Particle flow: refactoring and ML

Joosep Pata

October 7, 2019

PF ML discussion

Goals

- **Physics:** Improve offline particle flow for interpretability, extensibility, code readability in *RecoParticleFlow/PFProducer*
  - Make it possible to include additional information in the reconstruction
  - Understand what the algorithm is doing and if it degrades, why?
  - Match or exceed the existing performance
  - Unify with HGCAL reconstruction
- **Computing:** improve timing of PF, enable PF algorithm to run on accelerators
- **My interest in this**: refactoring pledge for PF group, understand and possibly improve basic CMS offline reco

Problem statement

- Given the set PFBlockElements in the event, create a set of PFCandidates
- On the event level, set(N) to set(M) translation, but can be factorized
  - **Block algo:** find which elements MUST be considered together to produce consistent and physically meaningful candidates
    - clustering with supervision by existing PFAlgo
  - **Candidate algo:** given a small set of connected elements, produce PF candidates
    - {TRK, ECAL} → pion,
    - {TRK, ECAL, HCAL} → K
- Ultimately interested in elements → genParticles, but start with PFCandidates, PFAlgo should be kept for a while in any case

What we have

- From CMSSW AOD, can dump std::vector<reco::PFBlock>, std::vector<reco::PFCandidate>, std::vector<reco::GenParticle>

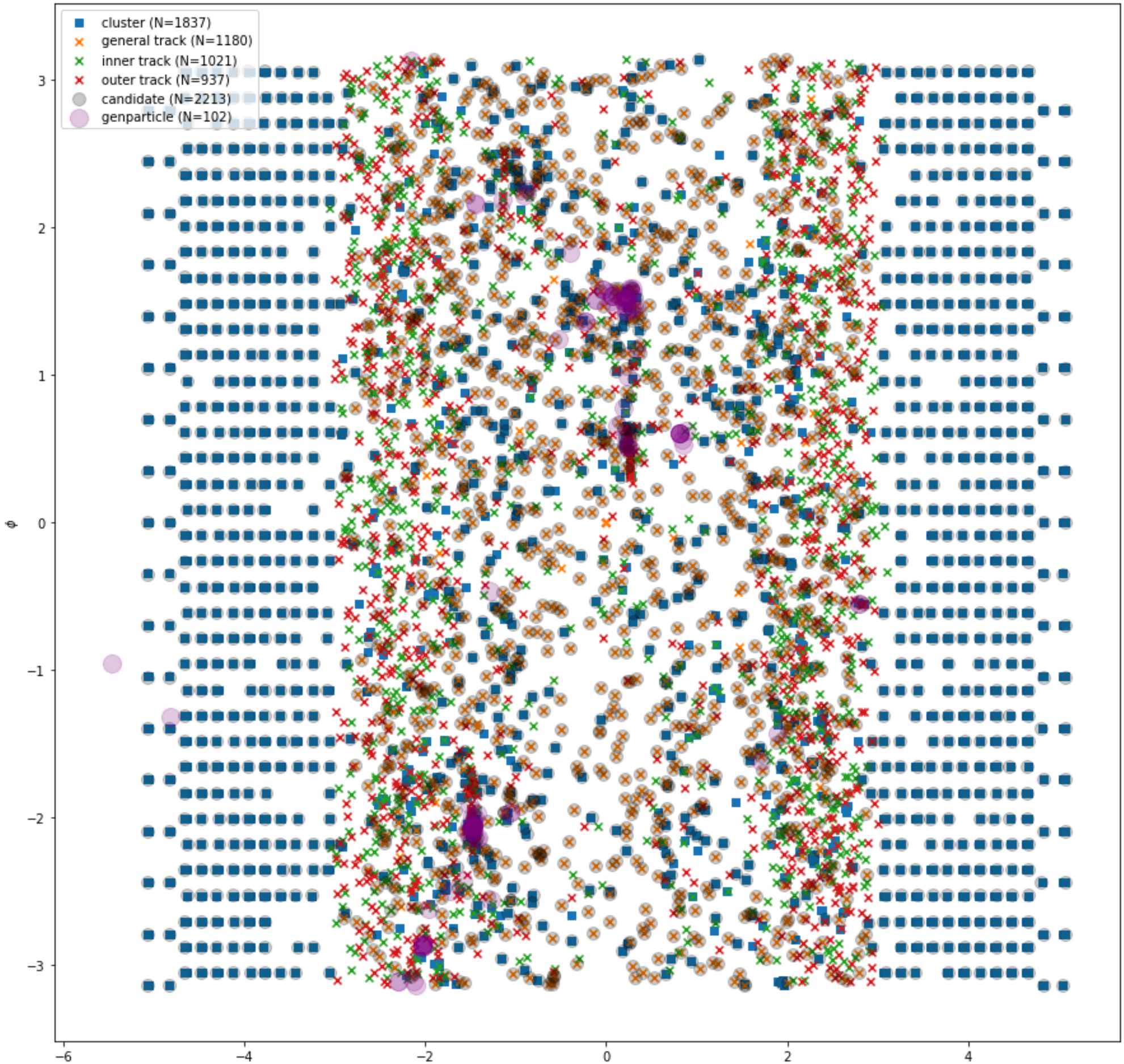
- PFBlock consists of elements of different types

  - in standard CMSSW, most of the event is in just two blocks: eta+ and eta-

- For each PFCandidate, we know the elements that were used to produce this candidate: edges of the graph with elements, candidates as nodes

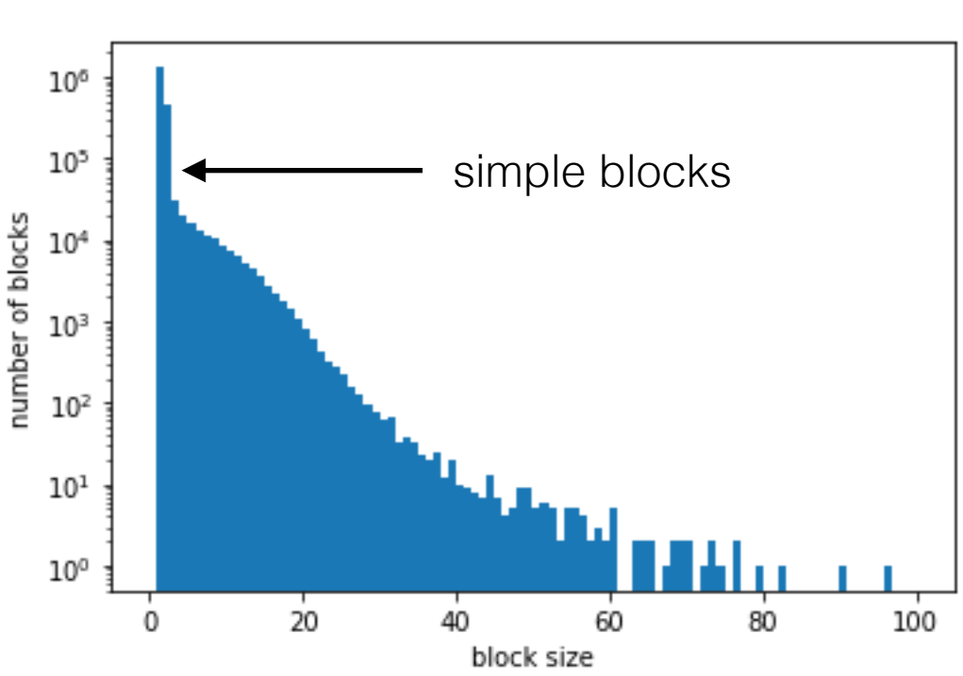
- Multiple elements can be associated to the same candidate, and multiple candidates to the same element!

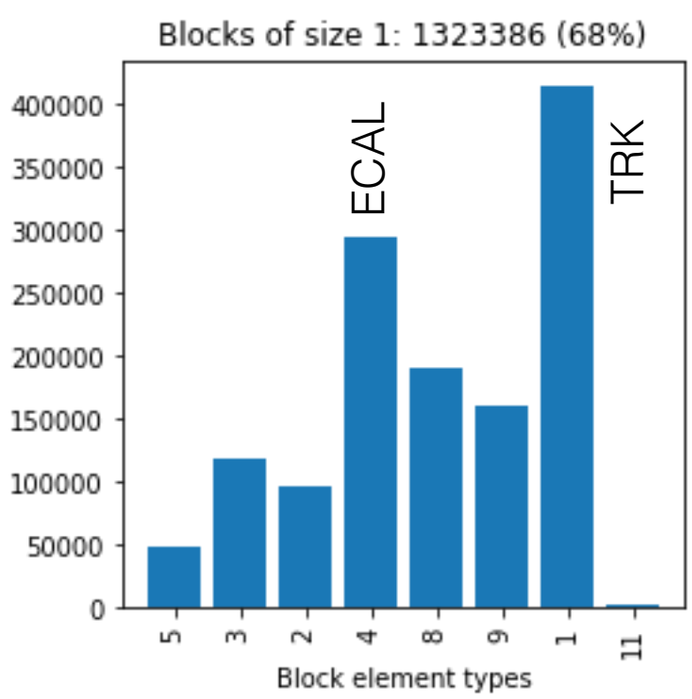
- We can look for elements that are disjoint subgraphs (miniblocks) as induced by PFAlgo based on *PFCandidate::elementsInBlock*

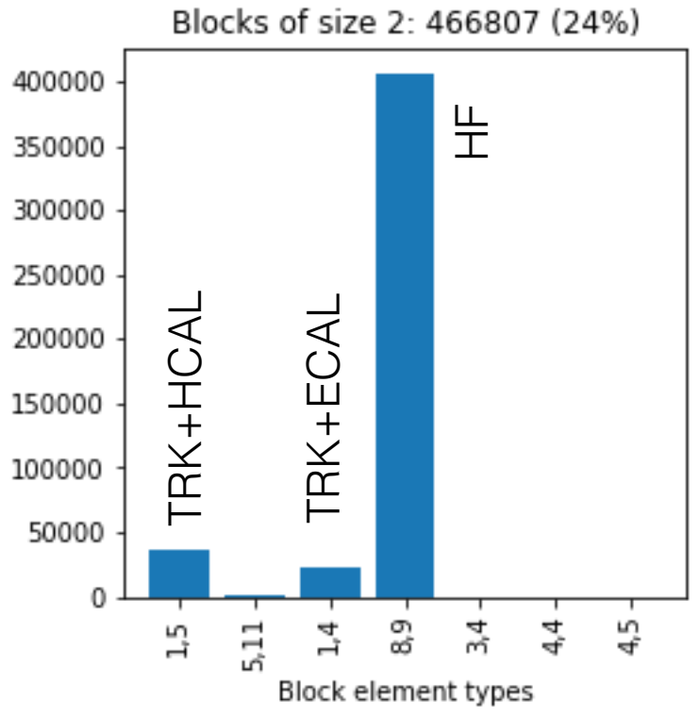

cluster (N=1837)
general track (N=1180)
inner track (N=1021)
outer track (N=937)
candidate (N=2213)
genparticle (N=102)
3
2
1
0
−1
−2
−3
−6
−4
−2
0
2
4

Miniblocks from PFAlgo

- 92% of the subblocks consist of 1 or two elements (per PFAlgo)
- Event could be significantly cleaned up by looking for tracks, ECAL & HCAL elements that cannot reasonably linked to 0 or 1 other elements







1 - TRK, 4 - ECAL, 5 - HCAL

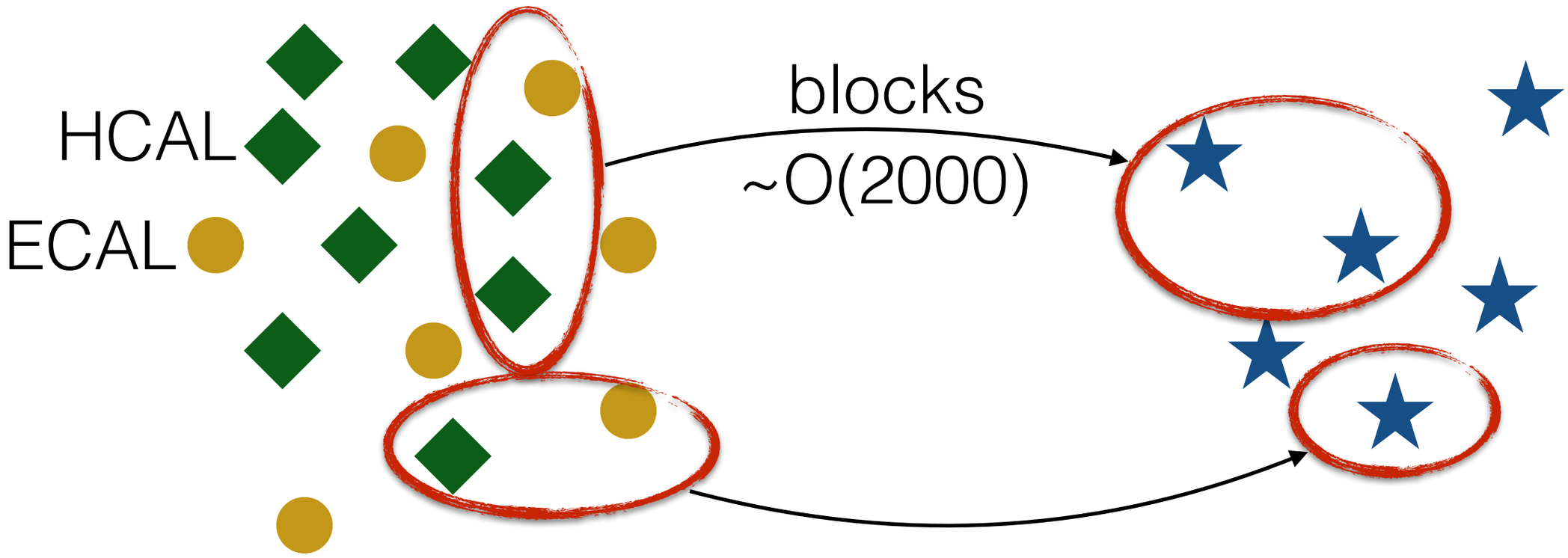
Dataset

input per event:

set of elements $\mathbf{e}$={●,◆,...}

output per event:

set of candidates $\mathbf{c}$={★,...}



~O(5000) ~O(2500)

Inside a miniblock, there is no further granularity between candidates and elements. Typical case: one HCAL cluster shared by several tracks, produce several candidates.

Dataset, numerically

- *elements = (N_elem, N_feat_elem)*
  - list of all input elements (energy, eta, phi, track coordinates); ~5k elements per event
- *element_block_id = (N_elem, 1)*
  - unique block ID for each element based on PFAlgo; ~2k different blocks
- *candidates = (Ncand, Nfeat_cand)*
  - list of all candidates (pdgid, pt, eta, phi); 2.5k candidates per event
- *candidate_block_id = (N_cand, 1)*
  - unique block ID for each candidate based on PFAlgo as above

Baseline algo

- Define inputs, outputs, "loss function", compare to ML solution

- ***particleflow(elements)*** → ***candidates***

  - ***cluster(elements)*** → ***blocks***: create blocks using clustering, e.g. DBSCAN, 2d grid scan

    - Needs to estimate elem-to-elem distance matrix (KD-tree)

    - Clustering loss can be monitored using labelled clustering metrics e.g. *adjusted_mutual_info_score* in sklearn

  - ***reconstruct(block)*** → ***candidates****:* translate each block into a small number of candidates

    - true candidates to reco candidates loss by e.g. MSE

ML ideas

- Elements to blocks
  - clustering with optional supervision by PFAlgo
  - Create initial neighbor map using approximate distance matrix
    - Turn edges on or off in a graph based on neighbours
- Block to candidates
  - simple regression on a few elements to a few candidates
  - Can try independently of above based on PFAlgo-induced miniblocks

Further ML ideas

- pix2pix: translate a multi-channel image of elements to an image of candidates/genparticles with a GAN
  - Convolutions can be helpful, as reconstruction should generally be local
- Sets: Learn to encode/decode permutation invariant functions of the inputs/outputs

Code and samples

- /RelValTTbar_13/CMSSW_11_0_0_pre6-PU25ns_110X_upgrade2018_realistic_v3-v1/GEN-SIM-DIGI-RAW samples processed, 9k events
  - EDM: /mnt/hadoop/store/user/jpata/RelValTTbar_13/pfvalidation/191004_163947/0000/step3_AOD*.root
  - ROOT: /storage/user/jpata/particleflow/data/TTbar/191007_162300/step3_AOD_*.root
  - npz: /storage/user/jpata/particleflow/data/TTbar/191007_162300/step3_AOD_*.npz
- https://github.com/jpata/particleflow
  - EDM → flat ROOT ntuplization
  - flat ROOT → numpy ntuplization, miniblock finding via subgraphs
  - Example notebooks, discussion