τ decay mode classification

Shenjian Chen, Bowen Zhang, Lei Zhang
NJU-tau-meeting
Nanjing University

Introduction

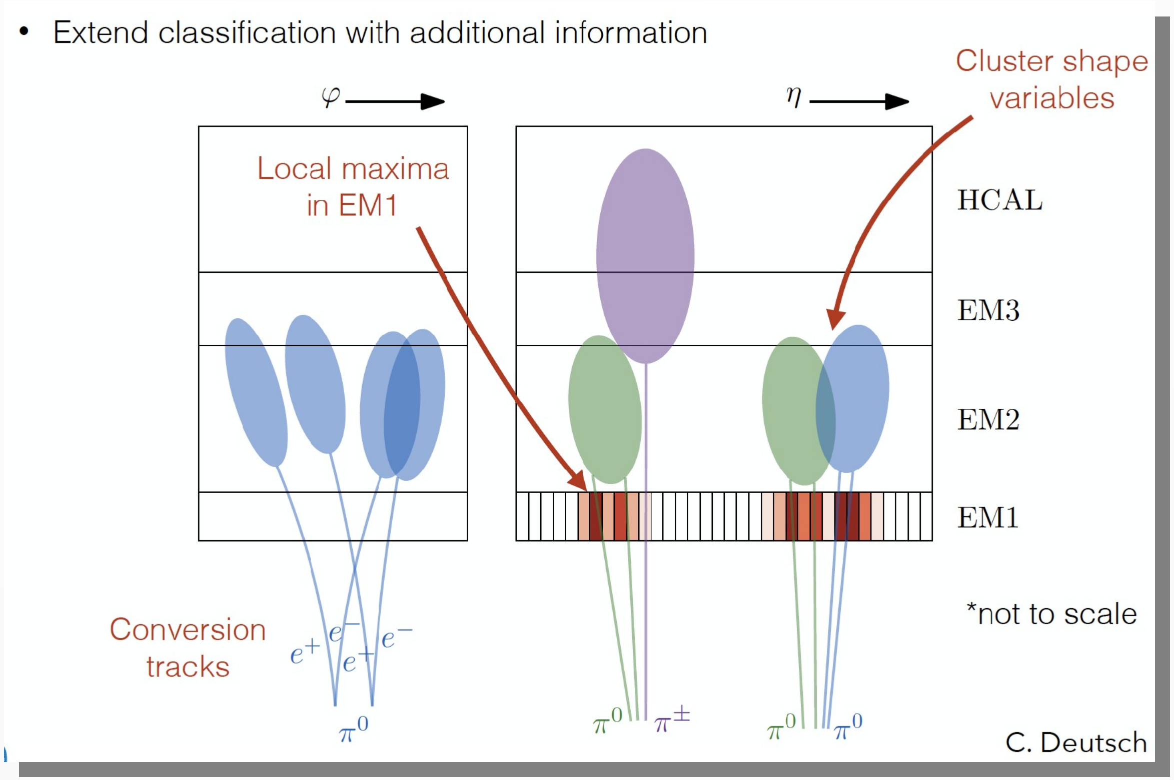
- Improve the five-way τ decay mode classification using machine learning techniques:
- Categorized by the number of charged and neutral products
- Previously use kinematics of the decay products, the identification score of pi0 and the number of photons

Decay mode test	$N(\pi^0_{\text{cand}})$	$N(\pi^0_{\text{ID}})$	Variables
$h^\pm\{0,1\}\pi^0$	≥ 1 1	0 1	S_1^{BDT}, $f_{\pi^0,1}$, $\Delta R(h^\pm,\pi^0)$, $D_{h^\pm}$, N_γ
$h^\pm\{1,\geq 2\}\pi^0$	≥ 2 ≥ 2	1 ≥ 2	S_2^{BDT}, f_{π^0}, m_{π^0}, N_{π^0}, N_γ
$3h^\pm\{0,\geq 1\}\pi^0$	≥ 1 ≥ 1	0 ≥ 1	S_1^{BDT}, f_{π^0}, $\sigma_{E_{\text{T}},h^\pm}$, $m_{h^\pm}$, N_γ

arxiv:1512.05955

Introduction

- The algorithm will be upgraded to neural network version in Run3
- RNN study by Christopher : link
- Use information from:
  - Charged PFO, Neutral PFO, Photon Shots and Conversion Tracks



Input variables

- Last presentation repeated the machinery and studied the high level inputs
  - Conclusion: PanTau BDT inputs doesn’t improve the results. See
- Continue on study input variables
  - Kinematics of each object
  - Replace Pi0 BDT score by the Pi0 BDT ID inputs

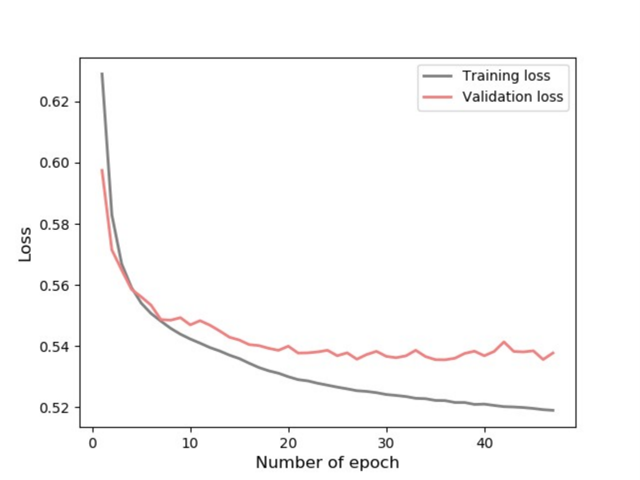
Baseline

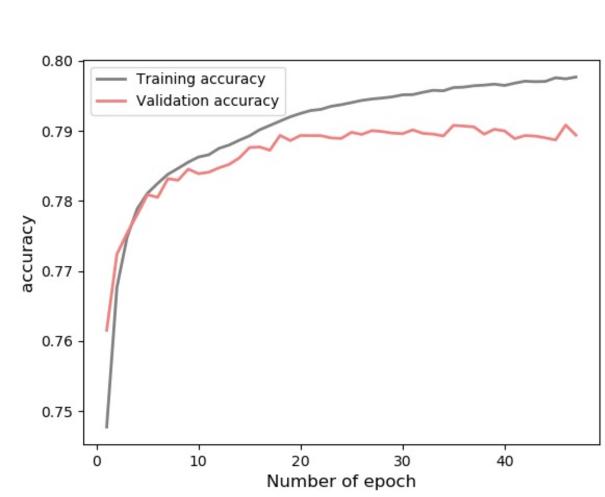
- The training sample is mc16d Gammatautau
  - `mc16_13TeV:mc16_13TeV.425200.Pythia8EvtGen_A14NNPDF23LO_Gammatautau_MassWeight.merge.AOD.e5468_s3126_r10201_r10210`
  - `~12M taus after baseline selection (backup)`
- Subset is used: 4M for training (20% of it for validation) 1M for testing

Baseline

- Slightly change the implementation of the model
  - Implemented using PyTorch framework so far.
  - Remove the TimeDistributed+Dense layer before the LSTM – Training becomes much faster on GPUs.
  - Use bidirectional RNN – hope it can learn both the forward and the backward relation
  - Use the RNN output of the full sequence – pass more information to the final dense layers. (need to check if this is really helpful)
  - Use two layers of RNN – proved to be better than just one.

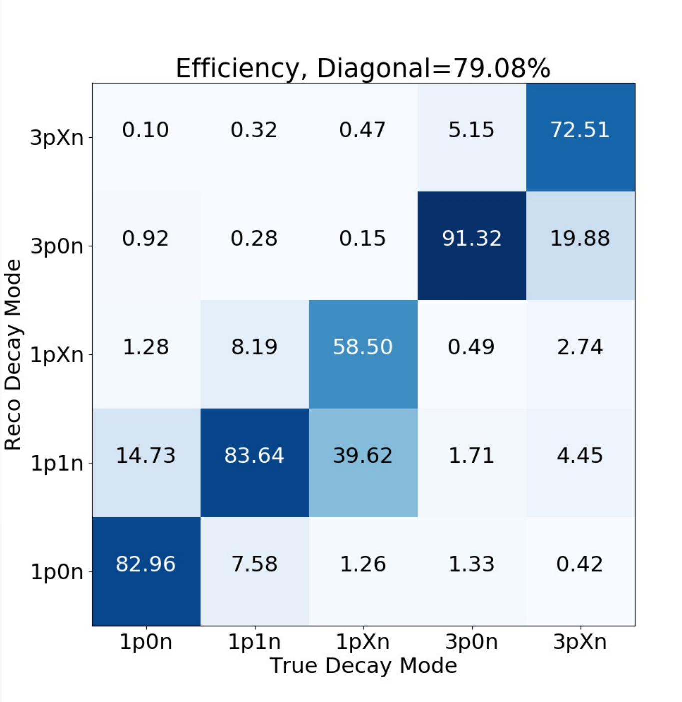
Baseline performance

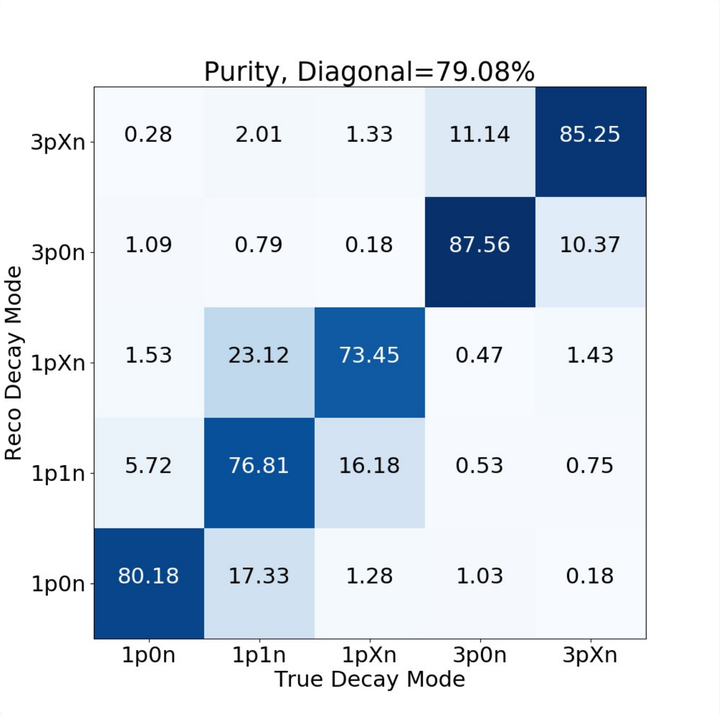




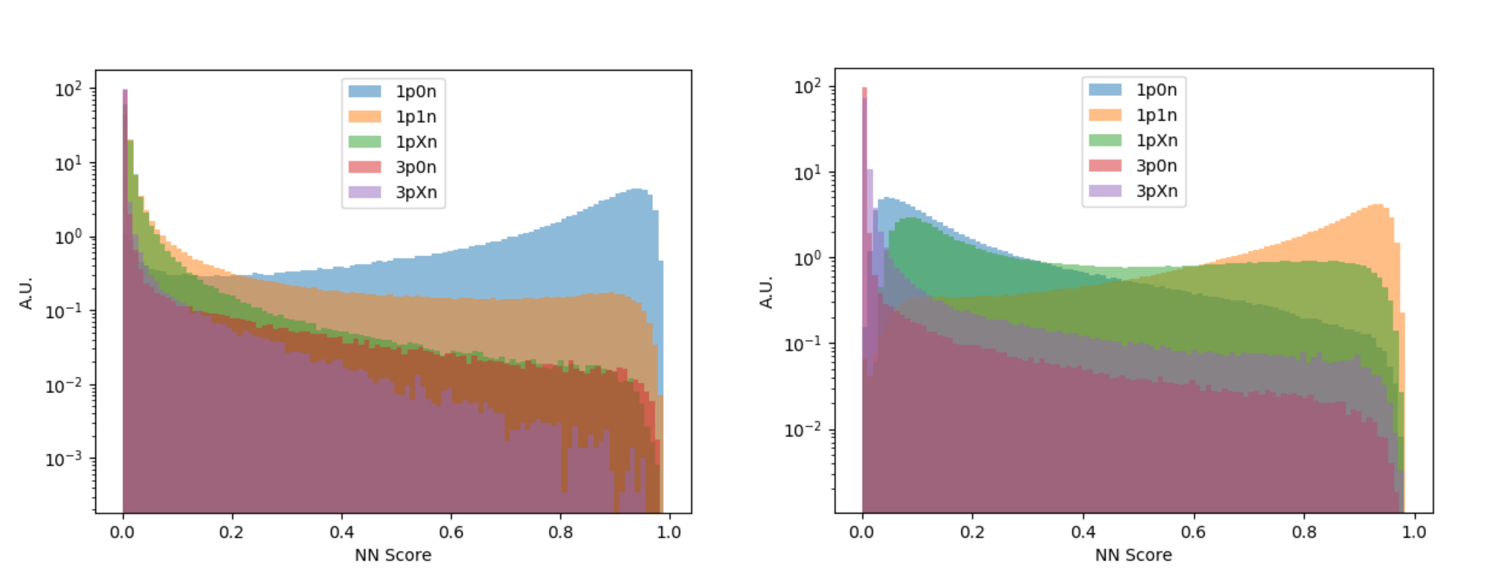
Next step needs to deal with the overtraining

A little bit worse then the previous run (80.4 diagonal). But here the dataset is smaller. And the model is different.

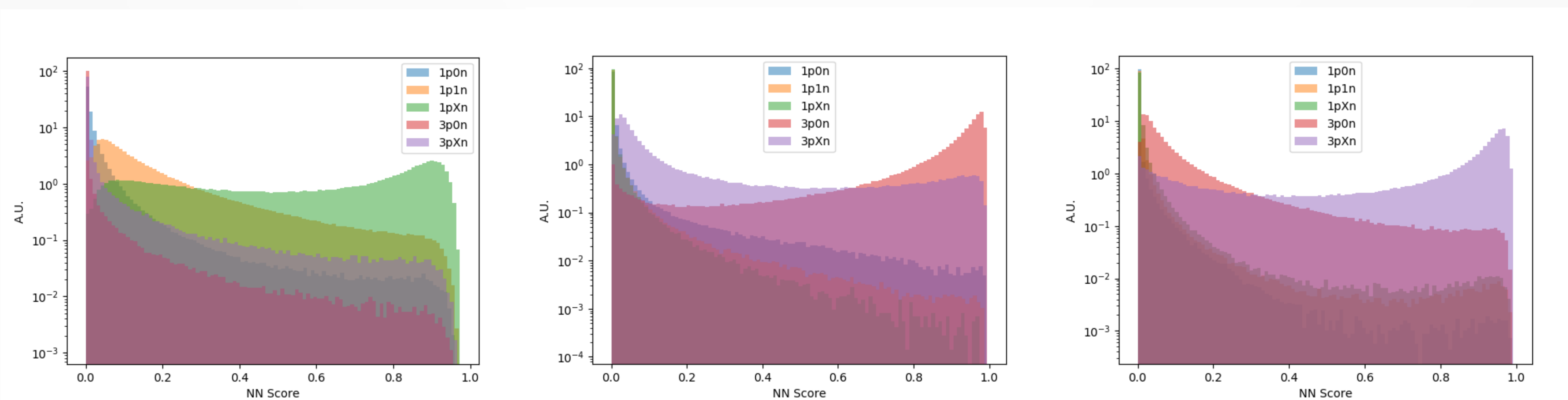




Baseline performance

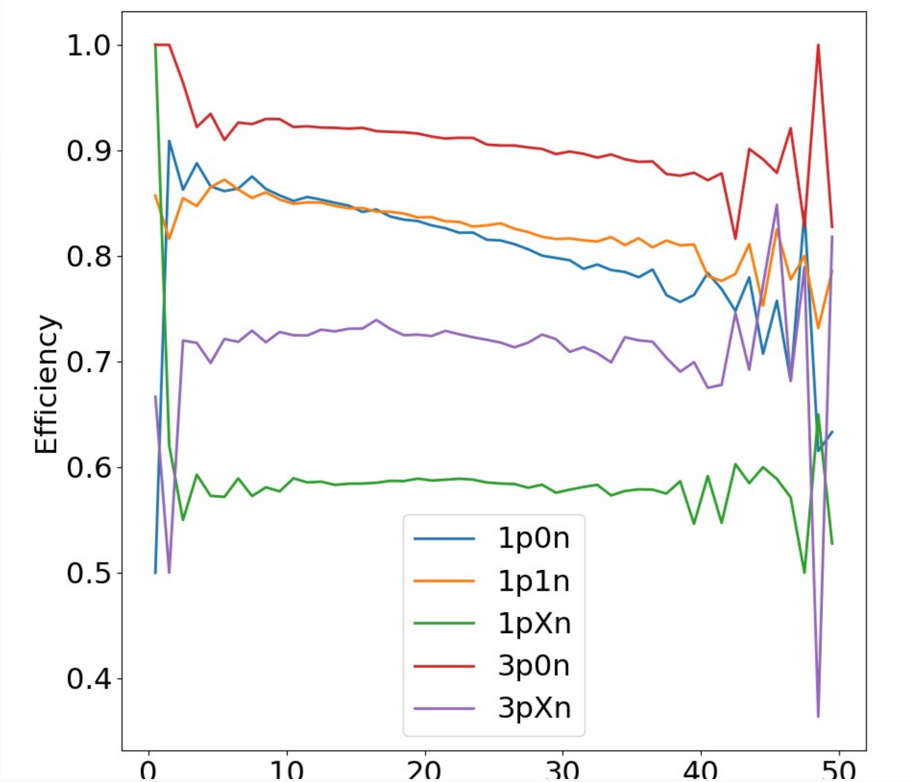


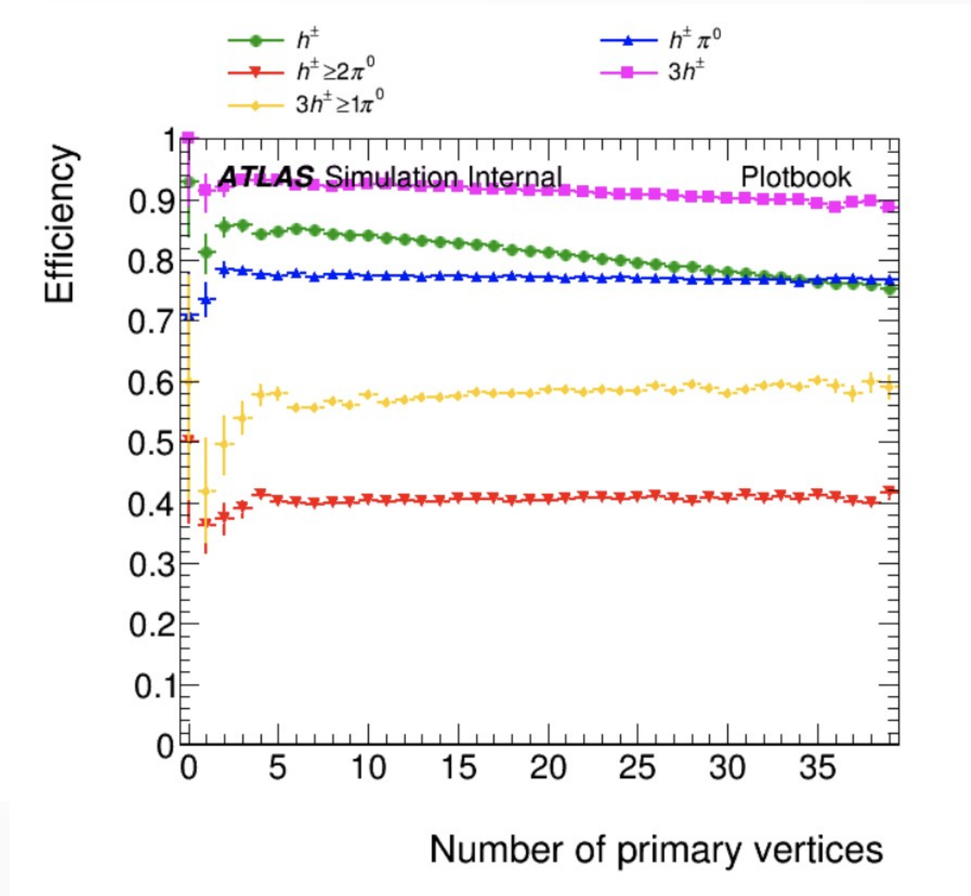
NN output from left to right:
1p0n 1p1n
1pXn 3p0n 3pXn



Baseline performance

Efficiency dependences on number of primary vertices

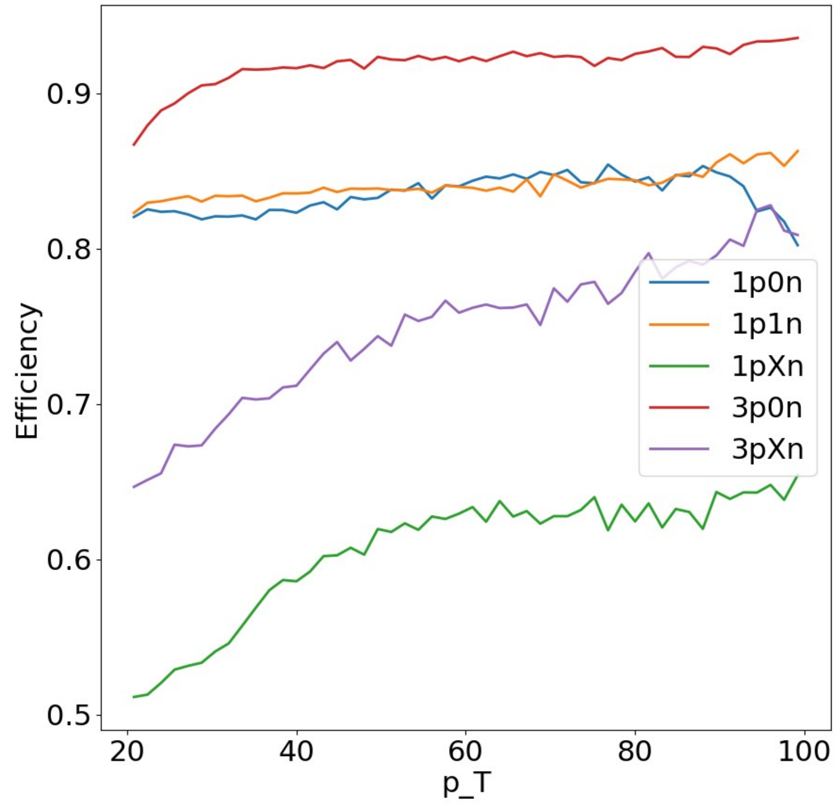


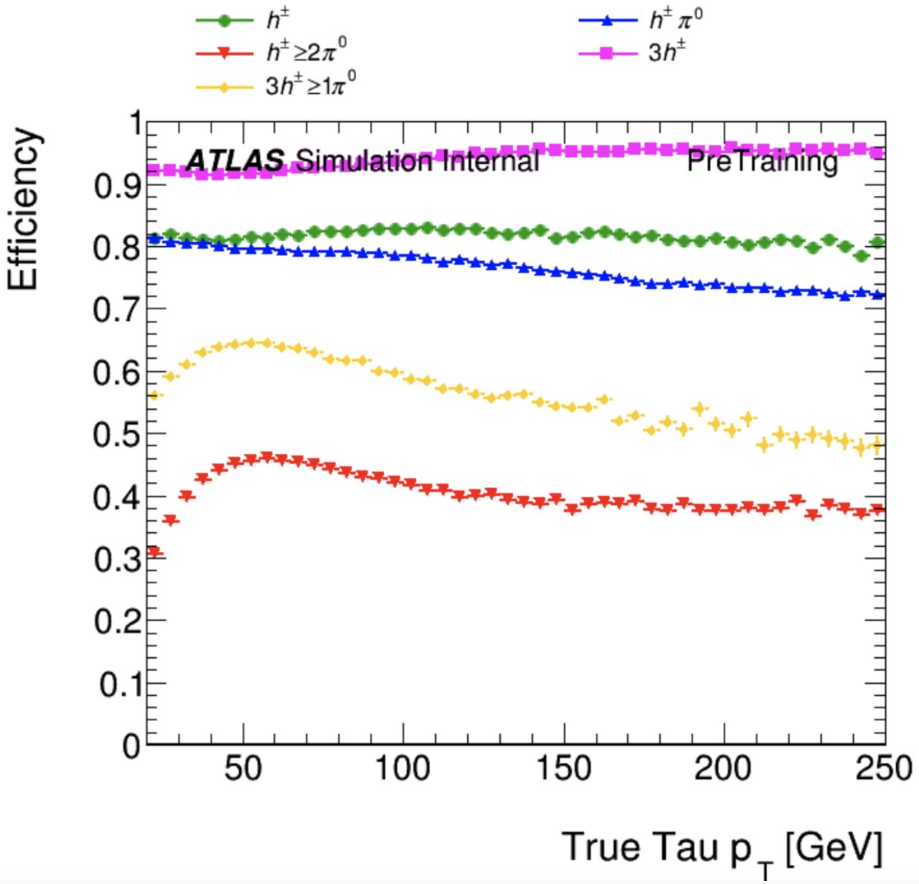


- Especially 1p1n drops at high pile up → Left orange, Right blue

Baseline performance

Efficiency dependences on tau pT

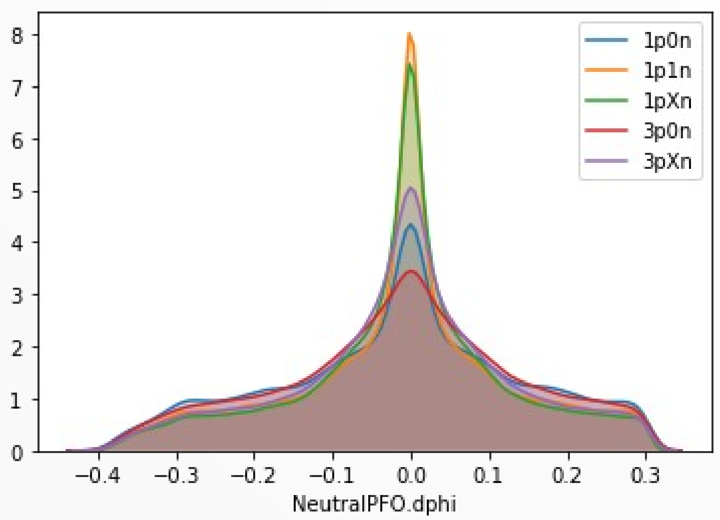


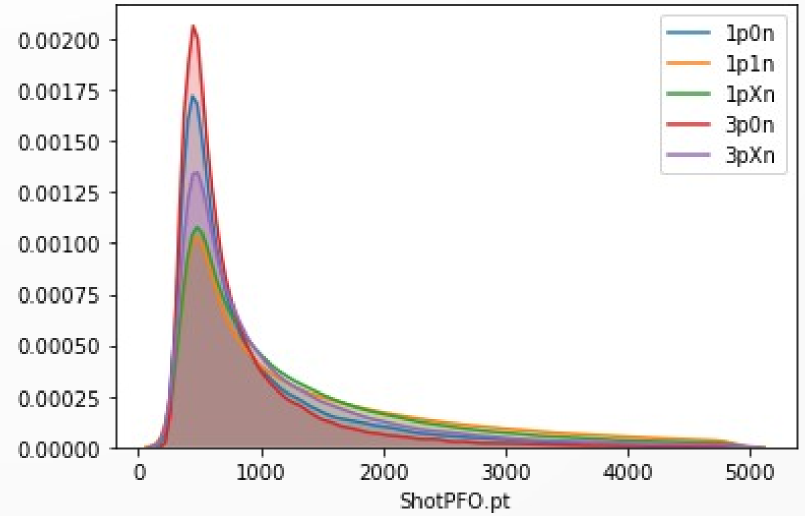


- RNN (Left) only trained on tau with pT<100GeV and doesn't require medium tau ID.
- Trend is different.

Kinematics (4-vectors)

- Only keep pt of the object, pt , eta, phi of tau and the relative eta, phi between the object and the tau axis in the input.
- The length of the sequence, the pt (relative) of the object, and the relative eta, phi are crucial to the classification. e.g.
  - N Charged PFO = 1, 3

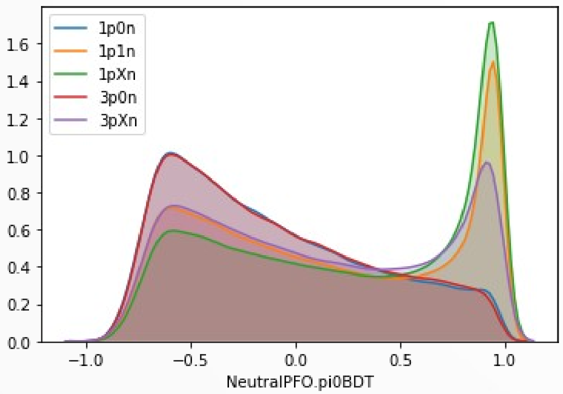




- Result shows ~76.9% diagonal efficiency.

Pi0 BDT ID score and the inputs

- Pi0 BDT score is a powerful discriminant:



- The BDT input variables are cluster properties of the pi0 candidates:
- *Path to Pi0 BDT weight file

Cluster pseudorapidity, $|\eta^{\mathrm{clus}}|$
Magnitude of the energy-weighted η position of the cluster

Cluster width, $\langle r^2 \rangle^{\mathrm{clus}}$
Second moment in distance to the shower axis

Cluster η width in EM1, $\langle \eta^2_{\mathrm{EM1}} \rangle^{\mathrm{clus}}$
Second moment in η in EM1

Cluster η width in EM2, $\langle \eta^2_{\mathrm{EM2}} \rangle^{\mathrm{clus}}$
Second moment in η in EM2

Cluster depth, $\lambda^{\mathrm{clus}}_{\mathrm{centre}}$
Distance of the shower centre from the calorimeter front face measured along the shower axis

Cluster PS energy fraction, $f^{\mathrm{clus}}_{\mathrm{PS}}$
Fraction of energy in the PS

Cluster core energy fraction, $f^{\mathrm{clus}}_{\mathrm{core}}$
Sum of the highest cell energy in PS, EM1 and EM2 divided by the total energy

Cluster logarithm of energy variance, $\log\langle \rho^2 \rangle^{\mathrm{clus}}$
Logarithm of the second moment in energy density

Cluster EM1 core energy fraction, $f^{\mathrm{clus}}_{\mathrm{core,EM1}}$
Energy in the three innermost EM1 cells divided by the total energy in EM1

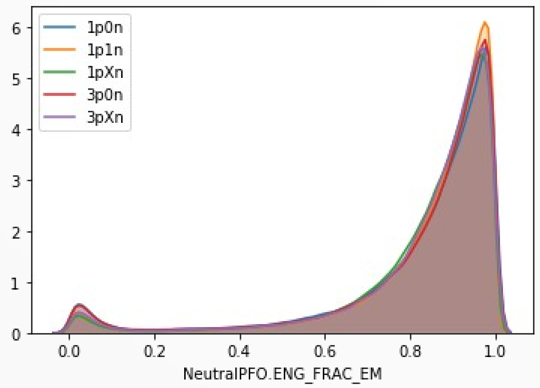
Cluster asymmetry with respect to track, $\mathcal{A}^{\mathrm{clus}}_{\mathrm{track}}$
Asymmetry in η–ϕ space of the energy distribution in EM1 with respect to the extrapolated track position

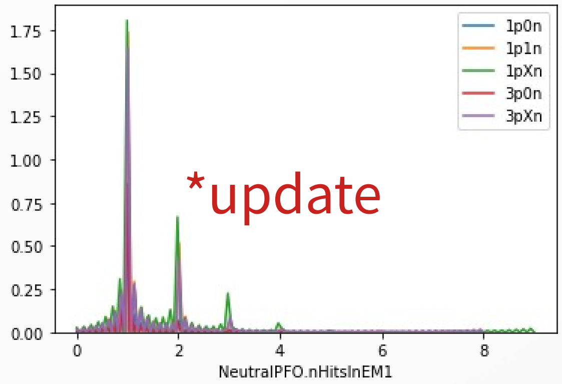
Cluster EM1 cells, $N^{\mathrm{clus}}_{\mathrm{EM1}}$
Number of cells in EM1 with positive energy

Cluster EM2 cells, $N^{\mathrm{clus}}_{\mathrm{EM2}}$
Number of cells in EM2 with positive energy

Pi0 BDT ID score and the inputs

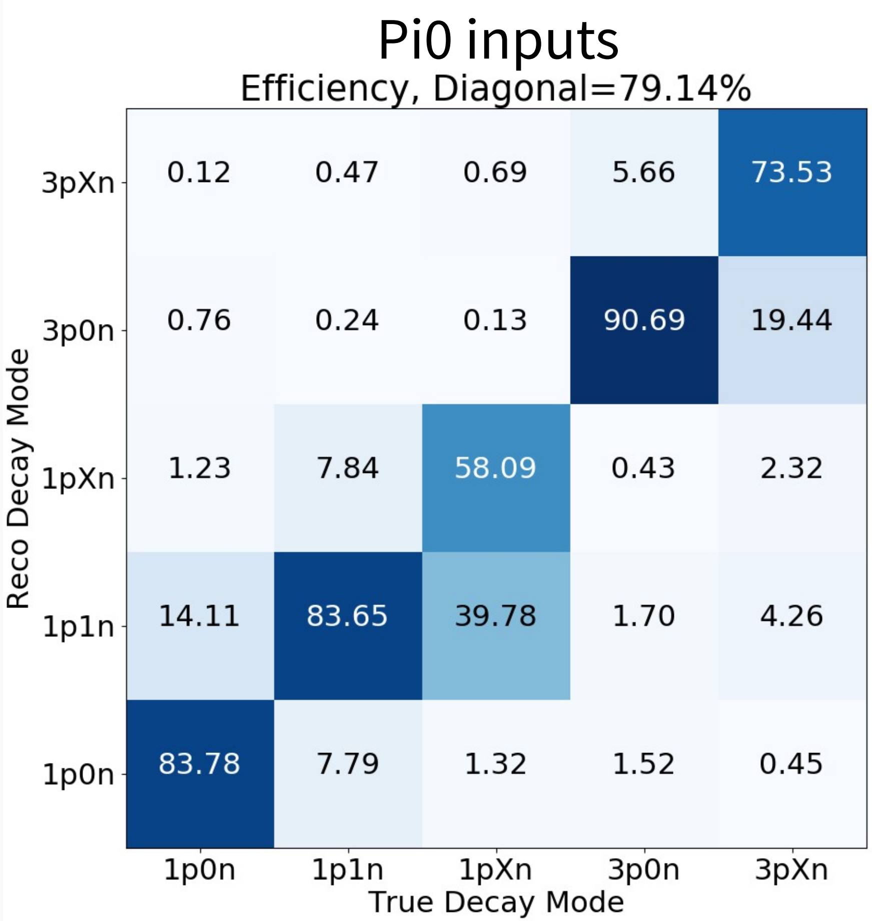
- Replace the pi0 score with the BDT inputs
- Select the important variables of Neutral PFO:
- Using “Boruta” feature selection techniques (scikit-learn)
  - Compare the performance of the ensemble ML algorithms (like BDT or Random Forest) running on the original dataset and the dataset with one variable shuffled.
  - Details: paper, python version
- "NeutralPFO.ENG_FRAC_EM" and "NeutralPFO.nHitsInEM1" are rejected

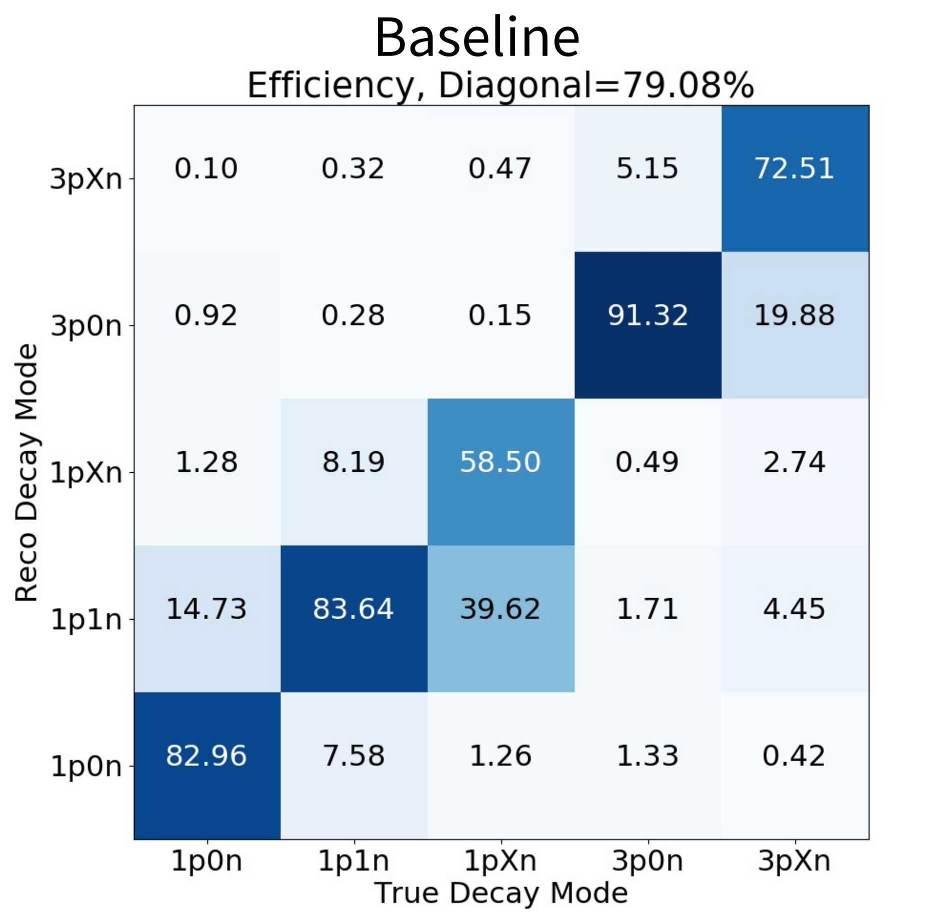




- Remove them in the input.

Pi0 ID inputs performance vs baseline *check more plots later





No obvious difference

Current setup

- Training/Testing with ML packages: JupyterLab provided by https://www.atlas-ml.org/
- *...

Summary

- Start to look at the performance of the RNN classifier
- Checked the impact of input variables:
  - 4-vectors are important
  - Can use pi0 ID inputs instead of pi0 BDT score → might avoid re-training of pi0 BDT
- To-do:
  - Finalize the current study
    - Deal with overtraining, improve baseline model, decide tau pt cut, ...
  - Start to implement in C++ and evaluate performance in THOR
    - convert the model to lwtnn format(?)

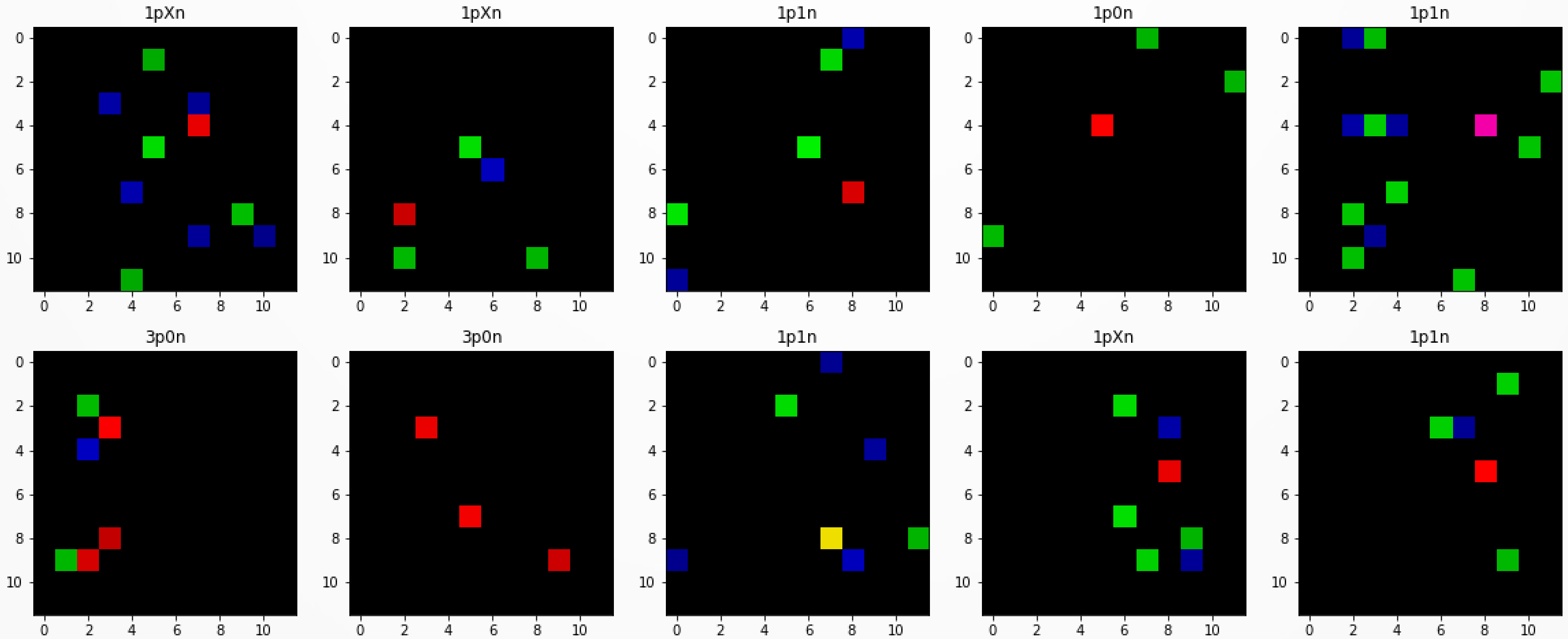
Backup

Selection

```
removed_indices = df[(df["TauJets.truthDecayMode"] > 4) | (df["TauJets.IsTruthMatched"] != 1) |
                     (df["TauJets.pt"] < 20000) | (df["TauJets.truthPtVis"] < 20000) |
                     (df["TauJets.pt"] > 100000) | (df["TauJets.truthPtVis"] > 100000) |
                     (df["TauJets.eta"] > 2.5) | (df["TauJets.truthEtaVis"] > 2.5) |
                     ((df["TauJets.eta"] > 1.37) & (df["TauJets.eta"] < 1.52)) |
                     ((df["TauJets.truthEtaVis"] > 1.37) & (df["TauJets.truthEtaVis"] < 1.52)) |
                     ((df["TauJets.nTracks"] != 1) & (df["TauJets.nTracks"] != 3)) |
                     ((df["TauJets.truthProng"] != 1) & (df["TauJets.truthProng"] != 3))].index
```

Kinematics (4-vectors)

- Or to visualize what the neural network would see (the relative positions in tau axis)

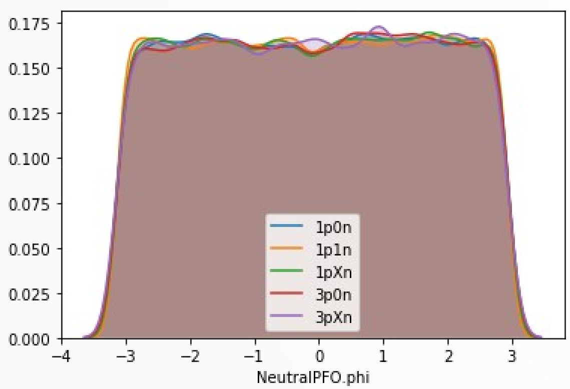


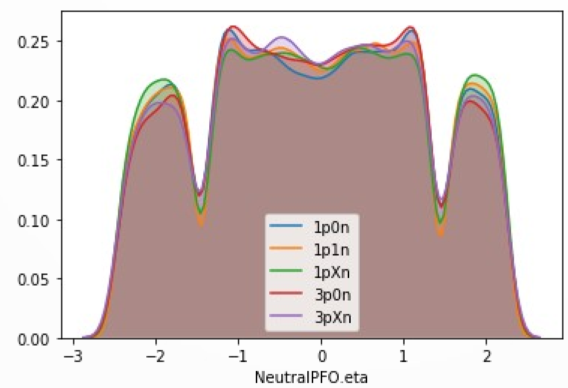
Red = ChargedPFO
Green = NeutralPFO
Blue = ShotPFO

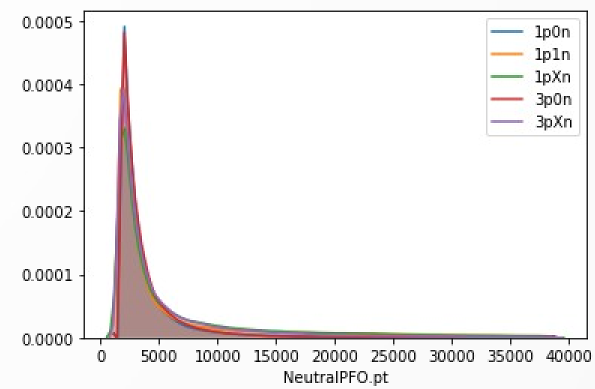
Neutral PFO vars

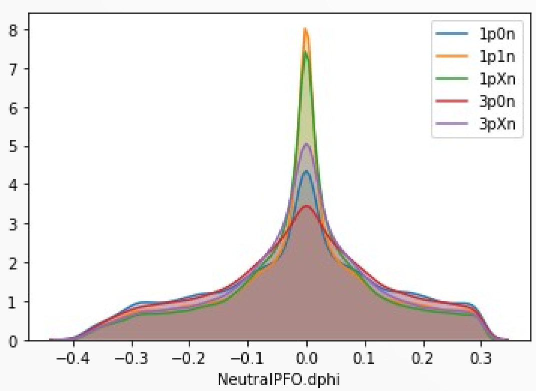
"NeutralPFO.FIRST_ETA",
"NeutralPFO.SECOND_R",
"NeutralPFO.DELTA_THETA",
"NeutralPFO.CENTER_LAMBDA",
"NeutralPFO.LONGITUDINAL",
"NeutralPFO.SECOND_ENG_DENS",
"NeutralPFO.ENG_FRAC_EM",
"NeutralPFO.ENG_FRAC_CORE",
"NeutralPFO.NPosECells_EM1",
"NeutralPFO.NPosECells_EM2",
"NeutralPFO.nHitsInEM1",
"NeutralPFO.ptSubRatio",
"NeutralPFO.energyfrac_EM2",
"NeutralPFO.EM1CoreFrac",
"NeutralPFO.secondEtaWRTClusterPosition_EM1",
"NeutralPFO.firstEtaWRTClusterPosition_EM1",
"NeutralPFO.secondEtaWRTClusterPosition_EM2",

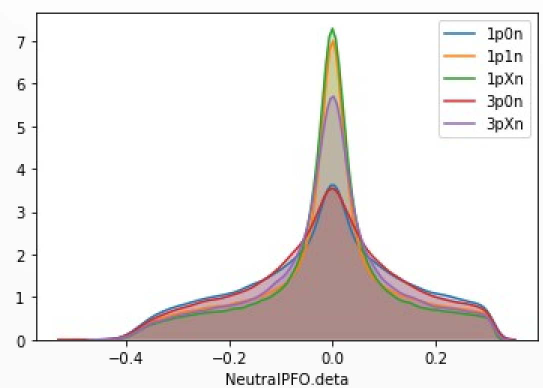
Input vars

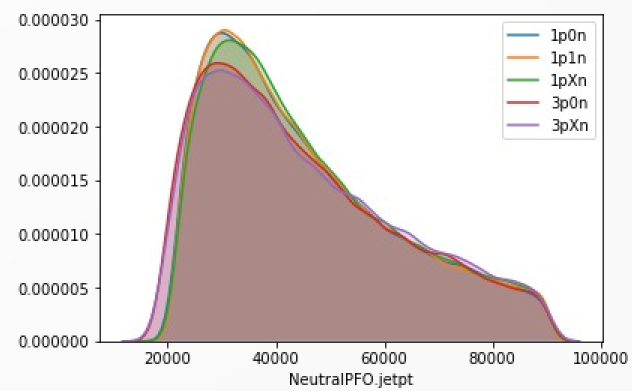

0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
−4 −3 −2 −1 0 1 2 3
NeutralPFO.phi
1p0n
1p1n
1pXn
3p0n
3pXn


0.25
0.20
0.15
0.10
0.05
0.00
−3 −2 −1 0 1 2
NeutralPFO.eta
1p0n
1p1n
1pXn
3p0n
3pXn

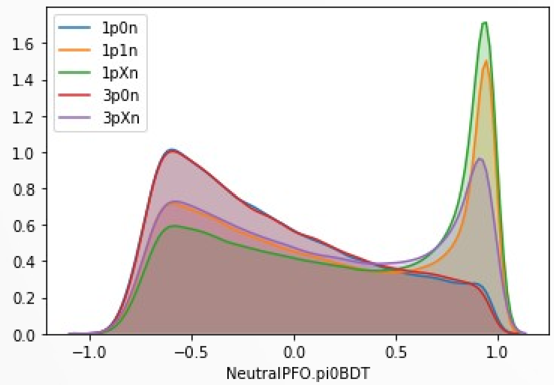

0.0005
0.0004
0.0003
0.0002
0.0001
0.0000
0 5000 10000 15000 20000 25000 30000 35000 40000
NeutralPFO.pt
1p0n
1p1n
1pXn
3p0n
3pXn

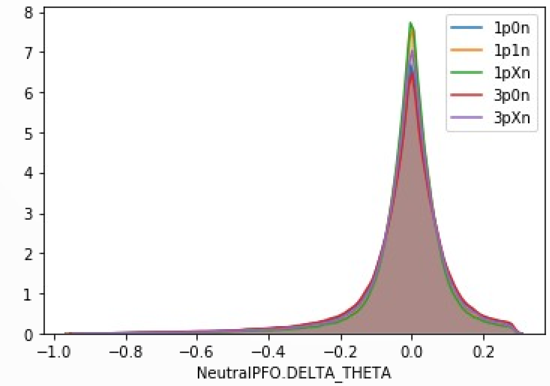

8
7
6
5
4
3
2
1
0
−0.4 −0.3 −0.2 −0.1 0.0 0.1 0.2 0.3
NeutralPFO.dphi
1p0n
1p1n
1pXn
3p0n
3pXn

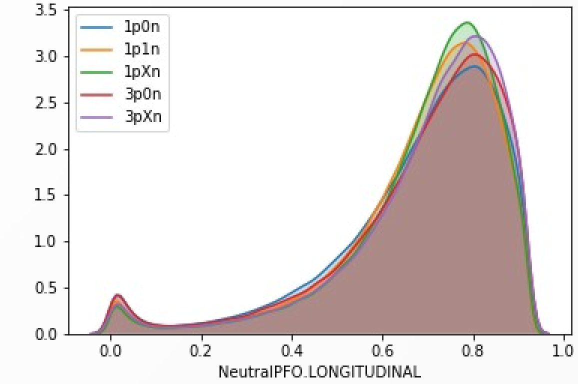

7
6
5
4
3
2
1
0
−0.4 −0.2 0.0 0.2
NeutralPFO.deta
1p0n
1p1n
1pXn
3p0n
3pXn

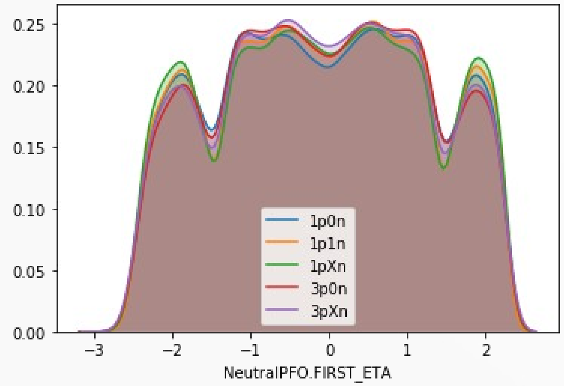

0.000030
0.000025
0.000020
0.000015
0.000010
0.000005
0.000000
20000 40000 60000 80000 100000
NeutralPFO.jetpt
1p0n
1p1n
1pXn
3p0n
3pXn

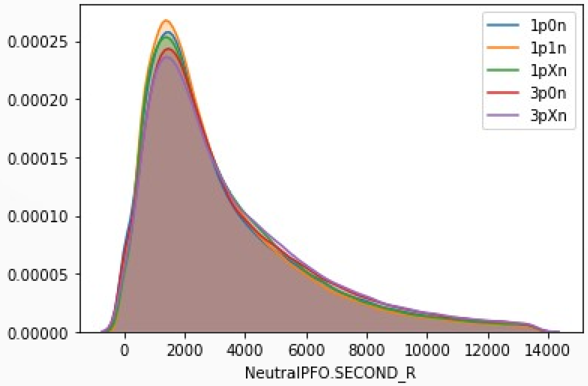
Input vars

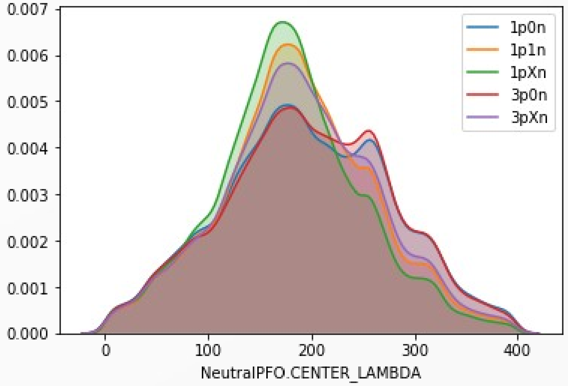

1p0n
1p1n
1pXn
3p0n
3pXn
NeutralPFO.pi0BDT


1p0n
1p1n
1pXn
3p0n
3pXn
NeutralPFO.DELTA_THETA

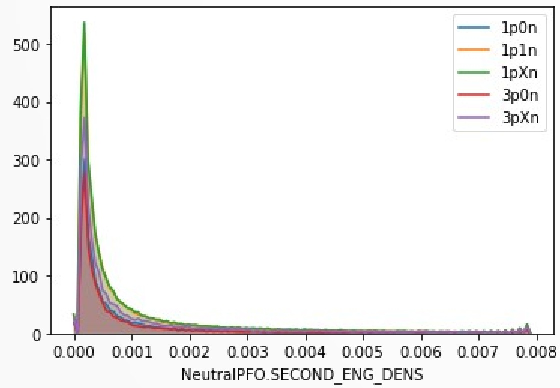

1p0n
1p1n
1pXn
3p0n
3pXn
NeutralPFO.LONGITUDINAL

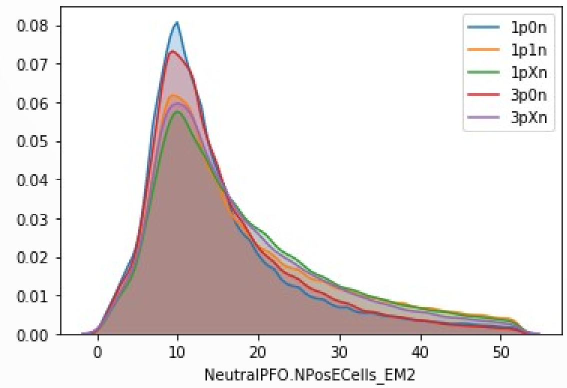

1p0n
1p1n
1pXn
3p0n
3pXn
NeutralPFO.FIRST_ETA

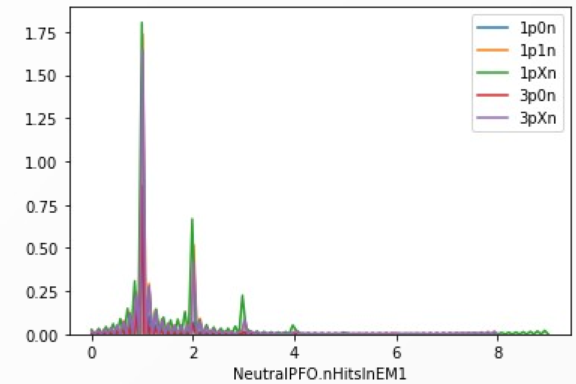

1p0n
1p1n
1pXn
3p0n
3pXn
NeutralPFO.SECOND_R

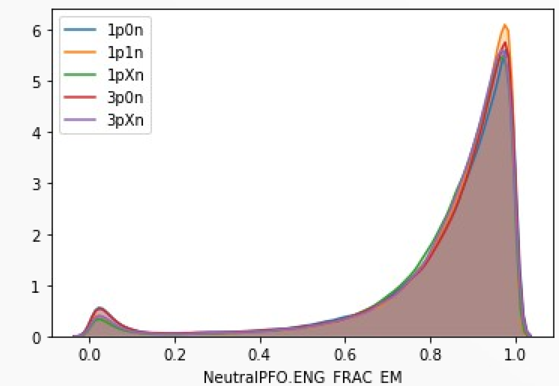

1p0n
1p1n
1pXn
3p0n
3pXn
NeutralPFO.CENTER_LAMBDA

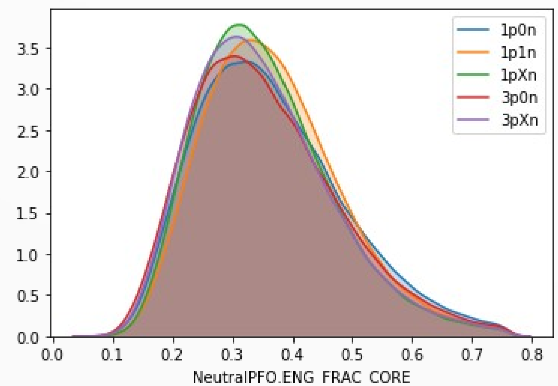
Input vars

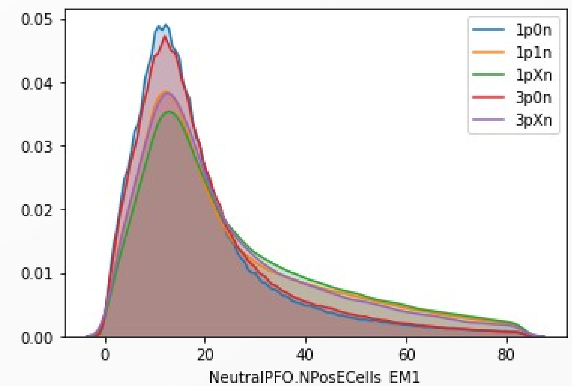

1p0n
1p1n
1pXn
3p0n
3pXn
NeutralPFO.SECOND_ENG_DENS


1p0n
1p1n
1pXn
3p0n
3pXn
NeutralPFO.NPosECells_EM2


1p0n
1p1n
1pXn
3p0n
3pXn
NeutralPFO.nHitsInEM1


1p0n
1p1n
1pXn
3p0n
3pXn
NeutralPFO.ENG_FRAC_EM


1p0n
1p1n
1pXn
3p0n
3pXn
NeutralPFO.ENG_FRAC_CORE


1p0n
1p1n
1pXn
3p0n
3pXn
NeutralPFO.NPosECells_EM1

Pi0 ID inputs performance