

b-hive: a CMS wide Machine Learning Framework

Ulrich Willemsen
On behalf of the CMS collaboration

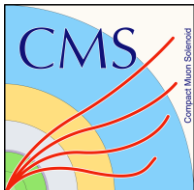
ML4Jets, Vienna, 2026



With funding from the:

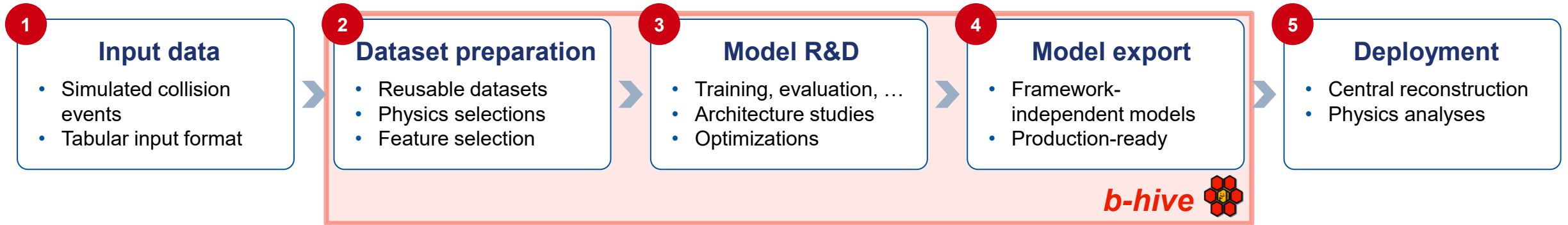


Federal Ministry
of Research, Technology
and Space



ML research workflows

Developed for R&D in CMS jet-flavour tagging



Targeting challenges in any working group with ML workflows

- Read HEP data formats
- Workflow management
- Keeping track of training configurations
- Efficient hardware utilization
- Private code bases
 - Everyone uses their own training scripts
- Comparability of results
 - Different preprocessing
 - Different metric definitions

A CMS wide ML framework

b-hive: end-to-end framework for HEP data formats with state-of-the-art ML tools

- Full pipeline: from dataset export to model evaluation
- Convert root/parquet data formats to layouts for ML libraries (e.g. Pytorch)
- Reproducible model evaluation



Modular task architecture

- Luigi analysis workflow (law) based pipeline
- Swap components with minimal code changes
- Contains dataset construction, training, inference, and model evaluation

Application-agnostic top-level code

- Modular use of physics-specific ML choices
- Plug-in architecture, hyperparameters, kinematic selections, ...
- YAML config files

Collaboration

- Central GitLab repository
- Components can be mixed-and-matched across groups
- Consistent quality standards

Framework building blocks

Task pipeline

- law and luigi
- Keeps track of parameterised task chains

Dataset

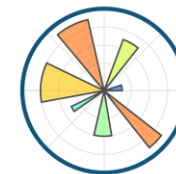
- awkward, coffea, uproot
- Storing in compressed LZ4 or NumPy format

Model training and inference

- PyTorch, Tensorflow, Keras, JAX, LightGBM
- Metrics tracking with MLflow

Evaluation and plotting

- Matplotlib and mplhep
- Predefined scripts for ROC curves, working point plots, and input histograms



Modular task architecture

Law-based training pipeline

- Reproducibility of results
- Independent execution of tasks with different task versions
- Modular choice of parameters

DatasetConstructionTask

- Converts ROOT files to LZ4/numpy files
- Advanced selections possible

TrainingTask

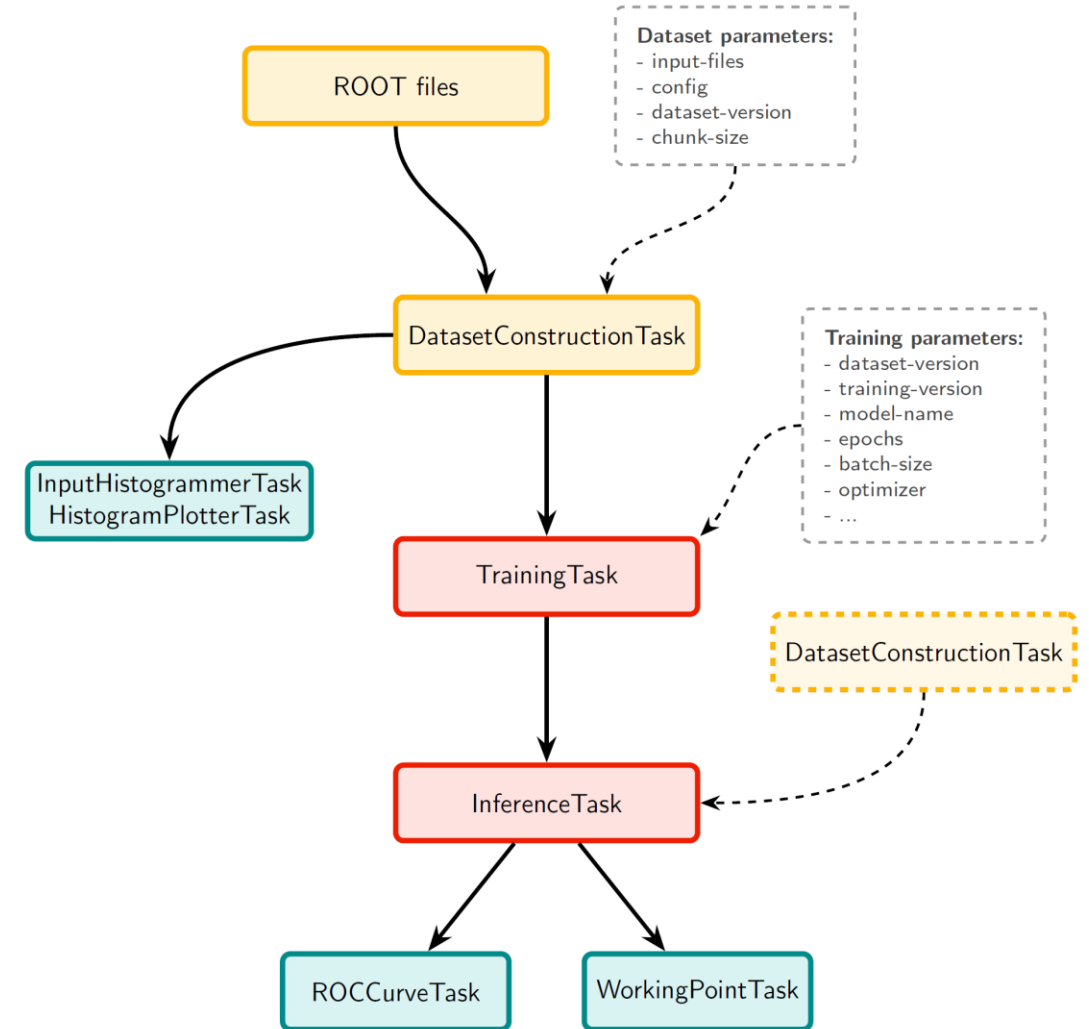
- Trains model on *dataset-version*

InferenceTask

- Evaluation on *test-dataset-version*

Plotting tasks

- Produces ROC curves, loss plots, working point plots, input histograms



Law keeps track of everything

```
$ law run ROCCurveTask --dataset-version trainset_01 --config JetClass --training-version JetClass_training_01  
--model-name ParticleTransformer --epochs 50 --test-dataset-version testset_01
```

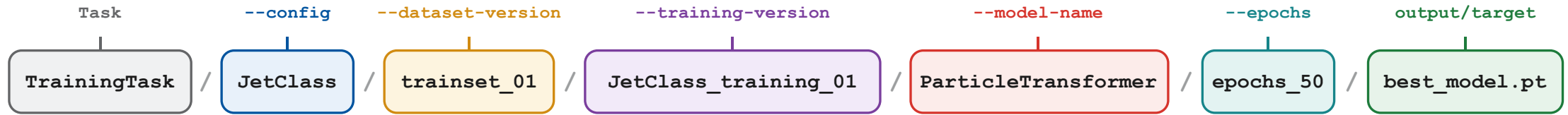
Dependency tree



➤ Pipeline checks for missing dependencies → Only runs necessary tasks → No need to re-train!

Parameters are the configuration

Command-line parameters



Path of the stored target

.../bhive_output/TrainingTask/JetClass/trainset_01/JetClass_training_01/ParticleTransformer/epochs_50/best_model.pt

- Law knows which tasks already ran
 - An existent task does not run again
- Nothing is overwritten
 - Different parameters → different path
 - Re-running requires verification of target removal
- Keeping track of metadata
 - Output stored under parameters that produced it
 - Tracing back to their exact configuration
- Reproducibility is enforced by b-hive

Collaborating

Started as project within the CMS jet-flavour tagging group

- Got many collaborators across different groups
- Provides state-of-the-art jet tagging algorithms
- Not restricted to object tagging

Users can specify different NN architectures or use pre-existing models

- Model configuration through config files
- No alternation of core code needed
- Develop in a modular piece to prevent diverging forks and hard-coding

2 types of configs

- Dataset config (yaml format)
- Model config (define as property of the model)

```
1 bins_pt:
2   - 30
3   - 35
4   - ...
5   - 600
6   - 2001
7
8 bins_eta:
9   - -2.5
10  - -2.0
11  - ...
12  - 2.0
13  - 2.5
14
15 tree_name:
16   "deepntuplizer/tree"
17
18 truths:
19   - "isB"
20   - "isC"
21   - "isUD"
22   - "isS"
23   - "isG"
24
25 reference_flavour:
26   "isB"
27
28 global_features:
29   - "jet_pt"
30   - "jet_eta"
31   - "jet_phi"
32   - ...
33
34 cpf_candidates:
35   - "Cpfcand_e"
36   - ...
37
38 npf_candidates:
39   - "Npfcand_ptrel"
40   - ...
41
42 vtx_features:
43   - "sv_pt"
44   - ...
45
46 global_custom_features:
47   - jet_px: "np.cos(jet_phi) * jet_pt"
48   - jet_py: "np.sin(jet_phi) * jet_pt"
49   - jet_pz: "np.sinh(jet_eta) * jet_pt"
50
51 cpf_custom_features:
52   - Cpfcand_mass: "np.sqrt(Cpfcand_e**2 - Cpfcand_px**2 - Cpfcand_py**2 - Cpfcand_pz**2)"
53
54 class DeepJet(Classifier_base):
55     n_cpf_candidates = 25
56     n_npf_candidates = 1
57     n_vtx_candidates = 5
58
59     classes = {
60         "b": ["isB"],
61         "c": ["isC"],
62         "uds": ["isUD", "isS"],
63         "g": ["isG"],
64     }
65
66     global_features = [
67         "jet_px",
68         "jet_py",
69         "jet_pz",
70         "jet_energy",
71         "TagVarCSV_trackSumJetEtRatio",
72         "TagVarCSV_trackSumJetDeltaR",
73         "TagVarCSV_vertexCategory",
74         "TagVarCSV_trackSip2dValAboveCharm",
75         "TagVarCSV_trackSip2dSigAboveCharm",
76         "TagVarCSV_trackSip3dValAboveCharm",
77         "TagVarCSV_trackSip3dSigAboveCharm",
78         "TagVarCSV_jetNSelectedTracks",
79         "TagVarCSV_jetNTracksEtaRel",
80     ]
81
82     cpf_candidates = [ ... ]
83     npf_candidates = [ ... ]
84     vtx_features = [ ... ]
85
86     def __init__(self, ...):
87         ...
88
89     def forward(self, inpt):
90         ...
```

pt, η bins

Truth labels

Processed features

Number of candidates

Targets

Custom features

Input features

Architecture

Adding your own model

Example: Add a new jet tagging model

1

Define the model

- Inherit an existing model, or write a new one
- Inputs and hyper-parameters come from the config
- The framework provides everything else: dataloading, training, ...

Option A inherit an existing model

```
class MyCustomTagger(Part):  
    def __init__(self, **kw):  
        super().__init__(**kw)  
  
    # only what differs from Part  
    self.n_cpf = 30  
    ...
```

Option B write a new architecture

```
class MyCustomTagger(ClassifierBase):  
    def __init__(self, **kw):  
        super().__init__(**kw)  
  
    def forward(self, inpt):  
        # your architecture  
        ...
```

2

Register the model

- Add the tagger to the model loader
- Only here core code is modified
- Everything else is handled by configs

Model loader `b-hive/utils/models/models.py`

```
class ModelName:  
    MyCustomTagger = "MyCustomTagger"  
  
case ModelName.MyCustomTagger:  
    return MyCustomTagger(args, **kwargs)
```

3

Train it

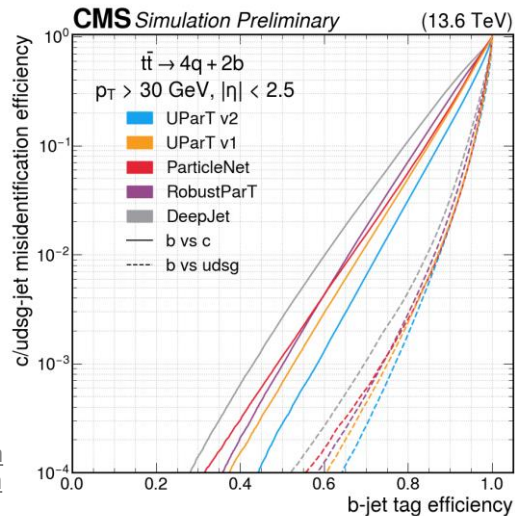
```
$ law run TrainingTask --model-name MyCustomTagger ...
```

➤ Easy adaptation of existing models and workflows!

Available Models

Jet tagging algorithms

- UParT v2  **Used in production**
- ParticleNet [Phys. Rev. D 101, 056019 (2020)]
- DeepJet [JINST 15 (2020) P12012]
- DeepJet Transformer [Eur.Phys.J.C 85 (2025) 2, 165]
- Particle Transformer [arXiv:2202.03772]
- PAIRed Tagger [JHEP09(2024)128]



[CMS-DP-2025-081]



➤ **And much more!**

Simple scripts and nice-to-haves

- Resume/extend training
- Inference on multiple datasets
- Adversarial auxiliary calculations
- Process, class, and kinematic weights
- ONNX export
- Iteration-based training
- Mlflow tracking of training metrics

Tools

- Learning rate schedulers
- SOTA optimizer
- Adversarial attacks
- Torch compilation

Talk about scaling laws in CMS by Pavlo Kashko on Thursday

Performance benchmarks

CPU overhead reduction

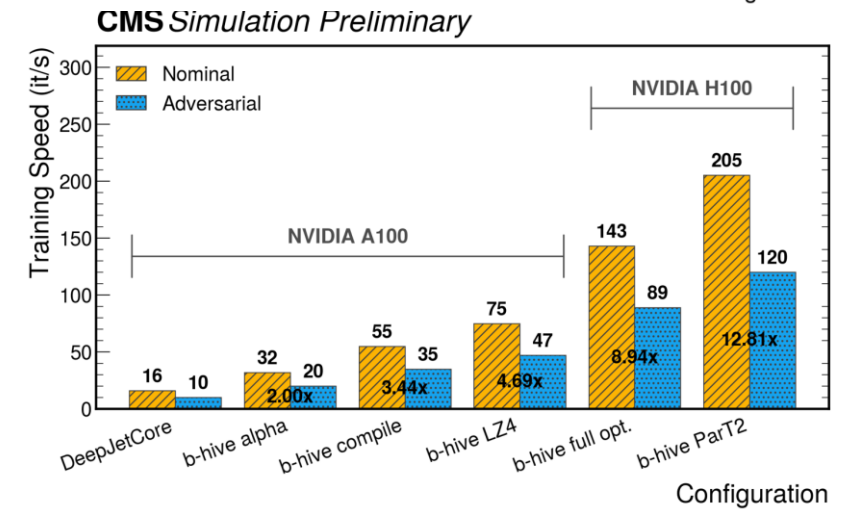
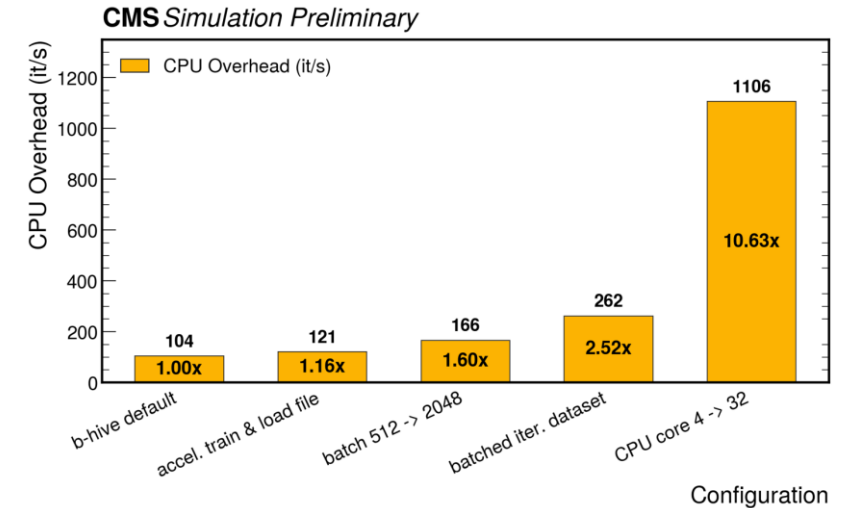
- Implemented many state-of-the-art PyTorch optimizations
 - Memory pinning
 - Persistent workers and prefetch tuning
 - Batched iterable dataloader

➤ You really spend time training your model and not iterating on data!

Training Speed

- Improved GPU utilization and FLOPS extraction
 - ~99% GPU utilization with torch compilation
- Further improve speed by implementing multi-GPU usage in the future

➤ Developments ongoing!



Training speed of a default ParT model (3+1 attention layers, dim=128 and nhead=8)

Summary

b-hive: a general state-of-the-art ML framework for CMS

- Modular framework
- Unified code base between subgroups
- SOTA models and tools
- Workflow management tools → reproducibility, metadata dump

Generalization of b-hive

- Include foundations for various applications (e.g. generative models, autoencoder, ...)
- Easy selection of already predefined architectures (using yaml model configs)
- Remove the jet-tagging naming conventions

Scaling of training capabilities

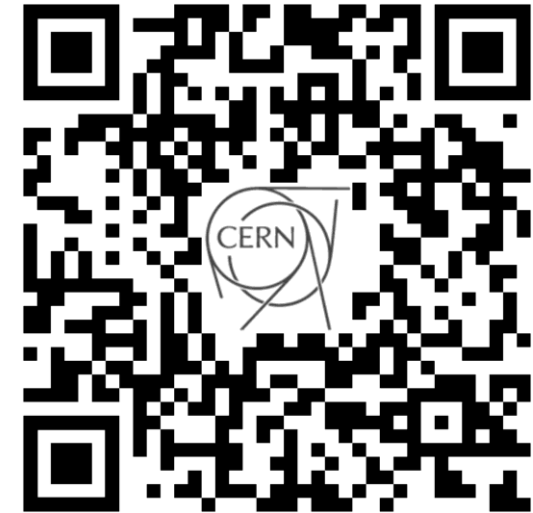
- Parallelization of training
- Stay up-to-date with even faster datasets



Condensed links



[Gitlab](#) code-base



DP Note

Backup

Using b-hive outside CMS

CMS-specific

- Jet-based inputs, feature naming conventions
- Naming convention of input categories (for now)
- Process assignment by substring match on filelist paths

➤ The CMS-specific part are the inputs, not the framework!

Generic

- low task graph and parameter-derived output paths
- Training loop, checkpointing, schedulers, optimizers, adversarial attacks
- LZ4 / numpy IterableDataset with multi-worker file splitting
- Model implementations: ParT, ParticleNet, DeepJet, PAIReD
- ROC, working-point and input-feature plotting

Adapting b-hive to another application

- Write one YAML config (and potentially one processor) → no core code changes!
- Merging, weighting, data loading, training and evaluation are reused unchanged
- Apache 2.0 license

Example: JetClass training in b-hive

JetClass dataset config

```
global_features:
- "jet_pt"
- "jet_energy"
- ...

cpf_candidates:
- "part_px"
- "part_py"
- "part_pz"
- "part_energy"
- ...

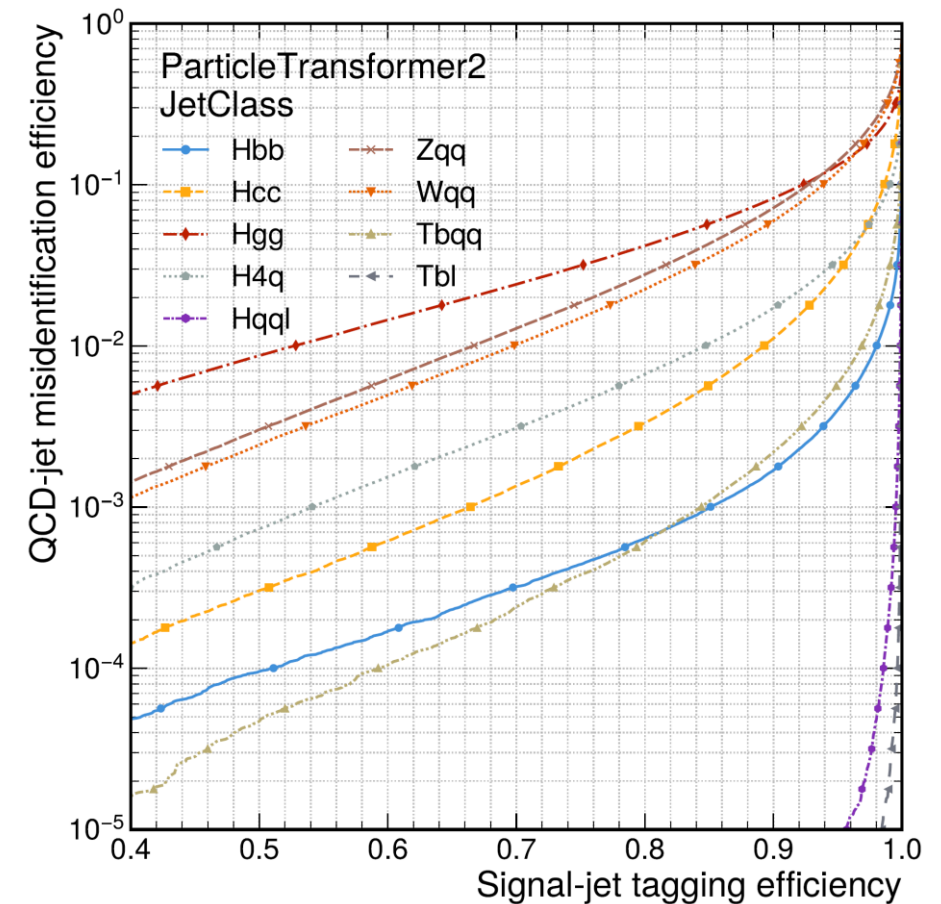
cpf_custom_features:
- part_logpt_rel: "np.log(np.sqrt(part_px**2 +
  part_py**2)/jet_pt)"
- ...

n_cpf_candidates: 128

truths:
- "label_QCD"
- ...
- "label_Tbl"

reference_flavour: "label_Hbb"
```

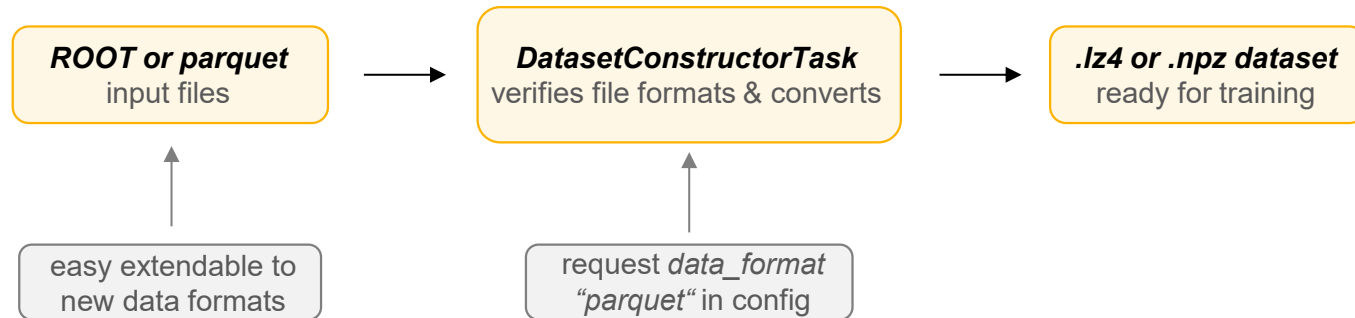
ROC curves produced with b-hive



Data formats

Parquet as input data type

- Default is flattened ROOT format
- New input formats can be easily added!
- No mixed filelists



What it looks like in practice

- Filelist example

```
/data/parquet/columns_0.parquet
/data/parquet/columns_1.parquet
/data/parquet/columns_2.parquet

/data/root/file_1.root
```

- Config example

```
data_format:
  "parquet"

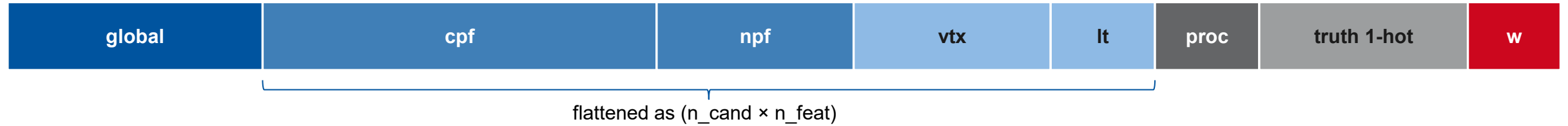
global_features:
  "n_jet"

cpf_candidates:
  "Jet_pt"

truths:
  "is_Hcc"
  "is_Hbb"
```

Constructed dataset format

One flat float row per jet



Layout

- One row per jet, dtype from the config *data_precision*
- Rows containing NaN or inf are dropped during processing
- Column edges are derived from the same config that built the file

Why flat rather than structured

- One contiguous read per file
 - no per-jet object decoding in the training loop
- Cheap random access after the merge step has already shuffled

pT- η reweighting and sampling

How the weights are built

- One (pT, η) histogram per truth label, filled during dataset construction
- Reference and class histograms are each normalised to their own maximum
- $\text{weights}[\text{class}] = \text{reference_hist} / \text{class_hist}$
- At training: keep the jet if $\text{rand}() < \text{weight}$, reject everything else
- Switched off during inference

Worked example: two pT bins

	bin 1	bin 2	total
Reference class	100	50	150
Other class	200	200	400
Sampling weight	1.0	0.5	—
Other class, resampled	200	100	300

- Shape now matches the reference (2:1 in both bins)

- The pT/ η shape of every class is matched to the reference class, not the total number of members per class
- Alternatively: enable loss weighting in b-hive

2 types of configs

Dataset config

- yaml format

Bins for histogram reweighting

- Default is *jet_pt* and *jet_eta*
- Change by specifying "*pt_key*", "*eta_key*" in config

Feature branches

- 4 input categories (for simple classifier, just use "*global_features*")
 - Specify **number of candidates** per jet per feature
 - Global features has one entry per jet/event per feature
- Compute **custom features** that are not available in the ROOT files

- Only properties of dataset! No training specification!
- Create one dataset with all features → Use subset of features from same *--dataset-version* as inputs
- Specification in model config!

```
config > ! example_hlt_config.yml
1  bins_pt:
2    - 15
3    - 30
4    - ...
5    - 400
6    - 1000
7
8  bins_eta:
9    - -2.5
10   - -2.0
11   - ...
12   - 2.0
13   - 2.5
14
15  treename:
16    "DeepJetNTupler/DeepJetvars"
17
18  truths:
19    - "isB"
20    - "isC"
21    - "isUD"
22    - "isS"
23    - "isG"
24
25  reference_flavour:
26    "isB"
27
28  processes:
29    - "TT"
30    - "QCD"
31    - "DY"
32
33  TT:
34    pt_min: 30
35    pt_max: 1000
36    eta_min: -2.5
37    eta_max: 2.5
38
39  n_cpf_candidates: 25
40  n_npf_candidates: 25
41  n_vtx_candidates: 5
42
43  global_features:
44    - "jet_pt"
45    - "jet_eta"
46    - "jet_phi"
47    - ...
48
49  cpf_candidates:
50    - "Cpfcane"
51    - ...
52
53  npf_candidates:
54    - "Npfcantptrel"
55    - ...
56
57  vtx_features:
58    - "sv_pt"
59    - ...
60
61  global_custom_features:
62    - jet_px: "np.cos(jet_phi) * jet_pt"
63    - jet_py: "np.sin(jet_phi) * jet_pt"
64    - jet_pz: "np.sinh(jet_eta) * jet_pt"
65
66  cpf_custom_features:
67    - Cpfcantmass: "np.sqrt(Cpfcane**2 - Cpfcantpx**2 - Cpfcantpy**2 - Cpfcantpz**2)"
```

p_T, η bins

Truth labels

Processed features

Process selections

2 types of configs

Model config

- Property of the model

Classes

- Output nodes need to correspond to number of target groups
- Aggregate labels into target groups

Input features

- Features used during training of the model
- Subsample of the features specified in dataset config
- Access **custom features** to use as inputs
- Add/remove features for different trainings
- Change **number of candidates** here to change them for the training

➤ Easy adaptation of existing models via inheritance

```
24 class DeepJet(Classifier_base):
25     n_cpf_candidates = 25
26     n_npf_candidates = 1
27     n_vtx_candidates = 5
28
29     classes = {
30         "b": ["isB"],
31         "c": ["isC"],
32         "uds": ["isUD", "isS"],
33         "g": ["isG"],
34     }
35
36     global features = [
37         "jet_px",
38         "jet_py",
39         "jet_pz",
40         "jet_energy",
41         "TagVarCSV_trackSumJetEtRatio",
42         "TagVarCSV_trackSumJetDeltaR",
43         "TagVarCSV_vertexCategory",
44         "TagVarCSV_trackSip2dValAboveCharm",
45         "TagVarCSV_trackSip2dSigAboveCharm",
46         "TagVarCSV_trackSip3dValAboveCharm",
47         "TagVarCSV_trackSip3dSigAboveCharm",
48         "TagVarCSV_jetNSelectedTracks",
49         "TagVarCSV_jetNTracksEtaRel",
50     ]
51
52 > cpf_candidates = [ ...
75
76 > npf_candidates = [ ...
89
90 > vtx_features = [ ...
107
108 > def __init__(self, ...
154
155 > def forward(self, inpt): ...
178
```

Number of candidates

Targets

Custom features

Input features

Architecture

Coming soon

New model definition

- Split each model into reusable pieces
- Wire a model together with a config

Config (YAML)

Specify architecture · inputs · classes · hyper parameters



Factory building from config

Reads config → looks up registry → assembles model



Model = Backbone + Head

Backbone

- Layer building blocks
- Pure architecture

Head

- Output & loss
- Classification, regression, generative, ...
- No architecture

- A model is one call

```
# config fully describes the model

from models import build

model = build(config)
```

- ...specified in the config

```
backbone:
  name: particletransformer
  embed_dim: 128
  num_heads: 8
head:
  name: classification
  num_classes: 6
inputs:
  category1: ...
  category2: ...
  :
```

Custom selections

Custom coffea processor

- Inherit from existing processors
 - LZ4Processing
 - PFCandidateAndVertexProcessing
- Modify your own processor
 - Include custom selections (hardcoded)
 - Will be a property of your dataset
 - A lot of control about your selections
 - But you need to keep track of changes yourself!
 - Also possible via dataset config (modular)
- Add it to the processor loader
- Specify **processor** in the dataset config

```
processor:  
  "LZ4_BestTaggerEverProcessor"
```

```
class LZ4_BestTaggerProcessor(LZ4Processing):  
  
    def callColumnAccumulator(self, output, events, flag, **kwargs):  
        global_arr, cpf_arr, npf_arr, vtx_arr, truth_arr, process = super().callColumnAccumulator(  
            output, events, flag, **kwargs)  
  
        selection_mask = (  
            (~global_arr['includesIsoLepJet'].astype(bool)) &  
            (global_arr['MC_gen_flav'] == 0) &  
            (truth_arr['label_ll'] == 1)  
        ) | (  
            truth_arr['label_cc'] == 1  
        ) | (  
            truth_arr['label_bb'] == 1  
        )  
  
        return (  
            global_arr[selection_mask],  
            cpf_arr[selection_mask],  
            npf_arr[selection_mask],  
            vtx_arr[selection_mask],  
            truth_arr[selection_mask],  
            process[selection_mask],  
        )  
  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
  
def ProcessorLoader(model: str = "", *args, **kwargs):  
    match model:  
        case ProcessorClasses.PFCandidateAndVertexProcessing:  
            return PFCandidateAndVertexProcessing(*args, **kwargs)  
        case ProcessorClasses.LZ4Processing:  
            return LZ4Processing(*args, **kwargs)  
        case ProcessorClasses.LZ4FP16Processing:  
            return LZ4FP16Processing(*args, **kwargs)  
        case ProcessorClasses.L1PFCandidateAndVertexProcessing:  
            return L1PFCandidateAndVertexProcessing(*args, **kwargs)  
        case ProcessorClasses.PairedTaggerProcessor:  
            return PairedTaggerProcessor(*args, **kwargs)  
        case ProcessorClasses.LZ4_PairedTaggerProcessor:  
            return LZ4_PairedTaggerProcessor(*args, **kwargs)  
        case ProcessorClasses.LZ4_BestTaggerEverProcessor:  
            return LZ4_BestTaggerEverProcessor(*args, **kwargs)  
        case _:  
            return PFCandidateAndVertexProcessing(*args, **kwargs)
```

Training-loop options

Default training-loop

- Epoch-based training
- Save model every epoch or only best model
- Standard mini-batch SGD

Additional training-loop possibilities

Gradient accumulation

- Simulates a larger effective batch size with no extra GPU memory
- Specify how many gradient calculations per batch
- Gradients from small steps are summed before optimizer step

Iteration-based training

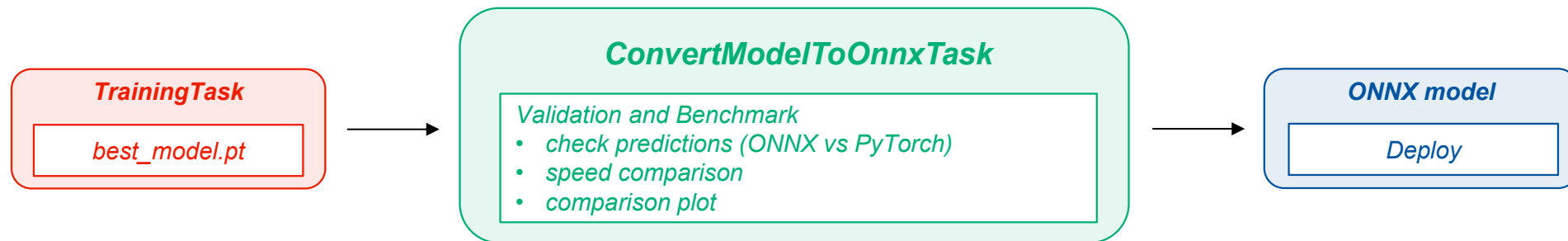
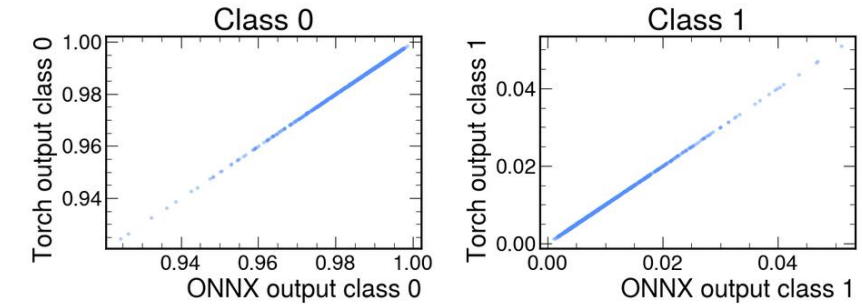
- Train for a fixed number of batches instead of epochs
- Checkpoint and validate on an iterative basis

Convert model to ONNX

Soon available

Conversion of the best model from the TrainingTask to ONNX

- For integration into CMSSW
- Takes best trained model checkpoint
- Feeding of realistic “dummy” example data → Trace output computation
- Numerical verification of converted model predictions with PyTorch model predictions
 - Across different batch-sizes
 - Across different particle-count scenarios



What changes

Adding a new model

Before: edit the source

```
# 1. add a name
class ModelName:
    MyNewModel = "MyNewModel"

# 2. add a case to the factory
def BTaggingModels(model):
    match model:
        case ModelName.MyNewModel:
# 3. import & return the class
        from ... import MyNewModel
        return MyNewModel(...)
```

30 hardcoded names in *ModelName* class and growing



After: add a config

```
# a new backbone: one decorator,
# zero edits to core code

@register_backbone("my_new_model")
class MyNewBackbone(BaseBackbone):
    def forward(self, inputs):
        ...

# most models: no new code at all –
# just a YAML file
```

Most new models: no new code at all – just a new YAML file

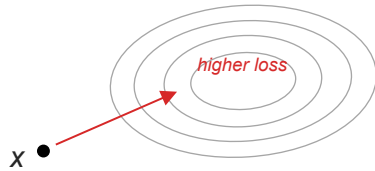
➤ Model variants are just new configs

Available adversarial attacks

FGSM

[arXiv:1412.6572 [stat.ML]]

Fast Gradient Sign Method



Method

- One step of gradient ascent on the loss
- $x_{adv} = x + \epsilon \cdot \text{sign}(\nabla_x L)$

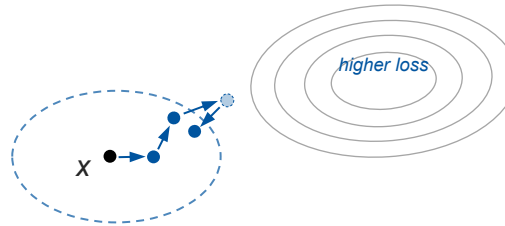
Use

- Sanity check
- Fast to generate & easy to defend against

PGD

[arXiv:1607.02533 [cs.CV]]

Projected Gradient Descent



Method

- Iterated gradient ascent on the loss
- After each step the point is projected back into the ϵ -ball

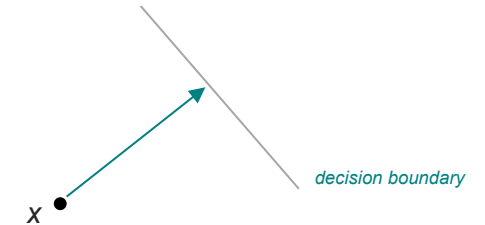
Use

- Standard strong white-box attack
- Robustness benchmark

JetFool

[arXiv:1511.04599 [cs.LG]]

DeepFool, adapted for jets



Method

- Linearise the classifier
- Step orthogonally to the nearest decision boundary
- Repeats until the decision flips

Use

- Sanity check on why the network is uncertain

Available adversarial attacks

MiniFool

- Physics-constraint-aware minimizer-based attack
- *What is the most probable change that flips the classification, given experimental uncertainties?*

Cost function

$$\lambda = \eta(\Delta x / \sigma) + \beta \cdot (f_{i^*} - g)^2$$

Uncertainty penalty (χ^2 -like)

- per-feature shift / uncertainty σ
- large, unphysical moves cost a lot
- attack parameter s scales uncertainties: $\sigma = s \cdot \sigma^0$

Target score

- push the output toward a flipped decision ($g = 0$)

- Use s as robustness measure for classification flip:
 - If $s \gg 1$: networks classification of the jet is robust
 - If $s \leq 1$ (perturbation $\leq$ uncertainty): networks classification of the jet is fragile
- MiniFool will not flip a decision if the required change cost too much vs. uncertainties

Law keeps track of everything

```
$ law run ROCCurveTask --dataset-version trainset_01 --config hlt_run3 --training-version hackathon_tutorial_01  
--model-name DeepJetHLT --epochs 5 --test-dataset-version testset_01 --test-filelist ../test_files.txt --batch-size 256  
--print-status 1
```

```
0 > ROCCurveTask(debug=False, terminal_plot=False, config=HLT_run3, verbose=False, seed=123456, dataset_version=trainset_01, training_precision=float32, test_dataset_version=testset_01, testing_precision=float32, training_version=hackathon_tutorial_01, epochs=5, model_name=DeepJetHLT, n_threads=4, batch_size=256, learning_rate=0.001, optimizer=AdamW, loss_function=CrossEntropyLoss, betas=(0.95, 0.999), eps=1e-06, lr_scheduler=epoch_linedecay, lr_decay_factor=0.01, mixed_precision=False, use_torch_compile=False, torch_compile_mode=default, attack=nominal, attack_magnitude=0.0, attack_restrict_impact=-1.0, attack_restrict_iterations=1, attack_restrict_overshoot=0.02, attack_restrict_uncertainty=1.0, test_attack_restrict_impact=-1.0, test_attack_restrict_overshoot=0.02, test_attack_restrict_uncertainty=1.0)  
LocalDirectoryTarget(fs=local_fs, path=/eos/user/u/uwillems/BTV/b-hive/task_artifacts/ROCCurveTask/hlt_run3/trainset_01/testset_01/hackathon_tutorial_01/DeepJetHLT/epochs_5/nominal/test_attack_nominal)  
absent  
-1 > TrainingTask(debug=False, terminal_plot=False, config=HLT_run3, verbose=False, seed=123456, dataset_version=trainset_01, training_precision=float32, test_dataset_version=testset_01, testing_precision=float32, training_version=hackathon_tutorial_01, epochs=5, model_name=DeepJetHLT, n_threads=4, batch_size=256, learning_rate=0.001, optimizer=AdamW, loss_function=CrossEntropyLoss, betas=(0.95, 0.999), eps=1e-06, lr_scheduler=epoch_linedecay, lr_decay_factor=0.01, mixed_precision=False, use_torch_compile=False, torch_compile_mode=default, attack=nominal, attack_magnitude=0.0, attack_restrict_impact=-1.0, attack_restrict_iterations=1, attack_restrict_overshoot=0.02, attack_restrict_uncertainty=1.0, test_attack_restrict_impact=-1.0, test_attack_restrict_overshoot=0.02, test_attack_restrict_uncertainty=1.0, resume_epoch=0, extend_training=0, train_val_split=0.85, n_train_files=1)  
training_metrics: LocalFileTarget(fs=local_fs, path=/eos/user/u/uwillems/BTV/b-hive/task_artifacts/TrainingTask/hlt_run3/trainset_01/hackathon_tutorial_01/DeepJetHLT/epochs_5/nominal/training_metrics.npz)  
validation_metrics: LocalFileTarget(fs=local_fs, path=/eos/user/u/uwillems/BTV/b-hive/task_artifacts/TrainingTask/hlt_run3/trainset_01/hackathon_tutorial_01/DeepJetHLT/epochs_5/nominal/validation_metrics.npz)  
model: LocalFileTarget(fs=local_fs, path=/eos/user/u/uwillems/BTV/b-hive/task_artifacts/TrainingTask/hlt_run3/trainset_01/hackathon_tutorial_01/DeepJetHLT/epochs_5/nominal/model_4.pt)  
best_model: LocalFileTarget(fs=local_fs, path=/eos/user/u/uwillems/BTV/b-hive/task_artifacts/TrainingTask/hlt_run3/trainset_01/hackathon_tutorial_01/DeepJetHLT/epochs_5/nominal/best_model.pt)  
absent  
-1 > InferenceTask(debug=False, terminal_plot=False, config=HLT_run3, verbose=False, seed=123456, dataset_version=trainset_01, training_precision=float32, test_dataset_version=testset_01, testing_precision=float32, training_version=hackathon_tutorial_01, epochs=5, model_name=DeepJetHLT, n_threads=4, batch_size=256, learning_rate=0.001, optimizer=AdamW, loss_function=CrossEntropyLoss, betas=(0.95, 0.999), eps=1e-06, lr_scheduler=epoch_linedecay, lr_decay_factor=0.01, mixed_precision=False, use_torch_compile=False, torch_compile_mode=default, attack=nominal, attack_magnitude=0.0, attack_restrict_impact=-1.0, attack_restrict_iterations=1, attack_restrict_overshoot=0.02, attack_restrict_uncertainty=1.0, test_attack_restrict_impact=-1.0, test_attack_restrict_overshoot=0.02, test_attack_restrict_uncertainty=1.0)  
LocalDirectoryTarget(fs=local_fs, path=/eos/user/u/uwillems/BTV/b-hive/task_artifacts/InferenceTask/hlt_run3/trainset_01/testset_01/hackathon_tutorial_01/DeepJetHLT/epochs_5/nominal/test_attack_nominal)  
absent
```

```
0 > ROCCurveTask(debug=False, terminal_plot=False, config=HLT_run3, verbose=False, seed=123456, dataset_version=trainset_01, training_precision=float32, test_dataset_version=testset_01, testing_precision=float32, training_version=hackathon_tutorial_01, epochs=5, model_name=DeepJetHLT, n_threads=4, batch_size=256, learning_rate=0.001, optimizer=AdamW, loss_function=CrossEntropyLoss, betas=(0.95, 0.999), eps=1e-06, lr_scheduler=epoch_linedecay, lr_decay_factor=0.01, mixed_precision=False, use_torch_compile=False, torch_compile_mode=default, attack=nominal, attack_magnitude=0.0, attack_restrict_impact=-1.0, attack_restrict_iterations=1, attack_restrict_overshoot=0.02, attack_restrict_uncertainty=1.0, test_attack_restrict_impact=-1.0, test_attack_restrict_overshoot=0.02, test_attack_restrict_uncertainty=1.0)  
LocalDirectoryTarget(fs=local_fs, path=/eos/user/u/uwillems/BTV/b-hive/task_artifacts/ROCCurveTask/hlt_run3/trainset_01/testset_01/hackathon_tutorial_01/DeepJetHLT/epochs_5/nominal/test_attack_nominal)  
absent
```

Law keeps track of everything

```
$ law run ROCCurveTask --dataset-version trainset_01 --config hlt_run3 --training-version hackathon_tutorial_01  
--model-name DeepJetHLT --epochs 5 --test-dataset-version testset_01 --test-filelist ../test_files.txt --batch-size 256  
--print-status 1
```

```
0 > ROCCurveTask(debug=False, terminal_plot=False, config=HLT_run3, verbose=False, seed=123456, dataset_version=trainset_01, training_precision=float32, test_dataset_version=testset_01, testing_precision=float32, training_version=hackathon_tutorial_01, epochs=5, model_name=DeepJetHLT, n_threads=4, batch_size=256, learning_rate=0.001, optimizer=AdamW, loss_function=CrossEntropyLoss, betas=(0.95, 0.999), eps=1e-06, lr_scheduler=epoch_linear_decay, lr_decay_factor=0.01, mixed_precision=False, use_torch_compile=False, torch_compile_mode=default, attack=nominal, attack_magnitude=0.0, attack_iterations=1, attack_individual_factors=False, attack_reduce=True, attack_restrict_impact=-1.0, attack_overshoot=0.02, attack_uncertainty=1.0, test_attack_restrict_impact=-1.0, test_attack_overshoot=0.02, test_attack_uncertainty=1.0)  
LocalDirectoryTarget(fs=local_fs, path=/eos/user/u/uwillems/BTV/b-hive/task_artifacts/ROCCurveTask/hlt_run3/trainset_01/testset_01/hackathon_tutorial_01/DeepJetHLT/epochs_5/nominal/test_attack_nominal)  
absent
```

```
—1 > TrainingTask(debug=False, terminal_plot=False, config=HLT_run3, verbose=False, seed=123456, dataset_version=trainset_01, training_precision=float32, training_version=hackathon_tutorial_01, epochs=5, model_name=DeepJetHLT, n_threads=4, batch_size=256, learning_rate=0.001, optimizer=AdamW, loss_function=CrossEntropyLoss, betas=(0.95, 0.999), eps=1e-06, lr_scheduler=epoch_linear_decay, lr_decay_factor=0.01, mixed_precision=False, use_torch_compile=False, torch_compile_mode=default, attack=nominal, attack_magnitude=0.0, attack_iterations=1, attack_individual_factors=False, attack_reduce=True, attack_restrict_impact=-1.0, attack_overshoot=0.02, attack_uncertainty=1.0, loss_weighting=False, resume_training=False, resume_epoch=False, extend_training=0, train_val_split=0.85, n_train_files=-1)  
training_metrics: LocalFileTarget(fs=local_fs, path=/eos/user/u/uwillems/BTV/b-hive/task_artifacts/TrainingTask/hlt_run3/trainset_01/hackathon_tutorial_01/DeepJetHLT/epochs_5/nominal/training_metrics.npz)  
existent  
validation_metrics: LocalFileTarget(fs=local_fs, path=/eos/user/u/uwillems/BTV/b-hive/task_artifacts/TrainingTask/hlt_run3/trainset_01/hackathon_tutorial_01/DeepJetHLT/epochs_5/nominal/validation_metrics.npz)  
existent  
model: LocalFileTarget(fs=local_fs, path=/eos/user/u/uwillems/BTV/b-hive/task_artifacts/TrainingTask/hlt_run3/trainset_01/hackathon_tutorial_01/DeepJetHLT/epochs_5/nominal/model_4.pt)  
existent  
best_model: LocalFileTarget(fs=local_fs, path=/eos/user/u/uwillems/BTV/b-hive/task_artifacts/TrainingTask/hlt_run3/trainset_01/hackathon_tutorial_01/DeepJetHLT/epochs_5/nominal/best_model.pt)  
existent
```

```
absent  
histogram: LocalFileTarget(fs=local_fs, path=/eos/user/u/uwillems/BTV/b-hive/task_artifacts/DatasetConstructorTask/hlt_run3/testset_01/histogram.npy)  
absent  
weights: LocalFileTarget(fs=local_fs, path=/eos/user/u/uwillems/BTV/b-hive/task_artifacts/DatasetConstructorTask/hlt_run3/testset_01/weights.json)  
absent
```

Law keeps track of everything

```
$ law run ROC Curve Task --dataset-version trainset_01 --config hlt_run3 --training-version hackathon_tutorial_01
--model-name DeepJetHLT --epochs 5 --test-dataset-version testset_01 --test-filelist ../test_files.txt --batch-size 256
--print-status 1
```

```
1 > InferenceTask(debug=False, terminal_plot=False, config=hlt_run3, verbose=False, seed=123456, dataset_version=trainset_01, training_precision=float32, test_dataset_version=testset_01, testing_precision=float32, training_version=hackathon_tutorial_01, epochs=5, model_name=DeepJetHLT, n_threads=4, batch_size=256, learning_rate=0.001, optimizer=AdamW, loss_function=CrossEntropyLoss, betas=0.95,0.999, eps=1e-06, lr_scheduler=epoch_lin_decay, lr_decay_factor=0.01, mixed_precision=False, use_torch_compile=False, torch_compile_mode=default, attack=nominal, attack_magnitude=0.0, attack_iterations=1, attack_individual_factors=False, attack_reduce=True, attack_restrict_impact=-1.0, attack_overshoot=0.02, attack_uncertainty=1.0, test_attack=nominal, test_attack_magnitude=0.0, test_attack_iterations=1, test_attack_individual_factors=False, test_attack_reduce=True, test_attack_restrict_impact=-1.0, test_attack_overshoot=0.02, test_attack_uncertainty=1.0)
  prediction: LocalFileTarget(fs=local_fs, path=/eos/user/u/uwillems/BTV/b-hive/task_artifacts/InferenceTask/hlt_run3/trainset_01/testset_01/hackathon_tutorial_01/DeepJetHLT/epochs_5/nominal/test_attack_nominal/prediction.npy)
    absent
  process: LocalFileTarget(fs=local_fs, path=/eos/user/u/uwillems/BTV/b-hive/task_artifacts/InferenceTask/hlt_run3/trainset_01/testset_01/hackathon_tutorial_01/DeepJetHLT/epochs_5/nominal/test_attack_nominal/process.npy)
    absent
  truth: LocalFileTarget(fs=local_fs, path=/eos/user/u/uwillems/BTV/b-hive/task_artifacts/InferenceTask/hlt_run3/trainset_01/testset_01/hackathon_tutorial_01/DeepJetHLT/epochs_5/nominal/test_attack_nominal/truth.npy)
    absent
  kinematics: LocalFileTarget(fs=local_fs, path=/eos/user/u/uwillems/BTV/b-hive/task_artifacts/InferenceTask/hlt_run3/trainset_01/testset_01/hackathon_tutorial_01/DeepJetHLT/epochs_5/nominal/test_attack_nominal/kinematics.npy)
    absent
  inference_time: LocalFileTarget(fs=local_fs, path=/eos/user/u/uwillems/BTV/b-hive/task_artifacts/InferenceTask/hlt_run3/trainset_01/testset_01/hackathon_tutorial_01/DeepJetHLT/epochs_5/nominal/test_attack_nominal/inference_time.npy)
    absent
```

```
1 > DatasetConstructorTask(debug=False, terminal_plot=False, config=hlt_run3, verbose=False, seed=123456, dataset_version=testset_01, training_precision=float32, chunk_size=100000)
  file_list: LocalFileTarget(fs=local_fs, path=/eos/user/u/uwillems/BTV/b-hive/task_artifacts/DatasetConstructorTask/hlt_run3/testset_01/processed_files.txt)
  absent
  histogram: LocalFileTarget(fs=local_fs, path=/eos/user/u/uwillems/BTV/b-hive/task_artifacts/DatasetConstructorTask/hlt_run3/testset_01/histogram.npy)
  absent
  weights: LocalFileTarget(fs=local_fs, path=/eos/user/u/uwillems/BTV/b-hive/task_artifacts/DatasetConstructorTask/hlt_run3/testset_01/weights.json)
  absent
```