Economical Jet Taggers

Equivariant, slim, and quantized

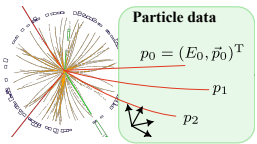
Antoine Petitjean

September 18th, 2026

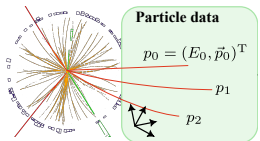
UNIVERSITÄT
HEIDELBERG
ZUKUNFT
SEIT 1386

Funded by
the European Union

Jet tagging



Jet tagging

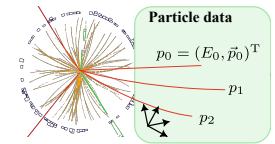


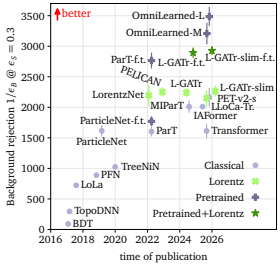
Point cloud $(p_1, \ldots, p_n)$ = set of vectors v_k with additional scalar features s_k

$$v = \begin{pmatrix} (E_0 & p_{x,0} & p_{y,0} & p_{z,0}) \\ \vdots & & & \\ (E_n & p_{x,n} & p_{y,n} & p_{z,n}) \end{pmatrix}$$

$$s = \begin{pmatrix} \mathrm{PID}_0 & \Delta\phi_0 & \Delta\eta_0 & \ldots \\ \vdots & & & \\ \mathrm{PID}_n & \Delta\phi_n & \Delta\eta_n & \ldots \end{pmatrix}$$

Jet tagging



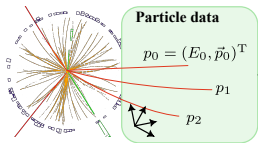


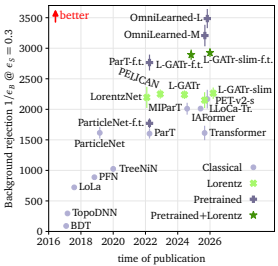
Point cloud $(p_1, \ldots, p_n)$ = set of vectors v_k with additional scalar features s_k

$$v = \begin{pmatrix} (E_0 & p_{x,0} & p_{y,0} & p_{z,0}) \\ \vdots & & & \\ (E_n & p_{x,n} & p_{y,n} & p_{z,n}) \end{pmatrix}$$

$$s = \begin{pmatrix} \mathrm{PID}_0 & \Delta\phi_0 & \Delta\eta_0 & \ldots \\ \vdots & & & \\ \mathrm{PID}_n & \Delta\phi_n & \Delta\eta_n & \ldots \end{pmatrix}$$

Jet tagging

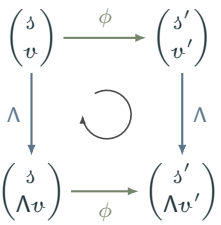




Point cloud $(p_1, \ldots, p_n)$ = set of vectors v_k with additional scalar features s_k

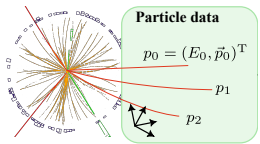
$$v = \begin{pmatrix} (E_0 & p_{x,0} & p_{y,0} & p_{z,0}) \\ \vdots & & & \\ (E_n & p_{x,n} & p_{y,n} & p_{z,n}) \end{pmatrix}$$

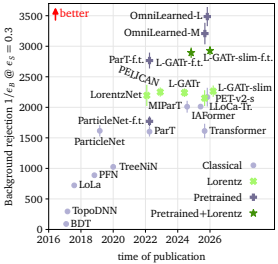
$$s = \begin{pmatrix} \mathrm{PID}_0 & \Delta\phi_0 & \Delta\eta_0 & \ldots \\ \vdots & & & \\ \mathrm{PID}_n & \Delta\phi_n & \Delta\eta_n & \ldots \end{pmatrix}$$



Lorentz equivariance

Jet tagging

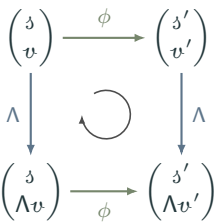




Point cloud $(p_1, \ldots, p_n)$ = set of vectors v_k with additional scalar features s_k

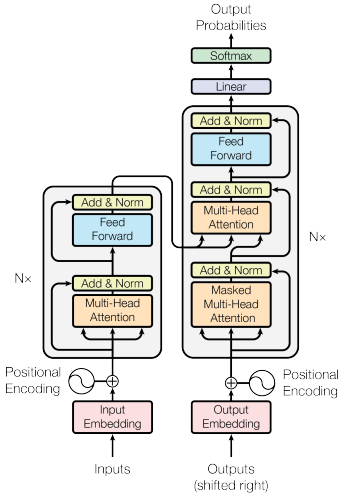
$$v = \begin{pmatrix} (E_0 & p_{x,0} & p_{y,0} & p_{z,0}) \\ \vdots & & & \\ (E_n & p_{x,n} & p_{y,n} & p_{z,n}) \end{pmatrix}_{n\times 1\times 4}$$

$$s = \begin{pmatrix} \mathrm{PID}_0 & \Delta\phi_0 & \Delta\eta_0 & \ldots \\ \vdots & & & \\ \mathrm{PID}_n & \Delta\phi_0 & \Delta\eta_0 & \ldots \end{pmatrix}_{n\times m\times 1}$$



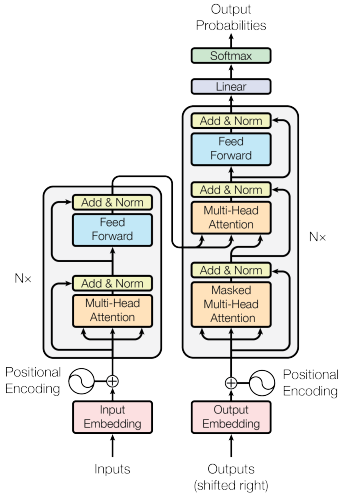
Lorentz equivariance

Standard transformer



Vaswani et al., 2017

Standard transformer



Vaswani et al., 2017

Linear

$$\text{Linear}(x) = Wx + b$$

Attention

$$A_{ij} = \frac{q_i \cdot k_j}{\sqrt{c}}$$

$$\text{out}_i = \sum_j \text{Softmax}_j(A_{ij})\, v_j$$

Other operations

- Activation function
- Layer norm

L-GATr-slim layers

Linear

$$\text{Linear}(x) = Wx + b$$

Attention

$$A_{ij} = \frac{q_i \cdot k_j}{\sqrt{c}}$$

$$\text{out}_i = \sum_j \text{Softmax}_j(A_{ij})\, v_j$$

L-GATr-slim layers

Linear

$$\text{Linear}(x) = Wx + b$$

Linear

$$\text{Linear}\begin{pmatrix} \mathfrak{s} \\ \mathfrak{v} \end{pmatrix} = \begin{pmatrix} w_{\mathfrak{s}}\,\mathfrak{s} + b_{\mathfrak{s}} \\ w_{\mathfrak{v}}\,\mathfrak{v} \end{pmatrix}$$

Attention

$$A_{ij} = \frac{q_i \cdot k_j}{\sqrt{c}}$$

$$\text{out}_i = \sum_j \text{Softmax}_j(A_{ij})\, v_j$$

L-GATr-slim layers

Linear

$$\text{Linear}(x) = Wx + b$$

Attention

$$A_{ij} = \frac{q_i \cdot k_j}{\sqrt{c}}$$

$$\text{out}_i = \sum_j \text{Softmax}_j(A_{ij})\, v_j$$

Linear

$$\text{Linear}\begin{pmatrix} \mathfrak{s} \\ \mathfrak{v} \end{pmatrix} = \begin{pmatrix} w_{\mathfrak{s}}\, \mathfrak{s} + b_{\mathfrak{s}} \\ w_{\mathfrak{v}}\, \mathfrak{v} \end{pmatrix}$$

- $(n, m, 1) \mapsto (n, c_{\mathfrak{s}}, 1)$
- $(n, 1, 4) \mapsto (n, c_{\mathfrak{v}}, 4)$

L-GATr-slim layers

Linear

$$\text{Linear}(x) = Wx + b$$

Linear

$$\text{Linear}\begin{pmatrix} \mathfrak{s} \\ \mathfrak{v} \end{pmatrix} = \begin{pmatrix} w_{\mathfrak{s}}\,\mathfrak{s} + b_{\mathfrak{s}} \\ w_{\mathfrak{v}}\,\mathfrak{v} \end{pmatrix}$$

Attention

$$A_{ij} = \frac{q_i \cdot k_j}{\sqrt{c}}$$

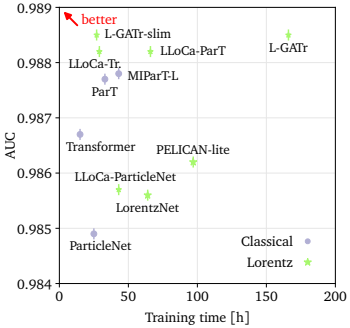
$$\text{out}_i = \sum_j \text{Softmax}_j(A_{ij})\, v_j$$

Attention

$$A_{ij} = \frac{q_{\mathfrak{s},i} \cdot k_{\mathfrak{s},j} + \langle q_{\mathfrak{v},i}, k_{\mathfrak{v},j} \rangle}{\sqrt{c_{\mathfrak{s}} + 4c_{\mathfrak{v}}}}$$

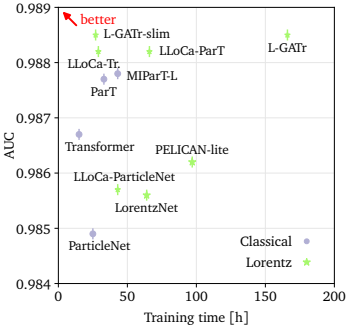
$$\text{out}_i = \sum_j \text{Softmax}_j(A_{ij}) \begin{pmatrix} v_{\mathfrak{s}} \\ v_{\mathfrak{v}} \end{pmatrix}_j$$

SOTA performance of L-GATr-slim

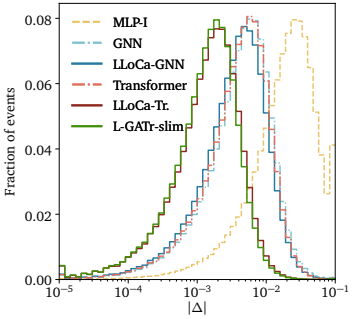


Jet tagging

SOTA performance of L-GATr-slim

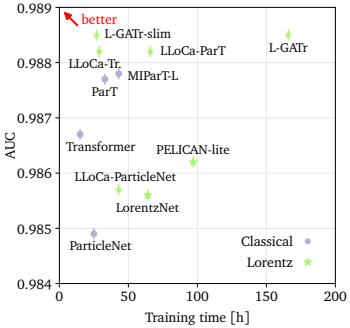


Jet tagging

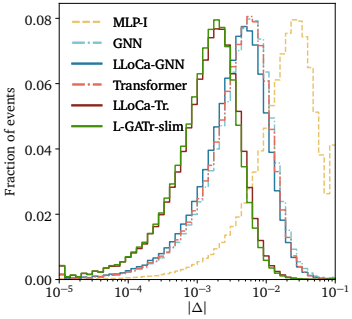


Amplitude regression

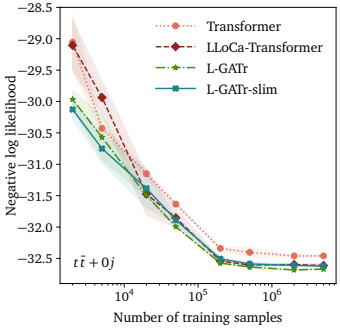
SOTA performance of L-GATr-slim



Jet tagging



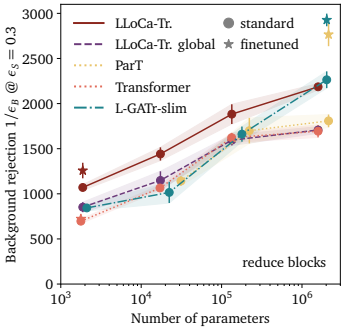
Amplitude regression



Event generation

Downscaling

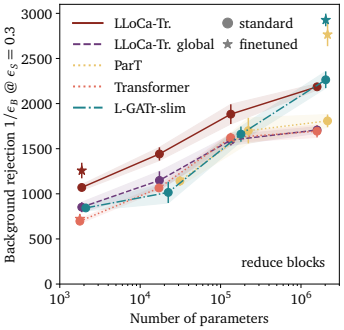
Top taggers from $O(10^6)$ down to $\mathbf{O(10^3)}$ **parameters**



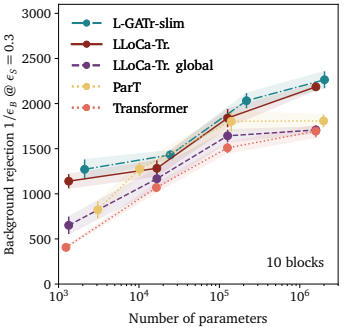
Fewer blocks: $10 \to 1$

Downscaling

Top taggers from $O(10^6)$ down to $\mathbf{O(10^3)}$ **parameters**

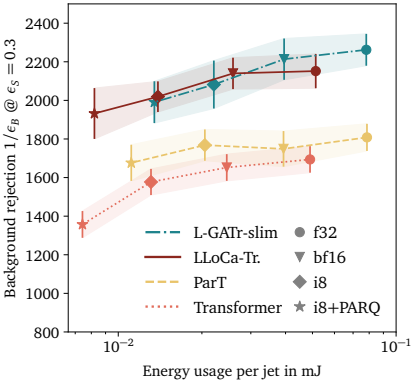


Fewer blocks: $10 \to 1$

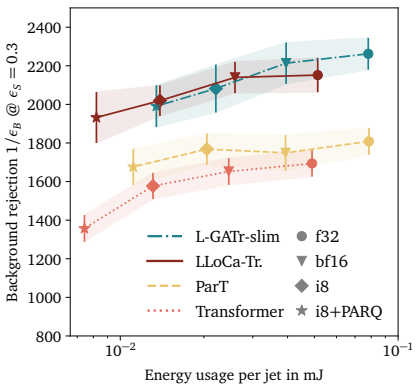


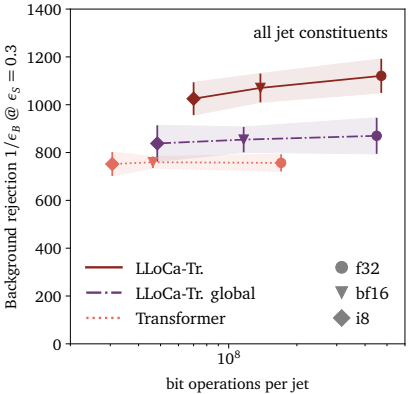
Narrower blocks: reduce latent dims

Quantization savings



Quantization savings





Conclusion

- **Simple equivariance**
  - L-GATr-slim = transformer with scalars and vectors streams
  - L-GATr performance at a much lower cost

Conclusion

- **Simple equivariance**
  - L-GATr-slim = transformer with scalars and vectors streams
  - L-GATr performance at a much lower cost
- **Down-scaling**
  - Shrink to 1k parameters
  - Quantization to `int8`

Conclusion

- **Simple equivariance**
  - L-GATr-slim = transformer with scalars and vectors streams
  - L-GATr performance at a much lower cost
- **Down-scaling**
  - Shrink to 1k parameters
  - Quantization to `int8`
- **Next steps**
  - Better L-GATr-slim performance at 1 block
  - Different quantization scheme (HGQ)
  - FPGA implementation

Conclusion

- **Simple equivariance**
  - L-GATr-slim = transformer with scalars and vectors streams
  - L-GATr performance at a much lower cost
- **Down-scaling**
  - Shrink to 1k parameters
  - Quantization to `int8`
- **Next steps**
  - Better L-GATr-slim performance at 1 block
  - Different quantization scheme (HGQ)
  - FPGA implementation

pip install lgatr

`arXiv:2512.17011`

Backup: Quantization

Matrix multiplications dominate the cost: run them at lower precision.

- inputs: zero-point quantization

$$x_q = \mathrm{clip}\left(\left\lfloor \frac{x}{s} \right\rceil + z\right)$$

- weights: quantization-aware training with proximal gradient methods (PARQ/STE)

	E_{add} [pJ]	E_{mul} [pJ]
`float32`	0.38	1.31
`bfloat16`	0.11	0.21
`int8`	0.007	0.07

Energy per operation, A100 GPU

Backup: weight quantization as a regularizer

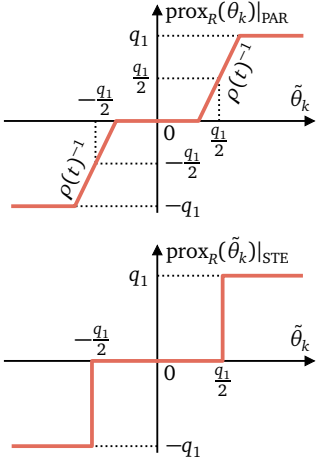
Push weights to ternary $\{0, \pm q\}$.

Constraint via proximal gradient

$$\mathcal{L}_{\text{QAT}} = \mathcal{L} + \lambda \sum_k R(\theta_k)$$

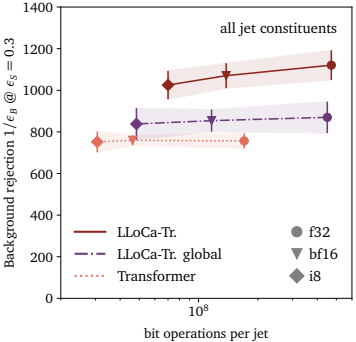
applied as a post-update map $\text{prox}_R(\tilde{\theta}_k)$

- **PARQ**: anneal map identity $\rightarrow$ step
- **STE**: hard step, special case

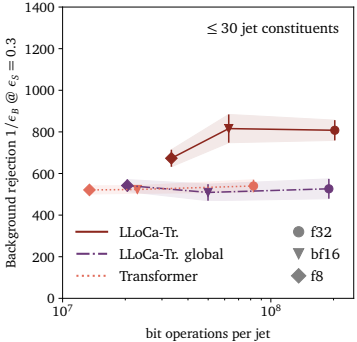


Backup: towards trigger-level tagging

Minimal LLoCa-Transformer: 1 block, 16-dim latent, static `int8`



All constituents



30 leading constituents

Backup: L-GATr-slim building blocks

$$\mathrm{GLU}\begin{pmatrix}\mathfrak{s}\\ v\end{pmatrix} = \begin{pmatrix}\mathrm{GELU}(\mathrm{Linear}(\mathfrak{s})) \cdot \mathrm{Linear}(\mathfrak{s})\\ \mathrm{GELU}(\langle \mathrm{Linear}(v), \mathrm{Linear}(v)\rangle) \cdot \mathrm{Linear}(v)\end{pmatrix}$$

$$\mathrm{RMSNorm}\begin{pmatrix}\mathfrak{s}\\ v\end{pmatrix} = \left(\frac{1}{c_v + c_\mathfrak{s}}\left(\sum_{c=1}^{c_v} |\langle v_c, v_c\rangle|^2 + \sum_{c=1}^{c_\mathfrak{s}} \mathfrak{s}_c^2\right) + \epsilon\right)^{-1/2}\begin{pmatrix}\mathfrak{s}\\ v\end{pmatrix}$$

Backup: LLoCa