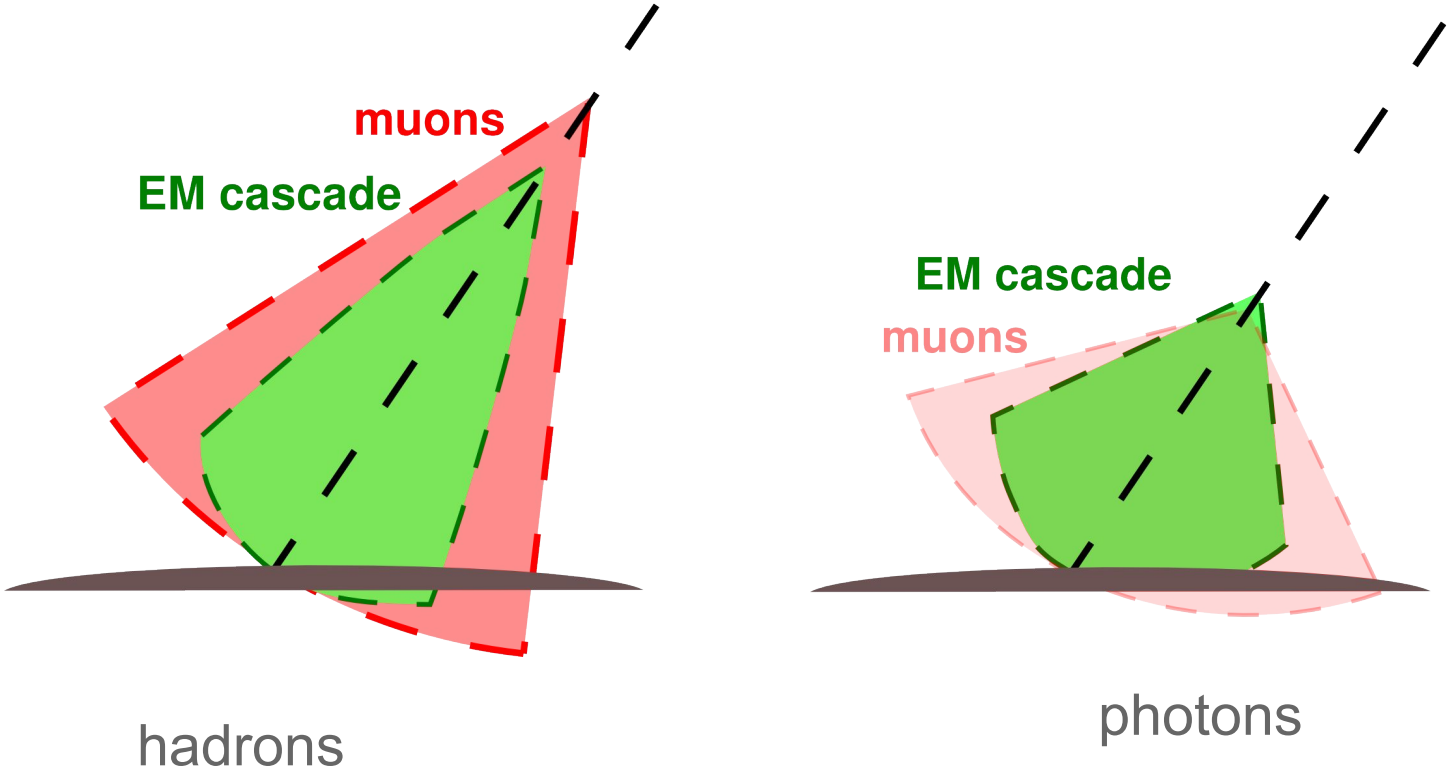

Machine Learning

Overview

- Overview of neural network architectures
  - BDT, CNN, GNN, …

- Robustness and Domain Adaptation

Hadron- and photon-induced air showers



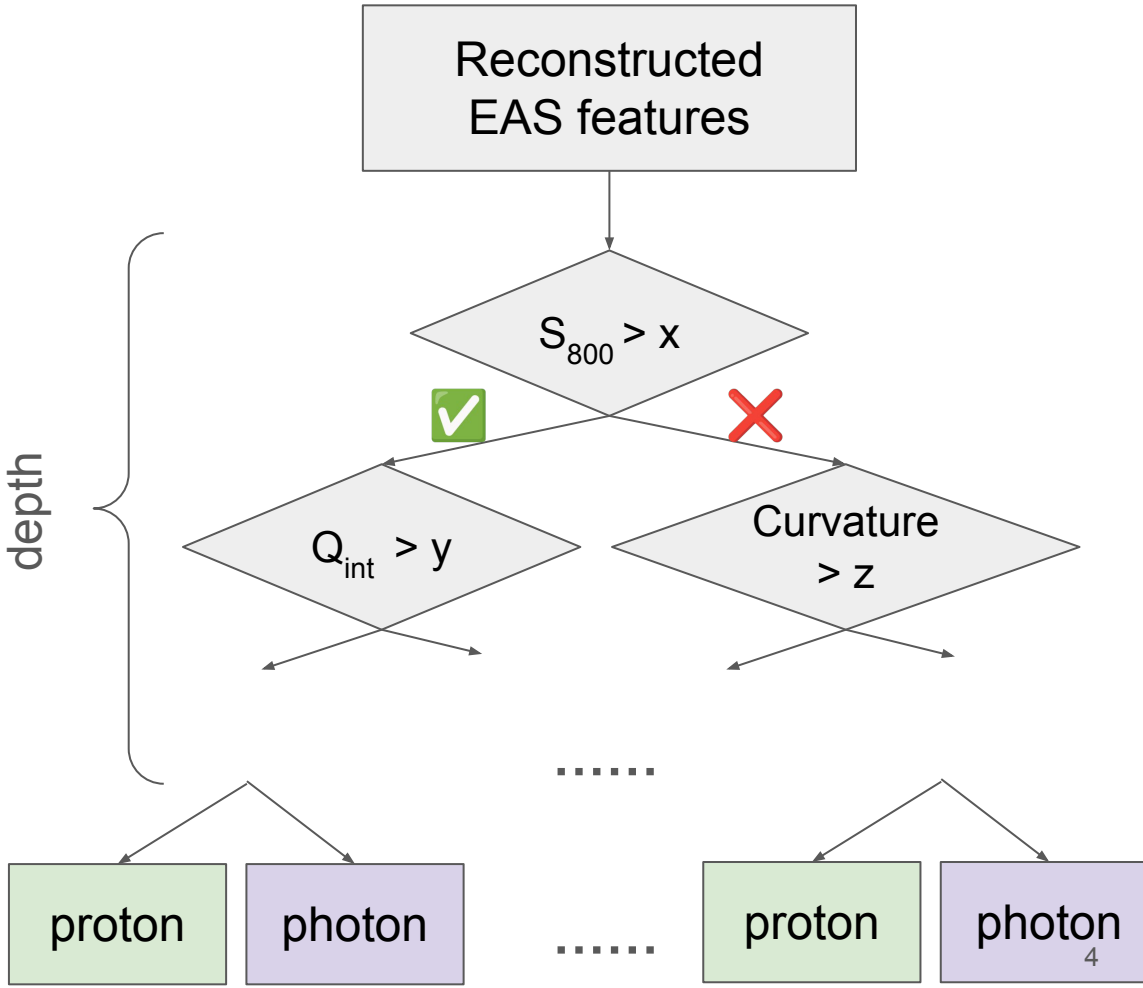
Boosted Decision Trees

Input data:
fixed set of features

Processing:
multistep binary partition

Pros:
Simple, effective, baseline.

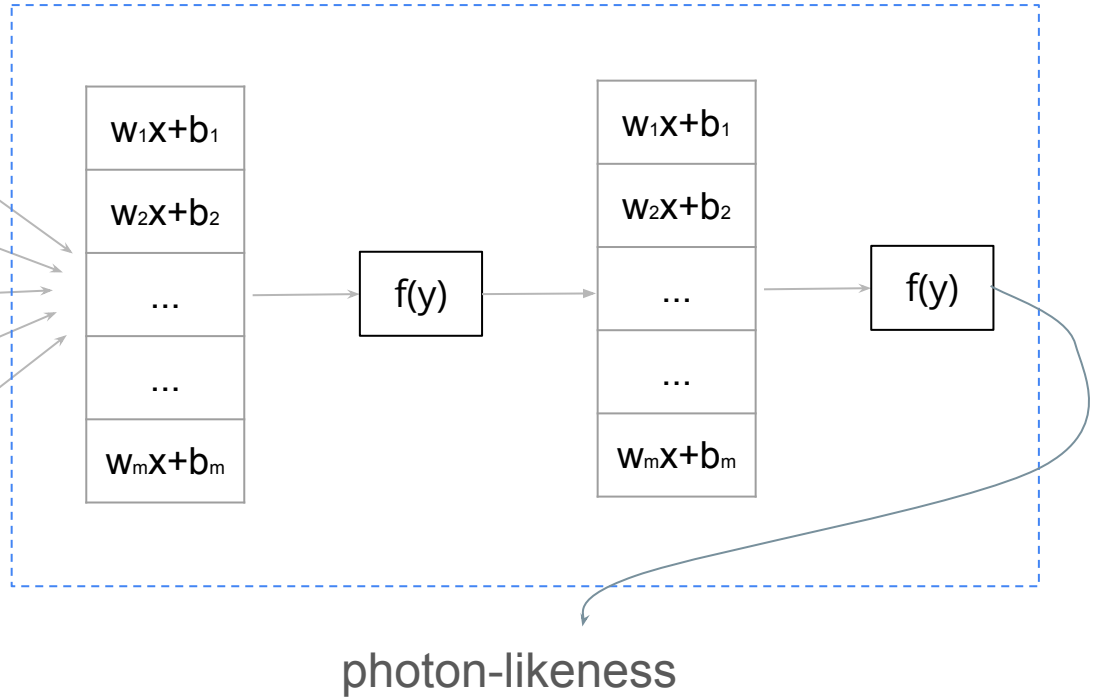
Cons:
Simple



Fully connected neural networks

Input parameters:

1. Linsley front curvature
2. Area-over-peak (AOP)
3. AOP slope
4. Reconstructed S800
5. Reconstructed E
6. Integral signal
7. …

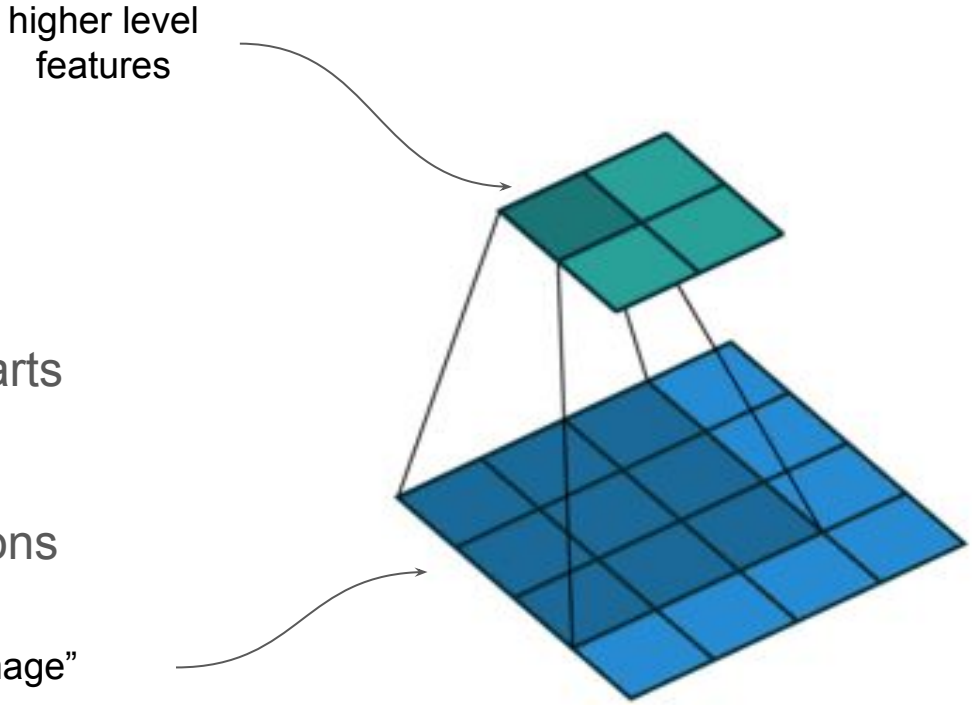


Convolutional neural networks

Convolutional neural networks

Requirements:

- Make use of image structure
- Uniform analysis of different parts of an image
- Shift-invariance of the predictions



Vincent Dumoulin, Francesco Visin -
A guide to convolution arithmetic for deep learning

Convolutional neural networks

$*$

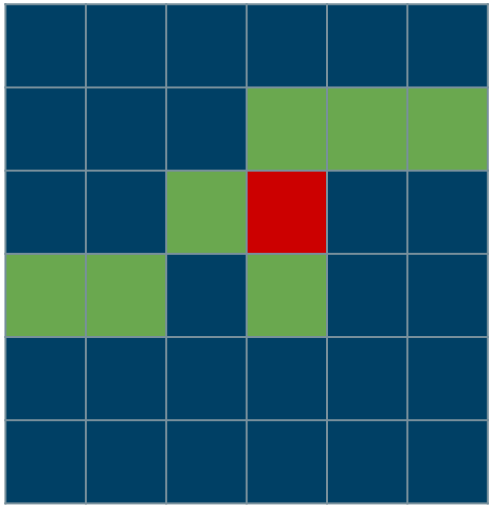
1	0	-1
2	0	-2
1	0	-1

Deeper layers of neural networks catch more complex structures:

edges → circles → general face pattern

CNN: TA SD stations grid

- x, y, and z coordinates of the detector
- Detector's total signal
- Time of the plane front arrival
- Difference in time between the start of the recorded signal and the wavefront arrival
- Masks:
  - Was triggered?
  - Was saturated?
  - Was excluded from the geometry fit?



CNN

High level features

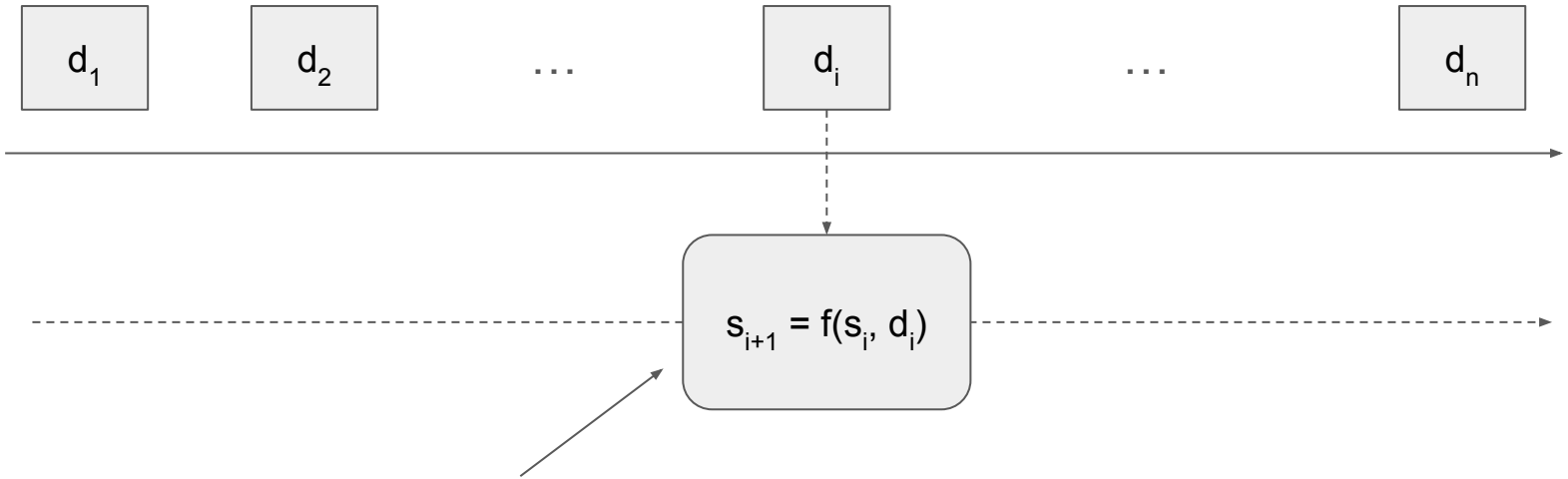
MLP

Photon-likeness

Recurrent neural networks

Recurrent neural networks

Designed to analyze sequences.

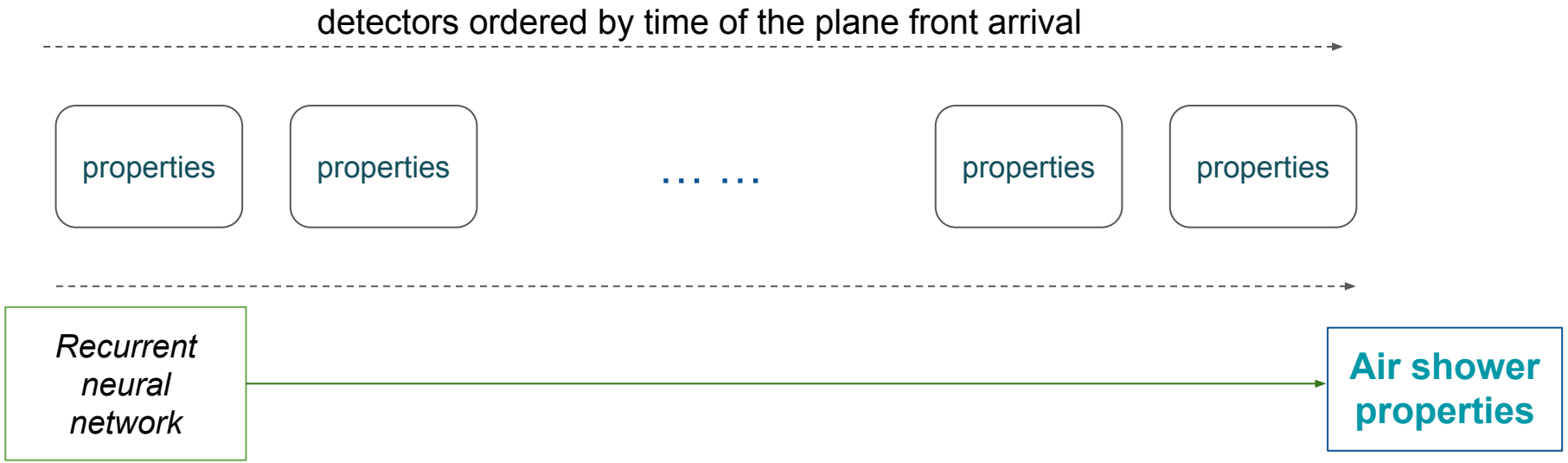


Recurrent block - “mini” neural network with internal state s_t

Recurrent neural networks

Due to hot weather he quickly became tired.

TA SD events as ordered sequence

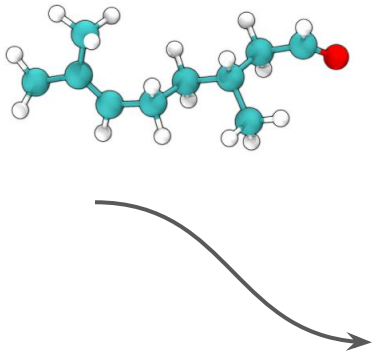


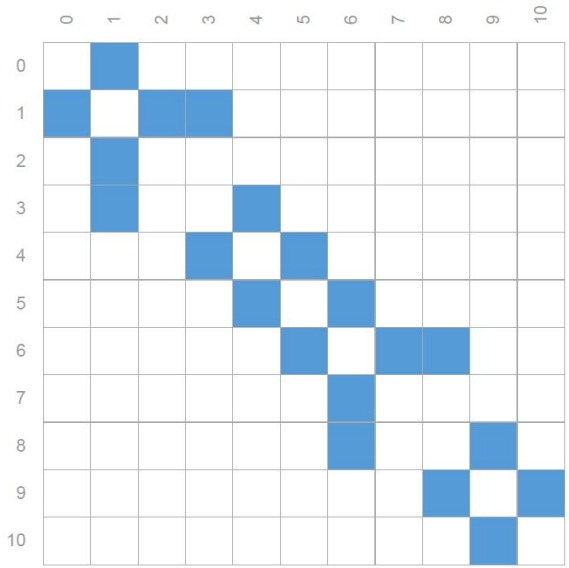
Arbitrary detector layout

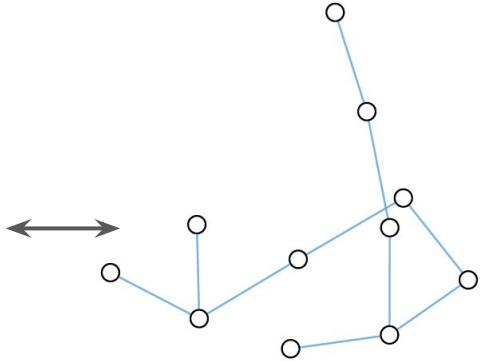
Graph neural networks

Graph neural networks

Graphs allow to represent complex data.

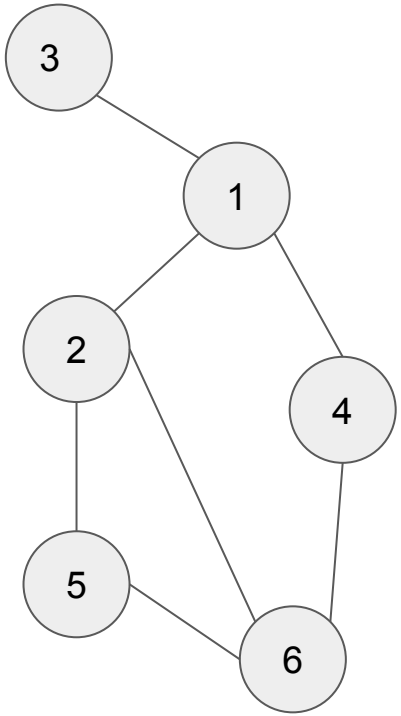






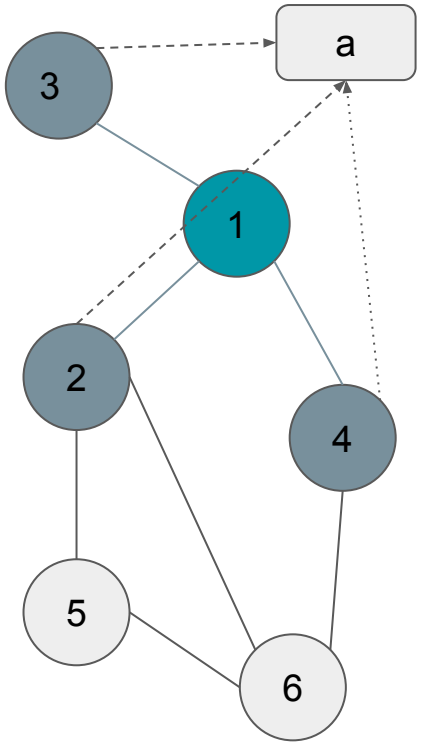
Nodes of the graph carry representation (features) of the corresponding object.
One can introduce features to edges as well.

Graph neural networks



Graph neural networks “update” graphs.
1 layer = 1 graph update

Graph neural networks



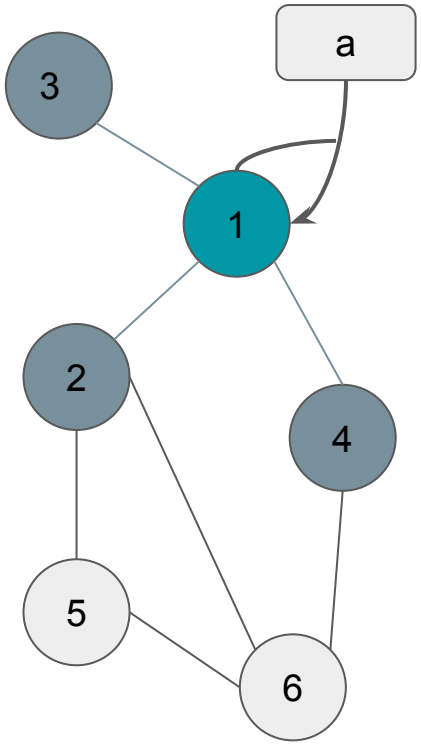
Graph neural networks “update” graphs.
1 layer = 1 graph update

Convolutional GNN:

For each node:

- Consider its neighbours
- Aggregate their features to a single vector:
  - Must be independent of graph representation
    For example, $a = 1/N_{neighb}(v_2+v_3+v_4)$

Graph neural networks



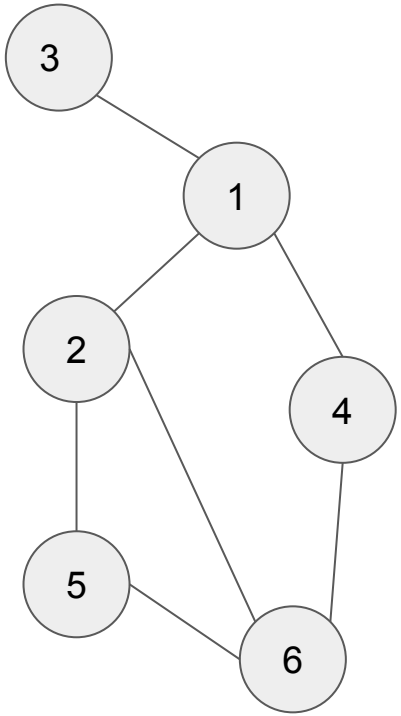
Graph neural networks “update” graphs.
1 layer = 1 graph update

Convolutional GNN:

For each node:

- Consider its neighbours
- Aggregate them to a single vector:
  - Must be independent of graph representation
    For example, $a = 1/N_{neighb}(v_2+v_3+v_4)$
- Update the node using a small MLP:
  - $v^{new}_1 = f(\theta)(v_1,a)$

GNN and TA SD



SD stations ↔ Graph nodes
Closest stations ↔ Connected nodes

Arbitrary detector layout
Need to define “close” stations (time, distance, …)

Transformers

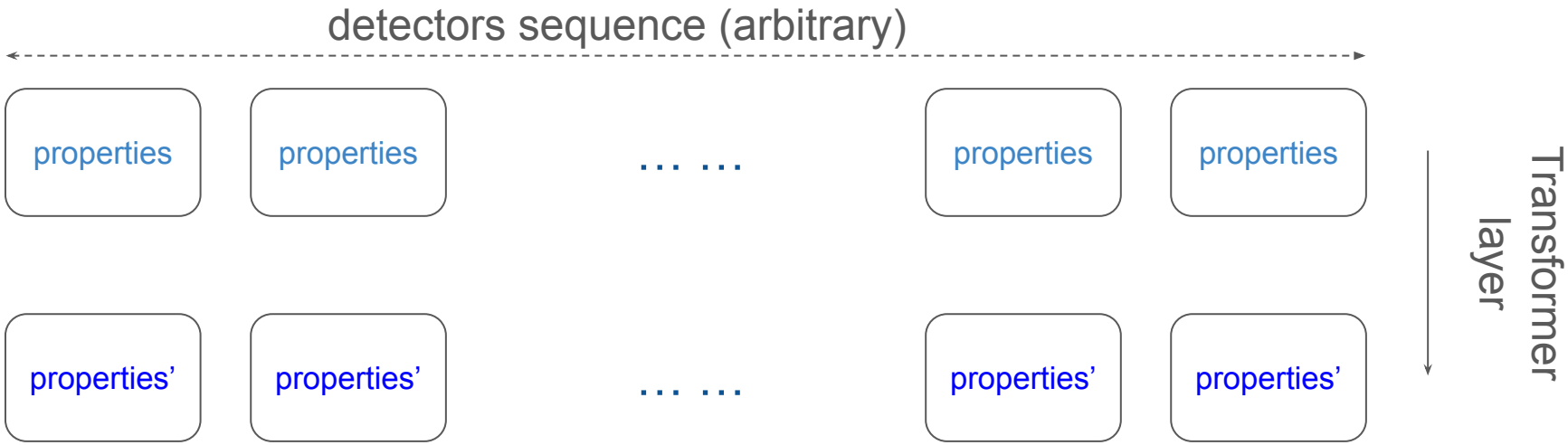
Transformer

=

Graph Neural Network with:

- All nodes connected
- Each nodes importance is estimated via self-attention mechanism

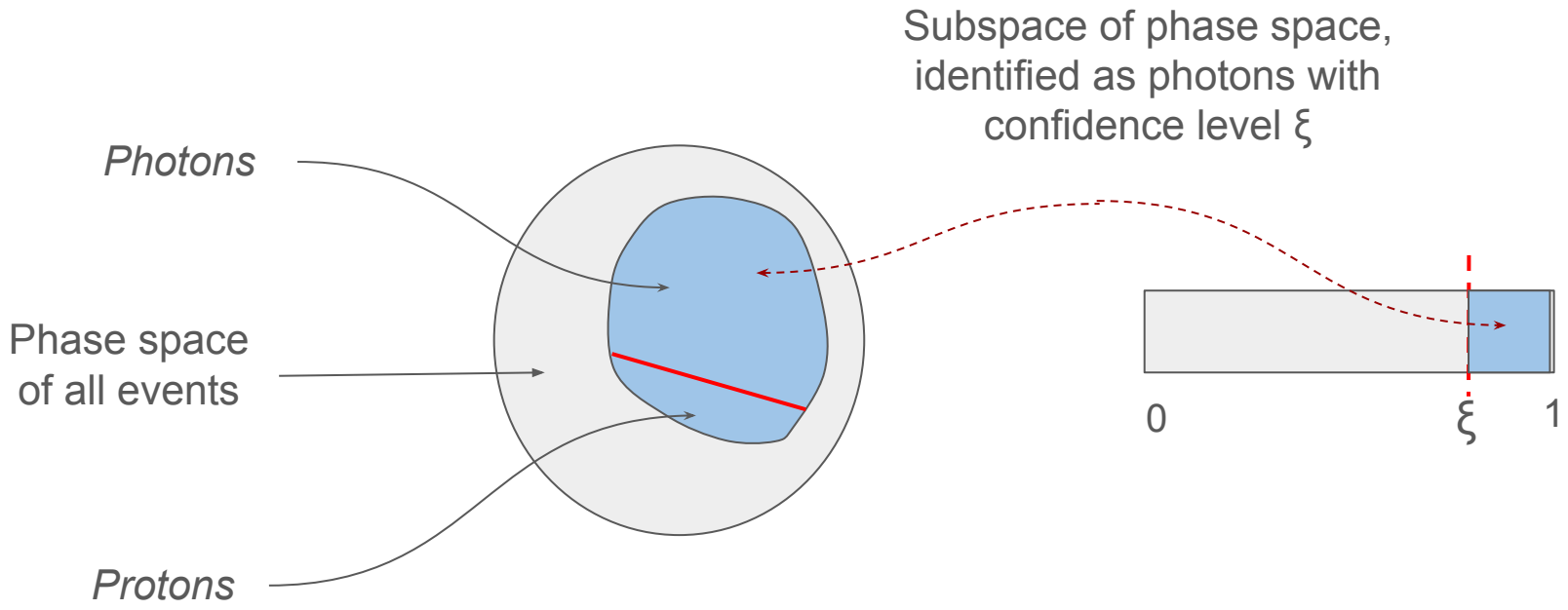
Simplified transformers



- Arbitrary detector layout
- Easy to implement:
  - Native PyTorch layer
  - DTS Parser provides suitable data representation

Robustness and domain shift

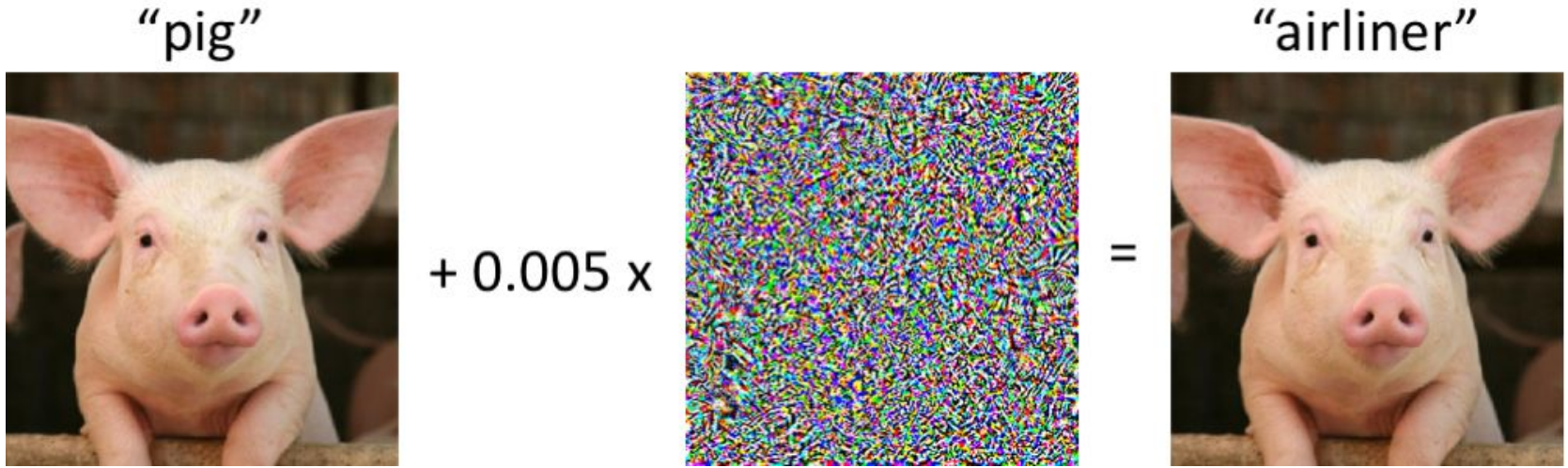
Phase space splitting



Applicability of trained neural networks

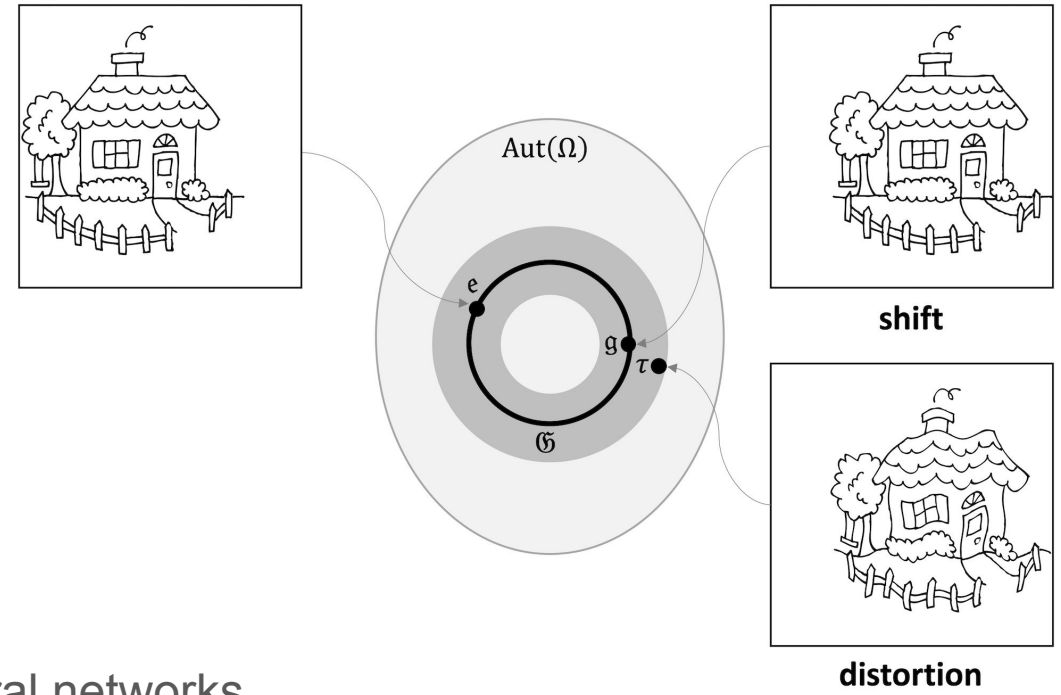
Neural network are trained on specific data with certain data distribution and are very sensitive.

Disagreement between training and to-be-applied-on data can be crucial:
systematic shifts, bad quality of MC data, …



Towards robustness

- Imitate noise in measurements
- Data augmentation
- Intentionally use simple neural networks
- Apply regularization techniques



M. Bronstein, Geometric Deep Learning

MC imperfections

Train NN to distinguish between MC and SD:

	Stations data only	With reconstructed parameters	With waveforms
MC (QGSJETII-04) vs SD	70%	75%	99.9%
QGSJETII-04 vs EPOS-LHC	55%	56%	60%

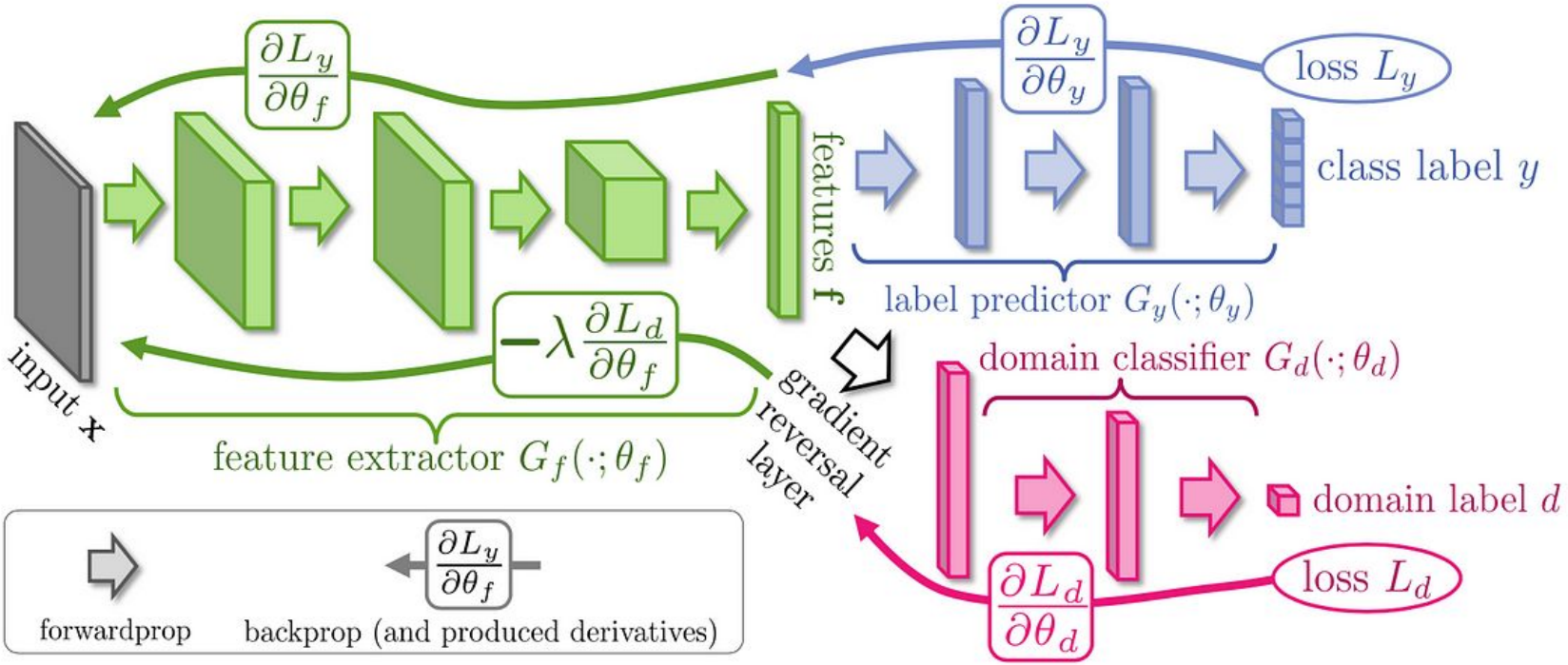
Domain adaptation as solution

arXiv:1409.7495

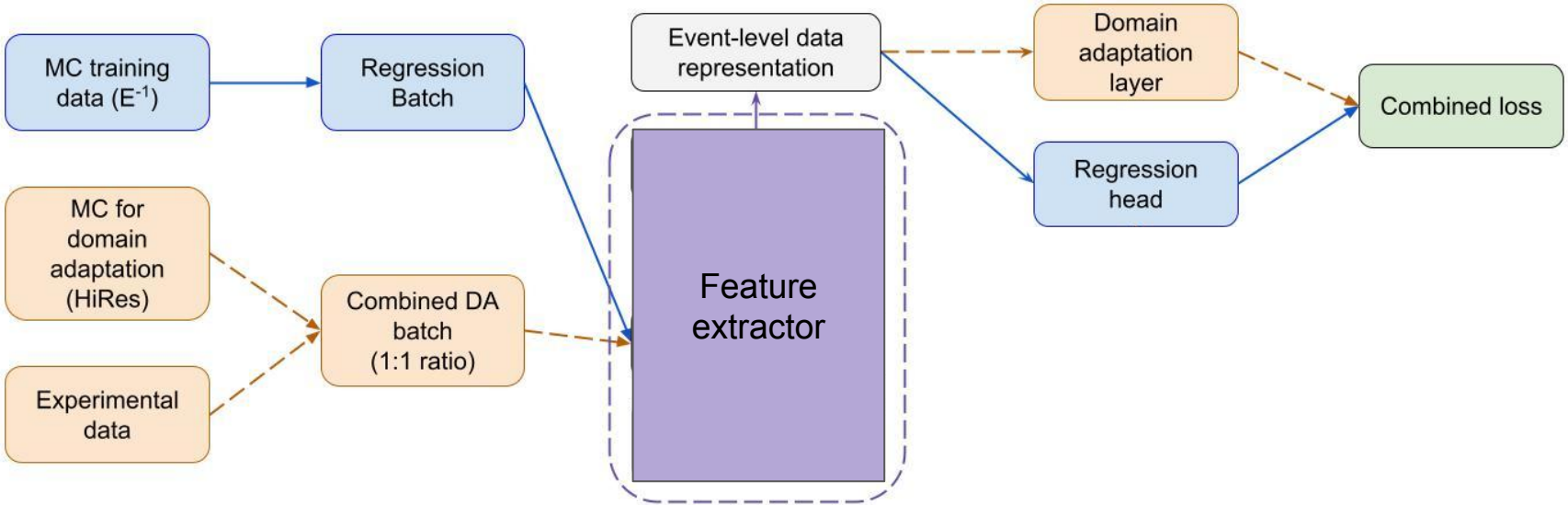
Source and target domains might be different.

But one can restrict NN to learn domain-invariant features.

Domain adaptation technique



Data processing for TA SD energy regression



Limitation of domain adaptation

X - phase space of TA SD responcies for UHECRs.

Neural network maps: $P(X) \rightarrow P(Z) \rightarrow P(E_{est})$

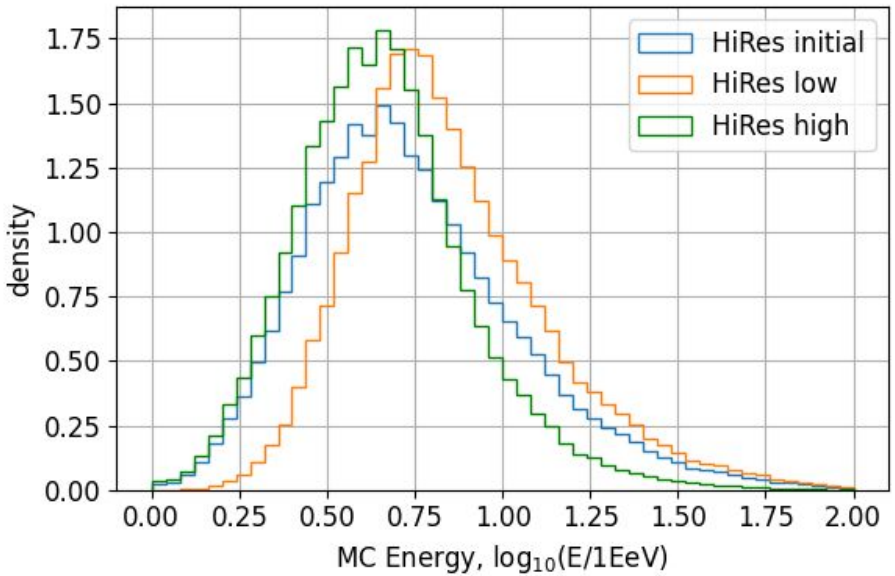
Domain adaptation aligns P(Z): $P(Z)_{MC} = P(Z)_{Exp}$

But: $P(z) = \int P(z|E)P(E)dE$

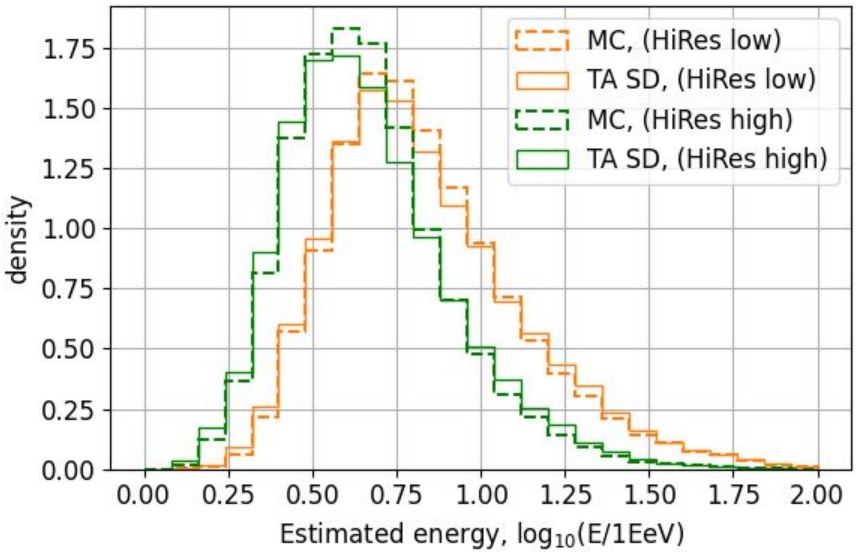
Hence domain adaptation may align spectra!

Domain adaptation technique

Ground-truth spectra for DA



Estimated TA SD spectra



Hence domain adaptation does align spectra!

Adaptive domain adaptation

Observation: predicted TA SD spectra deviates towards HiRes

Idea: iteratively reweight MC events to yield the currently estimated TA SD spectrum; find equilibrium point.

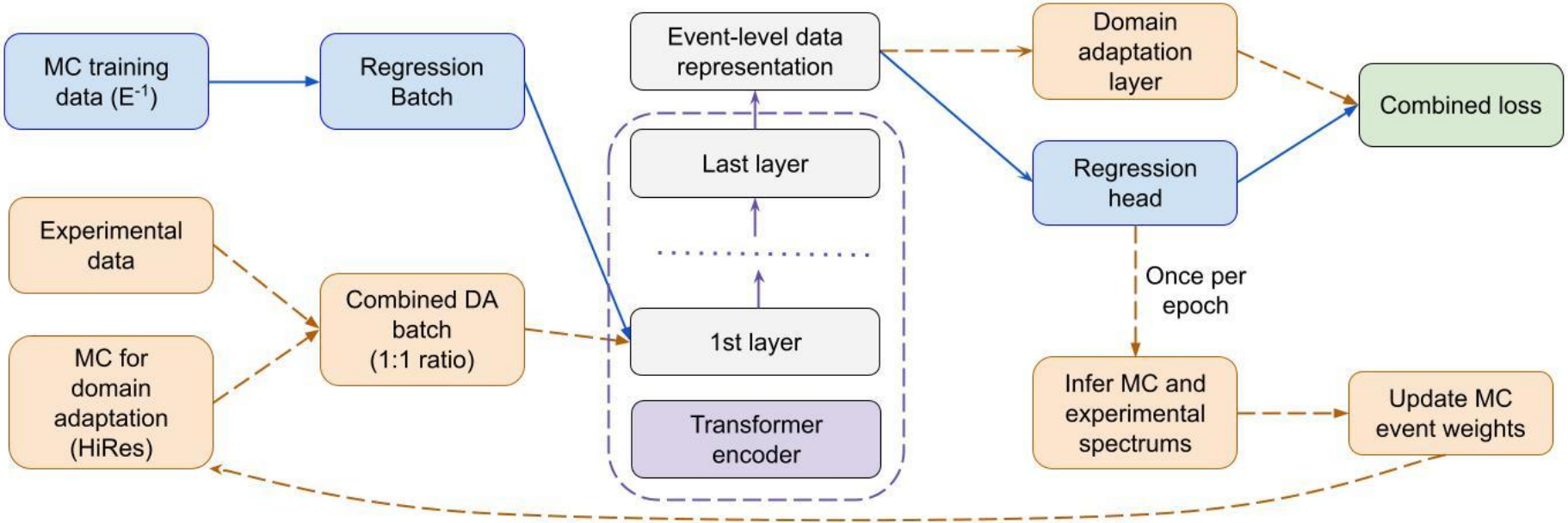
Domain adaptation is focused on getting domain-invariant P(z|E)

Algorithm 1 Adaptive domain adaptation (ADA)

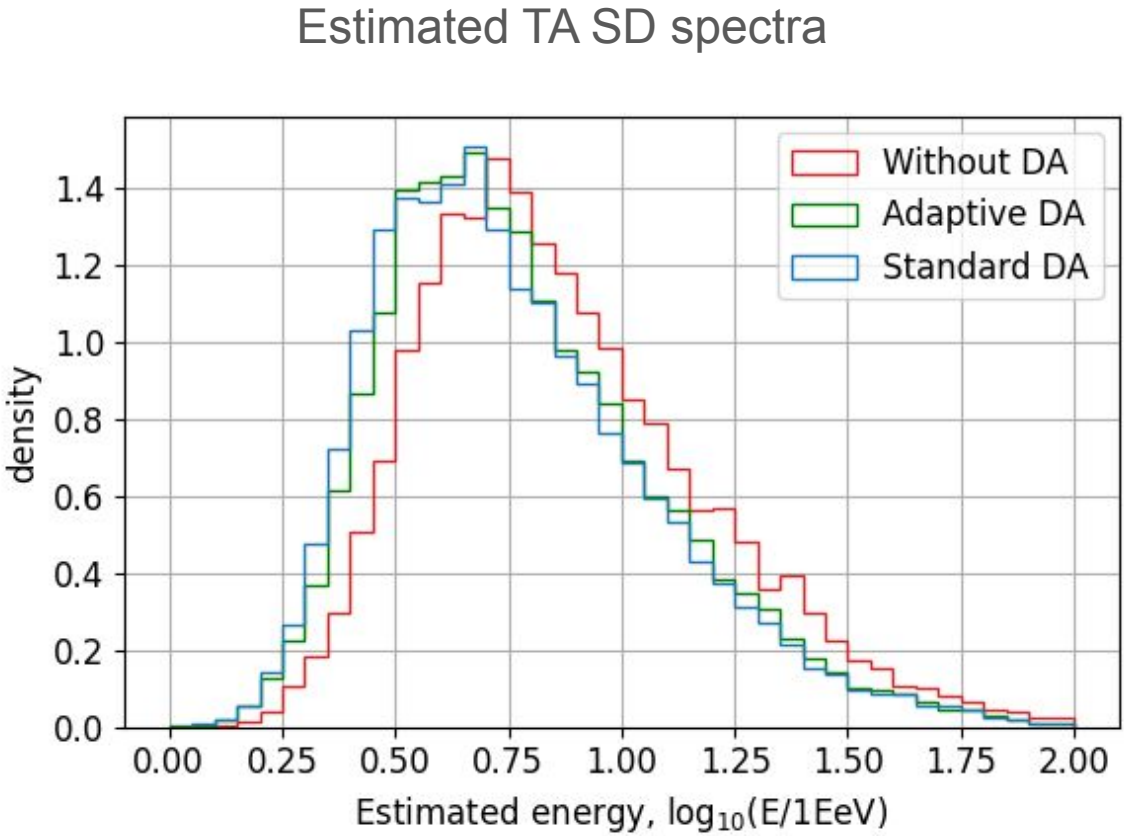
Require: Training data for label prediction $\mathcal{D}_{\text{tr}}$ (E^{-1} MC)
Require: Target domain data $\mathcal{D}_{\text{exp}}$ (experimental data)
Require: Source data for DA $\mathcal{D}_{\text{da}}$ (HiRes MC)
Require: Energy bin edges $\{E_i\}_{i=1}^{B}$, convergence tolerance τ
Require: Weights $w_{\text{rew}}(E)$ for casting E^{-1} spectrum to HiRes one

Initialize bin weights: $w_i^{(0)} \leftarrow w_i^{\text{init}}$ for $i = 1, \ldots, B$
Initialize neural network parameters θ
repeat
    Train for one epoch:
        Sample batch from $\mathcal{D}_{\text{tr}}$ with weights $w_{\text{rew}}(E) \cdot w_i^{(k)}$
        Sample batch from $\mathcal{D}_{\text{da}}$ with weights $w^{(k)}$
        Sample batch from $\mathcal{D}_{\text{exp}}$
        Update θ via gradient descent on $\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{reg}} + \mathcal{L}_{\text{domain}}$
    Estimate spectra:
        $p_{\text{da}}^{(k)}(E_i) \leftarrow$ histogram of $E(\mathcal{D}_{\text{da}})$
        $p_{\text{exp}}^{(k)}(E_i) \leftarrow$ histogram of $E(\mathcal{D}_{\text{exp}})$
    Update weights:
    $w_i^{(k+1)} \leftarrow p_{\text{exp}}^{(k)}(E_i) / p_{\text{da}}^{(k)}(E_i)$
until $\max_i |w_i^{(k)} - w_i^{(k-1)}| < \tau$
Return: Trained model, equilibrium weights $w_i^{(k)}$

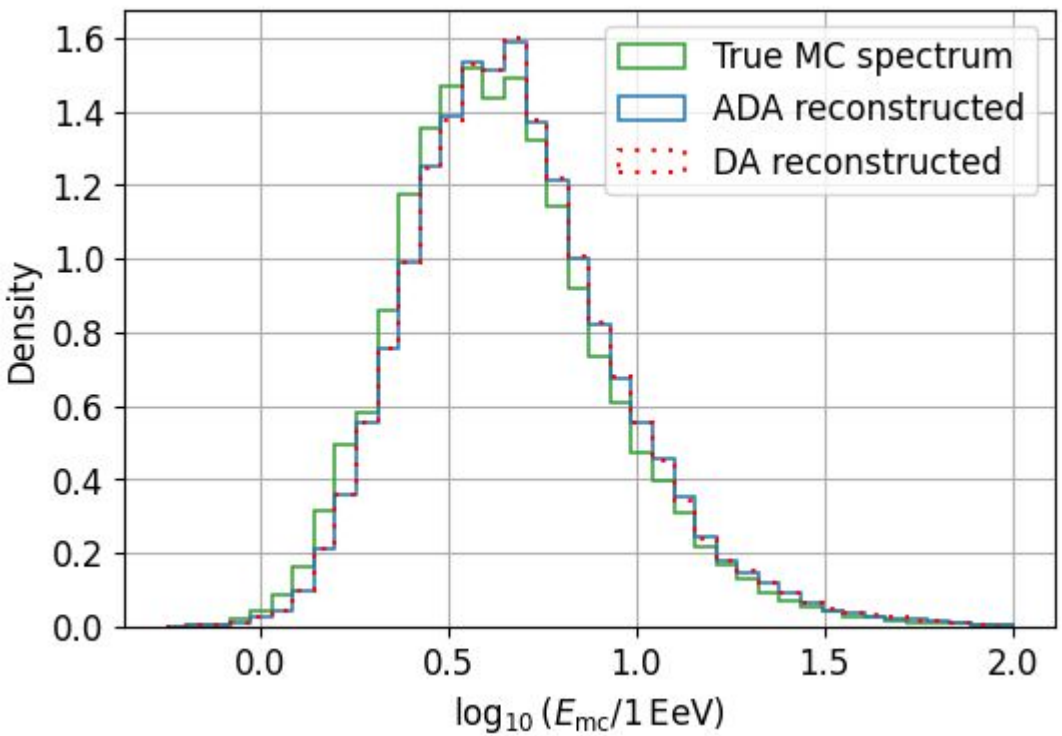
Adaptive domain Adaptation



Comparison of energy reconstructions



DA perfectly works on MC-to-MC

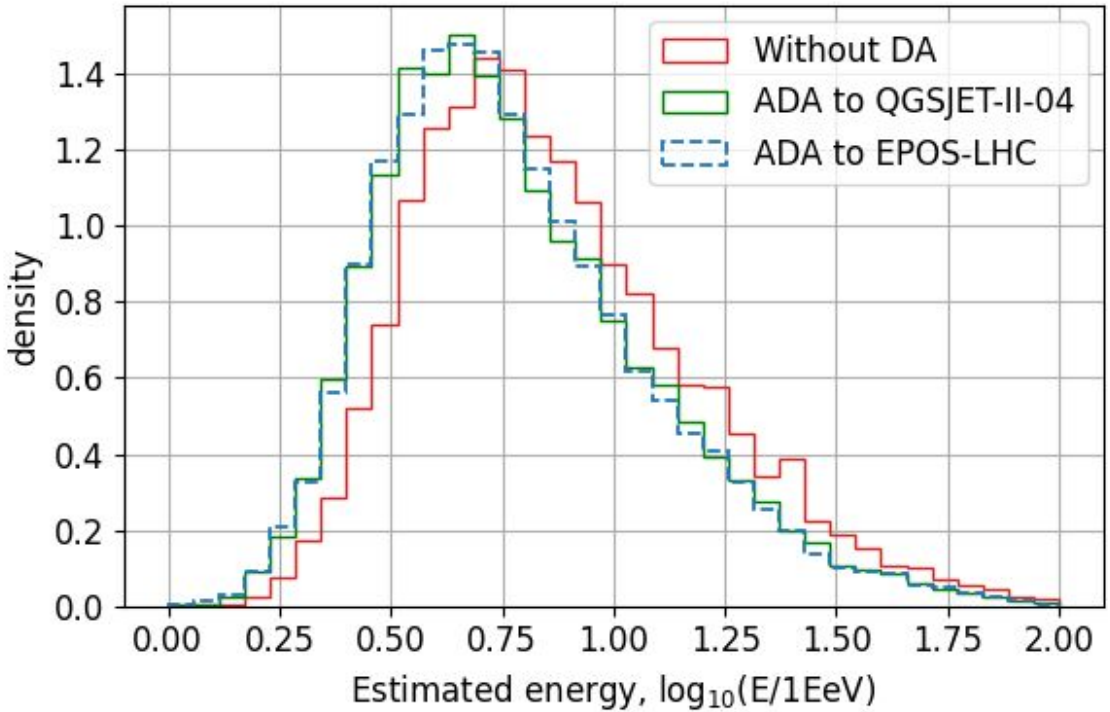


Train model on HiRes MC, try to reconstruct modified spectrum.

Both DA and ADA correctly reconstruct disturbed MC spectra

DA and choice of hadronic interaction model

Estimated TA SD spectra



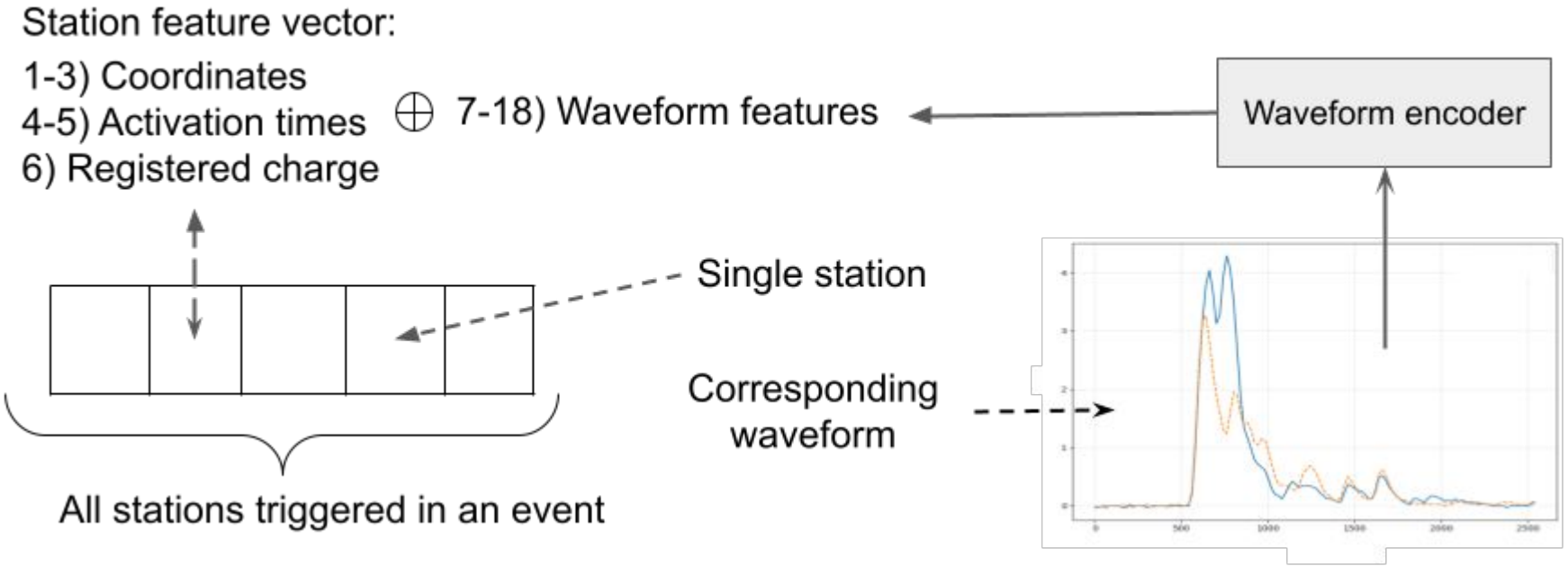
Predictions shift due to DA is similar to that of Changing hydronic interaction model.

Conclusion

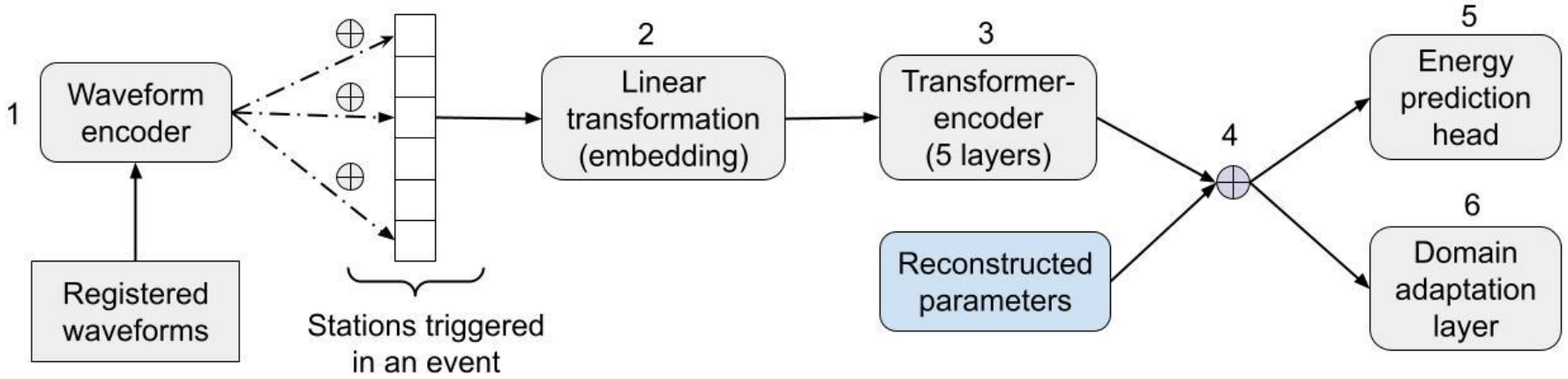
- Power and weakness of neural networks: sensitivity to fine details
  - Complex NNs require good MC-data agreement for reliability
- Adaptive domain adaptation is important for reliable energy reconstruction:
  - Converges to the same spectrum
- Systematic error due to domain shift is comparable to other uncertainties
- Domain adaptation disturbs data but in a way that MC and experimental data have consistent physics (P(z|E)).
- Preliminary: good SD/FD agreement.

Neural network

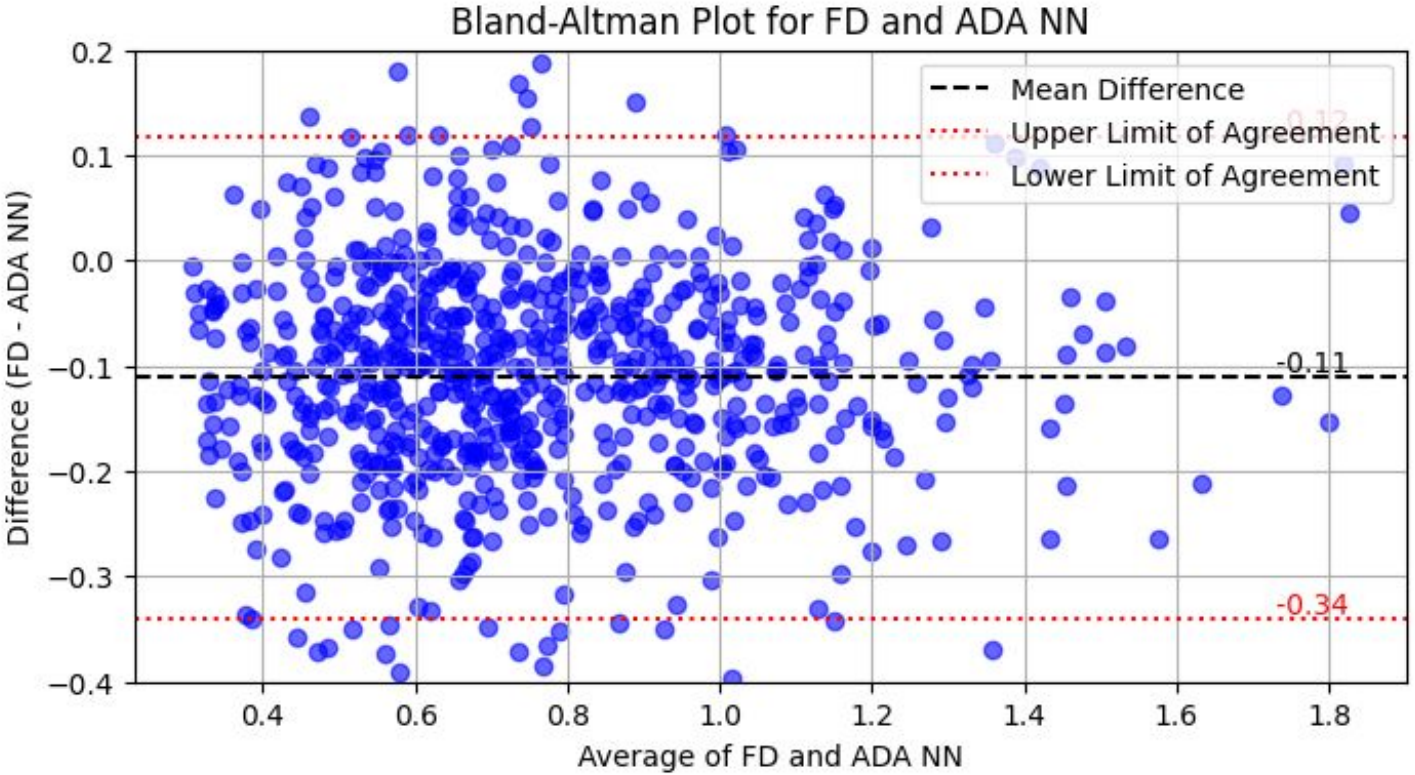
Data representation



NN architecture



Preliminary SD/FD comparison



Good agreement with FD: residual variance 0.0075 (0.01 for standard reco)