Multiclass classification application: LHCb Particle Identification

Tatiana Likhomanenko

IML meeting, 18 January 2017

Disclaimer

- All results are very preliminary

Challenge

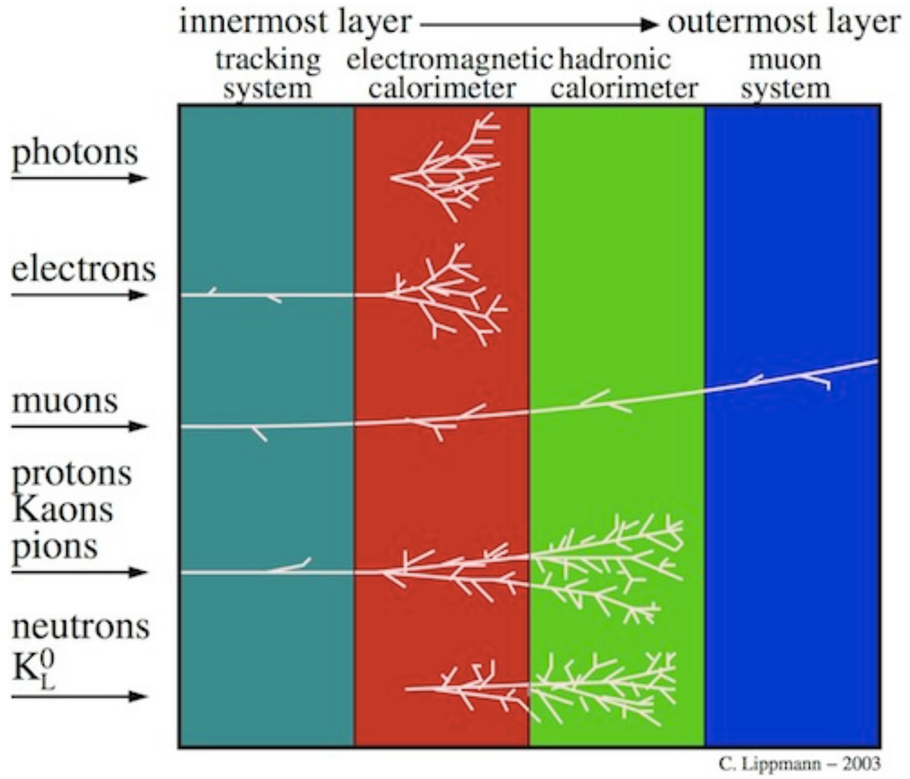
〉 Problem: identify charged particle associated with a track (multiclass classification problem)

〉 particle types: Ghost, Electron, Muon, Pion, Kaon, Proton.

〉 LHCb detector provides diverse plentiful information, collected by subdetectors: CALO, RICH, Muon and Track observables

〉 this information can be efficiently combined using ML

〉 Monte Carlo simulated samples for various decays are available (6 millions tracks in training and 6 millions in test)



Multiclass classification vs/with binary classification

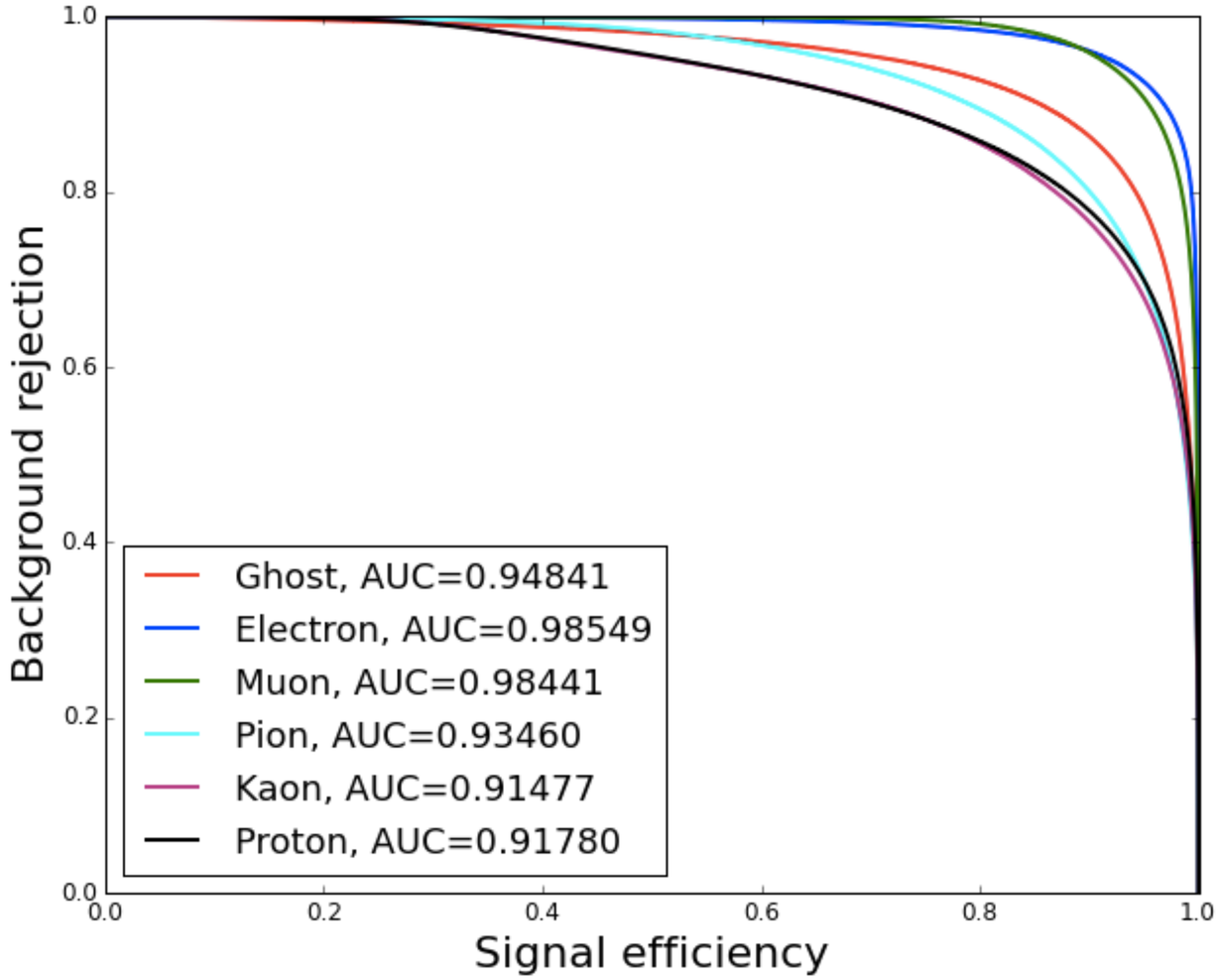
- Binary classification is widespread (major ML use-case in HEP).
- Binary classification is very simple to implement (and some algorithms are defined only for binary classification), that is why ML packages have it.
- Gradient Boosting and Neural Networks supports multiclass classification: XGBoost, sklearn and most neural networks implementations.
- Multiclass classification can be reduced (when multiclass mode is not available) to binary classification with two approaches:
  - one versus rest (OvR): for example, distinguish electron from all other particles
  - one versus one (OvO): for example, distinguish electron from muon
  - other approaches exist
- Sklearn supports this out-of-the box: http://scikit-learn.org/stable/modules/multiclass.html

PID

Quality Measures

Total Model Quality

- There are 6 classes, that is why OvR approaches is fine.
- In analyses, we are mostly interested in selecting one type (e.g., muon).
  - We use one-vs-rest ROC curves and area under the curves (AUC) to measure quality of the classification (also for multiclass classification algorithms).

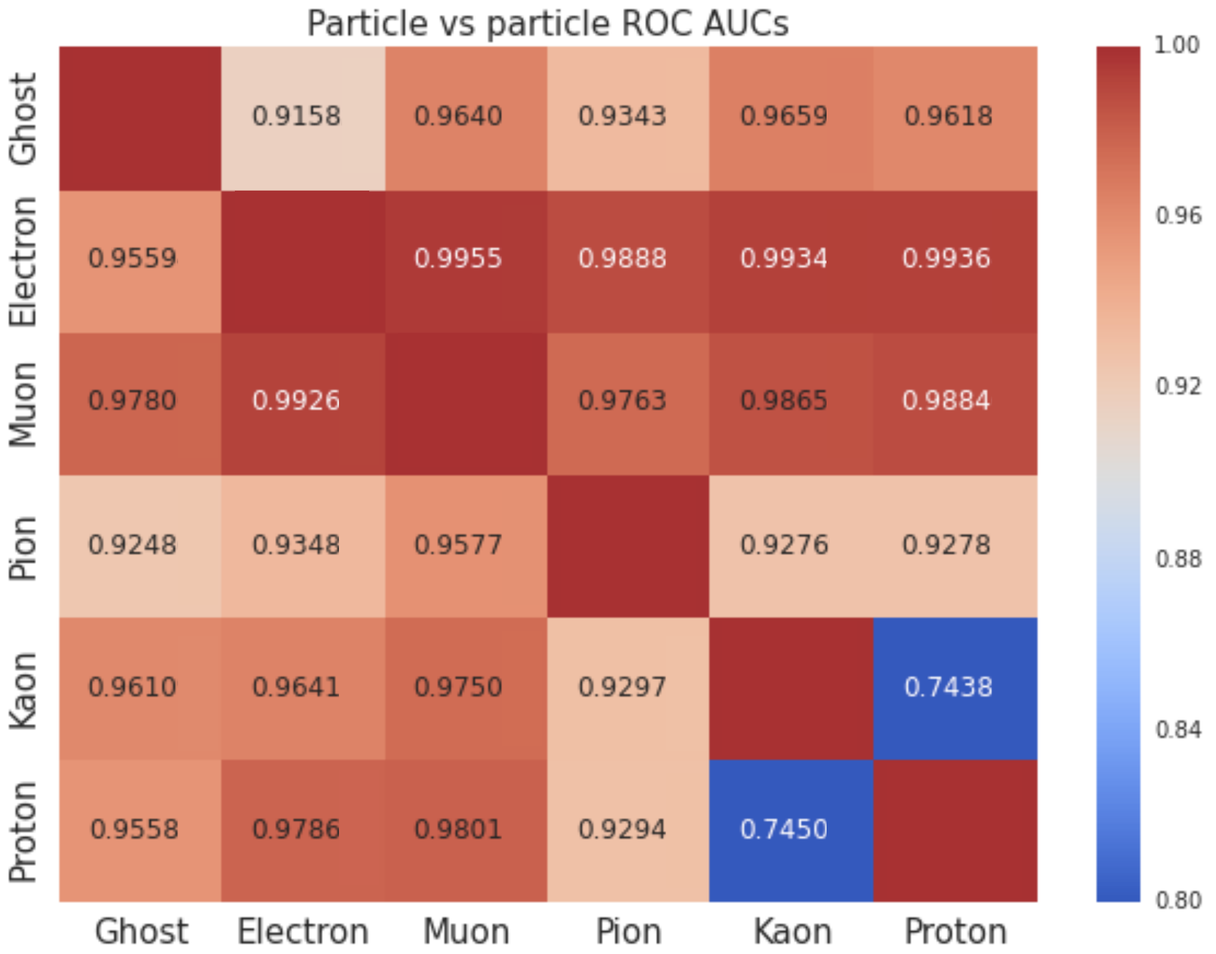


Detailed Model Quality

Sometimes we need detailed information about particle types discrimination.

Use ROC AUC score to compute quality between two classes:

- each block is a ROC AUC value between two particle types
- ROC AUC is area under the ROC curve and represents separation quality for the two particles



Baseline Model

- Simple neural network with one hidden layer (TMVA MLP)
- Consists of 6 binary classification models: one versus rest

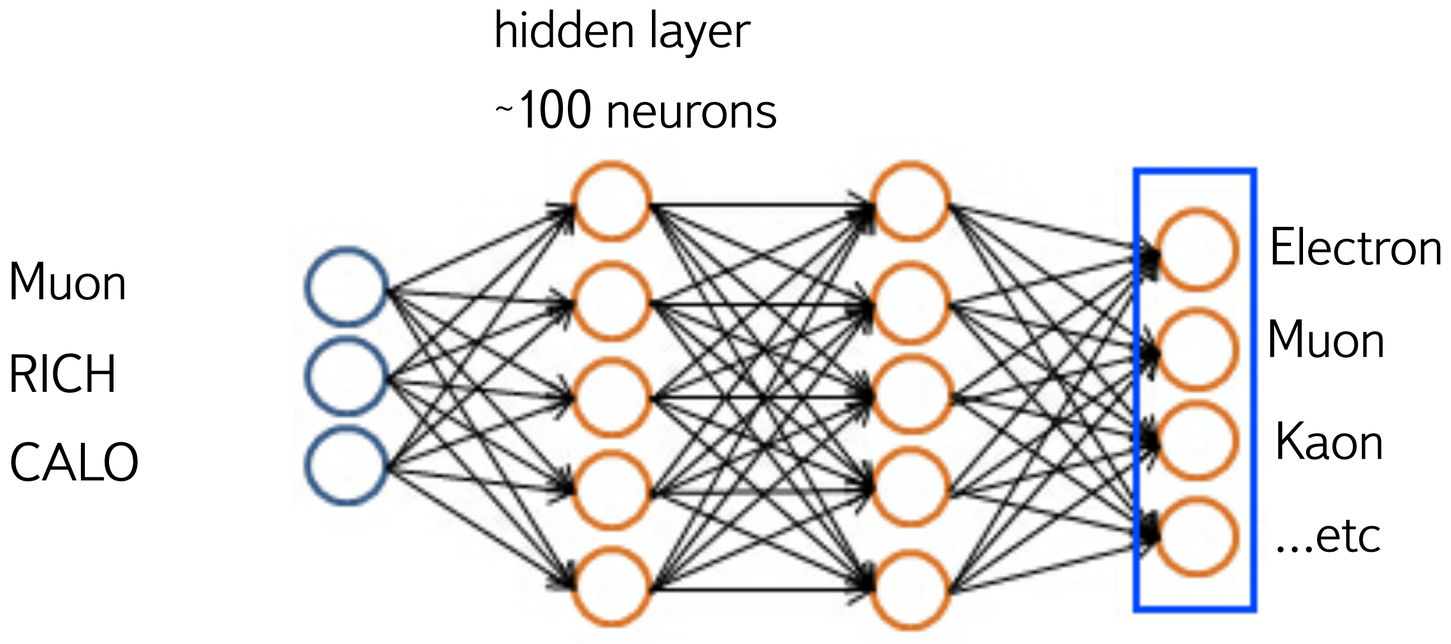
PID

Neural Networks

Multiclass classification for Neural Networks

〉 The number of weights (parameters) in Neural Network is similar for binary classification and multiclass classification

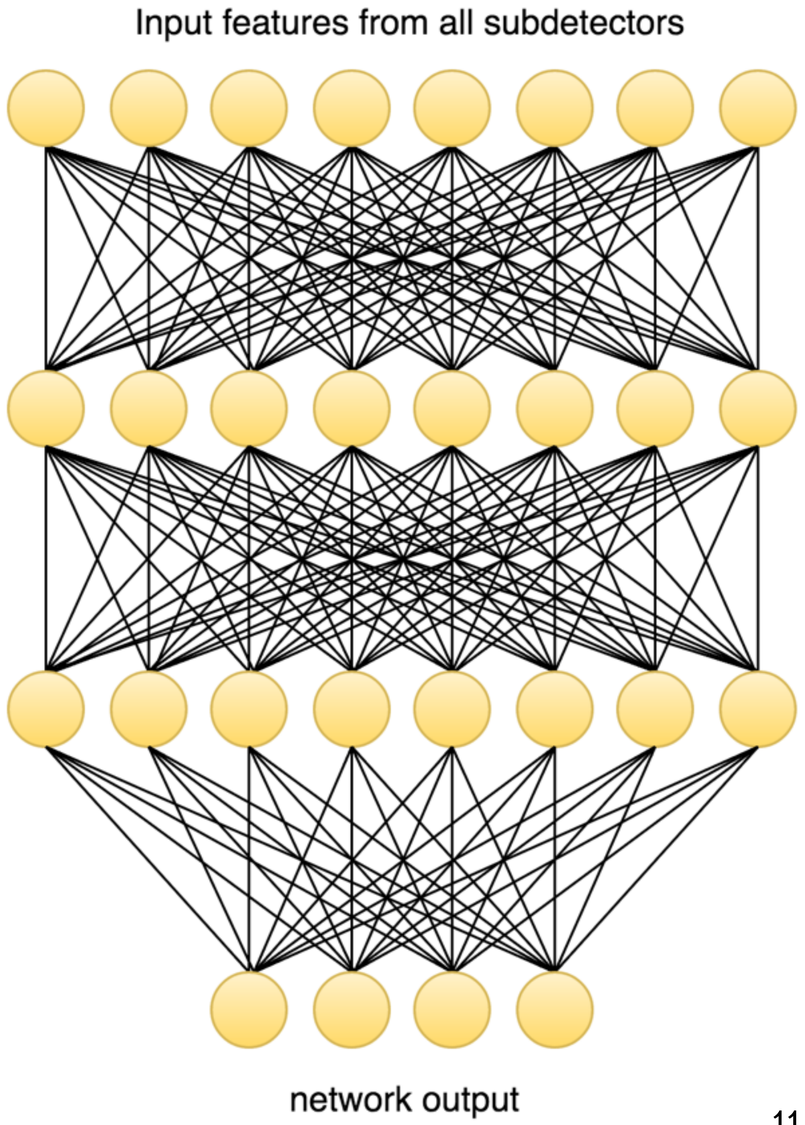
〉 Computationally multiclass classification has (almost) the same complexity as binary classification (unless there are hundreds of classes and more)



PID Neural Network

We use multiclass classification neural networks (NN) from Keras library:

- apply specific preprocessing (flattening of distributions)
- train deep NN with three hidden layers. Common problem for training deep architectures is gradient diminishing. We use well-known approaches to overcome this:
  - ReLU activation function
  - batch normalization
  - adaptive momentum training
  - dropout to prevent coadaptation of neurons



Neural Networks AUCs

		Ghost	Electron	Muon	Pion	Kaon	Proton
one vs rest	**baseline**	0.9484	0.9854	0.9844	0.9345	0.9147	0.9178
multiclass	**keras DL**	0.9632	0.9914	0.9925	0.9587	0.9319	0.9320

Understanding reasons for improvement

- We trained several simple models to investigate this result
- Directly solving multiclass classification problem provides significant improvement

  Multiclass classification approach is able to use the full information (multiclass labels) and provides the global optimization. In other cases the problem is divided into separate optimization problems.

  Multiclass classification has better property: probabilities are summed up to one automatically which can be very useful later. In other approaches this should be enforced.
- Modern DL (deep learning) tricks like adaptive optimization are also important
- Deep structure (more than one layer) gives quite moderate boost in quality :)

Neural Networks with Special Structure

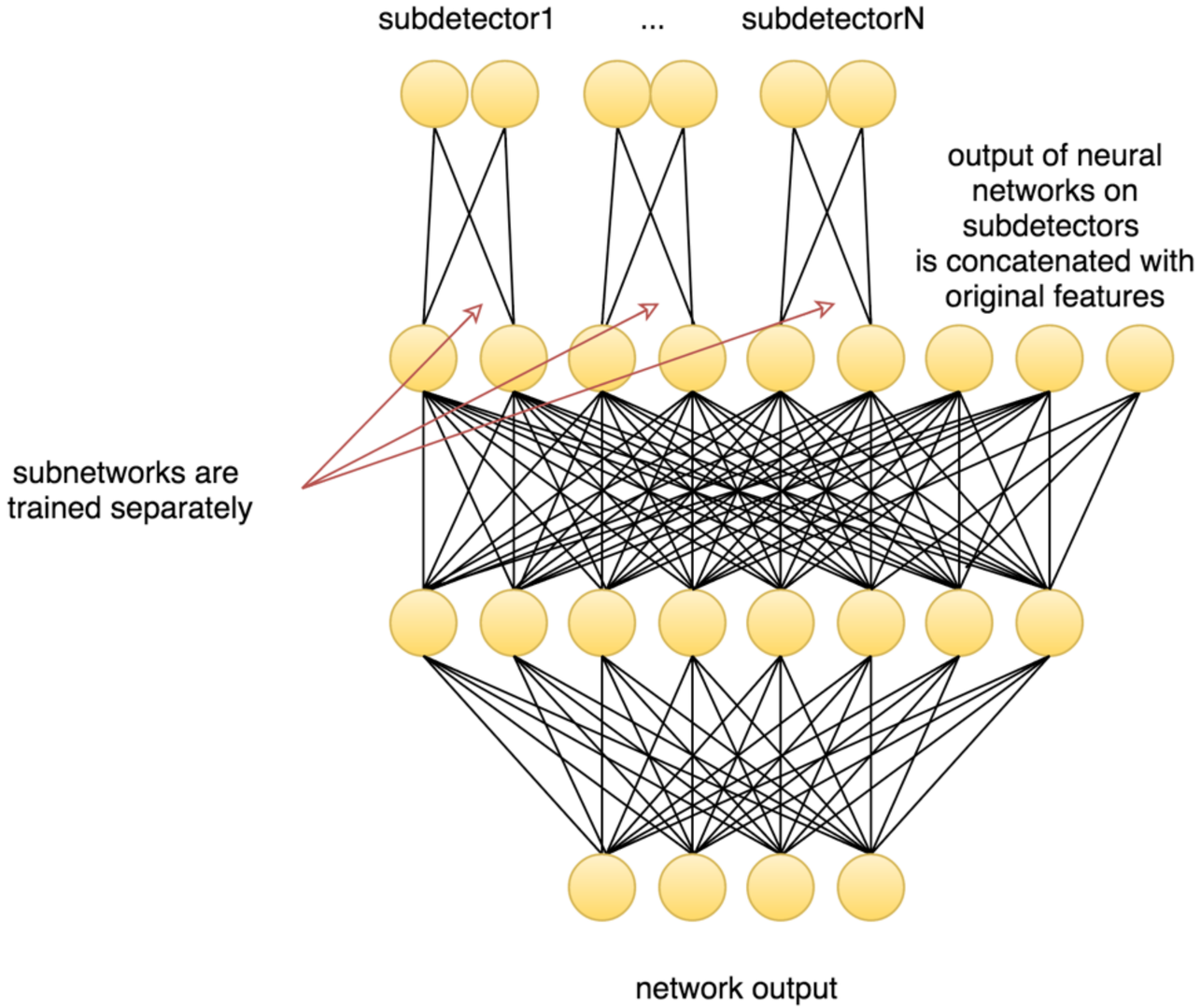
〉 Linear combination of features for a subdetector seems to be informative and possibly can serve as a better feature

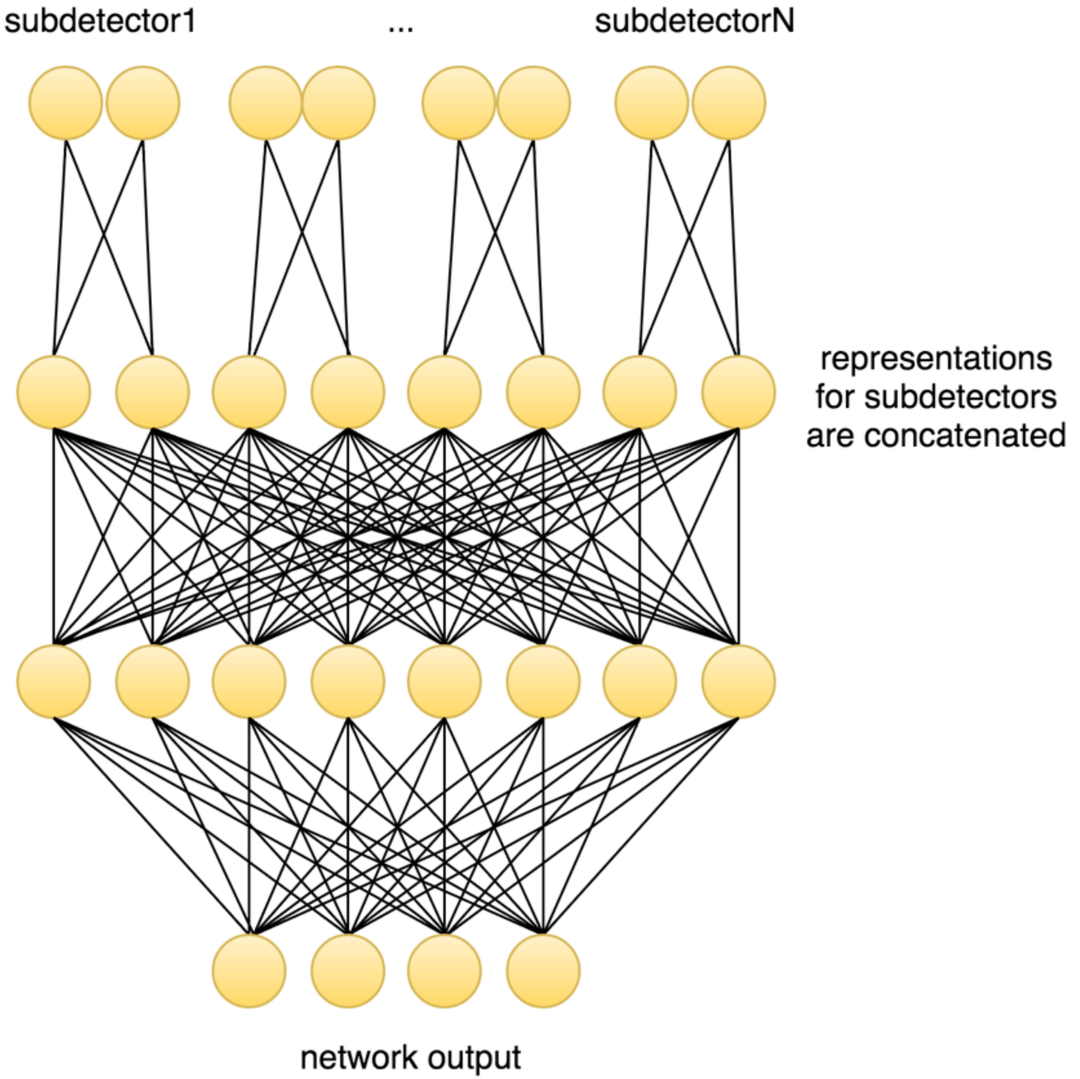
〉 We can optimize these combinations within neural network framework:

  〉 separately train NN on each subdetector features, later use as additional features for another NN (stacking approach)
  〉 initial layers combine information separately for each subdetector. These initial layers are optimized simultaneously with the rest of network.

〉 One can use the following groups from subdetectors: CALO features, RICH features, muon features, track features, others.

Neural Networks: Stacking and Special Structure





Neural Networks AUCs

	Ghost	Electron	Muon	Pion	Kaon	Proton
keras DL	0.9632	0.9914	0.9925	0.9587	0.9319	0.9320
stacked NN	0.9624	0.9911	0.9924	0.9580	0.9316	0.9314
special NN	0.9622	0.9910	0.9923	0.9573	0.9309	0.9307

- When preliminary experiment was performed on the small data, special NN demonstrated better quality than DL.
- When millions of samples are available for DL there is no problem in having many (probably not physically motivated) parameters in network.

PID

Boosted Decision Trees

Boosted Decision Trees

〉 Add linear combinations of initial features which can help to construct trees (NNs can reconstruct those by themselves, BDTs cannot)

〉 Train special BDT: boosted oblivious decision trees algorithm (one versus rest approach)

〉 Train multiclass classification XGBoost

Models AUCs

	Ghost	Electron	Muon	Pion	Kaon	Proton
baseline	0.9484	0.9854	0.9844	0.9345	0.9147	0.9178
keras DL	0.9632	0.9914	0.9925	0.9587	0.9319	0.9320
XGBoost	0.9609	0.9908	0.9922	0.9568	0.9303	0.9302
special BDT	0.9636	0.9913	0.9926	0.9576	0.9309	0.9310

- BDT has similar quality to DL
- Training procedure and prediction time for BDT grows up linearly depending on number of classes

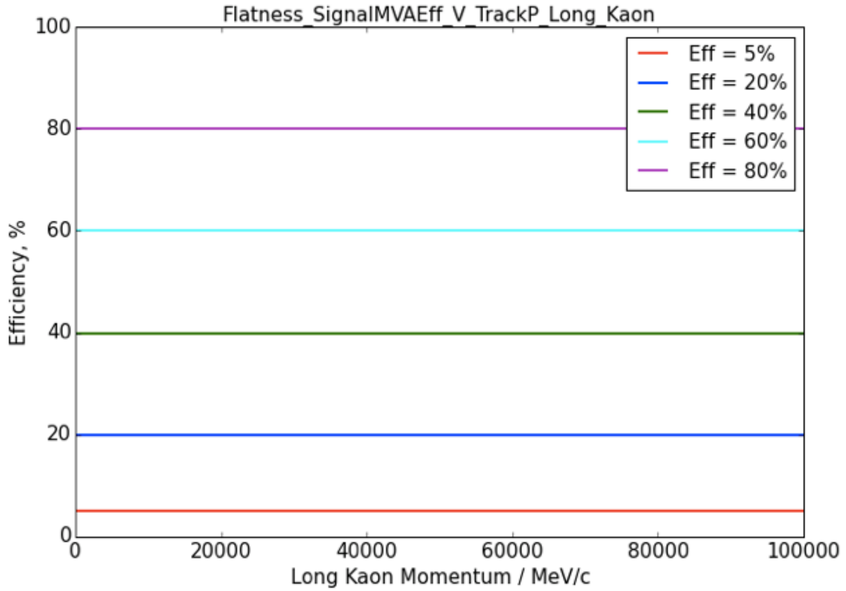
PID

Flat Models

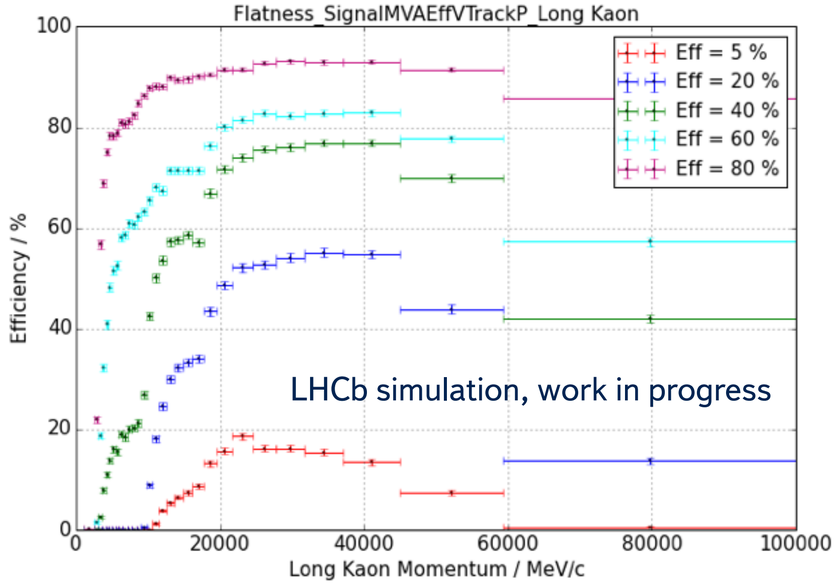
Improving PID with flat models

〉 The whole PID information strongly depends on particle momentum, that leads to strong dependency between PID efficiency and momentum

〉 In some analyses we need to have flat PID along signal particle momentum



Ideal world

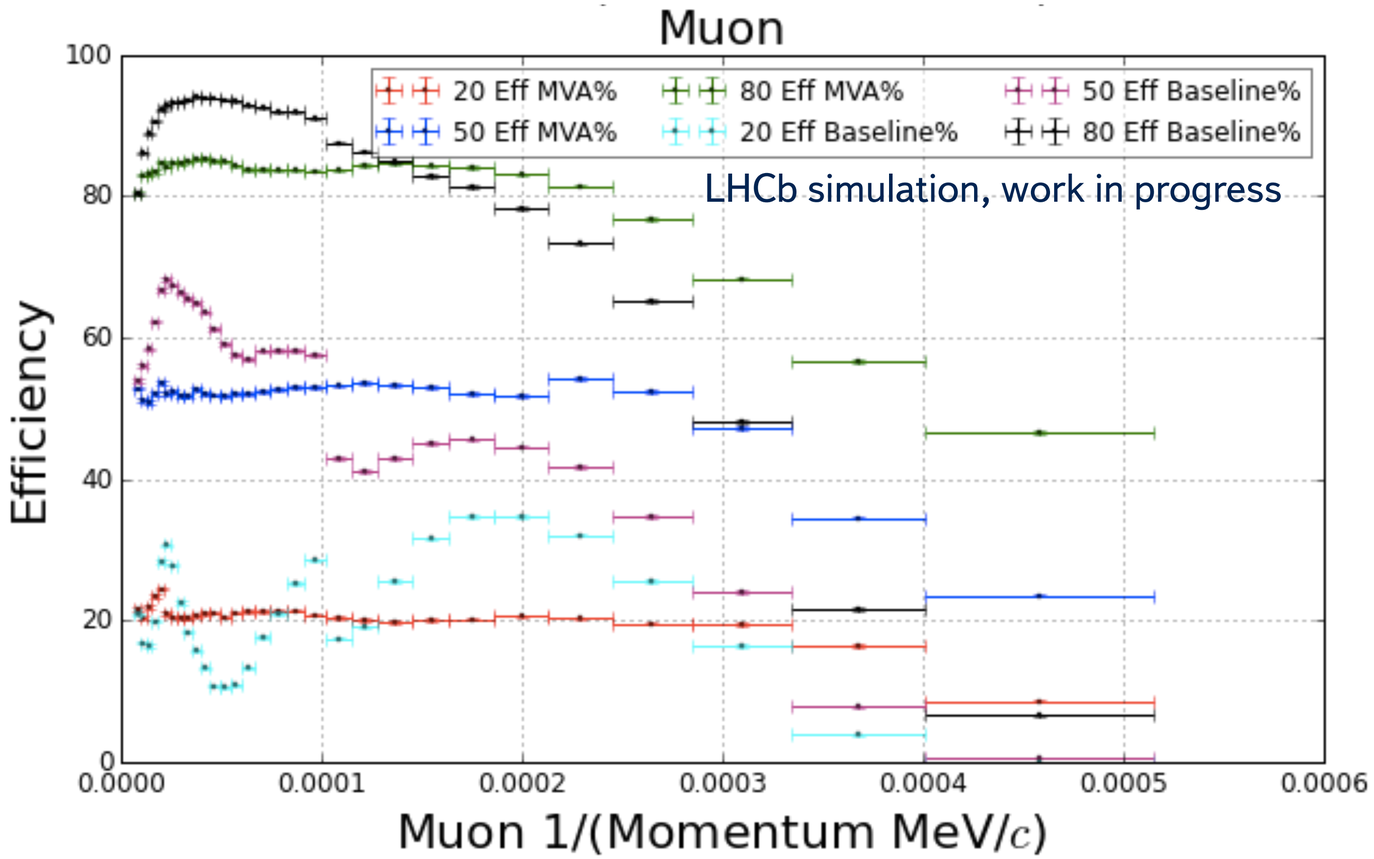


Real world

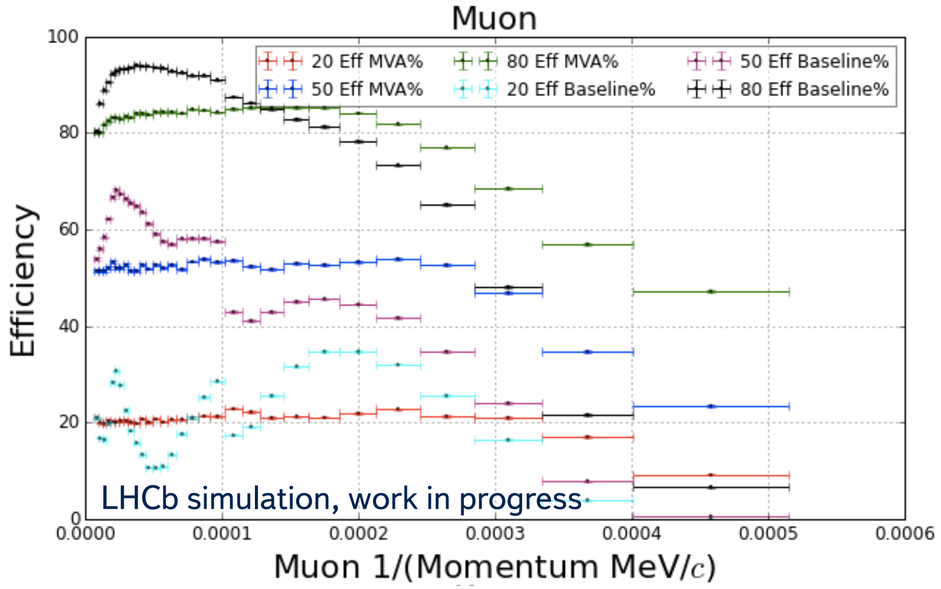
Flat Model vs Baseline

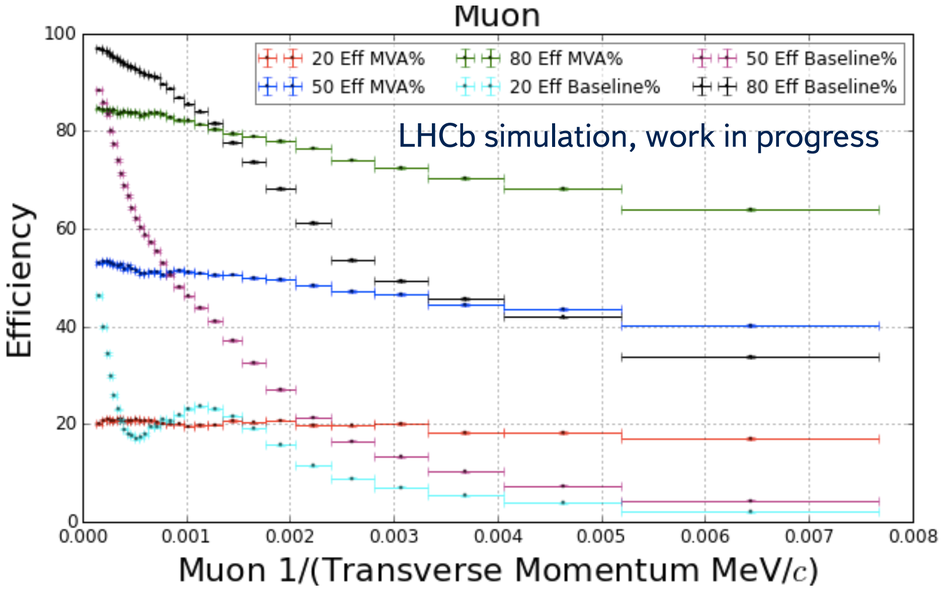
〉 There is a strong dependency on momentum for conventional models (see baseline efficiencies on the plot)

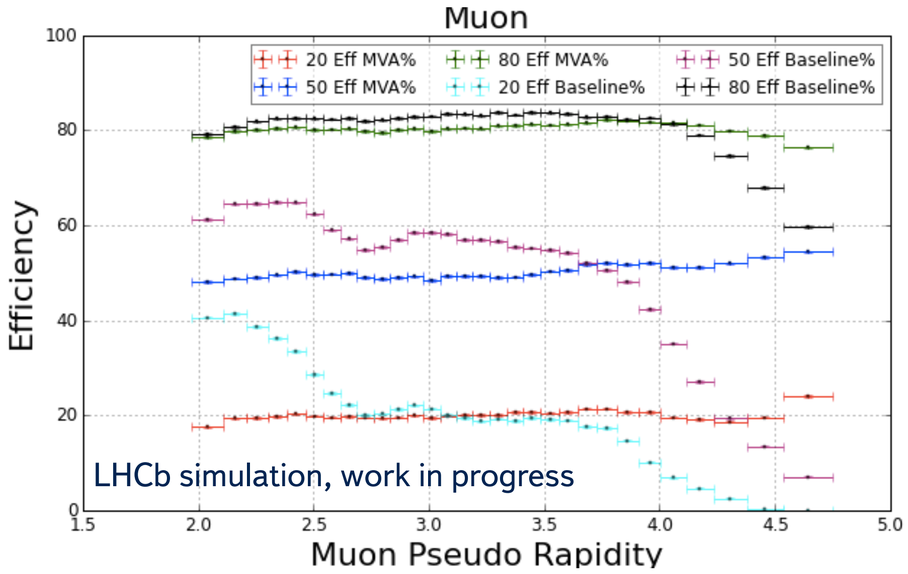
〉 Uniform boosting approach suppresses this dependency.

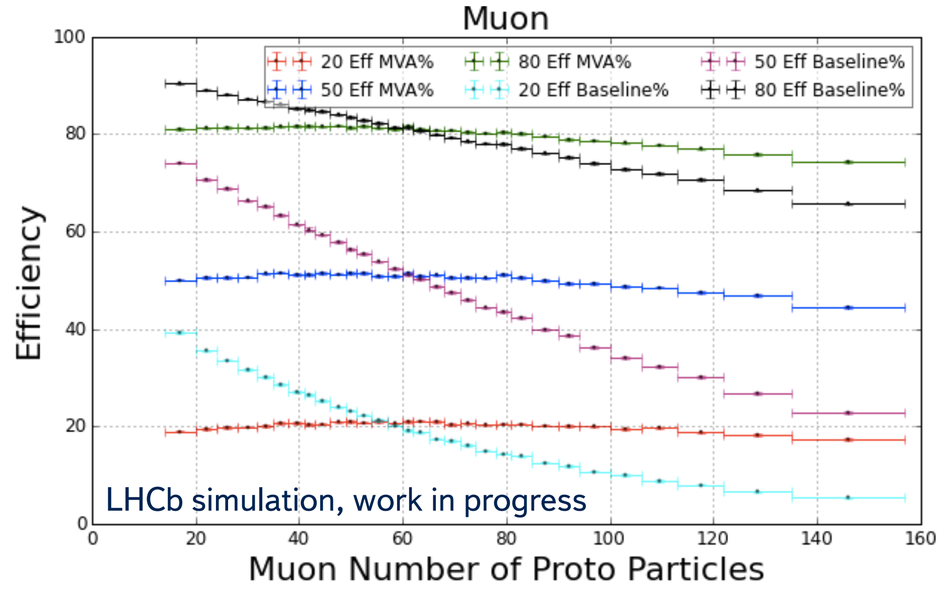


Uniform boosting provides flatness along 4 variables at once









Flat Models: ROC AUC

- Flatness is a very strong restriction, holding this restriction leads to quality decreasing.
- Still, flat model is not worse than baseline in ROC AUCs

	Ghost	Electron	Muon	Pion	Kaon	Proton
baseline	0.9484	0.9854	0.9844	0.9345	0.9147	0.9178
P + Pt flatness	0.9605	0.9883	0.9886	0.9514	0.9146	0.9139
2d(P, Pt) flatness	0.9594	0.9868	0.9865	0.9494	0.9049	0.8993
4d(P, Pt, eta, nTracks) flatness	0.9593	0.9861	0.9864	0.9474	0.9062	0.8976
P + Pt + eta + nTracks flatness	0.9600	0.9874	0.9884	0.9503	0.9130	0.9129

Conclusion

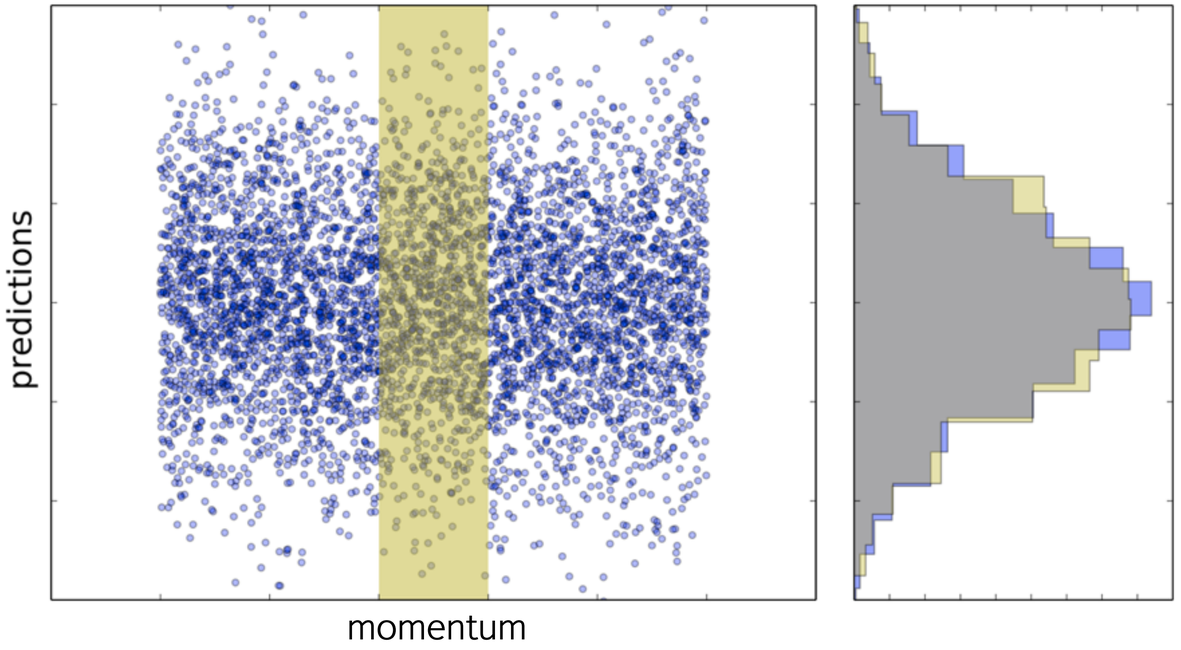
- Multiclass classification approach efficiently works in PID problem
- Modern approaches for neural networks allows to essentially improve PID quality
  - training dataset size matters
- BDTs and NNs reach very similar qualities in this problem
- Uniform boosting successfully works in PID problem diminishing undesired dependency without dropping quality :)

Thanks for attention

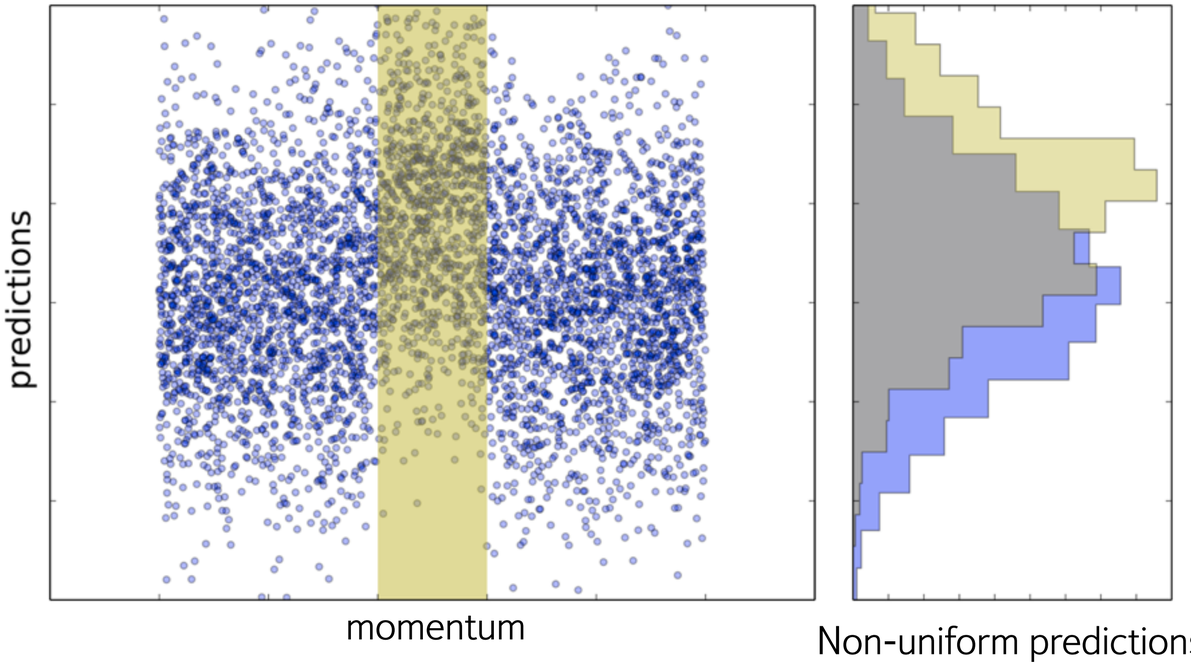
PID

Backup

Non-uniformity measure



Uniform predictions



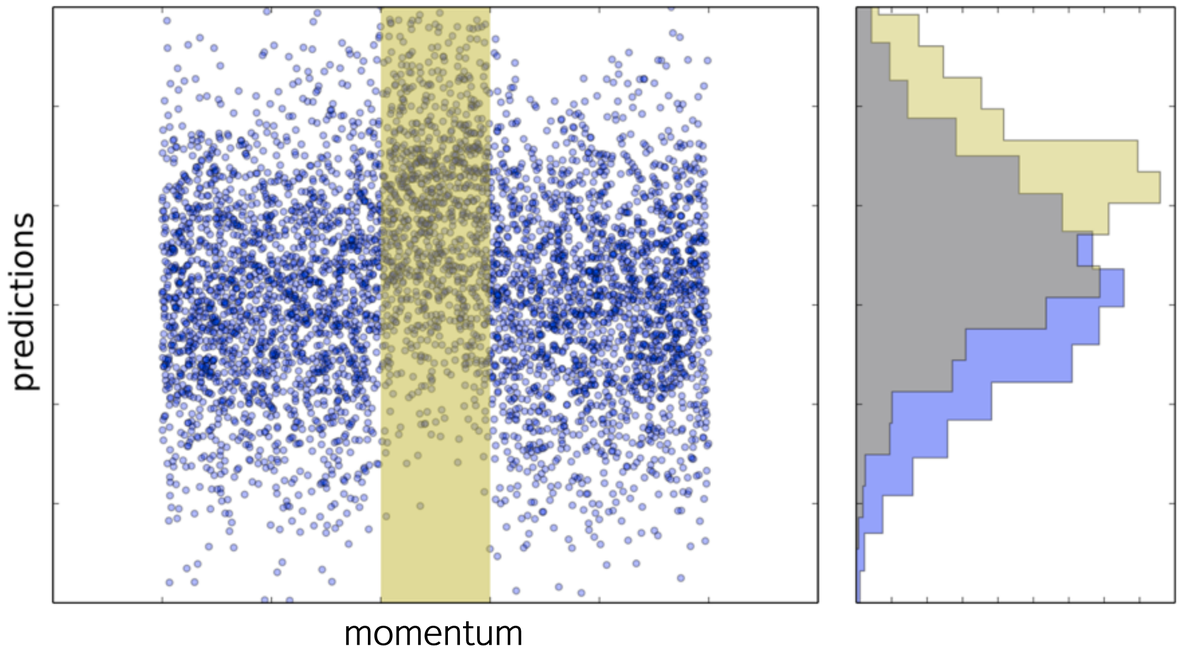
Non-uniform predictions (peak in highlighted region)

- difference in the efficiency can be detected by analyzing distributions
- uniformity = no statistical dependence between the momentum and predictions

Non-uniformity measure

Average contributions (difference between global and local distributions) from different regions in the momentum: use for this Cramer-von Mises measure (integral characteristic)

$$\mathrm{CvM} = \sum_{\mathrm{region}} \int |F_{\mathrm{region}}(s) - F_{\mathrm{global}}(s)|^2 \, dF_{\mathrm{global}}(s)$$



Flat model construction

- Classifier optimizes a loss function during training
- Idea is to use additional loss term in the optimization problem (FL is flatness loss):

$$loss = AdaLoss + \alpha FL$$

The ***AdaLoss*** term corresponds to the classification quality, the ***FL*** term - to the flatness, α is a parameter to control the trade-off

- Optimization methods use gradient of the loss
- Cramer-von Mises metric is not differentiable
- Flatness loss is similar to the Cramer-von Mises metric, but it is differentiable