LAMBDA

# Search for long-lived particles

Andrey Ustyuzhanin, Denis Derkach, Fedor Ratnikov, Alexander Panin

YSDA, HSE

Deep Machine Intelligence Workshop

4th June, 2016

*"I THINK IT'D BE COOL IF ONE DAY AN AI WAS INVOLVED IN FINDING A NEW PARTICLE."*

Demis Hassabis

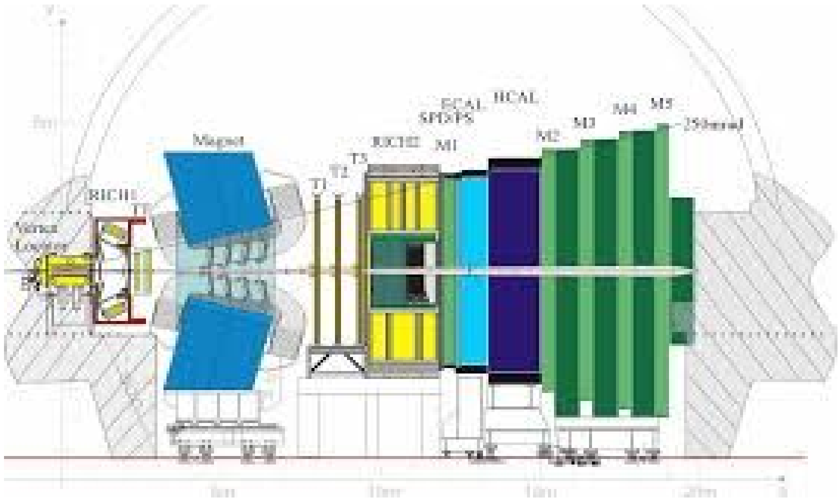
http://bit.ly/1sVqIMF

# Large Hadron Collider and the LHCb

The idea of LHC – accelerate particles (e.g. protons) to insane speeds and watch them collide.

velocity ~10^9 km/h

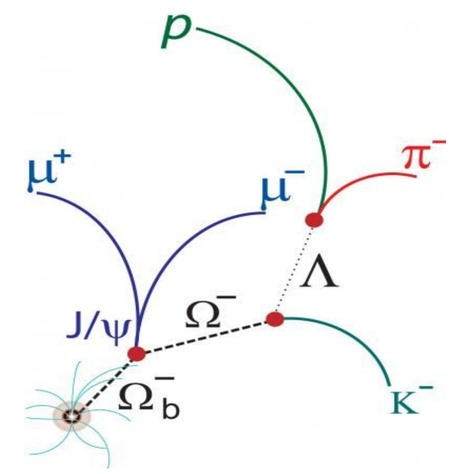
~10^7 events (collisions) per second




We're never low on insane numbers :)

# Particles and decays

Particles:




- Protons(p), Kaons(K), Pions(π) others alike
- Charge: {-1,0,+1}. denoted as $\pi^+$, $\pi$ , $\pi^-$
- Life cycle: get born, fly, decay into other stuff.

Decays:

- A nonsuspecting particle flies along, than **snap!** And you got several other particles
- Denoted as <before> → <after>
- e.g. $K \rightarrow \pi\pi$ - Kaon decay into 2 pions
- $B_d \rightarrow J/\psi\, K^0_S$, $J/\psi \rightarrow \mu\mu$, $\varphi \rightarrow KK$, $D^{*+} \rightarrow D^0 \pi^+$, $D^0 \rightarrow K^0_S \pi\pi$
- Piece'o'cake, right? wrong

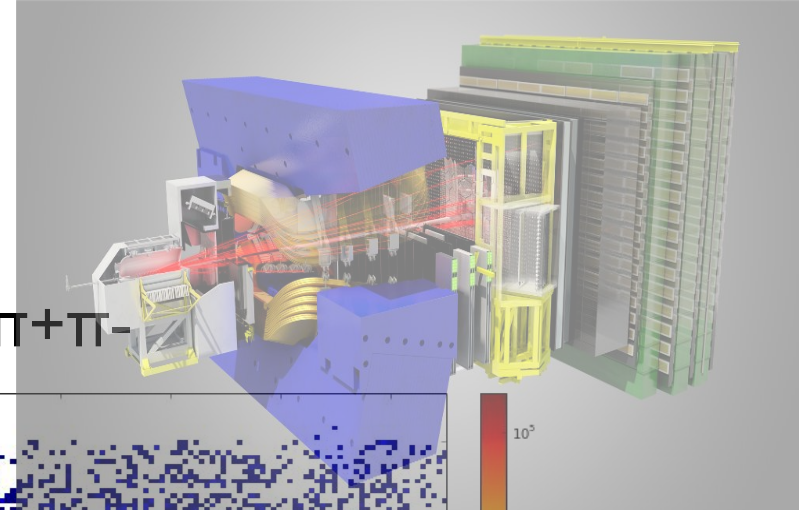
# Which decays do we want and why?

$K^0_S$ (K short, kaons) – leave almost no trace; tend fly a bit and decay into something more visible. Example of 'long lived' particles:

- $K^0_S \rightarrow \pi^+ \pi^-$ (69.2%) : most frequent decay
- $K^0_S \rightarrow 2\pi^0$ (30.69%) : almost untraceable
- $K^0_S \rightarrow$ other ( ~0.11%): too rare

Such decays are not interesting per se, but methods developed are quite different from state-of-the-art and can lead to new discoveries (new physics)

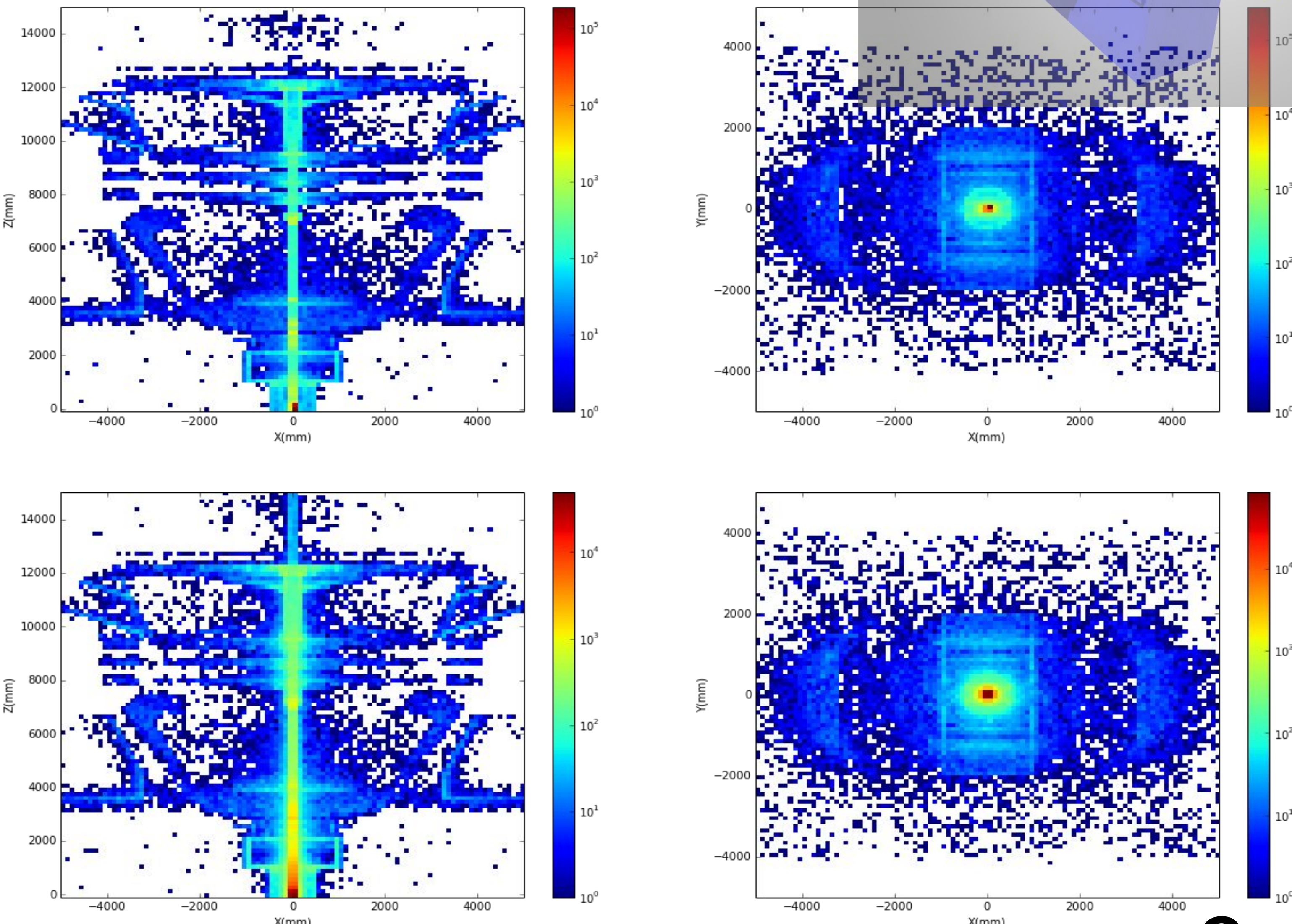
# $K^0_S$ Vertices



- Density of $K^0_S \rightarrow \pi^+\pi^-$

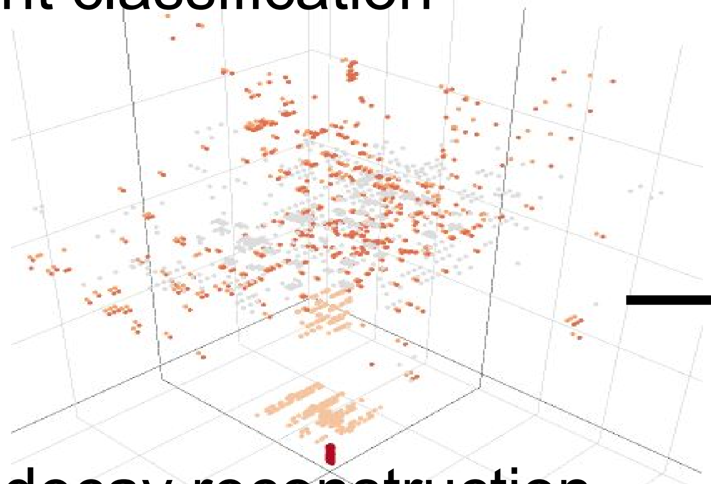
Origin

Decay



# Machine learning perspective

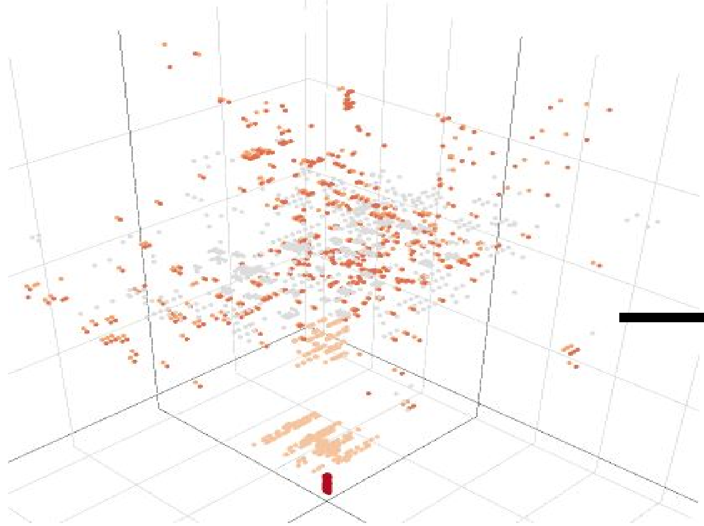
- Event classification



$\{1$ - some $K^0_S$, $0$ - no $K^0_S\}$

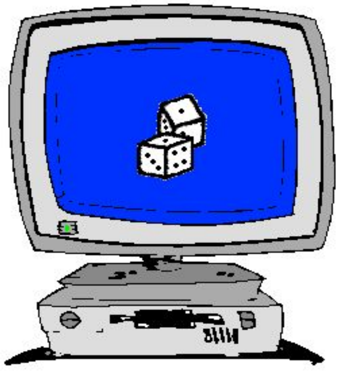
Metric: ROC

- $K^0_S$ decay reconstruction



Relevant $K^0_S$ decays :
$x, y, z, p_x, p_y, p_z$
Metric: cross-entroy

# Monte-carlo



Idea:

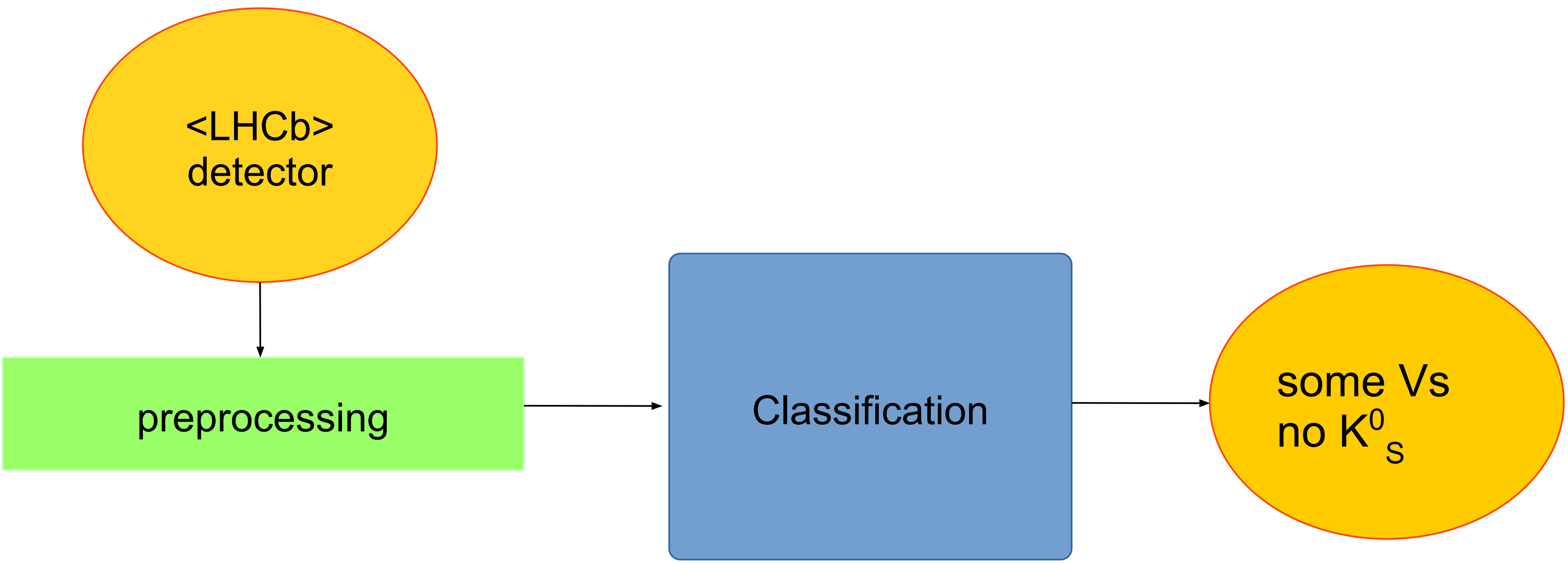
- We cannot explicitly label real world events
- But we want out algorithms to handle them

- Let’s build a stochastic event simulation and train from it
- Than use statistical tests to indirectly estimate classifier performance on real data

# State-of-the art approaches

- Template Matching + RANSAC
- Hough transform + track finding
- The Denby-Peterson method
- The Elastic arms method
- Kalman Filter as Local Method

# Classification problem

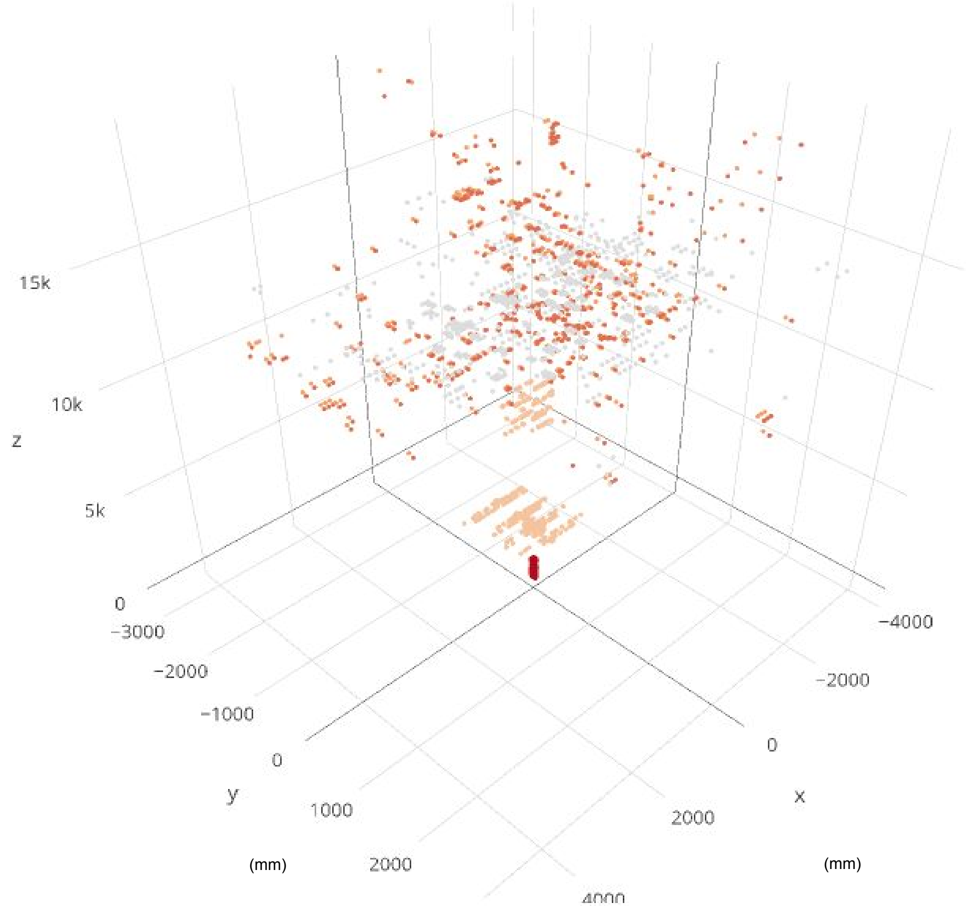


# Preprocessing

- Several data types
  - "hits"/points (e.g. Muon)
  - Tubes (e.g. OT)
  - Hit pairs (e.g. VELO)

- Variable amount of hits per event

- Sparse 3D representation



# Representing Hits

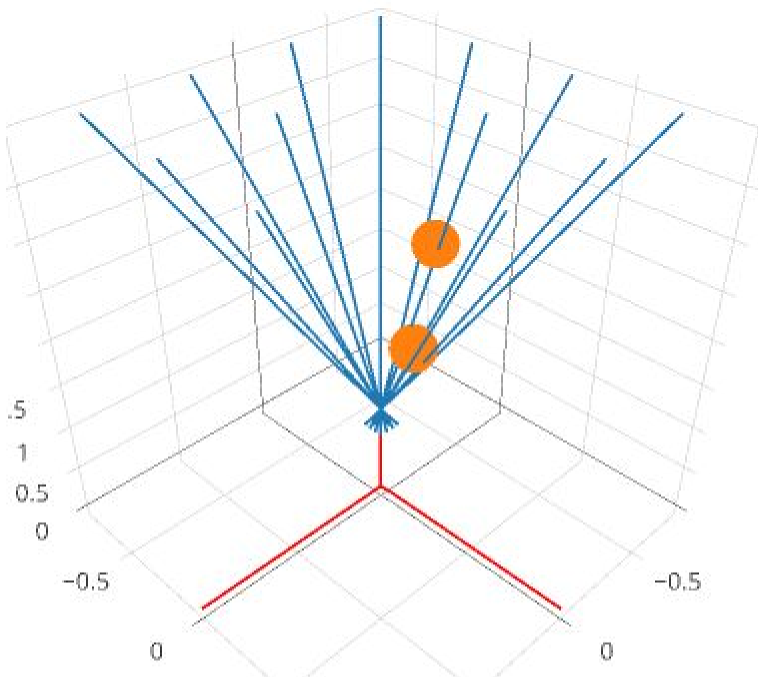
- An natural solution is to use linear template grid (a.k.a. Retina)
  - For each track in a grid, compute its activation given hits
  - Activation of each unit is defined as

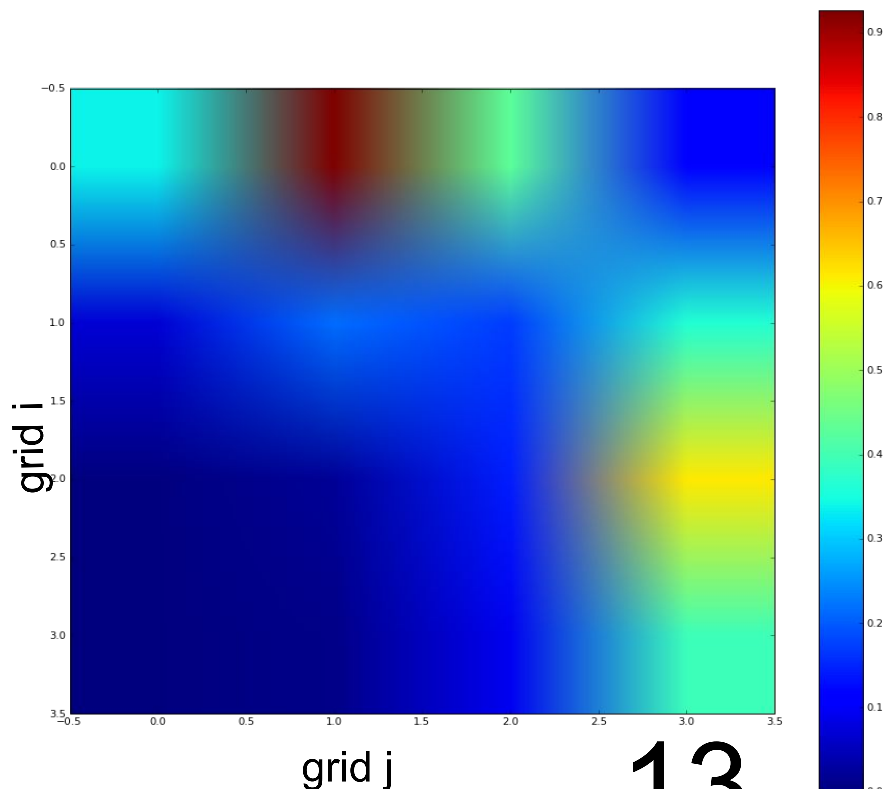
$$h_{line_j} = \sum_{hit_i} e^{\frac{-dist(hit_i, line_j)^2}{\sigma^2}}$$

  - Where dist(hit,line) is a geometric distance, σ is a parameter
  - Set of 3D hits → 2d \`image\` [matrix] of fixed size

# Representating Hits

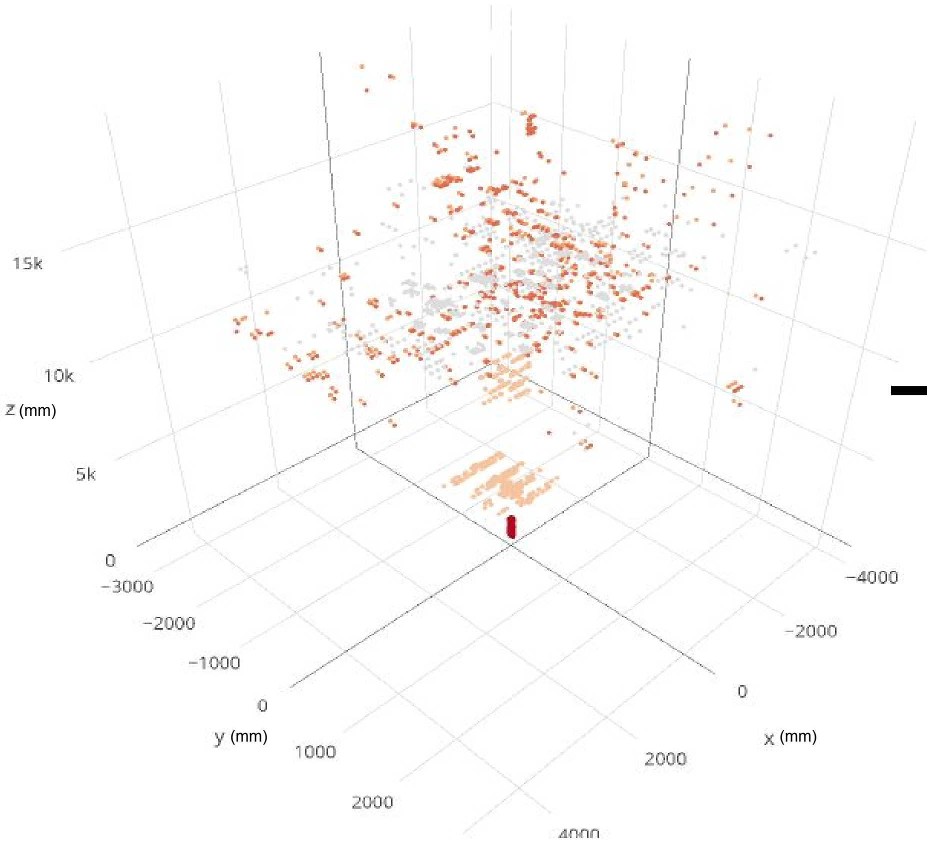
- A natural solution is to use linear template grid (a.k.a. Retina)
  - Grid parameters:
    - anchor point $(x_0,y_0,z_0)$,
    - 2 spheric angles $(alpha_0,beta_0)$
  - Lines are distributed uniformly in spheric space within a certain range around reference angles

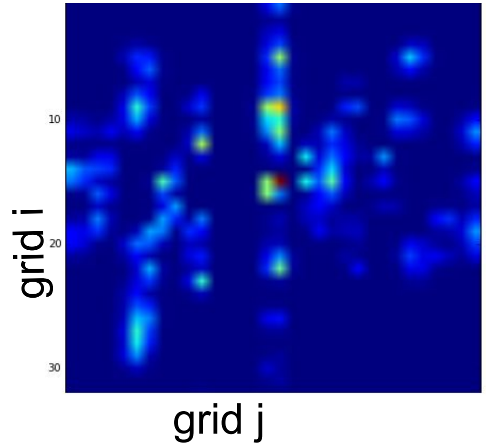


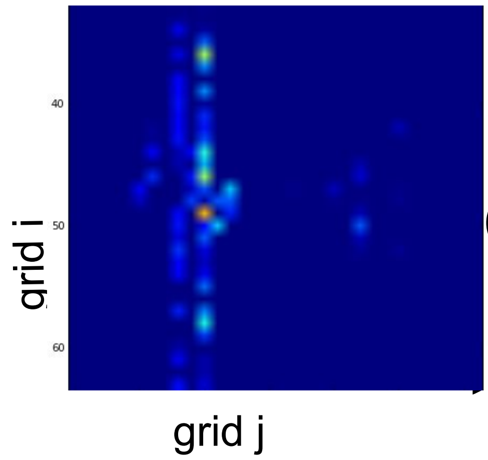
# Practical Implementation

- Grid size: 32x32 lines per image
- In this study, we use several such representations to extract more information about the event



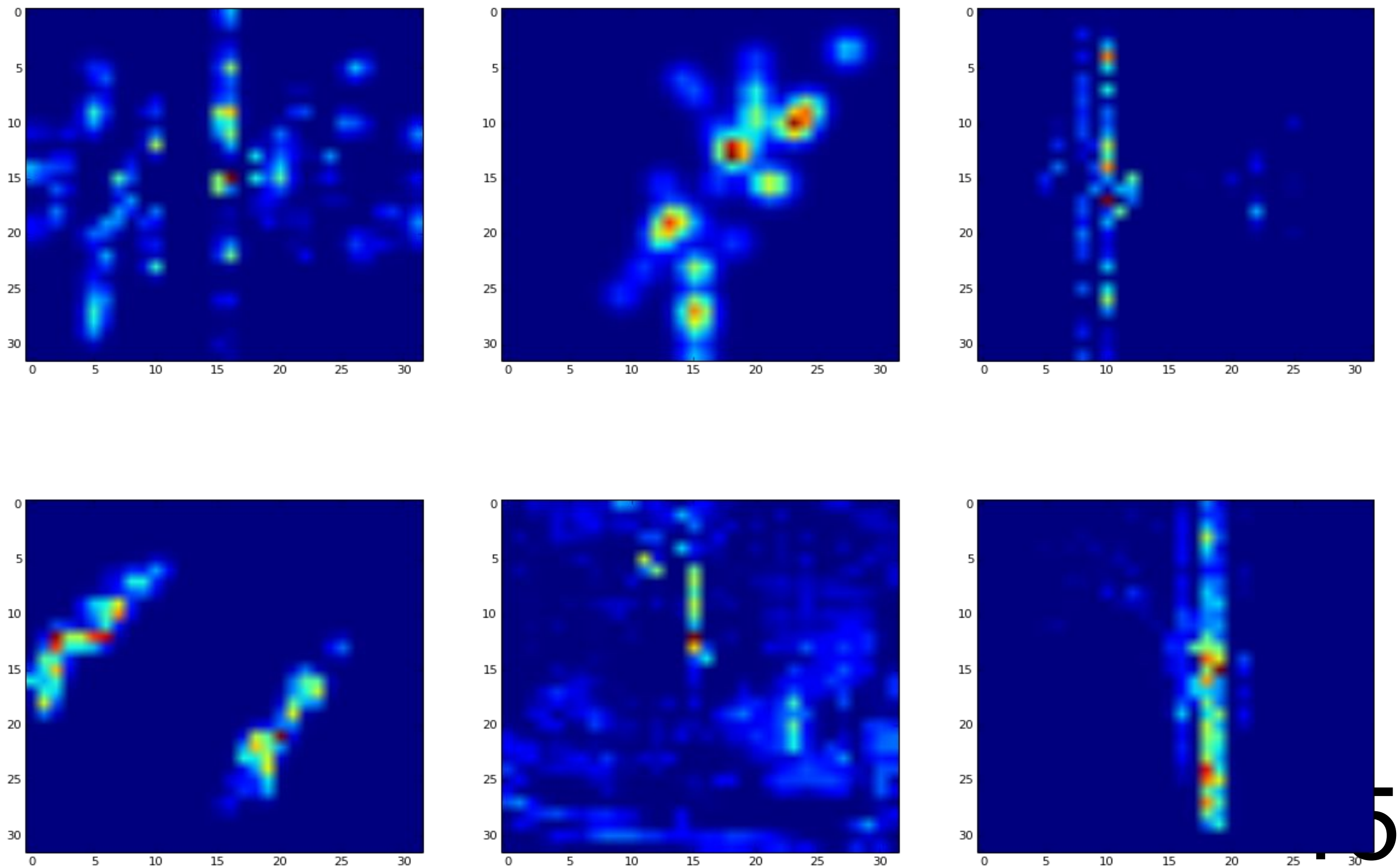

Retina 1
(near VELO)




Retina 2
(under HCAL)

# Practical Implementation

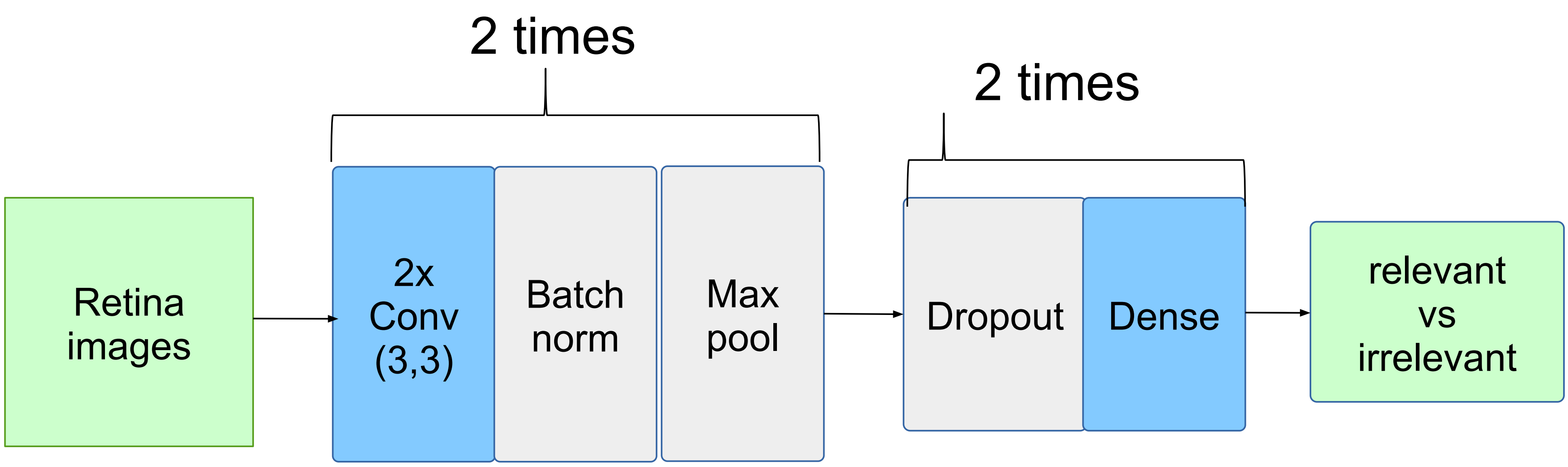
- Prototyping with 3~6 most relevant images
- 32 images used in the best-scoring model
  - Most of them probably irrelevant



# Classifier architecture

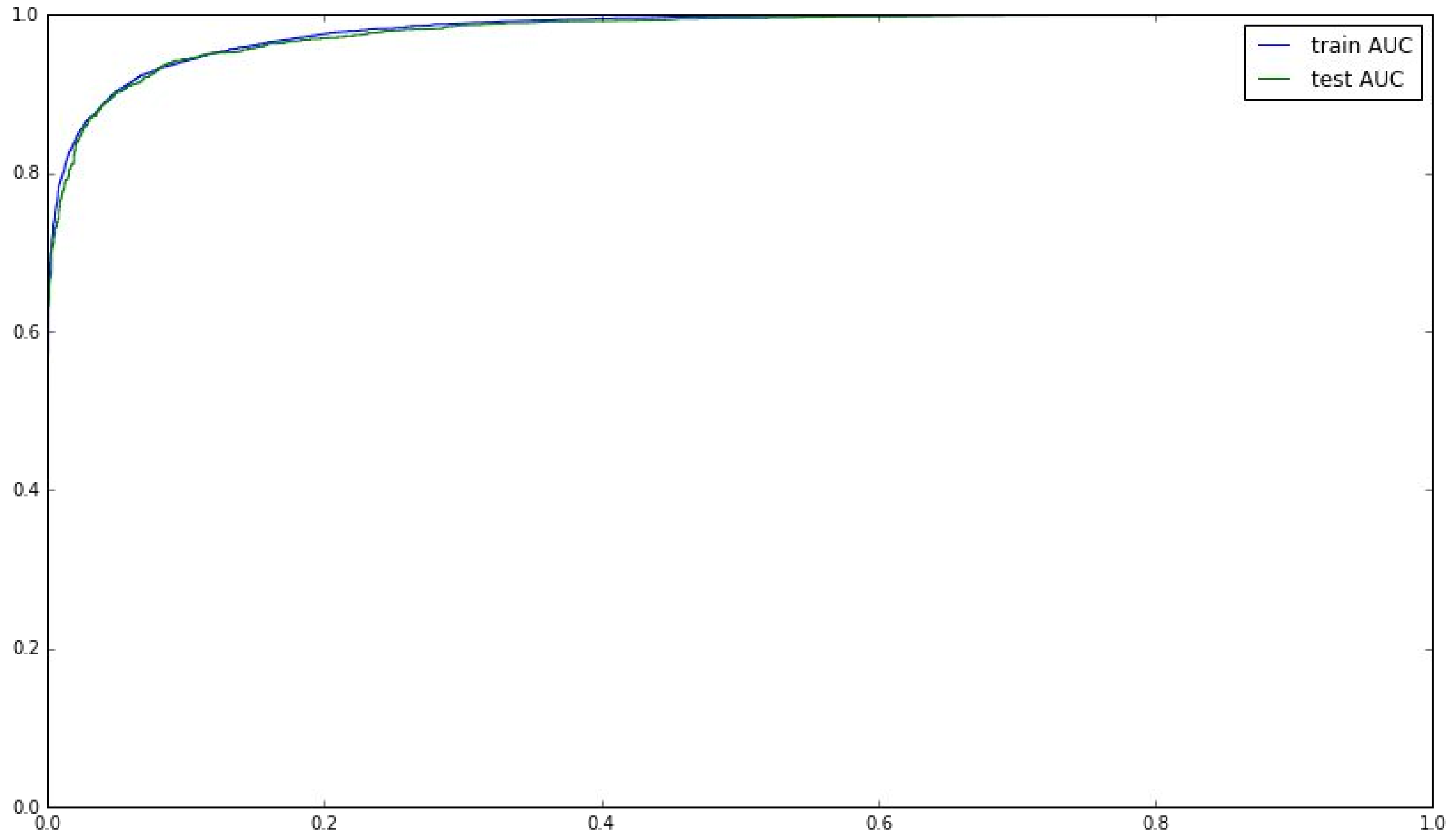
We've tried a lot of different architectures, below is the best performing one *among small architectures*

- even reinforcement learning (POMDP trick)



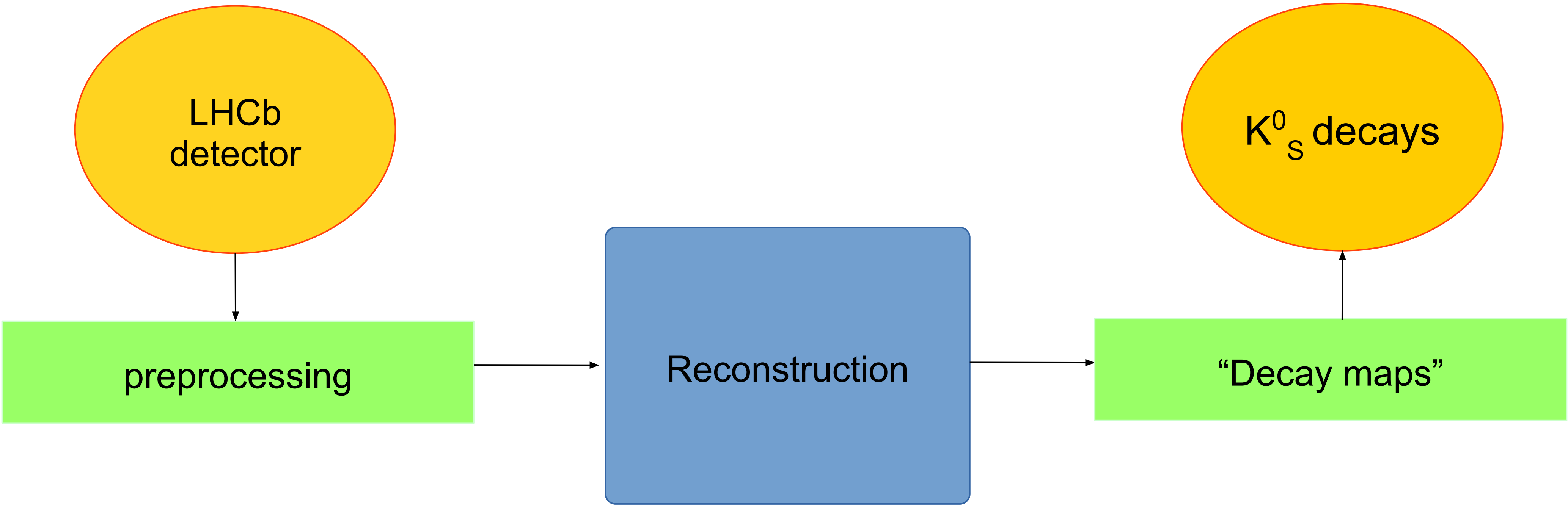
# Results: $K^0_S$ Detection

- ROC-curves




https://github.com/yandexdataschool/KSfinder

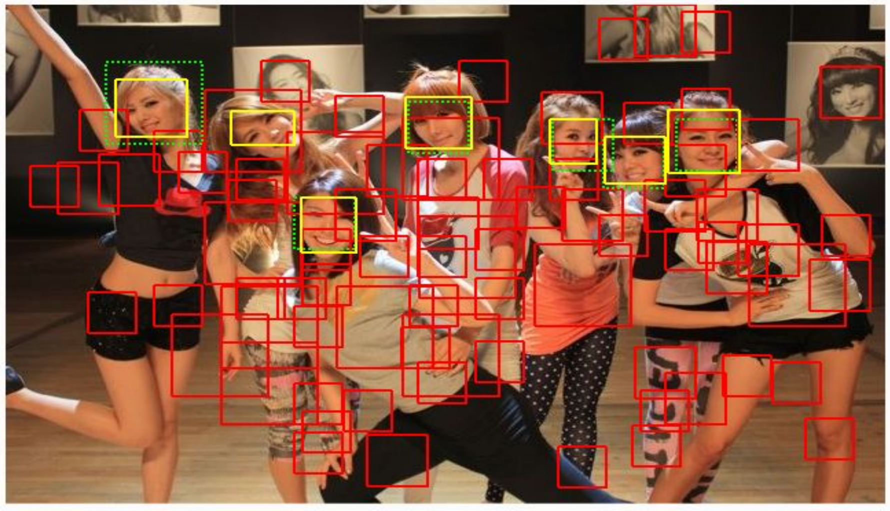
# $K^0_S$ decay reconstruction



# Detection Grid

Detection grid:

- Idea: throw a bunch of points on target space
  - In our case, target space is decay coordinate space
- Compute likelihood of each of them being the correct answer
- (+) Can handle arbitrary amount of targets (decays) per event
- (+) Can use classification techniques
- (-) Precision depends on the amount of points:
  - Scales exponentially with problem dimensionality
- (-) Computationally heavy



random distractive image

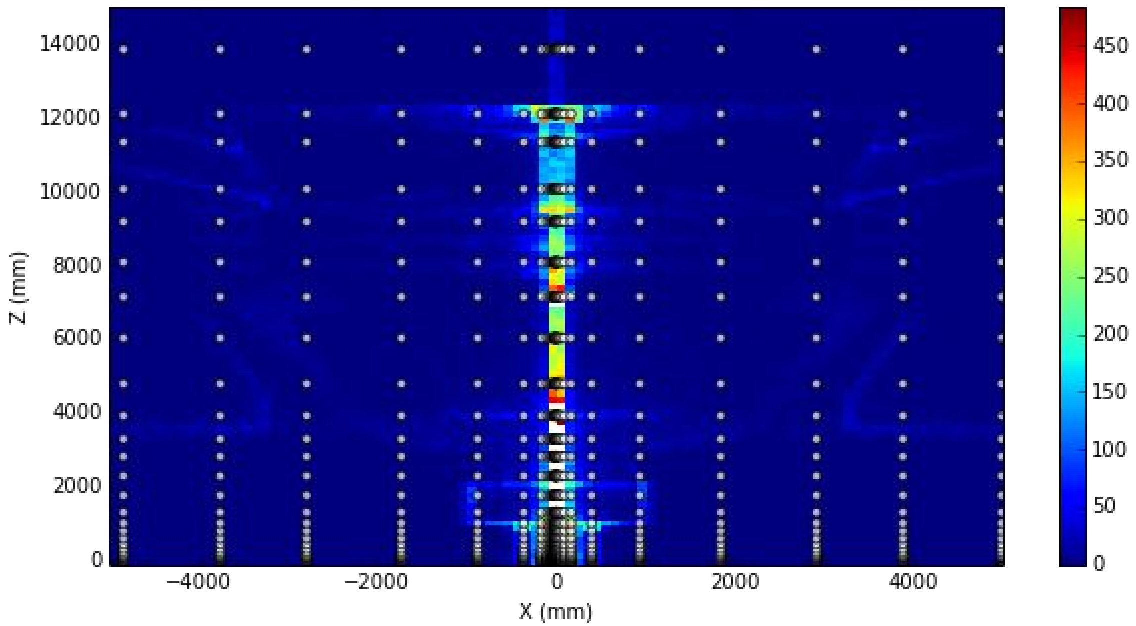
# Representing $K^0_S$ Vertices

What didn't work:

- Regression setup for coordinates
- Standard (uniform) detection grid

What does work 'so far'

- Percentile space detection grid
- Less complex: 1D quantile-space grids for each axis
  - Independent grids for x, y, z, Px, Py, Pz



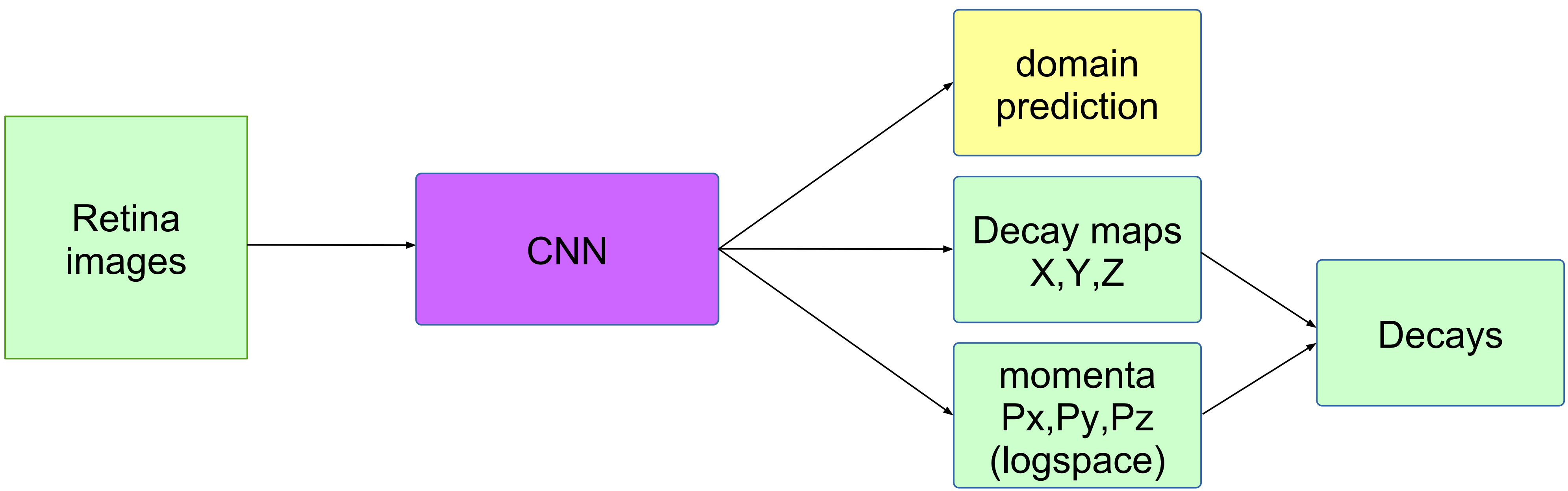

Percentile grid nodes for X/Z decay coordinates

# ML $K^0_S$ Reconstruction

- Objective: pointwise binary cross-entropy across the grid

  L = H(target, prediction)

- Cross-domain regularizer H(domain,predicted domain)
  MC vs real data



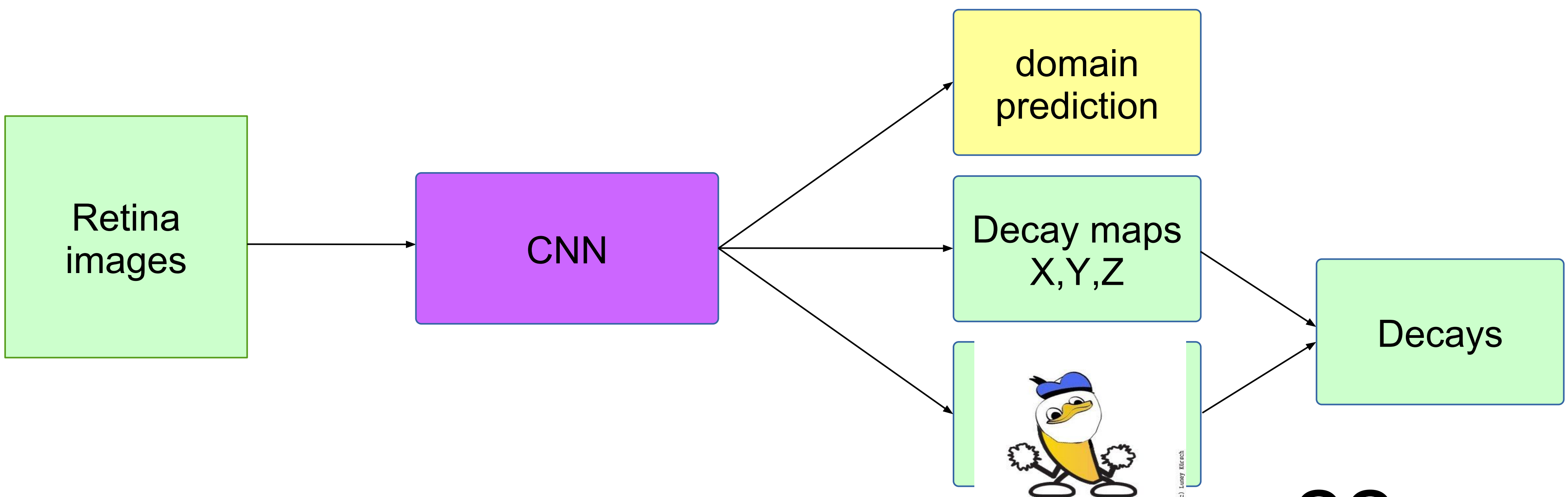

Network architecture, similar to classifier one

# ML $K^0_S$ Reconstruction

- Objective: pointwise binary cross-entropy across the grid

  L = H(target, prediction)

- Cross-domain regularizer H(domain,predicted domain)
  MC vs real data

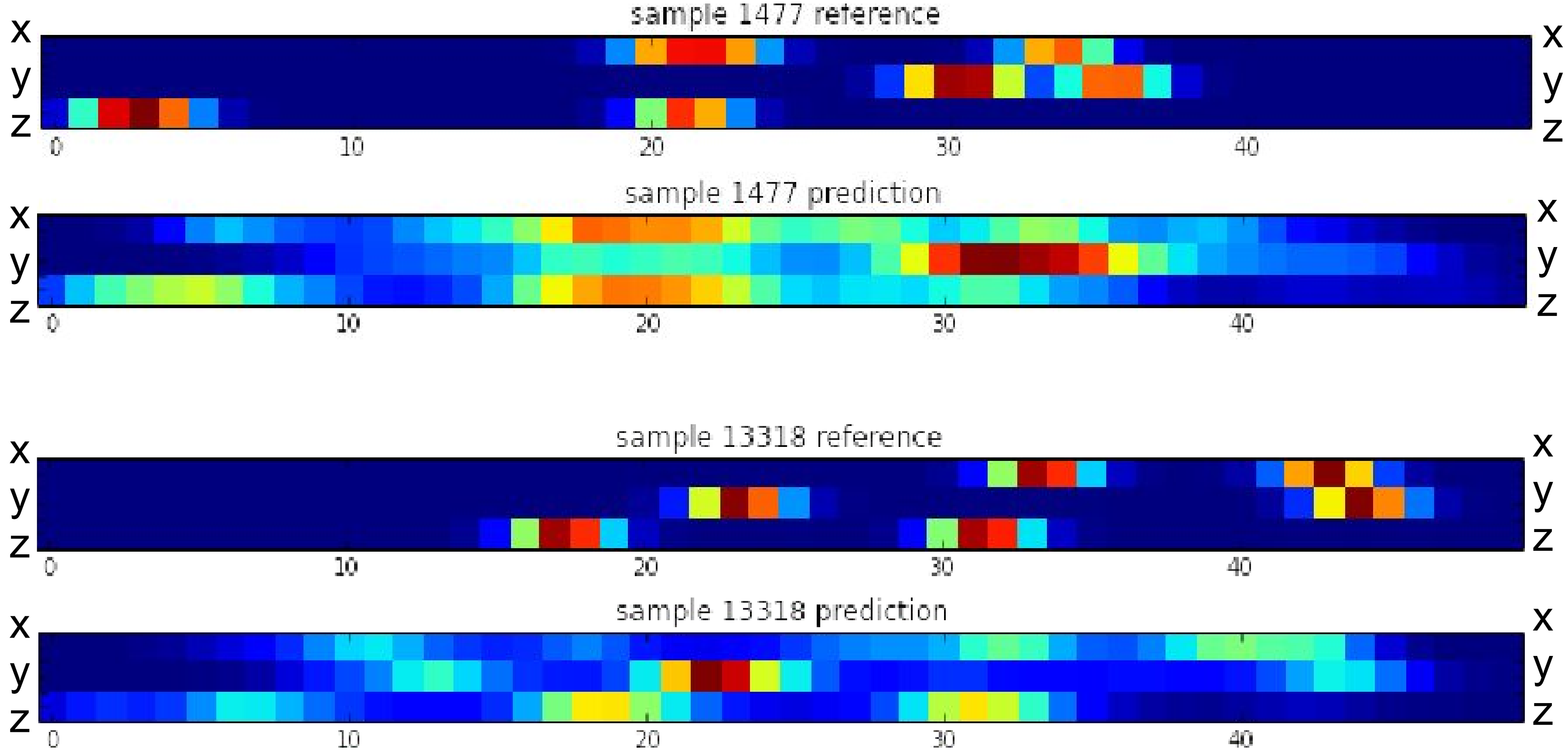



Network architecture, similar to classifier one

# Result Evaluation

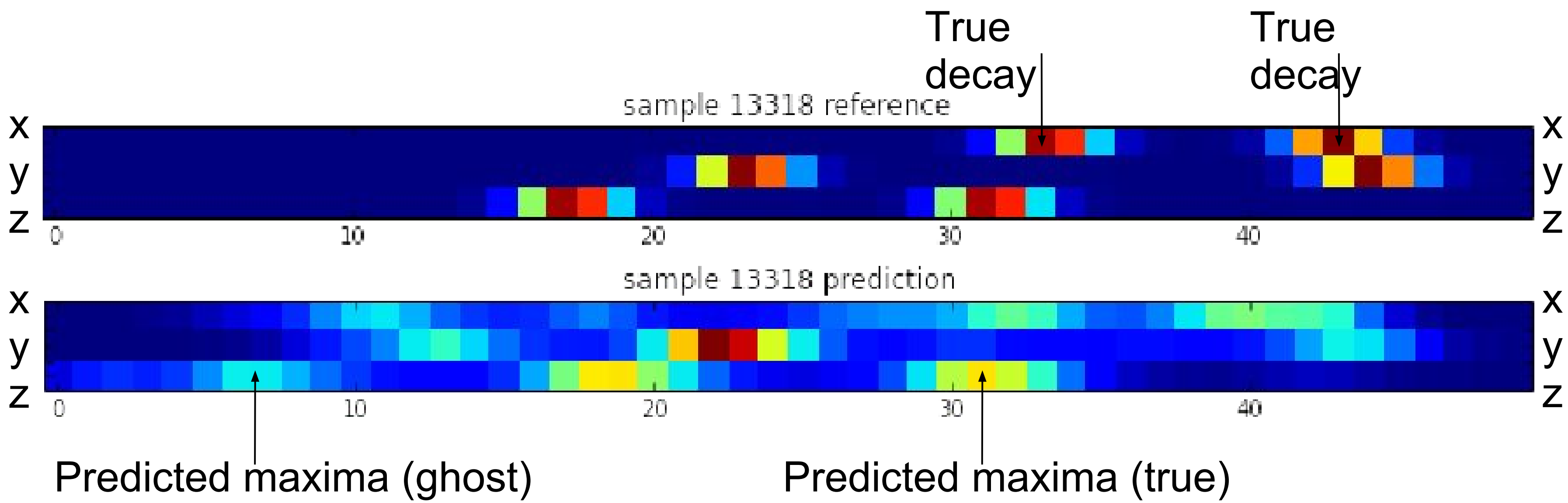
- $K^0_S$ decay maps : reference and prediction

Each map corresponds to a single event



# Result Evaluation

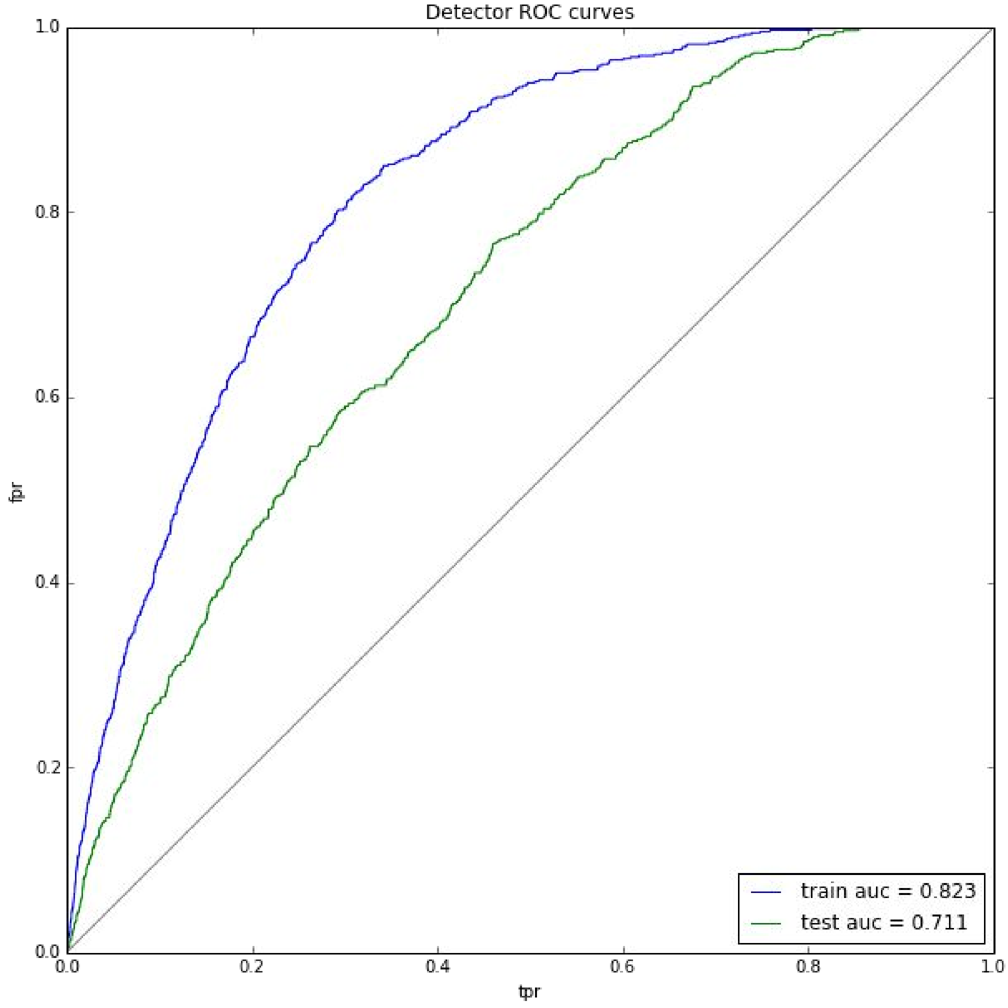
- A predicted decay is qualified as true prediction of the real decay if it corresponds to the same or neighboring cell in the detection grid.
- A ghost (false alarm) is a predicted maxima with no neighboring true decays






**All the metrics are tied to the detection grid bins**

# Results: $K^0_S$ Reconstruction



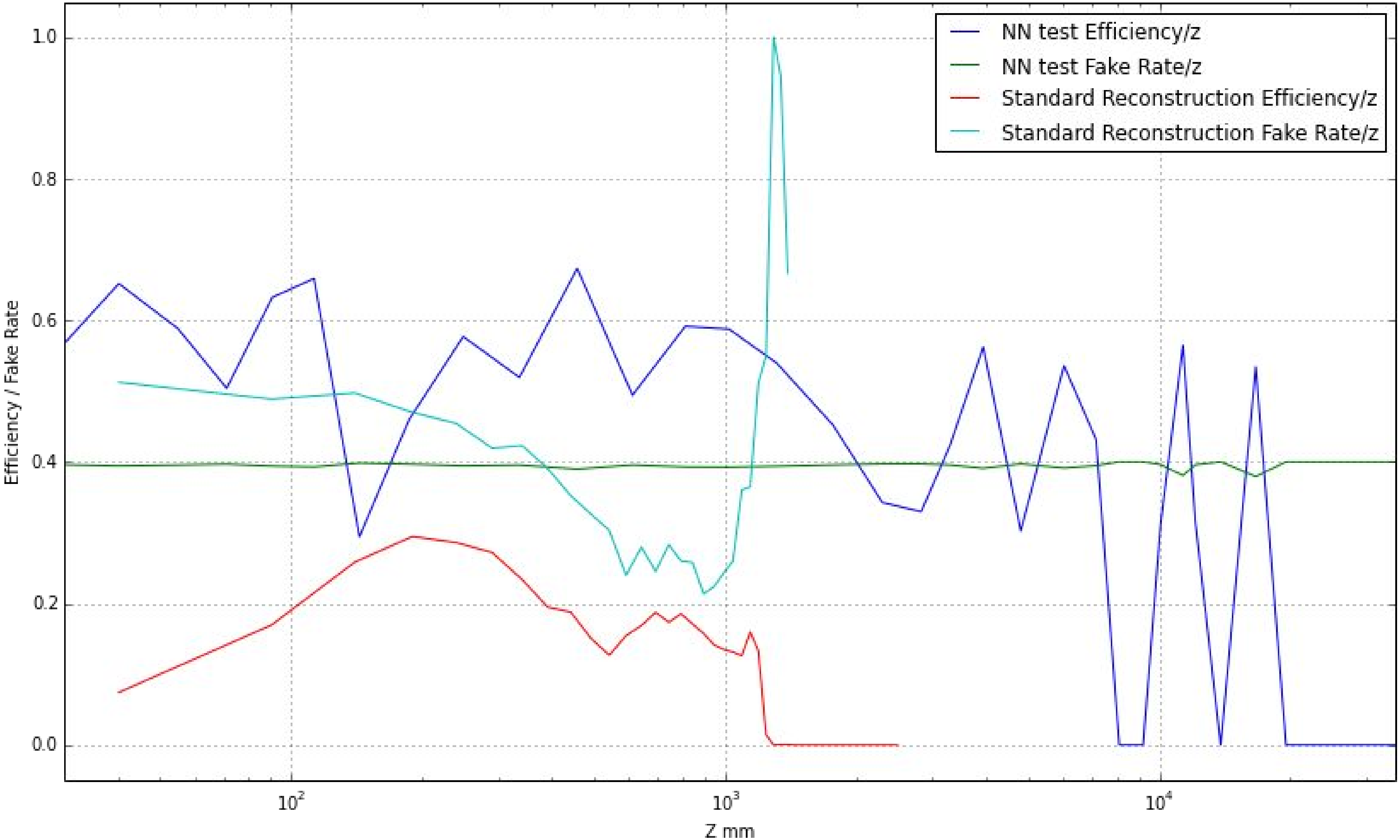

Overall ROC plots

# Comparing With Baseline RECO

$$Efficiency = \frac{N(Reconstructed\,True\,K_s^0)}{N(Reconstructed\,True\,K_s^0) + N(Unrecognized\,True\,K_s^0)}$$

$$FakeRate = \frac{N(Reconstructed\,Fake\,K_s^0)}{N(Reconstructed\,Fake\,K_s^0) + N(Reconstructed\,True\,K_s^0)}$$

# Efficiency/FakeRate vs Z of the Decay




Efficiency extends beyond Z=2m

# Instead of conclusion

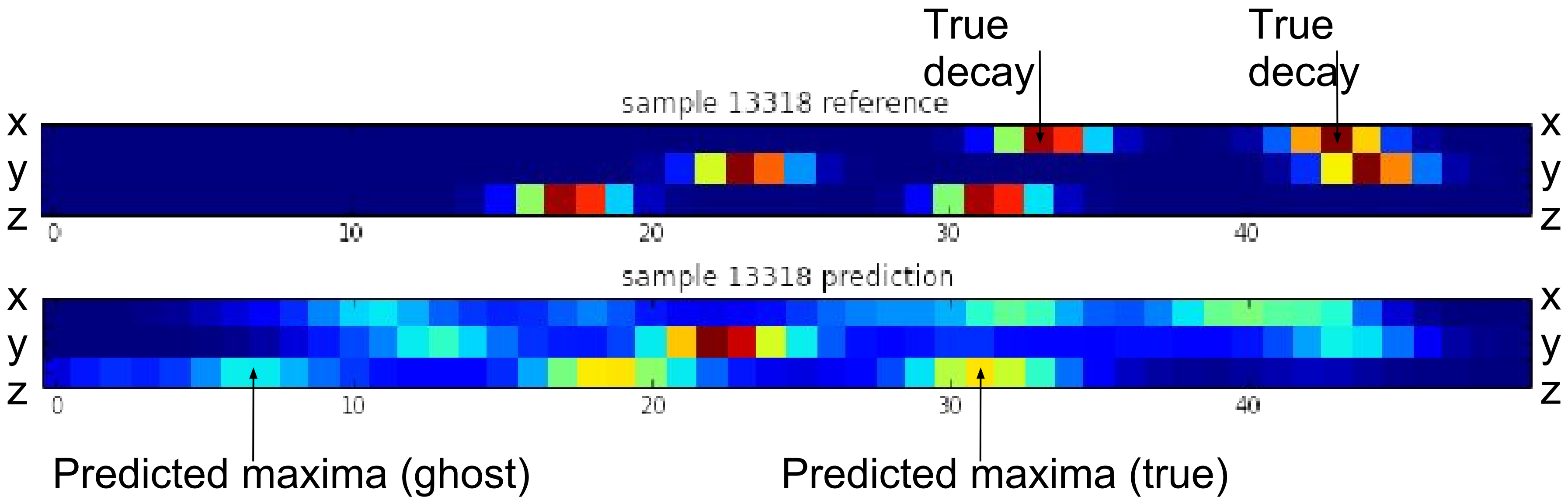
- New kind of problem(s)
  - decay identification
  - decay reconstruction
- Relatively accurate decay classification
  - can be proxy for the reconstruction
- Minimalistic (baseline) classifier:
  http://bit.ly/1sh5jxE
- Decay reconstruction(x, y, z, impulse) is still a major challenge
- Toolkit for deep reinforcement learning - AgentNet (see next talk)

# Thank you!

anaderi@yandex-team.ru

# Interpretation

- Predicted decays are maxima of the detection grid
- Prediction likelihoods are the values predicted at the maxima




**All the metrics are tied to the detection grid bins**

# Decay reconstruction $K^0_S$

- $K^0_S \rightarrow \pi+\pi-$ decay frequency *per event*