Brookhaven National Laboratory

U.S. DEPARTMENT of ENERGY

HEP Pilot: Treasure

Slides from:
Viviana Cavaliere (BNL),
Jeff Krupa (SLAC),
Beojan Stanislaus (LBNL),
Paolo Calafiura (LBNL)

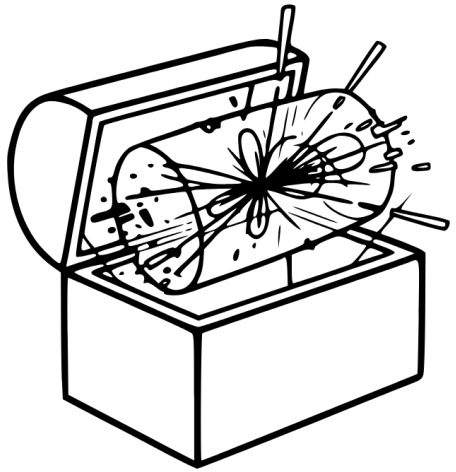
presented by
Haider Abidi (BNL)

@BrookhavenLab

Blueprint Workshop: Towards a National-Scale AI Collaboration in HEP 5/18/2026

What is TREASURE?

- *Tokenized Representations for Energy-frontier AI Searches via Understanding and REasoning*
  - DOE HEP American Science Cloud (AmSC) Intelligent Data Activities Pilot



Contacts:

Viviana Cavaliere (BNL, PI)

Paolo Calafiura (LBNL)

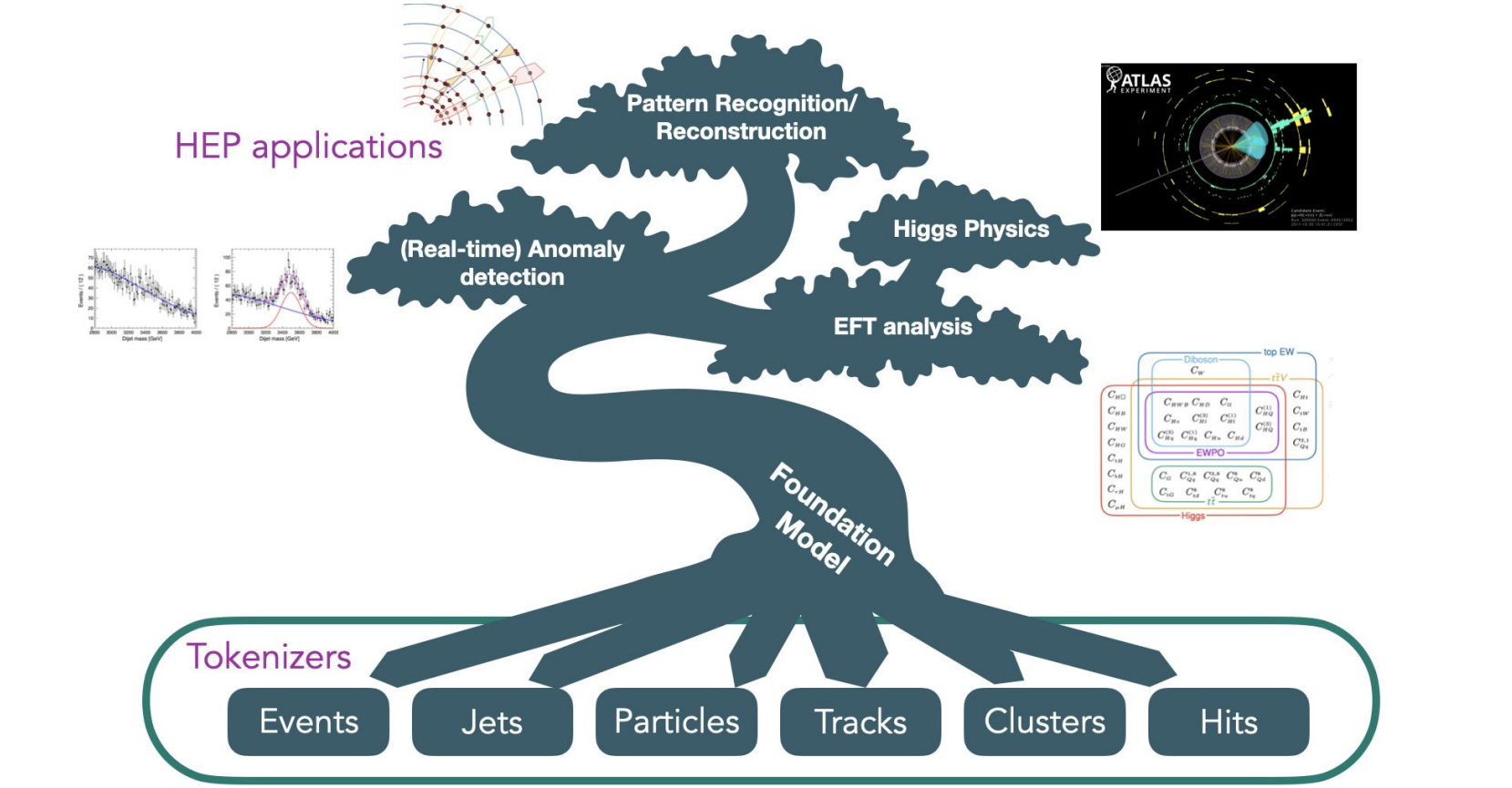
Walter Hopkins (ANL)

Michael Kagan (SLAC)

Kevin Pedro (FNAL)

Brookhaven National Laboratory · Fermilab · Berkeley Lab · SLAC National Accelerator Laboratory · Argonne National Laboratory

What is Treasure?



- *Treasure prepares **collider data** so it can be effectively used by modern AI methods*
- It transforms heterogeneous, experiment-specific data into standardized, **tokenized**, AI-ready representations.
- **Builds Foundation Models** to learn across experiments, detectors, and eras of data.

Goal: unlock new discovery potential from both current and legacy high-energy physics datasets.

Why This Matters

Collider experiments generate enormous data volumes in incompatible formats.

This fragmentation limits cross-experiment analysis and reuse of legacy data.

Multi-Experiment Training

Learn detector-independent physics features.

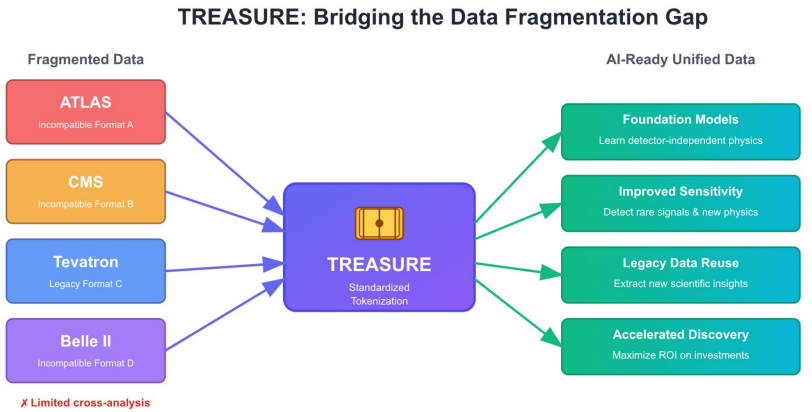
- Improved sensitivity to rare signals.
- More data!

Legacy Data Revaluation

Legacy datasets underutilized due to technical obstacles.

⇒ *Extract new information and extend scientific impact.*

TREASURE bridges this gap, accelerating discovery and maximizing return on investments.



Input Datasets Include

aspen
jetclass
jetset
atlas
cms
belle 2
...

Fermilab BERKELEY LAB SLAC NATIONAL ACCELERATOR LABORATORY Argonne NATIONAL LABORATORY

the TREASURE Pipeline

https://gitlab.cern.ch/vcavalie/bnl-treasure

Experiment Data (ROOT)
Object-level Data
(e.g., ATLAS PHYSLITE)

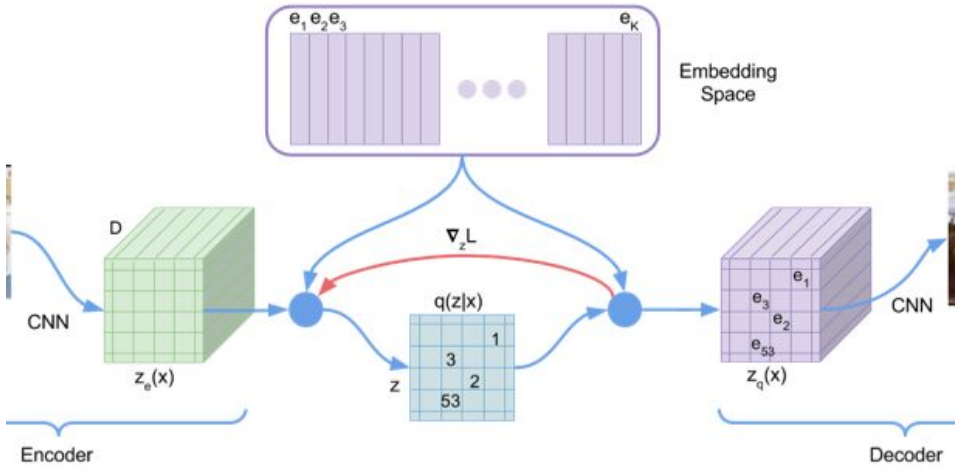
AI-READY (HDF5, PARQUET)
per-event arrays,
masked, padded

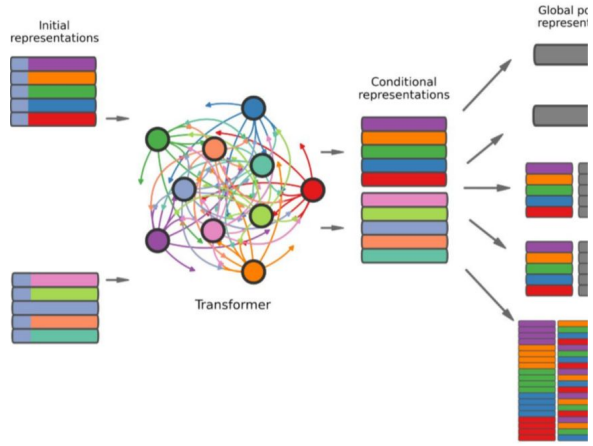
TOKENIZER
VQ-VAE / FSQ /
PQ / RVQ per type

PARQUET
Integer token
sequences

TRANSFORMER
Pretrain → fine-tune
→ transfer

```
ATLAS DAOD_PHYSLITE to HDF5 Converter  (cross-experiment edition)
================================================================

Produces two layers in every HDF5 file:

  /common/    — variables defined identically across ATLAS and CMS.
                A CMS NanoAOD converter should fill the same keys.
                All energies/momenta in GeV, angles in radians.

  /atlas/     — ATLAS-specific variables kept for ATLAS-only studies.
                Do NOT use these in cross-experiment training without
                explicit experiment conditioning.

  /metadata   — HDF5 group attributes (no datasets).  Covers:
                  experiment, format_version, input_file, n_events,
                  git_hash (if available), pt_cuts, max_objects,
                  common_iso_cone, common_iso_pt_floor_gev,
                  common_iso_z0sintheta_cut_mm

Schema summary
--------------
common/electrons : pt, eta, phi, charge, trk_iso03, mask, n
common/muons     : pt, eta, phi, charge, trk_iso03, mask, n
common/taus      : pt, eta, phi, charge, is_1prong, mask, n
common/photons   : pt, eta, phi, trk_iso03, mask, n
common/jets      : pt, eta, phi, mass, n_trk, mask, n
common/tracks    : pt, eta, phi, d0, z0, mask, n
common/met       : pt, phi, sumet          (scalar per event)
common/event     : pvx, pvy, pvz, mu, experiment_id (0=ATLAS, 1=CMS),
                   is_simulation (1=MC, 0=data)

  mask  — boolean array shape (n_events, max_objects).  True = real object,
          False = zero-padding.  Required because events have variable object
```





Status: Current Workstreams

Common LHC Data Format Specification

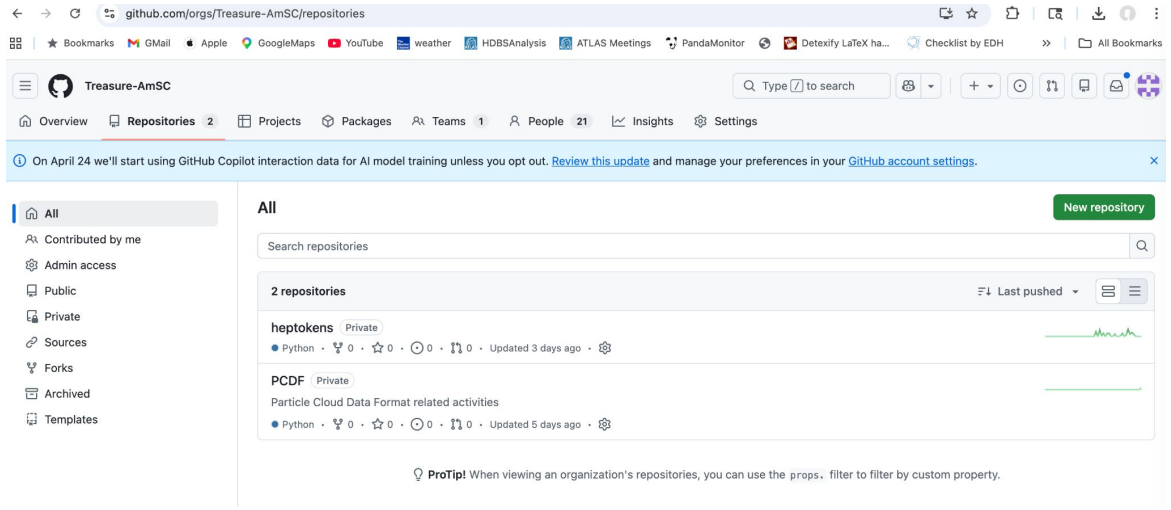
- v0 Schema and metadata for common format
- Validated via example dataset & experiment evaluation

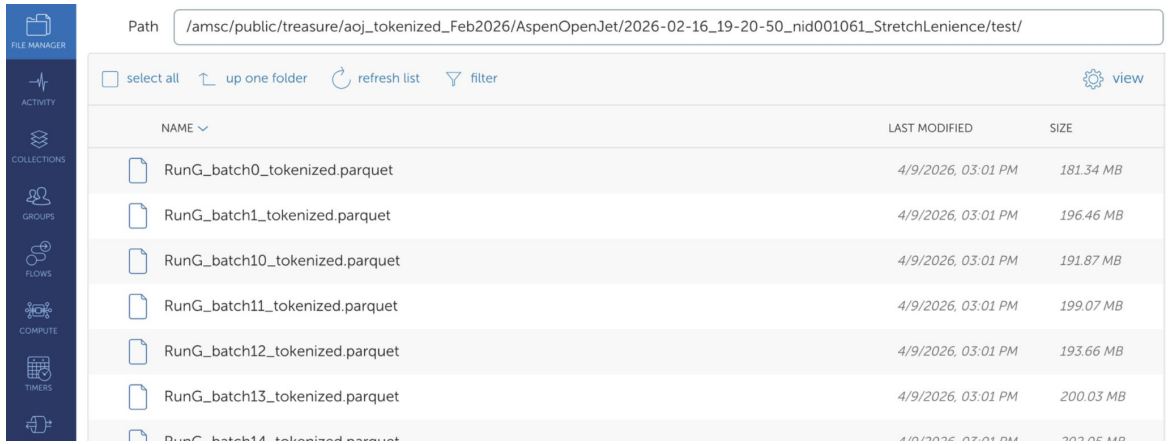
Tokenized Jet Datasets

- High-quality jet representations
- VQ-VAE model-based tokenization

Infrastructure & Hosting

- Partnering with AmSC Infrastructure Partners across labs
  - Fermi Data Platform tokenized dataset





PCDF: The Particle Cloud Data Format

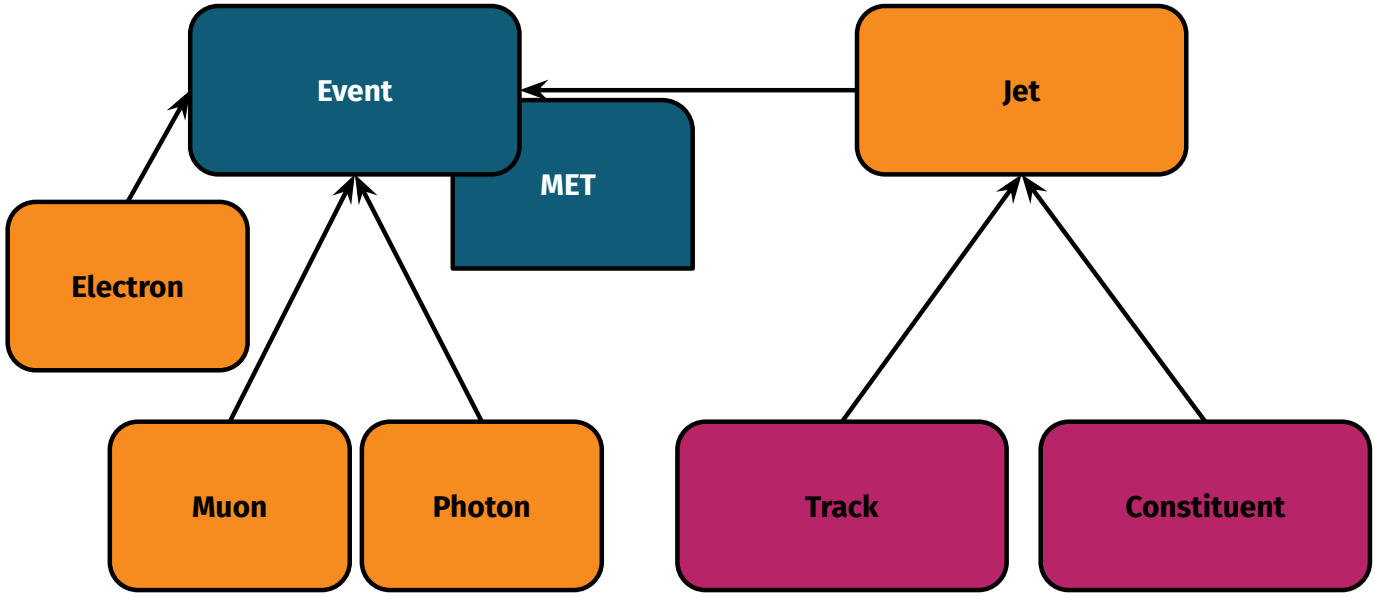
Beojan Stanislaus

Flat Table Architecture

Use of indices as cross-references allows tables to be **flat**.

Enables storage as **Parquet** files with rich metadata (column names, statistics).

Modern software (e.g., **Polars**) can group by indices to generate n-dimensional **HDF5** tables efficiently.



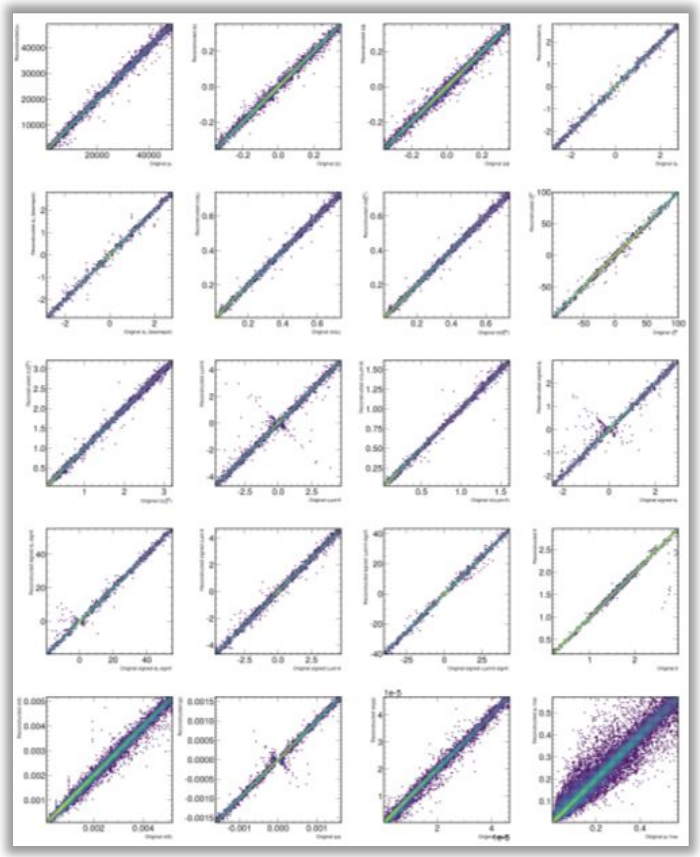
Next step: develop **web API** to automate pre-determined **transformations**.

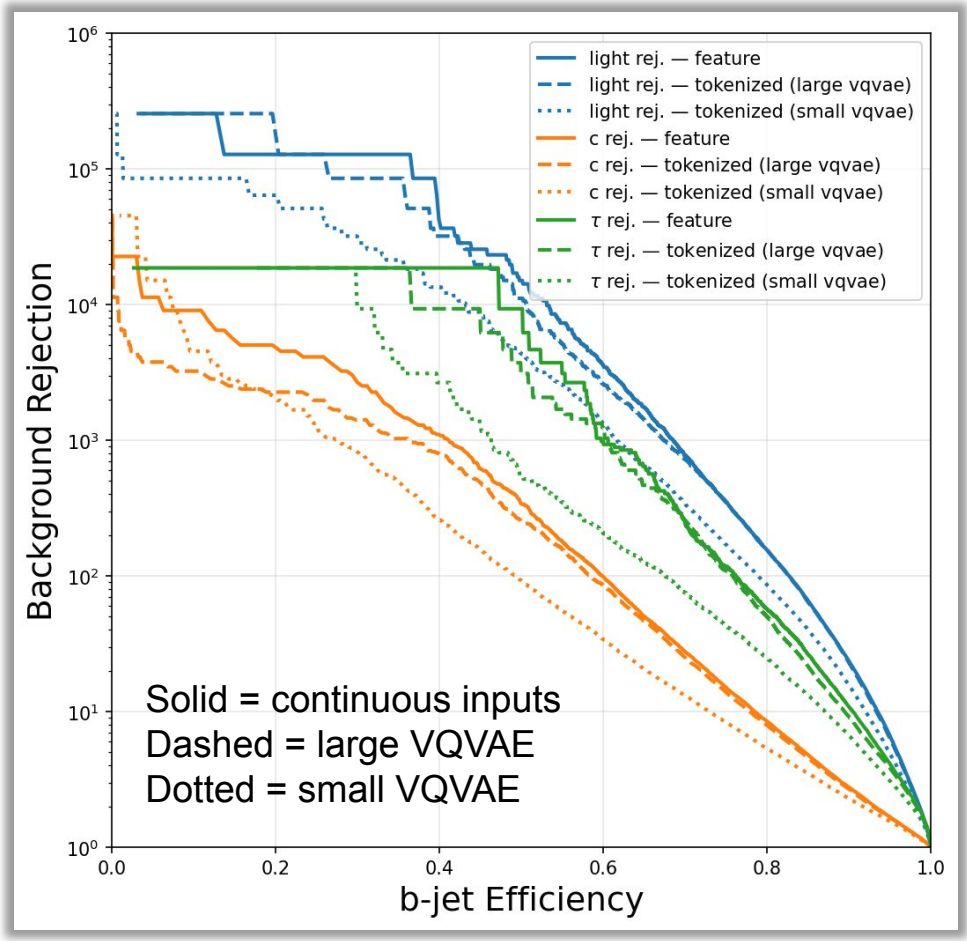
heptokens: A library for learning jet representations

Jeff Krupa, Michael Kagan

Quick start

```
from heptokens.data import
  SingleFileMapModule

from
heptokens.models.vq_vae
import
  LitVqVae

dm = SingleFileMapModule(
  data_path="mc-ttbar.h5",
  num_csts=40)

model = LitVqVae(
  codebook_size=512,
  num_quantizers=3)

trainer.fit(model, dm)
```

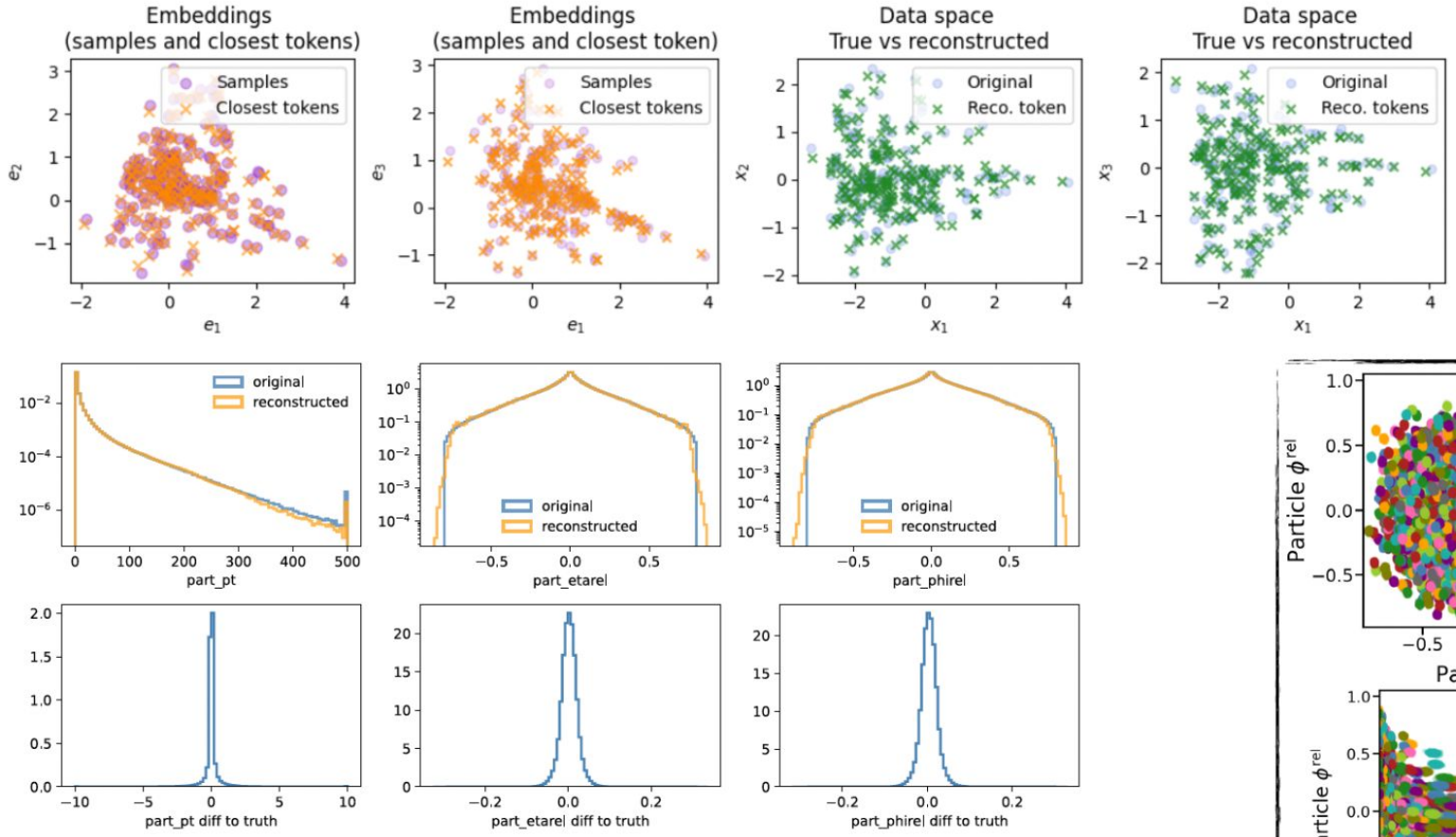




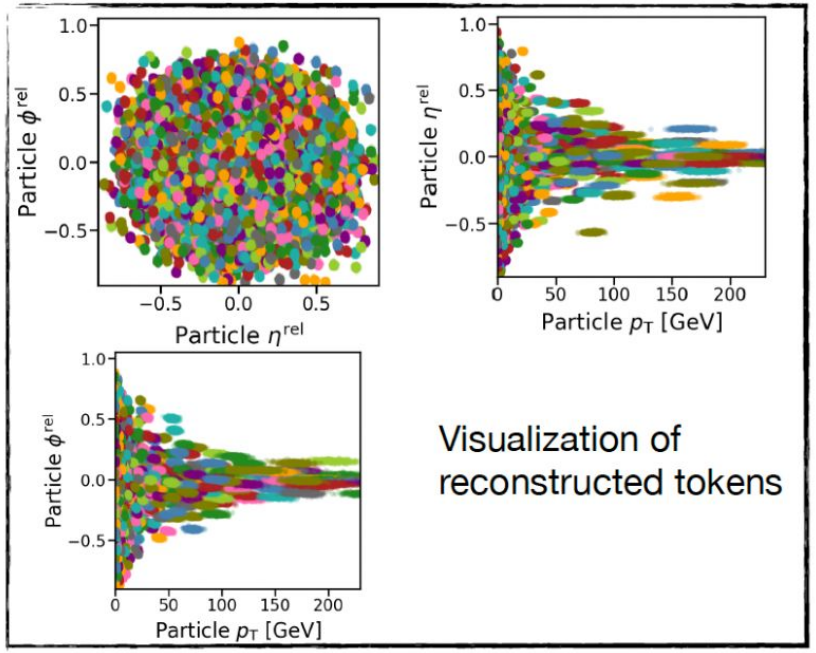
Tokenization of CMS jet open datasets

Aspen open jet dataset tokenization

(https://www.fdr.uni-hamburg.de/record/16505)



Codebook size 8192



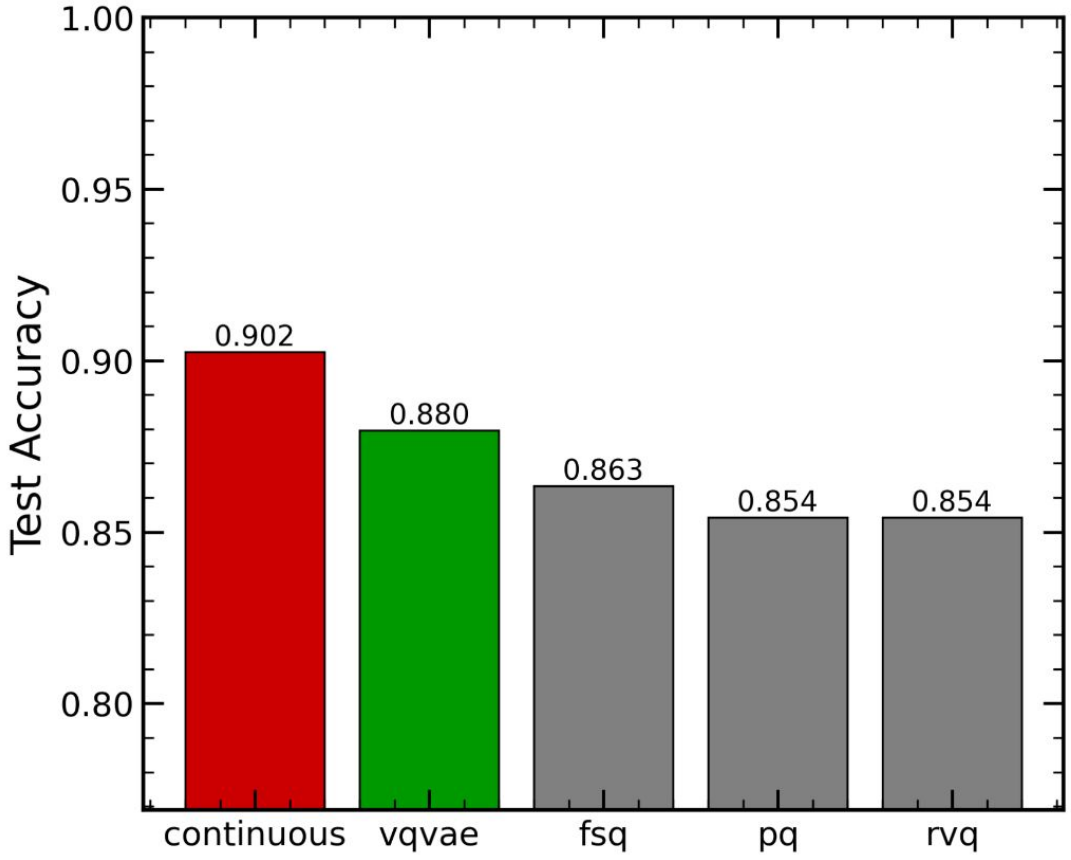
Visualization of reconstructed tokens

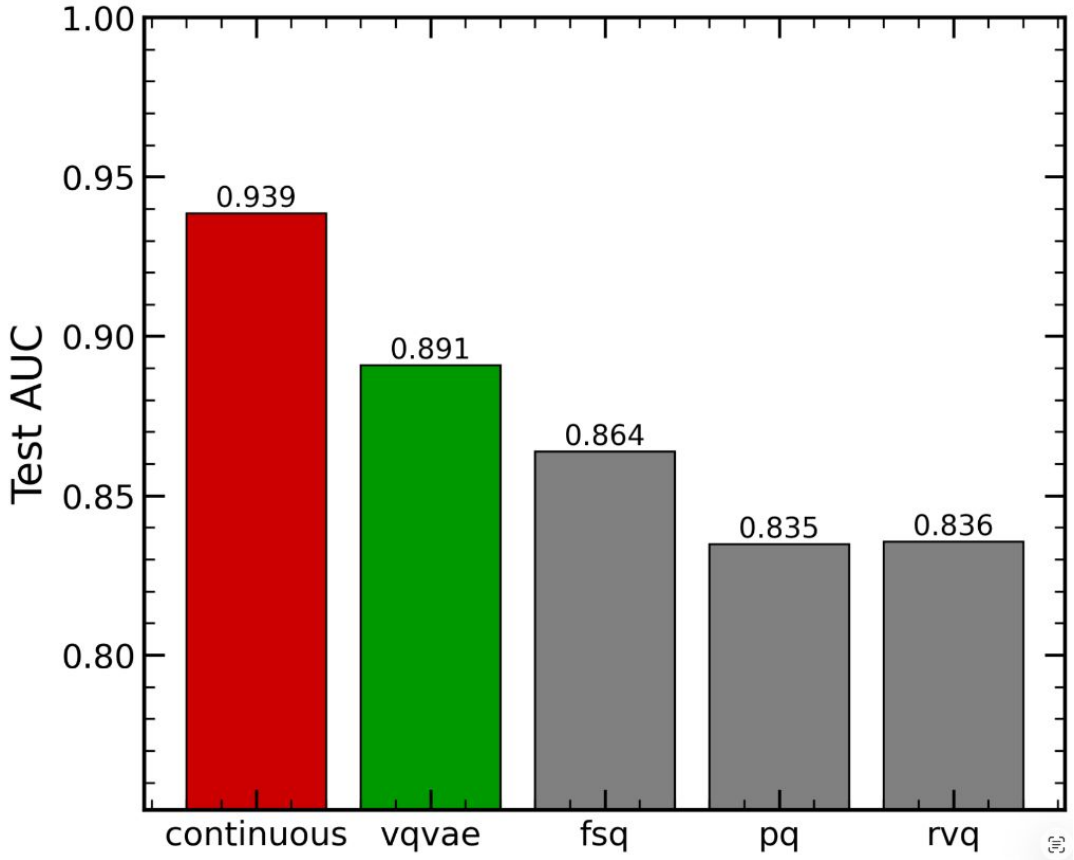
Comparison of original and reconstructed particle pt, η^{rel}, ϕ^{rel} inside jets

$\eta^{rel} = \eta^{particle} - \eta^{jet}$, $\phi^{rel} = \phi^{particle} - \phi^{jet}$

Benchmarking Event Tokenization

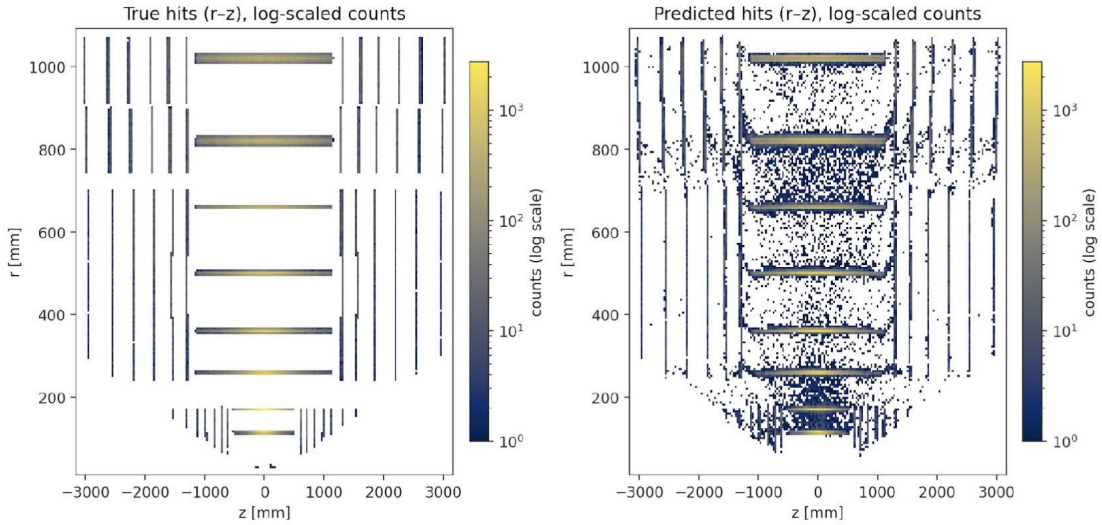
We compared five tokenization strategies on the same downstream task: distinguishing Higgs → ZZ → 4 leptons from Standard-Model background. AUC of the transformer classifier quantifies how much physics each representation preserves.*



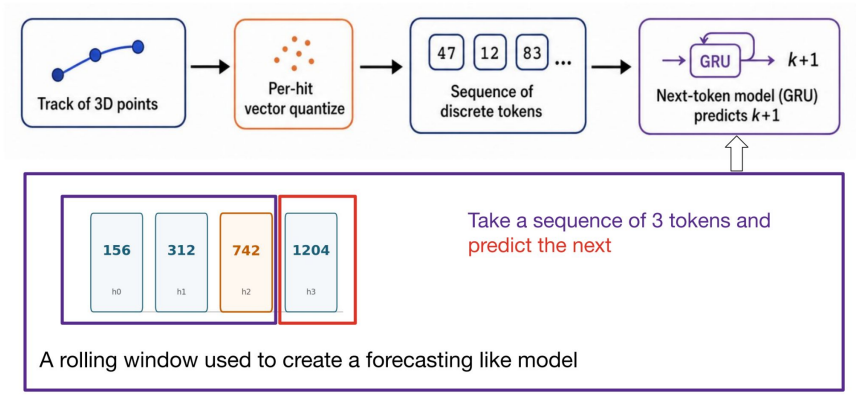


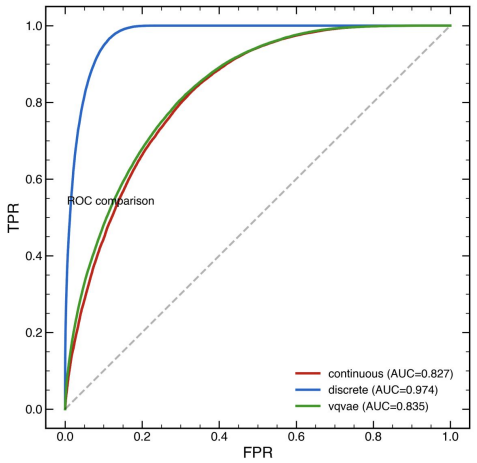
Looking forward: Low-Level & Other Experiments

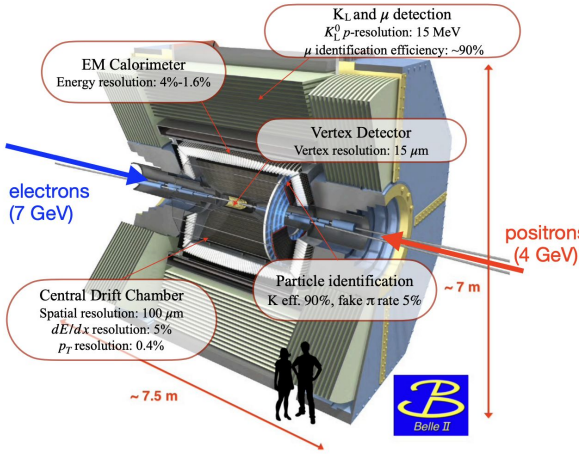
First tries at track reconstruction and token non-LHC data have started



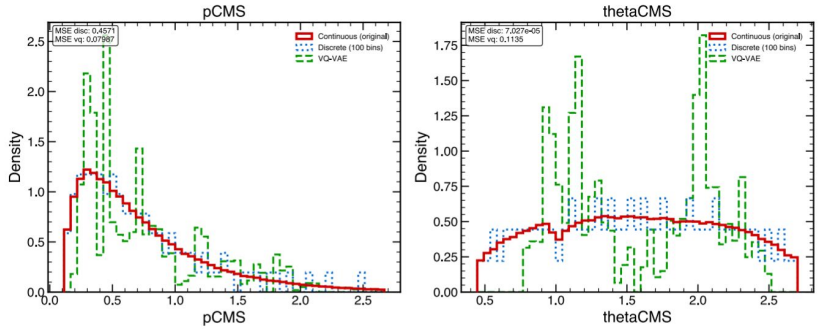
Punit Sharma







Riccardo Manfredi

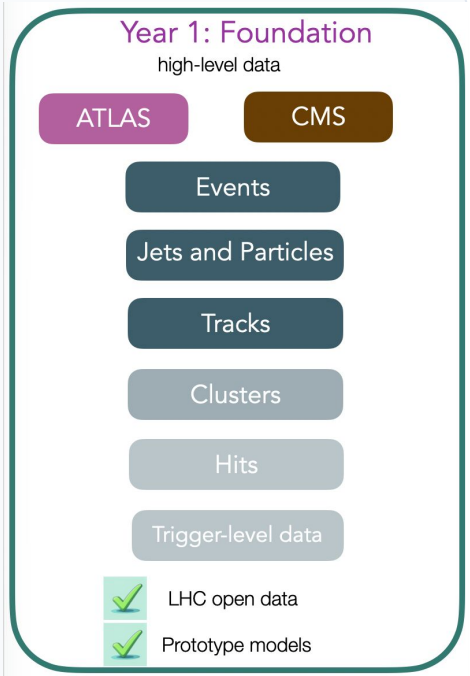


Project Timeline

Year 1

Foundations

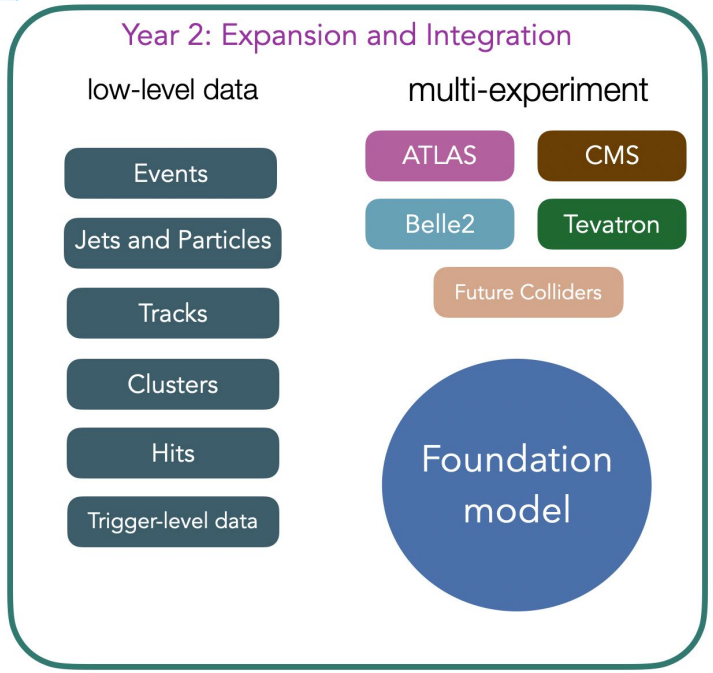
- AI-ready datasets from LHC Open Data
- Common data models, metadata, and tokenization schemes
- Assess AI-readiness and quantify tokenization impact
- **Demonstrate cross-experiment learning with prototype AI models**



Year 2

Expansion and Integration

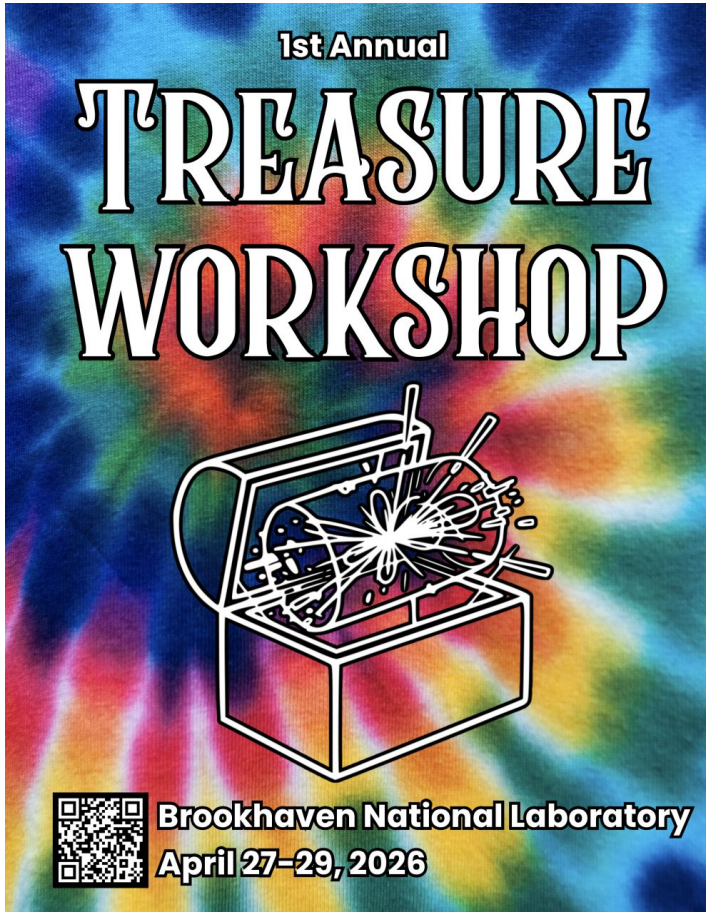
- **Detector data:** hits, clusters, streaming compression
- **Beyond the LHC:** Tevatron (ppbar), Belle II (e+e-), future experiments
- **Scale up cross-experiment foundation model training**



Brookhaven National Laboratory · Fermilab · Berkeley Lab · SLAC National Accelerator Laboratory · Argonne National Laboratory

What's next & How to get involved?

- Held workshop a few week ago to brainstorm the next steps
  - Establish Treasure workflow on AmSC - use the opportunity to setup all the technical pieces
  - Provide CERN open data and tokenized through ModCon
  - Create a context-aware event-level tokenization with HepTokens
  - Focus on low-level information tokenization - work with experiments to publish low-level information
- Have provided letters for multiple Genesis Phase 1 proposals
  - Will provide support & help as needed
  - Tokenized data, libraries, workflow are available
- **Happy to collaborate!** - reach out to Viviana Cavaliere (viviana.cavaliere@cern.ch) or Haider Abidi (sabidi@cern.ch)
  - 2nd workshop is planned for later year
  - Depending on how many genesis proposals/interested people are there, setup a common meeting for collaboration
- **Aim to bring together Phase 1 + collaborators for a combined submission for Phase 2 in december**



Conclusion: TREASURE Impact

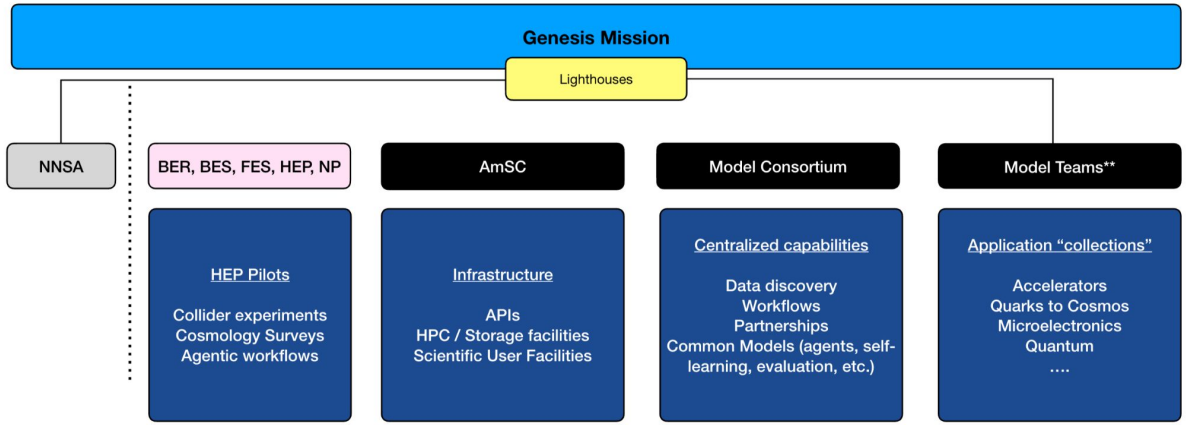
- Maximizes return on investment in past, current, and future experiments.
  - Creates reusable data infrastructure
- Accelerates time-to-discovery through AI-ready data.
- Working towards establishing the workflow on common resources for wider adoption
- Long-term outcomes:
  - Community-adopted standards for AI-ready scientific data.
  - Scalable foundation models for collider physics.
  - Blueprint for applying similar approaches in other scientific domains.
- Happy to collaborate! Please reach out to us

Brookhaven National Laboratory Fermilab BERKELEY LAB SLAC NATIONAL ACCELERATOR LABORATORY Argonne NATIONAL LABORATORY

Backup

Integration with Genesis mission

- Where are we in the bigger picture?



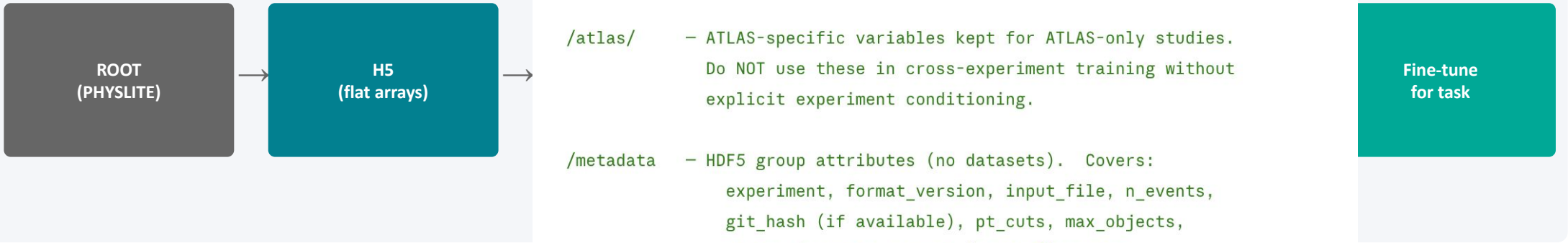
from Nhan Tran (FNAL)

- Currently Treasure is one of the "data-providers"
- But… **a big component of our proposal is the development of models to enable AI-discovery**
- **We think we are in the best position to contribute to Model Development**
- **Need and want a strong footprint in the Model Teams**
  - Domain science expertise critical for Model development
  - Our team has experience in scientific foundation models and scaling them up, and we can deliver real impact for science
  - Collaborators in several Gemini Phase 1&2 proposals

Brookhaven National Laboratory · Fermilab · Berkeley Lab · SLAC National Accelerator Laboratory · Argonne National Laboratory

Step 1

The Pipeline



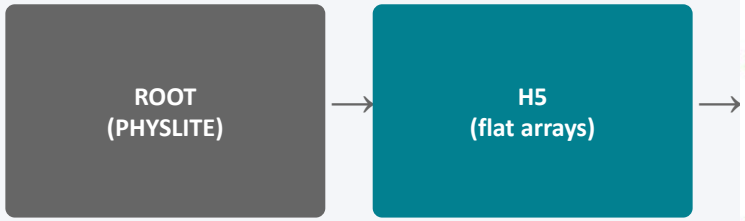
atlas_to_h5_converter.py

```
ATLAS DAOD_PHYSLITE to HDF5 Converter  (cross-experiment edition)
================================================================

Produces two layers in every HDF5 file:

  /common/    – variables defined identically across ATLAS and CMS.
                A CMS NanoAOD converter should fill the same keys.
                All energies/momenta in GeV, angles in radians.

  /atlas/     – ATLAS-specific variables kept for ATLAS-only studies.
                Do NOT use these in cross-experiment training without
                explicit experiment conditioning.

  /metadata   – HDF5 group attributes (no datasets).  Covers:
                  experiment, format_version, input_file, n_events,
                  git_hash (if available), pt_cuts, max_objects,
                  common_iso_cone, common_iso_pt_floor_gev,
                  common_iso_z0sintheta_cut_mm

Schema summary
--------------
common/electrons : pt, eta, phi, charge, trk_iso03, mask, n
common/muons     : pt, eta, phi, charge, trk_iso03, mask, n
common/taus      : pt, eta, phi, charge, is_1prong, mask, n
common/photons   : pt, eta, phi, trk_iso03, mask, n
common/jets      : pt, eta, phi, mass, n_trk, mask, n
common/tracks    : pt, eta, phi, d0, z0, mask, n
common/met       : pt, phi, sumet          (scalar per event)
common/event     : pvx, pvy, pvz, mu, experiment_id (0=ATLAS, 1=CMS),
                   is_simulation (1=MC, 0=data)

  mask  – boolean array shape (n_events, max_objects).  True = real object,
          False = zero-padding.  Required because events have variable object
```

Brookhaven National Laboratory

Step 1

The Pipeline



atlas_to_h5_converter.py

```
atlas/electrons  : LHLoose, LHMedium, LHTight, topoetcone20, ptvarcone30, mass
atlas/muons      : quality, muonType, passIDCuts, passPresel,
                   topoetcone20, ptvarcone30
atlas/taus       : NNDecayMode, RNNJetScore,
                   RNNEleScore, EleRNNLoose/Medium/Tight_v1
atlas/photons    : isLoose, isTight, isCleaning, author,  (no isMedium in PHYSLITE)
                   topoetcone20, topoetcone40, ptcone20
atlas/jets       : DL1d_pb, DL1d_pc, DL1d_pu,
                   GN2_pb,  GN2_pc,  GN2_pu,
                   QG_nTracks, QG_tracksWidth, QG_tracksC1
atlas/tracks     : qOverP, chiSquared, nDoF
atlas/event      : event_number, run_number, mcChannelNumber

Isolation definition (common/*/trk_iso03)
-----------------------------------------
  I_trk = sum_pT(tracks, ΔR < 0.3, pT > ISO_PT_FLOOR,
                 |z0·sinθ| < ISO_Z0ST_CUT w.r.t. PV)
        / pT(object)

  where ISO_PT_FLOOR  = 0.5 GeV  (500 MeV)
        ISO_Z0ST_CUT  = 3.0 mm

  CMS equivalent: computed from PackedCandidates (charged, PV-associated)
  with the same cone and pT floor.  Distributions will differ due to
  detector geometry; use experiment_id as a conditioning token.
"""
```

Brookhaven National Laboratory

How do you do event level tokenization?

- How to define event level tokens?
  - Each particle becomes a token via a shared linear projection:
  - Particle i input vector:
  - [pT, η, ϕ, E, mass, charge, isElectron, isMuon, isPhoton, isJet, ...]
  - ↓ shared linear layer (same weights for all particles)
  - d-dimensional embedding vector
- Particle type is encoded as a one-hot feature (isElectron, isMuon, ...) appended to the feature vector. This is the approach used by ParticleTransformer and similar models that work on **constituent-level data** where all particles have the same features (pT, η, ϕ, E).

How do you do event level tokenization?

Tokenizer (VQ-VAE/FSQ/ PQ/RVQ)

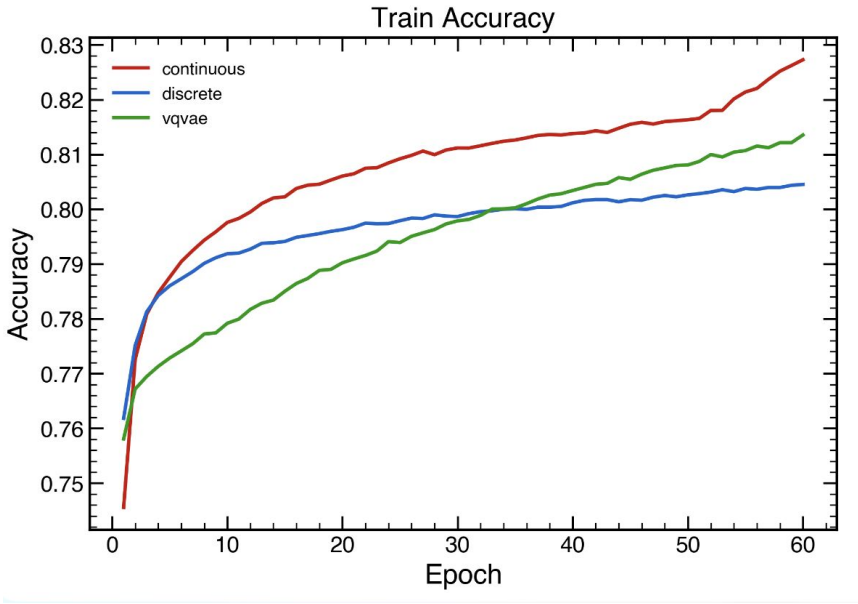
Or Each object type has its **own separate VQ-VAE** (or FSQ/PQ/RVQ) with its own codebook, because each object type has completely different features:

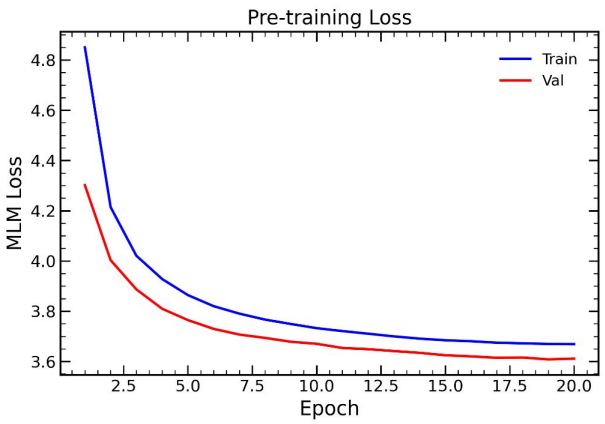
- Electron: pT, η, ϕ,
- Jet: pT, η, ϕ, mass, nTrkPt500, QG vars, DL1d, GN2 (14 features)
- Track: qOverP, η, ϕ, d0, z0, pT, χ^2, nDoF (8 features)
- **How we tell the transformer which type each token is:** We use token_type_ids — a separate embedding table where electron=4, muon=5, jet=6, etc. This is added to each token's representation:
  - token_repr = token_embedding(token_id) + type_embedding(type_id) + position_embedding(pos)
- So the transformer knows "this token is a jet" from the type embedding, not from a one-hot feature.
- Each object is tokenized independently. The VQ-VAE for electrons doesn't know about the jets in the same event. So , at tokenization time, we don't explicitly compute Δϕ(electron, jet) or ΔR or invariant masses between objects.
- **But the transformer sees all objects together:** After tokenization, the event sequence looks like:
  - [CLS] [event_ctx] [e_1] [e_2] [SEP] [μ_1] [SEP] [j_1] [j_2] [j_3] [SEP] [MET]
- Every token attends to every other token through self-attention. When the transformer computes attention between the electron token and the jet token, it has access to both objects' representations — which encode their ϕ values. The attention mechanism can learn that "electron$_1$ has ϕ=0.8 and jet$_1$ has ϕ=0.9, so they're close together" is a meaningful pattern.

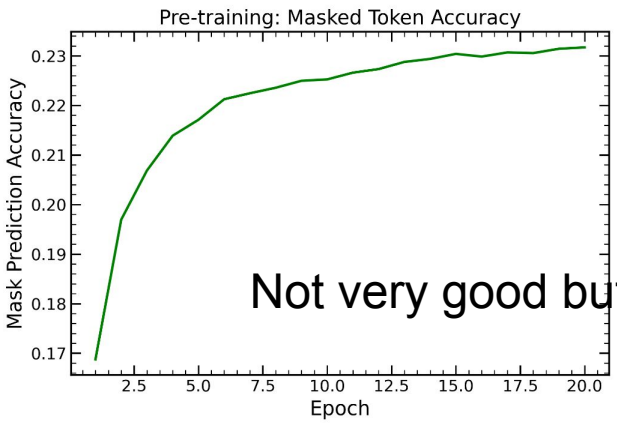
Task 1

$H \rightarrow ZZ \rightarrow 4\ell$

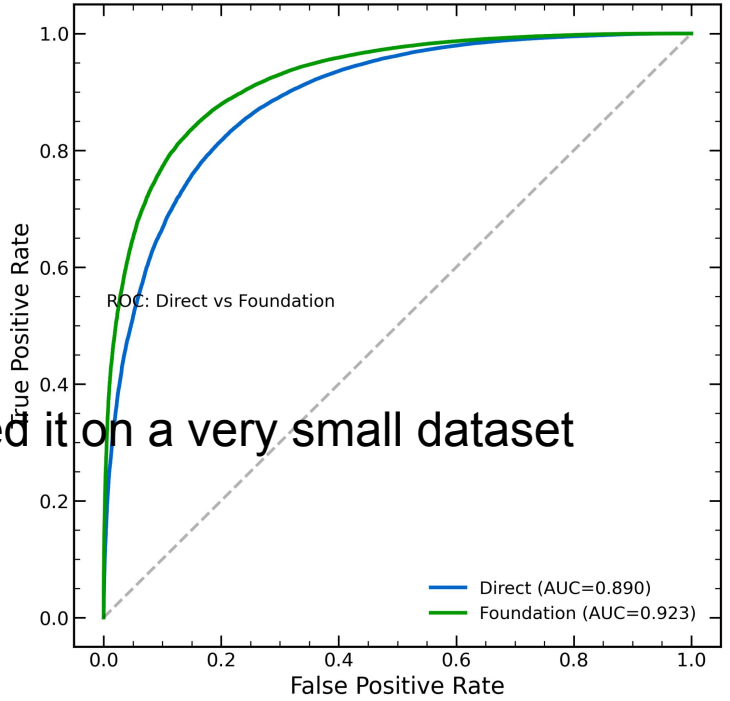
Train classifier to separate Higgs signal from background. Direct comparison: standard classifier vs foundation model.







Not very good but trained it on a very small dataset



Brookhaven National Laboratory

Hierarchical Feature-Group Tokenization

Multiple sub-tokens per object, aggregated by inner transformer

Flat (current)

Each object ⟶ 1 position in the sequence.
All features are projected through a single linear layer.
The transformer must figure out which features are kinematics, which are b-tagging, etc.

Jet ⟶ Linear([pT, η, φ, mass, ..., GN2_pu]) ⟶ hidden dim

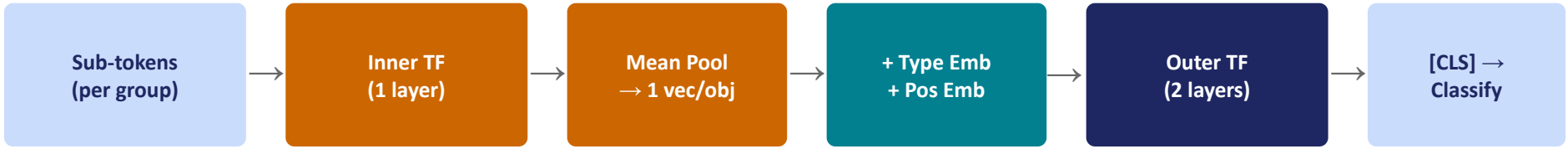
Hierarchical (feature groups)

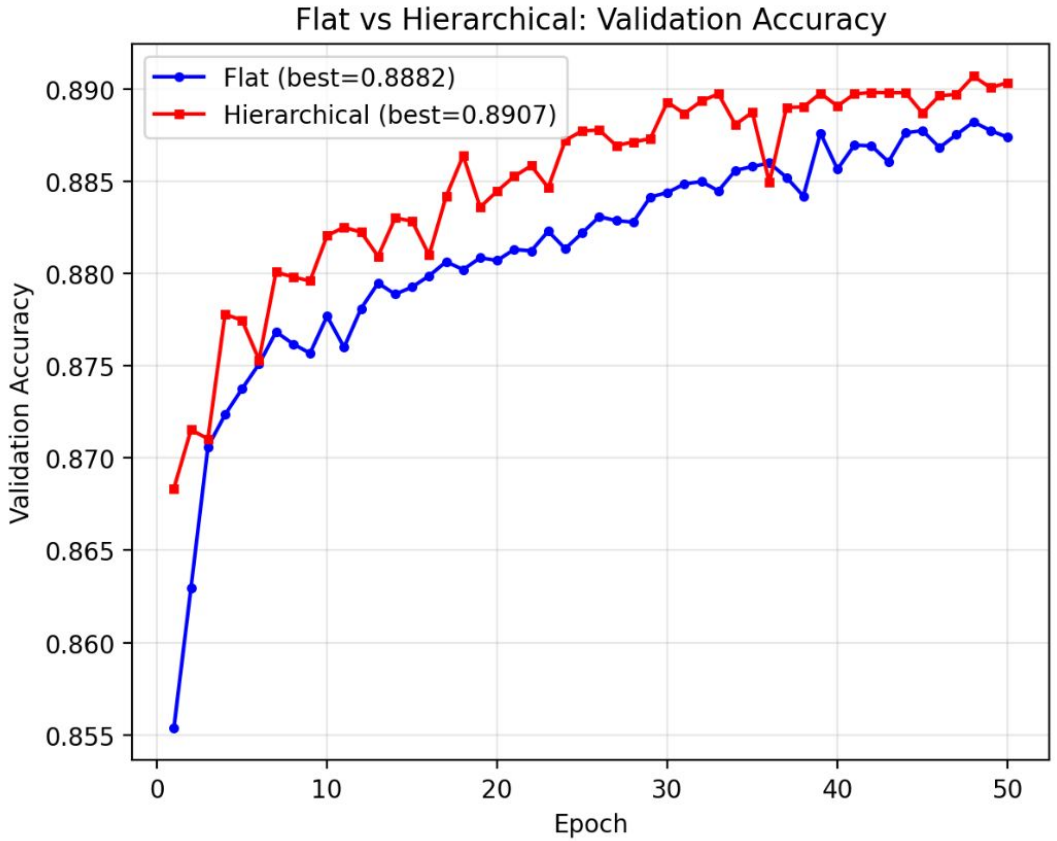
Each object ⟶ 2-3 sub-tokens by semantic group.
Inner transformer aggregates sub-tokens ⟶ 1 vector.
Outer event transformer reads aggregated objects.

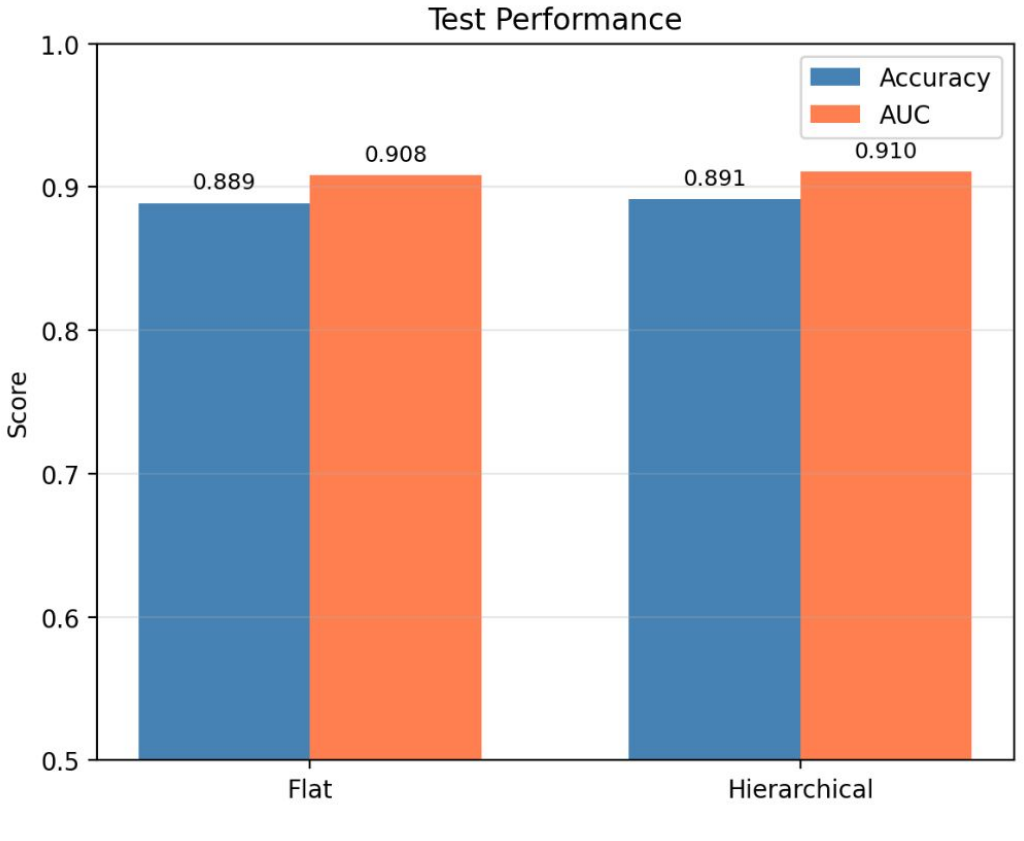
Jet ⟶ [kinematics: pT,η,φ,m] + [substructure: QG]
+ [b-tagging: DL1d, GN2]
⟶ inner attention ⟶ 1 jet vector ⟶ outer transformer

Feature groups per object:

Electron: [kinematics: pT,η,φ] + [ID: LHLoose/Med/Tight] Muon: [kinematics: pT,η,φ] + [ID: quality]
Tau: [kinematics: pT,η,φ] + [ID: nTracks, RNN scores] Photon: [kinematics: pT,η,φ] + [ID: isLoose/Med/Tight]
Jet: [kinematics: pT,η,φ,m] + [substructure: QG vars] + [b-tagging: DL1d+GN2] Track: [kin] + [impact: d0,z0] + [quality: χ^2,nDoF]



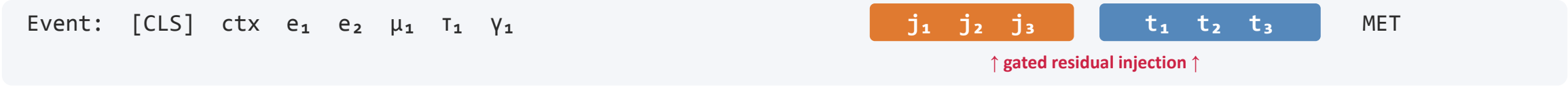




Brookhaven National Laboratory

Constituent + Hit Hierarchical Architecture

Sub-object info enriches jets (constituents) and tracks (hits) via gated residual



Jet Constituents (tracks/clusters inside jet)

Per jet: up to 50 constituent tracks/clusters
Each: pT, η, φ, charge, d0, z0

1. Project features → hidden dim
2. Constituent Transformer (1 layer)
3. Attention pooling → 1 vector per jet
4. Gated residual:
   jet_enriched = jet + σ(W·[jet // const]) × const

Gate learns how much constituent info to add.
When gate≈0: pure object-level. When gate≈1: constituent-driven.

Track Hits (detector measurements along track)

Per track: up to 30 detector hits
Each: η, φ, energy, layer, time

1. Project features → hidden dim
2. Hit Transformer (1 layer)
3. Attention pooling → 1 vector per track
4. Gated residual:
   track_enriched = track + σ(W·[track // hit]) × hit

Same architecture as jet constituents.
Hit patterns reveal particle type (e/μ/π),
track quality, and pile-up rejection.

Auto-detects constituent/hit data in H5. Falls back to object-only if absent. Works on current PHYSLITE data today.

Brookhaven National Laboratory

heptokens

A library for learning jet representations (Jeff Krupa, Michael Kagan (SLAC)

Tokenization model variants

MLP (no-context) · Transformer (in-context) — easily extendable to other models using LightningModule interface

Production ML stack

PyTorch Lightning · Hydra configs · Scalability with Snakemake · WandB logging · Pixi environments

Support for multi-stage pipelines

Ready for deployment into larger systems · Supports two-stage pipeline (decoupled tokenization+classification)

Tokenization performance

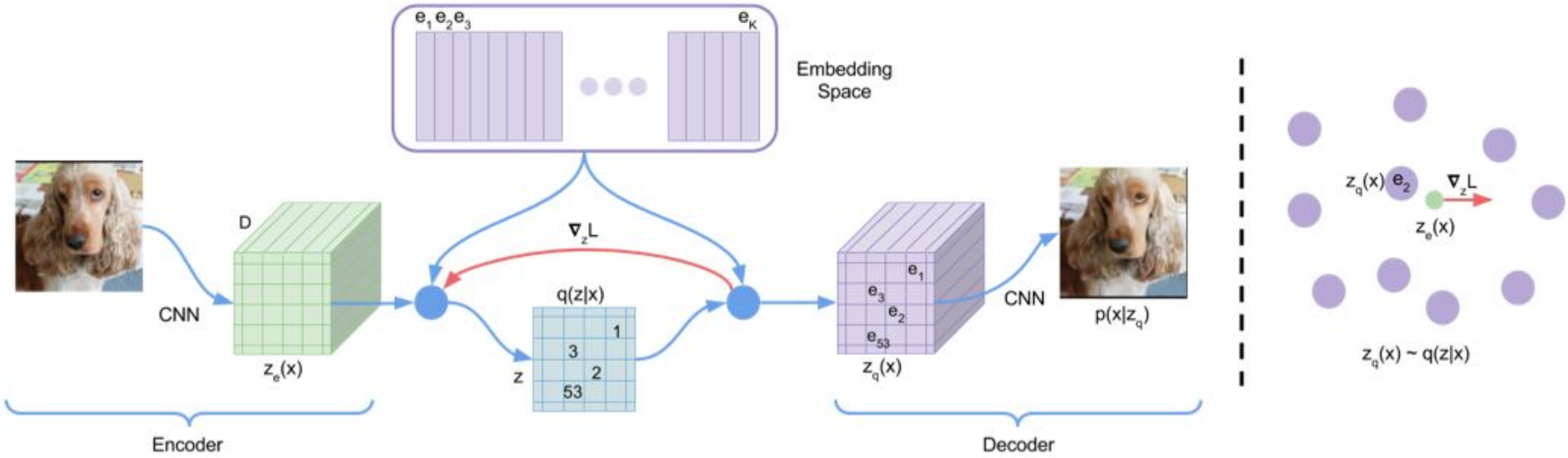
High-fidelity tokenization (<1 % jet resolution), high-performance classification on tokens (competitive with continuous features)

Quick start

```
from heptokens.data import
  SingleFileMapModule

from
heptokens.models.vq_vae
import
  LitVqVae

dm = SingleFileMapModule(
  data_path="mc-ttbar.h5",
  num_csts=40)

model = LitVqVae(
  codebook_size=512,
  num_quantizers=3)

trainer.fit(model, dm)
```

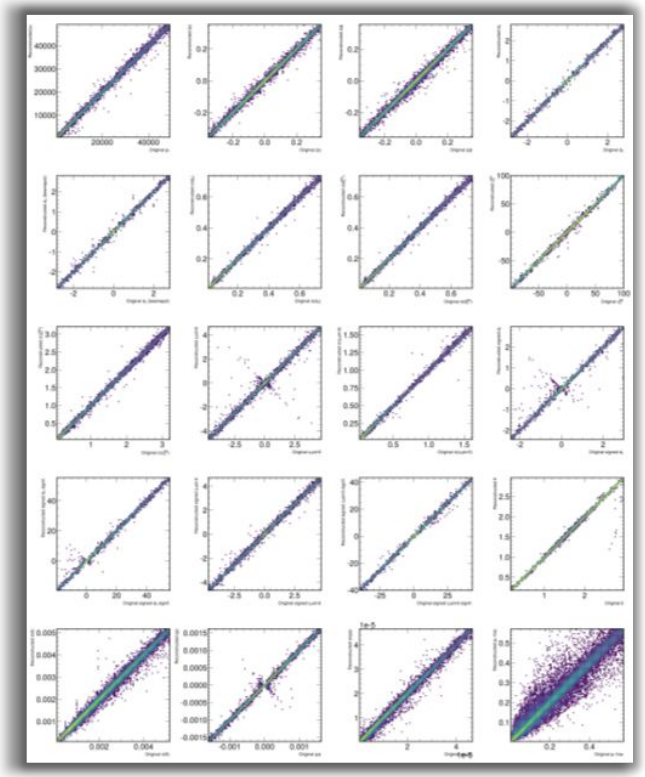
VQ-VAE

- Vector quantized variational autoencoder (VQ-VAE) compresses high-dimensional inputs into a learned codebook
  - Produces structured, compact tokens for use in downstream tasks

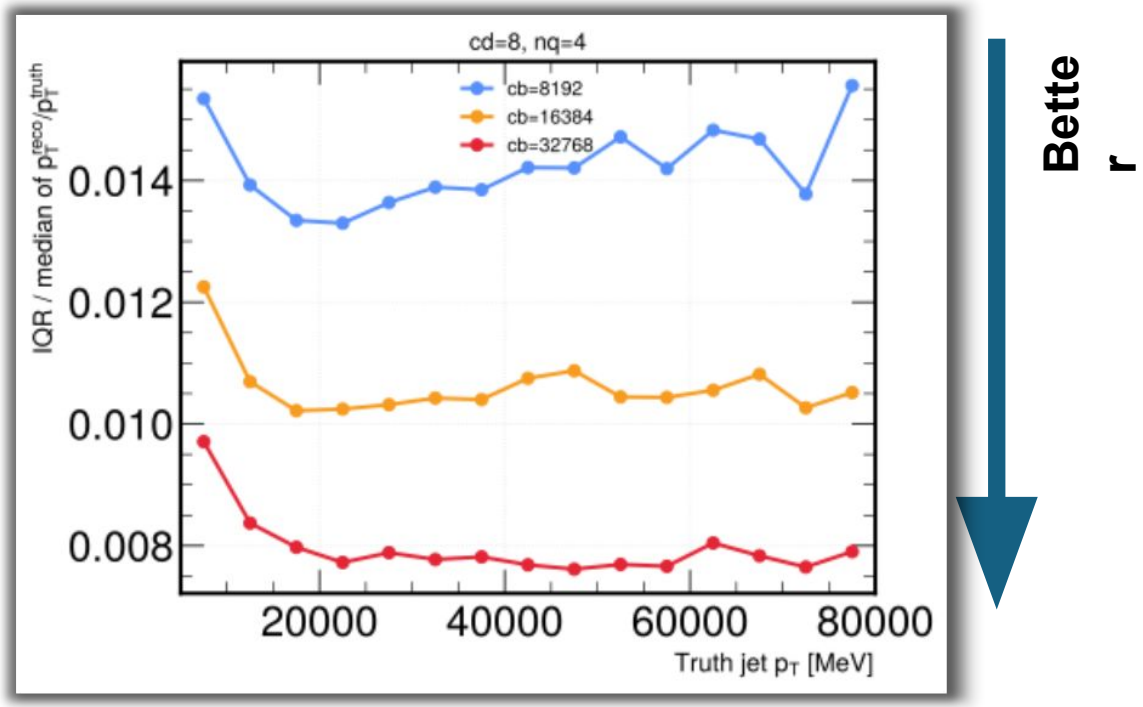


Tokenization

- JetSet = open ATLAS *tt* simulation with 200M jets for flavor tagging
  - 20 track features, event-level features
- Tokenizing tracks with variants of VQ-VAEs



Accurately reconstructs track features

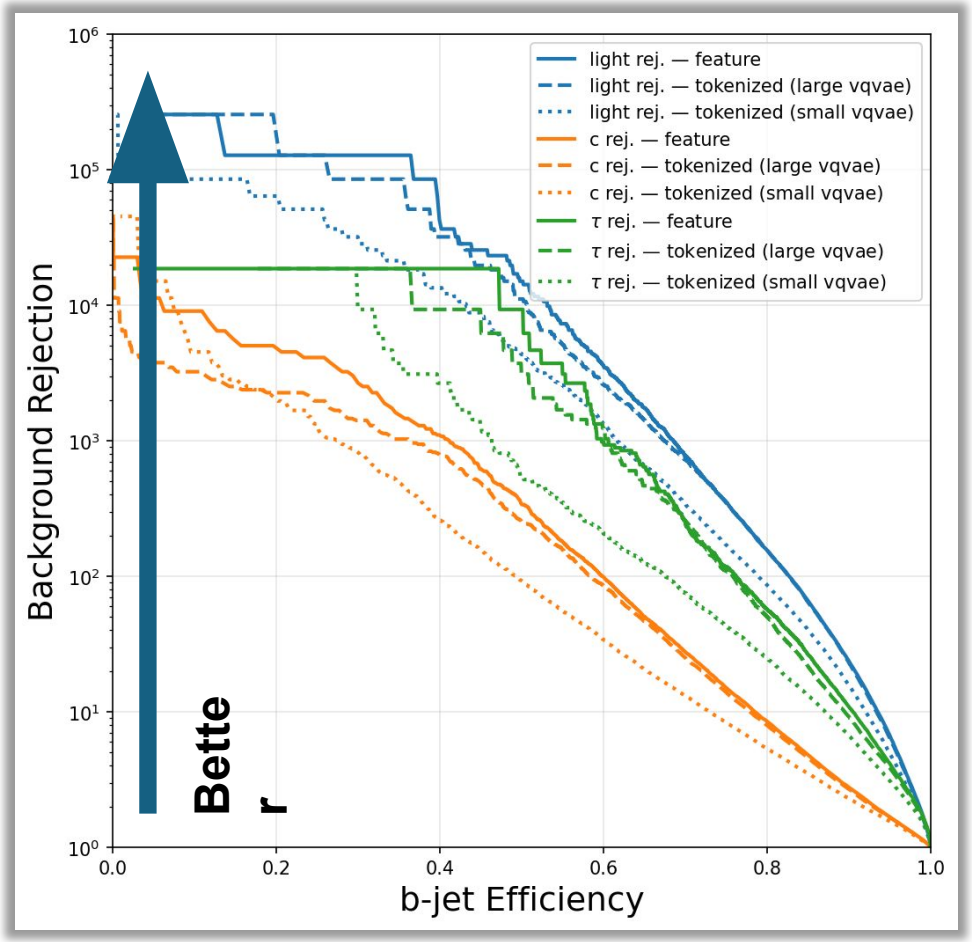


And achieves reconstructed jet resolution < 1% with large codebook size

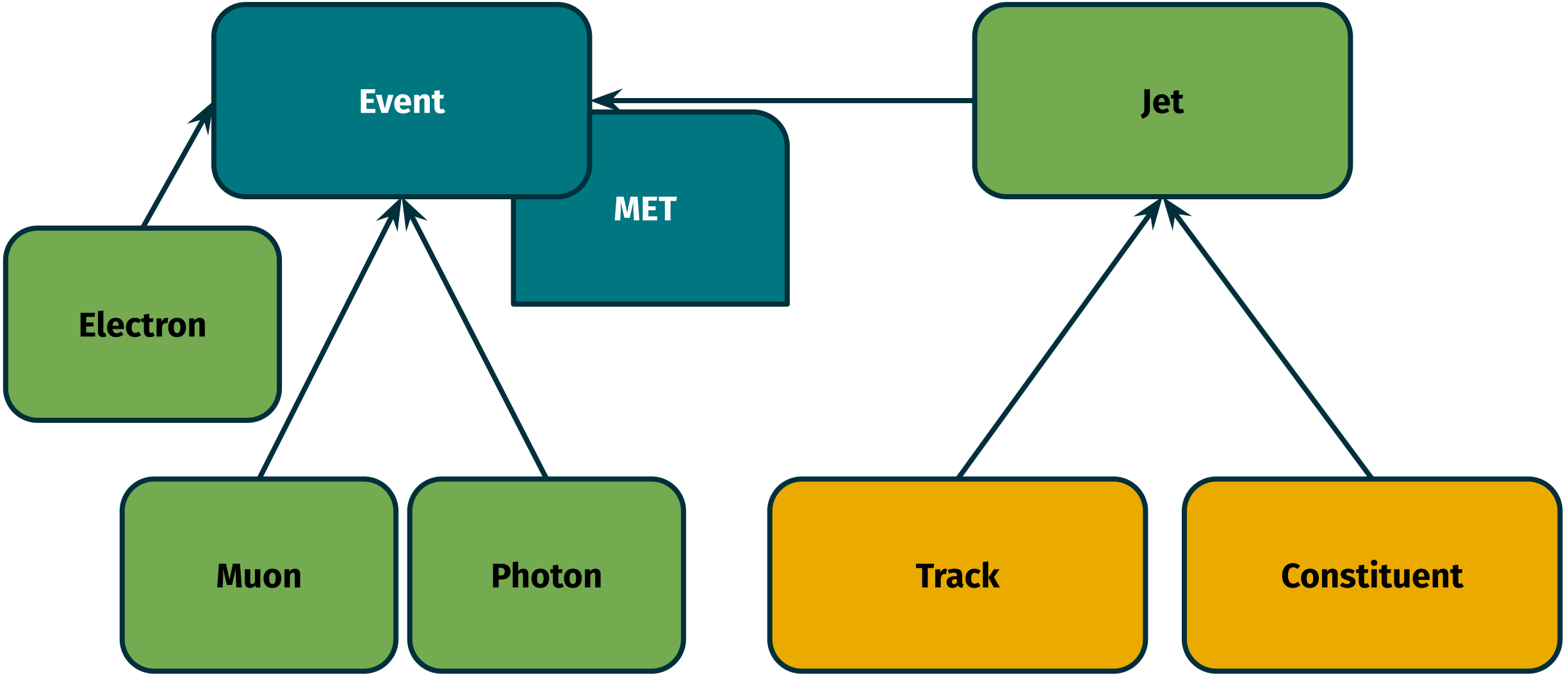
Flavor tagging (classification)

- Training flavor tagging algorithm on **large VQ-VAE** has close performance to **continuous features**

Solid = continuous inputs
Dashed = large VQVAE
Dotted = small VQVAE



Particle Cloud Data Format



Event

EventIndex **[Primary Key]**

Detector Index [Link to some other DB?]

Simulation tag

Run number

Event number

~~Primary Vertex x, y, z~~ Beam Position x, y, z

(Auxiliary Info ObjectID)

MET

MET_phi

Change to attach MET to event

Objects (Electron, Muon, Photon)

index	**[Primary Key]**
eventIndex	[Link to Event]
pt	
eta	
phi	
charge (electron and muon only)	
truth	

Jet (Large and Small *R*)

index	**[Primary Key]**
eventIndex	[Link to Event]
pt	
eta	
phi	
m	
truth / flavor (large-R / small-R)	

Constituents (Particle Flow)

index [Primary Key]

jetIndex

eventIndex

pt

eta

phi

m

- *For now, R = 0.4 only*
- *Read both charged and neutral constituents from PHYSLITE and follow jet → constituent element links*
- *NB: ElementLinks are not pleasant to use from Uproot*

Tracks

index **[Primary Key]**

eventIndex

jetIndex

q_p

theta

phi

d0

z0

Can later calculate a pt, eta, phi representation

File Format

- Use of indices as cross-references allows tables to be flat
- Enables storage as Parquet files with metadata such as column names and statistics
- Right software (e.g. Polars) can group by the indices to generate n-dimensional HDF5 tables