



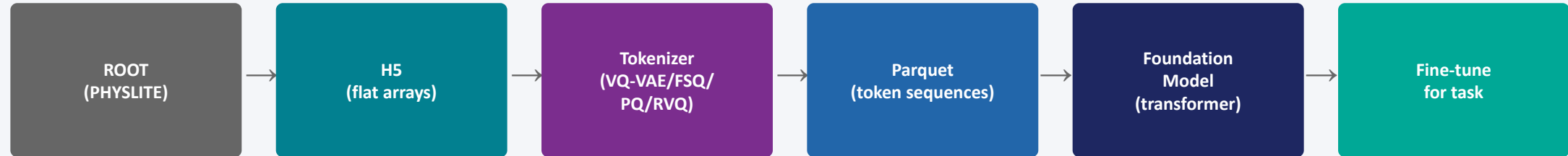
# Event Level tokenization

Viviana Cavaliere (BNL)



# How do you do event level tokenization?

## The Pipeline



All of this is here right now but will be moved to our github:  
<https://gitlab.cern.ch/vcavalie/bnl-treasure>

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

-----

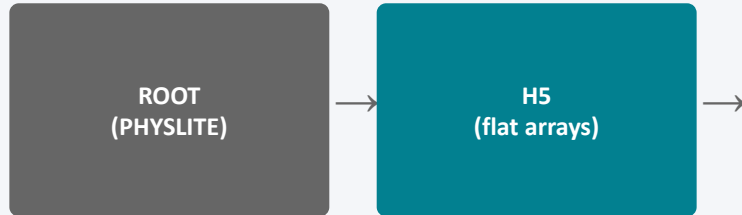
```

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

# Step 1

## The Pipeline

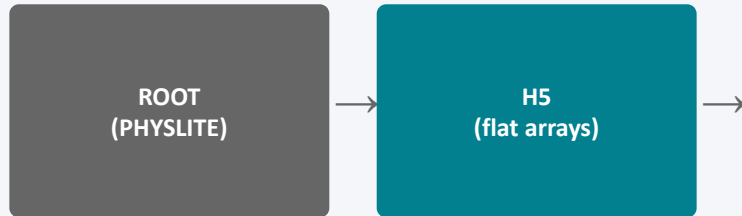


Fine-tune  
for task

*atlas\_to\_h5\_converter.py*

# Step 1

## The Pipeline



```
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    : q0verP, chiSquared, nDoF
atlas/event     : event_number, run_number, mcChannelNumber
```

## *atlas\_to\_h5\_converter.py*

```
Isolation definition (common/*/trk_iso03)
```

```
-----
```

```
I_trk = sum_pT(tracks,  $\Delta R < 0.3$ ,  $pT > ISO\_PT\_FLOOR$ ,
                $|z_0 \cdot \sin\theta| < ISO\_ZOST\_CUT$  w.r.t. PV)
        / pT(object)
```

```
where ISO_PT_FLOOR = 0.5 GeV (500 MeV)
```

```
ISO_ZOST_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.
```

```
""""
```

# How do you do event level tokenization?

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

- How to define event level tokens?
  - Each particle becomes a token via a shared linear projection:
  - Particle  $i$  input vector:
  - $[p_T, \eta, \phi, E, \text{mass}, \text{charge}, \text{isElectron}, \text{isMuon}, \text{isPhoton}, \text{isJet}, \dots]$
  - $\downarrow$  shared linear layer (same weights for all particles)
  - $d$ -dimensional embedding vector
- Particle type is encoded as a one-hot feature ( $\text{isElectron}, \text{isMuon}, \dots$ ) 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 ( $p_T, \eta, \phi, E$ ).

# How do you do event level tokenization?

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

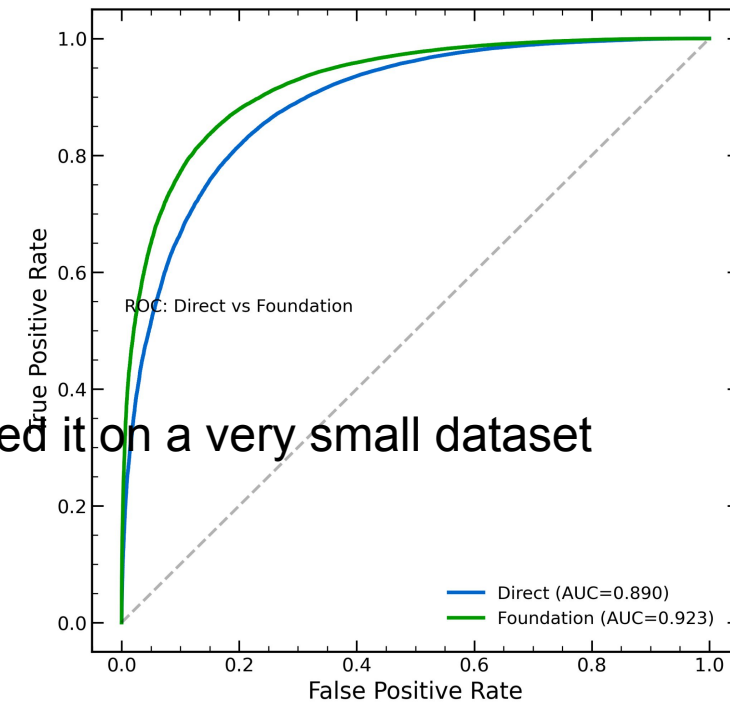
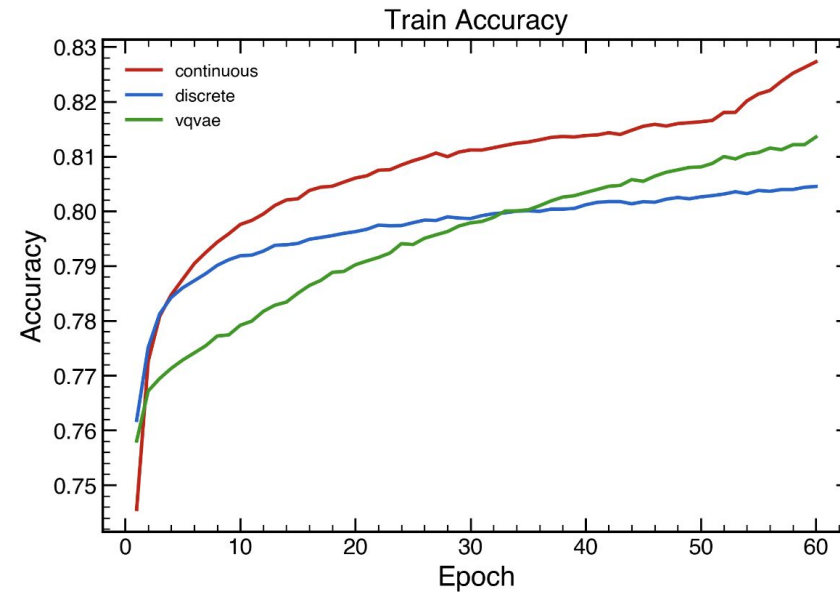
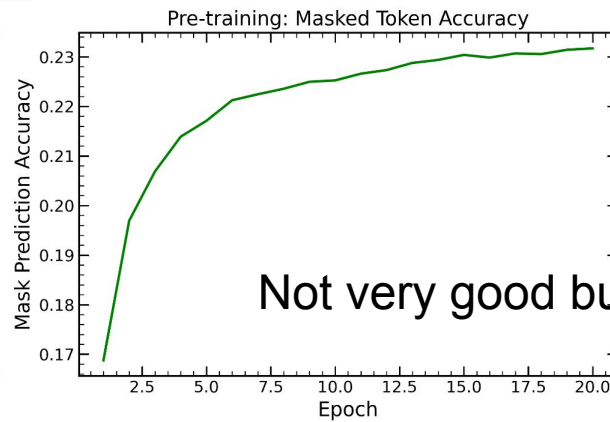
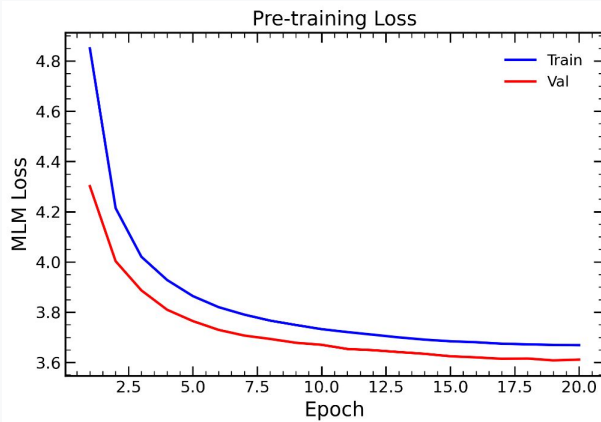
Or Each object type has its **own separate VQ-VAE** (or FSQ/PQ/RVQ) with its own codebook, because each object type has completely different features:

- Electron:  $p_T$ ,  $\eta$ ,  $\phi$ ,
- Jet:  $p_T$ ,  $\eta$ ,  $\phi$ , mass, nTrkPt500, QG vars, DL1d, GN2 (14 features)
- Track: qOverP,  $\eta$ ,  $\phi$ , d0, z0,  $p_T$ ,  $\chi^2$ , 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:
  - $\text{token\_repr} = \text{token\_embedding}(\text{token\_id}) + \text{type\_embedding}(\text{type\_id}) + \text{position\_embedding}(\text{pos})$
- 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  $\Delta\phi(\text{electron}, \text{jet})$  or  $\Delta R$  or invariant masses between objects.
- **But the transformer sees all objects together:** After tokenization, the event sequence looks like:
  - [CLS] [event\_ctx] [e<sub>1</sub>] [e<sub>2</sub>] [SEP] [ $\mu_1$ ] [SEP] [j<sub>1</sub>] [j<sub>2</sub>] [j<sub>3</sub>] [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  $\phi$  values. The attention mechanism can learn that "electron<sub>1</sub> has  $\phi=0.8$  and jet<sub>1</sub> has  $\phi=0.9$ , 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

# Hierarchical Feature-Group Tokenization

Multiple sub-tokens per object, aggregated by inner transformer

## Flat (current)

Each object  $\rightarrow$  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  $\rightarrow$  Linear([ $p_T$ ,  $\eta$ ,  $\phi$ , mass, ..., GN2\_pu])  $\rightarrow$  hidden dim

## Hierarchical (feature groups)

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

Jet  $\rightarrow$  [kinematics:  $p_T, \eta, \phi, m$ ] + [substructure: QG]  
+ [b-tagging: DL1d, GN2]  
 $\rightarrow$  inner attention  $\rightarrow$  1 jet vector  $\rightarrow$  outer transformer

## Feature groups per object:

Electron: [kinematics:  $p_T, \eta, \phi$ ] + [ID: LHLoose/Med/Tight]

Tau: [kinematics:  $p_T, \eta, \phi$ ] + [ID: nTracks, RNN scores]

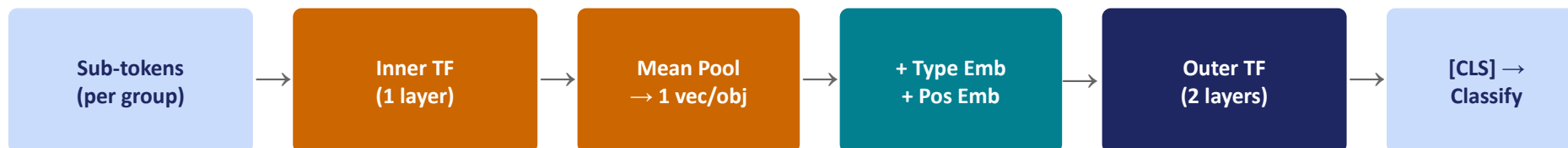
Jet: [kinematics:  $p_T, \eta, \phi, m$ ] + [substructure: QG vars] + [b-tagging: DL1d+GN2]

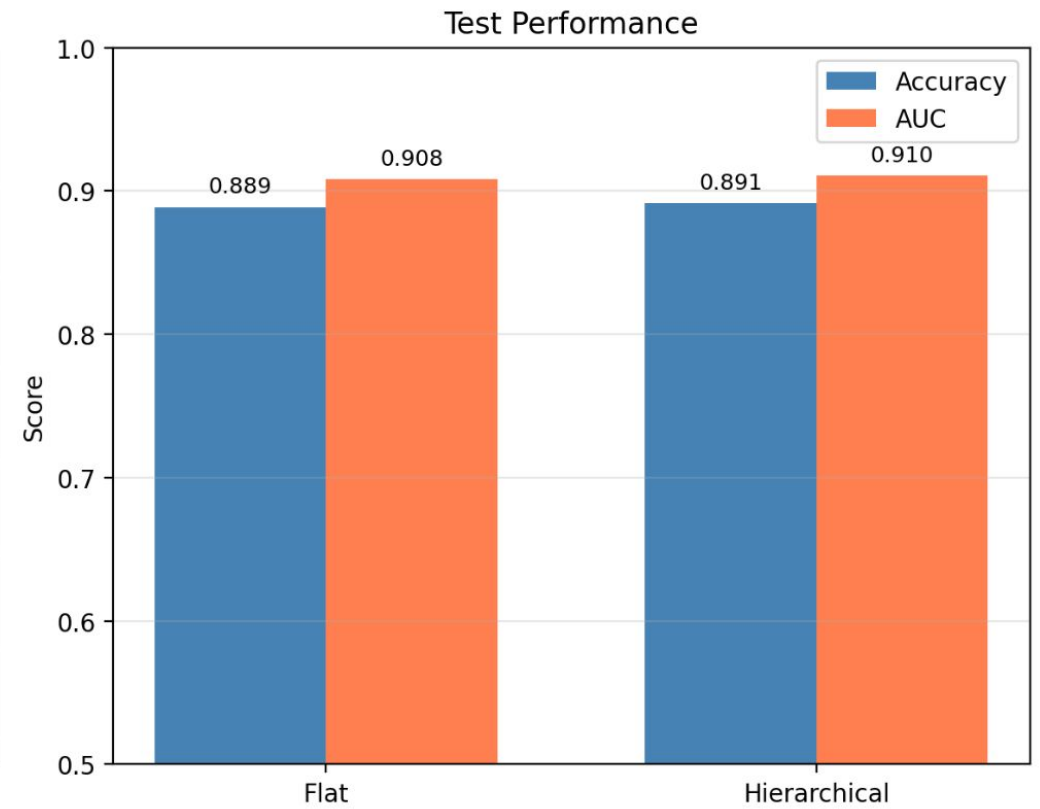
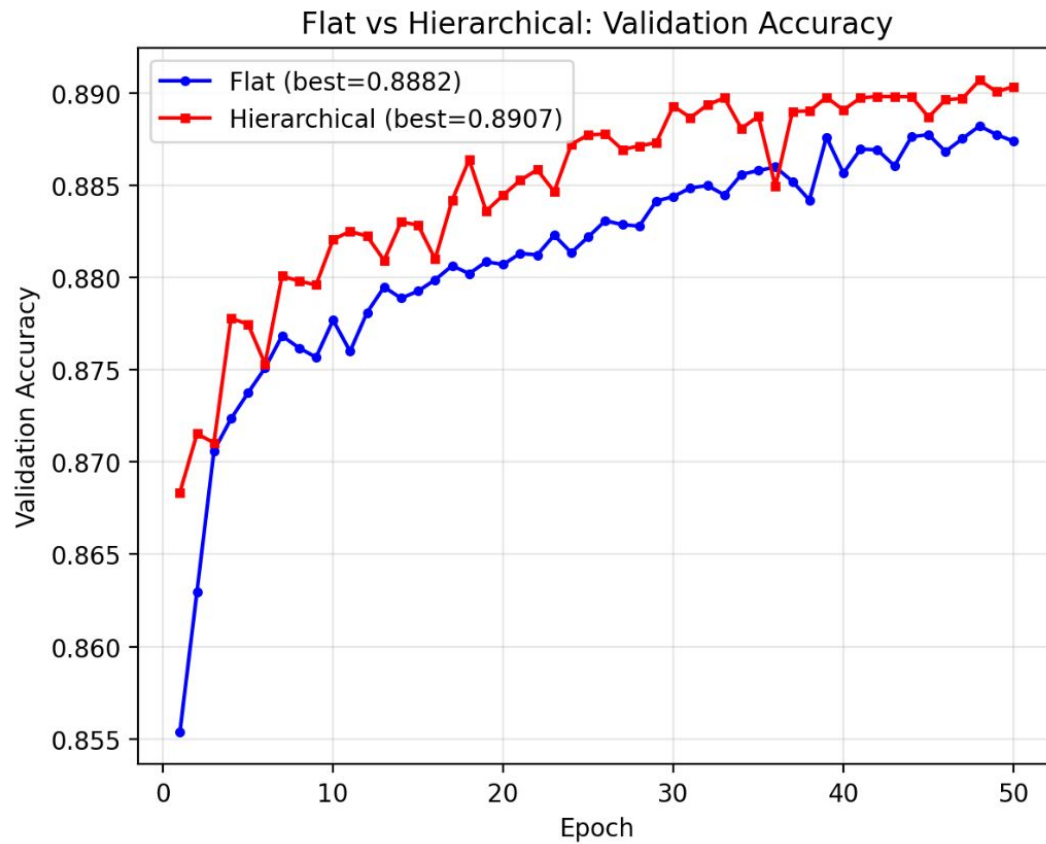
$\chi^2, nDoF$

Muon: [kinematics:  $p_T, \eta, \phi$ ] + [ID: quality]

Photon: [kinematics:  $p_T, \eta, \phi$ ] + [ID: isLoose/Med/Tight]

Track: [kin] + [impact:  $d_0, z_0$ ] + [quality:





# Constituent + Hit Hierarchical Architecture

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

Event: [CLS] ctx  $e_1$   $e_2$   $\mu_1$   $\tau_1$   $\gamma_1$

$j_1$   $j_2$   $j_3$

$t_1$   $t_2$   $t_3$

MET

↑ gated residual injection ↑

## Jet Constituents (tracks/clusters inside jet)

Per jet: up to 50 constituent tracks/clusters

Each:  $p_T$ ,  $\eta$ ,  $\phi$ , charge,  $d_0$ ,  $z_0$

1. Project features  $\rightarrow$  hidden dim
2. Constituent Transformer (1 layer)
3. Attention pooling  $\rightarrow$  1 vector per jet
4. Gated residual:

$$\text{jet\_enriched} = \text{jet} + \sigma(W \cdot [\text{jet} // \text{const}]) \times \text{const}$$

Gate learns how much constituent info to add.

When  $\text{gate} \approx 0$ : pure object-level. When  $\text{gate} \approx 1$ : constituent-driven.

## Track Hits (detector measurements along track)

Per track: up to 30 detector hits

Each:  $\eta$ ,  $\phi$ , energy, layer, time

1. Project features  $\rightarrow$  hidden dim
2. Hit Transformer (1 layer)
3. Attention pooling  $\rightarrow$  1 vector per track
4. Gated residual:

$$\text{track\_enriched} = \text{track} + \sigma(W \cdot [\text{track} // \text{hit}]) \times \text{hit}$$

Same architecture as jet constituents.

Hit patterns reveal particle type ( $e/\mu/\pi$ ), 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.

# Backup

# How the Hierarchical Models Work — Step by Step

## 1 Split each object's features into meaningful groups

Instead of feeding all 14 jet features through one projection, we separate them by what they describe:

*Each group becomes its own sub-token → the model explicitly sees "this is kinematics" vs "this is b-tagging"*

### Kinematics

$p_T$ ,  $\eta$ ,  $\phi$ , mass

*Where is it?*

*How energetic?*

### Substructure

nTrkPt500, QG vars

*What does it*

*look like inside?*

### B-tagging

DL1d, GN2 scores

*Does it contain*

*a b-quark?*

## 2 Inner transformer lets sub-tokens "talk to each other"

A small 1-layer transformer processes the 3 sub-tokens together. Through self-attention, it can learn patterns like:

"this jet has high  $p_T$  AND high DL1d\_pb" — combining kinematics with b-tagging information. The flat approach has to discover this structure from one blended vector, which is harder.

## 3 Pooling: condense the sub-tokens back into one object vector

Mean pooling = take the average of each dimension across all sub-tokens:

kinematics vector: [0.3, -0.7, 0.1, 0.5, ...]

substructure vector: [0.1, 0.2, -0.4, 0.8, ...]

b-tagging vector: [0.9, 0.1, 0.3, -0.2, ...]

---

mean pooled result: [0.43, -0.13, 0.0, 0.37, ...] ← average of each column

Attention pooling (used for constituents/hits):

Instead of equal weights, a learnable query attends over the sub-tokens and assigns different importance to each.

A displaced high- $p_T$  track inside a b-jet gets higher weight than a soft pion — the model learns which sub-parts matter most.

## 4 Event transformer reads the pooled object vectors — one per object — exactly like the flat approach

The outer transformer sees: [CLS] [event] [electron<sub>1</sub>] [muon<sub>1</sub>] [jet<sub>1</sub>] [jet<sub>2</sub>] [MET] — same as before, but each object vector is richer because it was built from attended sub-tokens instead of a single mixed projection.