ML and Bayesian Inference applications in heavy-ion physics

Andrea Dubla
Carolina Reetz

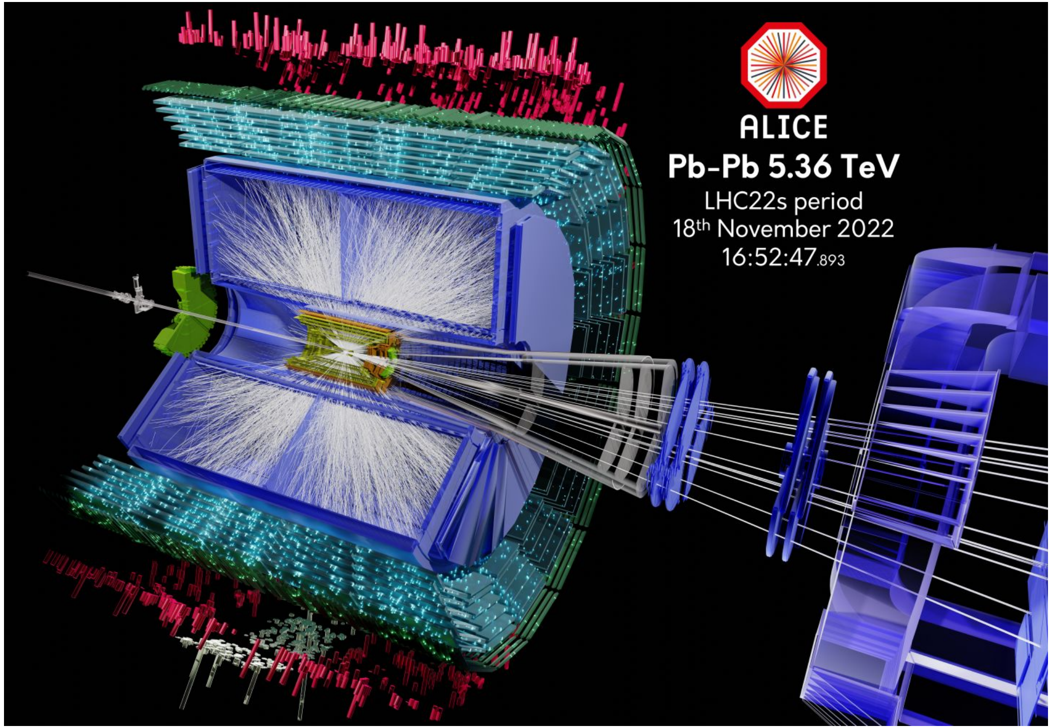
a.dubla@gsi.de
c.reetz@cern.ch

Part I
ALICE data analysis

Analysis of high energy heavy-ion collisions at the LHC

- Large charged particle multiplicity in high energy pp and heavy-ion collisions
- **Large background** makes measurement of **rare probes** extremely challenging
- ML techniques help to **isolate signals** based on topological, kinematic, and PID properties

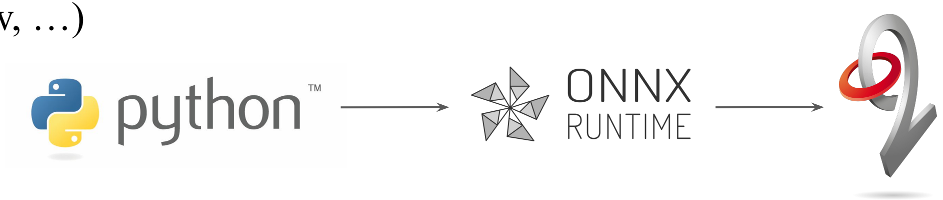
ML is a crucial analysis tool to take advantage of large statistics collected during Run 3 at the LHC



Usage of software in ALICE data analyses

- ML applications in ALICE analyses mainly based on python software stack:
  - scikit-learn, XGBoost, TensorFlow, PyTorch, …

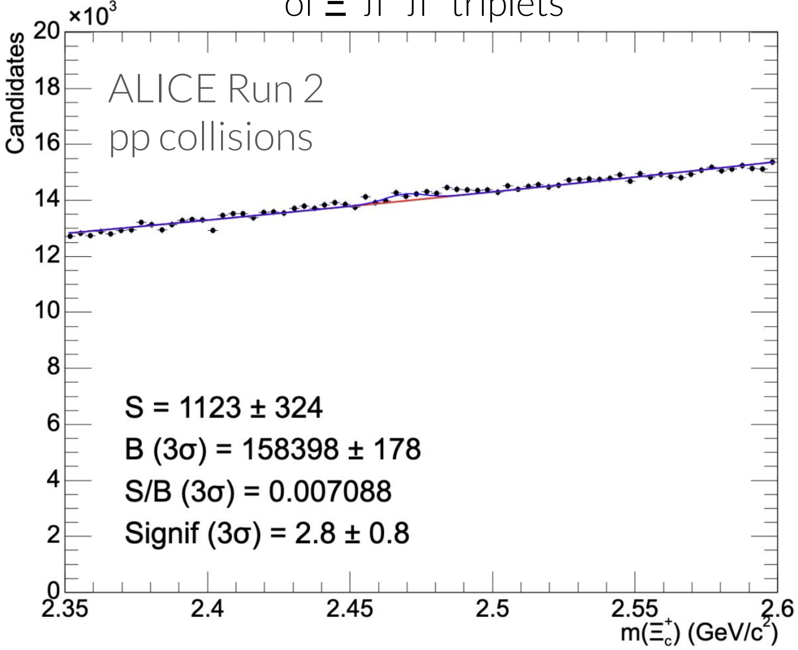
- **ALICE Run 3 data analysis:**
  - **ALICE O^2 (online-offline) and O^2Physics software:** framework and detector specific code for reconstruction, calibration, simulation and analysis for the ALICE experiment for Run 3 and Run 4
  - ML integration in O^2/O^2Physics framework via **ONNXRuntime**
  - Supports almost any ML model (BDT, NN, …) and library (XGBoost, PyTorch, TensorFlow, …)



A needle in a haystack

- Analysis of **short-lived charm-strange baryons** in Run 2 proton-proton collisions
- Reconstruction of decay vertex from measured daughter trajectories in the detector
- Measurement of $\Xi^+_c \rightarrow \Xi^- \pi^+ \pi^+$ **at low transverse momenta (p_T)** in pp collisions at $\sqrt{s}$ = 13 TeV
    - "Cascade"-like decay with high combinatorial background
    - "Playground" for measurement in high multiplicity Pb-Pb collisions

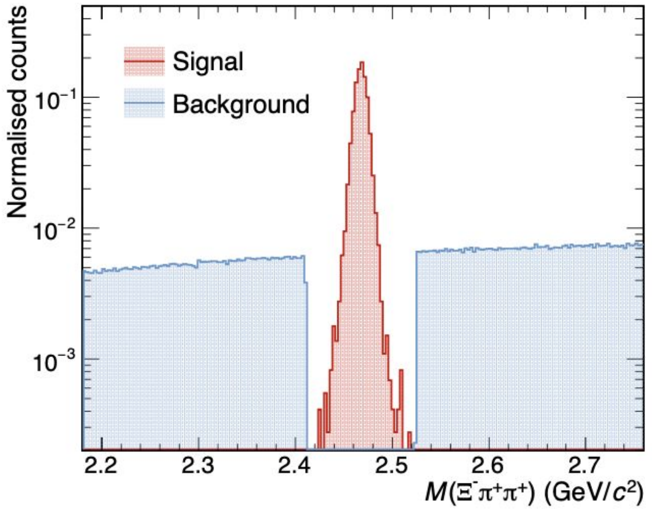
Invariant mass spectrum of $\Xi^- \pi^+ \pi^+$ triplets

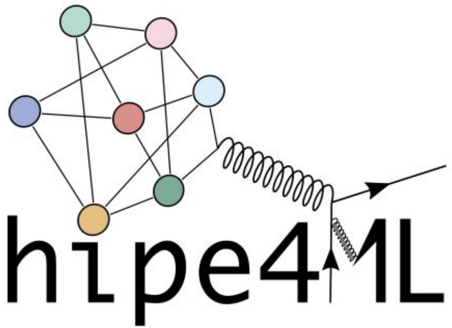


Usage of public ML package

- Focus on **Boosted Decision Trees (BDTs) for (multi-)classification** task
  → classification of reconstructed signal and background candidates
- BDT model is trained and tested on **signal from Monte Carlo** simulations and **realistic background from data**
  - Usage of **public ML package hipe4ml** developed in ALICE for ML based high-energy physics analyses
- Application of trained model to reconstructed data using the O^2Physics interface for ONNXRuntime

Invariant mass spectrum of
$\Xi^- \pi^+ \pi^+$ triplets

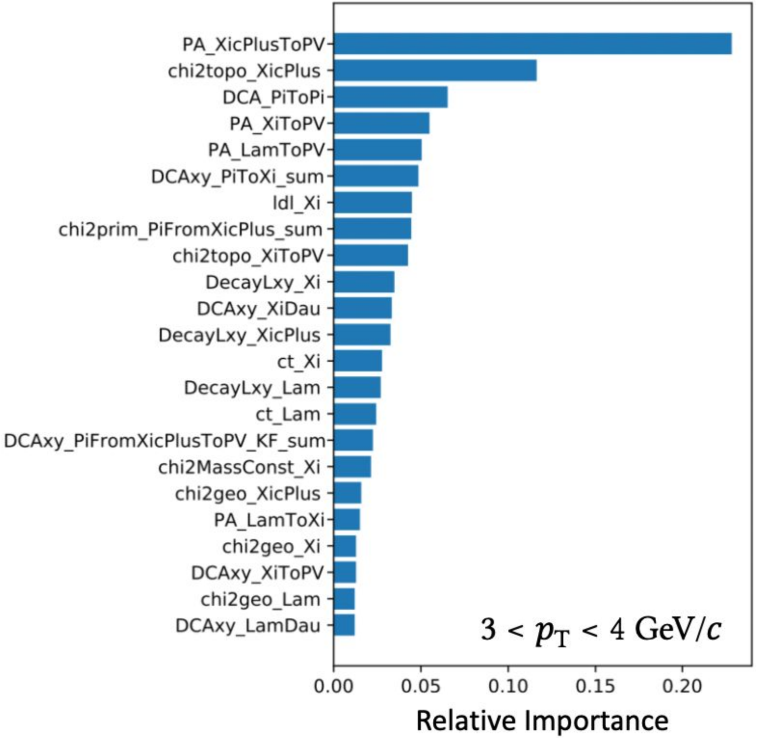




Binary classification with BDTs

- Different properties of the decay (topological, kinematic, daughter particle identification) are studied regarding their separation power

Input feature importance

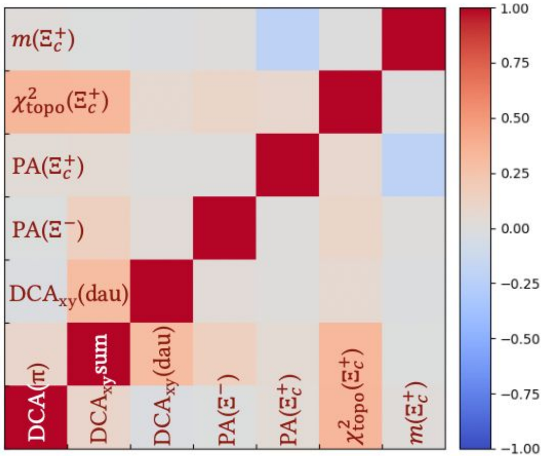


Carolina Reetz (master thesis)

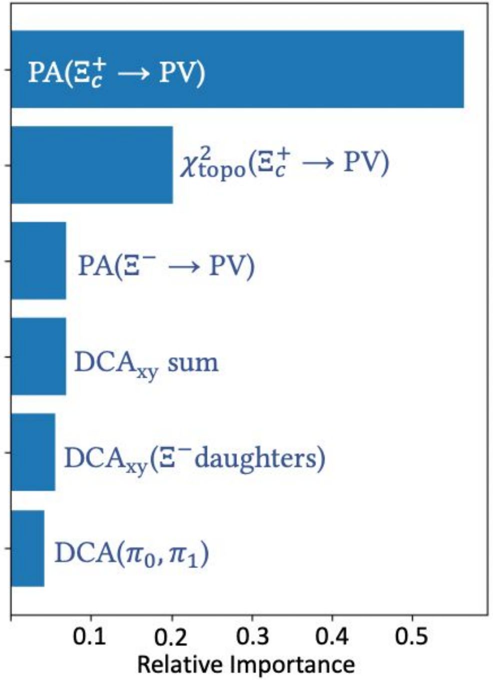
Binary classification with BDTs

- Different properties of the decay (topological, kinematic, daughter particle identification) are studied regarding their separation power
- The most relevant ones are selected as input features to train the BDT model
- Correlations between input features can be exploited by the model to separate the classes

Input feature correlations



Input feature importance

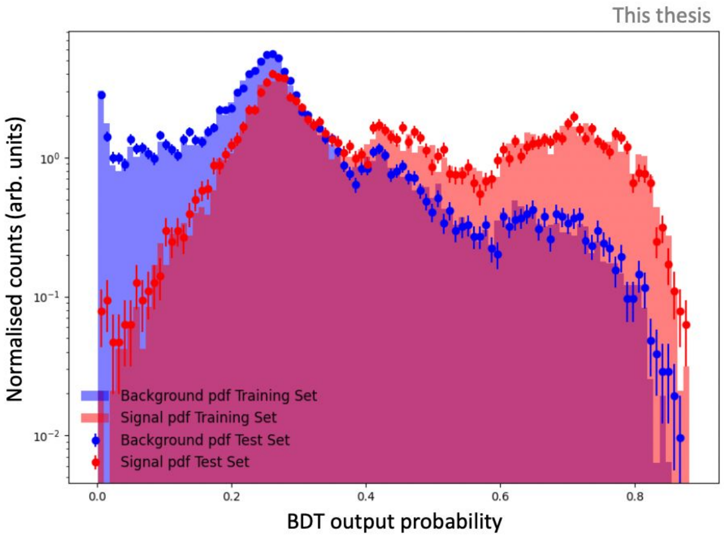


(a) $3 < p_T < 4\,\text{GeV}/c$.

Carolina Reetz (master thesis)

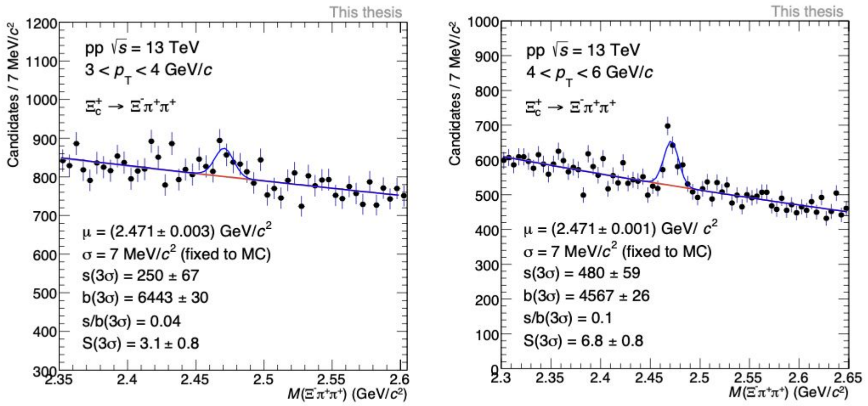
Binary classification with BDTs

- Reconstructed candidates get assigned a BDT output score
  → used to select signal



(a) $3 < p_T < 4\,\mathrm{GeV}/c$.

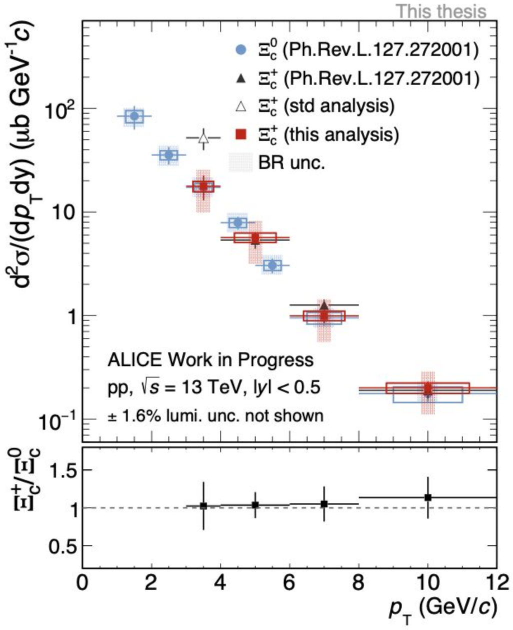
Invariant mass spectra of selected $\Xi^-\pi^+\pi^+$ triplets



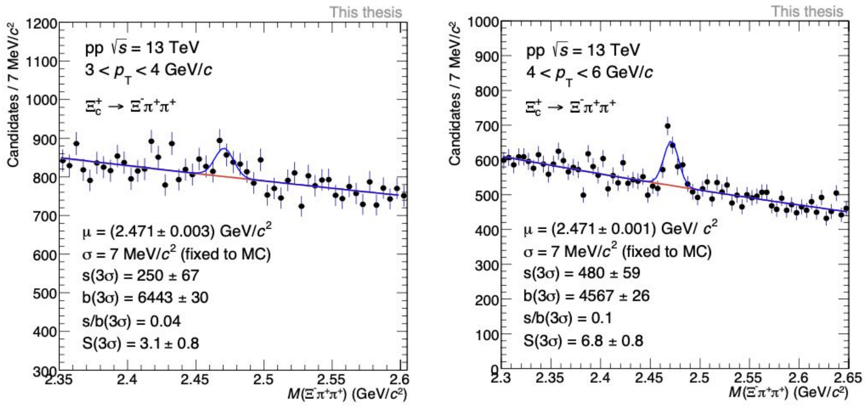
Carolina Reetz (master thesis)

Binary classification with BDTs

- Reconstructed candidates get assigned a BDT output score
  → used to select signal
- Makes production cross section measurement of rare probes possible!



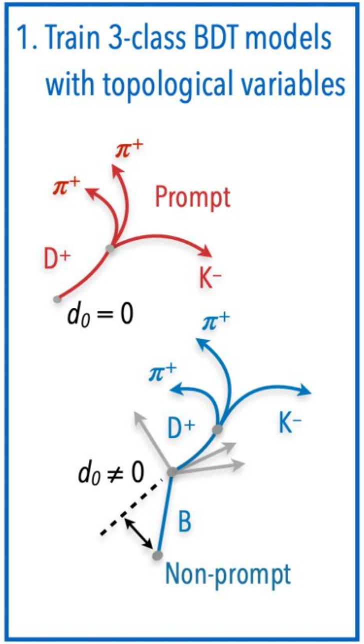
Invariant mass spectra of selected $\Xi^-\pi^+\pi^+$ triplets



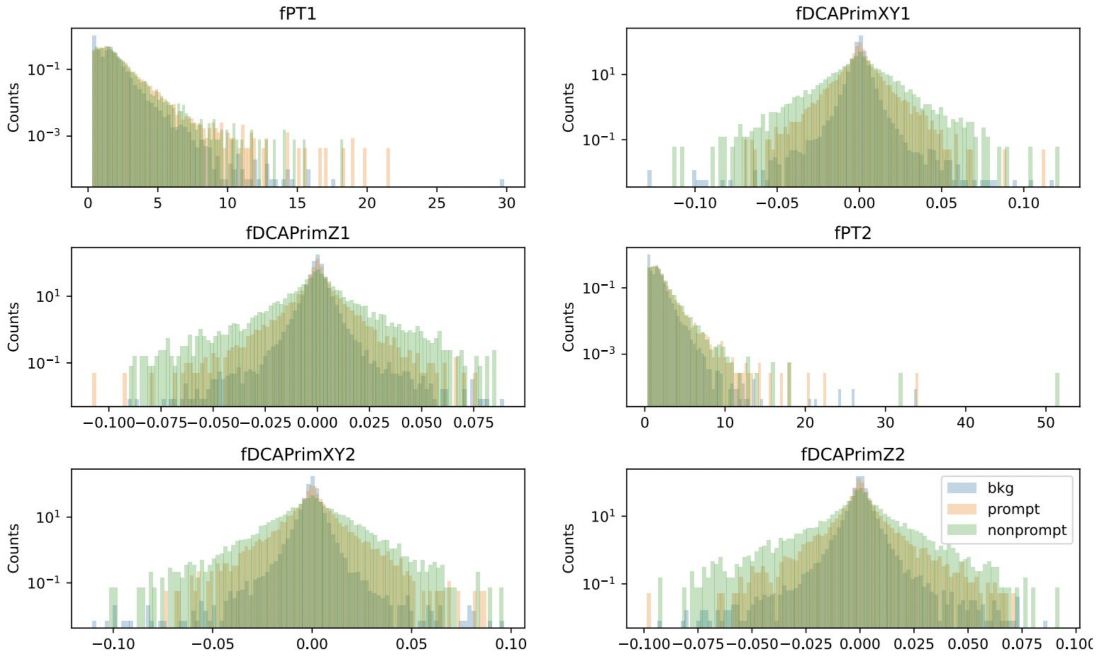
Carolina Reetz (master thesis)

Machine learning for (multi-)classification

- Measurement of charm mesons from beauty feed-down
  → classification of reconstructed prompt, non-prompt, and background $D^{\pm} \to K^{\mp}\pi^{\pm}\pi^{\pm}$ candidates



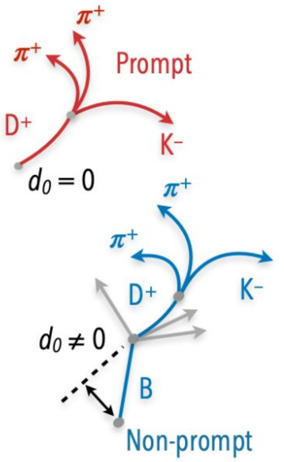
3-class BDT output scores



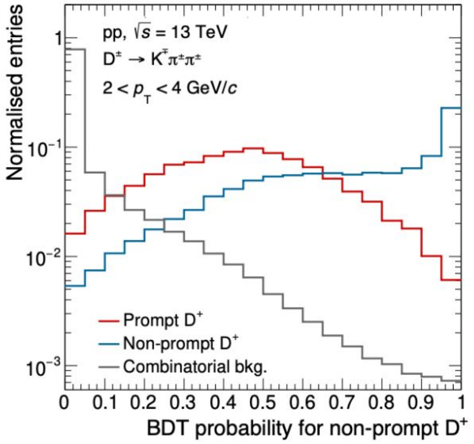
Machine learning for (multi-)classification

- Measurement of charm mesons from beauty feed-down
  → classification of reconstructed prompt, non-prompt, and background $D^{\pm} \rightarrow K^{\mp}\pi^{\pm}\pi^{\pm}$ candidates

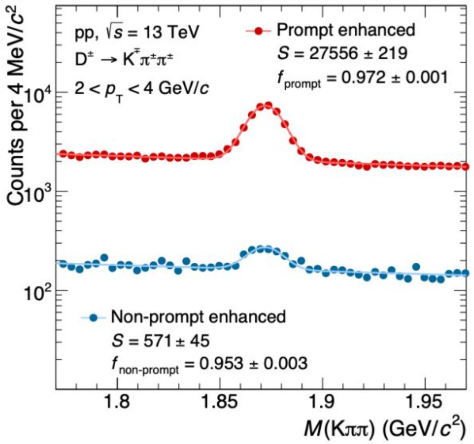
1. Train 3-class BDT models with topological variables



2. Apply models to D-meson candidates and obtain 3 output scores related to the probability to belong to each class

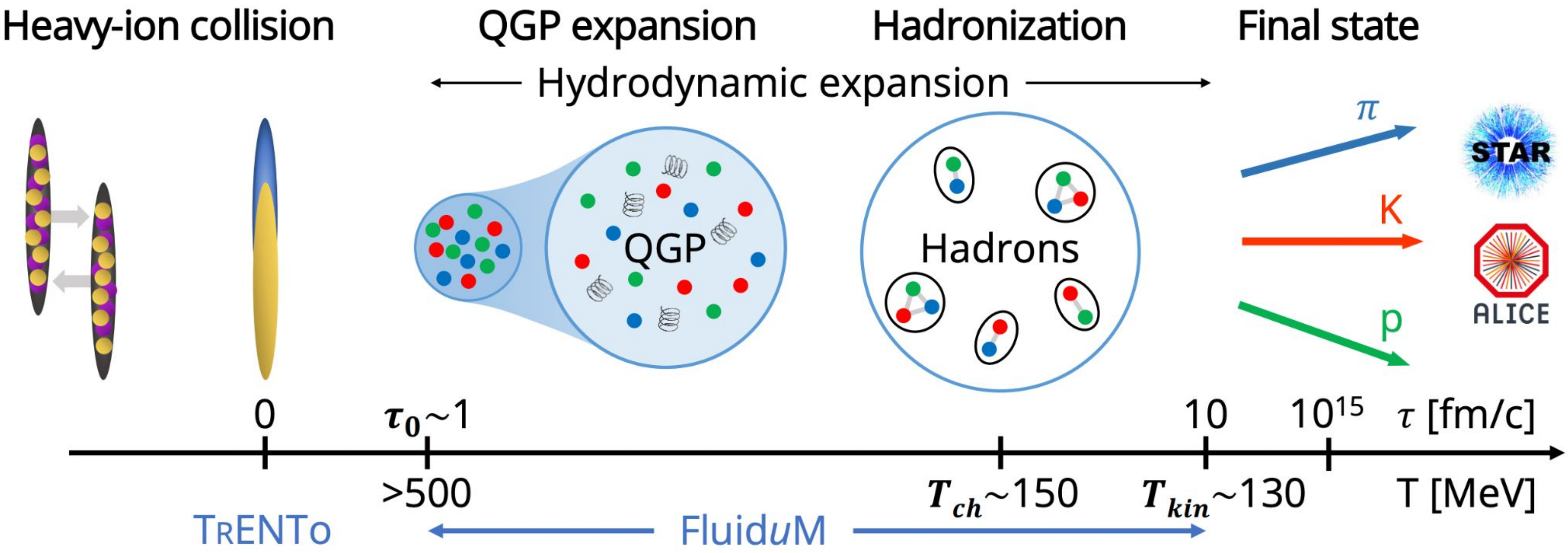


3. Apply selections on the output scorse and obtain very pure samples of D-meson candidates



Part II
Infer QCD parameters

Bayesian inference for QGP parameter estimation



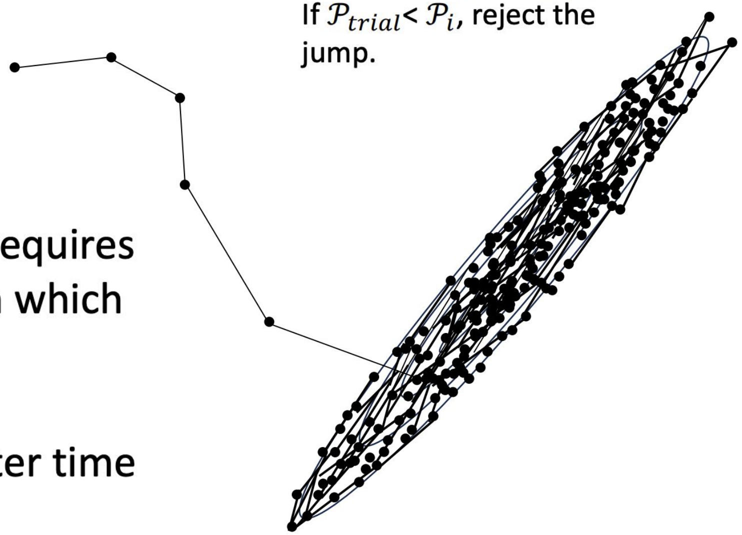
- Idea: Infer the model inputs (evolution parameters) using the experimental data
- Compute the likelihood: Probability of observing experimental data, given proposed parameters

Markov-Chain Monte Carlo

- How to calculate likelihood? Use Markov-Chain Monte-Carlo method (Metropolis algorithm)
- MCMC is a method for exploring the probability space (samples are drawn randomly but not independently).

$$P(Data|x) = \frac{1}{\sqrt{(2\pi)^n \det \Sigma}} \exp\left[-\frac{1}{2}\left[y_{model}(x) - y_{exp}\right]^T \Sigma^{-1}\left[y_{model}(x) - y_{exp}\right]\right]$$

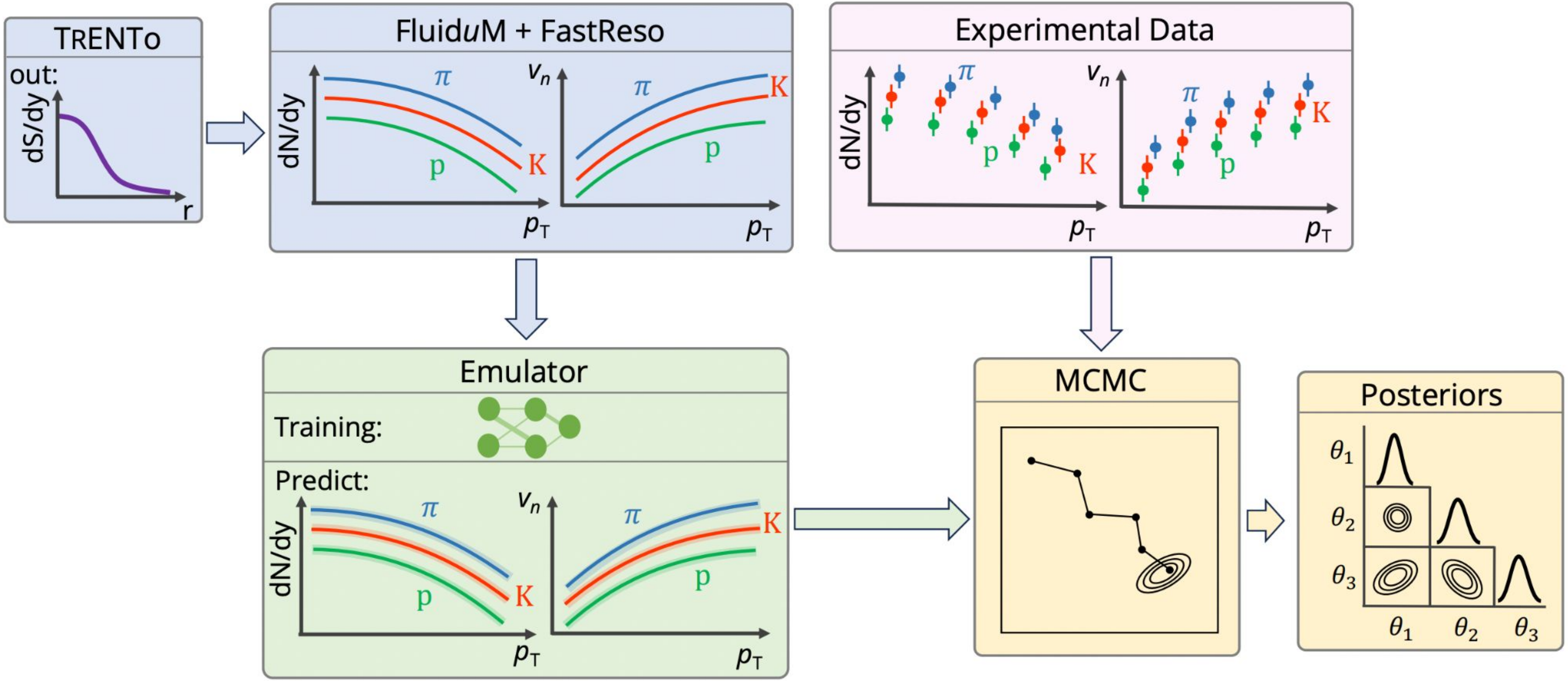
METROPOLIS RULE
If $\mathcal{P}_{trial} > \mathcal{P}_i$, accept the jump, so $\theta_{i+1} = \theta_{trial}$
If $\mathcal{P}_{trial} < \mathcal{P}_i$, reject the jump.



MCMC is powerful but…
Calculation of likelihood is very expensive. MCMC sampling requires many model evaluations to sample the posterior distribution which corresponds to long computational time.

Solution: Use **emulator** to predict the model results in shorter time

Bayesian inference chart



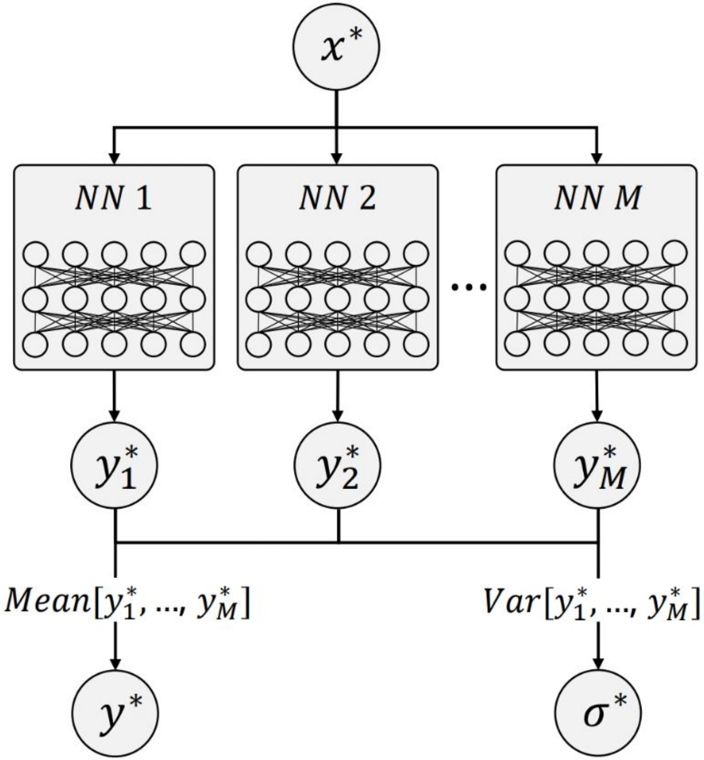
NN ensemble emulator

- Artificial Neural Networks can approximate arbitrary functions $f: \boldsymbol{x} \mapsto \boldsymbol{y}$, so they can approximate y_{model}
- Simple neural networks provide point predictions without any uncertainty

➢ Use ensemble of NNs for uncertainty estimation considering the correlation ρ among the NNs

1. Train multiple NN with different initialization to ensure diversity
2. Make predictions for one x^* for all NNs
3. Calculate mean and variance from the resulting predictions $y_1^*, \ldots, y_M^*$

$$\sigma_{ensemble} = \sqrt{\frac{\frac{1}{M} + \frac{M-1}{M}\rho}{1-\rho}}\,\sigma_{NN}$$



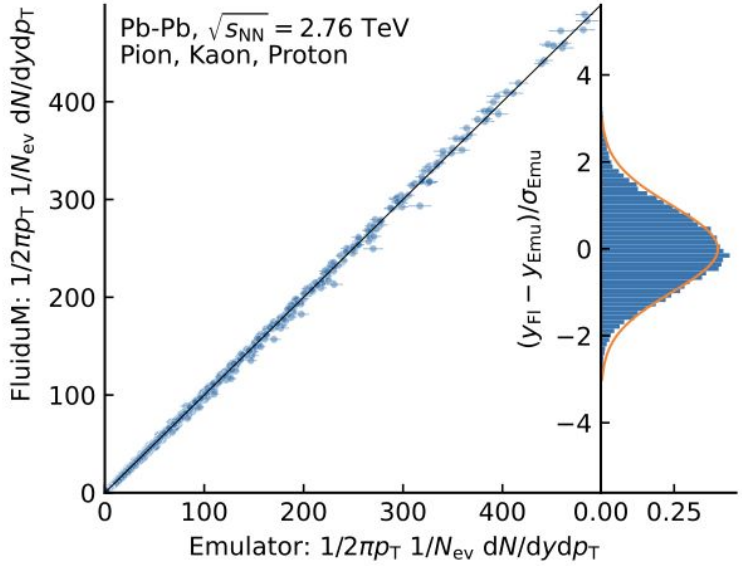
NN ensemble emulator performances

- Artificial Neural Networks can approximate arbitrary functions $f: \boldsymbol{x} \mapsto \boldsymbol{y}$, so they can approximate y_{model}
- Simple neural networks provide point predictions without any uncertainty

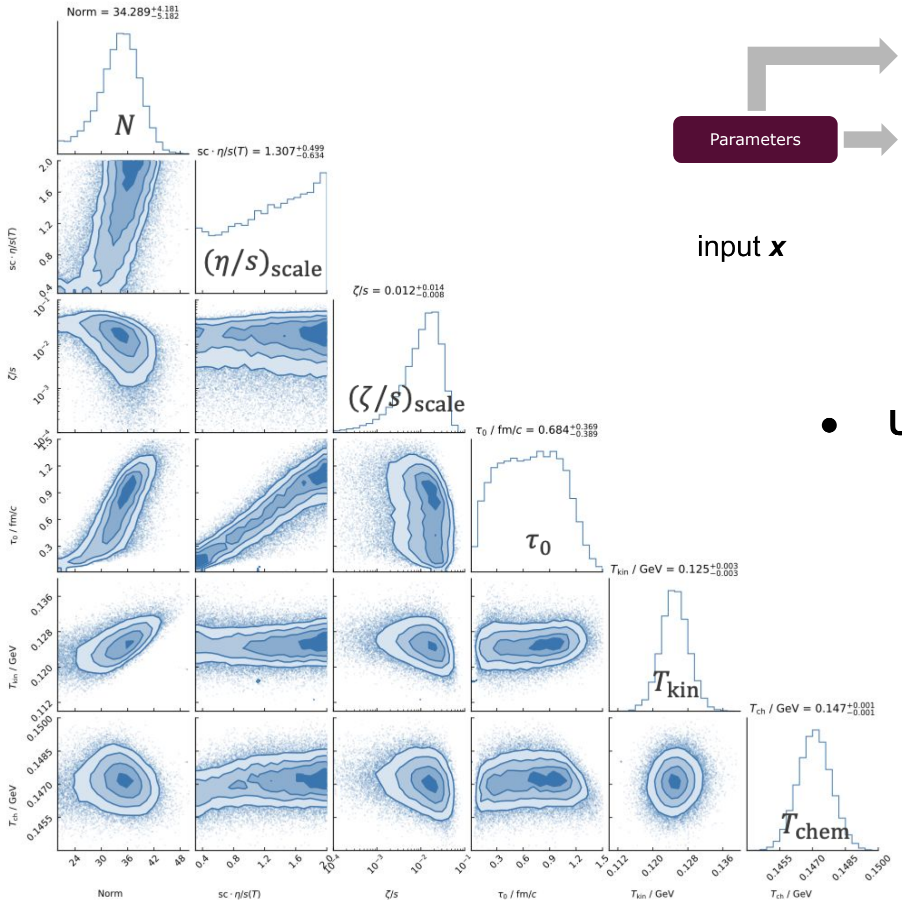
➢ Use ensemble of NNs for uncertainty estimation considering the correlation ρ among the NNs

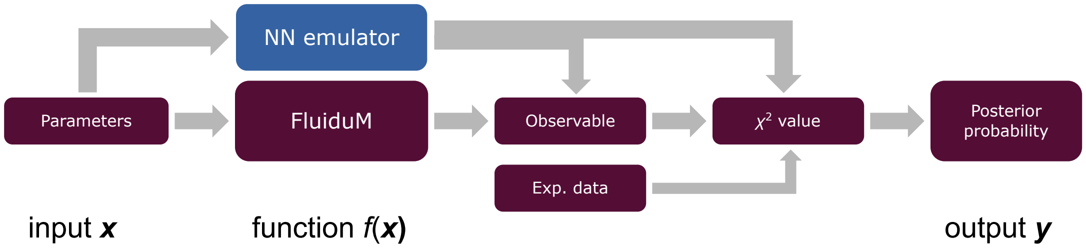
1. Train multiple NN with different initialization to ensure diversity
2. Make predictions for one x^* for all NNs
3. Calculate mean and variance from the resulting predictions $y_1^*, \ldots, y_M^*$

$$\sigma_{ensemble} = \sqrt{\frac{\frac{1}{M} + \frac{M-1}{M}\rho}{1-\rho}}\,\sigma_{NN}$$



Joint and marginal posterior distributions





- **Unique combination of tools** within Bayesian analyses
  - For the first time exploited an **ensemble of NN**
  - Expect a **significant advance** in pinning down QCD matter properties

Summary

Two main applications of ML and NN

1) Search for rare signal via topology decay reconstruction. Mainly in the area of charm and beauty hadrons reconstruction at the LHC with ALICE

2) Application of Bayesian inference analysis using NN emulator. Estimation of heavy-ion collision parameters and their uncertainties

Backup

Bayesian inference - priors

- Idea: Infer the model inputs (x = properties) using the experimental data
- Bayes' Theorem:

Likelihood, $\mathcal{L}$ Prior, π

Posterior, $\mathcal{P}$

$$P(\theta|Data, Model) = \frac{P(Data|\theta, Model)P(\theta|Model)}{P(Data|Model)}$$

Evidence, Z (Marginal likelihood)

- Prior: Initial knowledge of parameters

$$P(\theta|Model) = \begin{cases} 1, & if \min(\theta_i) \leq \theta_i \leq \max(\theta_i) \\ 0, & else \end{cases}$$

- Likelihood: Probability of observing experimental data, given proposed parameters

$$P(Data|x) = \frac{1}{\sqrt{(2\pi)^n \det \Sigma}} \exp\left[-\frac{1}{2}\left[y_{model}(x) - y_{exp}\right]^T \Sigma^{-1} \left[y_{model}(x) - y_{exp}\right]\right]$$