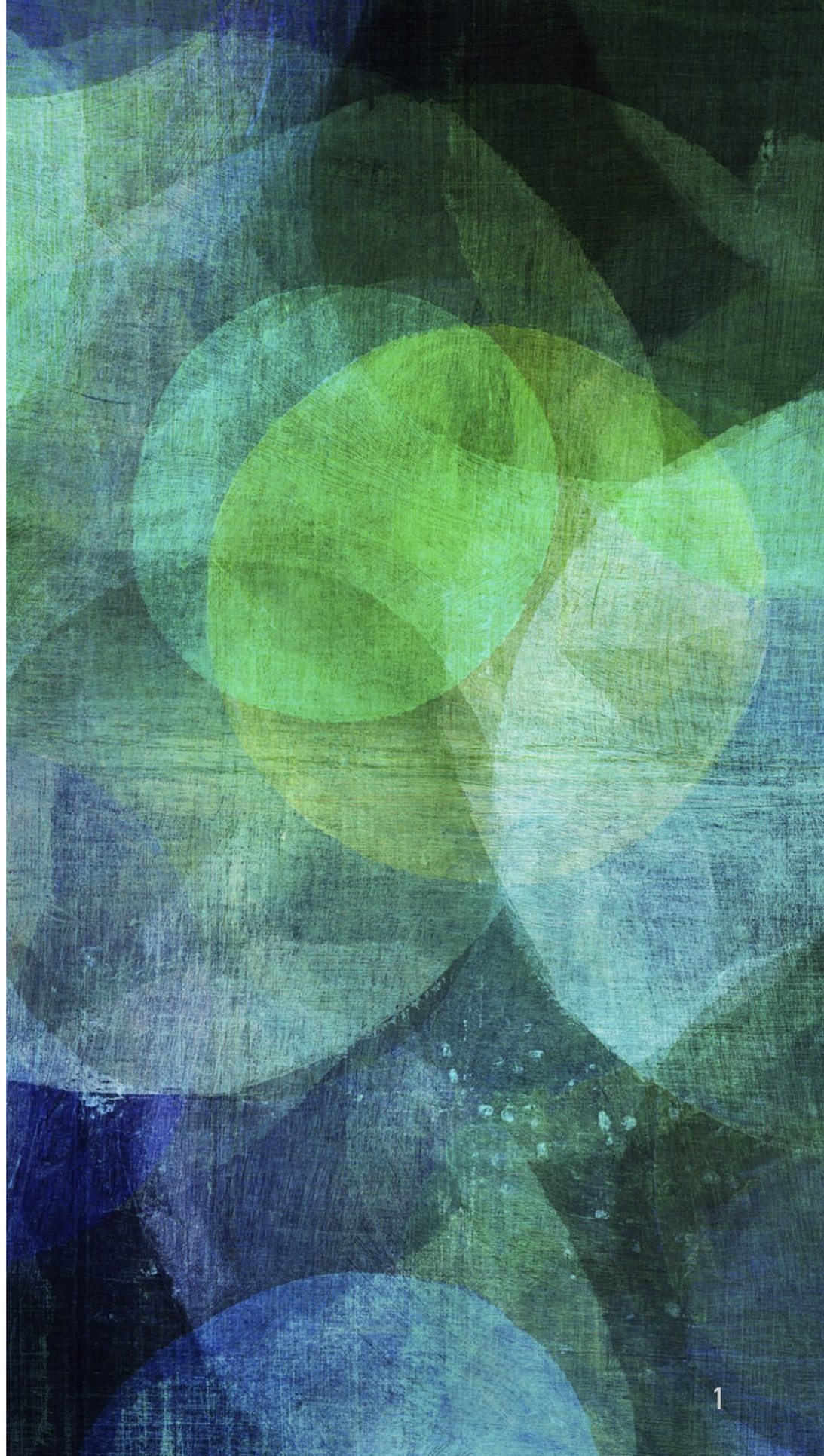
EXPLOITING PARALLELIZABLE ARCHITECTURES AND ARTIFICIAL INTELLIGENCE AT THE LHC EXPERIMENTS

Cesare Calabria

"INFN and The Future of Scientific Computing
Episode I: The HPC Opportunity"
2018/05/04



PAST ACTIVITIES AND RESPONSIBILITIES

- **Research activity within the CMS experiment**
  - **Analysis of the date collected with the CMS experiment at LHC**
    - Electroweak physics: measurement of the Z boson production cross section through its decay to a tau lepton pair
    - Search for a Standard Model Higgs boson produced in association with a W vector boson
  - **Study of the physics objects in CMS**
    - Study of the muon reconstruction and identification performance in CMS
    - Study of the hadronic tau reconstruction and identification performance in CMS
  - **Activities concerning the detectors**
    - Study of the RPC performance in CMS
    - Upgrade of the CMS Muon System forward region with new detectors based on GEM technology
    - Phase-2 upgrade of the Muon System

- **Responsibilities within the CMS experiment**
  - 2010 - 2012: Responsible for the study and monitoring of the RPC efficiency (L3)
  - **2014 - 2015: GEM Reconstruction and Validation Coordinator (L3)**
  - **2014 - 2017: GEM Software and Online Contact for Upgrade (L3)**
  - 2015 - 2017: GEM DPG Coordinator (L2)
  - **2016 - 2017: Muon Phase-II Simulation Coordinator (L3)**
  - **2016 - 2017: Contact person between Upgrade Studies Group and CMS Offline & Computing group (L3)**
  - 2016 - 2017: Link person CMS - Bari Tier2

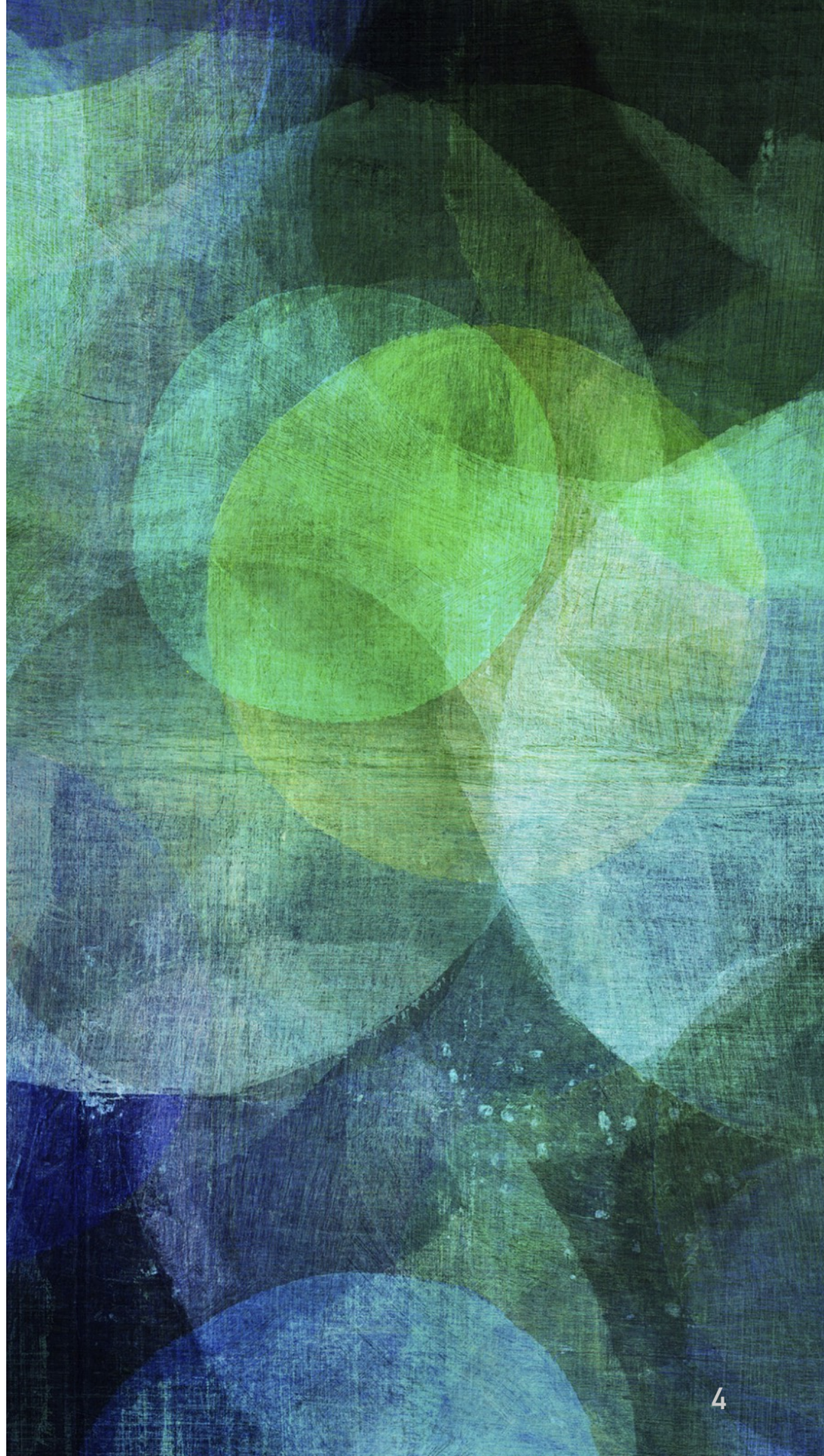
- **Development and integration of the software needed for the Phase2 Muon Upgrade studies (muon reconstruction and identification, validation tools, study of the neutron background and muon perfomance…)**

CURRENT ACTIVITIES

In the context of my INFN "fellowship" regarding the R&D on scientific computing for innovative solutions for the LHC experiments, my activity is twofold:

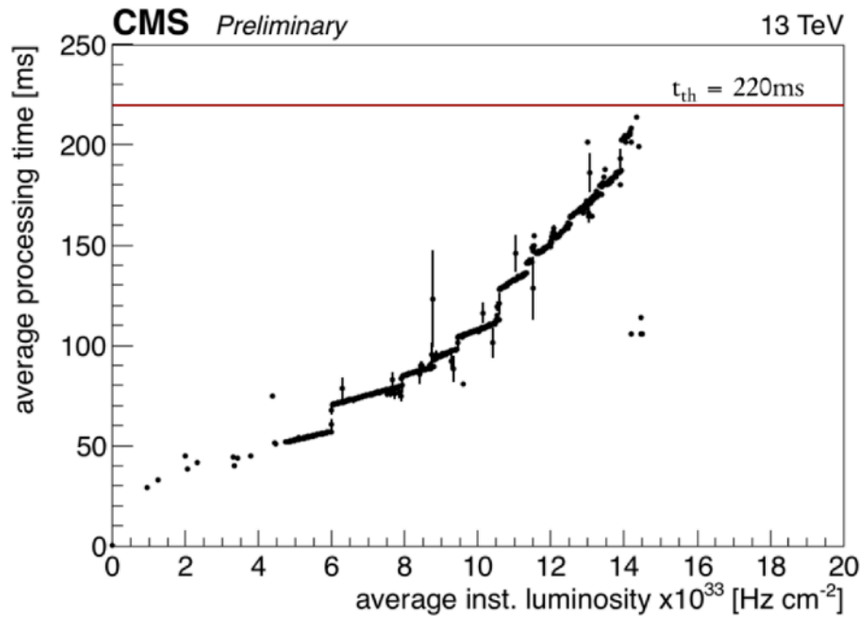
1. **Parallel programming on GPU: I am collaborating with the "Future tracking" group**
   - ➤ The group takes care in CMS of developing a demonstrator for the pixel tracking on GPU and all the other infrastructures needed to exploit at best the available hardware resources (heterogeneous computing)
   - ➤ **So far I contributed to the implementation on GPU and to the optimization of the first step of the chain: the unpacking of raw data (Raw2Digi) for the pixel detector (details in the next slides)**
2. **Machine learning application: I am collaborating with the "ML Muon" group**
   - ➤ The groups take care inside the CMS Muon community (DT, RPC, CSC, GEM) of developing innovative tools for monitoring the performance of the CMS Muon System and the detection of its anomalies
   - ➤ **Currently I am working on the development of a monitoring tool based on ML techniques for the DT Trigger System**
   - ➤ On long term this R&D work is meant to lead to a tool that can be run at different stages of the L1 Muon Trigger (details in the next slides)

CMS PIXEL TRACKING ON GPU



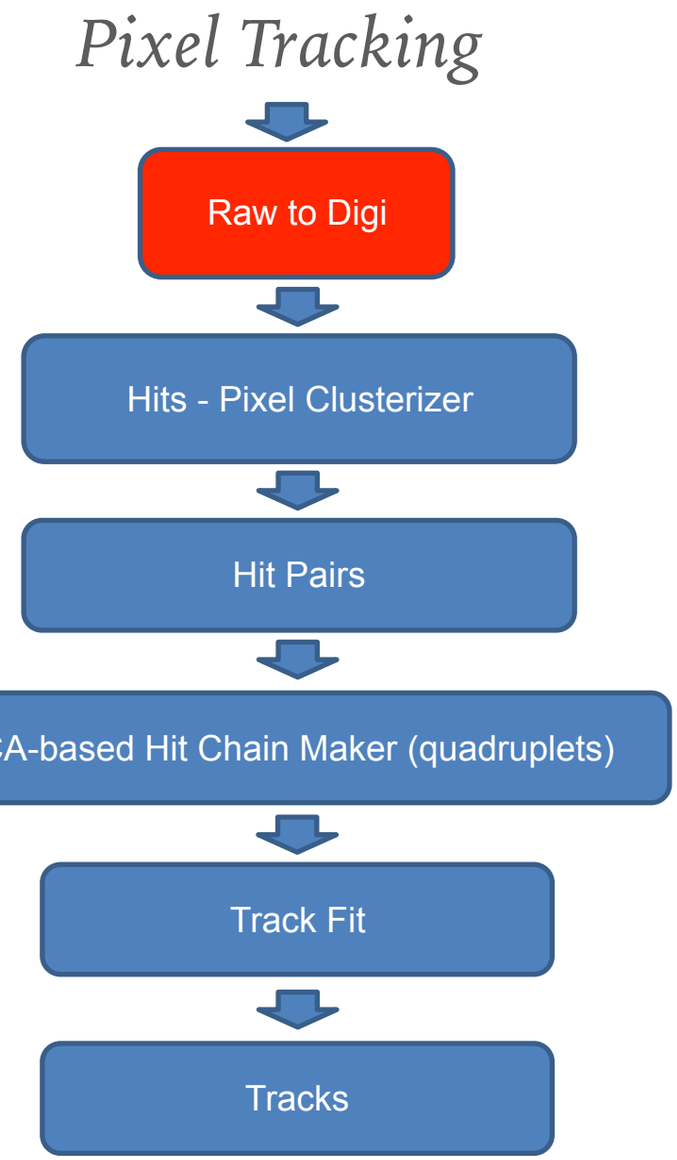
THE CMS TRIGGER SYSTEM

- **CMS Trigger System**
  - Reduce input rate (40 MHz) to a data rate (~1 kHz) that can be stored, reconstructed and analyzed Offline maximizing the physics reach of the experiment
  - **Level 1 Trigger**
    - Coarse readout of the Calorimeters and Muon detectors
    - Implemented in custom electronics (ASICs and FPGAs)
    - Output rate limited to 100 kHz by the readout electronics
  - **High Level Trigger**
    - Readout of the whole detector with full granularity
    - Output rate limited to an average of ~1 kHz by the Offline resources
    - **Today the CMS HLT online farm consists of ~22k Intel Xeon cores**
      - The current approach: one event per logical core
      - **Pixel Tracks cannot be reconstructed for all the events at the HLT**
      - This will be even more difficult at higher pile-up
        - Combinatorial time in pixel seeding O(pileup!) in worst case



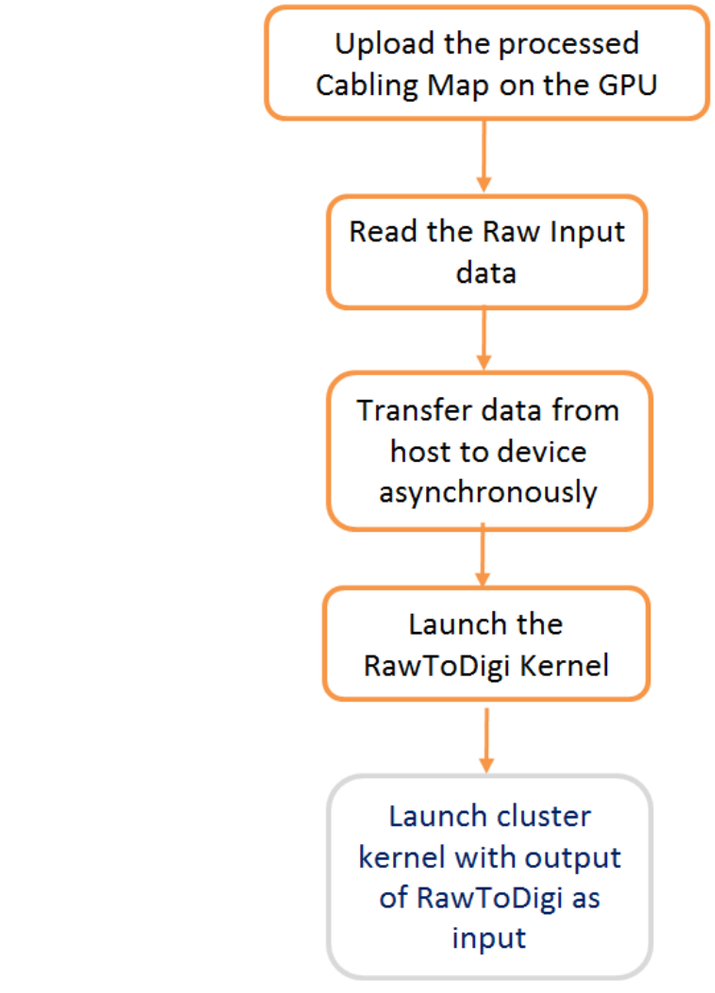
THE CMS PIXEL TRACKING AND THE PATATRACK PROJECT

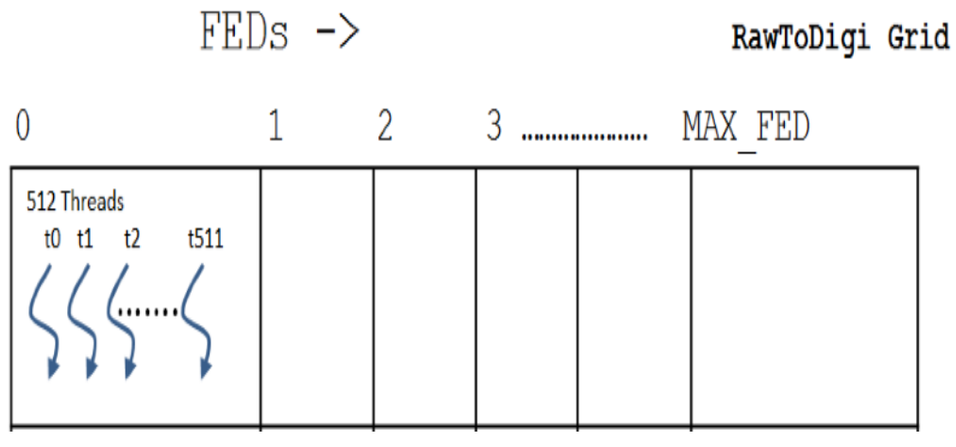
- **Solution (objective of the “Patatrack" project in CMS):**
  - **Develop a hybrid CPU-GPU application that takes RAW data coming from the pixel detector and gives Tracks as result**
  - Trigger average latency should stay within 220ms
  - GOAL: demonstrator ready by 09/2018 to run parasitically at the HLT farm (in order to be included for Run3 and then hopefully Run4)
- **Ingredients:**
  - Massive parallelism within the event
  - Avoid useless data transfers and transformations
  - Simple data formats optimized for parallel memory access
  - Renovation at algorithmic level
- **My contribution to the project is on the implementation on GPU of the first step of the pixel tracking chain: Raw2Digi step**



RAW2DIGI ON GPU: STATUS OF THE IMPLEMENTATION

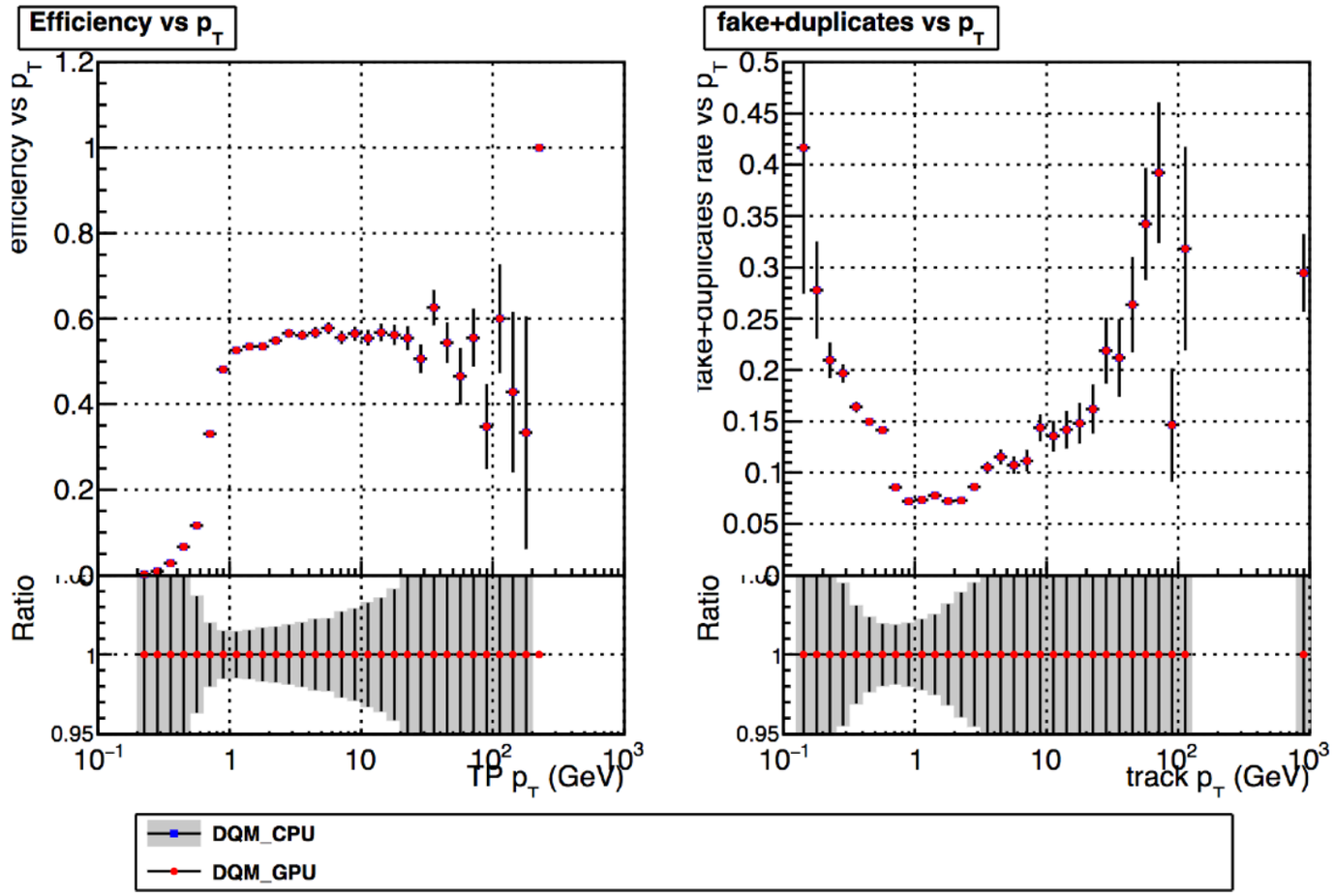
- **Main goal: reproduce what the CPU code does with a simpler and parallel implementation (CUDA) and try to speed up as much as possible the processing**
- **A fully working implementation of the pixel raw to digi algorithm on GPU is ready**
  - It unpacks x, y pixel coordinates and the corresponding adc count
  - It unpacks also FED errors
  - It is already integrated in the CMS framework analysis
- **How GPU parallel architecture is exploited**
  - Each FED is assigned to a block of threads
  - Words coming from a FED are saved in an array, copied to the GPU memory and assigned to the threads of the block where they are unpacked in parallel
  - Each thread executes the same set of instructions (kernel) on each word
- **Optimization of the memory usage and memory transfers for speeding up the algorithm (details in the backup slides)**



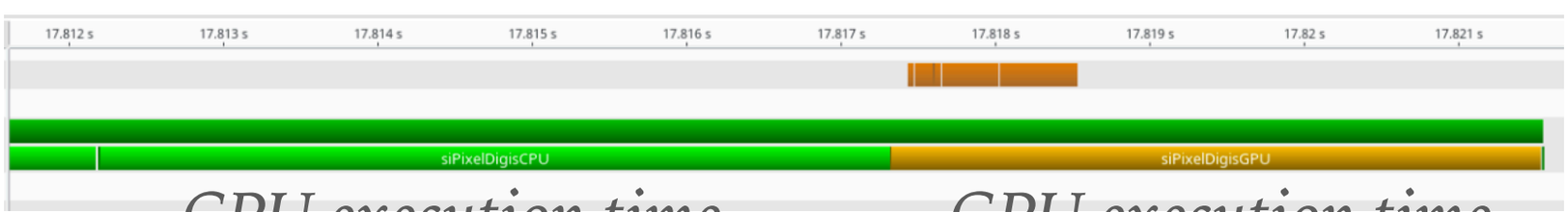


RAW2DIGI ON GPU: STATUS OF THE IMPLEMENTATION

- The GPU results have been validated against the CPU algorithm using the official validation plots
  - No differences between the two implementations
- We started to study the performance of the GPU with specific tool provided by NVIDIA
  - More optimization is possible
    - Reducing the time spent in the host-device and device-host memory copy
    - Optimizing the kernel
    - Maximizing the concurrency



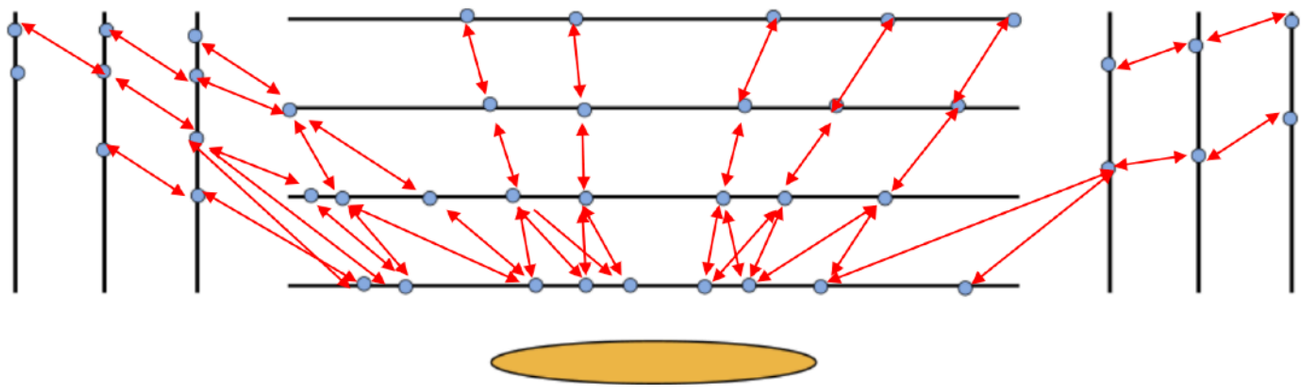
- We discovered almost all the time is spent on the CPU
  - We are also re-engineering the remaining part of the serial code (a.k.a. host code) to improve the performance and reduce the CPU execution time



CPU execution time *GPU execution time*

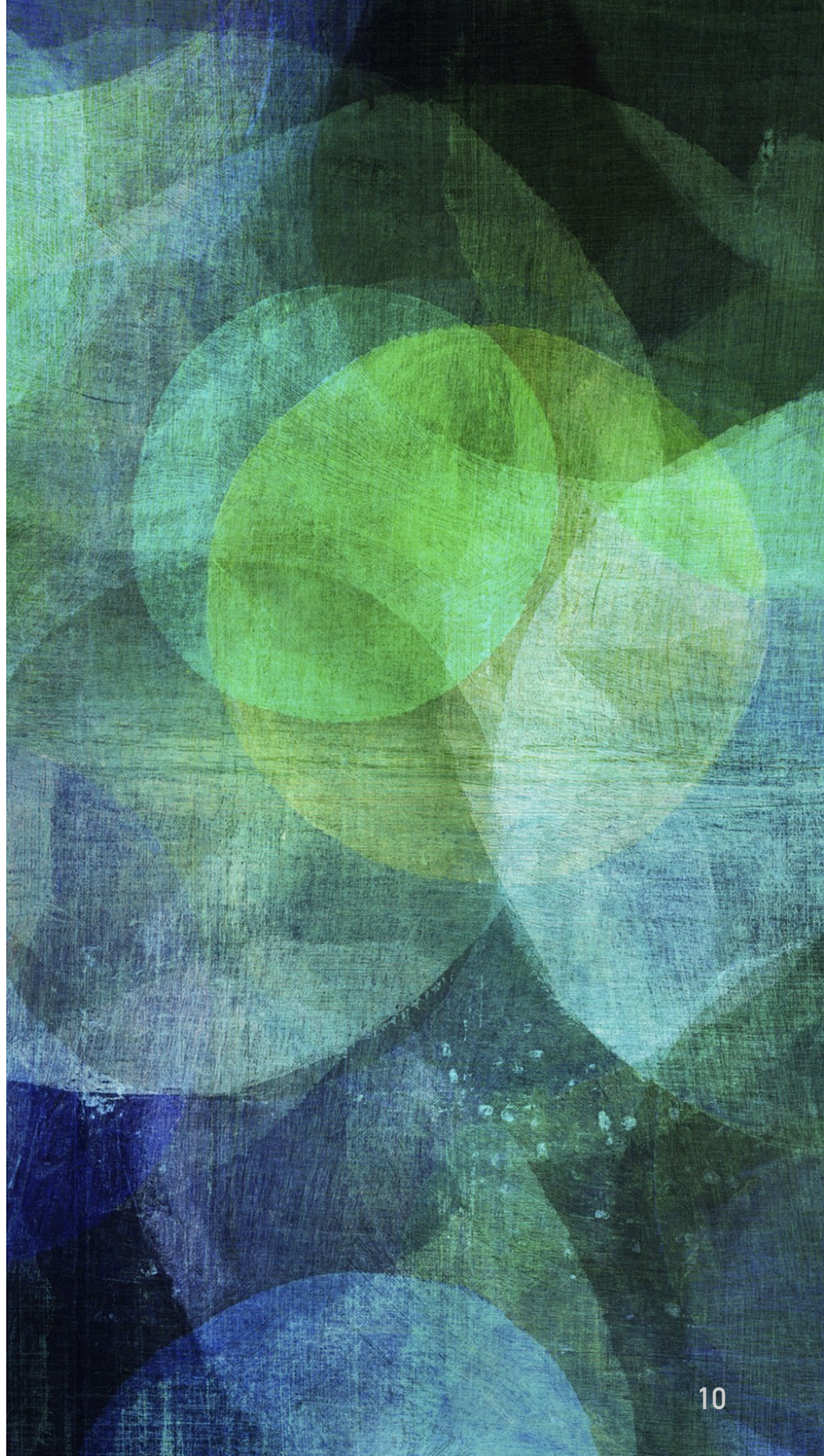
CONCLUSIONS AND NEXT STEPS

- **The work on the R2D step on GPU is completed**
- More refinements and improvements are possible, but those are left for the future
- **NEXT: develop a memory arena (in CUDA)**
  - A memory arena is simply a large, contiguous chunk of memory that is allocated once and then used to manage memory manually by handing out smaller chunks of that memory
  - This tool will be used to manage dynamically and efficiently the memory needed for saving the doublets used by the track seeding algorithm (for the quadruplet generation), since its number increases dramatically with PU



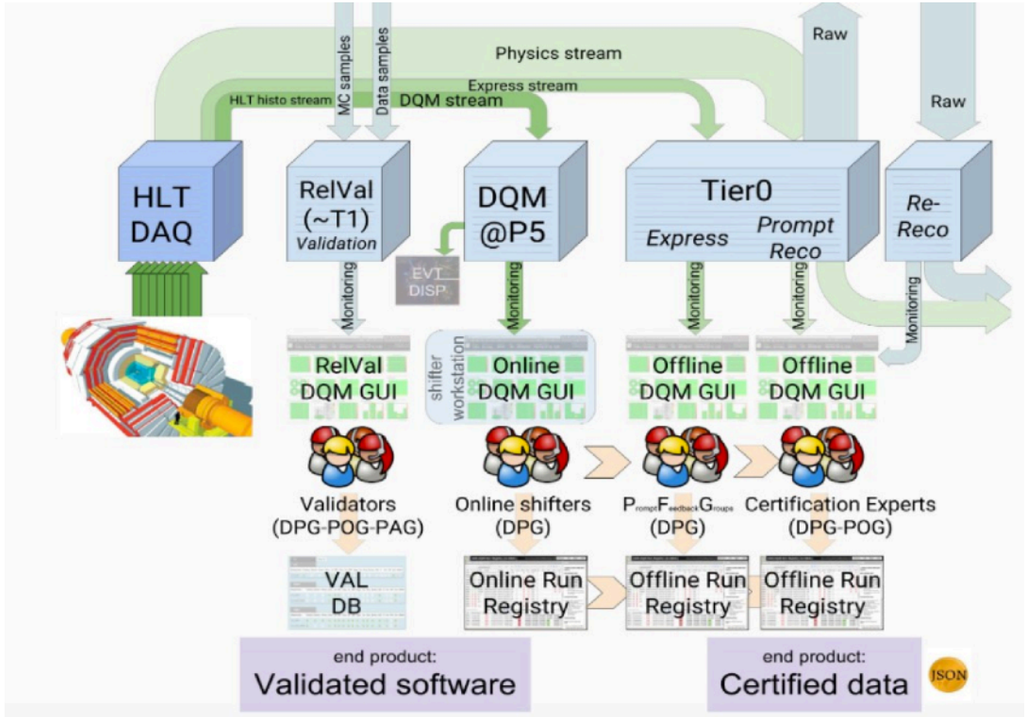
- **More infos about the Patatrack project at the link below:**
  - **https://patatrack.web.cern.ch/patatrack/**

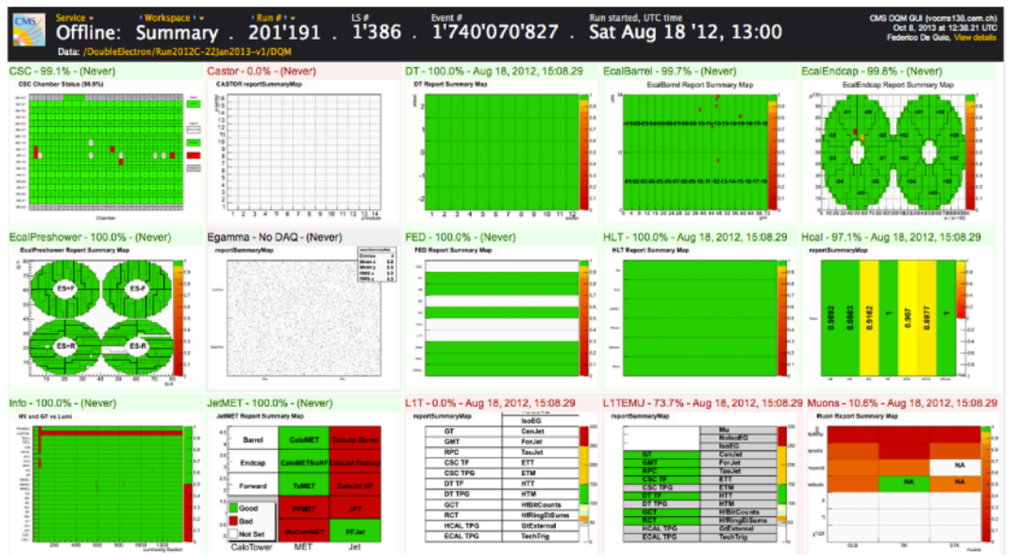
MACHINE LEARNING FOR THE CMS DATA QUALITY MONITORING



CMS DATA QUALITY MONITORING (DQM) SYSTEM

- **A critical asset to guarantee a high-quality data for physics analyses (online and offline)**
- **Online DQM assess data goodness and identifies emerging problems in the detector**
  - Data with poor quality is flagged by eyeballing DQM GUI and comparing a set of histograms to a reference good sample
- **Problems with current strategy:**
  - **Delay:** human intervention and tests require collecting sufficient statistics
  - **Volume budget:** amount of quantities a human can process in a finite time period
  - **Human driven decision process:** alarms based on shifter judgment
  - **Changing running conditions:** reference samples change over time
  - **Manpower:** the effort to train a shifter and maintain instructions

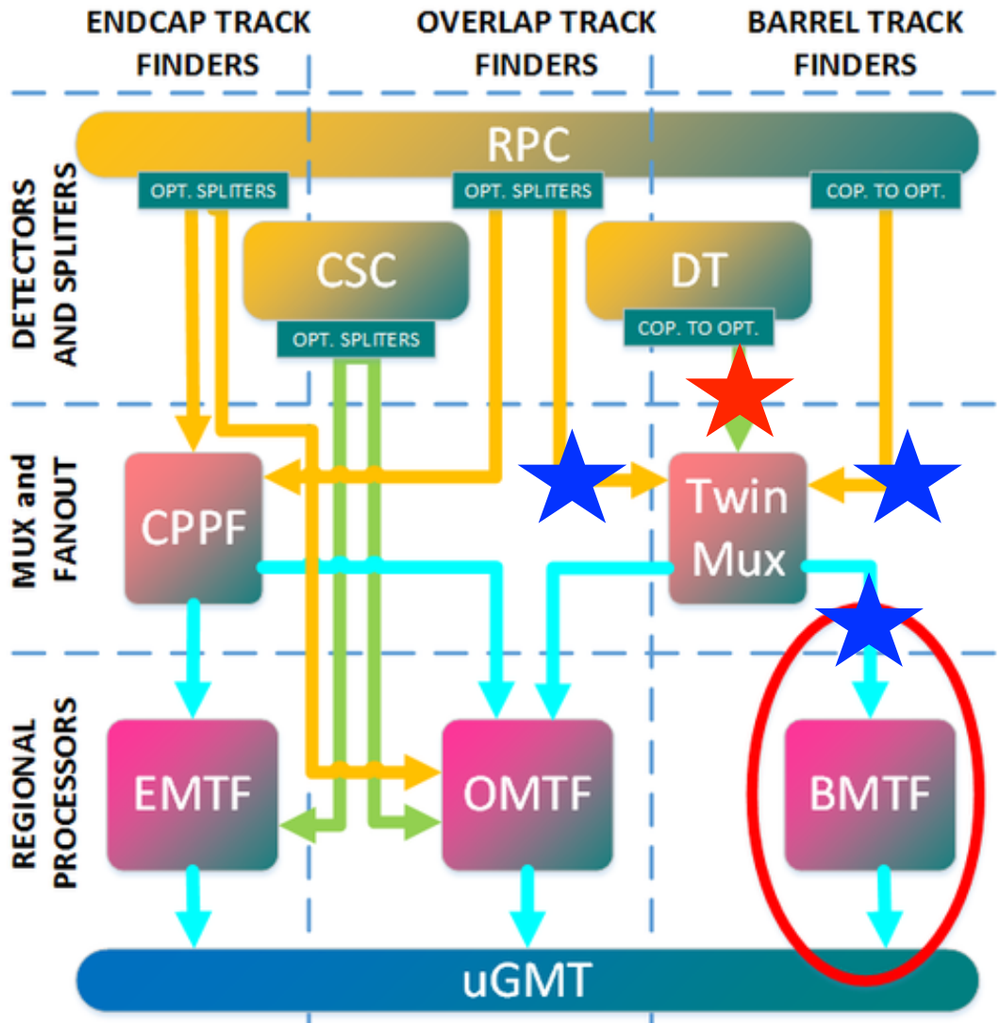




A TOOL FOR MONITORING THE L1 BARREL TRIGGER WITH ML

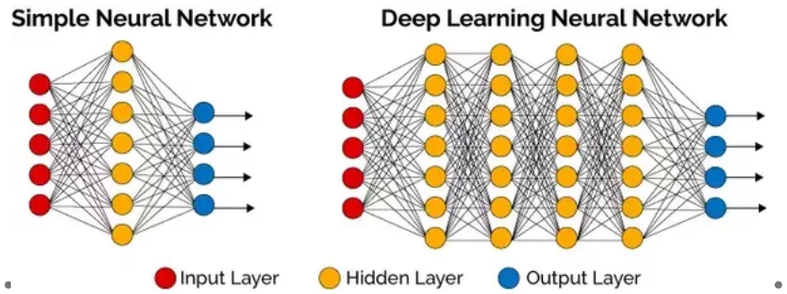
- **GOAL: Use ML/DL techniques for developing an innovative tool for the L1 Barrel Trigger rate monitoring in CMS**
  - To be run at the level of TwinMux inputs (DT and RPC inputs), TwinMux output and BMTF input
- The algorithm must:
  - correlate trigger rates and instantaneous luminosities coming from CMS database
  - identify chamber(s) with rate problem(s)
  - correlate different sources of information to make a diagnosis of the issue, e.g.:
    - all rates up to TwinMux output are in line with expectation for a given inst. lumi, but BMTF input is crazy ⇒ suspect communication issue between TwinMux Output and BMTF
- **Consumer: online operation teams**

Starting the project from the DT system!

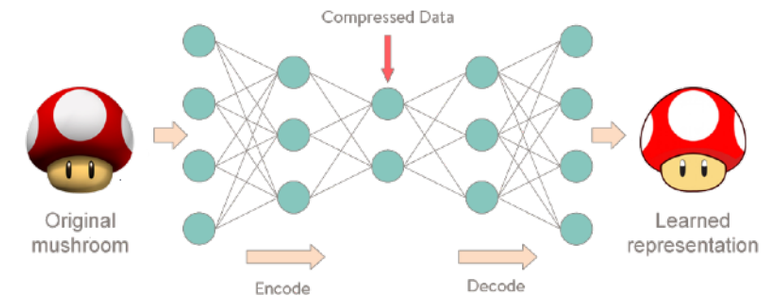


Details about the TwinMux and BMTF algorithm in the backup slides

ALGORITHMS



- **Input dimensionality: 10 features**
  - [system, wheel, sector, station, rate, rate uncertainty, inst. lumi., lumi/rate, uncertainty on ratio]
- **Building a deep neural network (DNN)**
  - Four hidden layers with 32 neurons
- **Trying an autoencoder (AE), i.e. a semi-supervised approach**
  - Only the sample with normalies is needed for the training
  - The network learns the features of good data in a processs of encoding-decoding
  - After that it should be able to re-reconstruct with some precision only the good data
  - Bottle neck leads to a dimensionality of 6 (from 10)
- **Testing on a DT known issue:**
  - Trigger board W+1, S4, MB3 is permanently off



Layer (type)	Output Shape	Param #
input_ann (Reshape)	(None, 10, 1)	0
flatten_ann (Flatten)	(None, 10)	0
dense_ann (Dense)	(None, 32)	352
dense_ann2 (Dense)	(None, 32)	1056
dense_ann3 (Dense)	(None, 32)	1056
dense_ann4 (Dense)	(None, 32)	1056
output_ann (Dense)	(None, 2)	66

Total params: 3,586
Trainable params: 3,586
Non-trainable params: 0

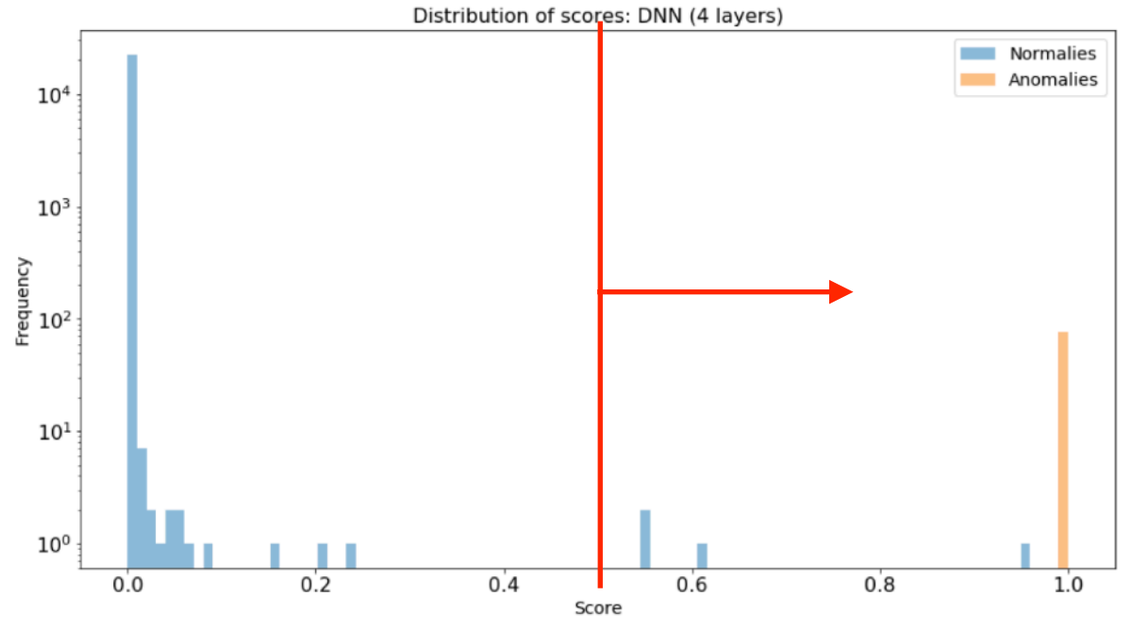
Autoencoder Architecture:

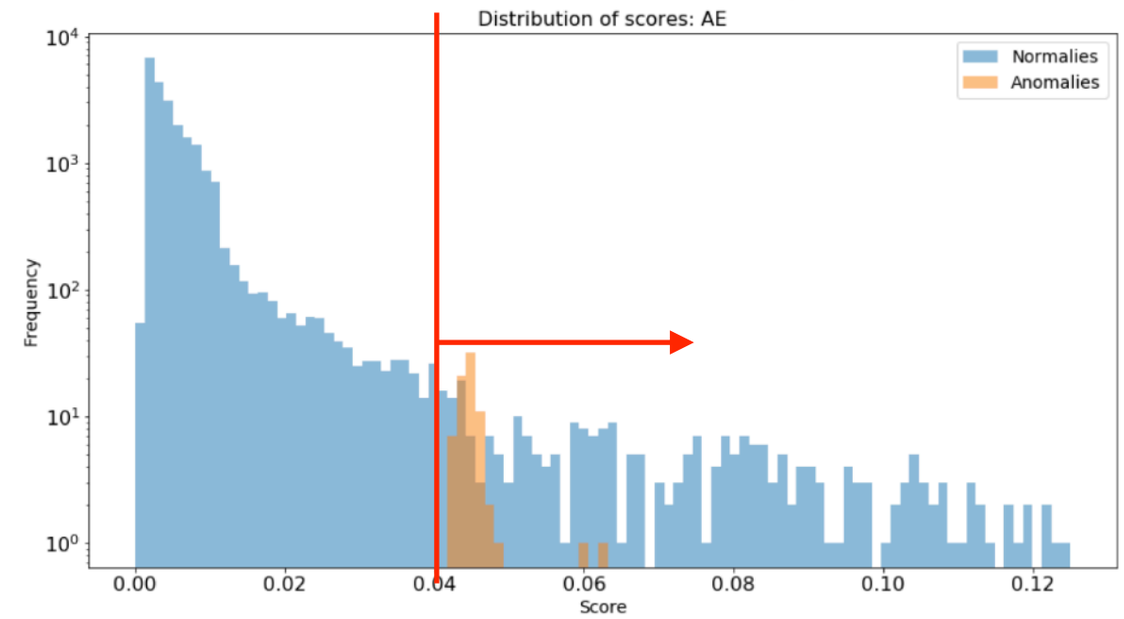
Layer (type)	Output Shape	Param #
input_29 (InputLayer)	(None, 10)	0
dense_281 (Dense)	(None, 10)	110
dense_282 (Dense)	(None, 9)	99
dense_283 (Dense)	(None, 8)	80
dense_284 (Dense)	(None, 7)	63
dense_285 (Dense)	(None, 6)	48
dense_286 (Dense)	(None, 6)	42
dense_287 (Dense)	(None, 7)	49
dense_288 (Dense)	(None, 8)	64
dense_289 (Dense)	(None, 9)	81
dense_290 (Dense)	(None, 10)	100

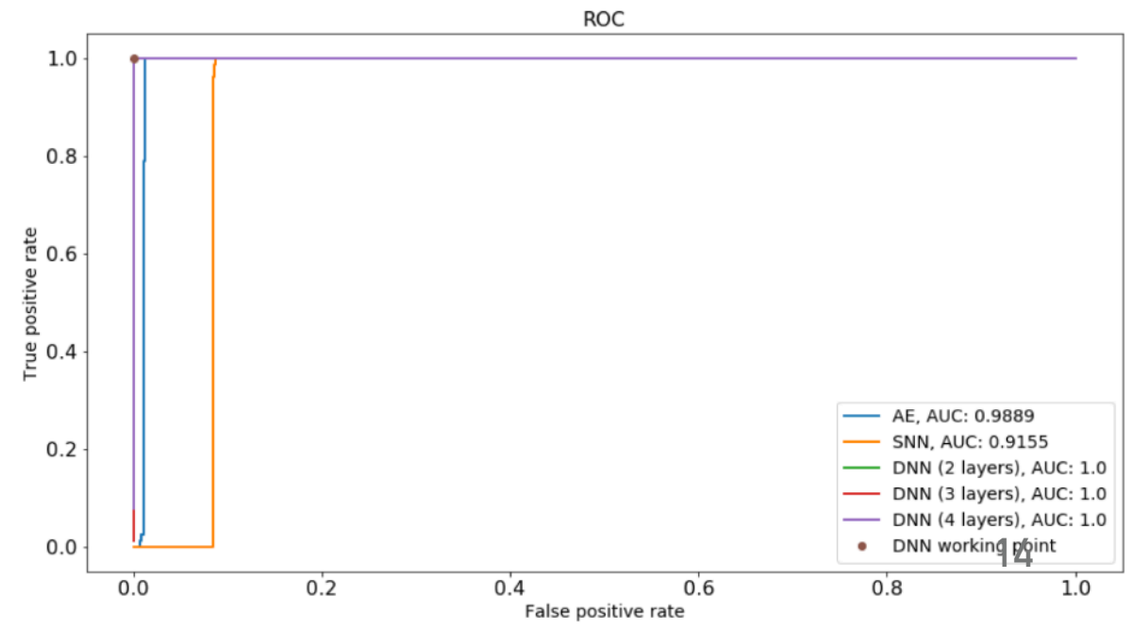
Total params: 736
Trainable params: 736
Non-trainable params: 0

PERFORMANCE

- **DNN shows very good separation between normalies and anomalies**
  - The DNN is able to classify in the correct way the normalies and anomalies in the test sample
  - Tried some options for the number of layers: good results are reached already with 2 layers
- **The classification with the AE is not clear as the DNN, anyway it is possible to fix a WP to recognize:**
  - 100% of the true positives with a 1% of false positives
  - AE approach seems promising:
    - no need to provide labels for anomalies
    - it can spot unforeseen problems
- Tried also some other classic approaches, but DNN and AE provide the best performance (see backup slides)

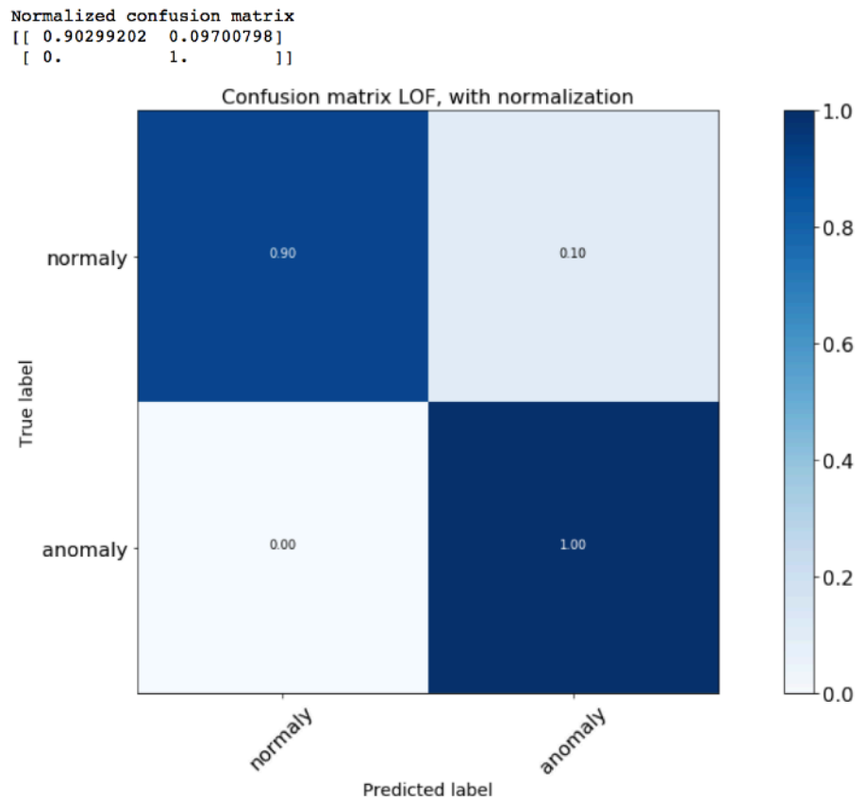


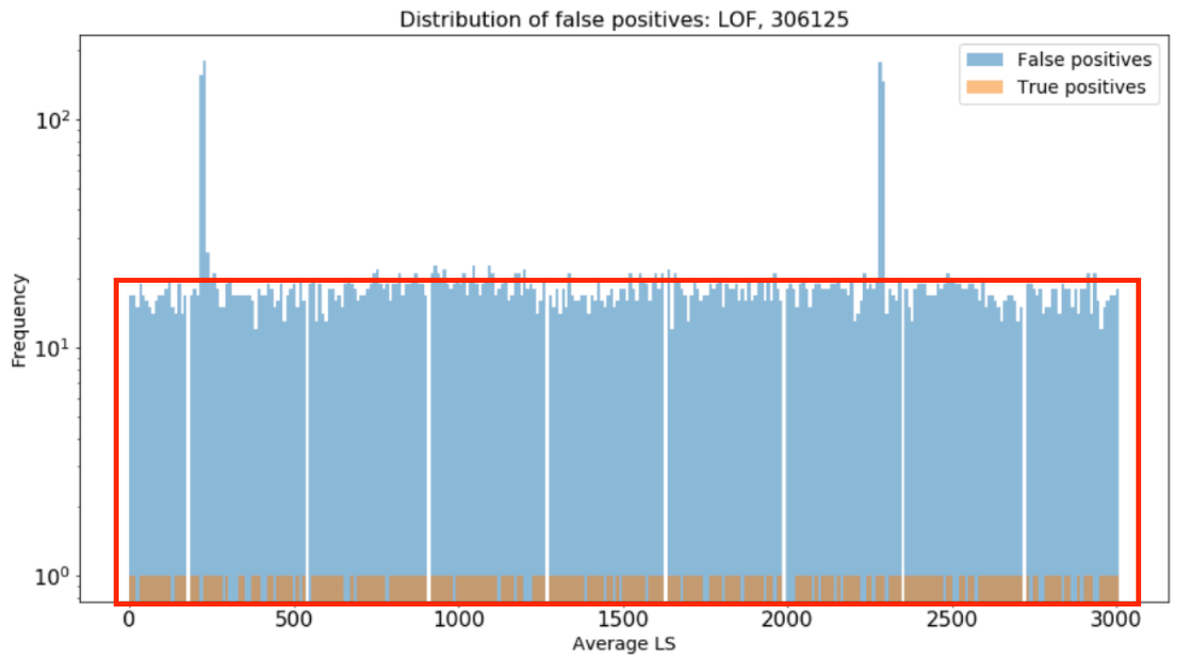




FURTHER ALGORITHMS: GOING COMPLETELY UNSUPERVISED

- **Moving to completely unsupervised algorithms is the best approach in order to detect all the possible anomalies without specific trainings in general performed over a limited number of anomalies that will never cover all the possible cases**
- **Local outlier factor (LOF)**
  - Based on k-NearestNeighbor algorithm
  - It detects all the known anomalies, but with too many false positives (10%):
    - < 2 FPs per LS, but still too much
  - Details here: http://scikit-learn.org/stable/modules/outlier_detection.html and in the backup slides
- **Looking into clustering algorithms (like K-Means clustering)**
  - Started working on it but the method and the results need to be better understood
  - Details here: http://scikit-learn.org/stable/modules/clustering.html#k-means and in the backup slides



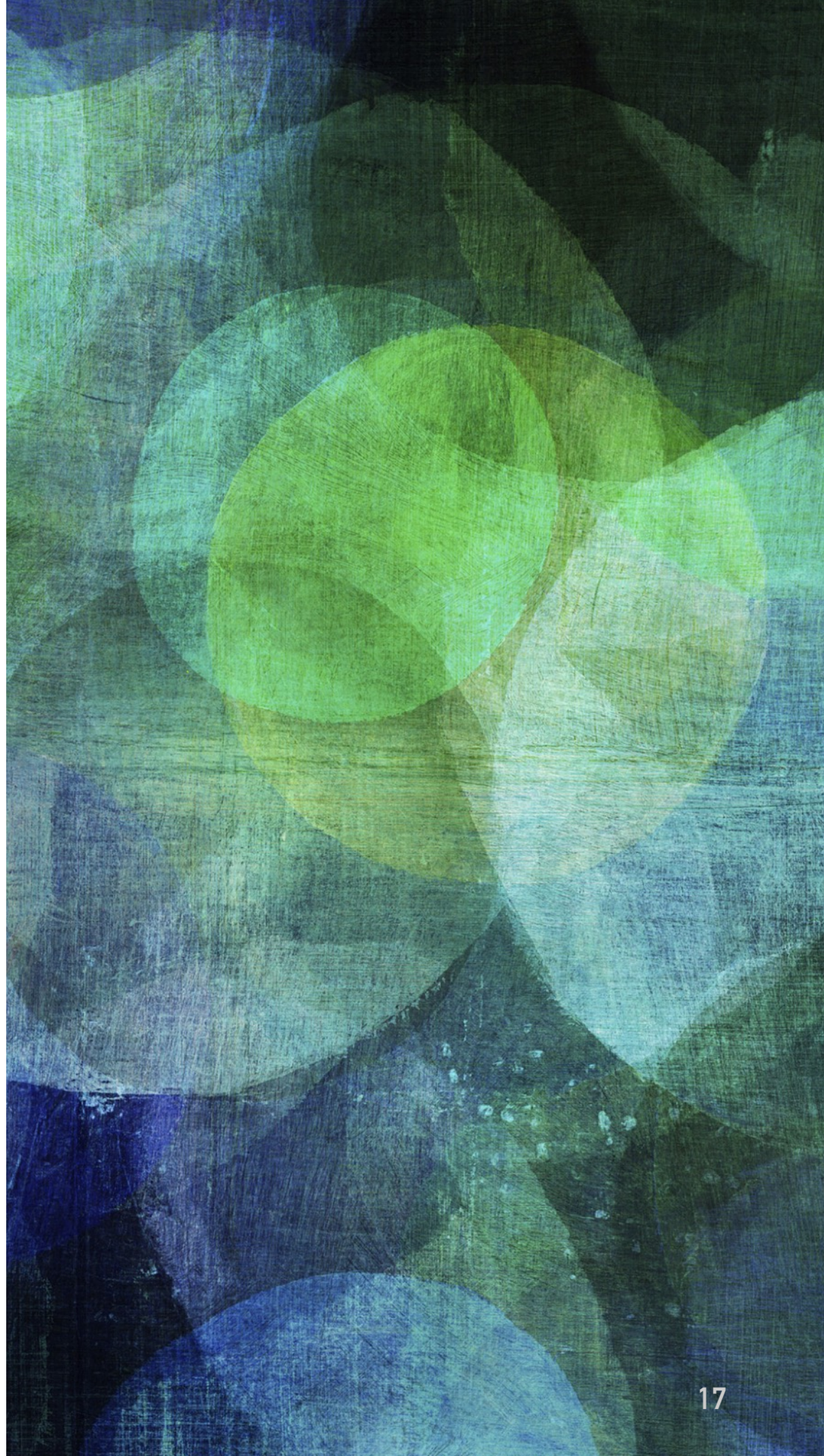


CONCLUSIONS AND NEXT STEPS

- DNN is performing very well, but it is strongly specialized on the issue it is trained on
- AE is also performing very well and it is promising for the detection of generic/unknown issues with a reasonable level of false positives
  - **NEXT: Test DNN and AE on further DT anomalies**
- Unsupervised learning is desirable because it can offer the chance to spot unforeseen problems, but it needs to be studies and understood better
  - **NEXT: Work on other algorithms, mainly unsupervised**
  - **NEXT: Extend the R&D project**
    - Extend the development to RPC in order to be able to check completely the TwinMux inputs
    - Extend the development to the TwinMux output
    - Cross the two informations and create a standalone monitoring tool able to determine the origin of the anomalies

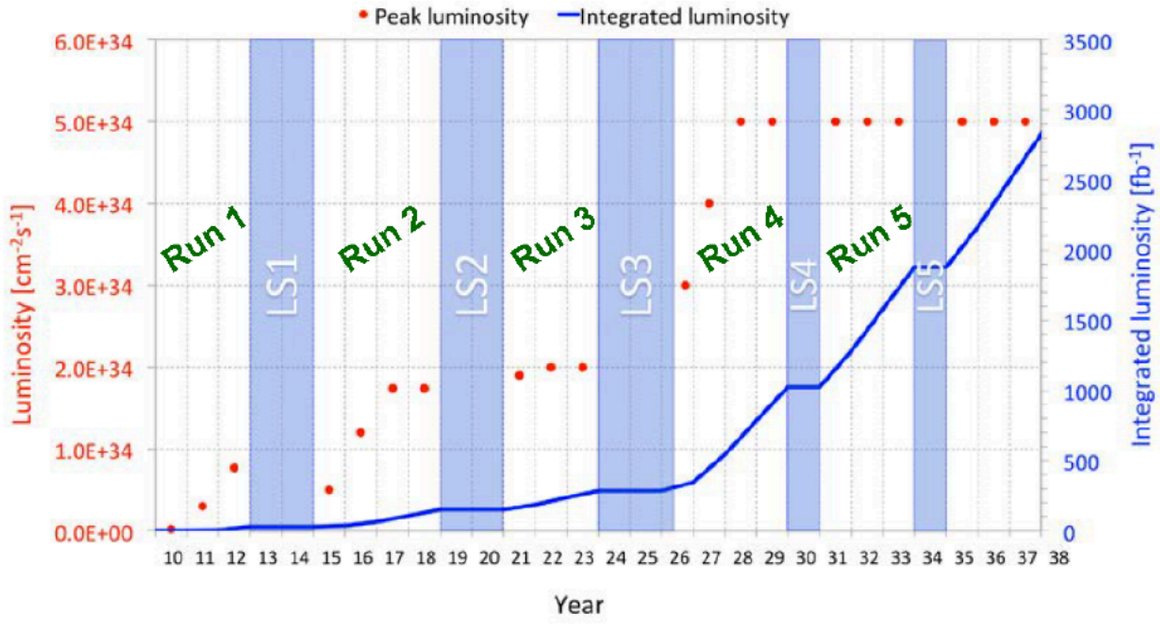
- **The python scripts to access the database and the jupiter-notebook used to train the model can be found here:**
  - **https://github.com/calabria/DTTriggerRateMonitoringWithML**

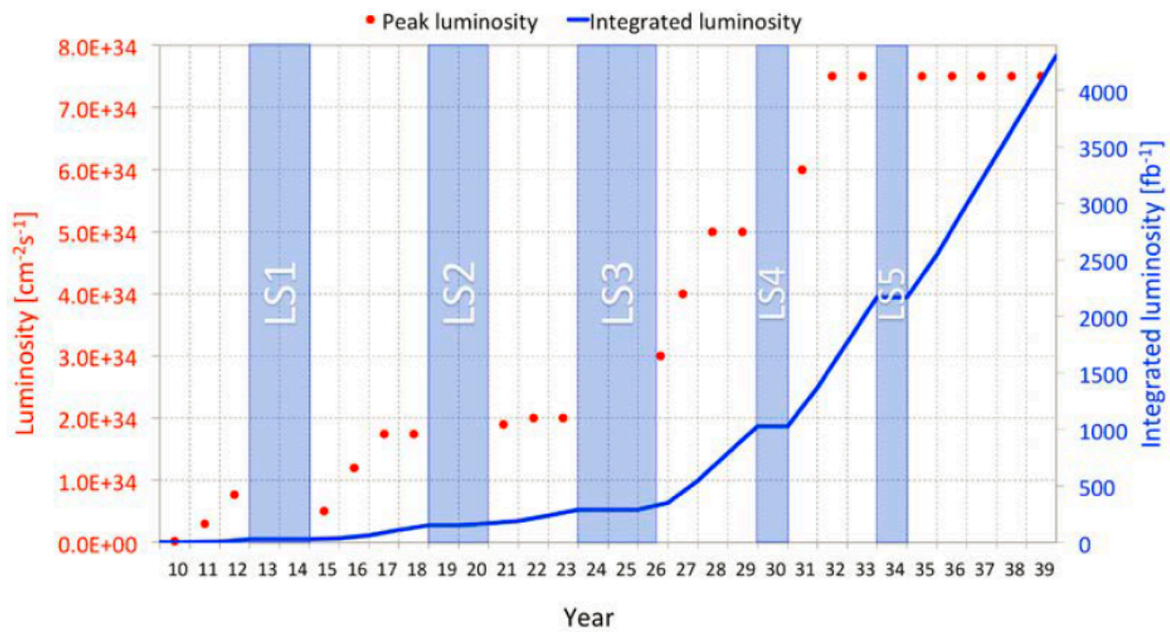
BACKUP



FUTURE CHALLENGES FOR HL-LHC

- **Higher instantaneous luminosity and consequently pile-up (up to 200 interactions per BX)**
  - Higher event size
  - More time needed for the pattern recognition algorithms
  - Increasing computing power both for the online selection and for the offline reconstruction
- **How we managed in the past:**
  - Increase computational and storage resources
  - There will not be financial resources to support this!
- **More improvements will have to be found in algorithm speed, by a combination of smarter algorithms and by making better use of parallel architectures, for instance:**
  - GPU accelerators and massive parallel programming
  - Machine learning algorithms
  - Performance tuning and software engineering

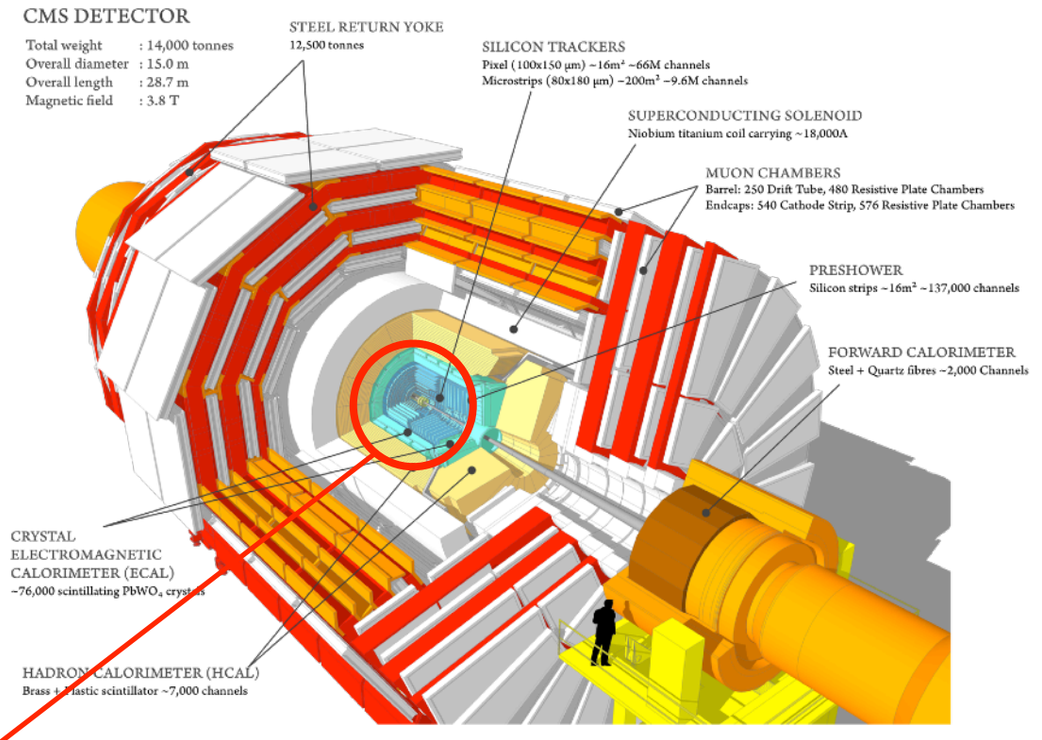


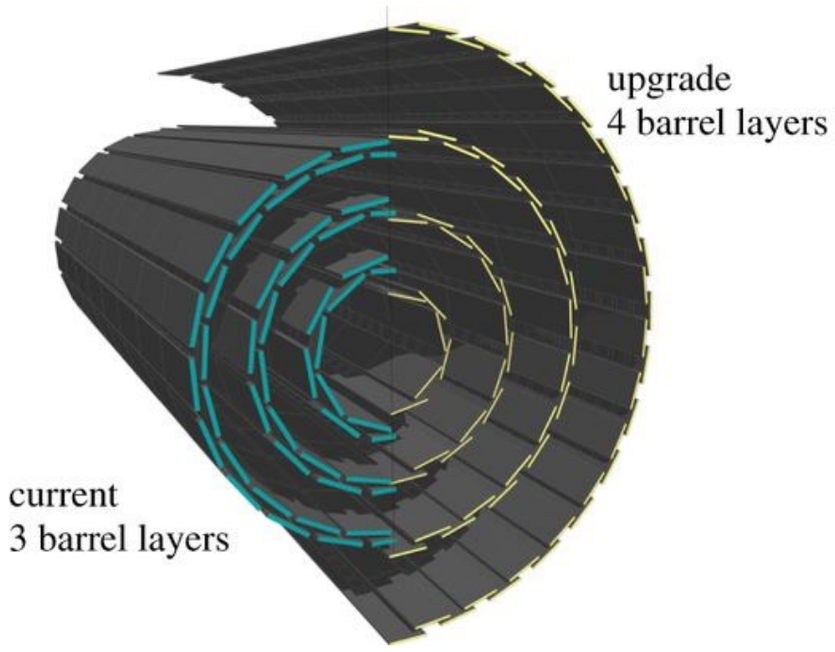


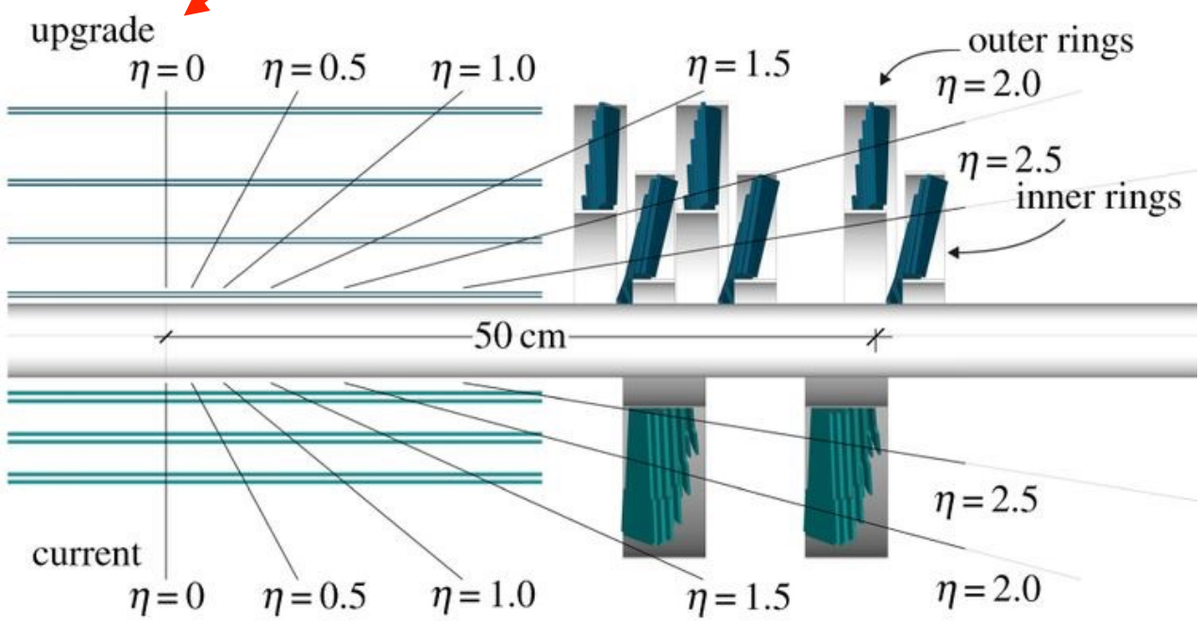
	LHC design	HL-LHC design	HL-LHC ultimate
peak luminosity (10^{34} cm^{-2}s^{-1})	1.0	5.0	7.5
integrated luminosity (fb^{-1})	300	3000	4000
number of pileup events	~30	~140	~200

THE CMS (PHASE-1) PIXEL DETECTOR

- n-on-n silicon sensor thickness: 300 µm
- Pixel size: 100 x 150 µm
- Four barrel layers instead of current three
- 3-disk forward system instead of current 2-disk
- Total Modules: 1856 (1184 + 672)
- Total Pixels: 124 million (79 M + 45 M)





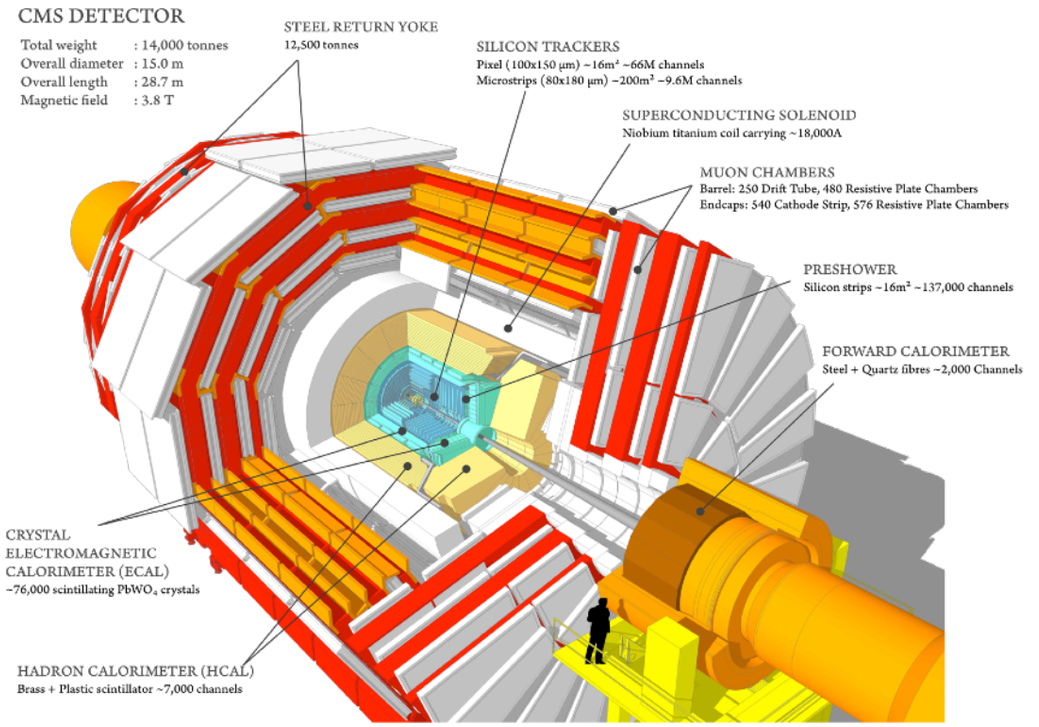


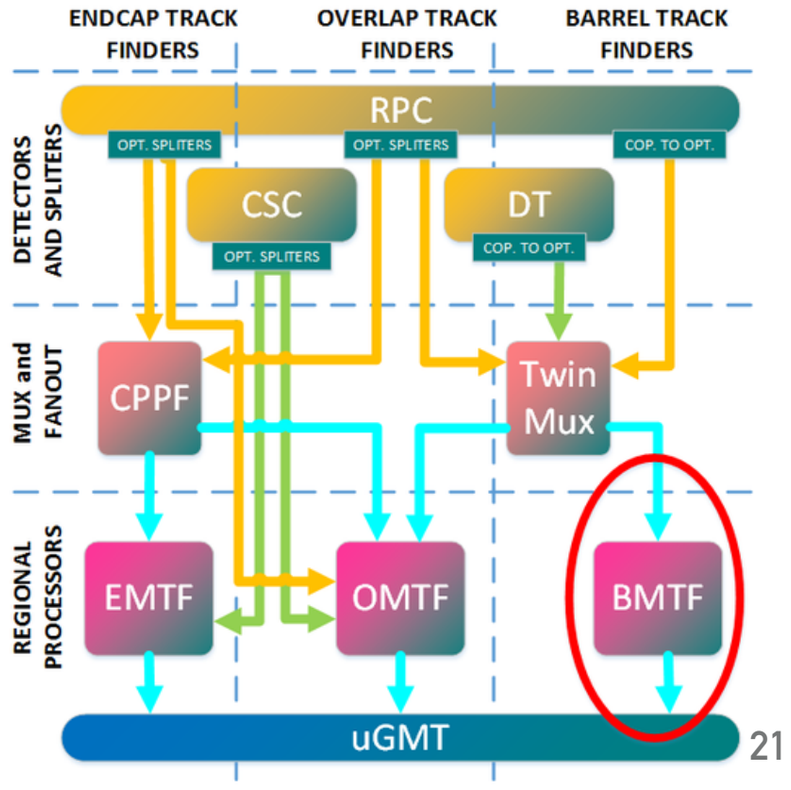
RAW2DIGI ON GPU: STATUS OF THE IMPLEMENTATION

- **Some details about the GPU implementation:**
  - **New GPU-friendly cabling map improves speed**
    - A GPU-friendly cabling map (basically a LUT) is generated and updated if it changes in the next event and copied again to the GPU memory
  - **Errors are treated and unpacked as in the serial code**
    - All the functions to check the status of the pixel rocs and recognize the type of error have been implemented as device functions (part of the kernel)
    - The possibility to exclude bad pixels and specific regions of the detector is also implemented
    - To this end the cabling map was also extended with the list of modules to unpack and the list of pixel bad rocs for the error unpacking
- **Optimized memory reserved for each kind of device array**
- **Optimized memory transfers packing the digi informations on the device and unpacking on the host avoiding to copy several arrays**
- **Using a GPU-friendly vector class instead of several arrays for the error unpacking**

THE CMS LEVEL-1 TRIGGER BARREL TRACK FINDER

- The muon barrel architecture groups the muon detectors in 12 wedges. Each wedge has five sectors and each sector, 4 DT detectors and 3 RPC
- The front-end electronics record muon primitives and send them to the TwinMux which concentrate data from different sectors. The TwinMux combines DT and RPC to create more reliable primitives which are called superprimitives. Then it fanout the data to the barrel and the overlap track finders
- The BMTF receive muon primitives from the DT and RPC detectors from the Barrel area of CMS ($|\eta|<1$)
  - The data primitives give muon coordinates, bending angle as well as quality bits that are used to evaluate the inputs
- The BMTF algorithm use the information to represent muon tracks and calculate physical parameters like the transverse momentum (pT), the total bending angle the quality of the track and the track addresses
- Each BMTF processor search for muon tracks in one wedge (own wedge) which may go also to the neighbor wedge (left and right)
- The algorithm runs in parallel for 2 muons in 6 sectors which correspond to 1 wedge (the sectors are 5 but the logic splits the middle to two). In the barrel there are 12 wedges. So it can find 2 x 6 = 12 muon tracks
- Every BMTF processor has a sorting logic which give the best 3 muons of the 12 possible tracks





A SIMPLE TEST CASE FOR DT TRIGGER RATE

- Testing some simple neural networks on a known issue:
  - Trigger board W+1, S4, MB3 is permanently off
- A supervised approach needs samples of normalies and anomalies for the training:
  - Take runs certified as “good” runs
  - Anomalies come for free in some sense, since the trigger board is always off
  - One can build the sample of normalies by exploiting the symmetry of the system and forcing the rate to the one of the symmetric trigger board, in this case: W-1, S3, MB3
    - [system, wheel, sector, station, rate, rate uncertainty, inst. lumi., lumi/rate, uncertainty on the ratio]

```
Normal chimney:
[2, -1, 3, 3, 4032.193, 53.42, 17987.064, 173.3344, 4.4609, 0.0731]
Anomalous chimney:
[2, 1, 4, 3, 3071.4035, 40.4563, 17987.064, 173.3344, 5.8563, 0.0956]
```

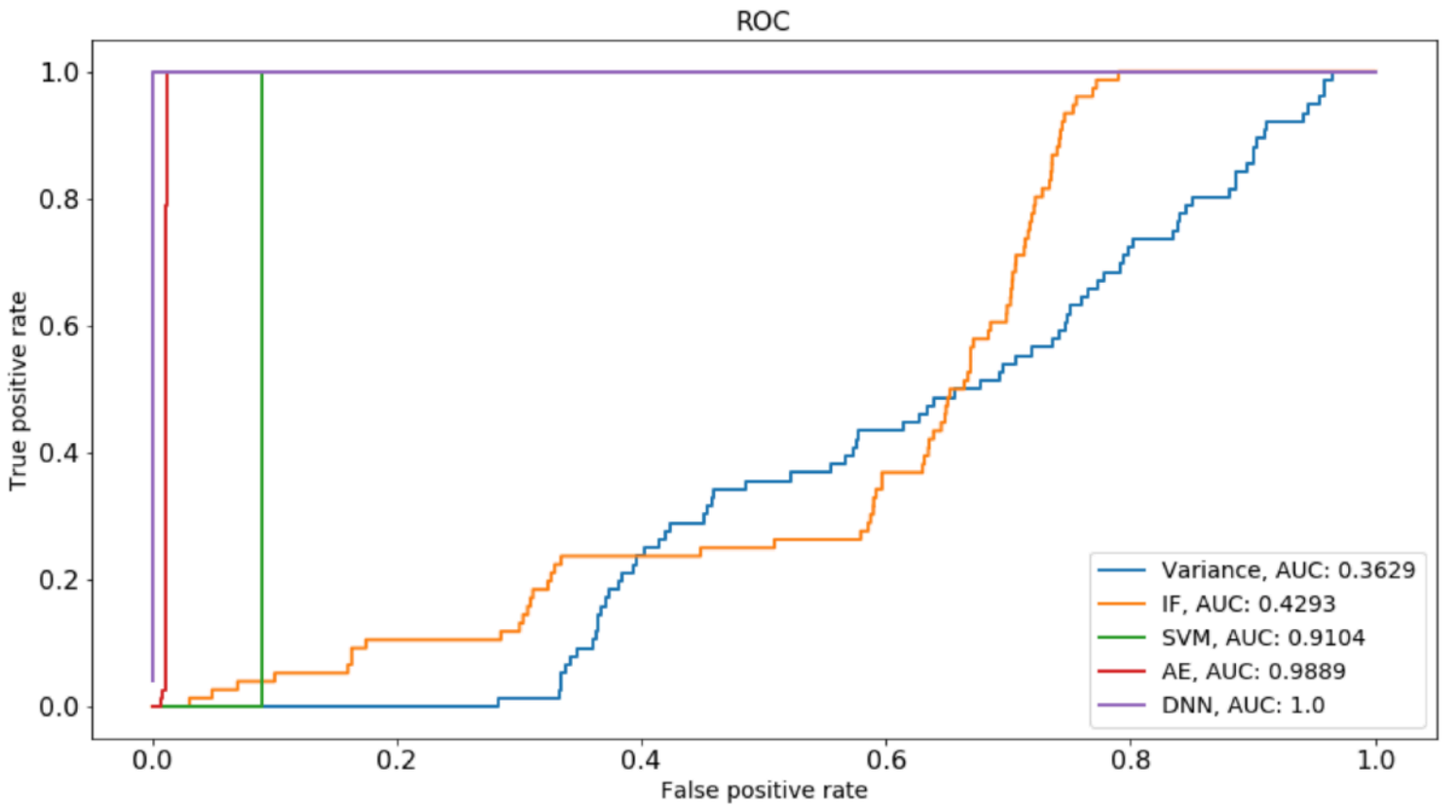
- Number of normalies and anomalies considered in this exercise
  - Very few faults, so anomalies and normalies are strongly unbalanced
    - Weighting properly anomalies and normalies
  - 20% of the data are reserved for the test

METRIC FOR PERFORMANCE EVALUATION

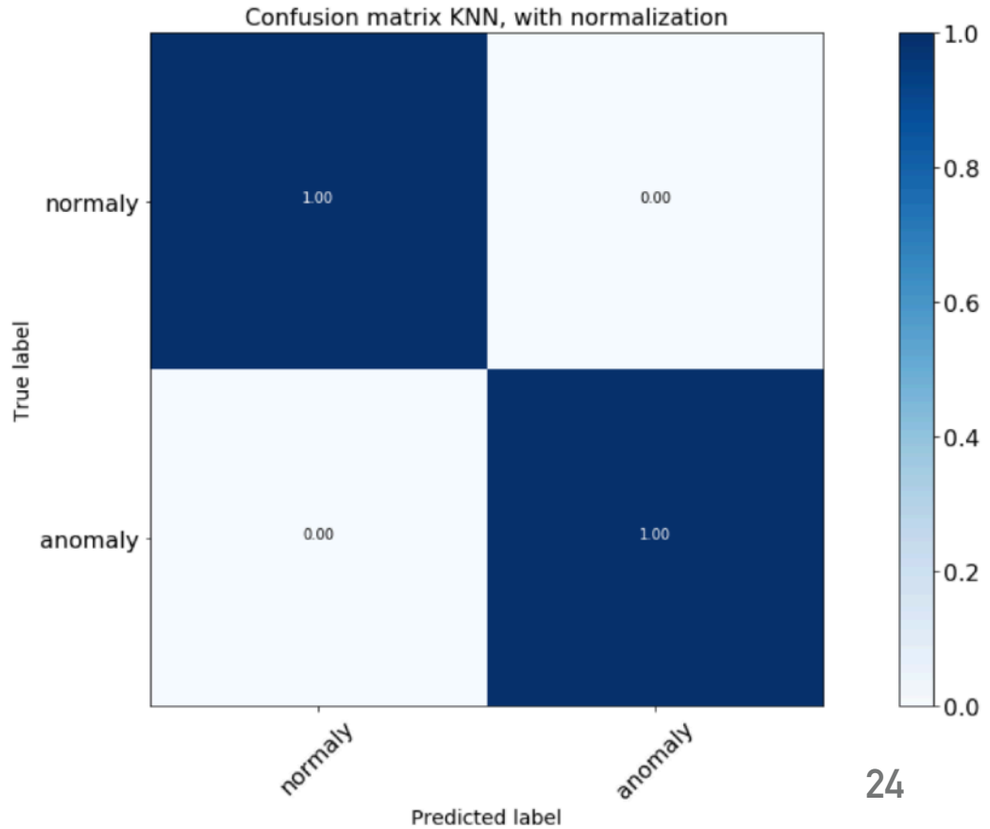
- **TP (true positive) is a correctly classified fault, while TN (true negative) is a correctly classified normal observation**
- Sensitivity TP/P: to keep high, i.e. maximize detection
- Specificity TN/N: to keep high, i.e. minimize false alarms
- Fall-out FP/N (1-Specificity): to keep low, i.e. minimize false alarms
- Receiver Operating Characteristic (ROC) curve and its Area Under Curve (AUC):
  - illustrates the performance of different classifiers when discrimination threshold is varied
- Deciding on the penalty of a false alarm versus false negative (or upper-bound false alarms) will be an essential in final implementation steps

COMPARISON WITH SOME BENCHMARK ALGORITHMS

- **Statistical: variance (probably not the best for facing this kind of problem)**
- **Outlier detection algorithms:**
  - Details : http://scikit-learn.org/stable/modules/outlier_detection.html
  - Classical machine learning: OneClassSVM (SVM) (need to perform a complete grid search for best parameters)
  - Unsupervised: Isolation Forest (IF)
- **Supervised nearest neighbor classifier (KNN): it performs very well but it is still a supervised approach**
  - Details: http://scikit-learn.org/stable/modules/neighbors.html#classification
- **DNN and AE still remains the best algorithms**



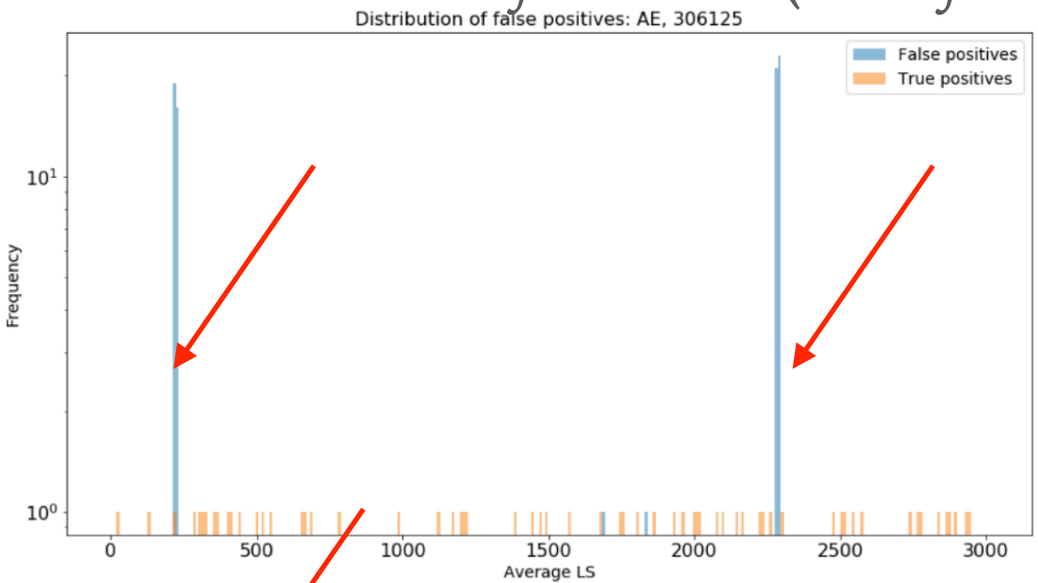

```
Normalized confusion matrix
[[ 1.  0.]
 [ 0.  1.]]
```

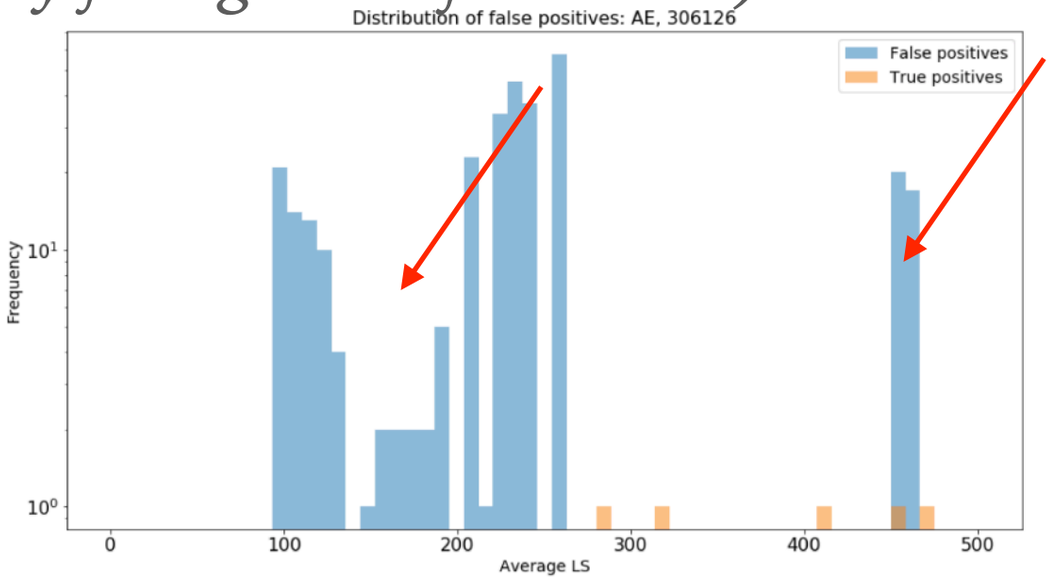


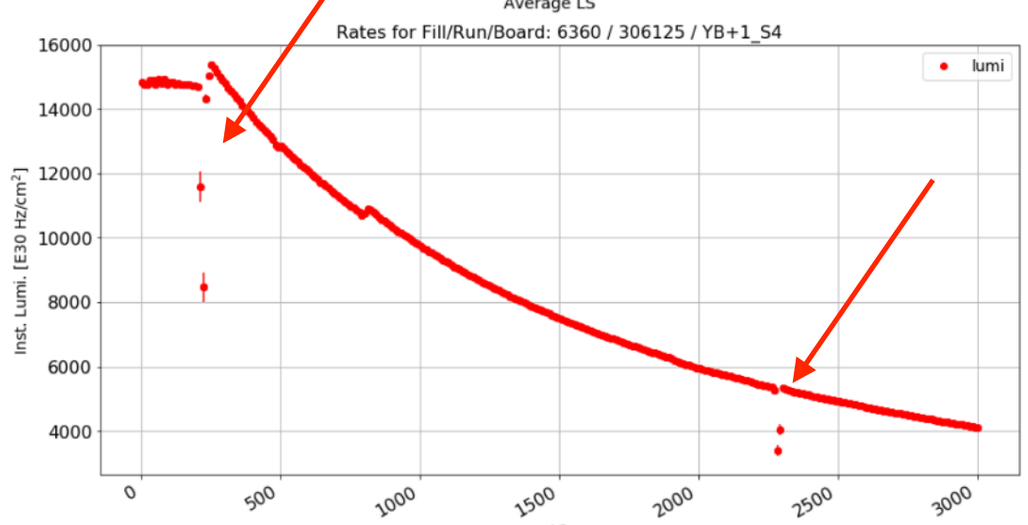
WHY IS THE AUTOENCODER APPROACH PROMISING?

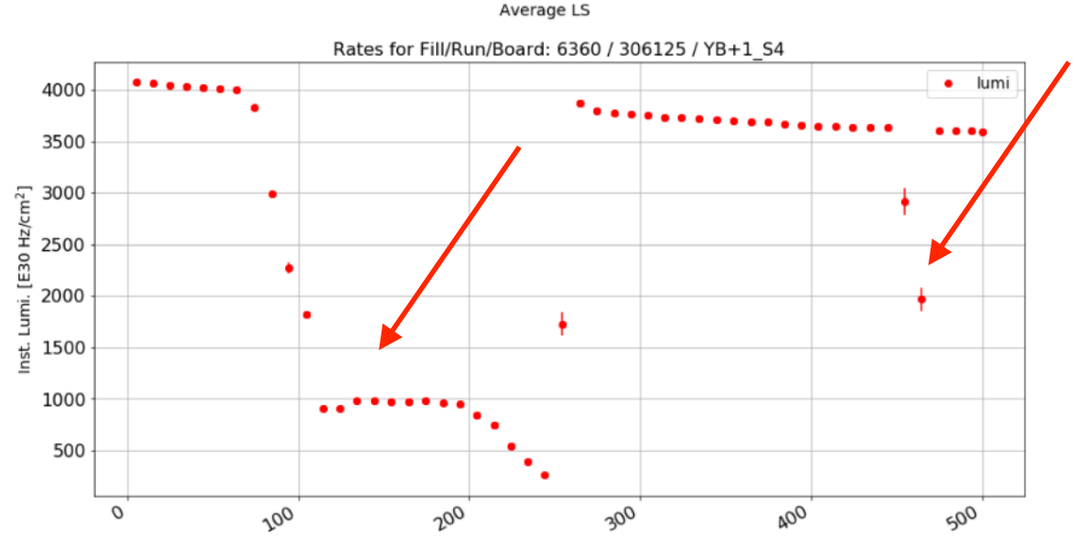
- DNN is performing very well, but it is trained against a specific issue
- AE is trained only using good data, so in principle is able to spot any kind of problem
  - AE is able to spot some luminosity oscillations during the fill, recognized as anomalies
  - This feature is not seen by DNN, since it is strongly specialized to find one type of issue

Distributions of the FPs (as defined by fixing a WP for the AE) vs. LS









SOME DETAILS ABOUT THE UNSUPERVISED ALGORITHMS

- **The Local Outlier Factor (LOF) algorithm** computes a score (called local outlier factor) reflecting the local density deviation of a given data point with respect to its neighbors
- The idea is to detect the samples that have a substantially lower density than their neighbors
- The LOF score of an observation is equal to the ratio of the average local density of his k-nearest neighbors, and its own local density: a normal instance is expected to have a local density similar to that of its neighbors, while abnormal data are expected to have much smaller local density

- **The KMeans algorithm** clusters data by trying to separate samples in n groups of equal variance, minimizing a criterion known as the inertia or within-cluster sum-of-squares
- This algorithm requires the number of clusters to be specified
- The k-means algorithm divides a set of samples into disjoint clusters, each described by the mean of the samples in the cluster (called the cluster “centroids”)
- The k-means algorithm aims to choose centroids that minimize the inertia that can be recognized as a measure of how internally coherent clusters are