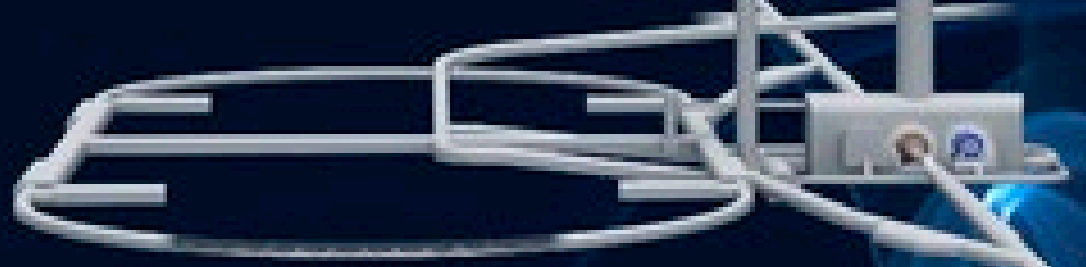


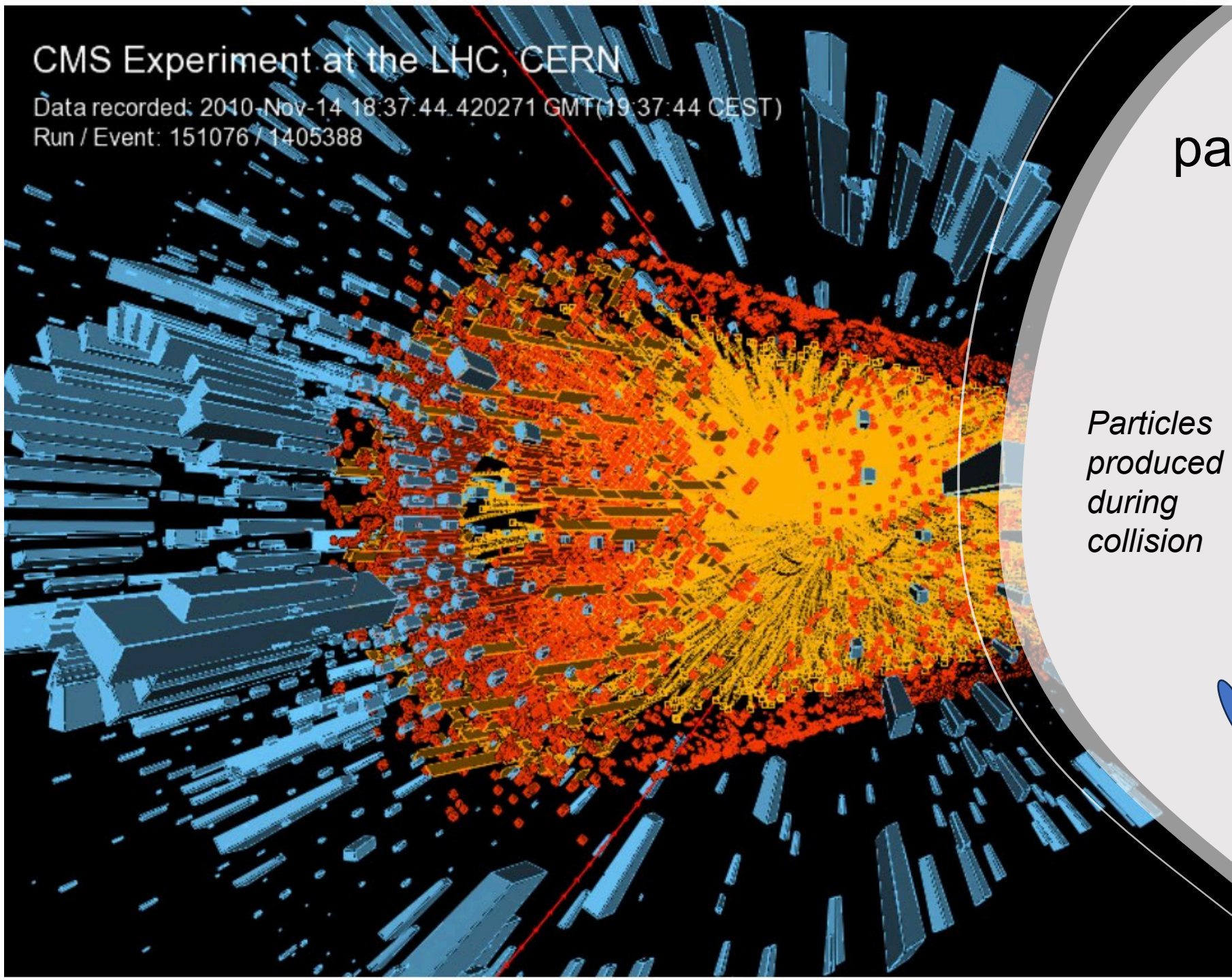
Implenting a Transformer as a Particle Flow Algorithm

06.06.2024

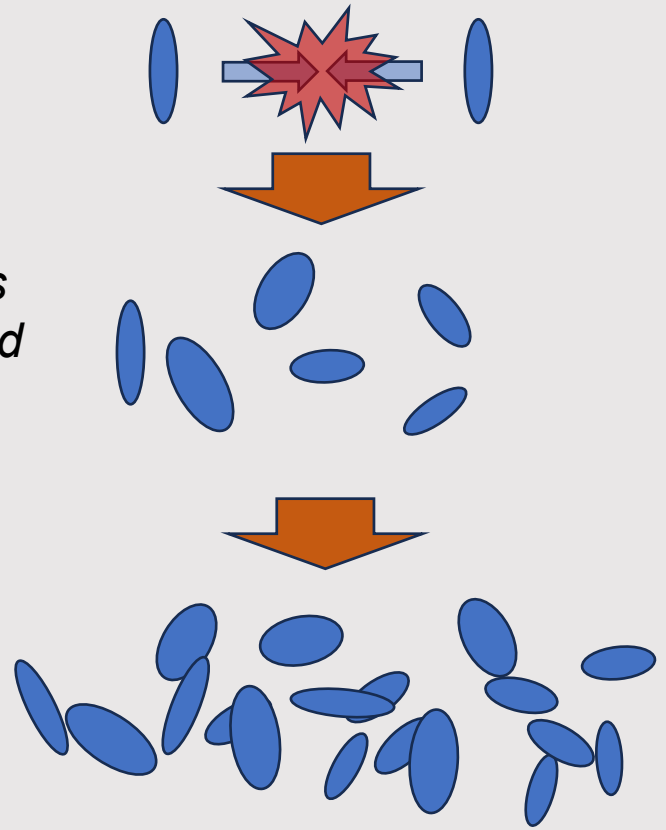
ILANCE Intern Fest

Paul Wahlen,

Supervised by Taikan Suehara & Junping Tian



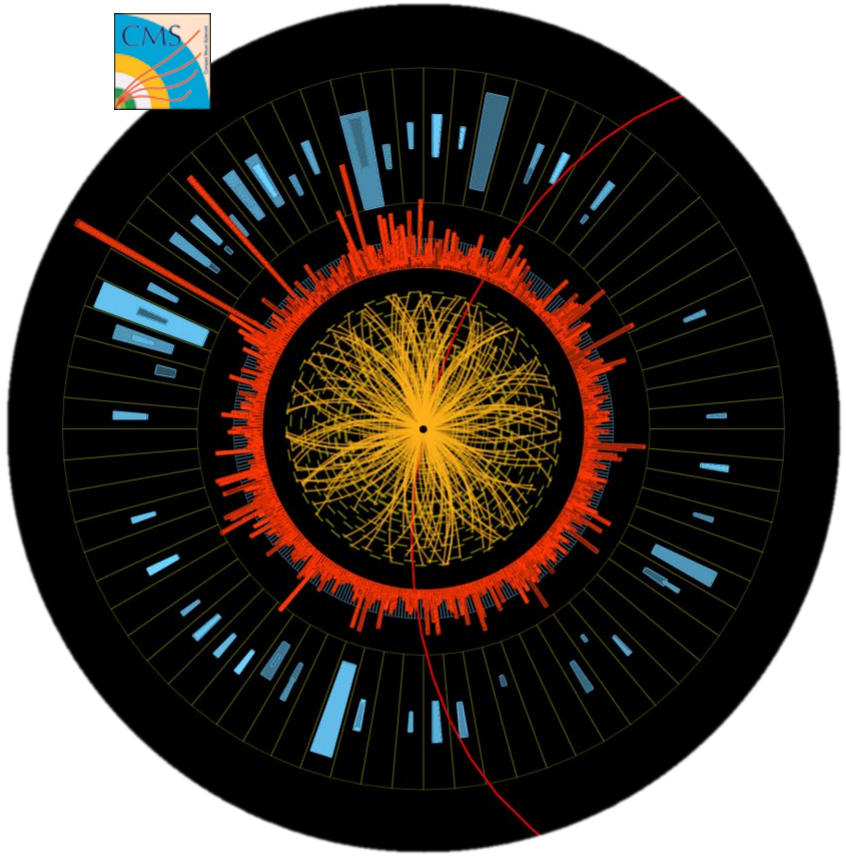
The problem with particle accelerators…



Particles produced during collision

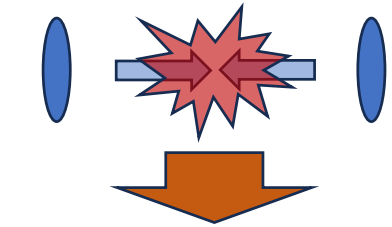
Going through the detectors…

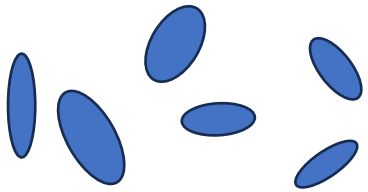
The problem with particle accelerators…



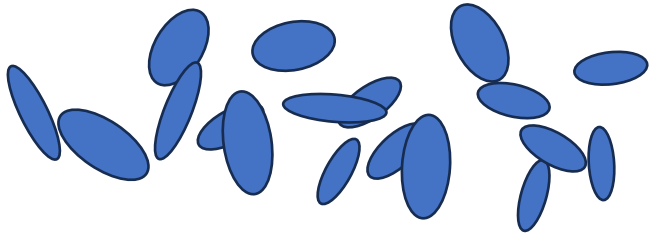
? ?

From the detectors, how do we retrieve the particles initially produced?



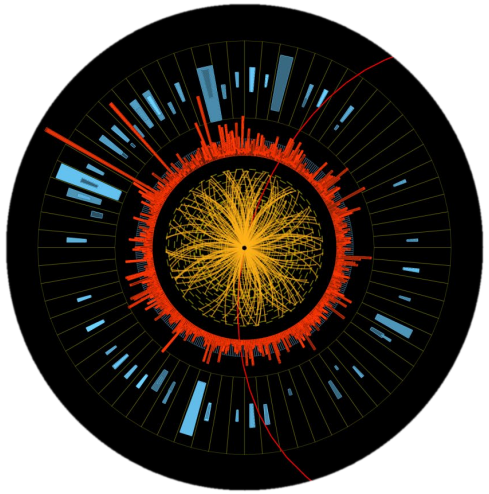


PDG?
Energy?
Momentum?

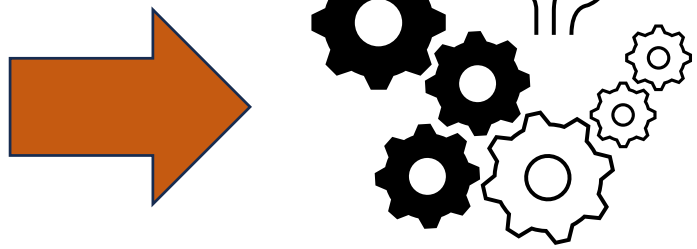


Only energy deposits and tracks are left behind

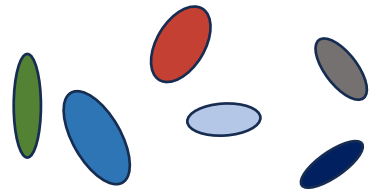
A possible solution (hopefully)



Energy deposit
Position

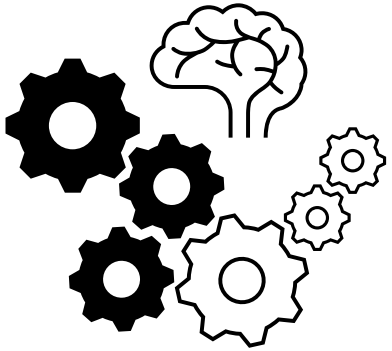


Charge
PDG
Energy
Momentum



Looks somewhat similar to a well known problem: machine translation

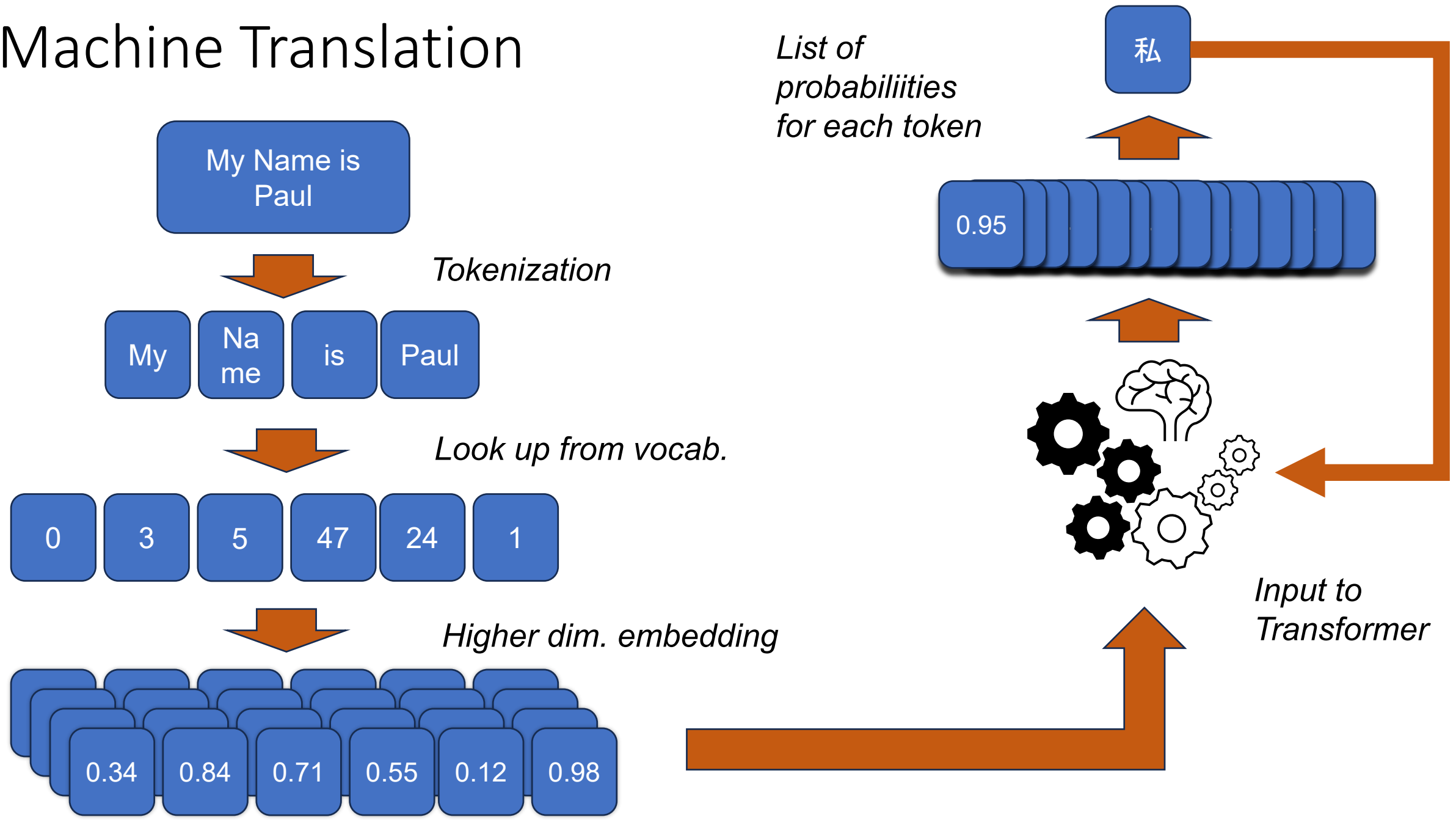
My name is Paul



私はポールです

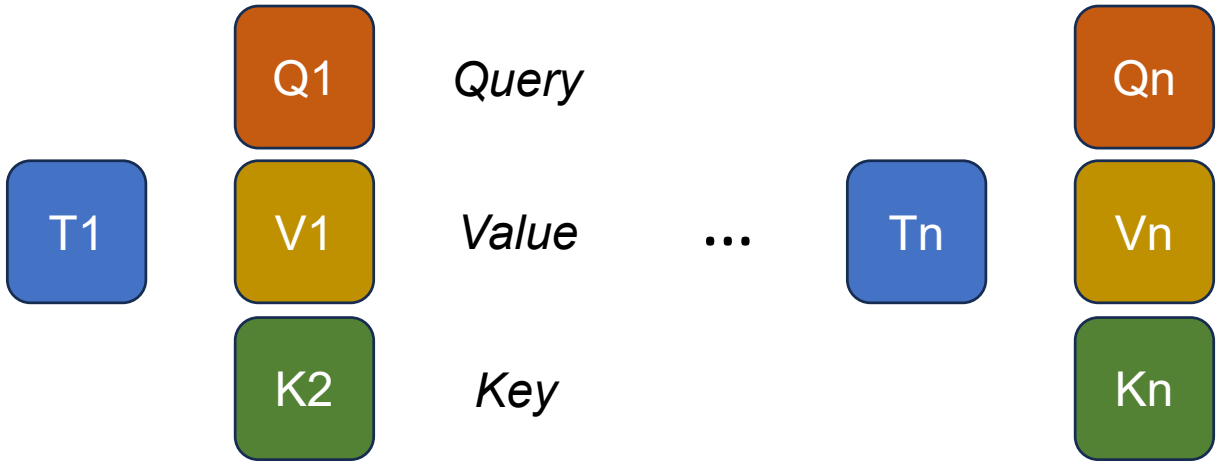
Nowadays, achieved using a Neural Network called Transformer

Machine Translation

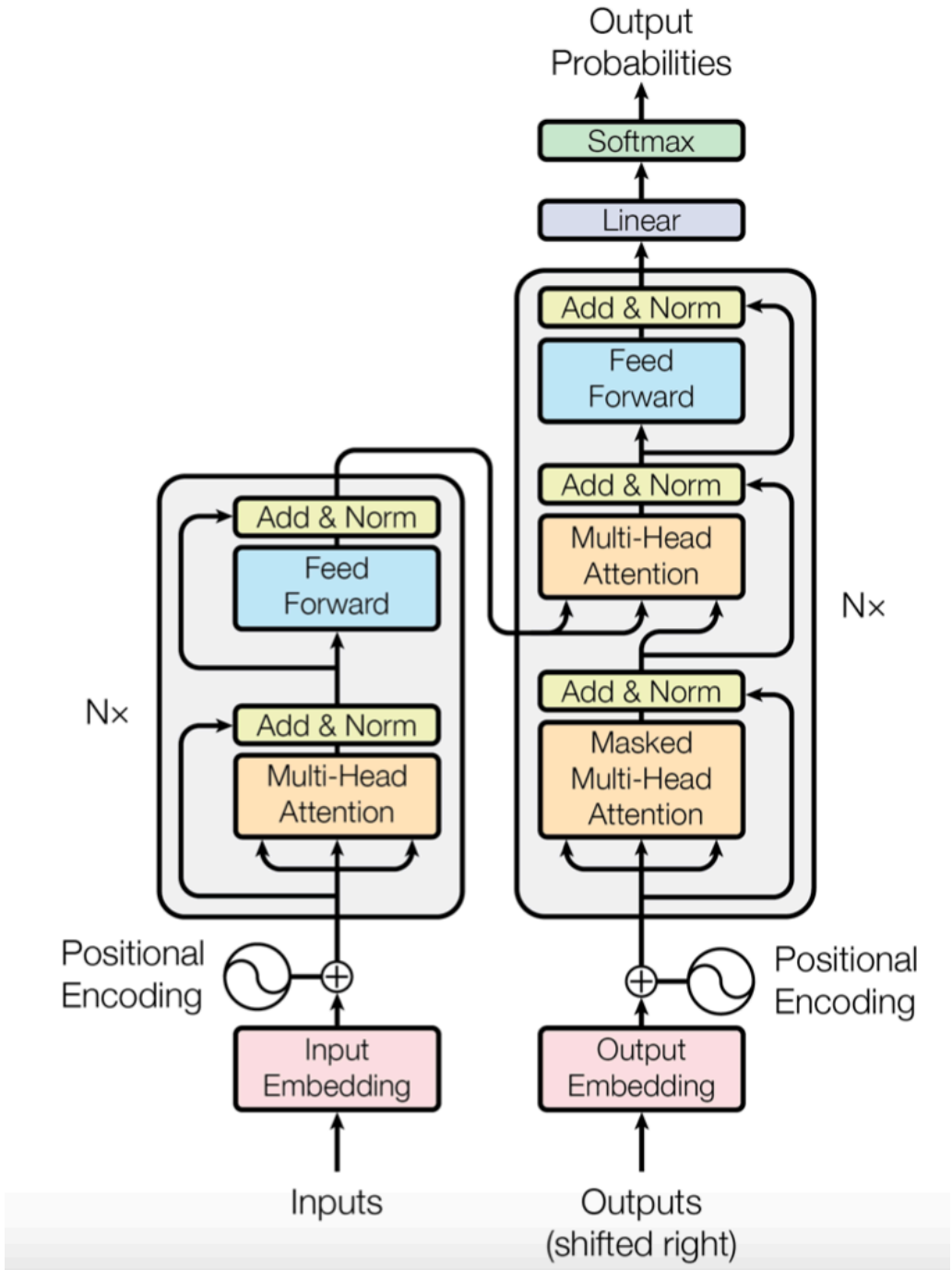


Inside a transformer

Multi-head Attention: Allows tokens to communicate with each other to better understand the context in which they are used.

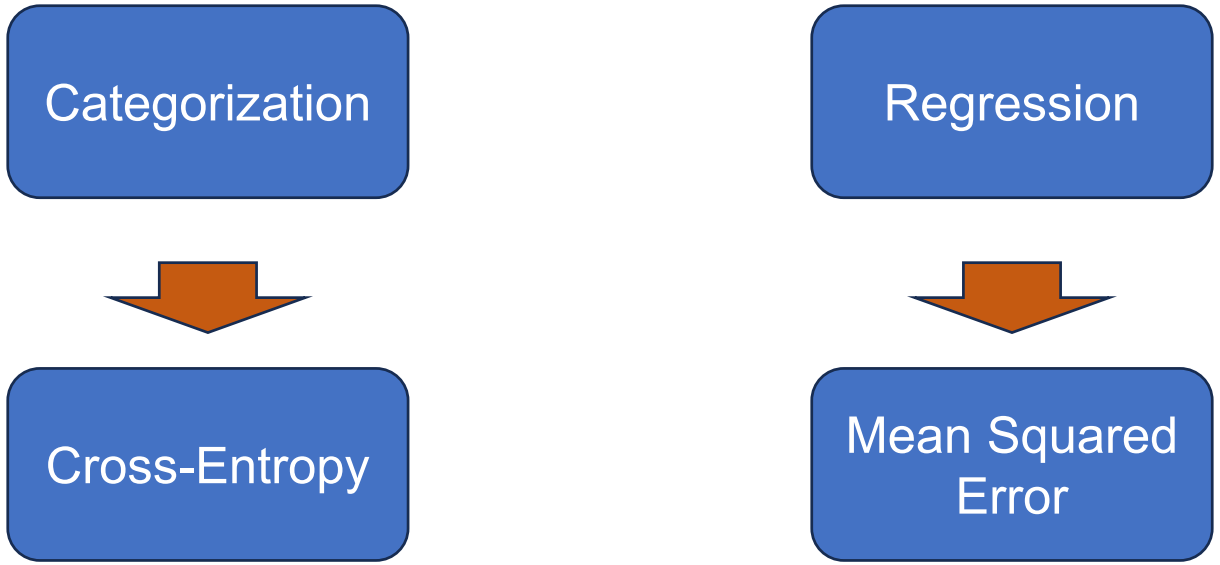


$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$



Shaping the output we want

- Determined during training, when the model adjusts its parameters to minimize *the loss function*
- Good prediction means that the ”distance” between model output and truth is small.



In both cases, the further away from the truth, the larger the value of the loss function

Project concept

Using a Transformer to predict the particles generating the clusters

	Sequence to Sequence	Physics
Input	Sentence	List of hits from 1 event
Output	Machine translation of Seq	List of clusters to which belongs each hit
token	Depends, words/ few char.	1 hit
Special tokens	bos, eos, unkwn, pad	bos, eos, sample, pad

$$\begin{pmatrix} \text{bos} & \text{hit} & \text{hit} & \text{eos} & \text{pad} & \text{pad} \\ \text{bos} & \text{hit} & \text{hit} & \text{hit} & \text{eos} & \text{pad} \\ \text{bos} & \text{hit} & \text{eos} & \text{pad} & \text{pad} & \text{pad} \end{pmatrix}$$

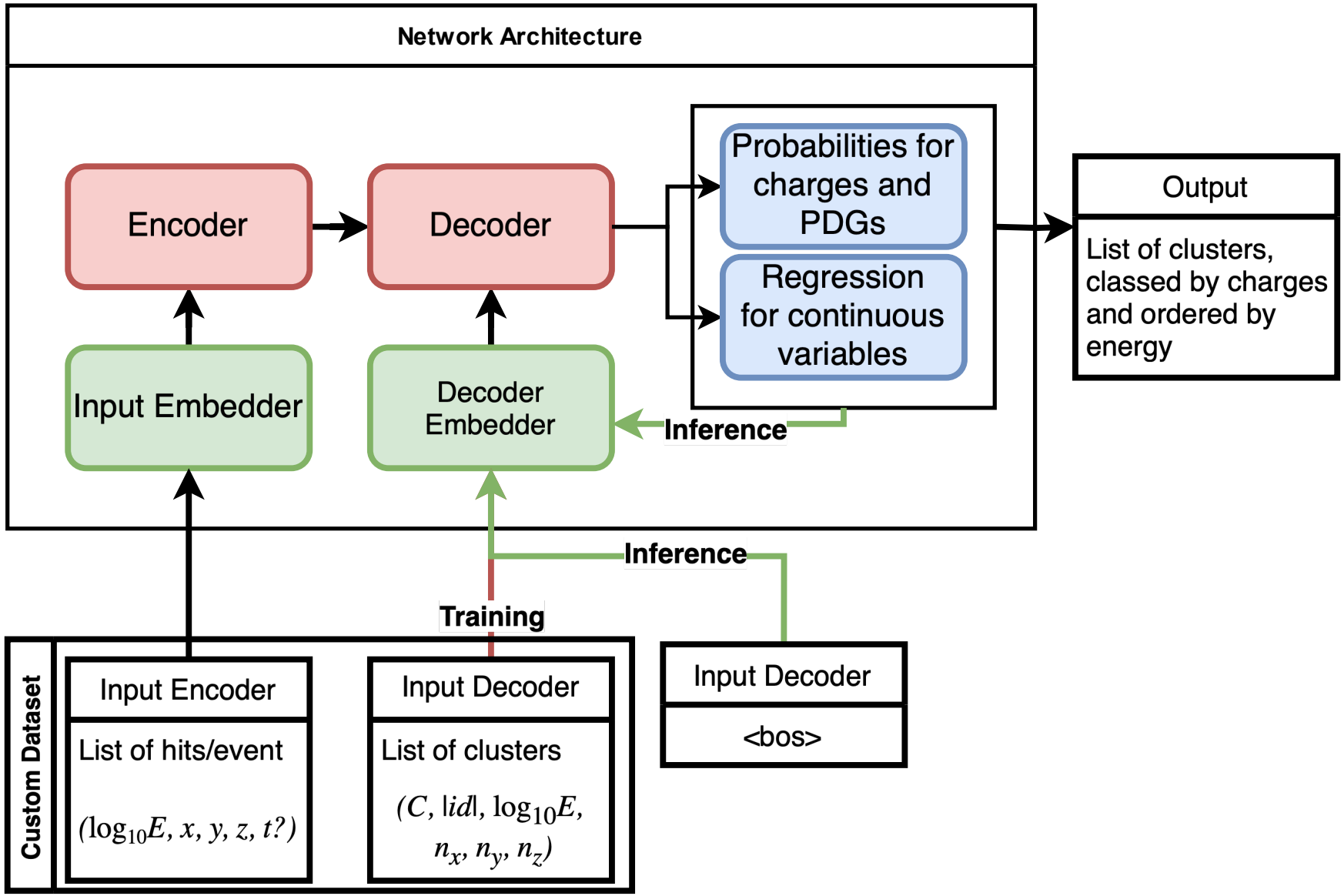
- Special symbols and general formatting of the raw dataset is done by a custom Pytorch's Dataset

General Architecture

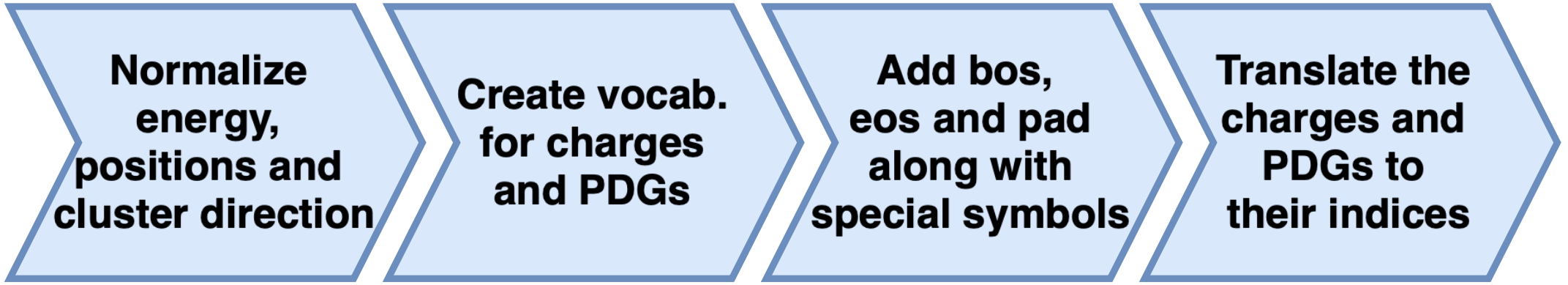
Cluster information are obtained from MC Particle truth information.

3 loss functions, weighted by hyperparameters:

- Most common particle ids form vocabulary: $\gamma, K_s, K_L, K^+, \mu^-, p, n, \pi^\pm, e^-$ CrossEntropyLoss
- Charges form other vocabulary. -1, 0 ,1. Also CrossEntropyLoss
- Continuous variables are obtained by regression. MSE for the loss function.

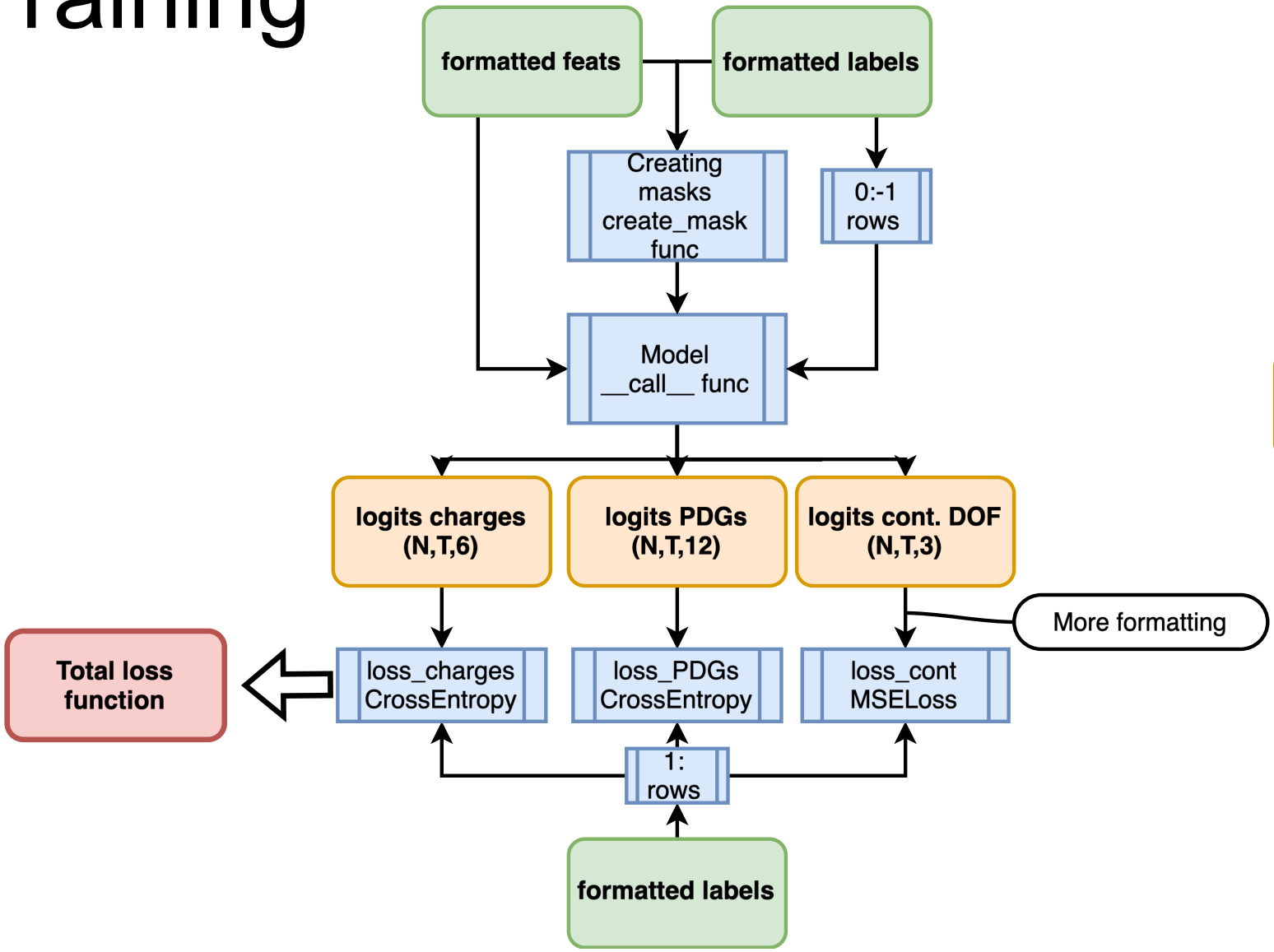


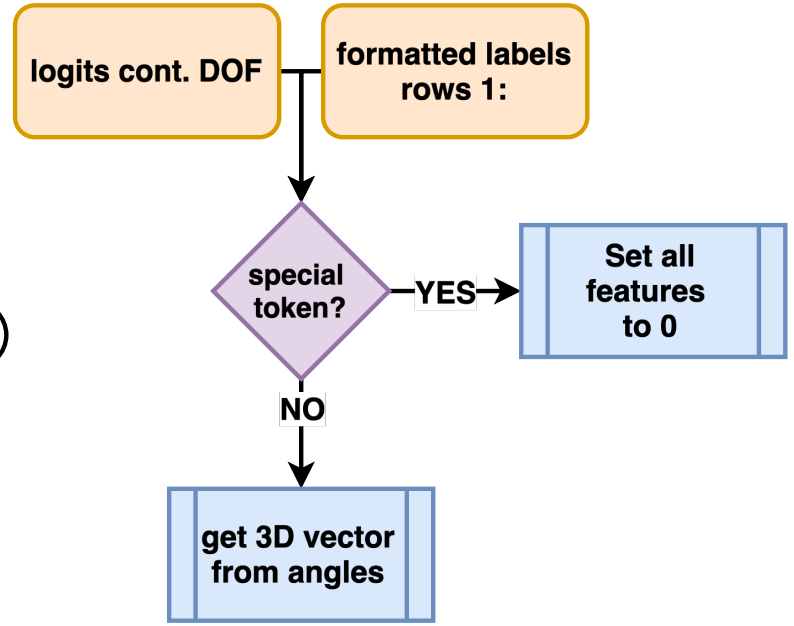
Data Processing



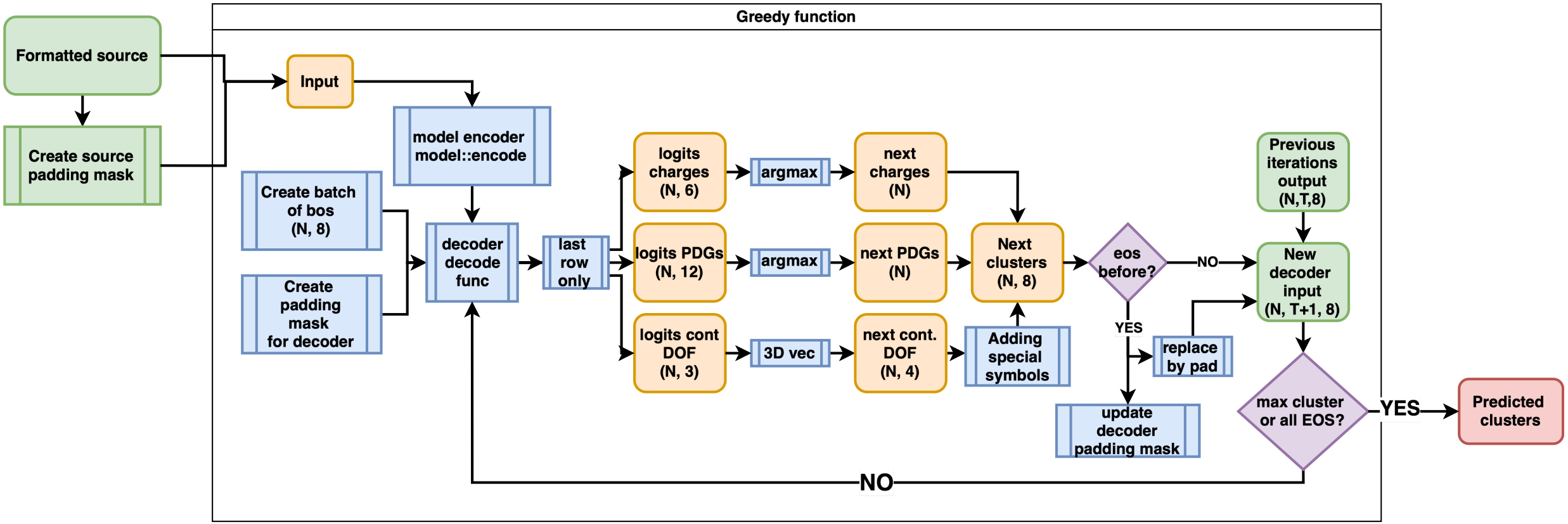
$$\begin{aligned}
\text{pad:}\quad & (-150, -150, 0, 0, 0, 0, 0, 1) \mapsto (0, 0, 0, 0, 0, 0, 0, 1) \\
\text{bos:}\quad & (-100, -100, 0, 0, 0, 0, 0, 1) \mapsto (1, 1, 0, 0, 0, 0, 1, 1) \\
\text{eos:}\quad & (-50, -50, 0, 0, 0, 0, 0, 1) \mapsto (2, 2, 0, 0, 0, 0, 1, 0) \\
\text{sample:}\quad & (-1, 11, E, n_x, n_y, n_z, 0, 0) \mapsto (3, 5, E, n_x, n_y, n_z, 0, 0)
\end{aligned}$$

Training


formatted feats
formatted labels
Creating masks create_mask func
0:-1 rows
Model __call__ func
logits charges (N,T,6)
logits PDGs (N,T,12)
logits cont. DOF (N,T,3)
More formatting
Total loss function
loss_charges CrossEntropy
loss_PDGs CrossEntropy
loss_cont MSELoss
1: rows
formatted labels


logits cont. DOF
formatted labels rows 1:
special token?
YES
Set all features to 0
NO
get 3D vector from angles

Inference


Greedy function
Formatted source
Create source padding mask
Input
model encoder model::encode
Create batch of bos (N, 8)
Create padding mask for decoder
decoder decode func
last row only
logits charges (N, 6)
argmax
next charges (N)
logits PDGs (N, 12)
argmax
next PDGs (N)
logits cont DOF (N, 3)
3D vec
next cont. DOF (N, 4)
Next clusters (N, 8)
Adding special symbols
eos before?
NO
YES
replace by pad
update decoder padding mask
Previous iterations output (N,T,8)
New decoder input (N, T+1, 8)
max cluster or all EOS?
YES
Predicted clusters
NO

Conclusion

- First implementation now finished and in training
- Need to play around with the different parameters to get best performances