Brookhaven National Laboratory

U.S. DEPARTMENT of ENERGY

Event Level tokenization: discussion points

Viviana Cavaliere (BNL)

@BrookhavenLab

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.

Studies we want to do

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

- Step 1 add this to heptokens
  - Lots of the existing codes is in private directories
- How are we going to quantify tokenization loss
  - Show plots
  - Accuracy for a particular task
  - What else?
- context vs non-context aware tokens
- Feature or non feature grouping
- Which Task are we targeting H->ZZ is easy and all doable with open data but has not jets

Backup

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

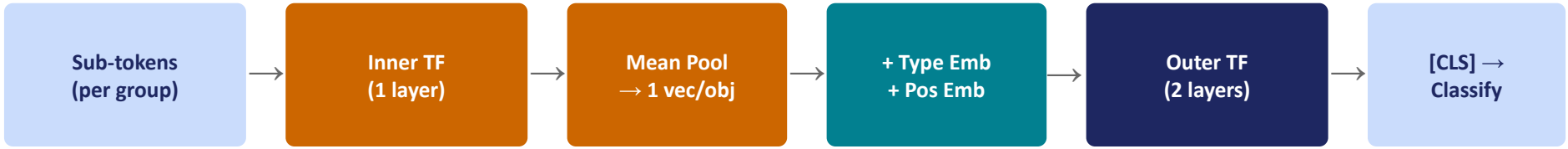
Hierarchical (feature groups)

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]

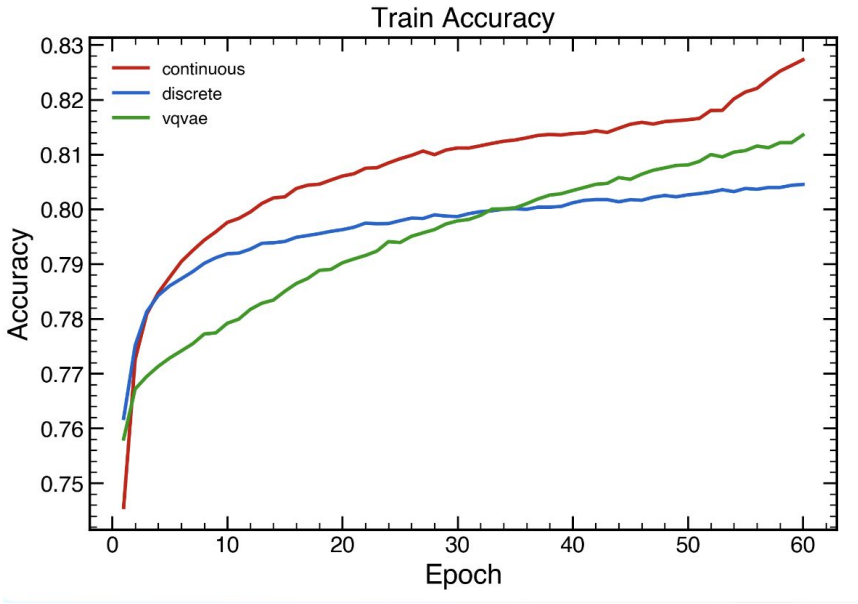


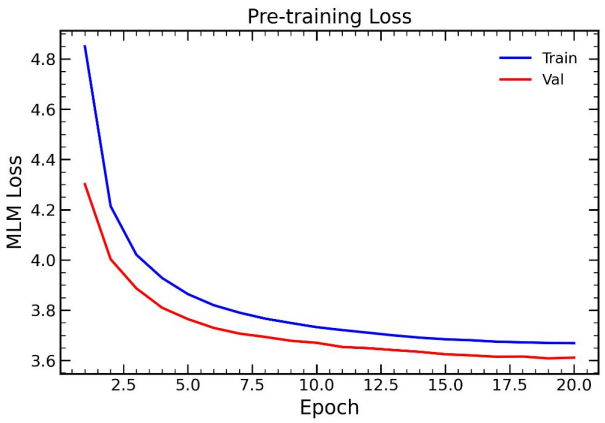
Brookhaven National Laboratory

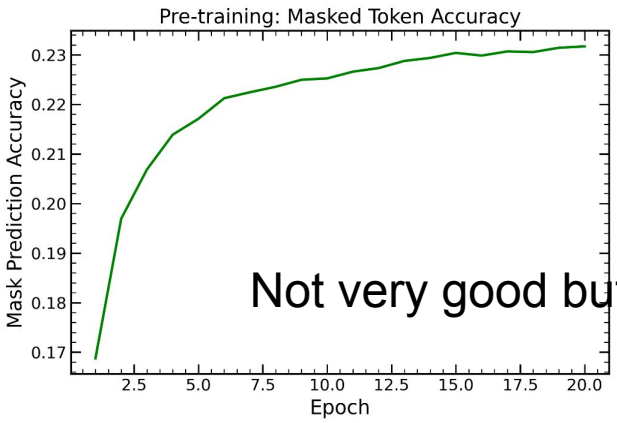
Task 1

$H \rightarrow ZZ \rightarrow 4\ell$

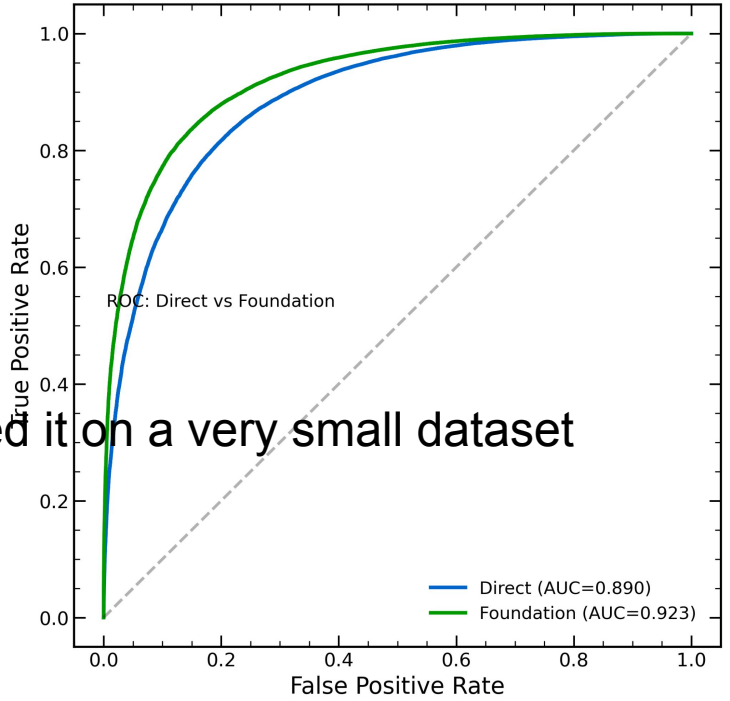
Train classifier to separate Higgs signal from background. Direct comparison: standard classifier vs foundation model.



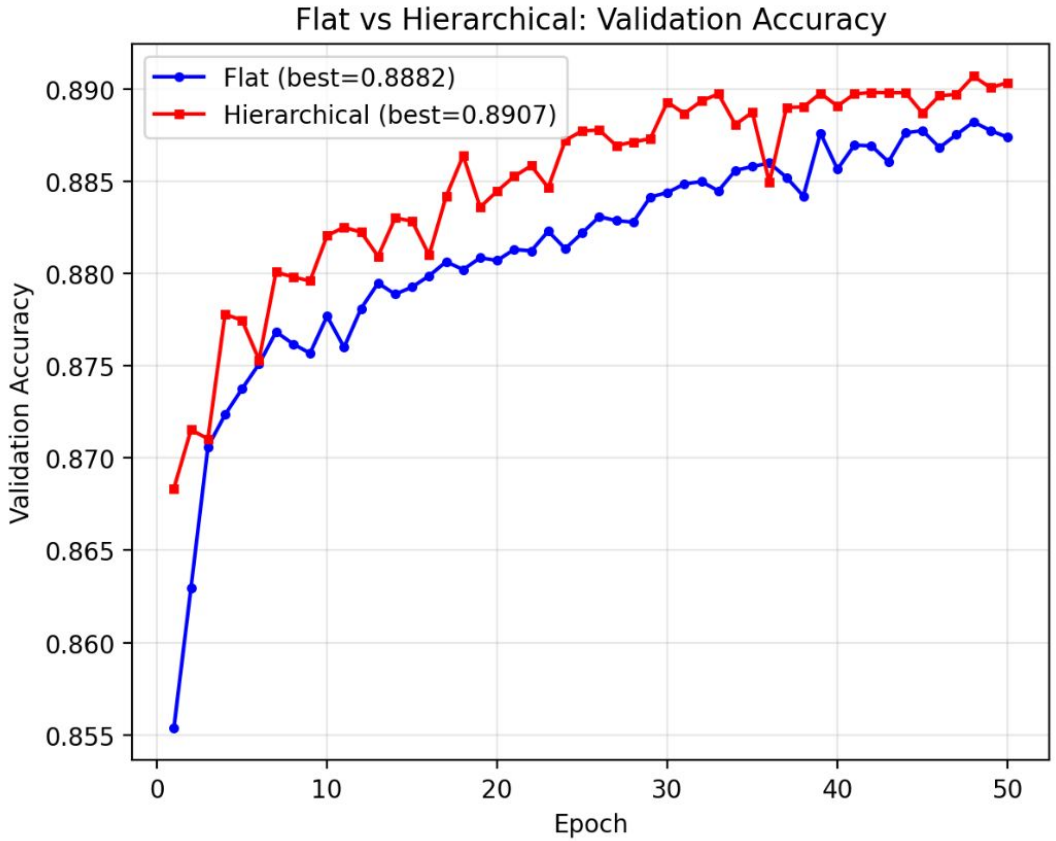


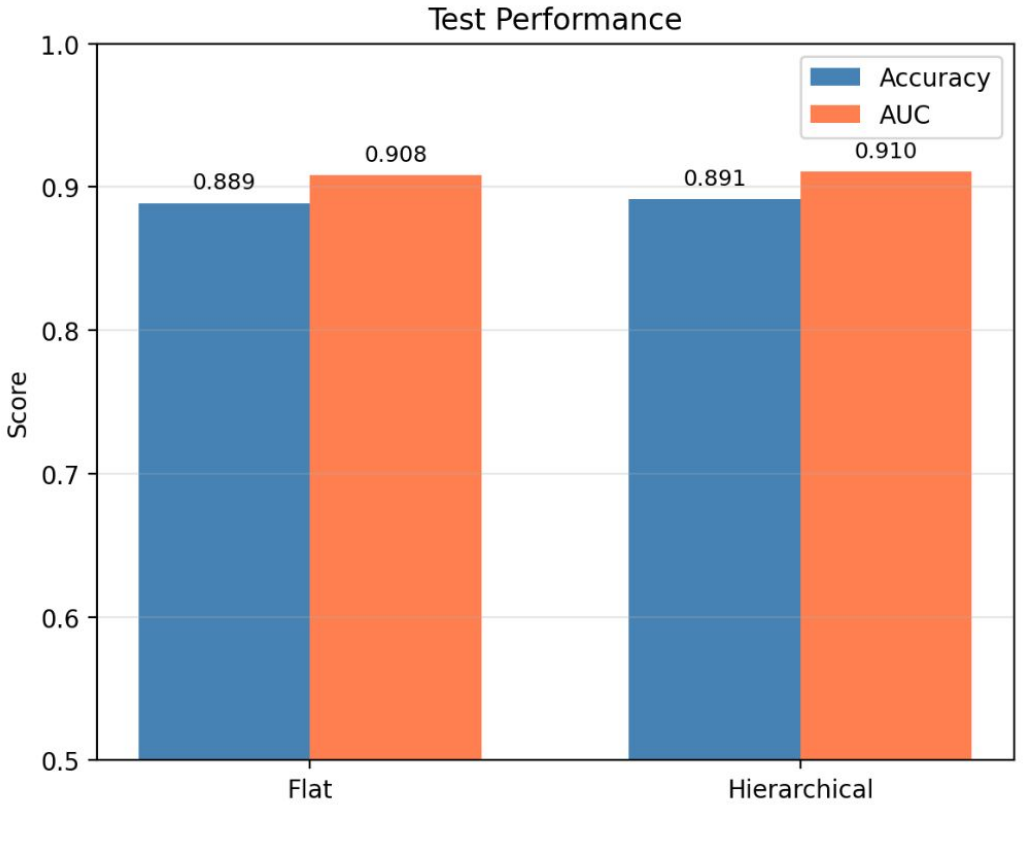


Not very good but trained it on a very small dataset



Brookhaven National Laboratory





Brookhaven National Laboratory

Constituent + Hit Hierarchical Architecture

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

Event: [CLS] ctx e_1 e_2 μ_1 τ_1 γ_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: 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

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	Substructure	B-tagging
`pT, η, φ, mass`	`nTrkPt500, QG vars`	`DL1d, GN2 scores`
Where is it? How energetic?	*What does it look like inside?*	*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 pT 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-pT 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$_1$] [muon$_1$] [jet$_1$] [jet$_2$] [MET] — same as before, but each object vector is richer because it was built from attended sub-tokens instead of a single mixed projection.