Strange Jet Tagging at FCCee using CNNs

Freya Blekman (DESY+UHH), Florencia Canelli (UZH), Kunal Gautam (VUB+UZH)
Eduardo Plörer (VUB+UZH), A.R. Sahasransu (VUB), Lode Vanhecke (VUB)

Why s-tagging?

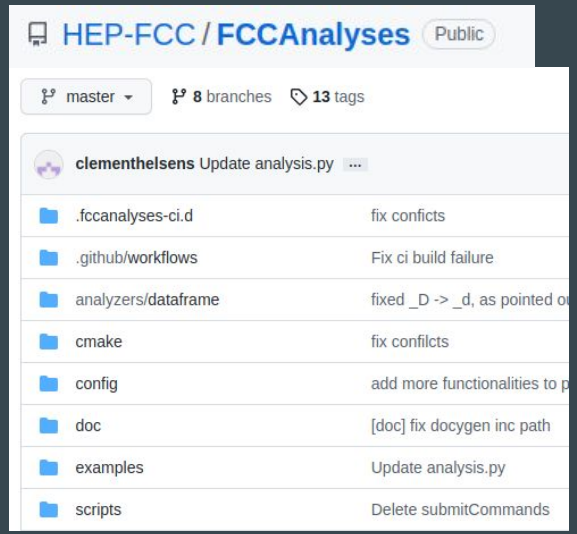
- Separating s- and u/d-jets is one of hardest problems in jet flavour tagging
- Z→ss decay width measurement, potential Higgs→strange studies, tool for various BSM studies (FCNC, etc)
- Improvement in s-tagging with the use of ML
- Good detector performance test

Related Work

- There have been similar studies for hadron colliders, and ongoing studies for e+e- colliders.
  - **2003.09517 (Nakai et. al.)**
  - 1811.09636 (Duarte-Campderros et. al.)
  - 2011.10736 (Erdmann et. al.)
  - And others
  - See also this talk (after this) from FCC-ee physics meeting

Spring2021 IDEA Samples

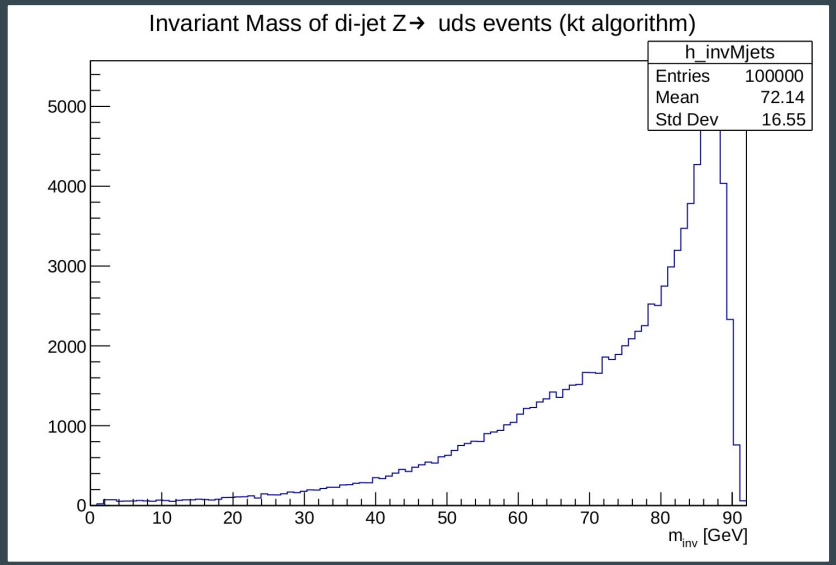
3	p8_ee_Zuds_ecm91	1,000,000,000
4	p8_ee_Zcc_ecm91	1,000,000,000
5	p8_ee_Zbb_ecm91	1,000,000,000

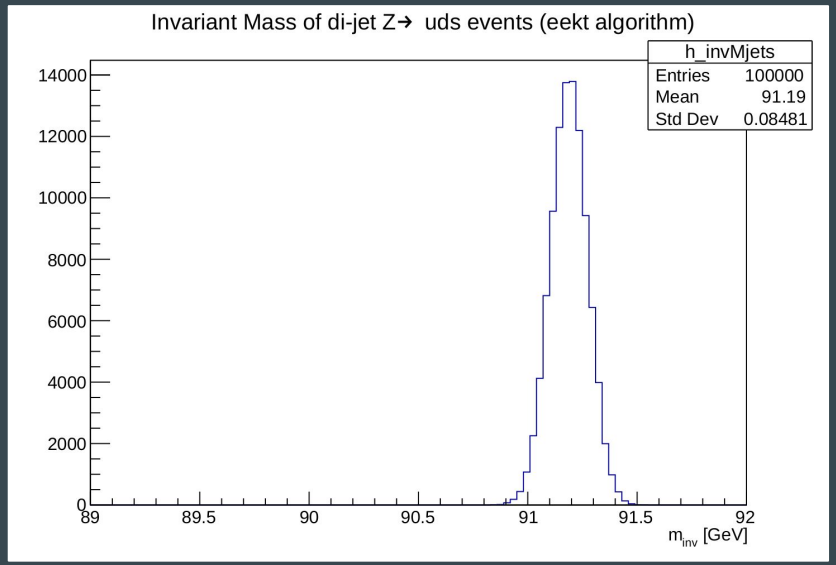


- Preselection in FCCAnalyses to make ntuples with 700,000 Zuds events (including both Reco-level and Gen-level information)
  - Trained the CNN model with these 100,000 events
  - Tested the model on the next 200,000 events
  - The low event numbers are used for these proof-of-principle studies
  - Eventually run on large sample
- Subsequent analysis and preparation of jet images performed in Coffea framework

Jet Clustering: Choice of Algorithm

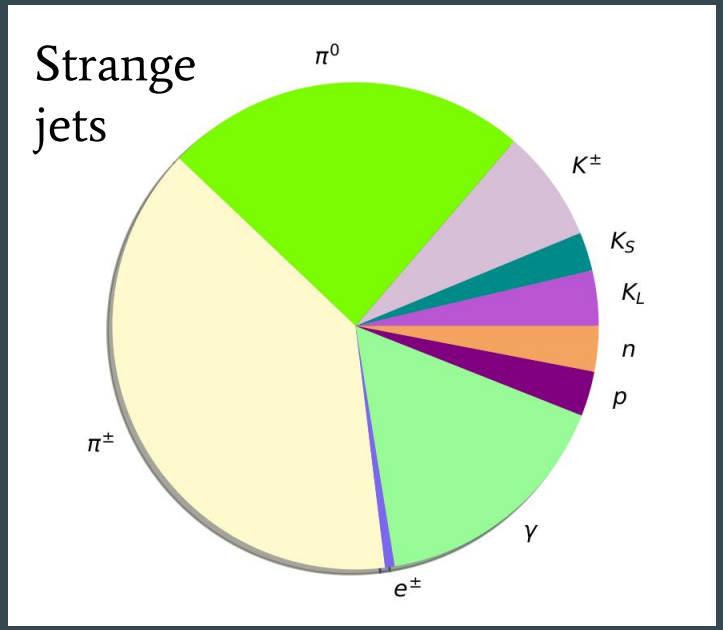
Exclusive clustering into exactly 2 jets using E-scheme for 100,000 Z⟶uds events

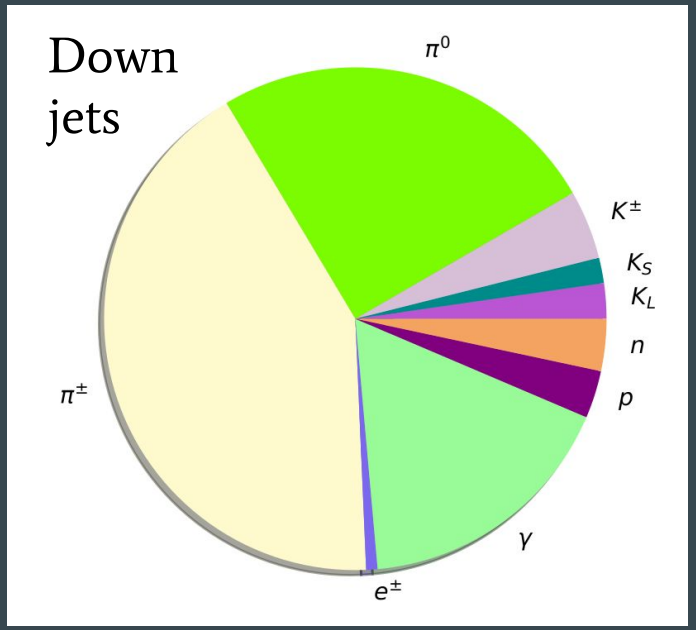




Due to absence of a distance parameter with respect to the beam in the *eekt-algorithm*, particles close to the beam are not thrown away but clustered into jets, unlike the *kt-algorithm*; therefore jet axis and jet cone spread depend on the choice of clustering algorithm.

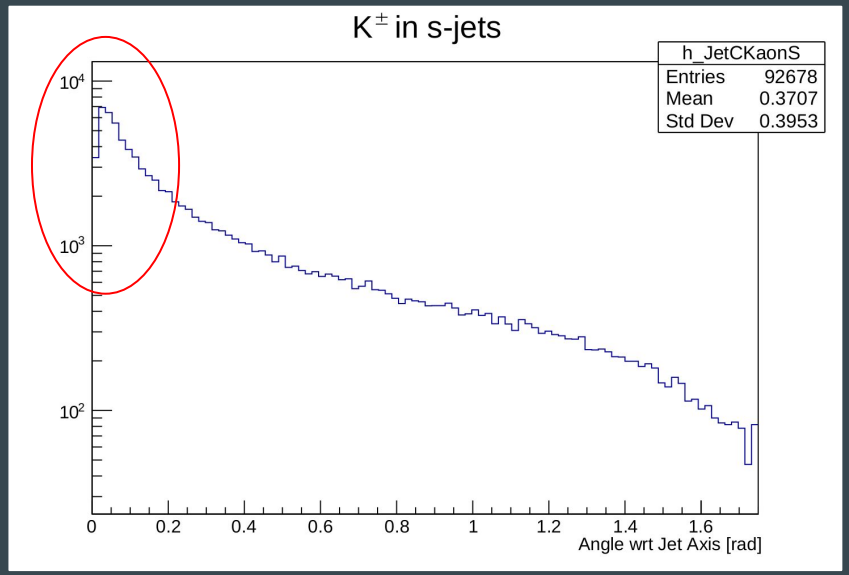
Jet Constituents: Gen-level Distribution

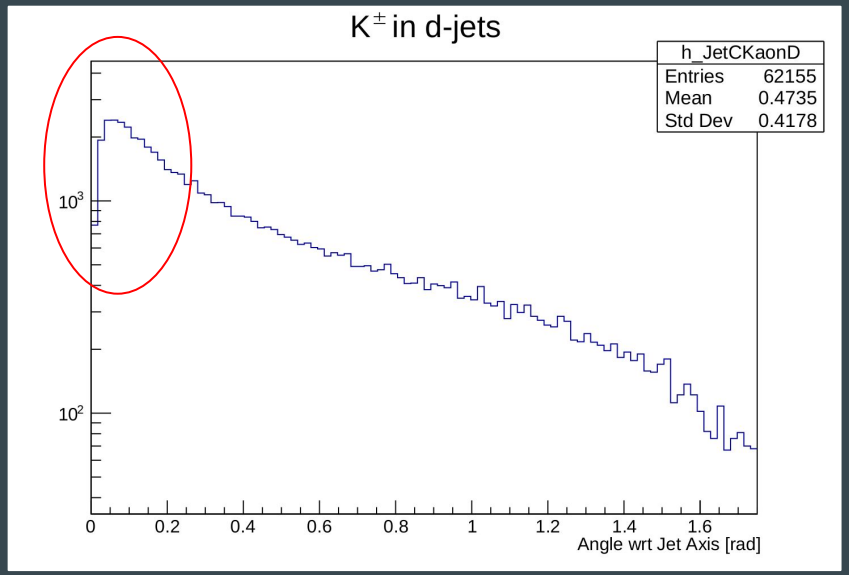




- Jet constituents have following basic cuts applied: pT> 500 MeV and |cos(θ)| < 0.97 (~=14°)
- **s-jets tend to have more kaons, while the d-jets tend to have more pions**.

Angular Distribution of Gen-level Constituents





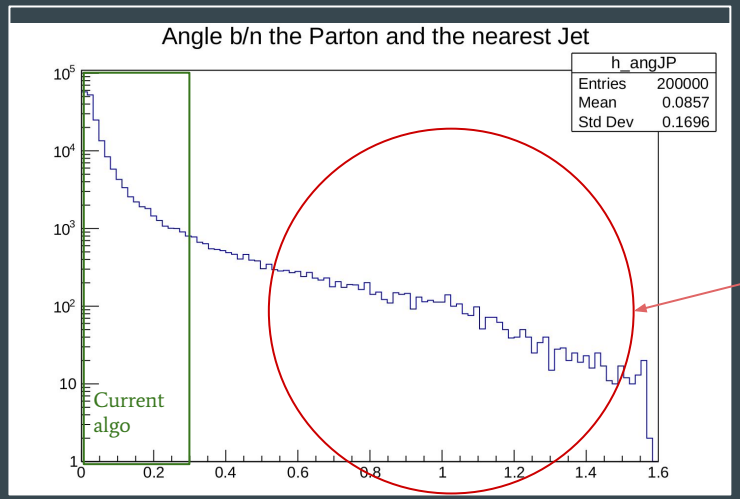
- Another potential distinguishing feature can be the angular distribution of these jet constituents around the jet axis.
- Kaons in the s-jets tend to be closer to the jet axis compared to those in d-jets.

Intermezzo:
Jet Flavour Assignment
&
Ghost Matching

(current) FCCAnalyses Jet Flavour Assignment

FCCAnalyses: JetTaggingUtils.cc:get_flavour()

```
Float_t dot = p.px()*parton.momentum.x+p.py()*parton.momentum.y+p.pz()*parton.momentum.z;
Float_t lenSq1 = p.px()*p.px()+p.py()*p.py()+p.pz()*p.pz();
Float_t lenSq2 = parton.momentum.x*parton.momentum.x+parton.momentum.y*parton.momentum.y+parton.momentum.z*parton.momentum.z;
Float_t norm = sqrt(lenSq1*lenSq2);
Float_t angle = acos(dot/norm);
if (angle <= 0.3) result[j] = std::max(result[j], std::abs ( parton.PDG ));
```



Flavour assignment seems derived for cone(-style) algorithms, not clear how appropriate for the eekT jets used here

These uds jets will either be inaccurately tagged or stay untagged with the current definition of flavour assignment.

Another potential issue is that gluon initiated jets are neglected -> need for a robust jet flavour defn.

Ghost Matching: Idea

In CMS ghost matching is used to map both hadrons and partons to jet flavours

The idea behind ghost matching is to **recluster the jets with "ghost" particles** that determine the jet's flavour

A ghost may be either a hadron or a parton (occurring earlier in the MC history)

> whose four momentum is rescaled by a small number (10^-18) such that its inclusion in the clustering step does not change the kinematics of the resulting jets

Jet flavour is determined via the following (simple) recipe

- If b hadron is clustered -> b jet → If b parton is clustered -> b jet
- If no b, but a c hadron is clustered -> c jet → If no b, but a c parton is clustered -> c jet
- If neither a b nor a c hadron are clustered -> light jet → If neither a b nor a c hadron are clustered -> hardest ghost parton determines flavour

Hadron flavour takes precedence

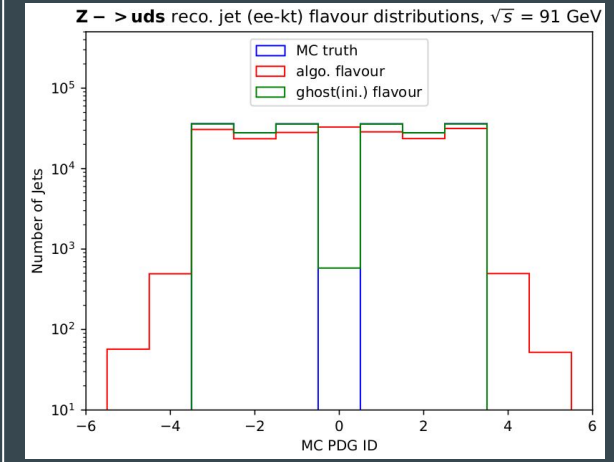
Ghost Matching: Our Implementation

Currently, we have implemented the ghost matching definition in a local version of FCCAnalyses

As our studies are on light jets we use the parton flavour

Ghost partons are defined as partons with **status code 21-29** (partons from the hardest subprocess) {Pythia8}

- Choice of ghost partons as those coming from the hardest subprocess is motivated by our physics case Z->uds, and the fact that we are exclusively clustering
- Expect potential subjets arising from showering to be merged
- Jets with flavour ‘0’ have both partons clustered into the other jet in the event



Ghost Matching: Status Codes + Pros/Cons

A more general approach is to consider partons right before they are hadronized (i.e. after showering). These correspond to **status code 71-79**

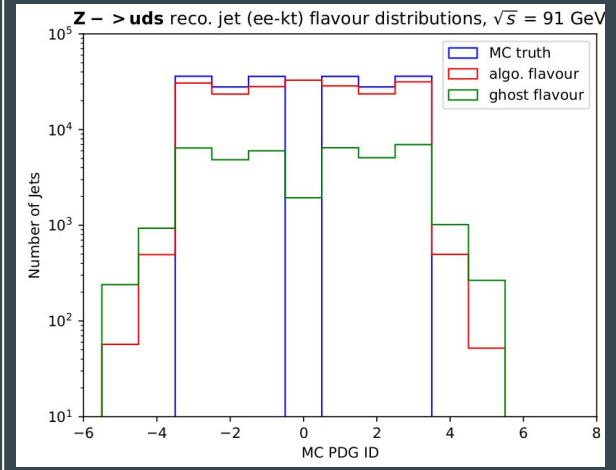
Currently implemented as an option that can be selected when making nTuples but not used in these studies

Pros -> Can reduce number of unmatched jets for non-cone like algorithms, Can include gluon jets

Cons -> Current ghost definition has predilection for heavy flavours (even when the partons are soft + off-axis)

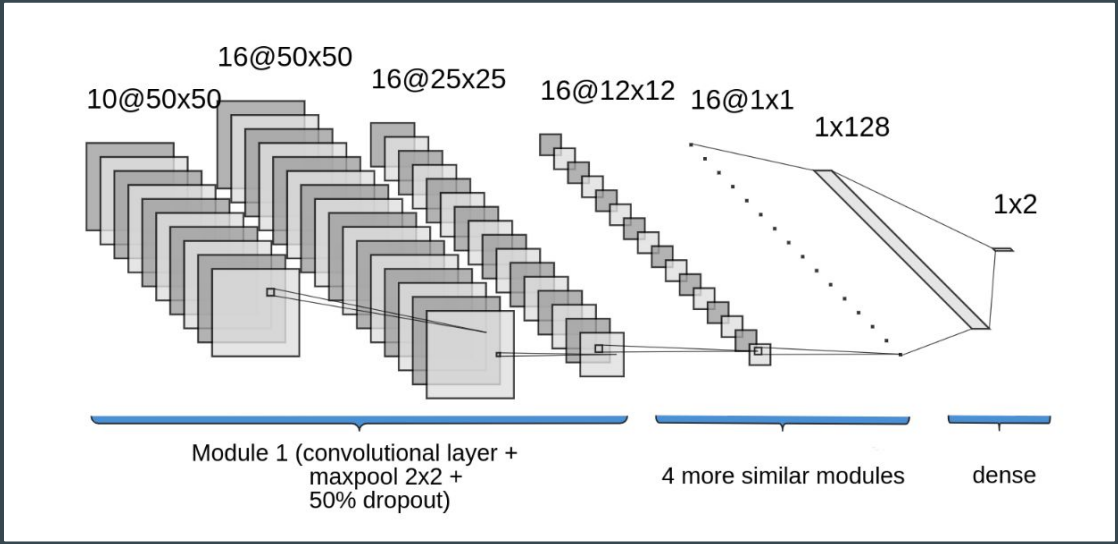
Exact ghost definition is analysis dependent and probably needs tuning

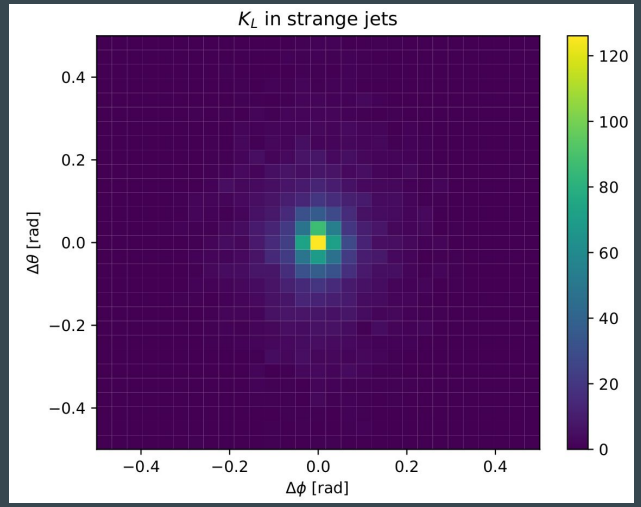
=> Code can be pushed to central repo if others are interested



The rest of this talk uses the **status 23** definition of the ghost matching

Jet Tagging with a CNN





*based on Lode Vanhecke's Bachelor Thesis at VUB (July 2021)

Network Structure:

- Similar structure as above, except the image resolution changed from 50x50 pixels to 29x29 pixels and one fewer module
  - 5-7 inputs depending on detector scenario
  - 4 modules with a convolutional layer (16 filters), a maxpool layer, and a 50% dropout layer
  - Flattened to a dense layer
  - Fully connected to the output layer (3 nodes) with softmax activation function

Reminder: Gen-level CNN

Jet Images from 10 Categories

1. $K^{\pm}$
2. $\pi^{\pm}$
3. K_L
4. $e^{\pm}$
5. $\mu^{\pm}$
6. γ
7. p
8. n
9. $K_s \rightarrow \pi^+\pi^-$
10. $\pi_0 \rightarrow \gamma\gamma$

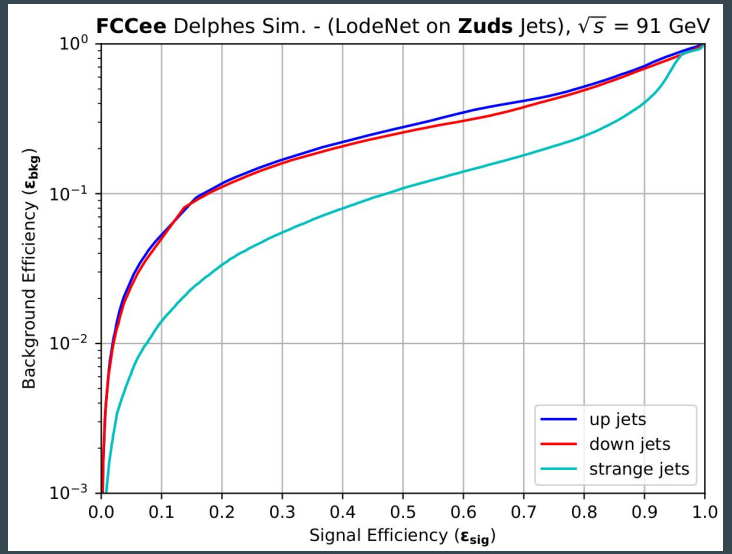
Each image is 29x29 pixels, encompassing the range (-0.5, 0.5) radians in the θ-ϕ space centered around the jet axis

θ-ϕ distribution is weighted by a normalisation factor = |p| (constituent)/ |p| (jet)

To make the tagger less dependent on momentum (we still need to check if this is true)

Details of algorithm in FCC-ee physics performance meeting talk october

Performance at 10% fake rate is ~50%



Reco-level CNN Input

Jet Images using particle flow candidates

1. charged hadron
2. neutral hadron
3. $e^{\pm}$
4. $\mu^{\pm}$
5. γ
6. $K_s \rightarrow \pi^+\pi^-$
7. $K^{\pm}$

Inspired by assumptions of particle ID (no fake rate etc included)

Present centrally produced samples have no general classification of PF candidates

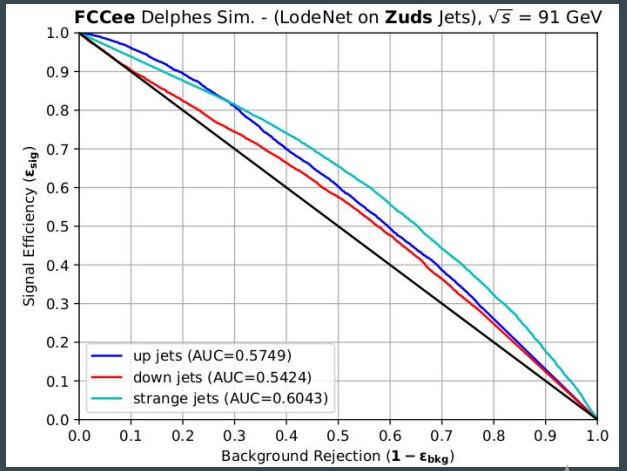
- Fix: Particle type assigned via associated MC particle (gen-level)

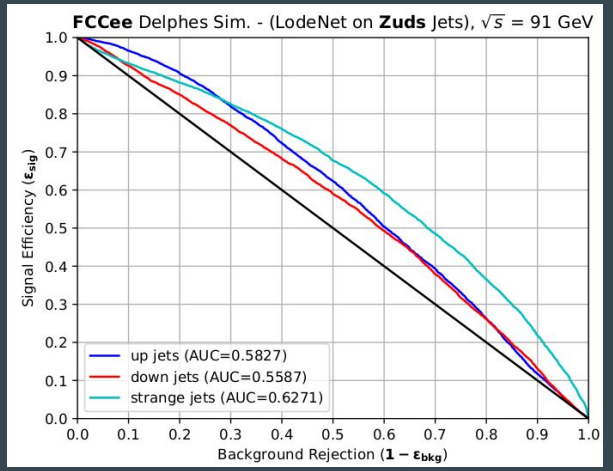
Three PID scenarios are defined:

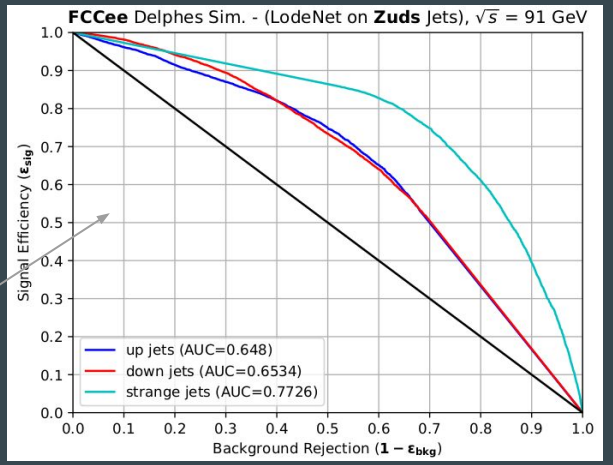
1. No PID beyond PF: Categories 1-5
2. PF+Ks: Categories 1-5 + K-shorts reconstructed from two pions (matched to gen level K-short, efficiency approximately ⅔)
3. PF+Ks+K±: Categories 1-5 + K-shorts + charged Kaons (by matching to MC particle)

Performance

Three working points defined at fake rates of **10%**, **5%**, and **1%**, respectively







Signal Efficiency	10% fake rate	5% fake rate	1% fake rate
Generator	47.2%	27.7%	7.5%
PF only	17.7%	9.7%	2.0%
PF + Ks	21.9%	12.9%	4.4%
PF + Ks + K+-	39.5%	24.8%	7.0%

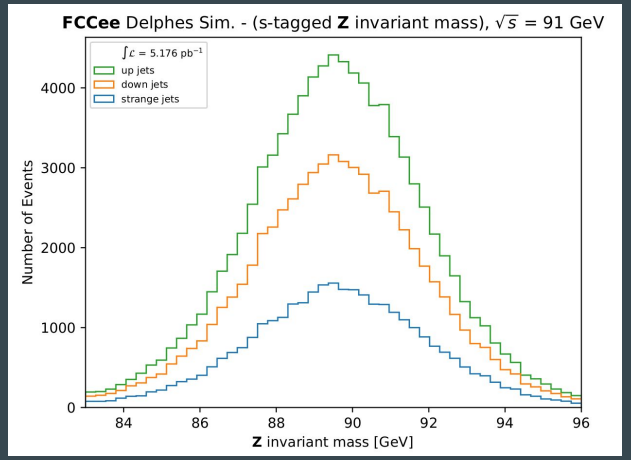
Our Suggestions on Detector Requirements

- A sub-detector for PID would improve tagging performance (R. Forty @ FCC Week'21)
- Timing Detector to identify soft charged-Kaons
- Vertex and tracking detector to reconstruct K_s, Lambda_0, and pi_0 accurately

Future studies can include representative efficiencies/fake rakes for Ks, Lamba_0, etc to get direct connection tagger performance ⟷ detector performance

Summary

- First look at a CNN based s-tagger on Spring2021 IDEA event samples
  - Studied jet constituent multiplicity and angular distribution around the jet axis as potential distinguishing variables
  - Implemented a CNN model in Tensorflow and trained with jet images from 100,000 events
  - Evaluated this trained network on the jets from 200,000 events to review its performance
  - 3 working points at fake rates of 10%, 5%, and 1%, with signal efficiencies of 47%, 28%, and 8% respectively in Gen-level
- Performed in FCCAnalyses + Coffea



ToDo -> More samples + include backgrounds

Back-up

Jet Clustering: MC truth Jet Flavour Assignment

	FCCAnalyses	GM (71-79)	GM (23)	MC Truth
s	593,859	779,398	718,799	720,898
d	595,055	588,638	716,600	718,604
u	474,399	556,247	558,859	560,498
c	8,316	36,159	0	-
b	824	4,814	0	-
untagged	327,547	34,744	5742	-
In a sample of 1M $Z \longrightarrow$ uds events, i.e. 2M uds jets				

This is a potential area for improvement in FCCAnalyses, since with the present definition of flavour assignment, a significant number of jets are not assigned a flavour and there are others which are mistagged.

Clustering the jets inclusively may reduce this effect. Though it hasn't been checked yet.

Development and analysis of a strange quark tagger for future electron-positron colliders

Lode Vanhecke

June 4, 2021

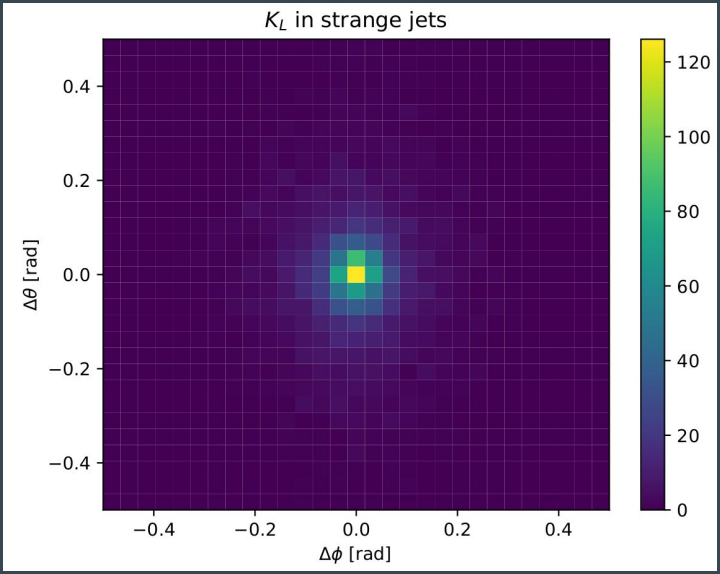
*Lode Vanhecke's Bachelor Thesis at VUB

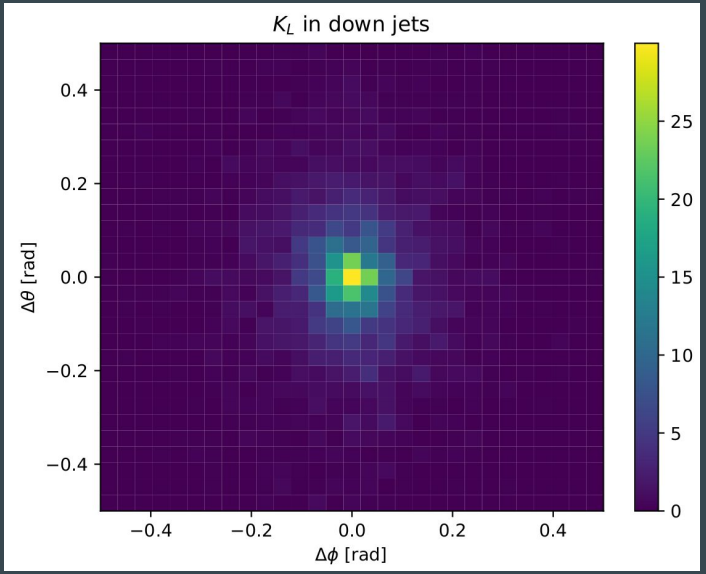
Developing on the work by Lode Vanhecke and AR Sahasransu:

- Event generation using MadGraph(v2.6.6) and Pythia(v8.243)
  - Now using FCCee samples
- Using **gen level** information only
- Here jet clustering was done using anti-kt algorithm with a radius of 0.4
  - Upgraded in the study presented today to eekt (Durham) algorithm
- 2D Angular distribution of jet constituents around the jet axis as the distinguishing variable
  - Particle ID assumptions
- **No decay lengths/lifetime info used**
- Strange jet tagging using a CNN

Strategy/This talk: To confirm/reproduce the results in that thesis using Spring2021 FCCee IDEA samples at reco level

Jet Images





- Images are made for different constituent types (next slide)
- θ-ϕ distribution is weighted by a normalisation factor = |p| (constituent)/ |p| (jet)
  - To make the tagger less dependent on momentum (we still need to check if this is true)
- Each image is 29x29 pixels, encompassing the range (-0.5, 0.5) radians in the θ-ϕ space centered around the jet axis

Gen-level CNN Training

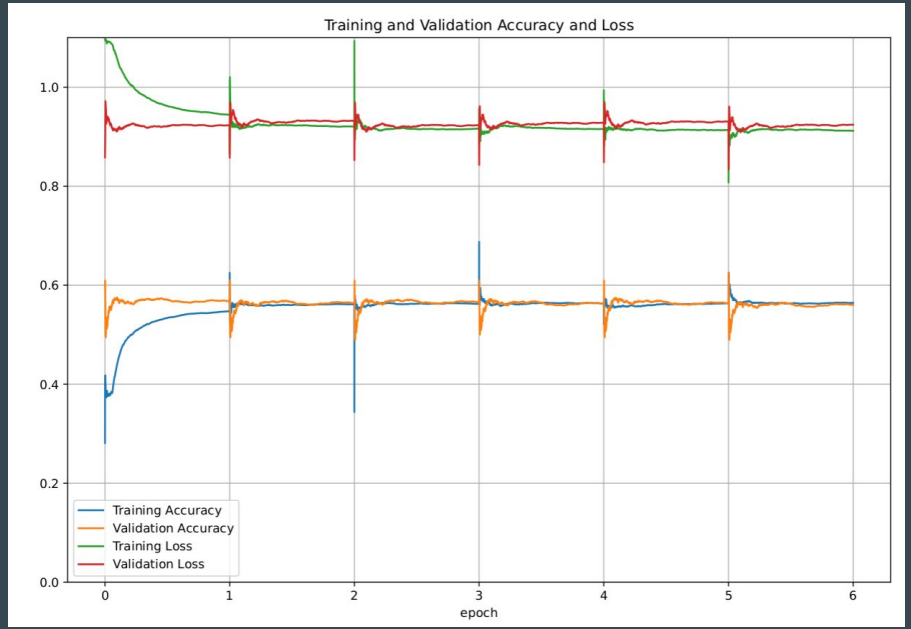
- Trained the CNN with 100,000 di-jet Z⟶uds events, i.e. 200,000 jets.
- Implemented in Tensorflow
- ~14k trainable parameters (lightweight)

  -> No obvious overfitting/overtraining

- Categorical cross entropy as loss function

$$L(\mathbf{y}, \mathbf{p}) = -\Sigma_i^C y_i log(p_i)$$

(**y** is true class label vector, and **p** is network prediction vector)



Reco-level CNN Training

- Trained the CNN with 100,000 di-jet $Z \rightarrow uds$ events, i.e. 200,000 jets.
- Implemented in Tensorflow
- ~14k trainable parameters (lightweight)

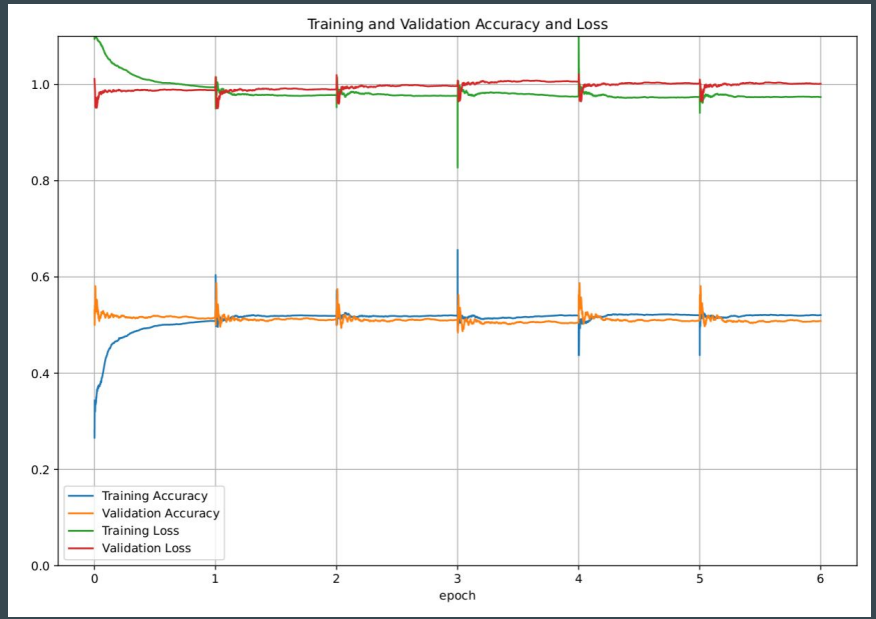
  -> No obvious overfitting/overtraining

- Categorical cross entropy as loss function

$$L(\mathbf{y}, \mathbf{p}) = -\Sigma_i^C y_i log(p_i)$$

(**y** is true class label vector, and **p** is network prediction vector)

- Very little hyperparamter tuning



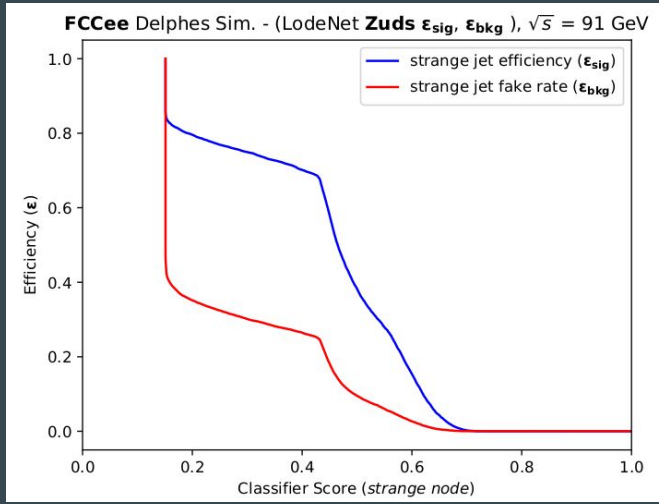
Working points

Three working points defined at fake rates of 10%, 5%, and 1%, respectively

Gen.	10% fake rate	5% fake rate	1% fake rate
Classifier Score (strange node)	0.5955	0.6320	0.6480
Signal Efficiency	47.2%	27.7%	7.5%

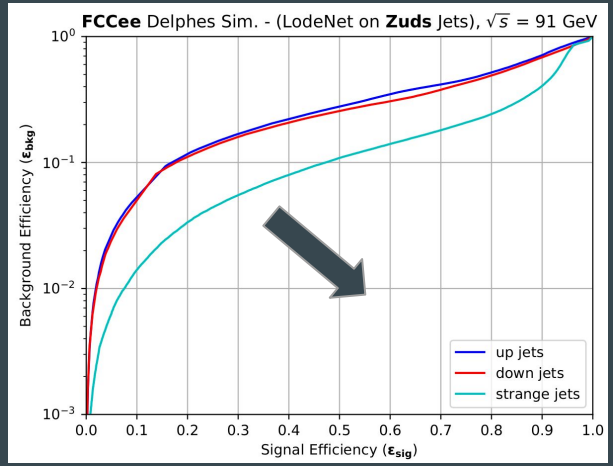
Reco (PF only)	10% fake rate	5% fake rate	1% fake rate
Classifier Score (strange node)	0.4138	0.4516	0.5025
Signal Efficiency	17.7%	9.7%	2.0%

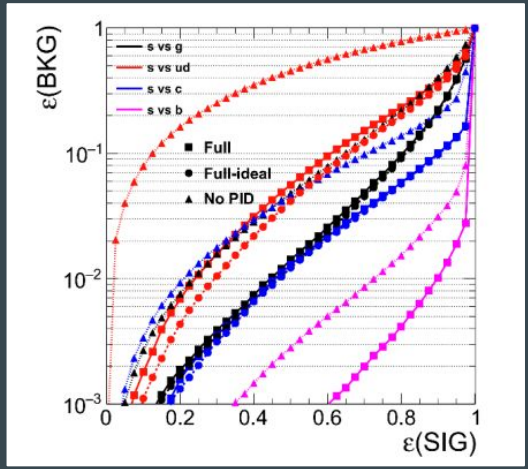
Reco (PF + K0s + K+-)	10% fake rate	5% fake rate	1% fake rate
Classifier Score (strange node)	0.4952	0.5652	0.6344
Signal Efficiency	39.5%	24.8%	7.0%



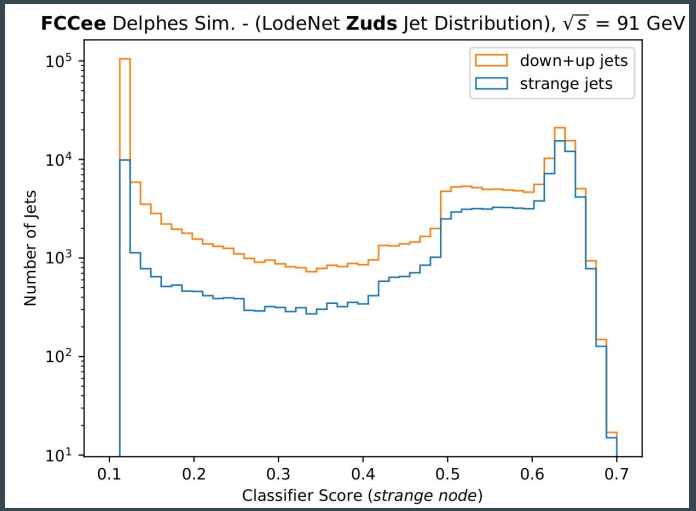
Gen-level Performance

While testing the model on 200,000 events





*compare to Michele Selvaggi (TOP2021)

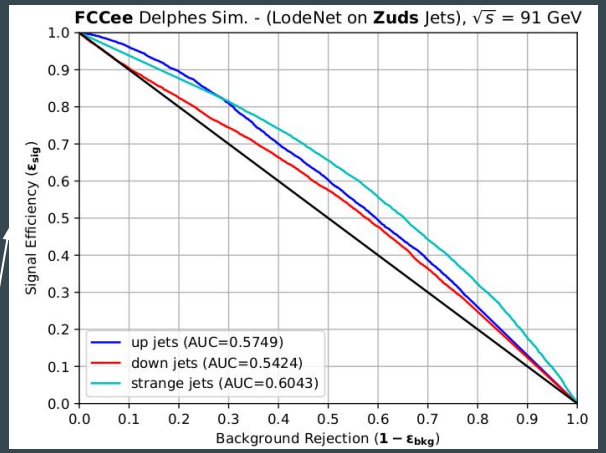


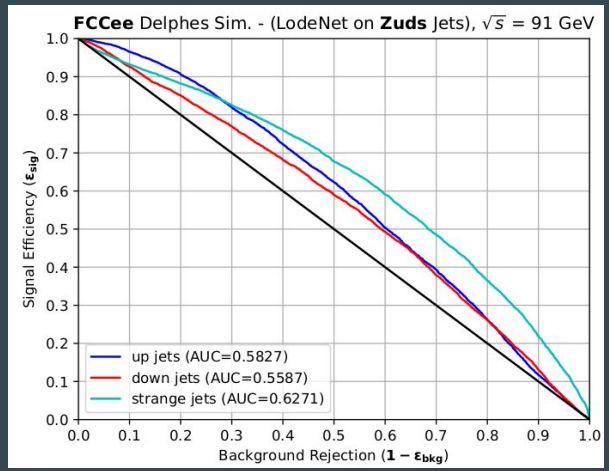
Performance at 10% fake rate is ~50%

Classifier score distribution that leads to these ROC curves

Reco-level Performance

While testing the model on 200,000 events

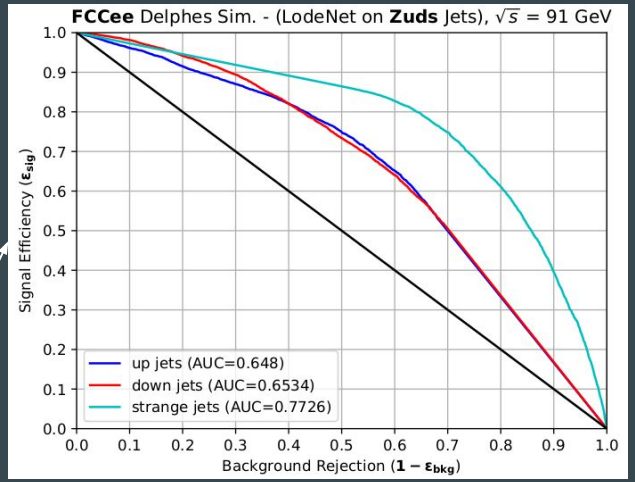




Classification performance with only the particle flow categories is very limited (though consistent with literature)

Extending the classifier with input with information when reconstructing K-shorts in the pi+pi- channel, performance improves only marginally (likely due to their scarcity)

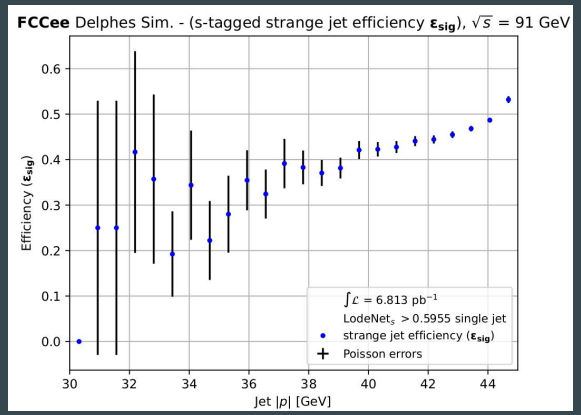
Extending the classifier with input with information discriminating charged Kaons provides a sizable boost in performance (almost on par with all gen categories)



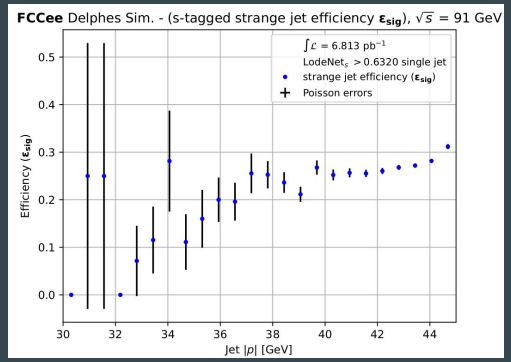
Signal Efficiency vs |p| (Gen-level)

- Increase in signal efficiency wrt |p| in 10% and 5% working points
- Little improvement in 1% working point

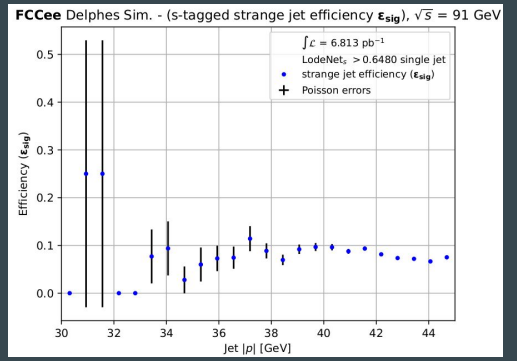
10%



5%



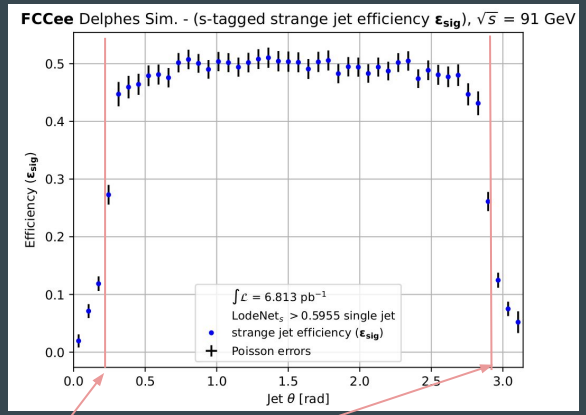
1%



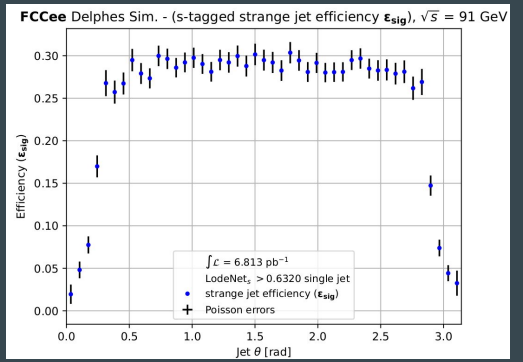
Signal Efficiency vs θ (Gen-level)

No signal efficiency dependence on the polar angle, up to forward/backward jets

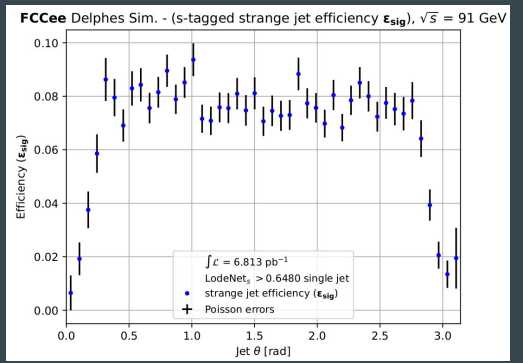
10%



5%



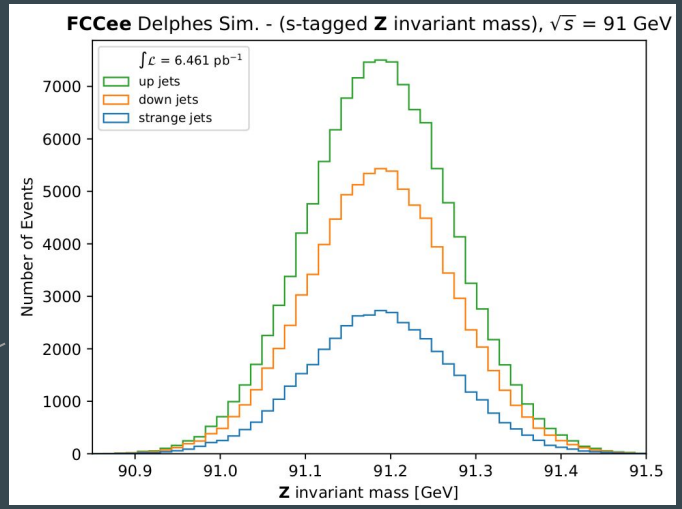
1%



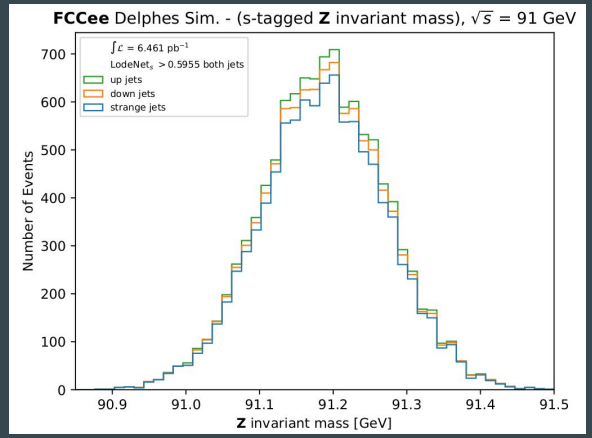
Drop in efficiency consistent with IDEA acceptance as cuts were introduced on particles before making the jet images

Gen-level Z Peak Reconstruction

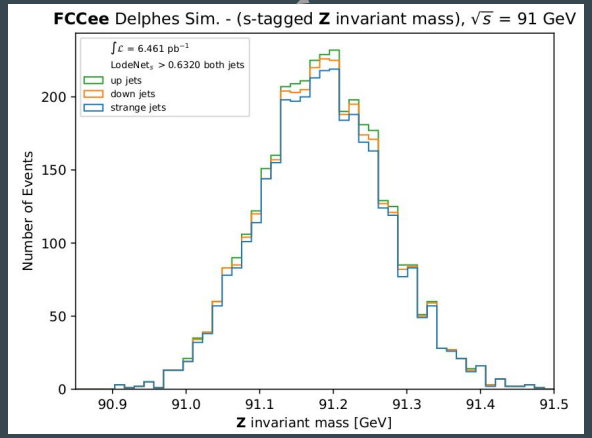
- Z peak before tagging vs after tagging both jets
- Ignored background and b/c quark decays (for now)



10%



5%



1%