τ Decay Mode Classification with Neural Network

Bowen Zhang

NJU τ meeting

30.03.2020

Introduction

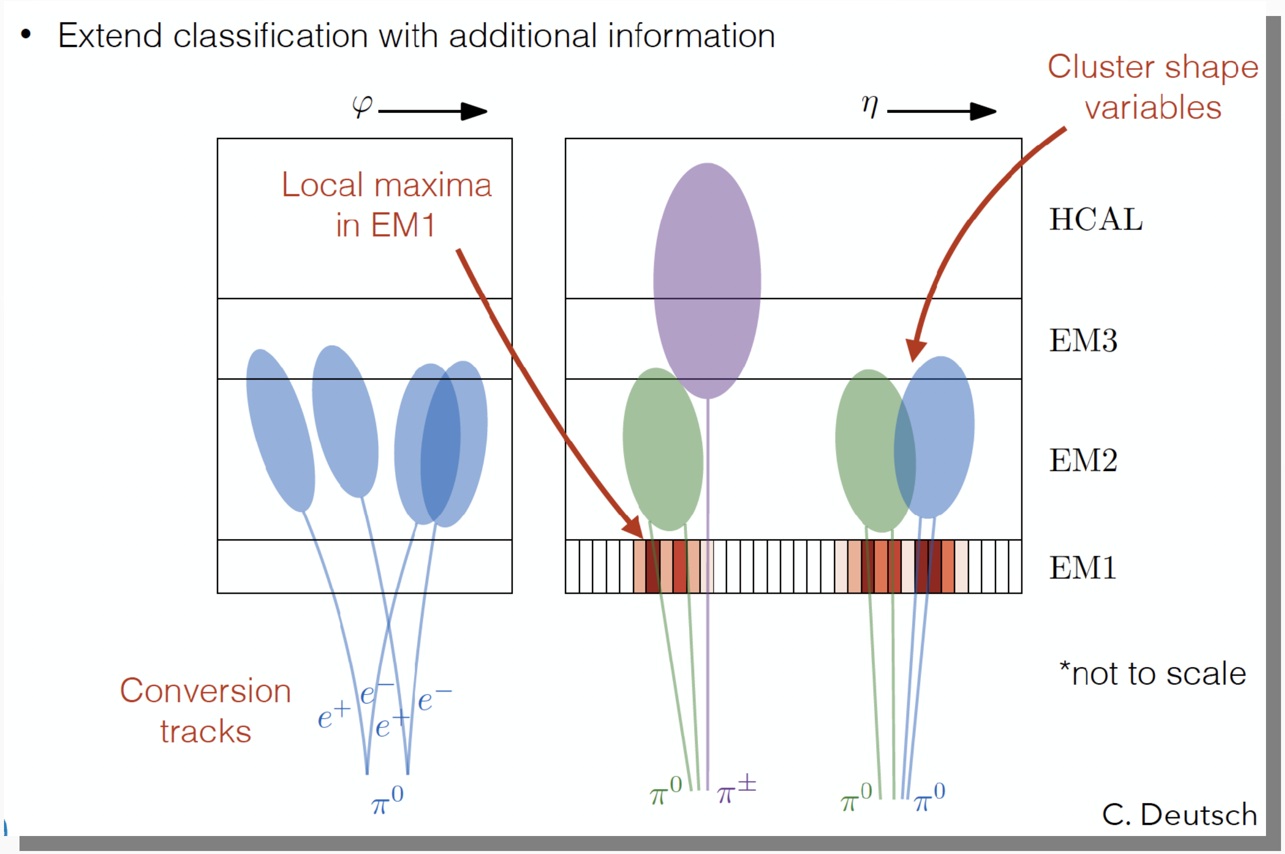
- Improve the five-way τ decay mode classification using machine learning techniques:
- Categorized by the number of charged and neutral products
- Previously use kinematics of the decay products, the identification score of pi0 and the number of photons

Decay mode test	$N(\pi^0_{\mathrm{cand}})$	$N(\pi^0_{\mathrm{ID}})$	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}, $\sigma_{E_{\mathrm{T}},h^\pm}$, $m_{h^\pm}$, N_γ

arxiv:1512.05955

Introduction

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

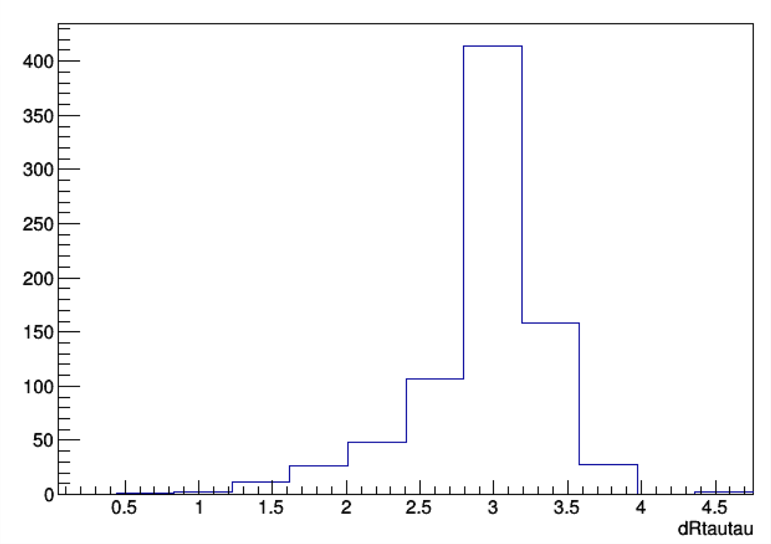


Recap

- Last presentation:
  - Replace Pi0 BDT score by the Pi0 BDT ID inputs → gain the same performance
  - Show rough performance plots with python package
- Remaining task:
  - Overfitting issue
  - Impact of pt and tau ID requirements
  - C++ implementation

Sample

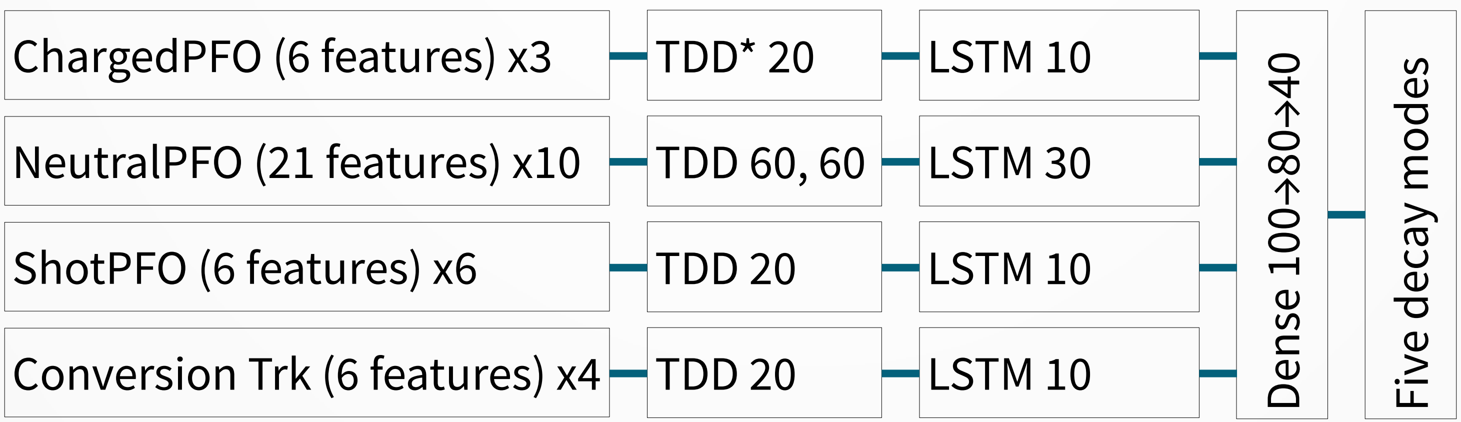
- Still using mc16d Gammatautau
  - mc16_13TeV:mc16_13TeV.425200.Pythia8EvtGen_A14NNPDF23LO_Gammatautau_MassWeight.merge.AOD.e5468_s3126_r10201_r10210
- 4M for training (20% of it for validation) 4M for testing (split by mcEventNumber)
- pt selection:
  - Last time: pt < 100 GeV, **this time: pt < 300 GeV**



DeltaR(tau0, tau1) in Gammatautau sample

RNN Model

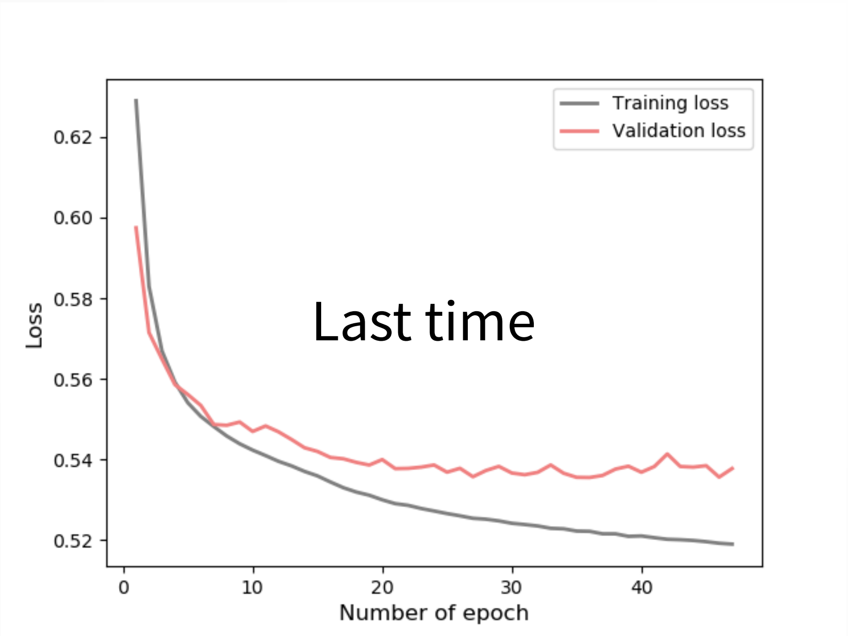
- Simplify the model. Number of parameters is reduced by ~ a factor of 3. (Now ~40k) → overfit mitigates (see next)

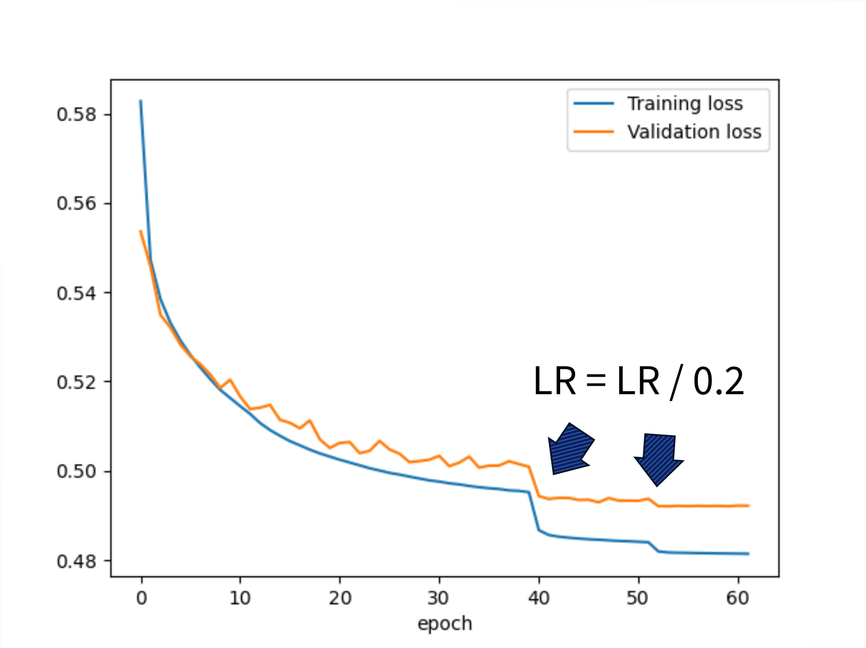


;; Number of nodes roughly optimised by RandomSearch using KerasTuner package

*TDD= TimeDistributed+Dense

Overfitting issue





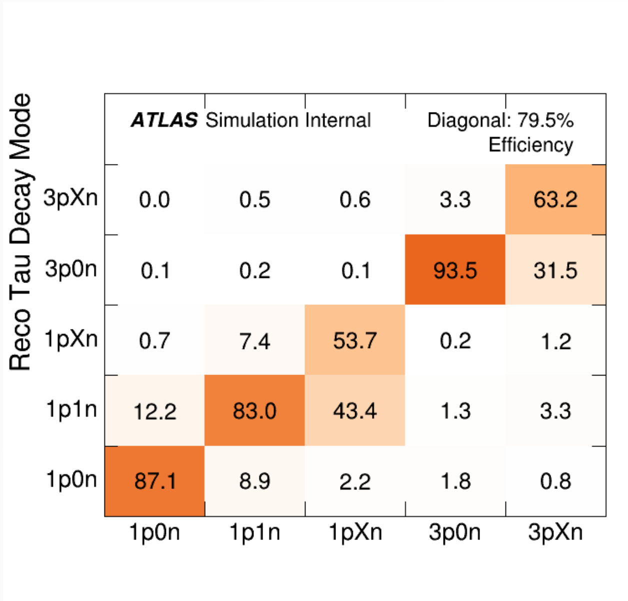
Conclusion:

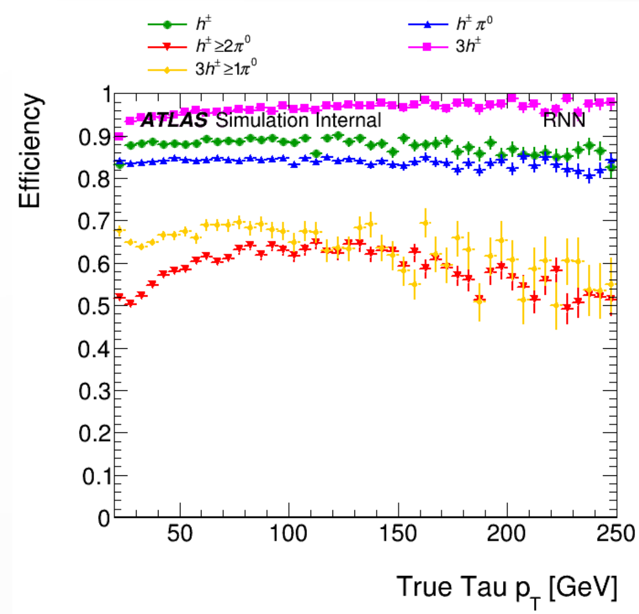
- The overfitting issue was mitigated and the validation performance improved a little.
- Further regularization method (Cross validation, drop out, …) and learning rate schedule can still be considered.

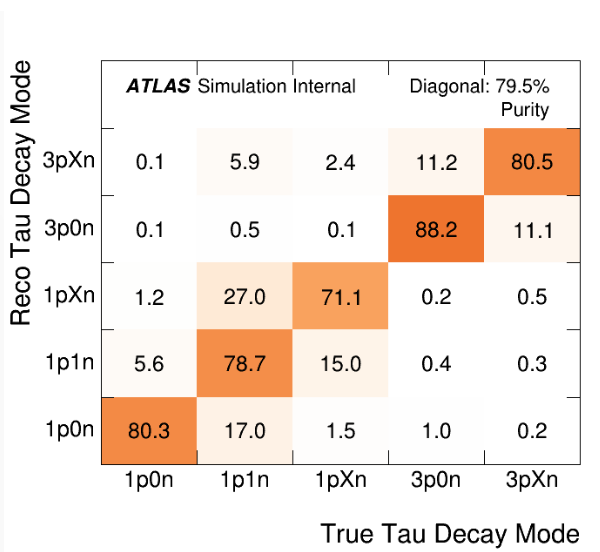
Evaluation

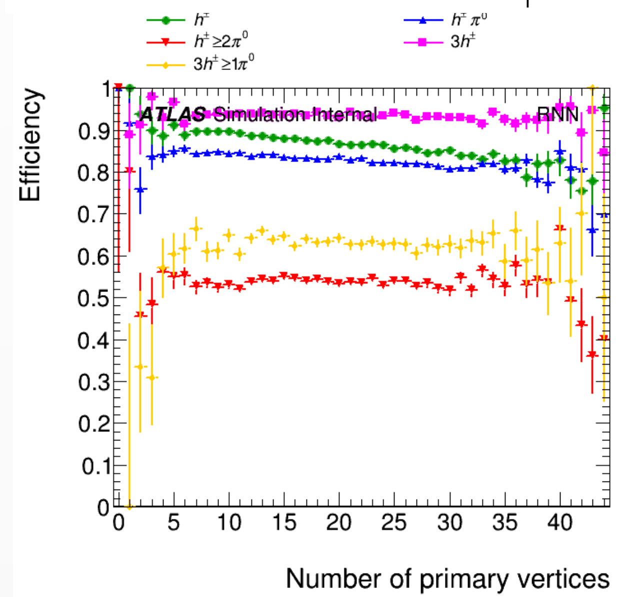
- The networks are converted and processed by LWTNN.
- The performance are evaluated by THOR/LOKI.
- THOR:
  - DecayModeClassifier (EnsureTrackConsistency=T)
  - NN Classifier prediction must be consistent with tau prongness. Avoid transition between 1-track and 3-track taus
- LOKI:
  - Same setup as PanTau PlotBook except some baseline cuts.
  - taus.pt10Truth → taus.pt20Truth, add truth match kinematic (eta25Truth, vetoCrackTruth)
  - No maximum pt requirement, ID WP = BDT medium

Performance of RNN



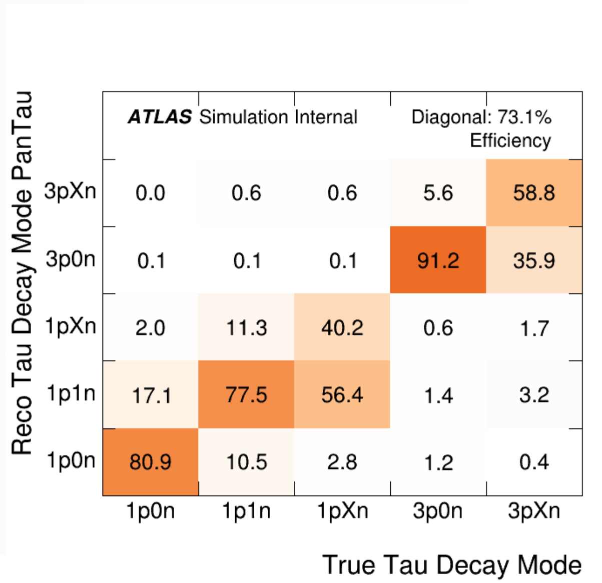


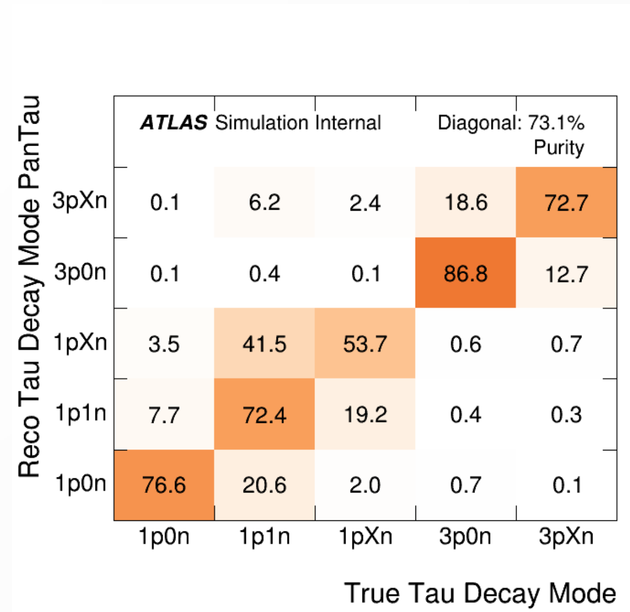




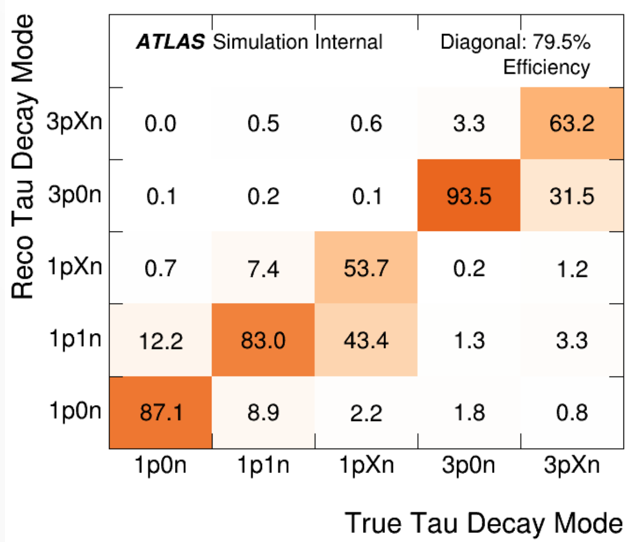
Performance of RNN vs BDT

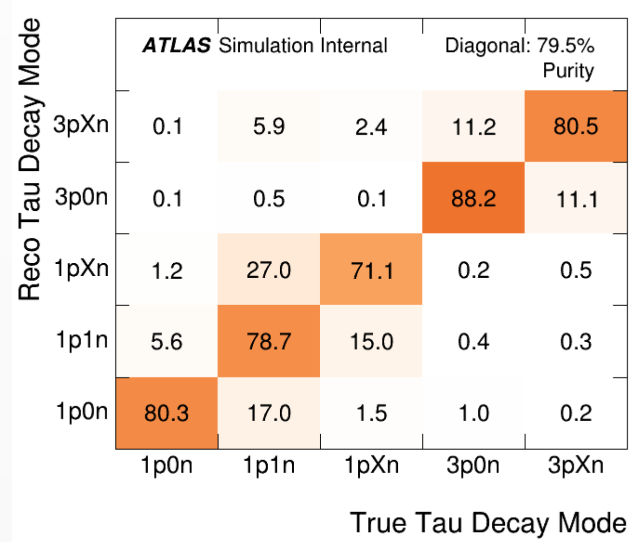
BDT



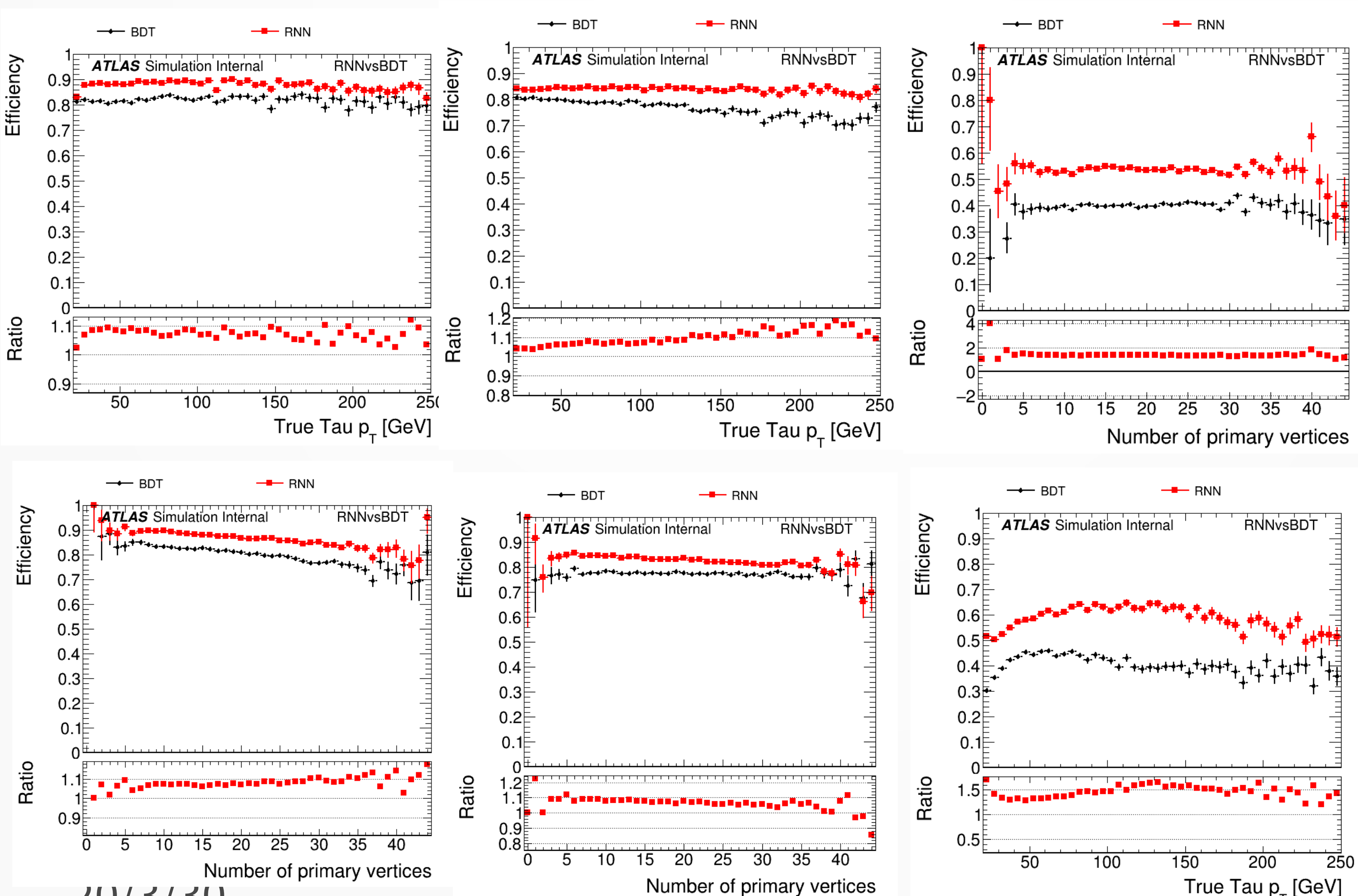


RNN



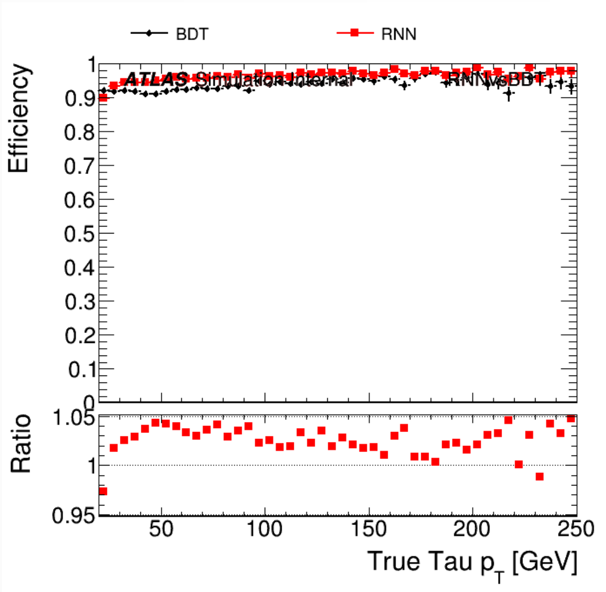


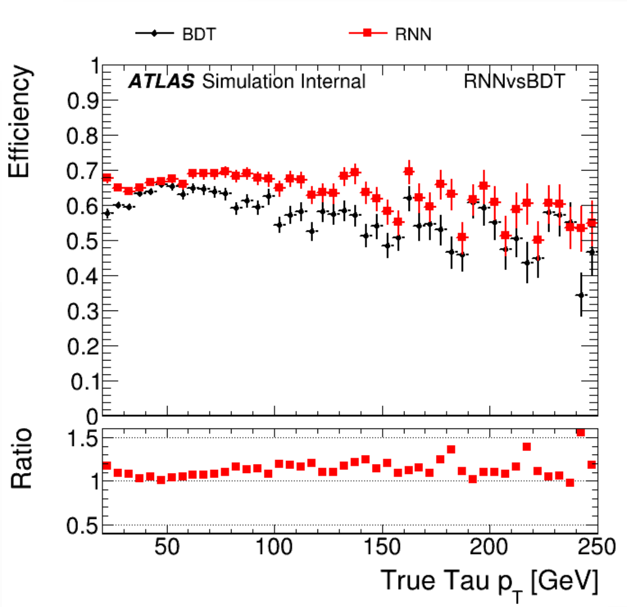
Performance of RNN vs BDT

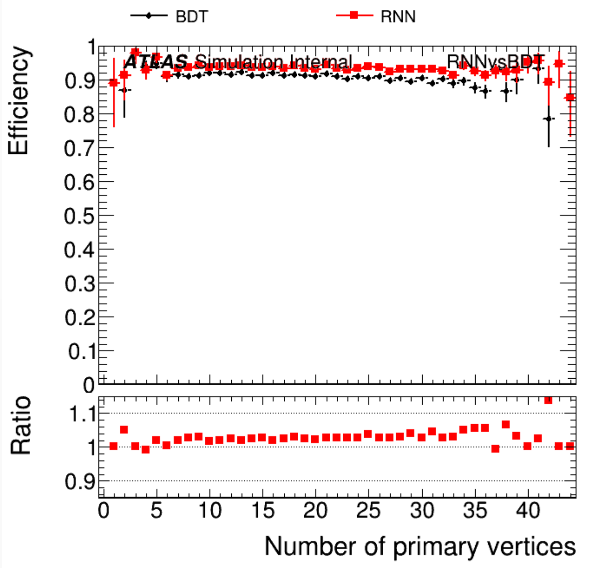


1p0n 1p1n 1pXn

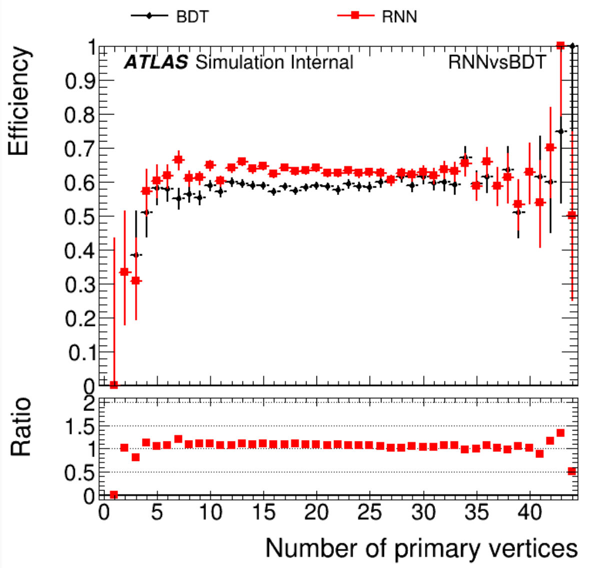
Performance of RNN vs BDT







3p0n



3pXn

Deep Sets: Brief Intodoction

- Developed by Machine learning community:
  - Paper | GitHub
- To handle to case when the inputs (or outputs) are permutation invariant and have variable size.
- Collision events have the similar properties.
- Current algorithms do not work well with them:
  - DNNs: inputs have fixed size.
  - RNNs: have to define the order of the sequences
- Deep Sets in HEP
  - Energy Flow Networks: Paper
  - Jet flavour tagging: RNNIP

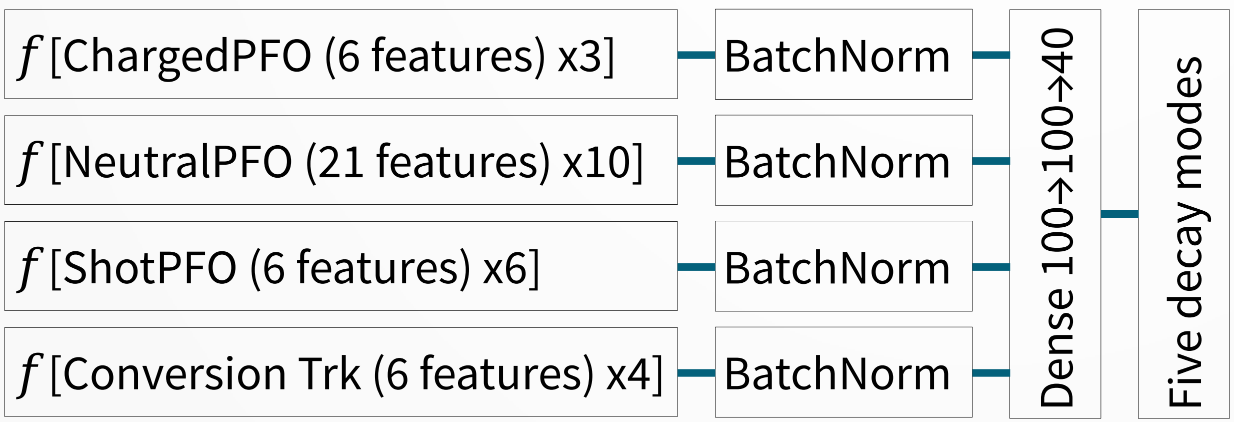
Deep Sets: Invariant model

- Theorem:
  - A function f(X) operating on a set X is invariant to the permutation of instances in X, iff it can be decomposed in the form $\rho(\sum\phi(x))$, summing over element x in set X, with suitable transformation ϕ and ρ.
- Network Architecture
  - x → representation $\phi(x)$ by ϕ network.
  - The representations are added up (SumLayer)
  - Then the sum is processed by ρ network.

There are also equivariant models,. in which all layers are equivariant to the permutations of x. Here only consider invariant model.

Deep Sets Model

- Each PFO/track set is represented by $f(x)=\rho(\sum\phi(x))$, then passed to a batch normalisation layer and then merged.
- ~45k parameters in total. Training history much similar like RNN.

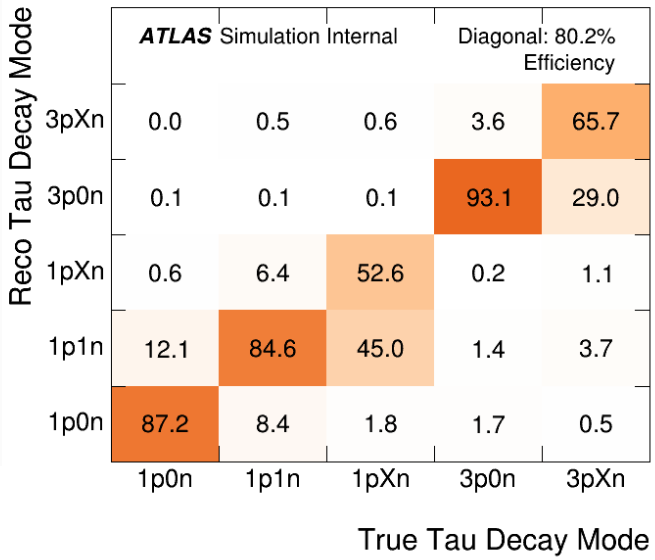


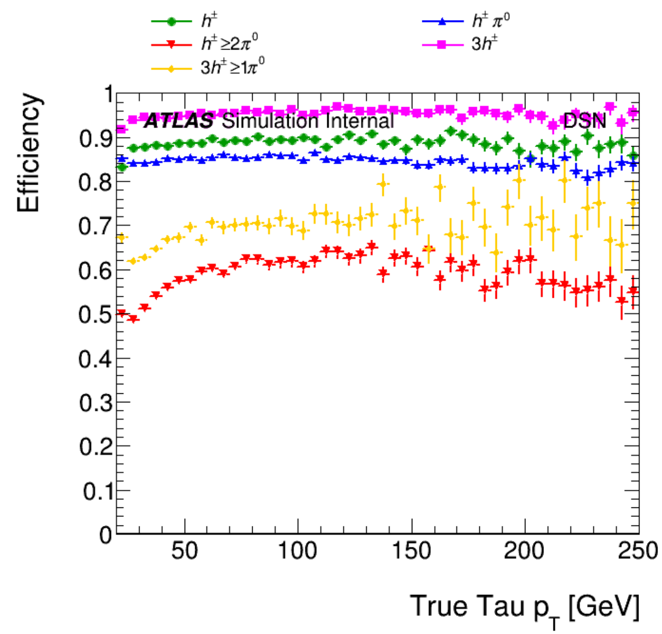
nodes	ϕ=TDD	ρ=Dense
Charged	20,20,20	20,20,20
Neutral	80,80,60,60	60,60,40
Shot	20,20,20	20,20,20
ConvTrk	20,20,20	20,20,20

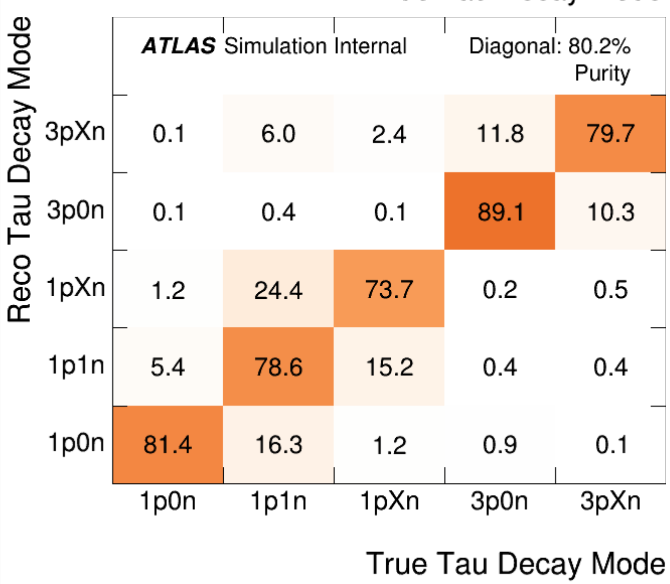
;; Number of nodes roughly optimised by RandomSearch using KerasTuner package

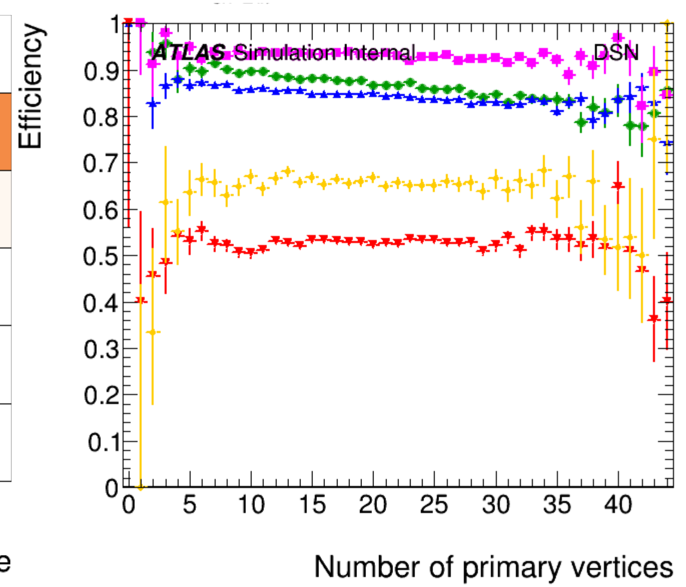
TDD= TimeDistributed+Dense

Performance of DSN

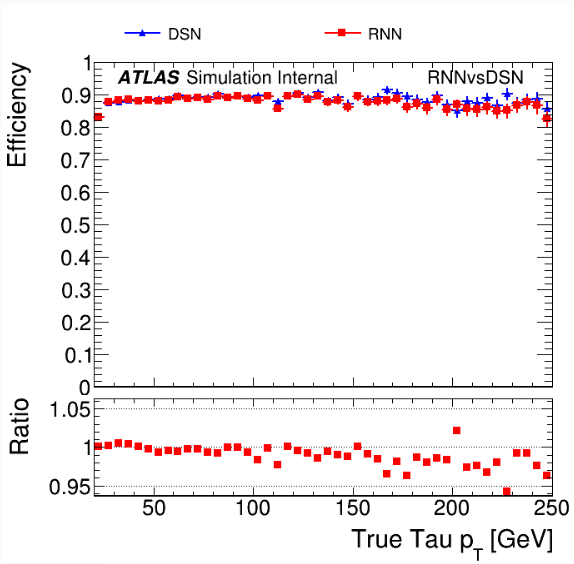


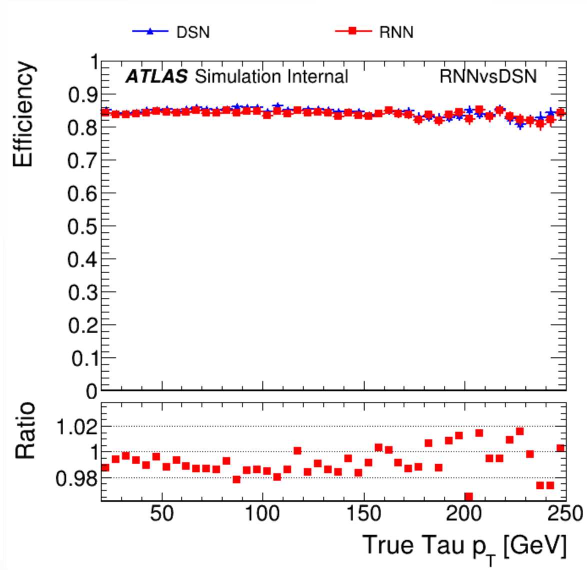


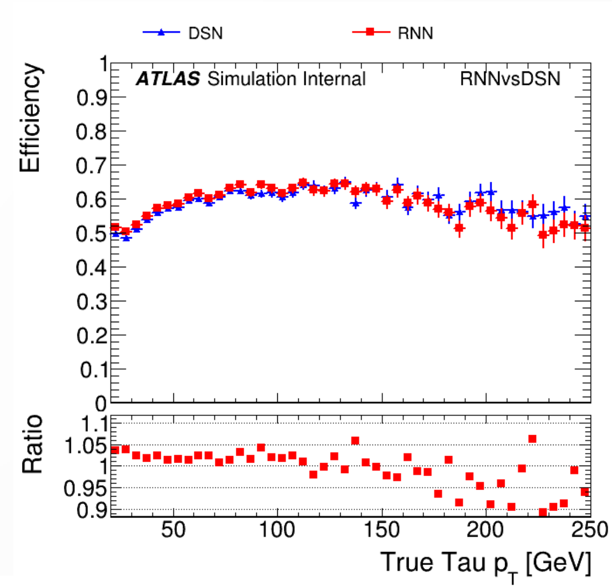


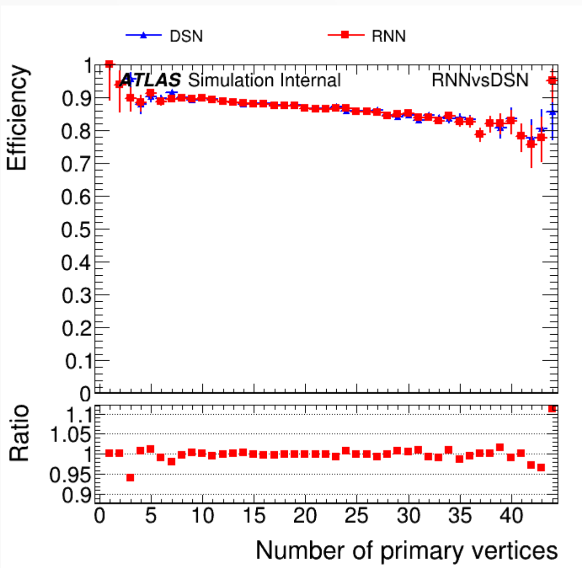


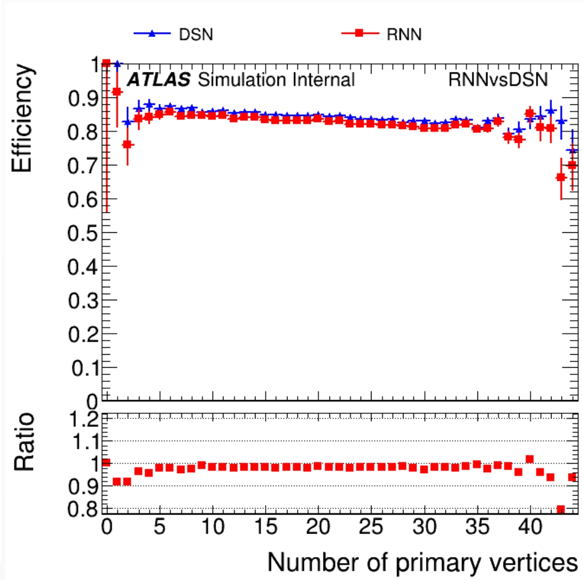
Performance of DSN vs RNN

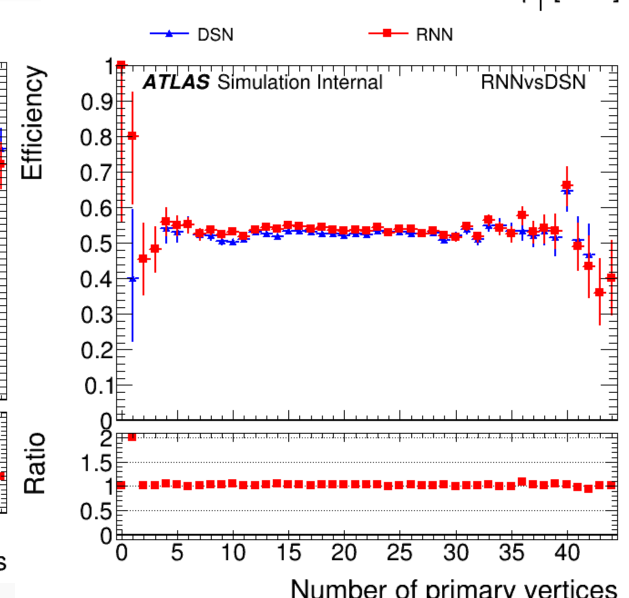






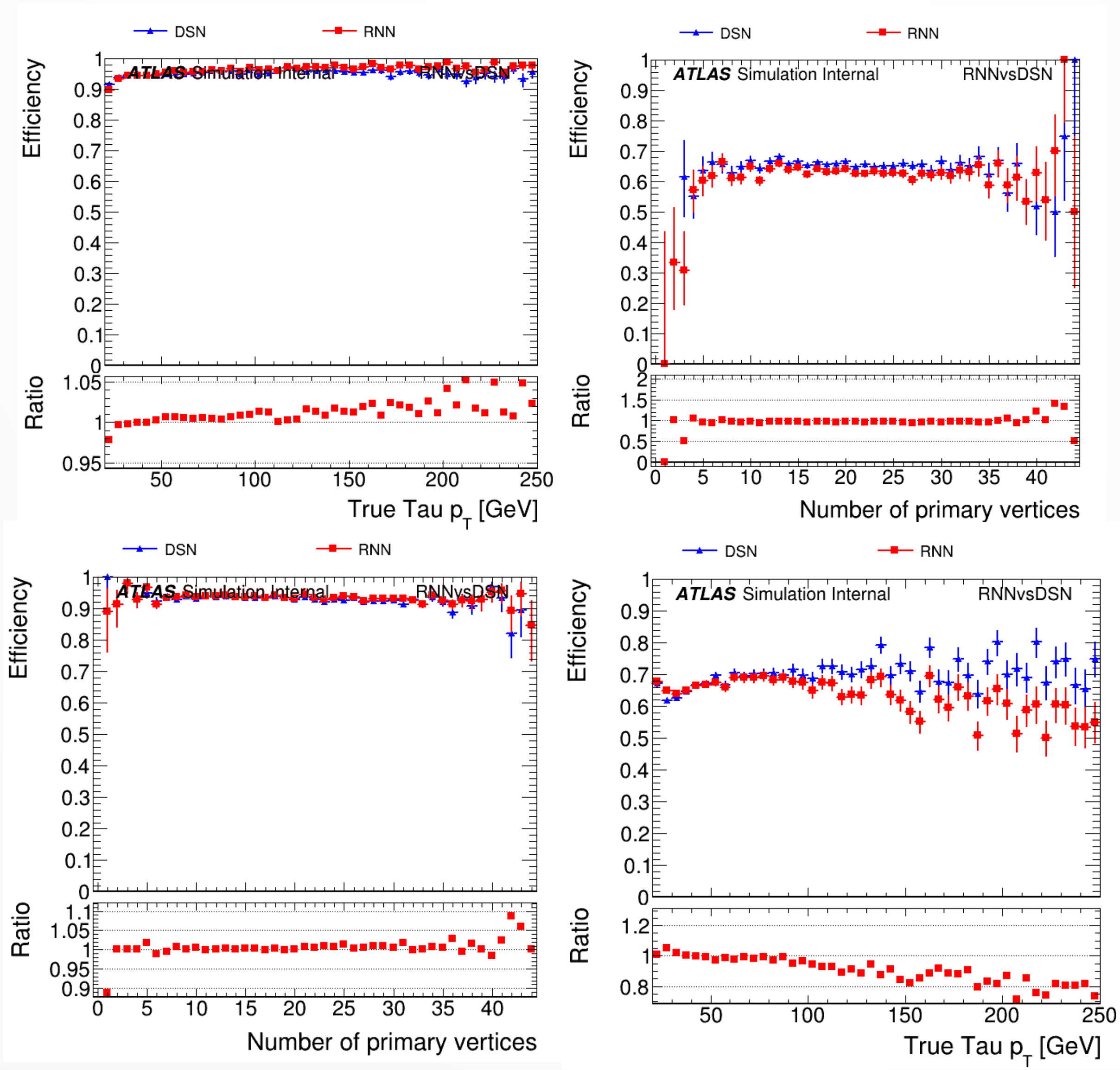






1p0n 1p1n 1pXn

Performance of DSN vs RNN



3p0n 3pXn

Summary

- Overtraining issue:
  - Mitigated by simplifying the model and tuning the LR schedule
    - The performance doesn’t get worse.
    - Better than last time but still need some investigate.
- Rough hyper-parameter scan was attempted for RNN (and DSN)
- Evaluate on medium BDT ID taus shows better result
- Can use up to 300GeV taus for training (or remove the pt requirement?)
- Performance are evaluated using THOR/LOKI. Networks are inferred by LWTNN.
- RNN is compared with BDT:
  - Better performance for all mode
- RNN is compared with DSN:
  - Consistent behaviour.
  - DSN slightly outperforms. Needs double-check.

Backup