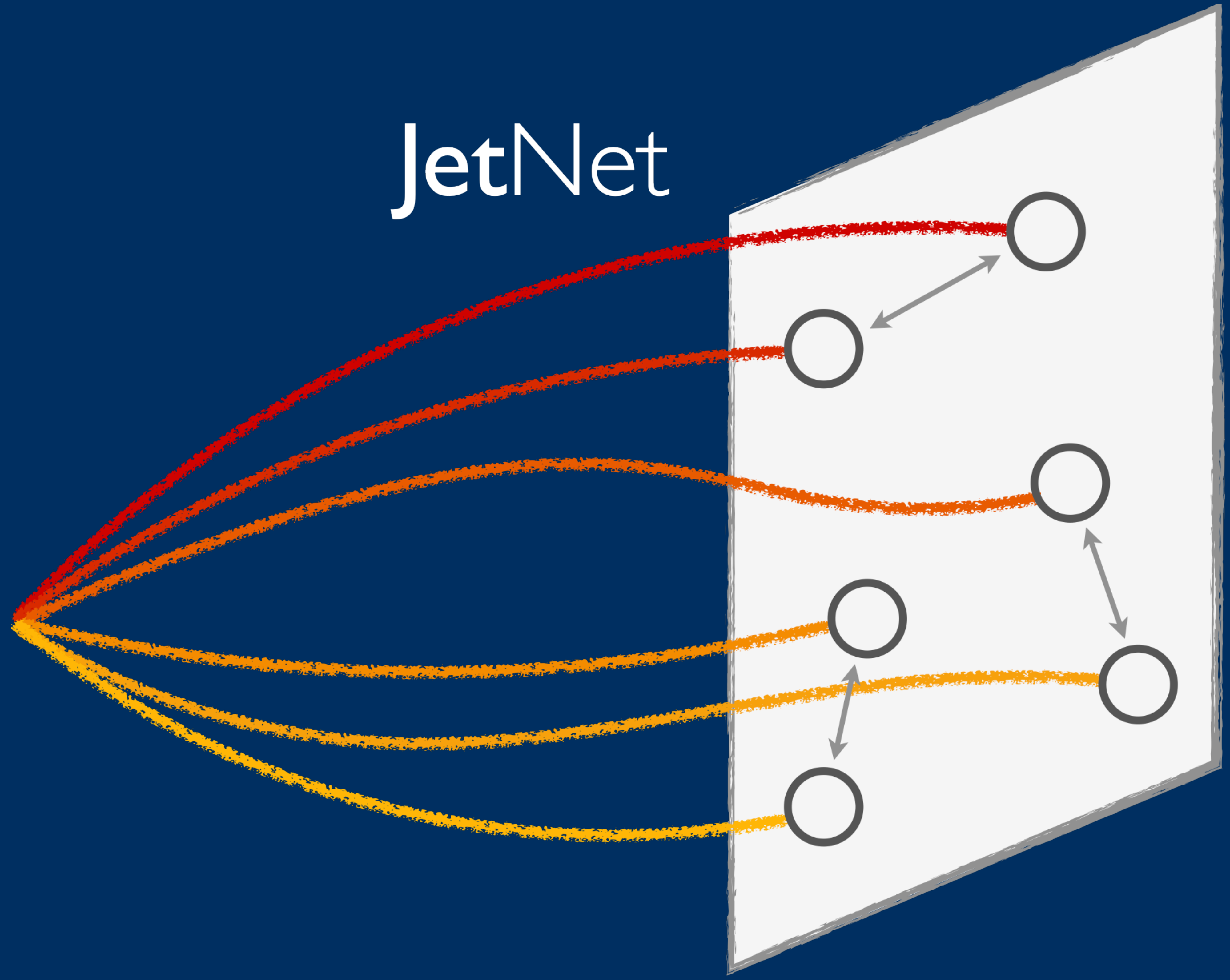


JetNet

CHEP

JetNet: Developing and reproducing ML + HEP Projects

```
pip install jetnet
```

Carlos Pareja, Raghav Kansal, Javier Duarte, UC San Diego, Duarte Lab

Introduction

Problems:

- How do I get started with machine learning in high energy physics?
- How do I evaluate my results?
- How do we reproduce and compare results?

Solution:

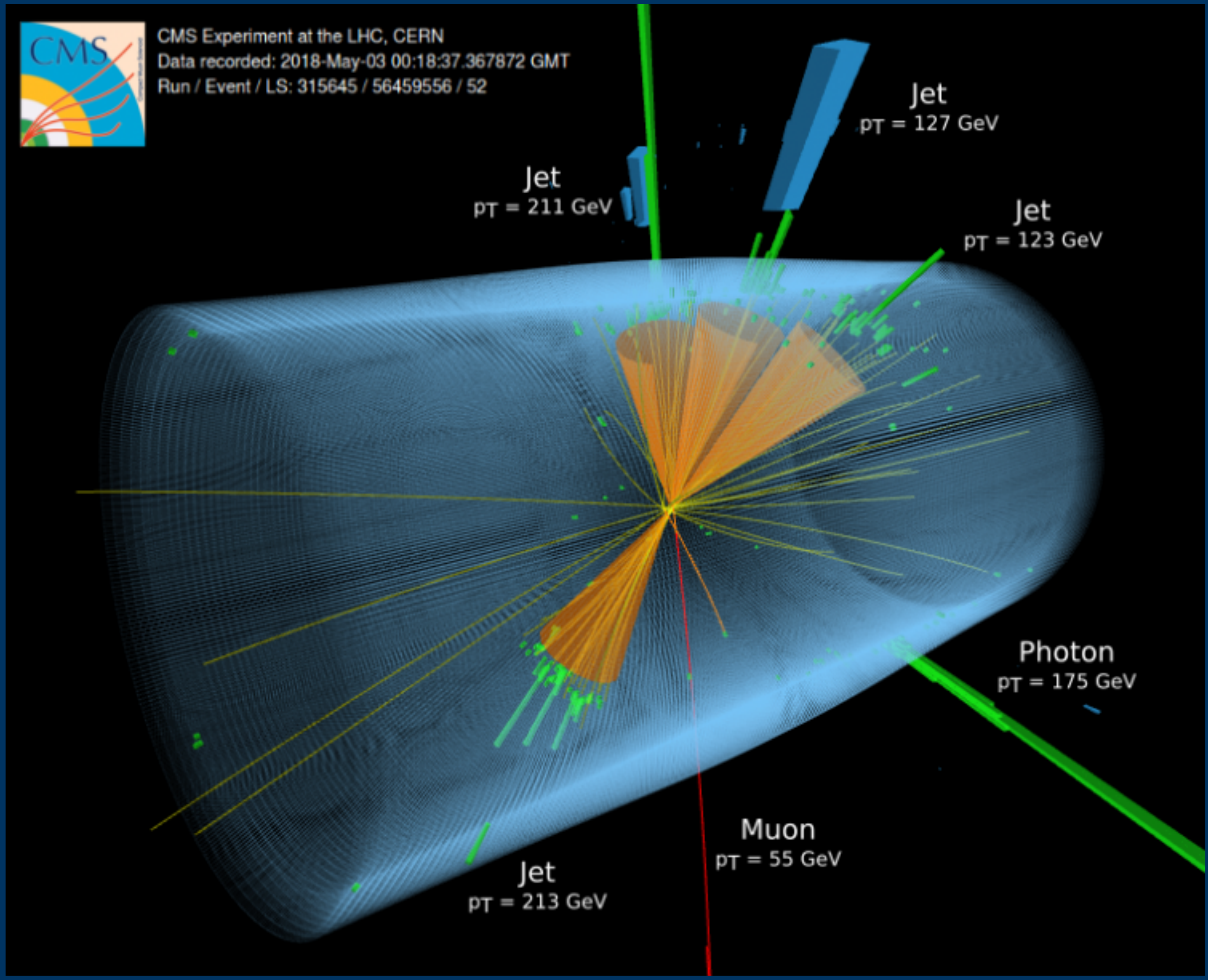
- JetNet: Python package with easy-to-access datasets, standardised evaluation metrics, and more utilities for improving accessibility and reproducibility in ML + HEP.
- JetNet is designed to allow the user to focus on building and training their model instead of worrying about the data being correctly formatted, etc.

Today

- Loading and looking at one of the JetNet datasets (JetNet)
- Preparing the dataset for training a model
- Evaluation Metrics
- Impact of JetNet
- Future of JetNet

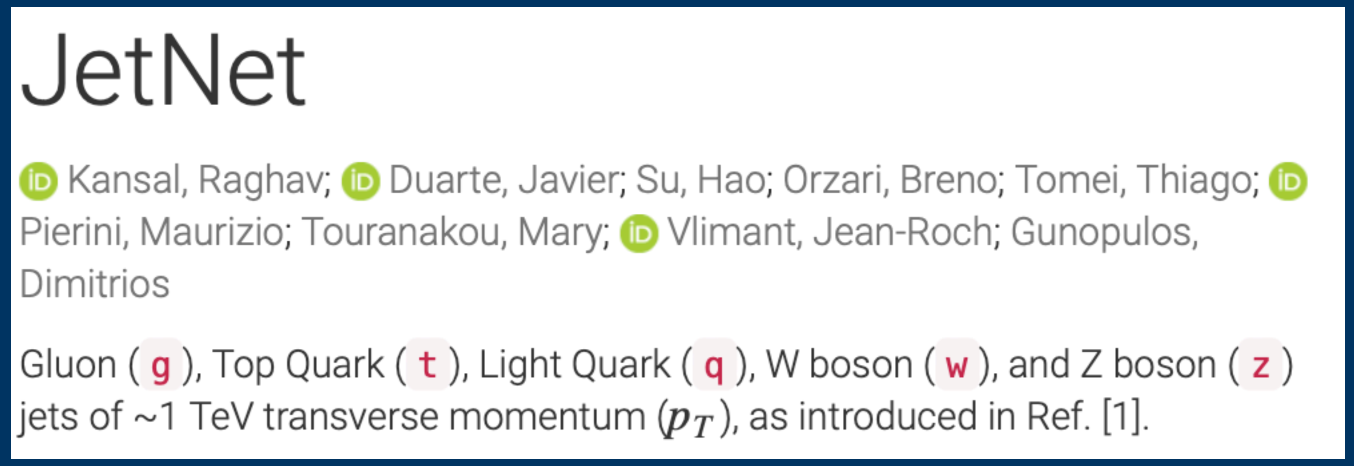
What are Jets? Why do we use Jets for Machine Learning?

- Jets are sprays of particles resulting from quarks and gluons produced in high-energy particle collisions
- We are often uncertain about the type and properties of the particle that produced the jet
- Jets can also be hard to model and reconstruct
- This is where ML comes into play!
- We can use ML for jet reconstruction, classification, regression, simulation, anomaly detection, etc.


CMS Experiment at the LHC, CERN
Data recorded: 2018-May-03 00:18:37.367872 GMT
Run / Event / LS: 315645 / 56459556 / 52
Jet p_T = 127 GeV
Jet p_T = 211 GeV
Jet p_T = 123 GeV
Photon p_T = 175 GeV
Muon p_T = 55 GeV
Jet p_T = 213 GeV

JetNet Datasets

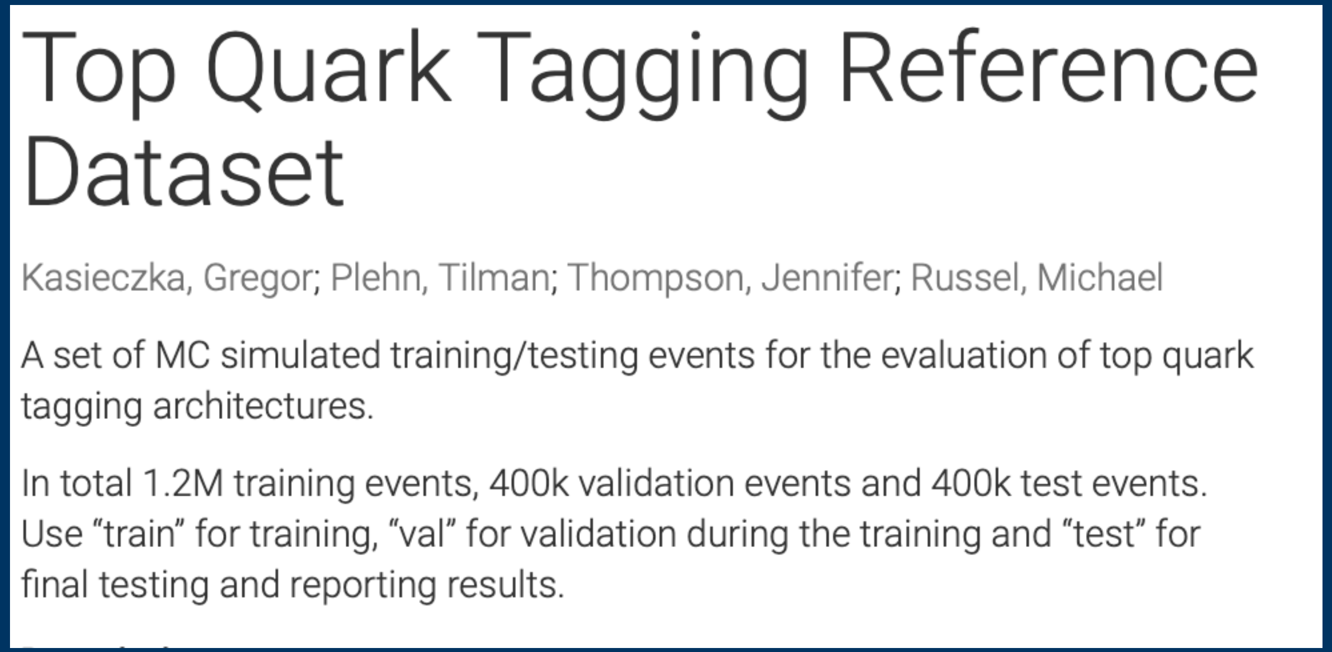
- **JetNet Dataset:**
  - Gluon, light quarks, top quarks, W bosons, Z bosons
- **TopTagging Dataset:**
  - QCD and Top Quark
- **QuarkGluon Dataset:**
  - Gluon and light quarks
- **JetClass Dataset (coming soon!):**
  - 10 different classes of jet types!



JetNet

Kansal, Raghav; Duarte, Javier; Su, Hao; Orzari, Breno; Tomei, Thiago; Pierini, Maurizio; Touranakou, Mary; Vlimant, Jean-Roch; Gunopulos, Dimitrios

Gluon (`g`), Top Quark (`t`), Light Quark (`q`), W boson (`w`), and Z boson (`z`) jets of ~1 TeV transverse momentum (p_T), as introduced in Ref. [1].

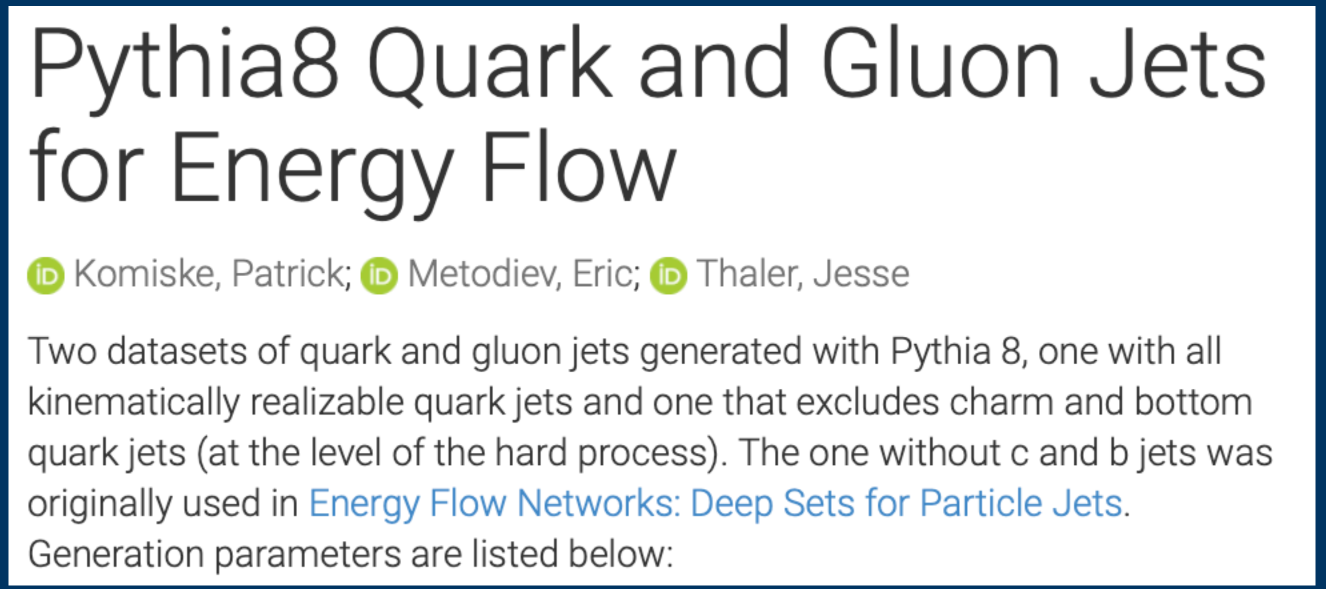


Top Quark Tagging Reference Dataset

Kasieczka, Gregor; Plehn, Tilman; Thompson, Jennifer; Russel, Michael

A set of MC simulated training/testing events for the evaluation of top quark tagging architectures.

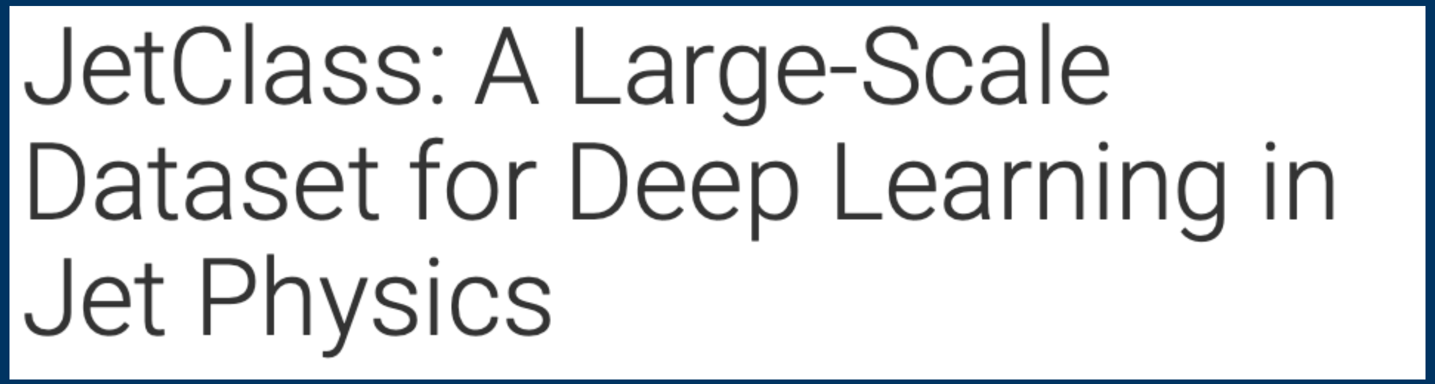
In total 1.2M training events, 400k validation events and 400k test events. Use "train" for training, "val" for validation during the training and "test" for final testing and reporting results.



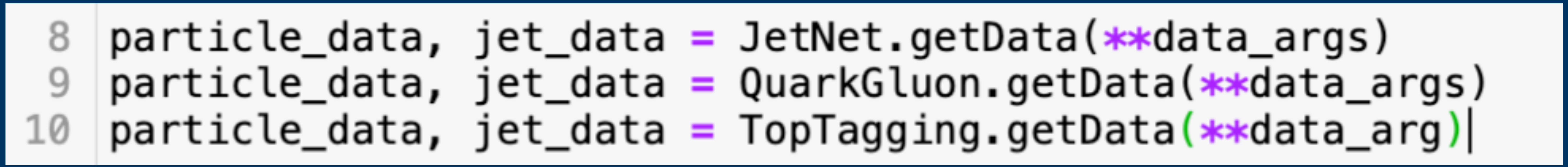
Pythia8 Quark and Gluon Jets for Energy Flow

Komiske, Patrick; Metodiev, Eric; Thaler, Jesse

Two datasets of quark and gluon jets generated with Pythia 8, one with all kinematically realizable quark jets and one that excludes charm and bottom quark jets (at the level of the hard process). The one without c and b jets was originally used in Energy Flow Networks: Deep Sets for Particle Jets. Generation parameters are listed below:

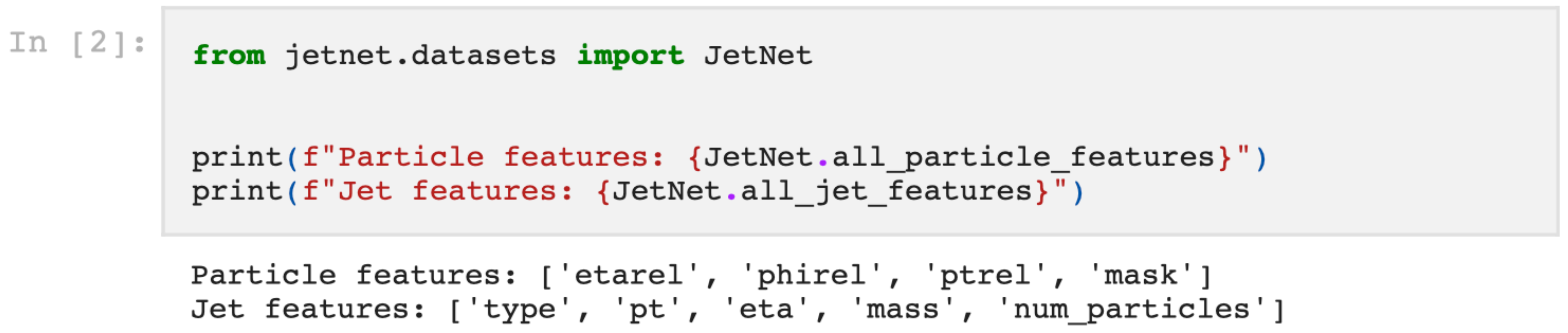


JetClass: A Large-Scale Dataset for Deep Learning in Jet Physics



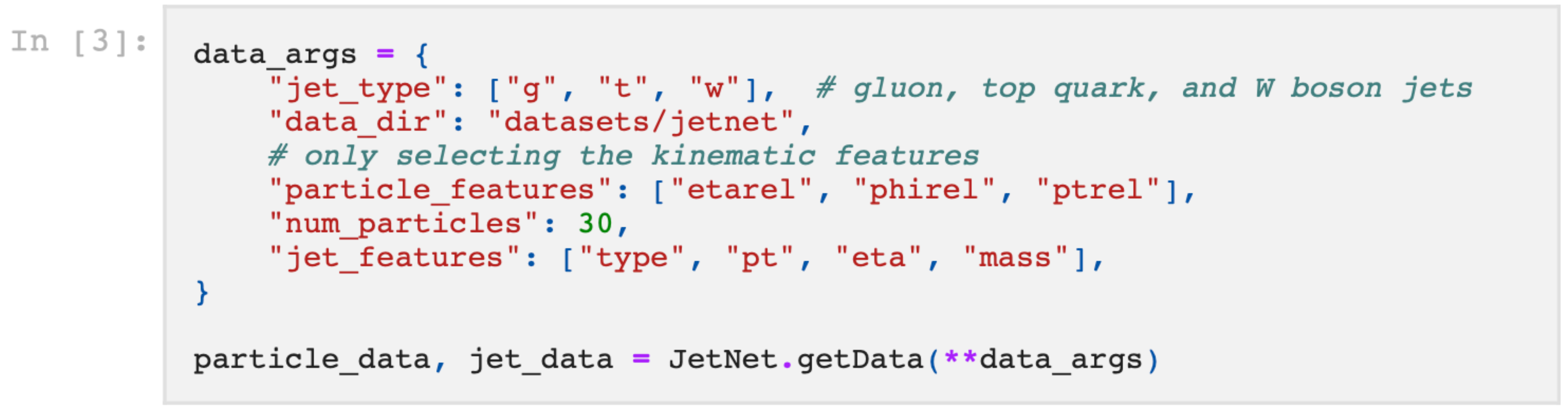
```
particle_data, jet_data = JetNet.getData(**data_args)
particle_data, jet_data = QuarkGluon.getData(**data_args)
particle_data, jet_data = TopTagging.getData(**data_arg)
```

Let’s take a look at the JetNet Dataset



```
In [2]:  from jetnet.datasets import JetNet


         print(f"Particle features: {JetNet.all_particle_features}")
         print(f"Jet features: {JetNet.all_jet_features}")

         Particle features: ['etarel', 'phirel', 'ptrel', 'mask']
         Jet features: ['type', 'pt', 'eta', 'mass', 'num_particles']
```

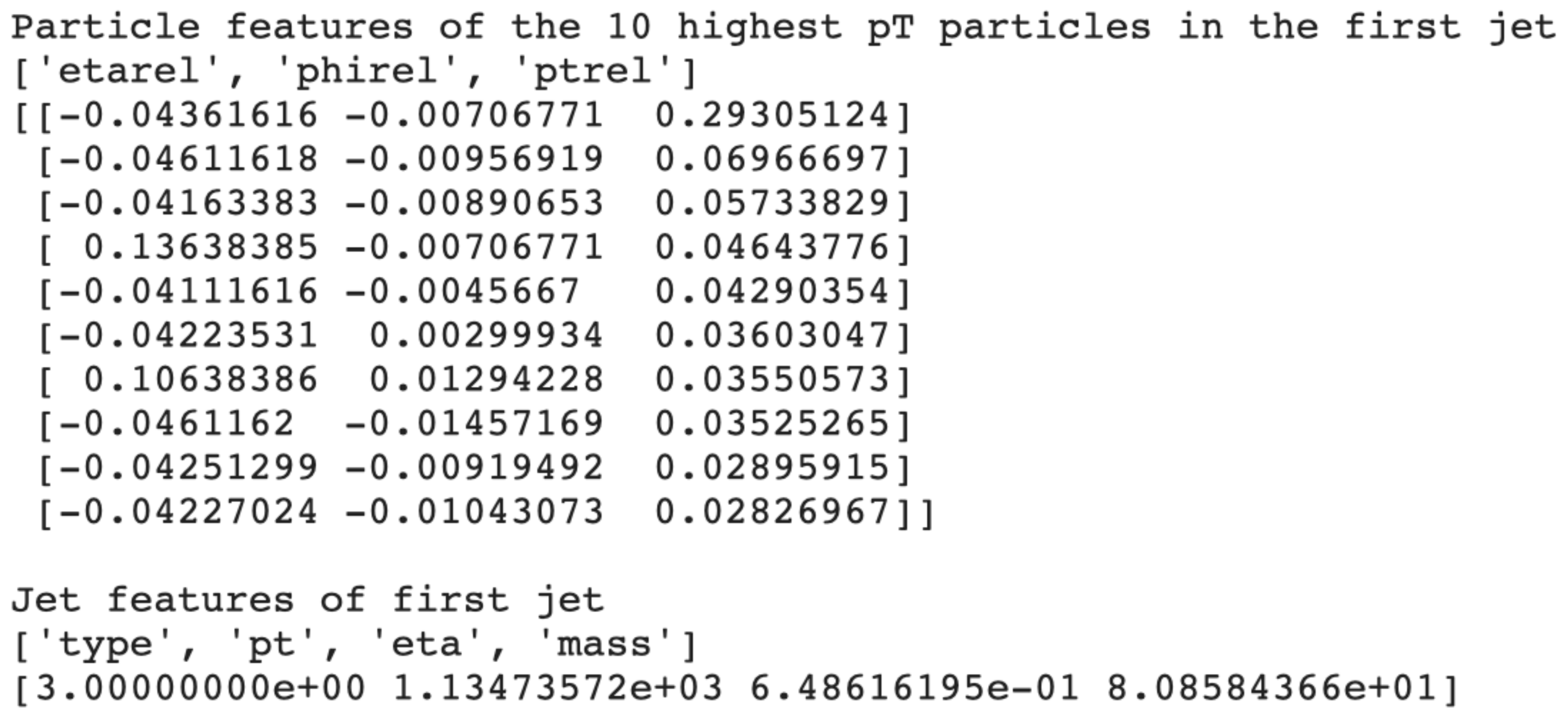
- First, we take a look at the particle and jet features that are offered in the JetNet dataset



```
In [3]:  data_args = {
             "jet_type": ["g", "t", "w"],  # gluon, top quark, and W boson jets
             "data_dir": "datasets/jetnet",
             # only selecting the kinematic features
             "particle_features": ["etarel", "phirel", "ptrel"],
             "num_particles": 30,
             "jet_features": ["type", "pt", "eta", "mass"],
         }

         particle_data, jet_data = JetNet.getData(**data_args)
```

- Second, we load the dataset with some arguments such as jet type, data directory, etc.

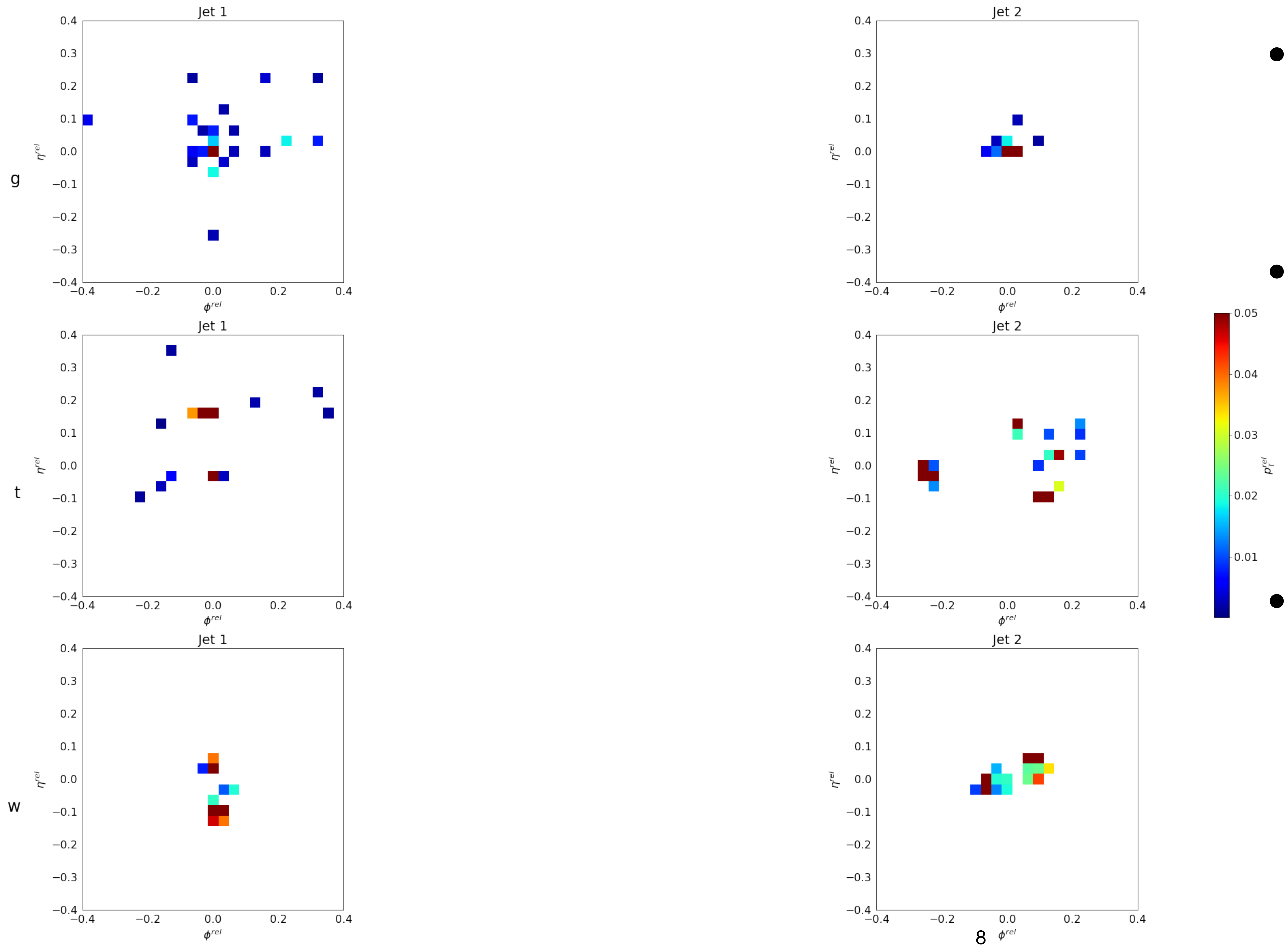
JetNet Dataset



```
Particle features of the 10 highest pT particles in the first jet
['etarel', 'phirel', 'ptrel']
[[-0.04361616 -0.00706771  0.29305124]
 [-0.04611618 -0.00956919  0.06966697]
 [-0.04163383 -0.00890653  0.05733829]
 [ 0.13638385 -0.00706771  0.04643776]
 [-0.04111616 -0.0045667   0.04290354]
 [-0.04223531  0.00299934  0.03603047]
 [ 0.10638386  0.01294228  0.03550573]
 [-0.0461162  -0.01457169  0.03525265]
 [-0.04251299 -0.00919492  0.02895915]
 [-0.04227024 -0.01043073  0.02826967]]

Jet features of first jet
['type', 'pt', 'eta', 'mass']
[3.00000000e+00 1.13473572e+03 6.48616195e-01 8.08584366e+01]
```

- First, we take a look at the particle features of the 10 highest momentum particles in the first jet.
- Second, we take a look at the jet features of the first jet

JetNet Utility Functions, to_image



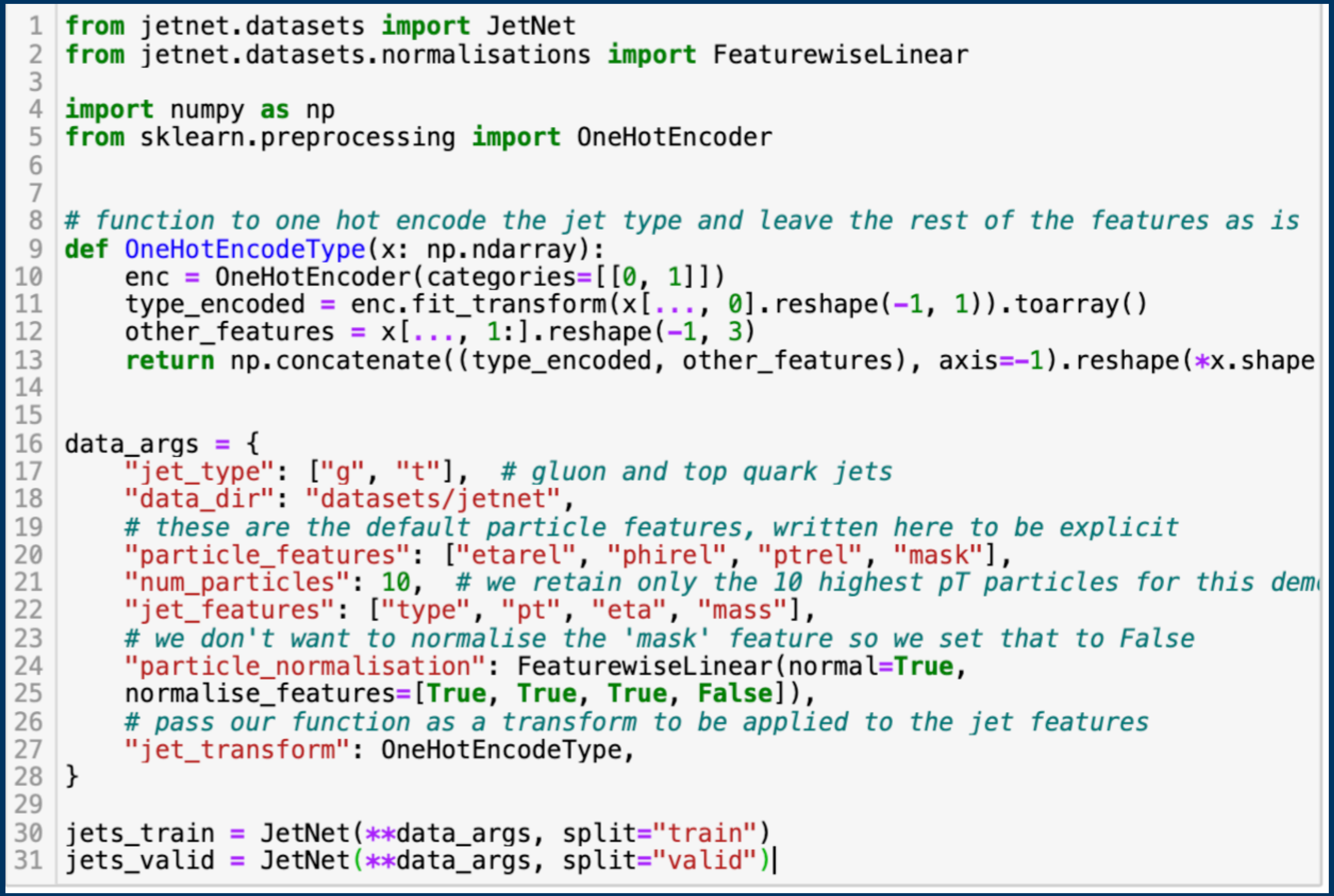
- Allows the user to convert jet data from one of the datasets into Jet Images
- Assists with data visualization to identify patterns, trends, and possible outliers with an eye test
- Many other built-in utility functions!

Dataset preparation

- To prepare the dataset for machine learning applications, we can use the *jetnet.datasets.JetNet* class itself, which inherits the pytorch.data.utils.Dataset class

- We'll also use the class to **normalize** the features to have zero means and unit standard deviations, and **transform** the jet type feature to be one-hot-encoded

- Data Splits
- We can now feed this into a **PyTorch DataLoader** and start training!

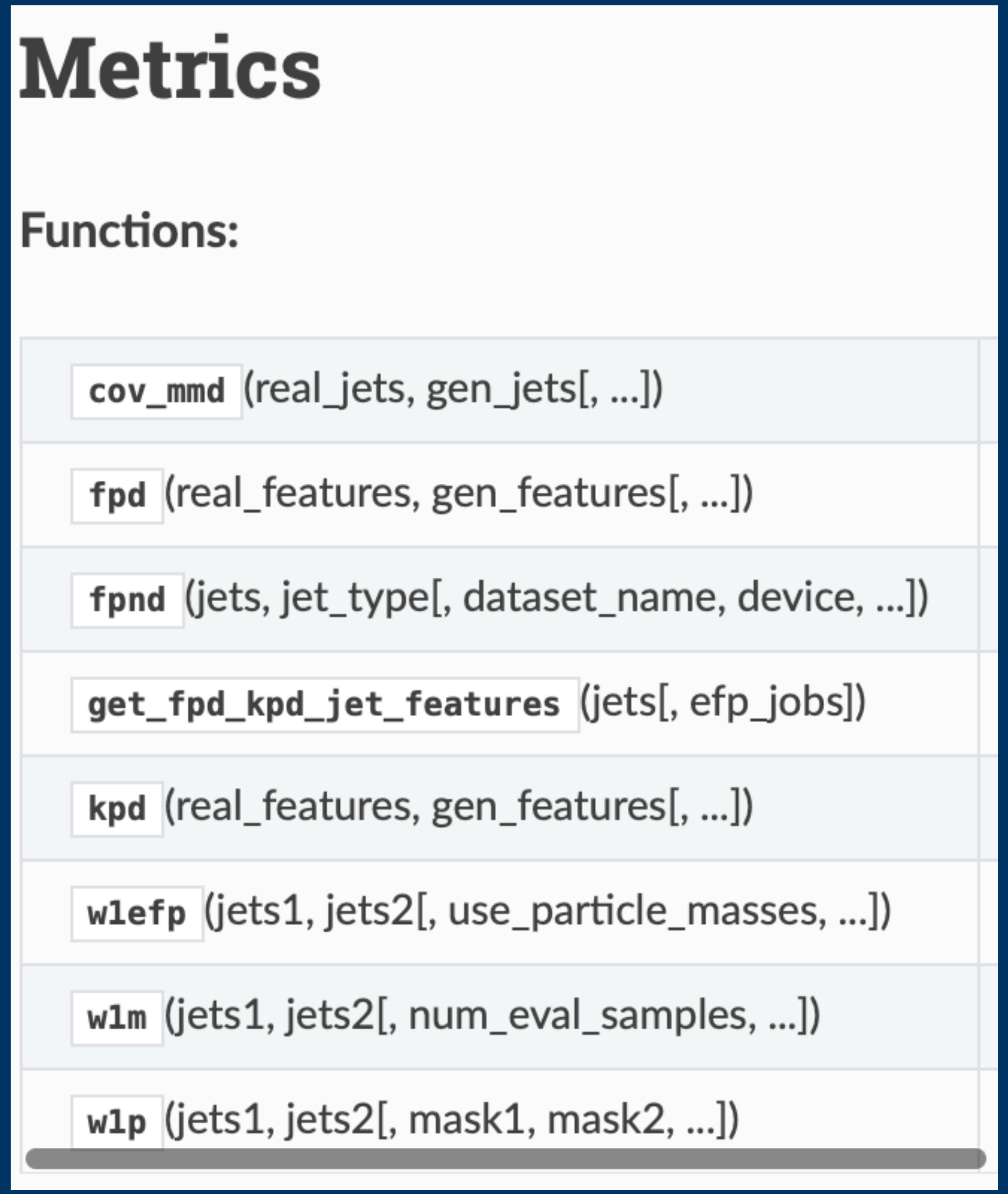


```python
from jetnet.datasets import JetNet
from jetnet.datasets.normalisations import FeaturewiseLinear

import numpy as np
from sklearn.preprocessing import OneHotEncoder


# function to one hot encode the jet type and leave the rest of the features as is
def OneHotEncodeType(x: np.ndarray):
    enc = OneHotEncoder(categories=[[0, 1]])
    type_encoded = enc.fit_transform(x[..., 0].reshape(-1, 1)).toarray()
    other_features = x[..., 1:].reshape(-1, 3)
    return np.concatenate((type_encoded, other_features), axis=-1).reshape(*x.shape


data_args = {
    "jet_type": ["g", "t"],  # gluon and top quark jets
    "data_dir": "datasets/jetnet",
    # these are the default particle features, written here to be explicit
    "particle_features": ["etarel", "phirel", "ptrel", "mask"],
    "num_particles": 10,  # we retain only the 10 highest pT particles for this dem
    "jet_features": ["type", "pt", "eta", "mass"],
    # we don't want to normalise the 'mask' feature so we set that to False
    "particle_normalisation": FeaturewiseLinear(normal=True,
    normalise_features=[True, True, True, False]),
    # pass our function as a transform to be applied to the jet features
    "jet_transform": OneHotEncodeType,
}

jets_train = JetNet(**data_args, split="train")
jets_valid = JetNet(**data_args, split="valid")
```

Evaluation Metrics

- We want objective, standardized metrics to evaluate ML results
- This is difficult in simulations
- JetNet includes metrics to help alleviate this difficulty

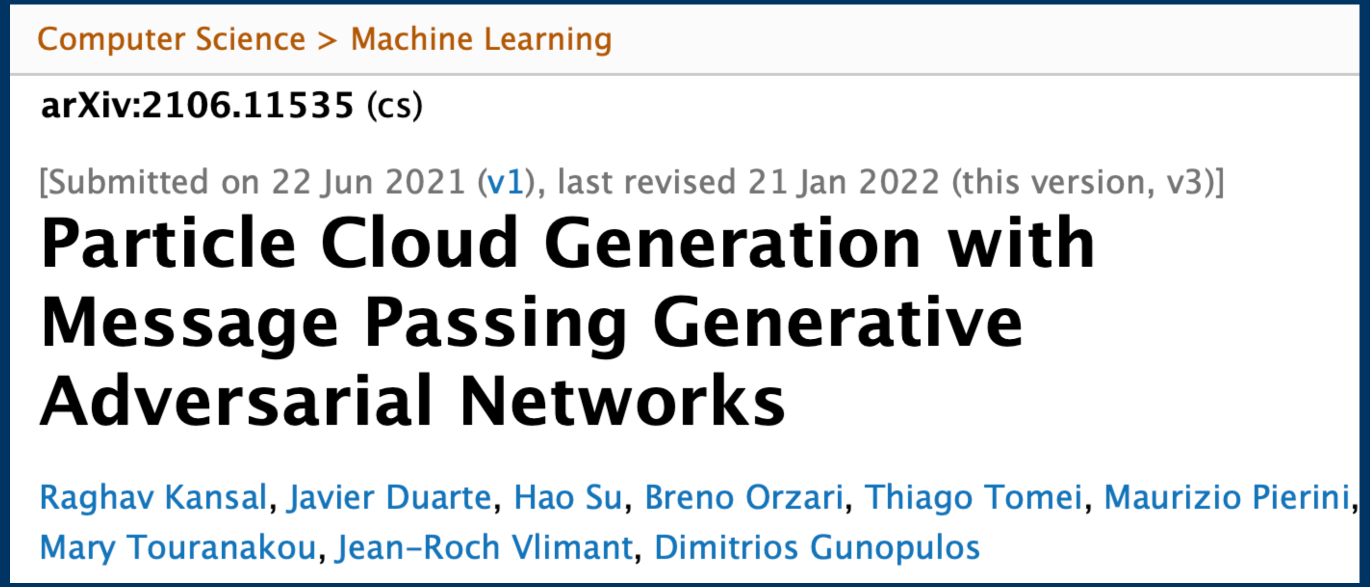
```
fpd = jetnet.evaluation.fpd(real_jets, gen_jets)
```



Metrics

Functions:

`cov_mmd`	(real_jets, gen_jets[, ...])
`fpd`	(real_features, gen_features[, ...])
`fpnd`	(jets, jet_type[, dataset_name, device, ...])
`get_fpd_kpd_jet_features`	(jets[, efp_jobs])
`kpd`	(real_features, gen_features[, ...])
`w1efp`	(jets1, jets2[, use_particle_masses, ...])
`w1m`	(jets1, jets2[, num_eval_samples, ...])
`w1p`	(jets1, jets2[, mask1, mask2, ...])



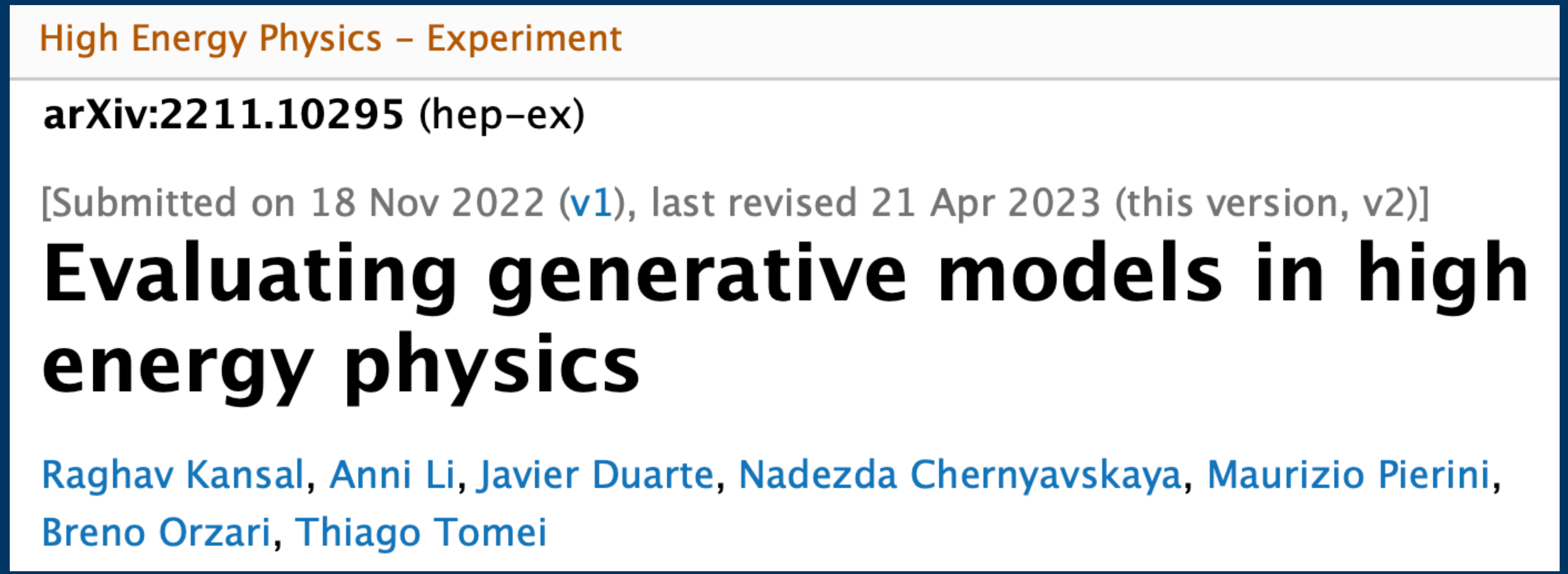
Computer Science > Machine Learning

arXiv:2106.11535 (cs)

[Submitted on 22 Jun 2021 (v1), last revised 21 Jan 2022 (this version, v3)]

Particle Cloud Generation with Message Passing Generative Adversarial Networks

Raghav Kansal, Javier Duarte, Hao Su, Breno Orzari, Thiago Tomei, Maurizio Pierini, Mary Touranakou, Jean-Roch Vlimant, Dimitrios Gunopulos



High Energy Physics – Experiment

arXiv:2211.10295 (hep-ex)

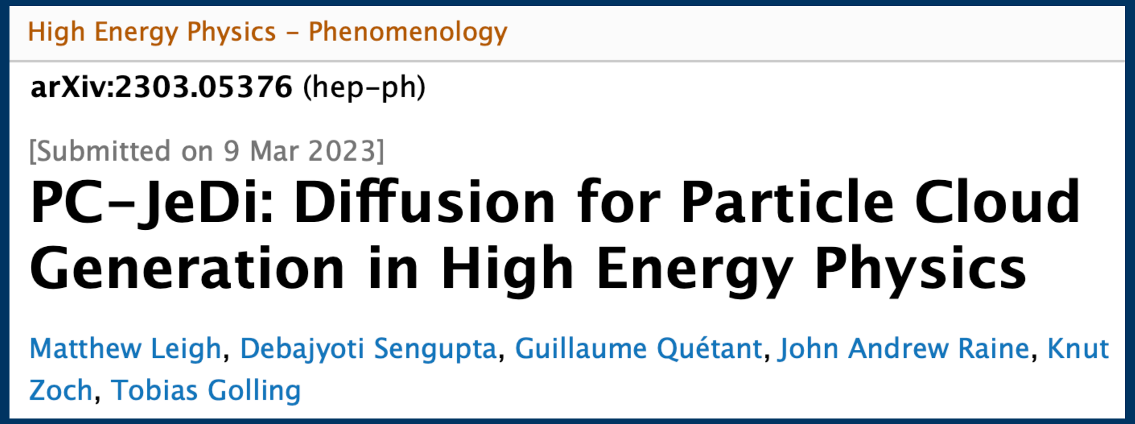
[Submitted on 18 Nov 2022 (v1), last revised 21 Apr 2023 (this version, v2)]

Evaluating generative models in high energy physics

Raghav Kansal, Anni Li, Javier Duarte, Nadezda Chernyavskaya, Maurizio Pierini, Breno Orzari, Thiago Tomei

Impact of JetNet

- **30k downloads**
- **JetNet has enabled a lot of exciting research in ML for jets!**

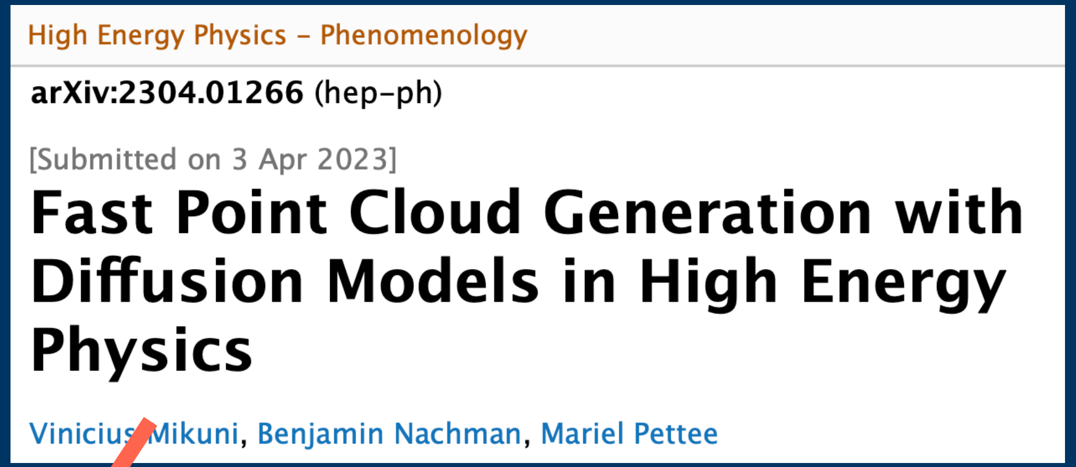

High Energy Physics – Phenomenology

arXiv:2303.05376 (hep-ph)

[Submitted on 9 Mar 2023]

PC-JeDi: Diffusion for Particle Cloud Generation in High Energy Physics

Matthew Leigh, Debajyoti Sengupta, Guillaume Quétant, John Andrew Raine, Knut Zoch, Tobias Golling


High Energy Physics – Phenomenology

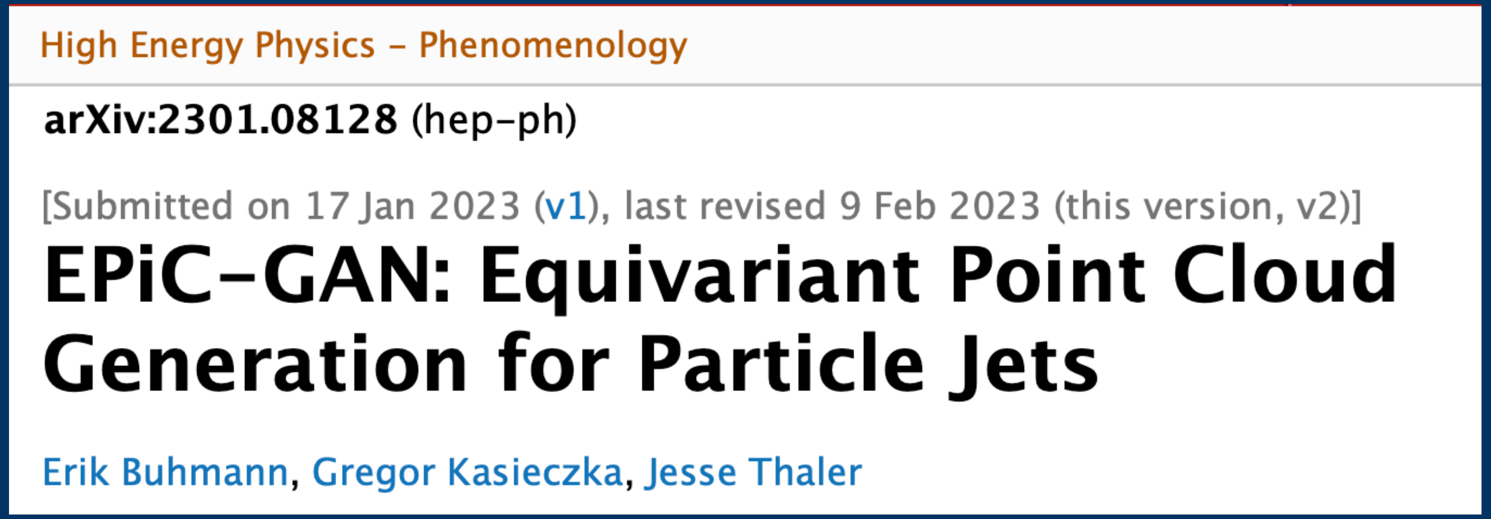
arXiv:2304.01266 (hep-ph)

[Submitted on 3 Apr 2023]

Fast Point Cloud Generation with Diffusion Models in High Energy Physics

Vinicius Mikuni, Benjamin Nachman, Mariel Pettee

Paper: 2303.05376 , 2304.01266
Use Case: accessed jets via JetNet tor diffusion models, and metrics via JetNet to evaluate the jets


High Energy Physics – Phenomenology

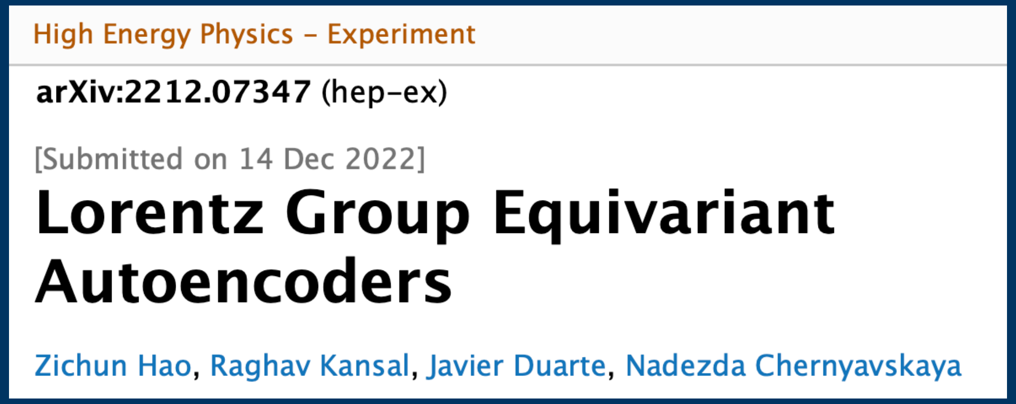
arXiv:2301.08128 (hep-ph)

[Submitted on 17 Jan 2023 (v1), last revised 9 Feb 2023 (this version, v2)]

EPiC-GAN: Equivariant Point Cloud Generation for Particle Jets

Erik Buhmann, Gregor Kasieczka, Jesse Thaler

Paper: 2301.08128
Use Case: accessed jets via JetNet tor training a GAN, and used JetNet Evaluation metrics to evaluate the jets


High Energy Physics – Experiment

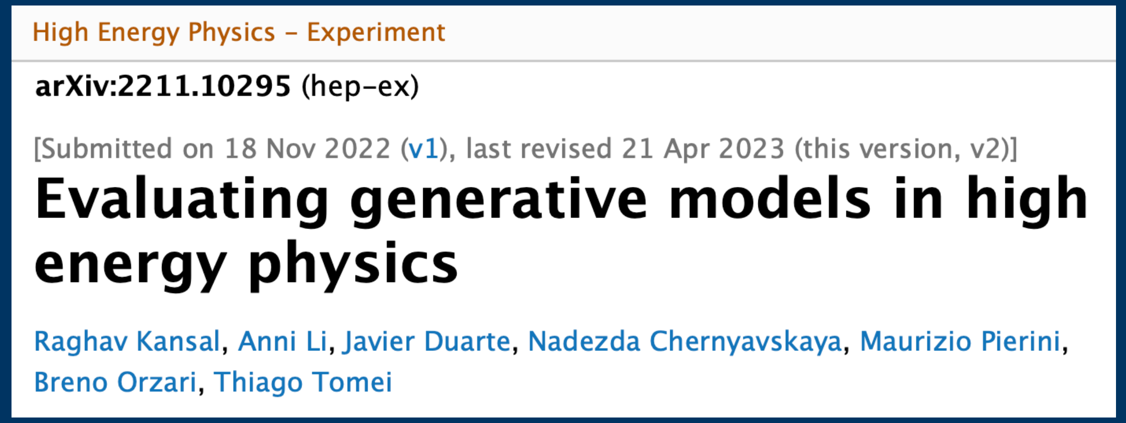
arXiv:2212.07347 (hep-ex)

[Submitted on 14 Dec 2022]

Lorentz Group Equivariant Autoencoders

Zichun Hao, Raghav Kansal, Javier Duarte, Nadezda Chernyavskaya

Paper: 2212.07347
Use Case: accessed jets via JetNet to train a Lorentz-equivariant auto encoder for compression and anomaly detection


High Energy Physics – Experiment

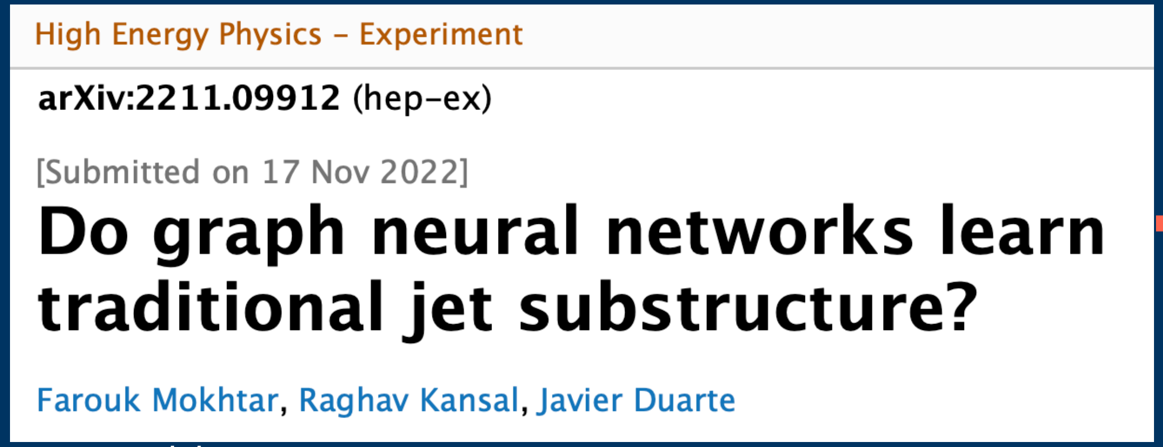
arXiv:2211.10295 (hep-ex)

[Submitted on 18 Nov 2022 (v1), last revised 21 Apr 2023 (this version, v2)]

Evaluating generative models in high energy physics

Raghav Kansal, Anni Li, Javier Duarte, Nadezda Chernyavskaya, Maurizio Pierini, Breno Orzari, Thiago Tomei

Paper: 2211.10295
Use Case: accessed jets via JetNet and distorts them to test evaluation metrics


High Energy Physics – Experiment

arXiv:2211.09912 (hep-ex)

[Submitted on 17 Nov 2022]

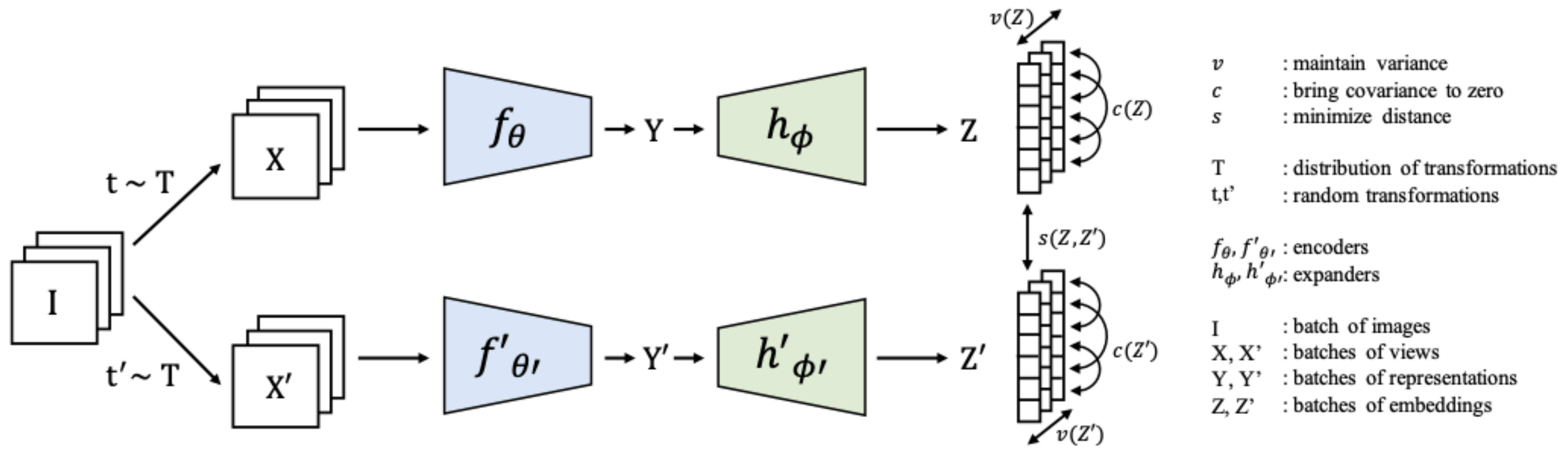
Do graph neural networks learn traditional jet substructure?

Farouk Mokhtar, Raghav Kansal, Javier Duarte

Paper: 2211.09912
Use Case: Uses TopTagging dataset and interpret's the ParticleNet model's classification score

Another example of using JetNet

Self-Supervised Learning (VICReg) for Jet Classification



$t \sim T$
$t' \sim T$
I
X
X'
f_θ
$f'_{\theta'}$
Y
Y'
h_ϕ
$h'_{\phi'}$
Z
Z'
$v(Z)$
$c(Z)$
$s(Z,Z')$
$c(Z')$
$v(Z')$
v : maintain variance
c : bring covariance to zero
s : minimize distance
T : distribution of transformations
t,t' : random transformations
$f_\theta, f'_{\theta'}$: encoders
$h_\phi, h'_{\phi'}$: expanders
I : batch of images
X, X' : batches of views
Y, Y' : batches of representations
Z, Z' : batches of embeddings

- **Perform data augmentations to produce different views of the same image —> jet**
- **VICReg loss consists of the Variance, Invariance, and Covariance terms all used to prevent an informational collapse that tends to be common in self-supervised learning approaches**

- **Classical Data Augmentations:**
  - Random Jitter, Random Resized Crop, Gaussian Blur, etc.
- **Physics-Inspired Data Augmentations:**
  - Rotations, boosts, collinear splitting, etc.

Summary: Access jets via JetNet, apply data augmentations in order to train a network to be invariant to these augmentations and classify the jets correctly.

Future of JetNet

- Expand to more datasets!
- Expand to datasets with calorimeter showers
- Integrate Lazy Loading
- Integrate Physics-inspired augmentations for Self-Supervised Learning

- Any feedback is greatly appreciated and welcomed!

Conclusion

- Today, we saw a small glimpse of the JetNet library:
    - **Datasets** —> Preparing dataset in 2-3 lines of code
    - **Evaluation Metrics**
    - **Utility functions**
    - **Impact of JetNet**
    - **Where we are heading**

- **By allowing researchers to access datasets easily and focus on the ML, JetNet is enabling state-of-the-art research in areas like equivariant neural networks, explainable AI, diffusion, and self-supervised learning!**

Thank you!

- Thank you to all of the organizers at CHEP, Javier Duarte, Raghav Kansal, and the EXPAND program at UC San Diego! Thank you to the DOE Office of Advanced Scientific Computing Research, FAIR4HEP, and the NSF, A3D3 program for their continued support!