Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

CLUSTER OF EXCELLENCE
QUANTUM UNIVERSE

PUNCH 4NFDI

Foundation models for HEP

Leveraging the power behind large language models for physics

PUNCHLunch 2024.09.19

Anna Hallin

anna.hallin@uni-hamburg.de

Outline

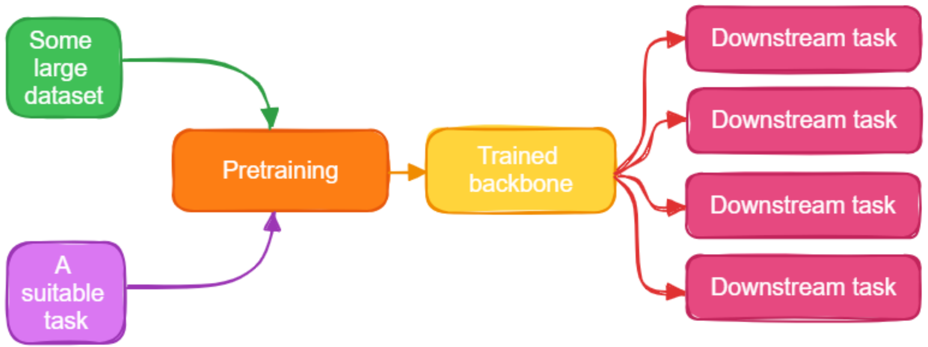
- Introduction to foundation models
- Foundation models in HEP
- A closer look at a foundation model for jet physics
- Outlook

Introduction to foundation models

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

What are foundation models?

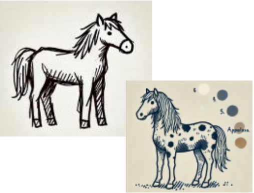
- **Pre-trained** on a certain (large) dataset for a certain task, **fine-tuned** to perform on a different dataset or a different task
- Better **performance** than training the downstream task from scratch



Why does it work?

- During pretraining, the model learns **aspects of the data** that are **useful** for downstream tasks
- The model has a “head start” compared to a model that needs to train from scratch

Pretraining



”Draw some of these animals”

Downstream task

”Which one of these is a horse?”

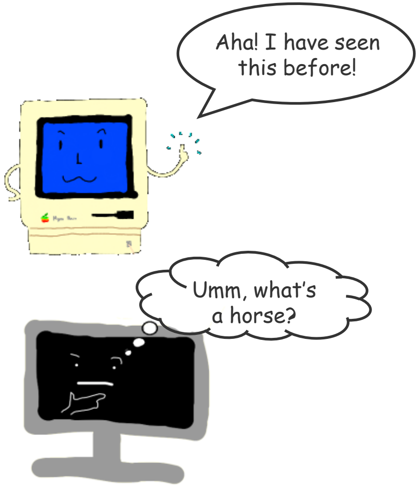


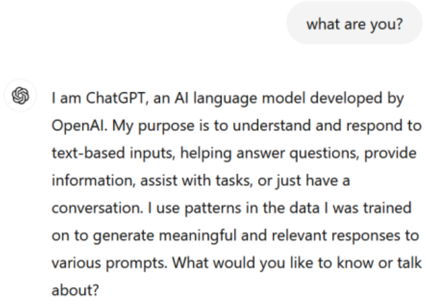
Image credits:
DALL-E
themarketingblog.co.uk
drawception.com

Benefits

- 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)

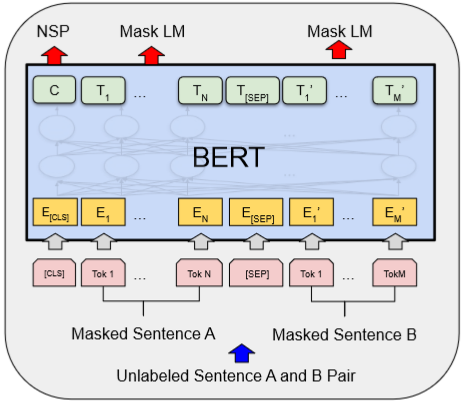
Examples of foundation models

- GPT-3 (arXiv 2005.14165)
    - Input: text
    - Pretraining: generate text (transformer)
    - **Finetuning**: conversational data + reinforcement learning with human feedback → ChatGPT
- CLIP (arXiv 2103.00020)
    - Input: text and images
    - Pretraining: match images with descriptions (transformer for text, ResNet/ViT for images)
    - **Zero shot**: image classification
- Note: a transformer in itself is not a foundation model



Pretraining

- Can be useful in itself, or a **surrogate task**
- Example of surrogate tasks: BERT (arXiv 1810.04805)
  - **Masked language modeling** in addition to **next sentence prediction**
  - Masking out tokens allows bidirectional training: sees both previous and future words in order to capture the context within a sentence
  - Next sentence prediction captures context between sentences: does sentence B follow sentence A?



Scale

Foundation models become powerful because of **scale**:

- **Data** amount
- **Architecture**
- **Compute**
- Example GPT-3: 300B tokens, 175 billion parameters, estimated thousands of GPUs trained over several weeks ($\sim 10^{23}$ flops)

In the context of language models (autoregressive transformers), empirical scaling laws [1] show that the cross-entropy loss improves with scale according to simple power laws.

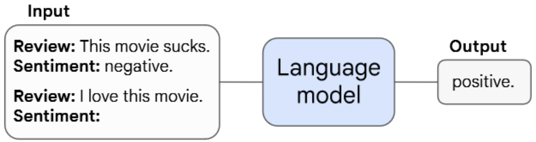
[1] Kaplan et al, *Scaling Laws for Neural Language Models*. arXiv 2001.08361

Emergent properties

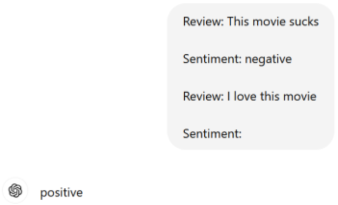
A foundation model might be able to perform tasks that it was **not trained for**, and that were not anticipated. This behavior comes with **scale** [2].

Examples from GPT-3 and BERT:

- Translation
- Coding
- Basic arithmetic
- Sentiment analysis
- Few-shot and zero-shot learning



2206.07682



[2] Bommasani et al, *On the Opportunities and Risks of Foundation Models*. arXiv 2108.07258

Foundation models for HEP

Natural language vs physics

Text

- Characters, (sub)words, symbols...
- Order matters
- Meaning builds across many sentences

Physics

- (Mostly) continuous numbers
  - Single numbers
  - Sets of numbers (vectors, time series)
- Can be permutation invariant
- Some sets of numbers like 4-vectors carry special meaning
- Symmetries might be present

Two approaches to foundation models in physics

- Teach LLMs to do maths and physics
  - Symbolic maths (arXiv 1912.01412)
  - Number embedding in text (arXiv 2310.02989)
- Take inspiration from LLMs+others, build from scratch
  - The remainder of the talk will focus on this approach

A foundation model example

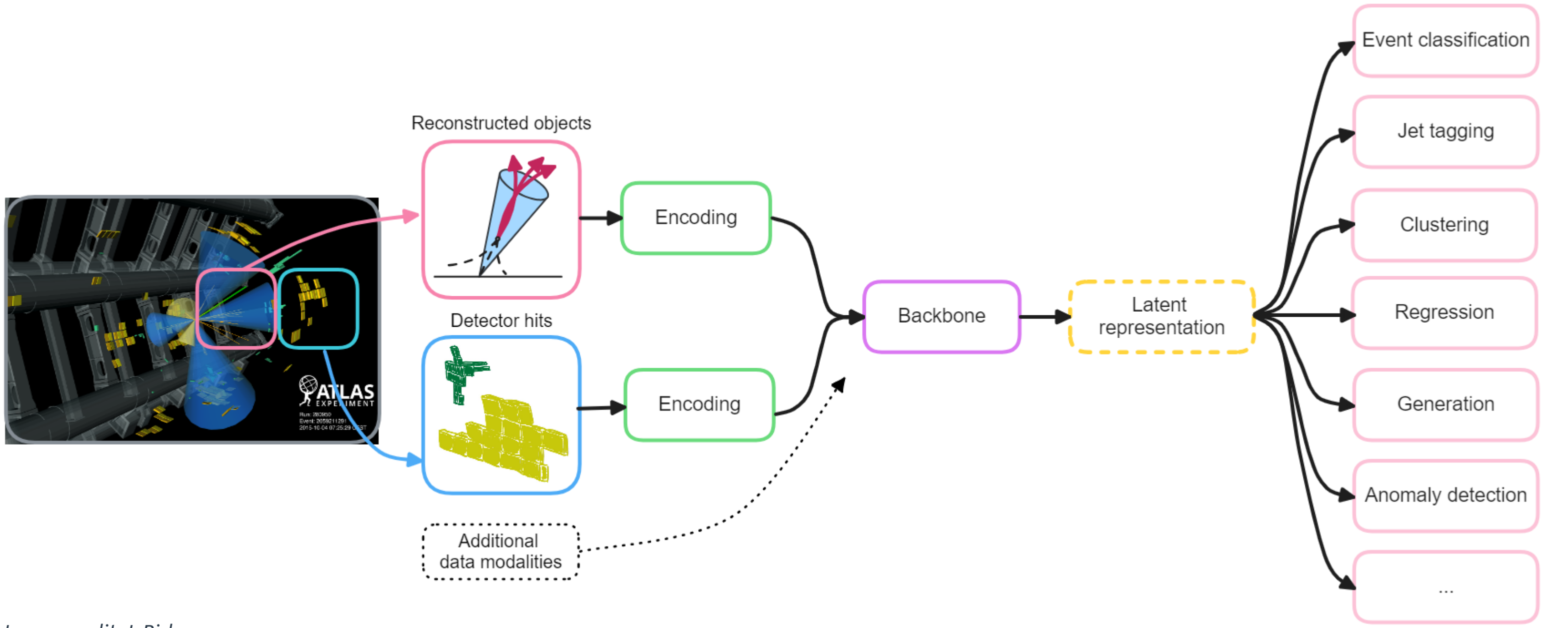


Image credit: J. Birk

A selection of foundation models for particle jets

- ParticleTransformer (ParT)
  - H. Qu, C. Li, S. Qian; arXiv 2202.03772
- Masked particle modeling (MPM)
  - T. Golling, L. Heinrich, M. Kagan, S. Klein, M. Leigh, M. Osadchy, J. A. Raine; arXiv 2401.13537
- OmniJet-α
  - J. Birk, **AH**, G. Kasieczka; arXiv 2403.05618
- OmniLearn
  - V. Mikuni, B. Nachman; arXiv 2404.16091

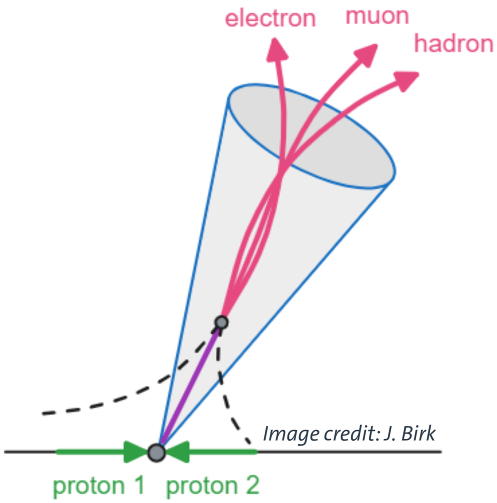


Image credit: J. Birk

Comparison of foundation models

Name	Pre-training goal	Architecture	Loss	Downstream tasks
ParT	Classification	Transformer	Cross-entropy class labels	Classification on different dataset

Comparison of foundation models

Name	Pre-training goal	Architecture	Loss	Downstream tasks
ParT	Classification	Transformer	Cross-entropy class labels	Classification on different dataset
MPM	Predict masked out tokens (surrogate task)	Transformer	Cross-entropy masked token prediction	Classification (tagging, anomaly detection)

Comparison of foundation models

Name	Pre-training goal	Architecture	Loss	Downstream tasks
ParT	Classification	Transformer	Cross-entropy class labels	Classification on different dataset
MPM	Predict masked out tokens (surrogate task)	Transformer	Cross-entropy masked token prediction	Classification (tagging, anomaly detection)
OmniJet-α	Next token prediction (generation)	Transformer	Cross-entropy next token prediction	Classification (tagging), Generation (unconditional)

Comparison of foundation models

Name	Pre-training goal	Architecture	Loss	Downstream tasks
ParT	Classification	Transformer	Cross-entropy class labels	Classification on different dataset
MPM	Predict masked out tokens (surrogate task)	Transformer	Cross-entropy masked token prediction	Classification (tagging, anomaly detection)
OmniJet-α	Next token prediction (generation)	Transformer	Cross-entropy next token prediction	Classification (tagging), Generation (unconditional)
OmniLearn	Generation + classification	Transformer + diffusion	Cross-entropy class labels + diffusion velocity parameter	Classification (tagging: different dataset, different experiment, different collision type; anomaly detection), Generation (conditional), Reweighting and unfolding

Tokenization

- LLMs need to turn text into numbers (which is what our models can work with), use tokenization: text → sequence of integer tokens
- In physics, to predict particle kinematics, as opposed to class labels:
  - Regression – so far no published results with this (seems to be more difficult)
  - Cross-entropy – need discrete numbers = tokens
- Example of a particle jet:
  - $\text{Jet} = \{p_1, p_2, \ldots, p_N\}$
  - $p_i = \{p_T, \eta, \phi, \text{PID}, \text{charge}, \ldots\} \rightarrow \text{token}_i$
  - Jets as sequences of integers:
    $\{<\text{start token}>, \text{token}_1, \text{token}_2, \ldots, \text{token}_\text{N}, <\text{stop token}>\}$

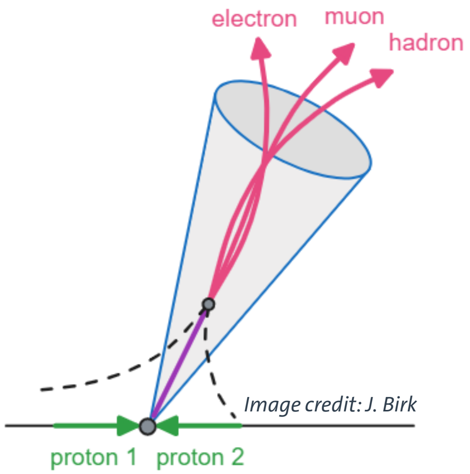
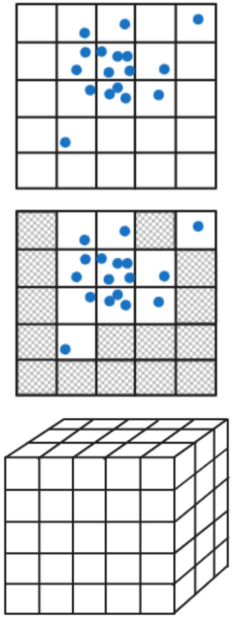


Image credit: J. Birk

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



VQ-VAE

1711.00937, 2305.08842

Learns an embedding space that gives the best reconstruction.

- 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

Vocab size is less sensitive to adding dimensions.

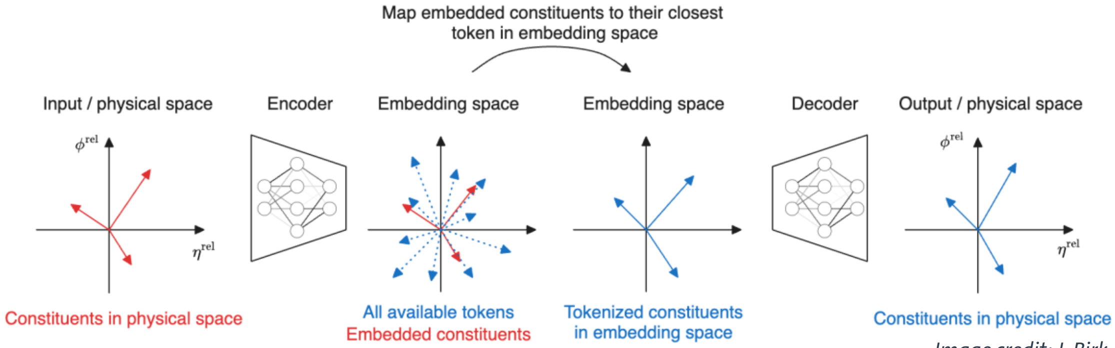
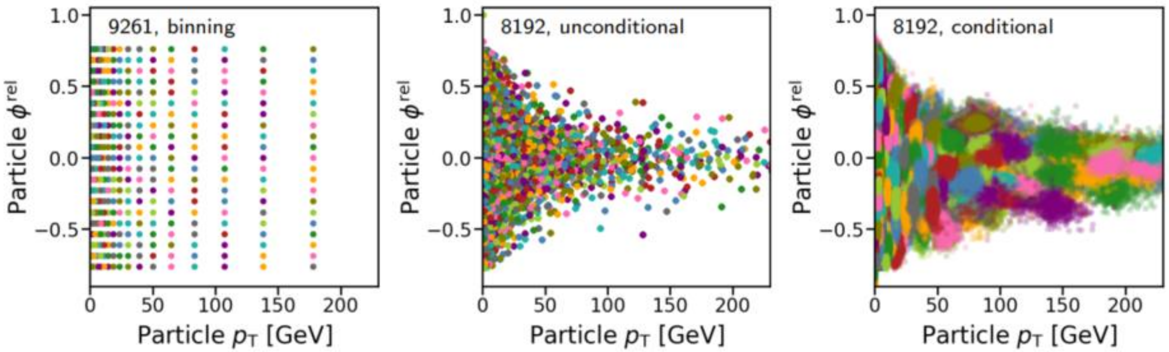


Image credit: J. Birk

Binning vs VQ-VAE

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

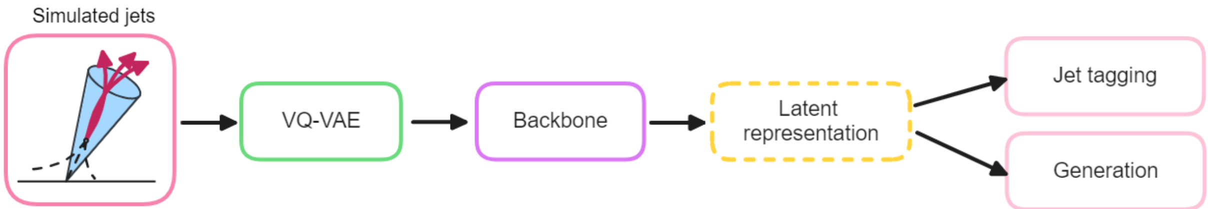


2403.05618

A closer look at OmniJet-α

A closer look at OmniJet-α

- OmniJet-α is the first foundation model for particle physics that is able to **task-switch**:
    - unsupervised **full jet generation**
    - supervised **classification**
- Tokenizes with **VQ-VAE**
- Uses a transformer for **generative pretraining** based on the GPT-1 architecture [3] with next-token-prediction as training target.



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

Dataset

- JetClass [4]: 10 classes of simulated jets with **10M jets of each type**, originally used in ParT
- **Tokenize all 10 classes** at once to evaluate tokenization performance
- For pretraining, generation and classification: 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

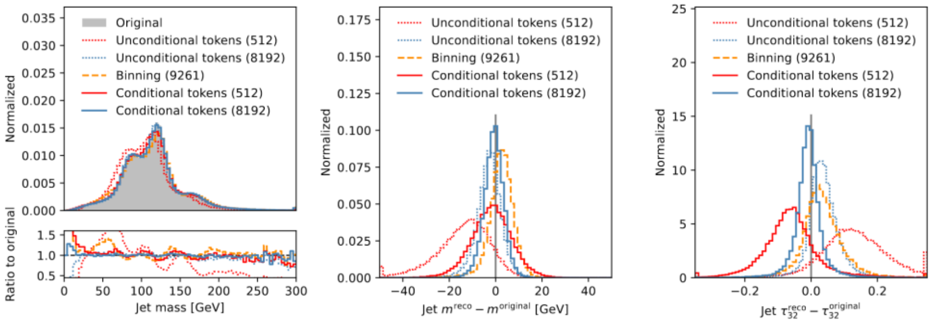
[4] http://dx.doi.org/10.5281/zenodo.6619767

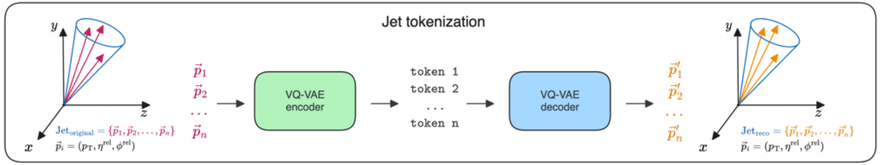
Tokenization

Compared several approaches:

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

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



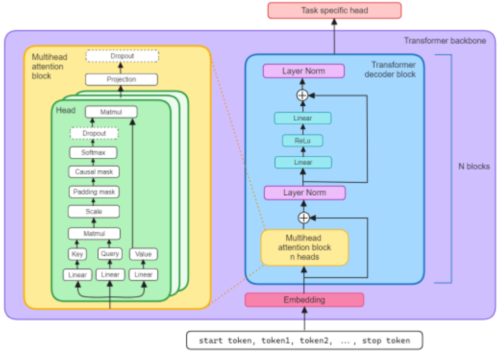


Backbone training

The transformer backbone is trained with the **next-token-prediction** head.

- **Causal mask** prevents attention to future tokens
- n heads = 8, N GPT blocks = 3 results in 6.7M parameters
- Model learns to predict the next token, given a sequence of previous tokens: $p(x_j|x_{j-1},\dots,x_1,<\textbf{start token}>)$

0				
0	153			
0	153	5489		
0	153	5489	51	
0	153	5489	51	8193



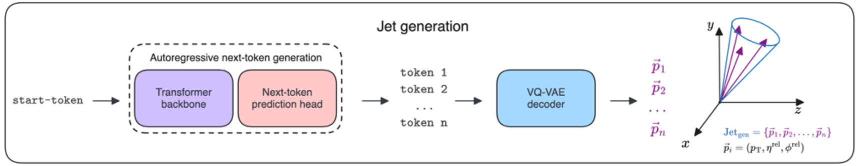
Generation

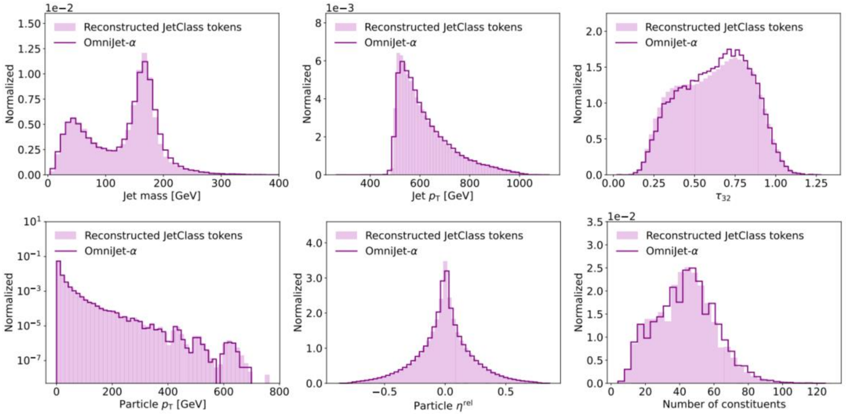
During generation, the model generates tokens **auto-regressively**:

- 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

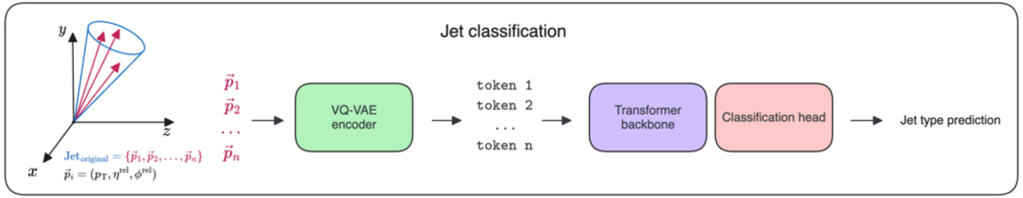




Transfer learning: classify quark/gluon vs hadronic top jets

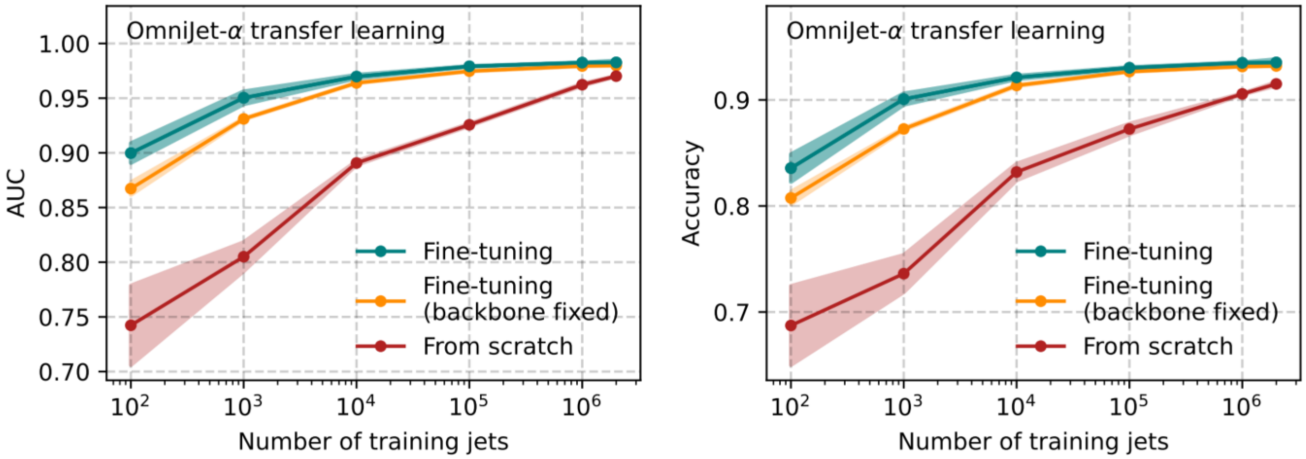
The next-token-prediction head is changed to a classification head. We tested three approaches:

- **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



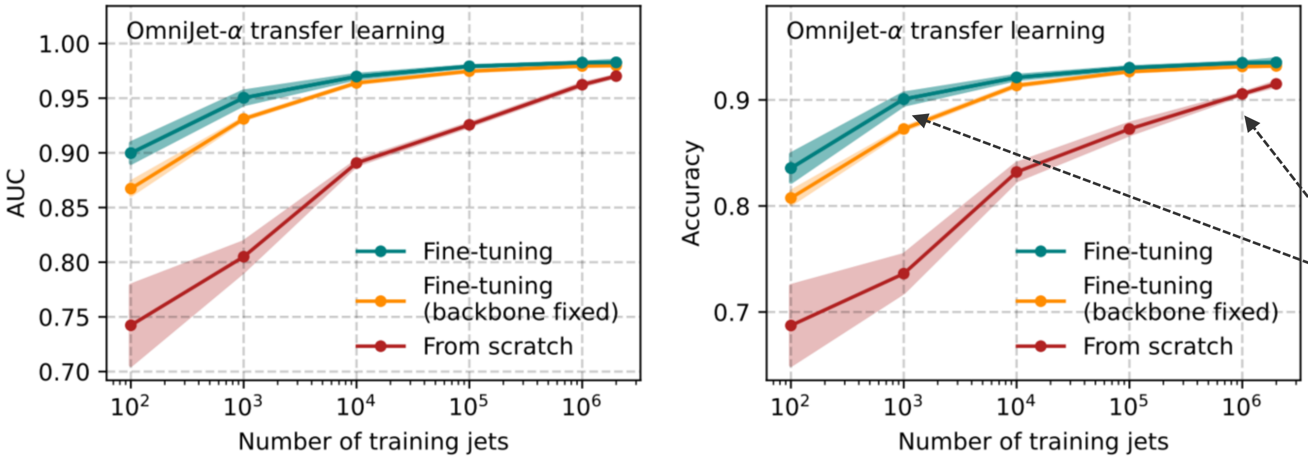
Transfer learning results

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



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 events to reach the same accuracy level that the "from scratch" model reaches with 1M events.

Outlook

Creating your first foundation model

- Downstream tasks
- Pretraining
  - Training goal
  - Architecture
  - Loss
  - Tokenization or not
  - Unsupervised, self-supervised, supervised...
- Input data
  - Multi-modal? Why and how?
  - Add physics info? Constraints, symmetries...

Conclusion and outlook

- Foundation models are **multi-task** and **multi-dataset** machine learning models that once **pretrained** can be applied to a variety of **downstream tasks**
- The successful development of foundation models for physics would be a **major breakthrough**, improving performance and saving human and compute resources
- Open questions:
  - What is the most efficient **representation** of the data?
  - How to introduce **multi-modal** data?
  - Exploring architectures and **pretraining strategies**
  - Expanding to further **downstream tasks**
  - Investigating effects of **scaling**