TrackingBERT: A Language Model for Particle Tracking

Andris Huang, *Xiangyang Ju*, Yash Melkani,

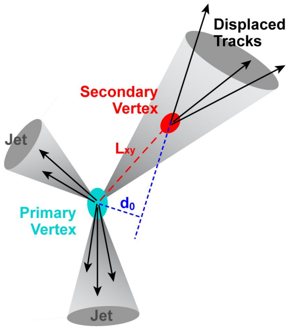
co-authored with Alina Lazar, Daniel Murnane, Minh-Tuan Pham, Paolo Calafiura

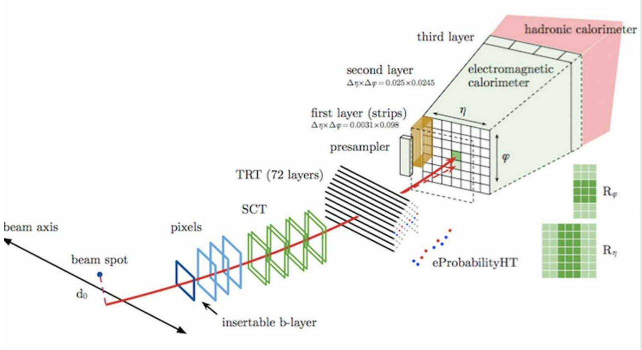
AI4EIC Annual Workshop, 11/30/2023

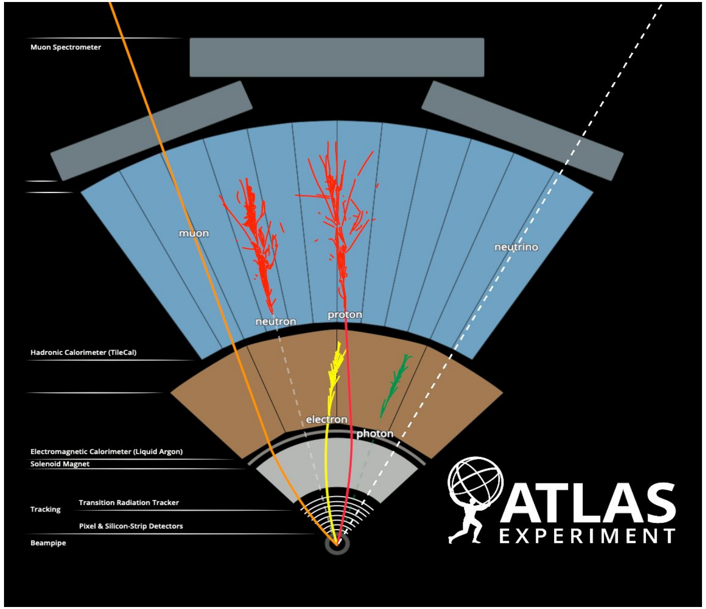
Particle tracking

Particle tracking is used in almost all physics object reconstruction

- Leptons
- Jet flavor tagging
- Primary vertices, displaced vertices
- Pileup removal for jets and missing energy







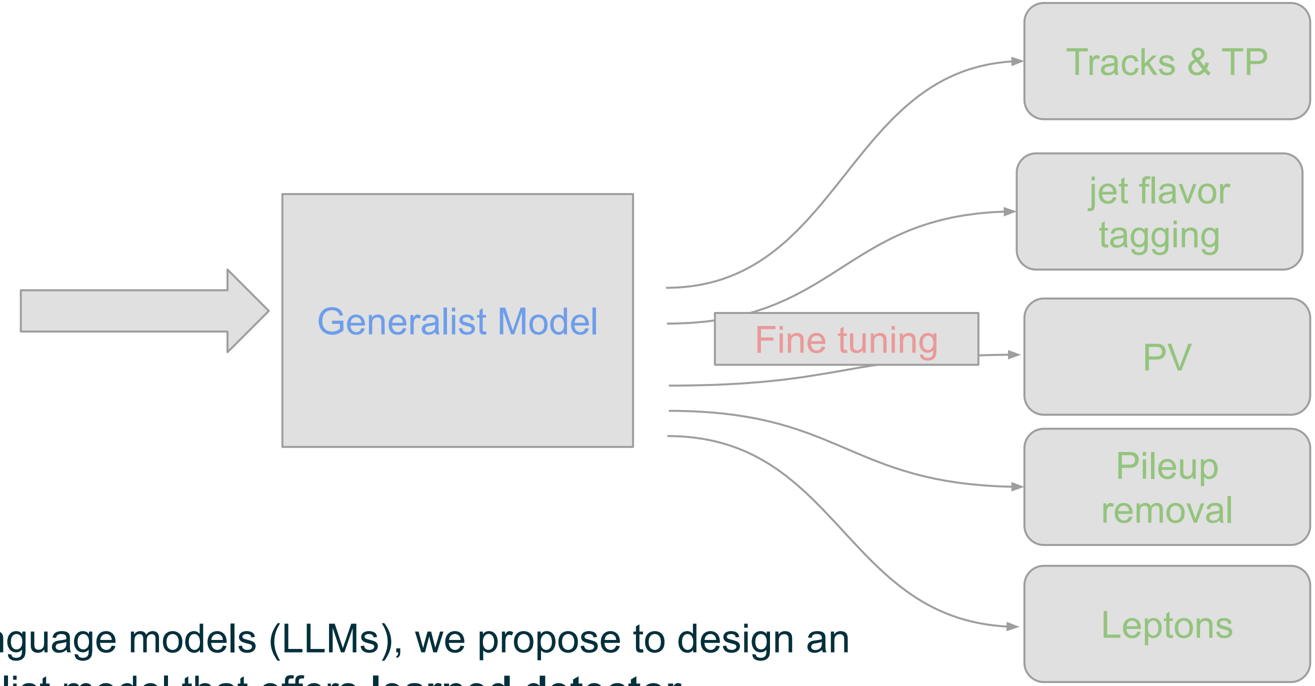
Machine Learning for Tracking-related tasks

Particle tracking is used in almost all physics object reconstruction

- Leptons → HeteroGNN(Huang, 2023)
- Jet flavor tagging → Transformers(Qu, 2022)
- Primary vertices, displaced vertices → DNN(Akar, 2023)
- Pileup removal for jets and missing energy → PUMML(Komiske, 2017), Attention(Maier, 2021)
- Tracking finding → GNN(Ju, 2021)

→ One model for one task. However, these tasks are so deeply intertwined that factorizing them will inevitably lose information and hurt overall performance

Generalist Model for particle tracking



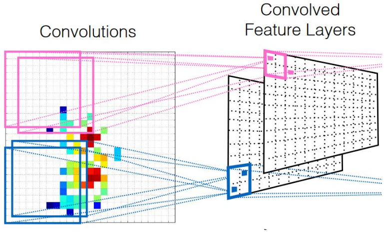
Inspired by large language models (LLMs), we propose to design an intermediate generalist model that offers **learned detector encodings** for various particle tracking tasks.

Data representation and ML

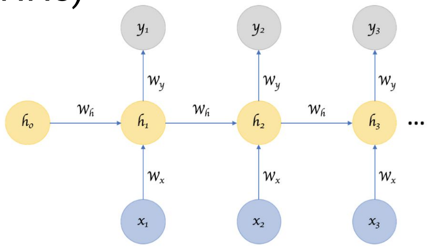
Data is a vector
→ multilayer perceptrons
(MLPs)

$$\begin{pmatrix} x_1 & x_2 & \cdots & x_n \end{pmatrix} \times \begin{pmatrix} w_{11} & w_{12} & \cdots & w_{1m} \\ w_{21} & w_{22} & \cdots & w_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n1} & w_{m2} & \cdots & w_{nm} \end{pmatrix}$$

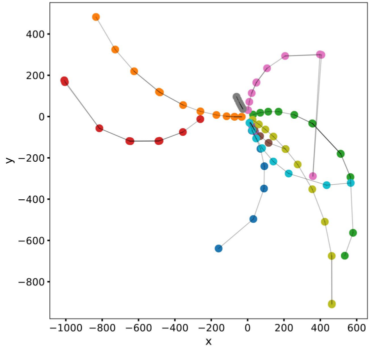
Data is an image or grid
→ Convolutional Neural Network
(CNNs)



Data is a sequence
→ Recurrent Neural Network
(RNNs)



Data is of dynamic size, irregular shape, sparse density
→ Graph Neural Network

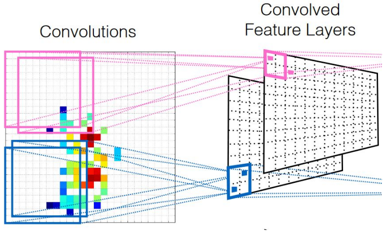


Data representation and ML

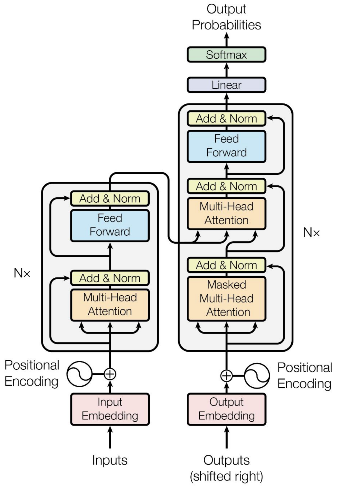
Data is a vector
→ multilayer perceptrons (MLPs)

$$\begin{pmatrix} x_1 & x_2 & \cdots & x_n \end{pmatrix} \times \begin{pmatrix} w_{11} & w_{12} & \cdots & w_{1m} \\ w_{21} & w_{22} & \cdots & w_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n1} & w_{m2} & \cdots & w_{nm} \end{pmatrix}$$

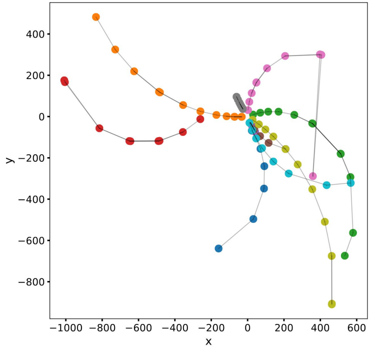
Data is an image or grid
→ Convolutional Neural Network (CNNs)



Data is a sequence
→ Transformers → LLMs

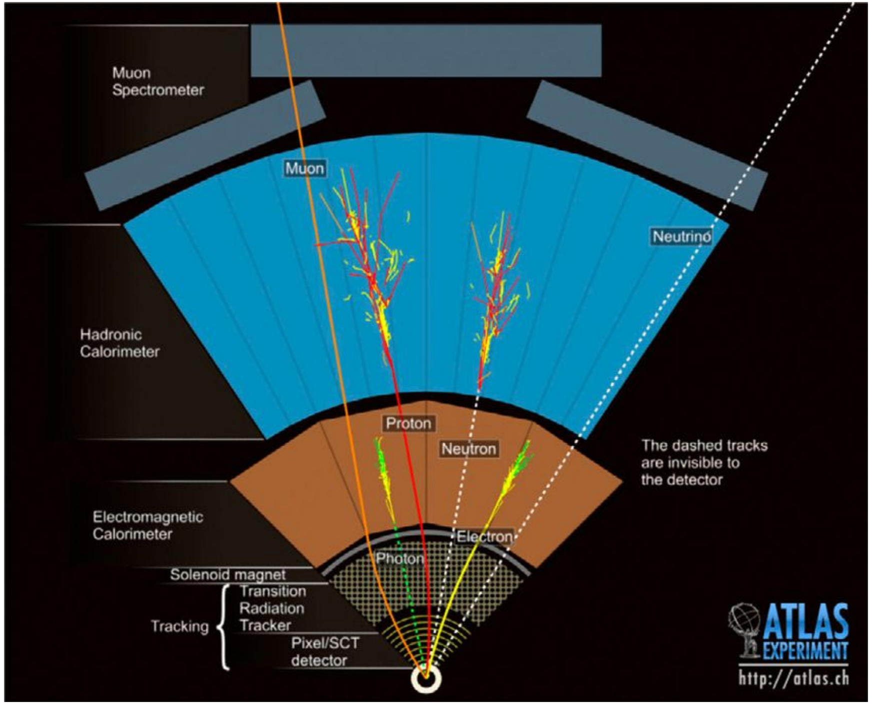


Data is of dynamic size, irregular shape, sparse density
→ Graph Neural Network (GNNs)



NLP vs ATLAS

Analogy between NLP and ATLAS	
Detector elements	Words
All detector elements	Vocabulary
Particle trajectories or showers	Sentences
Collision Events	Paragraphs
Events from the same physics process	Sections



BERT

arxiv:1810.04805

Pre-training of Deep Bidirectional Transformers for Language Understanding

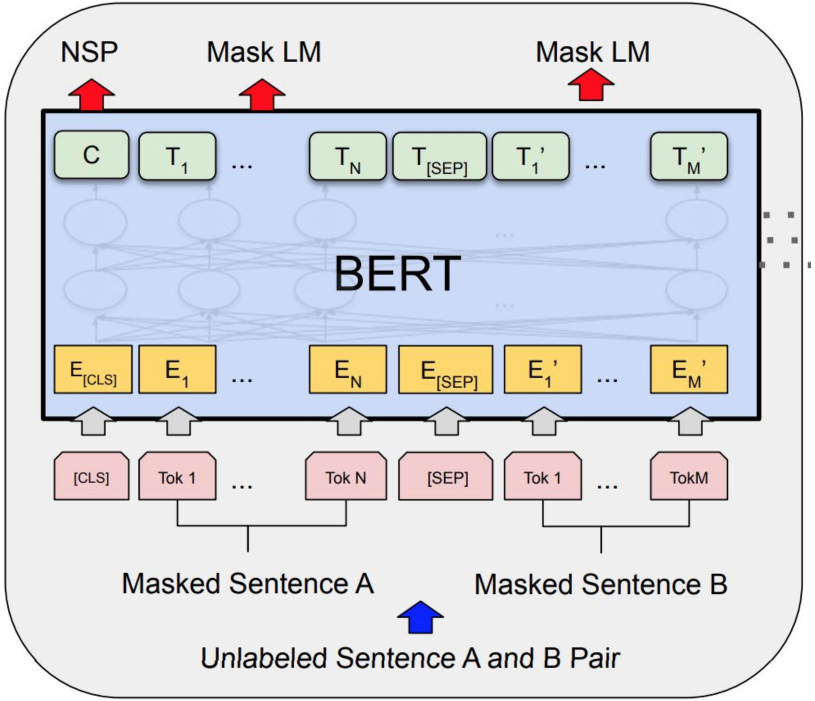
Inputs

- A pair of sentences (SA, SB)
- Randomly mask some words in each sentence
- Randomly swap the two sentences

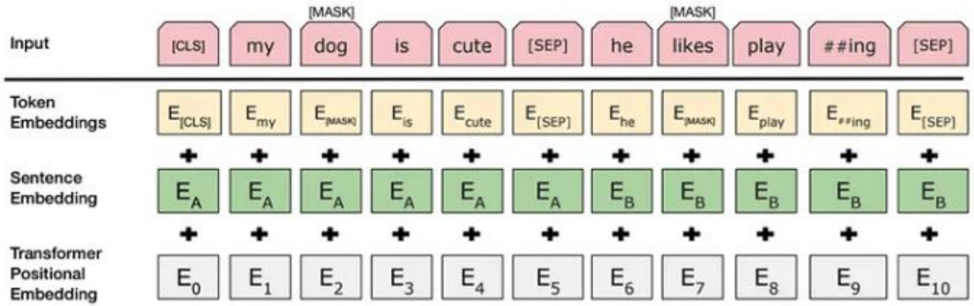
Outputs: continuous embedding for each word in the dictio

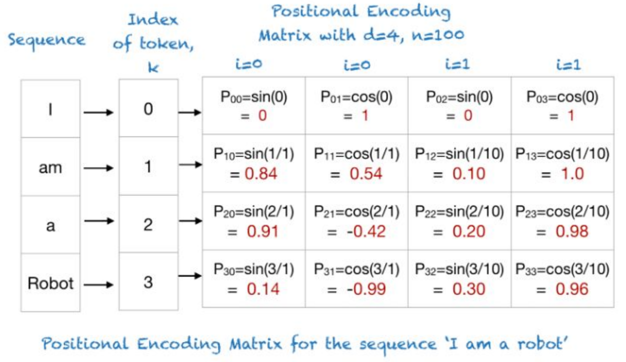
Loss Functions

- Masked Language Modelling (MLM): predict the mask
- Next Sentence Prediction (NSP): predict whether sen



BERT inputs





Sequence	Index of token, k	Positional Encoding Matrix with d=4, n=100 i=0	i=0	i=1	i=1
I	0	P_{00}=sin(0) = 0	P_{01}=cos(0) = 1	P_{02}=sin(0) = 0	P_{03}=cos(0) = 1
am	1	P_{10}=sin(1/1) = 0.84	P_{11}=cos(1/1) = 0.54	P_{12}=sin(1/10) = 0.10	P_{13}=cos(1/10) = 1.0
a	2	P_{20}=sin(2/1) = 0.91	P_{21}=cos(2/1) = -0.42	P_{22}=sin(2/10) = 0.20	P_{23}=cos(2/10) = 0.98
Robot	3	P_{30}=sin(3/1) = 0.14	P_{31}=cos(3/1) = -0.99	P_{32}=sin(3/10) = 0.30	P_{33}=cos(3/10) = 0.96

Positional Encoding Matrix for the sequence 'I am a robot'

Token Embeddings

- Indices of the words in dictionary

Sentence Embeddings

- Distinction for each sentence in the input pair

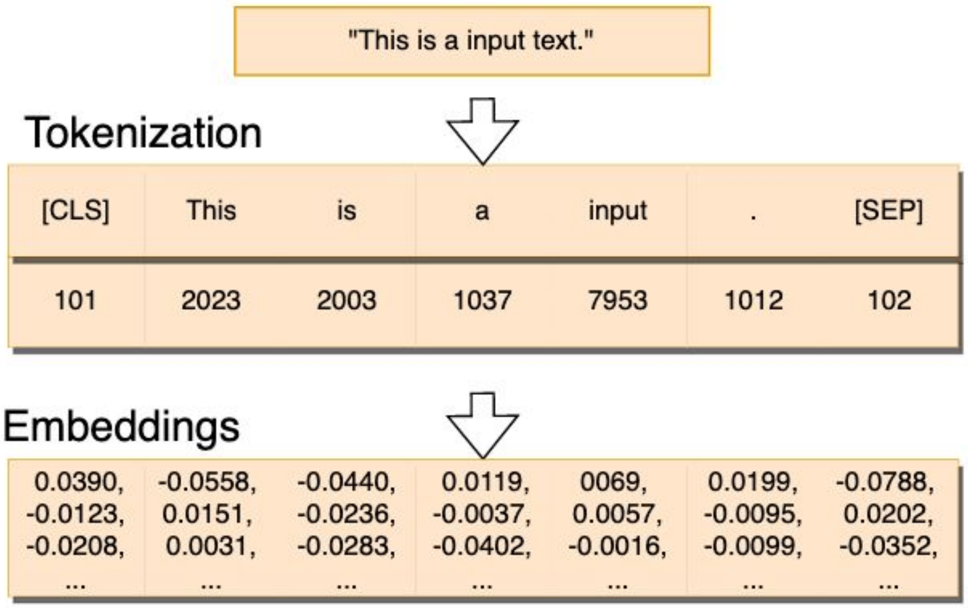
Position Embedding:

- Encode each word's position into a vector

$$P(k,\, 2i+1) = \cos\left(\frac{k}{n^{2i/d}}\right) \qquad P(k,\, 2i) = \sin\left(\frac{k}{n^{2i/d}}\right)$$

Sentence vs Tracks

Tracks are represented by a list of detector modules



[(x, y, z), …, (x, y, z)]

⇩

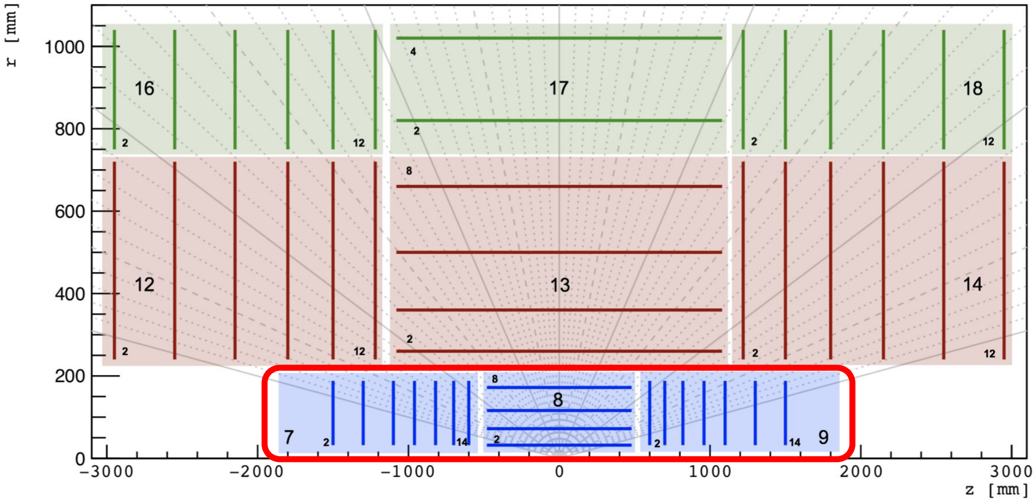
[CLS]	(x, y, z)	…	(x, y, z)	[SEP]
2	UMID	…	UMID	3

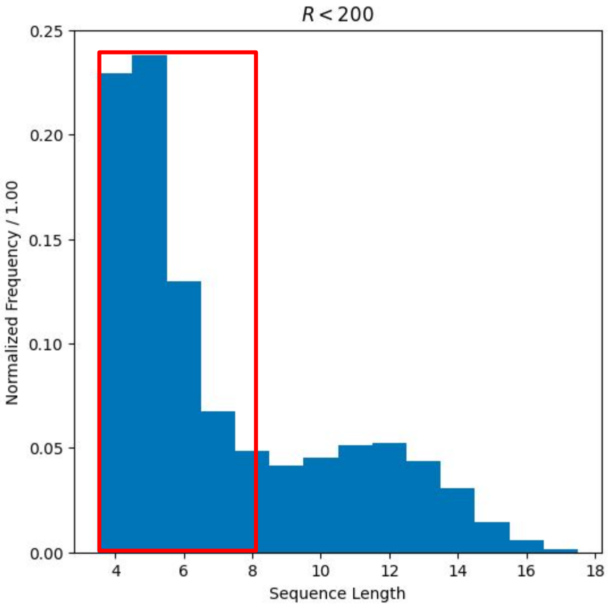
⇩

Embeddings

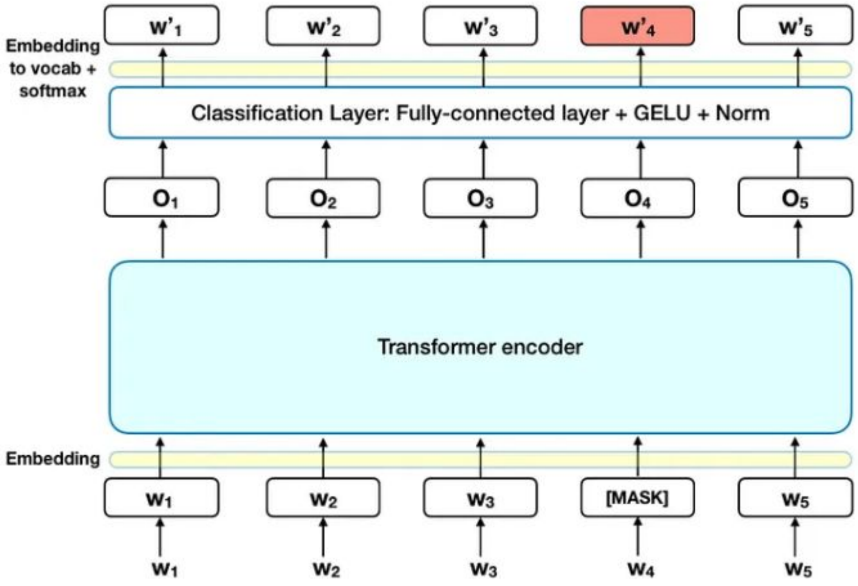
Input data

Focusing on Pixel detectors and tracks with 4 - 8 spacepoints. About 4M tracks are selected for training.





TrackingBert



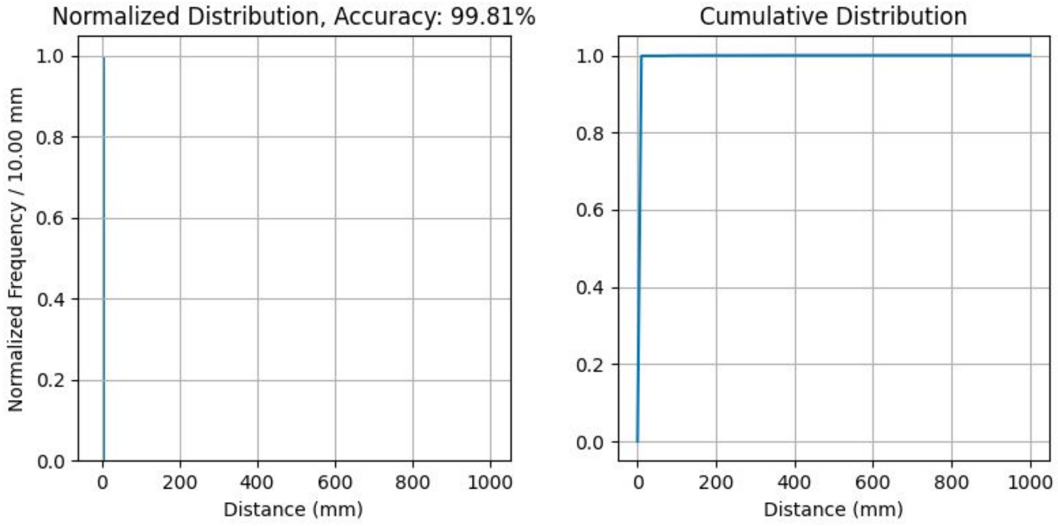
[Track A, Track B]

- Tune parameters of the Transformer model → *1M* trainable parameters
- Gradually increase the mask rate during the training: 15% → 30% → 50%
- Randomly select two tracks A, B; track A with higher pT
- Two tasks:
  - Predict the masked detector modules (UMID)
  - Predict if track B is with higher pT than track A

Results for first track

Accuracy in predicting masked detector modules

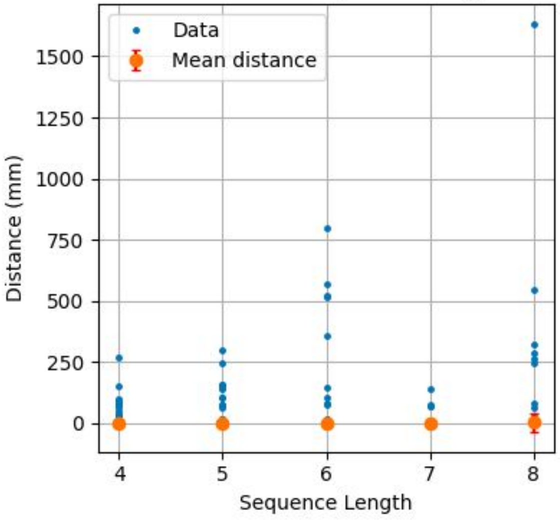
- Mask 1 module in the *first* track and ask the model to predict the masked module.
- Evaluate the distance between the predicted module and the true module.



Results on first track

The impact on the track length

- Mask the first module, middle modules, or the last module to check the performance



- No clear dependances on the sequence lengths
- The same test is performed on the second track → Mask detector modules in the second particle
- And we observe a similar performance

Conclusions

- Our work is the first application of (large) language models in HEP, thanks to the new data presentation for particles: ***tokenized data elements***
    - Particles can be presented as a sequence of detector-element tokens stemmed from the particle interacting with the detector

- We applied a language model (BERT) to new data presentation and obtained *a **novel detector representation*** learned from unsupervised training
    - We found larger training data and larger models often resulting in better results
    - And the model can accurately predict the masked detector modules

The talk was presented in the 2023 Connected To the Dots conference. Link to the proceeding.

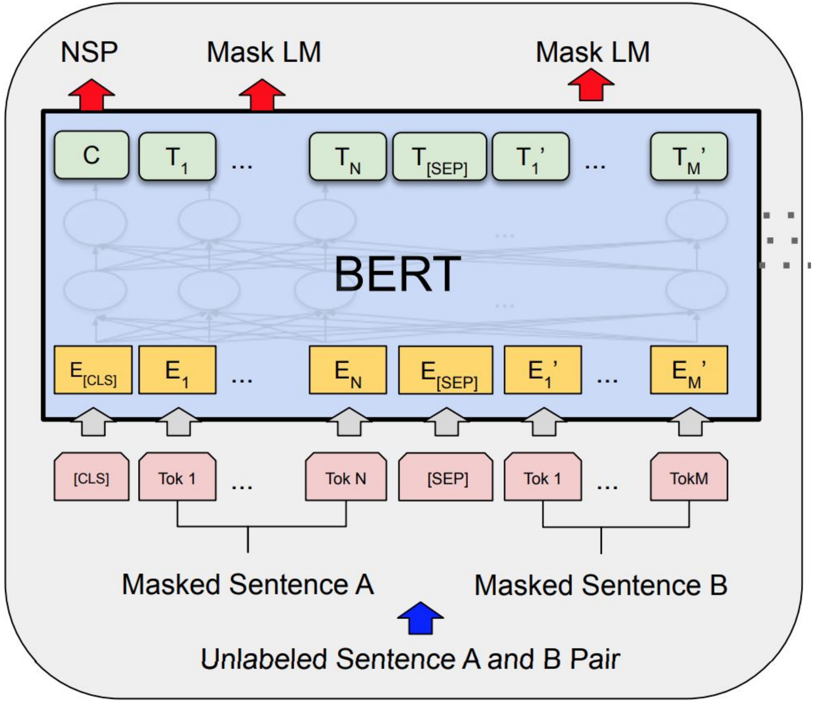
Outlooks

Short term aims:

- extract the detector module embedding from BERT to have a "deep representation of the Pixel detector"
- apply the learned detector presentation for other tasks, such as metric learning-based graph construction, end-to-end track finding

Long term aims:

- build a deep representation for calorimeters
- apply the representation for particle reconstructions



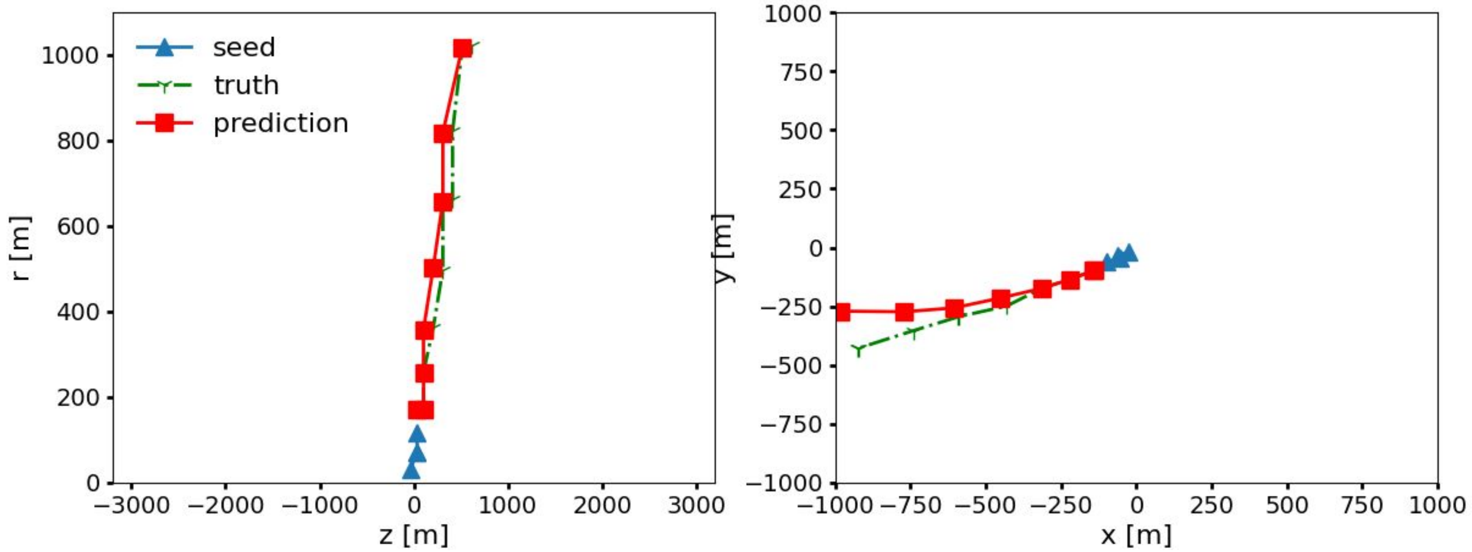
GPT for particle tracking

A different masking scheme

- Unlike BERT's bidirectional context, GPT models are trained using an autoregressive approach, where they predict the next word based solely on the preceding words
- The model performance follows the scaling law;
  - OpenAI can accurately predict what the evaluation loss would be if more data and computational resources were available

GPT extrapolating seed hits

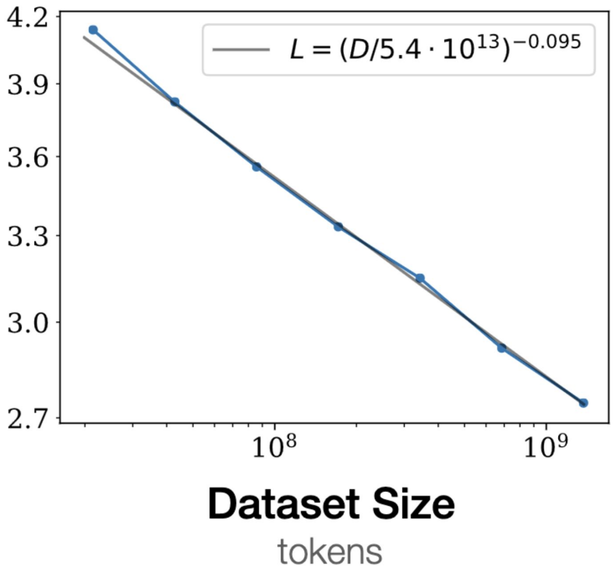
We sample the next hits based on the probability distribution predicted by GPT

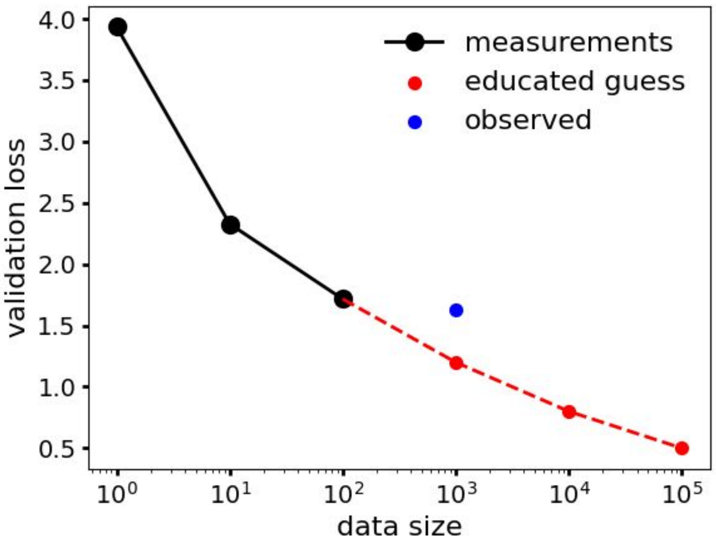


Scaling Laws for Neutral Language Model

arxiv:2001.08361

Scaling Law





Did not increase model size when training the model with 1000 events

More interesting research areas

- ***Detector tokenization***. For understanding the detector, which language should we use? Raw detector readouts?

- ***Masking*** is the sole way of expressing your intention on what LLMs should learn from data
    - Model = f(*physics* | *unmasked information*)
    - BERT vs GPT, choose your context
    - ***Hierarchical Masking.*** Can we treat both the low-level detector information and high-level physics objects as tokens and mask the high-level physics objects?

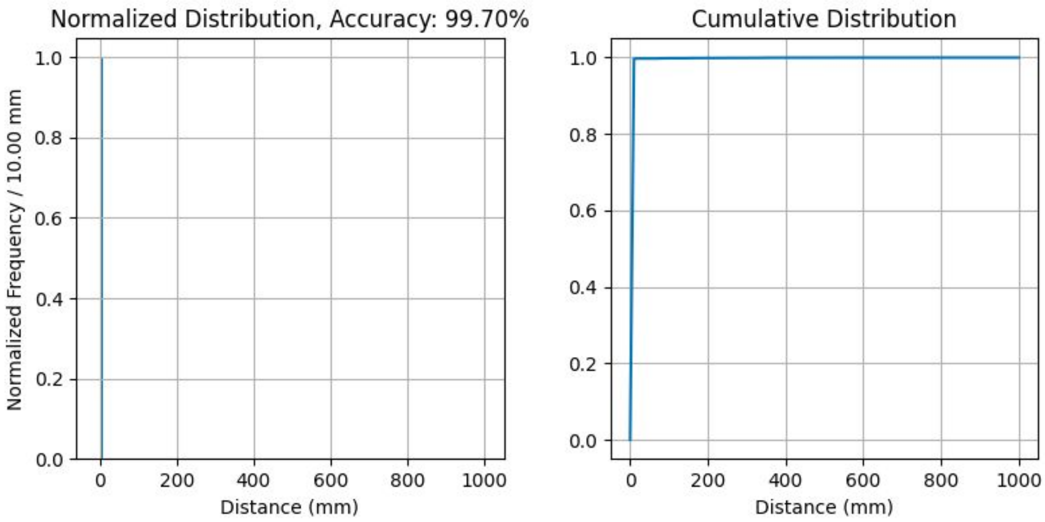
- ***Guided Trial and Error*** means one should always start from small and simple to large and complex, and verify the scaling law

Backup Slides

Results on second track

Accuracy in predicting masked detector modules

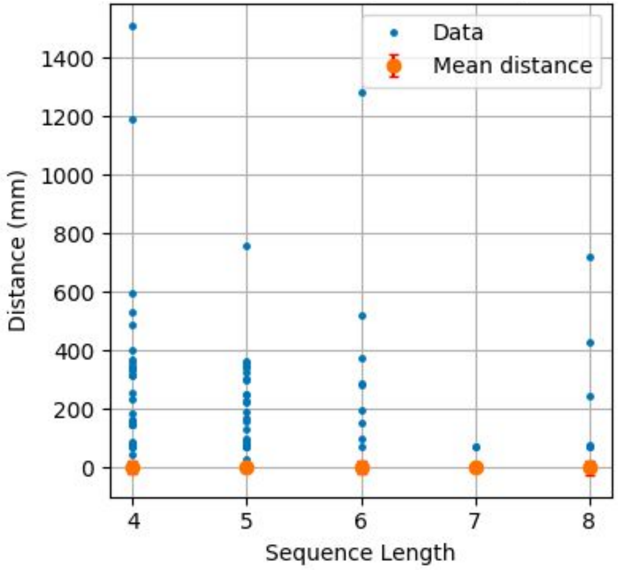
- Mask 1 module in the *second* track and ask the model to predict the masked module.
- Evaluate the distance between the predicted module and the true module.



Results for second track

The impact on the mask position and track length

- Mask the first module, middle modules, or the last module to check the performance



GPT Training data

From: https://arxiv.org/pdf/2303.10158.pdf