TÉCNICO LISBOA

LABORATÓRIO DE INSTRUMENTAÇÃO E FÍSICA EXPERIMENTAL DE PARTÍCULAS

Fundação para a Ciência e a Tecnologia

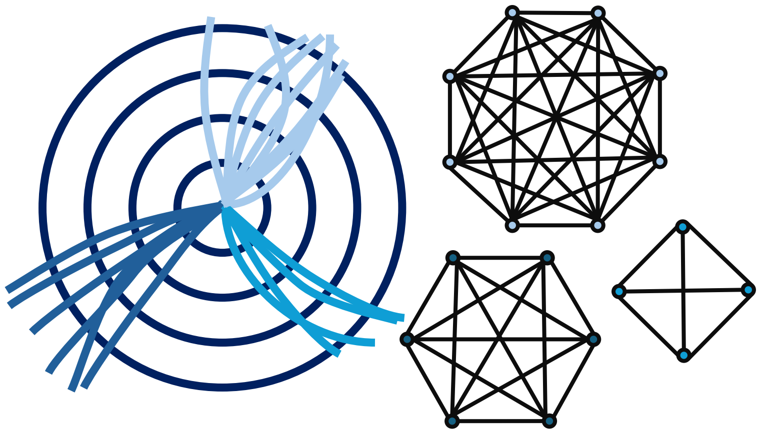
New Physics Searches with Graph-Based Anomaly Detection in High-Energy Collisions

Inês Moreira† *Nuno Castro* *Rute Pedro*

Laboratory of Instrumentation and Experimental Particle Physics

† *ines.p.moreira@tecnico.ulisboa.pt*

[ML4Jets2024 – 7/11/24]



Jets as New Physics Probes

Event level searches:

Examples of searches with focus on **fully hadronic final states**:

- "Search for new heavy resonances decaying to WW, WZ, ZZ, WH or ZH boson paris in the all-jets final state in proton-proton collisions at $\sqrt{s} = 13$ TeV"

  CMS letter

- "Search for diboson resonances in hadronic final states in 139 $fb^{-1}$of pp collisions at $\sqrt{s} = 13$ TeV with the ATLAS detector"

  ATLAS letter

Examining the multijet invariant mass to identify anomalous events

Jet level searches:

Several works focus on anomaly detection at jet level:

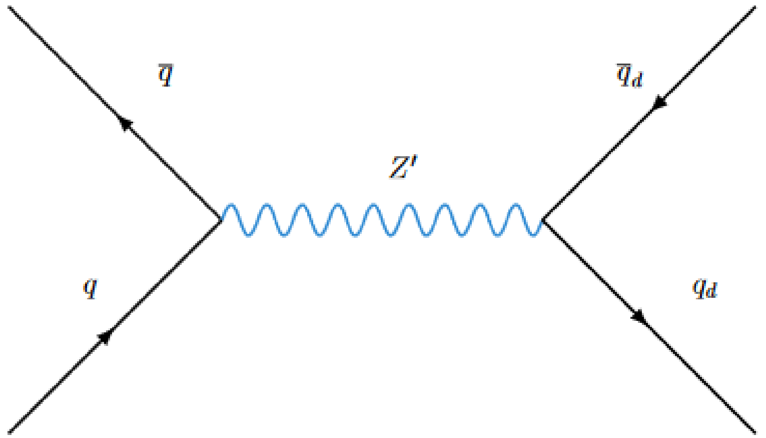
- ***QCD or what?*** QCD jets vs. top jets CNN based autoencoder on jet images
- ***Anomalous Jet Identification via Sequence Modeling*** VRNN model jets as sequences of constituent 4-vectors.
- ***What's Anomalous in LHC jets?*** QCD jets vs. Dark jets K-means clustering, Dirichlet VAE, invertible neural networks.

Dark Jets > jets with topological differences from QCD jets > **jet-level** identification of anomalous events

Dark Jets – Hidden Valley Models

- Dark sector with confining force $SU(N_d)$
- Dark partons, produce a dark shower and hadronize into dark hadrons
- Dark hadrons can decay to SM particles through certain portals, producing a jet-like signal - **Dark Jets**.

Considering resonant production of dark quark pair mediated Z':



Hidden group extended from $SU(N_d)$ to $SU(N_d) \times U(1)'$

⬇

Dark hadrons can decay into **dark photon** γ' pairs. Through kinetic mixing γ' can then decay into SM particles

- Dark hadrons stable → certain amount of missing energy → m_T useful for discriminating a dark jet pair
- Most of the dark hadrons decay promptly into SM particles → almost any displaced objects or missing energy

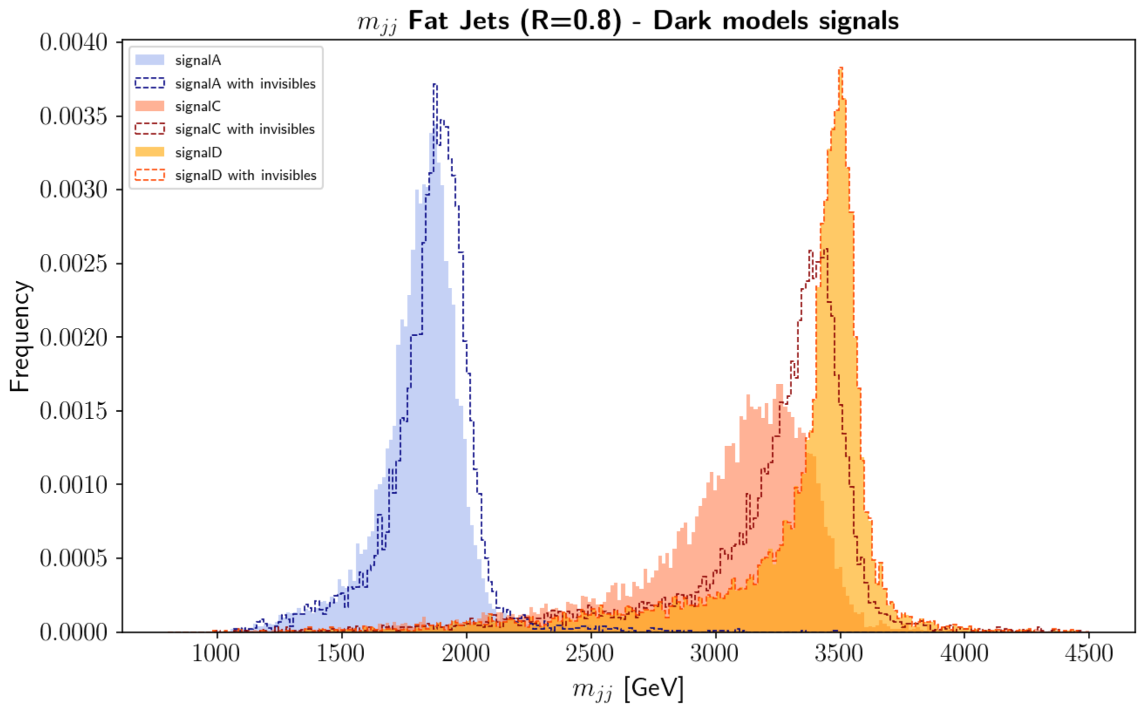
Data generation

Dark Jets events production

Benchmark signal models : dark hadrons decay **promply** into SM particles

signal	N_d	n_f	Λ_d	m_{q_d}	m_{π_d}	m_{ρ_d}	π_d decay channel	ρ_d decay channel
A	3	2	15	20	10	50	$\pi_d \to c\bar{c}$	$\rho_d \to \pi_d \pi_d$
C	3	2	15	20	10	50	$\pi_d \to \gamma' \gamma'$	$\rho_d \to \pi_d \pi_d$
D	6	6	2	2	2	4.67	$\pi_d \to \gamma' \gamma'$	$\rho_d \to \pi_d \pi_d$

Models described at *Tagging a jet from a dark sector with Jet-substructures at colliders* , where n_f represents the number of light dark quarks families and Λ_d is the confinement scale



signal A $m_{Z'} = 2000\ GeV$
signal C, D $m_{Z'} = 3500\ GeV$
$\Gamma_{Z'} = 0.1\ GeV$ for all signals

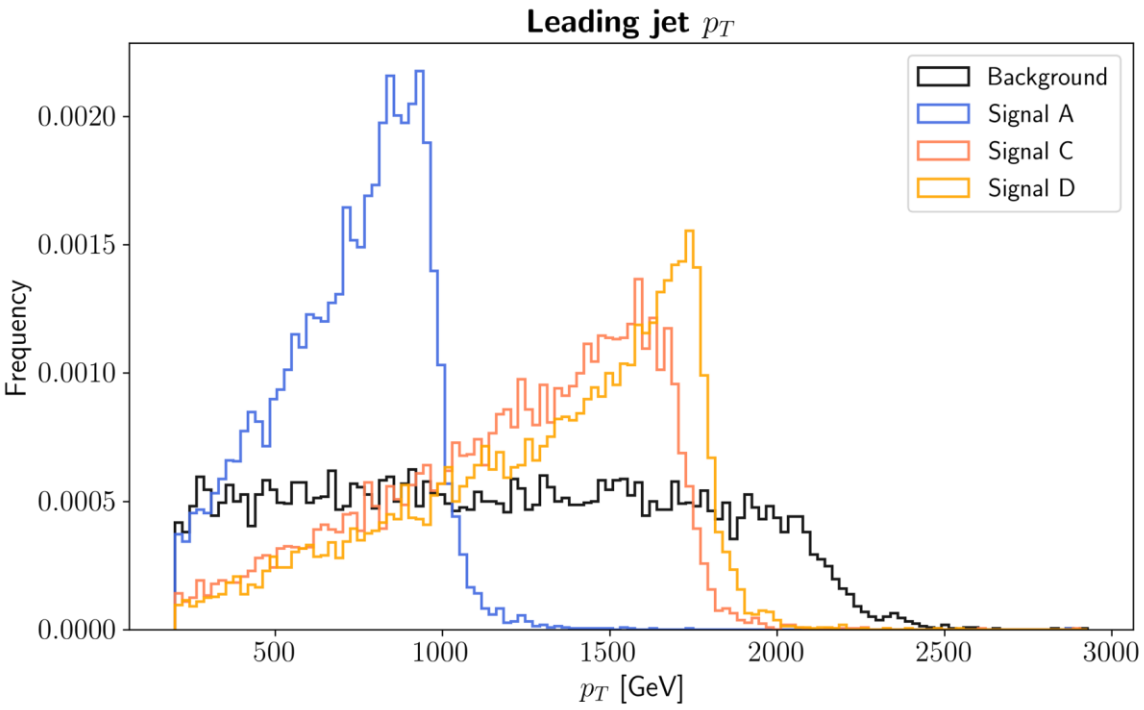
signalC, $m_{\gamma'} = 4\ GeV$
signalD, $m_{\gamma'} = 0.7\ GeV$

40 k events generated for each signal type

Dark jets signals simulation: Pythia8 and Delphes3

Data generation

Dark Jets and QCD dijet events production



QCD dijet background simulation: MadGraph5_aMC, Pythia8 and Delphes3

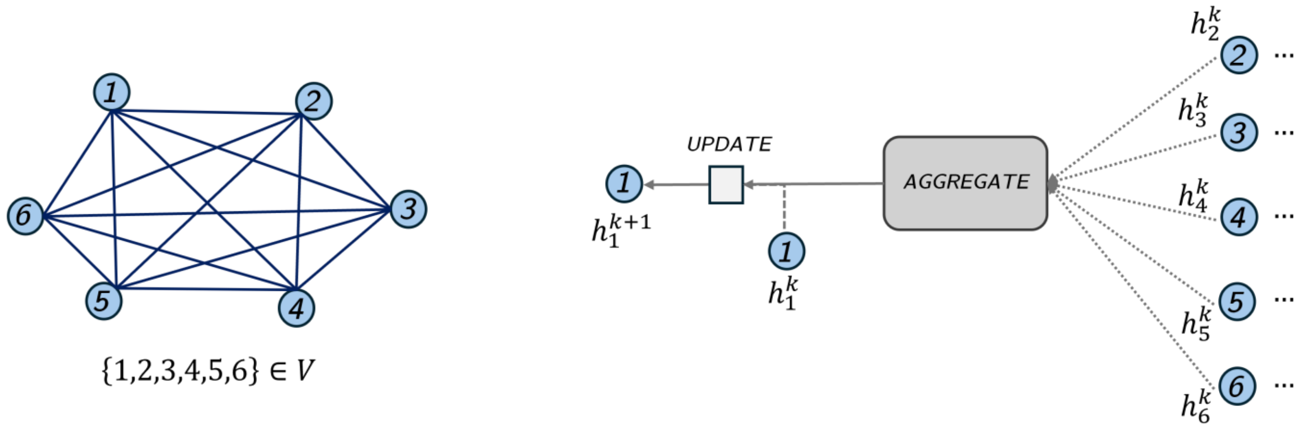
450 K QCD dijet events produced. 20 slices of 22.5 K events.

First slice 150 GeV < p_T < 250 GeV, Last slice 2050 GeV < p_T < 2150 GeV
Each covering a p_T range of 100 GeV

Accessing jet constituents through jet-reclustering:

Default Delphes card settings modified → access inputs of Delphes FastJet module
Output of EFlowMerger module → clustering with pyjet ($p = -1$, $R = 0.8$, p_T min $= 200$ GeV)

Graph Neural Networks architecture



Schemes adapted from ***Graph Neural Networks: Foundations, Frontiers,and Applications***

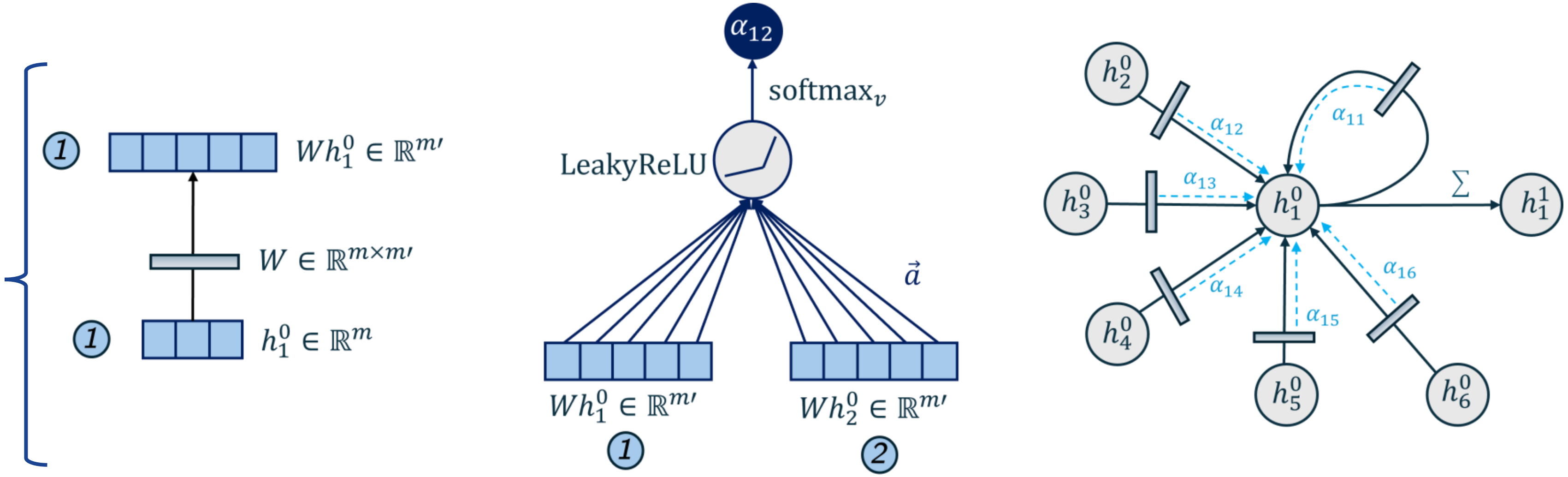
Message Passage Iteration: node representations h_u^k, at iteration k**,** **updated** by combining the node information with the **aggregated** message from the neighbouring nodes.

- **GCNs**: use the adjacency matrix A, to aggregate information of the neighbouring nodes.
- **GATs:** Attention mechanisms dynamically learn the importance of each neighbour node.

Attention Mechanisms:

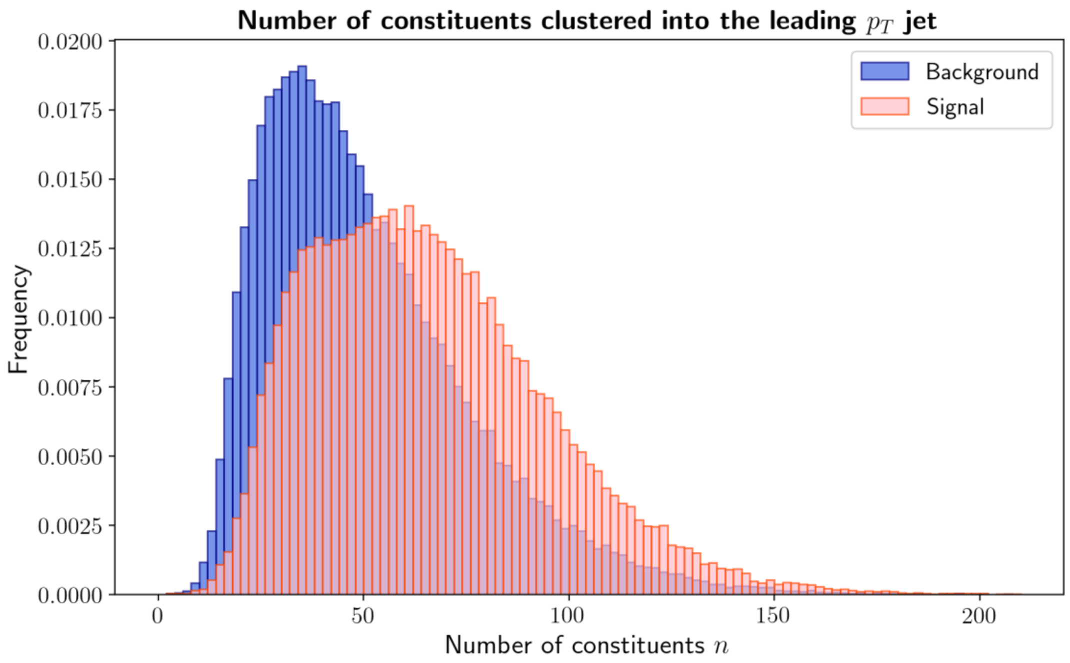
$$\alpha_{u,v} = \mathtt{softmax}_v\left(\mathrm{LeakyReLU}\left(a^\top [W h_u \oplus W h_v]\right)\right)$$

$$m_{\mathcal{N}(u)} = \sum_{v \in \mathcal{N}(u)} \alpha_{u,v} h_v$$



Pooling layer : transition from vector node representations to a graph level output

Creating Graph inputs

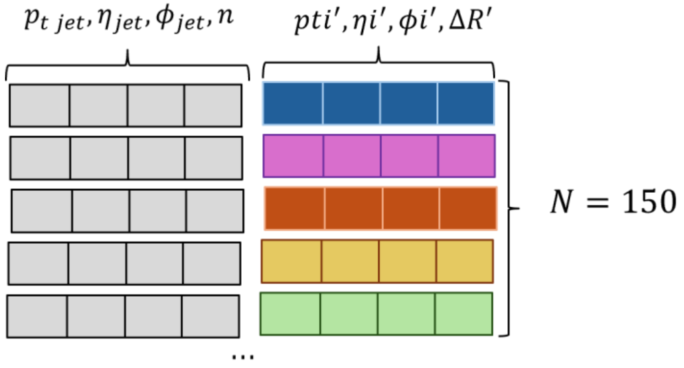


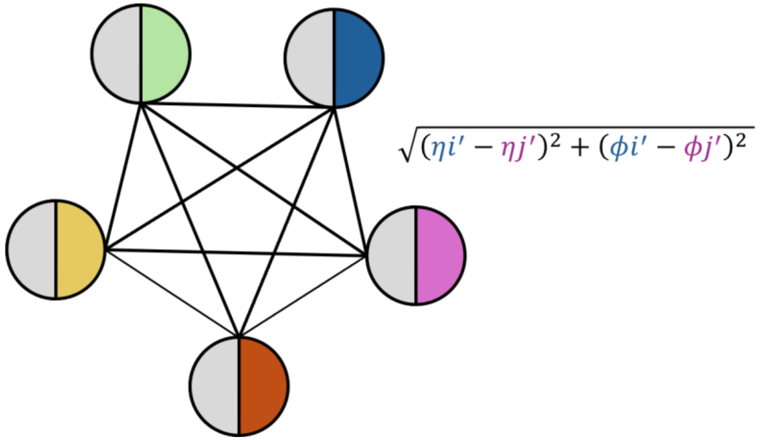
Considering the **most energetic 150 constituents** clustered into the jet after studying the effect of this choice on several leading jet distributions and m_{jj} distribution.

Adjacency matrix A and **node attribute matrix X** → edges and nodes embeddings

Adjacency matrix A entries → geometric distance between two constituents.

- Lorentz transformation to the jet rest frame, removing the dependence on ϕ_{jet}.
- Vector input with 604 entries (4 jet level features and 4 constituent features for 150 particles) → Input for baseline DNN and AD models .





Supervised Deep Learning Models

Using Tensorflow 2.6.2 and Keras API, ***Spektral*** library for graph layers

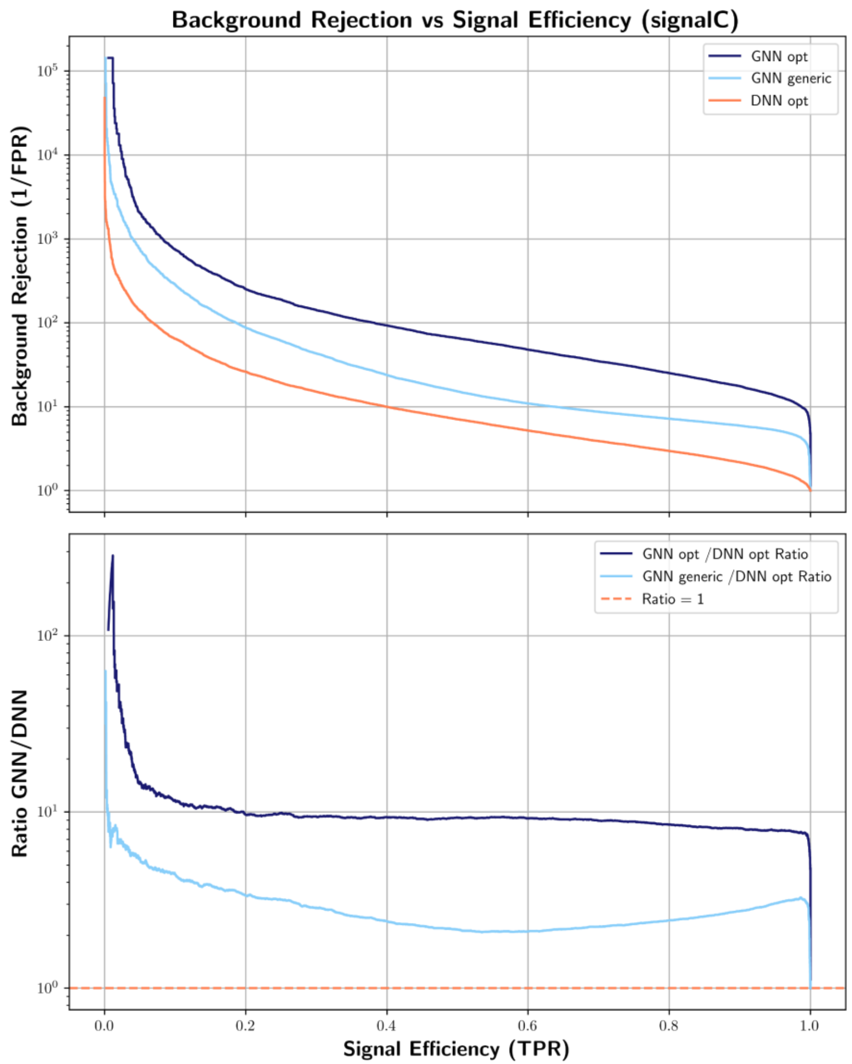
GNNs

- GAT layers, Global Attention Pooling Layer:

Optuna **optimization :** hyperparameter optimization maximizing the AUC score

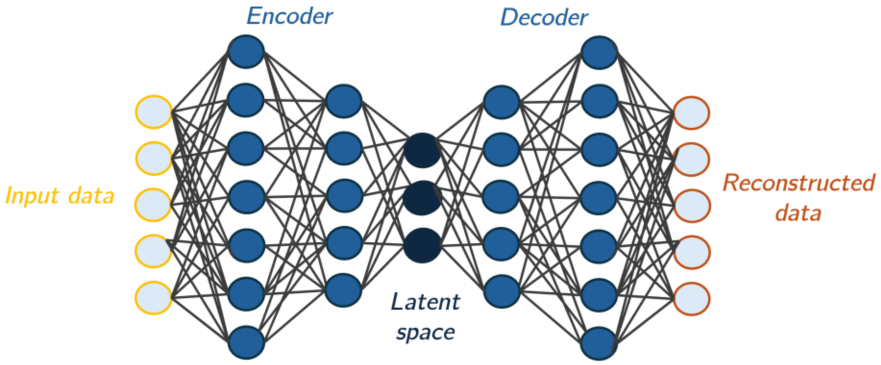
"Generic" GNN model: based on the best performance across all signals.
Single GNN architecture → hyperparameters not optimized for a specific signal type

Model	AUC
GNN Models	
Signal A optimized	0.99
Signal C optimized	0.98
Signal D optimized	0.95
Generic GNN	0.99, 0.92, 0.93
DNN Models	
Signal A optimized	0.93
Signal C optimized	0.80
Signal D optimized	0.77



- **Higher performance** for the graph-based models in terms of AUC.
- "Generic" GNN still outperforms optimized DNN.
- Highlight the **potential of representing jets as graphs** leveraging attention mechanisms to capture information about jet substructure

Both algorithms are **modular** allowing 2 different approaches:
- Operate on enriched embeddings generated by GNNs
- Directly process raw jet and constituents information.

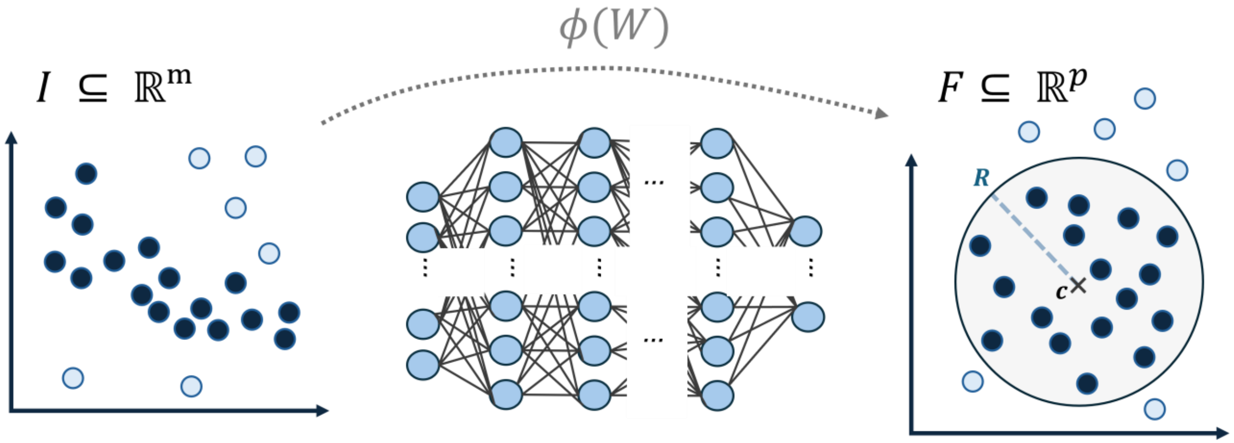


Autoencoders:

- Encoder-decoder structure
- Learns an approximate identity function
- Bottleneck layer compresses input data, limited number of hidden units
- Loss function measures the discrepancy between input and reconstructed output with MSE
  This loss is used as anomaly score

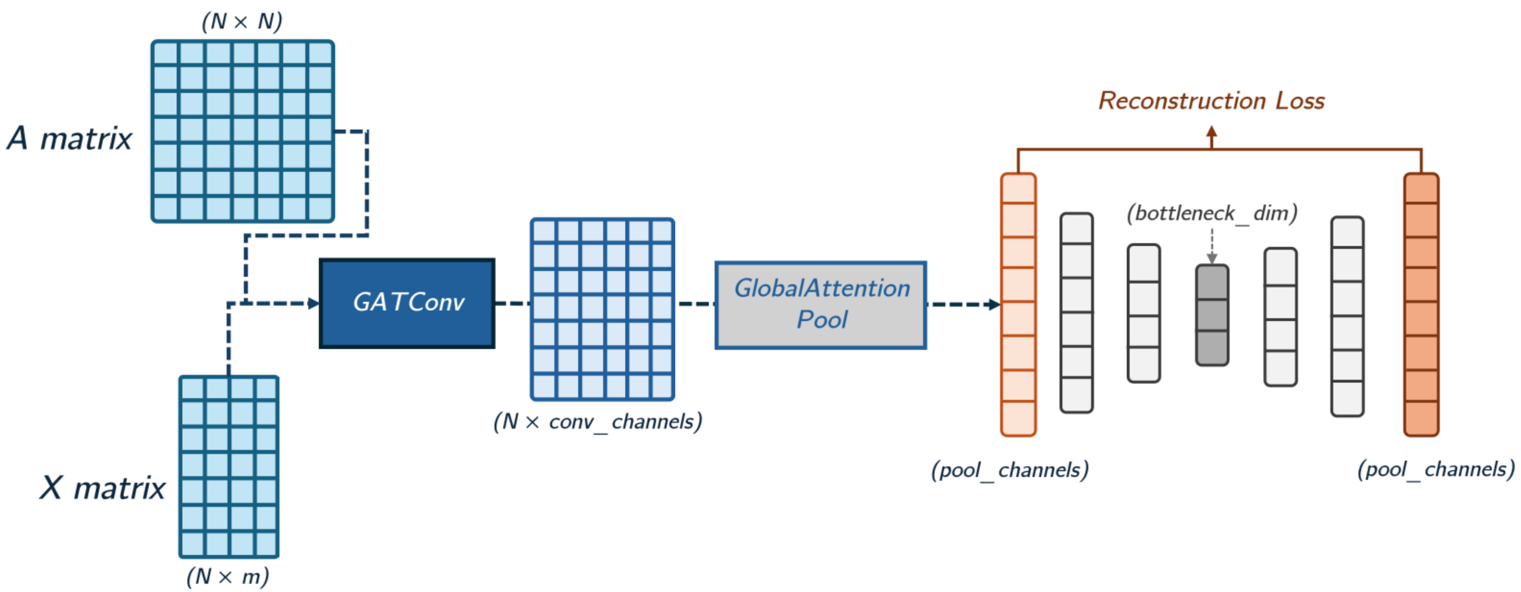
Deep Support Vector Data Description :

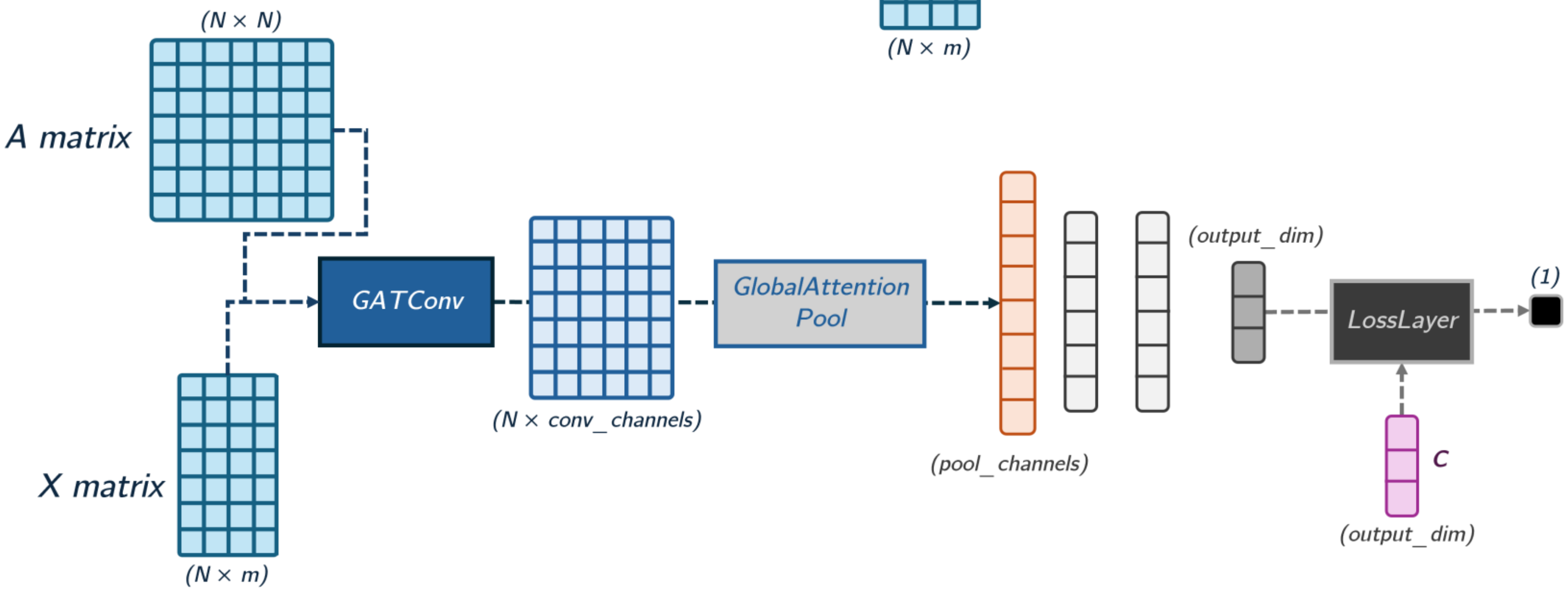
- Trains NN to minimize the volume of a hypersphere enclosing the data's representations
- Directly optimizes for AD by capturing common patterns and mapping data points near the sphere's centre.
- Distance of events in the output space to the sphere's centre is used as anomaly score



Scheme of DeepSVDD adapted from *Deep One-Class Classification*

GNN+AE



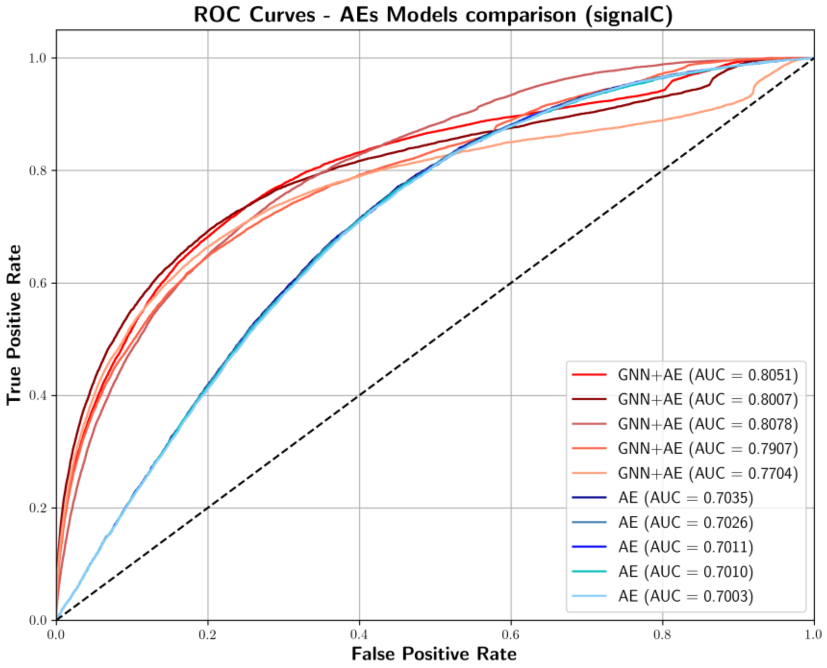


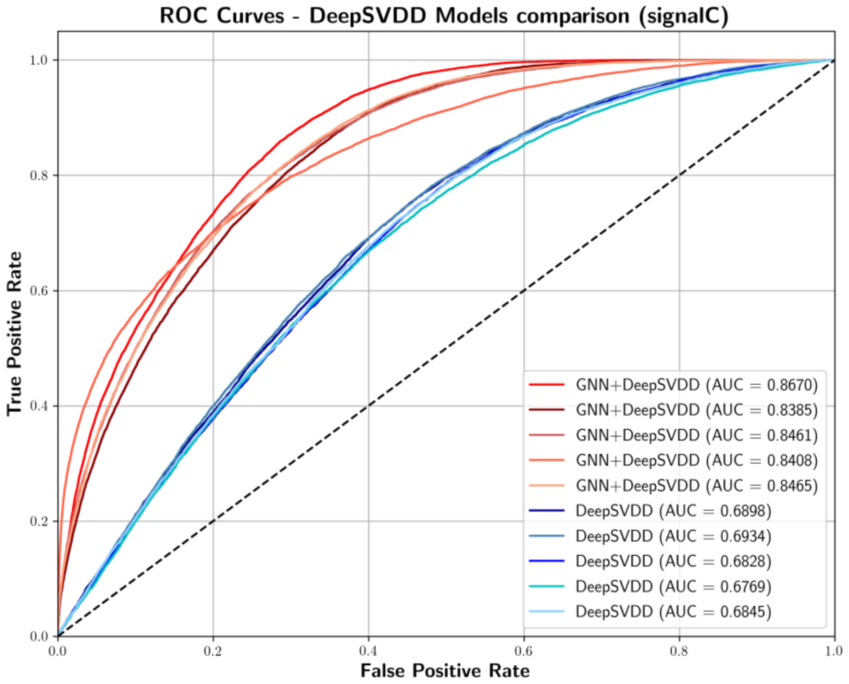
GNN+DeepSVDD

Performance Comparison: AD Algorithms on different Data Representations

- Train, val sets: only background.
- Test sets: background + specific type of signal.
- To assess how sensitive the performance of the models is on hyperparameters choices: Different combinations of hyperparameters were tested without any optimization strategy.

GNN based AD models outperform AEs and DeepSVDDs in terms of AUC

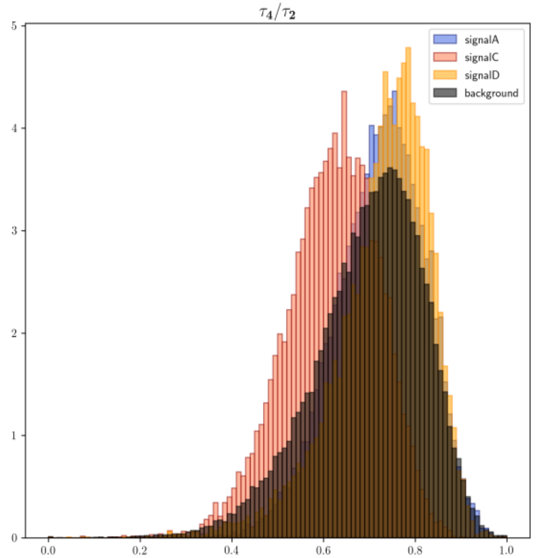


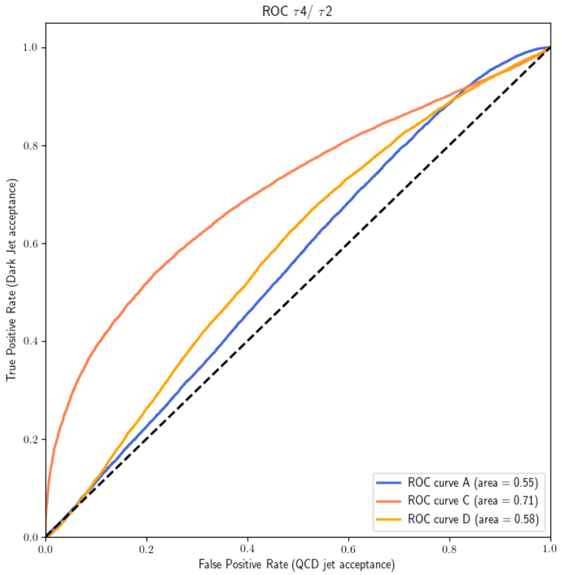


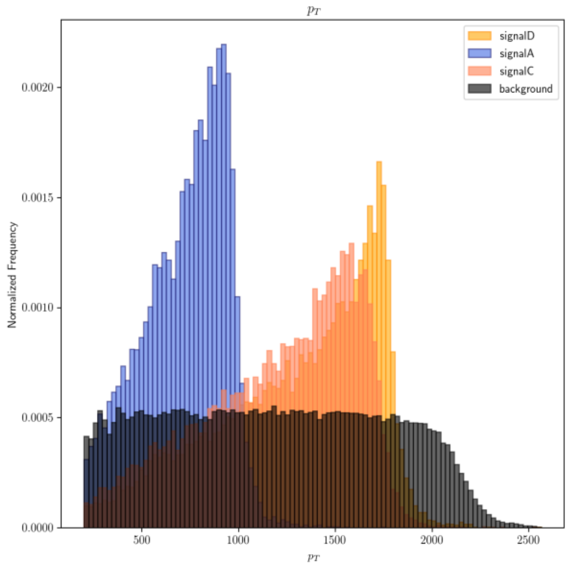
- All graph-based models have **only one graph convolution layer**. Increasing this value causes an **over-smoothing** problem
- The choice of hyperparameters and **architectural variations** have a **greater impact on GNN-based models** compared to AEs and DeepSVDDs
- GNN+DeepSVDDs show a **more robust** performance than GNN+AE models with higher AUC and no discontinuities in the ROC curves

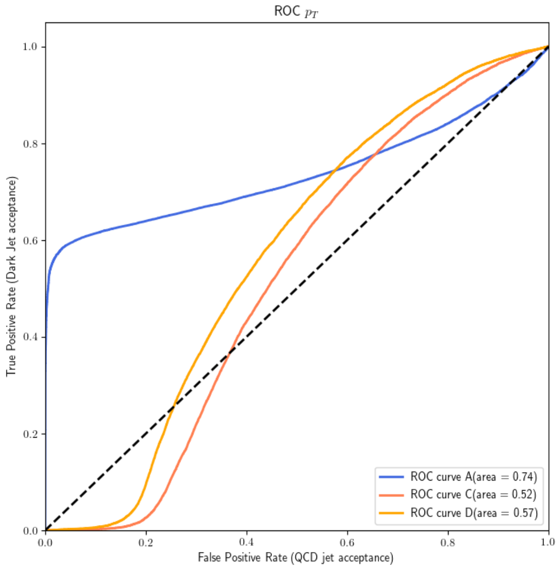
Performance Comparison of Deep Learning Methods with Classifiers Based on Discriminant Jet Features

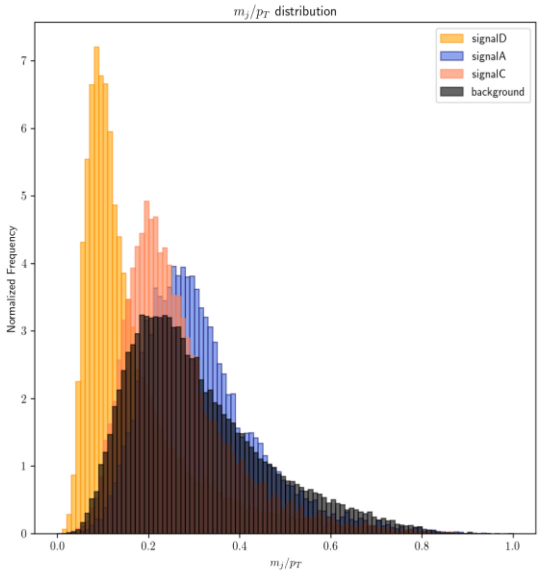
- Feature-based classifiers – interpretable alternatives to deep learning models.
- Graph-based model-agnostic models outperform these classifiers
- Jet subscruture features isolate a single signal

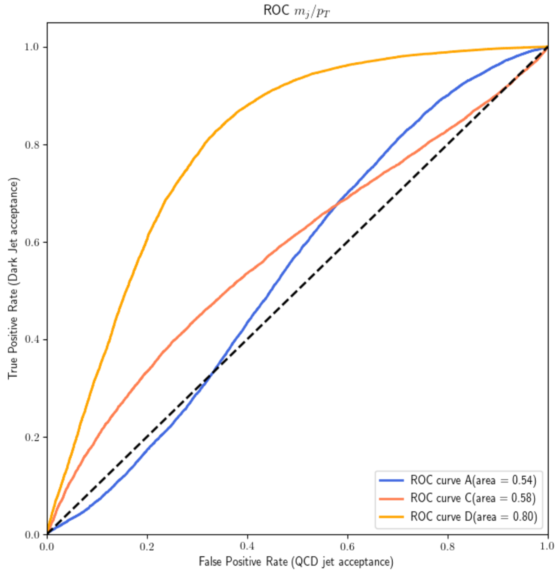












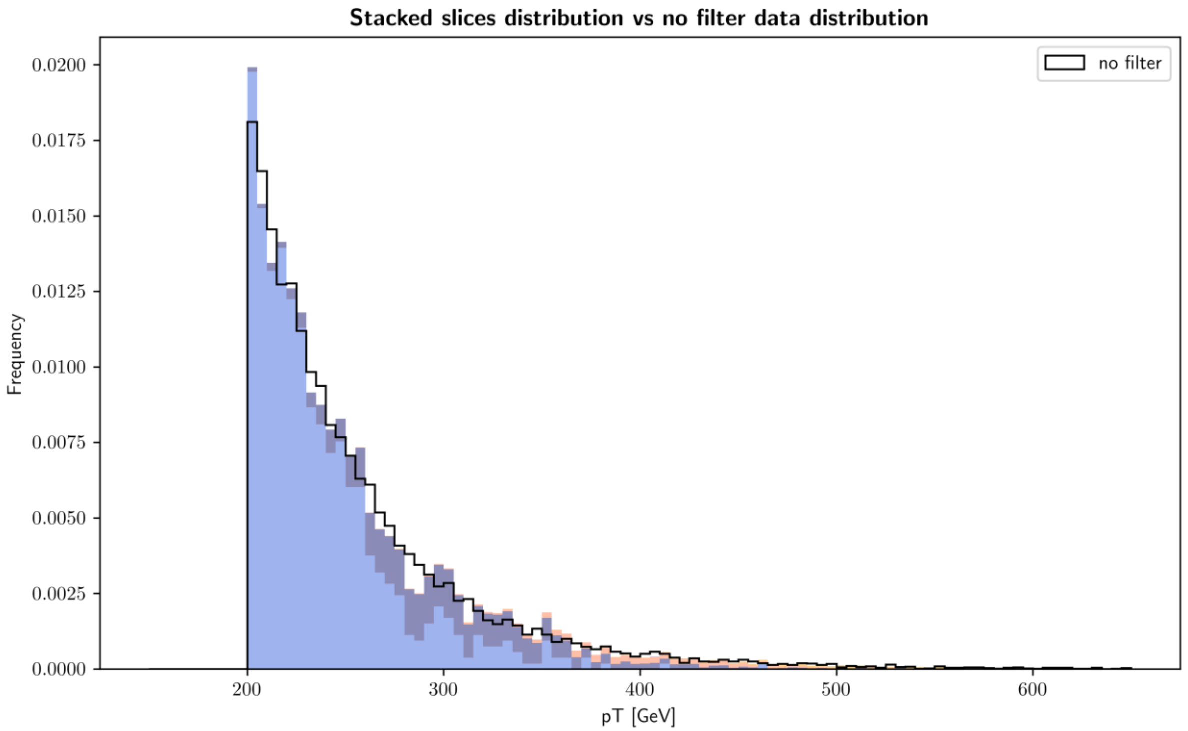
Conclusions

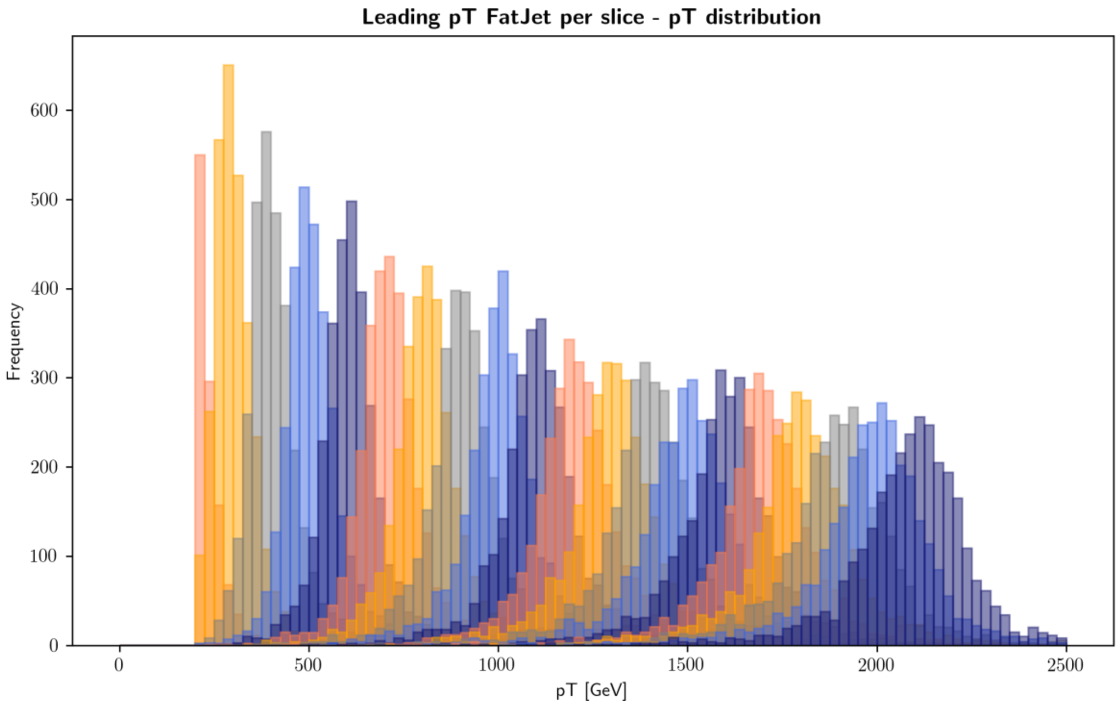
- Graphs capture structural information of jets that models receiving vector inputs do not.
- The attention mechanisms extract a vector embedding with discriminative features.
- Attempting to input the relational weights between constituents in vector input would increase complexity + number of trainable parameters

Thank you for your attention!

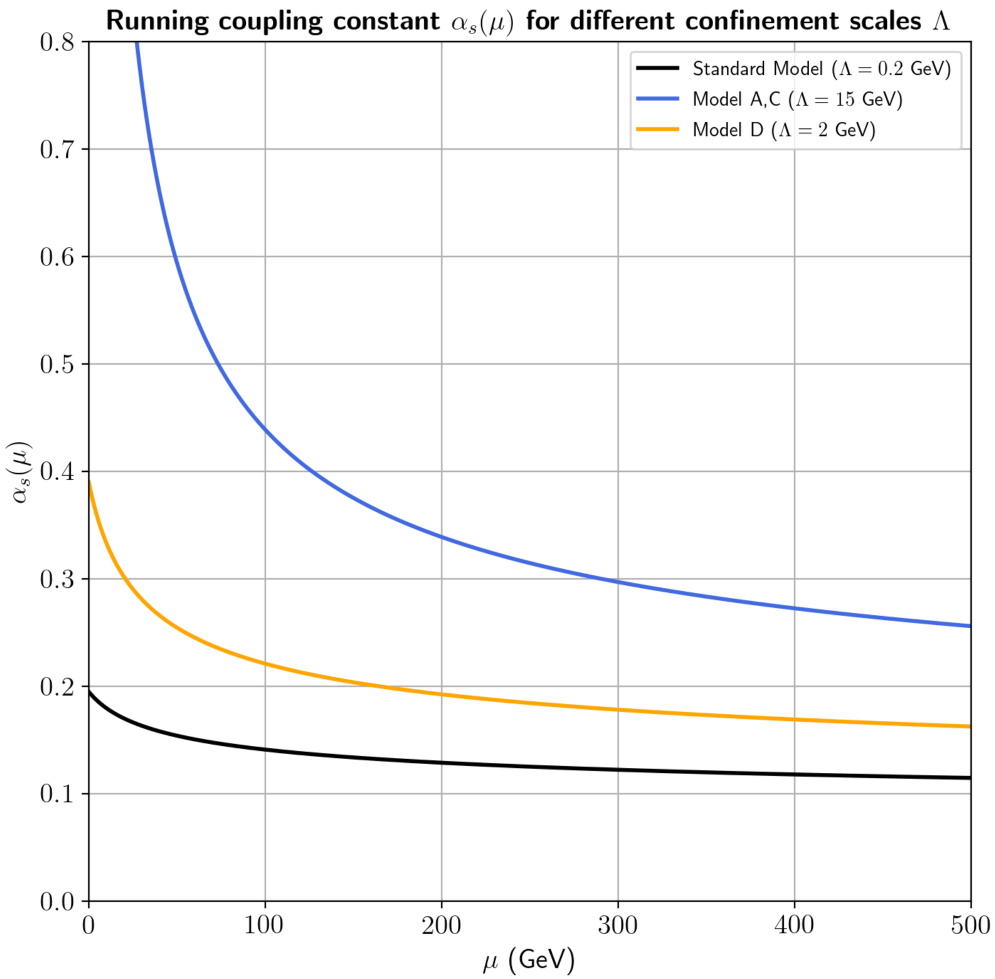
Back-Up Slides

Background production in slices



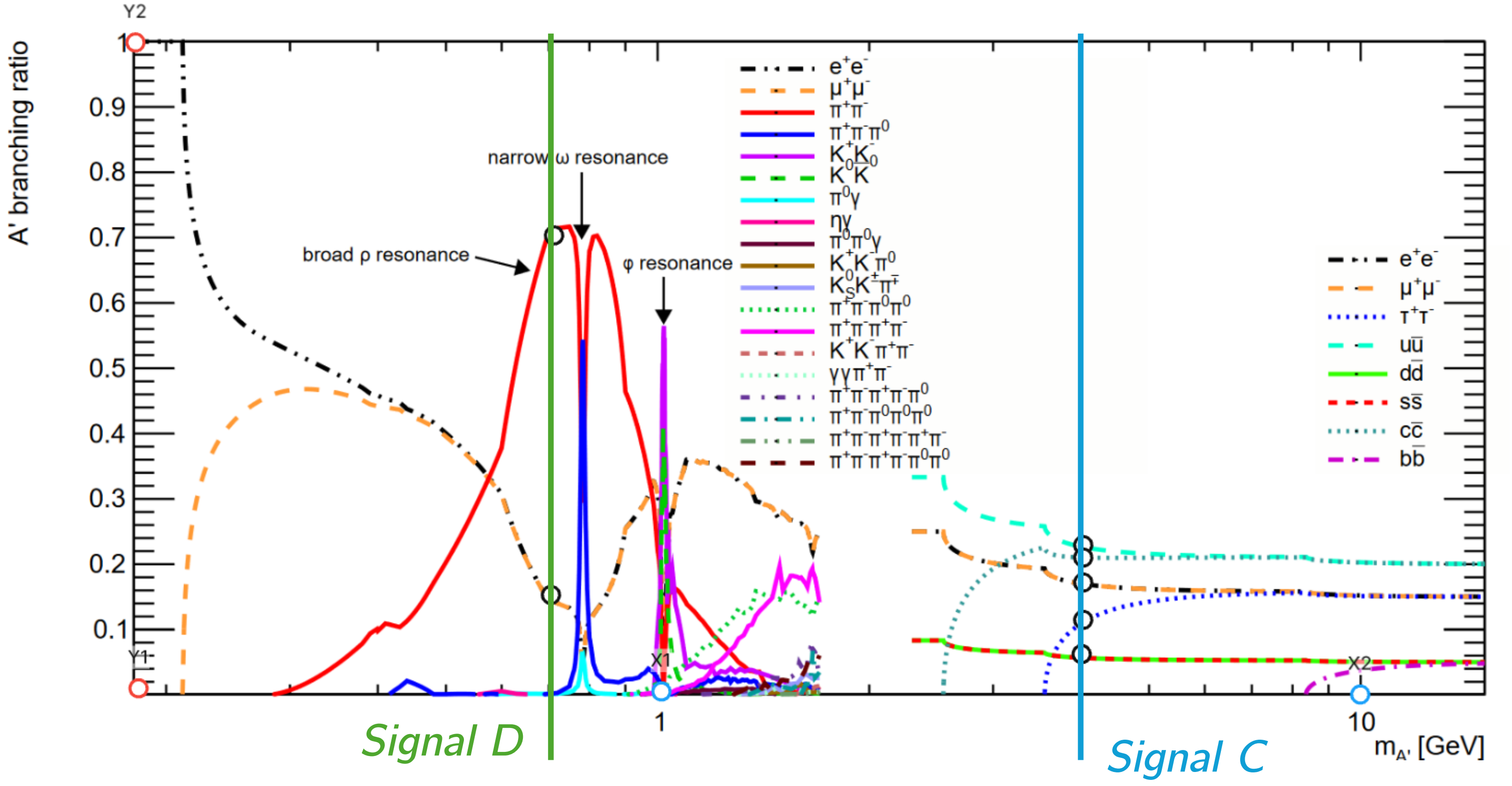


Running coupling α_s for different confinement scales

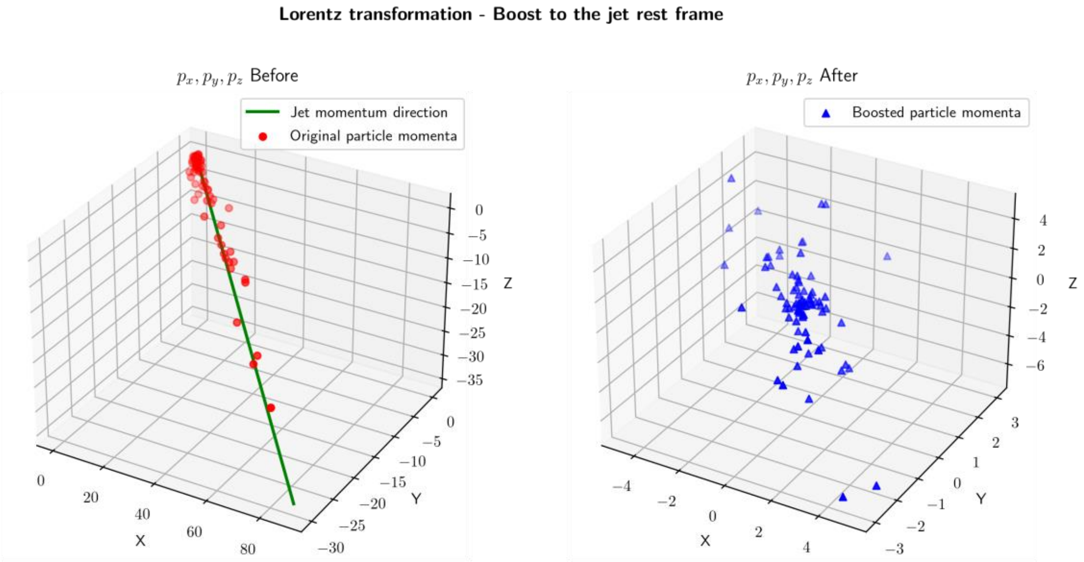


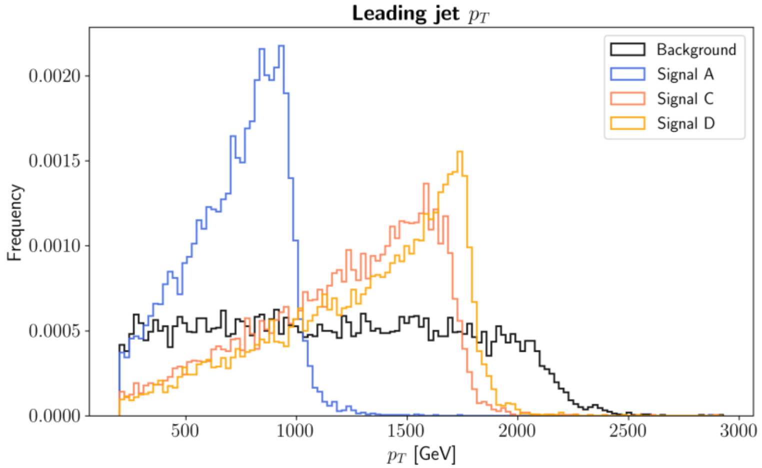
α_s running coupling in dark sector and SM QCD running coupling

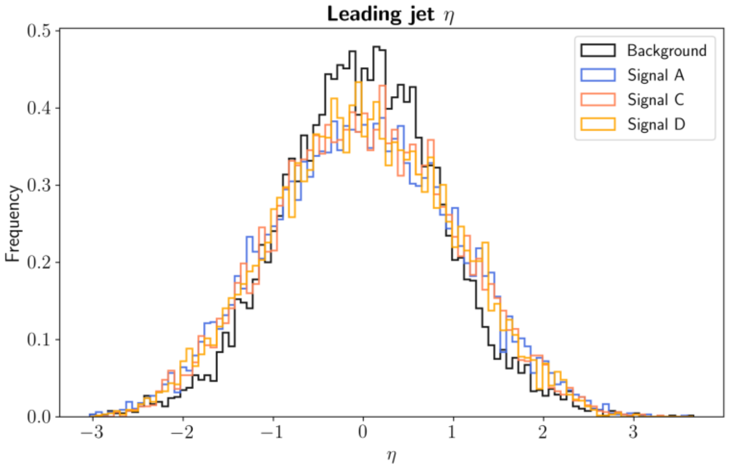
Branching Ratios of dark γ' decays for different $m_{\gamma'}$

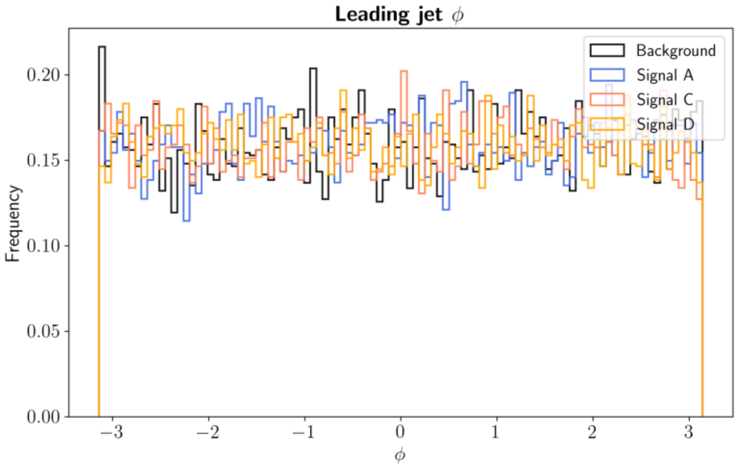


Leading p_T features and Lorentz transformation to jet rest frame example

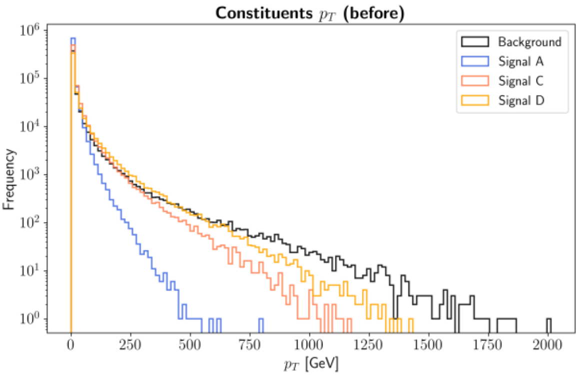


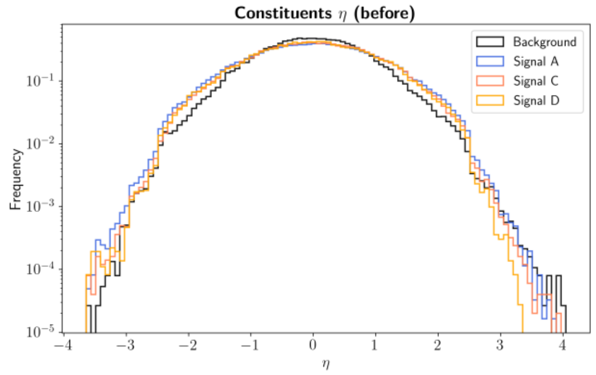


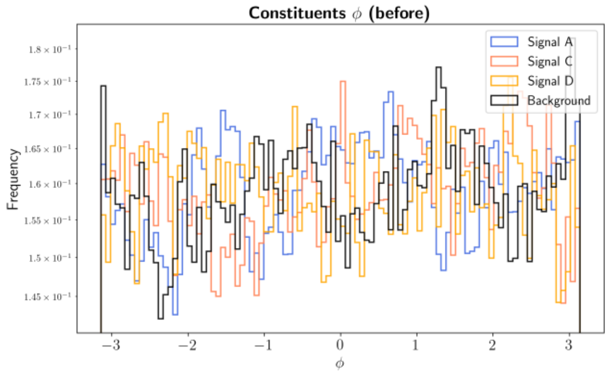


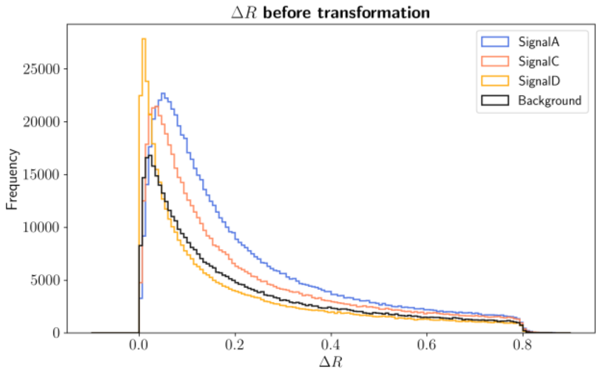


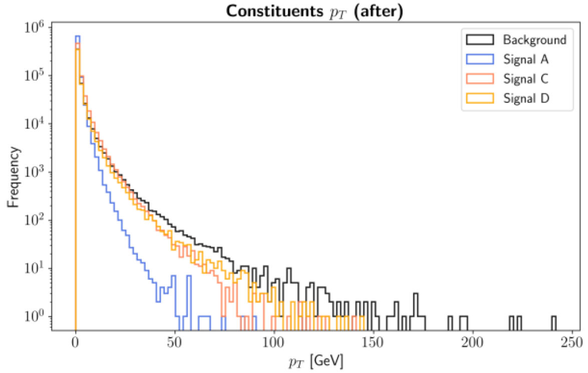
Constituent Level Features before and after the Lorentz Transformation

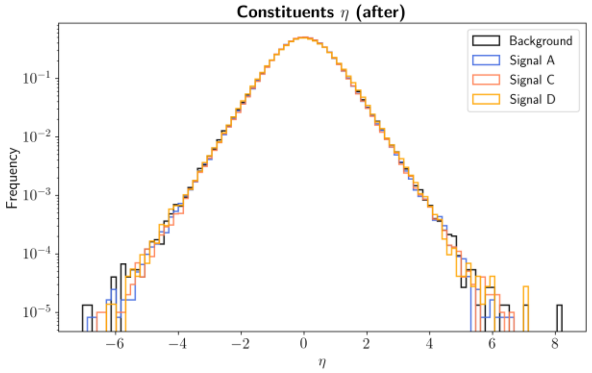


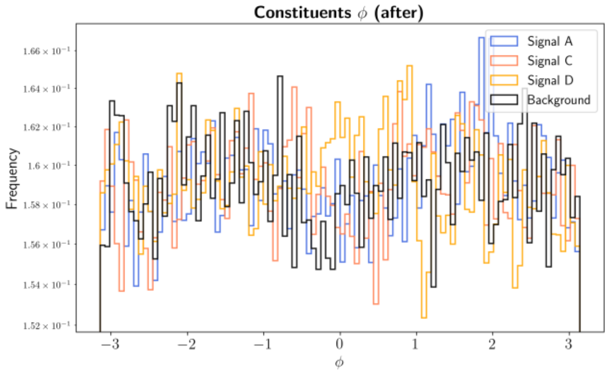


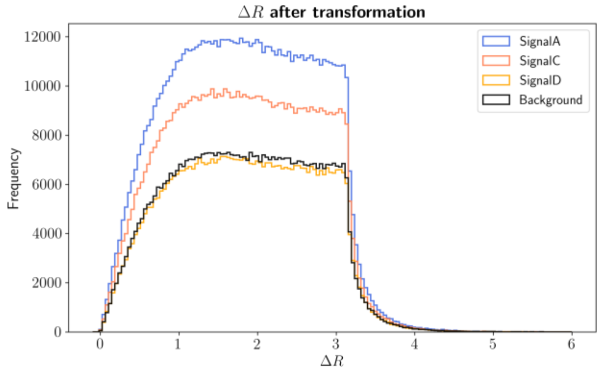




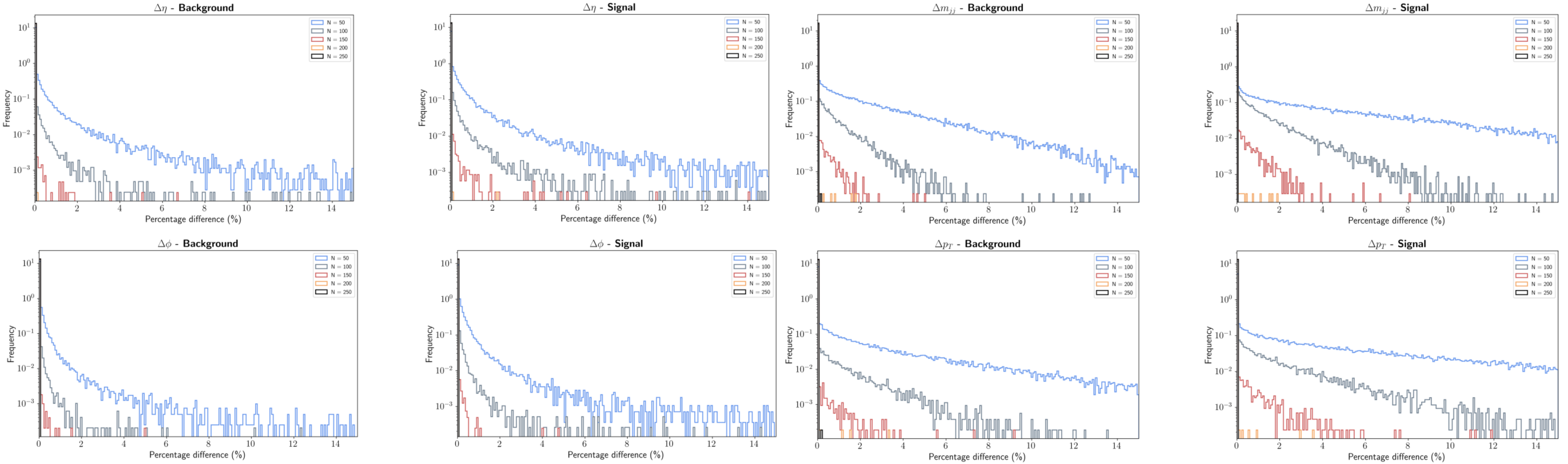




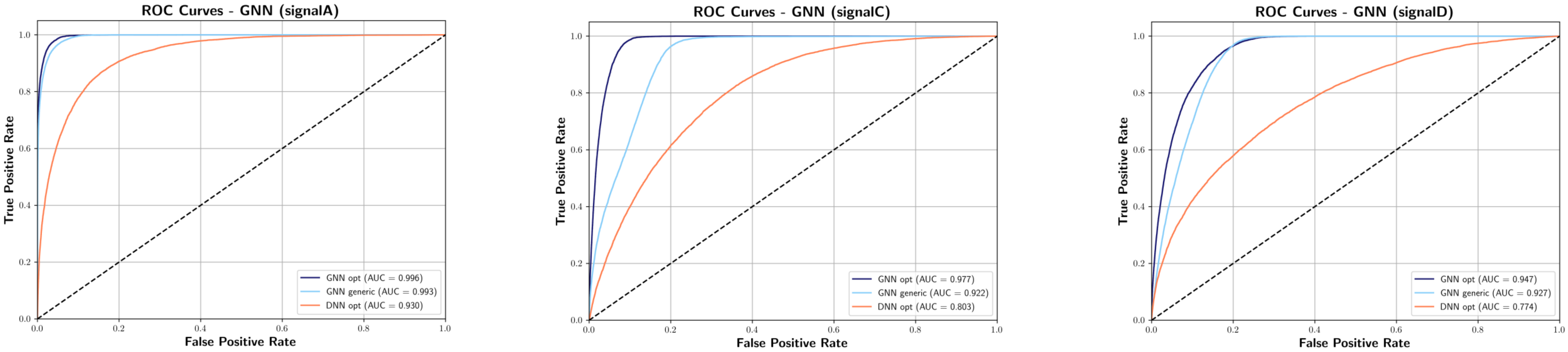




Percentage differences in jet features and m_{jj} considering the N most energetic constituents



Supervised Learning Models Results



model	attn_heads	n_conv_layers	channels_conv	channels_pool	n_dense_neurons	n_dense_layers	learning_rate	AUC
Signal A optimized	4	2	17	5	104	1	6.694e-04	0.996
Signal C optimized	3	2	28	75	7	4	9.541e-04	0.976
Signal D optimized	4	1	13	21	14	3	7.1065e-06	0.946
generic	2	2	15	100	35	3	1.000e-05	0.993, 0.920, 0.926

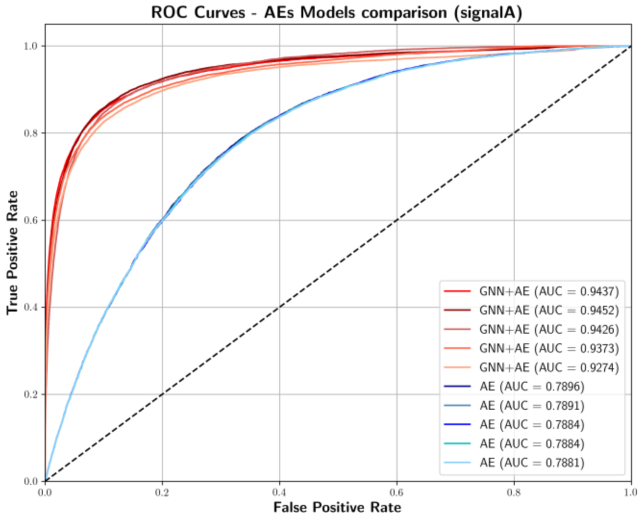
Table 1.1: Hyperparameters of Supervised GNN models

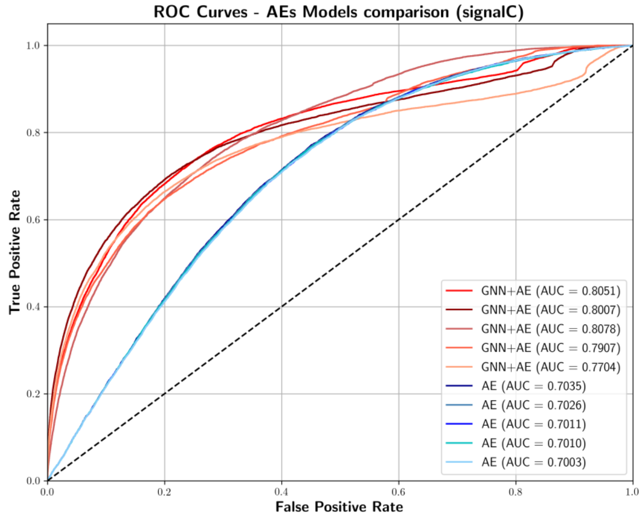
model	n_dense_neurons	n_dense_layers	learning_rate	AUC
Signal A optimized	5	1	4.848e-05	0.930
Signal C optimized	9	2	6.254e-04	0.803
Signal D optimized	37	4	5.797e-04	0.774

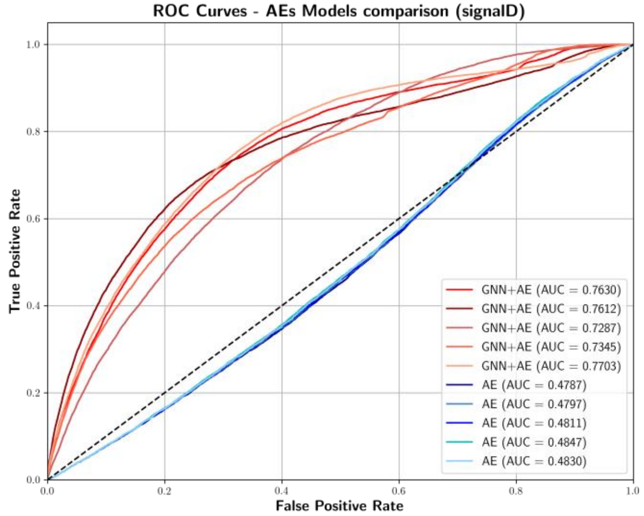
Table 1.2: Hyperparameters of Supervised DNN models

$$z_{\mathcal{G}} = \sum_{u \in \mathcal{V}} \sigma(W_1 z_u + b_1) \odot (W_2 z_u + b_2))$$

AD models : GNN+AE vs. AE







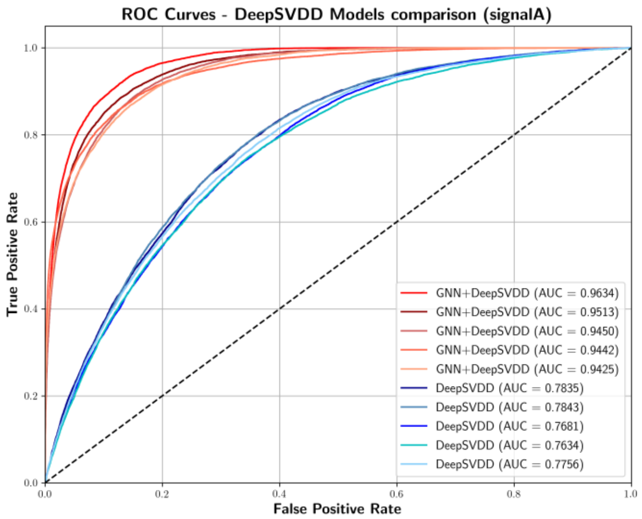
model	attn_heads	n_conv_layers	channels_conv	channels_pool	n_encoder_layers	bottleneck_dim	AUC (Signal A)	AUC (Signal C)	AUC (Signal D)
3	1	1	15	250	3	62	0.943	0.808	0.729
9	2	1	15	200	1	60	0.945	0.801	0.761
10	1	1	15	300	3	75	0.944	0.805	0.763
12	2	1	25	200	2	55	0.927	0.770	0.770
20	1	1	15	300	3	80	0.937	0.791	0.735

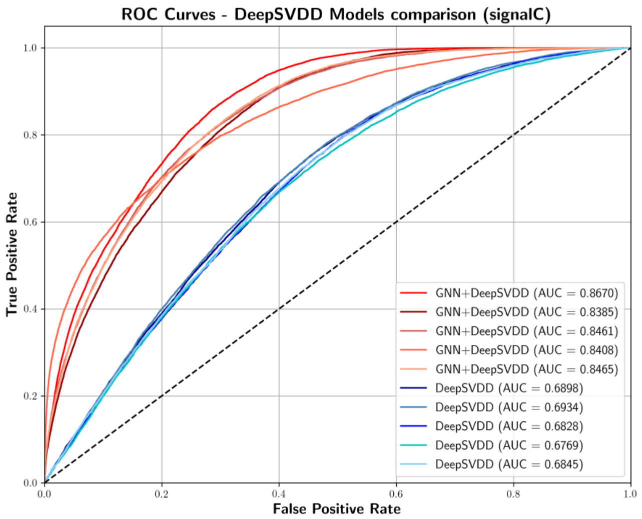
Table 1.3: Hyperparameters of GNN+AEs models that have the best performance evaluated across different signals

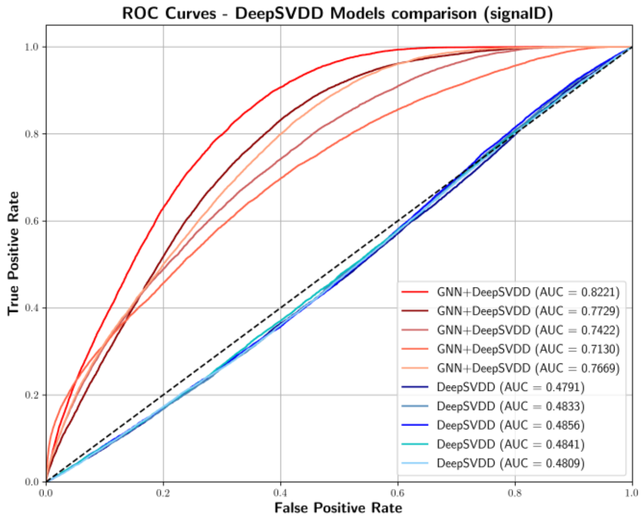
model	n_neurons_dense	n_encoder_layers	bottleneck_dim	AUC (Signal A)	AUC (Signal C)	AUC (Signal D)
1	600	2	150	0.789	0.704	0.479
2	500	2	125	0.754	0.789	0.483
3	400	1	100	0.788	0.700	0.484
9	10	2	5	0.788	0.701	0.485
20	400	2	100	0.788	0.701	0.481

Table 1.4: Hyperparameters of AE models that have the best performance evaluated across different signals

AD models : GNN+DeepSVDD vs. DeepSVDD







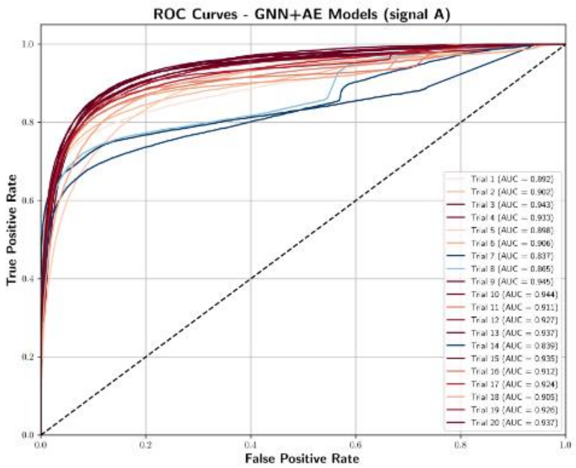
model	attn_heads	n_conv_layers	channels_conv	channels_pool	n_dense_neurons	n_dense_layers	output_dim	AUC (Signal A)	AUC (Signal C)	AUC (Signal D)
1	1	1	15	100	64	2	32	0.963	0.867	0.822
5	1	1	30	200	80	3	64	0.945	0.835	0.736
9	1	1	50	150	100	3	75	0.955	0.836	0.729
10	1	1	25	100	64	4	64	0.956	0.852	0.798
15	3	1	15	100	64	2	64	0.951	0.839	0.773

Table 1.5: Hyperparameters of GNN+DeepSVDD models that have the best performance evaluated across different signals

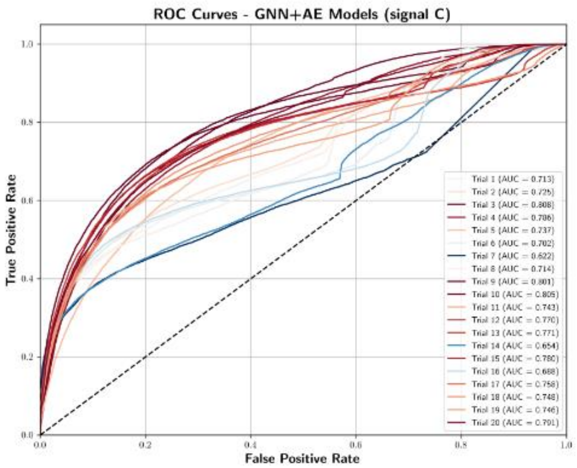
model	n_neurons_dense	n_encoder_layers	output_dim	AUC (Signal A)	AUC (Signal C)	AUC (Signal D)
5	80	2	64	0.783	0.690	0.479
8	120	2	64	0.783	0.693	0.483
11	150	2	32	0.768	0.683	0.483
13	150	3	64	0.763	0.677	0.484
18	100	2	16	0.776	0.685	0.481

Table 1.6: Hyperparameters of DeepSVDD models that have the best performance evaluated across different signals

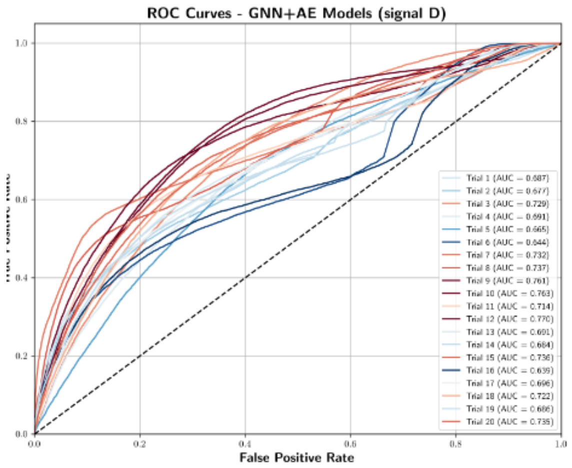
AD models : Different Hyperparameter combinations for AEs



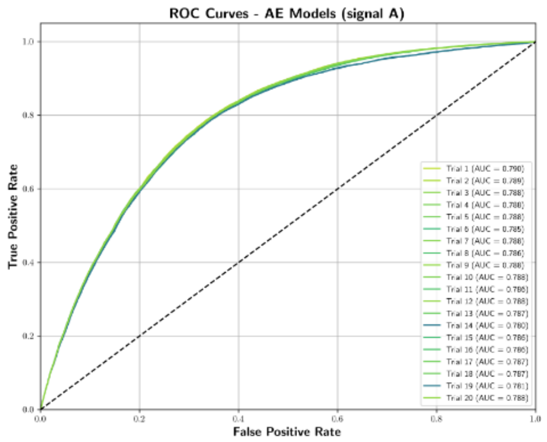
(a) ROC Curve GNNs+AEs signal A



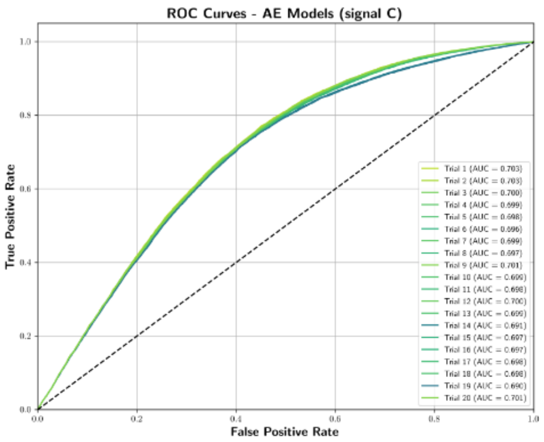
(b) ROC Curve GNNs+AEs signal C



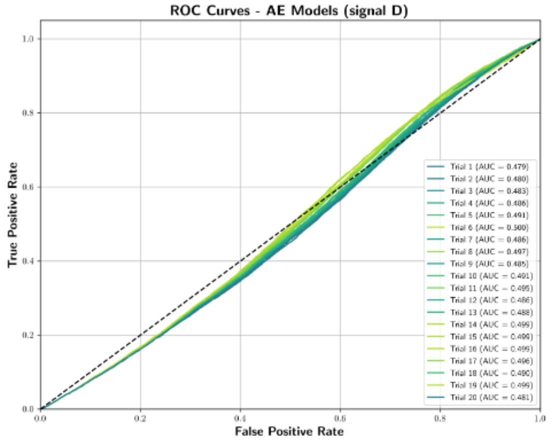
(c) ROC Curve GNNs+AEs



(a) ROC Curve AEs signal A

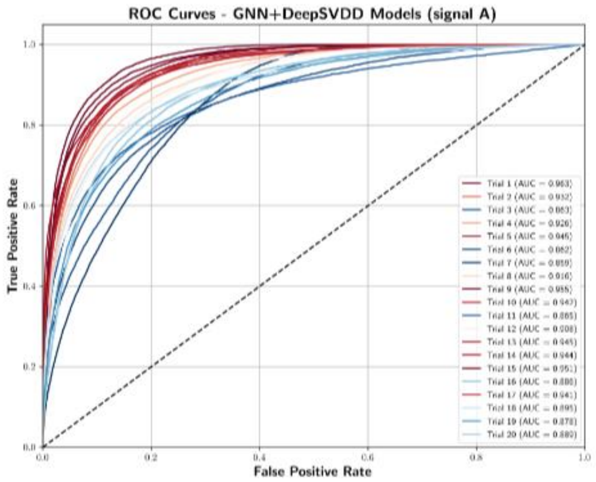


(b) ROC Curve AEs signal C

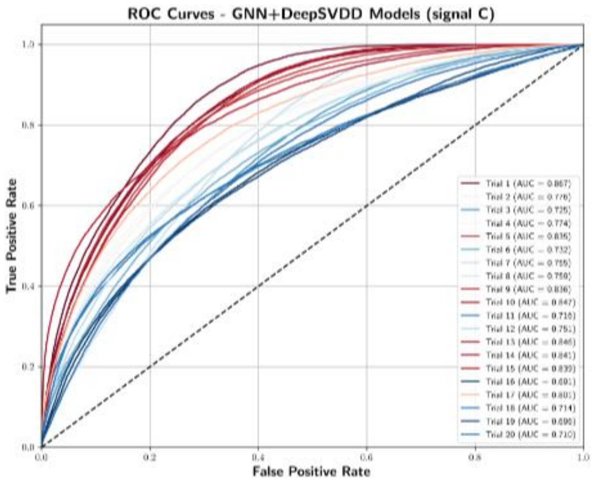


(c) ROC Curve AEs signal D

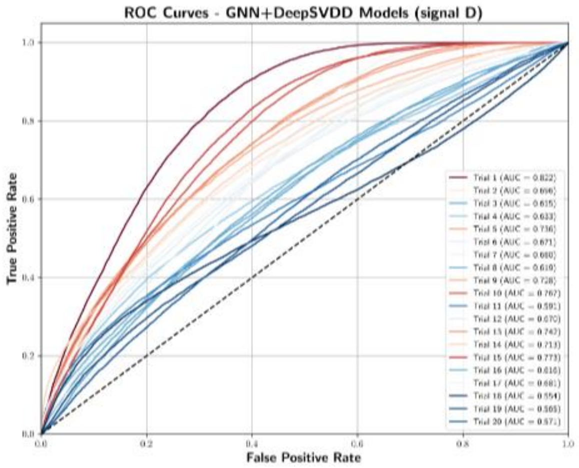
AD models : Different Hyperparameter combinations for DeepSVDDs



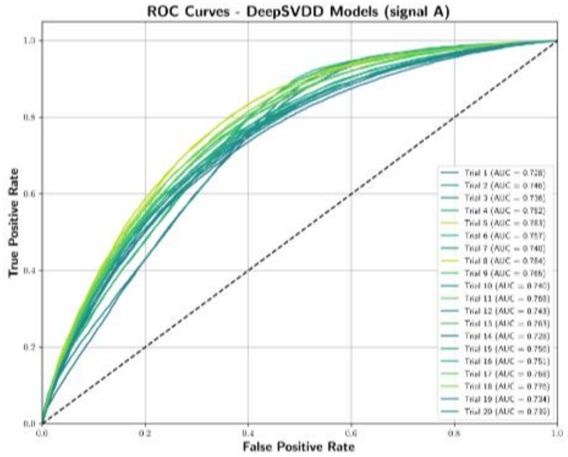
(a) ROC Curve GNNs+DeepSVDD signal A



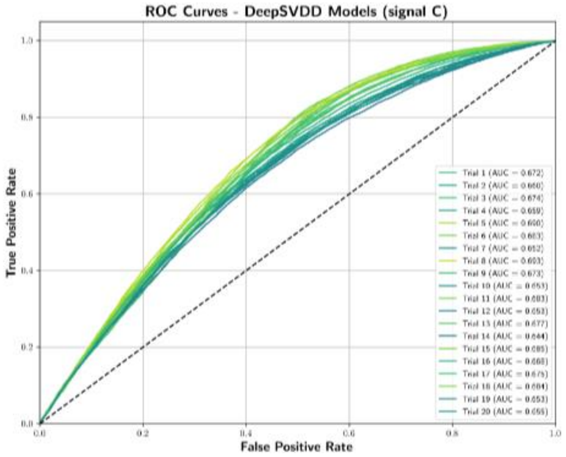
(b) ROC Curve GNNs+DeepSVDD signal C



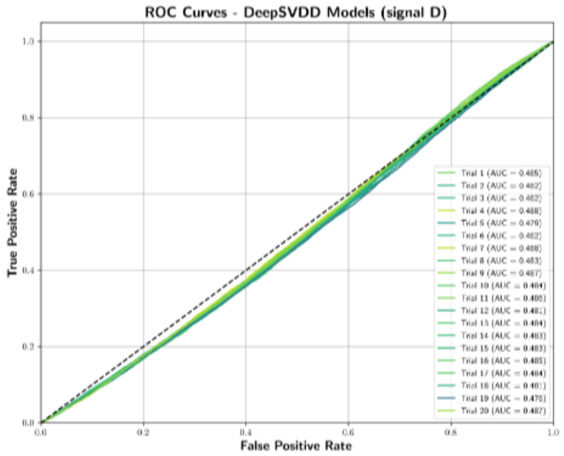
(c) ROC Curve GNNs+DeepSVDD



(a) ROC Curve DeepSVDDs signal A

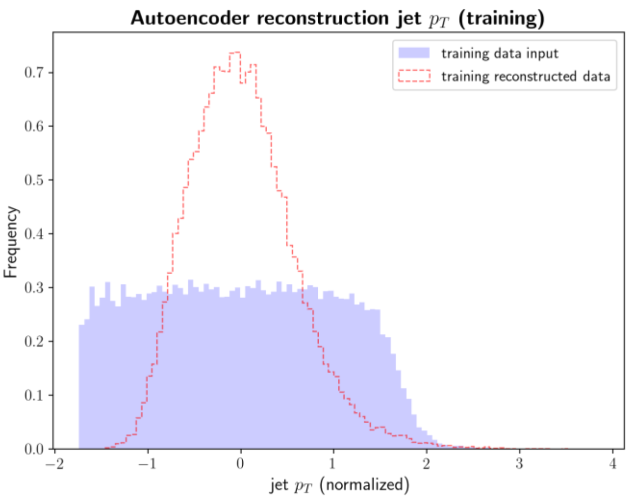


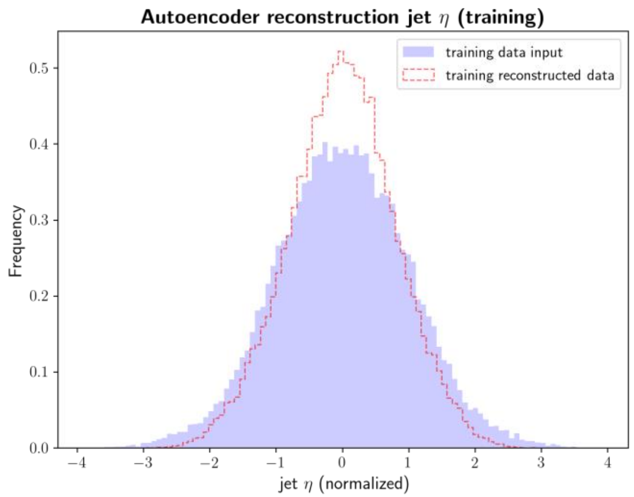
(b) ROC Curve DeepSVDDs signal C

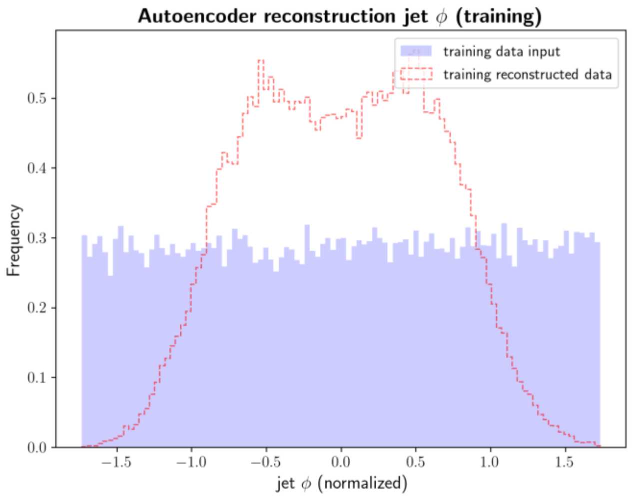


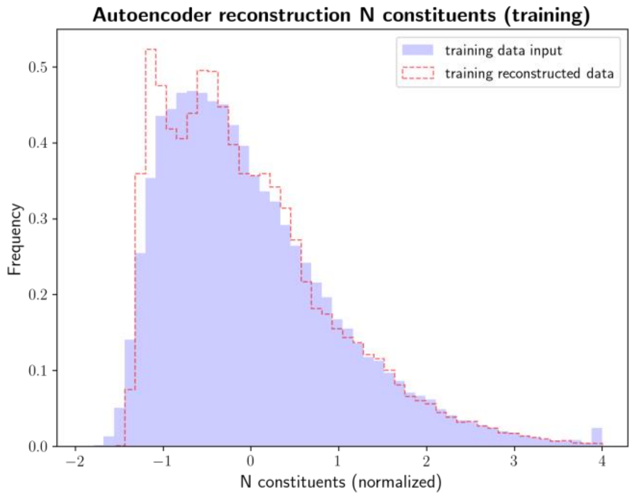
(c) ROC Curve DeepSVDDs signal D

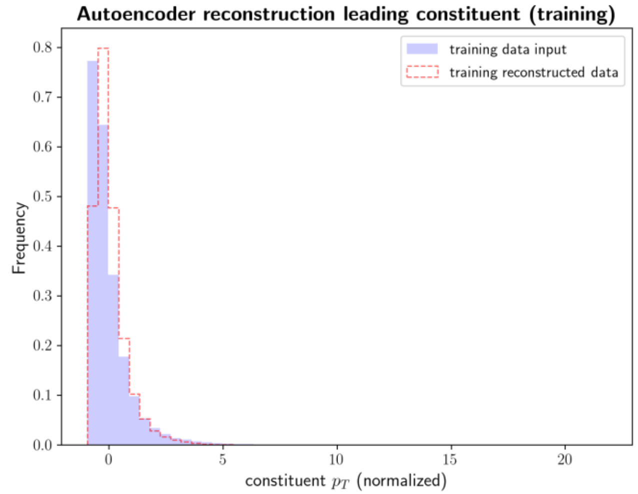
Normal Autoencoder reconstruction (Jet and leading pT constituent features)

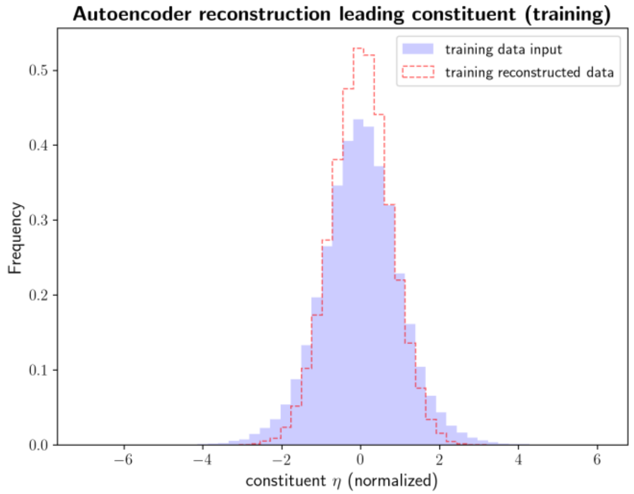


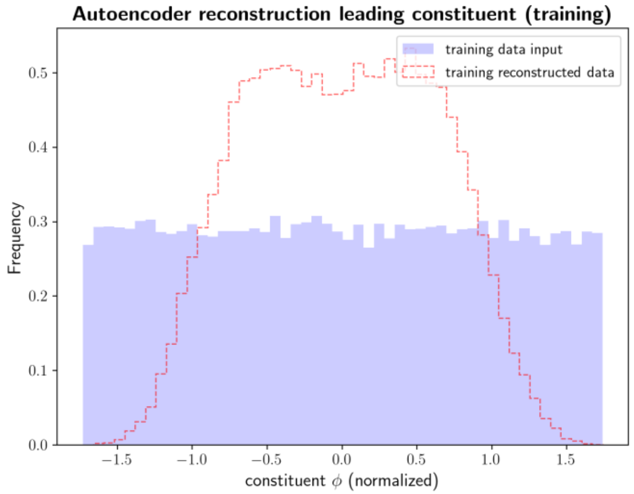


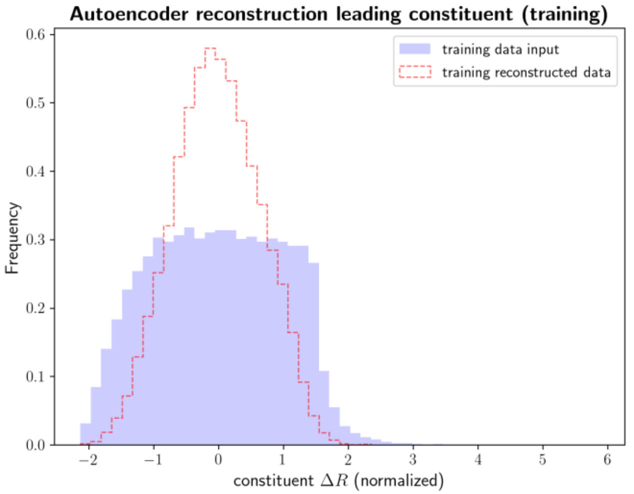




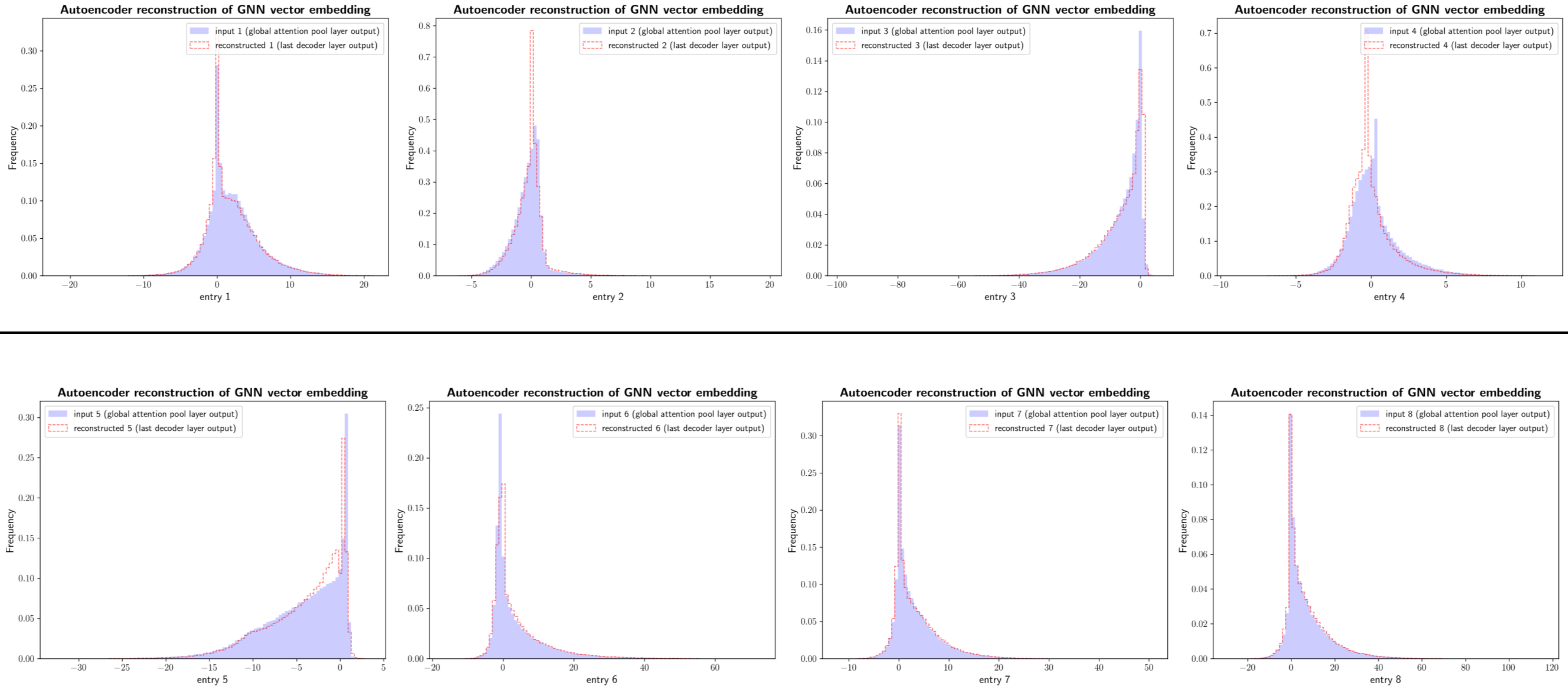






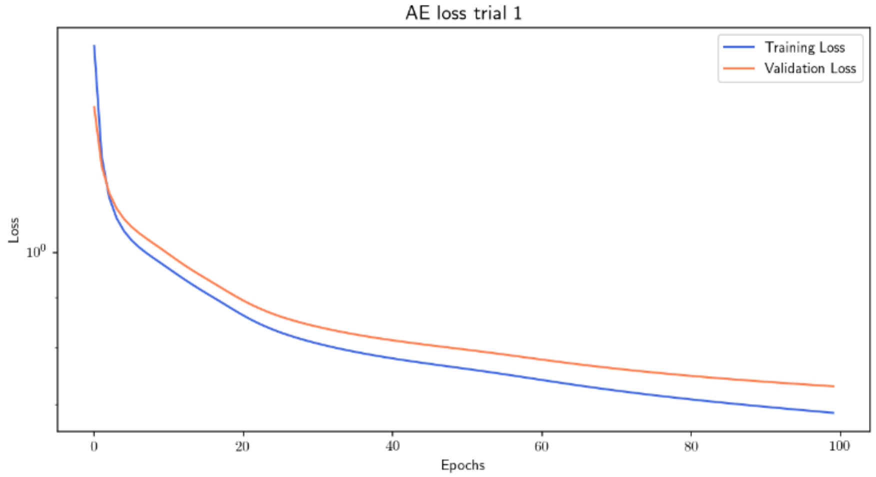


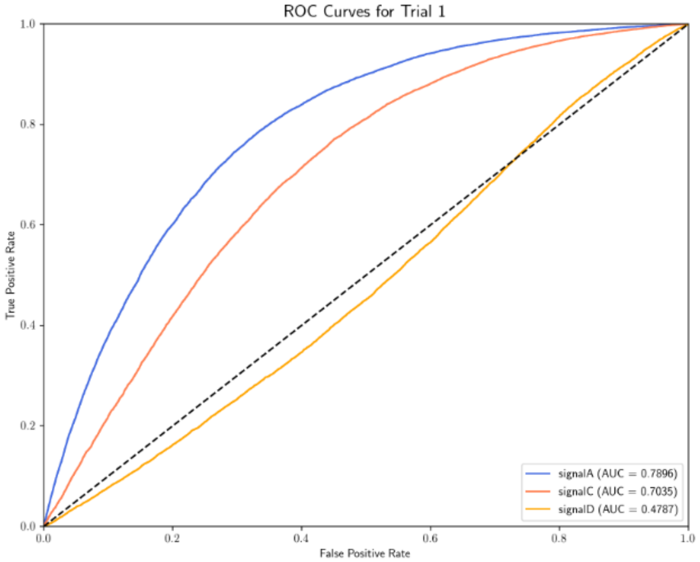
GNN Autoencoder reconstruction (First 8 entries of graph pooling layer output)



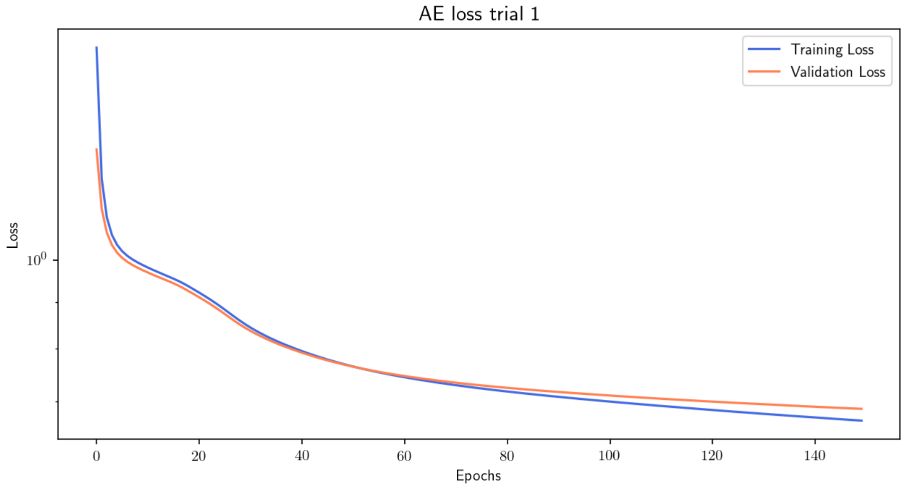
Normal Autoencoder performance receiving (p_T, η, ϕ) vs (p_x, p_y, p_z)

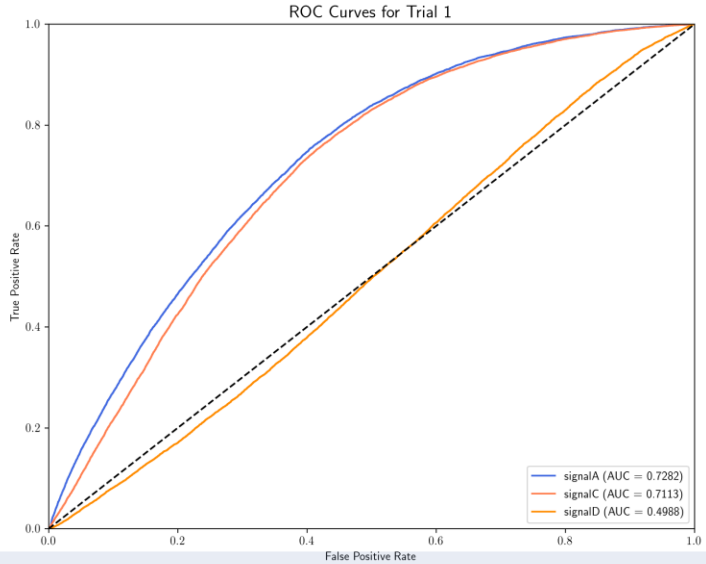
Angular



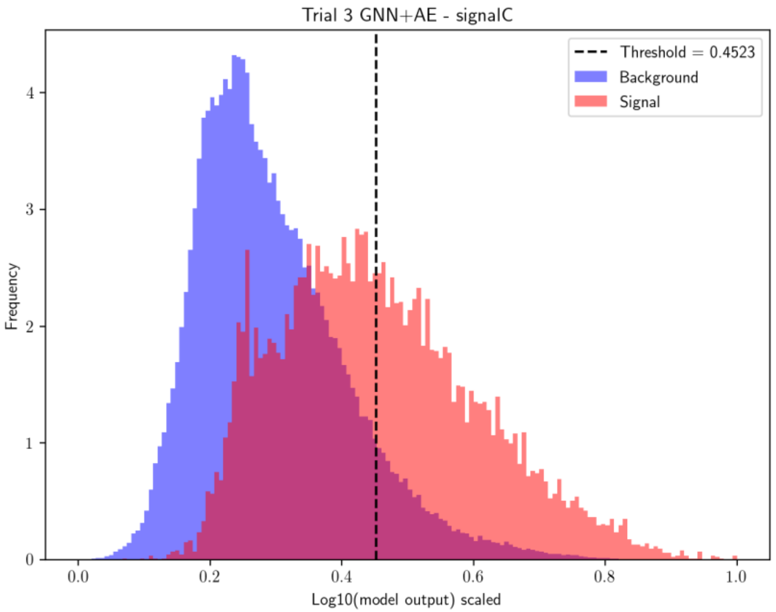


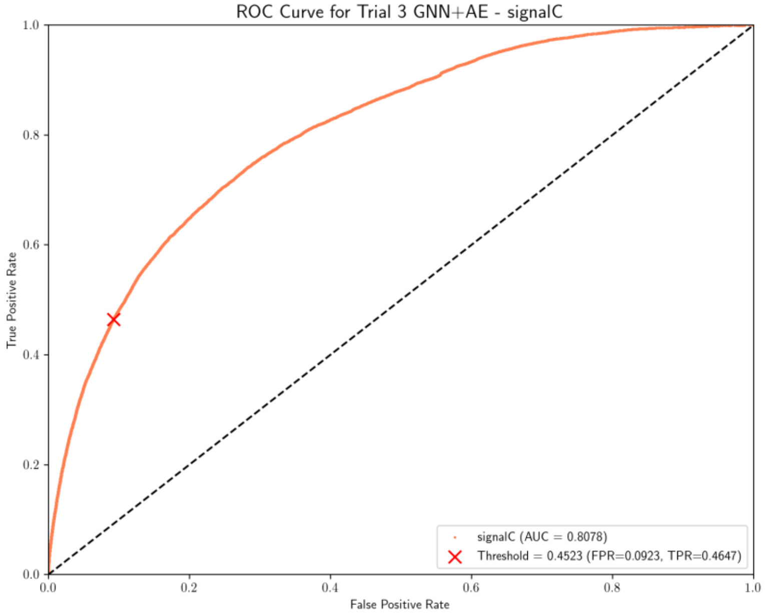
Cartesians

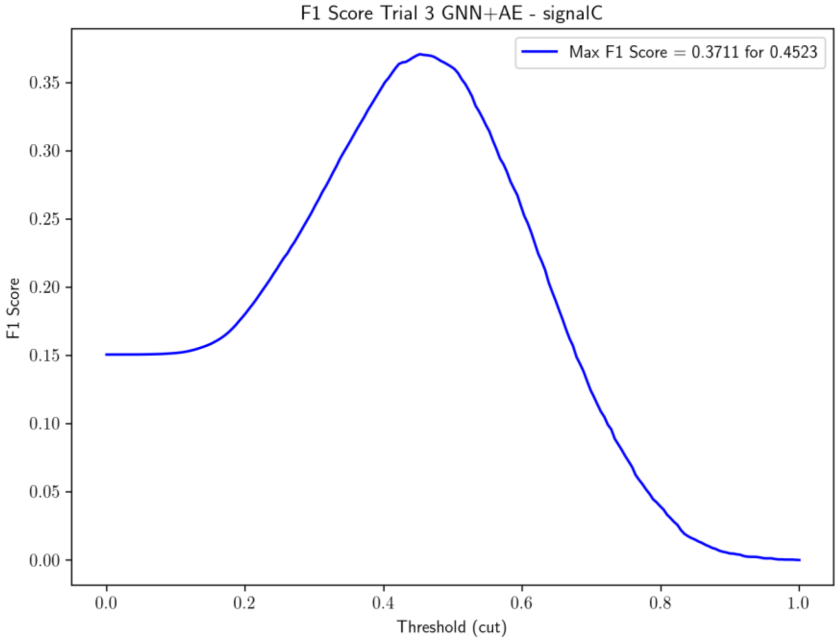


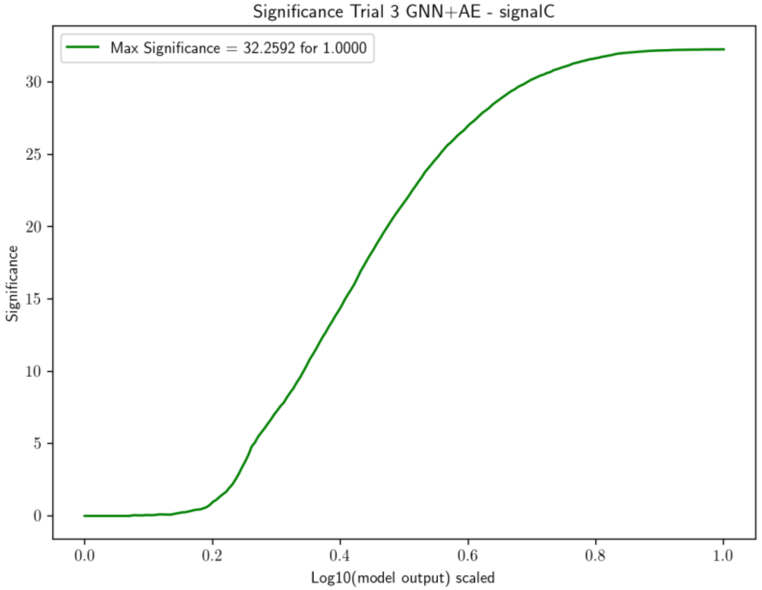


Finding the best working point for AD models as classifiers:

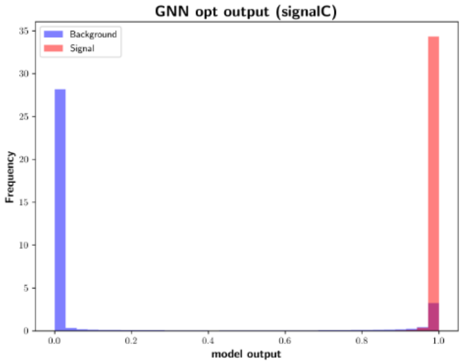




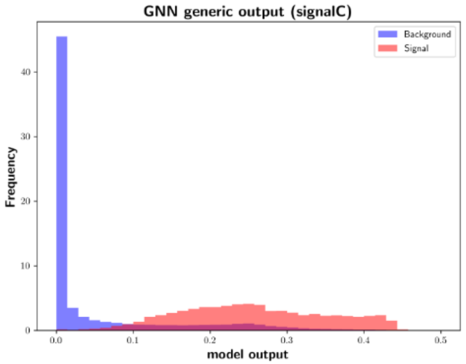




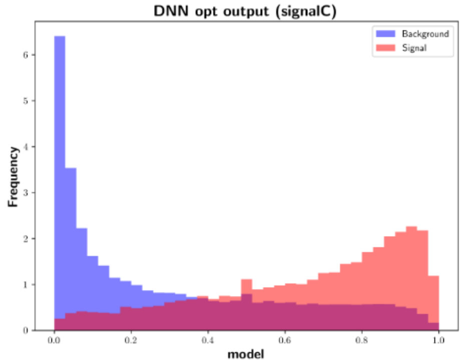
Outputs of Deep Learning Models for signal C



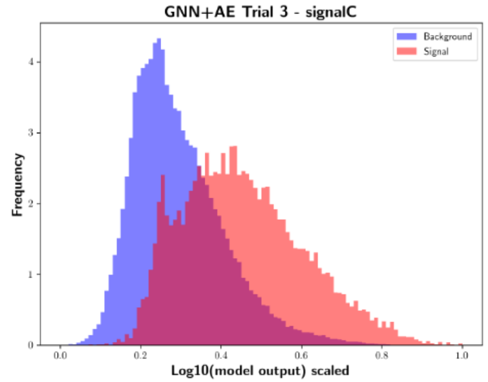
(a) Output Supervised GNN optimized



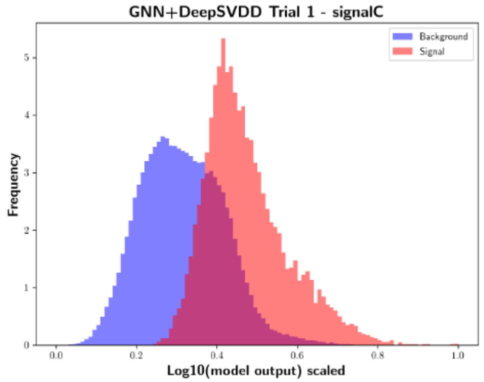
(b) Output Supervised GNN generic



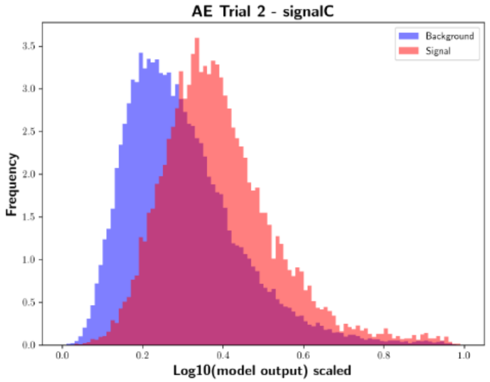
(c) Output Supervised DNN optimized



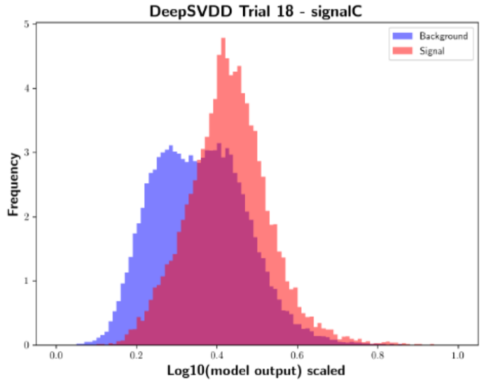
(a) GNN+AE output



(b) GNN + DeepSVDD output

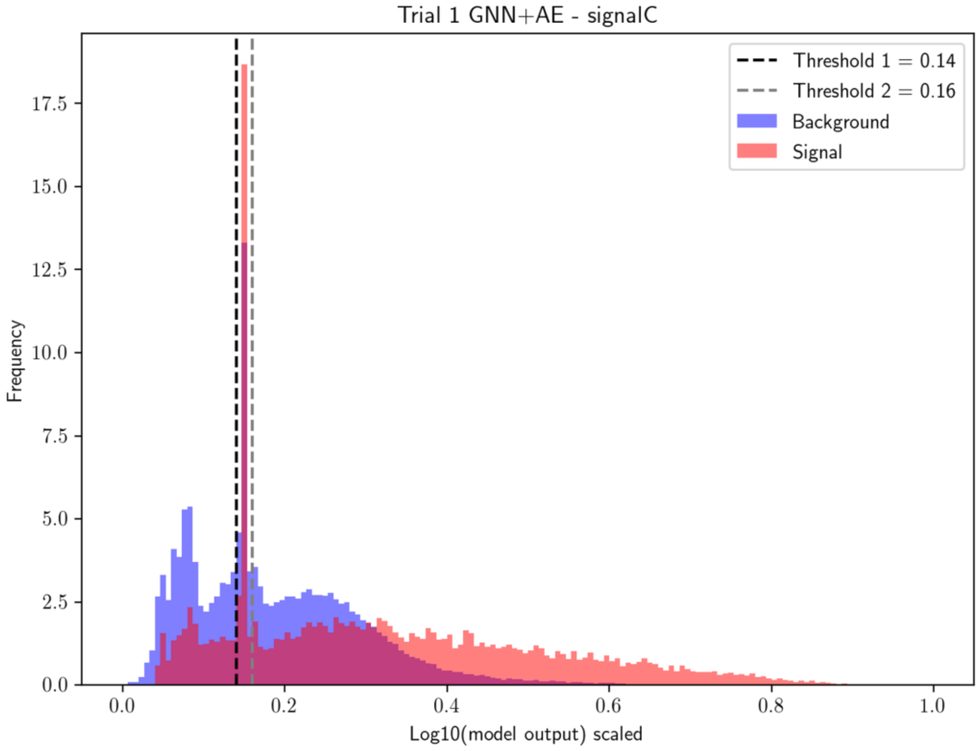


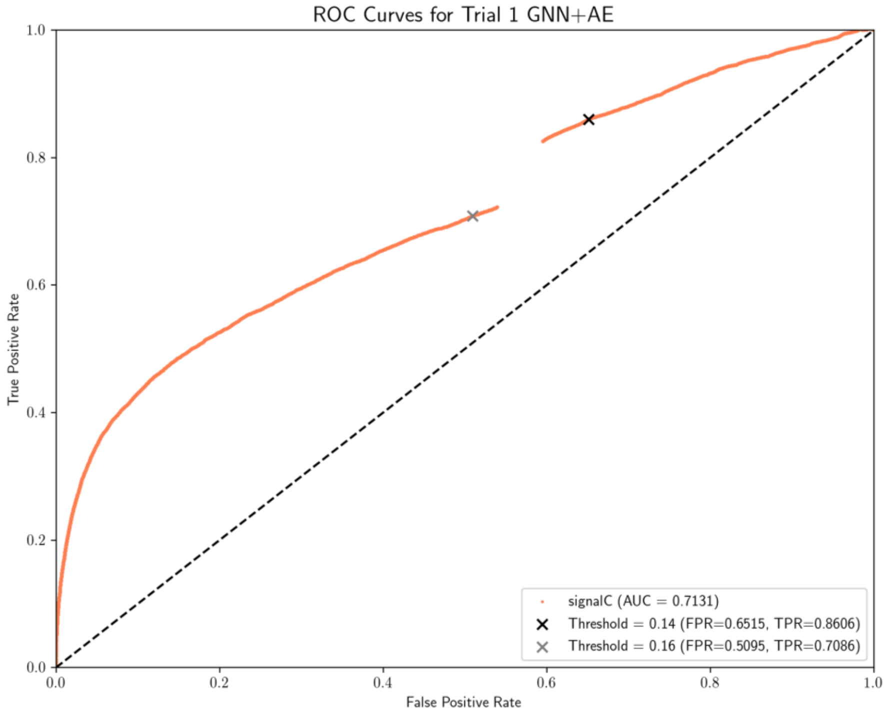
(c) AE output



(d) DeepSVDD output

ROC curves discontinuities in a few of GNN+AE models





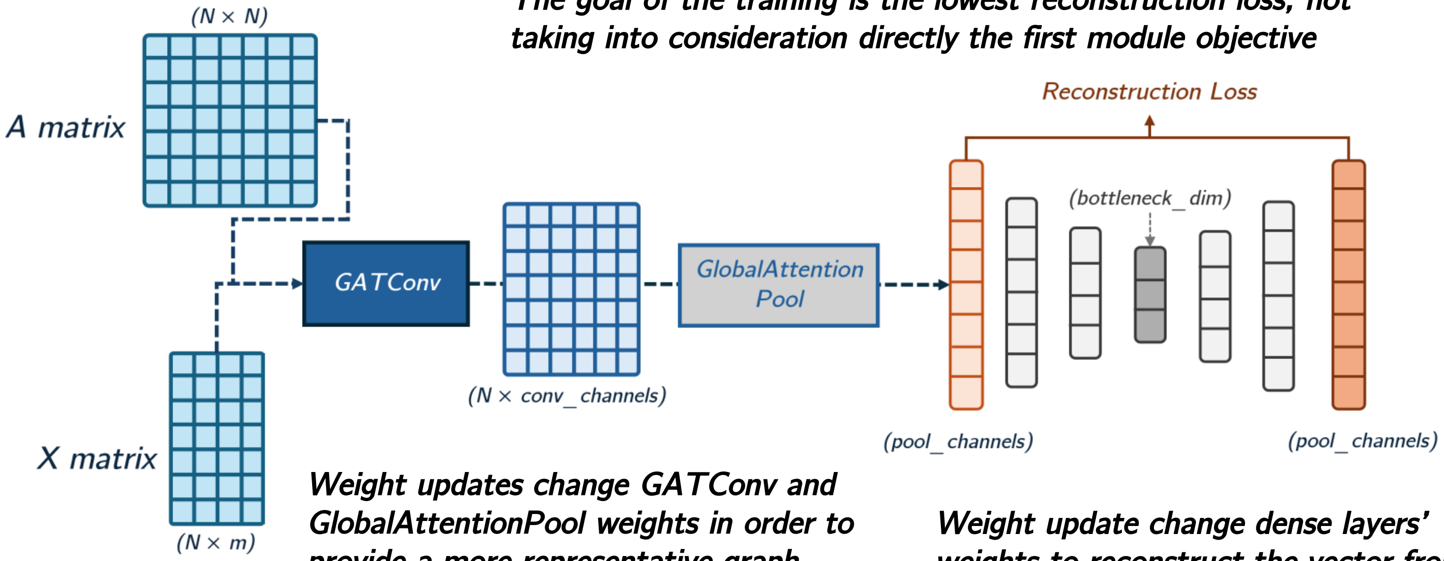
- *background peak in a specific score , the threshold after the peak increases TN and decreases FP abruptly lowering FPR discontinuasly (horizontal shift)*
- *signal peak in a specific score, the threshold after the peak increseases FN and decreases TP abruptly decreasing TPR quickly (vertical shift)*

Training GNN+AE in phases

Training in phases (freezing layers) vs Training all layers in simultaneous – GNN+AE models

If we train the graph module layers first , not updating the weights of the dense layers and then fix the weights of the graph layers updating the AE module, would this impact the performance of the models?

The goal of the training is the lowest reconstruction loss, not taking into consideration directly the first module objective

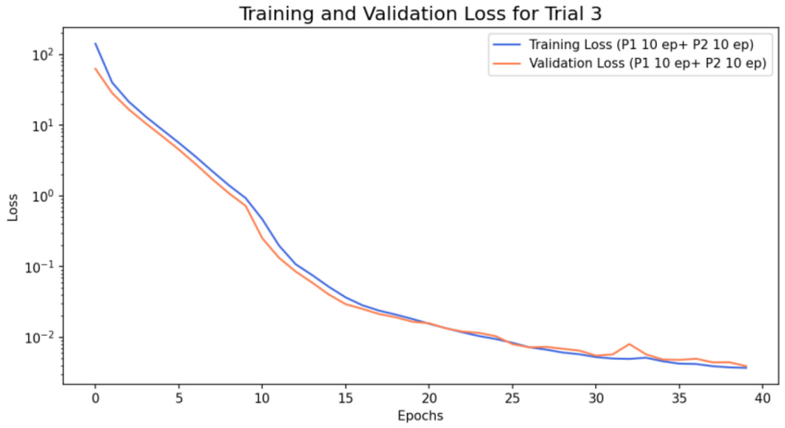


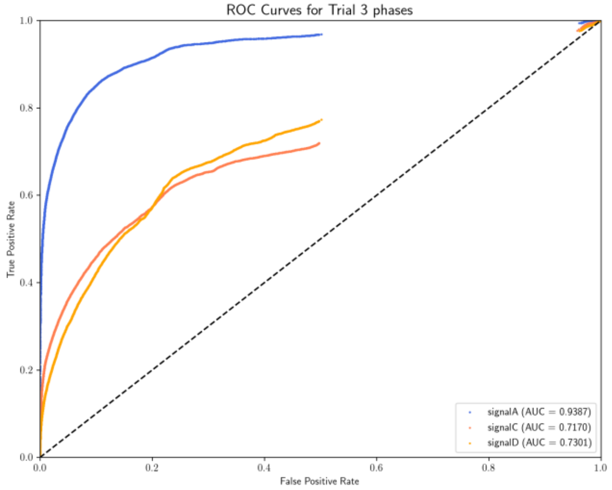
Weight updates change GATConv and GlobalAttentionPool weights in order to provide a more representative graph vector embedding on each epoch

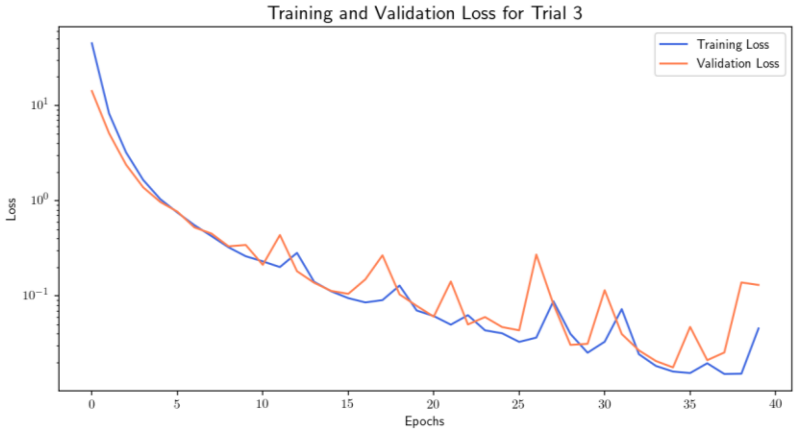
Weight update change dense layers' weights to reconstruct the vector from graph pooling in the best way

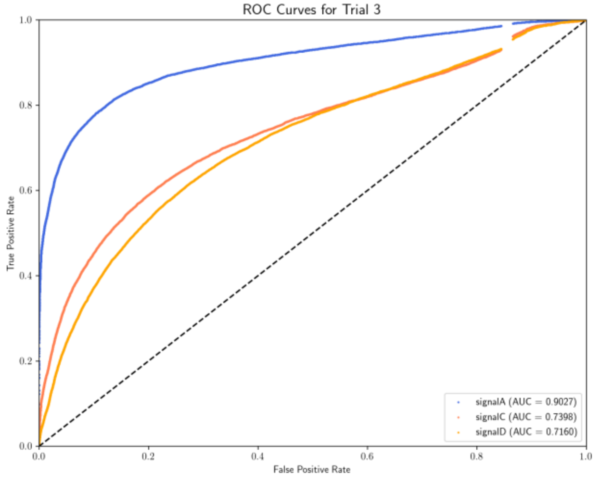
The AE module is trying to reconstruct a vector that is changing the information on each entry on every training epoch - it is pointing to a "moving target"

Training GNN+AE in phases





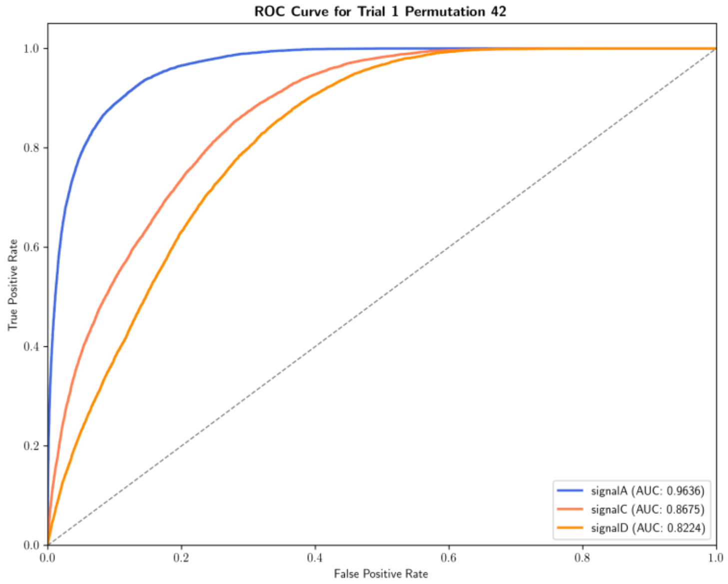


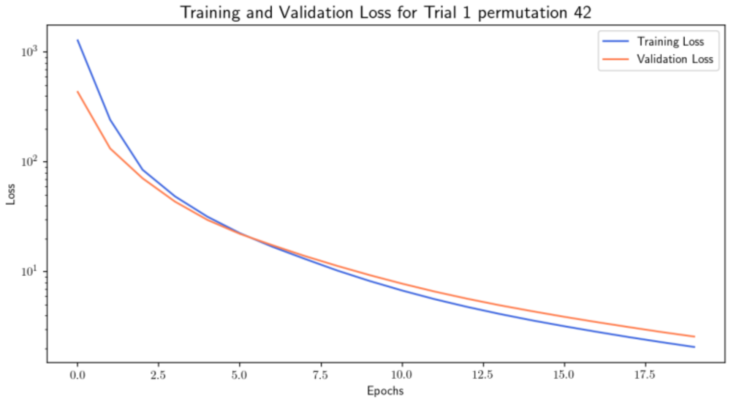


- Training with phases , which involves "freezing" the AE dense layers during the initial epochs to update the graph layer weights and then reversing the process, does not improve the performance of the AD models in any trial. It also does not significantly reduce the AUC values for the different signals.
- Training with phases shows less oscilations in the loss during training, compared to training all layers simultaneously. For this comparison, models that train all layers at the same models that train all layers at once were given increased patience for validation loss increasing.
- In the original models, patience was set to the minimum (patience = 1 epoch). Increasing the patience did not lead to higher AUC values.
- Models trained with phases show more a more pronounced problem with background/signal peaks in the specific AD score than models that train all layers at once. Models trained with high patience but without phased training exhibit this issue more clearly than the original models with a low epoch number.

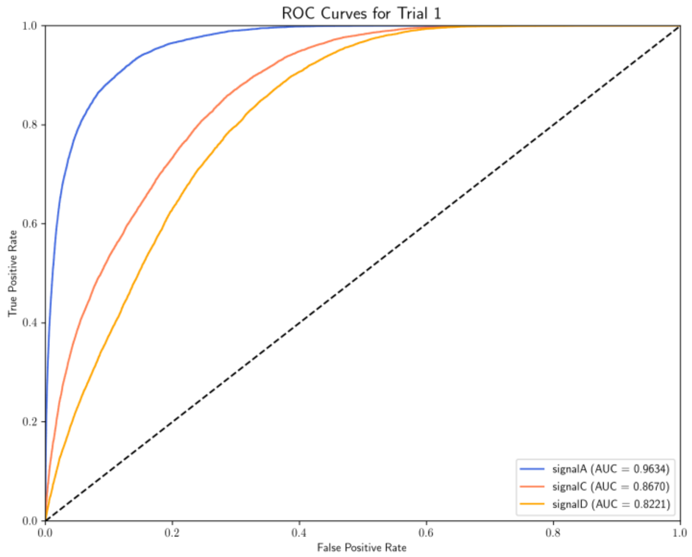
Permuting the input objects

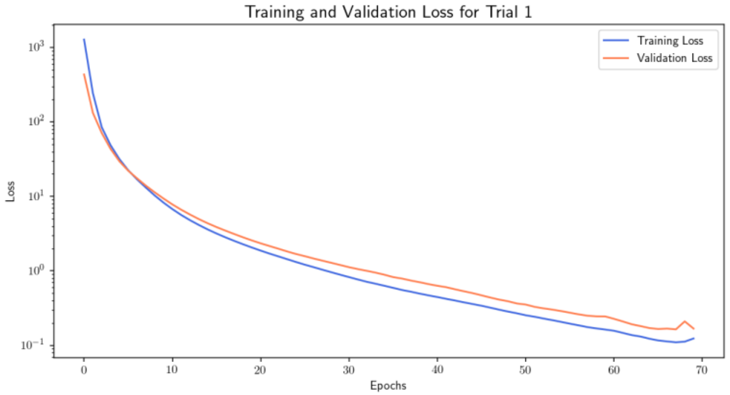
Permuted ordered inputs 1
Trial 1 – GNN+DeepSVDDs



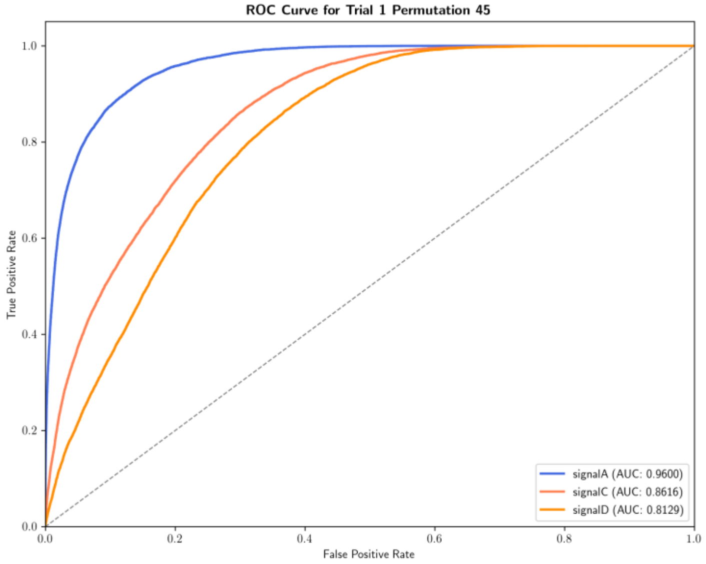


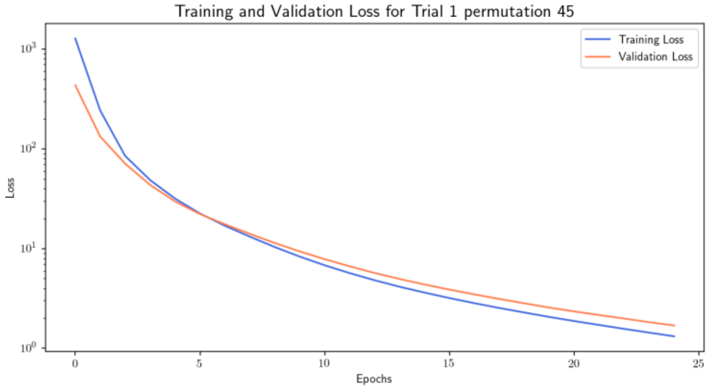
Pt ordered inputs
Trial 1 – GNN+DeepSVDDs





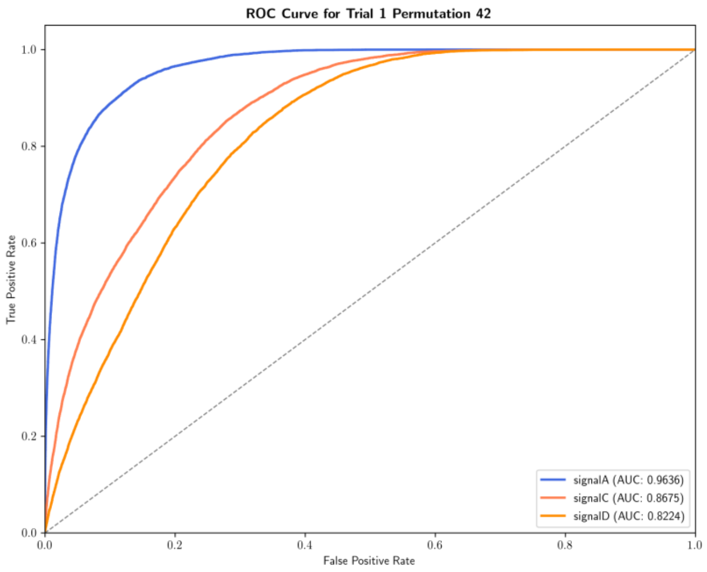
Permuted ordered inputs 2
Trial 1 – GNN+DeepSVDDs

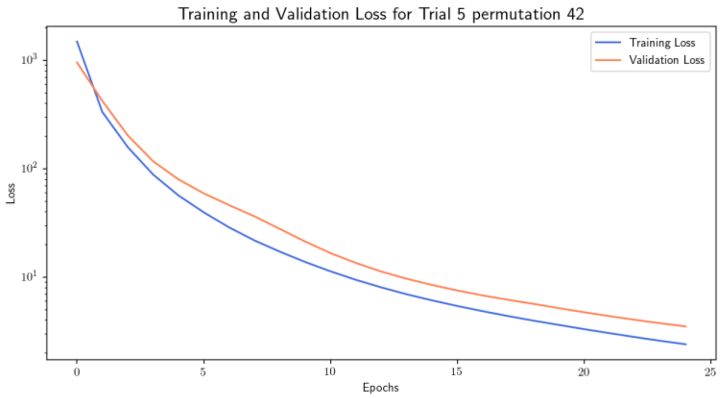




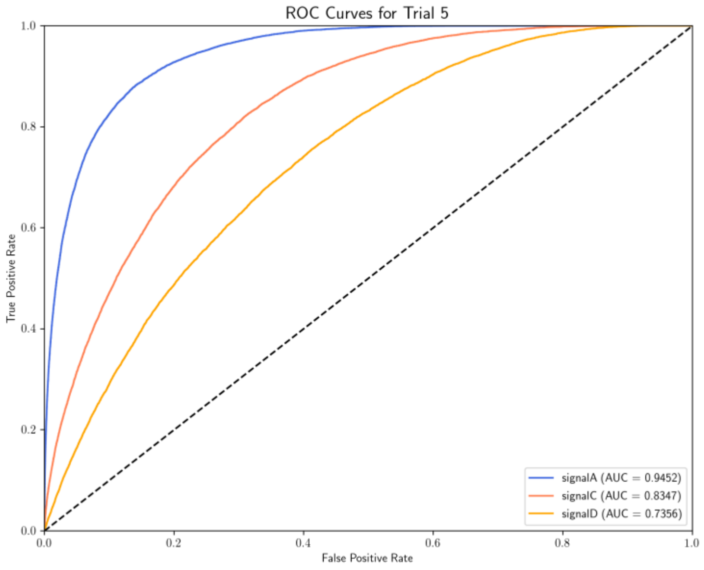
Permuting the input objects

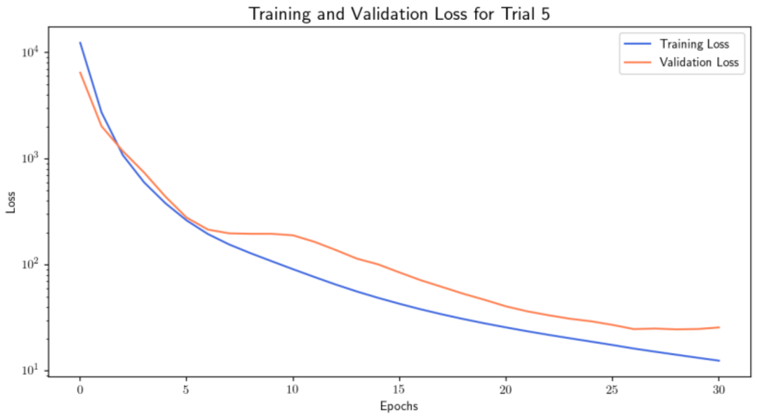
Permuted ordered inputs 1
Trial 5 – GNN+DeepSVDDs



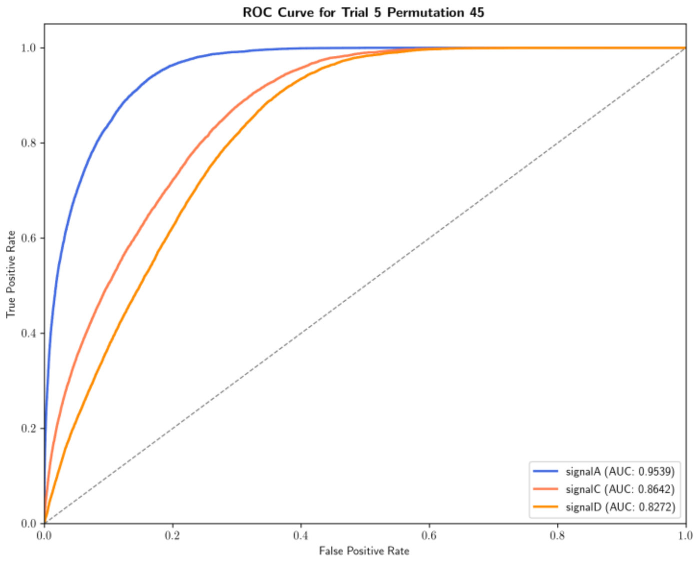


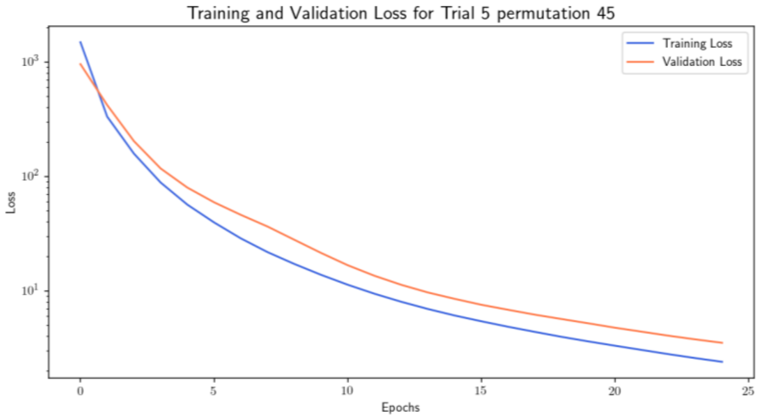
Pt ordered inputs
Trial 5 – GNN+DeepSVDDs





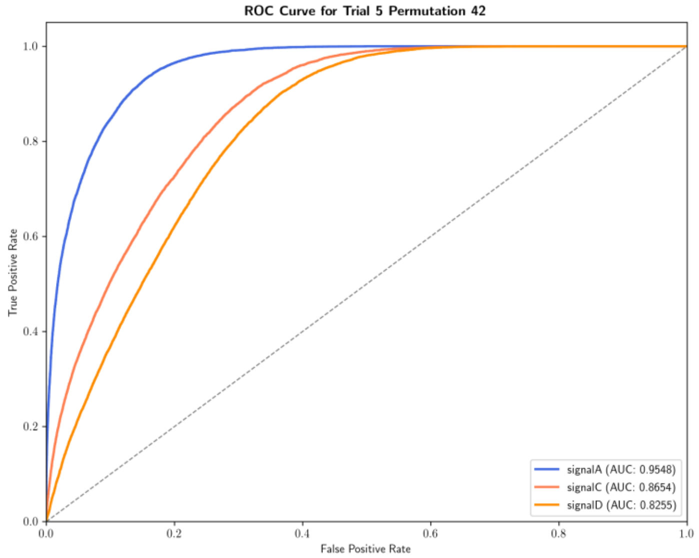
Permuted ordered inputs 2
Trial 5 – GNN+DeepSVDDs

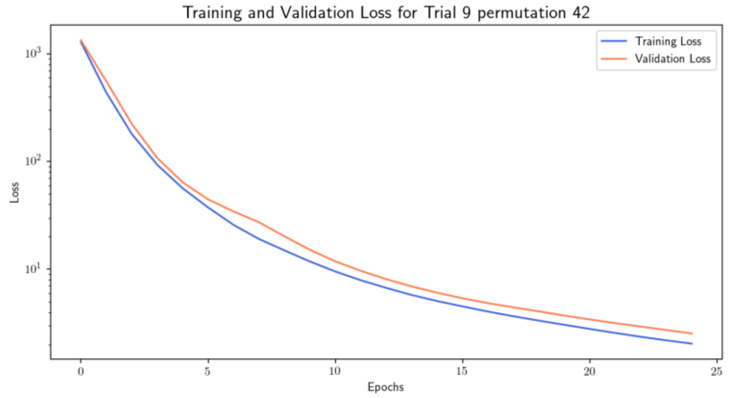




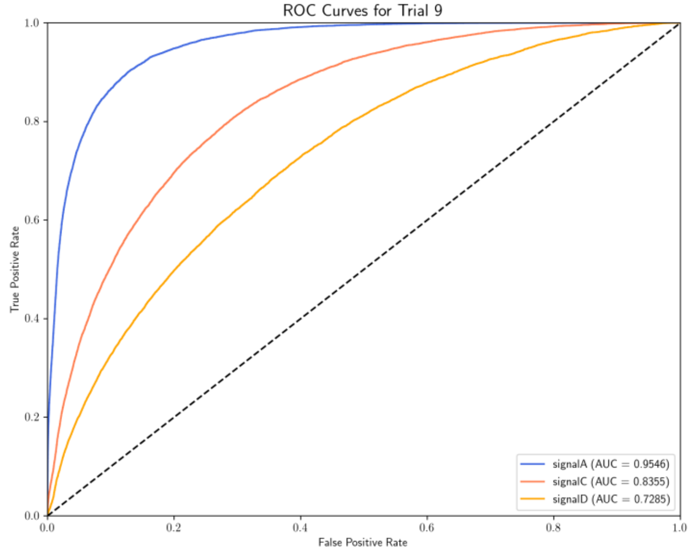
Permuting the input objects

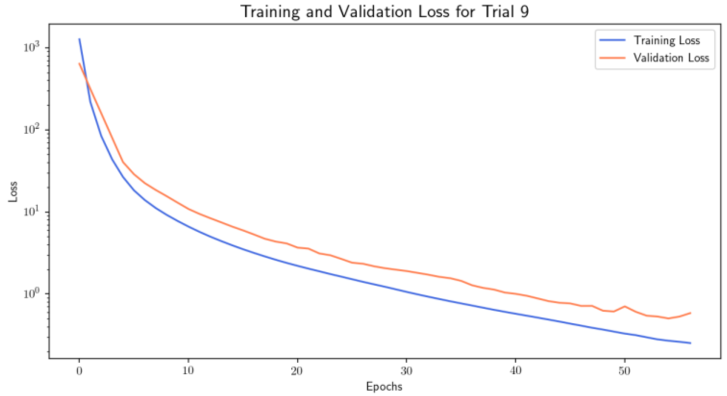
Permuted ordered inputs 1
Trial 9 – GNN+DeepSVDDs



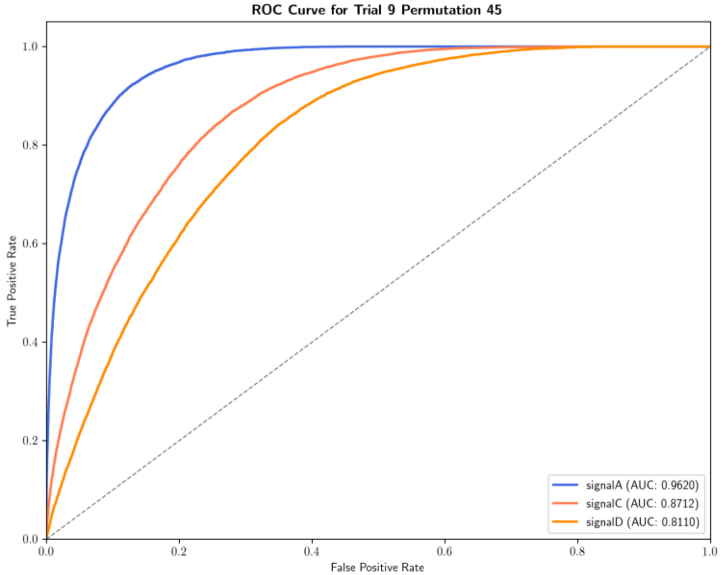


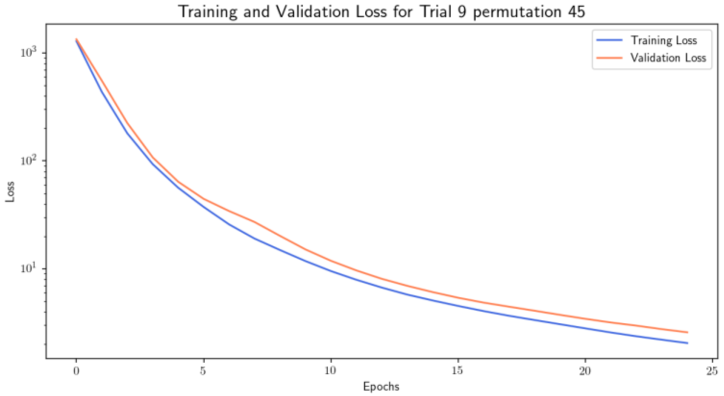
Pt ordered inputs
Trial 9 – GNN+DeepSVDDs



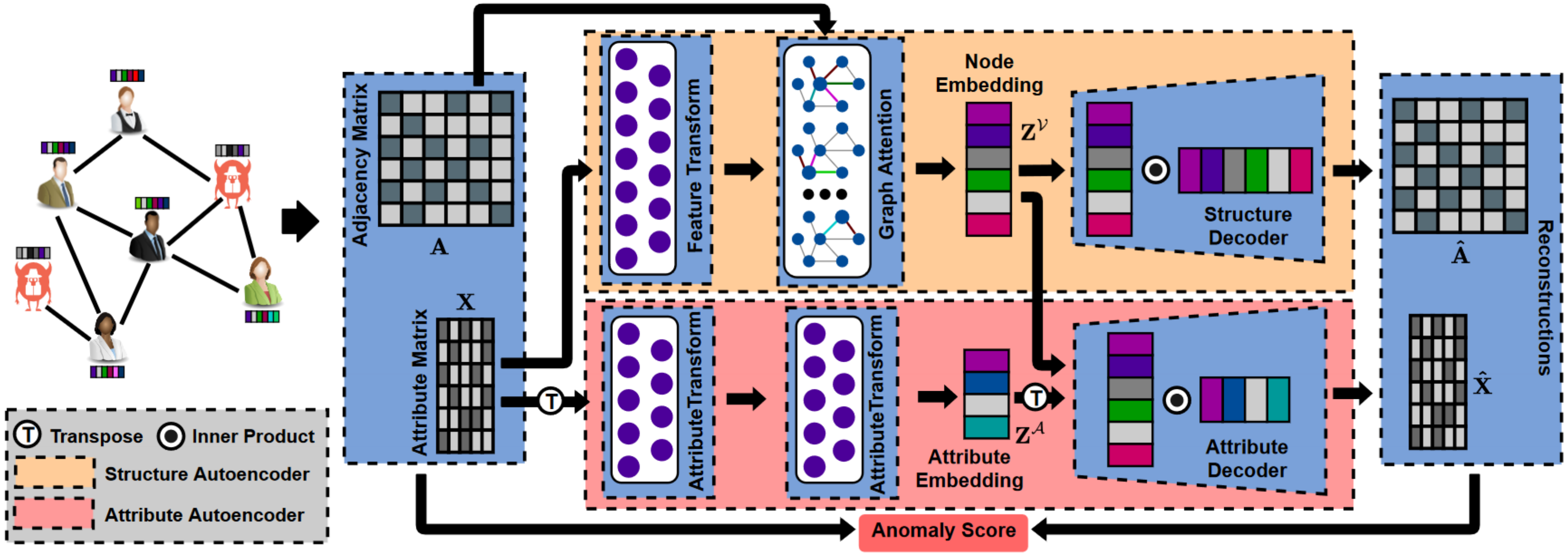


Permuted ordered inputs 2
Trial 9 – GNN+DeepSVDDs





Pygod – Pytorch implemented models that use GNN+AE



Anomaly DAE

- DOMINANT
- CONAD