When the $M_{(L)}$ meets the P

Luigi Favaro

24.03.2026

Seminar @ MBI Institute

When the M meets the P

Monday 19 May — 2pm — CYCL 01

agenda.irmp.ucl.ac.be/e/m-p

Programme:

- Frank FERRARI (ULB)
- Ben PAGE (UGent)
- IRMP researchers
- Drink

Registration:

link above

Contacts:

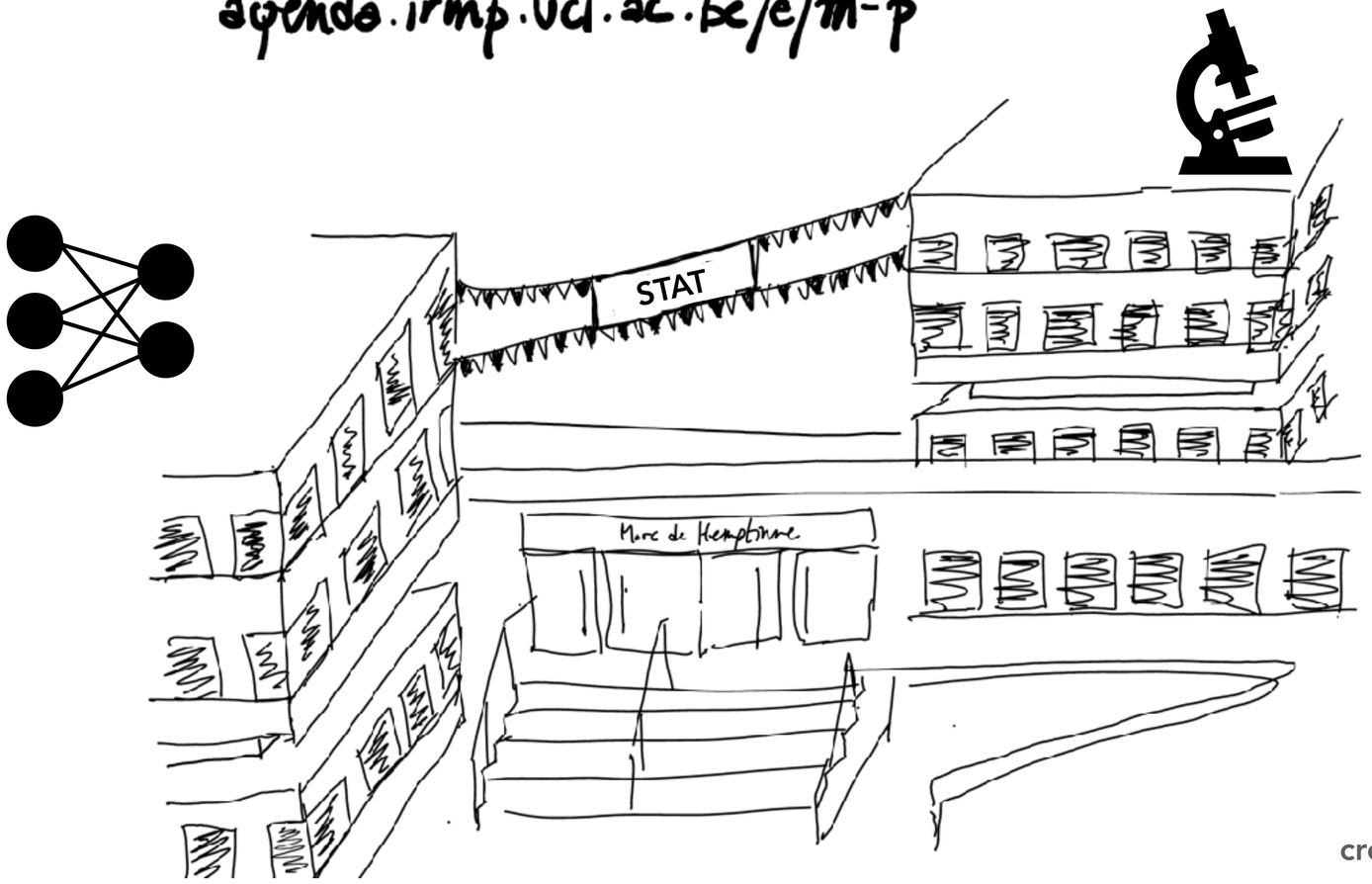
P. Bielievsky
G. Durieux

credits to Gauthier Durieux

When the M(L) meets the P

Monday 19 May — 2pm — CYCL 01

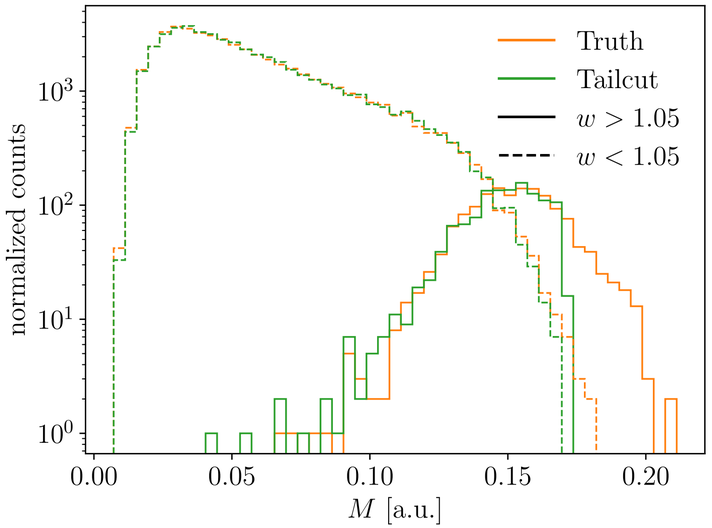
agenda.irmp.ucl.ac.be/e/m-p



credits to Gauthier Durieux

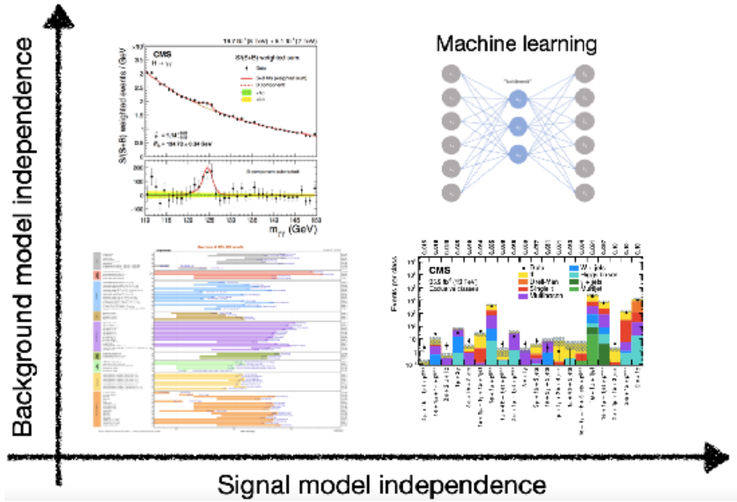
Disclaimer: this talk will be about…

… ML as a tool to explore high-dimensional spaces

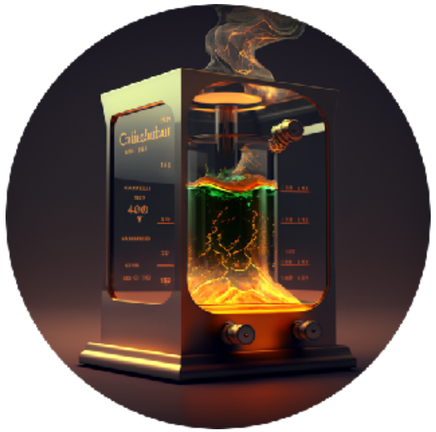


How to understand generative networks,

Das, LF, Heimel, Krause, Plehn, Shih, 2305.16774 [hep-ph]



credits to M. Krämer, cf. Karagiorgi, et al., 2112.03769 [hep-ph]



CaloChallenge, Krause C. LF, et al., 2410.21611 [hep-ph]

Disclaimer: this talk will not be about…

… ML as a supporting agent for scientists

David Shih, Claude Opus [2603.11164]

Learning to Unscramble:
Simplifying Symbolic Expressions via Self-Supervised Oracle Trajectories

David Shih*
NHETC, Dept. of Physics and Astronomy, Rutgers University, Piscataway, NJ 08854, USA
(Dated: March 13, 2026)

We present a new self-supervised machine learning approach for symbolic simplification of complex mathematical expressions. Training data is generated by scrambling simple expressions and recording the inverse operations, creating oracle trajectc
paths to reach them. A permutation-equivariant, t
on this data step-wise to predict the oracle action g
approach on two problems in high-energy physics: d
ing amplitude simplification. In both cases, our tra
rates across a wide range of difficulty levels, subst
on reinforcement learning and end-to-end regressi
and beam search, our model achieves a 100% full si
5-point gluon tree-level amplitudes in Yang-Mills th
terms.

Matthew D. Schwartz, Claude Opus [2601.02484]

Resummation of the C-Parameter Sudakov Shoulder
Using Effective Field Theory

Matthew D. Schwartz[1,2]
[1] *Department of Physics, Harvard University, Cambridge, MA 02138, USA*
[2] *Institute for Artificial Intelligence and Fundamental Interactions (IAIFI)*
schwartz@g.harvard.edu

AI Research Assistant: Claude Opus 4.5 (Anthropic)

January 7, 2026

Abstract

ibution in e^+e^- annihilation exhibits a kinematic shoulder at $C = 3/4$,
ıal states reach their maximum and a fourth parton is required to
ary generates large logarithms that must be resummed. Using soft-
ry, we derive a factorization theorem involving new jet and soft func-
parameter measurement, in which soft radiation contributes quadrat-
mentum. This quadratic structure explains the step discontinuity at
ıpute all ingredients at one loop, validate against Monte Carlo, and
+NLO results. Unlike thrust and heavy jet mass, the C-parameter
au pole, making momentum-space resummation straightforward. All
l analysis, and manuscript preparation were performed by Claude, an
by Anthropic, working under physicist supervision.

SciPost Physics Submission

MadAgents

Tilman Plehn[1,2], Daniel Schiller[1], and Nikita Schmal[1]

1 Institut für Theoretische Physik, Universität Heidelberg, Germany
2 Interdisciplinary Center for Scientific Computing (IWR), Universität Heidelberg, Germany

February 11, 2026

Abstract

We uncover an effective and communicative set of agents working with MadGraph. Agentic installation, learning-by-doing training, and user support provide easy access to state-of-the-art simulations and accelerate LHC research. We show in detail how MadAgents interact with inexperienced and advanced users, support a range of simulation tasks, and analyze results. In a second step, we illustrate how MadAgents automatize event generation and run an autonomous simulation campaign, starting from a pdf file of a paper.

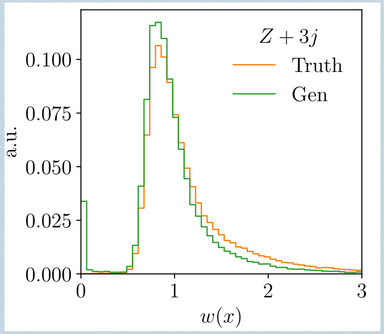
https://github.com/heidelberg-hepml/MadAgents

Plehn, Schiller, Schmal [2601.21015]

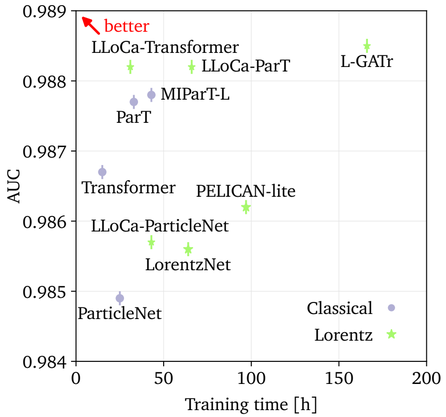
Being a “centaur scientist” means capitalizing on emerging AI technologies while insisting on scientific rigor and robustness

- Jesse Thaler

- Example: Approximating likelihood-ratios



$$w(x) = \frac{p(x)}{q(x)}$$

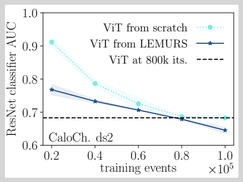


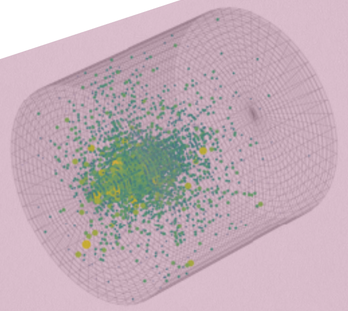
- Bittersweet lesson(?)

LLoCa

Embedding physics symmetries and structure in neural networks

- Extra: exploiting large data





- ML-emulators and their validation

Applications of generative networks for fast detector simulations

Approximating likelihood ratios

A simple example

Desiderata:

- Optimal test statistic;
- Invariant under coordinate reparametrization;
- Experimentally practical: profiling of nuisance parameters, composability, etc…

Approximating likelihood ratios

A simple example

Desiderata:

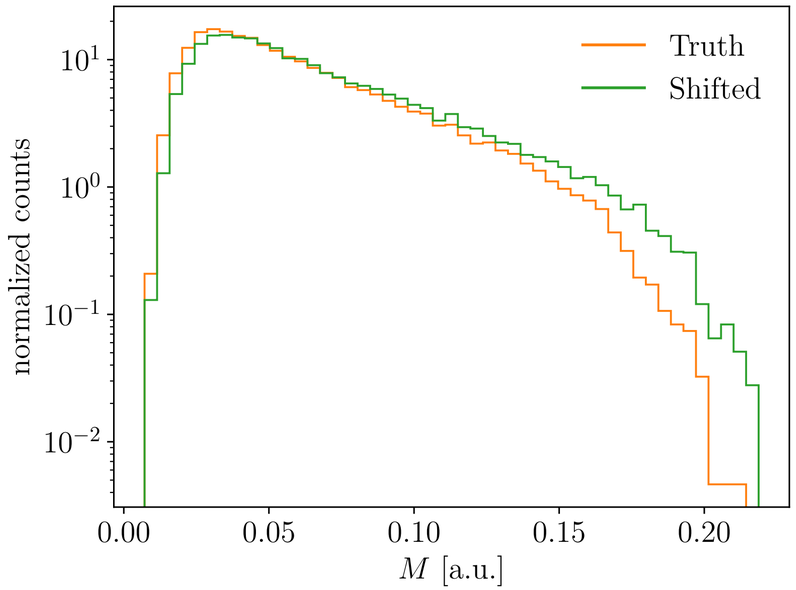
- Optimal test statistic (for a simple two hypotheses test);
- Invariant under coordinate reparametrization;
- Experimentally practical: profiling of nuisance parameters, composability, etc…

} **Likelihood ratio**

Given $p(x)$ and $q(x)$, there is a function $f(x)$ such that

$$\frac{p(x)}{q(x)} = f(x) \equiv w(x)$$ **phase space reweighting factor**

How to approximate $w(x)$?



Approximating likelihood ratios

LR: $w(x) = \dfrac{p(x)}{q(x)}$

A simple example

How to approximate $w(x)$?

We need training data $x_p \sim p(x)$, $x_q \sim q(x)$

Use a highly expressive function to obtain $w_\theta(x) \approx w(x)$

a neural network $f_\theta(x)$

Fit via stochastic gradient descent

$$\theta' = \theta - \eta \nabla_\theta \mathscr{L}$$

What is the proper loss function?

The one that satisfies the desired minimisation principle

$$\frac{\delta \mathscr{L}}{\delta f_\theta} = 0 \qquad \text{iff} \qquad f_\theta(x) = f(x)$$

$$\mathscr{L} = \langle \log f_\theta(x) \rangle_p + \langle f_\theta(x) - 1 \rangle_q$$

Monte Carlo average form

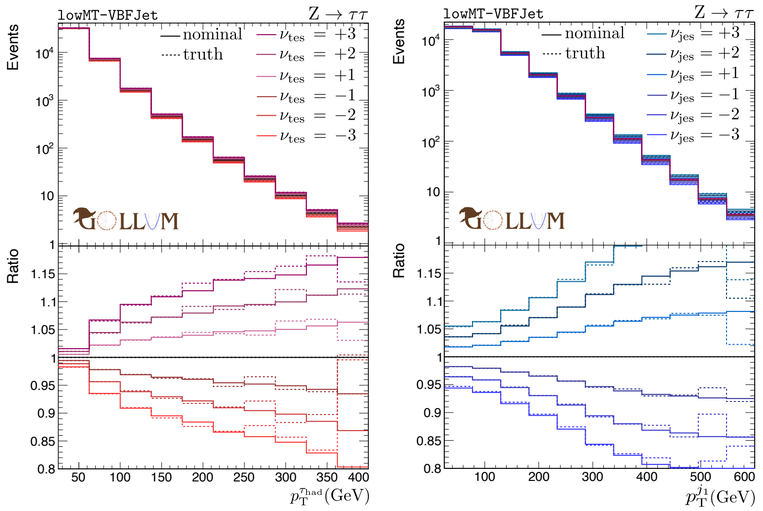
$$\mathscr{L} = \int \mathrm{d}x \ \left[p(x) \log f_\theta(x) - q(x)(f_\theta(x) - 1) \right]$$

Integral form $N \to \infty$

Changing the way we analyse data

The dimensionality of x was nowhere specified!

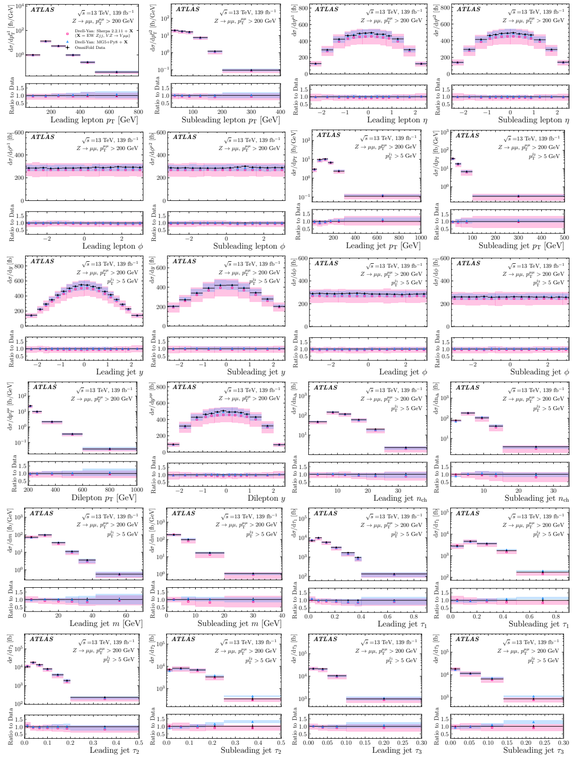
GOLLUM, Benato et al., 2505.05544 [hep-ph]



Unbinned measurements

simulation-based inference (SBI)

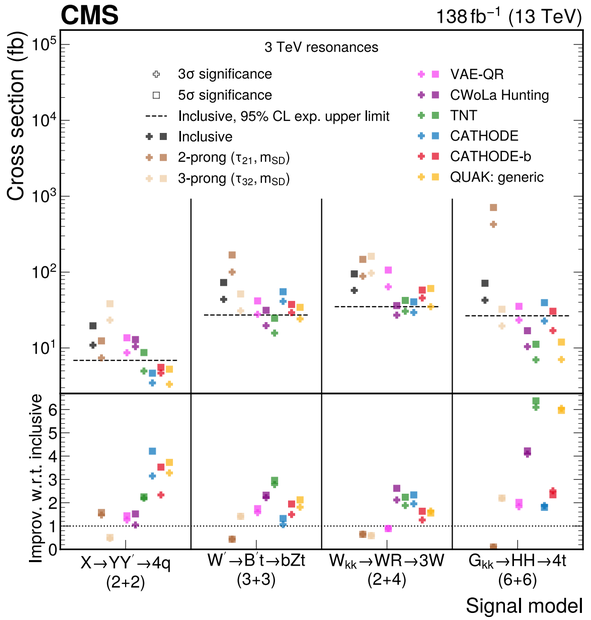
Z + jets **unfolding, ATLAS, 2405.20041 [hep-ex]**



Deconvolution

full phase space unfolding
of detector effects

dijet anomaly search, CMS, 2412.03747 [hep-ex]

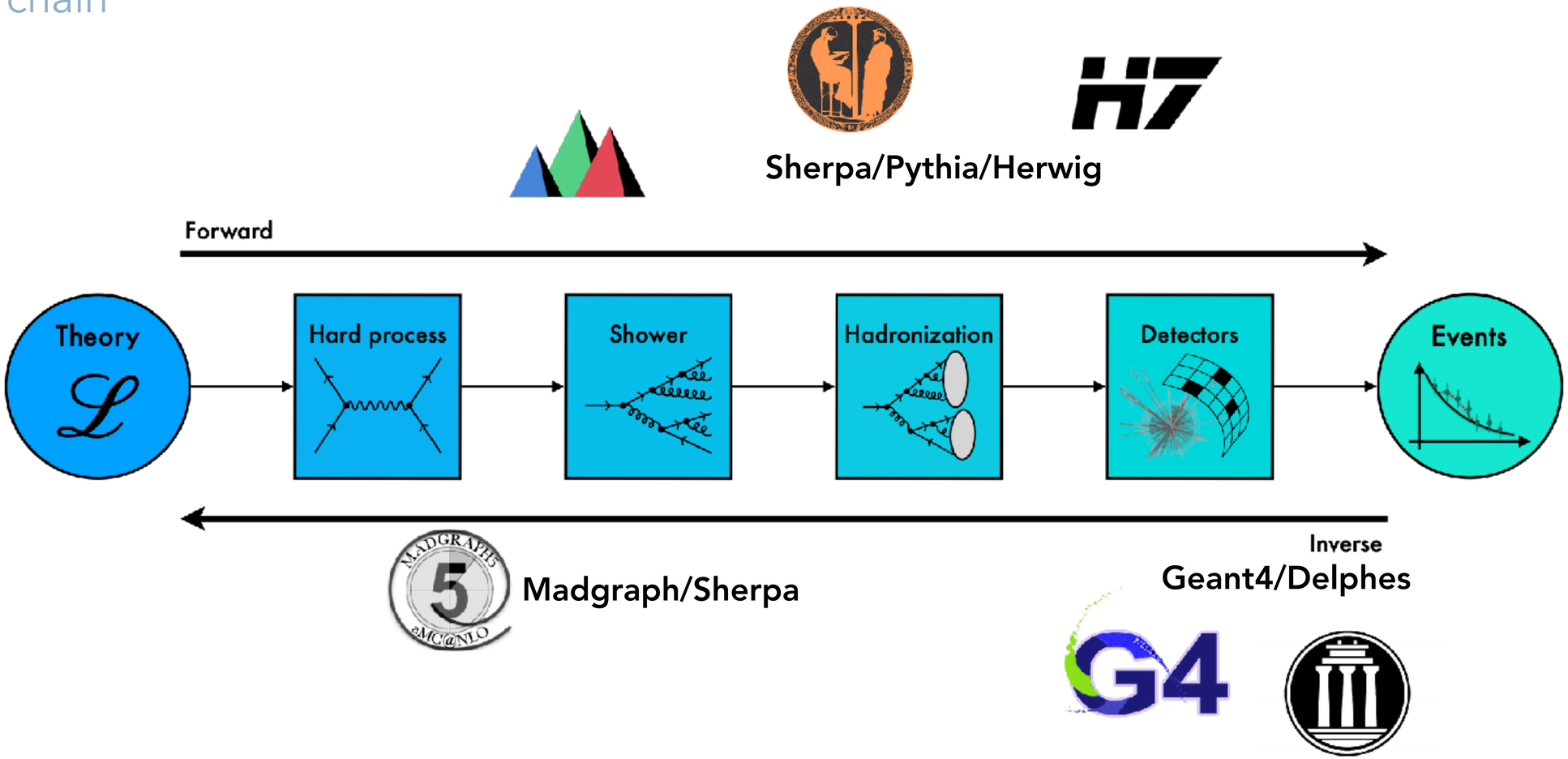


Anomaly detection

model-agnostic searches

HEP simulations

Simulation chain

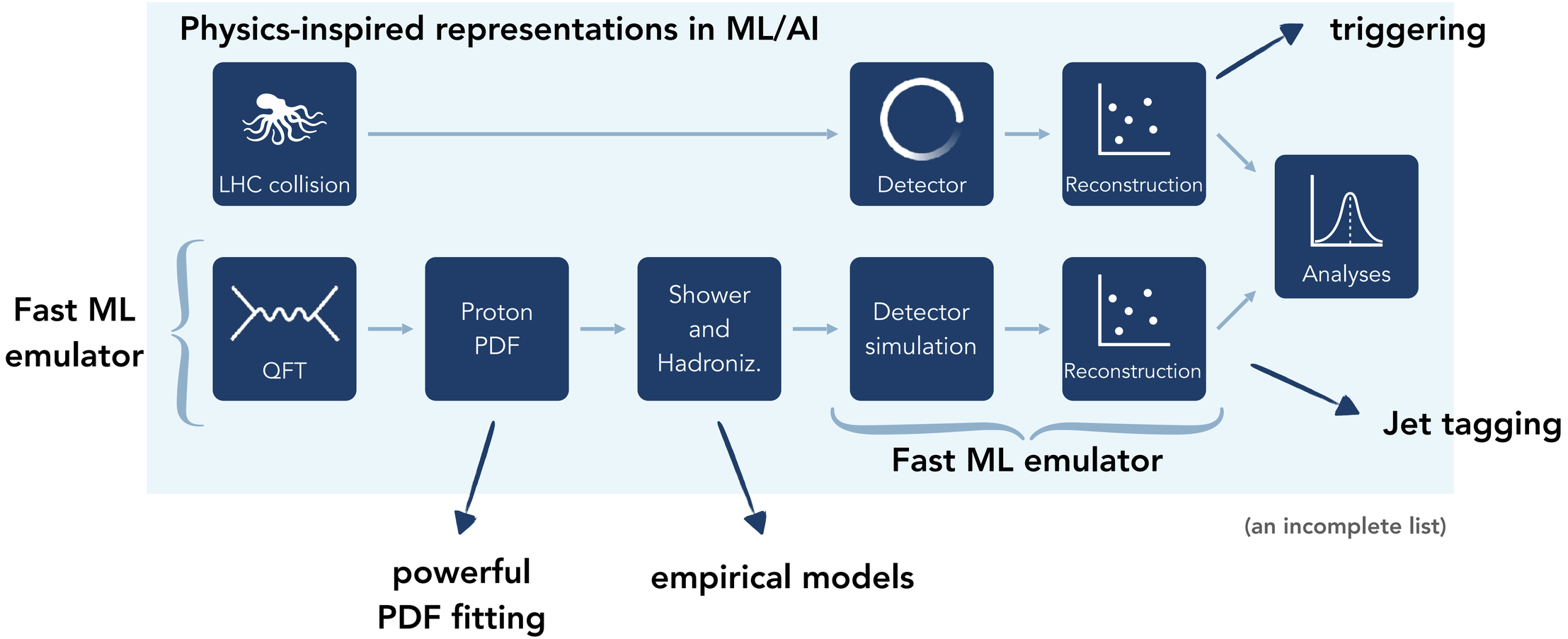


- MCMC samplers are the foundations of HEP simulations ⟶ provide large amount of trustable data

Maximising the LHC potential

The LHC provides high-dimensional and structured data:

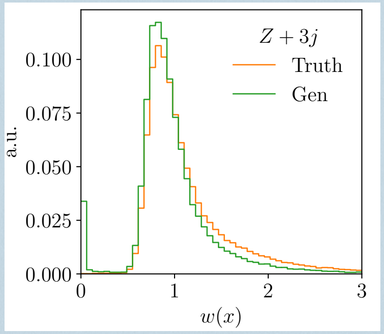
Machine learning allows to model complex correlations with minimal assumptions



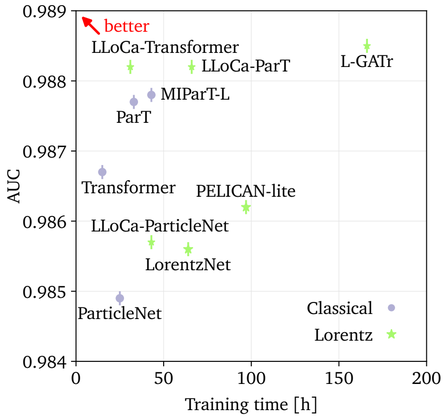
Being a “centaur scientist” means capitalizing on emerging AI technologies while insisting on scientific rigor and robustness

- Jesse Thaler

- Example: Approximating likelihood-ratios



$$w(x) = \frac{p(x)}{q(x)}$$

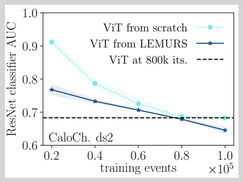


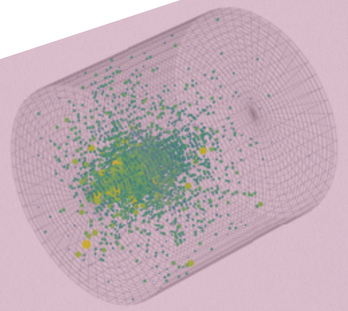
- Bittersweet lesson(?)

LLoCa

Embedding physics symmetries and structure in neural networks

- Extra: exploiting large data





- ML-emulators and their validation

Applications of generative networks for fast detector simulations

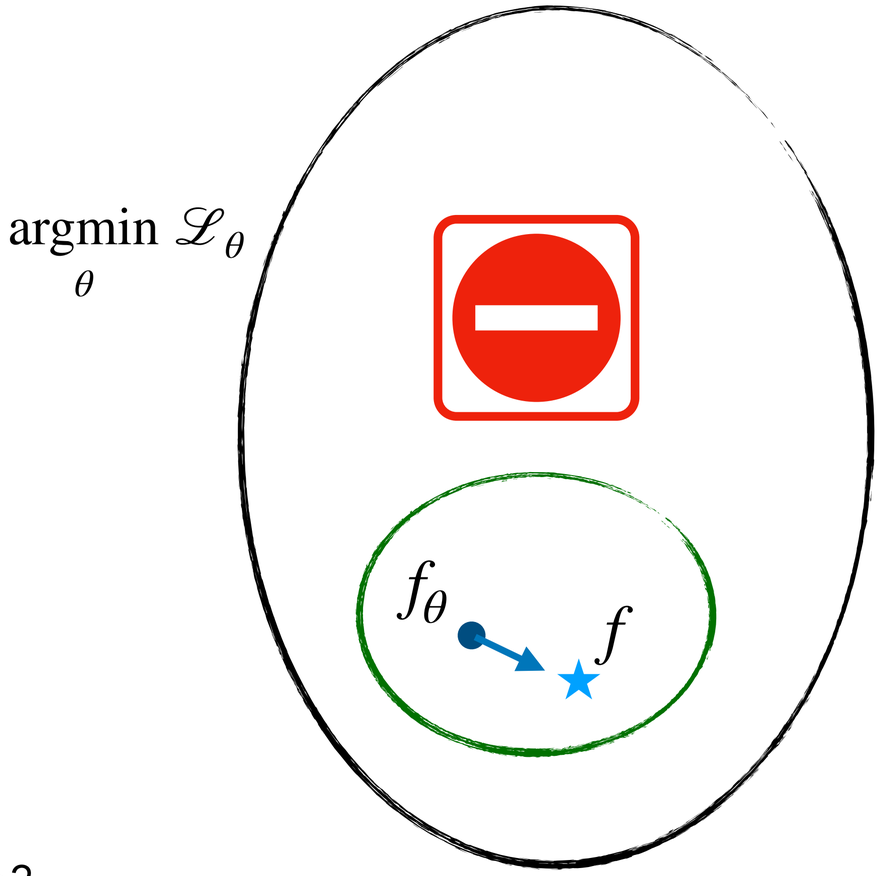
Equivariance in optimization problems

Equivariance:

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

Invariance:

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



Equivariance is appealing:

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

But: Is it necessary? Can I just scale up the dataset size?

The bittersweet lesson: Inductive biases (e.g. encoded symmetries) are not the best performing solutions

Kyle Cranmer, [link]

Lorentz-equivariance and data structure

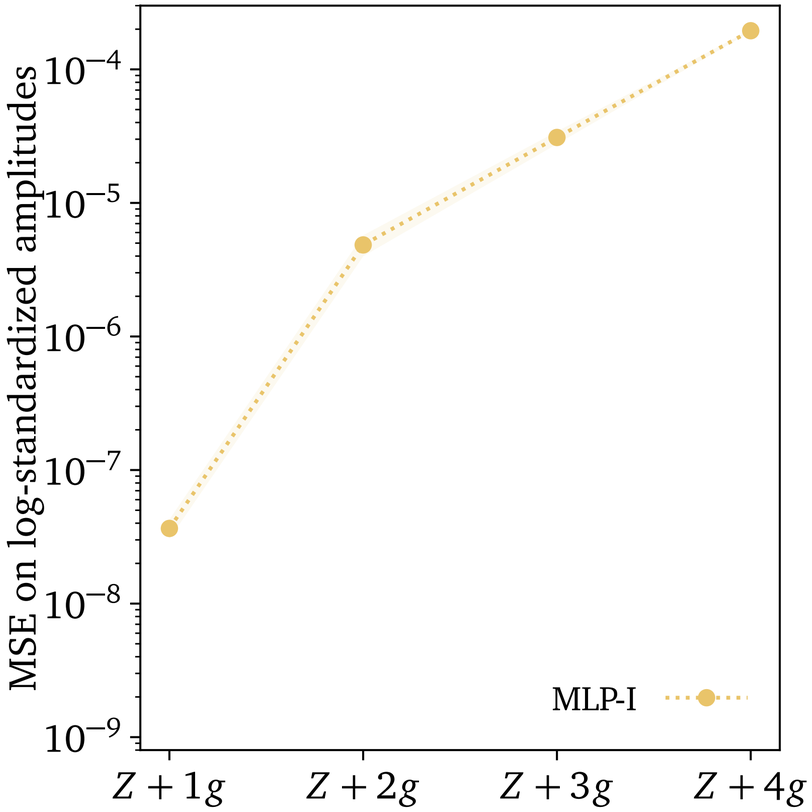
A regression example

Task: Predict the value of the squared matrix element of scattering amplitudes

Input: Four-vectors ⟶ best represented as a point cloud

Output: Lorentz-invariant quantity

MLP-I: simple fully-connected network with dot products



Lorentz-equivariance and data structure

A regression example

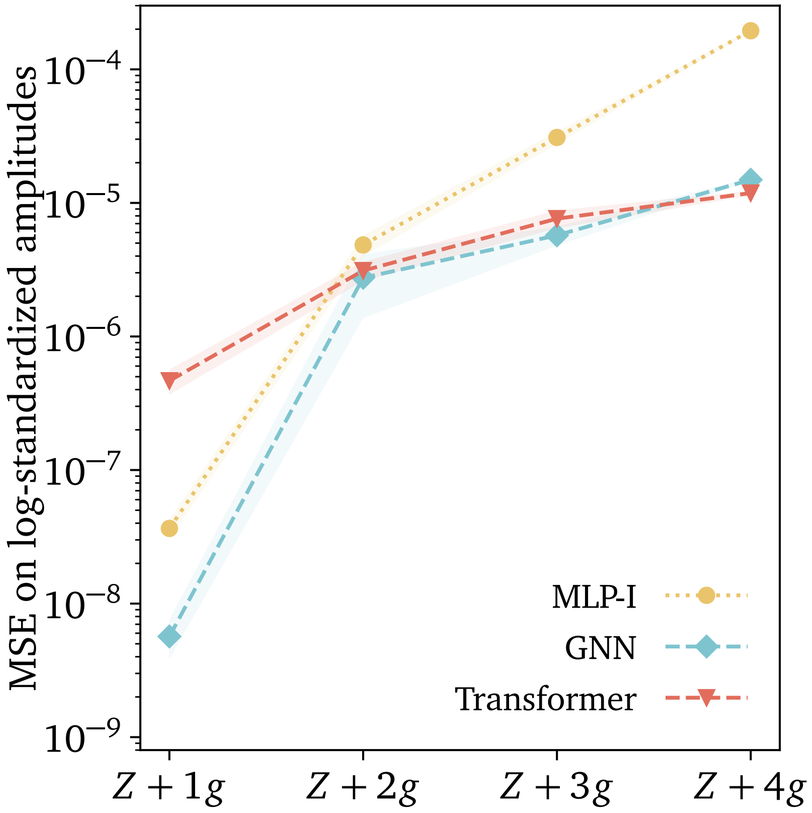
Task: Predict the value of the squared matrix element of scattering amplitudes

Input: Four-vectors ⟶ best represented as a point cloud

Output: Lorentz-invariant quantity

GNN & Transformer:

permutation invariance and variable-length input



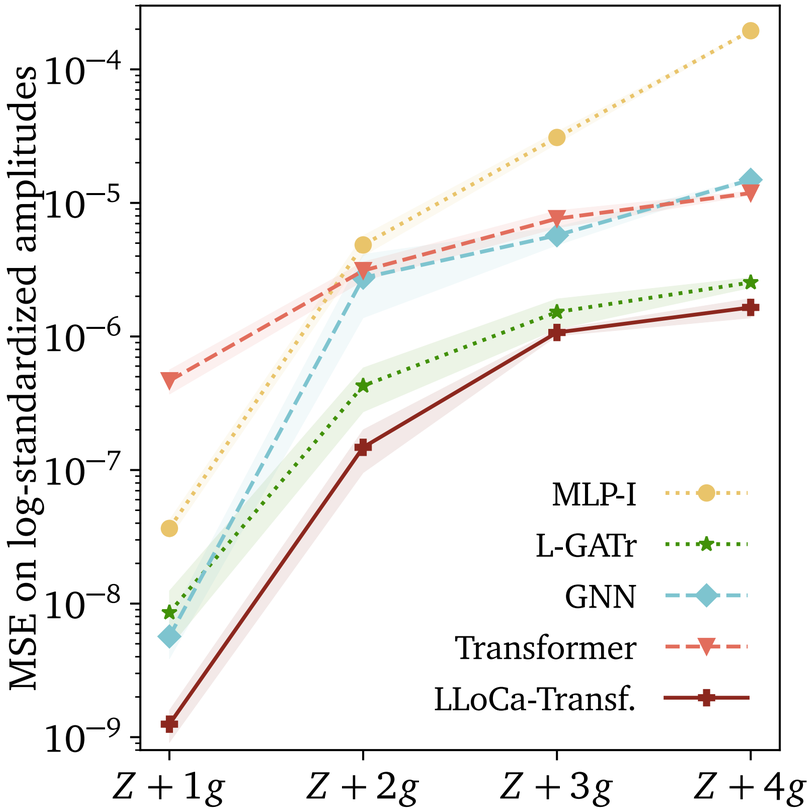
Lorentz-equivariance and data structure

A regression example

Task: Predict the value of the squared matrix element of scattering amplitudes

Input: Four-vectors ⟶ best represented as a point cloud

Output: Lorentz-invariant quantity



L-GATr & LLoCa: transformers + exact Lorentz-equivariance

LLoCa: efficient and flexible *canonicalization approach*

How to make any network Lorentz-equivariant, 2505.20280 [stat.ML]
Lorentz-equivariance without limitations, 2508.14898 [hep-ph]

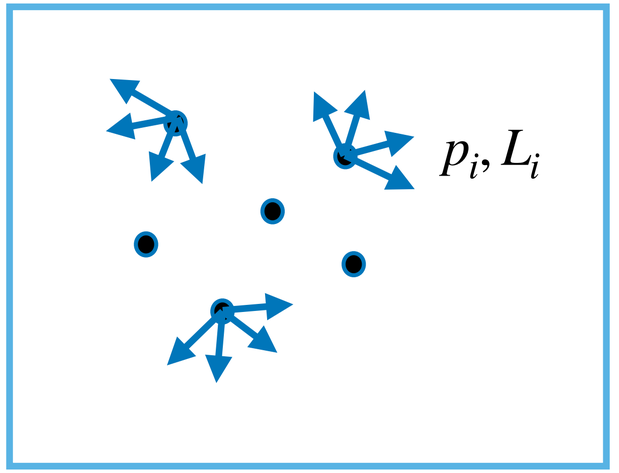
See also
L-GATr: Lorentz-equivariance from specialised layers
Spinner et al. 2411.00446 [hep-ph]

Lorentz Local Frames

- Our goal is to define a (local) reference frame L such that $x_L \xrightarrow{\Lambda} x'_L = x_L$ $\qquad x_L = Lx$

- Let's take L such that $L \xrightarrow{\Lambda} L' = L\Lambda^{-1}$. Then,

$$x'_L = L'x' = L\Lambda^{-1}\Lambda x = Lx = x_L$$

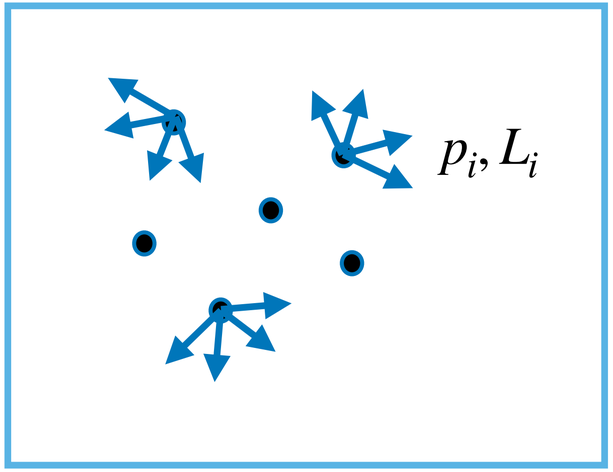


- Particle features in the local frames are Lorentz-invariant!
  (also true for higher-order representations)

- Output features in the global frame are Lorentz-equivariant:

$$y \xrightarrow{\Lambda} y' = L'^{-1}y'_L = \Lambda L^{-1}y_L = \Lambda L^{-1}y_L = \Lambda y$$

Constructing a local frame



- Take four Lorentz vectors u_i and define L as

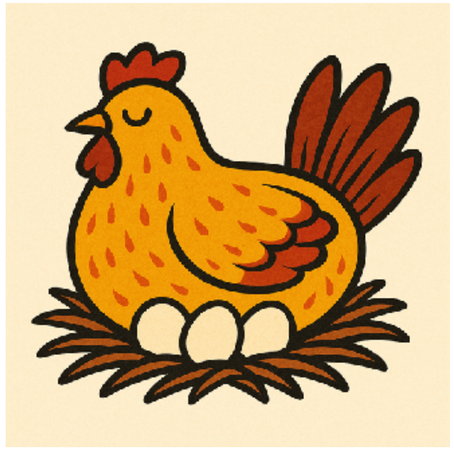
$$L = \begin{pmatrix} — & u_0^T g & — \\ — & u_1^T g & — \\ — & u_2^T g & — \\ — & u_3^T g & — \end{pmatrix} \quad \overset{\Lambda}{\to} \quad L' = \begin{pmatrix} — & u_0^T \Lambda^T g & — \\ — & u_1^T \Lambda^T g & — \\ — & u_2^T \Lambda^T g & — \\ — & u_3^T \Lambda^T g & — \end{pmatrix} = L\Lambda^{-1}.$$

- The definition of covectors imposes the correct transformation rule by construction

- L is also called a “tetrad” or “vierbein” in general relativity.

How to define a set of Lorentz vectors per particle?

Lorentz Local Canonicalization



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

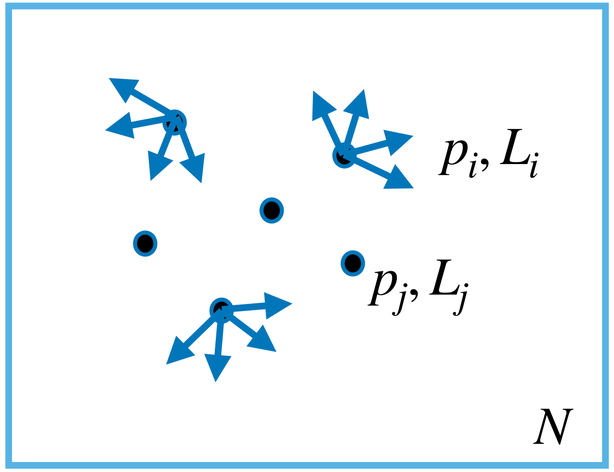
LLoCa

Lorentz Local Canonicalization

- Equivariantly predict three Lorentz vectors for each particle;

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

Use a small GNN φ which takes invariants as input



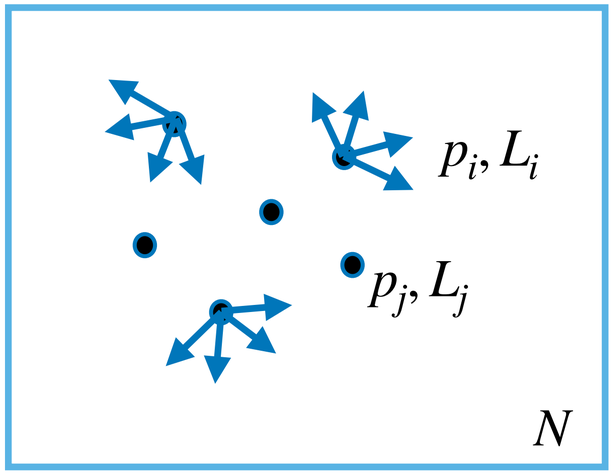
Lorentz Local Canonicalization

- Equivariantly predict three Lorentz vectors for each particle;

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

- Use the Gram-Schmidt orthogonalisation and predict the 4th vector;

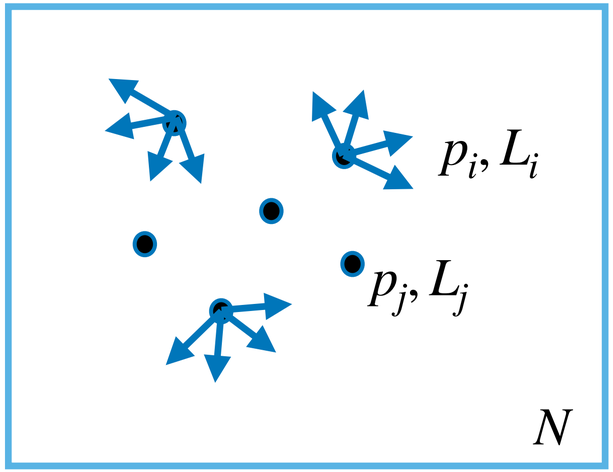
$$v_0, v_1, v_2 \xrightarrow{\mathrm{GS}} u_0, u_1, u_2 \qquad\qquad u_3^\mu = \sum_{\nu,\rho,\sigma,\kappa} g^{\mu\nu} \epsilon_{\nu\rho\sigma\kappa} u_0^\rho u_1^\sigma u_2^\kappa.$$



Lorentz Local Canonicalization

- Equivariantly predict three Lorentz vectors for each particle;

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



- Use the Gram-Schmidt orthogonalisation and predict the 4th vector;

$$v_0, v_1, v_2 \xrightarrow{\text{GS}} u_0, u_1, u_2 \qquad\qquad u_3^\mu = \sum_{\nu,\rho,\sigma,\kappa} g^{\mu\nu} \epsilon_{\nu\rho\sigma\kappa} u_0^\rho u_1^\sigma u_2^\kappa.$$

- Construct the local reference frame L;

$$L = \begin{pmatrix} — & u_0^T g & — \\ — & u_1^T g & — \\ — & u_2^T g & — \\ — & u_3^T g & — \end{pmatrix}$$

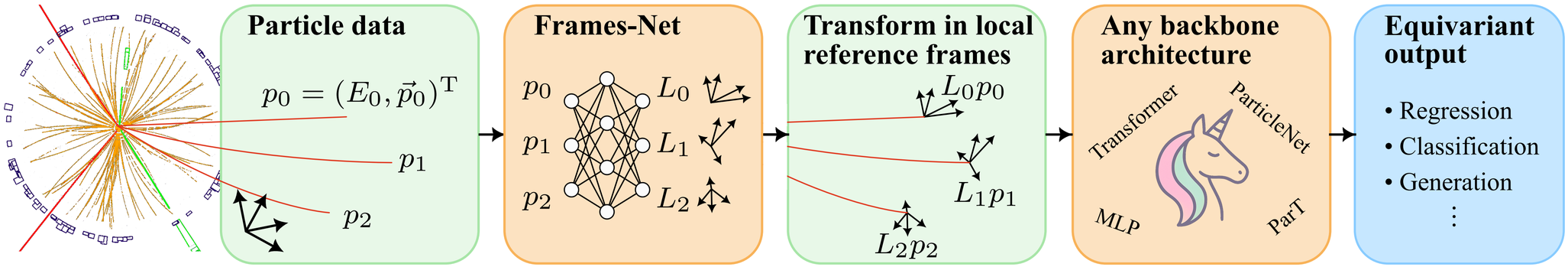
Technical aspects:

- Better to use a polar decomposition, $L = RB$, where R is a pure rotation and B a pure boost

Lorentz Local Canonicalization

- Equivariantly predict three Lorentz vectors for each particle;
- Use the Gram-Schmidt orthogonalisation and predict the 4th vector;
- Construct the local reference frame L;

I did not mention the backbone network anywhere!
It can be applied to any network that takes geometric inputs



Lorentz Local Canonicalization

Message-passing

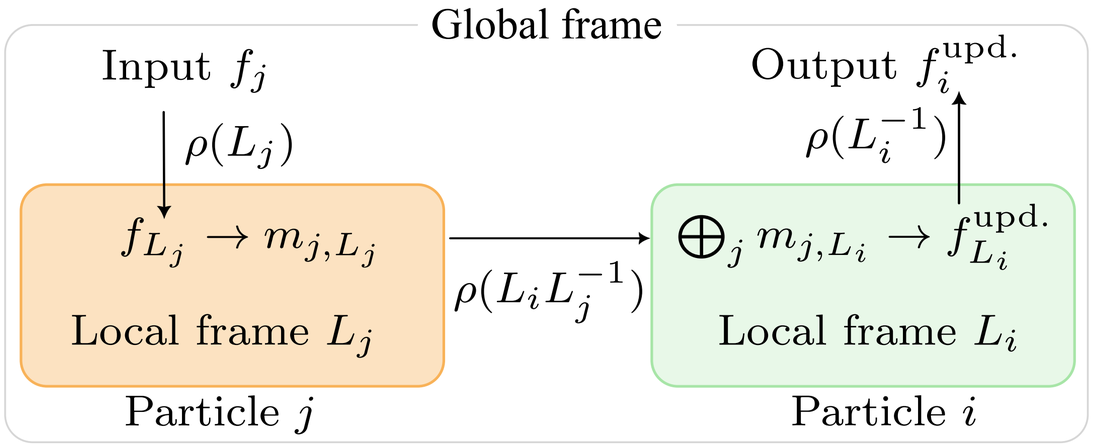
- Message-passing architecture benefit the most from local frames

$$f_i^{\text{updated}} = \psi\left(f_i, \sum_{j=1}^{N} \phi(m_j)\right) .$$

- General transformation rule reads
  - m_i are the messages at node i
  - f_i are the particle features
  - it applies to both GNNs and transformers

- The LLoCa message-passing becomes

$$\begin{aligned} f_{i,L_i}^{\text{updated}} &= \psi\left(f_{i,L_i}, \sum_{j=1}^{N} \phi\left(m_{j,L_i}\right)\right) \\ &= \psi\left(f_{i,L_i}, \sum_{j=1}^{N} \phi\left(\rho_m(L_i L_j^{-1}) m_{j,L_j}\right)\right) . \end{aligned}$$



Messages carry their own representation and are transformed accordingly

Tagging results

JetClass

Network	Accuracy	AUC	Time	FLOPs	Memory	Parameters
PFN	0.772	0.9714	3h	3M	1.6G	86k
P-CNN	0.809	0.9789	3h	12M	3.0G	354k
MIParT-L	0.861	0.9878	47h	225M	53.6G	2380k
LorentzNet*	0.847	0.9856	57h	676M	20.5G	223k
L-GATr*	0.866	0.9885	180h	2060M	19.0G	1079k
ParticleNet	0.844	0.9849	27h	413M	16.5G	366k
LLoCa-ParticleNet*	0.848	0.9857	41h	517M	23.5G	385k
ParT	0.861	0.9877	38h	211M	13.3G	2141k
LLoCa-ParT*	0.864	0.9882	67h	315M	19.9G	2160k
Transformer	0.855	0.9867	17h	210M	2.3G	1979k
LLoCa-Transformer*	0.864	0.9882	33h	301M	6.9G	1998k

Table 1: Performance and computational cost for taggers on the JetClass dataset. Lorentz-equivariant networks are denoted with an asterisk.

Tagging results

JetClass

Network	Accuracy	AUC	Time	FLOPs	Memory	Parameters
PFN	0.772	0.9714	3h	3M	1.6G	86k
P-CNN	0.809	0.9789	3h	12M	3.0G	354k
MIParT-L	0.861	0.9878	47h	225M	53.6G	2380k
LorentzNet*	0.847	0.9856	57h	676M	20.5G	223k
L-GATr*	0.866	0.9885	180h	2060M	19.0G	1079k
ParticleNet	0.844	0.9849	27h	413M	16.5G	366k
LLoCa-ParticleNet*	0.848	0.9857	41h	517M	23.5G	385k
ParT	0.861	0.9877	38h	211M	13.3G	2141k
LLoCa-ParT*	0.864	0.9882	67h	315M	19.9G	2160k
Transformer	0.855	0.9867	17h	210M	2.3G	1979k
LLoCa-Transformer*	0.864	0.9882	33h	301M	6.9G	1998k

Table 1: Performance and computational cost for taggers on the JetClass dataset. Lorentz-equivariant networks are denoted with an asterisk.

Lorentz-equivariant networks provide stronger accuracy and AUC

Tagging results

JetClass

Network	Accuracy	AUC	Time	FLOPs	Memory	Parameters
LorentzNet*	0.847	0.9856	57h	676M	20.5G	223k
ParticleNet	0.844	0.9849	27h	413M	16.5G	366k
LLoCa-ParticleNet*	0.848	0.9857	41h	517M	23.5G	385k
ParT	0.861	0.9877	38h	211M	13.3G	2141k
LLoCa-ParT*	0.864	0.9882	67h	315M	19.9G	2160k
Transformer	0.855	0.9867	17h	210M	2.3G	1979k
LLoCa-Transformer*	0.864	0.9882	33h	301M	6.9G	1998k

Table 1: Performance and computational cost for taggers on the JetClass dataset. Lorentz-equivariant networks are denoted with an asterisk.

LLoCa improves the performance of others well-established networks

Tagging results

JetClass

Network	Accuracy	AUC	Time	FLOPs	Memory	Parameters
L-GATr*	0.866	0.9885	180h	2060M	19.0G	1079k
Transformer	0.855	0.9867	17h	210M	2.3G	1979k
LLoCa-Transformer*	0.864	0.9882	33h	301M	6.9G	1998k

Table 1: Performance and computational cost for taggers on the JetClass dataset. Lorentz-equivariant networks are denoted with an asterisk.

Large reduction of costs compared to networks with specialised layers

Tagging results

JetClass

Network	Accuracy	AUC	Time	FLOPs	Memory	Parameters
ParT	0.861	0.9877	38h	211M	13.3G	2141k
LLoCa-ParT*	0.864	0.9882	67h	315M	19.9G	2160k
LLoCa-Transformer*	0.864	0.9882	33h	301M	6.9G	1998k

Table 1: Performance and computational cost for taggers on the JetClass dataset. Lorentz-equivariant networks are denoted with an asterisk.

Better than ParT with smaller overheads (no interaction matrices needed)

Symmetry-breaking effects

Tagging score is not a Lorentz-invariant quantity

Residual symmetry	Accuracy Arch.	Accuracy Input	AUC Arch.	AUC Input
Non-equivariant	0.855	0.864	0.9867	0.9882
$SO(2)_{beam}$	0.862	0.864	0.9878	0.9882
$SO^+(1,1)_{beam} \times SO(2)_{beam}$	0.863	0.864	0.9881	0.9882
$SO(3)$	0.859	0.860	0.9875	0.9876
$SO^+(1,3)$ (Lorentz)	0.856		0.9870	

Architecture: “hard” symmetry-breaking

the network is constrained to respect only the subgroup $\mathscr{R}$

Input: “soft” symmetry-breaking

add spurions to the point cloud and invariant scalar inputs

e.g.

$v_0 = (1,0,0,0)$ reference time-like vector

$v_0 = (1,0,0,0)$, $v_1 = (0,0,0,1)$

time and beam direction vectors

Symmetry-breaking effects

Tagging score is not a Lorentz-invariant quantity

Architecture: "hard" symmetry-breaking

the network is constrained to respect only the subgroup $\mathscr{R}$

	Accuracy		AUC	
Residual symmetry	Arch.	Input	Arch.	Input
Non-equivariant	0.855	0.864	0.9867	0.9882
$\mathrm{SO}(2)_{\mathrm{beam}}$	0.862	0.864	0.9878	0.9882
$\mathrm{SO}^+(1,1)_{\mathrm{beam}} \times \mathrm{SO}(2)_{\mathrm{beam}}$	0.863	0.864	0.9881	0.9882
$\mathrm{SO}(3)$	0.859	0.860	0.9875	0.9876
$\mathrm{SO}^+(1,3)$ (Lorentz)	0.856		0.9870	

Input wins: The network can decide to ignore the spurions and restore the full symmetry

Input: "soft" symmetry-breaking

add spurions to the point cloud and invariant scalar inputs

e.g.

$v_0 = (1,0,0,0)$ reference time-like vector

$v_0 = (1,0,0,0)$, $v_1 = (0,0,0,1)$

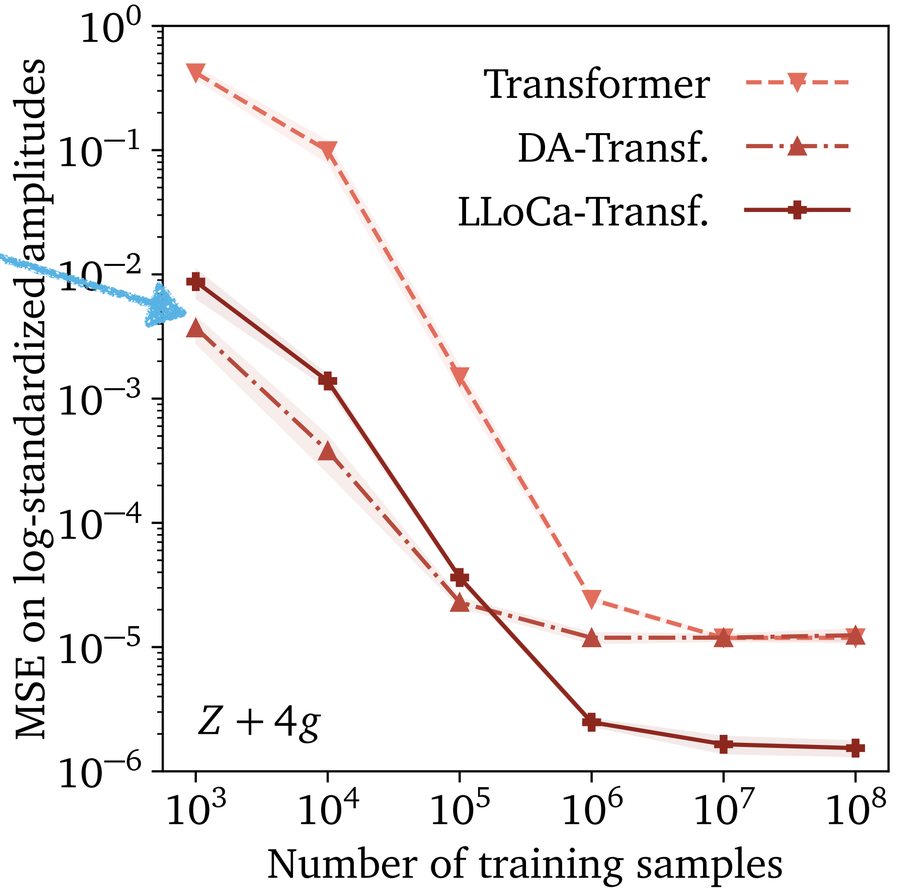
time and beam direction vectors

Not a bittersweet lesson

A regression example

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

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

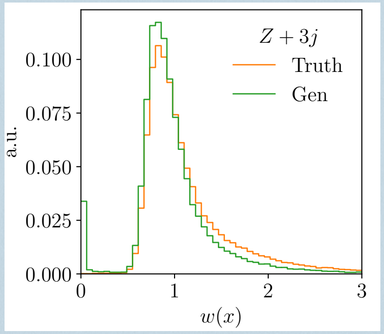


Equivariance improves predictions

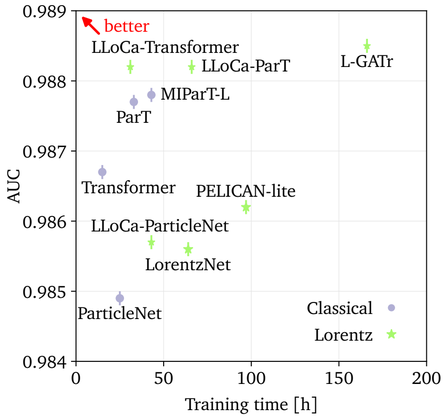
Being a “centaur scientist” means capitalizing on emerging AI technologies while insisting on scientific rigor and robustness

- Jesse Thaler

- Example: Approximating likelihood-ratios



$$w(x) = \frac{p(x)}{q(x)}$$

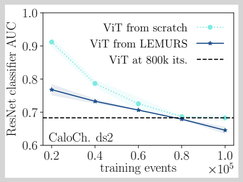


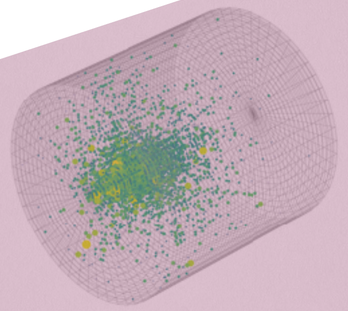
- Bittersweet lesson(?)

LLoCa

Embedding physics symmetries and structure in neural networks

- Extra: exploiting large data



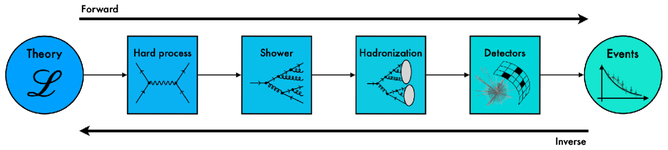


- ML-emulators and their validation

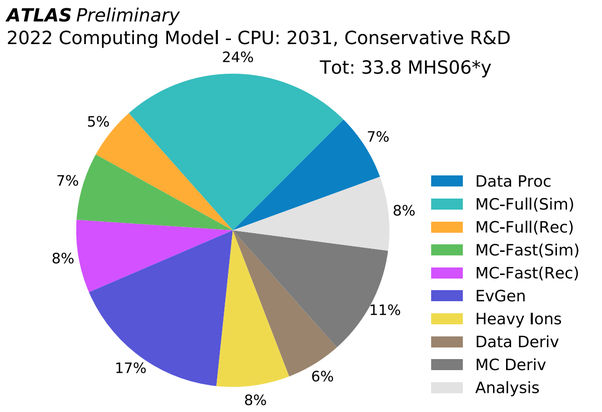
Applications of generative networks for fast detector simulations

HEP simulations

Challenges



- High-Luminosity LHC will produce unmatched amount of data;
- simulations need to match the collected statistics.

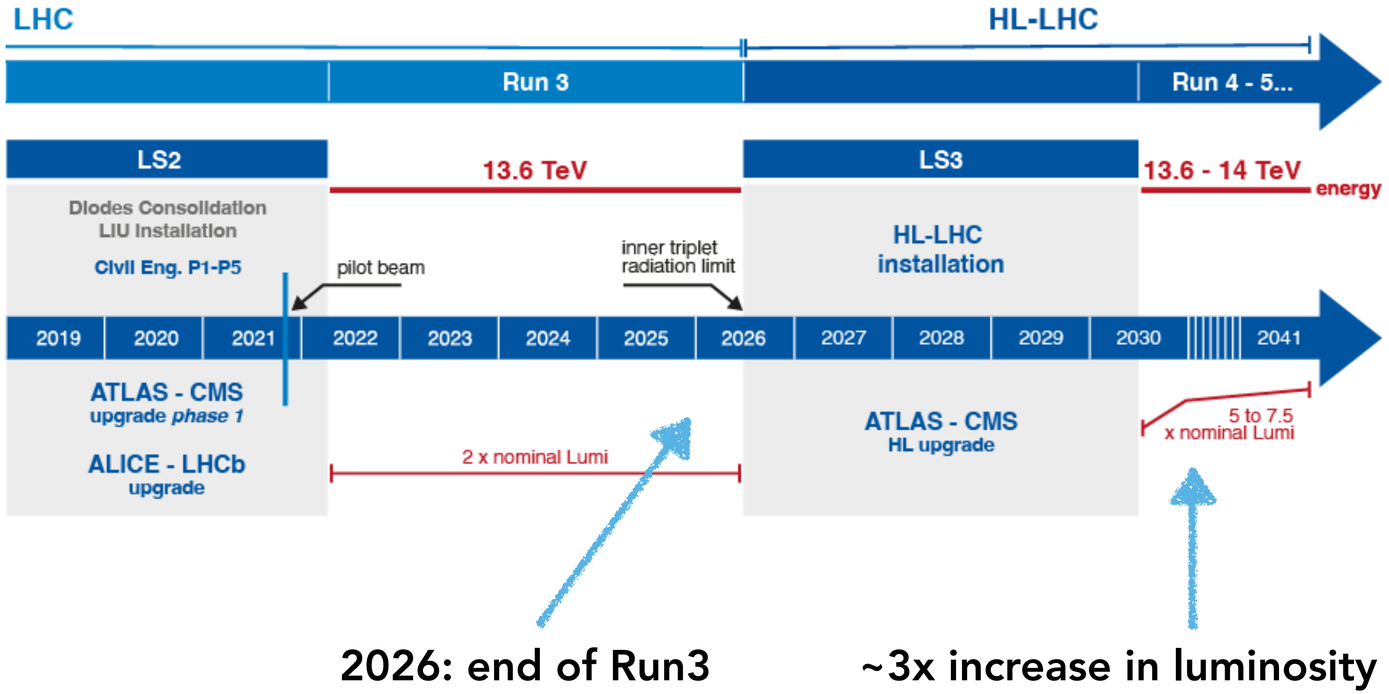


Overarching goal:

Maximise LHC potential

- find BSM physics;
- understand LHC data w/ SM.

Requires better and faster simulators

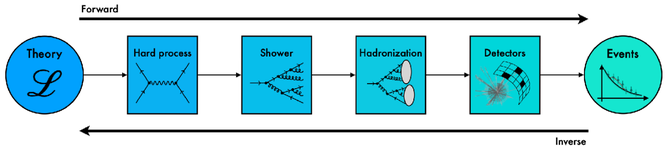


2026: end of Run3

~3x increase in luminosity

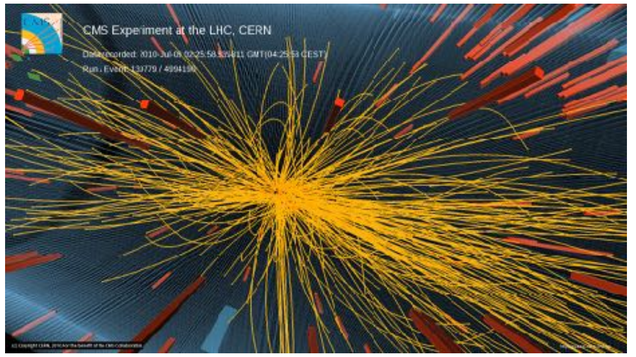
HEP simulations

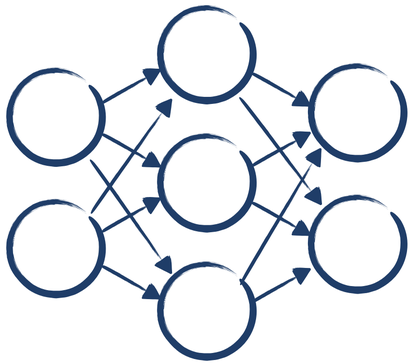
ML emulators



$$\sum |\mathscr{M}^2| \Big/ \Big/ \text{G4}$$

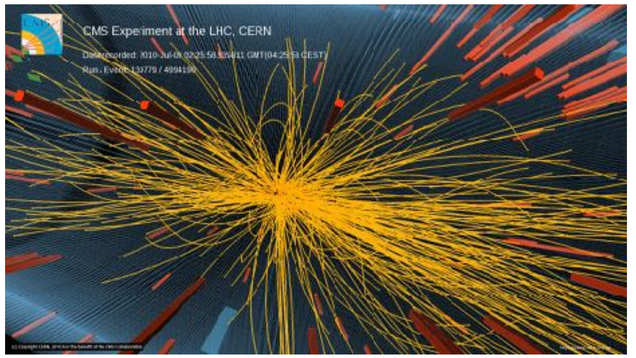
Slow





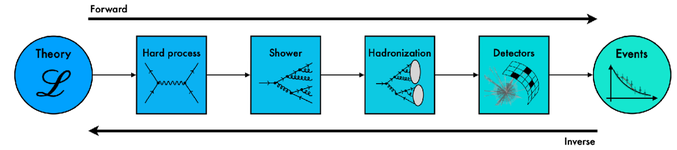
Train with $x \sim p(x \mid \text{simulator})$

Fast

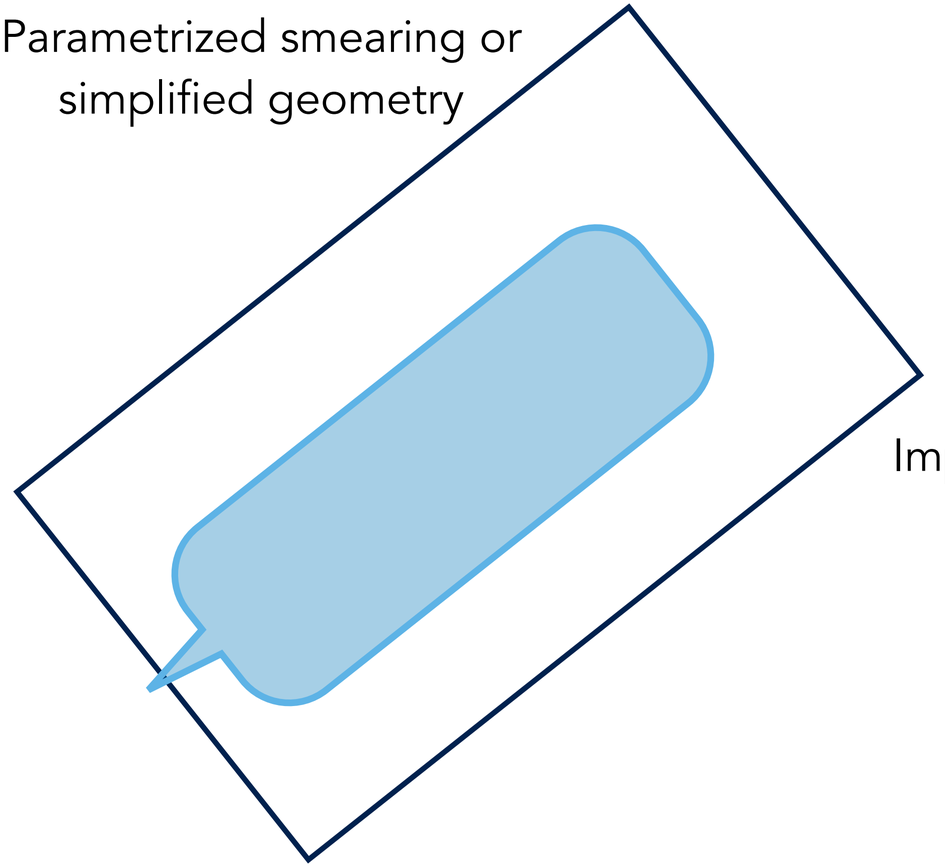


HEP simulations

Surrogate detector simulator



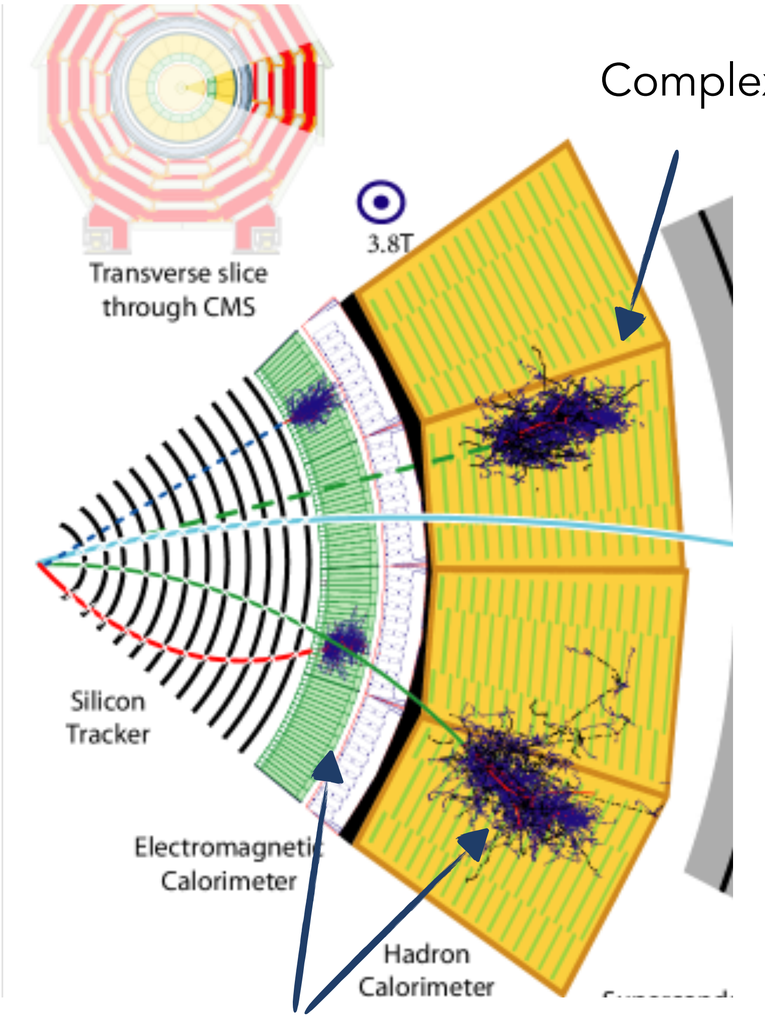
Parametrized smearing or simplified geometry



Simulations

Improved surrogate

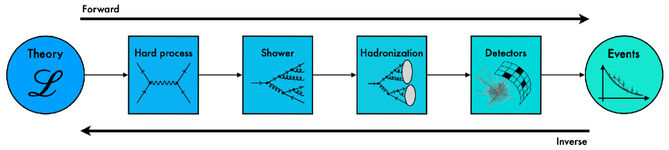
Complex structure



Different physics

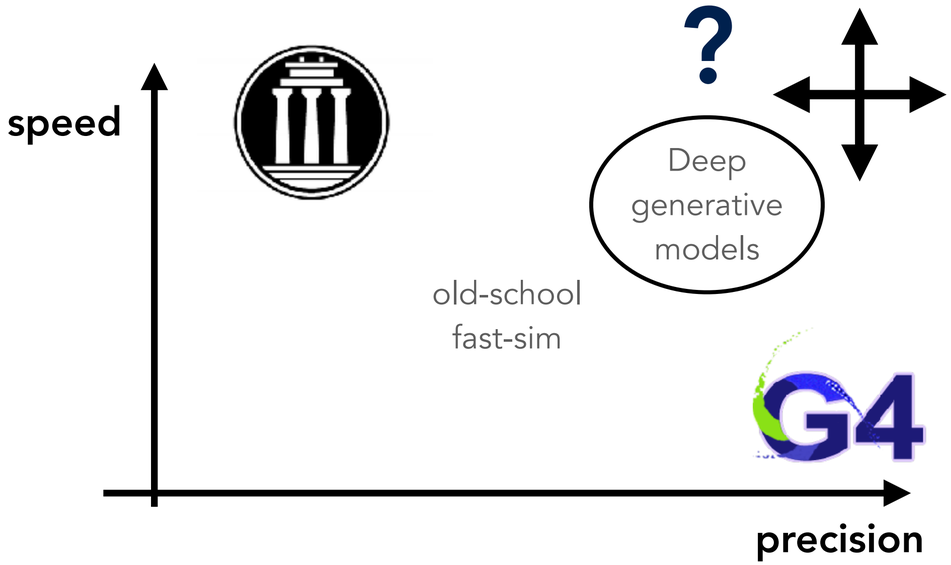
HEP simulations

Surrogate detector simulator



Modern generative networks:

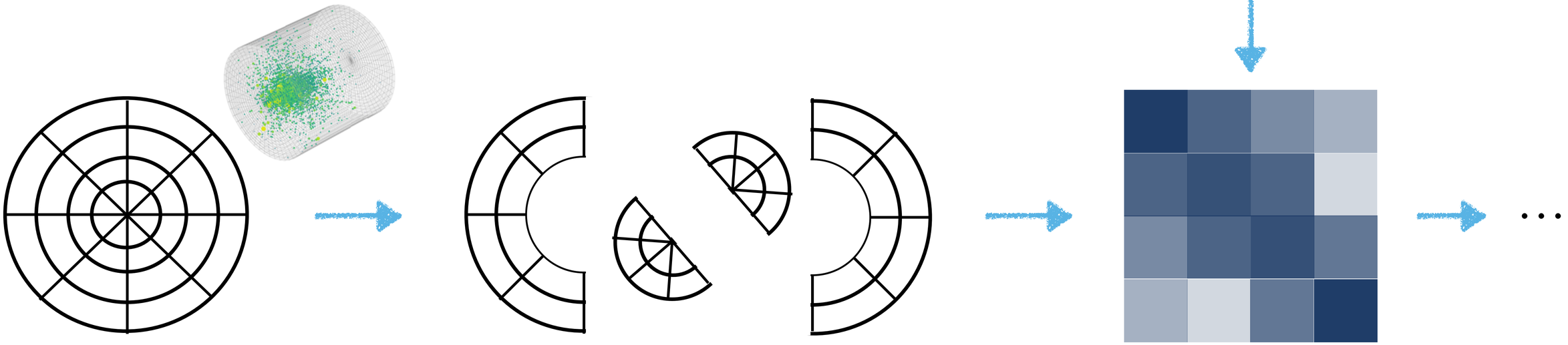
- Complex architectures but still fitting functions
- provided data, approximate Geant4
- speed and precision are key…
- … but no one-fits-all model



Vision Transformers

arXiv:2405.09629, **LF**, Ore A., Palacios Schweitzer S., Plehn T.

Paradigm



Detector geometry

- Energy deposition in each cell;
- high-dimensional input;
- 3D structure: (x, y, z) or (r, α, z).

Divide into patches

- Define a recipe for patching;
- domain-knowledge input.

Self-attention on patched objects

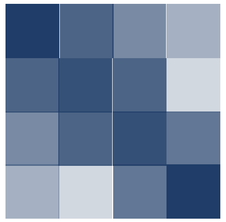
- Stack of transformer blocks;
- conditions: $(E_{\text{inc}}, E_i, \ldots)$.

Vision Transformers

& Generative networks

arXiv:2509.25169, **LF**, Giammanco A., Krause C.

Normalizing Flows



$$\theta(x) \in \mathbb{R}^{(3m-1)n}$$

Learn m bins RQS for n input feature:

$$\mathscr{L} = -\left\langle \log p_z(f_\theta(x)) + \log \left| \frac{\partial f_\theta}{\partial x} \right| \right\rangle$$

- Fast
- Discrete, less expressive

Conditional Flow Matching

$$v_\phi(x,t) \in \mathbb{R}^n$$

Regress a velocity vector comparing to a target conditional velocity:

$$\mathscr{L} = \left\langle \left|\left| v_\phi(x,t) - v(x,t\,|\,x_0) \right|\right| \right\rangle$$

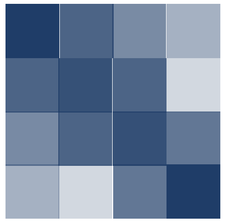
- Very expressive
- Slower, multiple function evaluations

Vision Transformers

& Generative networks

arXiv:2509.25169, **LF**, Giammanco A., Krause C.

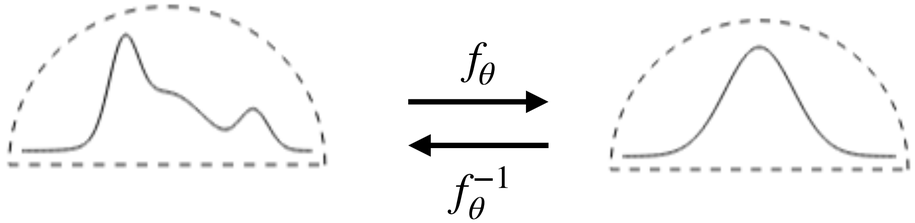
Normalizing Flows



$$\theta(x) \in \mathbb{R}^{(3m-1)n}$$

Learn m bins RQS for n input feature:

$$\mathscr{L} = -\left\langle \log p_z(f_\theta(x)) + \log \left| \frac{\partial f_\theta}{\partial x} \right| \right\rangle$$



Conditional Flow Matching

$$v_\phi(x,t) \in \mathbb{R}^n$$

Regress a velocity vector comparing to a target conditional velocity:

$$\mathscr{L} = \left\langle \left|\left| v_\phi(x,t) - v(x,t\,|\,x_0) \right|\right| \right\rangle$$

Sampling $\longrightarrow$ $x(t=0) = x(t=1) - \int_0^1 v_\phi(x,t)dt$

Vision Transformers

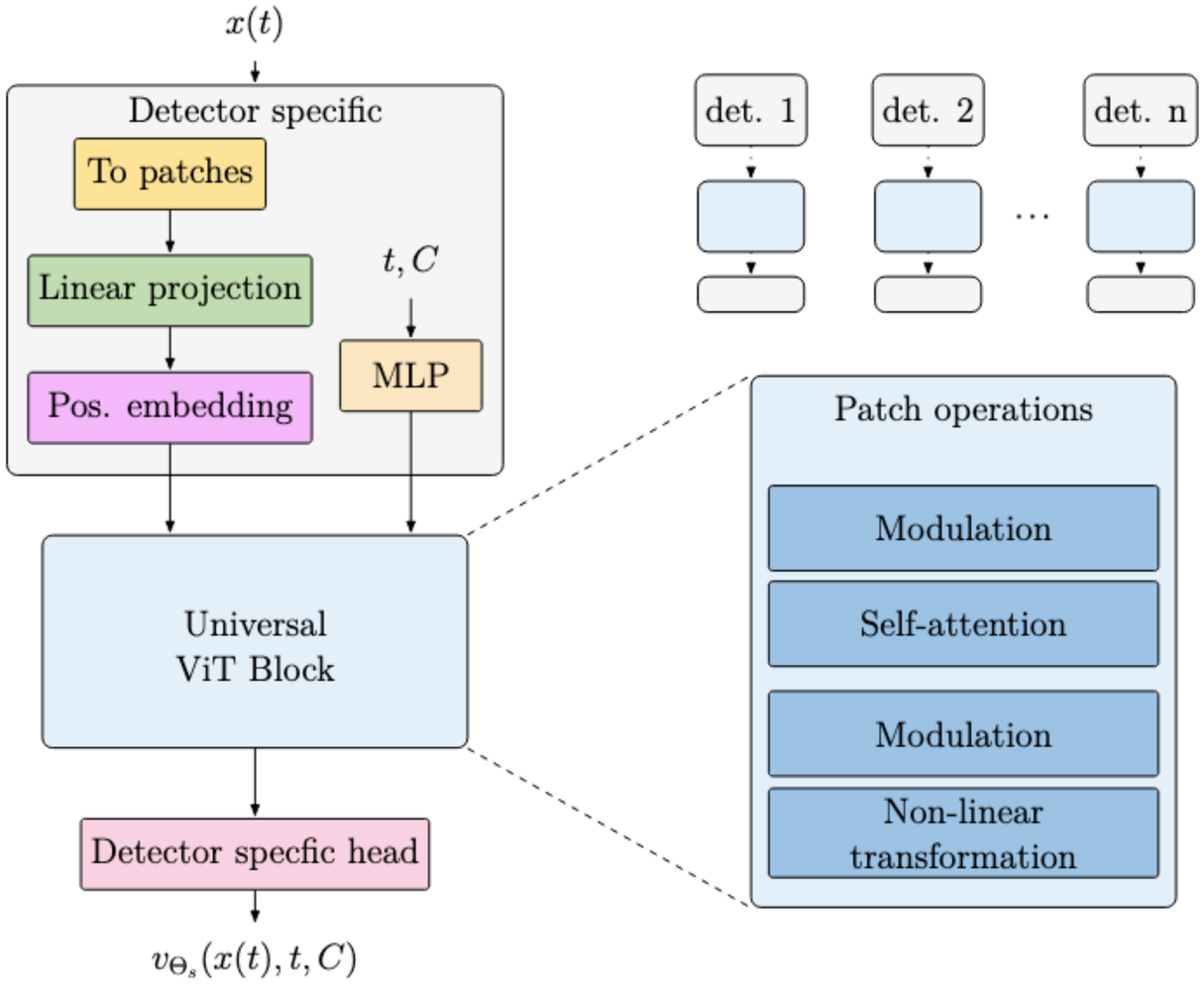
arXiv:2601.05289, **LF**, Giammanco A., Krause C.

- Split generation of layer energies and voxels;
  - Ensures energy conservation.
- split detector into patches and embed into a latent space
- apply a residual transformation to the inputs:
  - multi-head self-attention;
  - fully-connected network (non-linear).

$$x_l = x_h + \gamma_l \cdot g_l(a_l x_h + b_l)$$

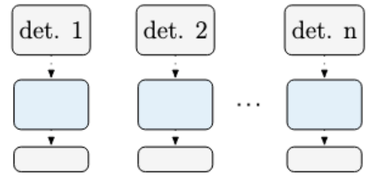
Modulation: γ_l, a_l, b_l learnable, conditioned on t, C

- predict a v_θ for each voxel.
- learned universal block can be reused for other detectors



Universal ViT

Energy profiles



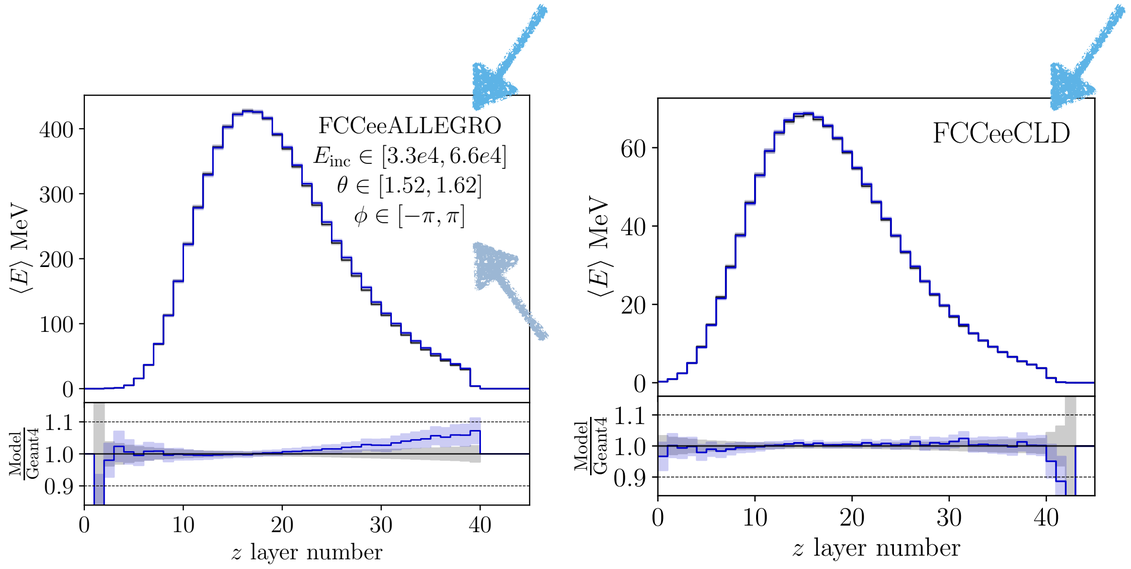
- Handle multiple conditions and geometries

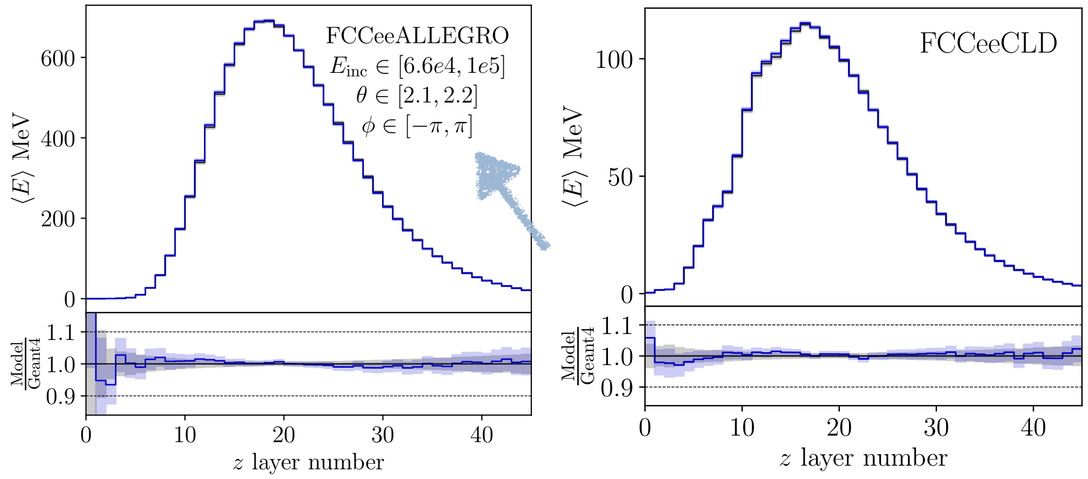
Global parameters: $(E_{\text{inc}}, \theta, \phi)$

Detector label: $(w_1, w_2, w_3, w_4, w_5)$

Included in a single network:

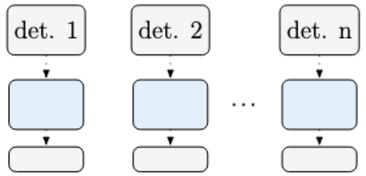
- Multiple detectors: five in this example
- Multiple initial conditions:
  position and energy of the particle



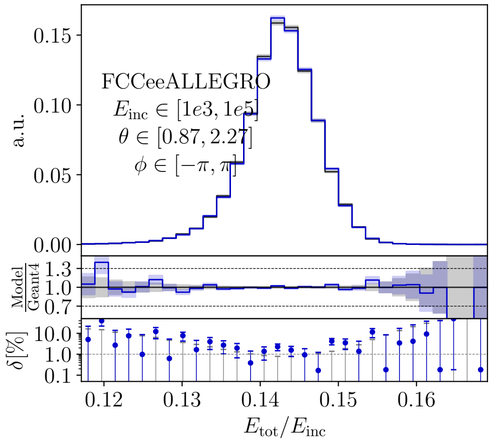


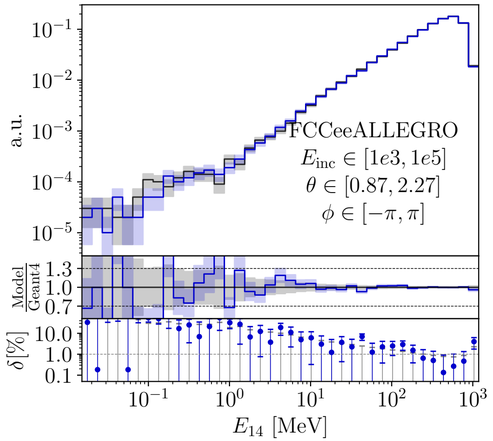
Universal ViT

Layer-wise features

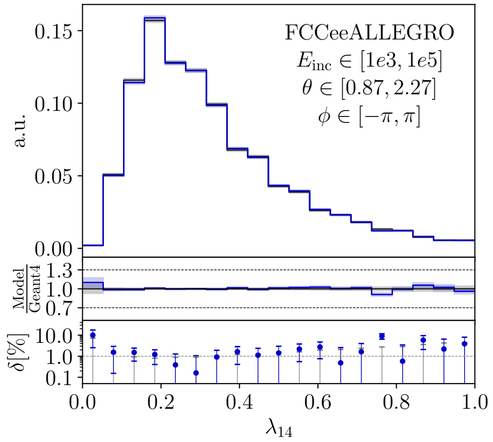


- Shower centre and width in a single layer:
- Total and layer-wise energy depositions



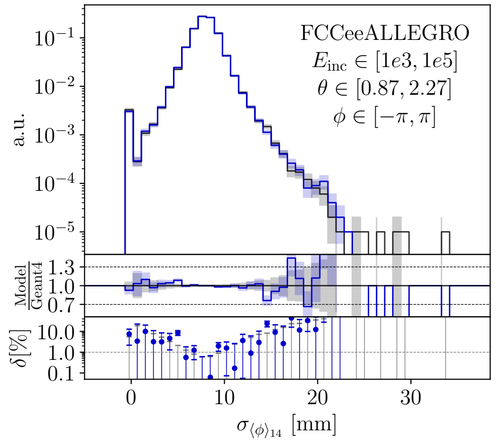


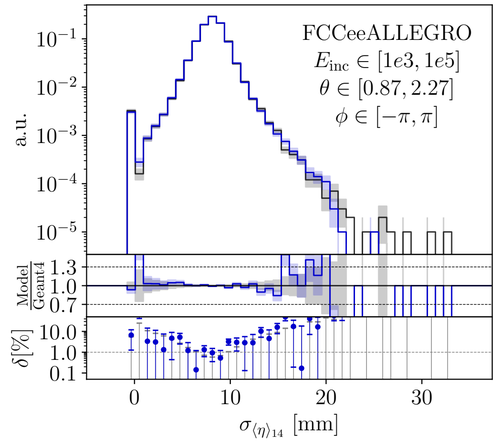
- Sparsity: number of hits

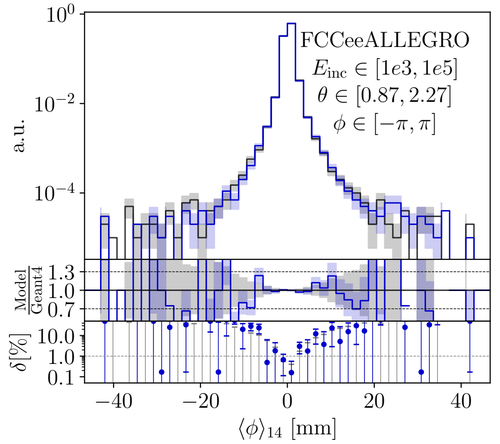


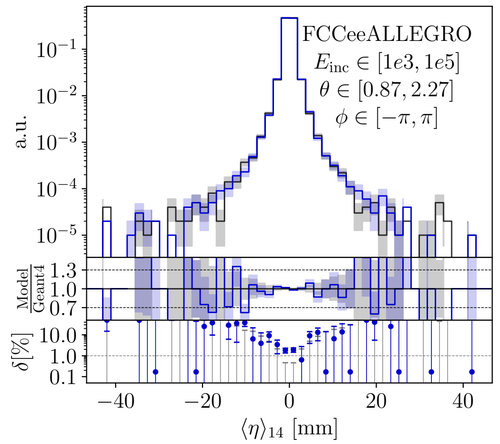
Not limited to profiles!

(but still high-level)





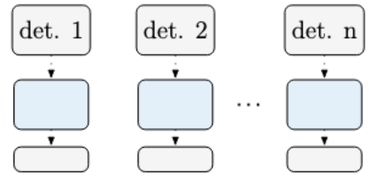




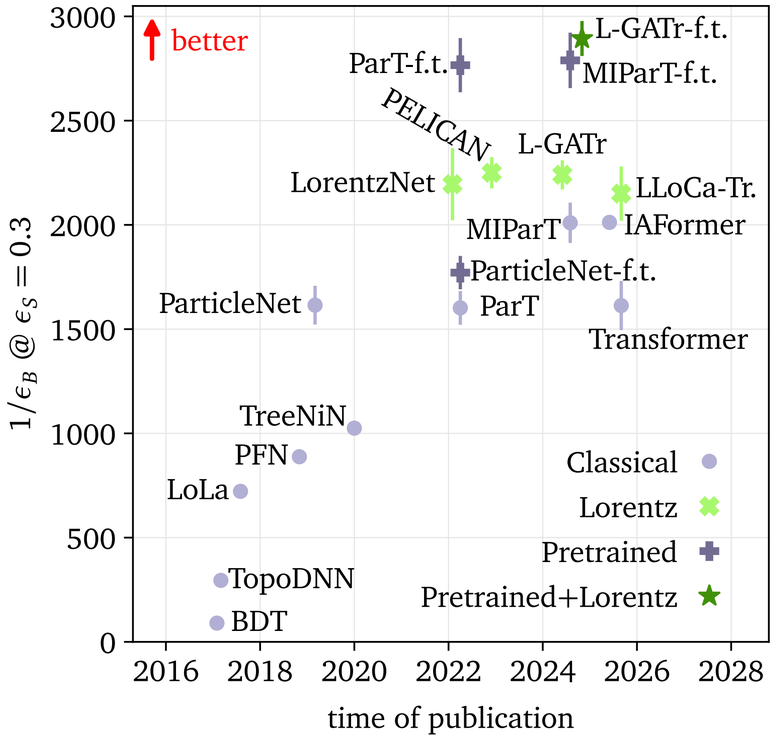
LEMURS — Geant4 — ViT-CFM

Extra

Transfer learning



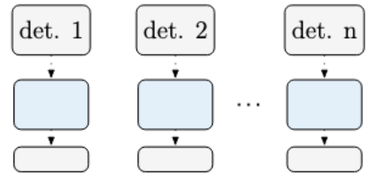
- Foundation models: pre-train on a large datasets, fine-tune on a smaller target.



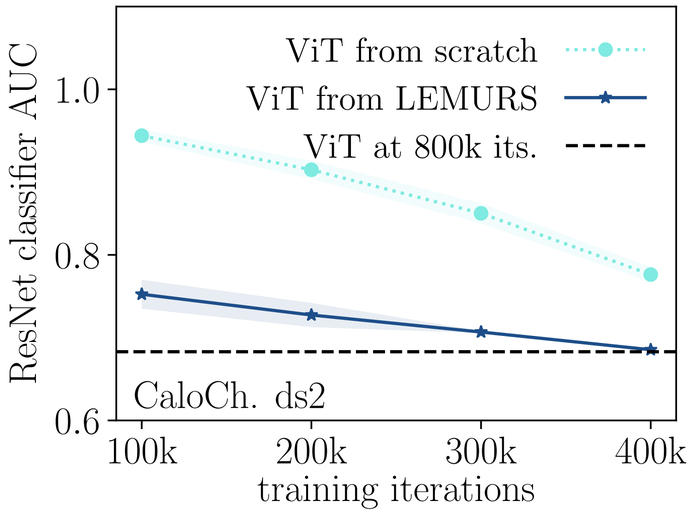
- Jet tagging is again a great example:
  - pre-training on JetClass;
  - fine-tune on top tagging (data limited)
- Allegedly learning a better/more efficient data representation
- We observe a similar behaviour for generative networks

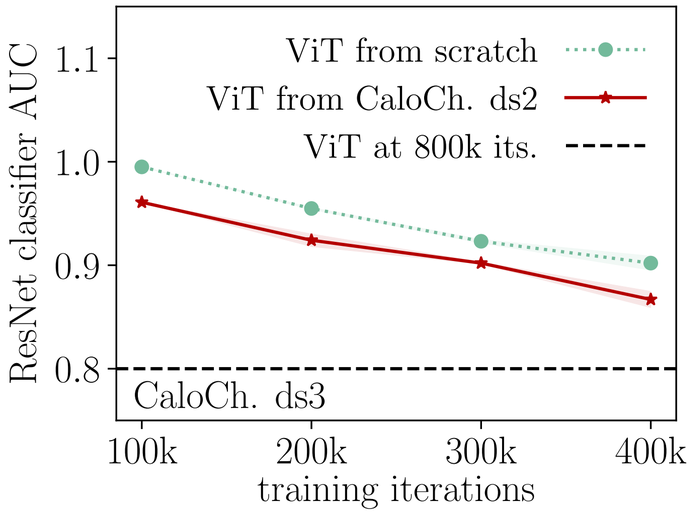
Extra

Transfer learning



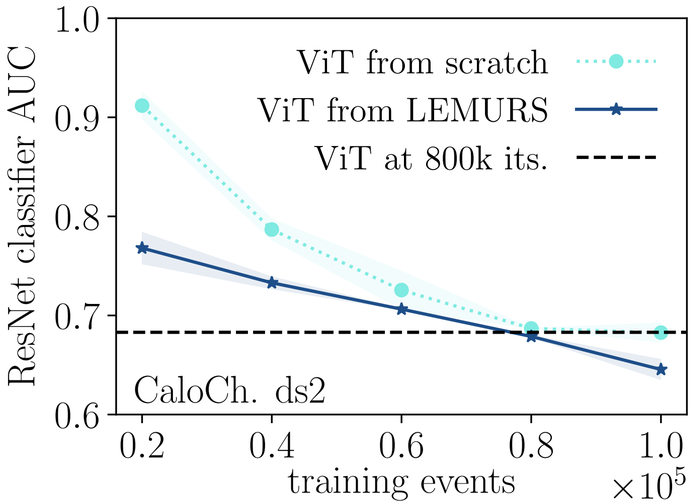
- Start from a pre-trained network — fine-tune for a reduced number of training iterations





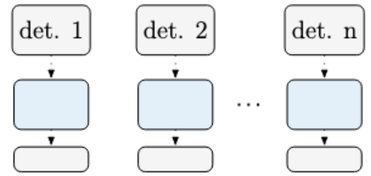
- … or fine-tune on a fraction of the original training data

  ⟶ reduce runtime for generating data from expensive simulators



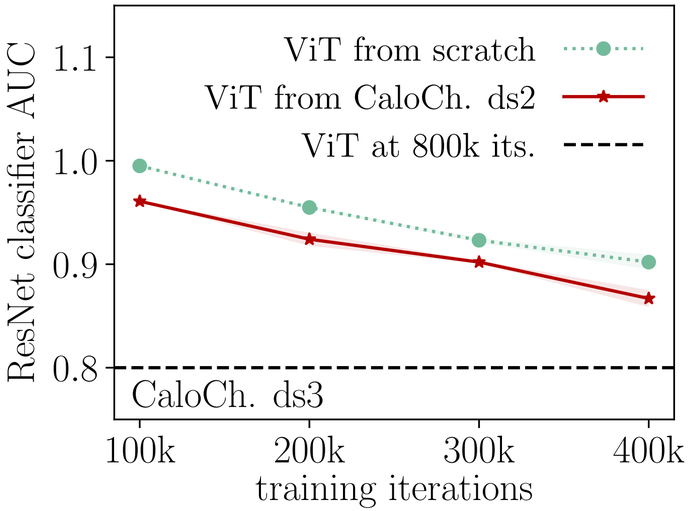
Extra

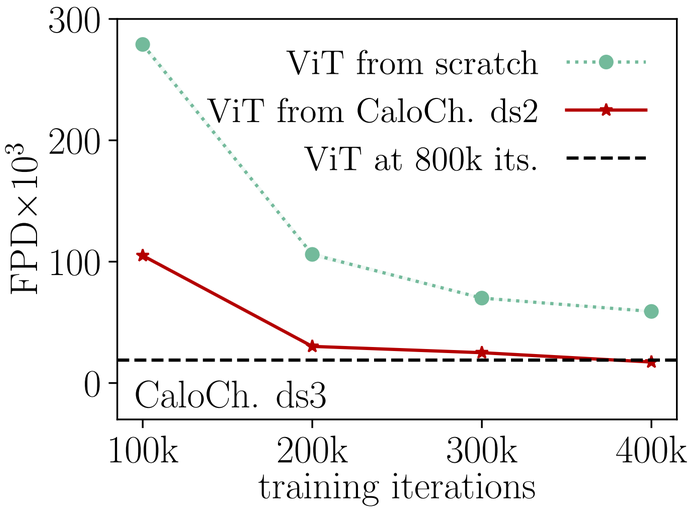
Evaluating transfer learning



- The selection of the evaluation metric can highly affect the observed efficiency gains.

Likelihood-ratio approximator





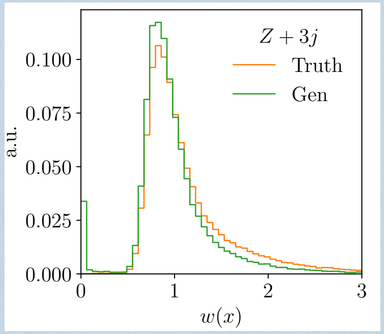
FPD — integral probability measure

- Holistic estimation should always be done on the most sensitive metric.
- Oversimple 1d metrics can give overconfident estimations.

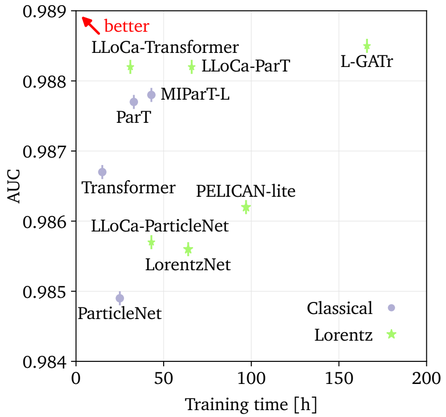
Being a “centaur scientist” means capitalizing on emerging AI technologies while insisting on scientific rigor and robustness

- Jesse Thaler

- Example: Approximating likelihood-ratios



$$w(x) = \frac{p(x)}{q(x)}$$



- Bittersweet lesson(?)

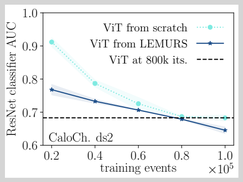
LLoCa

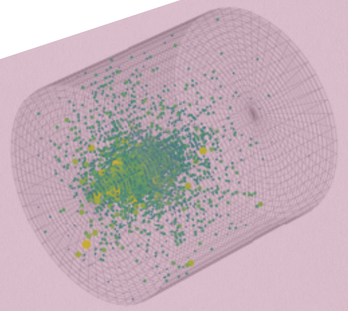
Embedding physics symmetries and structure in neural networks

As easy as a “pip install lloca”

Towards production: CMS HGCal

- Extra: exploiting large data





- ML-emulators and their validation

Applications of generative networks for fast detector simulations