Does *Lorentz-symmetric* design boost network performance in jet physics?

Congqiao Li *(Peking University)*

based on arXiv:2208.07814

in collaboration with Huilin Qu[2], Sitian Qian[1], Qi Meng[3], Shiqi Gong[3,4], Jue Zhang[3], Tie-Yan Liu[3], Qiang Li[1]

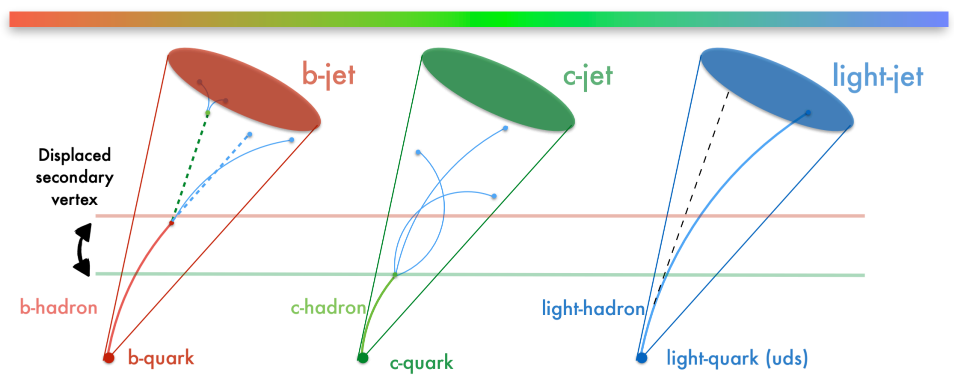
[1]PKU [2]CERN [3]MSRA [4]AMSS, CAS

ML4Jets 2022 · Rutgers University
2 November, 2022

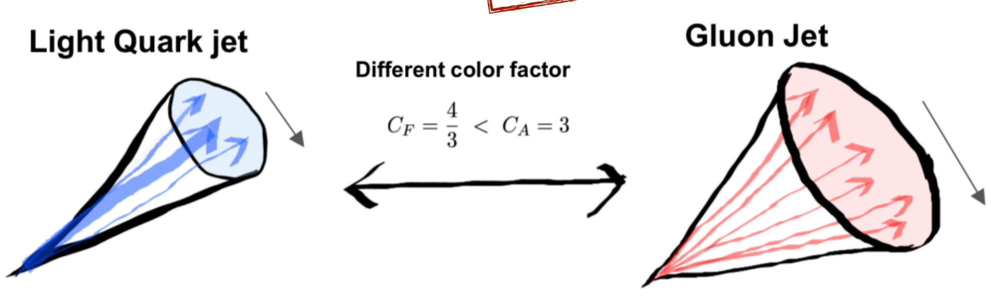
Jet tagging × deep learning

- ➔ Jet tagging in the deep learning era
  - ❖ has brought ***a new performance level*** for jet tagging 😆
  - ❖ has had ***a profound impact*** for many ***physics analyses***! 🎖️
  - ❖ efforts for ***further improvements still ongoing*** 🛣️

FLAVOUR TAGGING

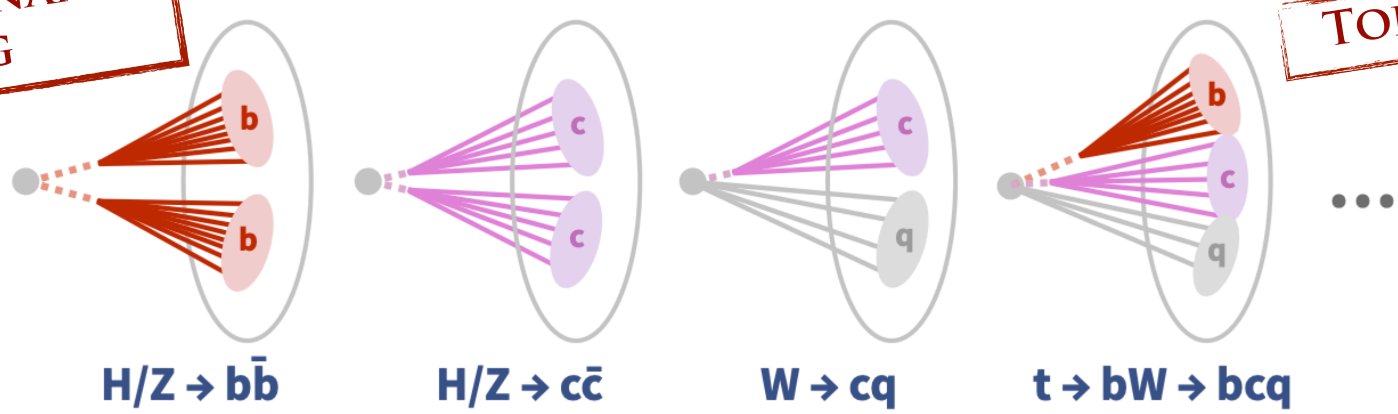


QUARK/GLUON DISCRIMINATION



FLAVOUR RESONANCE TAGGING

TOP TAGGING



"Post-ParticleNet" improvement

➔ Various works continuously aim for improving the network performance, after ParticleNet marked a success

- here allow me to summarize **some* tips & tricks discovered in the recent 1-2 years

Advanced deep learning experiences!

Physics-inspired designs!

"Post-ParticleNet" improvement

→ Various works continuously aim for improving the network performance, after ParticleNet marked a success

❖ here allow me to summarize **some** tips & tricks discovered in the recent 1-2 years

✦ "Graph" behaves better

i.e., Graph NN/Transformer architecture builds edges (interactions) between a pair of particles. Generally, in terms of performance: fully-connected edges > edges from k-NN > no edge

✦ Attentive pooling over average/max pooling

i.e., on aggregating features among all particles, using average/max pooling losses more information; assigning learnable weights to particles (or other similar approaches) usually works better

✦ "Multi" over "one"

i.e., delivering multiple trainings on a given NN structure performs generally better than doing it once. Examples: multi-head over single-head; multi-scale k-NN for edge construction; training an ensemble of networks vs. training once…

✦ Pairwise features help

i.e., constructing pairwise features between particles is a solution to improve network performance

✦ Physics-informed edges

i.e., build a certain form of graph based on physics-informed information. Example: define tree structures based on particle clustering information.

✦ Injecting symmetries

i.e., allow the network to obey a certain type of symmetry by the dedicated design of symmetry-preserving layers/architecture

Reference:
ABCNet: V. Mikuni et al. EPJC 2020; 135(6): 463
LGN: A. Bogatskiy et al. arXiv: 2006.04780, ICML 2020
ParticleNeXt: H. Qu. Talk@ML4Jets2021
LundNet: F. Dreyer et al. JHEP 03 (2021) 052
PCN: C. Shimmin. arXiv:2107.02908
LorentzNet: S. Gong et al. JHEP 07 (2022) 030
ParT: H. Qu et al. arXiv:2202.03772, ICML 2022
CPT : S. Qiu et al. arXiv:2203.05687
HMPNet : F. Ma et al. arXiv:2210.13869

“Post-ParticleNet” improvement

➔ Various works continuously aim for improving the network performance, after ParticleNet marked a success

❖ here allow me to summarize *some* tips & tricks discovered in the recent 1-2 years

- “Injecting symmetries” into a network is a popular and promising field
- Dedicated networks have been proposed such as to be invariant/equivariant to certain symmetries, e.g.:
  - boost on z-axis, rotation on x-y plane
  - rotation on the η–ϕ plain (similarly, around the jet axis)
  - boost along the “jet axis”
  - fully Lorentz symmetry
  - …
- ***Can we do it without a special network design? - Yes!***

✦ Physics-informed edges

i.e., build a certain form of graph based on physics-informed information. Example: define tree structures based on particle clustering information.

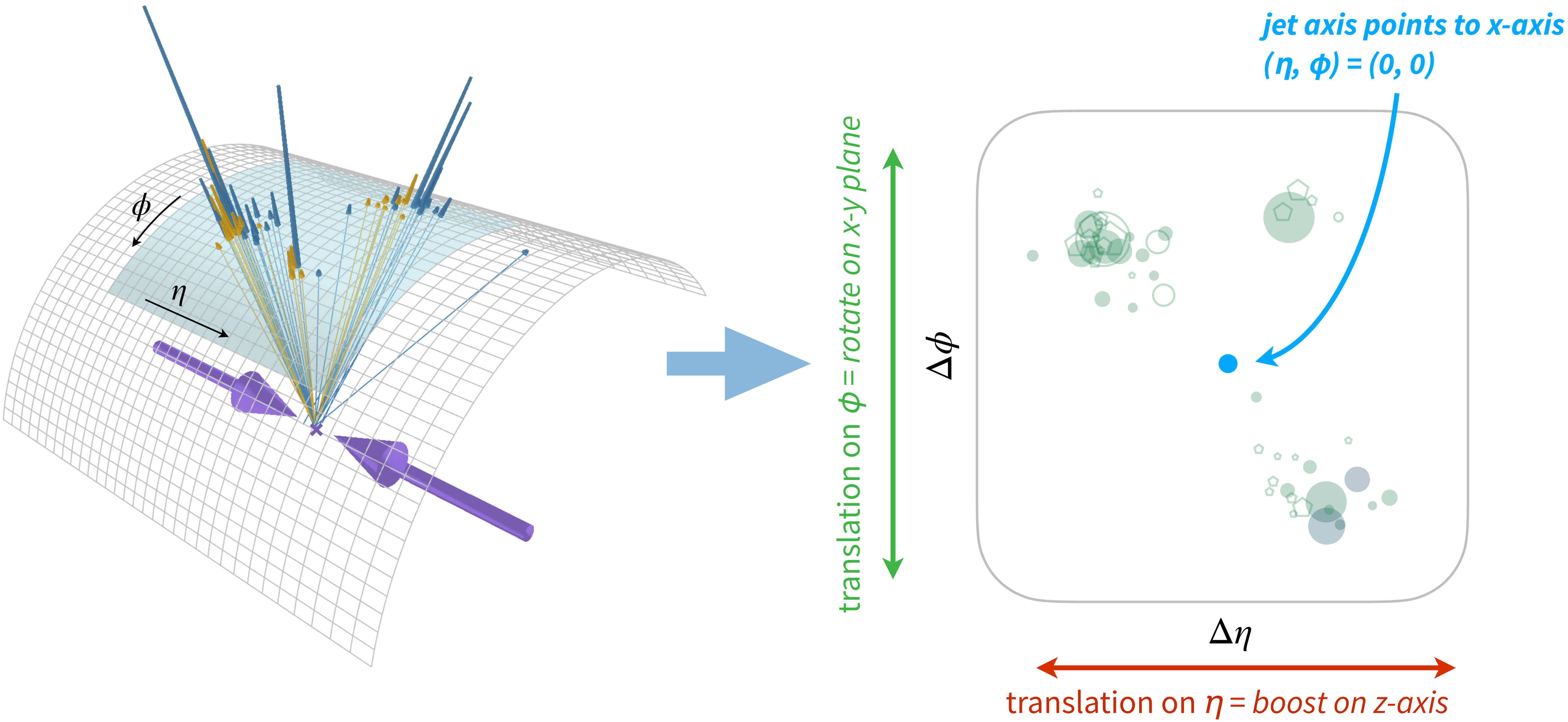
✦ Injecting symmetries

i.e., allow the network to obey a certain type of symmetry by the dedicated design of symmetry-preserving layers/architecture

Reference:
ABCNet: V. Mikuni et al. EPJC 2020; 135(6): 463
LGN: A. Bogatskiy et al. arXiv: 2006.04780, ICML 2020
ParticleNeXt: H. Qu. Talk@ML4Jets2021
LundNet: F. Dreyer et al. JHEP 03 (2021) 052
PCN: C. Shimmin. arXiv:2107.02908
LorentzNet: S. Gong et al. JHEP 07 (2022) 030
ParT: H. Qu et al. arXiv:2202.03772, ICML 2022
CPT : S. Qiu et al. arXiv:2203.05687
HMPNet : F. Ma et al. arXiv:2210.13869

Lorentz transformations and symmetry

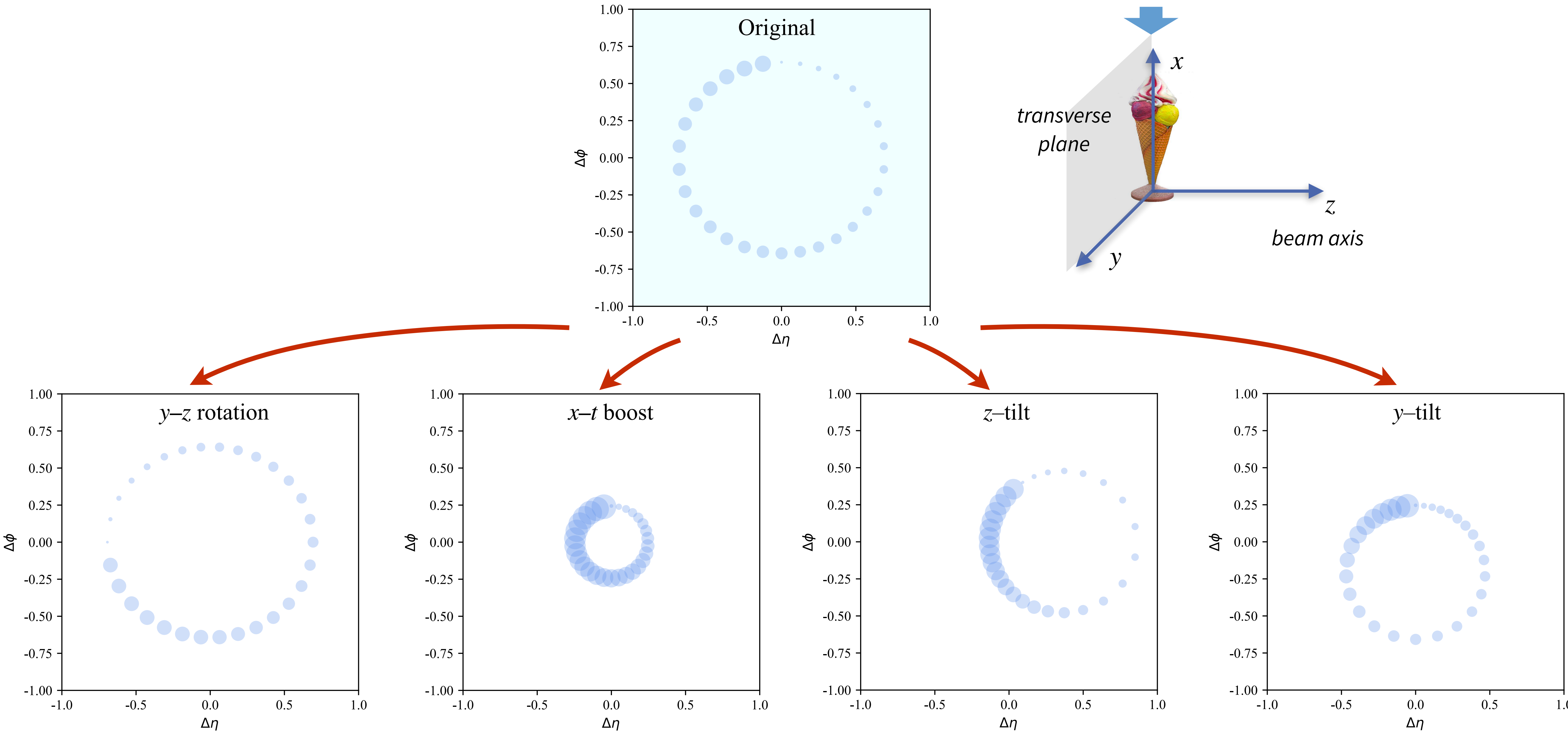
- ➔ By HEP convention, a jet is represented on Δη-Δϕ plane w.r.t. its axis
  - ❖ this pre-processing step is equivalent as:
    apply a boost on z-axis → ***then a rotation on x-y plane*** (transverse plane) → **now jet points to the *x*-axis**, *i.e.* $(\eta, \phi) = (0, 0)$



Lorentz transformations and symmetry

- ➔ By HEP convention, a jet is represented on Δη-Δϕ plane w.r.t. its axis
  - ❖ after the conventional pre-processing, we have ***four additional DoFs*** for Lorentz transformation!

A toy jet for illustration purpose
jet axis points to (η, ϕ) =(0, 0)



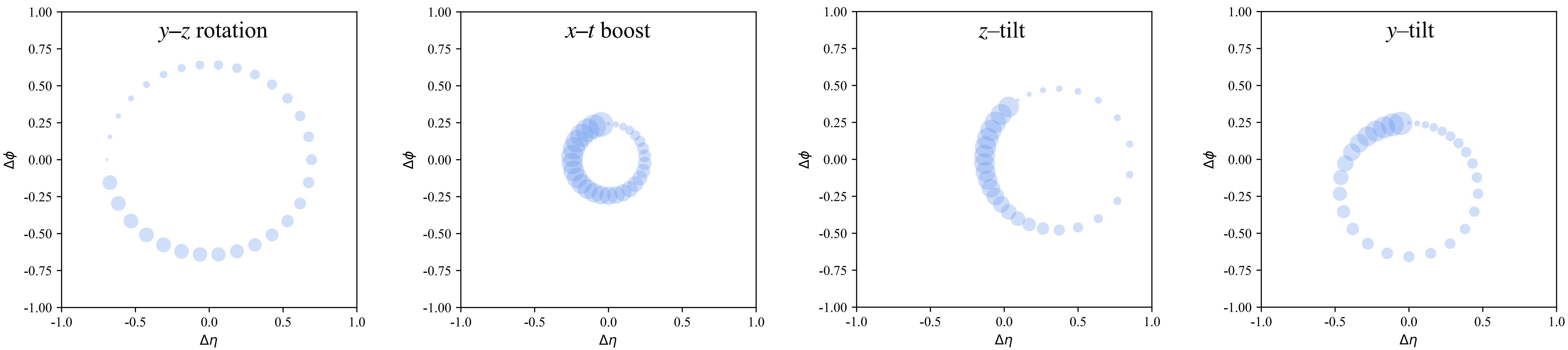
≈ η-ϕ rotation

z-boost + x-z rotation

y-boost + x-y rotation

Lorentz transformations and symmetry

- By HEP convention, a jet is represented on Δη-Δϕ plane w.r.t. its axis
  - after the conventional pre-processing, we have ***four additional DoFs*** for Lorentz transformation!
- If a network respects Lorentz symmetry…
  - it means the output score is invariant under any Lorentz transformation of the input jet
  - [solution 1] design a dedicated structure to maintain invariant/equivariant
  - ***[solution 2] only use those invariant features as input!***



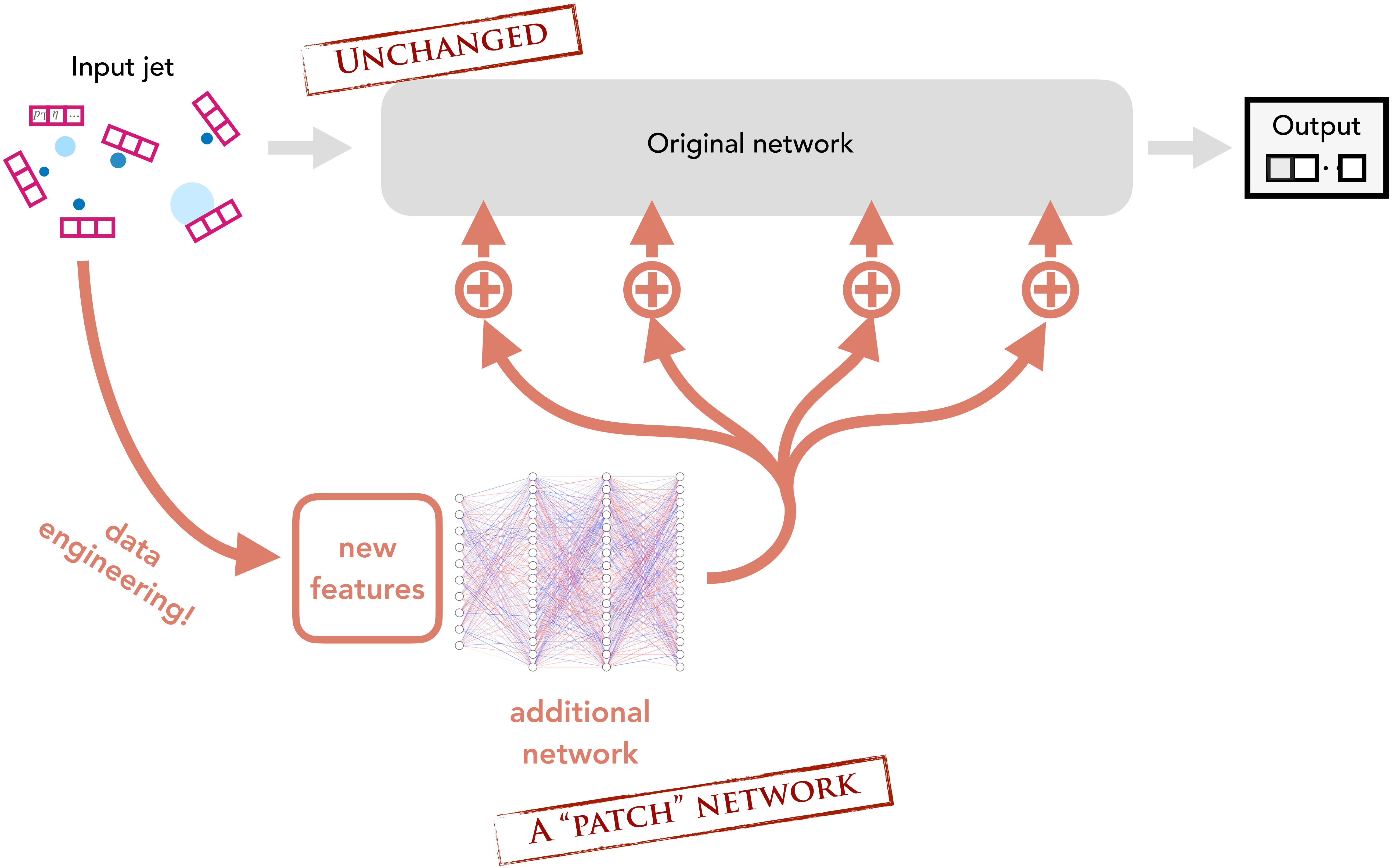
≈ η-ϕ rotation

z-boost + x-z rotation

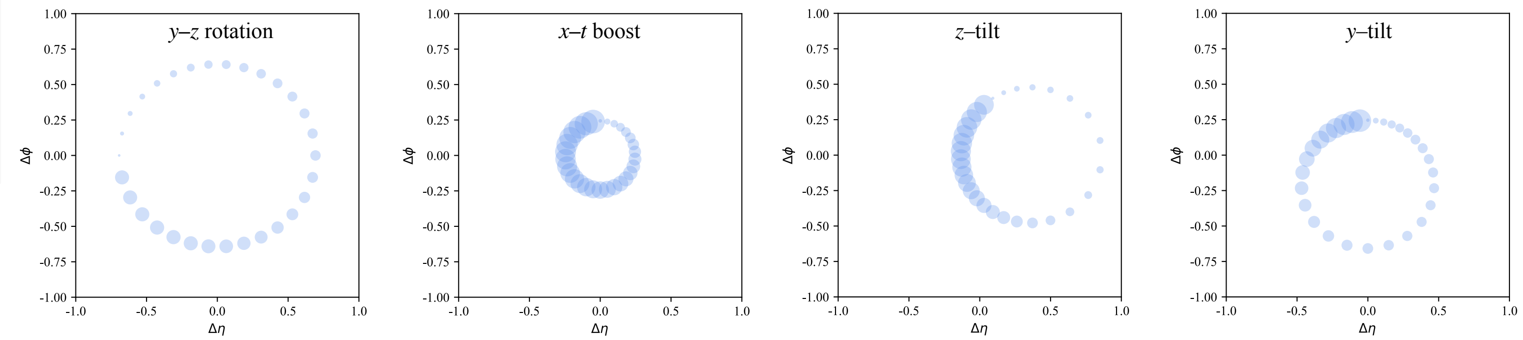
y-boost + x-y rotation

Our proposal on network architecture

Our proposal on network architecture



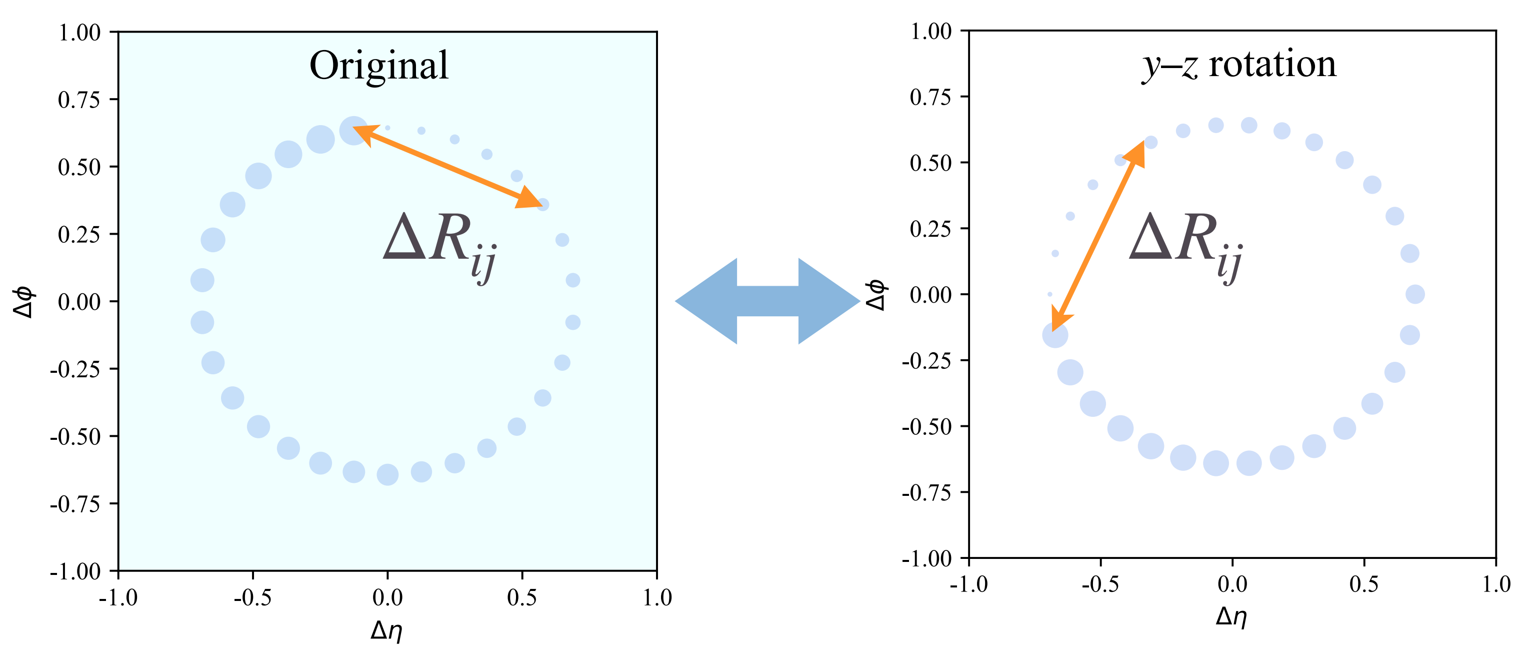
Feature design



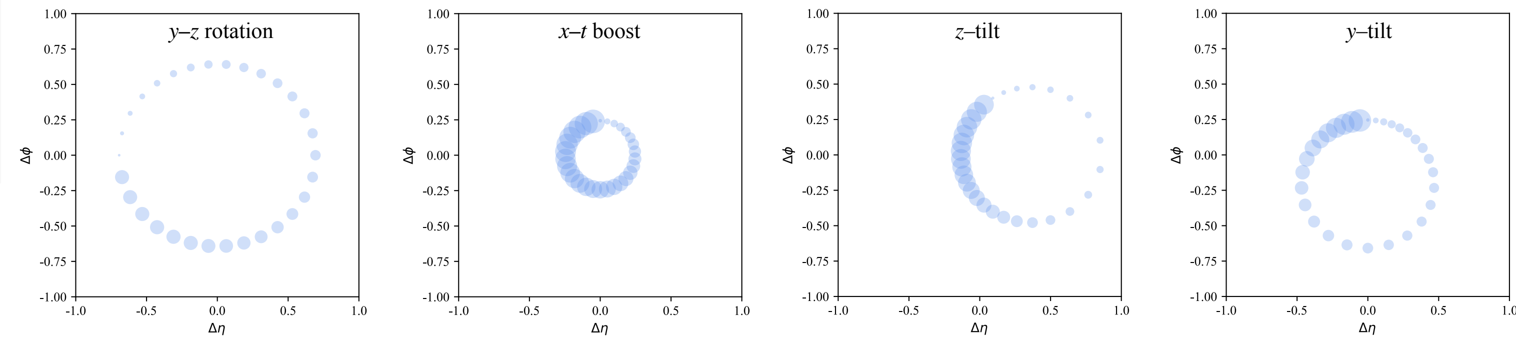
Pairwise variable	z–t boost	x–y rotation	y–z rotation ($y_{y,z} \sim o(1)$)	x–t boost ($y_{y,z} \sim o(1)$)	z-tilt ($y_{y,z} \sim o(1)$)	y-tilt ($y_{y,z} \sim o(1)$)
m_{ij}^2	✓	✓	✓	✓	✓	✓
ΔR_{ij}	✓	✓	✓			
$\Delta R_{ij}(p_{\mathrm{T},i} + p_{\mathrm{T},j})$	✓	✓	✓	✓		
E_{ij} (ablation study)		✓	✓			

➔ First, let's construct "pairwise" variables invariant under ***some or all Lorentz (sub)symmetries***

- **pairwise mass** $m_{ij}^2 = p_i^\mu p_{j,\mu}$: invariant under all transformations
- **pairwise** ΔR_{ij}: approx. invariant under y-z rotation (≈ η-ϕ rotation)



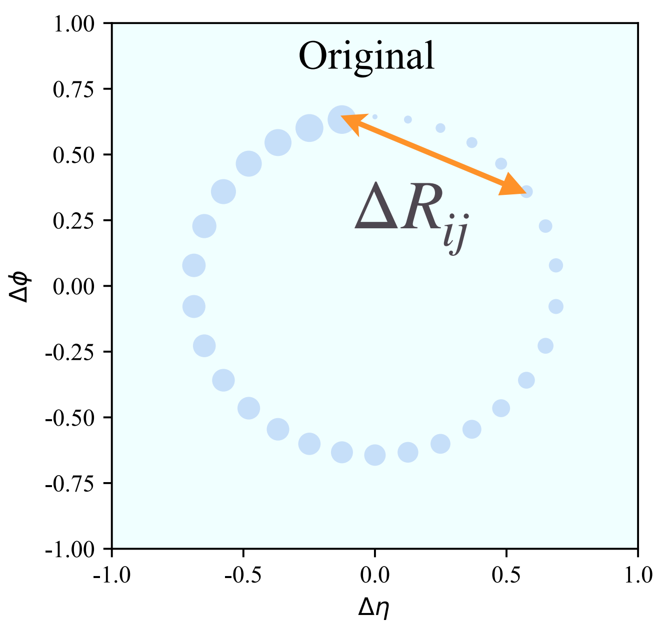
Feature design

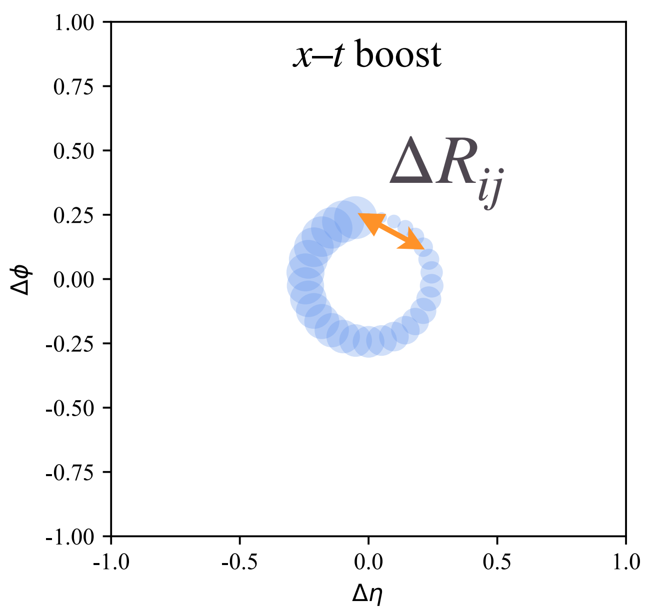


Pairwise variable	z–t boost	x–y rotation	y–z rotation ($y_{y,z} \sim o(1)$)	x–t boost ($y_{y,z} \sim o(1)$)	z-tilt ($y_{y,z} \sim o(1)$)	y-tilt ($y_{y,z} \sim o(1)$)
m_{ij}^2	✓	✓	✓	✓	✓	✓
ΔR_{ij}	✓	✓	✓			
$\Delta R_{ij}(p_{\mathrm{T},i} + p_{\mathrm{T},j})$	✓	✓	✓	✓		
E_{ij} (ablation study)		✓	✓			

➔ First, let's construct "pairwise" variables invariant under ***some or all Lorentz (sub)symmetries***

- **pairwise mass** $m_{ij}^2 = p_i^\mu p_{j,\mu}$: invariant under all transformations
- **pairwise** ΔR_{ij}: approx. invariant under y-z rotation (≈ η-ϕ rotation)
- manually construct variable $\Delta R_{ij}(p_{\mathrm{T},i} + p_{\mathrm{T},j})$: can prove that it is also approx. invariant under x-boost

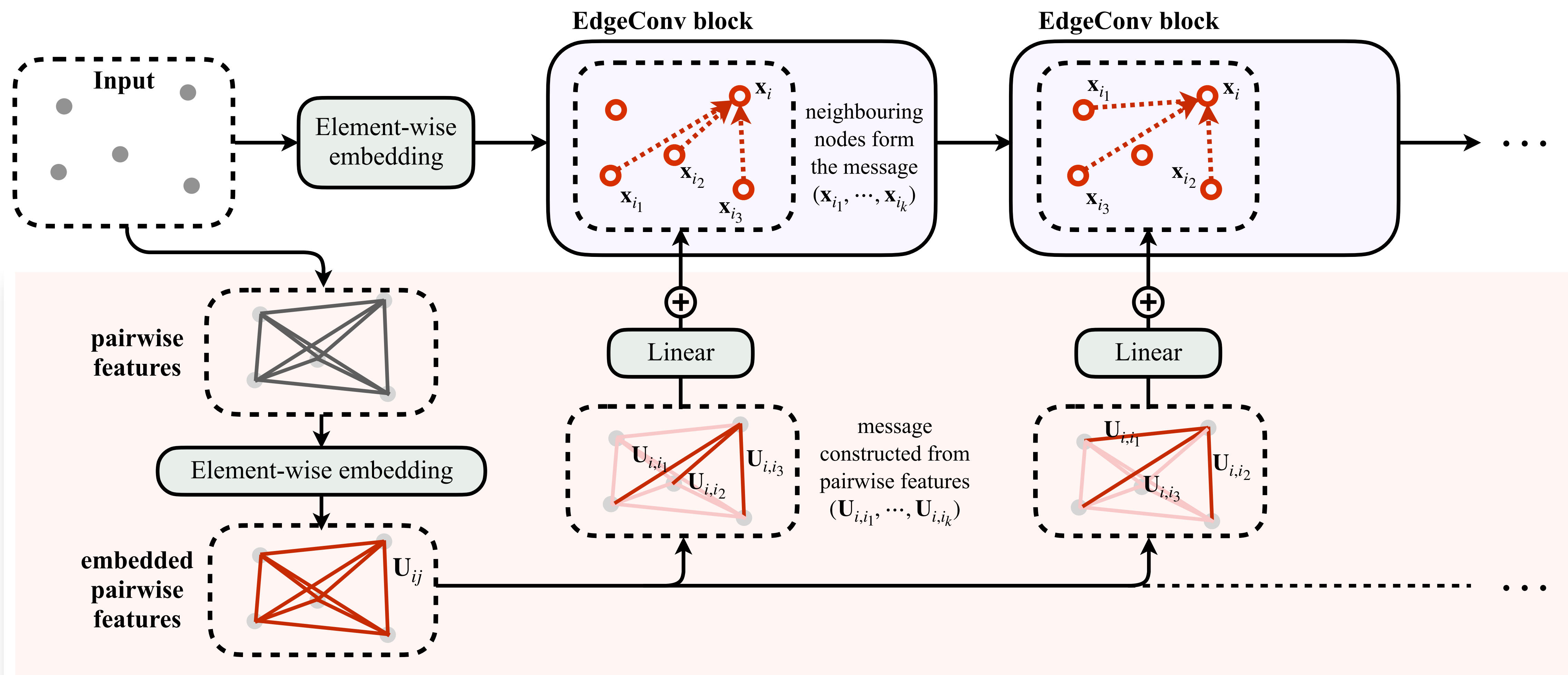




ΔR_{ij} smaller, but p_T larger

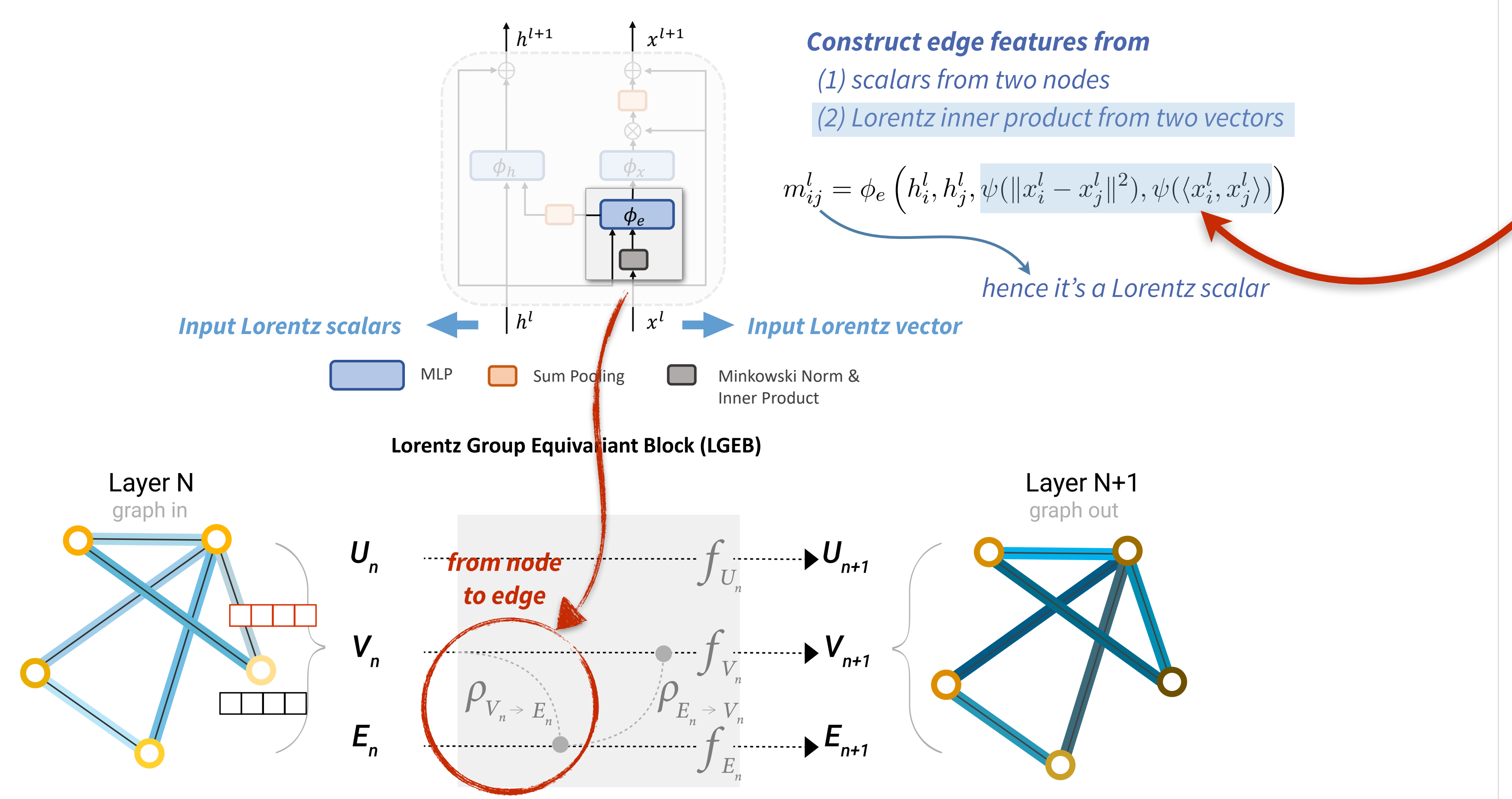
Experiments on ParticleNet and LorentzNet

- ➔ Two baseline networks studied ***ParticleNet*** & ***LorentzNet$_{base}$***:
  - ❖ how to combine pairwise features + additional patch network to the baseline network?
- ➔ ***ParticleNet***: integrate pairwise features into the network according to the intrinsic k-NN pairs



Experiments on ParticleNet and LorentzNet

- ➔ Two baseline networks studied ***ParticleNet*** & ***LorentzNet$_{base}$***:
  - ❖ how to combine pairwise features + additional patch network to the baseline network?
- ➔ ***LorentzNet$_{base}$***: LorentzNet has already included "pairwise mass": remove it to create our baseline (but complete all node features as the case of ParticleNet)



LorentzNet @ BOOST2022

figure from https://distill.pub/2021/gnn-intro/

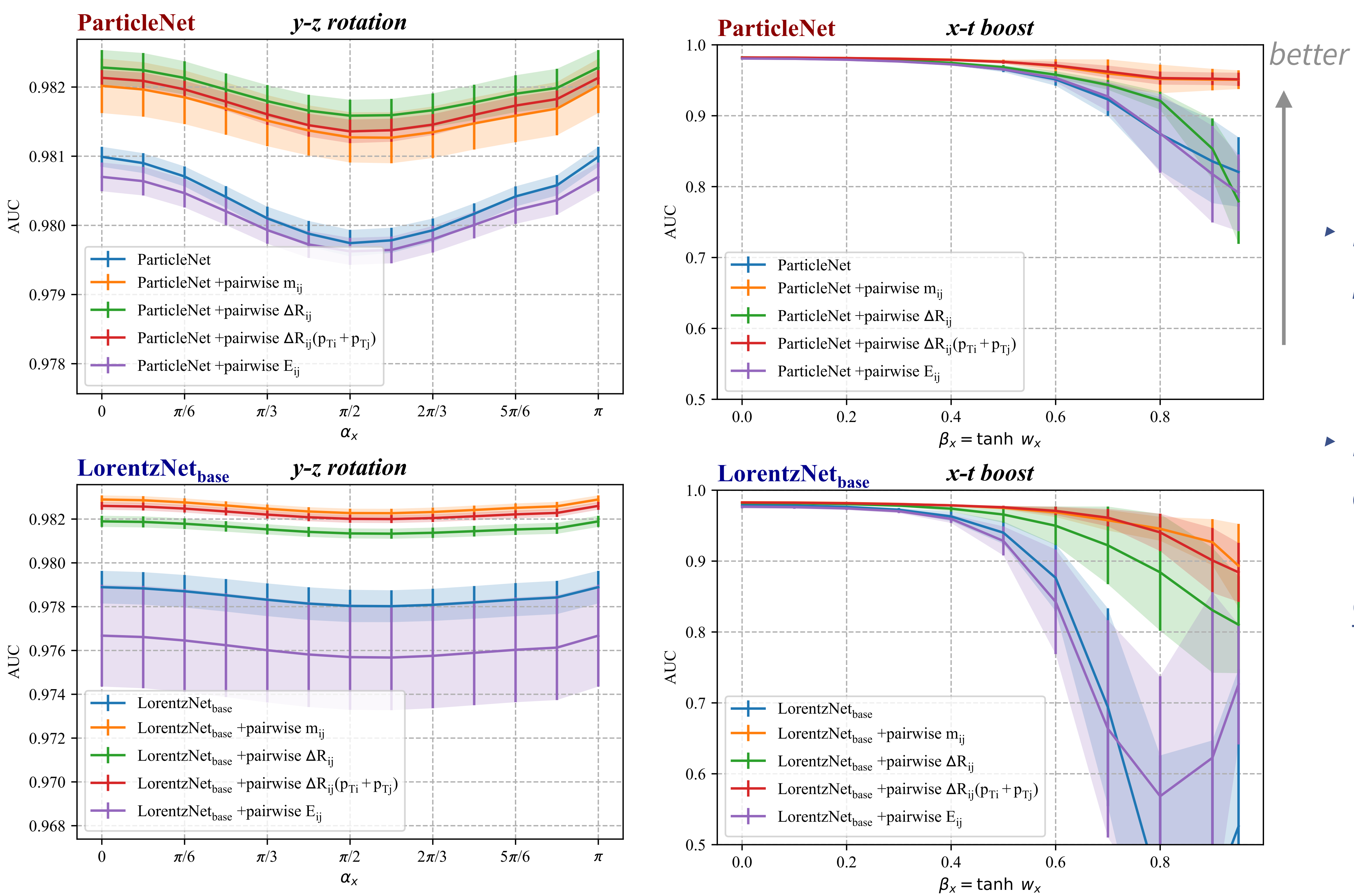
Performance for adding pairwise features

Training on 60k top tagging dataset (smaller dataset manifest the power of inductive bias)

Base model	Variation	Accuracy	AUC	$1/\epsilon_{\mathrm{B}}$ ($\epsilon_{\mathrm{S}} = 50\%$)	$1/\epsilon_{\mathrm{B}}$ ($\epsilon_{\mathrm{S}} = 30\%$)
ParticleNet	—	0.9310(3)	0.9810(2)	198 ± 7	640 ± 29
	+pairwise: m_{ij}	0.9334(8)	0.9820(4)	222 ± 13	722 ± 52
	+pairwise: ΔR_{ij}	0.9334(6)	**0.9823(3)**	$\mathbf{231 \pm 10}$	$\mathbf{752 \pm 43}$
	+pairwise: $\Delta R_{ij}(p_{\mathrm{T},i} + p_{\mathrm{T},j})$	**0.9337(3)**	0.9821(1)	223 ± 6	741 ± 36
	+pairwise: E_{ij}	0.9303(5)	0.9807(2)	200 ± 6	651 ± 23
$\mathrm{LorentzNet}_{\mathrm{base}}$	—	0.9276(12)	0.9789(7)	172 ± 13	581 ± 53
	+pairwise: m_{ij}	**0.9347(4)**	**0.9829(2)**	$\mathbf{260 \pm 6}$	$\mathbf{931 \pm 50}$
	+pairwise: ΔR_{ij}	0.9328(4)	0.9819(3)	232 ± 10	807 ± 35
	+pairwise: $\Delta R_{ij}(p_{\mathrm{T},i} + p_{\mathrm{T},j})$	0.9342(4)	0.9826(2)	251 ± 6	919 ± 34
	+pairwise: E_{ij}	0.9243(37)	0.9767(23)	144 ± 29	485 ± 108

better compared to baselines

Performance for adding pairwise features

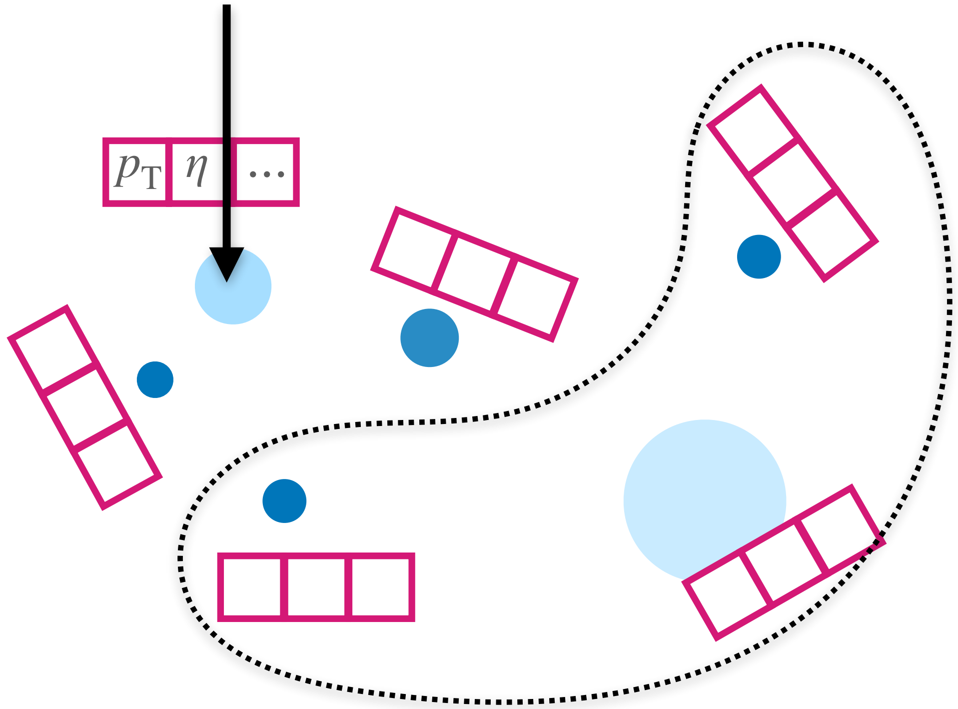


- ***Injecting ΔR to the network*** → *more robust to y-z rotation*
- ***Injecting $\Delta R(p_{Ti}+p_{Tj})$ or mass*** → *more robust to y-z rotation and now also the x-boost*

A general solution?

- ➔ Pairwise features have limitations
  - ❖ only applicable to GNN/Transformer networks which intrinsically build “edges”
- ➔ Upgrade to node-wise features
  - ❖ “mass features” carried per node, not edge between nodes

(1) for each node i



(2) find a ***friend group*** *G_i: composed of k nodes $i_m\,(m = 1,\cdots,k)$ having largest $p_i^\mu p_{i_m\mu}$*

this is a Lorentz invariant choice

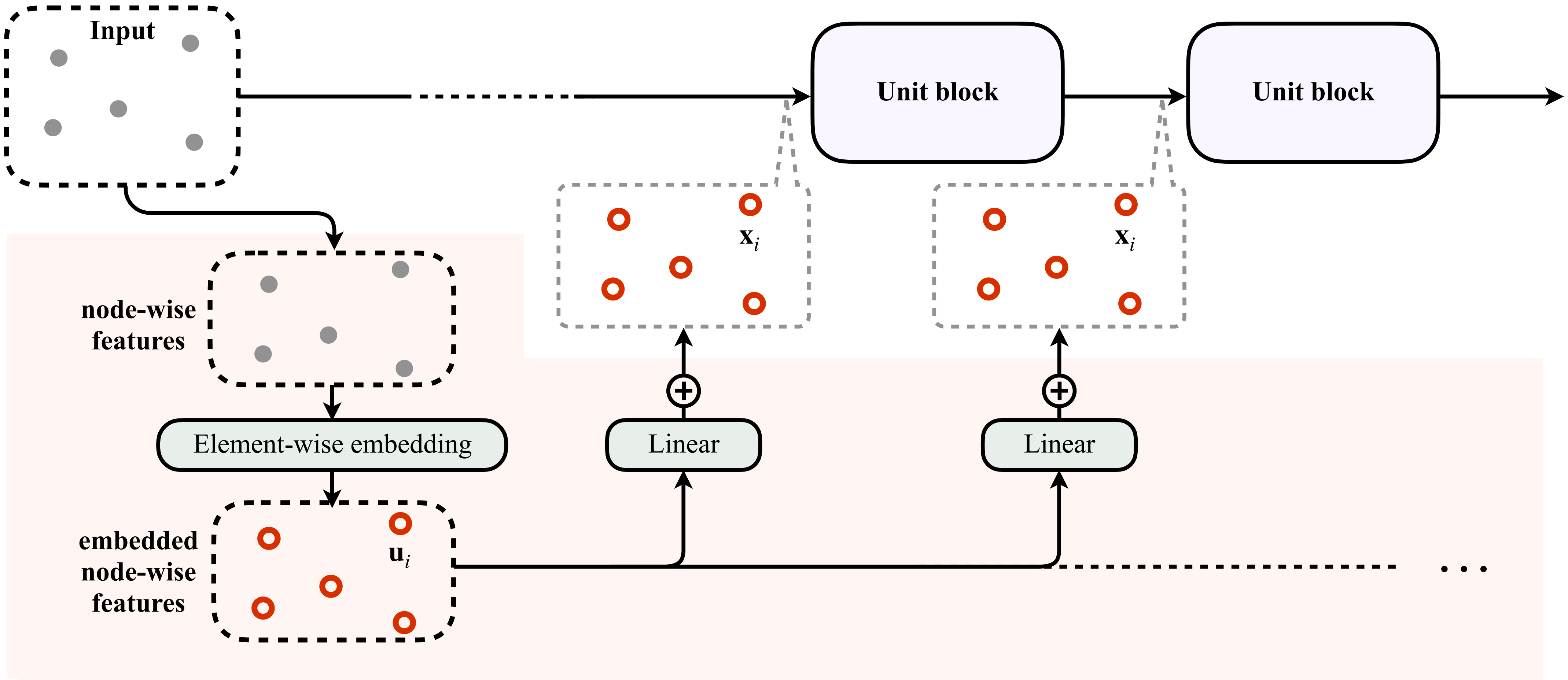
(3) calculate mass

$$m_{G_i}^2 = \Big(\sum_{j\in G_i} p_j\Big)^2 \approx 2 \sum_{j,k\in G_i}^{j<k} p_j^\mu p_{k\mu}$$

essentially, this is the pre-determined linear combination of all pairwise masses!

A general patch structure design?

Any baseline network

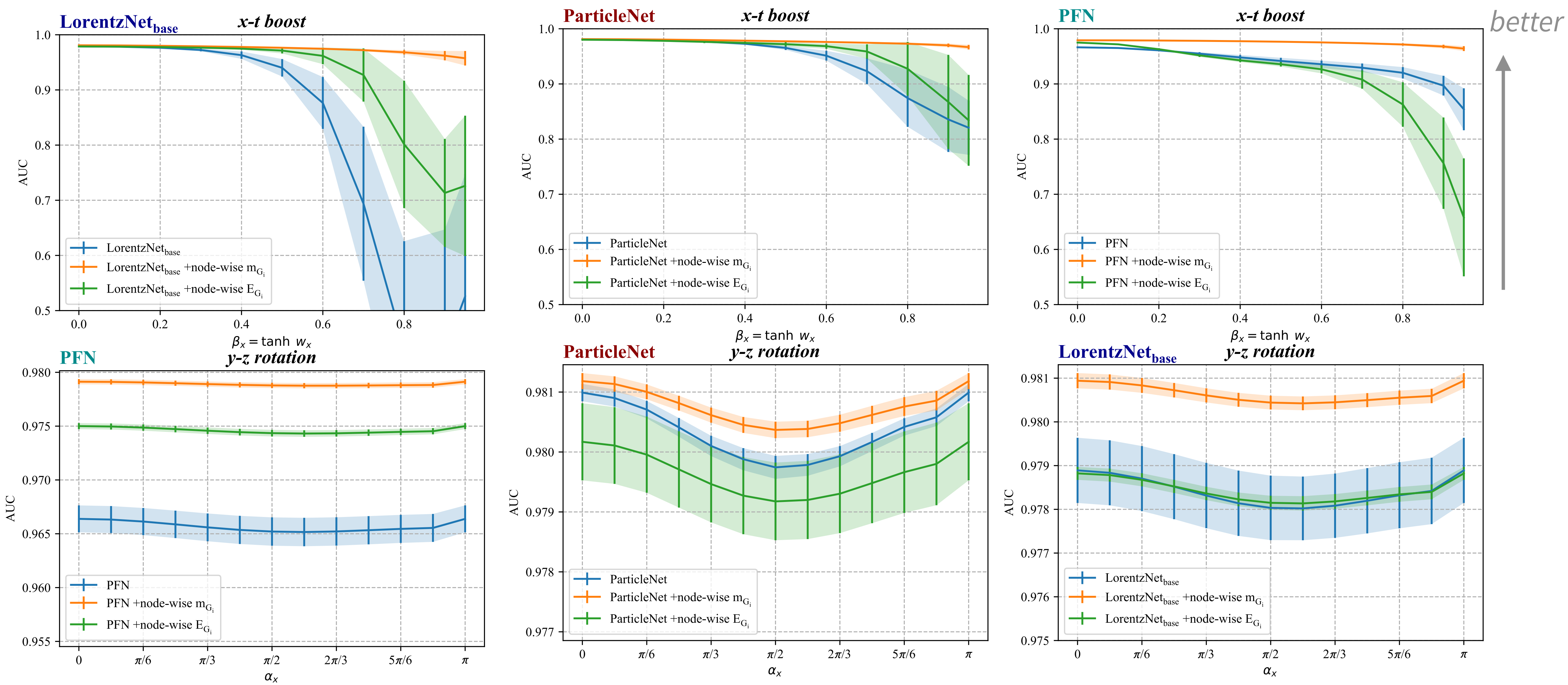


- ➔ Baseline networks can be any network that treats jet as a point cloud
- ➔ Integrate new node-wise features layer-by-layer
  - ❖ unit block is $\Phi(x)$ function for PFN, EdgeConv for ParticleNet, and LEGB for LorentzNet

Performance for adding node-wise features

Base model	Variation	Accuracy	AUC	$1/\epsilon_B$ ($\epsilon_S = 50\%$)	$1/\epsilon_B$ ($\epsilon_S = 30\%$)
PFN	—	0.9104(12)	0.9664(13)	67 ± 5	198 ± 21
	+node-wise: m_{G_i}	**0.9281(4)**	**0.9791(2)**	$\mathbf{184 \pm 5}$	$\mathbf{714 \pm 50}$
	+node-wise: E_{G_i}	0.9207(4)	0.9750(3)	125 ± 3	378 ± 19
ParticleNet	—	0.9310(3)	0.9810(2)	198 ± 7	640 ± 29
	+node-wise: m_{G_i}	**0.9313(3)**	**0.9812(1)**	$\mathbf{222 \pm 5}$	$\mathbf{800 \pm 40}$
	+node-wise: E_{G_i}	0.9300(12)	0.9802(6)	183 ± 12	572 ± 47
LorentzNet$_{\text{base}}$	—	0.9276(12)	0.9789(7)	172 ± 13	581 ± 53
	+node-wise: m_{G_i}	**0.9306(3)**	**0.9809(2)**	$\mathbf{219 \pm 3}$	$\mathbf{887 \pm 36}$
	+node-wise: E_{G_i}	0.9272(3)	0.9788(1)	171 ± 2	562 ± 16

Adding node-wise mass:
(1) improve network performance (especially for PFN!)
(2) more robust to Lorentz transformations on test data
(3) smaller error bars (illustrate more generalization ability)



Performance summary

→ What do the results mean?

- the full network tends to be ***more robust and performant***, when we incorporate Lorentz-symmetry-preserved variables (pairwise/node-wise ones) into the network
- even when we ***introduce a very small patch structure*** invariant under a certain symmetry (the original network is unaffected) helps the network to perform better
  - without need to let the network fully satisfy Lorentz symmetries
  - invariance property of the small sub-network has a big impact on the learning, and can be reflected in the entire network

Base model	Variation	# parameters	FLOPs
PFN	—	83.84 k	4.46 M
	+node-wise	+26.19 k	+3.41 M
ParticleNet	—	366.16 k	535.73 M
	+pairwise	+34.91 k	+285.29 M
	+node-wise	+21.97 k	+2.83 M
$\text{LorentzNet}_{\text{base}}$	—	226.23 k	1997.69 M
	+pairwise	+0.43 k	+7.02 M
	+node-wise	+37.35 k	+4.8 M

Performance summary

➔ What do the results mean?

- the full network tends to be ***more robust and performant***, when we incorporate Lorentz-symmetry-preserved variables (pairwise/node-wise ones) into the network
- even when we ***introduce a very small patch structure*** invariant under a certain symmetry (the original network is unaffected) helps the network to perform better
  - without need to let the network fully satisfy Lorentz symmetries
  - invariance property of the small sub-
    can be reflected in the entire network

Base model	Variation	# parameters	FLOPs
PFN	—	83.84 k	4.46 M
	+node-wise	+26.19 k	+3.41 M
ParticleNet	—	366.16 k	535.73 M
	+pairwise	+34.91 k	+285.29 M
	+node-wise	+21.97 k	+2.83 M
$\text{LorentzNet}_{\text{base}}$	—	226.23 k	1997.69 M
	+pairwise	+0.43 k	+7.02 M
	+node-wise	+37.35 k	+4.8 M

- The experiments show that "**pairwise mass**" is the key component in network design
- **We reveal that the underlying logic lies in the Lorentz symmetry preservation**
- We make a successful attempt to understand the interpretability of the network in terms of symmetry preservation

Short summary

➔ We have some new findings from our experiments, potentially helpful to the HEP × ML community

✦ *Engineering on network? Engineering on data!*

Data engineering provides a more general solution for symmetry preservation - for that we simply construct some symmetry-invariant input, introduce a patch structure, and inject them into the baseline network of any kind

✦ *Just a "hint" on symmetry will do*

Creating a network to strictly obey a source of symmetry is a solution, but usually brings potential limitations to network designs; adding a small symmetry-preserving "patch" structure to the baseline networks can be a new solution

✦ *Full Lorentz symmetry is better!*

By delivering a fair comparison among networks respecting to full Lorentz symmetry or some of its sub-symmetries, we find that the former performs better

➔ One more thing… Some inspirations to other HEP × ML tasks?

Inspirations

➔ One more thing… Some inspirations to other HEP × ML tasks?

Tasks	Eamples	Inspirations on network design? 🤔
object-based tasks	• event-classifier (BDT/NN) for analysis, using e.g. jet/lep information • jet assignment tasks • autoencoders for event anomaly detection • …	consider adding "masses" between objects (jet–jet, jet–lep, etc) as additional input
particle-based tasks	• jet tagging! • ML-based paticle-flow reconstruction • …	consider adding "pairwise mass" between particles
tasks using more primary input?	• processing track hits • processing energy deposit on calorimeters (data on fixed grids)	**fact:** the essence of these data are still based on particles **open question:** can we somehow design a Lorentz-symmetry-preserving (sub)network to adapt these sources of input to further improve network performance?

MORE POSSIBILITIES AHEAD!

Backup

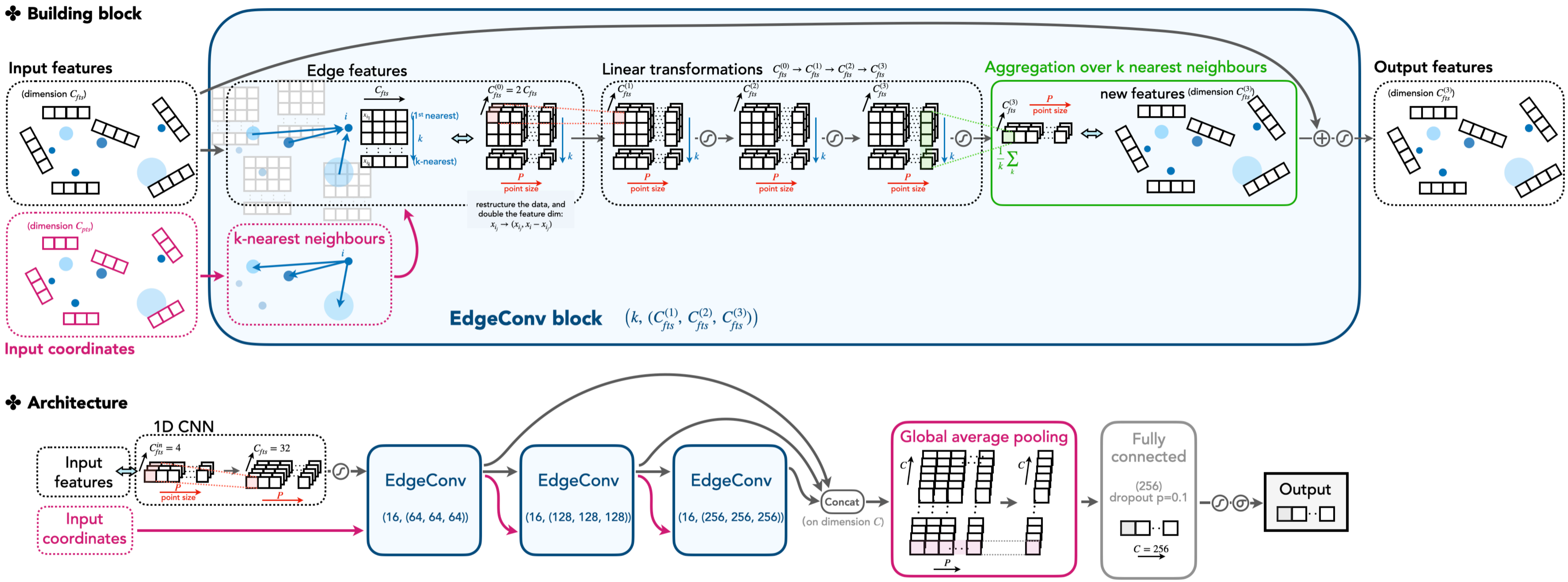
Recap on ParticleNet and LorentzNet

RECAP ON PARTICLENET

H.Qu, L.Gouskos. PRD 101 (2020) 056019

A powerful and popular model in the HEP community with a variety of applications

[image from link]



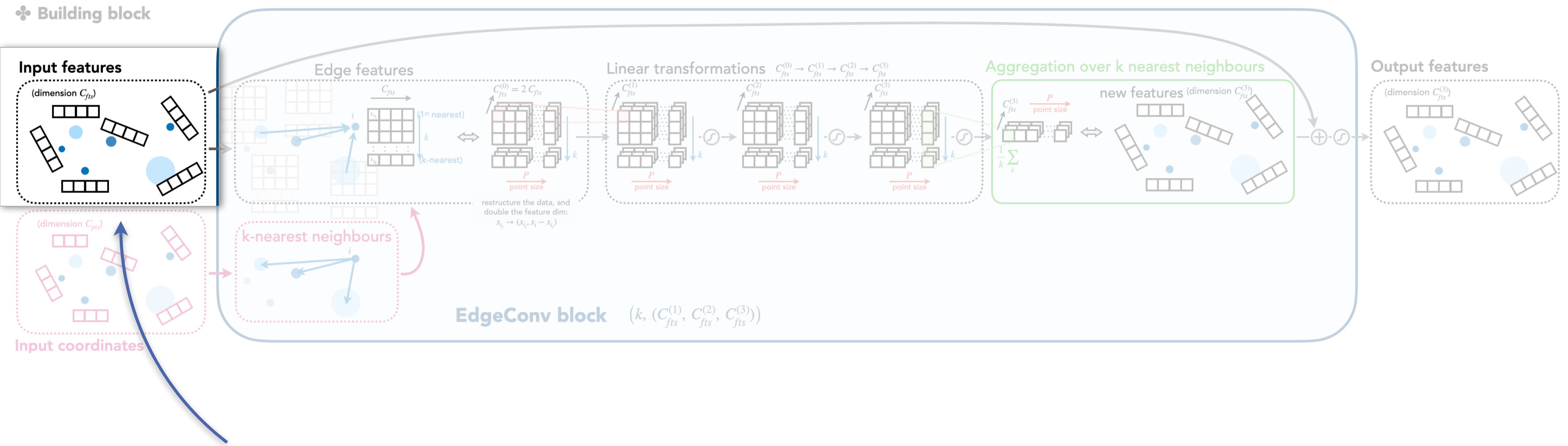
Recap on ParticleNet and LorentzNet

RECAP ON PARTICLENET

H.Qu, L.Gouskos. PRD 101 (2020) 056019

A powerful and popular model in the HEP community with a variety of applications

[image from link]



Point cloud representation of jet

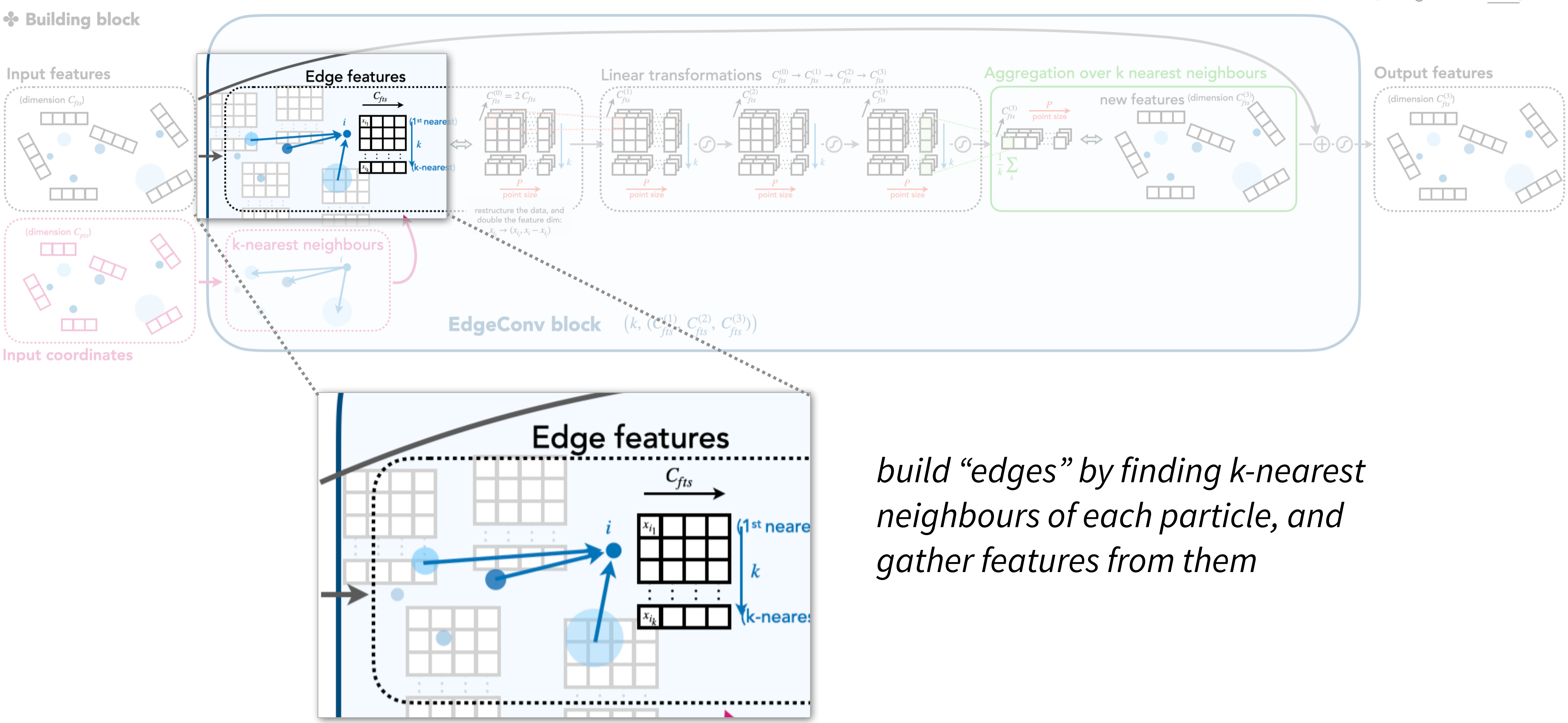
Recap on ParticleNet and LorentzNet

RECAP ON PARTICLENET

[H.Qu, L.Gouskos. PRD 101 (2020) 056019]

A powerful and popular model in the HEP community with a variety of applications

[image from link]



build "edges" by finding k-nearest neighbours of each particle, and gather features from them

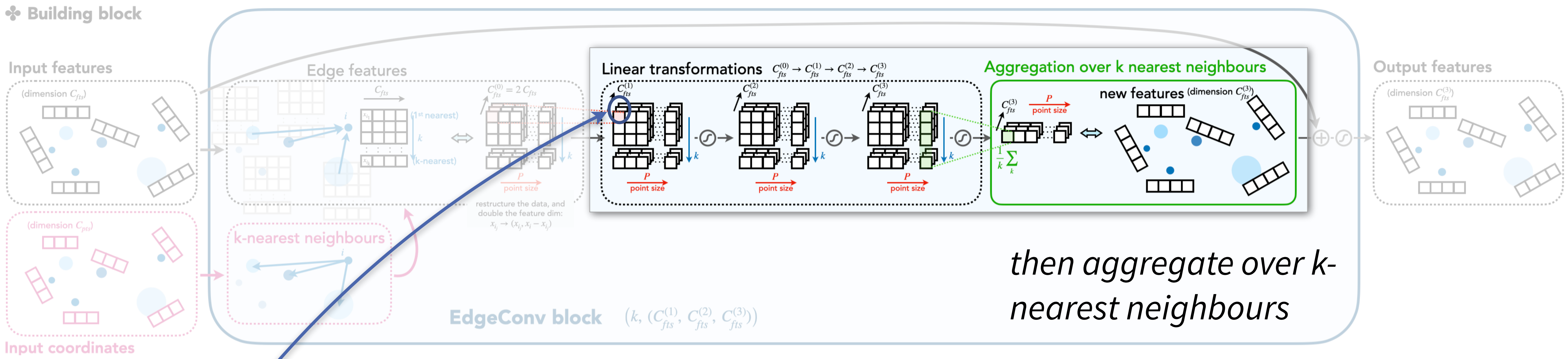
Recap on ParticleNet and LorentzNet

RECAP ON PARTICLENET

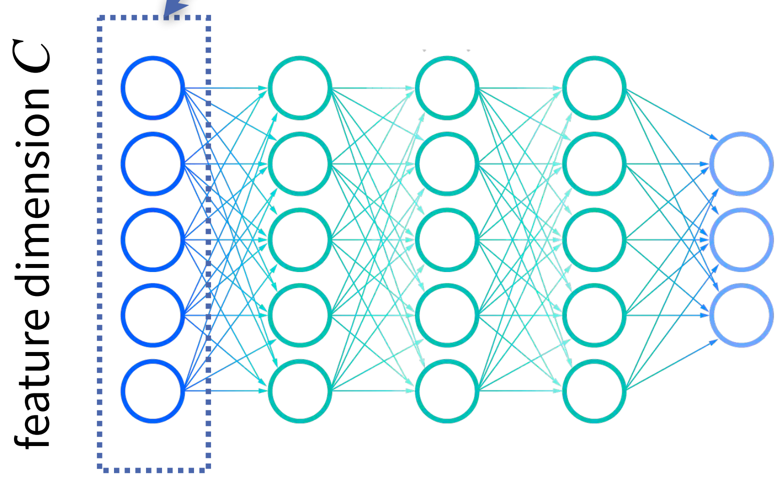
H.Qu, L.Gouskos. PRD 101 (2020) 056019

A powerful and popular model in the HEP community with a variety of applications

[image from link]



then aggregate over k-nearest neighbours

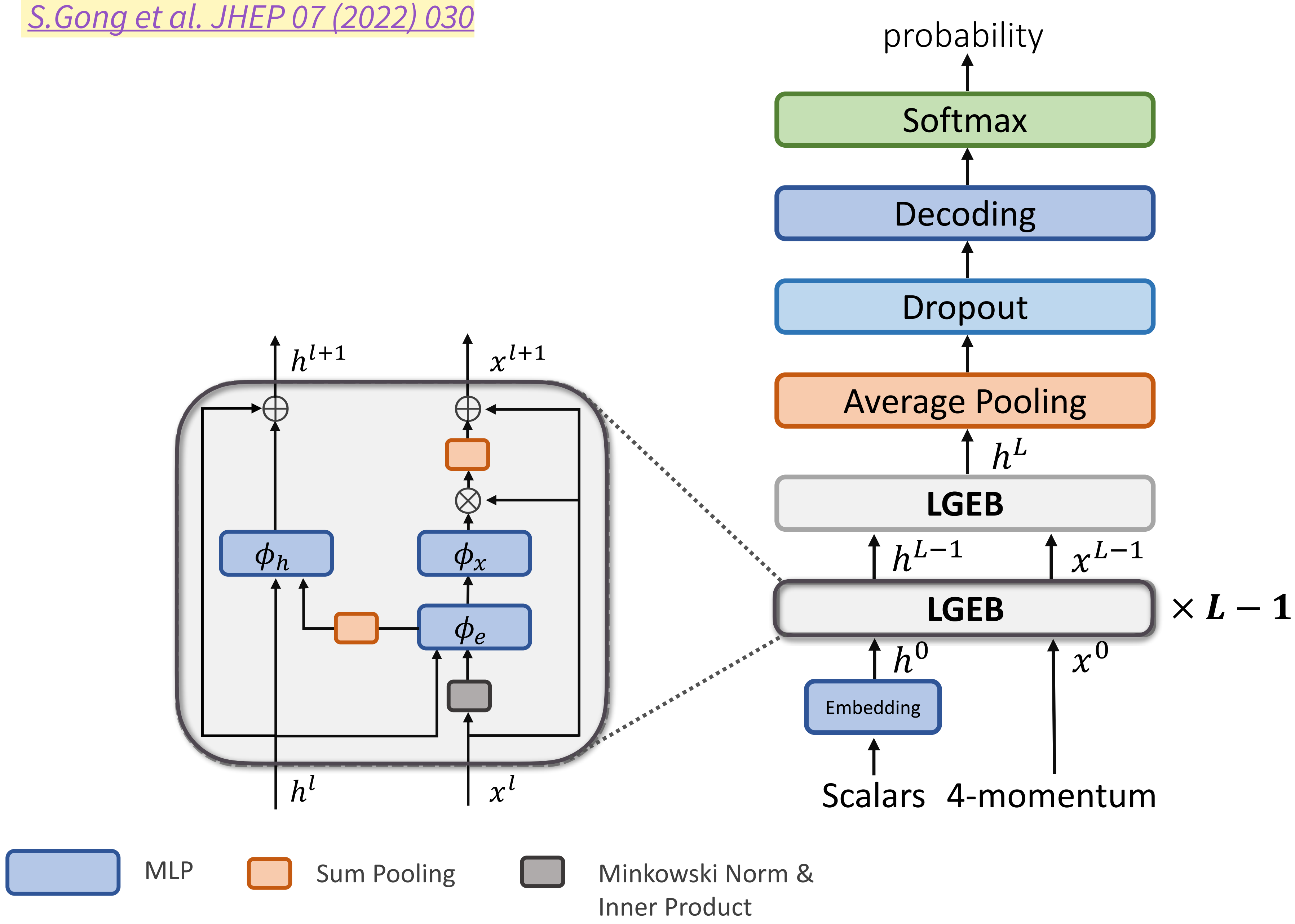


do linear transformation

Recap on ParticleNet and LorentzNet

RECAP ON LORENTZNET

S.Gong et al. JHEP 07 (2022) 030

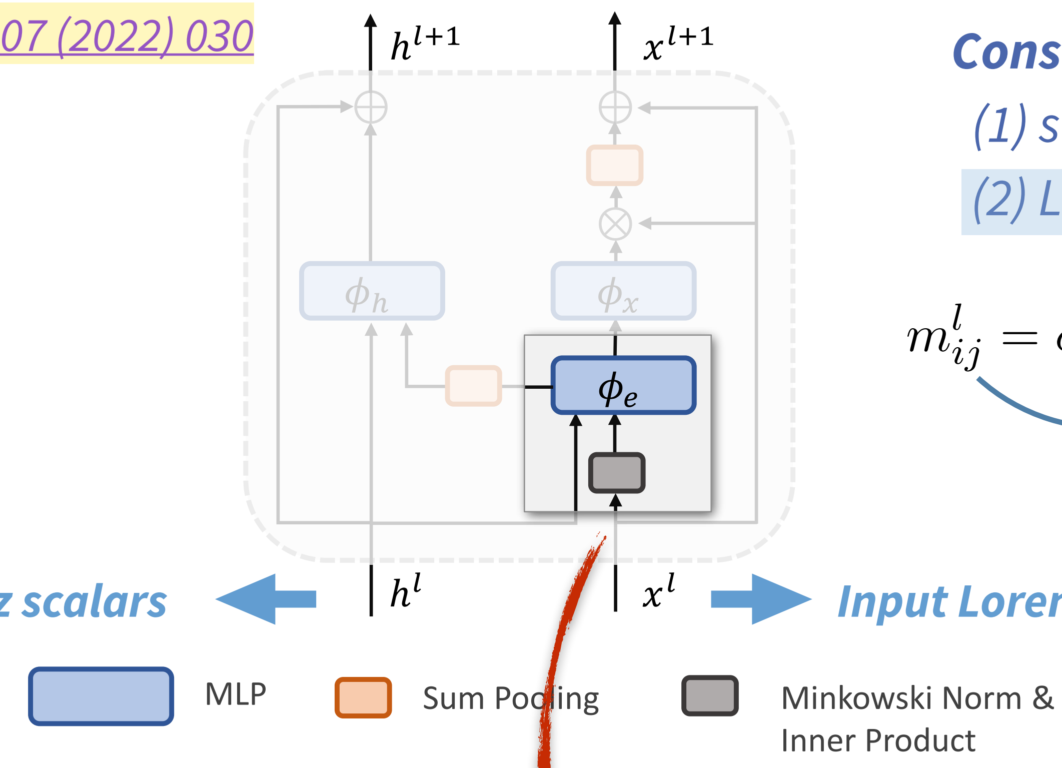


Lorentz Group Equivariant Block (LGEB)

Recap on ParticleNet and LorentzNet

RECAP ON LORENTZNET

S.Gong et al. JHEP 07 (2022) 030



Input Lorentz scalars

Input Lorentz vector

Lorentz Group Equivariant Block (LGEB)

Construct edge features from

(1) scalars from two nodes

(2) Lorentz inner product from two vectors

$$m_{ij}^l = \phi_e\left(h_i^l, h_j^l, \psi(\|x_i^l - x_j^l\|^2), \psi(\langle x_i^l, x_j^l\rangle)\right)$$

hence it's a Lorentz scalar

from node to edge

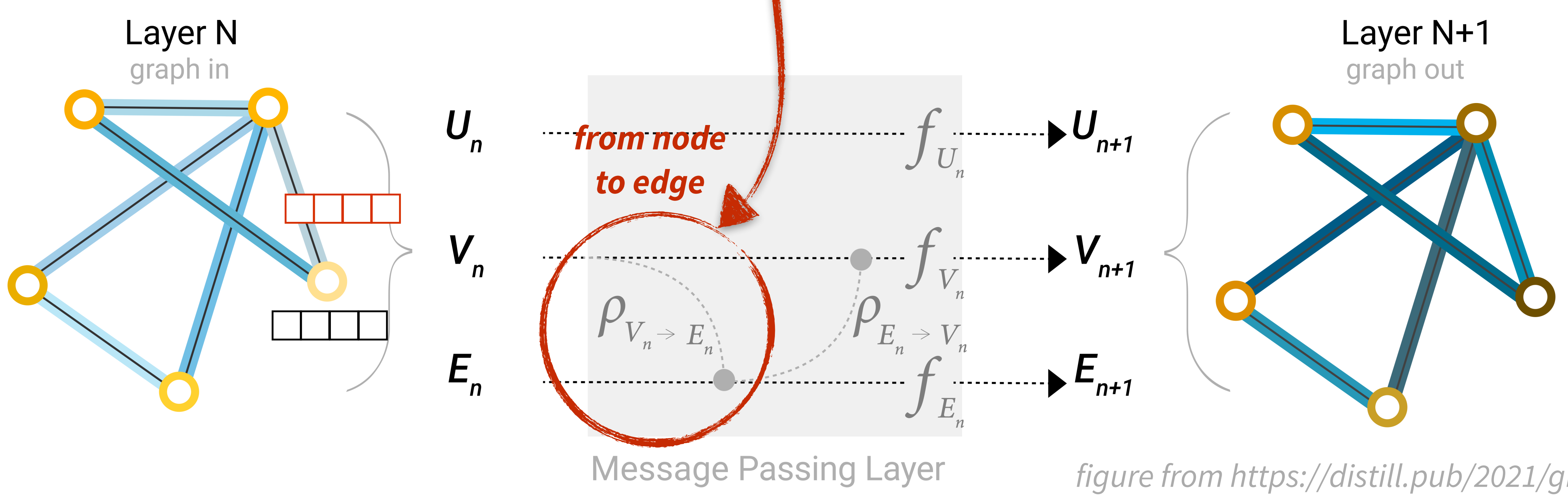
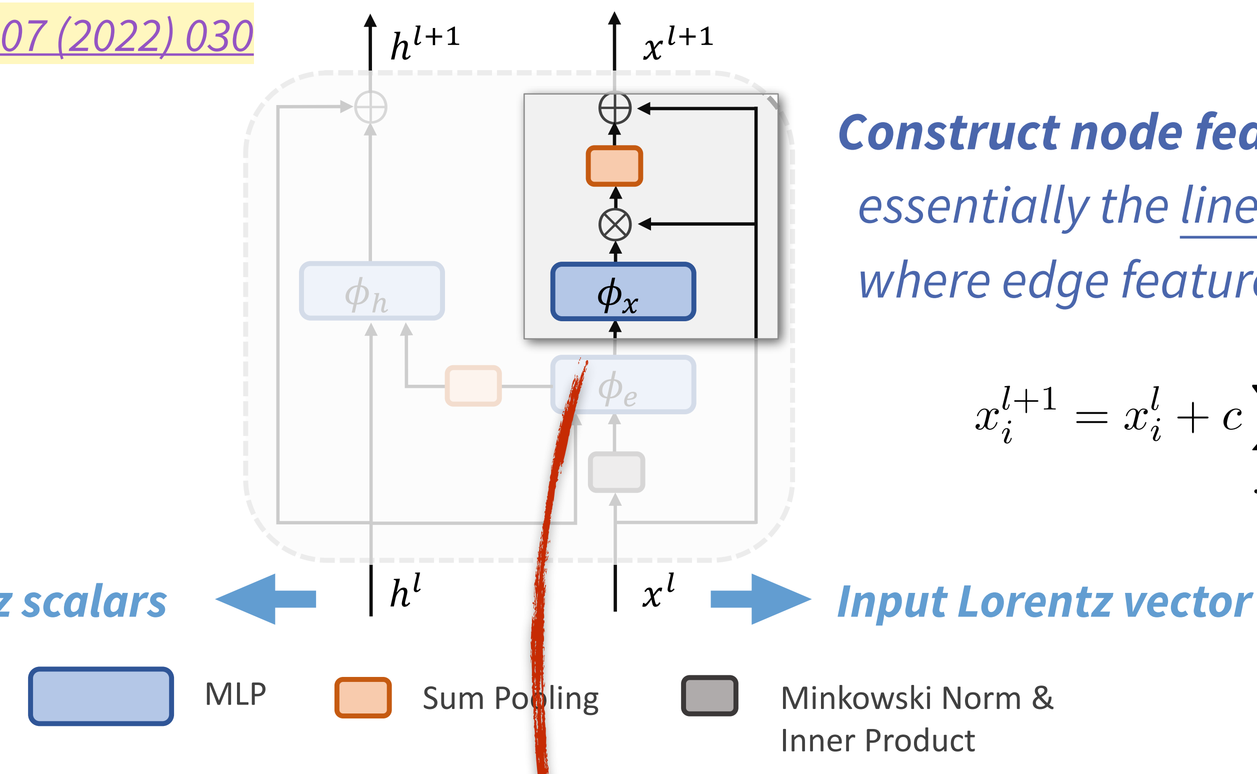


figure from https://distill.pub/2021/gnn-intro/

Recap on ParticleNet and LorentzNet

RECAP ON LORENTZNET

S.Gong et al. JHEP 07 (2022) 030



Lorentz Group Equivariant Block (LGEB)

Construct node features (vector):

essentially the linear combination of all vectors, where edge features are "weights"

$$x_i^{l+1} = x_i^l + c \sum_{j \neq i} \phi_x(m_{ij}^l) \cdot x_j^l$$

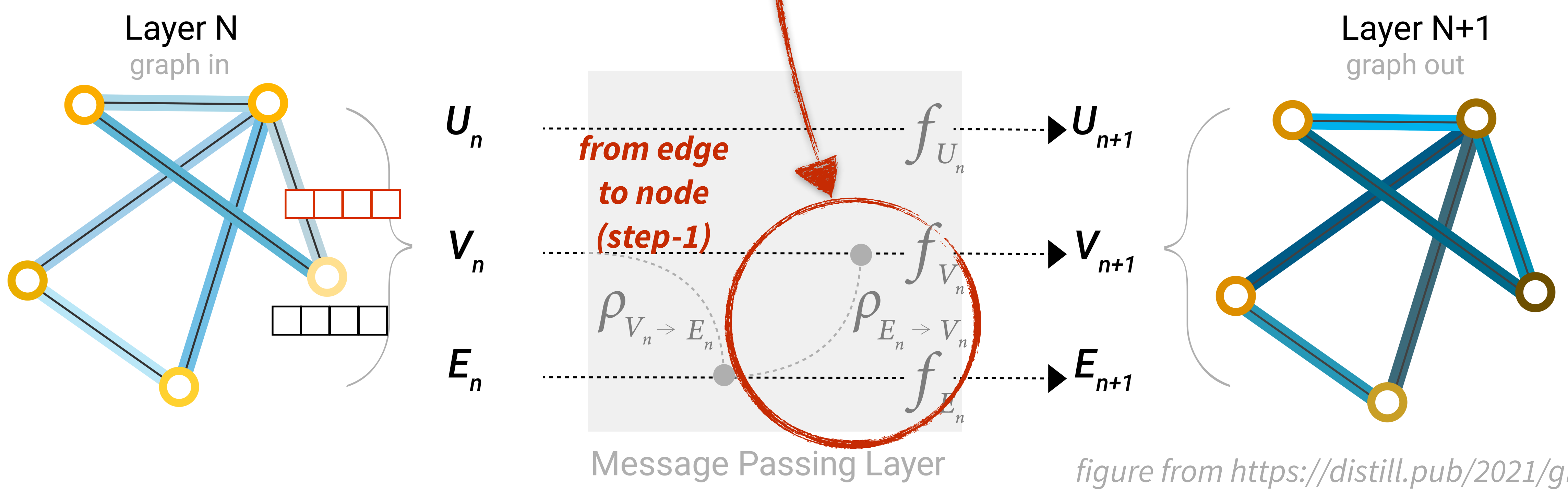


figure from https://distill.pub/2021/gnn-intro/

Recap on ParticleNet and LorentzNet

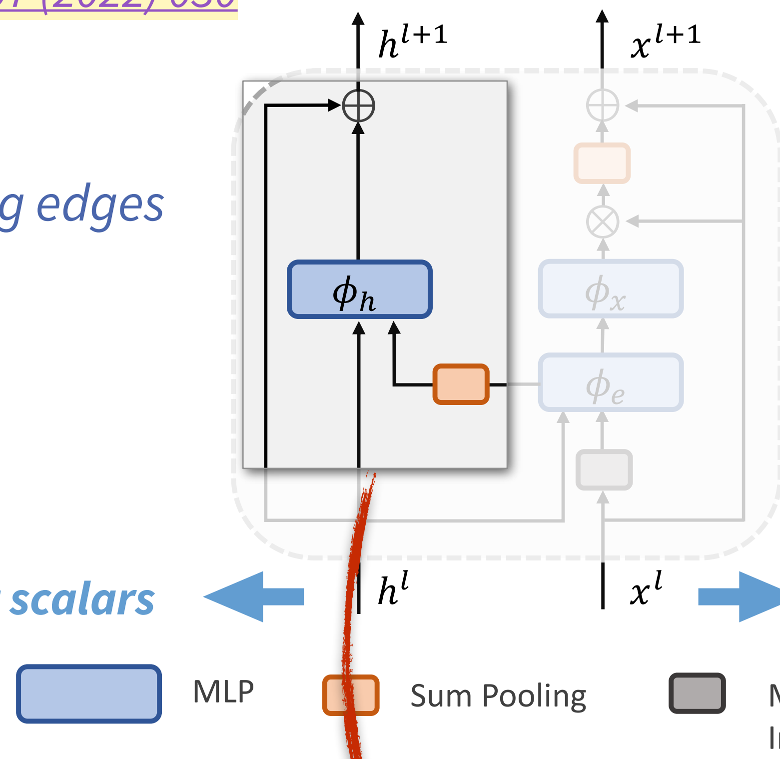
RECAP ON LORENTZNET

S.Gong et al. JHEP 07 (2022) 030

Construct node features (scalar):

attentive pooling on all connecting edges

$$h_i^{l+1} = h_i^l + \phi_h(h_i^l, \sum_{j \neq i} w_{ij} m_{ij}^l)$$



Input Lorentz scalars

Input Lorentz vector

Lorentz Group Equivariant Block (LGEB)

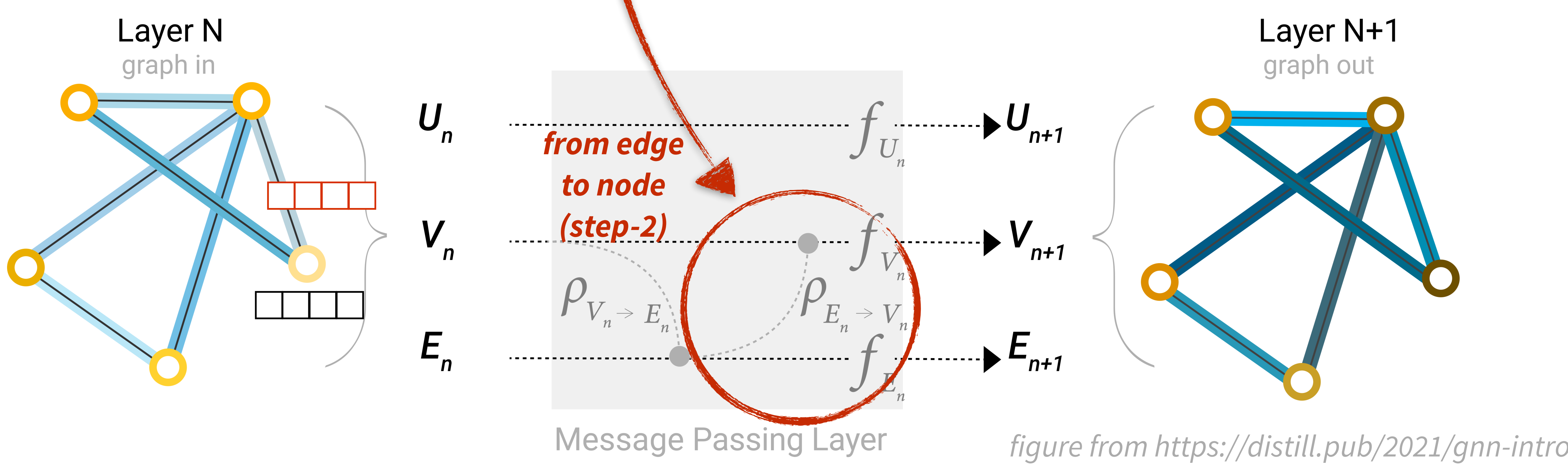


figure from https://distill.pub/2021/gnn-intro/