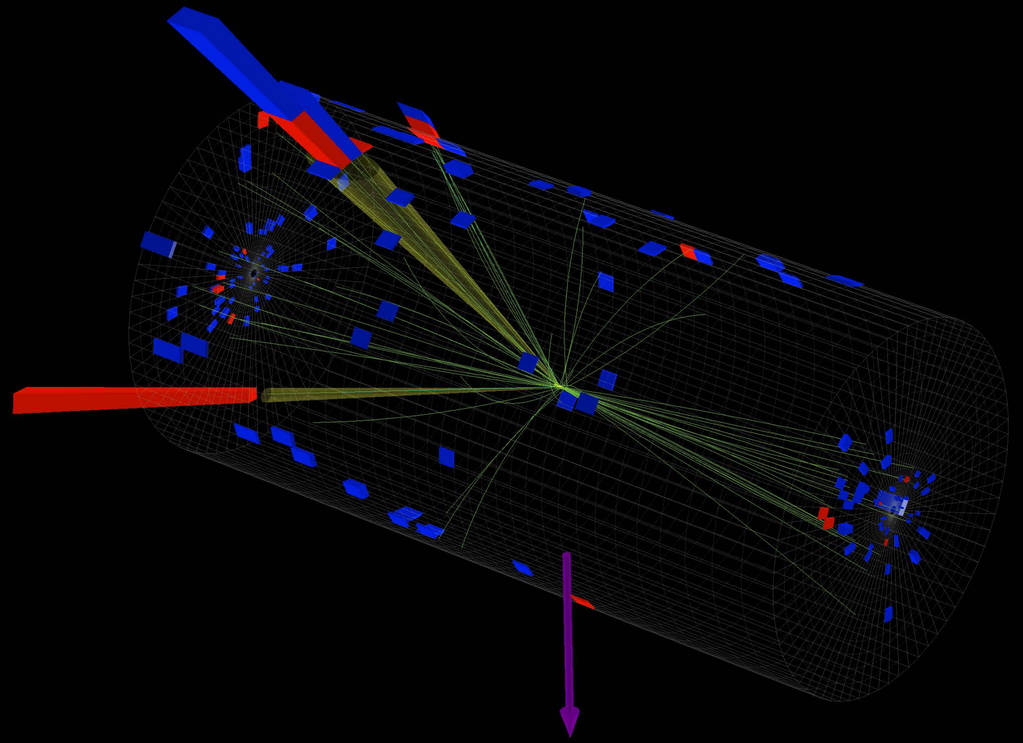
AI For HEP

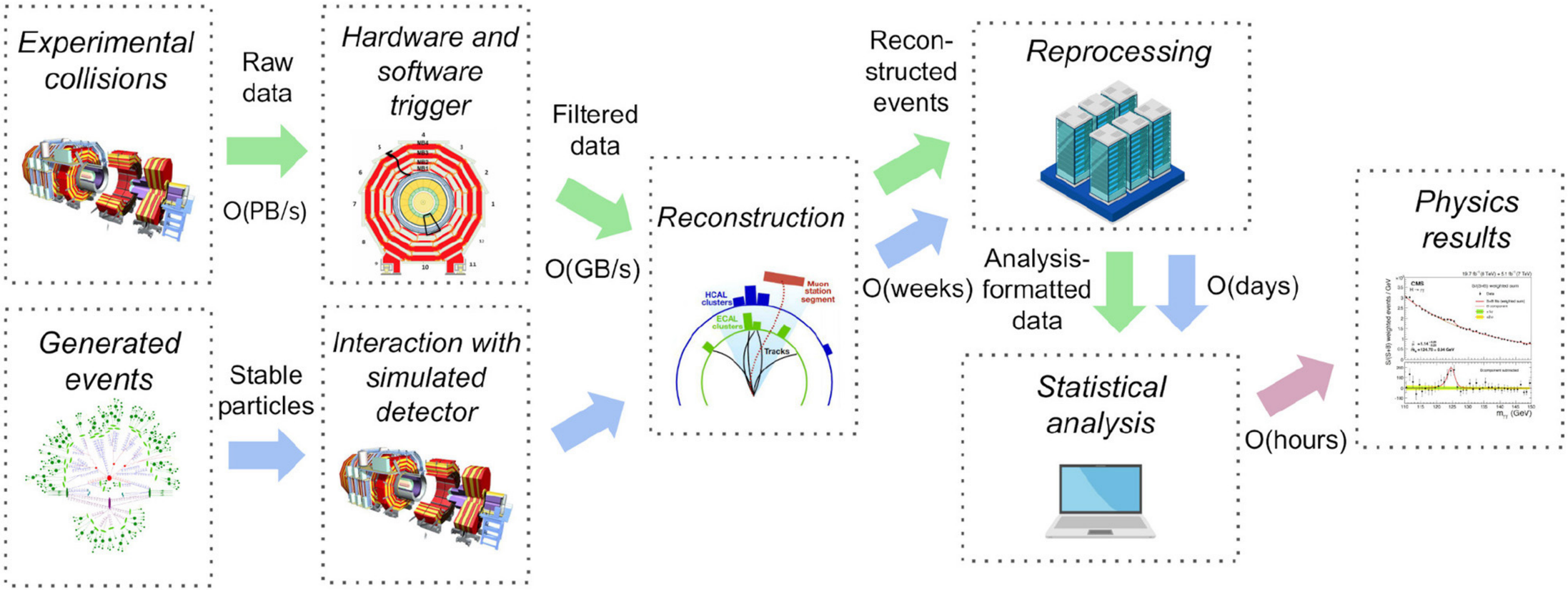
— A Representation Learning Perspective

Huilin Qu (曲慧麟)



New Opportunities for Particle Physics 2026
August 16, 2026

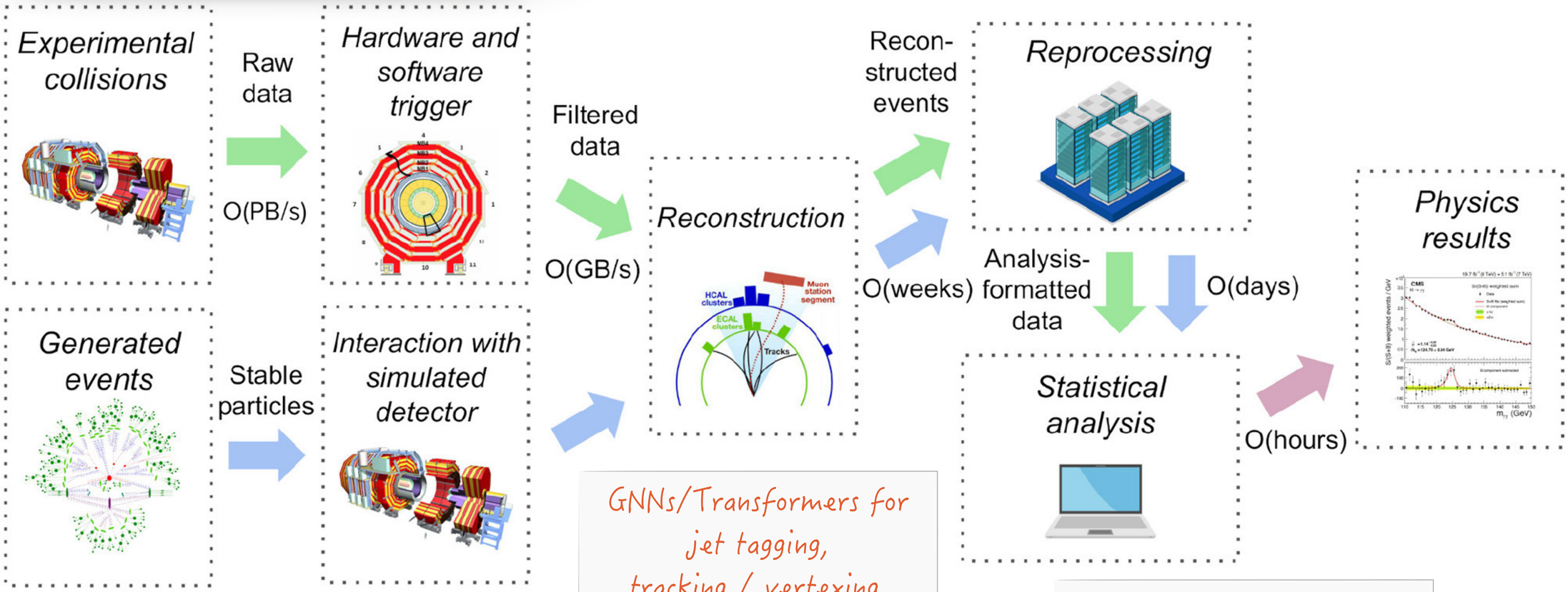
HEP Data Flow…



Front.Big Data 4 (2021) 661501

HEP Data Flow… Matched With ML!

Ultrafast inference (FPGA/ASIC), data compression, anomaly detection…



Generative models for event generation & fast simulation: GAN, VAE, normalizing flow, diffusion…

GNNs/Transformers for jet tagging, tracking / vertexing, particle-flow, calorimeter reconstruction…

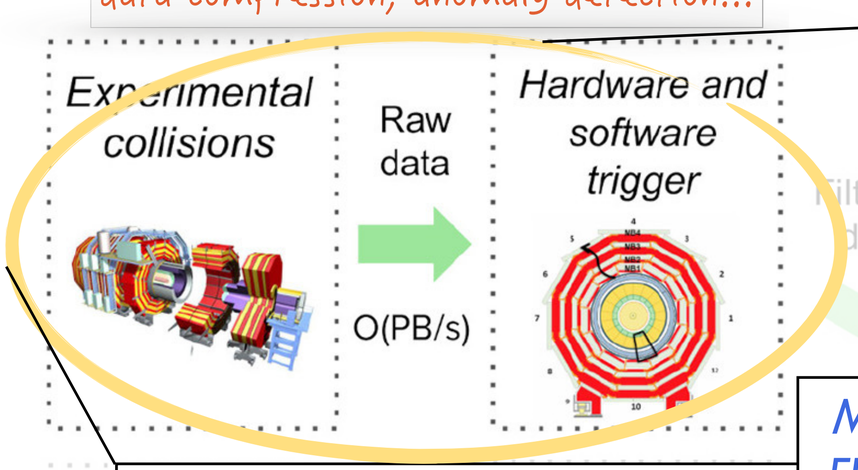
Unfolding, simulation-based inference, differentiable programming…

Front.Big Data 4 (2021) 661501

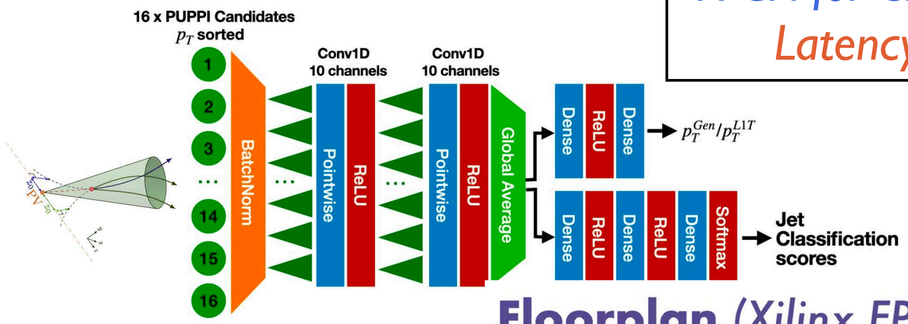
HEP Data Flow… Matched With ML!

Ultrafast inference (FPGA/ASIC), data compression, anomaly detection…

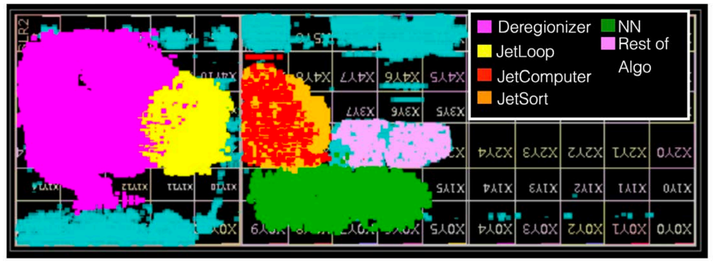
CMS-DP-2025-032

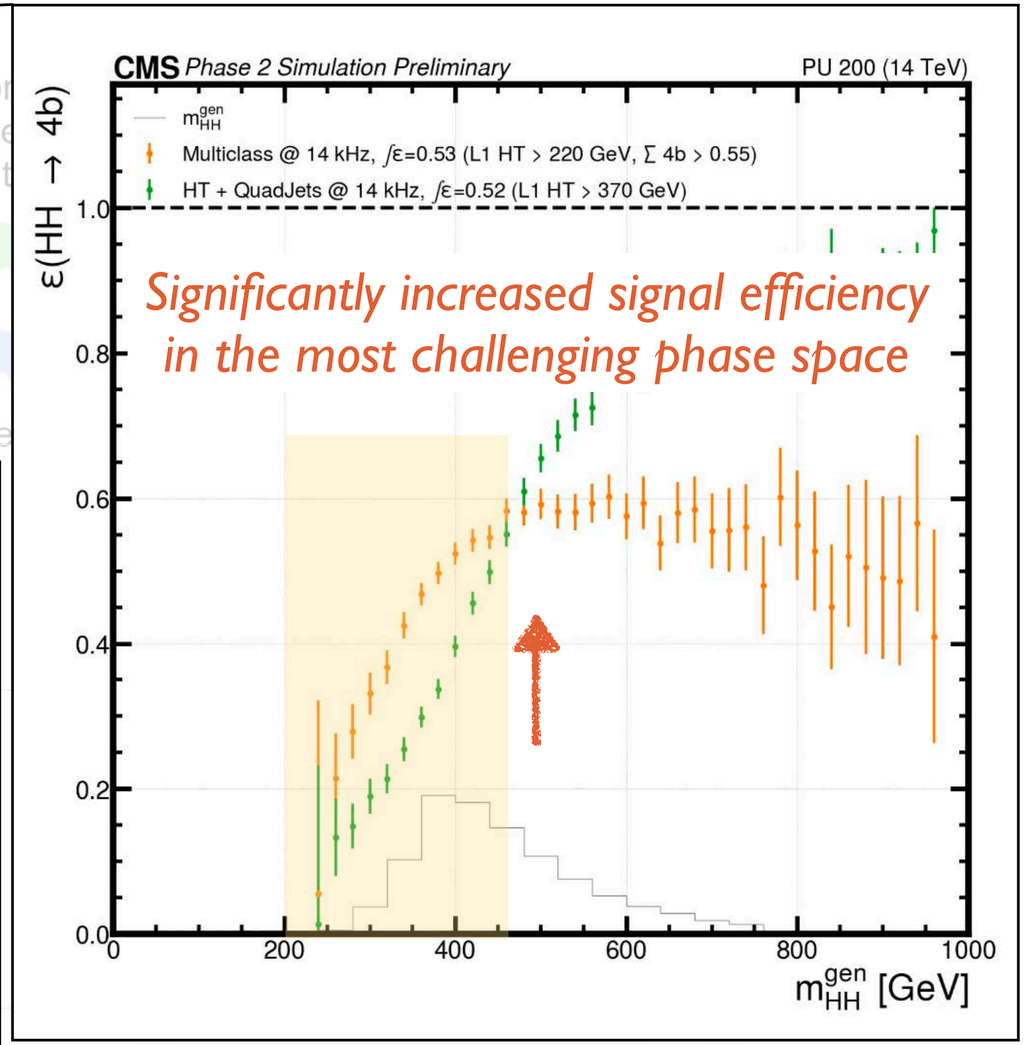


Multiclass jet tagging on FPGA for CMS L1 Trigger
Latency: ~150ns



Floorplan *(Xilinx FPGA VU9P-2)*



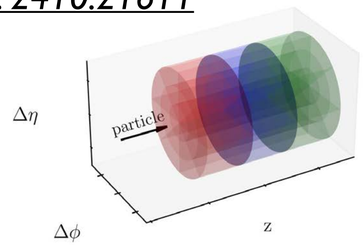


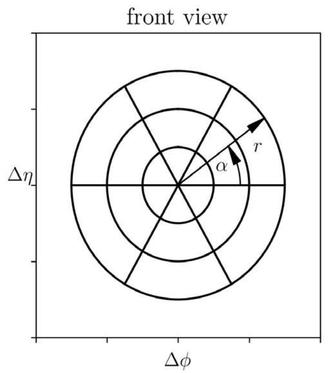
Front.Big Data 4 (2021) 661501

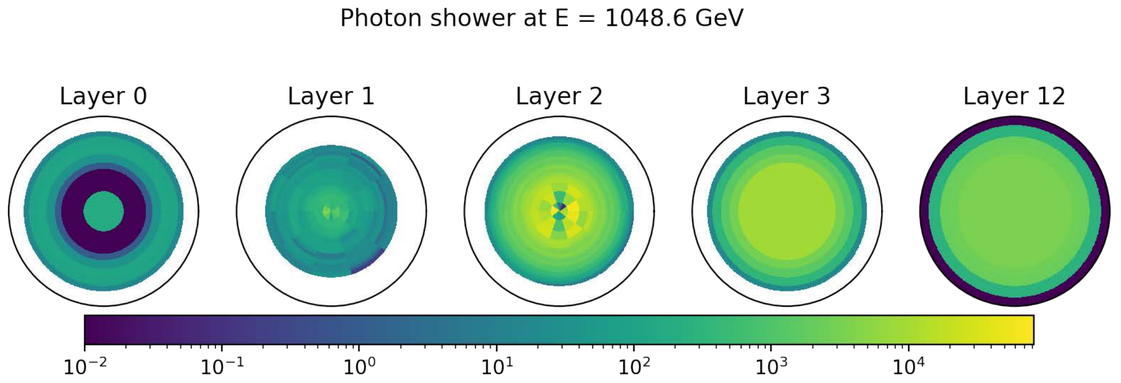
HEP Data Flow… Matched With ML!

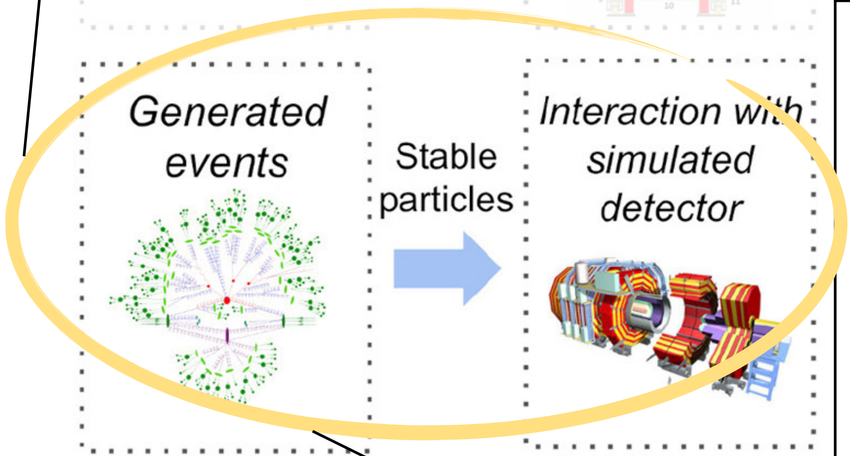
Fast calorimeter simulation using generative models

arXiv: 2410.21611

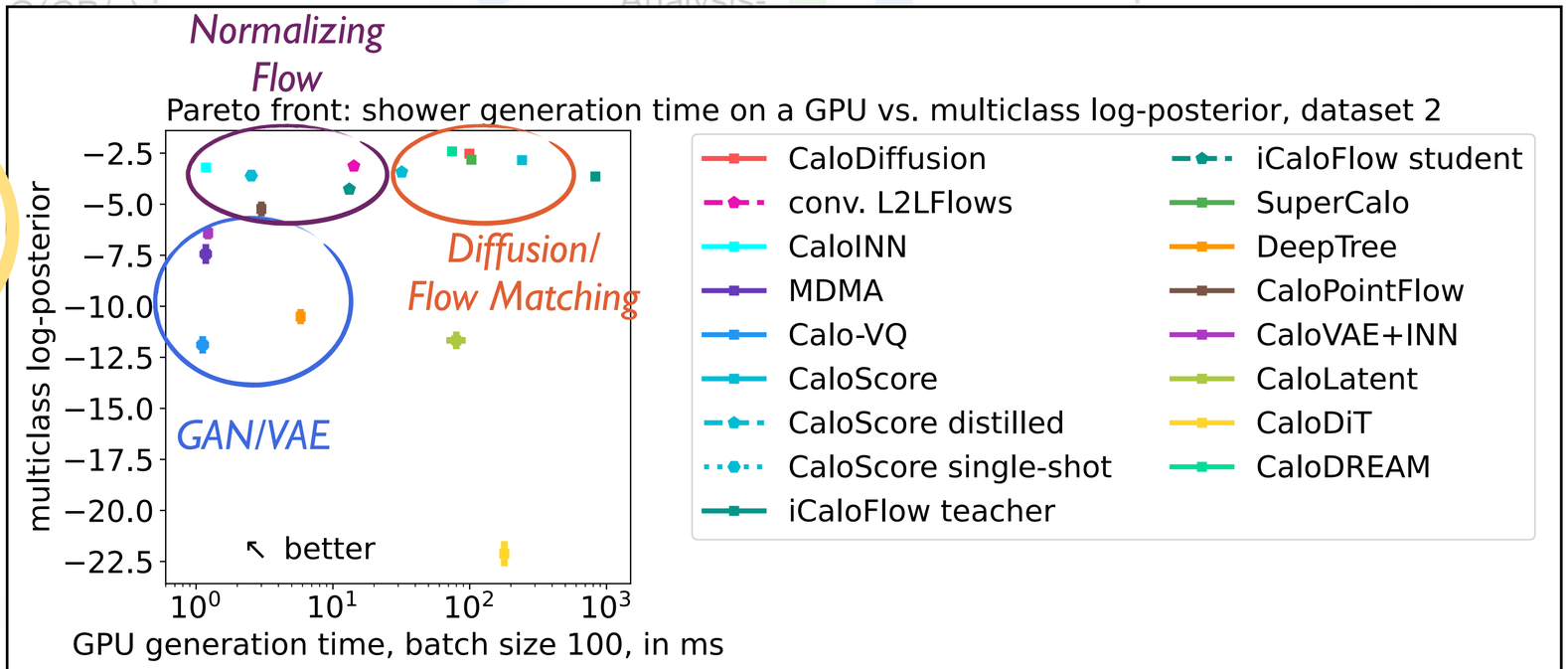




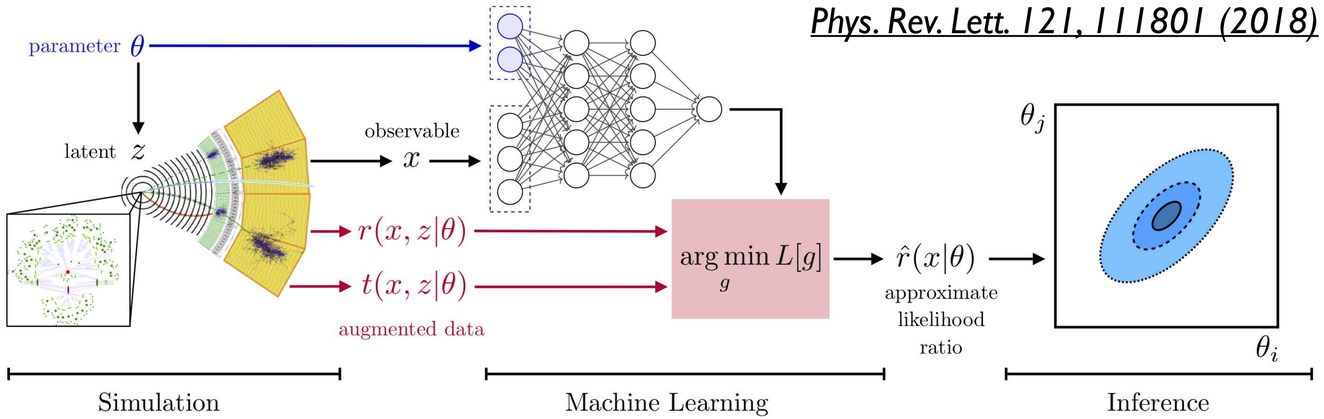




Generative models for event generation & fast simulation: GAN, VAE, normalizing flow, diffusion…

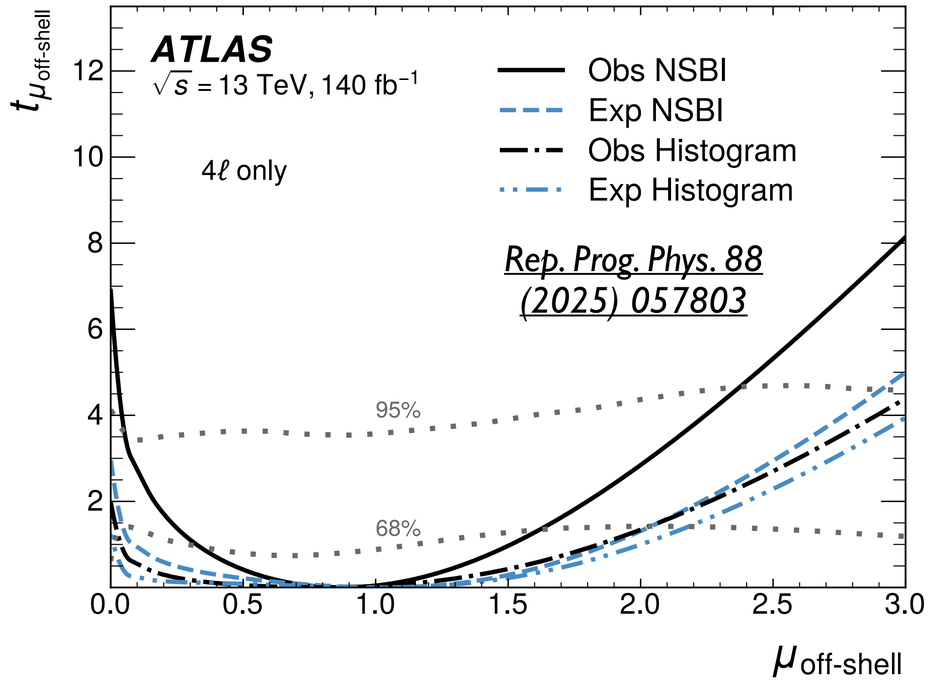


HEP Data Flow… Matched With ML!

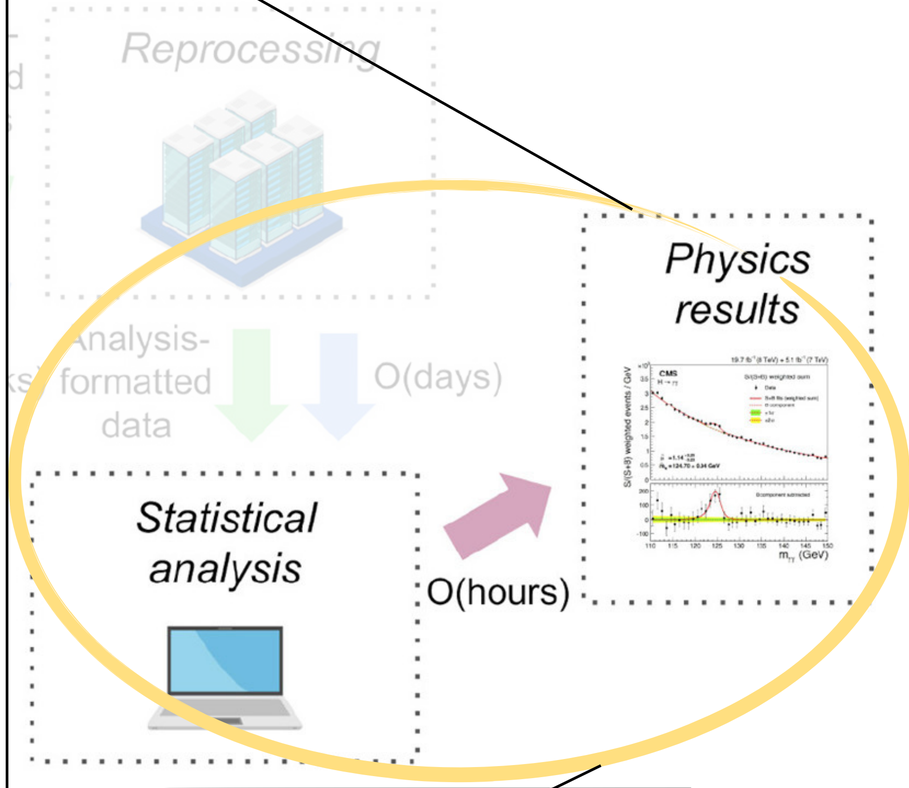


Phys. Rev. Lett. 121, 111801 (2018)

Approximating complex likelihood using neural networks for faster and more optimal statistical inference.



Rep. Prog. Phys. 88 (2025) 057803



Unfolding, simulation-based inference, differentiable programming…

Front.Big Data 4 (2021) 661501

HEP Data Flow… Matched With ML!

Ultrafast inference (FPGA/ASIC), data compression, anomaly detection…

Focus of today

Experimental collisions

Raw data

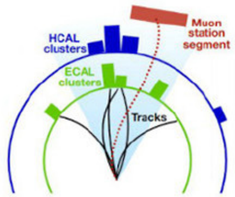
O(PB/s)

Hardware and software trigger

Filtered data

O(GB/s)

Reconstruction



Recon-structed events

O(weeks)

Reprocessing

Analysis-formatted data

O(days)

Statistical analysis

O(hours)

Physics results

Generated events

Stable particles

Interaction with simulated detector

GNNs/Transformers for jet tagging, tracking / vertexing, particle-flow, calorimeter reconstruction…

Generative models for event generation & fast simulation: GAN, VAE, normalizing flow, diffusion…

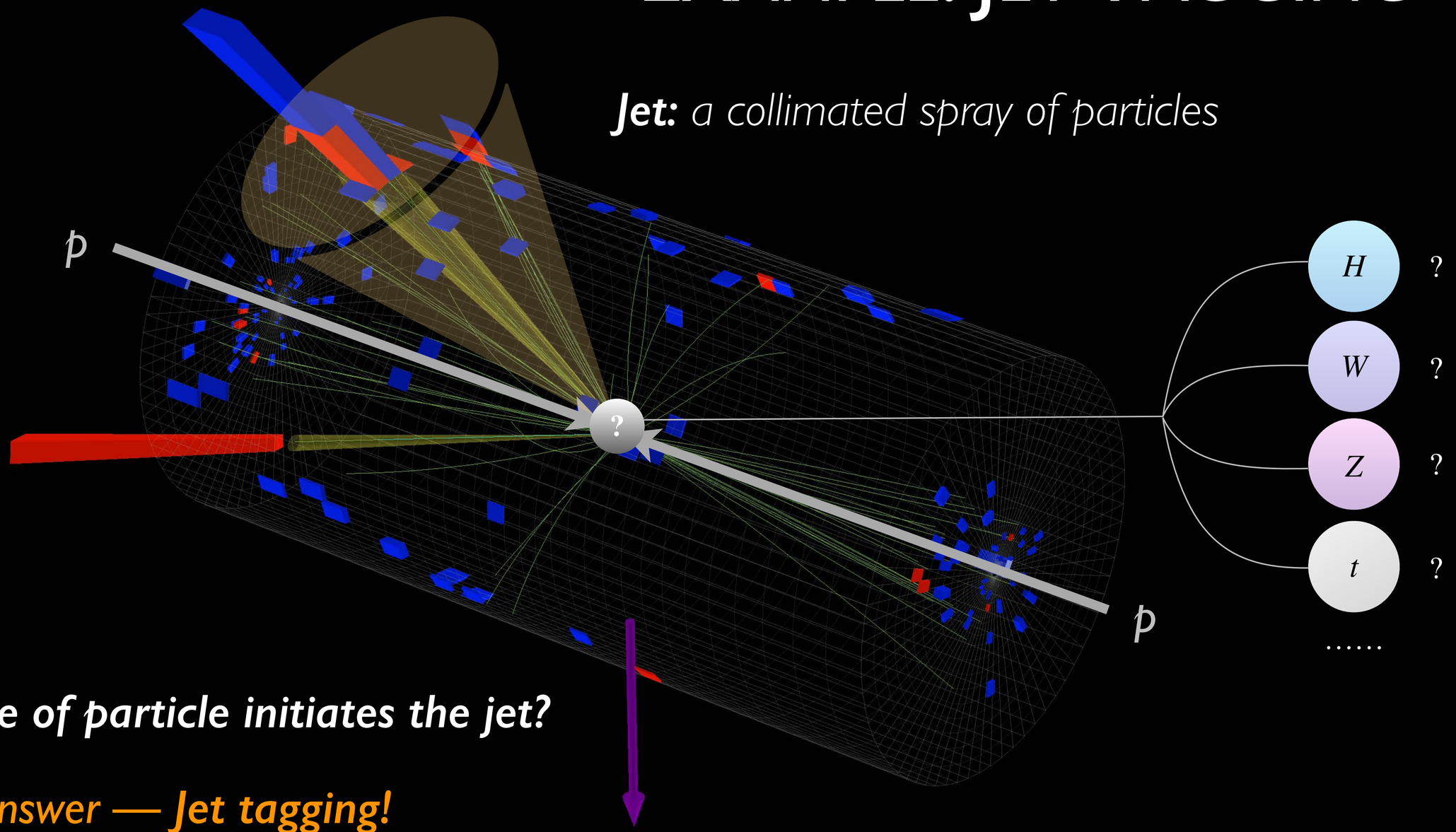
Unfolding, simulation-based inference, differentiable programming…

Front.Big Data 4 (2021) 661501

CMS Experiment at LHC, CERN
Data recorded: Sat Aug 5 15:32:22 2017 CEST
Run/Event: 300515 / 205888132

EXAMPLE: JET TAGGING

Jet: *a collimated spray of particles*



Key question:

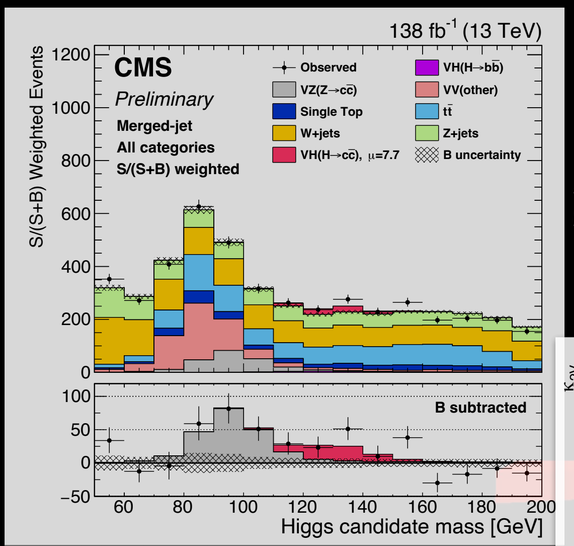
What type of particle initiates the jet?

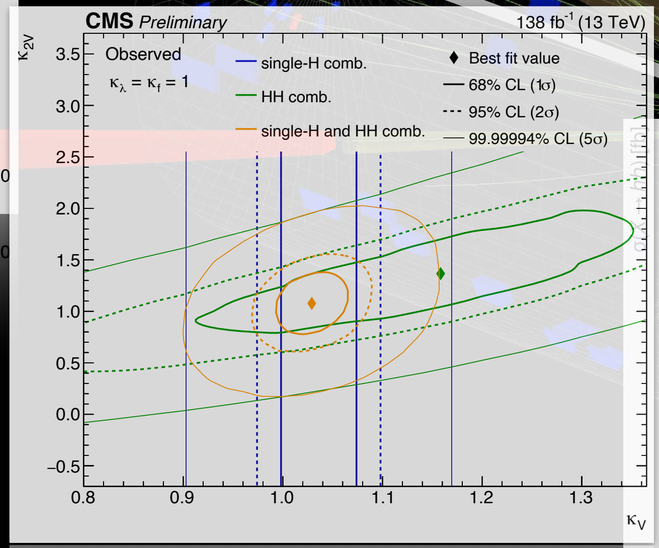
The answer — Jet tagging!

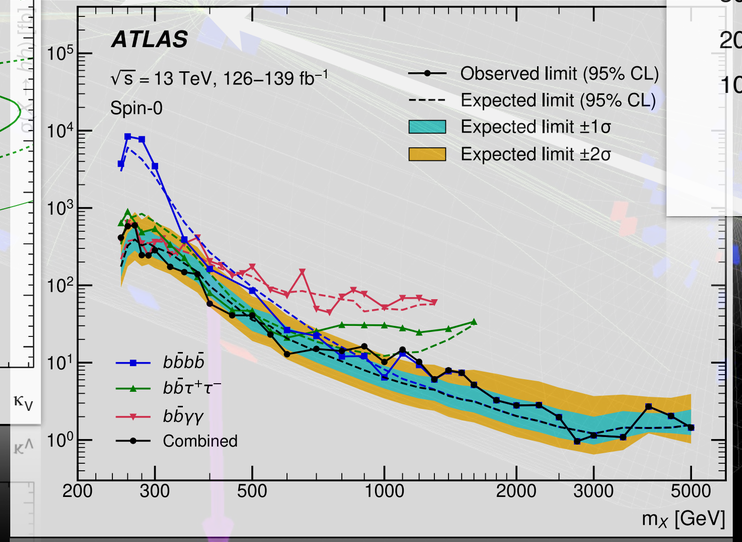
CMS Experiment at LHC, CERN
Data recorded: Sat Aug 5 15:32:22 2017 CEST
Run/Event: 300515 / 205888132

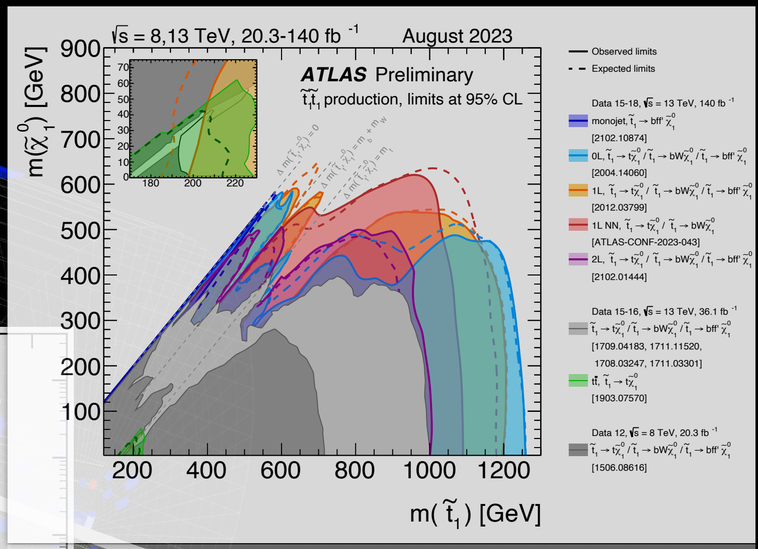
EXAMPLE: JET TAGGING

Powerful handle to search for new phenomena





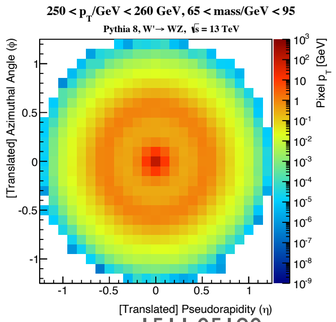




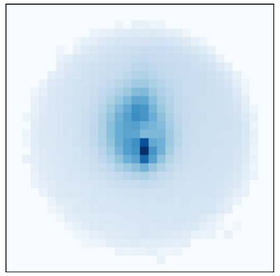
CMS Experiment at LHC, CERN
Data recorded: Sat Aug 5 15:32:22 2017 CEST
Run/Event: 300515 / 205888132

EXAMPLE: JET TAGGING

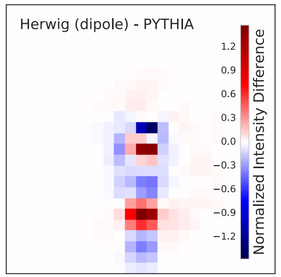
But also: excellent playground for machine learning



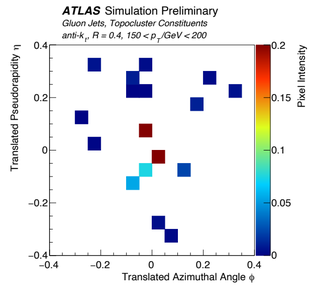
1511.05190



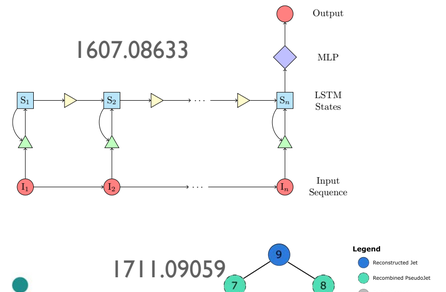
1603.09349

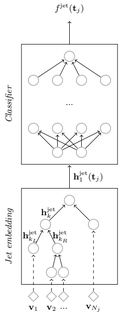


1609.00607

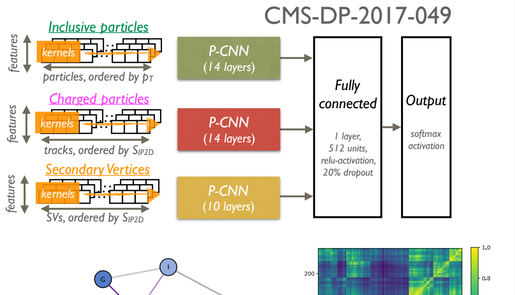


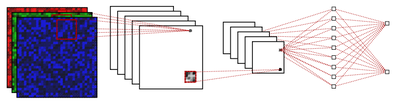
ATL-PHYS-PUB-2017-017





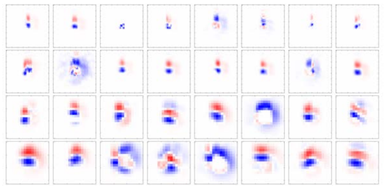
1702.00748



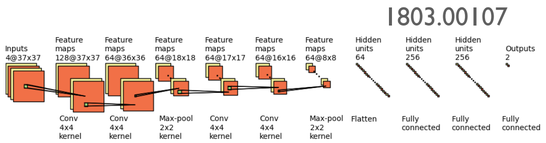


red = transverse momenta of charged particles
green = the transverse momenta of neutral particles
blue = charged particle multiplicity

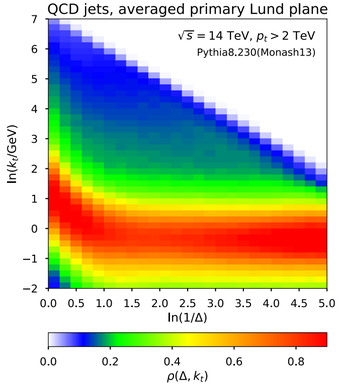
1612.01551



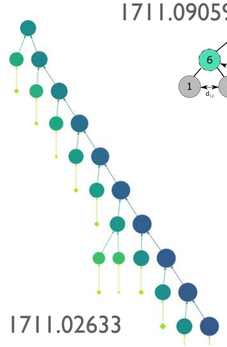
1701.08784



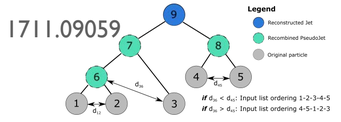
1803.00107

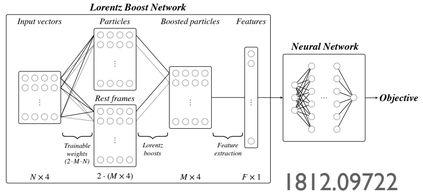


1807.04758



1711.02633





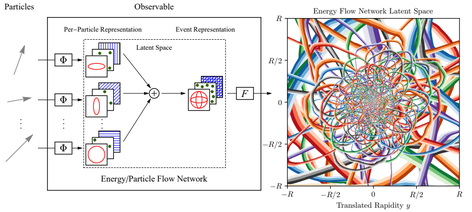
1812.09722

$$k_{\mu,i} \xrightarrow{\text{CoLa}} \tilde{k}_{\mu,j} = k_{\mu,i}\, C_{ij}$$

$$\text{with } C = \begin{pmatrix} 1 & 1 & \cdots & 0 & \chi_1 & \cdots & 0 & C_{1,N+2} & \cdots & C_{1,M} \\ \vdots & & \ddots & & & \ddots & & \vdots & \ddots & \vdots \\ 1 & 0 & \cdots & 1 & 0 & \cdots & \chi_N & C_{N,N+2} & \cdots & C_{N,M} \end{pmatrix}$$

$$\tilde{k}_j \xrightarrow{\text{LoLa}} \tilde{k}_j = \begin{pmatrix} m^2(\tilde{k}_j) \\ p_T(\tilde{k}_j) \\ p_T(\tilde{k}_j)\Delta R_{j,\text{jet}} \\ w_{jm}^{(E)} E(\tilde{k}_m) \\ w_{jm}^{(d)} d_{jm}^2 \\ E_T(\tilde{k}_j) E_T(\tilde{k}_m)(\Delta R_{jm})^{0.2} \end{pmatrix},$$

1707.08966, 1812.09223



1810.05165

NIPS2017 workshop [http://bit.ly/2AkwYRG]

And more…

Evolution of Jet Representations

image, sequence, point cloud, …

Attention Is All You Need?

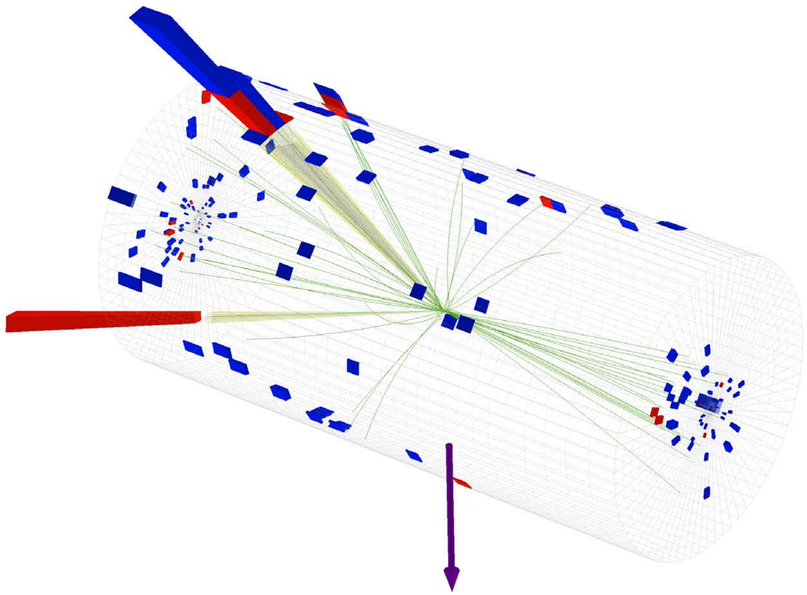
or how physics can help…

Towards a Foundation Model

scaling, self-supervision, and more…

Data Representation

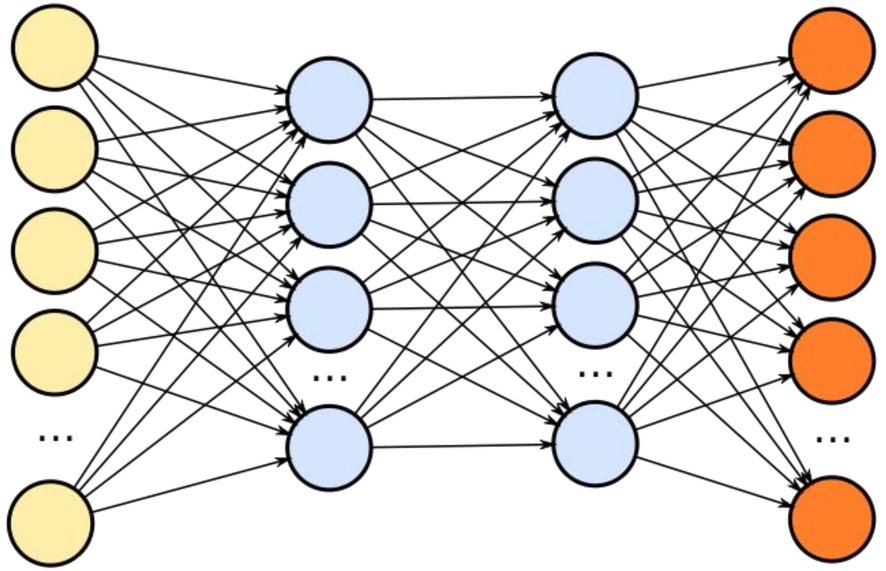
HEP



Collision events, detector hits, sensor arrays, …

×

ML

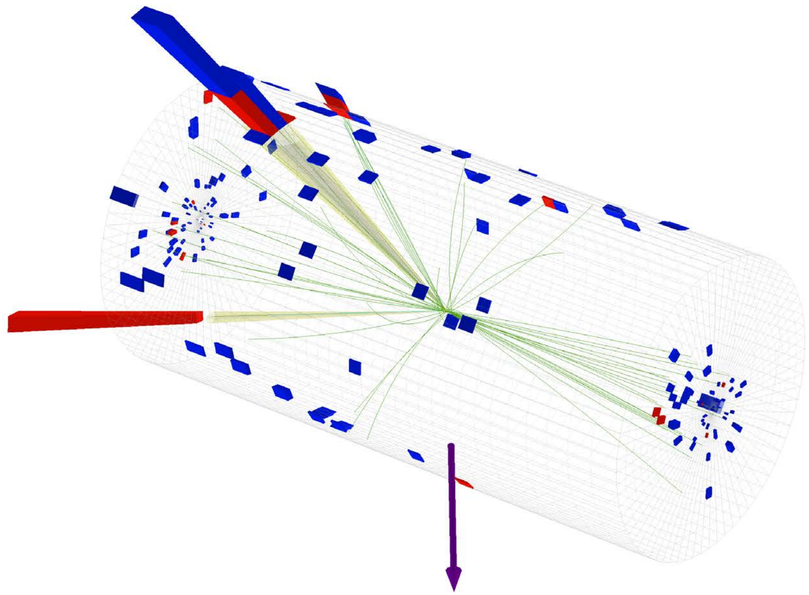


First and foremost:

How to represent the data?

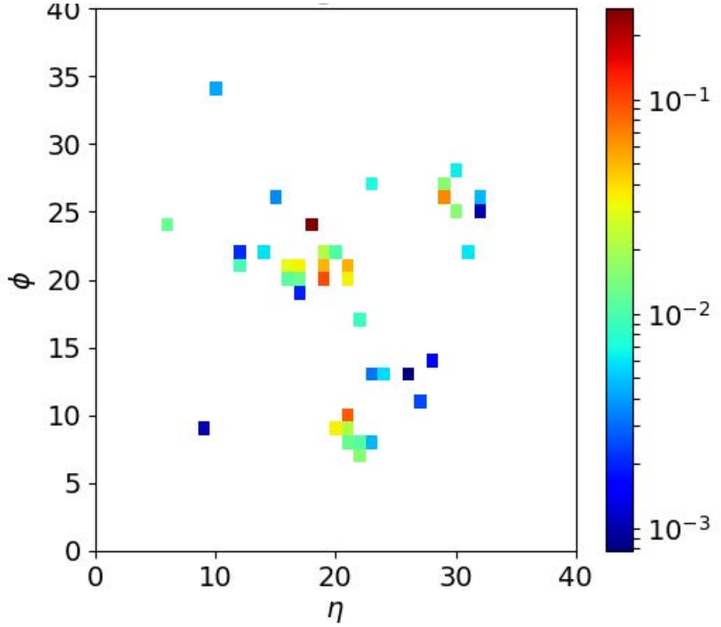
Data Representation: Image

HEP



Collision events, detector hits, sensor arrays, …

Image

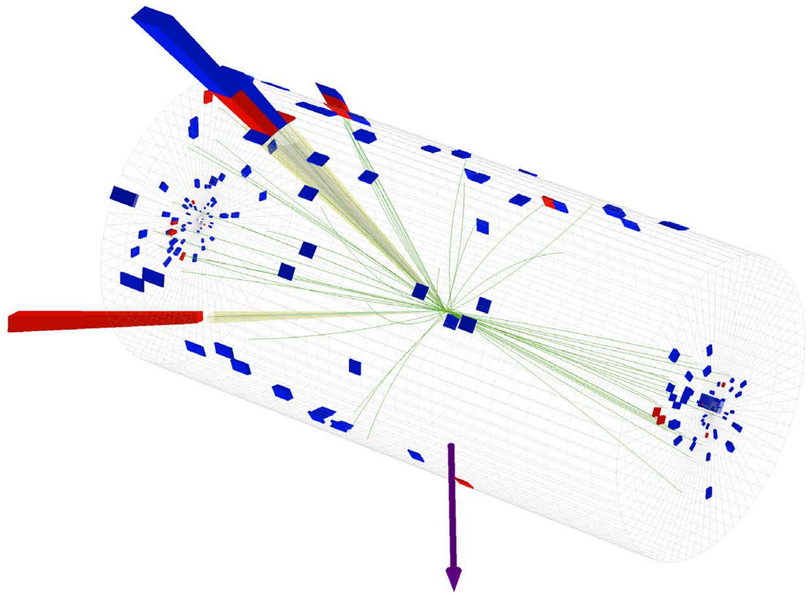


e.g., L. de Oliveira, M. Kagan, L. Mackey, B. Nachman and A. Schwartzman, arXiv: 1511.05190
(see also the review by Kagan, arXiv:2012.09719)

- Convert to 2D/3D image => **Computer vision**
  - then use convolutional neural networks (CNNs)
  - but: inhomogeneous geometry, high sparsity, …

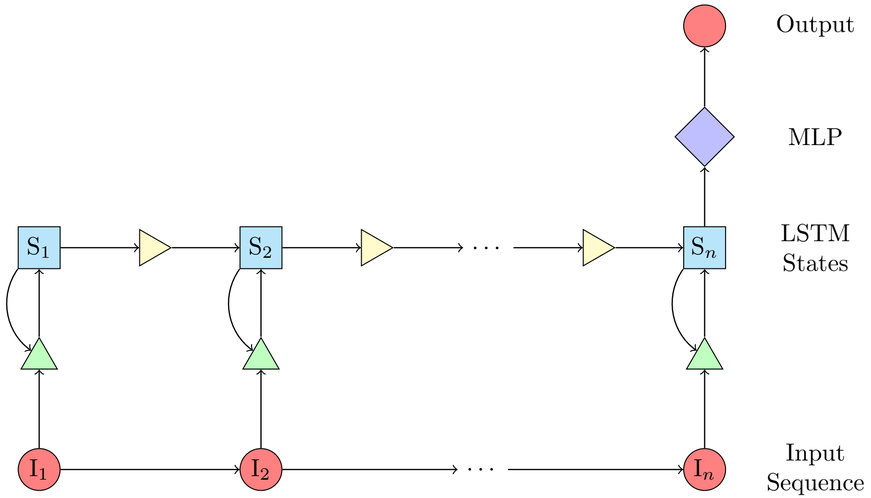
Data Representation: Sequence

HEP



Collision events, detector hits, sensor arrays, …

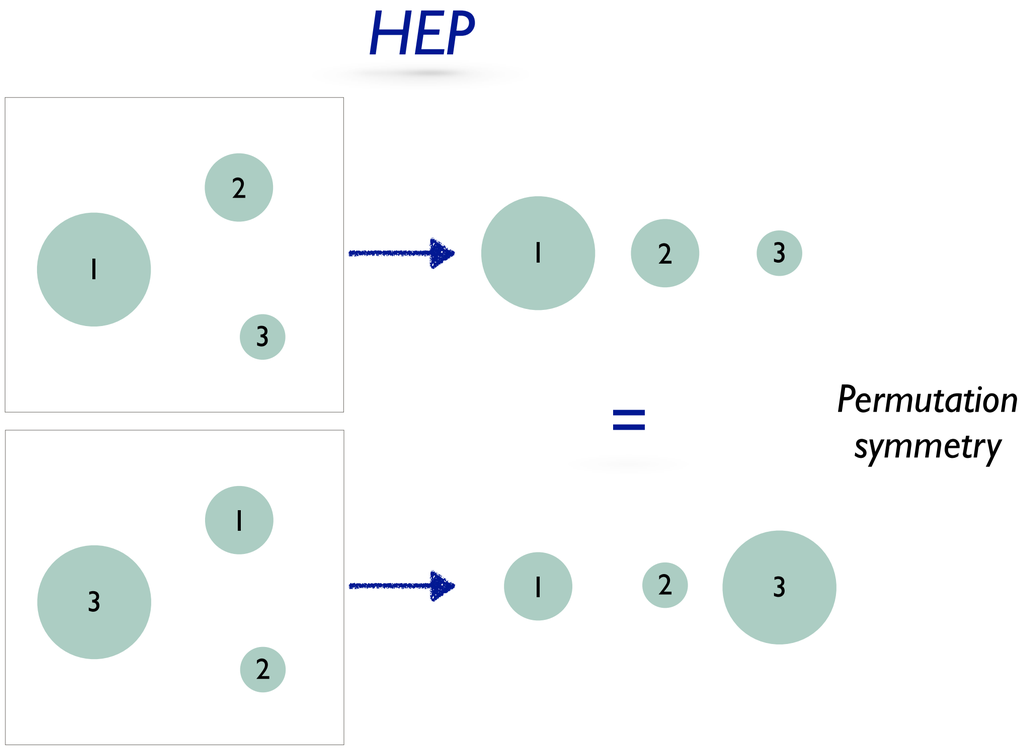
Sequence

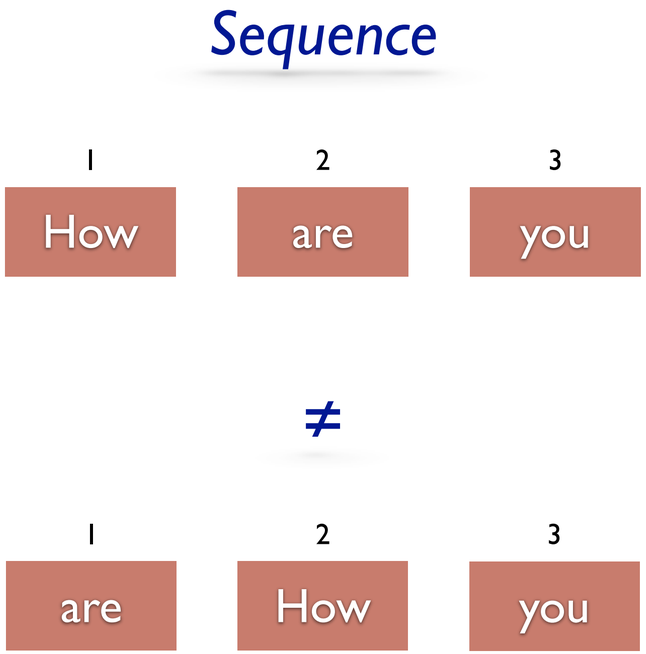


e.g., Guest, Collado, Baldi, Hsu, Urban, Whiteson
arXiv: 1607.08633

- Convert to a sequence => **Natural language processing (NLP)**
  - recurrent neural network (RNN), e.g., GRU/LSTM; 1D CNNs; etc.

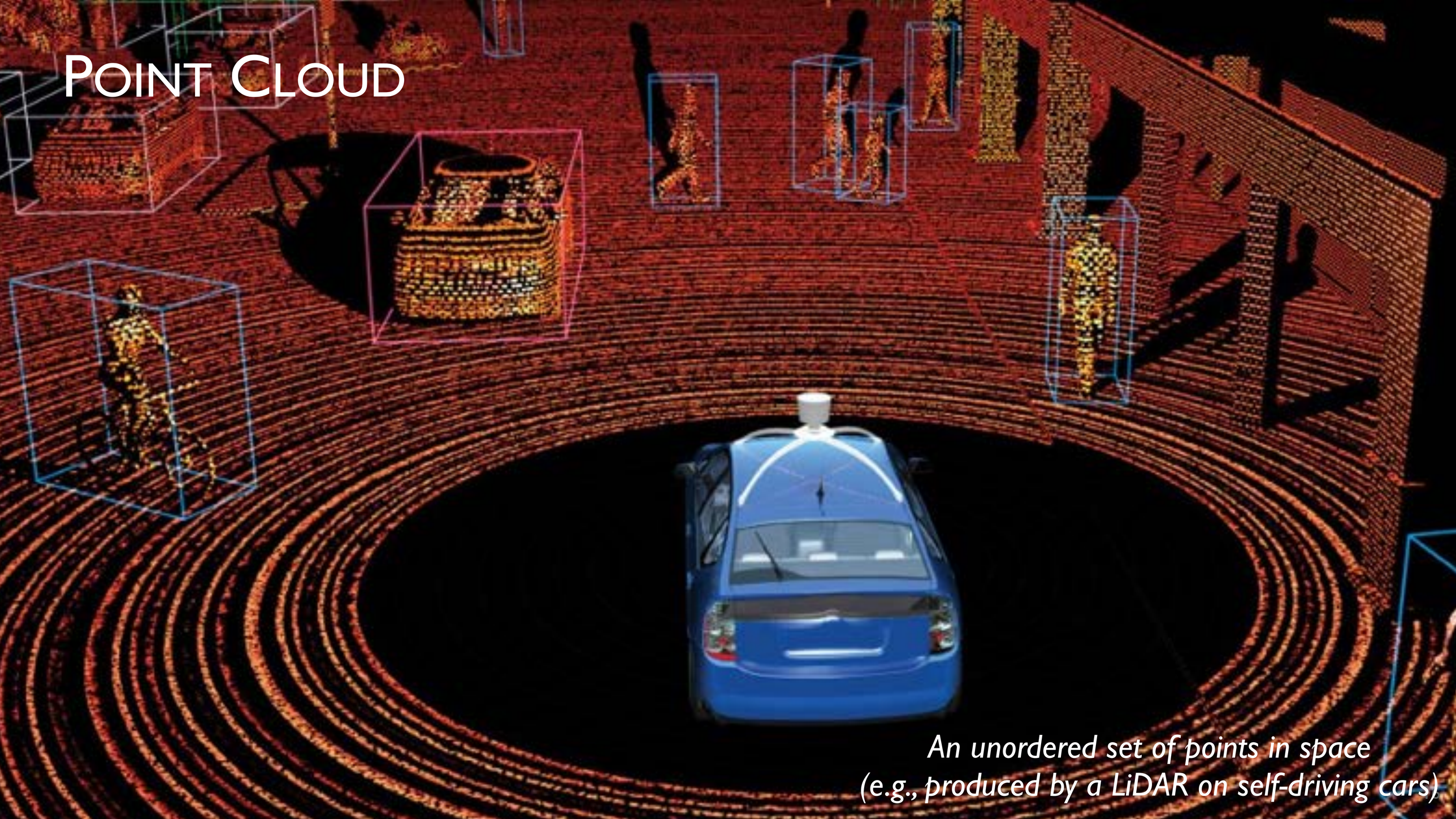
DATA REPRESENTATION: SEQUENCE?





- Convert to a sequence => **Natural language processing (NLP)**
  - recurrent neural network (RNN), e.g., GRU/LSTM; 1D CNNs; etc.
  - but: must impose an ***ordering*** on the particles/hits, which can limit the learning performance

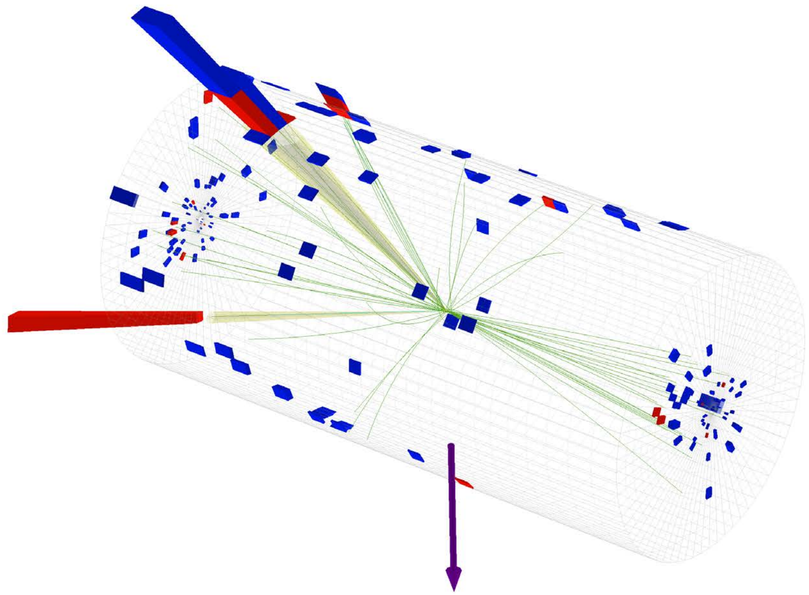
Point Cloud



An unordered set of points in space
(e.g., produced by a LiDAR on self-driving cars)

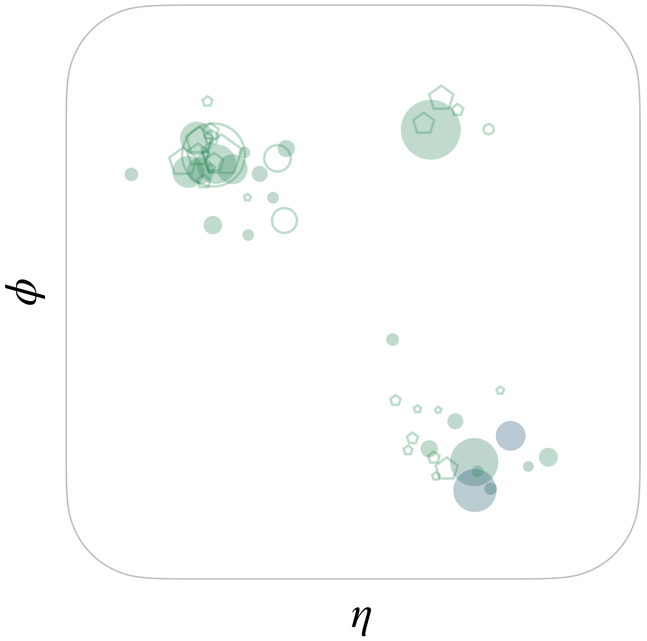
Data Representation: Point Cloud

HEP



Collision events, detector hits, sensor arrays, …

Point cloud

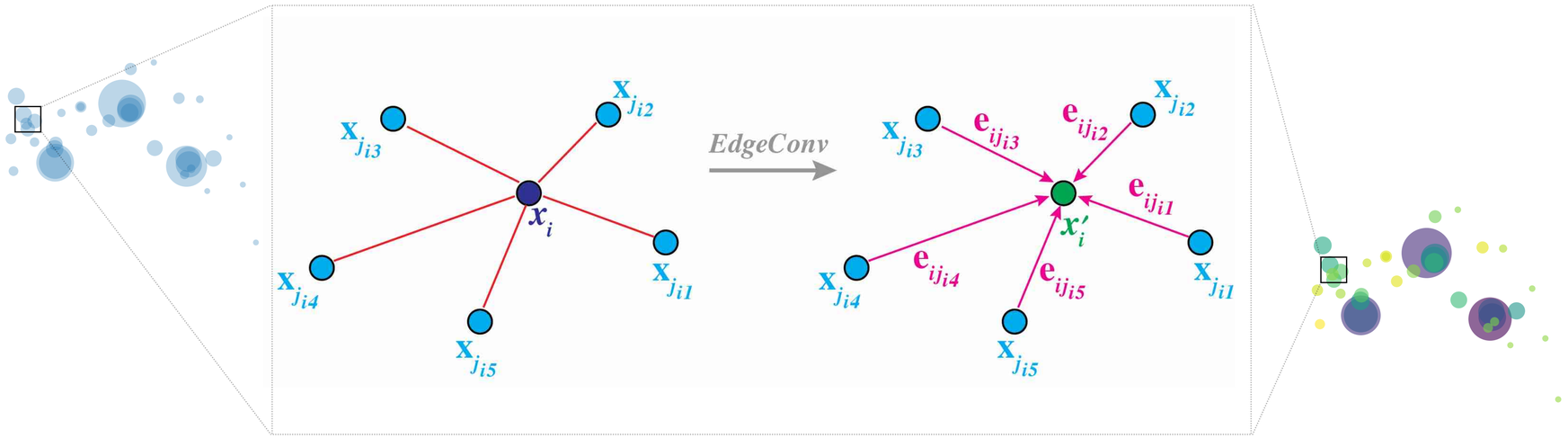


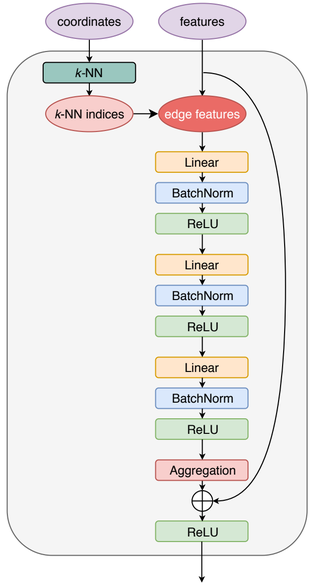
- HEP data as a point cloud
  - each particle / detector cell is a point in the cloud
    - for each point: (spatial) coordinates + any additional properties (energy/momentum, detector response, …)
  - key feature: ***permutation symmetry*** — *algorithm output should not depend on the input ordering*

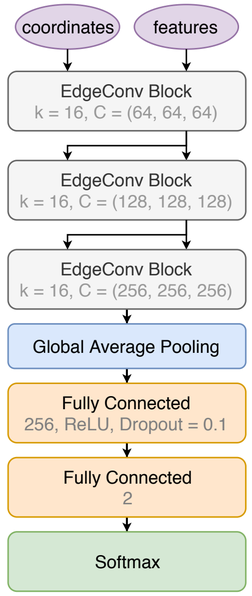
PARTICLENET

HQ and L. Gouskos
Phys.Rev.D 101 (2020) 5, 056019

- ParticleNet: jet tagging via particle clouds
  - treating a jet as an **unordered set of particles**, distributed in the $\eta - \phi$ space
  - **graph neural network architecture**, adapted from DGCNN [arXiv:1801.07829]
    - treating a point cloud as a graph: each point is a vertex
      - for each point, a local patch is defined by finding its k-nearest neighbors
    - designing a permutation-invariant "convolution" function
      - define "edge feature" for each center-neighbor pair: $e_{ij} = \text{MLP}(x_i, x_j)$
      - aggregate the edge features in a symmetric way: $x_i' = \text{mean}_j\, e_{ij}$







Performance on top quark tagging benchmark
[SciPost Phys. 7, 014 (2019)]

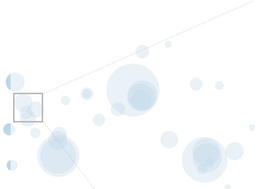
	$1/\varepsilon_b$ at $\varepsilon_s = 30\%$
ResNeXt-50	1147 ± 58
P-CNN	759 ± 24
PFN	888 ± 17
ParticleNet-Lite	1262 ± 49
ParticleNet	$\mathbf{1615 \pm 93}$

See also P.T. Komiske, E. M. Metodiev and J. Thaler, *JHEP 01 (2019) 121*

PARTICLENET

HQ and L. Gouskos
Phys.Rev.D 101 (2020) 5, 056019

- ParticleNet: jet tagging via particle clouds
  - treating a jet as an **unordered**
  - **graph neural network architec**
    - treating a point cloud as a gr
      - for each point, a local patch i
    - designing a permutation-inva
      - de
      - ac



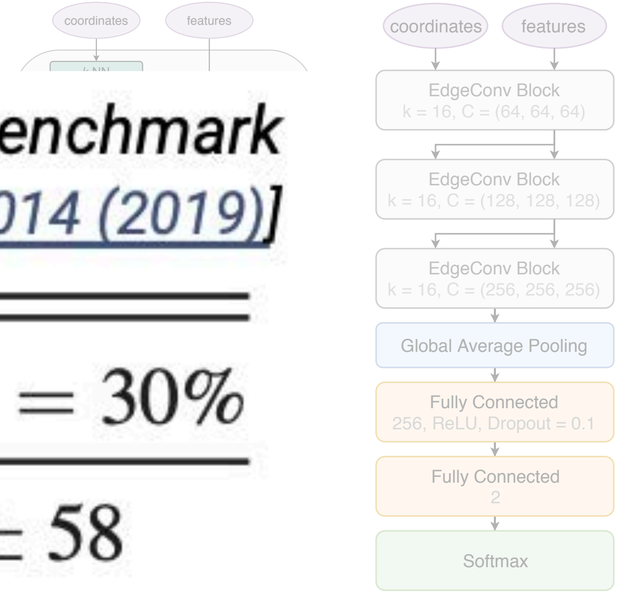
Image

Sequence

Point cloud

Performance on top quark tagging benchmark
[SciPost Phys. 7, 014 (2019)]

	$1/\varepsilon_b$ at $\varepsilon_s = 30\%$
ResNeXt-50	1147 ± 58
P-CNN	759 ± 24
PFN	888 ± 17
ParticleNet-Lite	1262 ± 49
ParticleNet	$\mathbf{1615 \pm 93}$

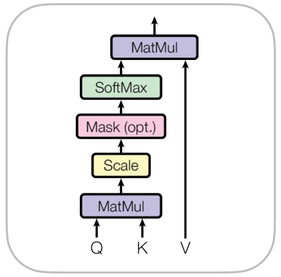


tagging benchmark
st Phys. 7, 014 (2019)]
$/\varepsilon_b$ at $\varepsilon_s = 30\%$
1147 ± 58
759 ± 24
888 ± 17
1262 ± 49
$\mathbf{1615 \pm 93}$

See also P.T. Komiske, E. M. Metodiev and J. Thaler, *JHEP 01 (2019) 121*

Evolution of Jet Representations

image, sequence, point cloud, …



Attention Is All You Need?

or how physics can help…

Towards a Foundation Model

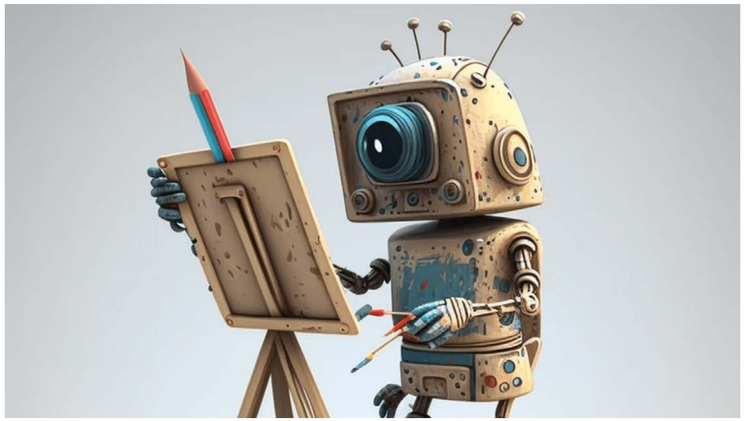
scaling, self-supervision, and more…

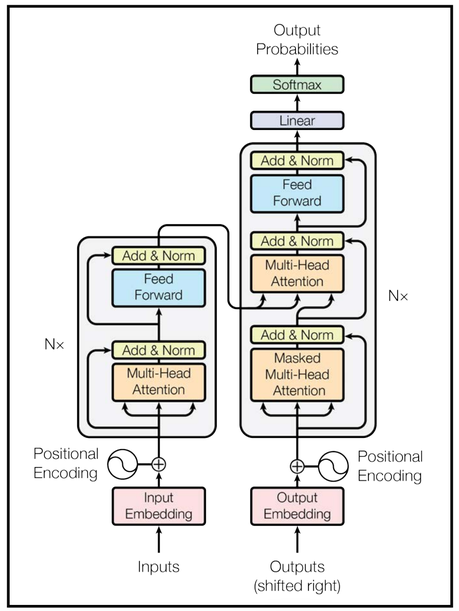
Attention Is All You Need?

- **Transformers** are taking over every task

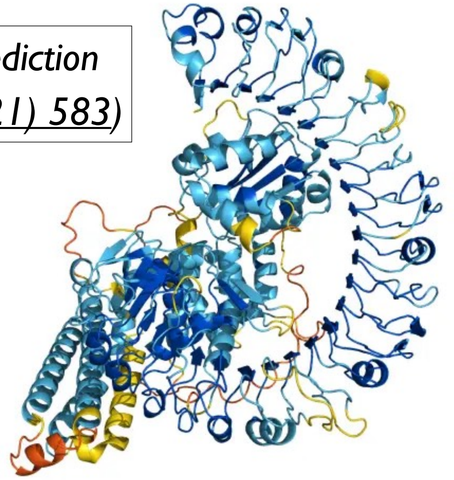
Large Language Models

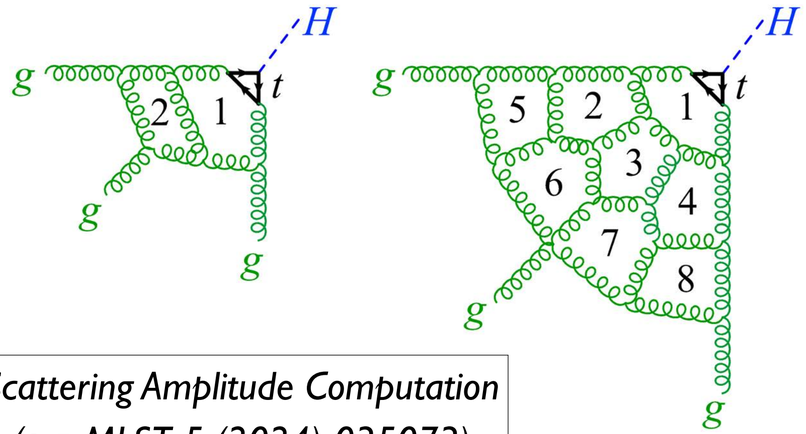
Image/Video Generation





Protein Structure Prediction
(e.g. Nature 596 (2021) 583)





Scattering Amplitude Computation
(e.g. MLST 5 (2024) 035073)

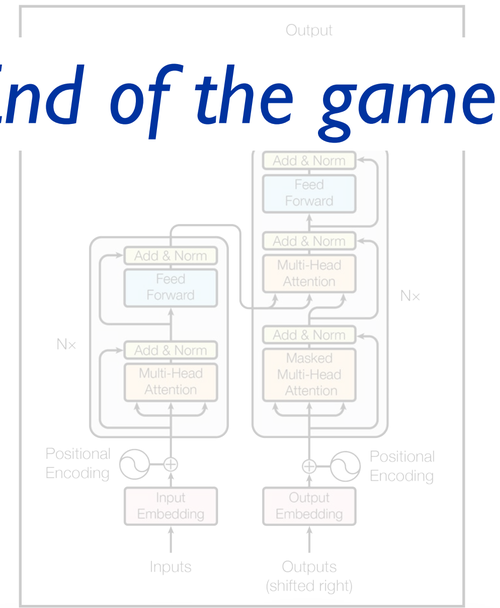
Attention Is All You Need?

- Transformers are taking over every task

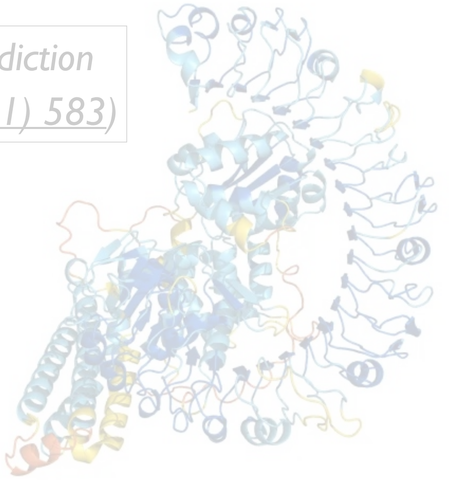
Large Language Models

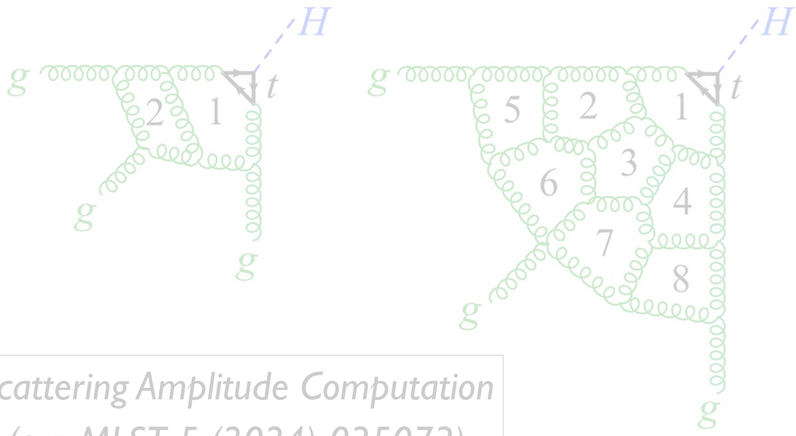
Image/Video Generation

End of the game?



Protein Structure Prediction
(e.g. Nature 596 (2021) 583)





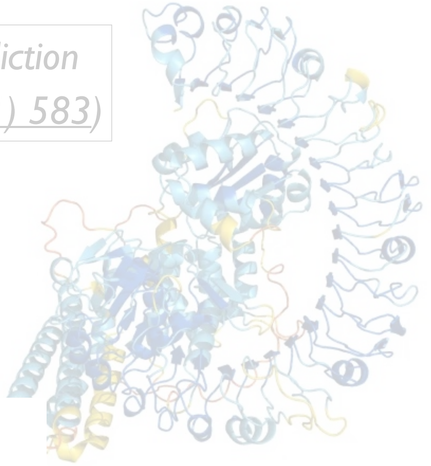
Scattering Amplitude Computation
(e.g. MLST 5 (2024) 035073)

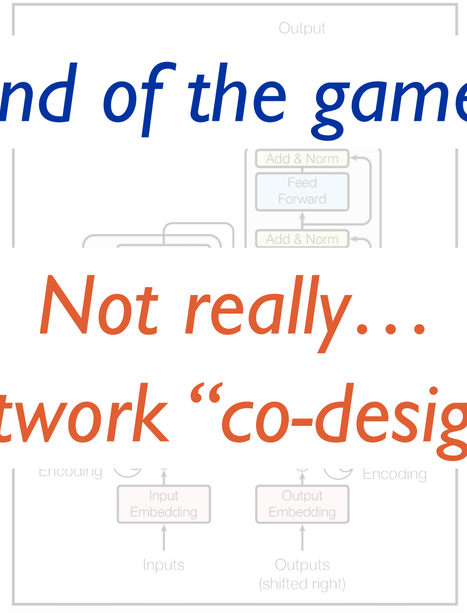
Attention Is All You Need?

- Transformers are taking over every task

Large Language Models

Protein Structure Prediction
(e.g. Nature 596 (2021) 583)





End of the game?

Not really…

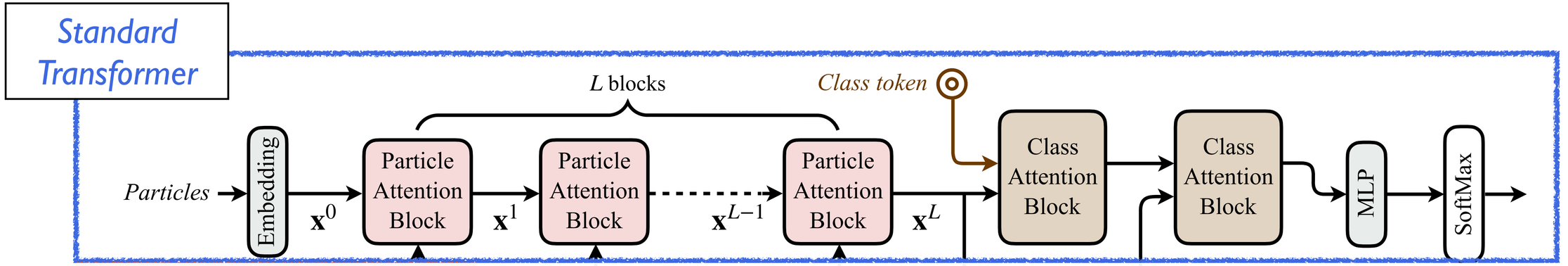
Physics + network "co-design" is the key.

Scattering Amplitude Computation
(e.g. MLST 5 (2024) 035073)

PARTICLE TRANSFORMER

HQ, C. Li, S. Qian, ICML 2022

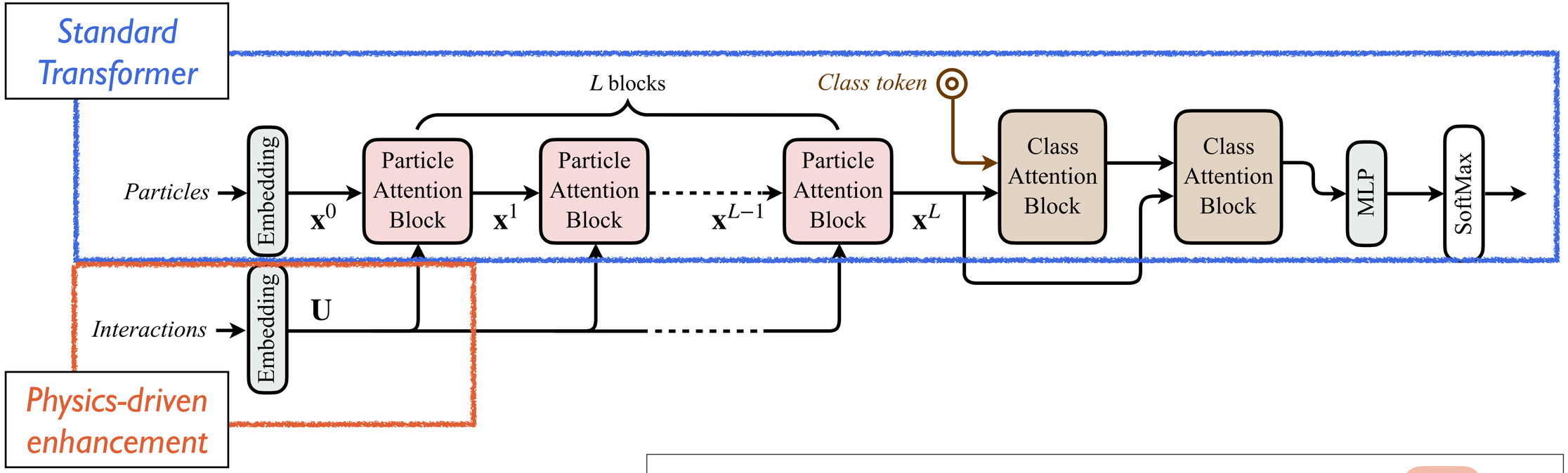
- **Particle Transformer** (ParT): Transformer model **tailored for particle physics**



PARTICLE TRANSFORMER

HQ, C. Li, S. Qian, ICML 2022

- **Particle Transformer** (ParT): Transformer model **tailored for particle physics**



$$\Delta = \sqrt{(y_a - y_b)^2 + (\phi_a - \phi_b)^2},$$
$$k_{\mathrm{T}} = \min(p_{\mathrm{T},a}, p_{\mathrm{T},b})\Delta,$$
$$z = \min(p_{\mathrm{T},a}, p_{\mathrm{T},b})/(p_{\mathrm{T},a} + p_{\mathrm{T},b}),$$
$$m^2 = (E_a + E_b)^2 - \|\mathbf{p}_a + \mathbf{p}_b\|^2,$$

$$\text{P-MHA}(Q, K, V) = \text{SoftMax}(QK^T/\sqrt{d_k} + \mathbf{U})V,$$

Injection of (physics-inspired) pairwise features to "bias" the dot-product self-attention

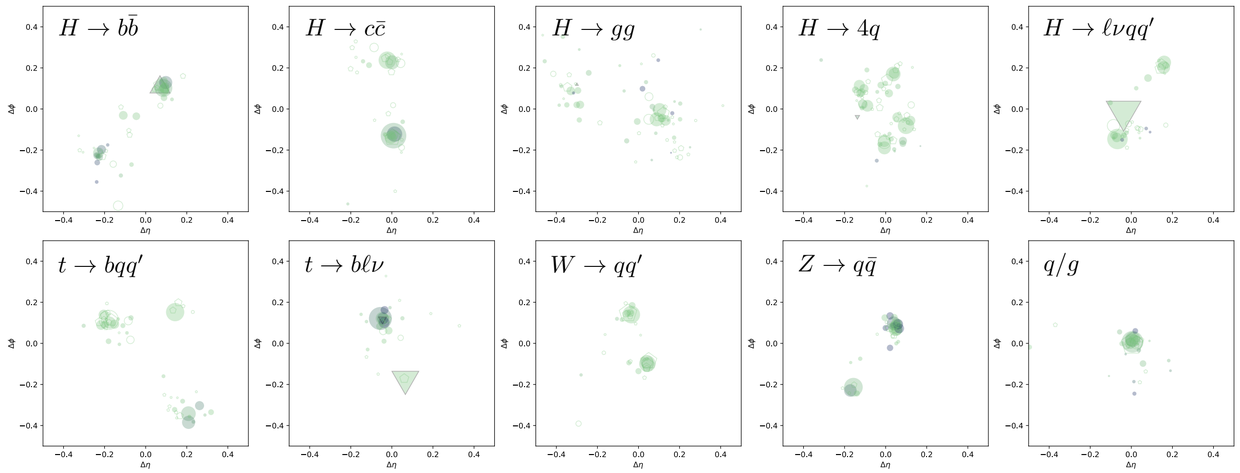
PARTICLE TRANSFORMER: PERFORMANCE

HQ, C. Li, S. Qian, ICML 2022

	All classes		$H \to b\bar{b}$	$H \to c\bar{c}$	$H \to gg$	$H \to 4q$	$H \to \ell\nu qq'$	$t \to bqq'$	$t \to b\ell\nu$	$W \to qq'$	$Z \to q\bar{q}$
	Accuracy	AUC	$\text{Rej}_{50\%}$	$\text{Rej}_{50\%}$	$\text{Rej}_{50\%}$	$\text{Rej}_{50\%}$	$\text{Rej}_{99\%}$	$\text{Rej}_{50\%}$	$\text{Rej}_{99.5\%}$	$\text{Rej}_{50\%}$	$\text{Rej}_{50\%}$
PFN	0.772	0.9714	2924	841	75	198	265	797	721	189	159
P-CNN	0.809	0.9789	4890	1276	88	474	947	2907	2304	241	204
ParticleNet	0.844	0.9849	7634	2475	104	954	3339	10526	11173	347	283
ParT	**0.861**	**0.9877**	**10638**	**4149**	**123**	**1864**	**5479**	**32787**	**15873**	**543**	**402**

JETCLASS dataset (100M jets)

- Particle Transformer (ParT):
  - significant gains compared to ParticleNet
    - 1.7% increase in accuracy
    - **up to 3x increase in background rejection** ($\text{Rej}_{X\%}$)



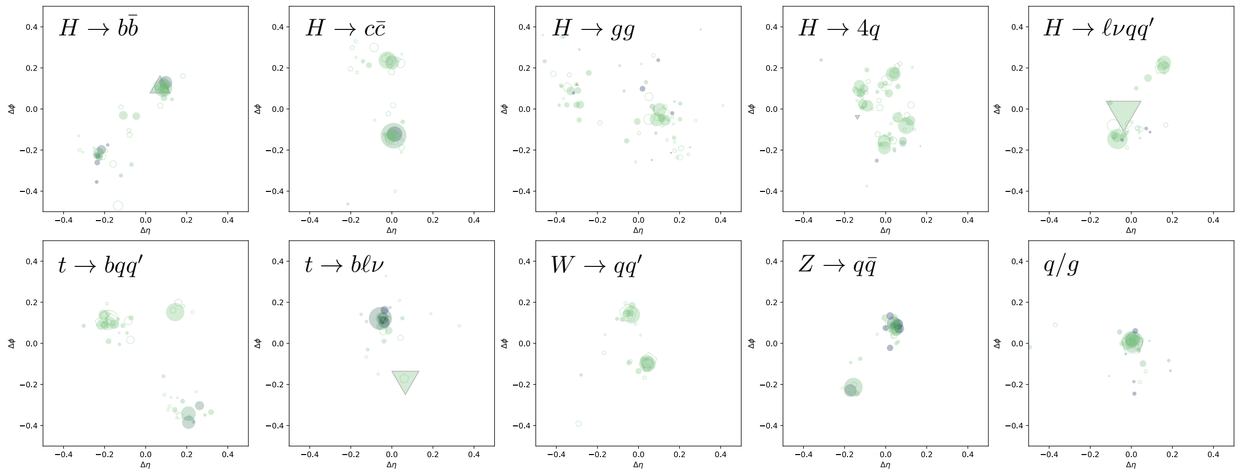
PARTICLE TRANSFORMER: PERFORMANCE

HQ, C. Li, S. Qian, ICML 2022

	All classes		$H \to b\bar{b}$	$H \to c\bar{c}$	$H \to gg$	$H \to 4q$	$H \to \ell\nu qq'$	$t \to bqq'$	$t \to b\ell\nu$	$W \to qq'$	$Z \to q\bar{q}$
	Accuracy	AUC	$\text{Rej}_{50\%}$	$\text{Rej}_{50\%}$	$\text{Rej}_{50\%}$	$\text{Rej}_{50\%}$	$\text{Rej}_{99\%}$	$\text{Rej}_{50\%}$	$\text{Rej}_{99.5\%}$	$\text{Rej}_{50\%}$	$\text{Rej}_{50\%}$
PFN	0.772	0.9714	2924	841	75	198	265	797	721	189	159
P-CNN	0.809	0.9789	4890	1276	88	474	947	2907	2304	241	204
ParticleNet	0.844	0.9849	7634	2475	104	954	3339	10526	11173	347	283
ParT	**0.861**	**0.9877**	**10638**	**4149**	**123**	**1864**	**5479**	**32787**	**15873**	**543**	**402**
ParT (plain)	0.849	0.9859	9569	2911	112	1185	3868	17699	12987	384	311

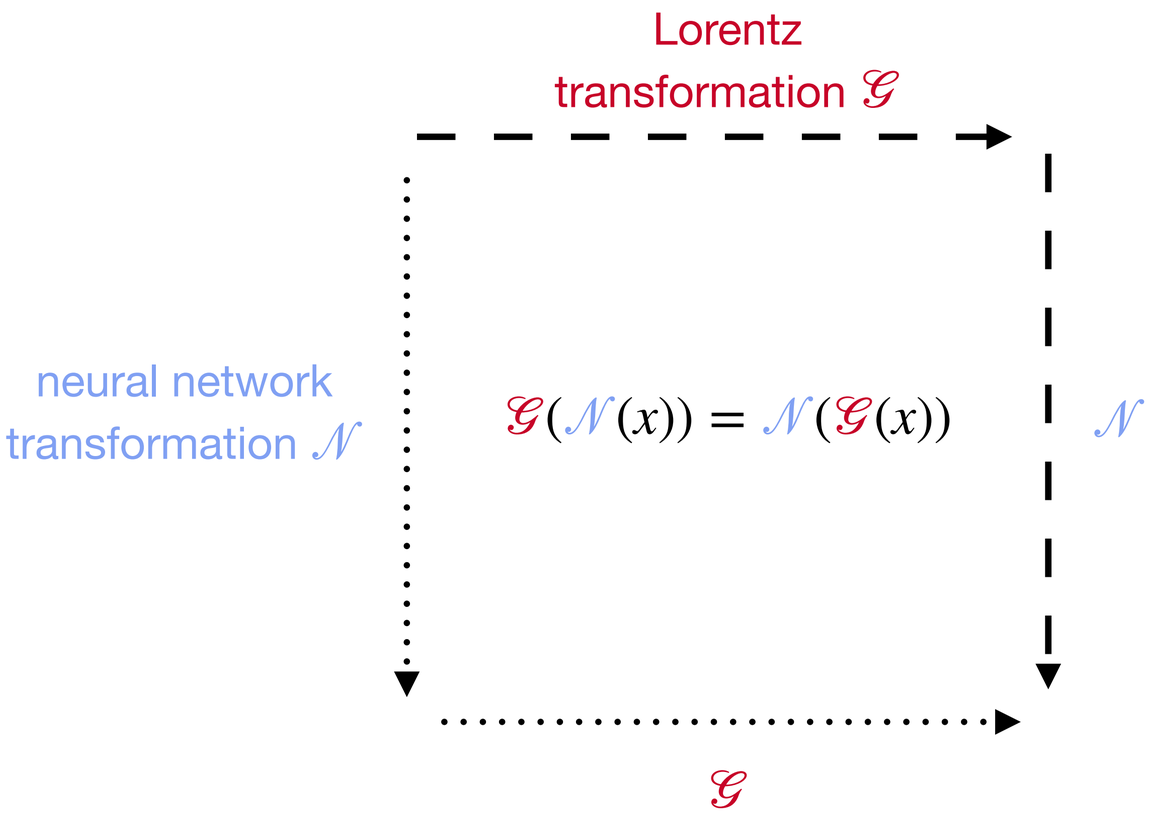
JETCLASS dataset (100M jets)

- Particle Transformer (ParT):
  - significant gains compared to ParticleNet
    - 1.7% increase in accuracy
    - **up to 3x increase in background rejection** ($\text{Rej}_{X\%}$)
- ParT (plain): Transformer w/o pairwise features
  - 1.2% drop in accuracy compared to full ParT
  - **Physics-driven modification plays a key role!**



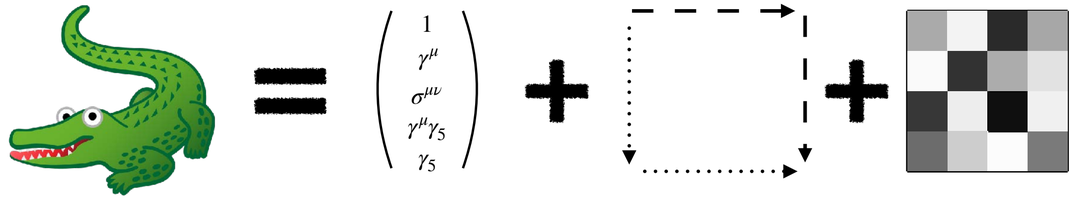
Symmetry-Aware Architectures

- One step further: symmetry-aware architectures
  - one important symmetry for particle physics: Lorentz symmetry
- Lorentz-equivariant architectures
  - equivariance (covariance) — preserve the symmetry by construction



L-GATr & LLoCa

L-GATr: Lorentz-Equivariant Geometric Algebra Transformer

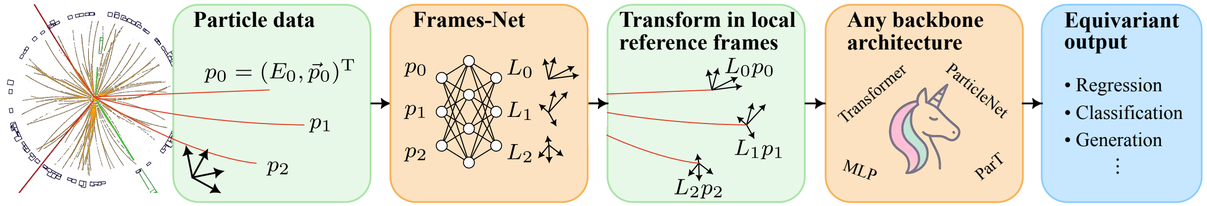


$$x = x^S\,1 + x^V_\mu\,\gamma^\mu + x^B_{\mu\nu}\,\sigma^{\mu\nu} + x^A_\mu\,\gamma^\mu\gamma^5 + x^P\,\gamma^5$$

Specialized Lorentz-equivariant linear/attention/norm layers based on geometric algebra representations.

J. Brehmer, V. Bresó, P. Haan, T. Plehn, HQ, J. Spinner and J. Thaler, arXiv: 2411.00446

LLoCa: Lorentz Local Canonicalization



Learned local reference frames to transform the inputs into a (learned) canonical frame and then the outputs back.
Can be applied to any architecture (GNN, transformer).

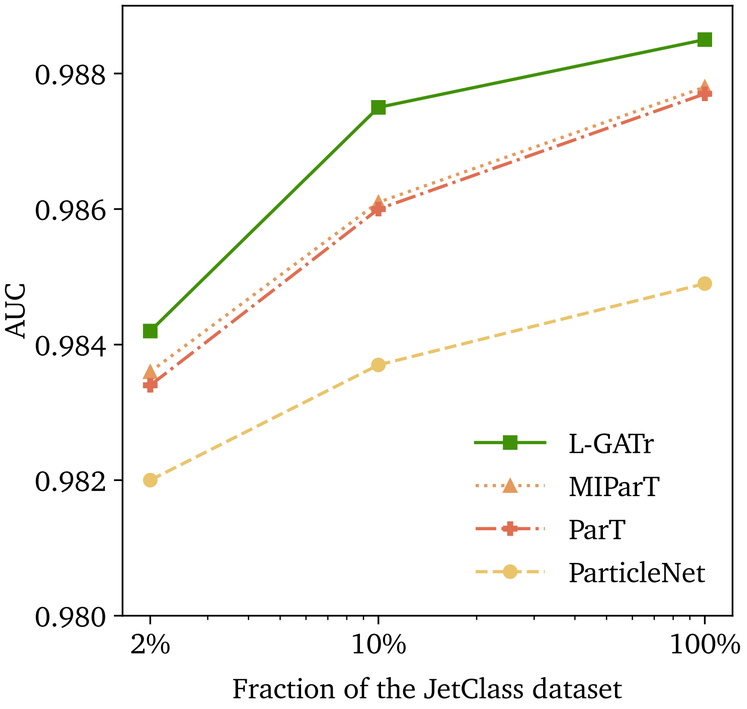
L. Favaro, G. Gerhartz, F.A. Hamprecht, P. Lippmann, S. Pitz, T. Plehn, HQ and J. Spinner, arXiv: 2508.14898

L-GATr & LLoCa Performance

Performance on JetClass (100M)

Network	Accuracy	AUC	Time	FLOPs	Memory	Parameters
MIParT-L [26]	0.861	0.9878	43h	225M	53.6G	2380k
LorentzNet* [11]	0.847	0.9856	64h	676M	20.5G	223k
PELICAN-lite* [17]	0.851	0.9862	97h	1370M	27.4G	244k
ParticleNet [9]	0.844	0.9849	25h	413M	16.5G	366k
LLoCa-ParticleNet* [17]	0.848	0.9857	43h	517M	23.5G	385k
ParT [18]	0.861	0.9877	33h	211M	13.3G	2141k
LLoCa-ParT* [17]	0.864	0.9882	66h	315M	19.9G	2160k
Transformer [17]	0.855	0.9867	15h	210M	2.3G	1979k
LLoCa-Transformer* [17]	0.864	0.9882	28h	219M	4.1G	1980k
L-GATr* [16]	0.866	0.9885	166h	2060M	19.0G	1079k
L-GATr-slim*	0.866	0.9885	27h	329M	8.1G	2031k

Scaling w/ training set size



* *L-GATr-slim: a more efficient version of L-GATr implemented using only Lorentz scalar and vectors.*

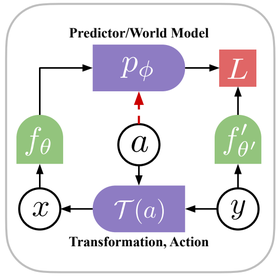
A. Petitjean, T. Plehn, J. Spinner and U. Köthe, arXiv: 2512.17011

Evolution of Jet Representations

image, sequence, point cloud, …

Attention Is All You Need?

or how physics can help…

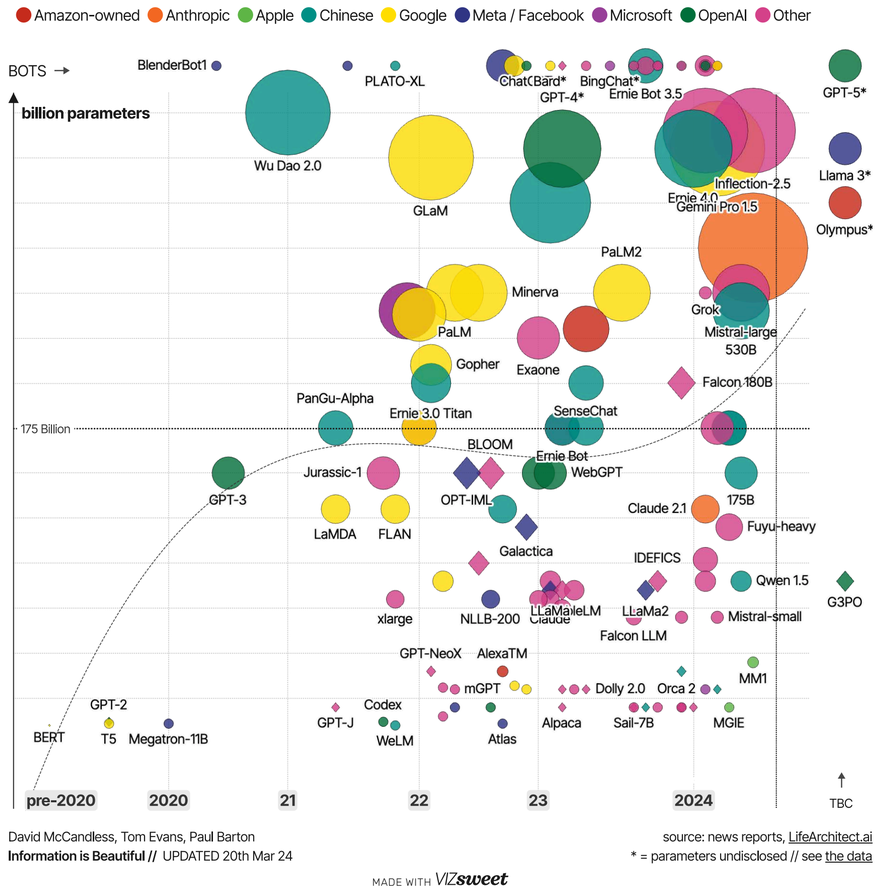


Towards a Foundation Model

scaling, self-supervision, and more…

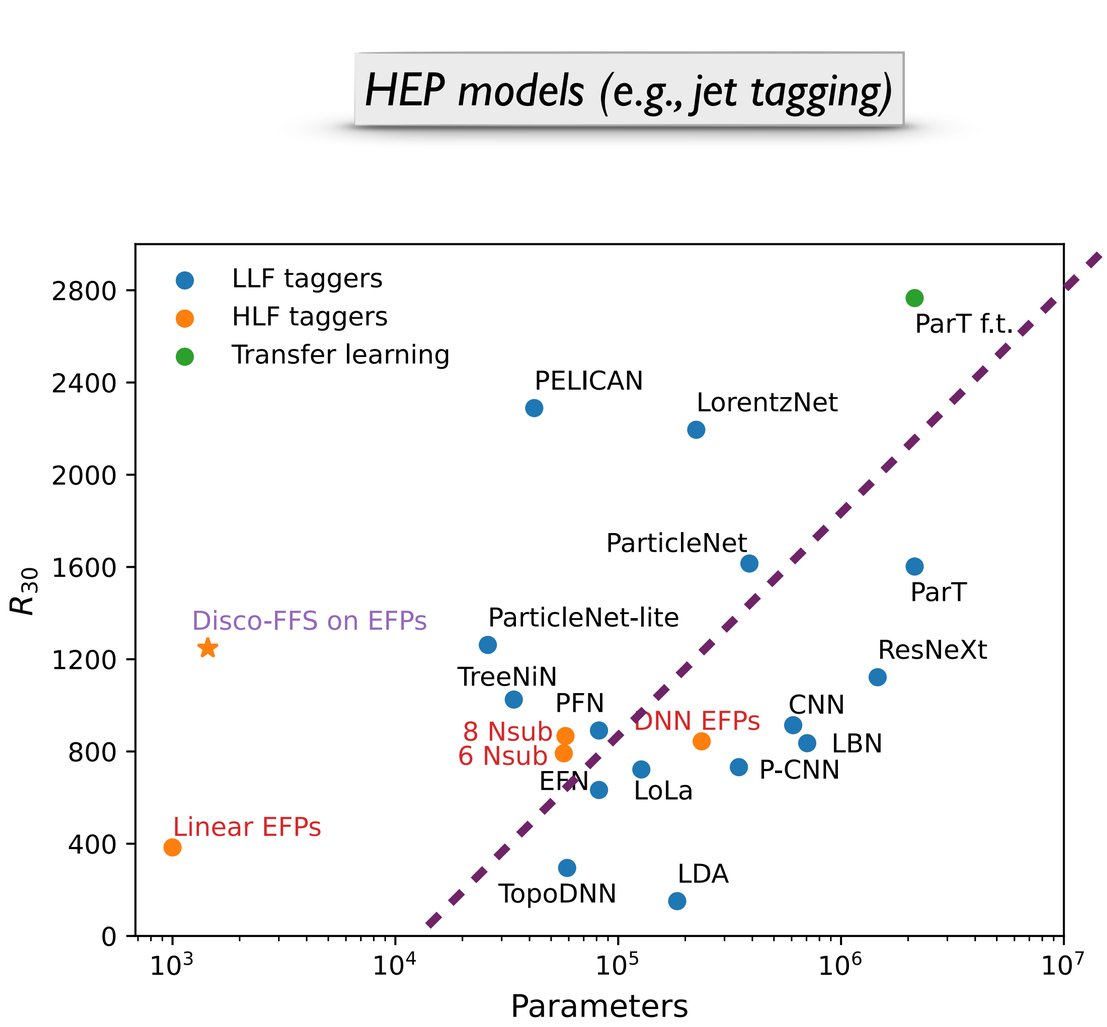
What's Next?

Natural language models



Source: informationisbeautiful.net

HEP models (e.g., jet tagging)



R. Das, G. Kasieczka and D. Shih, PRD 109, 054009

Scaling Laws

- Neural scaling law [arXiv: 2001.08361, 2203.15556]
  - **question:** for a fixed compute budget (C), how to determine the optimal dataset size (D) and model parameters (N), such that the loss is minimized?
  - **empirical power law:**

$$\hat{L}(N, D) \triangleq E + \frac{A}{N^{\alpha}} + \frac{B}{D^{\beta}}.$$

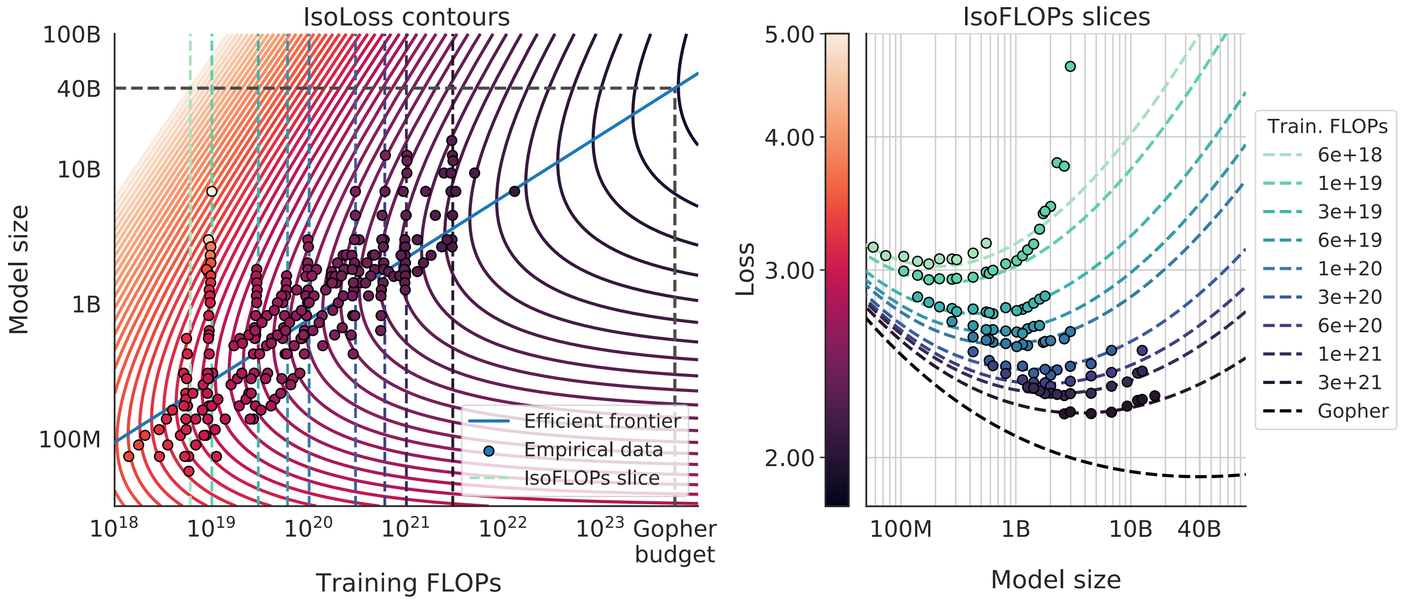
  - for a standard transformer:

$$\mathrm{FLOPs}(N, D) \approx 6ND$$

  - **compute-optimal scaling:**

$$N_{opt}(C) = G\left(\frac{C}{6}\right)^{a}, \quad D_{opt}(C) = G^{-1}\left(\frac{C}{6}\right)^{b},$$

$$\text{where} \quad G = \left(\frac{\alpha A}{\beta B}\right)^{\frac{1}{\alpha+\beta}}, \quad a = \frac{\beta}{\alpha+\beta}, \text{ and } b = \frac{\alpha}{\alpha+\beta}.$$



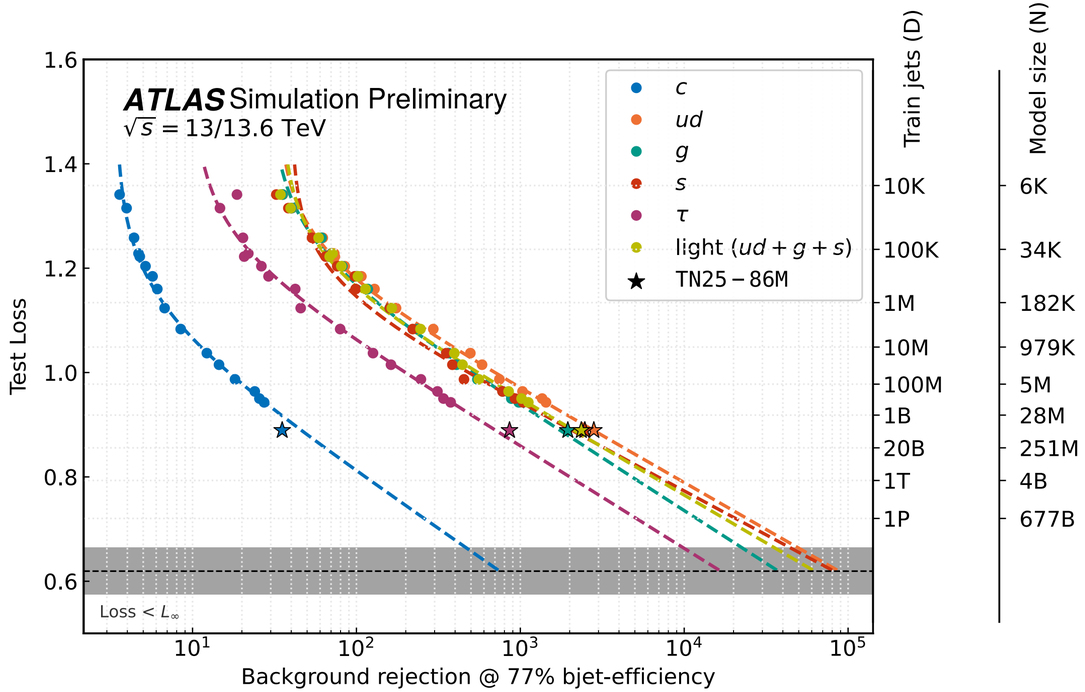
Interestingly, for LLMs, the optimal scaling is found to be $a \approx b \approx 0.50$ — same scaling for data and parameters.

JET SCALING LAW: STANDARD TRANSFORMER

$$L(N, D) = L_\infty + \frac{A}{N^\alpha} + \frac{B}{D^\beta},$$

Jet flavor tagging (Small-R)

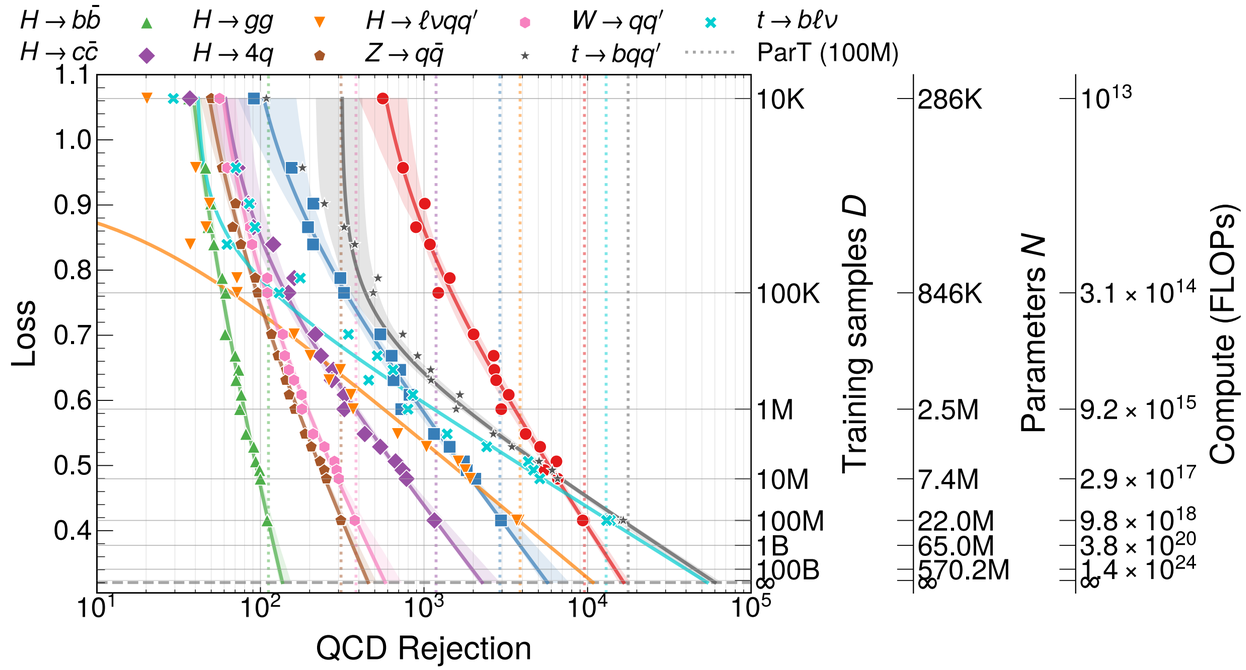
ATL-SOFT-PUB-2026-002



Quantity	Symbol	Description	Value
Irreducible loss	L_∞	$L(N_{\to\text{inf}}, D_{\to\text{inf}})$	0.619 (0.539, 0.671)
Model scaling exponent	α	$L_N \propto N^{-\alpha}$	0.677 (0.644, 0.764)
Data scaling exponent	β	$L_D \propto D^{-\beta}$	0.077 (0.065,0.086)
$C = 6ND$ optimal exponent	γ	$L \propto C^{-\gamma}$	0.07 (0.064, 0.074)

Boosted jet tagging (Large-R)

M.Vigl, N. Hartman, M. Kagan and L. Heinrich, arXiv:2602.15781



Parameter	Value (68% CI)
L_∞	0.32 (0.29, 0.34)
A	11.27 (8.01, 15.14)
α	0.44 (0.40, 0.48)
B	7.22 (6.66, 7.89)
β	0.22 (0.21, 0.23)
γ	0.15 (0.13, 0.16)

JET SCALING LAW: STANDARD TRANSFORMER

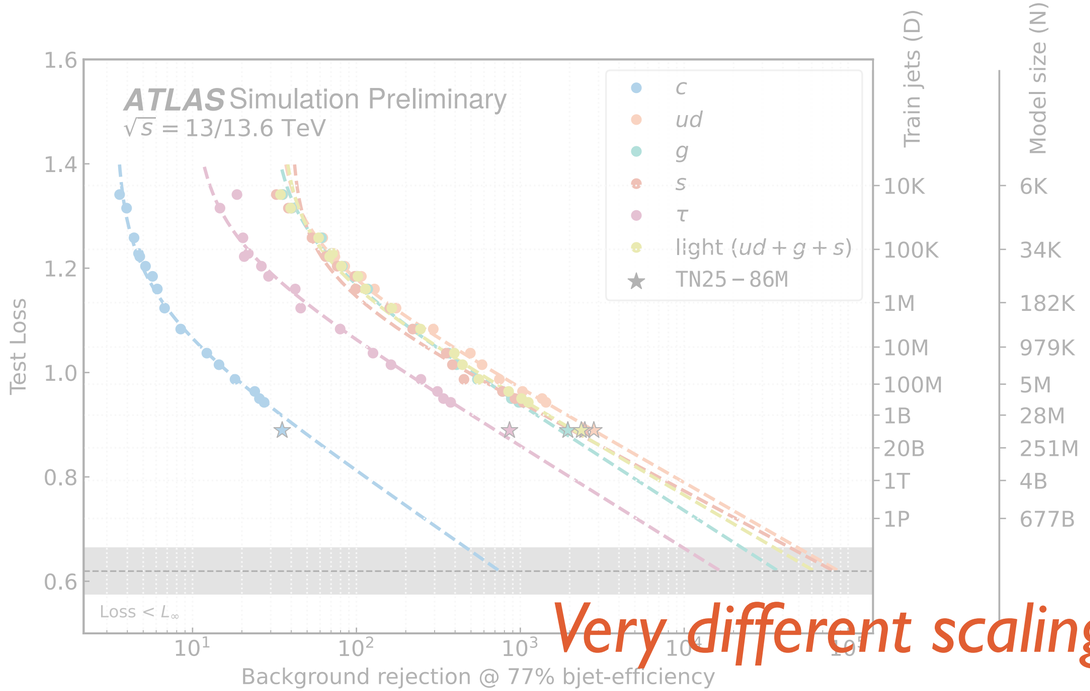
$$L(N, D) = L_\infty + \frac{A}{N^\alpha} + \frac{B}{D^\beta},$$

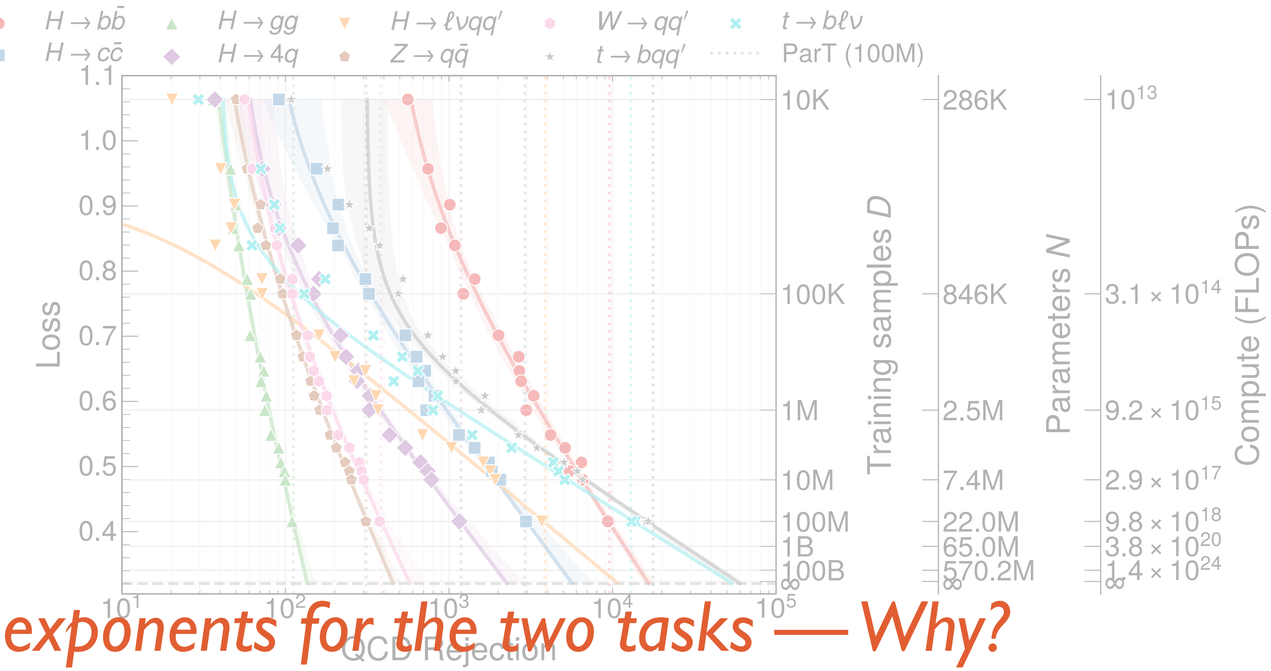
Jet flavor tagging (Small-R)

Boosted jet tagging (Large-R)

ATL-SOFT-PUB-2026-002

M.Vigl, N. Hartman, M. Kagan and L. Heinrich, arXiv:2602.15781





Very different scaling exponents for the two tasks — Why?

Quantity	Symbol	Description	Value
Irreducible loss	L_∞	$L(N_{\to\text{inf}}, D_{\to\text{inf}})$	0.619 (0.539, 0.671)
Model scaling exponent	α	$L_N \propto N^{-\alpha}$	0.677 (0.644, 0.764)
Data scaling exponent	β	$L_D \propto D^{-\beta}$	0.077 (0.065,0.086)
$C = 6ND$ optimal exponent	γ	$L \propto C^{-\gamma}$	0.07 (0.064, 0.074)

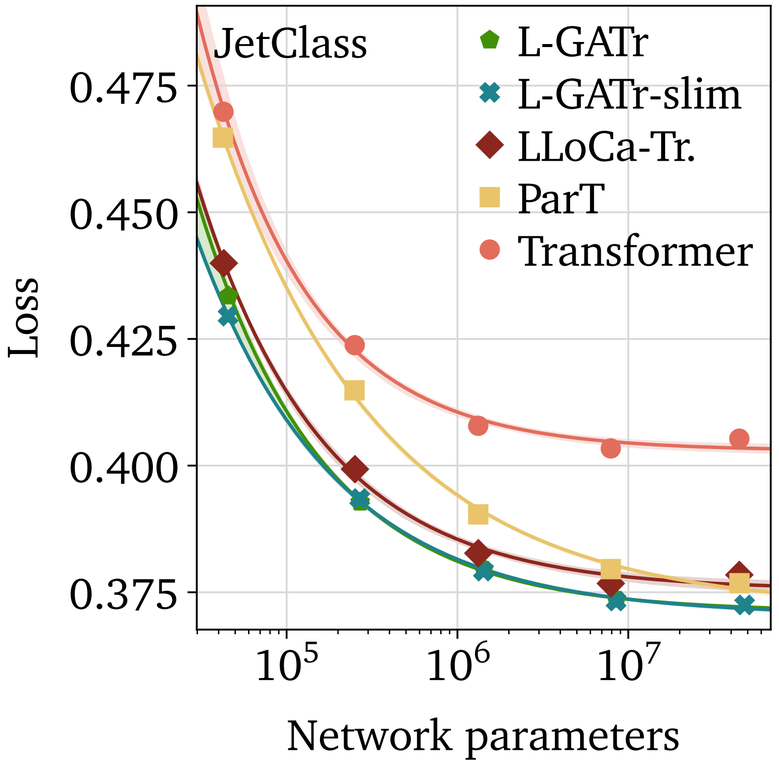
Parameter	Value (68% CI)
L_∞	0.32 (0.29, 0.34)
A	11.27 (8.01, 15.14)
α	0.44 (0.40, 0.48)
B	7.22 (6.66, 7.89)
β	0.22 (0.21, 0.23)
γ	0.15 (0.13, 0.16)

JET SCALING LAW: EQUIVARIANT TRANSFORMERS

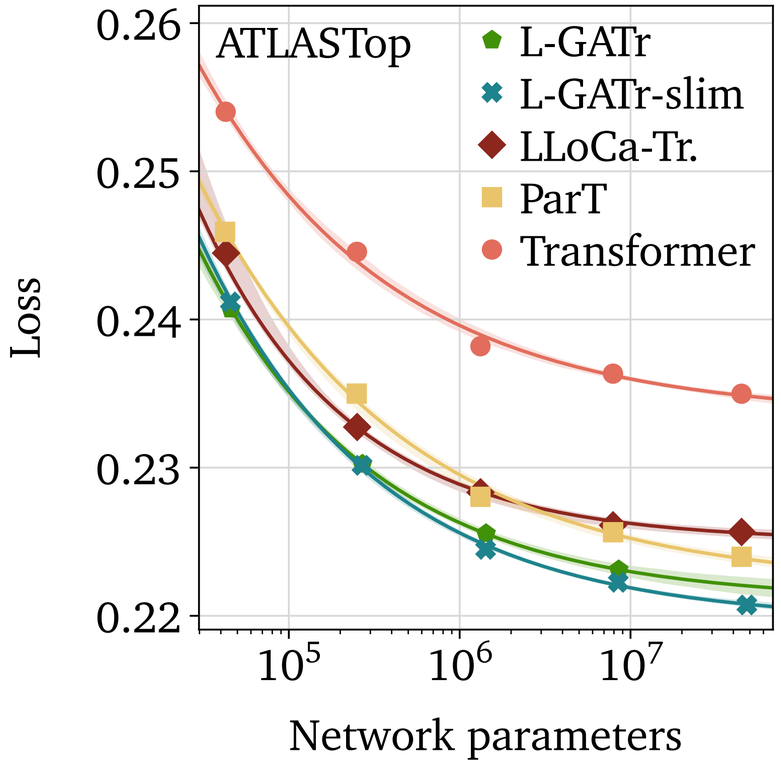
L. Favaro, T. Plehn, HQ, J. Spinner
arXiv: 2608.02735

- How do Lorentz equivariant transformers scale?
  - fixed dataset size (~100M); only varying model sizes

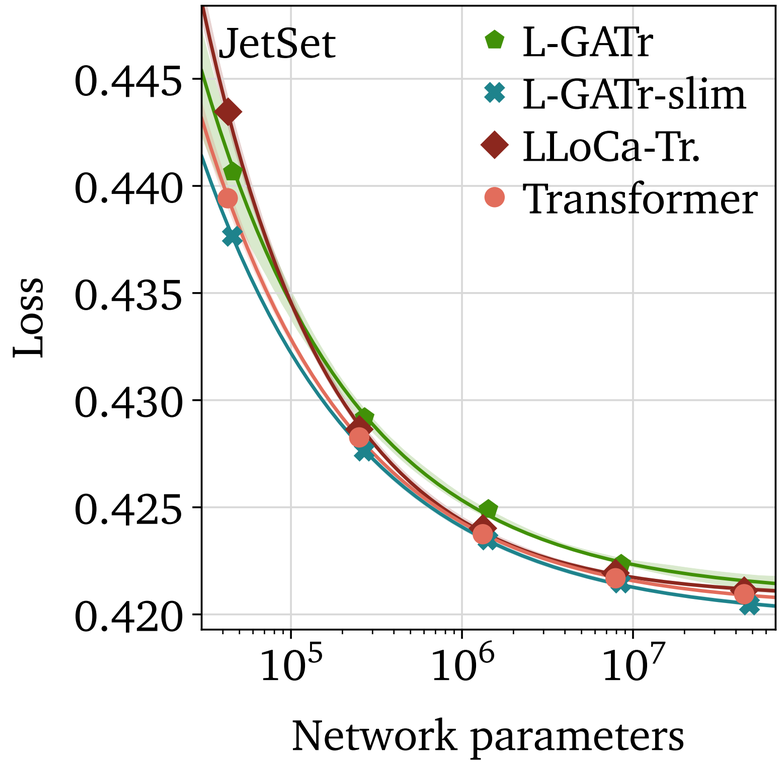
Boosted jet tagging (JetClass, all features)



Boosted jet tagging (Top vs QCD, 4-vector only)



Jet flavor tagging (Small-R jets, tracks only)



JET SCALING LAW: EQUIVARIANT TRANSFORMERS

L. Favaro, T. Plehn, HQ, J. Spinner
arXiv: 2608.02735

- How do Lorentz equivariant transformers scale?
  - fixed dataset size (~100M); only varying model sizes

Boosted jet tagging (JetClass, all features)

Boosted jet tagging (Top vs QCD, 4-vector only)

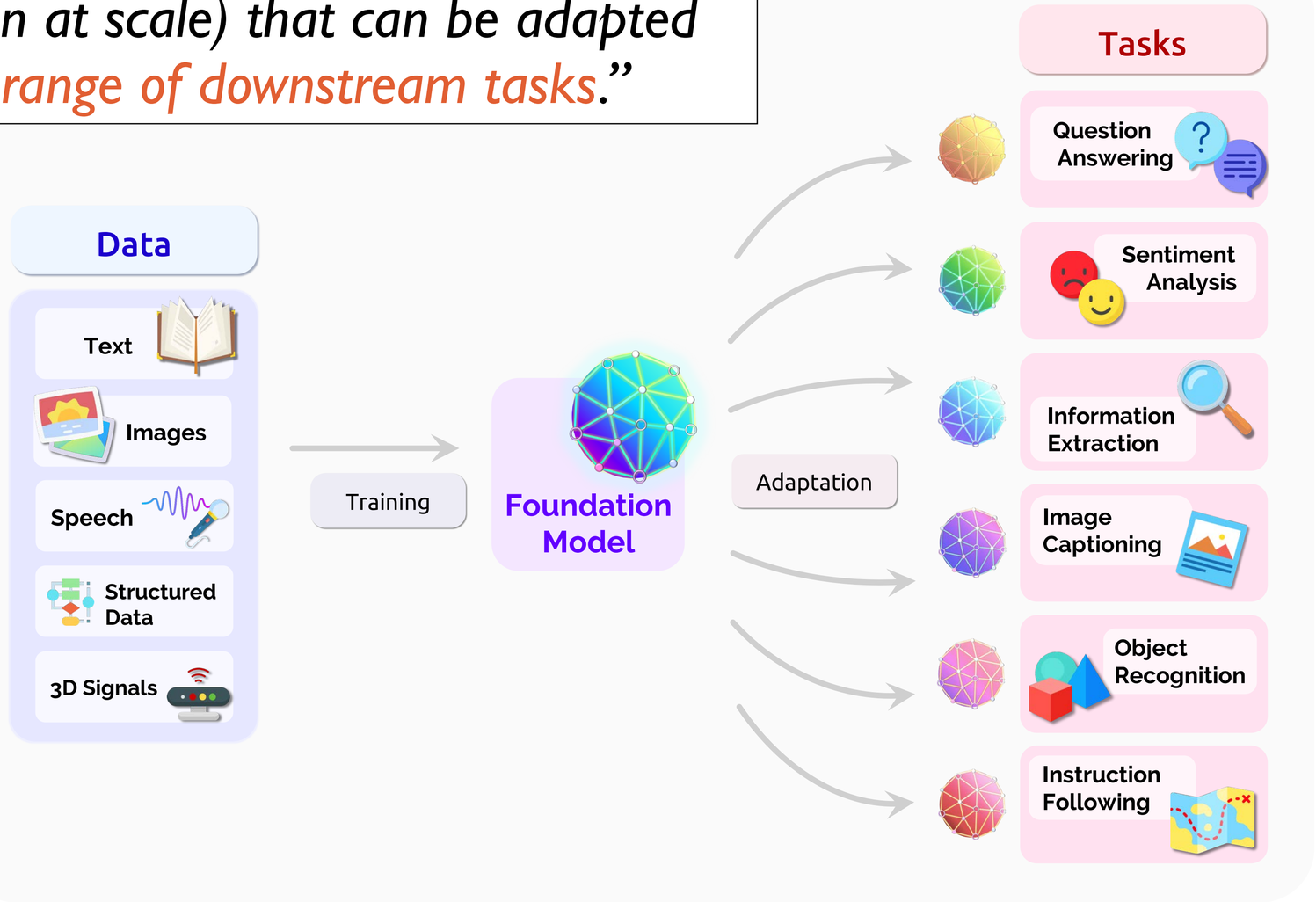
Jet flavor tagging (Small-R jets, tracks only)

$$\hat{L}(N) = L_\infty + \frac{A}{N^\alpha},$$

	JetClass		ATLASTop		JetSet	
	α	L_∞	α	L_∞	α	L_∞
Transformer	$0.70^{+0.06}_{-0.07}$	$0.4030^{+0.0010}_{-0.0014}$	$0.39^{+0.05}_{-0.03}$	$0.2335^{+0.0005}_{-0.0006}$	$0.50^{+0.02}_{-0.02}$	$0.4203^{+0.0001}_{-0.0001}$
ParT	$0.45^{+0.02}_{-0.01}$	$0.3716^{+0.0010}_{-0.0009}$	$0.38^{+0.04}_{-0.04}$	$0.2222^{+0.0007}_{-0.0007}$	—	—
LLoCa-Tr.	$0.59^{+0.06}_{-0.04}$	$0.3754^{+0.0013}_{-0.0019}$	$0.52^{+0.08}_{-0.05}$	$0.2251^{+0.0003}_{-0.0004}$	$0.59^{+0.02}_{-0.03}$	$0.4208^{+0.0001}_{-0.0002}$
L-GATr	$0.59^{+0.03}_{-0.06}$	$0.3710^{+0.0004}_{-0.0009}$	$0.43^{+0.04}_{-0.04}$	$0.2210^{+0.0009}_{-0.0009}$	$0.47^{+0.08}_{-0.07}$	$0.4207^{+0.0007}_{-0.0008}$
L-GATr-slim	$0.55^{+0.02}_{-0.02}$	$0.3706^{+0.0004}_{-0.0008}$	$0.42^{+0.03}_{-0.04}$	$0.2196^{+0.0004}_{-0.0004}$	$0.46^{+0.01}_{-0.02}$	$0.4198^{+0.0002}_{-0.0002}$

Foundation Model

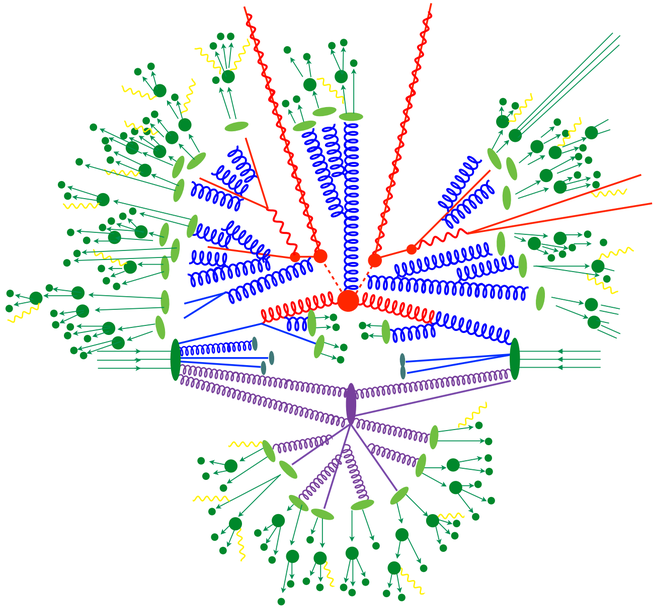
"A foundation model is any model that is trained on broad data (generally using self-supervision at scale) that can be adapted (e.g., fine-tuned) to a wide range of downstream tasks."



[arXiv: 2108.07258]

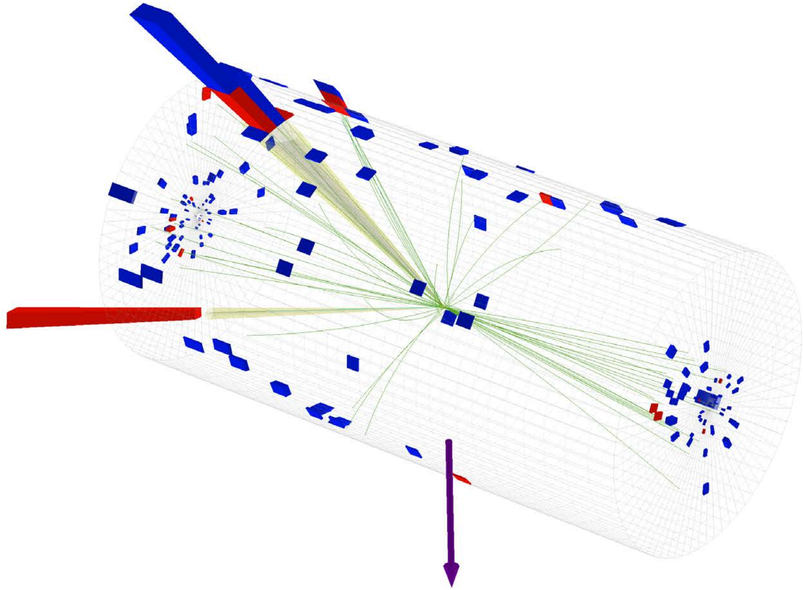
Towards a Foundation Model

- Foundation models need to be (pre-)trained on large-scale dataset



Train on MC simulations

Labels are available—
Supervised Learning

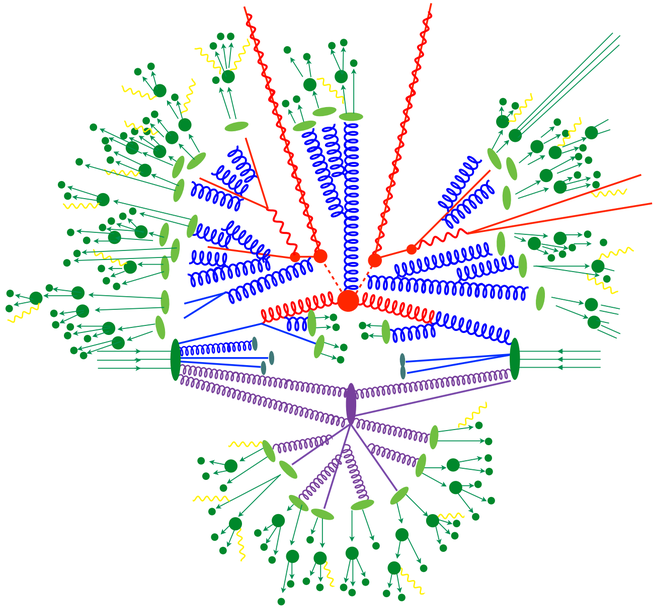


Train on real data

Labels are not available—
Self-Supervised Learning (SSL)

Towards a Foundation Model

- Foundation models need to be (pre-)trained on large-scale dataset



Train on MC simulations

Labels are available—
Supervised Learning

See e.g., **"OmniLearn"** *[V. Mikuni and B. Nachman, arXiv: 2404.16091, arXiv: 2502.14652],*

"Sophon" / "GloParT" *[C. Li, et al., arXiv: 2405.12972],*

"UParT" *[CMS, CMS-DP-2024-066, CMS-DP-2025-081],*

"OmniLearned" *[W. Bhimji, C. Harris, V. Mikuni and B. Nachman, arXiv: 2510.24066],*

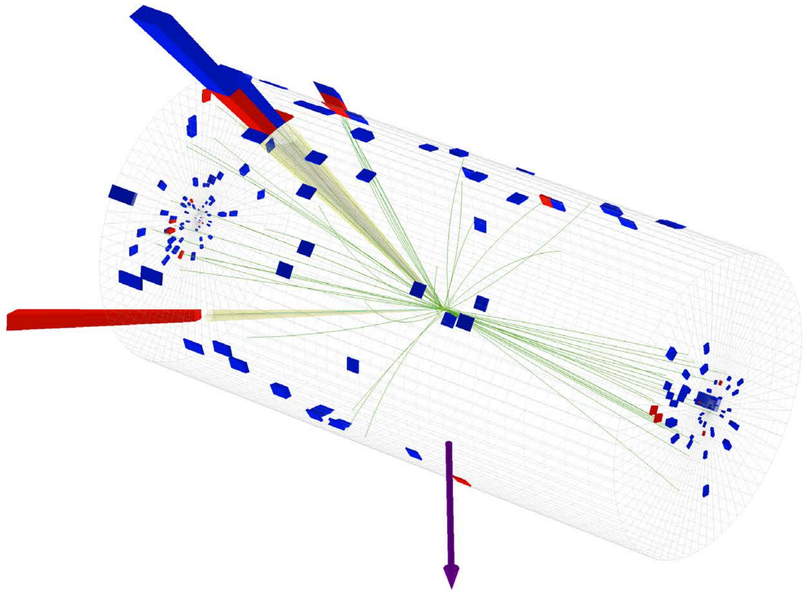
"EveNet" *[T. H. Hsu, et al., arXiv: 2601.17126].*

Towards a Foundation Model

- Foundation models need to be (pre-)trained on large-scale dataset

Train on MC simulations

Labels are available—
Supervised Learning

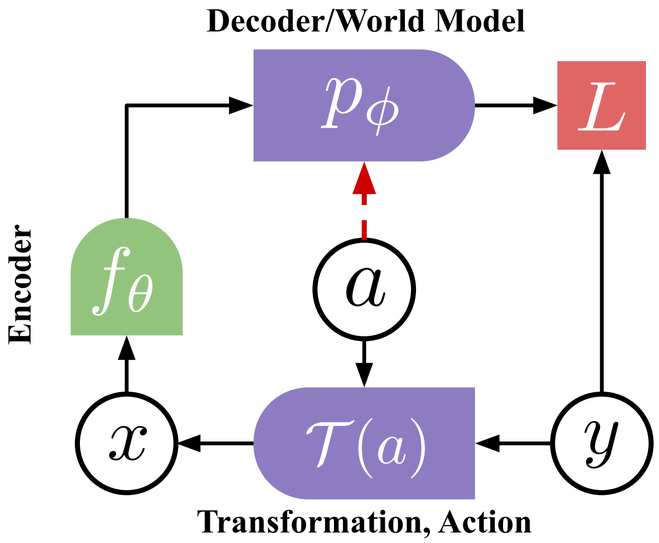


Train on real data

Labels are not available—
Self-Supervised Learning (SSL)

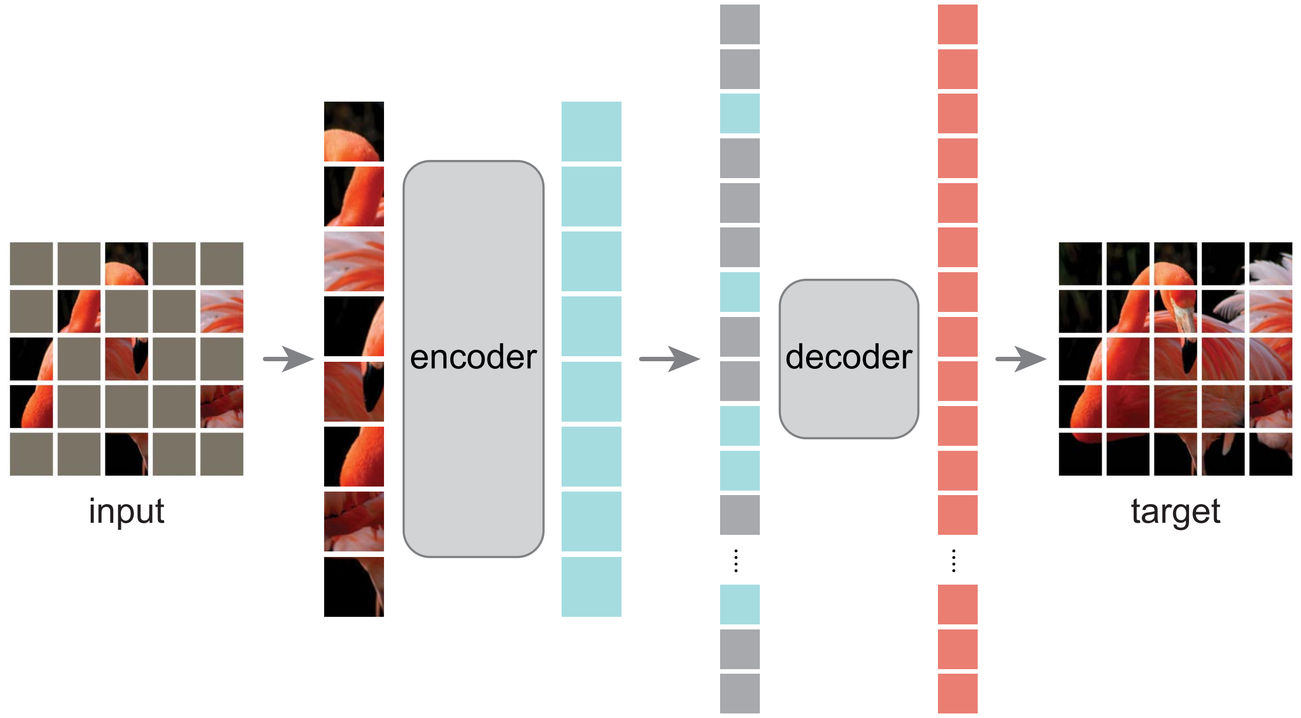
SELF-SUPERVISION: MASKED MODELING

Generative Architecture



*Learns to invert a transformation in the **input** space.*
e.g., Masked Autoencoder (MAE)

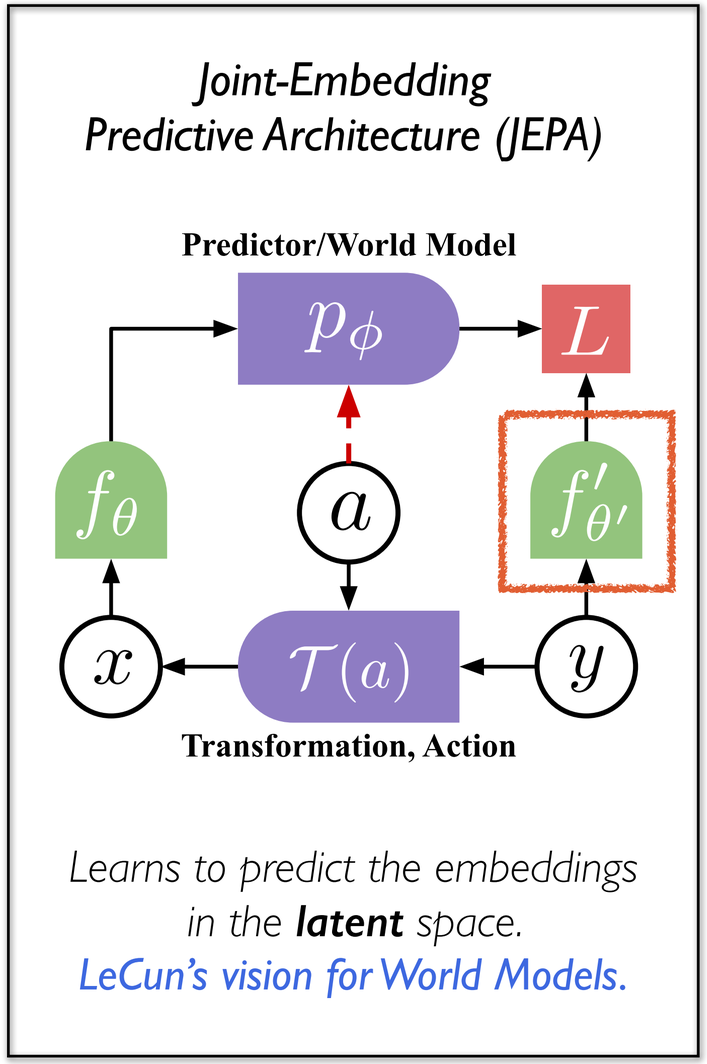
Masked Autoencoder [arXiv: 2111.06377]



… masks random patches of the input image and reconstructs the missing pixels

See e.g. "MPM" [T. Golling, L. Heinrich, M. Kagan, S. Klein, M. Leigh, M. Osadchy and J.A. Raine, arXiv: 2401.13537], "MPMv2" [M. Leigh, S. Klein, F. Charton, T. Golling, L. Heinrich, M. Kagan, I. Ochoa and M. Osadchy, arXiv: 2409.12589]

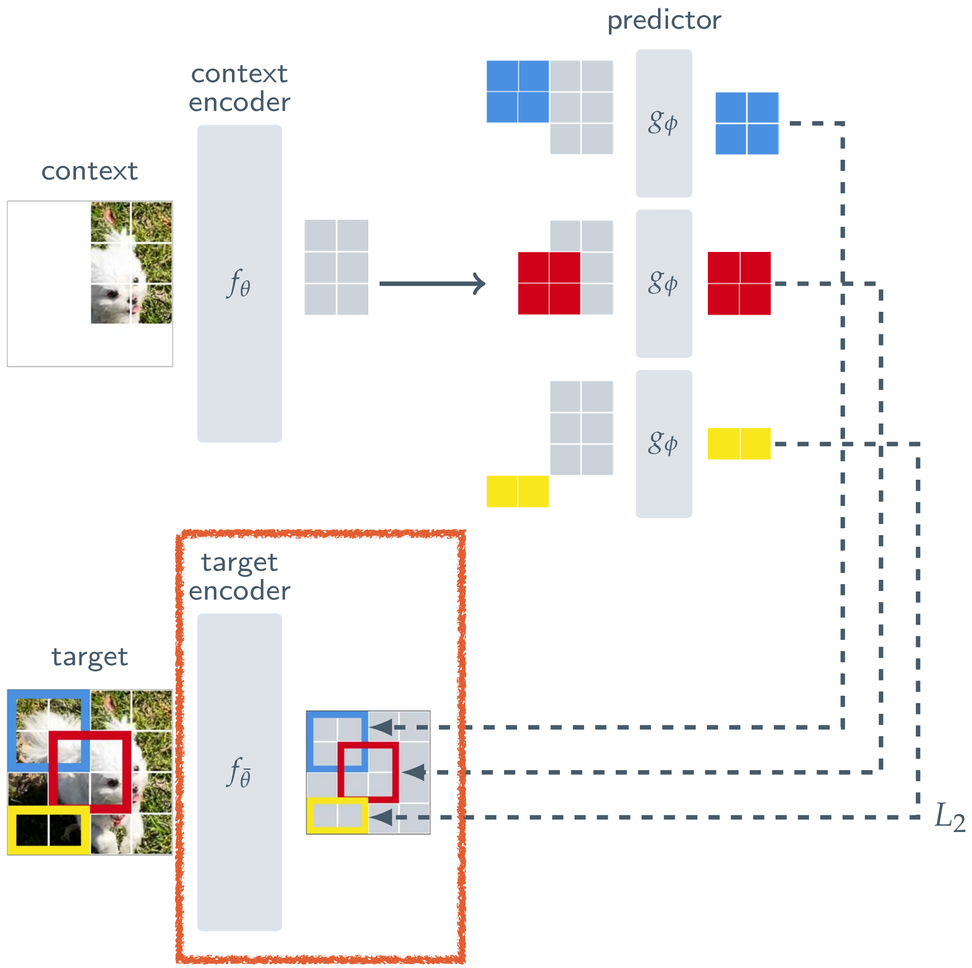
SELF-SUPERVISION: JEPA



*Learns to predict the embeddings in the **latent** space.*
LeCun's vision for World Models.

arXiv: 2403.00504

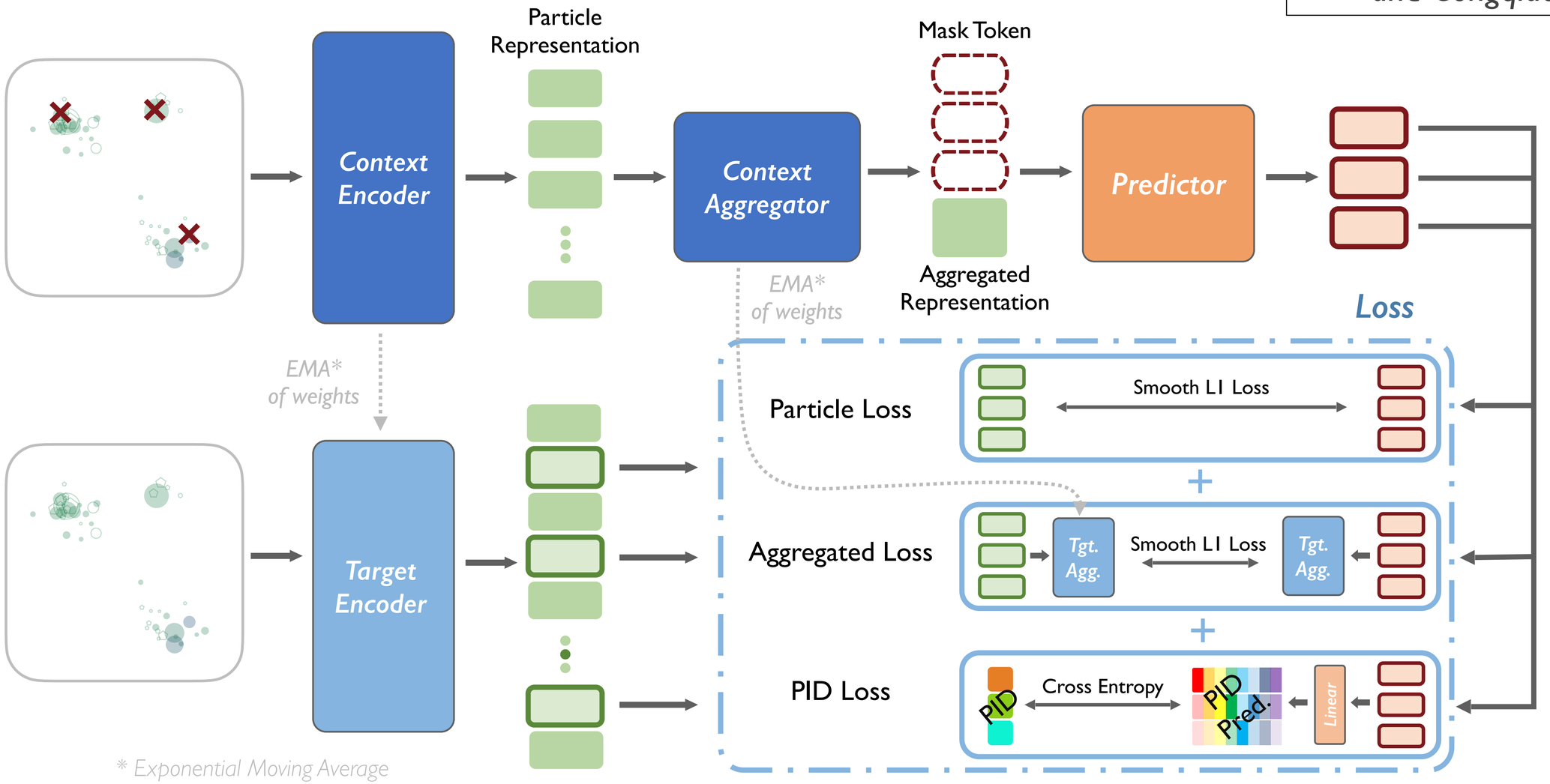
I-JEPA [arXiv: 2301.08243]



*… predicts the embeddings of masked image patches **in a (learned) latent space**.*

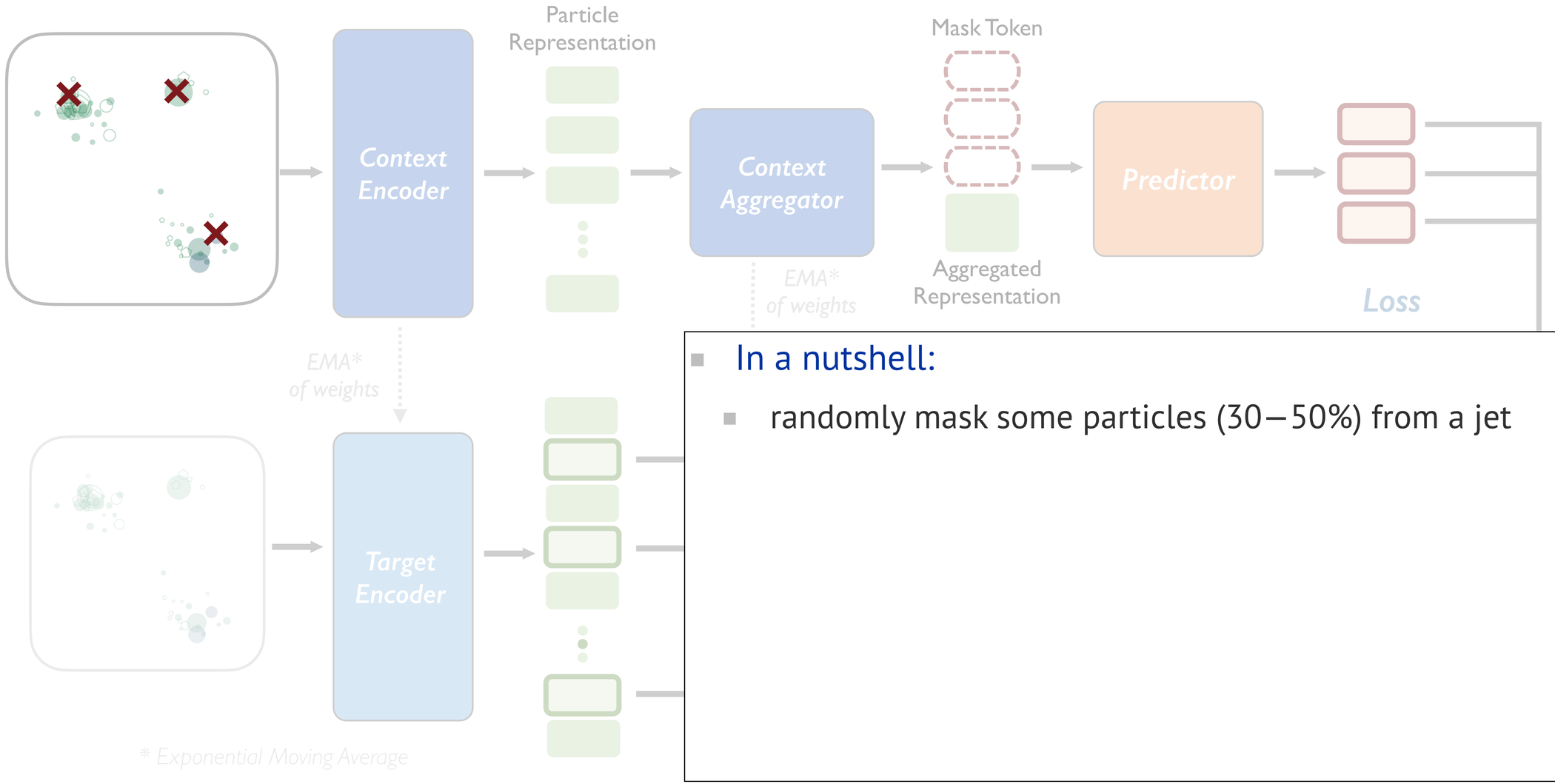
P-JEPA

Joint work with Qibin Liu, Shudong Wang and Congqiao Li



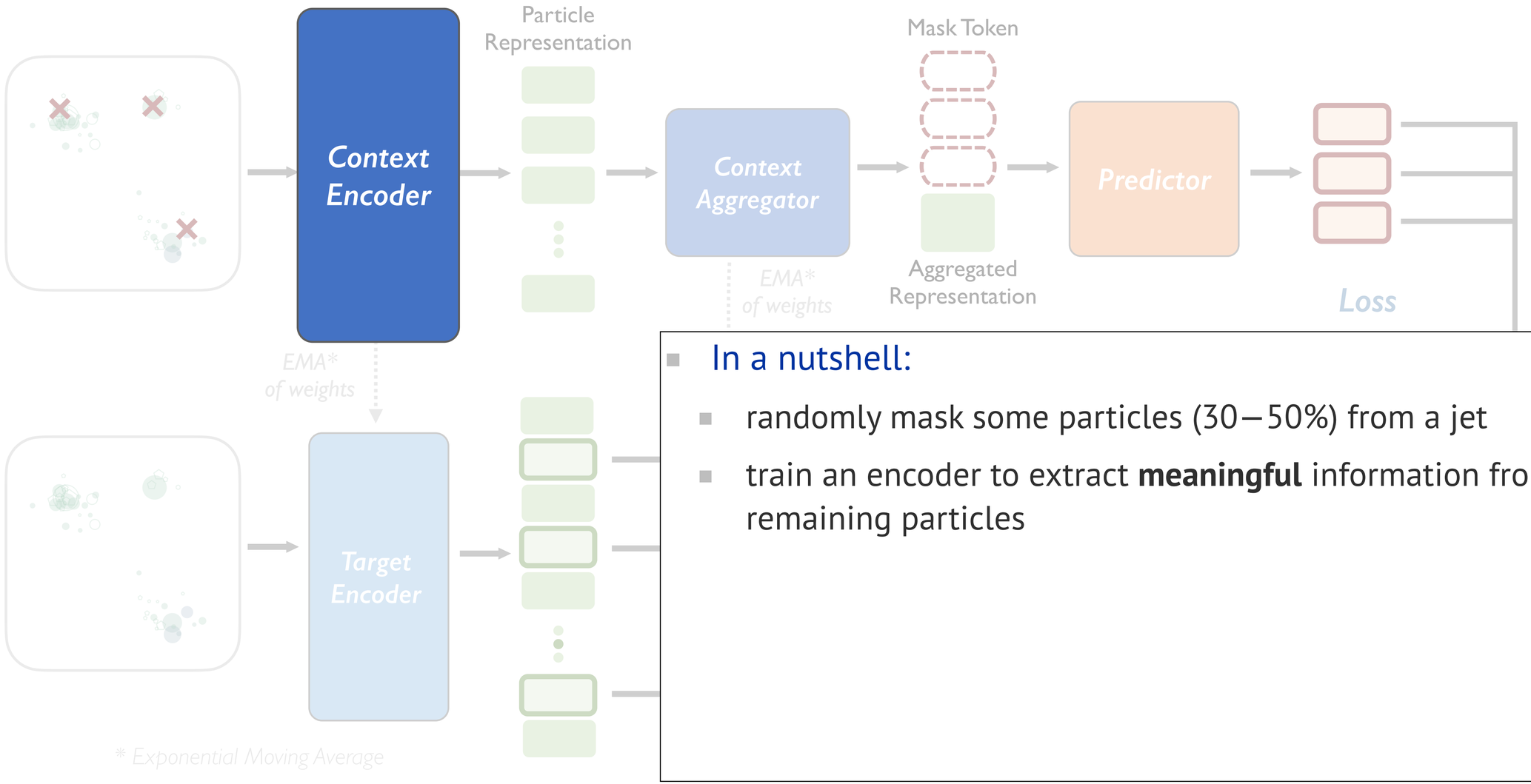
See also: "J-JEPA" [S. Katel, H. Li, Z. Zhao, F. Mokhtar, J. Duarte and R. Kansal, arXiv: 2412.05333], "HEP-JEPA" [J. Bardhan, R. Agrawal, A. Tilak, C. Neeraj and S. Mitra, arXiv: 2502.03933]

P-JEPA



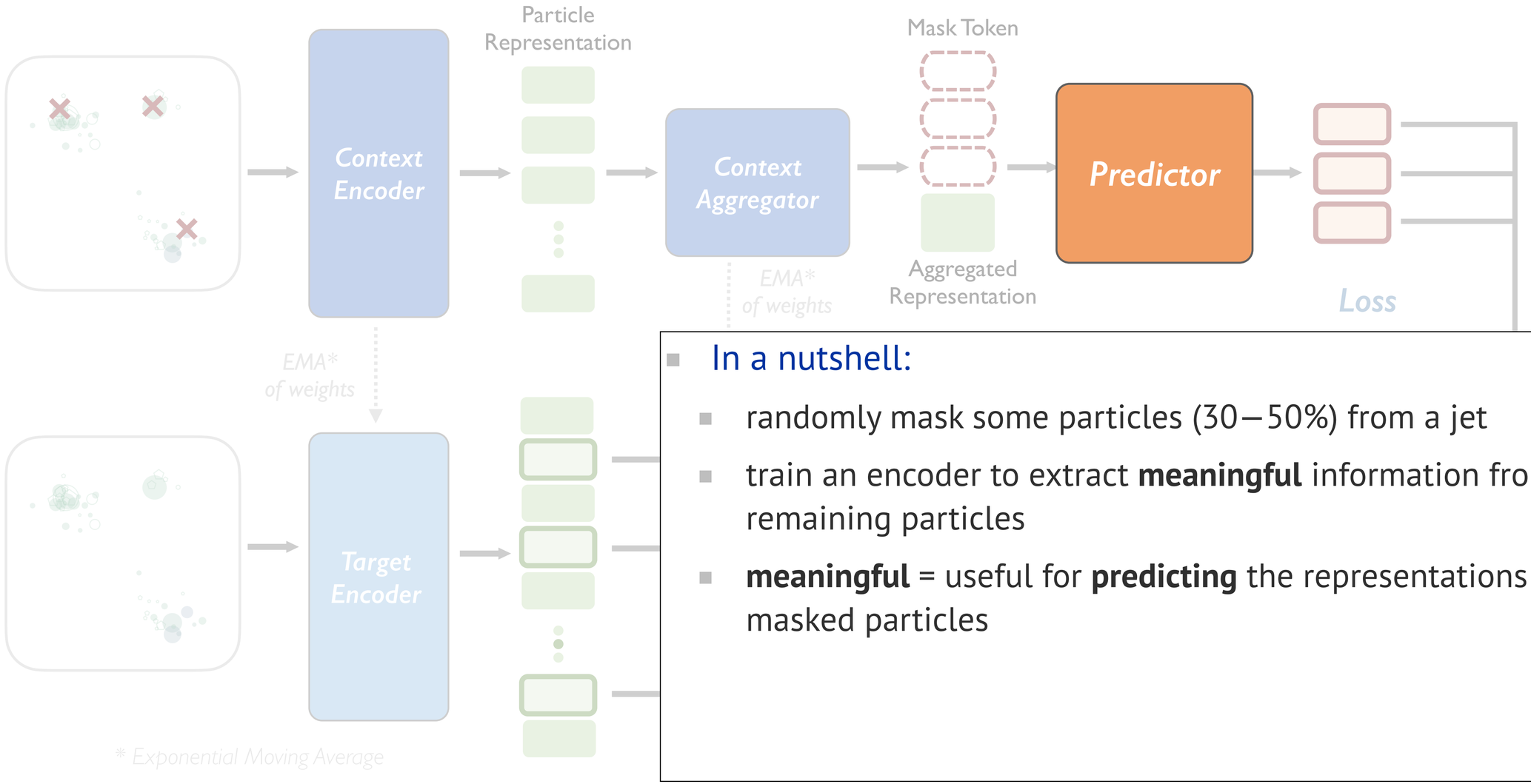
- In a nutshell:
  - randomly mask some particles (30–50%) from a jet

P-JEPA



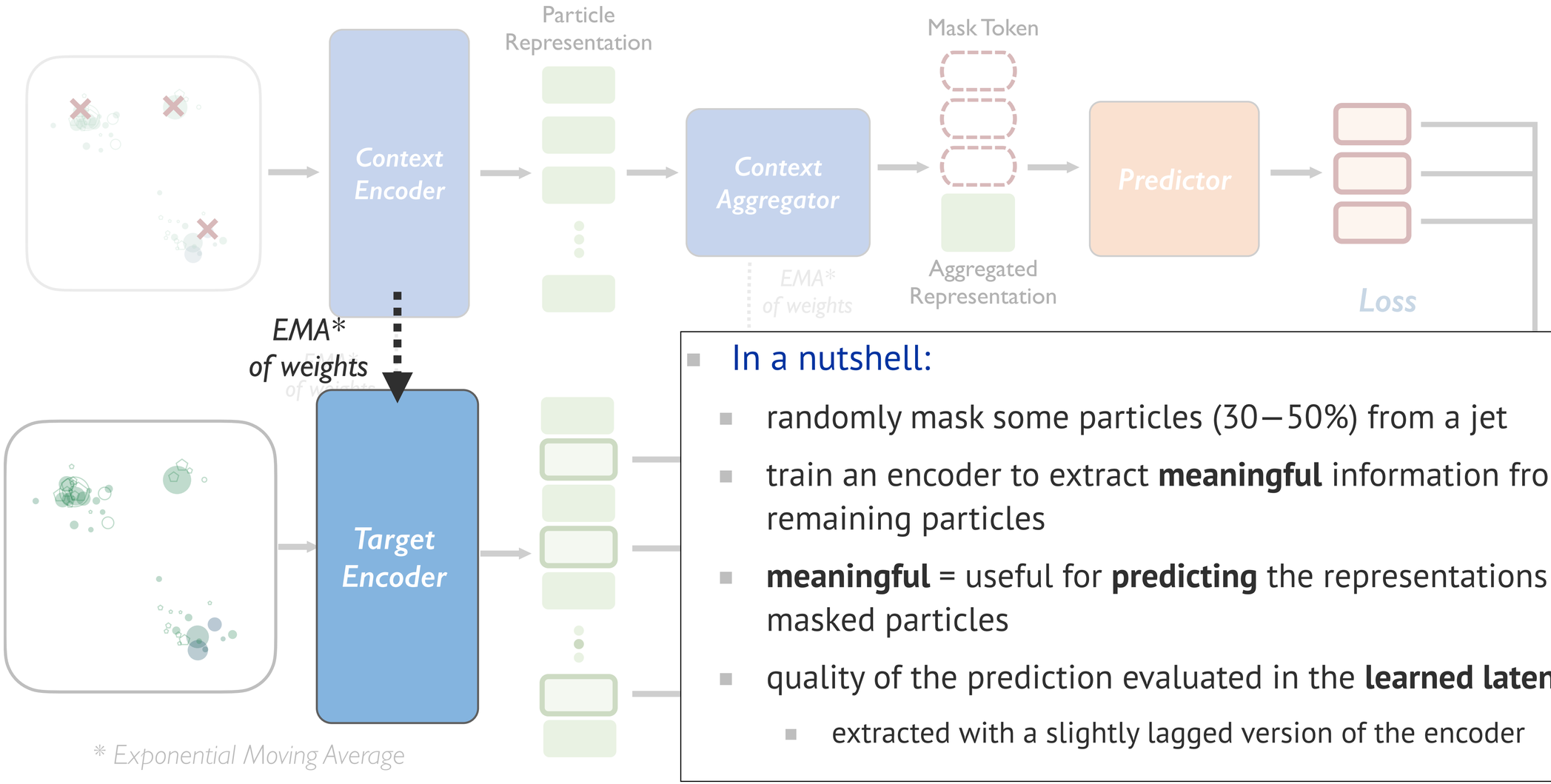
- In a nutshell:
  - randomly mask some particles (30–50%) from a jet
  - train an encoder to extract **meaningful** information from the remaining particles

P-JEPA



- In a nutshell:
  - randomly mask some particles (30—50%) from a jet
  - train an encoder to extract **meaningful** information from the remaining particles
  - **meaningful** = useful for **predicting** the representations of the masked particles

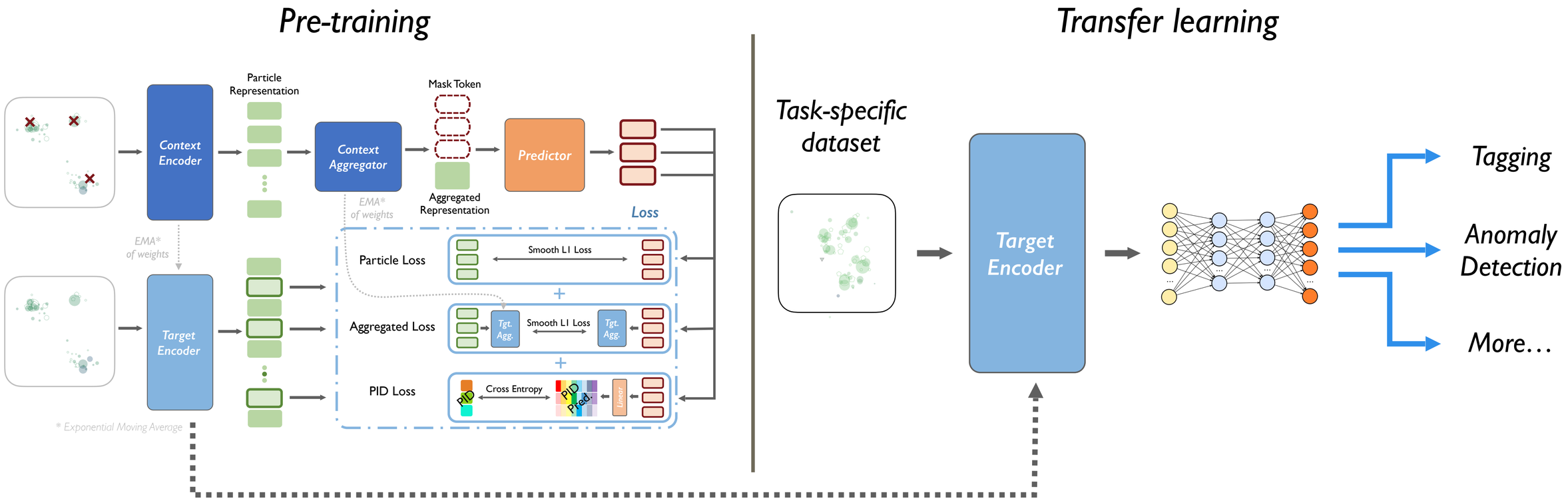
P-JEPA



- In a nutshell:
  - randomly mask some particles (30—50%) from a jet
  - train an encoder to extract **meaningful** information from the remaining particles
  - **meaningful** = useful for **predicting** the representations of the masked particles
  - quality of the prediction evaluated in the **learned latent space**
    - extracted with a slightly lagged version of the encoder

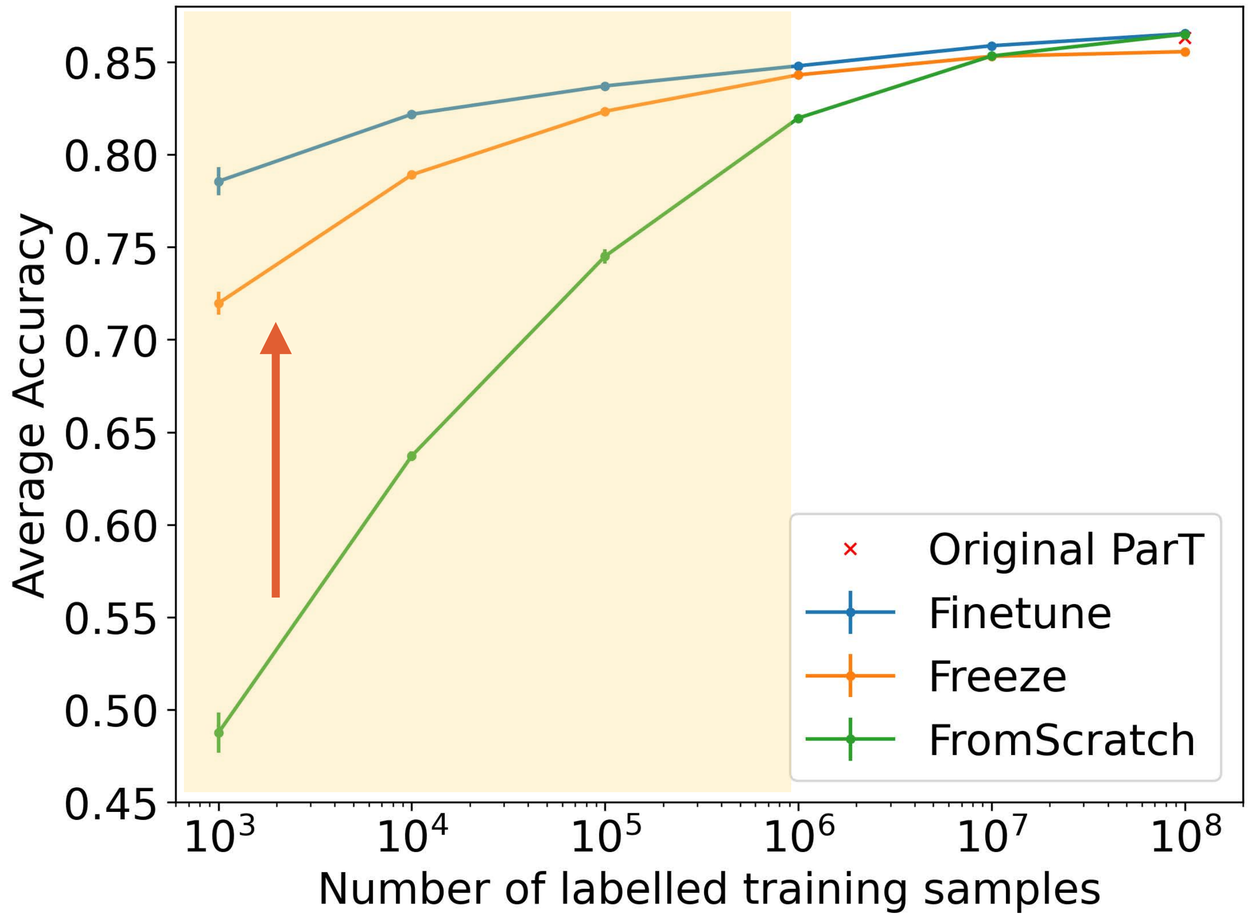
PRE-TRAINING AND TRANSFER LEARNING

- The pre-training of the P-JEPA model can be performed on large-scale real data
  - we demonstrate this by pre-training P-JEPA on the JetClass dataset (100M jets) **without using any truth labels**
- Once pre-trained, the **encoder** can be viewed as a **foundation model**
  - transfer learning to specific downstream tasks



TRANSFER LEARNING: JET TAGGING

- Benchmark: 10-class jet classification on JetClass



FineTune:

Encoder allowed to be slightly updated when trained with labelled jets for tagging

Freeze:

Encoder fixed when trained with labelled jets for tagging

FromScratch:

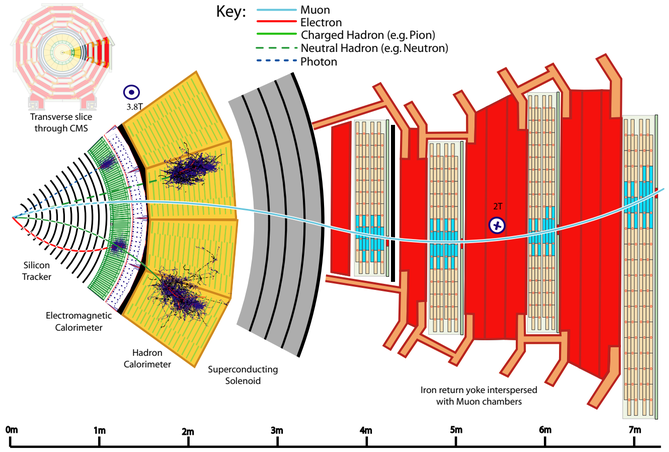
Same network architecture, but trained with labelled jets starting from randomly initialized weights

Pre-training + transfer learning shows a significant performance boost when labelled samples are limited.

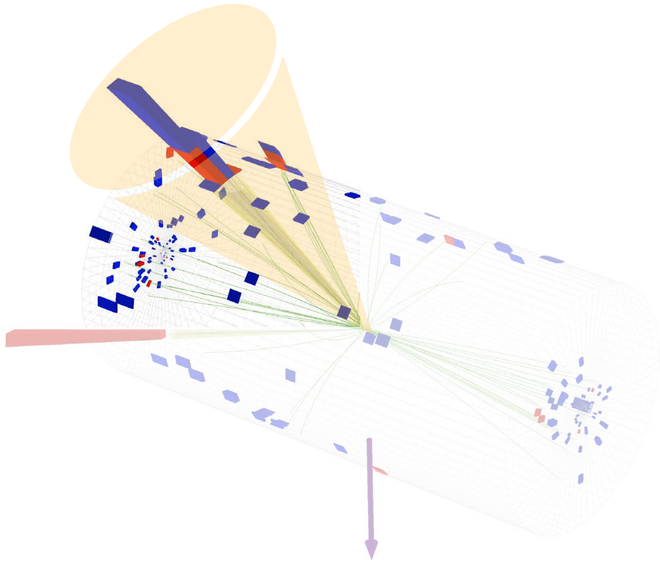
Outlook: Scaling Up

- **At colliders:** event reconstruction usually follows a hierarchical way
  - due to heterogeneous sub-detectors and high particle multiplicity
- **Going beyond: a foundation model that aligns data representations at various levels and cover all the stages?**
  - requires a multi-modal approach and efficient architectures

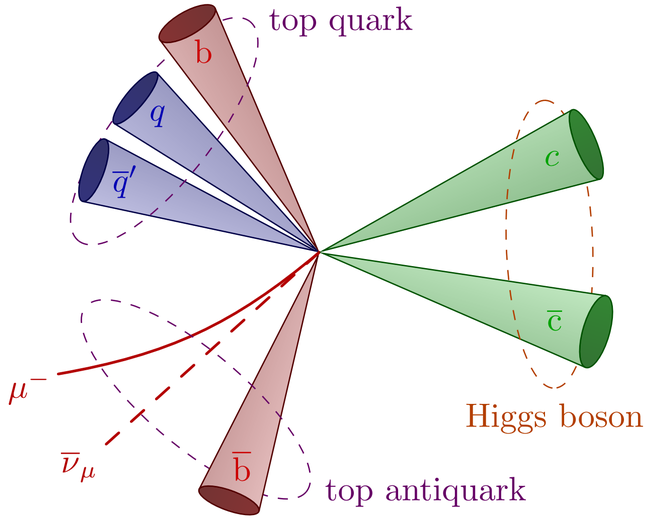
Detector/Particle-flow reconstruction



Jet/Object reconstruction



Event classification/interpretation

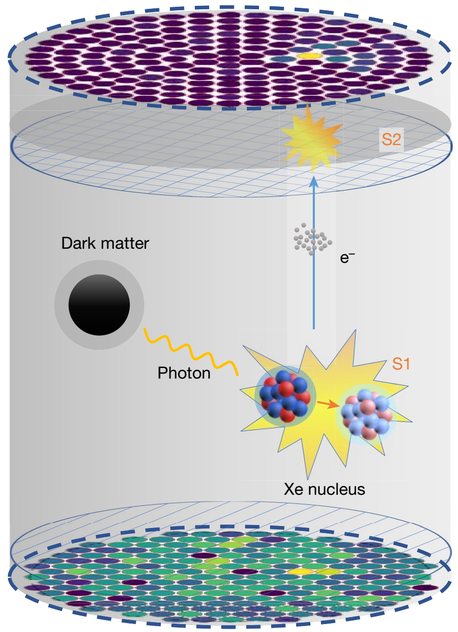


Raw detector hits => Tracks, Showers, Particles => Jets, leptons, taus => Signal vs background processes

Outlook: Scaling Out

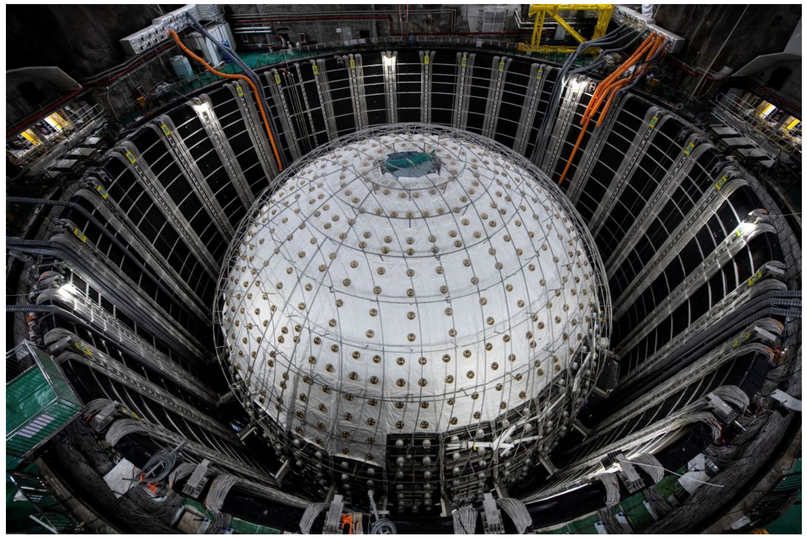
- **Dark matter/neutrino/cosmic ray experiments**
  - typically homogeneous detector — end-to-end ML reconstruction more promising
- **Going beyond: a foundation model that generalizes across experiments and covers all energy scales?**

PandaX

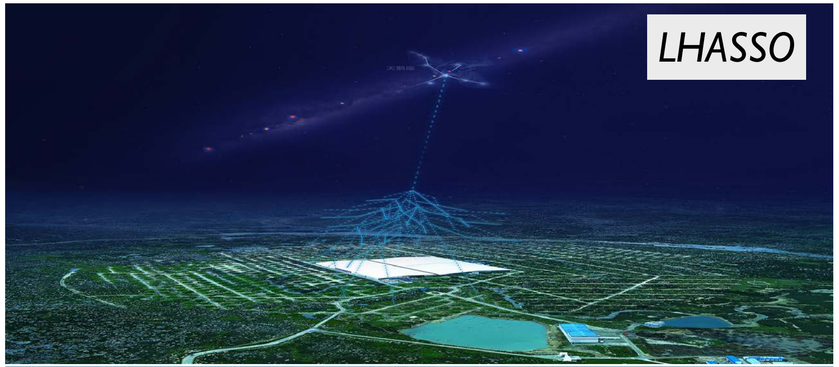


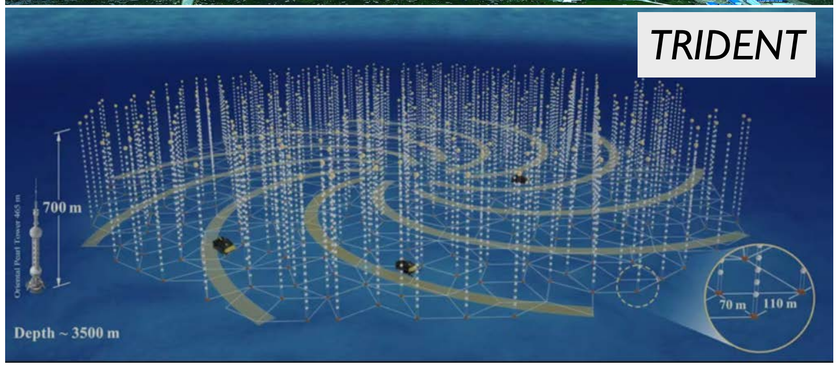
Dark Matter/Neutrino (keV~MeV)

JUNO



Neutrino (MeV~GeV)



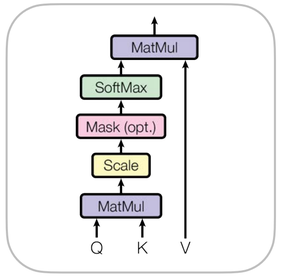


Cosmic Rays/Neutrinos (TeV~EeV)

SUMMARY

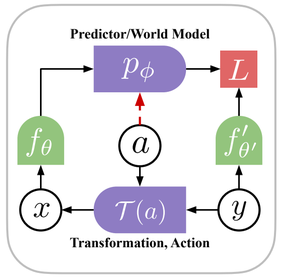
Evolution of Jet Representations

… leads to significant performance gains.



Attention Isn't All You Need

… more physics = fast learning & better generalization.



Towards a Foundation Model

… efficient and scalable representation learning is the key!