MACHINE LEARNING STATUS: GNN

Workshop Roma-Napoli, Napoli, 18/12/2024

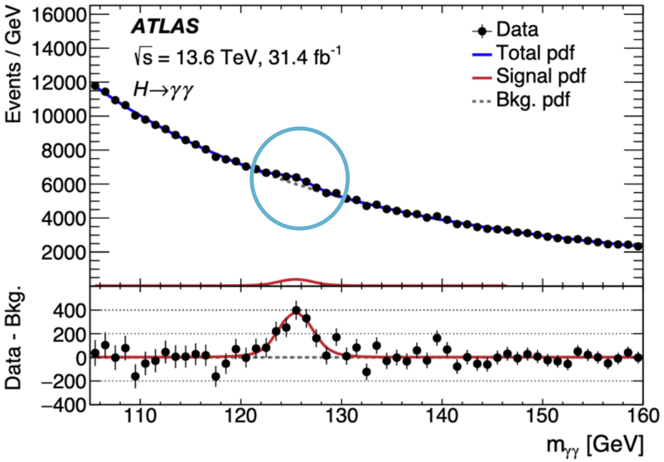
Anomaly Detection

- Anomaly Detection (AD) refers to Machine Learning (ML) techniques used to spot outliers in a dataset.

- Identification of features of detector data inconsistent with the expected background.

- Generality of prediction ∝ level of training supervision
  - Typically unsupervised scheme

Eur.Phys.J.C 84 (2024) 1, 78

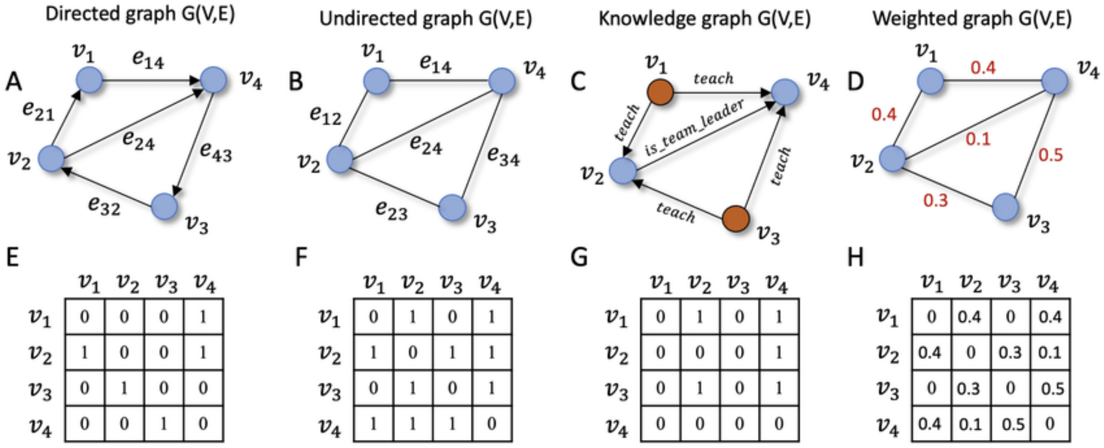


Graphs overview

Some data must be arranged in array-like objects in order to be processed by machine learning algorithms, but sometimes it just doesn't feel intuitive (protein chains, social networks between peope, ecc.)

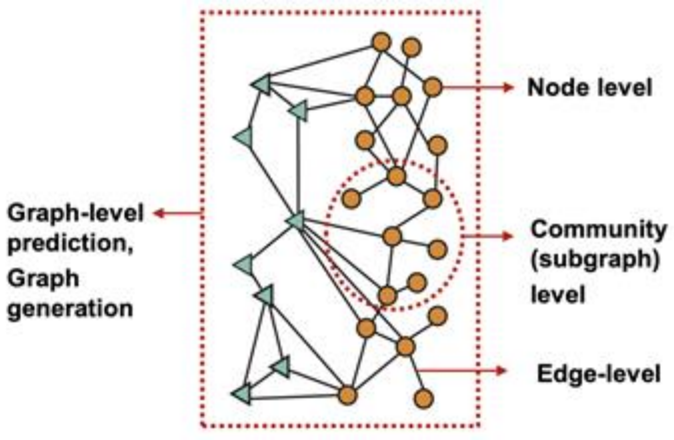
Graph representation!

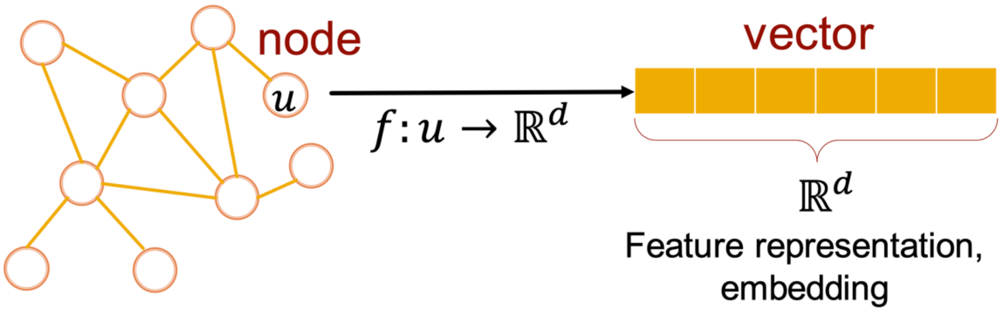
- Structured objects composed of entities used to describe and analyze relations and interactions (edges) between such entities (nodes).
  - Nodes and edges typically contain features specific to each element and each pair.
  - Represented compactly by adjacency matrix.
  - Many types of graphs based on the relations: directed, heterogeneous, bipartite, weighted ecc.



Graph Neural Networks overview

- <u>Graph Neural Networks</u> (GNNs) are ML architectures built specifically to make predictions on graphs, exploiting their relational nature.
  - The network during training learns the vector representation (embedding $\mathrm{h_v}$) of each node of the input graphs.
  - **Embedding** of a target node depends in some way on what the embeddings of the other nodes are and from its structure.





<u>Several task levels</u>, carried out by processing the final node embeddings in certain ways.

1. Node prediction by prediction $\hat{\mathrm{y}}(\mathrm{h_v})$, e.g. for identification of malevolous users among social network.
2. Edge prediction by predction $\hat{\mathrm{y}}(\mathrm{h_v}, \mathrm{h_u})$, e.g. estimate the probability of affiliation between two people of interest of a person to an object.
3. Graph identification by prediction $\hat{\mathrm{y}}(\mathrm{Pool}(\mathrm{h_v}))$, e.g. classification of protein structure.

Graph Neural Networks overview

- The embeddings are updated at each layer by aggregating the information passed between the target node and the nodes from its closest neighbourhood → message passing

$$\begin{cases} \mathrm{h}_v^{i} = \text{initial features of node } v \\ \boxed{\mathrm{h}_v^{(l+1)}} = \sigma\left(\boxed{\mathbf{W}_l} \boxed{\sum_{u \in \mathrm{N}(v)}} \boxed{\frac{\boxed{\mathrm{h}_u^{(l)}}}{|\mathrm{N}(v)|}} + \boxed{\mathrm{B}_l}\mathrm{h}_v^{(l)}\right), \boxed{\forall l \in \{0, \dots, L-1\}} \end{cases}$$

Bias term

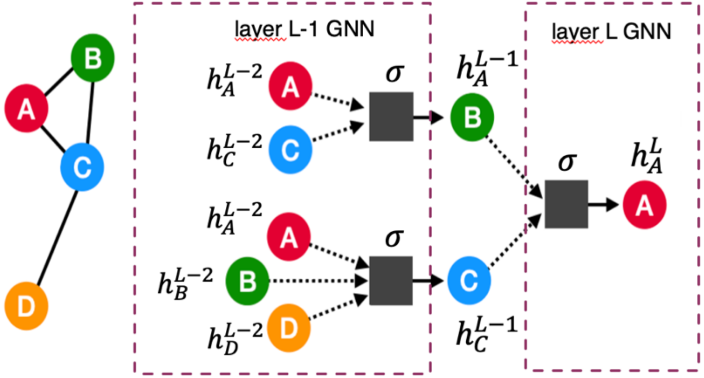
weights matrix

aggregation over N(v) neighbours u

message passed from u nodes to node v

set of L layers

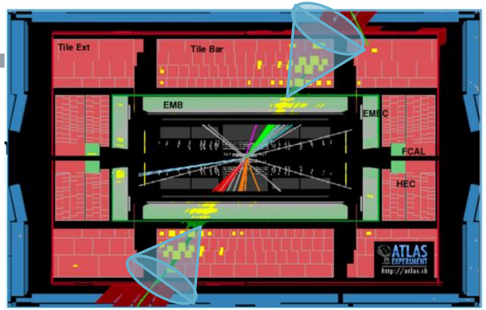
embedding representation of node v at layer (l+1)



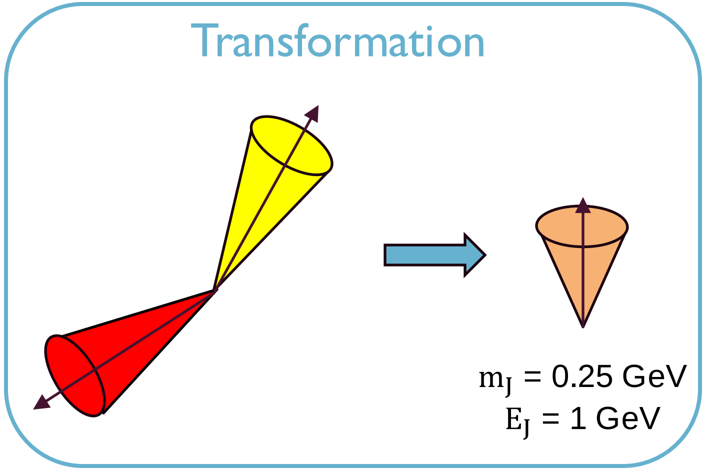
- G embedding is obtained by pooling the nodes embedding at the final layer into one global representation
  - Global sum pooling: $\mathrm{h}_G = \mathrm{Sum}(\{\mathrm{h}_v^L \in \mathbb{R}^d, \forall v \in G\})$
  - Global mean pooling: $\mathrm{h}_G = \mathrm{Mean}(\{\mathrm{h}_v^L \in \mathbb{R}^d, \forall v \in G\})$
  - Global max pooling: $\mathrm{h}_G = \mathrm{Max}(\{\mathrm{h}_v^L \in \mathbb{R}^d, \forall v \in G\})$

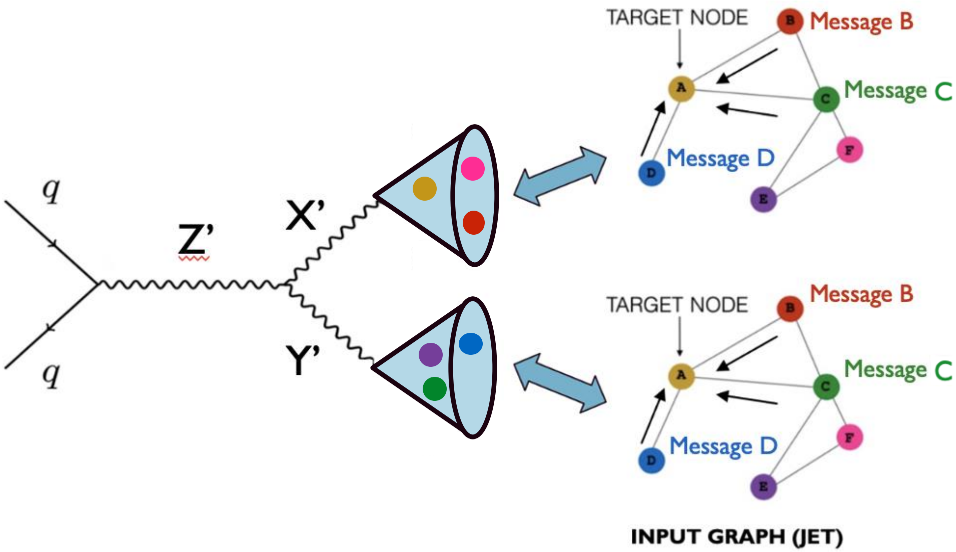
More on GNNs: https://web.stanford.edu/class/cs224w/

The idea: graphs are the new jets



- Only signal assumption: 2 boosted Large-R jets per event (Anti-kT algorithm with R = 1)
  - **Jets have sparse structure, suitable for graph representation!**
- Graph have messages:
  - **Nodes** ≡ constituents → [pT frac, η and ϕ] features
  - **Edges** ≡ relations → 1/ΔR features, present only if $\Delta R < 0.2$
- Messages propagated to obtain graph-level embedding by GNN by training optimized a specific objective
  - Data augmented for mass decorrelation of GNN prediction (transformed constituents)

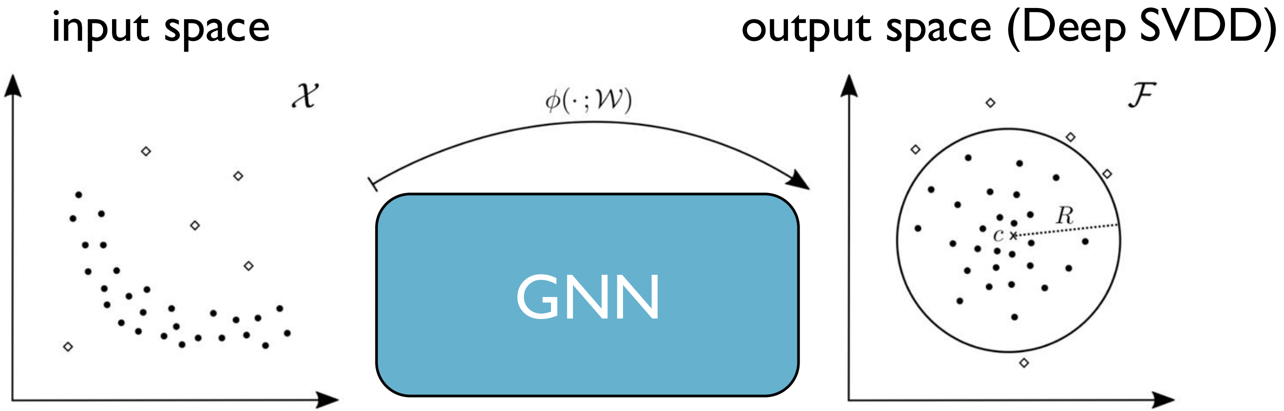




Anomaly Detection strategy

● = background
◇ = signal

C = average of initial graph features vectors

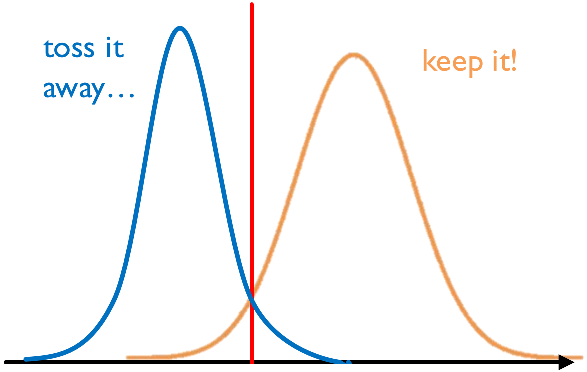


- **Key concept**: Unsupervised training on data (mostly QCD background)

- GNN maps graph features from parameters space X → F by Deep Support Vector Data Description (SVDD) objective

$$\min_{W} \quad \frac{1}{N}\sum_{i=1}^{N} \|\mathrm{GIN}(G_i;W) - \mathrm{c}\|^2 + \frac{\lambda}{2}\sum_{l=1}^{L}\|W^l\|_F^2$$

- From prediction an Anomaly Score (AS) per jet is derived



$$s(\boldsymbol{x}) = \|\phi(\boldsymbol{x};\mathcal{W}^*) - \mathbf{c}\|^2$$

Overview of available tools

- Gitlab: https://gitlab.cern.ch/atlas-roma1-napoli/lhc-olympics-with-gnns/
  - Modular structure, classes used during ML pipeline are imported from custom packages in subdirectories

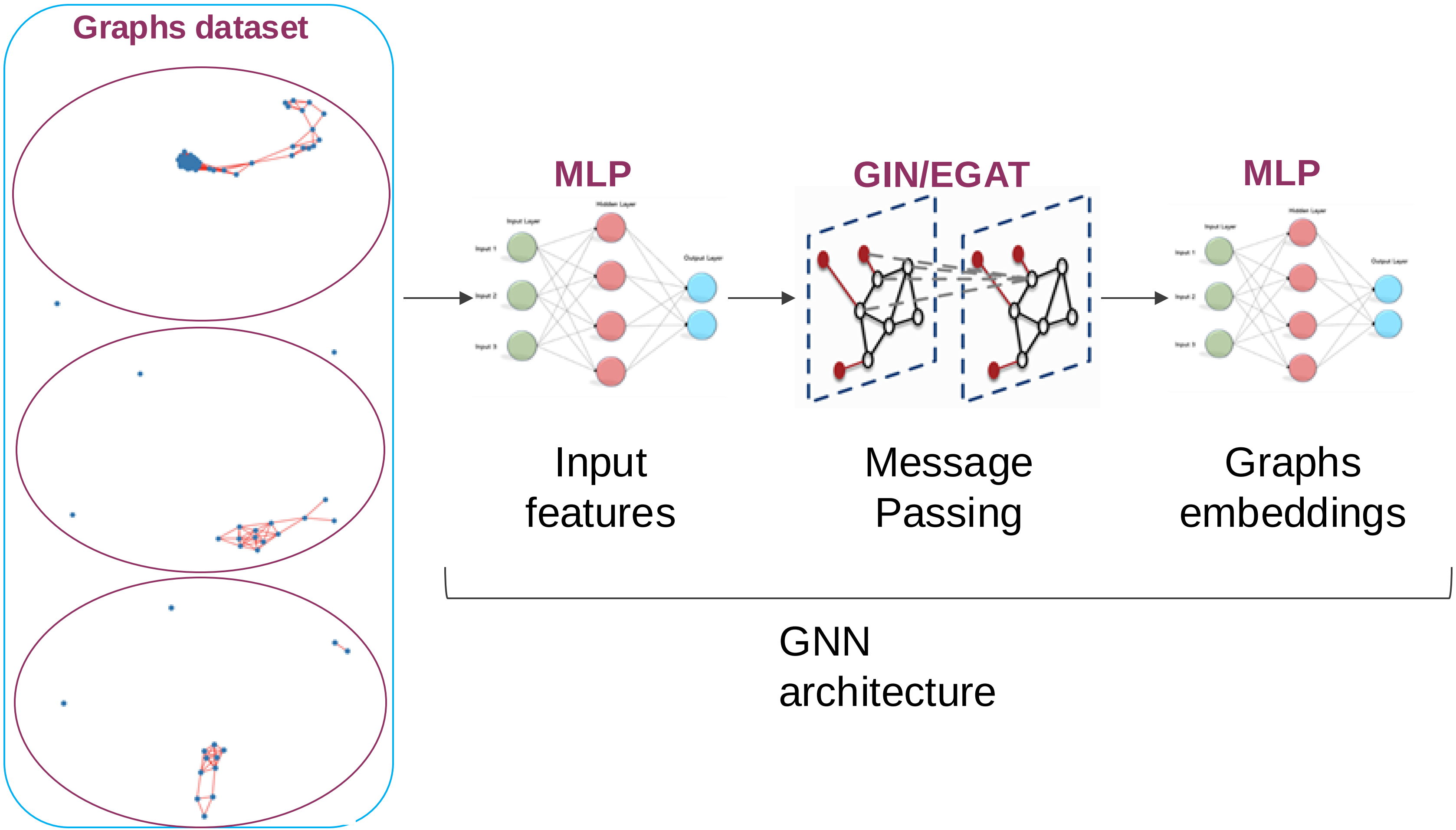
- Two simultaneous work progressions, one from Napoli (modular_Antonio) and one from Roma 1 (modular_graz)
  - Antonio: GNN, GNN + AE
  - Graziella: Transformer, Transformer + AE

- GNN notebooks and python scripts in «notebook» folder
  - LHCOly, Run 2 and Run 3 dataset oriented notebooks, scripts only for Run 3
  - Divided in supervised training scheme and unsupervised training scheme
  - Further separation between two main GNN models: Graph Isomorphism network (GIN) and Edge Graph Attention Network (EGAT)

- Training jobs can be sent on ibisco gpu node in batch mode, allowing for parallel execution (≅ 1min/epoch)
  - Useful for training on QCD samples (incompatibility between loss function and MC weights)

- Variables plotting → notebooks (Corvino's slides)
  - ROOT ntuples branches and graph distributions
  - Used now to check MC background for GNN training

Architecture structure



Graph Isomorphism Network (GIN)

- GIN formulation employs both message passing and MLPs, making it the most expressive GNN:

$$\mathrm{MLP}_{\Phi}\left(\underbrace{(1+\epsilon)}_{\text{learnable parameter}} \cdot \mathrm{MLP}_f(c^{(k)}(v)) + \sum_{u \in N(v)} \mathrm{MLP}_f(c^{(k)}(u))\right)$$

$$\boxed{c^{(k)}(u) \leftrightarrow h_j^{(l)}}$$

Embedding of node u (j) al layer k (l)

- This expression can be rewritten in a more general way, also allowing for edge weights to be considered in the graph convolution.

$$\boxed{h_i^{(l+1)} = f_{\Theta}\left((1+\epsilon)h_i^l + \text{aggregate}\left(\left\{e_{ji}h_j^l, j \in \mathcal{N}(i)\right\}\right)\right)}$$

- Aggregate can be any permutation invariant function (Sum, Mean, Max ecc.)

Edge Graph Attention Network (EGAT)

- EGAT extends on GAT model by implementing edge features in a different way and by allowing updating of the edge weights tensor between each layer of GNN (edge embedding).

GATConv

class **dgl.nn.pytorch.conv.GATConv**(*in_feats, out_feats, num_heads, feat_drop=0.0, attn_drop=0.0, negative_slope=0.2, residual=False, activation=None, allow_zero_in_degree=False, bias=True*) [source]

Bases: `torch.nn.modules.module.Module`

Graph attention layer from Graph Attention Network

$$h_i^{(l+1)} = \sum_{j\in\mathcal{N}(i)} \alpha_{i,j} W^{(l)} h_j^{(l)}$$

where α_{ij} is the attention score bewteen node i and node j:

$$\alpha_{ij}^l = \mathrm{softmax_i}(e_{ij}^l)$$
$$e_{ij}^l = \mathrm{LeakyReLU}\left(\vec{a}^T[Wh_i \| Wh_j]\right)$$

Returns:

- *torch.Tensor* – The output feature of shape $(N, *, H, D_{out})$ where H is the number of heads, and D_{out} is size of output feature.
- *torch.Tensor, optional* – The attention values of shape $(E, *, H, 1)$, where E is the number of edges. This is returned only when `get_attention` is `True`.

EGATConv

class **dgl.nn.pytorch.conv.EGATConv**(*in_node_feats, in_edge_feats, out_node_feats, out_edge_feats, num_heads, bias=True*) [source]

Bases: `torch.nn.modules.module.Module`

Graph attention layer that handles edge features from Rossmann-Toolbox (see supplementary data)

The difference lies in how unnormalized attention scores e_{ij} are obtained:

$$e_{ij} = \vec{F}(f'_{ij})$$
$$f'_{ij} = \mathrm{LeakyReLU}\left(A[h_i \| f_{ij} \| h_j]\right)$$

where f'_{ij} are edge features, A is weight matrix and

Returns:

- *pair of torch.Tensor* – node output features followed by edge output features The node output feature of shape (N, H, D_{out}) The edge output feature of shape (F, H, F_{out}) where:

  H is the number of heads, D_{out} is size of output node feature, F_{out} is size of output edge feature.

- *torch.Tensor, optional* – The attention values of shape $(E, H, 1)$. This is returned only when :attr: *get_attention* is `True`.

- **Selfloop** is required because of how the node representation is updated.

Results on LHCOlympics

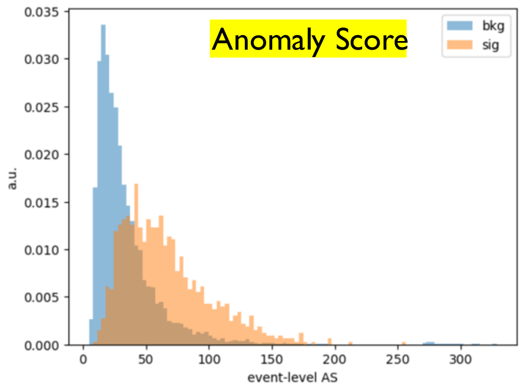
toy model

R&D LHC Olympics dataset

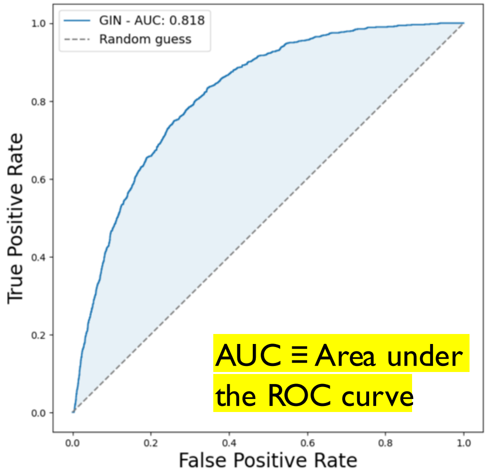
- QCD dijet events as background
- $W' \longrightarrow XY \longrightarrow qqqq$ signal events
- $m_{W'} = 3.5TeV, m_X = 500GeV, m_Y = 100GeV$
- reconstructed with anti-k_T with $R = 1.0$

benchmark models

- 3-prong signals with same masses
- W boson rediscovery (fully hadronic)
- 2-prong signals with new masses



Model	Transformer supervised	GIN supervised	EGAT supervised	Transformer *unsupervised*	GIN *unsupervised*	EGAT *unsupervised*
loss	CrossEntropy	CrossEntropy	CrossEntropy	MSE	DeepSVDD	DeepSVDD
AUC jet-level 2prong	91.3%	90.2%	89.9%	75.5%	73.7%	75.5%
AUC event-level 2prong		96.5%	96.5%		79.6%	81.8%
AUC jet-level 3prong	86.8%	75.5%	84.8%	69.1%	52.6%	67.2%
AUC event-level 3prong		84.1%	92.4%		54%	74.3%



*Event-level ≡ mean of AS pair (J_1, J_2)

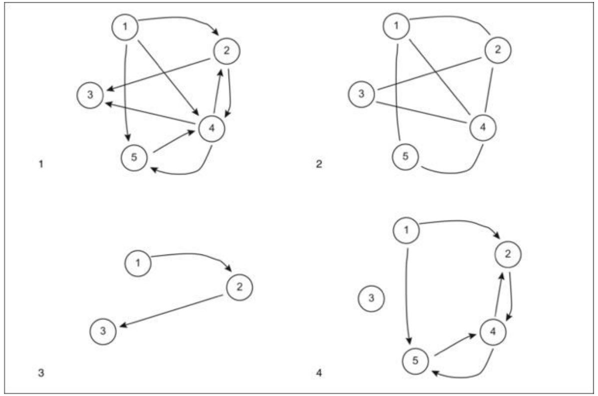
Run3 Input datasets

- Available in common workspace on ibisco:
  - /srv/Large01/ATLAS/LLJ1/datasets/dr02_SelfLoop/
  - /srv/Large01/ATLAS/LLJ1/datasets/dr02_noSeflLoop_extra/

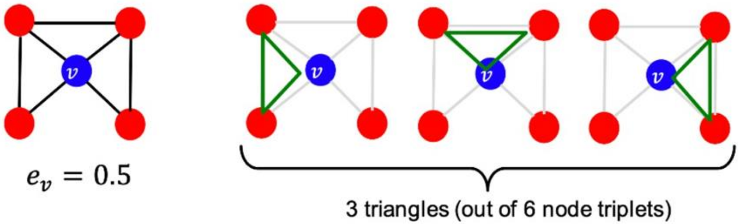
- Created with GraphDatasets.py script available at repo GNN_AD
  - code/GraphDatasets.py

- Input: ROOT ntuple (data, MC signal or background)
  - Mid-output: ROOT ntuple after skimming and transformation has been performed (very fast)
  - Final output: graph dataset (graphs list + global features dictionary), 20k events in about 40 seconds (≅ 2s/1k)
- If «extraFeatures» (clustering coefficient e_v, diameter, node degree, node and pT fractions of components 0 and 1)

- Datasets can be merged to have bkg + sig events with arbitrary ratio
  - Done by changing filenames list in notebooks/merge_datasets.py



$$e_v = \frac{\#(\text{edges among neighboring nodes})}{\boxed{\binom{k_v}{2}}} \in [0,1]$$



TRAINING ON QCD BACKGROUND

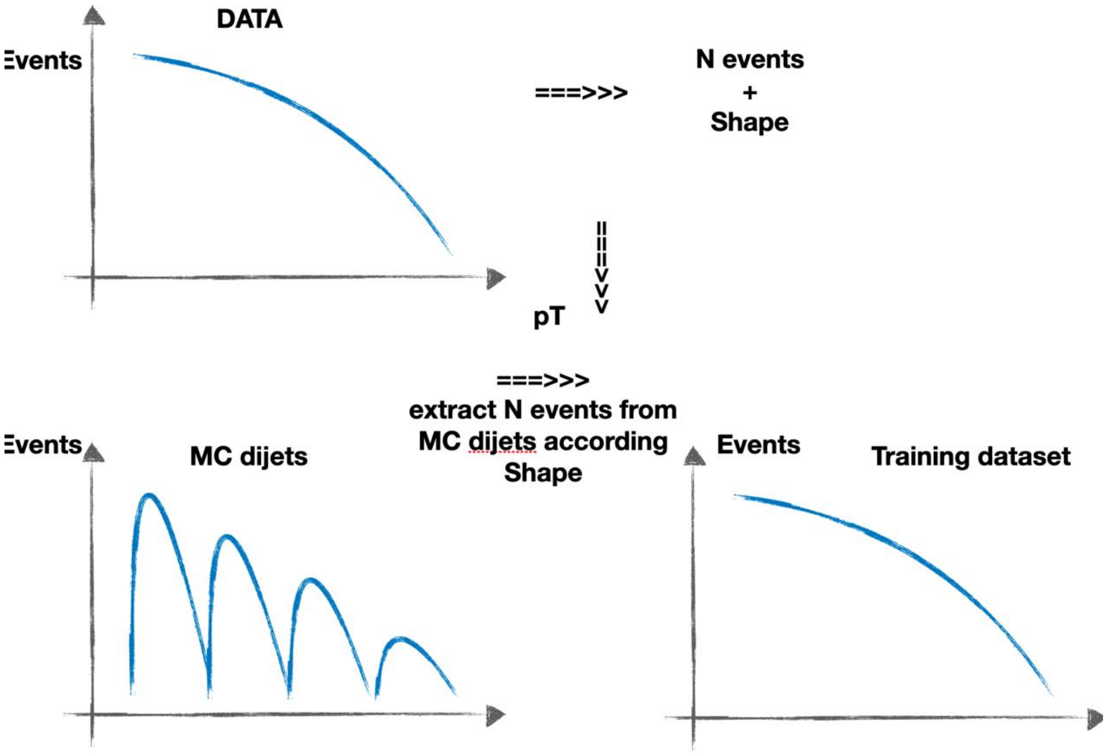
- Control region (CR) not yet defined, better train on run 3 MC samples to avoid unblinding

- Problem: training on MC QCD samples should be done on the whole datasets, since loss function can’t be easily reweighted in case of subsampling
  - Unfeasible due to time and size required for graph datasets creation

- Idea: train N networks separately on each QCD slice, then test their perfomance on benchmark signals + background dividing the test dataset in N slices (one for each network corresponding to the QCD slice phase space region)
  - At the end, we get an anomaly score for each jet like usual

TRAINING ON QCD BACKGROUND

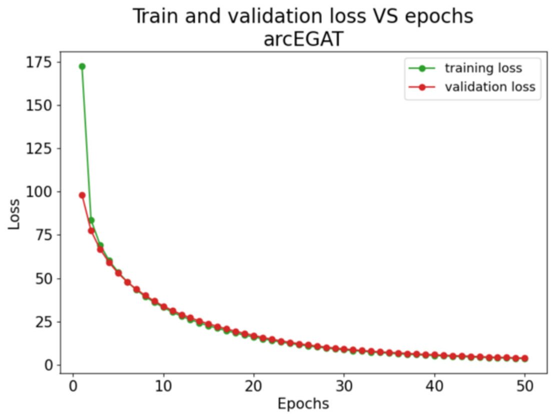
- Other idea: create a randomly extract events from the full MC background dataset considering the number of events and the shape of a variable (leading jet pT) in data
  - Single MC QCD dataset with events shape resembling data and no reweighting problematic



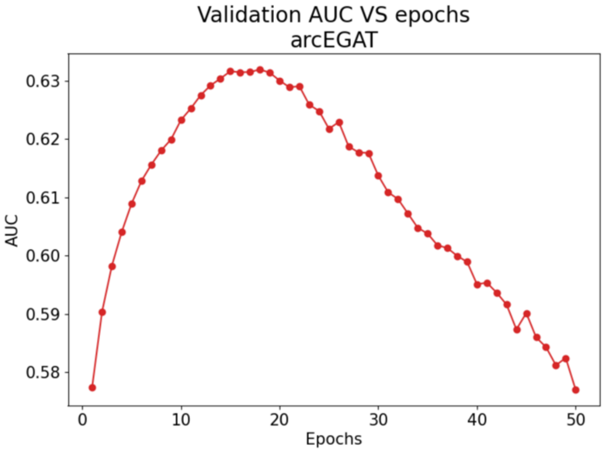
FIRST IDEA PROGRESS: EGAT

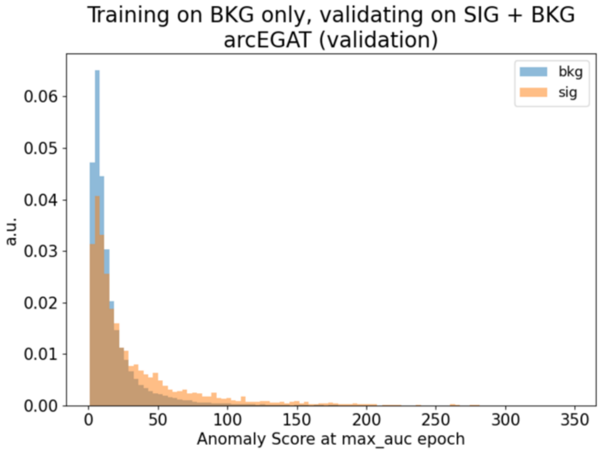
- Removed divergent loss term in EGAT training

$$\min_{W} \quad \frac{1}{N}\sum_{i=1}^{N}\|\mathrm{GIN}(G_i;W)-\mathrm{c}\|^2 + \cancel{\frac{\lambda}{2}\sum_{l=1}^{L}\|W^l\|_F^2}$$



- Performed on training to check everything is ok with one MC bkg slice (JZ5) and a YXH signal sample (2300_300)
  - Unsupervised, 150k jets in training set, 45k in validation (40k bkg – 5k signal)

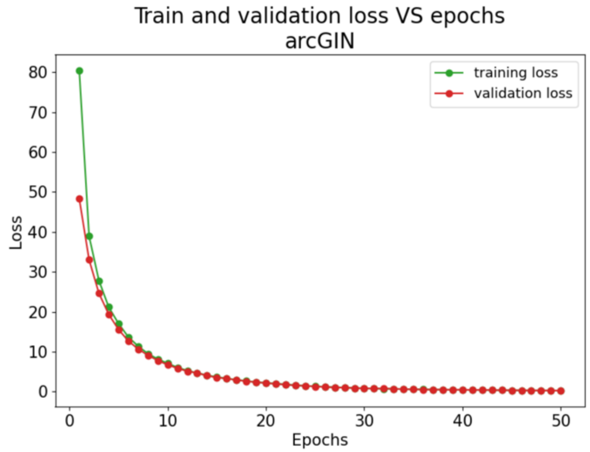


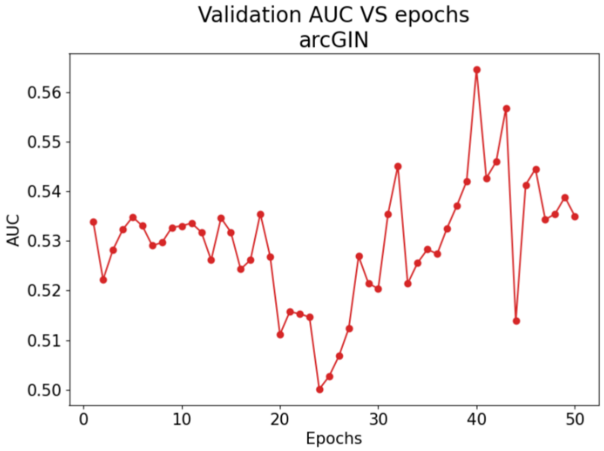


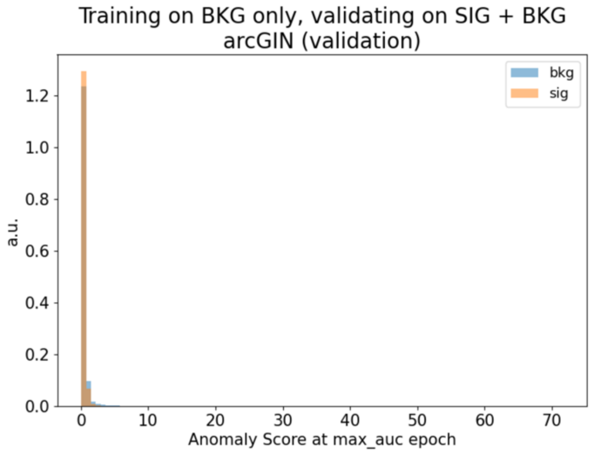
FIRST IDEA PROGRESS: GIN

- Performed on training to check everything is ok with one MC bkg slice (JZ5) and a YXH signal sample (2300_300)
  - Unsupervised, 150k jets in training set, 45k in validation (40k bkg – 5k signal)

$$\min_{W} \quad \frac{1}{N}\sum_{i=1}^{N} \|\mathrm{GIN}(G_i; W) - \mathrm{c}\|^2 + \frac{\lambda}{2}\sum_{l=1}^{L} \|W^l\|_F^2$$



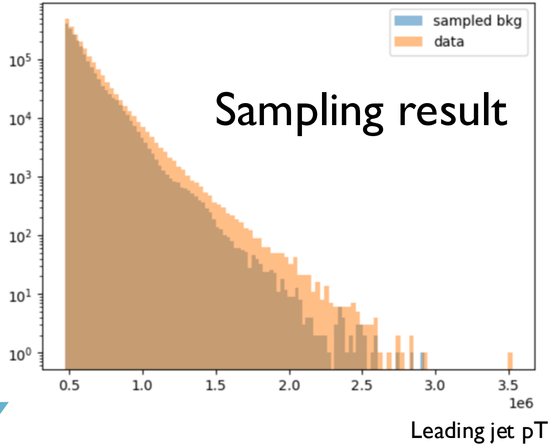


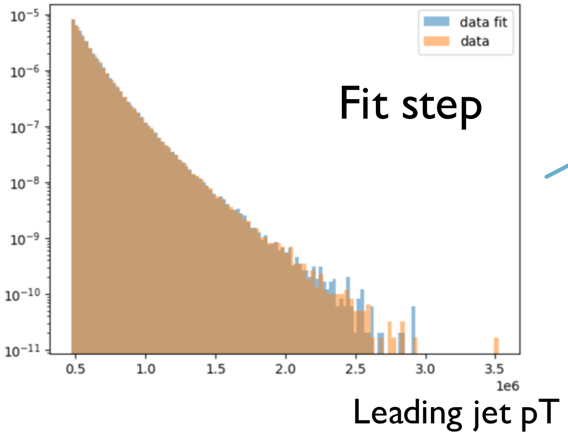


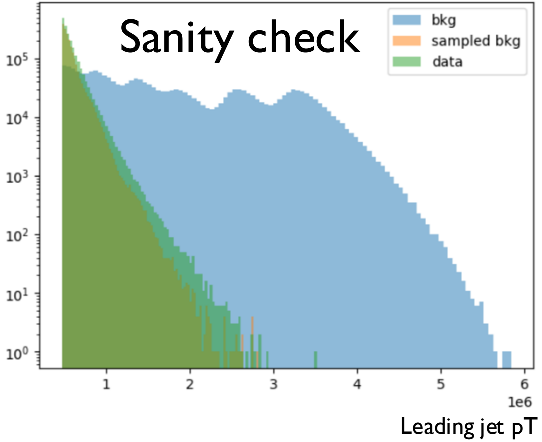
SECOND IDEA PROGRESS

- FastFrames ntuples, trigger selection applied
  - 2M events (pdf and sampling)

- pT shape of data is estimated from histogram using scipy library
  - A weight value is associated to each bkg event by evaluating the inferred pdf
  - A new bkg dataset is then created by sampling N events according to the probability (weight value) of each event to be extracted







- Starting bkg dataset is not uniform
  - Even though bkg events are randomly extracted by data pdf, most recurring events will create a modulation

TO DO LIST

- Adjust MC background training set method
- Create graph dataset from FF ntuples (only EJ working so far)
  - Segmentation Fault at dump step of Rdataframe, investigating
- Perform training with MC samples
- Let’s discuss other items in next session

BACKUP

GNN + AE ARCHITECTURE

- Main GNN model

- A popular anomaly detection loss function, the Mean Squared Error (MSE), could be used, but an AE must be connected to compare each graph representation with the autoencoder (AE) output.