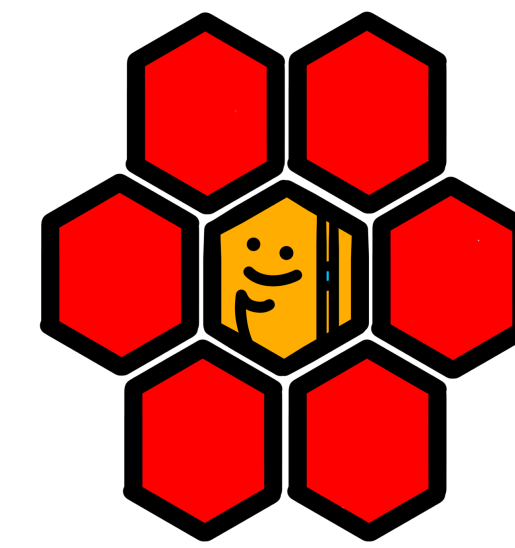


b-hive:

a modular training framework for state-of-the-art object-tagging within the Python ecosystem at the CMS experiment



Motivation:

- **Everybody wants to do Machine-Learning trainings**
- Full end-to-end pipeline is way harder than an example Notebook
 - Big data processing (ROOT files)
 - Conversion into a ML-friendly format (.npy/ .npz)
 - Deploy state-of-the-art models
 - ParticleNet (graph-convolutions)
 - Transformer models

Also:

- Have clearly defined workflows (not your 7 bash scripts!)
- Make trainings repeatable
- Standardized evaluation tools

Niclas Eich
On behalf of
the CMS Collaboration



III. Physikalisches
Institut A



EUROPEAN AI FOR FUNDAMENTAL PHYSICS CONFERENCE EuCAIFCon 2024

b-hive:
a modular training framework for state-of-the-art object-tagging within the Python ecosystem at the CMS experiment

Niclas Eich
On behalf of the CMS Collaboration
niclas.eich@rwth-aachen.de

Motivation:
Many working-groups have shared needs for ML training-tools!
• Software Framework for ML-tasks in HEP
• State-of-the-art object-tagging (b/c jets, taus)
• Purely pythonic
• Reproducible pipelines with **law**

Dataset Construction:
• Convert ROOT files into .npy files
• coffea and awkward for the big data processing
• Weight-calculation for stochastic sampling

Training:
• API defined with abstract base classes
• Any ML-library useable
• Task is agnostic of training details
→ applications implement their specifics
• Inheritance for fast development

Modularity:
Steering complex applications with many parameters leads to a big mess
→ Provide a modular setup to split apart code for different applications
• coffea-processors
• Models
• Dataloaders

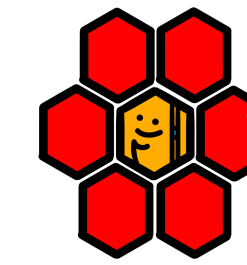
Inference:
• Models can be easily evaluated on different datasets
• Task tree is built to reuse existing datasets, models, ...

Evaluation:
• Automatic standardised evaluation tools for comparisons
• Production of ROC-curves (left), working point curves (right)
• Application specific tests can be added as tasks

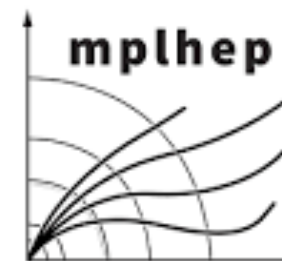
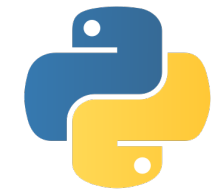
Example:
Retraining for new conditions leads to improved results, with a pipeline that is easily repeatable.

Example:
Automatic working point evaluation leads to faster development and deployment cycles.

b-hive attacks these problems



- Pythonic training framework
 - Workflow management with law
 - coffea, awkward, numpy for the heavy data lifting
 - No ML-framework lock in
 - TensorFlow and PyTorch can be used
- Modular Setup



- Easy configuration for different working-groups
 - New applications are embedded in the pipeline
- Knowledge-sharing by code-sharing
- Have a look: [CERN-CMS-DP-2024-020](#)

```
42 bins_pt:
43 - 30
44 - 100
45 - 500
46 - 2000
47
48 bins_eta:
49 - -2.5
50 - -2.0
51 - -0.5
52 - 0.5
53 - 2.0
54 - 2.5
55
56 treename:
57 "producer/custom_tree"
58
59 global_features:
60 - "jet_pt"
61 - "jet_eta"
62 - "n_Cpfcand"
63 - "n_Npfcand"
64 - "nsv"
65 - "npv"
66
67 pf_candidate_features:
68 - "pfcand_pt"
69 - "pfcand_eta"
70 - "pfcand_trackPt"
71 - "pfcand_trackEta"
72 - "pfcand_trackDeltaR"
73
74 truths:
75 - "isB"
76 - "isUDS"
77 - "isG"
78
```

