Lund plane based boosted top/boson taggers

COMHEP, December 5th 2024

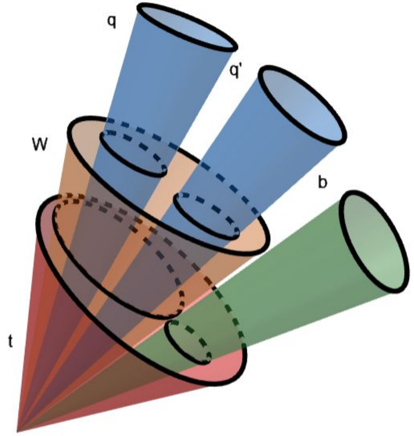
Jean Pierre Cifuentes[1], Rafael Vinasco[1], Reina Camacho[2], Jad M. Sardain[3], Carlos Sandoval[1], Mario Campanelli.[4]

1. Universidad Nacional de Colombia
2. LPNHE
3. The University of Arizona
4. UCL

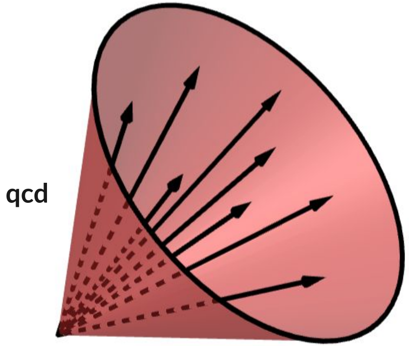
What is the goal?

Use **Lund plane variables** as input for machine learning methods to develop a **boosted particles** (With a very high Pt) tagger for hadronically decaying **top/W/H**:

Signal (Top):



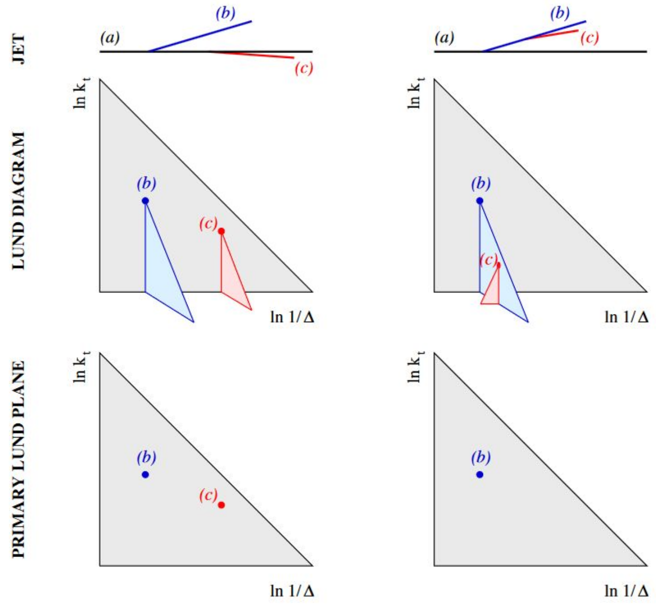
Background (qcd):



The Lund Jet Plane

- Each emission represented by a point in the **kT-emission angle** plane (log scale)

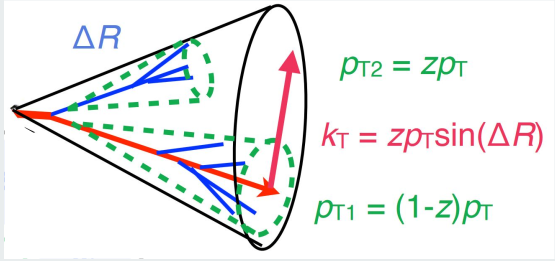
z Momentum fraction of the branching,
k_t Transverse momentum,
Δ Emission angle,



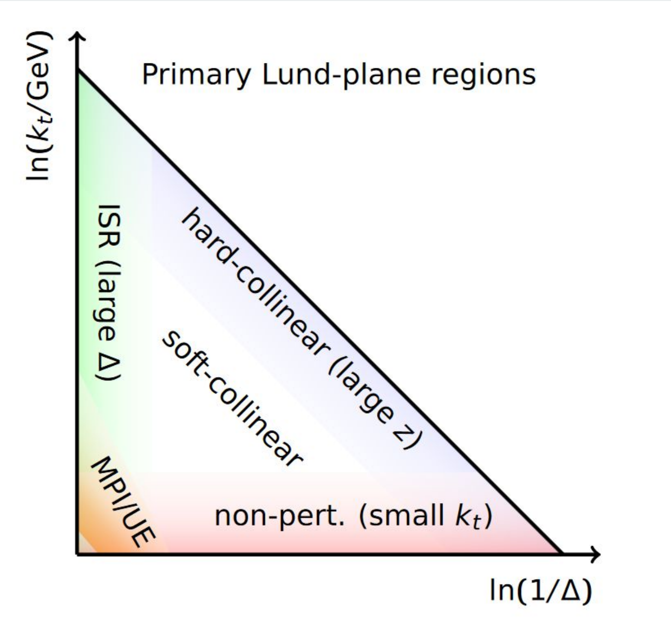
arXiv:1807.04758v2 [hep-ph]

The Lund Jet Plane

- Each emission represented by a point in the kT-emission angle plane (log scale)
- Hard scattering, collinear and large-angle emissions populate different regions of the plane.



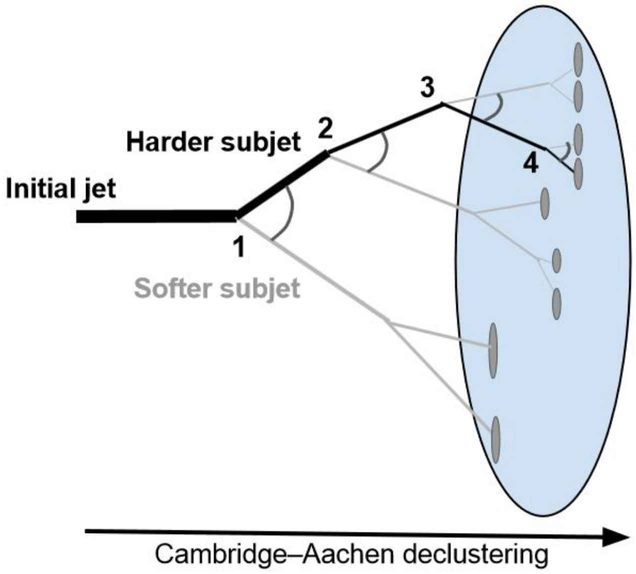
arXiv:1808.03689



arXiv:1807.04758v2 [hep-ph]

The Lund Jet Plane

- Each emission represented by a point in the kT-emission angle plane (log scale)
- Hard scattering, collinear and large-angle emissions populate different regions of the plane.
- Lund Plane is built by running back the **Cambridge-Aachen** jet clustering algorithm.

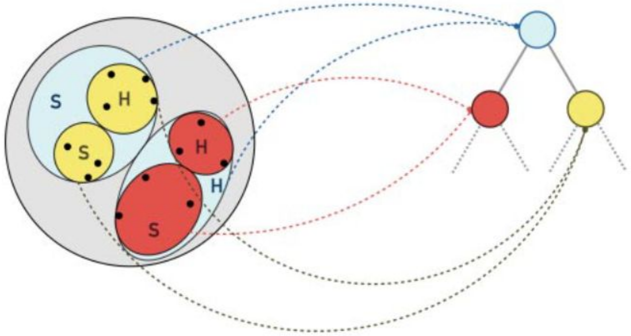


[arXiv:2312.16343](https://arxiv.org/abs/2312.16343)

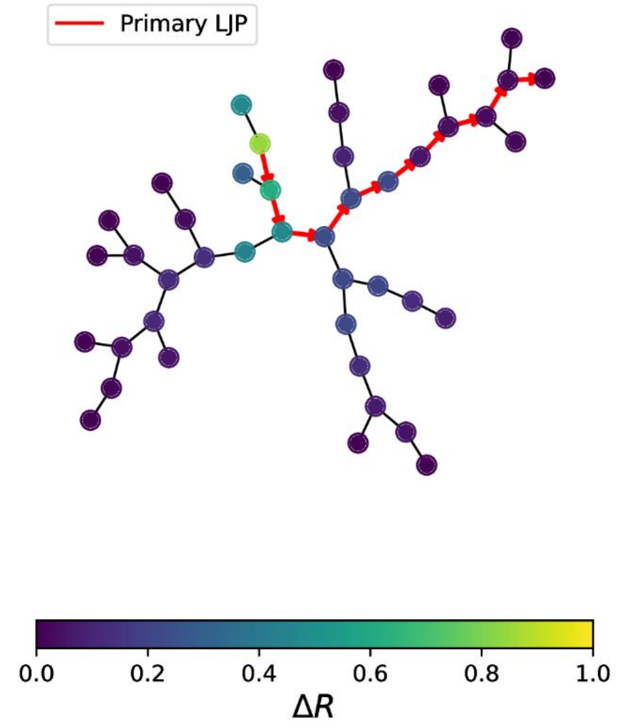
Jet reconstruction algorithm, that clusters first the large angle contributions

The Lund Jet Plane as a graph!

Lund planes is made up as a set of vertices and their connection, so this is an ideal input for Graph Neural Networks!



10.1393/ncc/i2024-24112-2



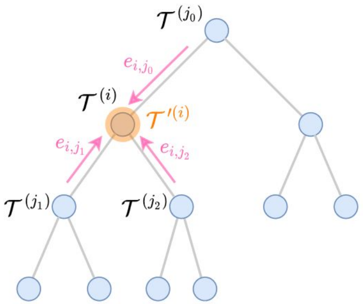
ATL-PHYS-PUB-2023-017

Jet tagging using Lund plane

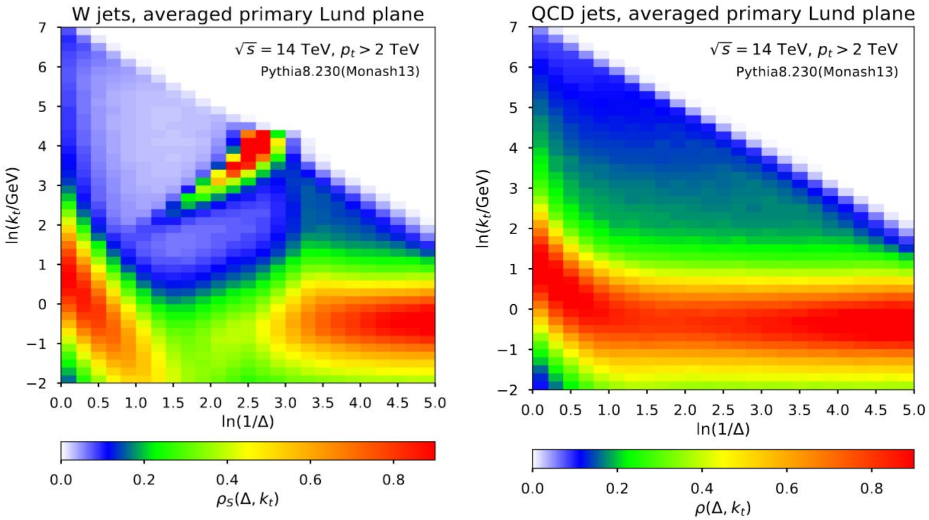
The Lund planes for the **top** (or **W** or **H**) and **QCD** jets already look quite different

-Use GNN where each node has 3 variables:

$$\mathcal{T}^{(i)} = \{k_t, \Delta, z\}$$

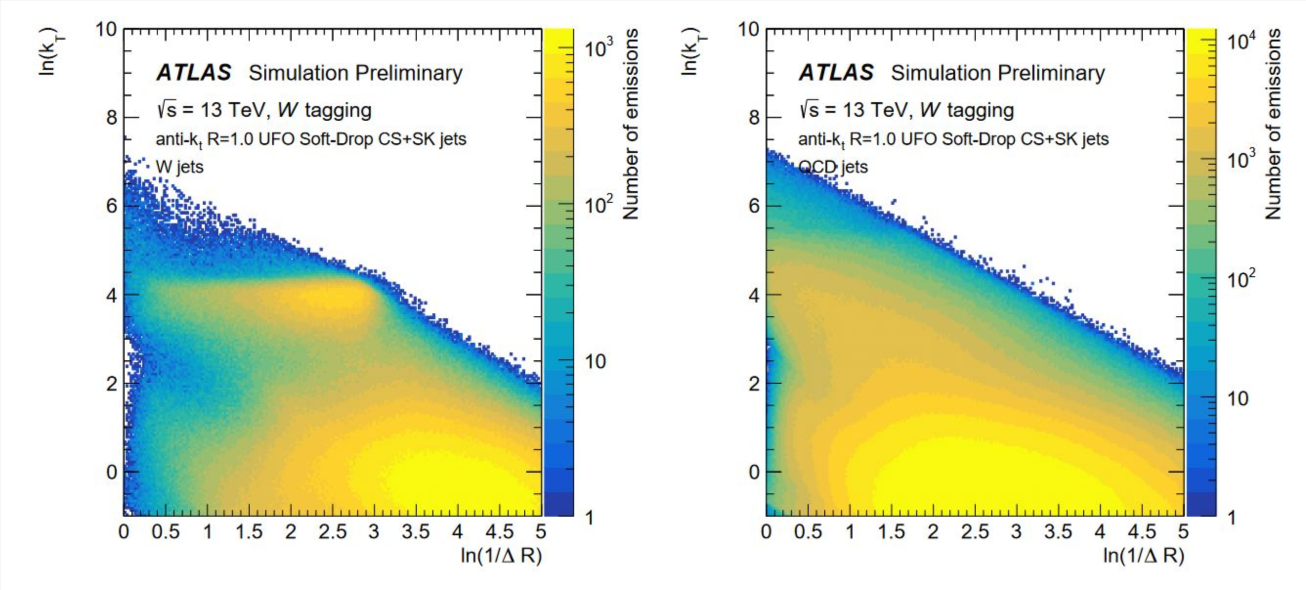


arXiv:2012.08526v2



arXiv:1807.04758v2 [hep-ph]

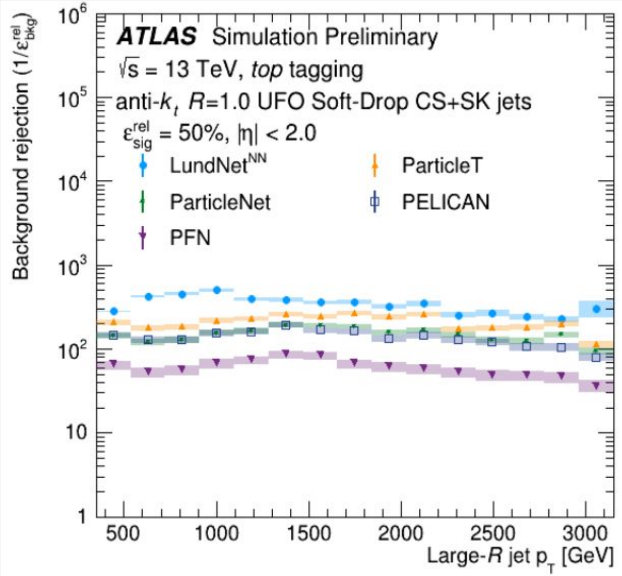
Implementation: LJP

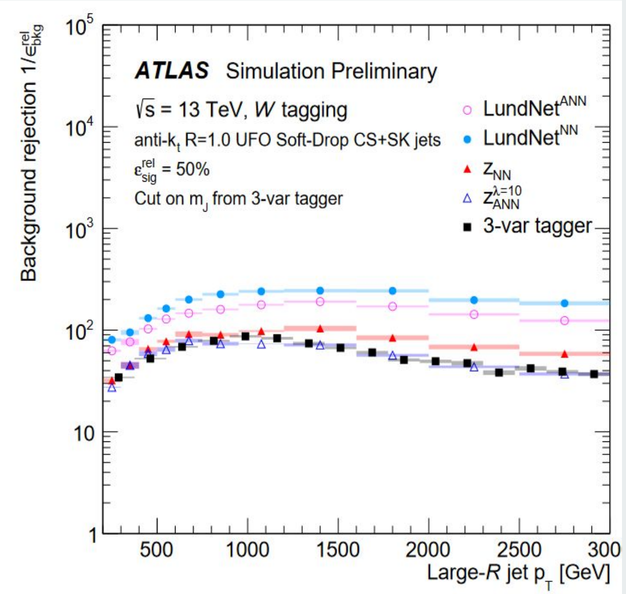


ATL-PHYS-PUB-2023-017

The planes obtained for both signal and background describe the expected behavior in the hard/soft collinear region

W/Top tagging training:





The important (for us) is the Blue line LundNetNN

ATL-PHYS-PUB-2023-017

The Background rejection give us information about the performance of the tagger

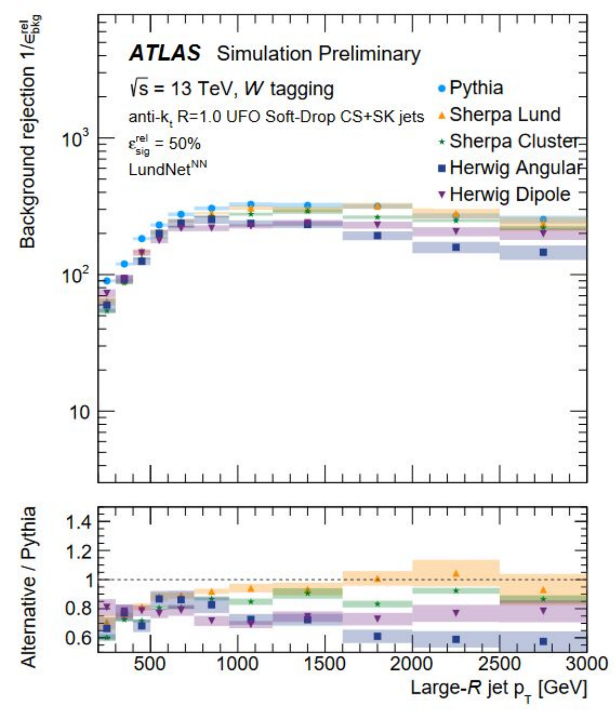
Different MC generators

But the tagger has a downside:
MC modeling dependence

It is important to study these differences in performance when changing the MC generators

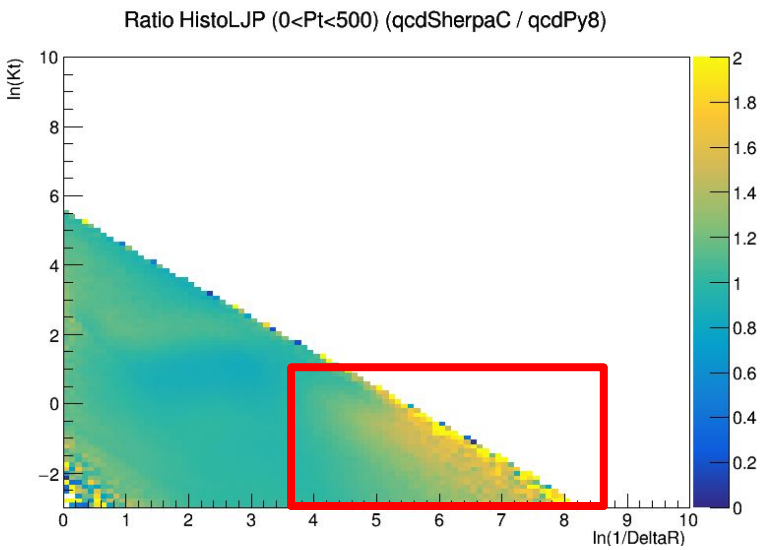
Jet collection is used.

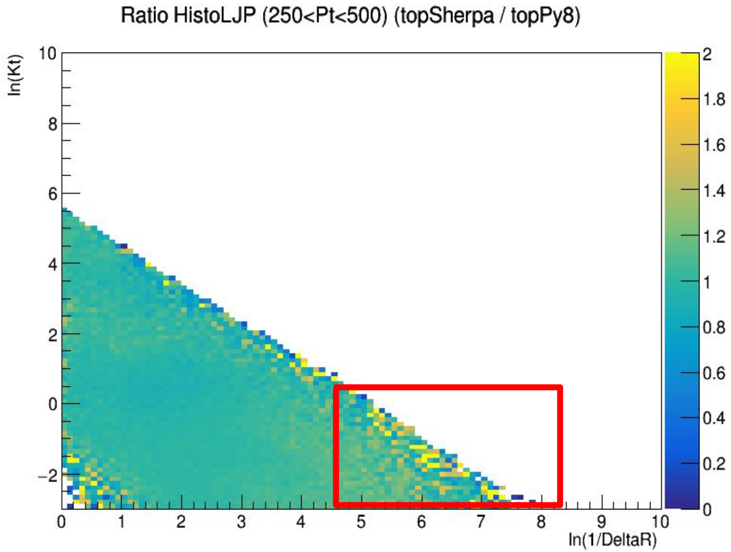
- Sherpa Lund
- Sherpa Cluster
- Herwig Dipole
- Herwig Angular



ATL-PHYS-PUB-2023-017

Ratio:



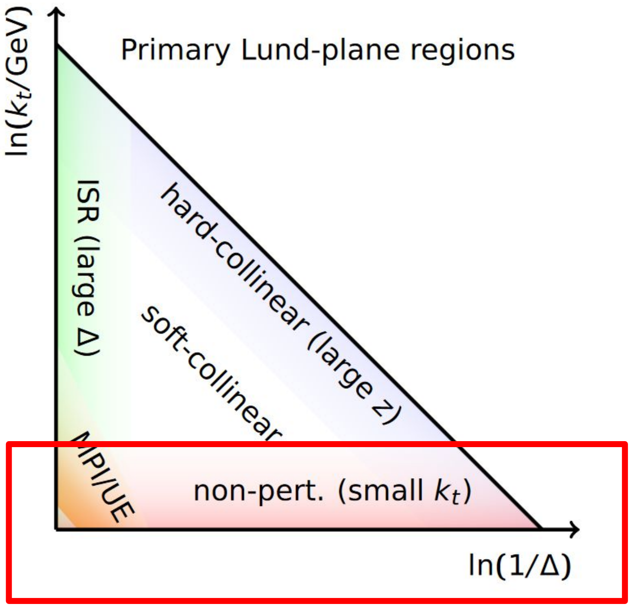


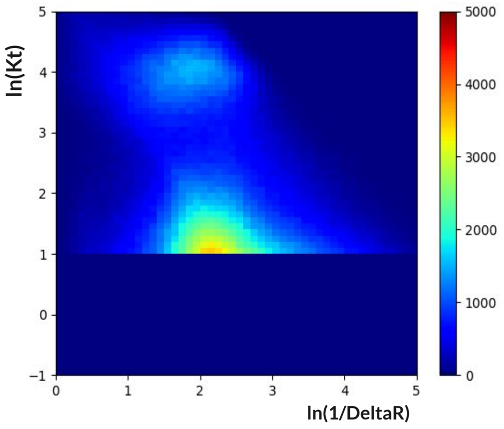
We see how the ratios between the Lund planes are quite **close to 1** in both cases, but slight differences are noted in the **low kt** region.

Lund Plane trimming using kt

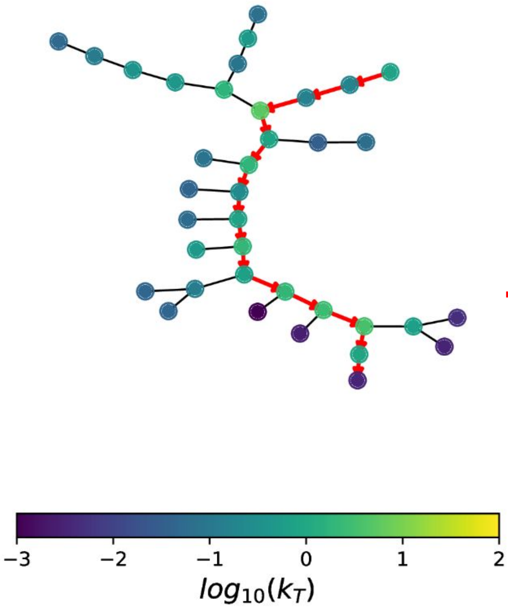
Lund Plane trimming using kt

In order to study and reduce the modeling dependence, it was proposed to make cuts in the Lund plane, more specifically in the low Kt region.



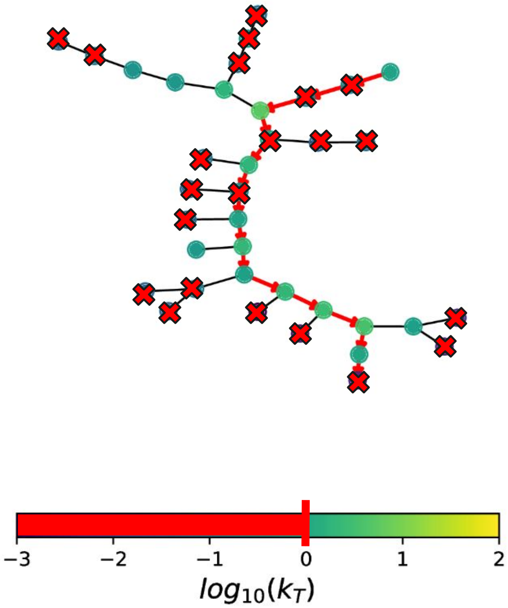


Lund Plane trimming using kt

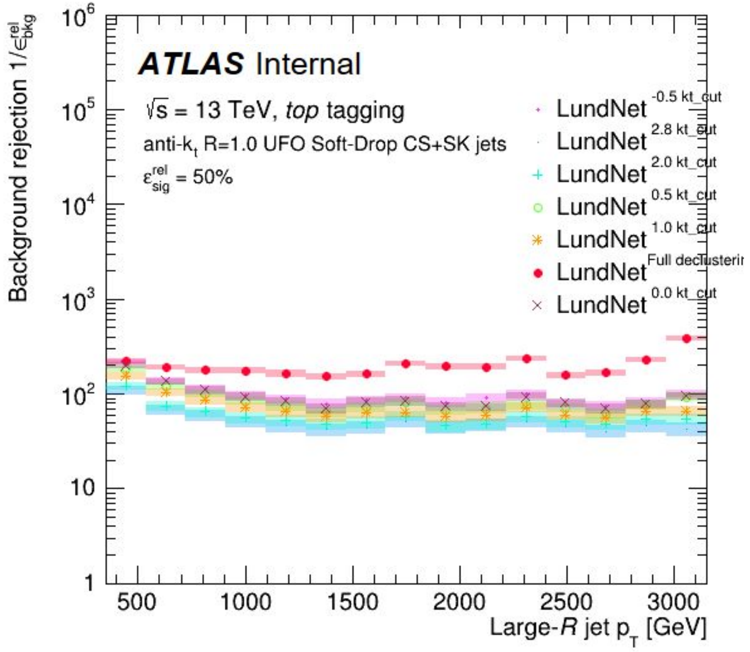


By making Kt cuts We are eliminating nodes from the graph

Then We must reconnect them



Performance comparison



Many information lost with the cut

It is necessary to investigate if making these kt cuts improves the MC modeling dependence.

Thanks!

And Backup...

Different MC generators:

Other samples with other mc generators

Samples (QCD) JETM2:

Sherpa-Lund:
mc20 13TeV - Sherpa Lund - dijet (JZ[2:9])

Sherpa-Cluster:
mc20 13TeV - Sherpa Cluster - dijet (JZ[2:9])

Herwig Dipole:
mc20 13TeV - Herwig Dipole- dijet (JZ[2:8])

[Documentation]

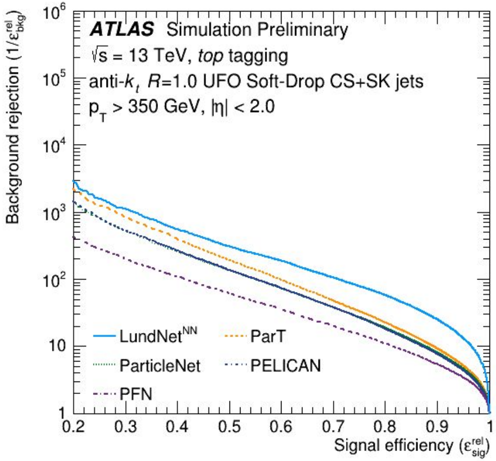
Main Differences:

➔ **Hadronization Model**:
  - **Pythia8** and **Sherpa Lund** use the **Lund model** for hadronization.
  - **Herwig** and **Sherpa Cluster** uses the **cluster model** for hadronization.

➔ **Pythia**, **Sherpa** and **Herwig** have a different **hard scattering modeling**.

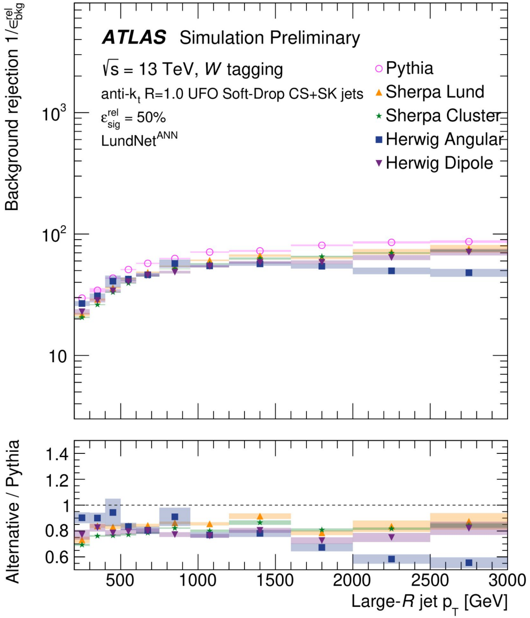
Previous Results (R21)

Obtained by a group from: LPNHE/UNAL/AZ/UCL



$$\epsilon_{\text{signal}} = \frac{N^{\text{identified}}_{\text{signal}}}{N^{\text{total}}_{\text{signal}}} \qquad \frac{1}{\epsilon_{\text{background}}} = \frac{N^{\text{identified}}_{\text{background}}}{N^{\text{total}}_{\text{background}}}$$

The Lund plane tagger performs better than other tagger



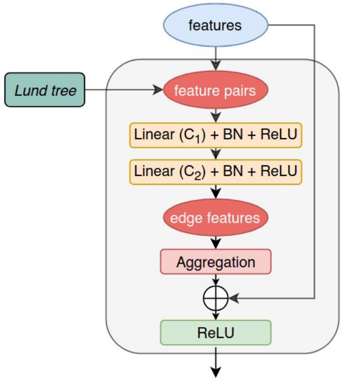
[ATL-PHYS-PUB-2023-017]

It has a downside: **MC modeling dependence**

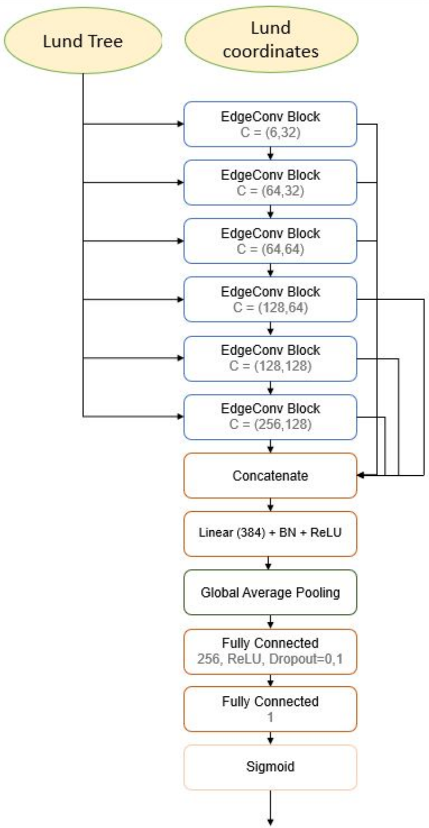
It is important to study these differences in performance when changing the MC generators

Classifier architecture

We use two layers for this shared MLP, each consisting of a linear layer followed by a batch normalization (BN) and a ReLU activation



LundNet: arXiv:2012.08526v2



- Six EdgeConv blocks
- Another MLP with 384 channels
- Read out information from all nodes in the graph
- Fully connected layer with 256 units and a dropout layer with a drop probability of 0.1

ParticleNet: [arXiv:1902.08570]

Data sets sizes

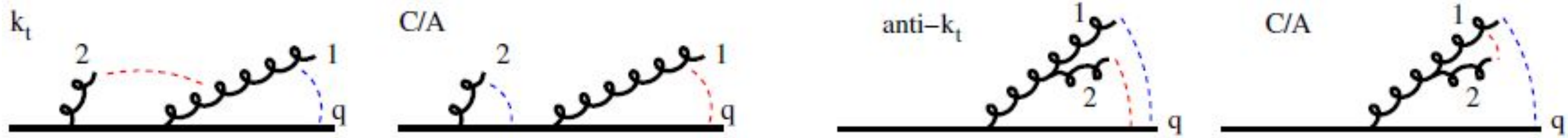
For some jets no node pass kt selection, so number of jets is reduced

ln(kt) cut = 2.8 ~~ 1.46 millions	1.49 nodes per graph	QCD=47.02%	Top=99.54%
ln(kt) cut = 2.0 ~~ 1.96 millions	1.94 nodes per graph	QCD=65.05%	Top=99.92%
ln(kt) cut = 1.0 ~~ 2.60 millions	3.60 nodes per graph	QCD=88.03%	Top=99.98%
ln(kt) cut = 0.5 ~~ 2.79 millions	5.24 nodes per graph	QCD=94.95%	Top=99.99%
ln(kt) cut = 0 ~~ 2.88 millions	7.44 nodes per graph	QCD=98.11%	Top=99.99%
ln(kt) cut = -0.5 ~~ 2.91 millions	10.14 nodes per graph	QCD=99.92%	Top=99.99%
ln(kt) NO cut ~~ 2.93 millions	38.36 nodes per graph		

Variables associated with the declustering

$$y = \ln \frac{E+p_z}{E-p_z} \qquad \Delta_{ij}^2 = (y_i - y_j)^2 + (\phi_i - \phi_j)^2$$

$$\boxed{\begin{array}{l} \Delta \equiv \Delta_{ab}, \quad k_t \equiv p_{tb}\Delta_{ab}, \quad m^2 \equiv (p_a + p_b)^2, \\ z \equiv \dfrac{p_{tb}}{p_{ta}+p_{tb}}, \quad \kappa \equiv z\Delta\,, \quad \psi \equiv \tan^{-1}\dfrac{y_b - y_a}{\phi_b - \phi_a}, \end{array}}$$



- The Declustering algorithms tries to go inside the hadronization history in order to determine where each emission is coming from.

Contributions at NLO:

$$\bar{\rho}_2^{(k_t)}(\Delta,\kappa) \simeq -4C_F^2 \ln^2\frac{\Delta}{\kappa} + \mathcal{O}(L) \,.$$

Kt algorithm

$$\bar{\rho}_2^{(\text{anti-}k_t)}(\Delta,\kappa) \simeq +8C_F C_A \ln^2\frac{\Delta}{\kappa} + \mathcal{O}(L) \,.$$

Anti-Kt algorithm

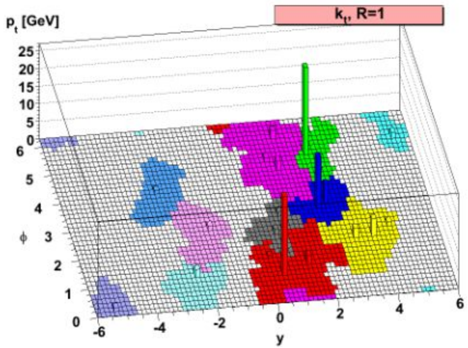
$$\bar{\rho}_{2,\text{rc}}^{(\text{C/A})}(\Delta,\kappa) = \bar{\rho}_1(\Delta,\kappa)\, 4\pi b_0 \ln\frac{1}{\kappa} + \mathcal{O}(1) \,.$$

C/A algorithm

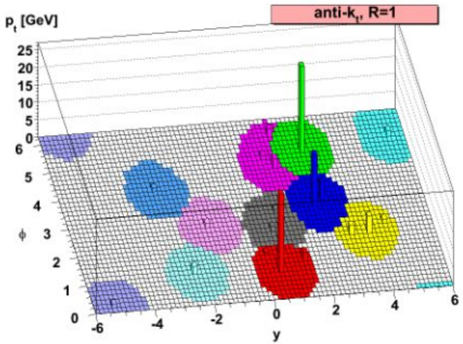
Jet-algorithm sequences

$$d_{ij} = \min\left(k_{ti}^{2p}, k_{tj}^{2p}\right)\frac{\Delta_{ij}^2}{R^2}$$

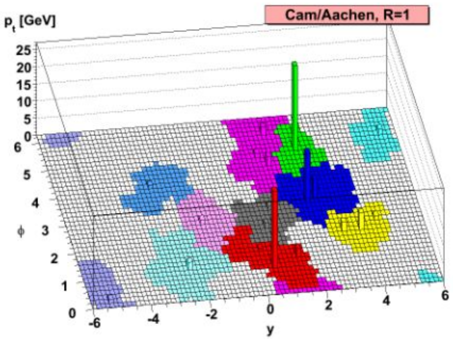
Kt (p=2)



Anti-Kt (p=-2)



C/A (p=0)



R_{22} code implementation

Calibrated UFO jet collection is used.

Samples:

mc20_13TeV.3647[2,12].Pythia8EvtGen_A14NNPDF23LO_jetjet_JZ[2,12]WithSW.deriv.DAOD_JETM2

mc20_13TeV.801661.Py8EG_A14NNPDF23LO_Zprime_tt_flatpT.deriv.DAOD_JETM2

///mc20_13TeV.801859.Py8EG_A14NNPDF23LO_WprimeWZ_flatpT.deriv.DAOD_JETM2

R_{22} code implementation

Other samples with other mc generators

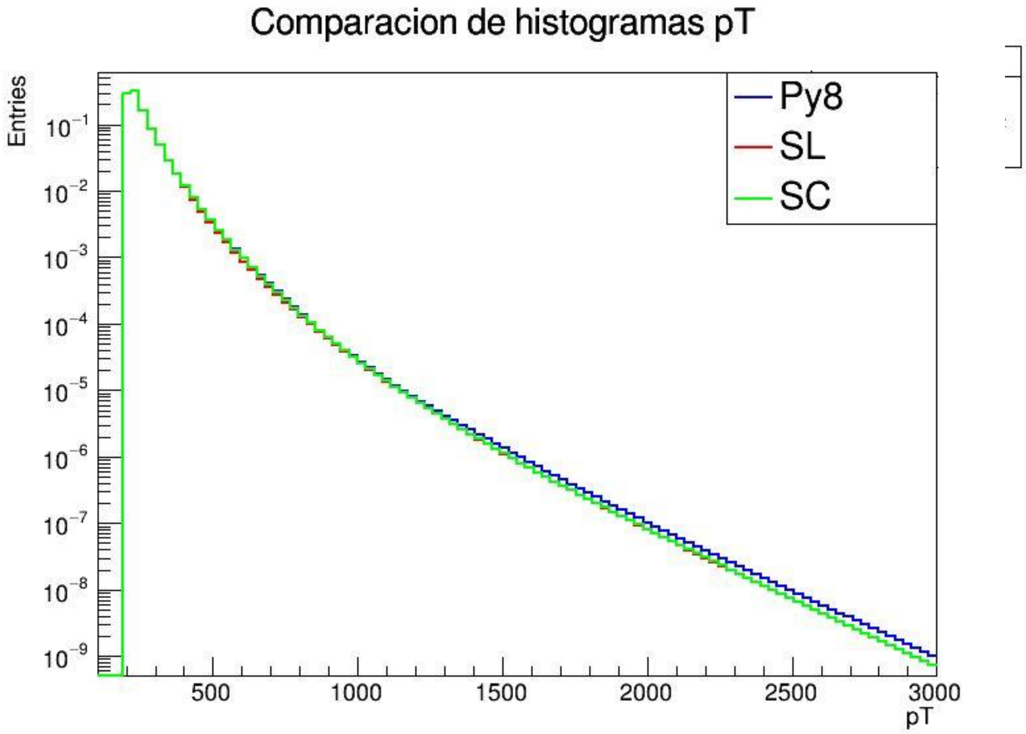
Samples (QCD):

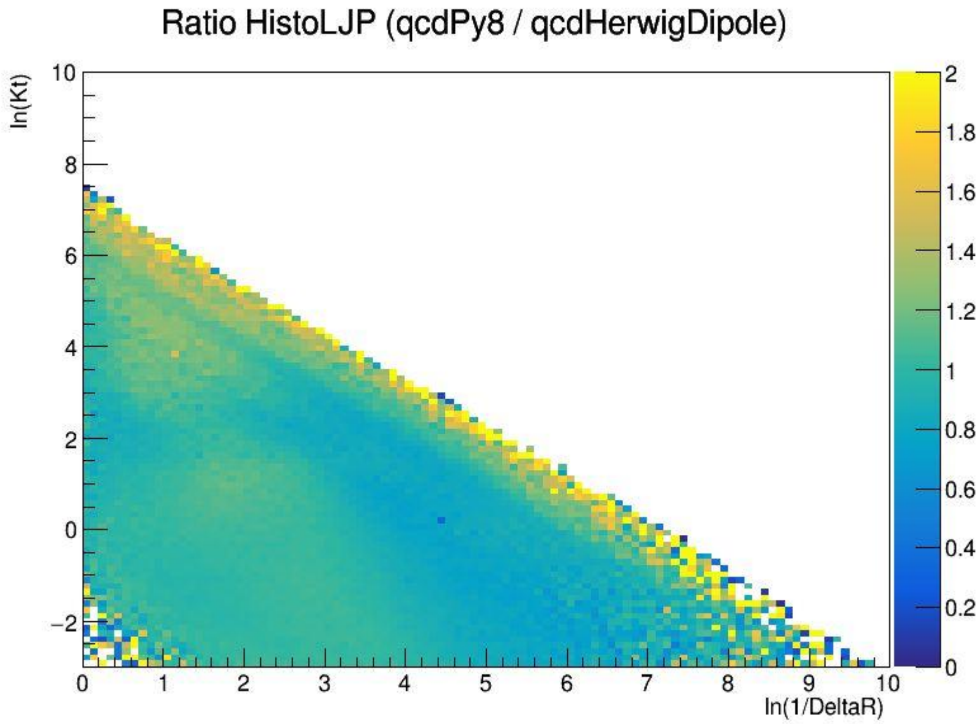
Sherpa-Lund:
mc20_13TeV.364686.Sherpa_CT10_CT14nnlo_CSShower_Lund_2to2jets_JZ[1:9]W.deriv.DAOD_JETM2.e6997_s3681_r13144_p5548

Sherpa-Cluster:
mc20_13TeV.364677.Sherpa_CT10_CT14nnlo_CSShower_2to2jets_JZ[1:9]W.deriv.DAOD_JETM2.e6997_s3681_r13144_p5548

Herwig Dipole:
mc20_13TeV.364902.H7EG_Matchbox_dipole_jetjetNLO_JZ[1:9]WithSW.deriv.DAOD_JETM2.e7482_s3681_r13144_p5548

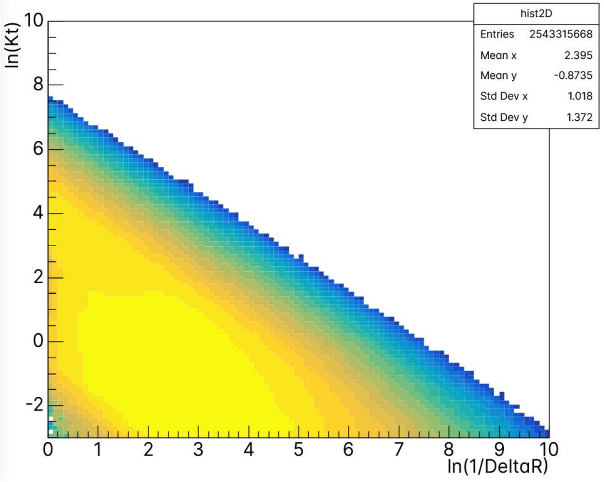
Comparison of Pt distribution



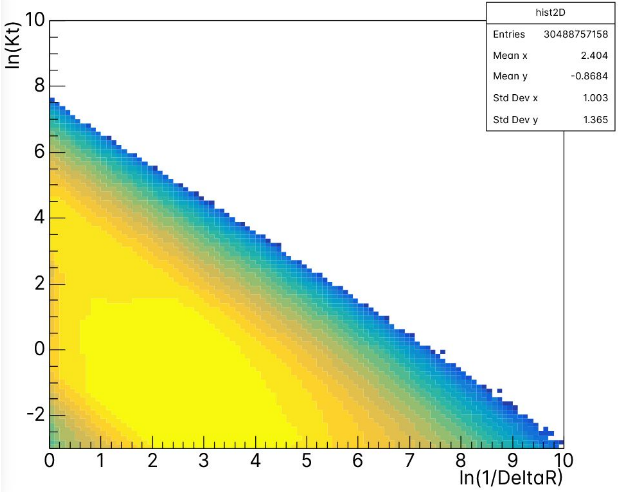

Ratio HistoLJP (qcdPy8 / qcdHerwigDipole)
ln(Kt)
ln(1/DeltaR)
10
8
6
4
2
0
−2
0
1
2
3
4
5
6
7
8
9
10
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0

Lund plane comparison:

Sherpa-Lund:

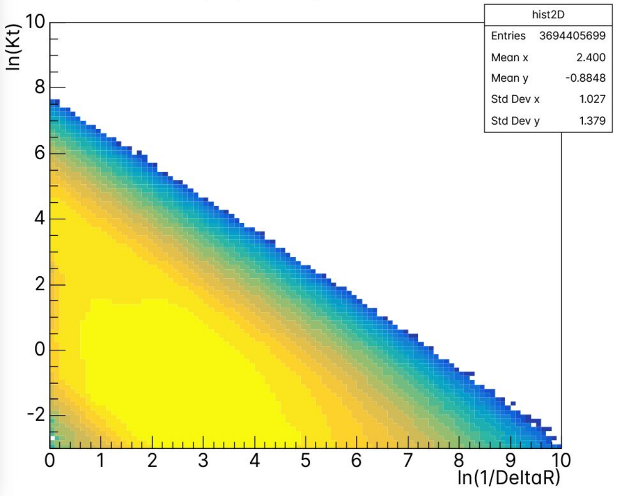


Pythia8:

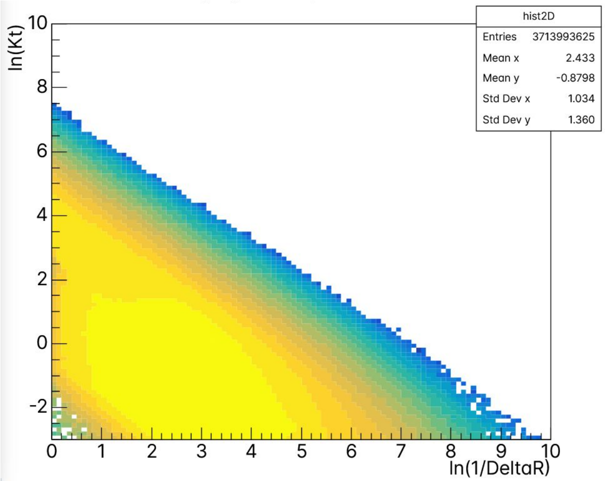


Lund plane comparison:

Sherpa-Cluster:

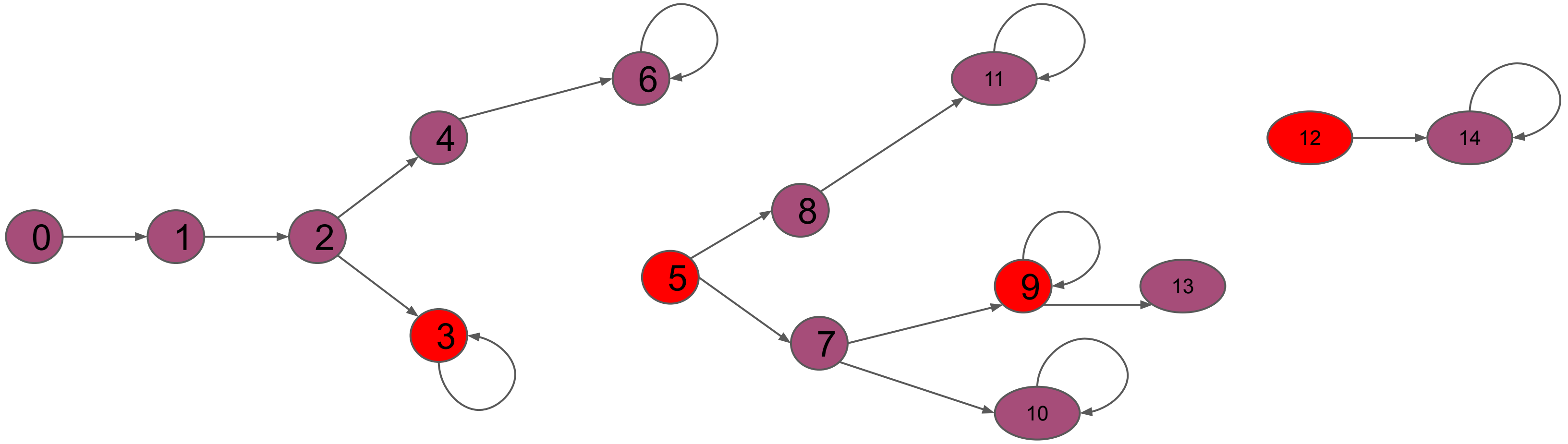


Herwig-Dipole



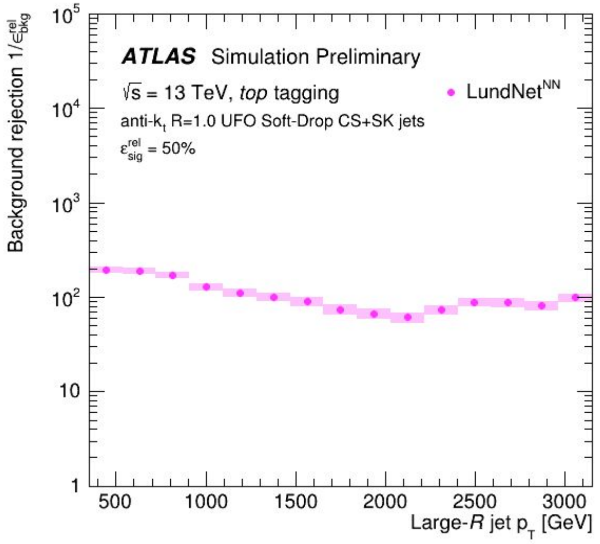
Disconnected Graphs in the Old Code

```
id=0 pt=1983.15  pt1=1083.78 idp1=-1  pt2=900.001 idp2=1
id=1 pt=900.001  pt1=894.165 idp1=2  pt2=5.83737 idp2=-1
id=2 pt=894.165  pt1=836.74 idp1=3  pt2=57.4251 idp2=4
id=3 pt=836.74  pt1=836.74 idp1=3  pt2=1.13144e-19 idp2=-1
id=4 pt=57.4251  pt1=47.887 idp1=-1  pt2=9.53886 idp2=6
id=5 pt=836.74  pt1=625.917 idp1=7  pt2=210.838 idp2=8
id=6 pt=9.53886  pt1=9.53886 idp1=6  pt2=3.47278e-20 idp2=-1
id=7 pt=625.917  pt1=618.913 idp1=9  pt2=7.00447 idp2=10
id=8 pt=210.838  pt1=202.719 idp1=-1  pt2=8.1196 idp2=11
id=9 pt=618.913  pt1=618.913 idp1=9  pt2=2.83072e-19 idp2=13
id=10 pt=7.00447  pt1=7.00447 idp1=10  pt2=8.68791e-20 idp2=-1
id=11 pt=8.1196  pt1=8.1196 idp1=11  pt2=3.52435e-20 idp2=-1
id=12 pt=618.913  pt1=608.354 idp1=-1  pt2=10.5586 idp2=14
id=13 pt=2.83072e-19  pt1=1.89236e-19 idp1=-1  pt2=9.38369e-20 idp2=-1
id=14 pt=10.5586  pt1=10.5586 idp1=14  pt2=5.12171e-20 idp2=-1
```

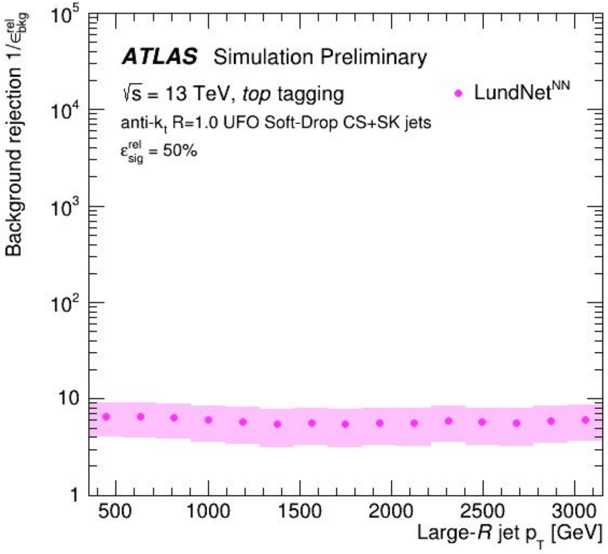


By allowing 1-node graphs

No kt cut



ln(kt) > -0.5




```
edge = torch.tensor([[0], [0]], dtype=torch.int64)
```

$$h_\Theta(\mathbf{x}_i, \mathbf{x}_j) = \bar{h}_\Theta(\mathbf{x}_i, \mathbf{x}_j - \mathbf{x}_i).$$