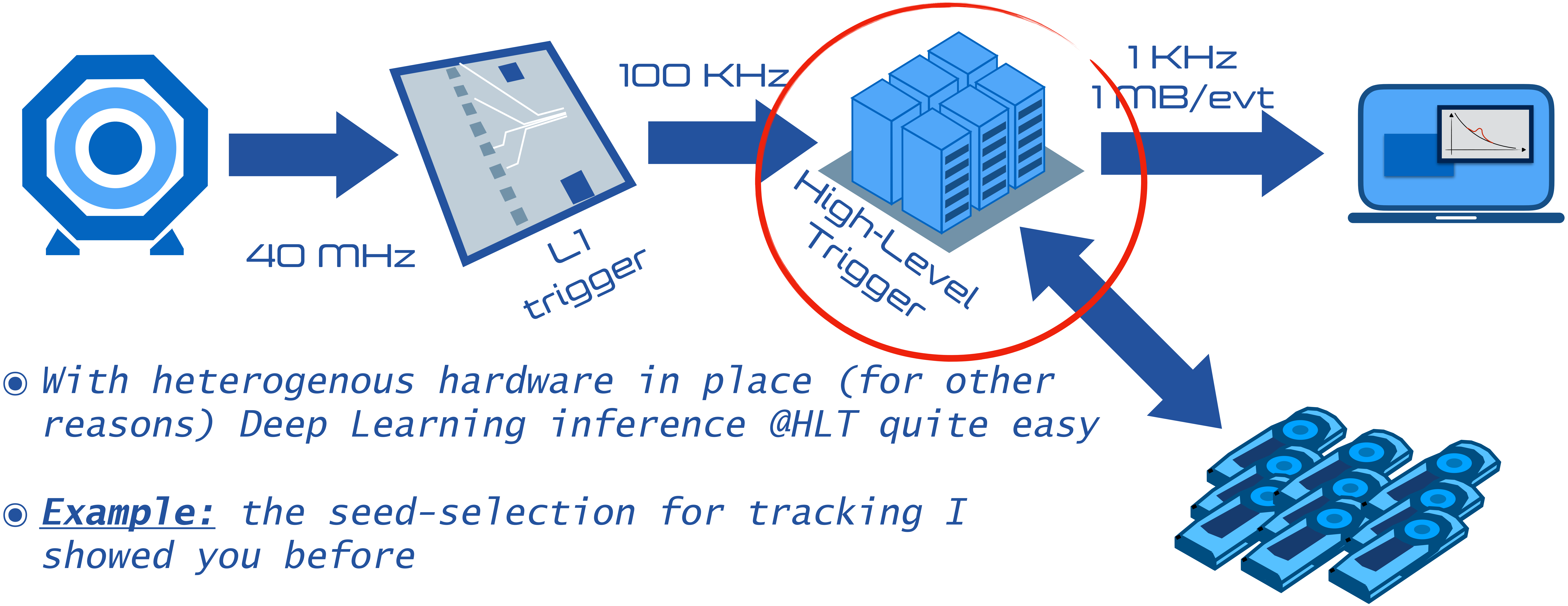
The next HLT

- Looking at current tendency, we expect the next trigger system to be based on heterogenous computing, with GPUs & FPGAs used as accelerators to compensate saturation of Moore's law
  - for tracking, clustering, etc
- In such a system, Deep Learning inference could be made very fast
  - On GPUs, as long as batching can be exploited
    - No big gain running one inference at once
    - Gain if many "samples" are sent at once. Example: 1K tracks per event
    - If objects are made on GPUs, no need to move them back and forth
  - On FPGAs, without need of batching, as long as the model can fit the available resources (including resource recycle with fast access to memory)

Patatrack project for CMS HLT on GPUs



Heterogeneous HLT



- *With heterogenous hardware in place (for other reasons) Deep Learning inference @HLT quite easy*

- ***Example:*** *the seed-selection for tracking I showed you before*

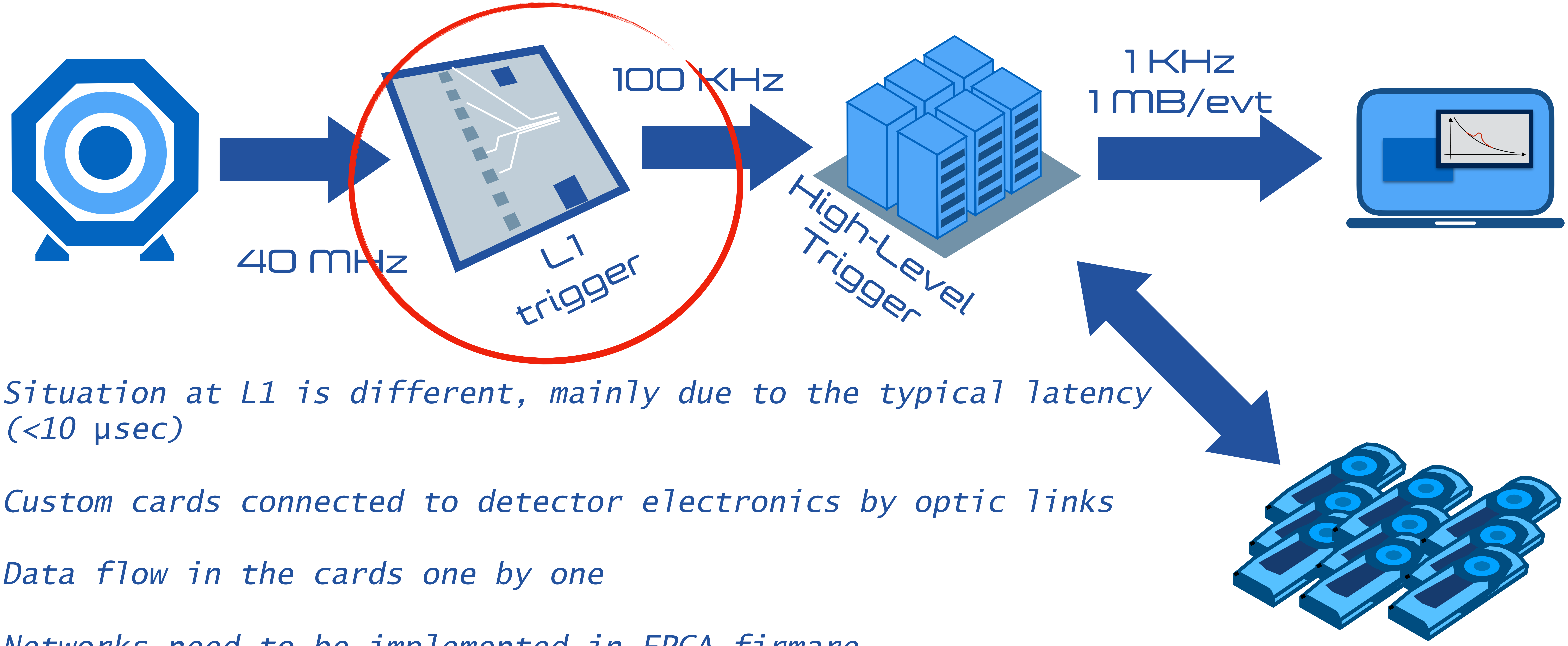
  - *1 µsec to know if a seed is good or not*

  - *1M seeds/event -> 1sec to process an event serially*

Deep Learning at L1

- *Situation at L1 is different, mainly due to the typical latency (<10 μsec)*
- *Custom cards connected to detector electronics by optic links*
- *Data flow in the cards one by one*
- *Networks need to be implemented in FPGA firmware*
  - *advanced design by expert engineers (not common resource in HEP)*
  - *automatic translation tools doing the job*

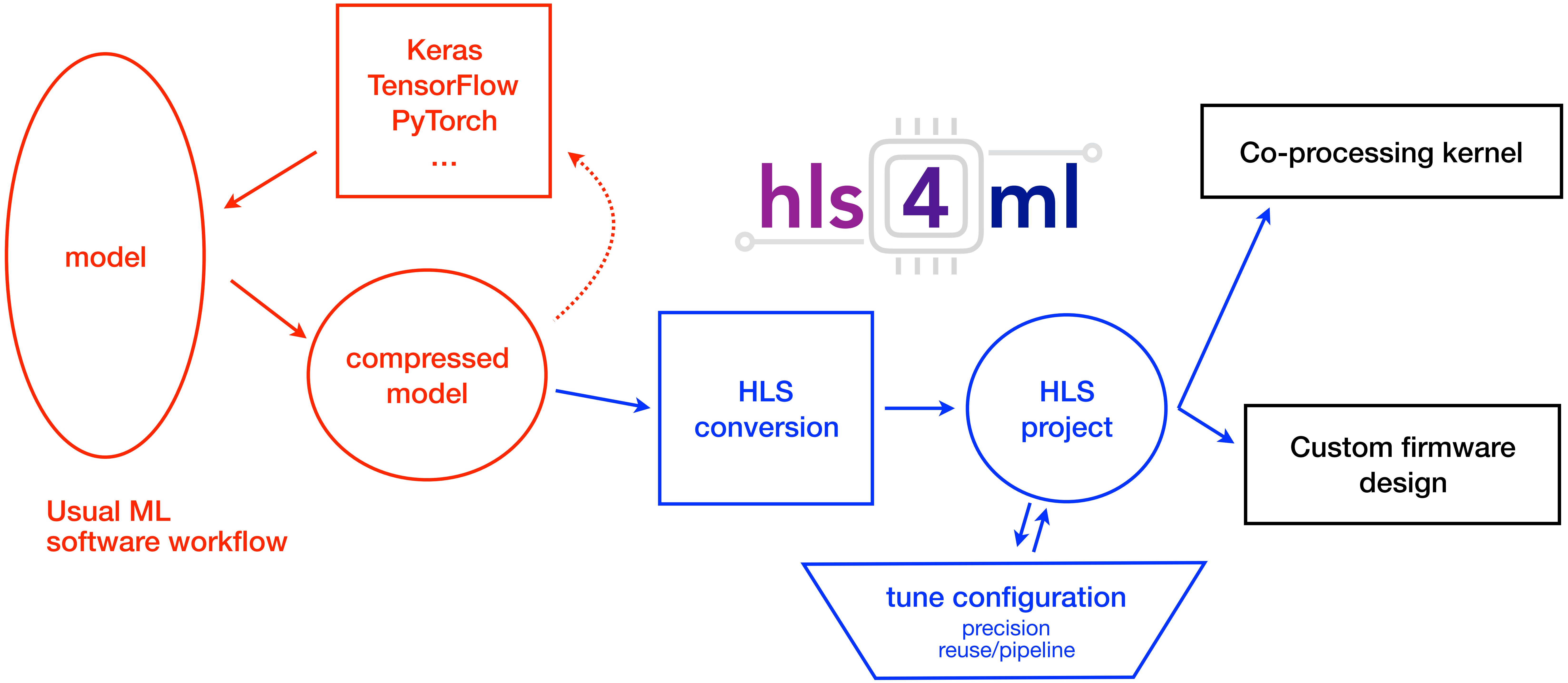
Deep Learning at L1

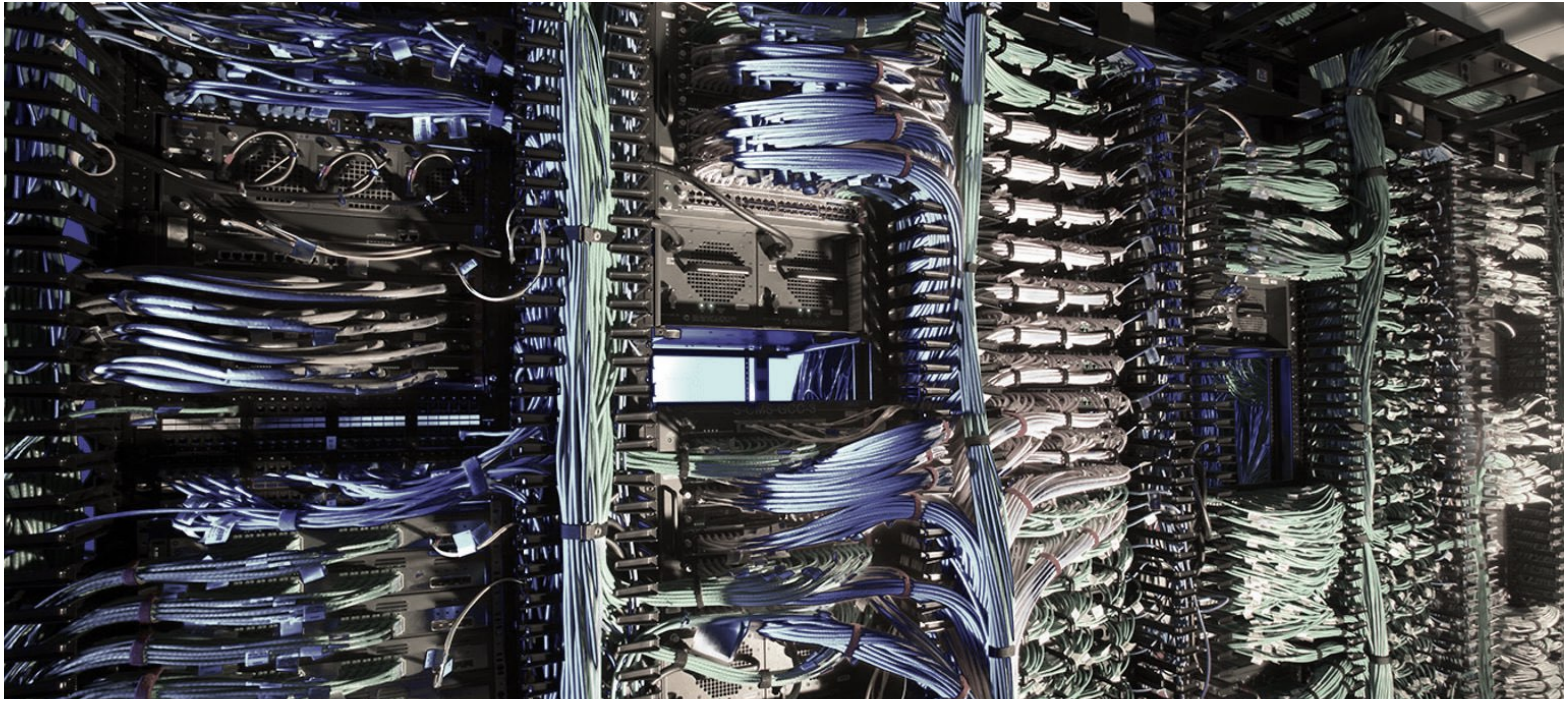


- Situation at L1 is different, mainly due to the typical latency (<10 μsec)
- Custom cards connected to detector electronics by optic links
- Data flow in the cards one by one
- Networks need to be implemented in FPGA firmare
  - advanced design by expert engineers (not common resource in HEP)
  - automatic translation tools doing the job

HLS4ML

- *HLS4ML aims to be this automatic tool*
  - *reads as input models trained on standard DeepLearning libraries*
  - *comes with implementation of common ingredients (layers, activation functions, etc)*
  - *Uses HLS softwares to provide a firmware implementation of a given network*
  - *Could also be used to create co-processing kernels for HLT environments*

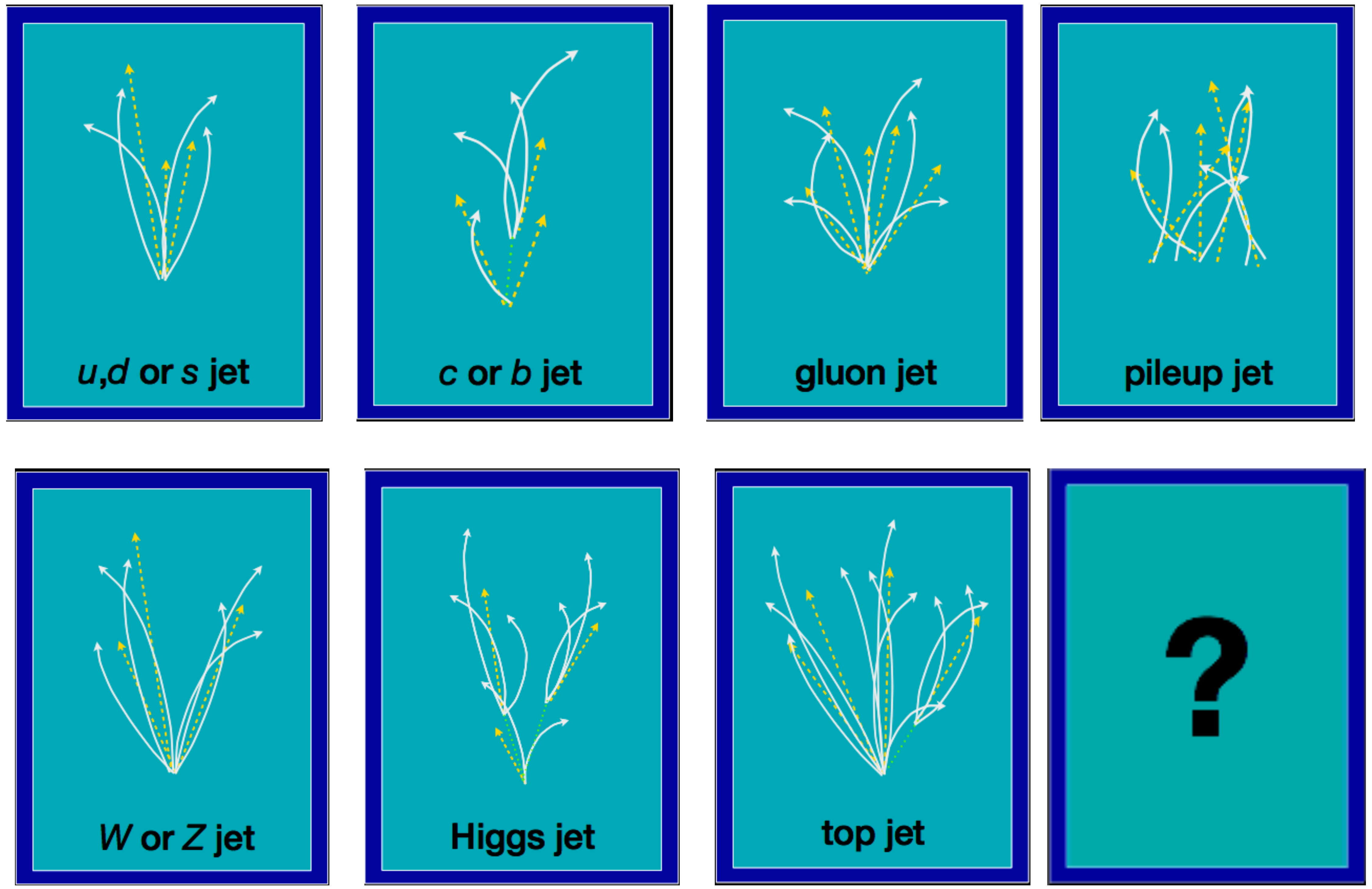




Fast Decision Taking

Example: fast inference

- You have a jet at LHC: spray of hadrons coming from a “shower” initiated by a fundamental particle of some kind (quark, gluon, W/Z/H bosons, top quark)
- You have a set of jet features whose distribution depends on the nature of the initial particle
- You can train a network to start from the values of these quantities and guess the nature of your jet
- To do this you need a sample for which you know the answer



Example: jet tagging

- *Simple DNN based on high-level features (jet masses, multiplicities, energy correlation functions)*

Observables

m_{mMDT}

$N_2^{\beta=1,2}$

$M_2^{\beta=1,2}$

$C_1^{\beta=0,1,2}$

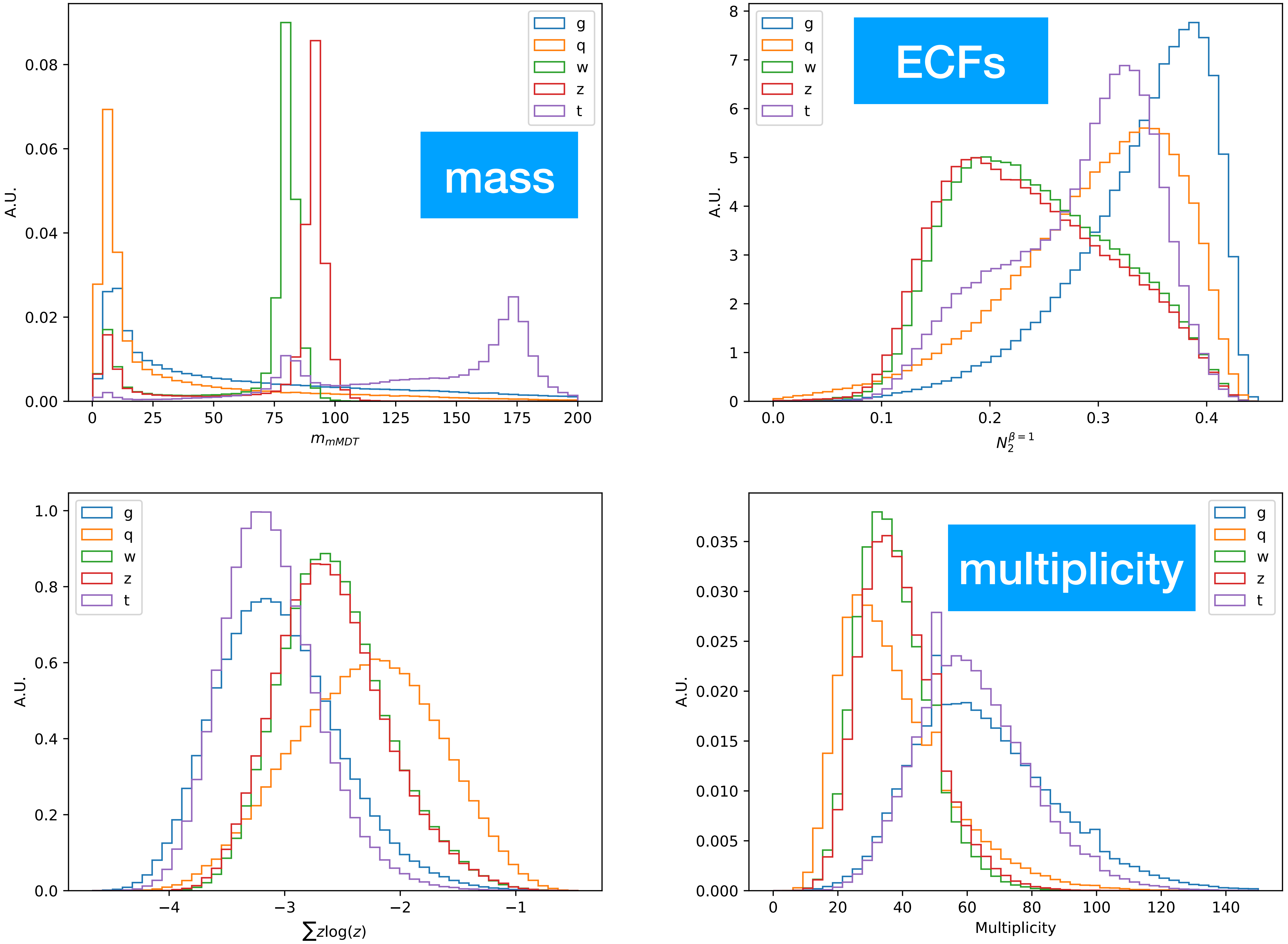
$C_2^{\beta=1,2}$

$D_2^{\beta=1,2}$

$D_2^{(\alpha,\beta)=(1,1),(1,2)}$

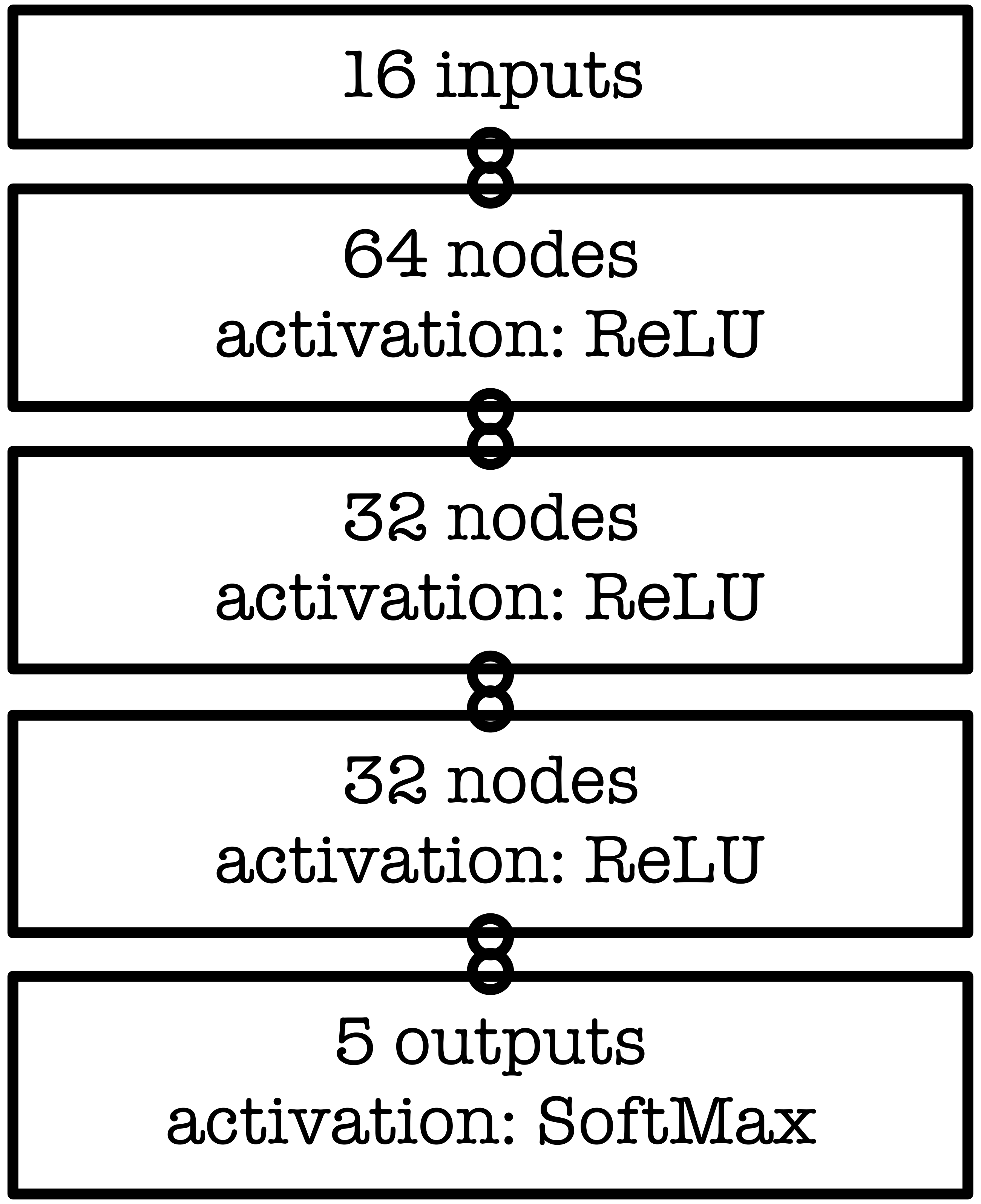
$\sum z \log z$

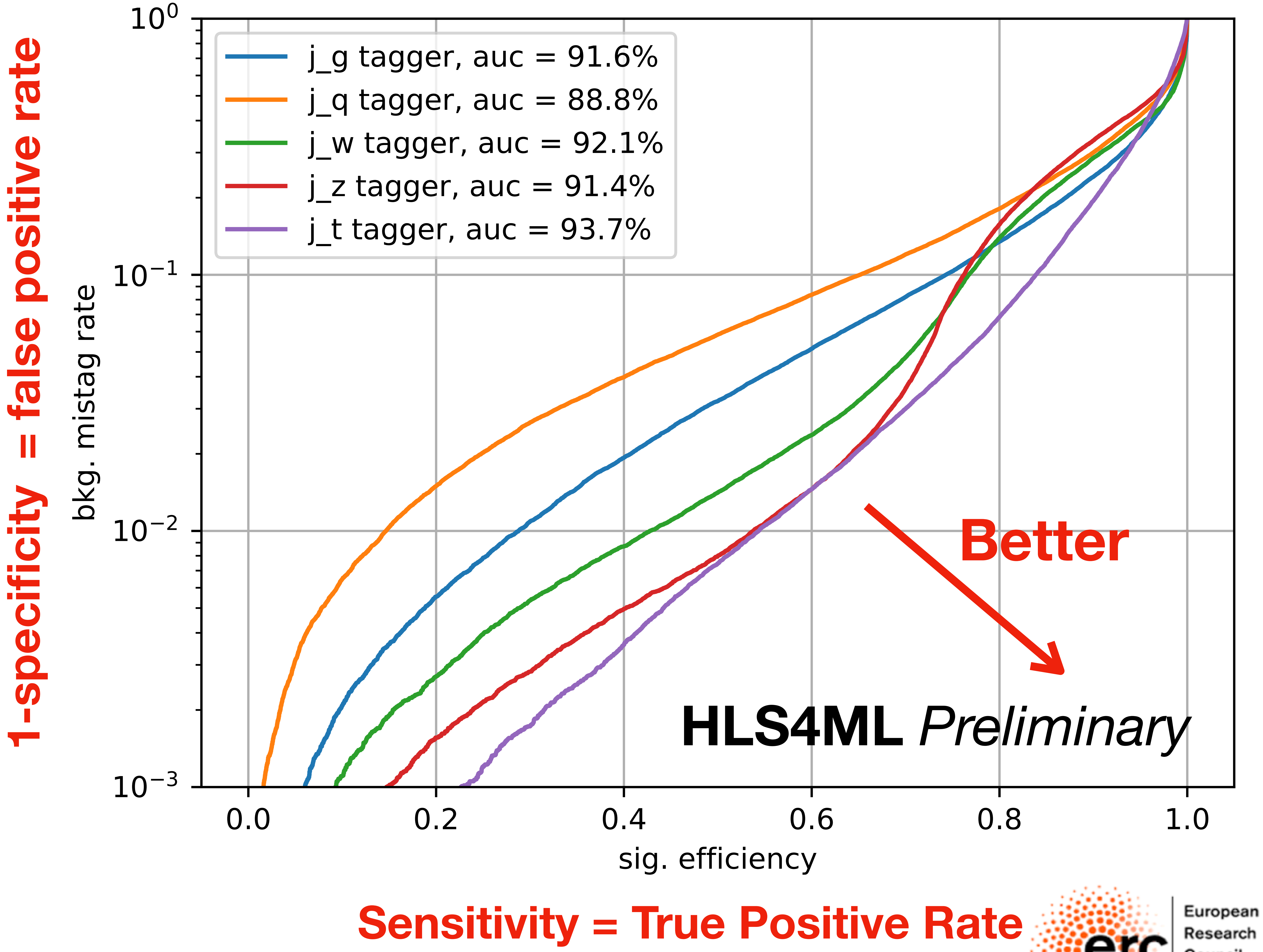
Multiplicity



Example: jet tagging

- *Simple DNN based on high-level features (jet masses, multiplicities, energy correlation functions)*

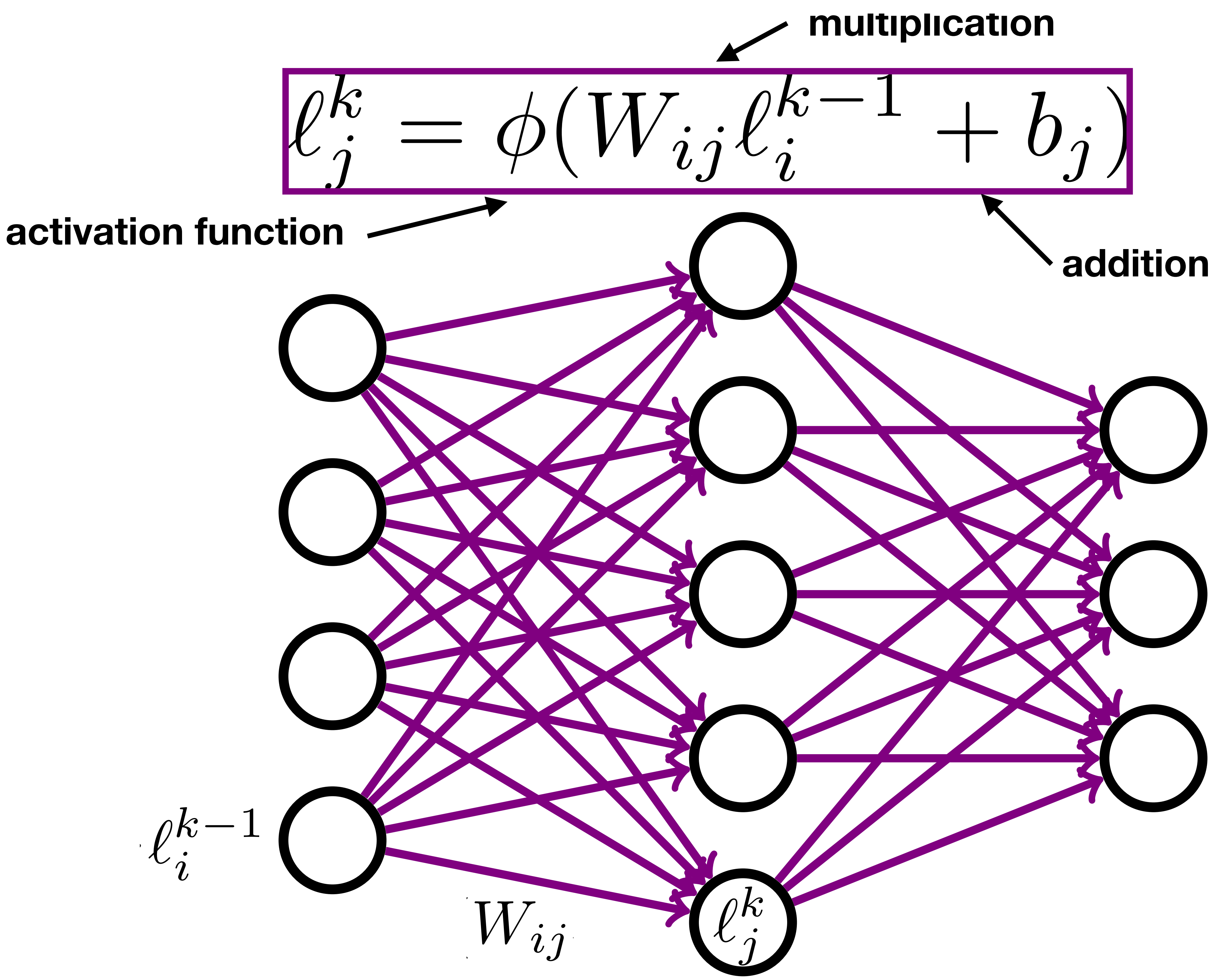


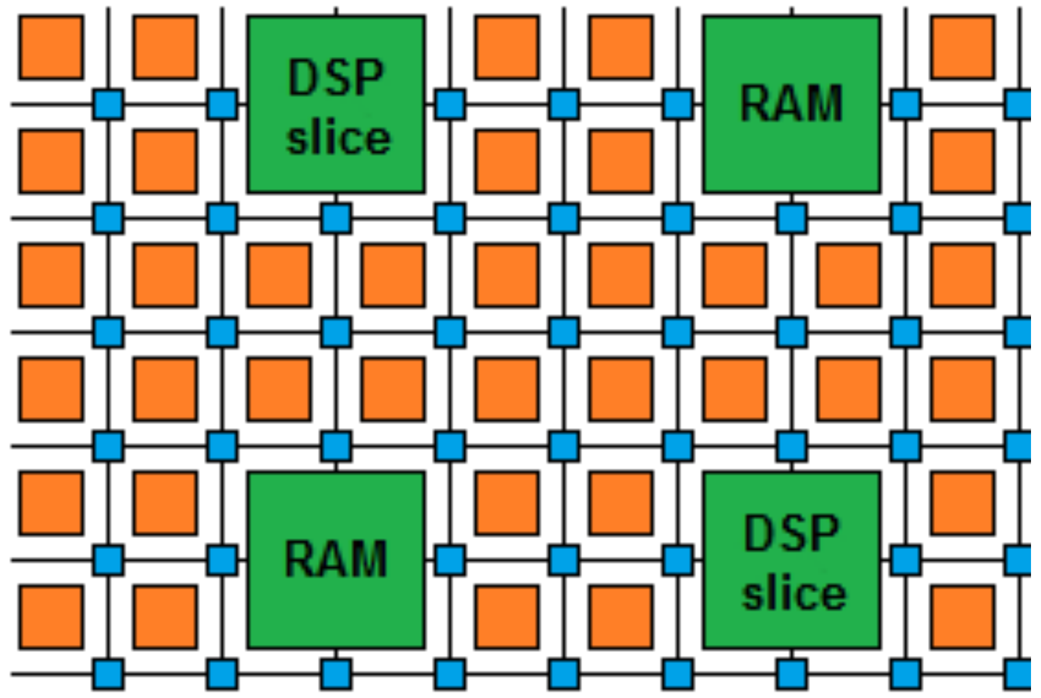


European Research Council

Network Operations

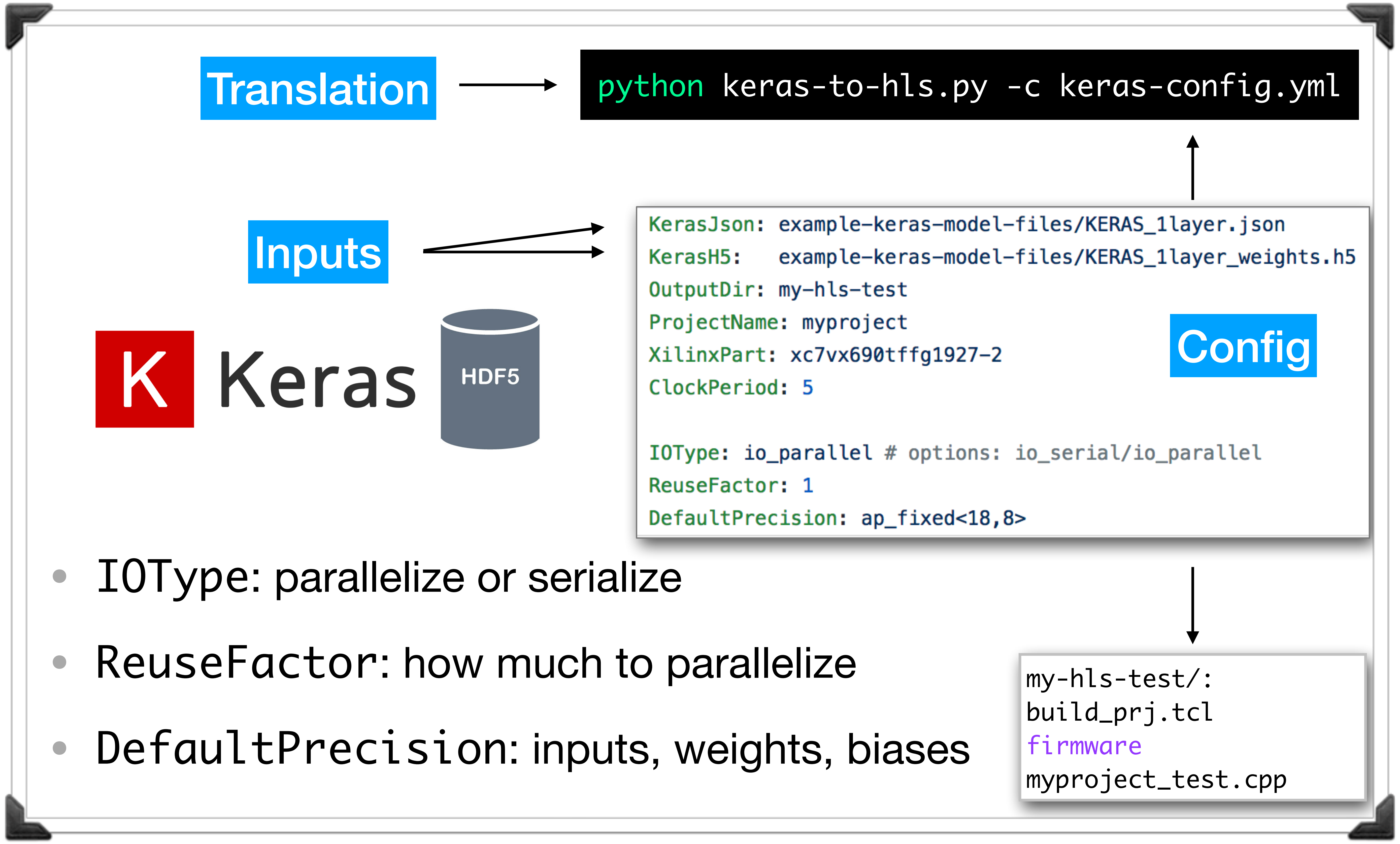
- A classic Dense NN manipulates the inputs in three ways
    - multiplying by weights
    - adding biases
    - applying activation functions
- All these operations map nicely into an FPGA
    - high IO, DSPs, LUTs, tunable precision



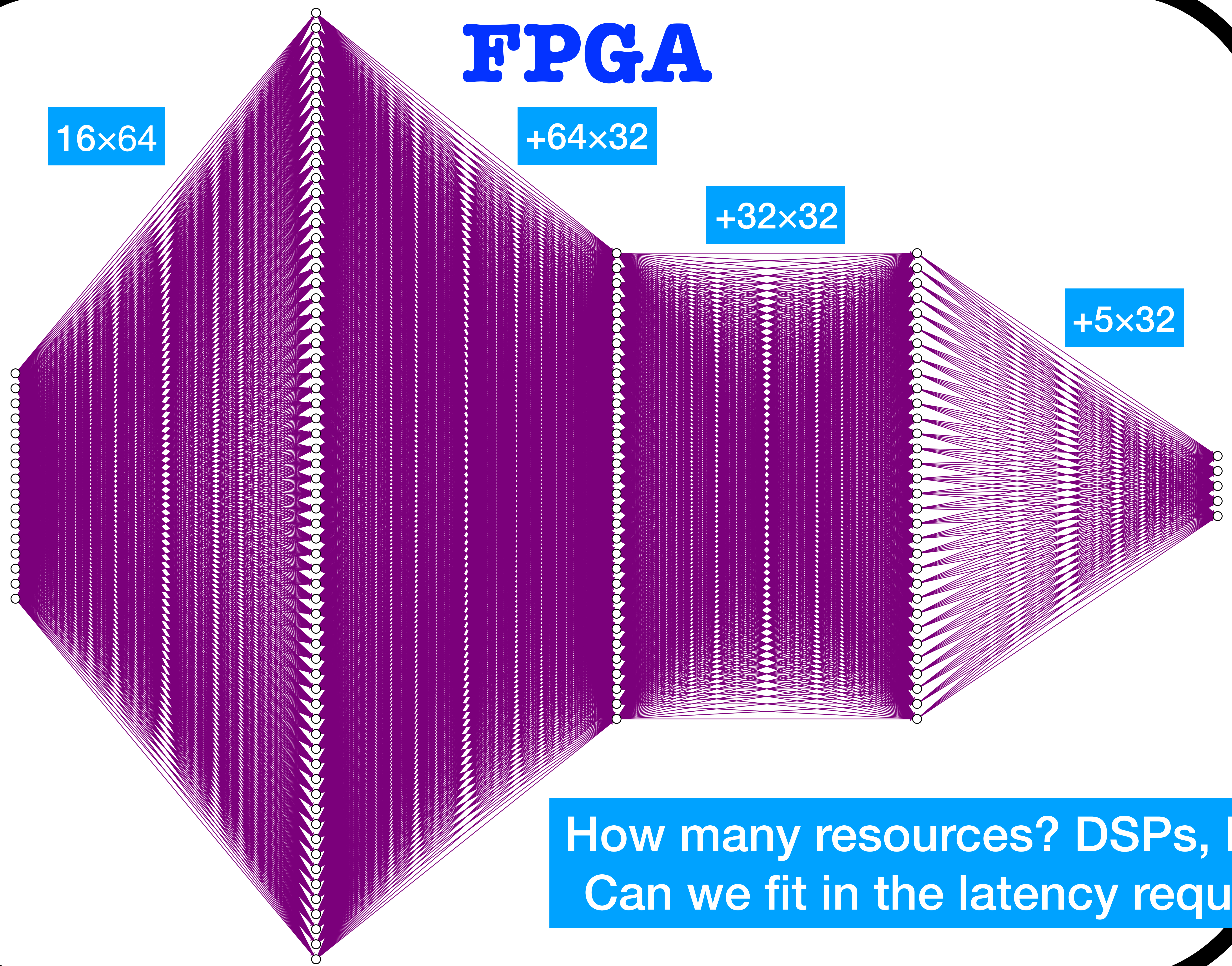


Bring the model to FPGA

- How this works in practice
  - A python based library that takes inputs via a yam file
  - Model architecture with supported format
  - FPGA configuration parameters (reuse factor, FPGA model, Clock period, etc)
- The library provides inputs for Vivado HLS



The full model



= 4,256 synapses / mult.

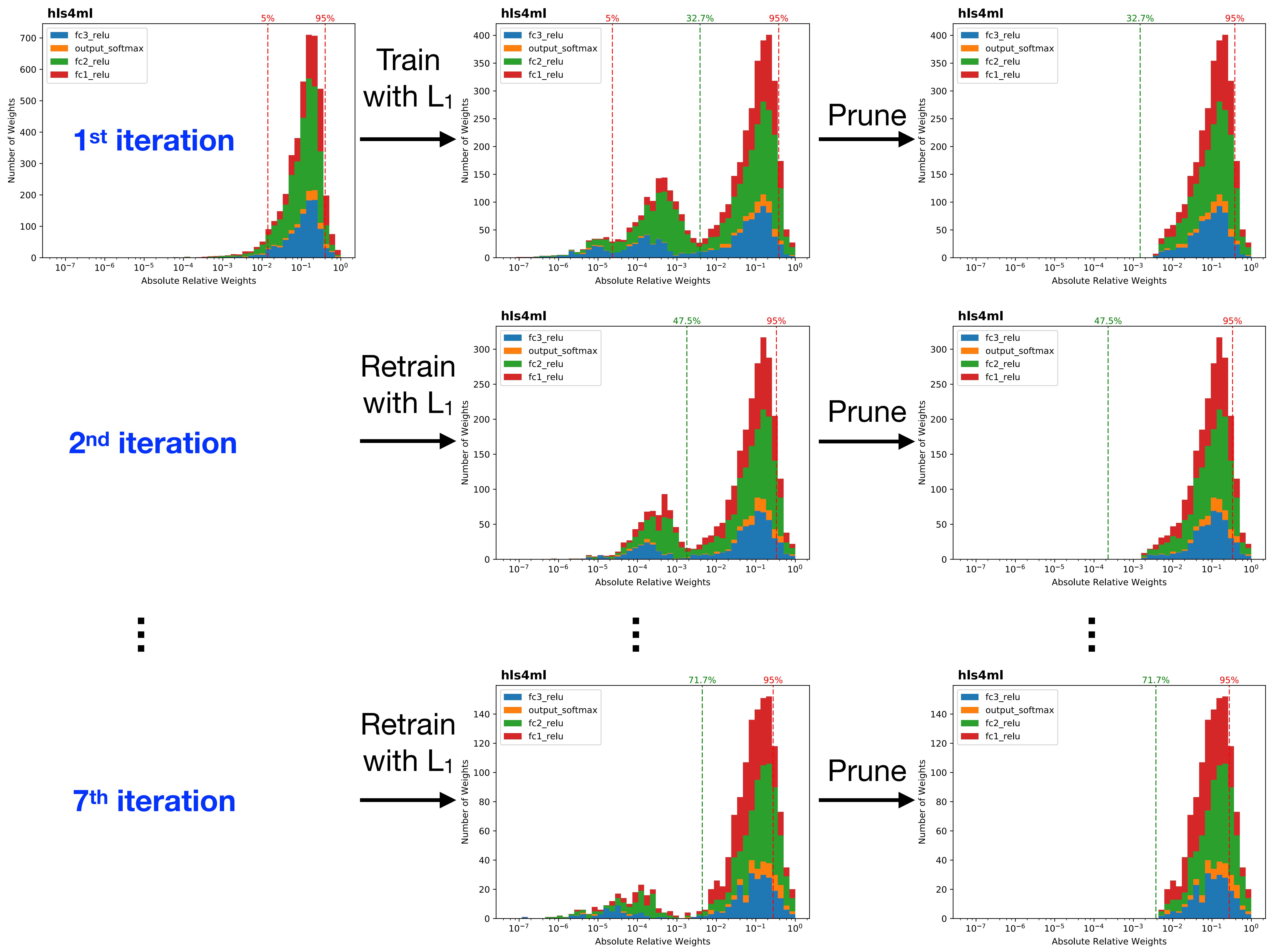
How many resources? DSPs, LUTs, FFs?
Can we fit in the latency requirements?

Compression

- **Pruning:** *remove parameters that don't really contribute to performances*
  - *force parameters to be as small as possible (regularization)*

$$L_\lambda(\vec{w}) = L(\vec{w}) + \lambda||\vec{w}_1||$$

  - *Remove the small parameters*
  - *Retrain*

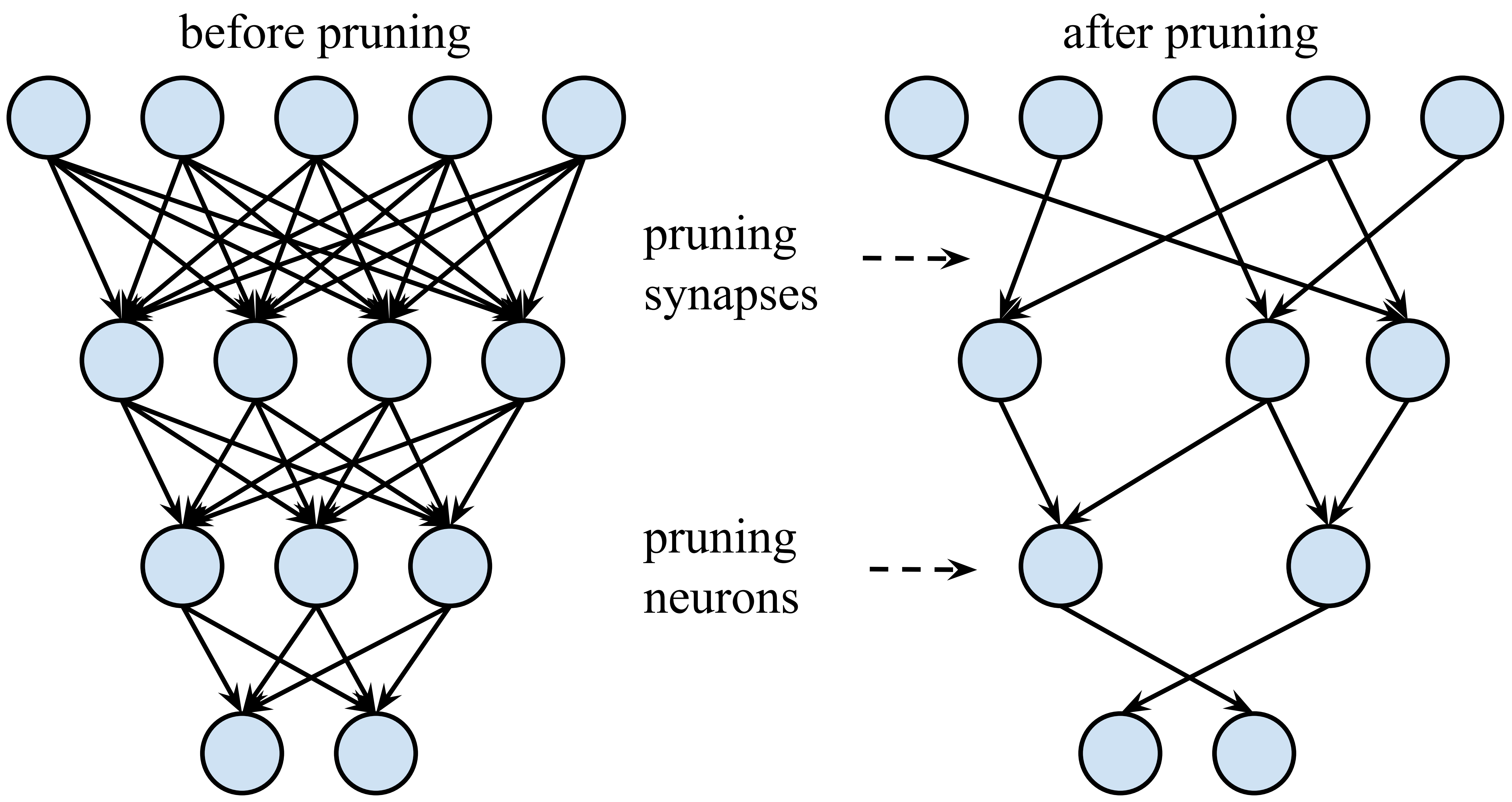


Compression

- ***Pruning:*** *remove parameters that don't really contribute to performances*
  - *force parameters to be as small as possible (regularization)*

$$L_\lambda(\vec{w}) = L(\vec{w}) + \lambda||\vec{w}_1||$$

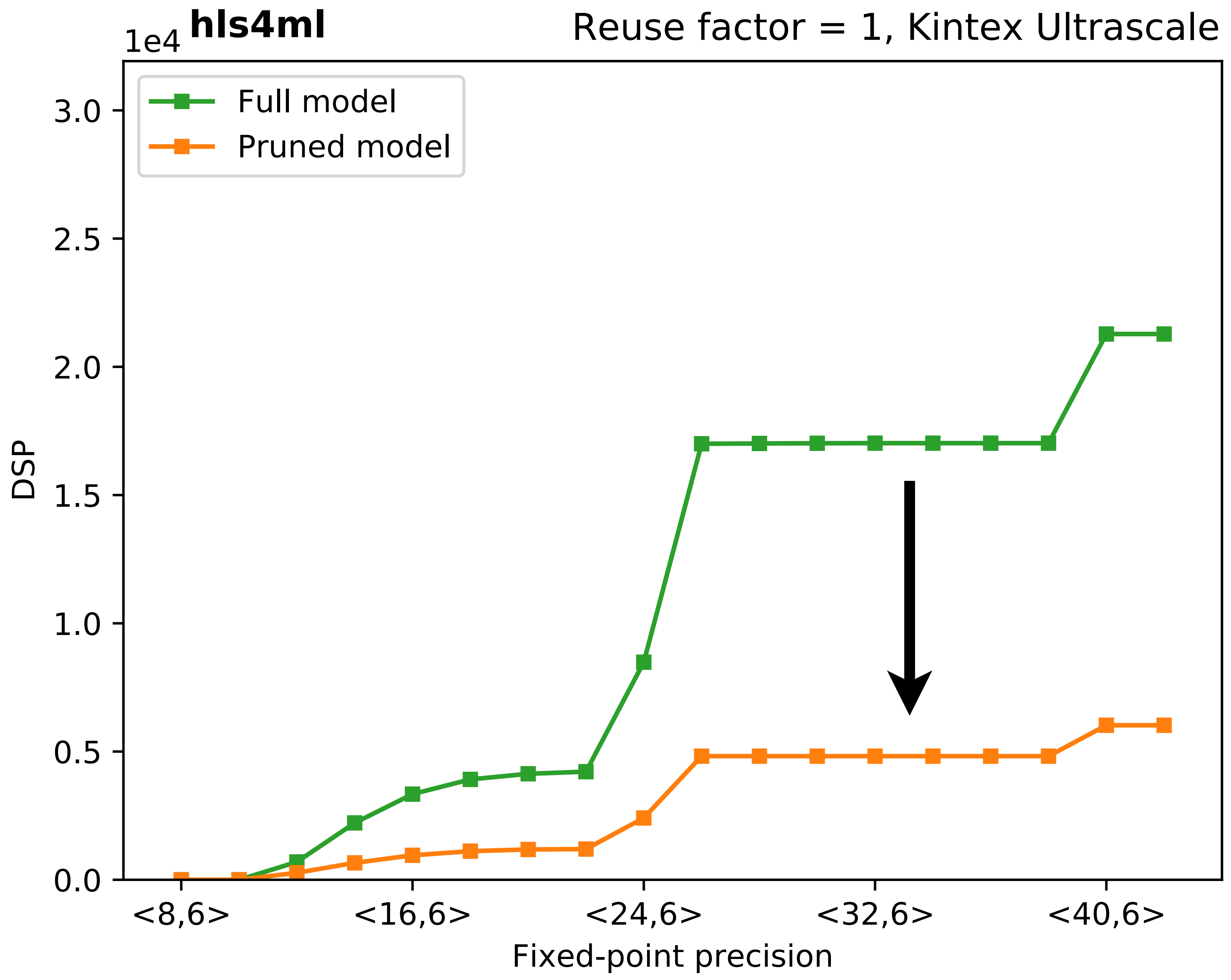
  - *Remove the small parameters*
  - *Retrain*

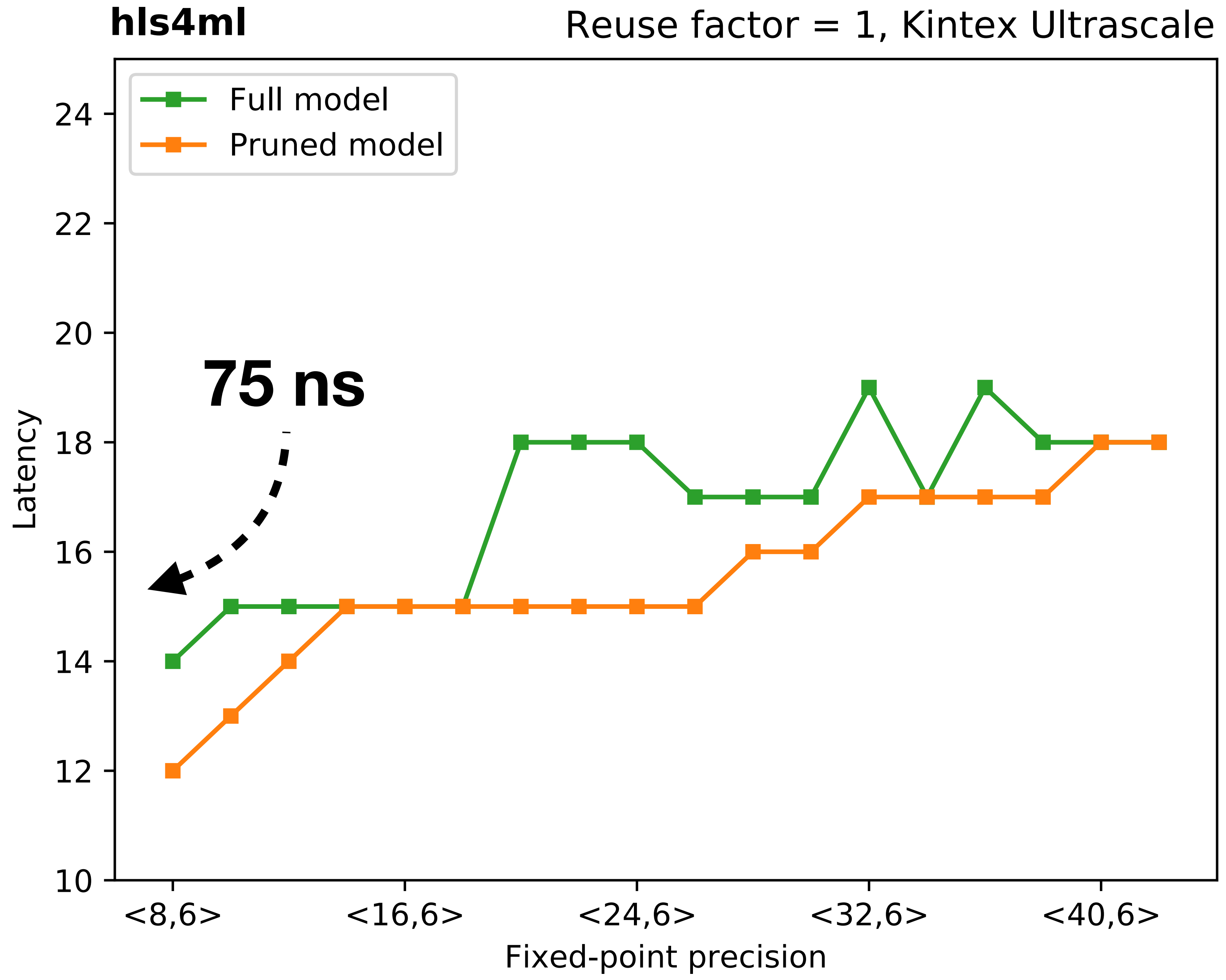


→ 70% reduction of weights and multiplications w/o performance loss

Compression

Xilinx Vivado 2017.2
Clock frequency: 200 MHz
FPGA: Xilinx Kintex Ultrascale
(XCKU115-FLVB2104)

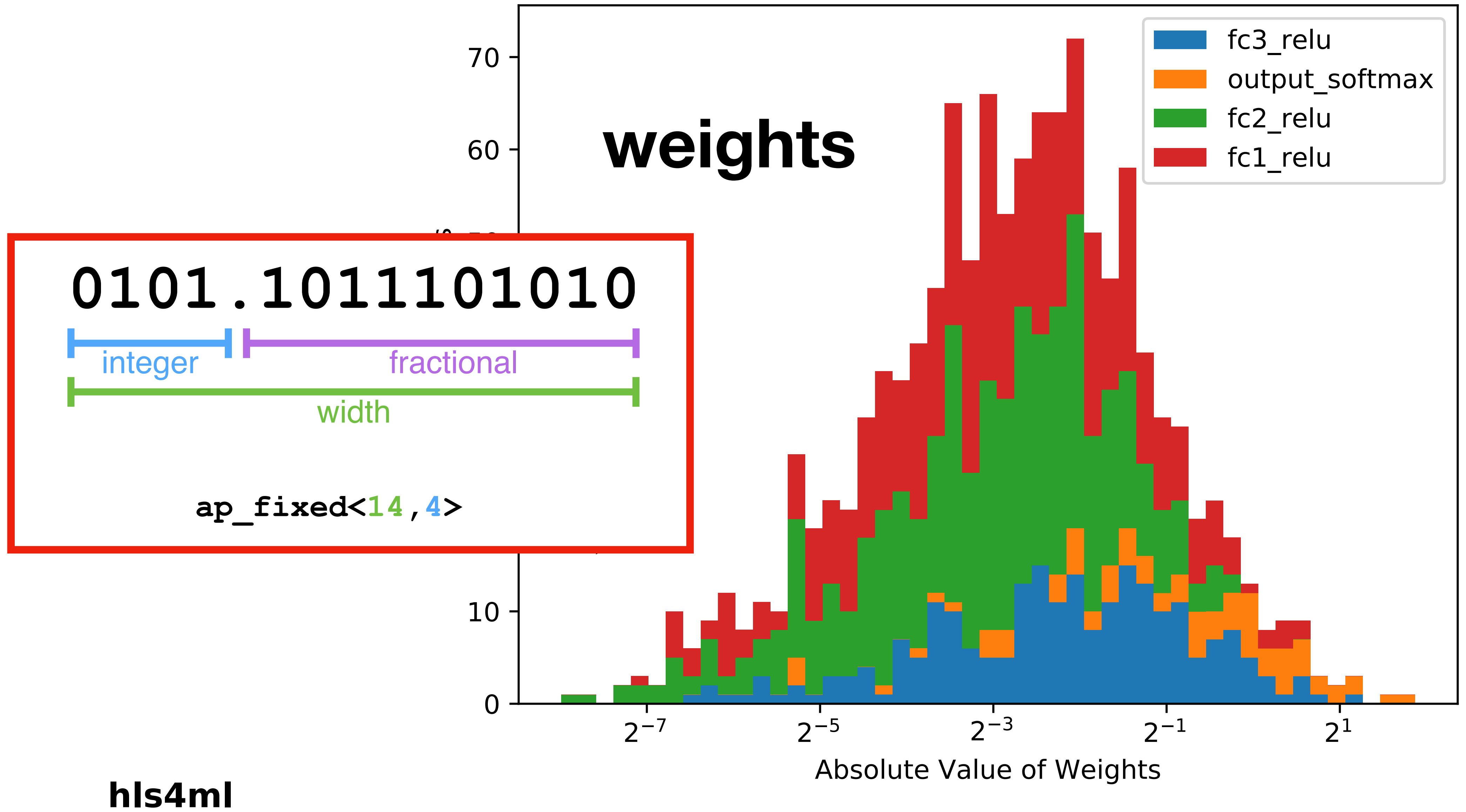


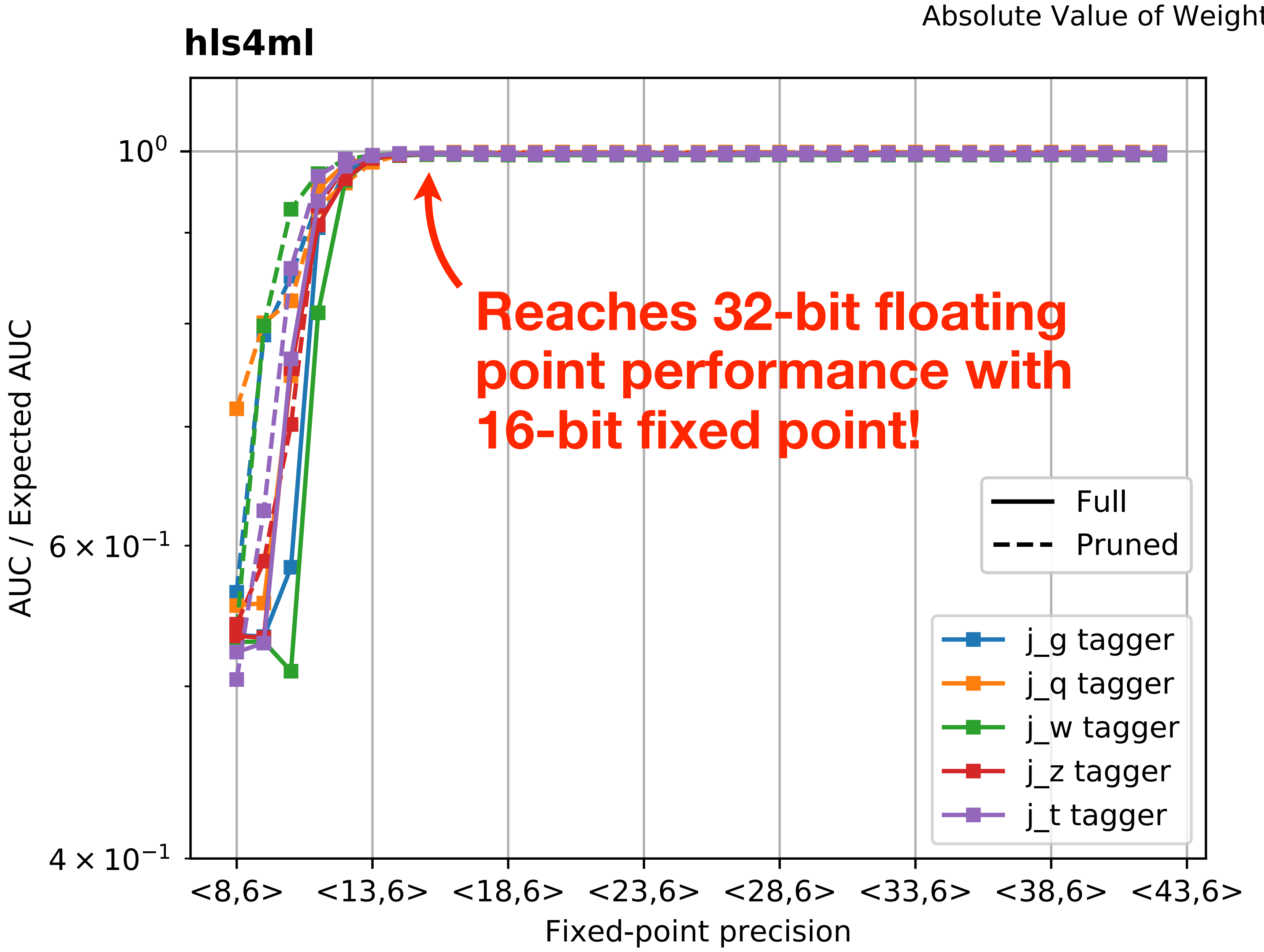


- Big reduction in DSP usage with pruned model!
- ~15 clocks @ 200 MHz = 75 ns inference

Quantisation

- ***Quantisation**: reduce the number of bits used to represent numbers (i.e., reduce used memory)*
    - *models are usually trained at 64 or 32 bits*
    - *this is not necessarily needed in real life*
- *In our case, we could reduce to 16 bits w/o loosing precision*
- *Beyond that, one would have to accept some performance loss*



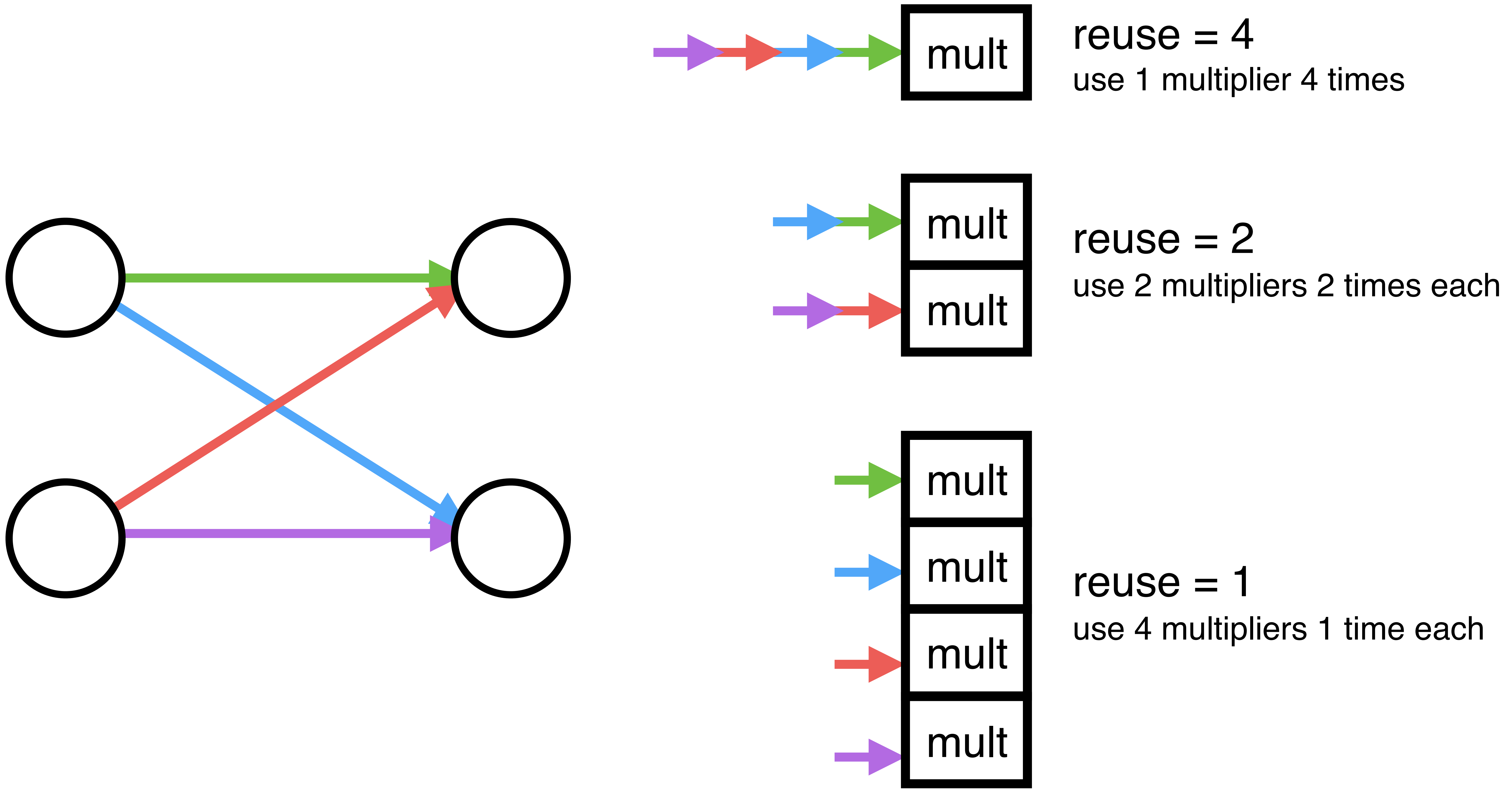


< total bits, integer bits >

European Research Council

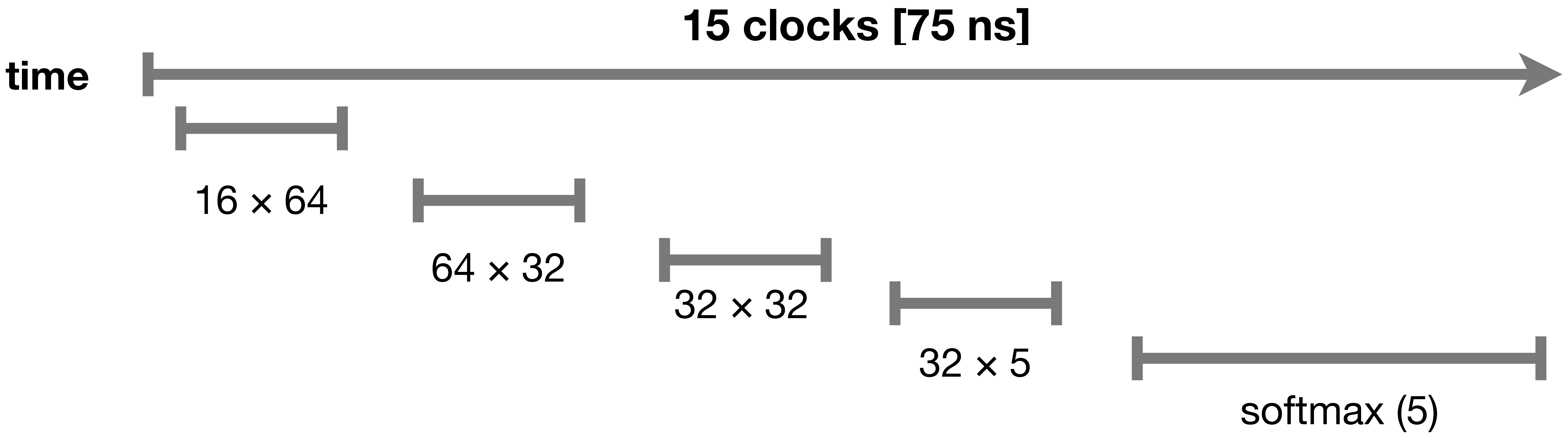
Parallelisation

- `ReuseFactor`: how much to parallelize



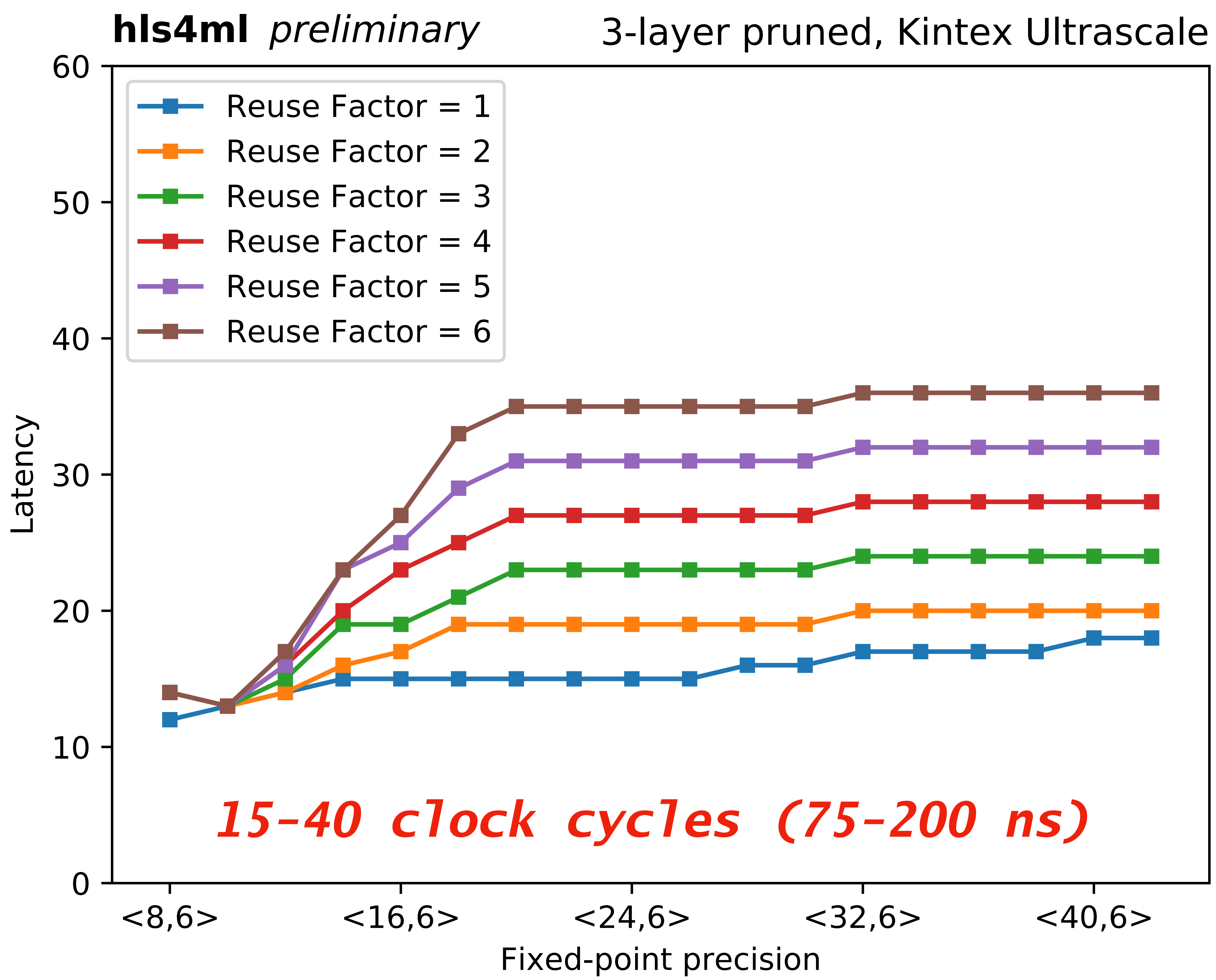
related to the Initiation Interval = when new inputs are introduced to the algo.

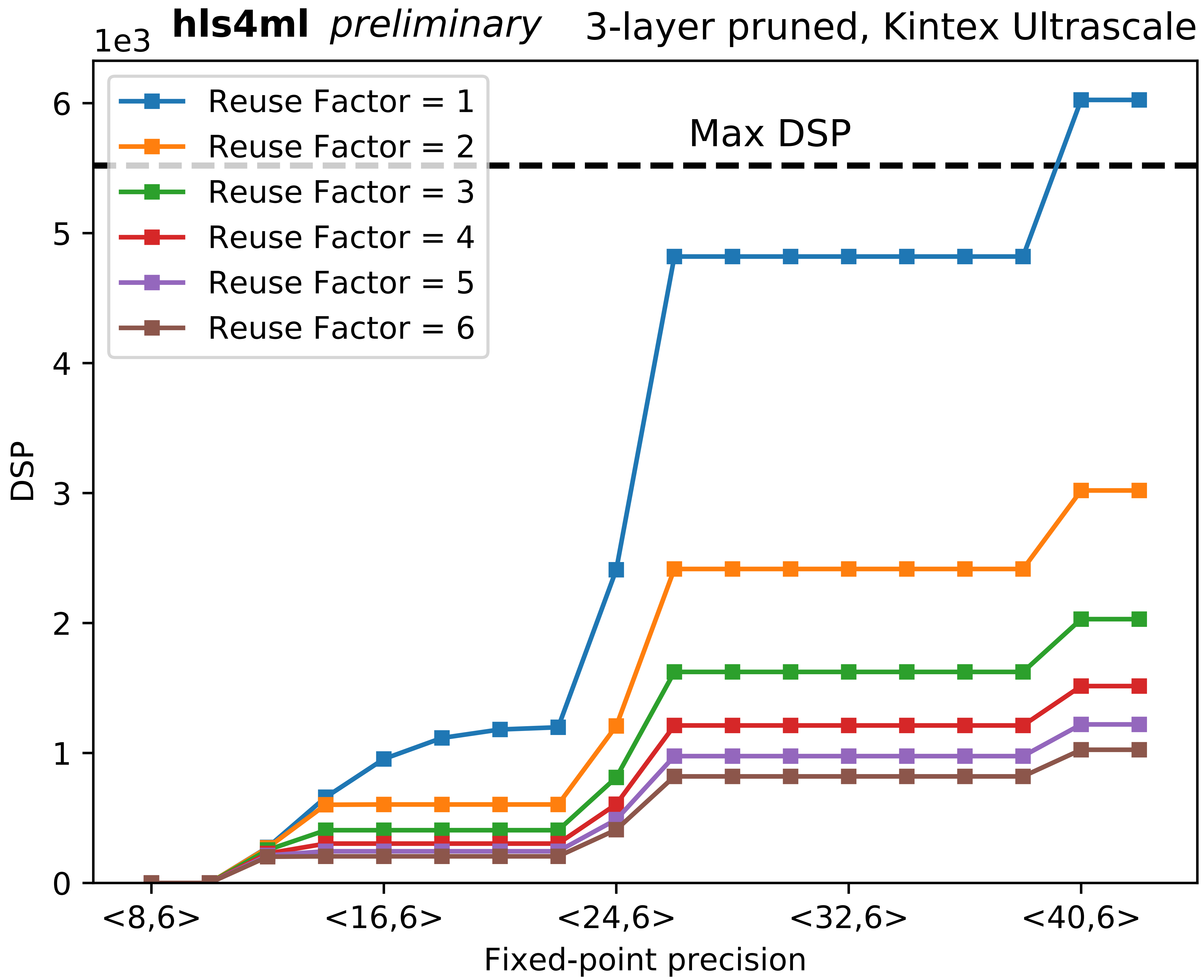
Parallelisation



reuse = 1 <16, 6> bits	BRAM	DSP	FF	LUT
Total	13	954	53k	36k
% Usage	~0%	17%	3%	5%

Parallelisation

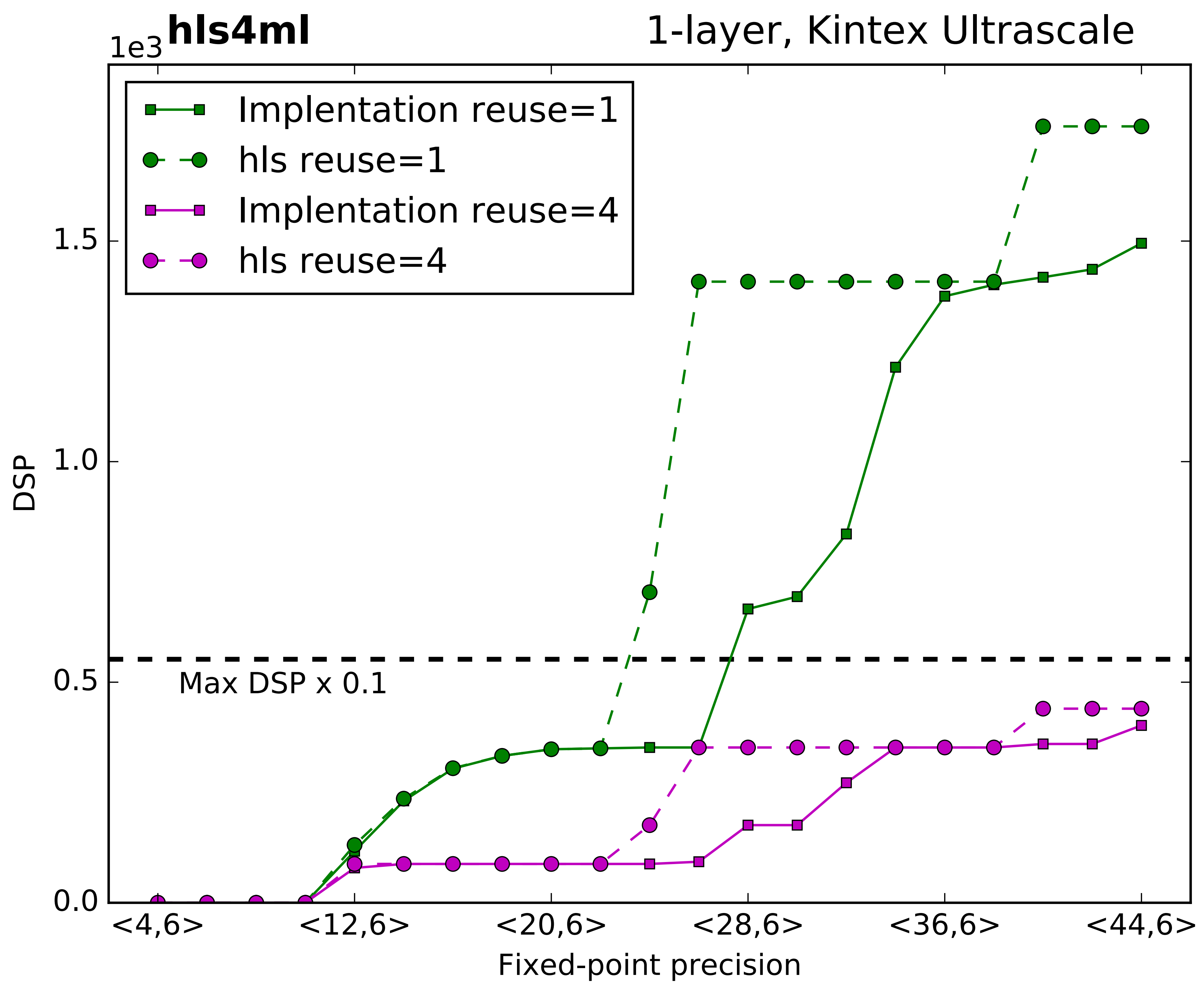


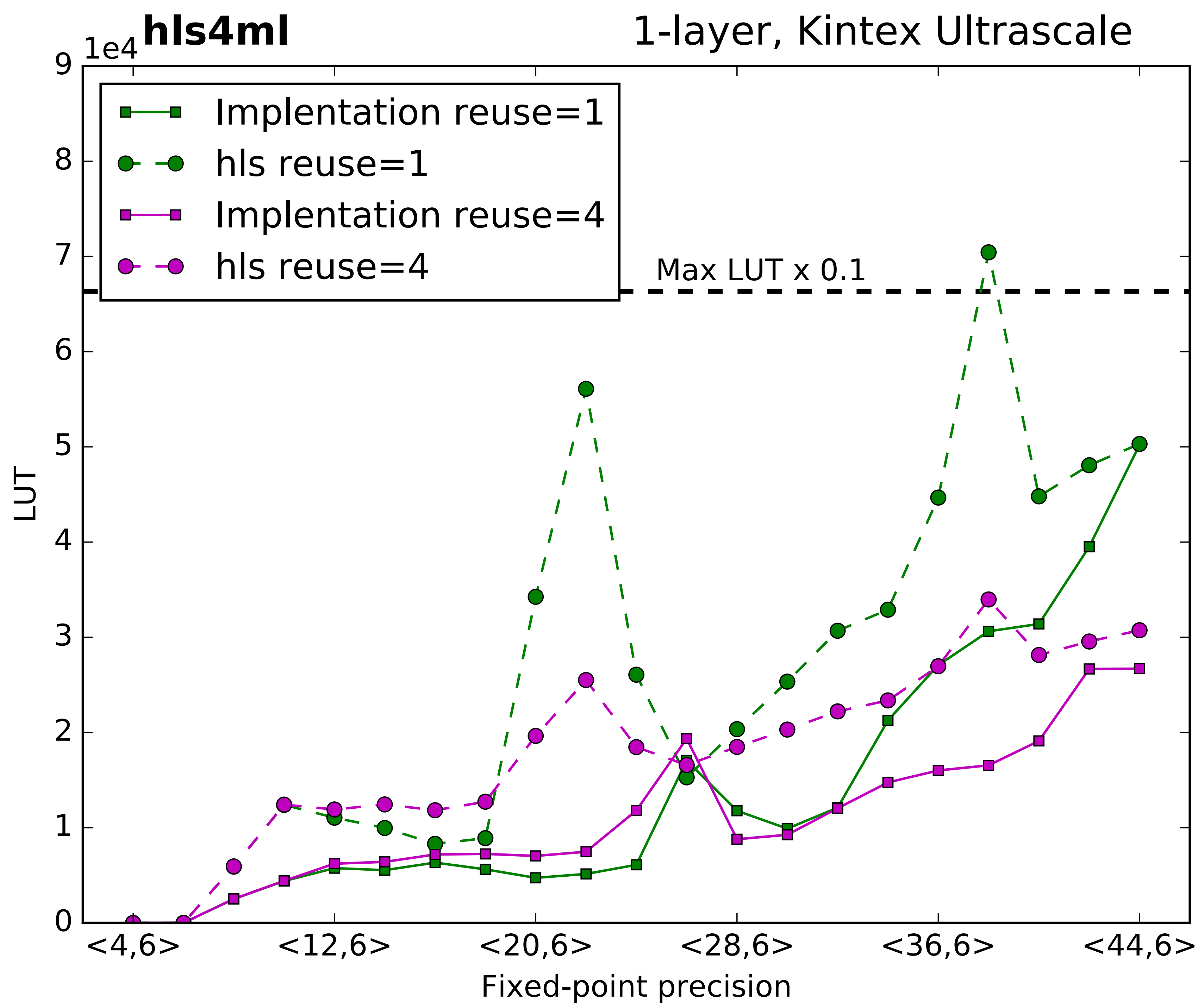


Foreseen architecture (FPGAs) will handle these networks
Inference-optimized GPUs could break the current paradigm
Looking forward to R&D projects with nVidia & E4 on this

Implementing HLS Design

- HLS gives us a **conservative** estimate of the resources needed
- It actually seems to give estimates close to the back-of-the-envelope optimal estimate
- Real life much more “smooth” than emulation: no spikes observed for LUTs



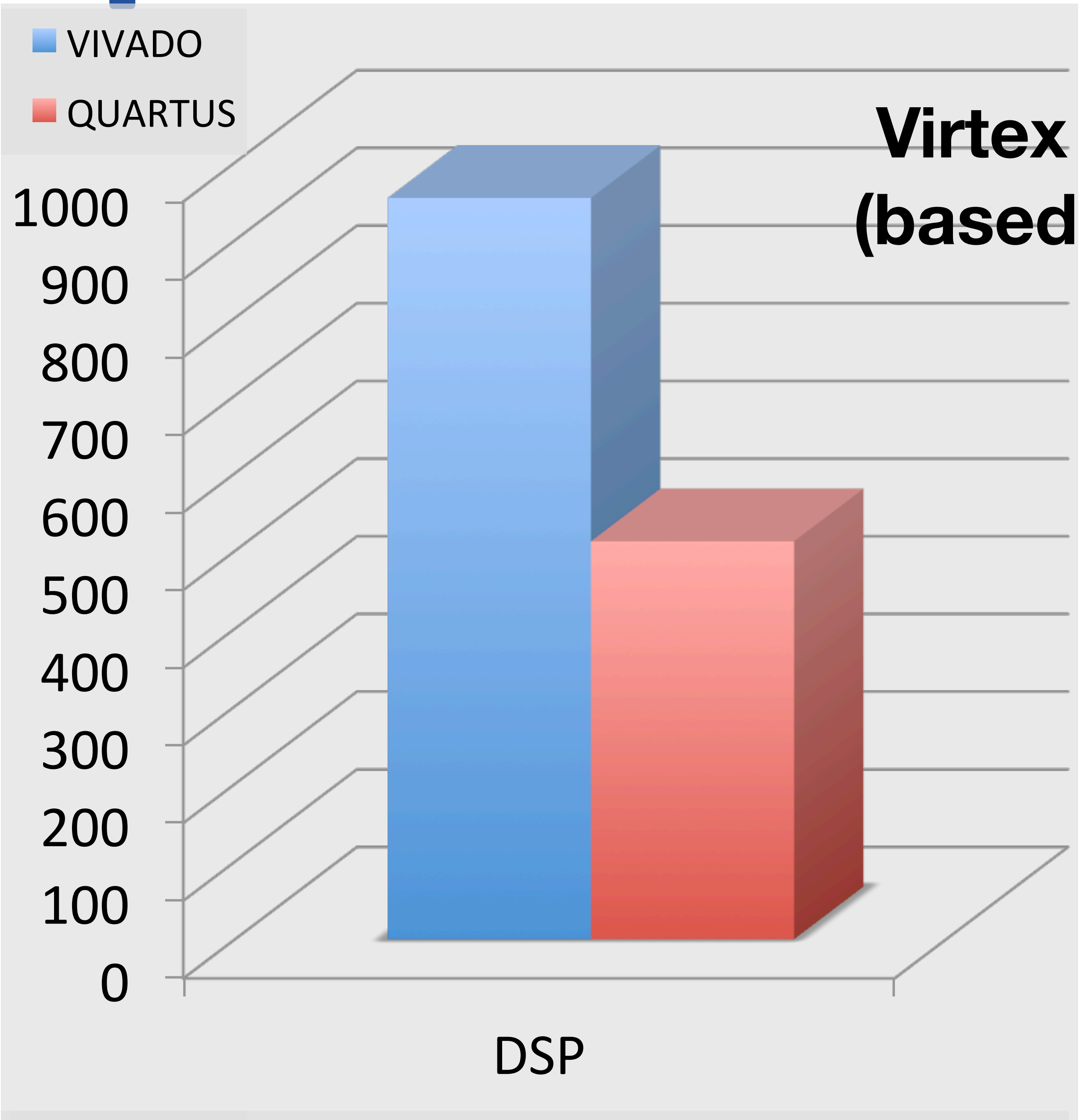


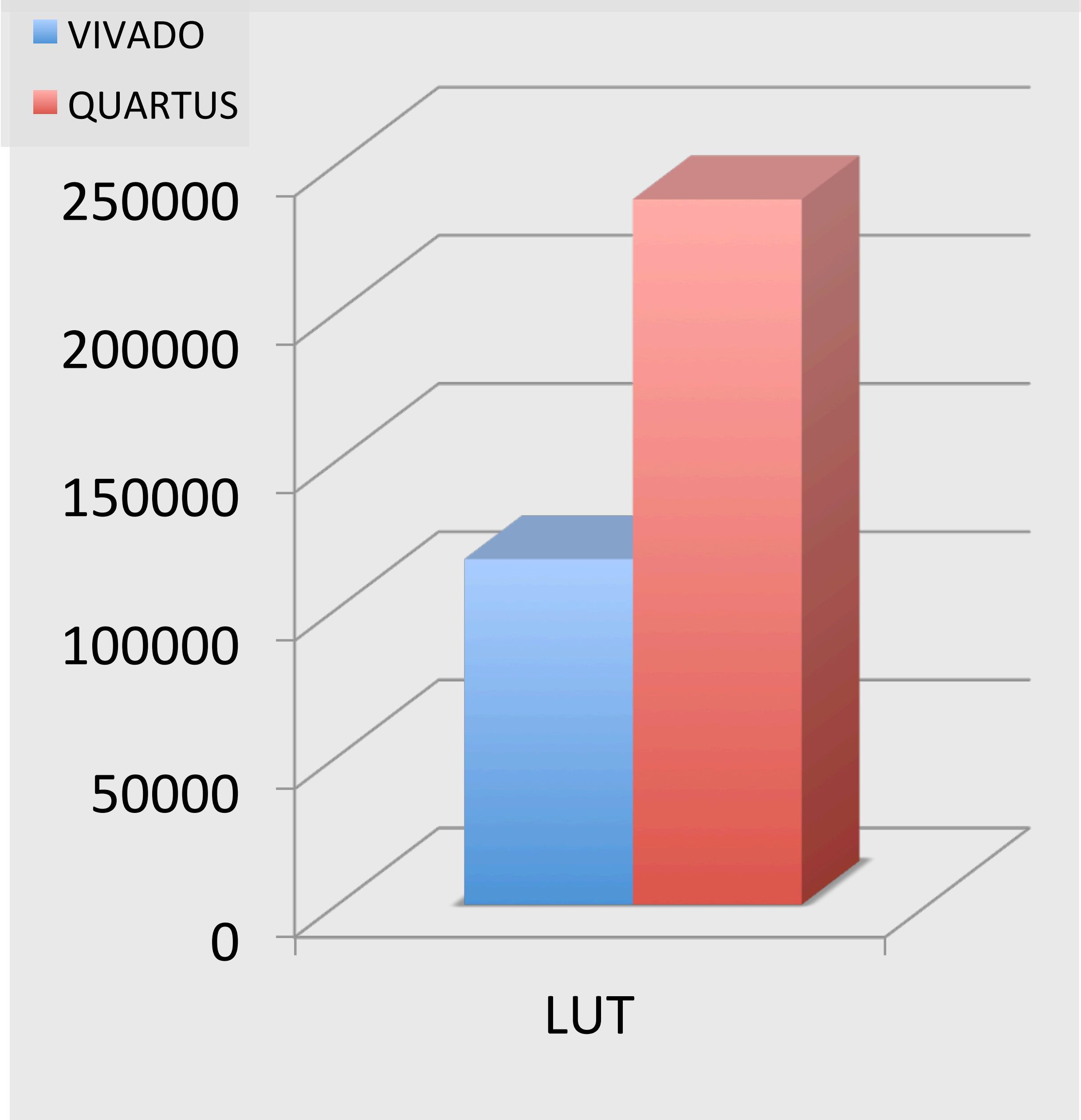
European Research Council

Altera Support

- HLS4ML supports Xilinx FPGAs from beginning
- Working to extend the package to work with Altera
  - Work in progress
  - Technical complications slowed us down (software licences, Quartus HLS version @ CERN, etc)
- First results encouraging (based on emulation. To be confirmed with actual deployment on card)

Virtex V vs. Stratix V (based on simulation)





Deep Learning on the Cloud

Inference on the cloud

- *In the (near) future, DAQ/HLT farms will be based on heterogenous computing*
  - *CPU+GPU / CPU+FPGA*
  - *Mainly to accelerate slow algorithms (e.g., tracking) through parallelisation*
  - *Also useful for ML inference*
- *R&D on heterogeneous environments on commercial clouds*
  - *provides easy-to-use CPU+FPGA (or GPU) ecosystem*
  - *allows further R&D: inference on demand from the CPU-based HLT farm to the FPGAs/GPUs on the cloud*

European Research Council

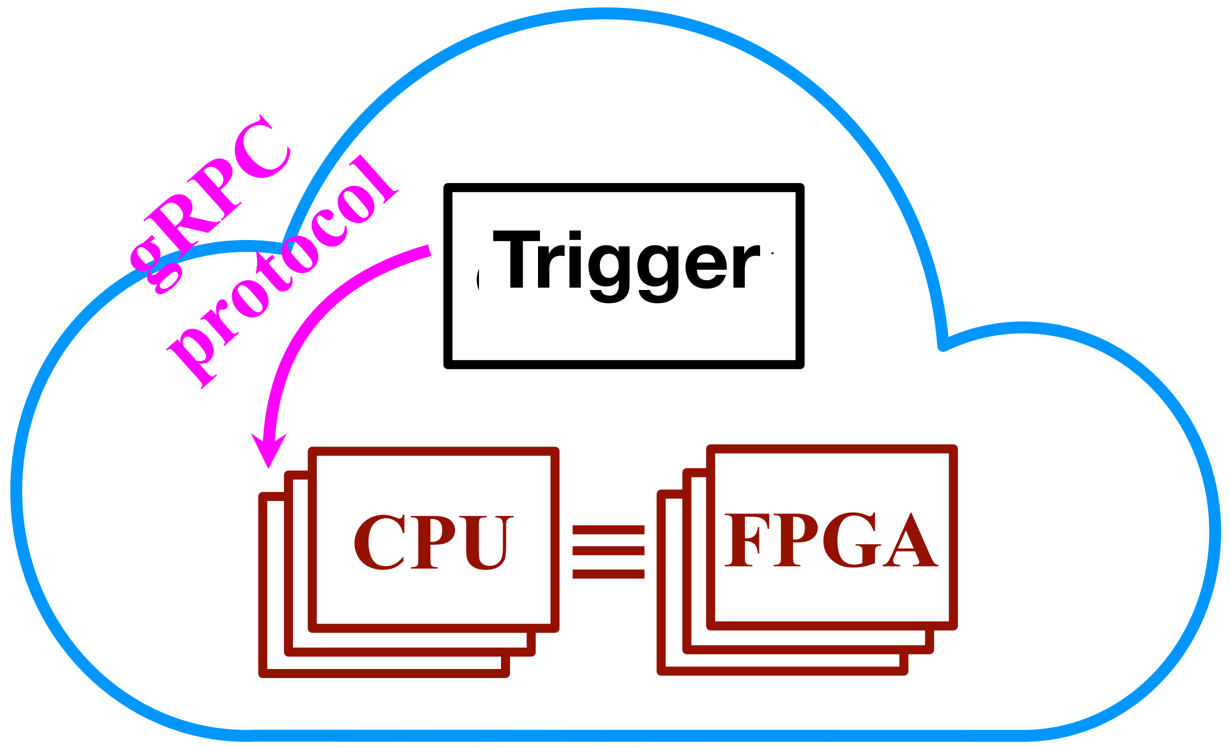
Microsoft Brainwave

A Reconfigurable Fabric for Accelerating Large-Scale Datacenter Services

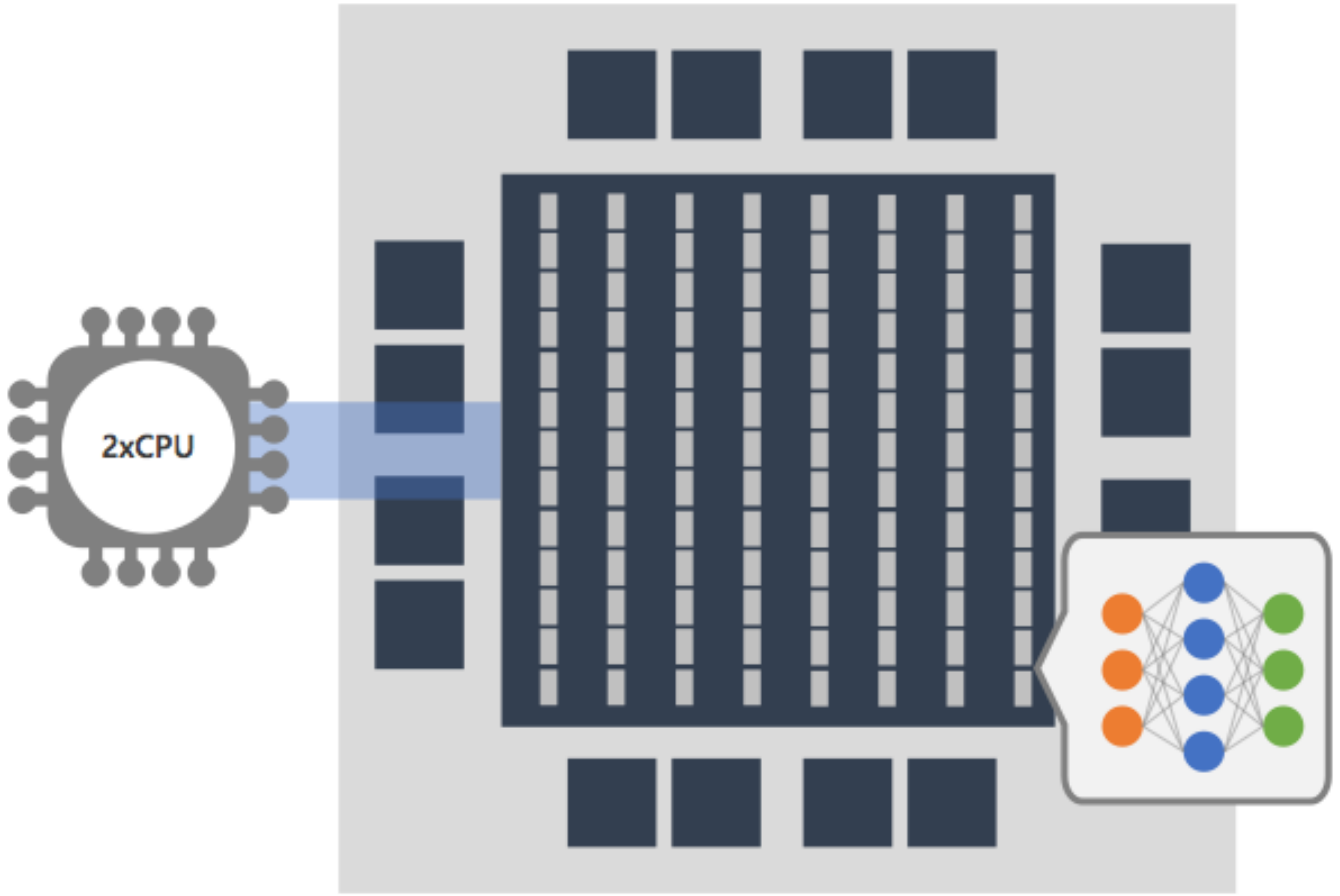
Andrew Putnam Adrian M. Caulfield Eric S. Chung Derek Chiou[1]
Kypros Constantinides[2] John Demme[3] Hadi Esmaeilzadeh[4] Jeremy Fowers
Gopi Prashanth Gopal Jan Gray Michael Haselman Scott Hauck[5] Stephen Heil
Amir Hormati[6] Joo-Young Kim Sitaram Lanka James Larus[7] Eric Peterson
Simon Pope Aaron Smith Jason Thong Phillip Yi Xiao Doug Burger

https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/Catapult_ISCA_2014.pdf

Heterogeneous Edge Resource



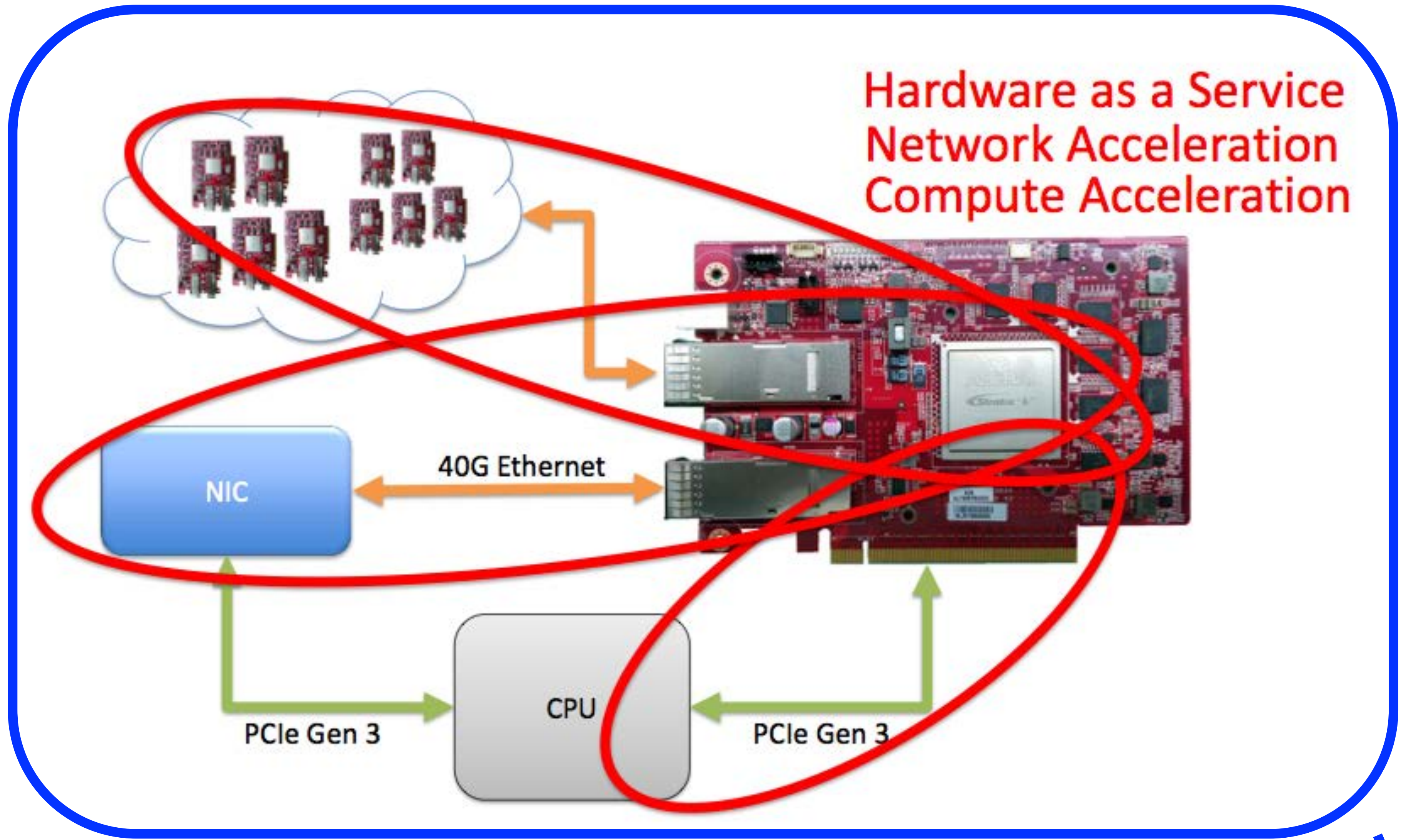
- Cloud service has latency
- Run CMSSW on Azure cloud machine → simulate local installation of FPGAs ("on-prem" or "edge")
- Provides test of "HLT-like" performance

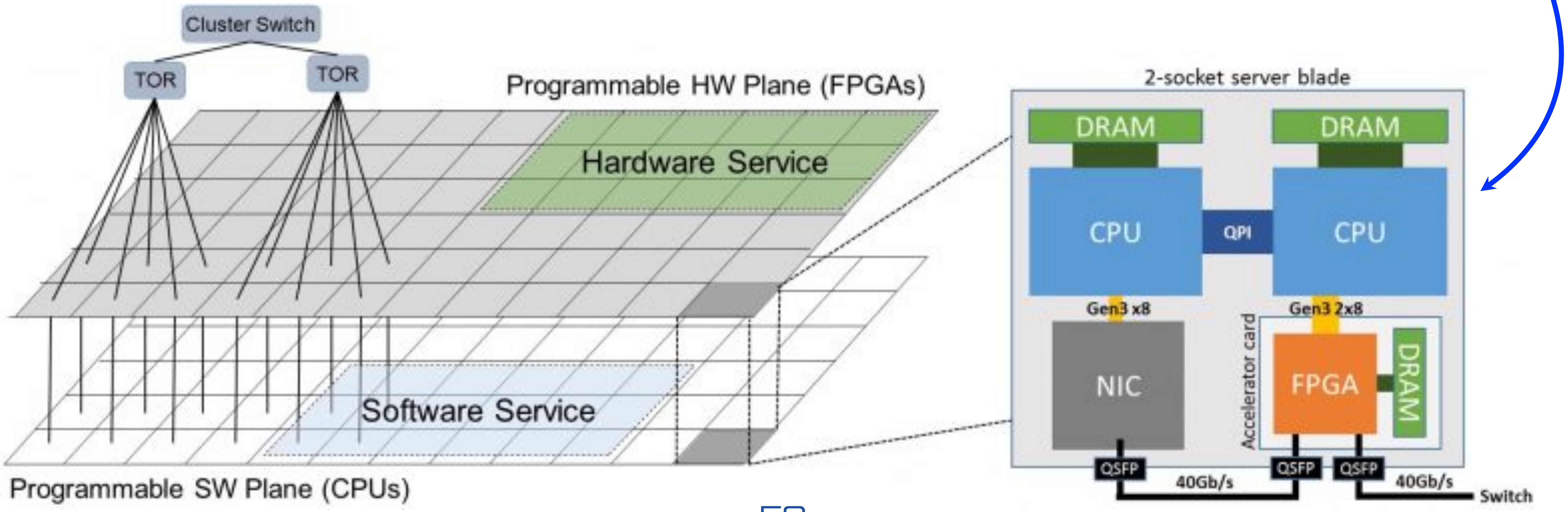


European Research Council

Brainwave at scale

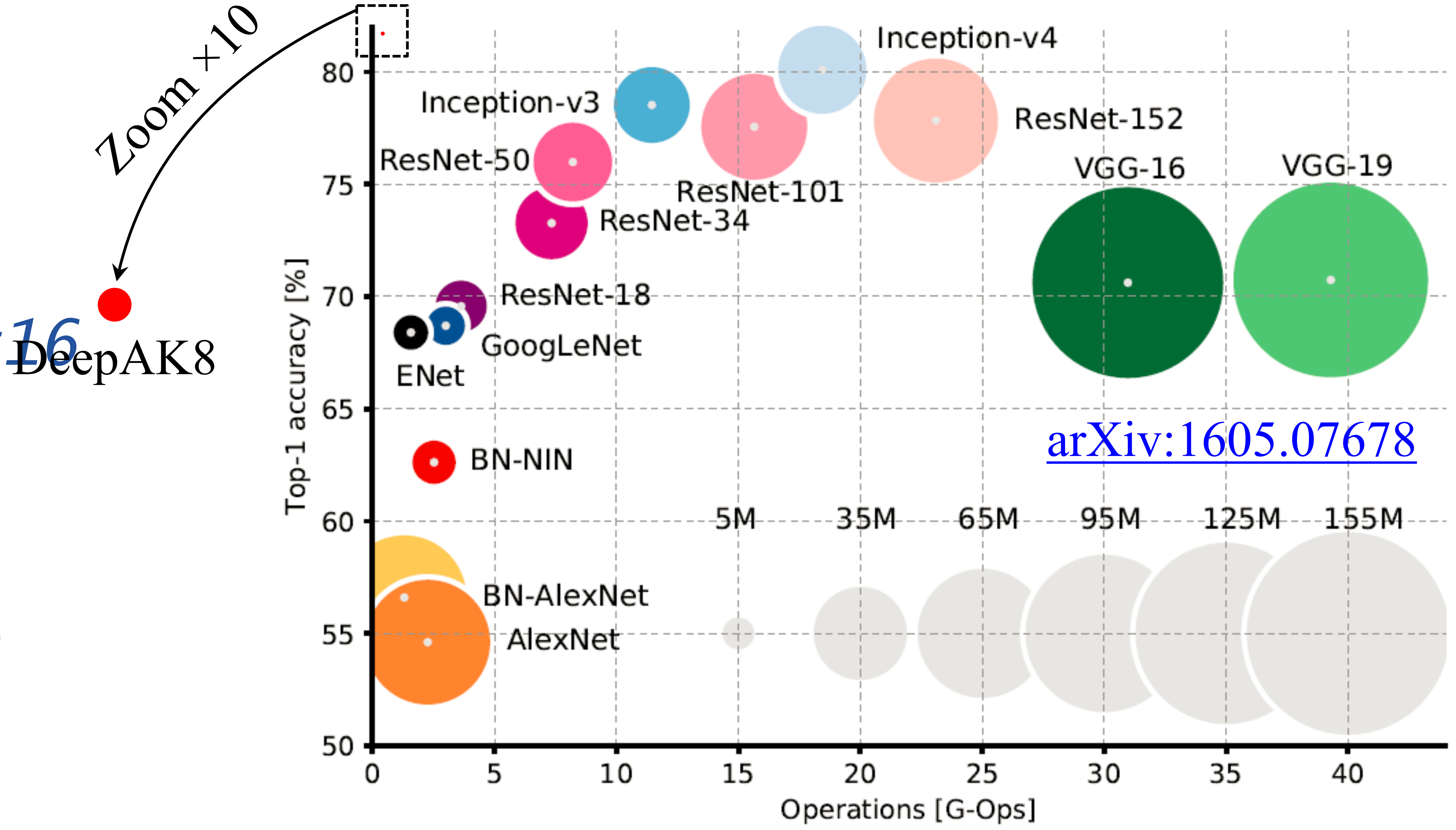
- Provides a full service at scale (more than just a single co-processor)
- Multi-FPGA/CPU fabric, accelerating both compute and network
  - ➢ Demonstrated large improvements in processing time for Bing searches
- Caveat: only selected pre-trained DNN models currently available





Pros & Cons

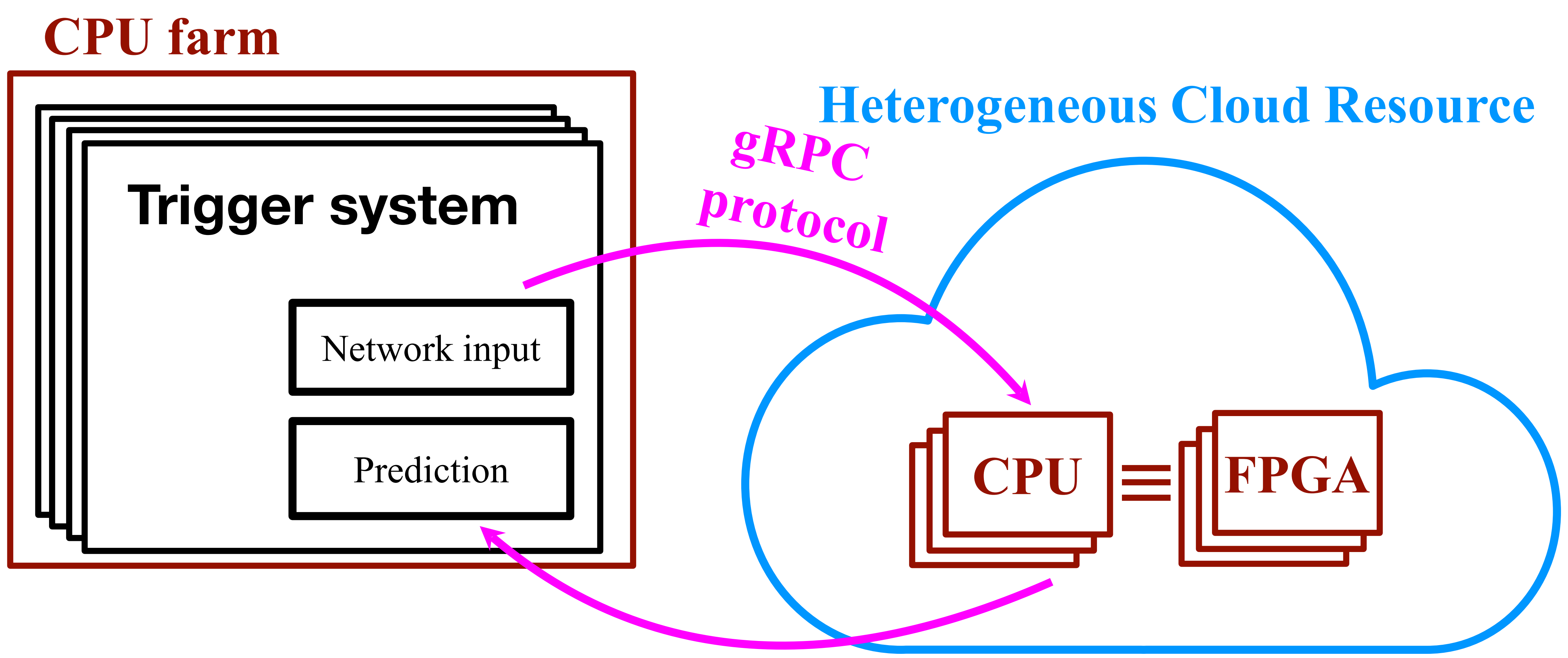
- Commercial clouds focus on what is sellable
  - supports computing-vision off-the-shelf networks (ResNet50, ResNet512, DenseNet121, VGGNet16
  - (for now) reduced flexibility: doesn't allow customised architectures. Input has to be an image
  - (longer term) more architectures will become available
- As long as one of these networks is good for the problem at hand, implementation is optimized (beyond what HLS might do)



- ResNet50: 25M parameters, 7B operations
- Examples of large networks used in CMS:
  - DeepAK8, 500K parameters, 15M operations
  - DeepDoubleB, 40K parameters, 700K operations

SONIC

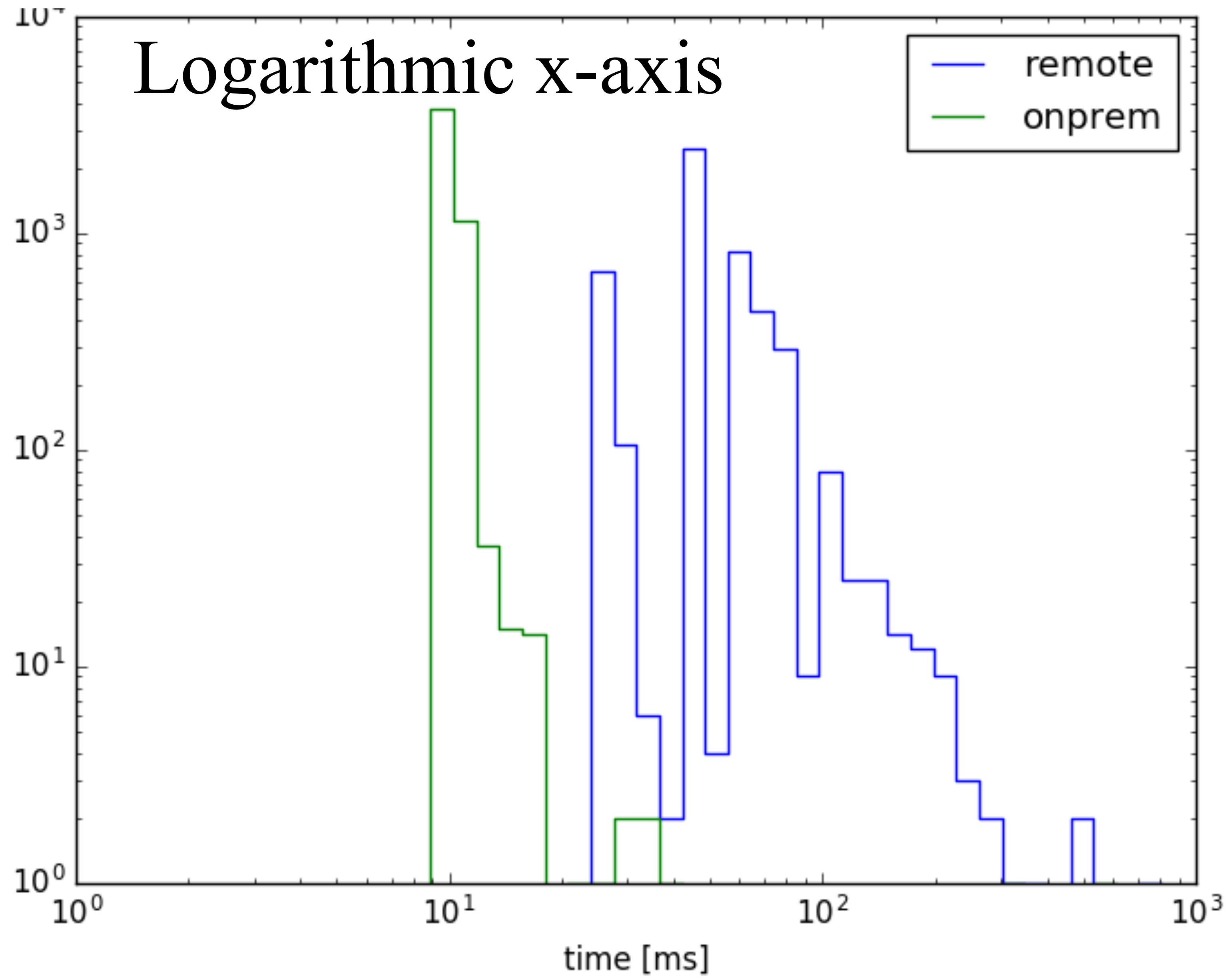
- SONIC (a Services for Optimized Network Inference on Coprocessors) is a framework to exploit cloud resources for on-demand inference
- CPU runs "locally" (for us at FNAL) and sends data to the cloud system
- FPGAs there set to run our inference problems
- answer communicated back via gRPC protocol (driven by Microsoft infrastructure boundaries)

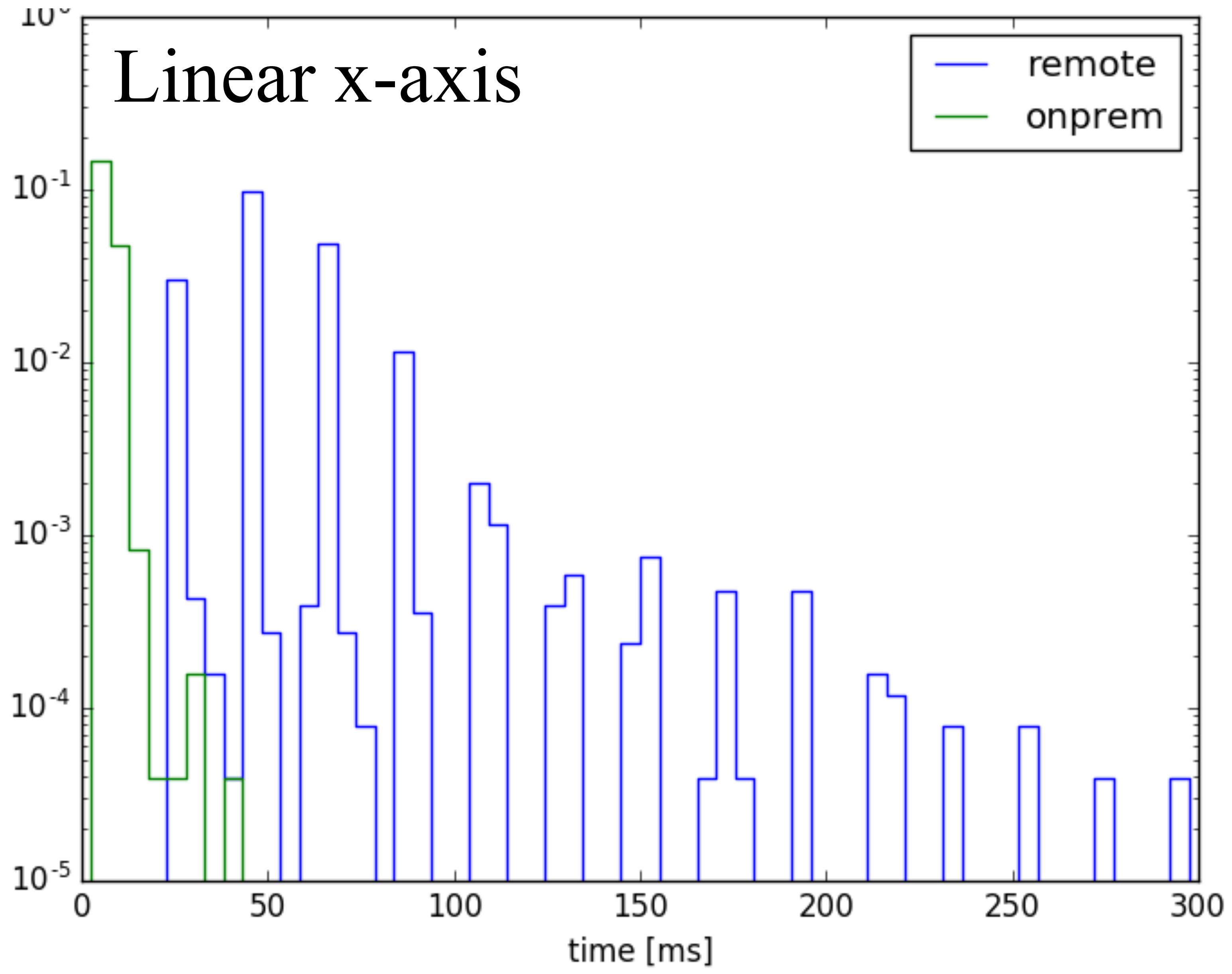


CPU comparison:

- Intel i7 3.6 GHz (8 core, TF v1.10) ~ 180 ms
- Intel i7 3.6 GHz (1 core, TF v1.10) ~ 500 ms
- Intel i7 3.6 GHz (1 core, TF v1.06) ~ 1.2 s
- Intel Xeon 2.6 GHz (1 core, TF v1.06) ~ 1.75 s [what we are running on cmslpc]

Testing SONIC

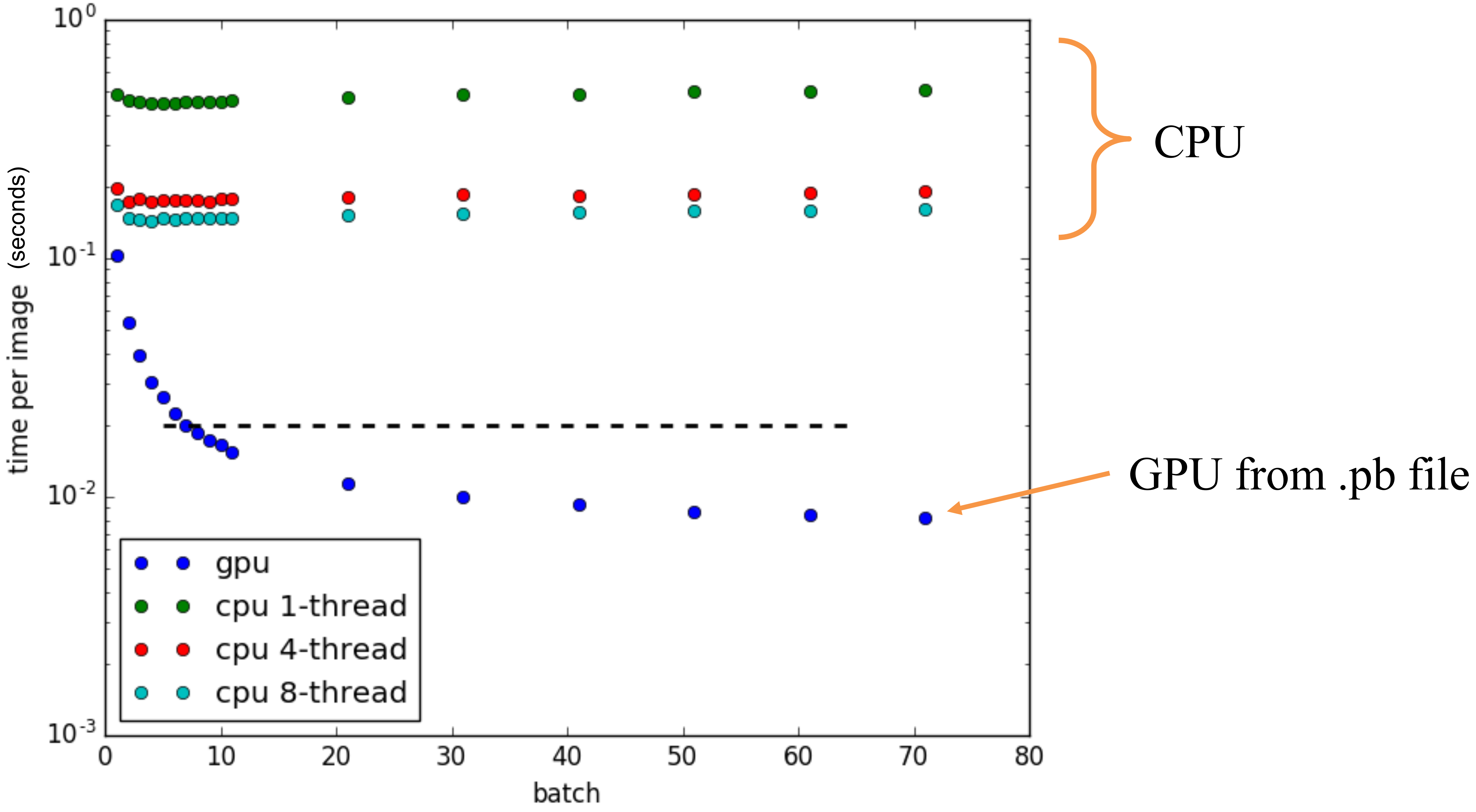




- Good performance in initial tests
  - “remote”: cmslpc @ FNAL to Azure (VA), ‹time› = 56 ms
  - “onprem”: run CMSSW on Azure VM, ‹time› = 10 ms
    (~2 ms on FPGA, rest is classifying and I/O)
  - CPU (cmslpc): 1.75 sec
    (6 min to load ResNet50 session)

➢ More than order of magnitude improvement!

Using GPUs instead



- Benchmark: Nvidia GTX 1080 (GPU), Intel i7 3.6 GHz (CPU)
- All tests use .pb file with Brainwave version of ResNet50
- Using classic ResNet50 implementation w/ CuDNN: faster on GPU by 5–10×

Comparison to GPUs and CPUs

Brainwave w/ SONIC

- Featurizer: 1.8 ms (FPGA)
- Classifier: 2 ms (CPU)
- Infrastructure: 6 ms (trying to improve)
- Transit time: 10 ms (speed of light, Chicago to Virginia) + network switching, etc.
- Total: ~10 ms (onprem), ~56 ms (remote)

CPU

- 1.75 s (cmslpc CPU)
- 500 ms (new CPU, TF version)

GPU

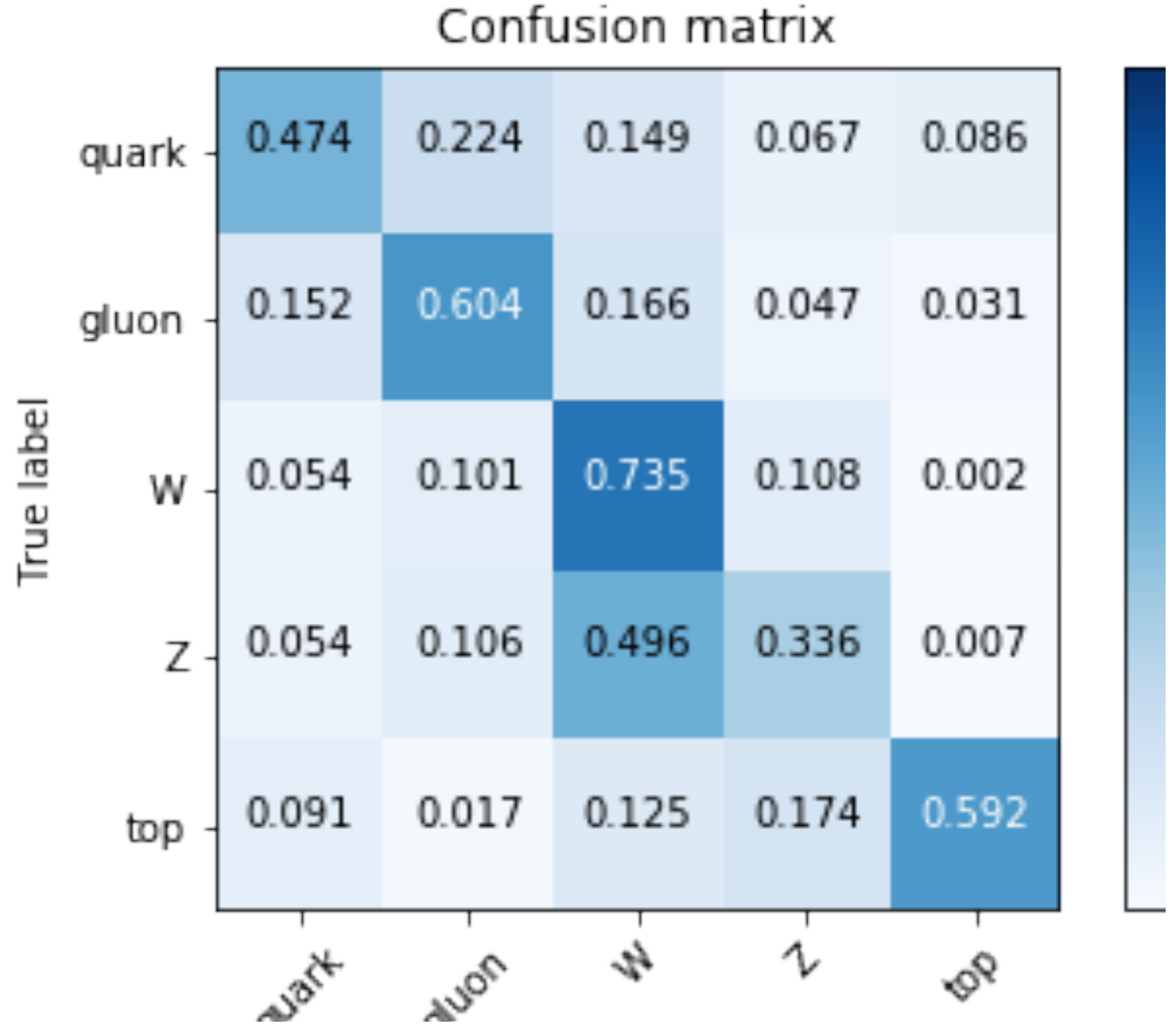
- 100 ms (batch 1)
- 15 ms (batch 10)

European Research Council

Jet Tagging with ResNet on Cloud

- Use feature set generated by ResNet50, train new fully-connected classifier
  - o Brainwave accelerates training
    (heaviest component is evaluating ResNet50 to produce feature set)
- CNNs have been used in jet image classification: arXiv:1709.04464
- Proposed "realistic" test:
  - o Compute jet discriminator values
    (q, g, W, Z, t)
  - o Run module in miniAOD sequence
- Can also be used for other experiments
  - o e.g. NOvA: identify neutrino events

Preliminary result w/ small datase



Outlook

- HLS4ML aims to be a flexible tool to implement your home-made NN in a trigger/DAQ system where low latency matters
  - Now works with TensorFlow and PyTorch for Dense Neural networks
    - Working to support ONNX format
  - Working on new architecture support
    - Boosted Decision Trees
    - Convolutional NNs (1D & 2D)
    - Recurrent NNs (GRUs, LSTMs, etc)
    - Graph Networks
  - Extra functionalities added
    - New activation functions
    - Batch Normalization
    - Layer concatenate
    - Max Pooling
    - …

European Research Council

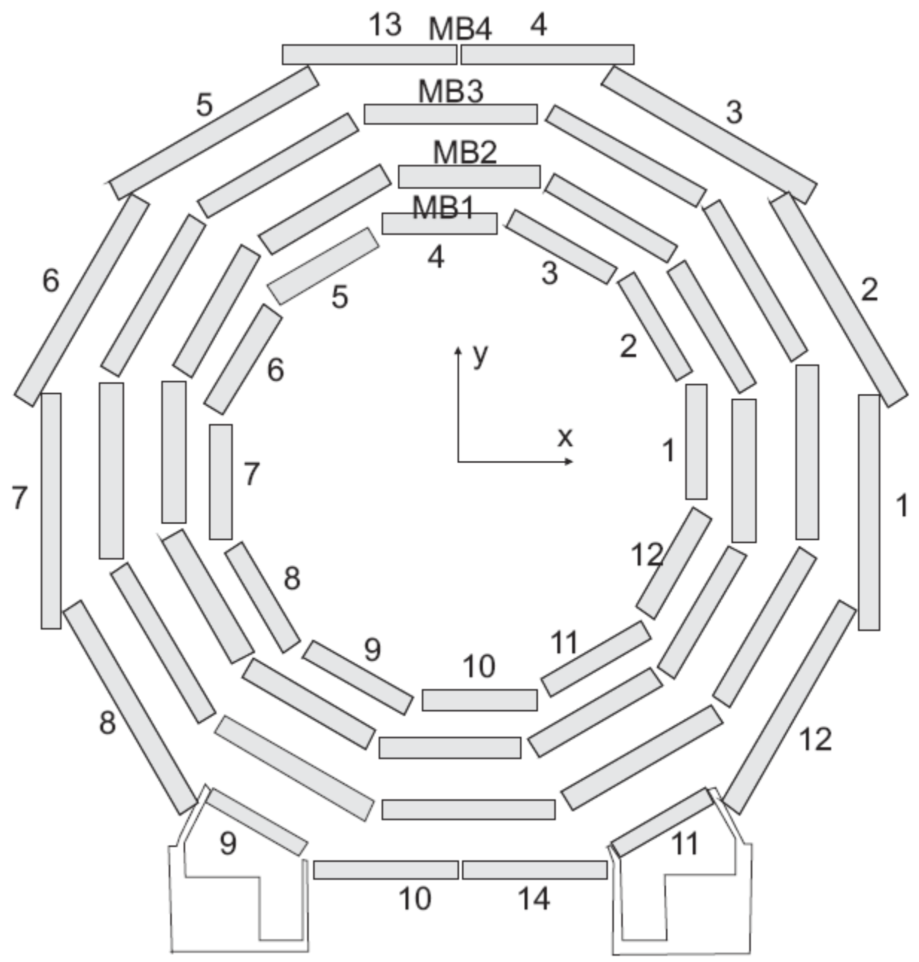
Backup

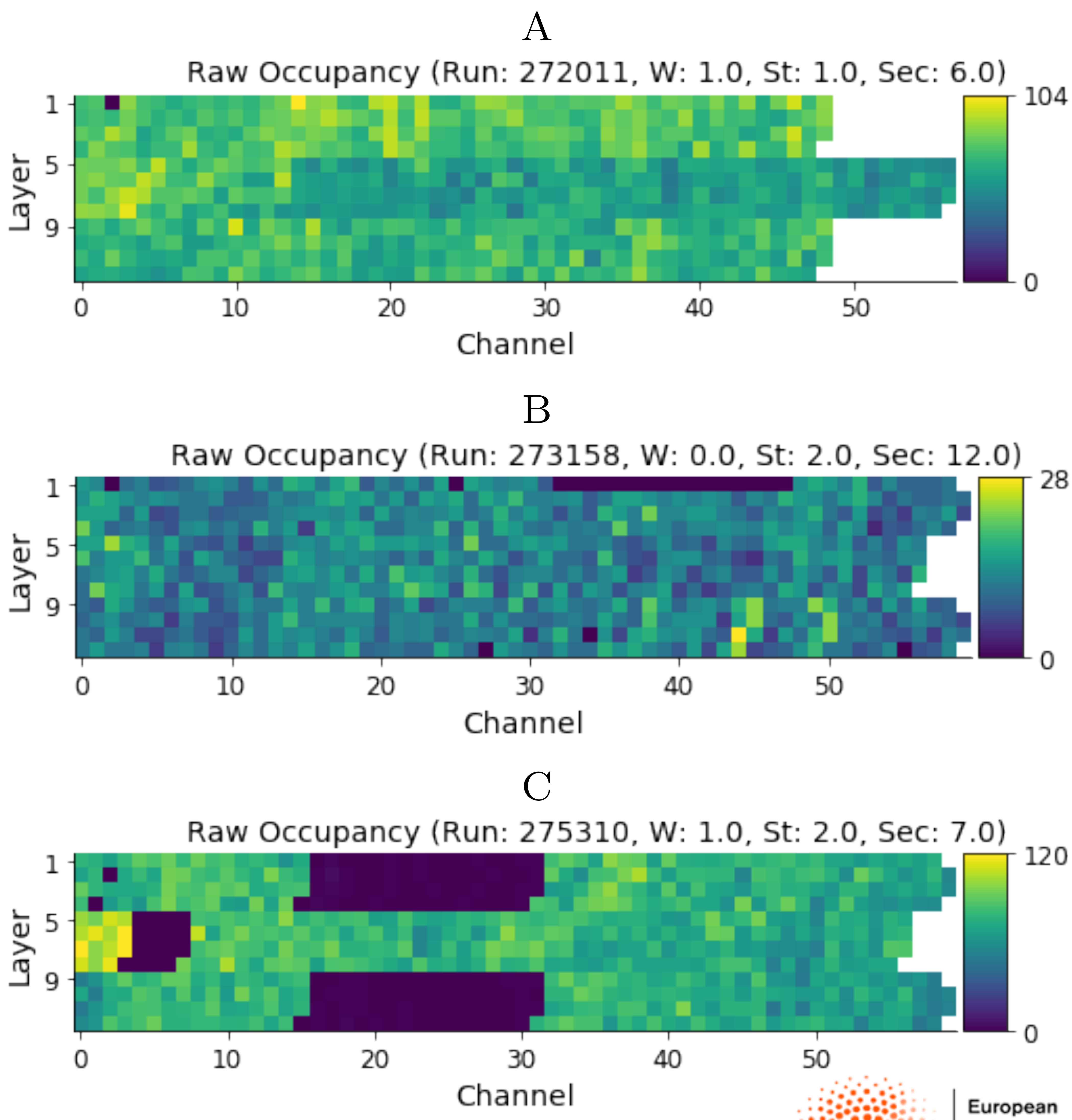
Detector Monitoring

European Research Council

Data Quality Monitoring

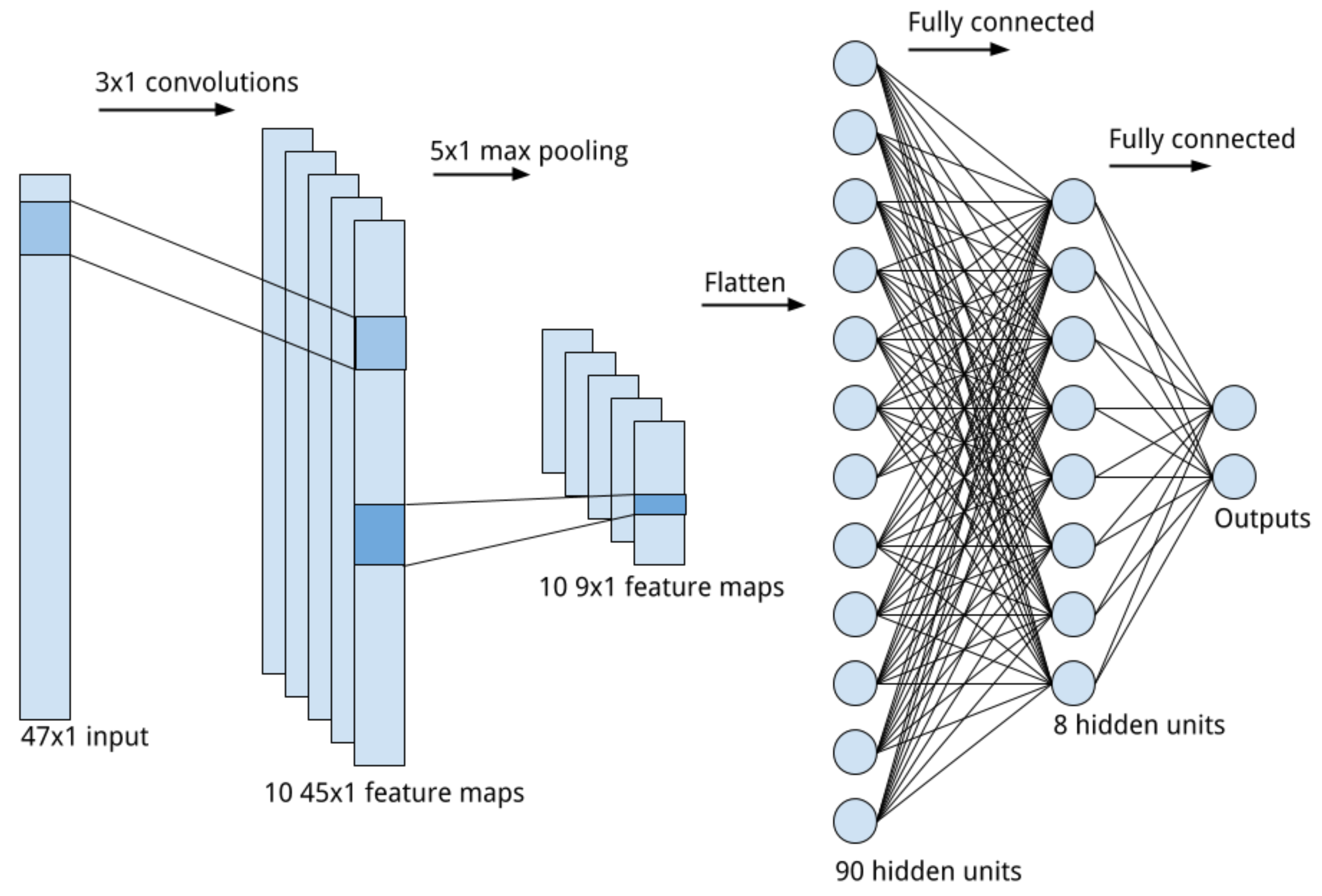
- When taking data, >1 person watches for anomalies in the detector 24/7
- At this stage no global processing of the event
- Instead, local information from detector components available (e.g., detector occupancy in a certain time window)

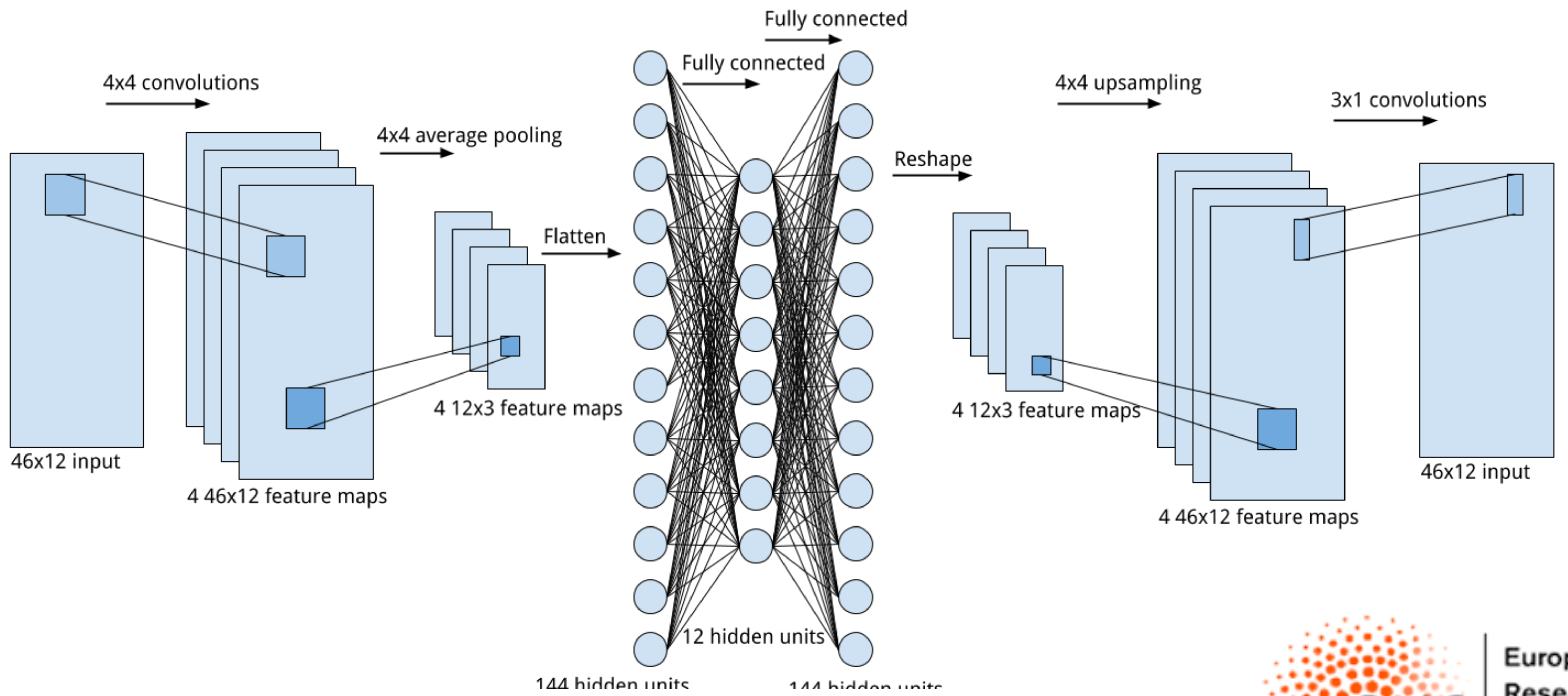




Two approaches

- Given the nature of these data, ConvNN are a natural analysis tool. Two approaches pursued
  - Classify good vs bad data. Works if failure mode is known
  - Use autoencoders to assess data “typicality”. Generalises to unknown failure modes



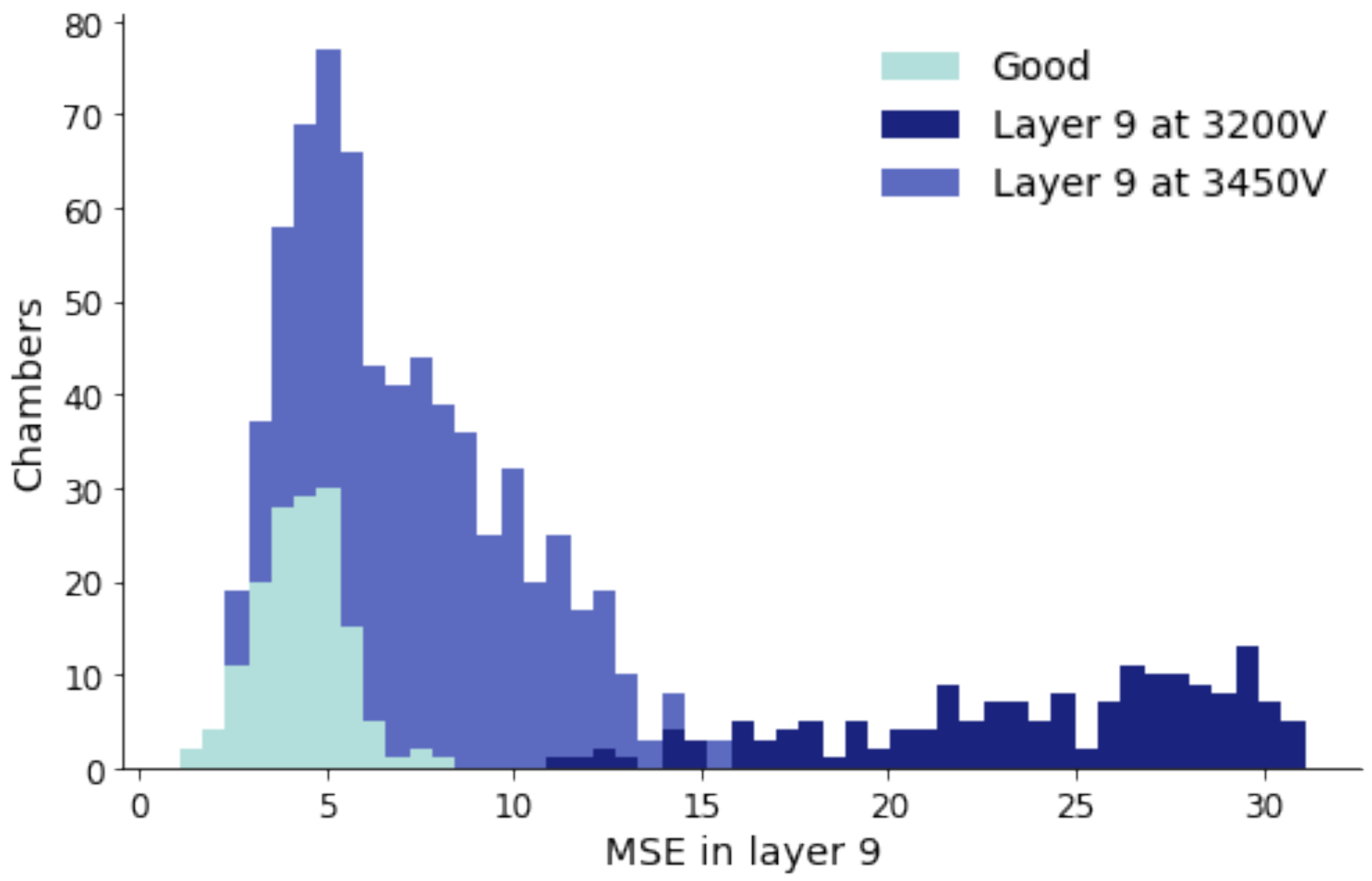


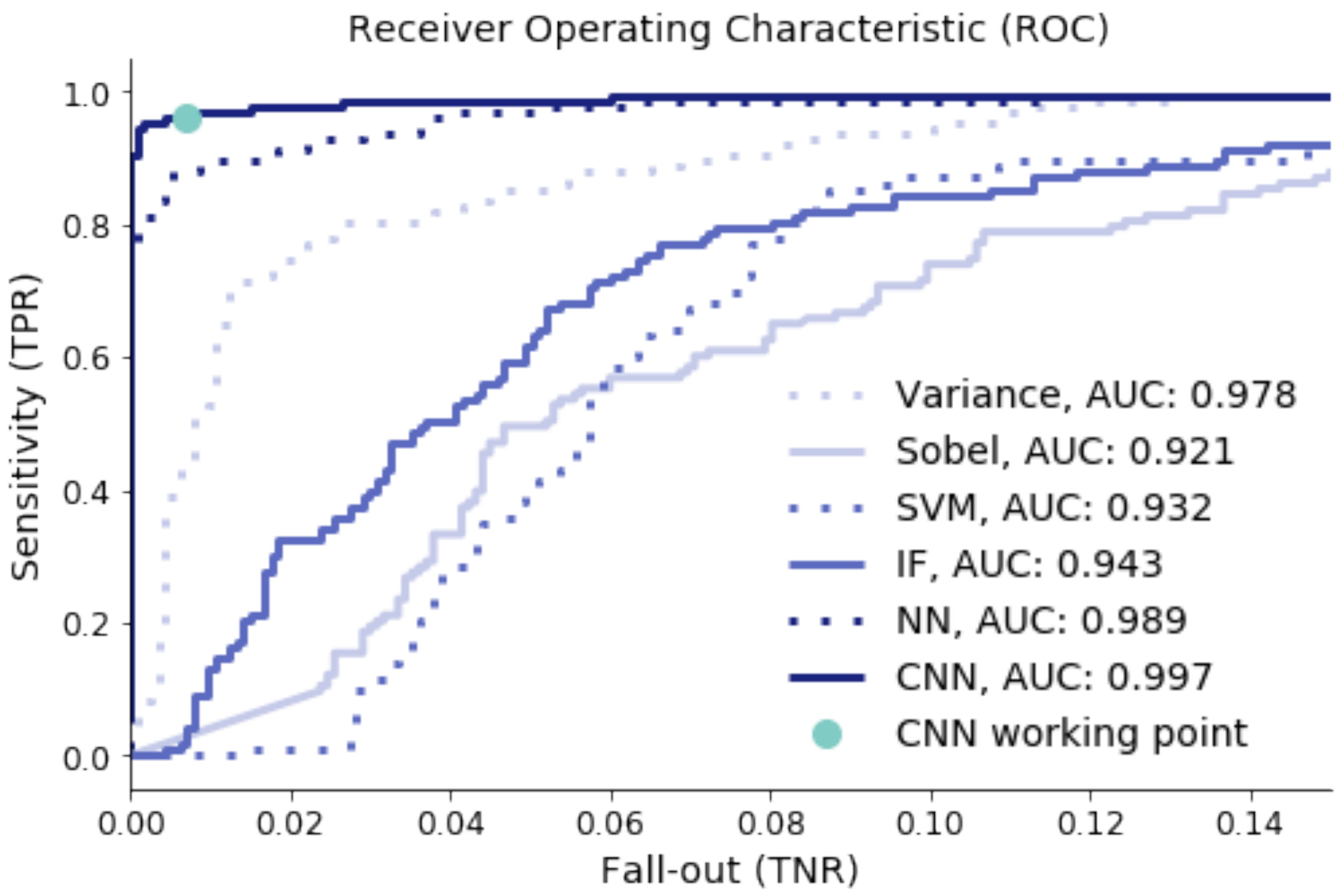
A. Pol et al., to appear soon

Two approaches

- *Given the nature of these data, ConvNN are a natural analysis tool. Two approaches pursued*
  - *Classify good vs bad data. Works if failure mode is known*
  - *Use autoencoders to assess data “typicality”. Generalises to unknown failure modes*

A. Pol et al., to appear soon





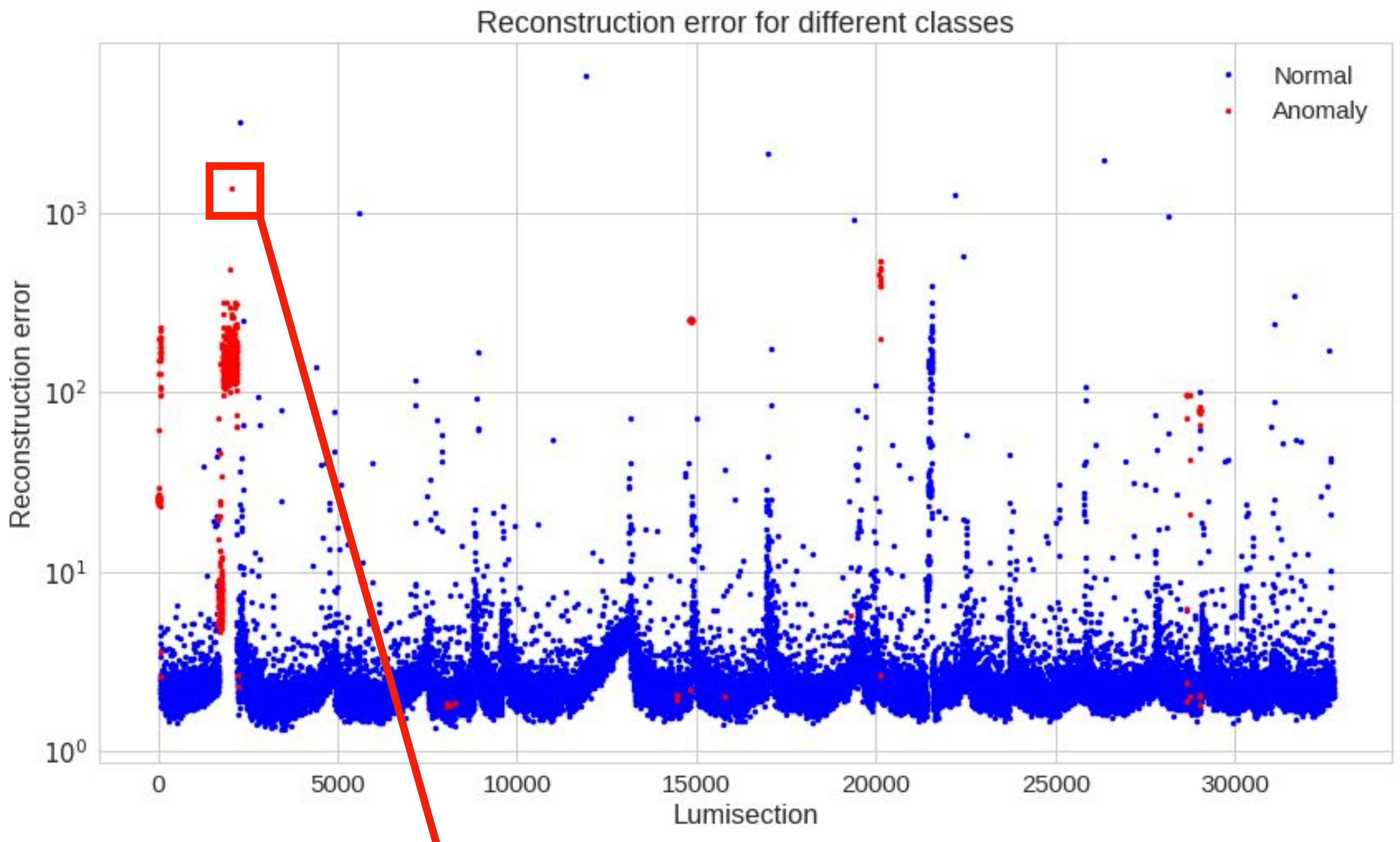
Data Quality Certification

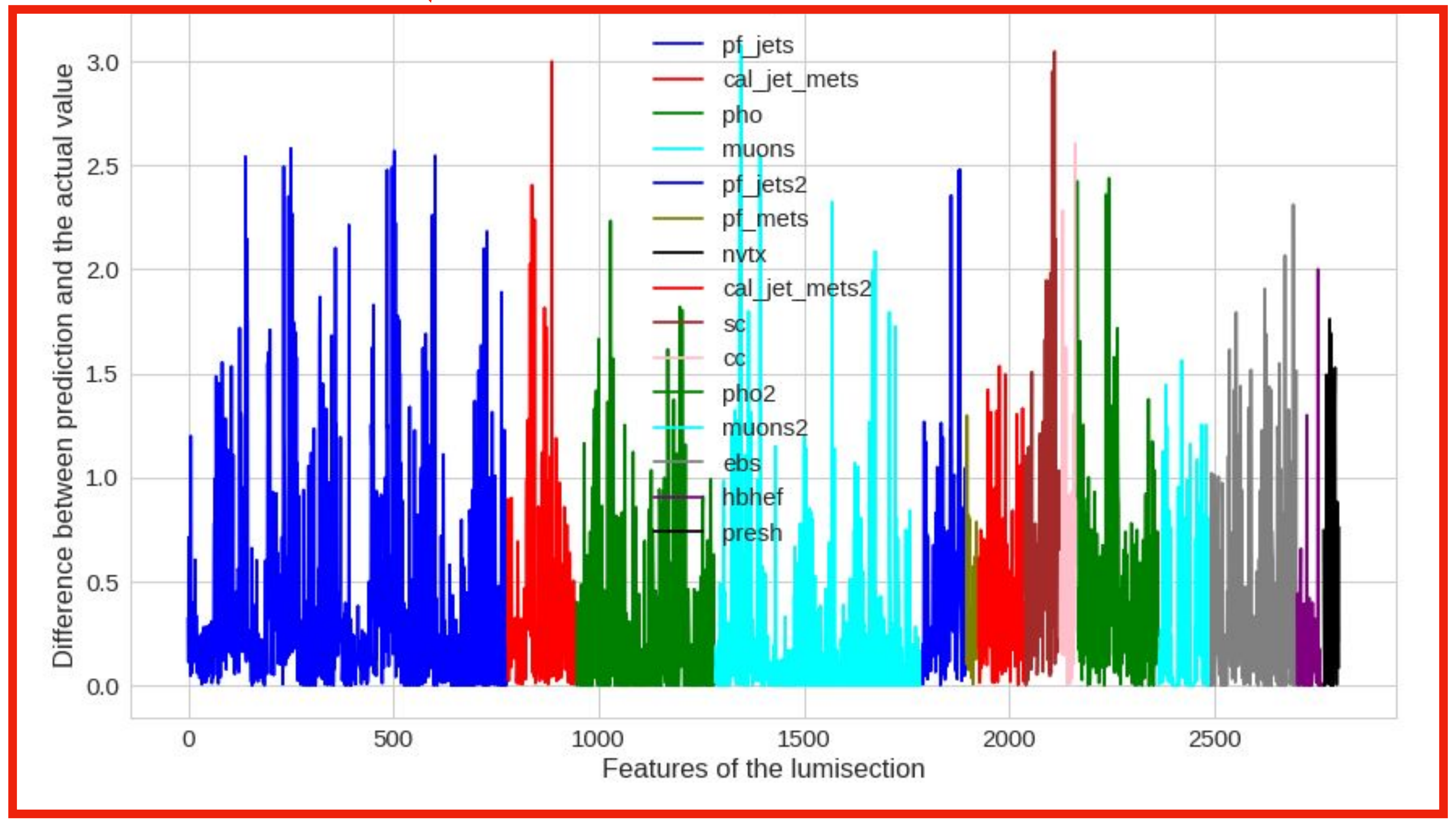
- *Autoencoder-based 1-class approach generalises to later stages of quality assessment*

  - *after reconstruction of the events, event reconstruction allows a global assessment (w.g., looking at electrons, muons, etc rather than hits in the detector)*

  - *A global autoencoder can spot all these features*

  - *Monitoring individual contributions to loss function (e.g., MSE) one can track the problem back to a specific physics object/detector component*





F. Široký et al., to appear sooner or later

HL4ML: FPGA details

Xilinx Vivado 2017.2

Results are slightly different in other versions of Vivado

e.g. 2016.4 optimization is less performant for Xilinx ultrascale FPGAs

Clock frequency: 200 MHz

Latency results can vary (~10%) with different clock choices

FPGA: Xilinx Kintex Ultrascale (XCKU115-FLVB2104)

Results are slightly different in other FPGAs

e.g. Virtex-7 FPGAs are slightly differently optimized

Why Deep Learning

- *Neural network can model non linear functions*
  - *the more complex is the network, the more functions it can approximate*
- *Neural network are faster to evaluate (inference) than typical reco algorithm.*
  - *This is the speed up we need*
- *Neural Networks (unlike other kind of ML algorithms) are very good with raw (non-preprocessed) data (the recorded hits in the event)*

$$(pT, \eta, \phi, E)_{\mathrm{OFFLINE}} = f\big((pT, \eta, \phi, E)_{\mathrm{ONLINE}} \big)$$

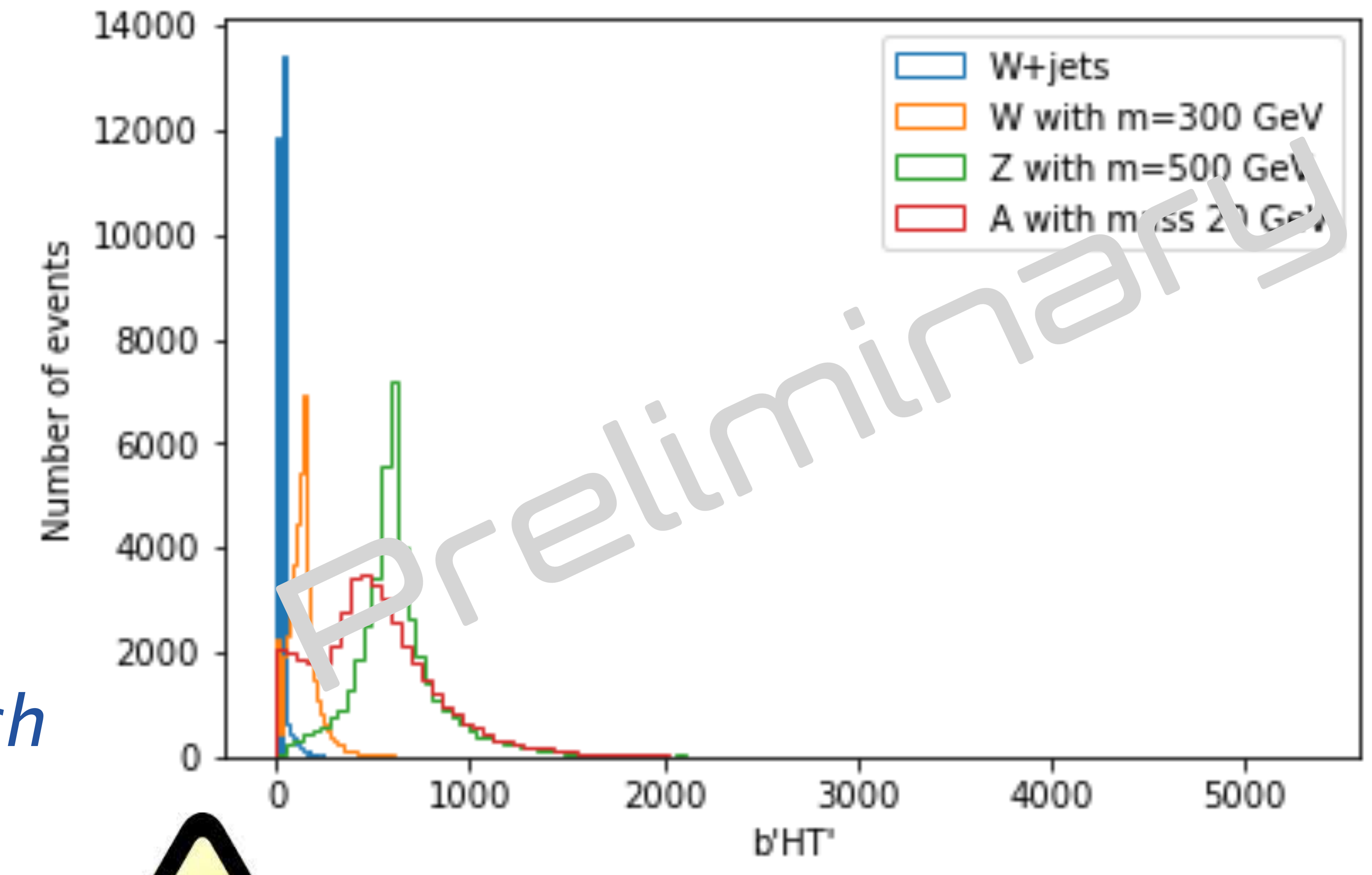
  - *could use them directly on the detector inputs*

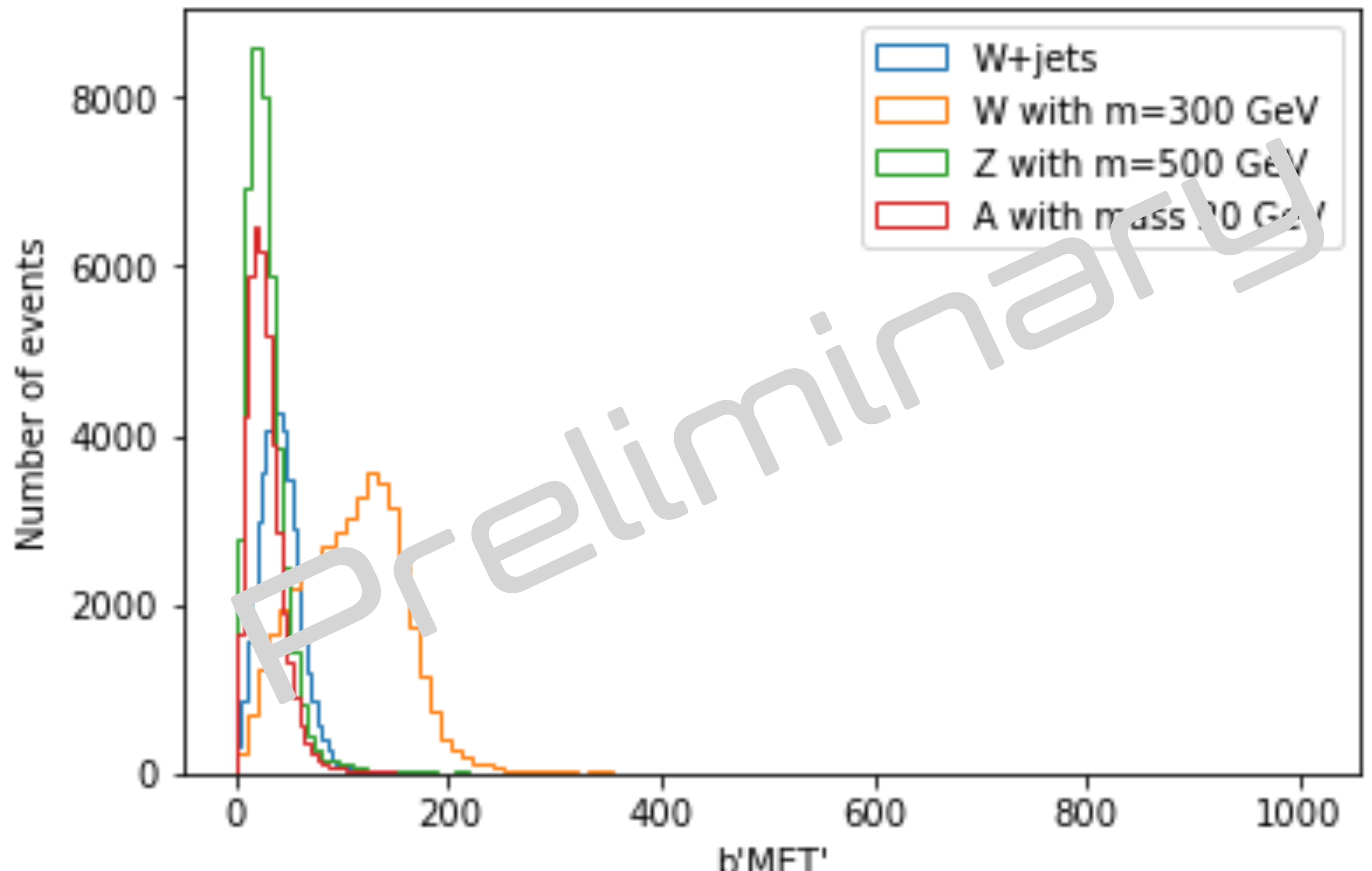
$$(pT, \eta, \phi, E)_{\mathrm{OFFLINE}} = g\big(\text{Event hits} \big)$$

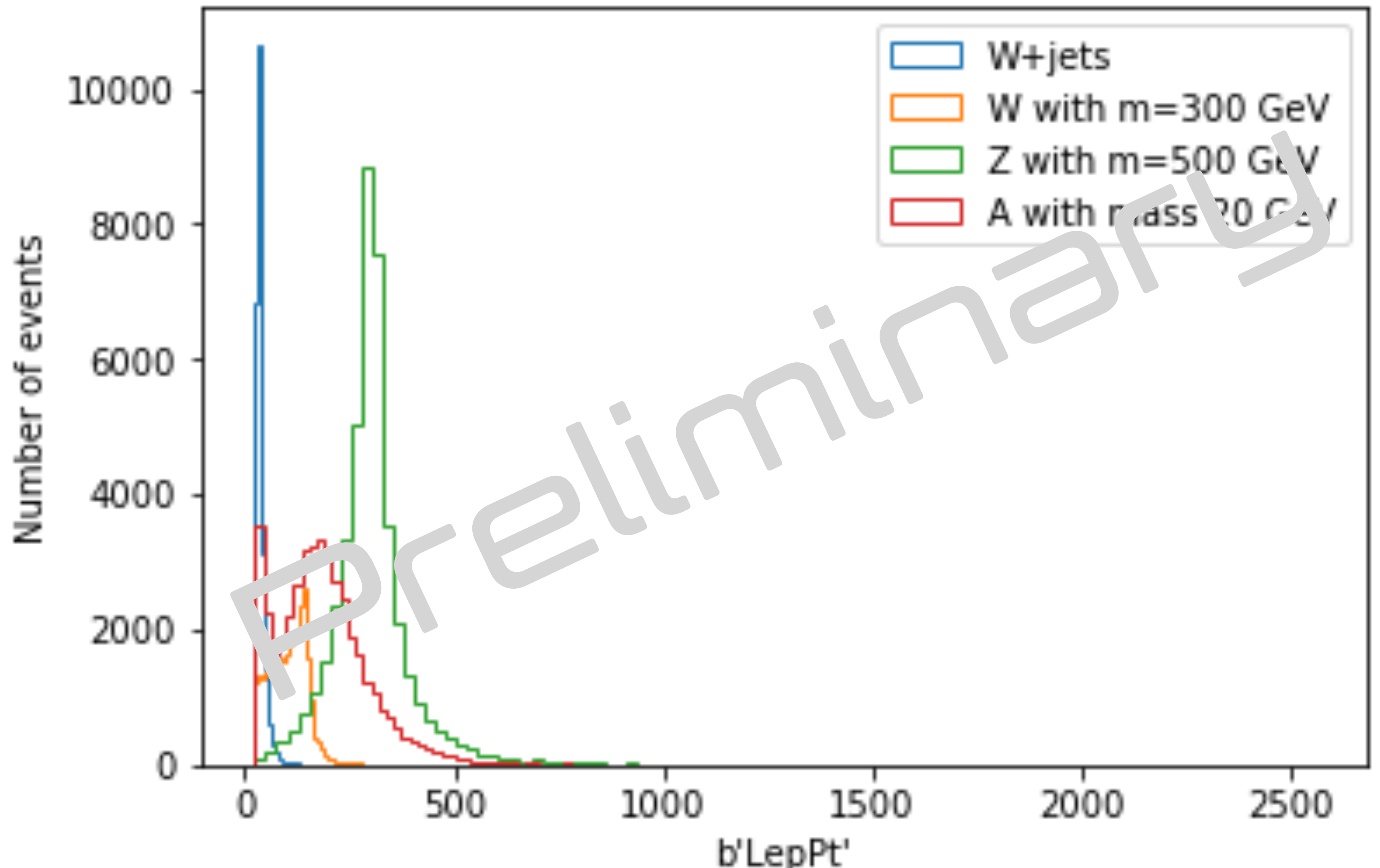
One would have to learn f and g to evaluate them at trigger. Online processing is replaced by offline training

Beyond the toy-model

- Approach works in principle
  - Can identity easily 2 of the 3 models
  - With enough statistics, could see the third
- Might not work in absolute
  - encoder based on physics motivate quantities which are not model-agnostic
  - Use deep:learning: train on raw data directly. ***<u>To be done next</u>***



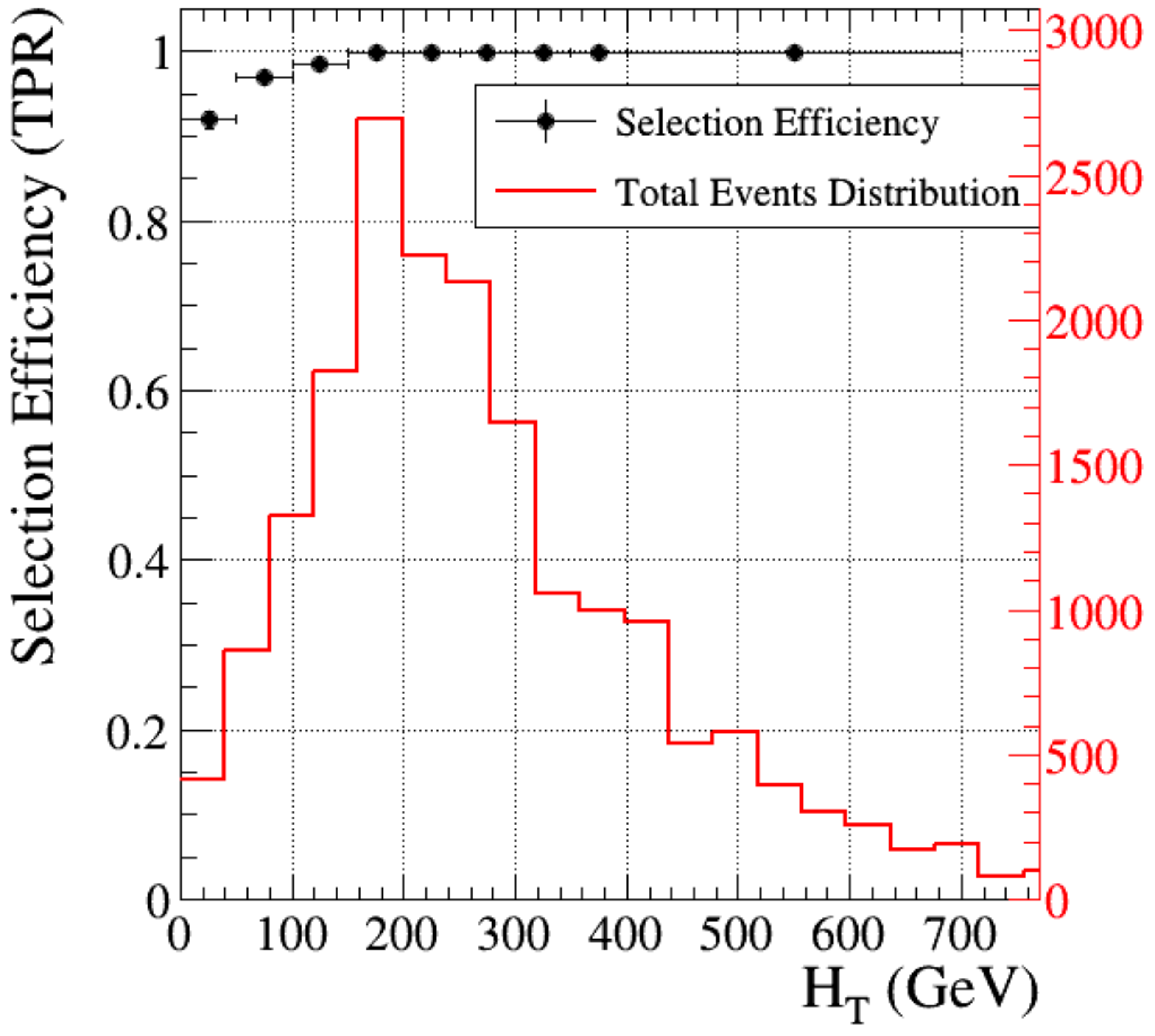


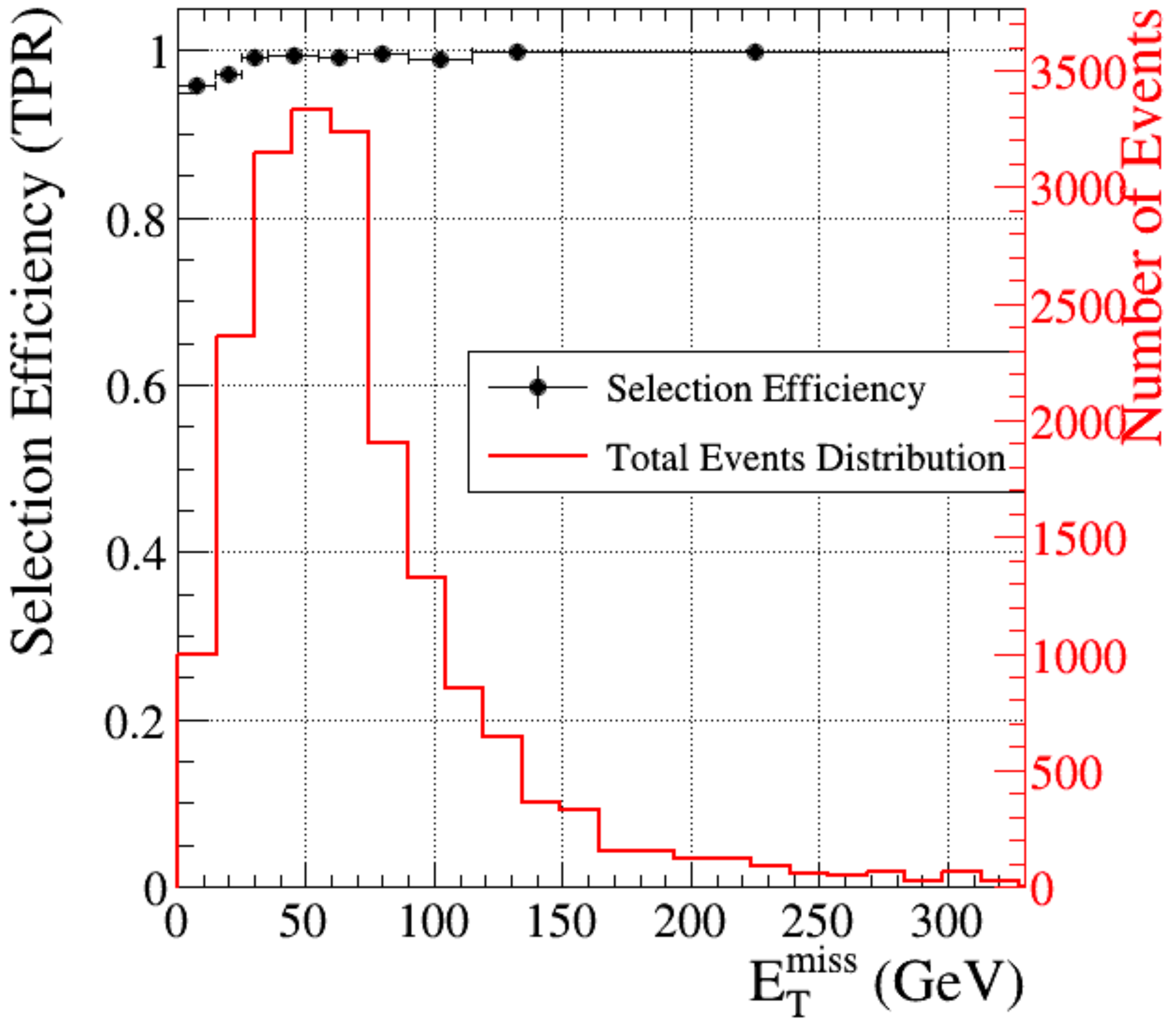


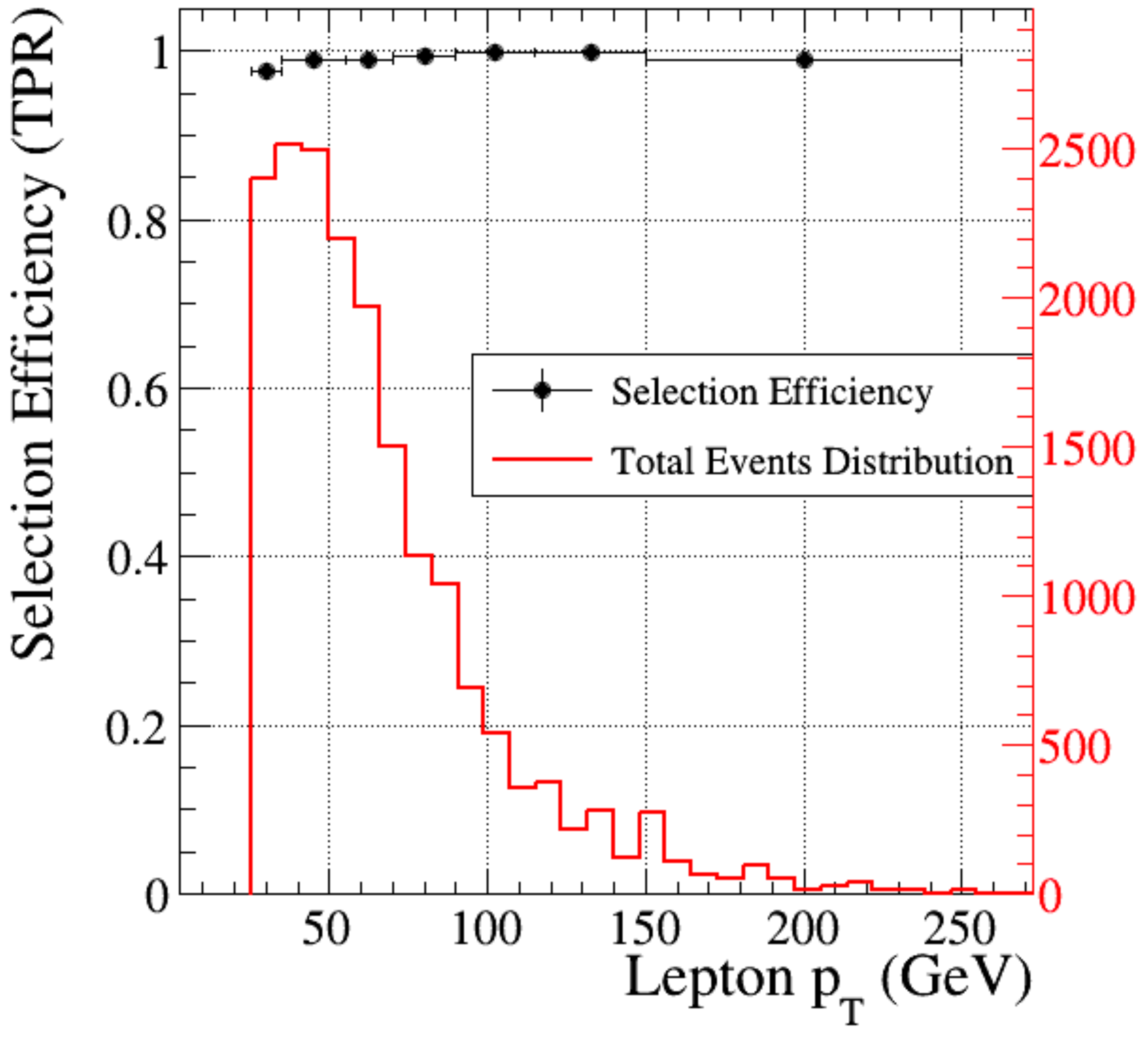
European Research Council

Kinematic Bias?

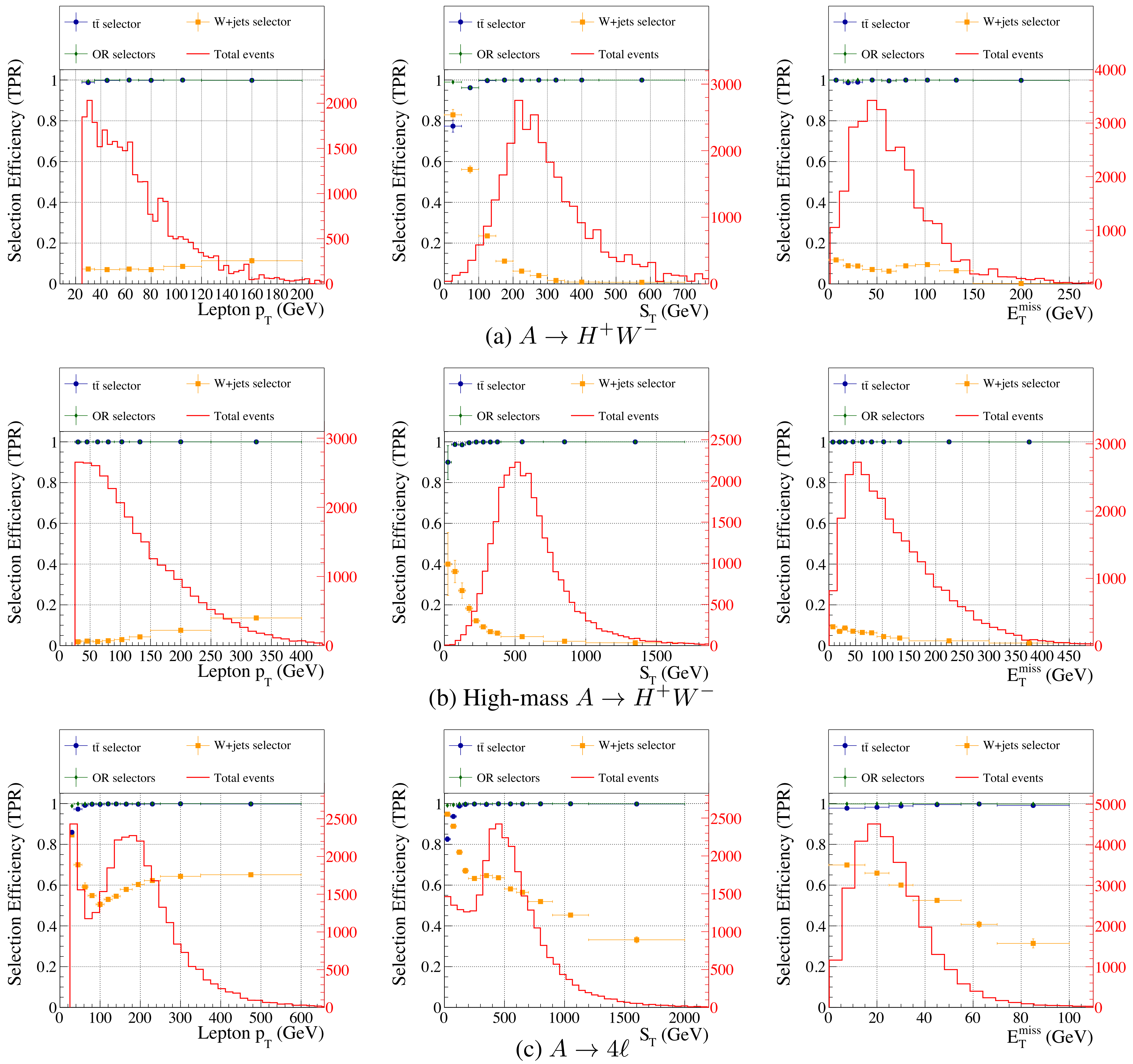
- *With 99% signal efficiency, bias on kinematic variables within the uncertainty of a trigger-efficiency measurement*



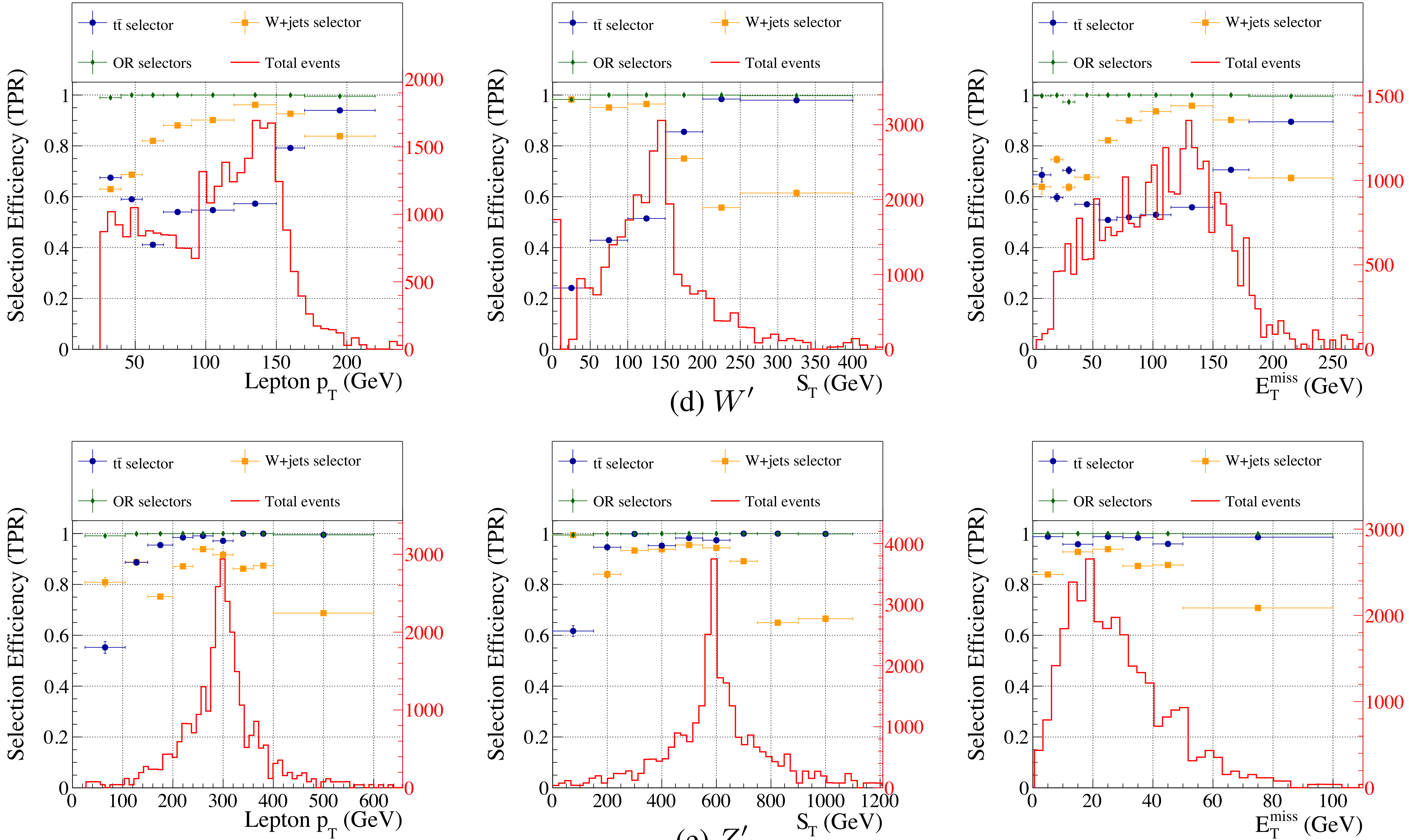




TOPCLASS: do we kill New Physics?



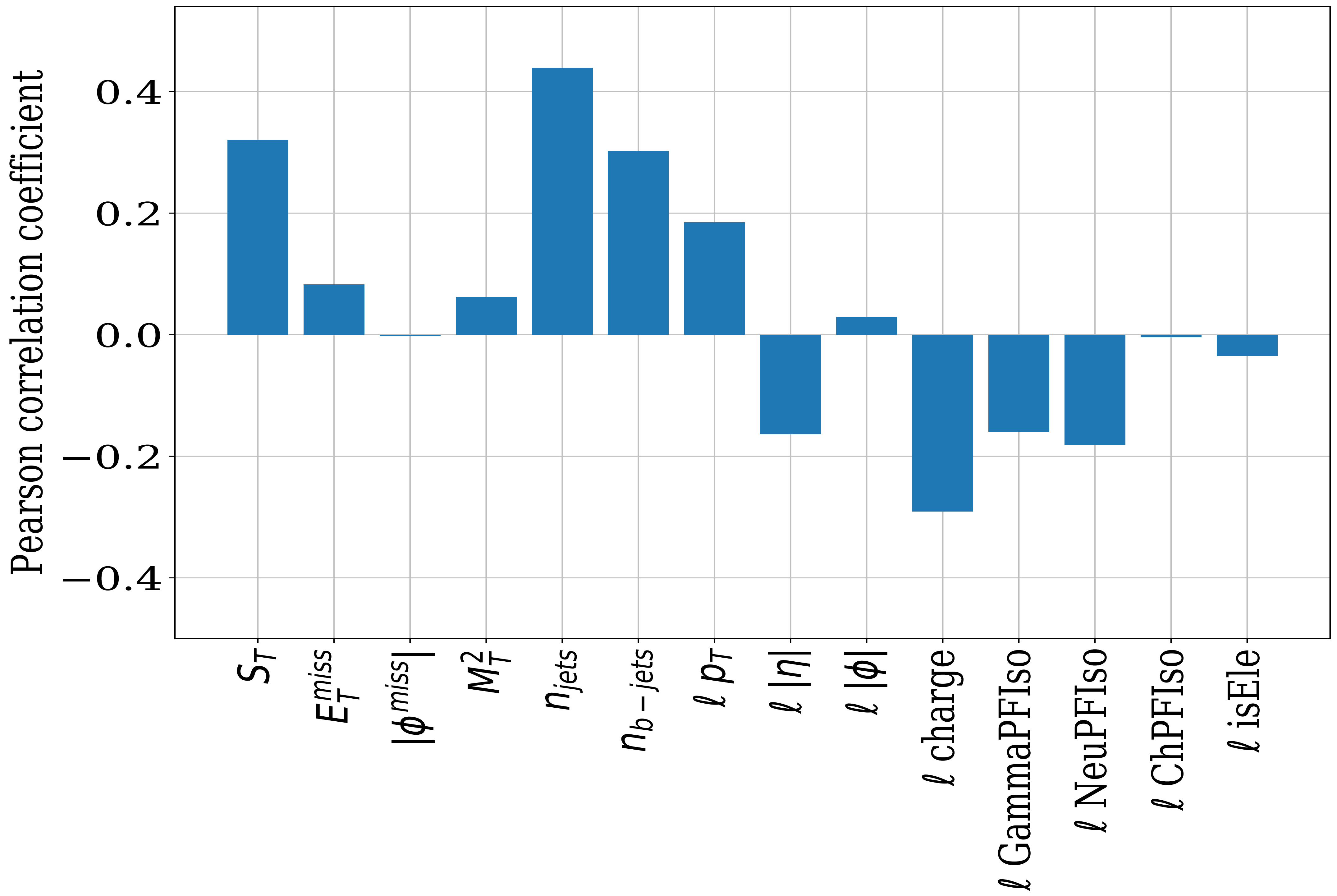
TOPCLASS: do we kill New Physics?

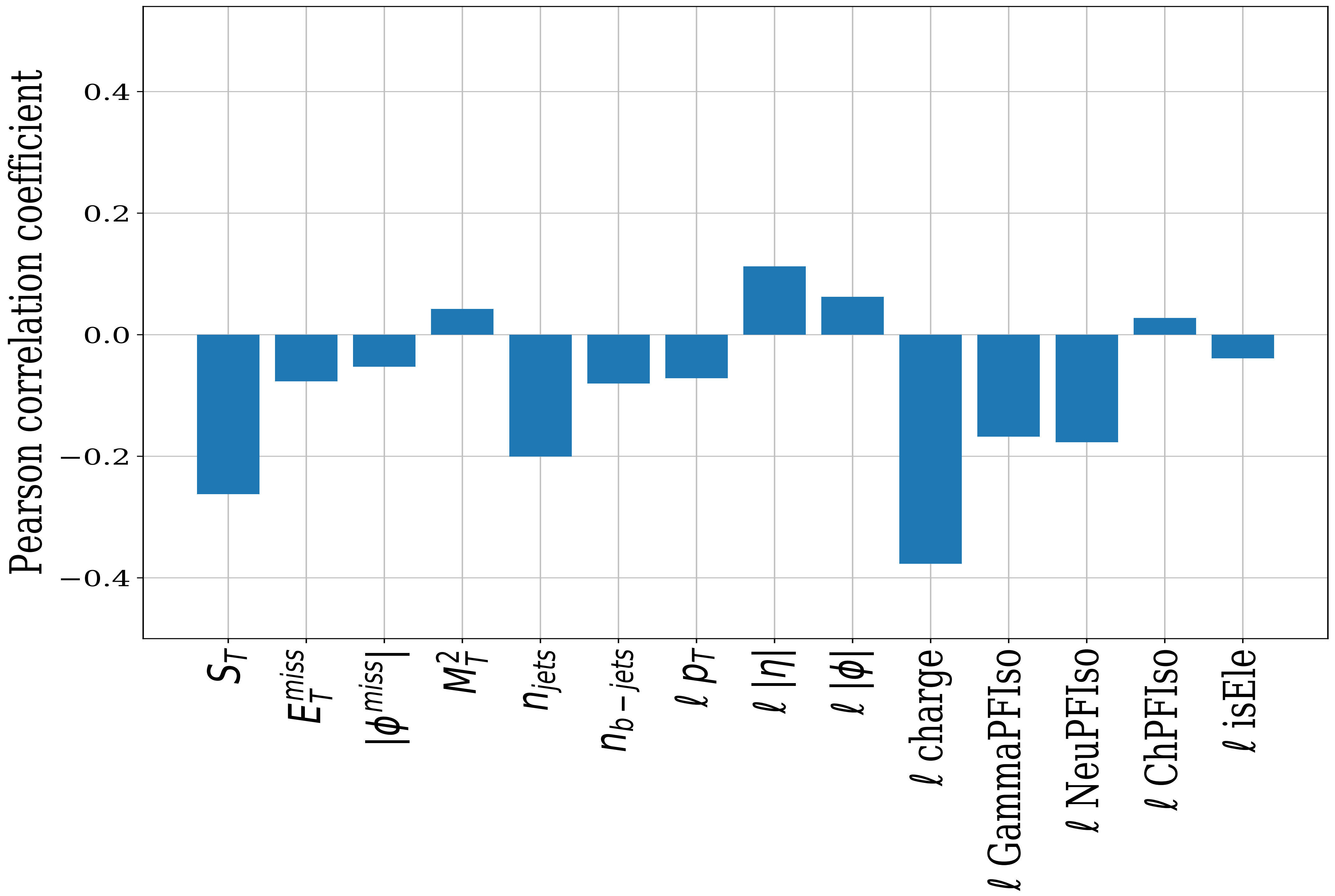


(d) W'

(e) Z'

Selection performances





The network is learning some physics…

- *tt events are more crowded that W events*
- *leptons in W and tt events are isolated from other particles*

European Research Council