Foundation Models as a new tool to uncover the dark sector

Roadmap for dark matter models for Run 3 workshop, CERN, May 17 2024


Lisa Benato


Joschka Birk


Anna Hallin


Gregor Kasieczka

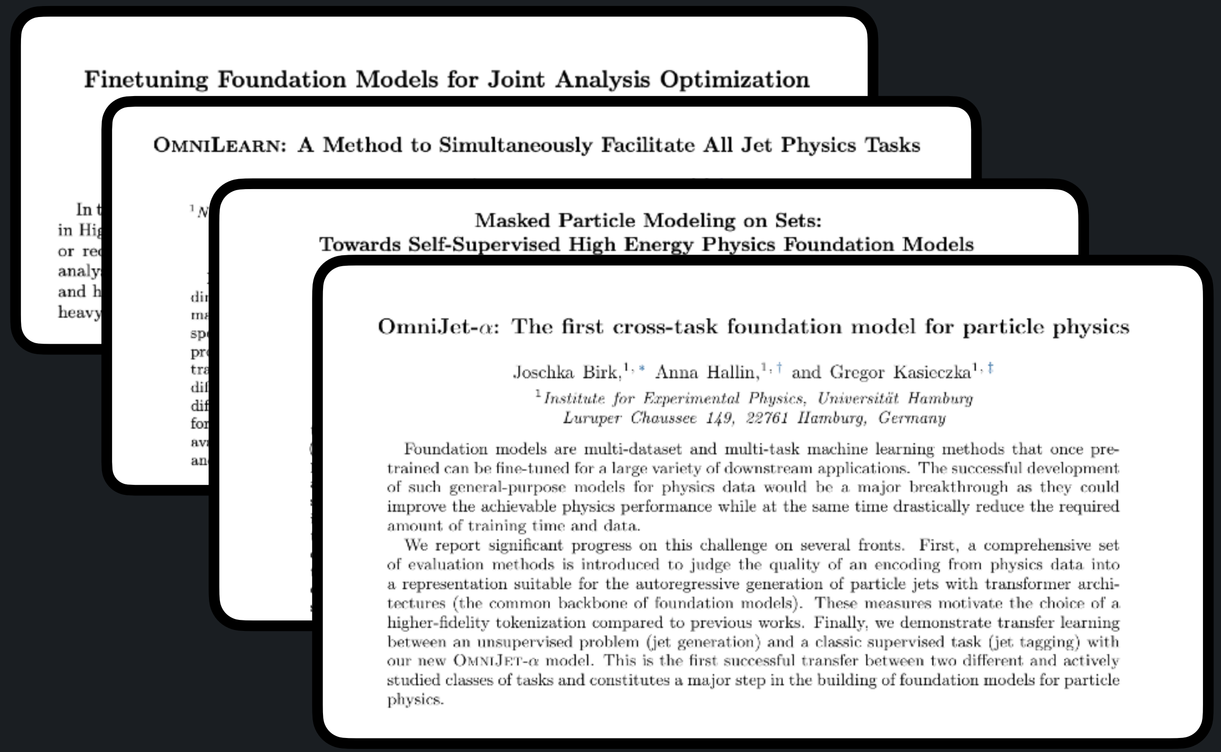
Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

CLUSTER OF EXCELLENCE
QUANTUM UNIVERSE

joschka.birk@uni-hamburg.de

arXiv:2403.05618

What is this talk about?


Finetuning Foundation Models for Joint Analysis Optimization

OmniLearn: A Method to Simultaneously Facilitate All Jet Physics Tasks

Masked Particle Modeling on Sets:
Towards Self-Supervised High Energy Physics Foundation Models

OmniJet-α: The first cross-task foundation model for particle physics

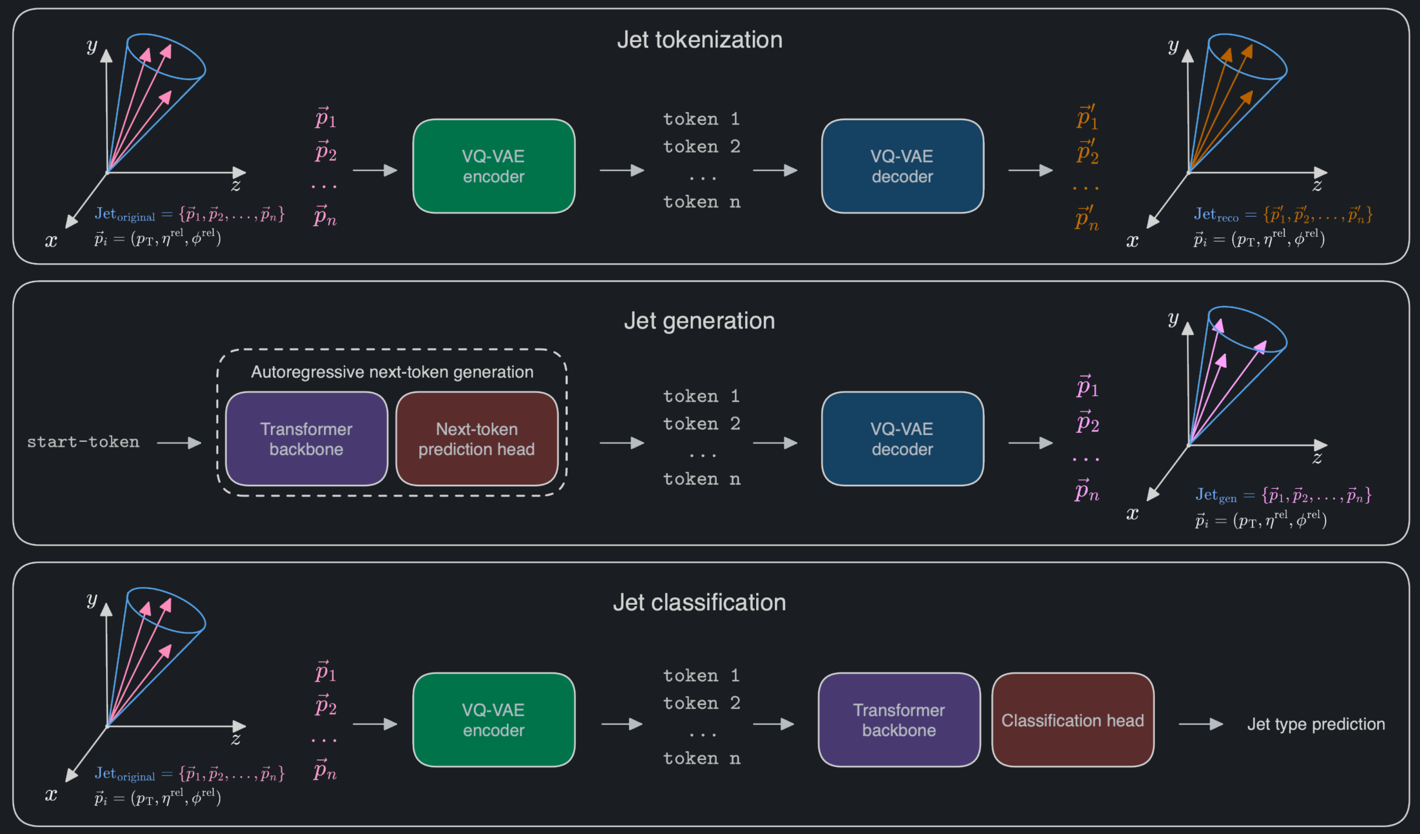
Joschka Birk,[1,*] Anna Hallin,[1,†] and Gregor Kasieczka[1,‡]
[1]Institute for Experimental Physics, Universität Hamburg
Luruper Chaussee 149, 22761 Hamburg, Germany

Foundation models are multi-dataset and multi-task machine learning methods that once pre-trained can be fine-tuned for a large variety of downstream applications. The successful development of such general-purpose models for physics data would be a major breakthrough as they could improve the achievable physics performance while at the same time drastically reduce the required amount of training time and data.

We report significant progress on this challenge on several fronts. First, a comprehensive set of evaluation methods is introduced to judge the quality of an encoding from physics data into a representation suitable for the autoregressive generation of particle jets with transformer architectures (the common backbone of foundation models). These measures motivate the choice of a higher-fidelity tokenization compared to previous works. Finally, we demonstrate transfer learning between an unsupervised problem (jet generation) and a classic supervised task (jet tagging) with our new OmniJet-α model. This is the first successful transfer between two different and actively studied classes of tasks and constitutes a major step in the building of foundation models for particle physics.

1. Foundation models in HEP - what is it and what is out there already?

2. Our OmniJet-α model - the first cross-task foundation model for HEP



3. How could this help in searches
for dark sector particles?

Towards foundation models in HEP

What is a foundation model?

Pre-train an ML model on one task/dataset, **then fine-tune** on other task/dataset

Promising avenue for particle physics:

- Use pre-trained (potentially large) models to fine-tune for specific tasks
- **No need to train every task from scratch**
- **Saves** compute and human **resources**
- **Pre-trained models need less data**

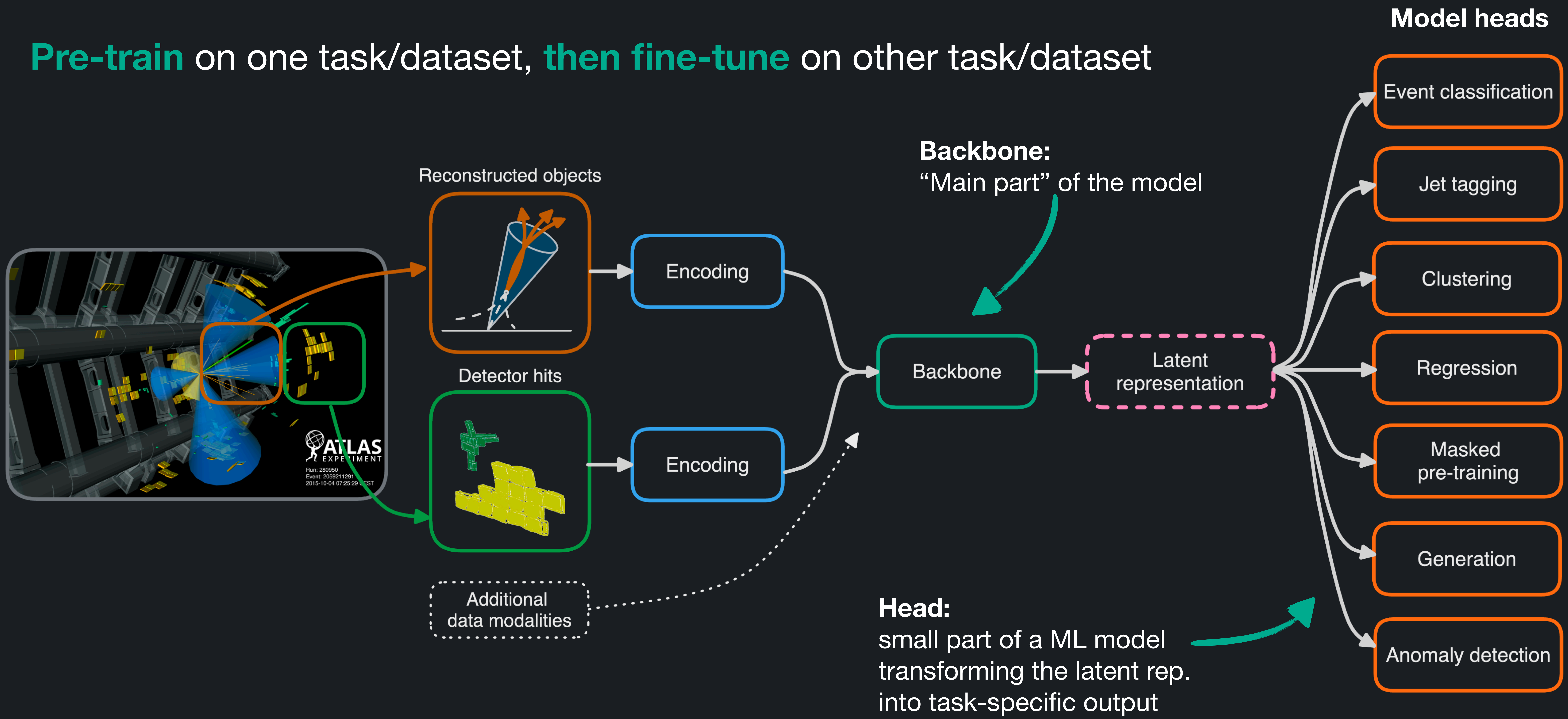
Towards foundation models in HEP

- Lots of interest for foundation models in HEP
- Have to be careful with the definition though, not just use as cool buzzword
- Few examples of what is out there already:
    - ParT (2202.03772): pre-training and fine-tuning on classification, but different datasets
    - MPM (2401.13537): pre-training on a surrogate task
    - OmniJet-α (2403.05618): generative pre-training (this talk)
    - OmniLearn (2404.16091): pre-training on multi-class classification and generation simultaneously

Consistent trend:
Pre-training results in better performance of the downstream task (compared to training that task from scratch)

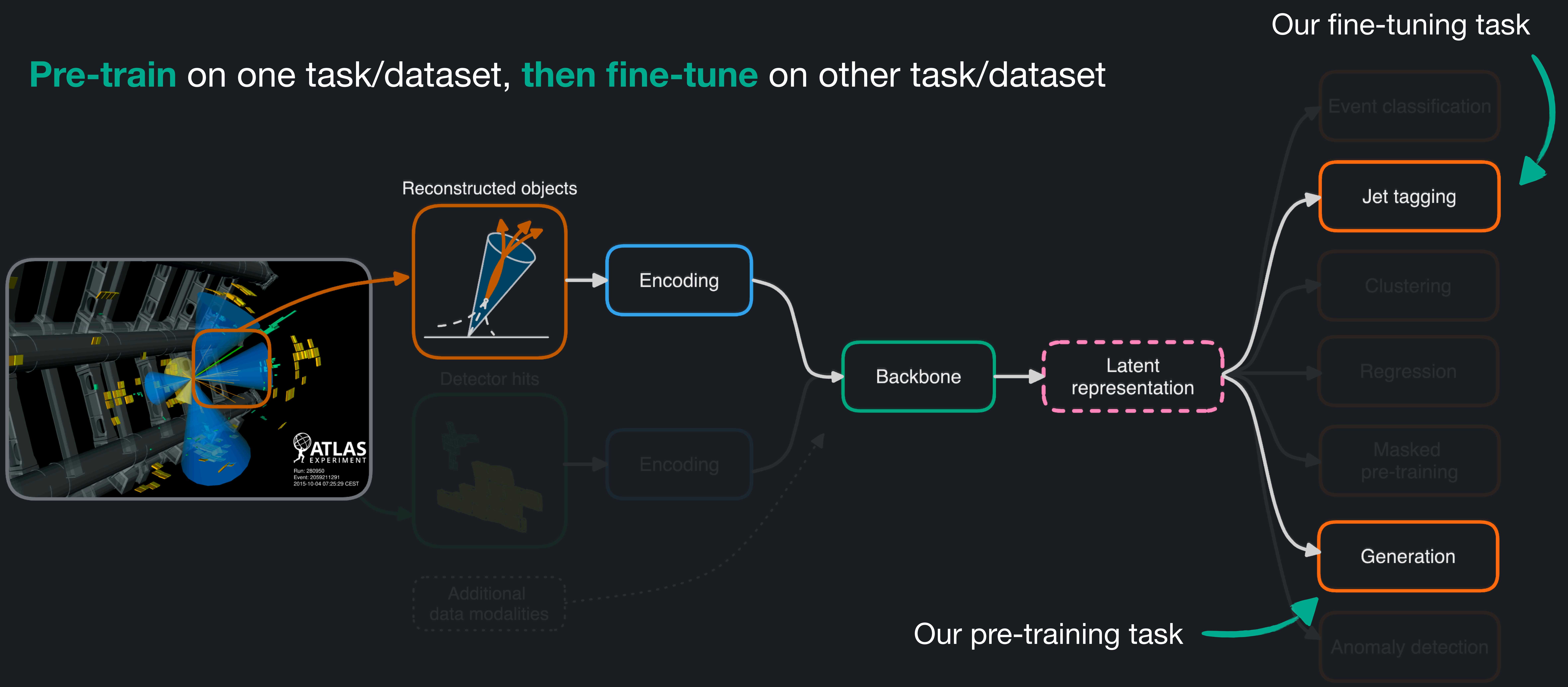
Foundation models in HEP

Pre-train on one task/dataset, **then fine-tune** on other task/dataset

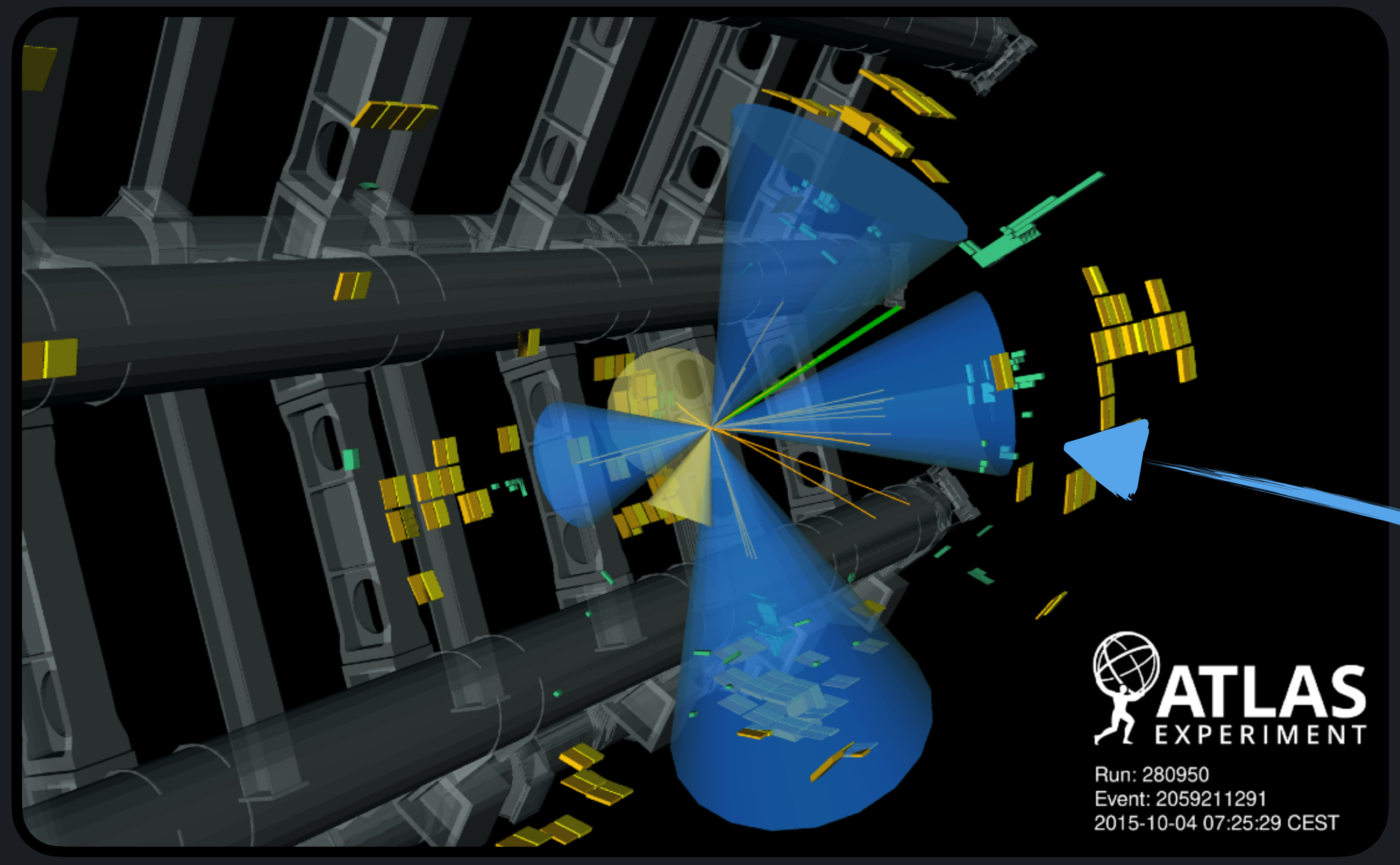


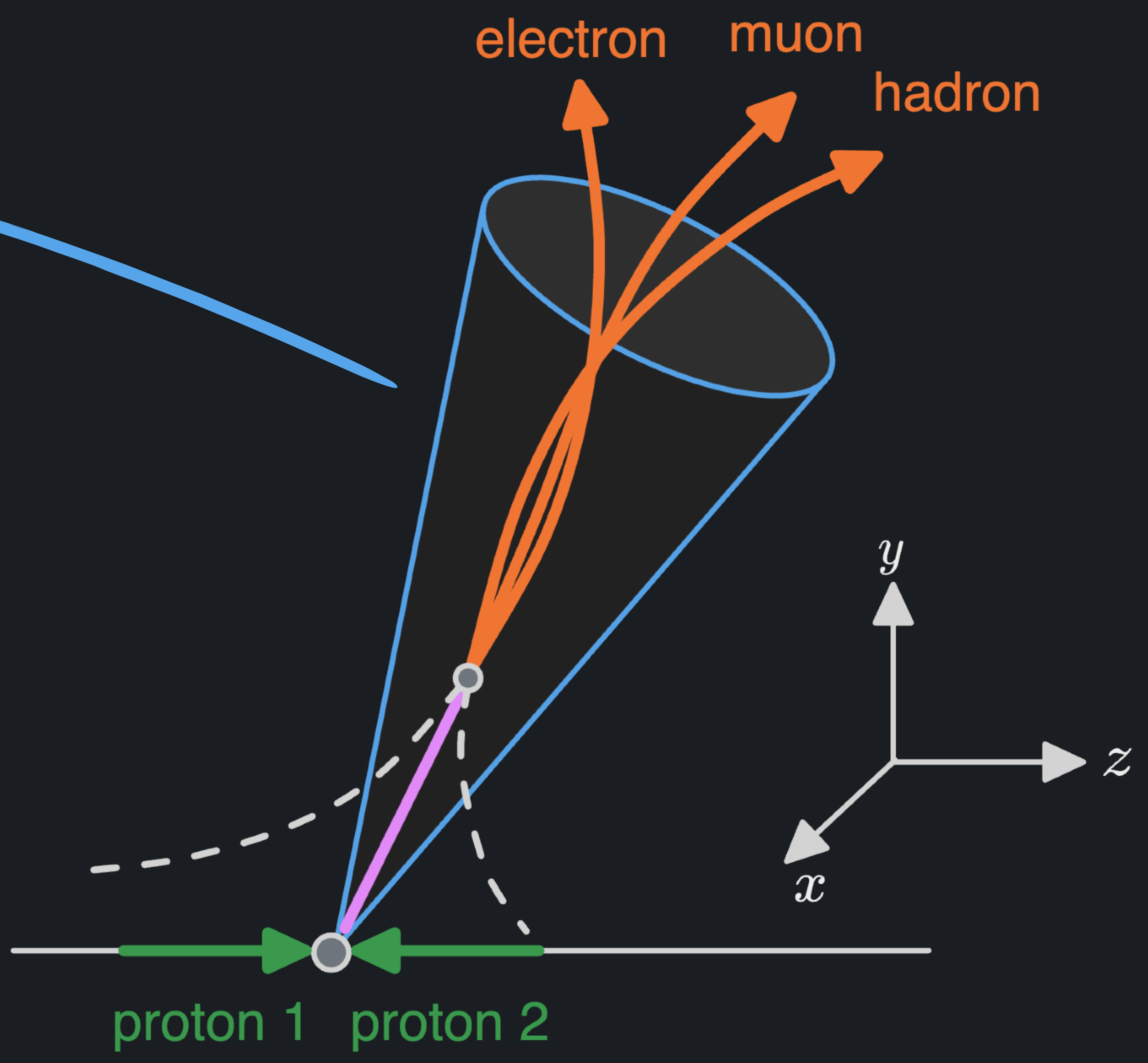
Foundation models in HEP

Pre-train on one task/dataset, **then fine-tune** on other task/dataset



Dataset



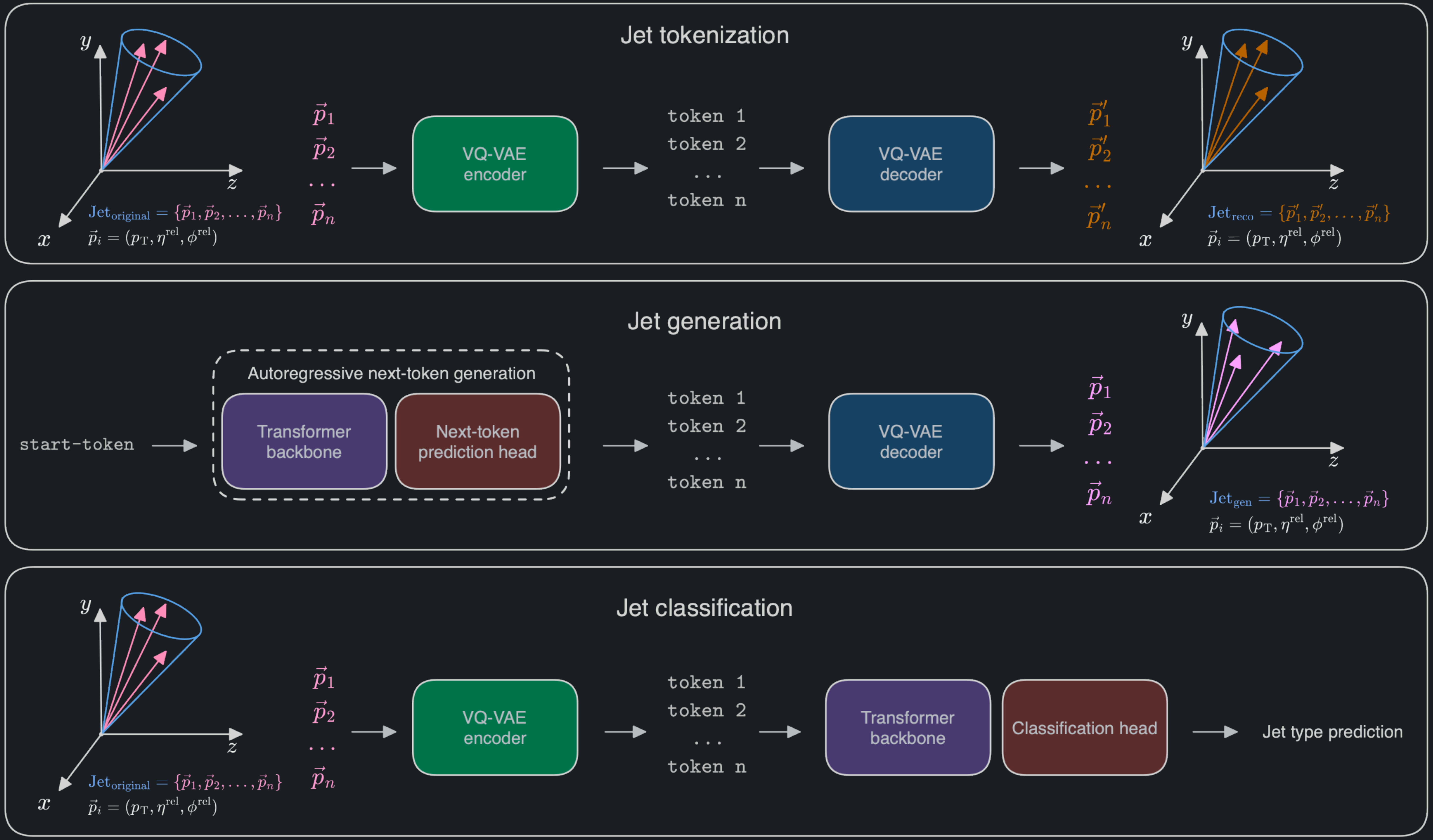


Dataset used here:

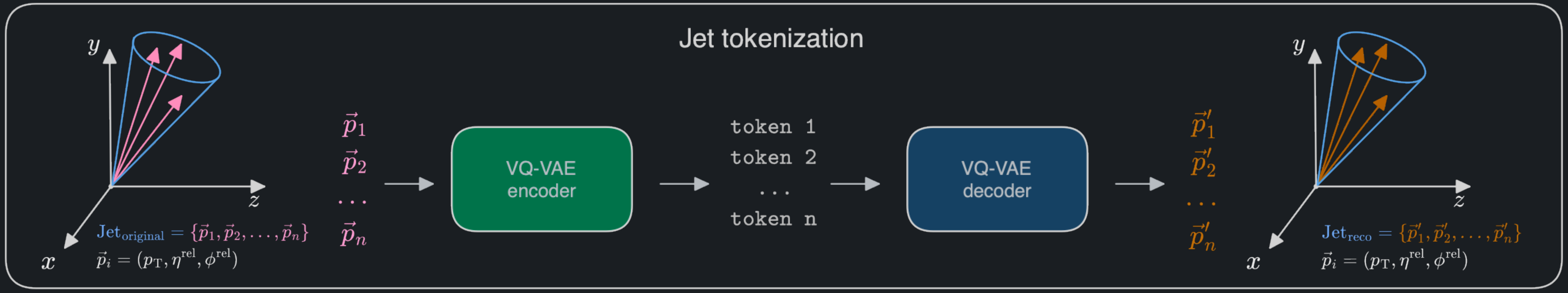
- JetClass dataset [1]
- Contains 10M training jets for 10 different jet types (e.g. q/g, $t \to bqq'$, $H \to b\bar{b}$, ...)
- Kinematic features: p_T, η, ϕ of the particles (subset of the available features)

[1] Qu, H., Li, C., & Qian, S. (2022). JetClass: A Large-Scale Dataset for Deep Learning in Jet Physics (1.0.0) [Data set]. Zenodo. https://doi.org/10.5281/zenodo.6619768

Model architecture overview



Model architecture overview - tokenization



Encoded particles are mapped to closest token vector

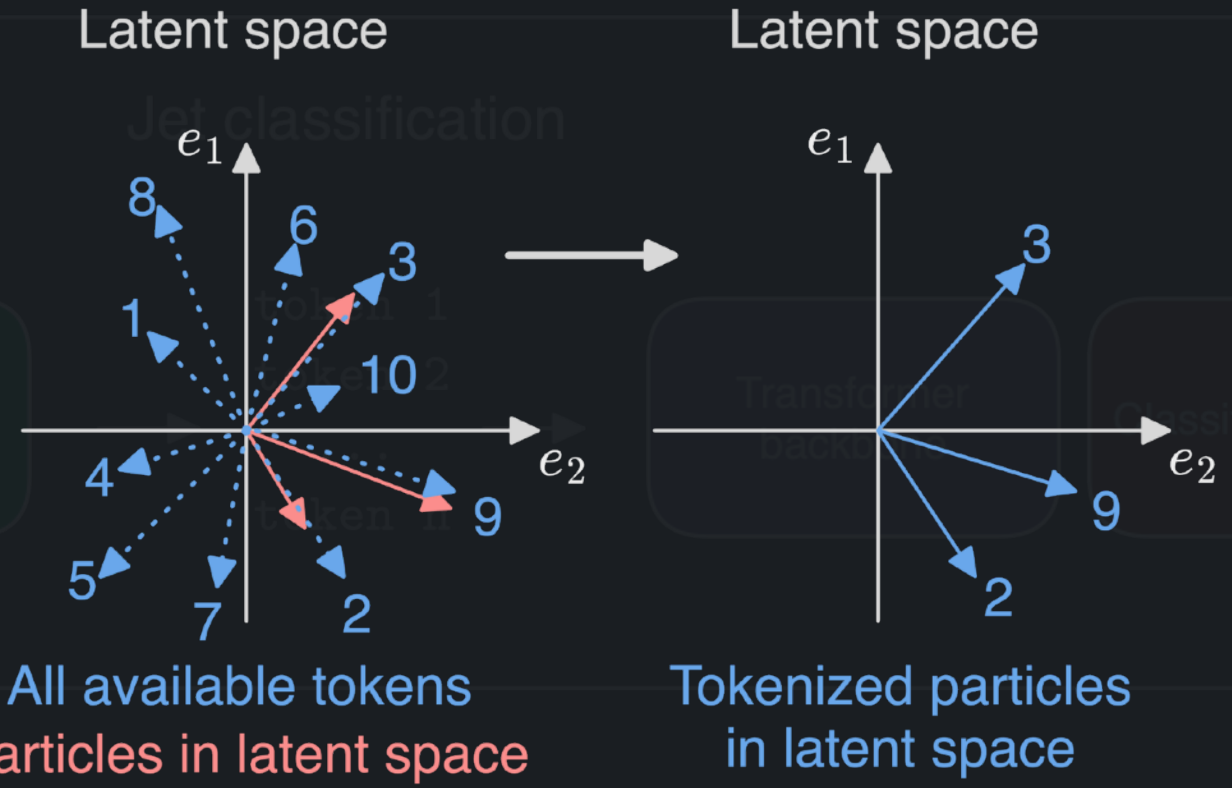
Codebook:

Look-up table *latent vector* ↔ *token-id*

Input for tokenization:

3 continuous features p_T, η, ϕ

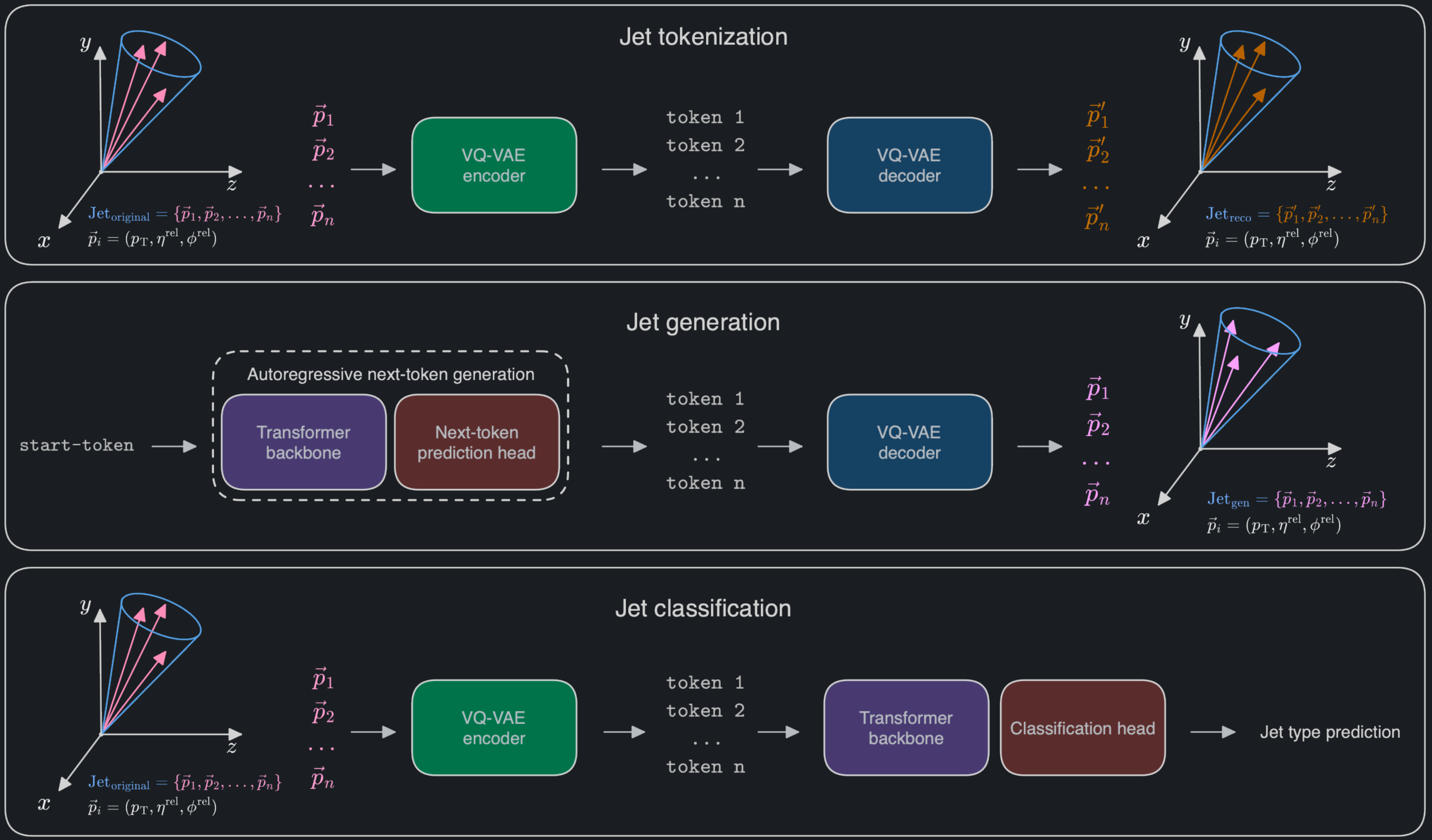
VQ-VAE encoder maps cont. features into latent space



VQ-VAE decoder maps latent vectors back to physical space

VQ-VAE = Vector-quantized variational autoencoder

Model architecture overview



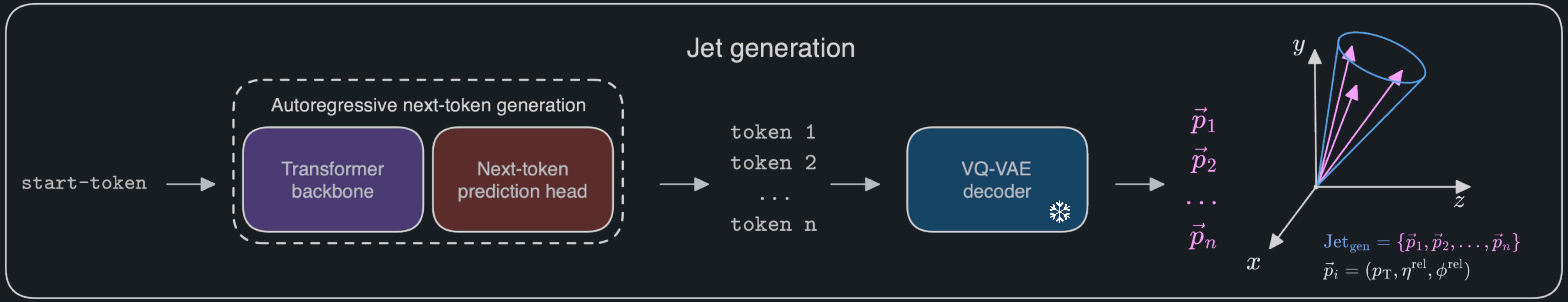
Model architecture overview - generation

Input for generative network: only the start token

Tokens are generated auto-regressively

VQ-VAE decoder (frozen) maps latent vectors back to physical space

Output: set of particles with p_T, η, ϕ values



Generative network yields a sequence of tokens

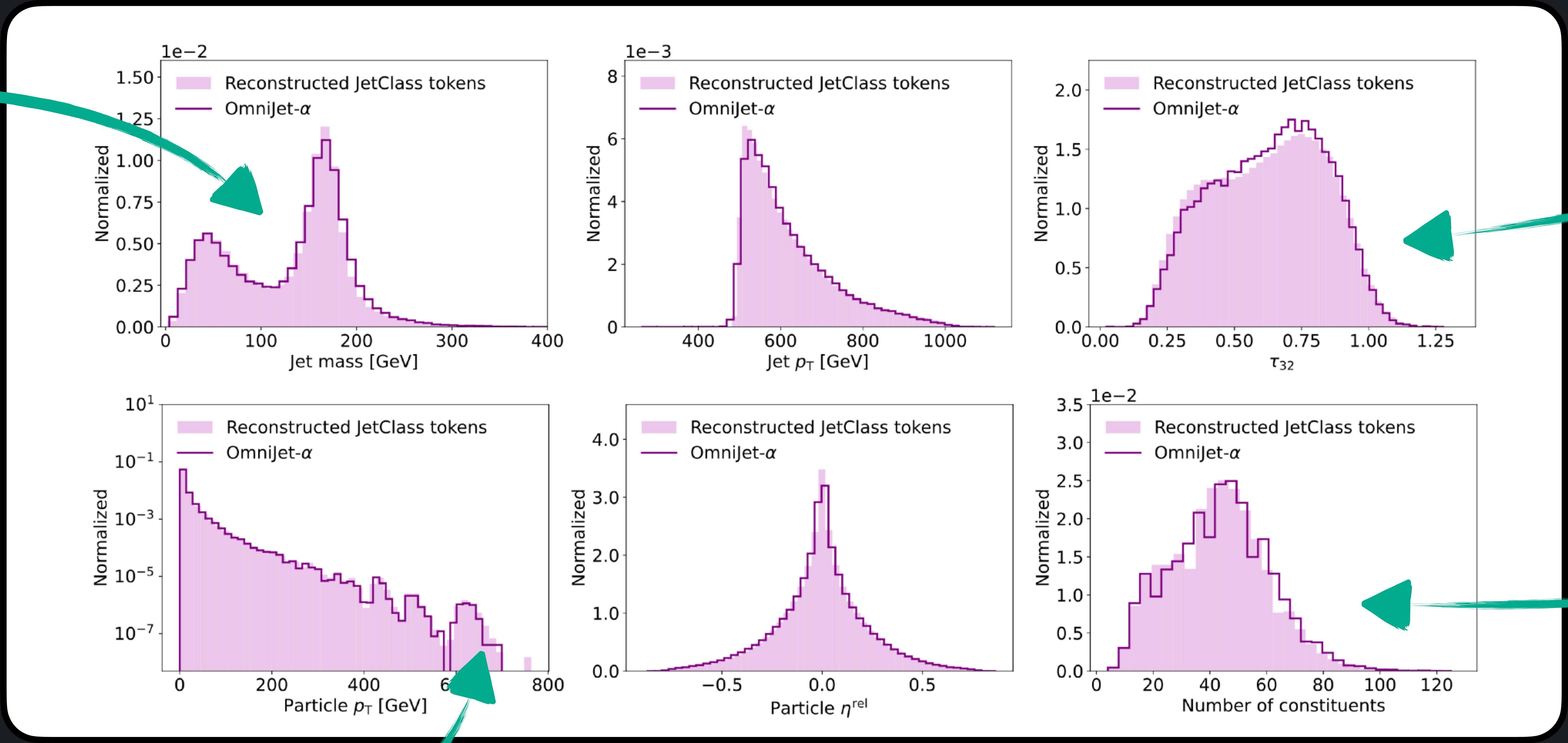
Jet generation results

- Idea: while learning to generate, a model also learns aspects of the data useful for other tasks
- Advantage of this approach: our pre-training task is a useful task by itself
- Generated jets show good agreement, both on jet-level and particle-level

  → Next-token prediction is a powerful method to generate jets

Jet substructure is quite well modeled

Generative (pre-)training done on dataset containing q/g and $t \to bqq'$ jets



The model also learns when to stop the generation

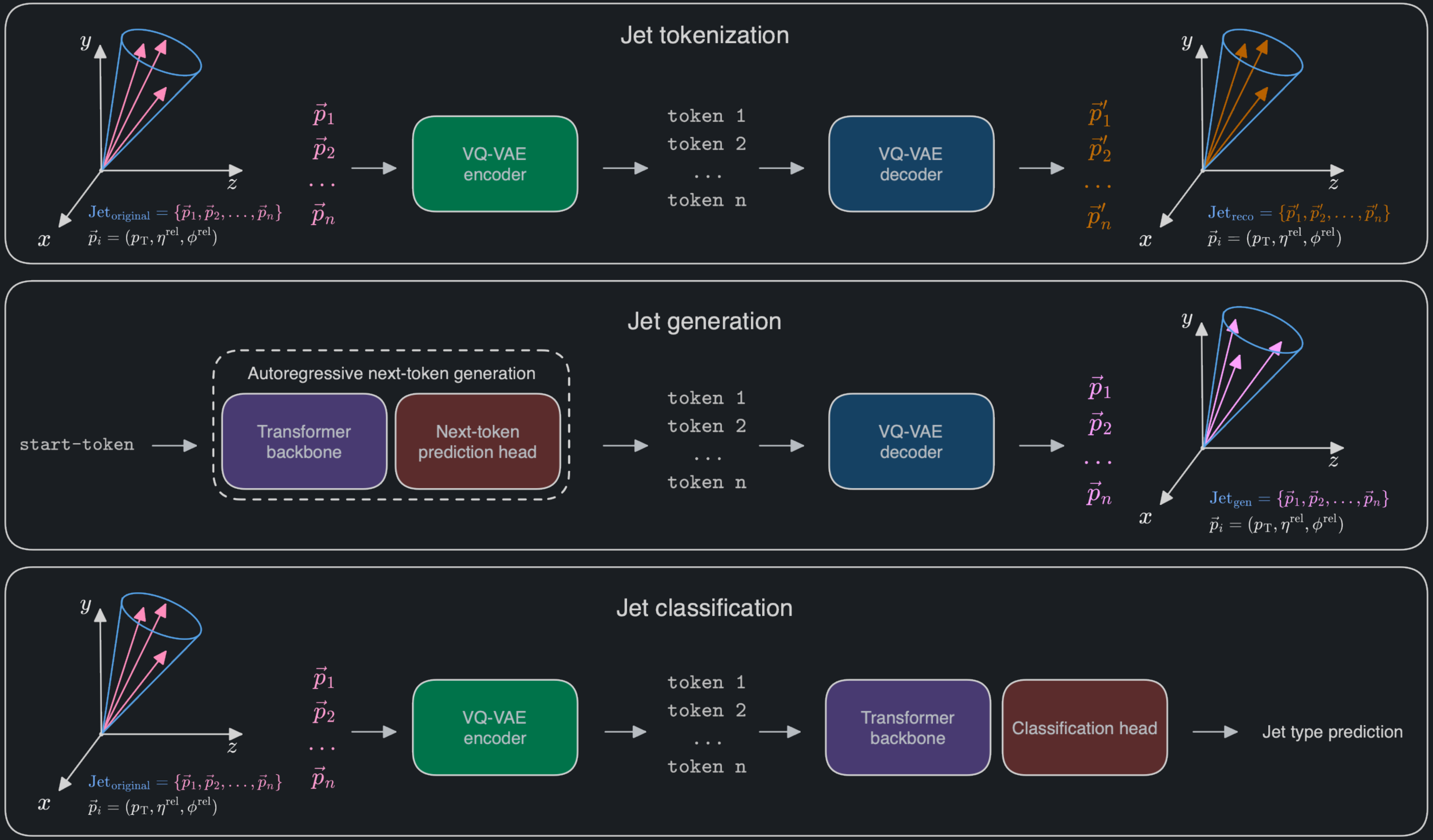
These bumps are due to the tokenization

Model architecture overview

✔

✔

⇨



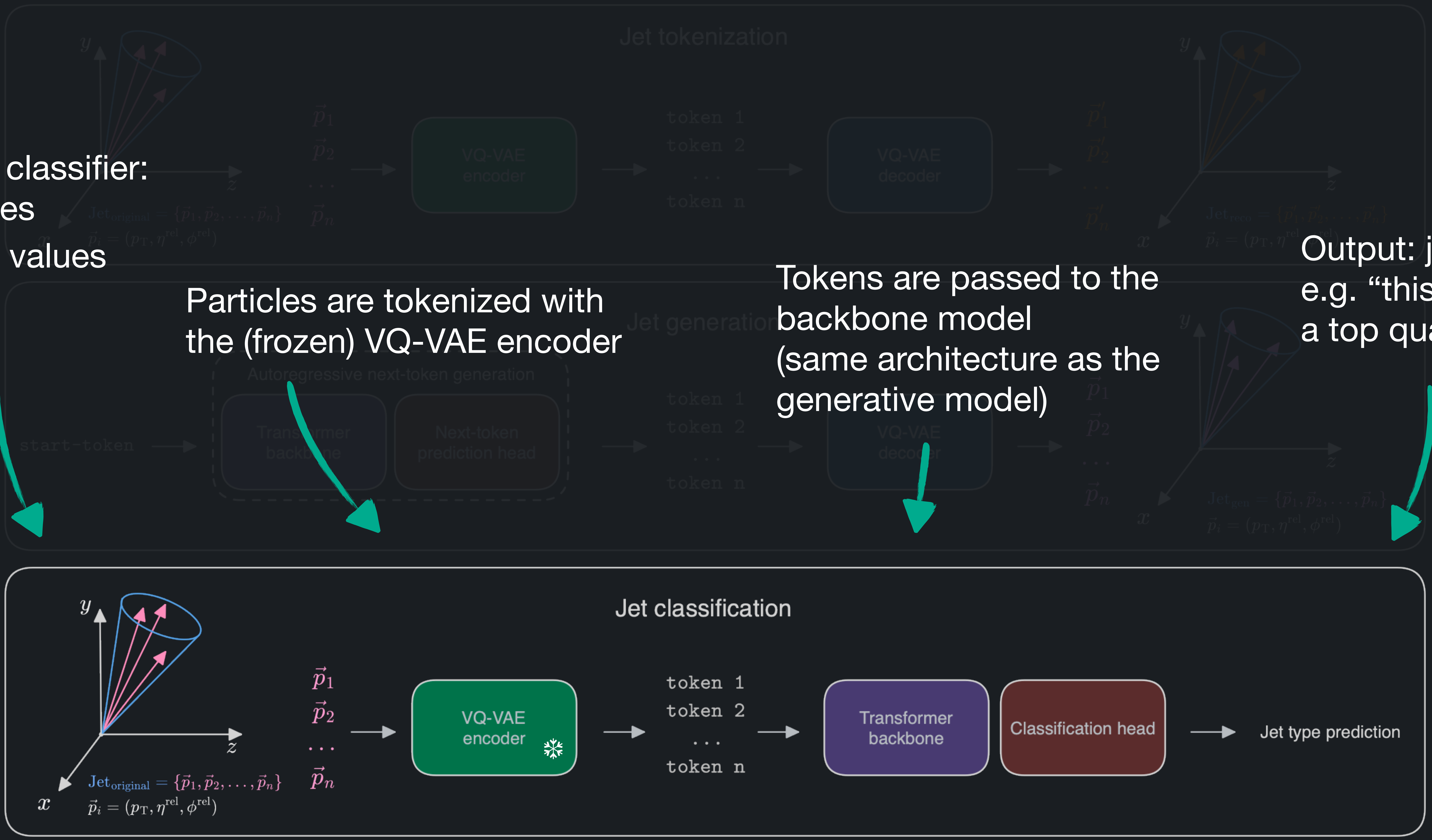
Model architecture overview - classification

Input for the classifier:
set of particles
with p_T, η, ϕ values

Particles are tokenized with the (frozen) VQ-VAE encoder

Tokens are passed to the backbone model (same architecture as the generative model)

Output: jet type prediction e.g. “this jet corresponds to a top quark”

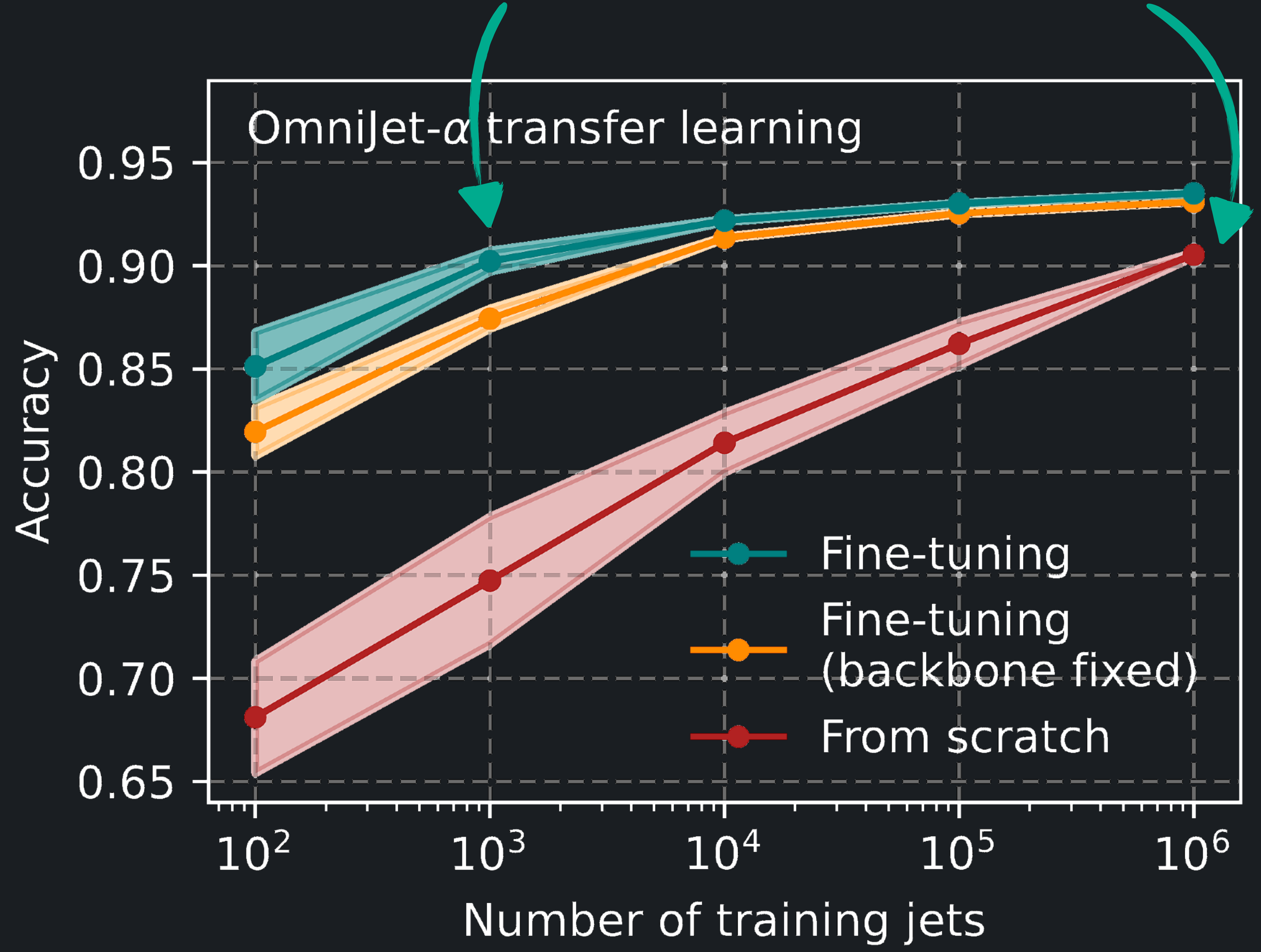


Transfer learning: classify q/g vs $t \to bqq'$

Does generative pre-training help for classification training?

- Generative pre-training on 20M jets
- Backbone architecture unchanged
- Swap the model head
- Train classifier with different strategies
    - **Fine-tuning**:
        - Use backbone weights from gen. model
        - Class. head randomly initialized
        - All weights allowed to change
    - **Fine-tuning (backbone-fixed)**:
        - Initial weights as above
        - Only class. head weights can change
    - **From scratch**:
        - All weights are randomly initialized

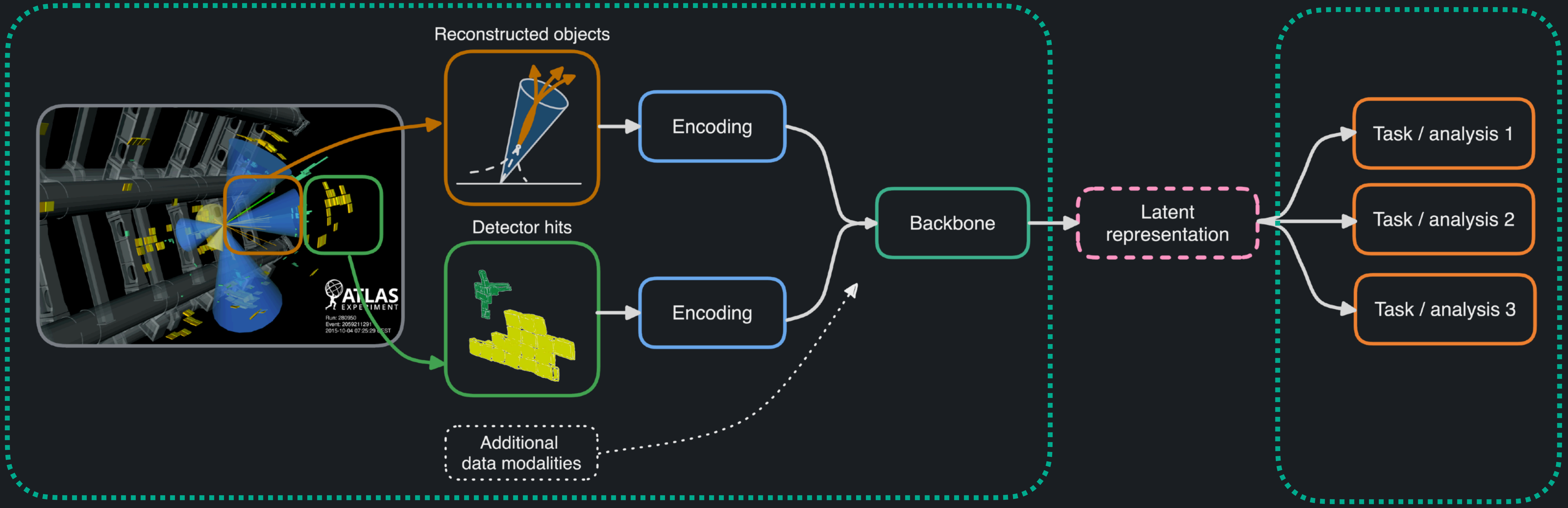
Fine-tuned model achieves 90% accuracy with only 1000 training jets, for which the "From scratch" model needs 1M training jets



How could this help in searches for dark sector particles?

- Available **dataset size can be very small → fine-tuning of foundation models could be promising**
- Different models of dark showers lead to very **different phenomenologies**
  → **Useful to have pre-trained classifiers performs well for every model**
- **Pre-training on data**: promising to deal with effects that are poorly modelled in MC
- This concept should be extendable to event level

Pre-trained on a large dataset (data, MC or maybe even mixture of MC and data)   Fine-tuning to specific analysis



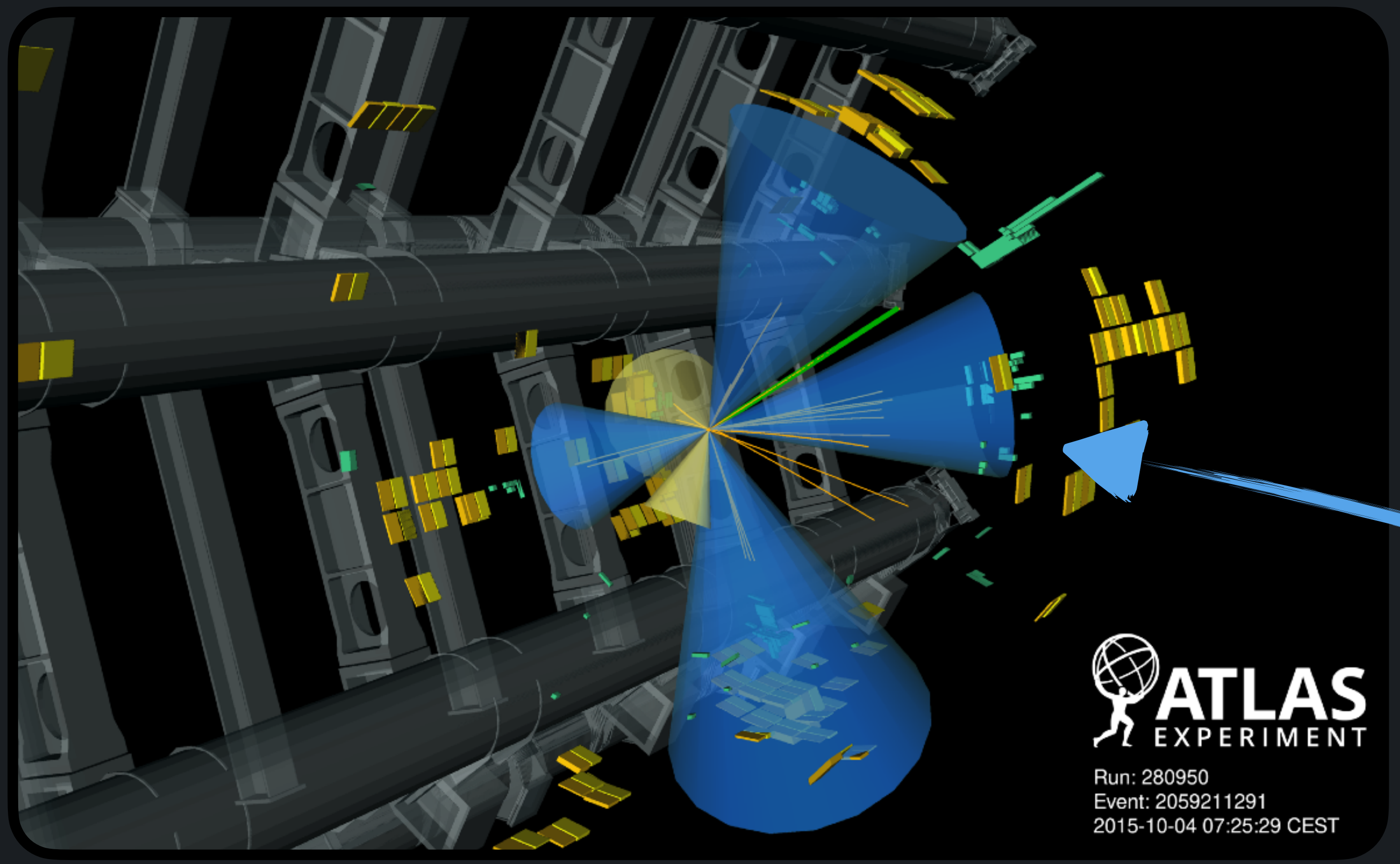
Summary

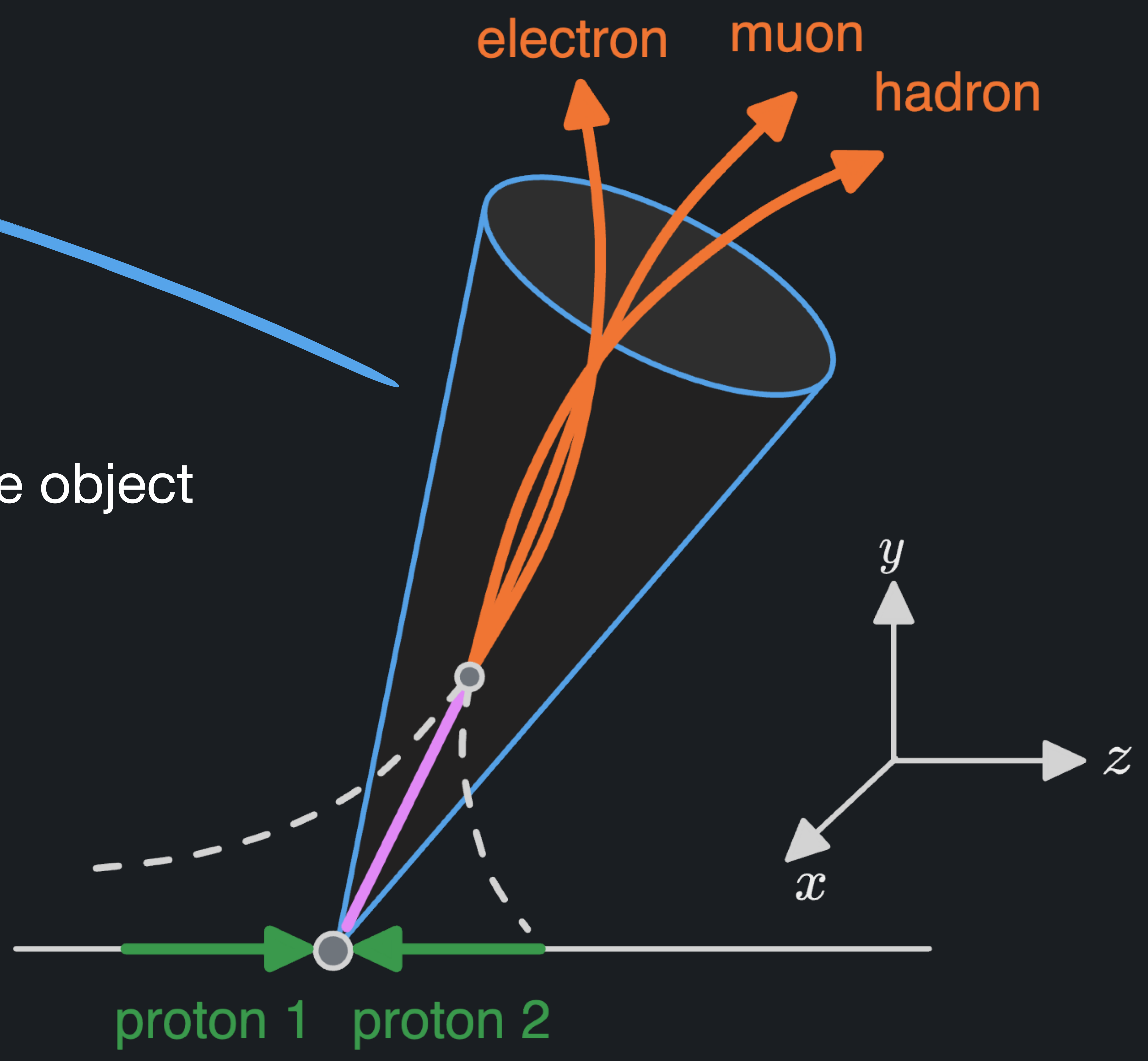
- **Foundation models getting more and more attention in HEP**
- Our work demonstrates the **first cross-task foundation model for jet physics**
- OmniJet-α is capable of **both generating and classifying** q/g and $t \rightarrow bqq'$ jets
- **Generative pre-training** leads to **significant improvements in classification** performance (especially for small dataset sizes)

- Promising concept to uncover the dark sector:
  - Unsupervised pre-training task: this **could be done on data directly**
  - **Small dataset can be sufficient to get good performance**

Thanks for listening!

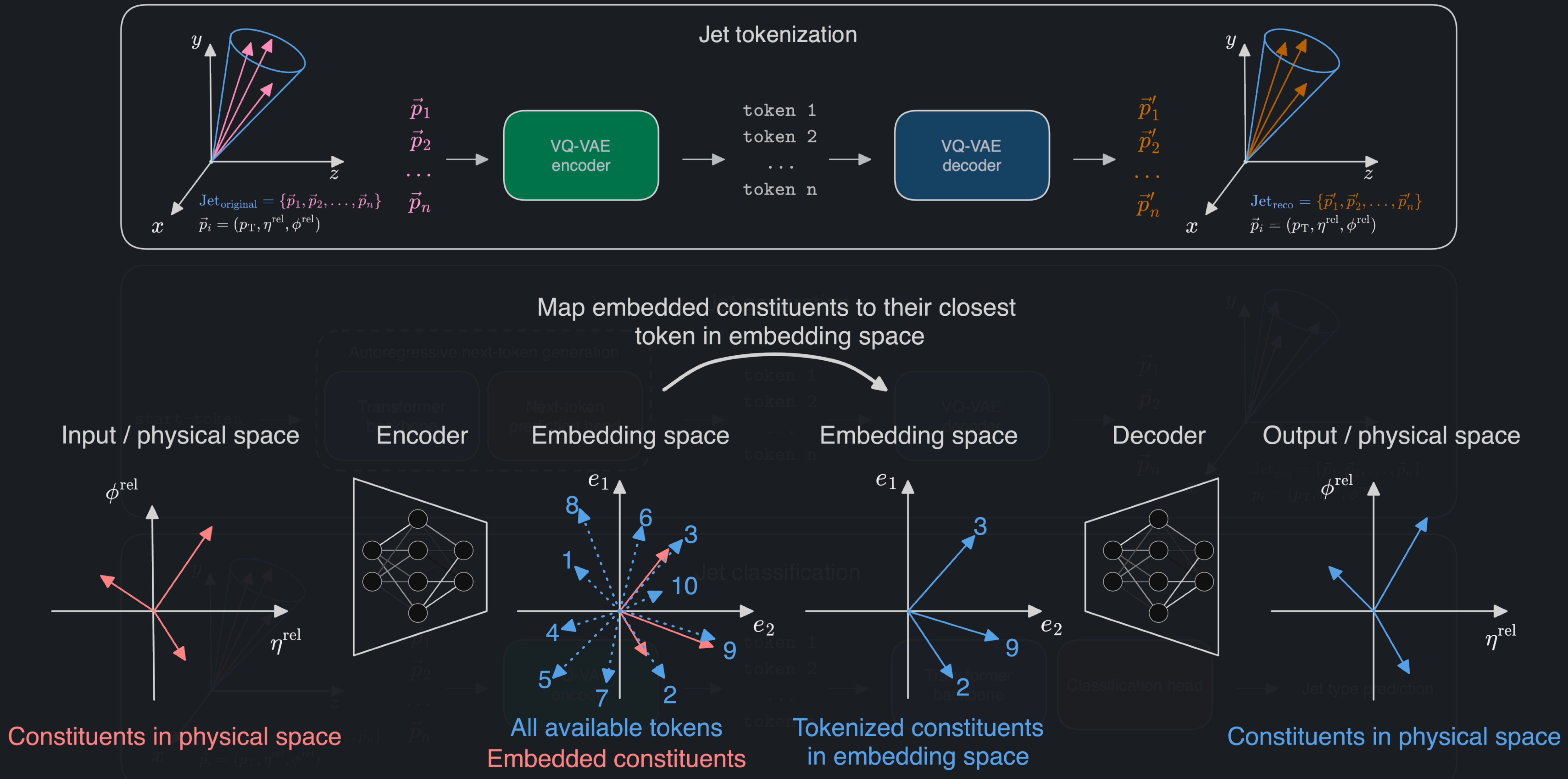
Dataset





- Particle jet: set of particles that are clustered to a cone-like object
- Important for most LHC analyses
- Most popular: Jet tagging
  (= identify the particle that “created” the jet)
- In the end: want to analyse full LHC events
- But: tasks on jet-level are also important & useful:
  - Jet generation for anomaly detection
  - Jet tagging
  - Reweighting/unfolding

Model architecture overview - tokenization

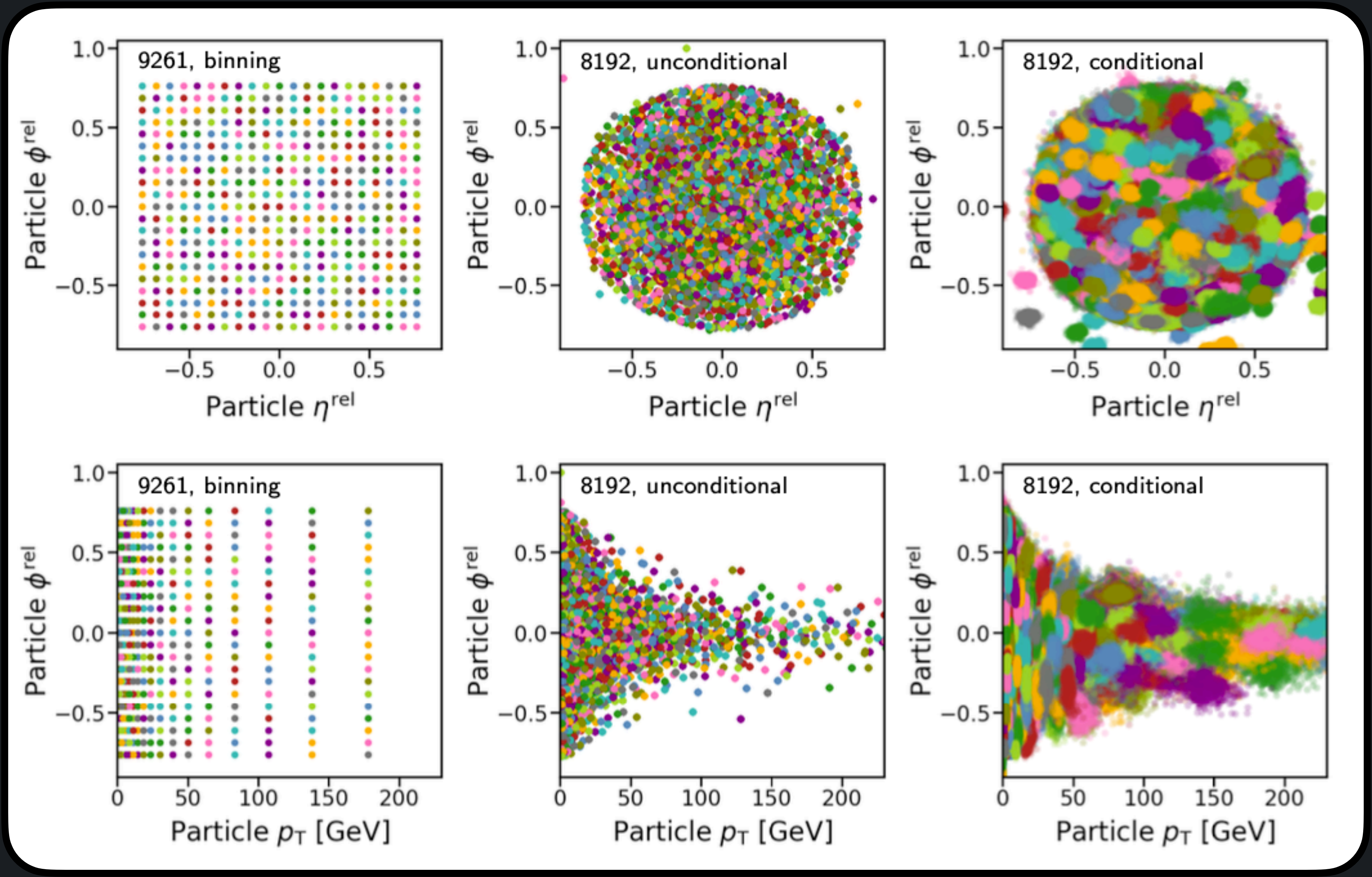


Tokenization resolution studies: 3 different approaches

Binning:
define bins in the 3-dimensional feature space, each bin/box is token

VQ-VAE unconditional:
VQ-VAE with an MLP for encoder/decoder

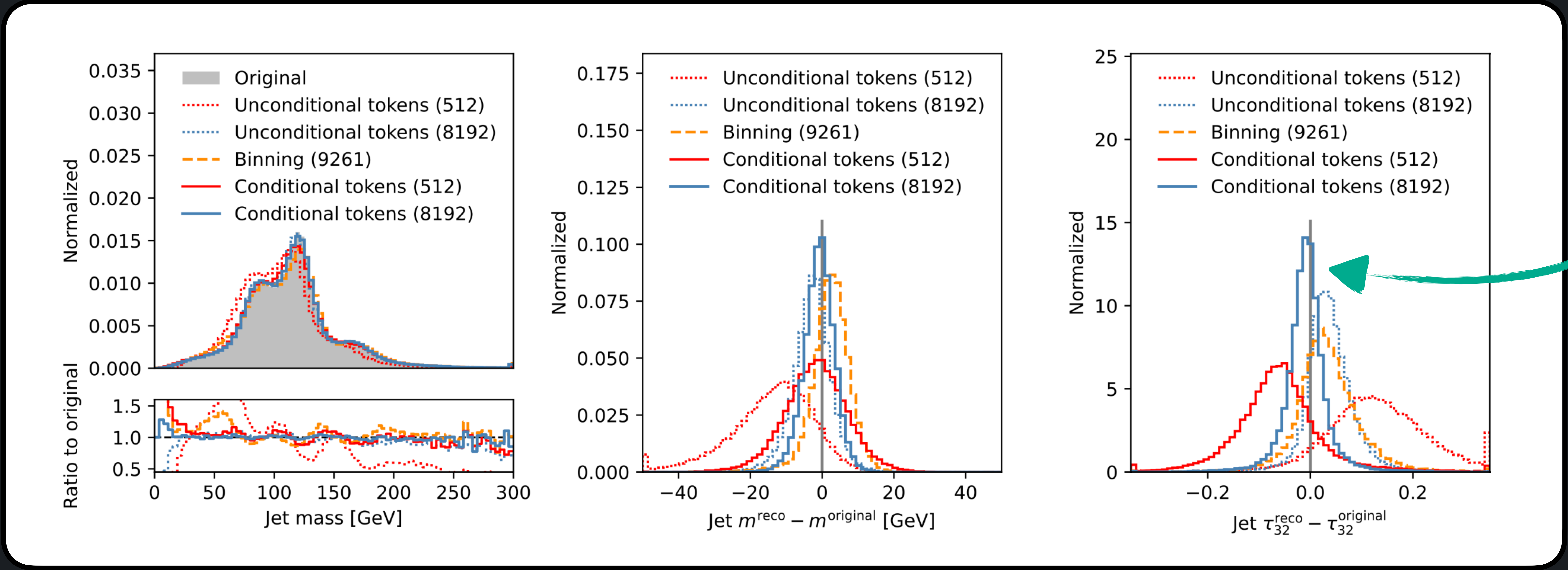
VQ-VAE conditional:
VQ-VAE with Transformer architecture for encoder/decoder



VQ-VAE conditional allows that a single token can be reconstructed to multiple physical values

Tokenization resolution studies

- Studied 3 tokenization approaches
  - Binning: define bins in the 3-dimensional feature space, each bin/box is token
  - VQ-VAE unconditional: use the VQ-VAE, but with an MLP for encoder/decoder
  - VQ-VAE conditional: VQ-VAE with Transformer architecture for encoder/decoder
- The final model used conditional tokens with codebook size 8192

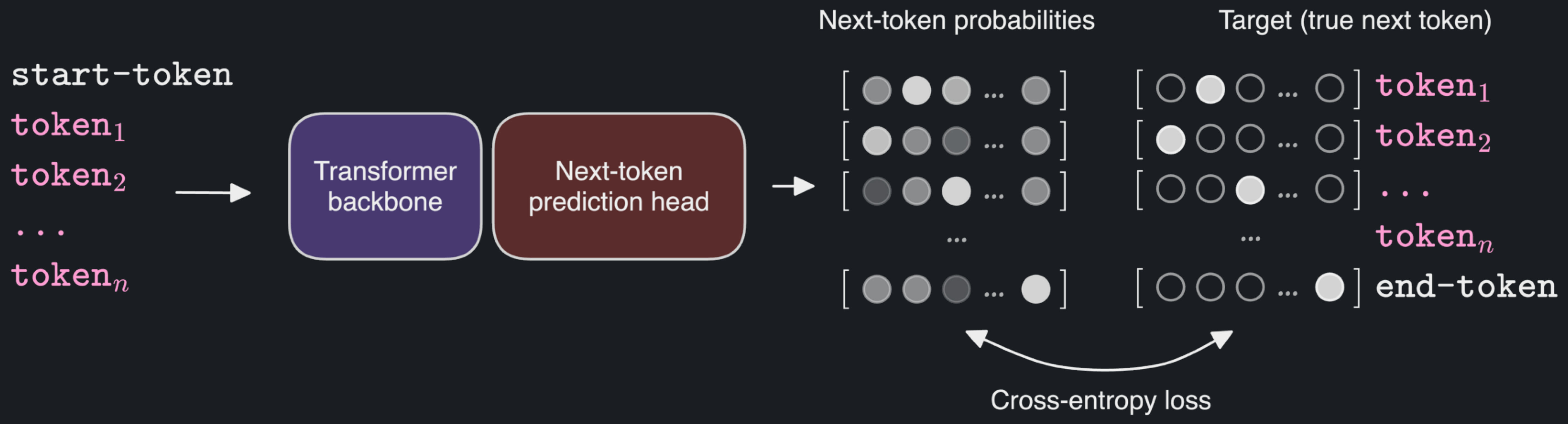


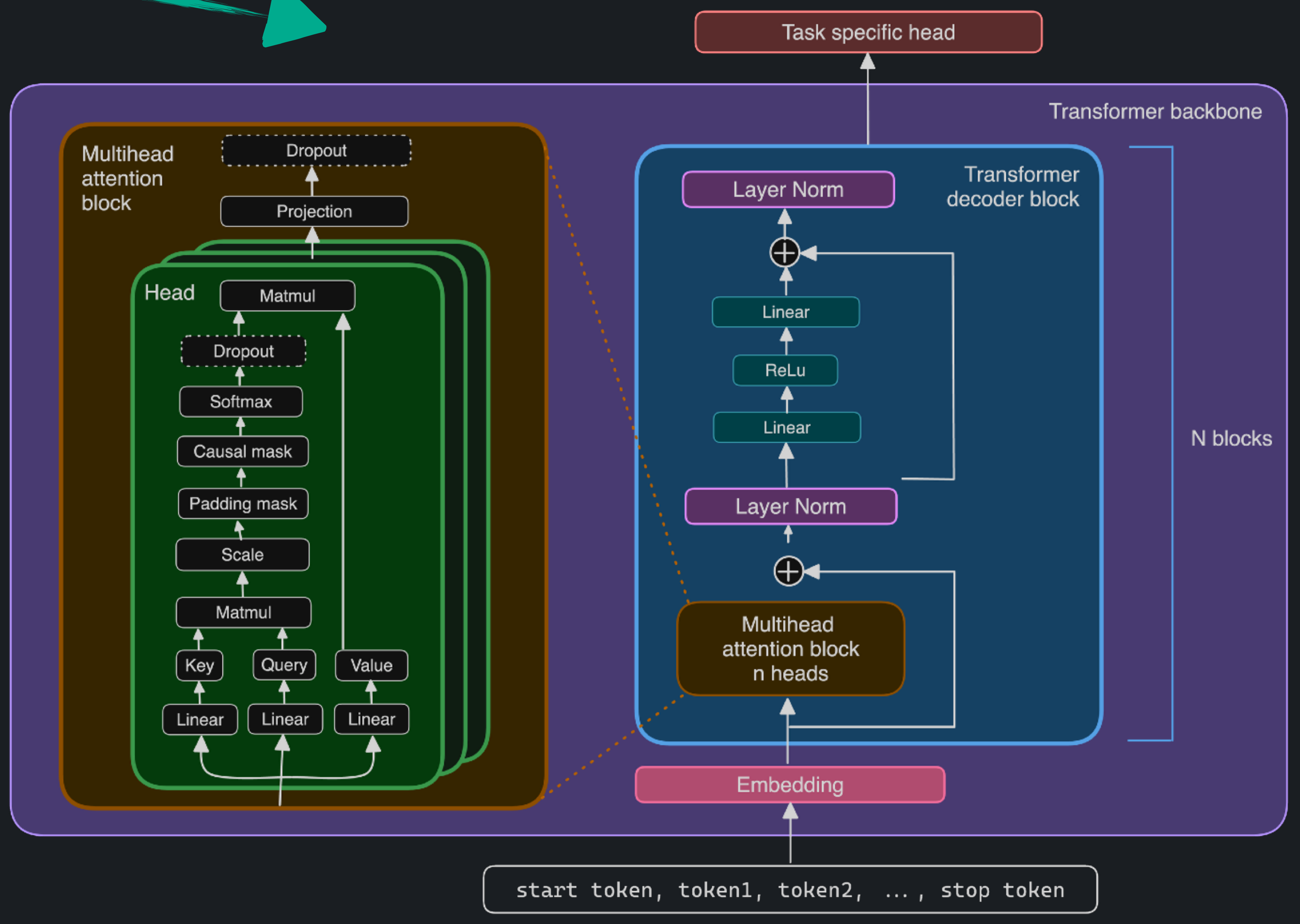
Encoding and decoding without loss of information would give delta peaks here

Generative (pre-)training

- Idea: while learning to generate, a model also learns aspects of the data useful for other tasks
- Advantage of this approach: our pre-training task is a useful task by itself
- Transformer architecture is commonly used in NLP
- We choose the original GPT-1 architecture [1]

Works like a language model, but generates particle tokens instead of word tokens



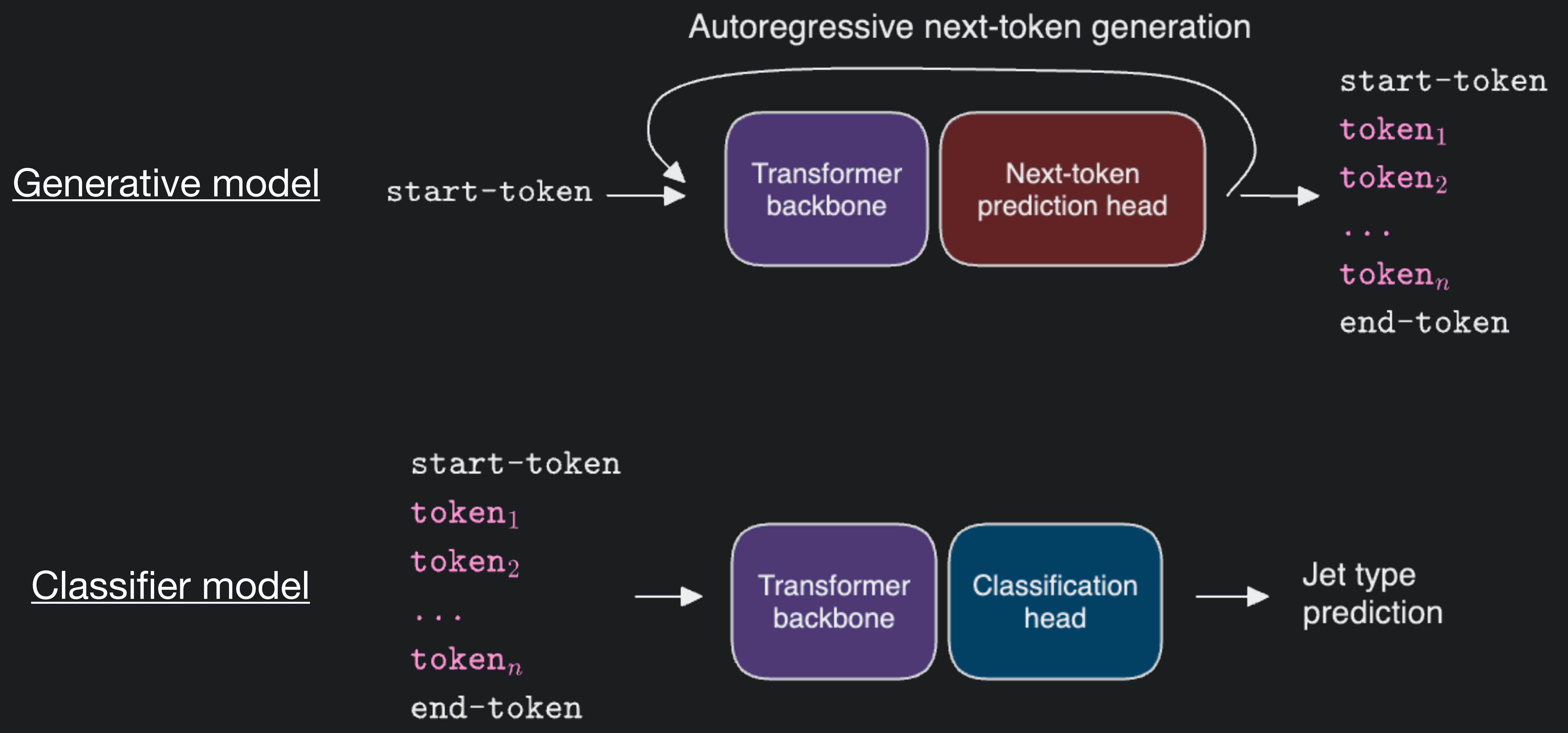


[1] Radford *et al*, "Improving language understanding by generative pre-training" (2018)

Transfer learning: classify q/g vs $t \rightarrow bqq'$

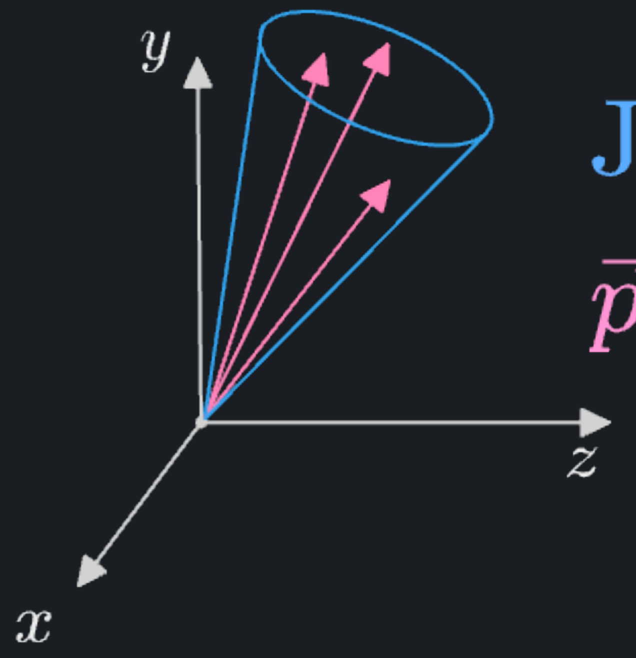
Does generative pre-training help for classification training?

- Backbone architecture unchanged
- Swap the model head
- Train classifier with different strategies
  - **Fine-tuning**:
    - Use backbone weights from gen. model
    - Class. head randomly initialized
    - All weights allowed to change
  - **Fine-tuning (backbone-fixed)**:
    - Initial weights as above
    - Only class. head weights can change
  - **From scratch**:
    - All weights are randomly initialized



Foundation model for jet tasks: our approach

Jet constituents with **continuous features**



$$\text{Jet} = \{\vec{p}_1, \vec{p}_2, \ldots, \vec{p}_n\}$$
$$\vec{p}_i = (p_\mathrm{T}, \eta^{\mathrm{rel}}, \phi^{\mathrm{rel}})$$

Constituents are tokenized with a Vector-Quantized-VAE
(Since we use a language-model like approach, where the data is represented as a sequence of integer tokens)

$$\text{Jet} = \{\texttt{start-token}, \texttt{token}_1, \ldots, \texttt{token}_n, \texttt{end-token}\}$$
$$\texttt{token}_i = \text{integer value} \in [1, \ldots, 8192]$$

Unsupervised pre-training of transformer backbone on generative task (next-token prediction)

Autoregressive next-token generation

$\texttt{start-token}$
$\texttt{token}_1$
$\texttt{token}_2$
$\ldots$
$\texttt{token}_n$
$\texttt{end-token}$

Fine-tuning to classification task:
Swap model head and copy over the weights from the pre-trained backbone

$\texttt{start-token}$
$\texttt{token}_1$
$\texttt{token}_2$
$\ldots$
$\texttt{token}_n$
$\texttt{end-token}$