Machine Learning

Current Projects @ Belle II

Isabel Haide
isabel.haide@kit.edu

Lea Reuter
lea.reuter@kit.edu

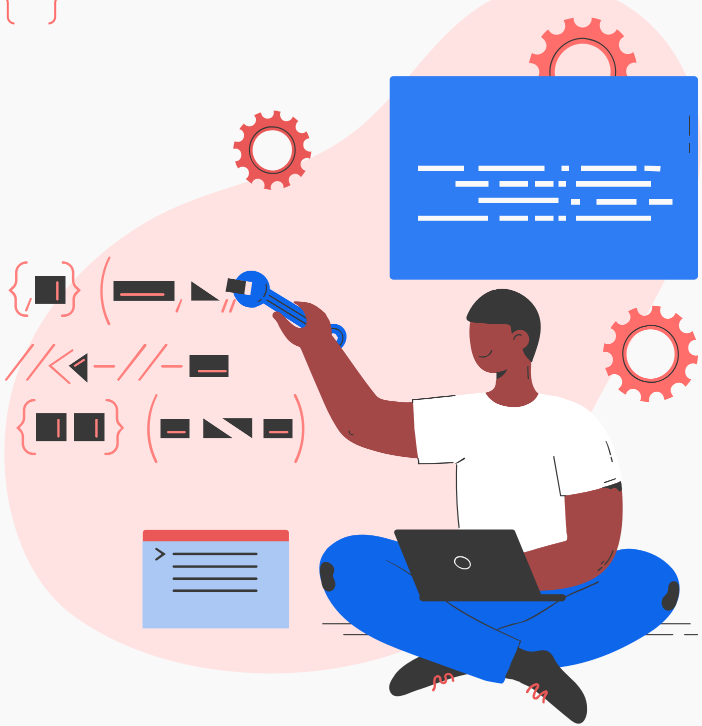
Karlsruhe Institute of Technology

Institute of Experimental Particle Physics

Table of contents

01

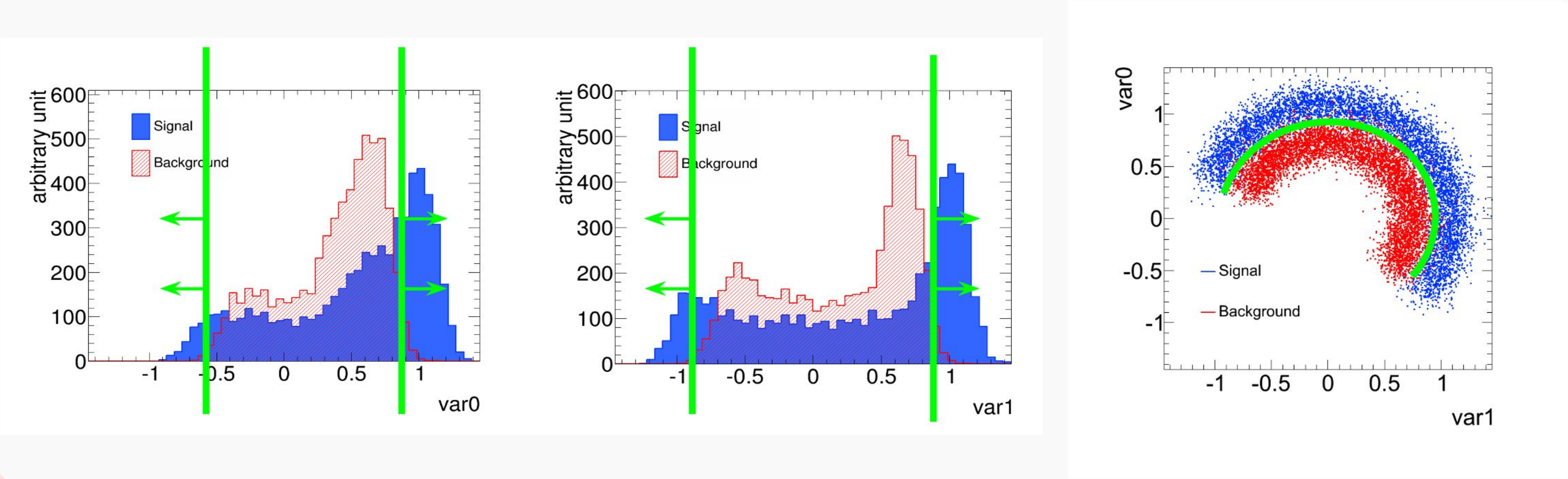
Introduction to Belle II MVA



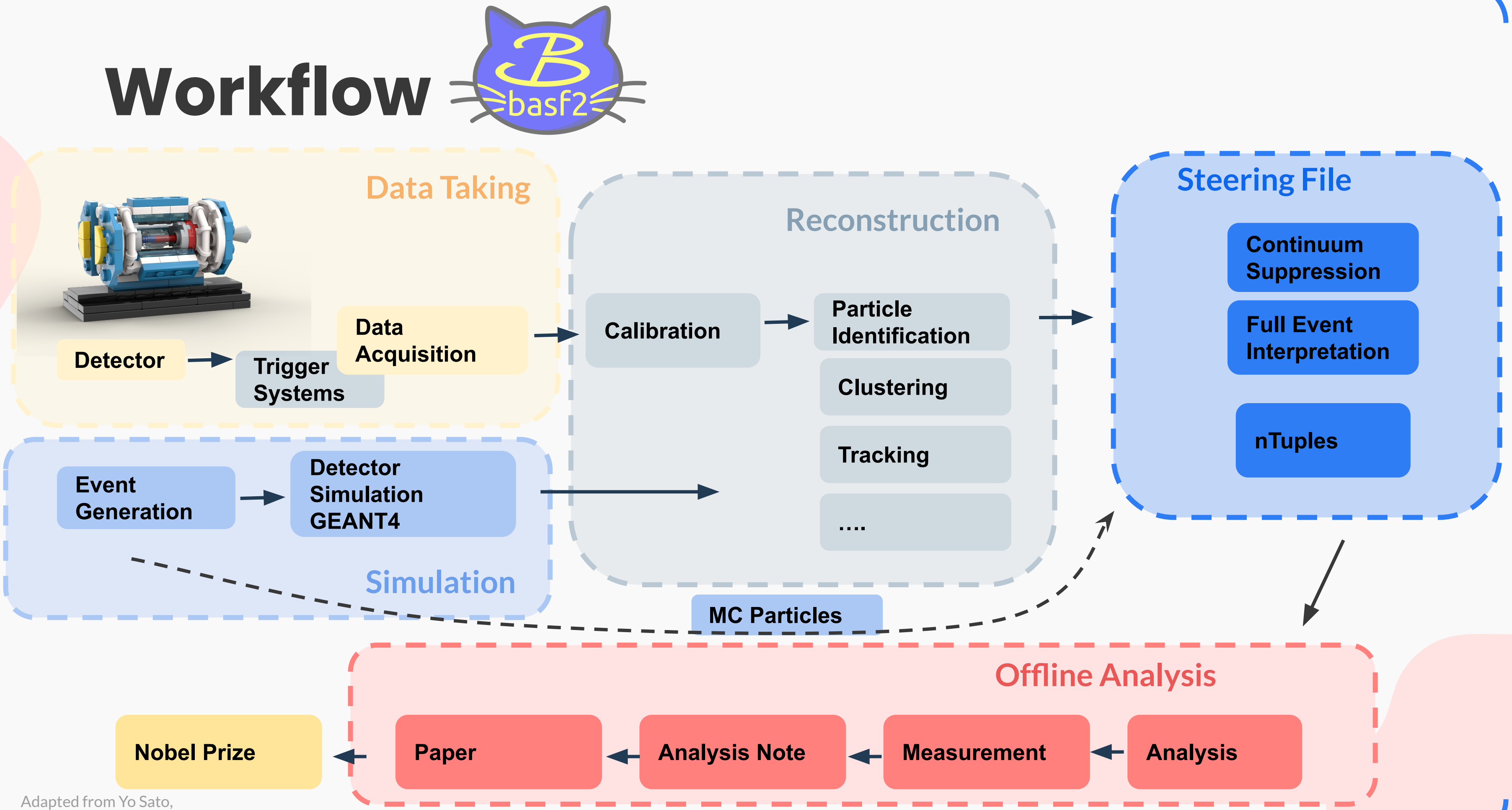
Why Multivariate?

Why not simply cut on variables?

- Correlations not visible



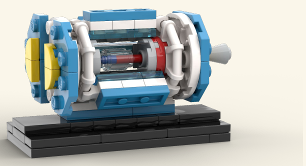
Workflow



Adapted from Yo Sato,
Introduction to analysis package

Workflow

Data Taking



Detektor

Trigger Systems

Data Acquisition

Calibration

Event Generation

Detector Simulation GEANT4

Simulation

MC P

g File

ntinuum
ppression

l Event
erpretation

uples

Paper

Analysis I

{ Every analysis ever }

Belle II Machine Learning

[ ] Reconstruction
( ) Event Interpretation
{ } Analysis
< > Simulation

Boosted Decision Trees

(FEI)
[Fake Photon Suppression]

Classic Neural Networks

[PID]
[Slow Pion Rescue]
{ Exploring ML for Analysis }
[Neural Z Trigger]
(Continuum Suppression)
[BGNet]

Generative Adversarial Networks

< PXD >
< TOP >
< ECL >

Transformers

(Deep FEI)
(Hyper Tagging)
(Deep Continuum Suppression)

Autoencoder

{ Anomaly Detection }
(graFEI)
< TOP >

Convolutional Neural Networks

[PIDs with CNNs]
{ New Physics Searches }

Graph Neural Networks

(graFEI)
(Flavor Tagger)
[CDC Tracking]
[ECL Clustering]

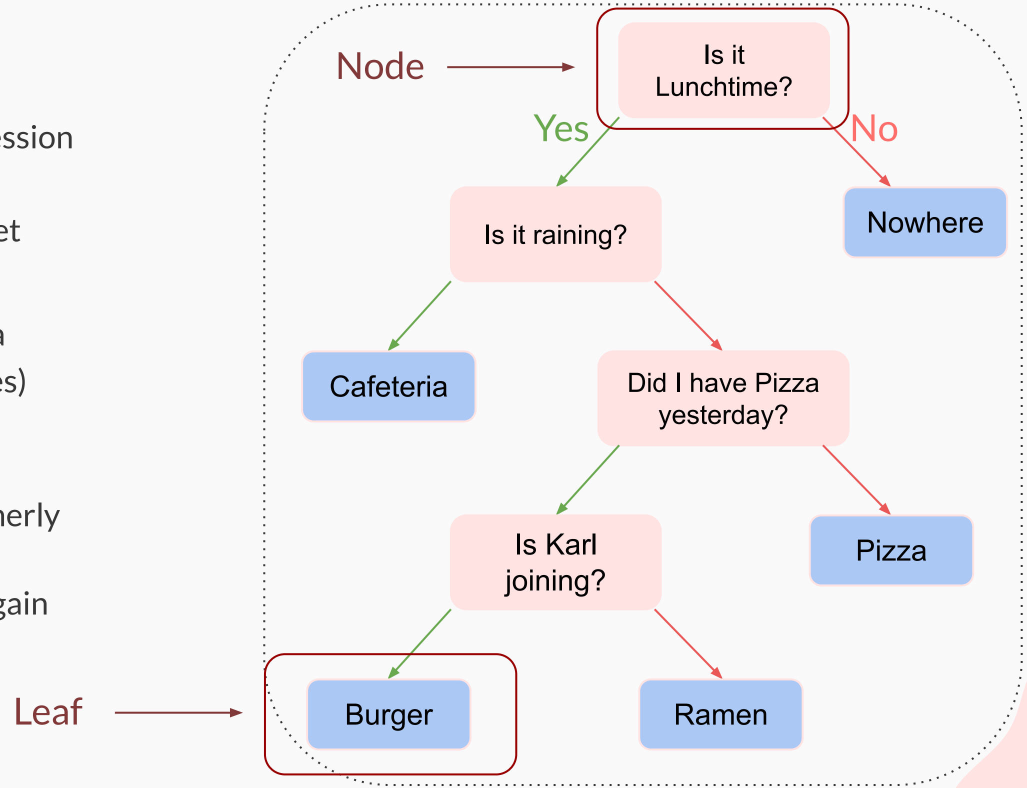
02

Overview over Belle II Projects

Decision Trees

- Popular for classification and regression tasks
  - Easy to understand/interpret
  - Fast training
  - Good with multivariate data
- Consecutive set of questions (nodes)
- Final verdict is reached after given maximum of nodes (leaf)
- Each question depends on the formerly given answers
- Trained on maximizing separation gain between nodes

What will I eat for Lunch?



(Boosted) Decision Trees

Is it Signal or Background?

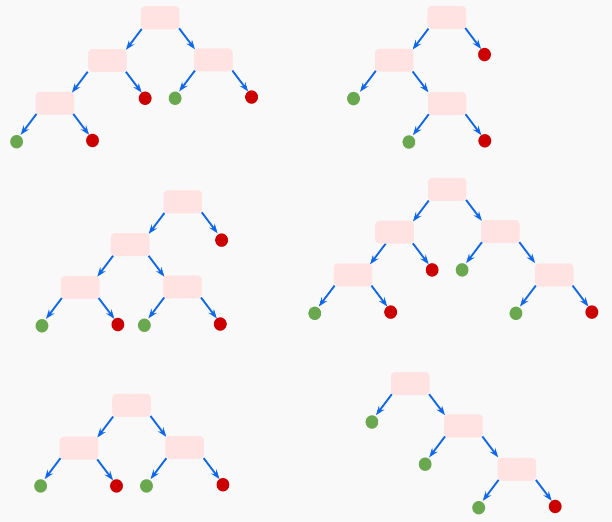
- Single trees are often not very strong
- Combination of different trees

  -> Random Forest

- Output is decided by majority vote

- Boosting:
  - Misclassified events are weighted higher
  - Future trees concentrate on misclassified events
  - Easy to overtrain

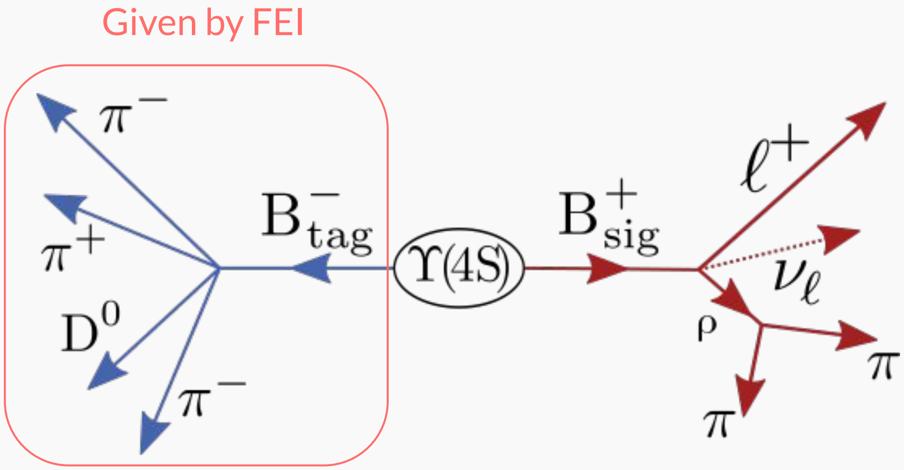
- Pruning: Reduce tree size by removing redundant nodes



-> Majority Vote Decision

Example: Full Event Interpretation (FEI)

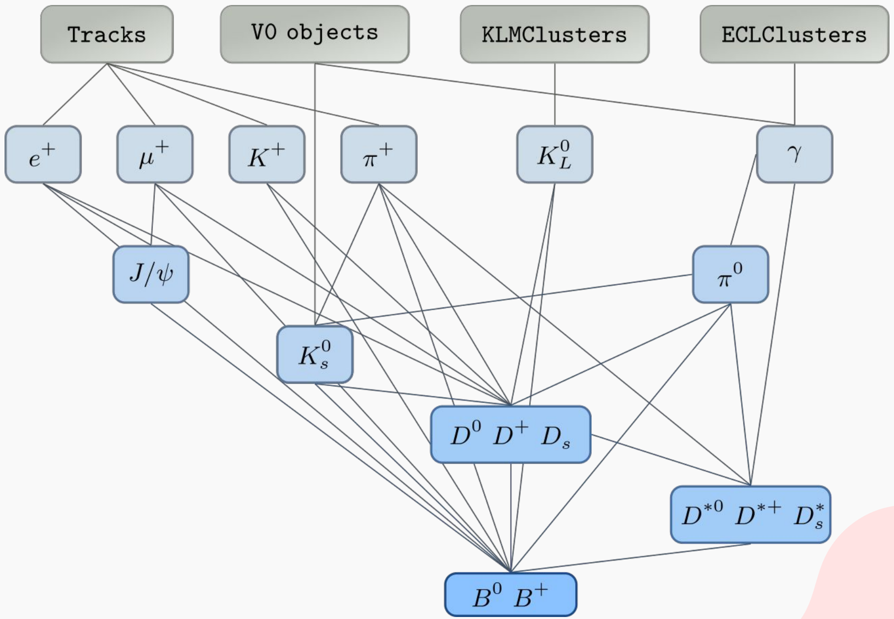
- Idea: Tagging B-meson decay chains through BDTs



Thomas Keck et al.: arXiv:1807.08680

Example: Full Event Interpretation

- Idea: Tagging B-meson decay chains through BDTs
- Hierarchical Structure



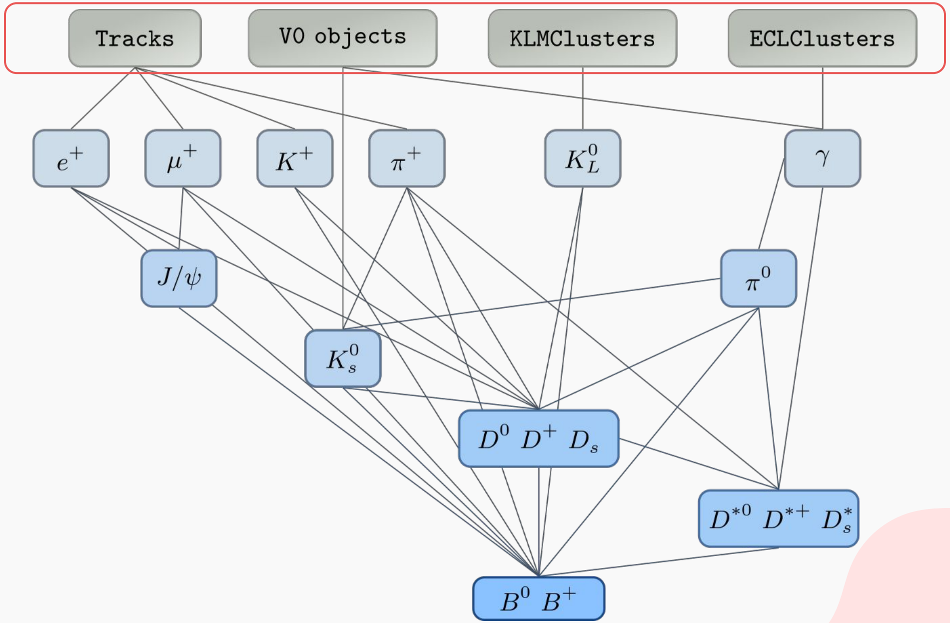
- Probability of each candidate being correct is estimated by BDTs

Thomas Keck et al.: arXiv:1807.08680

Example: Full Event Interpretation

- Idea: Tagging B-meson decay chains through BDTs
- Hierarchical Structure:
    1. Construction of final-state particles using reconstructed tracks and clusters

- Probability of each candidate being correct is estimated by BDTs

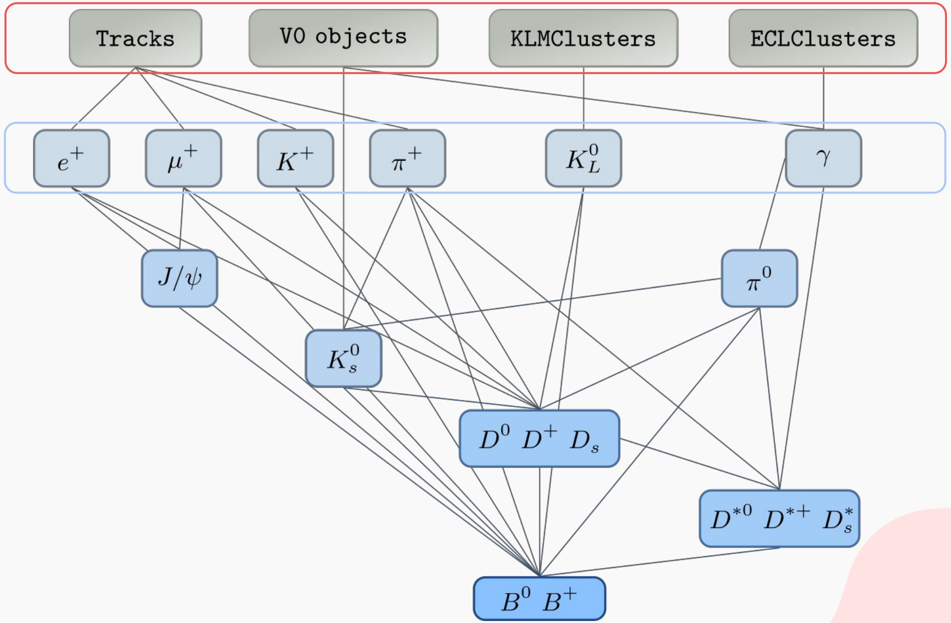


Thomas Keck et al.: arXiv:1807.08680

Example: Full Event Interpretation

- Idea: Tagging B-meson decay chains through BDTs
- Hierarchical Structure:
    1. Construction of final-state particles using reconstructed tracks and clusters
    2. Combination of final-state particles to intermediate particles

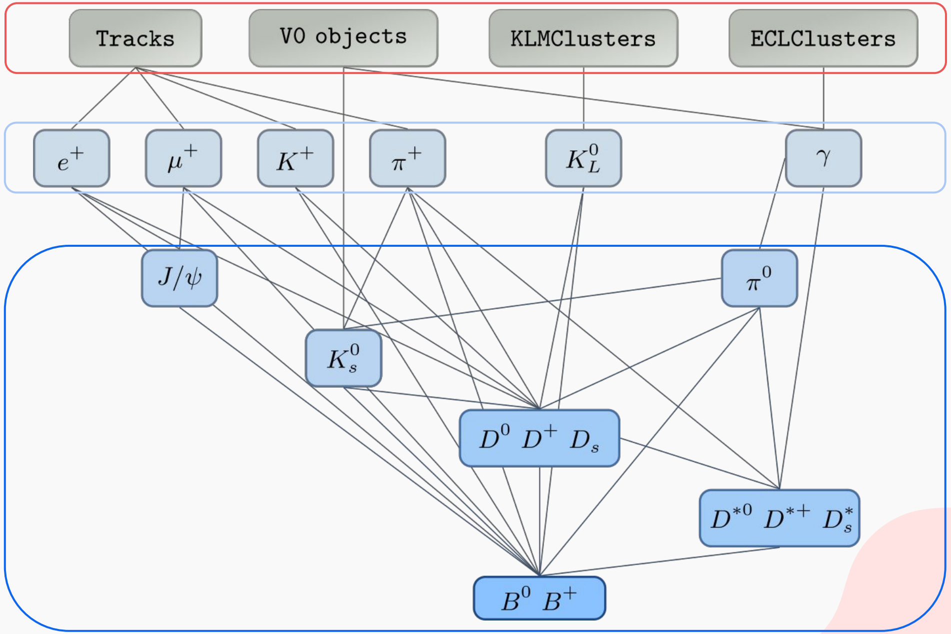
- Probability of each candidate being correct is estimated by BDTs



Thomas Keck et al.: arXiv:1807.08680

Example: Full Event Interpretation

- Idea: Tagging B-meson decay chains through BDTs
- Hierarchical Structure:
    1. Construction of final-state particles using reconstructed tracks and clusters
    2. Combination of final-state particles to intermediate particles



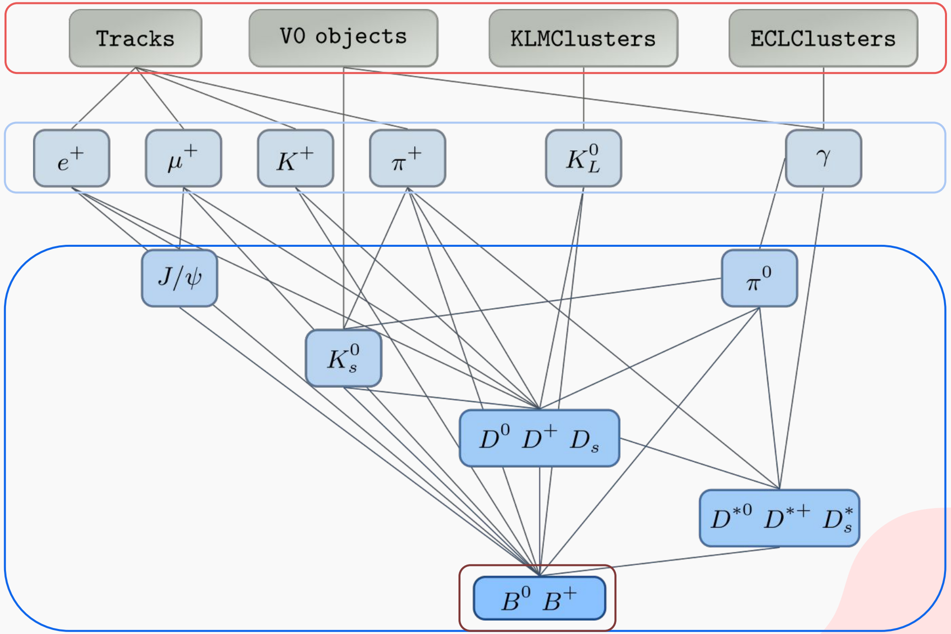
- Probability of each candidate being correct is estimated by BDTs
    - Classifier for intermediate particles uses signal probability of daughter particles as input features

Thomas Keck et al.: arXiv:1807.08680

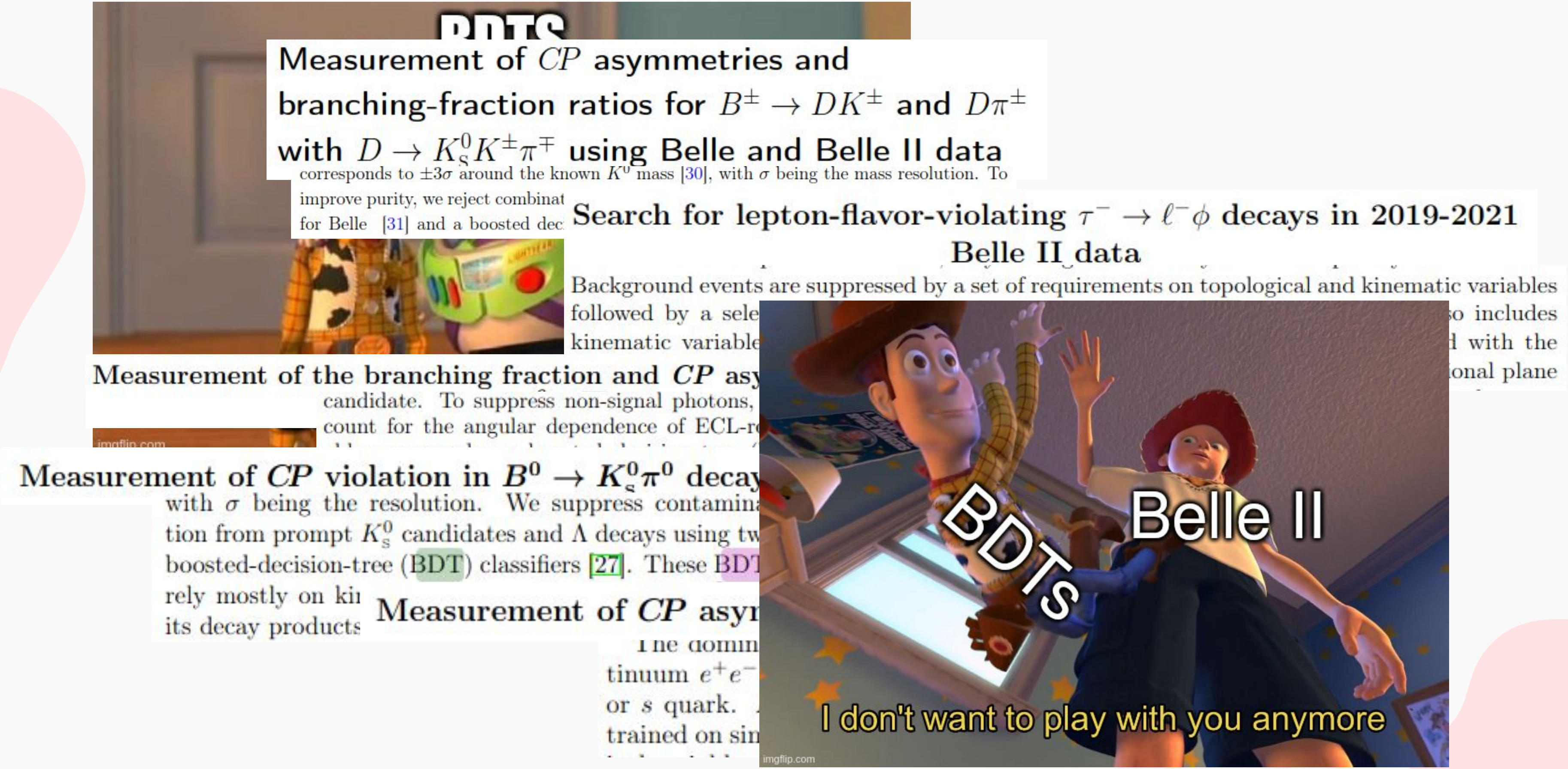
Example: Full Event Interpretation

- Idea: Tagging B-meson decay chains through BDTs
- Hierarchical Structure:
    1. Construction of final-state particles using reconstructed tracks and clusters
    2. Combination of final-state particles to intermediate particles
    3. Repeat 2. until final B candidates are formed

- Probability of each candidate being correct is estimated by BDTs
    - Classifier for intermediate particles uses signal probability of daughter particles as input features

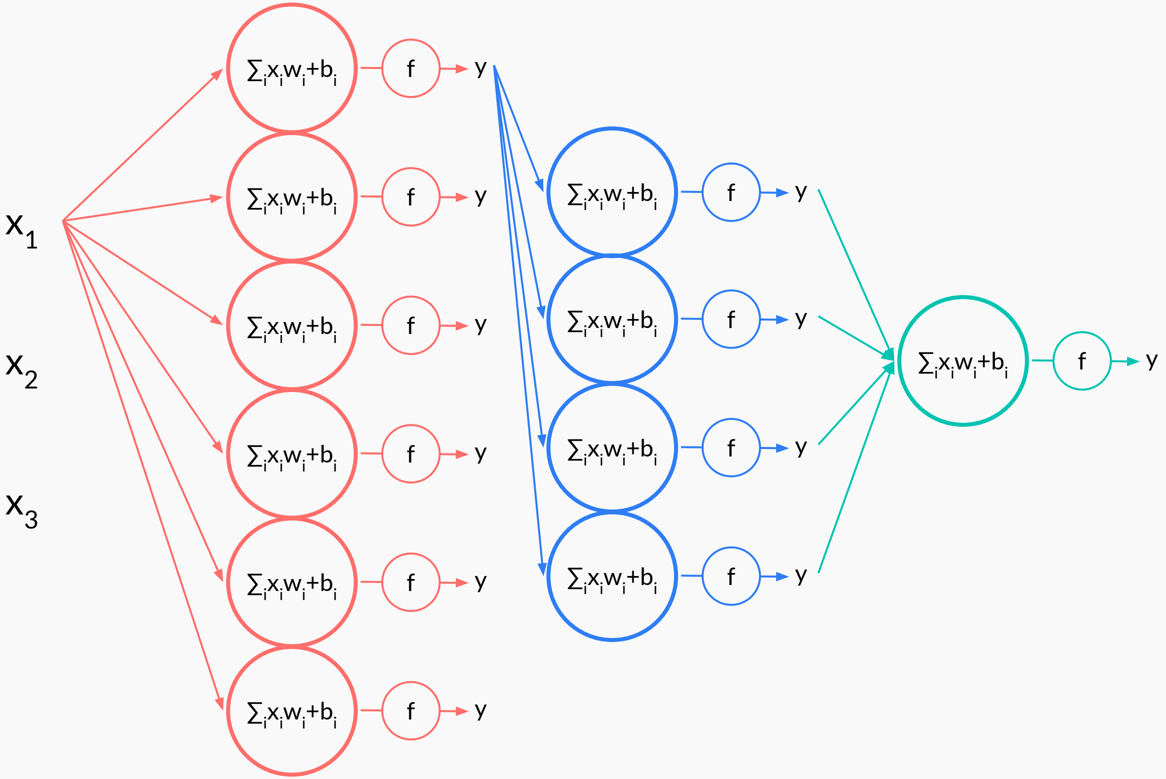


Thomas Keck et al.: arXiv:1807.08680


BDTS
Measurement of CP asymmetries and branching-fraction ratios for $B^{\pm} \to DK^{\pm}$ and $D\pi^{\pm}$ with $D \to K^0_S K^{\pm}\pi^{\mp}$ using Belle and Belle II data
corresponds to $\pm 3\sigma$ around the known K^0 mass [30], with σ being the mass resolution. To
improve purity, we reject combinat
for Belle [31] and a boosted dec
Search for lepton-flavor-violating $\tau^- \to \ell^- \phi$ decays in 2019-2021 Belle II data
Background events are suppressed by a set of requirements on topological and kinematic variables
followed by a sele
kinematic variable
so includes
l with the
ional plane
Measurement of the branching fraction and CP asy
candidate. To suppress non-signal photons,
count for the angular dependence of ECL-re
imgflip.com
Measurement of CP violation in $B^0 \to K^0_S \pi^0$ decay
with σ being the resolution. We suppress contamina
tion from prompt K^0_S candidates and Λ decays using tw
boosted-decision-tree (BDT) classifiers [27]. These BDT
rely mostly on ki
its decay products
Measurement of CP asyı
The domin
tinuum e^+e^-
or s quark.
trained on sin
BDTs
Belle II
I don't want to play with you anymore
imgflip.com

Classic Neural Networks

~~Boosted Decision Trees~~

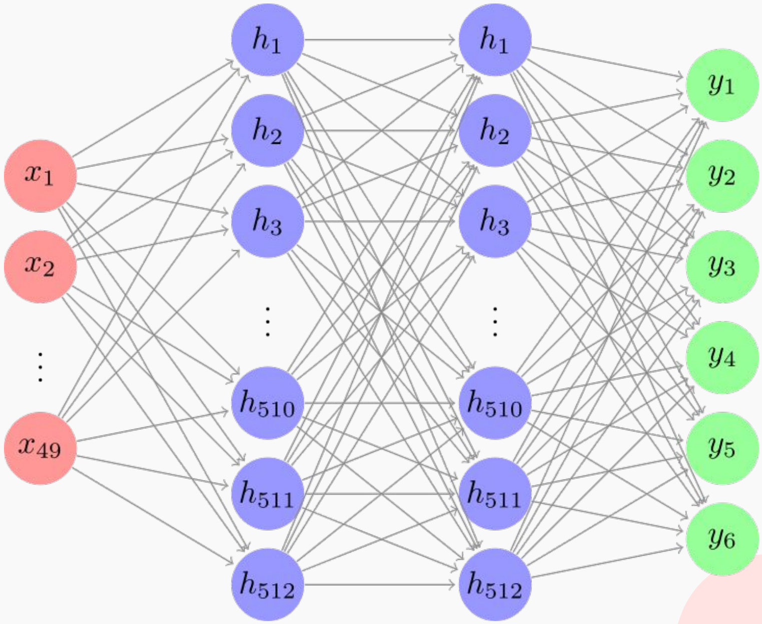


x_i: Inputs
w_i: Learnable Weights
b_i: Learnable Bias
f(x): Activation function
y: Output

Lea Reuter (lea.reuter@kit.edu), Isabel Haide (isabel.haide@kit.edu)

Classic Neural Networks

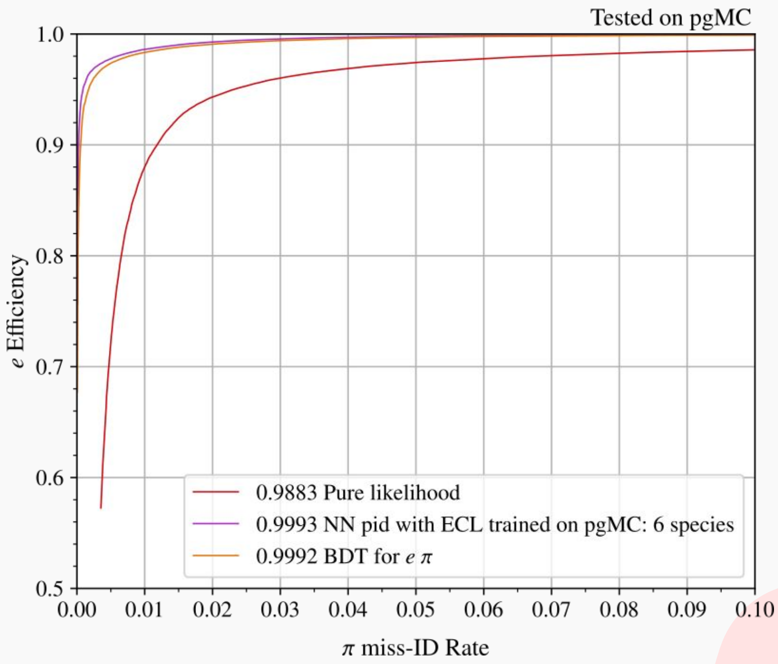
- Example: Improvement of Particle Identification at Belle II
- Currently likelihood of particle hypothesis is combined out of 36 subdetector likelihoods
    - $\mathcal{L}(h) = \prod_D \mathcal{L}_D(h)$
- NN solution can learn correlations among $\mathcal{L}_D(h)$
- Predict probabilities for all 6 particle species hypothesis (e, μ, π, K, p, d)
- Combination of high-level information as input:
    - $\mathcal{L}_D(h)$
    - Track momentum
    - Track charge
    - ECL Variables



Stefan Wallner, Xavier Simo: Particle identification at Belle II using Neural Networks

Classic Neural Networks

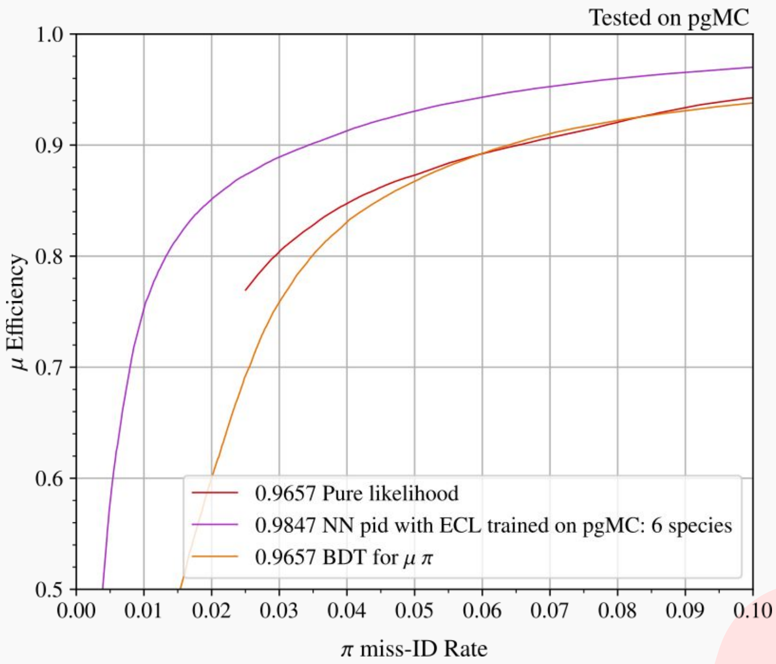
- Example: Improvement of Particle Identification at Belle II
- Currently likelihood of particle hypothesis is combined out of 36 subdetector likelihoods
  - $\mathcal{L}(h) = \prod_D \mathcal{L}_D(h)$
- NN solution can learn correlations among $\mathcal{L}_D(h)$
- Predict probabilities for all 6 particle species hypothesis (e, μ, π, K, p, d)
- Combination of high-level information as input:
  - $\mathcal{L}_D(h)$
  - Track momentum
  - Track charge
  - ECL Variables
- Improved performances for all particle hypotheses



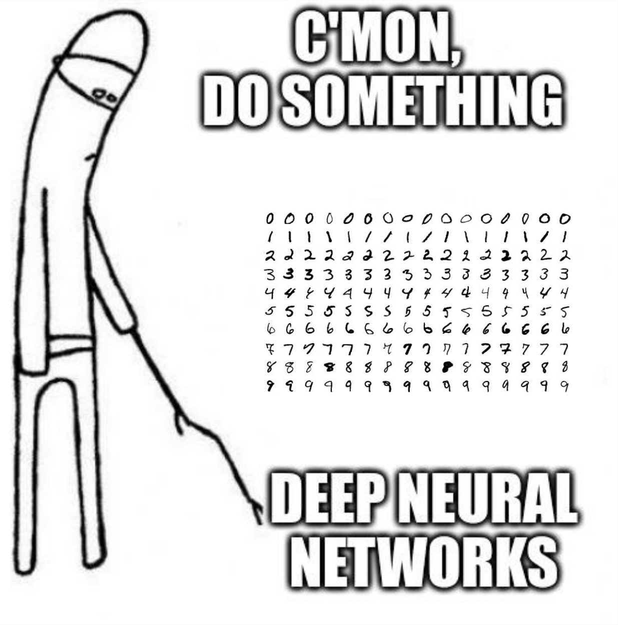
Stefan Wallner, Xavier Simo: Particle identification at Belle II using Neural Networks

Classic Neural Networks

- Example: Improvement of Particle Identification at Belle II
- Currently likelihood of particle hypothesis is combined out of 36 subdetector likelihoods
    - $\mathcal{L}(h) = \prod_D \mathcal{L}_D(h)$
- NN solution can learn correlations among $\mathcal{L}_D(h)$
- Predict probabilities for all 6 particle species hypothesis (e, μ, π, K, p, d)
- Combination of high-level information as input:
    - $\mathcal{L}_D(h)$
    - Track momentum
    - Track charge
    - ECL Variables
- Improved performances for all particle hypotheses



Convolutional Neural Networks



MNIST Results

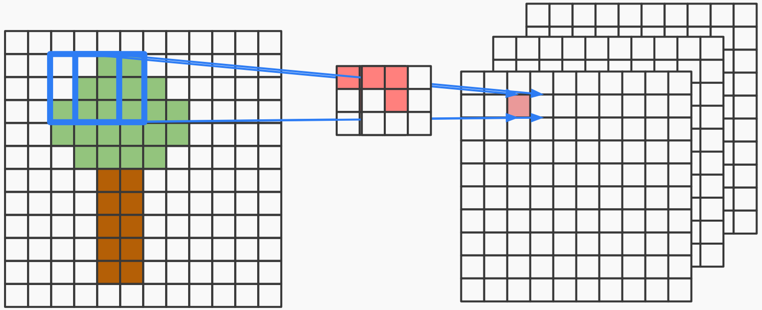
Architecture	Test Error	Weights (10^6)
DNN[1]	0.35 %	12.11
CNN[2]	0.25 %	5.4

- Deep neural networks scale badly with high amount of inputs
    - Images with 32 pixels already have 1024 inputs
- DNNs don’t take locality (nearby pixels are more strongly correlated) and translation invariance (meaningful patterns can occur anywhere) into account

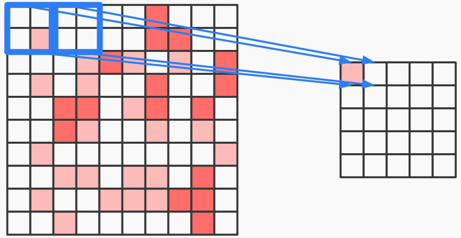
1: https://arxiv.org/abs/1003.0358
2: https://arxiv.org/abs/1608.06037

Convolutional Neural Networks

Convolutional Layer



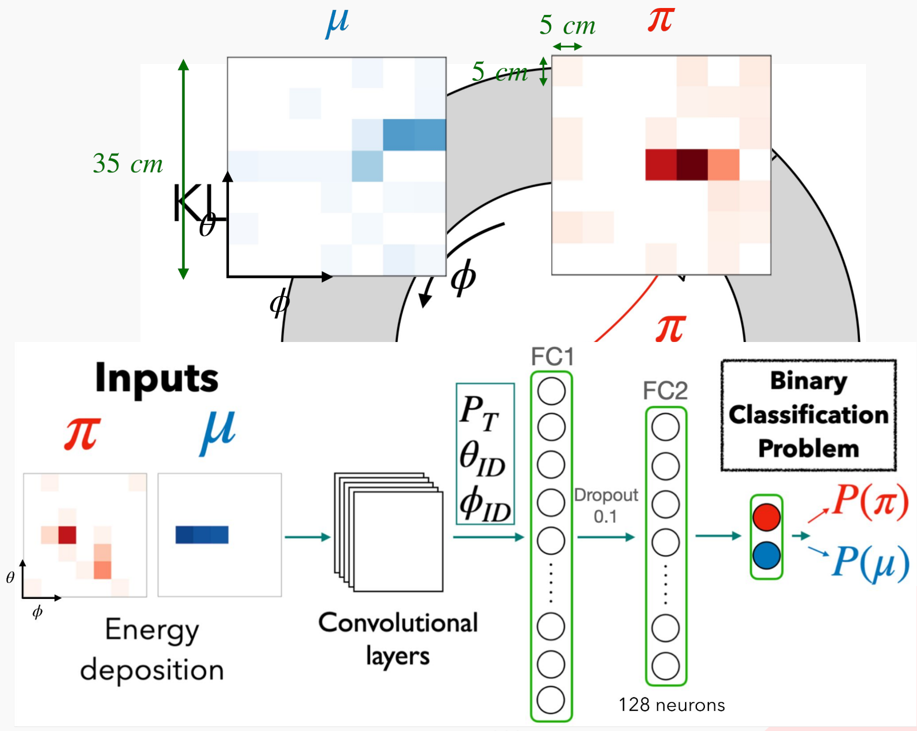
Pooling Layer



- CNNs are very good for image recognition:
  - Input stays in 2D shape instead of being flattened
  - Weights are shared in convolutions -> translational invariance
  - Nearby pixels are more highly correlated due to convolutions in kernels
- Convolutional layer:
  - Apply filter of size NxN to input
  - Slide filter over entire input layer with fixed stride
- Pooling layer:
  - Reduce layer size by downsampling
  - Different pooling types, such as max or mean pooling

Pion/Muon Identification with CNNs

- Identification of low-momenta muons relies on ECL
  - Better muon-pion separation useful for e.g. leptonic tau decays
- Default PID in ECL uses E/p
  - Not very powerful for low momentum pion-muon separation
- Energy deposition patterns for pions are more dispersed than muons
  - Neural network can employ pattern recognition
- Energy deposition in 7x7 crystals can be treated as images
  - Two separate CNNs for positive and negative tracks
  - CNNs outperform both default and BDT method

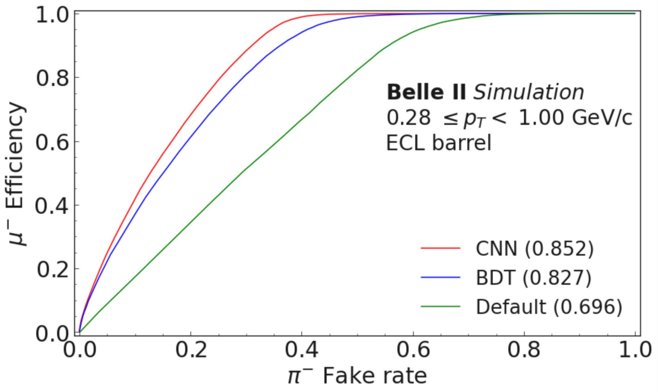


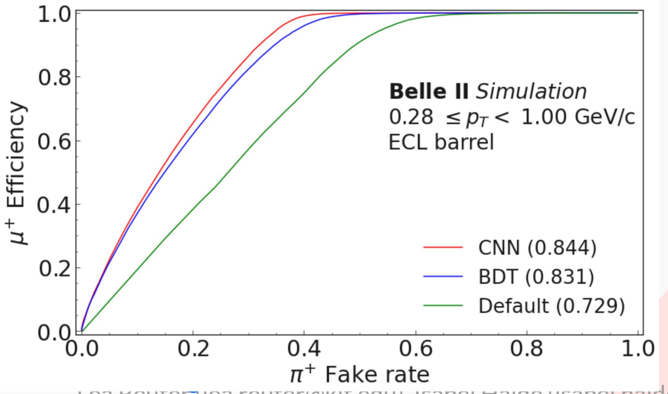
θ_{ID} and ϕ_{ID} are crystal numbers.

Abtin Narami, Torben Ferber: Identification of pions and muons with the Belle II calorimeter using convolutional neural network

Pion/Muon Identification with CNNs

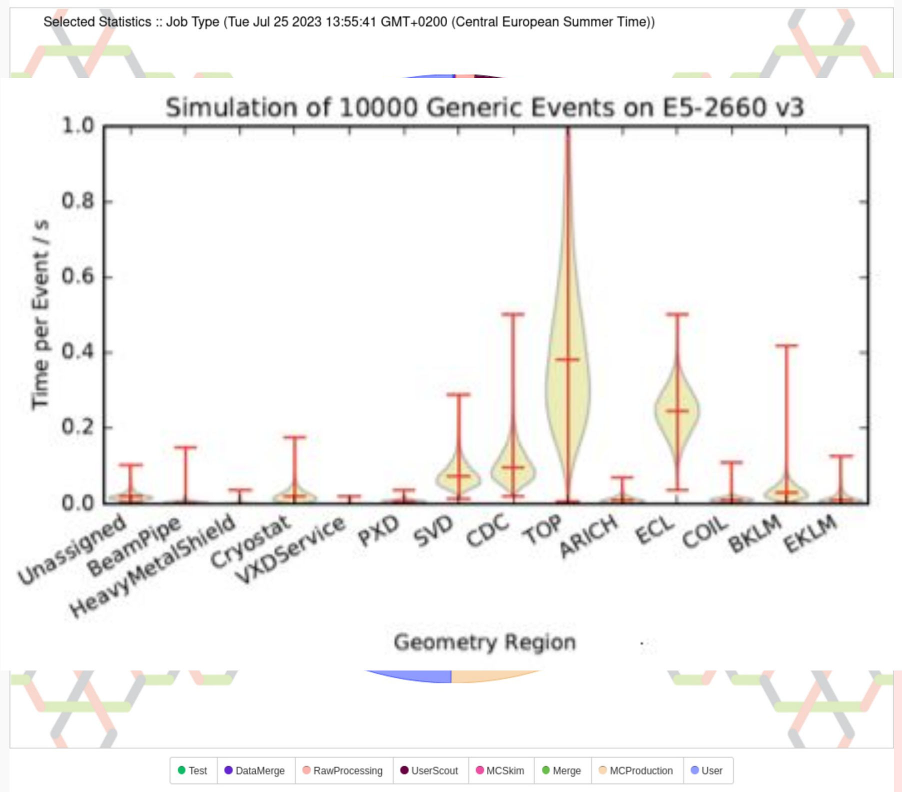
- Identification of low-momenta muons relies on ECL
  - Better muon-pion separation useful for e.g. leptonic tau decays
- Default PID in ECL uses E/p
  - Not very powerful for low momentum pion-muon separation
- Energy deposition patterns for pions are more dispersed than muons
  - Neural network can employ pattern recognition
- Energy deposition in 7x7 crystals can be treated as images
  - Two separate CNNs for positive and negative tracks
  - CNNs outperform both default and BDT method





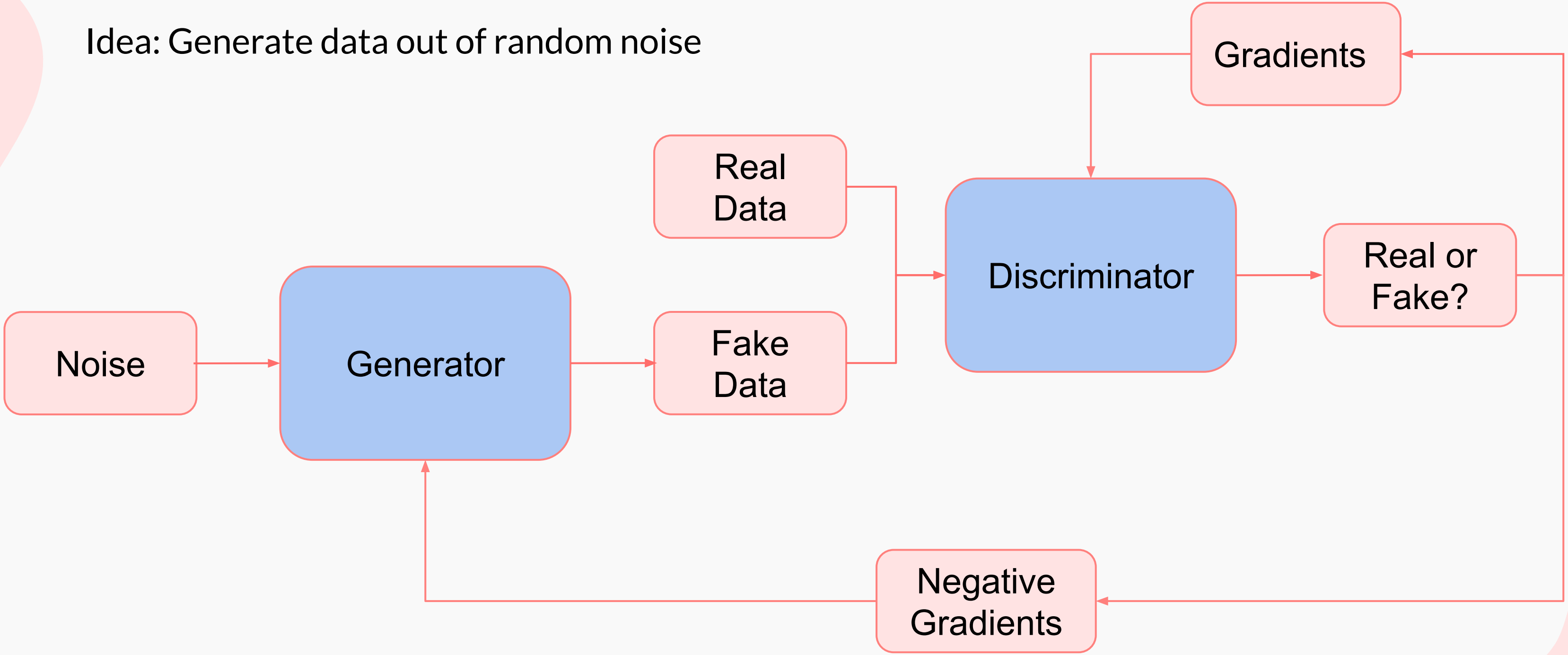
Generative Adversarial Networks

- Big chunk of Belle II computing is MC production
- With higher needs for MC data, computing time and resources might become bottleneck
  - Improvement of MC generation necessary
- TOP and ECL have highest simulation time of all subdetectors
- Improvement of MC simulation through generative ML algorithms
  - Generative adversarial networks
  - Variational autoencoders



Generative Adversarial Networks

Idea: Generate data out of random noise



Generative Adversarial Networks

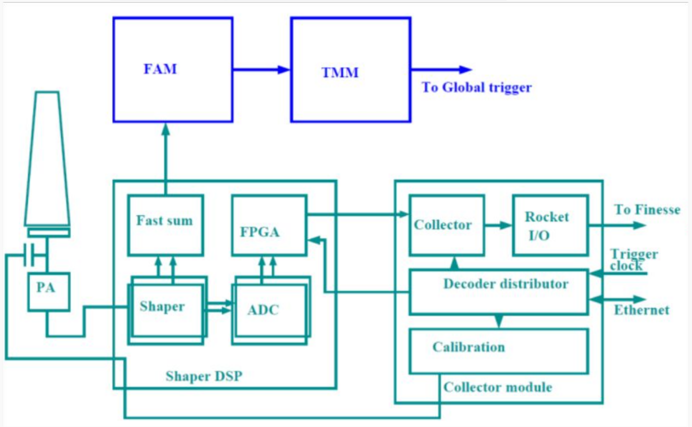
Idea: Generate data out of random noise

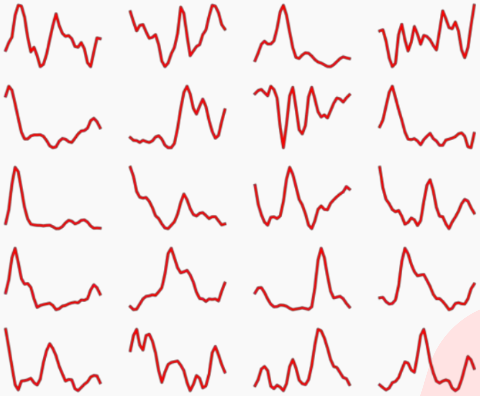
https://embracethered.com/blog/posts/2020/machine-learning-attack-series-generative-adversarial-networks-gan/

Simulation of Noise Waveforms in the ECL

- ECL consists of ~9000 crystals
- Crystal PMT measurements are digitized and a photon and hadron fit is applied
- Background waveform simulations require much data and bandwidth to store

ECL crystal electronics

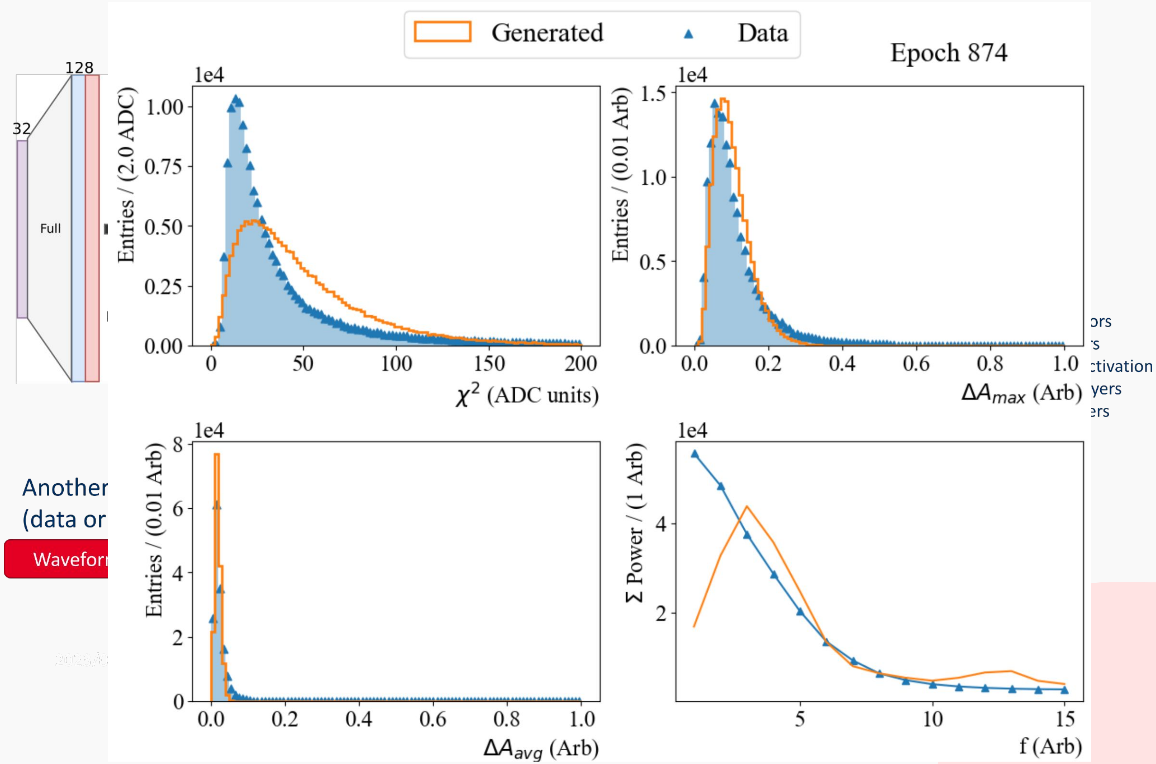




Alexandre Beaubien, John Michael Roney: Noise Waveform Generation Using GANs

Simulation of Noise Waveforms in the ECL

- Strategy: Only keep interesting (high energy, correlated backgrounds) waveforms and simulate the rest
- Train CNN GAN on generating waveforms
- Evaluate network on performance metrics
- GAN shows better performance than Autoencoders or Covariance Matrix Methods

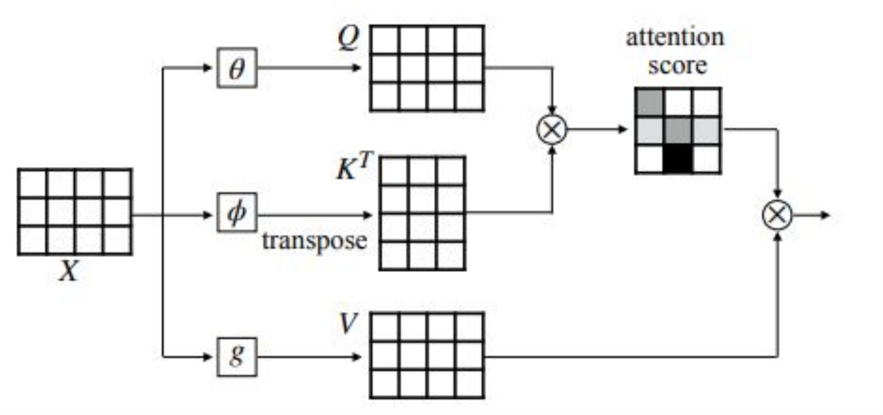


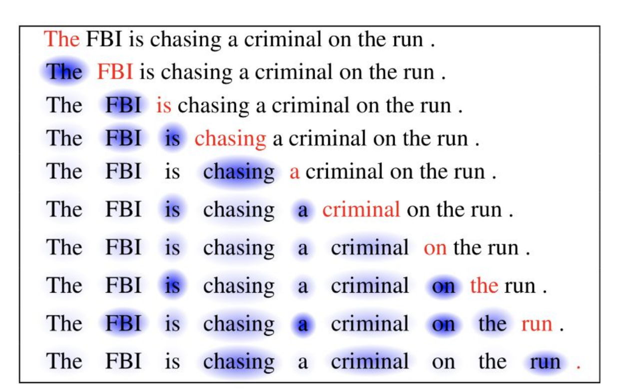
Alexandre Beaubien, John Michael Roney: Noise Waveform Generation Using GANs

Transformer

Developed for translations:

- Positional Encodings
  (position in sentence is important for the translation!)
- Attention
  (context is important)
- Self-Attention
  (synonymous words can have different meanings according to context)



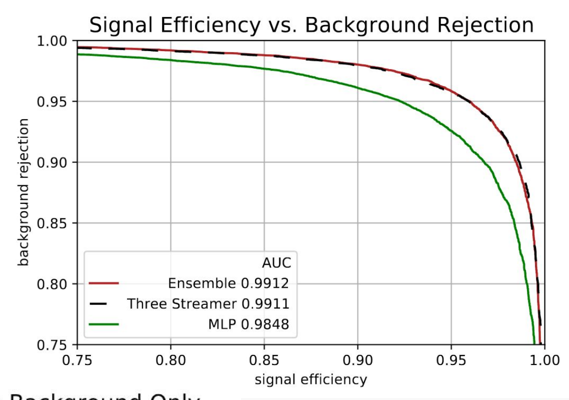


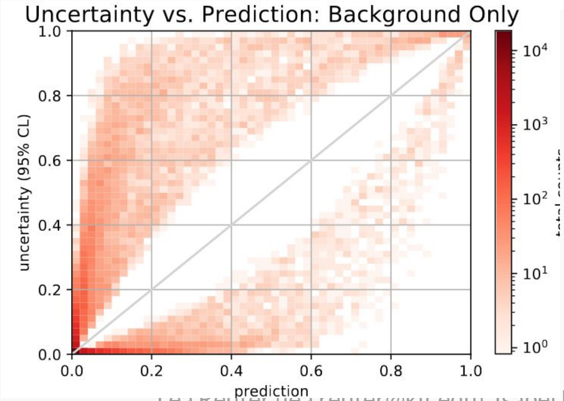
Deep Continuum Suppression

- BDTs and MLPs require fixed order for input particles, so some kind of sorting needed but this is prone to errors
  - Use self-attention-based input for permutation invariance

- Predictive uncertainties for the continuum classifications
  - Use deep ensembles

- Continuum suppression dependent on certain analysis variables, which is Introducing a bias for further studies
  - Decorrelation from analysis variables

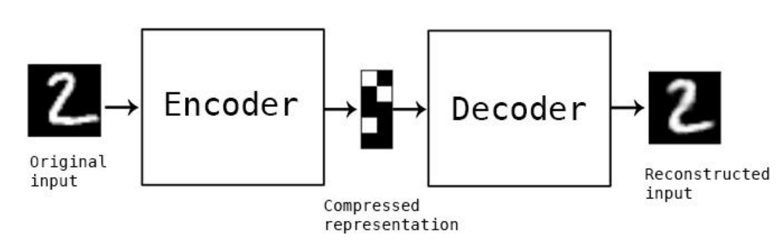


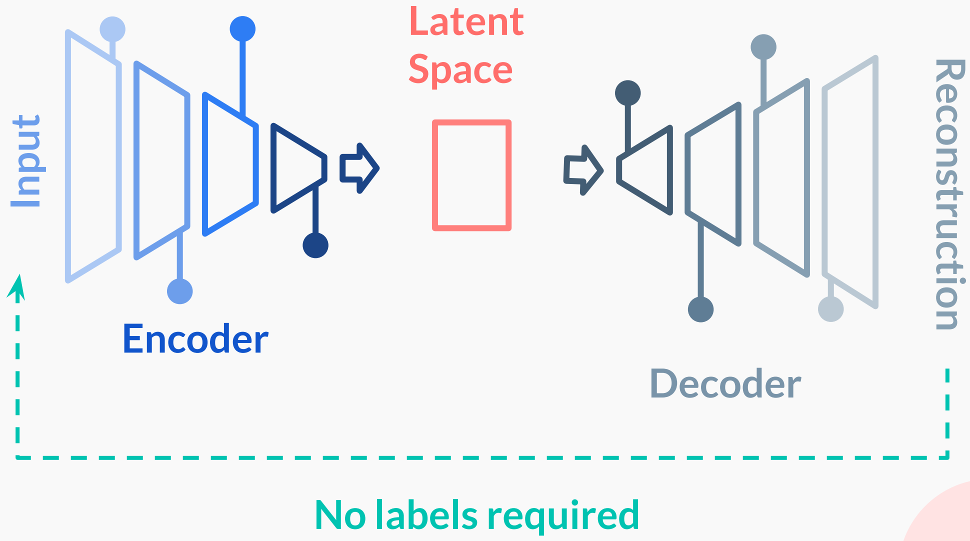


Autoencoder

- Unsupervised learning to replicate input as output
- Subnetwork **encoder** maps input to embedded representation
- Subnetwork **decoder** maps back to input space
- Lower-dimensional, condensed representation to capture important patterns

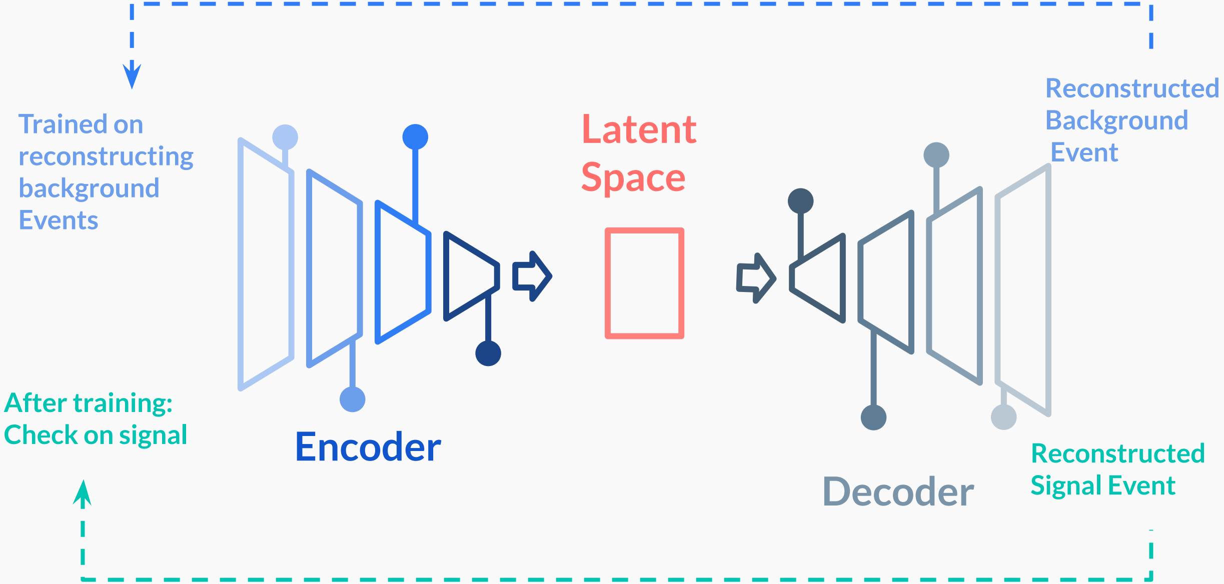
→ Network trained to learn the identity function



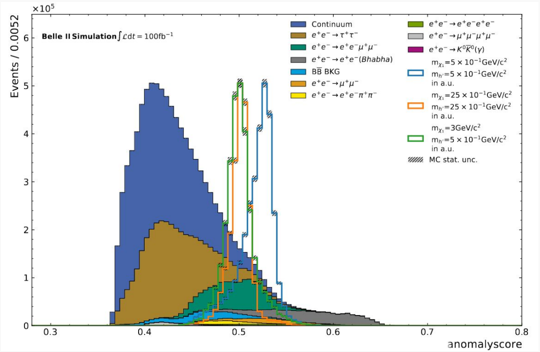


Autoencoder: Anomaly Detection

Goal: Find rare signature of new physics (e.g. dark Higgs searches) through anomaly detection in background dominated regions

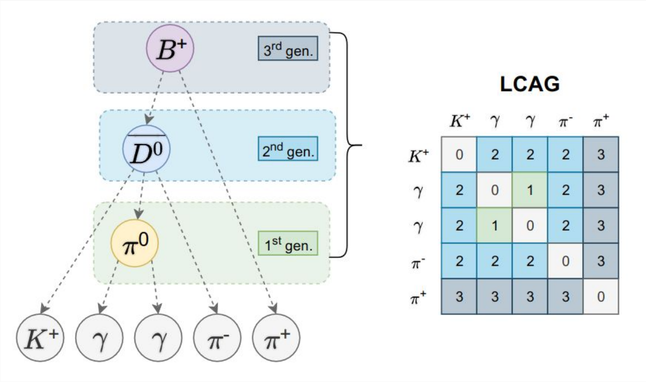


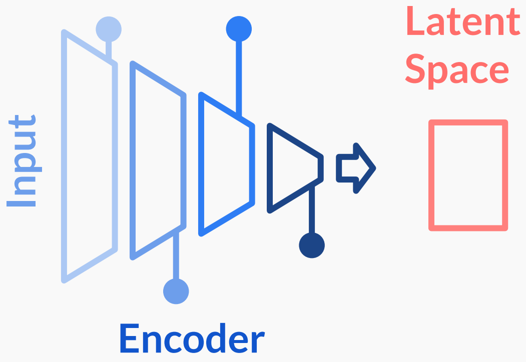
Reconstruction error used as anomaly score

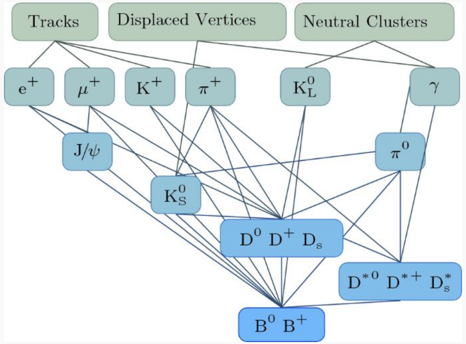


Encoder: graFEI

- Current FEI is restricted by branching fraction coverage (explicit decay structures covering O(10 000) decays so ~15%)
- Use encoder for latent space representation to predict the decay tree
- Permutation invariance







James Kahn, Ilias Tsaklidis, Oskar Taubert, Lea Reuter, Jacopo Cerasoli, Giulio Dujany, Tobias Boeckh , Arthur Thaller et al

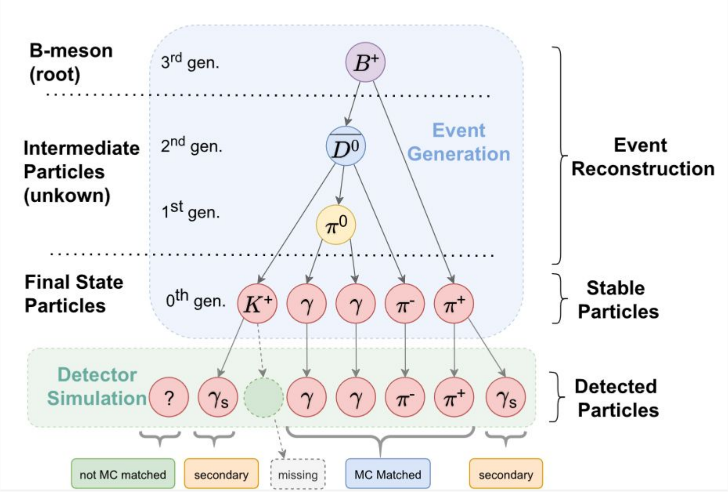
Graph Data Structure

What happens if you have varying input size, for example the different number of particles in Event that are not on grids?

Use Graph Representation:

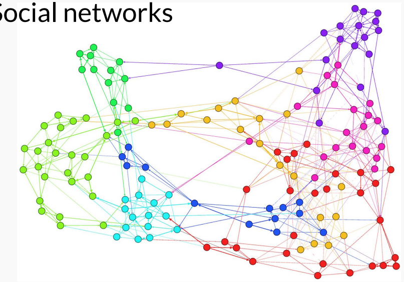
- Non-euclidian data structure
- Capture both information to the nodes and the relational information (edges)

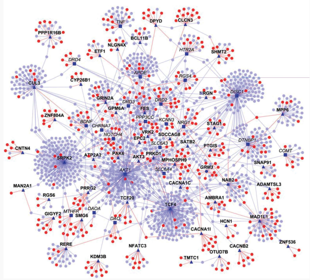
Cannot use CNNs here



Particle Decay Trees

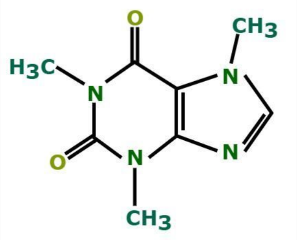
Social networks





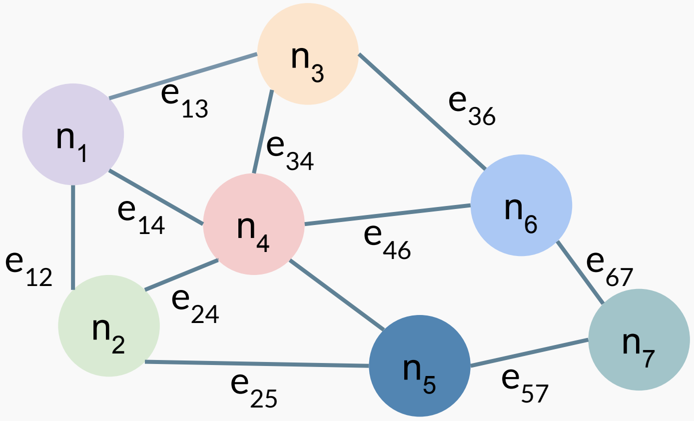
Protein Interaction

Molecules



Maps

Graphs



Graphs are build with:

Nodes n_i

- In our example particles

With node features

- Energy, momentum, charge, PID ....

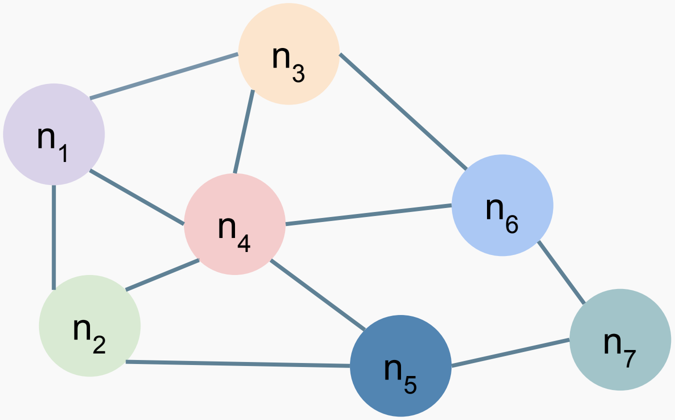
Edges e_{ij}

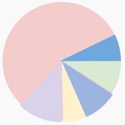
- Relations between particles

With edge features

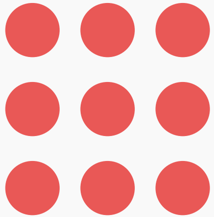
- Angle between two particles, distance, …

Graph Convolutional Networks




Red node after first update step

- Generalization of Convolutional neural networks to graph-structured data



- Exchange information between nodes
- One-dimensional edge features as edge-weights possible

Current Flavor Tagger

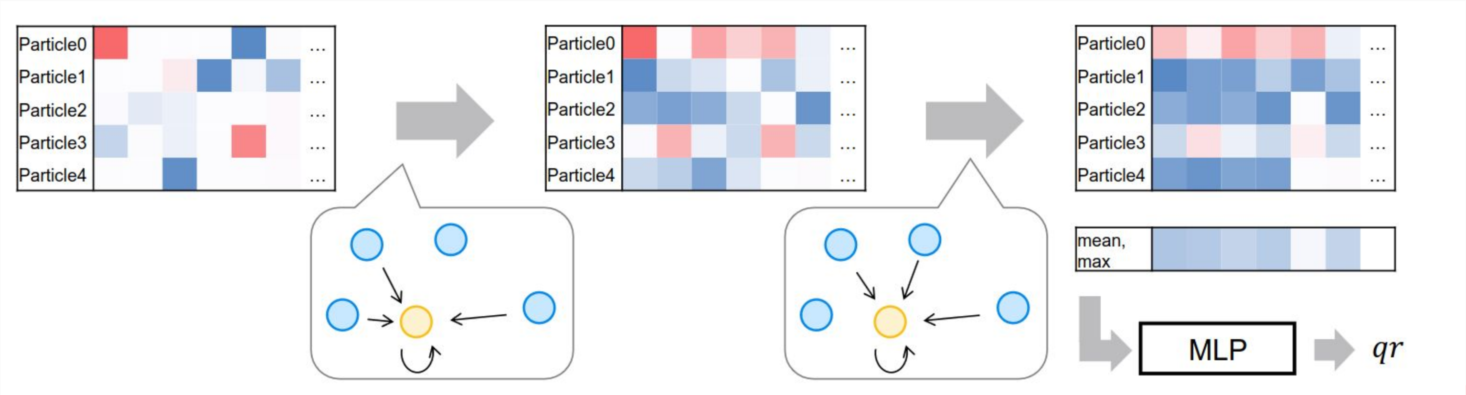
- Identify B-flavor from a single particle in the rest of event *tag-side) in 13 categories
- Combine the categories output to provide final output
- While each particle has 13 outputs, only the best scores for each category are used
- Information loss
  - Don`t know the category score of other particles
  - Don`t know which particle has the best score

Event #	Category's output, qp						
	Electron	Muon	Kaon	SlowPion	Int-electron	Int-muon	...
Particle0	**0.99**	0.00	0.01	0.00	**-0.99**	0.00	...
Particle1	0.00	0.00	0.12	**-0.96**	-0.04	**-0.51**	...
Particle2	-0.01	**-0.15**	-0.10	0.00	0.00	0.03	...
Particle3	-0.34	0.00	-0.09	0.00	0.80	0.03	...
Particle4	0.00	0.00	**-0.94**	0.00	0.00	0.02	...
Best	**0.99**	**-0.15**	**-0.94**	**-0.96**	**-0.99**	**-0.51**	...

FastBDT / MLP → qr

GNN-based Flavor Tagger, GFlaT

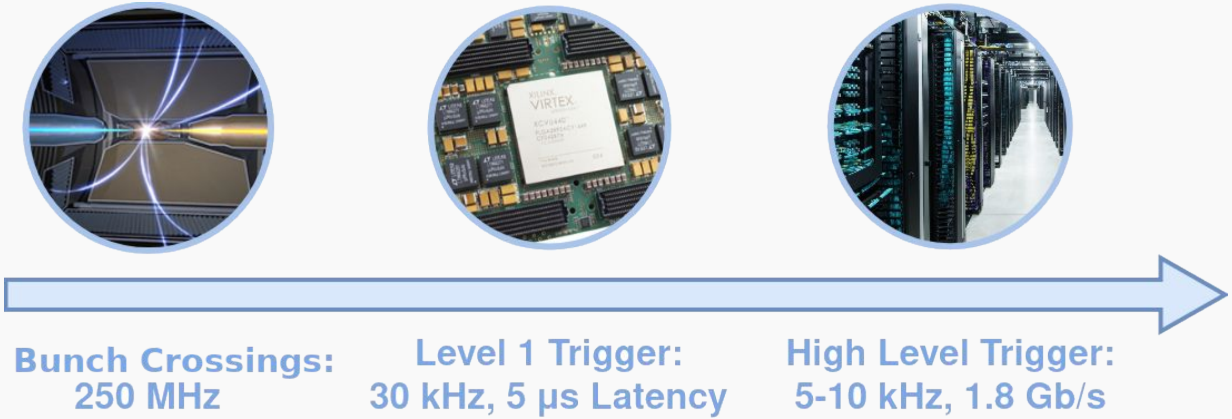
- Use Dynamic Graph Convolutional Network
- Update node features using edge information
    - Momentum, relative distance between two particles, 13-category outputs, 6-PID variables as input
- Measured tag-side efficiency with real data: **20 % increase of efficiency** on data



Yo Sato, Thibaud Humair, Petros Stavroulakis: Graph-Neural-Network based Flavor Tagger

Real Time Application

- Different trigger levels have to reduce data stream to match DAQ limitations
- Level 1 trigger: hardware-based on FPGAs
- High Level Trigger: full reconstruction CPU based
- If data is not triggered here, its not saved → not available



→ Bigger is not always better: real time machine learning applications require smart, small networks to be deployed on FPGAs with high throughput rates (e.g. 30 MHz)

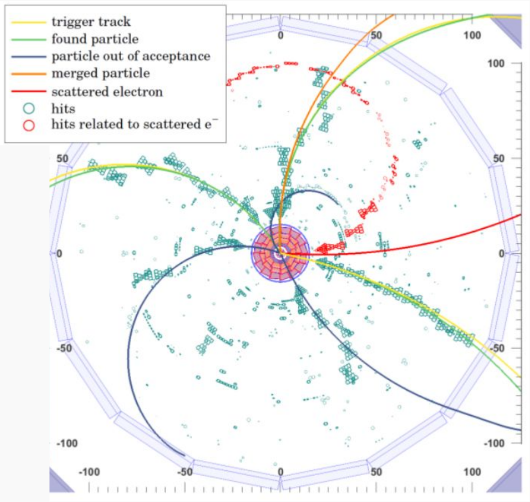
Neuro Z-Trigger

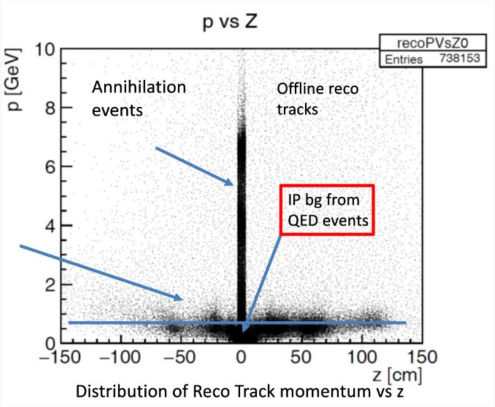
CDC Trigger Pipeline

- Track Segment Finder — Pattern Map
- 2D Track Finder — Hough Transformation
- **3D Track Info** — Neural Network

Trigger

27 inputs with
81 hidden nodes
and 2 output nodes





- Input of hough transformation used to predict **z-displacement** in respect to Interaction Point and **theta** of track
- Trained on **data** (reconstructed tracks)
- Used to reduce background from QED events (trigger rate too high from only track trigger)

Christian Kiesling, Felix Meggendorfer, Elia Schmidt, Marc Neu, Kai Unger, Alois Knoll, Simon Hiesl, Timo Forsthofer et al

Do you have questions?

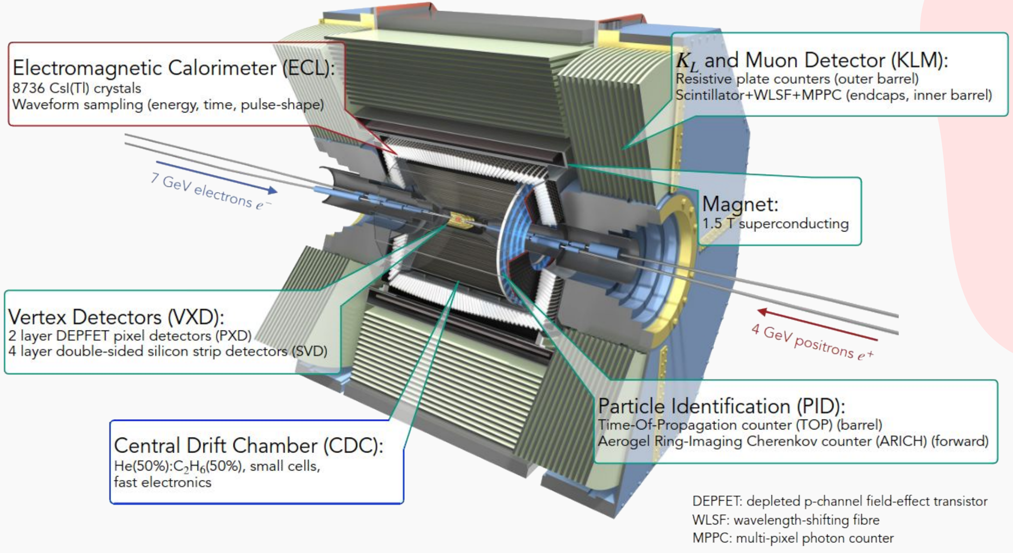
03

Object Condensation for Reconstruction

Motivation

- Searches for new physics for data we did not look at yet
- New tools with Machine Learning to improve trigger level reconstruction

➔ Improvement of current trigger algorithms

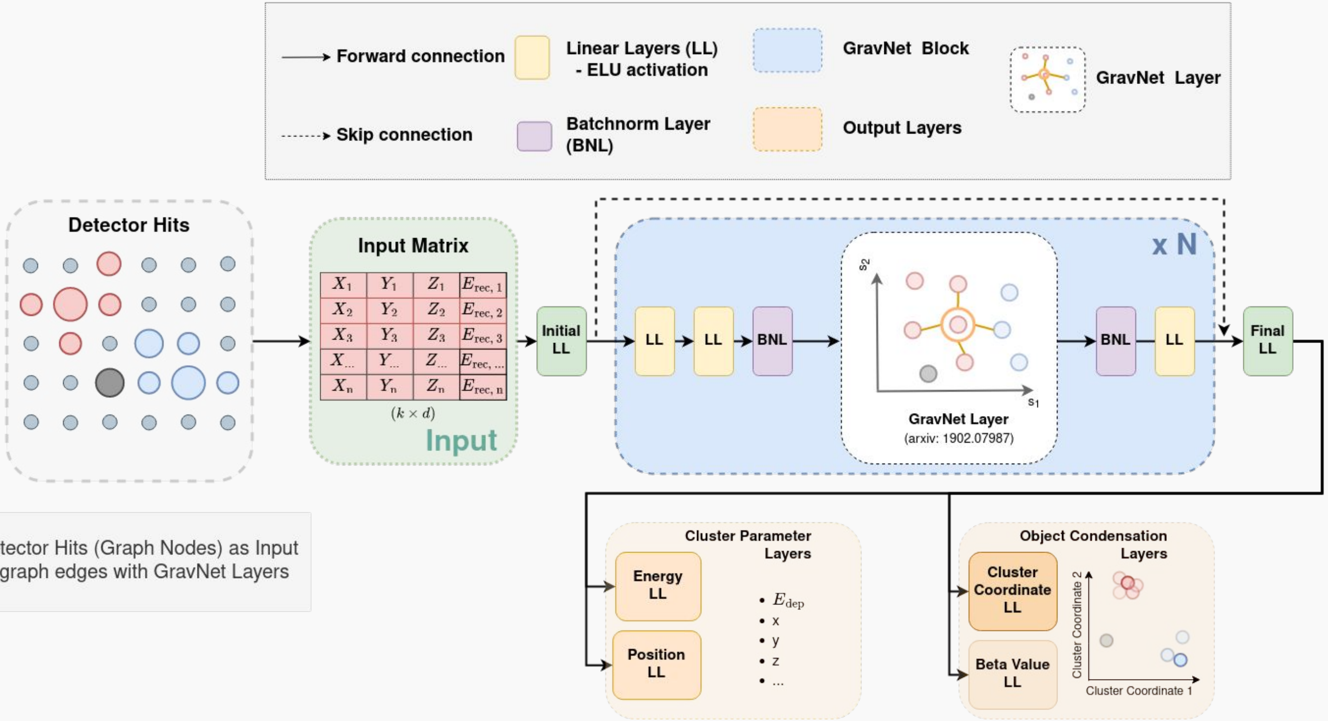


Requirements:

- Varying input sizes due to different number and type of particles in event
  - Use Graph Neural Networks
- We don't know how many clusters/tracks there will be
  - Object Condensation Algorithm[1]

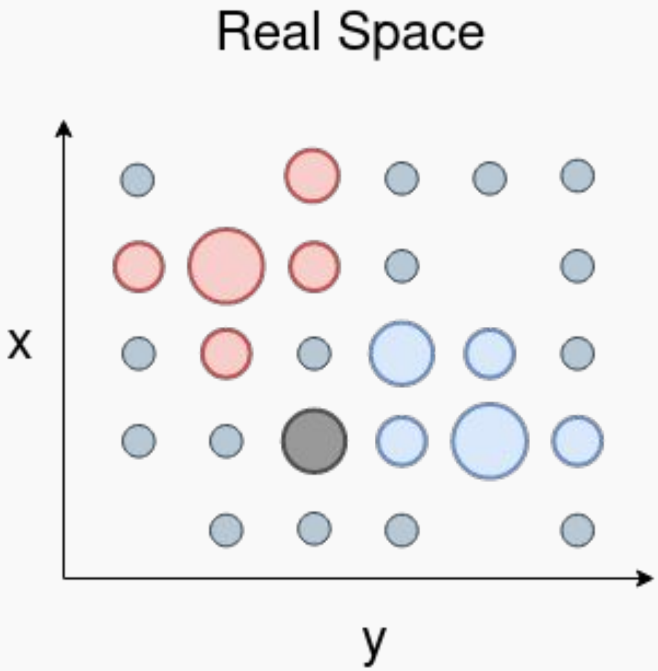
1: Jan Kieseler, arXiv:2002.03605

Graph Reconstruction Model



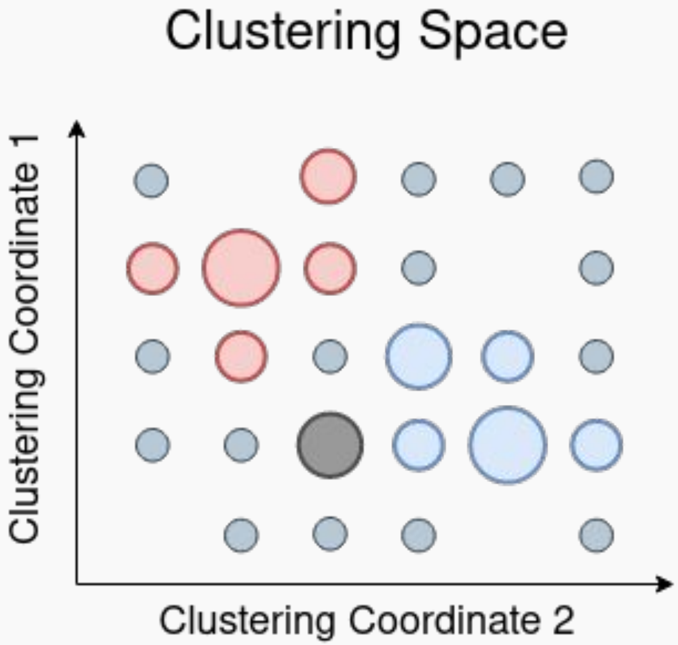
Use Detector Hits (Graph Nodes) as Input
Learn graph edges with GravNet Layers

Object Condensation



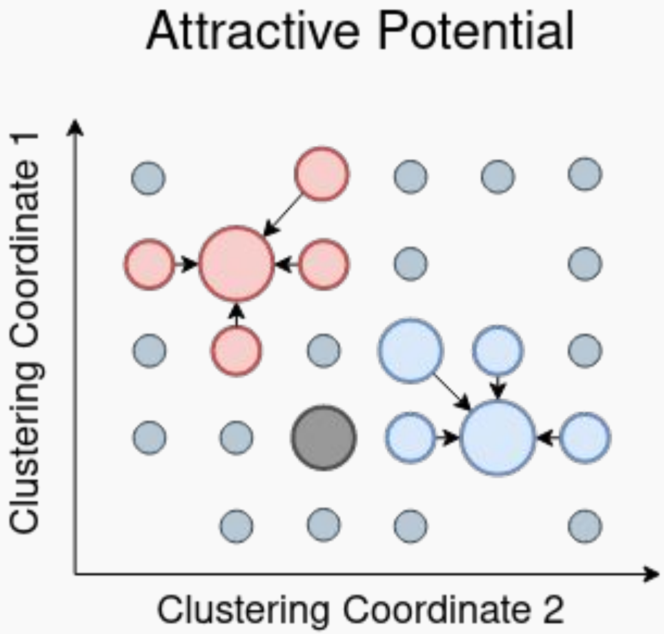
- OC algorithm takes input vertices (= detector hits) and translates them from real space to a clustering space
- Clustering space is learned by the network
  - In the beginning clustering space = real space

Object Condensation



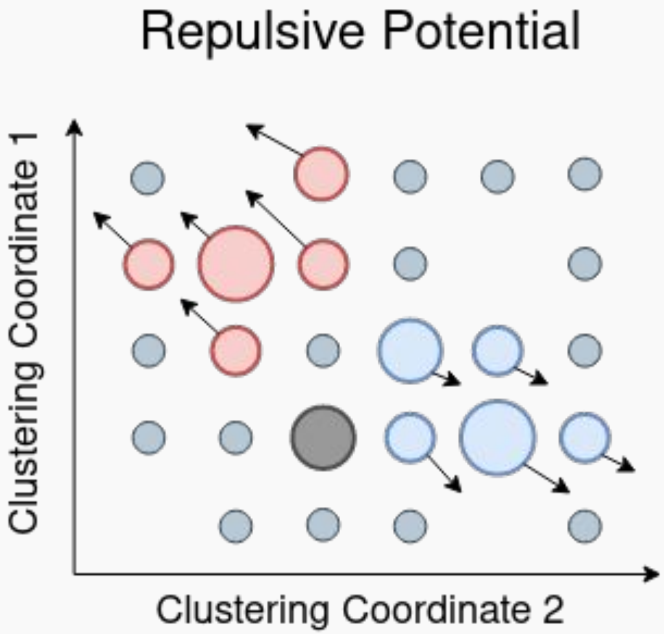
- OC algorithm takes input vertices (= detector hits) and translates them from real space to a clustering space
- Clustering space is learned by the network
  - In the beginning clustering space = real space
- Dimensionality of clustering space is hyperparameter
- OC algorithm introduces potential that clusters vertices from the same object together

Object Condensation



- An attractive potential draws vertices from the same object towards each other
- Background vertices are not influenced by the potential
- Attraction loss favors minimum distance between vertices of the same object

Object Condensation

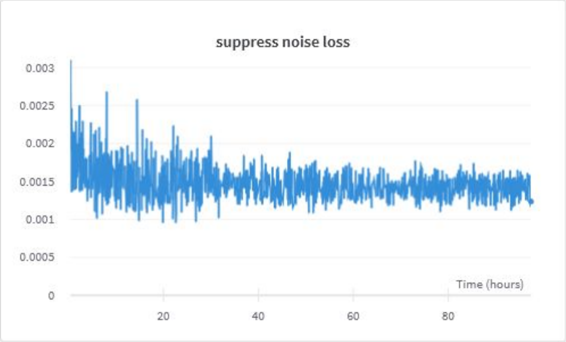


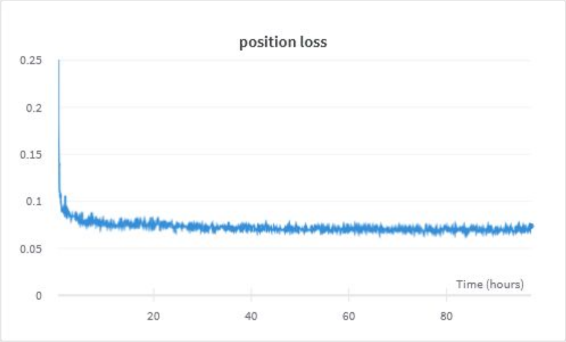
- An attractive potential draws vertices from the same object towards each other
- Background vertices are not influenced by the potential
- Attraction loss favors minimum distance between vertices of the same object
- A repulsive potential draws vertices from different objects away from each other
- Repulsion loss favors maximum distance between vertices of different objects
  - Clustering space is bounded

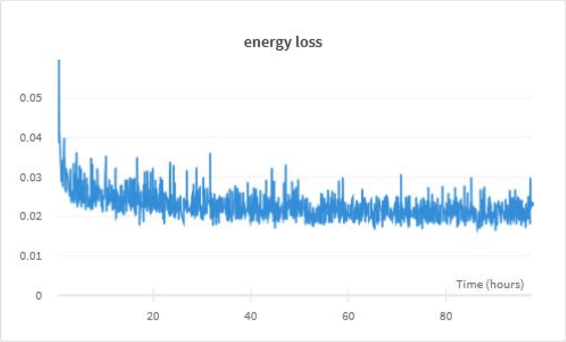
Object Condensation

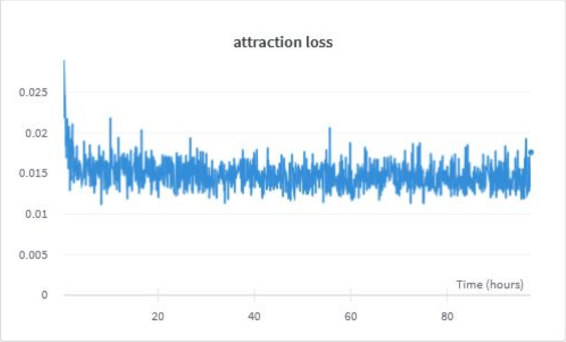
- Each vertex is assigned a β value
- Vertex with the highest β value is called a condensation point
  - Invokes the potential for vertices belonging to the same object
  - β loss favors one condensation point per object
- Condensation points “carry” values for their objects
  - ECL: energy, position, … of clusters
  - CDC: momentum, charge, displacement, … of tracks

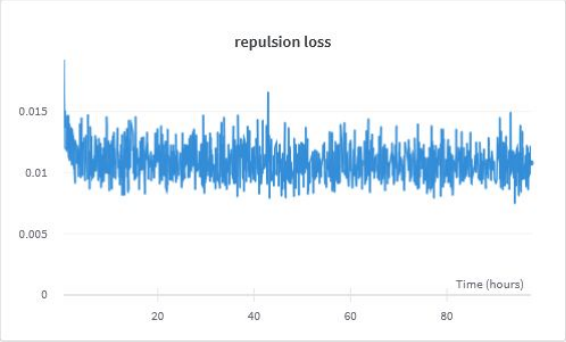
Object Condensation

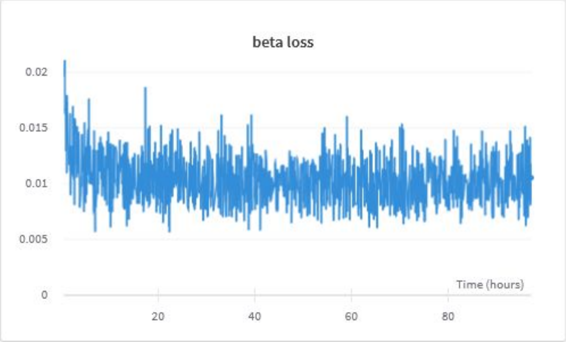




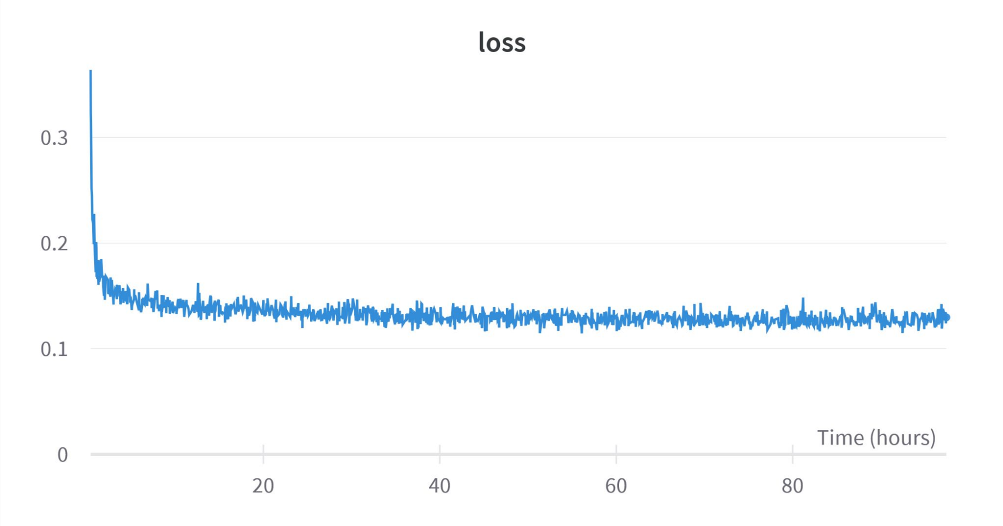








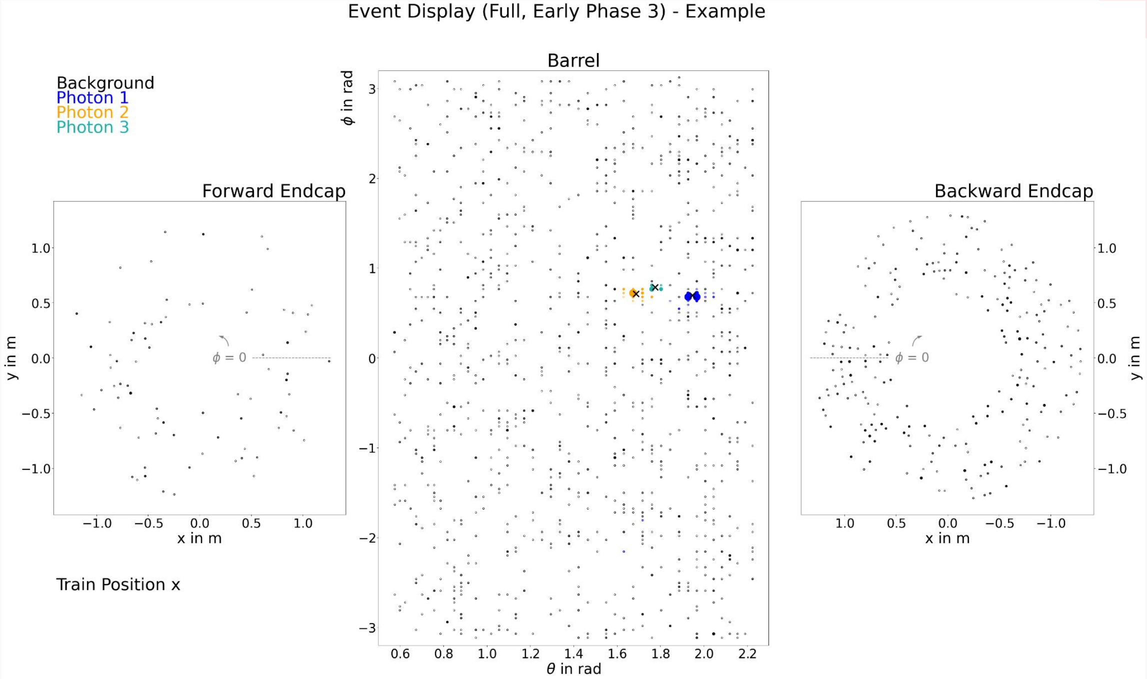
Object Condensation



- Full loss is sum of all sub-losses
- Scaling of losses can be done to improve training

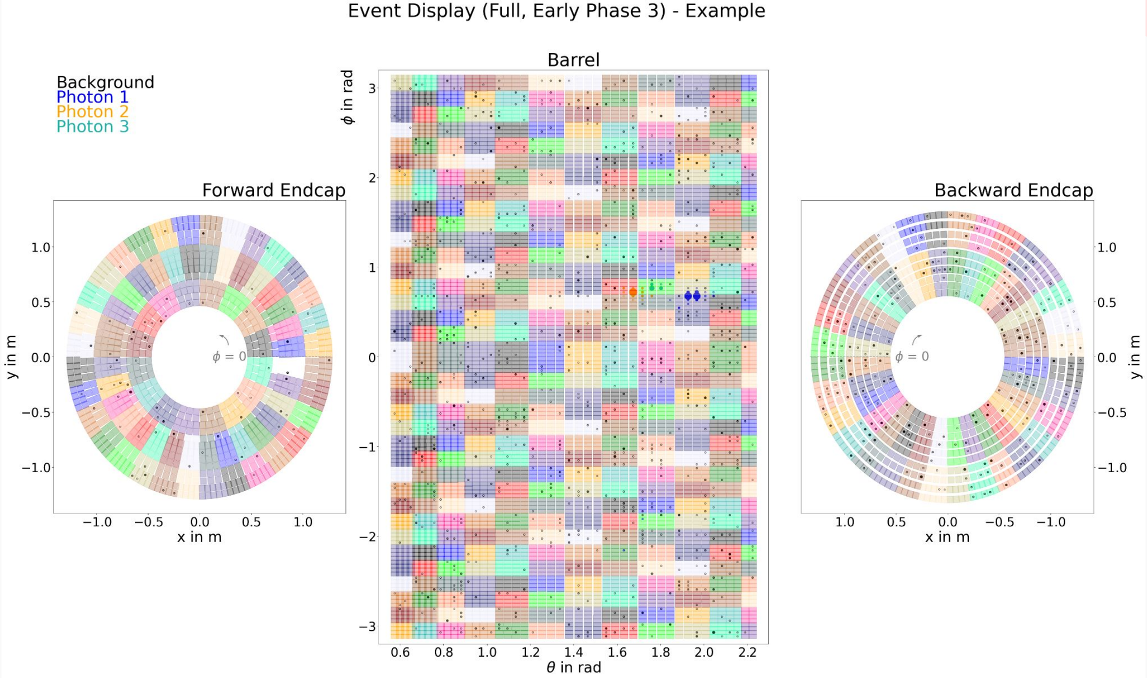
Object Condensation for ECL

- OC for ECL for both offline and online reconstruction
- For each cluster, predict existence, position and deposited energy
- Offline: Use every crystal with E > 1 MeV as input



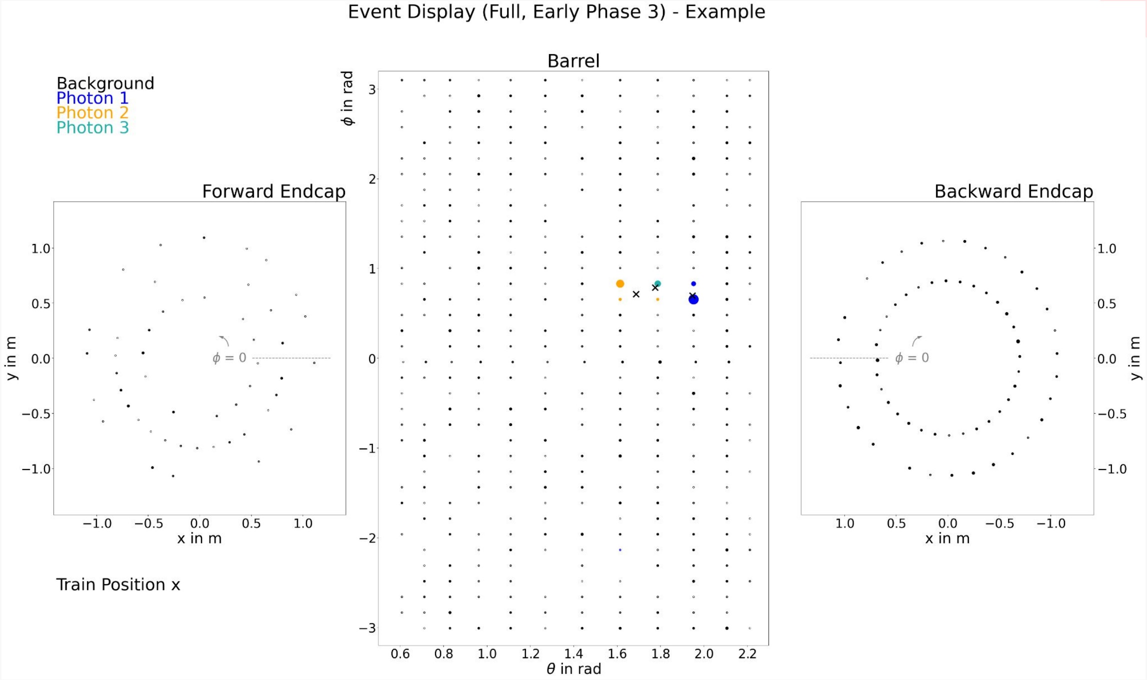
Object Condensation for ECL

- OC for ECL for both offline and online reconstruction
- For each cluster, predict existence, position and deposited energy
- Offline: Use every crystal with E > 1 MeV as input
- Online: Use current trigger mapping (4x4 crystals = triggercell) with energy cut as input



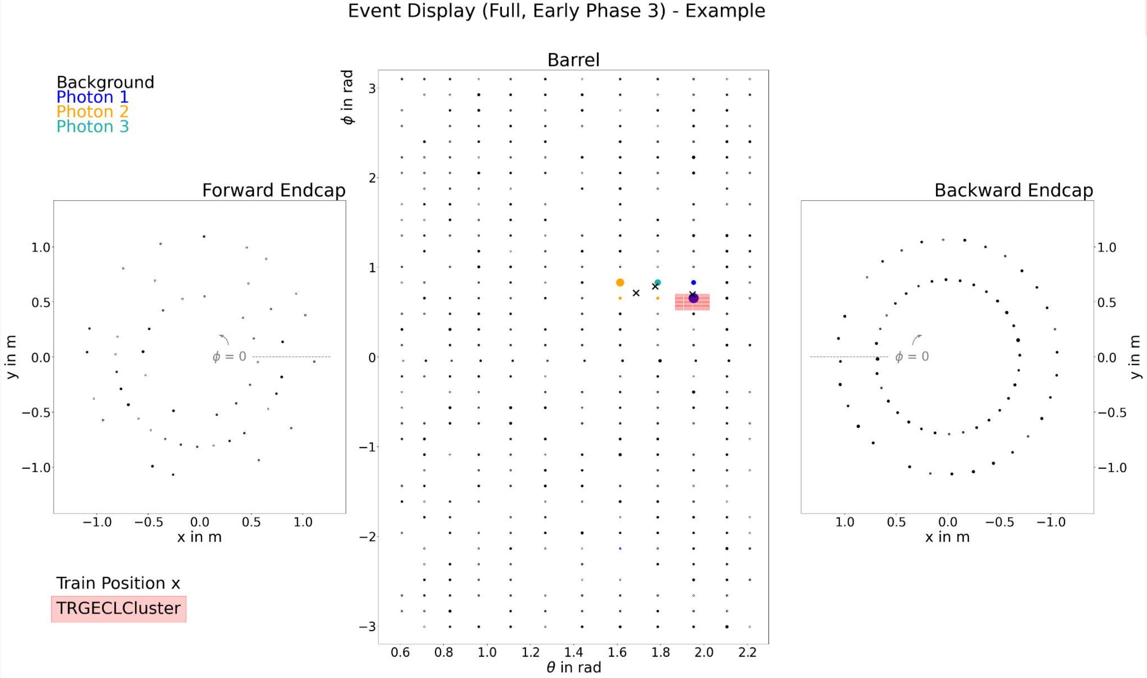
Object Condensation for ECL

- OC for ECL for both offline and online reconstruction
- For each cluster, predict existence, position and deposited energy
- Offline: Use every crystal with E > 1 MeV as input
- Online: Use current trigger mapping (4x4 crystals = triggercell) with energy cut as input



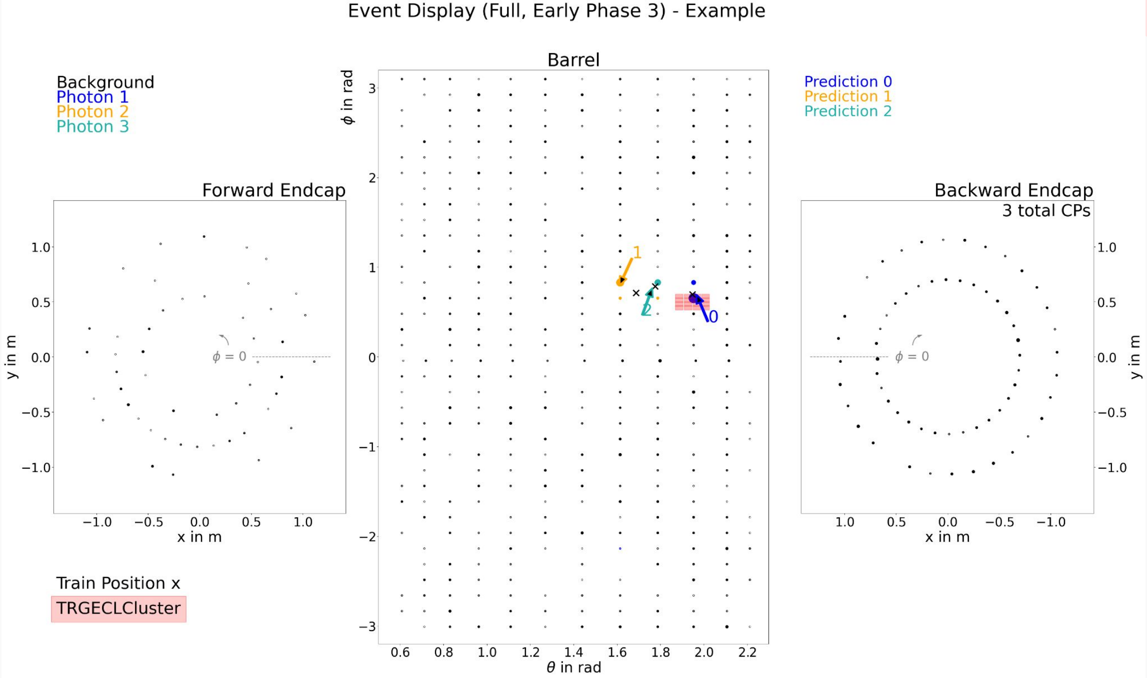
Object Condensation for ECL

- OC for ECL for both offline and online reconstruction
- For each cluster, predict existence, position and deposited energy
- Offline: Use every crystal with E > 1 MeV as input
- Online: Use current trigger mapping (4x4 crystals = triggercell) with energy cut as input
- Improve efficiency and resolution of ECL Trigger algorithm
  - Current ECL Trigger has low efficiency for overlapping clusters

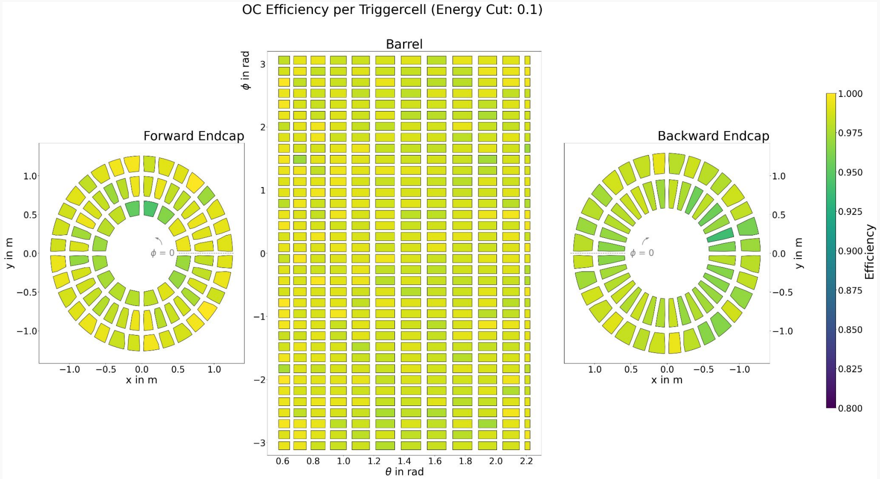


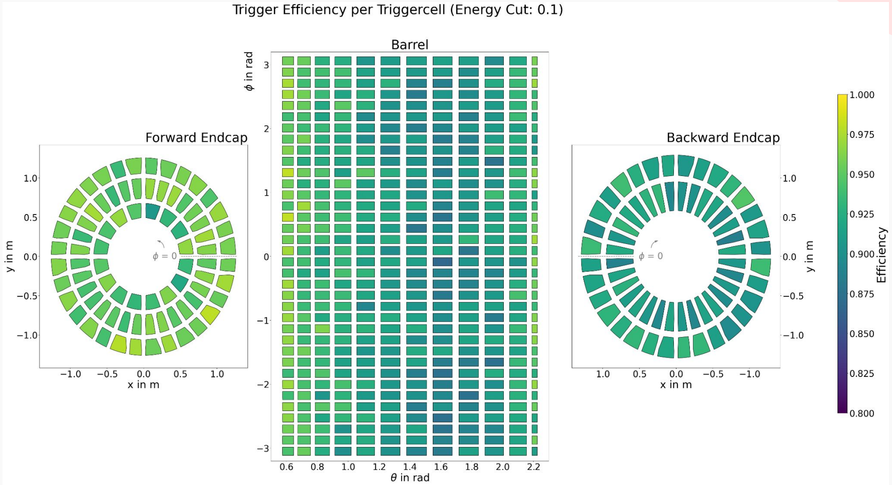
Object Condensation for ECL

- OC for ECL for both offline and online reconstruction
- For each cluster, predict existence, position and deposited energy
- Offline: Use every crystal with E > 1 MeV as input
- Online: Use current trigger mapping (4x4 crystals = triggercell) with energy cut as input
- Improve efficiency and resolution of ECL Trigger algorithm
    - Current ECL Trigger has low efficiency for overlapping clusters



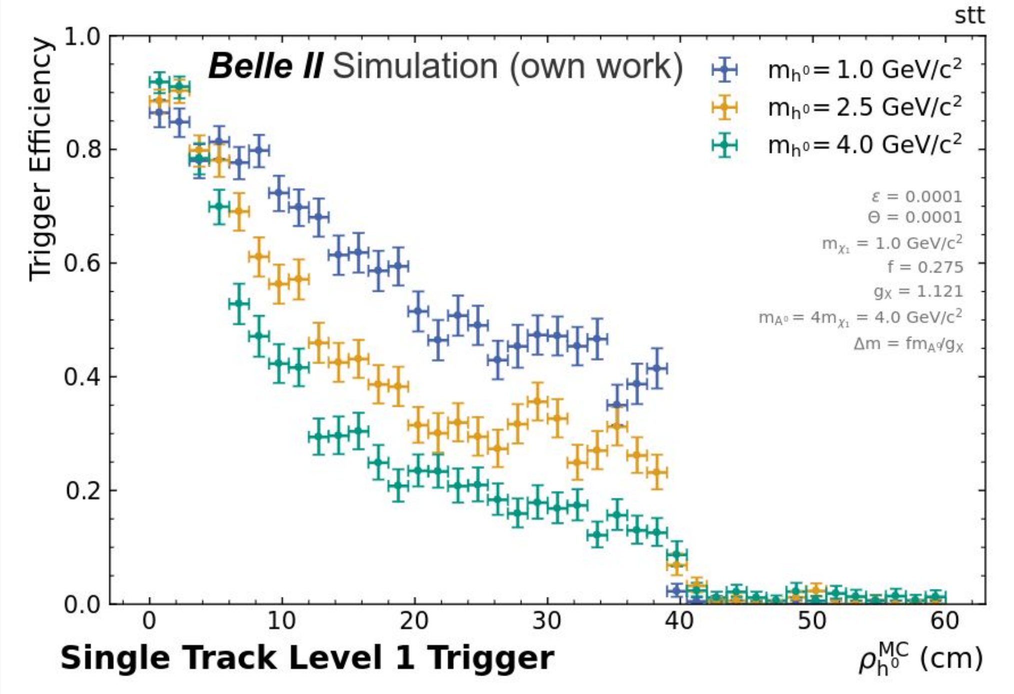
Object Condensation for ECL



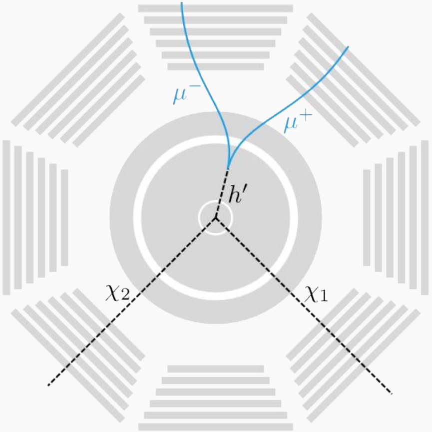


- OC algorithm trained on 1-6 photons with generated energy between 0.1 GeV and 2 GeV
- Energy of triggercells > 100 MeV (modelling of current trigger algorithm)
- Improvement in efficiency for full ECL
- Current modelhas ~40000 parameters

Object Condensation for CDC

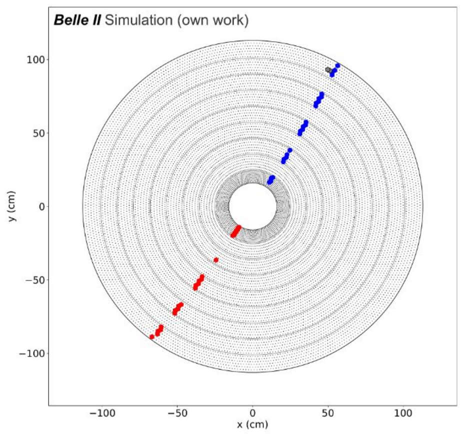


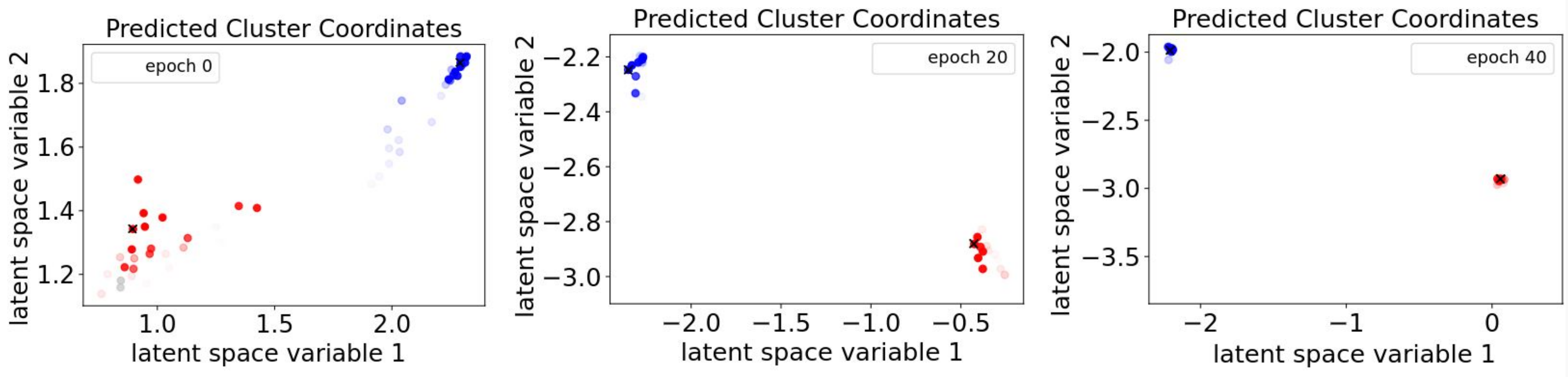
- Displaced vertices important signature in searches for new physics
- Improve both real time and offline reconstruction



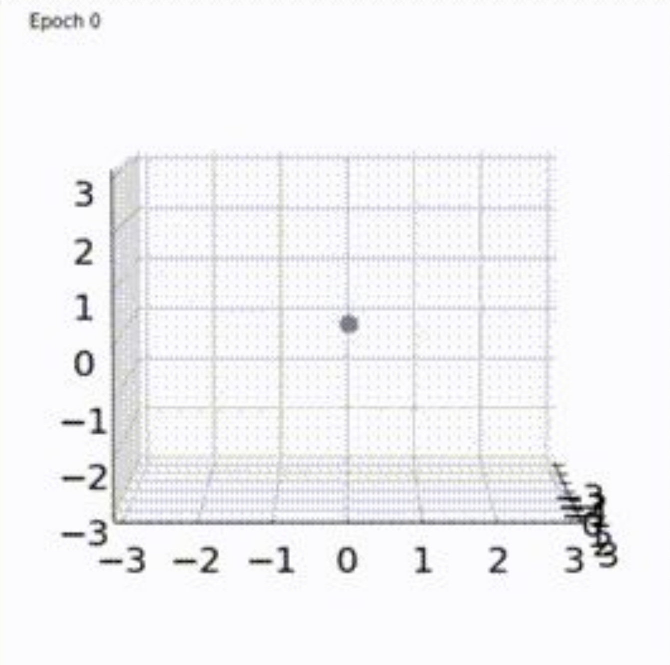
Object Condensation for CDC

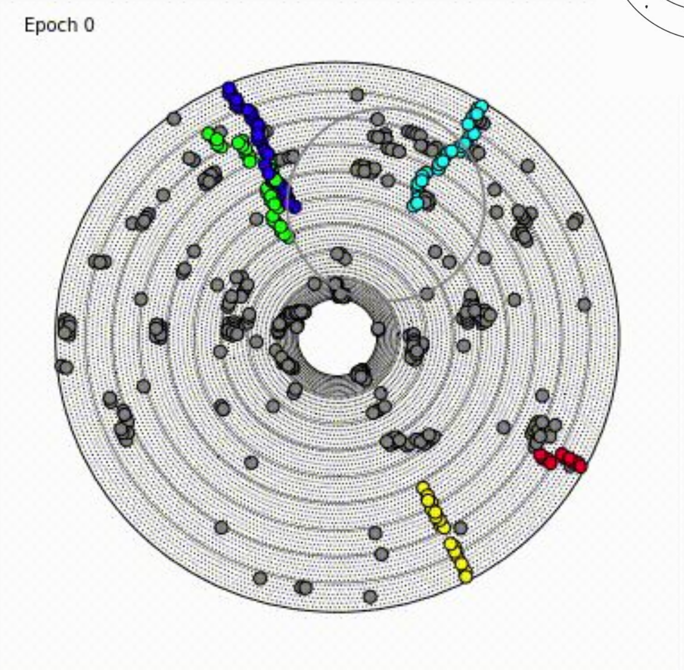
- Use CDC hits in the detector as input for the OC GNN
- Predict number of tracks
  - For each track, predict starting position, momentum and charge

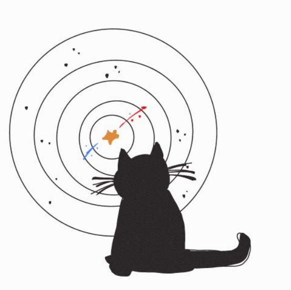




Object Condensation for CDC

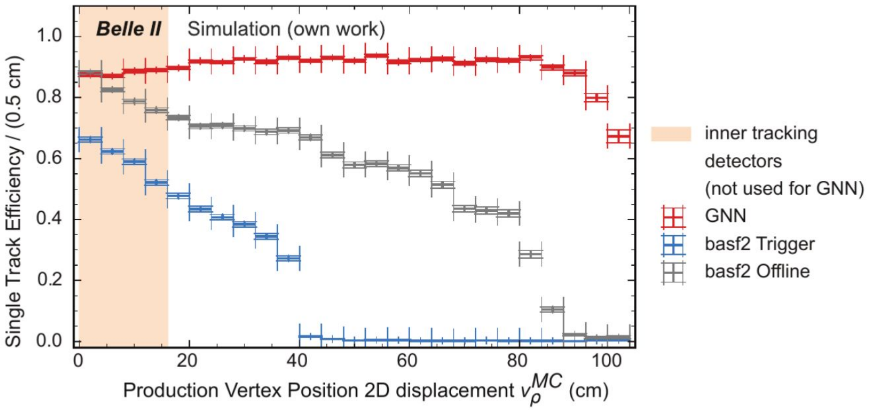


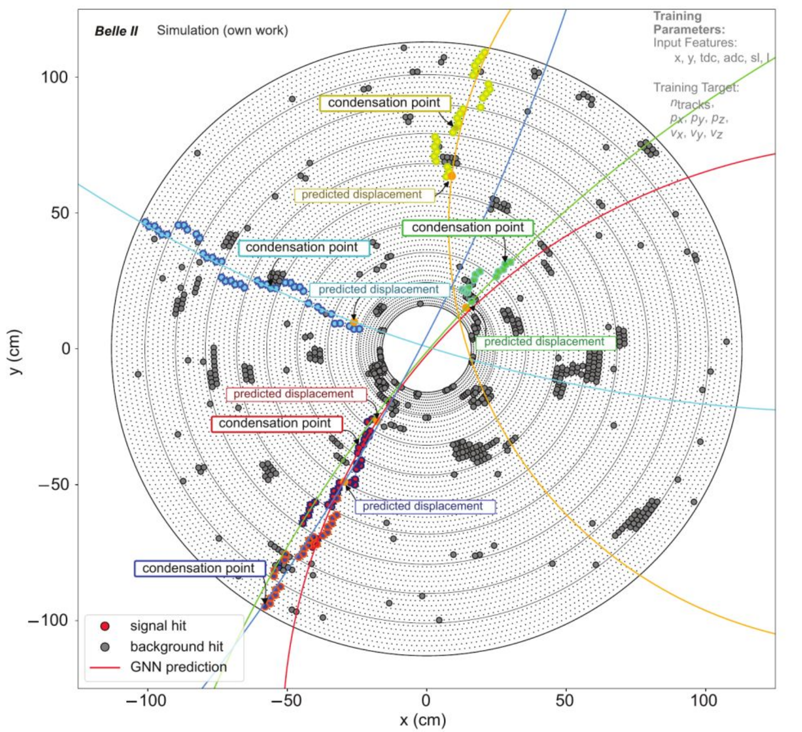




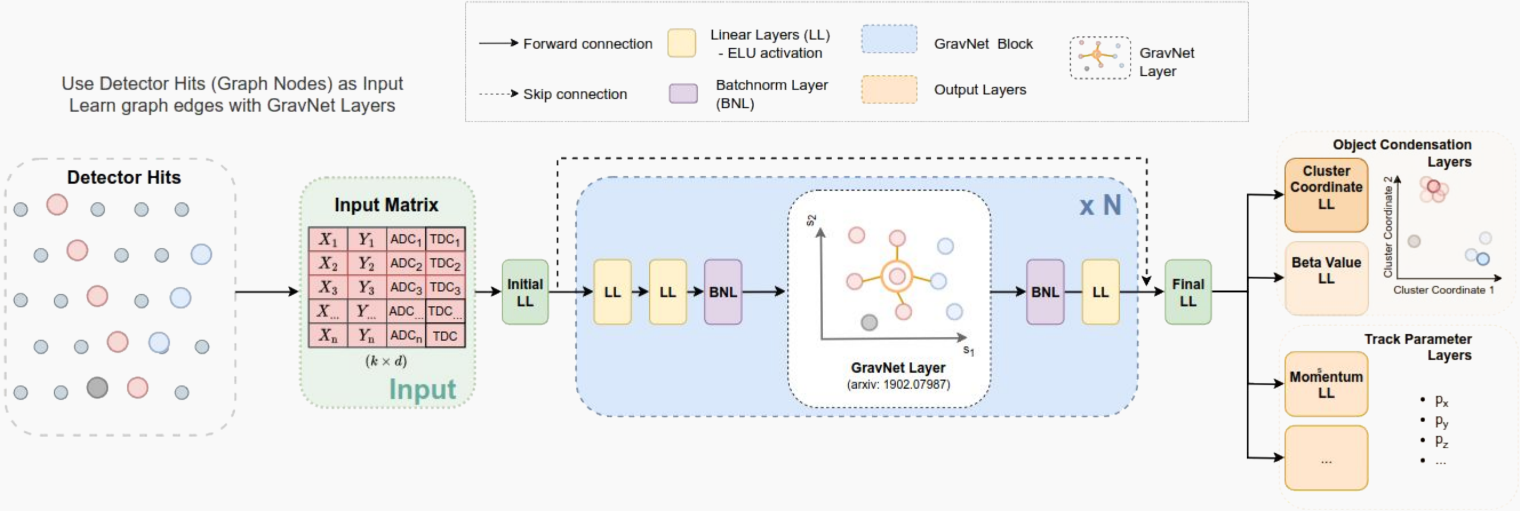
Object Condensation for CDC

GNN achieves 96.68% efficiency on displaced vertex samples $X \to \mu^- \mu^+$





Graph Reconstruction Model



Adjustable Parameters:

General Parameters:

- dimension **dim1**, **dim2** of Linear Layer
- number of Graph Blocks **nblocks**

GravNet Parameters:

- number of **k**-nearest neighbors in GravNet
- GravNet space dimensions (currently 4)

Output:

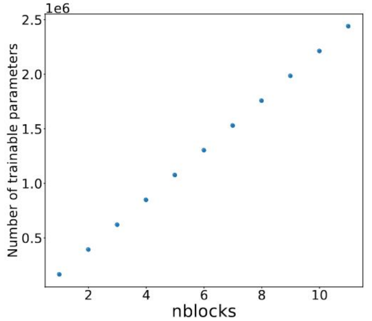
- Dimension of Cluster Coordinates for OC **coord**
- Number of output layers according to Track Parameter Predictions

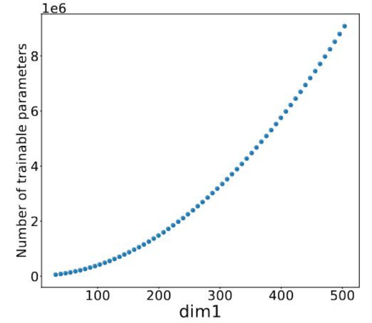
Lea Reuter (lea.reuter@kit.edu), Isabel Haide (isabel.haide@kit.edu)

Hyperparameter

This starting setup has ~669000 trainable parameters!

Model Parameters need to be greatly reduced to fit FPGA and gbasf2 CPU requirements





Hyperparameter	Current Value
Number of Neighbors **k**	9
Number of GravNet Blocks **nblocks**	3
Number of Nodes 1 **dim1**	128
Number of Nodes 2 **dim2**	32
Momentum	0.6
CC Space **ccoords**	3
Learning Rate **lr**	0.001
Optimizer **optim**	Adam

Training Documentation wandb.ai

- Machine Learning Platform for developers
- Cloud based ML experiment tracking tool
- Features:
  - Experiment Tracking
  - Hyperparameter Tuning
  - Data and Model Versioning
  - Model Management
  - Data Visualization
  - Collaborative Reports
  - Integration for **PyTorch**, Keras, PyTorch Lightning etc.
  - Private-Hosting

```
wandb.log({'full loss': loss,
           'repulsion loss': reploss,
           'attraction loss': attloss,
           'energy loss': energyloss,
           'beta loss': betaloss,
           'suppress noise loss': suppressloss,
           'pos loss': posloss})
```

Overview Reports **Projects** Likes

Projects — Create new project

Search — 1-12 of 12

Name	Last Run	Runs	Entity
validation_triggercells	2023-07-23	133	ihaide
validation_ep3	2023-06-13	3	ihaide
triggercells	2023-07-17	132	ihaide

Training Documentation [wandb.ai]

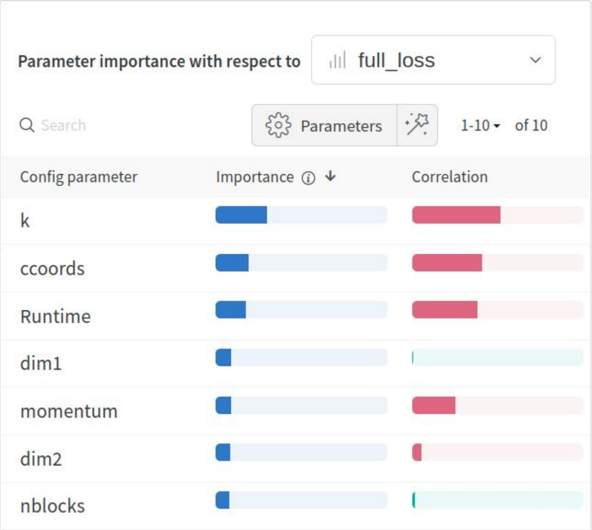
Very simple to use!
Everything at one place!

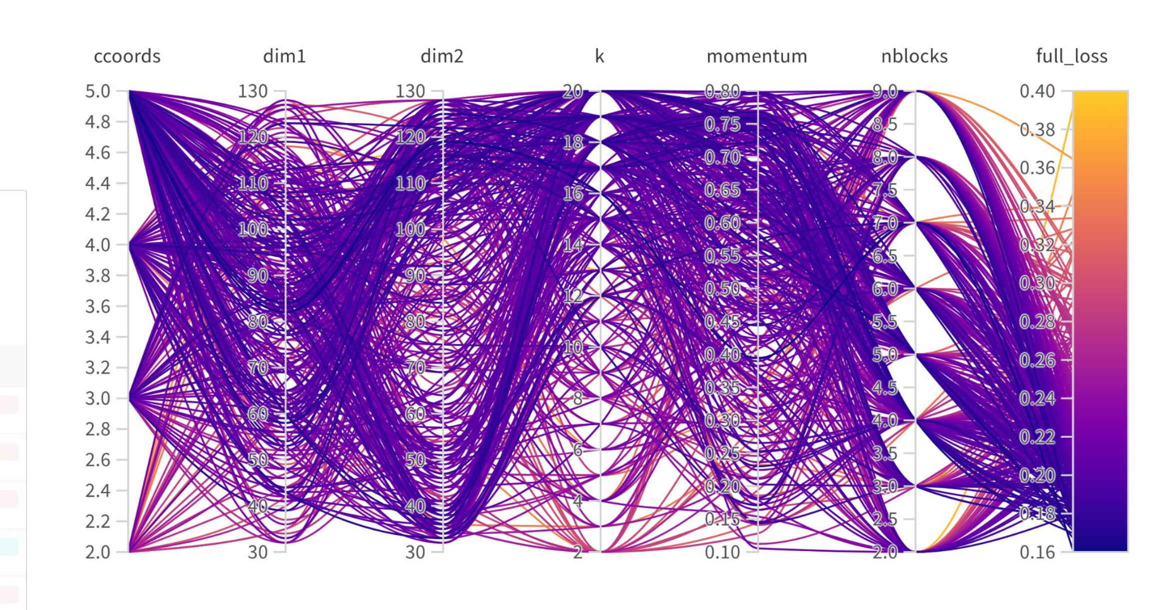
- Save configs and models to keep track of multiple trainings and version control
- Monitor training
- Log weights and biases for deep learning trainings (confirm that not only last layers get updated while training!)

Have to monitor more than 7 sub-losses for Object Condensation!

Hyperparameter Optimization

- Runs easily on one or more gpus (start a new sweep on another gpu with the same sweep agent)
- Optimizes model parameters

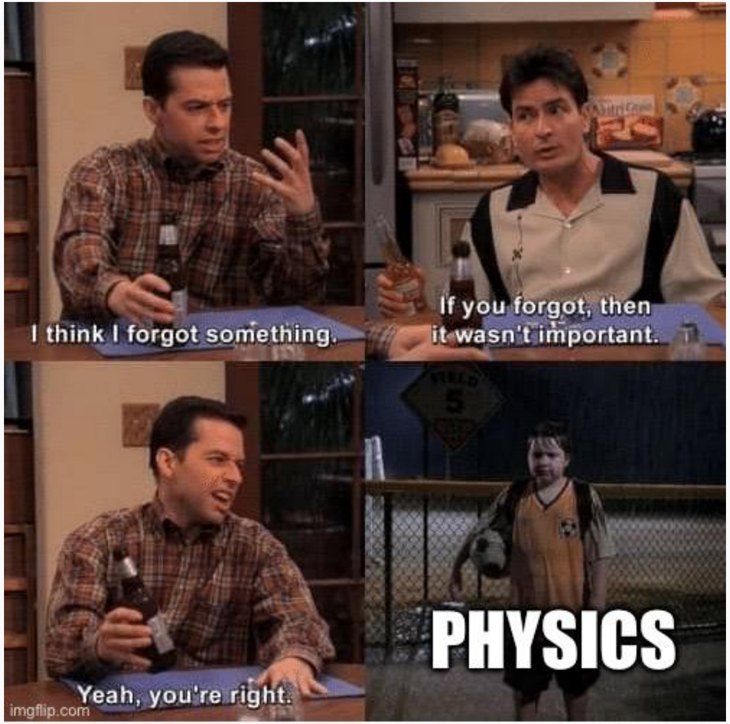




ECL OC model down to 40,000 parameters

Quality Control

- Machine learning very dependent on training samples
    - Know what is included in the simulated samples and make comparisons with data
    - Be careful about training sample composition (if training samples are 99% background its very efficient to just predict background)
        - Enrich dataset with rare cases
        - Introduce class weights
    - Check how your model will perform on other cases (empty events, background events..)
    - Check for biases in physics distributions for predictions
    - Use meaningful metrics for evaluation
        - Often loss/accuracy is not showing the whole picture



04

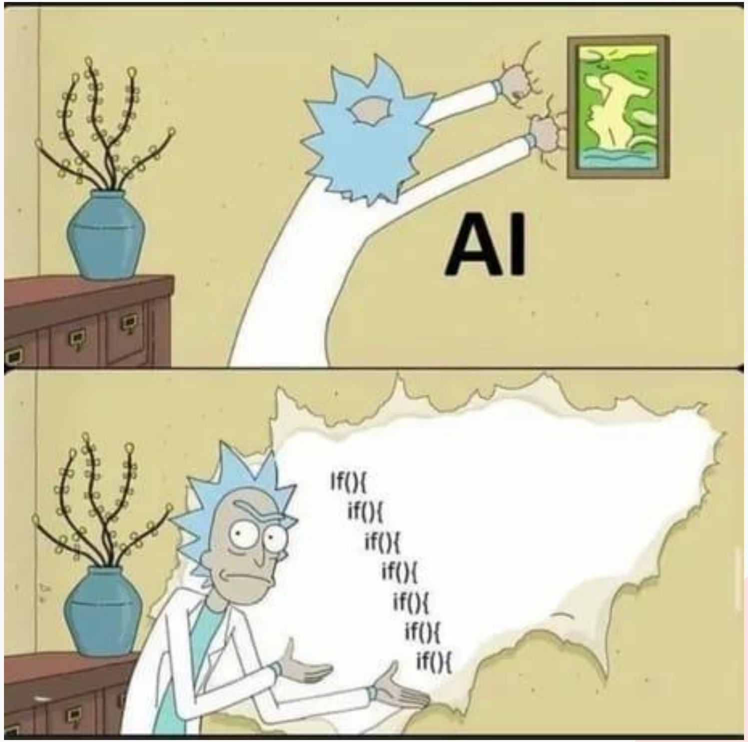
How can I Participate?

Find the right problem

- Lots of challenges in HEP can be tackled with ML algorithms
- Signs for possible improvement through ML:
  - High number of input variables
  - High correlations between variables
  - Unknown or difficult physical model

But:

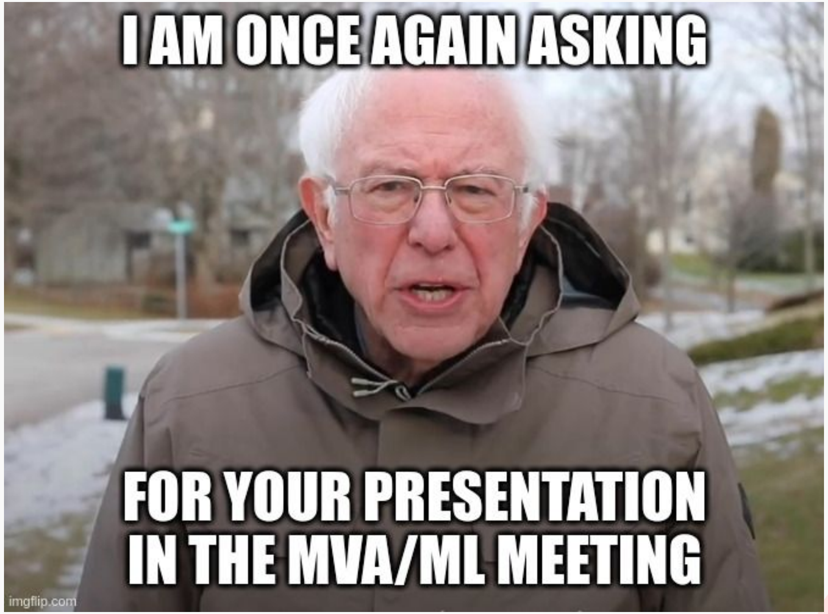
- Not everything can be improved with ML
- Analytical functions, if known, will often perform better and with less computing resources



Join our MVA/ML Meeting!

We want YOU!

Indico: https://indico.belle2.org/category/167/
Meetings every 2 weeks, American friendly time every 4 weeks!



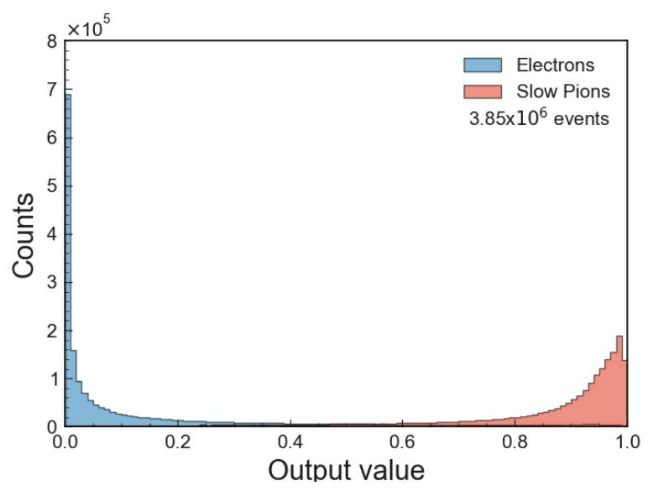
Backup

Slow Pion Rescue

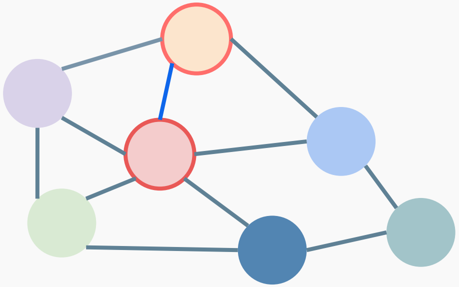
- Around 26% of slow pions are not reconstructed by SVD, need to reconstruct slow pions from the PXD
- PXD data rate may have to be reduced after shutdown
  - PXD cluster not associated to a track are dropped by region-of-interest algorithm
  - Loose PXD cluster
- PXD hits are dominated by QED (2000 electrons per 1 slow pion)

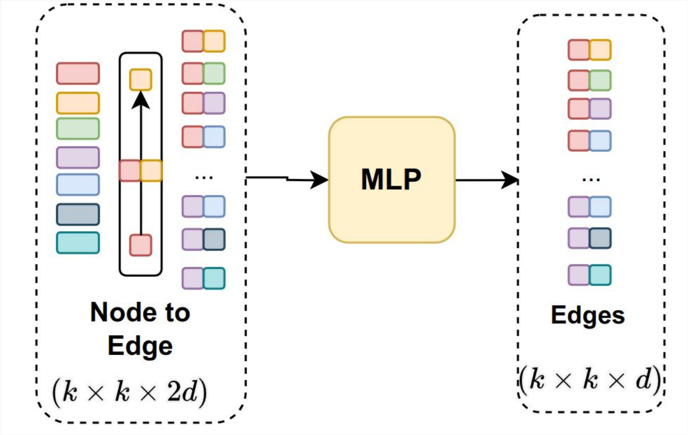
- **Goal**: Discriminate QED background from slow-pion using a neural network
  - 1 hidden layer with 100 nodes
  - PX cluster properties as input: size, shape position, …

➔ Tested various network models, achieved 89% efficiency for 90% electron rejection on simulated samples

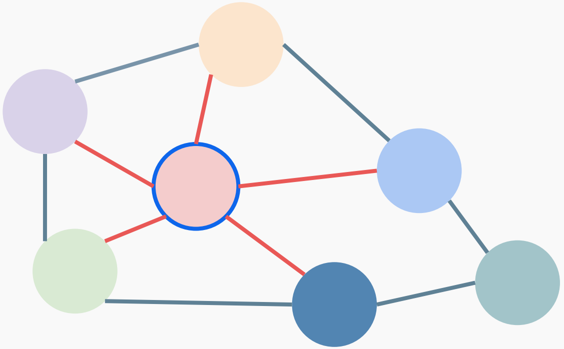


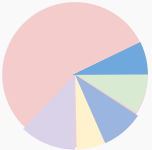
Message Passing: Update Edges



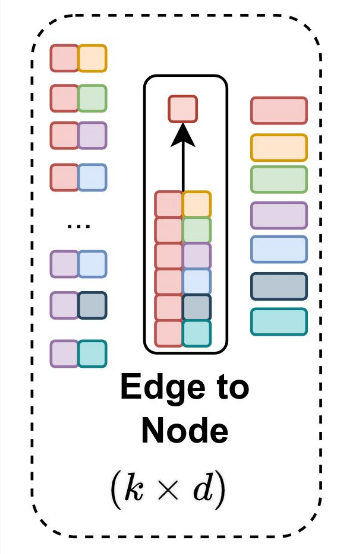


Message Passing: Update Nodes





Red node updated with information of neighbouring nodes and edges

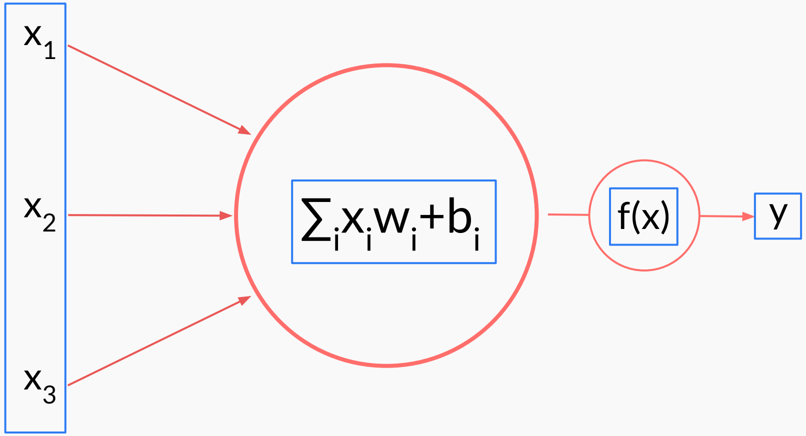


Classic Neural Networks

~~Boosted Decision Trees~~

Inputs:
- Data often encoded in floats
- Best normalized to [0, 1]
- Fixed Ordering

Weights and Bias:
- Learnable Parameters
- Often adapted through backpropagation



Activation Function:
- Nonlinear function, such as tanh(x) or max(0, x)
- Necessary to compute nontrivial problems

Output:
- Classification or regression
- Best scaled to [0, 1]
- Input to loss function, evaluation of the network's performance

x_i: Inputs
w_i: Learnable Weights
b_i: Learnable Bias
f(x): Activation function
y: Output