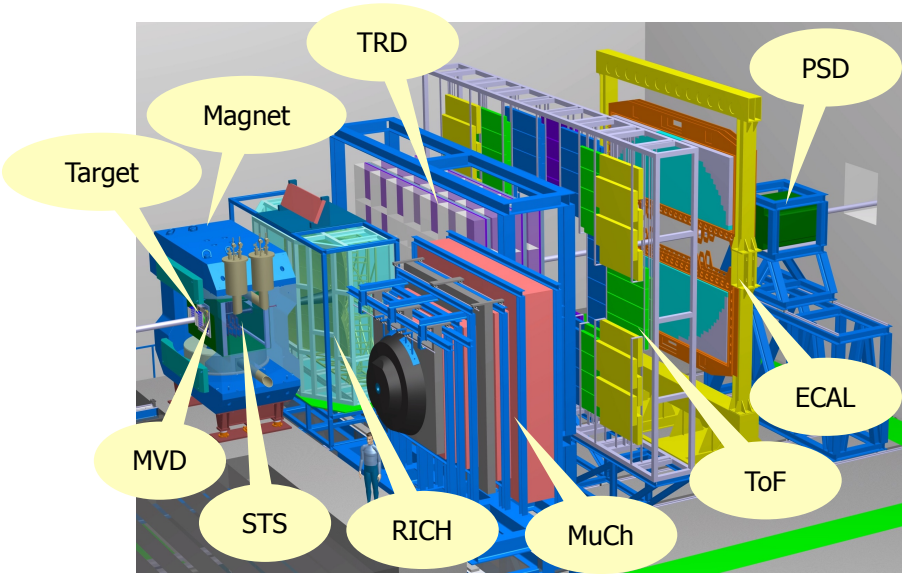
ANN4FLES:
A Neural Network Package for Applications in the CBM Experiment at FAIR

A. Belousov[1], A. Bender[1], Q. Du[1], D. Griesemer[1], I. Kisel[1,2,3,4], R. Lakos[1,2], A. Mithran[1,2], O. Tyagi[1,2], I. Vassiliev[3], T. Zhang[1], G. Zischka[1], A. Correia[5], N. Garroum[5], V. Gligorov[5]

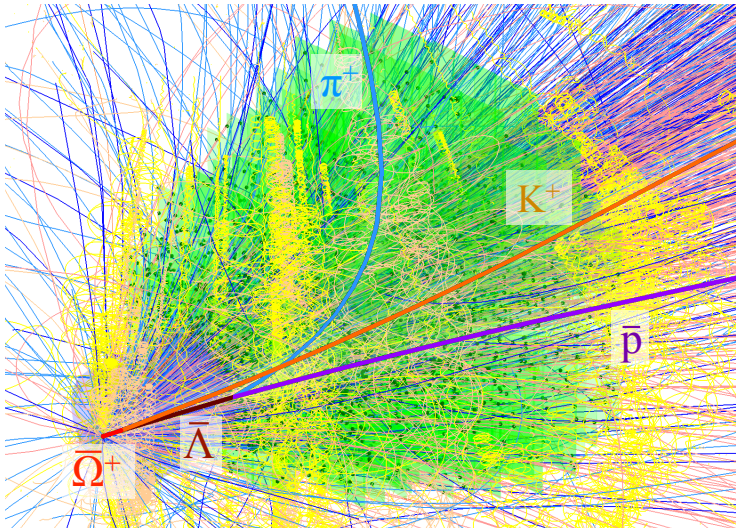
[1]Goethe-University Frankfurt, Frankfurt am Main, Germany
[2]Frankfurt Institute for Advanced Studies, Frankfurt am Main, Germany
[3]Helmholtz Research Academy Hesse, Frankfurt am Main, Germany
[4]Helmholtz Center for Heavy Ion Research, Darmstadt, Germany
[5]Laboratoire de Physique Nucleaire et de Hautes Energies, Paris, France

Challenges in CBM

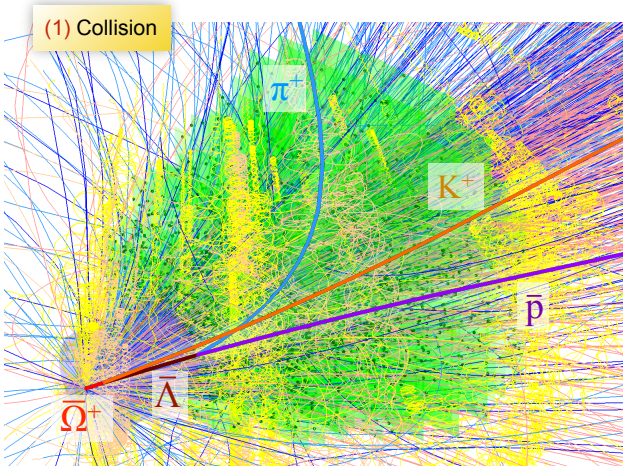


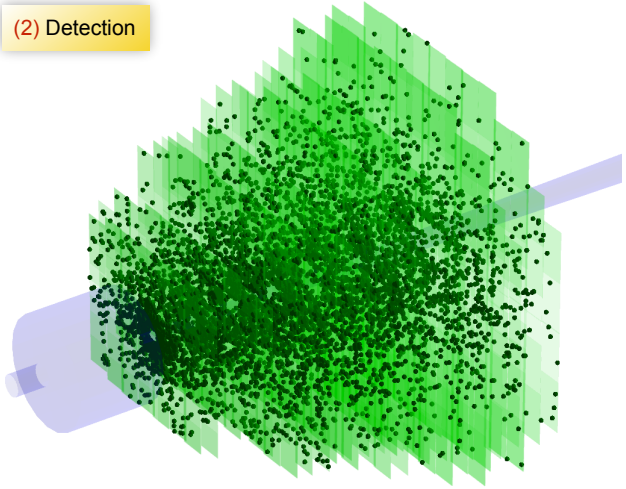
- CBM - future fixed-target heavy-ion experiment at FAIR, Darmstadt, Germany
- 10^5 - 10^7 collisions per second
- Up to 1000 charged particles/collision
- Free streaming data
- No hardware triggers
- On-line time-based event reconstruction and selection is required in the first trigger level

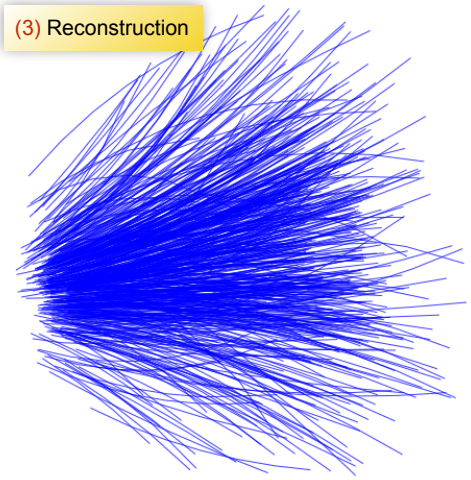
- On-line reconstruction at the on-line farm with 60000 CPU equivalent cores
- High speed and efficiency of the reconstruction algorithms are required
- The algorithms have to be highly parallelised and scalable
- CBM event reconstruction: Cellular Automaton and Kalman Filter



First Level Event Selection (FLES)



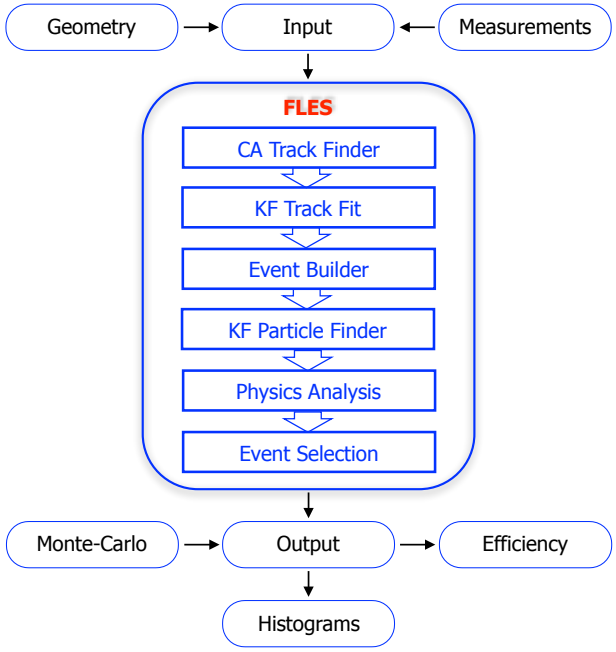




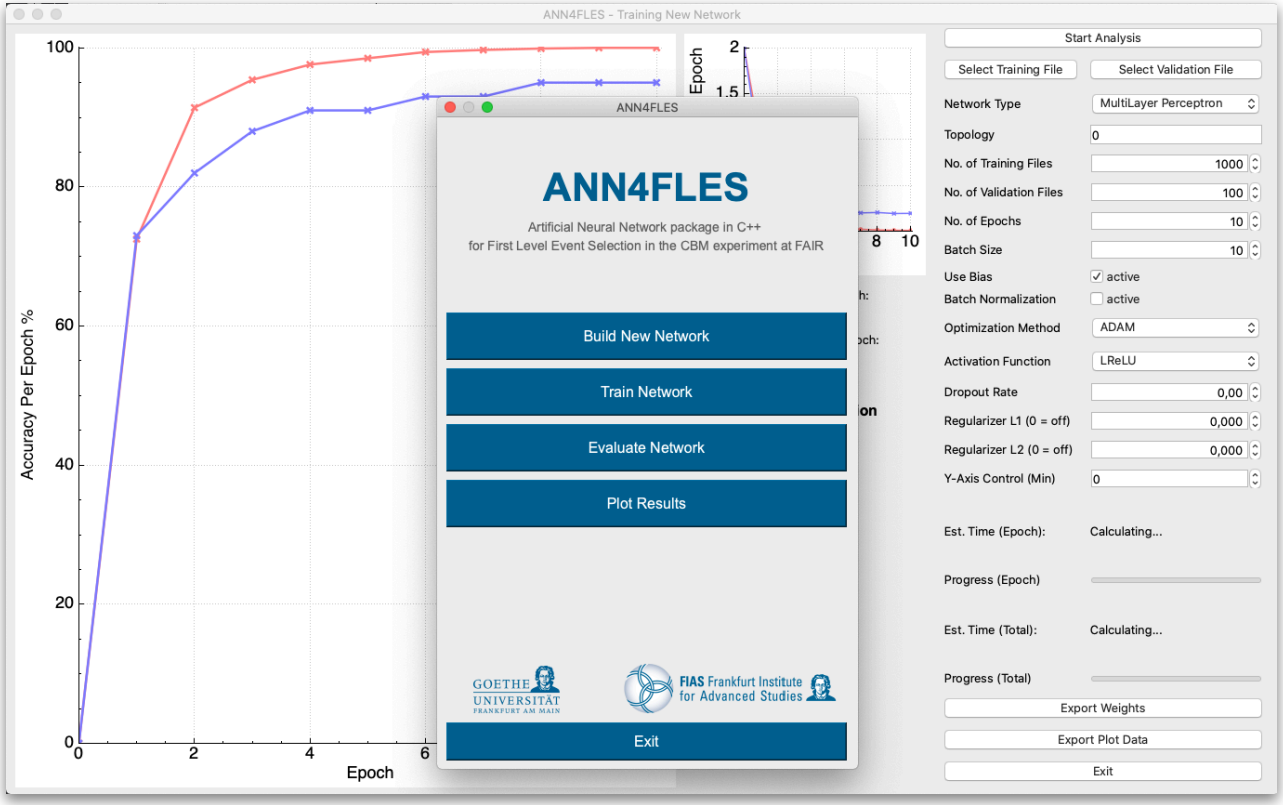
The full event reconstruction will be done on-line at the First-Level Event Selection (FLES) and off-line using the same FLES reconstruction package.

- Cellular Automaton (CA) Track Finder
- Kalman Filter (KF) Track Fitter
- KF short-lived Particle Finder

All reconstruction algorithms are vectorized and parallelized.



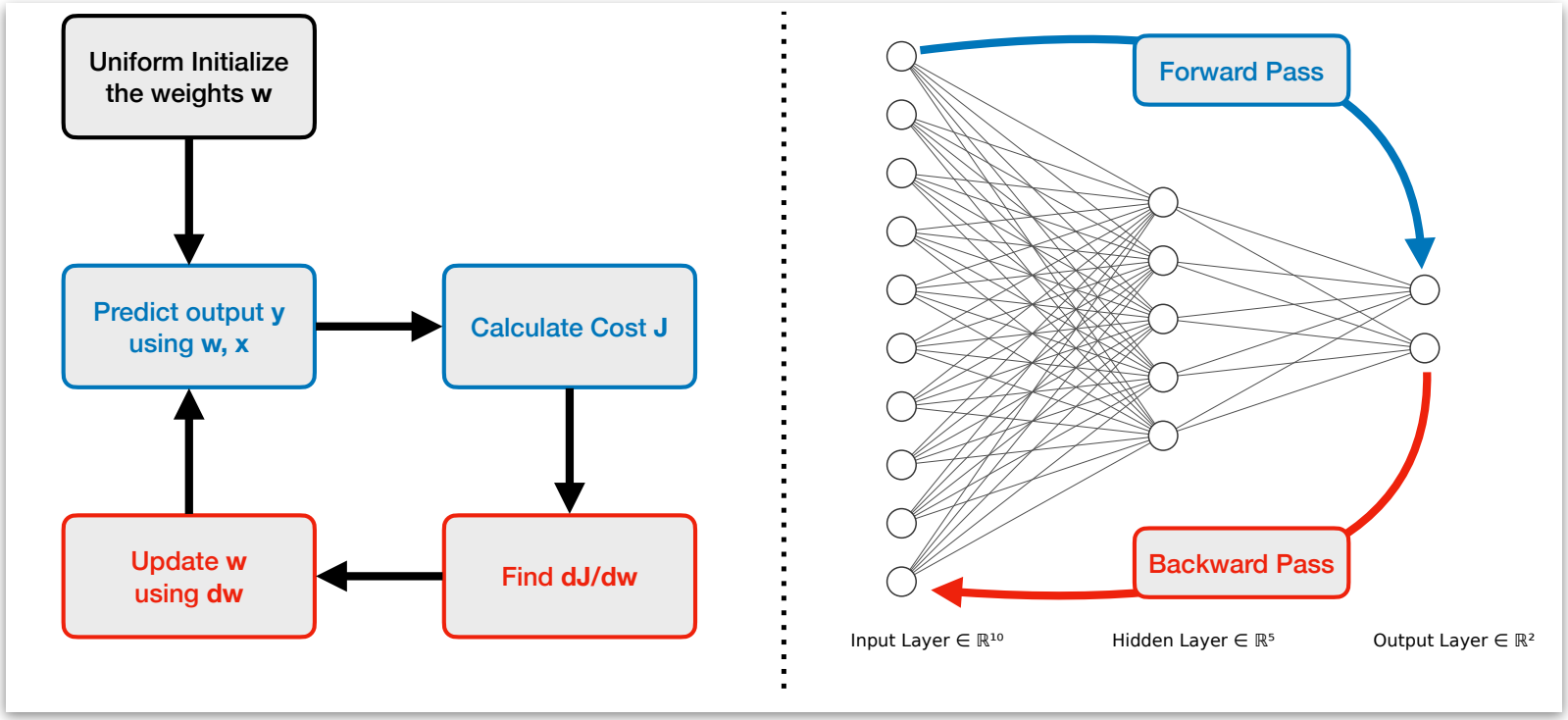
ANN4FLES : Artificial Neural Networks for First Level Event Selection



- **ANN4FLES** is a fast **C++ package** designed for Artificial Neural Networks (ANN) in CBM
- It provides a variety of network architectures with **minimal additional programming**
- The package includes a Graphical User Interface (**GUI**) for **network selection** and **hyperparameter** adjustment
- **Implemented networks** in **ANN4FLES** include Multilayer Perceptron (**MLP**), Convolutional Neural Network (**CNN**), Recurrent Neural Networks (**RNN**), Graph Neural Networks (**GNN**), and Bayesian Neural Network (BNN).
- Extensive **testing** on datasets like **MNIST**, **CIFAR**, **Cora**, etc., has been **performed** and **compared** with **PyTorch**

Multilayer Perceptron (MLP)

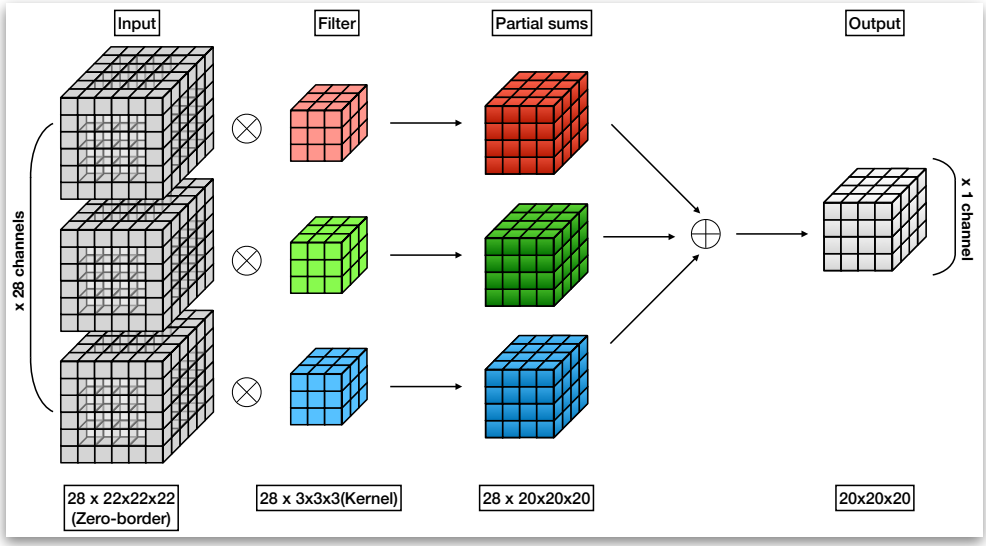
A multilayer perceptron (MLP) is a type of artificial neural network that consists of multiple layers of interconnected neurons, organized into an input layer, one or more hidden layers, and an output layer. It is a feedforward neural network, meaning the data flows only in one direction, from the input layer through the hidden layers to the output layer. Each neuron in an MLP receives input from the neurons in the previous layer, and each connection between neurons is associated with a weight. The network is trained using supervised learning methods, adjusting the weights to minimize the difference between the predicted outputs and the actual outputs, allowing it to learn complex patterns and solve various machine learning tasks such as classification and regression.



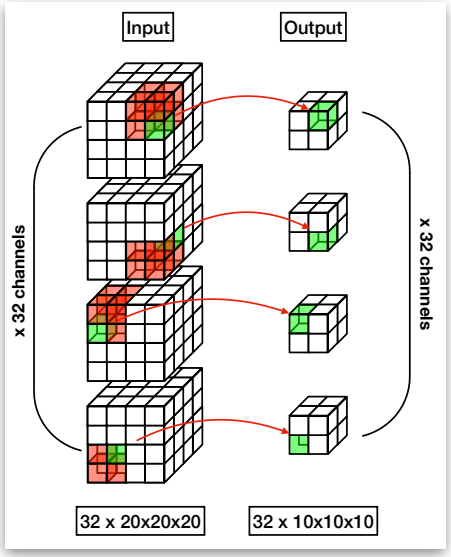
- Each neuron in a **fully-connected layer** in an MLP is constructed from all the neurons of the previous layer.
- The **blue** color is the **forward propagation** of information, and the **red** color is the **backpropagation** of information.

Convolutional Neural Network (CNN)

A Convolutional Neural Network (CNN) is a type of deep learning model specifically designed for image and video processing tasks. It utilizes a specialized architecture that includes convolutional layers, pooling layers, and fully connected layers. The key feature of CNNs is their ability to automatically learn hierarchical patterns and features from input data by applying convolutional filters. These filters slide over the input image to detect local patterns, edges, and textures. The pooling layers then downsample the feature maps, reducing their spatial dimensions while retaining important information. CNNs have been highly successful in tasks like image classification, object detection, and image generation due to their ability to efficiently capture and represent complex visual patterns.



An illustration of the **three-dimensional convolution** operation applied to the input layer using a single filter, composed of 28 kernels. This process transforms the 28 input channels into a **singular output channel**. The **quantity** of **output channels** directly corresponds to the **number of filters** used during the convolution.



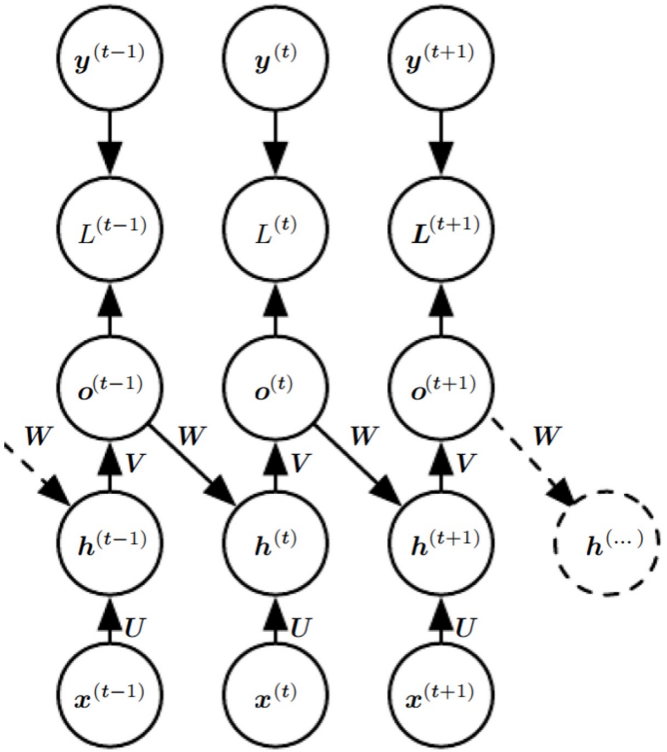
An illustration of the **three-dimensional pooling** operation. Despite retaining the original number of channels, the data dimensions are **halved**. For instance, a **20 x 20 x 20** cube is **reduced** to a **10 x 10 x 10** cube through this process.

Recurrent Neural Network (RNN)

A Recurrent Neural Network (RNN) is a type of artificial neural network designed to process sequential data, such as time series, natural language, and audio. Unlike feedforward neural networks, RNNs have feedback connections that allow them to retain information from previous time steps, making them capable of handling inputs with temporal dependencies. This feedback loop allows RNNs to maintain an internal state, also known as a hidden state, which captures the context and history of the sequential data. The hidden state is updated at each time step as the network processes new input. RNNs are widely used for tasks like language modeling, speech recognition, machine translation, and sentiment analysis, where the order and context of the input data are crucial for making accurate predictions. However, standard RNNs can suffer from vanishing or exploding gradient problems, leading to difficulties in learning long-term dependencies. As a result, variants like Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) were developed to address these issues and became more popular choices for many sequence-related tasks.

- Used for **time-ordered** (sequential) **data**
- Can **generate future states**
- Successful in **translation**, **text generation tasks**
- Use cases in **HEP** being actively **investigated**
- **Limited parallelization** (**slow**)

Training this class of networks will **benefit** greatly from more **computing power**.



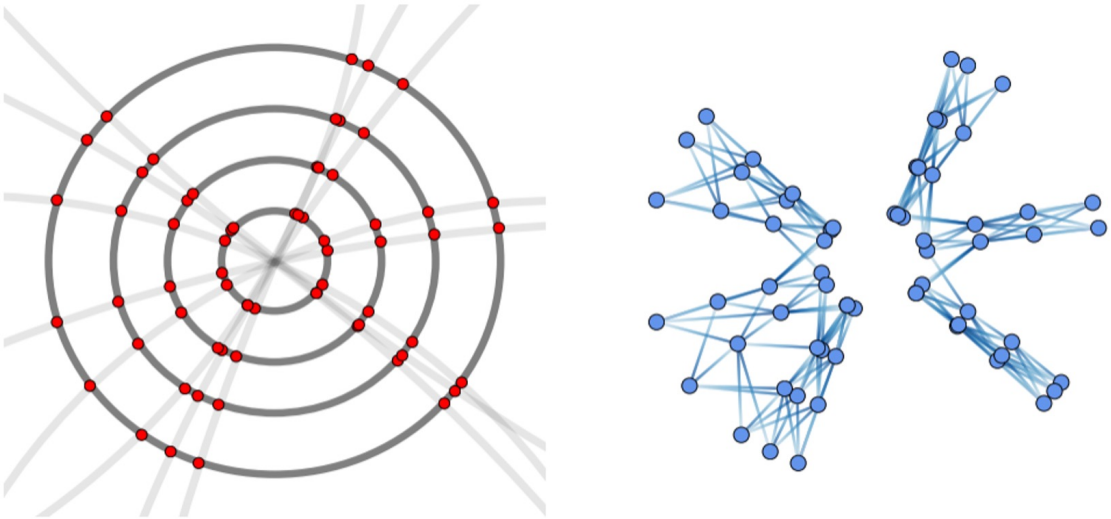
Goodfellow et al. (2016)

Graph Neural Network (GNN)

A Graph Neural Network (GNN) is a type of neural network designed to process and learn from graph-structured data. Graphs consist of nodes representing entities or data points, and edges representing relationships or connections between the nodes. GNNs leverage the structural information present in the graph to perform tasks like node classification, link prediction, graph classification, and more.

GNNs typically operate in a message-passing framework, where information from neighboring nodes is propagated and aggregated iteratively through the graph. Each node updates its representation by considering information from its neighbors and itself, enabling the network to capture local and global dependencies within the graph. By repeatedly passing messages and updating node features, the GNN can learn to extract meaningful representations and make predictions based on the graph's topology and node attributes.

Their ability to handle irregular and complex data structures makes them powerful tools for tasks involving interconnected data points.

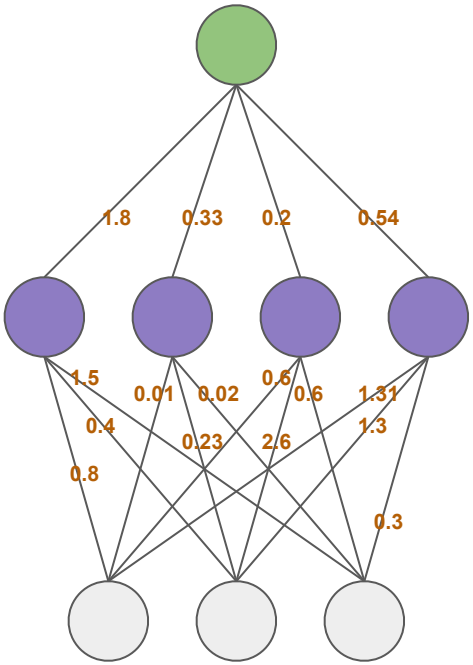


(left) Segments of hits in a tracking detector, (right) representation as a graph
Fig. from arXiv: 2012.01249

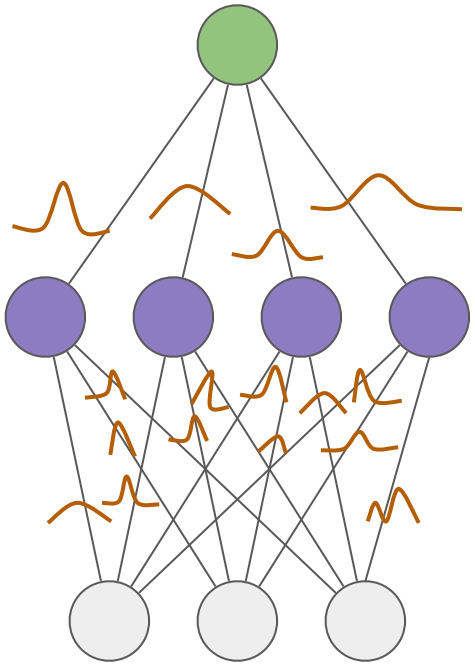
- A lot of **HEP data** can be represented as **graphs**.
- For example: **Hits** in a detector can be seen as **nodes** and so **track finding** becomes edge **classification problem**.

Bayesian Neural Network (BNN)

A Bayesian Neural Network (BNN) is a type of neural network that incorporates Bayesian inference principles to provide uncertainty estimates for its predictions. Unlike traditional neural networks, which output deterministic values, BNNs output probability distributions over the possible outcomes. They assign uncertainty to predictions, making them useful in decision-making processes that require quantifying uncertainty, such as in safety-critical applications or when dealing with limited or noisy data. BNNs have shown promise in applications like regression, classification, and reinforcement learning.



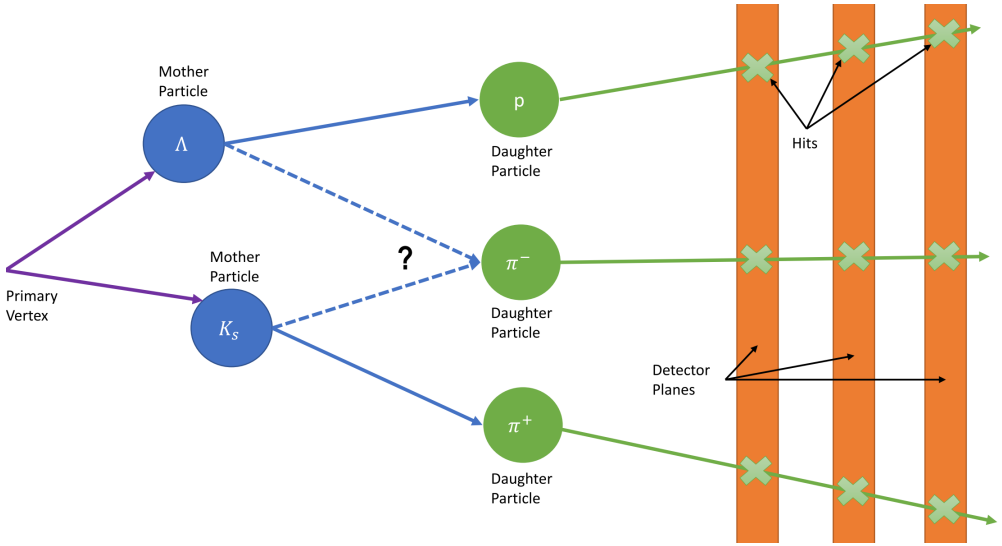
Point estimate for
the model parameters/weights



Probability distribution over weights

- ***Stochastic*** Neural Networks
- ***Data-efficient*** and reduce ***over-fitting***
- Combine ***flexibility*** of neural networks with ***Bayesian*** framework
- Capture ***Epistemic Uncertainty***

ANN Based Particle Competition in the KF Particle Finder

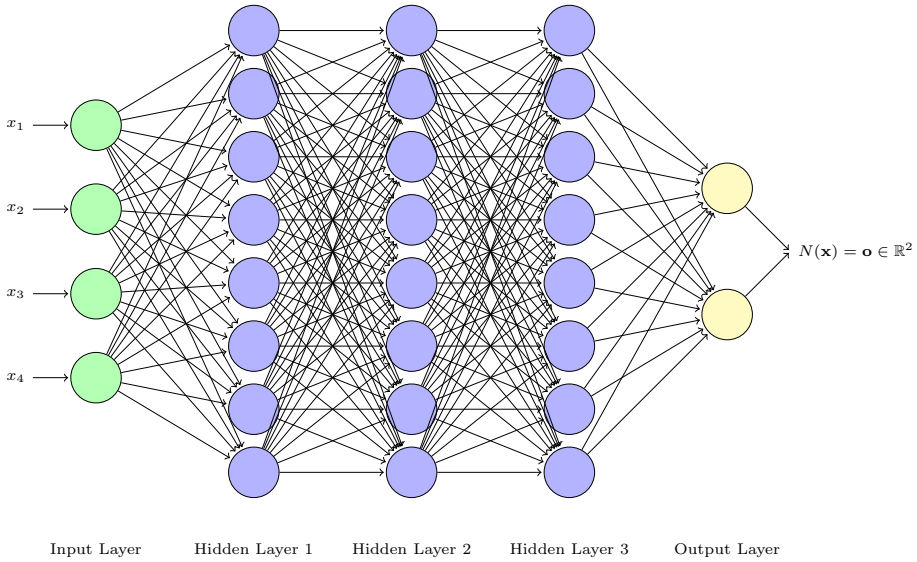


The search for short-lived particles requires a full event reconstruction, including all particle decays and decay chains. The KF Particle Finder reconstructs **all possible mother particle** candidates based on the information of daughter particles that intersect in time and space.

- For instance, π^- can be combined with π^+ as a decay of K_s or with p as a decay of Λ. KF Particle Finder **reconstructs both**, if reasonable
- Only one mother particle candidate exists (**signal**), the other does not (**ghost / background**)
- To **reduce** the created ghost / background particles by this approach, a **competition between mother particle** candidates can be enabled
- The competition searches for the **best matching mother particle** candidate and rejects the other

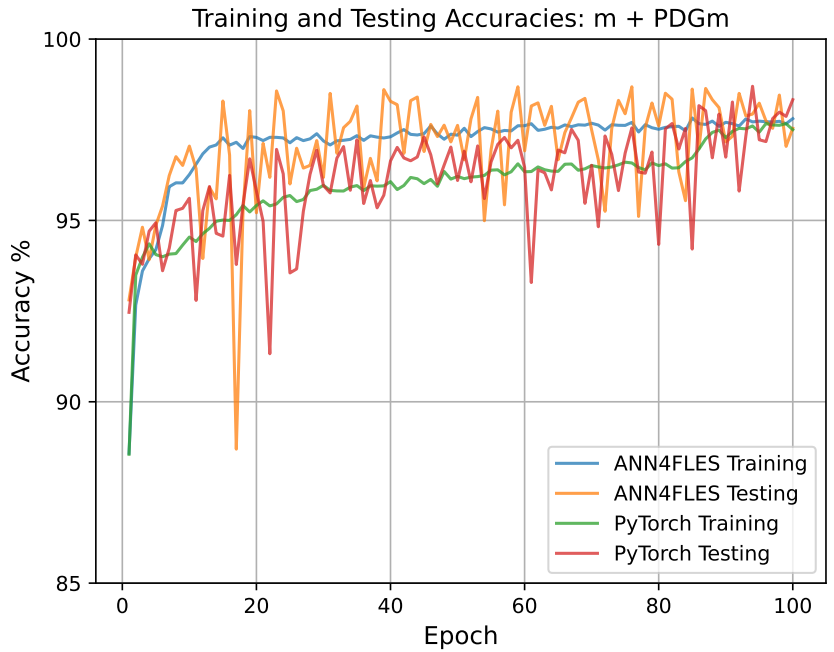
A Multilayer Perceptron is used to solve the particle competition in KF Particle Finder

MLP Architecture for Particle Competition

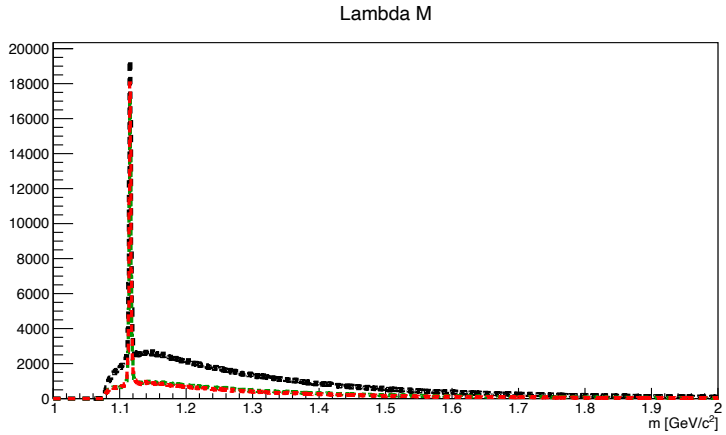


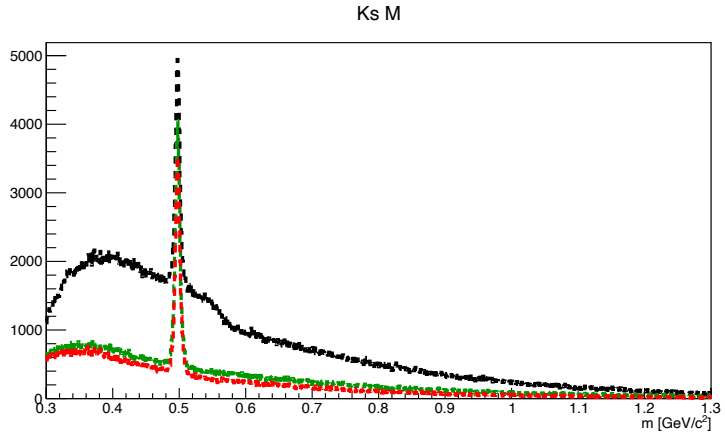
- Reconstructed **particle mass** and **PDG mass** of each candidate used for classification of Λ and K_S^0
- **LeakyReLU** hidden activation
- **Softmax** output activation
- **Cross-Entropy loss**
- **ADAM** weight optimisation

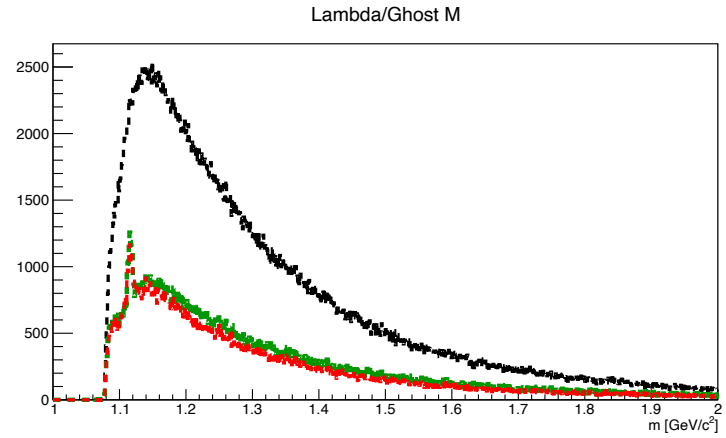
- **ANN4FLES** comparison to **similar PyTorch** model
- Both achieve **over 98% accuracy** of raw classification performance on the test set

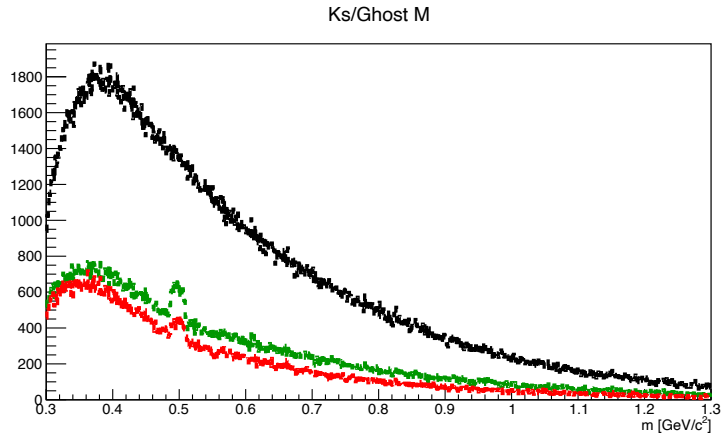


MLP for Particle Competition: Results









ANN4FLES inside **KF Particle Finder** - Total mass spectra and ghosts

Black: No Competition

Green: KF Particle Finder deterministic competition

Red: Neural network based competition

ANN based competition reduced the number of ghost particles compared to the existing competition of the KF Particle Finder

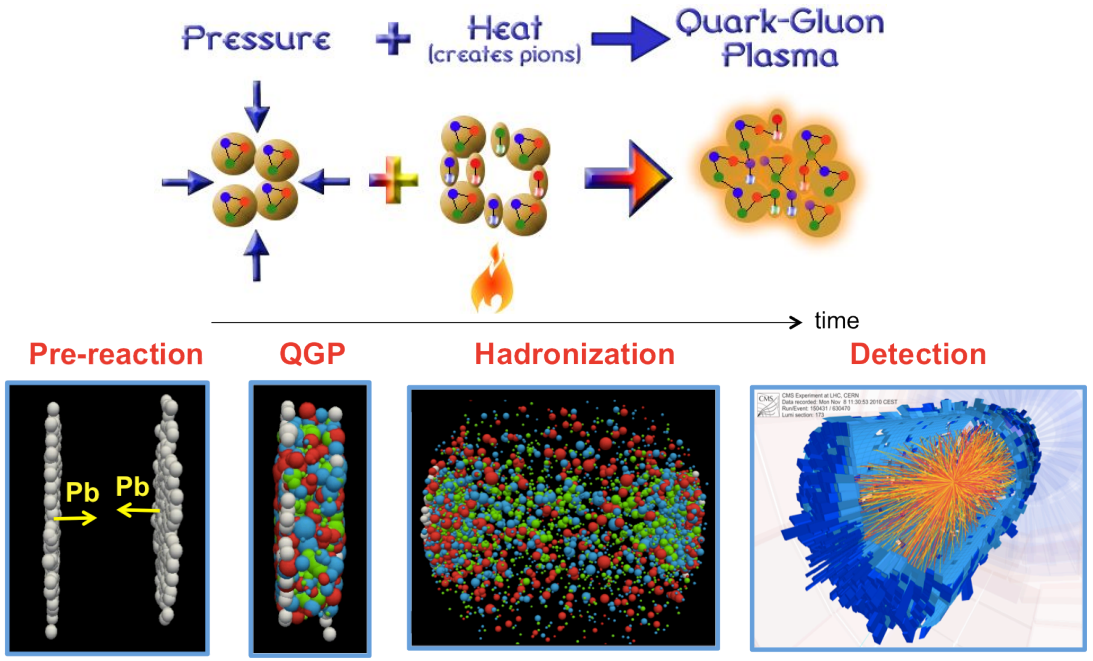
QGP - a State of Strongly-Interacting Matter

Quark-Gluon Plasma (QGP) is a state of matter that is believed to have existed in the very early universe, shortly after the Big Bang, and can also be created in extreme conditions, such as high-energy collisions between heavy atomic nuclei. It is a unique and exotic form of matter in which individual quarks and gluons, which are elementary particles and the building blocks of protons and neutrons, are no longer confined within hadrons (protons and neutrons).

Under normal conditions, quarks and gluons are always bound together inside particles like protons and neutrons due to the strong force, one of the four fundamental forces of nature. However, in extreme temperatures and energy densities, such as those present in the early universe or in high-energy nuclear collisions, the strong force becomes weaker, leading to the liberation of quarks and gluons from their hadronic confinement.

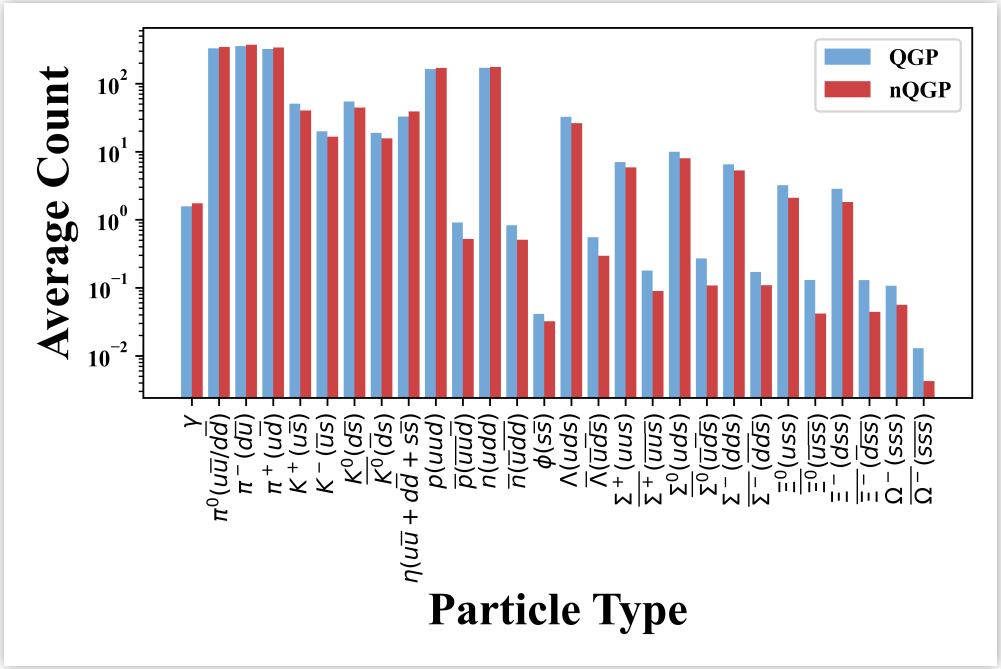
The study of Quark-Gluon Plasma is a crucial area of research in high-energy nuclear physics, particularly in experiments conducted at facilities like the Large Hadron Collider (LHC) and the Relativistic Heavy Ion Collider (RHIC). Understanding QGP helps scientists gain insights into the fundamental properties of strong nuclear interactions and the behavior of matter under extreme conditions, shedding light on the early universe and the evolution of the cosmos.

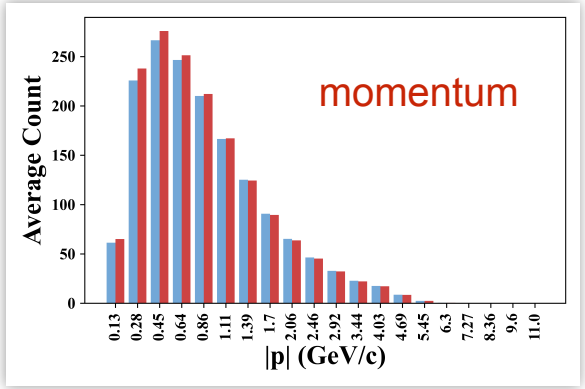
- A QGP can be formed by **compressing** a large amount of energy into a small volume
- collide **heavy nuclei**
- **control/vary the energy deposited** in the collision region by varying the collision system
- **no direct observation** of the QGP is possible
- rely on emerging particles as **probes**
- classify events based on final-state **charged particle multiplicity**

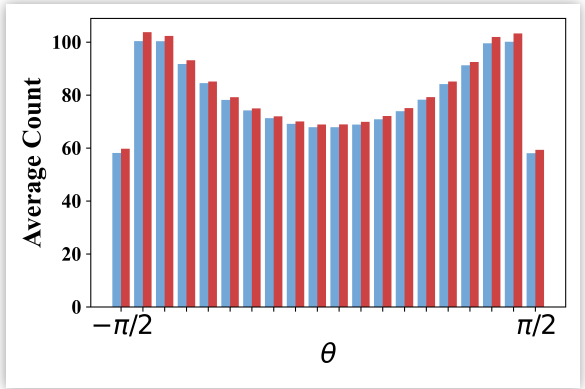


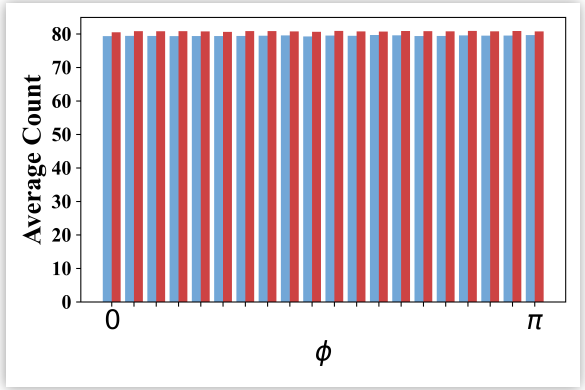
A **QGP Trigger** is necessary for the classification of events

Dataset for QGP Trigger



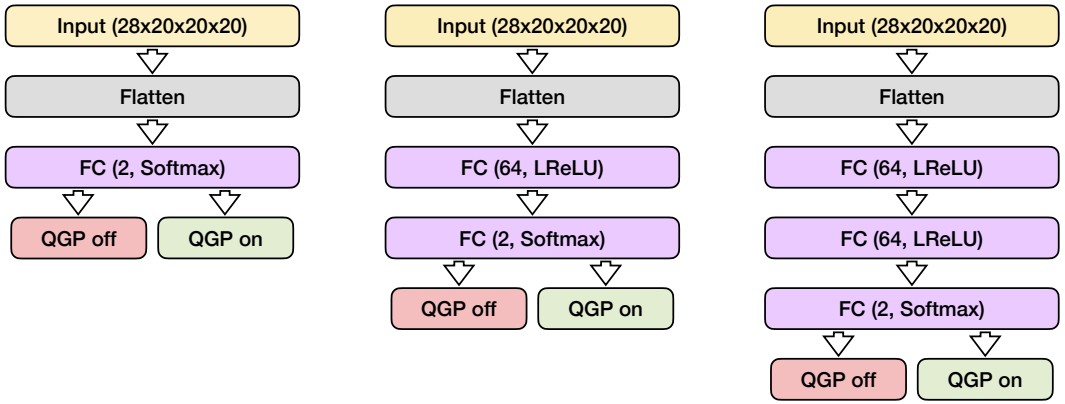




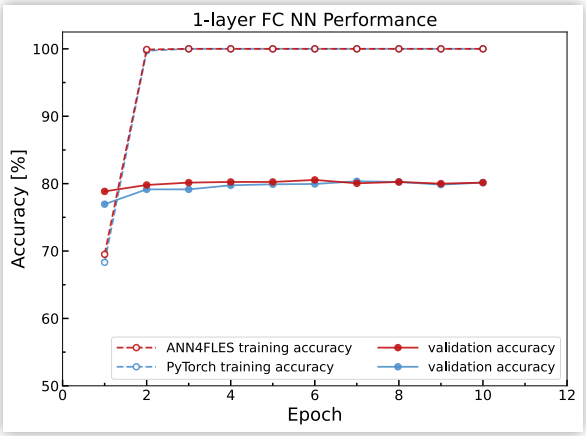


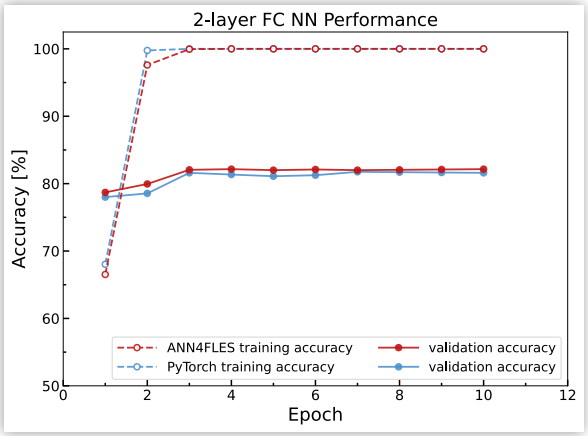
- 10000 simulated events (5000 **with QGP** and 5000 **without QGP**)
- **central** AuAu **collisions** at 31.2 GeV
- 1 event consist information about **~1600 particles**
- particles are distributed into **4D** matrix:
  - 28 bins - **particle types**
  - 20 bins with log scale - particles **momentum**
  - 20 bins - inclination **angle Θ**
  - 20 bins - azimuth **angle φ**

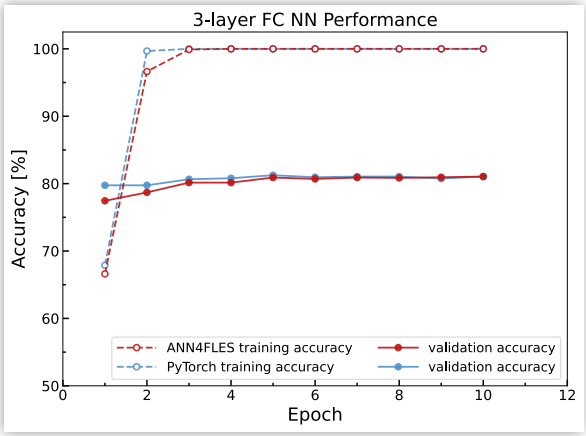
Architectures of Neural Networks in ANN4FLES



From left to right: structure of **one-**, **two-** and **three-layer fully-connected** neural networks used for QGP detection

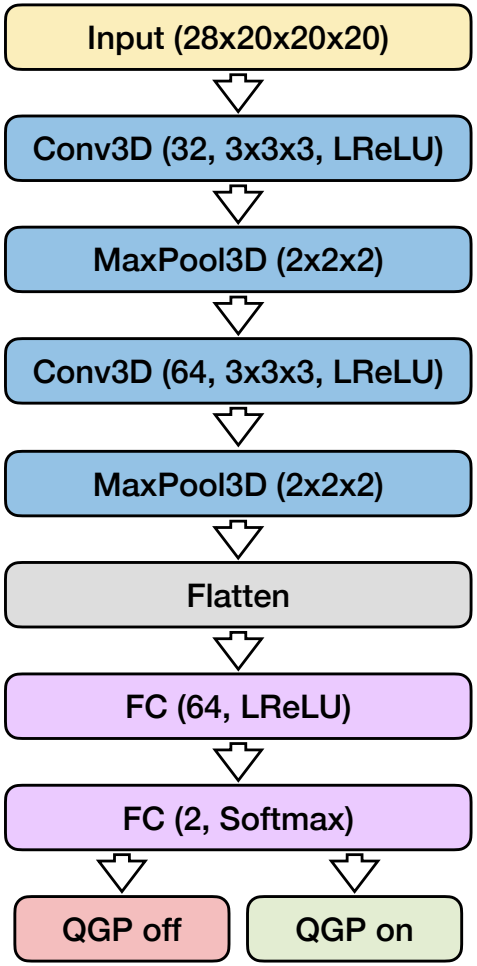




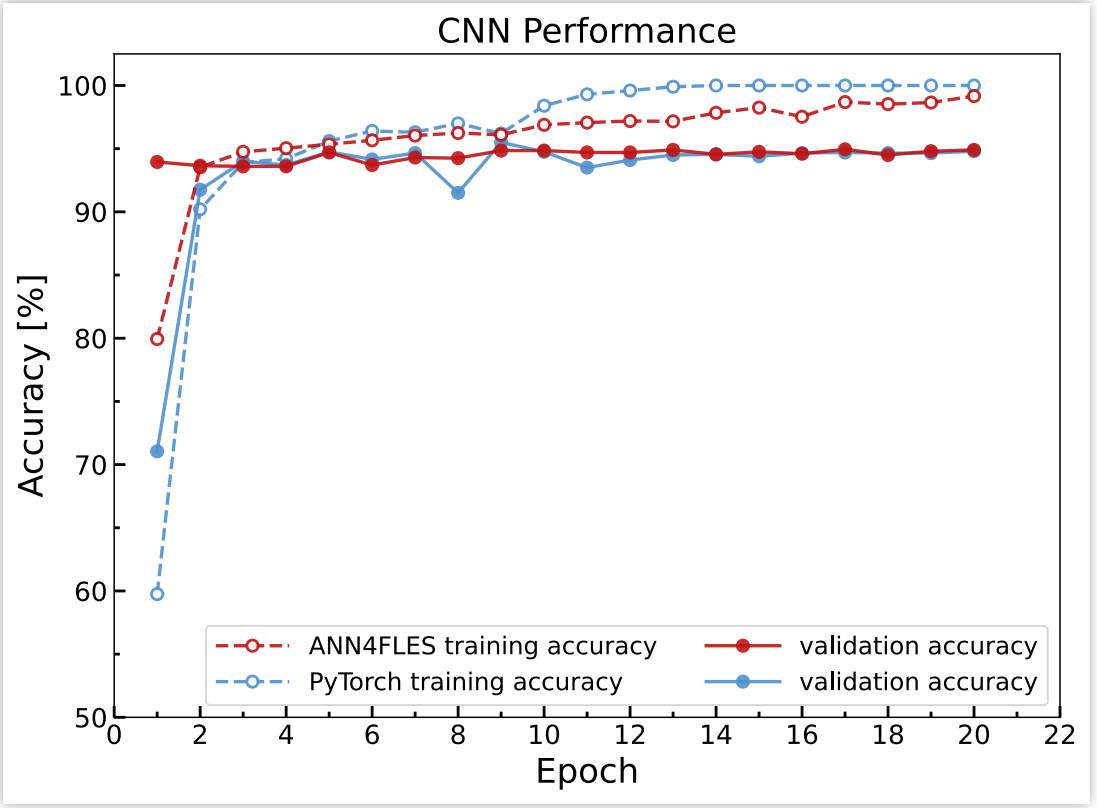


Training and **validation** accuracy for the **FC** networks

Architectures of Neural Networks in ANN4FLES



Structure of the **convolutional** neural network used for QGP detection.



Training and **validation** accuracy for the **CNN**

Shapley Additive Explanations (SHAP)

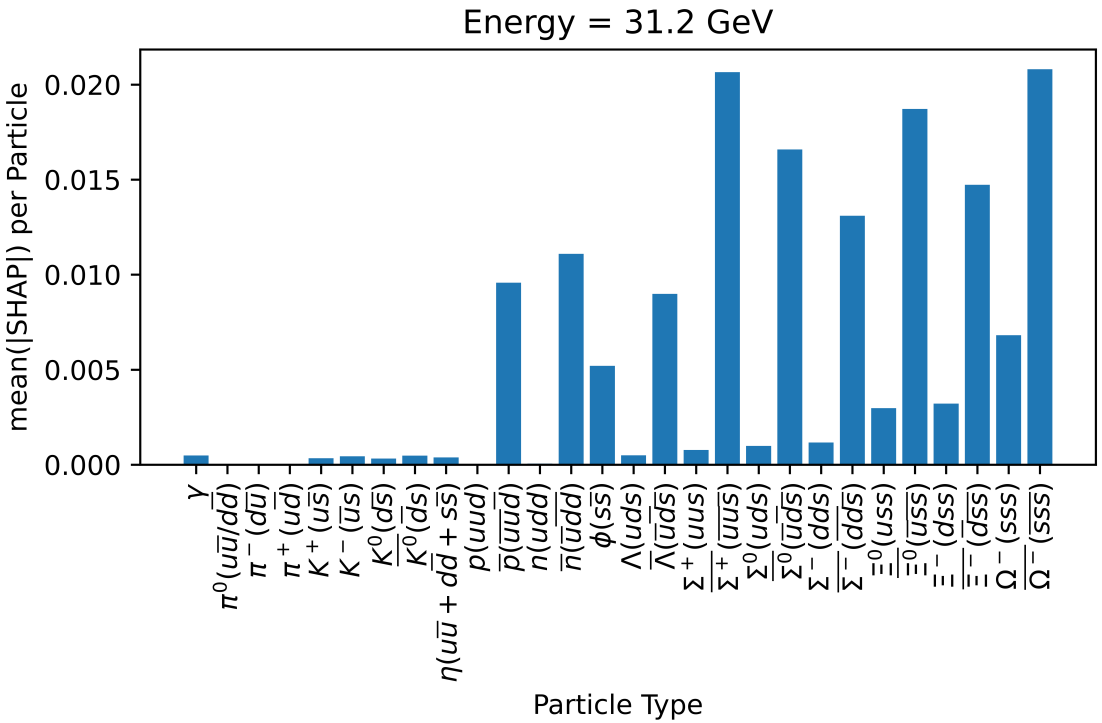
Method based on cooperative game theory used to increase transparency and interpretability of machine learning models.

For each feature, SHAP score is determined by evaluating the average contribution of adding the feature over all possible feature subsets defined without that feature.

- **Pions contribute little** to model prediction
- **Anti-baryons more important** than baryons for all particles

From theory we expect **QGP formation** to involve

- more **strange quarks**
- more **anti-baryons than baryons**



SHAP analysis reveals that the neural network has learned the correct characteristics associated with QGP production

Summary

- A C++ package of Artificial Neural Networks for the First Level Event Selection (ANN4FLES) was created in the CBM experiment at FAIR, Germany.
- The ANN4FLES package allows the user to build different neural network architectures with minimal additional programming overhead.
- The package features a graphical user interface (GUI) where the user can not only select the type of network but also customize various hyperparameters.
- The package is built in a modular fashion, so users who wish to implement additional non-trivial features can do so without changing the basic structure of the package.
- The following networks are currently implemented in ANN4FLES: Multilayer Perceptron (MLP), Convolutional Neural Network (CNN), Recurrent Neural Networks (RNN), Graph Neural Networks (GNN), and Bayesian Neural Network (BNN).
- All networks implemented in the package have been successfully tested on a number of standard datasets and show comparable results to the PyTorch library.
- The ANN4FLES package has been successfully applied to two classification tasks in the CBM experiment:
  (1) to search for short-lived particles and
  (2) to select collisions with Quark-Gluon Plasma.