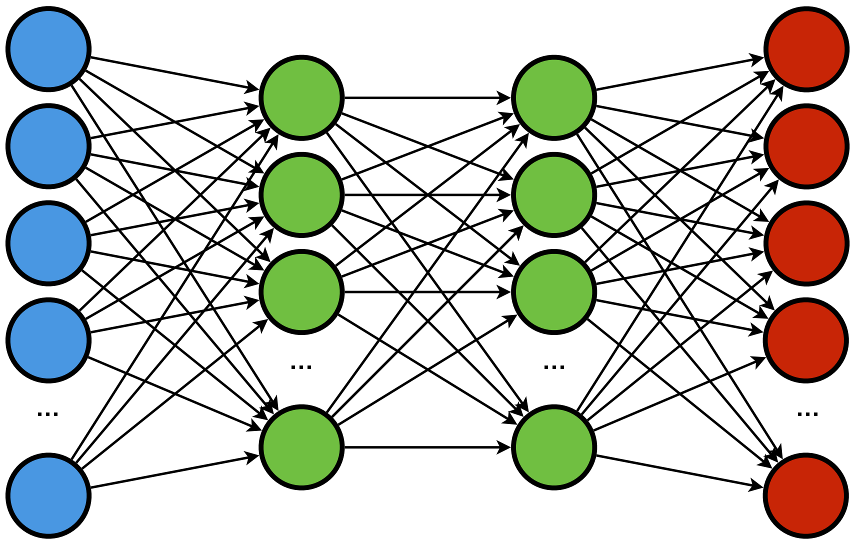
+



Lorentz Boost Networks

– Autonomous Physics-Inspired Feature Engineering –

Martin Erdmann, Erik Geiser, Yannik Rath, Marcel Rieger

HAP Workshop

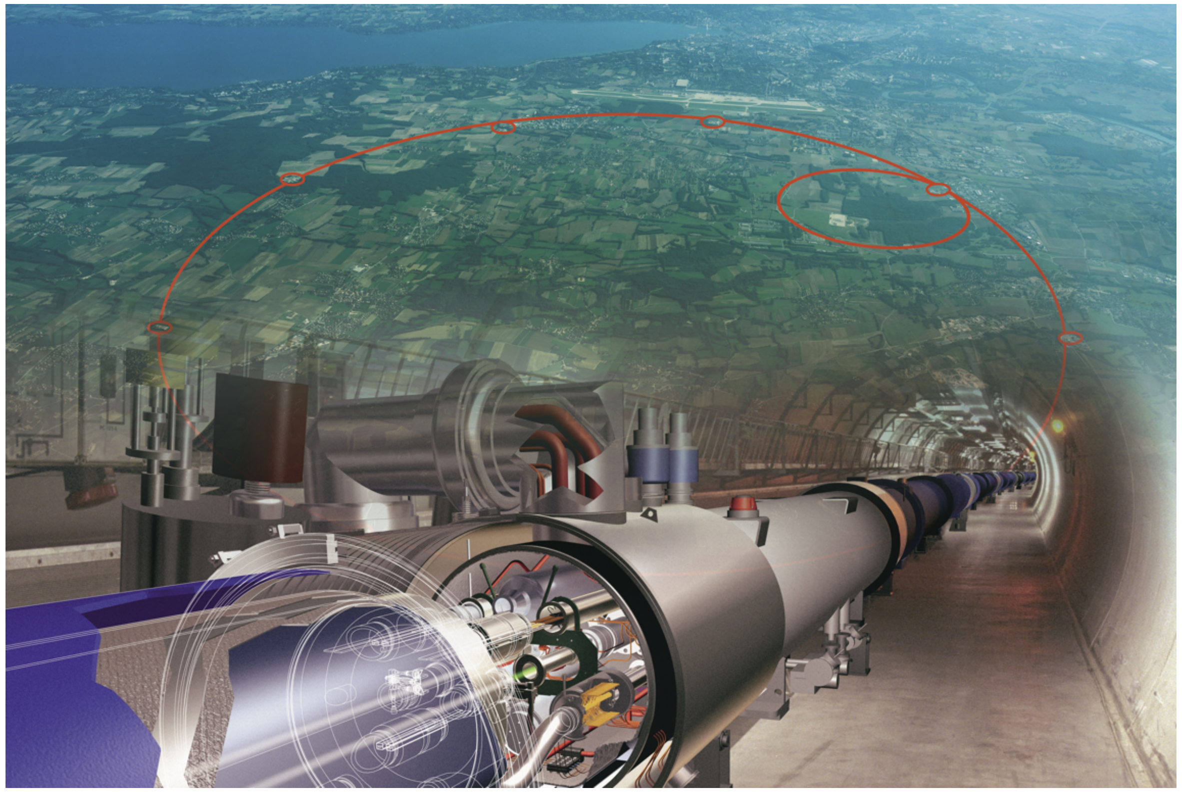
19 February 2019

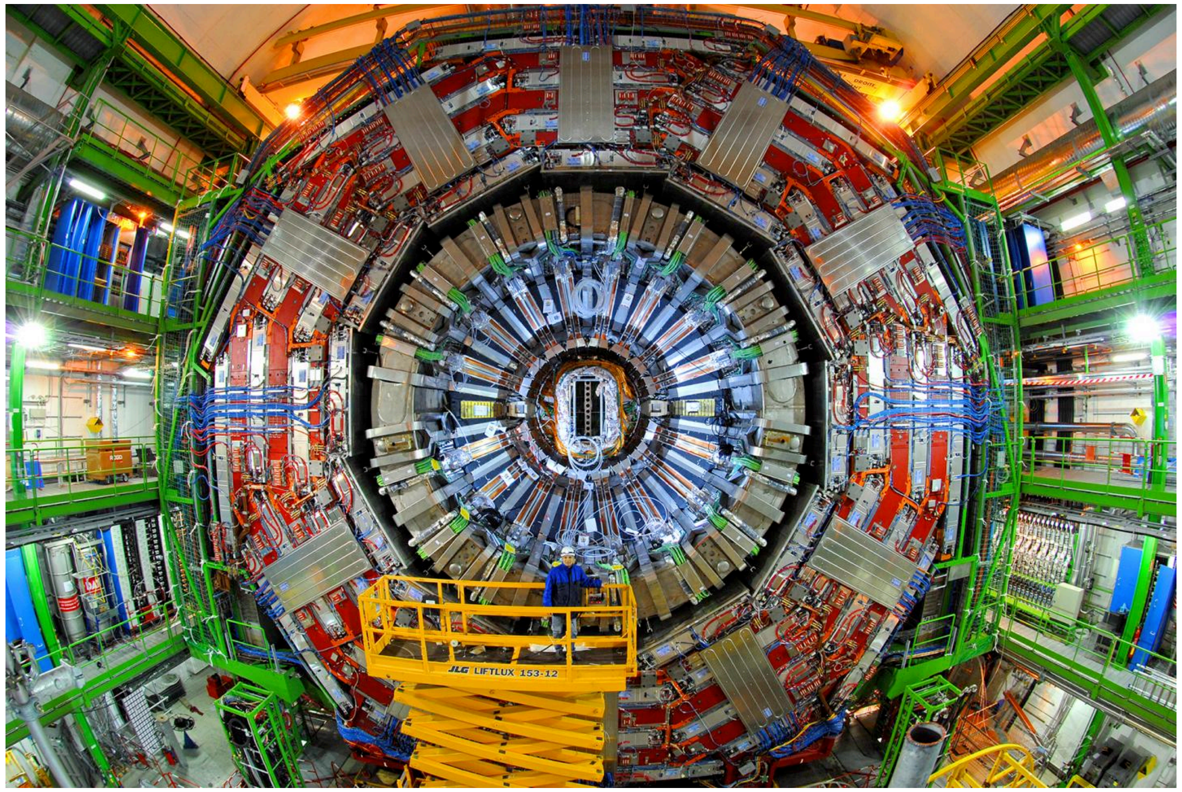
The Large Hadron Collider

- pp collider @ CERN
- 26.7 km, 1600 magnets, 8.3 T
- $\sqrt{s} = 13$ TeV (2016-2018)
- 800 mio. events / s

The Compact Muon Solenoid

- Multi-purpose detector
  → SM, Higgs, SUSY, DM, extra dim.
- 21 x 15 m (l x ⌀), 12.500 tons
- 3.8 T solenoid

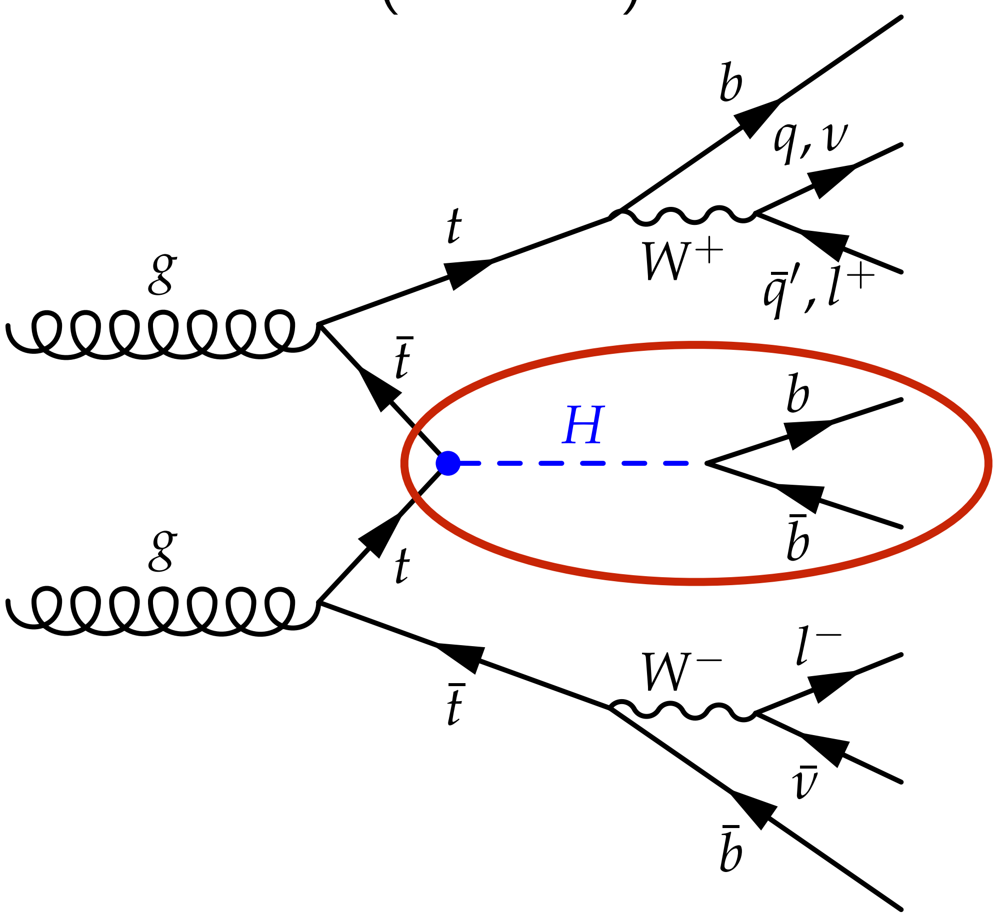




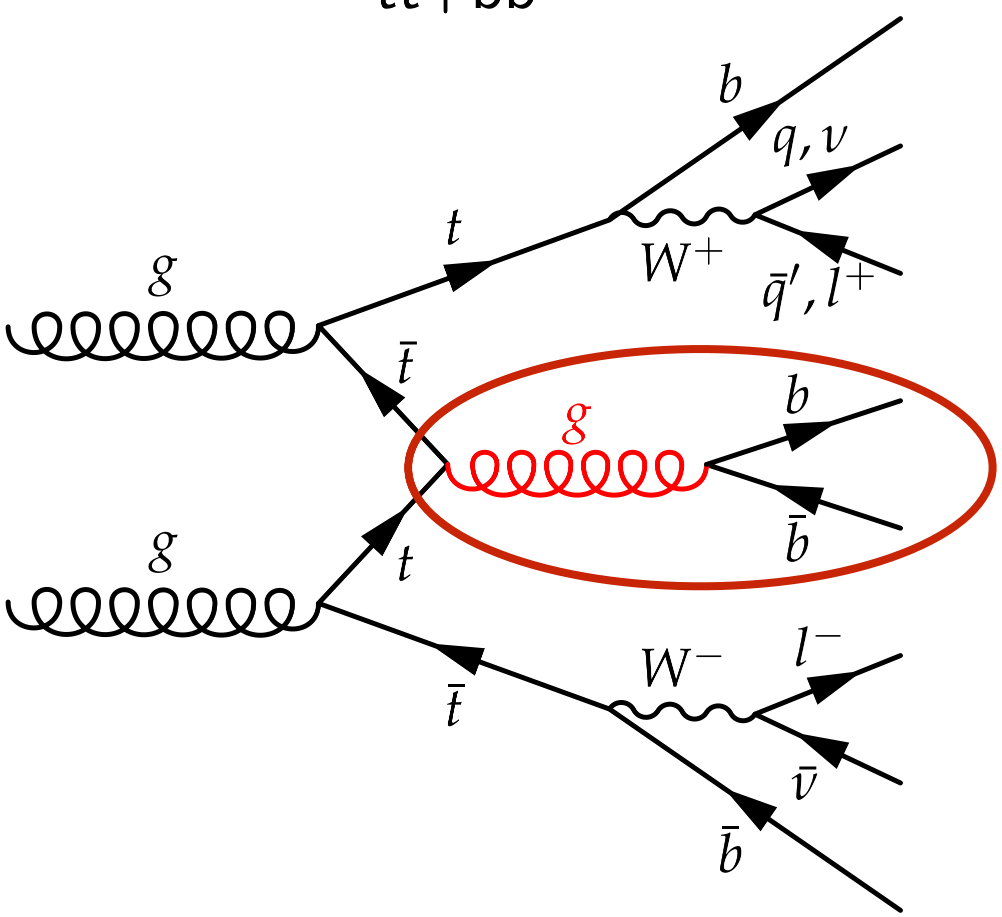
Marcel Rieger - 19.2.19

- **Classification task**: separate ttH (H → bb) from tt+bb

ttH (H → bb)



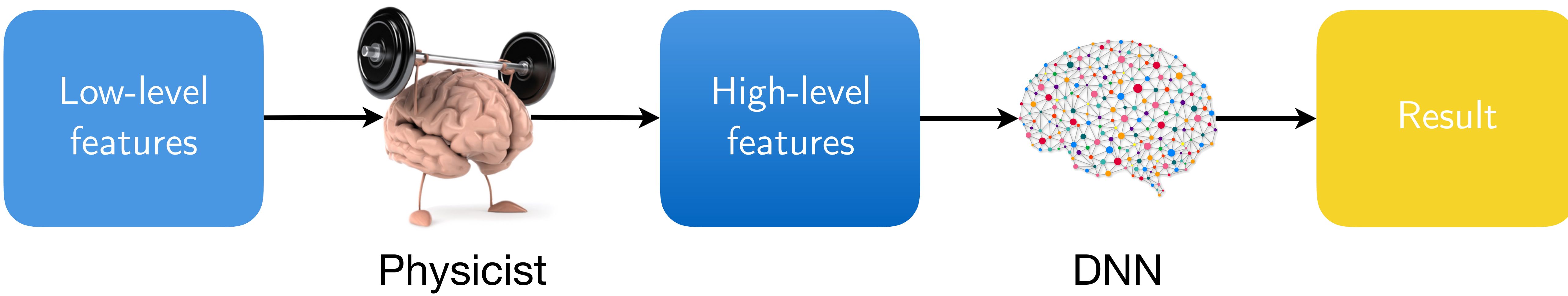
tt+bb



- **Final state**:
  - 6 jets (2 from tops, 2 from W, 2 from Higgs)
  - 1 charged lepton
  - 1 neutrino (missing transverse energy)

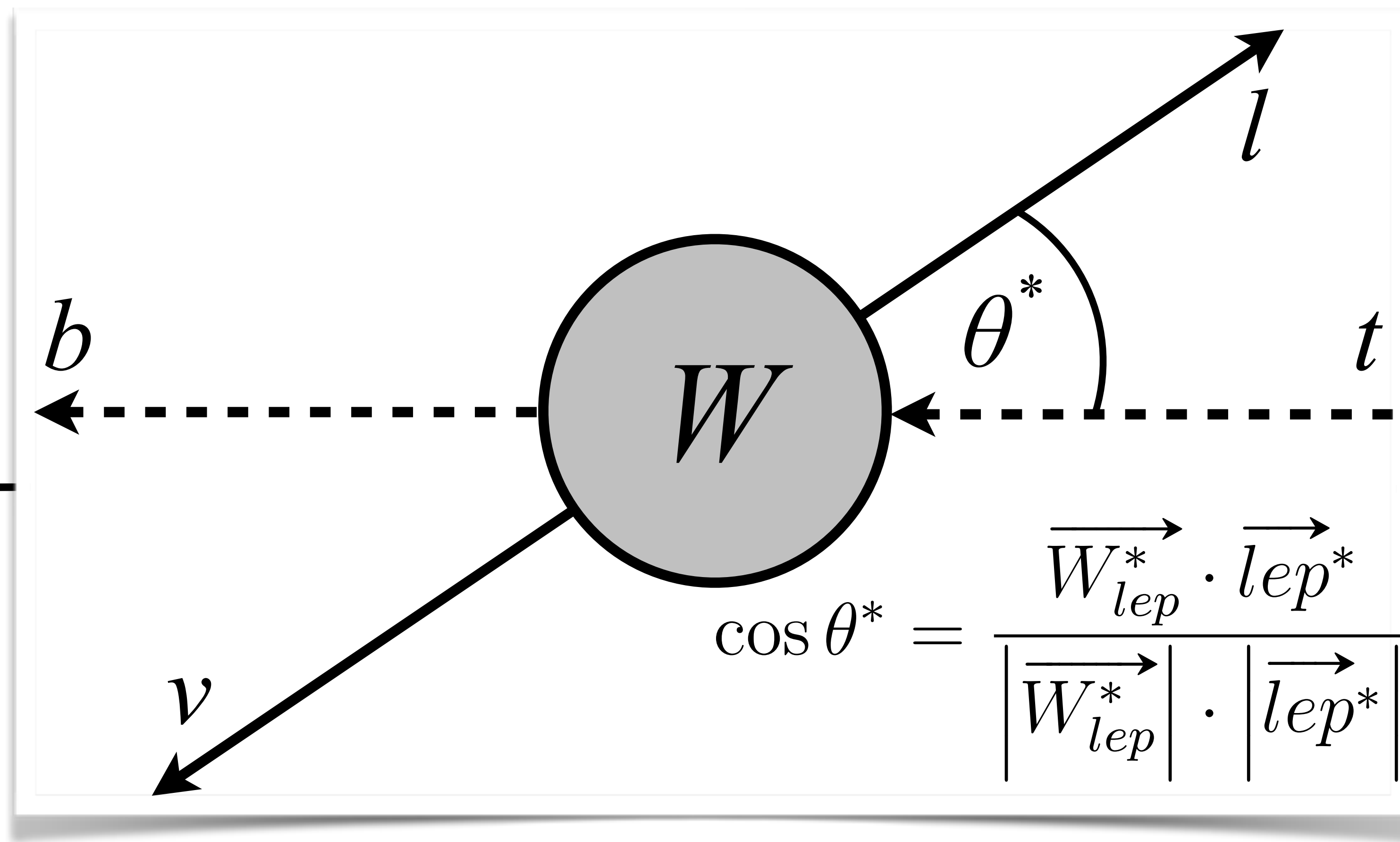
} 8 x four-vector components
E, p_x, p_y, p_z
→ **Low-level features**

Feature engineering in HEP



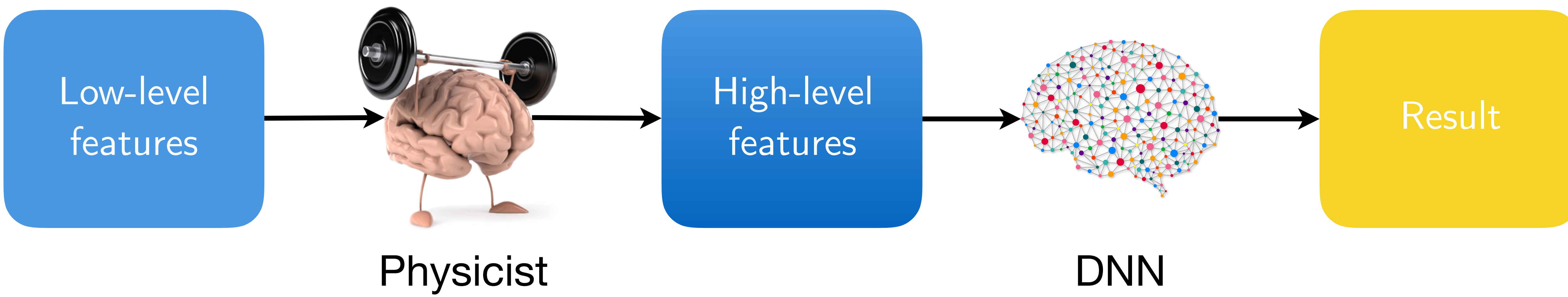
4 Feature engineering in HEP

Low-level features

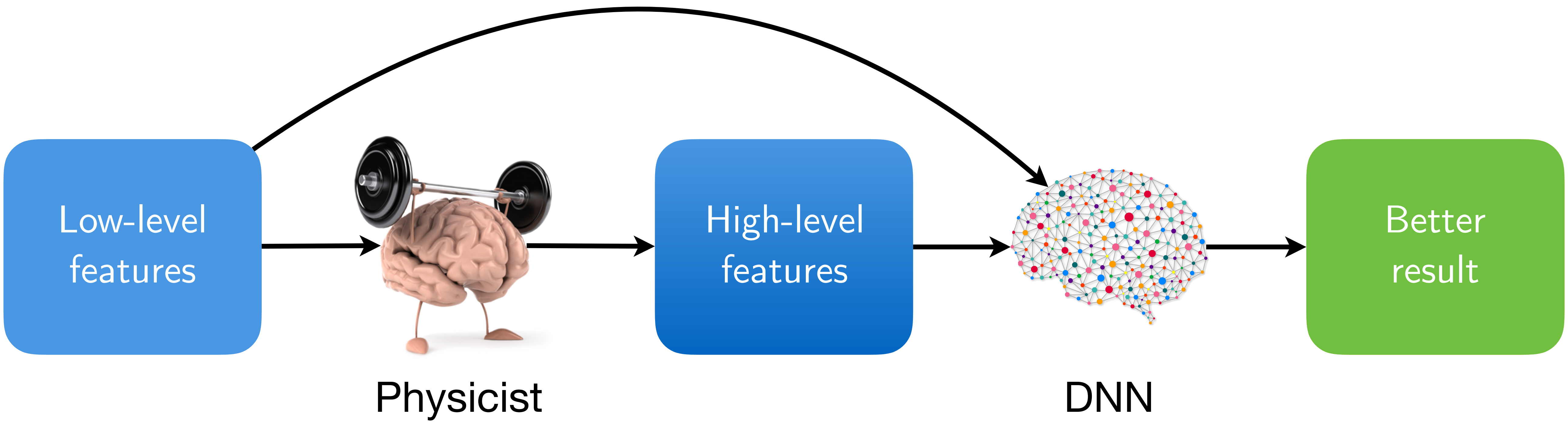


Result

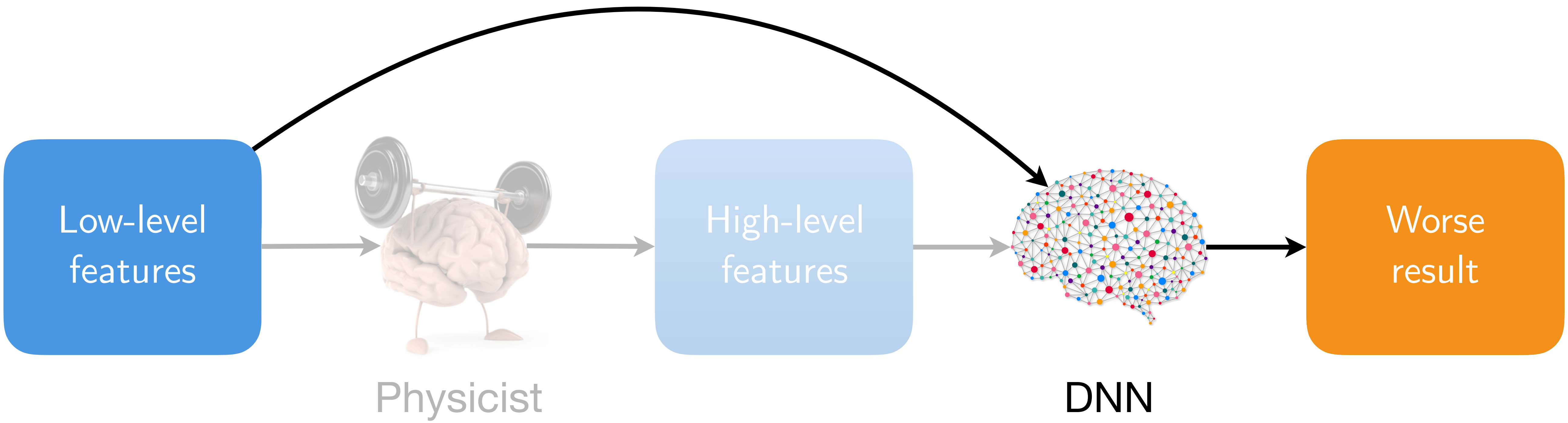
Feature engineering in HEP

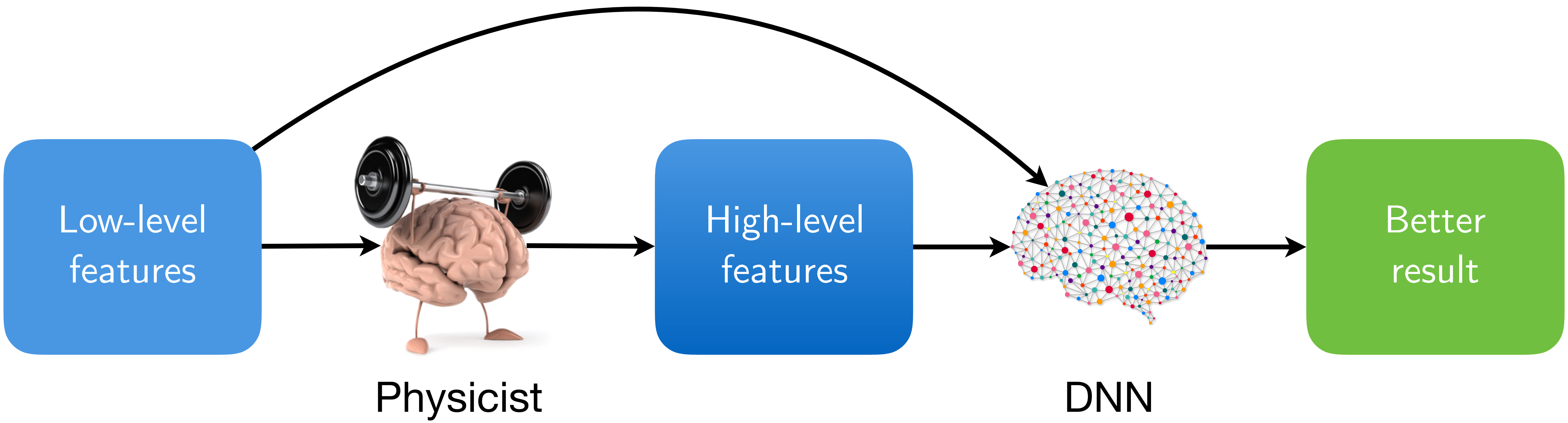


Feature engineering in HEP



Feature engineering in HEP

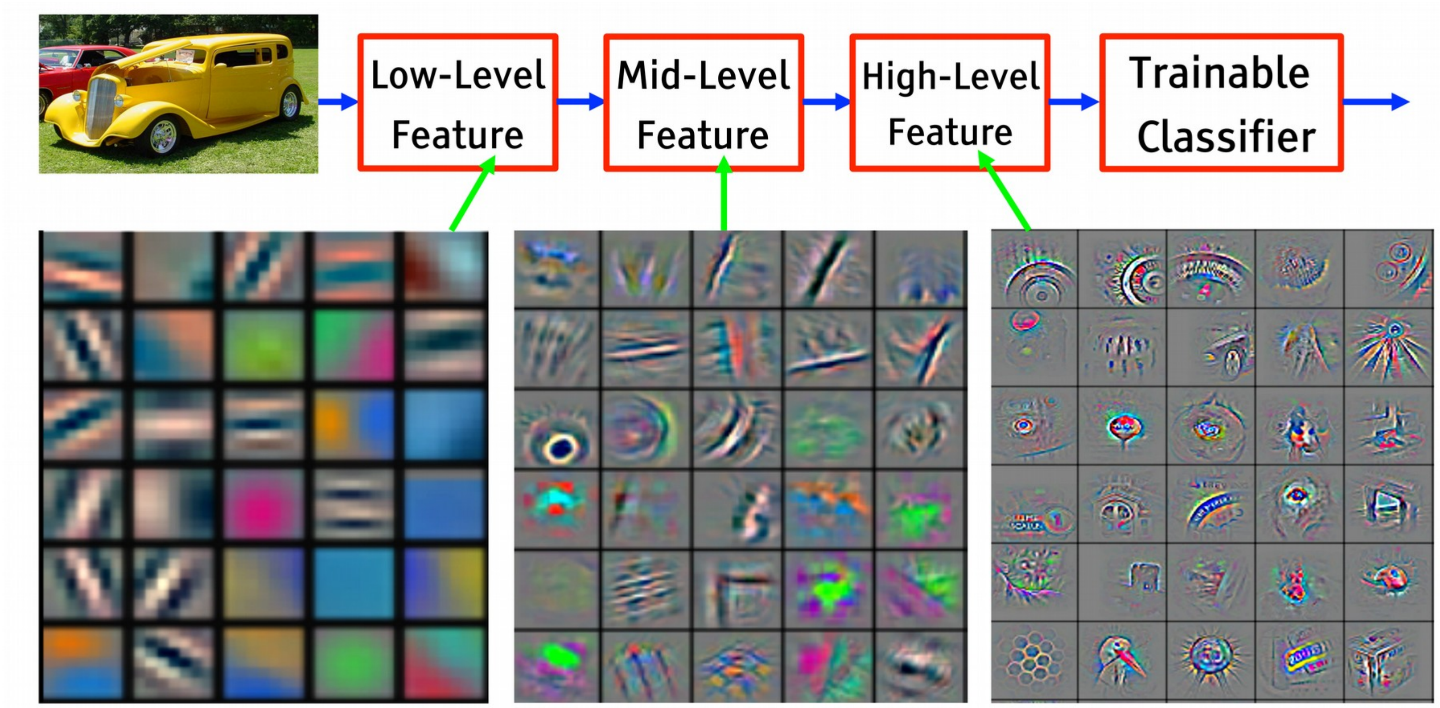




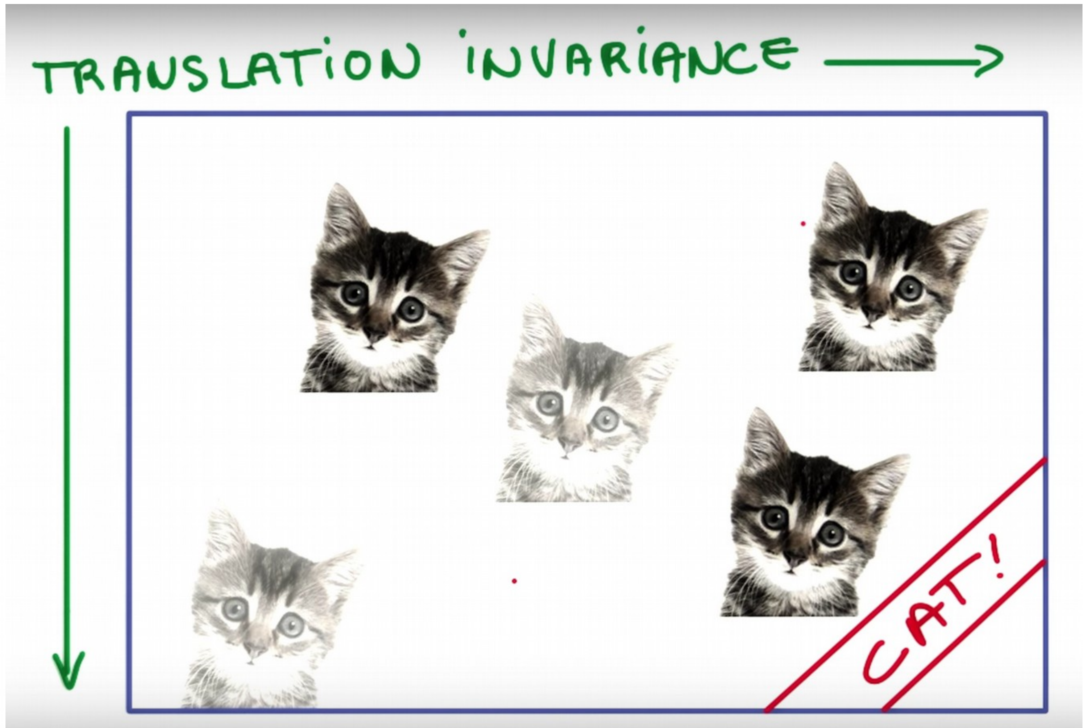
- Observations

  1. Physicists' crafted high-level features might not exploit all available information
  2. In practice, it is hard for "standard" DNNs to learn representations of complex features

- Similar situation to **FCNs → CNNs**:
  - Images contain information in translation invariant adjacency of pixels
  - → Exploit information by changing the network structure!



[Zeiler & Fergus 2013], adapted by Yann LeCun



Udacity Course 730, Deep Learning

→ Encode first-principles of domain (*physics*) into network structure

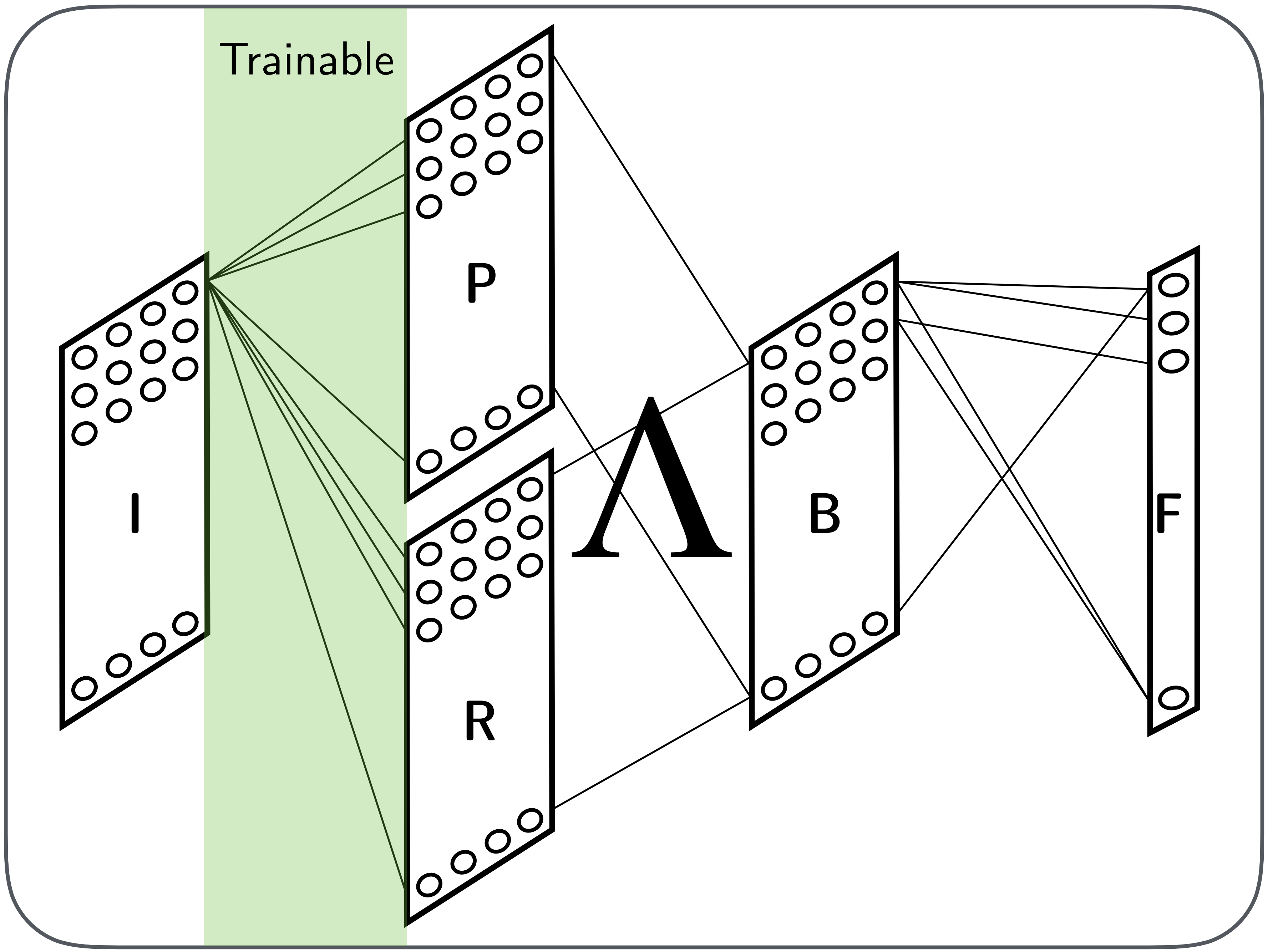
Input (4-vectors)

Combine to **P**articles & **R**est frames

Boost P into R

Feature engineering

arXiv:1812.09722 (submitted to *JINST*)



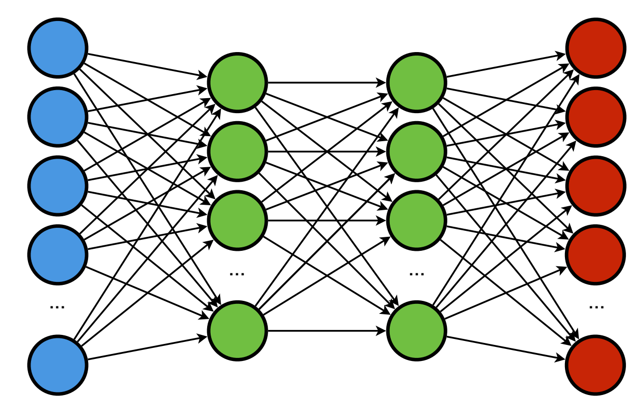
N x 4

2 · (M x 4)

M x 4

F x 1

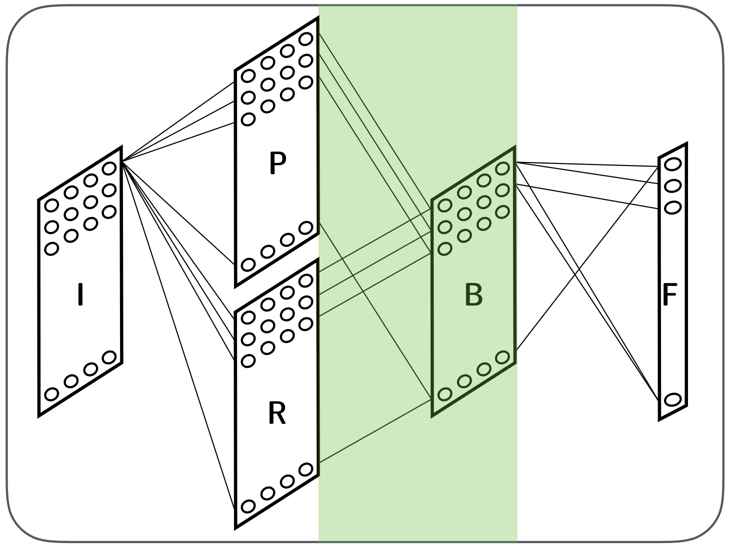
Application-specific neural network



- Lorentz transformation with boost matrix

$$\Lambda = \begin{bmatrix} \gamma & -\gamma\beta n_x & -\gamma\beta n_y & -\gamma\beta n_z \\ -\gamma\beta n_x & 1+(\gamma-1)n_x^2 & (\gamma-1)n_x n_y & (\gamma-1)n_x n_z \\ -\gamma\beta n_y & (\gamma-1)n_y n_x & 1+(\gamma-1)n_y^2 & (\gamma-1)n_y n_z \\ -\gamma\beta n_z & (\gamma-1)n_z n_x & (\gamma-1)n_z n_y & 1+(\gamma-1)n_z^2 \end{bmatrix}$$

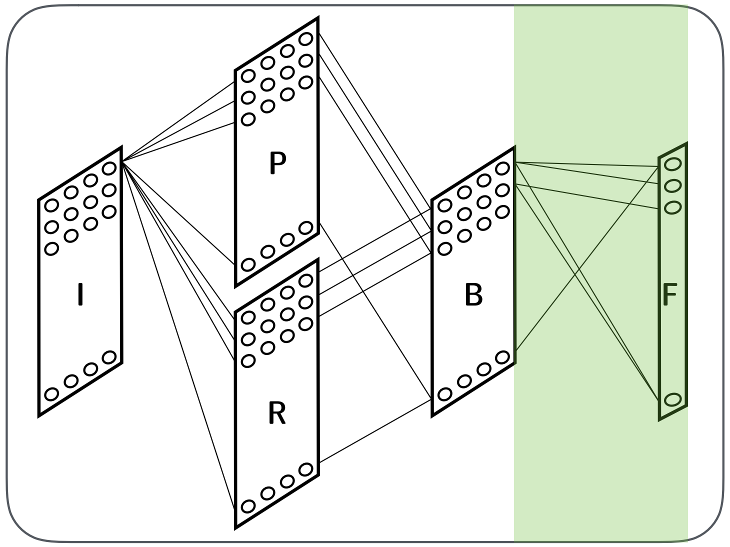
with $\vec{n} = \vec{\beta}/\beta$



- Vectorized formulation to run efficiently on GPUs
  - 4D tensor (batch x particle x 4 x 4)

$$\boxed{\Lambda = I + (U \oplus \gamma) \odot ((U \oplus 1) \cdot \beta - U) \odot (e \cdot e^T)}$$

with $U = \begin{bmatrix} -1^{1\times1} & 0^{1\times3} \\ 0^{3\times1} & -1^{3\times3} \end{bmatrix}$ $e = \begin{bmatrix} 1^{1\times1} \\ -\vec{n}^{3\times1} \end{bmatrix}$

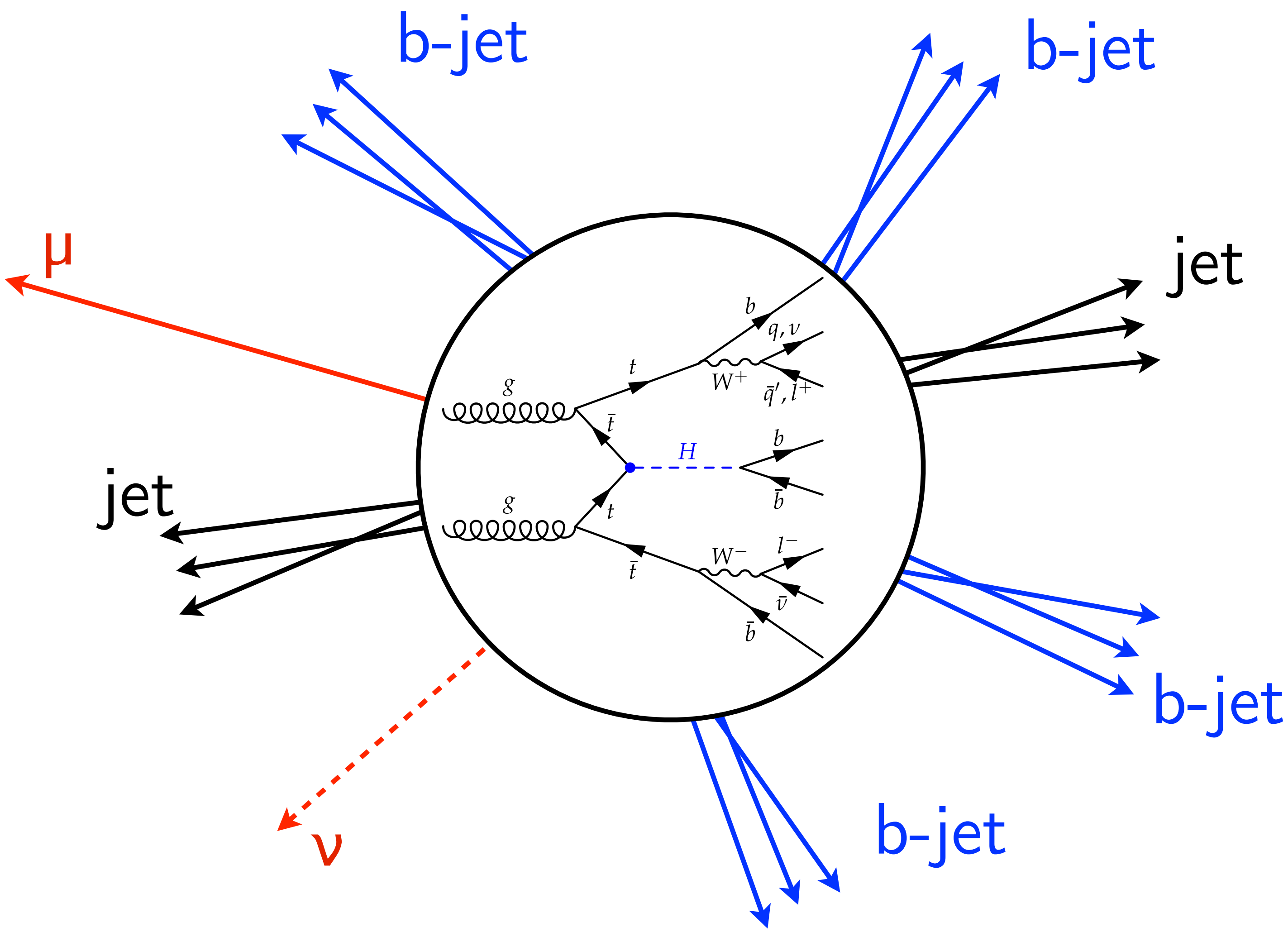


- Project features from M boosted 4-vectors:
  - **Features per vector**:
    - E, pt, eta, phi, mass
  - **Pairwise features**:
    - cos(ϕ) between vectors
  - More features possible, but not necessarily required

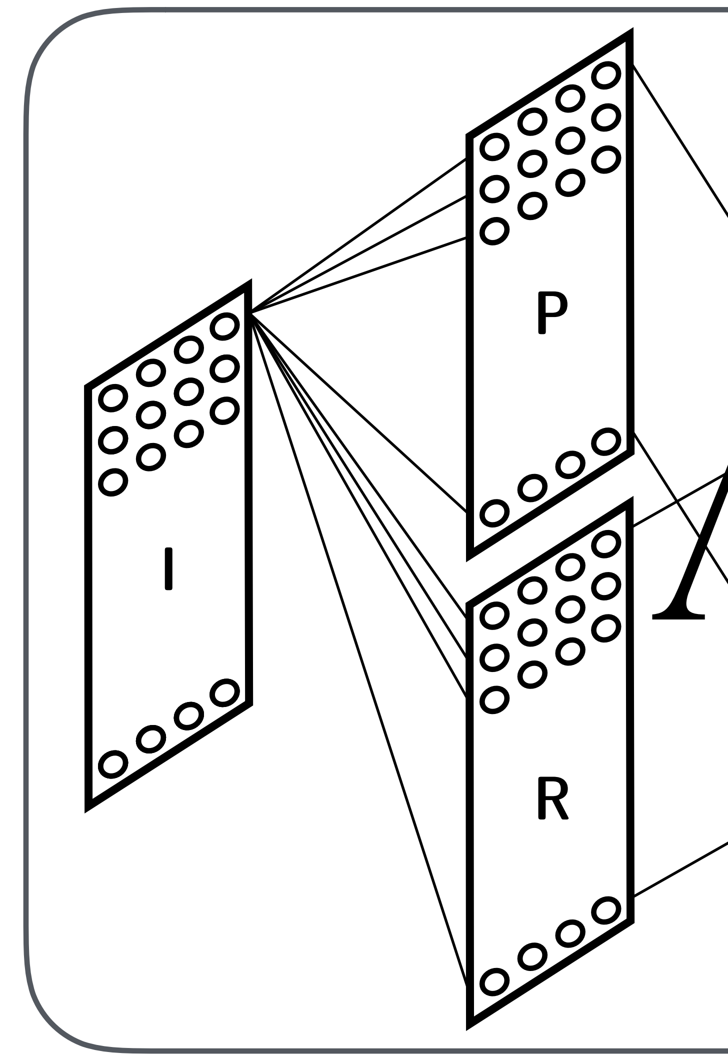
- Input feature scaling / normalization not applicable

  → Batch normalization applied after feature layer

Application
ttH vs. tt+bb



How to sort jets?



Option 1

Use generator (truth) information

→ Isolated study of LBN performance

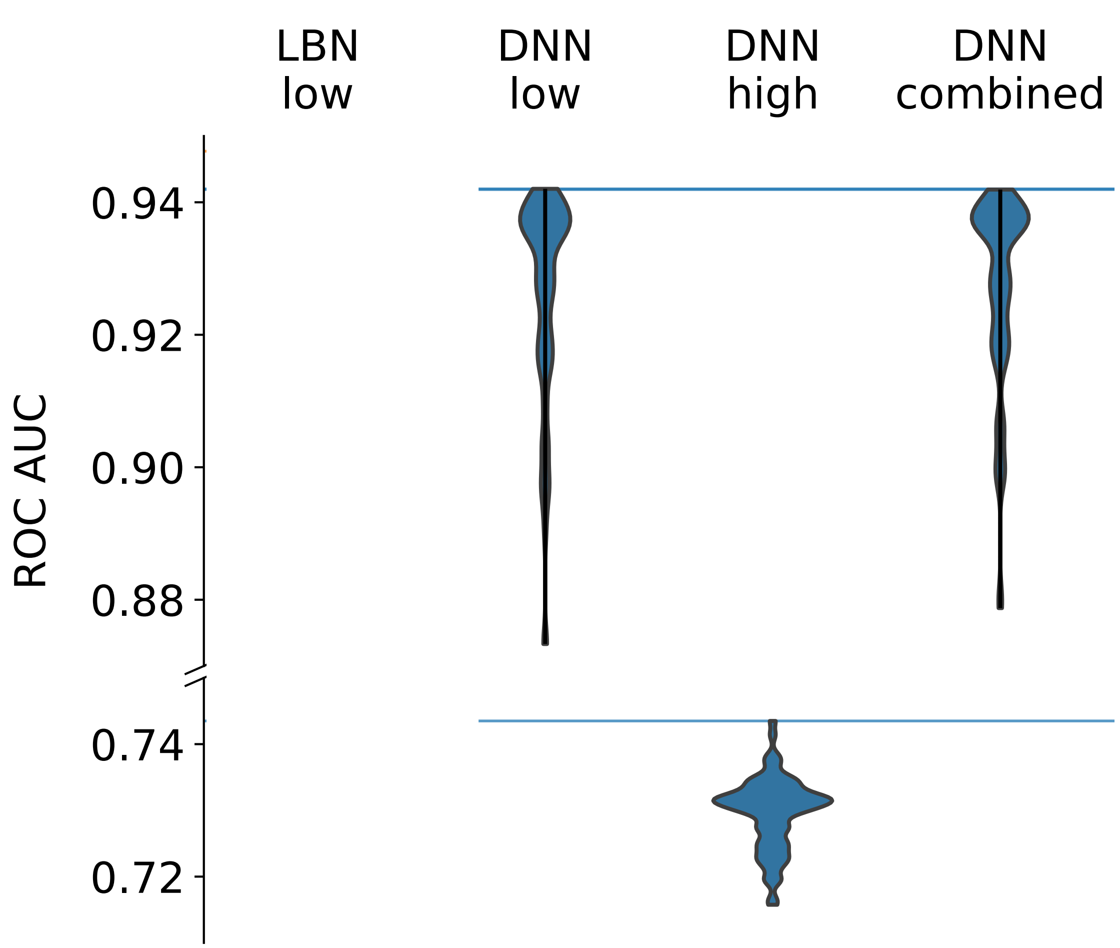
Option 2

Sort jets by transverse momentum p_T

→ Challenging "real-life" scenario

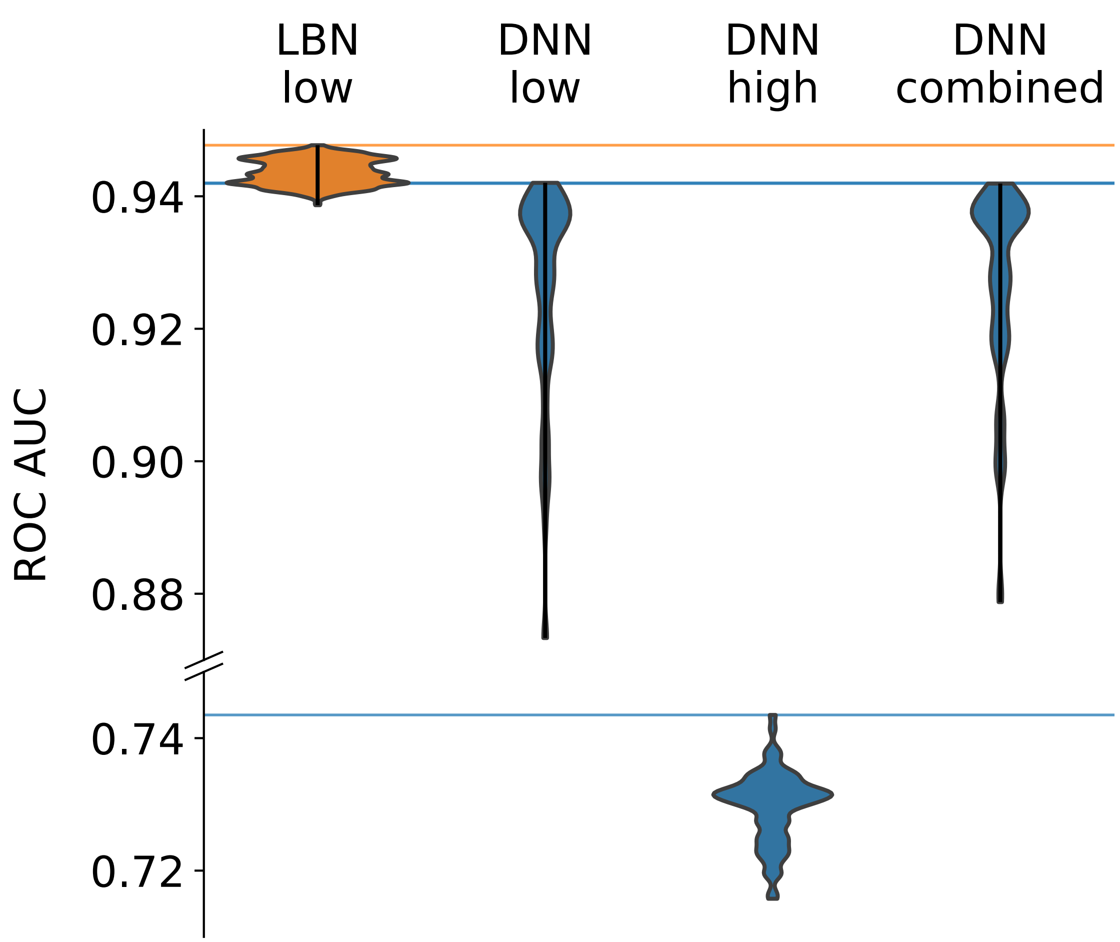
- O(100) networks per configuration
- LBN: only few hyper-parameters, essentially number of combinations M
- DNN: varied layers, units, activations, FCN/Dense/Residual setup, learning rate, L2 norm.

Generator (truth) sorting



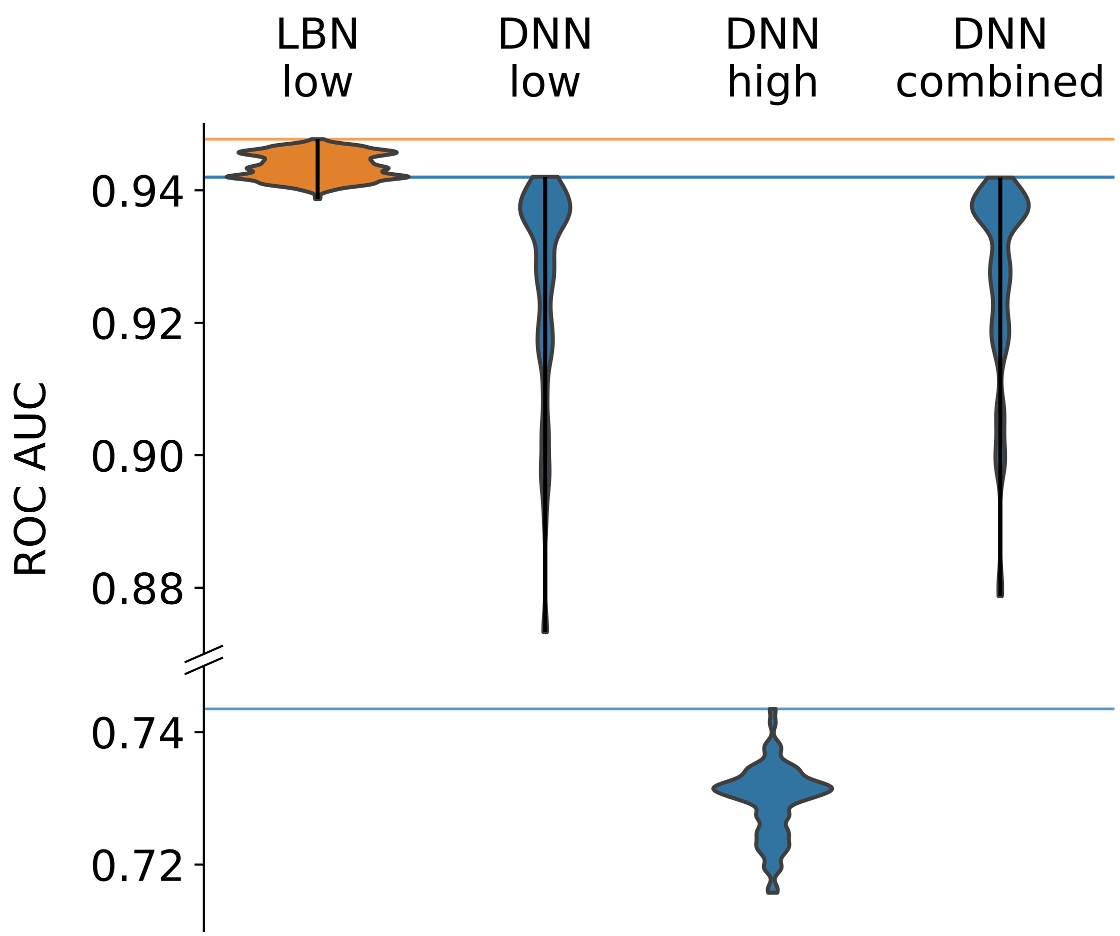
- $\mathcal{O}(100)$ networks per configuration
- LBN: only few hyper-parameters, essentially number of combinations M
- DNN: varied layers, units, activations, FCN/Dense/Residual setup, learning rate, L2 norm.

Generator (truth) sorting

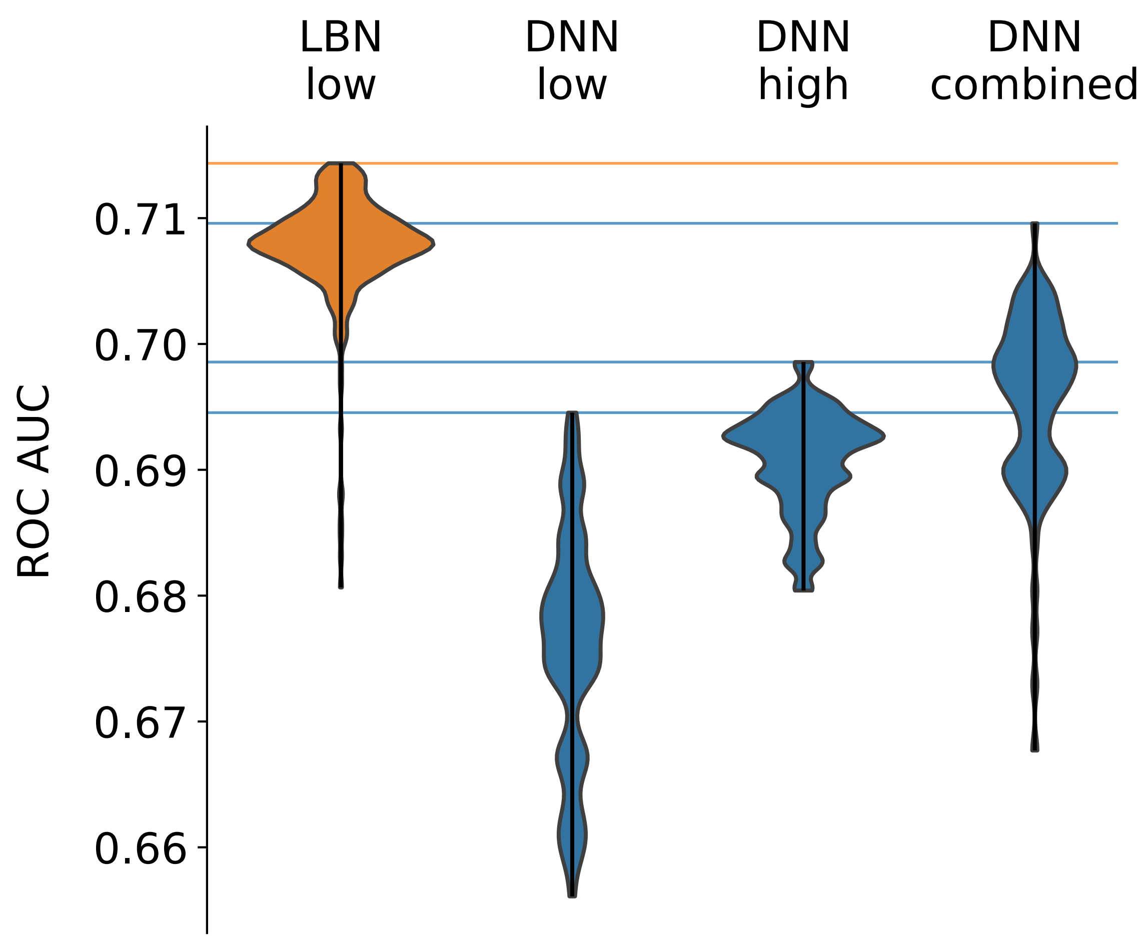


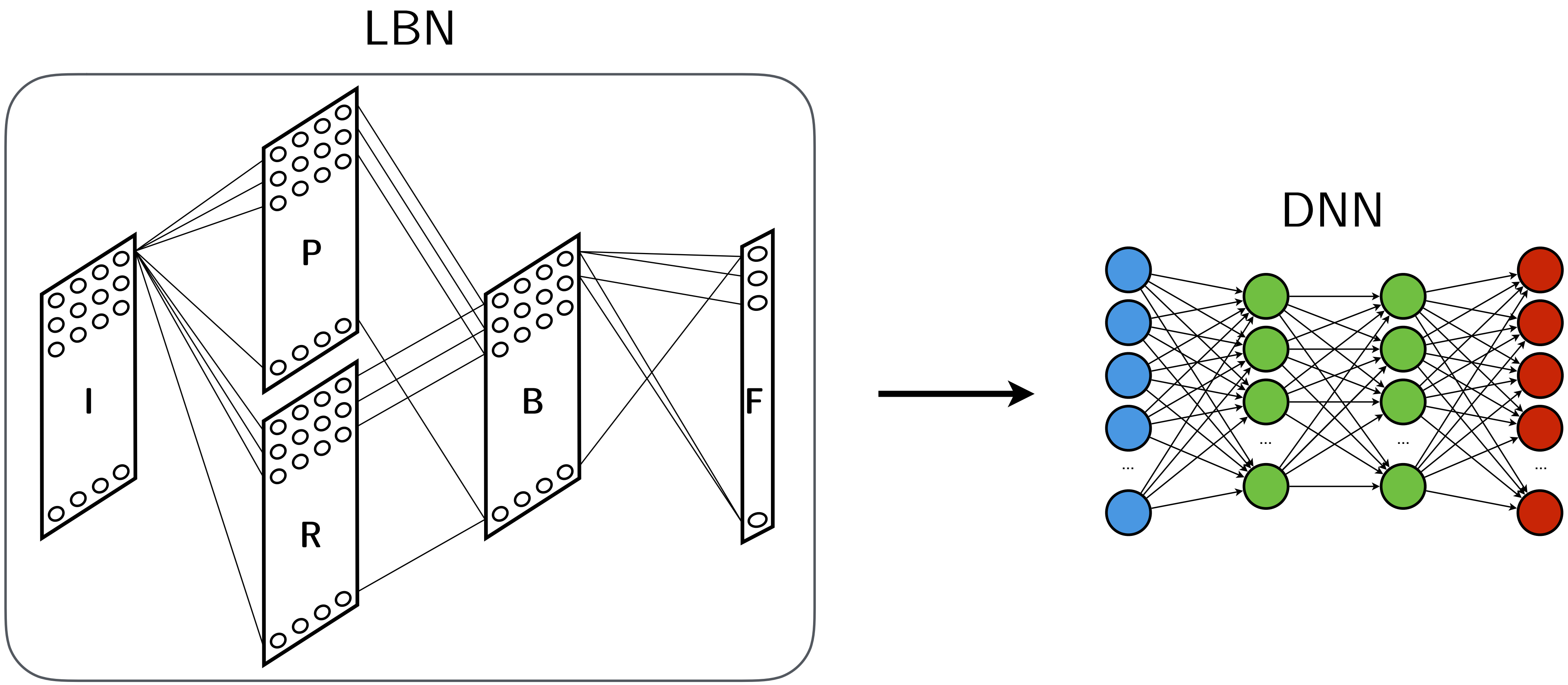
- O(100) networks per configuration
- LBN: only few hyper-parameters, essentially number of combinations M
- DNN: varied layers, units, activations, FCN/Dense/Residual setup, learning rate, L2 norm.

Generator (truth) sorting



p_T sorting





1. DNN behind LBN can be **rather shallow**
   - Feature representation moved to LBN
   - DNN can focus on transformation to output
2. LBN is **not a black box**
   - Extract which (combined) particles and features are important

Combination coefficients (generator sorting)

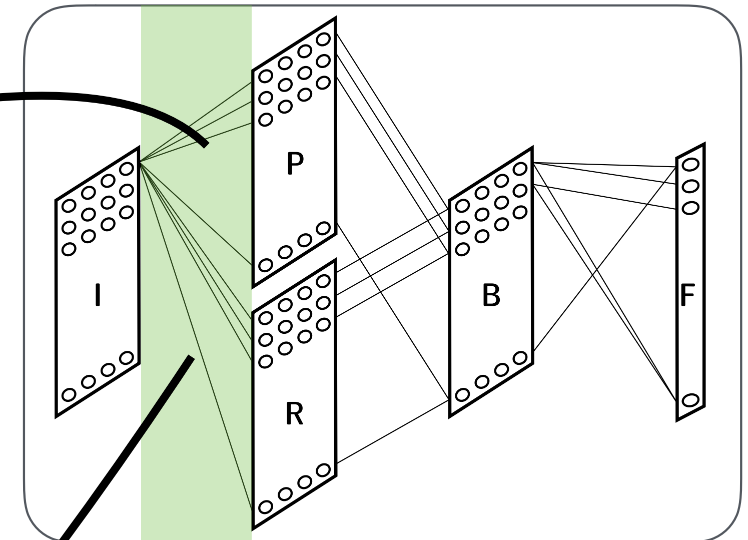
Combined Particle

Input Particle	0	1	2	3	4	5	6	7	8	9	10	11	12	
b_1	39	0	8	51	7	90	0	0	40	0	10	51	47	H^1
b_2	61	0	1	21	8	0	32	0	3	0	84	18	16	
b_{had}	0	68	14	0	22	7	12	1	17	11	1	2	0	
q_1	0	17	13	0	27	0	9	1	22	12	1	12	1	t_{had}
q_2	0	15	16	2	27	2	11	0	10	21	1	1	9	
b_{lep}	0	0	13	3	2	0	11	68	0	18	1	13	4	
lep	0	0	16	8	0	0	14	25	8	38	1	3	13	t_{lep}
ν	0	0	19	15	5	0	10	6	0	0	1	1	9	

...

Combined Restframe

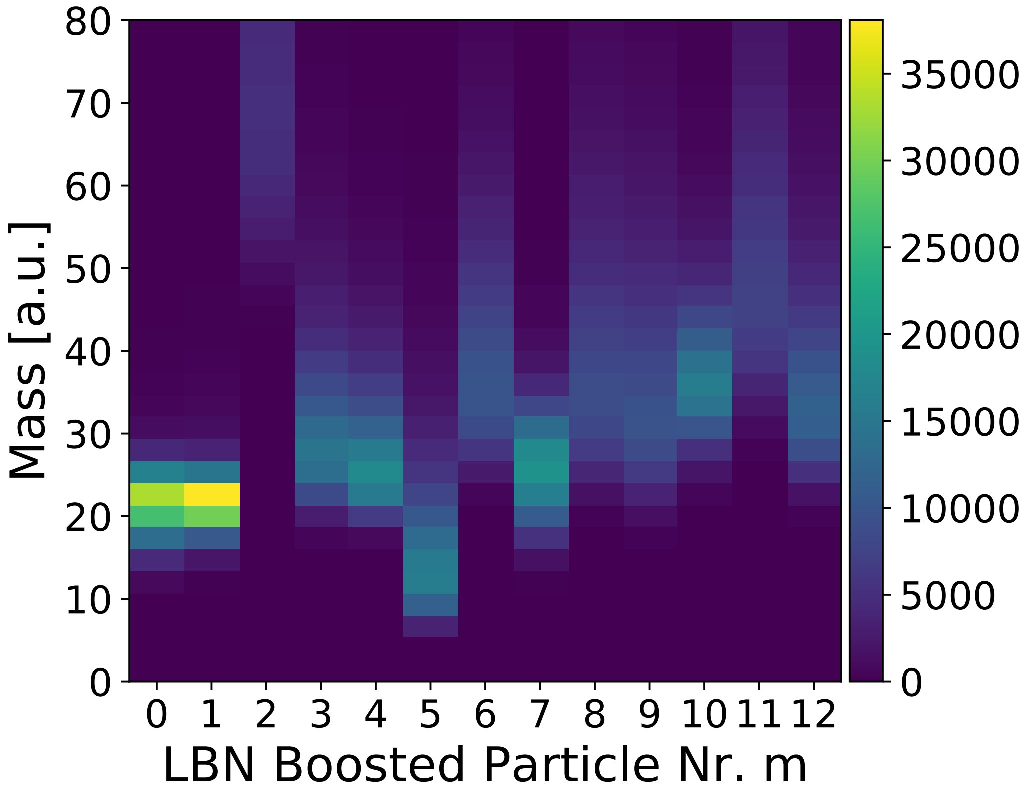
Input Particle	0	1	2	3	4	5	6	7	8	9	10	11	12	
b_1	7	5	0	7	11	15	99	5	14	25	47	17	5	H^1
b_2	6	33	100	5	8	32	0	56	45	0	8	58	41	
b_{had}	11	30	0	0	7	15	0	4	10	3	8	2	5	
q_1	36	15	0	0	10	8	0	3	11	5	6	12	5	t_{had}
q_2	24	13	0	3	9	6	0	2	5	9	8	1	17	
b_{lep}	2	1	0	30	19	7	0	7	2	11	8	9	6	
lep	4	1	0	29	21	12	0	13	8	11	9	1	7	t_{lep}
ν	9	3	0	26	16	4	0	9	3	35	8	0	14	



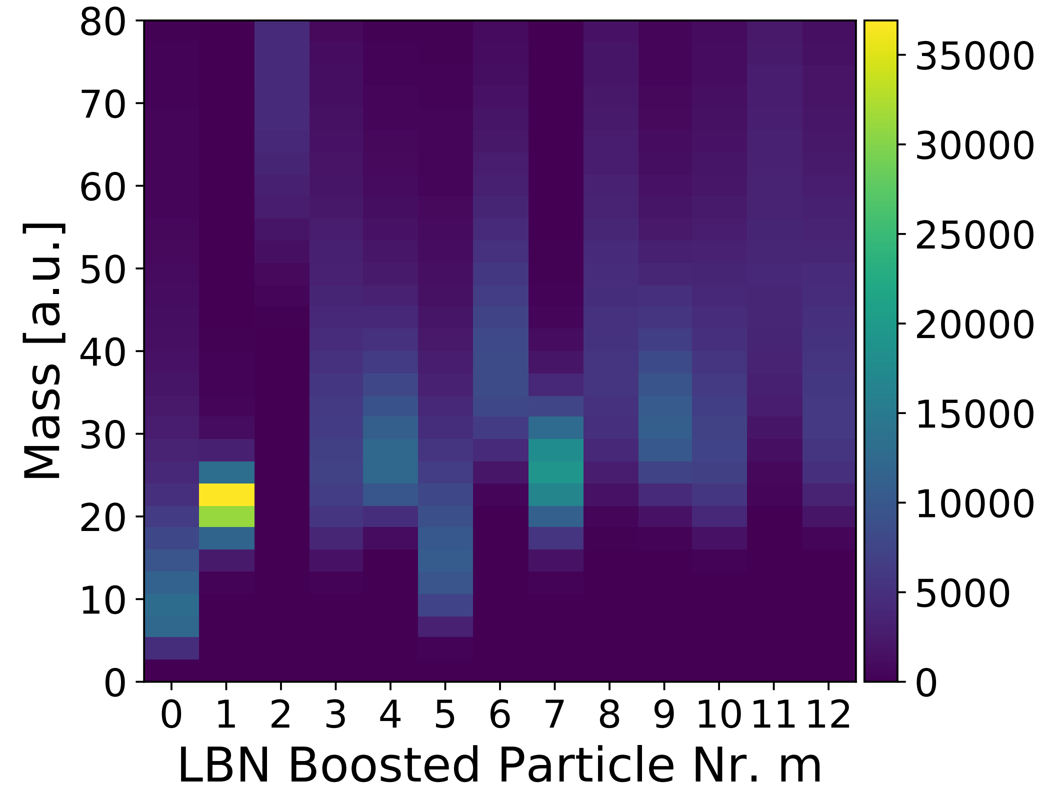
LBN finds particles & rest frames that lead to features which are important to solve task!

Feature example: particle mass

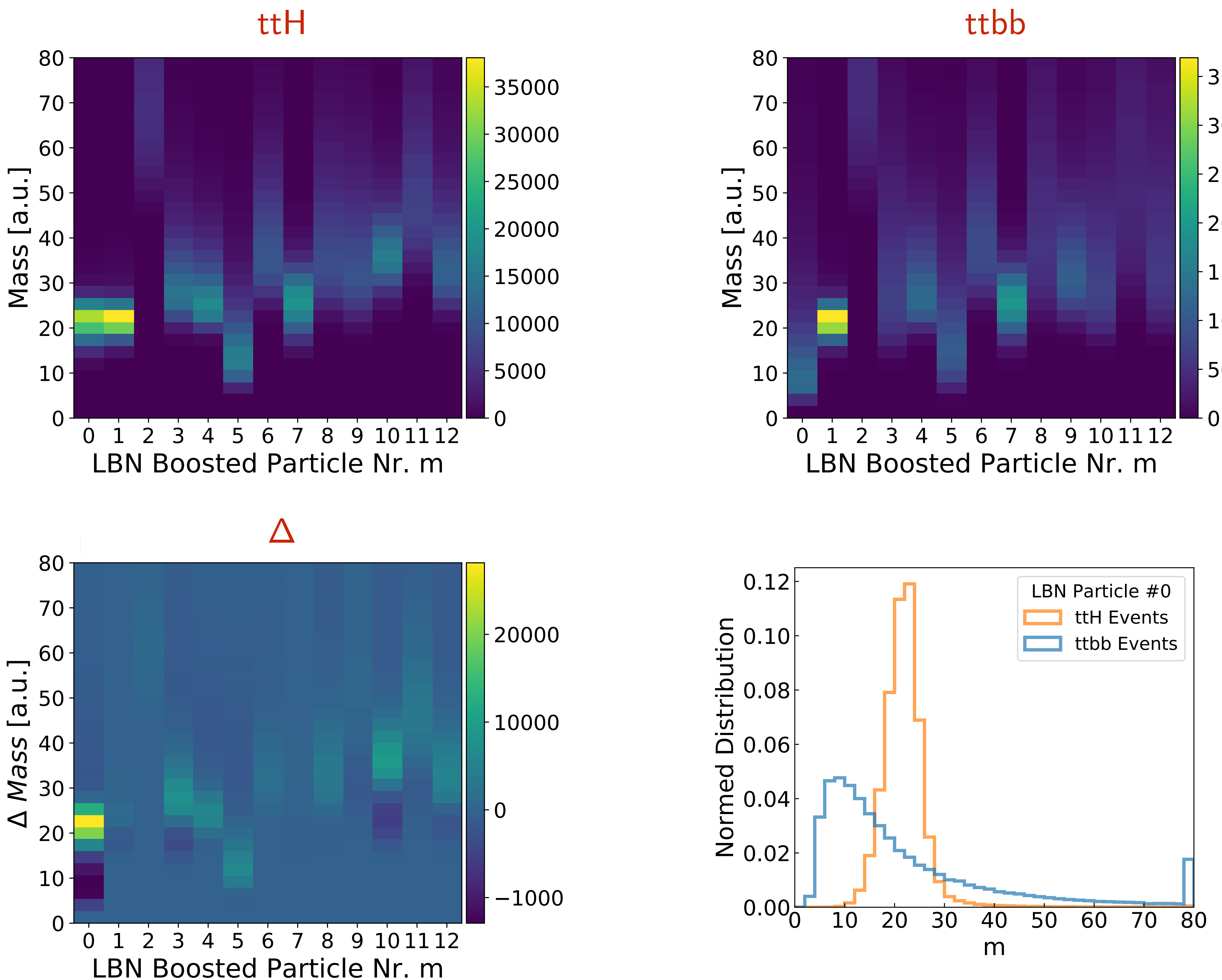
ttH



ttbb



Feature example: particle mass

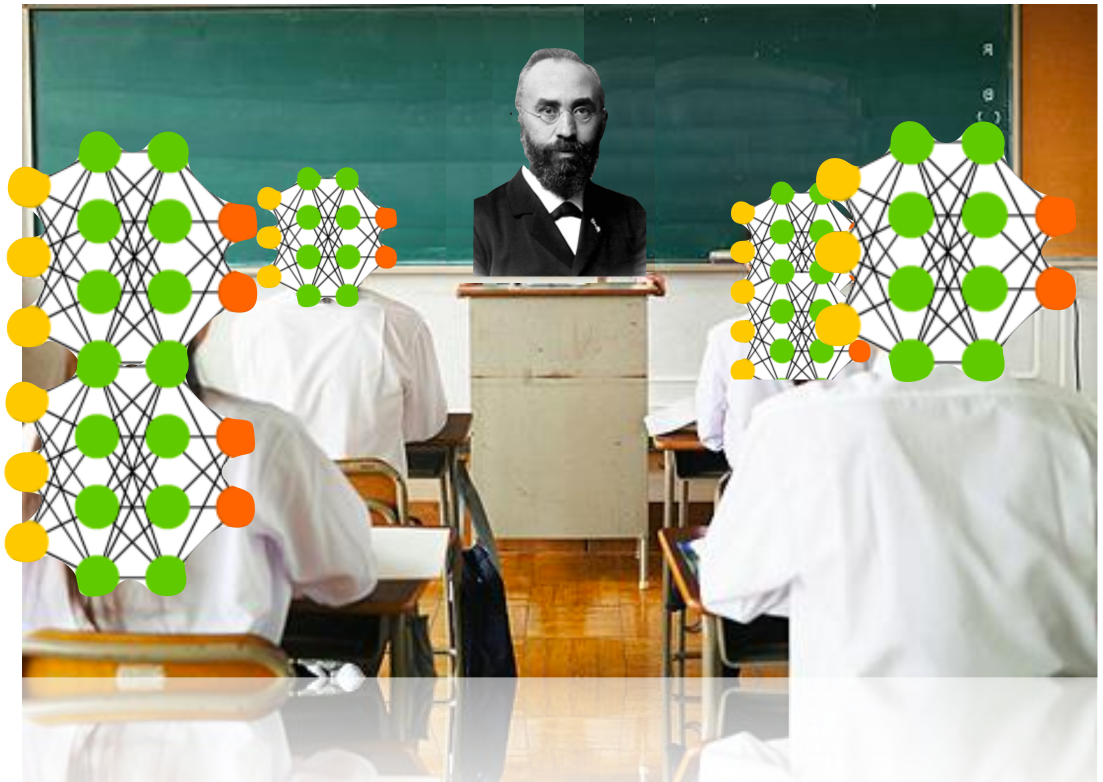


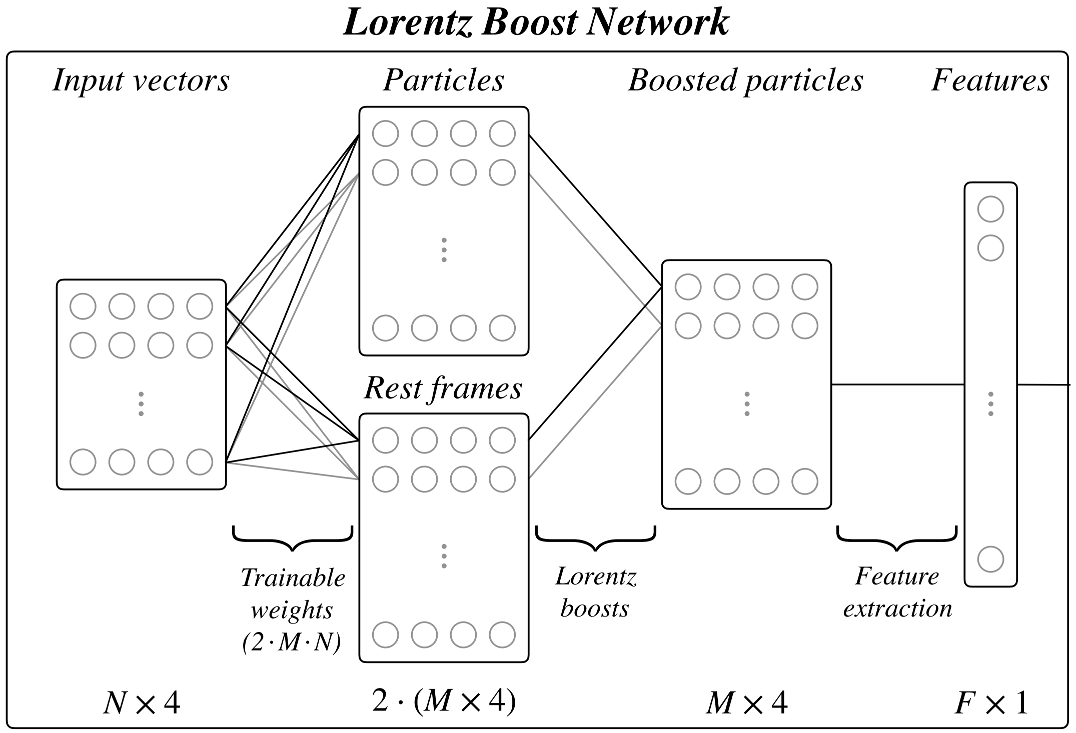
- *"Which additional feature could increase my network performance?"*

  should rather become

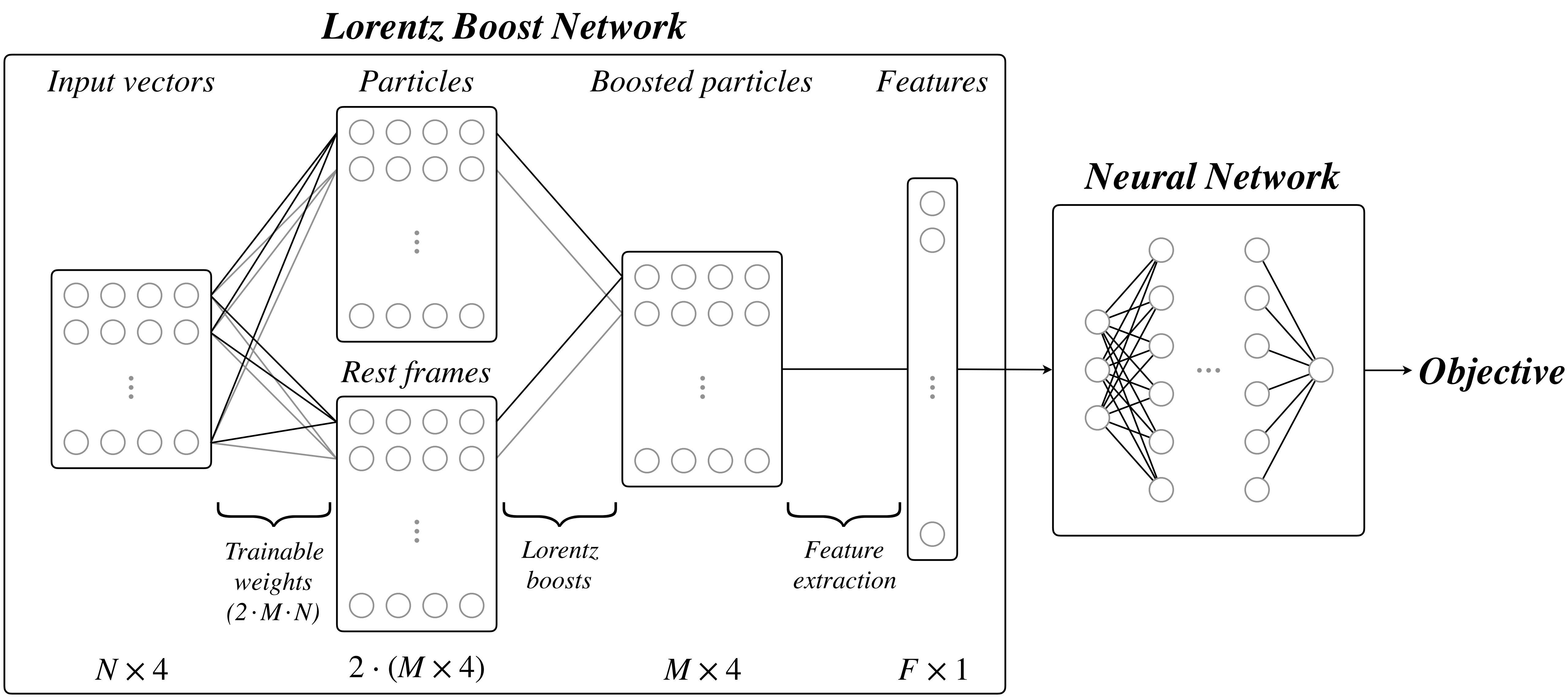
  "How can I design my network to (even better) work with raw features?"
  - ▹ Encode physics knowledge right into network **rather than** into features

- Lorentz Boost Network possible candidate for many use cases
  - ▹ Architecture able to autonomously engineer features
  - ▹ Allows to open the **black box:** LBN finds meaningful relations in physics context
  - ▹ git.rwth-aachen.de/3pia/lbn

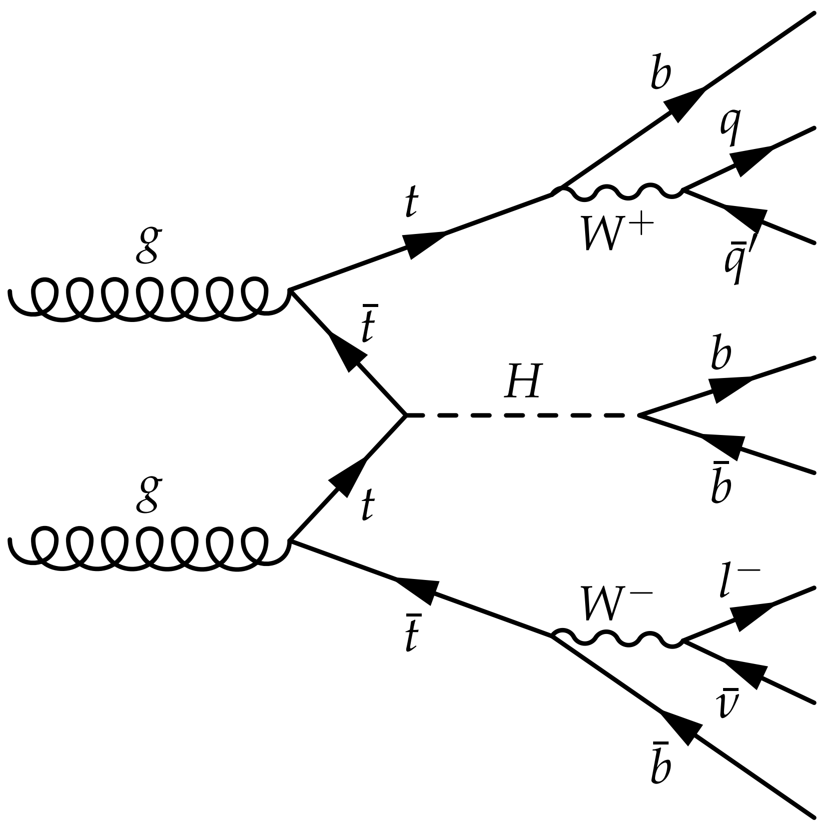


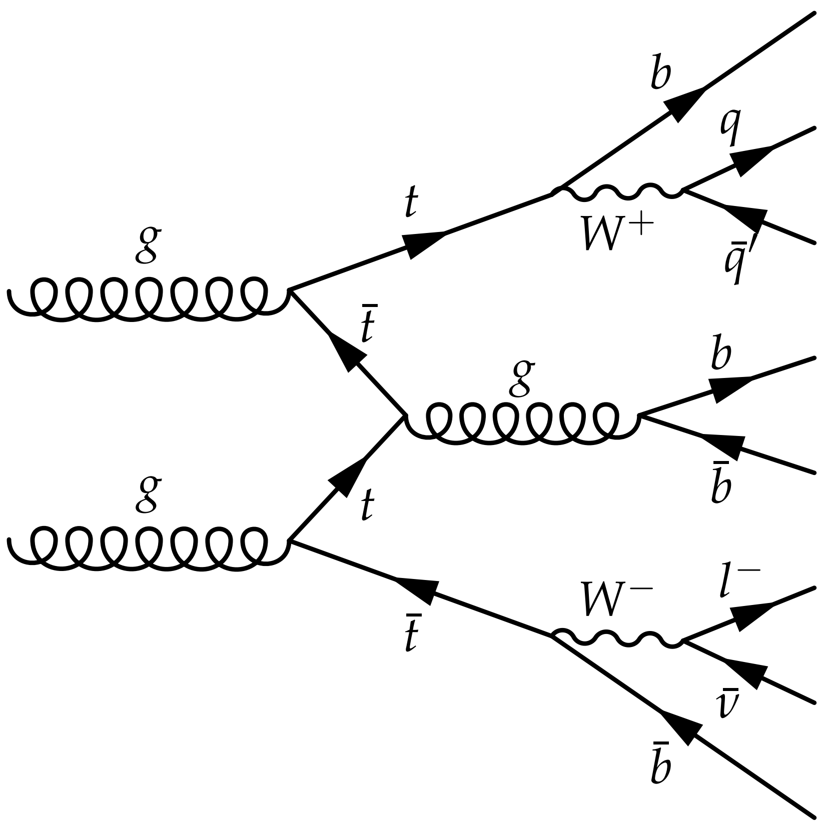


Backup


Lorentz Boost Network
Input vectors
Particles
Boosted particles
Features
Rest frames
Trainable weights $(2 \cdot M \cdot N)$
Lorentz boosts
Feature extraction
$N \times 4$
$2 \cdot (M \times 4)$
$M \times 4$
$F \times 1$
Neural Network
Objective

Classification: ttH (H → bb) vs. ttbb

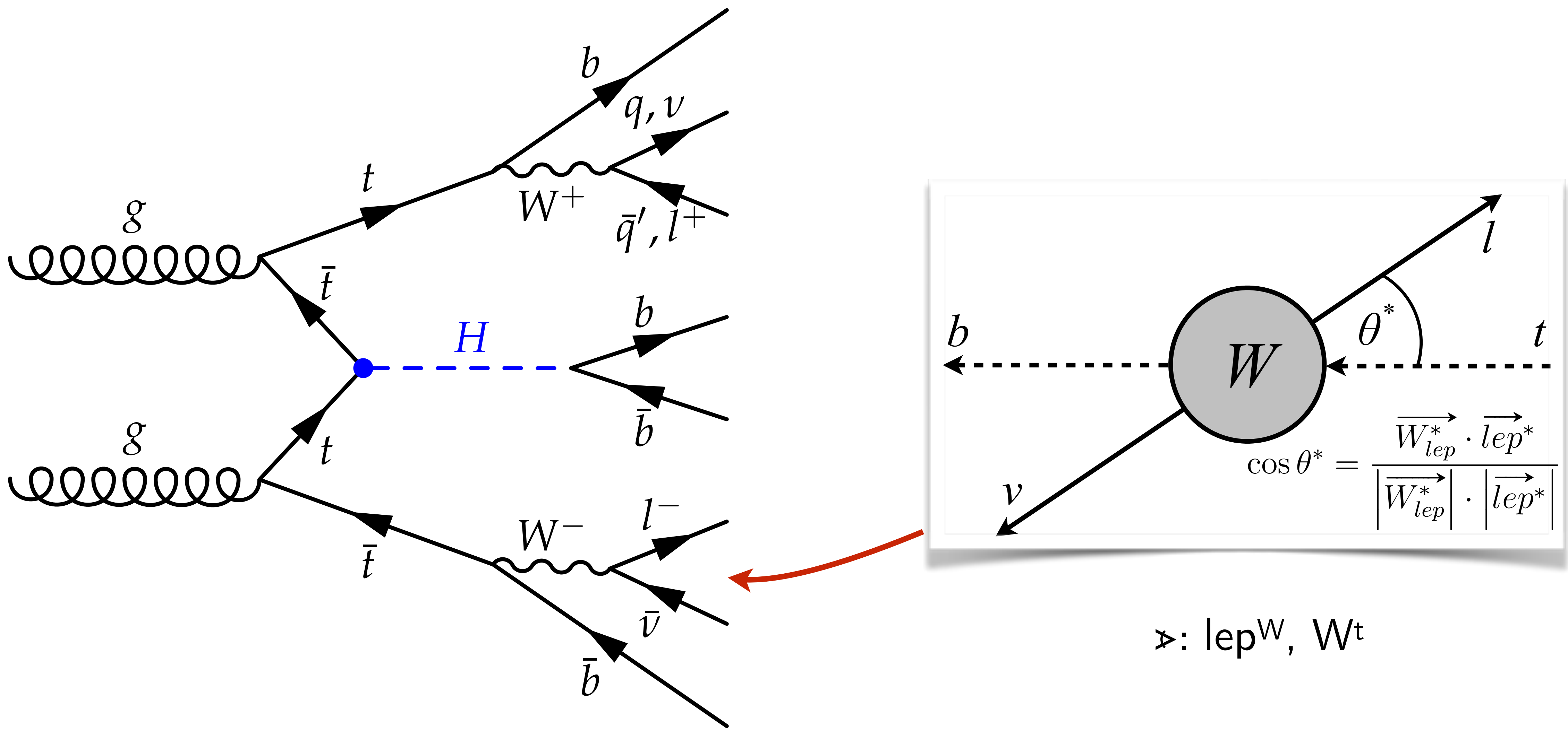




- Final state: 6 jets, 1 charged lepton, 1 neutrino (missing transverse energy)

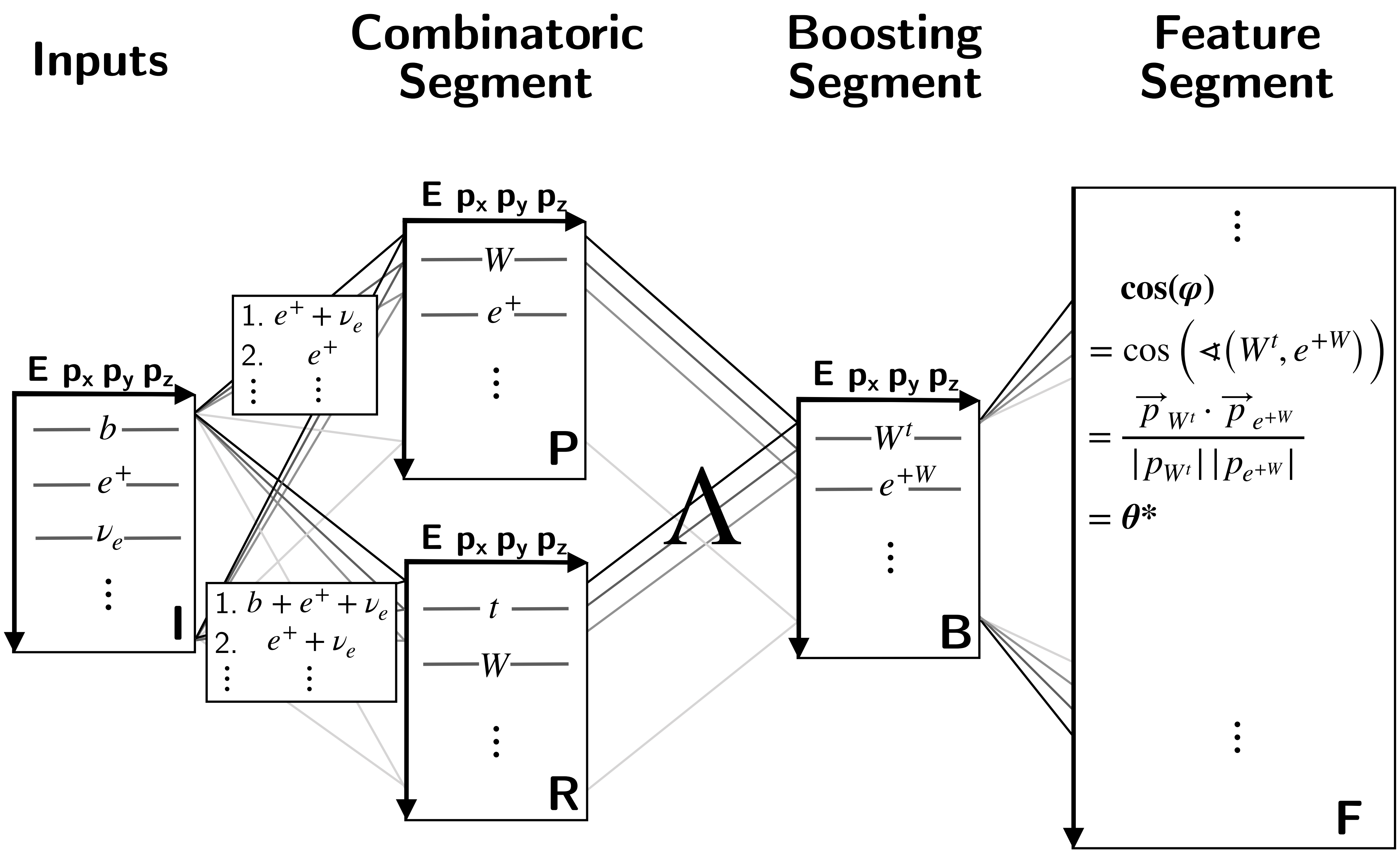
- Variables
  - Low-level: 6 + 2 four-momenta
  - High-level: 26, from published ttH analysis
  - Combined

- How to order jets?
  a) Using truth information
  b) Simple sorting by p_T



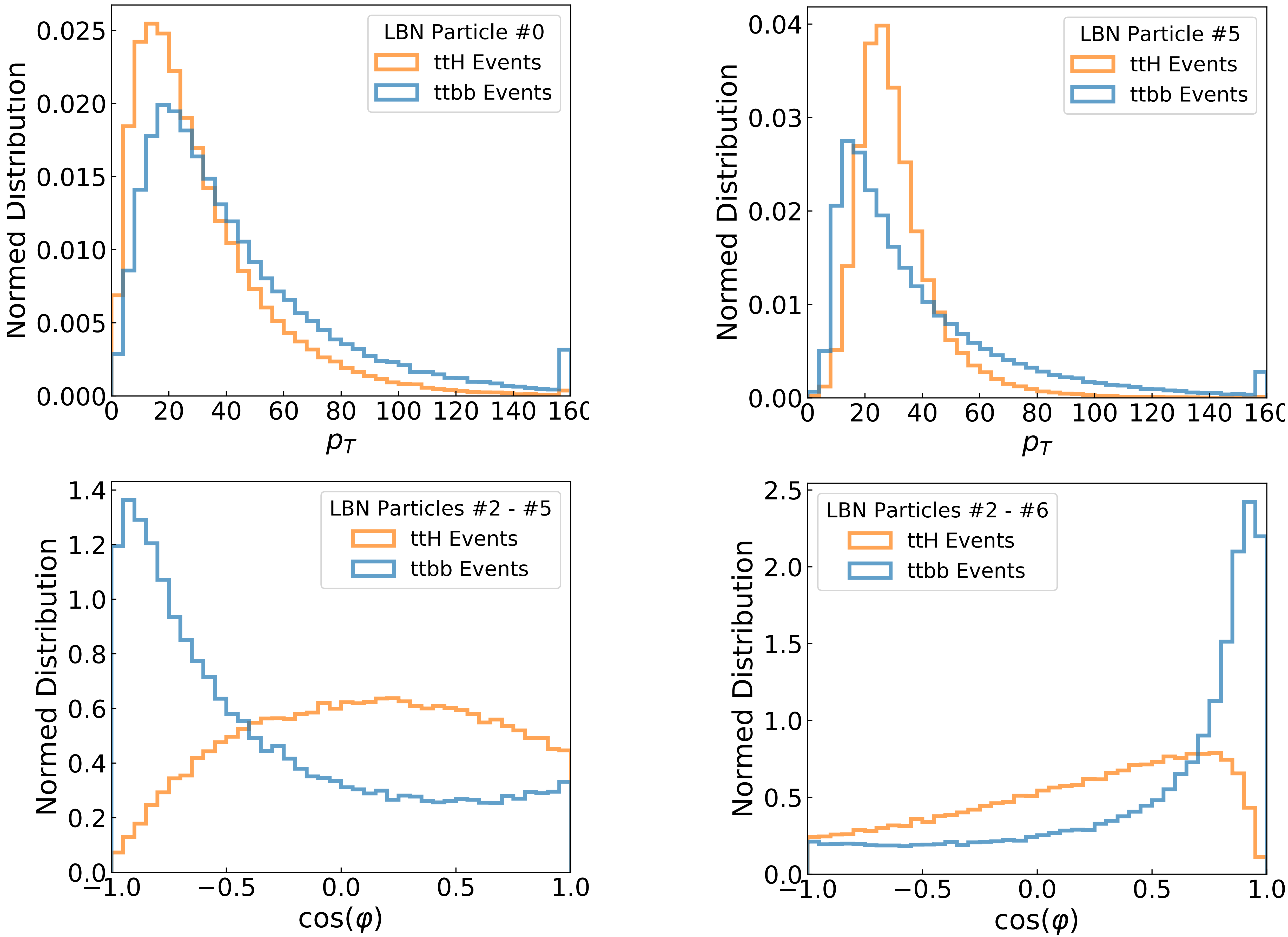
$\sphericalangle$: lep^{W}, W^{t}

RWTHAACHEN UNIVERSITY
Marcel Rieger - 19.2.19



Network	LBN+NN		DNN					
Observables	**low-level**		**low-level**		**high-level**		**combined**	
Input Ordering	**gen.**	$\mathbf{p_T}$	**gen.**	$\mathbf{p_T}$	**gen.**	$\mathbf{p_T}$	**gen.**	$\mathbf{p_T}$
$M_{\text{part.,restfr.}}$	13	16	–	–	–	–	–	–
LBN-gate	on	off	–	–	–	–	–	–
n_{layers}	8	8	8	8	4	4	8	6
n_{nodes}	1024	1024	1024	1024	512	512	1024	1024

Engineered features



Coefficients correlations (generator ordering)

Combined to Particles

	b_1	b_2	b_{had}	q_1	q_2	b_{lep}	lep	ν
b_1	100	-4	-37	-36	-52	-38	-43	-14
b_2	-4	100	-37	-43	-44	-29	-37	-14
b_{had}	-37	-37	100	58	51	-24	-22	-19
q_1	-36	-43	58	100	78	-21	-7	-11
q_2	-52	-44	51	78	100	-16	27	11
b_{lep}	-38	-29	-24	-21	-16	100	61	12
lep	-43	-37	-22	-7	27	61	100	22
ν	-14	-14	-19	-11	11	12	22	100

Groups: H^1 (b_1, b_2), t_{had} (b_{had}, q_1, q_2), t_{lep} (b_{lep}, lep, ν)

Combined to Restframes

	b_1	b_2	b_{had}	q_1	q_2	b_{lep}	lep	ν
b_1	100	-46	-26	-27	-30	-21	-26	-18
b_2	-46	100	-7	-20	-36	-39	-40	-52
b_{had}	-26	-7	100	49	49	-32	-22	-28
q_1	-27	-20	49	100	75	-27	-25	-14
q_2	-30	-36	49	75	100	-15	-8	23
b_{lep}	-21	-39	-32	-27	-15	100	91	69
lep	-26	-40	-22	-25	-8	91	100	68
ν	-18	-52	-28	-14	23	69	68	100

Groups: H^1 (b_1, b_2), t_{had} (b_{had}, q_1, q_2), t_{lep} (b_{lep}, lep, ν)