OmniJet-α and beyond: foundation model updates

ML4jets
2024-11-07

Anna Hallin

anna.hallin@uni-hamburg.de

Foundation models

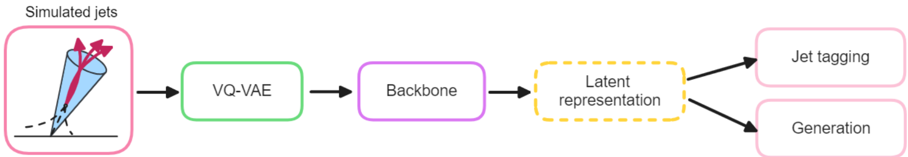
- Foundation models **pre-train** on a certain (large) dataset for a certain task, **fine-tune** to perform on a different dataset or a different task
- Foundation models may be expensive to train, but once pre-trained, downstream tasks require **less resources**
  - Human resources
  - Compute resources
- Can leverage the pretraining to **boost performance on small datasets**
- **Sharing** pre-trained models can provide others with access to resources that are normally not accessible for them (data, computing resources)

OmniJet-α

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

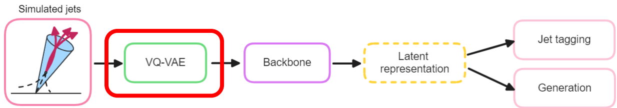
Mach. Learn.: Sci. Technol. 5 035031 (2024) (2403.05618); With Joschka Birk and Gregor Kasieczka

- OmniJet-α is the first foundation model for particle physics that is able to **task-switch**:
  - unsupervised **full jet generation**
  - supervised **classification**
- Uses a transformer for **generative pretraining** based on the GPT-1 architecture [1] with next-token-prediction as training target: $p(x_j|x_{j-1}, \dots, x_0)$
- **Idea:** as the model learns to generate jets, it learns aspects of the data that are useful for the downstream task.



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

Tokenization for generative tasks



- **Language models** need to **turn text into numbers** (which is what our models can work with), use tokenization: text → sequence of integer tokens
- In physics, we already have numbers, but our **architecture** or **training goals** can force us to **tokenize**:
  - Cross-entropy loss – powerful, but need discrete numbers = tokens
- Example of a particle jet:
  - $\text{Jet} = \{p_1, p_2, \dots, p_N\}$
  - $p_i = \{p_T, \eta, \phi, \text{PID}, \text{charge}, \dots\} \rightarrow \text{token}_i$
  - Jets as **sequences of integers**:
    $\{< \text{start token} >, \text{token}_1, \text{token}_2, \dots, \text{token}_\text{N}, < \text{stop token} >\}$

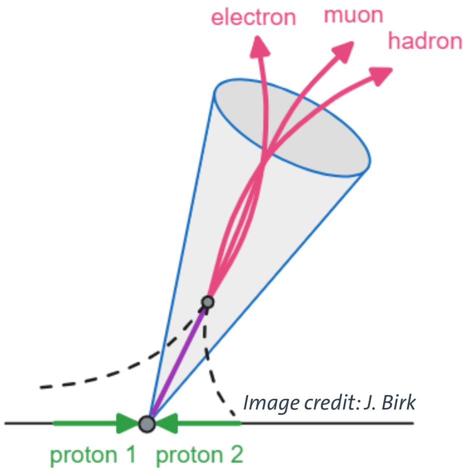
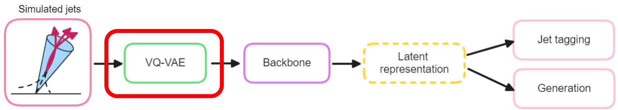


Image credit: J. Birk

Vector Quantized Variational autoencoder



The VQ-VAE [2,3] learns an **embedding space** of discrete tokens

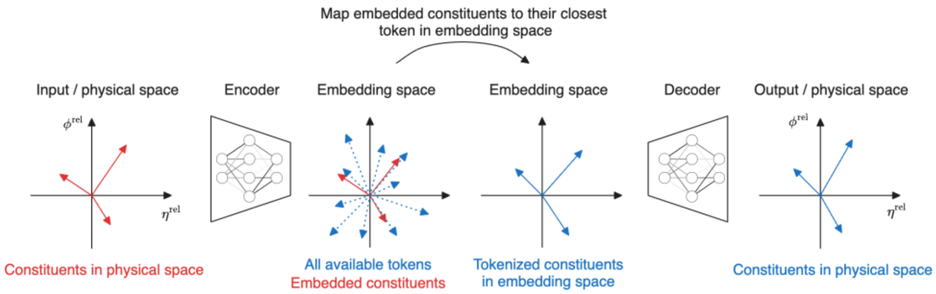
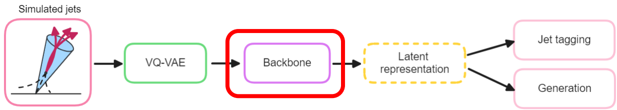


Image credit: J. Birk

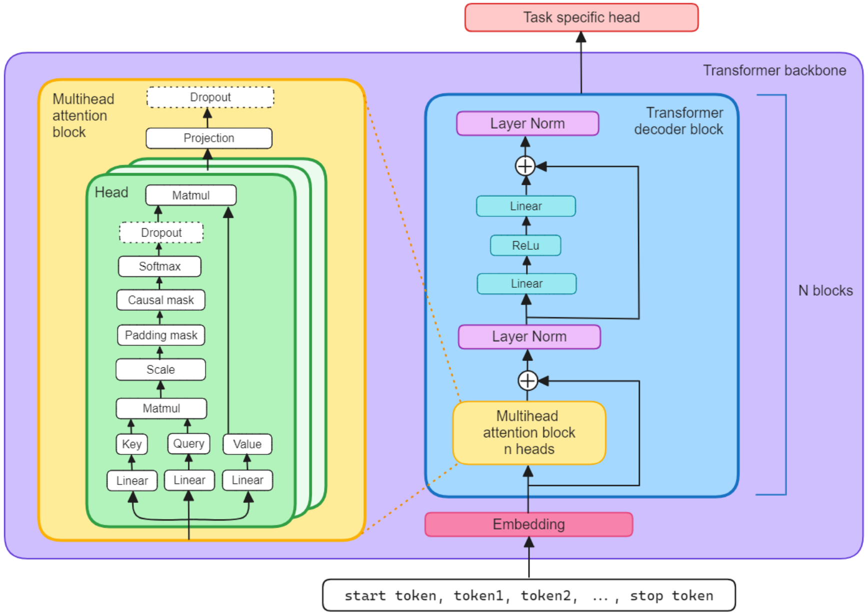
- Unconditional tokens: tokenize one constituent at a time, **1:1 correspondence**
- Conditional tokens: sees all constituents, adapts the tokens → one token can **cover multiple parts** of feature space

[2] van den Oord et al, *Neural Discrete Representation Learning*. arXiv 1711.00937
[3] Huh et al, *Straightening Out the Straight-Through Estimator: Overcoming Optimization Challenges in Vector Quantized Networks*. arXiv 2305.08842

Backbone



- **Transformer backbone** takes tokens as input, outputs to task specific head.
  `<start token>, token 1, …, token n, <stop token>`
- **Multihead attention block** receives a causal mask that prevents attention to future tokens and a padding mask to allow jets with different number of constituents
- **Task specific heads**
  - Generation – linear layer
  - Classification – linear layer, ReLU, sum, linear layer, softmax

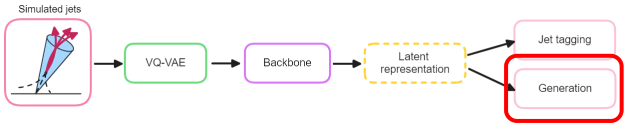


Dataset

- JetClass [4]: 10 classes of simulated jets with **10M jets of each type**
- For pretraining: use **10M *q/g*** jets and **10M $t \rightarrow bqq'$** jets.
- **No class labels** are passed to the model during pretraining.
- Use **constituent features** p_T, η^{rel}, φ^{rel} (rel = relative to the jet axis), no jet-level information, no PID etc

[4] http://dx.doi.org/10.5281/zenodo.6619767

Generation

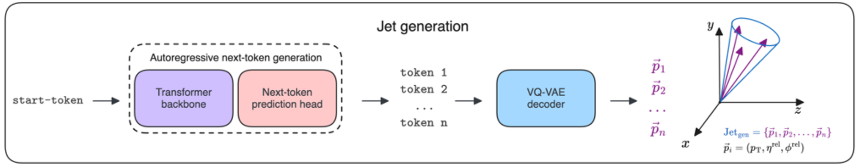


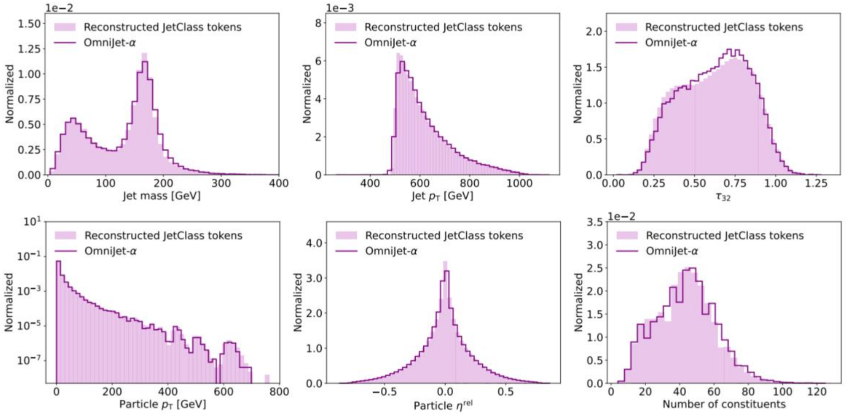
During generation, the model generates tokens **auto-regressively**:

- Model has learned $p(x_j|x_{j-1}, \dots, x_1, <\text{start token}>)$
- Model recieves <`start token`> and generates until it generates a <`stop token`> or the maximum sequence length is reached

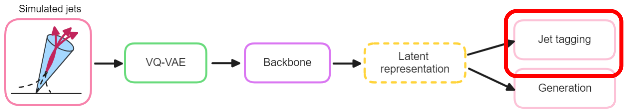
Generally **good agreement** to truth distribution

Constituent p_T spectrum tail has few events → the limited codebook size shows up as bumps





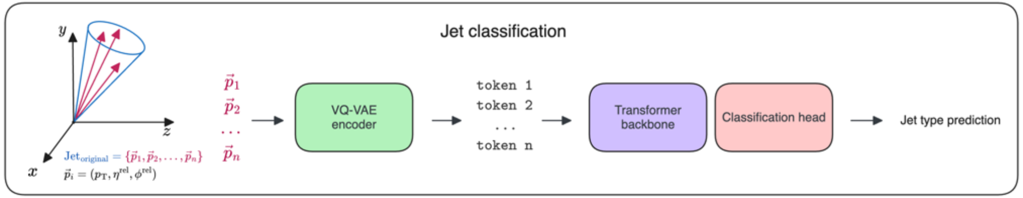
Transfer learning



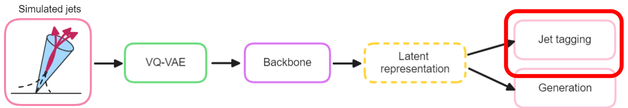
Task switching: classify quark/gluon vs hadronic top jets

The next-token-prediction head is changed to a classification head. Test three approaches:

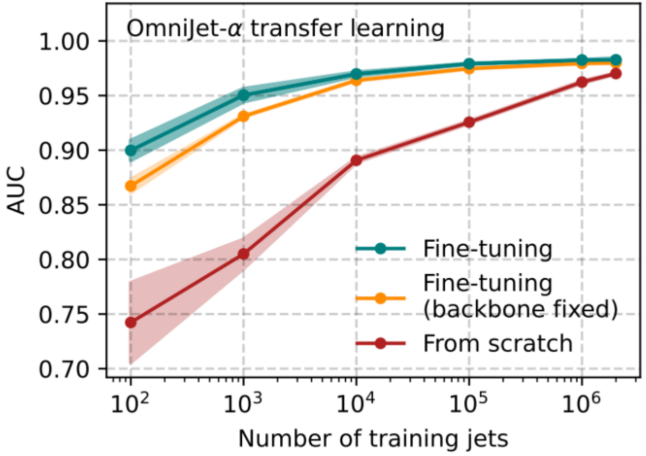
- **From scratch**: all weights are initialized from scratch, no pre-training is used
- Fine-tuning: load weights of the pre-trained generative model
    - regular **fine-tuning**: all weigths can change
    - **backbone fixed**: weights of the pre-trained transformer backbone are held fixed

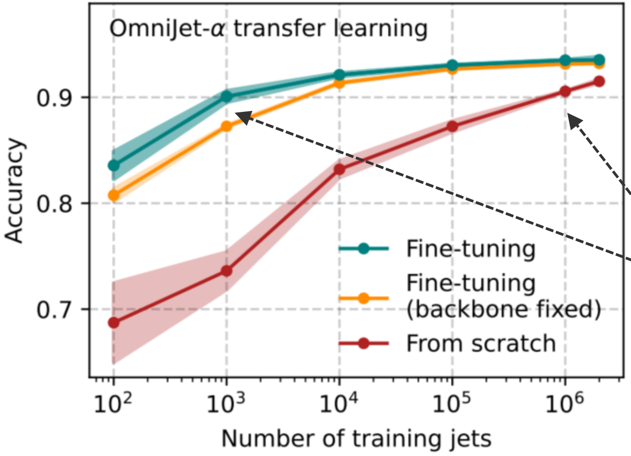


Transfer learning results



- Significantly better result when using pre-training
- Full fine-tuning slightly better than backbone fixed





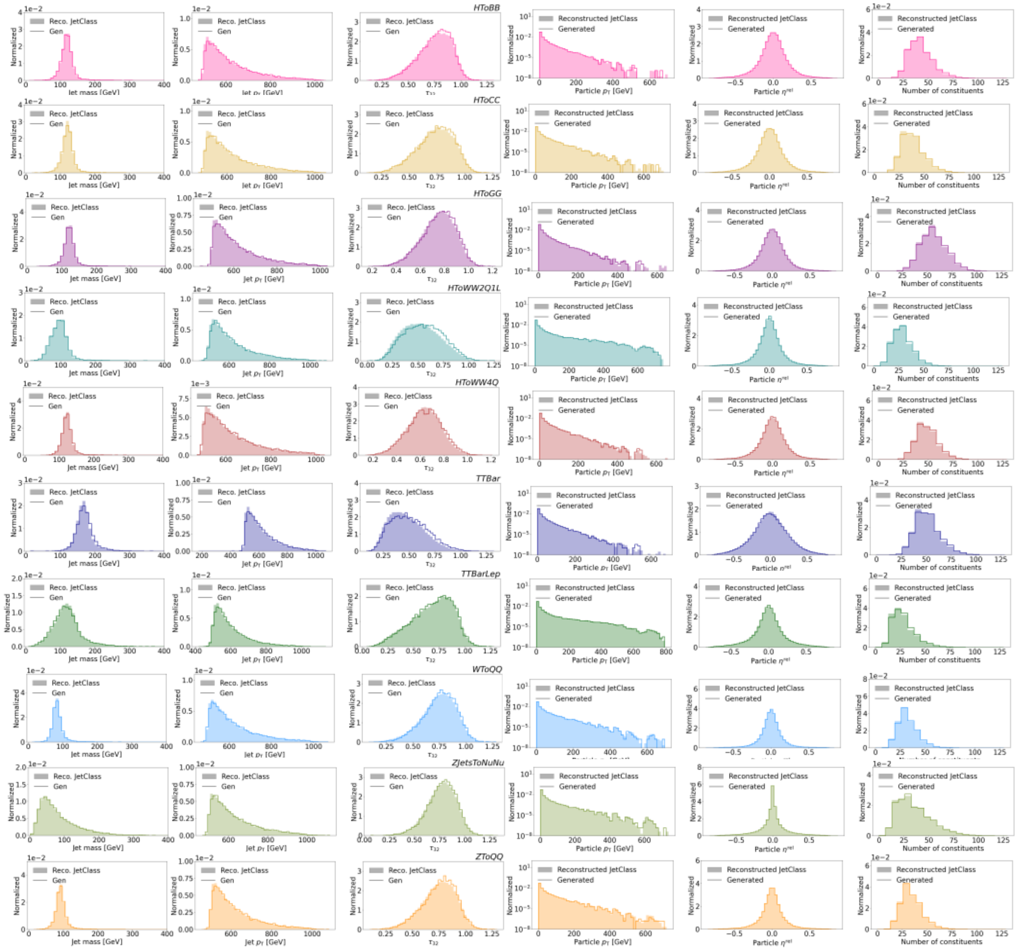
Pre-trained model requires only 1000 training jets to reach the same accuracy level that the "from scratch" model reaches with 1M jets.

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

OmniJet-α extensions

Conditional generation

- Requires labeled data
- Include class token when training:
  <start token>, <class token>, token 1, …, token n, <stop token>
- The model automatically learns to associate certain jets with certain classes
- Train on all 10 classes of JetClass
- Generate specific jet types by feeding the model the start token and class token
- Generally good agreement



Pre-training on real data, transfer learning to different jet type

With Oz Amram, Luca Anzalone, Joschka Birk, Darius A. Faroughy, Gregor Kasieczka, Michael Krämer, Ian Pang, Humberto Reyes-Gonzalez and David Shih
2411.XXXXX

- See Ian's talk on Monday!
- OmniJet-α pre-trains on unlabelled data, which we have a lot of
- Can pre-train on real data, fine-tune on simulations
- Aspen Open Jets
  - a dataset is derived from CMS Open Data 2016
  - contains 170M unlabelled jets
- We show that we can pre-train on AOJ, and then fine-tune on JetClass top jets, to generate more top jets

Introducing Aspen Open Jets: a real-world ML-ready dataset for jet physics

Nov 4, 2024, 3:10 PM
20m
LPNHE, Paris, France

Speaker

Ian Pang

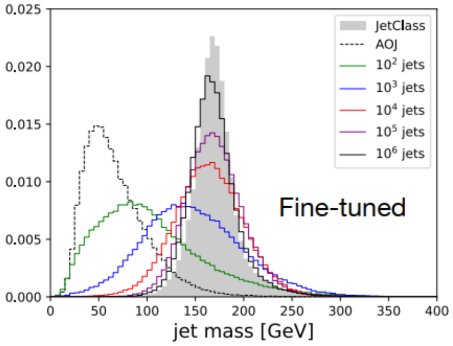


Image credit: Ian Pang

Using OmniJet-α with non-jet data: shower generation

With Joschka Birk, Gregor Kasieczka, Martina Mozzanica and Henning Rose
2411.XXXXX

- See [Henning's talk on Tuesday](#)!

Generative transformers for learning point-cloud simulations

Nov 5, 2024, 4:00 PM
20m
LPNHE, Paris, France

Speaker

Henning Rose

- The model only ever sees integers:
  - not dependent on being fed physics information
  - not restricted to jet physics
- A first step for building a foundation model for all particle physics must be to put tasks from different subdomains in the same computational framework
- Calorimeter shower generation:
  - Tokenize detector hits using the VQ-VAE
  - Train to generate point-cloud showers
  - Model learns how "long" a shower is, no need to condition on number of hits

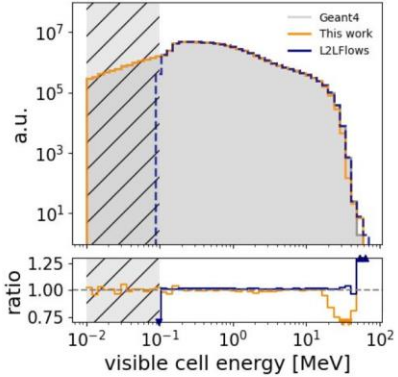


Image credit: Henning Rose

Outlook

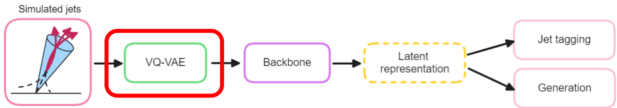
Conclusion and outlook

- OmniJet-α is the **first cross-task foundation model for particle physics**, capable of both **generating jets** and **classifying** them.
- Pre-training offers **significant improvements** in downstream task
- Current expansions:
  - Conditional jet generation
  - Pre-training on real data: Aspen Open Jets
  - Shower generation: a first step towards a multi-subdomain foundation model
  - More to come…

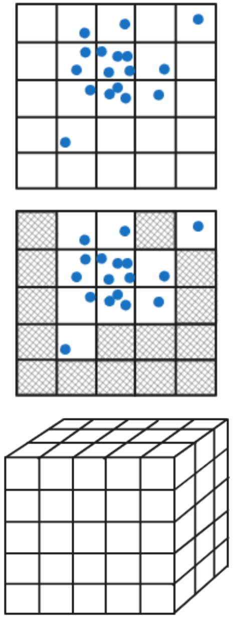
OmniJet-α references: Paper | Code

Backup

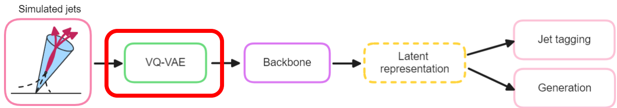
Binning



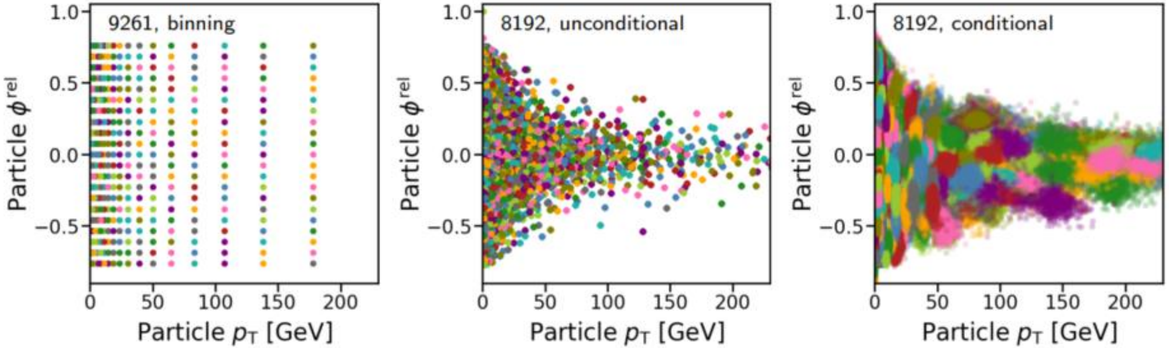
- Divide each dimension into bins
- Sub-optimal **coverage**
- **Vocab size** becomes $\prod_{i \in features} n_{bins,i}$
    - Tokens → Embedding: `Linear` $(n_{\text{tokens}}, d_{\text{embed}})$
    - Embedding → Tokens: `Linear` $(d_{\text{embed}}, n_{\text{tokens}})$
    - Example: 100 000 tokens with embedding dimension 128 → 25.6M parameters



Binning vs VQ-VAE



- VQ-VAE adapts to the shape of the data
- Conditional tokenization covers more of the phase space



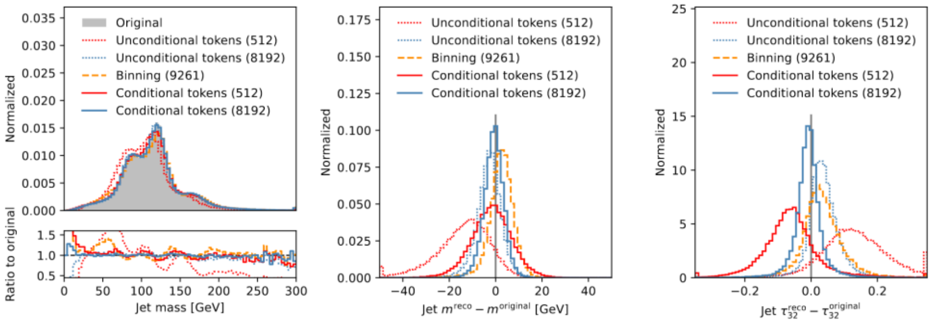
2403.05618

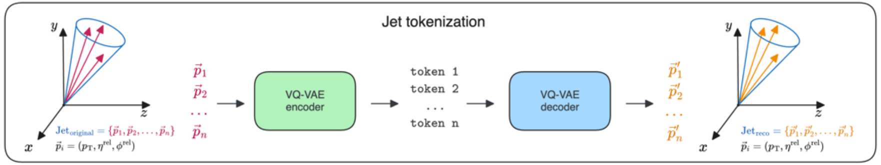
Tokenization results

Compared several approaches:

- Binning
- VQ-VAE
  - Unconditional
  - Conditional
  - Different codebook sizes (vocab sizes)

We proceed with **conditional tokens** with codebook size **8192**.

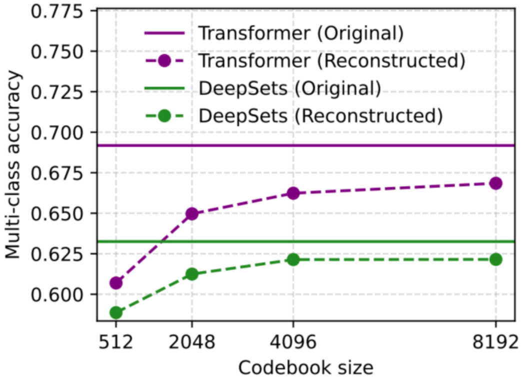




Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

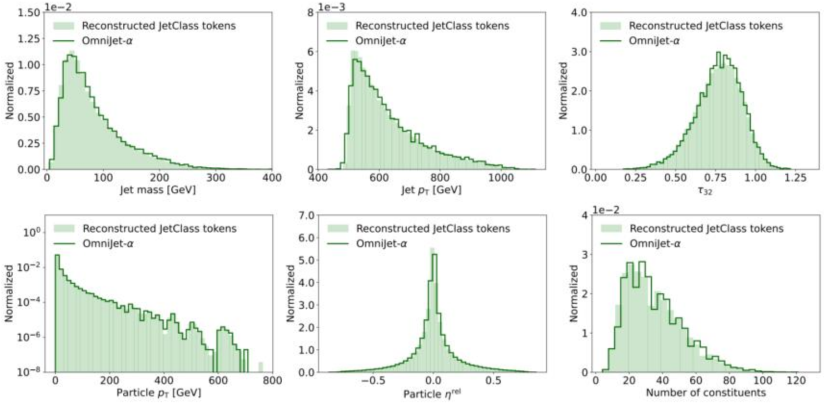
Quantifying tokenization information loss in OmniJet-α

- Train a **multi-class classifier** on all 10 classes of JetClass (note: this is not a reconstructed vs truth test)
- Two types of classifiers are tested: **transformer** and **Deep sets**
- Train on original JetClass data to obtain an **upper limit**
- Accuracy starts **plateauing** at a codebook size of 8192

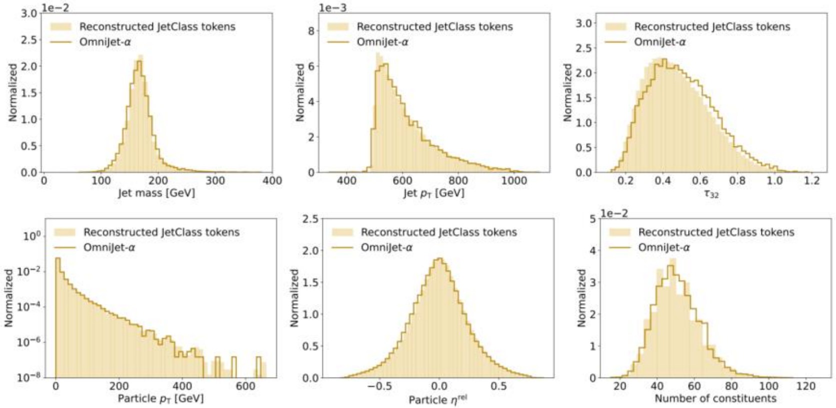


Generative results, single-jet type training

- *q/g* jets



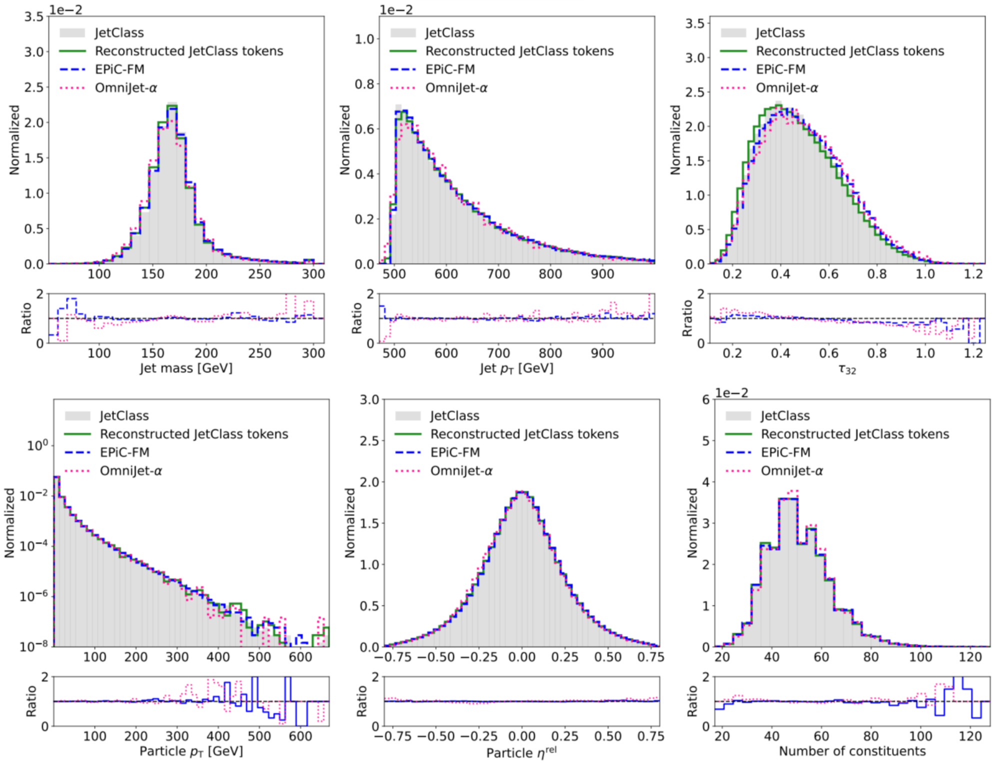
- $t \to bqq'$ jets



Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Comparison of generative capabilities, $t \to bqq'$

- EPiC-FM [5]: flow matching, **no tokenization**
- Ratios compare OmniJet-α and EPiC-FM (kinematics version) to **their respective truths**
- **Both** models are **doing well**
- OmniJet-α has a slightly higher discrepancy in the tails, except for constituent η^{rel} and number of constituents



[5] Birk et al, *Flow Matching Beyond Kinematics: Generating Jets with Particle-ID and Trajectory Displacement Information*. arXiv 2312.00123.