Study for jet flavor tagging with ~~recent Neural Network~~ → Transformer

Masahiro Morinaga

The University of Tokyo
(ICEPP ,Beyond AI)

Boost23 @Laurence Berkeley National Lab

What is Flavor Tagging?

Jet Flavor

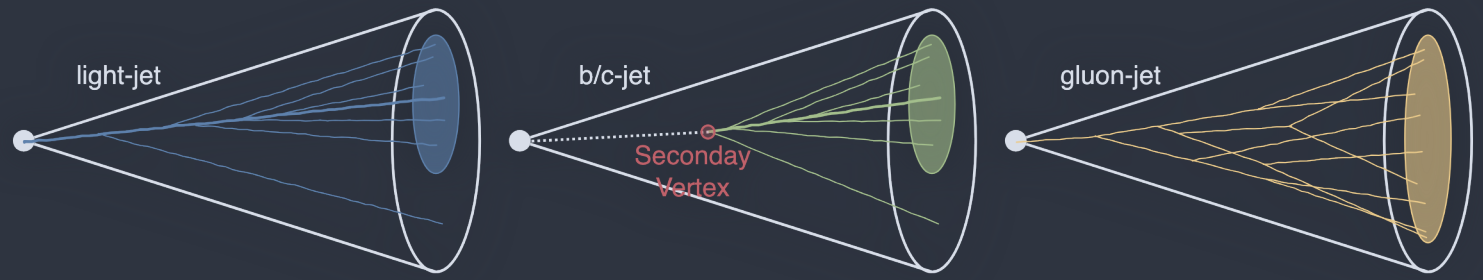
- QCD jets originated from quark or gluon decay products.
- Tagging jet flavor is an important technology to improve analysis sensitivity
- Quark and Gluon classification
- W/Z boson, Higgs, Top tagging :
  - Using a large R-jet can identify the jet if a particle is boosted
  - Only higher p_{T}

Heavy Flavor Tagging

- Rely on secondary vertex from B-meson or D-meson decay
  - b-quark : $b \to B \to D \to K$
  - c-quark : $c \to D \to K$
- Machine Learning : BDT, CNN, RNN, DeepSets, Graph NN

Quark/Gluon Tagging

- The separation between light flavor and gluon
- Rely on #of tracks or similar variables
- ML : Graph or similar specialized model(ParticleNet, Lorentz Group NN)



Strategy

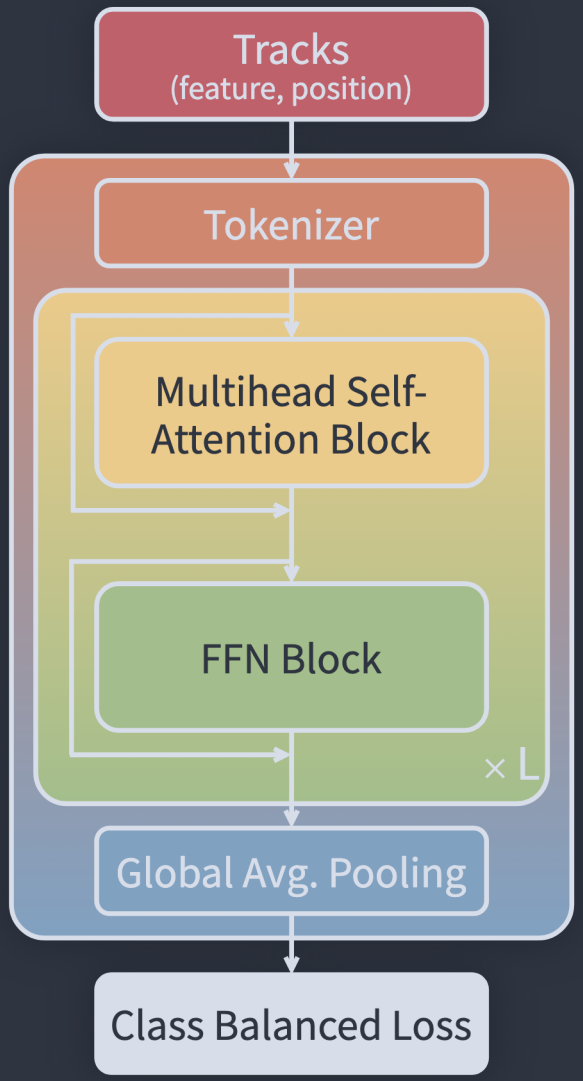
Problem

- Models that are too specialized for HEP are hard to use in implementation and application.
- Standard models (ResNet, Transformer, etc.) are easy to implement and apply.
- Flavor tagging, such as b-tagging, uses only track information, but essentially clusters are also considered to have useful information
- Flavor tagging and $H/Z/W/$top-tagging are often constructed separately.

Aim of this study

- Use models widely used in the ML community $\rightarrow$ Transformer
- Create an all-in-one tagger that includes all flavors.
  - Classify light-flavor, gluon-jet, c-jet, b-jet simultaniously
  - Same model can be applied for the large-R jet ($q\bar{q}$)
- Adding cluster information
  - Expect better performance at higher p_{T} by using calorimeter information

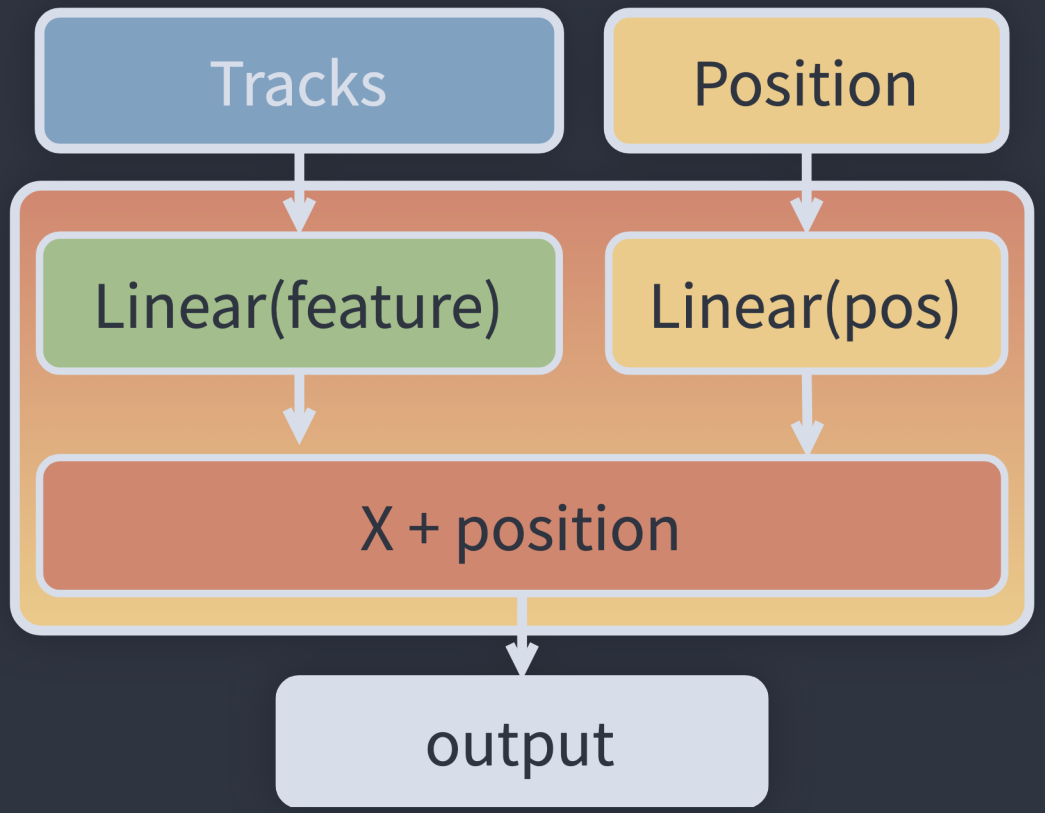
Model Architechture : Transformer



Transformer

- Simple Transformer encoder
- Taking input sequences into a tokenizer
- Repeat two residual blocks L times:
  - Multihead self-attention:
  - Feed Forward Network:
- Pool by Global Average Pooling
  - There is a Linear layer after the pooling layer as a head
- No jets physics acknowledgments
- No Lorentz summation

Model Architechture : Tokenizer



Tokenizer

- Tokenizer make tokens from track inputs
  - Feature : 4-vector or other features for each tracks
  - Position : Relative position difference to parent jet, $\Delta\phi$, $\Delta\eta$
- Sum up X and position, not concat!

Training Setup

Training Sample

- top-data: Open access data with Top and QCD dijet samples
  - 14 TeV, Delphes ATLAS detector card with Pythia8
  - Clustering of particle-flow with anti-kT 0.8 jets in the pT range [550,650] GeV
  - 1.2M for training, 0.4M for test/validation samples

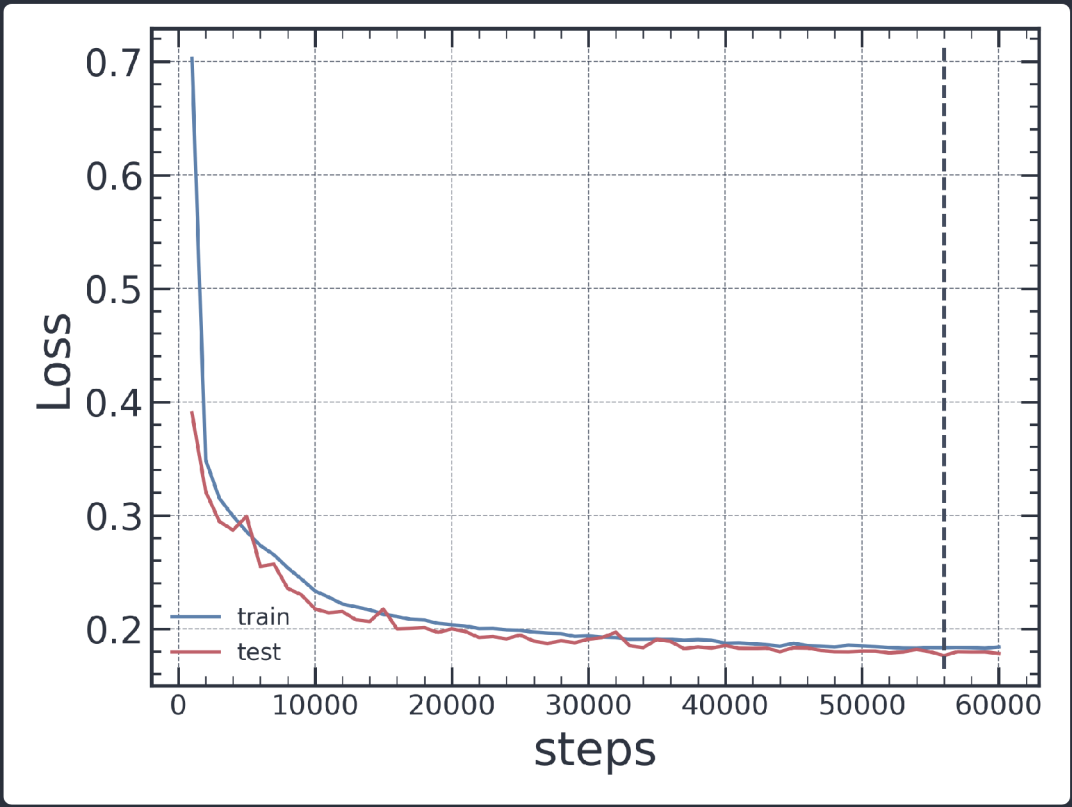
Model Setup

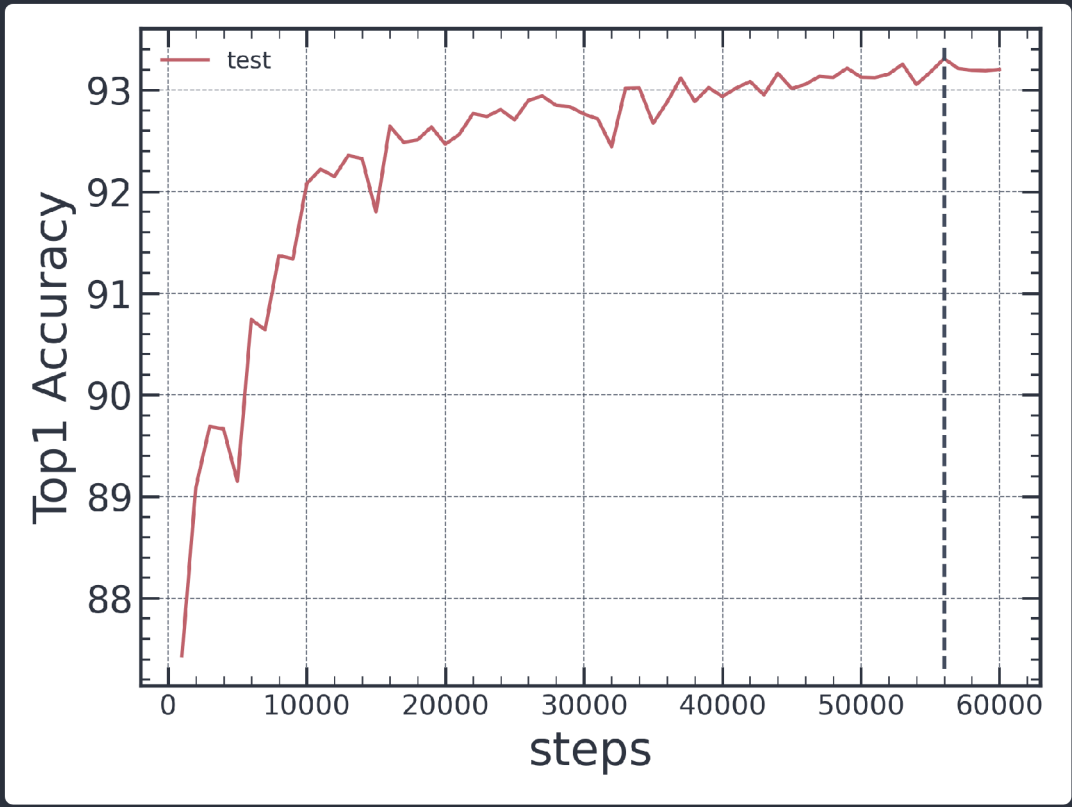
- Total #of Parameters : `101,794`
  - Depth : `4`
  - Latent dimension : `32` , factor `x4` for internal
  - Attention head : `4`
  - Dropout : `0.1`
  - Stochastic path : `0.1`

Training setup

- Trained by JAX/Haiku
- Optimizer : `Adam` with wieght decay `1.0e-4`
- Learning rate : `CosineAnnealingWarmupRestart`
  - `1.0e-4` to `1.0e-7` ,
  - Wramup : `10%` of total update steps
- Using NVidia `A100` GPU x 8
  - data/model/operator parallel by alpa
- Using mixed precision `bfloat16`
- Batch size : default is `4096 = 4k` .
  - Gradient accumulation : #of micro batch `8`
- Target loss: Class Balanced Loss
  - The number of each signal is not equal in size
  - To correct the imbalance in training

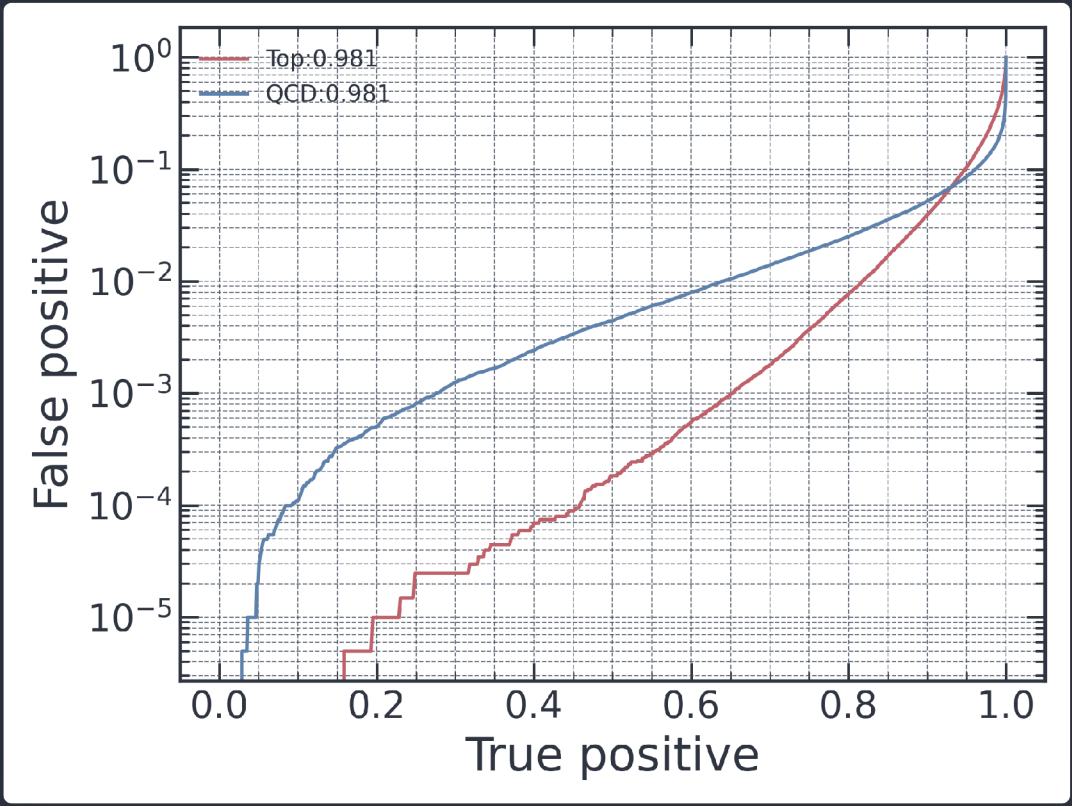
Loss Curve

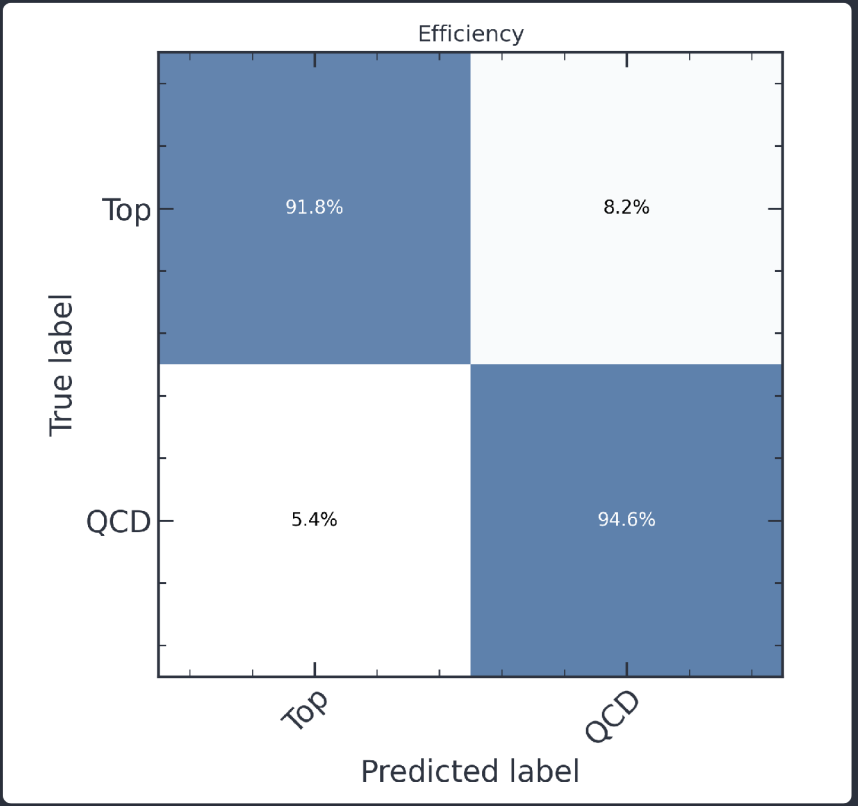




- training loss curve and top-1 accuracy
  - 60000 steps, roughly ~200 epochs
  - Total training time ~2.5h

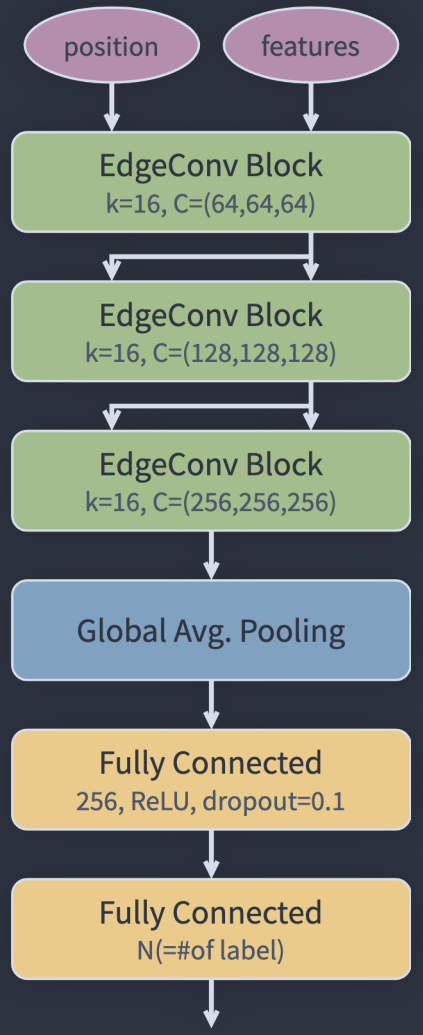
Performance : Transformer

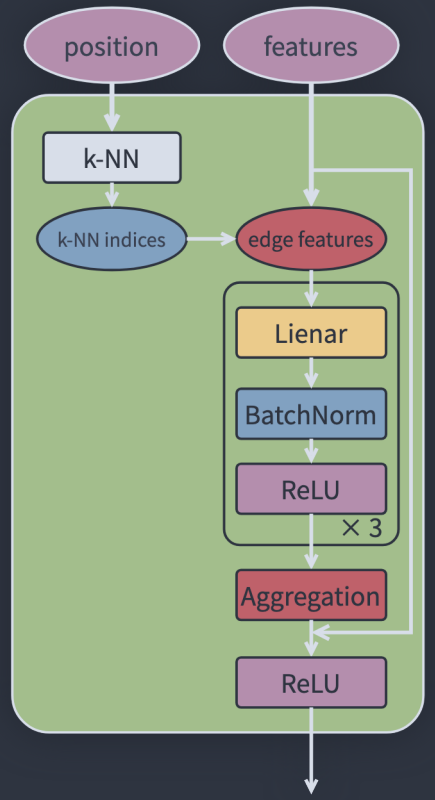




- ROC curve
- Confusion matrix:
  - 91% of the true top can be labeled as the predicted top.

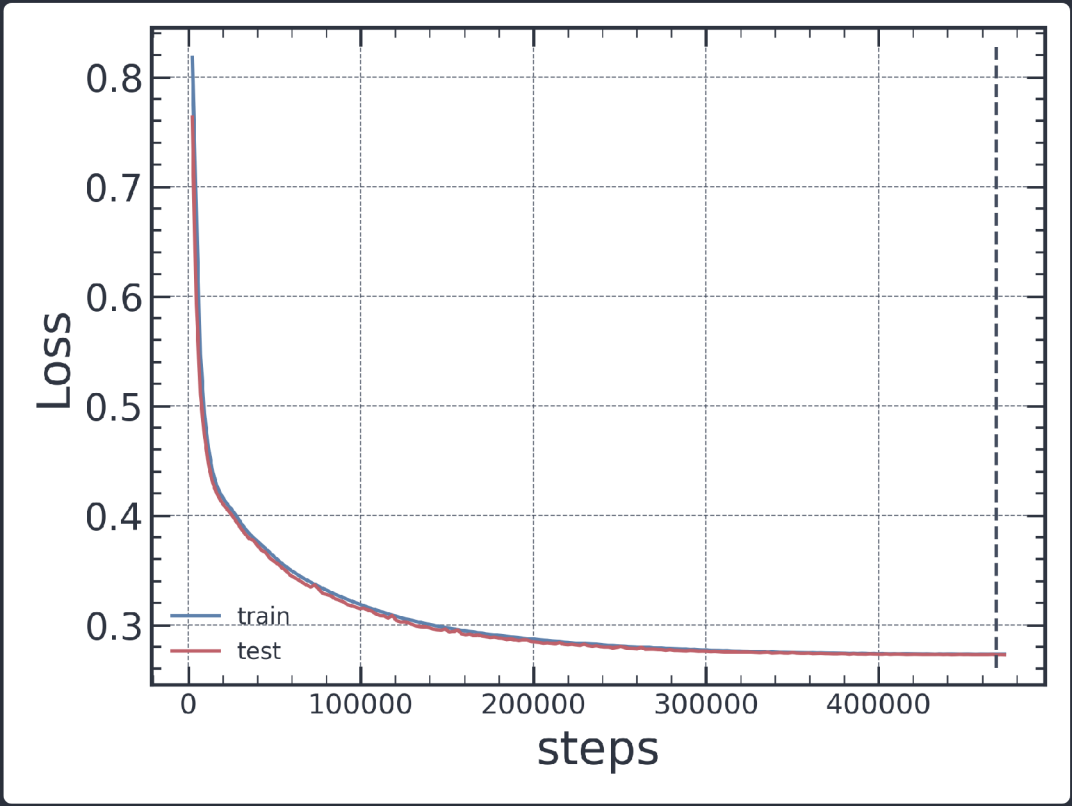
Comparison : ParticleNet

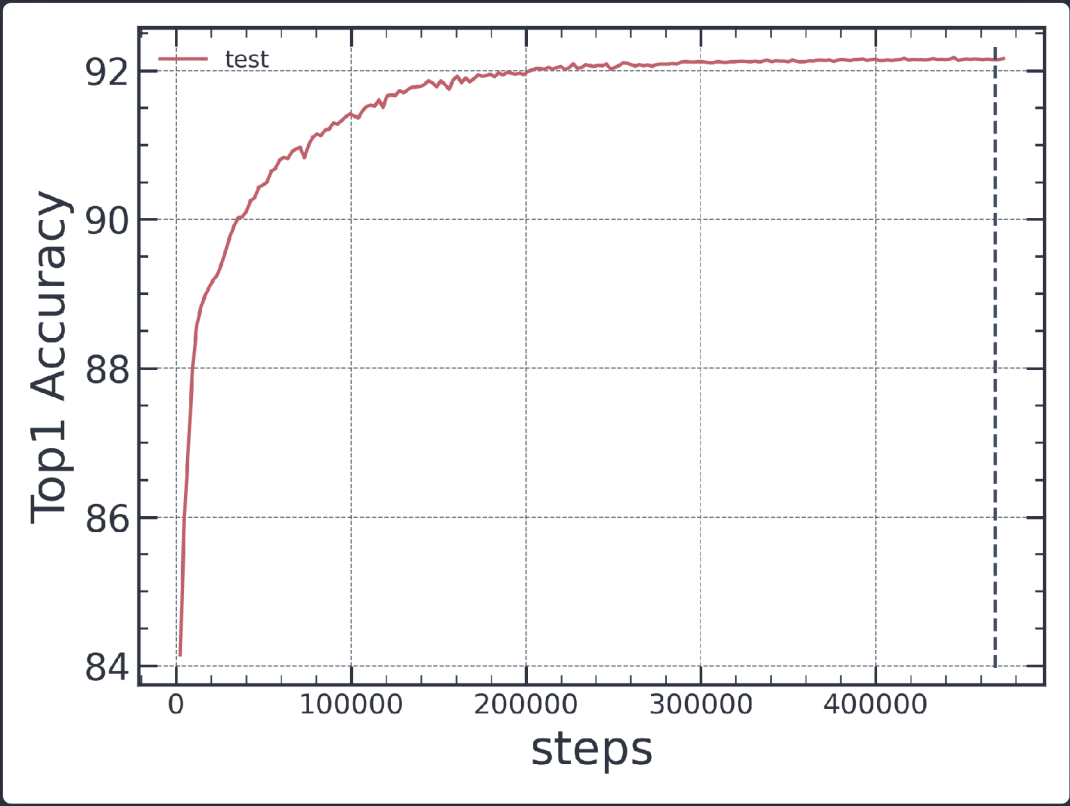




- ParticleNet is a good candidate to compare with this work.
  - Graph Neural Network like style
  - Edge convolution with k-NN
  - Total # of paramters : `365,162`
    - Three times larger than Transformer in this study.

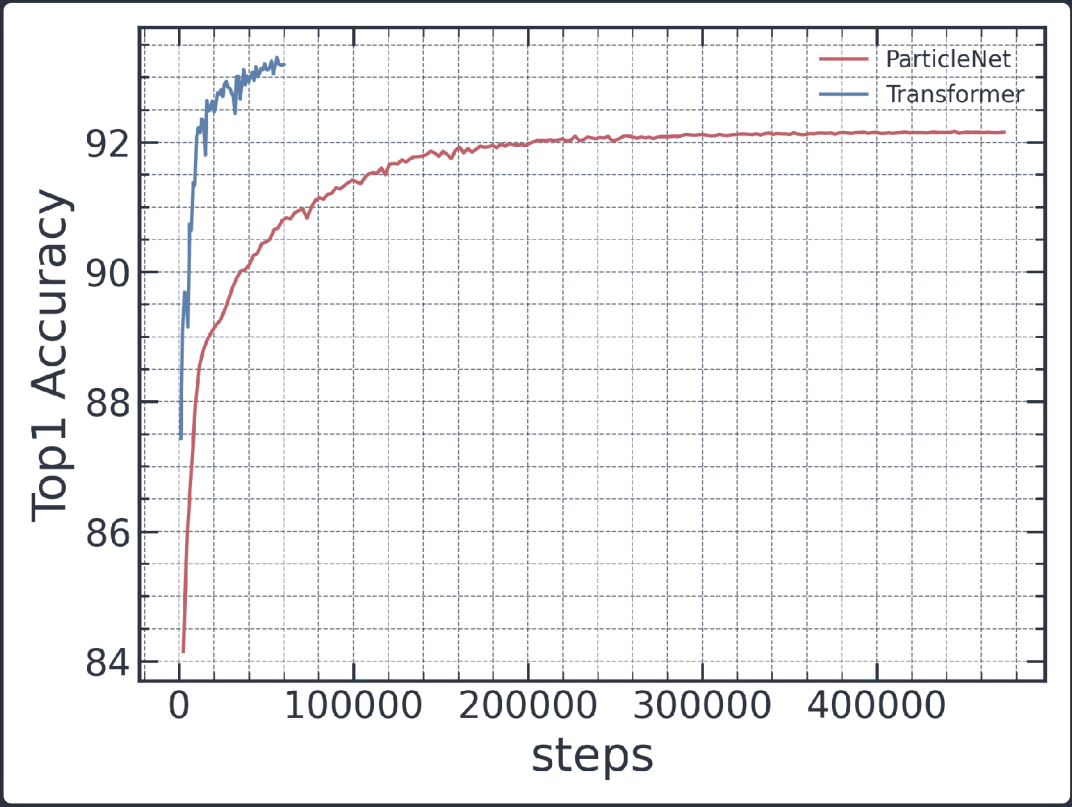
Performance : ParticleNet

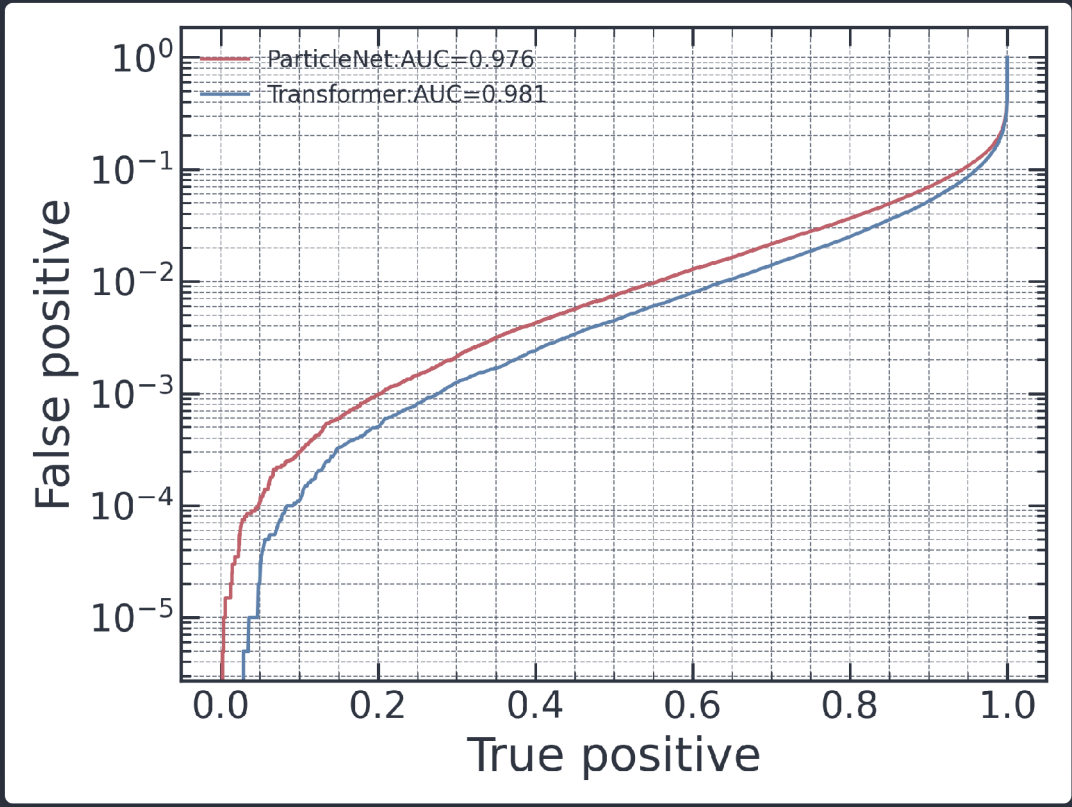




- Slightly worse than Transformer
- Note that : difference from the paper
  - We used a bigger batch size and a different learning rate scheduler. The original learning scheduler cannot handle training
  - The number of total steps is much more significant due to different batch size compared with Transformer training

Performance : Comparison





- Slightly worse(roughly 1%) than Transformer, *both results are worse than the paper!!*
- Even if my implementation/training of the ParticleNet is wrong, the Transformer performs similarly.
- At least the Transformer has good ability for the top tagger?

Sample with Clusters

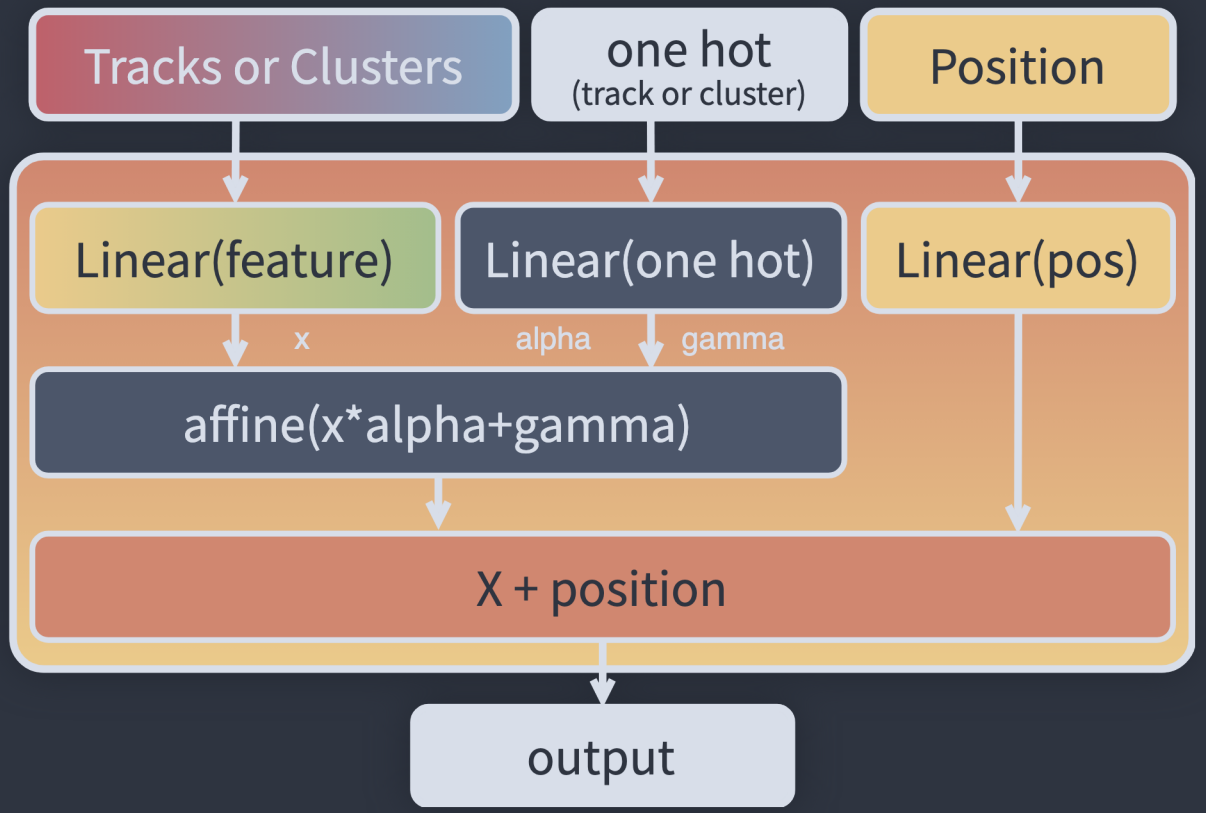
Original sample

- To test model performance with more realistic situations, add cluster information
- Making new samples by Delphes with more labels:
  - Lables: total 8 labels, $l, s, c, b, g, l\bar{l}, s\bar{s}, c\bar{c}, b\bar{b}$ where l is u or d
  - $Z \to q\bar{q}, H \to s\bar{s}/c\bar{c}/b\bar{b}, g \to u\bar{u}/d\bar{d}/s\bar{s}/c\bar{c}/b\bar{b}$,other
  - Label composition:

Label	uu/dd	ss	cc	bb	u/d	s	c	b	gluon
	2.9%	9.0%	8.8%	8.0%	24.3%	8.24%	9.3%	15.4%	14.1%

- Using Delphes with ATLAS geometry
  - MG5_aMC of v3.2.0 w/ Pythia8
  - Delphes with ATLAS no pileup card using a partcle flow based jets
- Generate pp
  - Selection : $p_{\mathrm{T}} > 250$ GeV and $|\eta| < 2$
  - At least one jet constituent(track or cluster)

Model Architechture : Tokenizer with Clusters



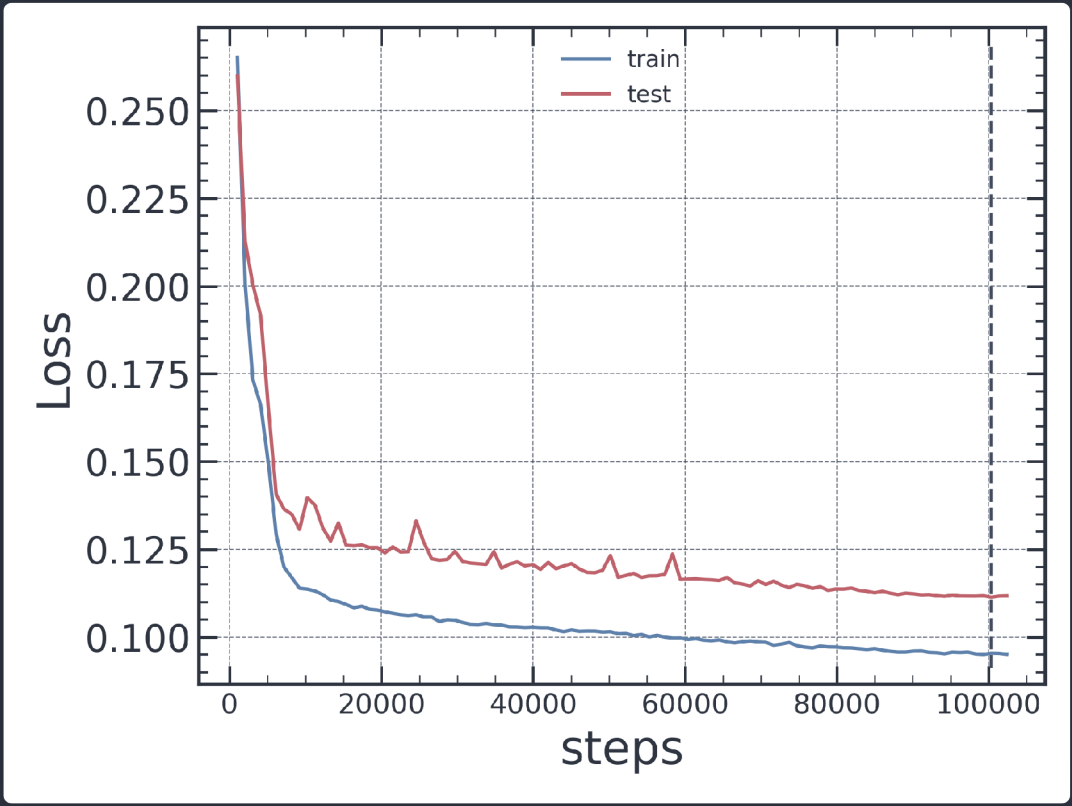
Tokenizer

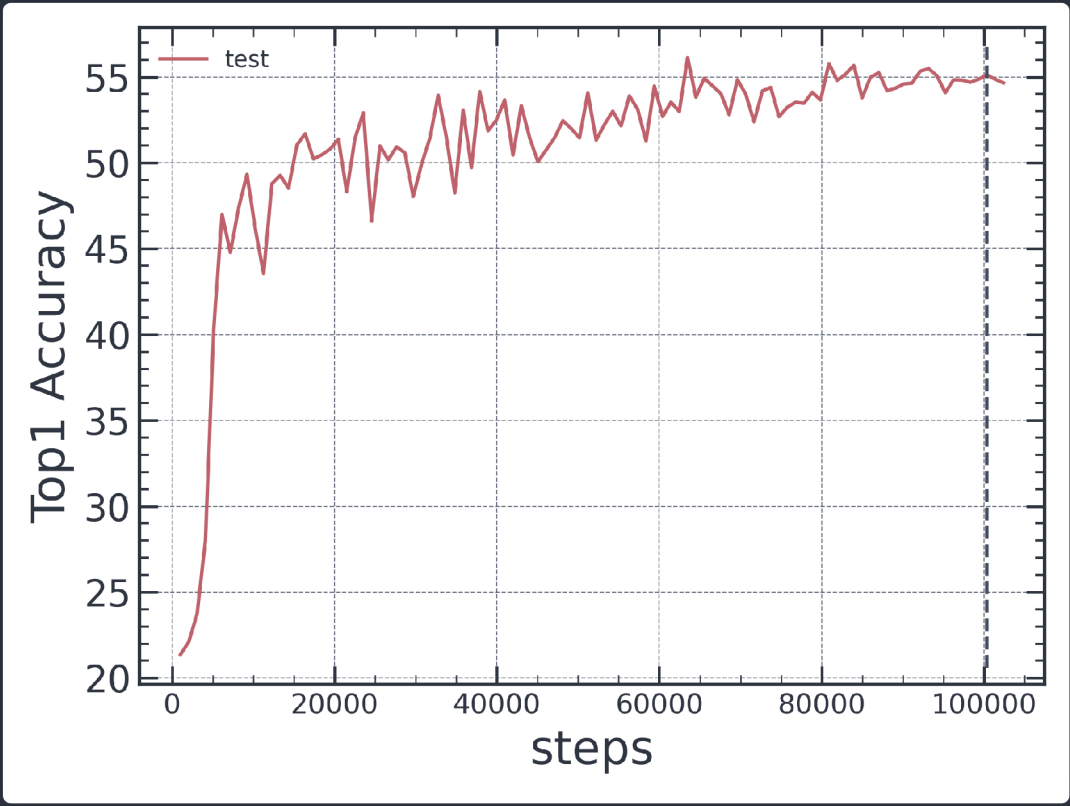
- Tokenize tracks and clusters as tokens
- Absorb the feature differences between tracks and clusters
- one hot vector are additionally taken as input
- Affine transformation is applied

Feature

- input feature
  - Tracks: p_{T}, $\Delta\eta$, $\Delta\phi$, d_0
  - Cluster: p_{T}, $\Delta\eta$, $\Delta\phi$, $E_{\mathrm{EM}}/E_{\mathrm{total}}$, $E_{\mathrm{Had}}/E_{\mathrm{total}}$
- Position : $\Delta\eta$, $\Delta\phi$ for both tracks and clusters
- Tracks and clusters are input into the same linear layer

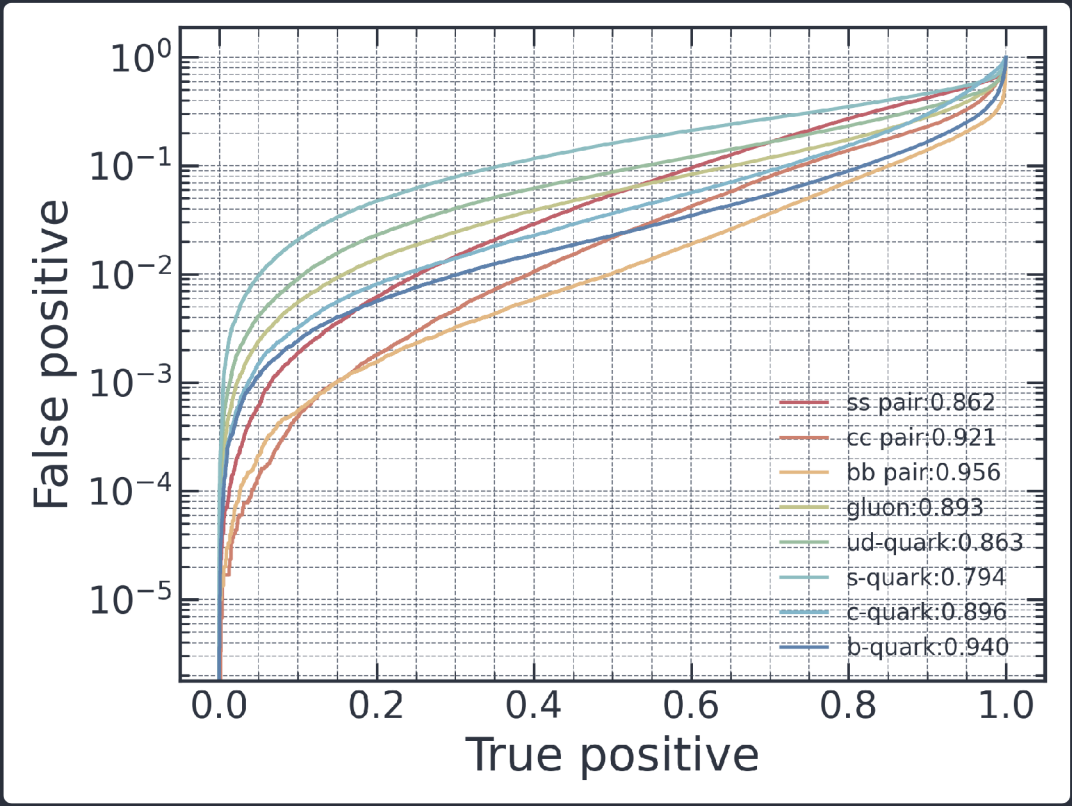
Orignal Samples

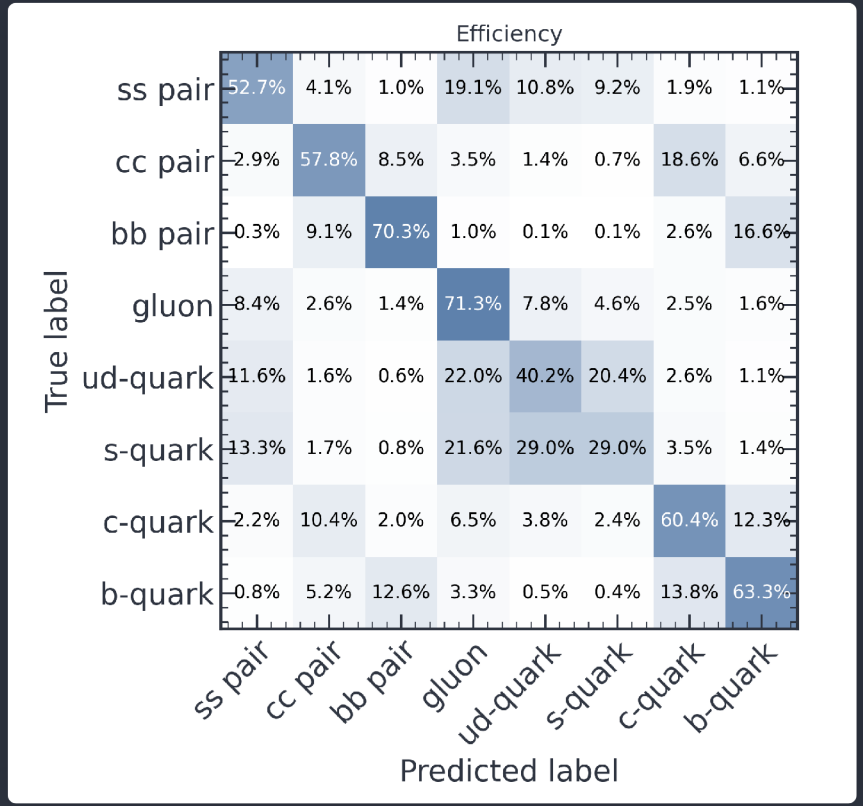




- Transformer with 4M samples for 200 epochs.
  - Difference between train/test comes from difference of sample composition between train/test samples.
- ParticleNet cannot be converged, or poor separation power

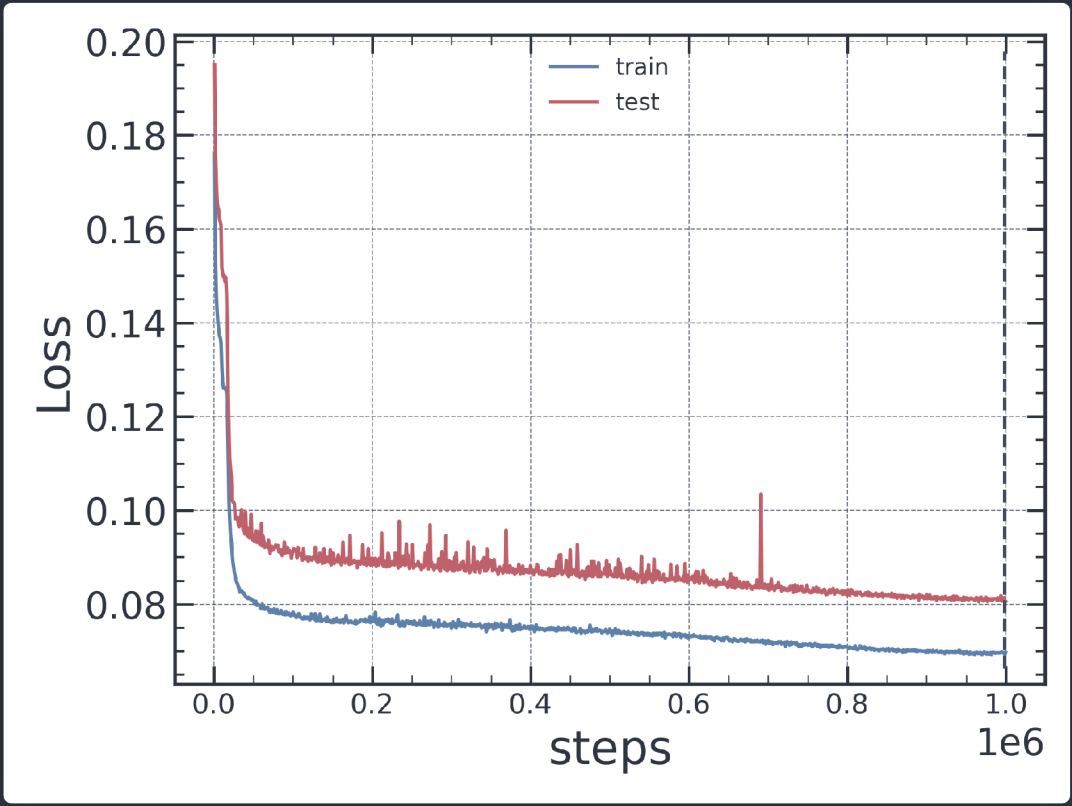
Orignal Samples

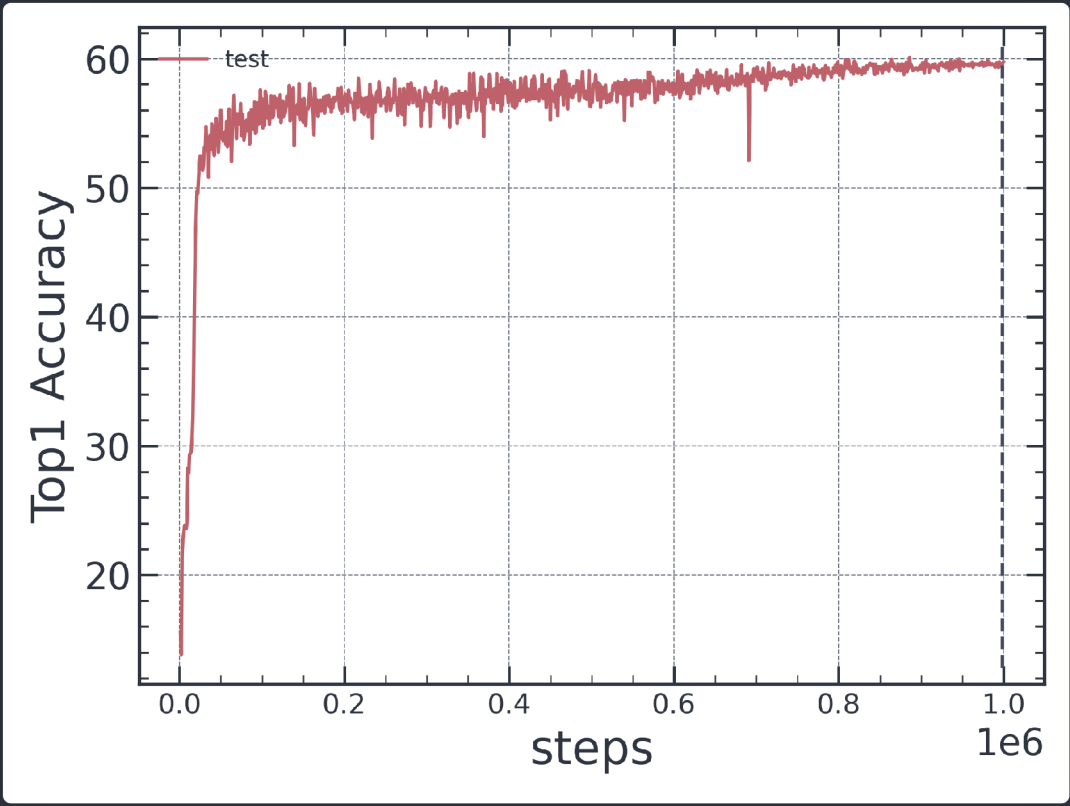




- Transformer with 4M samples for 200 epochs.
- The most difficult pair is strange and up/down separation.

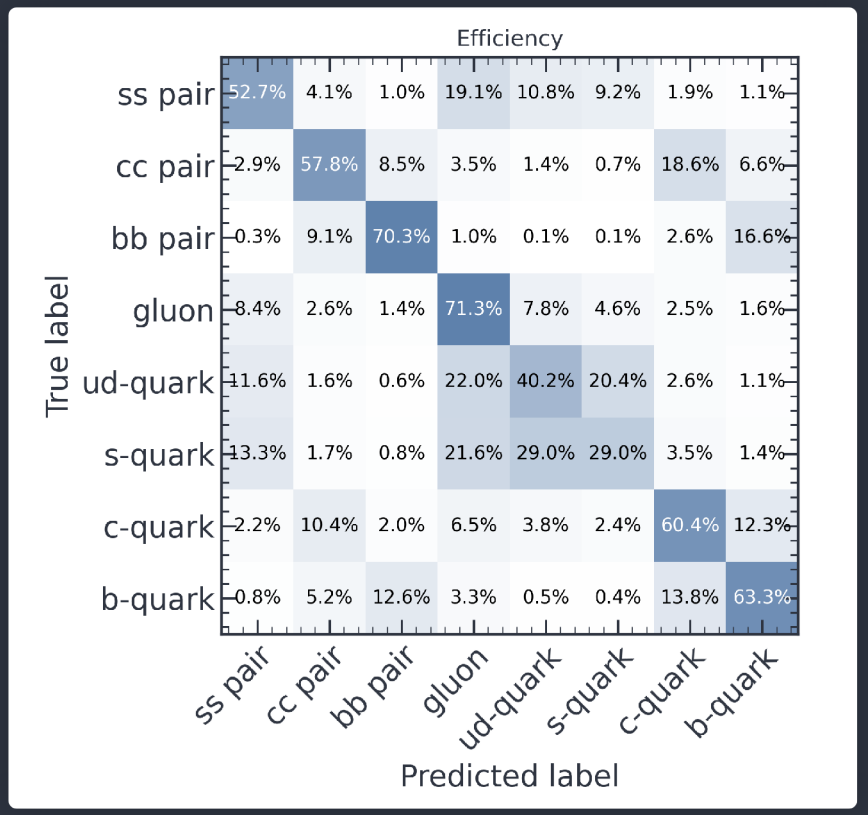
Orignal Samples : More Samples



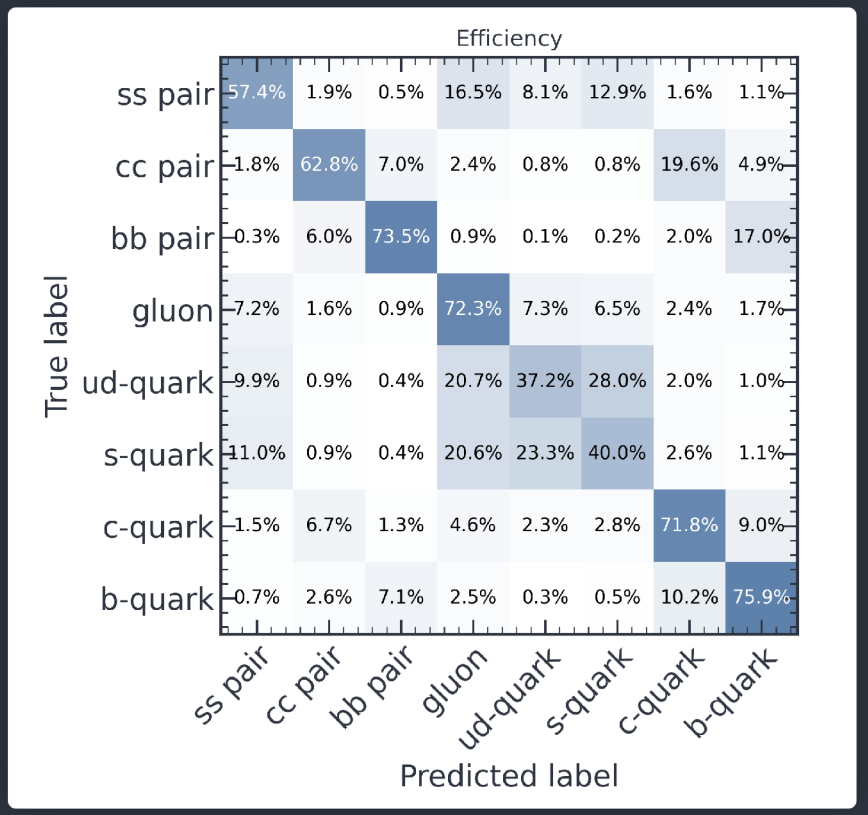


- Transformer with 256M samples for 64 epochs.
- Compared with 4M results, better loss values.
  - Training time about 1 week

Orignal Samples : Comparison



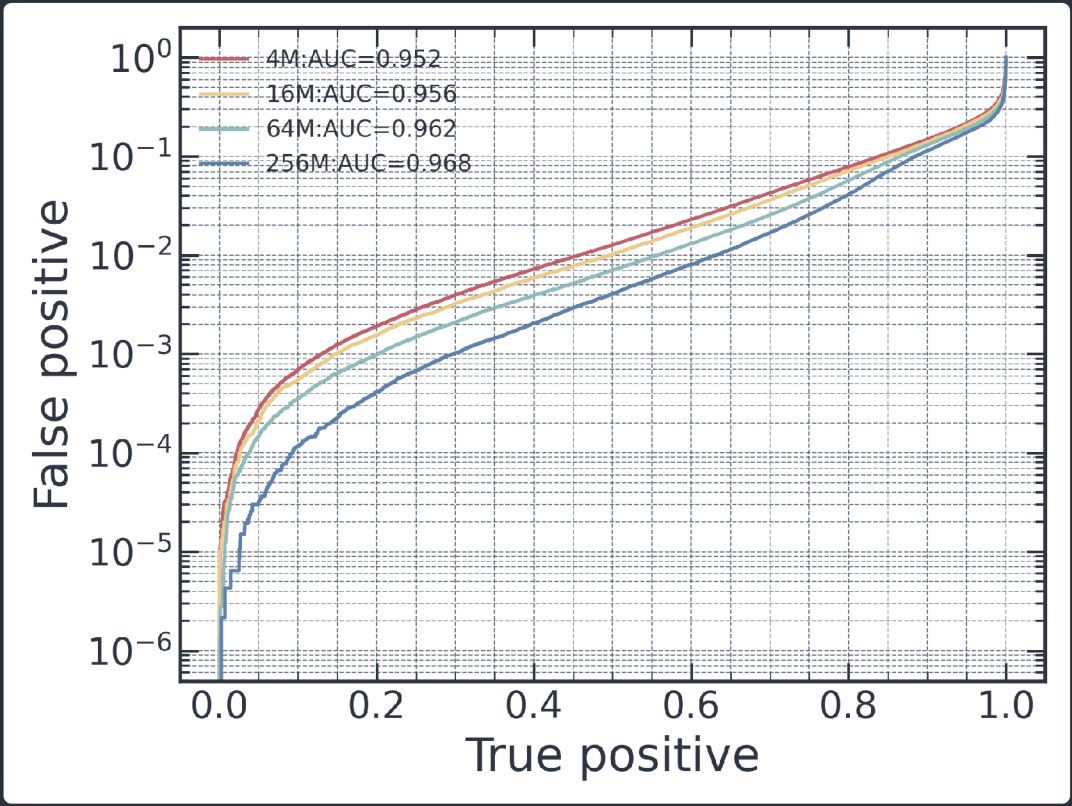
4M



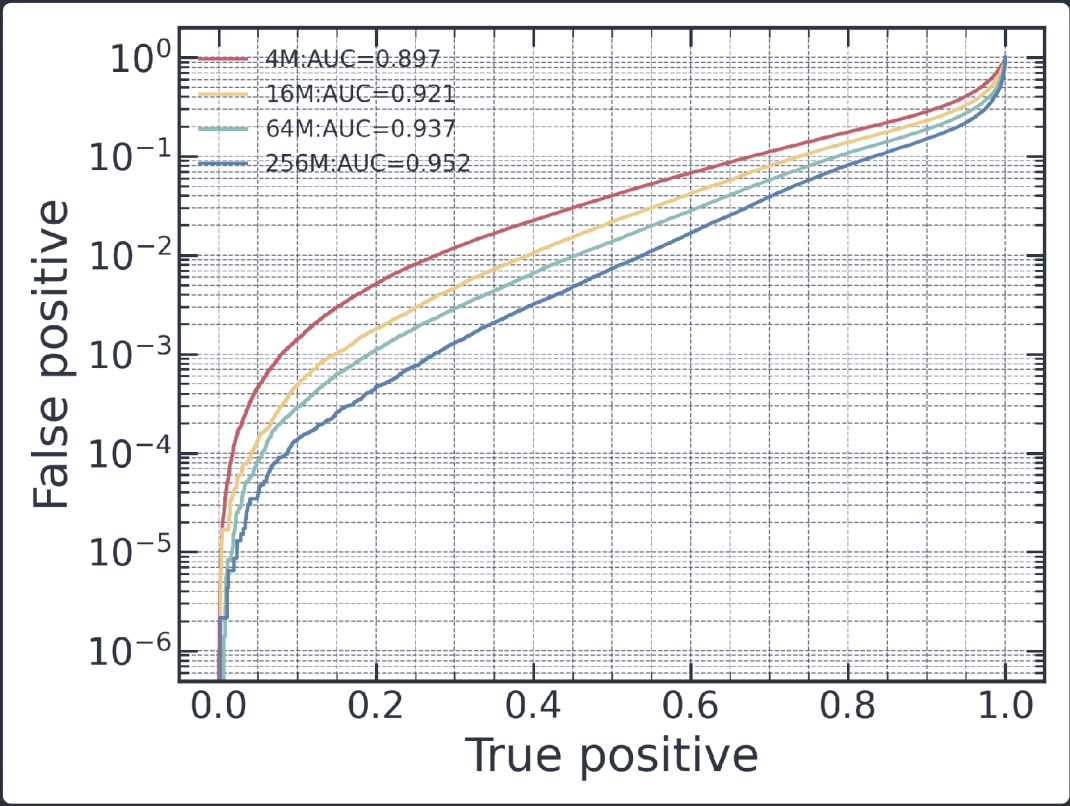
256M

- Training with 256M samples improved most of the values.

Orignal Samples : Dataset size



$b\bar{b}$ pair



$c\bar{c}$ pair

- Better performance with a larger dataset.
- Training with 256M samples took about one week $\rightarrow$ need to improve.

Conclusion

Conclusion

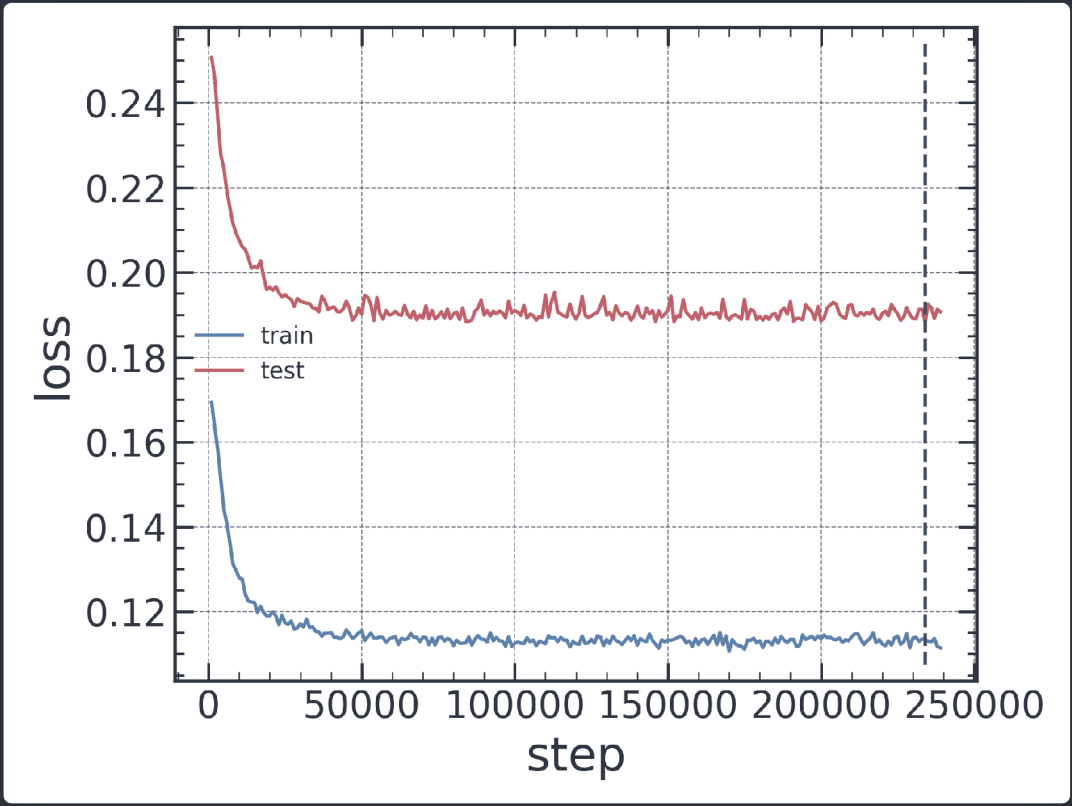
- A flavor tagger with all-in-one labels seems to be working.
- Transformer would be a good candidate as a jet flavor tagger.
- Tracks and clusters can be used simultaneously as inputs to the Transformer.
- Again, the amount of data is totally justice.

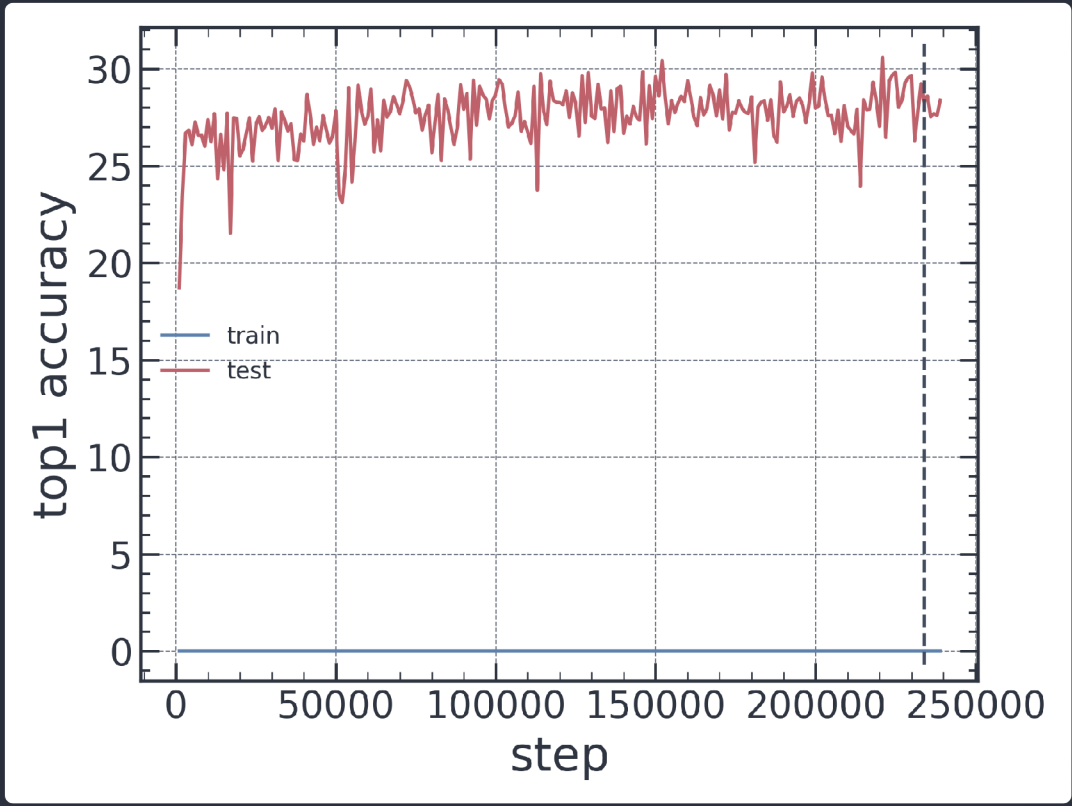
Plans

- Check the scaling law for our data
  - More data
  - Bigger model
  - More training epochs
- The ultimate goal : Supervised training with real data

Backup

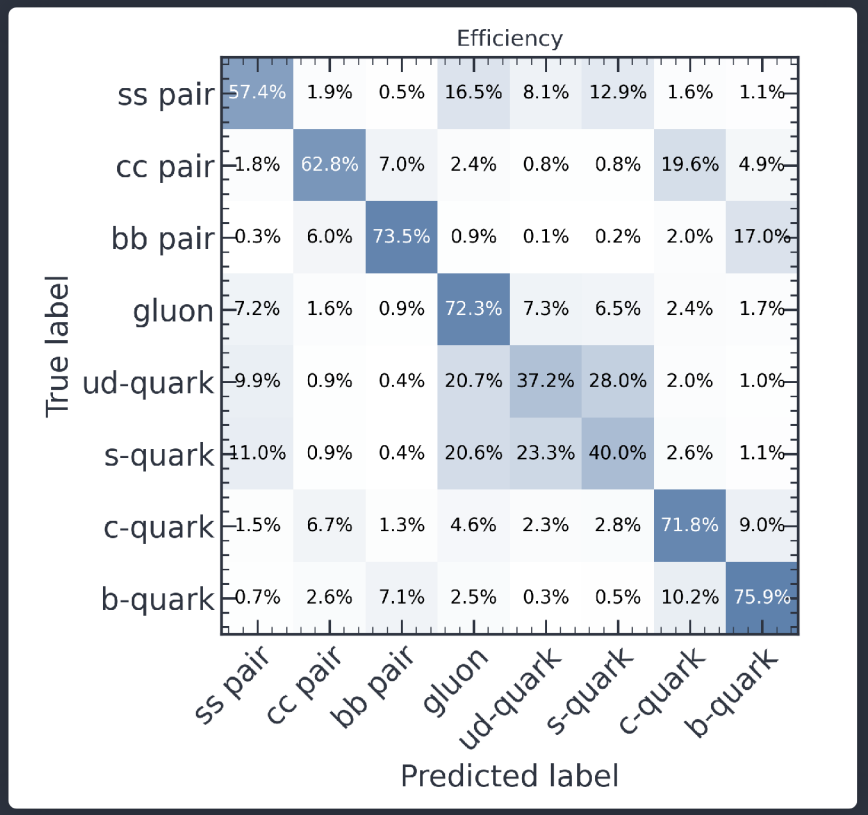
Original Samples with ParticleNet



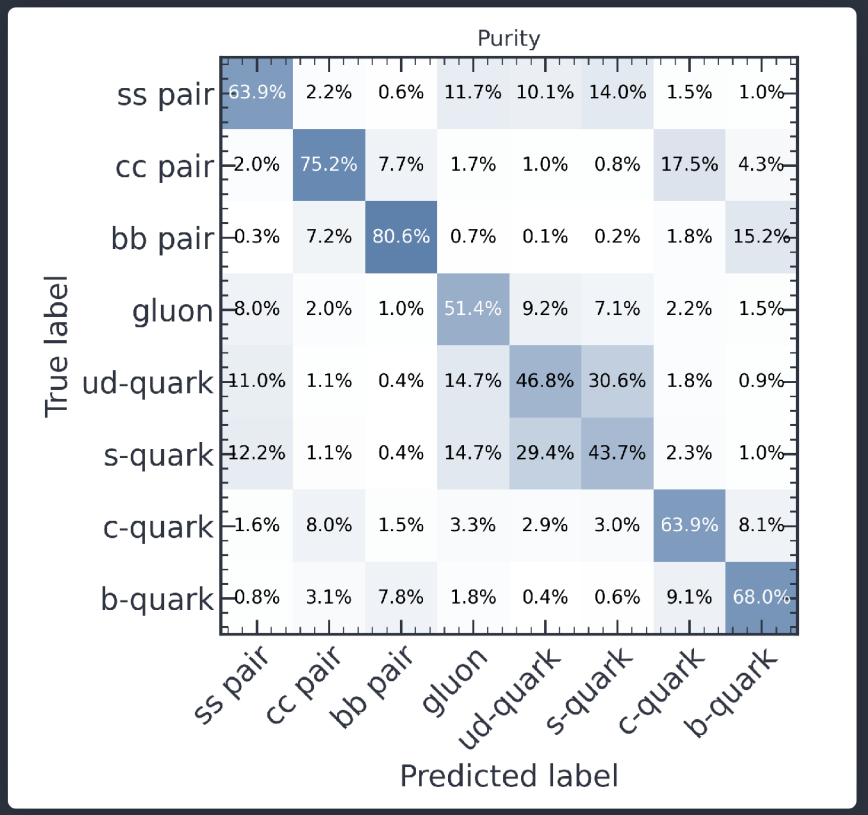


- ParticleNet with original samples
- No hyperparameter optimization

Orignal Samples : Comparison



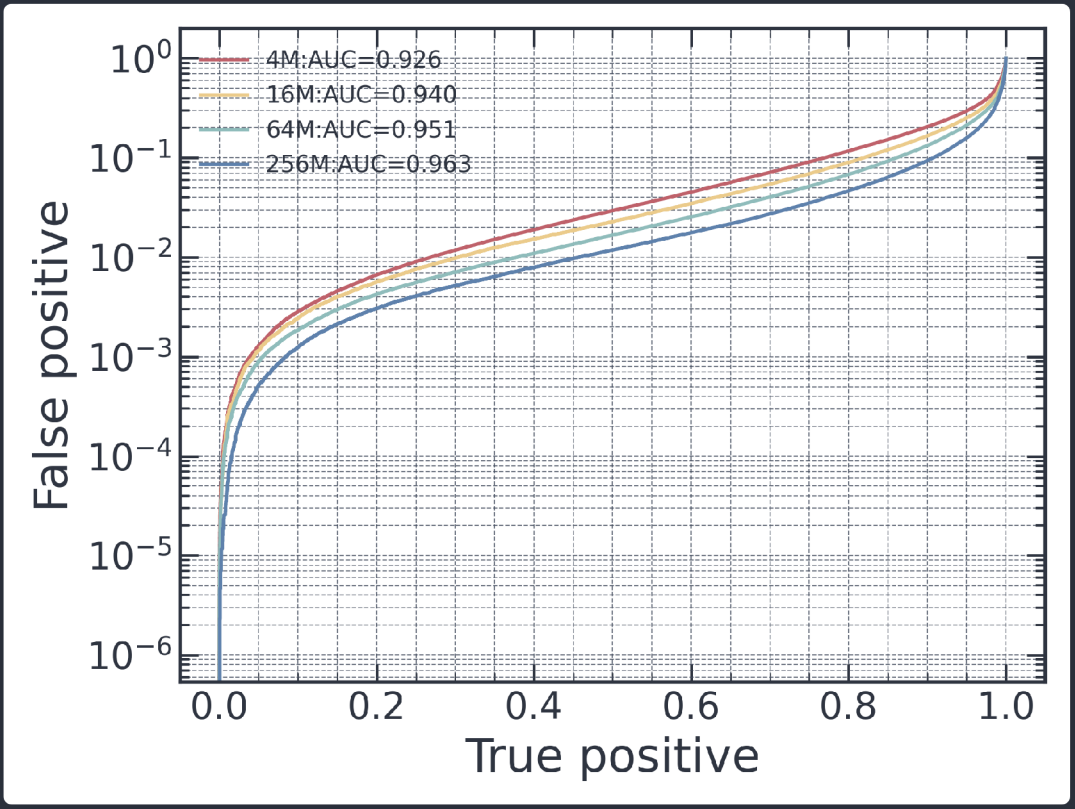
Efficiency



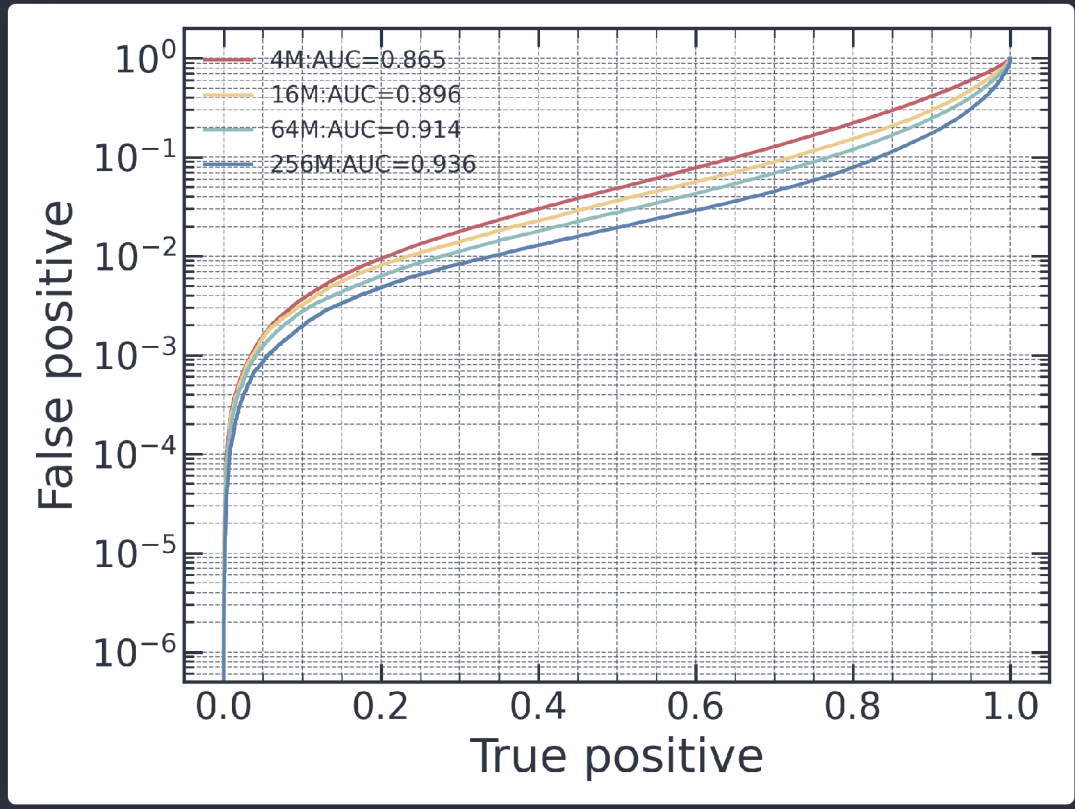
Purerity

- Training with 256M samples improved most of the values.

Orignal Samples : Dataset size

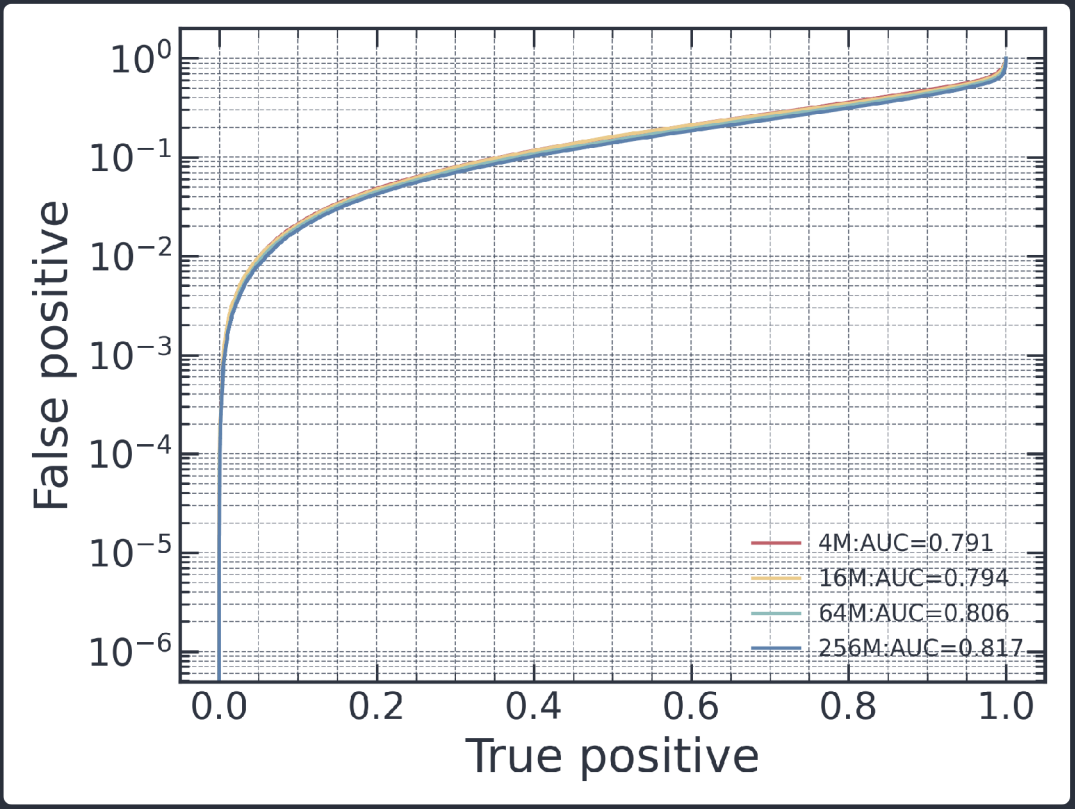


b-quark

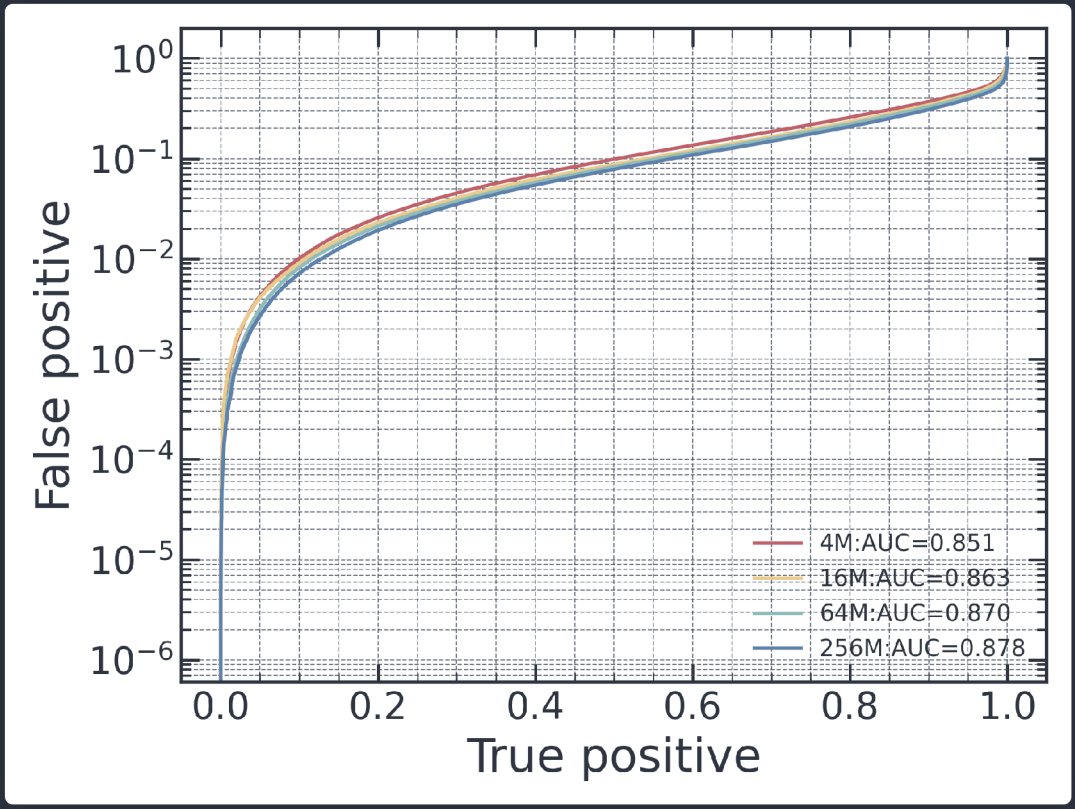


c-quark

Orignal Samples : Dataset size

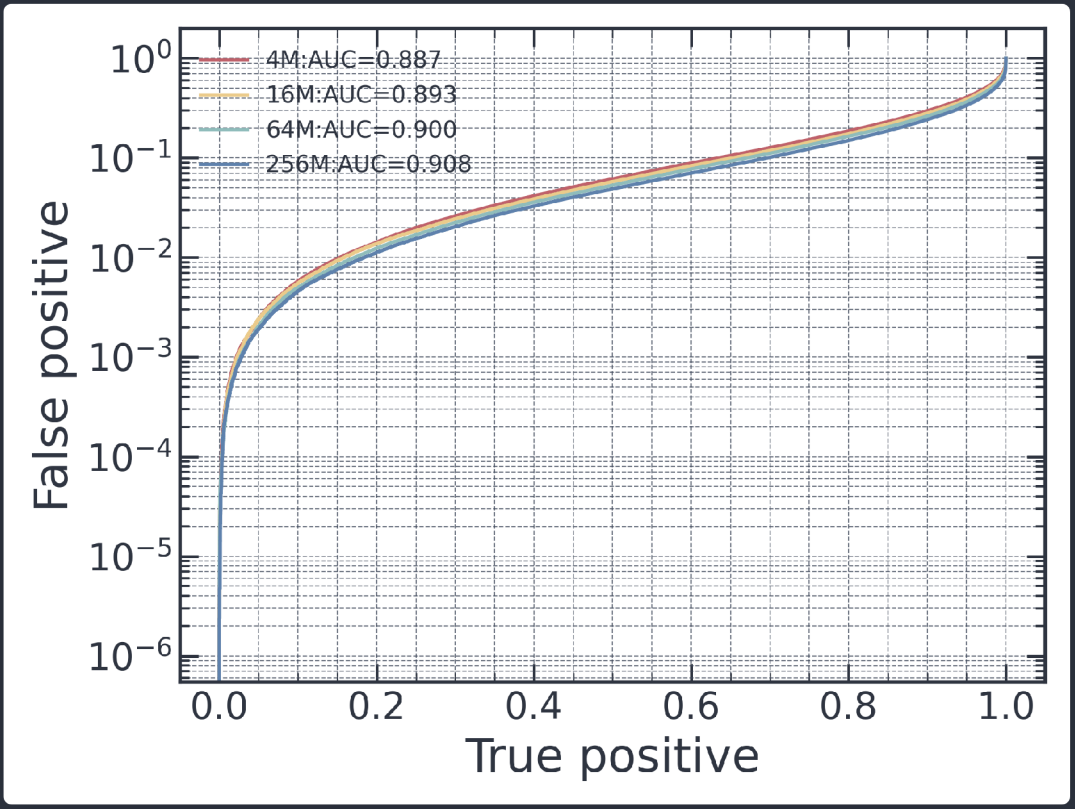


c-quark

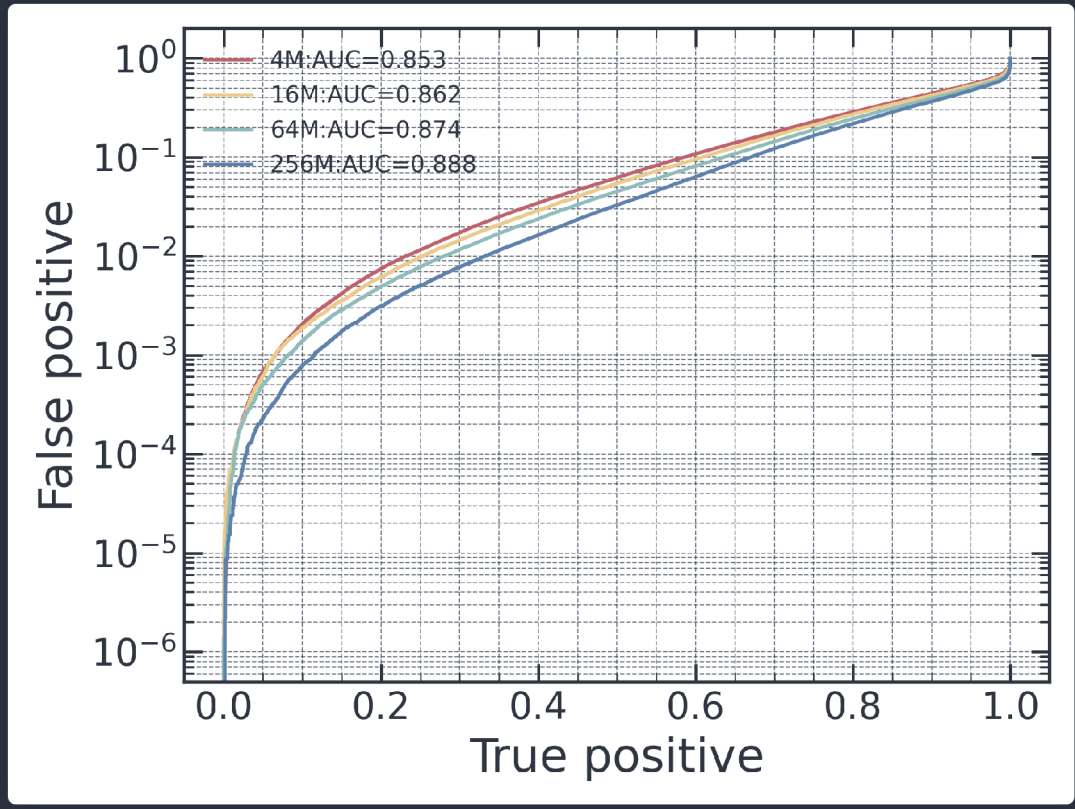


u/d-quark

Orignal Samples : Dataset size



gluon



$s\bar{s}$ pair