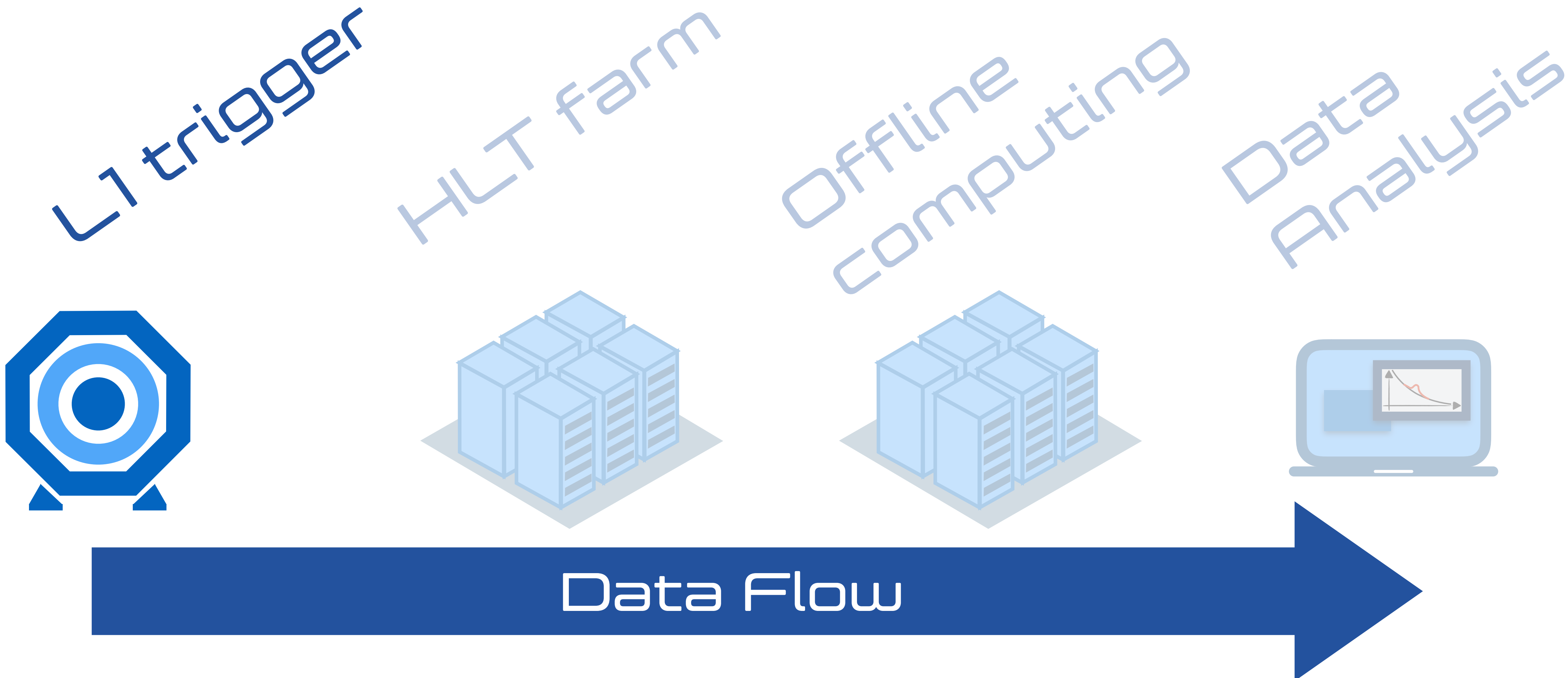
Machine Learning for (collision-event) reconstruction

Maurizio Pierini

Outline

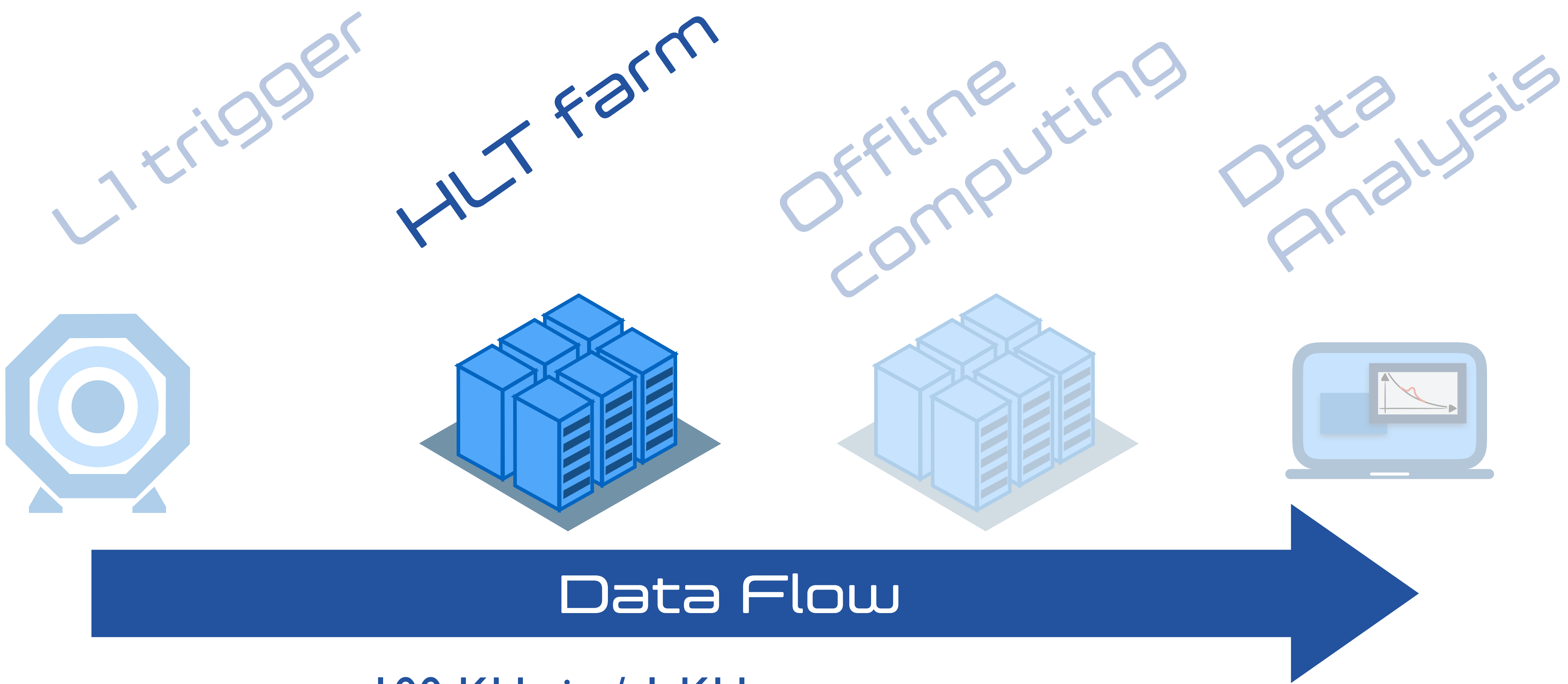
- Four HEP environments & their ML use cases
- What we are currently doing
- Going deep: proofs of principle
- First "production ready" DL algorithms
- Future ahead
  - New challenges
  - New instruments
  - New solutions

Four HEP environments



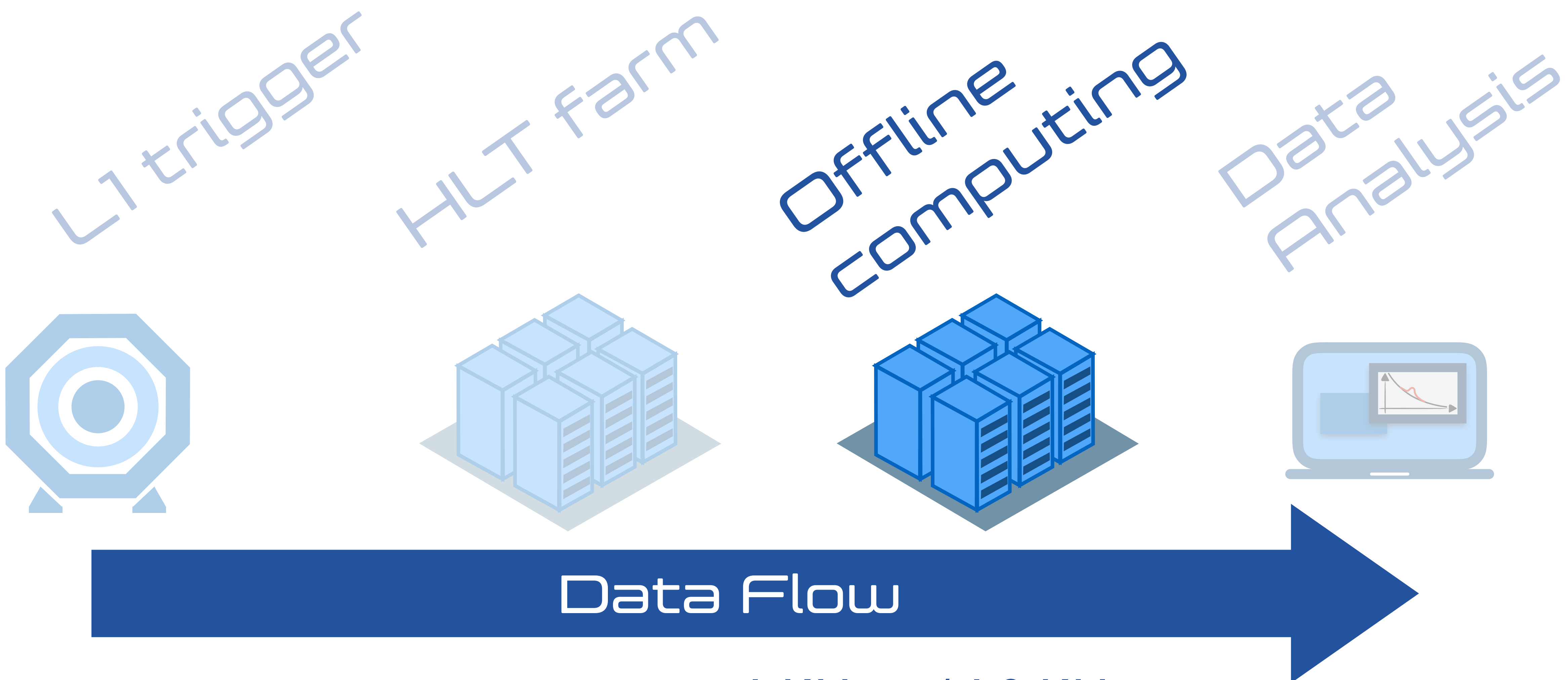
- 40 MHz in / 100 KHz out
- ~ 500 KB / event
- Processing time: ~10 µs
- Based on coarse local reconstructions
- FPGAs / Hardware implemented

Four HEP environments



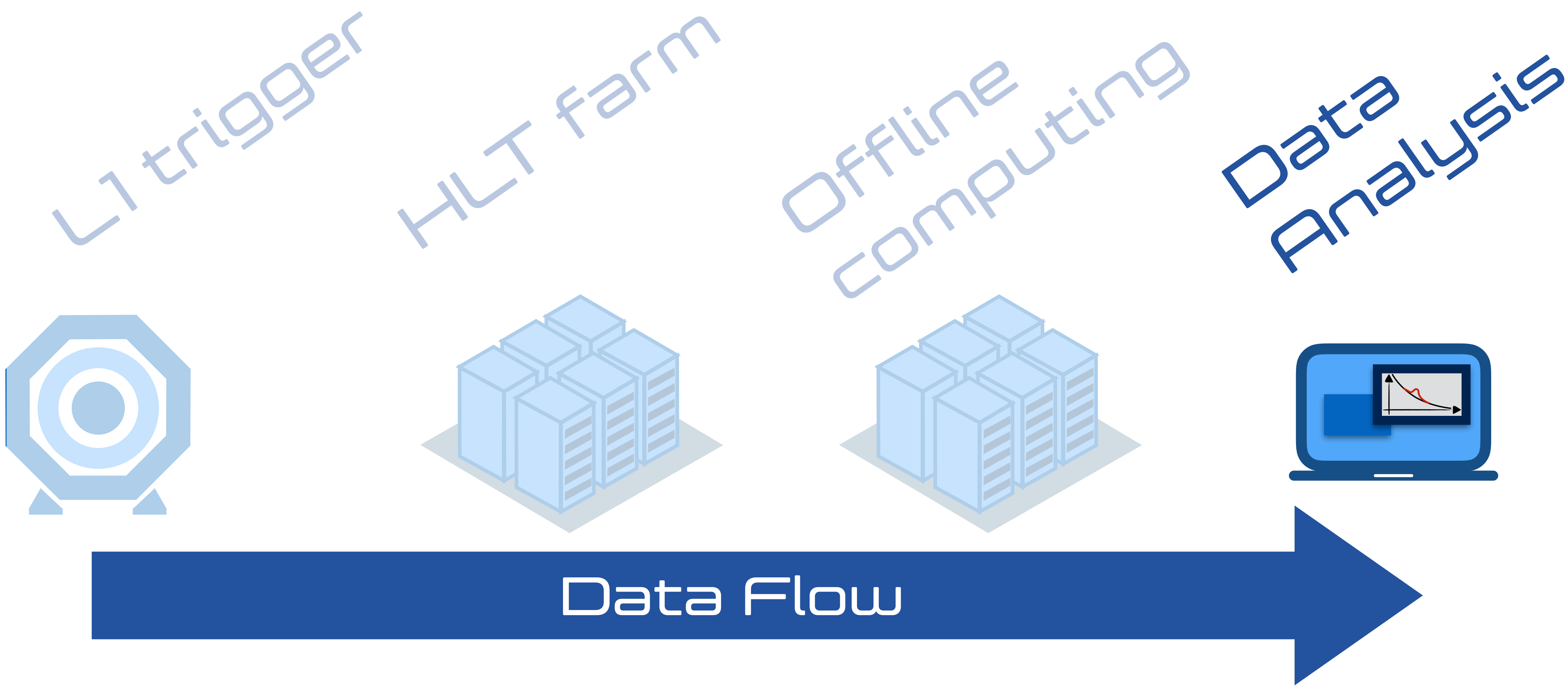
- 100 KHz in / 1 KHz out
- ~ 500 KB / event
- Processing time: ~30 ms
- Based on simplified global reconstructions
- Software implemented on CPUs

Four HEP environments



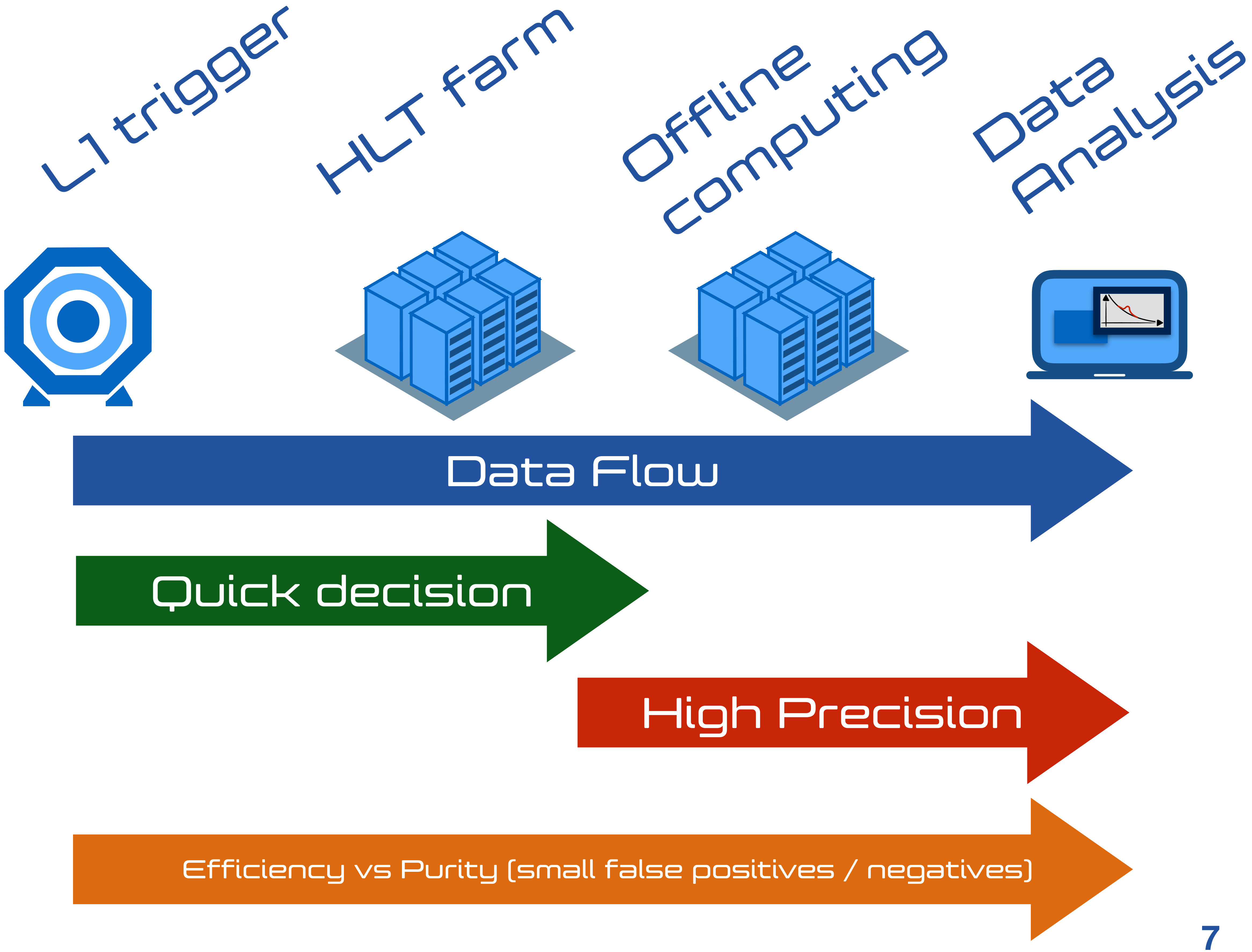
- 1 KHz in / 1.2 KHz out
- ~ 1 MB / 200 KB / 30 KB per event
- Processing time: ~20 s
- Based on accurate global reconstructions
- Software implemented on CPUs

Four HEP environments



- Up to ~ 500 Hz In / 100-1000 events out
- <30 KB per event
- Processing time irrelevant
- User-written code + centrally produced selection algorithms

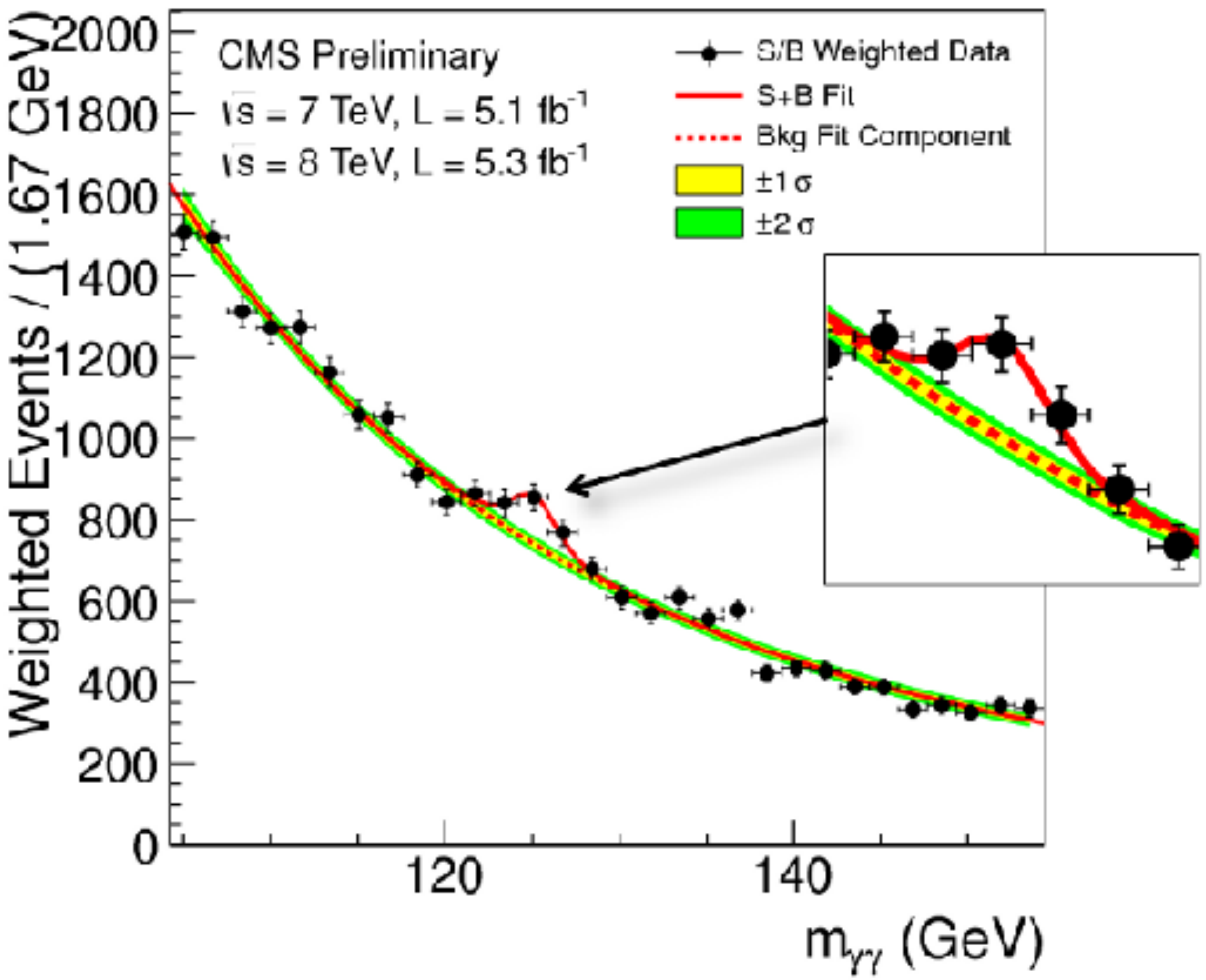
Four HEP environments



What we are doing today

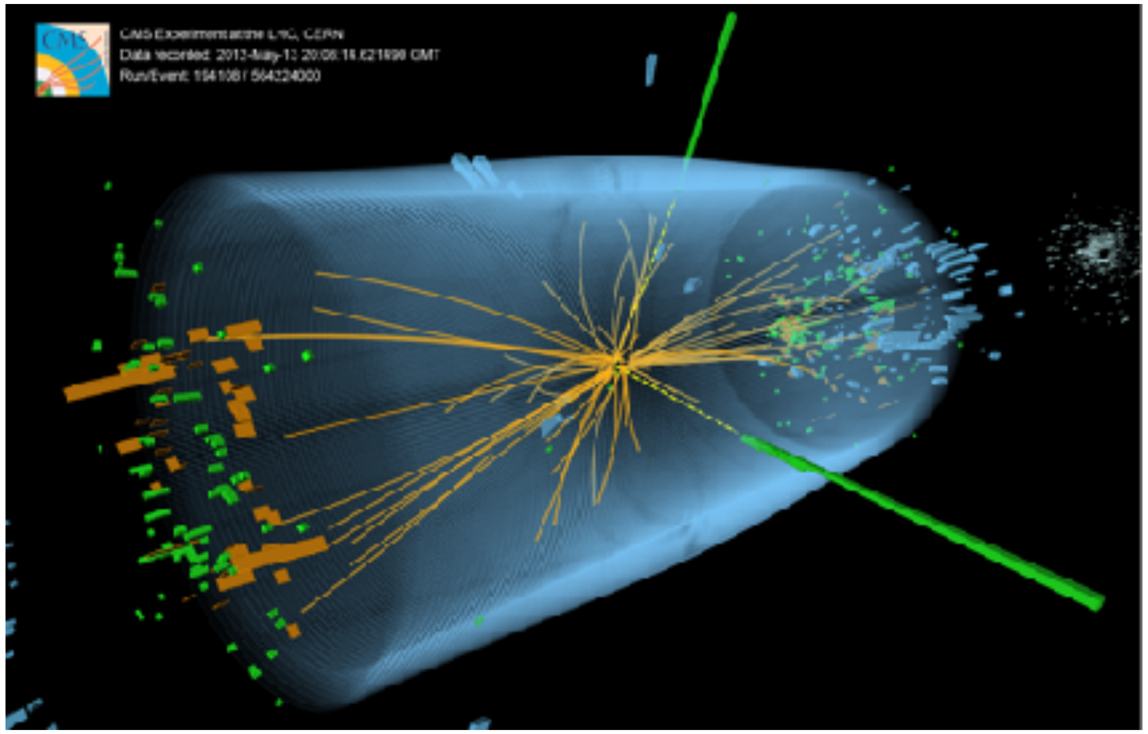
ML applications today

- Traditionally, object reco/identification based on ML
  - regressions to improve energy measurement
  - classification to suppress “fakes” (i.e false positives)

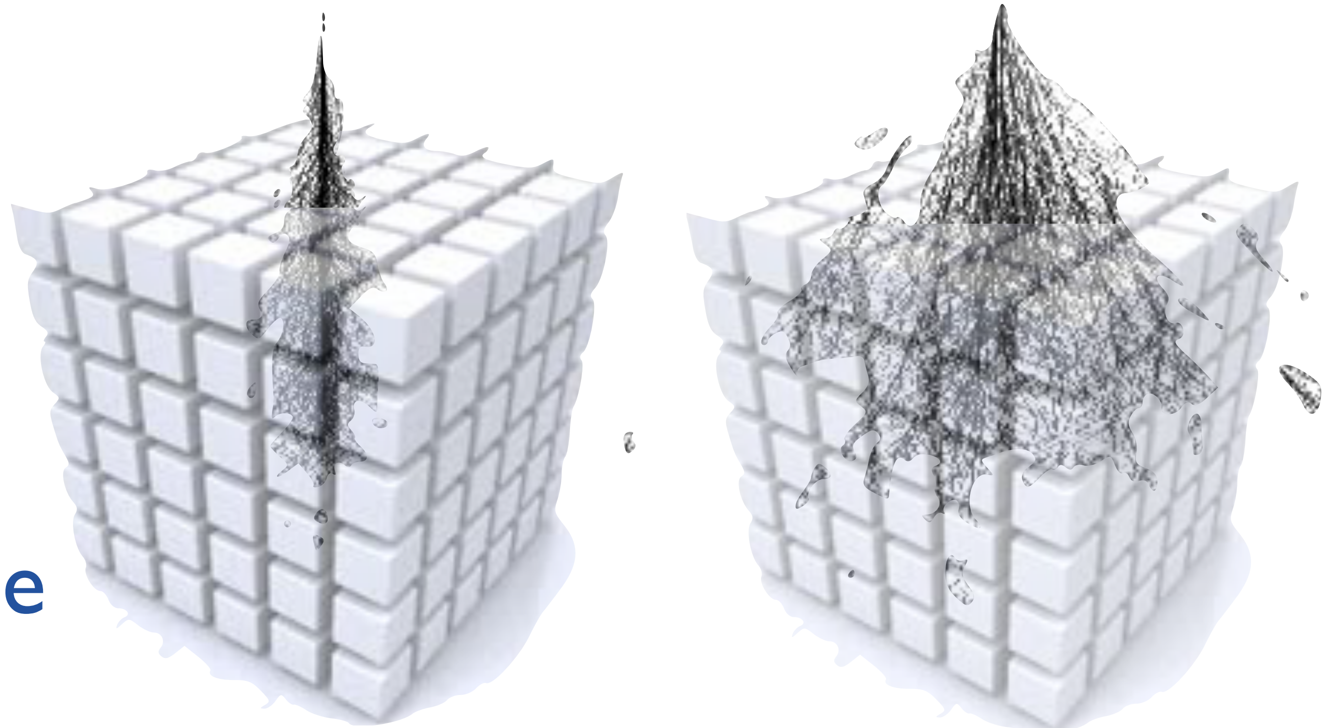


- Nowadays, mainly based on BDTs
- Crucial ingredient to discover the Higgs boson much earlier than anticipated

Photon identification

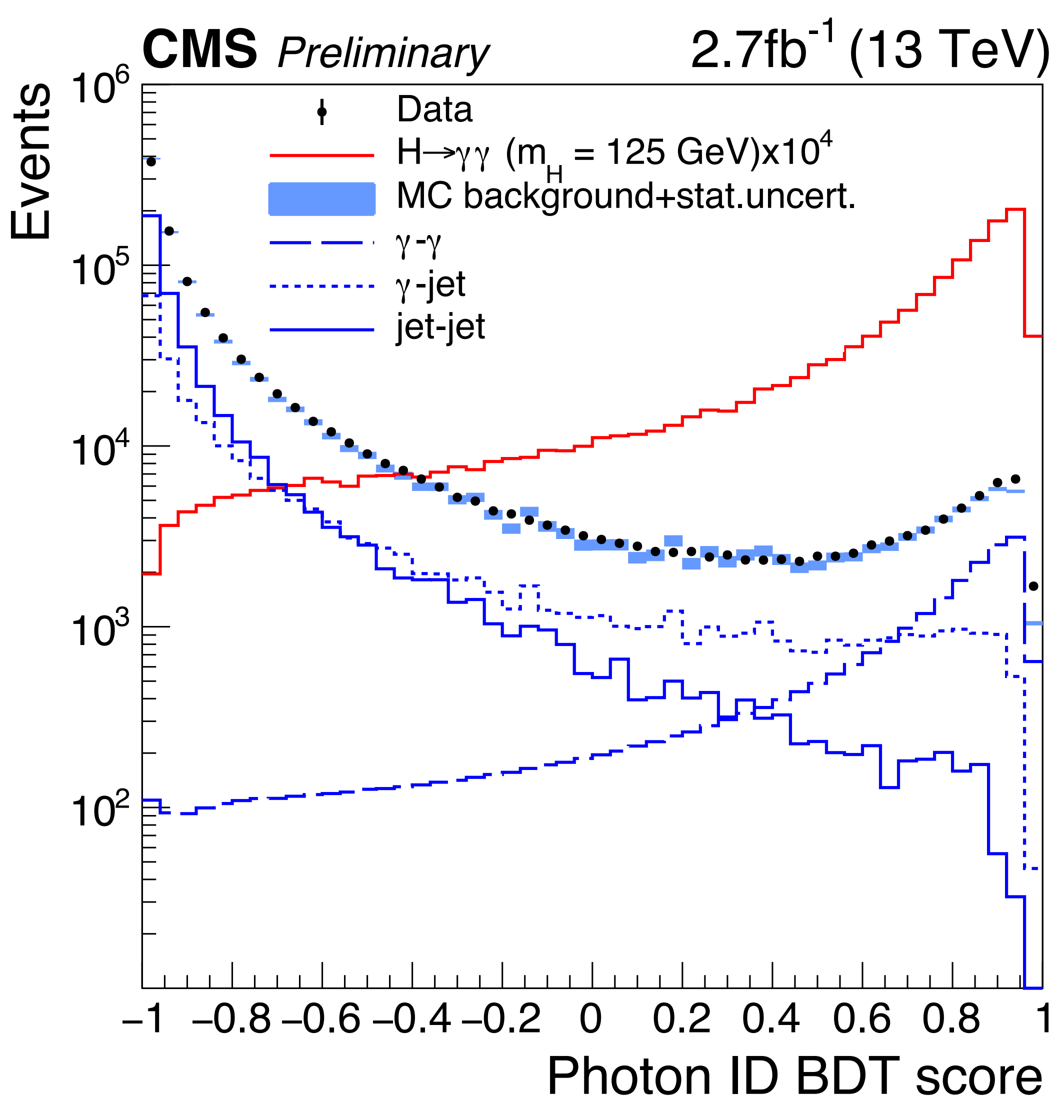


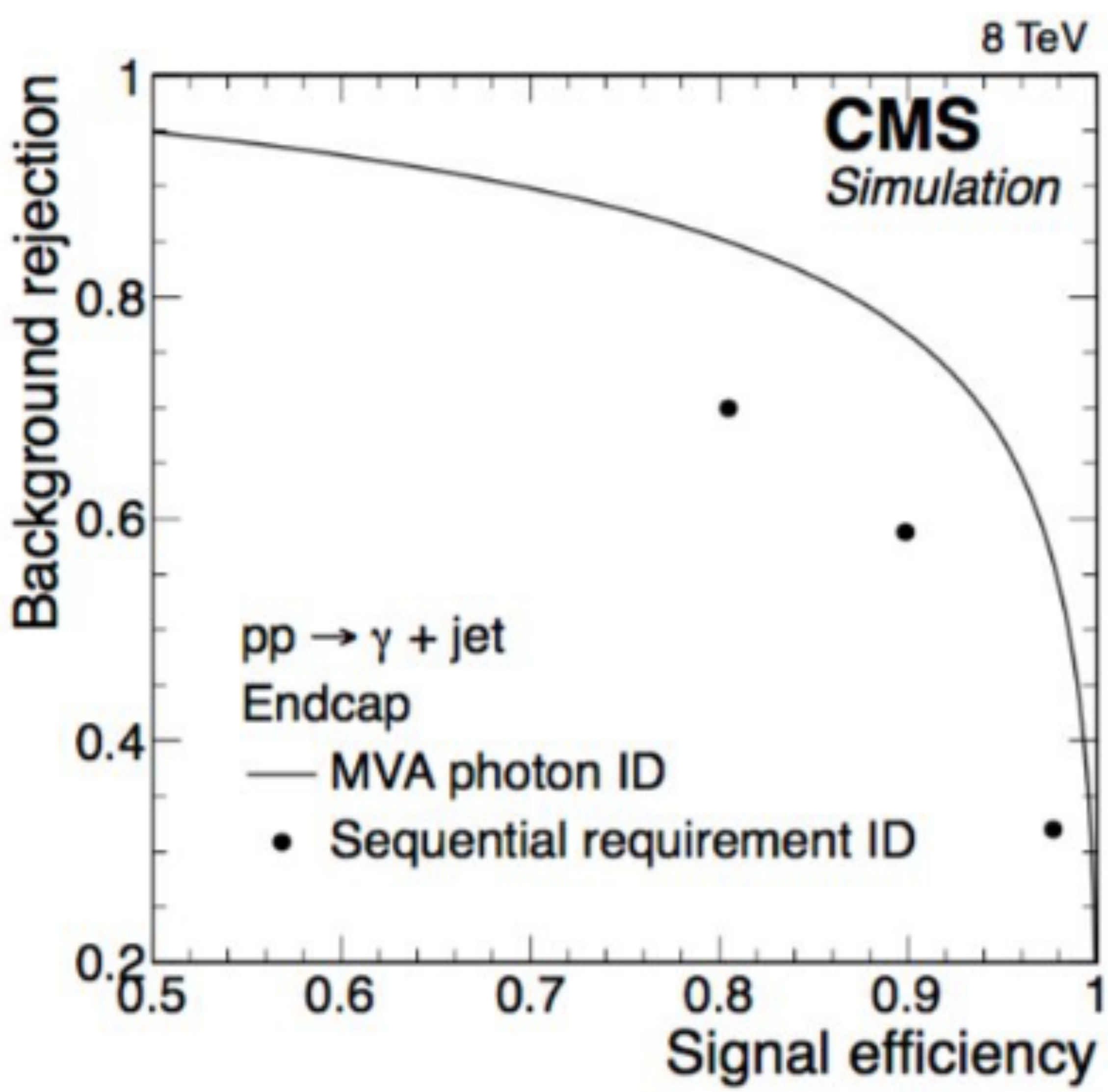
- Photons are complicated to reconstruct
  - signal in calorimeter
  - nothing in the tracker
- Fakes from several sources
  - hadrons in jets
  - π^0-> $\gamma\gamma$
- Main discrimination handle from shape of energy cluster in the calorimeter



Photon identification

- BDT classifier used to separate true from fake photons
  - improvement wrt "cut-based" approach



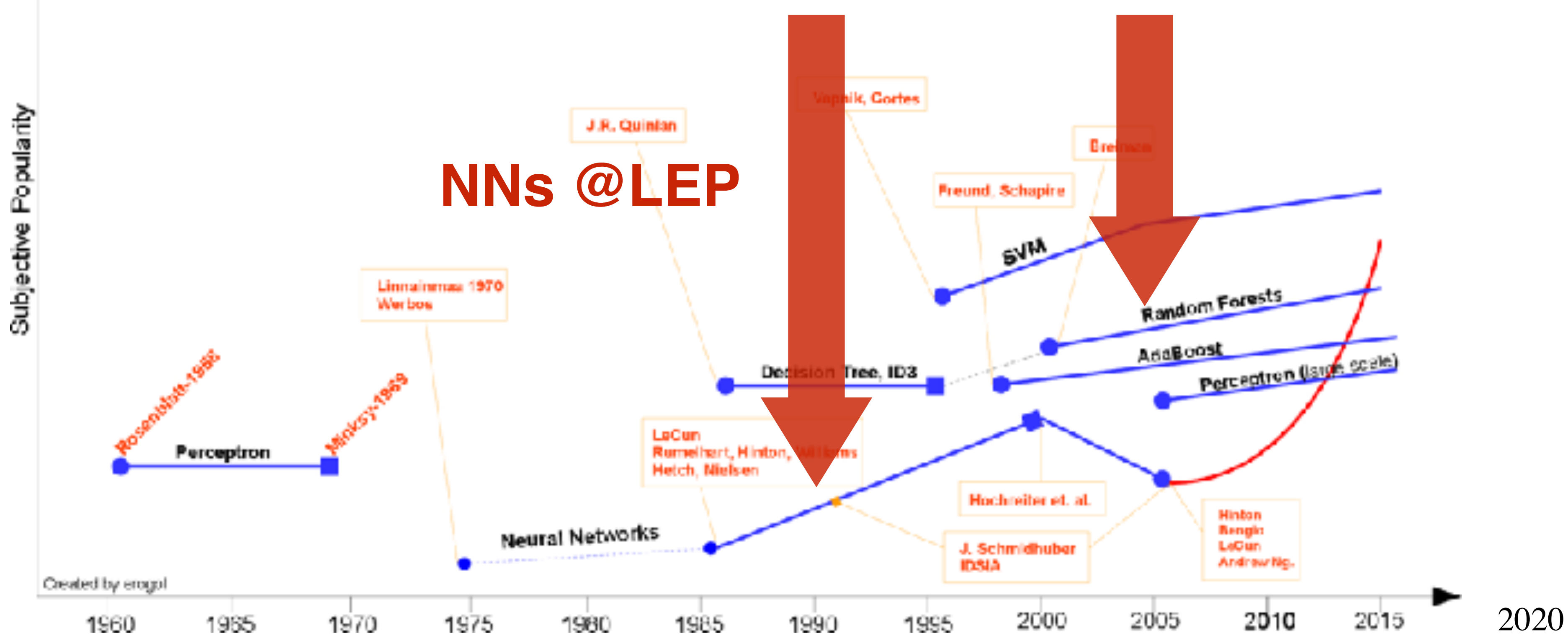


Deep Learning @HEP today

Going Deep: proof of principles

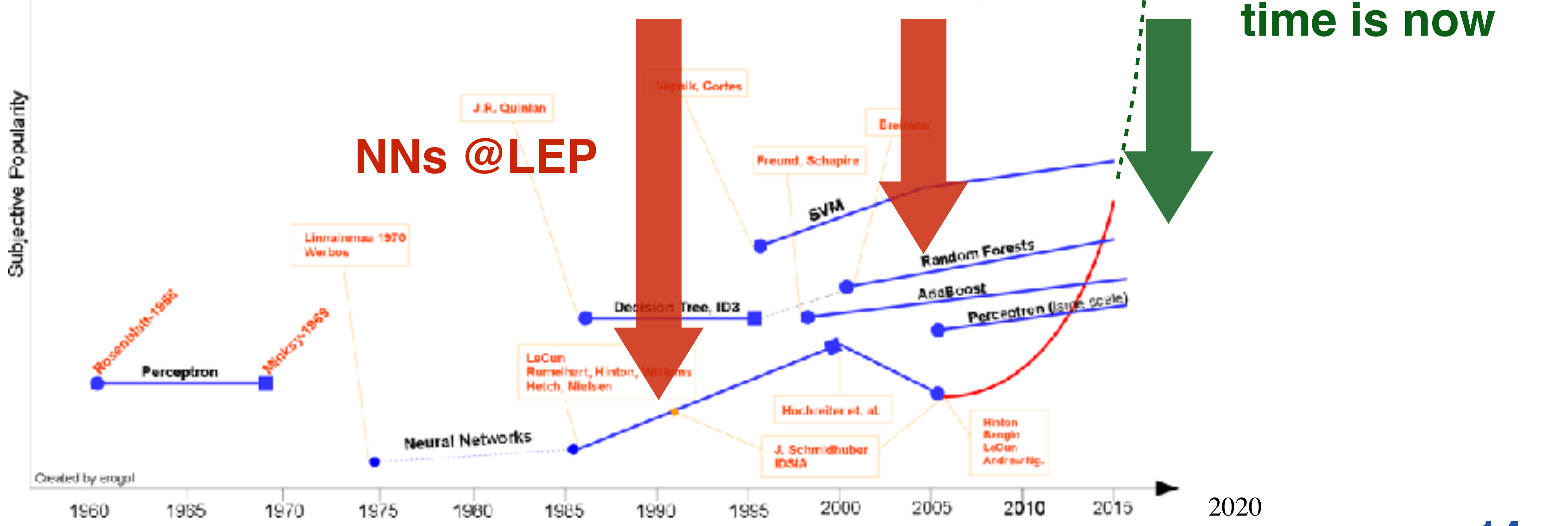
- Many people looking at Deep Learning applications for LHC use cases
  - Majority of applications concern jet identification
    - Deep Shallow Networks
    - Convolutional NNs
    - Recursive NNs
    - GANs for simulation
  - First applications in production
    - b-tagging

BDTs @MiniBooNE, Babar, etc

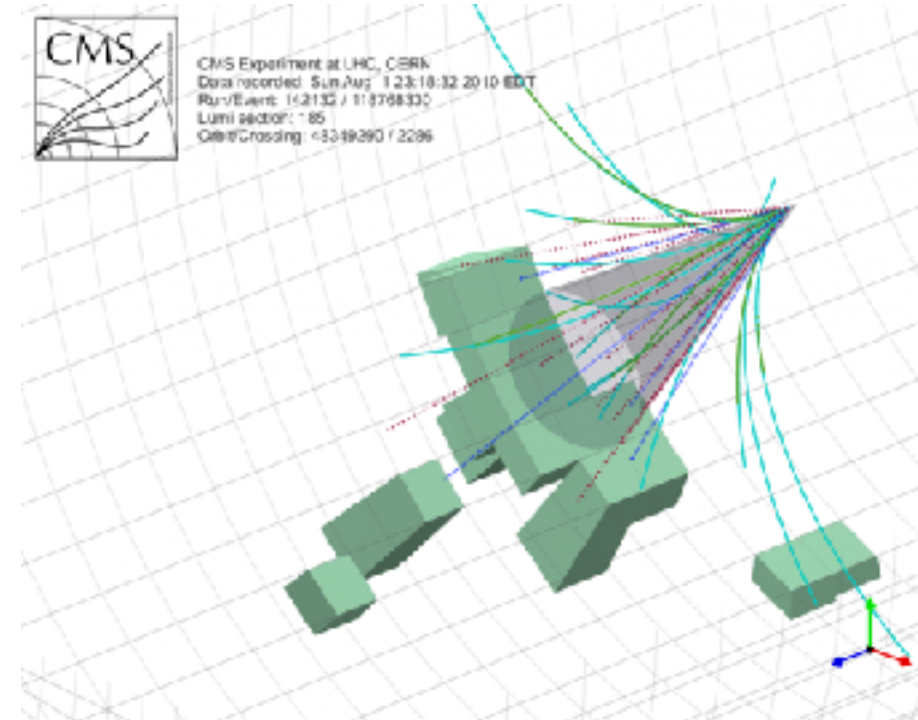


Going Deep: proof of principles

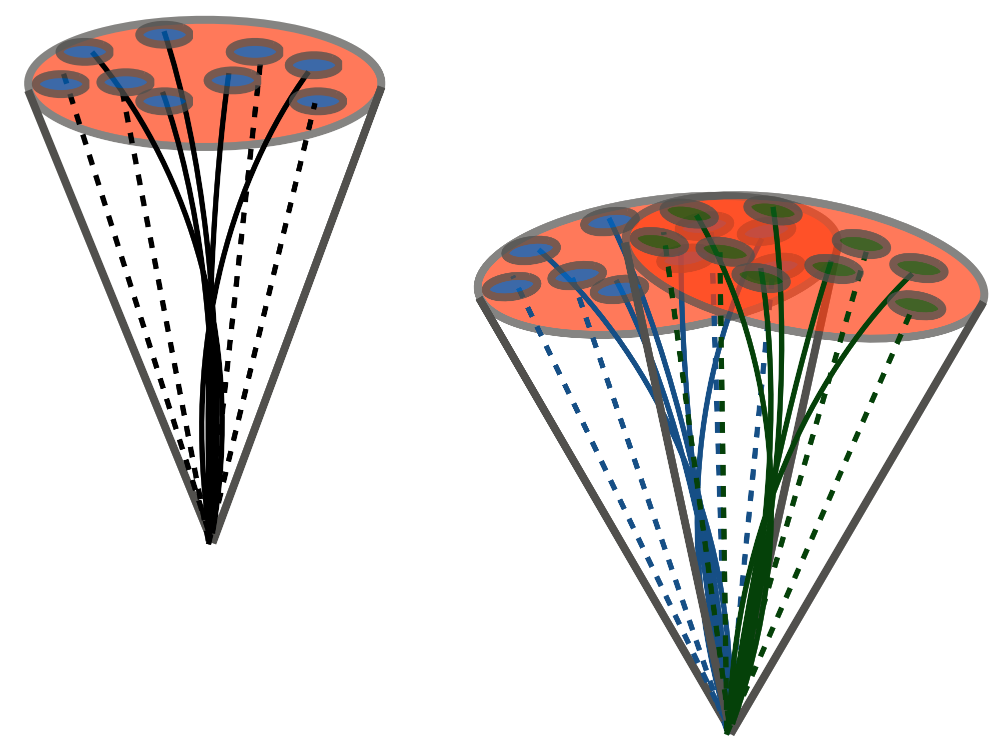
- Many people looking at Deep Learning applications for LHC use cases
  - Majority of applications concern jet identification
    - Deep Shallow Networks
    - Convolutional NNs
    - Recursive NNs
    - GANs for simulation
  - First applications in production
    - b-tagging



Jet ID with ML

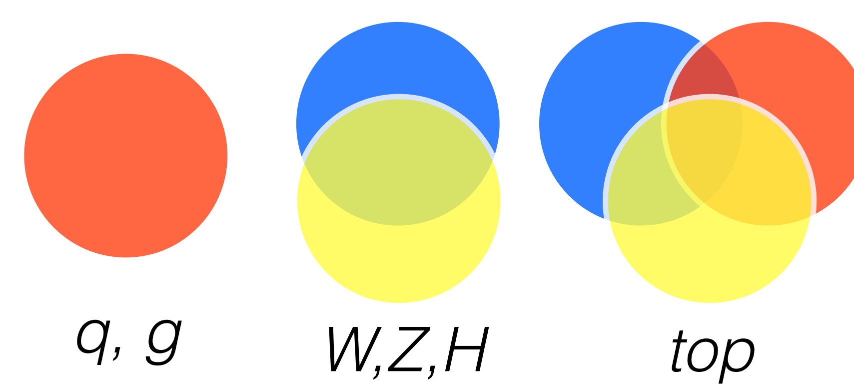


- Jets are cone-like showers of quarks and gluons that produce tens of particles, all close to each other



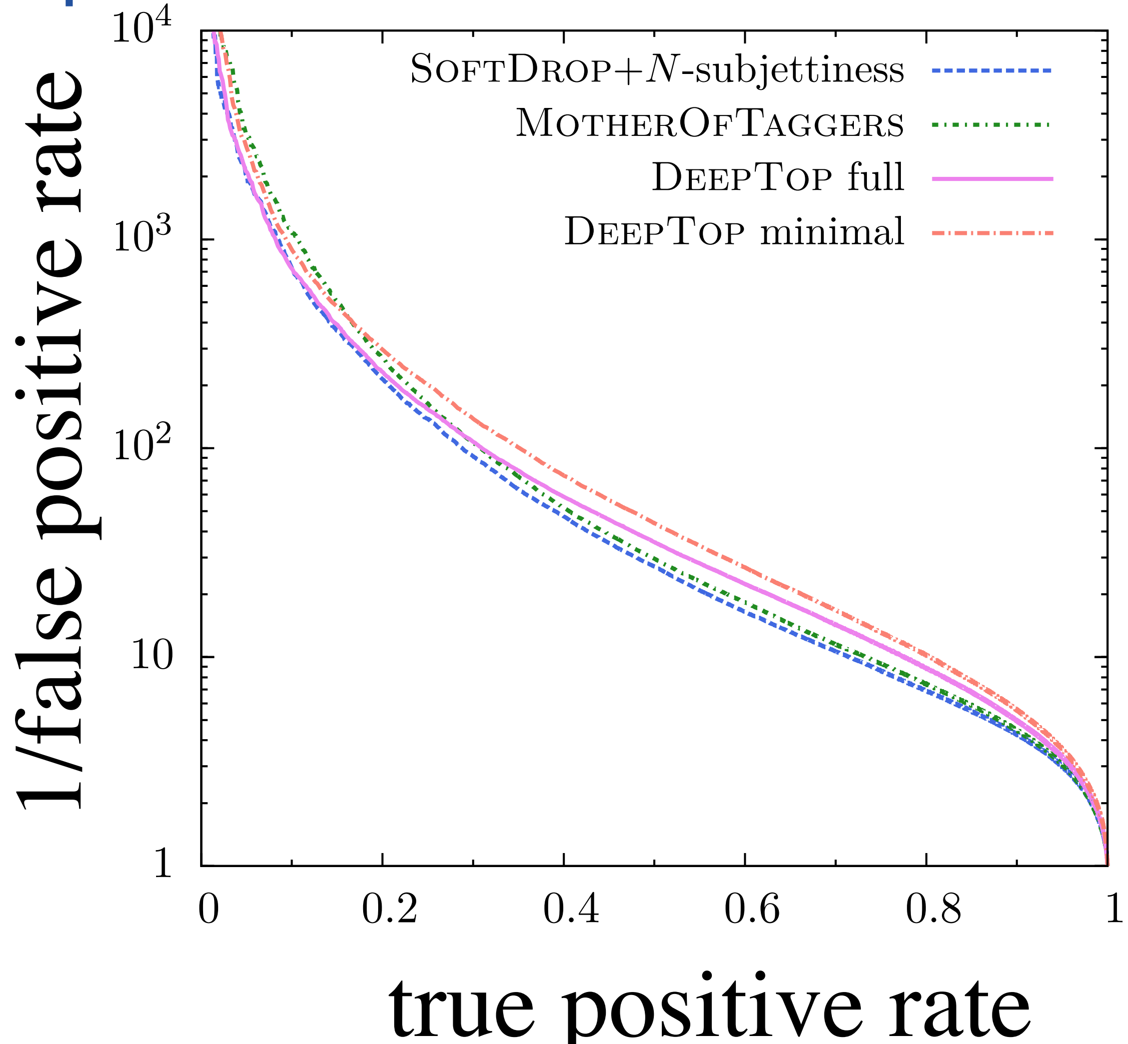
- With large energies (e.g., LHC), jets can also come from H, W, top particles (decaying to jets, which overlap)

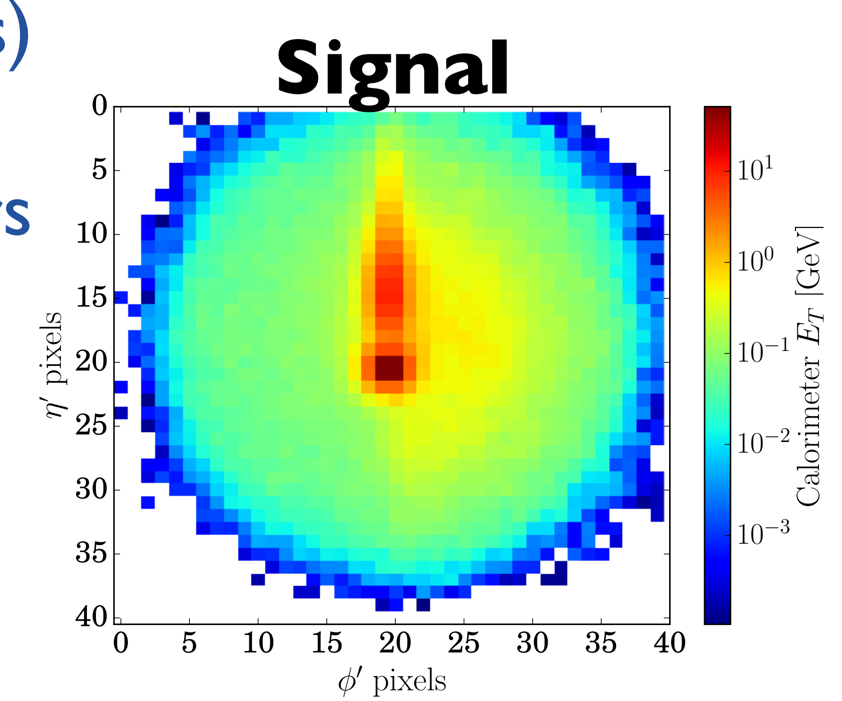
- Several papers in the last two years on DNN solutions to this problem

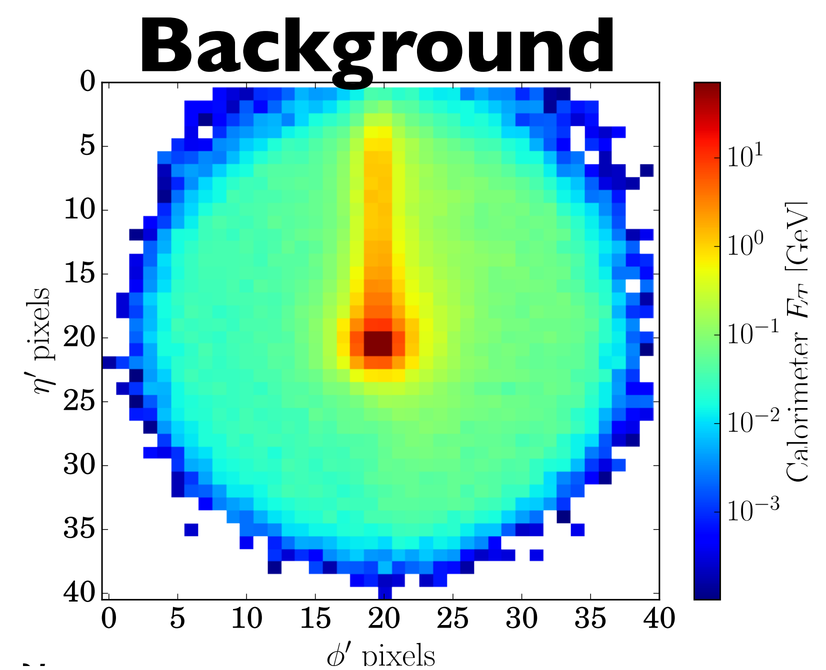


Jet ID with ConvNNs

- Major challenge: irregular detector geometry (vs "regular arrays" assumed in DL applications)
- Jet image processing to "regularise" jet showers and make DL work easier (cantering, rotating, flipping image)
- Good performances on simulated events



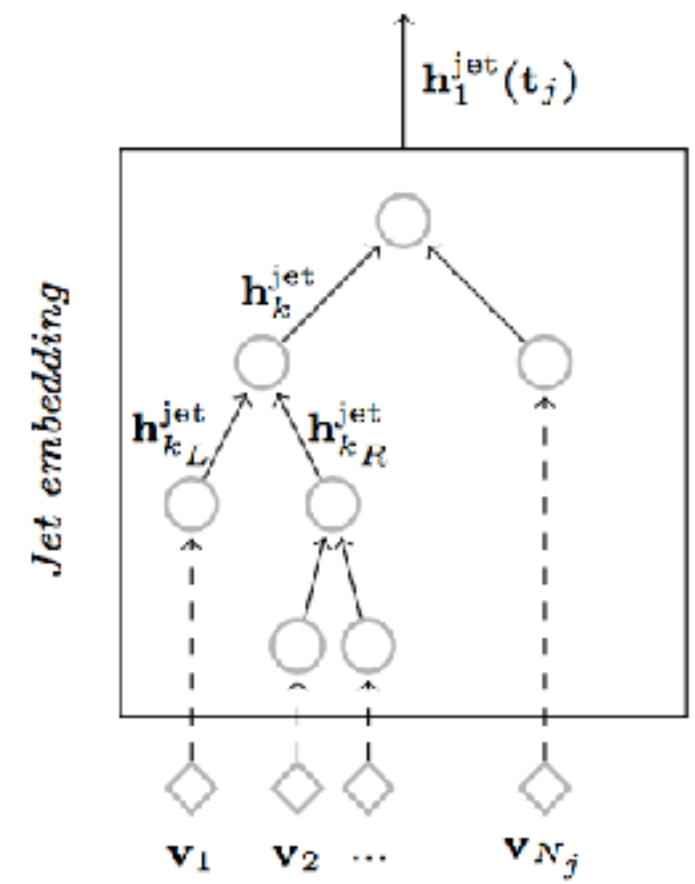


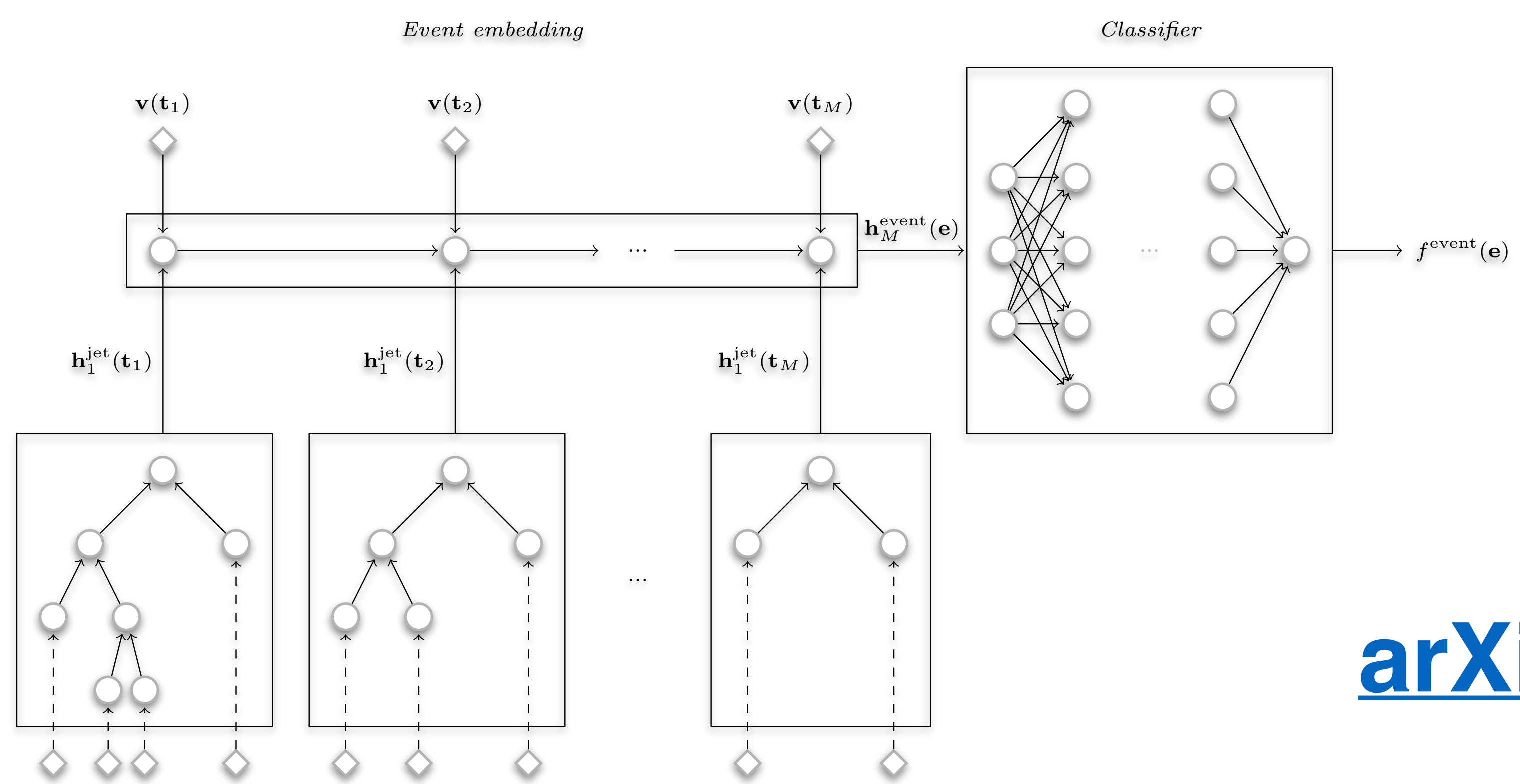


arXiv: 1701.08784

Jet ID with RNNs

- Computing vision techniques assumes regular arrays of "pixels" as input
- Our detector have often irregular geometry
- We reconstruct particles from decor "pixels"
  - DL can take as input directly the variables
- Recursive NNs are ideal for this task
  - natural order provided by jet algorithms
  - variable #particles/jet can be handled

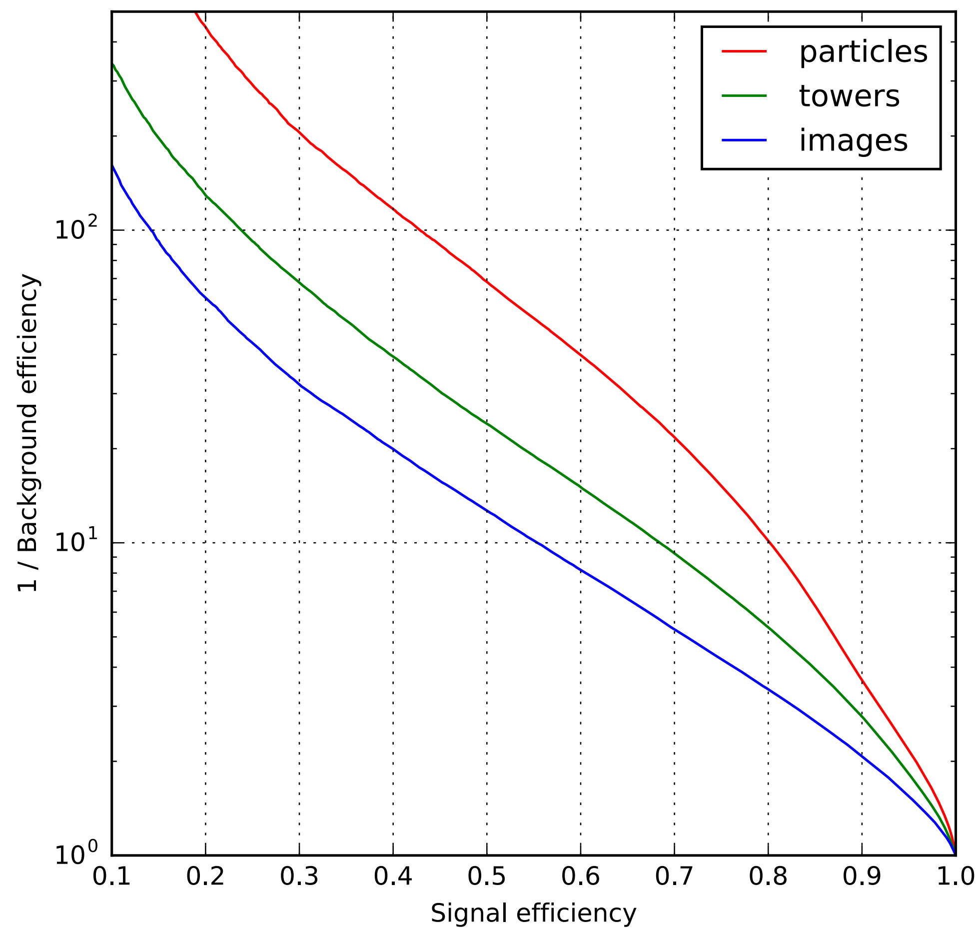


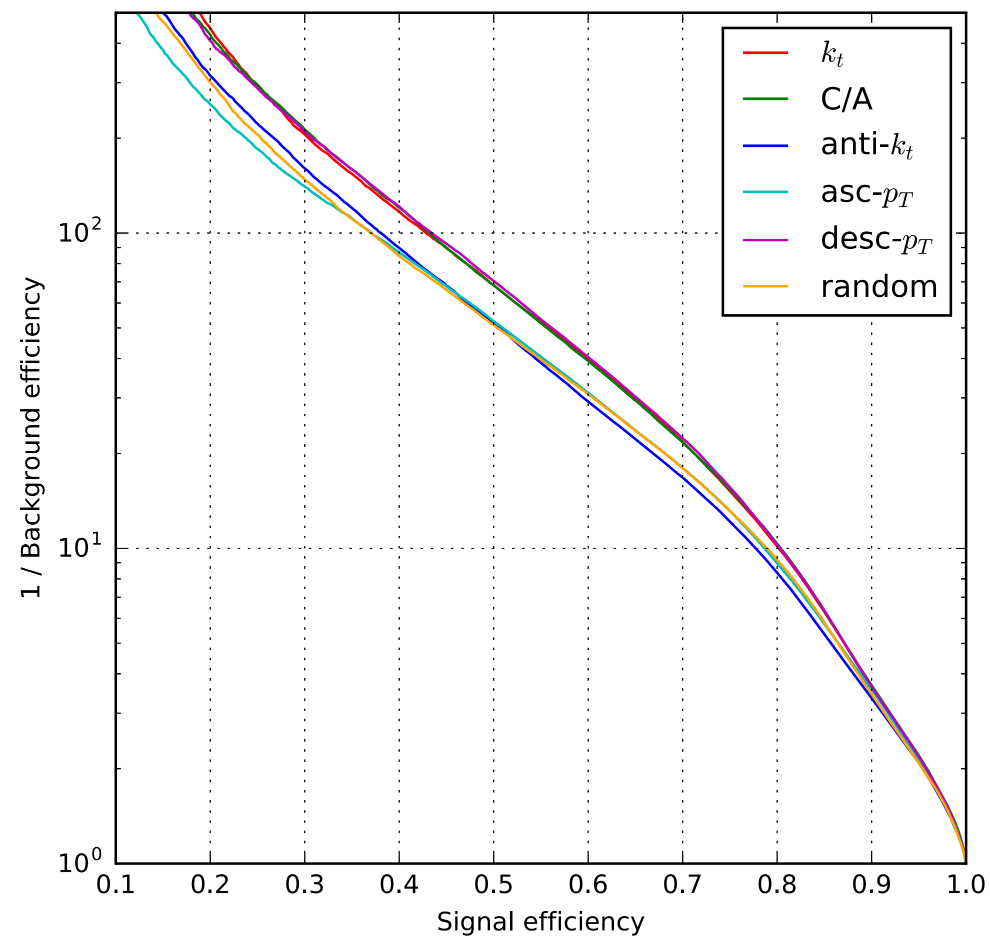


[arXiv:1702.00748](https://arxiv.org/abs/1702.00748)

Jet ID with RNNs

- Better performances using jet constituents rather than jet image (RNN & QCD at work)
- The used jet algorithm matters
- Particles work better than jets (angular resolution matters. That's why you want a granular detector)

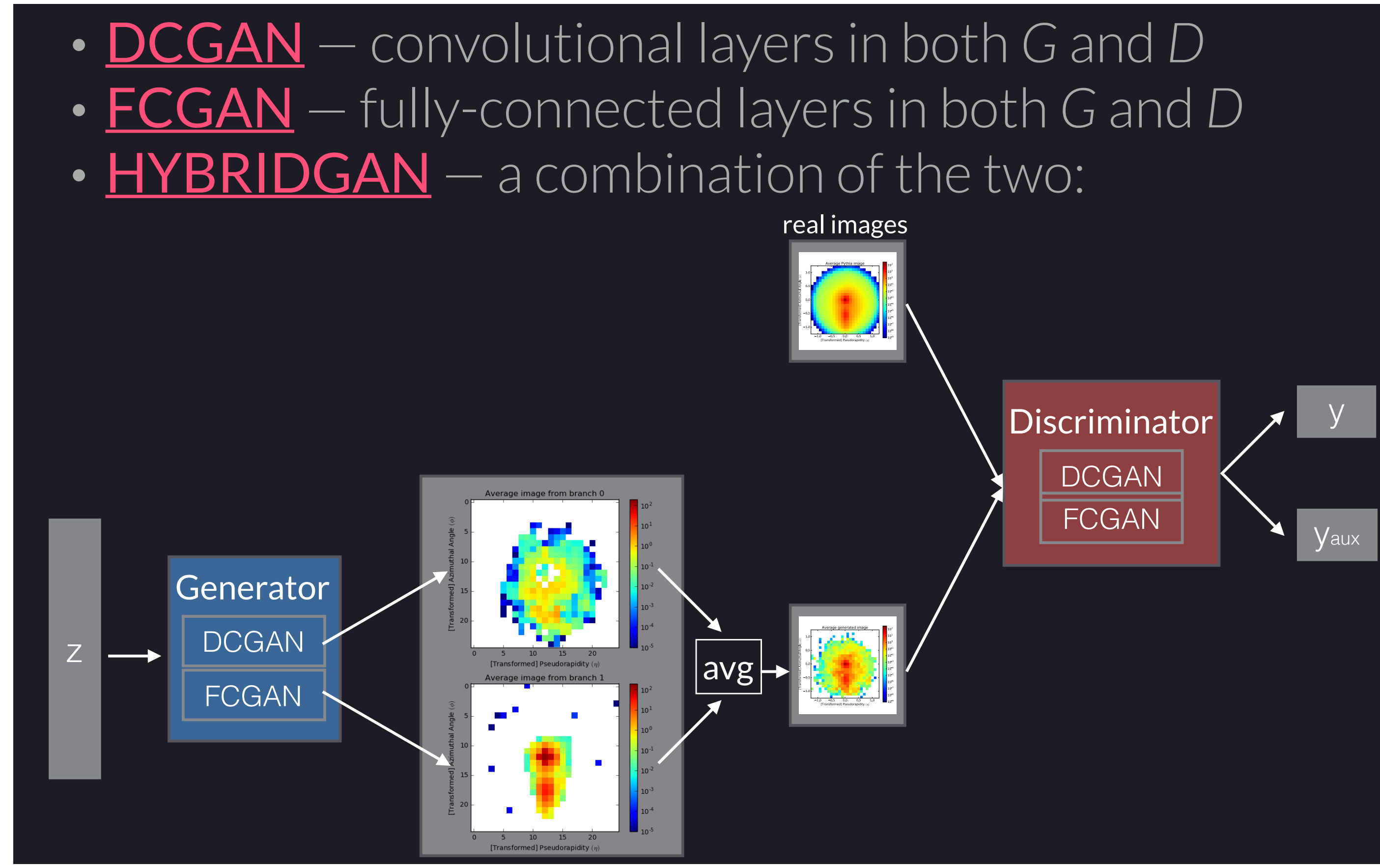


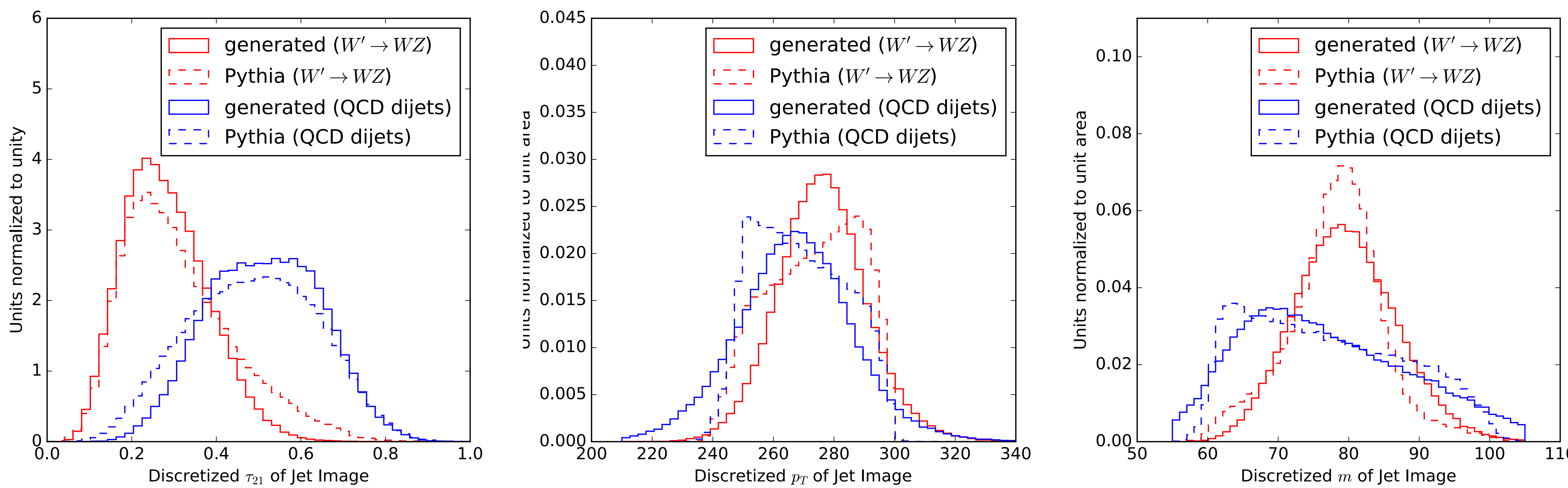


arXiv:1702.00748

GANs for Jets

- First HEP application of GAN
- See Sofia's talk for more on this topic
- Very promising, but there is work to do



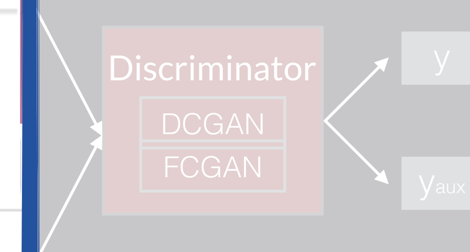


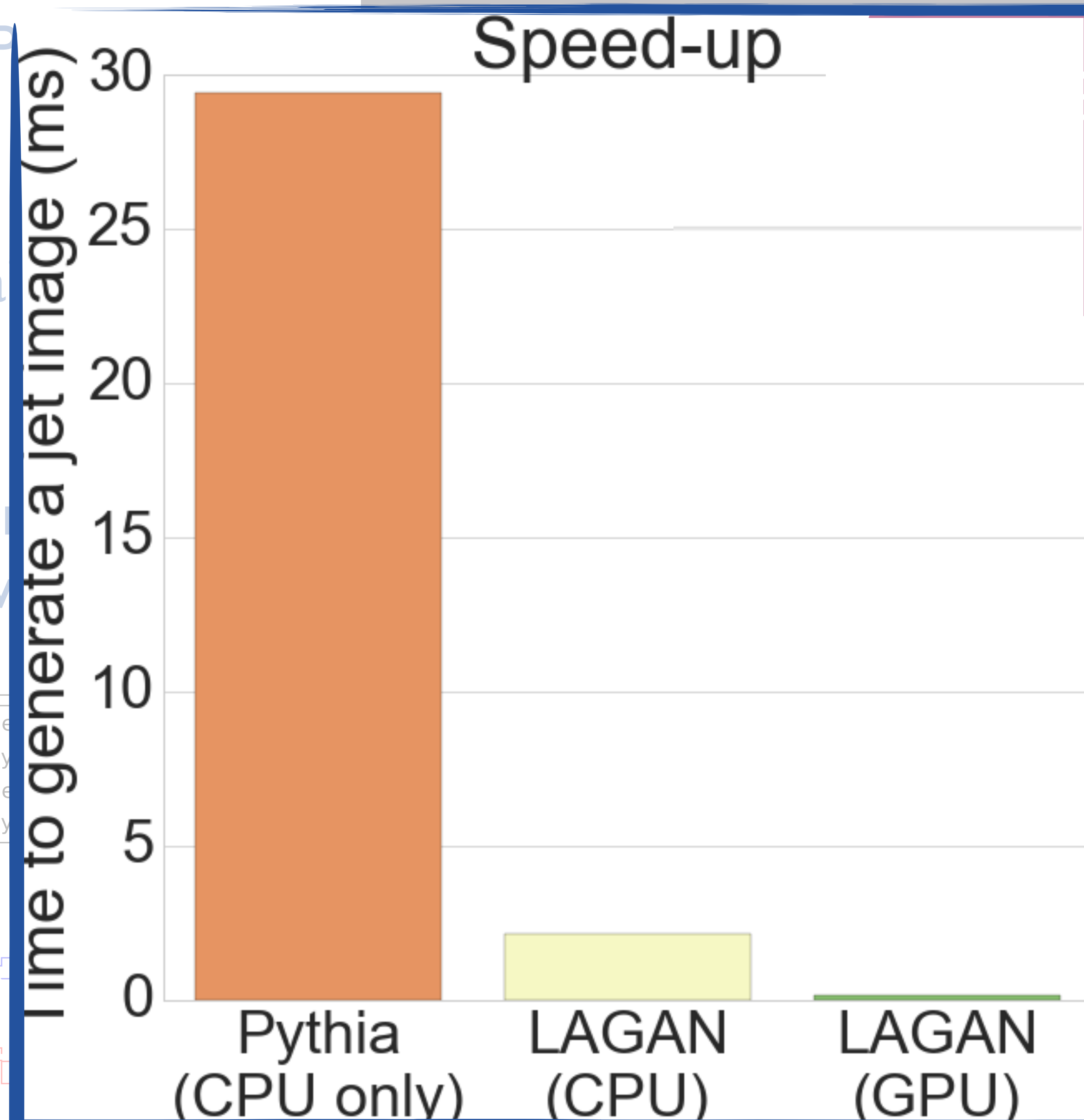
arXiv:1701.05927

GANs for Jets

- First HEP
  of GAN
- See Sofia
  more on
- Very pro
  there is v

• DCGAN — convolutional layers in both G and D
both G and D
e two:

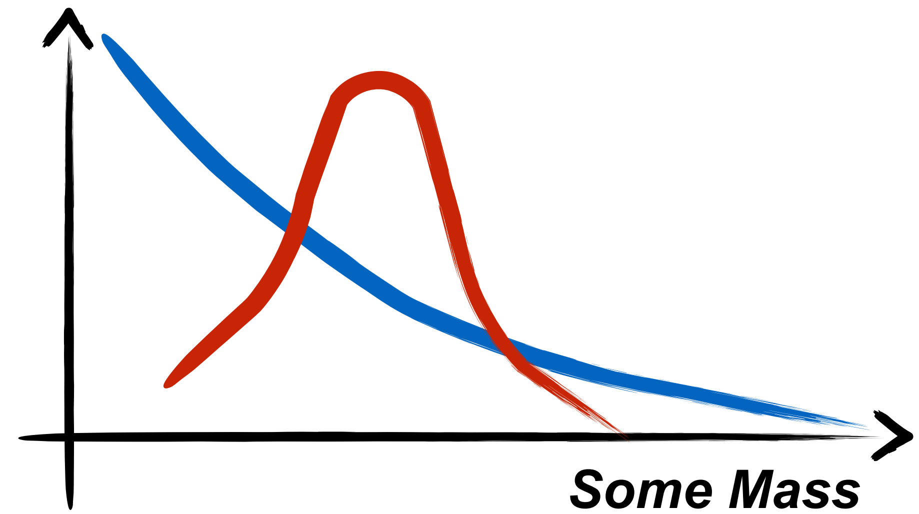




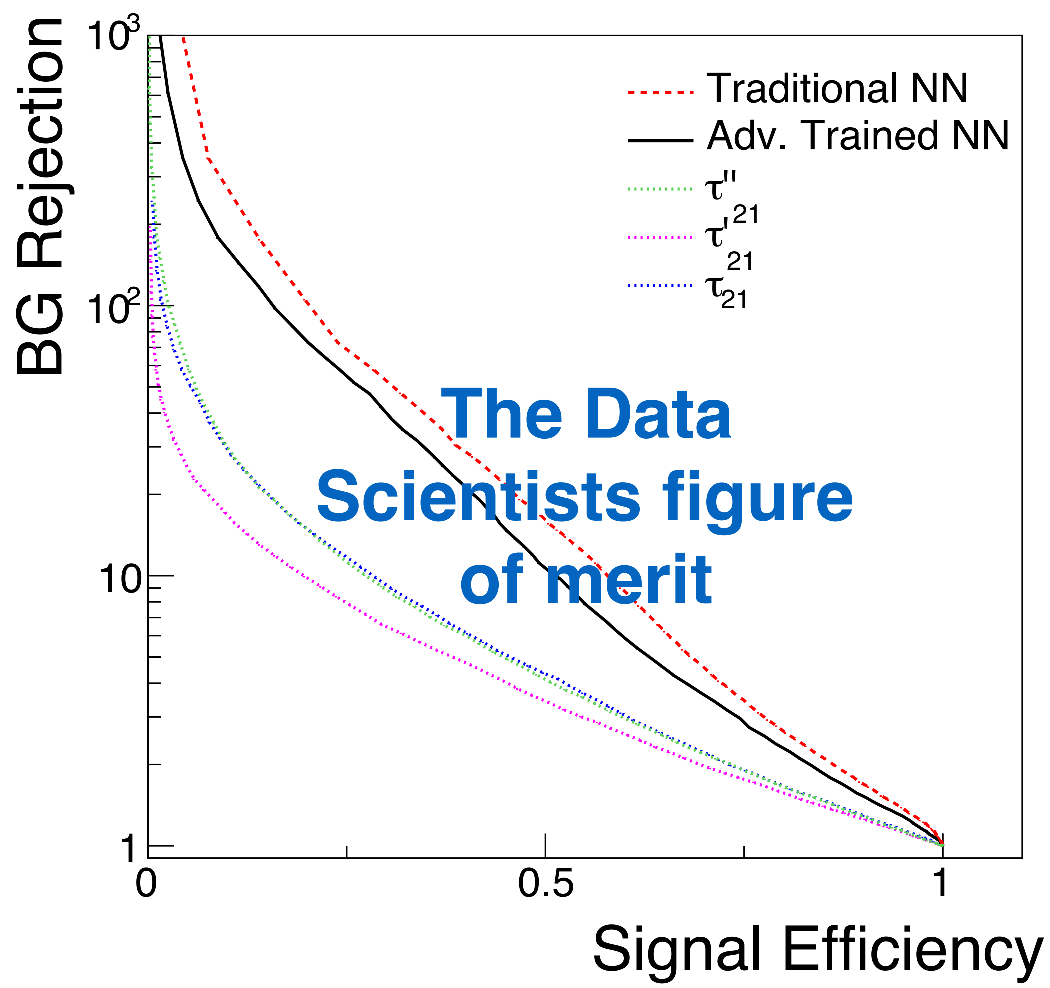
arXiv:1701.05927

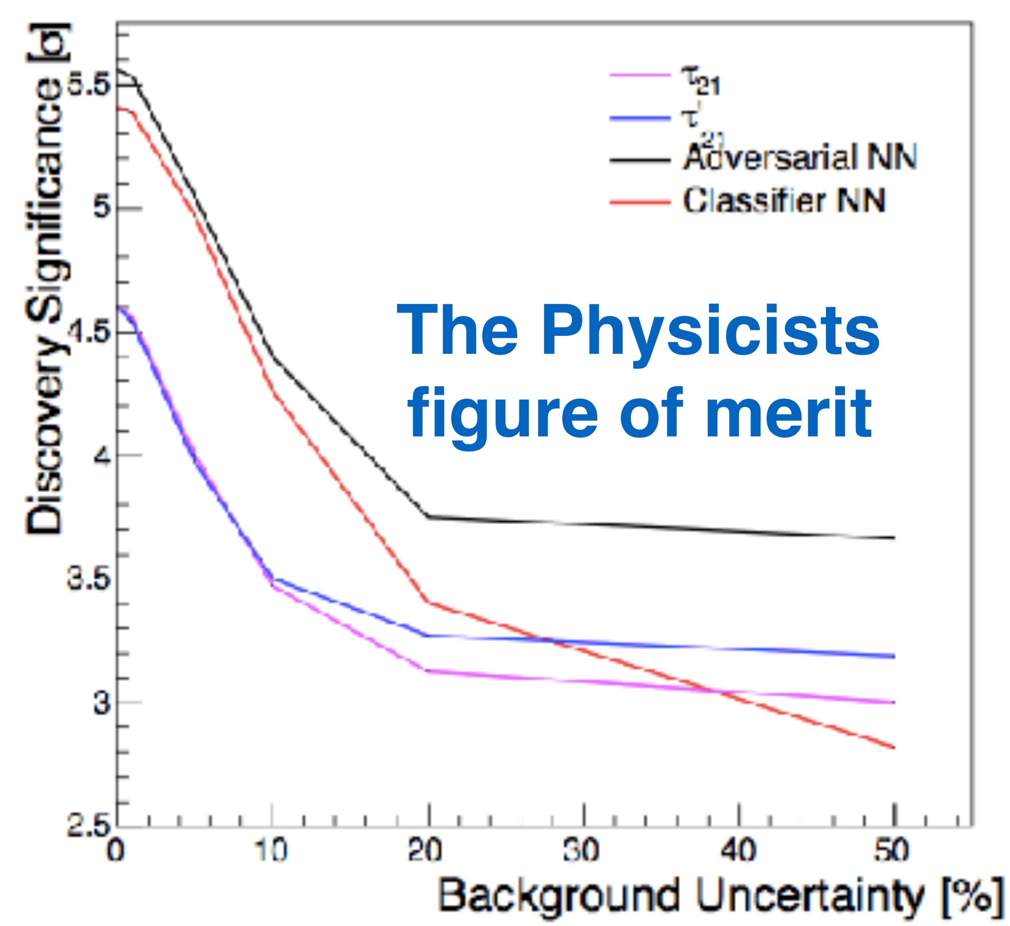
Adversarial Training

- Adversarial training allows us to impose physics-driven conditions to our training
  - Decorrelate DNN score from given physics quantities (as done with BDTs in the past)
  - Look for a (local?) minimum which minimises systematic effects (e.g., data/MC agreement)



arXiv:1703.03507

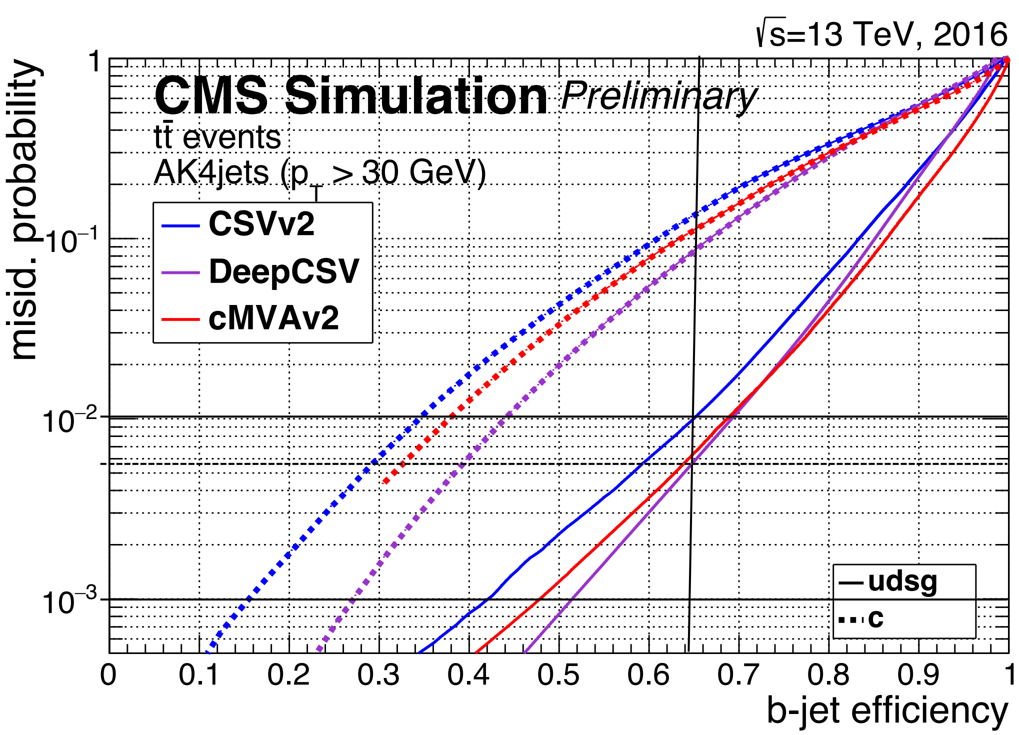


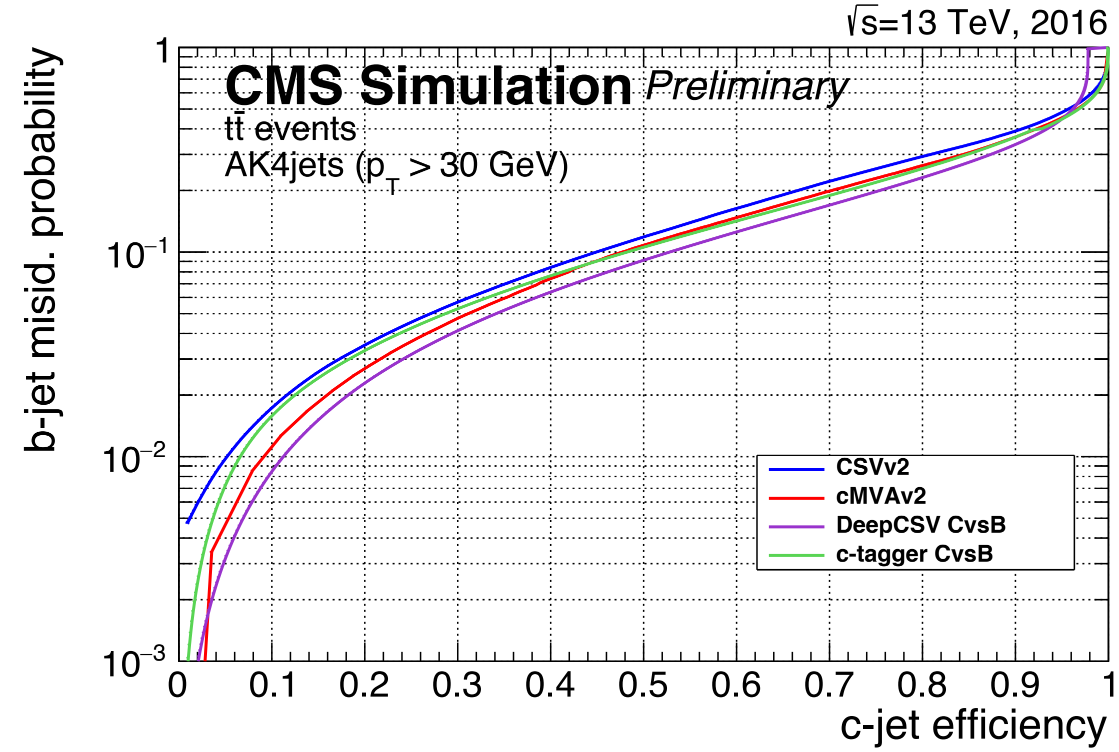


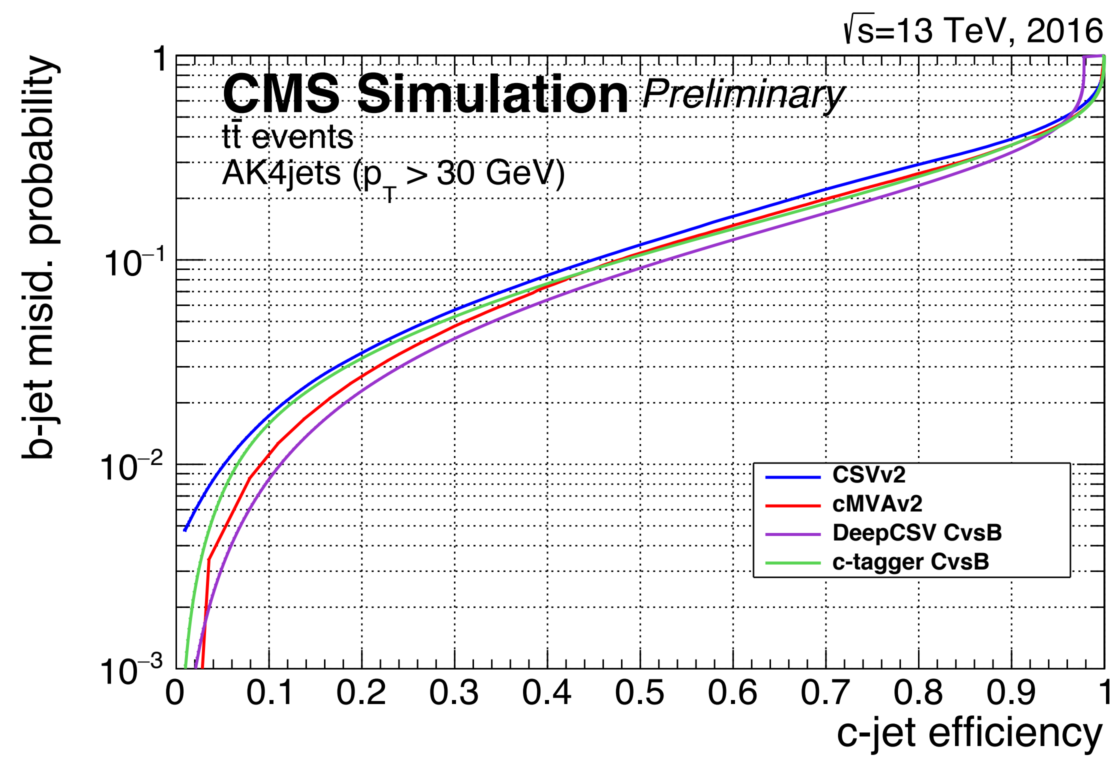
DL in real life: b-tagging

- Repeating history: b-tagging is the first "production-ready" algorithm that made it to mainstream data analysis usage

- Example from CMS: tagging based on high-level features (previously used in BDTs)



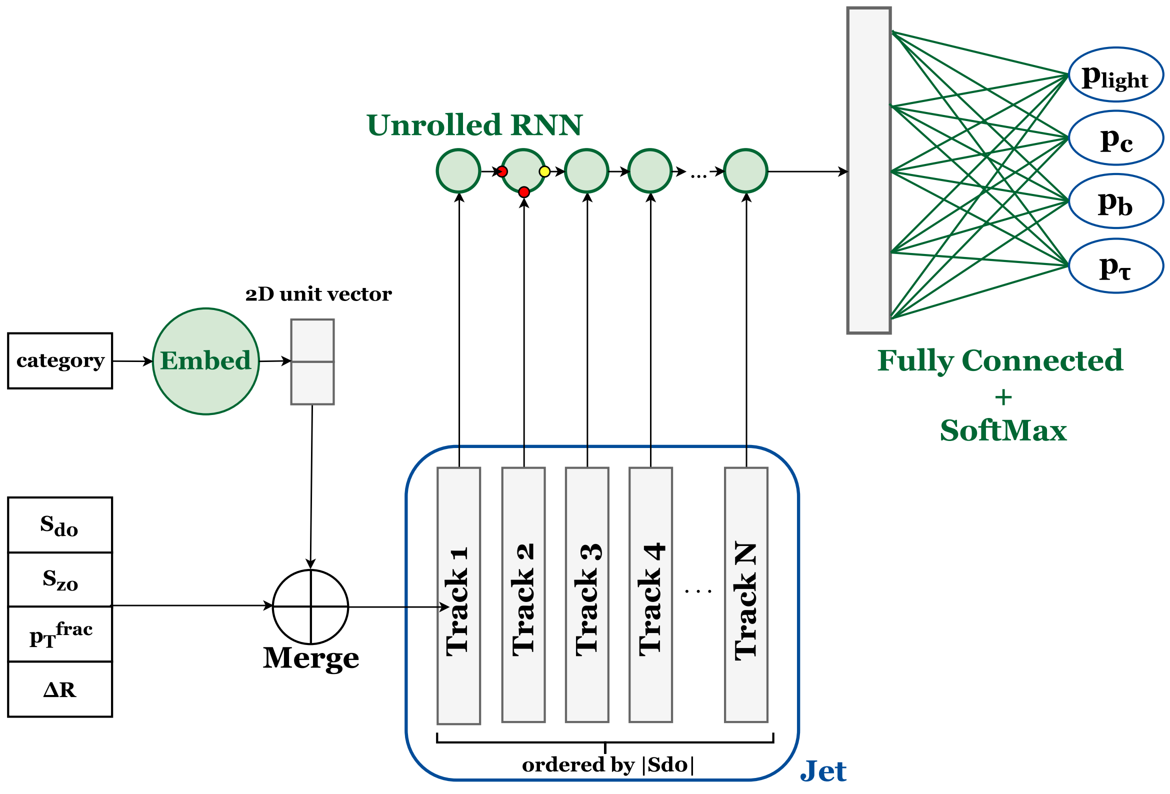


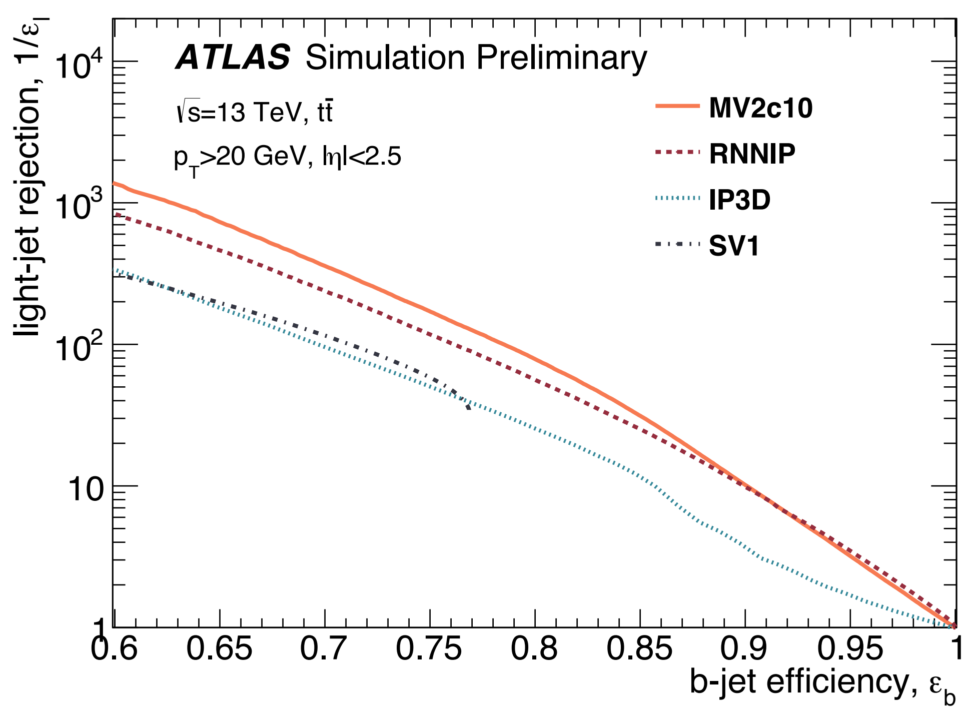


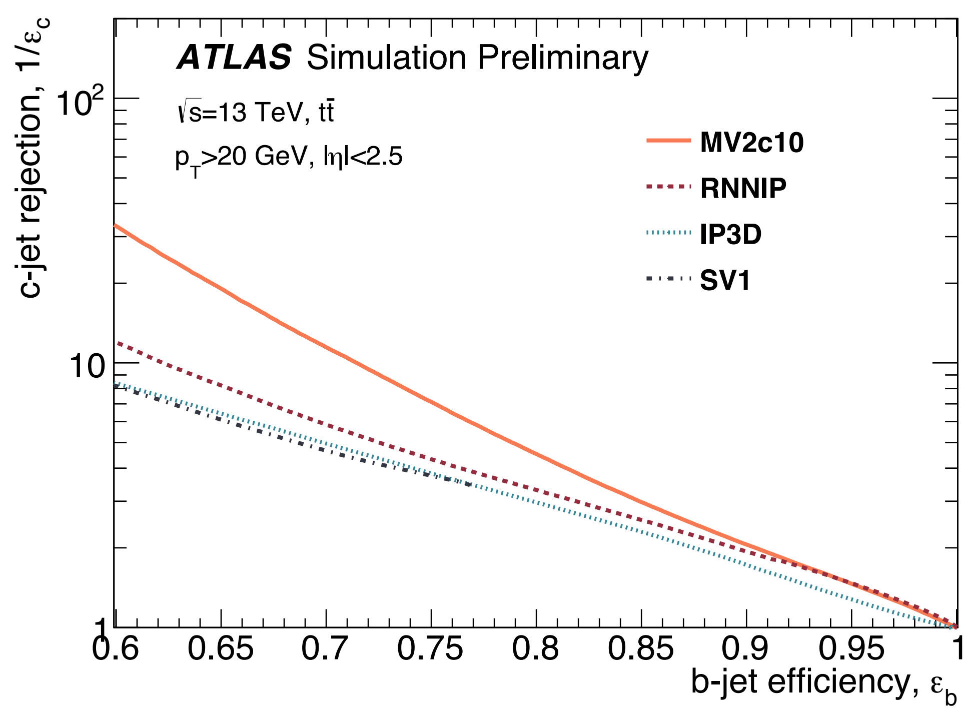
CMS-DP-2017-005

DL in real life: b-tagging

- MV2 using IP3D still rejects more background for εb < 0.9
- But this uses JetFitter and SV → much more information RNN as input for MV2 is outside the scope of this talk
- But we can imagine replacing IP3D with the RNN





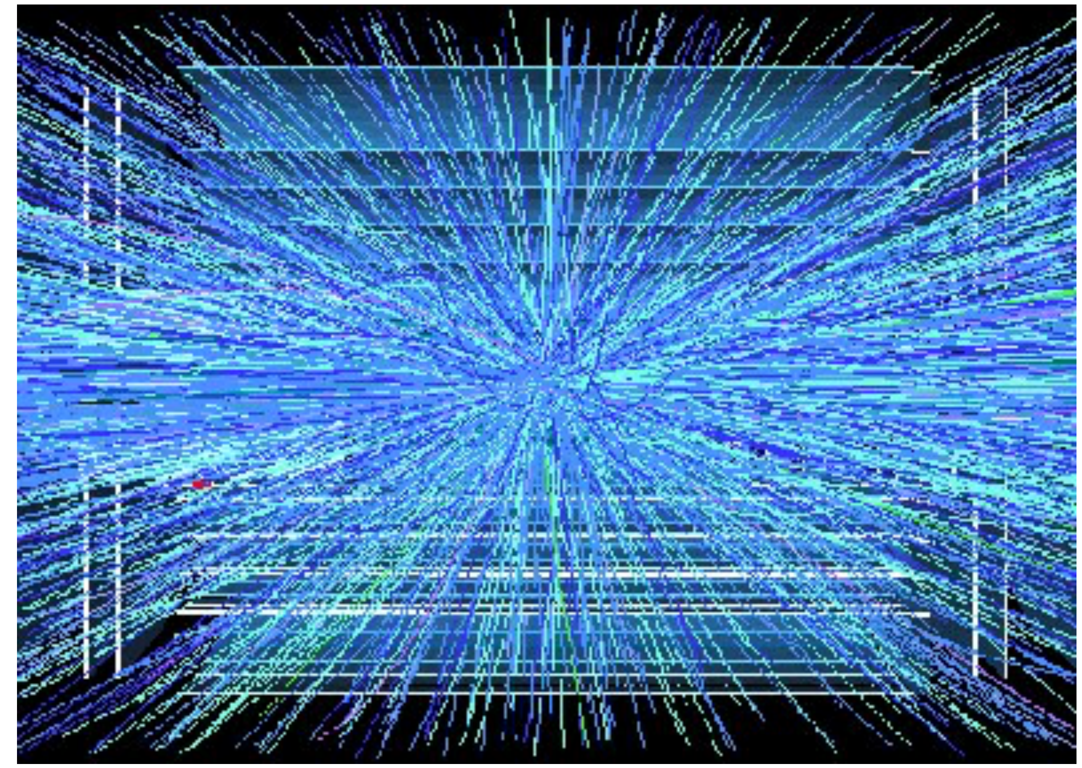


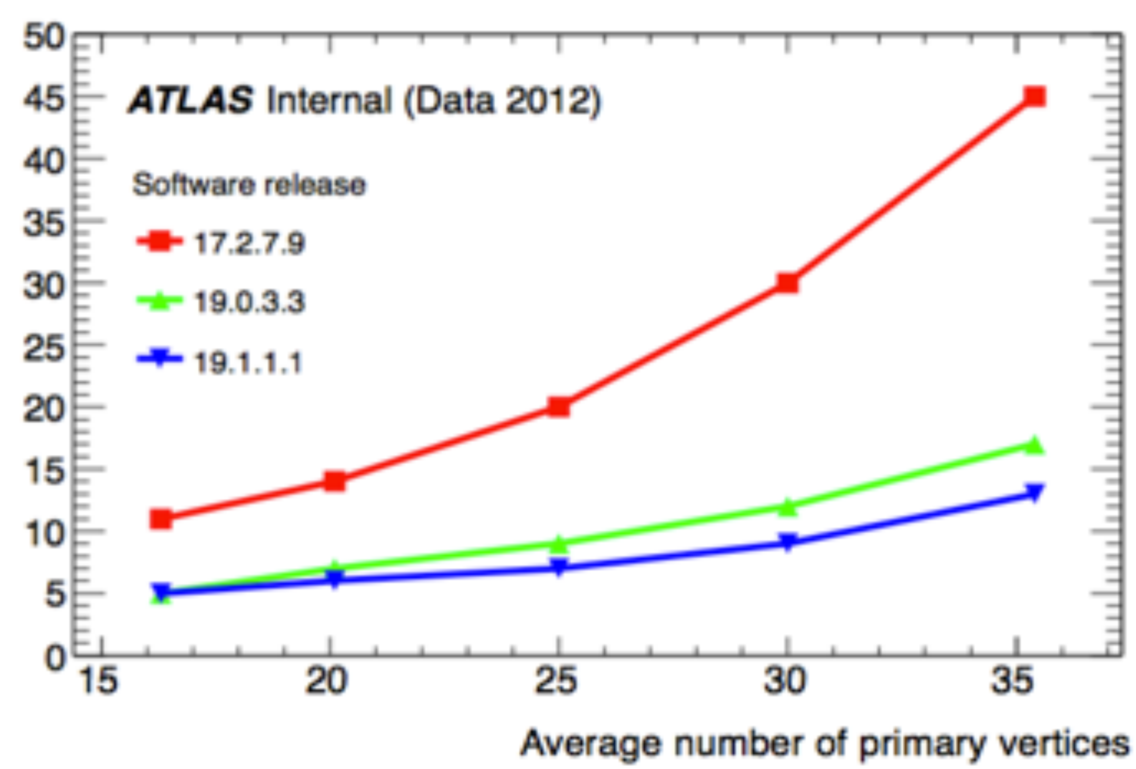
ATL-PHYS-PUB-2017-003

Deep Learning for @HEP future

The challenge ahead

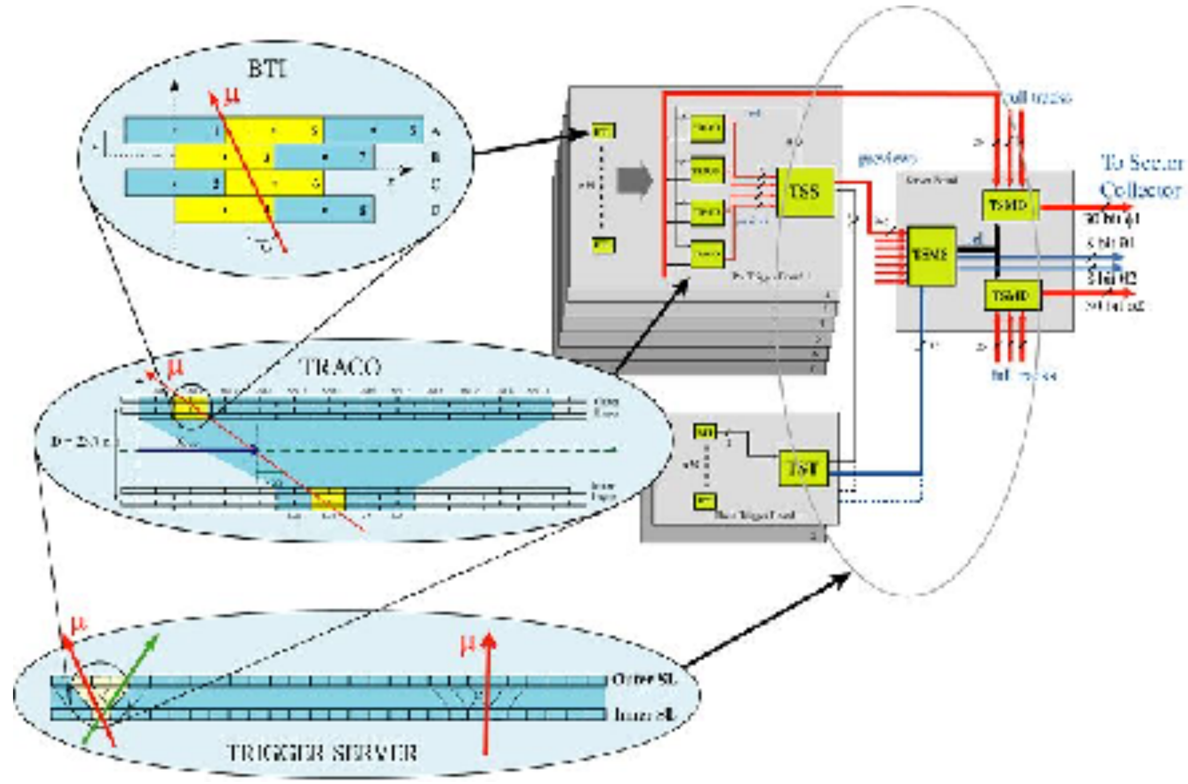
- The evolving conditions of the machine are drifting the experiments to more prohibitive environments (luminosity comes with a cost)
- More (& bigger) events to handle
- More noise from pileup interactions
- Increase in resources will not scale with needs
- Flat (or decreasing?) budget
- (Non linearly) increasing demand
- Need to find better ways to do things
- Problems can be formulated as image detection, where big progresses are happening (see ConvNNs)



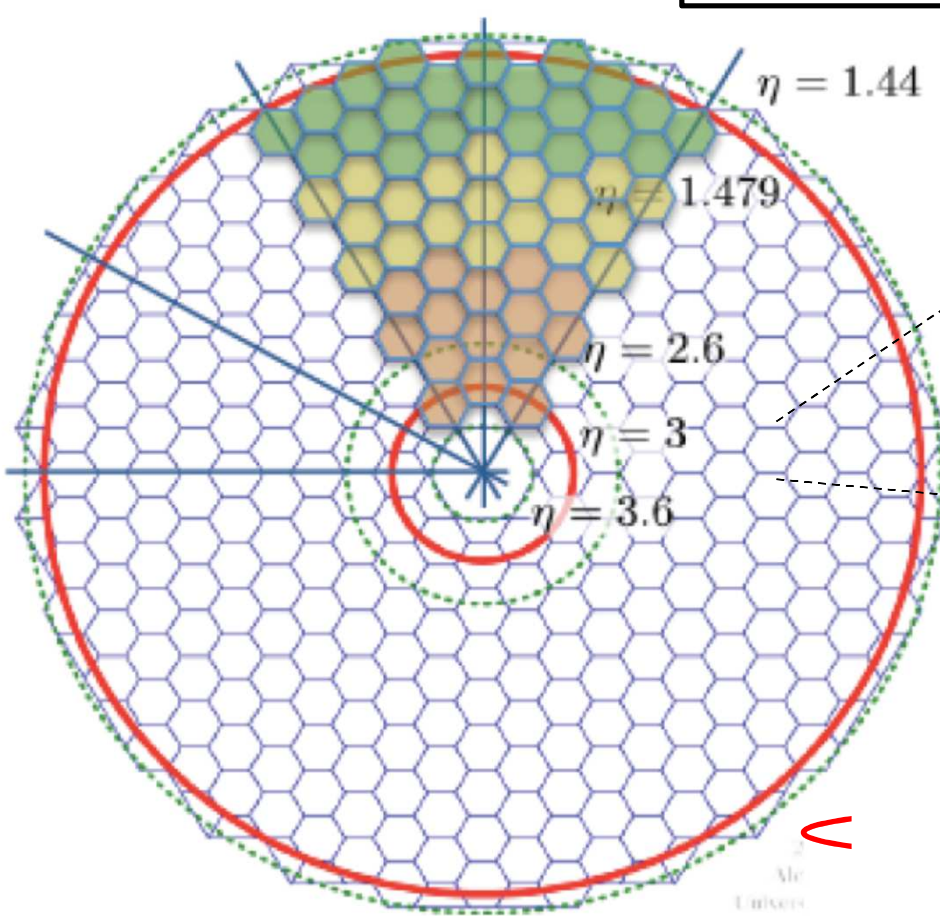


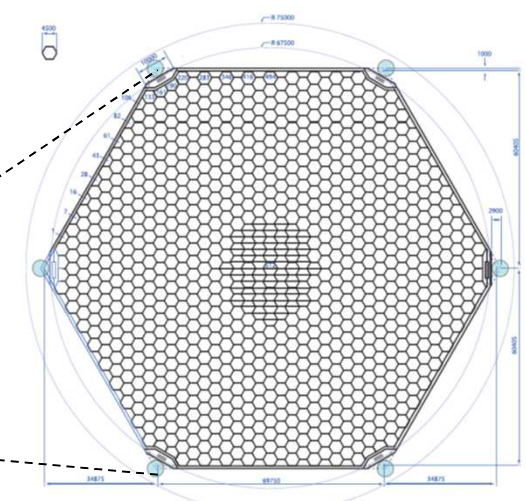
New instruments

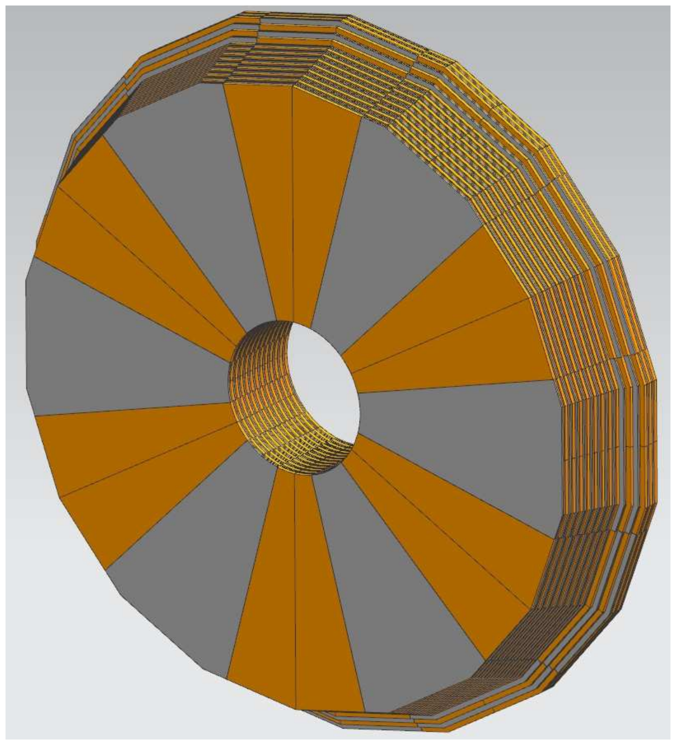
- The High-Luminosity challenges will be faced improving the detector
  - add tracking capability earlier in the game (@L1 trigger)
  - improve detector coverage
  - improve detector granularity



Hexagonal 6" Si wafer (256 or 512 channels



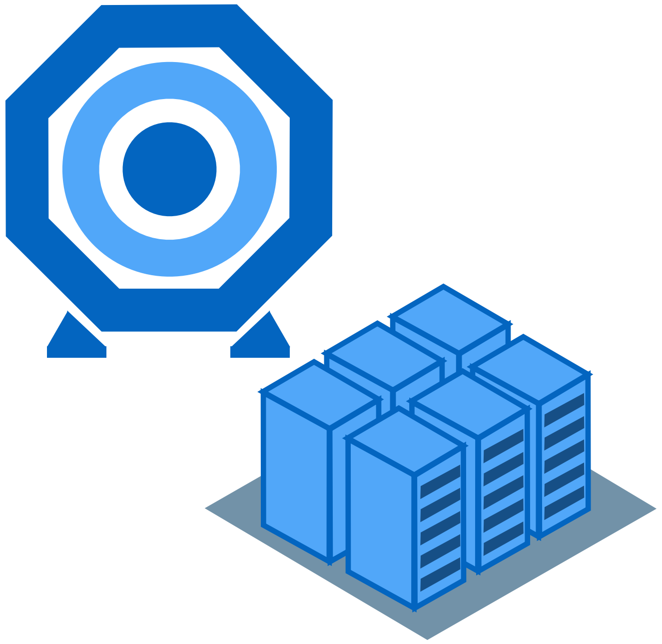




New instruments

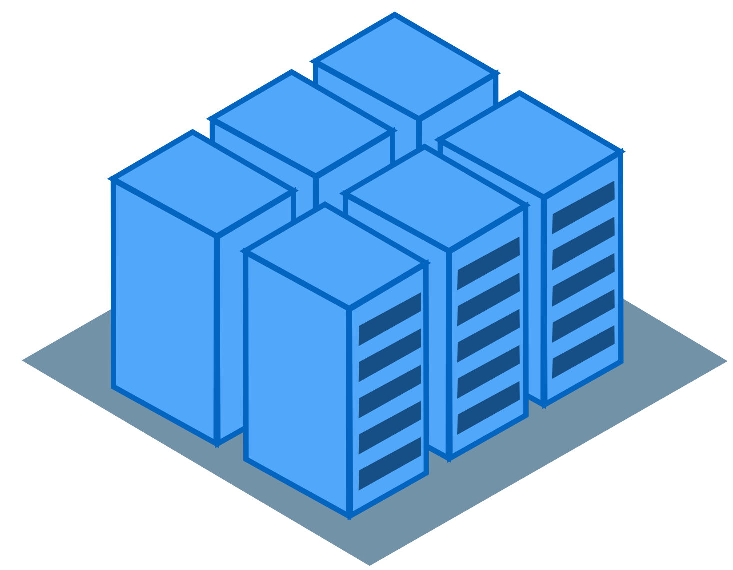
- More resources needed to exploit detector improvements
  - heavier reconstruction
  - more reconstruction steps
- But budget is flat, so we need a paradigm shift

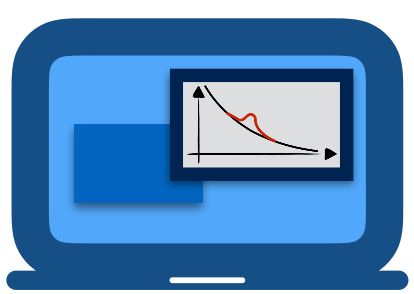
What DL can do for us



Online Data Taking (real time)
Fast trigger algorithms for topology classification
Fast reconstruction algorithms (clustering, tracking, classification)
Monitor detector operation conditions & data quality

Offline event processing (centralised)
Event indexing based on topology classification
Fast collision simulation based on generative models
Fast reconstruction algorithms (clustering, tracking, classification)

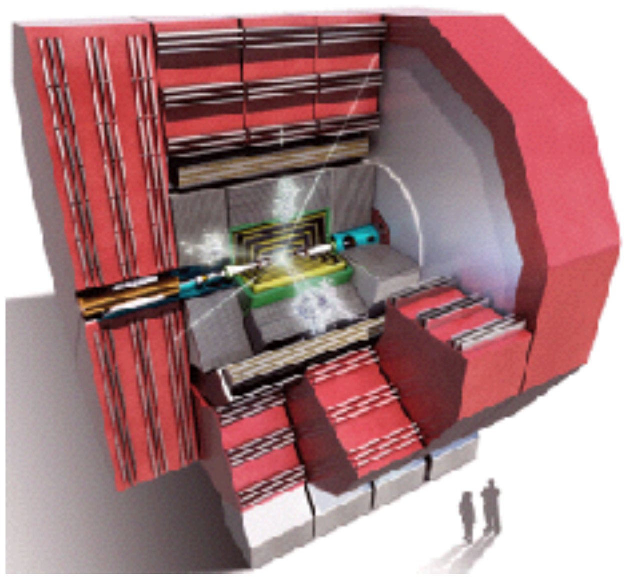


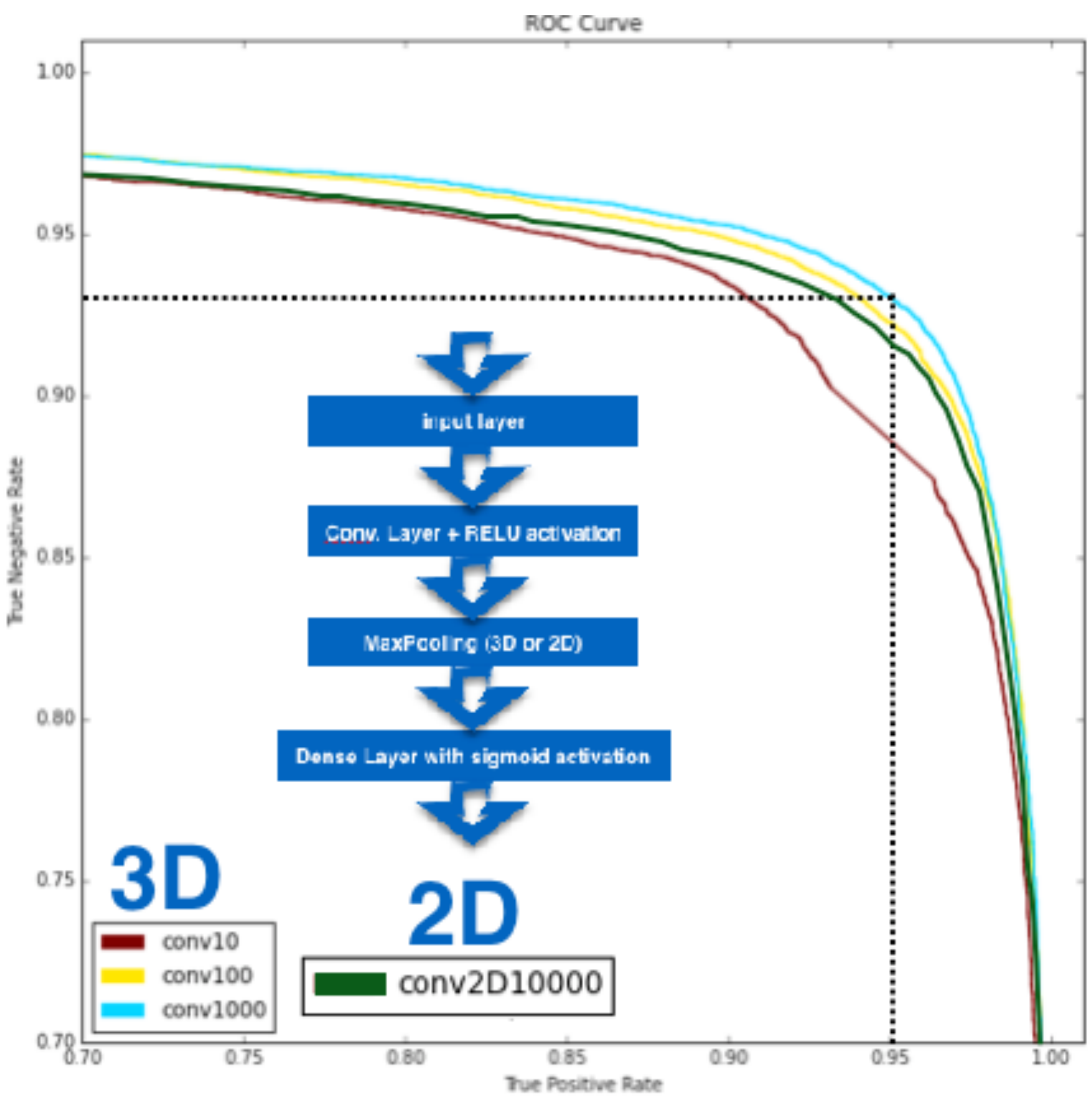


Data analysis (by users)
Particle identification
Jet tagging (g vs q vs b vs W/Z vs H vs t)
Unsupervised search for new physics as outlier detection

Examples: HG calorimetry

- Using as a benchmark the LCD detector design
- Accessible beyond the boundaries of experimental collaborations (eg, ATLAS+CMS)
- Example of next-generation highly-granular detector
- FullSIM available out of the box
- Defined single-particle benchmark datasets
- pions, electrons, and photons
- Used to train 2D and 3D reconstruction and identification based on ConvNN





Examples: Tracking

Several Times

Hits Preparation

Seeding

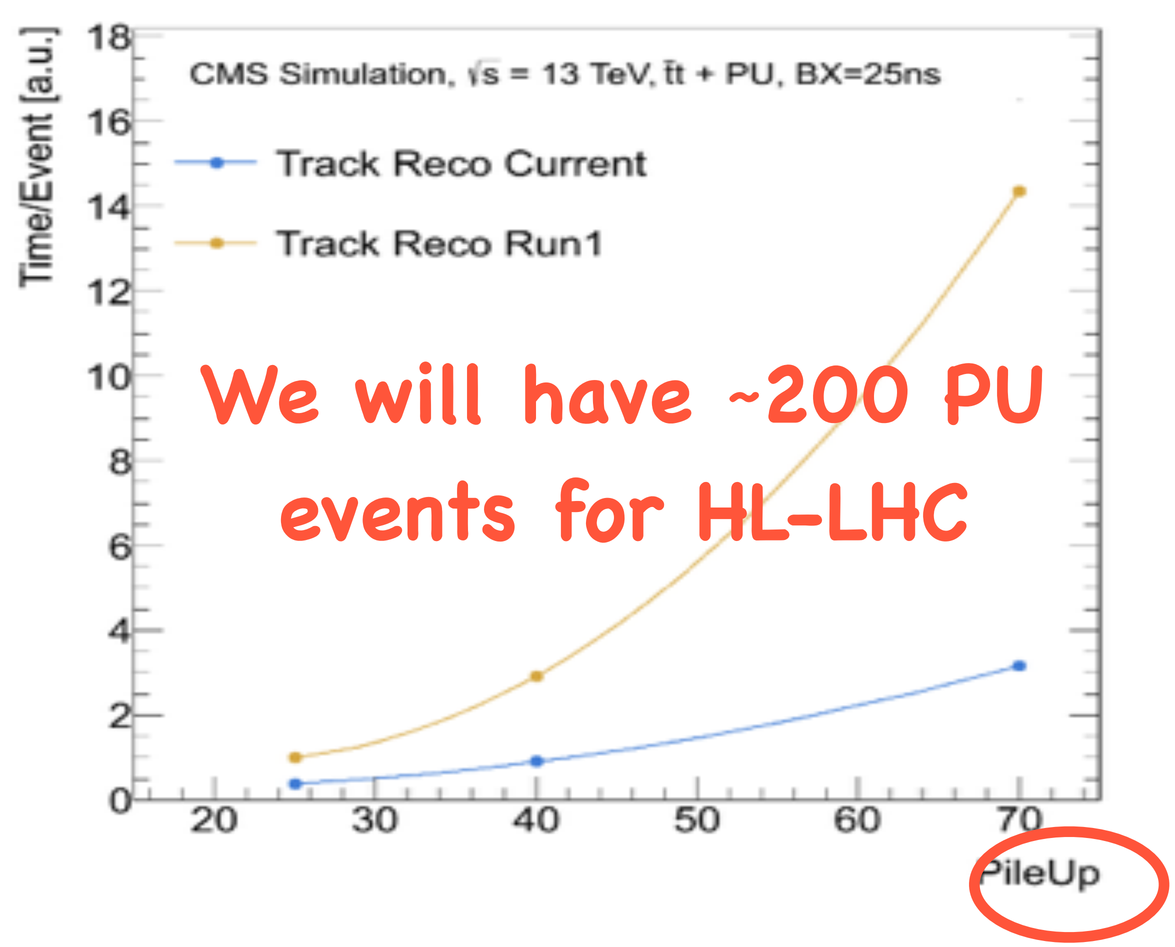
Pattern Recognition

Track Fitting

Track cleaning

Find particle trajectories using energy depositions on several sequential layers of pixel detectors
Connecting the dots, with…

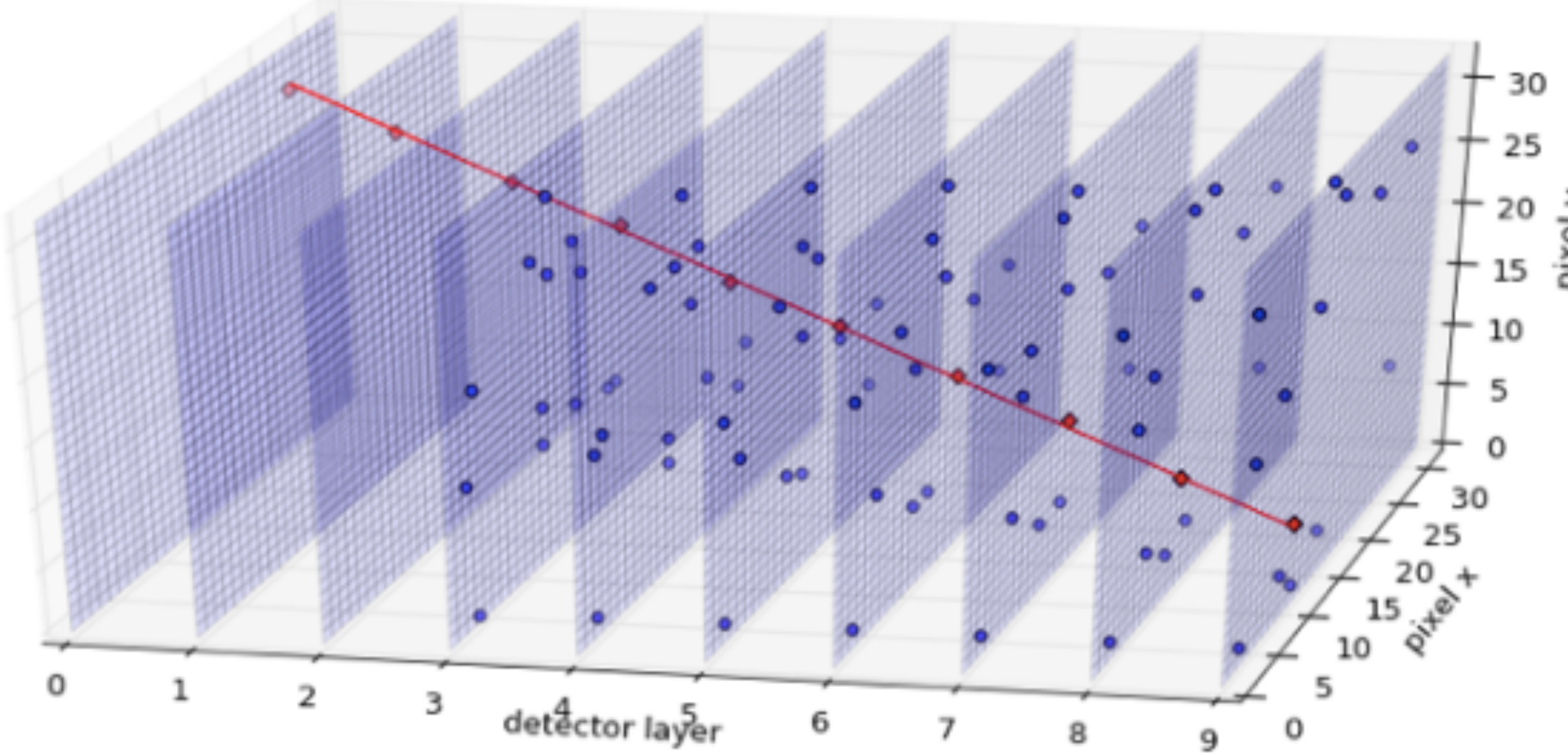
- Thousands of particles, each leaving energy depositions on O(10) layers of detectors
- High particle-density regions w/ multiple particles passing through same cluster of energy deposition
- Different kind of "hits" (plain, split, merged, …)



What you want to do

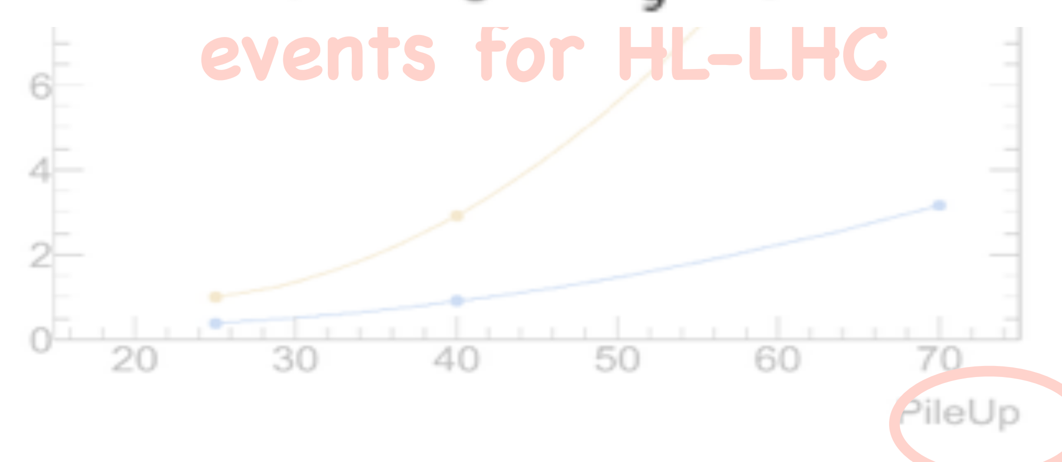
Find particle trajectories using energy depositions
on several sequential layers of pixel detectors
Connecting the dots, with…

Hits Preparation



…)

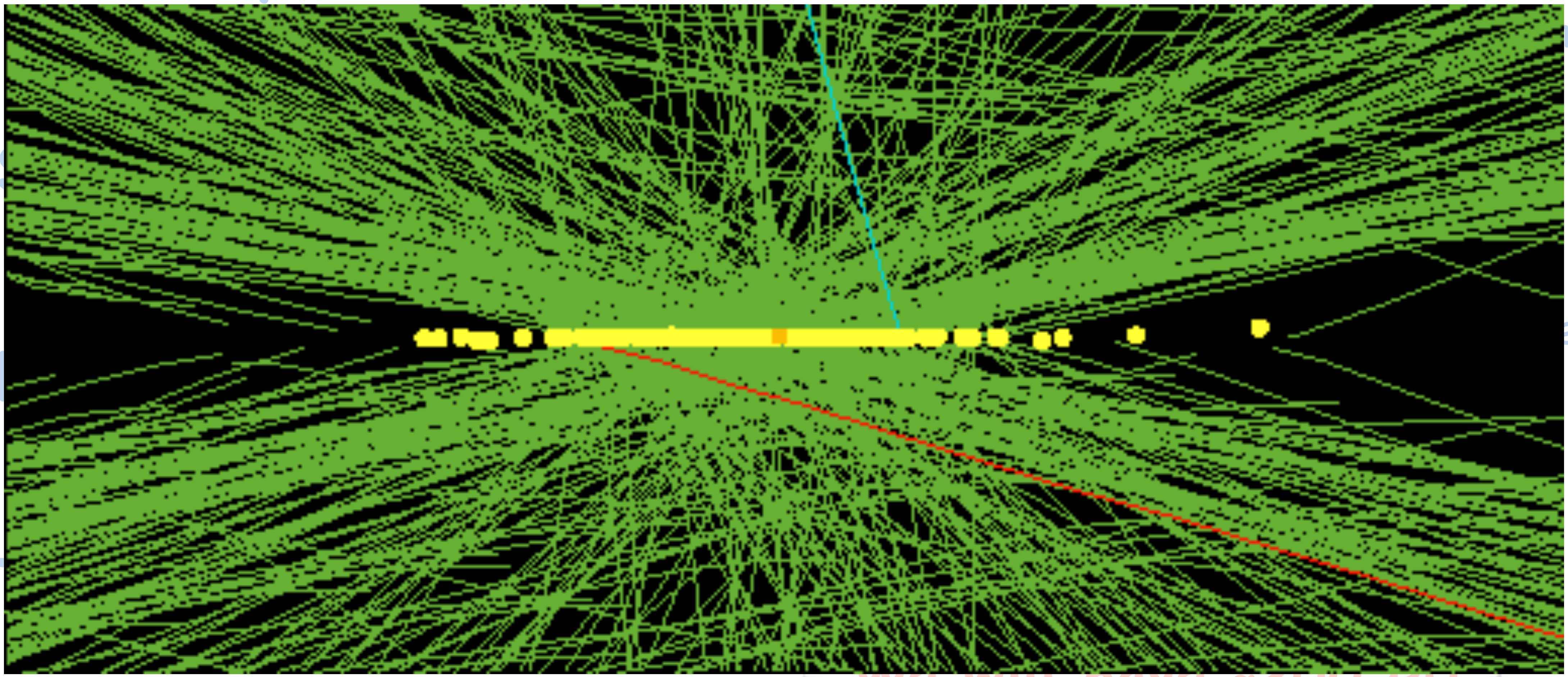
Track cleaning



What real life looks like

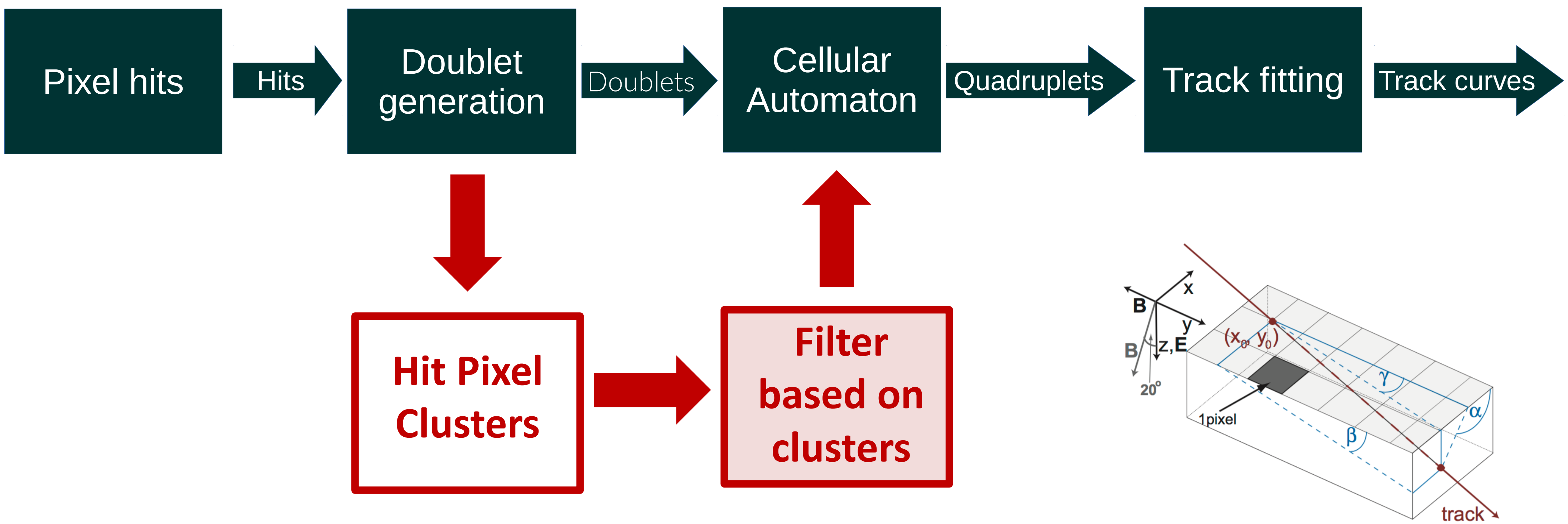
Find particle trajectories using energy depositions on several sequential layers of pixel detectors
Connecting the dots, with...

Hits Preparation



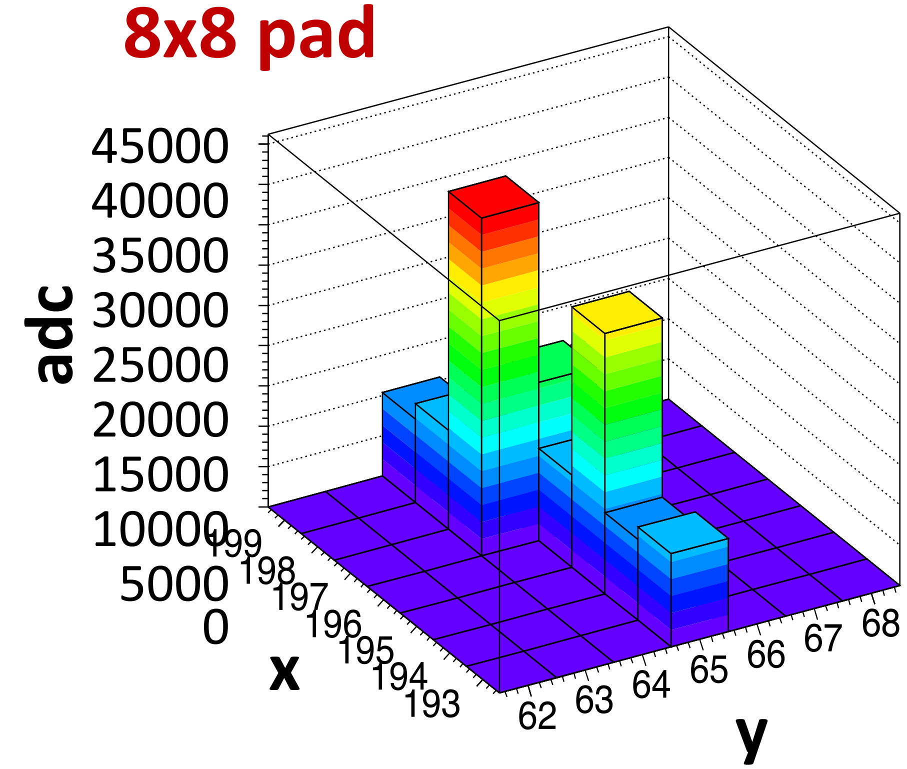
Track cleaning

Examples: Tracking



- Represent hits as 8x8 images
  - use the deposited energy (ADC counts) as temperature
- Use DNN to decide if a given pair of hits is a good match or a fake

8x8 pad



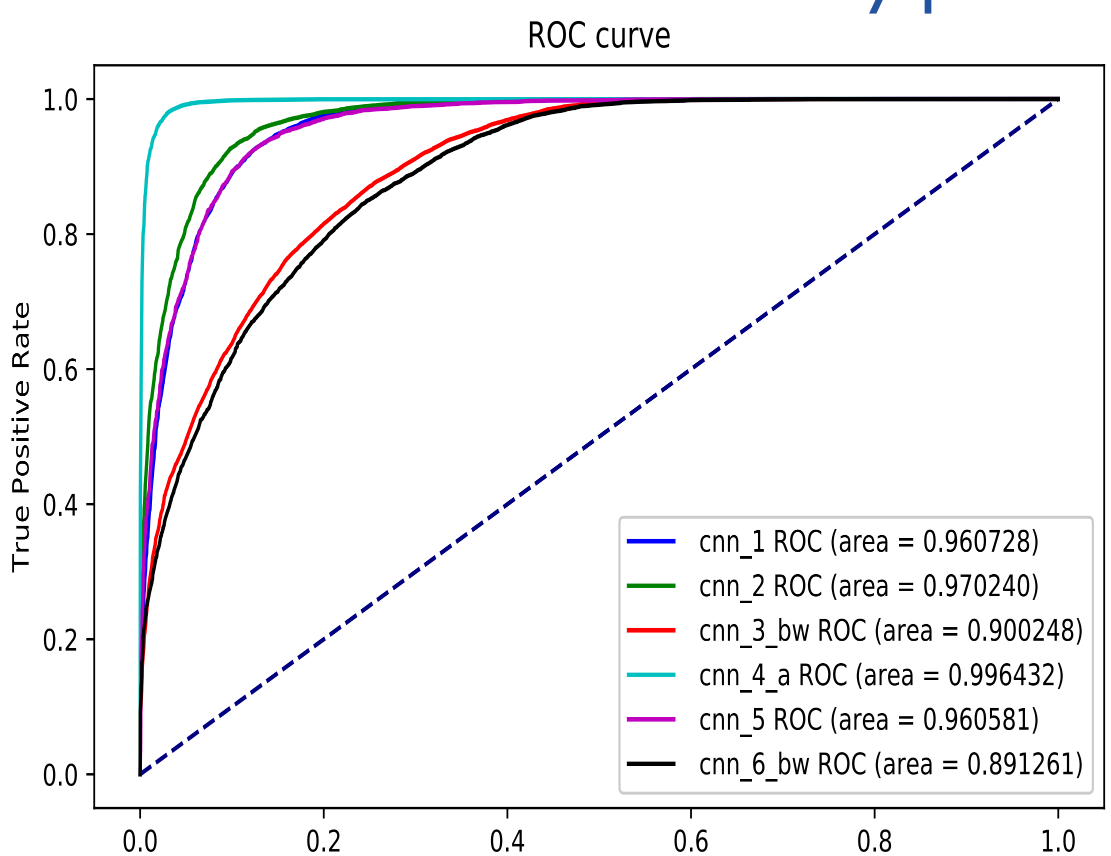
Examples: Tracking

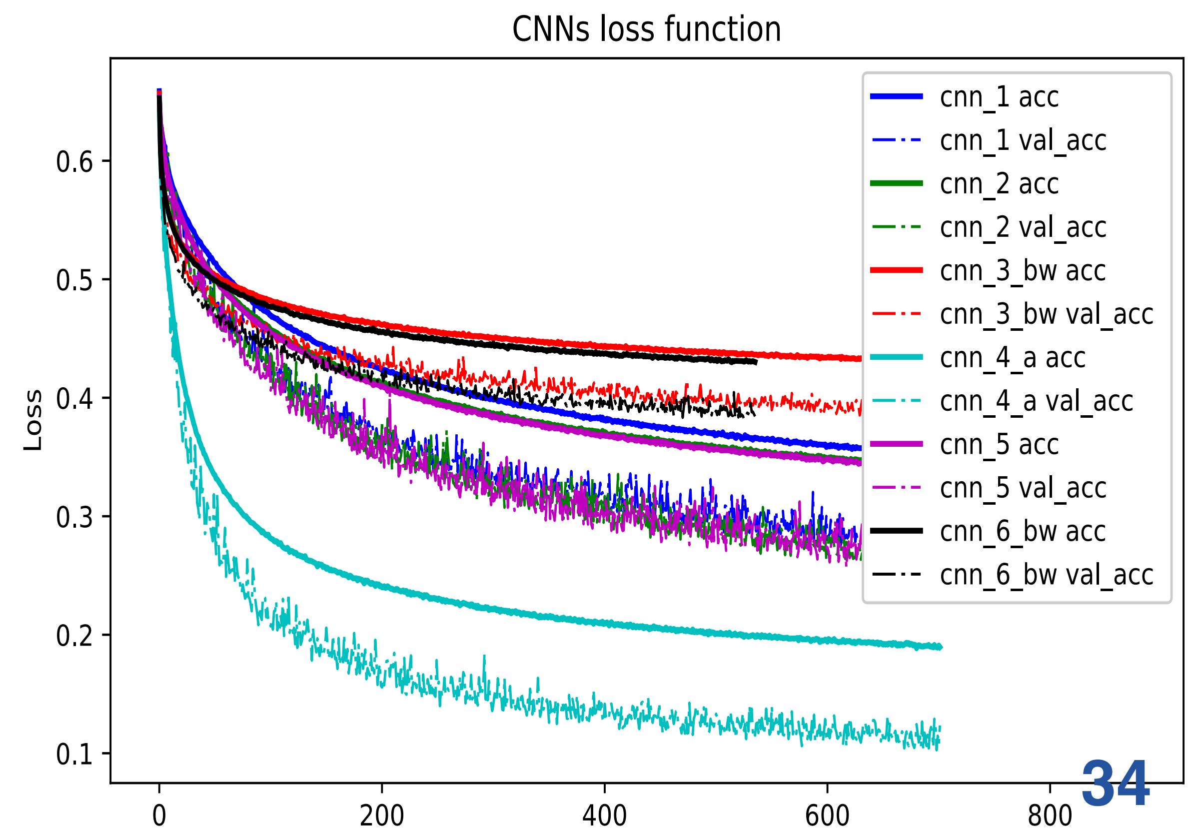
- Ongoing work with CMS simulations
- Currently exploring possibilities
  - data representation
  - network architecture
  - …
- First results are extremely promising

1-false positive %

CNN eff. @ rej	0.5	0.75	0.9	0.99
cnn_1	0.99	0.98	0.89	0.38
cnn_2	0.99	0.98	0.92	0.53
cnn_3_bw	0.99	0.87	0.63	0.28
cnn_4_a	**1.0**	**0.99**	**0.99**	**0.91**
cnn_5	0.99	0.98	0.89	0.42
cnn_6_bw	0.99	0.85	0.61	0.26

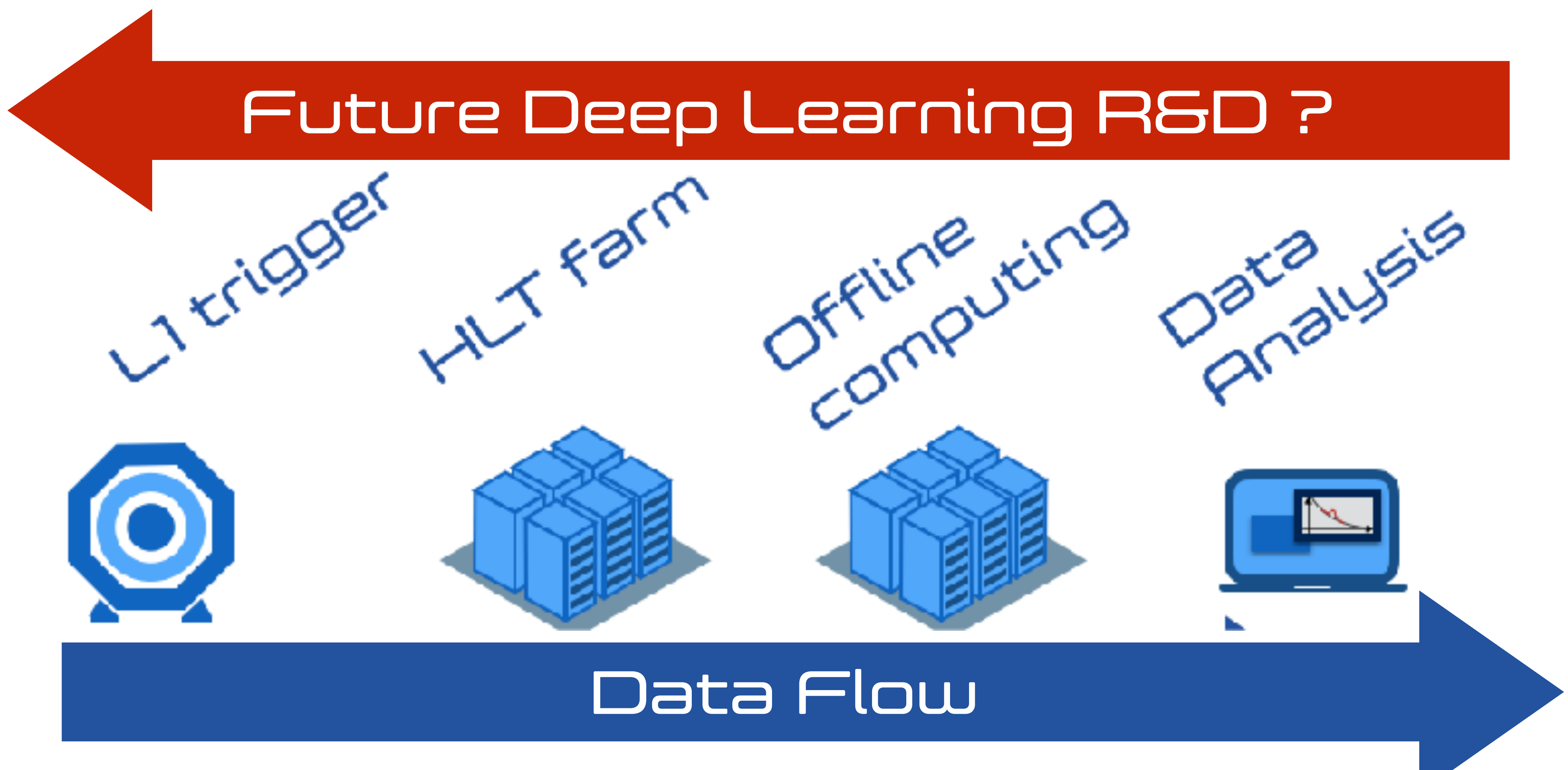
true positive %





Better? Faster? Both?

- We would be extremely happy to have fast DLL algorithms that behave as well as our current offline reconstruction
  - we could move these algorithms earlier in our data flow
  - Benefits downstream of a better triggers
    - more "good events" to write for given resources
    - less events to write for a given number of good events
- Of course, if we could achieve better performances as well



Training: BDTs vs DLs

- People are used to train their own ML algorithm on their laptop
  - BDTs allow to do that
- This promoted the use of BDT to > 50% our data analysis, 100% of the event reconstruction, etc
- To transition to DL, training as to be made user friendly
  - Software is there (Keras, TensorFlow, etc in LHC-physicists-friendly python echo system)
  - We need the hardware (& the competence) to run small (and large) training
- If this becomes part of the central processing workflows, we will certainly need adequate central resources (e.g., a GPU cluster @CERN?)

Inference: BDTs vs DLs

- Offline inference is not an issue (see)
- Online inference comes with constraints
- We are currently running ML@Trigger
  - BDTs at HLT (b-tagging, photon ID) to "clean up" the reconstructed objects (regressions & classifications)
  - BDT at L1 (as look-up tables) to improve energy measurements (low-dimension regressions)
- We want to use DL at trigger for more
  - go back to RAW data
  - use DL to "predict" the reconstruction outcome
    - save time in the trigger
    - improve decision? (i.e., save bandwidth, disk space, etc)
- We need R&D to see how much of this is realistic