Istituto Nazionale di Fisica Nucleare
SEZIONE DI FIRENZE

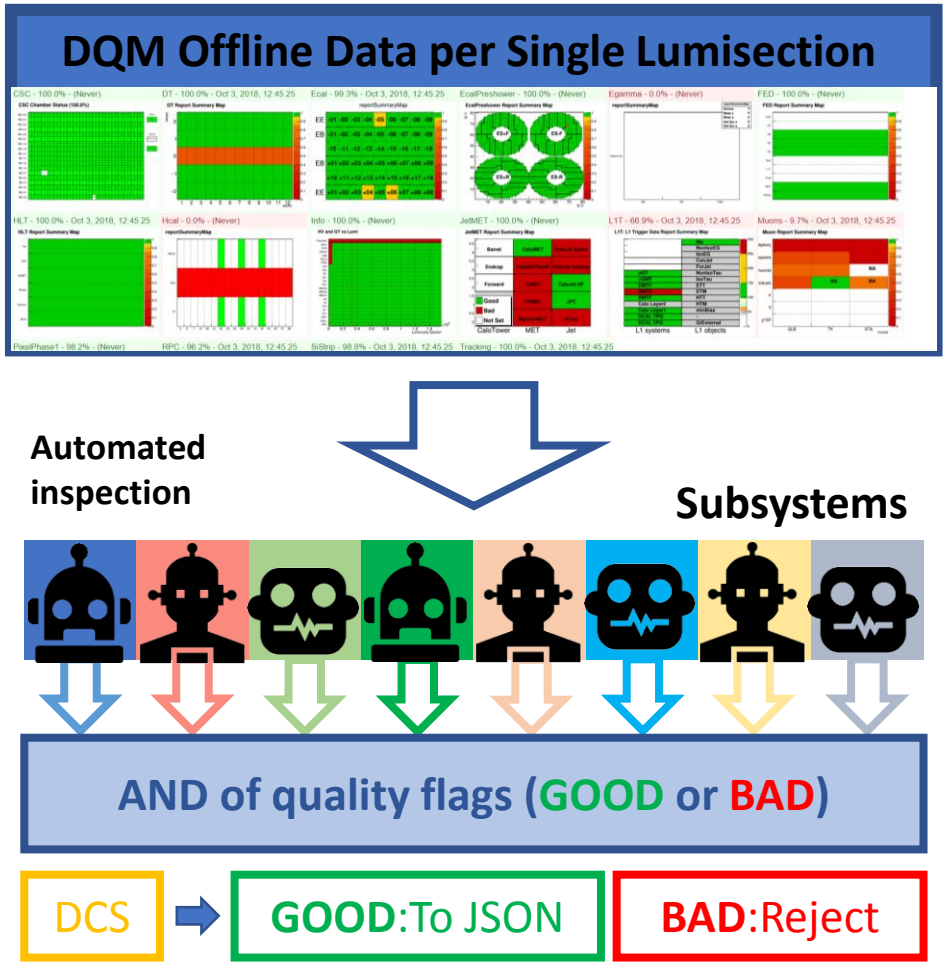
Machine Learning Applied to Data Certification: Status and Plans

F.Fiori *(INFN Florence)*

on behalf of the CMS DQM-DC Team

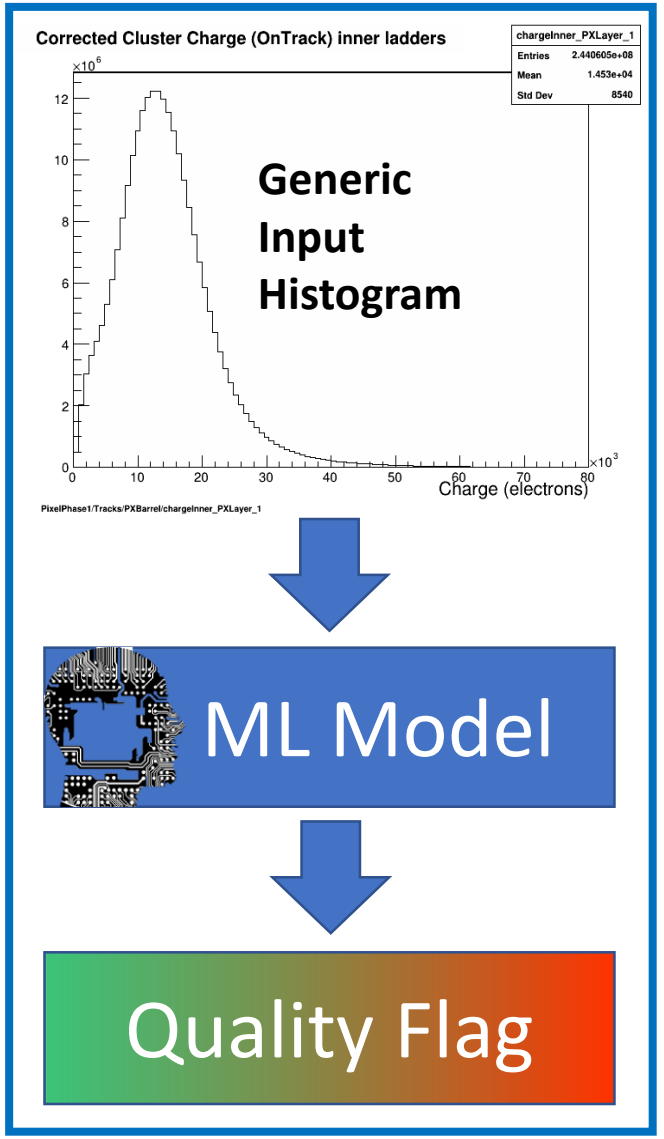
Main reasons to automate the DC process

- **Save expert time**
  - **Around 50-70 people involved in the full process!**
- Provide DC results with **single Lumisection (LS) granularity**!
  - Done per Run at the moment
- Goal to **provide a set of shared tools** based on ML to help subsystems in DC automation
  - Along with Documentation and extensive test results
- See CMS-Week talk for more details



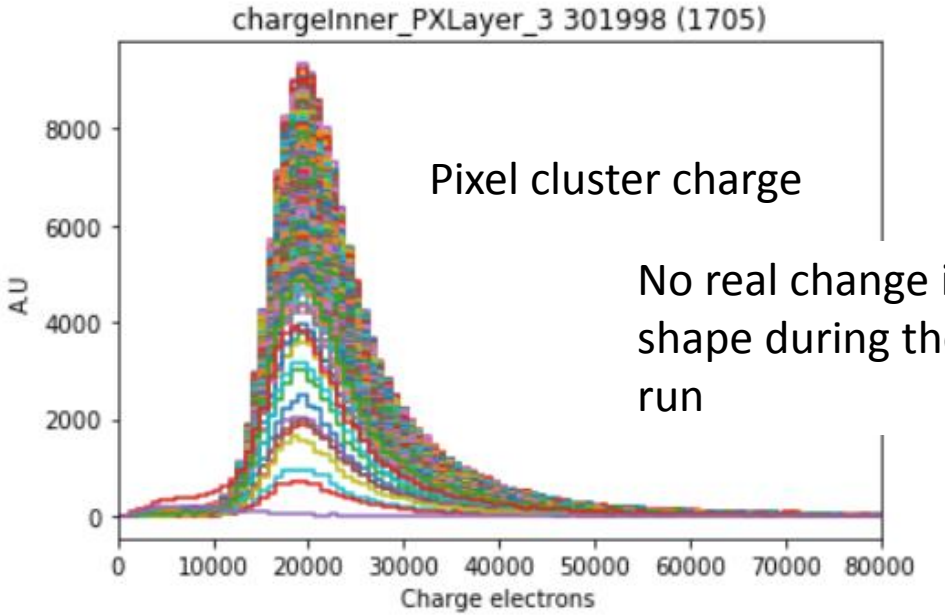
Plan to automate DC using ML

- Automate the “elementary action” in DC: **establish quality of single Histograms** (per Lumisection)
- **Semi-supervised or Unsupervised models preferable**
  - True labels not available, Class imbalance
- **Proposed design features**:
  I. Keep the model as simple as possible
  II. The same model should generalize well to different types of histograms
  III. The outcome should be **human interpretable**
  IV. **Sensitivity only to significant anomalies not to small effects (as DC is)**

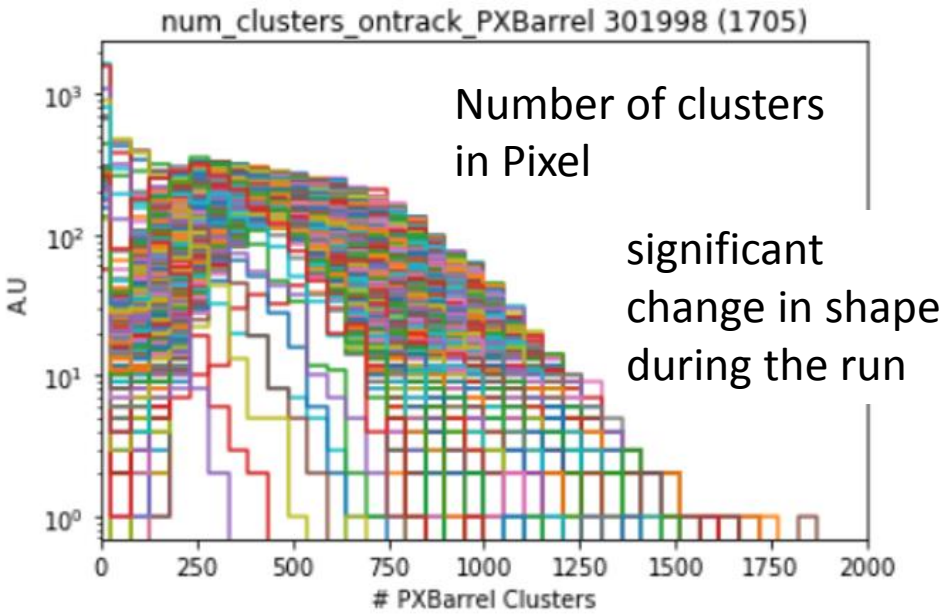


Two main kind of histograms

Representing a **quantity**



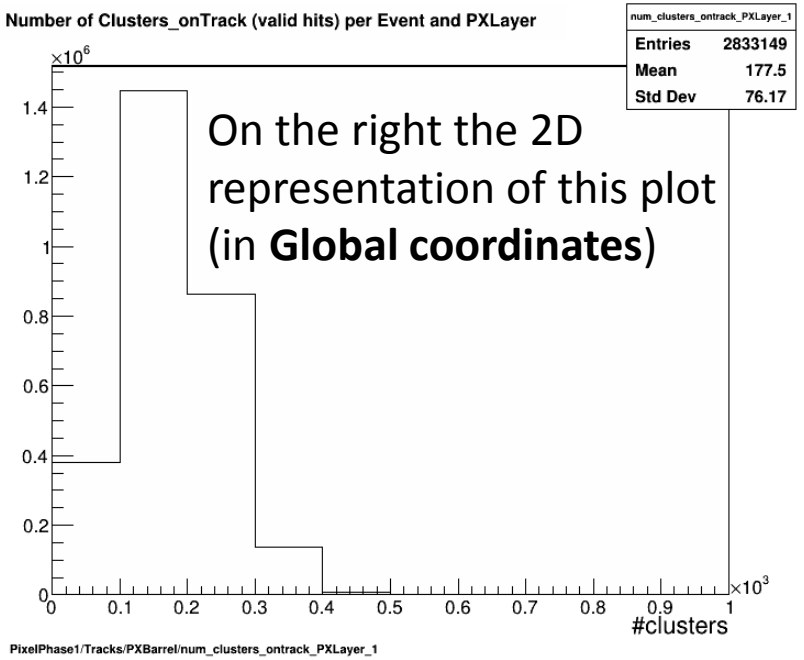
Representing a **the number of objects (aka: occupancy plots)**

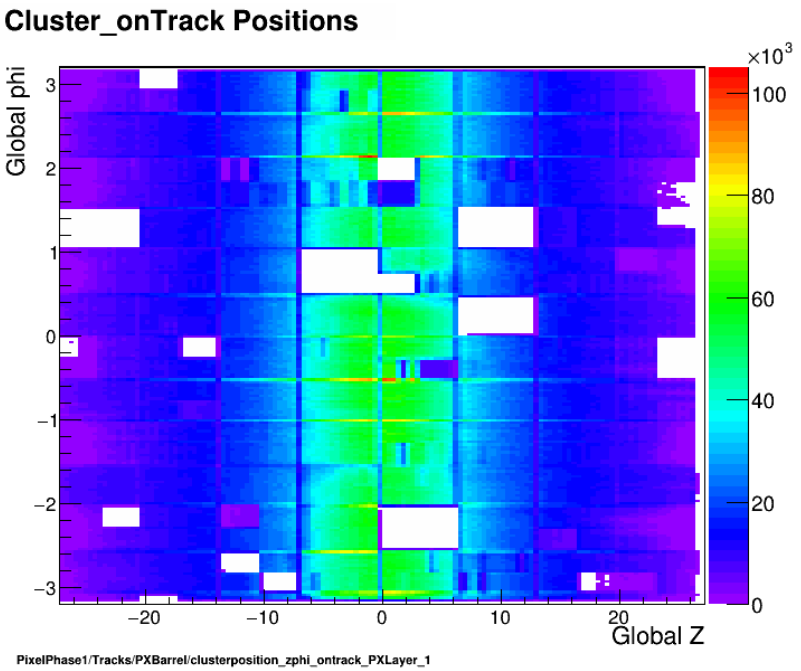


- In general, histograms representing a quantity used in DC are not strongly dependent on PU and/or luminosity, while occupancy obviously are
- **Occupancies in 1D are not suitable for the AE approach** (the variance is too large), however occupancies can be represented in 2D as a function of global coordinates as an example ...

Two main kind of histograms (II)

- Even if 1D representations of Occupancies are currently used in DC, it is much more informative to use the 2D ones
  - The right plot is much more informative than the 1D representation
  - In DC is important to locate dead regions!

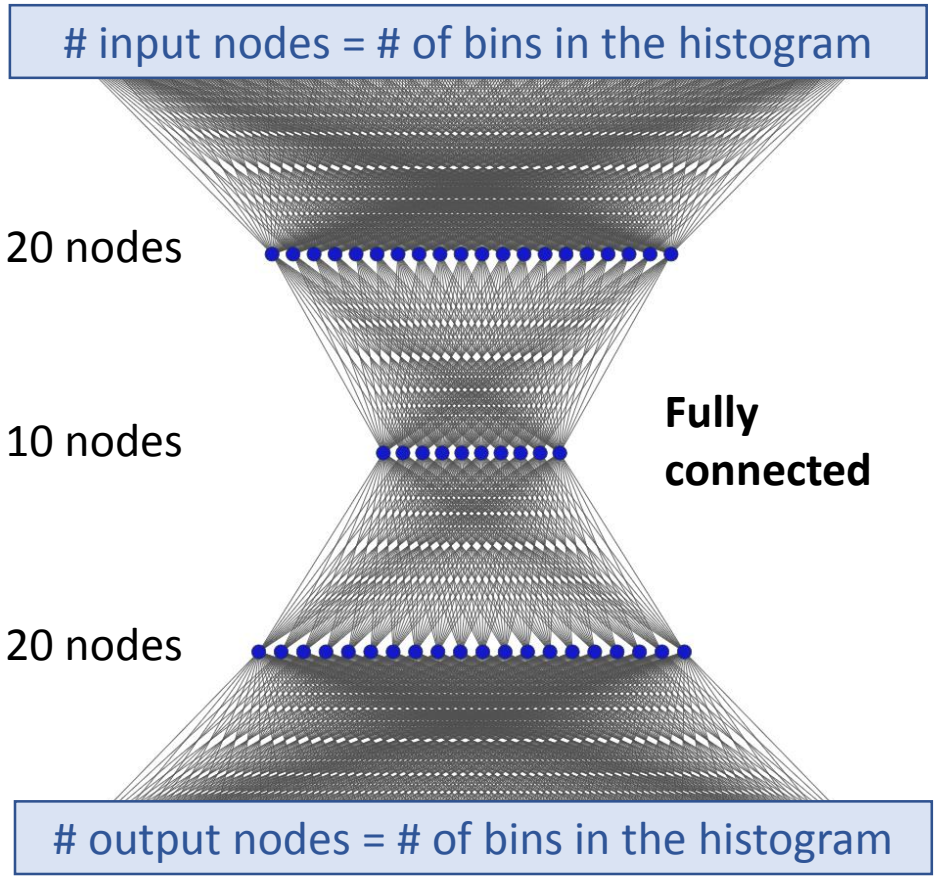




To manage occupancies we plan to use 2D plots, some work ongoing using convolutional AE, **we need your help here!**

The Autoencoder approach

- Train an Autoencoder on GOOD data (Semi-Supervised), use the reconstruction error (MSE) to identify "anomalies"
- A model made of **3 hidden layers (20-10-20)** seems to be suitable
  - The very same architecture suitable for all histograms
  - Deep enough to learn small variations (i.e PU, Luminosity)
  - Further optimization possible ...
- **Sigmoid** as activation function, loss **MSEtop10**, input histograms **normalized** (no further *standardization*)

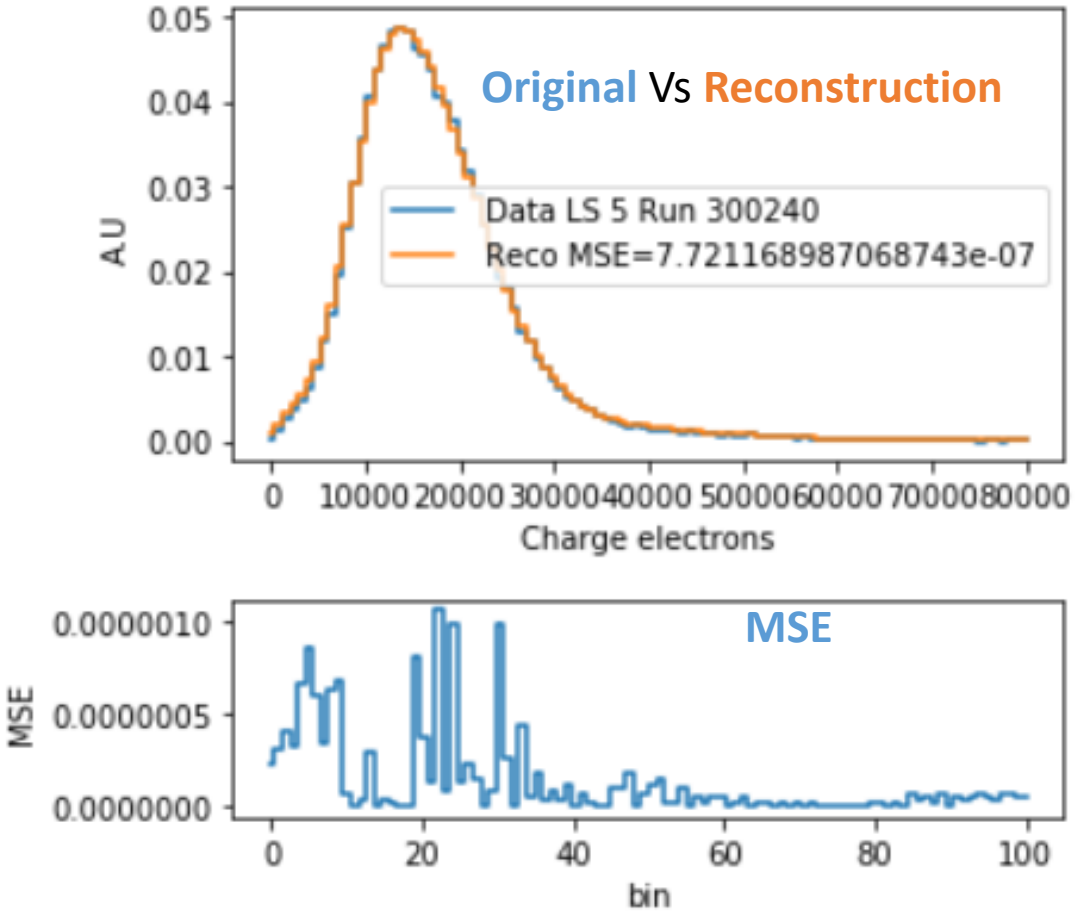


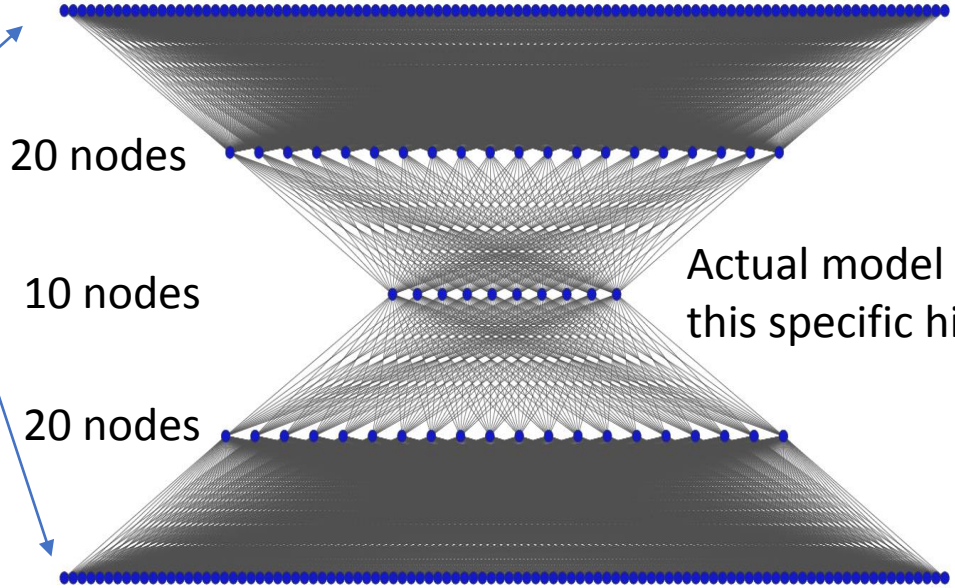
$$\text{MSE} = 1/\text{N} \sum_{i=1}^{N} (y_i^{in} - y_i^{out})^2$$

Need your help to design a convenient way to optimize (hyper) parameters!

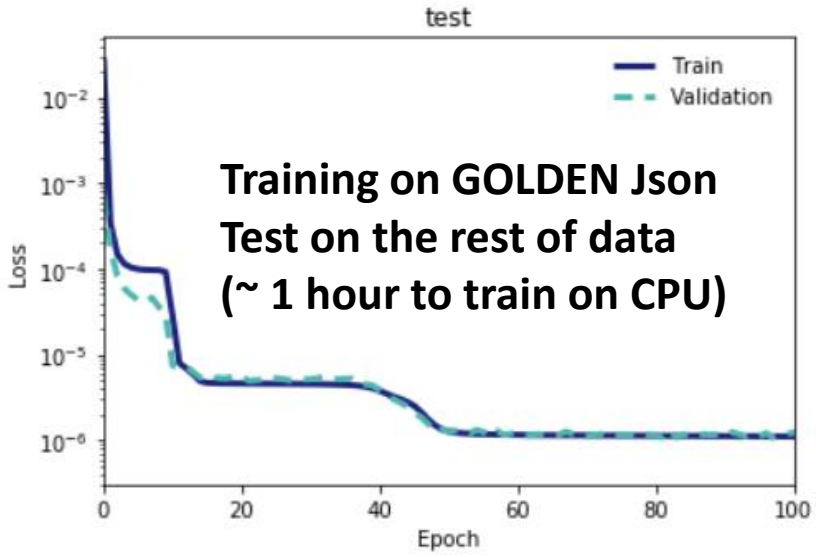
Examples: Pixel Layer1 Inner charge

- 100 bins in the histogram -> 100 input/output fully connected nodes

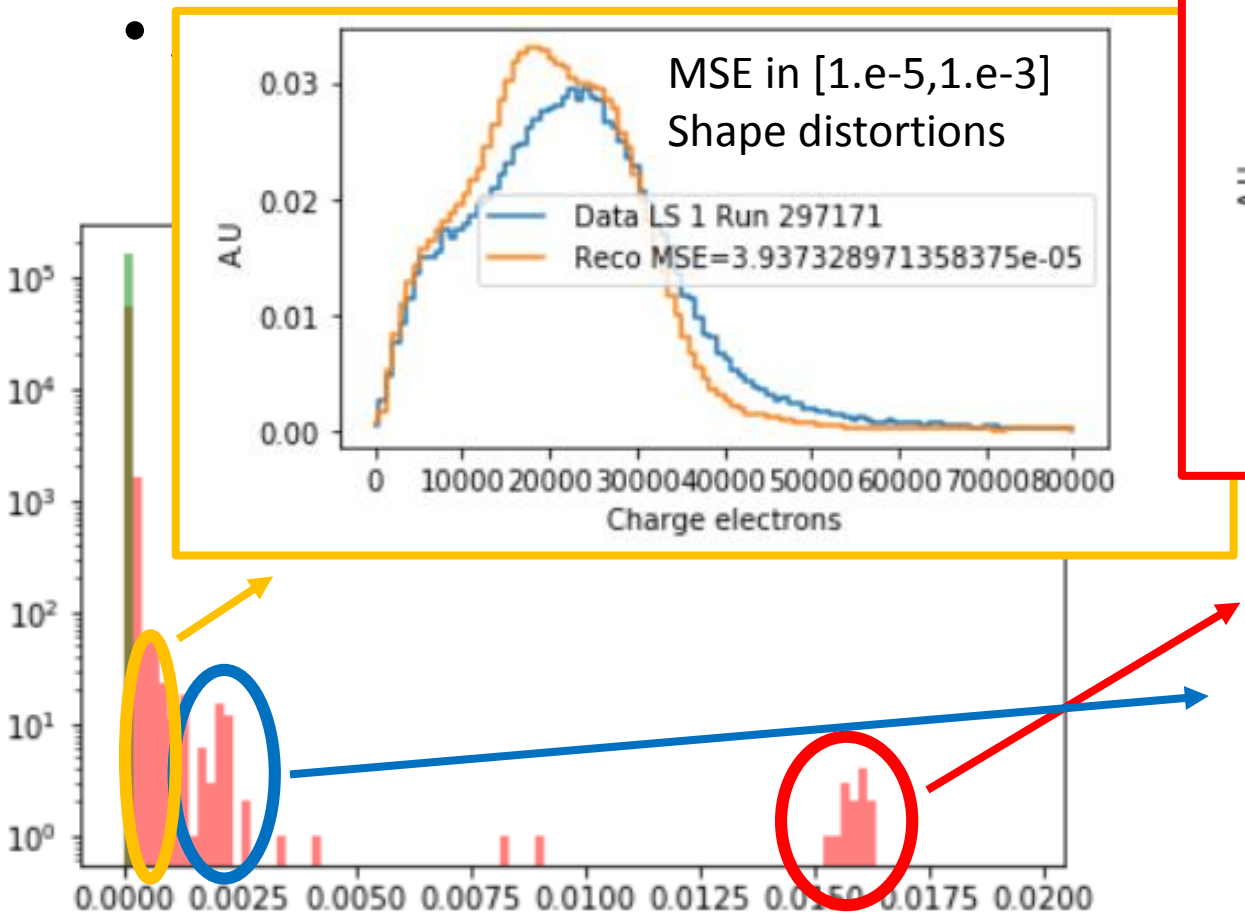


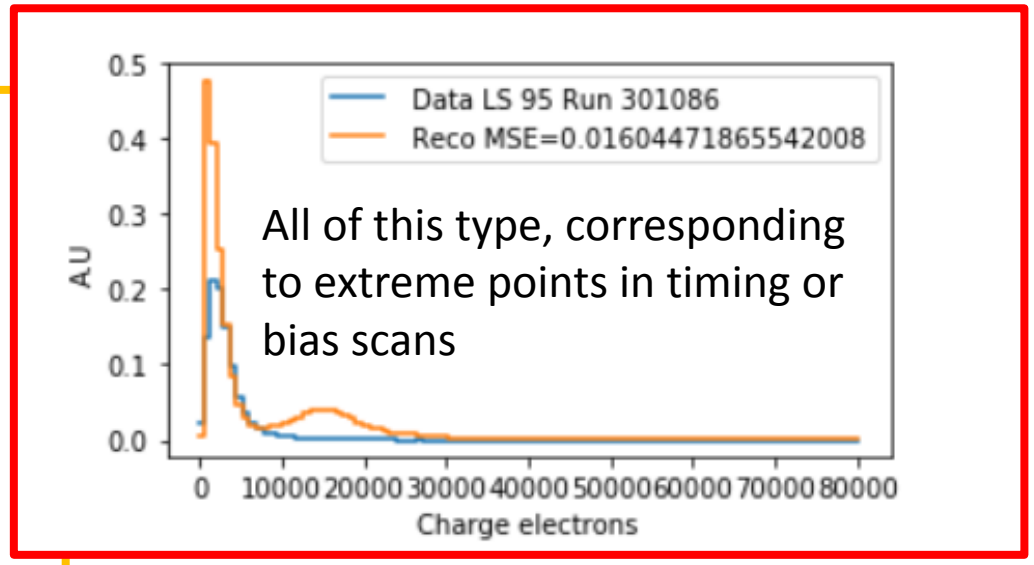


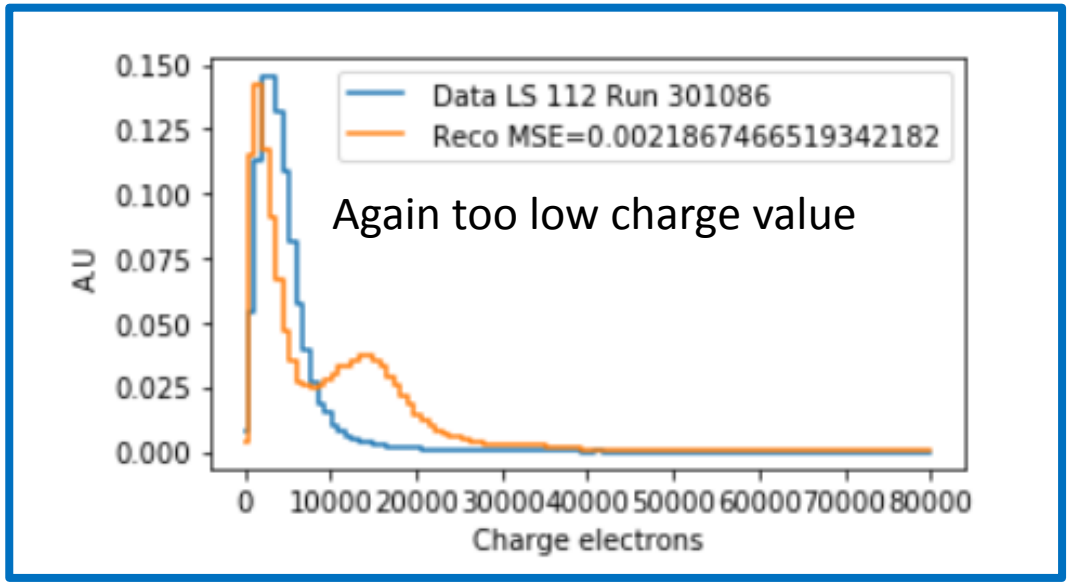
Actual model used for this specific histogram



MSE and Anomalies

- 



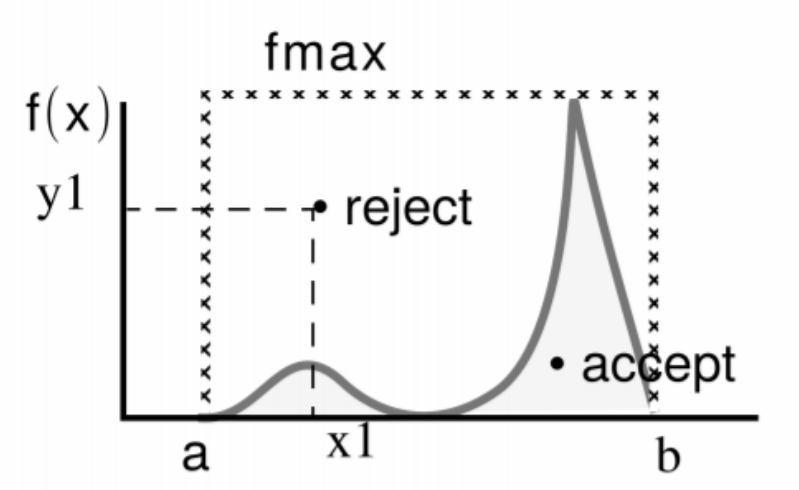


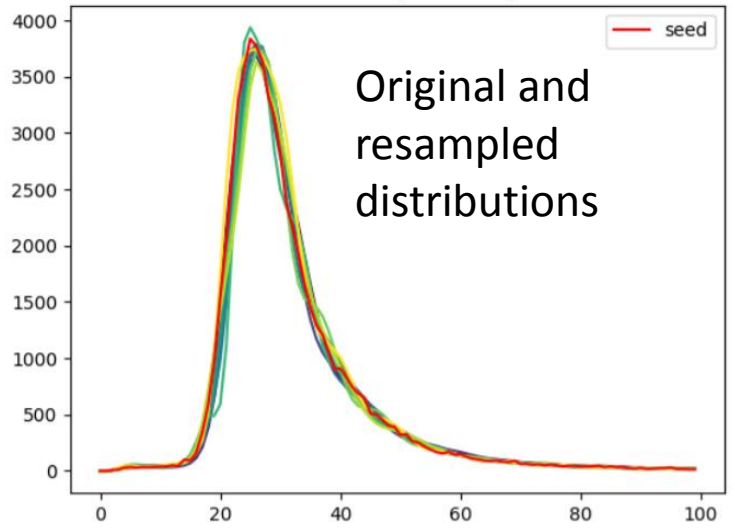
Semi-Supervised Autoencoder seems to be effective in spotting anomalies!

How to quantify performances?

L. Lambrecht, M. Niedziela

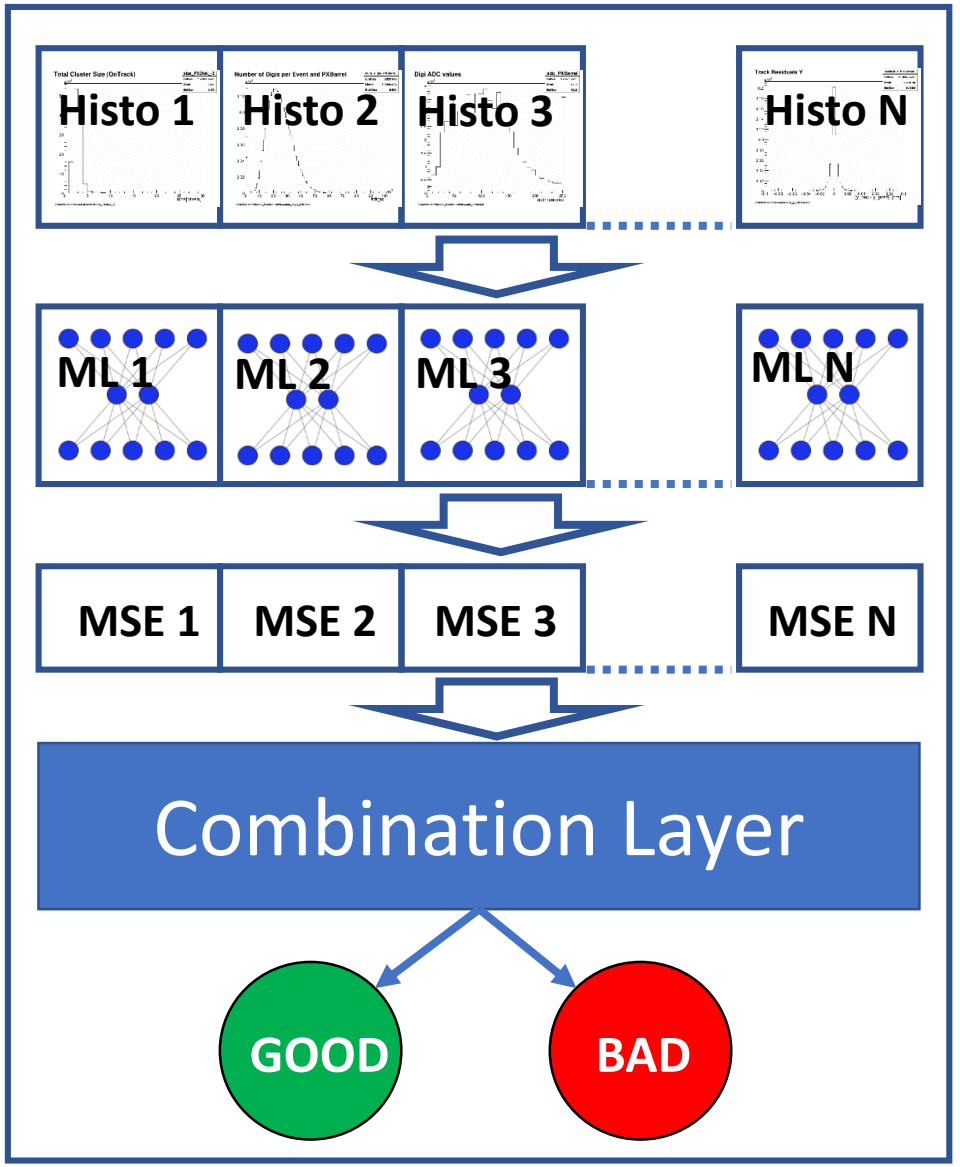
- Tricky to assess performance without **true labels**
  - BAD histograms are too few to quantify performance (and should be selected manually)
- Strategy: generate a suitable amount of GOOD and BAD data for validation
- A tool is recently available for labelled data generation (talk here)
  - Can resample a given distribution (MC method)
  - Can compute random linear combinations of a given set of histograms
  - Can add noise on top of each bin
- **Are there other possibilities we could explore?**





Next step: histogram combination

- Single histograms flags have to be combined to give the final flag for the given subsytem
- **Linear way**: different possibilities
  - Define a MSE threshold for each histogram and combine with a simple AND
  - Compute a average MSE and set a threshold
  - Possible also to assign weights to specific histograms (**need of subsytems expertise**!)
- **ML way**: use of NN on the sample of MSEs
- **Do you have comments suggestion about combination of ML results?**



Documentation, code, datasets and meeting

- The ML4DQM-DC twiki is the **main source of information**
- Some example code to read data, perform an exploratory analysis (see backup) and run few standard ML models (AE, PCA, NMF) is available here https://github.com/cms-DQM/ML4DQM-DC_SharedTools
  - Code can be used on SWAN or using GPUs in the IBM Minsky Cluster
- Several datasets with **Per Lumisection saving** are available on disk
  - **Available PDs: UL2017**: ZeroBias ,**UL2018**: ZeroBias, JetMet, SingleMuon, EGamma
- **Talks and discussion about ML4DQM-DC topics are hosted in the DQM General meeting on Fridays 14h-16h** (https://indico.cern.ch/category/3904/)
  - DQM task twiki: https://twiki.cern.ch/twiki/bin/viewauth/CMS/PPDOpenTasksDQMDC
  - **Generous EPR reward!** Please have a look and contact us if interested!
- Please, do not hesitate to join our meetings, or contact us! **cms-ml4dqm**, **cms-ml4dc** e-groups plus **ml4dqm-dc.slack.com**

Summary and Conclusions

- We are on the way to develop ML tools to automate the DC process
- **The bottom-up approach of using ML on single histograms is promising**
  - Results are human-interpretable, ML models can be kept simple, adding/removing variables is easy
  - **Room for new ideas and contributors!**
- A small group of contributors is (enthusiastically) gaining **experience in ML and on the full DC process**
  - We need this kind of people for a future deployment in production
- Huge amount of data available, not manageable by the DQM alone, **looking forward having relevant Subsystems involved!**
- Looking forward for your feedback on this project!

Backup

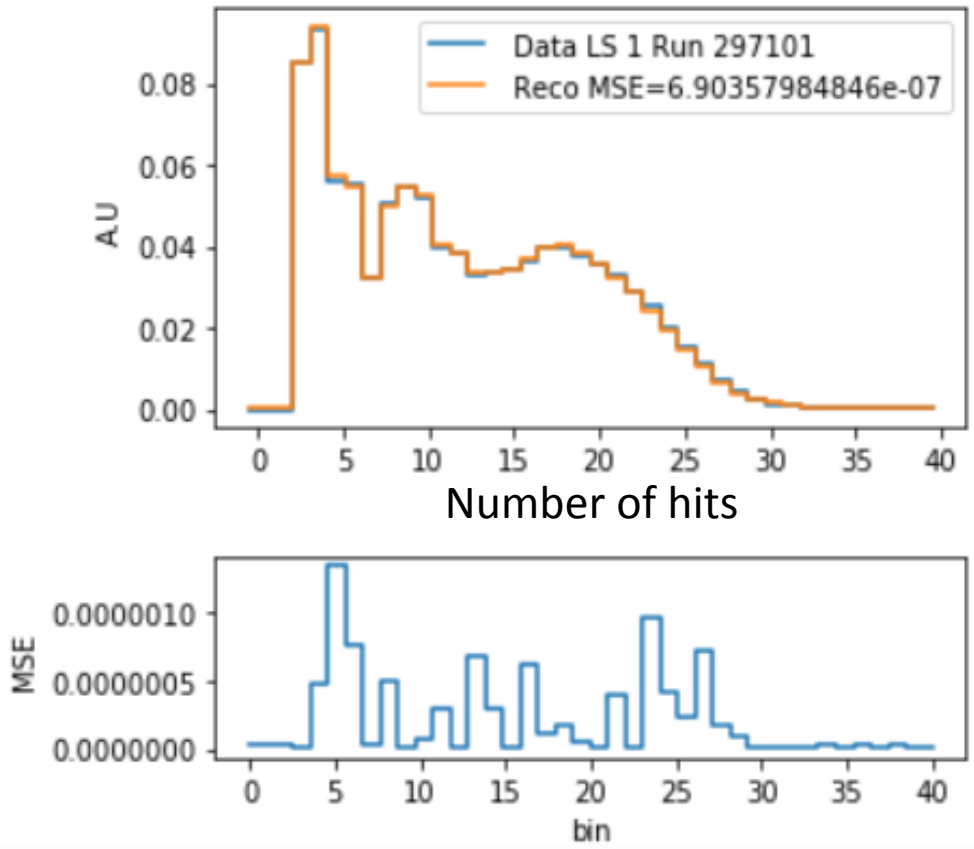
The DCS Bit

- **DCS = Detector Control System**, the DCS bit tells us if a detector is fully operational in the given LS. There is one DCS bit for each **DAQ partition** (which in general does not correspond to a full CMS sub-detector!), again a simple AND logic is used to give the collective value for the full CMS detector
- We have 23 DAQ partitions associated to DCS bits:
  - bpix, fpix, tibtid, tecm, tecp, tob, ebm, ebp, eem, eep, esm, esp, hbhea, hbheb, hbhec, hf, ho, dtm, dtp, dt0, cscm, cscp, rpc (if any is “0” the LS is BAD)
  - **NOTE: this step is already automated, but not based on any actual control of the quality of data**

Examples: Application to a different histogram

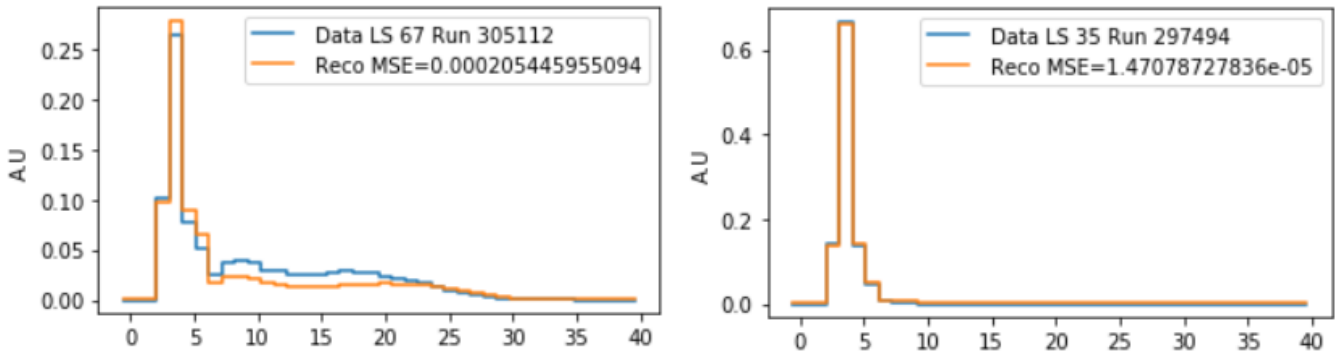
- Very same architecture of hidden layer, in/out layer reflecting the different binning (40 in this case)

Example of Good Histogram

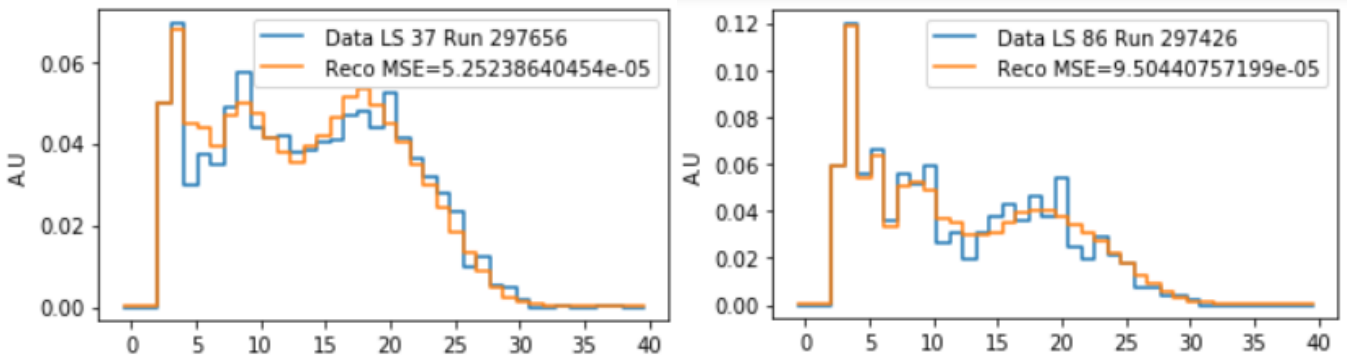


Number of hits

Anomalies



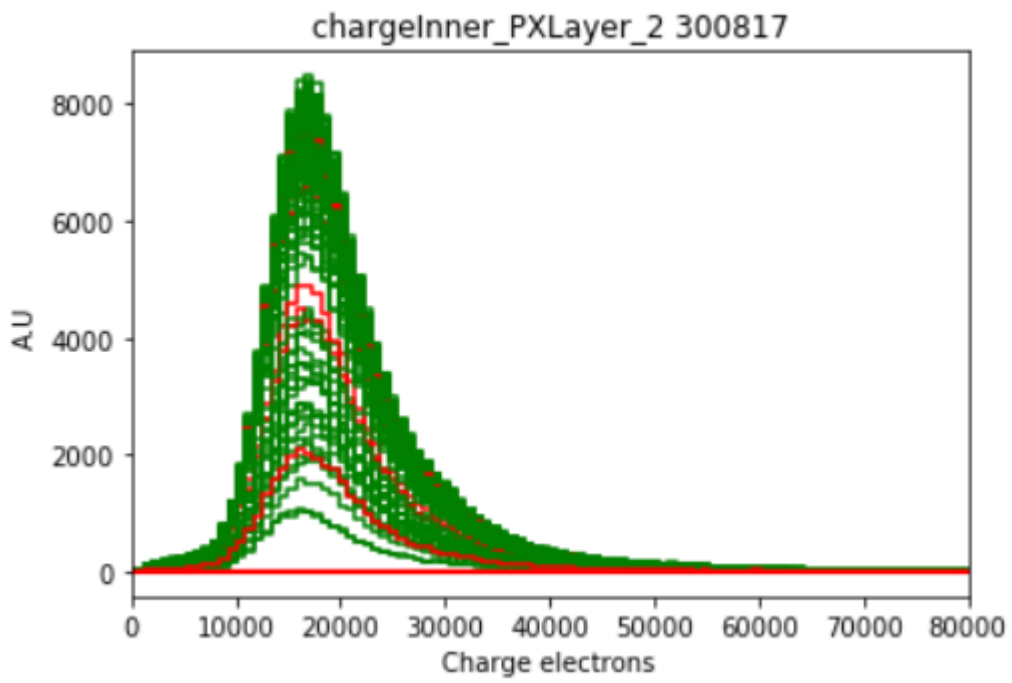
Very few or no Strip hits

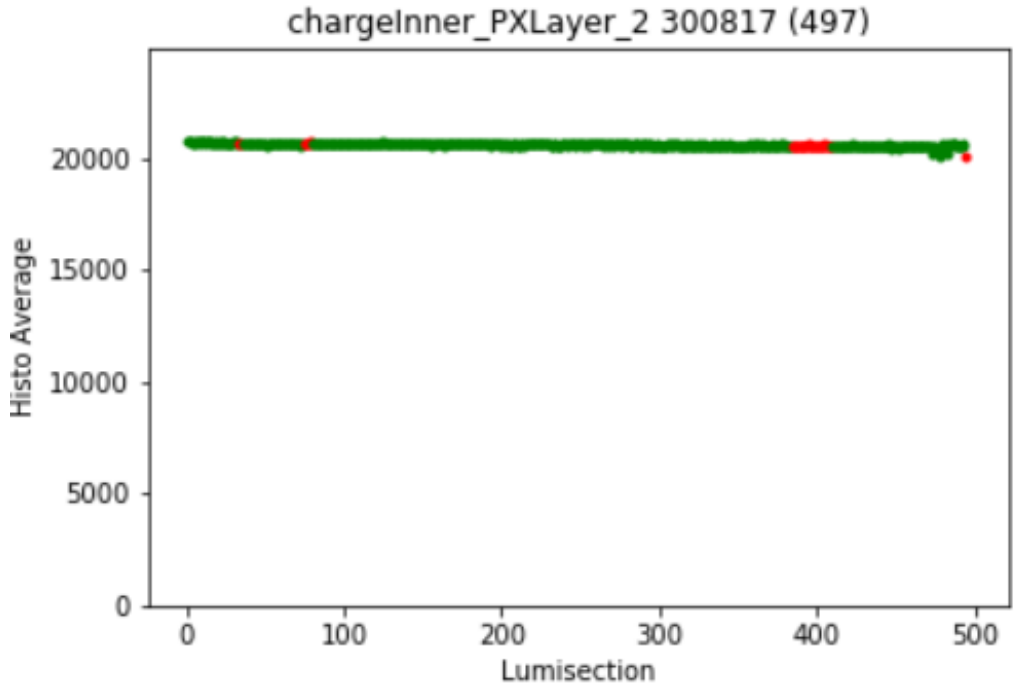


Anomalous shape

Common code for exploratory analysis

- JSON plots: curve (or points) in red belongs to data not included in the Golden JSON, while green plots (or points) represent Golden data



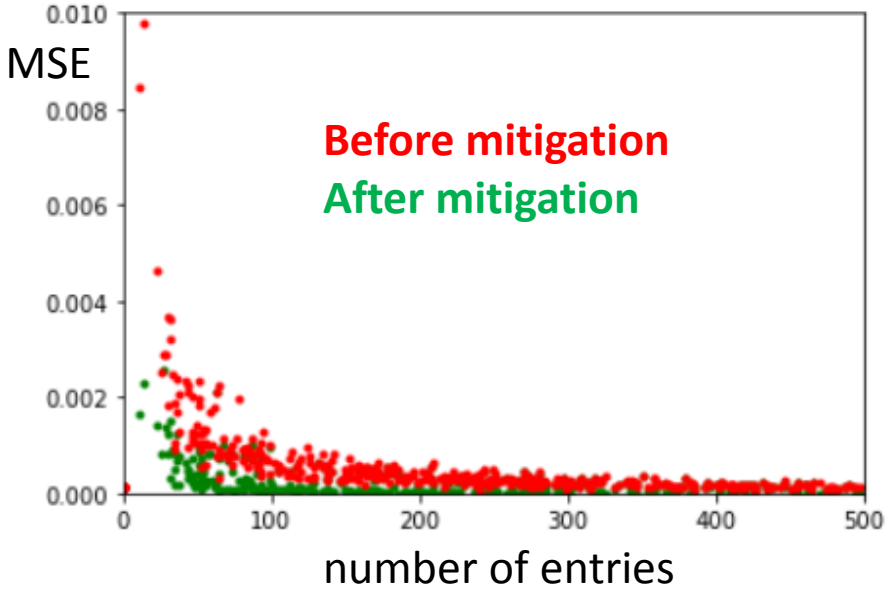


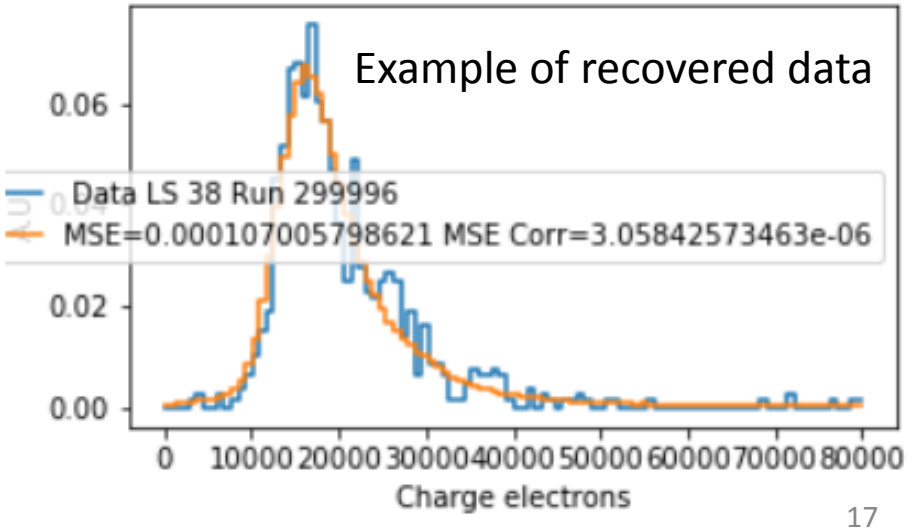
Powerful plots to have a quick analysis of input data, the code to produce such plots is available in the notebook below:

https://github.com/cms-DQM/ML4DQM-DC_SharedTools/blob/master/ML_Model_Examples/Standard_AE_Step-by-Step.ipynb

How to treat low statistics histograms

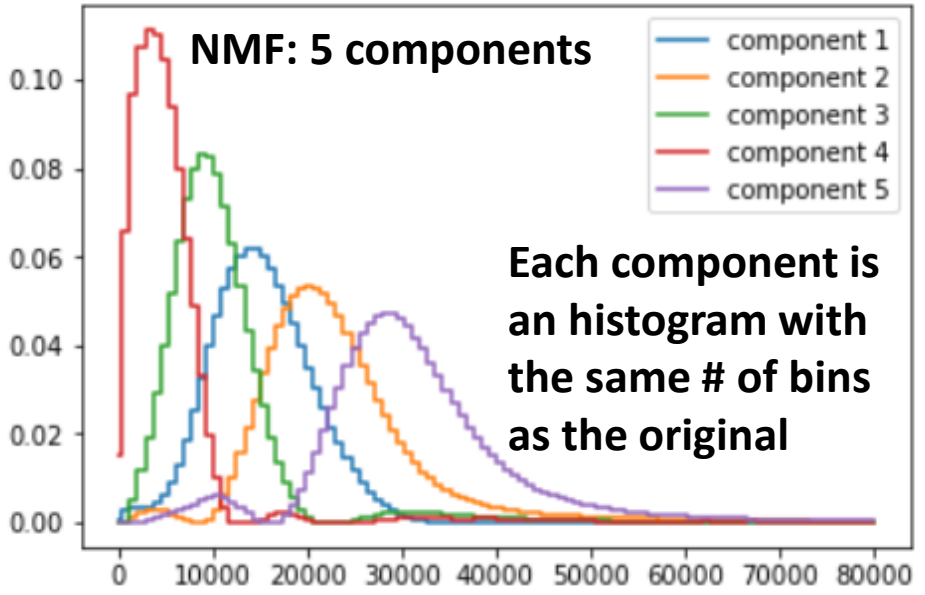
- If the statistics is really low, there is no point to attempt any recovery
- Some cases are more borderline (see bottom plot), and should be recovered
- Strategy: exclude from the MSE bins in which the reconstruction error is inside the statistical error
- As a consequence the MSE is, on average, reduced for low stat histograms

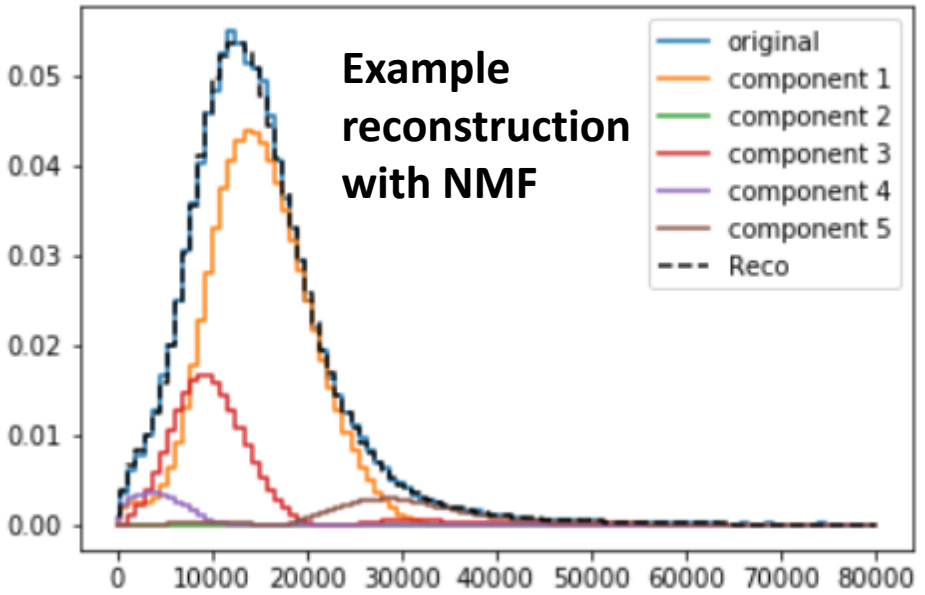




Different open possibilities

- Use of Unsupervised models for **dimensionality reduction**
- Commonly used methods (see backup):
  - Principal Component Analysis (PCA)
  - Non Negative Matrix Factorization (NMF)
- Represent data in a "base" of components:
  - **regular data** are well reconstructed using only the **first few components**
  - **anomalous data** show **higher order terms**
- Can be used in combination with the Autoencoder (improve robustness)
- **Starting a serious study …**





ML models under study: NMF factorization

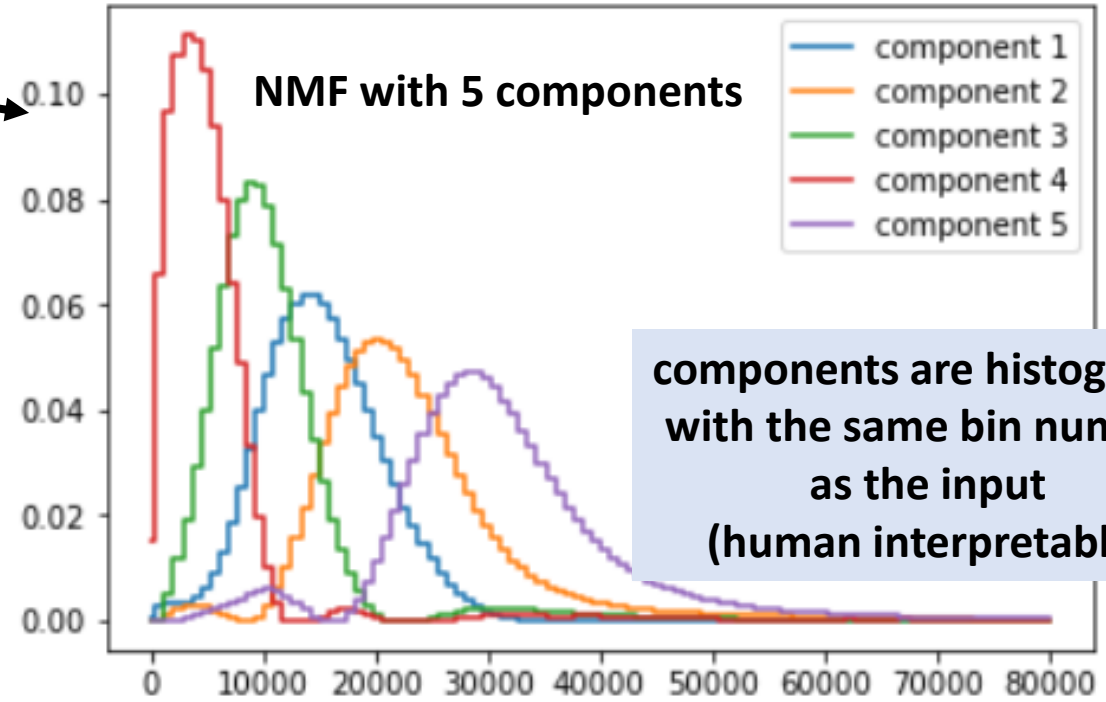
- **N**on-negative **M**atrix **F**actorization (NMF), unsupervised clustering
  - Deal only with non negative values (i.e histogram frequencies), fast
- Given the matrix **X** of input data, the model compute an approximate decomposition in **W·H**
  - each row (i.e histograms) can be approximated by a linear combination of a predefined set of components

Input histogram

components

$$x_i = [w_{i1}\ w_{i2}\ \ldots\ w_{ik}] \times \begin{bmatrix} h_1 \\ h_2 \\ \ldots \\ h_k \end{bmatrix} = \sum_{j=1}^{k} w_{ij} \times h_i$$

w_i: weights



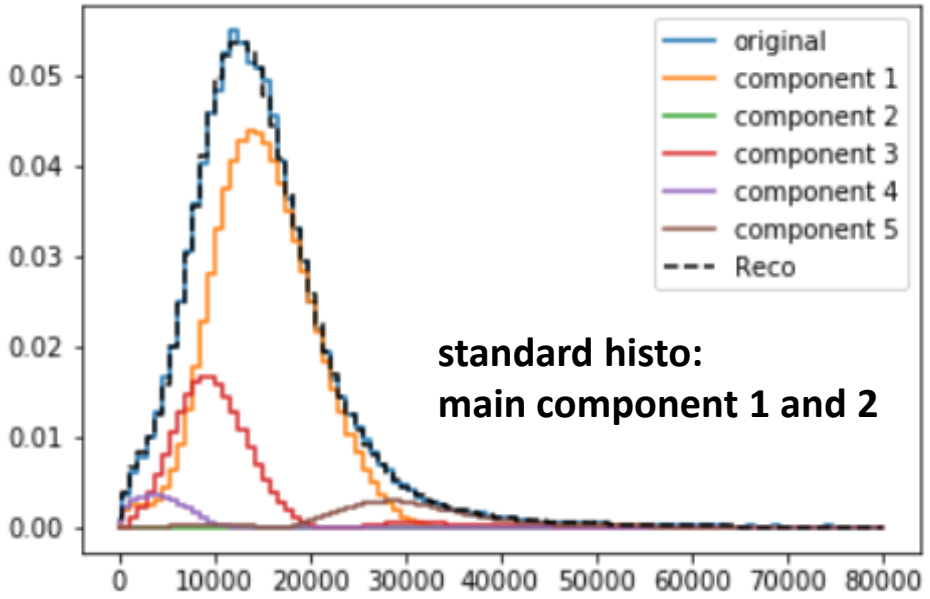
components are histograms with the same bin number as the input (human interpretable)

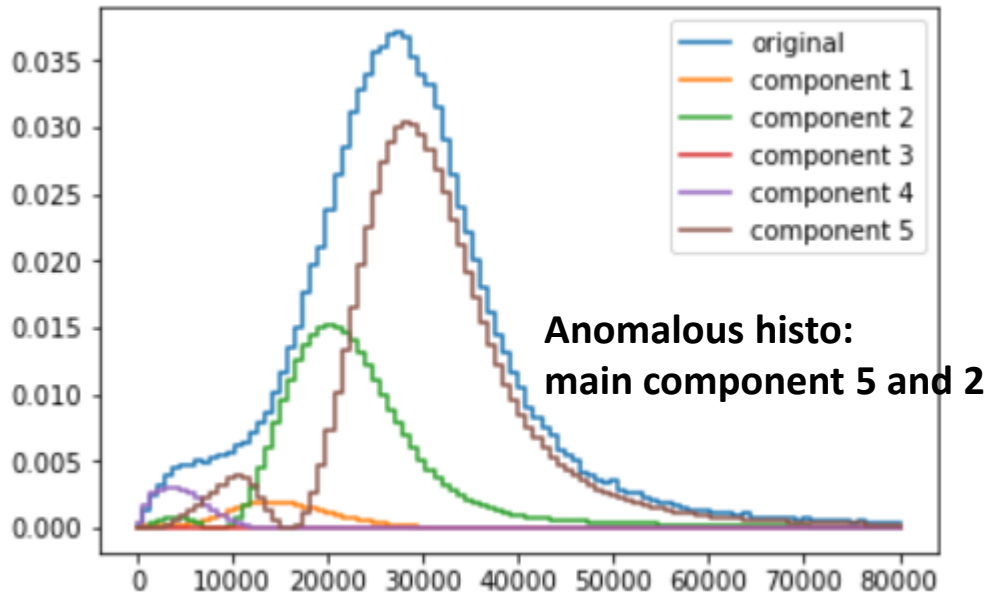
In principle, could distinguish standard and anomalous data by the values of weights for the different components

Can also be used for 2D plots (image decomposition)!

ML models under study: NMF factorization (II)

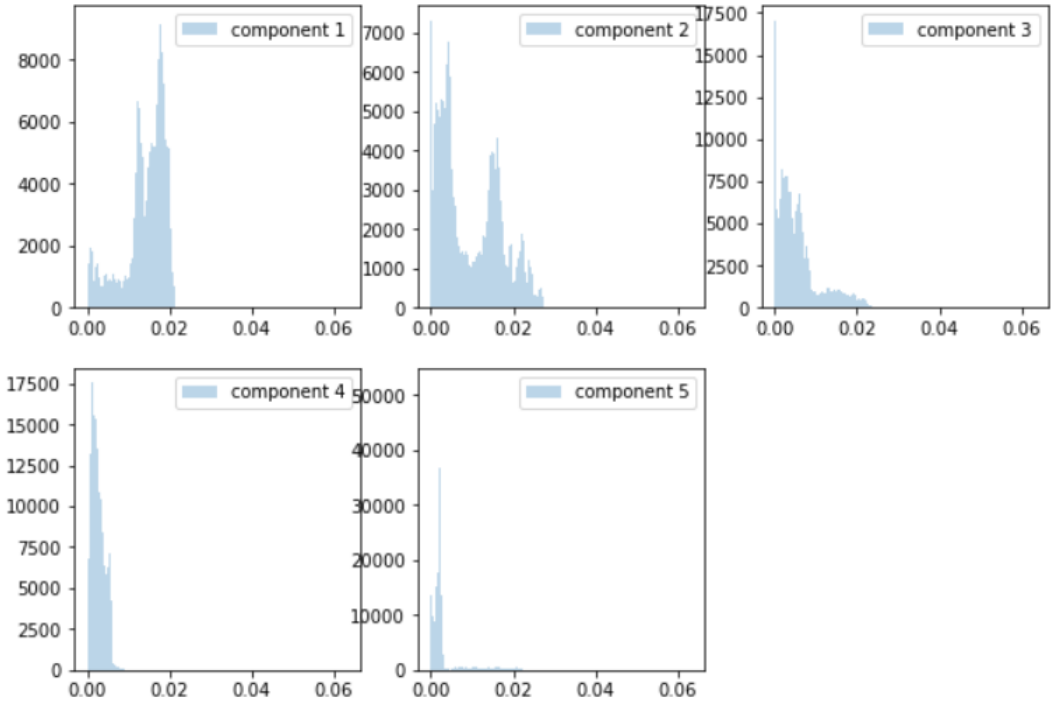
- Preliminary study on the same Pixel Layer1 charge histogram and 2017 data, using 5 components
  - Input matrix (100x200k), ~ 1 minute to obtain coefficients on Swan
  - Again, quite a lot of optimization work is needed, however it looks promising ...

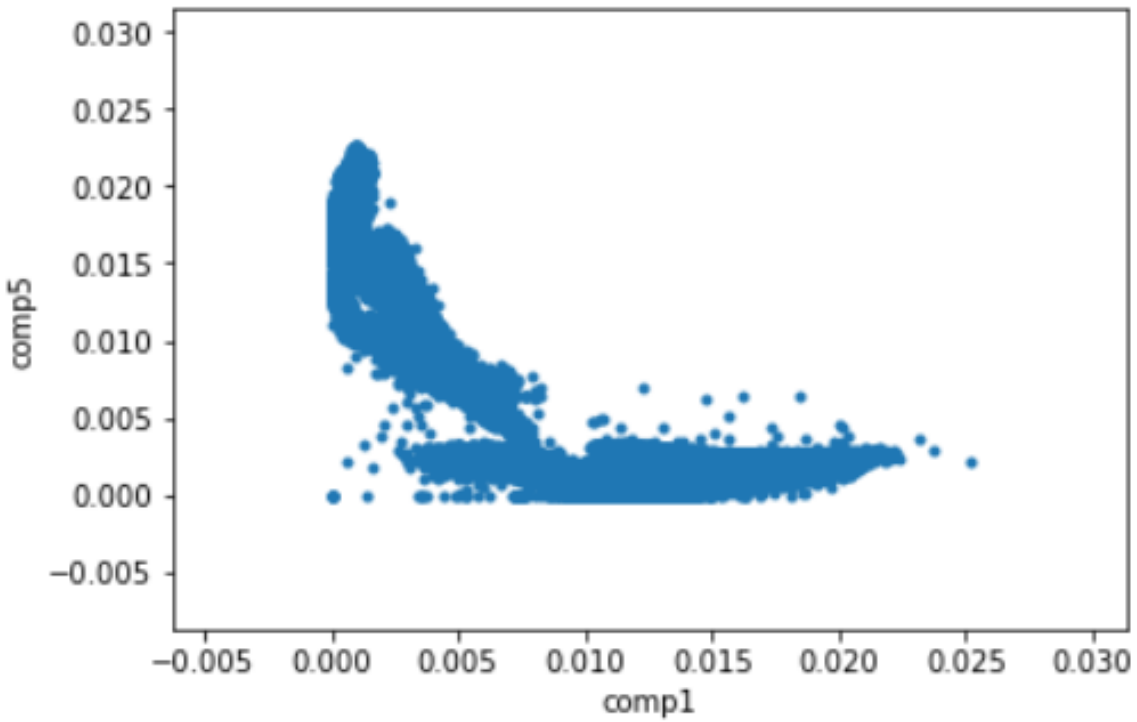




ML models under study: NMF factorization (III)

- Classify data based on the contribution of the different components, and undertand which are the cases in which the anomaly traduces to a BAD flag

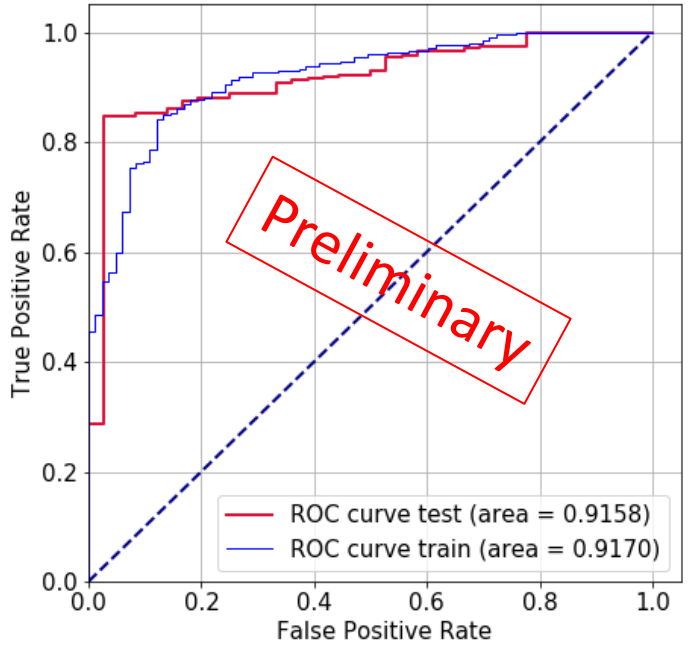


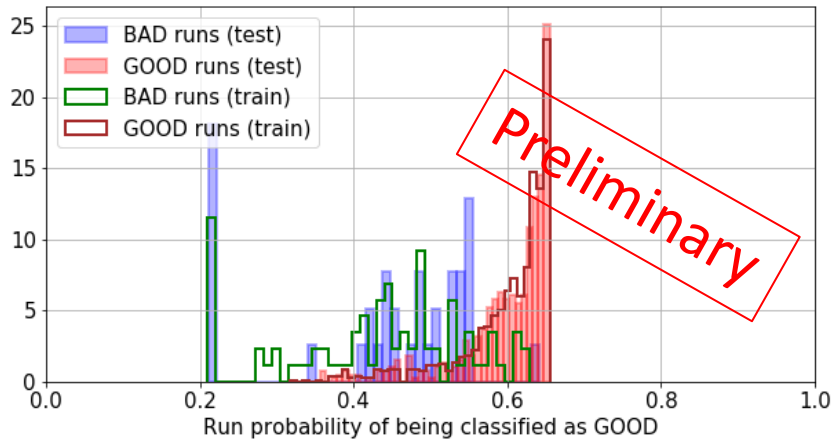


Automate DC per single Run

J. Fernandez, A. Trapote

- There are (rare) cases in which a full Run is BAD due to outstanding issues
  - i.e part of the detector not turned on properly
  - No need for a detailed LS analysis
- Plan to use ML to filter out these trivial cases before applying the DC per single LS
- Make use of "standard" DQM GUI files with single run granularity
  - Histograms moments available in csv files (details here)
- Preliminary studies ongoing using Random Forest and Muon data
  - Possible to use Cosmic runs also





Comparison with “standard” test

- Often requested for a comparison with Kolmogorov-Smirnov test
  - Old good algorithms appear to be simple and robust, why to use ML?
- Implementing a system with KS test is as much work as using ML
  - KS test need a “reference”, to be defined for 100s of plots (and then maintained)
  - The reference is a snap-shot in time, ML can incorporate better expected (small) variations (ML can learn the History)
- KS doesn’t give an uncontroversial GOOD/BAD classification, but an indicator of the “goodness” of the comparison (just as the MSE for ML)
  - need a significant amount of work to be tuned (i.e no gain w.r.t ML)
- In conclusion, setting up an automatic DC system based on “standard” statistical tools is as hard (if not harder) than use ML