Reconstructing a generic B decay with a graph neural network

GDR-InF Annual Workshop 2023

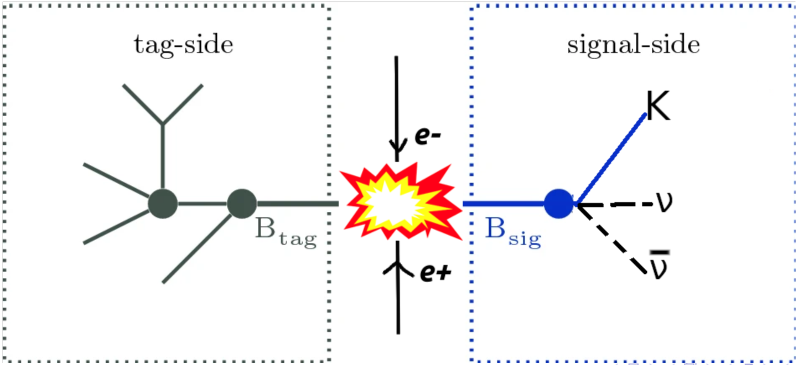
Corentin Santos
with Jacopo Cerasoli and Giulio Dujany

6 November 2023

Table of Contents

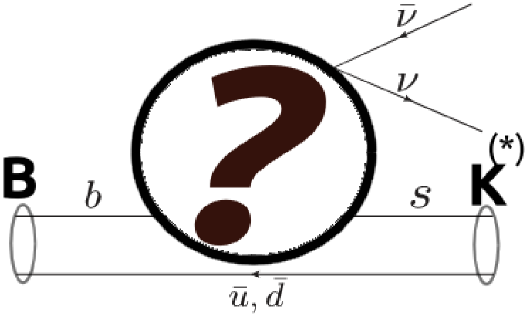
Motivations

- Need to reconstruct the tag side to infer information about the signal side (*e.g.* $B \to K\nu\bar{\nu}$)



Motivations

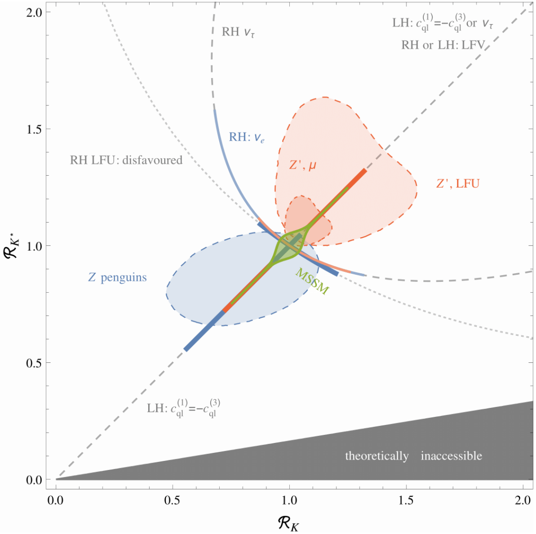
- **New physics** : new physics particles may appear in the loop and modify probability
- One expects a statistically significant discrepancy from the SM
- Possible presence of new physics invisible particles at the place of ν



Schematic representation of new physics in $B \to K^{(*)}\nu\bar{\nu}$

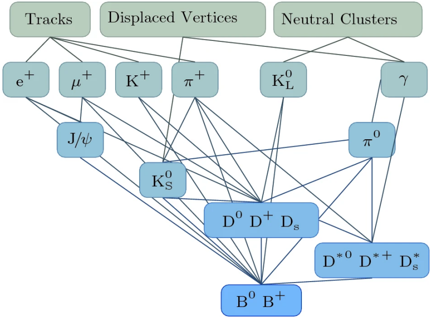
$$\mathcal{R}_{K^{(*)}} = \frac{\mathrm{BR}\left(B \to K^{(*)}\nu\bar{\nu}|\mathrm{exp}\right)}{\mathrm{BR}\left(B \to K^{(*)}\nu\bar{\nu}|\mathrm{SM}\right)}$$

Representation of different NP models as a function of $\mathcal{R}_K$ and $\mathcal{R}_{K^*}$ [1]



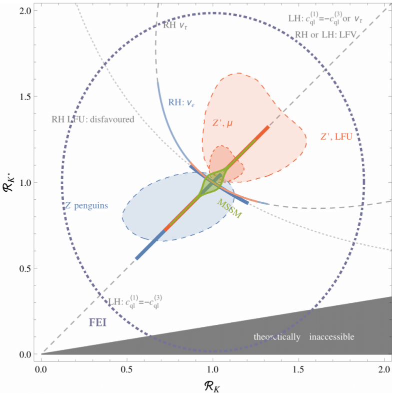
Full Event Interpretation (FEI) algorithm

- Reconstructs the tag side based on a six stages approach using Boosted Decision Trees
- Need to hard-code decay channels $\longrightarrow$ $\approx 15\%$ of B decays considered



Full Event Interpretation (FEI) algorithm

- 15% of B decays hard-coded $\Longleftrightarrow$ few % B reconstruction efficiency
- Sensitive to large $\mathcal{R}_{K^{(*)}}$ only (at $\int \mathcal{L} \approx 360$ fb^{-1})



Introduction to deep learning

- **Input** : $\mathbf{x} \in \mathbb{R}^n, n \in \mathbb{N}^*$
- **Neural network** : does a weighted sum and apply a non-linear function, the *activation function f* : $\mathbb{R}^m \to \mathbb{R}^m$ where $m \in \mathbb{N}^*$ is the number of neurons at the output of the network
- Neural network can be summarized as a funtion $\mathcal{F} : \mathbb{R}^n \to \mathbb{R}^m$ such that $\mathcal{F}(\mathbf{x}) = f(\mathbf{W}\mathbf{x} + \mathbf{b})$, where $\mathbf{W} \in M_{m,n}(\mathbb{R})$ is the weights' matrix and $\mathbf{b} \in \mathbb{R}^m$ is a bias

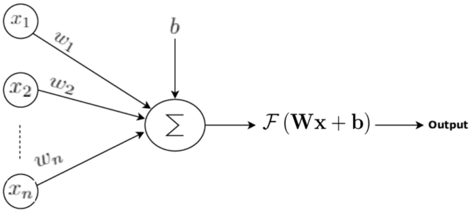


Figure – From [2]

Introduction to deep learning

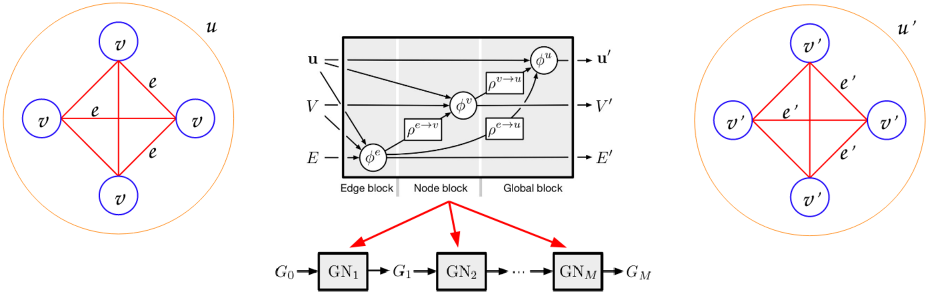
- **Training** : modify the weights in order to minimize the *loss function* $\mathcal{L}$
- **Loss function** : quantify the difference between the results given by the model and the expected ones
- **Training technique** : the *gradient descent* - the weights are modified following the formula :

$$\mathbf{w}' = \mathbf{w} - \frac{\varepsilon}{N} \cdot \sum_{i=1}^{N} \nabla \mathcal{L}_i$$

with $N \in \mathbb{N}^*$ the number of events ; $\varepsilon \in \mathbb{R}_{>0}$ the *learning rate* ; $\mathbf{w}$ and $\mathbf{w}'$ the parameters' vectors

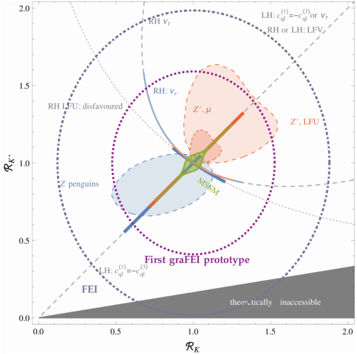
Deep learning algorithm : GRAFEI

- Based on a deep graph neural network (GNN)
- Reconstructs the B_{tag} decay via the Final State Particles
- Trained over generic B decays $\longrightarrow$ No need to hard-code decay channels



Deep learning algorithm : GRAFEI

- Based on a deep graph neural network (GNN)
- Reconstructs the B_{tag} decay via the Final State Particles
- Trained over generic B decays $\longrightarrow$ No need to hard-code decay channels



Overview of the GRAFEI's principles

- Takes as input a fully connected graph representing Final State Particles
- Uses update functions, ϕ, and aggregation functions, ρ
- Returns a fully connected graph with **LCAS** matrix elements as edge features

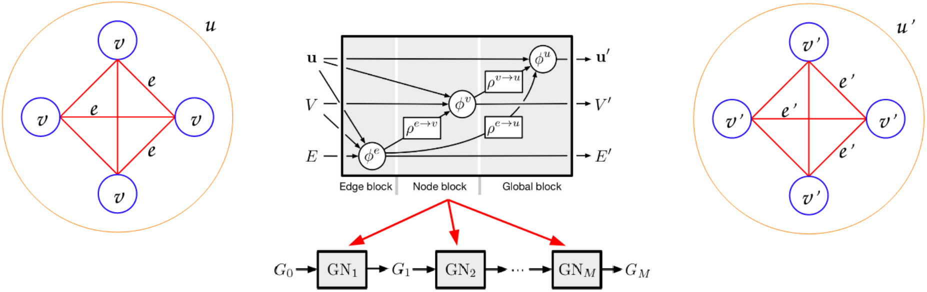
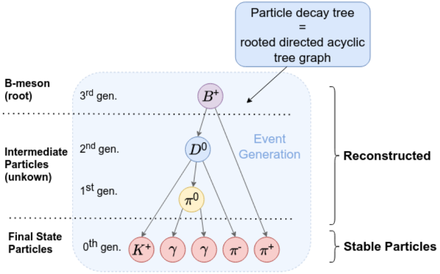


Figure – From [3]

Overview of the GRAFEI's principles

- **LCAS** matrix = representation of a decay tree
- Rows and Columns = Final State Particles
- Elements = **Lowest Common Ancestor**
- Identify them via a **class** between 0 and 5



Adjacency Matrix

	B^+	D^0	π^0	K^+	γ	γ	π^-	π^+
B^+	0	1	0	0	0	0	0	1
D^0	1	0	1	1	0	0	1	0
π^0	0	1	0	0	1	1	0	0
K^+	0	1	0	0	0	0	0	0
γ	0	0	1	0	0	0	0	0
γ	0	0	1	0	0	0	0	0
π^-	0	1	0	0	0	0	0	0
π^+	1	0	0	0	0	0	0	0

$\longleftrightarrow$

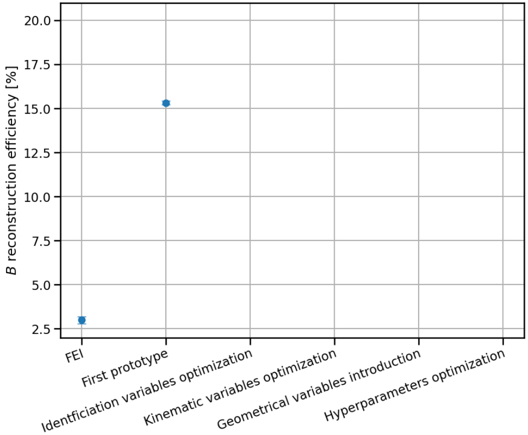
Lowest Common Ancestor (LCA) Matrix

	K^+	γ	γ	π^-	π^+
K^+	K^+	D^0	D^0	D^0	B^+
γ	D^0	γ	π^0	D^0	B^+
γ	D^0	π^0	γ	D^0	B^+
π^-	D^0	D^0	D^0	π^-	B^+
π^+	B^+	B^+	B^+	B^+	π^+

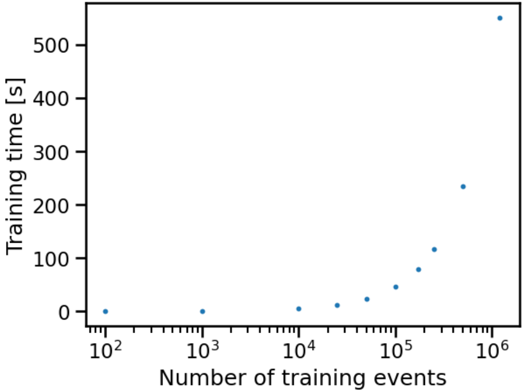
Figure – From [4]

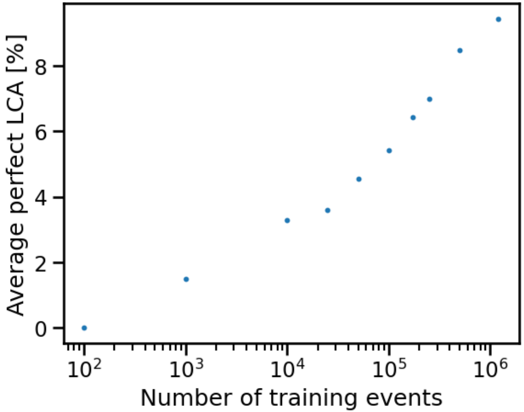
Optimization start

FEI vs GRAFEI prototype : GRAFEI already **5 times for efficient** !



Optimization start



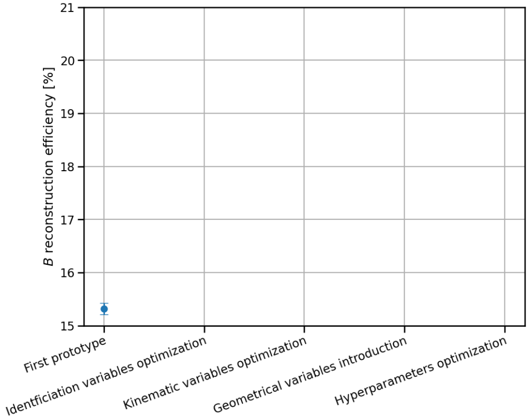


Settled on 50 **training cycles** and 1 000 000 events

Optimization start

Prototype optimizations :

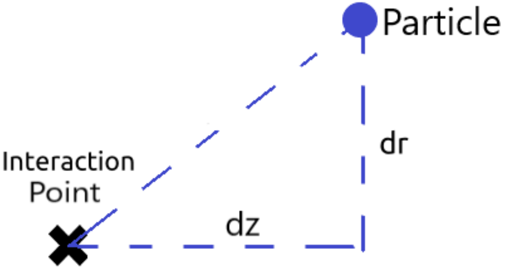
- Input optimization
- Reconstructed particles' list optimization
- Hyperparameters optimization



Optimization start

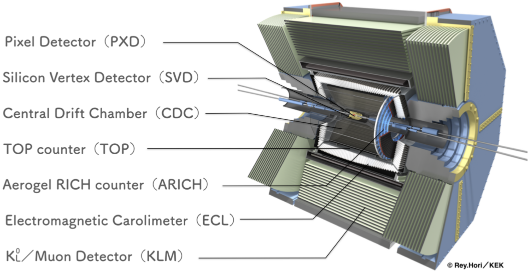
Initial input features :

- Node features :
  - PIDs
  - $p, p_t, (p_x, p_y, p_z)$
  - $\mathrm{d}r$ and $\mathrm{d}z$
  - E and M
  - the charge q
- Edge features :
  - $\cos(\theta)$
- Global features :
  - Number of particles in the event



Node features optimization

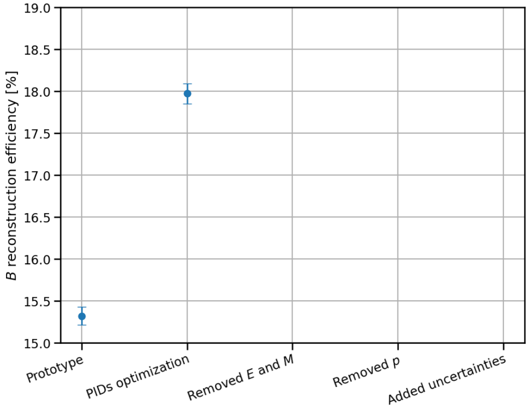
Identification variables' optimization : study and choice of the good sub-detectors for identification



Node features optimization

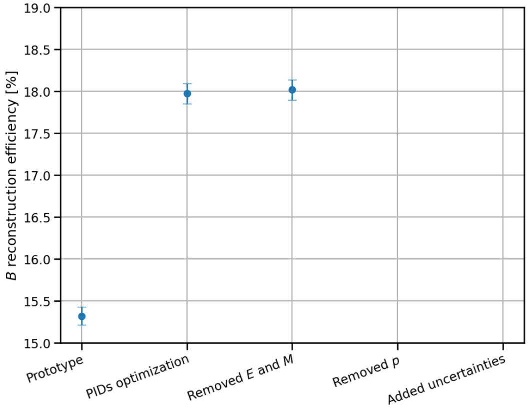
Identification variables' optimization : study and choice of the good sub-detectors for identification

Relative enhancement of $\approx$ 20% with respect to prototype



Node features optimization

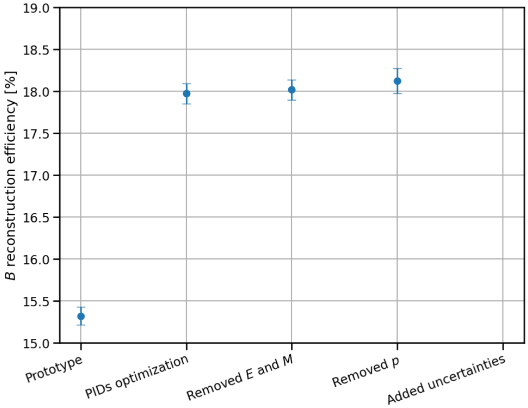
Kinematic variables optimization : study of the mass hypothesis impact on the performances



Node features optimization

Kinematic variables optimization : study of the impact of redundancies between variables (such as p, p_x, p_y and p_z)

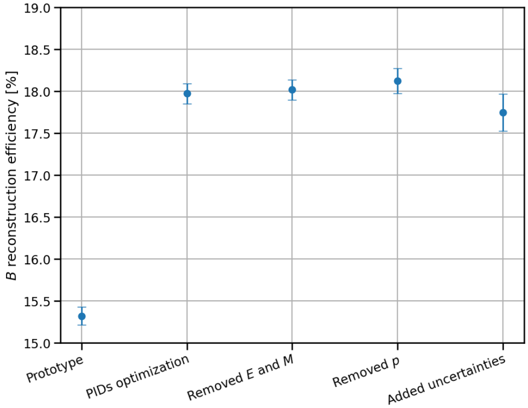
7% decrease of the training time



Node features optimization

Kinematic variables optimization : study of the impact of uncertainties

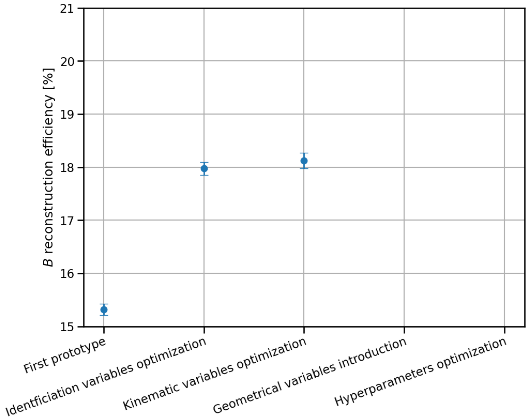
No positive impact on the performances



Node features optimization

Summary of the input features for now :

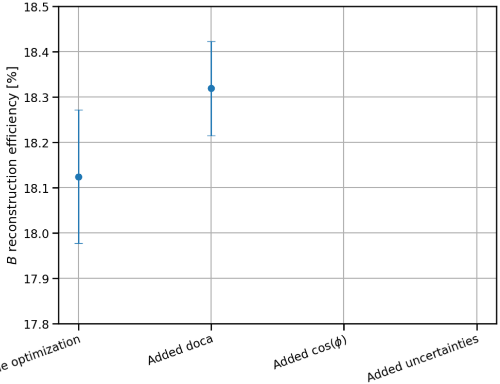
- Node features :
  - PIDs
  - p_t, p_z
  - $\mathrm{d}r$ and $\mathrm{d}z$
  - the charge q
- Edge features :
  - $\cos(\theta)$
- Global features :
  - Number of particles in the event



Edge features optimization

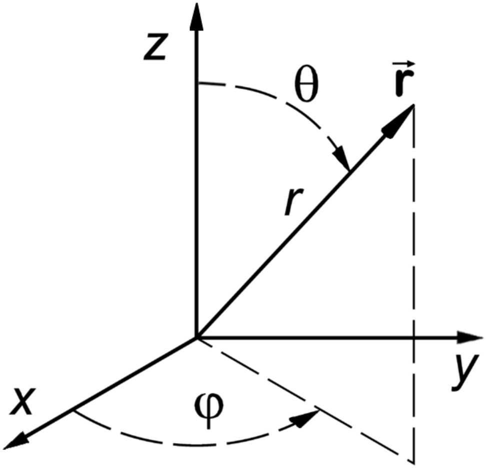
Introduction of geometrical variables : added the Distance Of Closest Approach (doca)

Relative enhancement of $\approx$ 3% with respect to precedent optimization



Edge features optimization

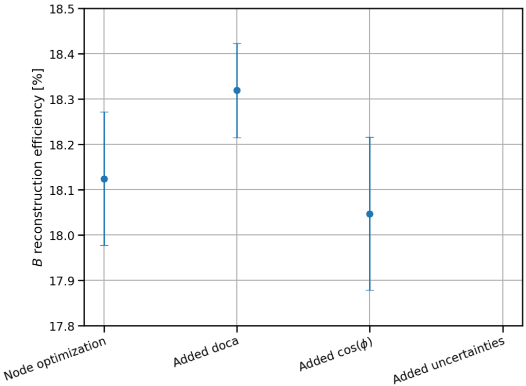
Introduction of geometrical variables : added the cosine of the azimuthal angle ϕ



Edge features optimization

Introduction of geometrical variables : added the cosine of the azimuthal angle ϕ

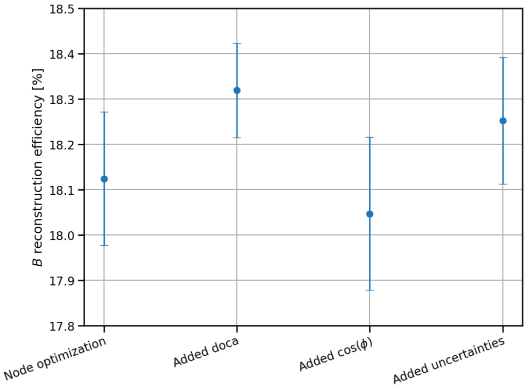
No enhancement with respect to precedent optimization



Edge features optimization

Introduction of geometrical variables : added the uncertainties

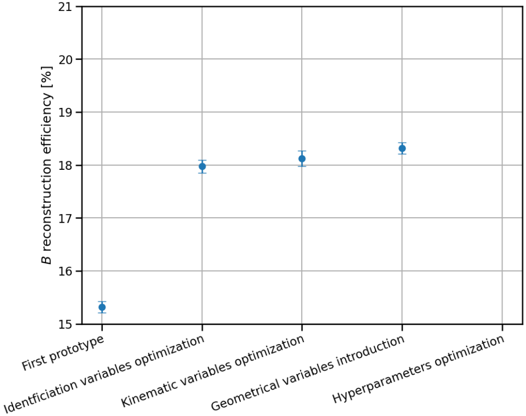
No enhancement with respect to prior the introduction of $\cos(\phi)$



Edge features optimization

Summary of the input features for now :

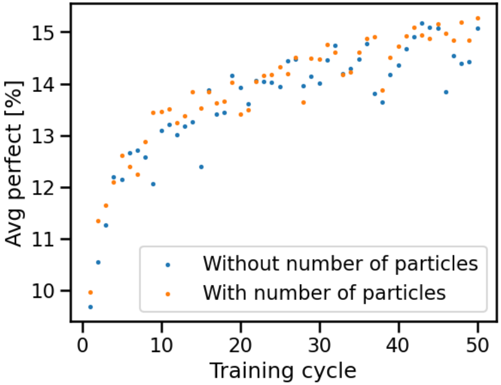
- Node features :
  - PIDs
  - p_t, p_z
  - $\mathrm{d}r$ and $\mathrm{d}z$
  - the charge q
- Edge features :
  - $\cos(\theta)$
  - Distance Of Closest Approach (doca)
- Global features :
  - Number of particles in the event



Global features optimization

Only global feature : number of final state particles

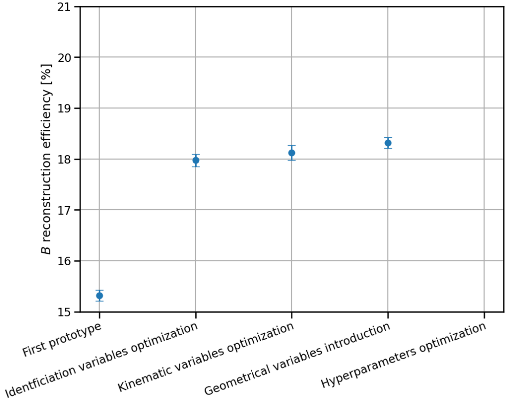
No change between with and without the number of particles



Final list of input features

Final input features :

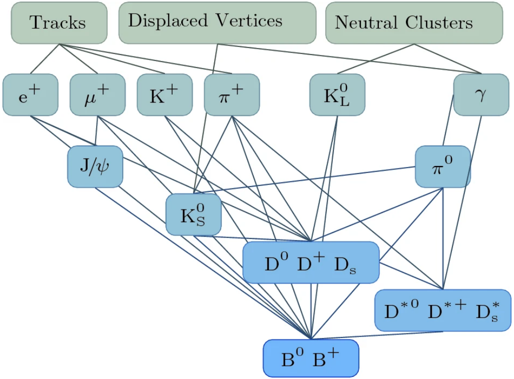
- Node features :
  - PIDs
  - p_t, p_z
  - $\mathrm{d}r$ and $\mathrm{d}z$
  - the charge q
- Edge features :
  - $\cos(\theta)$
  - Distance Of Closest Approach (doca)
- Global features :
  - *None*



Reconstructed particles' list optimization

Current list : B, D^*, D, K_S^0, π^0, J/ψ
$\longrightarrow$ inspired by FEI

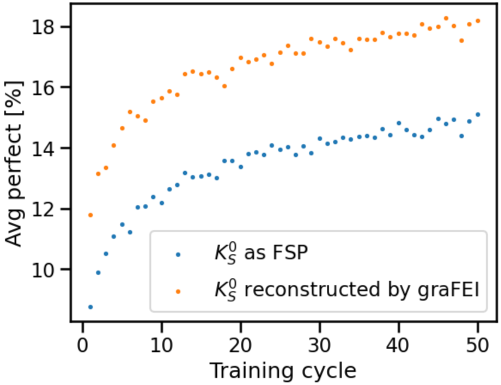
Organized in classes : 5 for B ; 4 for D^* ; 3 for D ; 2 for K_S^0 ; 1 for π^0 and J/ψ ; 0 if not in the decay tree



Reconstructed particles' list optimization

Test : Move K_S^0 from reconstructed to Final State Particle (FSP)

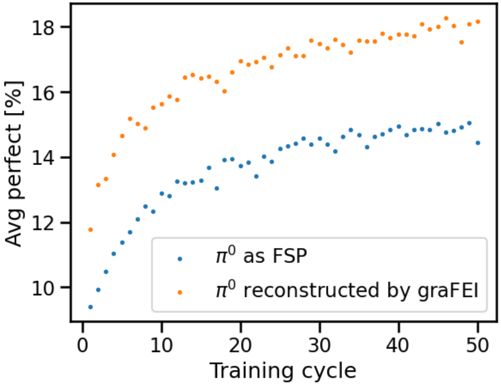
Drastically decreases the performances



Reconstructed particles' list optimization

Test : Move π^0 from reconstructed to Final State Particle (FSP)

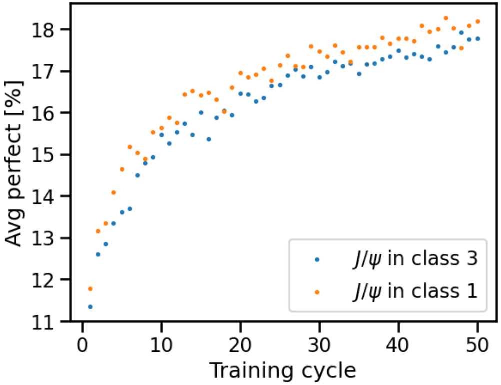
Drastically decreases the performances



Reconstructed particles' list optimization

Test : Move J/ψ from class 1 to class 3

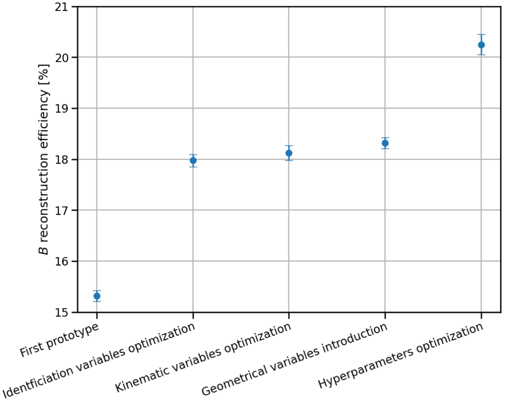
Lower average perfect when J/ψ in class 3 with respect to class 1



Hyperparameters optimization

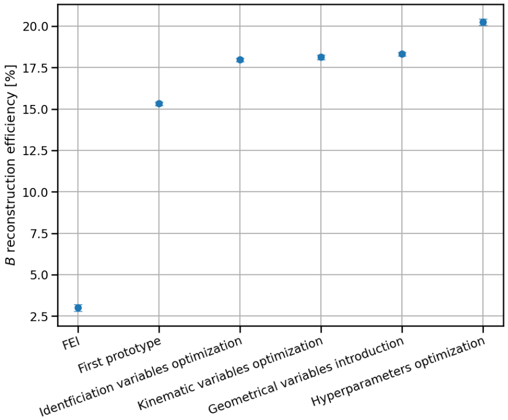
Hyperparameters : define the training and the structure of the network

Relative improvement of 11% with respect to the geometrical variables optimization



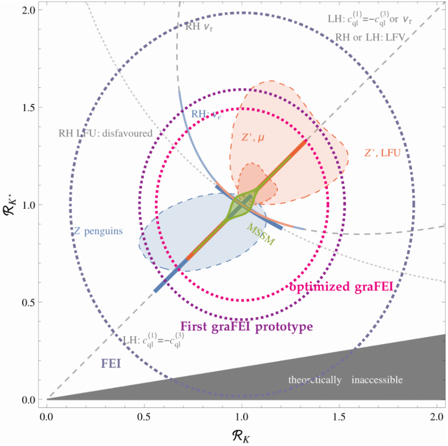
Summary on the results

Finally, **improvement of $\approx$ 33%** with respect to the GRAFEI prototype



Summary on the results

Thanks to graFEI and its optimization : sensitivity to smaller $\mathcal{R}$



Conclusions

Summary :

- Current algorithm, FEI, not efficient enough for this study : development at IPHC of the GRAFEI
- This work brought significant improvements on the performances (about 33% with respect to prototype)
- Physical intuitions are not always right when in the context of deep learning...

Prospects :

- Added new predictions for the model (*e.g.* predict the mass hypothesis)
- Implemented this tool in the Belle II software framework

[1] A. J. Buras, J. Girrbach-Noe, C. Niehoff et D. M. Straub, *$B \to K^{(*)}\nu\bar{\nu}$ decays in the Standard Model and beyond*, 2014, arXiv :1409.4557.

[2] N. Jebreel et al., *Efficient Detection of Byzantine Attacks in Federated Learning Using Last Layer Biases*, in (août 2020), p. 154-165.

[3] P. W. Battaglia et al., *Relational inductive biases, deep learning, and graph networks*, 2018, arXiv :1806.01261.

[4] *Full Event Interpretation using Graph Neural Networks*, https://publish.etp.kit.edu/record/22115.

[5] S. R. Dubey, S. K. Singh et B. B. Chaudhuri, *Activation Functions in Deep Learning : A Comprehensive Survey and Benchmark*, 2022, arXiv :2109.14545.

Activation functions

- Non-linear functions that will activate the neuron based on the "*input strength*"
- Simplest one : Heaviside/step function. For $x \in \mathbb{R}$

$$\mathcal{H}(x) = \begin{cases} 0 \text{ if } x < 0, \\ 1 \text{ otherwise} \end{cases}$$

but has many problems $\longleftarrow$ need new functions

- **Exemples** : sigmoid, tanh, ReLU, ELU...
- For more detailed list, check arXiv :2109.14545 [5]

Branching ratio predictions

Decay channel	$B \to K_S^0 \nu\bar{\nu}$	$B \to K^+ \nu\bar{\nu}$
Branching ratio prediction $\times$ $\mathbf{10^6}$	$2.05 \pm 0.07 \pm 0.12$	$5.06 \pm 0.14 \pm 0.28$

Decay channel	$B \to K^{*0} \nu\bar{\nu}$	$B \to K^{*+} \nu\bar{\nu}$
Branching ratio prediction $\times$ $\mathbf{10^6}$	$9.05 \pm 1.25 \pm 0.55$	$10.86 \pm 1.30 \pm 0.59$

FEI versus GRAFEI

- GRAFEI performs better than FEI $\longrightarrow$ Two times more efficient with same background rejection
- My goal : to increase the efficiency while improving the background rejection