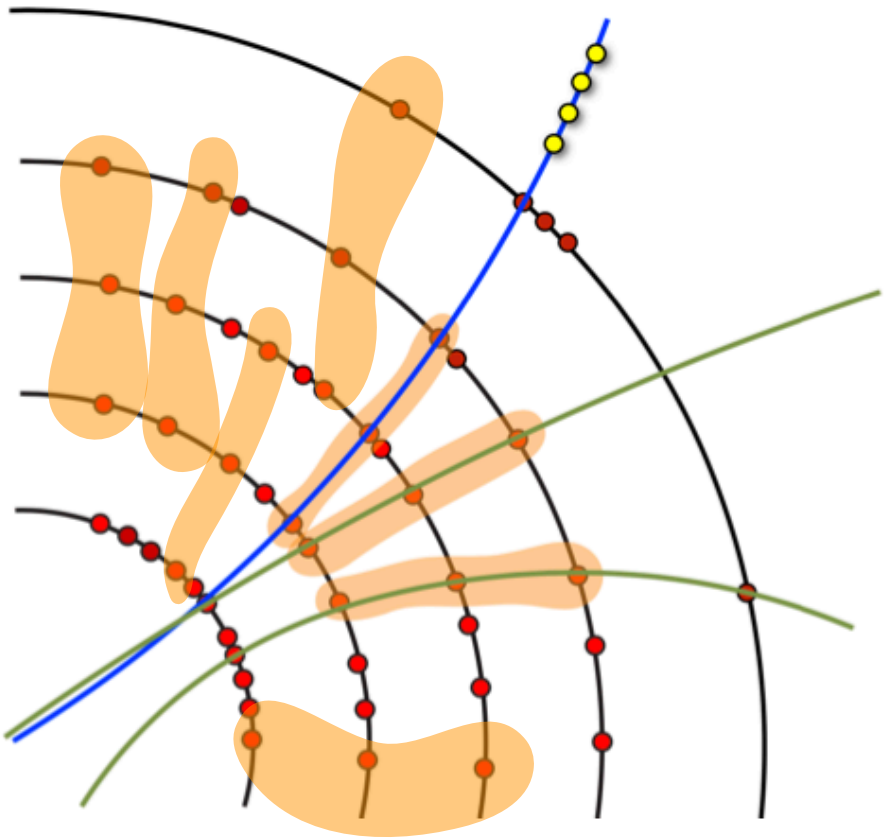
# Machine learning based seed filtering in ACTS

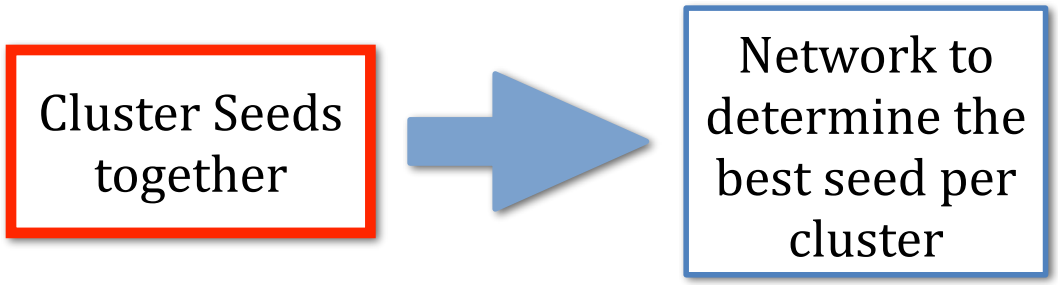
Corentin Allaire

# Seed filtering

- ~100k Seed produced by the seeding but ~800 tracks reconstructed at the end
- A lot of seed duplication (and useless seed)
- Each seed is then processed by the CKF => takes a lot of time to execute
- The Ambiguity solver can then remove duplicate / Fake track produced, but the time has already been lost
- Can we evaluate the quality of the seeds before the tracks have been reconstructed?
- Can we use this to speed up the tracking chain?



# Machine learning based Seed filtering

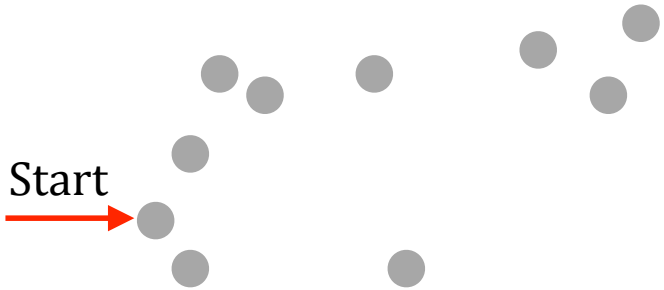



Same approach as the ML based Ambiguity Solving :

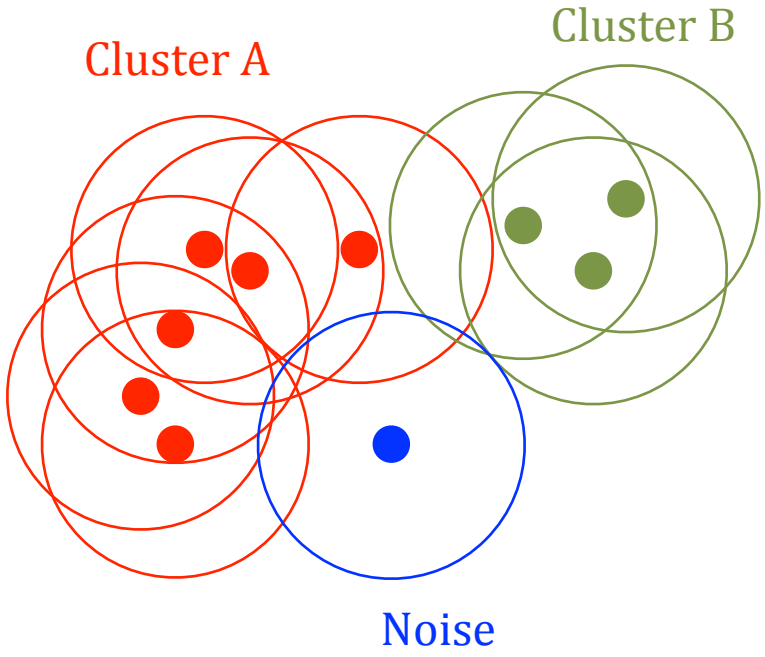
- **Clustering**: cluster seeds together, ideally 1 cluster ~ 1 truth particle
  - Using **DBScan** (a classic clustering algorithm) ➡ mlpack C++ implementation of many ML algorithms, really old (16+ years) still maintained to this day!
  - Cluster in a 4D space using: Phi, Eta, Z0 and Pt
  - Those 4 parameters might need to be scaled depending on the detector here Pt = Pt/10
  - Clustering can be finetuned to affect efficiency/purity balance

# DBScan clustering

- Idea : 1 cluster = 1 truth particle
- Clustering based on data density
- Use 2 parameters :
  - ε: Max distance between neighbour
  - $Min_{sample}$: Min number of elements per cluster
- More than $Min_{sample}$ neighbour ➡ Create a cluster
- For each element of the cluster, do the same ➡ extend the cluster
- In the Seed Filter :
  - distance in (η,ϕ); ε=0.07 ; $Min_{sample}$=2




$Min_{sample}$=3



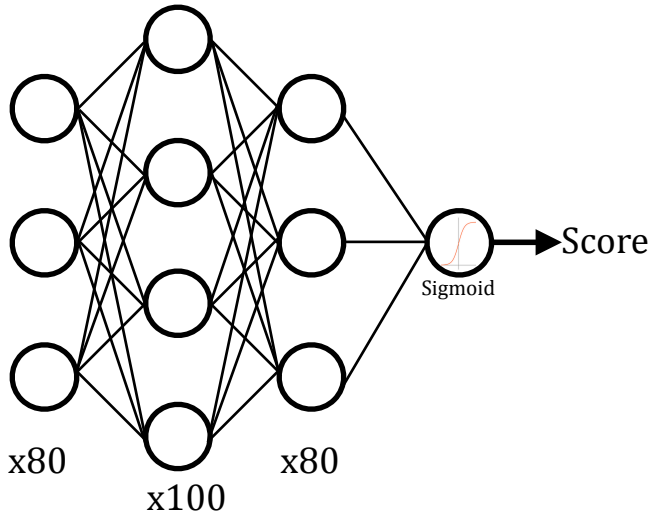
# Machine learning based Ambiguity Solving

Cluster Seeds together

Network to determine the best seed per cluster

**Neural Network**: Score each seed, keep the highest score per cluster :

- Four layers NN, used for ranking
- Use 14 seed variables as input
- **One score per seed ➔ Select the best one**




Input parameters :

- Pt
- Eta
- Phi
- Z0
- Seed Quality
- Space point 1 (x,y,z)
- Space point 2 (x,y,z)
- Space point 3 (x,y,z)

**Advantages** :

- No parameter tuning, only a short training
- Available in Acts vis **Onnxruntime**

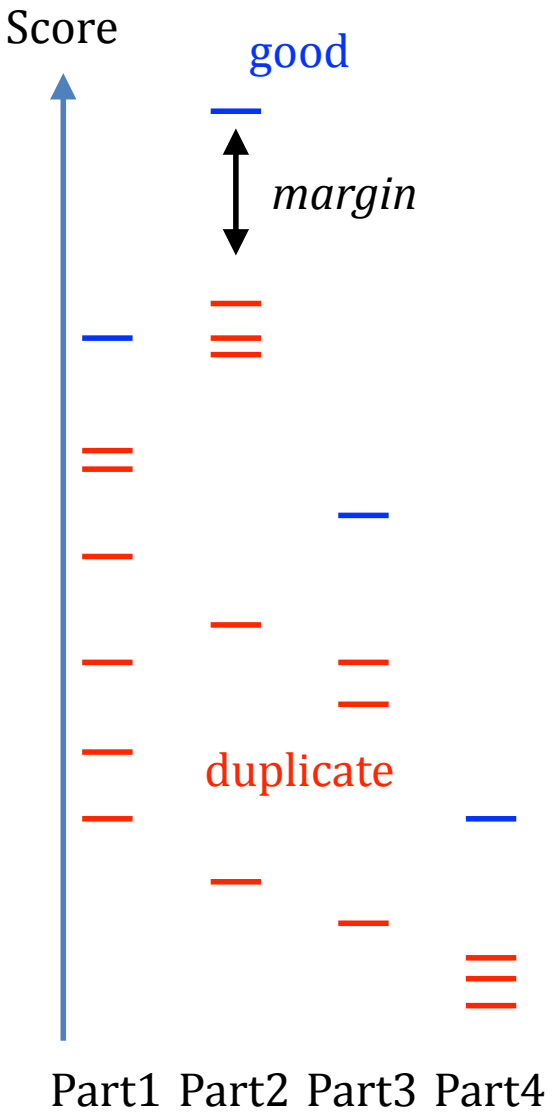
# Ranking Neural Network

**Goal:** one score per seed, highest for the **good** one

- Training without clustering ➔ use **truth** matching instead
- Compute one loss per **truth particle**
- Use a **Margin Ranking Loss :**

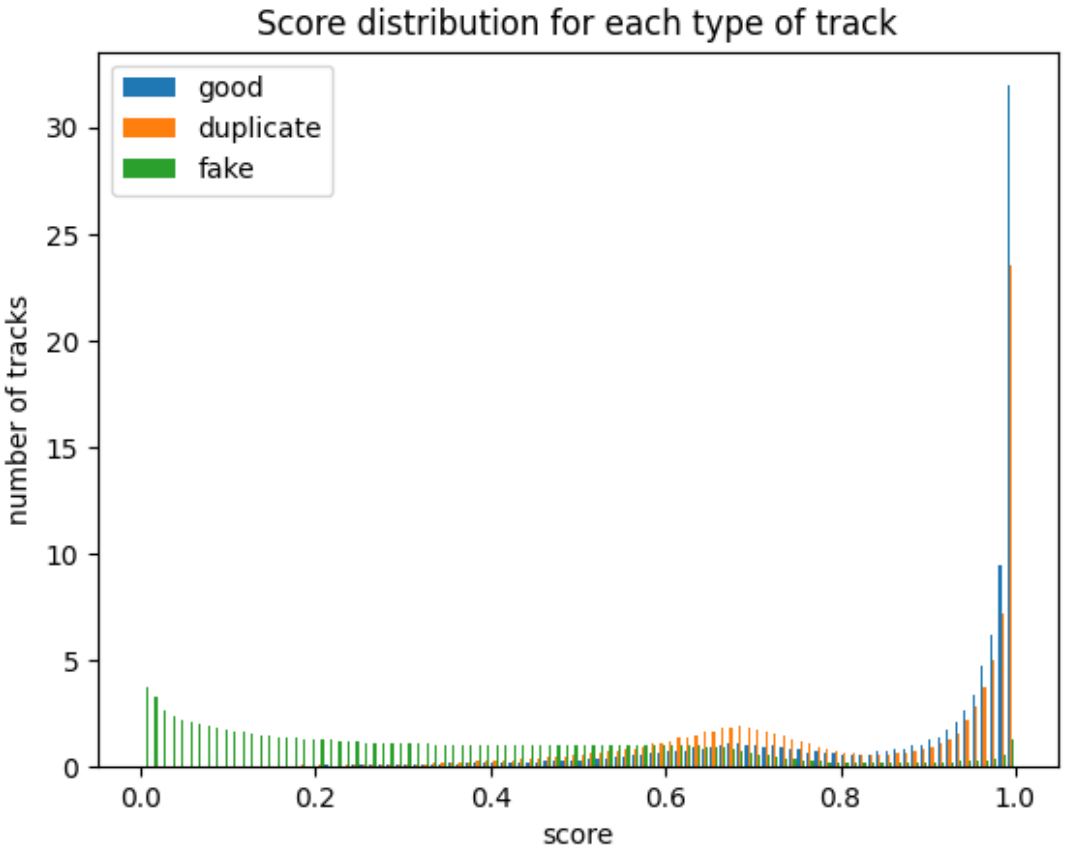
$$loss_{part} = \frac{1}{N_{tracks}} \sum^{tracks} max(0,\ x - y + margin)$$

- x: track score;  y: good track score; margin = 0.05
- If $score_{bad} < (score_{good} - margin)$, then loss = 0
  Else, loss = score difference minus the margin
- Try to **separate the good and bad scores** by at least a *margin*



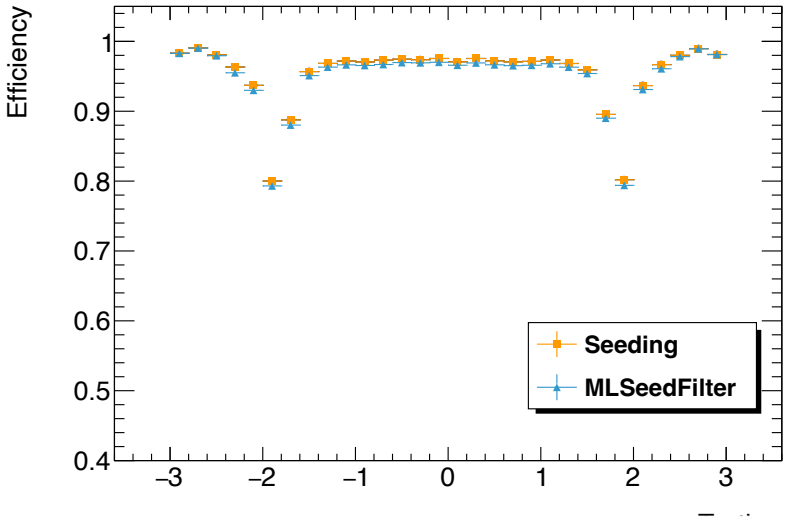
# Performances : Score for the seeds

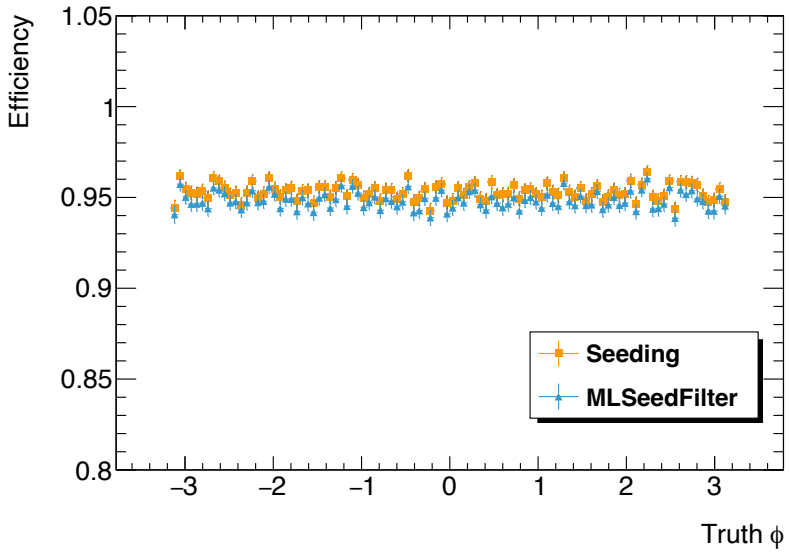
- Scores for good, duplicate and fake seed
- The network is capable of separating the 3
- Really good at finding fakes
- Uses 2 different margins for the duplicates and fakes (0.3 and 0.9)
- In the process of changing the definition of good :
  - Now: Closest to the truth
  - Futur: Seed that leads to the best track

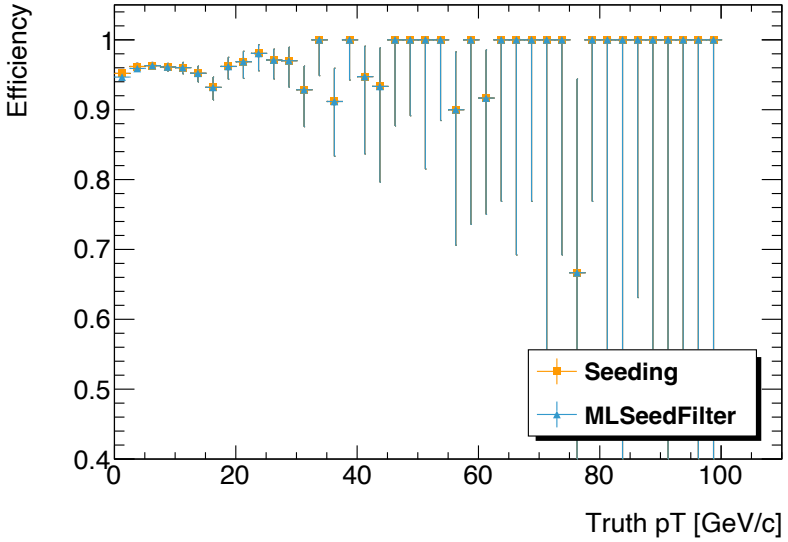


# Performances : Seed Efficiency

- Seeding efficiency before and after the ML seed filter
- Loss of ~0.5% efficiency
- Track finding speed-up (x1.5 faster) : 38.6 event/s ➔ 26.3 event/s
- Filtering ~ 0.1 event/s

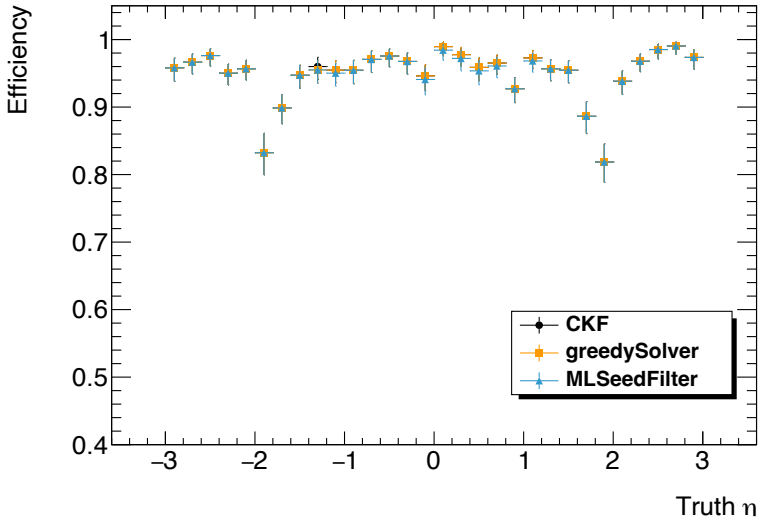


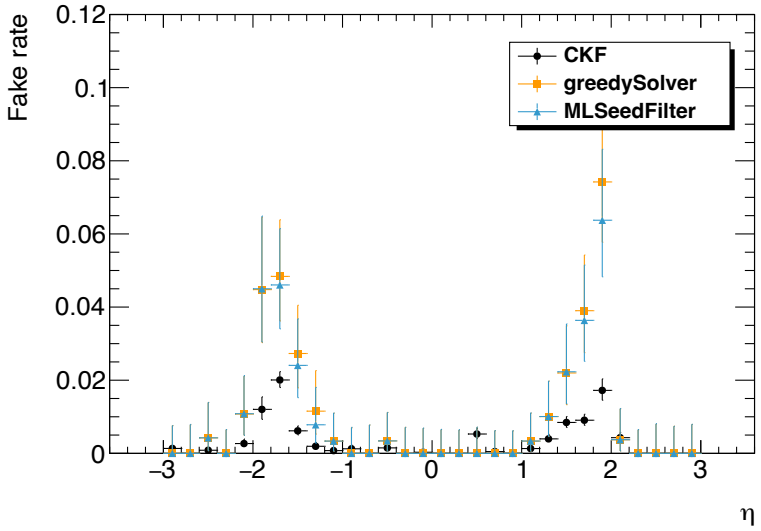



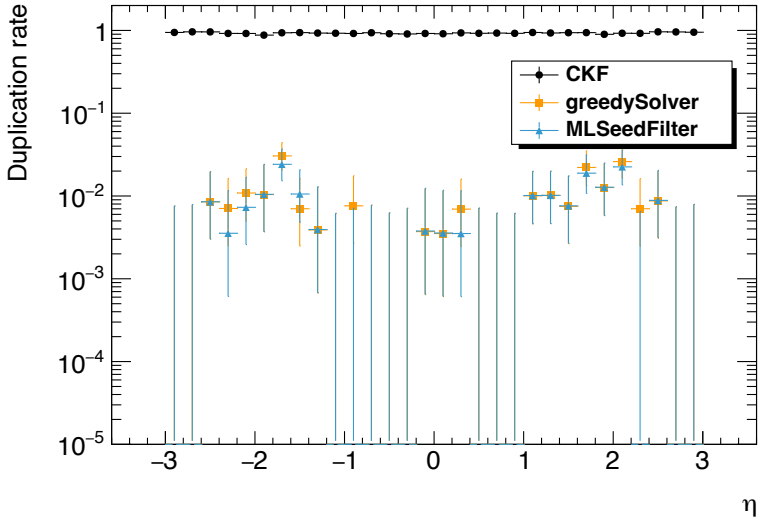



# Performances : Solver Efficiency

- Same efficiency with the ML Seed Filter after the Greedy solver
- **Perform constantly** across the entire detector range
- Reduction of the number of tracks per event (÷2.6) at the CKF level: 12185 ➡ 4587



# Summary

- Great improvement -> x1.5 faster at almost no cost in efficiency
- Haven't had the time to optimise it yet
- ML part seems to already perform quite well, but the clustering should be improved

# Outlook

- In the process of using the association between seed and track to select the good seeds
- The impact on the reconstructed tracks should be evaluated (track parameters, hits content)

# BACKUP