Lorentz Group Equivariant Networks for High Energy Physics

4/3/2021

Aditya Mishra & Farouk Mokhtar (& Zichun Hao)

Lorentz Group Equivariant Networks (LGN)

IDEA:

- Make a ML model which respects symmetries in physics
- **Practically**: one event observed from different frames should be classified the same way

  i.e. model should be invariant to lorentz boosts/rotations of events

Why pick this project for this course:

- At the heart of intersection between data science and physics

HEP preliminaries

-Farouk

Intro to HEP & the LHC

- **H**igh **E**nergy **P**hysics = Particle Physics
- Energies of order 10TeV ~ 10^{-6} J

Questions we're trying to answer:

• **Spectroscopy**:

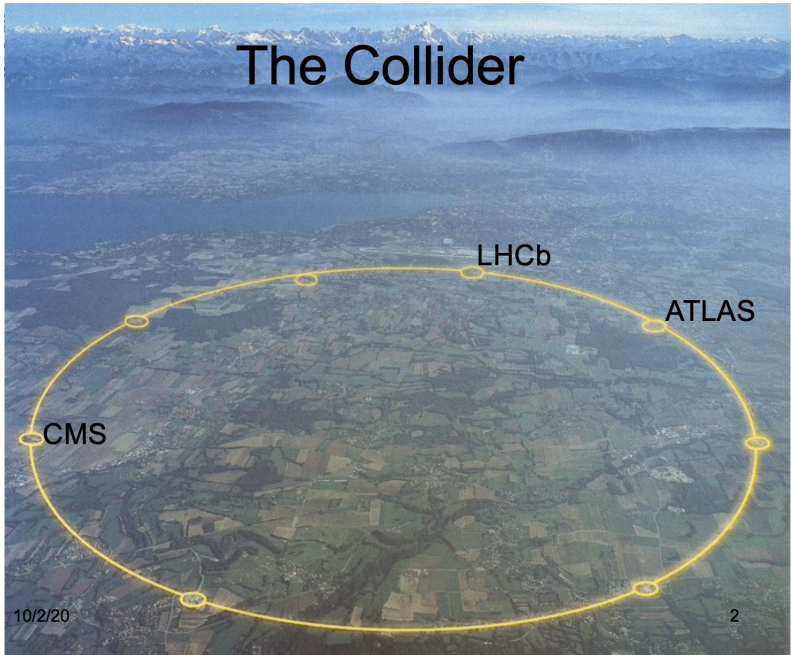
– What are the particles that exist ? – What are their properties ?

• **Dynamics**:

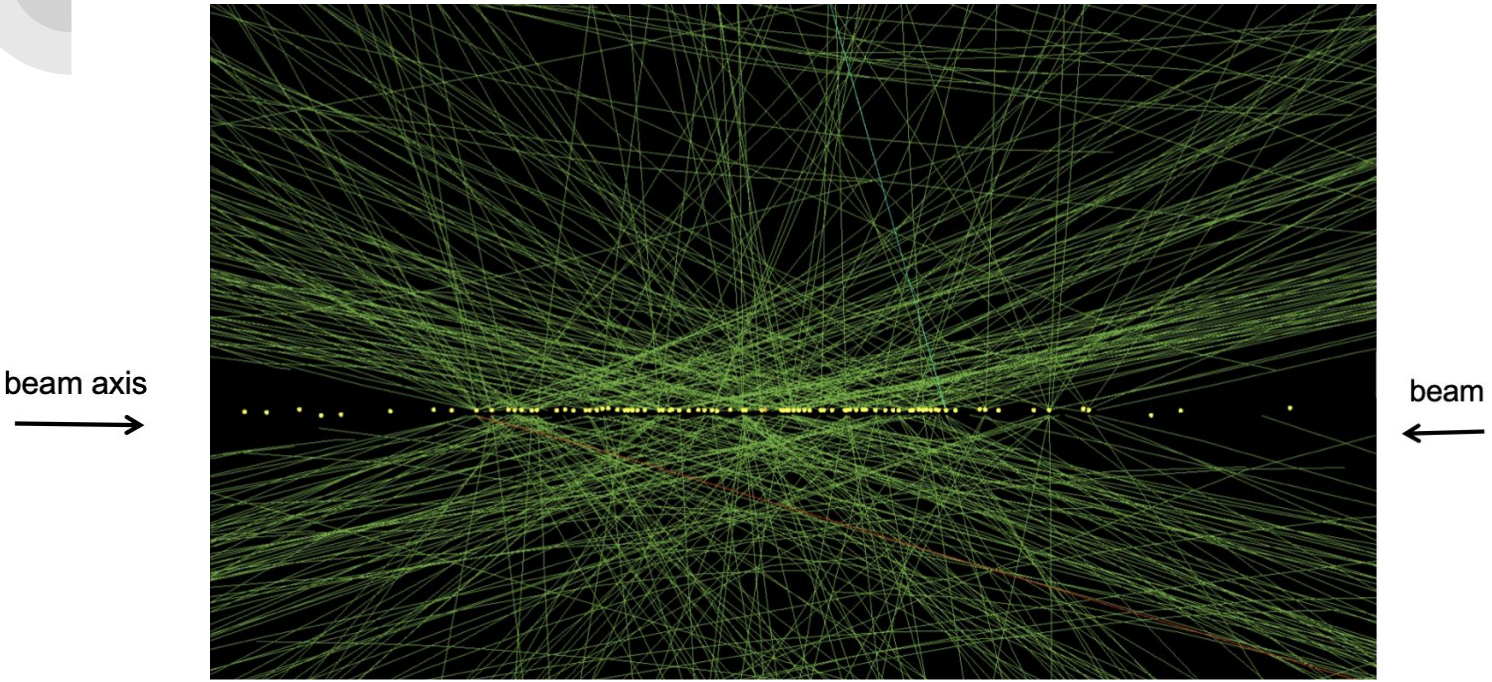
– What are the forces ? – How do the particles couple to the forces ?

The Large Hadron Collider (LHC)

- 27 km in circumference
- Colliding protons on protons at energies of 13TeV
- 2808 bunches colliding every 25ns with 115 billion protons per bunch
- Every bunch crossing is called an **event**



Zoomed-in event



- Green lines indicate reconstructed charged particle trajectories.
- Yellow dots indicate reconstructed collisions

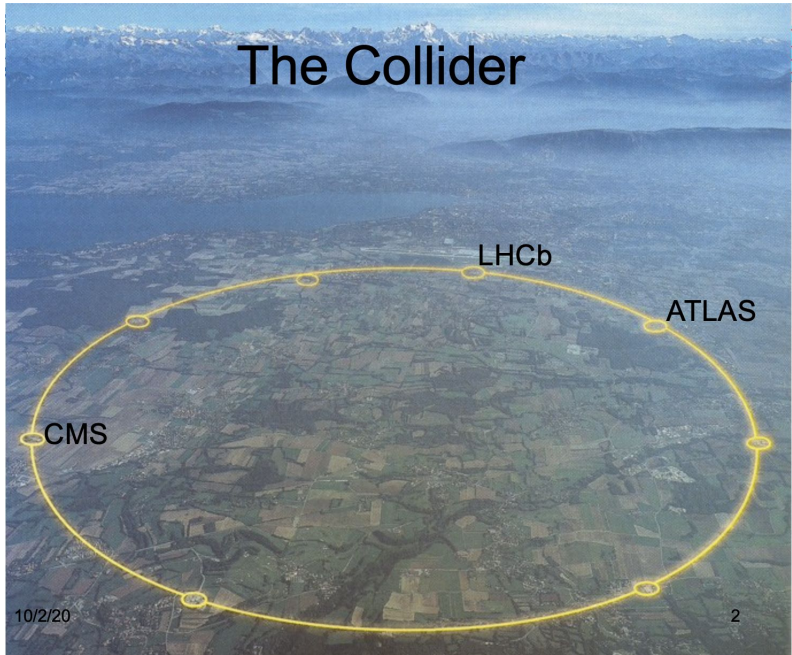
“Big bang” in the laboratory

- We gain insight by colliding protons at the highest energies possible to measure:
  - Production rates
  - Masses & lifetimes
  - Decay rates
- Aim towards higher energies and more pp collisions per beam crossing:

(1) **More energy** => probing higher masses, smaller distances & earlier times

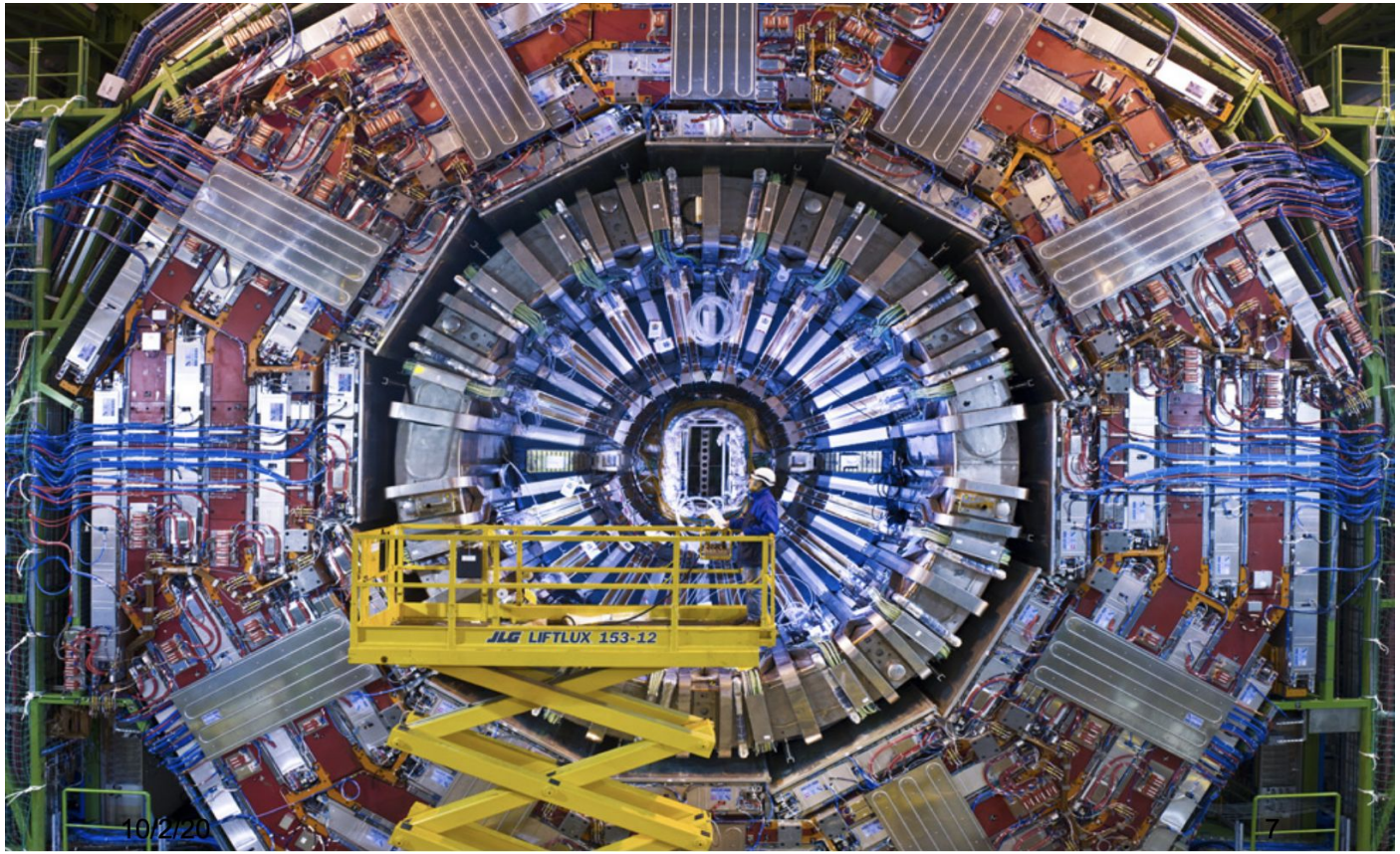
(2) **More collisions** => increased sensitivity to rare events

40 million collisions / second
Higgs boson produced 1/10 billion collisions (every 4 minutes)



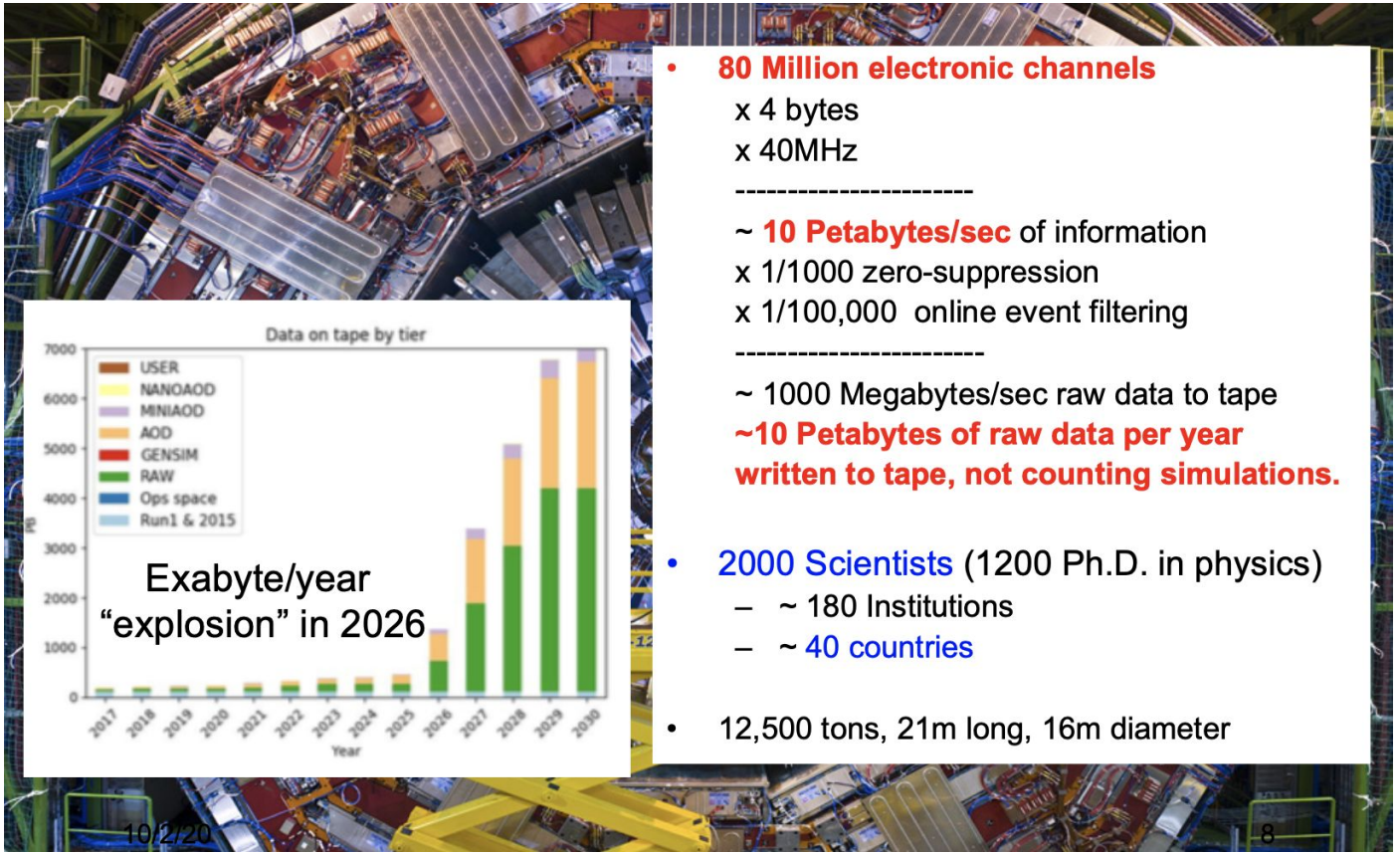
The CMS experiment

B=4 Tesla.. that is 100,000 times stronger than the Earth



The CMS experiment

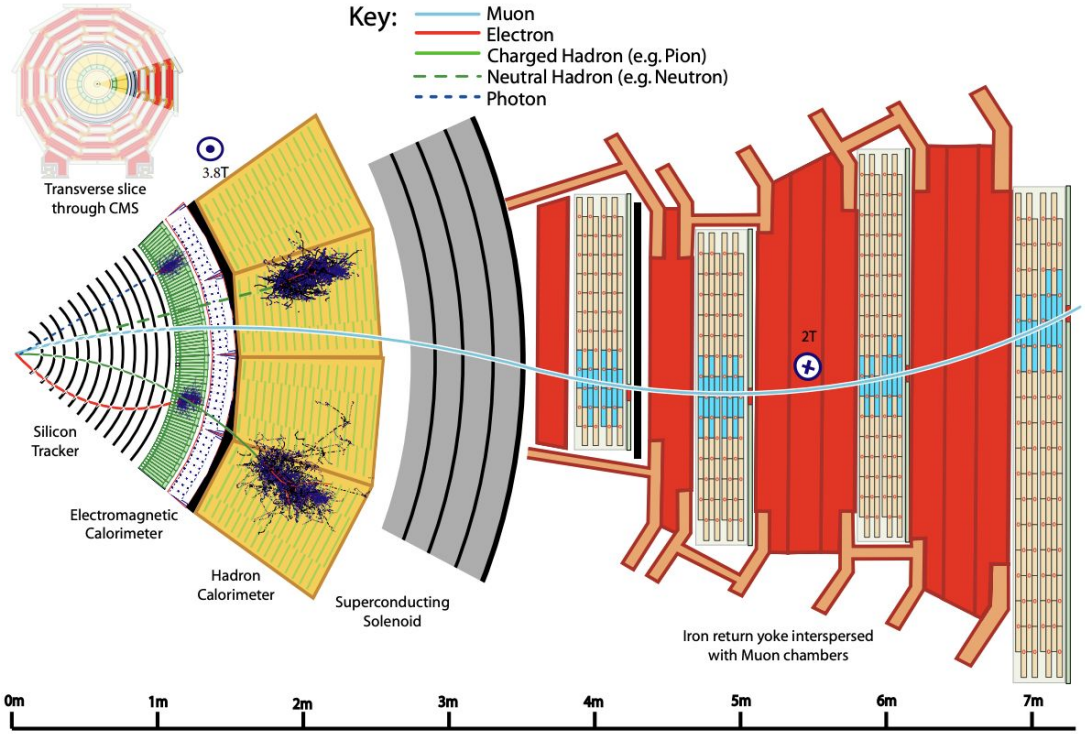
B=4 Tesla.. that is 100,000 times stronger than the Earth



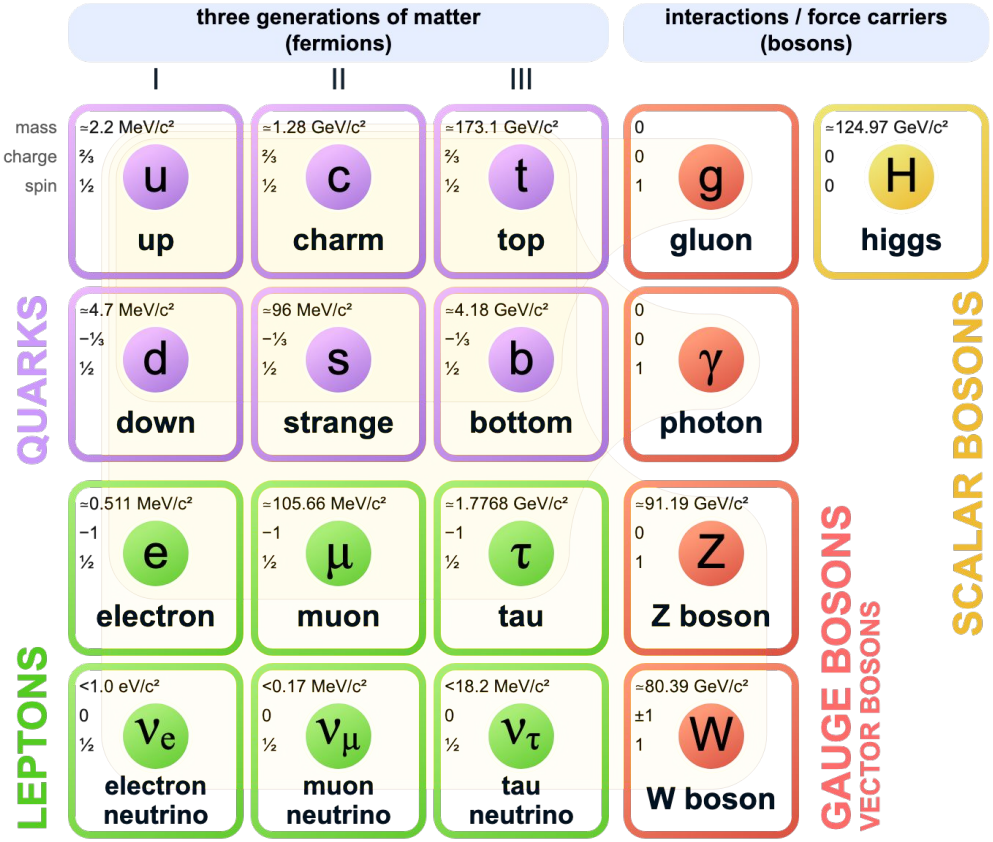
- **80 Million electronic channels**
  - x 4 bytes
  - x 40MHz
  - ---
  - ~ **10 Petabytes/sec** of information
  - x 1/1000 zero-suppression
  - x 1/100,000 online event filtering
  - ---
  - ~ 1000 Megabytes/sec raw data to tape
  - **~10 Petabytes of raw data per year written to tape, not counting simulations.**
- 2000 Scientists (1200 Ph.D. in physics)
  - ~ 180 Institutions
  - ~ 40 countries
- 12,500 tons, 21m long, 16m diameter

Particle Flow

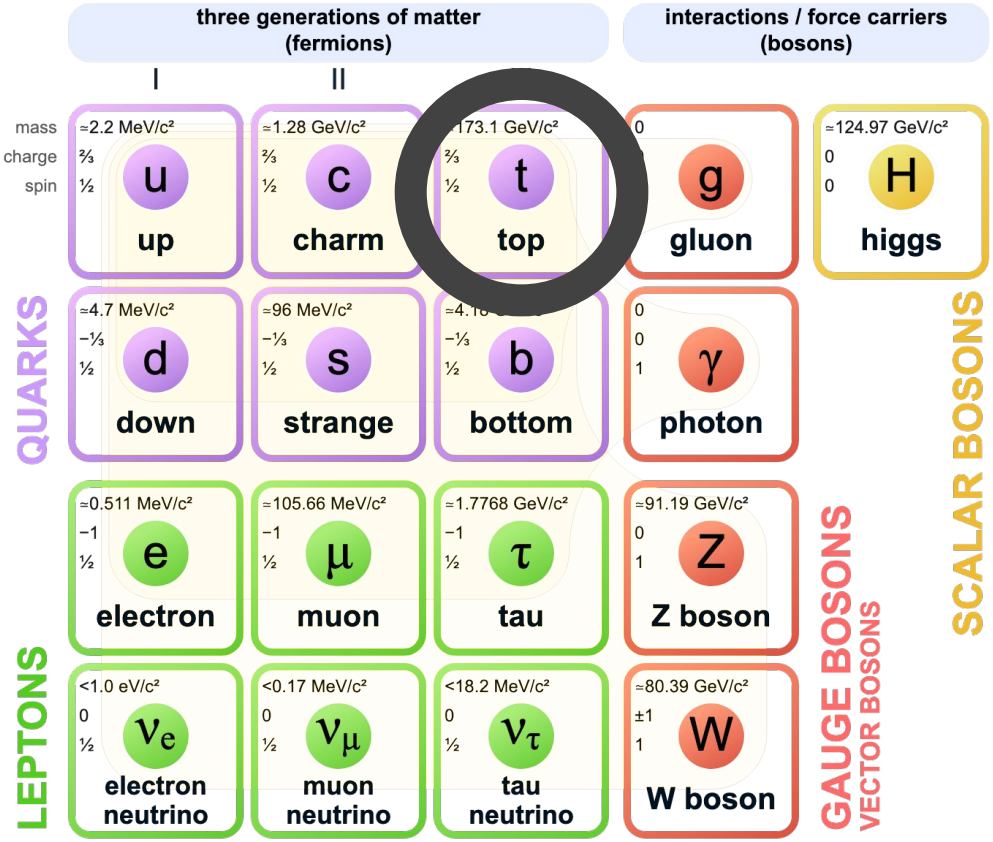
- Different parts of the detector detect different particles

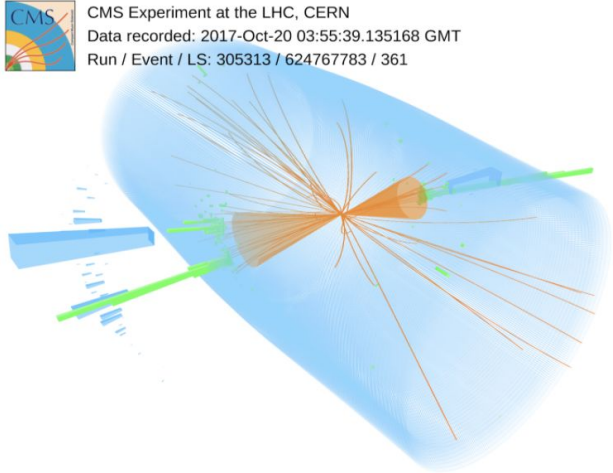


Standard Model of Elementary Particles



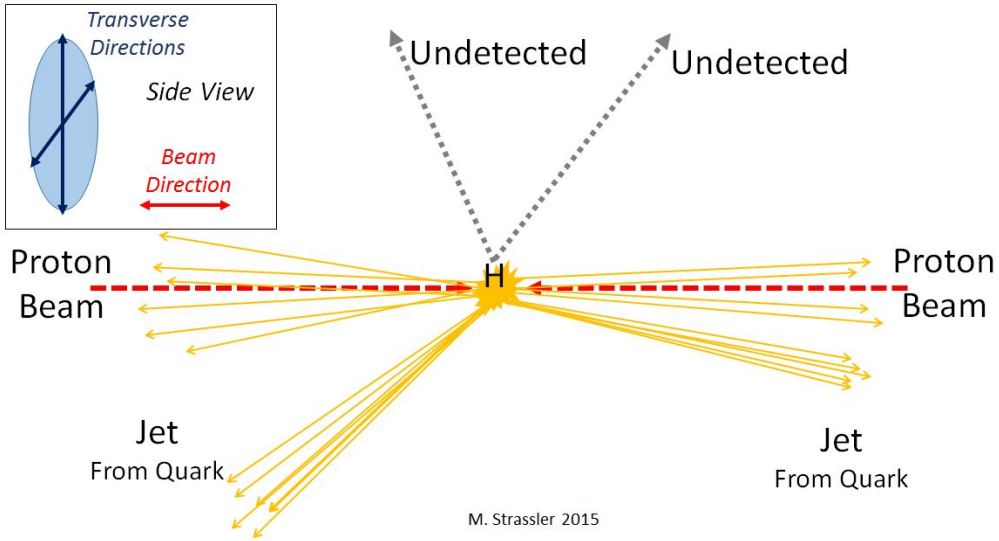
Standard Model of Elementary Particles





What is a Jet?

- A jet is a spray of particles going in the same direction
- We detect the particles, their direction (momenta), and reconstruct the “jet” via some sort of **“cone” algorithm** that sums the momentum in that cone



Where does ML fit?

-Farouk

ML in HEP

- Many applications:

(1) **Jet-classification (jet-tagging)**  **<u>Our project</u>: classify jets coming from:**

(1) top quarks (signal)

(2) QCD jets (background)

ML in HEP

- Many applications:

(1) **Jet-classification (jet-tagging)**  **<u>Our project</u>: classify jets coming from:**

(1) top quarks (signal)

(2) QCD jets (background)

In ML terms: a binary classification task

- **Input**: jet
- **Output**: label (0,1)

ML in HEP

- Many applications:

(1) Jet-classification (jet-tagging)

(2) Jet building → **choosing the set of nearby tracks and combine them into a jet**

ML in HEP

- Many applications:

(1) Jet-classification (jet-tagging)

(2) Jet building

(3) Event classification  **classifying events (broader than classifying jets.. as 1 event contains many jets)**

ML in HEP

- Many applications:

(1) Jet-classification (jet-tagging)

(2) Jet building

(3) Event classification

(4) Charged particle tracking → **identifying and classifying particle trajectories**

ML in HEP

- Many applications:

(1) Jet-classification (jet-tagging)

(2) Jet building

(3) Event classification

(4) Charged particle tracking

(5) Event simulation  **using Generative Adversarial Networks to generate detector simulated events**

This project: LGN for jet-tagging

- Link to their paper: https://arxiv.org/pdf/2006.04780.pdf
- Link to their code: https://github.com/fizisist/LorentzGroupNetwork
- Link to our code: https://github.com/faroukmokhtar/LGN
- Link to dataset: https://zenodo.org/record/2603256#.YAm6xsUzY-Q

Lorentz Group Equivariant Neural Network for Particle Physics

---

Alexander Bogatskiy [1] **Brandon Anderson** [2 3] **Jan T. Offermann** [1] **Marwah Roussi** [1] **David W. Miller** [1 4]
Risi Kondor [2 5 6]

ML preliminaries

-Aditya

ML Jargon

- **Activation function:** defines the output of the node given a set of inputs

$$f(x) = \max(0, a + bx)$$

- **Loss function:** prediction error of the neural network $||Y_{pred} - Y||_2^2$

For binary classification tasks we use binary cross entropy loss:

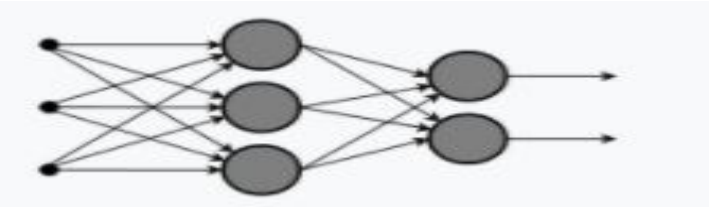
$$BCE = -\frac{1}{N}\sum_{i=0}^{N} y_i \cdot log(\hat{y}_i) + (1 - y_i) \cdot log(1 - \hat{y}_i)$$

- **Backpropagation:** algorithm to train a neural network which computing the gradient of the loss function with respect to each weight by the chain rule, computing the gradient one layer at a time, iterating backward from the last layer to avoid redundant calculations of intermediate terms in the chain rule.
- **Hyperparameter:** a parameter whose value is used to control the learning process

$$||Y_{pred} - Y||_2^2 + \lambda||Y_{pred}||_1$$

ML Jargon

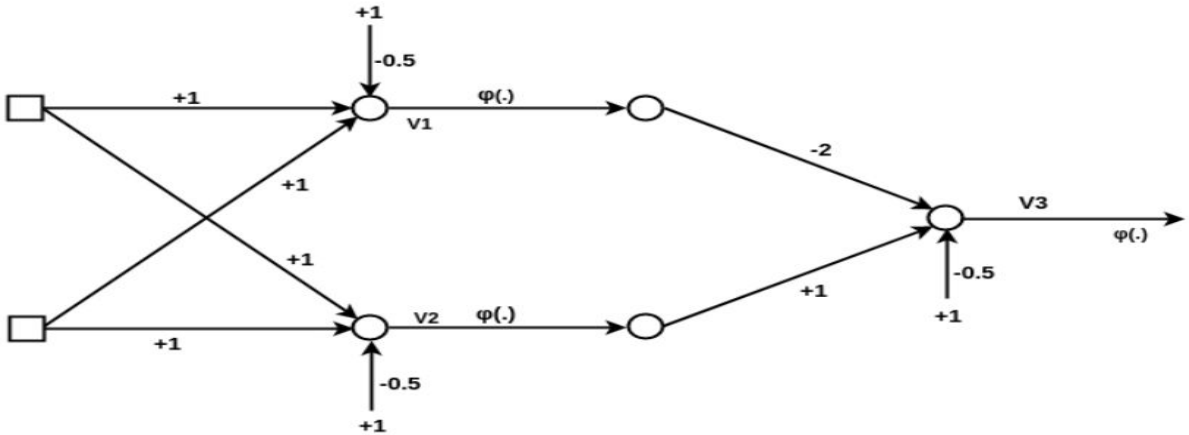
- **Epoch:** One epoch means training the neural network for one cycle. One cycle consists of one forward pass and one backward pass.
- **Multilayer Perceptron (MLP):** An MLP consists of at least three layers of nodes: an input layer, a hidden layer and an output layer. Except for the input nodes, each node is a neuron that uses a nonlinear activation function. MLP utilizes a supervised learning technique called backpropagation for training. Its multiple layers and non-linear activation distinguish MLP from a linear perceptron. It can distinguish data that is not linearly separable.



ML Jargon

- **Forward pass in MLP:** The output of each cell in the forward pass is governed by the following equation

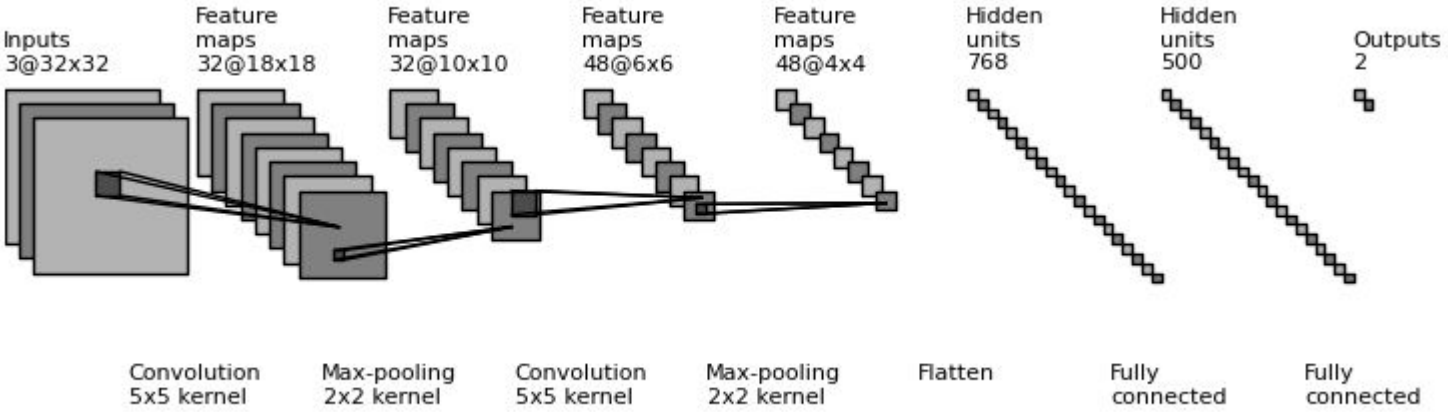
$$f(\vec{w}.\vec{x} + \vec{b})$$



Common NN models: Convolutional Neural Networks

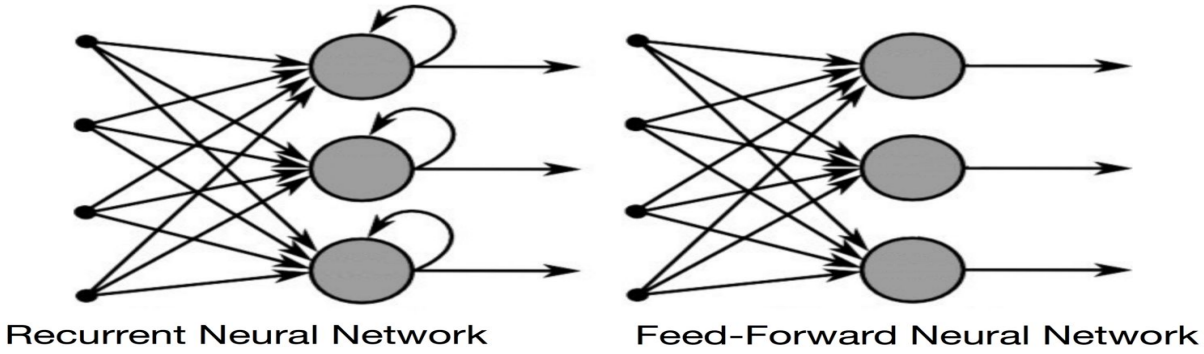
- **ConvNet:** takes advantage of the hierarchical pattern in data and assemble patterns of increasing complexity using smaller and simpler patterns embossed in the filters.
  - **Convolutional Layer:** A convolutional layer within a neural network should have the following attributes:
    - Convolutional filters/kernels defined by a width and height (hyper-parameters).
    - The number of input channels and output channels (hyper-parameter).
    - The depth of the convolution kernel/filter (the input channels) must equal the number channels (depth) of the input feature map.
    - The hyperparameters of the convolution operation, like padding size and stride.
  - **Pooling Layer:** In addition to reducing the sizes of feature maps, the pooling operation grants a degree of translational invariance to the features contained therein, allowing the CNN to be more robust to variations in their positions.

ConvNet


Inputs
3@32x32
Feature
maps
32@18x18
Feature
maps
32@10x10
Feature
maps
48@6x6
Feature
maps
48@4x4
Hidden
units
768
Hidden
units
500
Outputs
2
Convolution
5x5 kernel
Max-pooling
2x2 kernel
Convolution
5x5 kernel
Max-pooling
2x2 kernel
Flatten
Fully
connected
Fully
connected

Common NN Models: Recurrent Neural Networks

- Connections between nodes form a directed graph along a temporal sequence.
- RNNs can use their internal state (memory) to process variable length sequences of inputs.



ML models adapting on data

- **Much like:**

(1) CNNs → image-like data

(2) RNNs → sequence-like data

(3) GNNs → graph-like data

- **Motivates:**

(4) LGN ? → physics-like data?

LGN Model in theory

-Aditya

Network Architecture Overview

- The **inputs** to the network consist of the 4-momenta (E, p_x, p_y, p_z) of N_{obj} particles in the jet with associated Lorentz scalars such as spins, charges, and labels.
- The **input layer**, W_{in} , is a fully connected linear layer that divides the input into multiple channels $N_{ch}^{(0)}$
- The **CG layers,** L_{CG} , are the layers with nonlinear mappings which update the features stored in the nodes while preserving Lorentz invariance.
- The **Multi-layer perceptron,** MLP_{inv} , act on the scalar components to ensure the nonlinearity while doing nothing to other representations.
- P_{inv} projects the features in the final layer into scalars. These scalars are then added up to ensure permutation invariance. The final layer of the network is the **output layer,** W_{out} , which produces scalar weights for binary classification **[is_background_weight, is_signal_weight]** for each jet.

INPUT

- Input consists of 4-momenta of N_{obj} particles in the jet from a collision event with Lorentz scalars with them such as label, charge, spin, etc.
- The irreducible representations (without any invariant subspaces) of the Lorentz group are the tensor products of representations of $SU(2)$
- Thus, the inputs are a set of vectors $\in T^{(0,0)^{\oplus \tau_0}} \bigoplus T^{(1,1)}$ representation of the Lorentz group. In our case $\tau_0 = 2$.

$$\mathrm{SU}(2) = \left\{ \begin{pmatrix} \alpha & -\overline{\beta} \\ \beta & \overline{\alpha} \end{pmatrix} : \ \alpha, \beta \in \mathbb{C}, |\alpha|^2 + |\beta|^2 = 1 \right\},$$

INPUT LAYER (W_{in})

- The input layer W_{in} is fully connected linear layer which produces channels given by vectors in each irreducible components.
- The layer acts on the momenta+lorentz scalar vector of each particle separately but the weights are shared to enforce permutation invariance.
- The layer produces N_{obj} features $F_i^{(0)}$ for $i \in \{1, \ldots\ldots, N_{obj}\}$ Hence the dimension of the output from this layer is

$$N_{obj} \times 6^{N_{ch}^{(0)}}$$

CG LAYERS

- The input to the *p-th* CG layer (starting with *p=0*) are $N_{obj}^{(p)}$ features $\mathscr{F}_i^{(p)}$ belonging to some representations of the Lorentz group.
- The CG layers update the features according to the update rule

$$\mathscr{F}_i^{(p+1)} = \mathscr{L}_{\mathrm{CG}}\left(\mathscr{F}_i^{(p)}\right)$$

$$= W \cdot \left(\mathscr{F}_i^{(p)} \oplus \mathrm{CG}\left[\mathscr{F}_i^{(p)}\right]^{\otimes 2} \oplus \mathrm{CG}\left[\sum_j f(p_{ij}^2) p_{ij} \otimes \mathscr{F}_j^{(p)} \right] \right).$$

Tensor products -> CG decompositions -> Learnable linear operations

CG LAYERS

$$\mathcal{F}_i^{(p+1)} = \mathcal{L}_{\mathrm{CG}}\left(\mathcal{F}_i^{(p)}\right)$$

$$= W \cdot \left(\mathcal{F}_i^{(p)} \oplus \mathrm{CG}\left[\mathcal{F}_i^{(p)}\right]^{\otimes 2} \oplus \mathrm{CG}\left[\sum_j f(p_{ij}^2) p_{ij} \otimes \mathcal{F}_j^{(p)}\right] \right).$$

Models self-interaction

Mixes each irrep component to a specific $N_{\mathrm{obj}}^{(p+1)}$ channels.

Stores features in the previous layer.

Models pair-wise two-particle interactions (Ensuring permutation invariance)

CG LAYERS

$$\mathcal{F}_i^{(p+1)} = \mathscr{L}_{\mathrm{CG}}\left(\mathcal{F}_i^{(p)}\right)$$

Independent of i for permutation invariance

$$= W \cdot \left(\mathcal{F}_i^{(p)} \oplus \mathrm{CG}\left[\mathcal{F}_i^{(p)}\right]^{\otimes 2} \oplus \mathrm{CG}\left[\sum_j f(p_{ij}^2) p_{ij} \otimes \mathcal{F}_j^{(p)}\right] \right).$$

$f : \mathbb{R} \to \mathbb{R}$ is a function with learned parameters and weights the interaction.

The $\oplus$ symbol basically means stacking features together.

$p_{ij} \equiv p_i - p_j$ is the distance between two particles (in momentum space), where p_i refers to the momentum of the i-th particle. This is basically the edge features in the (fully-connected) message passing.

Clebsch-Gordan (CG) Decomposition

- Main nonlinearity in this architecture is the tensor product followed by a decomposition into irreducibles which is called CG decomposition.
- Its coefficients in certain canonical basis are called CG coefficients.
- The CG operator decomposes the tensor product into isotypic components and only a few are kept to control memory usage.

CG LAYERS: Technical details

- Only the first few irreducible representation components are kept to control memory storage.
- Tensor products are performed channel-wise to minimize computations.
- The learnable matrix W belongs to the set of matrices that parametrize all linear equivariant map between irreducible representation.
- The learnable function $f(p_{ij}^2)$ is a linear combination of Lorentzian bell-shaped curves $a + \frac{1}{b+c^2 p_{ij}^2}$ where the parameters a,b and c take 10 values each.

- In total there are 4 CG layers, and the number of channels chosen as the input of each CG layer are $N_{ch}^{(0)} = 2 \;\; N_{ch}^{(1)} = 3 \;\; N_{ch}^{(2)} = 4 \;\; N_{ch}^{(3)} = 3$

Multi-layered Perceptrons

- MLP is applied to the component $\left(T^{(0,0)}\right)\oplus N_{ch}^{(p)}$ which is the $N_{ch}^{(p)}$ scalar components in the node features.
- The input to MLP layers are $N_{ch}^{(p)}$ scalar inputs which is then produced into the same number of outputs.
- The MLP layer acts on the scalar values of each particle separately but the parameters are shared across all particles (N_{obj}) in the CG layer for permutation invariance.
- The activation function used is a linear combination of basis Lorentzian bell-shaped curves:

$$a + \frac{1}{b + c^2 x^2}$$

Output layer

- The output layer takes the arithmetic sum of N_{obj} features produced after the last layer to maintain permutation invariance.
- For classification task, the Lorentz-invariant outputs are the only vital result. Hence the output only extracts the invariant isotypic component of this sum and applies a fully connected linear layer W_{out} to the $N_{ch}^{(0)}$ channels producing two scalar weights for binary classification.

$$\vec{w}_{out} = W_{\text{out}} \cdot \left(\sum_i \mathcal{F}_i^{(N_{\text{CG}})} \right)_{(0,0)},$$

LGN Model in practice

-Farouk

(1) Computational cost
(2) Training result

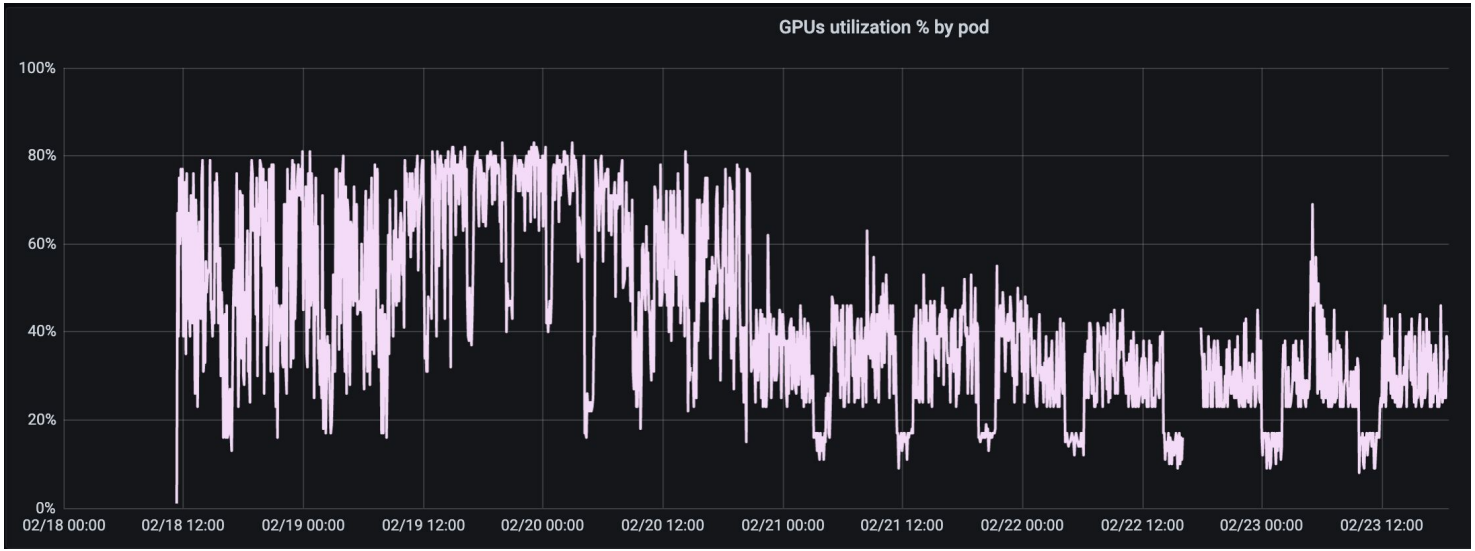
(1) Computational cost:

- **Training sample**: 1.2m jets (1.6m including validation)
- **Batch_size**=32 (more than that gives CUDA memory error)
- **GPU Utilization (on prp):**

(1) GTX 1080Ti: 100% (negligible fluctuations)

(2) RTX 2080Ti: between 30-80% (big fluctuation)

- **6-day training:**

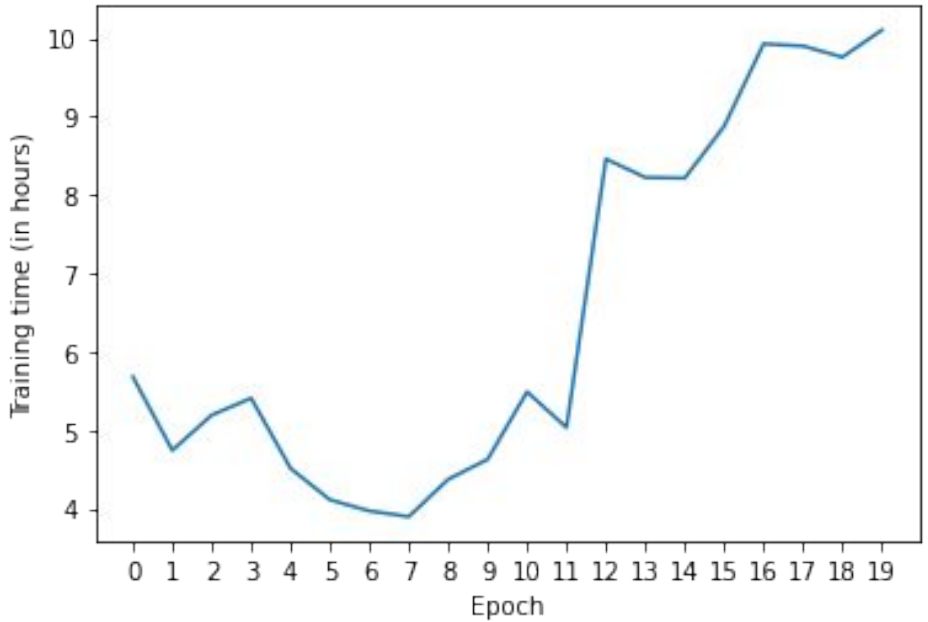


Performance and Cost The architecture was coded up using PyTorch and trained on two clusters with GeForce RTX 2080 GPU's. Each training session used one GPU and with the hyperparameters listed above it used about 3700MB of GPU memory with a mini-batch size of 8 samples. The wallclock time was about 7.5 hours per epoch, and our models were trained for 53 epochs each.

Big training

Performance and Cost The architecture was coded up using PyTorch and trained on two clusters with GeForce RTX 2080 GPU's. Each training session used one GPU and with the hyperparameters listed above it used about 3700MB of GPU memory with a mini-batch size of 8 samples. The wallclock time was about 7.5 hours per epoch, and our models were trained for 53 epochs each.

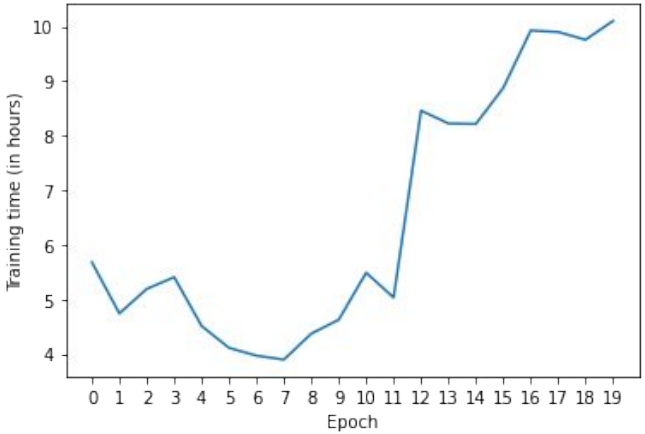
- We trained it for **6 days** over 20 epochs on RTX 2080Ti
- Time/epoch is not uniform

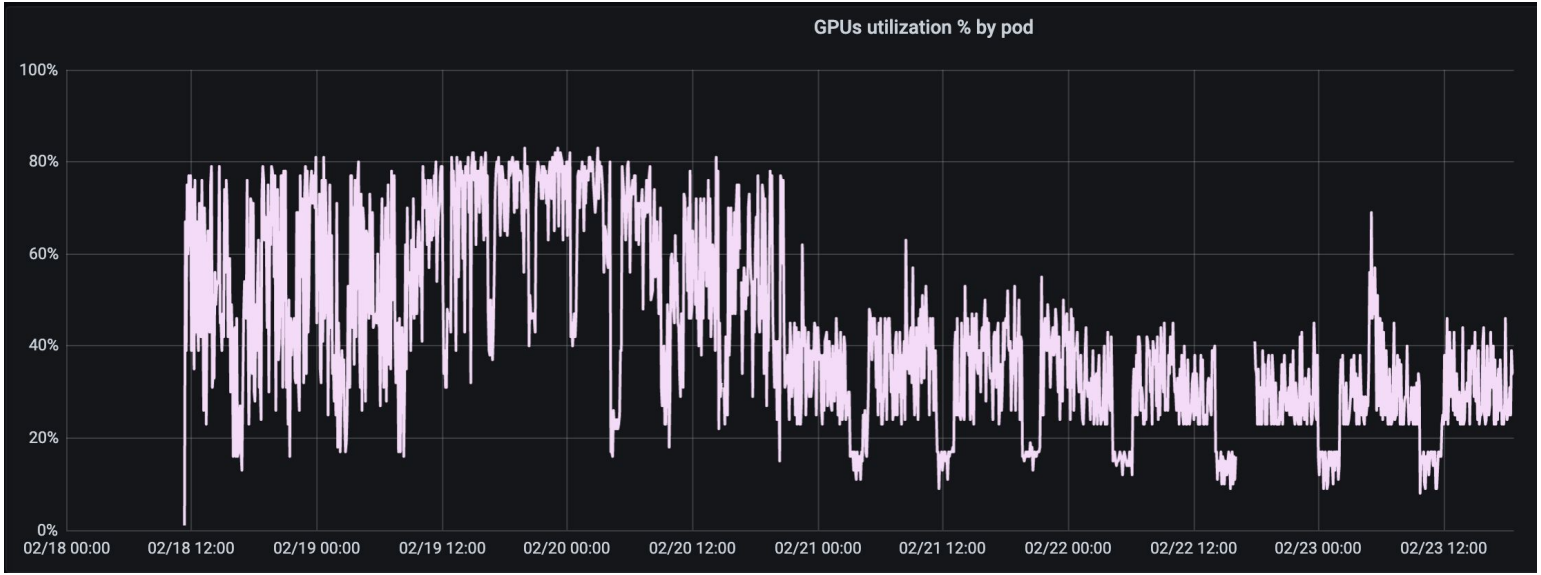


Computational difficulty

- Hardware-failure? overheating..
- Some CG decompositions become more expensive?

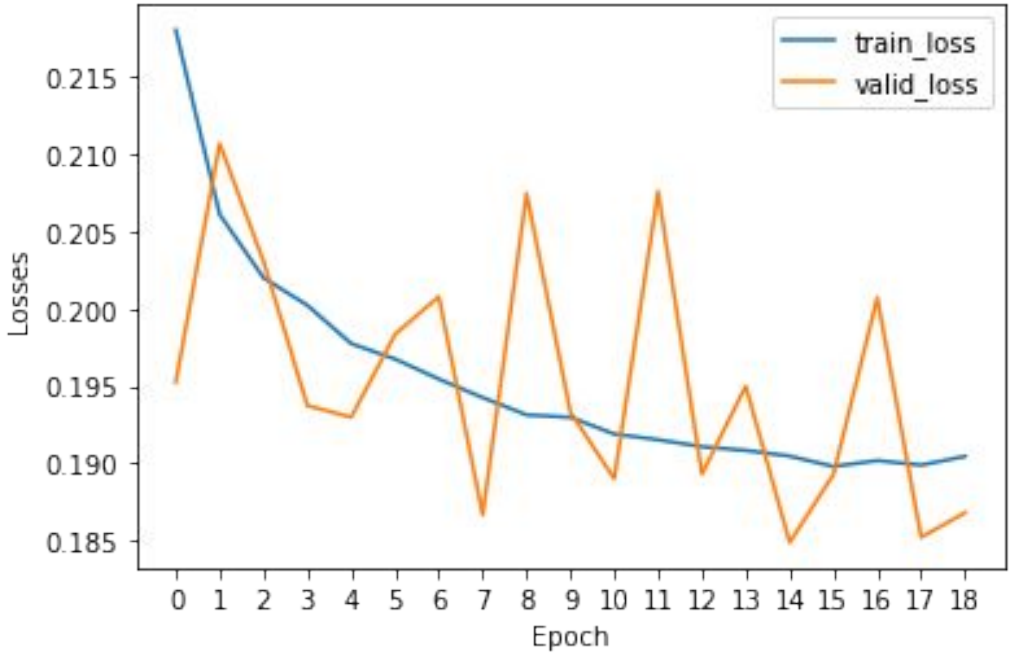
* we have to do more trials to figure the real reason..





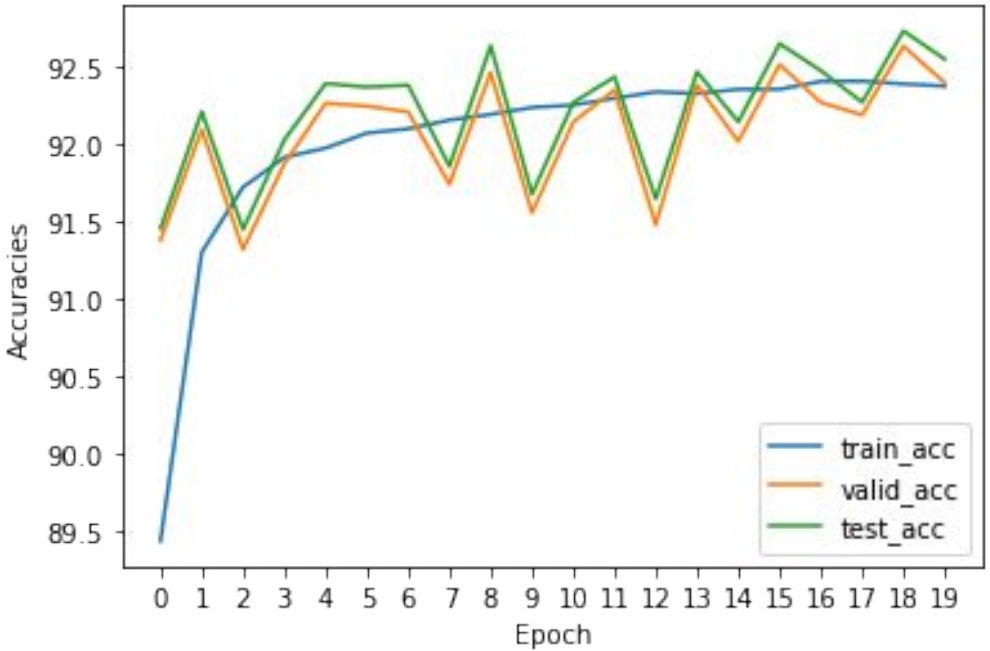
(2) Training result

- **Training is working**: loss functions are converging
- Validation loss is not smooth because the training sample is huge (1.2m jets)



Accuracy plots

- We achieve a test accuracy of **92.7% (at epoch=19)**
- Comparable to the test accuracy mentioned in the paper (**92.9%**)



Architecture	Accuracy	AUC	$1/\epsilon_B$	#Param
ParticleNet	0.938	0.985	1298 ± 46	498k
P-CNN	0.930	0.980	732 ± 24	348k
ResNeXt	0.936	0.984	1122 ± 47	1.46M
EFP	0.932	0.980	384	1k
EFN	0.927	0.979	633 ± 31	82k
PFN	0.932	0.982	891 ± 18	82k
TopoDNN	0.916	0.972	295 ± 5	59k
LGN	0.929 $\pm$ .001	0.964 $\pm$ 0.018	435 ± 95	4.5k

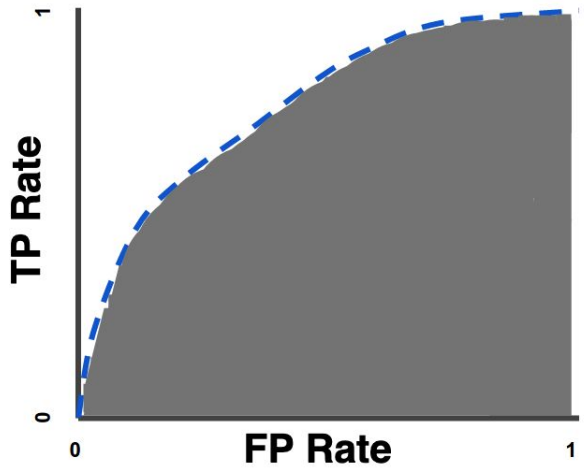
LGN Model evaluation

-Farouk

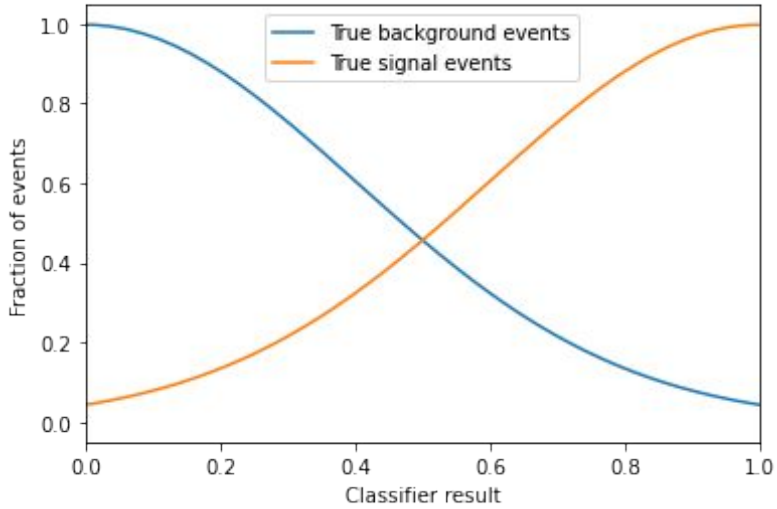
(1) ML metrics (roc curves, confusion matrix)
(2) Equivariance test

(1) ML metrics: (a) Roc curves

- **TPR**: rate at which signal is being classified as signal (more is good)
- **FPR**: rate at which background is being classified as signal (less is good)

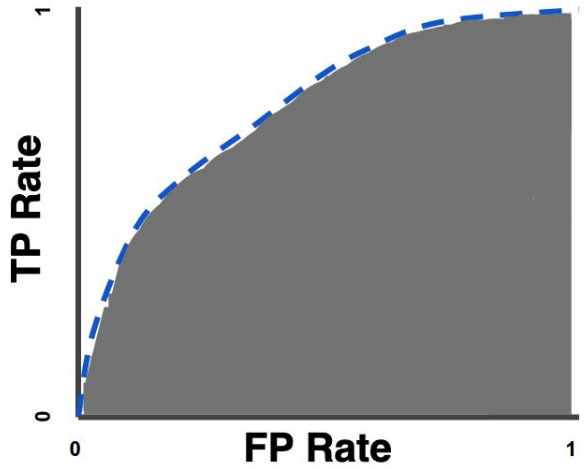


Roc curves are constructed by counting the number of events classified as “signal” vs “background” at each classifier cut

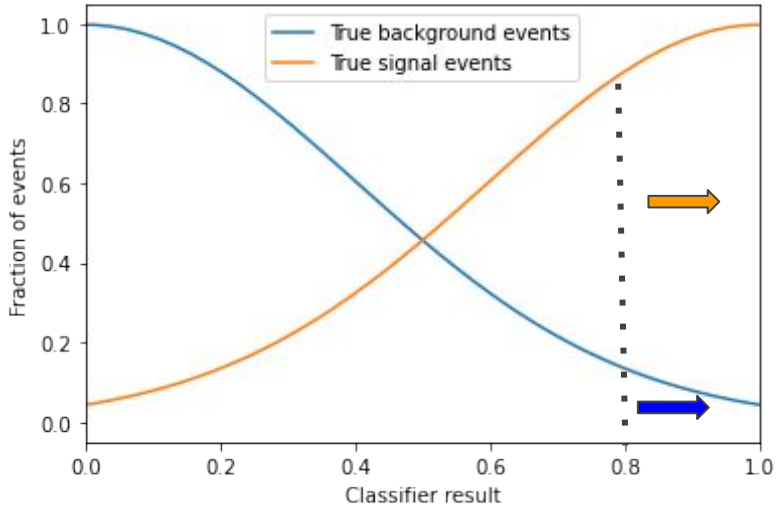


(1) ML metrics: (a) Roc curves

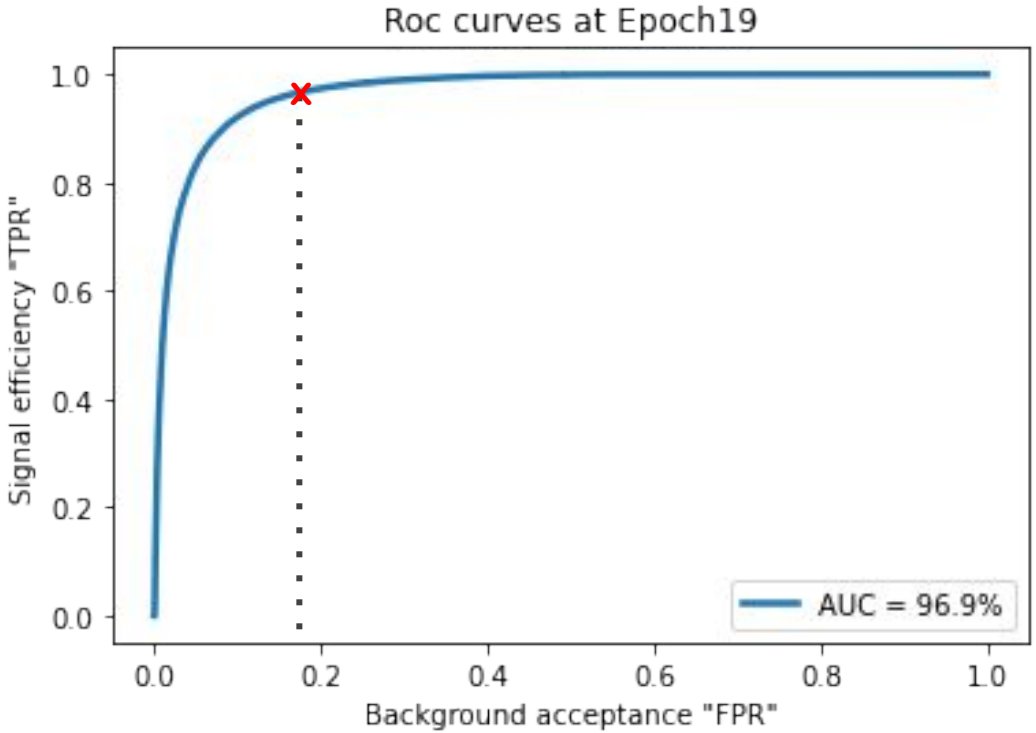
- Pick a cut.. For ex: 0.8
- # of events to the right of 0.8 of the **blue** histogram is the **FPR**
- # of events to the right of 0.8 of the **orange** histogram is the **TPR**
- Do this for all cuts (0,1) to make a Roc curve plot



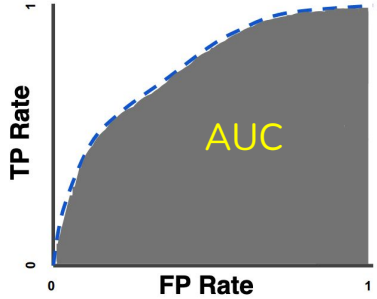
Roc curves are constructed by counting the number of events classified as “signal” vs “background” at each threshold



(1) ML metrics: (a) Roc curves

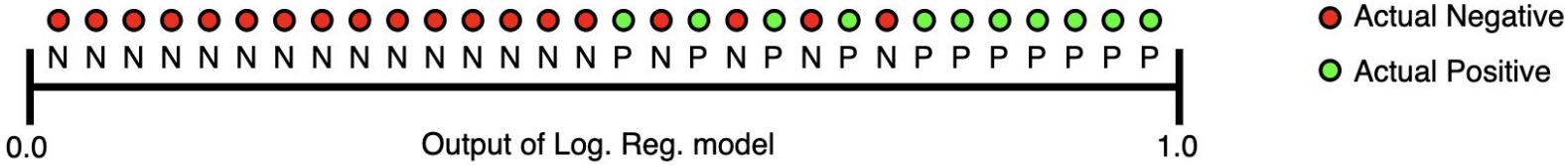


(1) ML metrics: (a) Roc curves

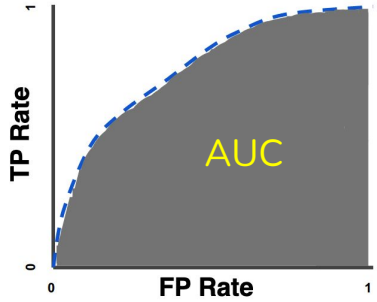


What is AUC?

- AUC provides an aggregate measure of performance across all possible classification thresholds
- AUC represents the probability that a random positive (green) example is positioned to the right of a random negative (red) example. See figure below.

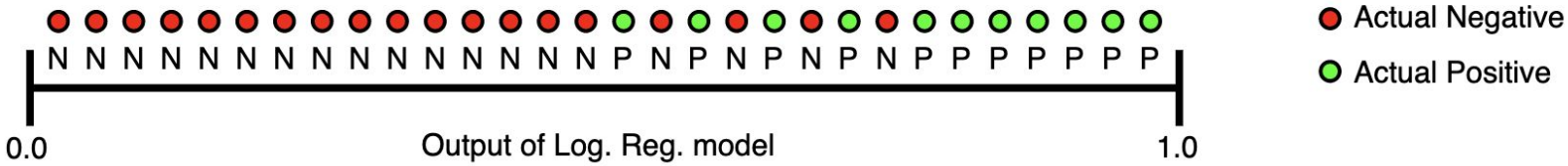


(1) ML metrics: (a) Roc curves



What is AUC?

- AUC provides an aggregate measure of performance across all possible classification thresholds
- AUC represents the probability that a random positive (green) example is positioned to the right of a random negative (red) example. See figure below.

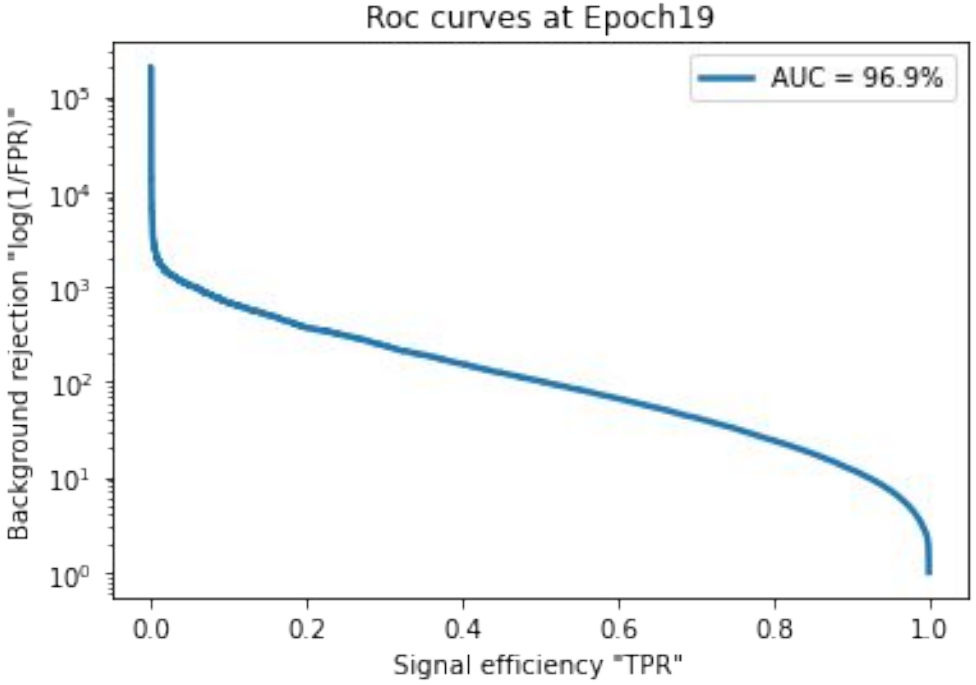


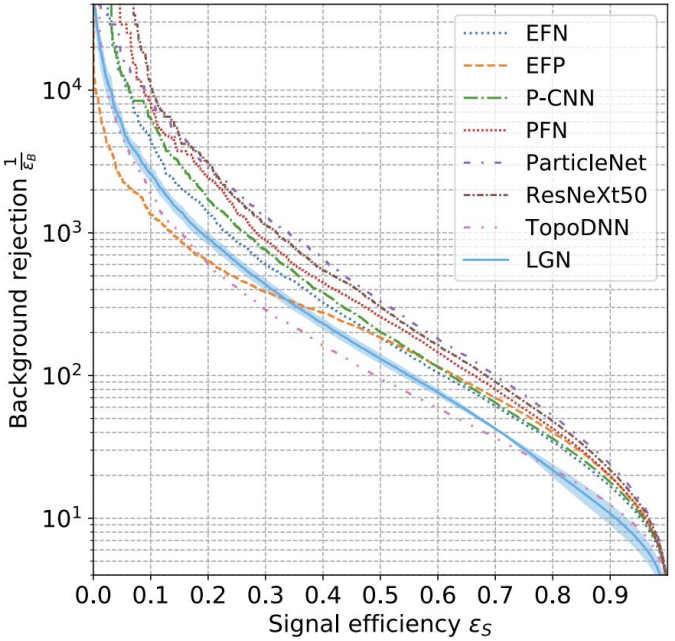
AUC closer to 100% = better

(1) ML metrics: (a) Roc curves

- Inverted Roc curve

Architecture	Accuracy	AUC	$1/\epsilon_B$	#Param
ParticleNet	0.938	0.985	1298 ± 46	498k
P-CNN	0.930	0.980	732 ± 24	348k
ResNeXt	0.936	0.984	1122 ± 47	1.46M
EFP	0.932	0.980	384	1k
EFN	0.927	0.979	633 ± 31	82k
PFN	0.932	0.982	891 ± 18	82k
TopoDNN	0.916	0.972	295 ± 5	59k
LGN	0.929 $\pm$ .001	0.964 $\pm$ 0.018	435 ± 95	4.5k

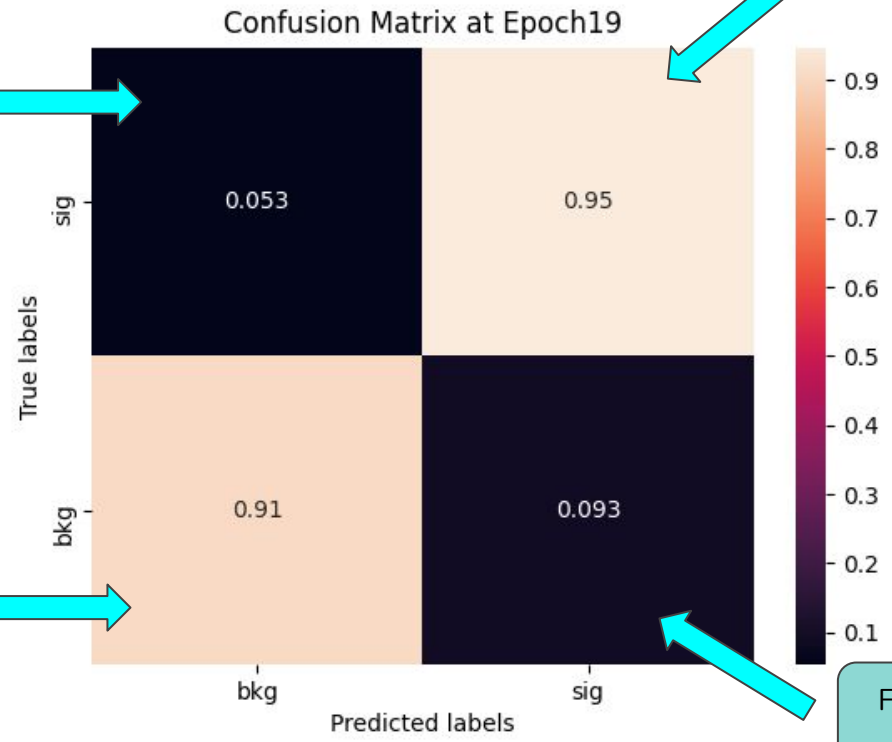




(1) ML metrics: (b) Confusion Matrix

True Positive Rate (TPR)

False Negative Rate (FNR)

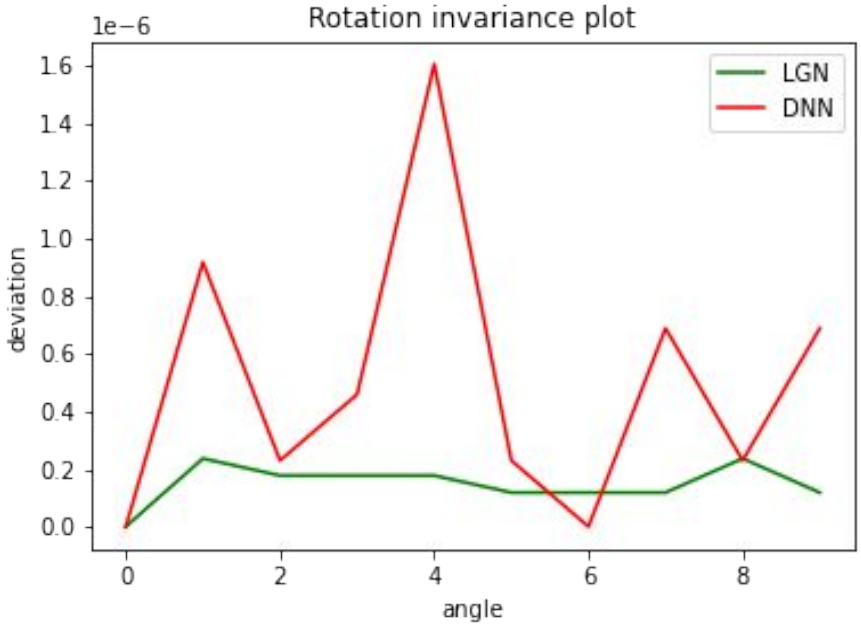


True Negative Rate (TNR)

False Positive Rate (FPR)

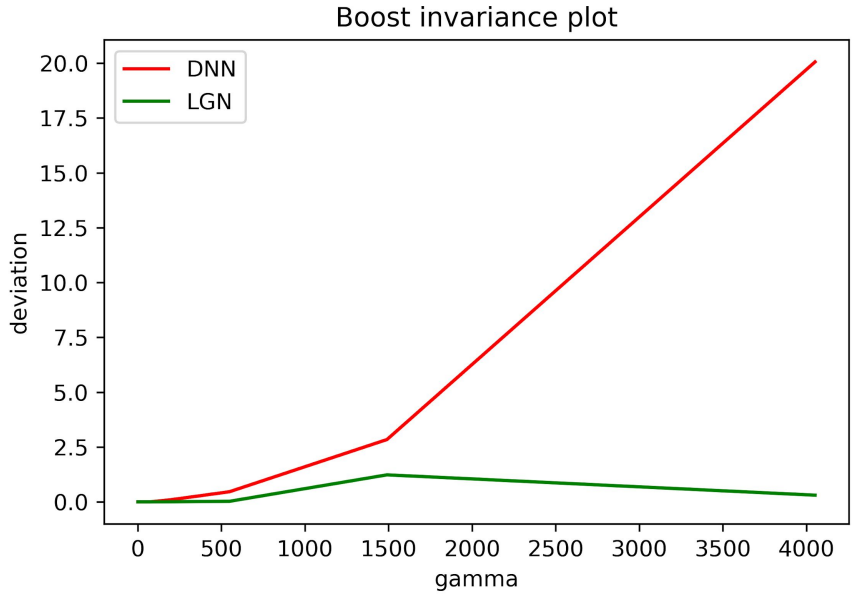
(2) Equivariance Test: (a) Rotation Invariance

- Pass an event to the model and predict its output label **Y**
- Rotate the same event and pass it to the model to predict its label $\mathbf{Y^{rot}}$
- **Deviation = |Y - Y^{rot}|**



(2) Equivariance Test: (b) Boost Invariance

- Pass an event to the model and predict its output label **Y**
- Lorentz boost the same event and pass it to the model to predict its label Y^{boost}
- **Deviation = $|Y - Y^{boost}|$**



*Gamma characterizes the speed of the moving frame

$$\gamma = \frac{1}{\sqrt{1 - \frac{v^2}{c^2}}}$$

Conclusion

Accomplished

- We've built a model which respects symmetries in physics
- The model achieved state-of-the-art benchmark performance on HEP simulated data

Further extensions:

- Apply this model to different tasks (other than jet-tagging)
- Explore other similar models which respect other symmetries in physics

Useful references

Actual paper:

- http://proceedings.mlr.press/v119/bogatskiy20a.html

Other Equivariant Networks papers:

- https://scipost.org/10.21468/SciPostPhys.5.3.028
- https://openreview.net/forum?id=J8_GttYLFgr

Thank you