HEP ML Lab

An end-to-end framework for machine learning application in high energy physics

Jing Li, Hao Sun
Dalian University of Technology

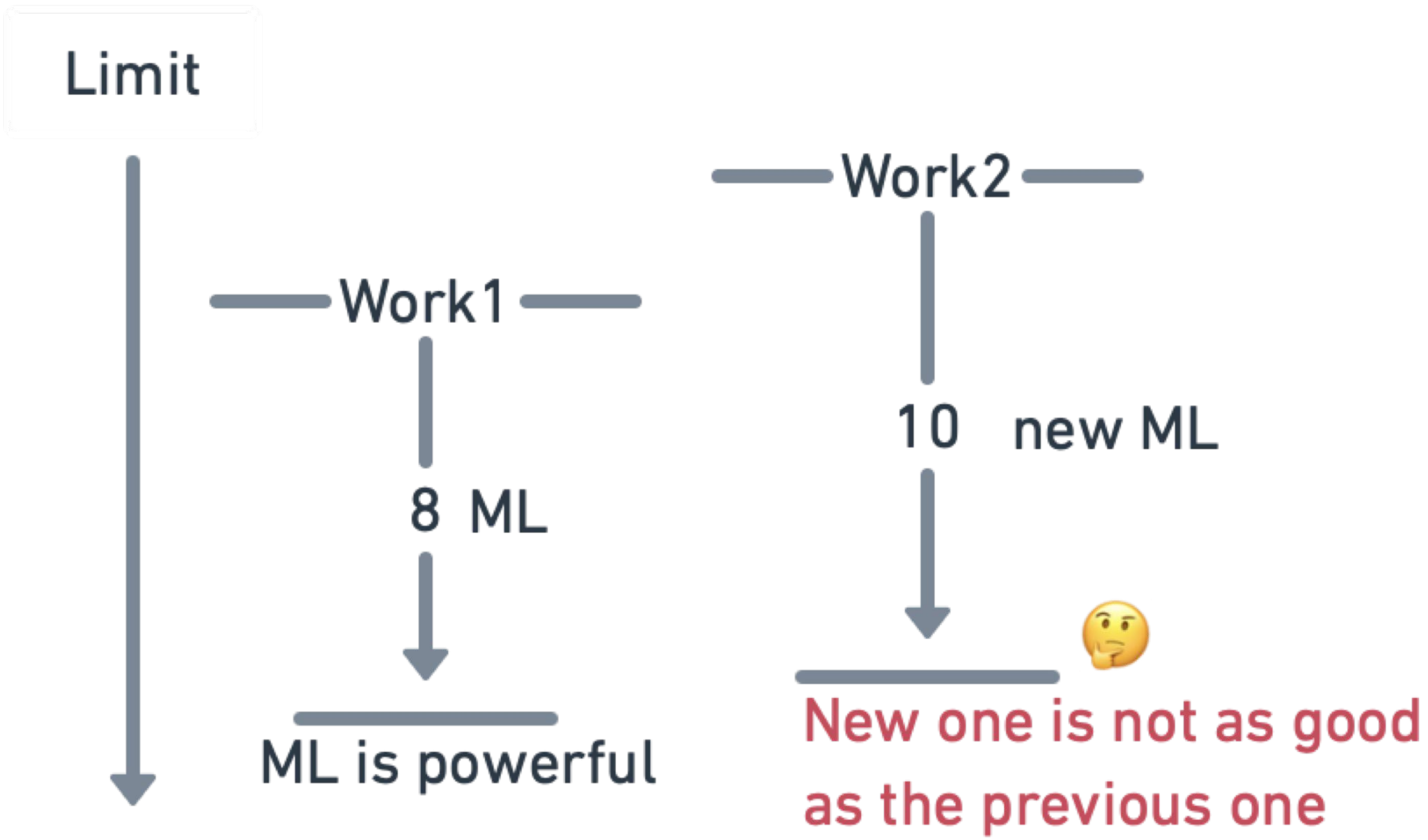
28th Mini-workshop on the frontier of LHC

Contents

Introduction

Reproduction Issues

- A large amount of work explores the performance improvement brought by machine learning methods. Results are **promising** for the new physics search.
- However the lack of source codes makes it **quite difficult to reproduce** the results.
- If they are generated under different conditions, at what extent we can say that new methods are truly powerful and worth to try in broader subjects?



Introduction

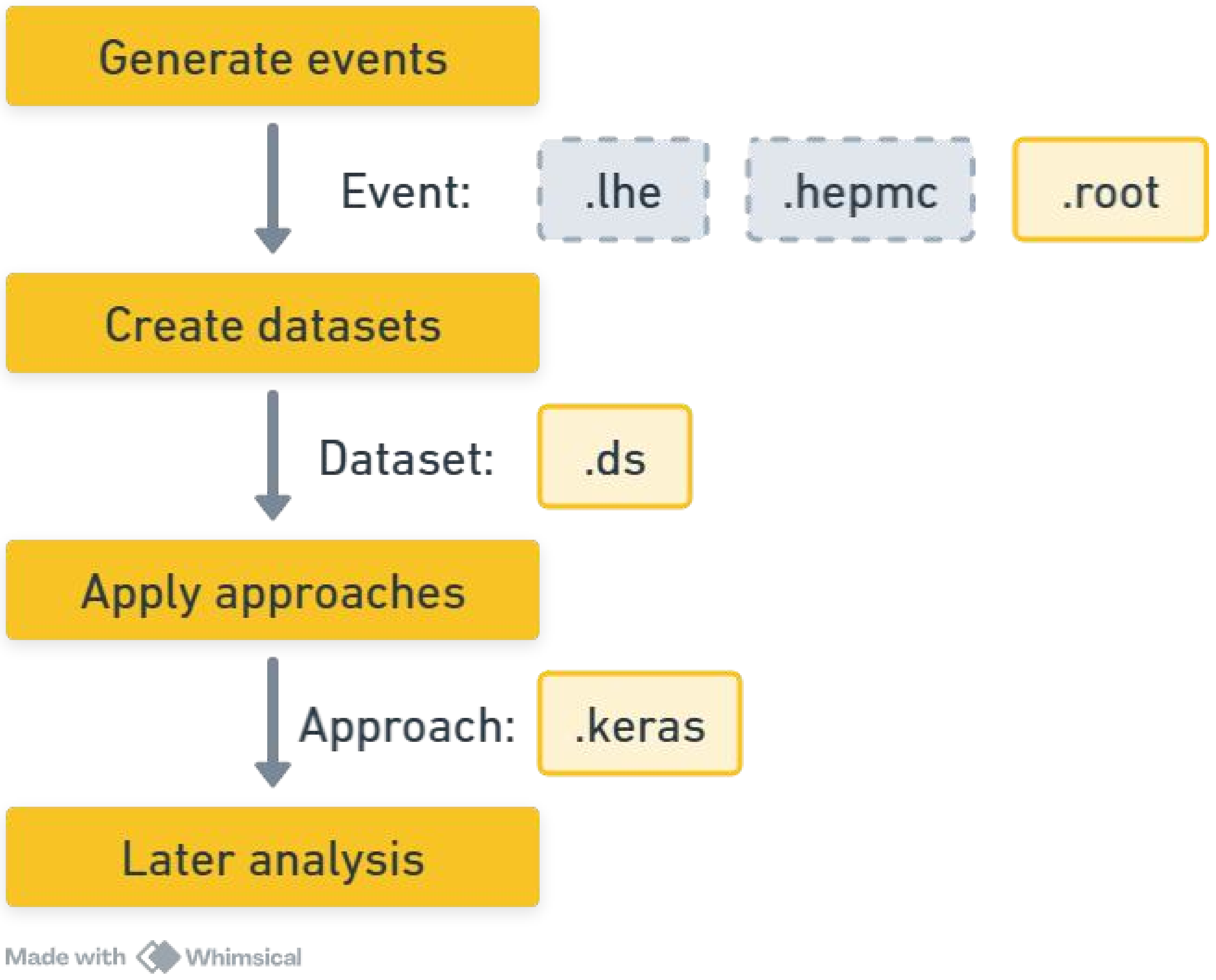
Existing framrworks

Name	Data	Model	Style	Highlight
hep_ml	No	Yes	sklearn	Low correlation, theano-based, sklearn compatible
weaver	Yes	Yes	CLI + config	Support many dataset formats, config for all
JetNet	Yes	No	custom	Three datasets, generative evaluation metrics
pd4ml	Yes	Yes	modified Keras	Five datasets, model templates
MLAnalysis	Yes	Yes	custom	LHE/LHCO data, three ML algorithms
mapyde	Yes	No	CLI + TUI + config	Madgraph5 workflow, for a specific problem
madminer	Yes	Yes	custom	Madgraph5 workflow, for a specific problem
hep-ml-lab	Yes	Yes	Keras	Madgraph5 workflow, not specific for one problem

Introduction

Overall workflow

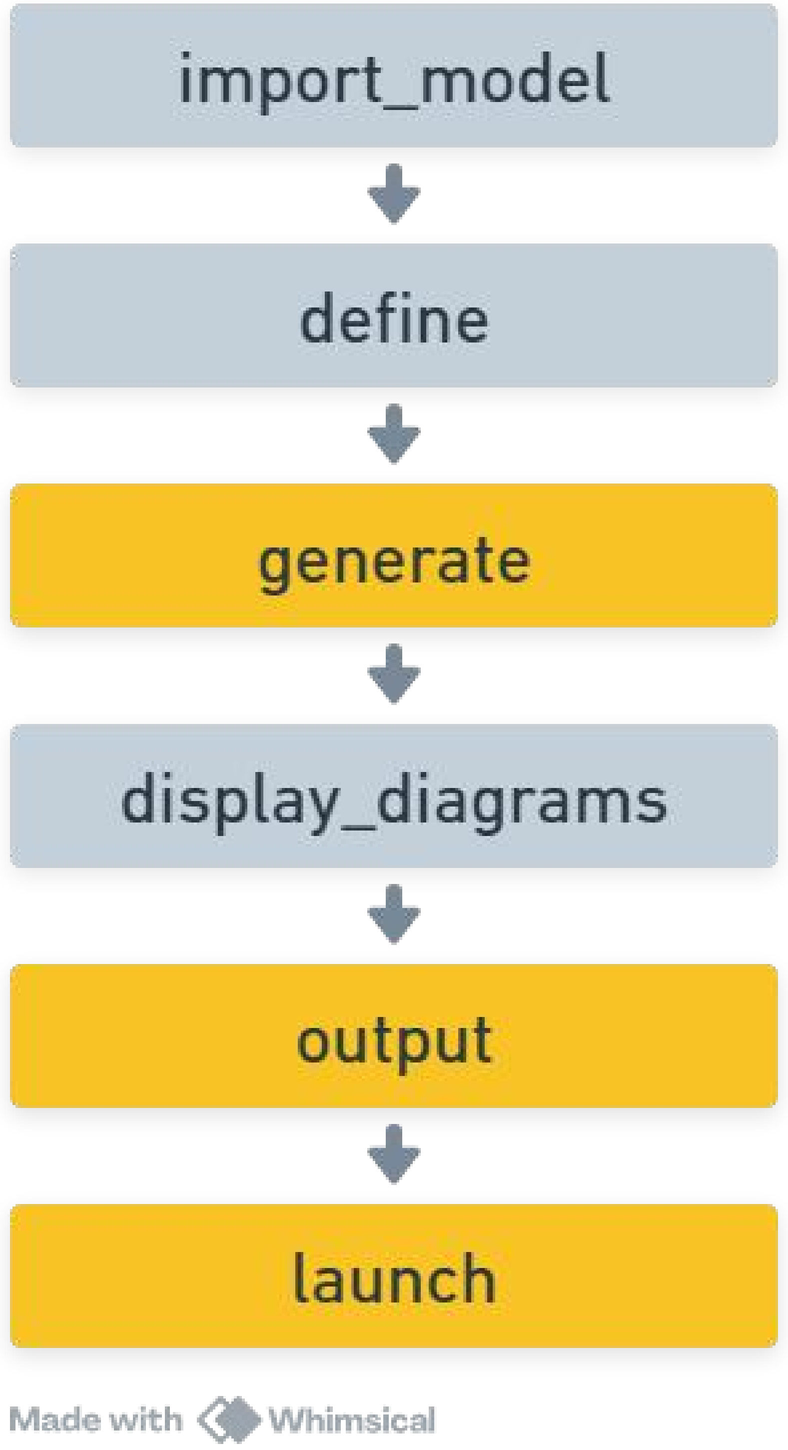
- An **end-to-end** framework for applying machine learning into HEP studies.
- **Simplify the data flow**: easier to manage and track different data.
- **Unify programming style** across all stages: objected-oriented and Keras.
- All makes the results more reliable and reproducible.



APIs

Generate events

- Minimal wrapper of Madgraph5




```python
g = Madgraph5(executable="mg5_aMC", verbose=1)
g.generate(["p p > w+ z"])
g.output("data/pp2wz")
g.launch(
    shower="pythia8",
    detector="delphes",
    madspin="none",
    settings={
        "nevents": 1000,
        "run_tag": "250-300",
        "pt_min_pdg": {24: 250},
        "pt_max_pdg": {24: 300},
    },
    decays=[
        "w+ > j j",
        "z > vl vl~",
    ],
    seed=42,
)
```

APIs

Generate events

- Flexible control of intermediate output.
  - verbose=0: quiet
  - verbose=1: suppressed output as shown
  - verbose=2: original output

- Check the results via **summary**

```
Madgraph5_aMC@NLO v3.4.2
Total: 2 processes with 6 diagrams
1 diagrams saved in ./data/pp2wz/Diagrams
Output saved in ./data/pp2wz
run_01...survey...pythia8...delphes...storing...✓
```

p p > w+ z

#	Name	Collider	Tag	Cross section (pb)	N events	Seed
0	run_01[1]	pp:6500.0x6500.0	250-300	4.371e-02 +- 4.200e-04	1,000	42

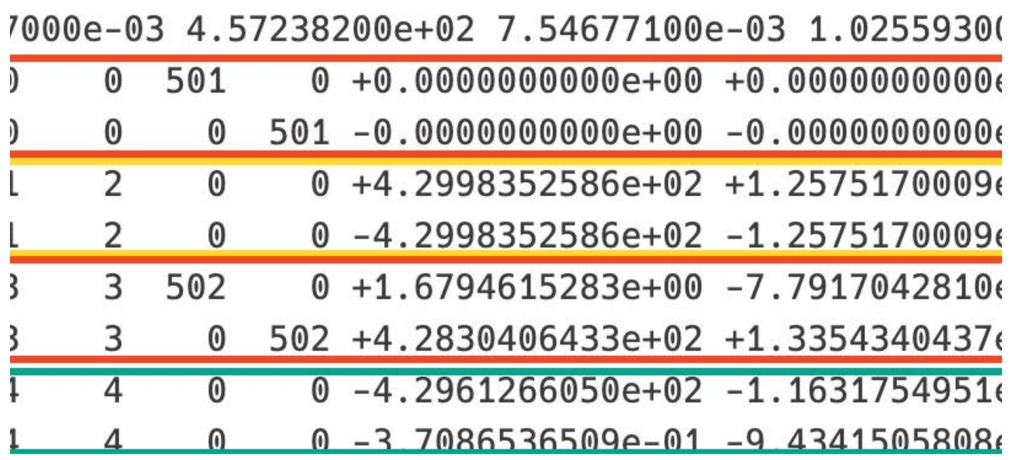
Output: data/pp2wz

APIs

Events

Parton-level

.lhe

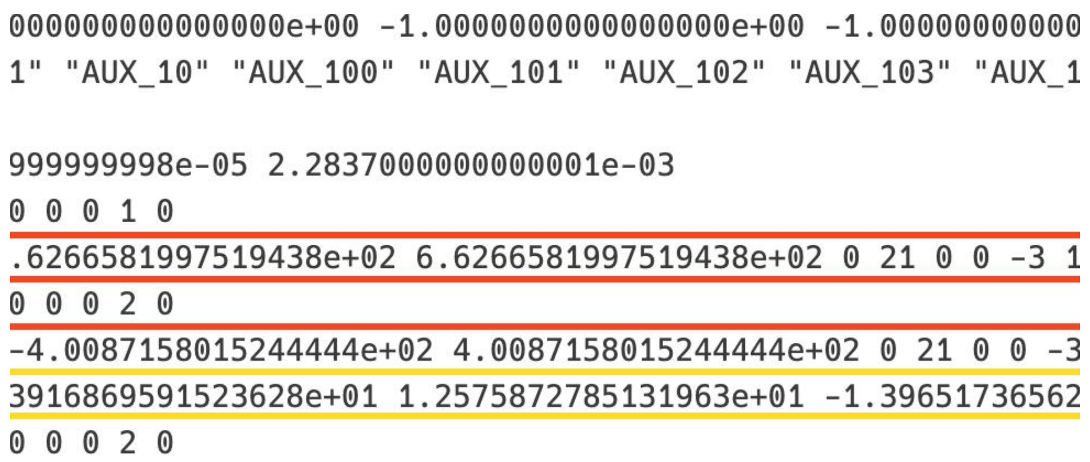


```
7000e-03 4.57238200e+02 7.54677100e-03 1.0255930
   0  501     0 +0.0000000000e+00 +0.0000000000
   0    0   501 -0.0000000000e+00 -0.0000000000
   2    0     0 +4.2998352586e+02 +1.2575170009
   2    0     0 -4.2998352586e+02 -1.2575170009
   3  502     0 +1.6794615283e+00 -7.7917042810
   3    0   502 +4.2830406433e+02 +1.3354340437
   4    0     0 -4.2961266050e+02 -1.1631754951
   4    0     0 -3.7086536509e-01 -9.4341505808
```

- particle.pid
- particle.eta
- particle.phi
- particle.mass

Particle-level

.hepmc

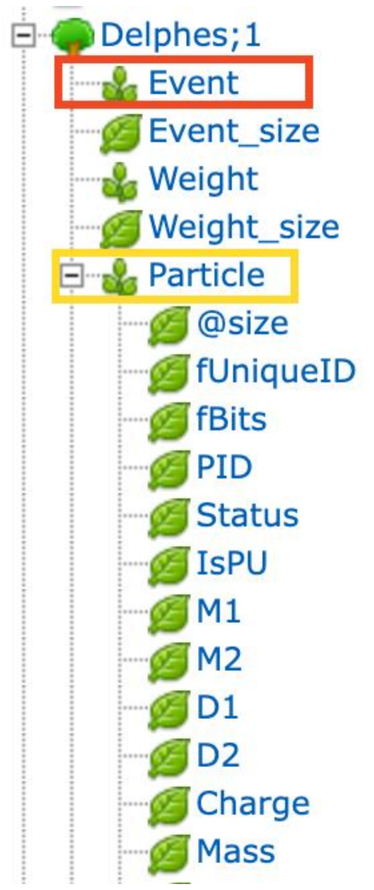


```
000000000000000e+00 -1.0000000000000000e+00 -1.00000000000
1" "AUX_10" "AUX_100" "AUX_101" "AUX_102" "AUX_103" "AUX_1

999999998e-05 2.2837000000000001e-03
0 0 0 1 0
.6266581997519438e+02 6.6266581997519438e+02 0 21 0 0 -3 1
0 0 0 2 0
-4.0087158015244444e+02 4.0087158015244444e+02 0 21 0 0 -3
3916869591523628e+01 1.2575872785131963e+01 -1.39651736562
0 0 0 2 0
```

- particle.pid
- particle.eta
- particle.phi
- particle.mass

Detector-level

.root



- Electron.PT
- Jet.PT
- Muon.Charge
- Tower.ET
- Track.Mass
- ...

APIs

Events

```
events = ROOTEvents.load("../../tag_1_delphes_events.root")
events["jet.pt"]
```

```
[[520, 42.4],
 ...,
 [545]]
------------------------------
type: 100 * var * float32
```

- Data is recorded as an awkward array.
- It could be interpreted as 100 events with variable number of jets.

Naming convention: PhysicsObject

- Physics object name = <type/alias>[<index>]

All muons.

muon

The first 200 constituents of the leading jet.

jet0.constituents:200

The first subjet clustered by all fatjets' constituents with kt algorithm and 0.3 radius.

kt3fatjet0

Naming convention: Observable

- Observable name = <physics object>.<type/alias>

The charge of all muons's .

muon.charge

The transverse momentum of the first 200 constituents of the leading jet.

jet0.constituents:200.pt

The distance between the first and the second subjets defined as before.

kt3jet0,kt3jet1.delta_r

→ The number of physics objects is determined by the observable.

APIs

Naming convention

```
events = load_events("../../tag_1_delphes_events.root")

my_object = parse_physics_object("ak8jet.constituents")
my_object.read(events)
my_object.array
```

- Agnostic usage to ignore the event type.
- The familiar string, not programming classes.

```
parse_physics_object("ak8jet.constituents").read(events).array
```

- Method chaining makes it more compact.

⇒ **100 * var * var * {**

events number | jets number | constituents number

pt: float32,
eta: float32,
phi: float32,
mass: float32
}

- recorded observables

Naming convention

- Users can **register** aliases for physics objects.

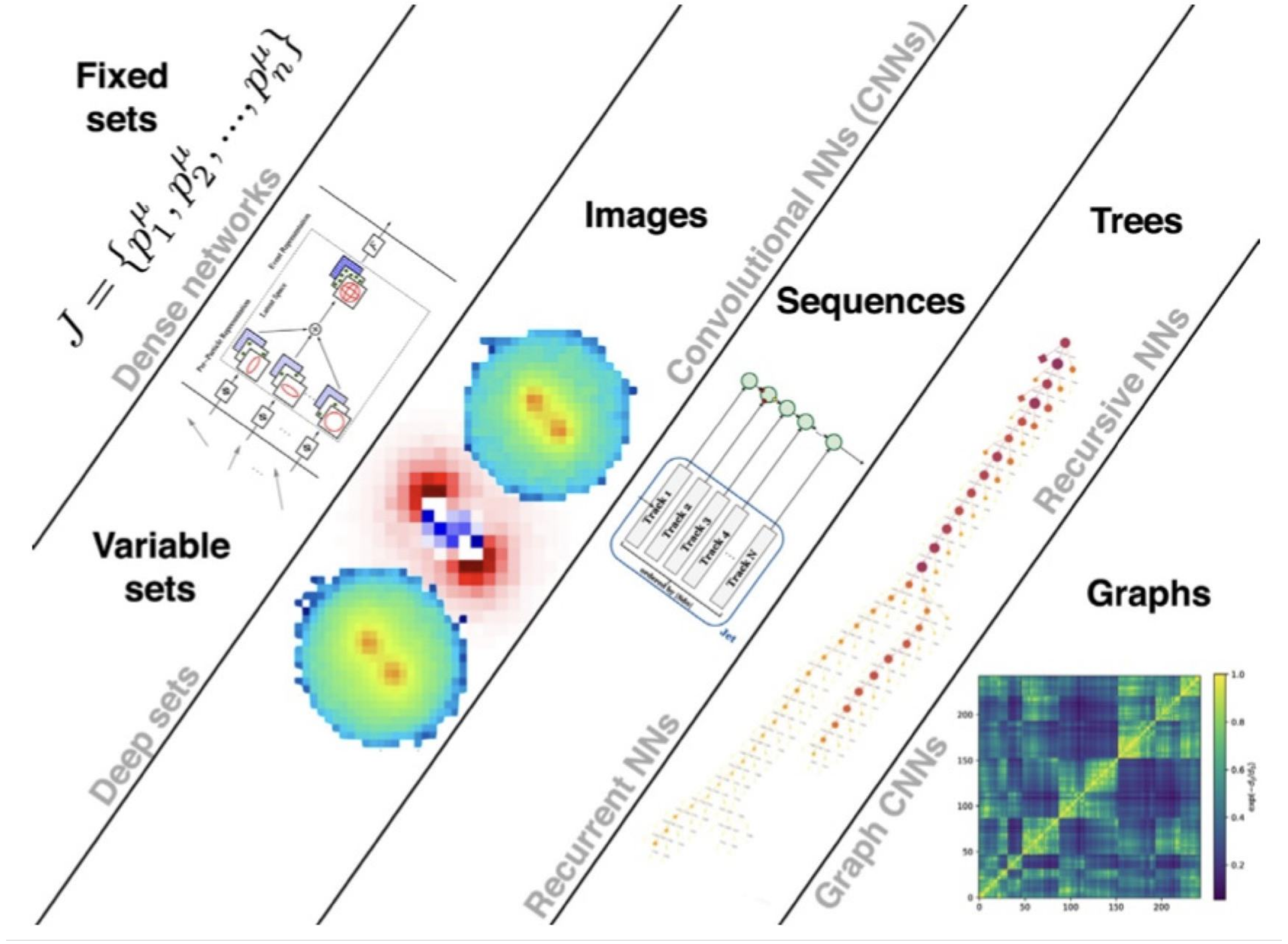
```python
subjets = parse_physics_object("ak3jet")
register(subjets, "subjets")

# In another place, get back the subjets quickly
parsed = parse_physics_object("subjets")
```

- This alias applies to observables as well.

```python
n_subjets = parse_observable("subjets.size")
n_subjets.read(events).array
```

Representations

- Set: 1D (N x F)
  Use a set of observables to represent an event or a jet.
- Image: 3D (N x H x W x C)
  Project particles onto a 2D plane.
- Graph: 2D (N x P x F)
  Record particles' features.



[1709.04464] Jet Substructure at the Large Hadron Collider:
A Review of Recent Advances in Theory and Machine Learning

APIs

Representations: Set

```python
from hml5.approaches import Cut
from hml5.events import load_events
from hml5.representations import Set


events = load_events("../../tag_1_delphes_events.root")

cut1 = Cut("fatjet.size > 0").read(events)
cut2 = Cut("ak4fatjet.size > 1").read(events)

sets = Set("fatjet0.mass", "fatjet0.tau21", "ak4fatjet0,ak4fatjet1.delta_r")
sets.read(events, cut1.array & cut2.array)
```

- Names are valid through very part of the workflow.
- Use **read** to extract data from events.

Representations: Image

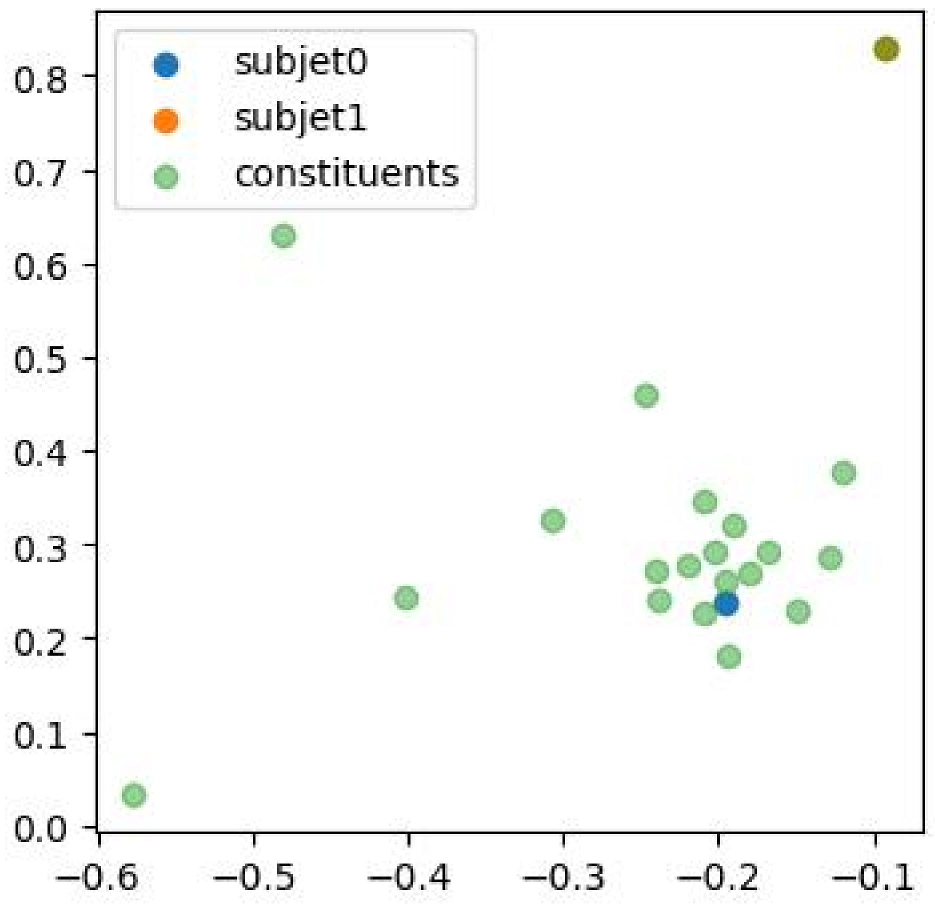
```python
images = (
    Image(
        "fatjet0.constituents.eta",
        "fatjet0.constituents.phi",
        # "fatjet0.constituents.pt",
    )
    .translate("ak4fatjet0.reclustered0")
    .rotate("ak4fatjet0.reclustered1", -90)
    .pixelate((32, 32), ((-1.5, 1.5), (-1.5, 1.5)))
    .read(events, cut0.array & cut1.array)
)
```

- Recluster the first fatjet and get the first subjet.
- Define how images are pre-processed.

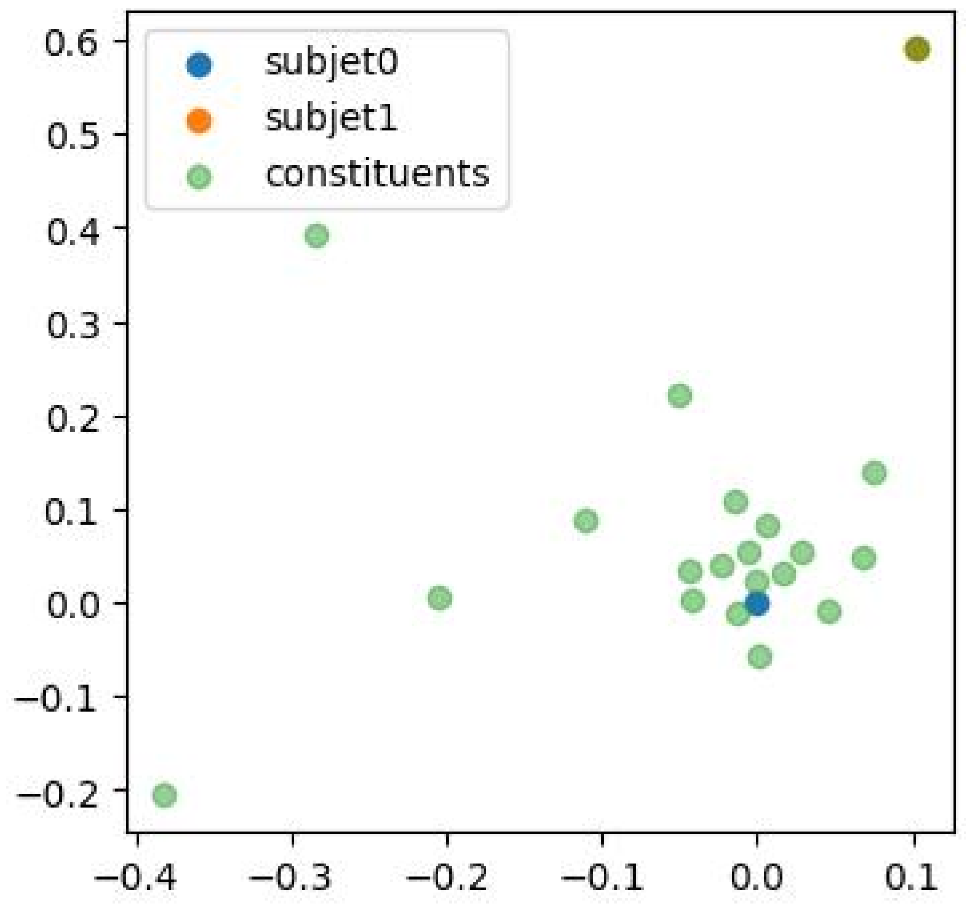
APIs

Representations: Image

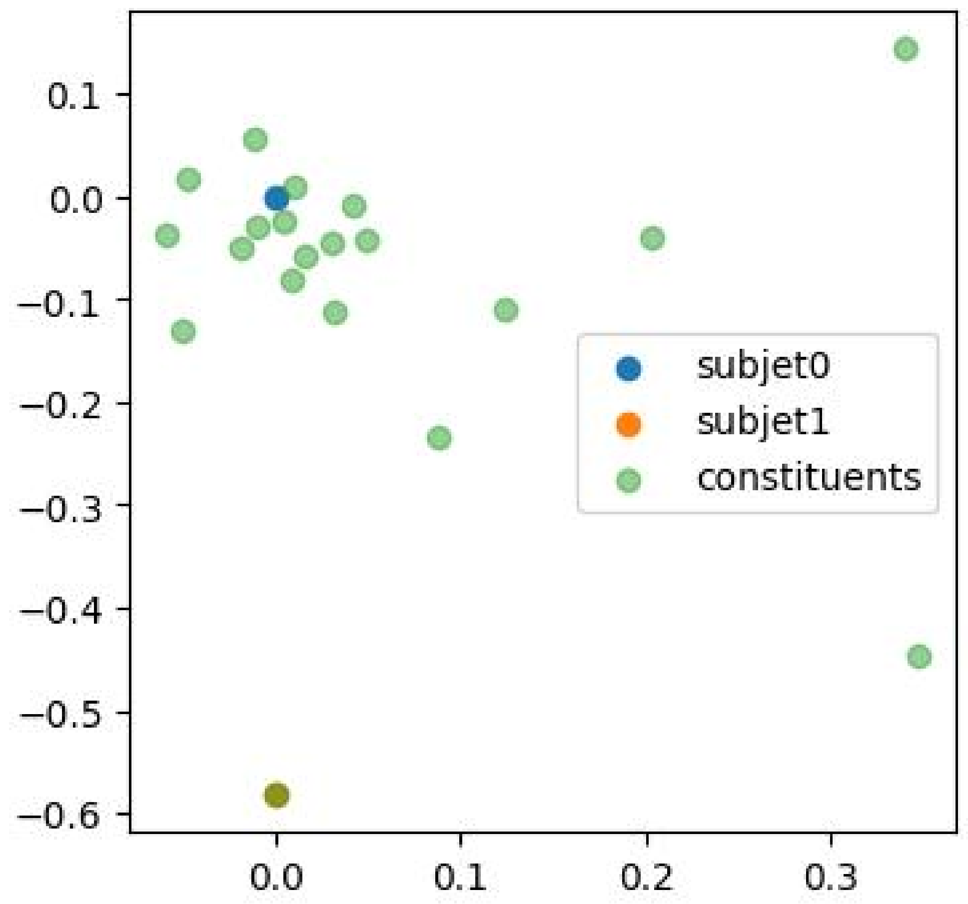
Raw



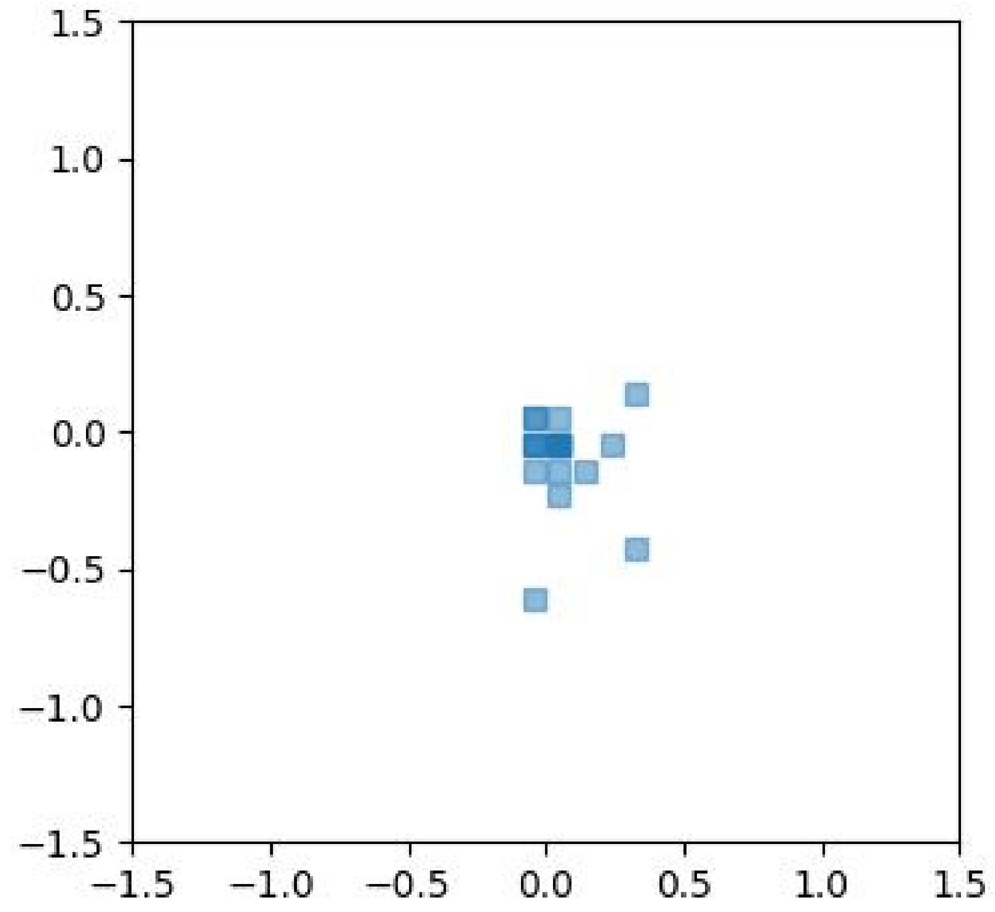
Translated



Rotated



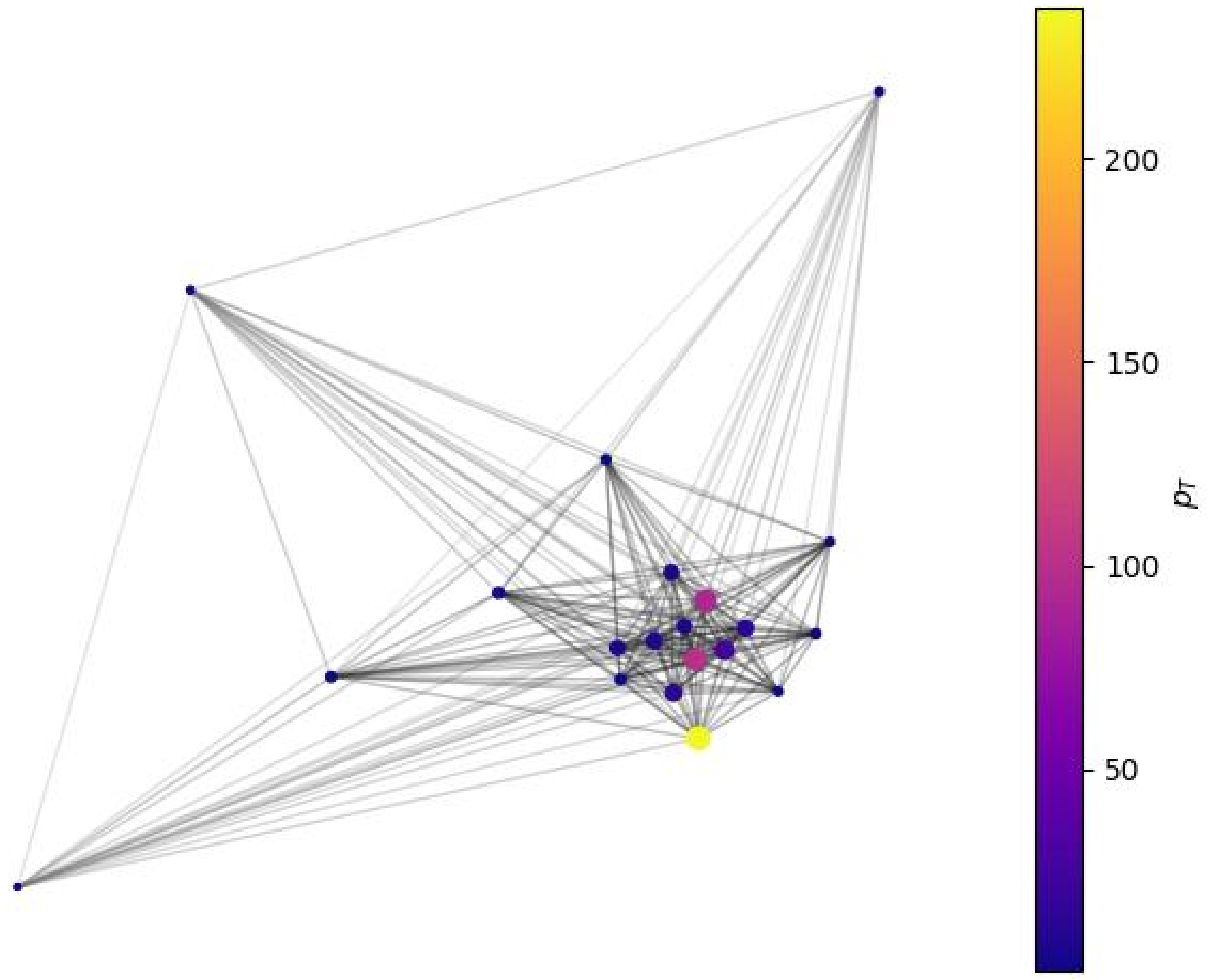
Pixelated



Representations: Graph

- The same goes for the graph representation.

```
graphs = Graph(
    Set(
        "fatjet0.constituents.eta",
        "fatjet0.constituents.phi",
        "fatjet0.constituents.pt",
    ),
    "fatjet0.constituents,fatjet0.constituents.delta_r",
).read(events, cut.array)
```



APIs

Create datasets

- Create datasets with the data fetched by different representations.

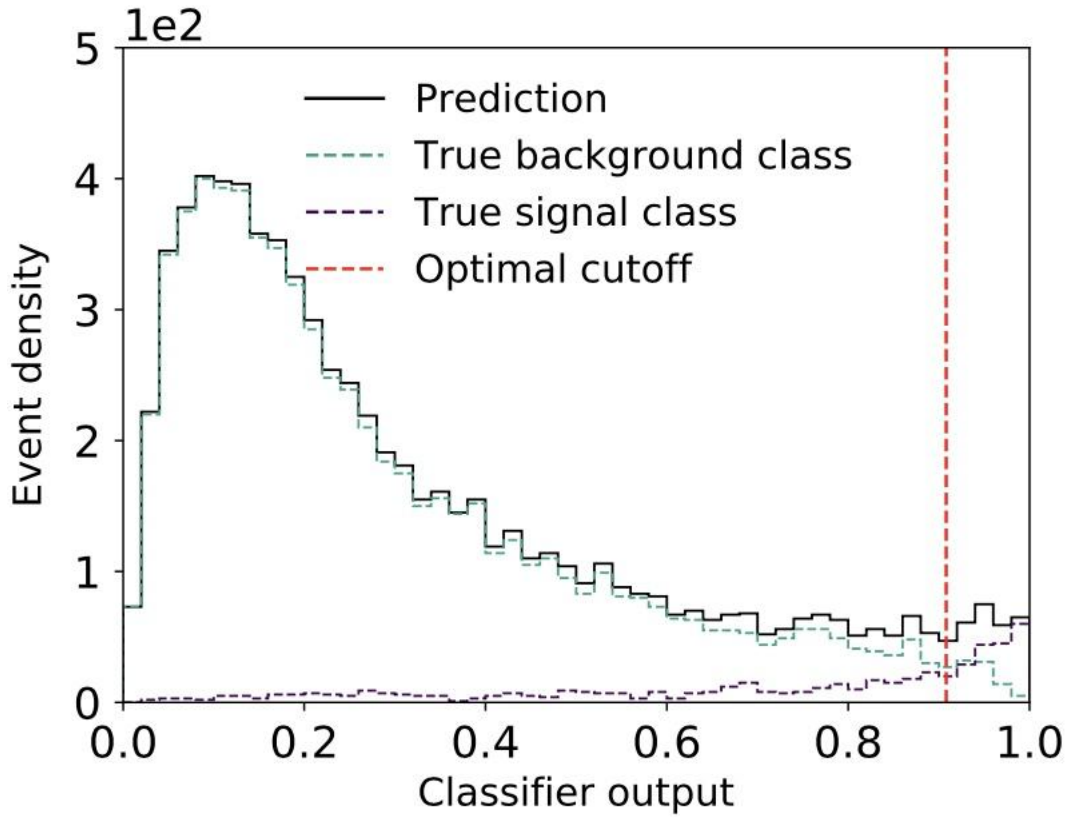
```
ds = SetDataset(
    [
        "fatjet0.mass",
        "fatjet0.tau21",
        "ak4fatjet0,ak4fatjet1.delta_r",
    ]
)
ds.read(events, target=0, mask=cut1.array & cut2.array)
ds.split(0.7, 0.2, 0.1)
ds.save("dataset.ds", piece_size=10)
```

```
A set dataset:
- train: x: 65 * 3 * float32, y: 65 * int32
- test: x: 19 * 3 * float32, y: 19 * int32
- val: x: 10 * 3 * float32, y: 10 * int32
```

Approaches

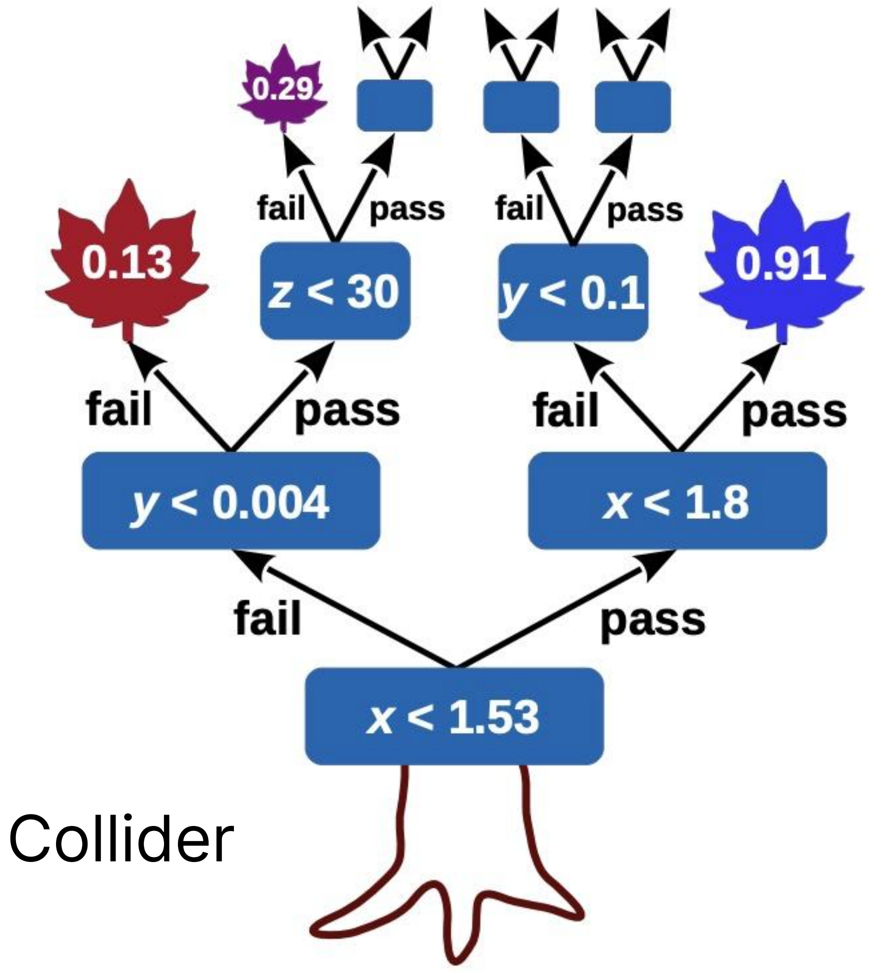
- Designed to contain three kinds of approaches:
  - cut and count
  - tree
  - neural networks

[2108.03125] Beyond Cuts in Small Signal Scenarios

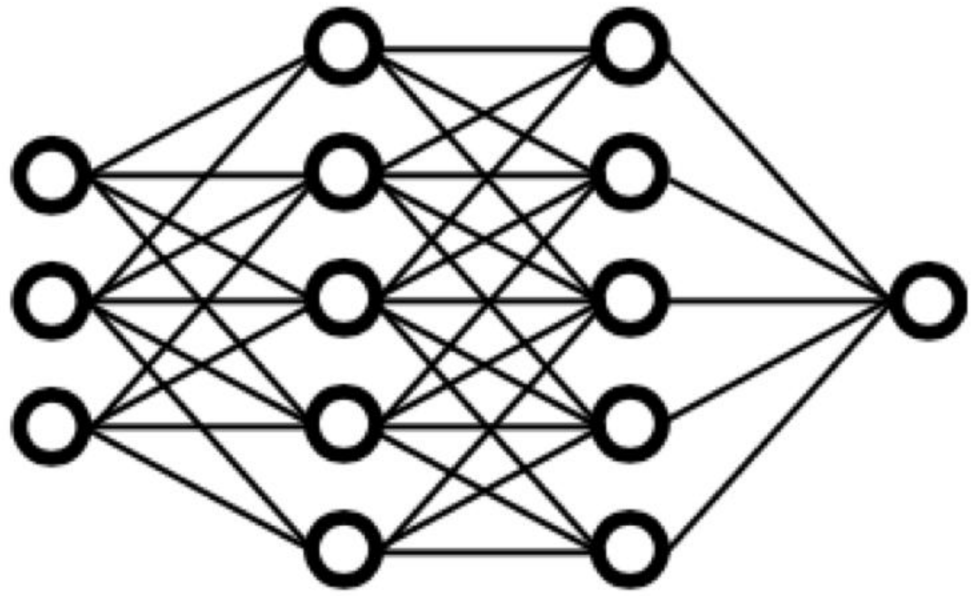


(a) XGBoost with optimized cutoff at 0.9081.

[2206.09645] Boosted decision trees



[1709.04464] Jet Substructure at the Large Hadron Collider



APIs

Approaches: protocol

init(...)	What does structure it have?
compile(optimizer, loss, metrics)	How to improve the weights?
fit(x, y, epochs...)	Other parameters of train process
evaluate(x, y)	How performant is it on other data?
summary()	Show model information
predict(x)	Given data, what is predicted value?

- No need to re-wrap Keras model.

Approaches: metrics

- **MaxSignificance** calculates the maximum significance under uniform distributed thresholds.

$$\text{significance} = \sqrt{2\left((S+B)\ln\left(1+\frac{S}{B}\right)-S\right)}$$

- **RejectionAtEfficiency** calculates the background rejection at a given signal efficiency.

$$1/\varepsilon_b \text{ at } \varepsilon_s = 50\,\%$$

APIs

Apply approaches

- Initialize methods to define their structure respectively.
- Compile each to determine how to improve itself and monitor performance.
- Fit methods' weights on dataset.

```
m1 = BoostedDecisionTree(n_estimators=10)
m2 = CutAndCount(n_bins=100)
m3 = ToyMLP(input_shape=(x_train.shape[1],))

m1.compile(
    loss=CategoricalCrossentropy(),
    metrics=[
        CategoricalAccuracy(name="acc"),
        MaxSignificance(name="max_sig"),
        RejectionAtEfficiency(name="r50"),
    ],
)
m2.compile(...)
m3.compile(...)

m1.fit(x_train, y_train)
m2.fit(...)
m3.fit(...)

```

APIs

Apply approaches

```python
from tabulate import tabulate

results1 = method1.evaluate(x_test, y_test)
results2 = method2.evaluate(x_test, y_test)
results3 = method3.evaluate(x_test, y_test, verbose=2)
results = {}

results["name"] = [method1.name, method2.name, method3.name]
for k in results1.keys():
    results[k] = results1[k] + results2[k] + results3[k]

print("> Results:")
print(tabulate(results, headers="keys", floatfmt=".4f"))
```

```
> Results:
name                      loss     acc    max_sig       r50
----------------------  ------  ------  ---------  --------
boosted_decision_tree   0.2611  0.9586   601.7032  647.3771
cut_and_count           4.4163  0.8037   243.9667   33.6241
toy_mlp                 0.5475  0.9350   111.5401  444.2333
```

- Evaluate methods using metrics defined in compile methods. Could also compile once again to use other metrics.
- Later more metrics will be added to complete benchmark.

Roadmap

v0.4.3(May)

- Prepare basic scaffold.
- Design all modules

v0.5.0(~Aug.)

- Refactor the naming convention.
- New representation: Graph
- New datasets
  - W
  - Z
  - Top
- New models:
  - ParticleNet

v0.6.0(~Sep.)

- Benchmarks
  - Jet tagging
  - New physics

Roadmap

- [2405.02888] HEP ML Lab: An end-to-end framework for applying machine learning into phenomenology studies
- [2303.15920] Probing Heavy Neutrinos at the LHC from Fat-jet using Machine Learning
- https://github.com/Star9daisy/hep-ml-lab

```
pip install hep-ml-lab
```

THANK YOU!