Mod-ParT: Physics Aware and Efficient Jet Tagging

A performance efficiency study across jet-classification tasks

Osman BAYRAKTAR, Hale SERT

Istanbul University

ML4Jets 2026

Vienna, Austria

14 September 2026

Why Physics Aware and Efficient Jet Tagging?

- Jet tagging is central to many analyses at the LHC.
- Particle level transformers provide strong discrimination, but attention can be computationally demanding.
- Physics motivated pairwise information is especially useful for resolving jet substructure.

Goal: retain competitive tagging performance while reducing model size, latency, and peak GPU memory.

Primary benchmark

Top-quark jets

versus

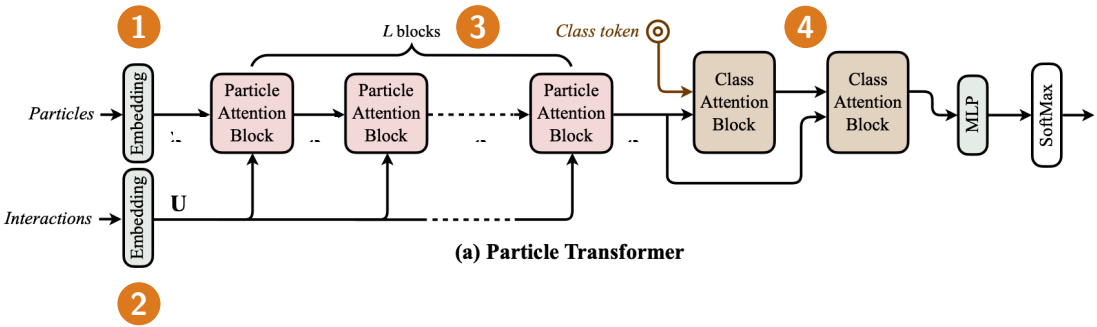
QCD jets

Main evaluation

Accuracy and ROC AUC

Background rejection at 50% and 30% signal efficiency

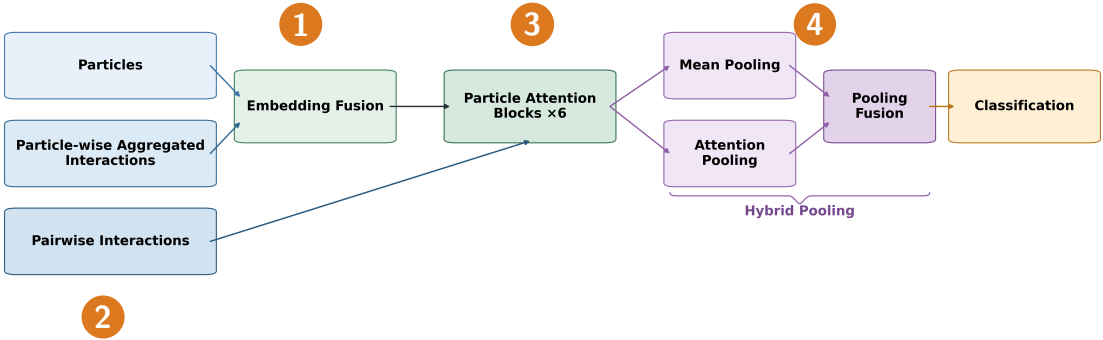
Particle Transformer (ParT)



(a) Particle Transformer

1 Particle embedding
2 Pairwise interaction bias
3 Eight particle-attention blocks
4 Class-token attention readout

From ParT to Mod-ParT



1 **Relation-aware** embedding fusion
2 Enriched **pairwise interactions**
3 Six particle-attention blocks
4 **Hybrid Pooling** readout

Exact Mod-ParT: **1.295M trainable parameters**

Datasets and Evaluation Protocol

Task	Dataset	Training	Role
Top tagging	TopTag(reference)	1.2M jets	Primary benchmark
	TopTagXL	10M jets	Large scale test
Quark gluon	Pythia 8 based	1.6M jets	Cross task test

Predictive performance

- Accuracy and ROC AUC
- Background rejection $R_{\epsilon_S} = 1/\epsilon_B$

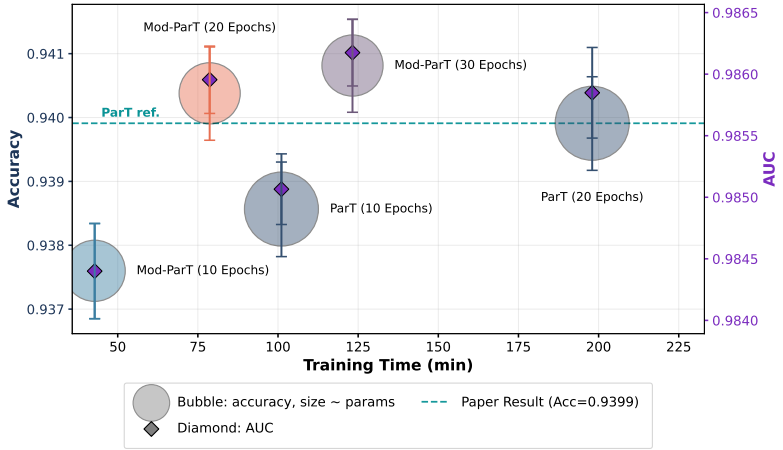
Computational performance

- Training throughput
- Latency and peak GPU memory
- Parameters and GFLOPs

Predictive Performance Results

Accuracy, ROC AUC, background rejection, and significance improvement

Main Result: Reference Top Tagging Benchmark

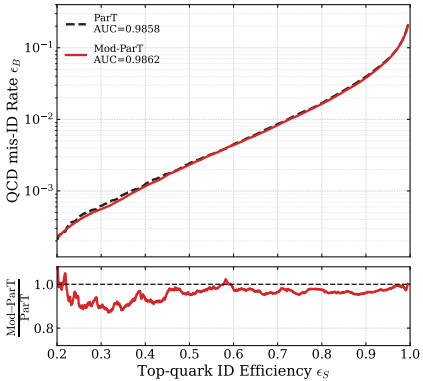


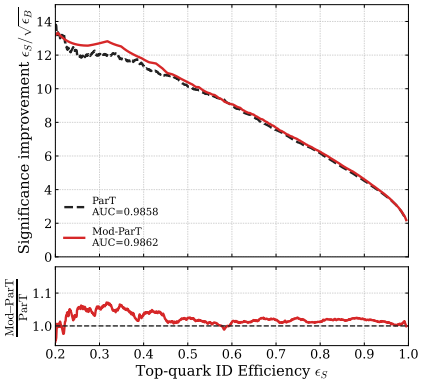
Six seed summary

- Accuracy: 0.9400 ± 0.0005
- ROC AUC: 0.9857 ± 0.0002
- Bkg rej 50%: 398 ± 23
- Bkg rej 30%: 1609 ± 66

The figure summarizes the trade-off between predictive performance, training time, and model size. Mod-ParT reaches the published ParT accuracy while remaining computationally efficient.

QCD Mistag Rate and Significance Improvement





Mod-ParT remains competitive with ParT across most of the signal-efficiency range. The relative gain or loss depends on the selected working point; therefore, AUC, QCD mistag rate, and significance improvement should be interpreted together.

Supporting Results Beyond the Primary Benchmark

TopTagXL: large scale top tagging

- 10M training jets, 20 epochs
- 25M test jets

	ParT	**Mod-ParT**
Accuracy	0.957	**0.951**
ROC AUC	0.992	**0.990**
Bkg. rej 50%	1345	**842**
Bkg. rej 30%	7321	**4143**

Pythia 8 quark gluon discrimination

- 1.6M/200k/200k train/val/test split
- Kinematics, charge, and PID inputs

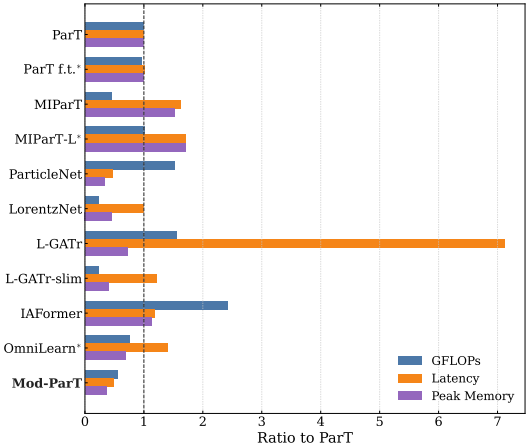
	ParT	**Mod-ParT**
Accuracy	0.840	**0.841**
ROC AUC	0.912	**0.912**
Bkg. rej 50%	41.3	**40.85**
Bkg. rej 30%	101.2	**102.77**

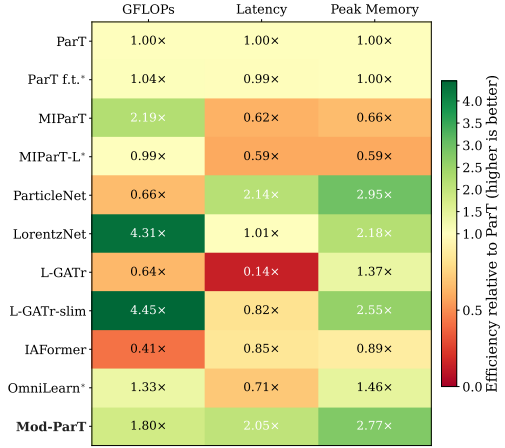
Mod-ParT achieves comparable performance to ParT for quark gluon discrimination, while ParT performs better on the larger TopTagXL dataset.

Computational Performance Metrics

Parameters, GFLOPs, latency, throughput, and peak GPU memory

Matched A30 Computational Efficiency





Result: Mod-ParT outperforms all other models under the same conditions, using the same dataset and hardware.

Conclusions

1. **Physics aware design:** relational representations and enriched pairwise attention are incorporated into a compact ParT variant.
2. **Competitive tagging:** Mod-ParT gives comparable performance on the primary top tagging benchmark when trained from scratch, with supporting results on TopTagXL and quark gluon discrimination.
3. **Improved efficiency:** matched A30 tests show higher throughput, fewer parameters, and lower peak memory.

Overall, Mod-ParT maintains competitive tagging performance while reducing computational cost and model size.

Acknowledgments

We would like to thank **COST Action CA24146**
Machine Learning and Quantum Computing for Future Colliders
for supporting our participation in this event.

Thank you

Questions?

Backup: Six-Seed TopTag Result

Seed	Accuracy	ROC AUC	Background rejection 50%	Background rejection 30%
45	0.940143	0.985721	387.82	1566.56
46	0.939752	0.985600	376.89	1672.80
48	0.940579	0.985965	432.79	1527.18
49	0.940554	0.986034	417.90	1678.52
50	0.939762	0.985637	395.92	1649.75
51	0.939267	0.985413	378.50	1559.81
Mean $\pm$ SD	0.9400 ± 0.0005	0.9857 ± 0.0002	398 ± 23	1609 ± 66

- All runs use the same 30 epoch protocol and 403,999-jet test set.
- Rejection uses grouped ROC interpolation at the exact target signal efficiency.
- The uncertainty is the sample standard deviation across independent trainings.

Backup: Matched A30 Numbers

Model	Params	GFLOPs/jet	ms/batch	jets/s	Peak GiB
ParT	2.14M	0.699	97.33	1315	5.341
Mod-ParT	1.29M	0.389	47.45	2697	1.925
Efficiency ratio	1.66×	1.80×	2.05×	2.05×	2.77×

Protocol and interpretation

Batch size 128, sequence length 128, matched inputs and precision on an NVIDIA A30. Throughput is 128/batch latency. GFLOPs are complemented by measured latency and memory because profiler coverage and GPU kernel efficiency differ.

Backup: Compute Benchmark Validity and Scope

Was the comparison matched?

- Same NVIDIA A30 and PyTorch 2.7/CUDA 12.6 stack
- Same real TopLandscape batch: 128 jets, 128 constituents
- Same AMP FP16/TF32 policy and benchmark harness
- 20 warm-up steps; CUDA synchronization before/after timing

What was timed?

- Inference: 100 timed forward steps
- Training: 50 steps including zero-grad, forward, loss, backward, AdamW step, and scaler update
- Data loading and epoch-level I/O were excluded

Measured Mod-ParT advantage over ParT

Metric	ParT	Mod-ParT
Forward GFLOPs/jet	0.669	0.389
Inference ms/batch	28.26	18.89
Training ms/batch	97.33	47.45
Peak train memory	5.341 GiB	1.925 GiB

1.50× inference speed; 2.05× training speed;
1.72× forward-FLOP and 2.77× train-memory efficiency.

Interpretation boundary

- Hardware-, batch-, sequence-length-, and backend-dependent
- Not a batch-1 trigger-latency or energy measurement
- FLOPs are forward operations; separately profiled training FLOPs exclude the optimizer