Masked Particle Modelling

CCNSB Seminar

Abhiram Tilak

CND Dual Degree
IIIT Hyderabad

October 14, 2024

Table of Contents

1 Introduction

2 Background

Introduction

This paper proposes a self-supervised model which uses unordered sets of particles with paramters as input and aims to build a generic foundational model.

- The aim is to use this large foundation model and fine-tune it to be used in variety of down-stream tasks.

Introduction

This paper proposes a self-supervised model which uses unordered sets of particles with paramters as input and aims to build a generic foundational model.

- The aim is to use this large foundation model and fine-tune it to be used in variety of down-stream tasks.
- Proposes a masking strategy based on models like BERT and BEiT to mask a set of particles in each jet.

Introduction

This paper proposes a self-supervised model which uses unordered sets of particles with paramters as input and aims to build a generic foundational model.

- The aim is to use this large foundation model and fine-tune it to be used in variety of down-stream tasks.
- Proposes a masking strategy based on models like BERT and BEiT to mask a set of particles in each jet.
- The model that is built should be capable of inferring the original particles using the information from other particles.

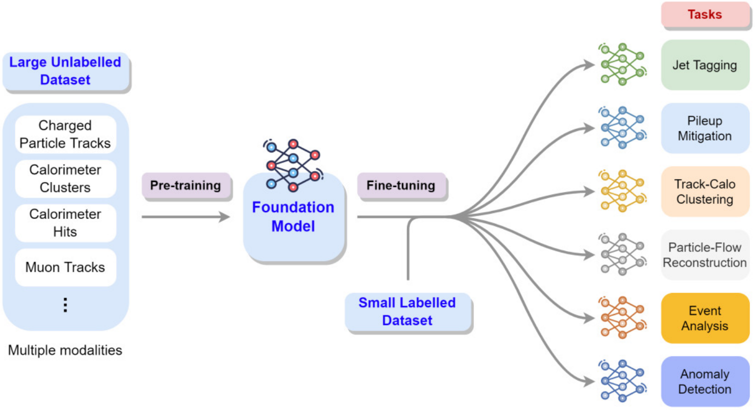
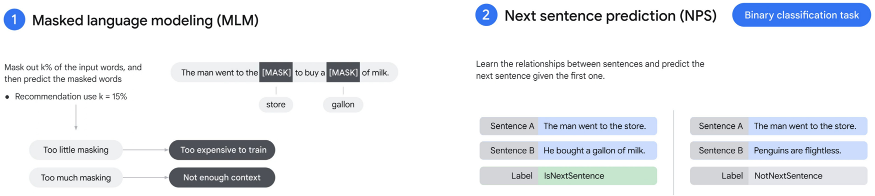


Figure: Overview of the proposed Large foundation Model

Background

The idea of masking in MPM comes from Masked Language Models widely used in the NLP and Computer Vision. Some of those popular models commnoly referred are:

- BERT (Bidirectional Encoder Representation from Transformers):
  BERT is a language model introduced in October 2018 by researchers at Google. It learns to represent text as a sequence of vectors using self-supervised learning. It uses the encoder-only transformer architecture.



  BERT uses a bi-directional approach considering both the left and right context of words in a sentence, instead of analyzing the text sequentially, BERT looks at all the words in a sentence simultaneously.

Background

- BEiT (Bidirectional Encoder for Image Transformers):
  - BEiT is a self-supervised vision representation model that uses a masked image modeling task to pretrain vision Transformers, achieving competitive results on downstream tasks like image classification and semantic segmentation.
  - Unlike BERT, the BEiT model accepts continuous inputs so before pre-training, we learn an "image tokenizer" via autoencoding-style reconstruction, where an image is tokenized into discrete visual tokens according to the learned vocabulary.
  - Labels of different image patches were defined using a Vector Quantized Variational AutoEncoder (VQ-VAE). A VQ-VAE uses an encoder to map a set of inputs to latent vectors, which are subsequently projected onto the nearest element within a finite codeboo

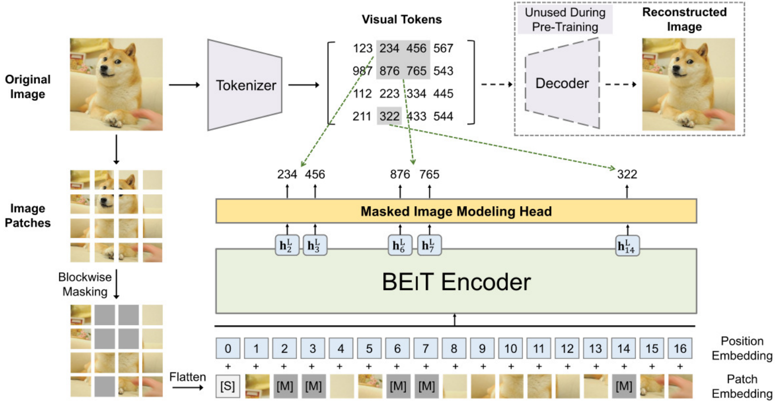


Figure: Overview of BEiT pre-training. The patches are fed to a backbone vision Transformer. The pre-training task aims at predicting the visual tokens of the original image based on the encoding vectors of the corrupted image

Why we chose this approach?

- **Why not supervised models?** Unlike supervised learning, which typically acquires limited domain representations and focuses on a few key features for high prediction accuracy that must be learned anew for each task, SSL aims to learn generic representations summarizing domain features that prove useful across various downstream tasks. SSL tasks can be formulated on unlabeled data.

Why we chose this approach?

- **Why not supervised models?** Unlike supervised learning, which typically acquires limited domain representations and focuses on a few key features for high prediction accuracy that must be learned anew for each task, SSL aims to learn generic representations summarizing domain features that prove useful across various downstream tasks. SSL tasks can be formulated on unlabeled data.
- **Why to mask the tokens?** In BERT the MLM model gave it a dramatic improvement over previous state-of-the-art models and served as on of the earliest examples of large language model.
  Traditional language models process text sequentially, either from left to right or right to left. This method limits the model's awareness to the immediate context preceding the target word.
  By masking random tokens, the model is forced to use both left and right context to predict the masked word. This enables the model to learn bidirectional representations, unlike traditional left-to-right language model

Challenges in using HEP data for Foundation Models

The results from models in NLP seem promising, but there are a few fundamental problems in dataset in the context of HEP which differ the training scheme used, which need to be addressed:

- **Tokenization**: Unlike language models, which operate on a finite and discrete vocabulary of words, many of the features one that describe particles are continuous, such as momentum, direction, distance of closest approach to the primary collision, etc. This changes the way we pass the input or parse the output when pre-training.

Challenges in using HEP data for Foundation Models

The results from models in NLP seem promising, but there are a few fundamental problems in dataset in the context of HEP which differ the training scheme used, which need to be addressed:

- **Tokenization**: Unlike language models, which operate on a finite and discrete vocabulary of words, many of the features one that describe particles are continuous, such as momentum, direction, distance of closest approach to the primary collision, etc. This changes the way we pass the input or parse the output when pre-training.
- **Permuation of datasets**: The particles in a jet don't have a specific order, so it makes sense to use a model that treats them all equally. However, if we replace all masked particles with the same placeholder, the model can't tell them apart. This means it would produce the same output for all masked particles, which isn't very useful. To avoid this problem, we need to either give the particles an order or add some way for them to interact with each other uniquely.

Tokenization Strategies

To tokenize the inputs the following methods have been experimented:

1. Simple binning: Each feature is divided into a finite set of ranges, creating discrete bins.
   Limitation: Lacks contextual information about relationships between particles

Tokenization Strategies

To tokenize the inputs the following methods have been experimented:

1. Simple binning: Each feature is divided into a finite set of ranges, creating discrete bins.
   Limitation: Lacks contextual information about relationships between particles
2. Vector Quantized Variational AutoEncoder (VQ-VAE): Uses an encoder to map inputs to latent vectors. Projects these vectors onto the nearest element in a finite codebook. Incorporates context from all input elements. Each particle is encoded to a single codebook element, considering all other particles in the jet.
   Limitation: More complex to implement and potentially computationally expensive

Tokenization Strategies

To tokenize the inputs the following methods have been experimented:

1. Simple binning: Each feature is divided into a finite set of ranges, creating discrete bins.
   Limitation: Lacks contextual information about relationships between particles
2. Vector Quantized Variational AutoEncoder (VQ-VAE): Uses an encoder to map inputs to latent vectors. Projects these vectors onto the nearest element in a finite codebook. Incorporates context from all input elements. Each particle is encoded to a single codebook element, considering all other particles in the jet.
   Limitation: More complex to implement and potentially computationally expensive
3. K-means clustering:k Uses the K-means++ algorithm to define clusters. Each cluster is assigned an index, which is used as the target label. This method is context-independent, unlike the VQ-VAE approach
   Limitation: Context-independent, may miss important relationships between particles in a jet

Ordering Strategies

The following ordering strategies have been experimented with:

- **Ordering the input to the backbone:** Order particles by decreasing transverse momentum (pT) at the input stage of the model.
  Limitation: Breaks permutation invariance for all downstream tasks, which may adversely affect predictive performance.

Ordering Strategies

The following ordering strategies have been experimented with:

- **Ordering the input to the backbone:** Order particles by decreasing transverse momentum (pT) at the input stage of the model.
  Limitation: Breaks permutation invariance for all downstream tasks, which may adversely affect predictive performance.
- **Ordering only at the input to the pre-training prediction:** This method helps break symmetry for masked predictions without affecting the backbone's permutation invariance.
  Apply pT ordering just before the prediction head, using learned positional embeddings.
  Limitation: Adds complexity to the model architecture and may increase computational cost, though less severe than method 2.

Ordering	Inputs	Loss	Accuracy
no ordering	continuous	VQ-VAE classification	54.1%
order head	**continuous**	**VQ-VAE classification**	**56.8%**
order backbone	continuous	VQ-VAE classification	53.4%
order head	quantized	VQ-VAE classification	51.1%
order head	quantized	K-means classification	49.3%
order head	continuous	K-means classification	56.2%
order head	continuous	regression	48.9%
order backbone	continuous	regression	46.3%

Table: Comparison of different ordering strategies, input types, and loss functions

Masked Particle Modeling Objective

Let's say a jet is given by:

$$X = \{x_i\}_{i=1}^{N} \qquad (x_i \text{ represents a particle})$$

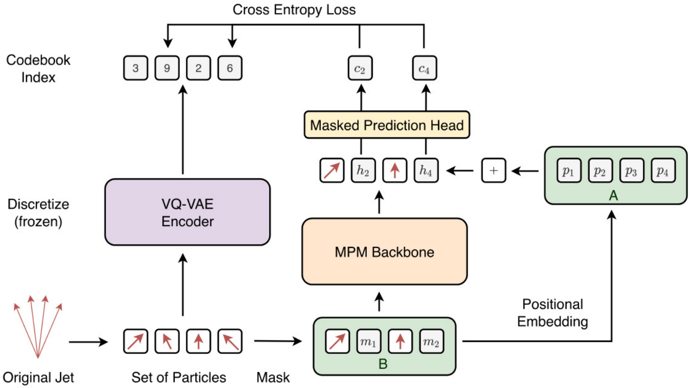
MPM partitions the set of particles in each jet in the dataset into masked and unmasked set:

$$\mathcal{M}_x = \{x_i\}_{i \in \mathcal{M}}, \quad \mathcal{U}_x = \{x_i\}_{i \in \mathcal{U}}$$

The goal of pretraining is to find out a parametric function $f_\theta : X \to \mathbb{R}^{Nxd}$ (assume d-dimentional paramter set for each particle) such that the expectation of loss $\mathcal{L}$ can be minimized.

$$\mathbb{E}_x \left[\frac{1}{|\mathcal{M}_m|} \sum_{i \in \mathcal{M}_m} \mathcal{L}(x_i, f_{\theta,i}(\mathcal{M}_m, \mathcal{U}_x)) \right]$$

Overview diagram of MPM



Datasets Used

1. **JetClass Dataset:** Used for pretraining the backbone

Datasets Used

1. **JetClass Dataset:** Used for pretraining the backbone
   - It contains 100 million samples and has 10 different classes, with equal samples in each class. Examples include jets from Higgs boson decay, quarks, gluons, and other particles.

Datasets Used

1. **JetClass Dataset:** Used for pretraining the backbone
   - It contains 100 million samples and has 10 different classes, with equal samples in each class. Examples include jets from Higgs boson decay, quarks, gluons, and other particles.
   - Each class represents jets from different particle decays

Datasets Used

1. **JetClass Dataset:** Used for pretraining the backbone
    - It contains 100 million samples and has 10 different classes, with equal samples in each class. Examples include jets from Higgs boson decay, quarks, gluons, and other particles.
    - Each class represents jets from different particle decays
    - Uses particle momentum and direction as features

Datasets Used

1. **JetClass Dataset:** Used for pretraining the backbone
   - It contains 100 million samples and has 10 different classes, with equal samples in each class. Examples include jets from Higgs boson decay, quarks, gluons, and other particles.
   - Each class represents jets from different particle decays
   - Uses particle momentum and direction as features
2. **RODEM Dataset:** We are used as a test-set and for fine-tuning.

Datasets Used

1. **JetClass Dataset:** Used for pretraining the backbone
   - It contains 100 million samples and has 10 different classes, with equal samples in each class. Examples include jets from Higgs boson decay, quarks, gluons, and other particles.
   - Each class represents jets from different particle decays
   - Uses particle momentum and direction as features
2. **RODEM Dataset:** We are used as a test-set and for fine-tuning.
   - Has 10 million samples for each class. Contains additional samples of top quark jets and combined quark/gluon jets.

Datasets Used

1. **JetClass Dataset:** Used for pretraining the backbone
    - It contains 100 million samples and has 10 different classes, with equal samples in each class. Examples include jets from Higgs boson decay, quarks, gluons, and other particles.
    - Each class represents jets from different particle decays
    - Uses particle momentum and direction as features
2. **RODEM Dataset:** We are used as a test-set and for fine-tuning.
    - Has 10 million samples for each class. Contains additional samples of top quark jets and combined quark/gluon jets.
    - Uses slightly different parameters for jet clustering Covers a wider range of jet energies and directions

Datasets Used

1. **JetClass Dataset:** Used for pretraining the backbone
   - It contains 100 million samples and has 10 different classes, with equal samples in each class. Examples include jets from Higgs boson decay, quarks, gluons, and other particles.
   - Each class represents jets from different particle decays
   - Uses particle momentum and direction as features
2. **RODEM Dataset:** We are used as a test-set and for fine-tuning.
   - Has 10 million samples for each class. Contains additional samples of top quark jets and combined quark/gluon jets.
   - Uses slightly different parameters for jet clustering Covers a wider range of jet energies and directions
   - Only uses particle's four momentum as feature, assuming particles are massless

Results

Terminology:

- Fixed Backbone - The model just after the pretraining, without fine-tuning

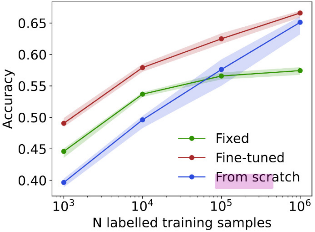


Figure: Accuracy of different training strategies as a function of the number of labelled training samples. Pretained in JetClass

Results

Terminology:

- Fixed Backbone - The model just after the pretraining, without fine-tuning
- Fine-tuned Model - The pretrained backbone, is fine tuned for that specific downstream task.

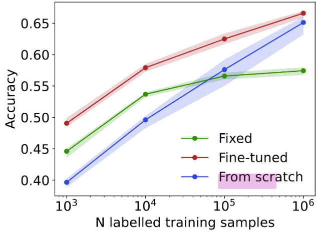


Figure: Accuracy of different training strategies as a function of the number of labelled training samples. Pretained in JetClass

Results

Terminology:

- Fixed Backbone - The model just after the pretraining, without fine-tuning
- Fine-tuned Model - The pretrained backbone, is fine tuned for that specific downstream task.
- From-Scratch - Best Supervised model for that specific task

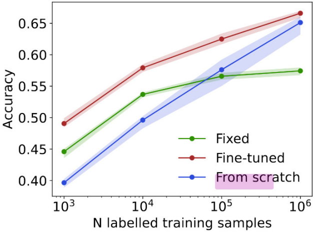


Figure: Accuracy of different training strategies as a function of the number of labelled training samples. Pretained in JetClass

Results

When tested and fine-tuned on a different dataset (RODEM), but pretained in JetClass:

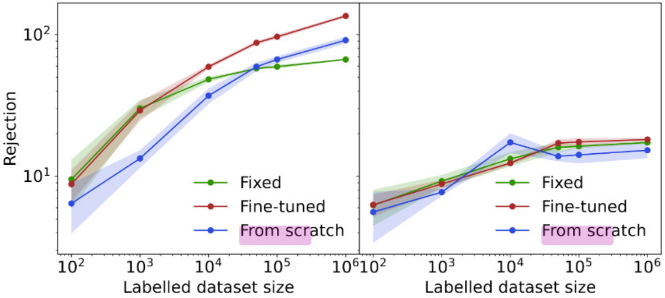


Figure: The QCD rejection evaluated on (left) the RODEM test set and (right) the JetClass test set, as a function of the size of the RODEM data set used for fine-tuning

Conclusion

Masked particle modeling adapts well to unordered inputs in high energy physics.

- Pre-trained models perform well on downstream tasks with minimal fine-tuning. Models generalize to unseen classes and show strong performance in weak supervision.
- Pre-training on experimental data shows promise for addressing domain adaptation. Larger datasets for pretraining and better models may further improve performance in this field.

Thank You