Deep Learning and the challenges of High-Luminosity LHC

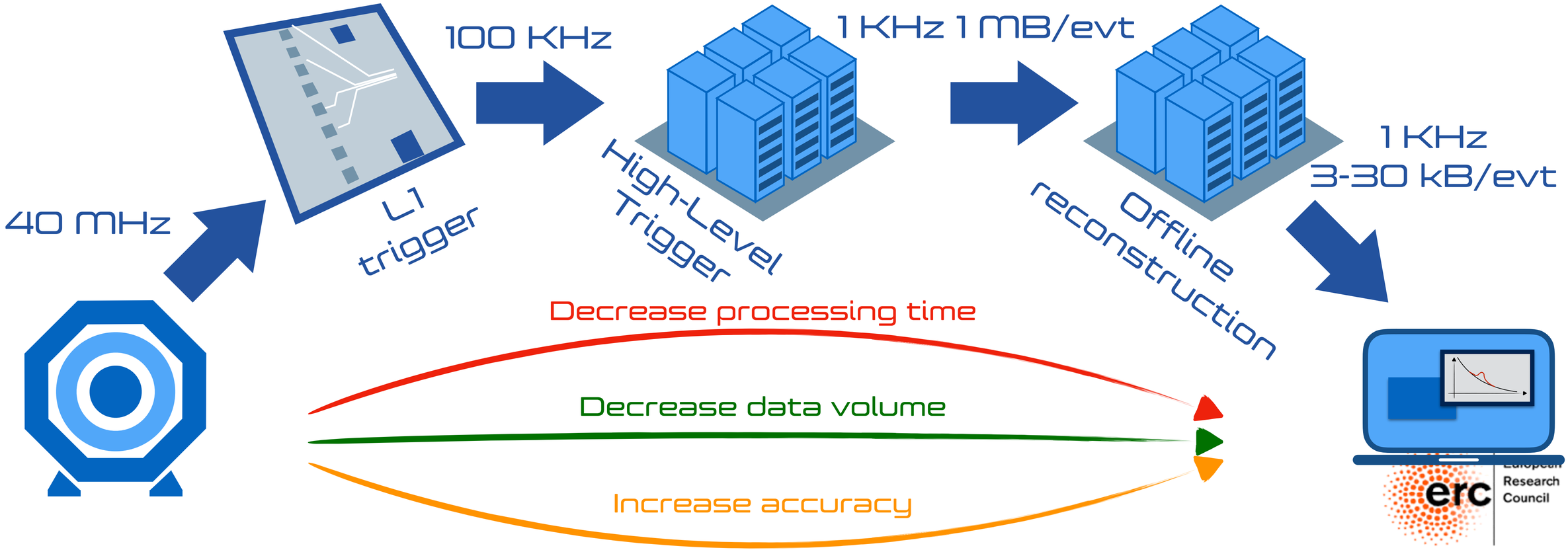
Maurizio Pierini

XII International Conference on Interconnections between Particle Physics and Cosmology, Aug. 20-24 Zurich

erc European Research Council

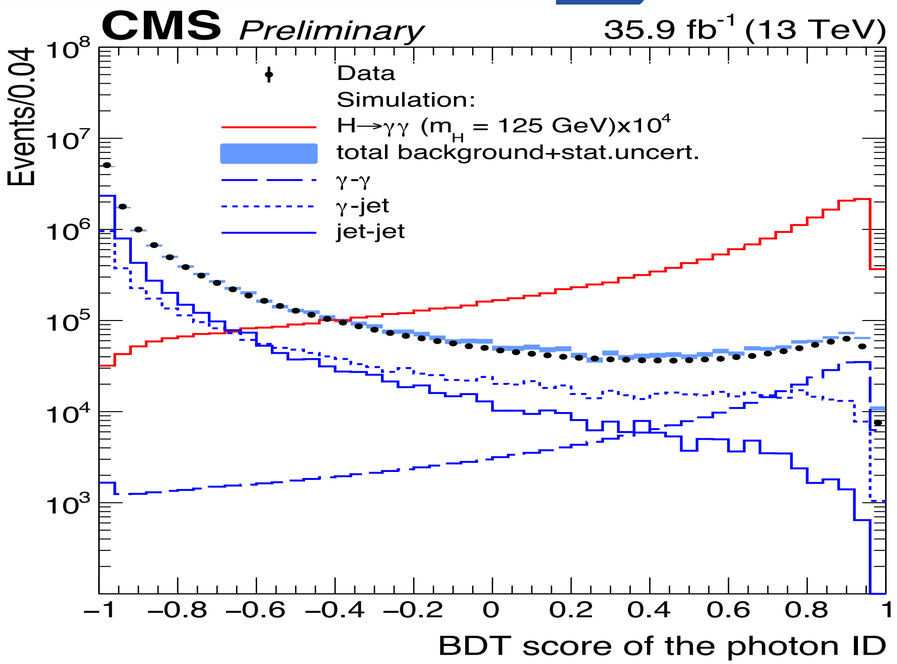
Data taking/processing @LHC

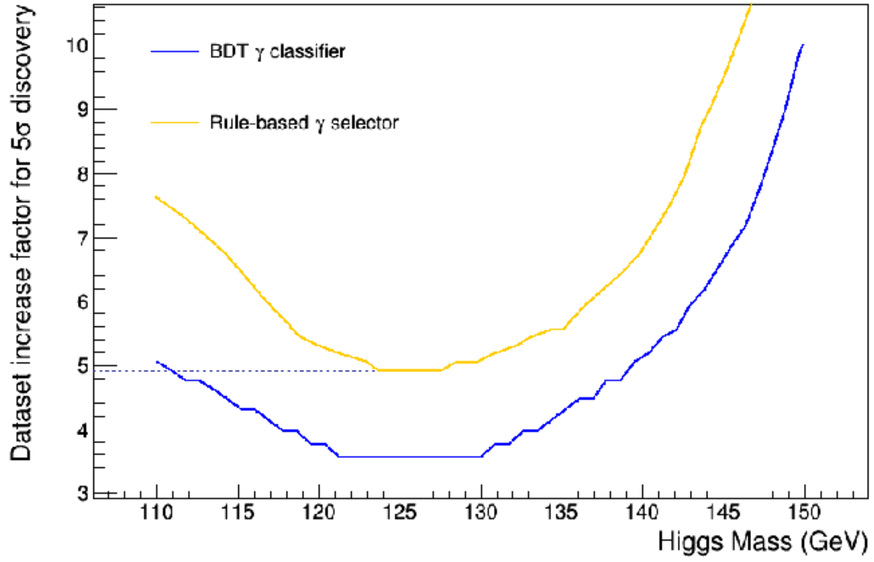
- *Process goes through 4 steps*
  - *L1 trigger: local, hardware based, on FPGA, @experiment site*
  - *HLT: local/global, software based, on CPU, @experiment site*
  - *Offline: global, software based, on CPU, @CERN T0*
  - *Analysis: user-specific applications running on the grid*



Workflows nowadays

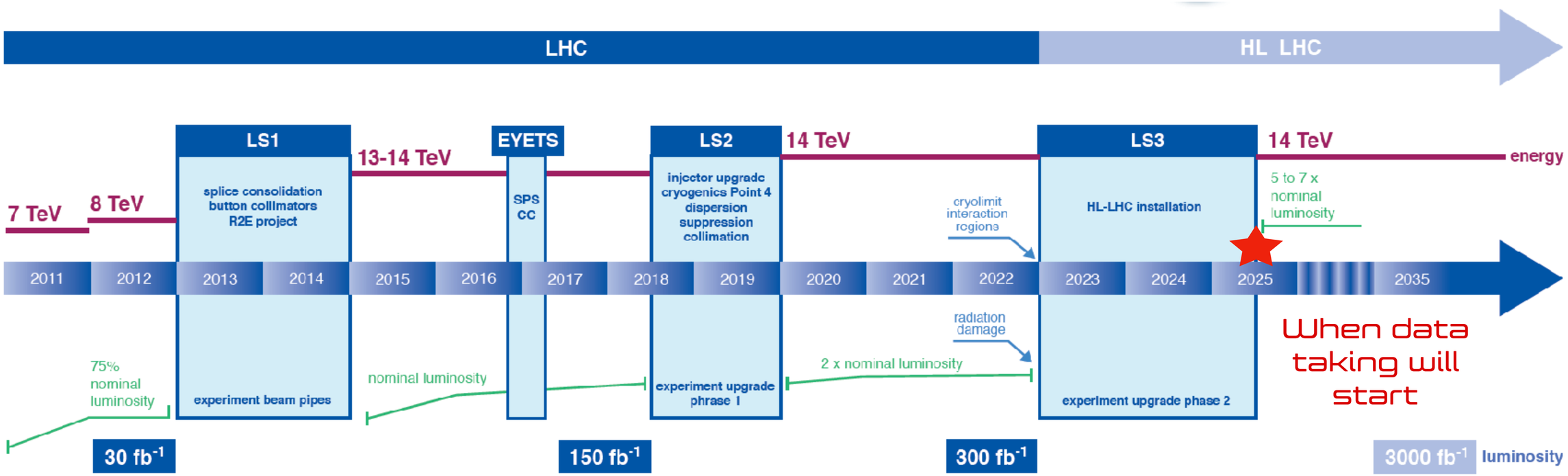
- Central processing is mainly based on rule-based algorithms
  - Kalman filter for tracking
  - Physics-motivated clustering (e.g., jet clustering algorithms)
  - Detector-tailored identification (e.g., expert features for particle identification
- Machine learning has already a role on it
  - many classifiers (mainly based on boosted decision trees) for particle identification, energy regression, etc
- Machine learning has a much bigger role in data analysis
  - signal/background discrimination based on expert features
  - notably, for Higgs boson discovery ∃





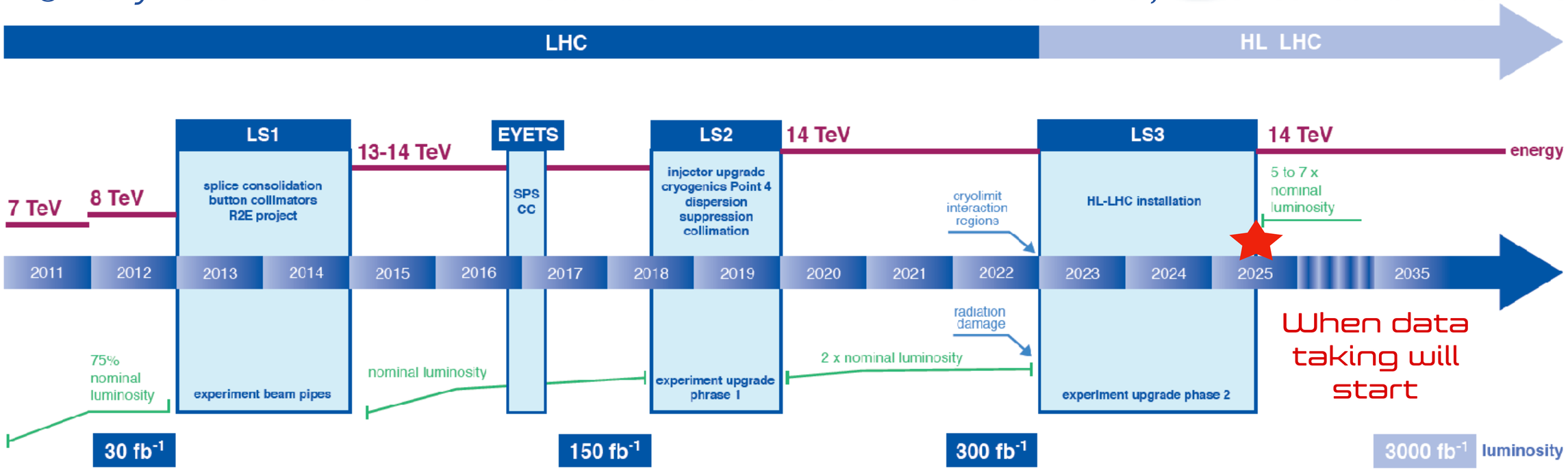
The challenge ahead

- *The LHC will enter its "High-Luminosity" phase in ~2025*
  - *Will deliver in in 10 years 3000 fb^{-1}*
  - *(in average) Each year, same dataset collected in 2008-2022*

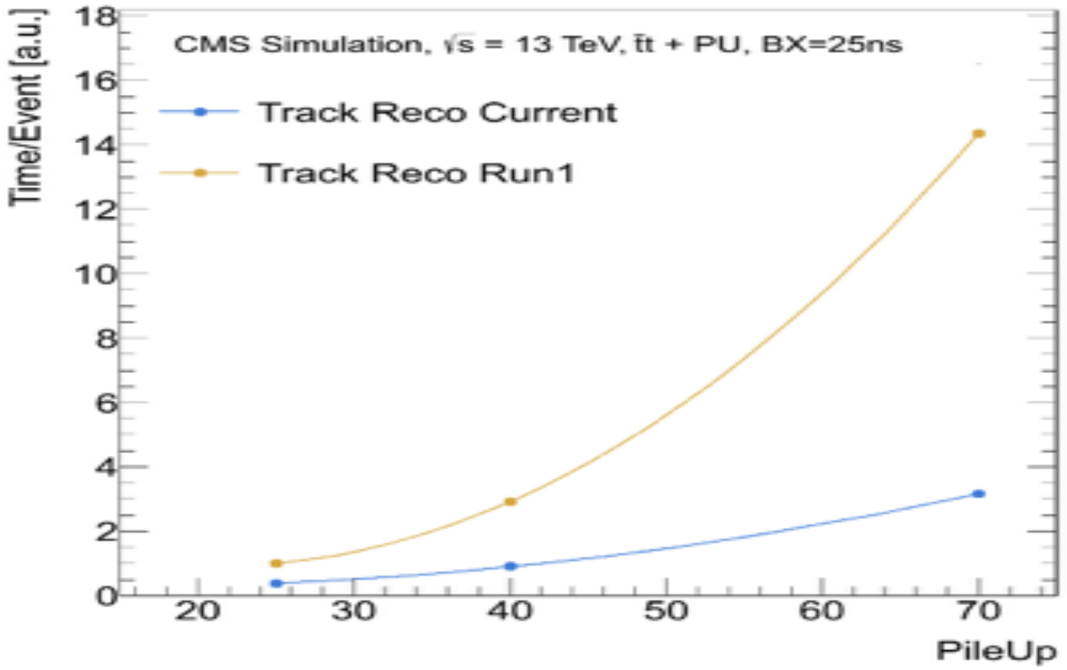


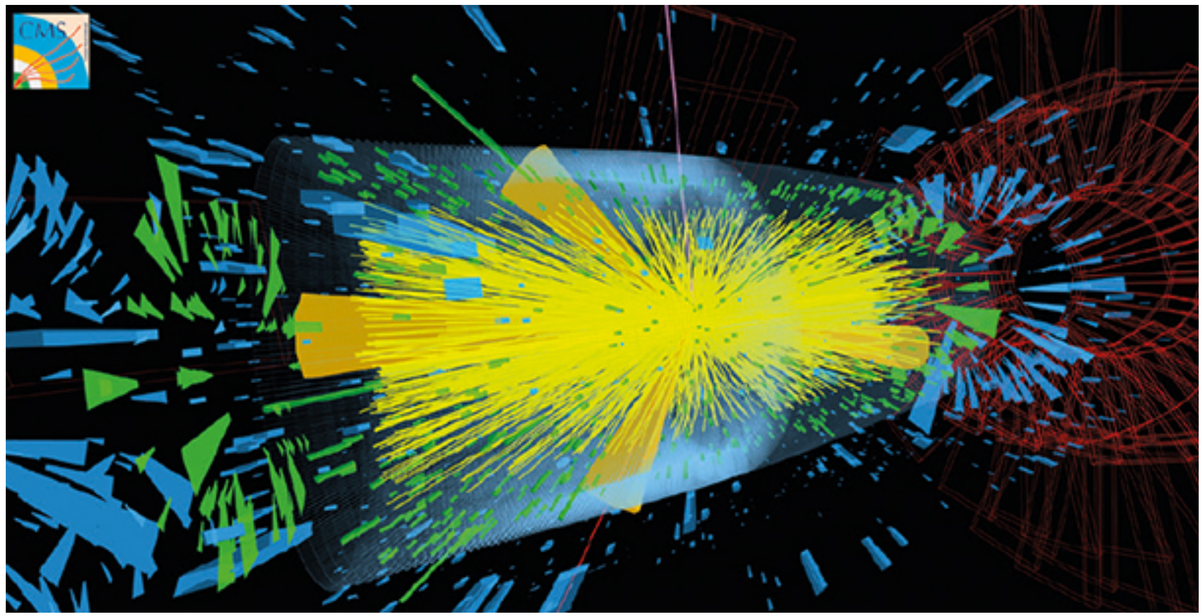
The challenge ahead

- *The price to pay will be in event complexity*
  - *(in average) 200 interactions at each bunch crossing (now <60)*
  - *Need to discard hits from non-interesting 199 collisions, focusing on the interesting one (the one firing trigger)*
  - *Many tasks have non-linear behavior vs # interactions, due to combinatorics*



Present vs Future





- ~40 collisions/event
- ~10 sec/event processing time
- Computing system designed (and pledge for) based on these conditions

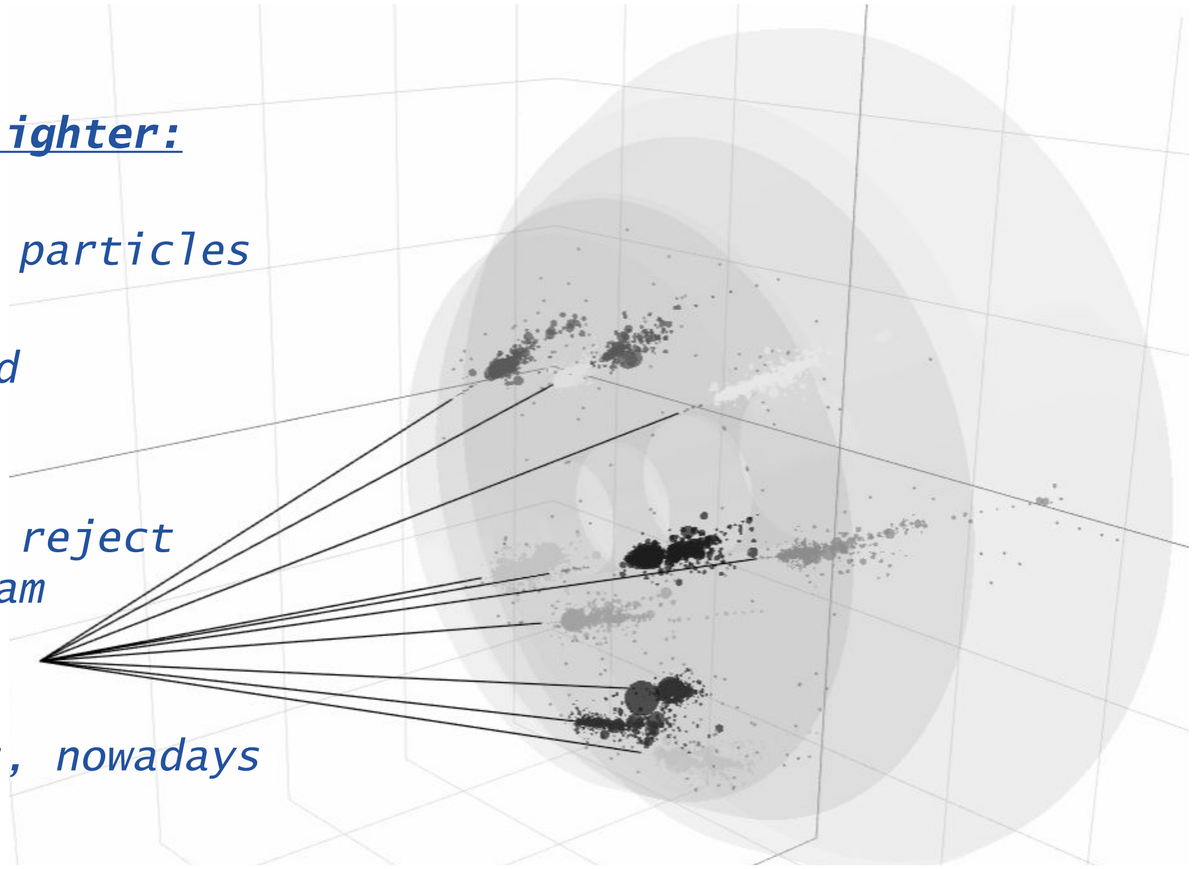
- ~200 collisions/event
- ~minute/event processing time(*)
- (at best)Same computing resources as today

(*)With nowadays software development

European Research Council

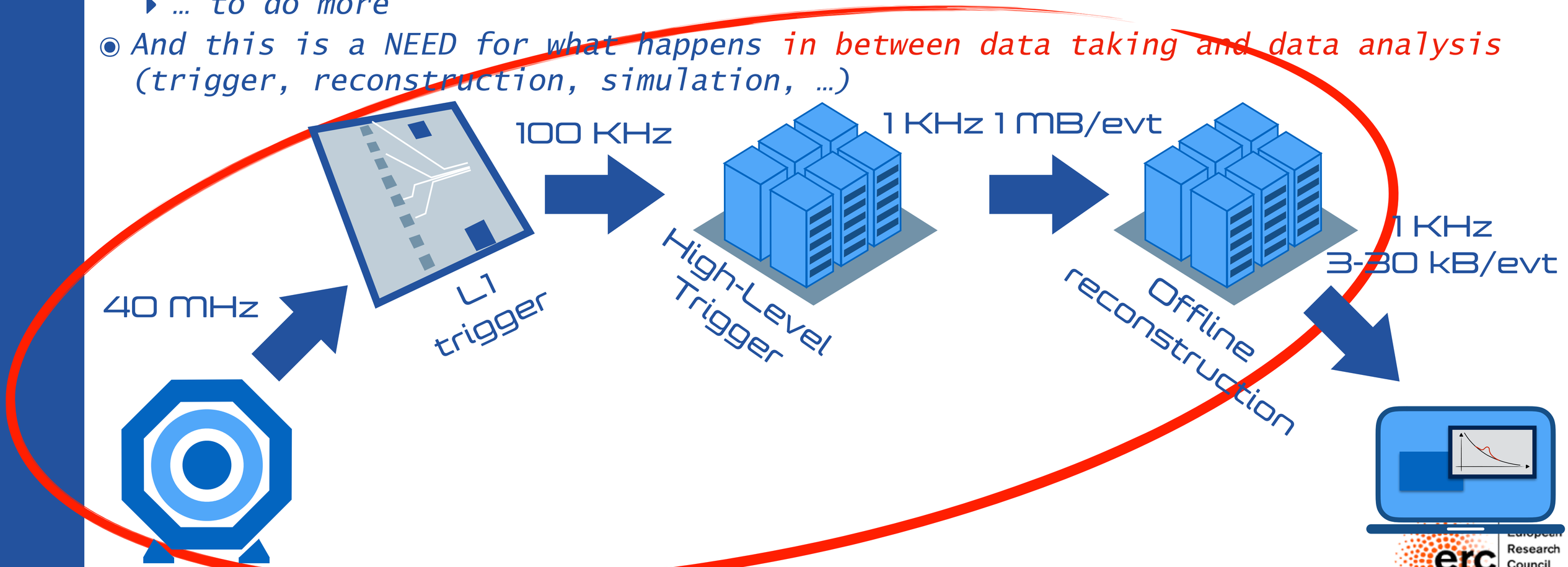
A heavier workflow

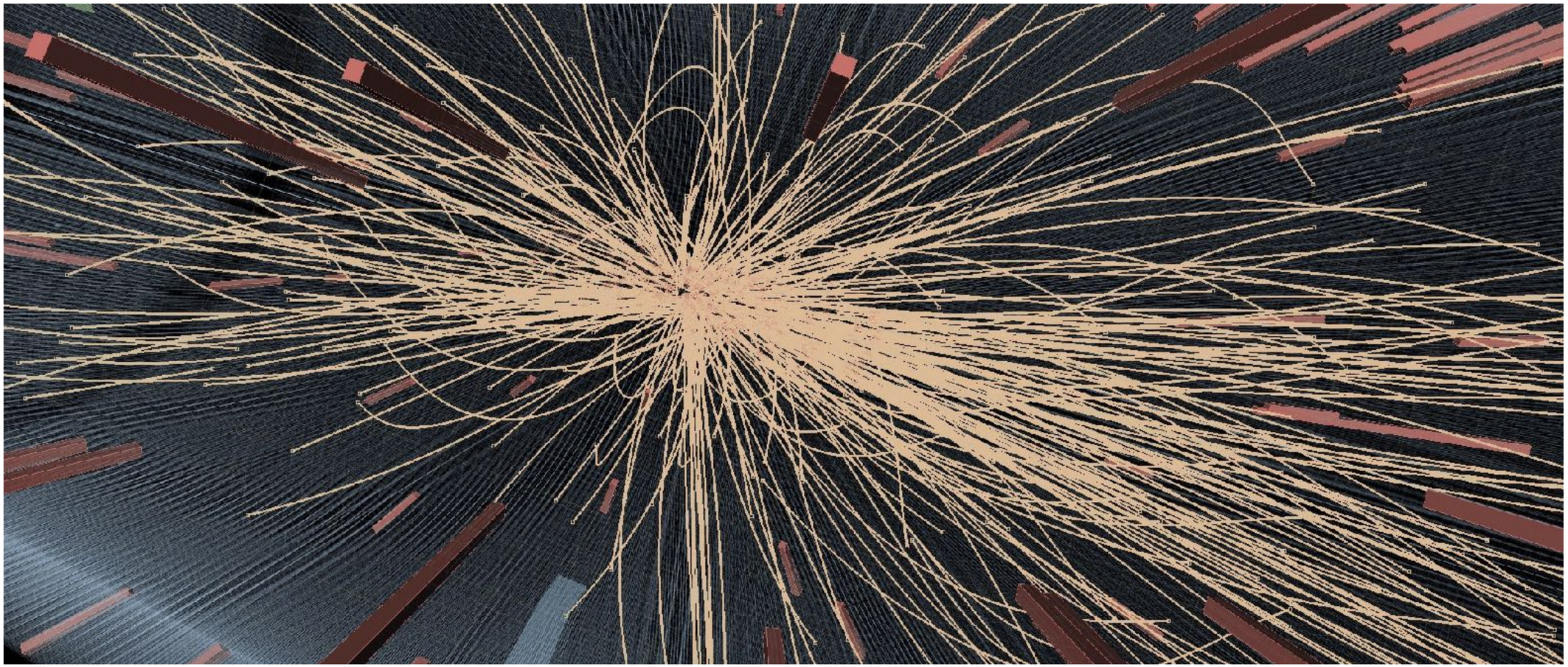
- *Flat budget vs. more needs = **current rule-based reconstruction algorithms will not be sustainable** with current detector*
- ***Future detectors will not making the task lighter:***
  - *more granularity to disentangle incoming particles*
  - *more detector components (e.g., dedicated detectors for tracking @L1)*
  - *4D reconstruction (position and time) to reject particles from previous and following beam crossing*
- *With heavier tasks and <= current resources, nowadays adopted solution will not scale*
- ***Modern Machine Learning might be the way out***



What DL could do for us

- *The solution to the HL-LHC problem: modern Machine Learning …*
  - *… to be faster*
  - *… to do better*
  - *… to do more*
- *And this is a NEED for what happens in between data taking and data analysis (trigger, reconstruction, simulation, …)*

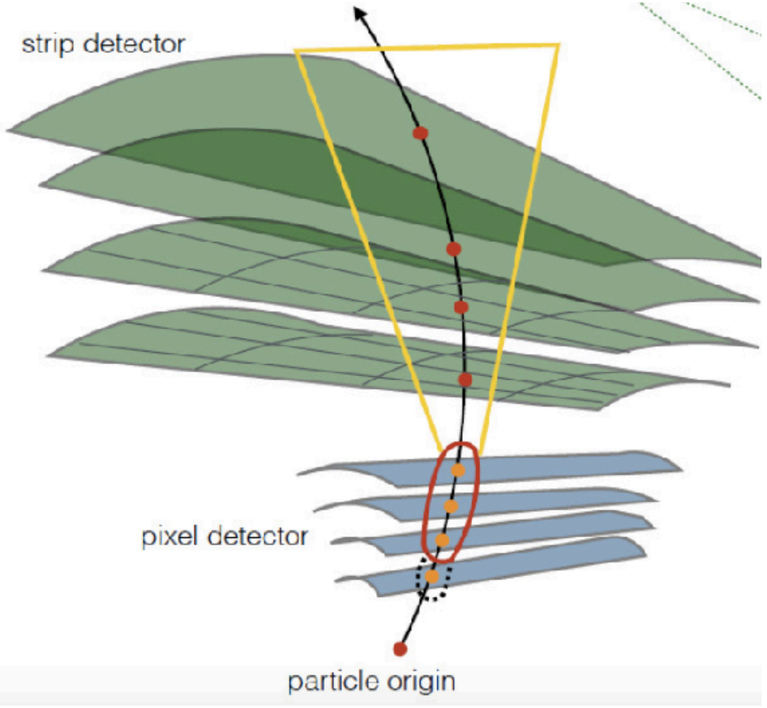


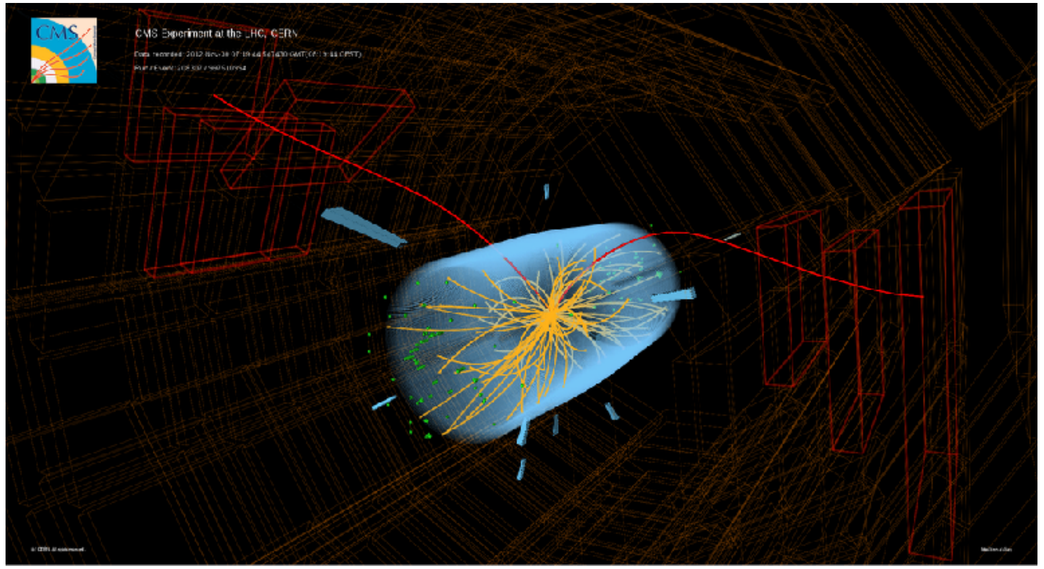


Example 1: speeding up tracking with computing vision

Particle tracking

- Connecting the dots, with…
  - Thousands of particles, each leaving energy depositions on O(10) layers of detectors
  - Particles bend due to magnetic field
  - Particles interact with the material / decay in flight
  - Each energy deposition isn't necessarily in a single pixel (trajectories cross)
- This is the most expensive part of our reconstruction software, despite our push for improvements





Particle tracking

Several Times

Hits Preparation

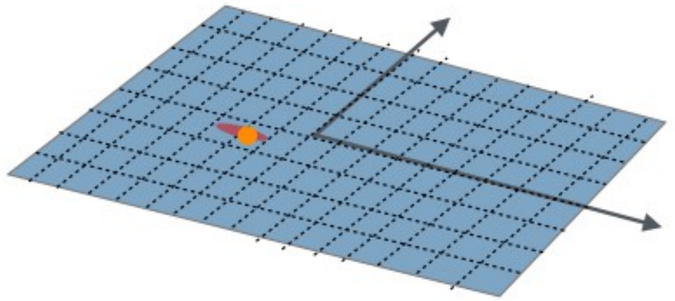
Seeding

Pattern Recognition

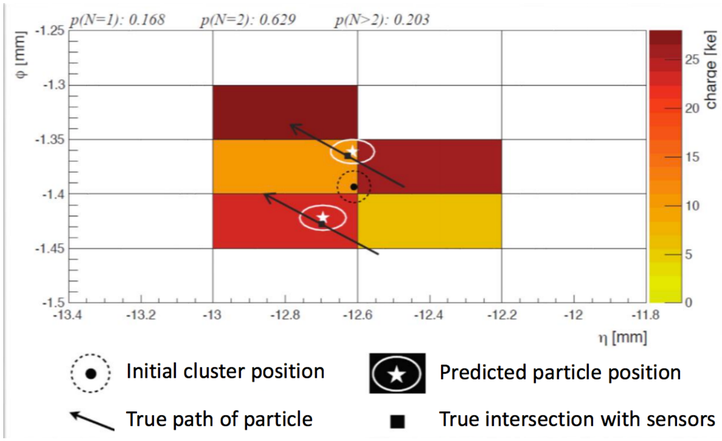
Track Fitting

Track cleaning

Local clustering of energy deposits into a Hit



ML already in use to reject fakes or to split overlapping clusters



European Research Council

Particle tracking

Several Times

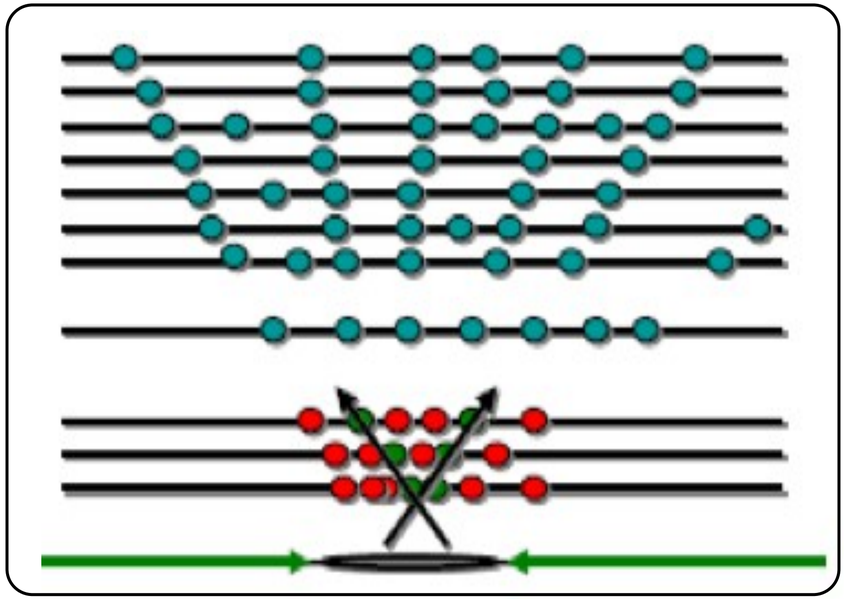
Hits Preparation

Seeding

Pattern Recognition

Track Fitting

Track cleaning



3D clustering of hits from inner layers into track segments
Fixes combinatoric of the problem -> CPU needs of following steps

European Research Council

Particle tracking

Several Times

Hits Preparation

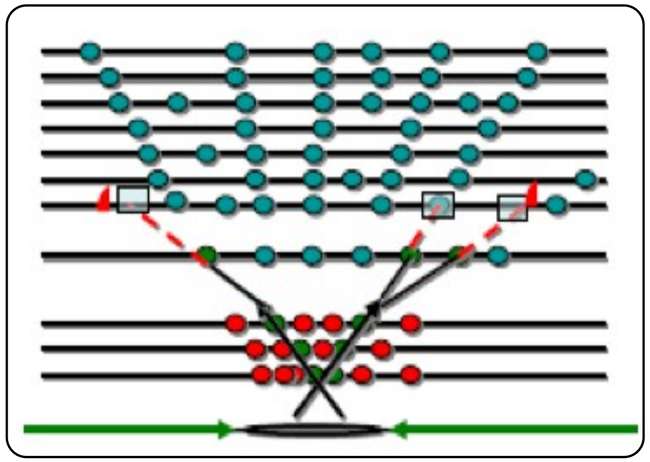
Seeding

Pattern Recognition

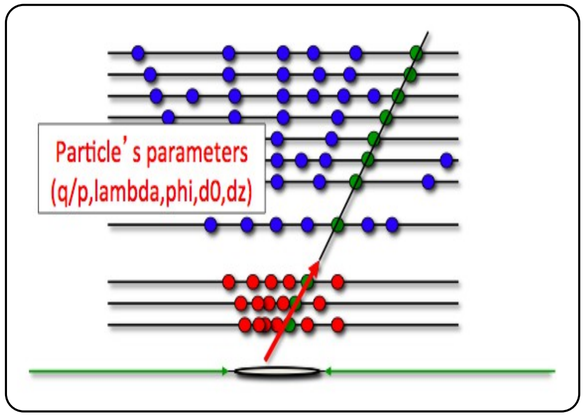
Track Fitting

Track cleaning

Kalman Filter to predict hit position on the next layer
Deal with physics effects (interaction between particle and material)



Fit the track through points, using Kalman-Filter weights, uncertainty on detector alignment, etc



European Research Council

Particle tracking

Several Times

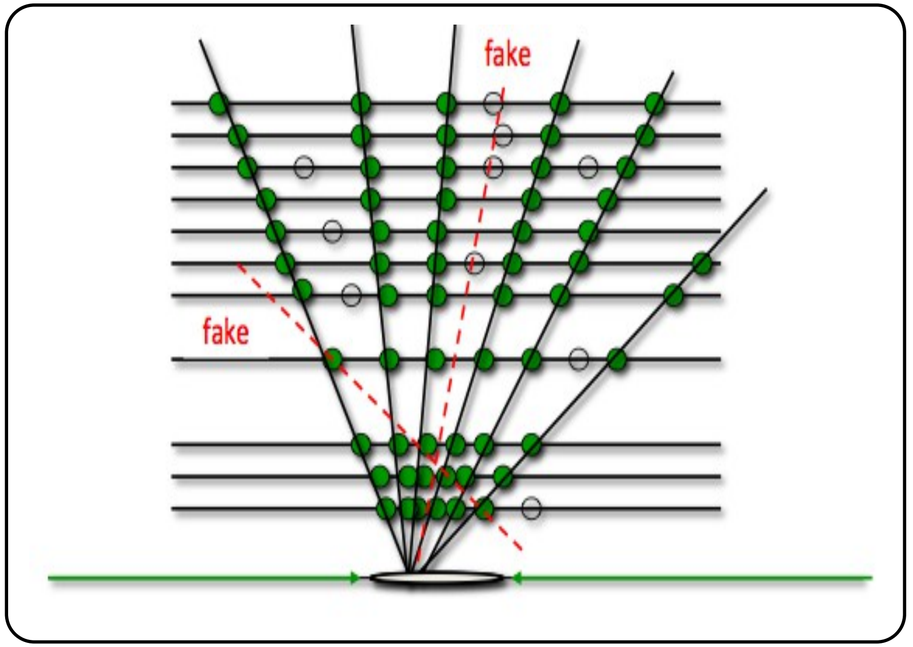
Hits Preparation

Seeding

Pattern Recognition

Track Fitting

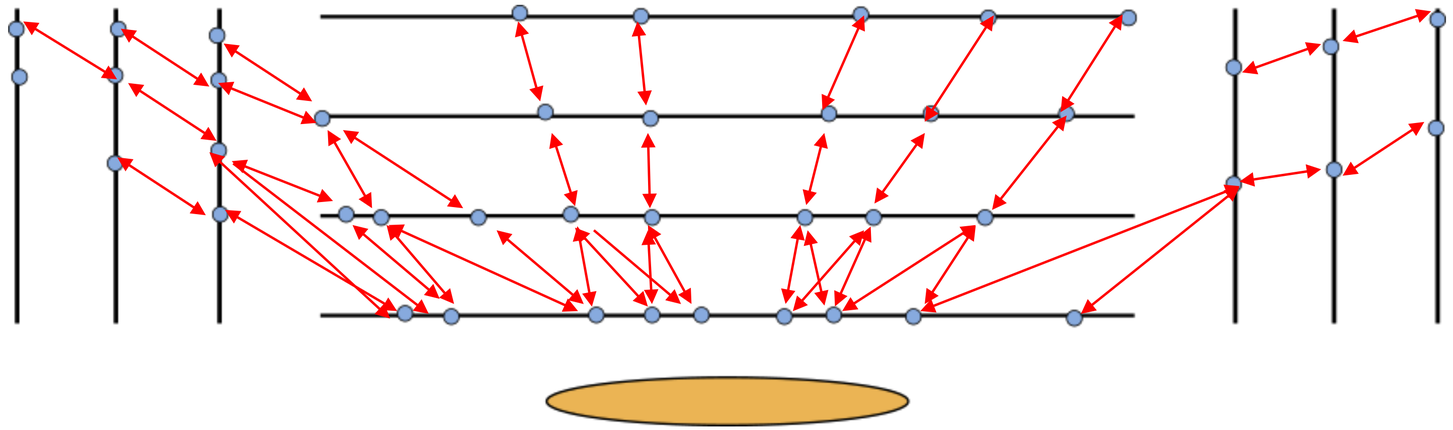
Track cleaning



Reject fake tracks using classification or ranking methods
(ML already exploited here, e.g. with BDTs in CMS)

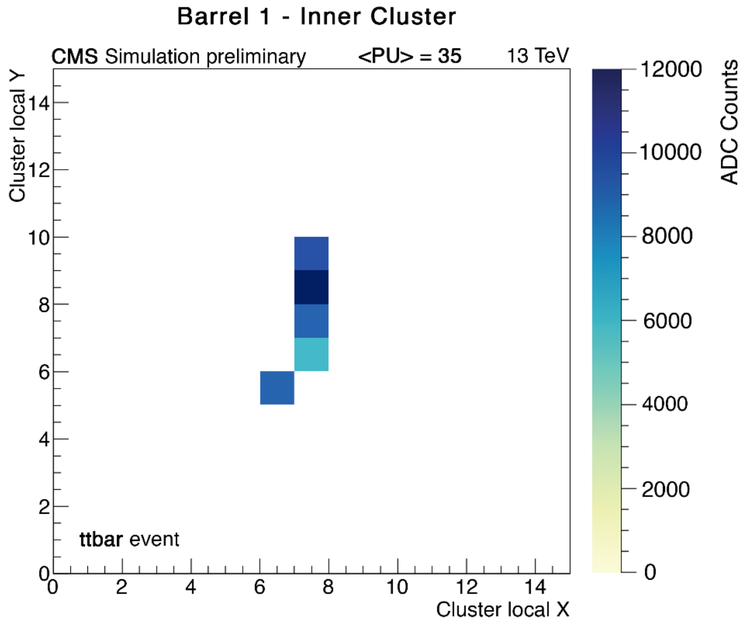
Track Seeds

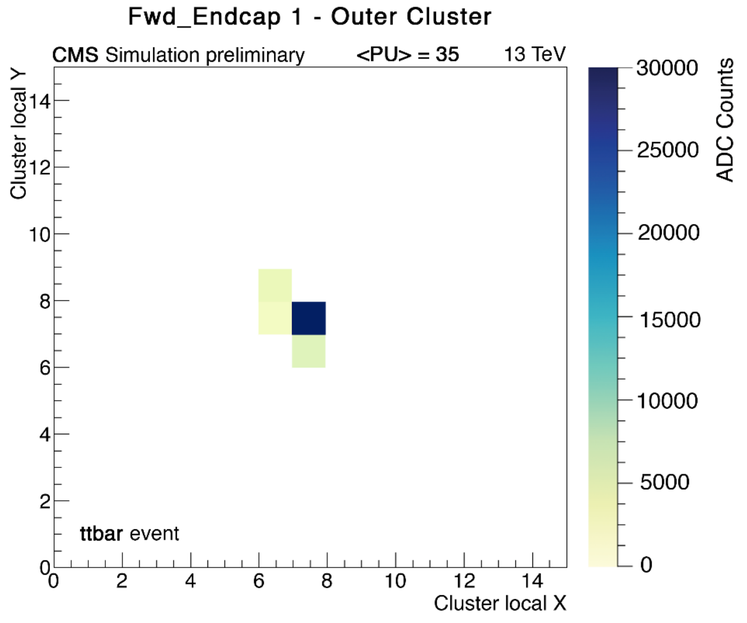
- *Finding seeds is not the most CPU intensive aspect*
- *But its outcome dictates the combinatoric of the following steps*
  - *one can speed up tracking by reducing the number of fake seeds*
- *We tried to solve this problem using Deep Learning*



Seeds as images

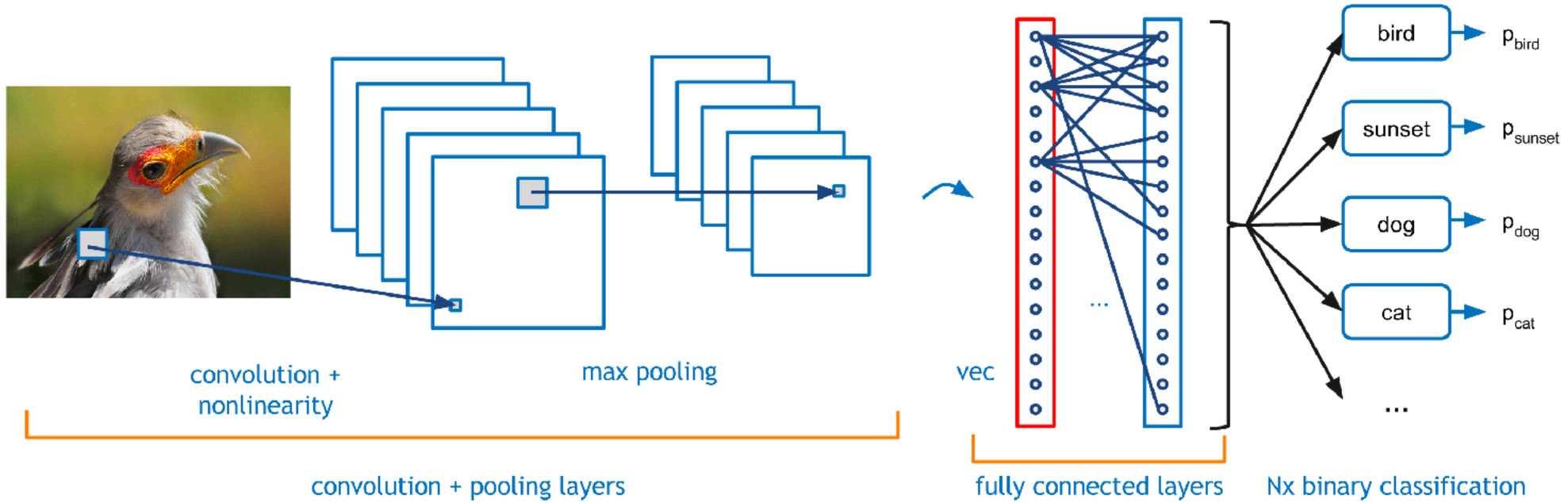
- *The detector sees the charge deposited by the crossing particle: a hit*
- *A hit is a window of sensors (16x16 here) with its deposited charge. This can be seen as a sparse digital image.*
- *Given two images, one can train a network to decide if a pair of hits is a good or bad match*





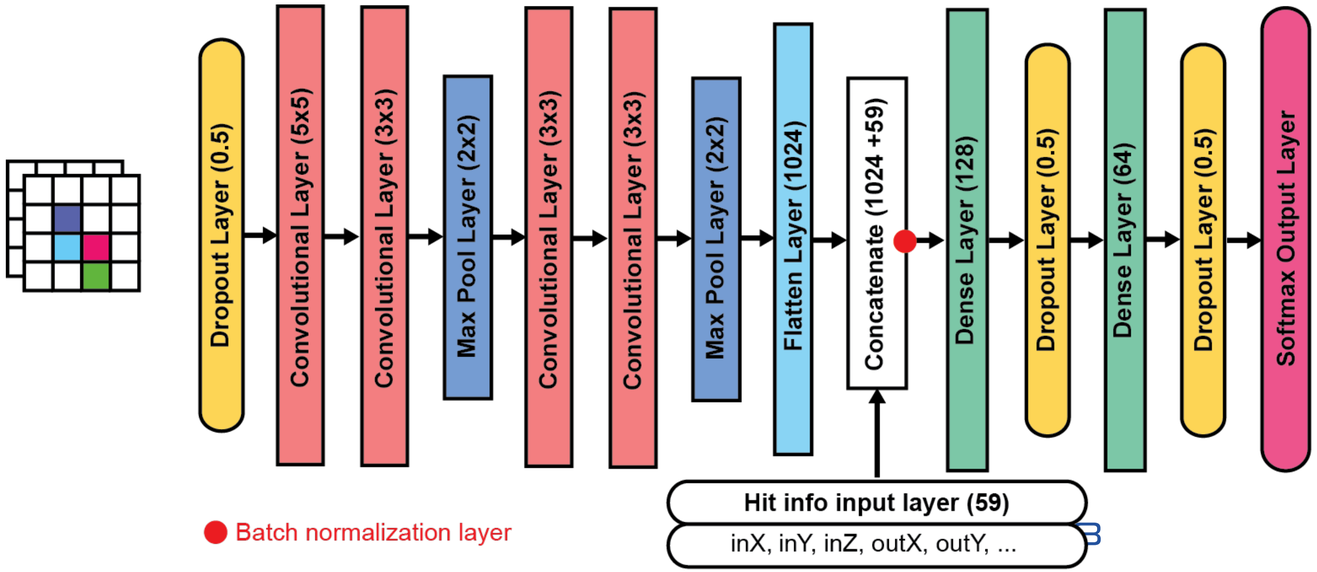
Convolutional Layer

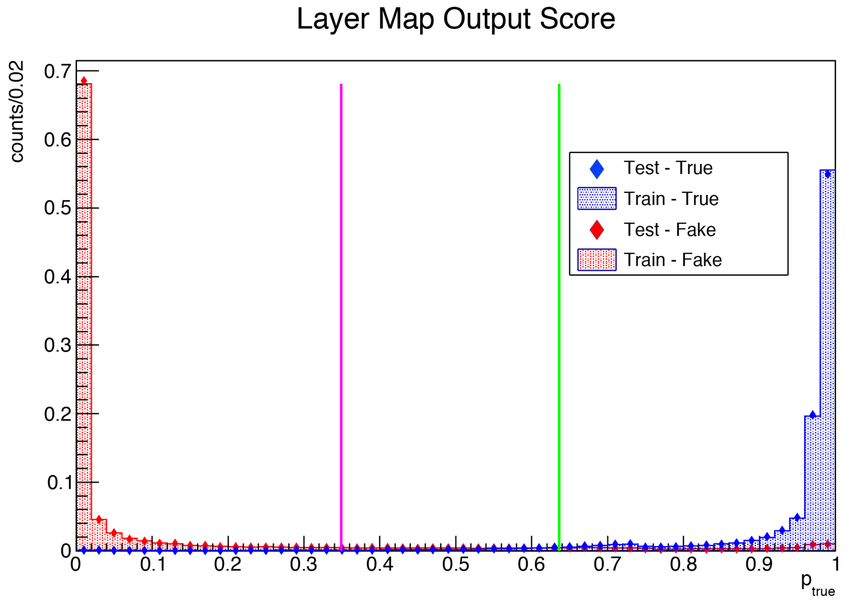
- *Convolutional neural networks are a special architecture suitable for image processing & computing vision*
  - *defined by filters scanning images, keeping near-neighbour pixel information with translation invariance*
  - *through the filters, the network process the raw image and produces high-level features, then used to accomplish the task (e.g., classification)*



PixelSeed ConvNN

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





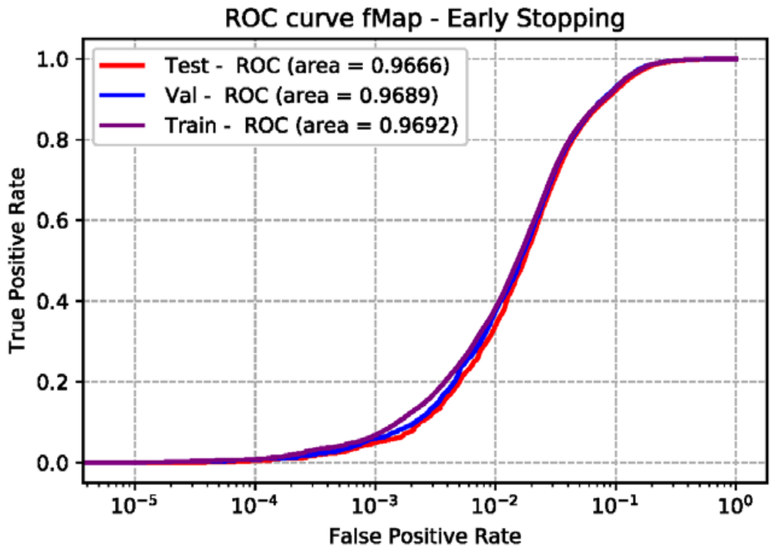
Results

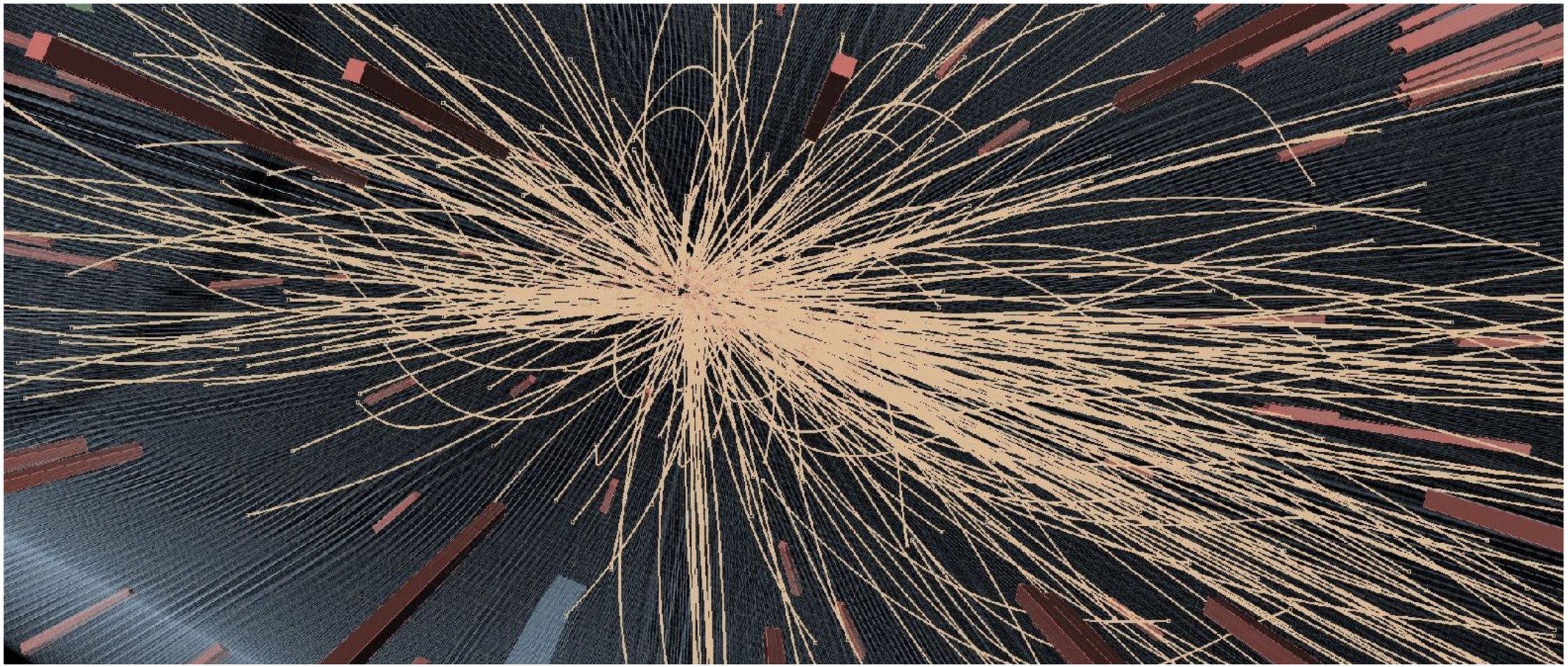
- *One could remove almost all the fake seeds. But the price to pay would be loosing 1/2 of the good seeds*
- *One could keep 99% of the good seeds, but only 1/3 of the fake seeds would be removed*
- ***One could reduce the fake rate by one order of magnitude with a few % loss in efficiency***

This is where the user expertise matters. An optimal algorithm could still be used very badly!

Efficiency (tpr) @ fake rejection

```
tpr @ rej 50%: 0.998996700259
tpr @ rej 75%: 0.990524391331
tpr @ rej 90%: 0.922210826719
tpr @ rej 99%: 0.338669401587
```

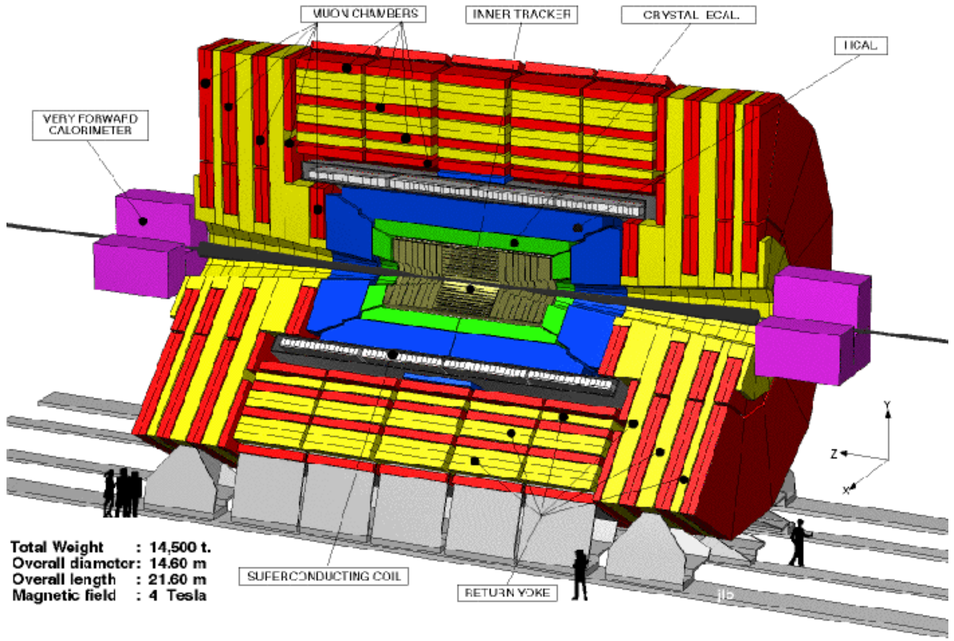


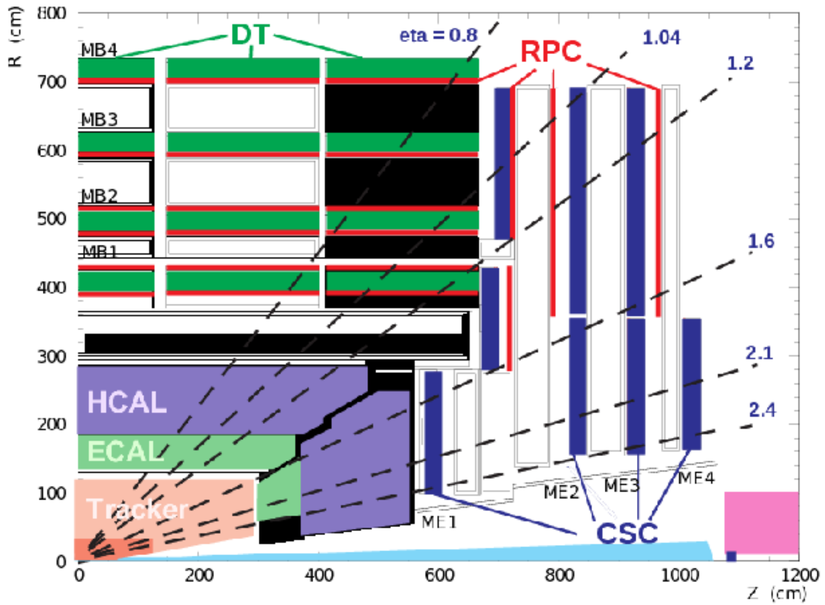


Example 2: event identification as natural-language processing

HEP data are irregular images

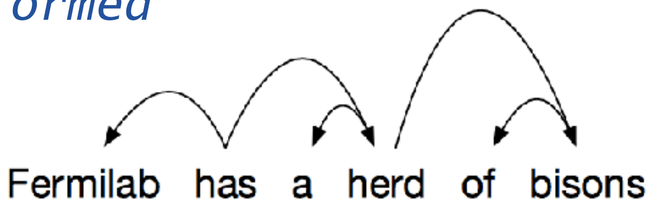
- *We saw you how ConvNN could solve some of our problems*
- *But ConvNN comes with assumption*
  - *data as a regular array or sensors*
- *Not all HEP data could be approximated like that*
  - *irregular detector geometry*
  - *tricky regions of overlap*
  - *…*

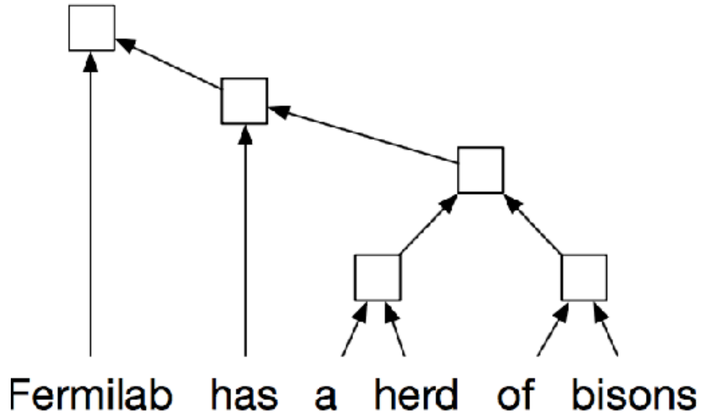


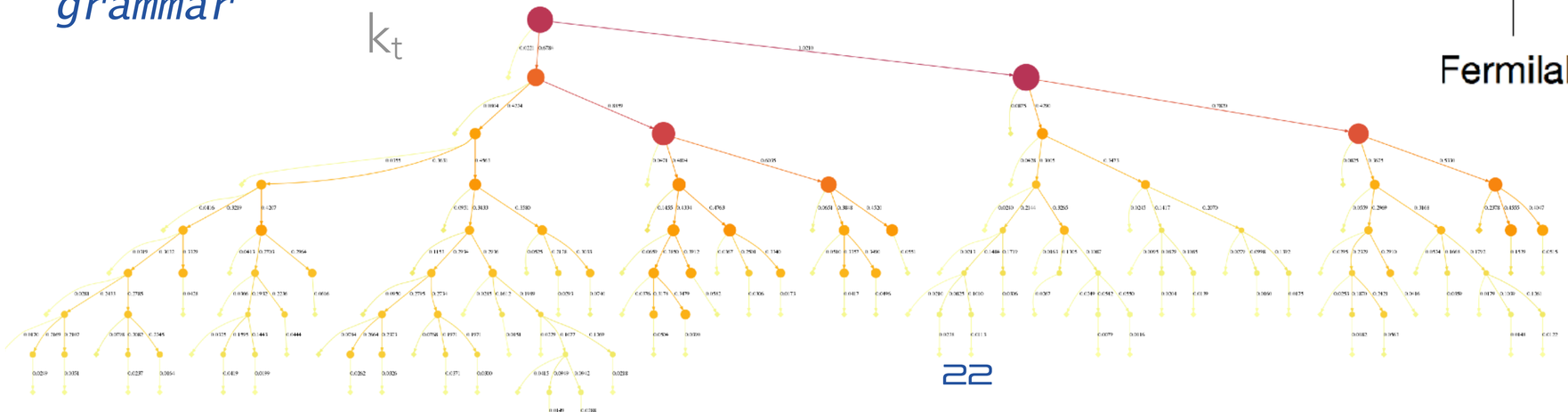


Particles & language processing

- *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*
- *For each particle we have certain features (energy, momentum, direction, point of origin, charge, kind of particle, etc.)*
- *We could see a jet as an ensemble of particles ordered in space by physics rules (QCD)…*
- *… like a sentence is made of words ordered by a grammar*

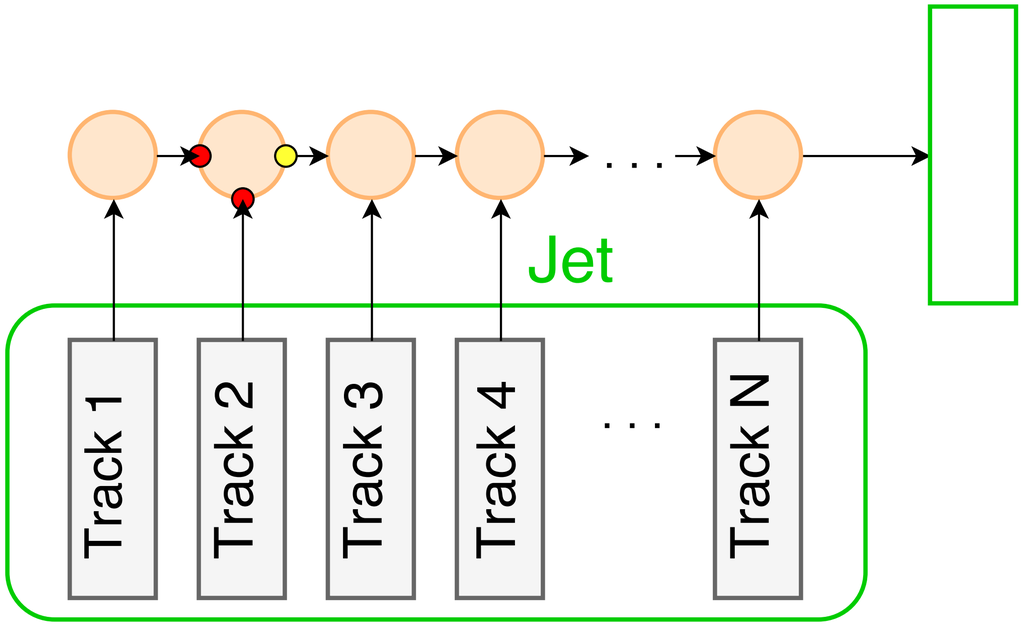






Recursive Neural Networks

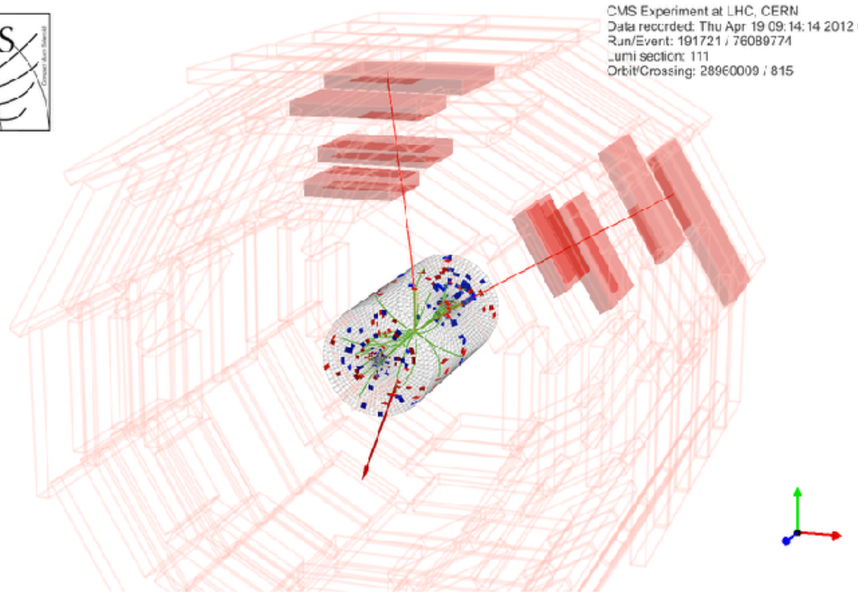
- *A network architecture suitable to process an ordered sequence of inputs*
  - *words in text processing*
  - *a time series*
  - *…*
- *Complex gate mechanisms mimic memory, storing outputs of previous iterations*
- *Processing of the i_{th} input depends on $(i-1)_{th}$ output (and through it on the full sequence)*

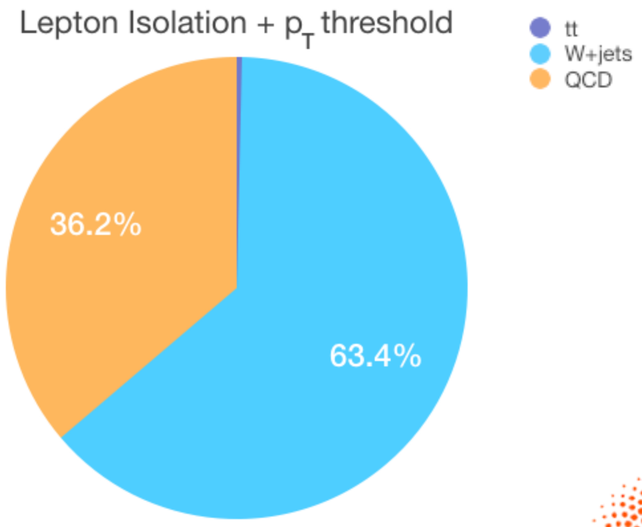


A Topology Classifier

A typical example: leptonic triggers

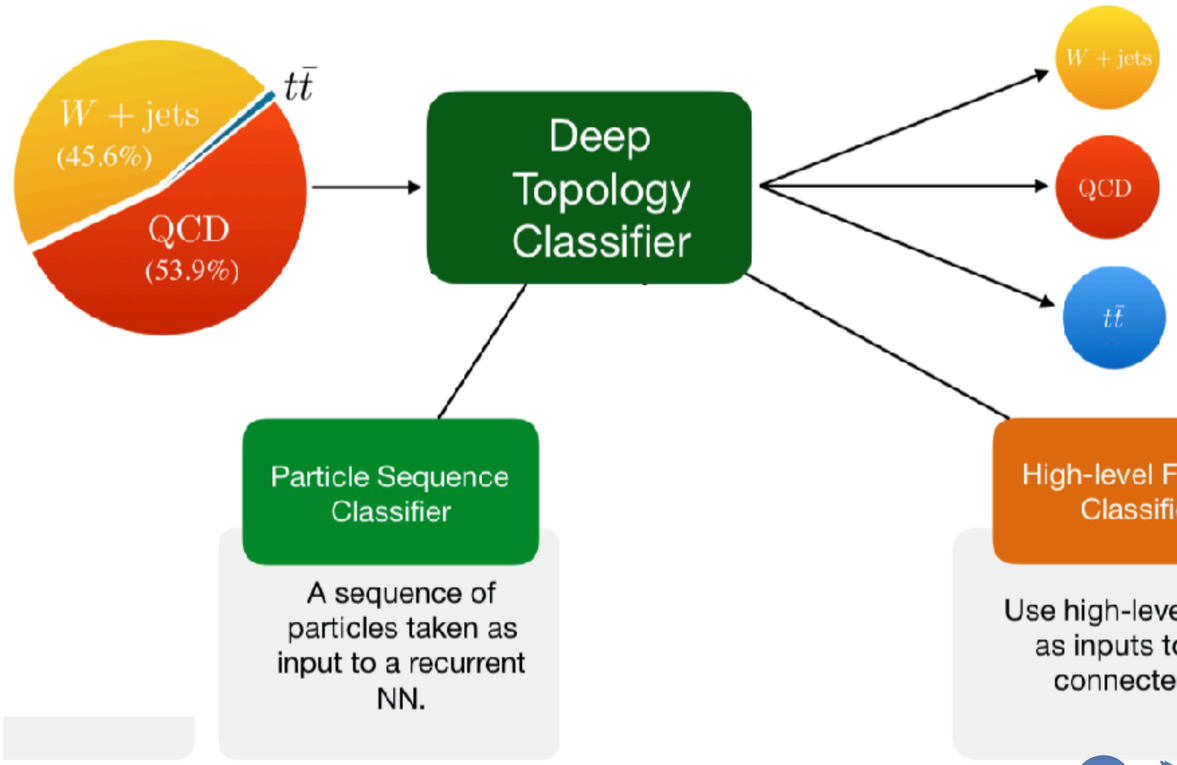
- *at the LHC, producing an isolated electron or muon is very rare. Typical smoking gun that something interesting happened (Z,W,top,H production)-> **TAKE THEM!***
- *Triggers like those are very central to ATLAS/CMS physics*
- ***The sample selected is enriched in interesting events, but still contaminated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



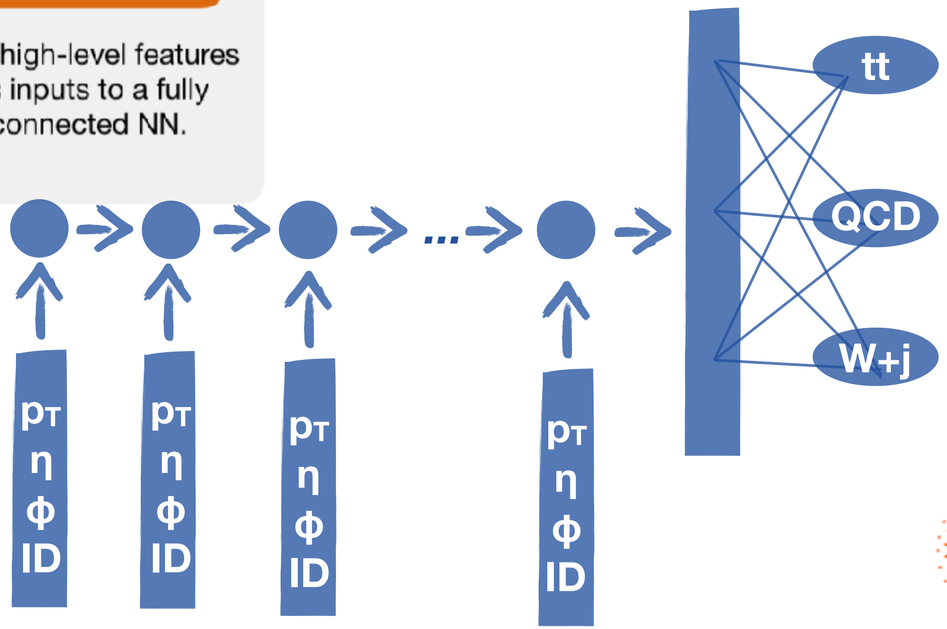


See contribution to NIPS workshop

A Topology Classifier



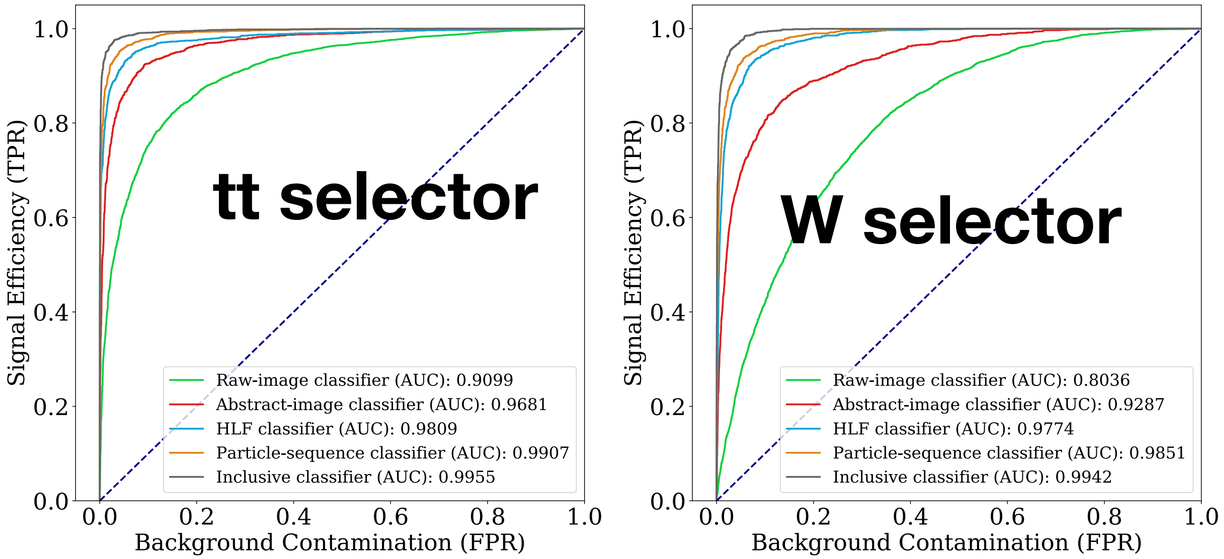
Recurrent nets on the list of particles



https://arxiv.org/abs/1807.00083

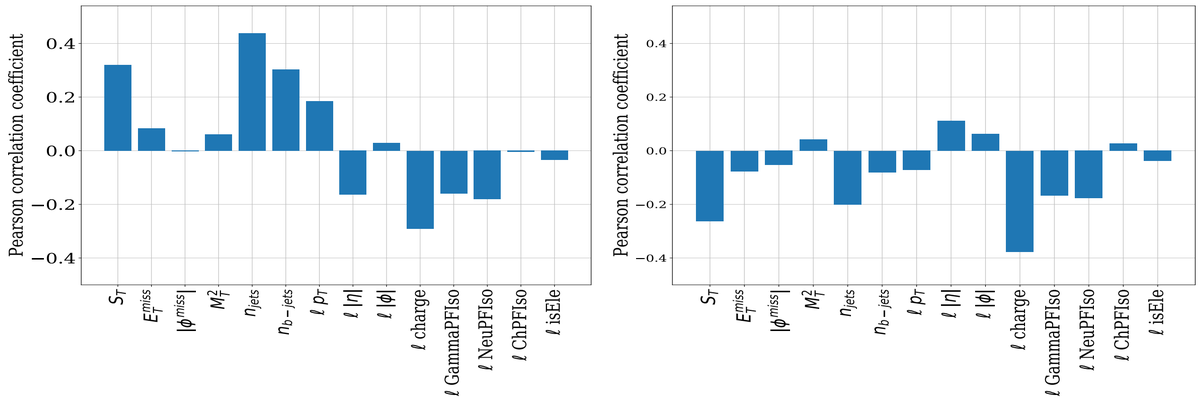
Selection performances

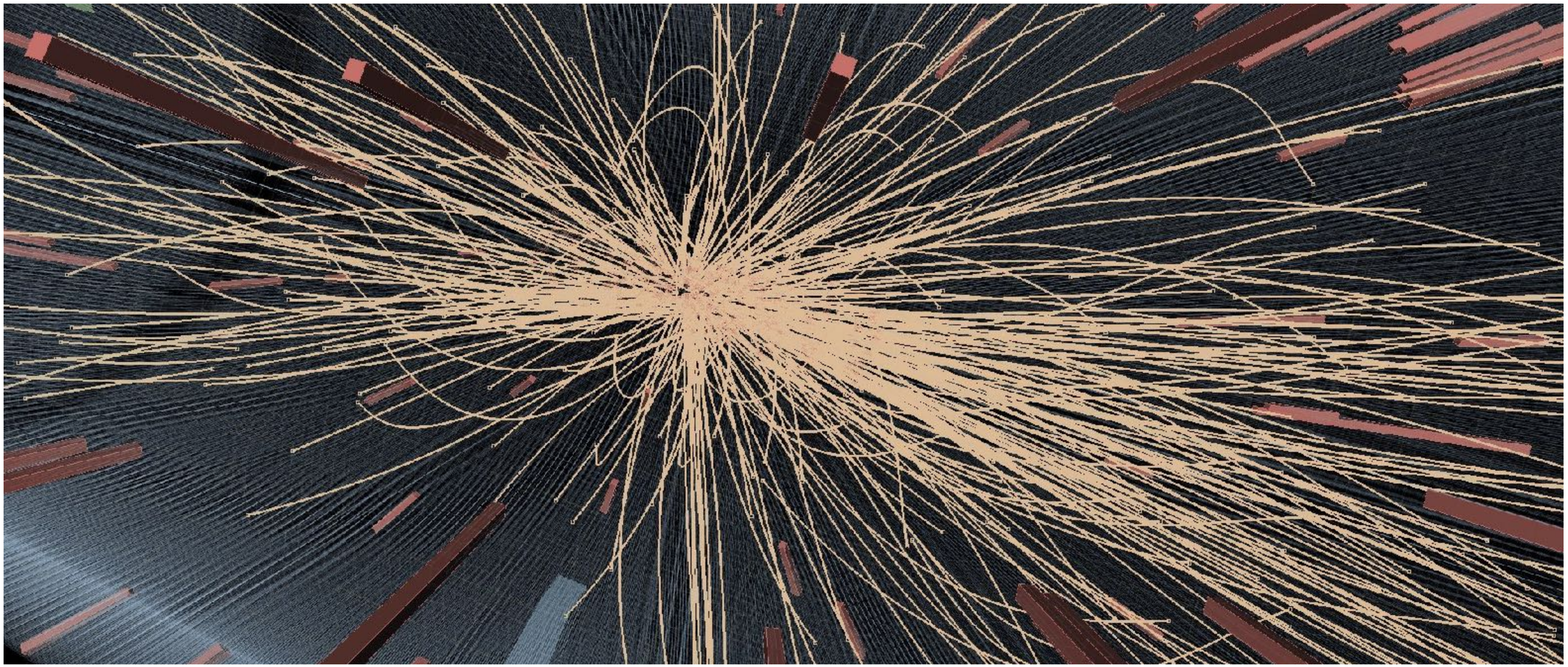
- *The classifier accomplish the tasks with good precision*
- *The network learns some physics (as shown by the correlation of the RNN output with physics motivated quantities)*
- *The use of expert knowledge of the problem (through the use of physics motivated quantities as extra inputs) improves the result*
- *Can select 99% of the top events and reduce the fraction of written events by a factor ~ 7*



(a) $t\bar{t}$ selector (b) W selector

Figure 5: ROC curves for the $t\bar{t}$ (left) and W (right) selectors described in the paper.



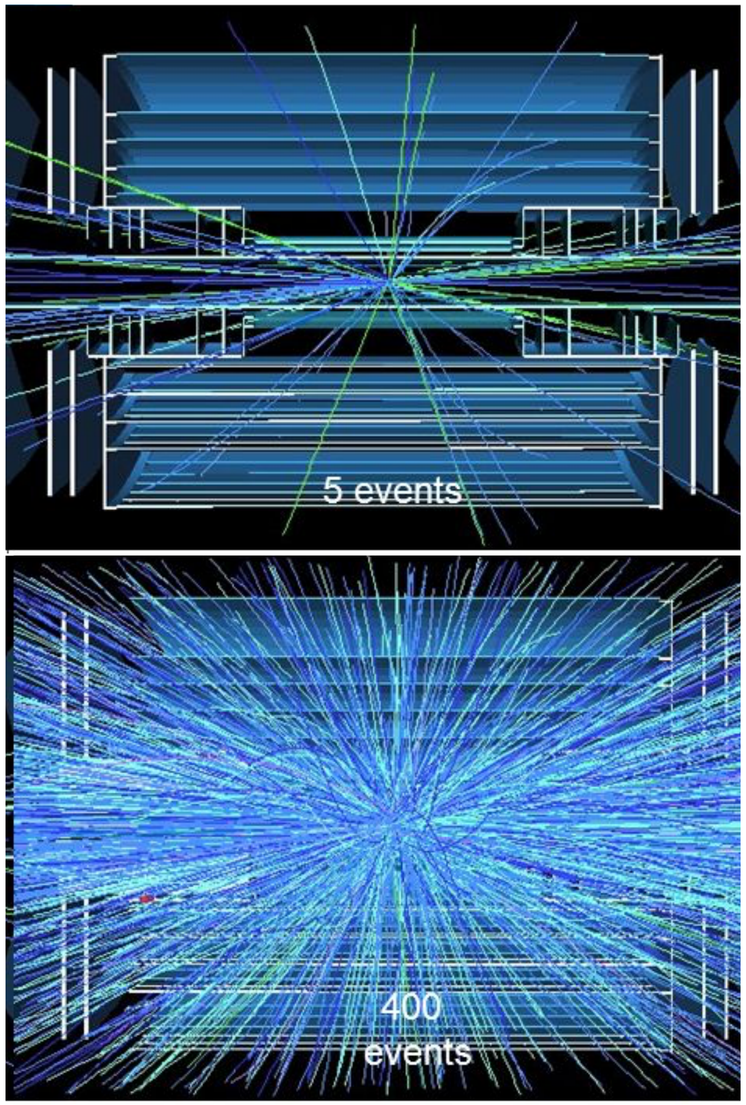


Example 3: event simulation with adversarial networks

European Research Council

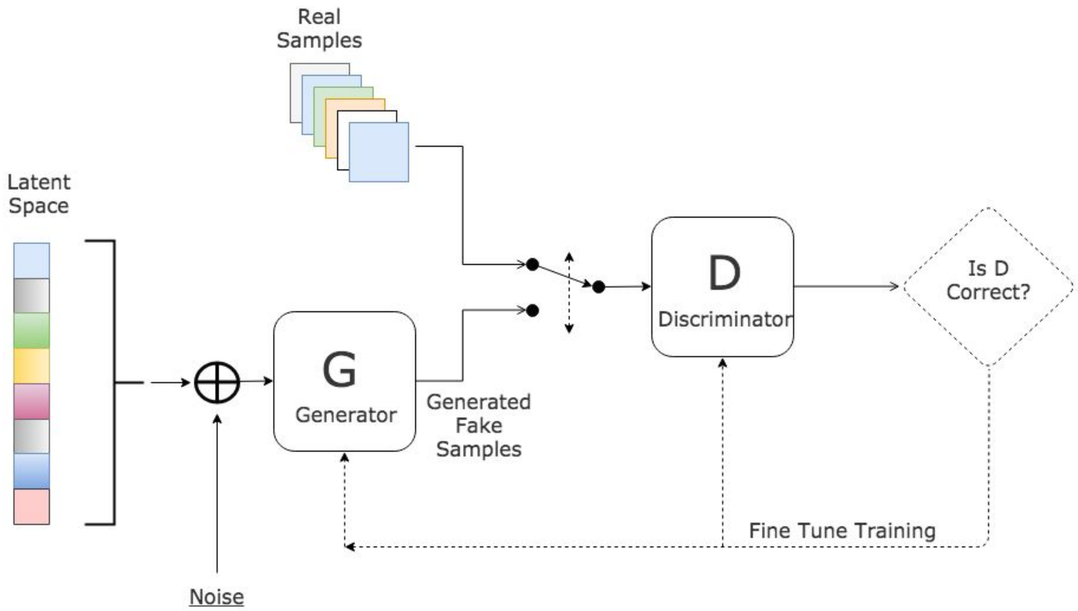
More data → More simulation

- *The ultimate goal of collecting x20 more data is to probe new phenomena by comparing accurate measurements to precise predictions*
  - *To do so, one need to increase the amount of produced simulated events*
  - *Moreover, **the simulation will be more CPU expensive, because the events are more complex***
- *The needs of reference simulated samples will go beyond the budget we foresee to have assuming a flat $$$ budget*



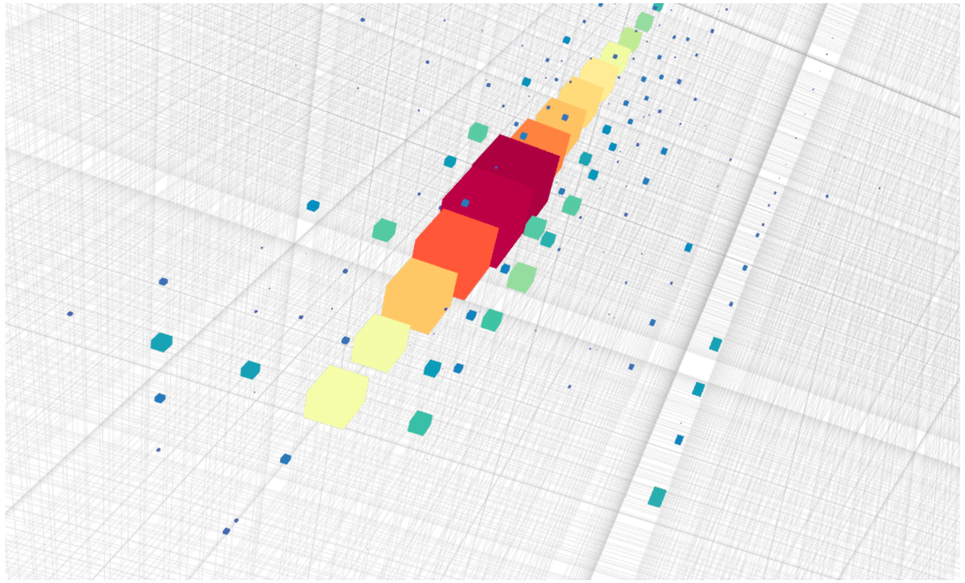
Neural Networks as Generators

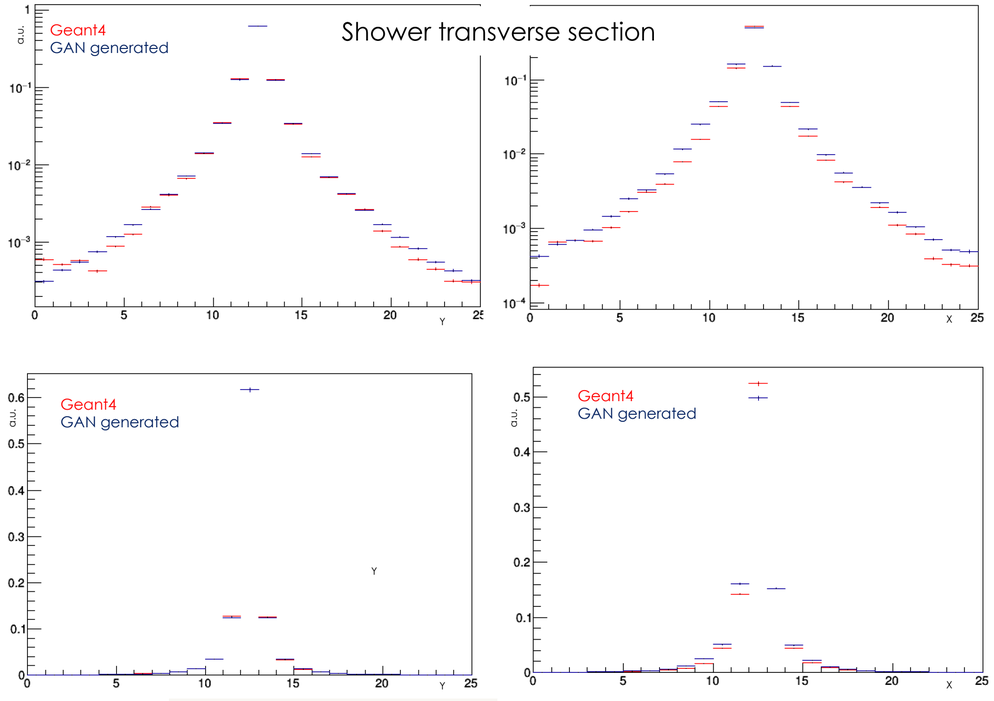
- *Networks can be trained to generate image*
- *This is done with an adversarial training*
  - *A generator produces images starting from random noise (+ some other image, optionally)*
  - *A discriminator is asked to distinguish these generated images from the real ones*
  - *The two networks are trained against each other, until the discriminator is maximally confused*



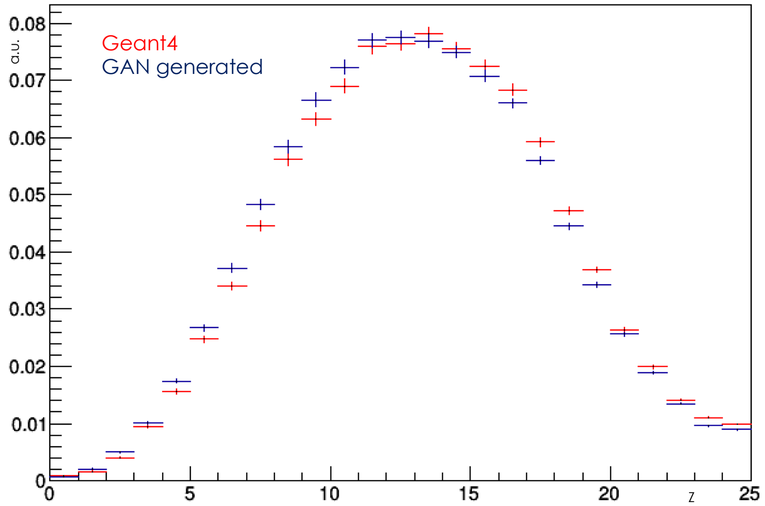
GAN for calorimeter showers

- *Future detectors will be 3D arrays of sensors*
  - *ideal for ConvNN-based models*
- *Conv. GANs can then be used to speed up shower simulation*





Shower longitudinal section

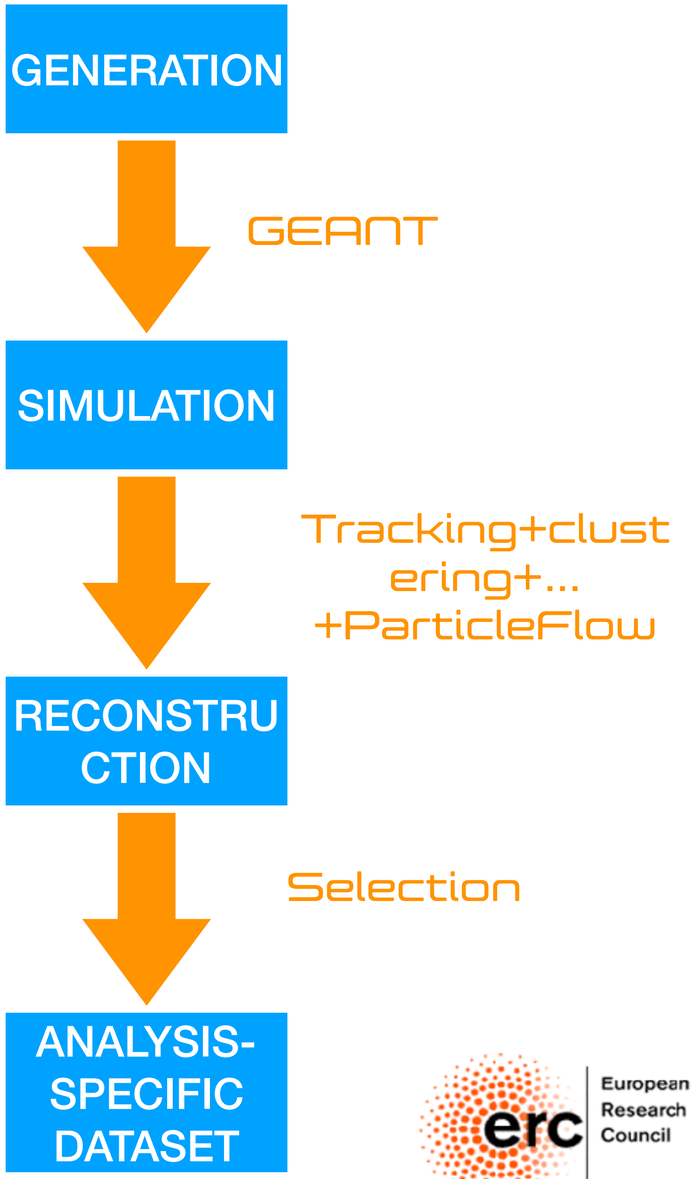


https://dl4physicalsciences.github.io/files/nips_dlps_2017_15.pdf

European Research Council

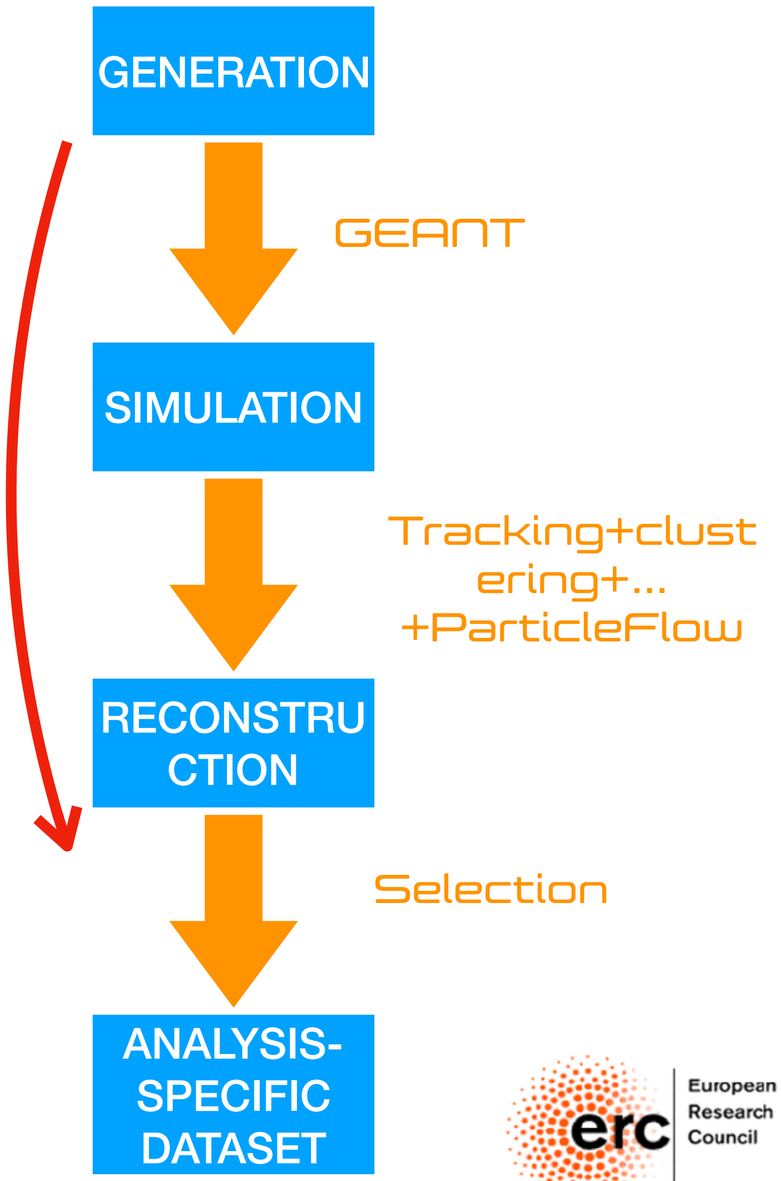
Simulation is half of the problem

- *The simulation step (proton collision) and the interaction of the produced particles with the detector) is 1/2 of the CPU load in a simulation workflow*
- *Its output needs to be digitised (emulation of detector electronics) to look like recorded data*
- *The digitised event is then processed by standard reconstruction code (as heavy as the SIM step) to then be used in analysis*



Simulation is half of the problem

- *Reconstruction involves more than one detector (e.g., tracker + calorimeter) and produces (at least in CMS) a list of particles*
- *GANs were proved to be useful to emulate the SIM+RECO step in one goal*
  - *jets out of full particle reconstruction emulated in a GAN*
  - *trained on actual SIM+RECO synthetic data by CMS*



Simulation is half of the problem

- *Reconstruction involves more than one detector (e.g., tracker + calorimeter) and produces (at least in CMS) a list of particles*
- *GANs were proved to be useful to emulate the SIM+RECO step in one goal*
  - *jets out of full particle reconstruction emulated in a GAN*
  - *trained on actual SIM+RECO synthetic data by CMS*

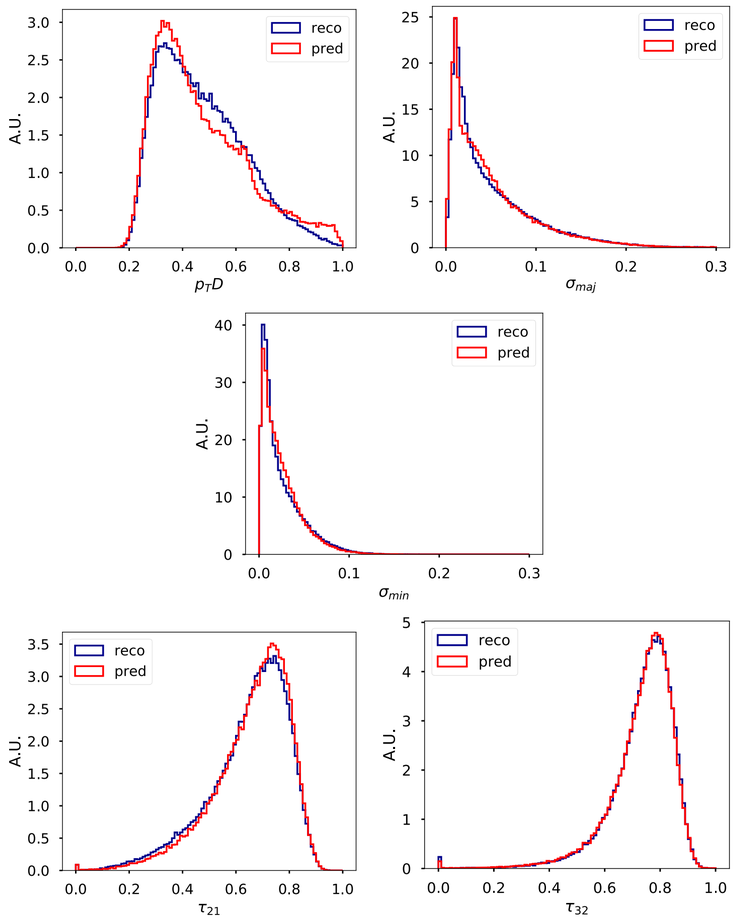


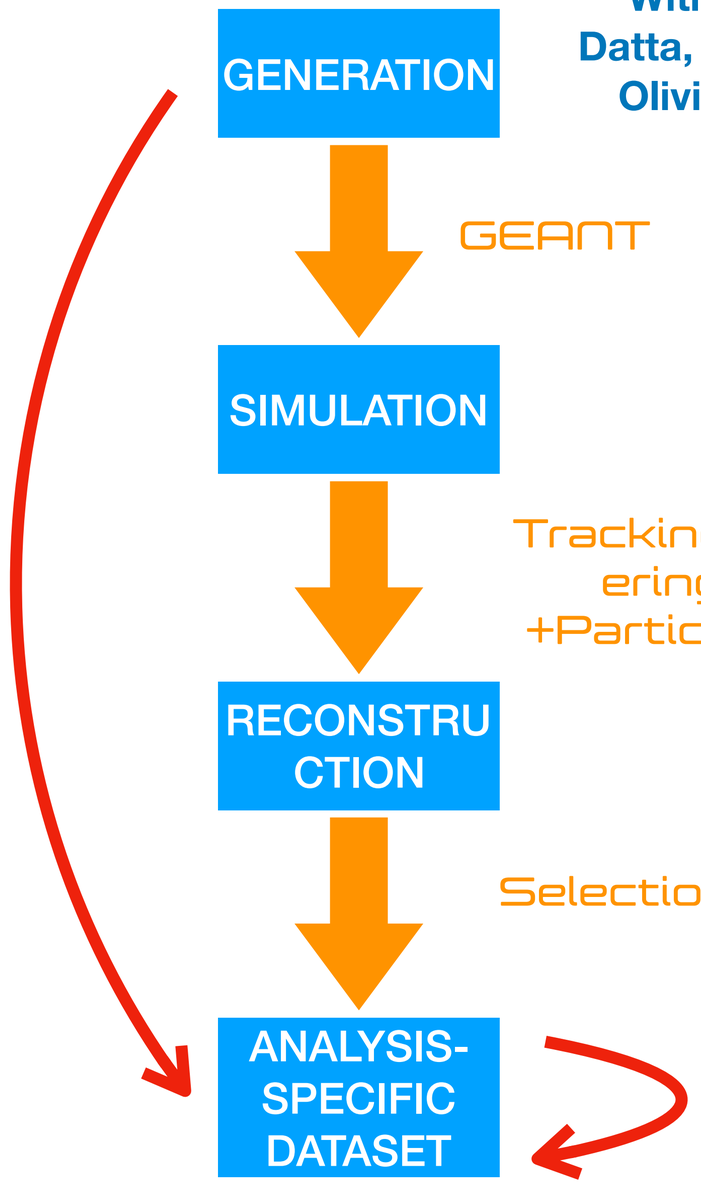
Fig. 6 Distribution of high level variables used for quark/gluon discrimination (first two rows) and merged jets tagging (last row). Blue histograms are obtained from the input data, while red ones are obtained using the generative model.

https://inspirehep.net/record/1671151

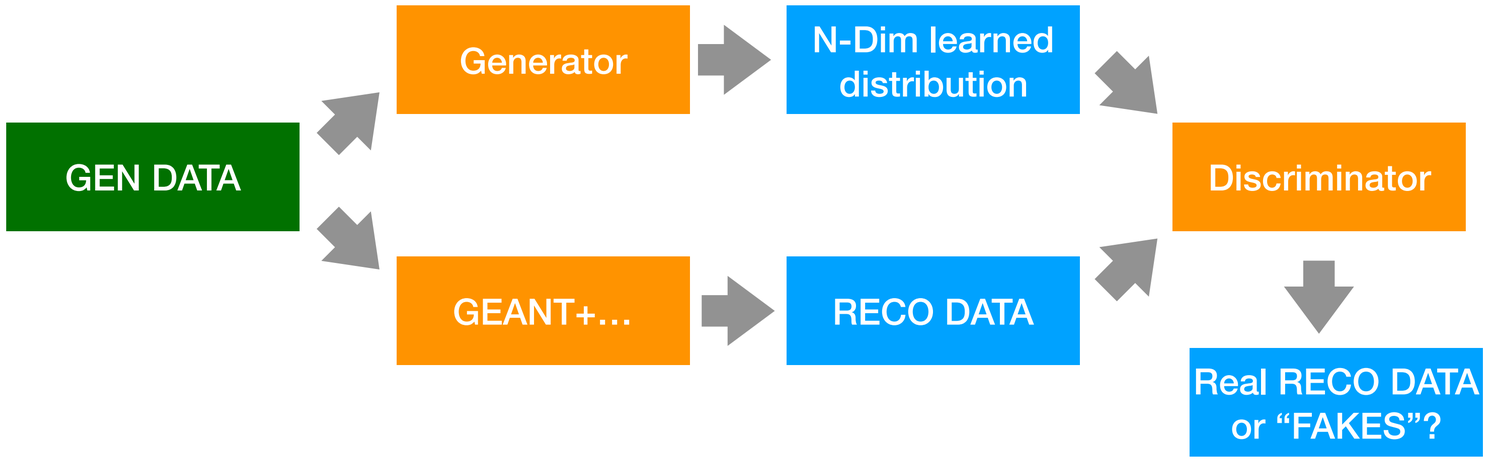
Dataset-specific generation

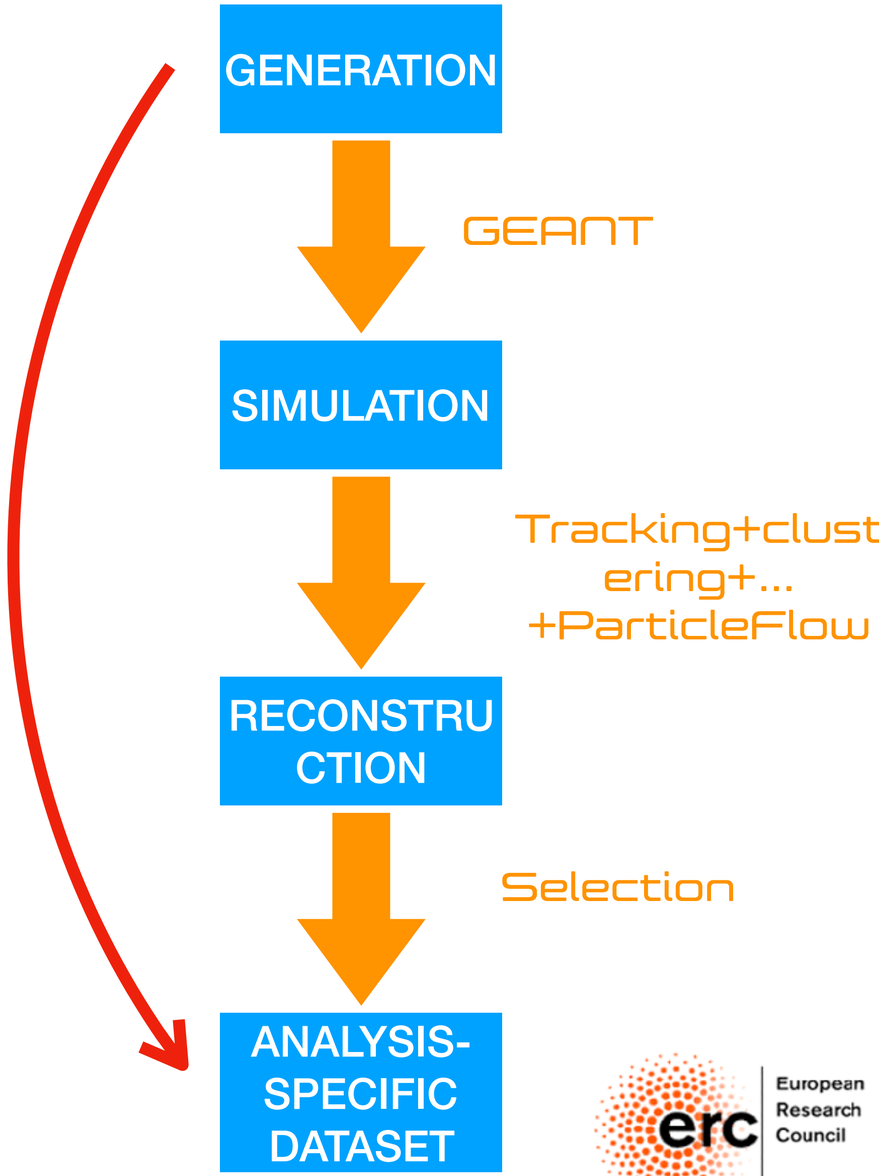
- *GANs can be trained to directly learn the multi-D distribution of the N relevant quantities for some physics analysis*
  - *save of CPU*
  - *save of storage*
- *Two approaches explored*
  - *Event "interpolation": train a generative model on RECO events, as a way to characterize a small-size dataset and generalise it to a large-size dataset (see backup)*
  - ***Fast simulation: learn the transfer function from the N quantities at generator level to the N quantities after reconstruction***

With N. Amin, K. Datta, B. Hashemi, D. Olivito, to appear soon



Fast Simulation





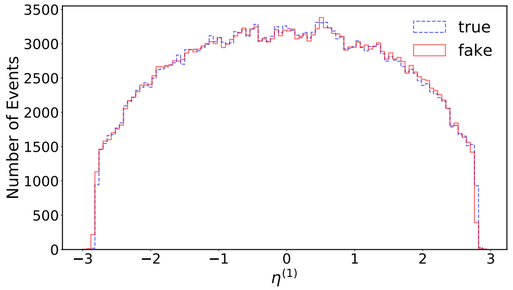
- *GANs could be used to replace parametric simulations of the detector response (fastsim)*
- *Given NN capability of approximating complex functions, one could in principle retain fullsim-like performances*
- *We gave this a try with our analysis-specific setup*

Fastsim Results

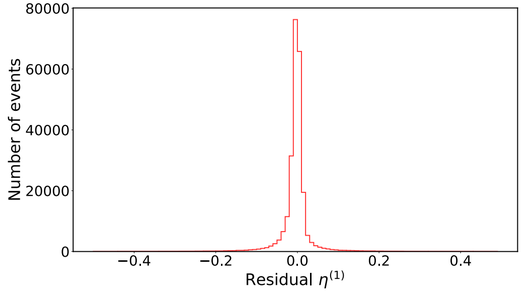
- *GAN trained from cartesian coordinates 4momenta*
- *Momenta well modelled in cylindrical coordinates too*
  - *This means that not only the 1D distributions, but also the correlations are learned*
- *Notably, the invariant mass of the dimuon system is correctly model*

With N. Amin, K. Datta, B. Hashemi, D. Olivito, to appear soon

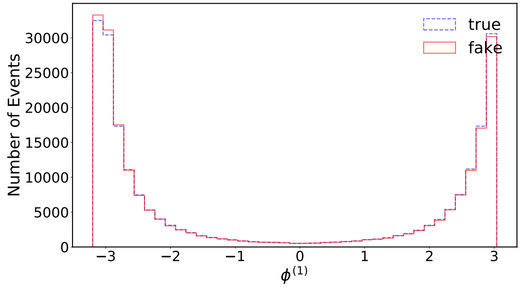
Distribution



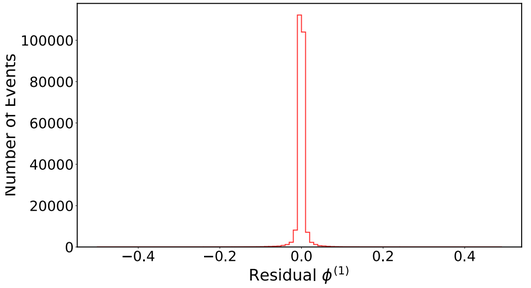
Residual

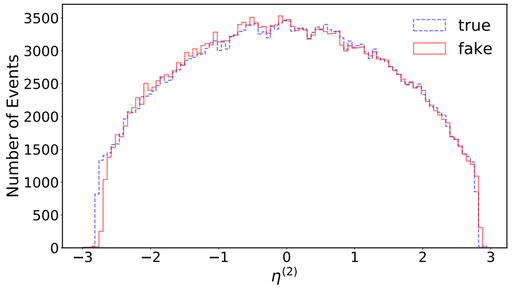


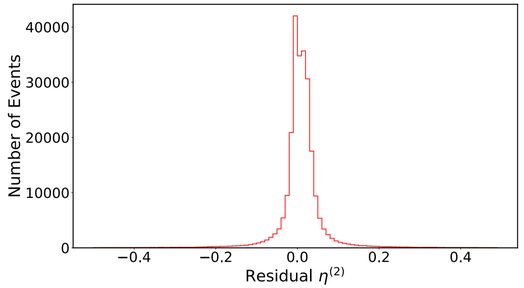
Distribution

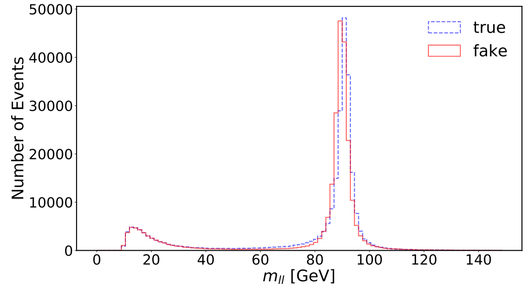


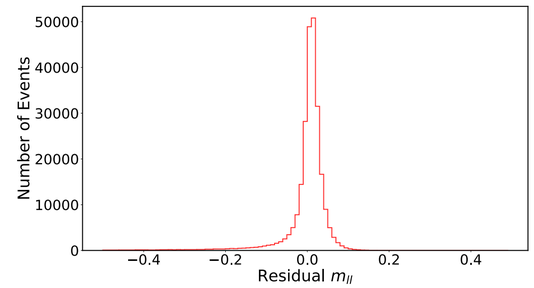
Residual



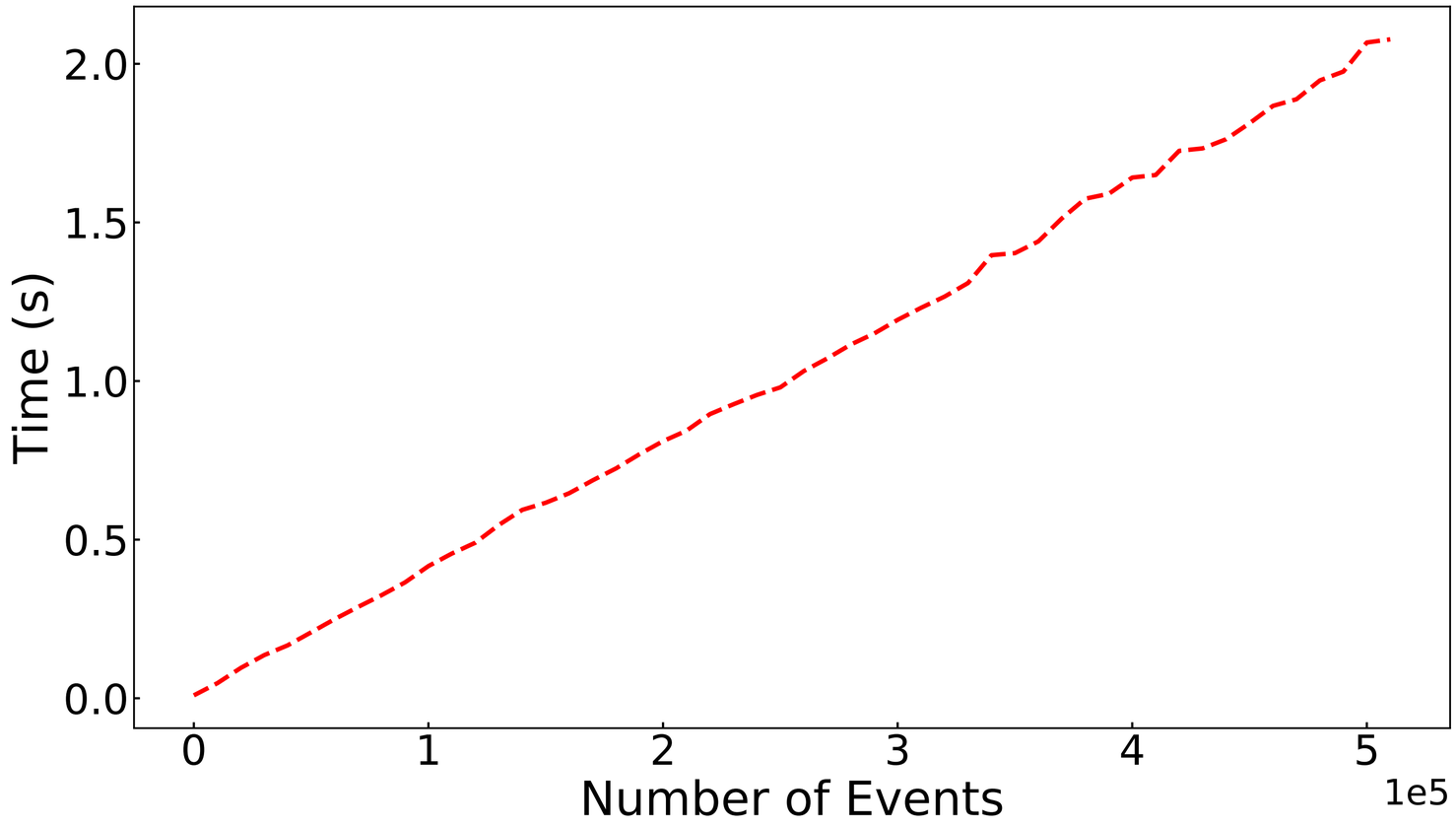


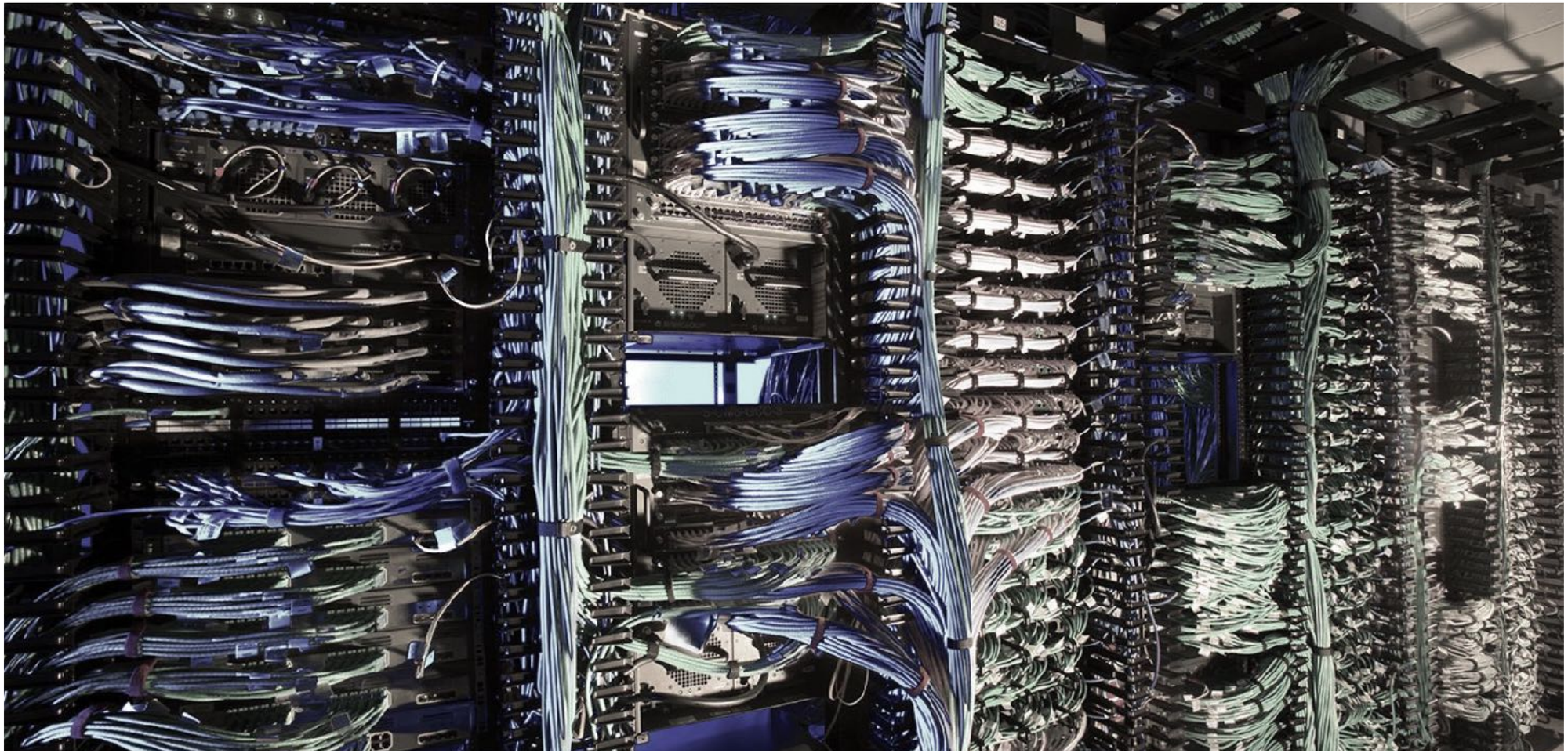






And it's fast...

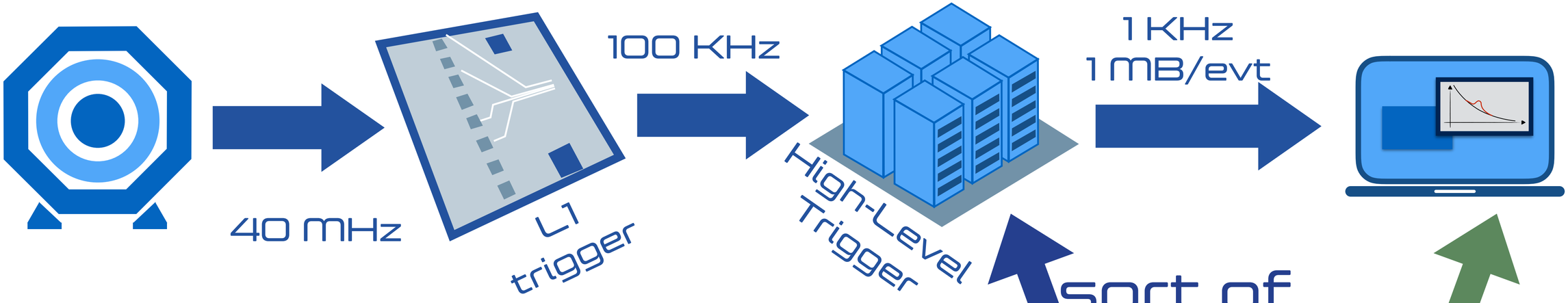




Fast Decision Taking

European Research Council

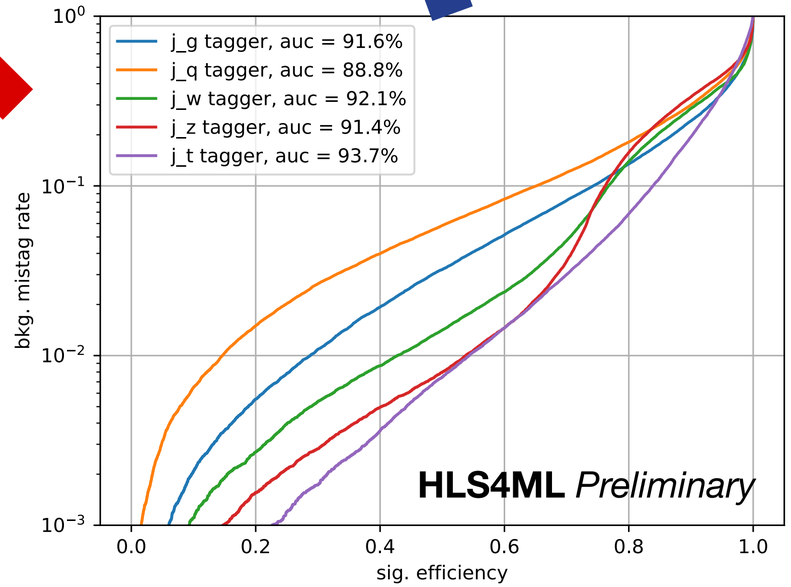
The LHC Big Data Problem

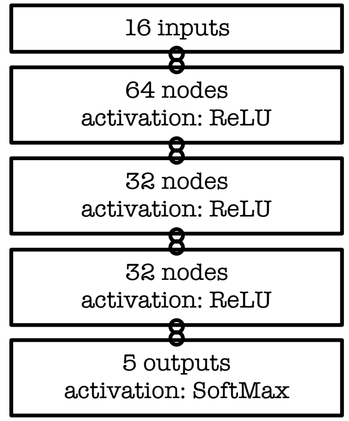


sort of OK

Easy

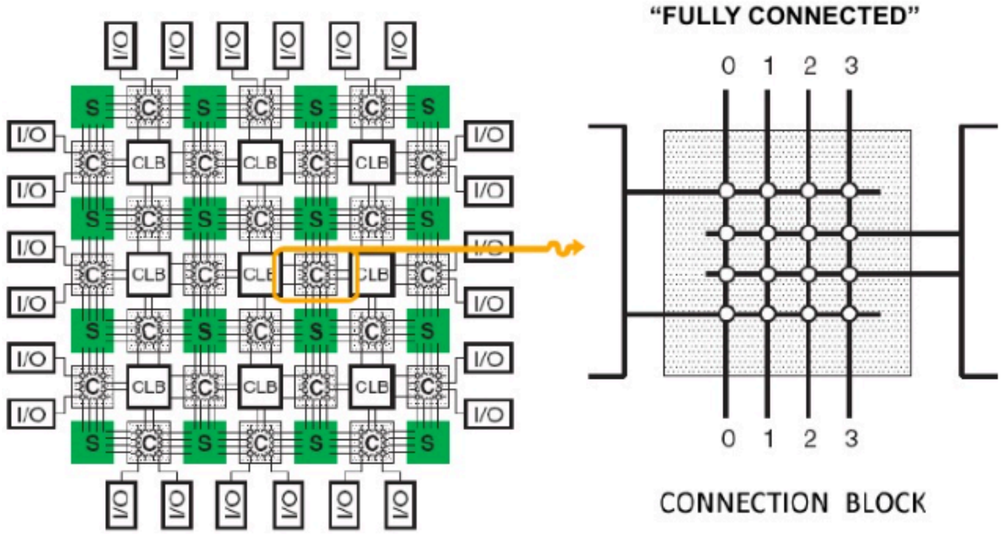
Complicated: need to be fast (10 ms) and with very small resources



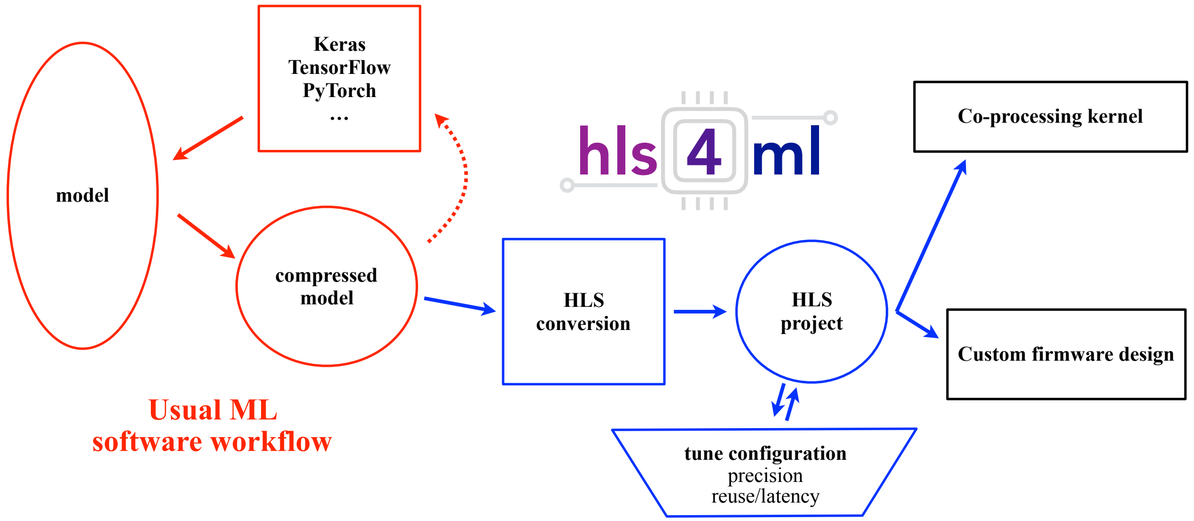


Bring DL to L1

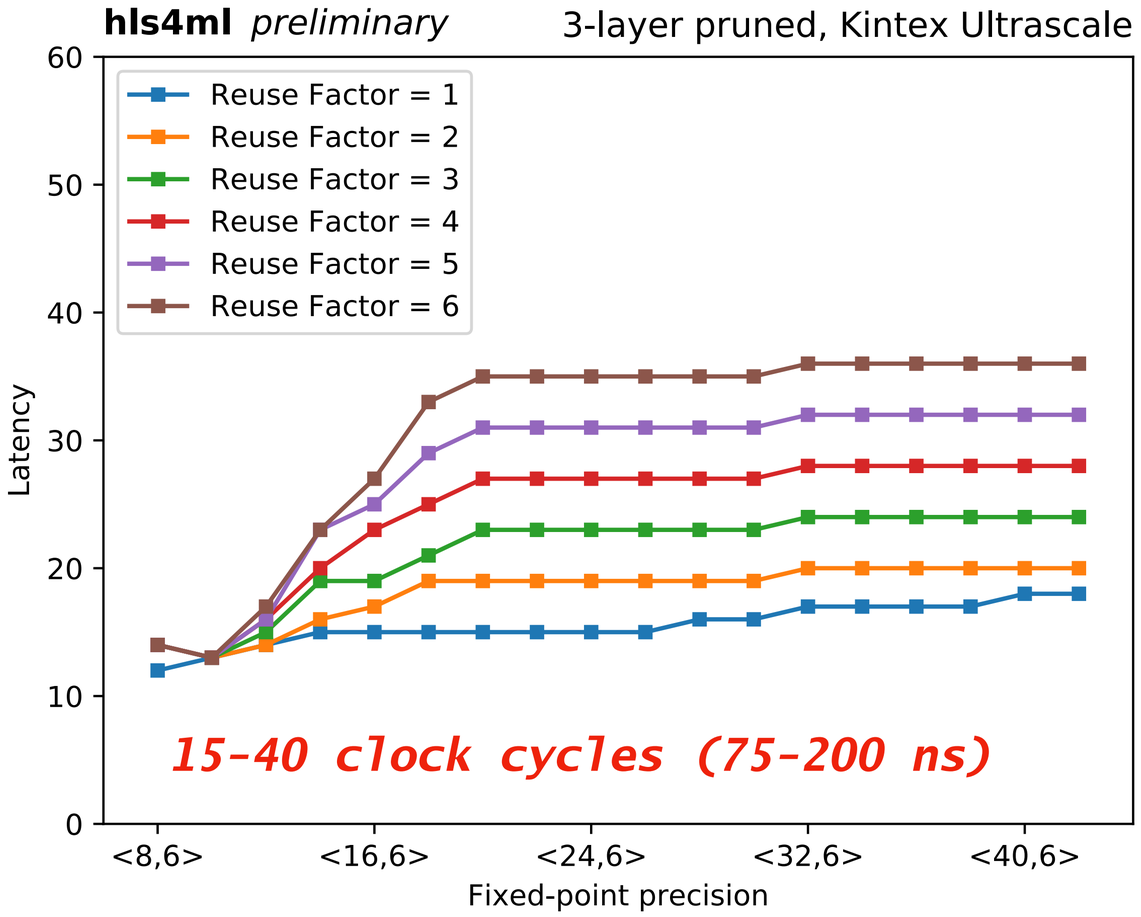
- *The L1 trigger is a complicated environment*
  - *decision to be taken in ~10 µsec*
  - *only access to local portions of the detector*
  - *processing on Xilinx FPGA, with limited memory resources*
- *Some ML already running @L1*
  - *CMS has BDT-based regressions coded as look-up tables*
- *Working to facilitate DL solutions @L1 with dedicated library*
  - *Can fit a model on FPGA playing some trick (see backup slides)*

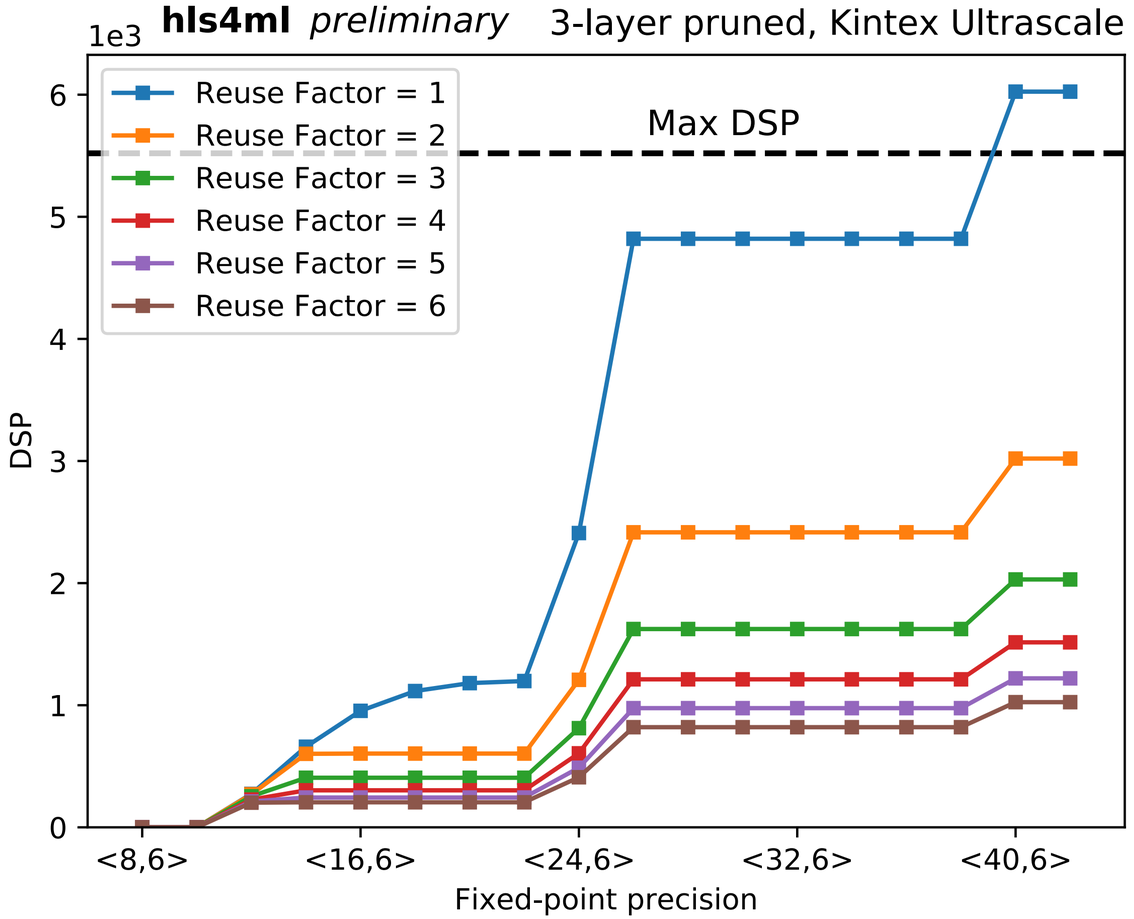


HLS4ML: CERN/FNAL/MIT collaboration



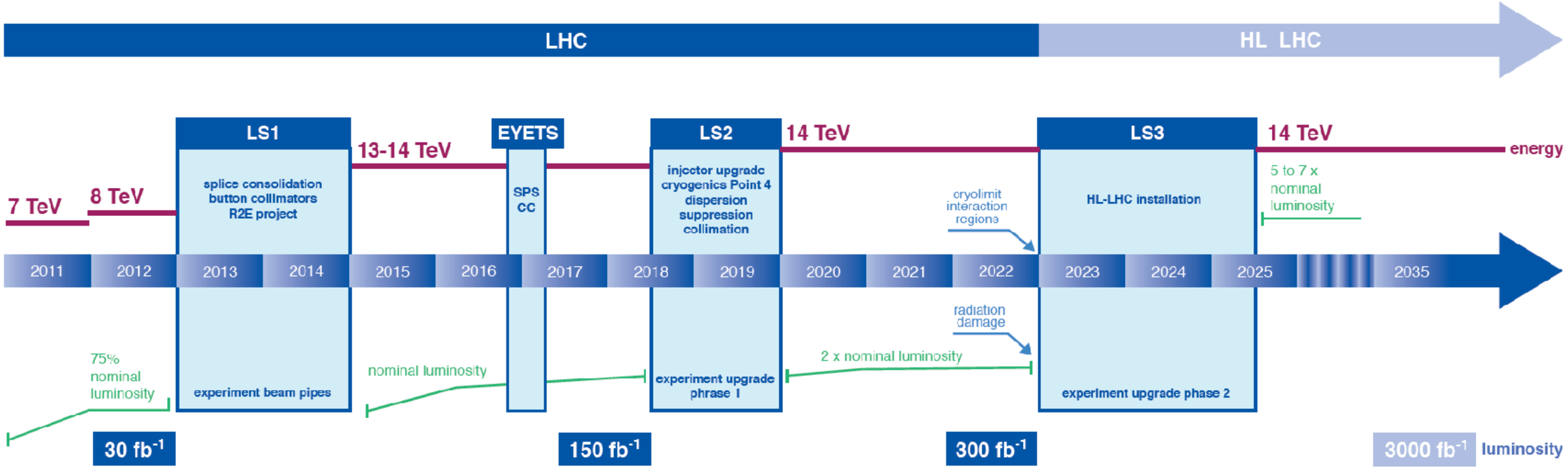
Performances





NB: FPGA emulator over-estimates resource needs by a factor 2-4 (tested our emulation vs actual deployment)

A roadmap towards HL-LHC



- *We need to be ready by **2025** (High-Luminosity LHC)*
- *LHC Run 3 (2020-2022) is the ultimate demonstration opportunity*
  - *produce proof-of-principle studies on simulations and open datasets*
  - *bring ML expertise at CERN and in the experiments*
  - *within experiments, develop/test/deploy ML solutions to solve technical tasks*

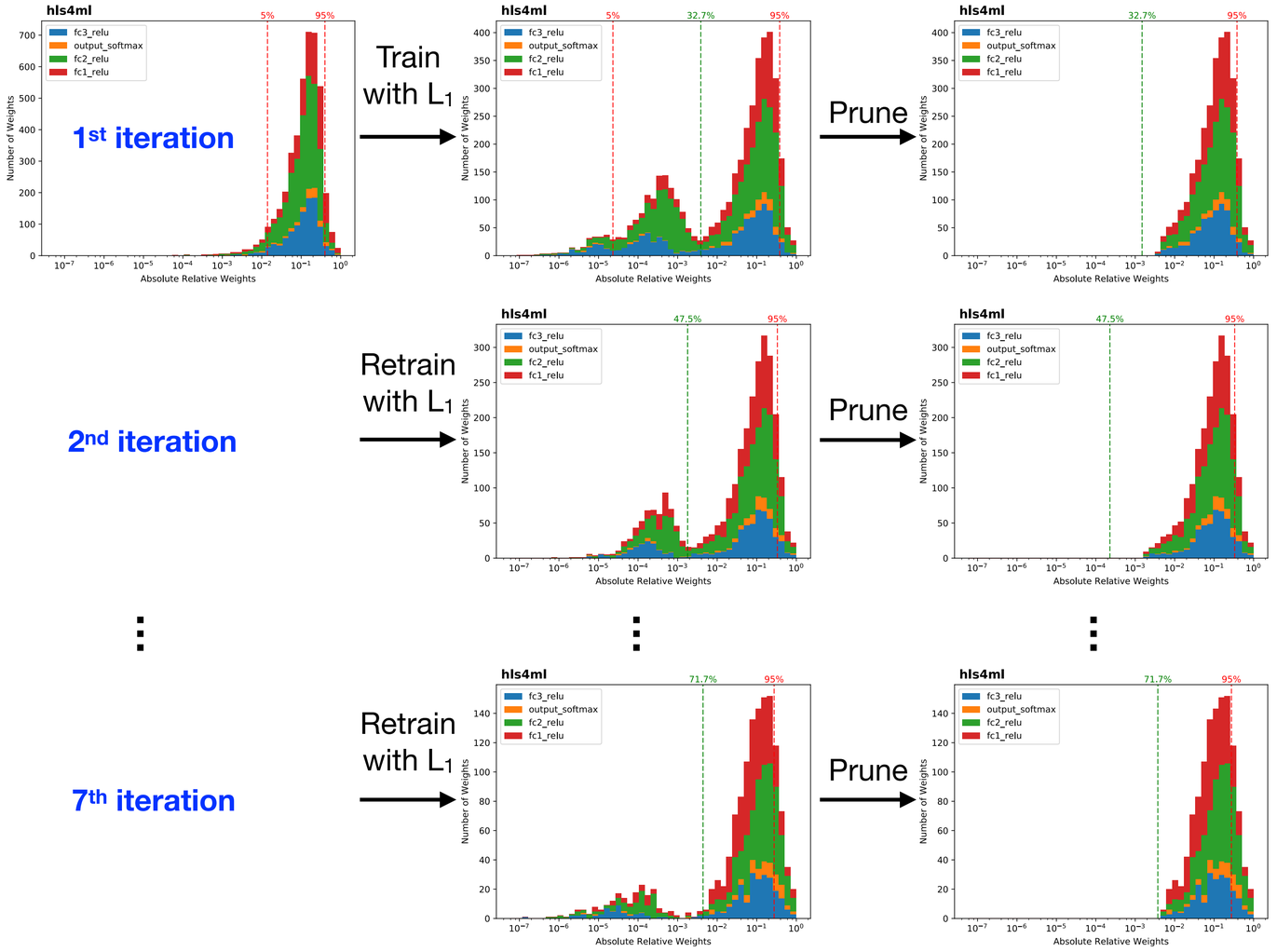
Backup

Make the model cheaper

- ***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*

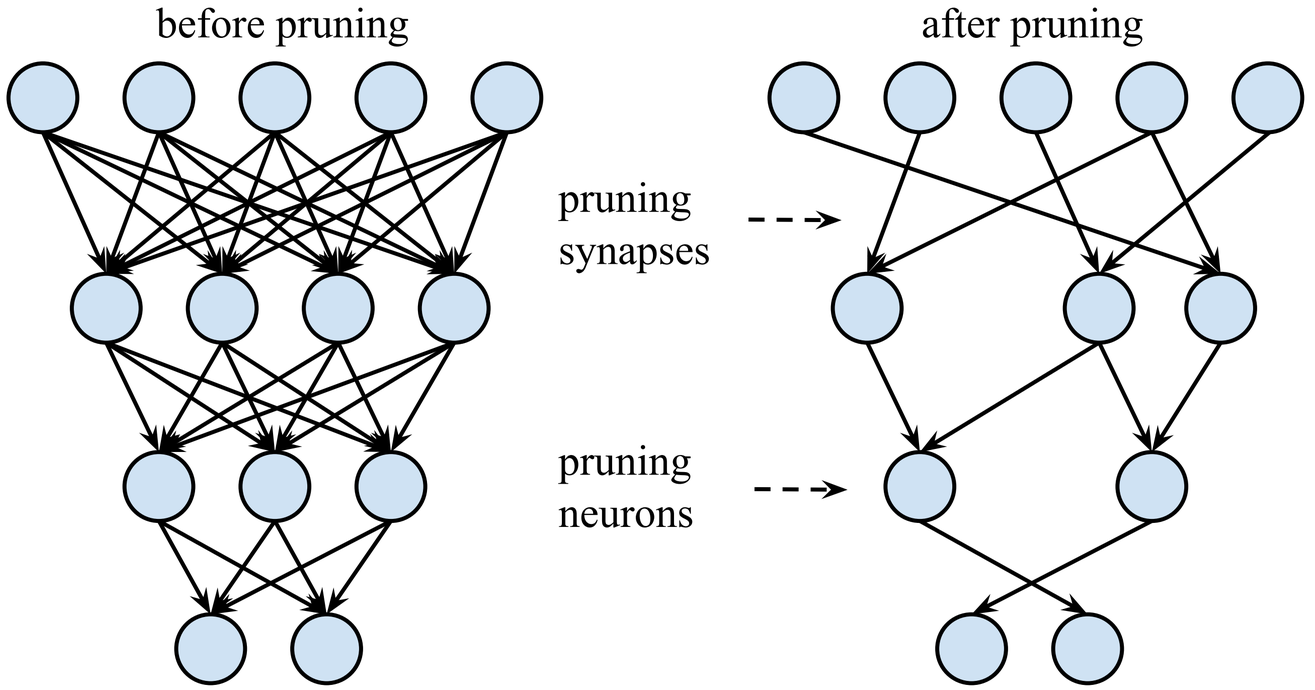


Make the model cheaper

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

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

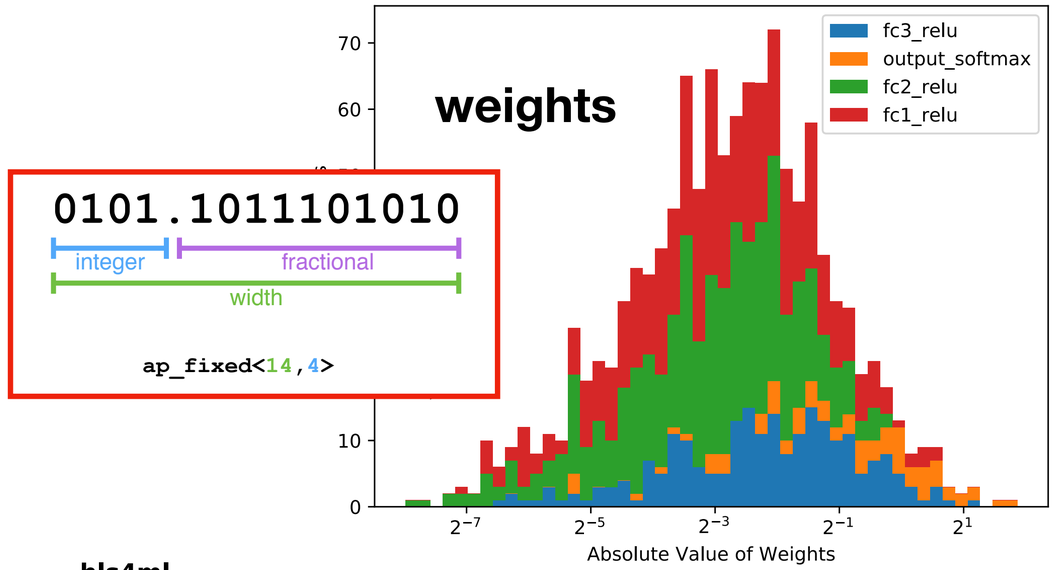
  - *Remove the small parameters*
  - *Retrain*

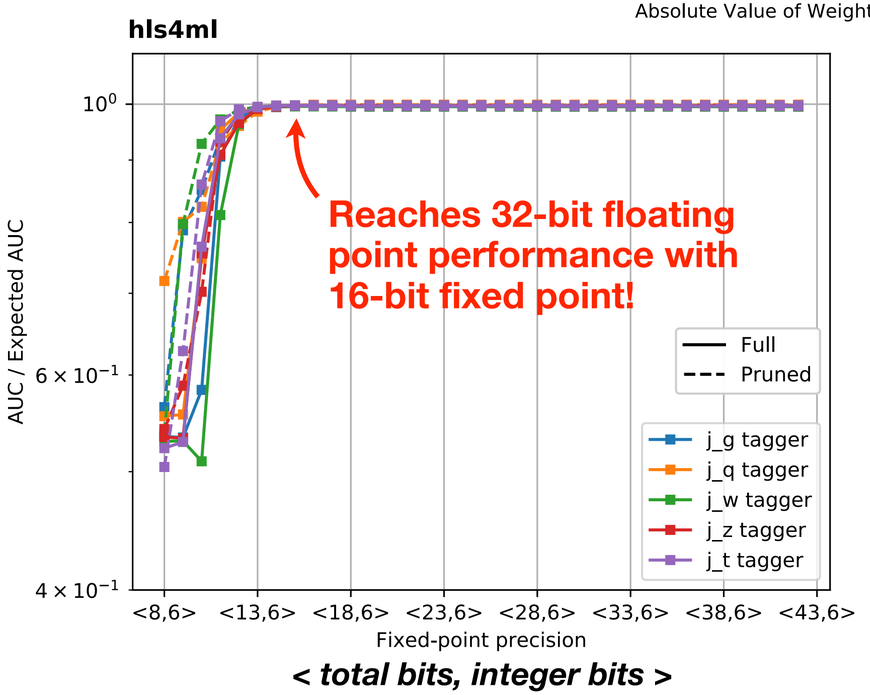


→ 70% reduction of weights and multiplications w/o performance loss

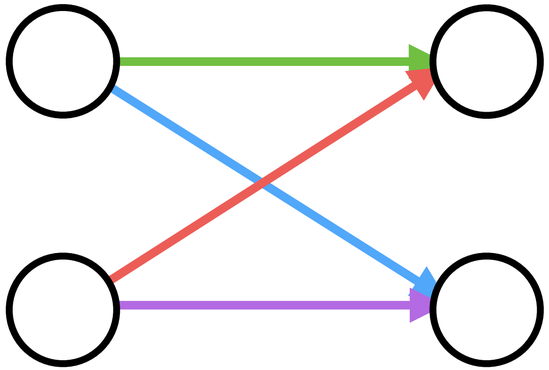
Make the model cheaper

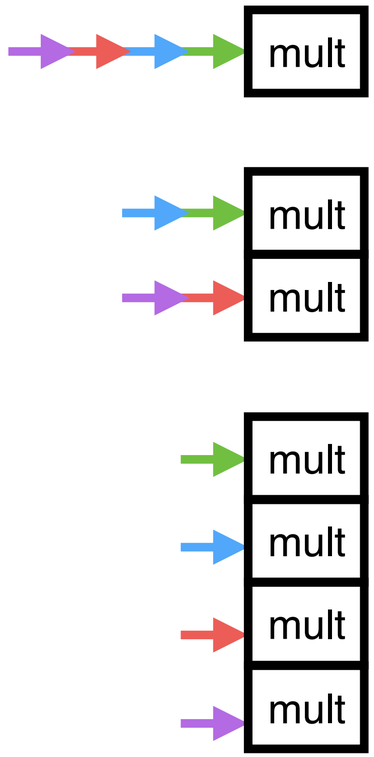
- ***Quantization***: *reduce the number of bits used to represent numbers (i.e., reduce used memory)*
  - *models are usually trained at 64 or 32 bits*
  - *this is not necessarily needed in real life*
- *In our case, we could reduce to 16 bits w/o loosing precision*
- *Beyond that, one would have to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





Speed vs Memory





Fully serial

reuse = 4
use 1 multiplier 4 times

reuse = 2
use 2 multipliers 2 times each

reuse = 1
use 4 multipliers 1 time each

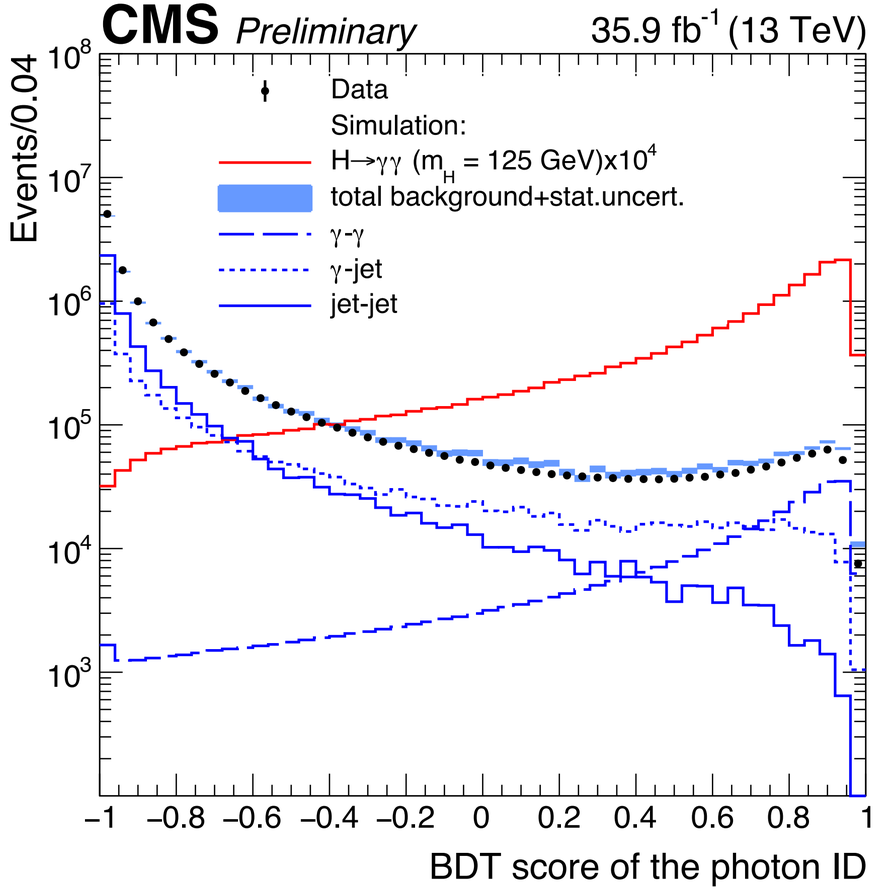
Fully parallel

Less resources/
Less throughput

Reuse factor: how much to parallelize operations in a hidden layer

What we do with ML today

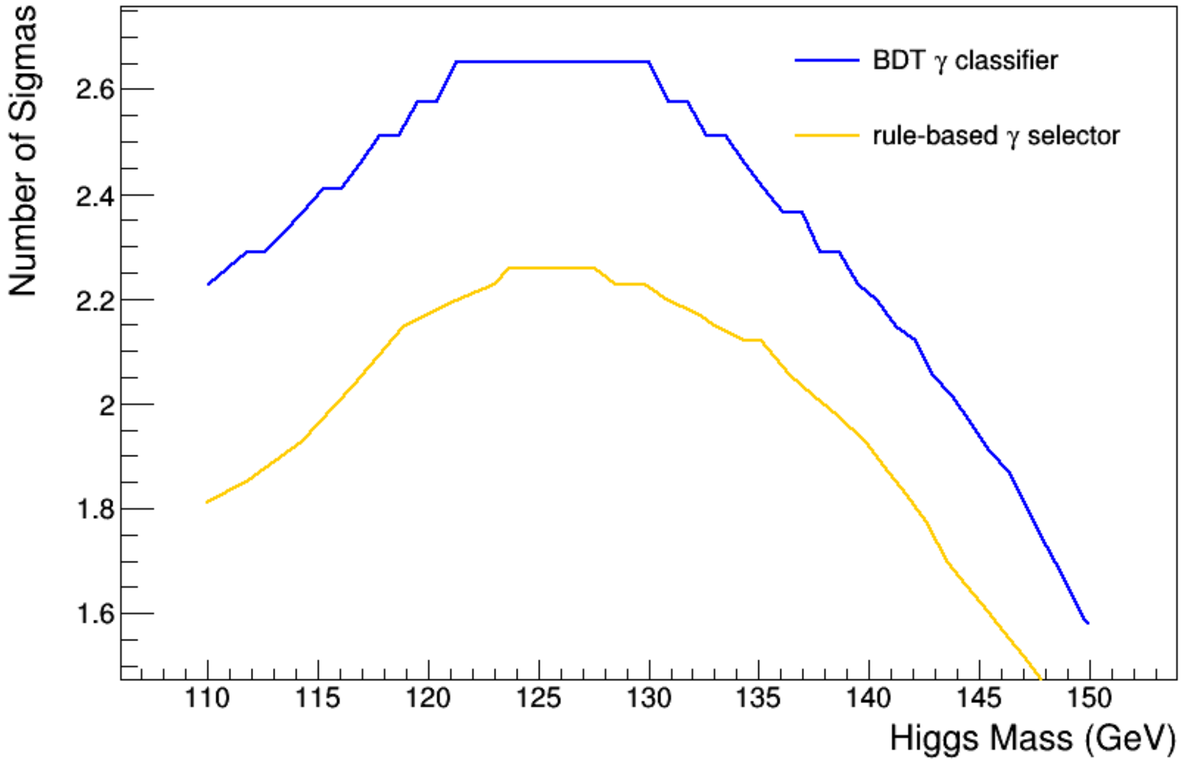
- *Classification:*
  - *identify a particle & reject fakes*
  - *identify signal events & reject background*
- *Regression:*
  - *Measure energy of a particle*
- *We typically use BDTs for these task*
  - *moved to Deep Learning for analysis-specific tasks*
  - *same will happen for centralised tasks (eventually)*

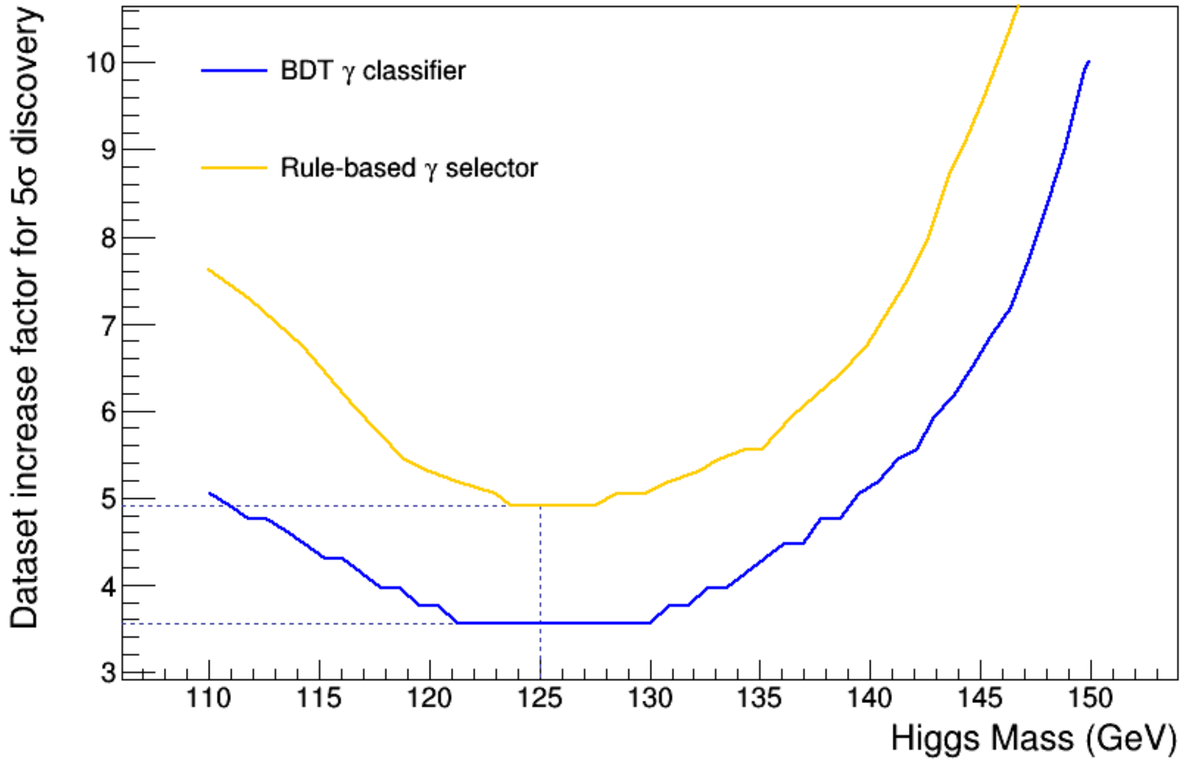


Centralised task (in online or offline reconstruction)
Analysis-specific task (by users on local computing infrastructures)

Example: ML for Higgs discovery

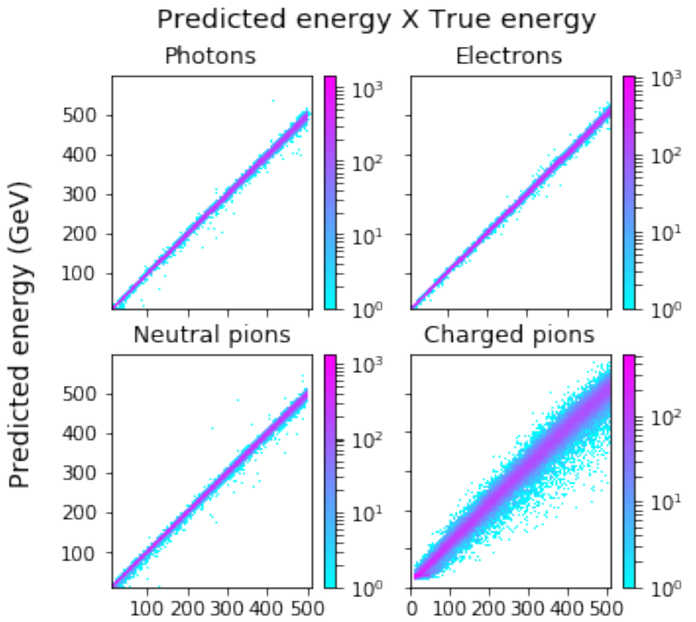
- *We were not supposed to discover the Higgs boson as early as 2012*
- *Given how the machine progressed, we expected discovery by end 2015 /mid 2016*
- *We made it earlier thanks (also) to Machine Learning*

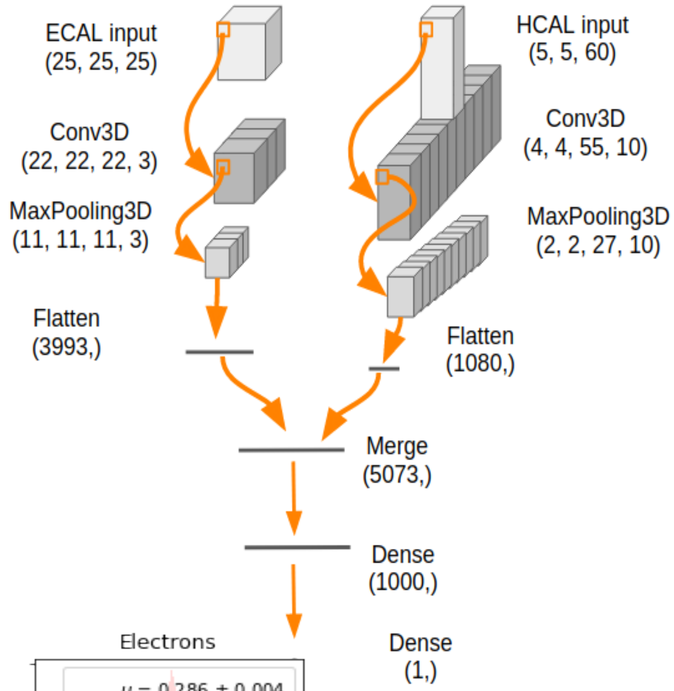


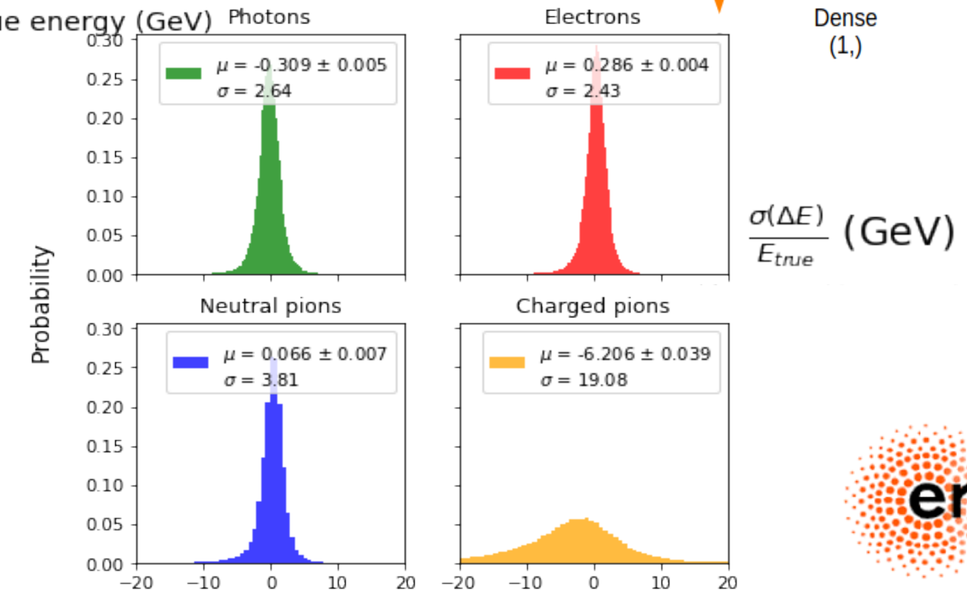


Proof of Principle: Energy Regression

- *3D Convolution NN can learn true energy of an incoming particle from the recorded hit pattern*
  - *Correctly reconstruct energy*
  - *ECAL performances better than HCAL (as expected)*
  - *π^0 resolution ~ $\sqrt{2}$ γ resolution (as expected)*
- *No high-level knowledge of physics and/or detector features*
  - *used only RAW data as inputs*
- In real life, this could be used offline, at HLT, and (maybe) even at L1





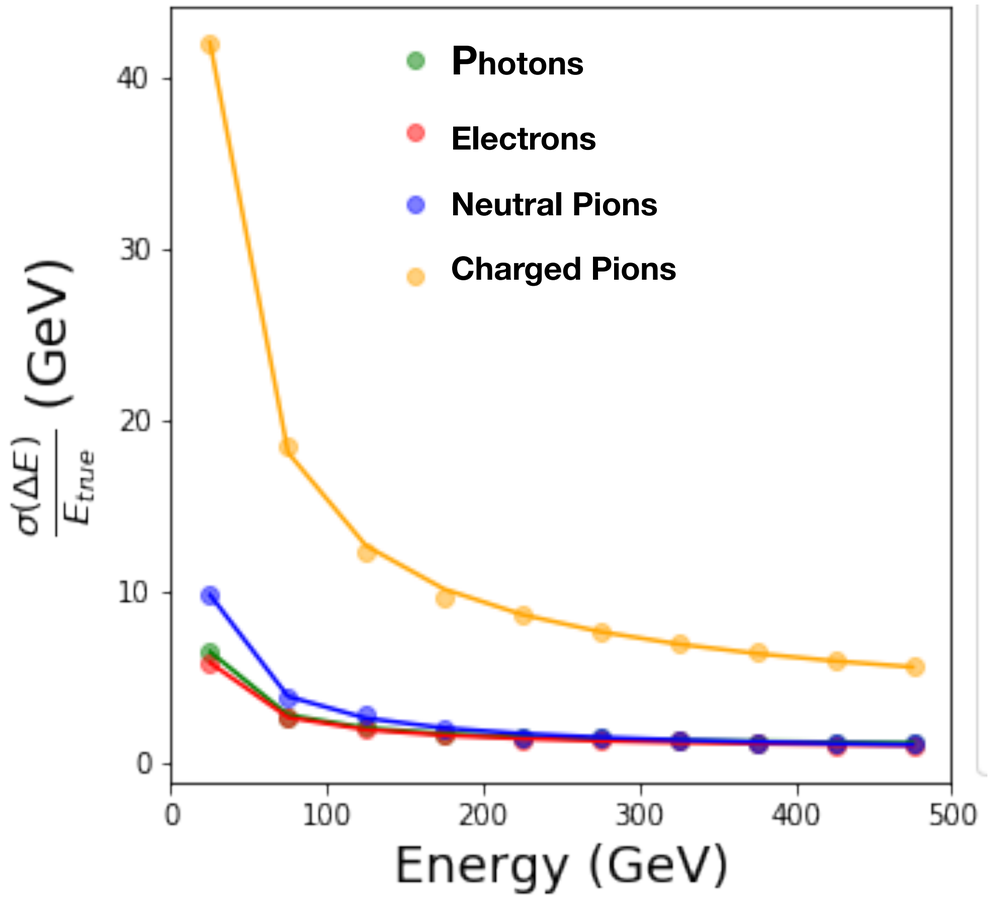


Work by K. Datta & V. Pacela, (2016/2017 Caltech Summer Students) and J.Mohapatra (OpenLab Summer student 2016)

Proof of Principle: Energy Regression

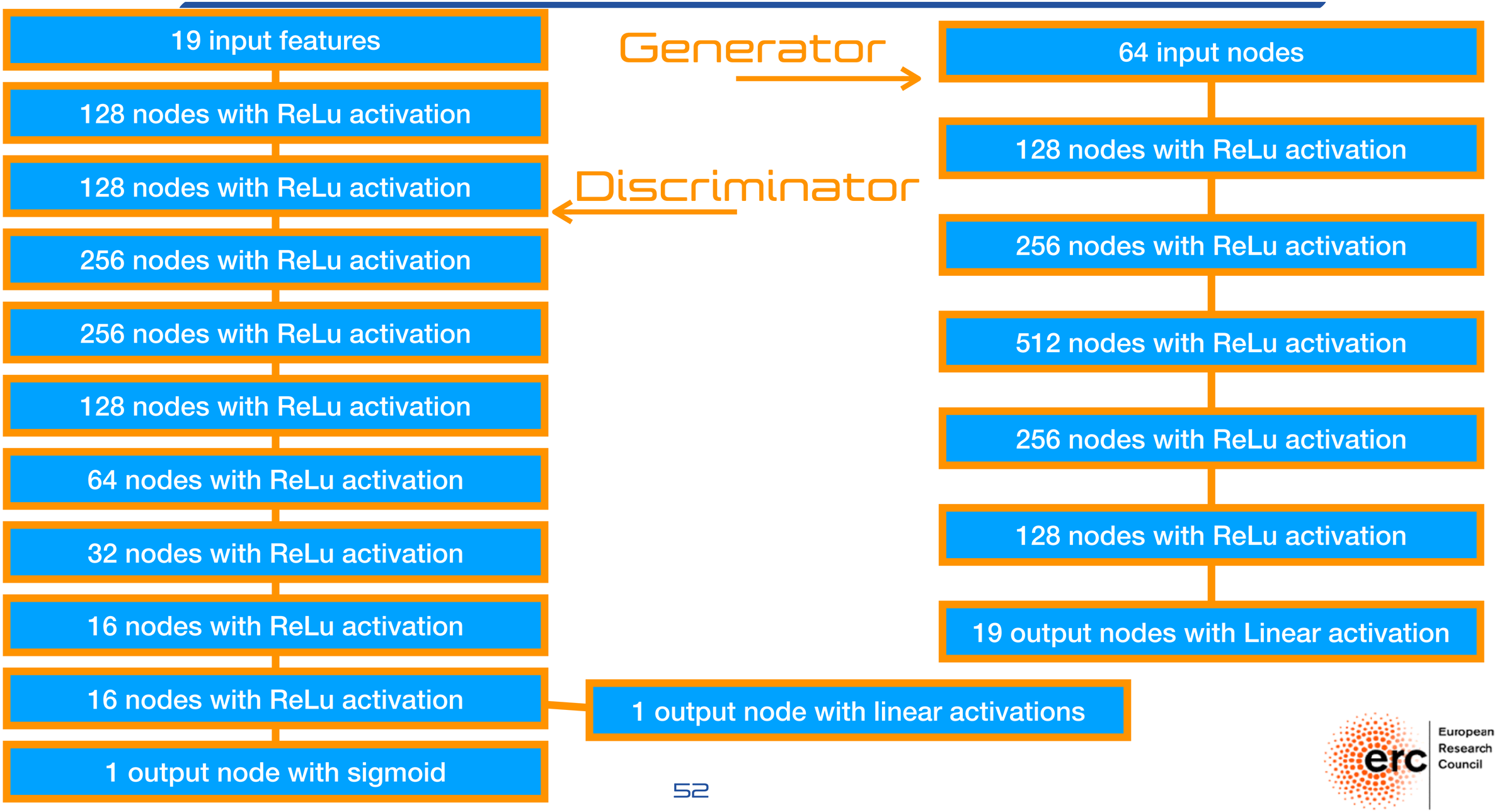
- *Competitive and meaningful results*
- *Processing time reduced by 10^3 wrt traditional approaches*

- In real life, this could be used while selecting events in real time ("trigger")



Work by K. Datta & V. Pacela, (2016/2017 Caltech Summer Students) and J.Mohapatra (OpenLab Summer student 2016)

Model details: Reco to Reco



Model details: Gen to Reco

Generator

- 5 input nodes
- 100 nodes with SeLu activation
- 250 nodes with SeLu activation
- 500 nodes with SeLu activation
- 1000 nodes with SeLu activation
- 100 nodes with SeLu activation
- 5 output nodes with Linear activation

Discriminator

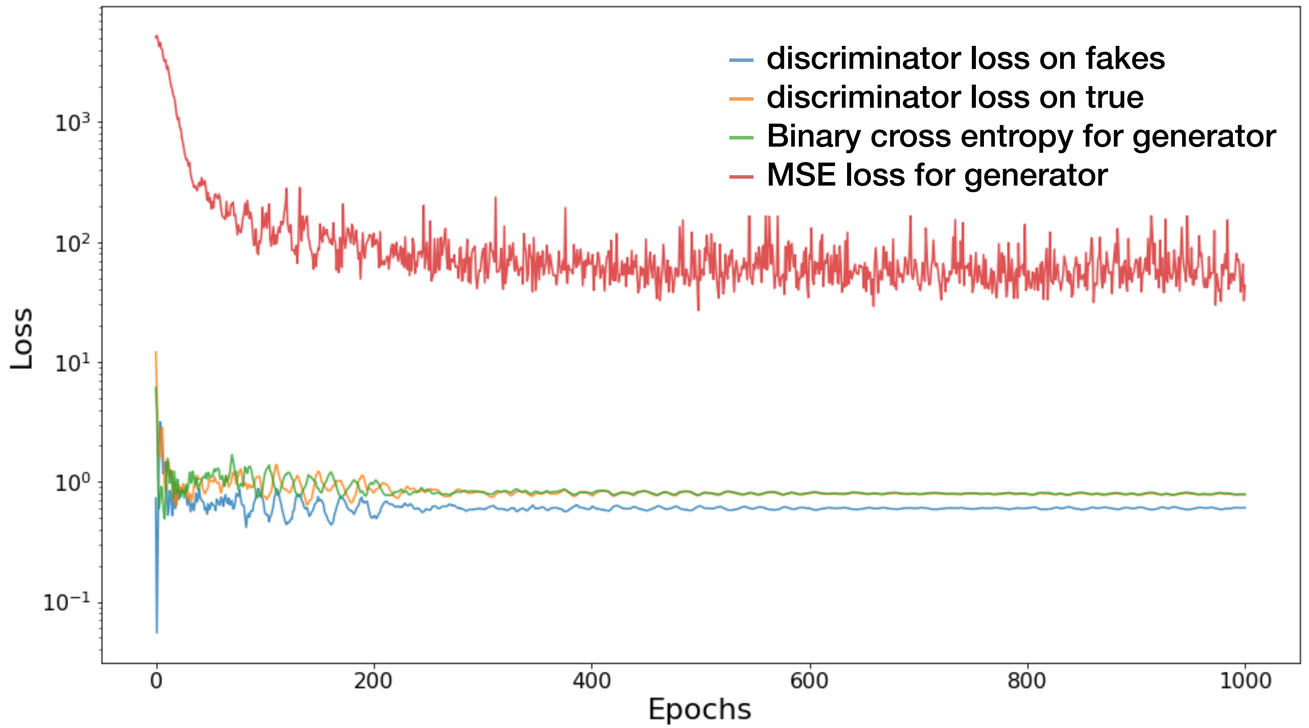
- 5 input nodes
- 1000 nodes with LeakyReLu + Dropout
- 750 nodes with LeakyReLu + Dropout
- 500 nodes with LeakyReLu + Dropout
- 250 nodes with LeakyReLu + Dropout
- 100 nodes with LeakyReLu + Dropout
- 50 nodes with LeakyReLu + Dropout
- 1 output node with sigmoid

Helping GAN training

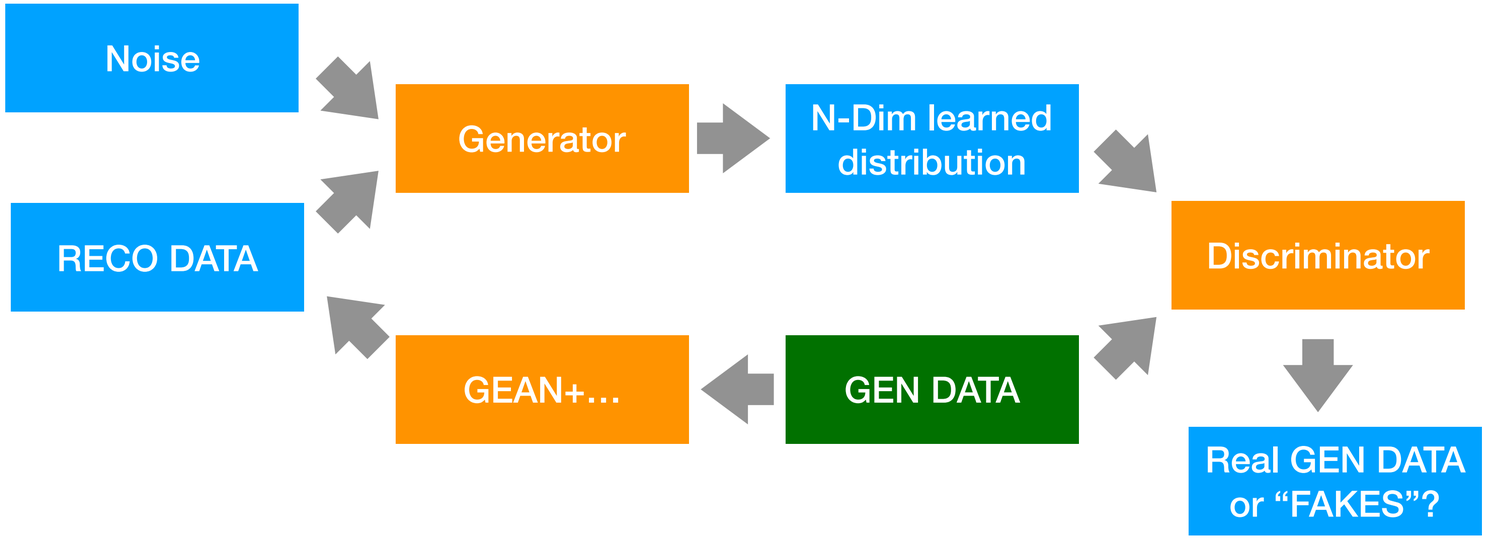
- *GANs are notoriously difficult to train*
- *MSEGan has been proposed in this paper as a way to help the training*
  - *The "trick" is to stabilize training by adding a regression step (with mse loss function) for the generator, before freezing it & training the discriminator*
  - *Similar to what is done in CALOGan & CMS-OpenData Jet GAN*
- *We tried this approach on a few setups:*
  - *GEN -> RECO fast simulation*
  - *RECO -> GEN unfolding (see backup)*

Training

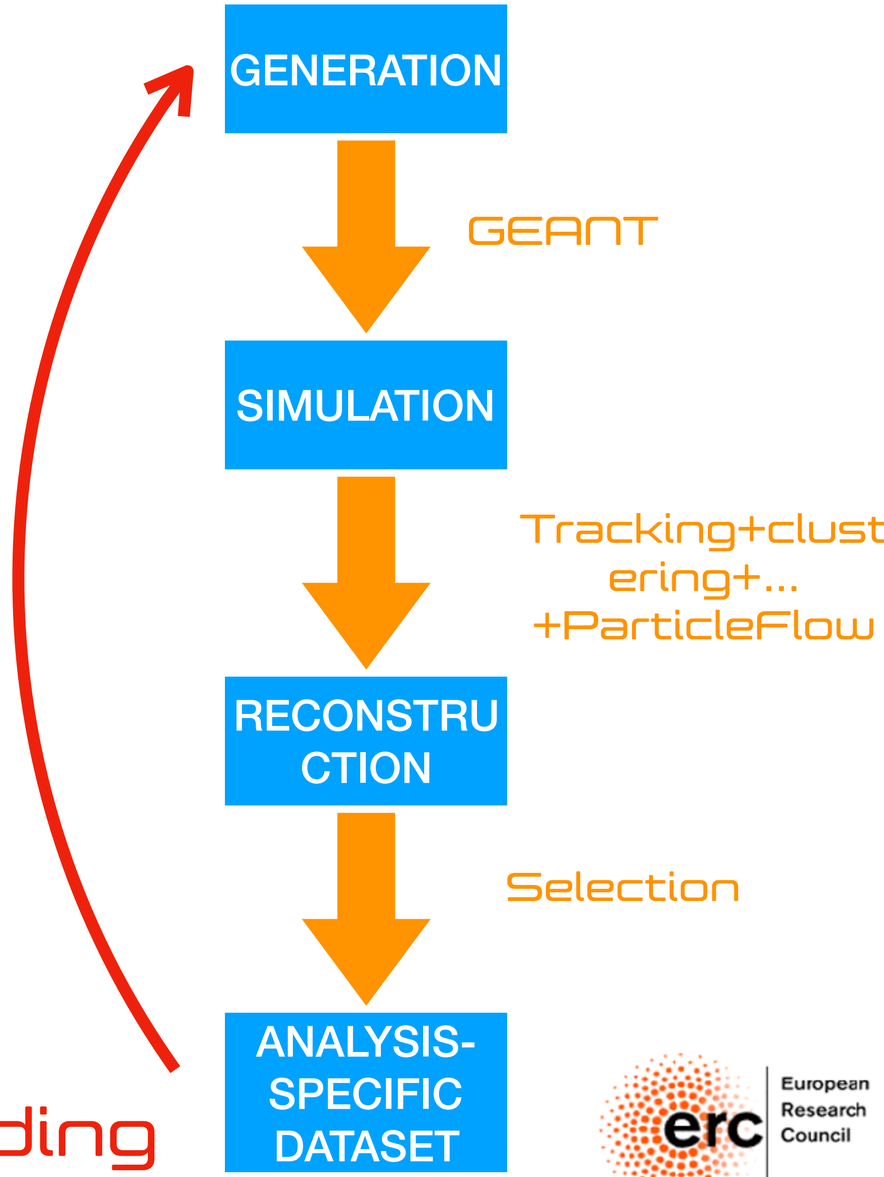
- *Simplified problem (as a first try): only three momenta of the leptons - 1 (fixed $\varphi_2 = 0$ to break rotational symmetry)*
- *Used mse for regression loss + binary cross-entropy for discriminator*
- *Much stabler training history observed, since the problem is now more supervised*



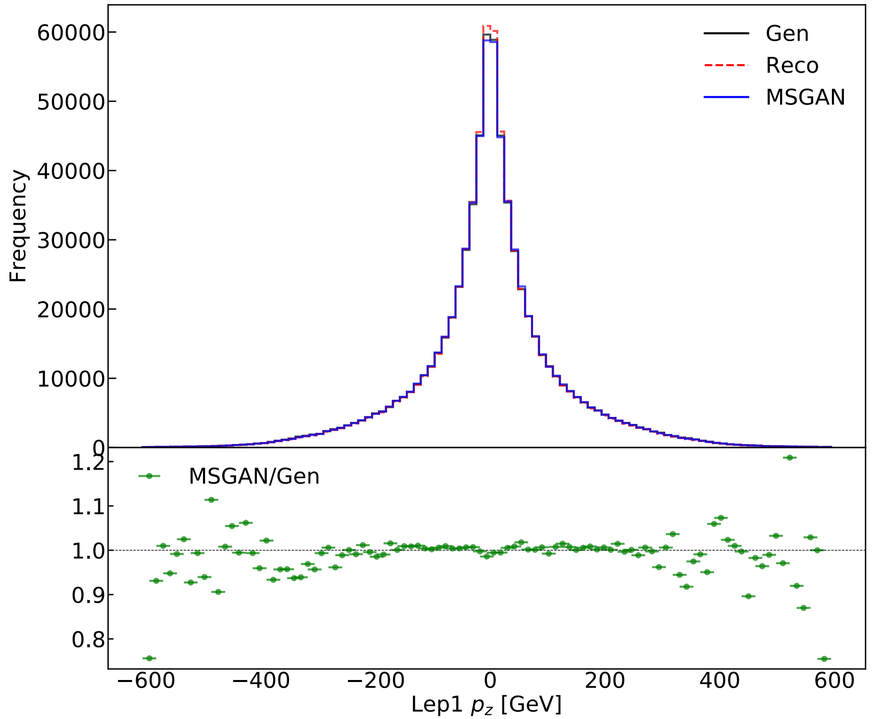
Analysis-specific unfolding

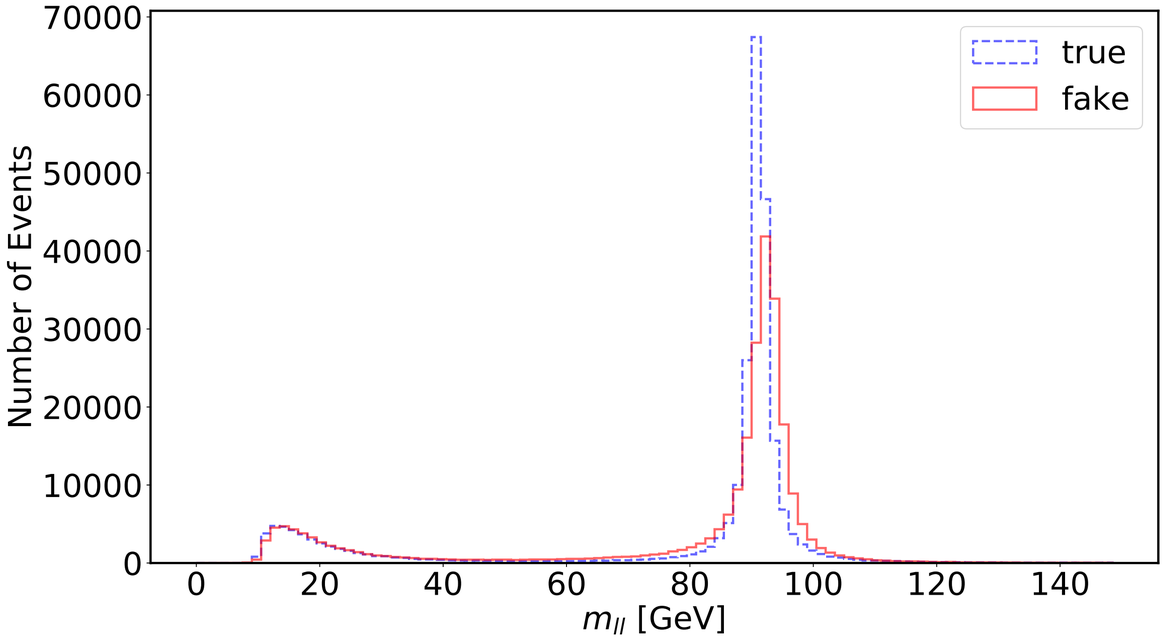


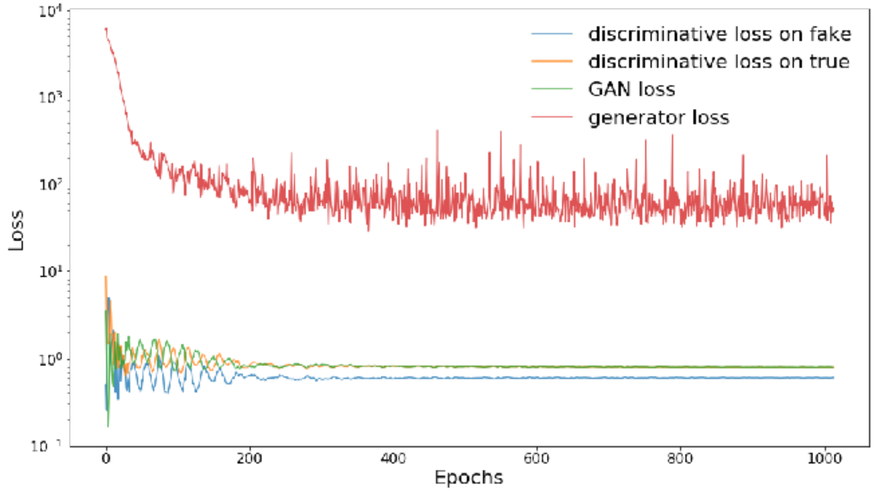
- *The same approach could be used for unfolding, i.e. to undo the effect of detector resolution*
- *Formally, the only change is the switch of the conditioned and target dataset*



MSGan Unfolding Results



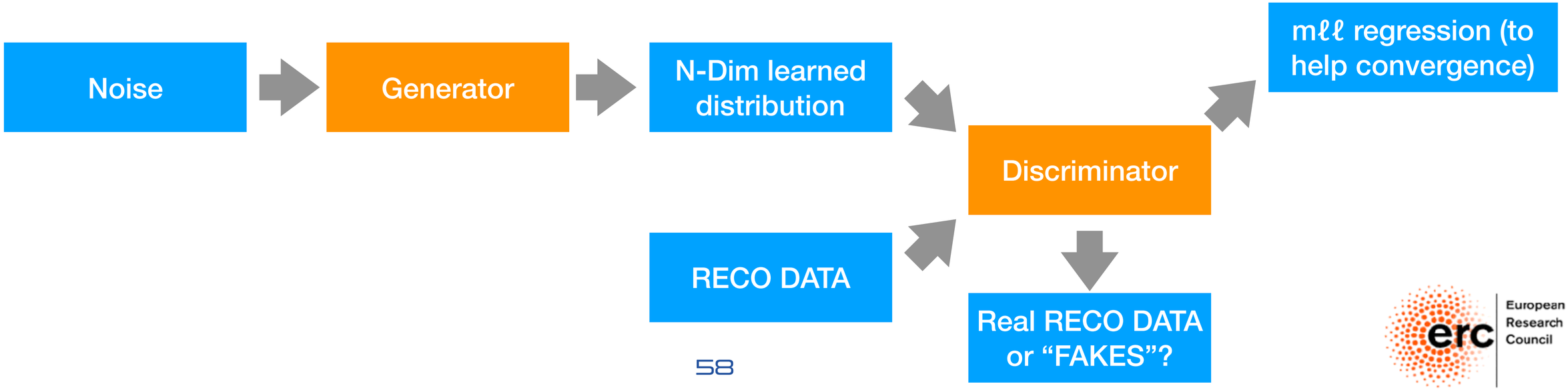




- Even in this case, results are promising but need to be tested with more "challenging" transfer functions
- More work needed to improve the result to match the typically needed accuracy

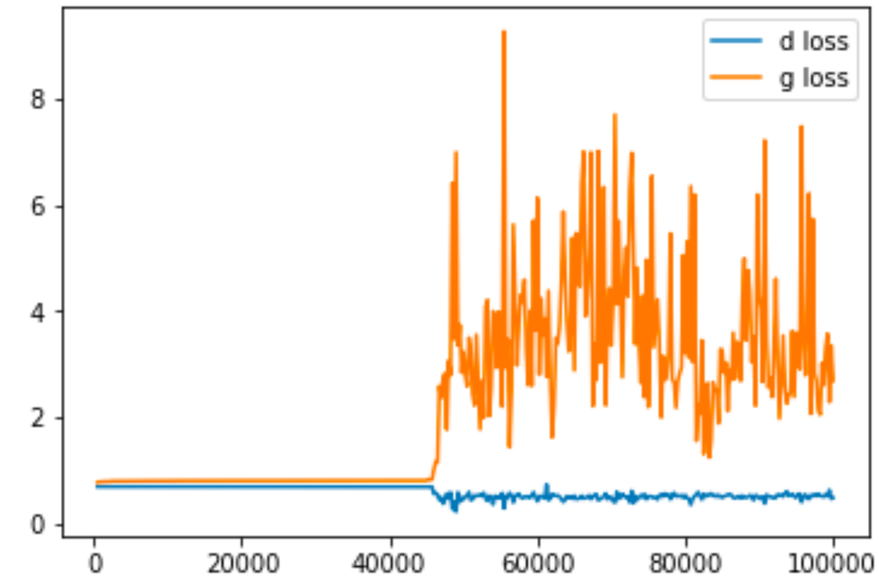
RECO→ RECO GAN

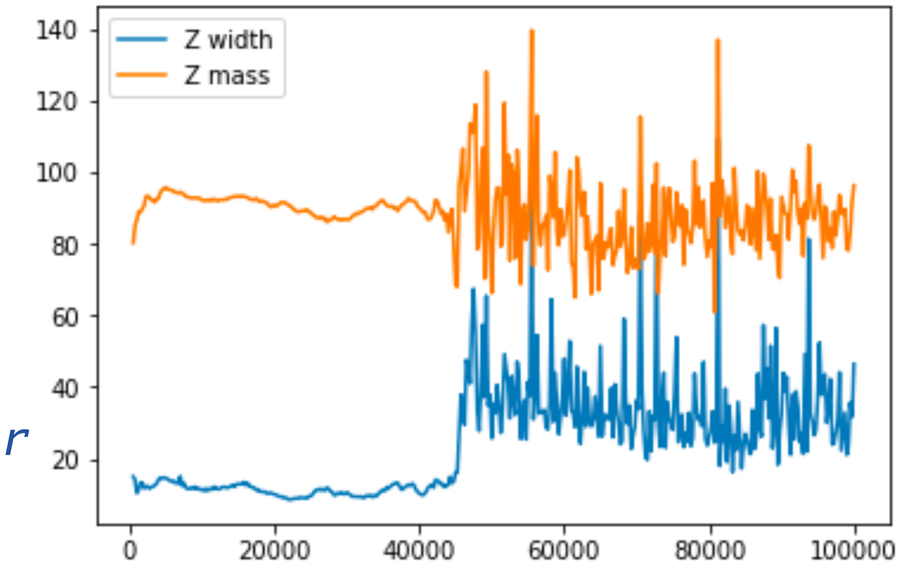
- *We consider a classic GAN setup as baseline (also tried wGAN)*
  - *Train a generator an discriminator in an adversarial fashion*
  - *Add regression of $m_{\ell\ell}$ to the generator cost function, in order to stabilize the training*
- *Implemented in keras+TF*
- *Running on server mounting GTX1080 cards + CSCS Piz Daint (project cn01)*



Training

- *Loss function:* ***cross entropy + c · mse($m_{\ell\ell}$)***
  - *fixed c = 0.01*
- *Dataset size: 2M events*
- *100K epochs / 512 events per batch*
- *Multiple trainings of the same model with randomized starting point, to minimise dependency on initial conditions*
- ***(as normal with GANs) training quite unstable and wGAN didn't really help***
  - *Cannot use the loss function itself as a guide to the best model*
  - *instead, work on defining a generator quality assessment based on statistics tests*





Generator quality

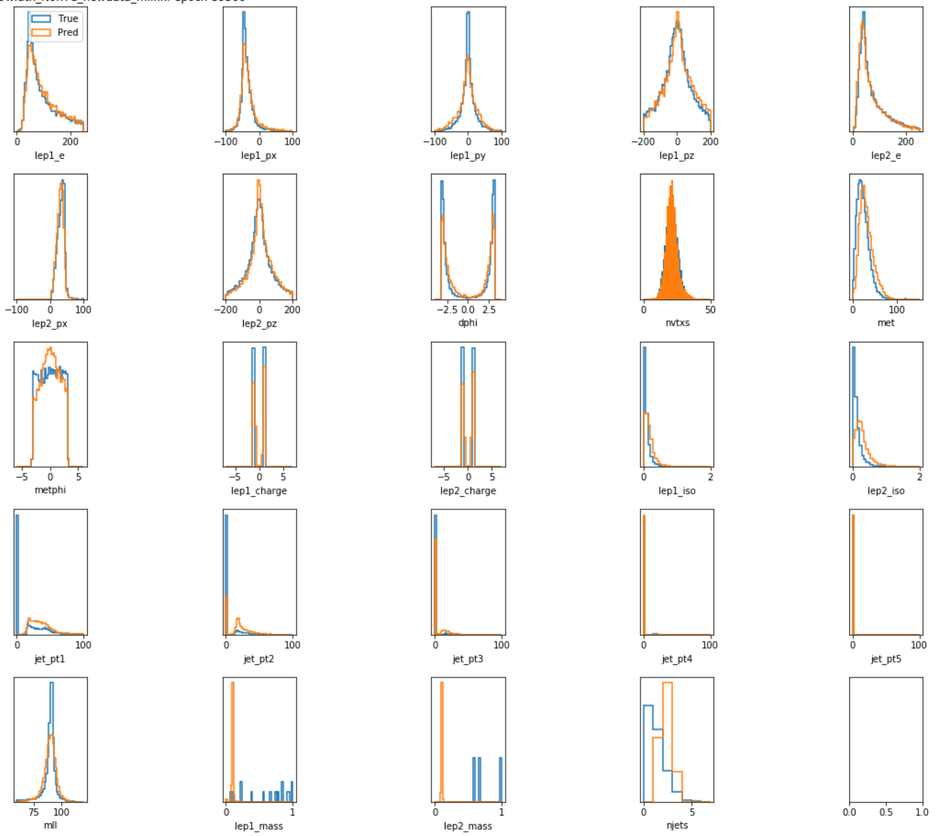
- *Define global- and feature-specific quality tests*
  - Statistics Score (SS) $= \frac{1}{19^2}\sum_{i,j}^{19}\frac{\Delta\sigma_{ij}}{\sigma_{ij}^{\text{real}}} + \frac{1}{19}\sum_{i}^{19}\frac{\Delta\mu_i}{\mu_i^{\text{real}}}$.
    - Roughly the normalized sum of the difference between covariances and averages.
    - $\Delta\sigma_{ij}$ = difference between the covariance matrix elements in the real and generated data
    - $\sigma_{ij}^{\text{real}}$ = covariance matrix element for real data
    - $\Delta\mu_i$ = difference between the average value of the ith variable in the real and generated data
    - μ_i^{real} = average value of the ith variable in the real data
  - MLLKS = Kolmogorov-Smirnov test statistic for the MLL
  - MetPhiKS = Kolmogorov-Smirnov test statistic for the MET-ϕ
  - LepIsoKS = Kolmogorov-Smirnov test statistic for the leading lepton isolation
- *Work in progress: investigating usage of standard pdf-distance definitions (KL divergence, earth-mover distance, etc)*

Results

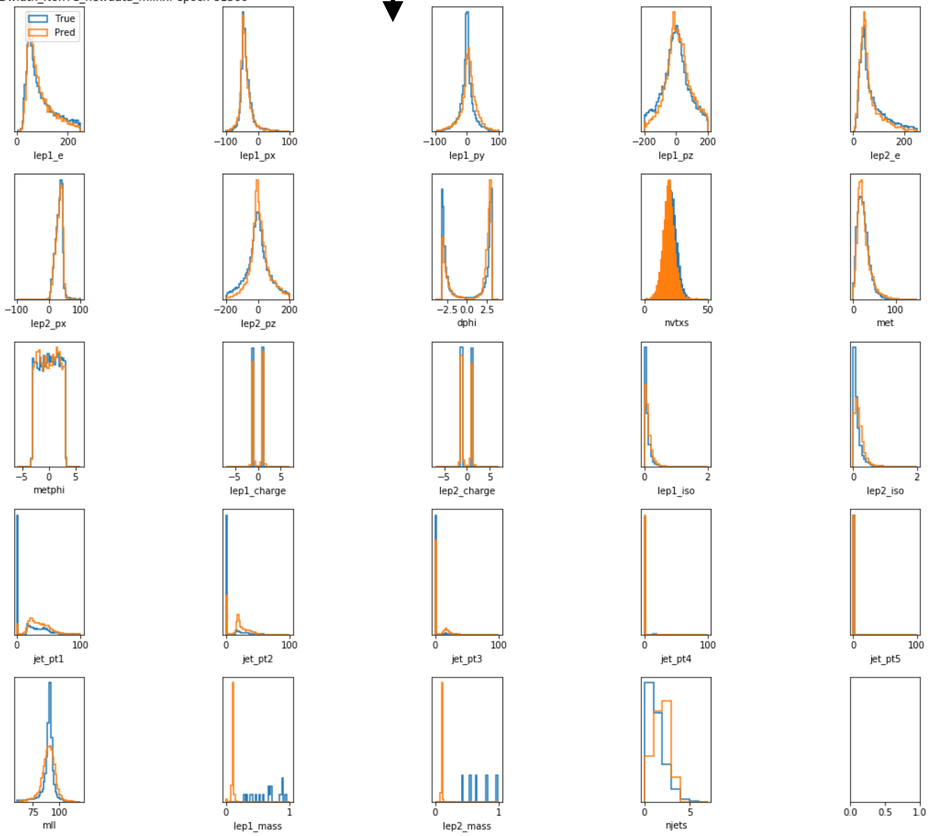
Trial	Epoch	LepIsoRank	MetPhiRank	MLLRank	ScoreRank	StatsScore	MLLKS	MetPhiKS	LepIsoKS	SortKey
12	69500	160	298	59	43	27.826635	0.085580	0.032840	0.236480	560.000000
12	99000	34	166	107	392	47.069988	0.096820	0.024580	0.188500	699.000000
6	29000	144	309	79	228	40.682663	0.090900	0.033440	0.232180	760.000000
12	53000	571	139	23	67	30.737935	0.075540	0.022300	0.305880	800.000000
7	89500	85	448	88	181	38.685347	0.092740	0.041520	0.214840	802.000000
12	81500	284	74	306	154	37.152292	0.122060	0.013740	0.263080	818.000000

Probably Best Overall

tch512_bgbd_mllANDwidth_NonTC_newdata_mllfix: epoch 89500



atch512_bgbd_mllANDwidth_NonTC_newdata_mllfix: epoch 81500

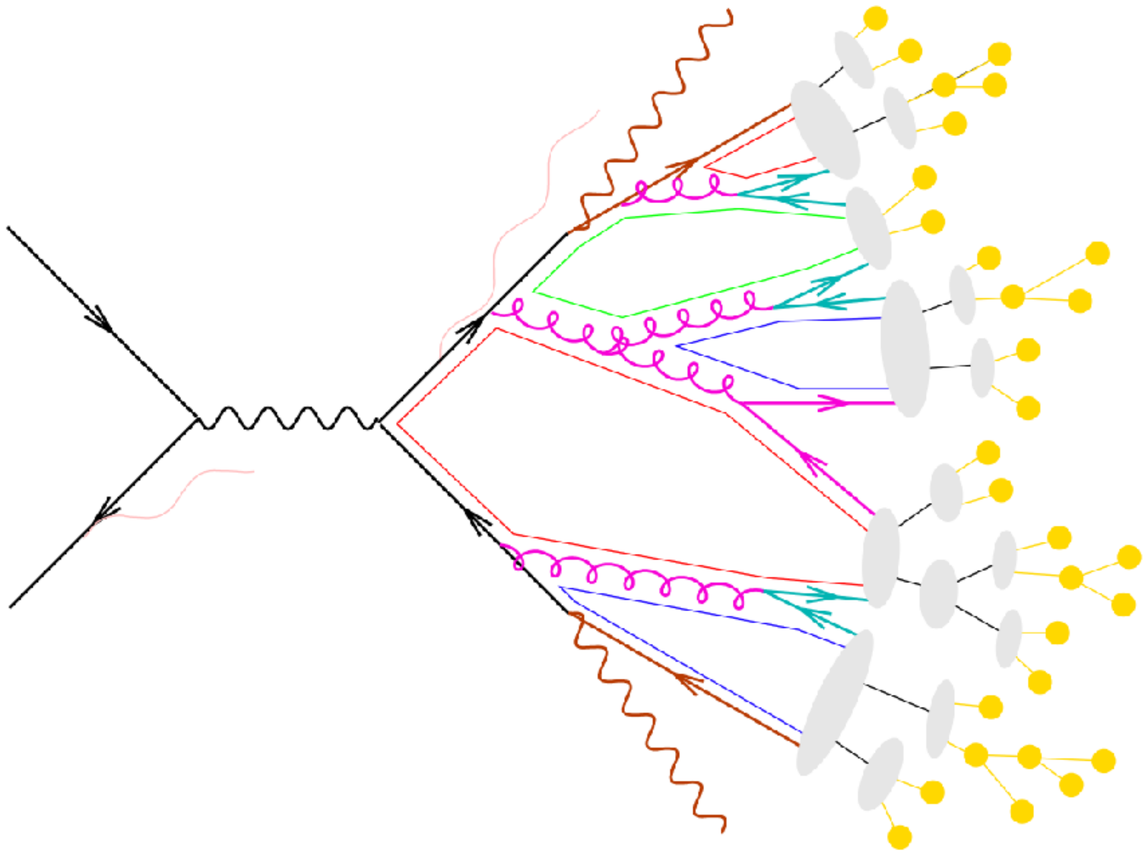


Recurrent Neural Networks

- ***A jet is made putting togethers particles in a "recombination algorithm": attempt to reconstruct the jet showering backward***

- *This jet-dependent structure can be used as input to a RNN, which is trained to map the jet into a q-dimensional array*

- *This 'standardised' input can then be used as input to a traditional layer, e.g to solve a jet tagging problem*

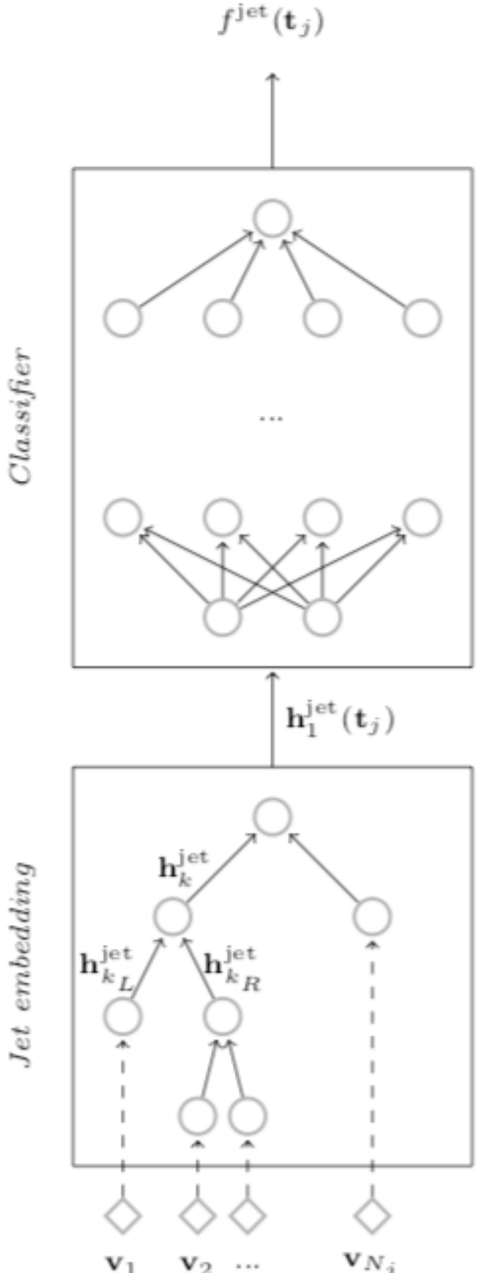


Recurrent Neural Networks

- *A jet is made putting togethers particles in a "recombination algorithm": attempt to reconstruct the jet showering backward*

- ***This jet-dependent structure can be used as input to a RNN, which is trained to map the jet into a q-dimensional array***

- *This 'standardised' input can then be used as input to a traditional layer, e.g to solve a jet tagging problem*



Recurrent Neural Networks

- *A jet is made putting togethers particles in a "recombination algorithm": attempt to reconstruct the jet showering backward*
- *This jet-dependent structure can be used as input to a RNN, which is trained to map the jet into a q-dimensional array*
- ***This 'standardised' input can then be used as input to a traditional layer, e.g to solve a jet tagging problem***