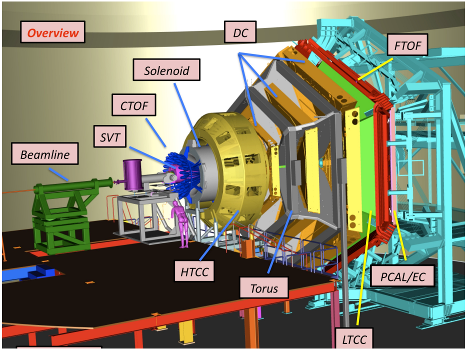
Update on CLAS12 PID and Machine Learning

Daniel Lersch, Michael C. Kunkel

07.03.2018

Introduction

- **Wanted:** Particle masses
  $m_{e^\pm}$, $m_{\pi^\pm}$, m_p, $m_{K^\pm}$, ...
- **Given:** Information from CLAS12 sub detector systems
- **Approach:** Combine detector information and feed them into a machine learning algorithm
  $\Rightarrow$ Get particle type
- All results shown here are based on:
  - Simulated single particle tracks
  - GEMC
  - COATJAVA
  - Apache-Spark framework
- **Last Meeting:**
  - ROC-Curves and -Metrics
  - Separation of of e^- / π^-
  - Efficiency for e^-: $\sim$ 95%
  - No systematics related to magnetic field
  - Reliable performance for resolution effects $\lesssim$ 15%

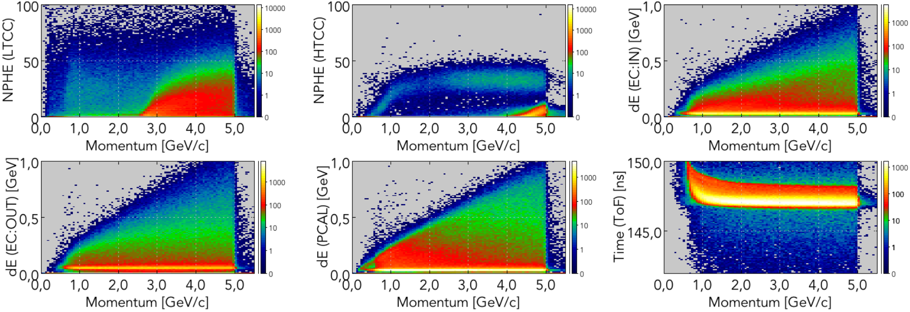


$\Rightarrow$ **Todays Meeting:**

i) Updates on the identification of e^- / π^-
ii) Separation of K^- / π^-
iii) Deep Learning in PANDA

Update on e^-/π^- Separation: Change of Momentum Range

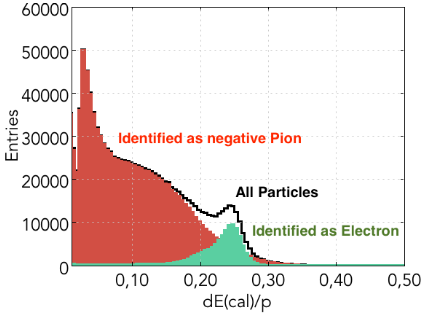
Last Meeting

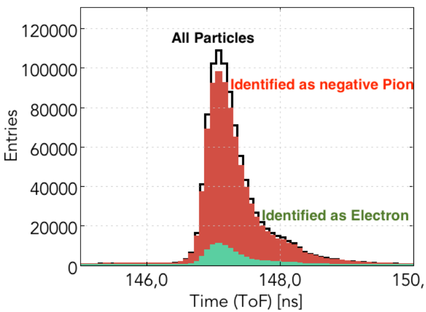


- Last meeting: Momentum range up to 5 GeV
- Variables used for PID: p, ΔE_{in}, ΔE_{out}, ΔE_{pcal}, $N_{phe}(LTCC)$, $N_{phe}(HTCC)$
- Efficiency for e^-: $\sim 95\%$
- Classifier: Neural Network, Boosted Decision Tree

Update on e^-/π^- Separation: Change of Momentum Range

Last Meeting

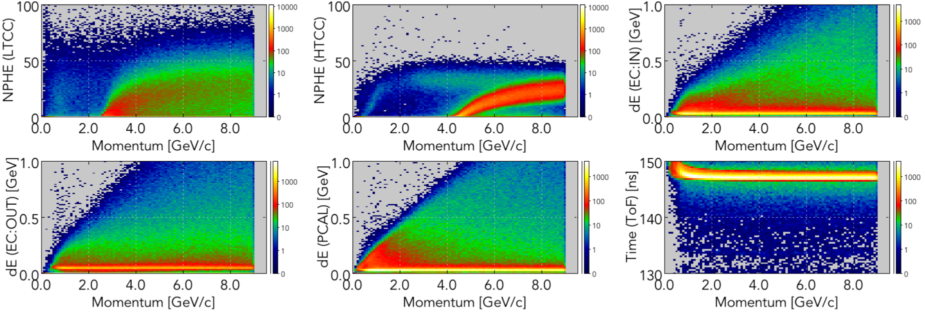




- Last meeting: Momentum range up to 5 GeV
- Variables used for PID: p, ΔE_{in}, ΔE_{out}, ΔE_{pcal}, $N_{phe}(LTCC)$, $N_{phe}(HTCC)$
- Efficiency for e^-: $\sim 95\%$
- Classifier: Neural Network, Boosted Decision Tree

Update on e^-/π^- Separation: Change of Momentum Range

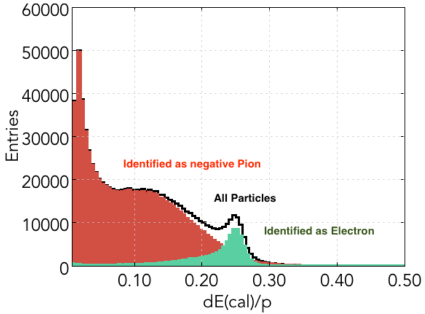
This Meeting

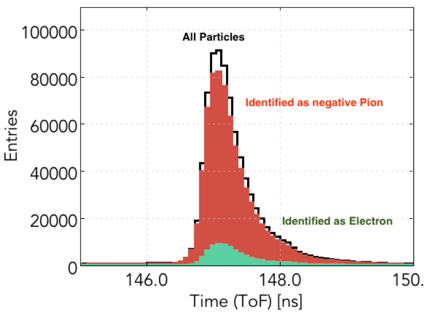


- Momentum range up to 9 GeV
- Variables used for PID: p, θ, ΔE_{in}, ΔE_{out}, ΔE_{pcal}, $N_{phe}(LTCC)$, $N_{phe}(HTCC)$
- Efficiency for e^-: $\sim 95\%$
- Classifier: Neural Network, Boosted Decision Tree

Update on e^-/π^- Separation: Change of Momentum Range

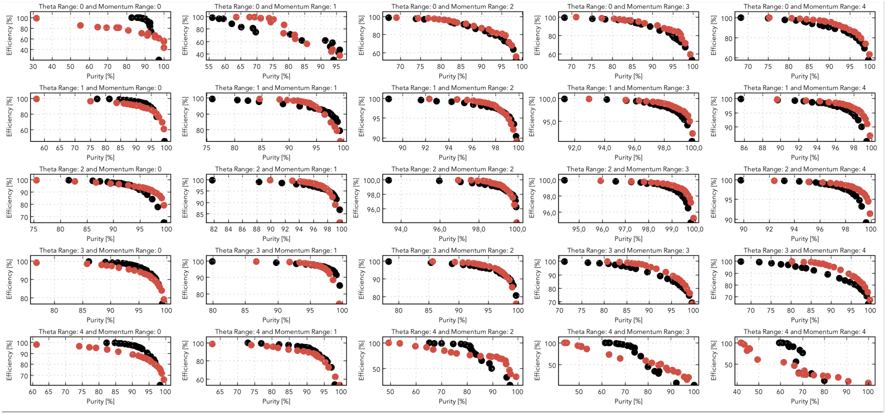
This Meeting





- Momentum range up to 9 GeV
- Variables used for PID: p, θ, ΔE_{in}, ΔE_{out}, ΔE_{pcal}, $N_{phe}(LTCC)$, $N_{phe}(HTCC)$
- Efficiency for e^-: $\sim 95\%$
- Classifier: Neural Network, Boosted Decision Tree

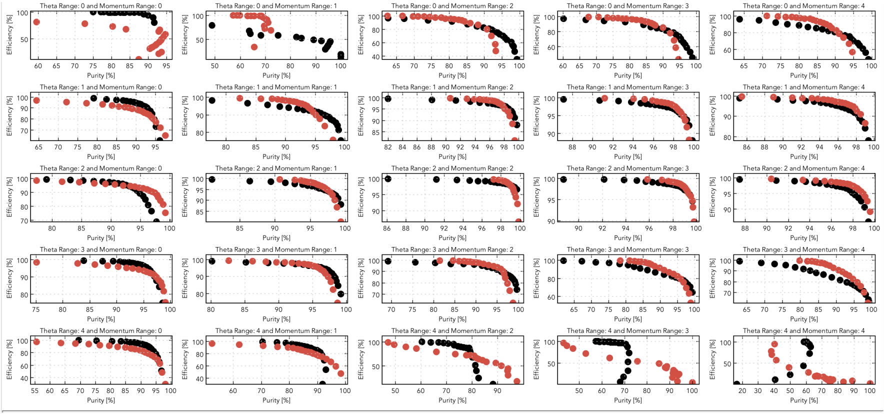
Update on e^-/π^- Separation: Geometrical Performance



- ROC-Curves for different detector regions $\downarrow \theta \in [0°, 45°] \rightarrow p \in [0.05\,\mathrm{GeV}, 9.0\,\mathrm{GeV}]$
- Performance shown here for a boosted decision tree
- See different separation performance

$\Rightarrow$ **Use different classifier in different detector regions $\Rightarrow$ Optimize performance**

Update on e^-/π^- Separation: Geometrical Performance

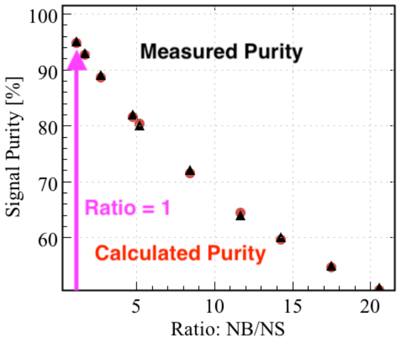


- ROC-Curves for different detector regions $\downarrow \theta \in [0^\circ, 45^\circ] \rightarrow p \in [0.05\,\mathrm{GeV}, 9.0\,\mathrm{GeV}]$
- Performance shown here for a neural network
- See different separation performance

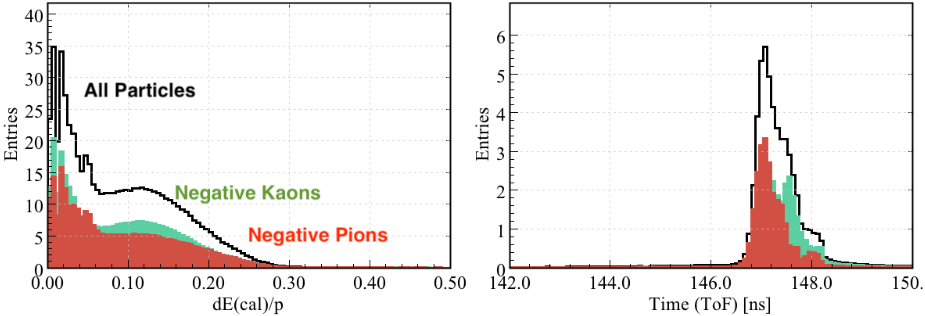
$\Rightarrow$ **Use different classifier in different detector regions $\Rightarrow$ Optimize performance**

Update on e^-/π^- Separation: Purity Study

- Issue/Problem: Purity changes with ratio: $R \equiv \frac{\#\text{Background Events}}{\#\text{Signal Events}}$
- Having ϵ_S as signal efficiency and *FPR* as false positive rate, leads to:
  Purity $= \epsilon_S \times [\epsilon_S + R \times \text{FPR}]^{-1}$ / Here: $\epsilon_S = 95\%$ and FPR $= 5\%$
- ϵ_S and FPR characterise the classifier used and are (ideally) constant
  $\Rightarrow$ Calculate purity for different R and monitor performance
- Maximize Purity:
  i) FPR $\mapsto 0$ (e.g. deep learning, combining classifier)
  ii) Have control over R (e.g. cascade classification)

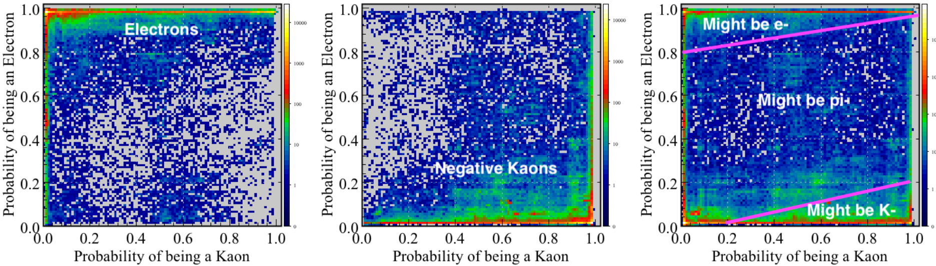


Separation of K^- / π^-



- Variables for PID:
  p, θ, $N_{phe}(LTCC)$, $N_{phe}(HTCC)$, $\sum \Delta E_{cal}$, ToF
- Obtain efficiency for K^-: $\sim 87\%$
- Used a boosted decision tree for classification
- Note: Variable selection has not been tuned
  $\rightarrow$ There is some room for improvement!

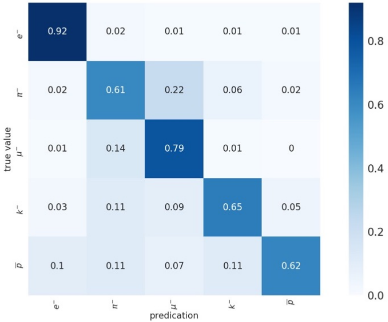
Separation of K^- / π^-: Correlations

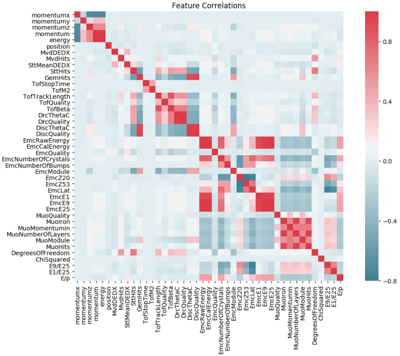


- Look at correlations between classifier output(s)
- Left: Just e^- / Centre: Just K^- / Right: e^-, K^- and π^-
- Perform a 2D cut $\Leftrightarrow$ handle correlations
- Just clustering algorithm to optimize event selection

Deep Learning with PANDA

ML Group at IKP1 Juelich: M.C. Kunkel & W. Esmail

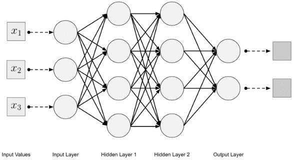


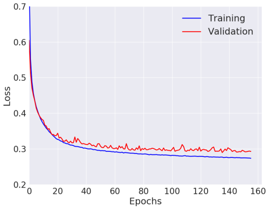


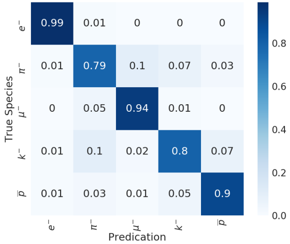
- PID performed at PANDA: $\sim$ 43 input variables (e.g. momentum, position, raw energy, ToF, EMC hits, Muon quality,..)
- Cut based PID

Deep Learning with PANDA

ML Group at IKP1 Juelich: M.C. Kunkel & W. Esmail





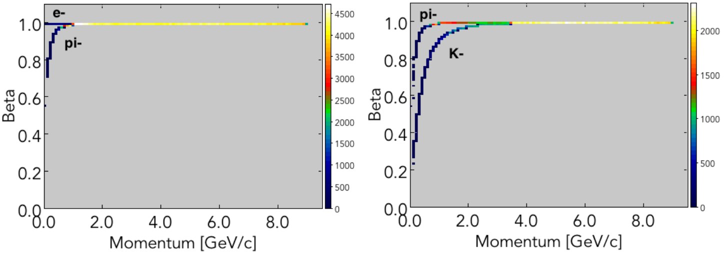


- PID performed at PANDA: $\sim$ 43 input variables (e.g. momentum, raw energy, position, ToF, EMC hits, Muon quality,..)
- Used/Tested boosted decision tree
- Go over to Deep Learning in order to handle vast amount of information
  - Trained a (deep) neural net with 3 hidden layers
  - Optimization of node response functions

  $\Rightarrow$ **Apply and test this method for CLAS12**
  - Pros: Use raw data / second level trigger / handle multiple variable input
  - Cons: Variable importance on specific PID / Redundancy / Complexity

Summary and Outlook

- ☑ Set up and use Machine Learning for PID of: e^-, π^- and K^-
  $\rightarrow$ With efficiencies: 95%, 97% and 87%
- ☑ Checked systematic influences and robustness
  $\rightarrow$ Magnetic field, resolution effects
- ☑ First insights into deep learning in PANDA
- ☐ Perform PID studies for p (ongoing)
- ☐ **Estimation of (model independent) Variable Importance** (done, but lost code...)
- ☐ Include further classifier algorithms (e.g. SVM, Likelihood,...) (partly done)
- ☐ Possible X-check of PID-results (e.g. compare β(PID) with β(ToF))
- ☐ **Test methods on Data**

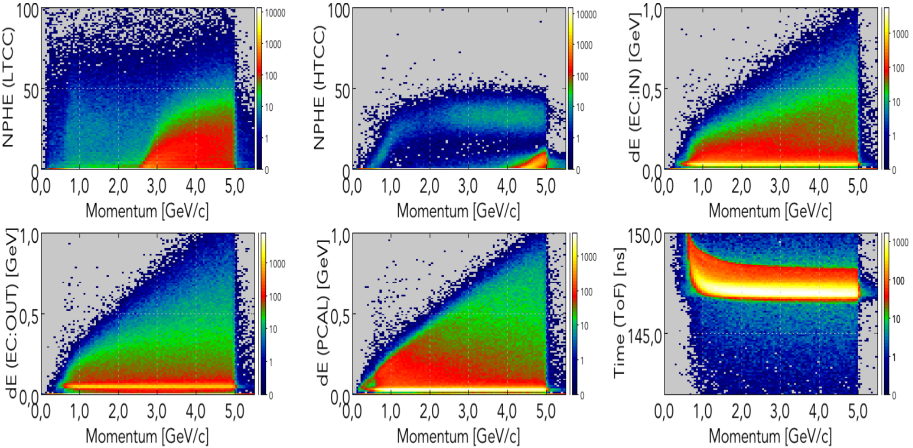


Content

Backup Stuff

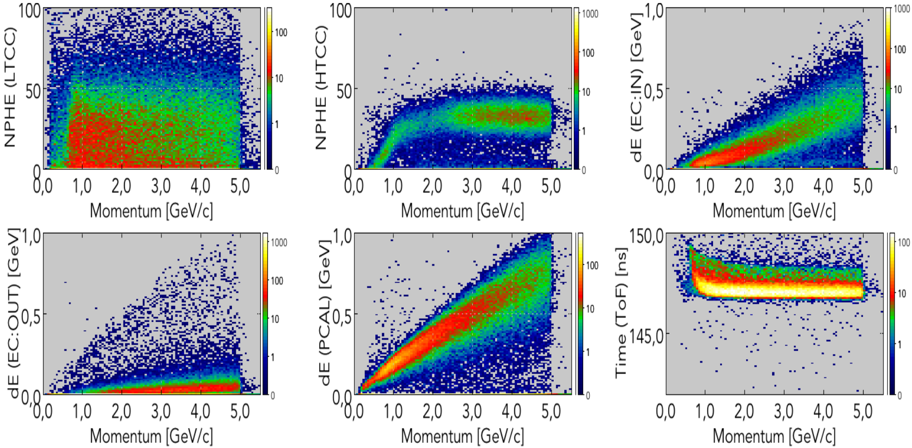
Backup: The $e^-\pi^-$ Data Set

All Particles



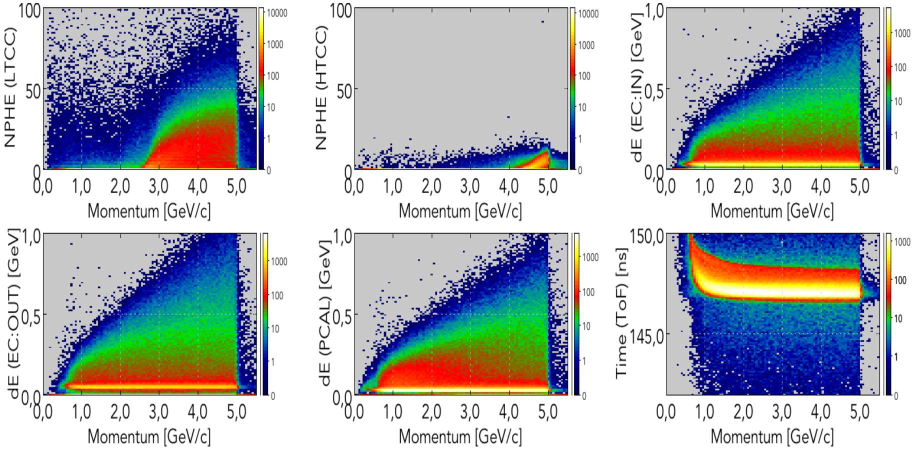
Backup: The $e^-\pi^-$ Data Set

Electrons

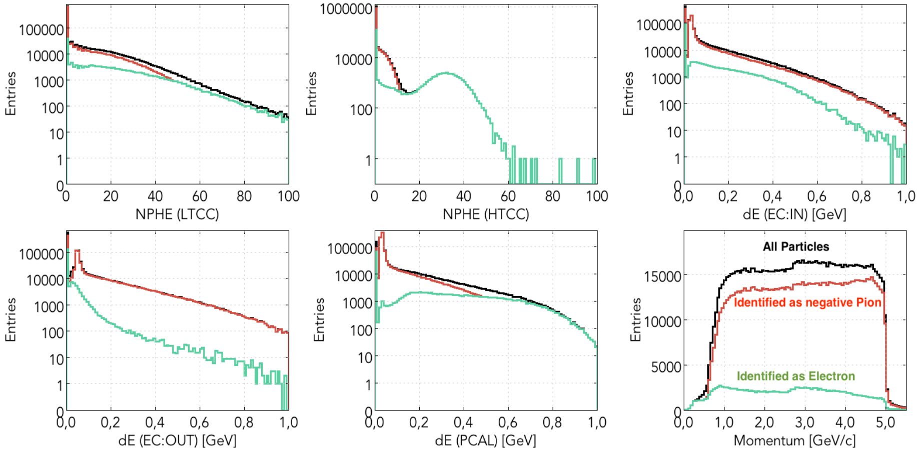


Backup: The $e^-\pi^-$ Data Set

Negative Pions



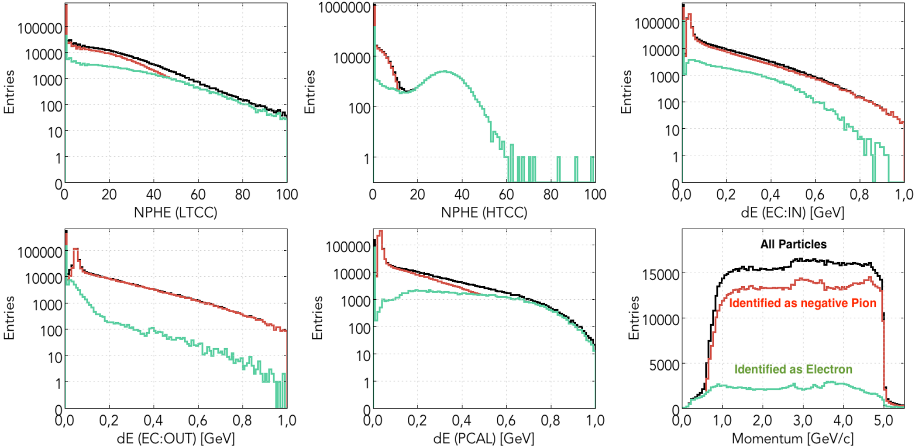
Backup: 1D Distributions before and after PID



Apply neural network algorithm on $e^{-}\pi^{-}$ data set:

- $\epsilon_S = 94\%$
- $P_S = 62\%$

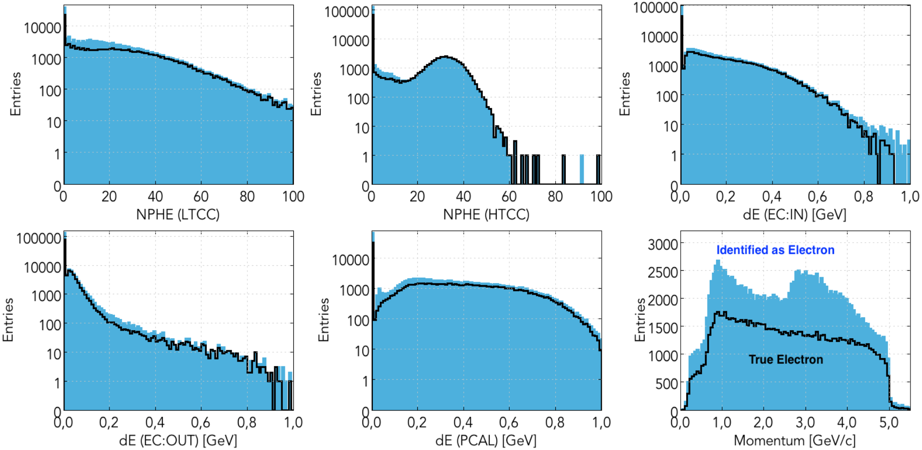
Backup: 1D Distributions before and after PID



Apply boosted decision tree algorithm on $e^-\pi^-$ data set:

- $\epsilon_S = 96\%$
- $P_S = 58\%$

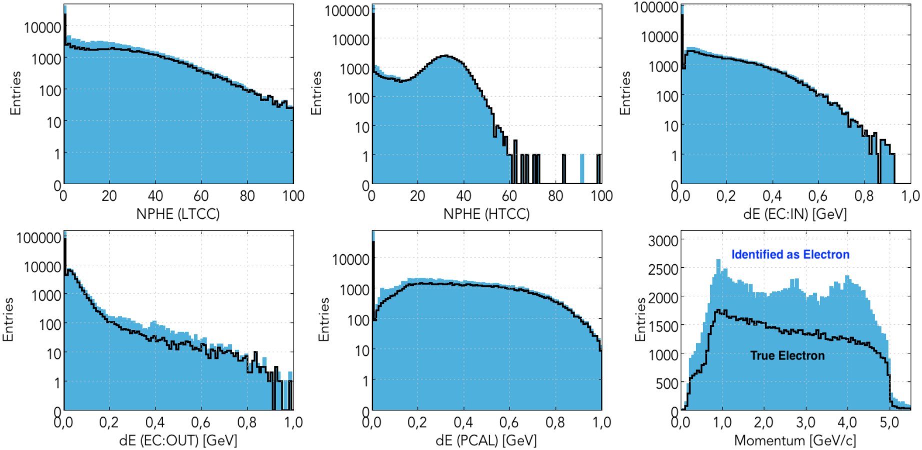
Backup: Compare True and ID Distributions



Apply neural network algorithm on $e^-\pi^-$ data set:

- $\epsilon_S = 94\%$
- $P_S = 62\%$

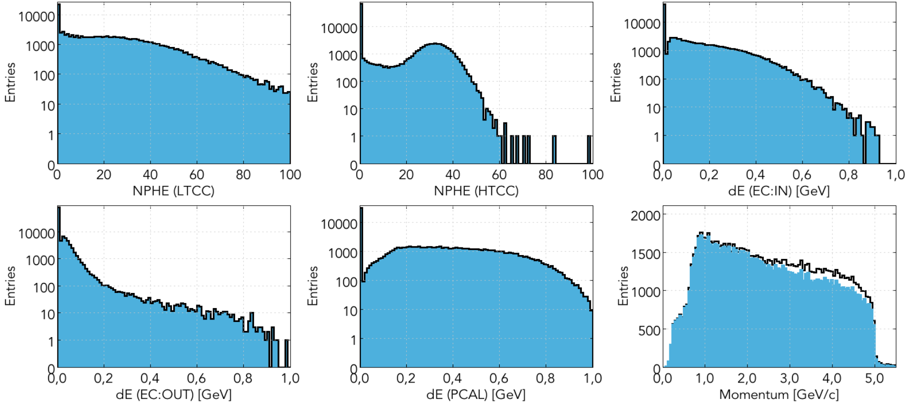
Backup: Compare True and ID Distributions



Apply boosted decision tree algorithm on $e^-\pi^-$ data set:

- $\epsilon_S = 96\%$
- $P_S = 58\%$

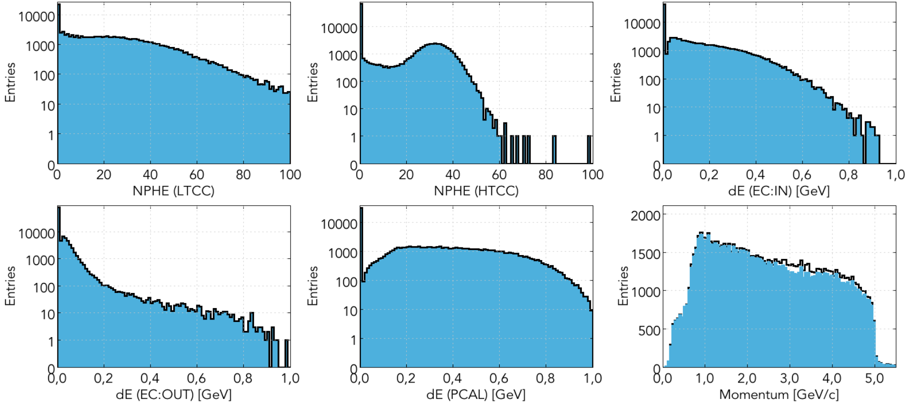
Backup: Compare True and ID Distributions for true Electrons



Apply neural network algorithm on $e^{-}\pi^{-}$ data set:

- $\epsilon_S = 94\%$
- $P_S = 62\%$

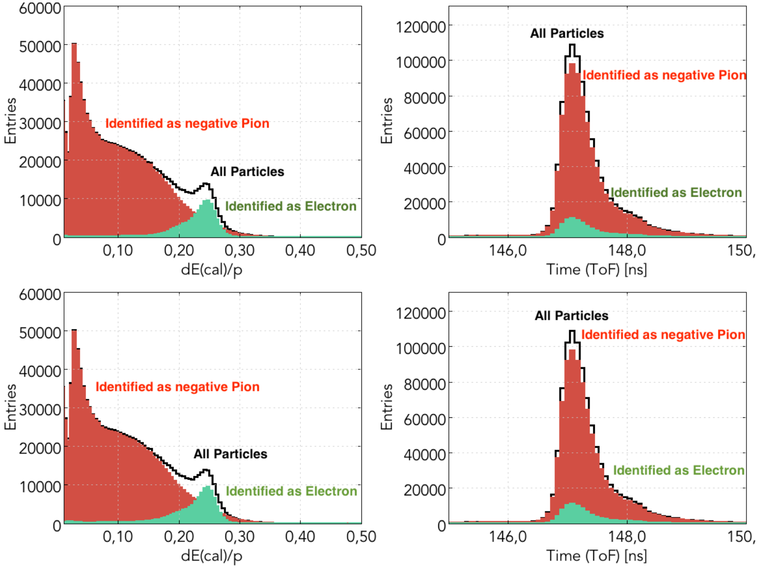
Backup: Compare True and ID Distributions for true Electrons



Apply boosted decision tree algorithm on $e^-\pi^-$ data set:

- $\epsilon_S = 96\%$
- $P_S = 58\%$

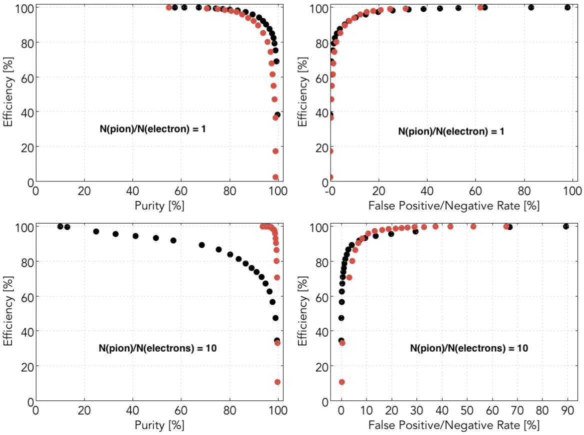
Backup: Comparison of PID-Plots



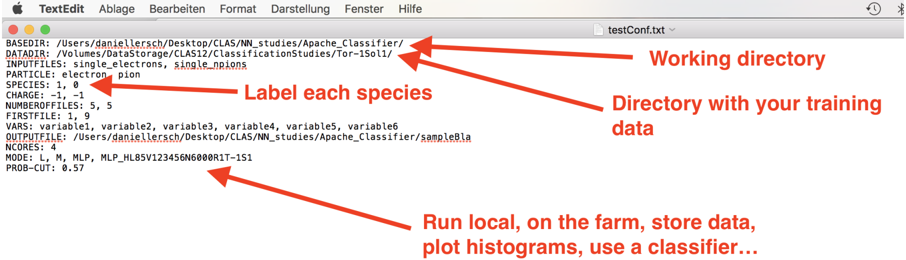
Top: MLP / Bottom: GBT

Backup: Useful Relations

- ϵ_S and FPR are features of the classifier and (ideally) independent of the ratio between the species
- $\left(\frac{N_B^{all}}{N_S^{all}}\right)_{rec} = \left[\frac{P_S}{\epsilon_S} \times \frac{\epsilon_B}{P_B}\right] \times \left(\frac{N_B^{all}}{N_S^{all}}\right)_{true}$
- $P_S = \epsilon_S \times \left[\epsilon_S + FPR \times \left(\frac{N_B^{all}}{N_S^{all}}\right)_{true}\right]^{-1}$

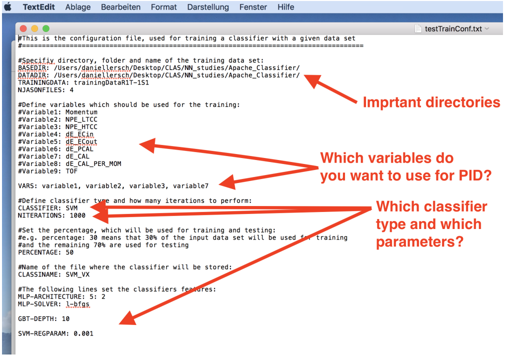


Backup: Generating single Track Training Data



- Java-based Module:
  - Generate training data from reconstructed CLAS12 data
  - Test a classifier on reconstructed CLAS12 data
  - Can be run by/with multiple users/configurations
- Module parameters are specified in a txt-file (see top figure)
- Txt-file generation and submission to farm are run by a single script

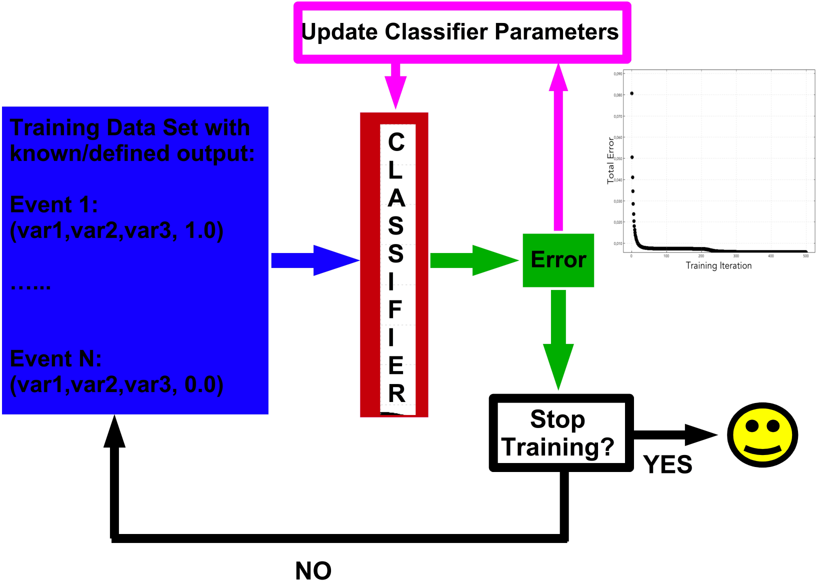
Backup: Training and Choice of the Classifier



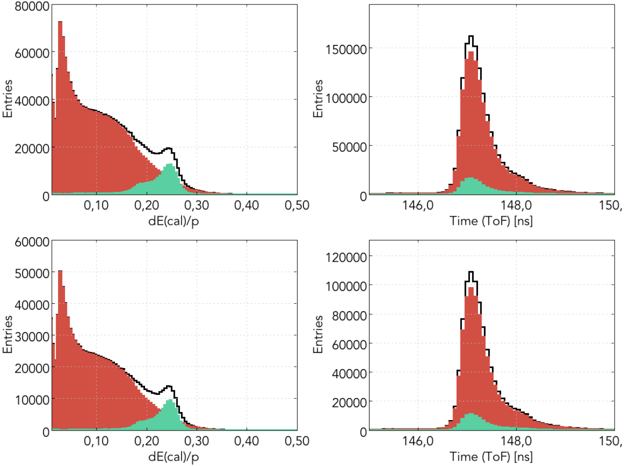
```
#This is the configuration file, used for training a classifier with a given data set
#=====================================================================================

#Specifiy directory, folder and name of the training data set:
BASEDIR: /Users/daniellersch/Desktop/CLAS/NN_studies/Apache_Classifier/
DATADIR: /Users/daniellersch/Desktop/CLAS/NN_studies/Apache_Classifier/
TRAININGDATA: trainingDataR1T-1S1
NJASONFILES: 4

#Define variables which should be used for the training:
#Variable1: Momentum
#Variable2: NPE_LTCC
#Variable3: NPE_HTCC
#Variable4: dE_ECin
#Variable5: dE_ECout
#Variable6: dE_PCAL
#Variable7: dE_CAL
#Variable8: dE_CAL_PER_MOM
#Variable9: TOF

VARS: variable1, variable2, variable3, variable7

#Define classifier type and how many iterations to perform:
CLASSIFIER: SVM
NITERATIONS: 1000

#Set the percentage, which will be used for training and testing:
#e.g. percentage: 30 means that 30% of the input data set will be used for training
#and the remaining 70% are used for testing
PERCENTAGE: 50

#Name of the file where the classifier will be stored:
CLASSNAME: SVM_VX

#The following lines set the classifiers features:
MLP-ARCHITECTURE: 5: 2
MLP-SOLVER: l-bfgs

GBT-DEPTH: 10

SVM-REGPARAM: 0.001
```


- Java-based Module:
  - Specify the variables used for training
  - Select the classifier that shall be trained
  - Define how the classifier shall be trained
  - Can be run by/with multiple users/configurations
- Module parameters are specified in a txt-file (see top figure)
- Txt-file generation and submission to farm are run by a single script

Backup: Training a Classifier



Backup: Changing the Magnetic Field



- Top: Classifier trained on: Tor. $= -0.75$ and Sol. $= 0.8$ and applied on: Tor. $= -0.75$ and Sol. $= 0.8$
- Bottom: Classifier trained on: Tor. $= -0.75$ and Sol. $= 0.8$ and applied on: Tor. $= -1$ and Sol. $= 1$