

Neural Networks meet Lorentz Symmetry

Jonas Spinner

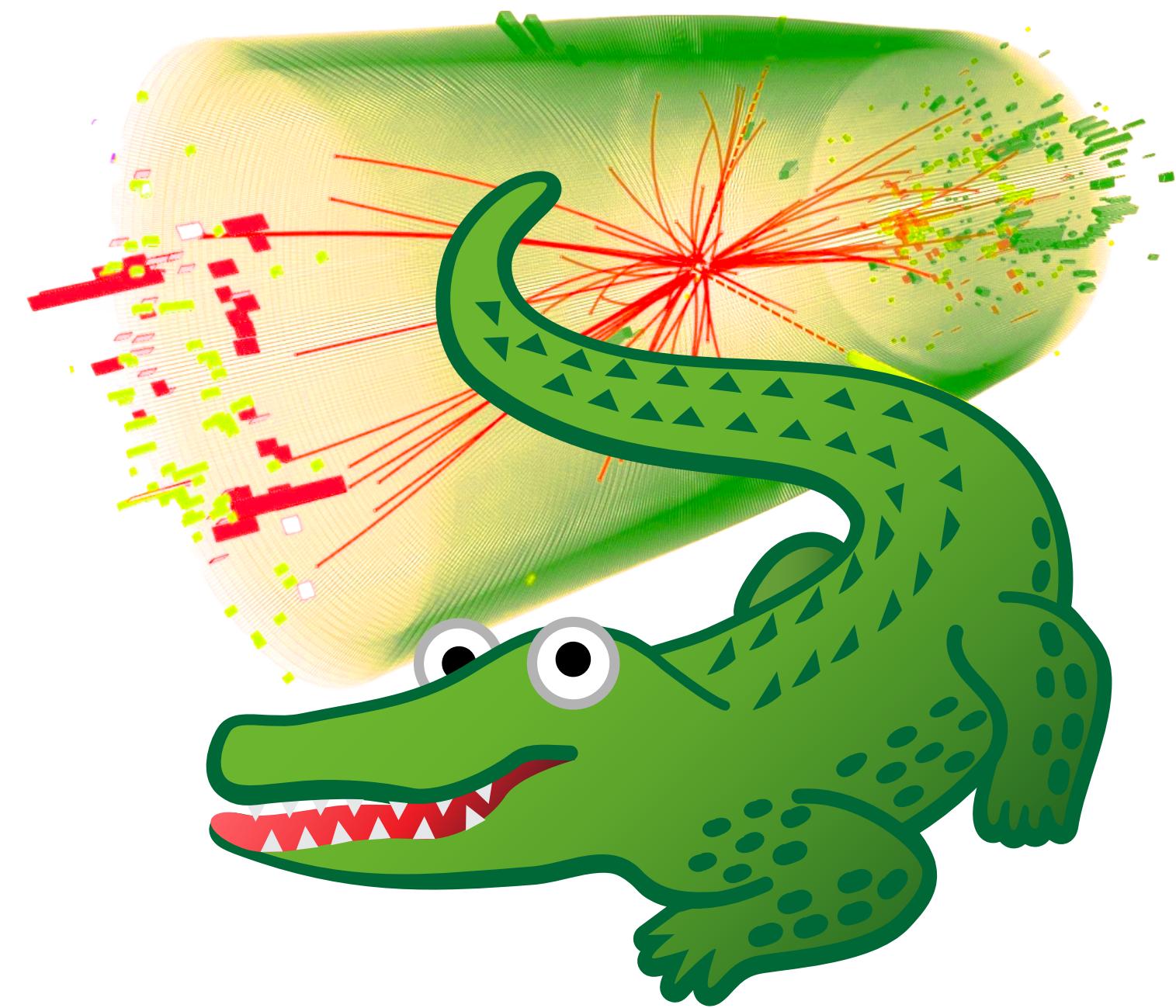
arXiv:2405.14806

arXiv:2411.00446

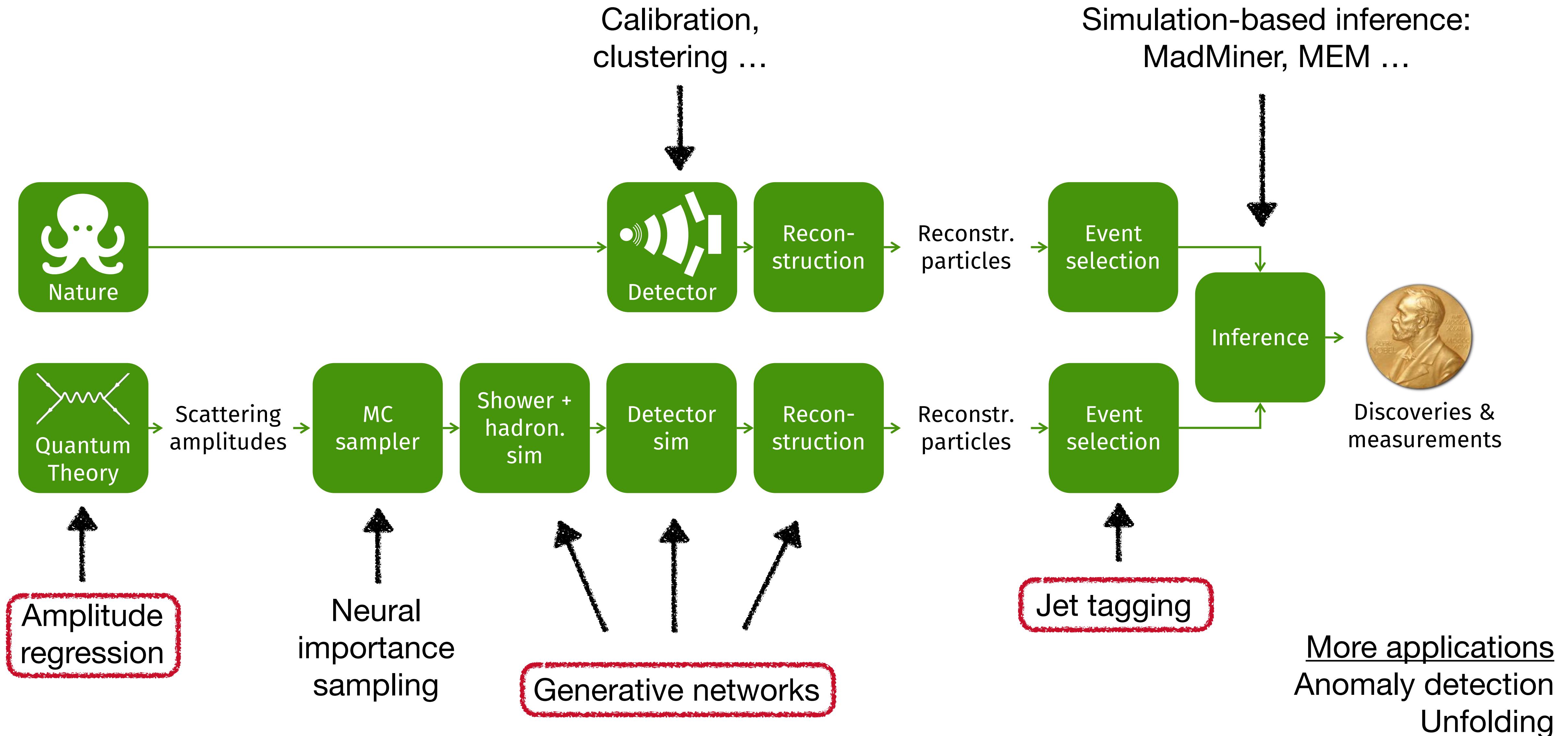
arXiv:2505.20280

in collaboration with Johann Brehmer, Víctor Bresó,
Luigi Favaro, Gerrit Gerhartz, Pim de Haan,
Fred A. Hamprecht, Peter Lippmann, Sebastian Pitz,
Tilman Plehn, Huilin Qu, Jesse Thaler

IPPP Seminar
16.07.2025



UNIVERSITÄT
HEIDELBERG
ZUKUNFT
SEIT 1386



Symmetries

... in LHC physics

Permutation
symmetry

$$q\bar{q} \rightarrow Zgggg$$

Lorentz
symmetry

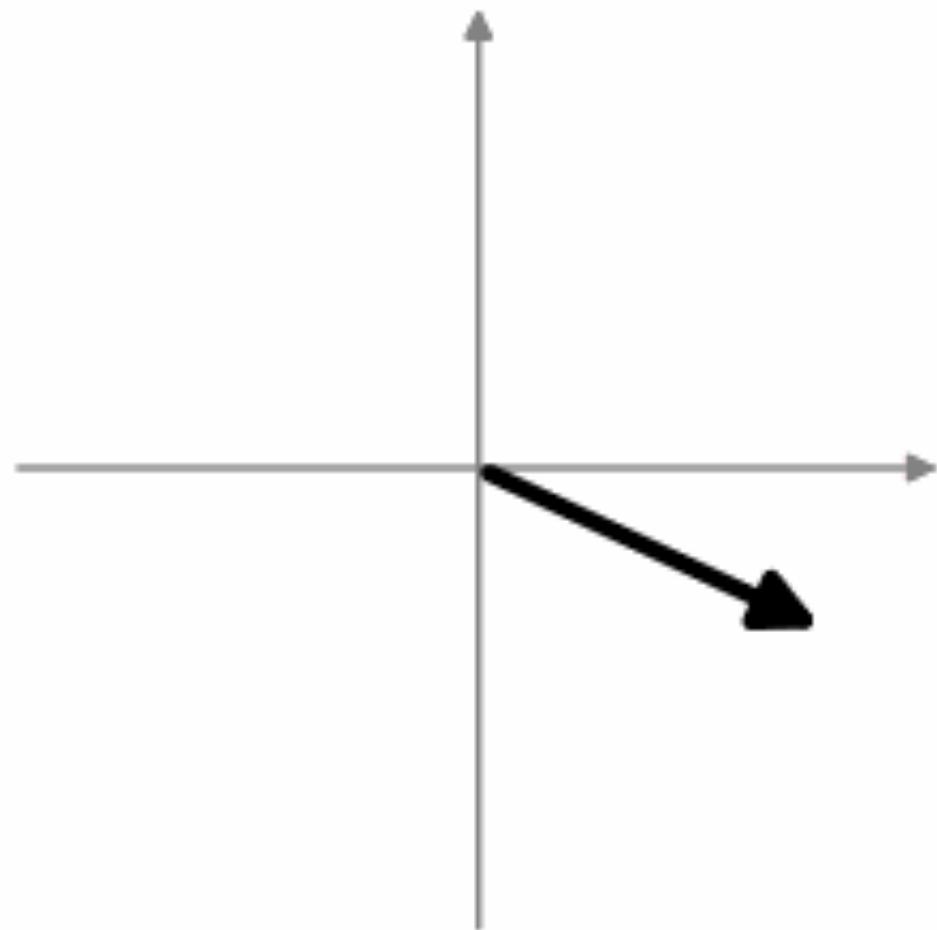
$$\begin{pmatrix} E \\ p_x \\ p_y \\ p_z \end{pmatrix}$$

Many more...

Symmetries

Invariance

Input



Output
(not invariant)

0.8



Output
(invariant)

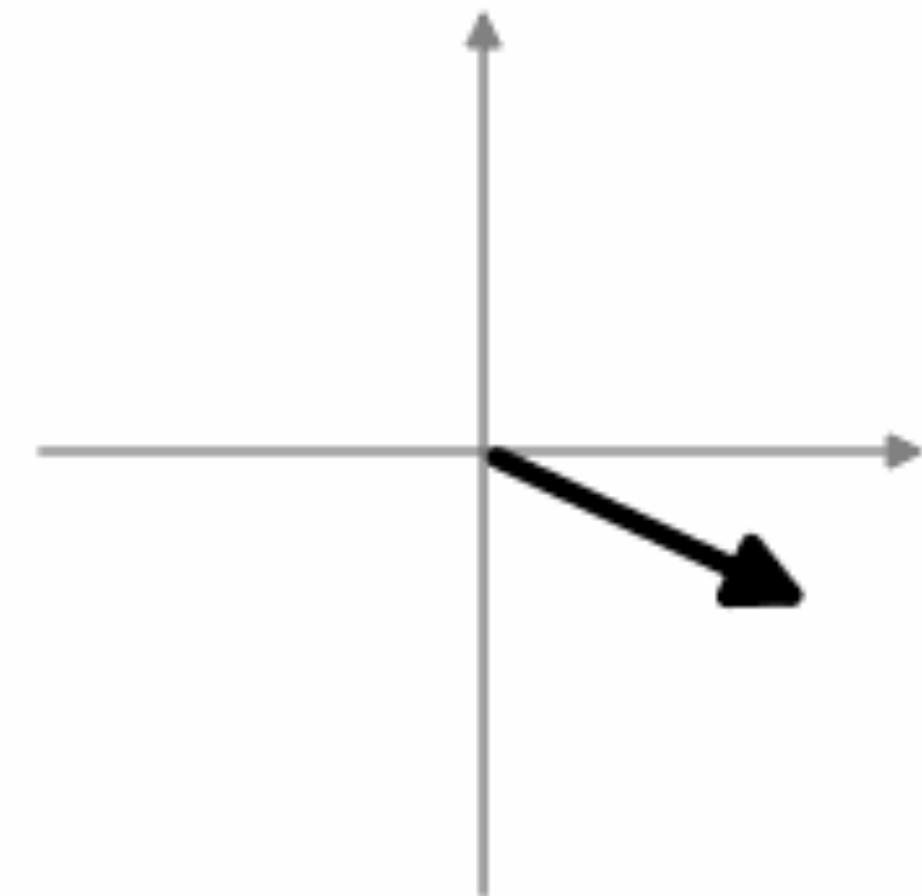
0.8



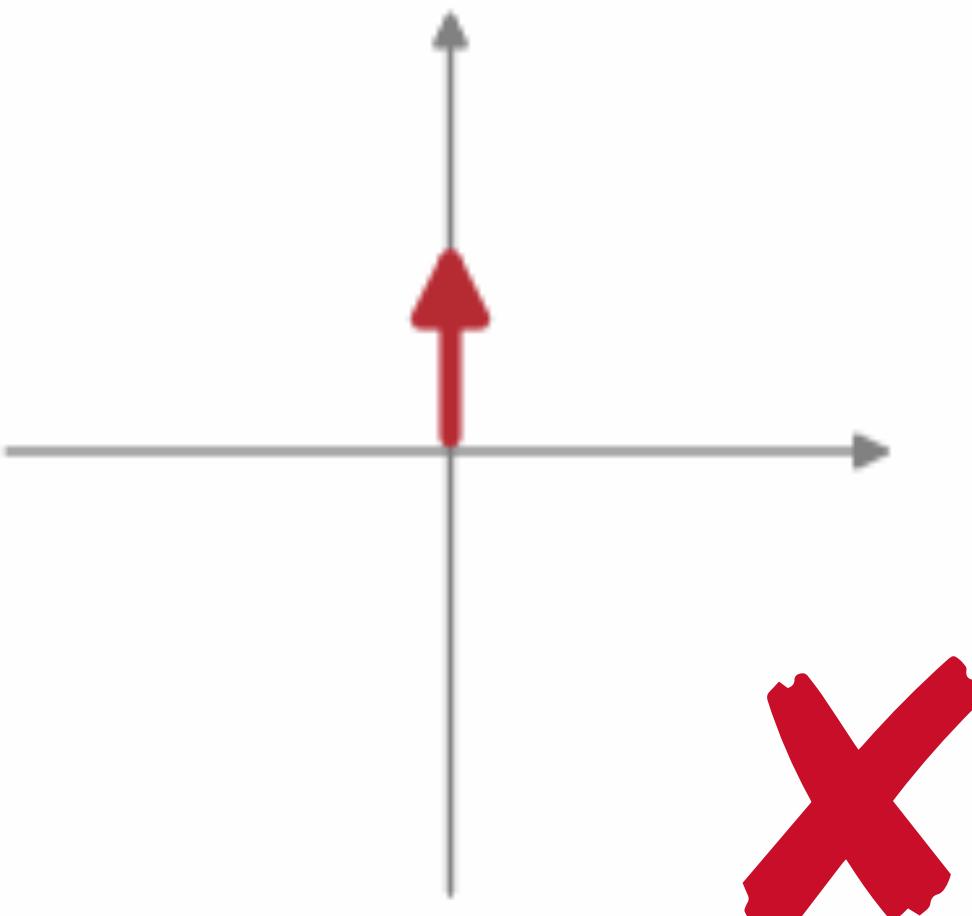
Symmetries

Equivariance = Covariance

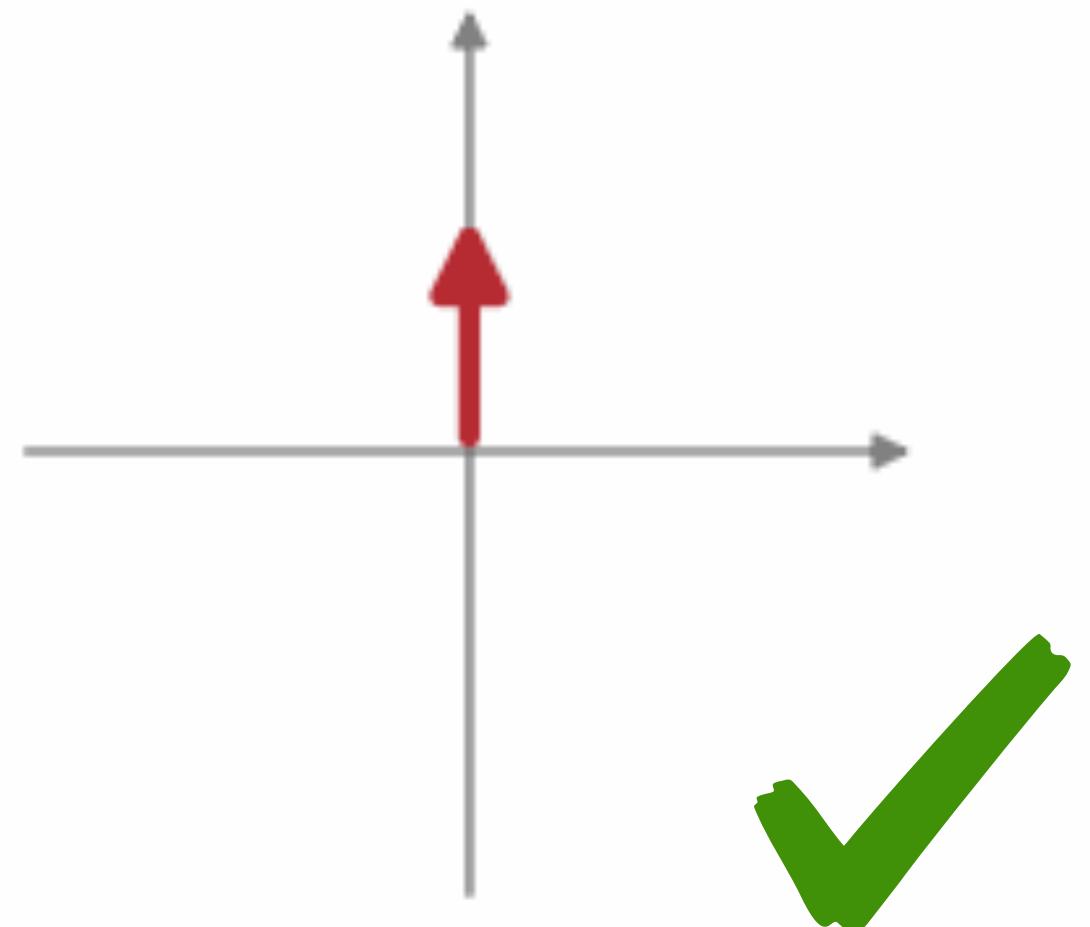
Input



Output
(not equivariant)



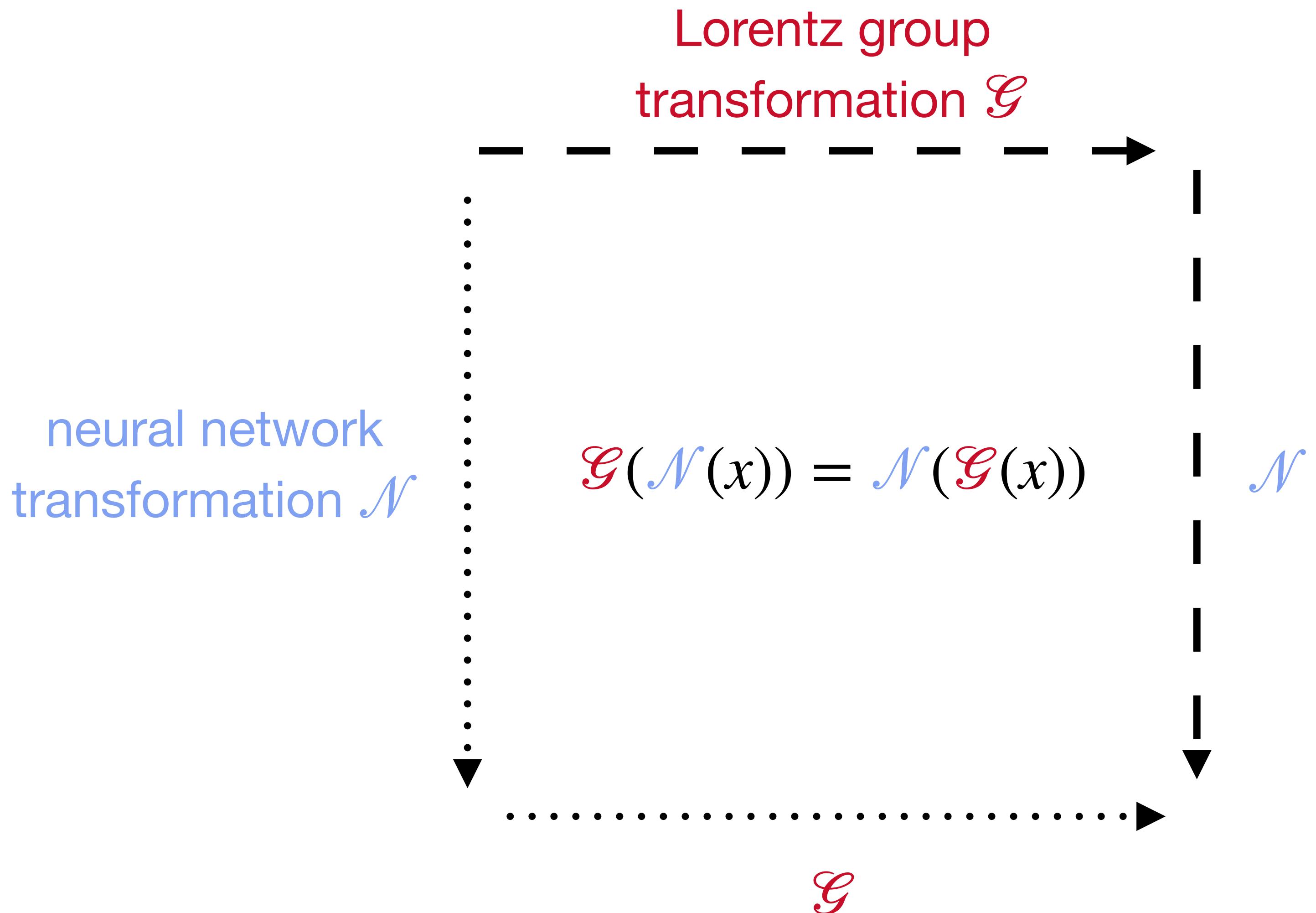
Output
(equivariant)



Invariance = Equivariance with scalar output

Symmetries

Equivariance - formalised



Symmetries

How to be equivariant

**Invariant network inputs
+ simple architecture
(easy)**

$$\sum_i p_i \quad p_i p_j$$

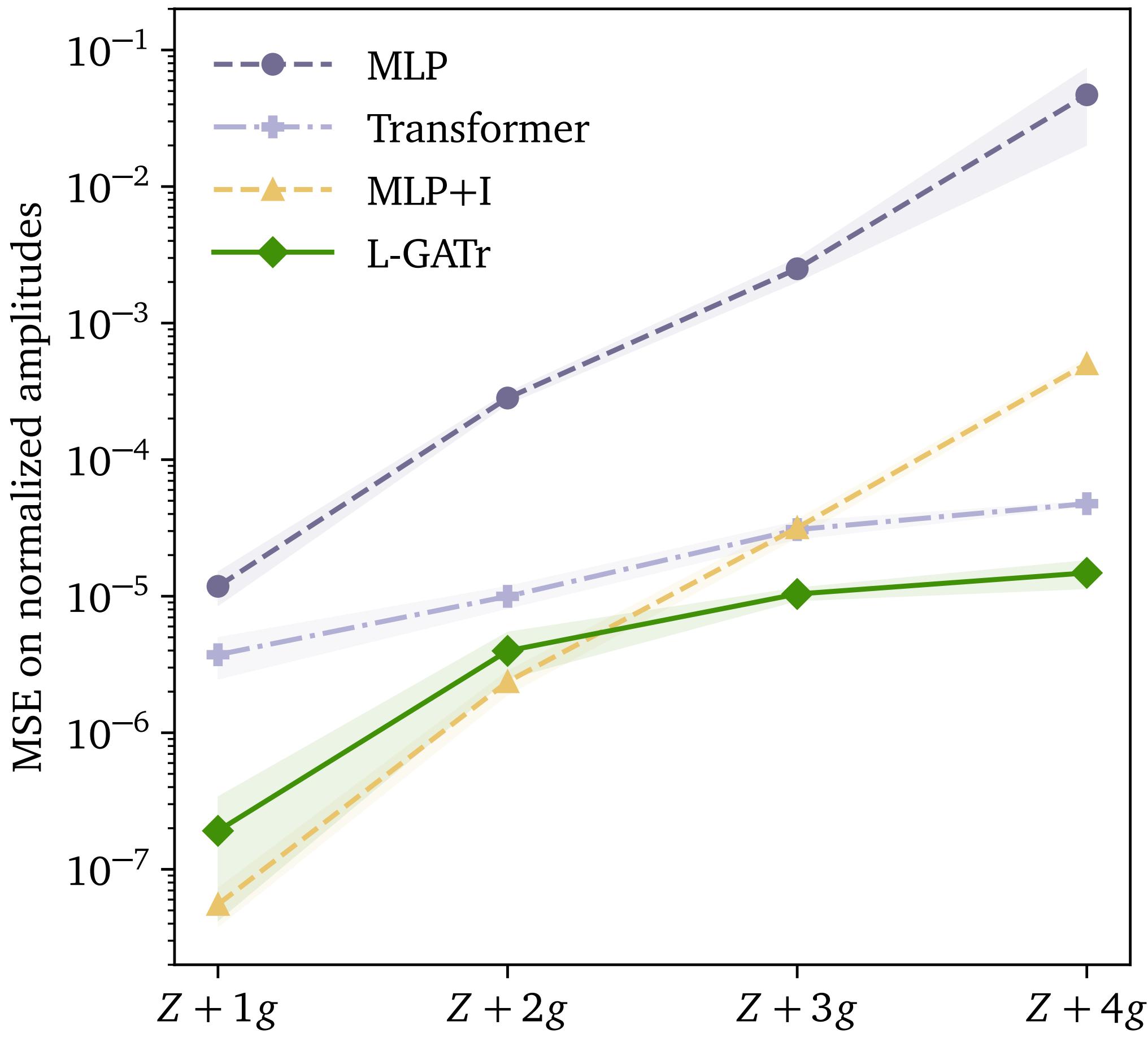
-  can only do invariance
-  doesn't work very well

**Internal representations
(requires new architecture)**

PELICAN
LorentzNet
L-GATr
LLoCa

Amplitude regression

Adding symmetry, step by step



Naive

$$\begin{pmatrix} E_1 \\ p_{x,1} \\ p_{y,1} \\ p_{z,1} \\ \text{PID}_1 \\ E_2 \\ p_{x,2} \\ p_{y,2} \\ p_{z,2} \\ \vdots \\ \text{PID}_2 \end{pmatrix}$$

MLP

Permutation symmetry

$$\begin{pmatrix} E_1 \\ p_{x,1} \\ p_{y,1} \\ p_{z,1} \\ \text{PID}_1 \\ E_2 \\ p_{x,2} \\ p_{y,2} \\ p_{z,2} \\ \vdots \\ \text{PID}_2 \end{pmatrix}$$

Transformer/
Graph Network

Lorentz invariants

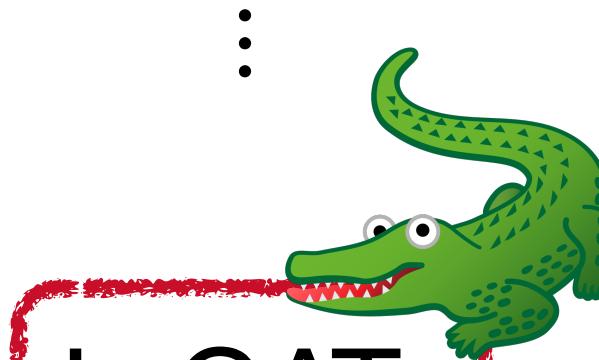
$$\begin{pmatrix} p_1^\mu p_{1,\mu} \\ p_1^\mu p_{2,\mu} \\ p_1^\mu p_{3,\mu} \\ \vdots \\ p_2^\mu p_{2,\mu} \\ p_2^\mu p_{3,\mu} \\ \vdots \end{pmatrix}$$

MLP+I

Permutation + Lorentz

$$\begin{pmatrix} p_1^\mu \\ \text{PID}_1 \end{pmatrix}$$

$$\begin{pmatrix} p_2^\mu \\ \text{PID}_1 \end{pmatrix}$$



Permutation equivariance

Why Transformers?



Attention is simple + scalable

$$x'_{ic} = \sum_j \text{Softmax}_j \left(\sum_{c'} \frac{q_{ic'} k_{ic'}}{\sqrt{d}} \right) v_{jc} = \sum_{j,c'} A_{ij} W_{cc'}^V x_{jc'}$$

$$q_{ic} = \sum_{c'} W_{cc'}^Q x_{ic'}$$

$$k_{ic} = \sum_{c'} W_{cc'}^K x_{ic'}$$

$$v_{ic} = \sum_{c'} W_{cc'}^V x_{ic'}$$



Everybody uses them



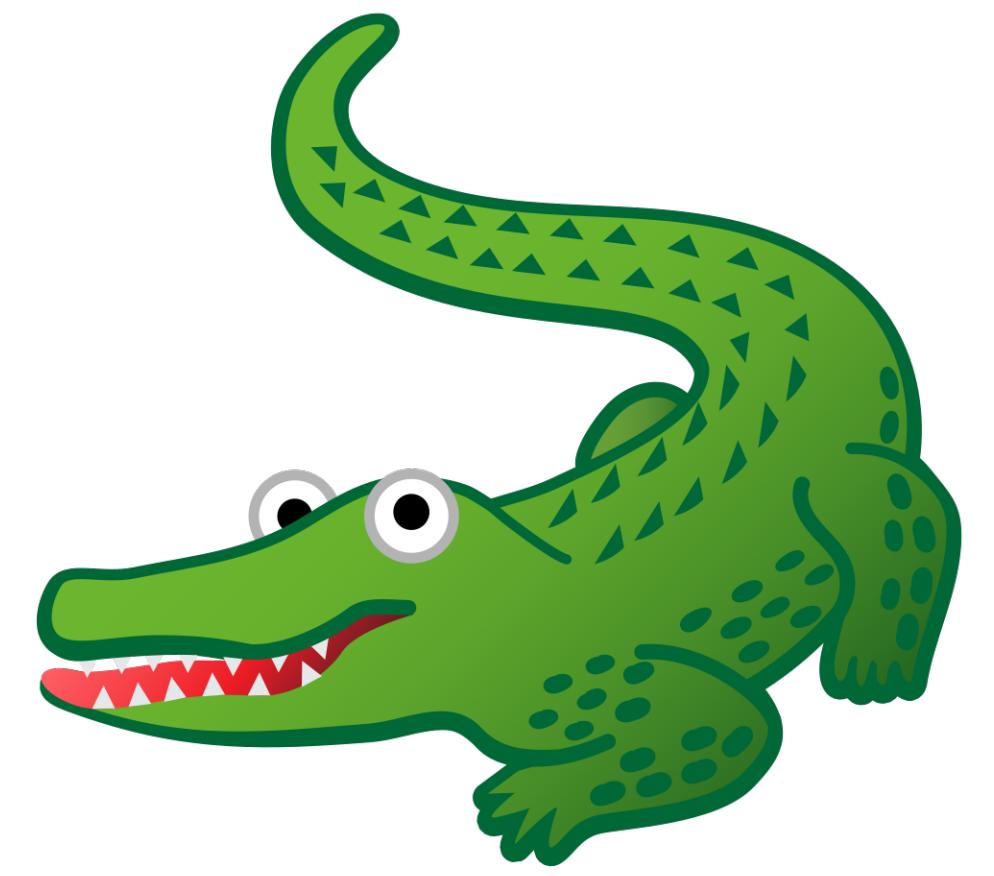
ChatGPT



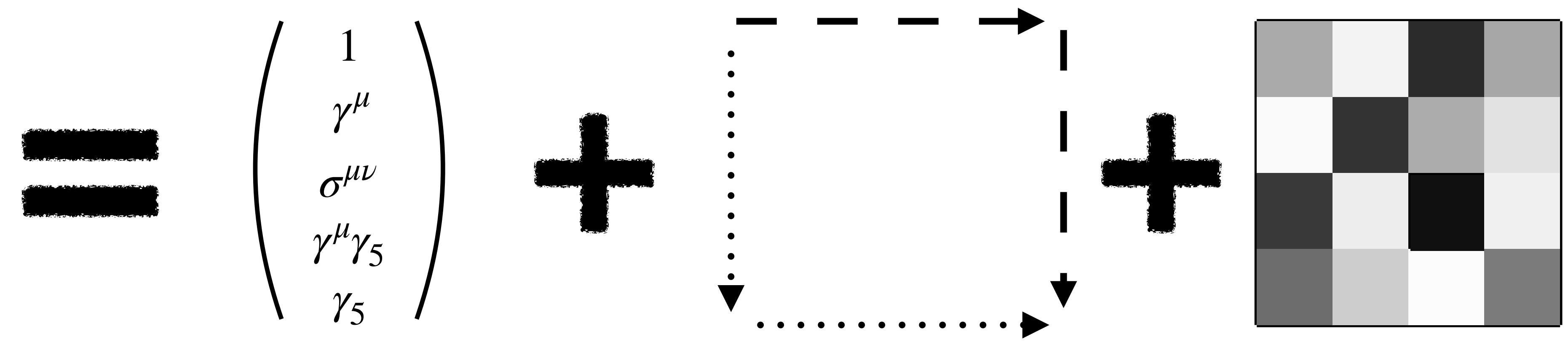
Midjourney



Attention is all you need .



Lorentz-Equivariant
Geometric **A**lgebra
Transformer



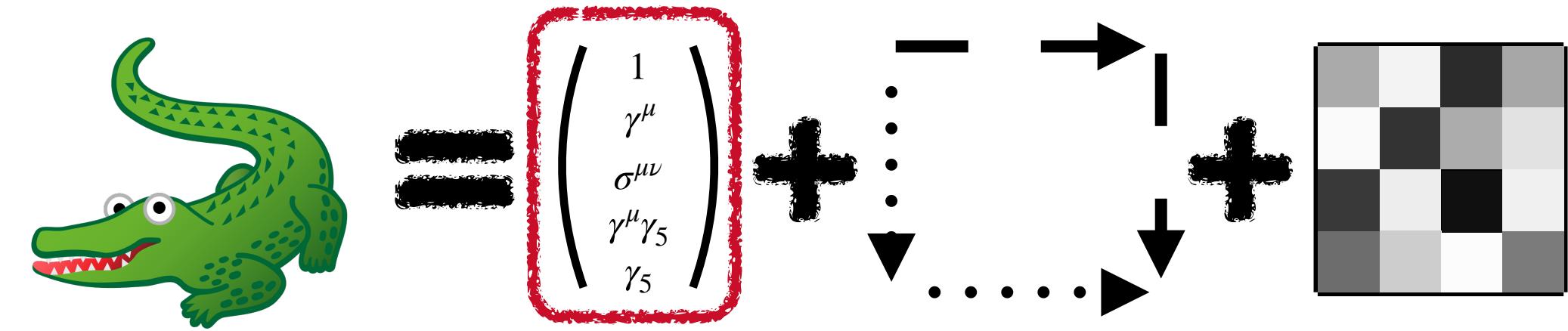
Geometric algebra
representations

Lorentz-Equivariant
layers

Transformer
architecture

GATr was originally
developed for E(3)
arXiv:2305.18415

L-GATr



Geometric algebra = Clifford algebra

Geometric algebra

= Vector space + geometric product $xy = \frac{\{x, y\}}{2} + \frac{[x, y]}{2}$

Spacetime geometric algebra

= Minkowski space + $\{\gamma^\mu, \gamma^\nu\} = 2g^{\mu\nu}\mathbf{1}$ over \mathbb{R}
≠ Dirac algebra (uses \mathbb{C})

L-GATr

Building the algebra

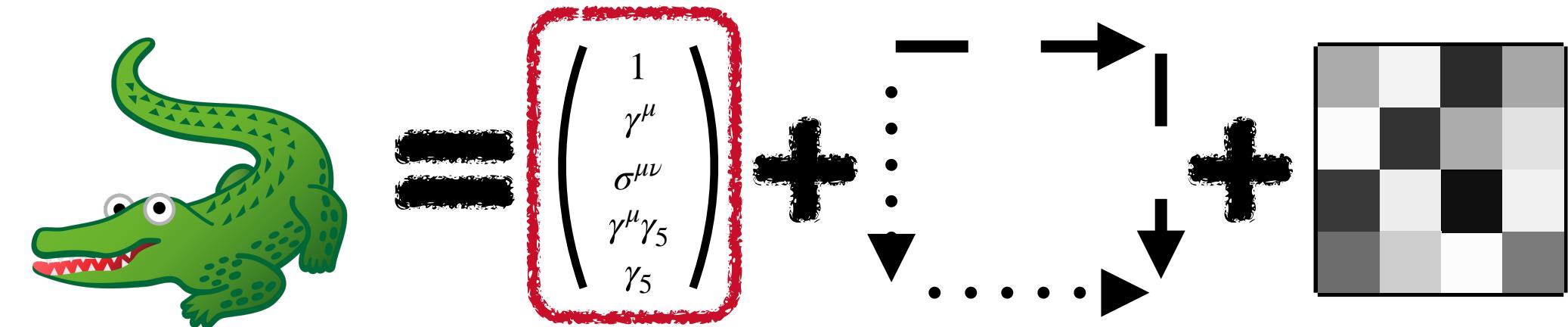
$$\{\gamma^\mu, \gamma^\nu\} = 2g^{\mu\nu}1$$

Linear combinations
+ Geometric product

$$x = x^S 1 + x_\mu^V \gamma^\mu + x_{\mu\nu}^B \sigma^{\mu\nu} + x_\mu^A \gamma^\mu \gamma^5 + x^P \gamma^5$$

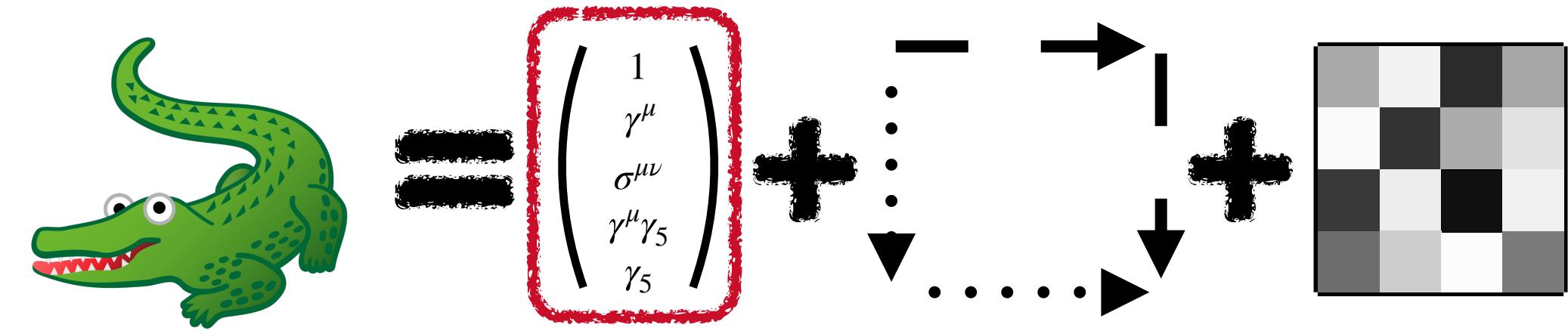
Multivector representation: $(x^S, x_\mu^V, x_{\mu\nu}^B, x_\mu^A, x^P) \in \mathbb{R}^{16}$

Particle: $x^V = (E, p_x, p_y, p_z)$, $x^S = \text{PID}$, $x^{\text{others}} = 0$

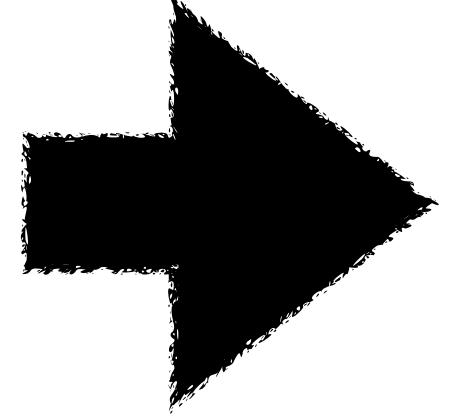


L-GATr

Geometric algebra representations



$$x^S \in \mathbb{R}$$



scalar channels

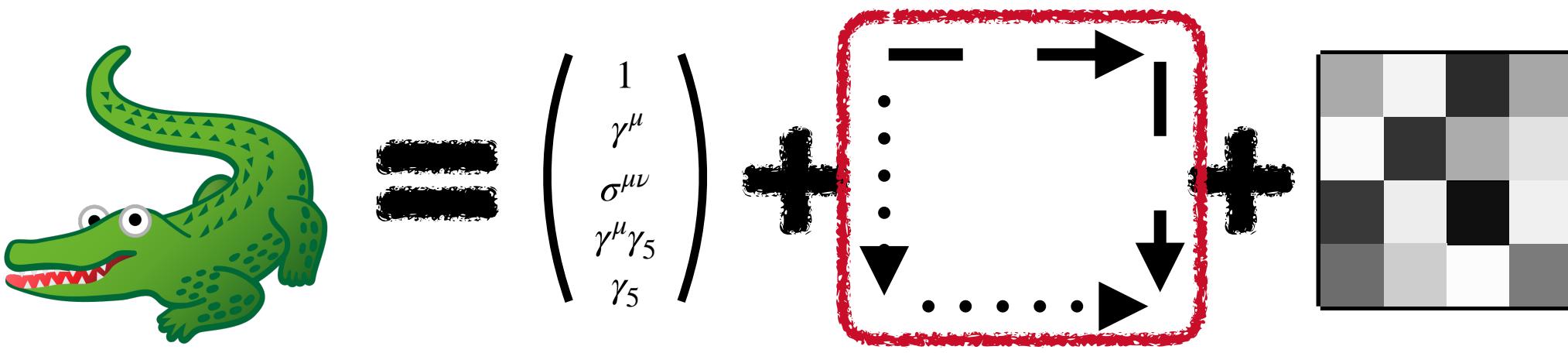
$$\begin{aligned} &x^S \\ &x_\mu^V \\ &x_{\mu\nu}^B \\ &x_\mu^A \\ &x_\nu^P \end{aligned}$$

$$\in \mathbb{R}^{16}$$

multivector channels

L-GATr

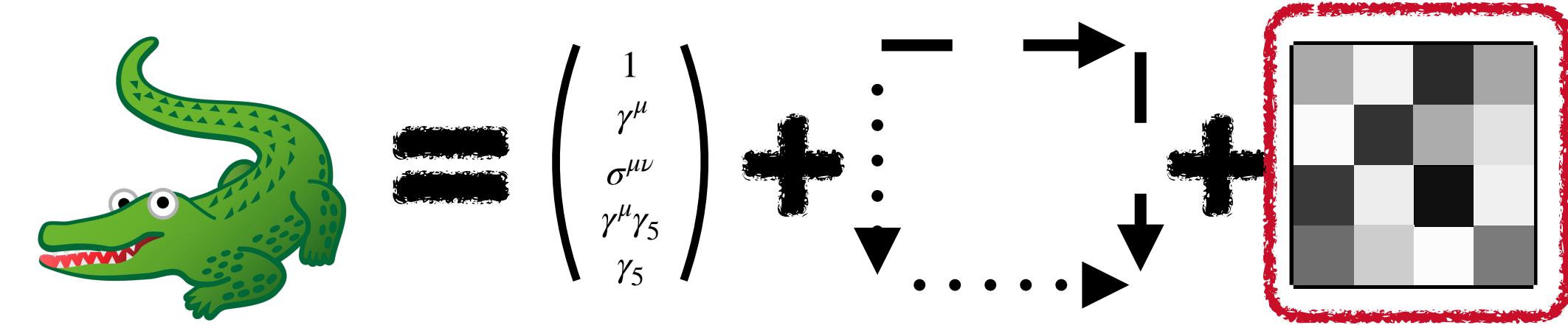
GATr-ing all transformer layers



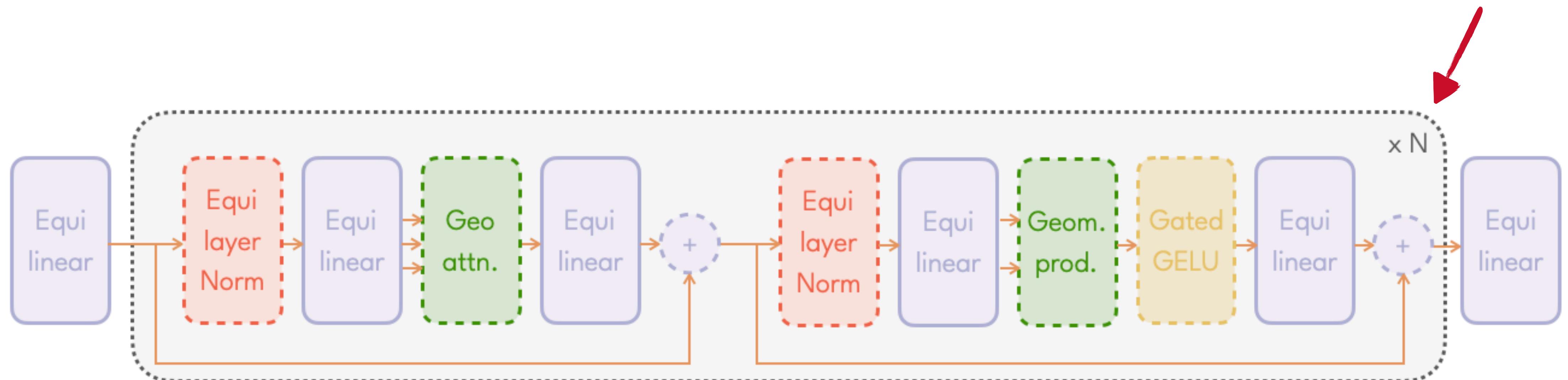
Layer type	Transformer	L-GATr
Linear(x)	$vx + w$	$\sum_{k=0}^4 v_k \langle x \rangle_k \left(+ \sum_{k=0}^4 w_k \gamma^5 \langle x \rangle_k \right)$
Attention(q, k, v) _{ic}	$\sum_{j=1}^{n_t} \text{Softmax}_j \left(\sum_{c'=1}^{n_c} \frac{q_{ic'} k_{jc'}}{\sqrt{n_c}} \right) v_{jc}$	$\sum_{j=1}^{n_t} \text{Softmax}_j \left(\sum_{c'=1}^{n_c} \frac{\langle q_{ic'}, k_{jc'} \rangle}{\sqrt{16n_c}} \right) v_{jc}$
LayerNorm(x)	$x \left[\frac{1}{n_c} \sum_{c=1}^{n_c} x_c^2 + \epsilon \right]^{-1/2}$	$x \left[\frac{1}{n_c} \sum_{c=1}^{n_c} \sum_{k=0}^4 \left \langle \langle x_c \rangle_k, \langle x_c \rangle_k \rangle \right + \epsilon \right]^{-1/2}$
Activation(x)	GELU(x)	GELU($\langle x \rangle_0$) x
GP(x, y)	—	xy

L-GATr

Full architecture



GATr blocks
can be stacked
to large depth

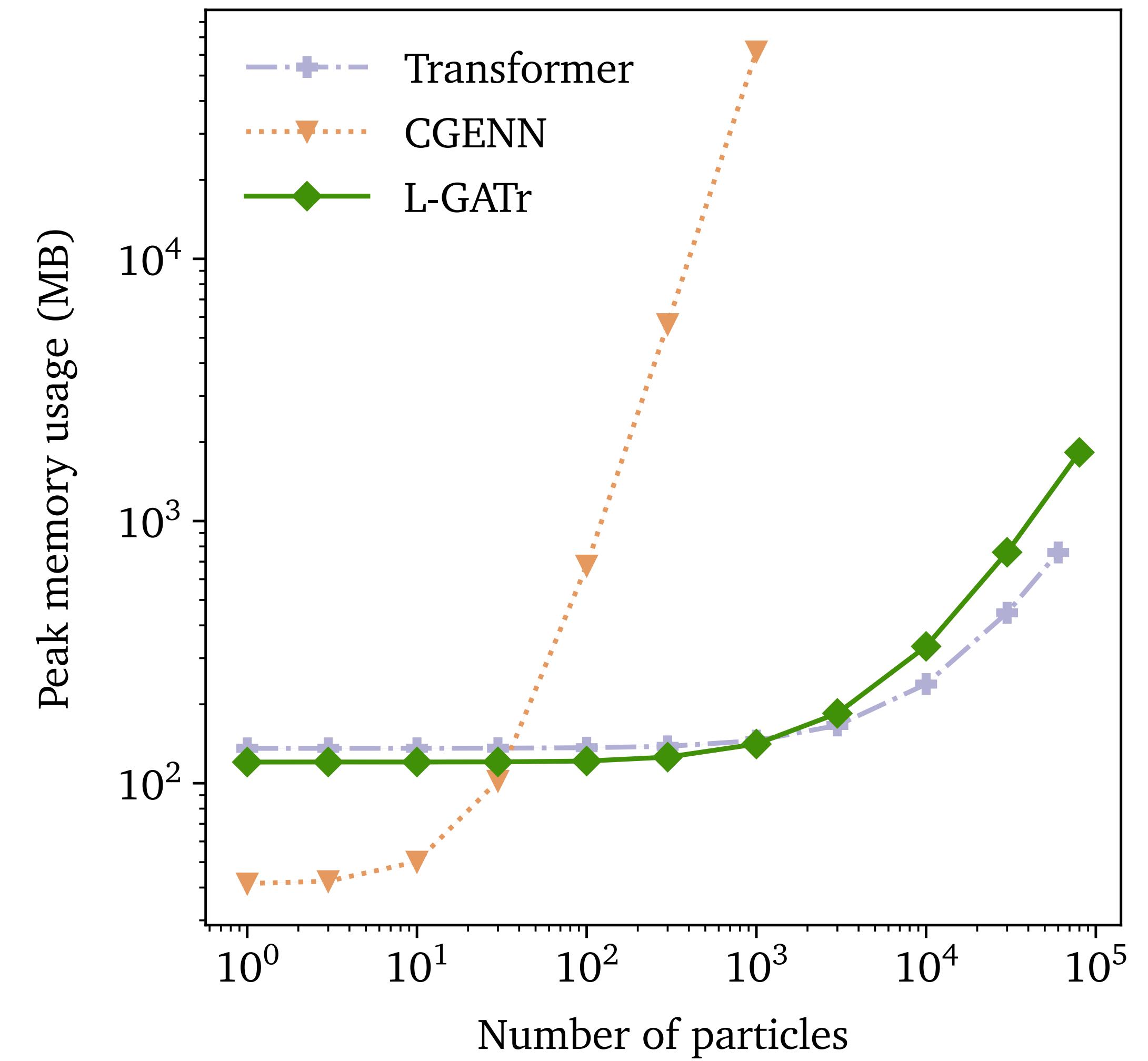
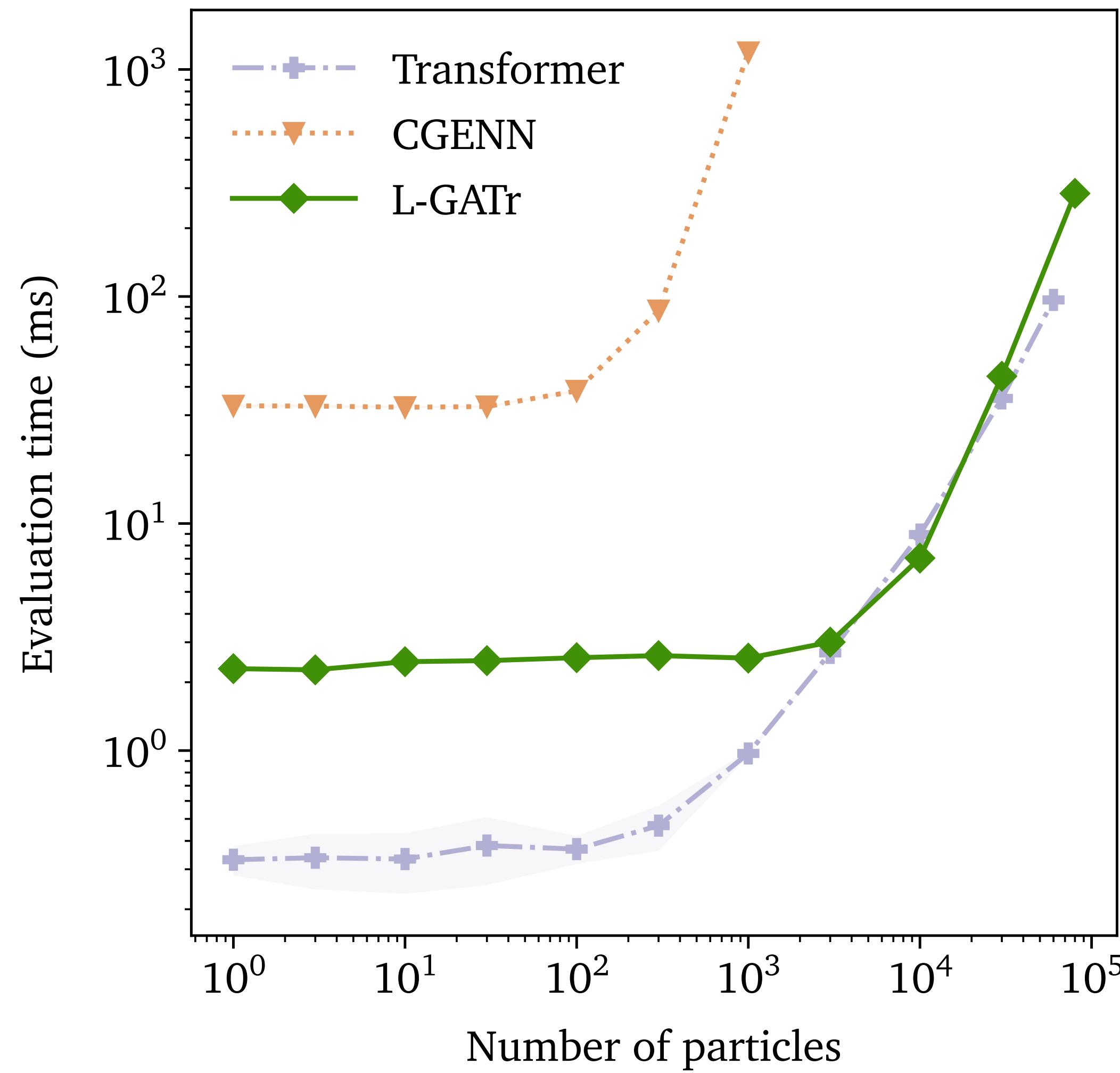
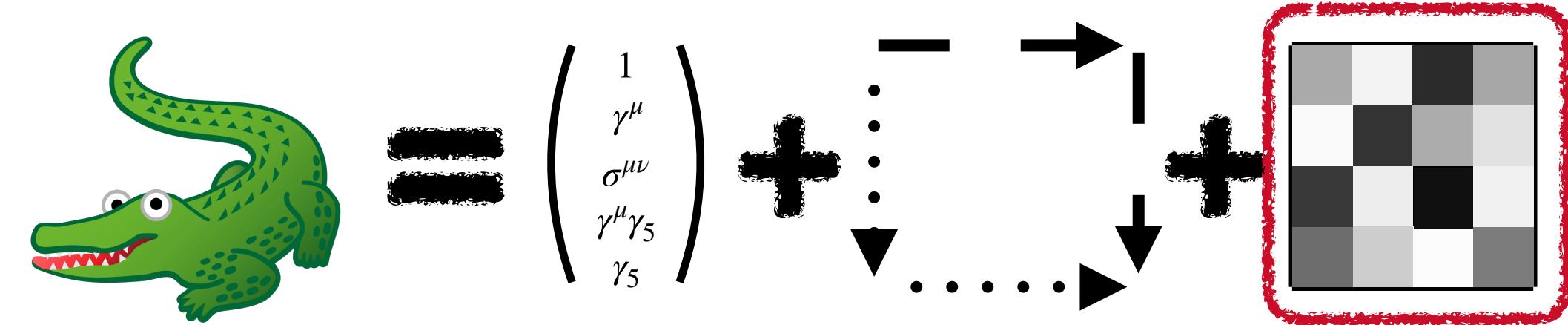


Equivariant Self-Attention

Equivariant MLP

L-GATr

Transformers scale better than graphs



USAGE

[Quickstart](#)[Spacetime Geometric Algebra](#)[Attention Backends](#)[Lorentz Symmetry Breaking](#)

REFERENCE

[API Reference](#)

Quickstart

This page sets you up for building and running L-GATr models.

Installation

Before using the package, install it via pip:

```
pip install lgatr
```

Alternatively, if you're developing locally:

```
git clone https://github.com/heidelberg-hepml/lgatr.git
cd lgatr
pip install -e .
```

Building L-GATr

You can construct a simple `LGATr` model as follows:

L-GATr is on pypi!

```
attention = dict(num_heads=2)
mlp = dict()
```

```
from lgatr import LGATr
attention = dict(num_heads=2)
mlp = dict()
lgatr = LGATr(
    in_mv_channels=1,
    out_mv_channels=1,
    hidden_mv_channels=8,
    in_s_channels=0,
    out_s_channels=0,
    hidden_s_channels=16,
    attention=attention,
    mlp=mlp,
    num_blocks=2,
)
```

```
from lgatr.interface import embed_vector, extract_scalar
multivector = embed_vector(p)
print(multivector.shape) # torch.Size([128, 20, 1, 16])
```

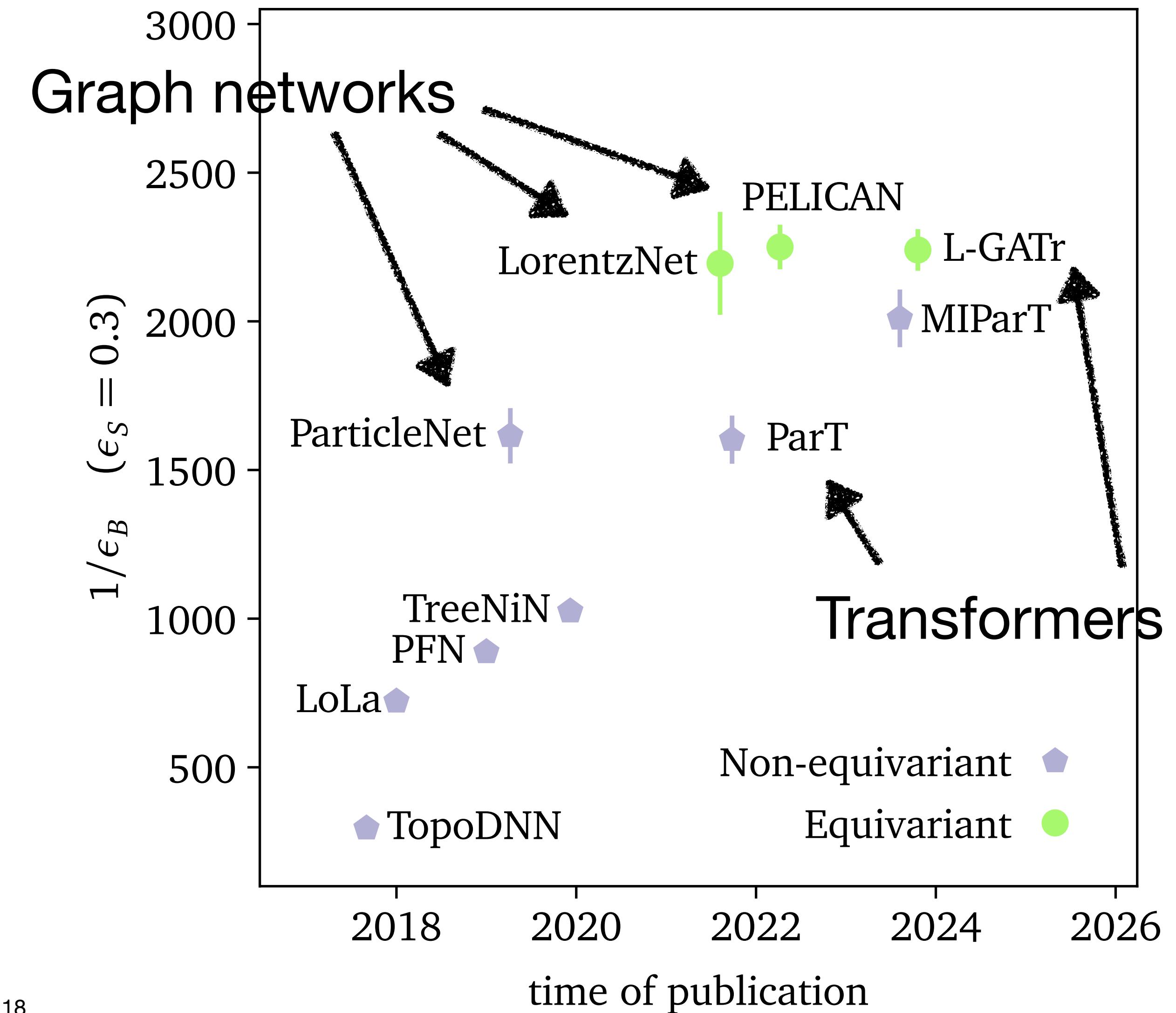
```
output_mv, output_s = lgatr(multivectors=multivector, scalars=None)
out = extract_scalar(output_mv)
print(out.shape) # torch.Size([128, 20, 1, 1])
```

Jet Tagging

The history of top tagging

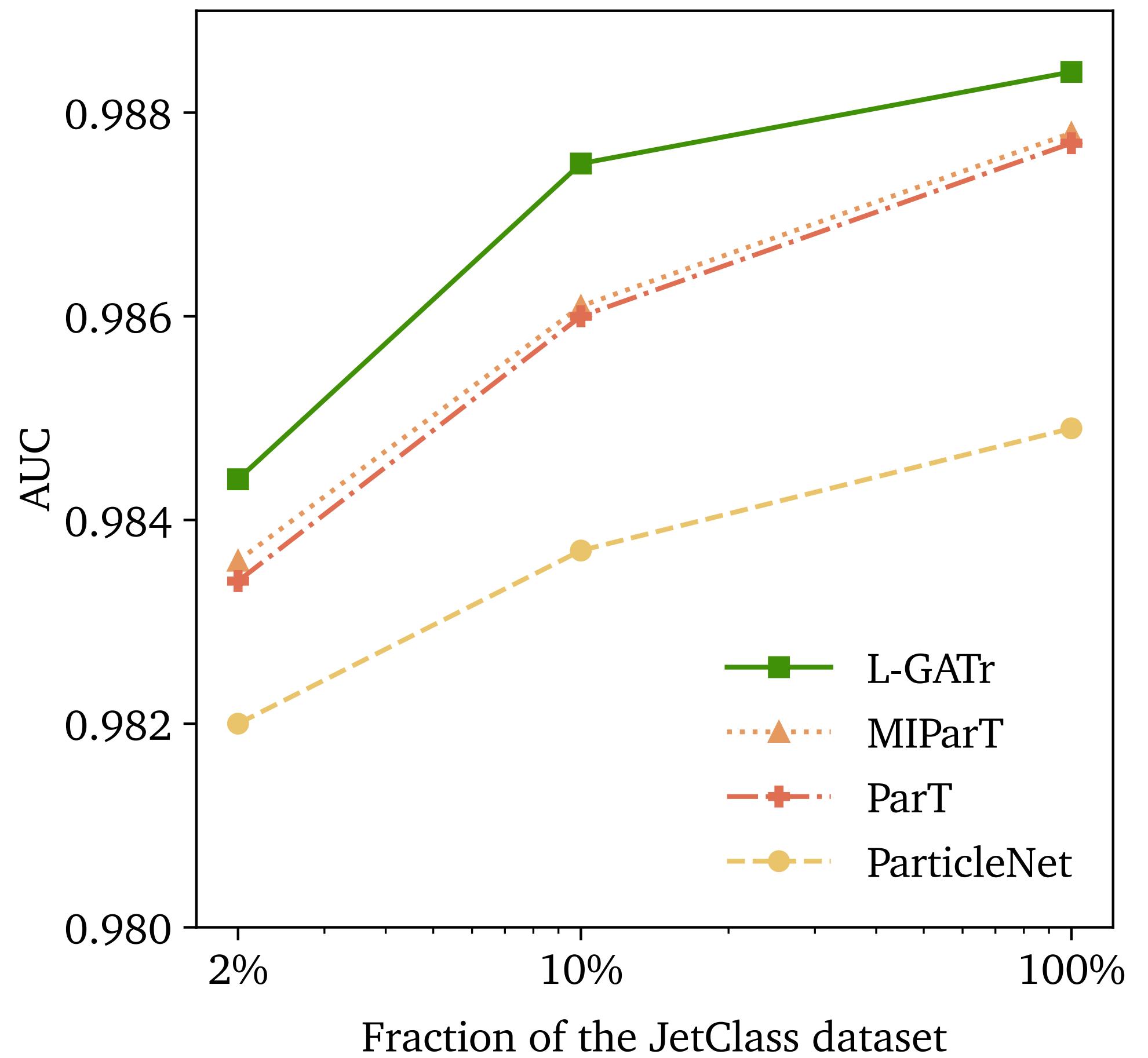
All top-performing taggers
are Lorentz-equivariant

Graph networks and transformers
perform similarly well



Jet Tagging

Training on 100M jets

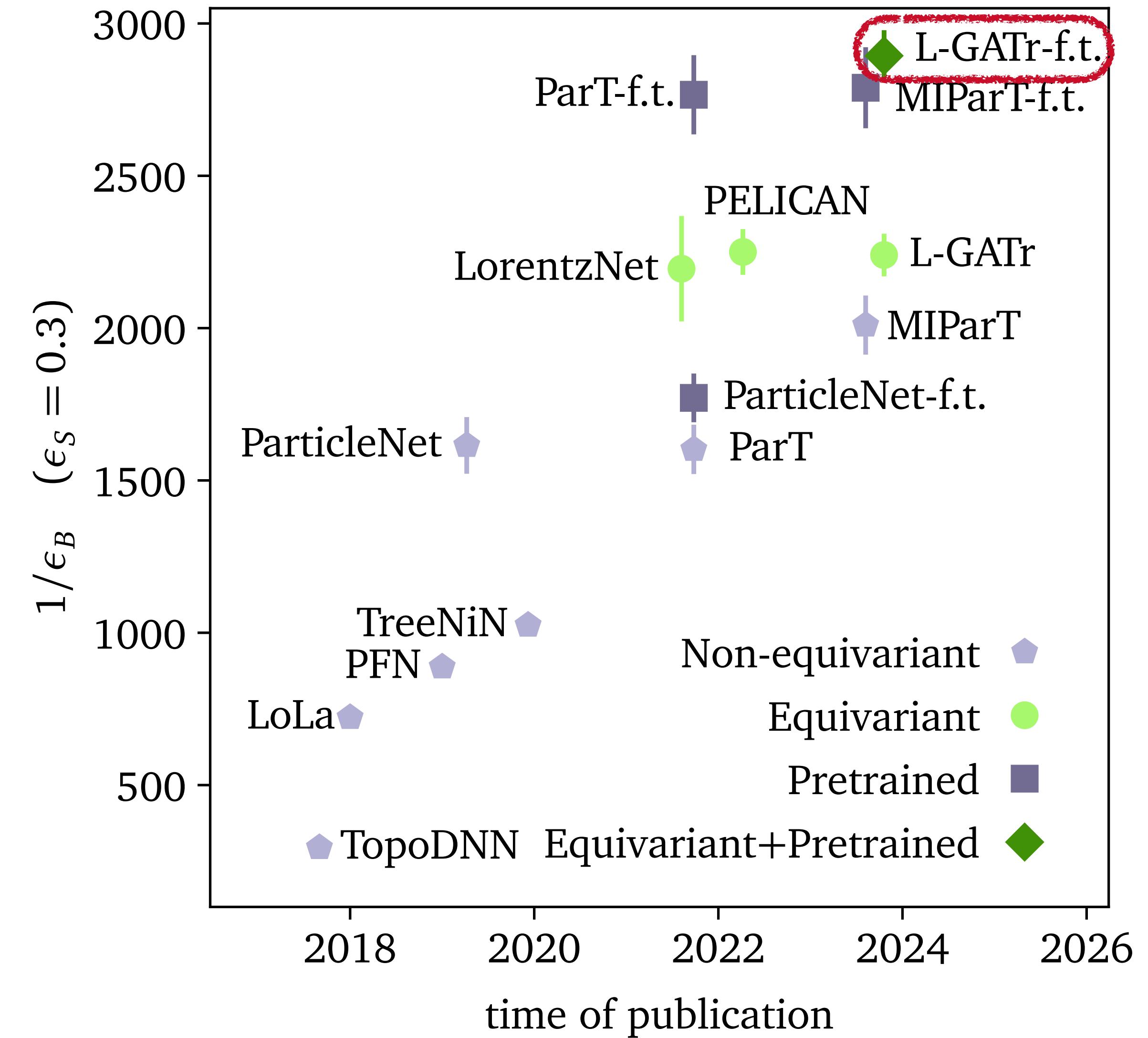


Transformers outscale
graph networks on
large datasets

Jet Tagging

Transfer learning for top tagging

L-GATr-f.t. combines the benefits of pretraining and Lorentz equivariance



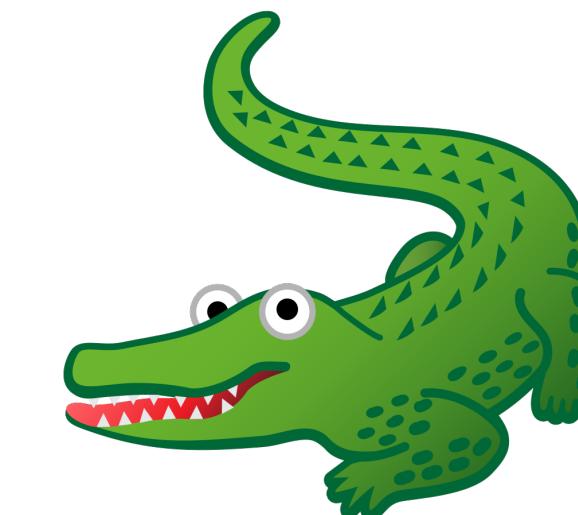
The landscape of Lorentz-equivariant networks

Specialized layers

PELICAN

LorentzNet

CGENN



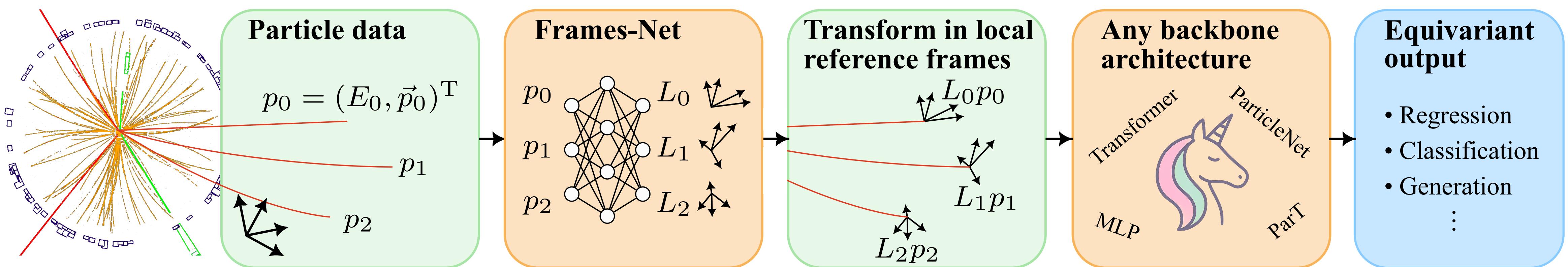
L-GATr

How good is a
Lorentz-equivariant ParT?

Can we ‘upgrade’ a given architecture
to make it Lorentz-equivariant?

⇒ LLoCa

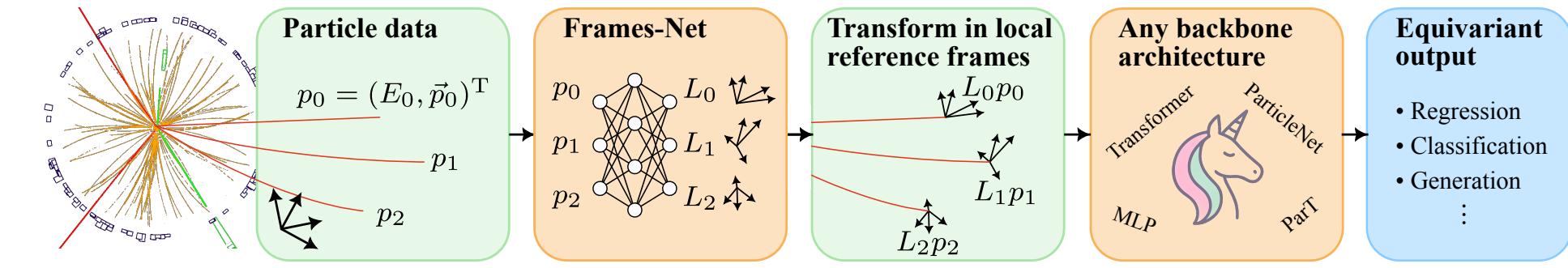
How to make any network Lorentz-equivariant:



Lorentz Local Canonicalization

LLoCa

Local frames



Features in local frame should be **invariant**:

$$x_L = Lx \xrightarrow{\Lambda} L'x' = L'\Lambda x \stackrel{!}{=} x_L \quad \Rightarrow \quad L' = L\Lambda^{-1}$$

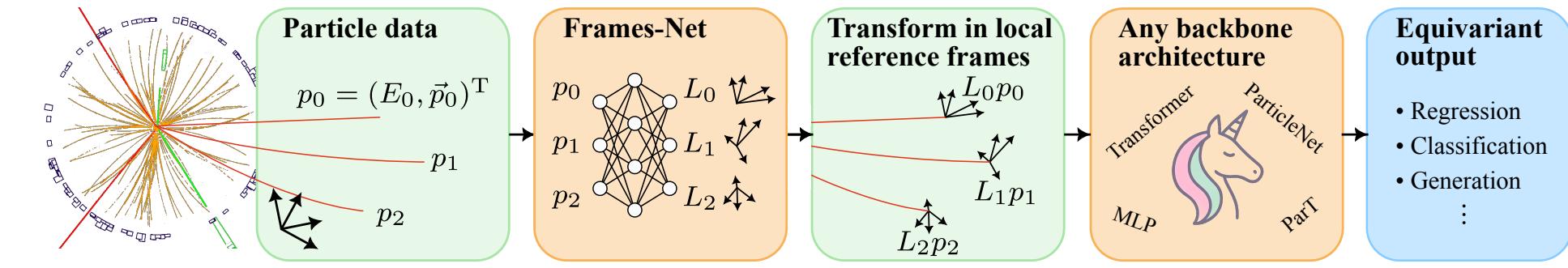
Achieve this with **Lorentz vectors** u_a^μ , $a = 0,1,2,3$:

$$L = \begin{pmatrix} u_0^T g \\ u_1^T g \\ u_2^T g \\ u_3^T g \end{pmatrix} \xrightarrow{\Lambda} L' = \begin{pmatrix} u_0^T \Lambda^T g \\ u_1^T \Lambda^T g \\ u_2^T \Lambda^T g \\ u_3^T \Lambda^T g \end{pmatrix} = L\Lambda^{-1}$$

$$\sum_{\mu,\nu} u_a^\mu u_b^\nu g_{\mu\nu} = g_{ab}$$

LLoCa

Equivariance proof



Latent features in the local frame are **invariant**:

$$\begin{aligned} x' &= \Lambda x \\ f' &= \rho(\Lambda)f \\ \rho(\Lambda_1\Lambda_2) &= \rho(\Lambda_1)\rho(\Lambda_2) \end{aligned}$$

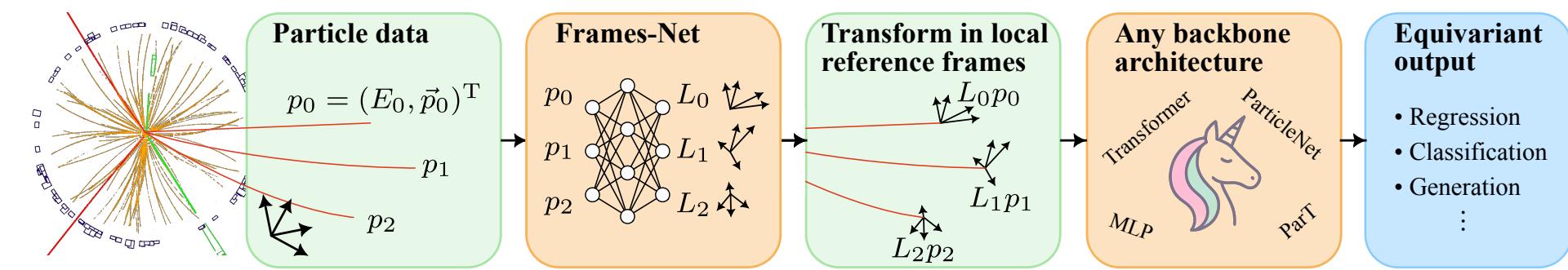
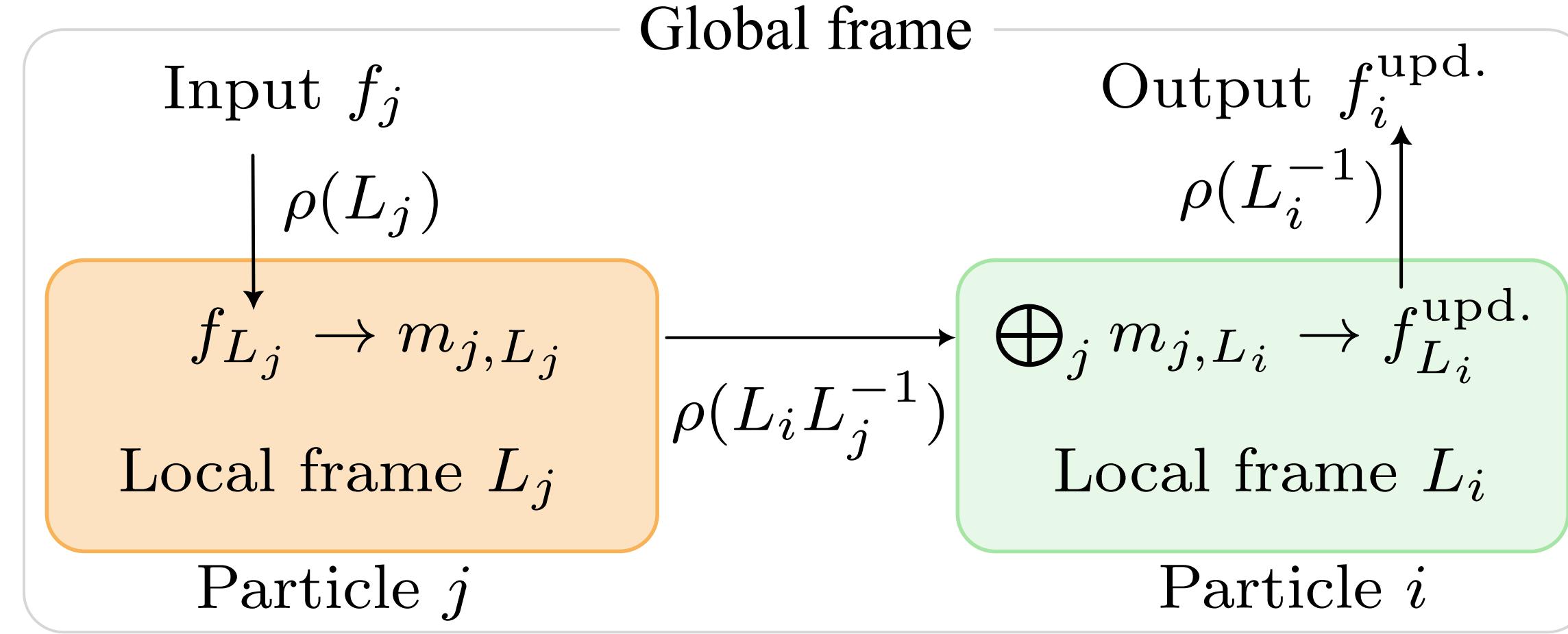
$$f_L \xrightarrow{\Lambda} f'_L = \rho(L')f' = \rho(L\Lambda^{-1})\rho(\Lambda)f = \rho(L\Lambda^{-1}\Lambda)f = \rho(L)f = f_L$$

Output features in the global frame $y = \rho(L^{-1})f_L$ are **equivariant**

$$y \xrightarrow{\Lambda} y' = \rho(L'^{-1})f'_L = \rho(\Lambda L^{-1})f_L = \rho(\Lambda)\rho(L^{-1})f_L = \rho(\Lambda)y$$

LLoCa

Tensorial message passing



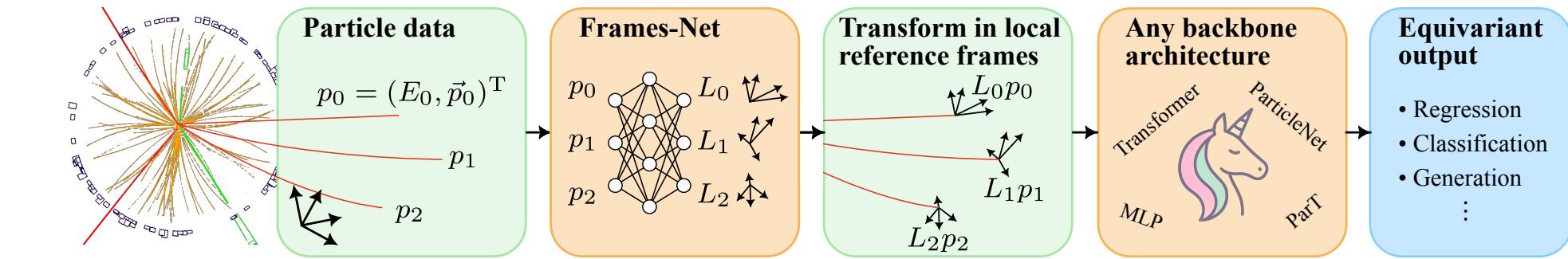
Non-trivial frame-to-frame transformations in backbone improve performance

But what representations should the latent space have?

LLoCa

Which internal representations?

Across our experiments,
a equal mix of scalar and vector
representations works best.



Method	MSE ($\times 10^{-6}$)
Non-equivariant	11.9 ± 0.8
Global canonical.	5.9 ± 0.4
LLoCa (16 scalars)	54.0 ± 3
LLoCa (single 2-tensor)	2.4 ± 0.3
LLoCa (4 vectors)	2.0 ± 0.2
LLoCa (8 scalars, 2 vectors)	1.5 ± 0.1

LLoCa with scalar representations
Is worse than non-equivariant!

LLoCa

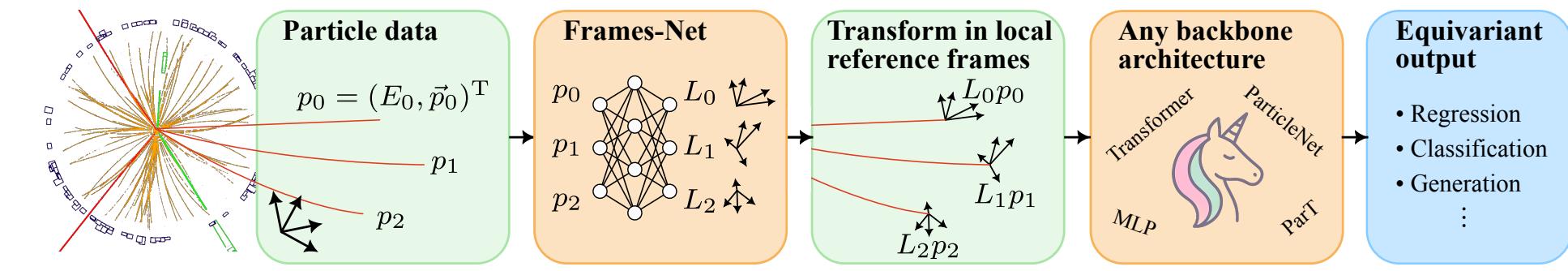
How to predict local frames

1) Predict vectors with simple **equivariant architecture**

$$v_{i,k} = \sum_{j=1}^N \text{softmax}\left(\varphi_k(s_i, s_j, \langle p_i, p_j \rangle)\right)(p_i + p_j) \quad \text{for } k = 0, 1, 2.$$

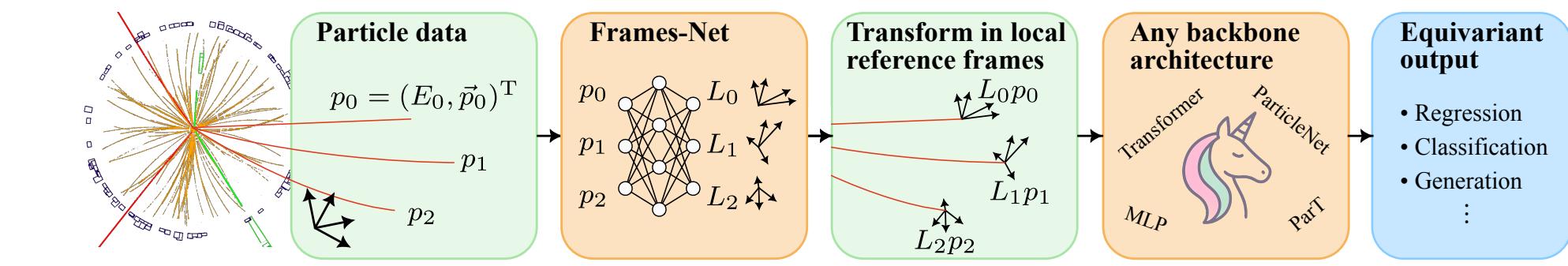
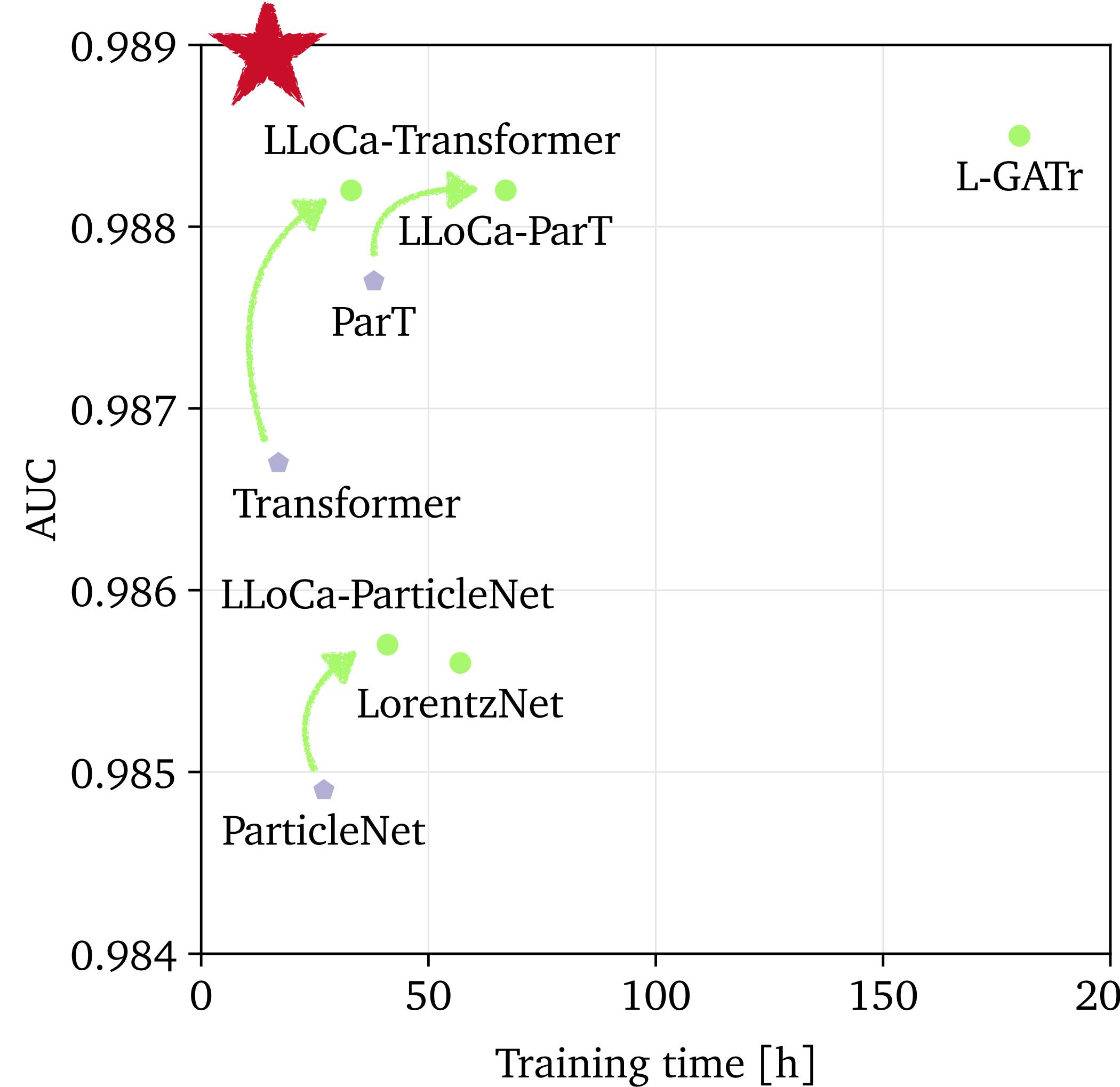
↑
MLP

2) Orthogonalize vectors with **Gram-Schmidt algorithm**



Jet Tagging

Speed vs performance

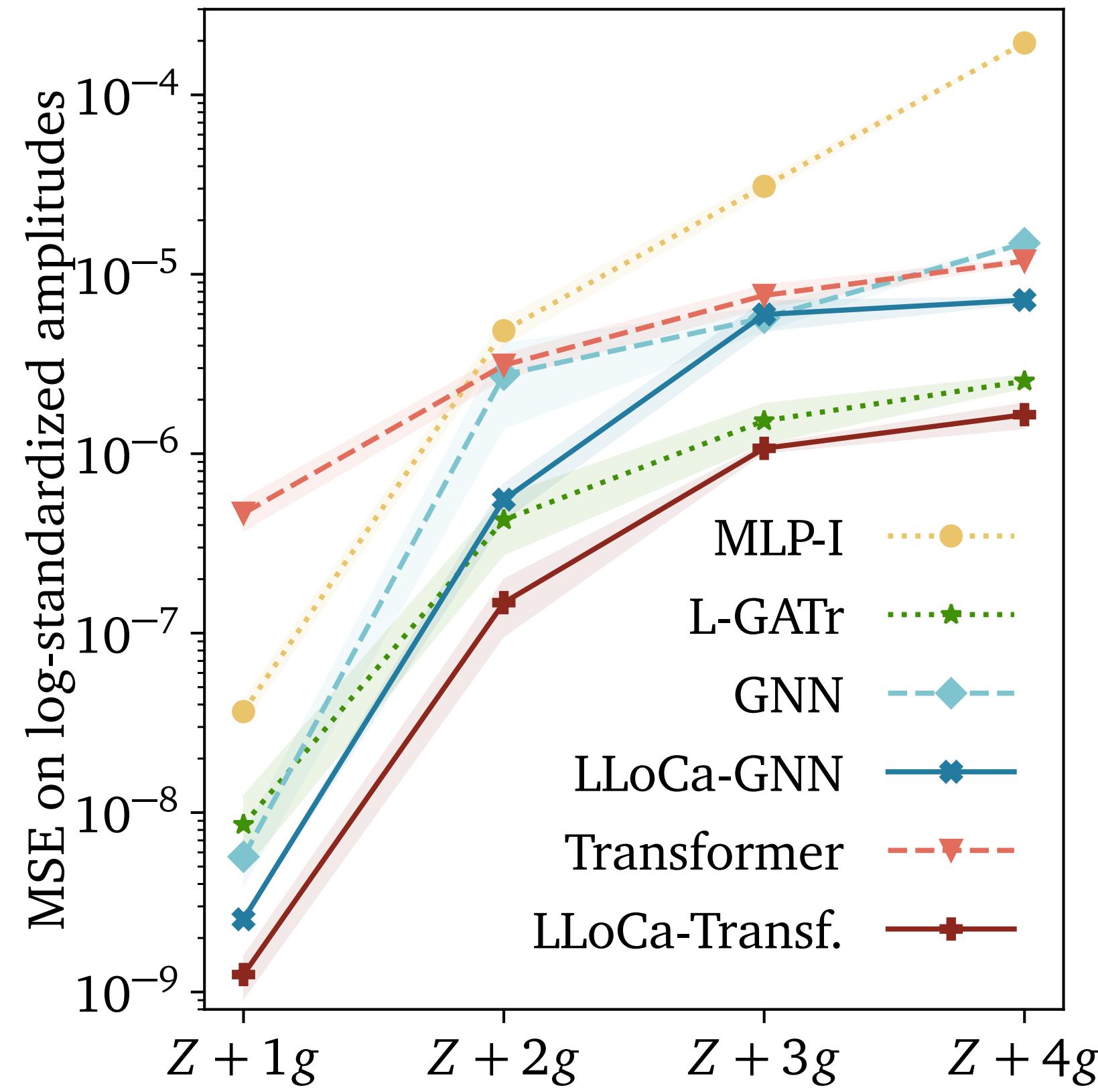


LLoCa networks improve performance at the cost of slower training

LLoCa-ParT is not better than a vanilla LLoCa-Transformer

Amplitude regression

Speed vs performance



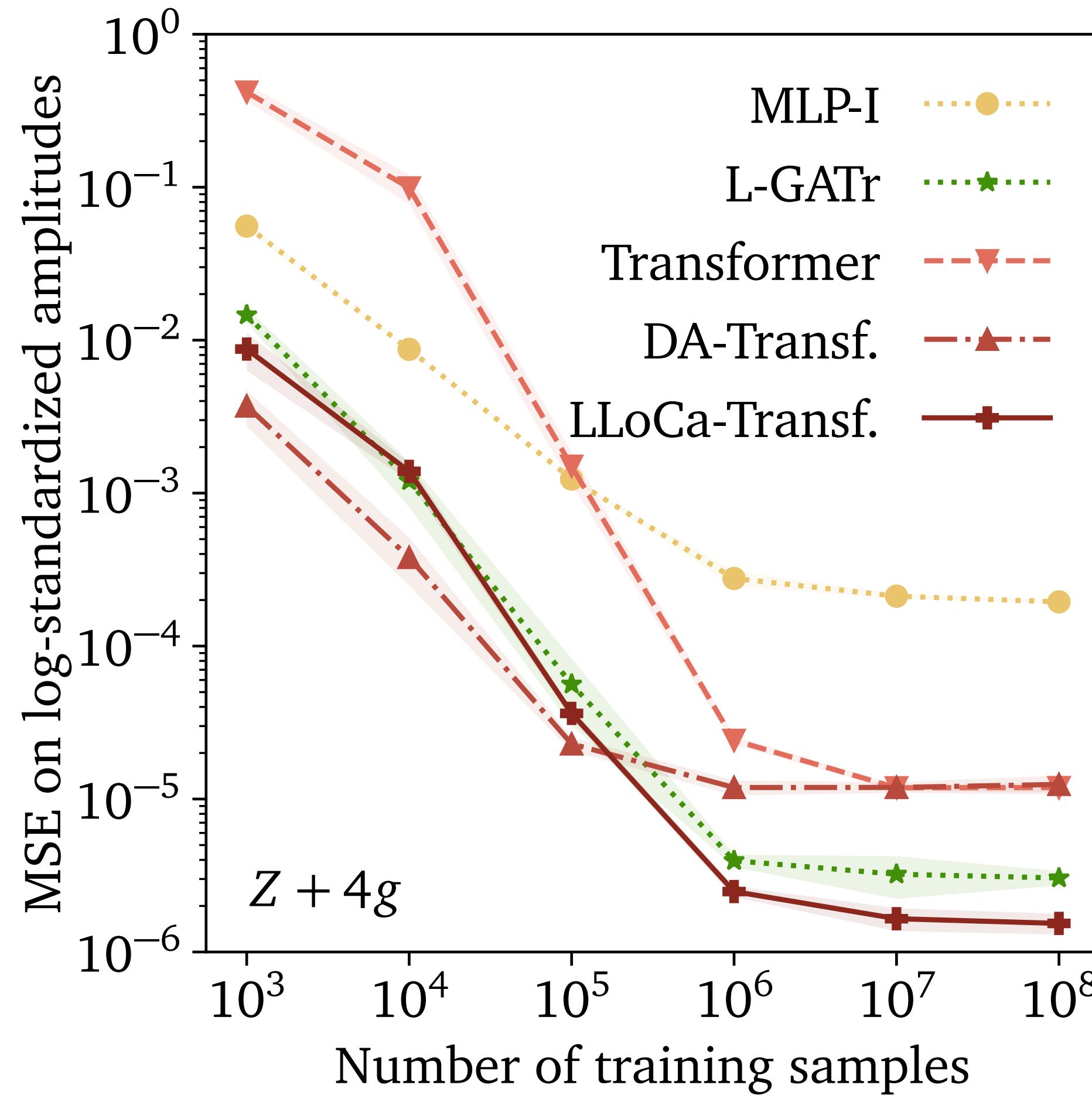
LLoCa-Transformer and L-GATr
achieve similar performance

Model	MSE $\times 10^{-6}$ (\downarrow)	FLOPs	Time
MLP-I [38]	195.0 \pm 2	0.1M	0.6h
L-GATr [38]	2.5 \pm 0.2	1160.0M	19.5h
GNN	14.9 \pm 0.3	9.9M	1.3h
LLoCa-GNN	7.2 \pm 0.3	10.6M	2.3h
Transformer	11.9 \pm 0.5	10.6M	3.5h
LLoCa-Transformer	<u>1.5</u> \pm 0.1	11.3M	4.5h

... but LLoCa is 4x faster
and requires 100x less FLOPs!

Amplitude regression

Scaling with the dataset size



Data augmentation (DA)
emerges from LLoCa as a
special choice of local frame

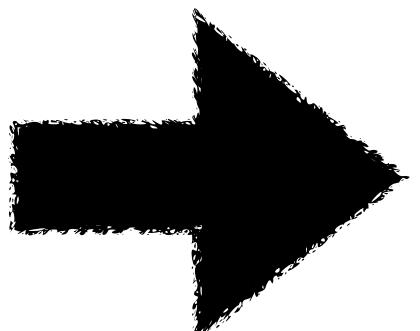
Data augmentation beats
Lorentz-equivariance in
the limit of little training data

Symmetry breaking

Symmetry breaking with reference vectors

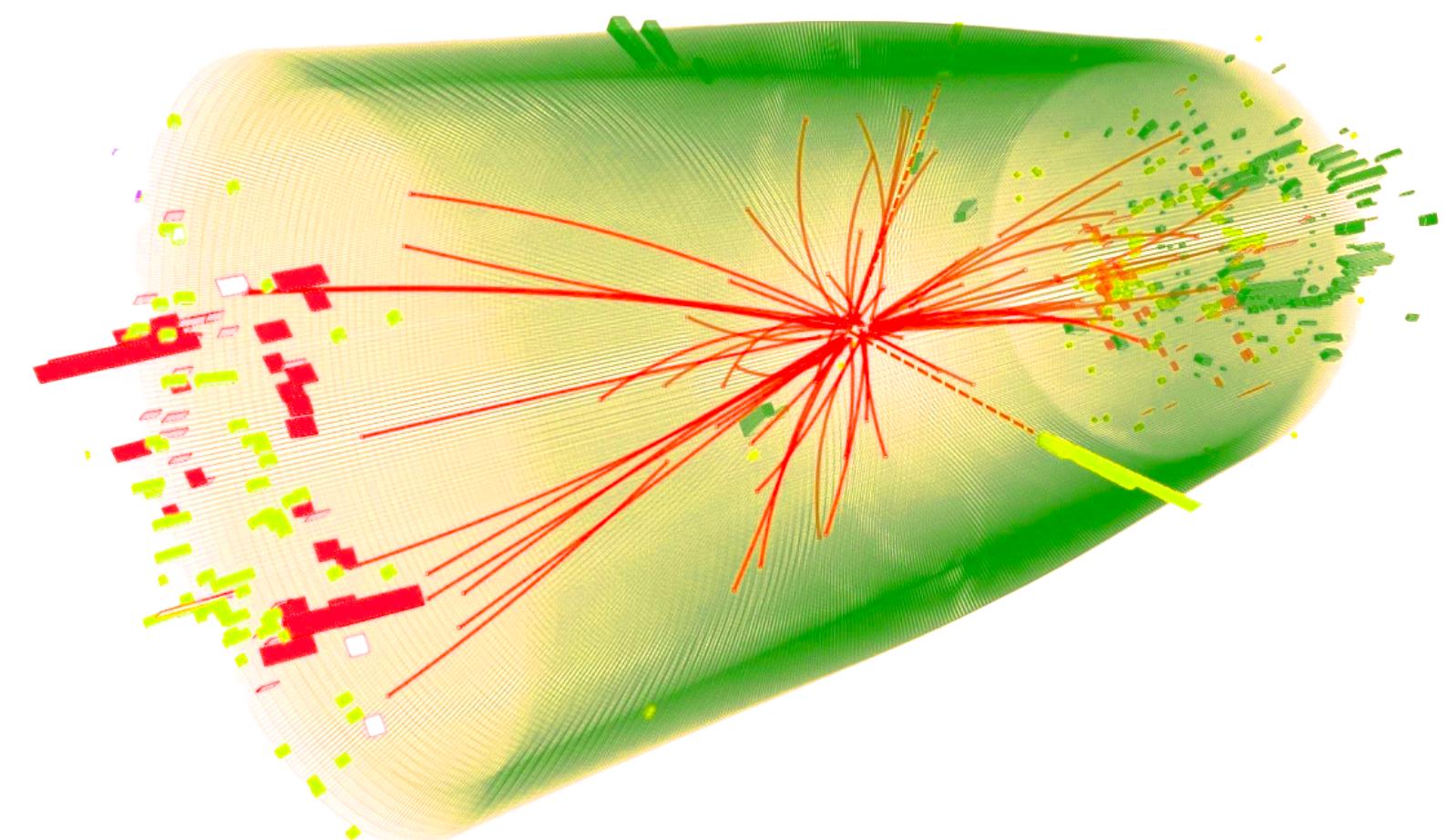
The jet tagging score is not Lorentz-equivariant

- Beam direction breaks invariance under rotations around the x- and y-axis
- Detector breaks boost invariance



Add reference vectors as extra particles to break Lorentz symmetry

- Beam reference vector:
 $x^V = (0,0,0,1)$ or $x_{12}^B = 1$
- Time reference vector:
 $x^V = (1,0,0,0)$



All Lorentz-equivariant taggers use symmetry-breaking inputs

Jet tagging

Effect of symmetry breaking

All networks are formally SO(2)-invariant

Network	Accuracy AUC	
Non-equivariant	0.855	0.9867
L-equivariant w/o SB	0.856	0.9870
$SO(2)$ -equivariant	0.862	0.9878
$SO(3)$ -equivariant w/ SB	0.863	0.9880
L-equivariant w/ SB	0.864	0.9882

SO(2)-equivariant network
> SO(2)-invariant inputs

Lorentz-equivariance
+ symmetry breaking
> SO(2)-equivariance

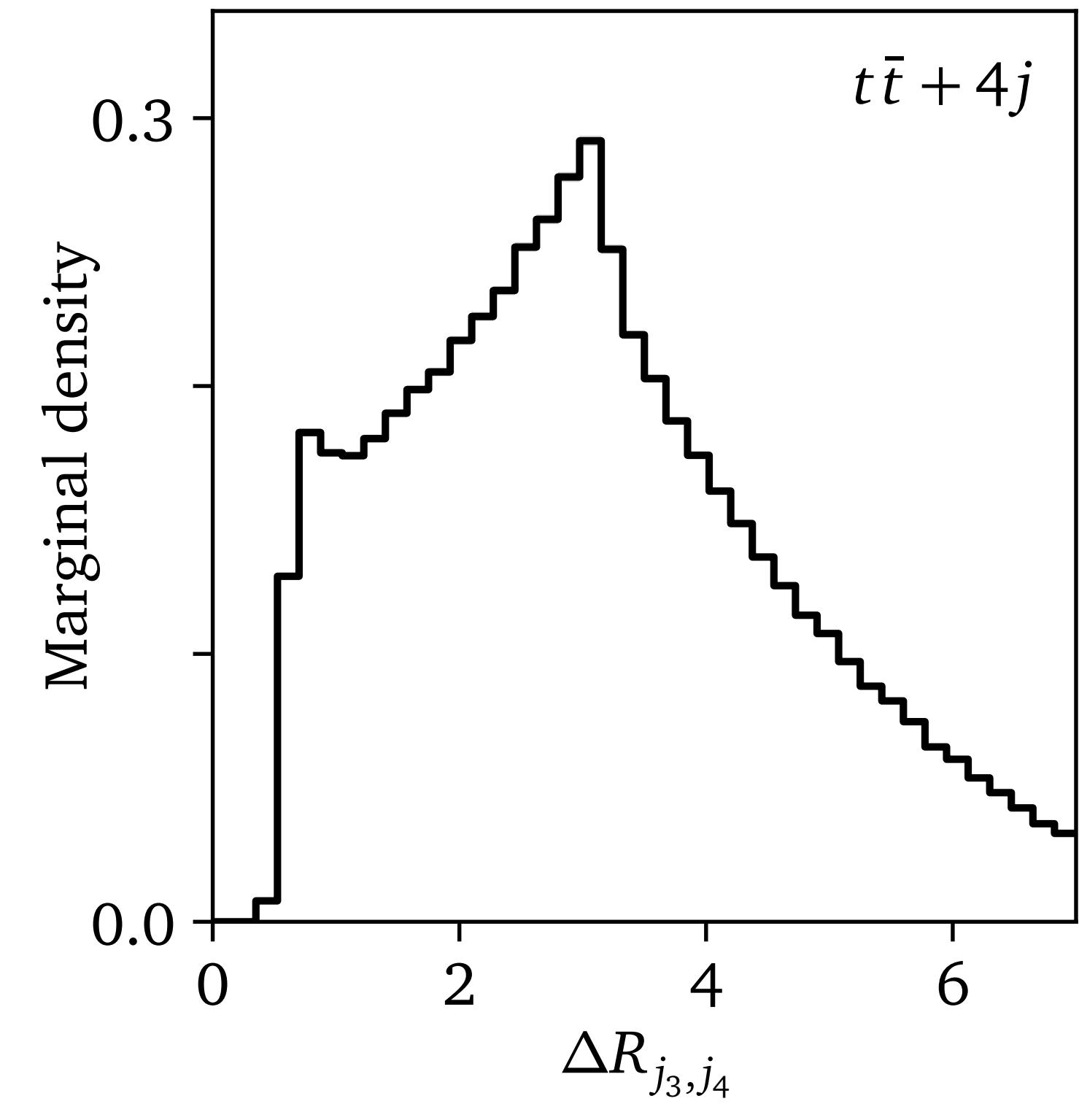
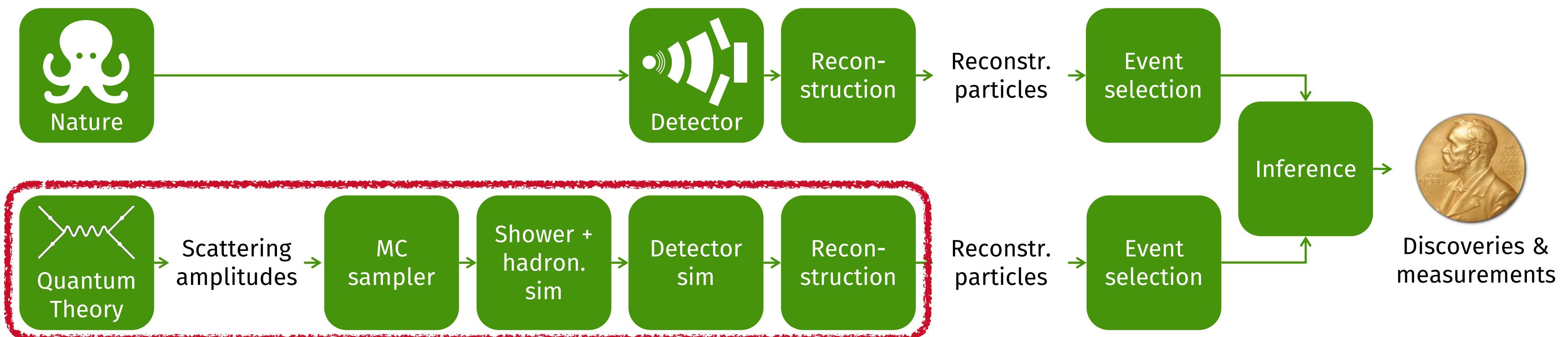
Still SO(2)-invariant inputs

Event generation

Task

Dataset: $pp \rightarrow t_h\bar{t}_h + nj, n = 0\dots4$

- MadGraph + Pythia + Delphes + Reconstruction
- Challenging features: $m_t, m_W, \Delta R_{jj} > 0.5$
- Symmetry breaking with reference multivectors required at reconstruction level



Event generation

Conditional Flow Matching

Continuous normalising flows (CNFs)

connect a simple base density
to a complex target density
through a neural differential equation

$$\frac{d}{dt}x = v_t(x)$$

arXiv:1806.07366

Conditional flow matching (CFM)

is a simple way to train CNFs

by comparing the learned velocity $v_t(x)$
to a conditional target velocity $u_t(x | x_1)$

$$\mathcal{L} = \left\langle (v_t(x) - u_t(x | x_1))^2 \right\rangle$$

arXiv:2210.02747

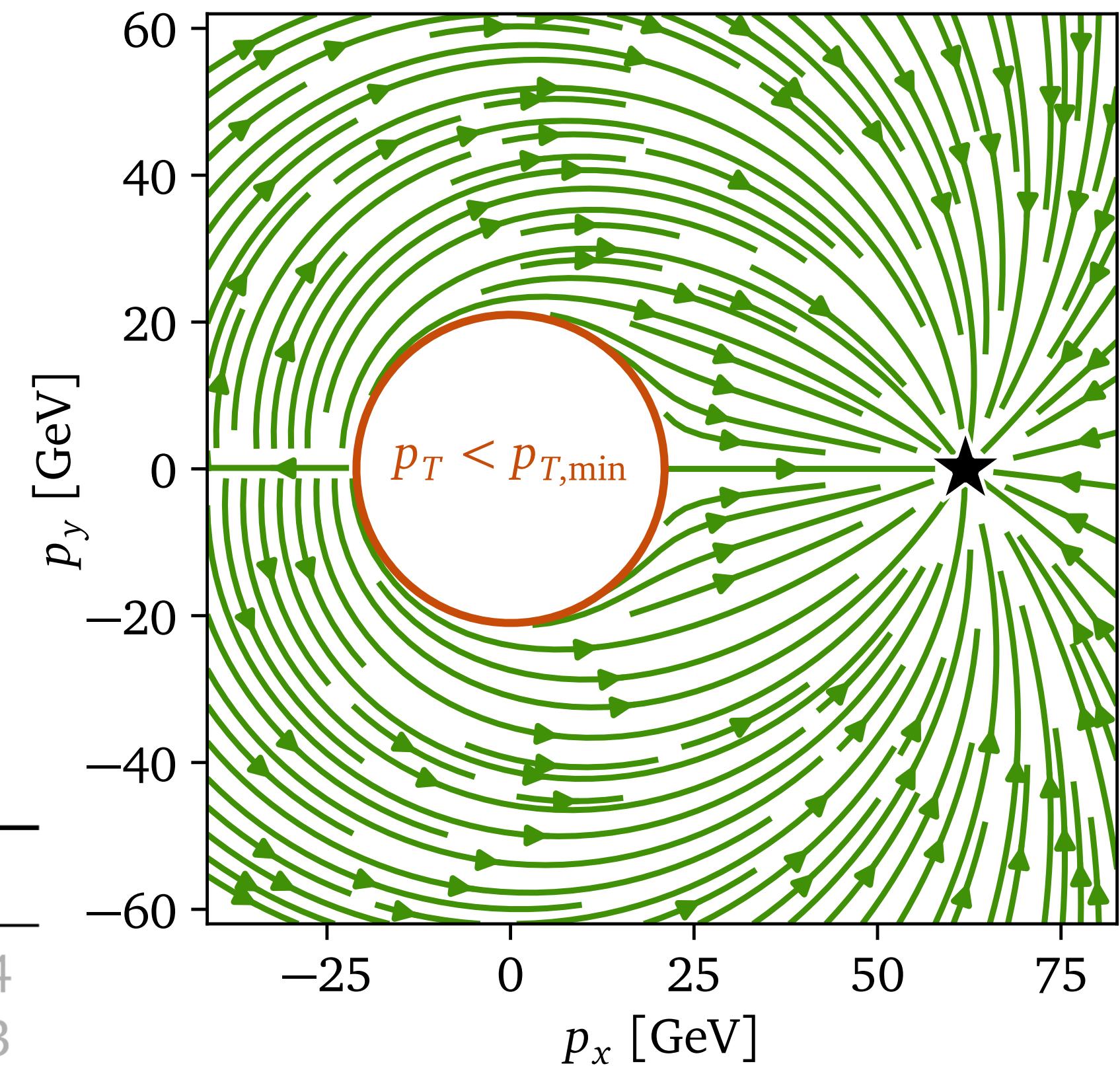
How to pick the target velocity $u_t(x | x_1)$?

Event generation

Physics-inspired target trajectories

Straight trajectories in ‘modified jet momenta’ x :

$$p = \begin{pmatrix} E \\ p_x \\ p_y \\ p_z \end{pmatrix} \rightarrow f^{-1}(p) = x = \begin{pmatrix} x_p \\ x_m \\ x_\eta \\ x_\phi \end{pmatrix} = \begin{pmatrix} \log(p_T - p_{T,\min}) \\ \log m^2 \\ \eta \\ \phi \end{pmatrix}$$



Data	Architecture	Base distribution	Periodic	Neg. log-likelihood	AUC
p	L-GATr	rejection sampling	✓	-30.80 ± 0.17	0.945 ± 0.004
x	MLP	rejection sampling	✓	-32.13 ± 0.05	0.780 ± 0.003
x	L-GATr	rejection sampling	✗	-32.57 ± 0.05	0.530 ± 0.017
x	L-GATr	no rejection sampling	✓	-32.58 ± 0.04	0.523 ± 0.014
x	L-GATr	rejection sampling	✓	-32.65 ± 0.04	0.515 ± 0.009

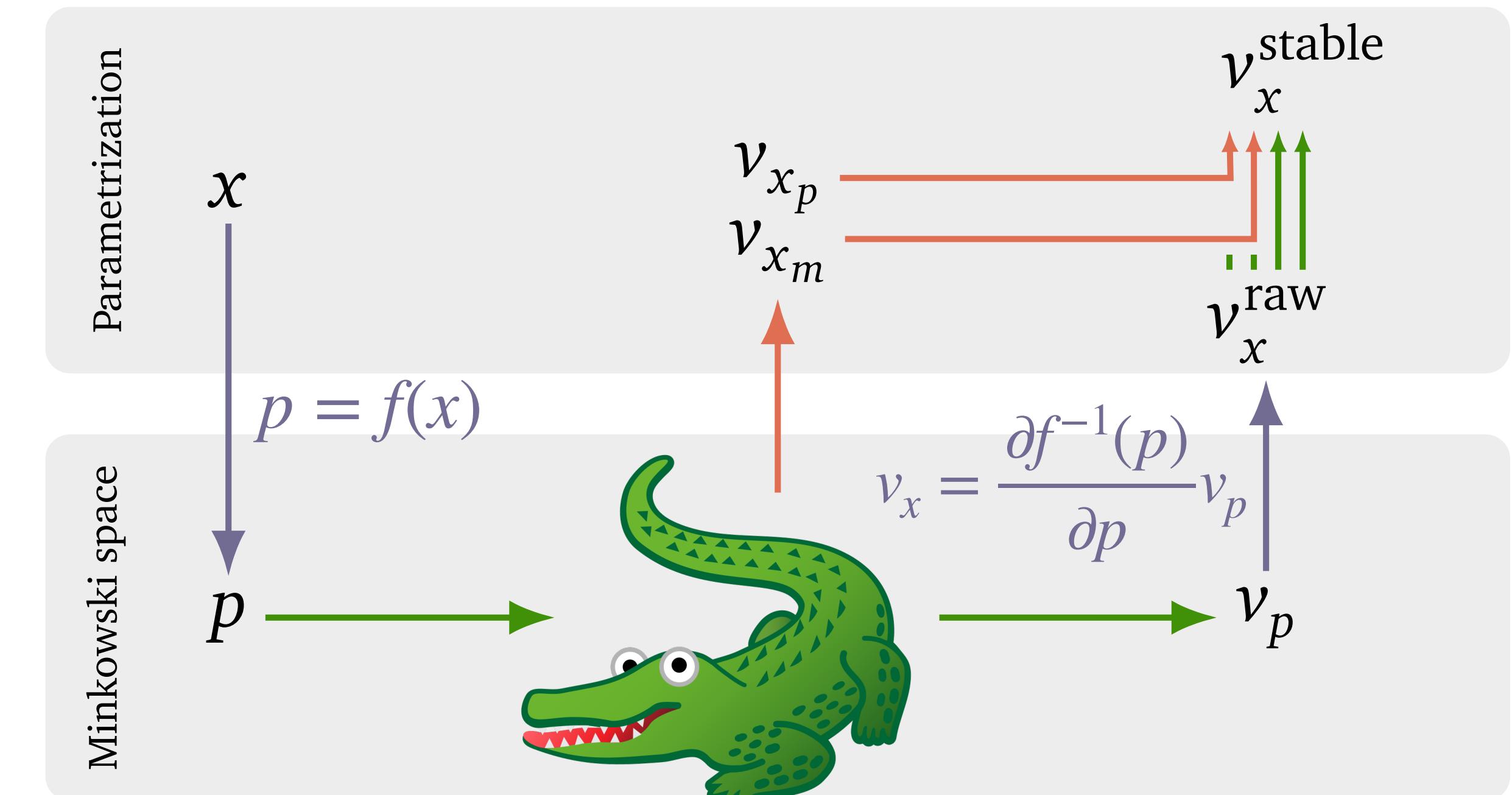
→ default

Event generation

How to build an equivariant CFM field $v_x(x)$?

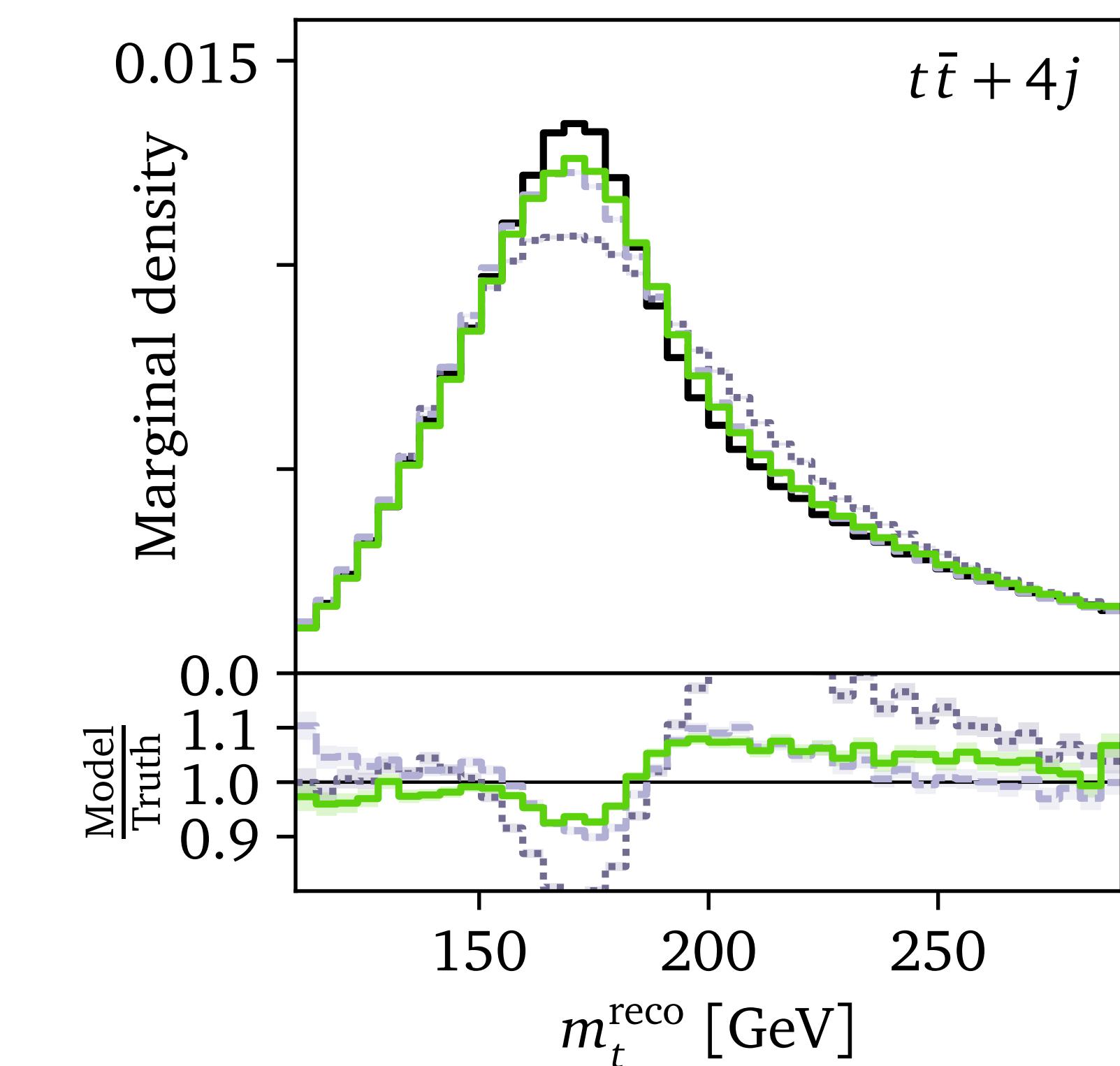
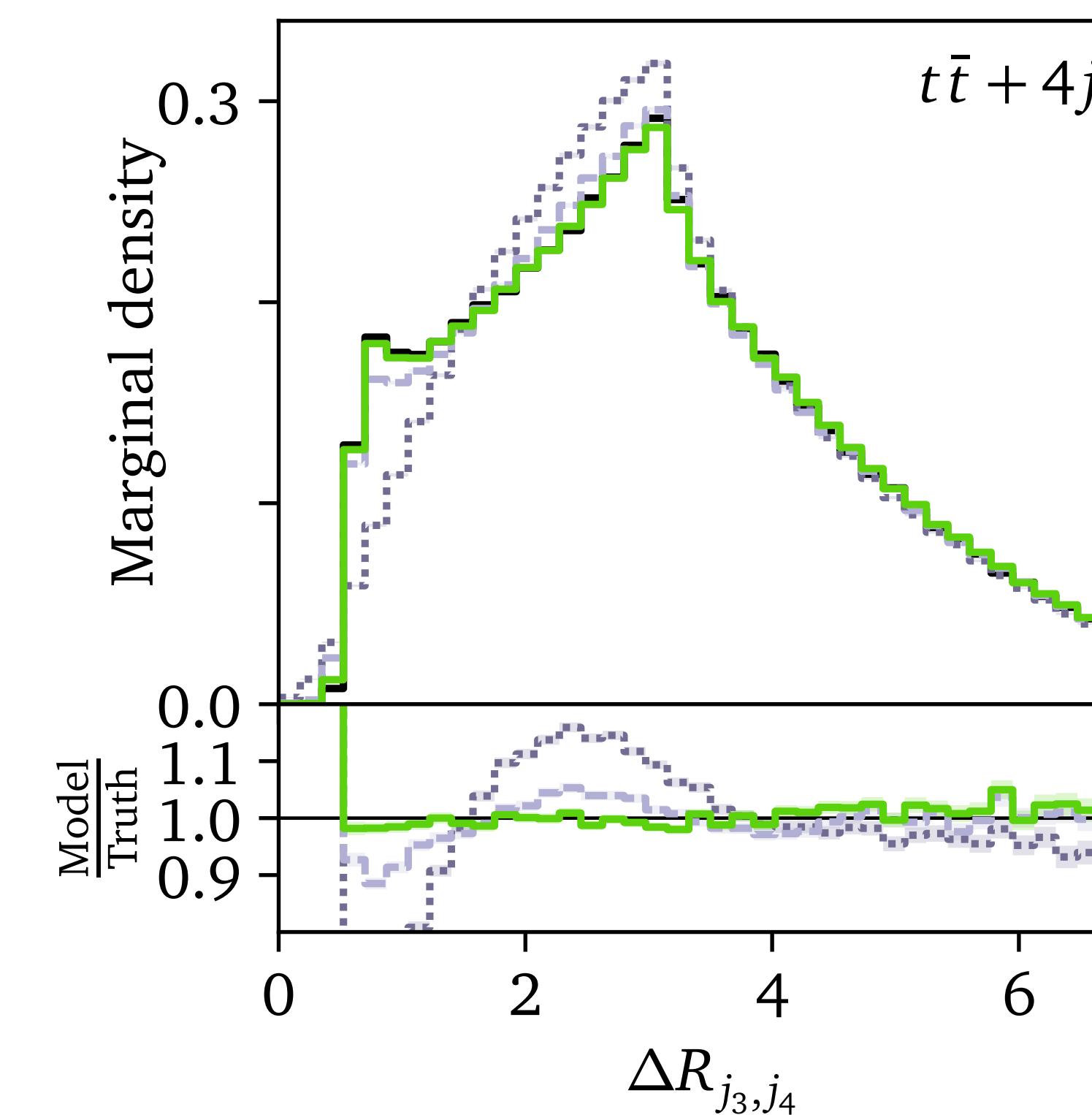
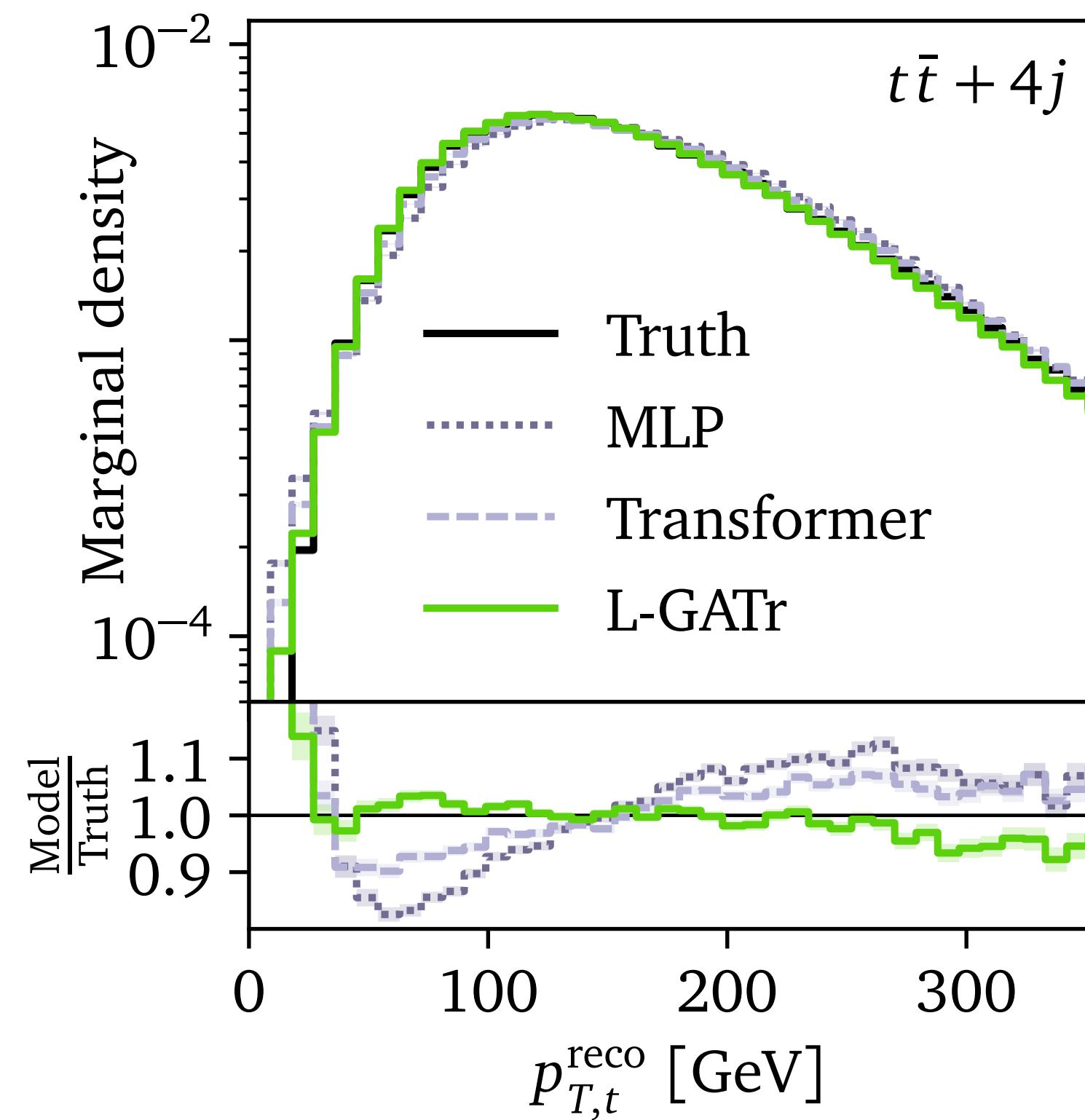
Extend standard CFM workflow
with L-GATr:

- Transformations $f(x)$ between Minkowski space p and the parametrization x
- Equivariant L-GATr operations using multivectors
- Symmetry-breaking operations using scalars (required for numerical stability)



Event generation

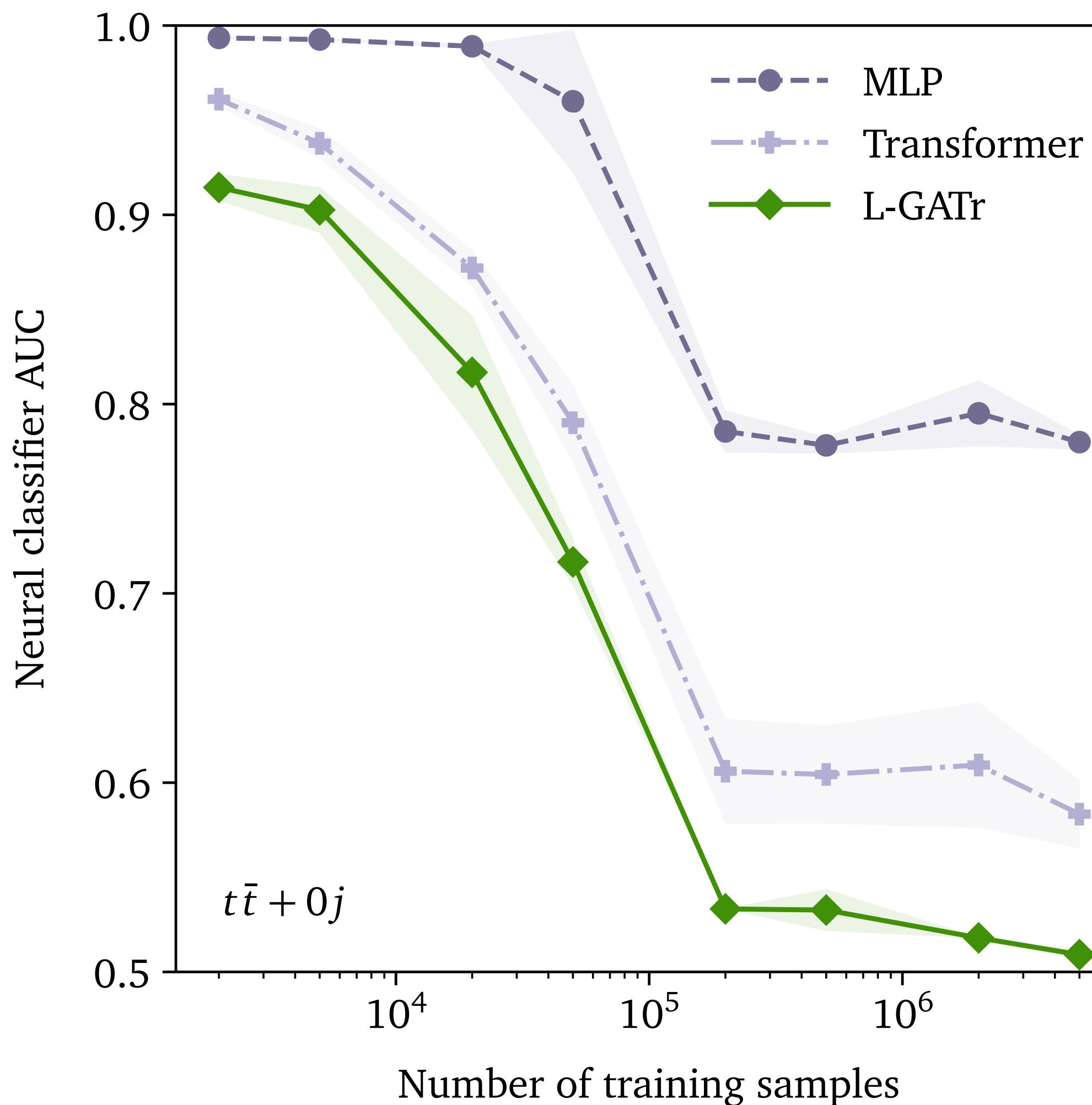
Kinematic distributions



Equivariance helps,
especially for angular correlations

Event generation

Neural classifier metric



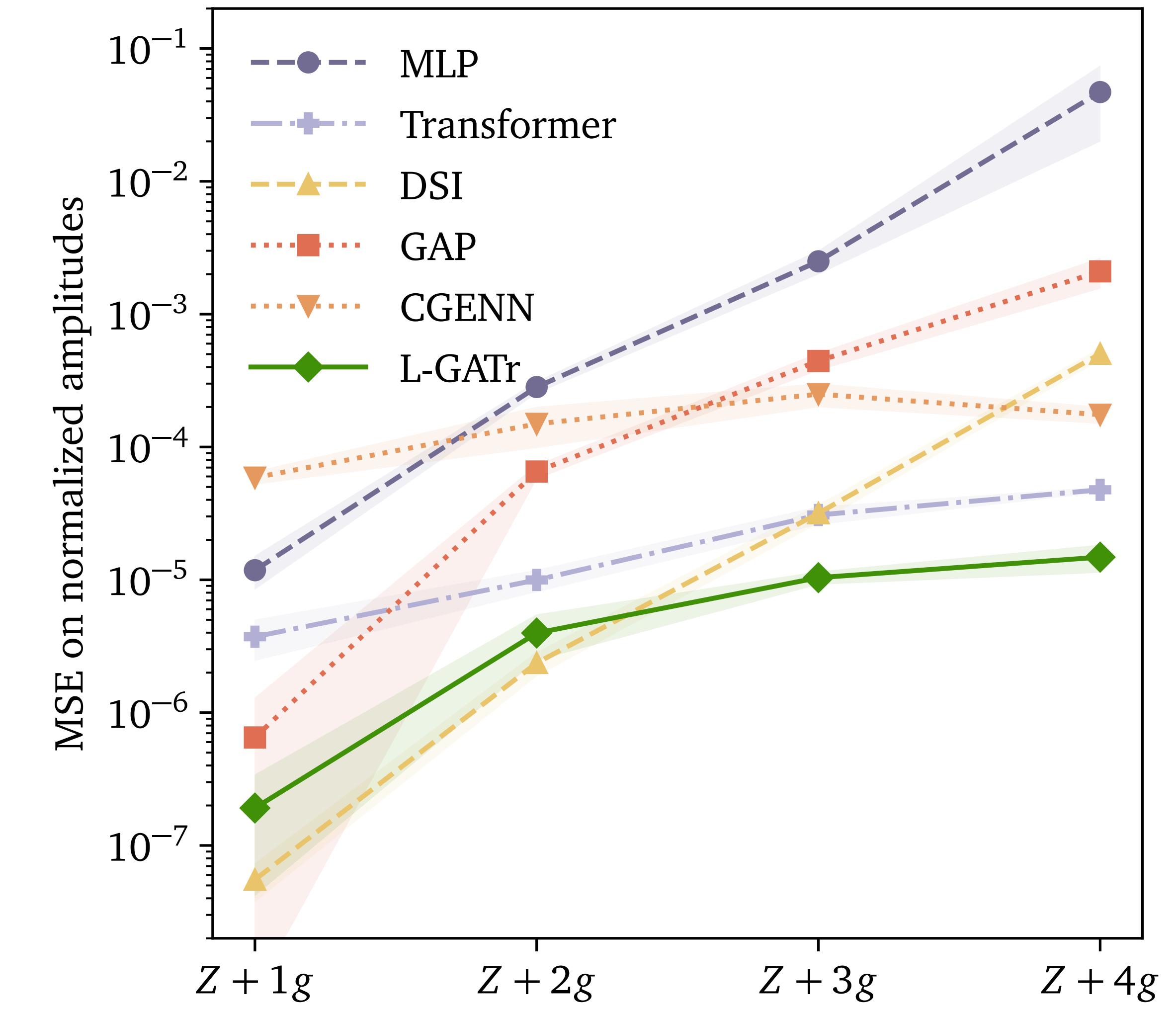
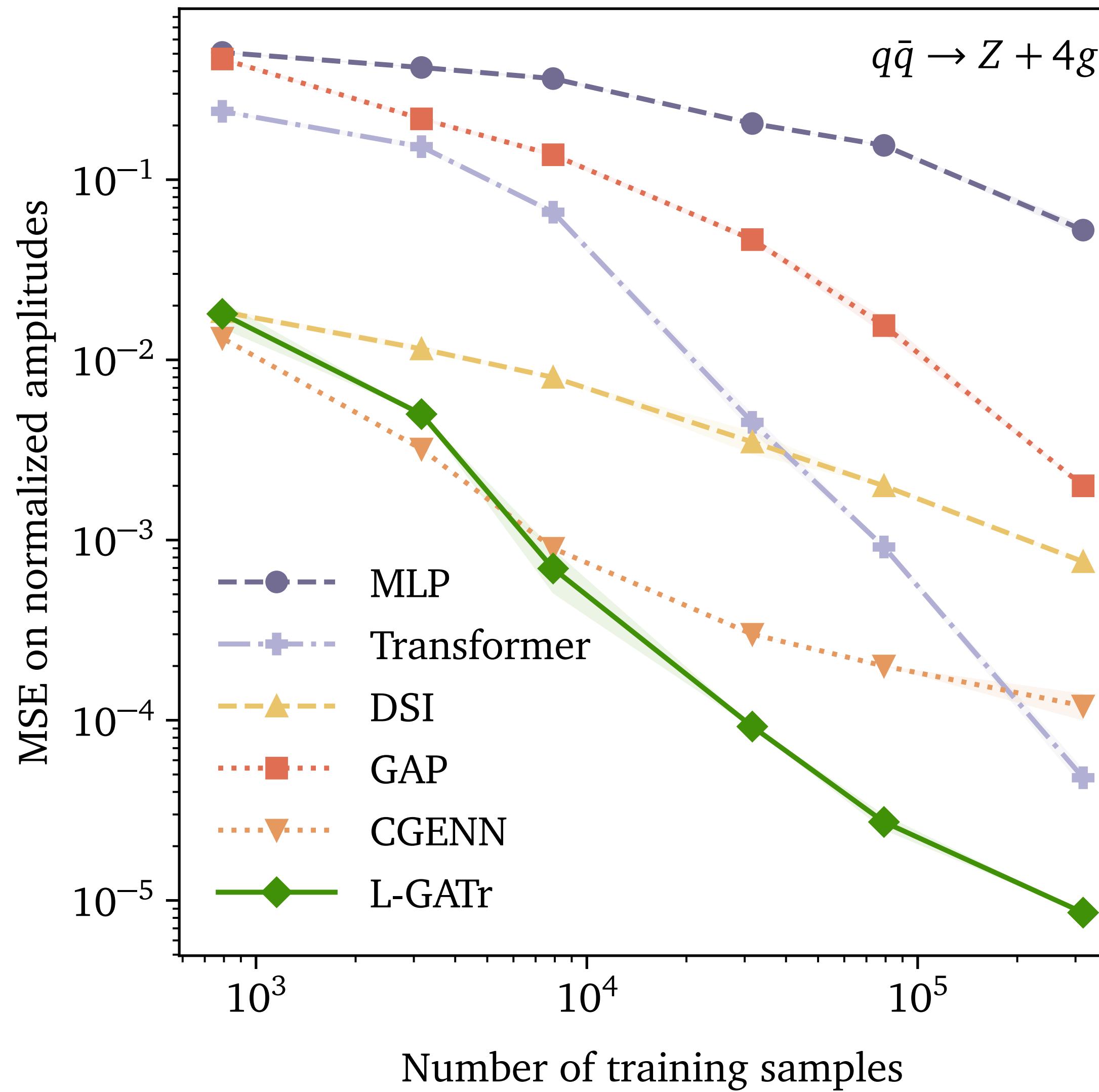
L-GATr generates samples that a classifier can barely distinguish from the ground truth

Summary

- Lorentz and permutation equivariance (L-GATr, LLoCa) help in LHC applications
- L-GATr is much slower, LLoCa only a bit
- Don't forget Lorentz symmetry breaking!
- LLoCa contains data augmentation and $\text{SO}(2)$ -, $\text{SO}(3)$ -equivariance as special cases
- Didn't find the application for higher-order representations yet

Bonus material

Amplitude regression



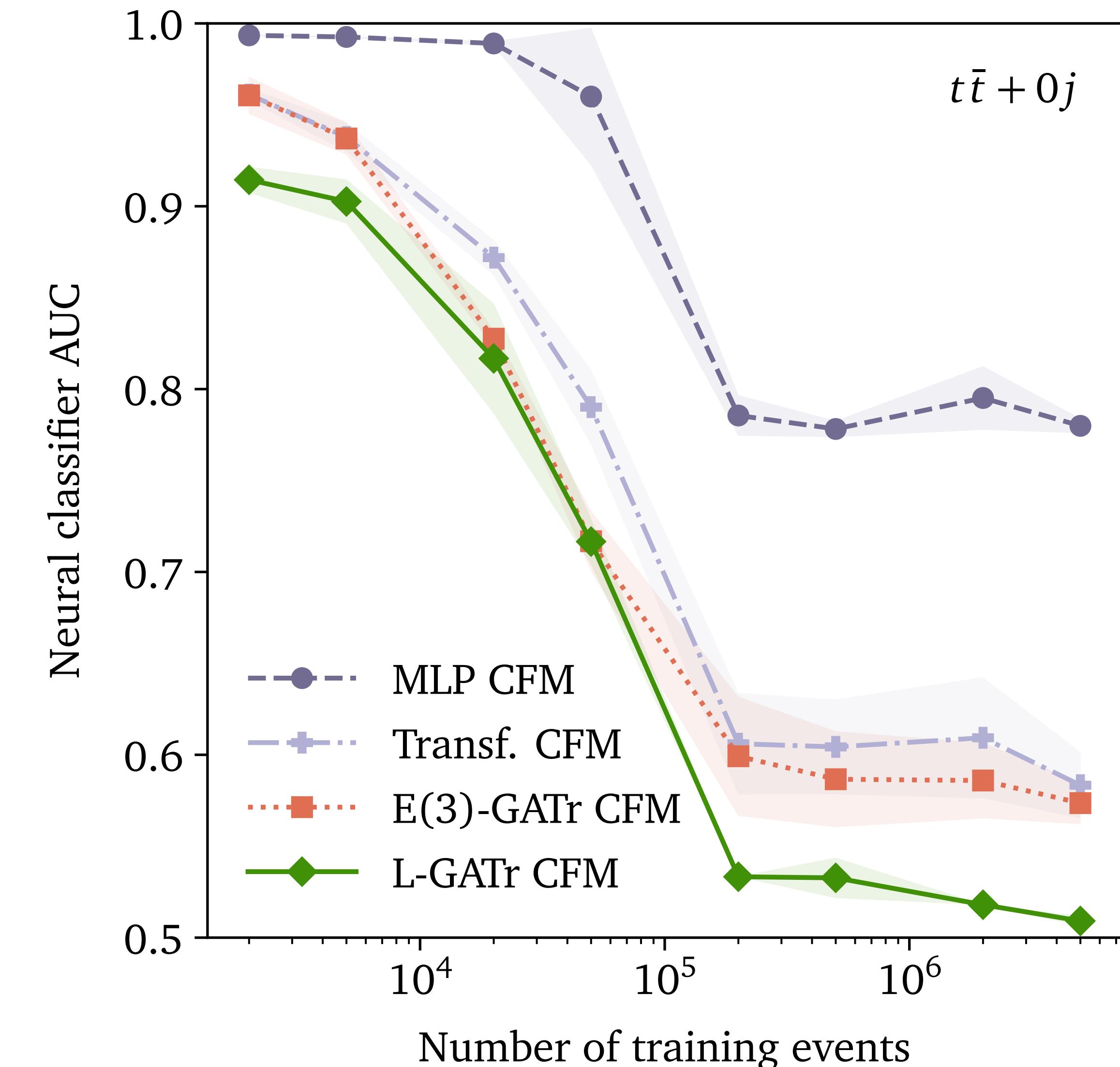
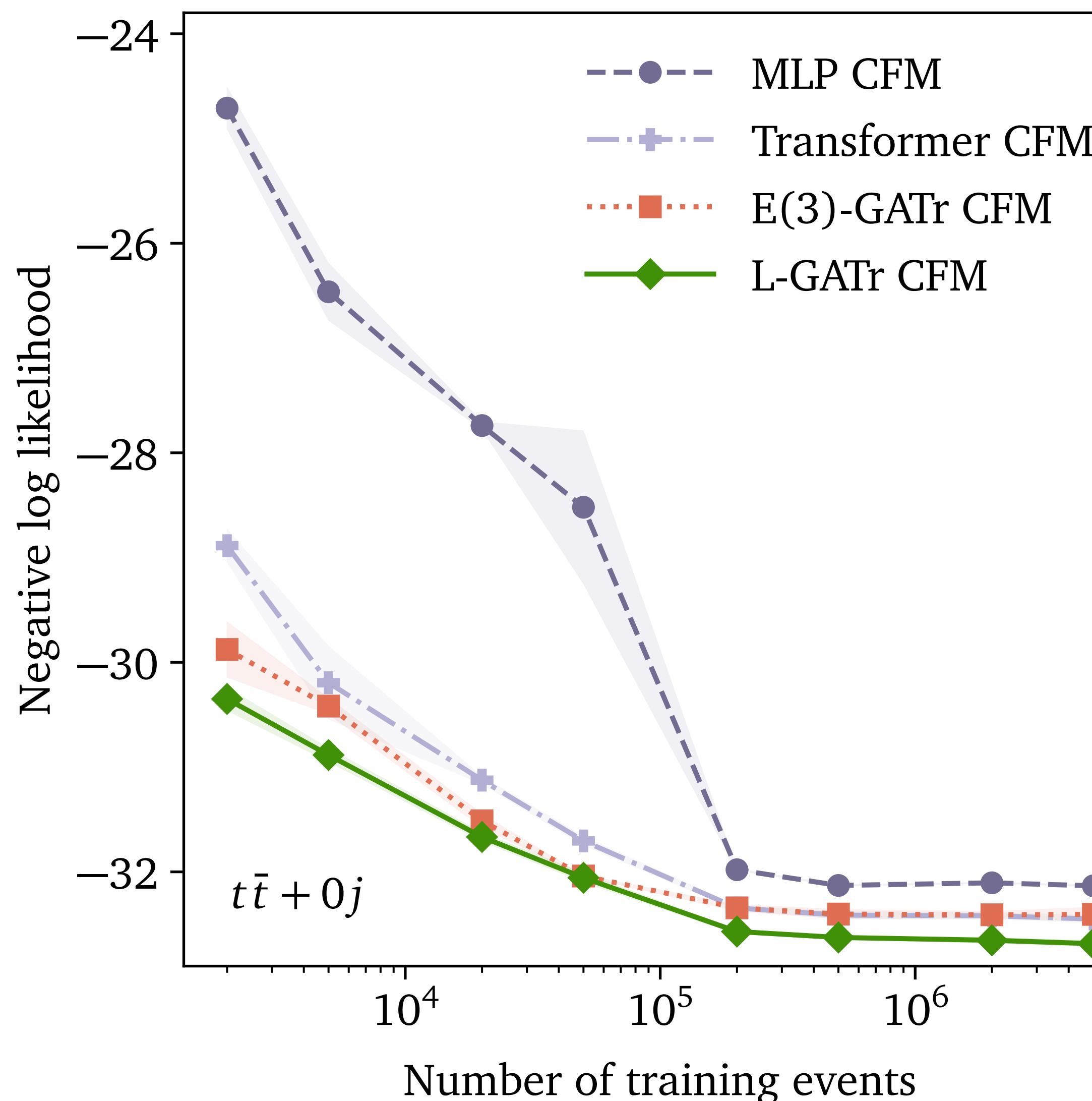
Network	Accuracy	AUC	$1/\epsilon_B$ ($\epsilon_S = 0.5$)	$1/\epsilon_B$ ($\epsilon_S = 0.3$)
TopoDNN [54]	0.916	0.972	–	295 ± 5
LoLa [9]	0.929	0.980	–	722 ± 17
<i>N</i> -subjettiness [55]	0.929	0.981	–	867 ± 15
PFN [56]	0.932	0.9819	247 ± 3	888 ± 17
TreeNiN [57]	0.933	0.982	–	1025 ± 11
ParticleNet [58]	0.940	0.9858	397 ± 7	1615 ± 93
ParT [59]	0.940	0.9858	413 ± 16	1602 ± 81
MIParT [60]	0.942	0.9868	505 ± 8	2010 ± 97
LorentzNet* [10]	0.942	0.9868	498 ± 18	2195 ± 173
CGENN* [14]	0.942	0.9869	500	2172
PELICAN* [42]	0.9426 ± 0.0002	0.9870 ± 0.0001	–	2250 ± 75
L-GATr* [35]	0.9423 ± 0.0002	0.9870 ± 0.0001	540 ± 20	2240 ± 70
ParticleNet-f.t. [60]	0.942	0.9866	487 ± 9	1771 ± 80
ParT-f.t. [60]	0.944	0.9877	691 ± 15	2766 ± 130
MIParT-f.t. [60]	0.944	0.9878	640 ± 10	2789 ± 133
L-GATr-f.t.* (new)	0.9442 ± 0.0002	0.98792 ± 0.00004	661 ± 24	3005 ± 186

	All classes	$H \rightarrow b\bar{b}$	$H \rightarrow c\bar{c}$	$H \rightarrow gg$	$H \rightarrow 4q$	$H \rightarrow l\nu q\bar{q}'$	$t \rightarrow bq\bar{q}'$	$t \rightarrow bl\nu$	$W \rightarrow q\bar{q}'$	$Z \rightarrow q\bar{q}$	
	Accuracy	AUC	Rej _{50%}	Rej _{50%}	Rej _{50%}	Rej _{99%}	Rej _{50%}	Rej _{99.5%}	Rej _{50%}	Rej _{50%}	
ParticleNet [58]	0.844	0.9849	7634	2475	104	954	3339	10526	11173	347	283
ParT [59]	0.861	0.9877	10638	4149	123	1864	5479	32787	15873	543	402
MIParT [60]	0.861	0.9878	10753	4202	123	1927	5450	31250	16807	542	402
L-GATr	0.865	0.9884	12195	4819	128	2304	5764	37736	19231	580	427

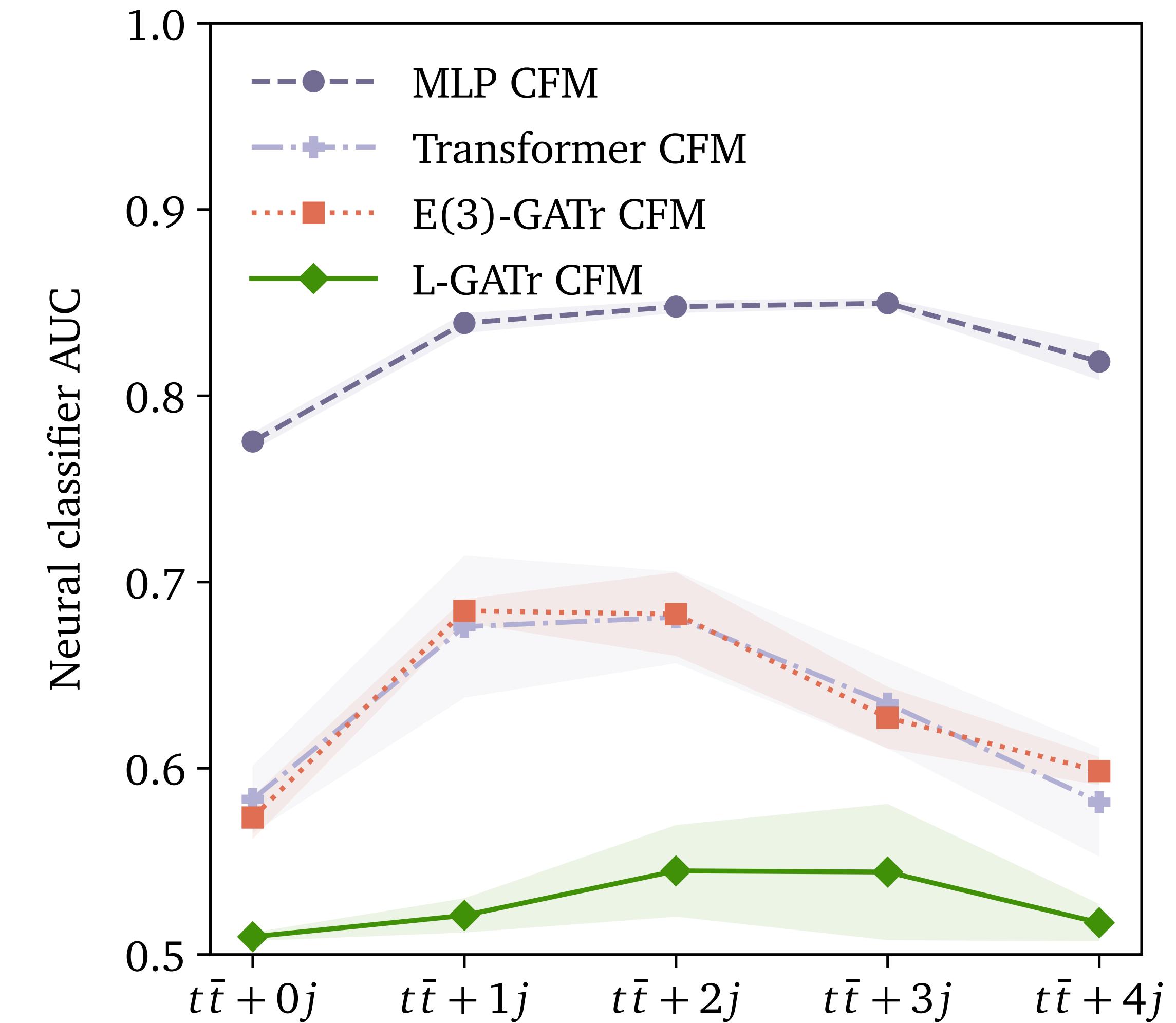
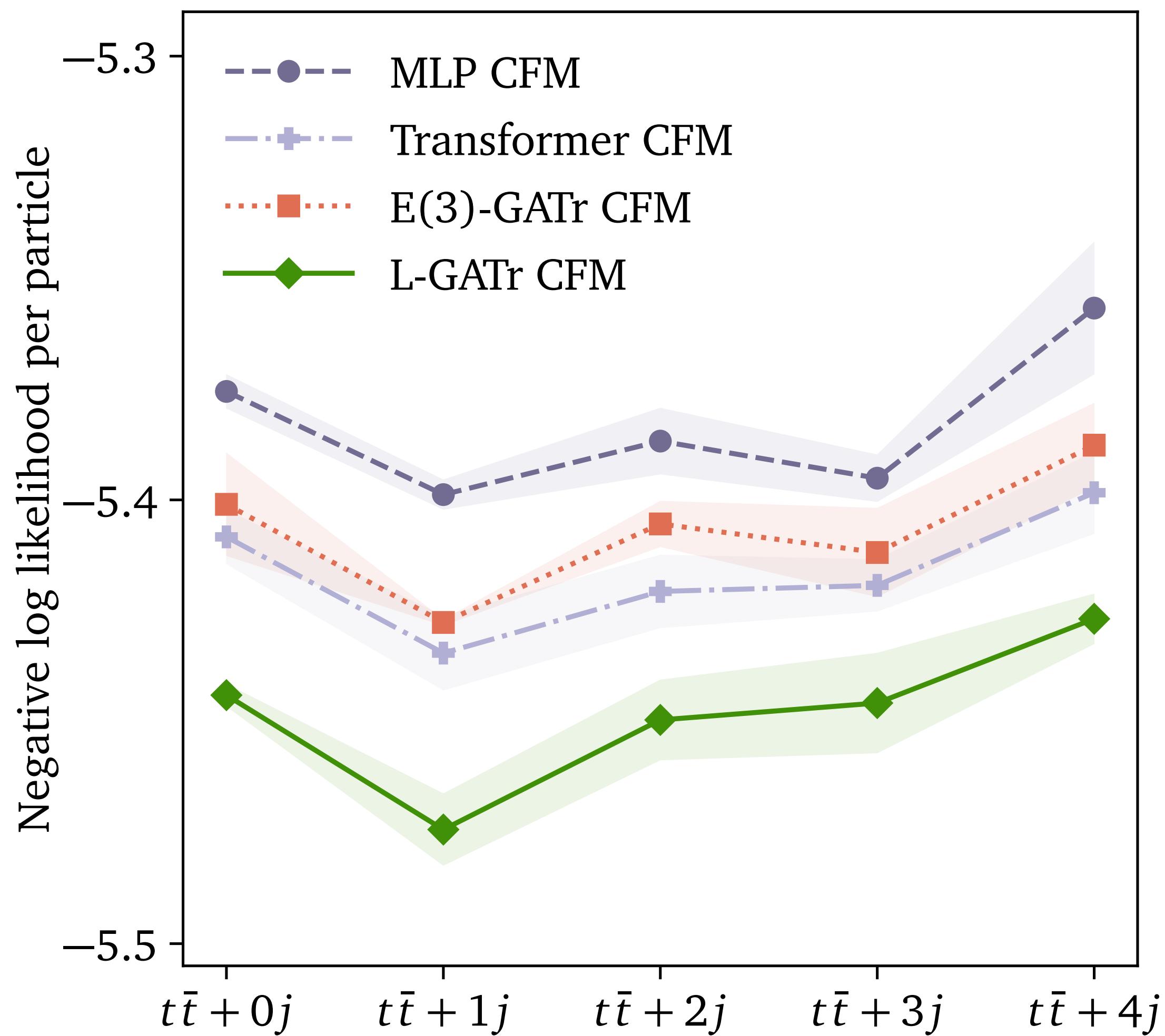
Model	Accuracy (\uparrow)	AUC (\uparrow)	Time	FLOPs
PFN [25]	0.772	0.9714	3h	3M
P-CNN [38]	0.809	0.9789	3h	12M
MIParT [45]	0.861	0.9878	–	–
LorentzNet* [19]	0.847	0.9856	57h	676M
L-GATr* [9]	0.866	0.9885	180h	2060M
ParticleNet [34]	0.844	0.9849	27h	413M
LLoCa-ParticleNet*	<u>0.848</u>	<u>0.9857</u>	41h	517M
ParT [35]	0.861	0.9877	38h	211M
LLoCa-ParT*	<u>0.864</u>	<u>0.9882</u>	67h	315M
Transformer	0.855	0.9867	17h	210M
LLoCa-Transformer*	<u>0.864</u>	<u>0.9882</u>	33h	301M

Beam	Time	Embedding	AUC	$1/\epsilon_B$ ($\epsilon_S = 0.3$)
-	-	Token	\times	0.9846 ± 0.0002 1494 ± 62
$x_3^V = \pm 1$	-	Token	\times	0.9854 ± 0.0005 1684 ± 221
$x_{12}^B = x_{13}^B = x_{23}^B = 1$	$x_0^V = 1$	Token	\times	0.9864 ± 0.0002 2076 ± 75
-	$x_0^V = 1$	Token	\times	0.9865 ± 0.0001 2179 ± 123
$x_{12}^B = 1$	$x_0^V = 1$	Channel	\times	0.9866 ± 0.0001 2150 ± 82
$x_0^V = 1, x_3^V = \pm 1$	$x_0^V = 1$	Token	\times	0.9869 ± 0.0001 2282 ± 174
$x_3^V = \pm 1$	$x_0^V = 1$	Token	\times	0.9869 ± 0.0001 2293 ± 141
-	-	Token	\checkmark	0.9869 ± 0.0001 2179 ± 156
$x_{12}^B = 1$	$x_0^V = 1$	Token	\checkmark	0.9870 ± 0.0001 2173 ± 83
$x_{12}^B = 1$	$x_0^V = 1$	Token	\times	0.9870 ± 0.0001 2240 ± 70

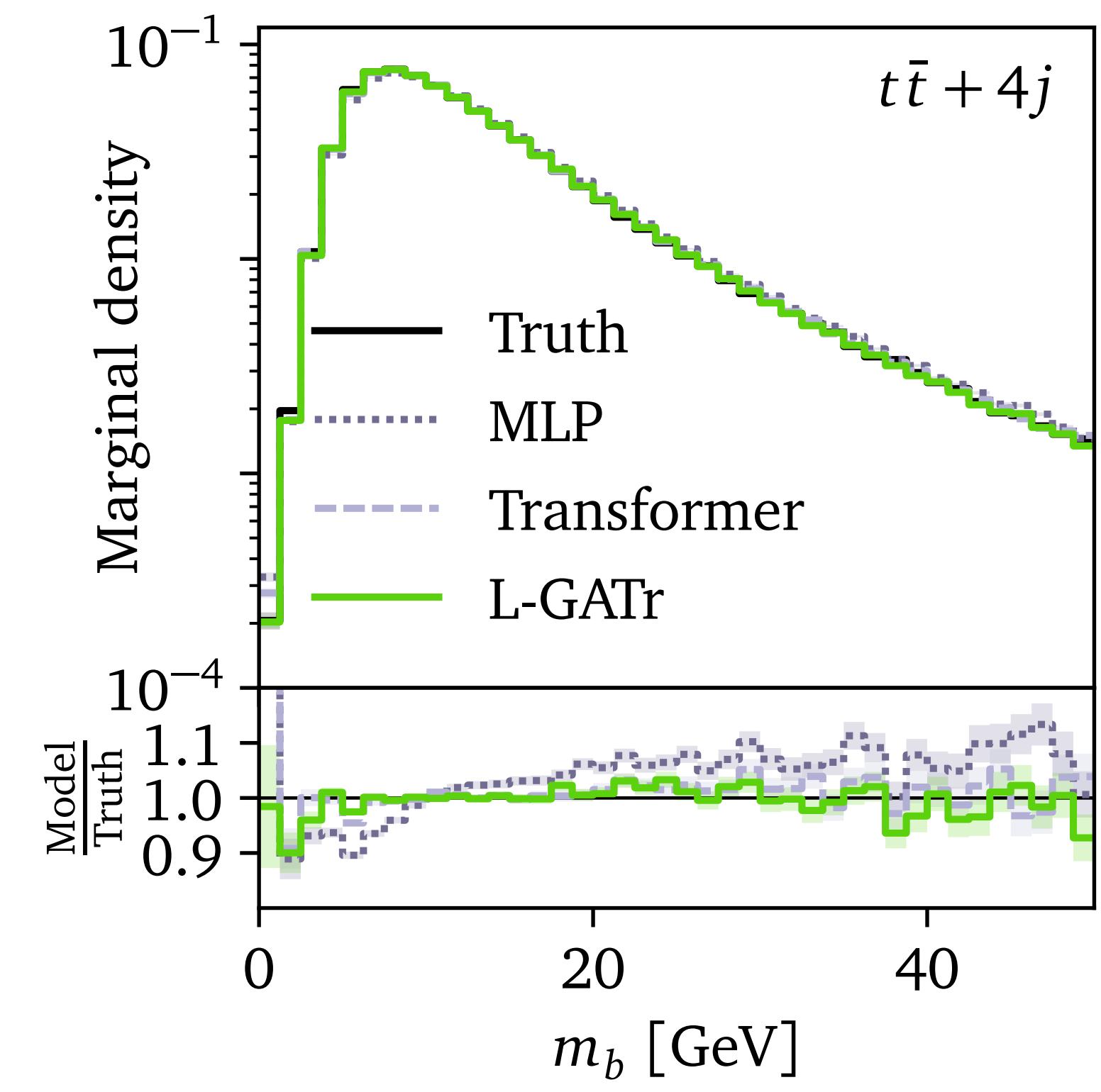
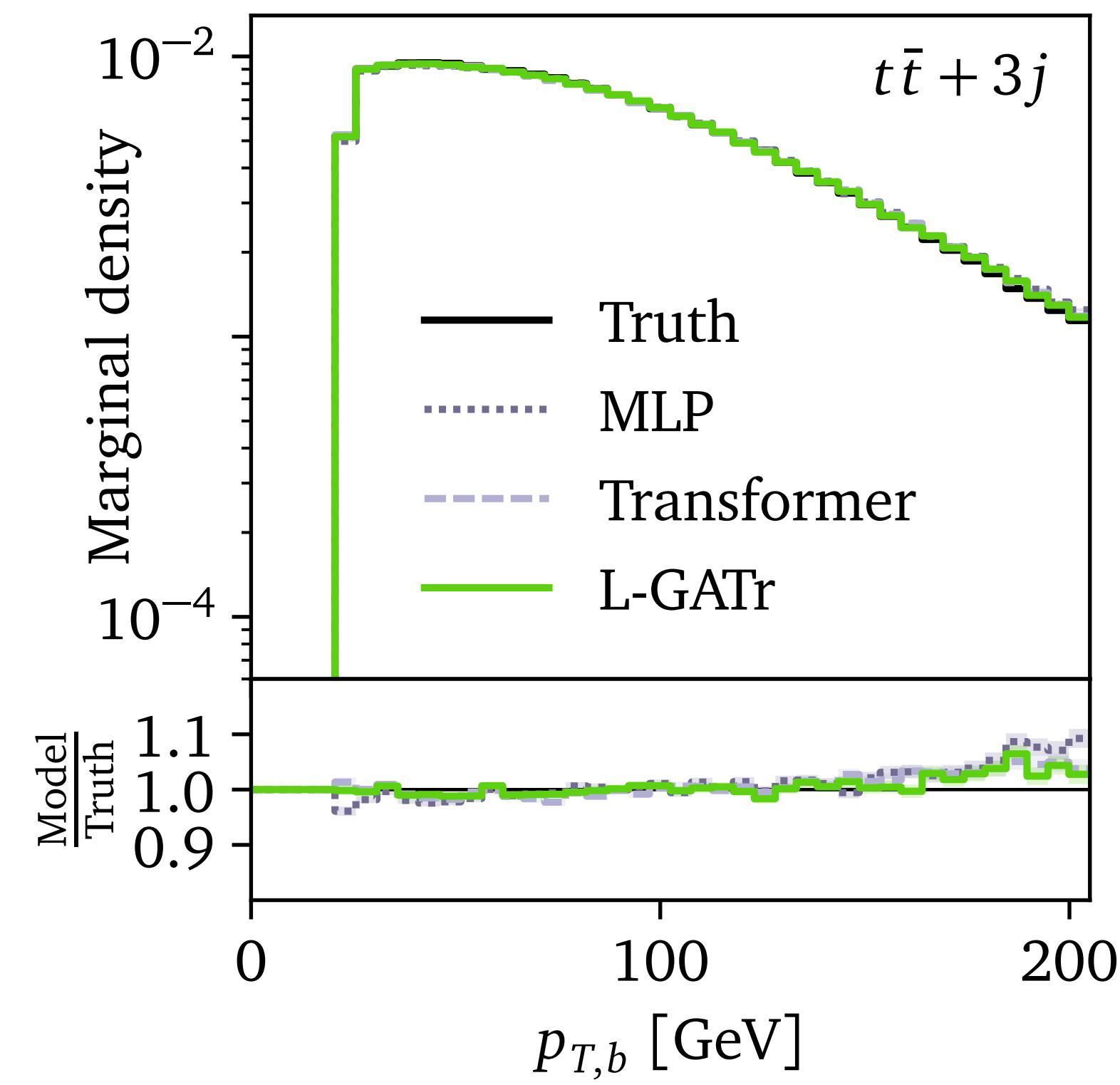
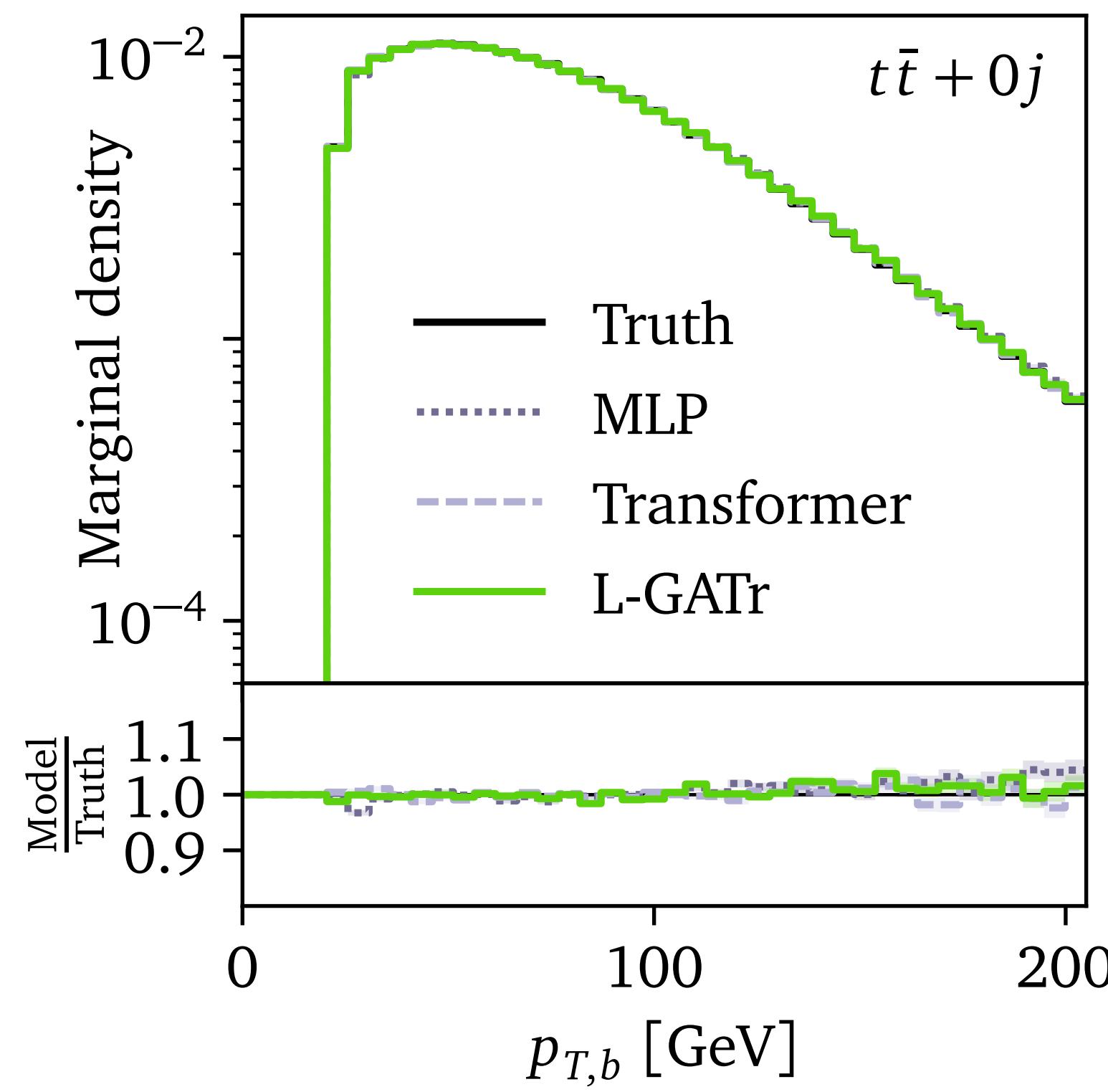
Event generation



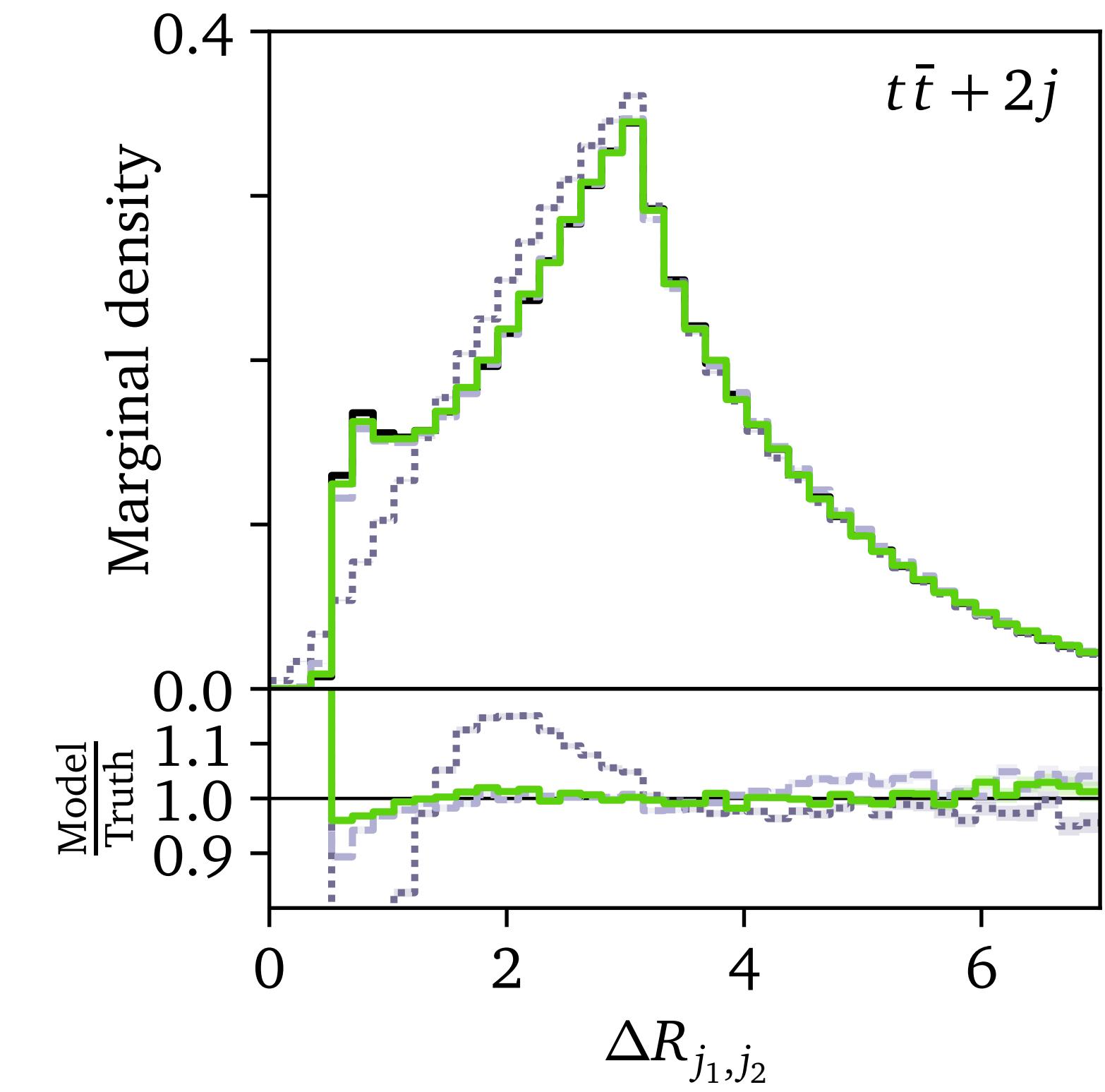
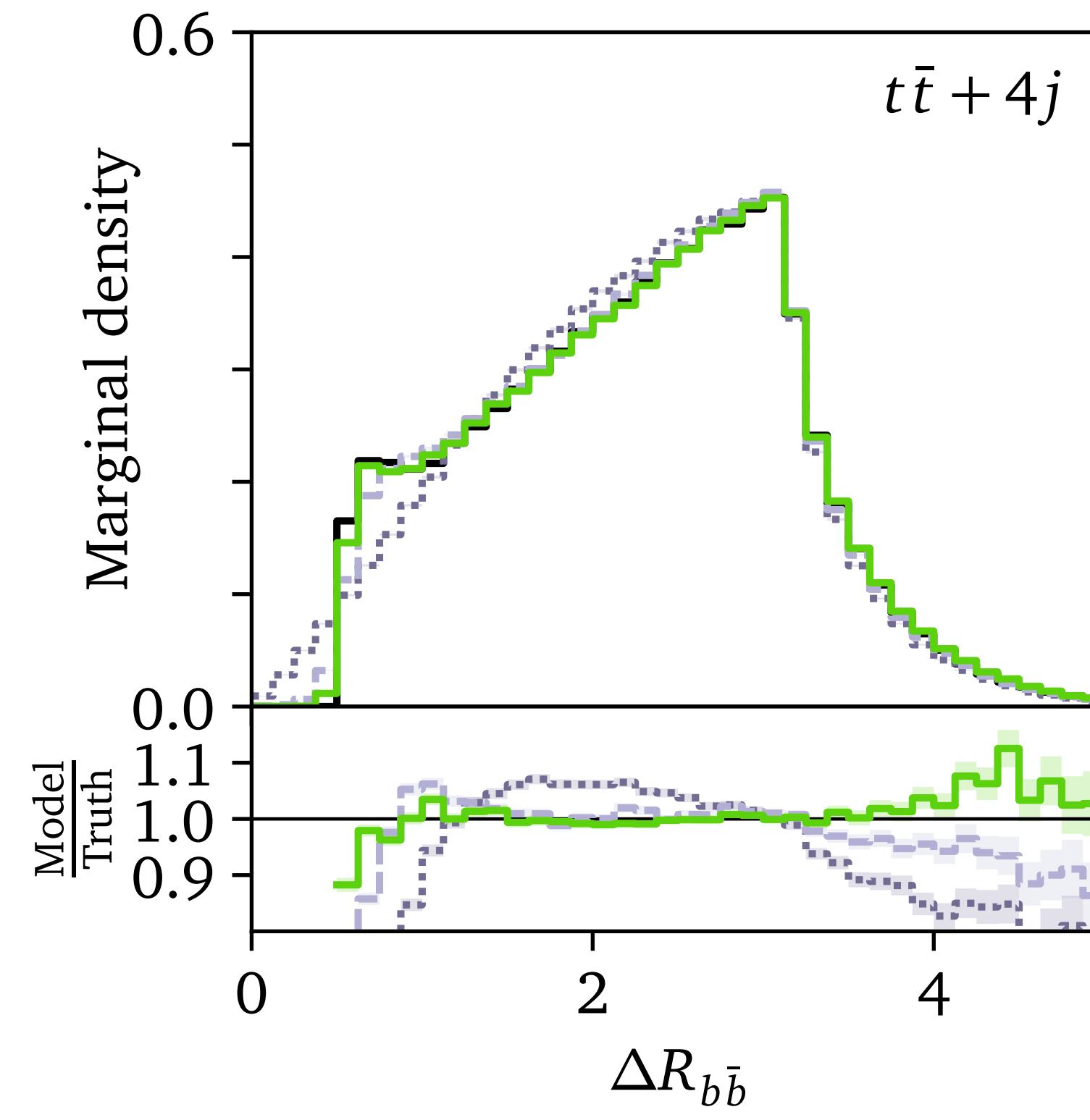
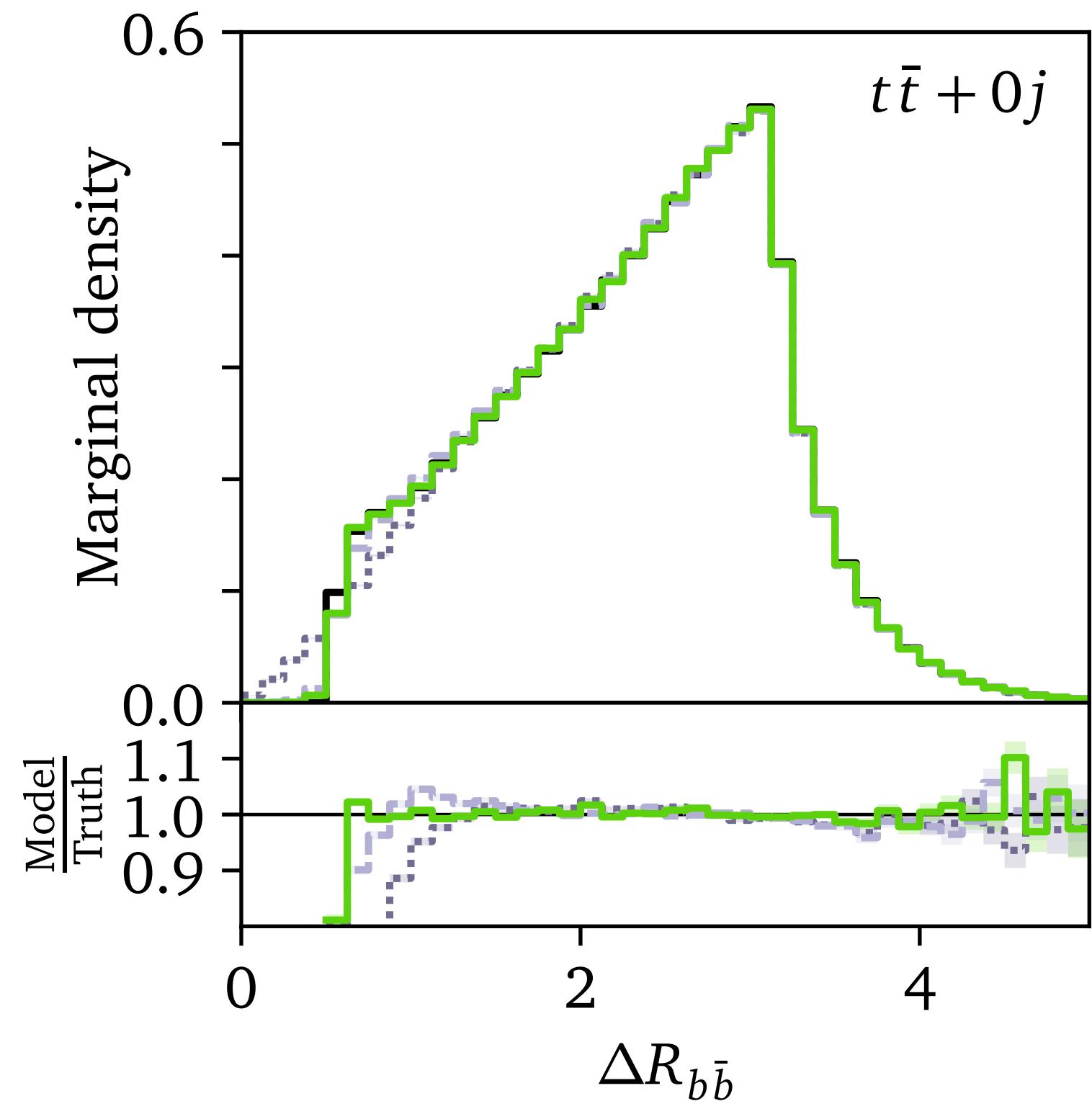
Event generation



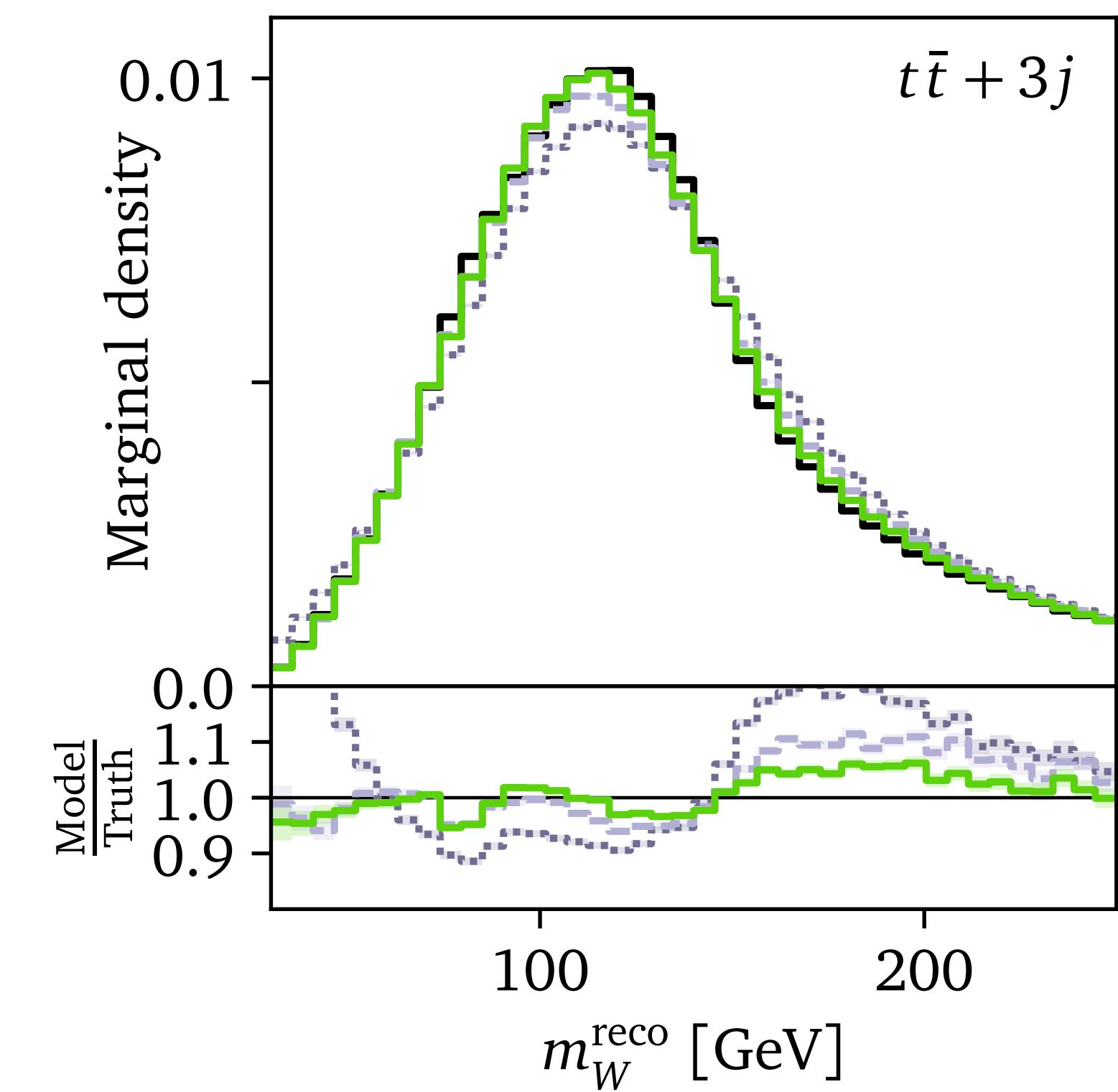
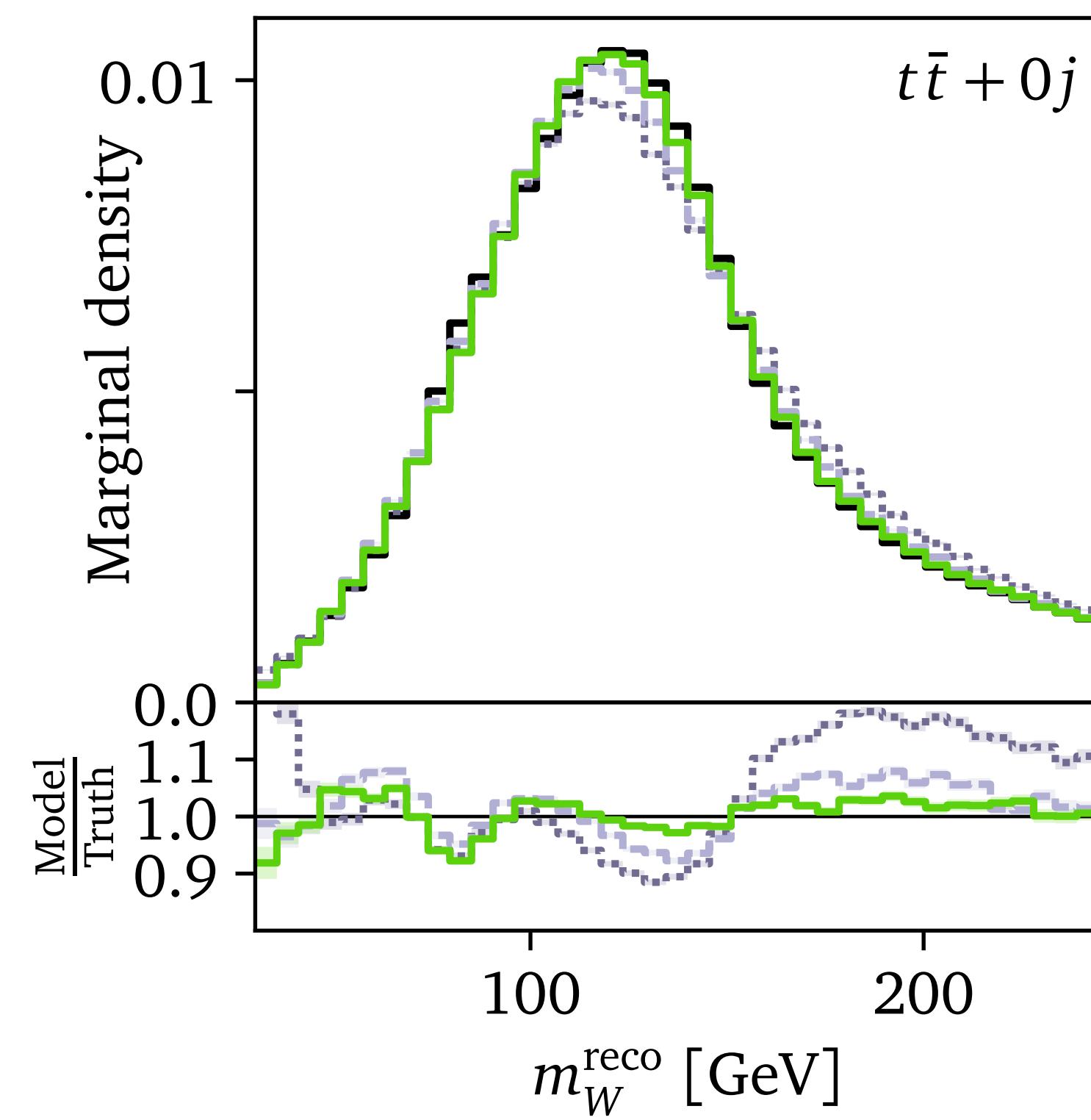
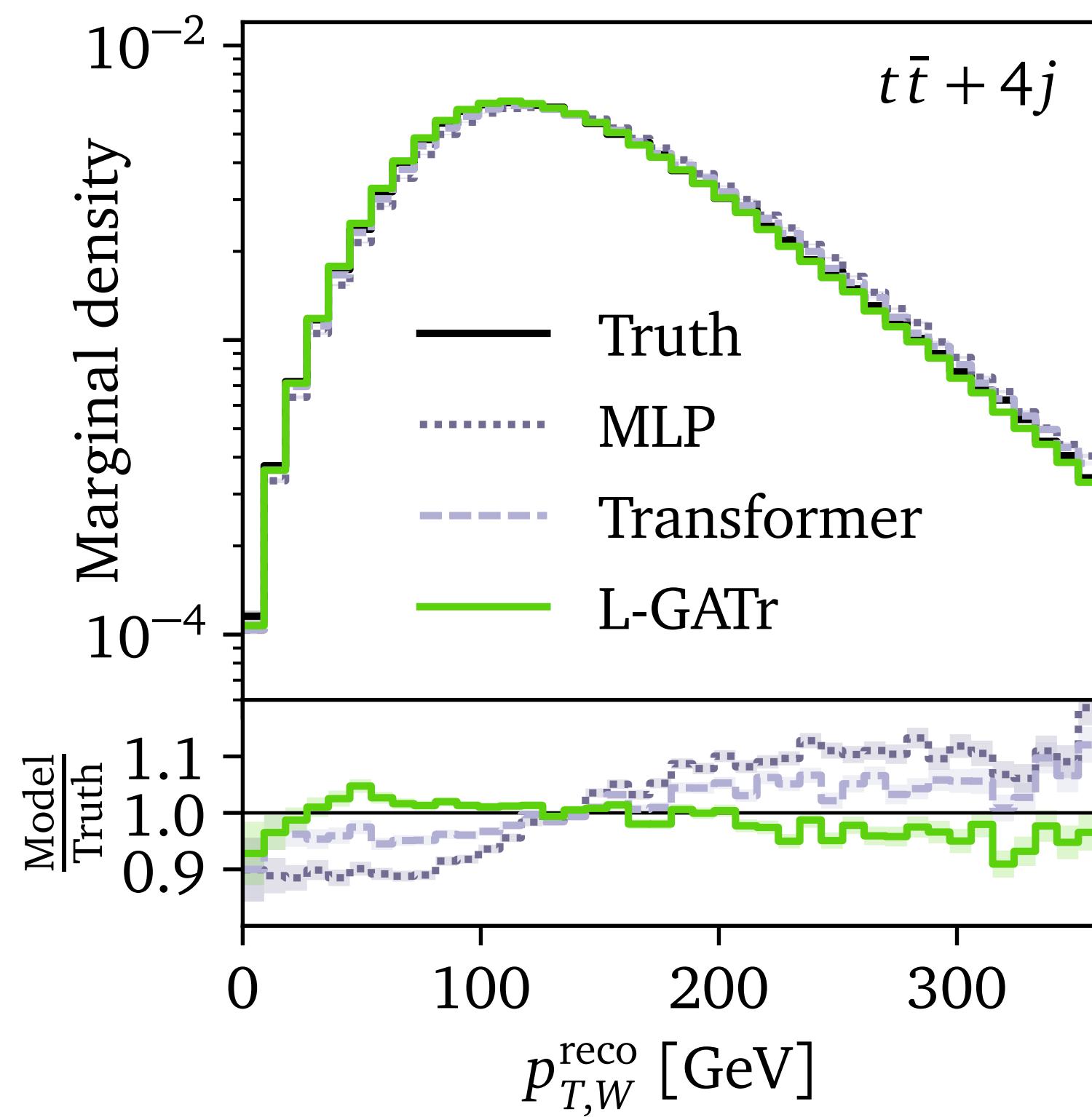
Event generation



Event generation



Event generation



Event generation

