How to make any neural network Lorentz-Equivariant

Luigi Favaro

When the M meets the P

19.03.2025

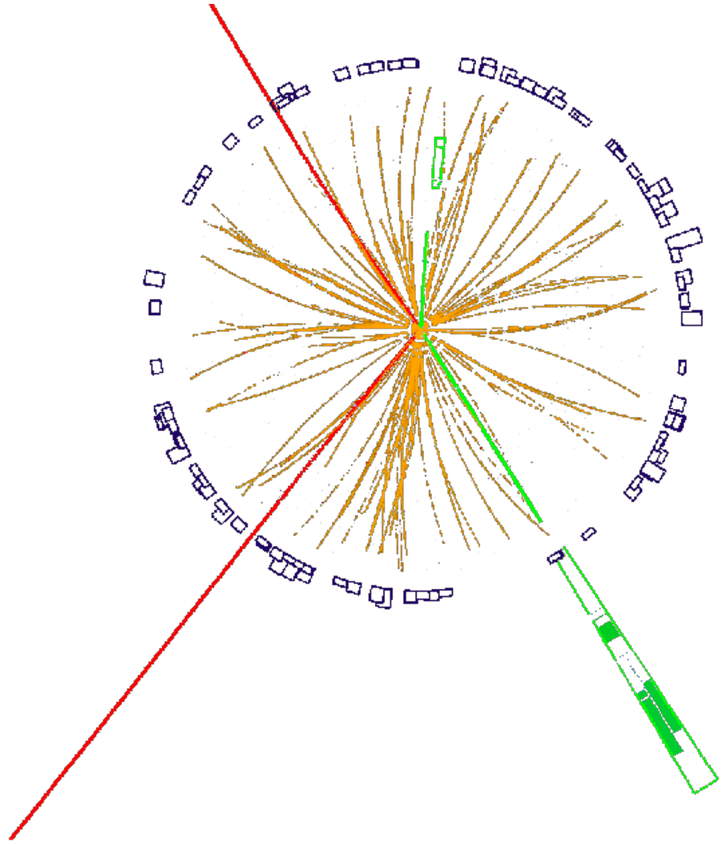
How to make any **neural network** Lorentz-Equivariant

Neural networks are fitting functions,

e.g. regression/classification $f_\theta : \mathbb{R}^n \to \mathbb{R}$

Deep neural networks:

- thrive with a lot of data;
- useful if data is hard to analyse, e.g correlations;
- high-dimensional structure.



How to make any **neural network** Lorentz-Equivariant

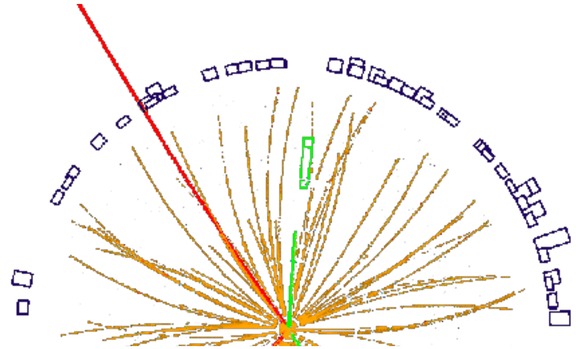
Neural networks are fitting functions,

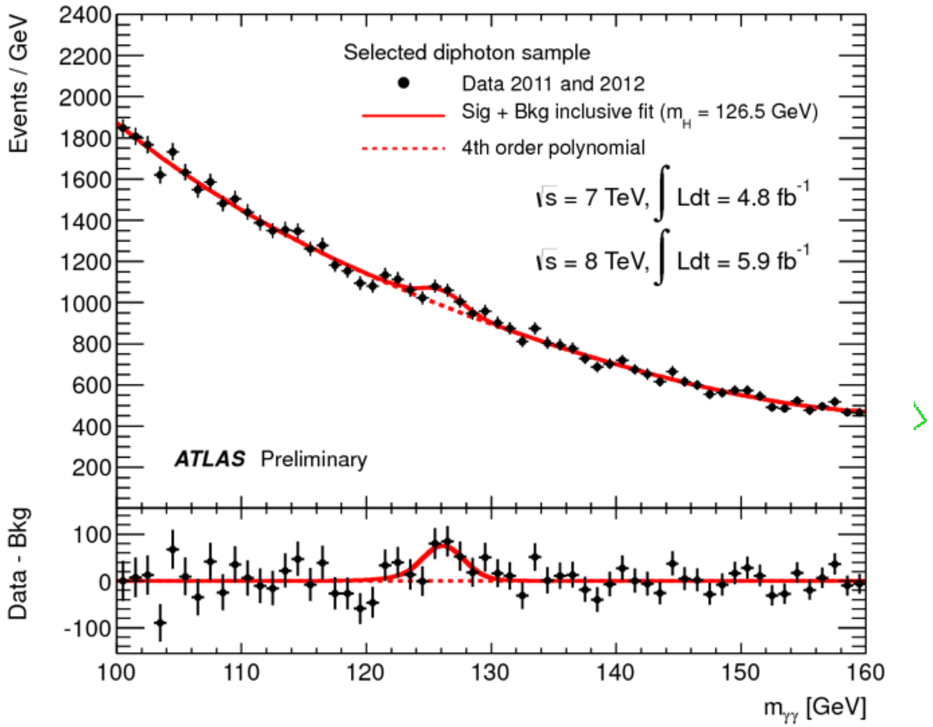
e.g. regression/classification $f_\theta : \mathbb{R}^n \to \mathbb{R}$

Deep neural networks:

- thrive with a lot of data;
- useful if data is hard to analyse, e.g correlations;
- high-dimensional structure.

- Already extensively used at the LHC;
- Accelerated the discovery of the Higgs boson.





How to make any neural network Lorentz-Equivariant

Equivariance:

$f(g \cdot x) = g \cdot f(x)$ for any $g \in G$

Invariance:

$g \cdot f(x) = f(x)$ for any $g \in G$

How to make any neural network Lorentz-Equivariant

Equivariance:

$$f(g \cdot x) = g \cdot f(x) \text{ for any } g \in G$$

Invariance:

$$g \cdot f(x) = f(x) \text{ for any } g \in G$$

Equivariance is appealing:

- introduce physics bias in neural networks;
- reduce optimisation error;
- exploit symmetries in geometric inputs.

How to make any neural network Lorentz-Equivariant

Equivariance:

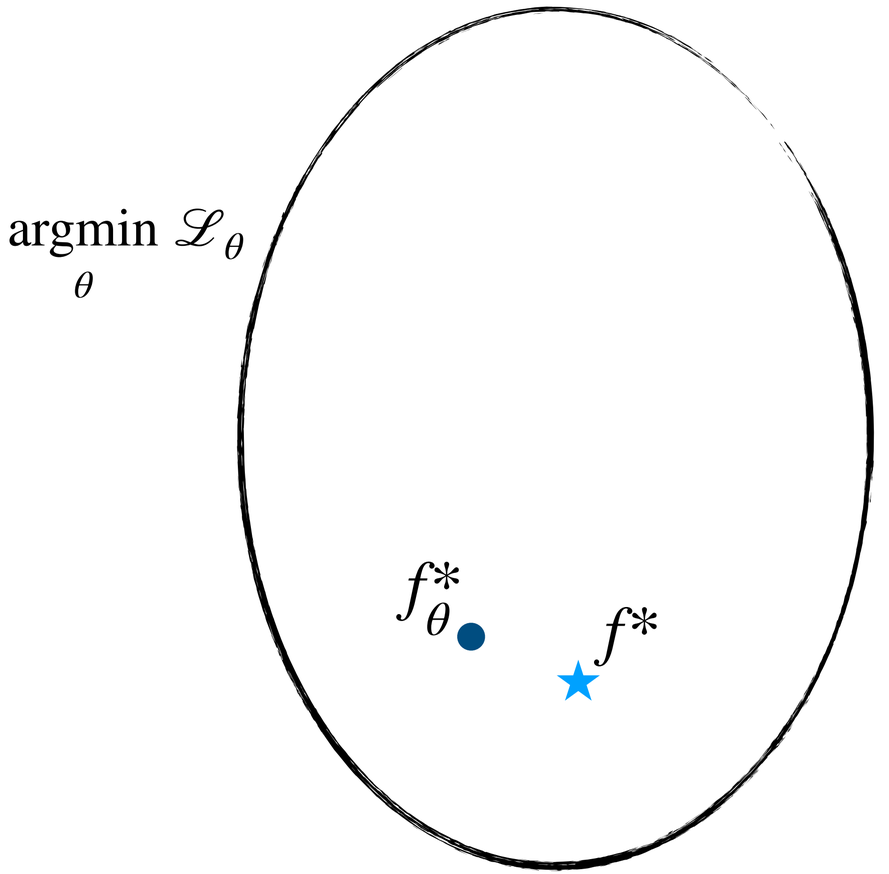
$$f(g \cdot x) = g \cdot f(x) \text{ for any } g \in G$$

Invariance:

$$g \cdot f(x) = f(x) \text{ for any } g \in G$$

Equivariance is appealing:

- introduce physics bias in neural networks;
- reduce optimisation error;
- exploit symmetries in geometric inputs.



How to make any neural network Lorentz-Equivariant

Equivariance:

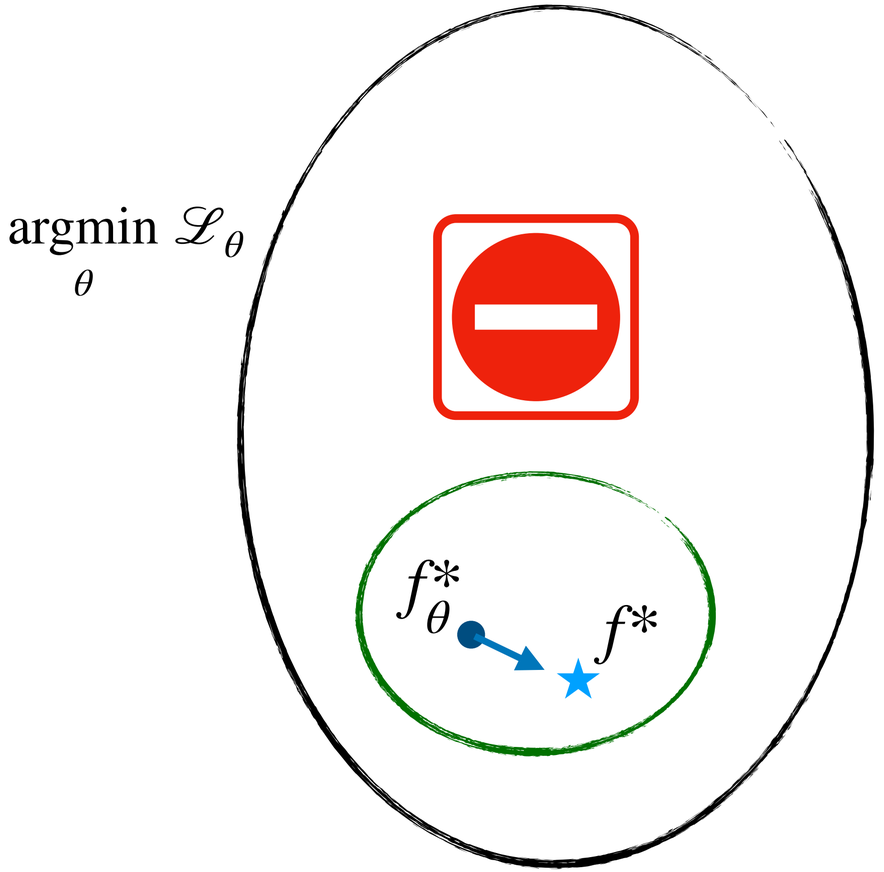
$$f(g \cdot x) = g \cdot f(x) \text{ for any } g \in G$$

Invariance:

$$g \cdot f(x) = f(x) \text{ for any } g \in G$$

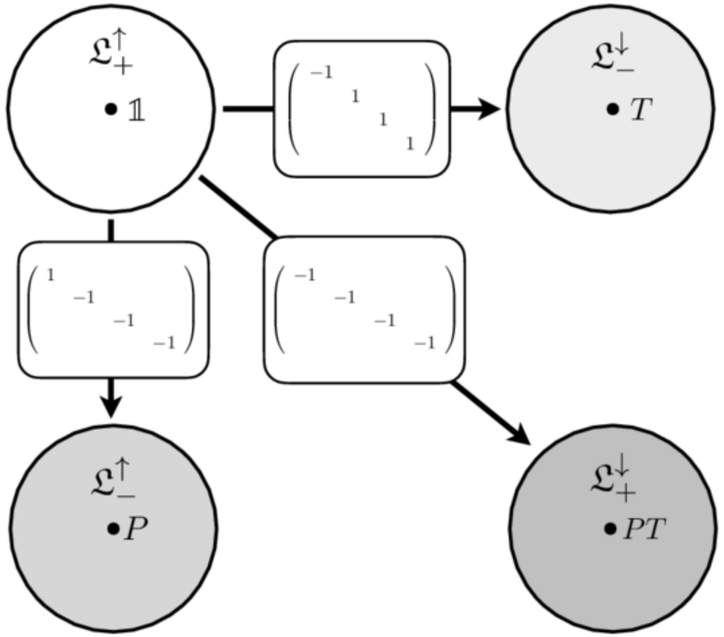
Equivariance is appealing:

- introduce physics bias in neural networks;
- reduce optimisation error;
- exploit symmetries in geometric inputs.



How to make any neural network Lorentz-Equivariant

- Particle physics data has a rich structure, typically we work with four-vectors x^μ
- Lorentz group $\longrightarrow$ $\mathrm{O}(1,3)$ but causality restricts our interest to $\mathrm{SO}^+(1,3)$



How to make any neural network Lorentz-Equivariant

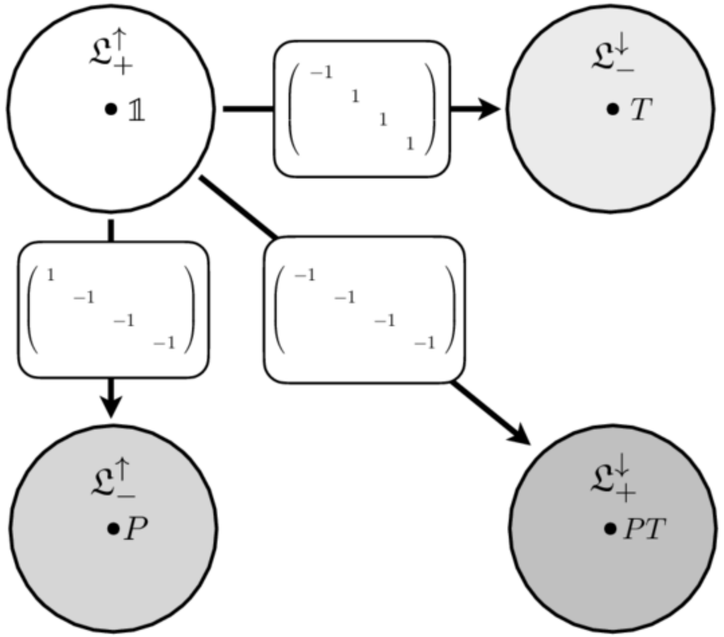
- Particle physics data has a rich structure, typically we work with four-vectors x^μ
- Lorentz group $\longrightarrow \mathrm{O}(1,3)$ but causality restricts our interest to $\mathrm{SO}^+(1,3)$
- Define the symmetric bilinear form, or "Minkwoski" product:

$$\langle x, y \rangle = x_0 y_0 - x_1 y_1 - x_2 y_2 - x_3 y_3$$

  or with the metric $g = \mathrm{diag}(1, -1, -1, -1)$ $\quad \langle x, y \rangle = g_{\mu\nu} x^\mu y^\nu$

- Condition on the elements of the Lorentz group reads

$$\Lambda^T g \Lambda = g$$



How to make any neural network Lorentz-Equivariant

- Local reference frames

$$L = \begin{pmatrix} — \; u_0^T g \; — \\ — \; u_1^T g \; — \\ — \; u_2^T g \; — \\ — \; u_3^T g \; — \end{pmatrix} \quad \overset{\Lambda}{\to} \quad L' = \begin{pmatrix} — \; u_0^T \Lambda^T g \; — \\ — \; u_1^T \Lambda^T g \; — \\ — \; u_2^T \Lambda^T g \; — \\ — \; u_3^T \Lambda^T g \; — \end{pmatrix} = L\Lambda^{-1}.$$

Ex. verify that

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

How to make any neural network Lorentz-Equivariant

- Local reference frames

$$L = \begin{pmatrix} — \; u_0^T g \; — \\ — \; u_1^T g \; — \\ — \; u_2^T g \; — \\ — \; u_3^T g \; — \end{pmatrix} \quad \overset{\Lambda}{\to} \quad L' = \begin{pmatrix} — \; u_0^T \Lambda^T g \; — \\ — \; u_1^T \Lambda^T g \; — \\ — \; u_2^T \Lambda^T g \; — \\ — \; u_3^T \Lambda^T g \; — \end{pmatrix} = L\Lambda^{-1}.$$

Ex. verify that

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

- Particle features in the local frames are Lorentz-invariant:

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

How to make any neural network Lorentz-Equivariant

- Local reference frames

$$L = \begin{pmatrix} — \; u_0^T g \; — \\ — \; u_1^T g \; — \\ — \; u_2^T g \; — \\ — \; u_3^T g \; — \end{pmatrix} \quad \overset{\Lambda}{\to} \quad L' = \begin{pmatrix} — \; u_0^T \Lambda^T g \; — \\ — \; u_1^T \Lambda^T g \; — \\ — \; u_2^T \Lambda^T g \; — \\ — \; u_3^T \Lambda^T g \; — \end{pmatrix} = L\Lambda^{-1}.$$

Ex. verify that

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

- Particle features in the local frames are Lorentz-invariant:

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

- Get Lorentz-equivariant output by applying a final transformation:

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

How to make any neural network Lorentz-Equivariant

- Local reference frames

$$L = \begin{pmatrix} — \; u_0^T g \; — \\ — \; u_1^T g \; — \\ — \; u_2^T g \; — \\ — \; u_3^T g \; — \end{pmatrix} \quad \overset{\Lambda}{\to} \quad L' = \begin{pmatrix} — \; u_0^T \Lambda^T g \; — \\ — \; u_1^T \Lambda^T g \; — \\ — \; u_2^T \Lambda^T g \; — \\ — \; u_3^T \Lambda^T g \; — \end{pmatrix} = L\Lambda^{-1}.$$

Ex. verify that

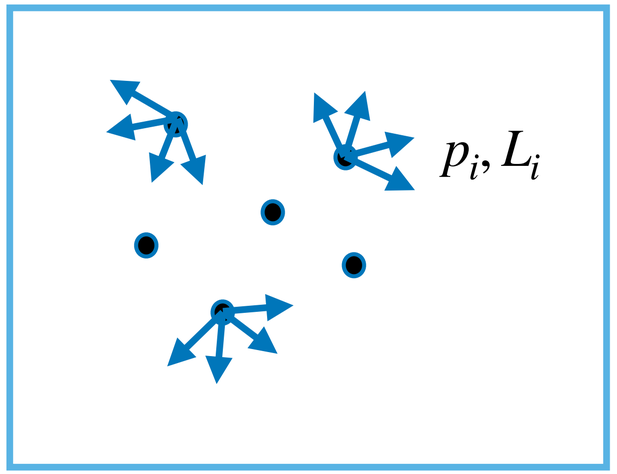
$$\sum_{\mu,\nu} {u_a}^{\mu} {u_b}^{\nu} g_{\mu\nu} = g_{ab} \; .$$

- Particle features in the local frames are Lorentz-invariant:

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

- Get Lorentz-equivariant output by applying a final transformation:

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



How to make any neural network Lorentz-Equivariant

Lorentz Local Canonicalization, or LloCa

- Equivariantly Predict three vectors for each particle*;
- Apply Gram-Schmidt and predict the 4th vector;
- Construct the local reference frame L;

lloca (catalan)
clueca(es), broody hen (en)

Applicable to any neural network which uses geometric input

Technical aspects:

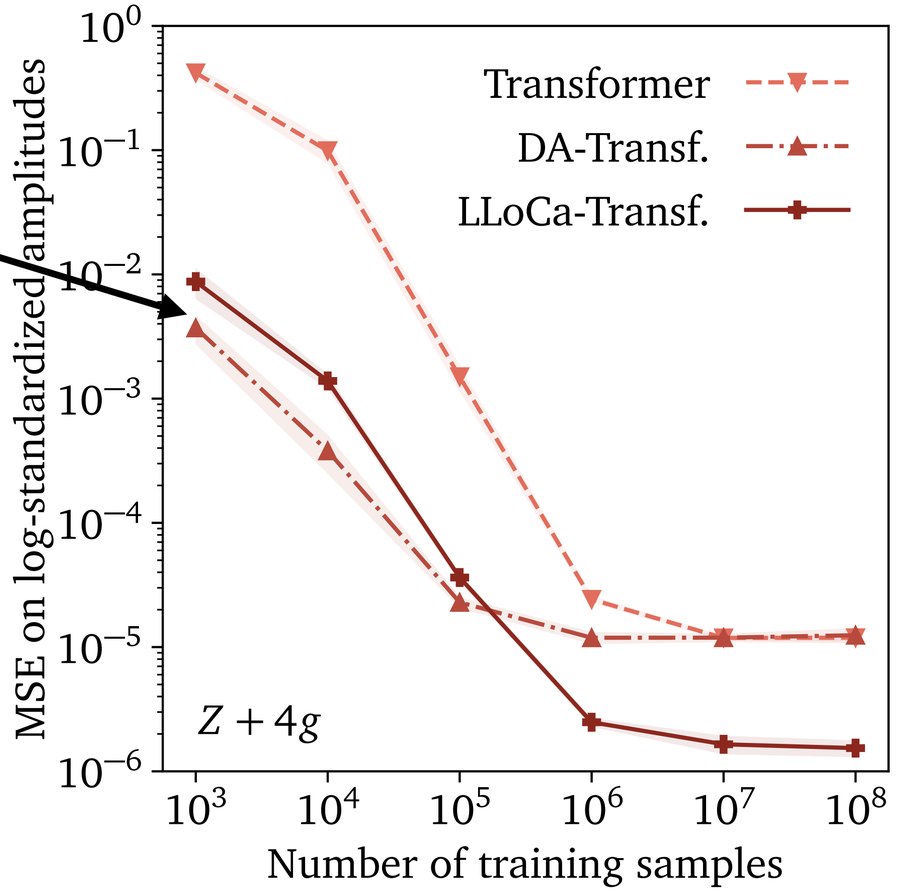
- Better to use a polar decomposition, $L = RB$;
- Predictor can be a (small) neural network;

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

How to make any neural network Lorentz-Equivariant

"Data augmentation" is a valid alternative if little data is available

Data augmentation: pick a random global reference frame $g \in G$



Regression of an "amplitude", a Lorentz-invariant scalar

Equivariance improves predictions

Thanks for your attention!