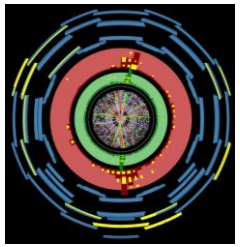
A Very Simple Tutorial on ParticleTransformer

Shudong WANG

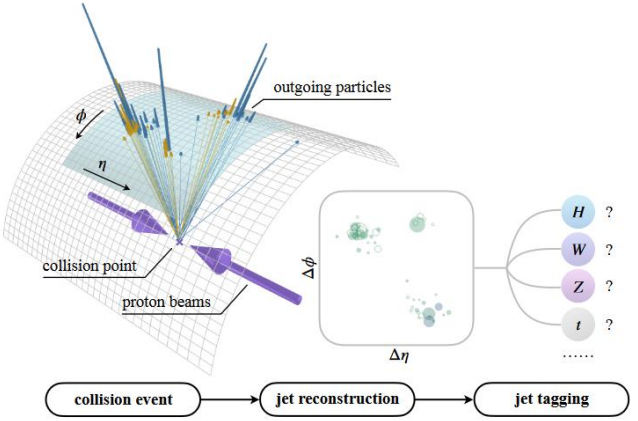
EPD, IHEP, CAS

Outline

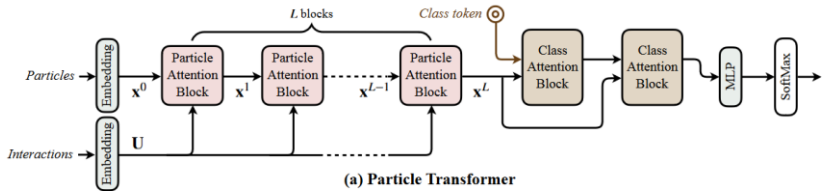
- **A little background knowledge - Jets**



- **A Brief Introduction on ParticleTransformer**
- Attention & Self-Attention
- The architecture of ParticleTransformer

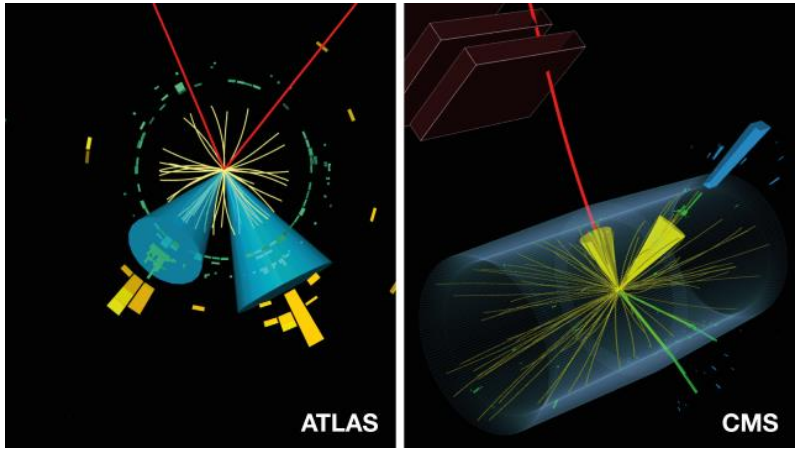


- **How to run ParticleTransformer using Weaver**
- Try it yourself!

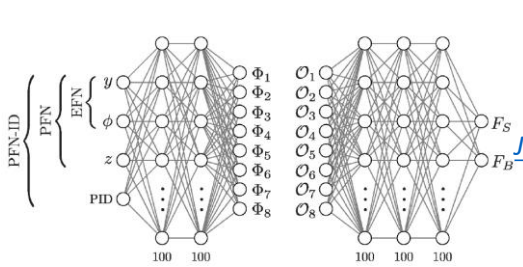


A little background knowledge-Jets

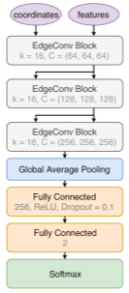
- Jets are ubiquitous at colliders, especially for hadron colliders.
- Jets are collimated sprays of particles initiated by quarks or gluons.



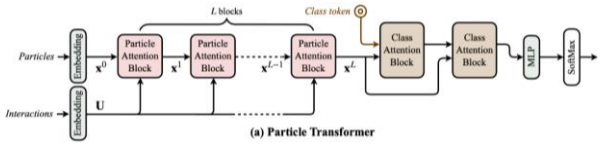
- Jet tagging: identifying the hard scattering particle that initiates the jet.
- The rise of machine learning (ML) has brought lots of new progresses to jet tagging.



JHEP01(2019)121



Phys.Rev.D 101 (2020) 5, 056019



2202.03772

A Brief Introduction on ParticleTransfomer

Attention & Self-Attention

- **Attention**
- ***Attention*** is a very broad concept.
- Background: as the model grows, computing resources become increasingly strained. How to better allocate the limited computing resources by importance?
  -> ***Attention***

- **Attention mechanism – general case**
- **STEP1:** Calculate the attention distribution on the input information to obtain the importance distribution of different input information, i.e. different weights
- **STEP2:** Calculate the weighted average of the current input information according to the different importance, in order to achieve for more efficient use of computing resources

Attention & Self-Attention

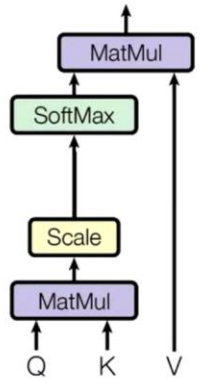
- **Calculate the attention distribution**
- For N input vectors: $[x_1, \ldots, x_N]$, to pick out information that is important to one's goal, one needs to introduce a representation of the goal -> ***query vector (q)***.
- Then the problem turn into investigating correlations between different inputs and the query vector.
- A simple approach: using ***attention scoring function:***
  - additive attention: $s(x, q) = v^T tanh(Wx + Uq)$
  - dot-product attention: $s(x, q) = x^T q$
  - scaled dot-product attention: $s(x, q) = \frac{x^T q}{\sqrt{D}}$
  - bilinear attention: $s(x, q) = x^T W q$
- Weights: $softmax(s(x, q))$
- **Calculate the weighted average**
- Soft attention: $att(X, q) = \sum_{n=1}^{N} a_n x_n$
- Hard attention: focus on one input vector only -> non-differentiable (cannot use BP)
  - pick vector with the largest weight
  - random sampling on attention distribution

Attention & Self-Attention

- **Mutations of attention mechanism**

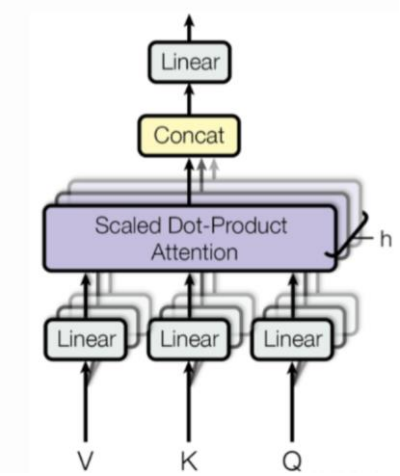
- ***Key-Value Pair Attention***
- inputs are k-v pairs $(\mathrm{K}, \mathrm{V}) = [(k_1, v_1), \ldots, (k_N, V_N)]$
- attention:

$$att((K, V), q) = \sum_{n=1}^{N} softmax(s(q, k_n))v_n$$



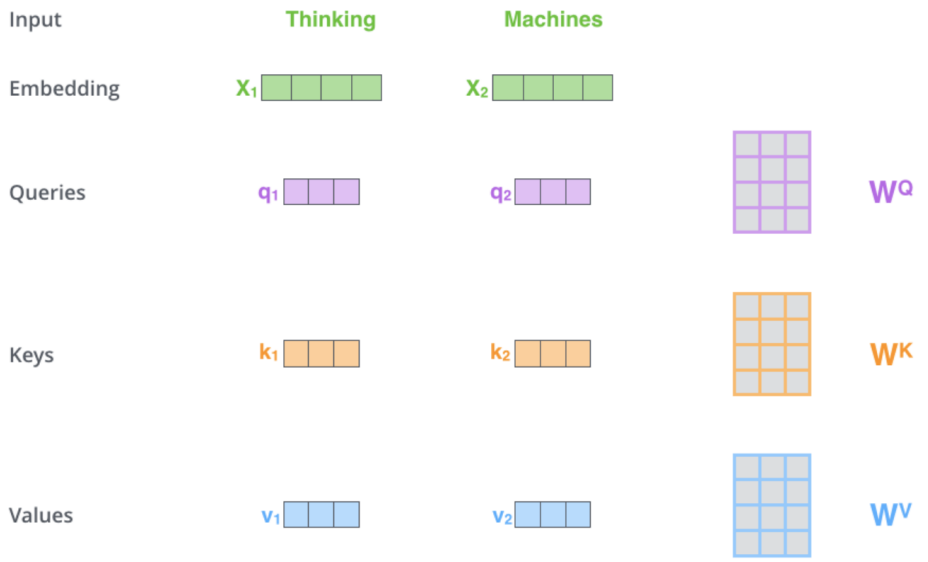
- ***Multi-Head Attention***
- multiple queries, $\mathrm{Q} = [q_1, \ldots, q_M]$
- search for information from inputs in a parallel way
- attention:

$$att((K, V), Q) = att((K, V), q_1) \oplus \ldots \oplus att((K, V), q_M)$$



Attention & Self-Attention

- **Self-Attention**
- ***Query-Key-Value (QKV) Attention***
- Q, K, V are identical (Q=K=V=X) / from the same origin:
- input: $\boldsymbol{X} = [x_1, \dots, x_N]$
- attention: $att(\boldsymbol{X}) = softmax(s(\boldsymbol{X}))\boldsymbol{X}$
- or
- $\boldsymbol{Q} = \boldsymbol{W}_q\boldsymbol{X}, \boldsymbol{K} = \boldsymbol{W}_k\boldsymbol{X}, \boldsymbol{V} = \boldsymbol{W}_v\boldsymbol{X}$
- attention: $att\big((\boldsymbol{K}, \boldsymbol{V}), \boldsymbol{Q}\big) = softmax(s(\boldsymbol{Q}, \boldsymbol{K}))\boldsymbol{V}$



The architecture of ParticleTransformer

H. Qu , C. Li, S. Qian [2202.03772]

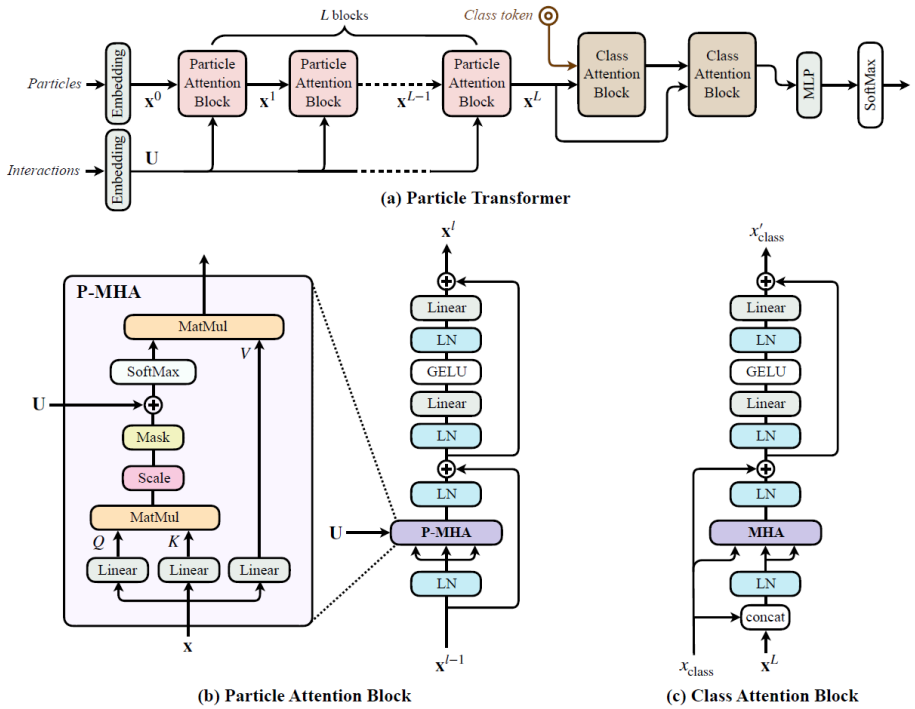


image credit

Choice of the pair-wise features: from LundNet

$$\text{P-MHA}(Q,K,V) = \text{SoftMax}(QK^T/\sqrt{d_k} + \mathbf{U})V$$

d_k: dimension of K

$$\Delta = \sqrt{(y_a - y_b)^2 + (\phi_a - \phi_b)^2}$$
$$k_T = \min(p_{T,a}, p_{T,b}) \cdot \Delta$$
$$z = \min(p_{T,a}, p_{T,b})/(p_{T,a} + p_{T,b})$$
$$m^2 = (E_a + E_b)^2 - ||\mathbf{p}_a + \mathbf{p}_b||^2$$

S. Qian@ML4Jets2022

- ParticleTransformer
  - Transformer designed for particle physics
  - TWO sets of inputs
    - *Particle*: Features of every single particle
    - *Interaction*: Pair-wise features
- Particle Attention Block
  - Multi-Head Attention (MHA) Module
  - Pair-wise feature are introduced as the attention mask (P-MHA)
- Class Attention Block
  - Multi-Head Attention (MHA) Module
  - Class token is used for the MHA calculation

$$\text{MHA}_C(Q_C, K_C, V_C) = \text{SoftMax}(Q_C K_C^T/\sqrt{d_{kC}})V_C$$

$Q_C = W_{qC}x_{\text{class}} + b_{qC}$ $K_C = W_{kC}\mathbf{z} + b_{kC}$ $V_C = W_{vC}\mathbf{z} + b_{vC}$ d_{kC}: dimension of K_C

$$\mathbf{z} = [x_{\text{class}}, \mathbf{x}^L]$$

Concatenate class information and particle embedding

Official implementation of "Particle Transformer for Jet Tagging". (github.com)

Performance of ParticleNet

- Jet tagging performance on the JETCLASS dataset.

	All classes		$H \to b\bar{b}$	$H \to c\bar{c}$	$H \to gg$	$H \to 4q$	$H \to \ell\nu qq'$	$t \to bqq'$	$t \to b\ell\nu$	$W \to qq'$	$Z \to q\bar{q}$
	Accuracy	AUC	$\text{Rej}_{50\%}$	$\text{Rej}_{50\%}$	$\text{Rej}_{50\%}$	$\text{Rej}_{50\%}$	$\text{Rej}_{99\%}$	$\text{Rej}_{50\%}$	$\text{Rej}_{99.5\%}$	$\text{Rej}_{50\%}$	$\text{Rej}_{50\%}$
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**
ParT (plain)	0.849	0.9859	9569	2911	112	1185	3868	17699	12987	384	311

- 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

similar computation complexity with ParticleNet, but more performant than ParticleNet

How to run ParticleTransformer using Weaver

- First thing to do:
  Login to IHEP cluster and do:

```
cp -r /scratchfs/bes/wangshudong/particle_transformer/ /PATH/TO/YOUR/SPACE/particle_transformer/
```

Try it yourself!

- **Weaver**
  - *Weaver* aims at providing a streamlined yet flexible machine learning R&D framework for high energy physics (HEP) applications. (Github: hqucms/weaver-core)
- **Set up your environment (you can use mine)**
  - Install Miniconda (if you don't already have it)
  - Set up a conda environment and install the required packages
  - On IHEP cluster, simply type commands below to use my conda env (you don't even need to do this):

```
#this conda env only support training using CPU, since most of you
#don't have access to GPU cluster
source "/cefs/higgs/wangshudong/miniconda3/etc/profile.d/conda.sh"
conda activate weaver
```

- **Prepare your configuration files**

  To train a neural network using *Weaver*, you need to prepare:
  - A **YAML data configuration file** describing how to process the input data.
  - A **python model configuration file** providing the neural network module and the loss function.
  - ***Let's move to codes now***

Try it yourself!

- **Start running! (general case)**
  - The `weaver` command is the top-level entry to run for training a neural net, getting prediction from trained models, and exporting trained models to ONNX for production. The corresponding script file is weaver/train.py. To check all the command-line options for `weaver`, run `weaver -h`
  - Examples for training, inference and model exportation are shown below:

- **Training**

```
weaver --data-train '/path/to/train_files/*/*/*/*/output_*.root' \
--data-test '/path/to/train_files/*/*/*/*/output_*.root' \
--data-config data/ak15_points_pf_sv.yaml \
--network-config networks/particle_net_pf_sv.py \
--model-prefix /path/to/models/prefix \
--gpus 0,1,2,3 --batch-size 512 --start-lr 5e-3 --num-epochs 20 --optimizer ranger \
--log logs/train.log
```

How to run ParticleNet

- **Prediction/Inference**

```
weaver --predict --data-test '/path/to/test_files/*/*/*/*/output_*.root' \
 --data-config data/ak15_points_pf_sv.yaml \
 --network-config networks/particle_net_pf_sv.py \
 --model-prefix /path/to/models/prefix_best_epoch_state.pt \
 --gpus 0,1,2,3 --batch-size 512 \
 --predict-output /path/to/output.root
```

- **Model exportation**

```
weaver -c data/ak15_points_pf_sv.yaml -n networks/particle_net_pf_sv.py -m
 /path/to/models/prefix_best_epoch_state.pt --export-onnx model.onnx
```

Try it yourself!

- **Start running! (for this tutorial only)**

```
cd /PATH/TO/YOUR/SPACE/particle_transformer/
source train_JetClass_bcg.sh #run on login node
```

Then just wait!