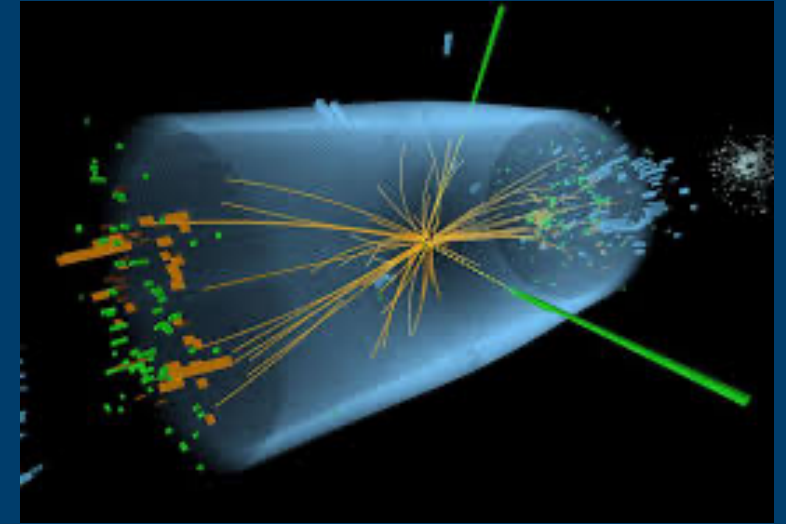


Foundation Models

for High Energy Physics



Lorenzo Moneta

CERN — EP-SFT

Lorenzo.Moneta@cern.ch



4th COMCHA School

Zaragoza, 8 – 15 April 2026

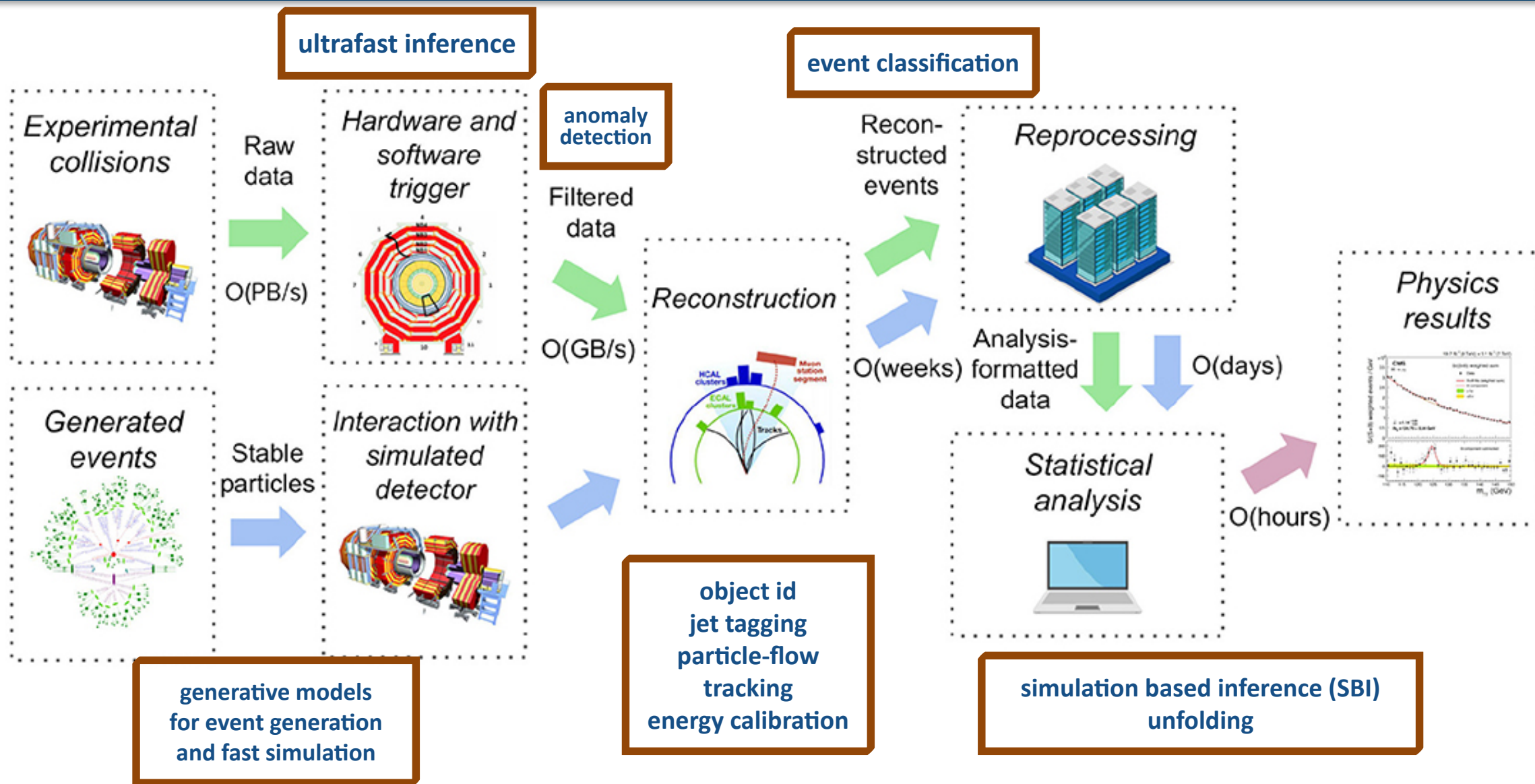


1. Motivation: the challenge of HEP data analysis
2. What are Foundation Models?
3. Transformers and Attention
4. HEP data: how to convert particles data to tokens
5. Some Foundation models in HEP with different pre-training strategies
 - A. Supervised classification (ParT)
 - B. Masked particle modeling (MPM)
 - C. Next-token prediction (OmniJet- α)
 - D. Contrastive learning (RS3L)
 - E. Hybrid approach (OmniLearn)
6. Foundation model for Fast Simulation of Calorimeter Showers
7. Towards Large Physics Models
8. Outlook and Open Questions

Motivation

The challenge of HEP data analysis at the LHC

AI in High Energy Physics Data Analysis (LHC)



AI tasks in LHC Data Processing

At the LHC, perform many different tasks on the collision data. Some of the major ones are:

- ▶ **Jet tagging:** classify quark / gluon / top / W / H jets
- ▶ **Anomaly detection:** search for new physics without specific signal model
- ▶ **Event classification:** select signal events from background
- ▶ **Generation / simulation:** produce realistic synthetic events
- ▶ **Unfolding:** correct measured distributions for detector effects
- ▶ **Regression:** reconstruct particle energies, momenta, vertices

Today: **each task → separate model**, separate training, separate resources
Can we train ONE model that serves as a foundation for ALL tasks?

This is the idea behind Foundation Models

Why is This Relevant Now?

- LHC experiments collect vast amounts of data
- High-quality simulated (labeled) datasets exist — unlike most other fields
- ML is already used at every stage: triggering, reconstruction, analysis
- The HL-LHC (≈ 2030) will multiply data volumes and computing needs
- Large amount of training data needed to fully exploit AI

The traditional approach — one model per task, trained from scratch — does not scale to the HL-LHC era.

Current approach

Train many separate models
Each needs labeled data
Each needs GPU training time
No real knowledge sharing between tasks

Foundation model approach

Pre-train ONE large model on vast data sets
Fine-tune model for each task
Share learned representations
Much less labeled data needed per fine-tune task

What are Foundation Models?

Definition, transfer learning, and the role of scale

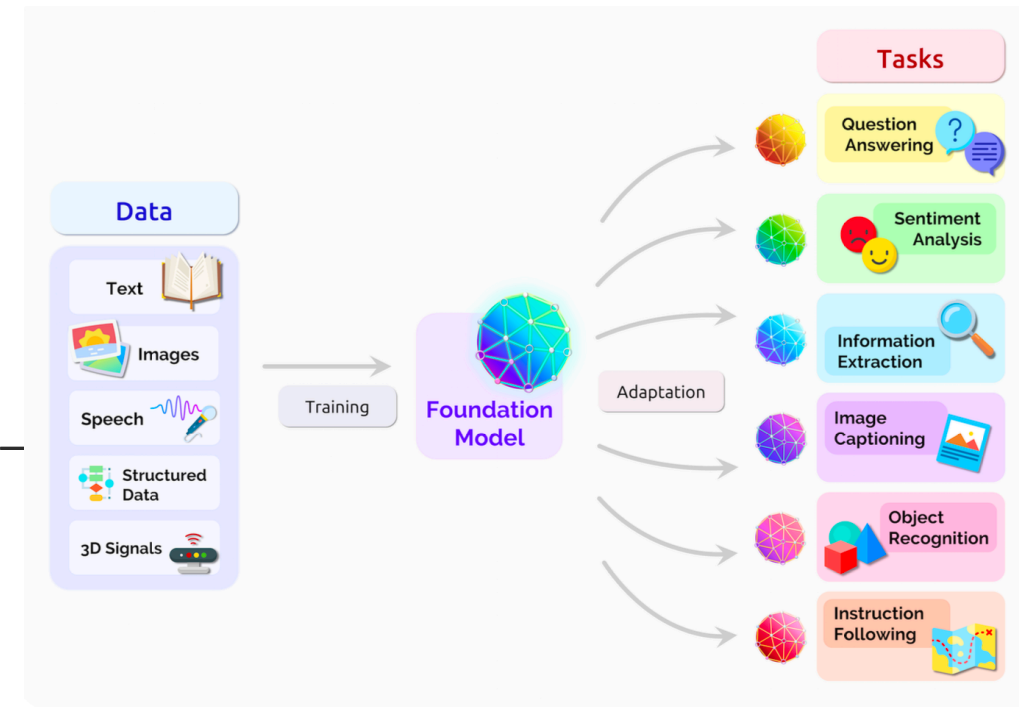
Definition

A foundation model is any model that is trained on broad data (generally using self-supervision at scale) that can be adapted (e.g., fine-tuned) to a wide range of downstream tasks.

— Bommasani et al. (2021), Stanford HAI [arXiv: 2108.07258]

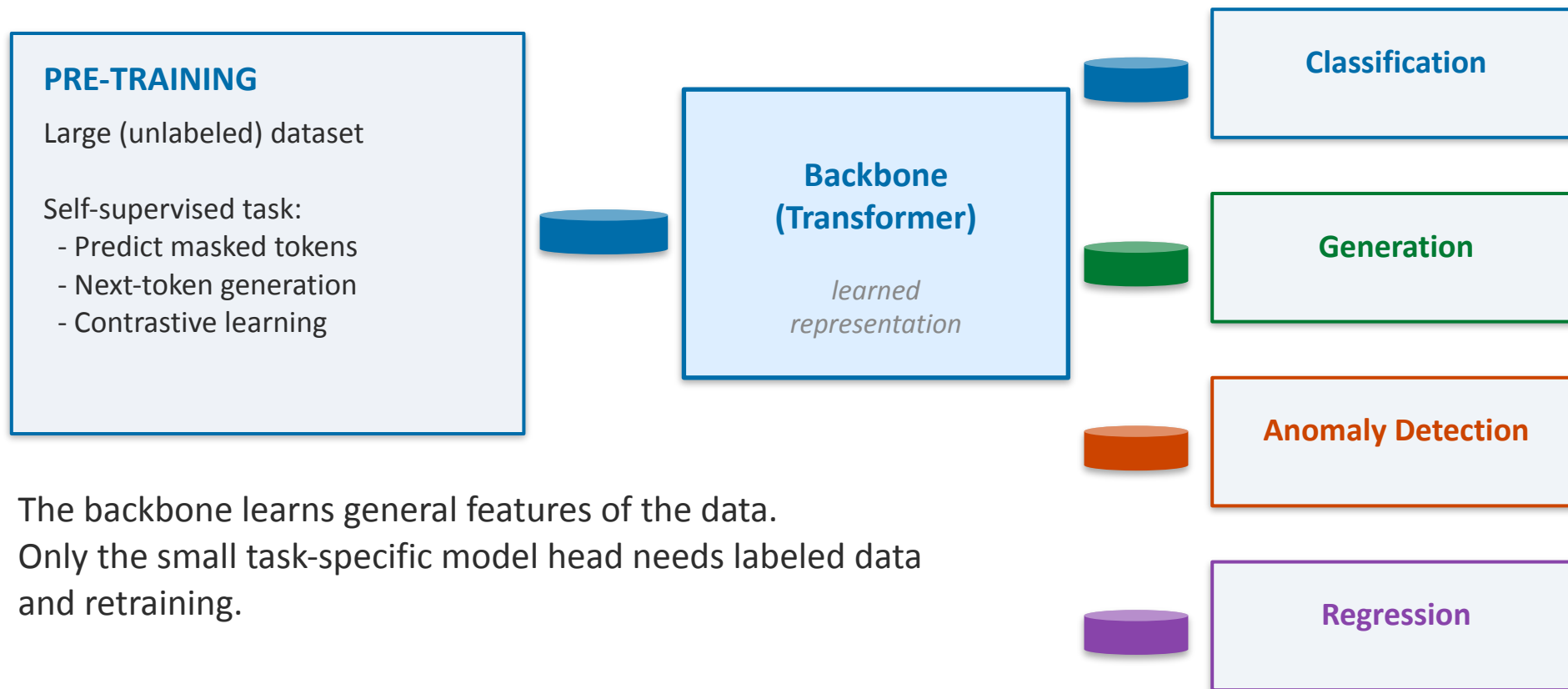
In practice, a foundation model must satisfy three criteria:

- 1 Pre-trained on large data**
The model is trained on a large dataset with the explicit goal of learning a rich latent representation of the data.
- 2 Transferable to downstream tasks**
The learned representation is reused (via fine-tuning) for different tasks — possibly the same task on different data, or entirely different tasks.
- 3 Pre-training improves performance**
Using the pretrained representation gives better results than training the downstream task from scratch (especially with limited labeled data).



The Pre-train → Fine-tune Paradigm

Transfer learning is not new — but the scale of foundation models is.



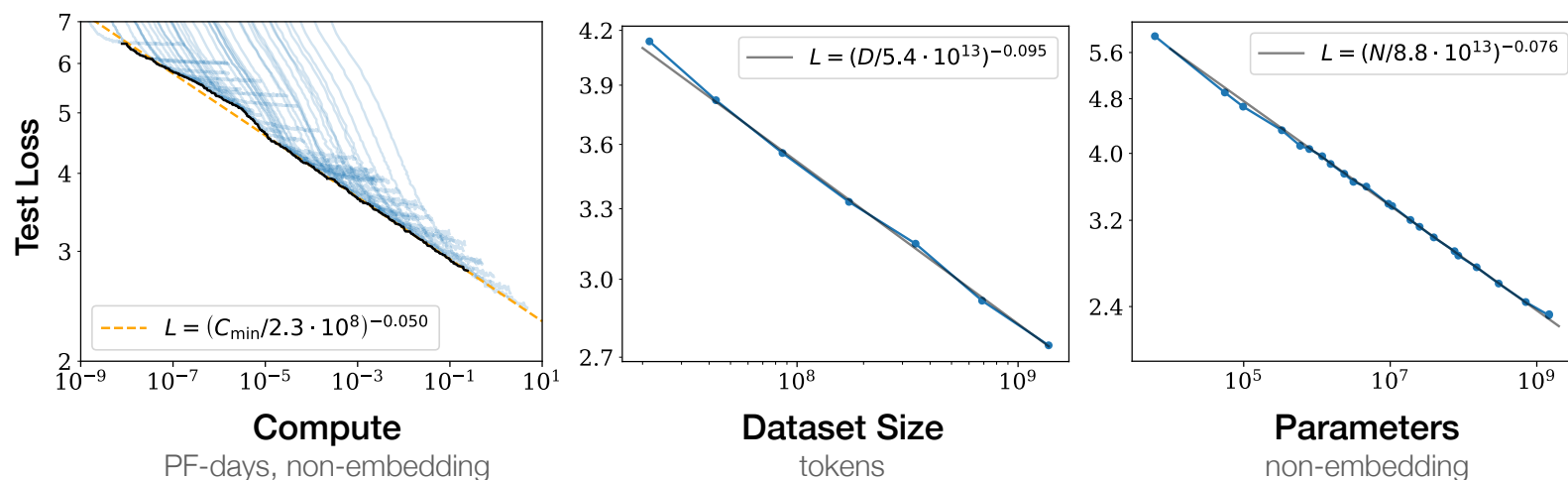
The backbone learns general features of the data.
Only the small task-specific model head needs labeled data and retraining.

During pre-training the model learns some features of the data that are useful for downstream tasks
The model starts with a learned understanding of the data, unlike one that starts with no prior knowledge.

Scale is the Key Ingredient

Foundation models work because of scaling data, parameters, compute.

Scaling laws (Kaplan et al. 2020):



Loss decreases predictably with compute, data size, and number of parameters.

Performance depends strongly on scale, weakly on model shape (depth vs. width)

Empirical power law performance : e.g. $L(C) \propto C^{-\alpha}$ where C = compute, $\alpha \approx 0.05$

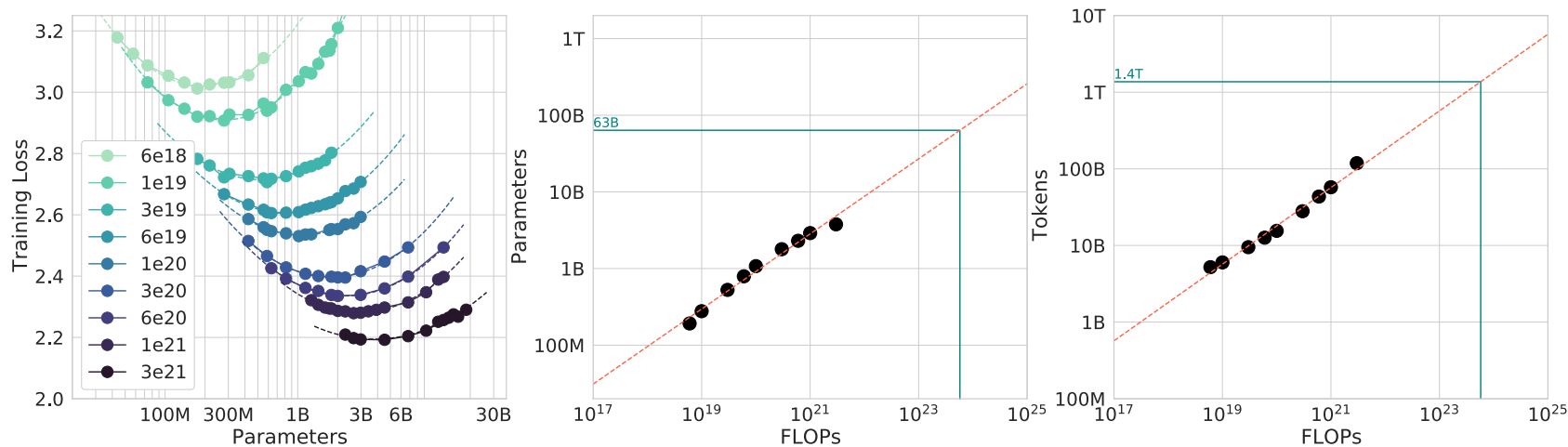
Given a fixed FLOPs budget, how should one trade-off model size and the number of training data?

Scaling in LLMs

Plot loss for fixed computational FLOPs

Find optimal parameters and data size given fixed computation FLOPs

Hoffman et al., 2020 [arXiv: 2203.15556](https://arxiv.org/abs/2203.15556)



Estimate loss scaling as $L(N, D) = L_{\infty} + \frac{A}{N^{\alpha}} + \frac{B}{D^{\beta}}$

FLOPS (N, D) $\sim 6 ND$

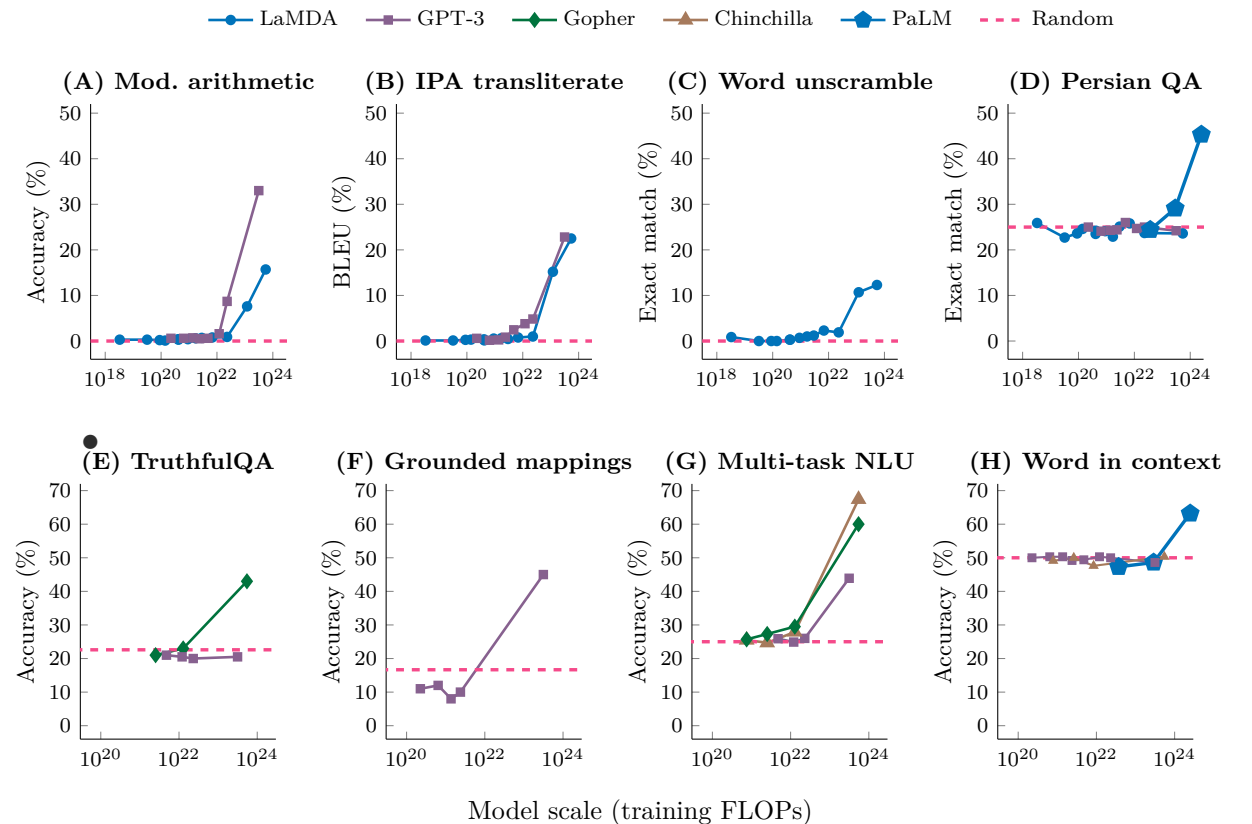
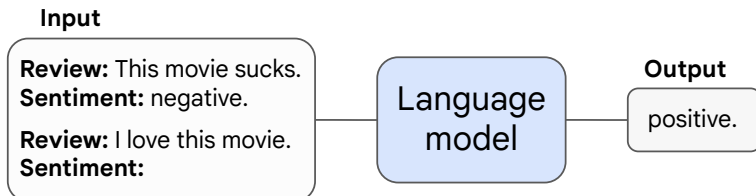
Question: is scaling happening also for HEP models ?

Emergent Abilities

Emergent abilities (Wei et al. 2022, [arXiv:2206.0768](https://arxiv.org/abs/2206.0768)):

With increasing scale, models acquire capabilities they were never explicitly trained for:

- Translation, coding, arithmetic
 - Examples:
 - 3 digit addition and subtraction and 2 digits multiplication
 - decoding the International Phonetic Alphabet (IPA)
 - unscrambling world letters
- Few-shot and zero-shot learning: perform tasks from just a few examples
- These abilities seem to emerge as phase transitions, not gradual improvements



Open question: can HEP foundation models also develop emergent capabilities at sufficient scale?

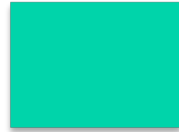
Scale is the key ingredient

BERT (2018)



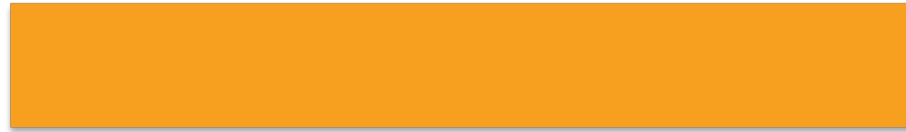
340M params

GPT-2 (2019)



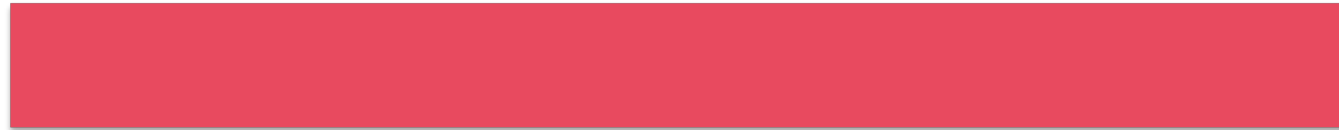
2B params

GPT-3 (2020)



175B params

PaLM (2022)



540B params

Example: GPT-3 (Brown et al. 2020)

Parameters: 175 billion

Training data: 300 billion tokens

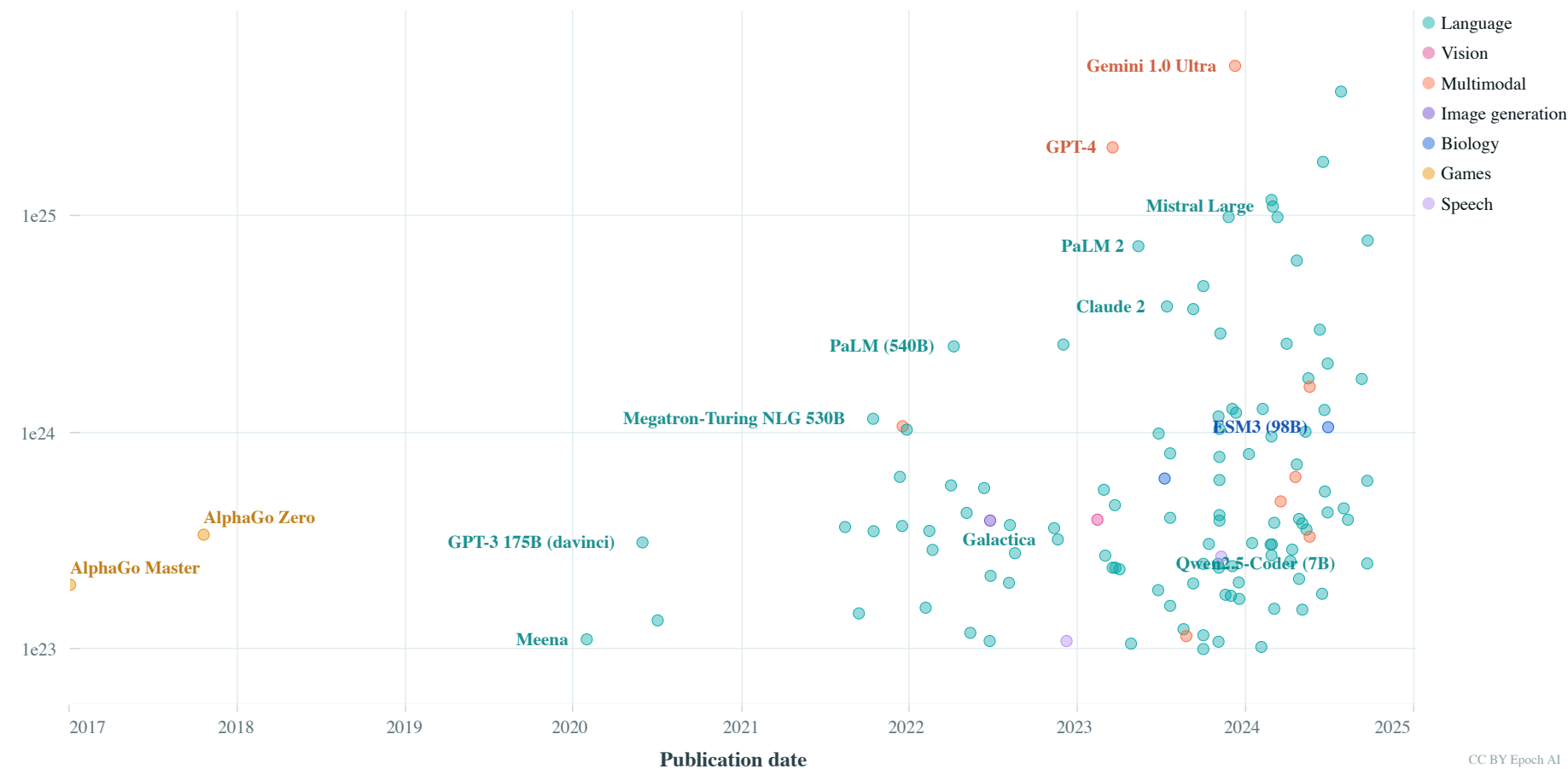
Compute: $\sim 10^{23}$ FLOPs (thousands of GPUs weeks)

Some Large Foundation Models

Large-Scale AI Models



Training compute (FLOP)



Transformers and Attention

The architectural backbone of most foundation models

The Attention Mechanism

Vaswani et al. (2017) — "Attention is All You Need" [arXiv: 1706.03762]

- Need (from NLP): an operation that lets each element in a sequence look at every other element and decide what information is relevant — in a data-dependent way.
- Combine information at locations far apart (e.g words at different positions)
 - Limitation of architectures like Recurrent Neural Networks (RNN) and Graph Neural Networks (GNN)
- Each token looks (attends) at every other token and depending on the data (with learnable parameters) the model decides what matters

Attention Layers are the key building blocks of the Transformer architecture

Attention Layer Operations

Vaswani et al. (2017) — "Attention is All You Need" [arXiv: 1706.03762]

Given an input sequence X of n tokens, each of dimension d :

Queries: $Q = X \cdot W_Q \quad (n \times d_k)$

Keys: $K = X \cdot W_K \quad (n \times d_k)$

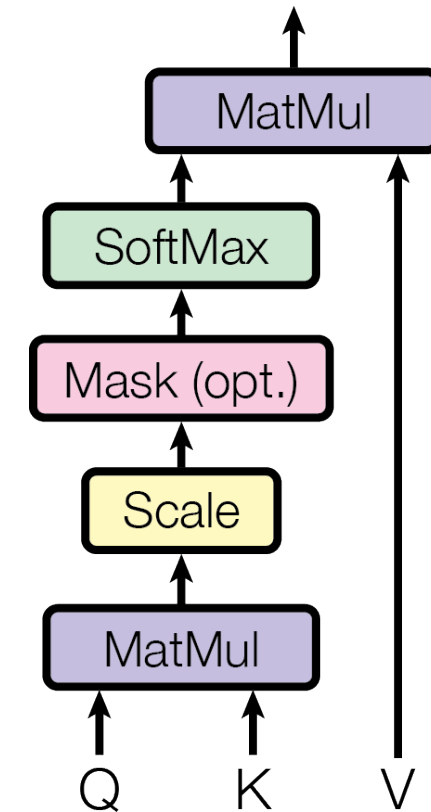
Values: $V = X \cdot W_V \quad (n \times d_v)$

$$Y = \text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad \text{matrix } (n \times d_v)$$

- **self attention:** same input sequence for query, keys and values
- **cross attention:** one sequence for queries and one for keys and values

they are still 3 distinct learnable weight matrices W_Q , W_K , W_V

Scaled Dot-Product Attention

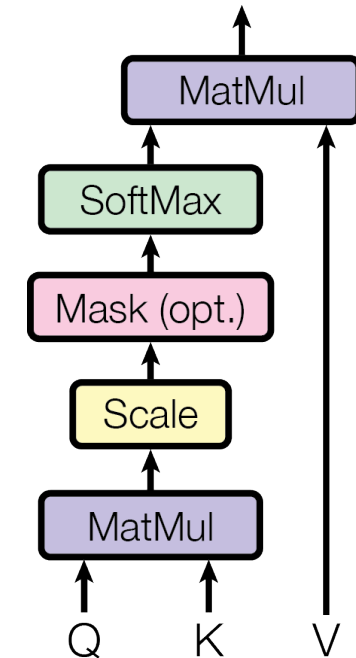


Attention: How does it work?

Intuition:

- Q: "what am I looking for?"
- K: "what do I contain?"
- V: "the actual information"
- $Q \cdot K^T$ measures compatibility between tokens
- softmax normalises scores (QK^T values) to attention weights (sum to 1)
- If a query Q_n matches with the key K_m the result after the softmax will be closer to 1 otherwise close to 0
- scaling $1/\sqrt{d_k}$ prevents gradients from vanishing for large d_k
- The results $Y_q = \sum_k A_{q,k} V_k$ are the weighted (by the attention scores) average values: a summary of the relevant information

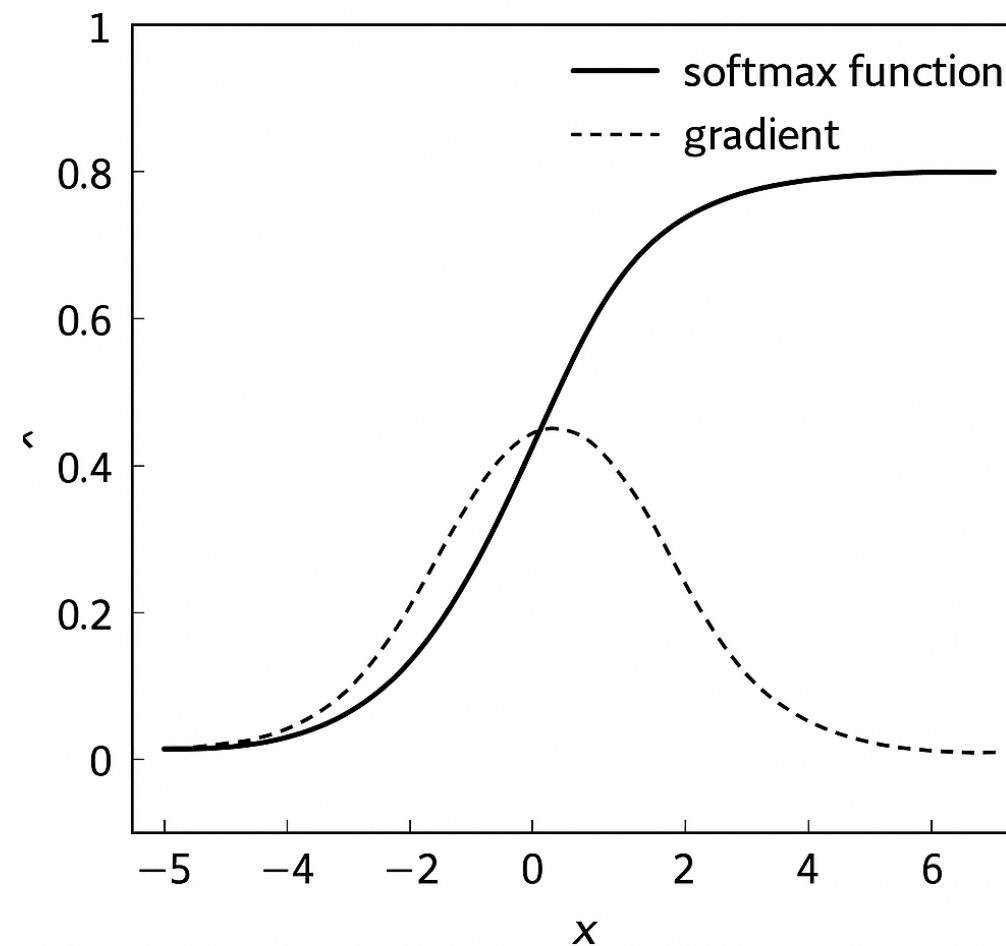
Scaled Dot-Product Attention



$$Y = \text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Softmax Gradient

- $Q \cdot K^T$ dot product is of length d_k
 - ➔ its standard deviation grows as $\propto \sqrt{d_k}$ pushing the softmax in a saturated regime preventing learning
- Solution: scaling the attention scores by $1/\sqrt{d_k}$
 - ➔ prevents gradients from vanishing for large d_k



Mask in the Attention Operator

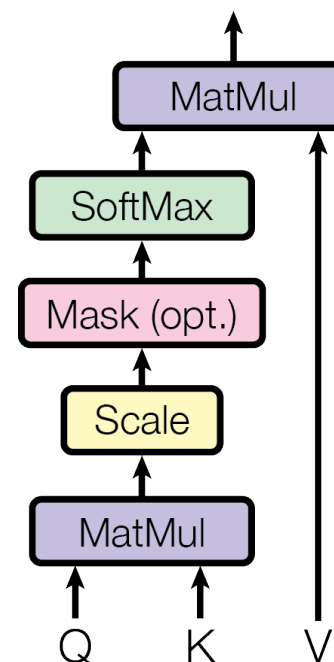
In some case it is needed to make the Attention operation casual

Don't want that a token has a dependence from a token in the future, but only from a token in the past

This is achieved by using a mask before the softmax operation setting the $Q \cdot K^T$ values to zero when $k > q$

➡ output will not depend on keys and values with indices greater than the current q

Scaled Dot-Product Attention



$$Y = \text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

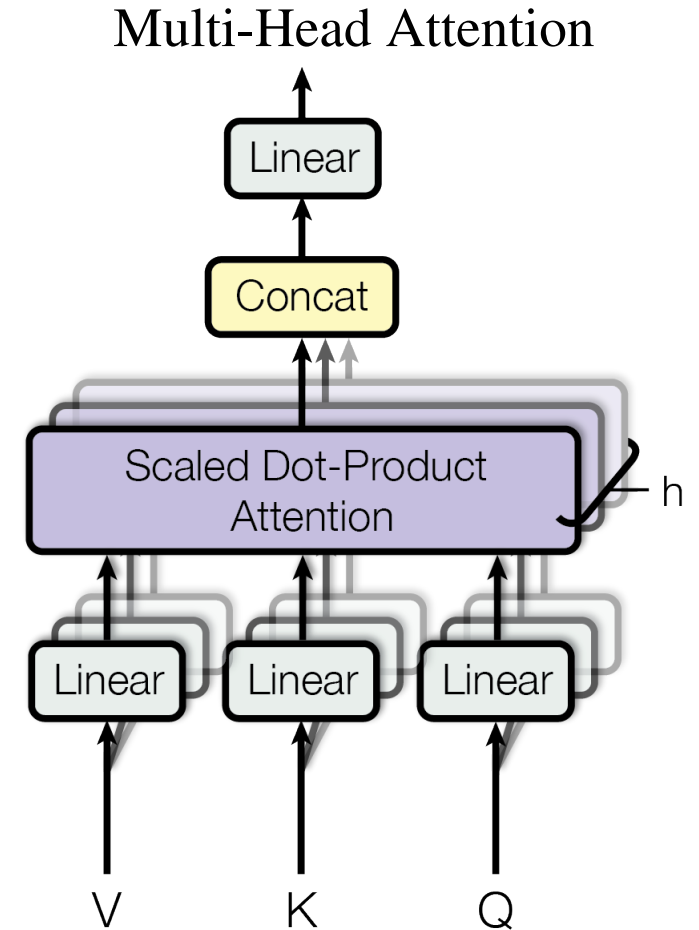
Multi-Head Attention

- The same Attention operations are repeated for h number of heads
 - using weight matrices of size $h \times d \times d_k$
- The result is a sequence of matrices $Y_1, \dots, Y_h$
- Afterwards the result is concatenated and a linear operation is applied using a weight learnable matrix

$$Y = (Y_1, \dots, Y_h)W^0$$

All Attention operations are simple matrix multiplications which can be computed in parallel and efficiently using GPU

Each head can focus on different aspects of the data (e.g., different particle correlations, different distance scales).
This gives the model diverse "views".



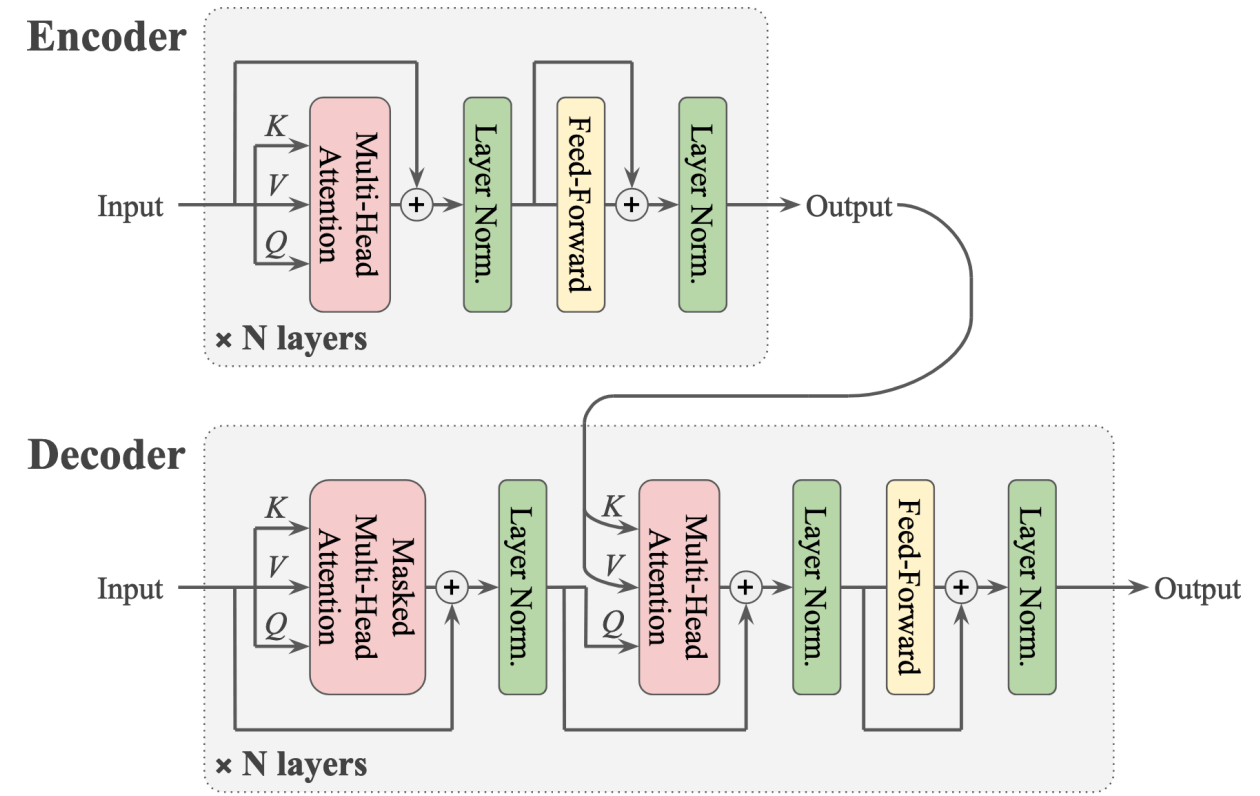
The Transformer Block

The Transformer Block:

- 1 Input tokens → Multi-Head Attention
- 2 Add residual connection + Layer Normalization
- 3 Feed-Forward Network (two linear layers + activation)
- 4 Add residual connection + Layer Normalization
- 5 Output and repeat for N layers

Original model designed for language translation tasks composed by

- **Encoder**
 - N layers of self attention transformer blocks
- **Decoder**
 - A first casual first layer (masked attention)
 - Second layer uses output of decoder as input for K, and V (performing cross-attention)



(from Vaswani et al. (2017), "Attention is All You Need" [[arXiv: 1706.03762](https://arxiv.org/abs/1706.03762)])

Computational Cost of the Attention

Computational costs (FLOPs) for a single head attention:

Stage	Key operation (with dimensions)	FLOPs per head	Why N^2 ?
Score matrix	$Q \in R^{N \times d_k}, K \in R^{N \times d_k} : S = QK^T \in R^{N \times N}$	$N^2 d_k$	Each of the N queries dots with all N keys
Soft-max	Row-wise softmax: $A = \text{softmax}(S)$ on $S \in R^{N \times N}$	$\approx 2N^2$	Exponentiate and normalise all N^2 scores
Weighted sum	$V \in R^{N \times d_v}, A \in R^{N \times N} : Z = AV \in R^{N \times d_v}$	$N^2 d_v$	Each of the N outputs mixes N value vectors

$$\text{FLOPs} = h(N^2 d_k + 2N^2 + N^2 d_v) \approx \mathcal{O}(hN^2(d_k + d_v))$$

Attention operator scales quadratic with the sequence length

Important Distinctions

Not every transformer is a foundation model:

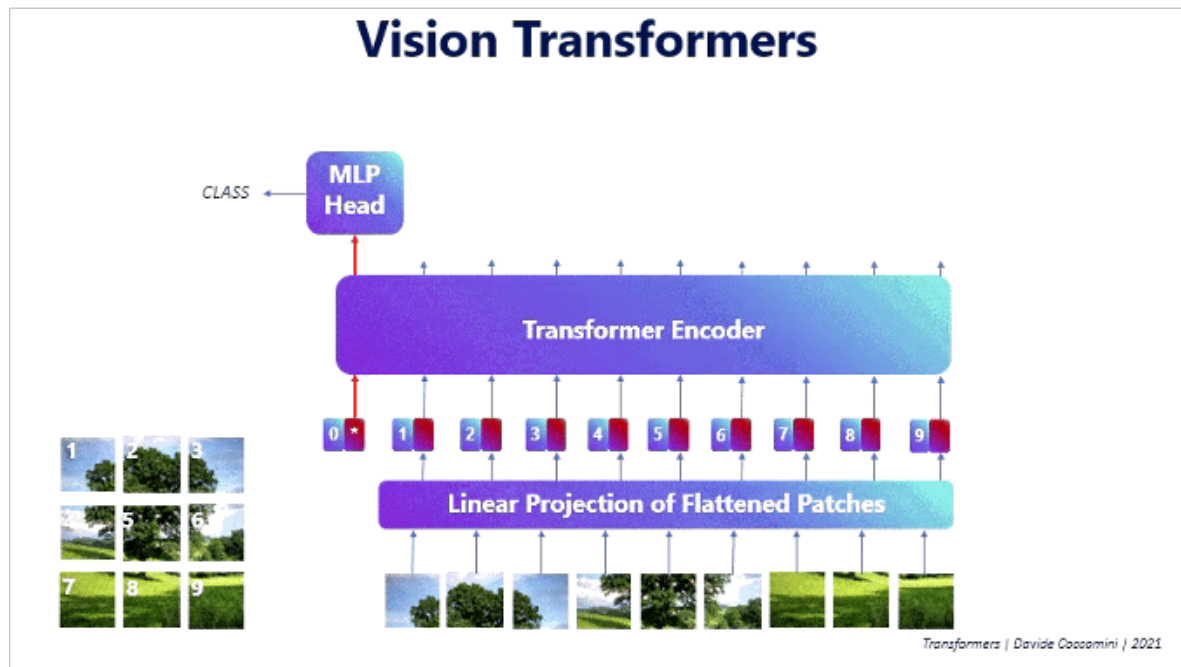
A transformer is an architecture. A foundation model is a training paradigm. You can build a foundation model with a different architecture, and you can use transformers without any pre-training/fine-tuning.

Not every foundation model is an LLM:

LLMs (GPT, BERT, etc.) are the most famous foundation models, but the concept applies to any data type: images (CLIP, DALL-E), audio (Whisper), proteins (AlphaFold), and particle physics.

Vision Transformer (ViT)

Applying the Transformer architecture directly to image patches



- ① Split into patches (e.g. 16×16 px each)
- ② Flatten + linear project → patch tokens
- ③ Tokens through L Transformer layers → MLP → class probs

Model can be finetuned for different tasks: image classification, object detection, image segmentation

ViT replaces convolutions entirely, with enough data, pure attention over patches learns richer representations than CNN feature hierarchies, and scales more predictably with compute.

1 Patch Embedding

- Each P×P patch is flattened to a vector, then linearly projected.
- A learnable 1D positional embedding is added to each token so the model knows spatial layout.

2 Encoder Layer (repeated L times)

- Multi-head Self-Attention (MSA) over all tokens (every patch attends to every other)
- MLP block with GELU activation wrapped with LayerNorm and residual connections.

3 Scale & Pretraining

- ViT underperforms CNNs on small datasets, it lacks the inductive biases (locality, translation equivariance that CNNs have).
- With large-scale pretraining (e.g. JFT-300M, ImageNet-21k) ViT matches or beats CNNs.

Three approaches to Foundation Models in HEP:

1. Use existing LLMs for HEP tasks

ChATLAS (AI assistant for ATLAS), ACCGPT (CERN knowledge Retrieval ChatBOT)

2. Adapt LLMs to understand physics/math

Symbolic integration as translation [Lample & Charton 2019], continuous numbers tokenisation [xVal, Golkar 2024]

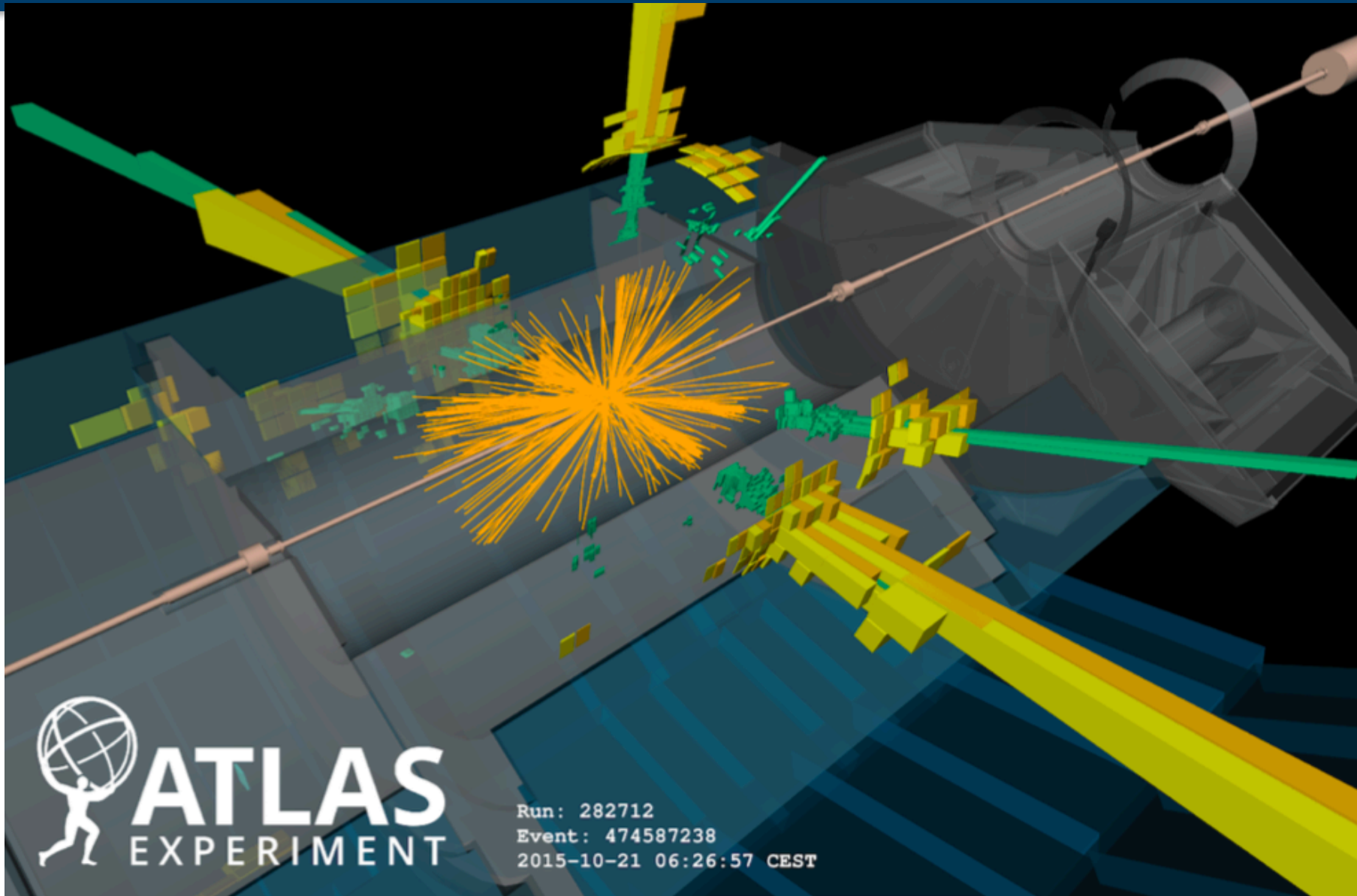
3. Build foundation models from scratch for HEP data

Train on HEP data: jets, events, calorimeter showers. Will now focus here

Foundation Models for HEP

From text to Particle data (e.g. Jets)

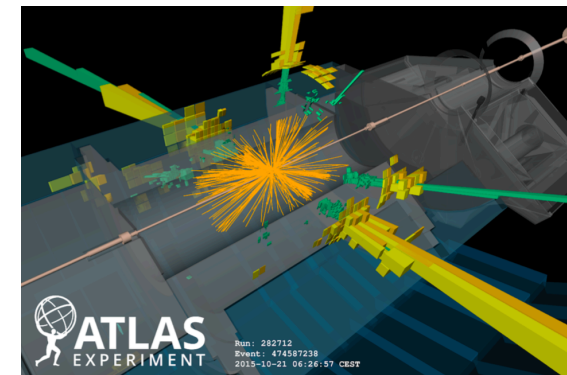
High Energy Physics Data: Particle Jets produced from pp collisions



Particle Jets

What is a Jet?

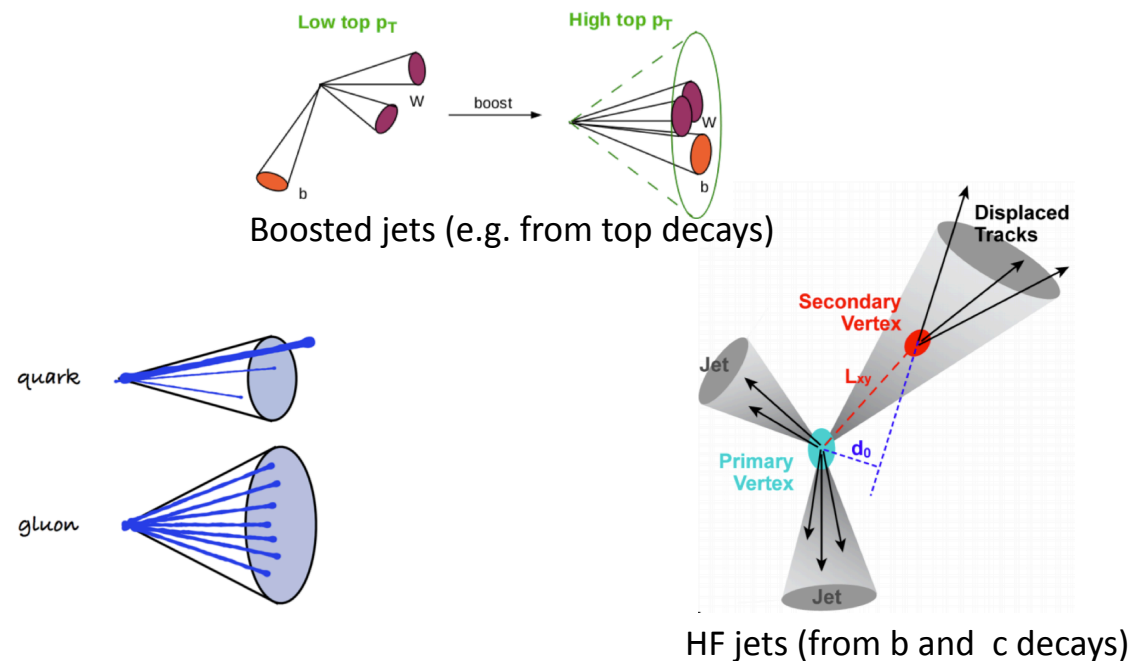
- A collimated shower of particles in the detector
- Particles are reconstructed as tracks and calorimeter towers



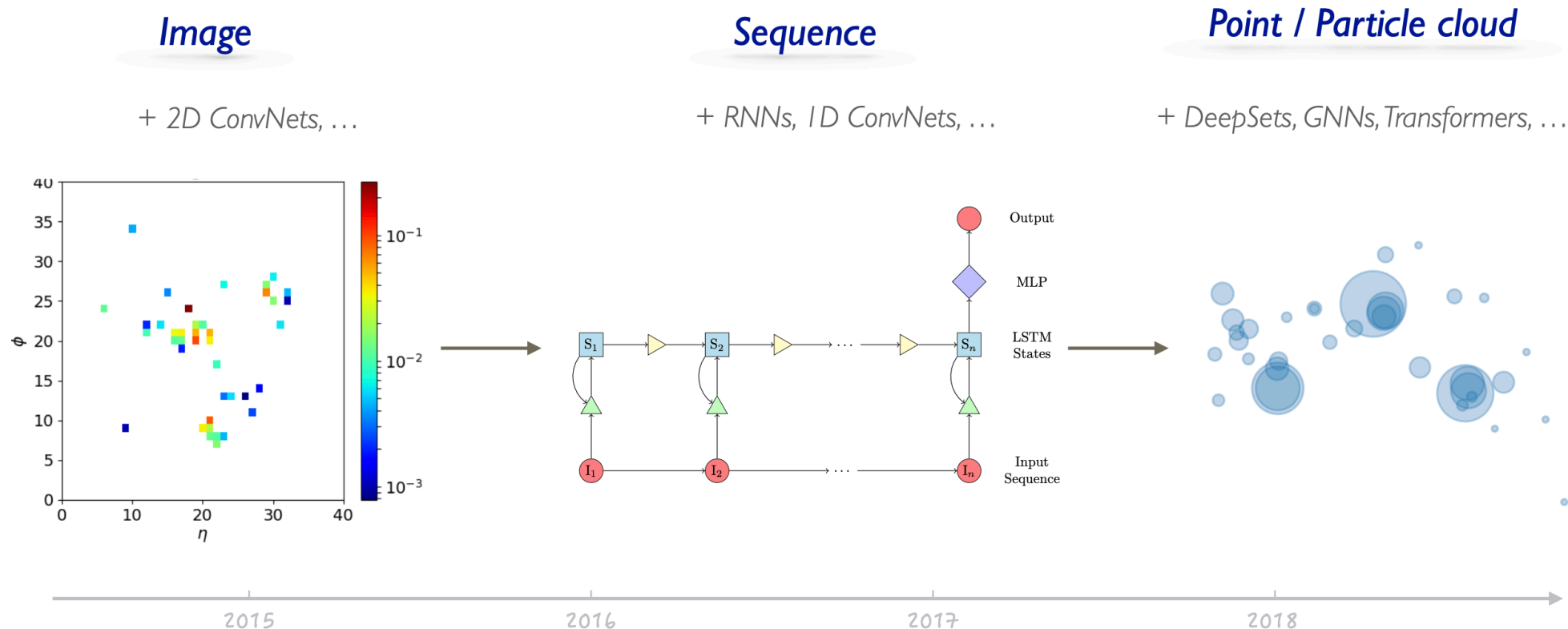
Jet Tagging

- Distinguish different types of jets (classification task)
- quark vs gluon jets
- boosted jets produced by heavy resonances (W, Z, Higgs, top) for new physics searches
- heavy flavour jets (secondary vertices with displaced tracks)

Tagging is a difficult task: natural to use ML algorithms



History of Jet Tagging in HEP

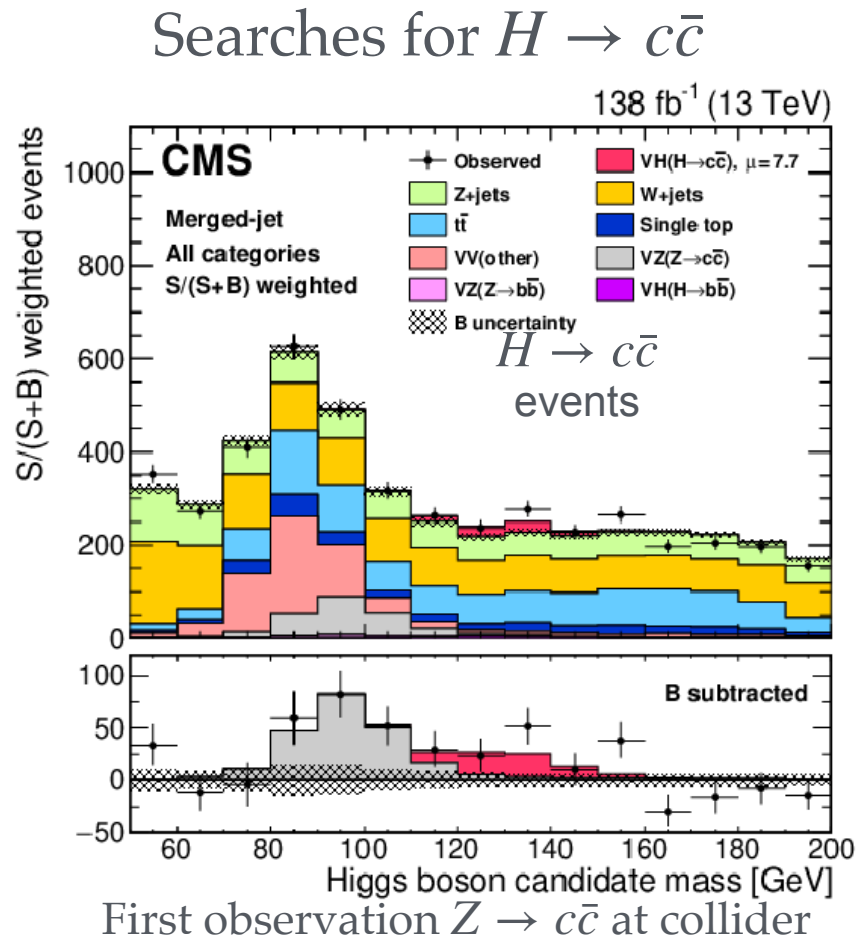
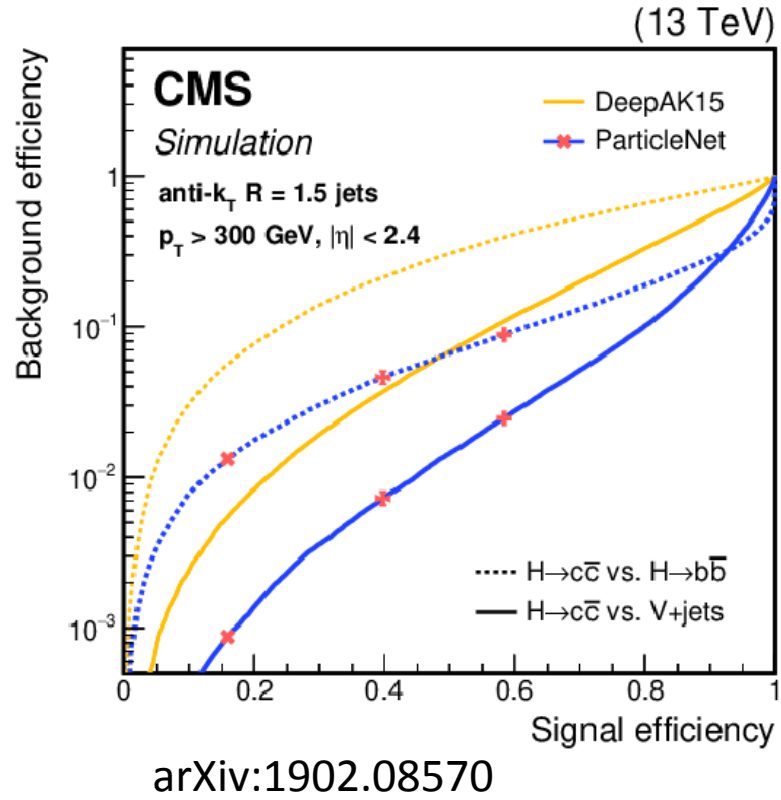
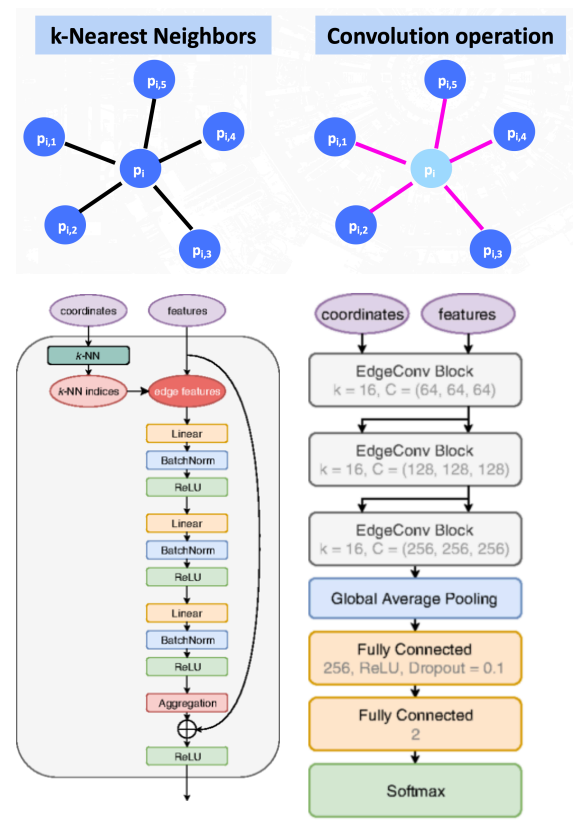


Evolution of Jet tagging

1. consider jets as 2D images and use **Convolutional Neural Networks (CNN)**
2. jets are like a sequence of particles, analogy with NLP: use **Recurrent NN** and 1D CNN
 - ➡ physics (QCD) is the grammar that is going to be learnt
3. **Graph Neural Network (GNN)** to deal with variable length data
4. **Transformers** to exploit scaling and achieve optimal performances

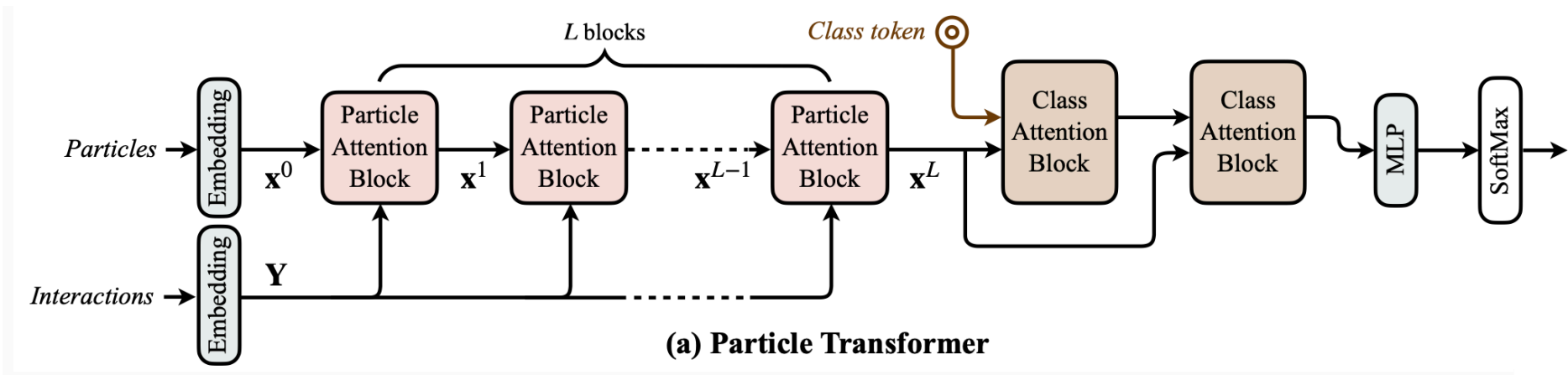
CMS Jet Tagger: Particle Net

- GNN for tagging heavy flavour jets (b/c jets) DEVELOPED IN CMS
- based on Dynamic Edge Convolution (DGCNN)
 - large improvement in performance compared to previous taggers (based on 1D-Convolution + recurrent architecture)



ParT : jet tagging using transformers

[arXiv:2202.03772](https://arxiv.org/abs/2202.03772)



	All classes		$H \rightarrow b\bar{b}$	$H \rightarrow c\bar{c}$	$H \rightarrow gg$	$H \rightarrow 4q$	$H \rightarrow \ell\nu qq'$	$t \rightarrow bq q'$	$t \rightarrow b\ell\nu$	$W \rightarrow qq'$	$Z \rightarrow q\bar{q}$
	Accuracy	AUC	Rej _{50%}	Rej _{50%}	Rej _{50%}	Rej _{50%}	Rej _{99%}	Rej _{50%}	Rej _{99.5%}	Rej _{50%}	Rej _{50%}
PFN	0.772	0.9714	2924	841	75	198	265	797	721	189	159
P-CNN	0.809	0.9789	4890	1276	88	474	947	2907	2304	241	204
ParticleNet	0.844	0.9849	7634	2475	104	954	3339	10526	11173	347	283
ParT	0.861	0.9877	10638	4149	123	1864	5479	32787	15873	543	402

Table 2. Number of trainable parameters and FLOPs.

	Accuracy	# params	FLOPs
PFN	0.772	86.1 k	4.62 M
P-CNN	0.809	354 k	15.5 M
ParticleNet	0.844	370 k	540 M
ParT	0.861	2.14 M	340 M
ParT (plain)	0.849	2.13 M	260 M

Example: performance in $H \rightarrow c\bar{c}$

- doubled background rejection: $\rightarrow \sim 1.4x$ significance!
- Same significance reach with half of the data!

GNN vs Transformer

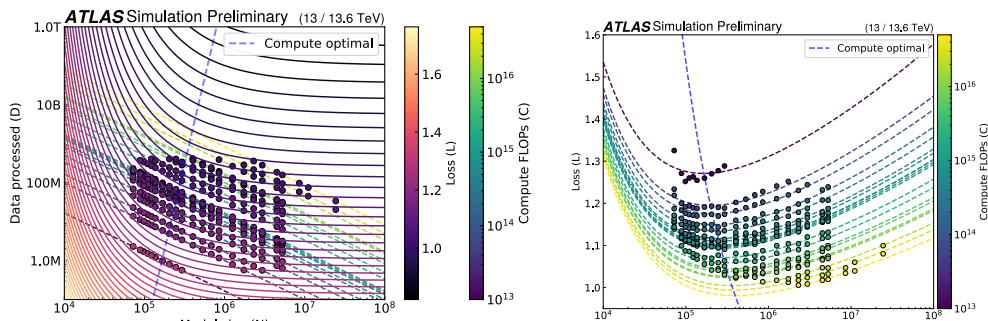
Architectural comparison

Graph Neural Networks (GNN)	Aspect	Transformers
Graph: nodes + edges	Input	Sequence of tokens
Exploits explicit topology	Structure	No prior structure assumed
Sparse, local message passing	Connectivity	Dense, all-pairs attention
Permutation equivariance	Inductive bias	Positional encodings
Scales to large sparse graphs	Scalability	$O(n^2)$ attention cost
Molecules, social & knowledge graphs	Main use	NLP, vision, time series

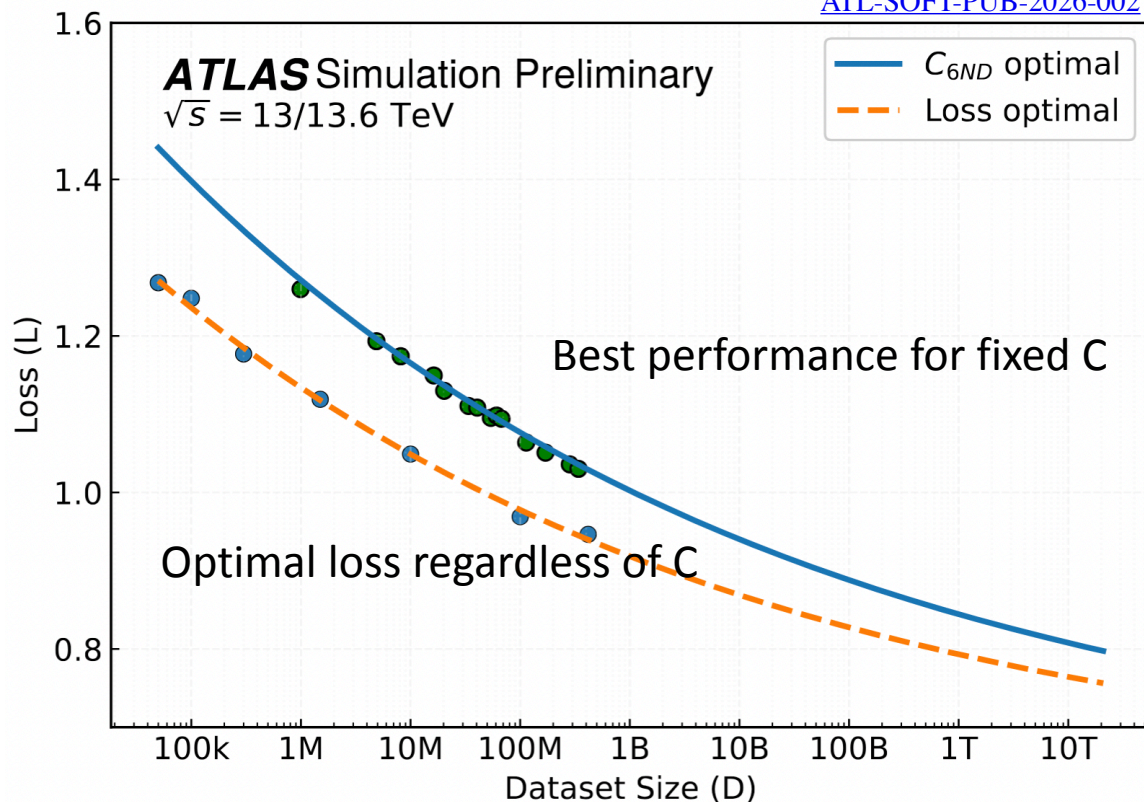
Key insight: Graph Transformers merge both — applying attention over graph-structured inputs, gaining long-range context while preserving relational inductive bias.

- GNN: strong inductive bias encoded in graph structure, but problem with over-squashing and over-smoothing: don't work for deep structures (several layers)

Scaling for HEP Models: Study on Jet Tagger

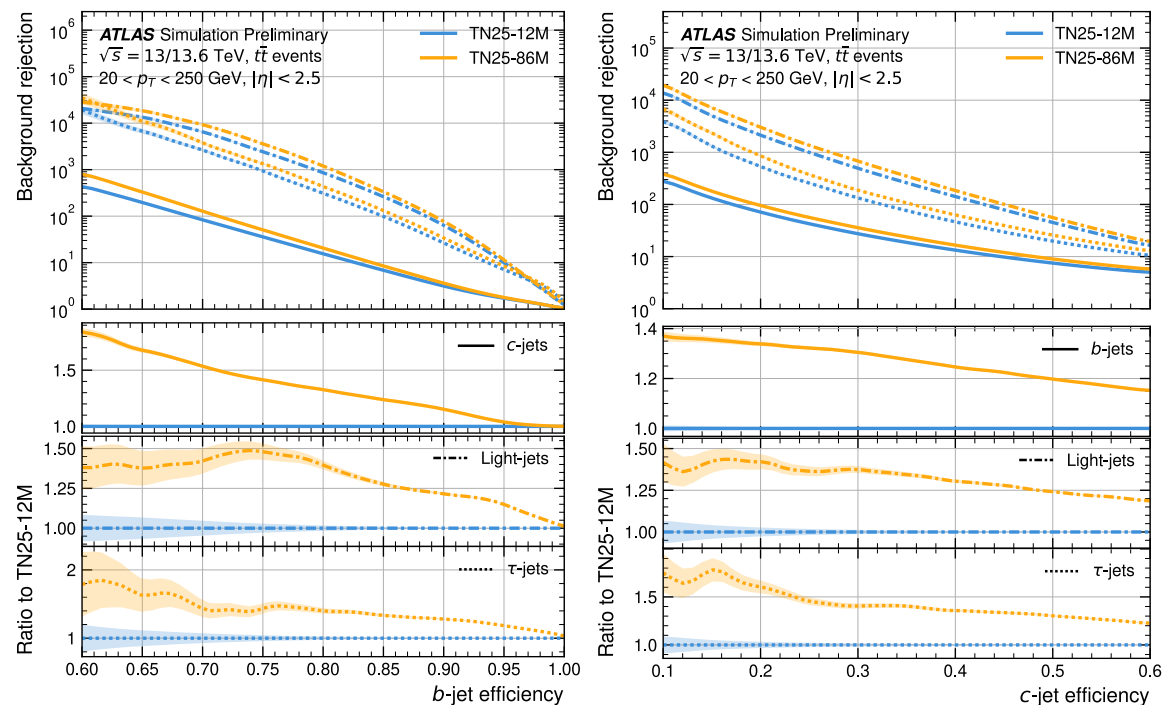


ATL-SOFT-PUB-2026-002



$$L(N, D) = L_{\infty} + \frac{A}{N^{\alpha}} + \frac{B}{D^{\beta}}$$

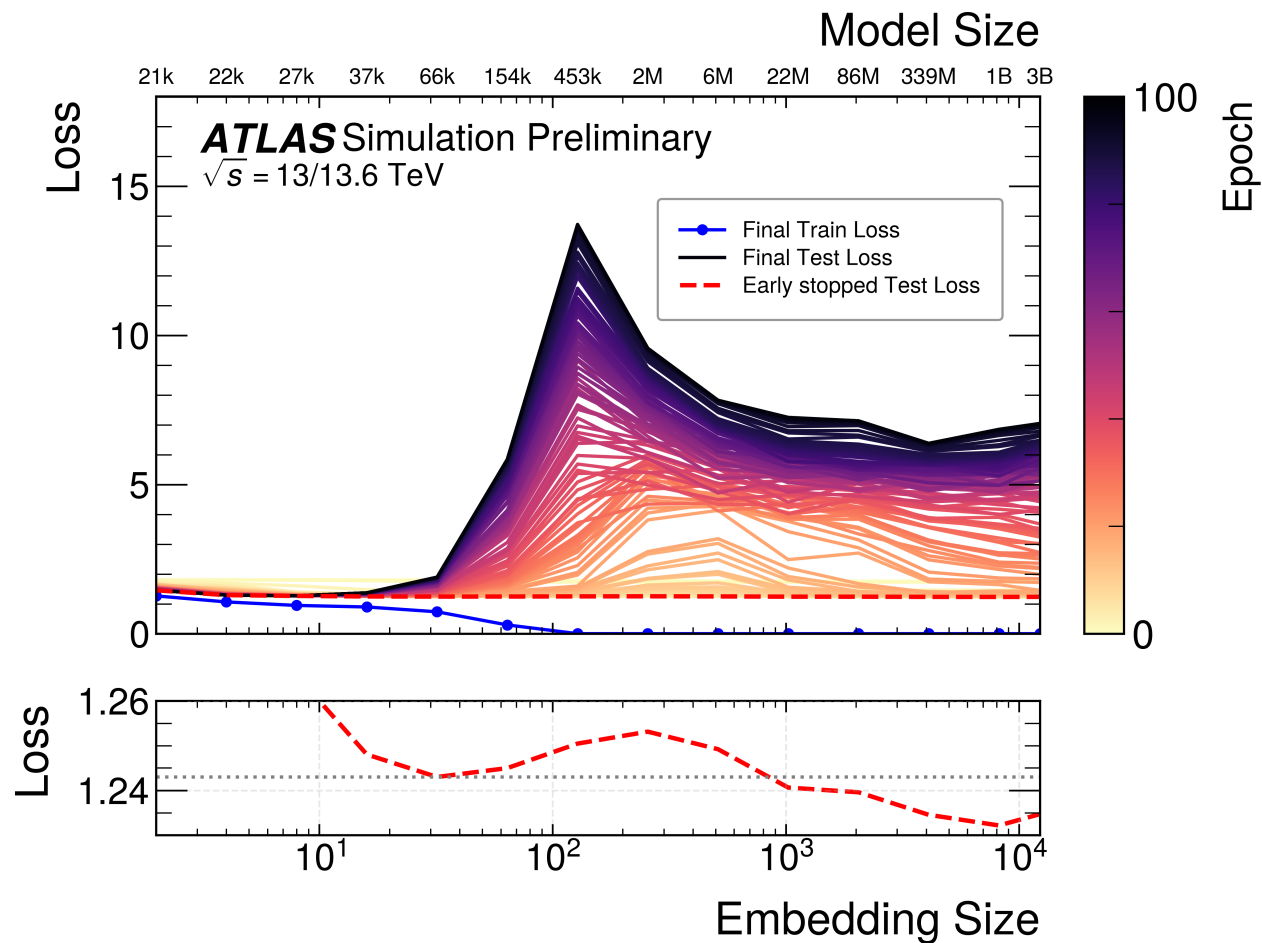
$$C(N, D) \sim 6ND$$



ATLAS-SOFT-PUB-2026-002

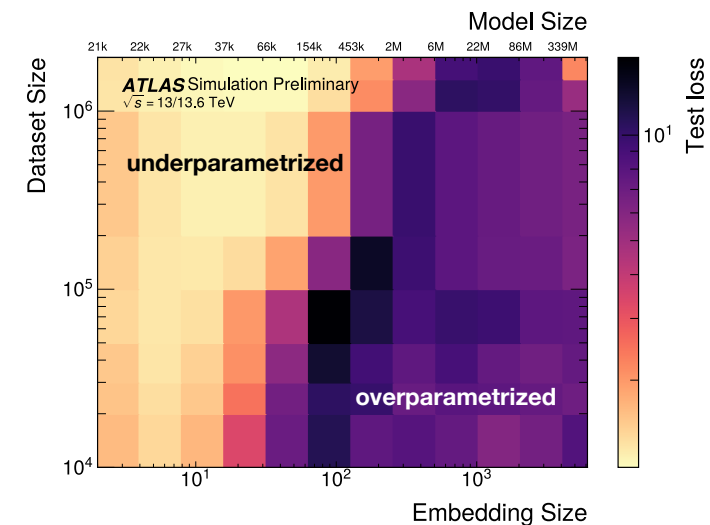
Double Descent

Study test loss after 100 epochs and using early stopping as function of model size (and data set size)



- Two distinct regions:
- underparametrized
 - overparametrized

Phenomena already observed in large ML models (see [Belkin et al. arXiv:1812.11118](#))



Natural Language vs. High Energy Physics Data

Large Foundation Models originally designed for NLP.

Natural Language	Particle Jets (HEP)
Discrete tokens (words, subwords)	Continuous numbers (p_T , η , ϕ , ...)
Ordered sequence — syntax matters	Unordered set — permutation invariant
Vocabulary $\approx$ 50 000 tokens	No natural vocabulary $\rightarrow$ need tokenization
Meaning builds across many sentences	Physics encoded in correlations between particles
1 embedding per token	Multiple features per particle (4-vector, PID, ...)

How do we turn continuous, permutation-invariant physics data into something a transformer can process?

Particle Jets: the Main Building Block

A jet is a collimated spray of particles produced by the fragmentation of a quark or gluon.

Jet = $\{ p_1, p_2, \dots, p_N \}$ (unordered set, N varies from jet to jet)

Each particle: $p_i = \{ p_T, \eta_{\text{rel}}, \phi_{\text{rel}}, \text{particle ID, displacement, ...} \}$

Important properties:

- Variable multiplicity: $N \approx 5 - 150$ particles per jet
- The ordering of particles is physically meaningless (permutation invariant)
- Different jet origins (quark, gluon, top, W, H) leave different signatures
- Feature correlations encode the physics (e.g., n -subjettiness τ_{32} for top quarks)

Downstream tasks on jets:

Tagging: classify jet origin (quark/gluon/top/W/H/...)

Generation: learn $p(\text{jet})$ and sample new realistic jets

Anomaly detection: identify jets that don't match known physics

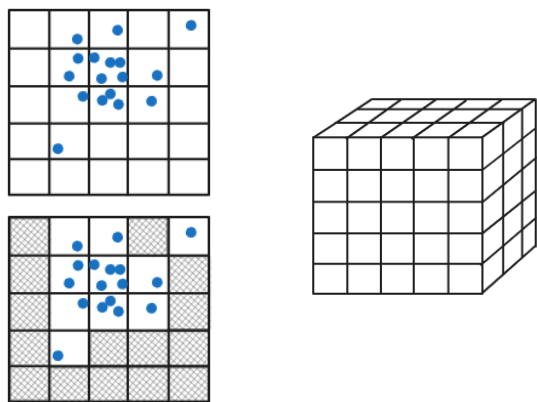
The Tokenization Of Physics Data

Transformers were designed for discrete tokens. Our physics data are continuous.
Some approaches exist to bridge this gap:

Approach 1: Binning

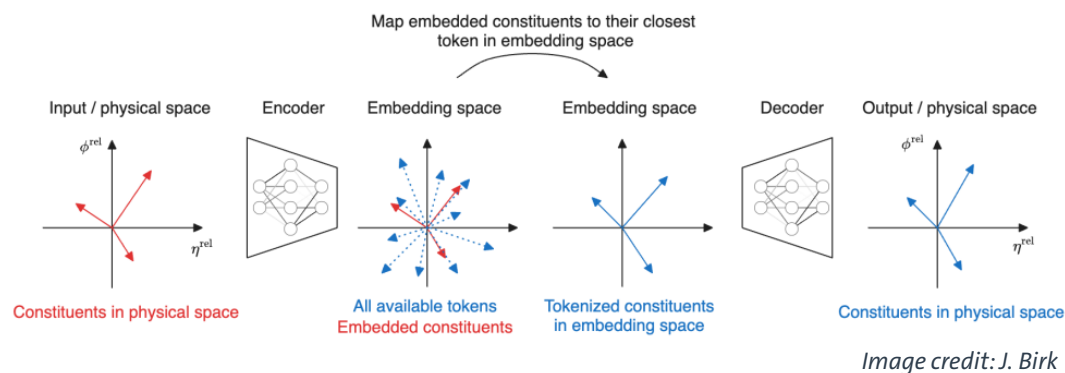
Divide each feature dimension into N bins.
Map each particle $\rightarrow$ bin index $\rightarrow$ integer token.

Simple but: vocab size = $\prod_i N_{\text{bins},i}$
Sub-optimal coverage of feature space in case of multi-dim data



Approach 2: VQ-VAE (used by OmniJet- α , MPM)

Vector Quantized Variational Autoencoder.
Learn an embedding space of K discrete codes.
Encoder maps particle $\rightarrow$ nearest code.
Adapts to the shape of the data distribution!



Note: Discretisation is needed if model is trained to predict next tokens, not if supervised trained (e.g for classification: ParT)

Foundation Models in HEP : Different Pre-Training Strategies

Different approaches, one goal: reusable representations

Overview of Pre-training Strategies

Different groups have explored different pre-training objectives for building foundation models for HEP
Here is a list not exclusive of some of the HEP Foundation Models.

A

Supervised Classification

[ParT \(2022\)](#)

Pre-train on labeled jet classification.
Directly optimizes for the downstream task.

B

Masked Particle Prediction

[MPM \(2024\)](#)

Mask particles, predict their properties.
Inspired by BERT.

C

Next-Token Prediction

[OmniJet- \$\alpha\$ \(2024\)](#)

Generate jets token by token.
Inspired by GPT.

D

Contrastive Learning

[RS3L \(2024\)](#)

Same event, different simulations $\rightarrow$ close.
Different events $\rightarrow$ far apart.

E

Hybrid (Generation + Classification)

[OmniLearn \(2024\)](#)

Combine generation and classification in a single pretraining.

Strategy A: Supervised Pre-training — Particle Transformer

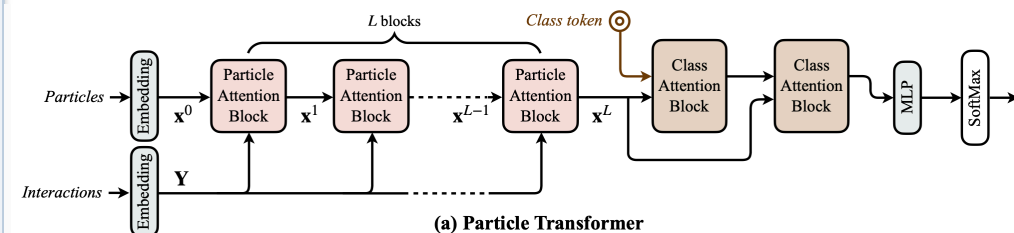
Qu, Li, Qian — PMLR 2022 [arXiv: 2202.03772]

Approach: pre-train on supervised multi-class jet classification on a large dataset, then fine-tune on a different classification task / dataset.

Training loss: cross-entropy Loss over 10 jet classes

Dataset used for training : JetClass — 100M jets, 10 classes (q/g, top, W, Z, H, ...)

Features: p_T , η , ϕ , particle ID, trajectory displacement, interaction features

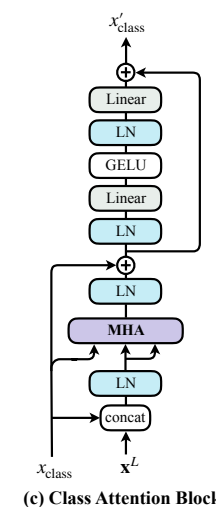
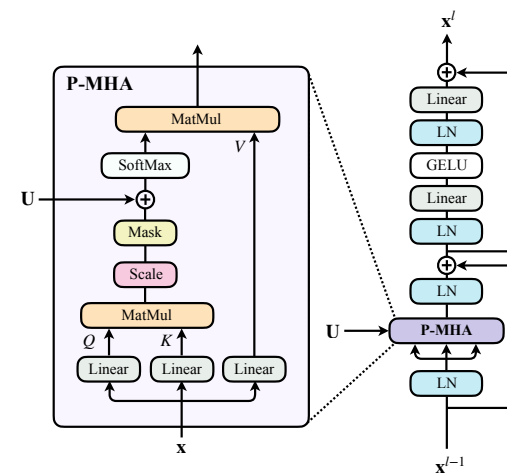


Key architectural idea — interaction features:

ParT computes pairwise features between particles (inspired by jet clustering history). These are used as attention biases in the transformer, informing the model about the angular and momentum structure of the jet.

Results:

- State-of-the-art jet tagging at time of publication
- Fine-tuning to different datasets shows transfer learning improvement
- Adding more physics features (particle ID, displacement) consistently helps
- **Limitation: requires labels for pretraining — cannot pretrain on real data**



Strategy B: Masked Particle Modeling (MPM)

Golling, Heinrich, Kagan, Klein, Leigh, Osadchy, Raine — 2024 [arXiv: 2401.13537]

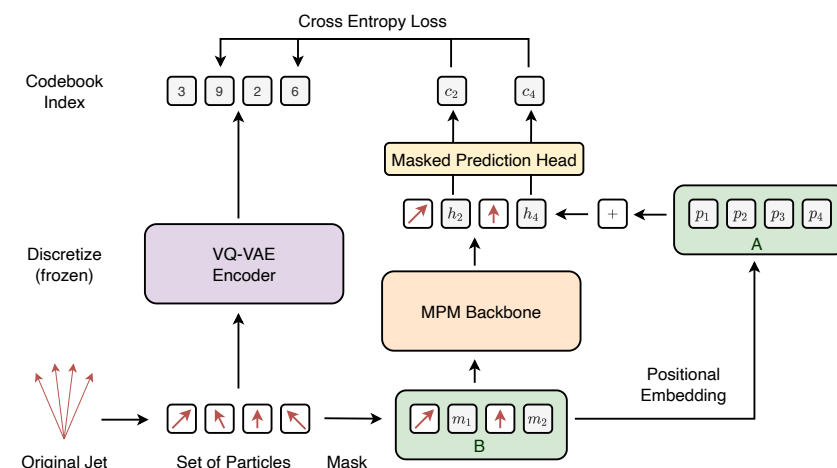
Inspired by BERT: mask out parts of the input, train the model to predict them.



→ predict properties of masked particles

Challenges for MPM model:

1. Particles have continuous features (p_T , η , ϕ), not discrete tokens
→ Solution: tokenize with VQ-VAE or K-Means clustering
→ Predict the discrete token index using cross-entropy loss
2. Particles are unordered (unlike words in a sentence)
→ Without ordering, all masked positions predict the same distribution
→ Solution: sort particles before prediction head (backbone stays permutation-invariant)
3. Multi-modal features: continuous vs. prediction approach
→ Alternatively: use normalizing flows for continuous density estimation



Model: 40M parameter transformer backbone, pretrained on 100M jets (JetClass)

MPM: Results and Domain Transfer

10-class jet classification with a linear probe on the frozen backbone:

Three evaluation modes:

1. Standard training: train classifier from scratch on task-specific data
2. Fixed backbone: freeze pretrained backbone, train only the classifier head
3. Fine-tuned backbone: load pretrained weights, fine-tune everything end-to-end

Fine-tuned backbone consistently outperforms training from scratch, especially with limited labeled data.

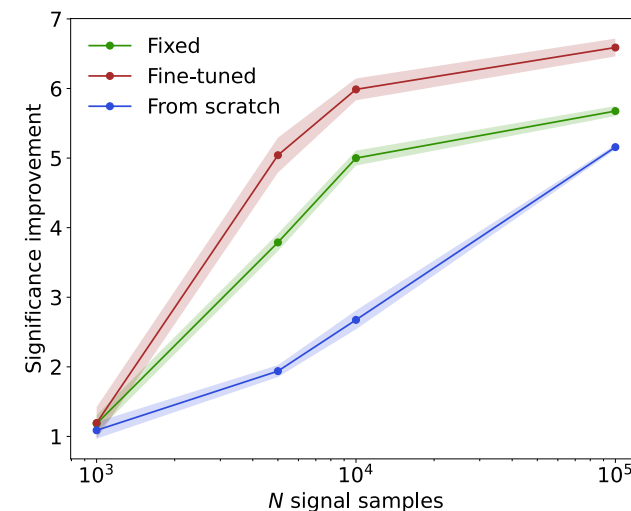
Domain transfer experiment:

1. Pretrain on "unlabeled" dataset A → treat like real data
2. Fine-tune on labeled dataset B → treat like simulation
3. Evaluate on dataset A → test on "real data"

Result: model performance is similar on both datasets.

This is a step towards using real data for pretraining and simulation for fine-tuning

This is the scenario we want for foundation models in experiments.



Strategy C: Next-Token Prediction — OmniJet- α

Birk, Hallin, Kasieczka — Mach. Learn.: Sci. Technol. 2024 [arXiv: 2403.05618]

Cross-task foundation model for particle physics:
simultaneously capable of unsupervised generation AND supervised classification

Architecture: GPT-1 style transformer (decoder-only, no positional encoding)

Pretraining: autoregressive next-token prediction (self-supervised, no labels needed):

Learn: $p(x_j \mid x_{j-1}, x_{j-2}, \dots, x_1)$

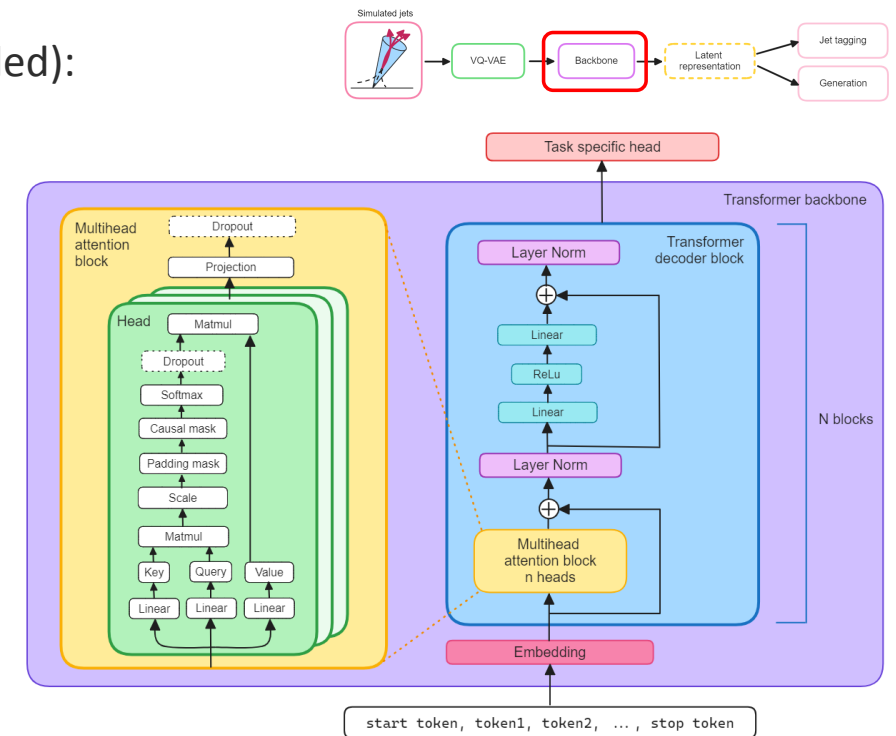
Loss: $L = - \sum_j \log p(x_j \mid x_{< j})$ (cross-entropy, next-token prediction)

Training input: $\langle \text{start} \rangle, \text{token}_1, \text{token}_2, \dots, \text{token}_N, \langle \text{stop} \rangle$
(tokenize using VQ-VAE, one token per particle constituent)

Idea: as the model learns to generate realistic jets token by token, it necessarily learns the correlations and structure of jet physics. This learned representation is then useful for classification.

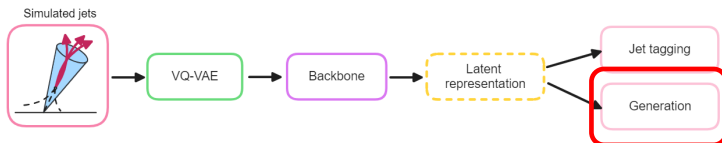
Dataset: JetClass, 12.5M quark/gluon + hadronic top jets. No class labels used in pretraining.

Can also be trained on real data



OmniJet- α : Generation and Transfer to Classification

Generation:



Once trained, the model generates jets autoregressively:

<start> $\rightarrow$ sample token₁
 $\rightarrow$ sample token₂ conditioned on token₁
 $\rightarrow$... $\rightarrow$ until <stop>

VQ-VAE decoder maps tokens
back to physics space (p_T , η , ϕ).

Good agreement with truth distributions

KEY RESULT:

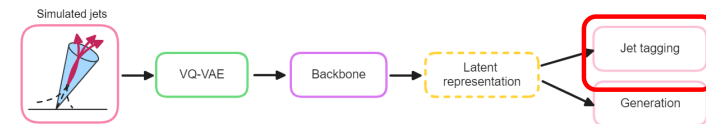
Fine-tuning significantly outperforms training from scratch.

The pre-trained model needs only $\sim 1\,000$ labeled events to reach the same accuracy that the "from scratch" model reaches with $\sim 1\,000\,000$ events.

$\rightarrow$ Factor $\sim 1000\times$ in data efficiency!

1 000 events vs. 1 000 000 events for the same accuracy

Transfer to classification:



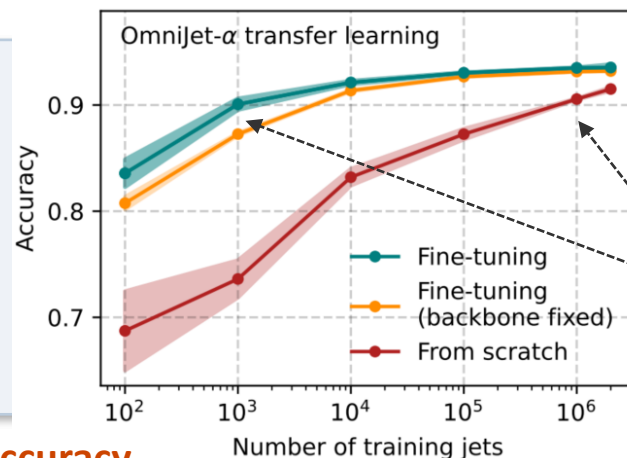
Swap the generation head for a classification head:

Linear $\rightarrow$ ReLU $\rightarrow$ Sum $\rightarrow$ Linear $\rightarrow$ Softmax

Test three modes:

1. From scratch (random init)
2. Fine-tune all weights
3. Backbone fixed (only head trains)

Task: classify quark/gluon vs. hadronic top jets



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

OmniJet- α on Real Data: Aspen Open Jets

Amram, Anzalone, Birk, Faroughy, Hallin, Kasieczka, et al. — 2024 [arXiv: 2412.10504]

Since OmniJet- α requires no labels for pretraining, it can train on real collider data.

The Aspen Open Jets dataset:

- From CMS Open Data — Run 2 (2016), 16.4 fb^{-1} 180 million real jets — publicly available
- Expected composition: mostly light-quark and gluon jets (no labels)

Results of Jet pretraining on real data (generation task)

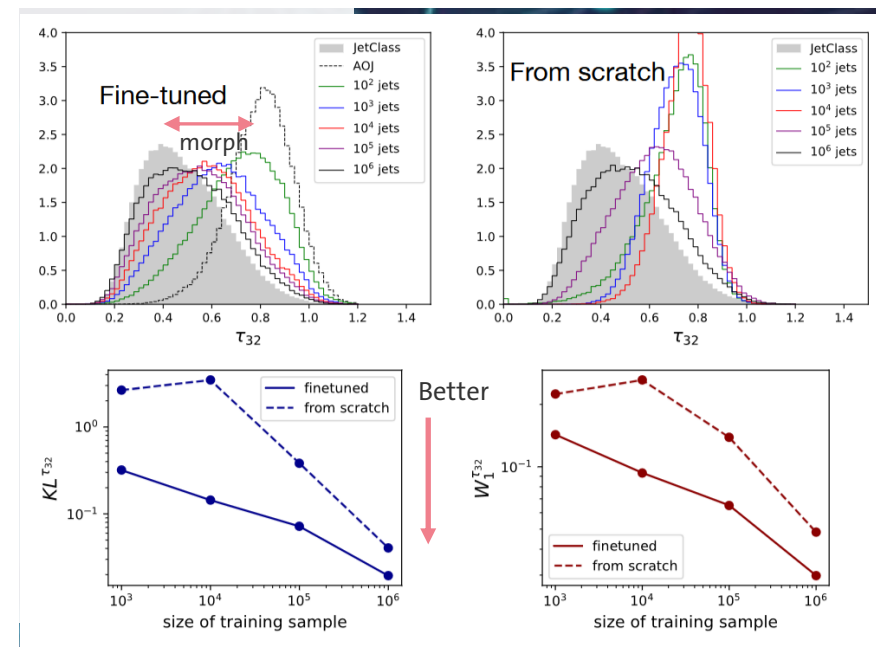
Pretrain on Aspen Open Jets (real, unlabeled, mostly q/g)
→ fine-tune on JetClass (simulated, labeled, different jet types)

Finding: pretraining on real data helps downstream tasks,
particularly for hard-to-model observables like n-subjettiness τ_{32} .

First foundation model for collider physics trained on real data.

The model only requires sequences of integers — any data that can be tokenized
can use the same architecture without modification.

Can be used for example for calorimeter shower simulation



Strategy D: Contrastive Learning — RS3L

Harris, Kagan, Krupa, Maier, Woodward — 2024 [arXiv: 2403.07066]

Re-Simulation-based Self-Supervised Learning:

Augmentation strategy unique to HEP: for the same hard process (e.g., $pp \rightarrow t\bar{t}$), re-run the stochastic simulation steps to get different but related jets.

Hard process $\rightarrow$ Parton shower + Hadronization + Detector simulation

$\hookrightarrow$ Run TWICE with different random seeds before the parton shower

$\hookrightarrow$ Produces two different jets from the SAME physics

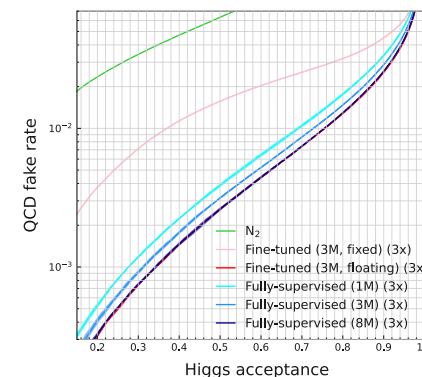
Positive pair: same hard process, different hadronization and simulations $\rightarrow$ should be CLOSE in latent space

Negative pair: different hard processes $\rightarrow$ should be FAR APART

Use a Contrastive loss function (e.g. InfoNCE)

Key properties:

- No labels needed — only simulation infrastructure for re-running events
- Aims for domain completeness: cover the stochastic phase space of simulation
- Fine-tune to supervised classification $\rightarrow$ competitive with other approaches
- **Limitation: requires simulation (can't pretrain directly on real data)**



Strategy E: Hybrid Pretraining — OmniLearn

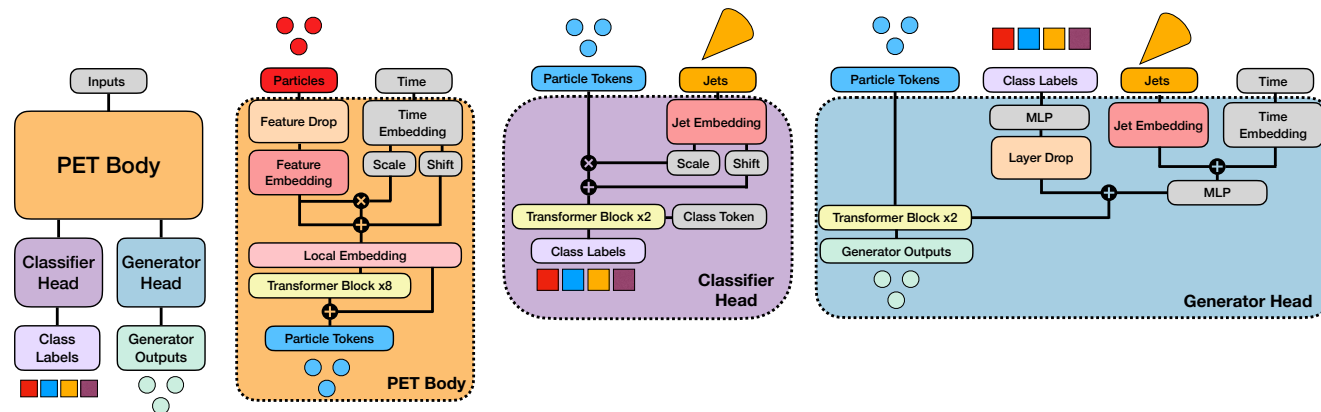
Mikuni & Nachman — 2024 [arXiv: 2404.16091]

Combines generation and supervised classification in a single pretraining objective.

Architecture: Transformer encoder backbone (PET)
+ Diffusion head (EPiC-FM) for generation
+ MLP head for classification

Pretraining loss: $L = L(\text{classification}) + L(\text{diffusion})$

Features: jet + constituent kinematics, particle ID



Downstream tasks — broadest coverage of any HEP Foundation Model:

- Classification: jet tagging across different jet types, detectors, AND collision systems
- Anomaly detection: unsupervised search for new physics
- Conditional generation: generate jets of specified type
- Reweighting: match simulation to data
- Unfolding: correct for detector effects

OmniLearn is currently the most multi-task foundation model in HEP.

	AUC	Acc	$1/\epsilon_B$	
			$\epsilon_S = 0.5$	$\epsilon_S = 0.8$
ResNet 50	0.885	0.803	21.4	5.13
EFN	0.901	0.819	26.6	6.12
h1DNN	0.938	0.863	51.5	10.5
DNN	0.942	0.868	67.7	12.0
PFN	0.954	0.882	108.0	15.9
ParticleNet	0.961	0.894	153.7	20.4
PET classifier (4M)	0.959	0.890	146.5	19.4
OMNILEARN (4M)	0.961	0.894	172.1	20.8
PET classifier (40M)	0.964	0.898	201.4	23.6
OMNILEARN (40M)	0.965	0.899	207.3	24.1

Other Recent Models (Late 2024 – 2025)

L-GATr (2024)

[*Brehmer et al. \(arXiv:2411.00446\)*](#)

Lorentz-equivariant transformer using geometric algebra (multivectors).
Encodes Lorentz symmetry directly in the architecture.
Current state-of-the-art for supervised jet classification.
Pretrain on supervised classification, fine-tune to other datasets.

J-JEPA / HEP-JEPA (2024–25)

[*Katel et al. \(arXiv:2412.0533\)*](#) / [*Bardhan et al. \(arXiv:2502.0393\)*](#)

Joint-Embedding Predictive Architecture (inspired by LeCun).
Re-cluster jets into subjects (J-JEPA) or patches (HEP-JEPA).
Predict masked-out parts in latent space (not feature space).
Self-supervised, no labels needed.

Bumblebee (2024)

[*Wildridge et al. \(arXiv:2412.0786\)*](#)

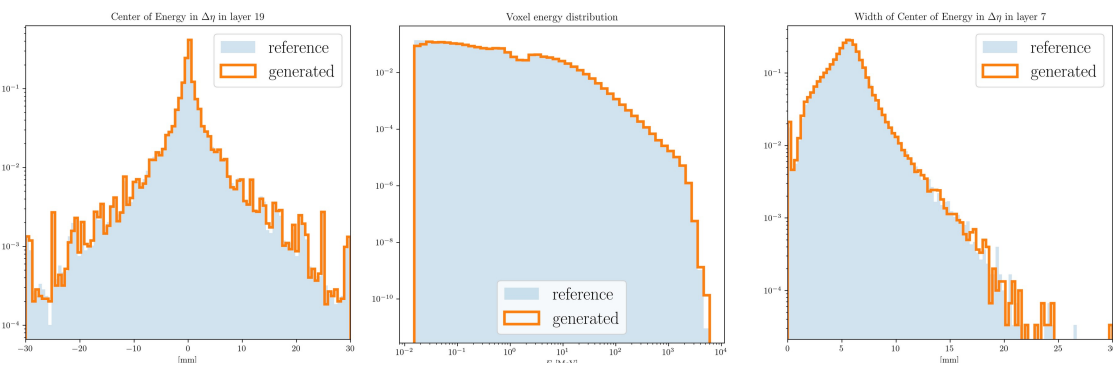
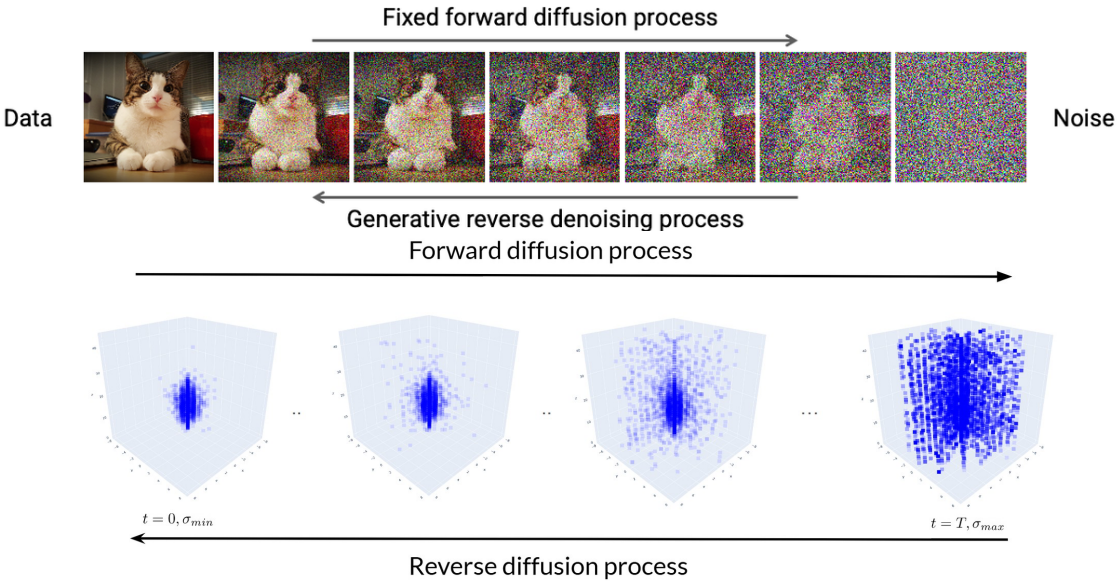
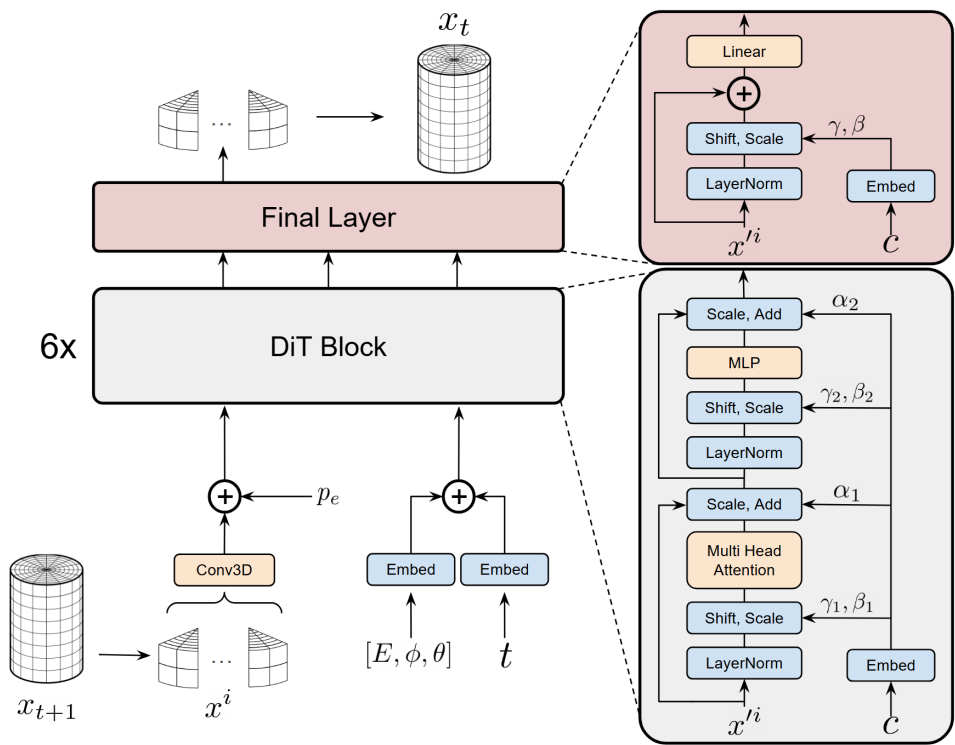
First event-level (not just jet-level) foundation model.
Uses high-level features (jets as objects, not constituent-level).
Pretrain: masked event reconstruction (predict masked particles).
Downstream: top quark reconstruction, event classification.

See also FM Review from A. Hallin — SciPost Phys. Proc. (2025) — [arXiv: 2509.21434](#)

Foundation Model for Fast Calorimeter Simulation

Simulating calorimeter shower using Geant4 is a time consuming process for HEP experiments

CaloDiT-2 model developed at CERN is based on diffusion and visual transformers.

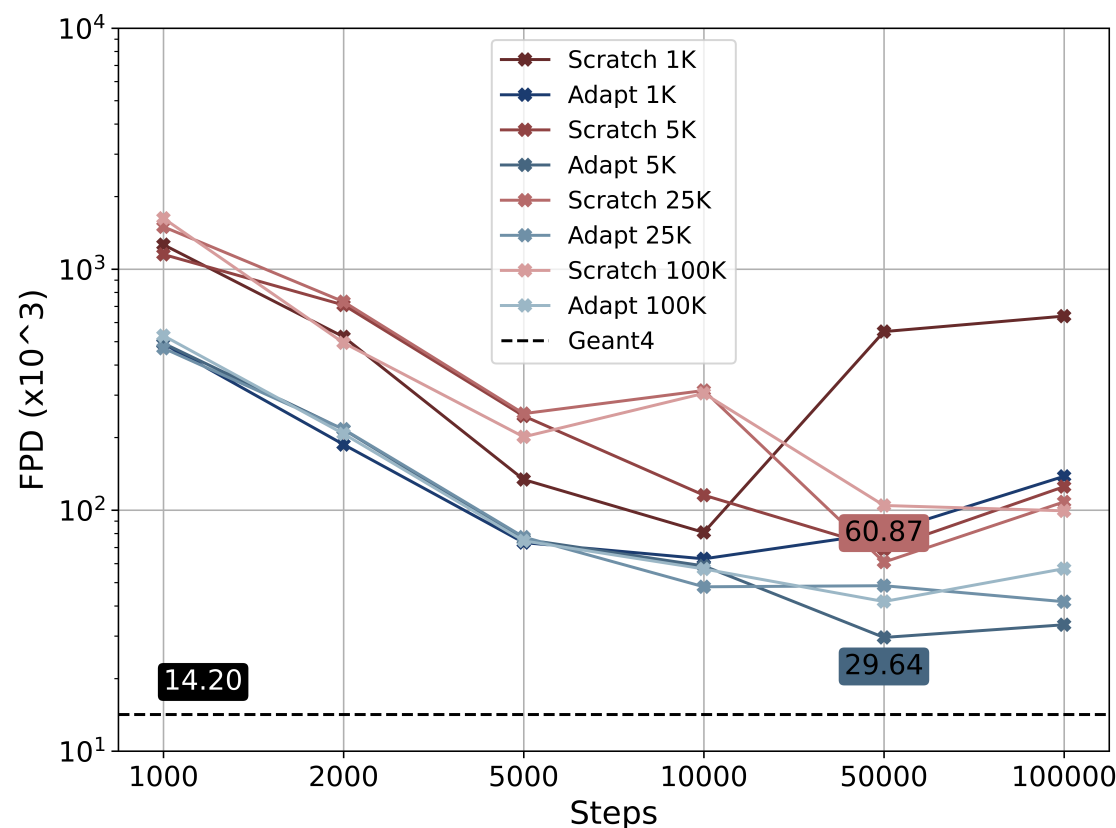
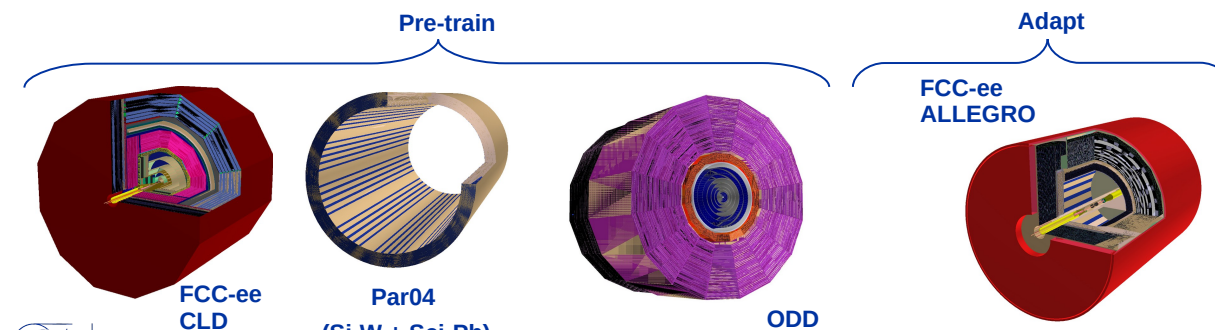


Very good results in reproducing shower observables

[P. Raikwar et al. \(arXiv:2509.0770\)](https://arxiv.org/abs/2509.0770)

Generalisation Capabilities of CaloDiT-2

Pre-train model on some datasets and fine-tune (adapt) on target data



compared to training from scratch:

- ~ 20% less training steps
- ~ 25 % less data

Some of the available datasets for training Foundation Models for HEP

Simulated data (with labels)

- **JetClass**, 100M jets quark/gluon + hadronic top jets. 10 class labels available (see [Zenodo](#))
- **Top Quark tagging** dataset : 2 M jets from top quark decaying plus lighter quarks (for binary classification) (see [Zenodo](#))
 - used more for fine-tuning the models (e.g. for HEP-JEPA model)
- **ColliderML** : 1M events with full simulation of OpenData detector), available for several tasks (jet tagging, simulation, tracking, physics analysis) (available in [HaggingFace](#))

Real data (unlabelled)

- **Aspen Jet Data** set: 180 M boosted jets derived from CMS Open data (see [link](#)) and ready for ML tasks
- [ATLAS](#) and CMS Open data available in the CERN [Open data portal](#)

Outlook and Open Questions

Where do we go from here?

The Road Ahead: HL-LHC and Beyond

The HL-LHC (from 2030) will deliver $\sim 10\times$ more data than in Run 3.

Computing requirements will increase dramatically — foundation models can help:

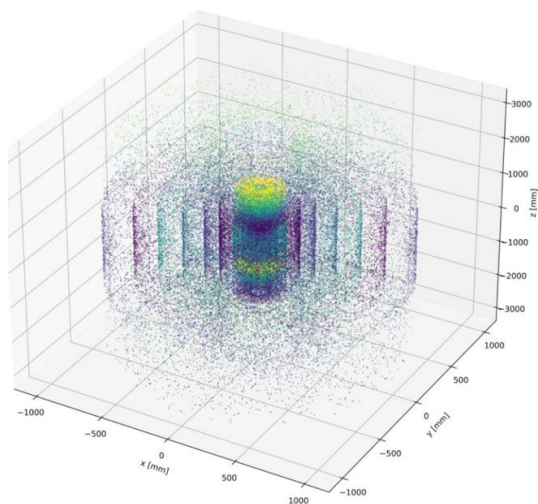
- Pre-train once on massive (unlabeled) data $\rightarrow$ amortize cost across all analyses
- Fine-tune quickly for each new analysis with small labeled samples
- Share pre-trained models across experiments (ATLAS $\leftrightarrow$ CMS)
- Leverage real data for pre-training — no simulation required
- Multi-modal foundation models: combine tracker + calorimeter + muon data

What we need:

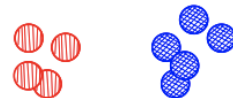
- Standardized benchmarks for cross-task evaluation
- Scaling studies: how large do HEP foundation models need to be?
- Community infrastructure: shared compute, open-source models, collaboration

End-to-End ?

([from S. Caron](#))



$O(10000)$ hits (no pre-processing)

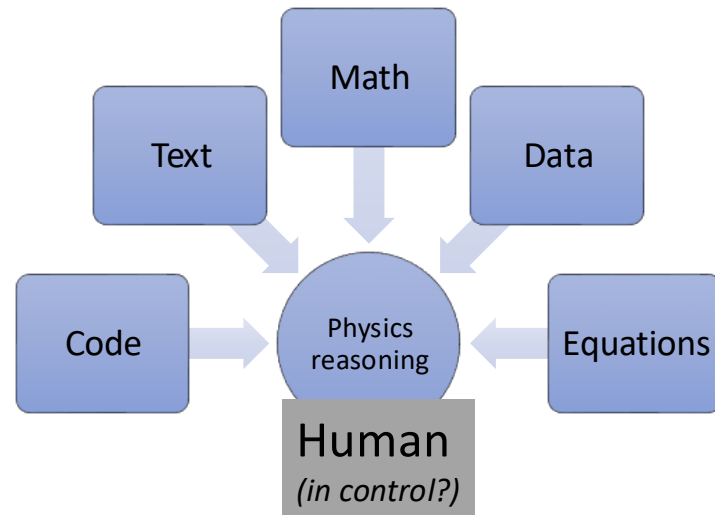


- challenging: interpretability, control, uncertainty estimation, etc.
- Need flexible + interpretable model
- Chat with data ?

What Are Large Physics Models (LPMs)?

([from S. Caron](#))

- Inspired by chatgpt like foundation models (LLMs, vision models, multimodal models)
- LPMs are **AI models trained natively on physics data, structure, and tasks** (*under the control of the science community*)
- Go beyond chatbots → integrate **symbolic reasoning, simulation, mathematics, and data-driven inference via links to physics (raw) data**



LPMs are built for physics reasoning and integration across our modalities.

- Require collaboration between physicists, computer scientists and philosophers of science
- Need access to full experimental data (use federated learning for dealing with ownership issues ?) and large computing infrastructures (e.g AI GigaFactory)

Summary

- 1 Foundation models pre-train on large data to learn general representations, then fine-tune small task-specific heads for downstream tasks.
- 2 HEP is an excellent setting: massive datasets, high-quality simulations, many tasks sharing the same pattern recognition needs.
- 3 Multiple pre-training strategies have been explored: supervised (ParT), masked (MPM), generative (OmniJet- α), contrastive (RS3L), hybrid (OmniLearn). All show improvement over training from scratch.
- 4 The field is young and moving fast. Scaling, standardized evaluation, and experiment-wide collaboration are the key next steps for the HL-LHC era.
- 5 Future: Build Large Physics Model able to perform multiple tasks end to end

Thank you!

Lorenzo.Moneta@cern.ch

