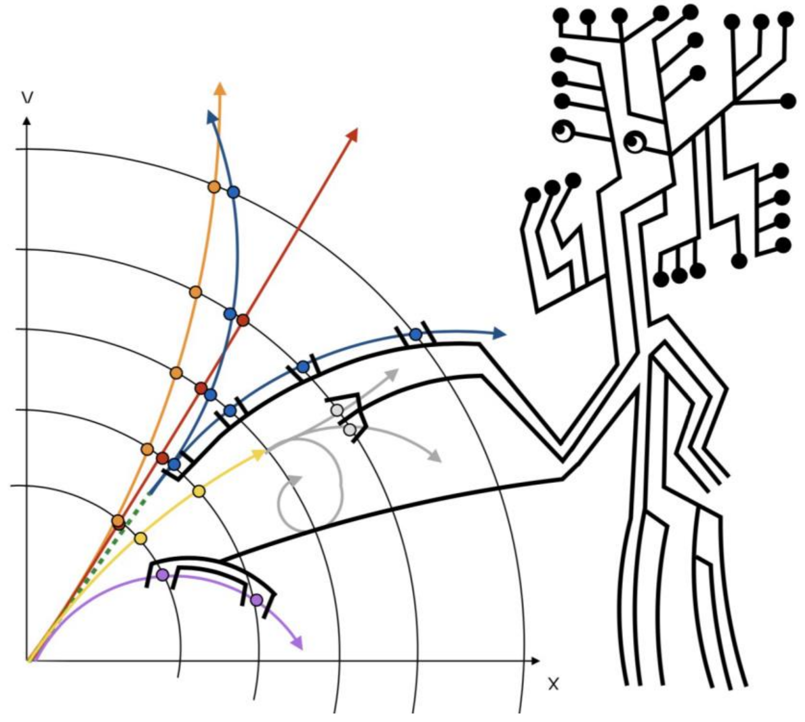
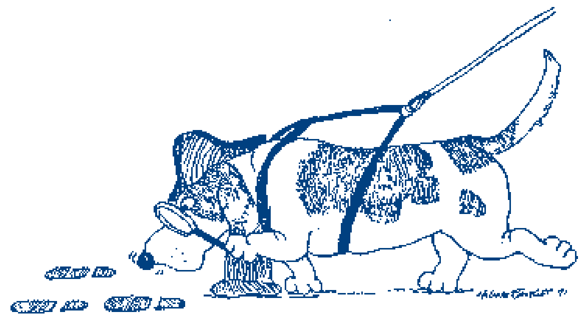
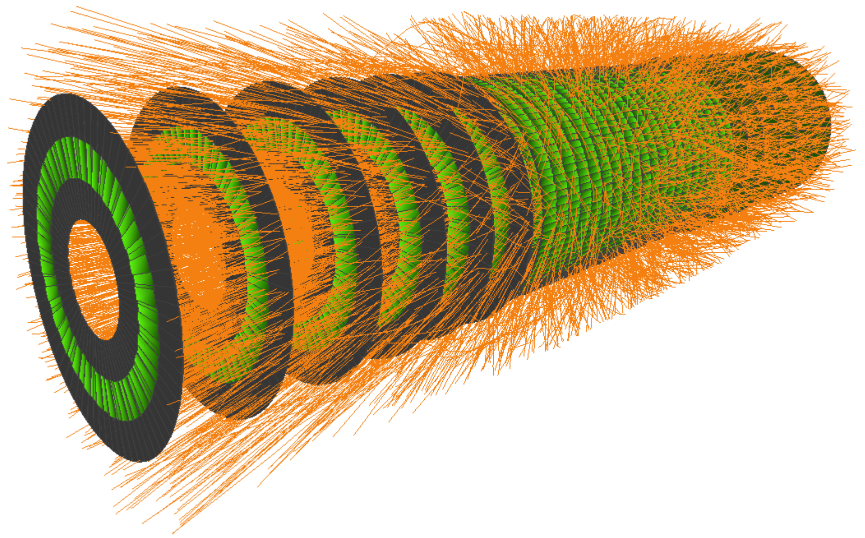
The TrackML particle tracking challenge

Dmitry Emeliyanov (RAL)







Introduction

- The TrackML was a data science competition organized in 2018 on Kaggle and Codalab platforms
- The aim of the challenge was to
  - stimulate development of new particle tracking algorithms for the High-Energy Physics (HEP)
  - get the best ideas and techniques from the Machine Learning (ML) community
- As a tracking expert (and competition participant) I will discuss today
  - the most interesting solutions submitted by the TrackML contestants
  - utilization of the ML techniques in the proposed tracking algorithms
  - potential directions for future research

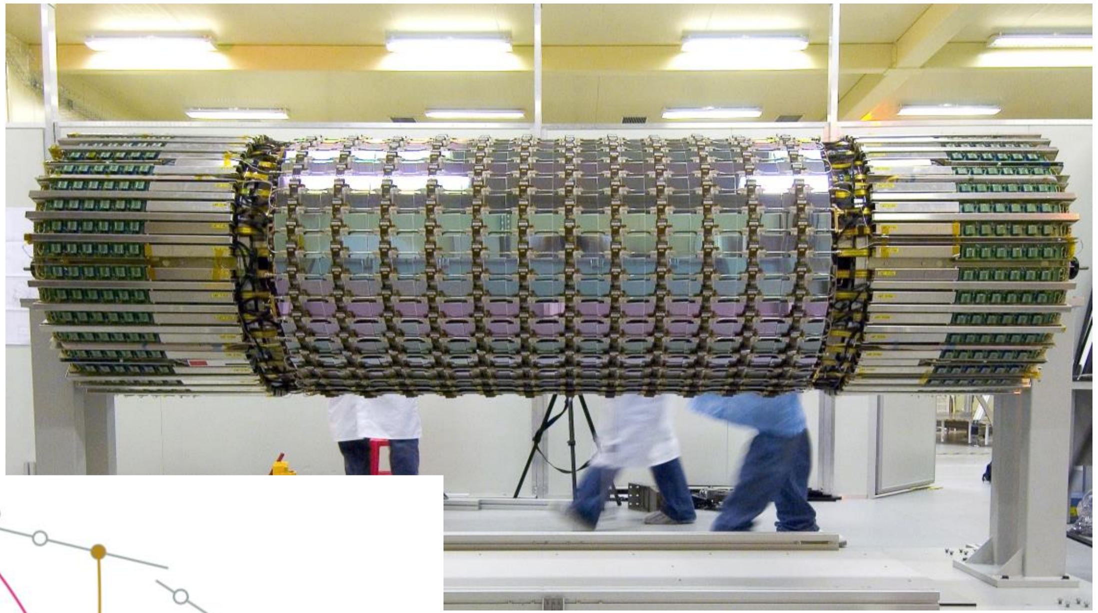
TrackML **sponsors**

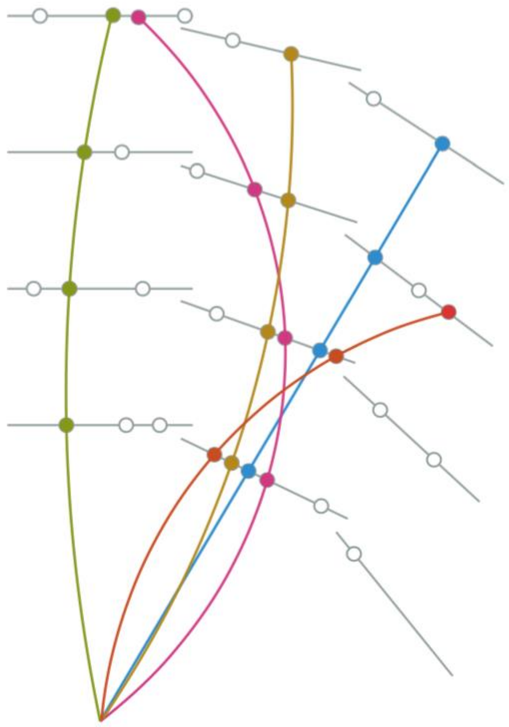
Organisation team

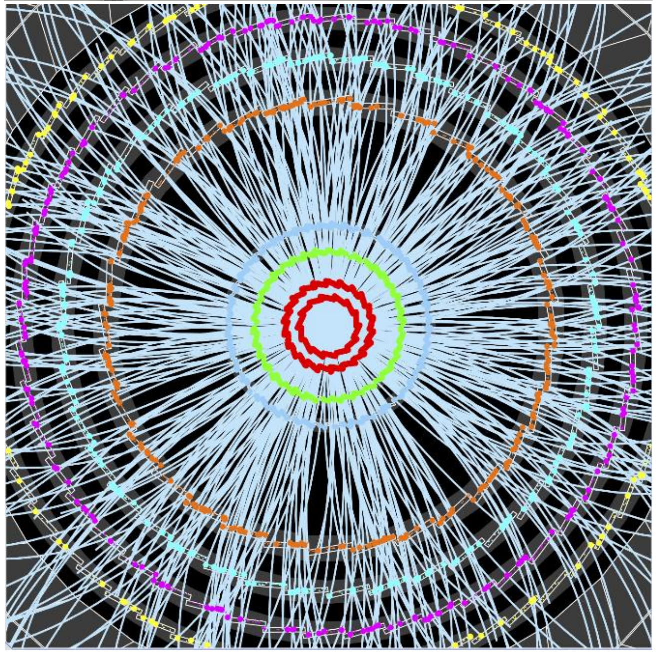
Jean-Roch Vlimant (Caltech),
Vincenzo Innocente, Andreas Salzburger (CERN),
Isabelle Guyon (ChaLearn),
Sabrina Amrouche, Tobias Golling, Moritz Kiehn (Geneva University), David Rousseau, Yetkin Yilmaz (LAL-Orsay),
Paolo Calafiura, Steven Farrell, Heather Gray (LBNL),
Vladimir Vava Gligorov (LPNHE-Paris),
Laurent Basara, Cécile Germain, Victor Estrade (LRI-Orsay),
Edward Moyse (University of Massachussets),
Mikhail Hushchyn, Andrey Ustyuzhanin (Yandex, HSE)

The tracking problem

- A future tracking detector at the LHC:
  - many thousands silicon sensors providing “shapshots” of charged particles positions with MHz frame (event) rate
  - 100 million measurement channels
  - up to 400000 measurements per event
- To find tracks we need to solve the data association problem :
  - combine measurements (“hits”) in the detector into particle trajectories
  - one of the most computationally challenging tasks in the data analysis
    - can take up to 70% of overall computing budget

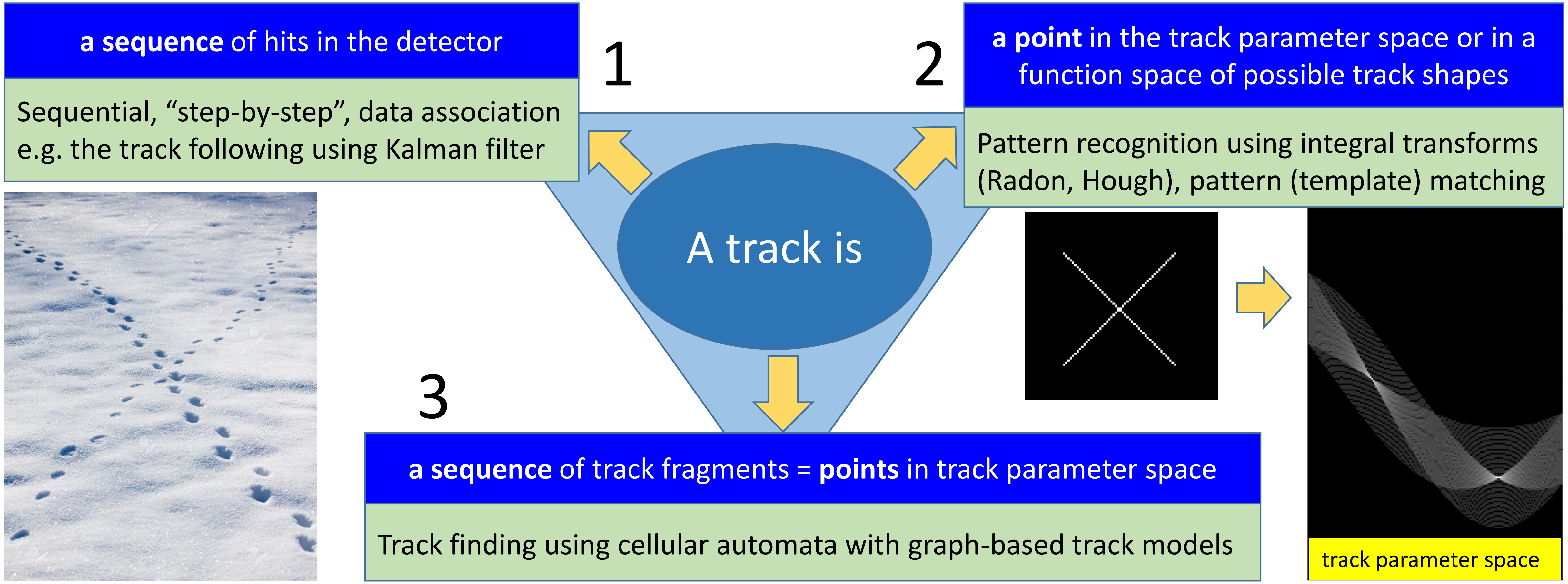






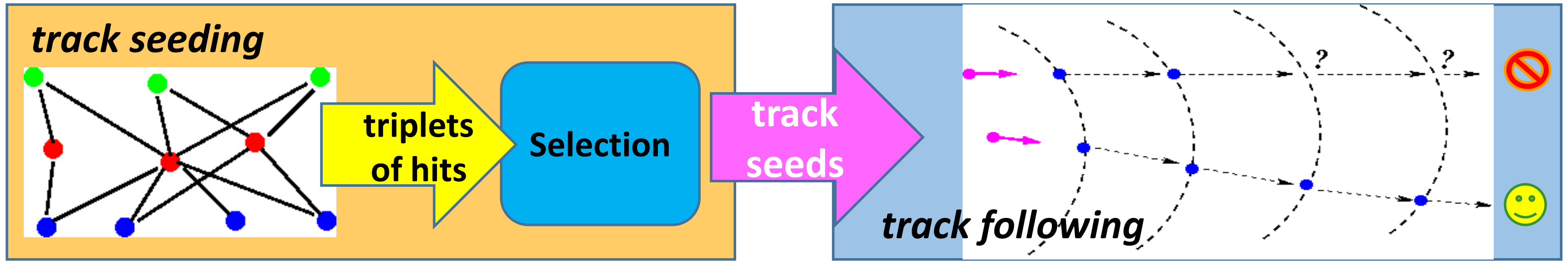
Classification of the track finding methods

- There are several groups of the track finding methods – the fundamental difference lies in the way how they define a track in the detector :


a sequence of hits in the detector
Sequential, "step-by-step", data association e.g. the track following using Kalman filter
1
2
a point in the track parameter space or in a function space of possible track shapes
Pattern recognition using integral transforms (Radon, Hough), pattern (template) matching
A track is
3
a sequence of track fragments = **points** in track parameter space
Track finding using cellular automata with graph-based track models
track parameter space

The state-of-the-art

N seeds (PU80)		N tracks
Total	Accepted	
19750	4280	1220

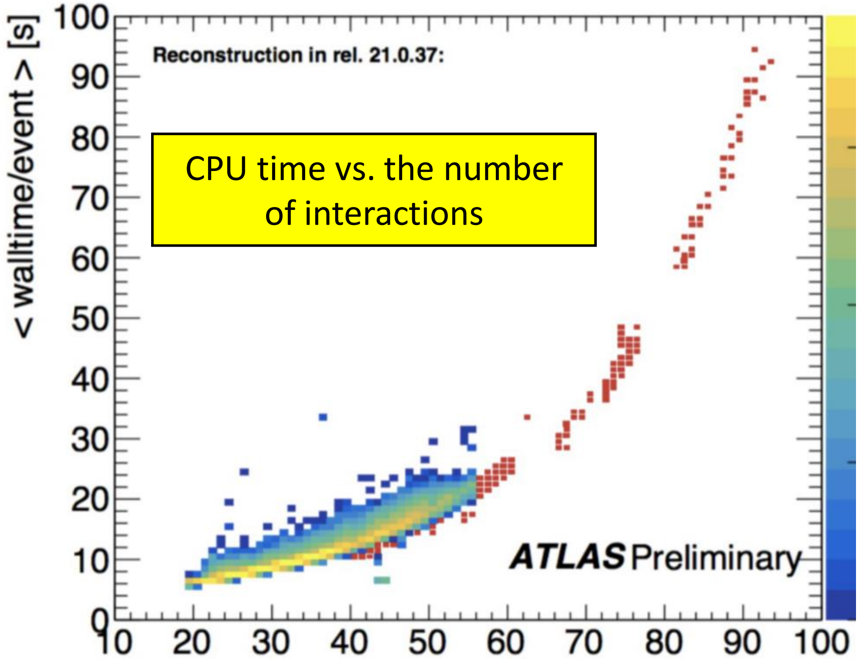


The CPU timing model :

$$T_{CPU} \sim T_{seed} + N_{seed} * T_{KF}$$

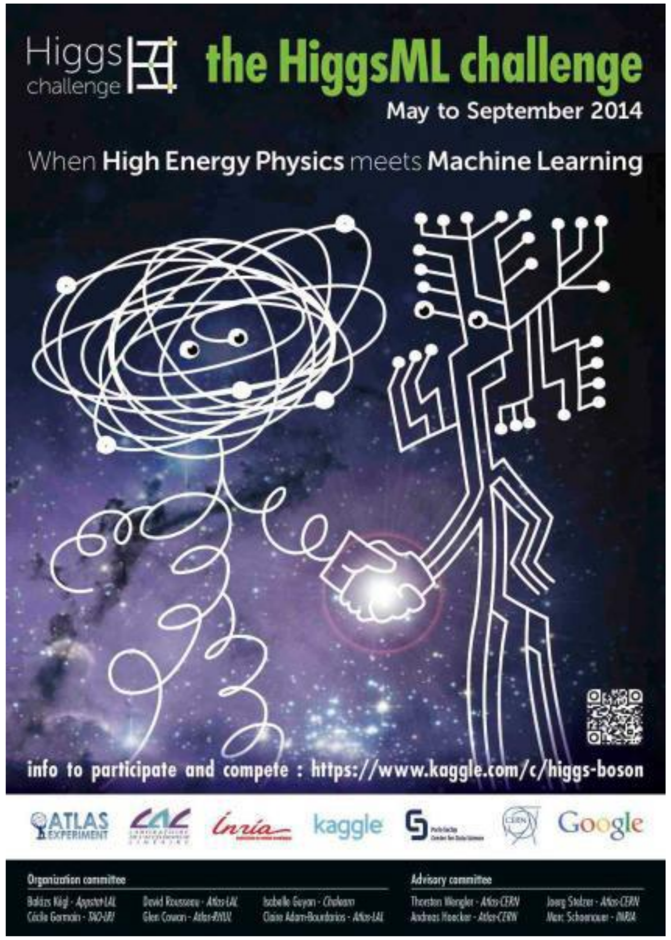
where $N_{seed} \sim N_{hits}^{\alpha}$ and $\alpha \approx 2.5$

- The Kalman filter-based track following is the main track finding algorithm used in ATLAS and CMS experiments
  - high track finding efficiency provided by the multi-pass seeding
  - Kalman filter accounts for various detector material effects providing precise track parameter estimates
- The problem with this approach is that its compute time scales non-linearly hit multiplicity
  - determined the number of simultaneous pp collisions (PileUp)
  - For the future LHC upgrade with pileup levels up to 200 we will need x10 speed-up of the tracking in order to stay within the flat budget of computing resources

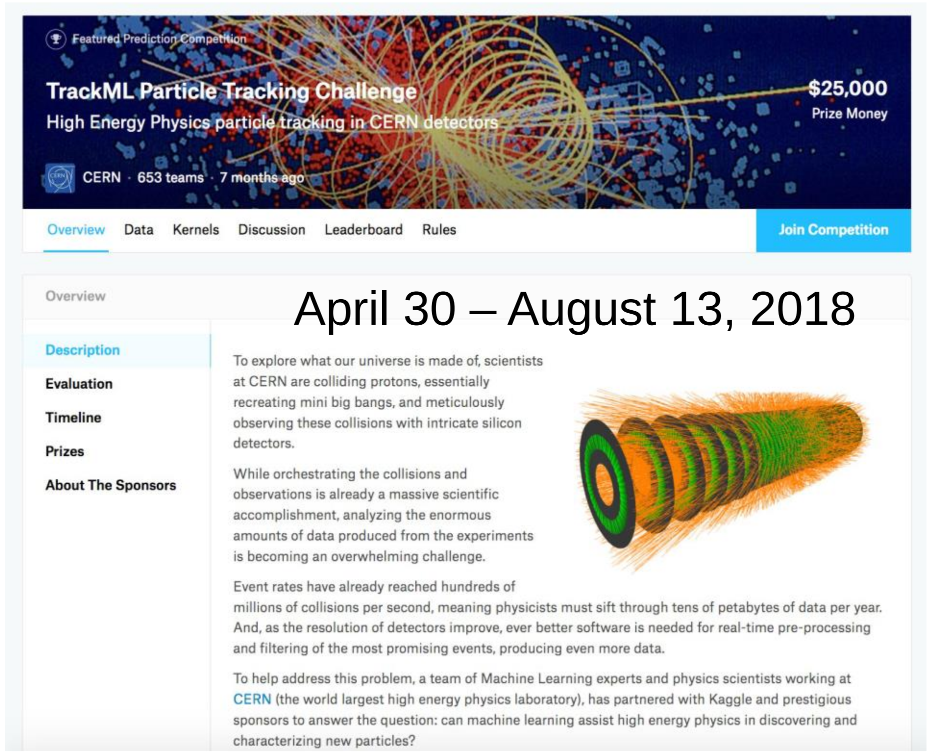


Search for alternatives

- Current algorithms for tracking are highly performant physics-wise but scale badly computation-wise
- Faster implementation of the tracking algorithms is possible with <u>dedicated hardware</u>, e.g.
  - FPGA-based Hough Transform
  - pattern matching implemented on CAM ASICs
- One of the ideas was to reach out to the ML community for new methods and possible solutions which can be implemented on <u>commodity hardware</u>
- This approach has been already successfully tried
  - the HiggsML challenge in 2014 – almost 2000 participants, the largest Kaggle competition at the time



TrackML competition



Featured Prediction Competition

TrackML Particle Tracking Challenge

High Energy Physics particle tracking in CERN detectors

CERN · 653 teams · 7 months ago

$25,000 Prize Money

Overview | Data | Kernels | Discussion | Leaderboard | Rules | Join Competition

Overview

Description
Evaluation
Timeline
Prizes
About The Sponsors

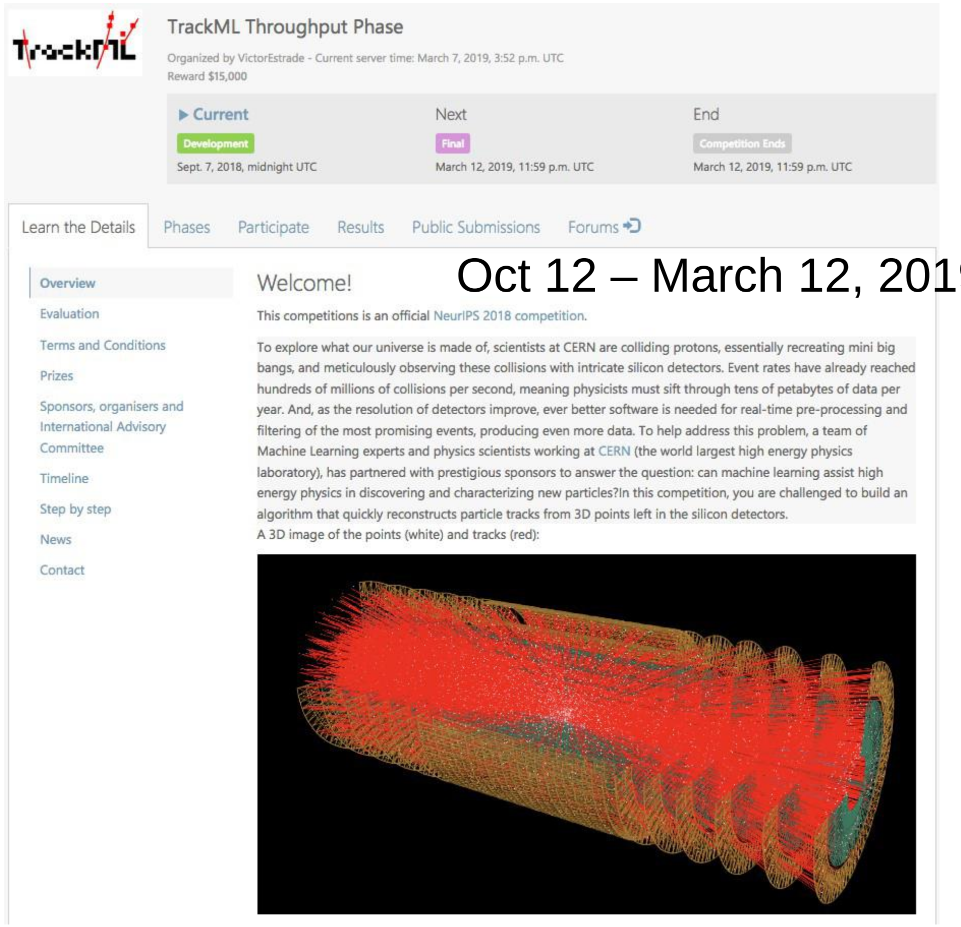
To explore what our universe is made of, scientists at CERN are colliding protons, essentially recreating mini big bangs, and meticulously observing these collisions with intricate silicon detectors.

While orchestrating the collisions and observations is already a massive scientific accomplishment, analyzing the enormous amounts of data produced from the experiments is becoming an overwhelming challenge.

Event rates have already reached hundreds of millions of collisions per second, meaning physicists must sift through tens of petabytes of data per year. And, as the resolution of detectors improve, ever better software is needed for real-time pre-processing and filtering of the most promising events, producing even more data.

To help address this problem, a team of Machine Learning experts and physics scientists working at CERN (the world largest high energy physics laboratory), has partnered with Kaggle and prestigious sponsors to answer the question: can machine learning assist high energy physics in discovering and characterizing new particles?

April 30 – August 13, 2018



TrackML Throughput Phase

Organized by VictorEstrade - Current server time: March 7, 2019, 3:52 p.m. UTC
Reward $15,000

▶ Current	Next	End
Development	Final	Competition Ends
Sept. 7, 2018, midnight UTC	March 12, 2019, 11:59 p.m. UTC	March 12, 2019, 11:59 p.m. UTC

Learn the Details | Phases | Participate | Results | Public Submissions | Forums

Overview
Evaluation
Terms and Conditions
Prizes
Sponsors, organisers and International Advisory Committee
Timeline
Step by step
News
Contact

Welcome!

This competitions is an official NeurIPS 2018 competition.

To explore what our universe is made of, scientists at CERN are colliding protons, essentially recreating mini big bangs, and meticulously observing these collisions with intricate silicon detectors. Event rates have already reached hundreds of millions of collisions per second, meaning physicists must sift through tens of petabytes of data per year. And, as the resolution of detectors improve, ever better software is needed for real-time pre-processing and filtering of the most promising events, producing even more data. To help address this problem, a team of Machine Learning experts and physics scientists working at CERN (the world largest high energy physics laboratory), has partnered with prestigious sponsors to answer the question: can machine learning assist high energy physics in discovering and characterizing new particles?In this competition, you are challenged to build an algorithm that quickly reconstructs particle tracks from 3D points left in the silicon detectors.
A 3D image of the points (white) and tracks (red):

Oct 12 – March 12, 2019

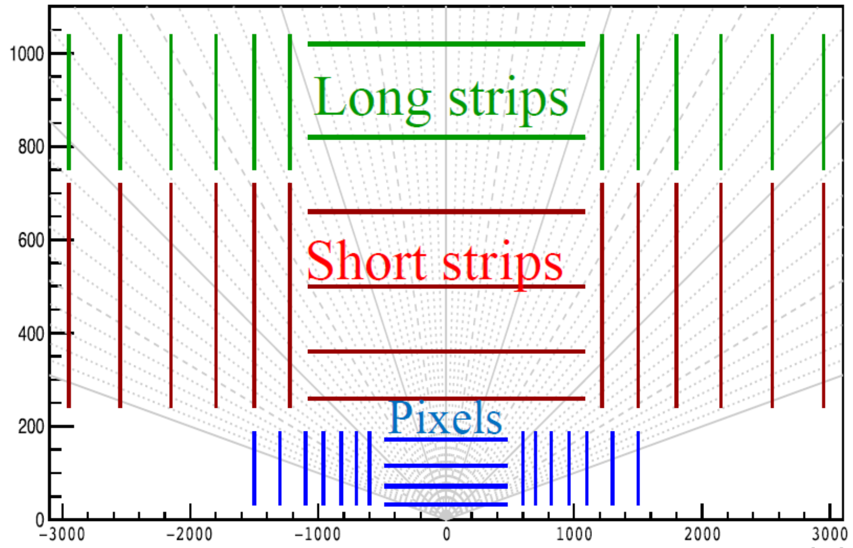
https://www.kaggle.com/c/trackml-particle-identification https://competitions.codalab.org/competitions/20112

- Data Science challenge to seek for new algorithms, including machine learning, for charged particle reconstruction
- 2 phases : accuracy (data association quality) and throughput (the inference speed)

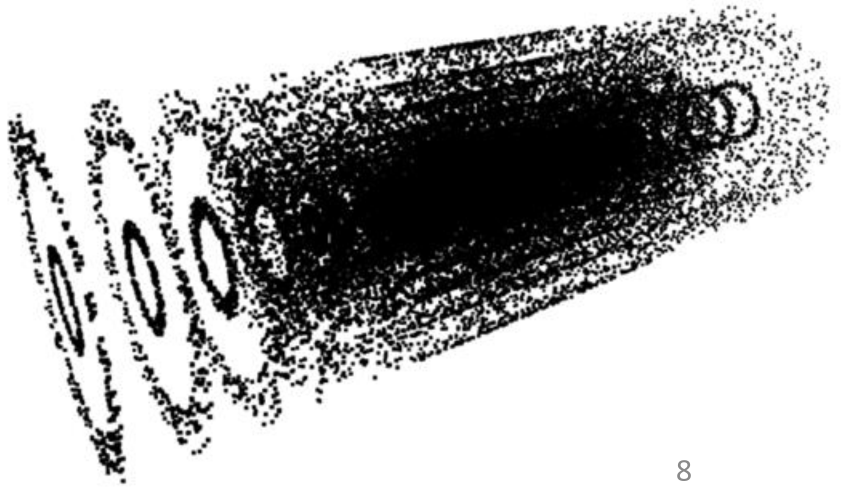
TrackML challenge in a nutshell

- Based on a simplified, yet realistic detector model:
  - non-uniform magnetic field similar to ATLAS solenoid
  - detailed simulation of particle interactions with detector material
  - three types of Si-detectors: pixel, short strips, long strips
- The goal is reconstruct all tracks in the detector:
  - 10K tracks/event, min pT = 120 MeV, min number of hits = 4
- Test data: 125 events, each event consists of
  - a list of particle position measurements (hits) in 3D space (x,y,z)
  - a list of individual silicon detector cells associated with each hit
- Training data (10K events) : the above + ground truth
  - 0.1 billion truth tracks, 1 billion hits, size O(100 Gb)
- Solution – unique hit-to-track associations for test events
  - a CSV file submitted to Kaggle for evaluation/scoring

TrackML detector geometry : r-z view



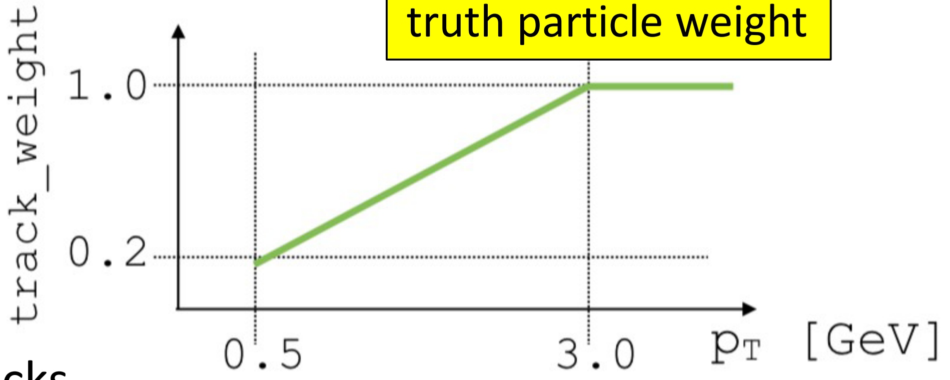
TrackML event : 100K points, 10K tracks



The solution score

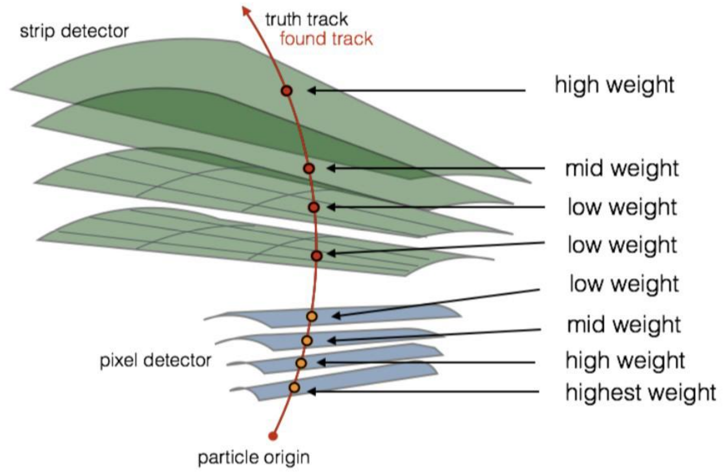
- To evaluate a data association solution we need to
  - establish correspondence between reconstructed and true tracks
  - characterize the quality of "hit-to-track" association in an event
- A track is reconstructed correctly if two requirements are met
  - at least 50% of the track hits originate from the same ground truth particle
  - at least 50% hits produced by the ground truth particle are on the track
- The score of a correct track is the sum of weights of the truth hits on the track
  - truth hit weight = truth particle weight x hit position weight
- The sum of scores over all hits in an event is normalized to 1 so that the ideal data association should result in the sum of track scores also equal to 1

truth particle weight



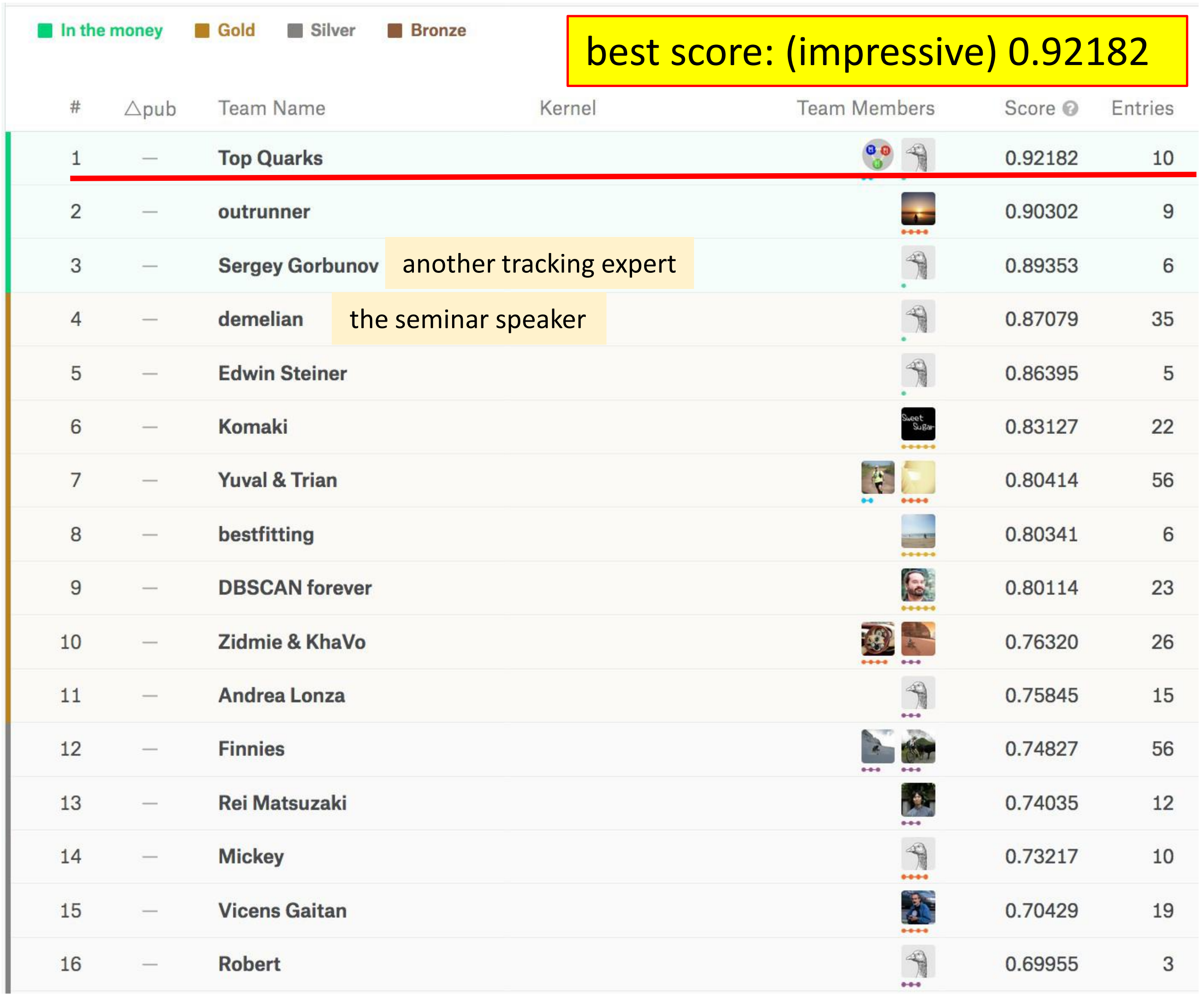
the weight depends on particle momentum to give advantage to faster tracks (more important for physics)

hit position weights



Phase 1: **accuracy phase**

TrackML Phase I : final leaderboard

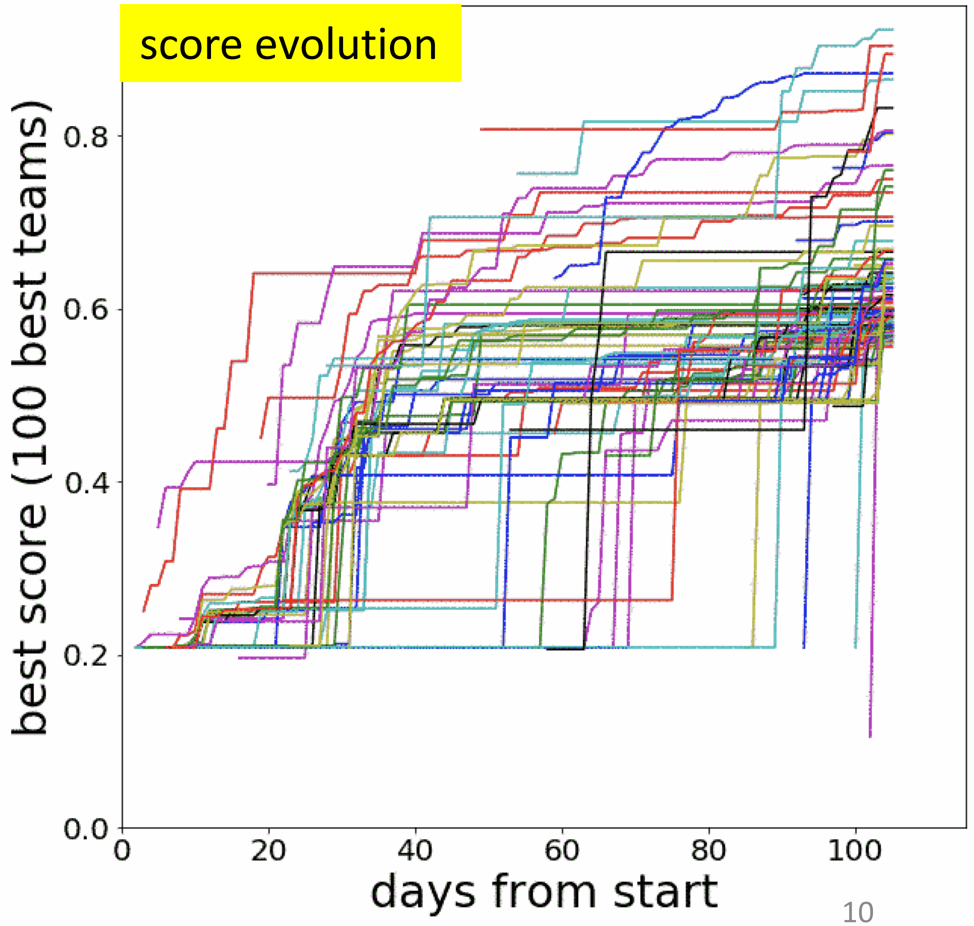


🟩 In the money 🟫 Gold ⬜ Silver 🟫 Bronze

best score: (impressive) 0.92182

#	△pub	Team Name	Kernel	Team Members	Score	Entries
1	—	Top Quarks			0.92182	10
2	—	outrunner			0.90302	9
3	—	Sergey Gorbunov	another tracking expert		0.89353	6
4	—	demelian	the seminar speaker		0.87079	35
5	—	Edwin Steiner			0.86395	5
6	—	Komaki			0.83127	22
7	—	Yuval & Trian			0.80414	56
8	—	bestfitting			0.80341	6
9	—	DBSCAN forever			0.80114	23
10	—	Zidmie & KhaVo			0.76320	26
11	—	Andrea Lonza			0.75845	15
12	—	Finnies			0.74827	56
13	—	Rei Matsuzaki			0.74035	12
14	—	Mickey			0.73217	10
15	—	Vicens Gaitan			0.70429	19
16	—	Robert			0.69955	3

- Total : more that 650 teams
  - but only 9 with the score above 0.8



The "Top Quarks" solution (1st place)

Author: J.S. Wind, master student (CS/ML) from Norway

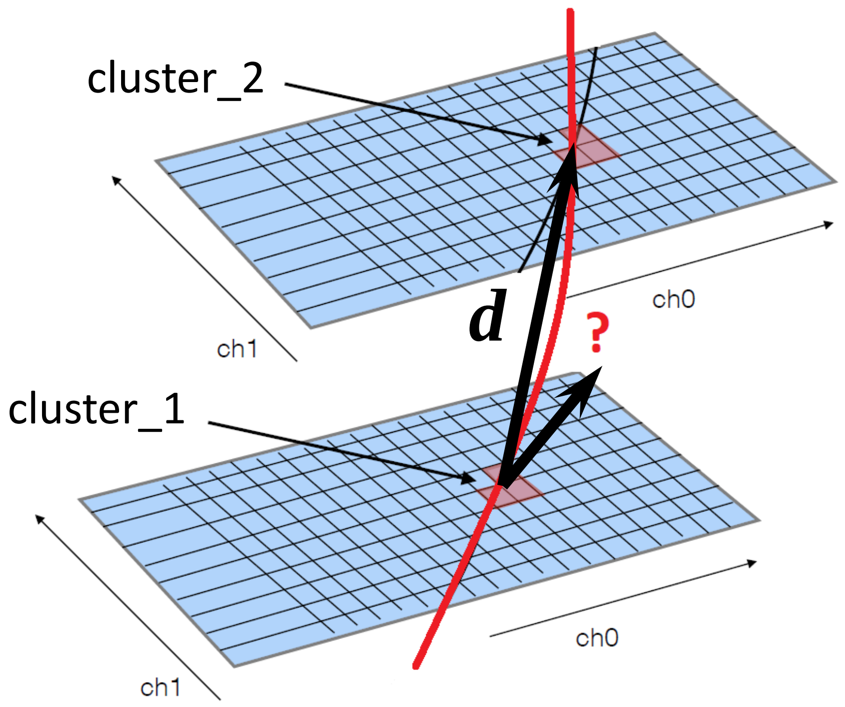
score: 0.92182

	Wall clock time	Peak memory usage
Average	7m17s	2.78GB
Max	11m20s	4.07GB

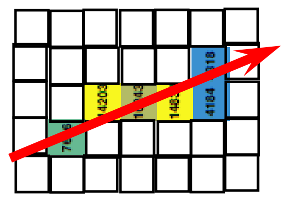
- From hit pairs to tracks:
  - Detector layers are arranged into 50 most probable pairs learnt from the training tracks
  - Hit pairs (doublets) are formed from all hits in a layer pair and a logistic regression model is used to reject wrong doublets:
    - on average, 7 million doublets are created, with enough redundancy to cover 99% of the total event score
  - Doublets are extended to the adjacent layers to form triplets, another logistic regression is used to reject bad triplets
    - up to 10 nearest neighbours per doublet are considered, 12 million triplets are created, score coverage is 97%
  - Triplets are extended to adjacent layers to form the candidate tracks – basically it's a track following which starts from all triplet seeds simultaneously
    - track extension uses the 3-point helix fitting and approximated magnetic field model learnt from data
    - 12 million tracks containing 60 million hits, many hits are shared among multiple tracks
  - Resolving hit assignment ambiguity:
    - introduce track quality = probability that the track is not random combination of hits ("noise" hit density learnt from data)
    - iterative track "competition" : the winner takes all the hits, tracks with too few remaining hits are removed

Connecting the dots

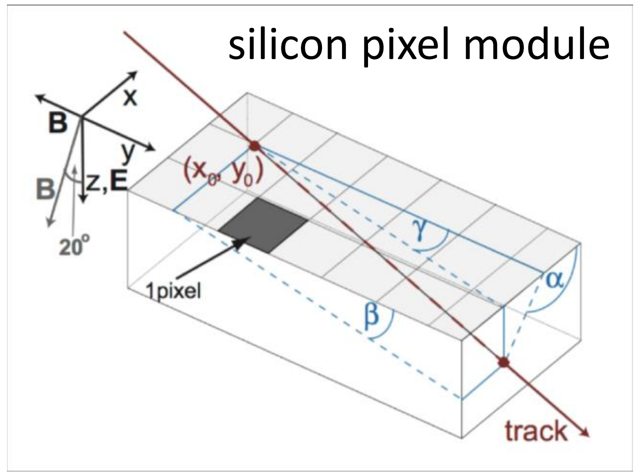
- Given two hits (clusters of silicon cells) predict if they belong to the same track



- Estimate track direction from the cluster shape :



eigenvector of covariance matrix of the silicon cells



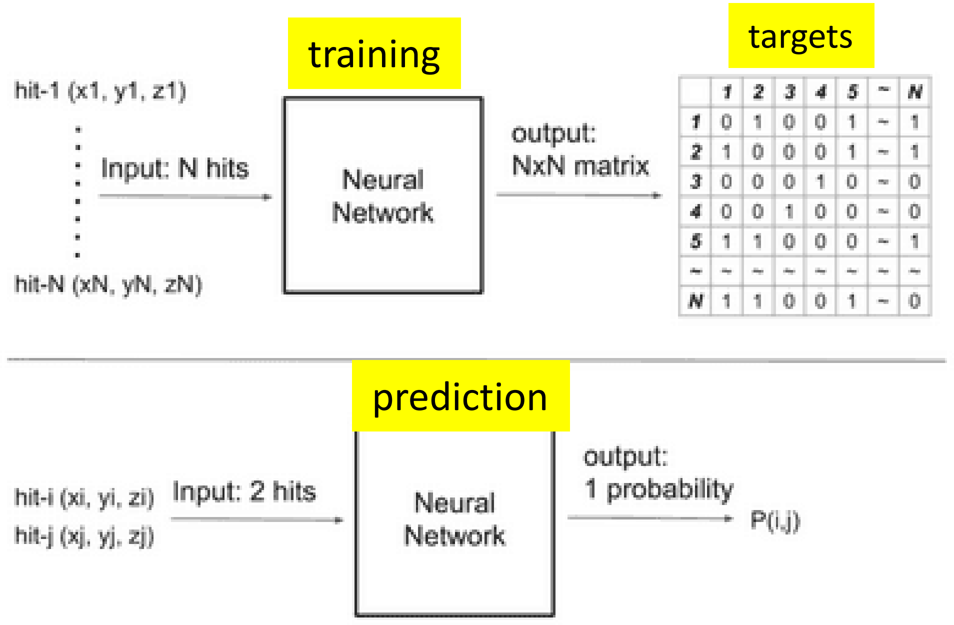
- Features for the logistic regression model :
  - angle/length deviations of the vector $\boldsymbol{d}$ projection from the values predicted by the shape of cluster 1
  - angle/length deviations of the vector $\boldsymbol{d}$ projection from the values predicted by the shape of cluster 2
  - impact parameters of the hit doublet: $z0$ and $d0$

The “outrunner” solution (2nd place)

Author: P-L. Chou, ML/software engineer (Taiwan)

score: 0.90302

- Predict hit adjacency matrix using a Deep CNN
  - 5 hidden layers with 4k - 2k - 2k - 2k - 1k nodes
  - input features (for each pair of hits) :
    - hit positions (layer+coordinates), cell counts, sums of charge depositions
    - cluster shape-based predictions for track direction
    - impact parameters of the tw-hit track segment

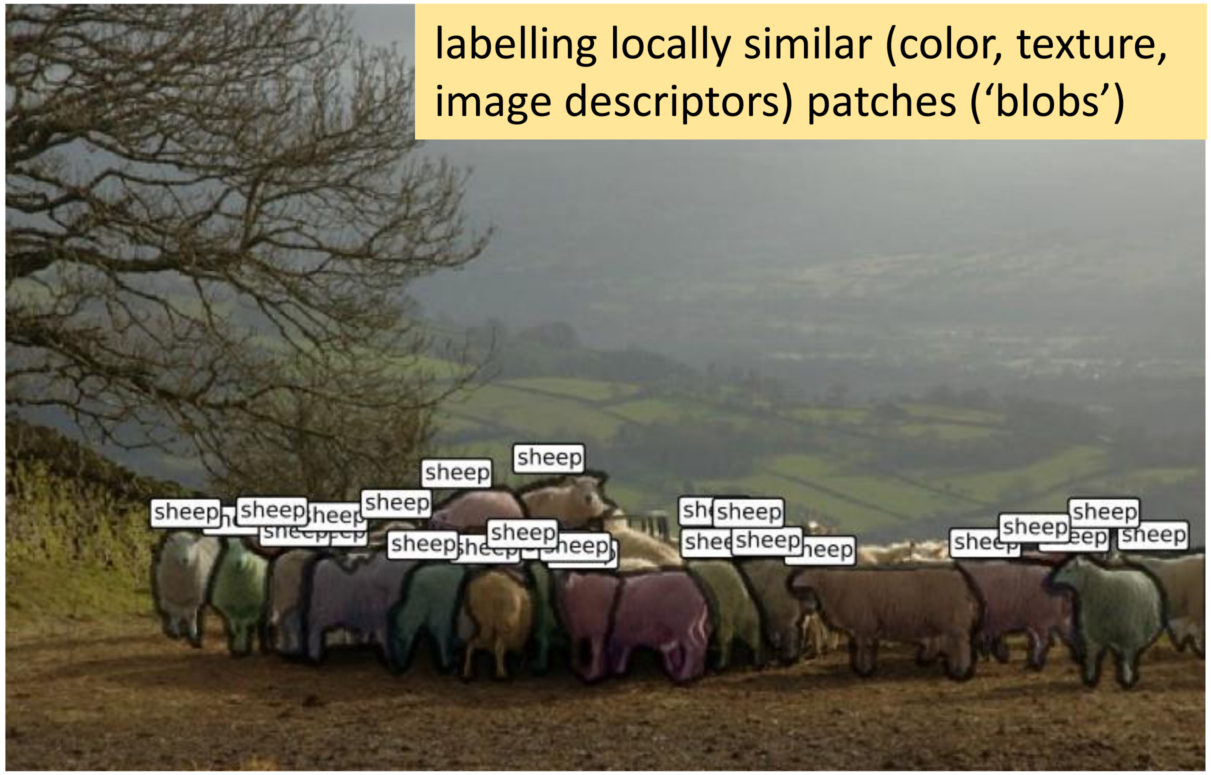


	1	2	3	4	5	~	N
1	0	1	0	0	1	~	1
2	1	0	0	0	1	~	1
3	0	0	0	1	0	~	0
4	0	0	1	0	0	~	0
5	1	1	0	0	0	~	1
~	~	~	~	~	~	~	~
N	1	1	0	0	1	~	0

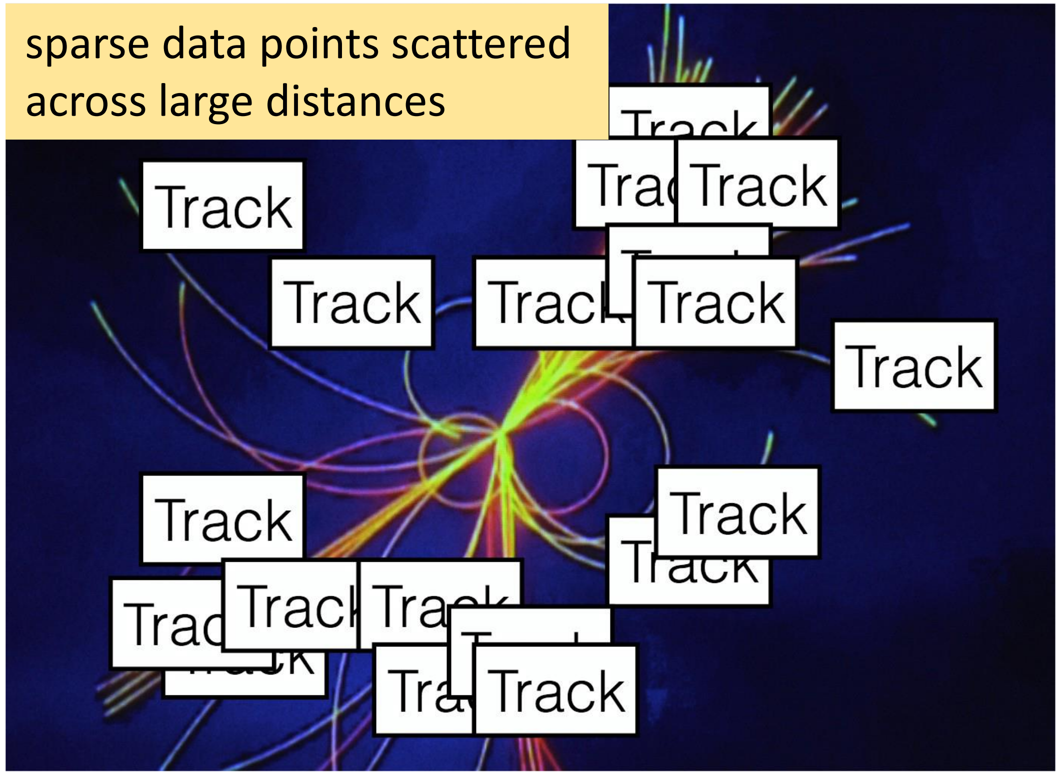
- Track extraction: start with the most probable hit pair, search the adjacency matrix for the next hit
  - check if the hit is compatible with the other hits on track by a simple helix fit
  - basically, it’s a track following which uses the adjacency matrix as the navigation tool
- Interesting algorithm but disappointingly slow implementation
  - some events took up to 3 days to process !

Sparse data association as a ML problem

- Re-casting it as a computationally tractable ML problem is a non-trivial task


labelling locally similar (color, texture, image descriptors) patches ('blobs')

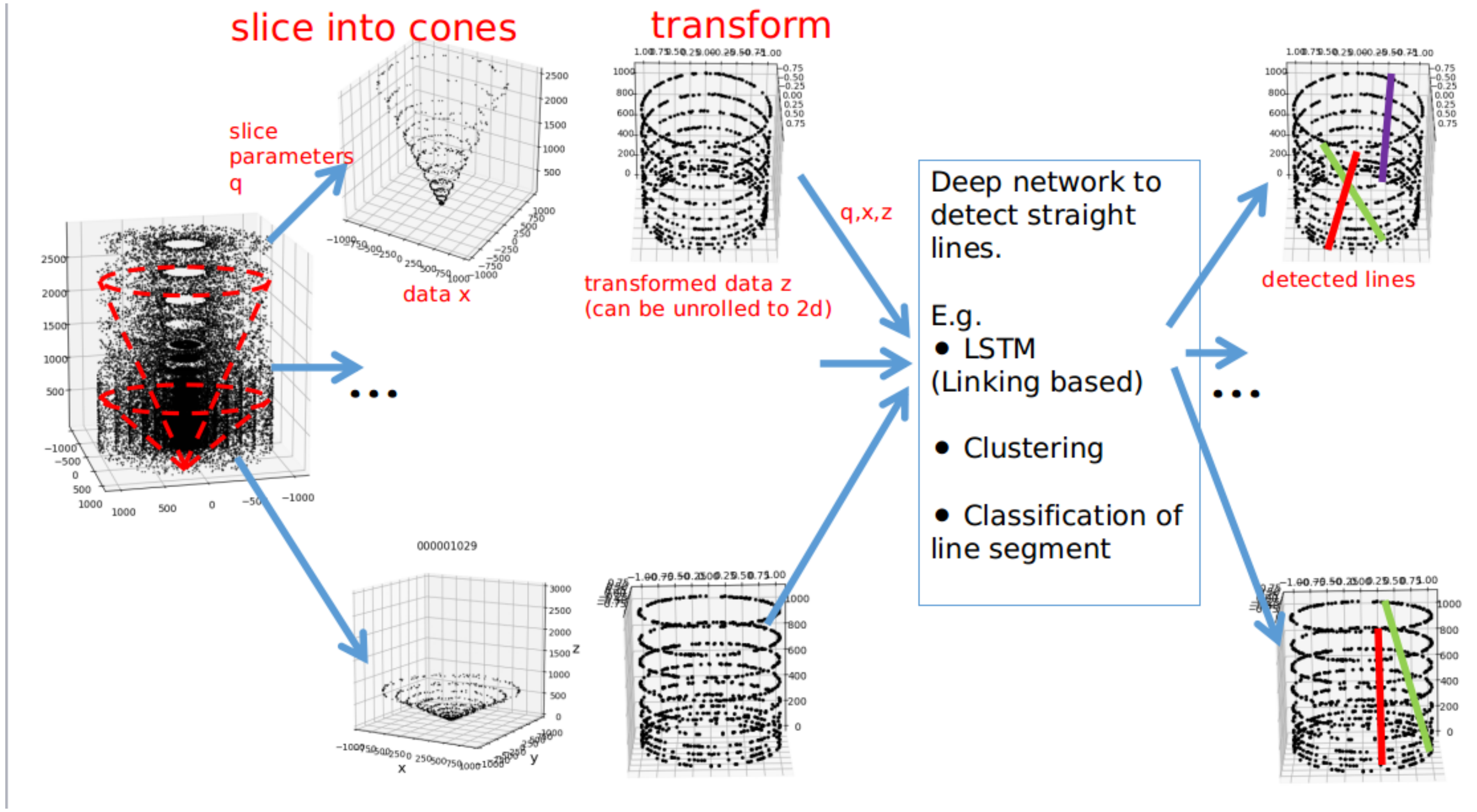
vs


sparse data points scattered across large distances

- Data association by clusterization is the most straightforward ML approach

Solutions based on clusterization and ML

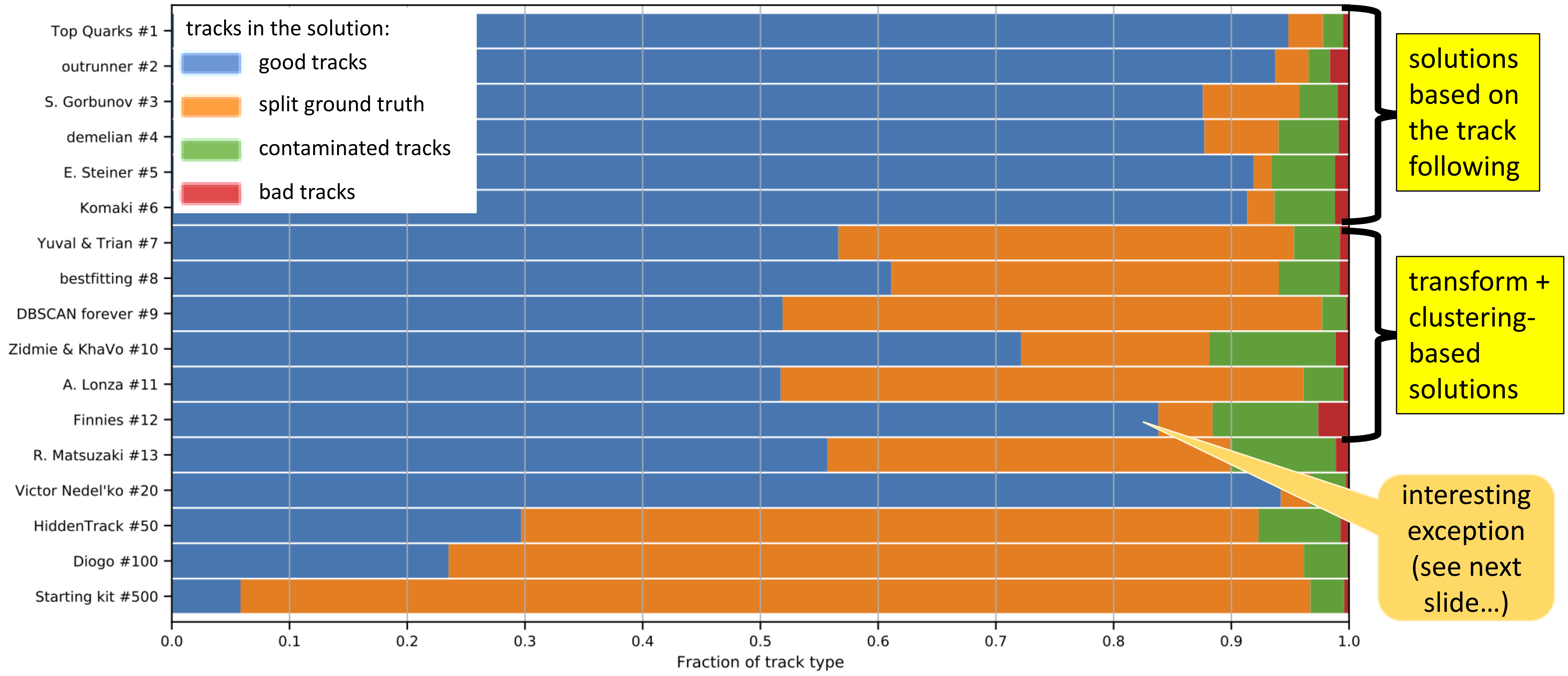
- Several submitted solutions (places #7, #9, #11) were based on the following idea:



- parametrized transform of hit coordinates so that hits from tracks with parameters close to those of the transform become close to each other in the transformed space
- track candidates extracted by clusterization of hits in the transformed space
- iterate over the transform parameters ***q*** and combine ('ensemble') the track candidates :
    - merge the track candidates into the output tracks using the "winner-takes-all" approach with some track feature-based ML classifier to calculate hit quality

- In general, the performance observed was worse than that of the track following solutions:
    - significantly lower scores, longer reported processing time (up to hours per event)

Solutions overview: score vs track set quality



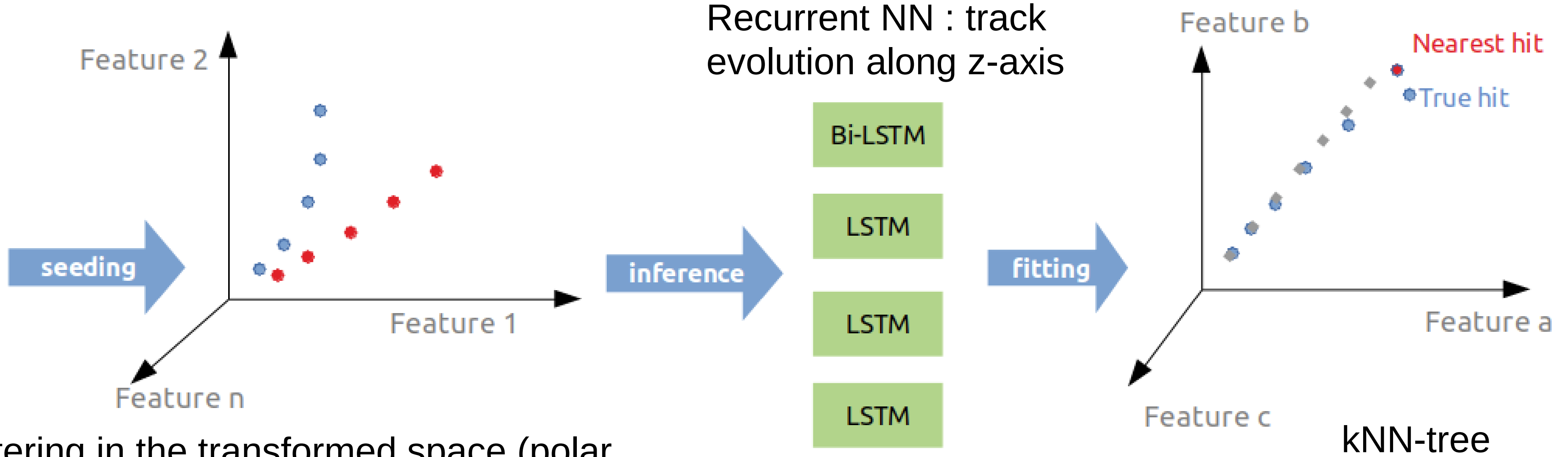
RNN-based track building (12th place)

Authors: Nicole and Liam Finnie (ML/Data Scientists, Germany)

The Deep Learning prize

score: 0.74827

- Two-step approach: seeding and track extension:



kNN-tree

clustering in the transformed space (polar coordinates) using DBScan algorithm

Track seeding (clustering)

Inference & Ensembling

Track fitting (k-nearest neighbour)

seeds are clusters truncated to the first 5 seeds

Averaged track path predictions from 5 RNN models

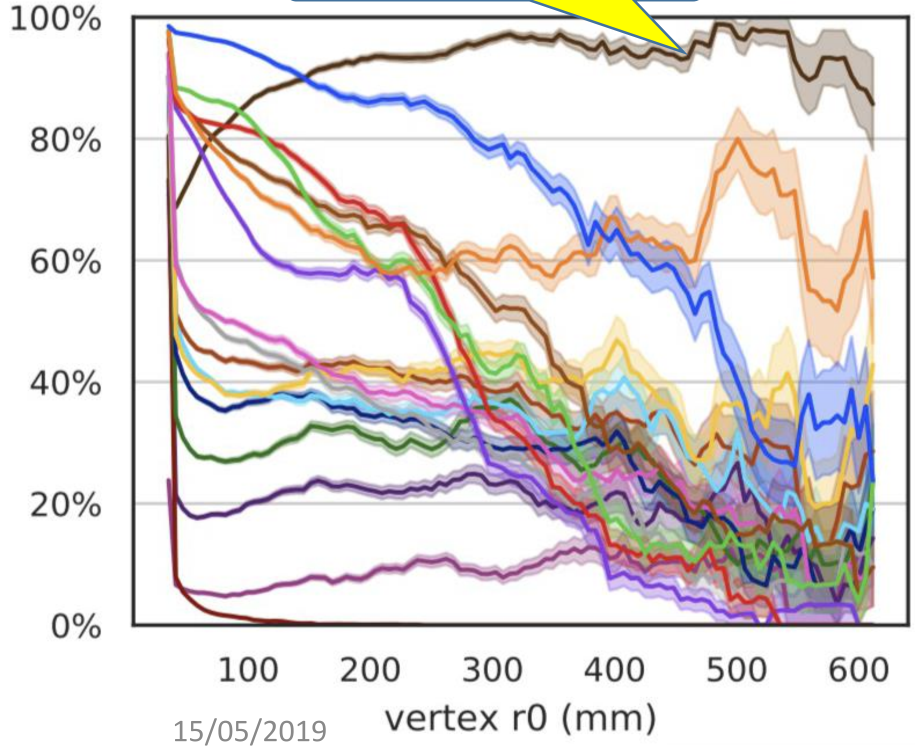
Pattern matching solution (100th place)

Author: Diogo R. Ferreira (University of Lisbon)

Organizers' prize

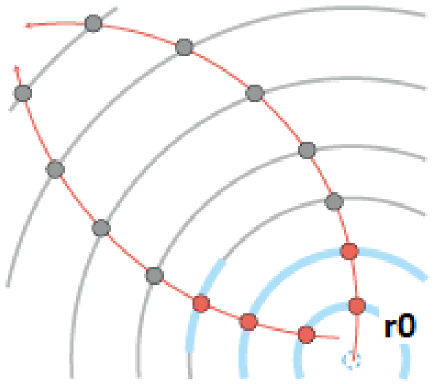
- Not the best score but a quite remarkable plot of tracking efficiency vs. track origin radius:

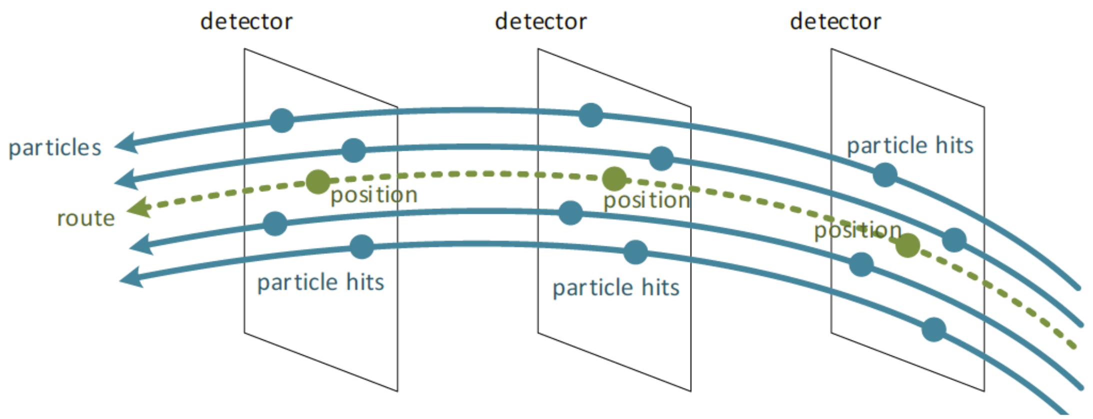
the #100 solution



The algorithm summary:

- First step is a route (i.e. pattern) data bank building :
  - *Geometry identifier (module, layer, volume) used to pre-build route patterns, route is a sequence of modules assuming training set contains all possible patterns*
- Second step is hit matching :
  - *searching through all possible routes and check if you have hits on each module – this defines a track candidate*

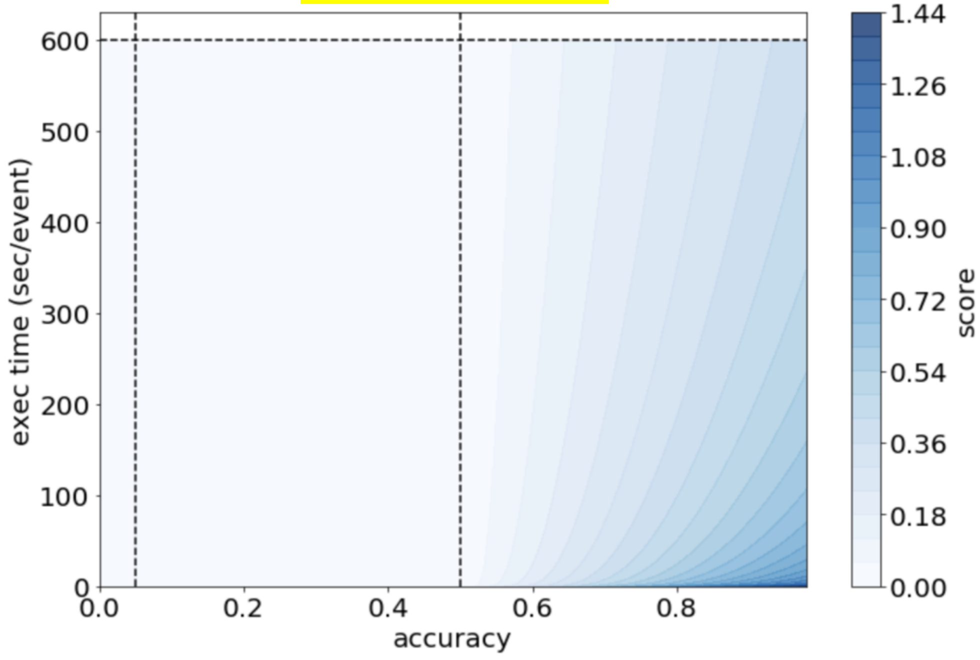




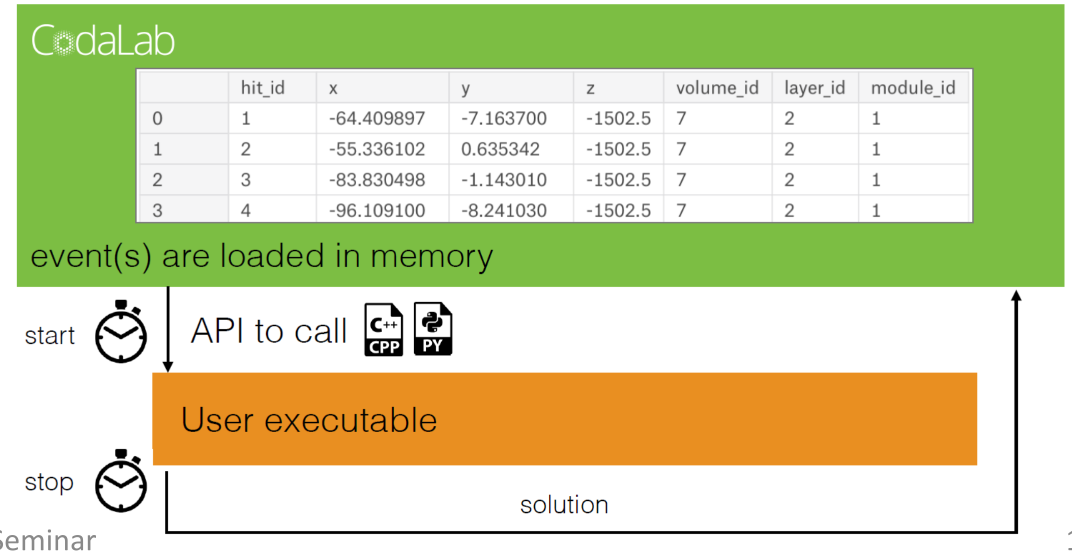
TrackML Phase II

- Focus on algorithm execution time by using the multiplicative accuracy/time score:
  - score = 0, if CPU time $t > tmax$, $tmax = 600$ seconds or accuracy score is below 0.5
  - otherwise, score = $(accuracy\ - 0.5)^2\sqrt{\log\left(1+\frac{t_{max}}{t}\right)}$

2D score map



- The CPU time was measured on CodaLab platform:
  - virtual machine: 2 x86-64 cores, 4 GB of RAM
  - CodaLab runs user-submitted Python-wrapped algorithm

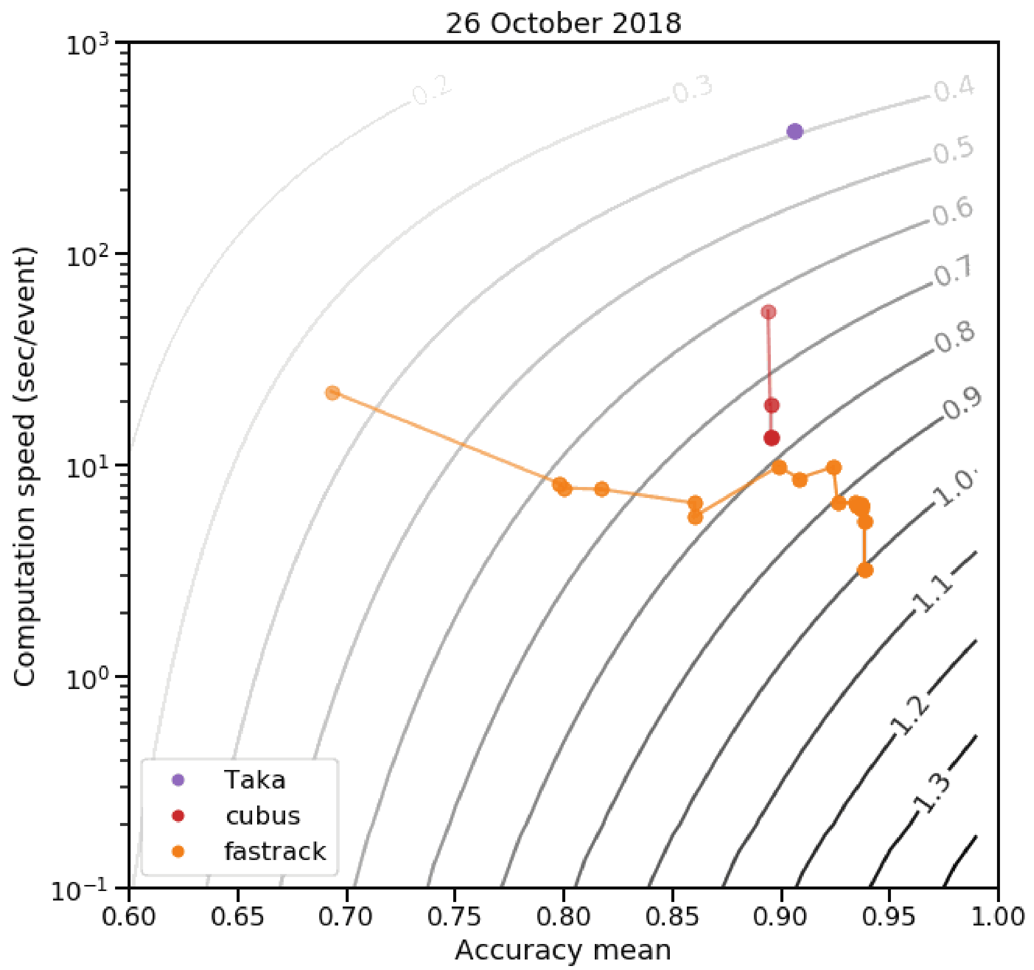


	hit_id	x	y	z	volume_id	layer_id	module_id
0	1	-64.409897	-7.163700	-1502.5	7	2	1
1	2	-55.336102	0.635342	-1502.5	7	2	1
2	3	-83.830498	-1.143010	-1502.5	7	2	1
3	4	-96.109100	-8.241030	-1502.5	7	2	1

TrackML: Phase I vs. Phase II

- Dataset: the same detector + some bugfixes:
  - bugfix in beamspot width : σ_z =5 cm (mm -> cm !)
  - max r0 = 100 mm
- Phase II had a very high entry threshold
  - due to the realistic CPU time and memory limits
  - thus excluding some of the computationally demanding Phase I solutions with good accuracy
- Much lower number of participants:
  - Kaggle competitions have greater visibility (?)
  - 150 registered (vs. 650 for Phase I), but only 3 participants with non-zero score solutions after the first three months of the competition
  - Due to low initial participation the organizers decided to extend the original 3-month Phase II by additional 4 months

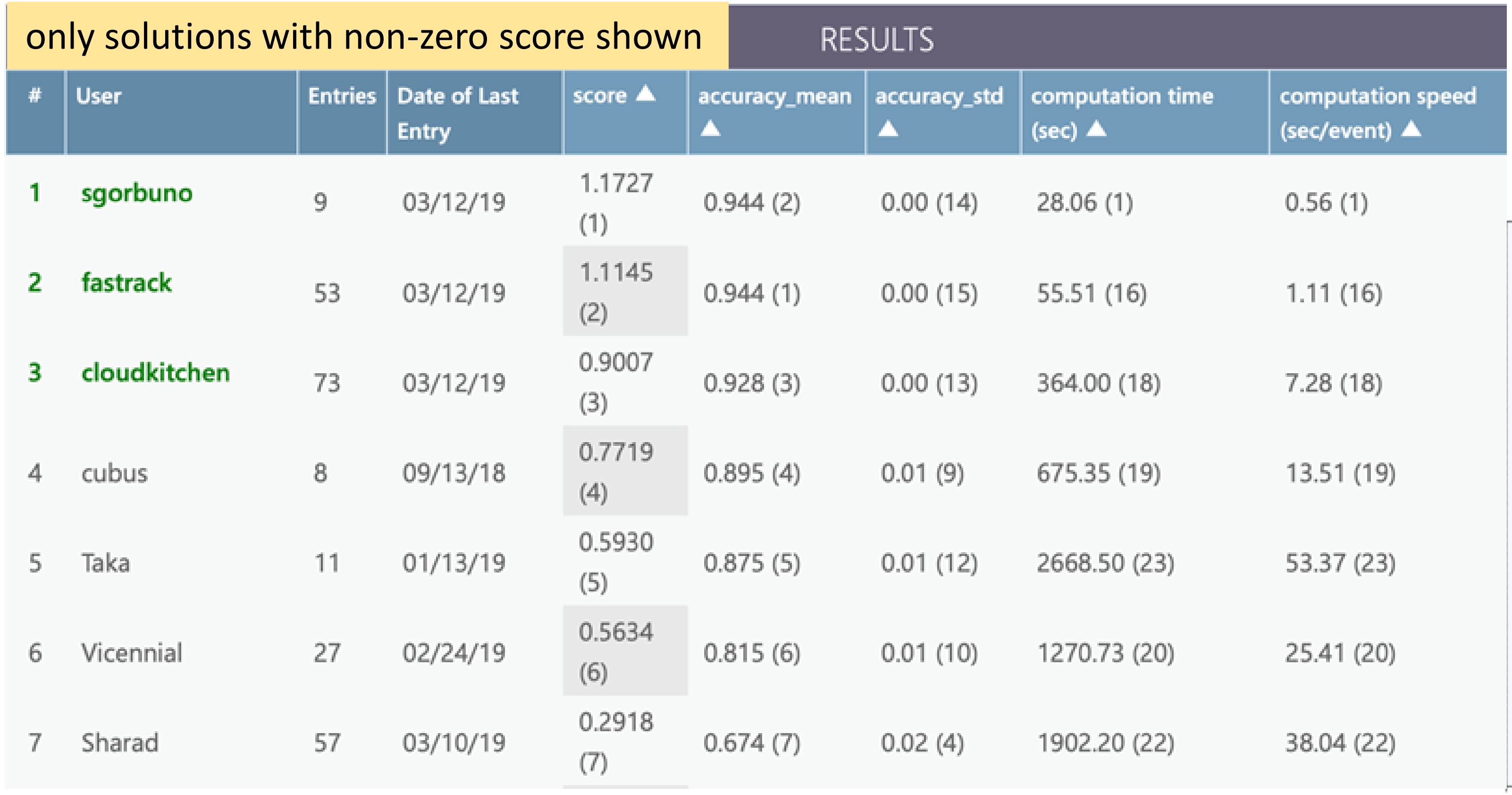
Score evolution with time



Phase 2: **throughput phase**

TrackML Phase II : final leaderboard

- preliminary conclusion from the TrackML organizers:
  - « So we've organised a tracking competition to reach out to Computer Science and the winner and runner-up are our own HEP tracking experts… »

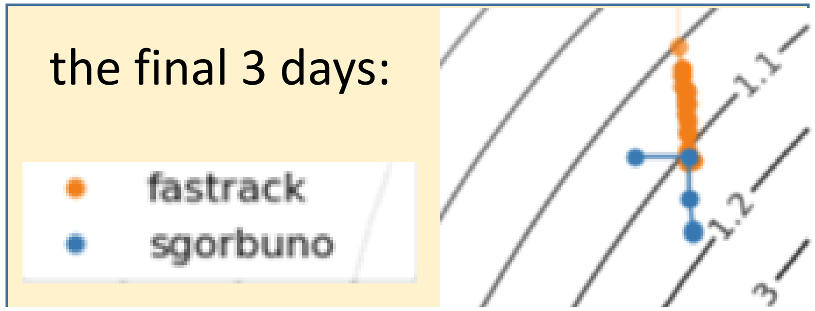


only solutions with non-zero score shown

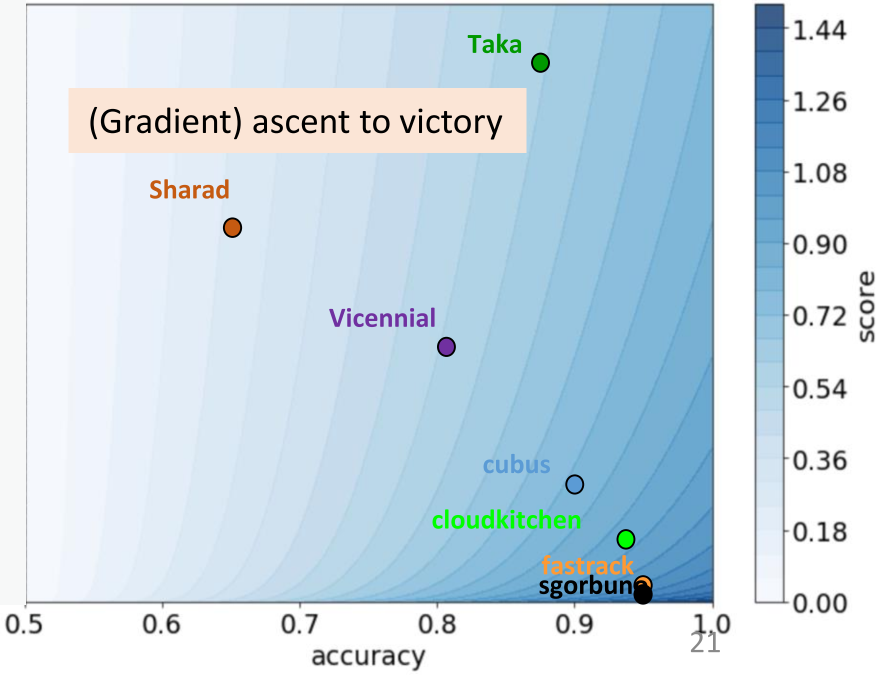
RESULTS

#	User	Entries	Date of Last Entry	score ▲	accuracy_mean ▲	accuracy_std ▲	computation time (sec) ▲	computation speed (sec/event) ▲
1	sgorbuno	9	03/12/19	1.1727 (1)	0.944 (2)	0.00 (14)	28.06 (1)	0.56 (1)
2	fastrack	53	03/12/19	1.1145 (2)	0.944 (1)	0.00 (15)	55.51 (16)	1.11 (16)
3	cloudkitchen	73	03/12/19	0.9007 (3)	0.928 (3)	0.00 (13)	364.00 (18)	7.28 (18)
4	cubus	8	09/13/18	0.7719 (4)	0.895 (4)	0.01 (9)	675.35 (19)	13.51 (19)
5	Taka	11	01/13/19	0.5930 (5)	0.875 (5)	0.01 (12)	2668.50 (23)	53.37 (23)
6	Vicennial	27	02/24/19	0.5634 (6)	0.815 (6)	0.01 (10)	1270.73 (20)	25.41 (20)
7	Sharad	57	03/10/19	0.2918 (7)	0.674 (7)	0.02 (4)	1902.20 (22)	38.04 (22)

the final 3 days:



(Gradient) ascent to victory

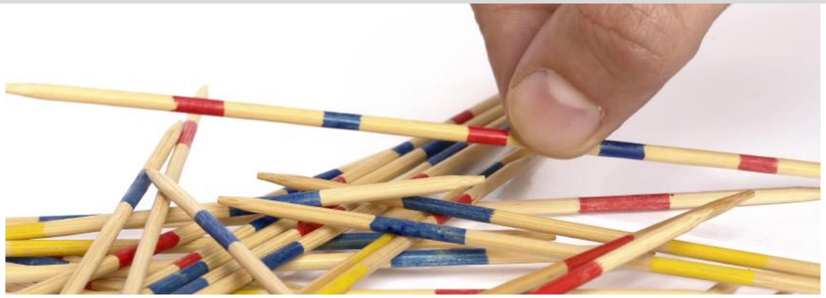


Mikado tracker: 1st place solution

Accuracy score: 0.944
Time/event: 0.56 sec
Memory: 0.1-0.18 Gb (1-2 cores)

Author: S. Gorbunov, HEP tracking expert (ALICE) (FIAS, Germany)

Wikipedia: *Mikado* is a pick-up sticks game originating in Europe. The players take turns, in which one stick after another should be taken up without moving others

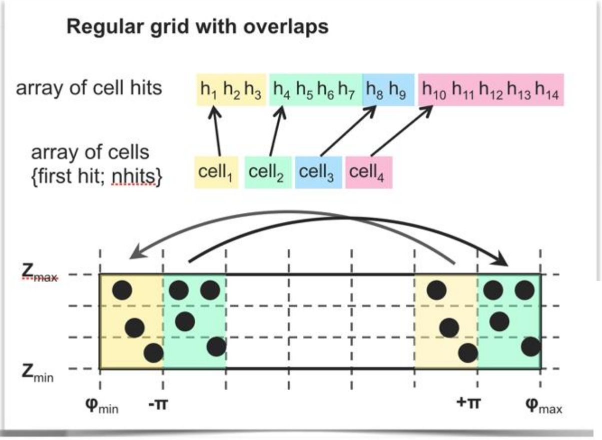


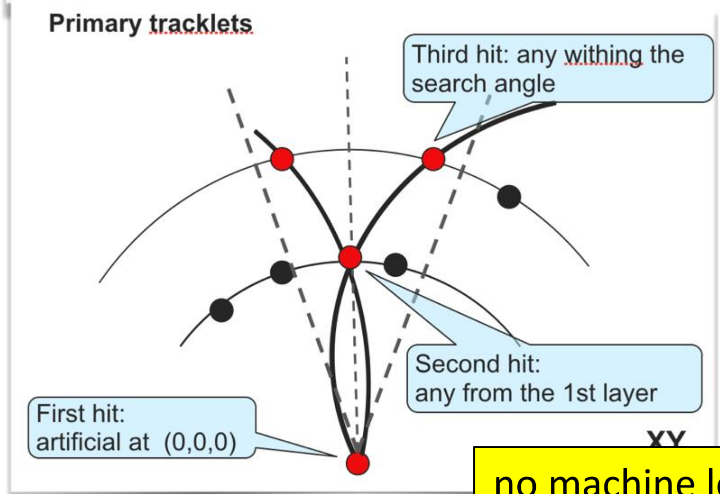
Phase 1 Sergey Gorbunov

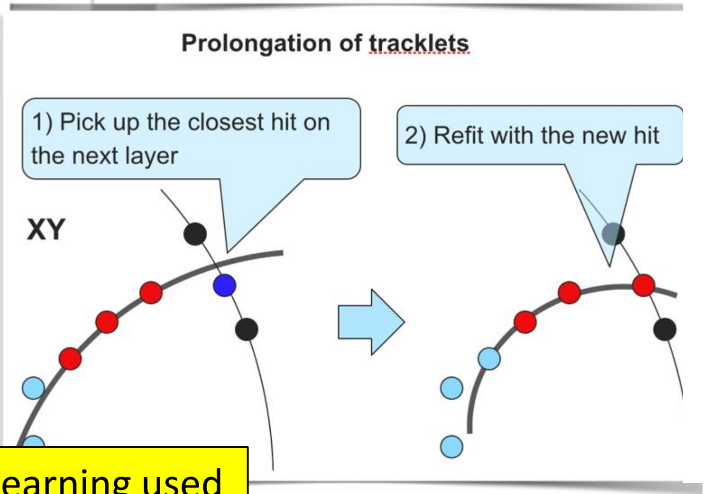
Execution time
1.2 min on single core 2.6 GHz CPU

3rd place in Phase I, score 0.8935

- A combinatorial algorithm, based on the track following method
- No search branches enabled in Phase II
- Simple track model: local 3-hit helix
- Fast data access


Regular grid with overlaps
array of cell hits
$h_1\ h_2\ h_3\ h_4\ h_5\ h_6\ h_7\ h_8\ h_9\ h_{10}\ h_{11}\ h_{12}\ h_{13}\ h_{14}$
array of cells {first hit; nhits}
$cell_1$ $cell_2$ $cell_3$ $cell_4$
Z_{max}
Z_{min}
φ_{min} $-\pi$ $+\pi$ φ_{max}


Primary tracklets
Third hit: any withing the search angle
Second hit: any from the 1st layer
First hit: artificial at (0,0,0)
XY


Prolongation of tracklets
1) Pick up the closest hit on the next layer
2) Refit with the new hit
XY

no machine learning used

- Multi-pass (80 passes!) track finding:
  - starting with high-momentum (i.e. straight-line) tracks from the origin
  - associated hits removed after each pass
  - magnetic field map learnt from data
  - each pass has it's own set of parameters
    - in total, there are O(10K) parameters to tune using the hyper-parameter scan – takes a few days to run on a 40-core machine with 40 training events

FASTrack: 2nd place solution

Author: D. Emeliyanov, HEP tracking expert (RAL PPD)

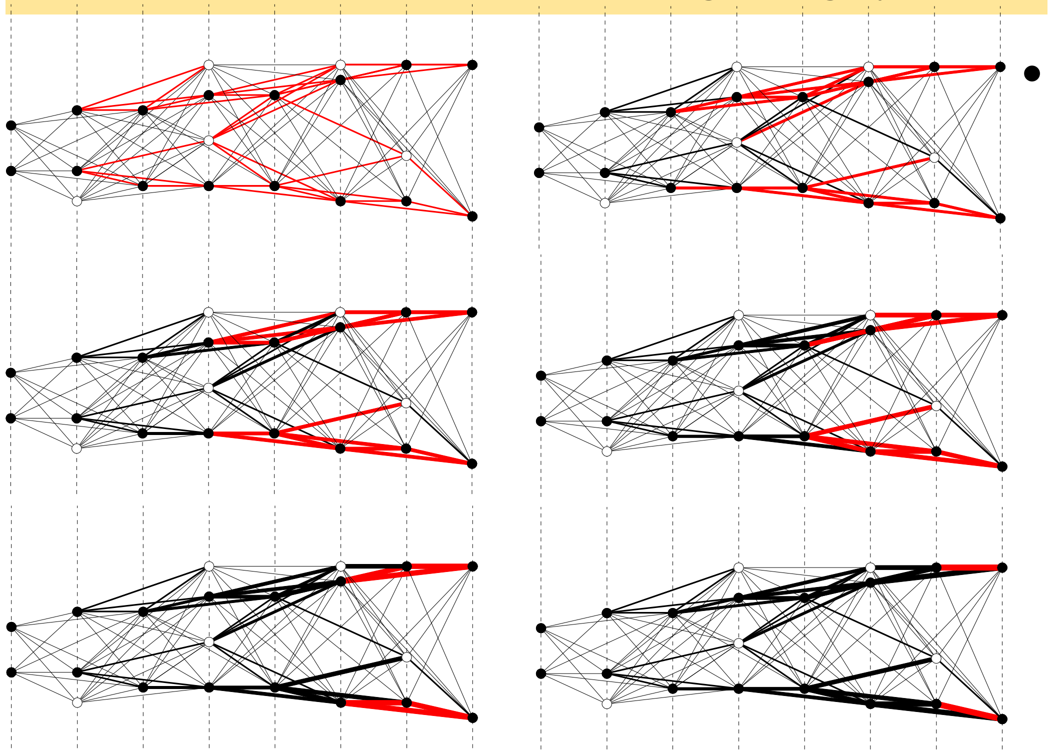
Accuracy score: 0.944
Time/event: 1.1 sec
Memory: 0.6 Gb

after post-competition bug-fixes:

Accuracy score: 0.948
Time/event: 0.8 sec

4th place in Phase I, score 0.87

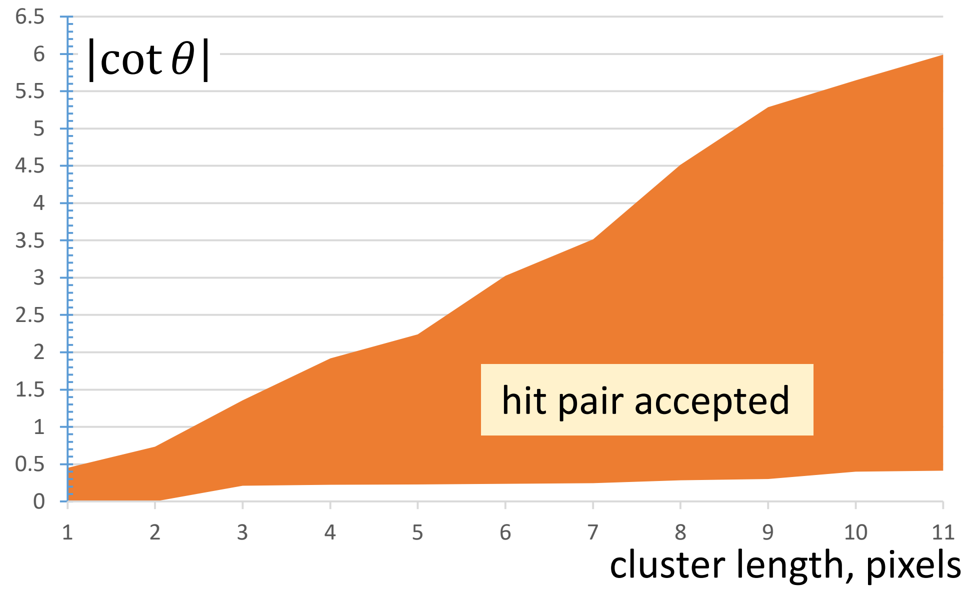
Cellular Automaton on a track segment graph

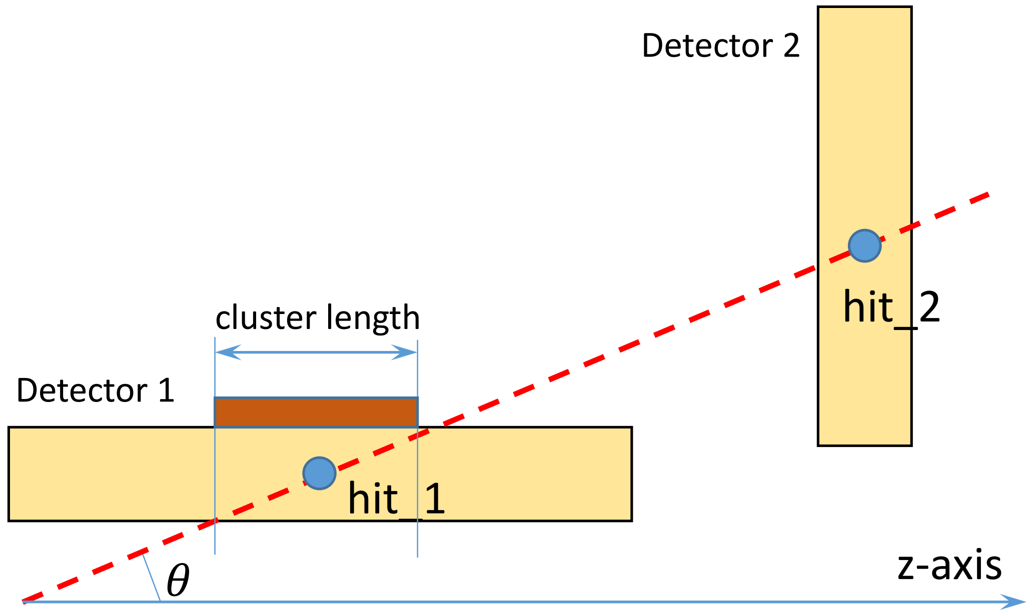


- Key ideas and techniques used :
  - track finding in 3 passes with hit masking after each pass:
    - high-pT central tracks, low-pT central tracks, the rest of the tracks
  - the same tracking workflow used for each pass :
    - traversal of a hit-pair graph with an embedded Cellular Automaton (CA) and Kalman filter for fast discovery of track candidates
    - track completion by the track following
  - using clusters shape to predict track inclination angles and avoid hit pairs incompatible with the prediction:
    - it was added for the Phase II version and boosted the accuracy to the record 0.948 and made the algorithm much faster

- The CA and track following (TF) complements each other: the CA discovers the bulk of the tracks and the TF puts “finishing touches” by
  - extending tracks to the layers not covered by the CA network and filling “holes” on tracks, basically, compensating loss of hits if only a short fragment of a track was found by the CA

FASTrack : hit pair prediction



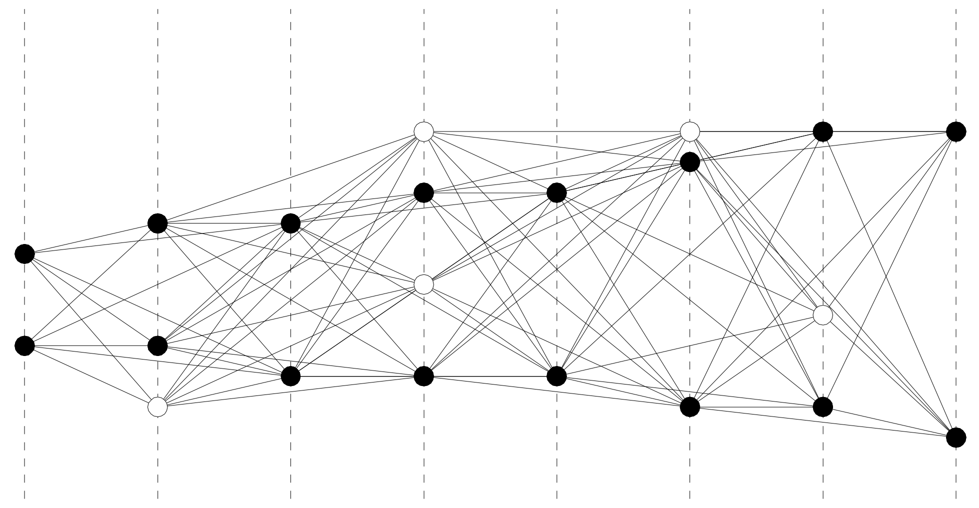


- The hit pair creation and the CA graph building is the time-critical part of the algorithm
  - because of huge number of pairs (1-1.5 million)
  - similar to the “Top Quarks” and “outrunner” solutions the FASTrack uses cluster shapes to predict if two hits belong to the same track
  - min/max $|\cot\theta|$ as functions of cluster width were extracted from training data
  - for faster prediction the decision function was approximated by a LookUp Table (LUT)

Track finding on graphs

- General idea:
  - find short track segments, e.g. pairs or triplets of hits
  - create the graph of track segments by setting *admissible* connection between segments
  - a track is an optimal (in some sense) path on the graph

- Example:
  - the segments connect hits
    - in the adjacent layers
    - across one layer
  - admissible connections:
    - segments have common hit
    - small breaking angle $\Delta\varphi(s_i, s_{i+1})$



Recursive track search on a graph

- Let $S_k = \{s_1, \ldots, s_k\}$ be a sequence of *connected* track segments
- Consider the following "best-path" criterion
  - the number of connected segments with imposed cut on the breaking angles

$$J(S_k) = \sum_{i=1}^{k-1} I\left(\left|\Delta\varphi\left(s_i, s_{i+1}\right)\right| \le \Phi\right), \quad I(x) = \begin{cases} 1, \text{if } x = \text{true} \\ 0, \text{if } x = \text{false} \end{cases}, \quad \Phi \text{ is a cut}$$

- $J^* = \max_S J(S)$ satisfies Bellman recursion property:

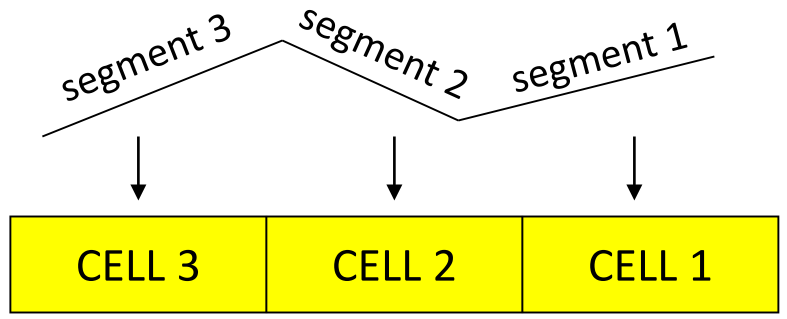
$$J^*\left(S_{k+1}\right) = J^*\left(S_k\right) + \max_{s_{k+1}} I\left(\left|\Delta\varphi\left(s_k, s_{k+1}\right)\right| \le \Phi\right)$$

  – so that optimal segment sequences (i.e. track candidates) can be constructed recursively by the dynamic programming

The cellular automaton approach

Track segments are called ***cells***, cell can have ***neighbours*** – connected segments from the left

Each cell has a ***state*** – positive integer number



Iter 0	s=1	s=1	s=1
Iter 1	s=1	s=2	s=2
Iter 2	s=1	s=2	s=3

The algorithm works on the cell set and has two passes:

• <u>Forward recursion</u>:
if cell ***i*** with the state s_i has a neighbour ***j*** with the equal state $s_j=s_i$, then, at the next iteration, the state is incremented:

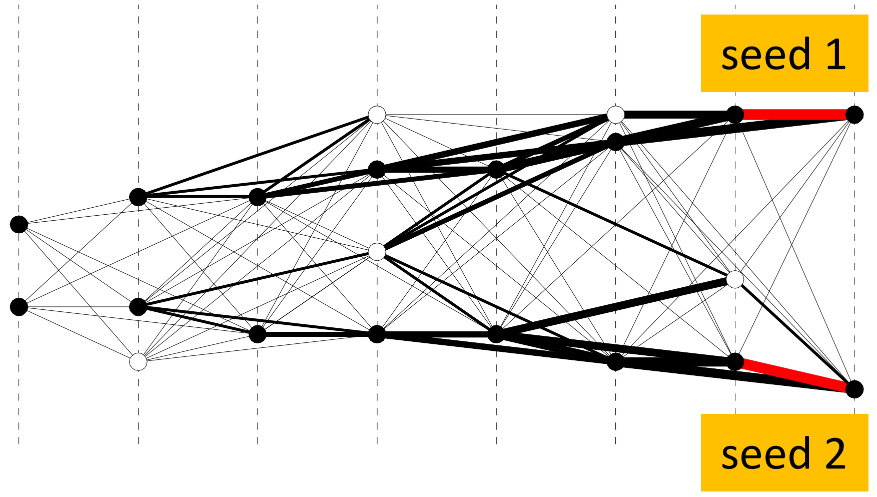
$$s_i=s_i+1$$

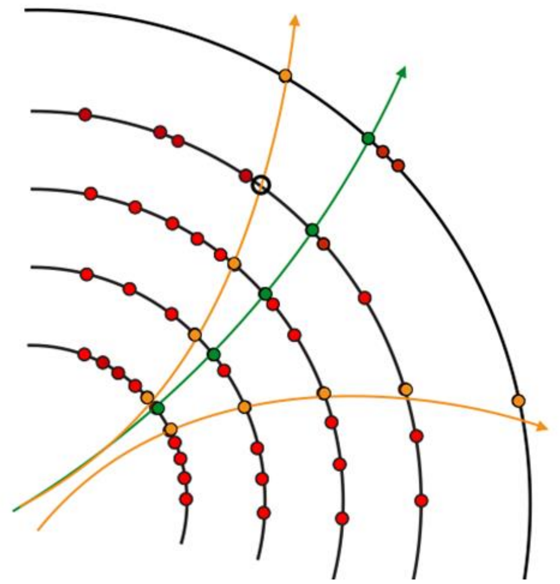
The iterations stop when there is no neighbouring cells with equal states. The final cell state is equal to the length of the segment sequence which can be traced to the left starting from this segment

• <u>Backward pass</u>: Find the cell with the highest state S, then find its neighbour with $S-1$ and so on until state=1. If there are a few neighbours with the same state, take the segment with the smallest breaking angle. The backward pass is repeated the next best cell/state until all the track candidates are collected

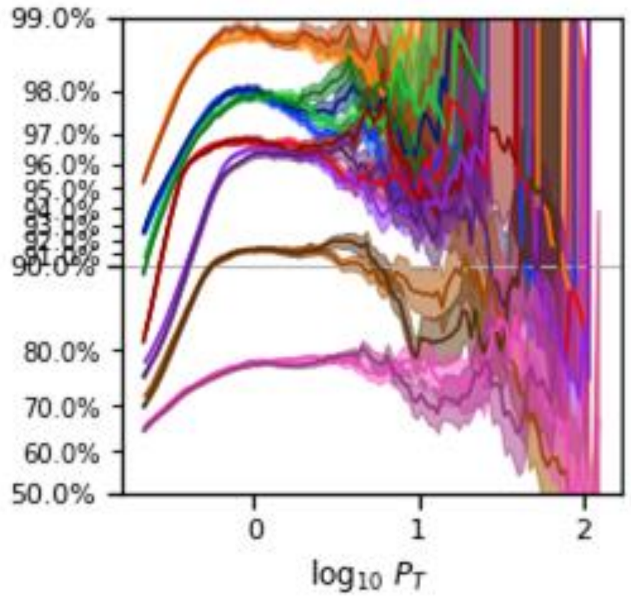
Track following on a graph

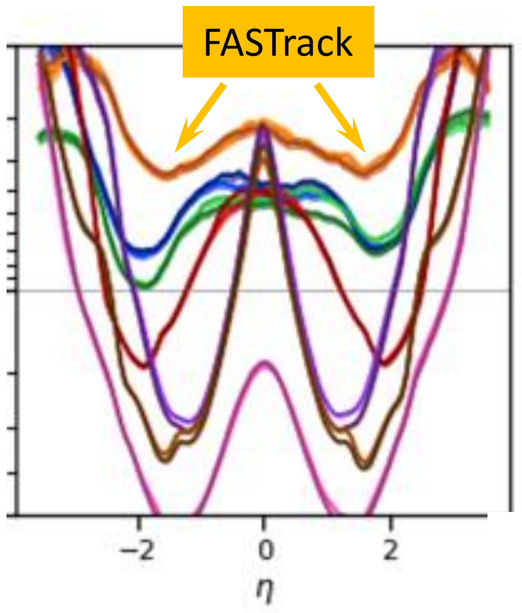
- The idea is to embed Kalman filter in the backward pass of the CA :
  - “Filter and Automaton for Silicon Tracking” : FASTrack
  - it’s a track following which operates on connected track segments rather than individual hits
  - works with “seeds” which are guaranteed to lead to good track candidates thus saving CPU time
- The filter tracks (φ,θ) angles rather than positions using simple dynamics model :
  - random walk for the r-z plane (non-bending), AR(1) model for the r-phi plane (bending) – fitting a trend in φ evolution along a track rotating in the magnetic field



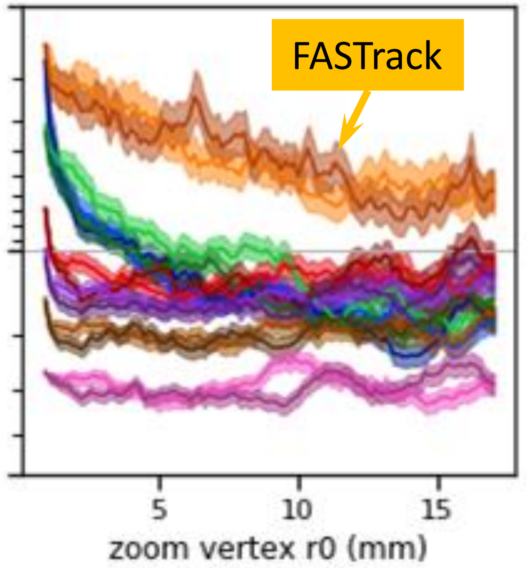


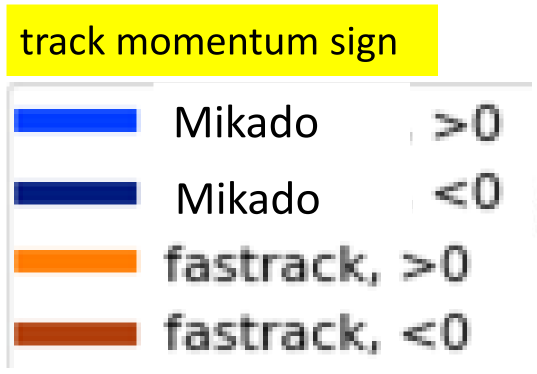
Physics performance : Phase II solutions

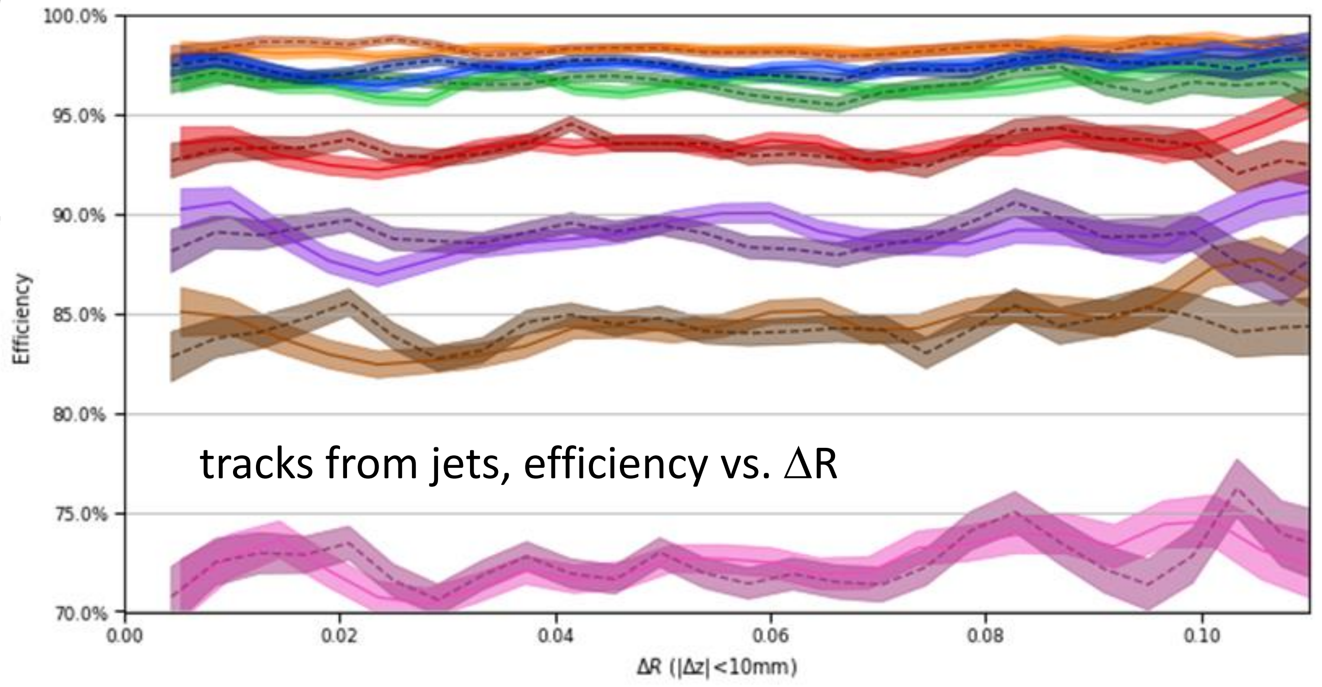




- Track finding efficiency vs. truth track parameters for "+" and "-" charged tracks
- The FASTrack algorithm demonstrated
  - the best track finding efficiency especially for lower momentum tracks difficult to reconstruct
  - stable efficiency in dense environment (jets)







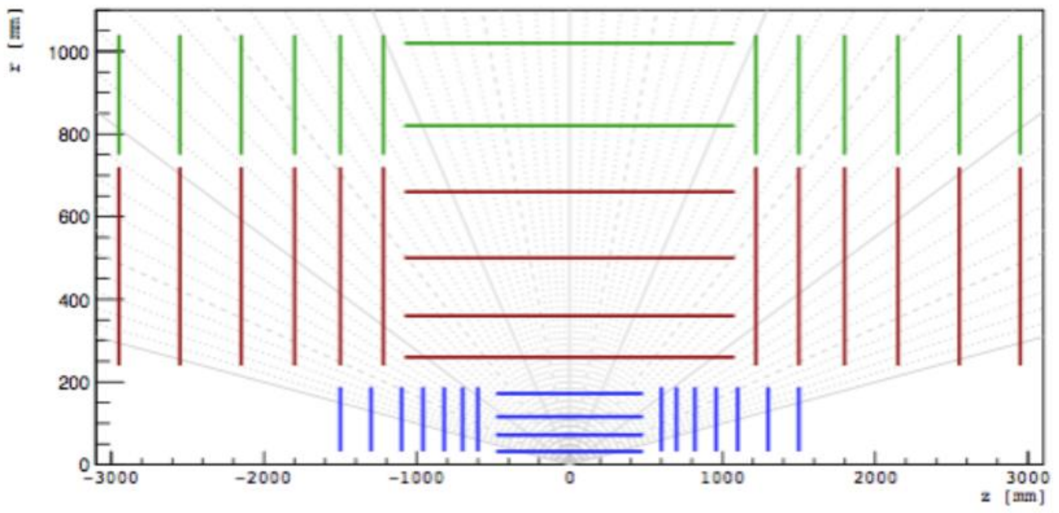
tracks from jets, efficiency vs. ΔR

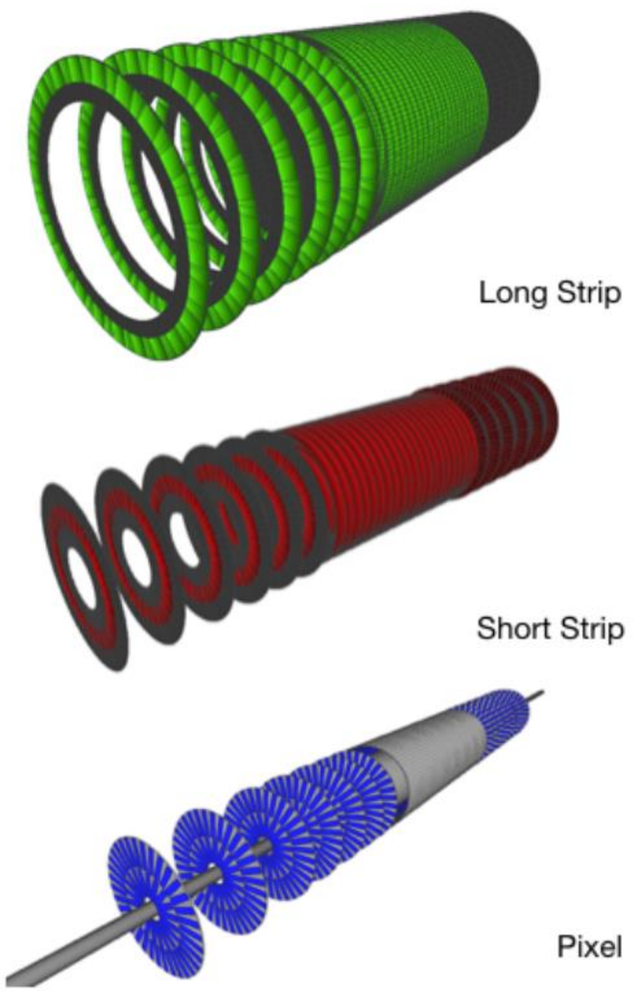
Competition spin-off

- the dataset will be released permanently to serve as a benchmark for future studies:

Reference detector&dataset

Common dataset for development within the community
- detector used for TrackML
- dataset produced with ACTS fast simulation
- **proposal**: iron out the few features we discovered & dataset (LHC/HL-LHC), publish on opendata CERN





Conclusion and outlook

- The TrackML challenge resulted in accurate and superfast (time/event 0.5-1s compared to the state-of-the-art 10-50s) solutions for the track finding problem
- Fast hit pair prediction was important for achieving the best tracking efficiency
- Some of the best solutions employ the multi-stage approach :
  - they disentangle the combinatorial puzzle of data association starting with “easy” tracks and gradually reducing the amount of data while progressing towards more complex cases
  - it more like progressive “explaining data away” approach rather than just tracks labelling
- Ultimate goal for further research:
  - some reasonably fast (e.g. hardware-accelerated) AI-like solutions which could reproduce the above type of approach/reasoning
  - will require systematic efforts (i.e. not just 3 months in summer) and collaboration with AI/ML experts
  - plans for potential R&D collaboration to be discussed at the TrackML Workshop @ CERN in July
    - https://indico.cern.ch/event/813759

Thank you very much !