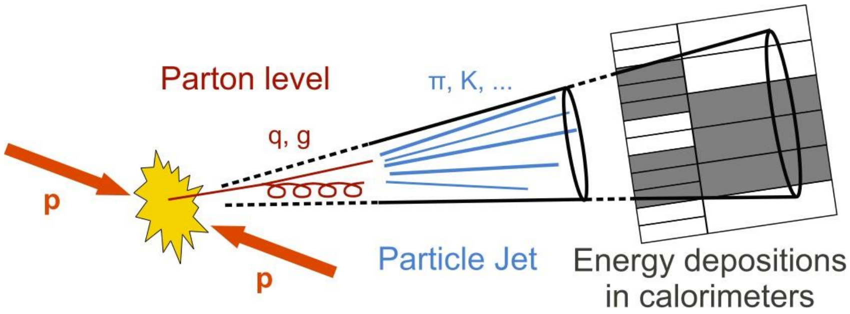
Identification of Jets Containing b-Hadrons with Recurrent Neural Networks on Trigger Jets

Zhongyukun Xu
Shandong University

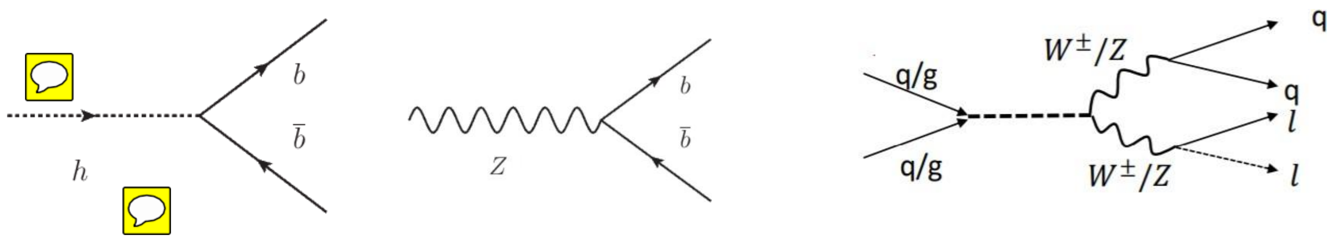
Outline

- Introduction
- RNNIP algorithm
- RNNIP implementation
- MV2 with RNNIP
- Summary

Introduction

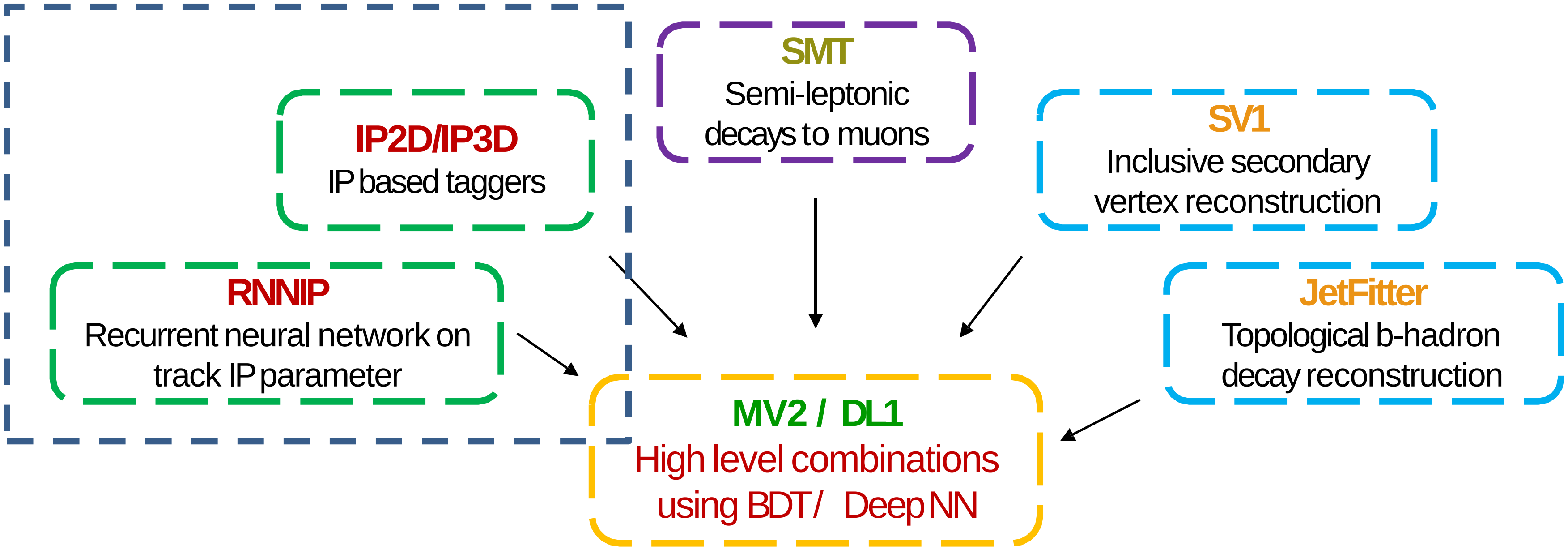


- A jet is a narrow cone of hadrons and other particles produced by the hadronization of a quark or gluon in a particle physics or heavy ion experiment.
- Jets are divided into different types according to the hadron type they contain. (b-jet, c-jet, tau-jet, light-jet)



- Many physics process have b-jet involvement, so it is vital to do b-tagging to distinguish them from other jets (c-jet, light jet)
- We use different b-tagging algorithms to tag b-jet.

B-tagging algorithm



Impact-parameter based (IP) algorithm: IP2D/IP3D, RNNIP
Secondary-vertex (SV) based algorithm: SV1,JetFitter
Soft muon tagger algorithm: SMT
Multi-variable (MV) algorithm: MV2, MV2mu, MV2rnn
Deep learning algorithm: DL1

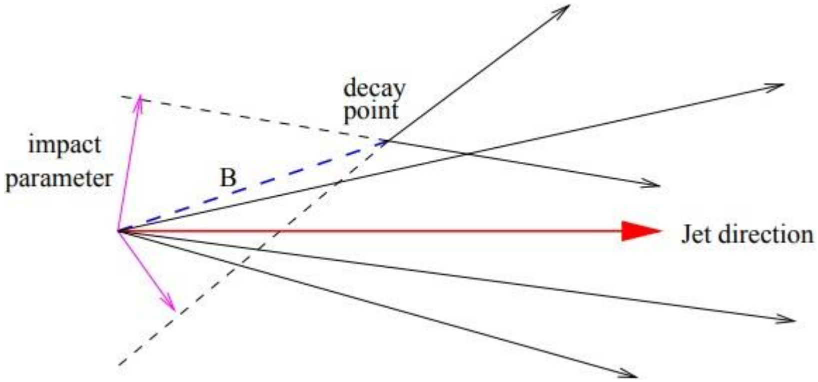
Impact Parameter

IP calculation:

1、The transverse impact parameter d0 is the distance of closest approach of the track to the primary vertex point in the rφ projection.

2 、 The z coordinate of the track at this point of closest approach is referred to as z0(longitudinal impact parameter). *

ATLAS-CONF-2011-102



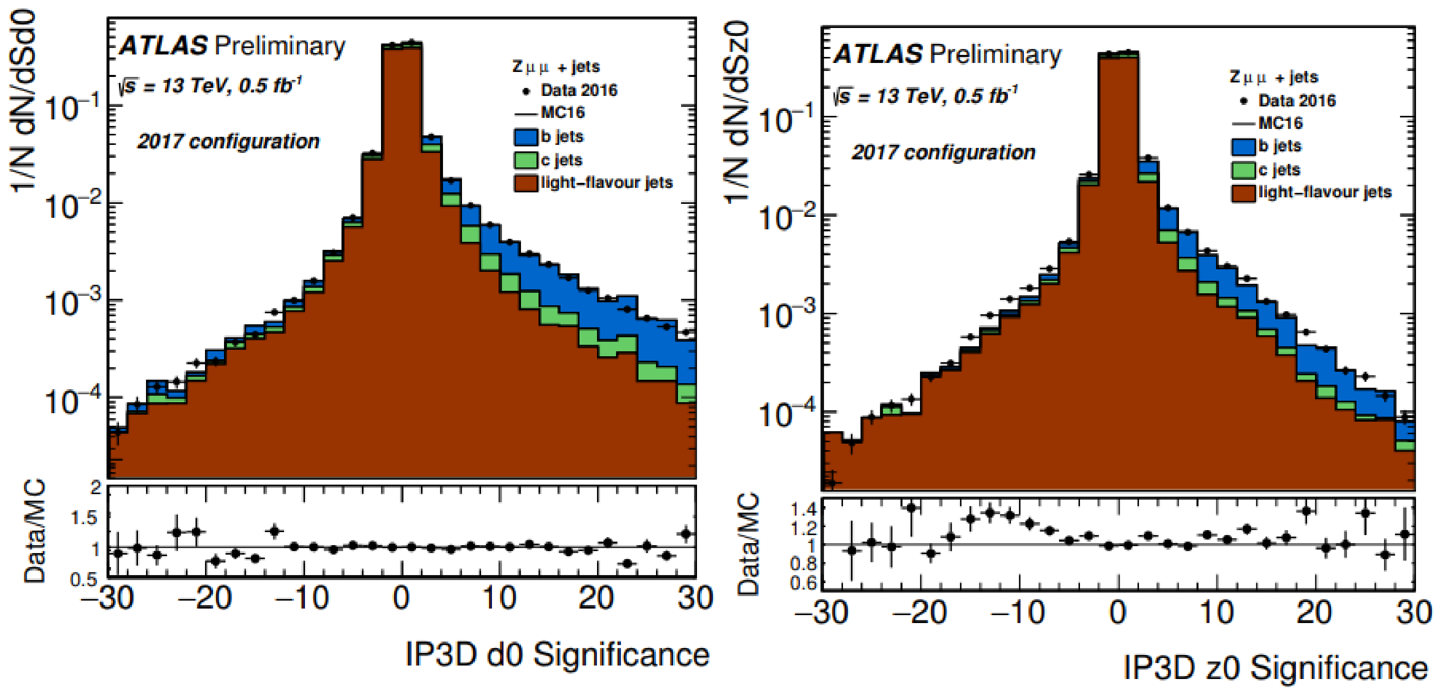
But to take into account the experimental resolution, the track impact parameter significance, the ratio between the track impact parameter and its uncertainty, is introduced.

$$S_{d_0} = d_0/\sigma(d_0)$$

which is called IP significance.

*Strictly speaking the longitudinal impact parameter is |z0|sinθ where θ is the polar angle of the track.

IP significance



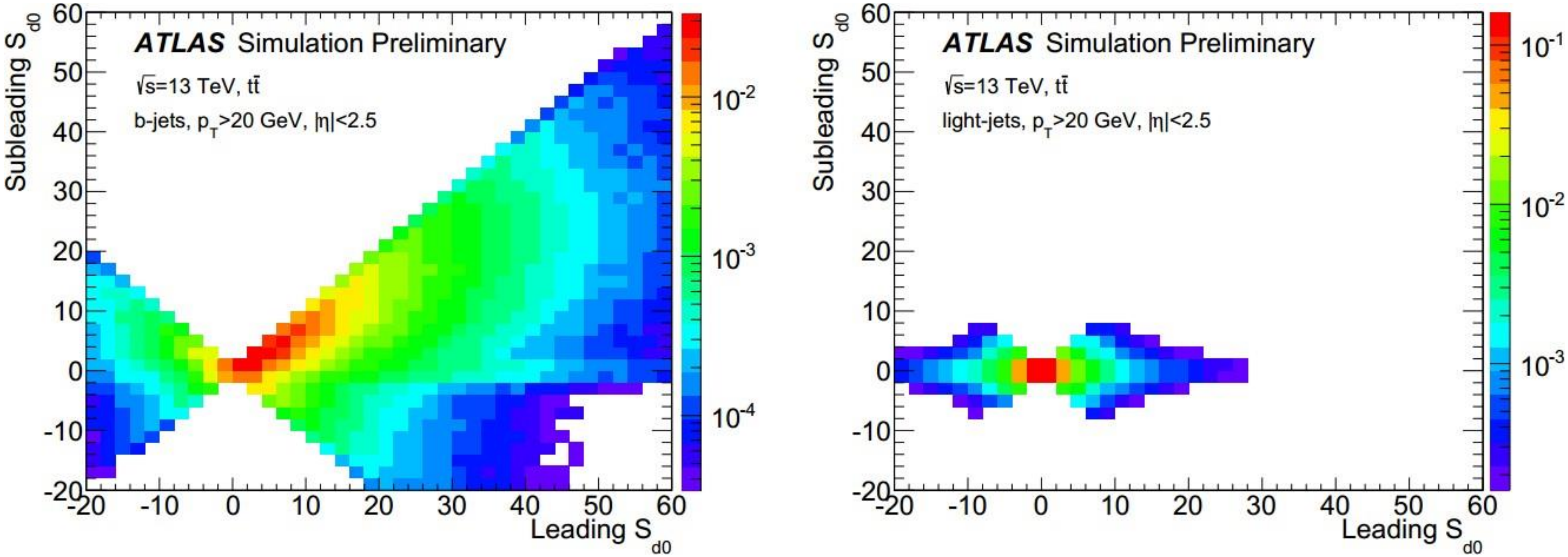
The b-jet distribution shows an asymmetric tail on positive values due to long lifetime. IP2D/IP3D identify b-jets based on impact parameter distributions

ATL-PHYS-SLIDE-2017-219

Motivation for RNN

IP2D/IP3D algorithm do not consider the correlation between tracks.

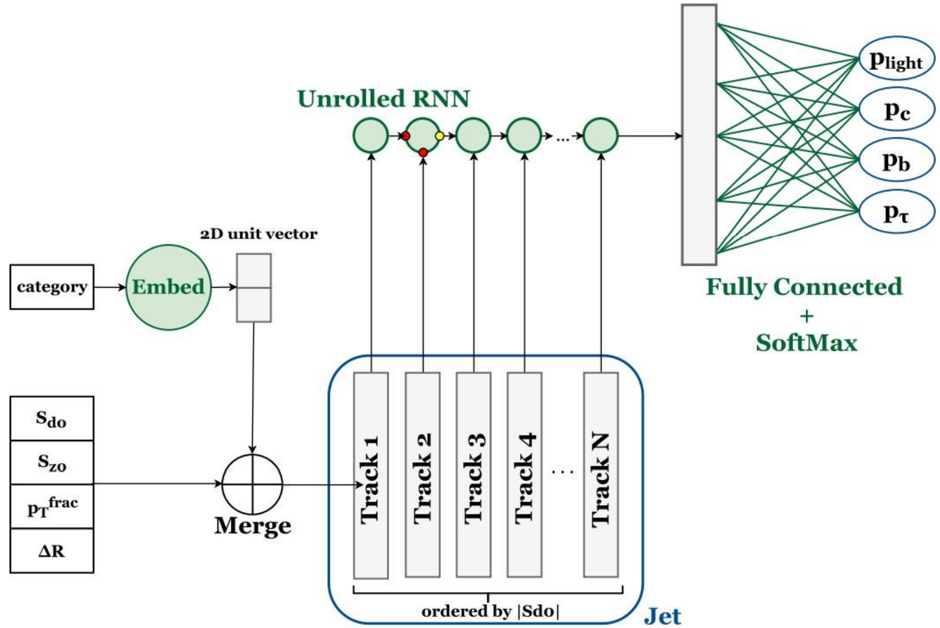
For b-jet, we find track impact parameters are intrinsically correlated: If one track is found with a large impact parameter, then the chances that finding a second track with large impact parameter are also very large. If no displaced decay is present, like in light-flavor jets, such correlations should not exist.



So recurrent neural network is introduced to make full use of the track impact parameter correlation.

RNNIP algorithm

Recurrent neural networks are used to directly learn sequential dependencies for arbitrary-length sequences. The fundamental unit of an RNN is a cell encapsulating an internal state vector.



To fully use the track IP information, we put following track information to be trained by RNN:

- IP3D significance(S_{d0},S_{z0})
- $p_T^{frac}(\mathrm{pT}_{track}/\mathrm{pT}_{jet})$
- ΔR (between track and jet)

$$\sqrt{(\phi_{\text{track}} - \phi_{\text{jet}})^2 + (\eta_{\text{track}} - \eta_{\text{jet}})^2}$$

- grade: track category defined by hits ➡

#	Description	Rate [%] *b* jets	Rate [%] *c* jets	Rate [%] light jets
0	No hits in first two layers; exp. hit in L0 and L1	1.5	1.6	1.6
1	No hits in first two layers; exp. hit in L0 and no exp. hit in L1	0.1	0.1	0.1
2	No hits in first two layers; no exp. hit in L0 and exp. hit in L1	0.03	0.03	0.03
3	No hits in first two layers; no exp. hit in L0 and L1	0.03	0.03	0.02
4	No hit in L0; exp. hit in L0	2.4	2.3	2.1
5	No hit in L0; no exp. hit in L0	0.9	0.9	0.9
6	No hit in L1; exp. hit in L1	0.5	0.5	0.5
7	No hit in L1; no exp. hit in L1	2.4	2.4	2.3
8	Shared hit in both L0 and L1	0.01	0.01	0.04
9	Shared pixel hits	2.1	1.6	1.8
10	Two or more shared SCT hits	2.4	2.2	2.2
11	Split hits in both L0 and L1	1.2	1.1	0.8
12	Split pixel hit	2.1	1.6	1.1
13	Good: a track not in any of the above categories	84.3	85.5	86.6

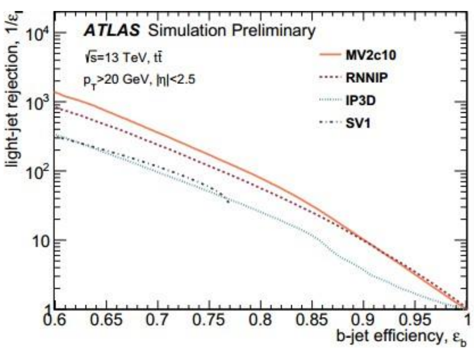
RNNIP performance

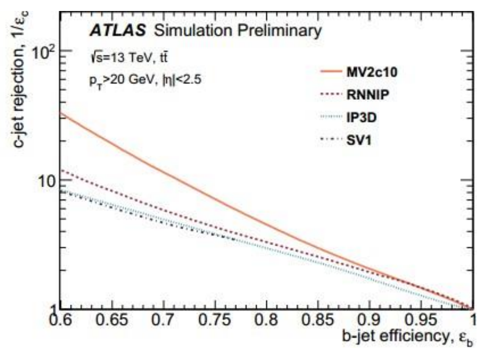
■ RNN discriminant

RNN tagger has a four-class output (p_b, p_c, p_τ, p_{light}) providing a flexible class of discriminants. But for better visualization, these outputs are combined into the following discriminant function:

$$D_{\mathrm{RNN}} = \ln \frac{p_b}{f_c p_c + f_\tau p_\tau + (1 - f_c - f_\tau) p_{\mathrm{light}}}$$ *$f_\tau = 0$, $f_c = 0.07$

*p_i is possibility being i-jet, f_i is the fraction of i-jet in the training samples





RNNIP algorithm shows a decent improvement wrt IP3D algorithm, even close to MV2 combined under high b-jet efficiency.

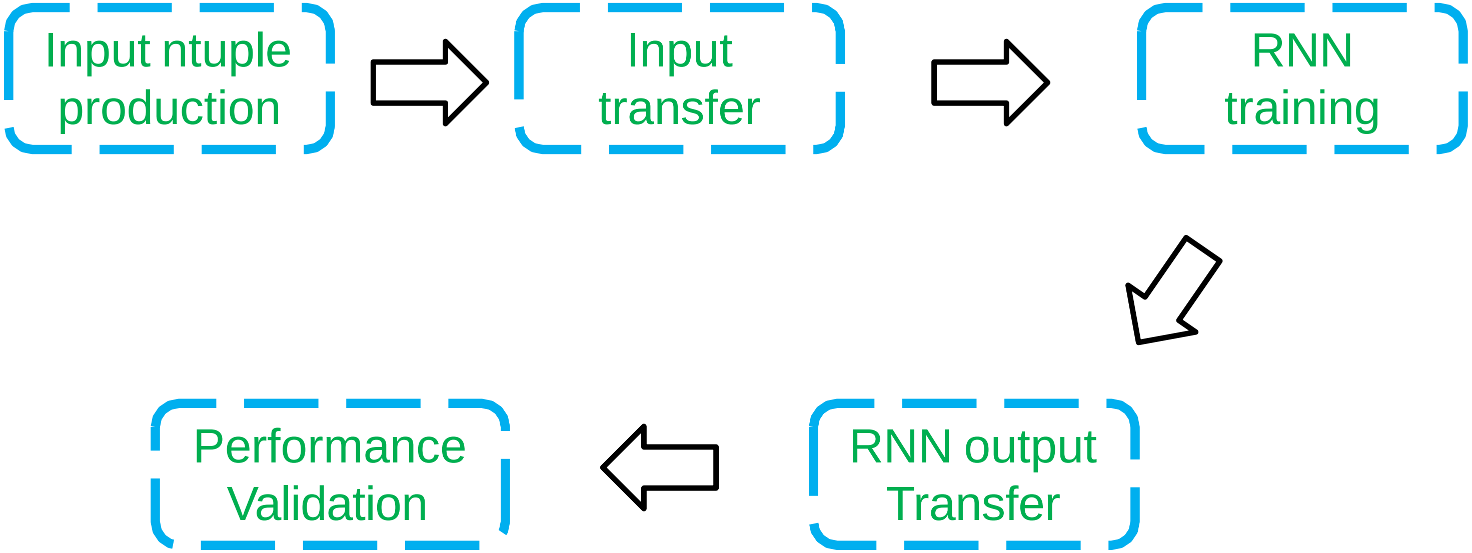
Rejection = 1 / efficiency

ATL-PHYS-PUB-2017-003

MV2c10 contains IP3D, SV1, JetFitter

Implement RNNIP on online jets

- Since RNNIP shows promising results for offline b-tagging performance, my job is to implement it at trigger-level.



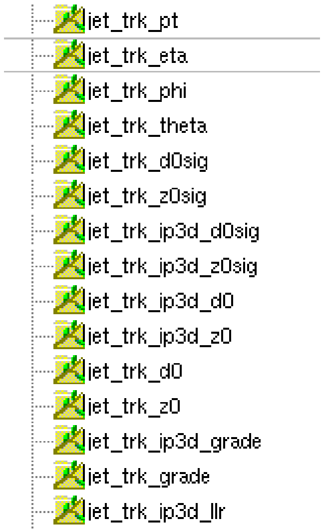
RNNIP training

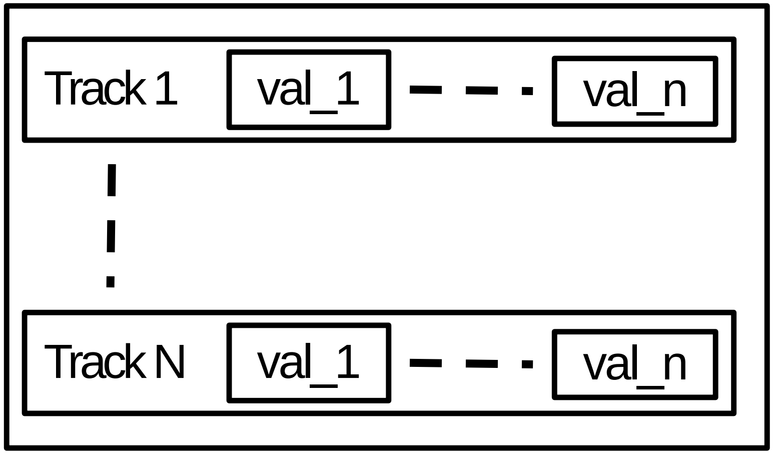
- **Input preparation:**

  Track selection: $p_T > 1$ GeV; $|d_0| < 1$ mm and $|z_0 \sin\theta| < 1.5$ mm;

- **Data structure:**

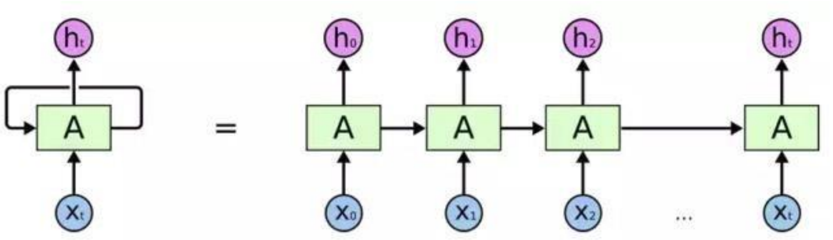
  Impact parameter info is transferred from tree to linear sequence.





RNNIP input

- Input:
  400k jets produced
- Training configuration:
  Model: LSTM, 50 nodes
  Half training, half validation.
  1.7 milion jet events

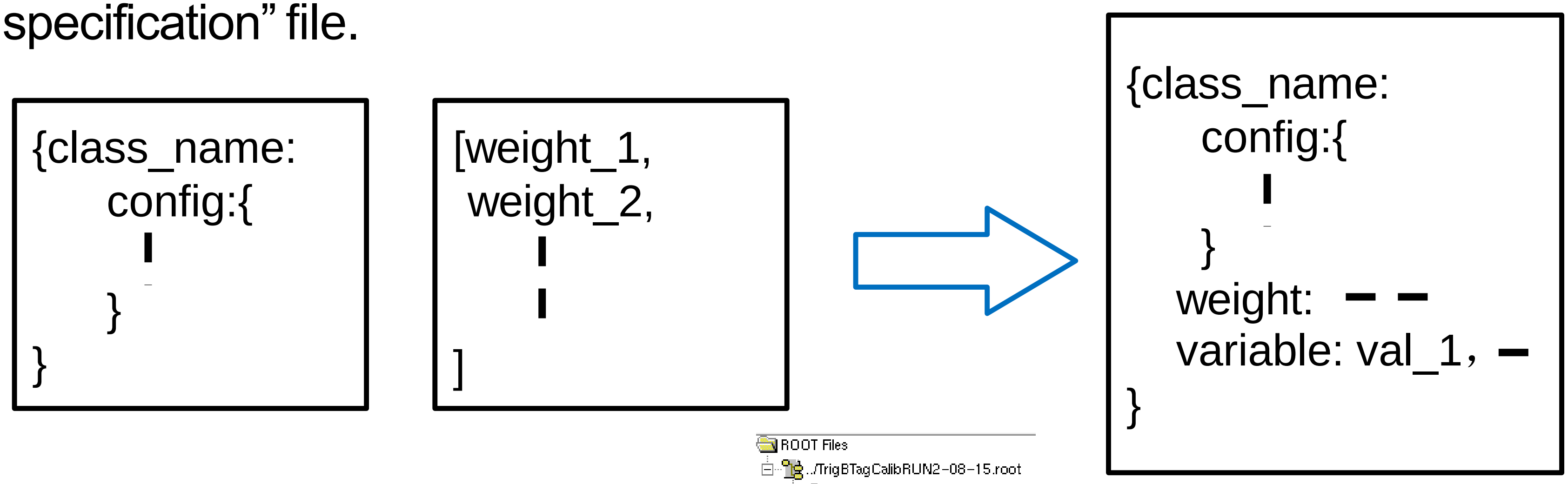


- Training output:
  RNN architecture: the architecture of the input data (arch.json)
  RNN variable weight: weight of variables or nodes (weight.h5)

RNNIP output Transfer

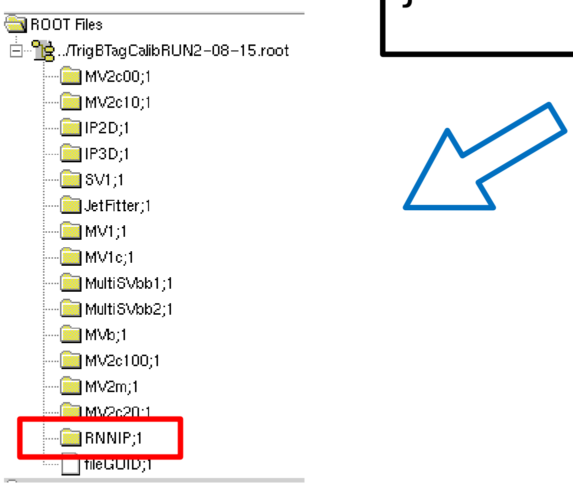
- Output combination

  The architecture json file and weight h5df file are combined to create a complete text output along with a manual "variable specification" file.



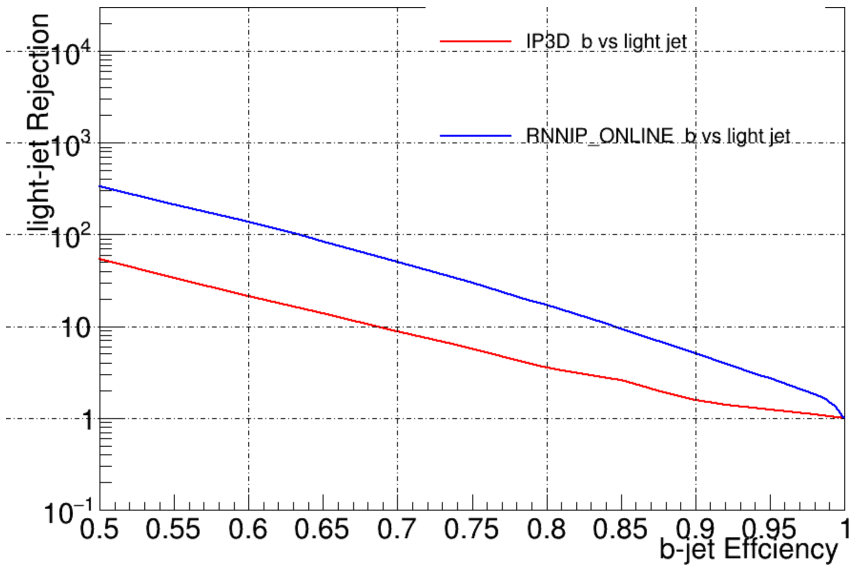
- TextToRoot

  The combined output is finally stored as a string in the root, so the RNNIP algorithm can use the DB file when it is called in the future.

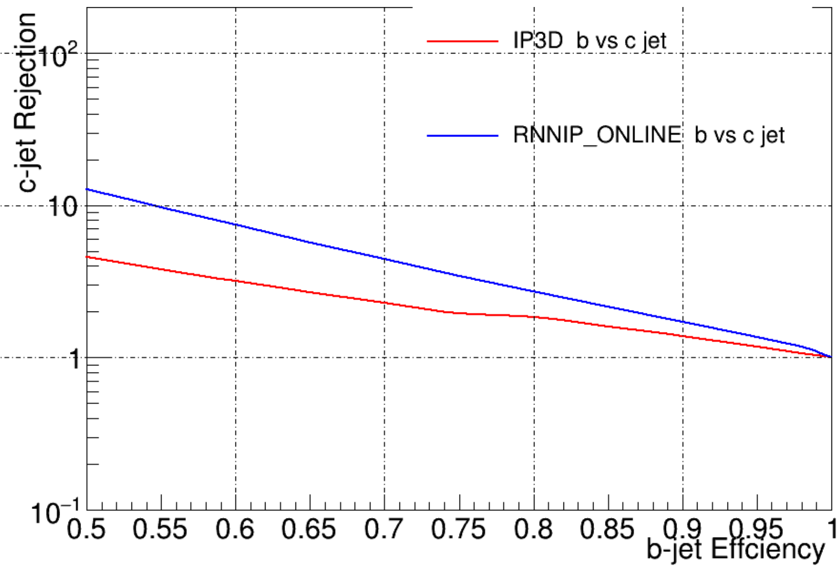


Performance validation

Light jet rejection



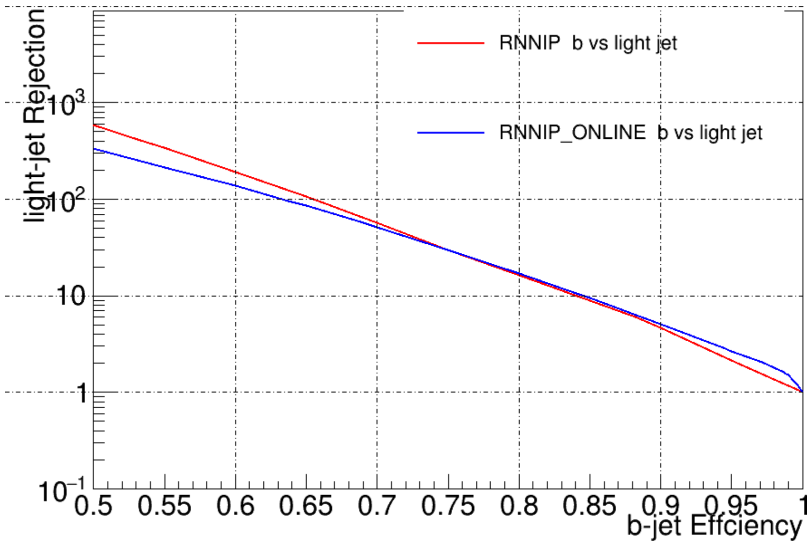
c-jet rejection



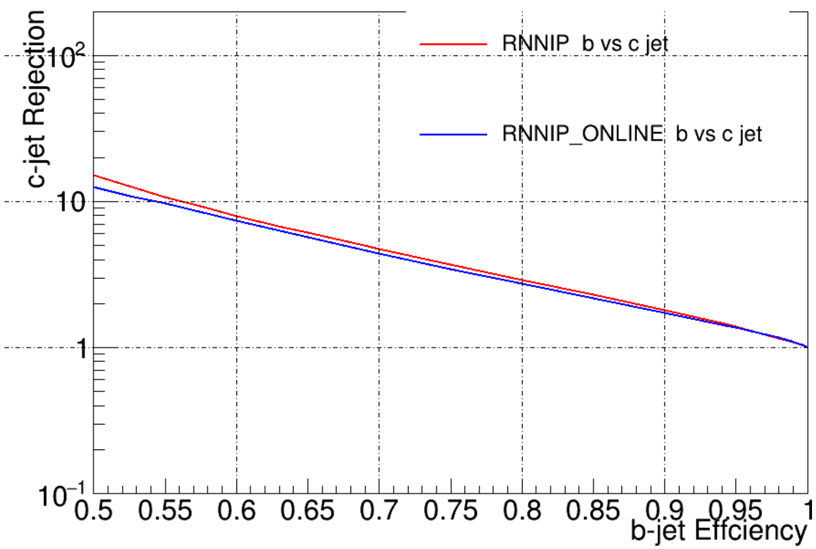
RNNIP provides a significant improvement on b-tagging performance than IP3D based on same Impact parameter information, which agrees with offline result.

Performance validation

Light jet rejection

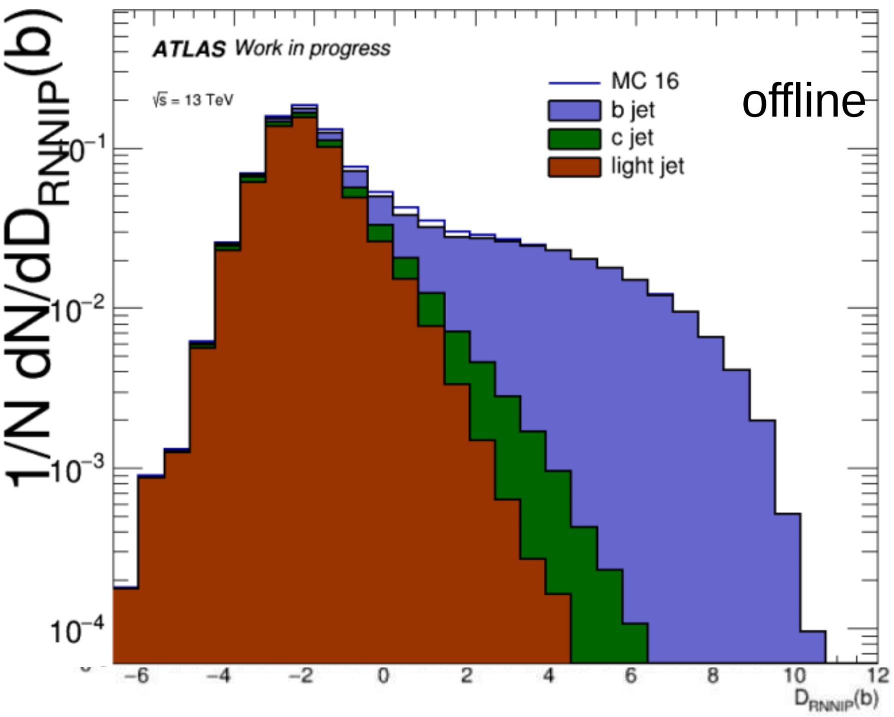


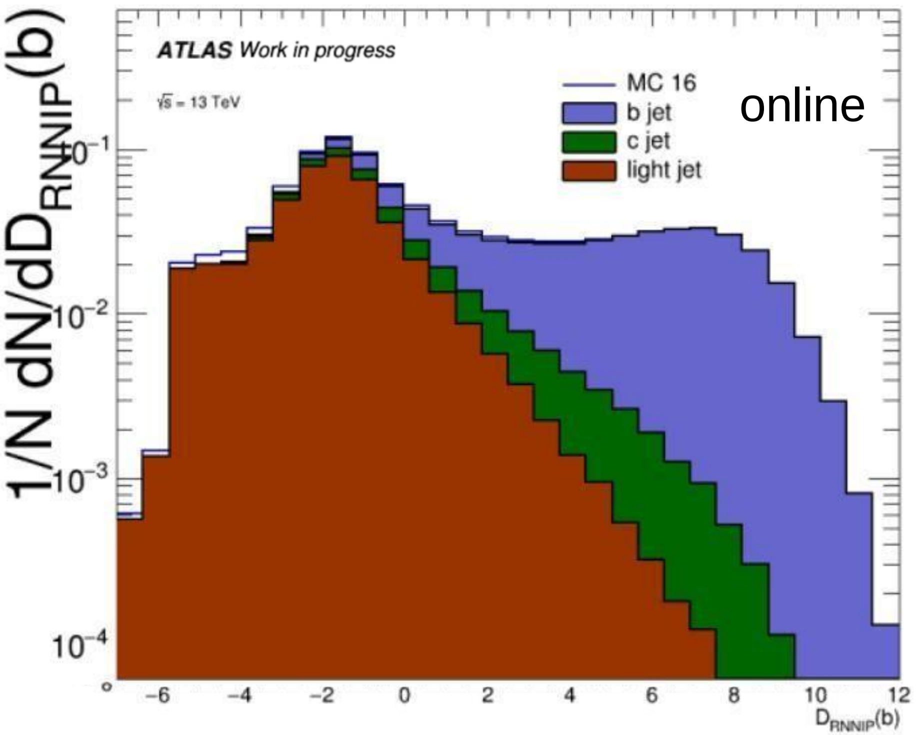
c-jet rejection



Comparing online result with offline, no significant degradation is discovered. Online RNNIP b-tagging performance seems promising.

Discriminant distribution





- Online distribution is similar with offline result on general distribution, but some differences are needed for further investigation.

MV2 with RNNIP

Try combining RNNIP with MV2 to MV2rnn

Problems come up with SMT (Soft Muon Tagger):

1. More than 95% events faill to find a muon on trigger level in tests, and makes it hard to do training with insufficient events.
2. MV2rnn is used by offline group. For consistency between online and offline, we are not able to remove SMT from MV2rnn.

MV2c10	MV2c10rnn
IP2D	IP2D
IP3D	IP3D
SV1	SV1
JetFitter	JetFitter
	SMT
	RNNIP

The implementation for MV2rnn is not practical for all reasons above.

MV2 with RNNIP

- New plan for MV2

MV2mu	MV2c10rnn
IP2D	IP2D
IP3D	IP3D
SV1	SV1
JetFitter	JetFitter
SMT	SMT
	RNNIP

→

MV2r	MV2rmu
IP2D	IP2D
IP3D	IP3D
SV1	SV1
JetFitter	JetFitter
RNNIP	SMT
	RNNIP

We will be able to implement RNNIP without the requirement of SMT.

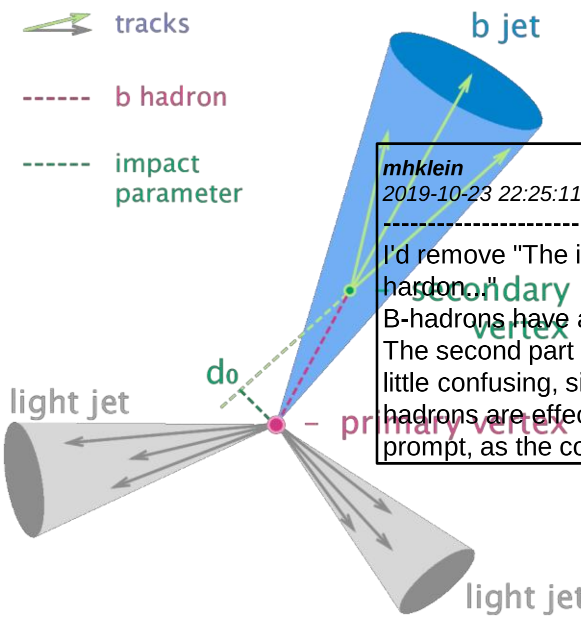
Summary

- A new IP-based algorithm RNNIP is implemented and tested on trigger level.
- The online RNNIP performance is closer to offline result, but some differences like input variables due to differences between trigger jets and offline jets.
- MV2rnn implementation is called off due to SMT, but we are planning on new algorithms like MV2r.

Backup

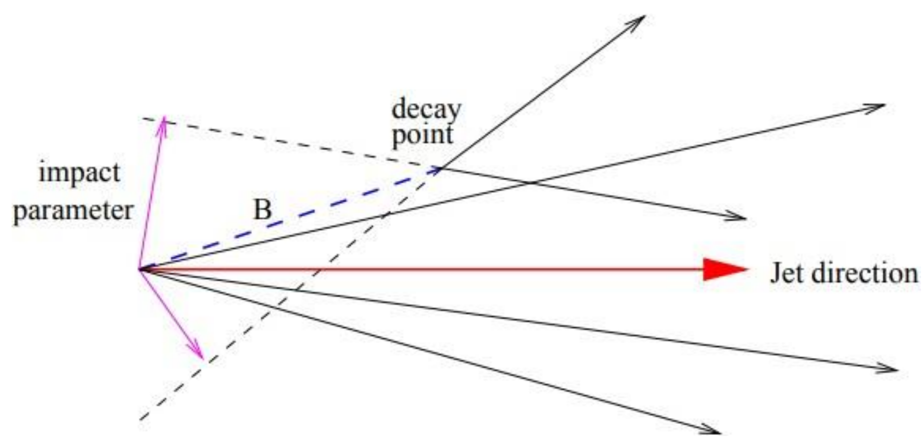
B-jet

The identification of jets containing B-hadron relies on the properties of b hadrons decays. B hadron has a lifetime around 1.6 ps, which corresponds to a 500μm distance which is higher than the other flavor.



mhklein
2019-10-23 22:25:11

I'd remove "The identification of jets c... b-hardon..."
B-hadrons have a lifetime of 1.6ps, or 5...
The second part of the sentence is a little confusing, since most other flavor hadrons are effectively immediate (or prompt, as the cool kids say).



Tracks originating from b decays have significant differences since they come from a displaced vertex, while the other tracks coming from the primary vertex are compatible with the tracking resolution.

mhklein
2019-10-23 22:25:39

I'd remove this sentence

Input preparation

Parameter implementation

B-jet Trigger Codes are updated to produce ntuples with needed parameters. https://gitlab.cern.ch/atlas-trigger/b-jet/TrigBtagAnalysis/tree/SG_mv2c10rnn

Several Parameter added:

- Hit information
- Track grade
- Track θ
- Track IP

```
+ std::vector<float> *v_jet_JVT;
+ std::vector<bool> *v_jet_aliveAfterOR;
  std::vector<int> *v_jet_truthflav;

  std::vector<float> *v_jet_dRminToB;
@@ -121,8 +126,36 @@ class BTriggerTuning : public :: AthHistogramAlgorithm {
  std::vector< std::vector <float> > *v_jet_trk_pt;
  std::vector< std::vector <float> > *v_jet_trk_eta;
  std::vector< std::vector <float> > *v_jet_trk_phi;
+ std::vector< std::vector <float> > *v_jet_trk_theta;
+ std::vector< std::vector <float> > *v_jet_trk_ip3d_d0sig;
+ std::vector< std::vector <float> > *v_jet_trk_ip3d_z0sig;
+ std::vector< std::vector <float> > *v_jet_trk_ip3d_d0;
+ std::vector< std::vector <float> > *v_jet_trk_ip3d_z0;
  std::vector< std::vector <float> > *v_jet_trk_d0sig;
  std::vector< std::vector <float> > *v_jet_trk_z0sig;
+ std::vector< std::vector <float> > *v_jet_trk_d0;
+ std::vector< std::vector <float> > *v_jet_trk_z0;
+ std::vector< std::vector <float> > *v_jet_trk_ip3d_llr;
+ std::vector< std::vector <int> > *v_jet_trk_ip3d_grade;
+ std::vector< std::vector <int> > *v_jet_trk_grade;
+
+ std::vector< std::vector <int> > *v_jet_trk_nInnHits;
+ std::vector< std::vector <int> > *v_jet_trk_nNextToInnHits;
+ std::vector< std::vector <int> > *v_jet_trk_nsplitBLHits;
+ std::vector< std::vector <int> > *v_jet_trk_nsplitPixHits;
+ std::vector< std::vector <int> > *v_jet_trk_nBLHits;
+ std::vector< std::vector <int> > *v_jet_trk_nPixHits;
+ std::vector< std::vector <int> > *v_jet_trk_nSCTHits;
+ std::vector< std::vector <int> > *v_jet_trk_nsharedBLHits;
+ std::vector< std::vector <int> > *v_jet_trk_nsharedPixHits;
+ std::vector< std::vector <int> > *v_jet_trk_nsharedSCTHits;
+ std::vector< std::vector <bool> > *v_jet_trk_expectBLayerHit;
+ std::vector< std::vector <bool> > *v_jet_trk_expectInnHit;
+ std::vector< std::vector <bool> > *v_jet_trk_expectNextToInnHit;
+
+ std::vector< std::vector <float> >*v_jet_sv1_vtx_x;
+ std::vector< std::vector <float> >*v_jet_sv1_vtx_y;
+ std::vector< std::vector <float> >*v_jet_sv1_vtx_z;
  // ---
```

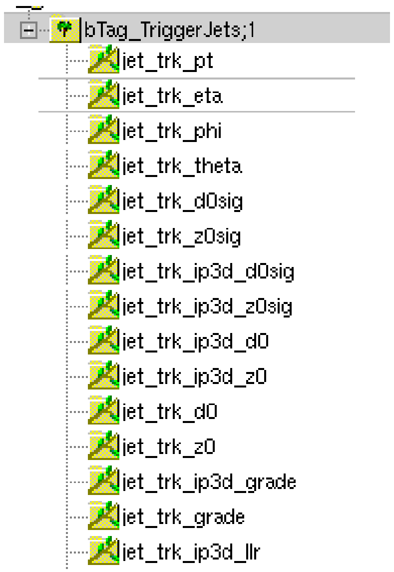
Input ntuples

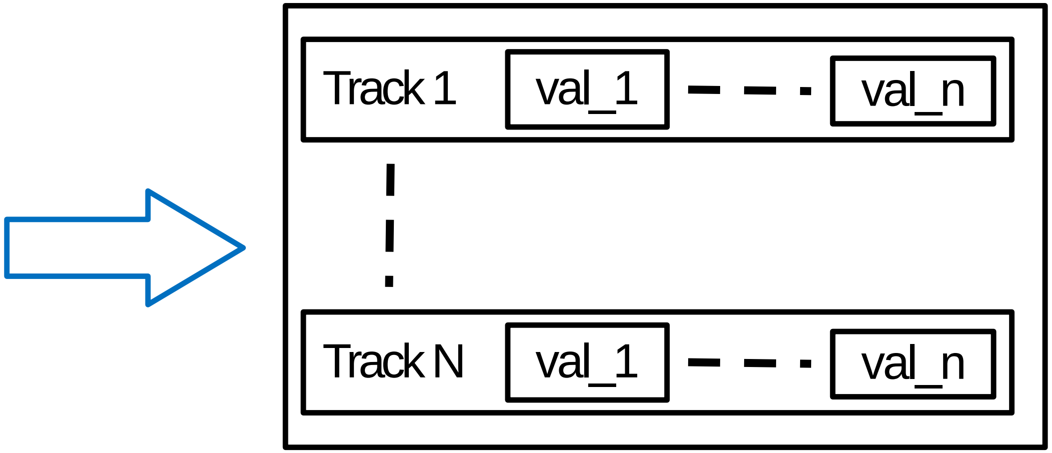
- Ntuple production：

  400k events ntuple from ttbar mcsamples*

- Ntuple transfer:

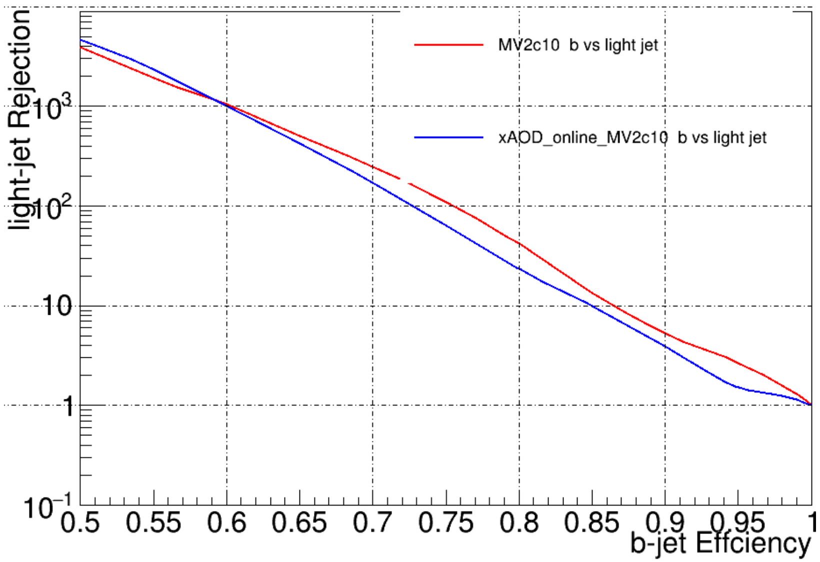
  RNN receives variable-length sequences as input, so Tree is converted to sequences data (*.pkl file).

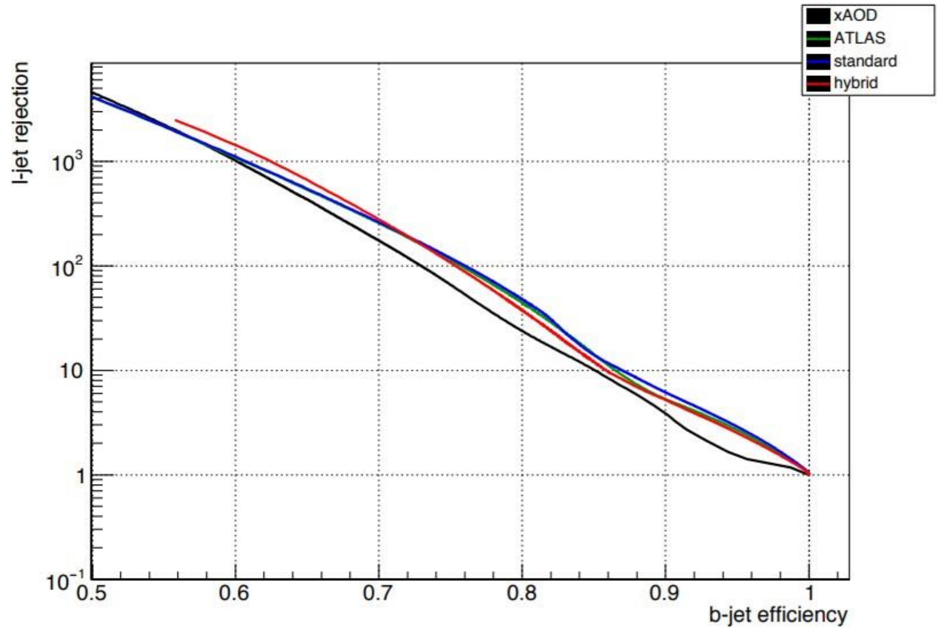




*mc16_13TeV.410000.PowhegPythiaEvtGen_P2012_ttbar_hdamp172p5_nonallhad.merge.AOD.e3698_s2997_r8903_r8906

Package check





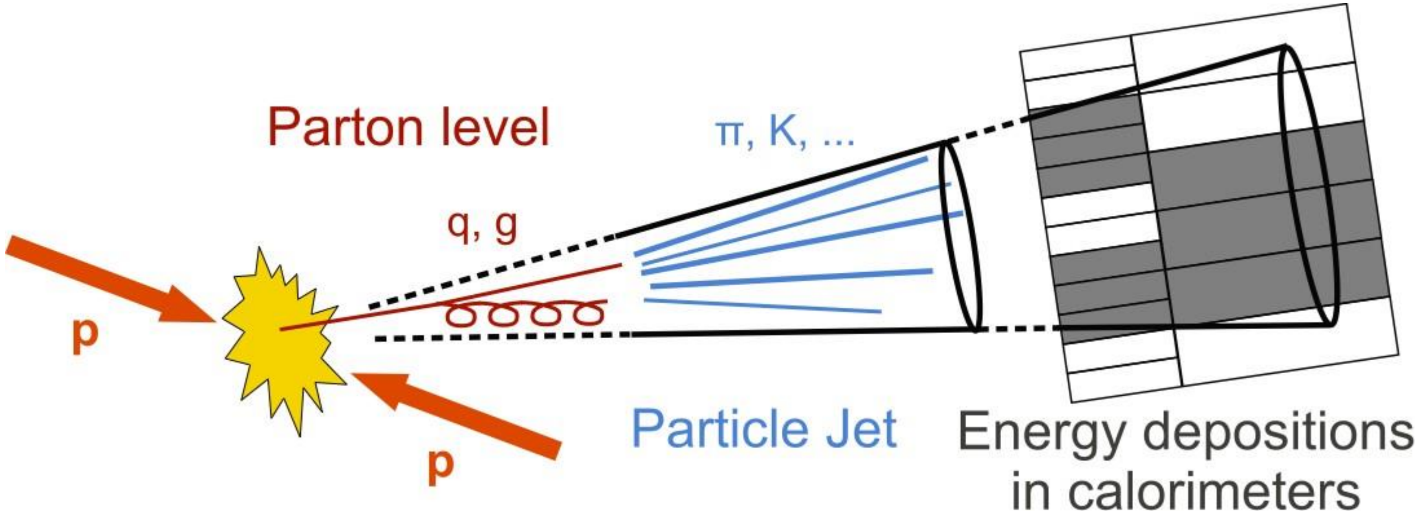
Before the implementation work, a consistency check was done between me and Francesco to make sure nothing wrong.

Blue(left) to Black(right)
Red(left) to blue (right)

Jet

What is a jet?

- A jet is a narrow cone of hadrons and other particles produced by the harmonization of a quark or gluon in a particle physics or heavy ion experiment.
- Jets are divided into different types according to the hadron type they contain. (b-jet, c-jet, tau-jet, light-jet)



Input preparation

- **Track selection:**

  RNNIP uses the same track cut with IP3D

  - p_T >1 GeV;
  - $|d_0|$ <1 mm and $|z_0 \sin\theta|$ <1.5 mm;
  - seven or more silicon hits with at most two silicon holes, at most one of which is in the pixel detector, where a hole is defined as a hit expected to be associated with the track but not present [10].

First, we manually applied the cut in the online codes, but a mismatch comes up between IP3D container and track container.

Later, we found that IP3D IP information is slightly different from the track container one. So track container objects are required to be matched with IP3D tracks to make sure they are using the same cut.

Performance validation

- The ROC curves cross over between online and offline result, which did not show off on other algorithms before. So further check on input variables are done:
- Input track variables:

IP3D significance(sd0,sz0)

pTFrac($\mathrm{pT}_{track}/\mathrm{pT}_{jet}$)

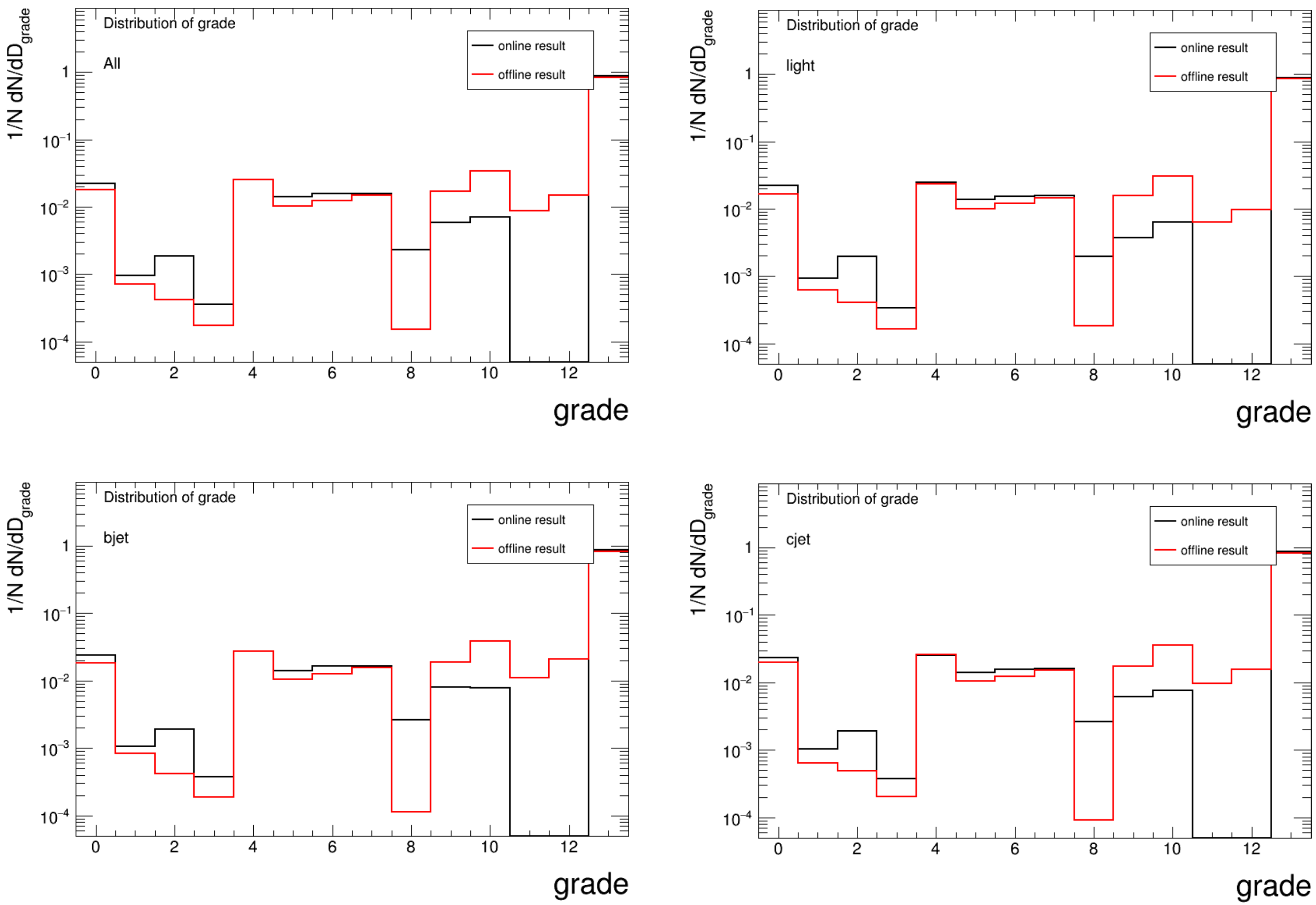
dR (between track and jet)

grade: track category defined by hits

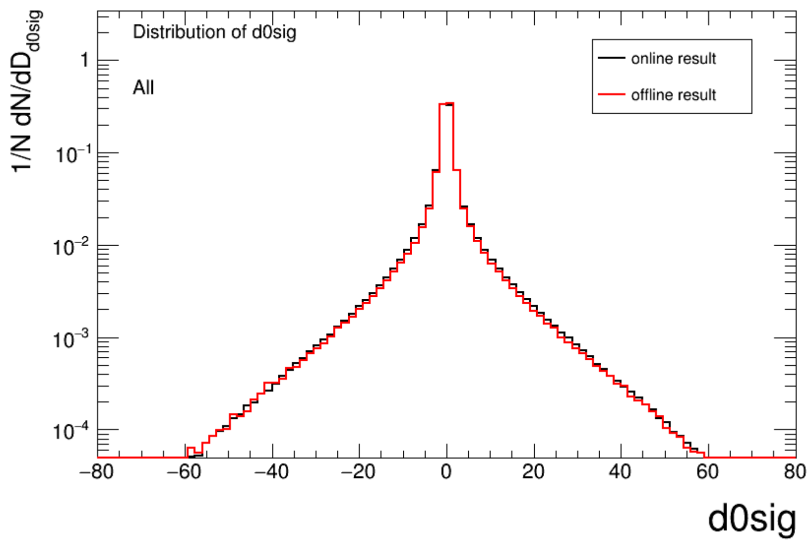
- Output variables:

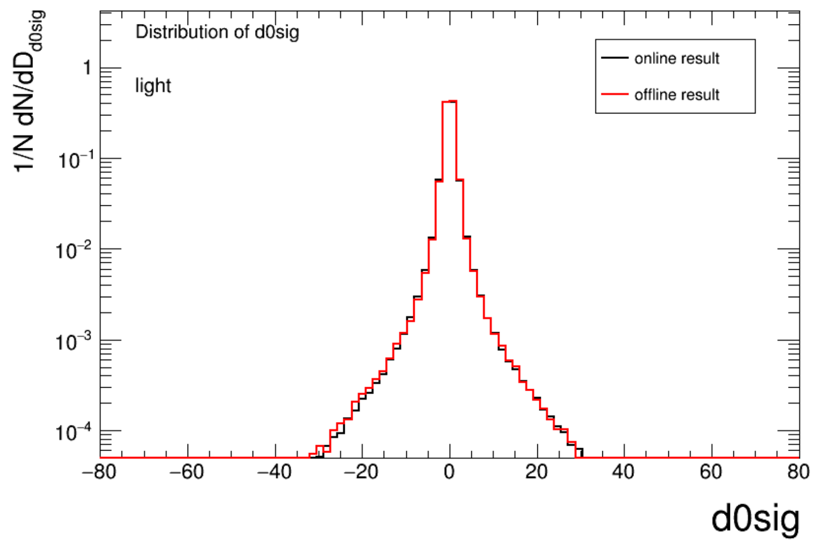
rnnip,pb,pc,pu (the possibility of a jet being b-jets, c-jets, light-jets), D_rnn

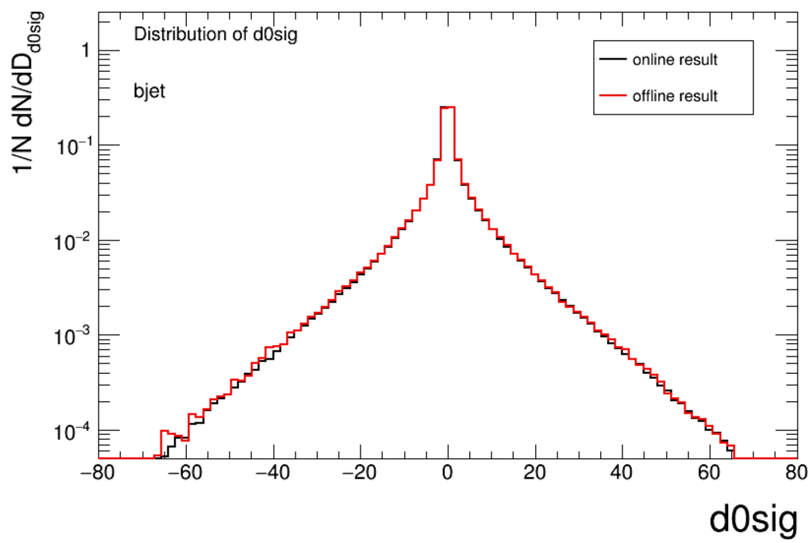
Grade(Track Category)

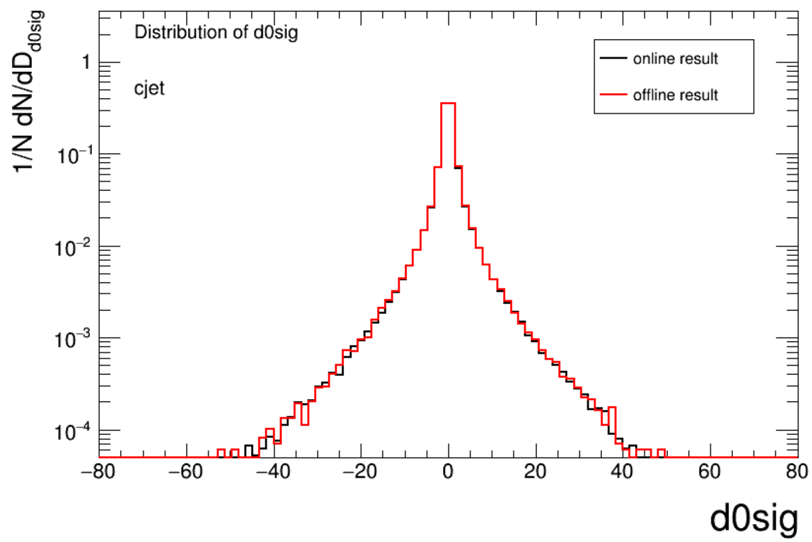


d0 significance

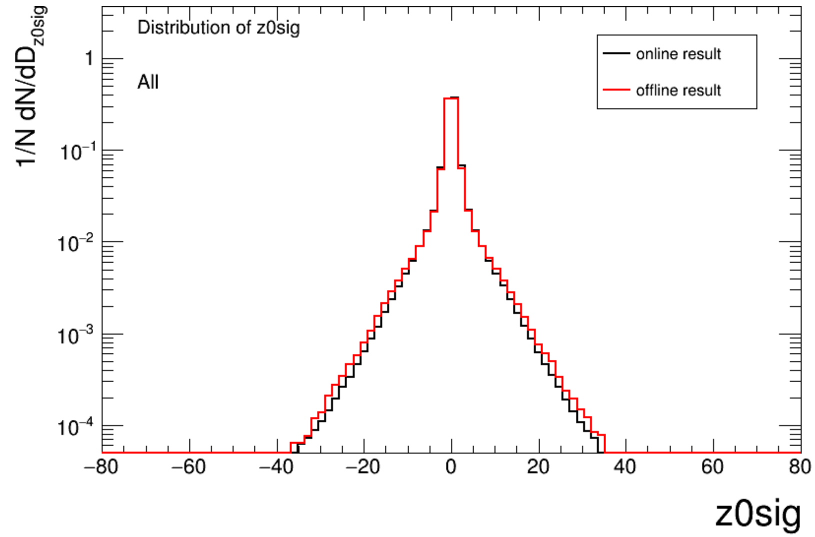


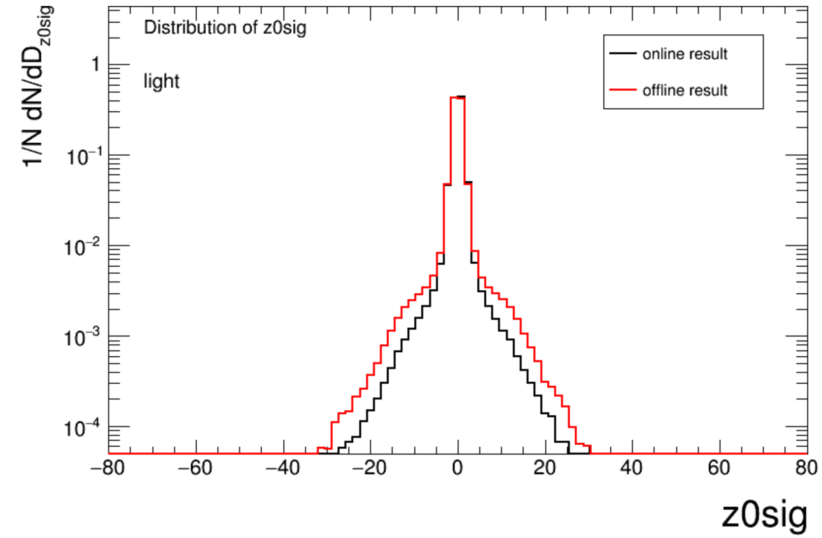


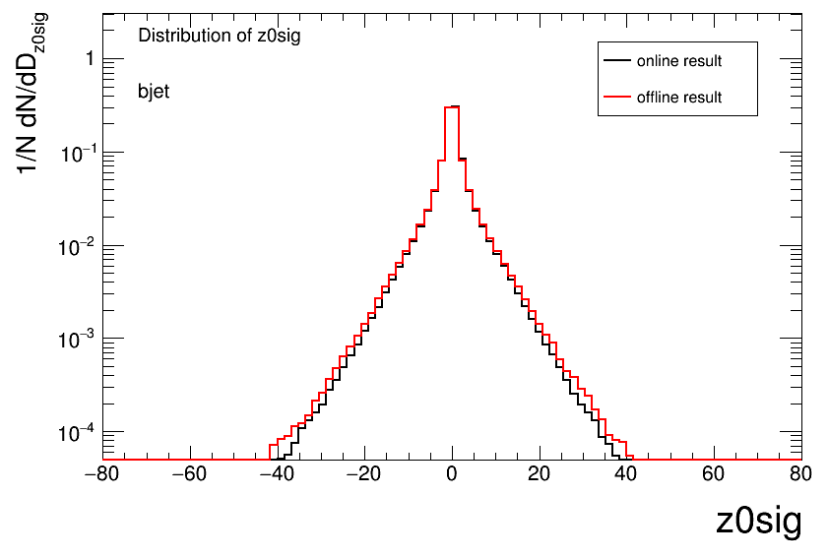


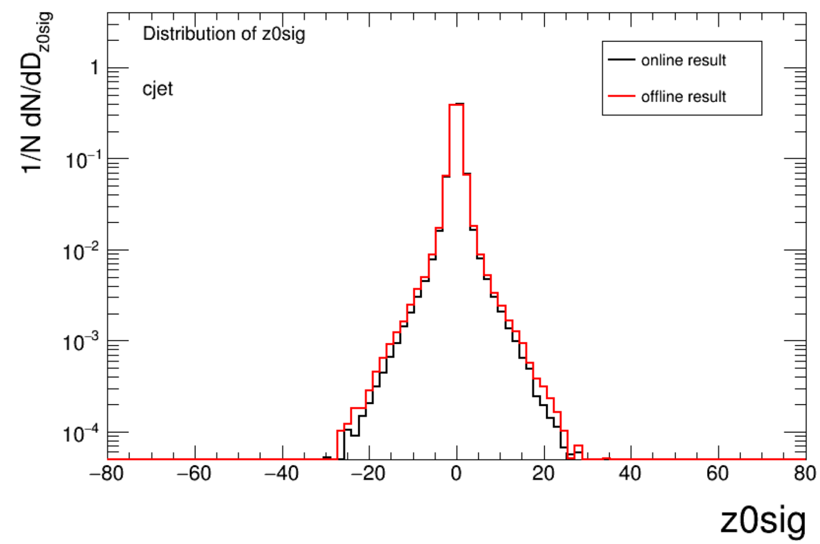


z0 significance



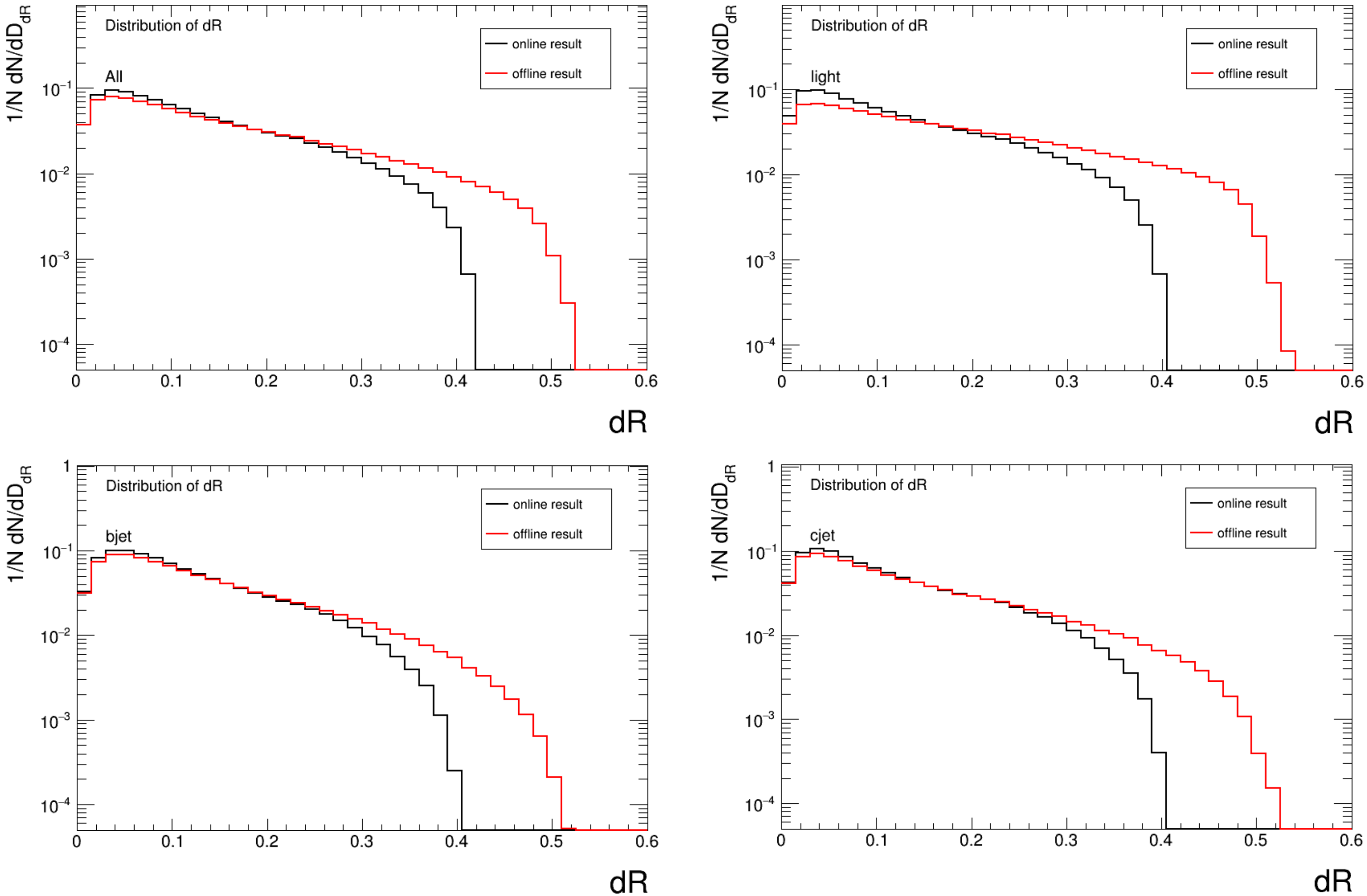






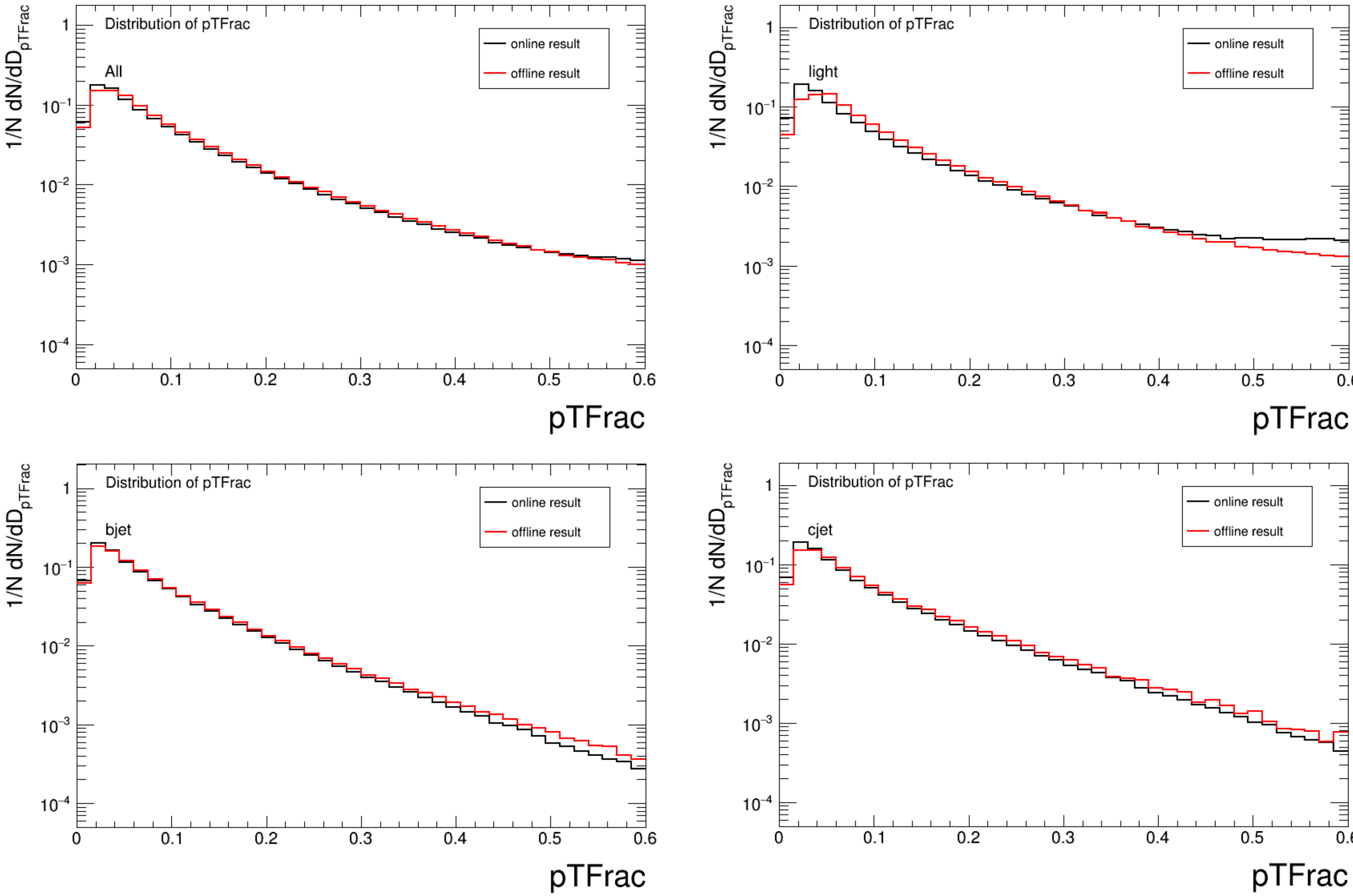
dR (between track and jet)

$$\sqrt{(\phi_{\text{track}} - \phi_{\text{jet}})^2 + (\eta_{\text{track}} - \eta_{\text{jet}})^2}$$

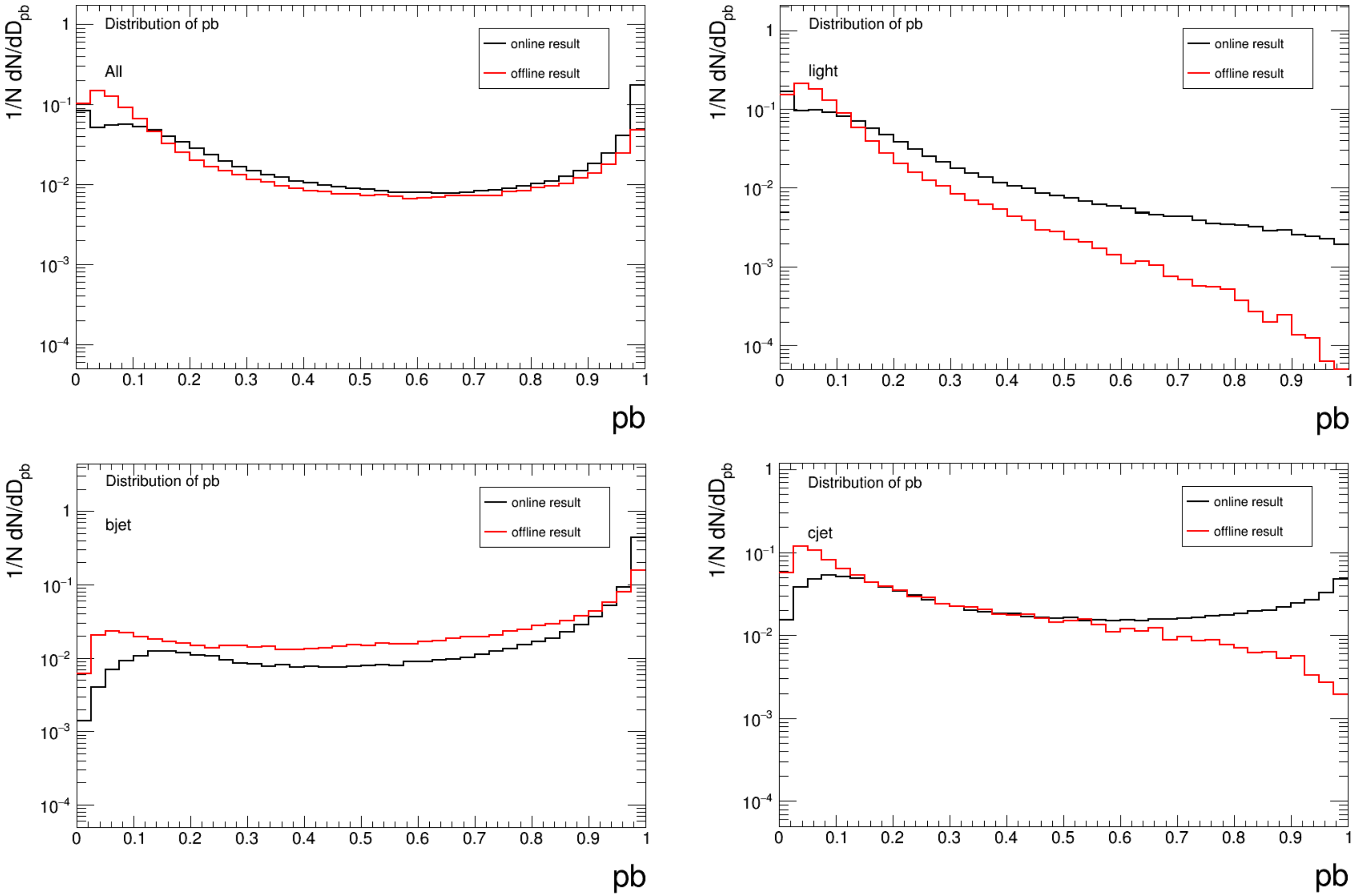


pTFrac

p_T^{track} / p_T^{jet}

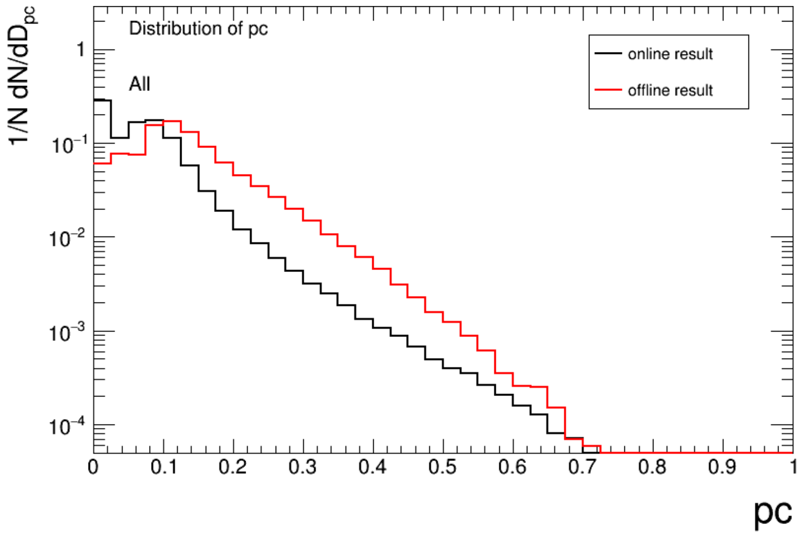


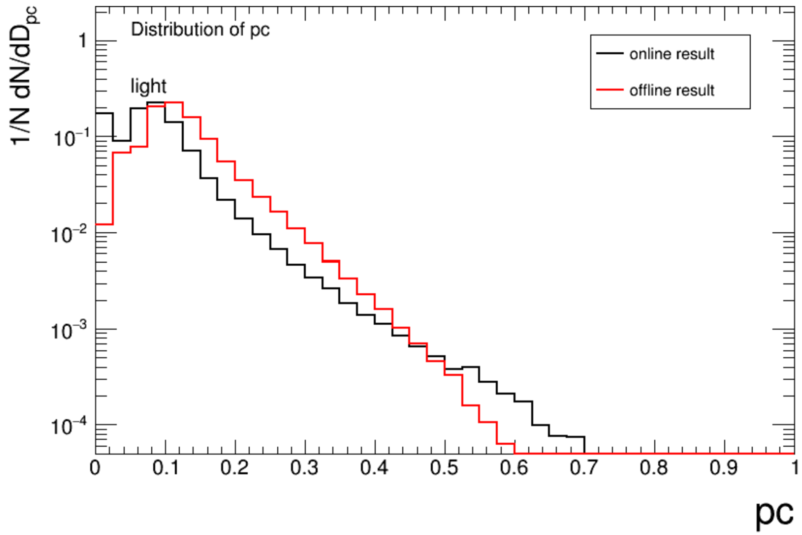
rnnip_pb

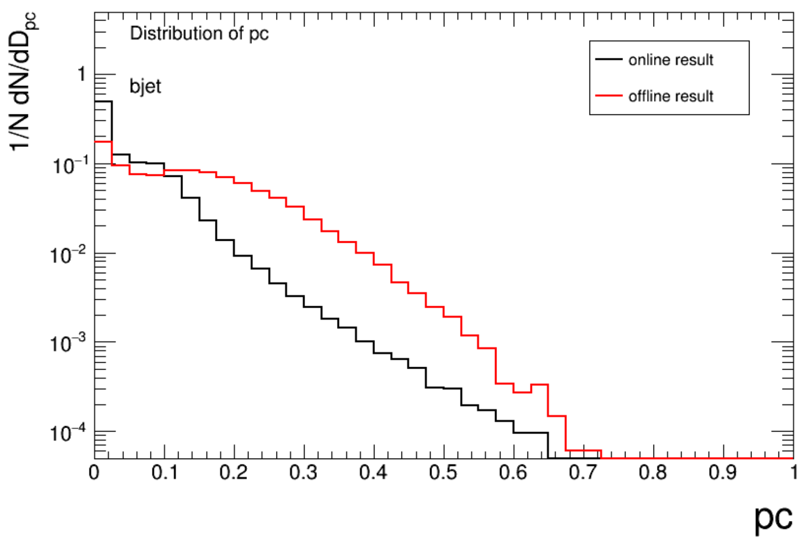


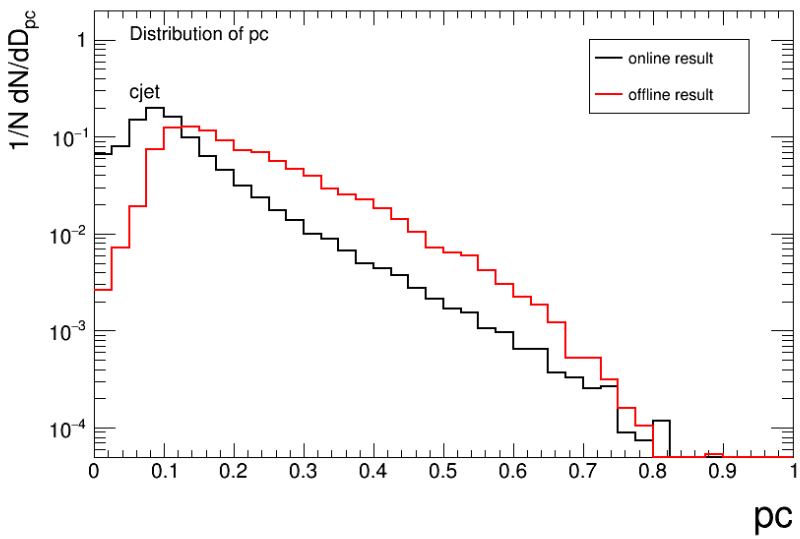
Distribution of pb
All
bjet
light
cjet
online result
offline result
1/N dN/dD$_{pb}$
pb

rnnip_pc

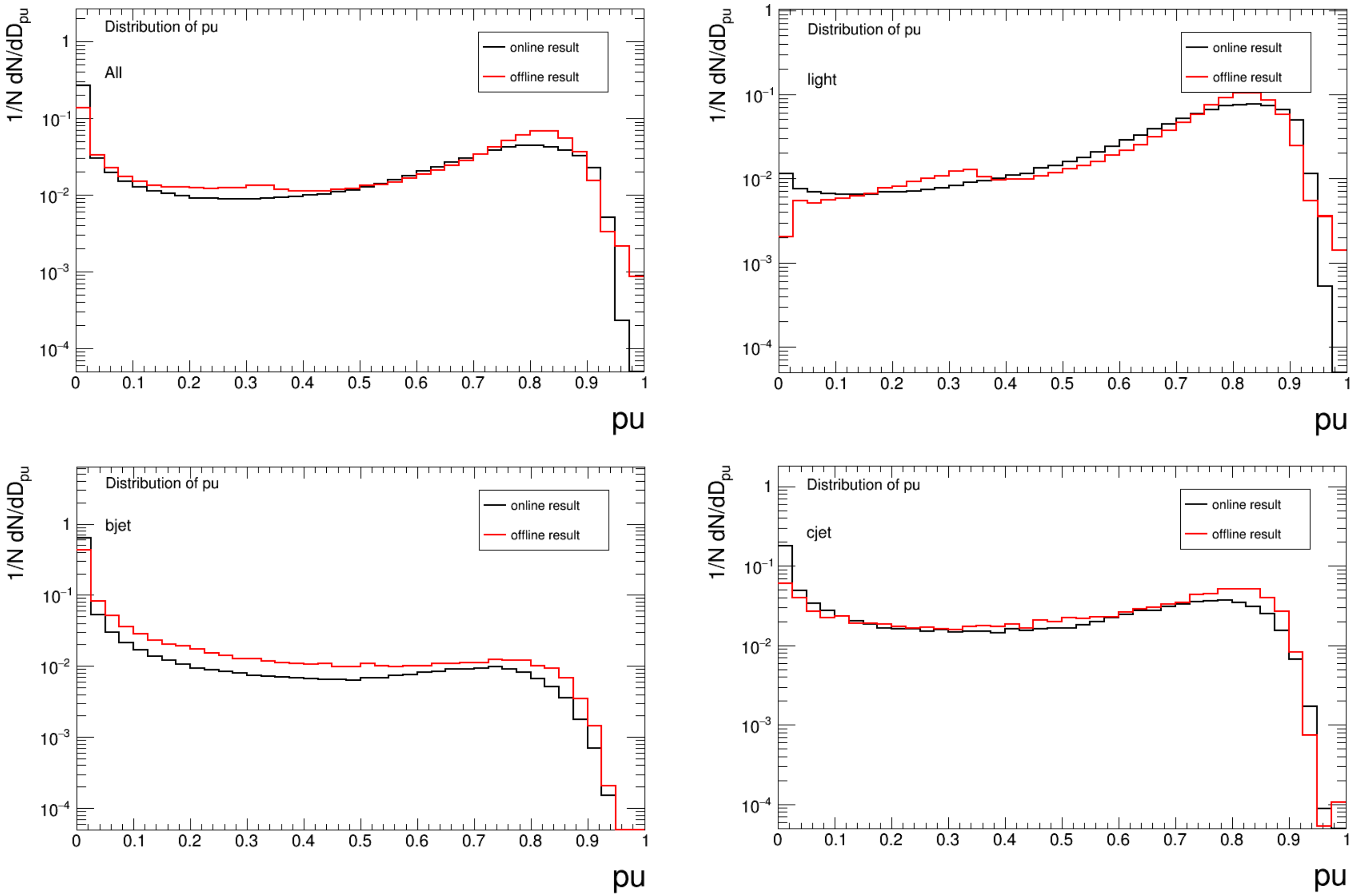

Distribution of pc
All
online result
offline result
1/N dN/dD_pc
pc


Distribution of pc
light
online result
offline result
1/N dN/dD_pc
pc


Distribution of pc
bjet
online result
offline result
1/N dN/dD_pc
pc


Distribution of pc
cjet
online result
offline result
1/N dN/dD_pc
pc

rnnip_pu



Distribution of pu
All
online result
offline result
1/N dN/dD_pu
pu
Distribution of pu
light
online result
offline result
1/N dN/dD_pu
pu
Distribution of pu
bjet
online result
offline result
1/N dN/dD_pu
pu
Distribution of pu
cjet
online result
offline result
1/N dN/dD_pu
pu