Phys. Lett. B 860 (2025) 139141

Contents lists available at ScienceDirect

Physics Letters B

journal homepage: www.elsevier.com/locate/physletb

Letter

Interpretable machine learning approach for electron antineutrino selection in a large liquid scintillator detector

A. Gavrikov [a,b,*], V. Cerrone [a,b,*], A. Serafini [a,b,*], R. Brugnera [a,b], A. Garfagnini [a,b], M. Grassi [a,b], B. Jelmini [a,b], L. Lastrucci [a,b], S. Aiello [e], G. Andronico [e], V. Antonelli [f], A. Barresi [c], D. Basilico [f], M. Beretta [f], A. Bergnoli [a,b], M. Borghesi [c], A. Brigatti [f], R. Bruno [e], A. Budano [i], B. Caccianiga [f], A. Cammi [d], R. Caruso [e], D. Chiesa [c], C. Clementi [g], S. Dusini [b], A. Fabbri [i], G. Felici [j], F. Ferraro [f], M.G. Giammarchi [f], N. Giudice [e], R.M. Guizzetti [a], N. Guardone [e], C. Landini [f], I. Lippi [b], S. Loffredo [i], L. Loi [d], P. Lombardi [f], C. Lombardo [e], F. Mantovani [h], S.M. Mari [i], A. Martini [j], L. Miramonti [f], M. Montuschi [h], M. Nastasi [c], D. Orestano [i], F. Ortica [g], A. Paoloni [j], E. Percalli [f], F. Petrucci [i], E. Previtali [c], G. Ranucci [f], A.C. Re [f], M. Redchuck [a,b], B. Ricci [h], A. Romani [g], P. Saggese [f], G. Sava [e], C. Sirignano [a,b], M. Sisti [c], L. Stanco [b], E. Stanescu Farilla [i], V. Strati [h], M.D.C. Torri [f], A. Triossi [a,b], C. Tuvè [e], C. Venettacci [i], G. Verde [e], L. Votano [j]

[a] Università degli Studi di Padova, Dipartimento di Fisica e Astronomia, Italy
[b] INFN, Sezione di Padova, Italy
[c] INFN, Sezione di Milano Bicocca e Dipartimento di Fisica, Università di Milano - Bicocca, Italy
[d] INFN, Sezione di Milano Bicocca e Dipartimento di Energia, Politecnico di Milano, Italy
[e] INFN, Sezione di Catania e Università di Catania, Dipartimento di Fisica e Astronomia, Italy
[f] INFN, Sezione di Milano e Università degli Studi di Milano, Dipartimento di Fisica, Italy
[g] INFN, Sezione di Perugia e Università degli Studi di Perugia, Dipartimento di Chimica, Biologia e Biotecnologie, Italy
[h] INFN, Sezione di Ferrara, Università degli Studi di Ferrara, Dipartimento di Fisica e Scienze della Terra, Italy
[i] INFN, Sezione di Roma Tre e Università degli Studi di Roma Tre, Dipartimento di Matematica e Fisica, Italy
[j] INFN, Laboratori Nazionali dell'INFN di Frascati, Italy

ARTICLE INFO

Editor: Maurizio Pierini

Keywords:
Interpretability
Machine learning
Event selection
Neutrino physics

ABSTRACT

Several neutrino detectors, KamLAND, Daya Bay, Double Chooz, RENO, and the forthcoming large-scale JUNO, rely on liquid scintillator to detect reactor antineutrino interactions. In this context, inverse beta decay represents the golden channel for antineutrino detection, providing a pair of correlated events, thus a strong experimental signature to distinguish the signal from a variety of backgrounds. However, given the low cross-section of antineutrino interactions, the development of a powerful event selection algorithm becomes imperative to achieve effective discrimination between signal and backgrounds. In this study, we introduce a machine learning (ML) model to achieve this goal: a fully connected neural network as a powerful signal-background discriminator for a large liquid scintillator detector. We demonstrate, using the JUNO detector as an example, that, despite the already high efficiency of a cut-based approach, the presented ML model can further improve the overall event selection efficiency. Moreover, it allows for the retention of signal events at the detector edges that would otherwise be rejected because of the overwhelming amount of background events in that region. We also present the first interpretable analysis of the ML approach for event selection in reactor neutrino experiments. This method provides insights into the decision-making process of the model and offers valuable information for improving and updating traditional event selection approaches.

* Corresponding authors.
E-mail addresses: arsenii.gavrikov@pd.infn.it (A. Gavrikov), vanessa.cerrone@pd.infn.it (V. Cerrone), andrea.serafini@pd.infn.it (A. Serafini).

https://doi.org/10.1016/j.physletb.2024.139141
Received 20 June 2024; Received in revised form 25 October 2024; Accepted 15 November 2024
Available online 19 November 2024
0370-2693/© 2024 The Author(s). Published by Elsevier B.V. Funded by SCOAP³. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

1. Introduction

Over the past decades, particle physics has experienced a paradigm shift in the data analysis approach, integrating well-established traditional methods with advanced machine learning (ML) techniques [1,2]. ML tools have found extensive applications in different fields of particle physics, including neutrino physics, offering solutions to a wide range of challenges, e.g., particle identification [3], background rejection [4], vertex and energy reconstruction [5,6], fast event generation [7], end-to-end detector optimization [8]. In neutrino physics, examples also include the use of deep convolutional neural networks for selecting inclusive charged-current interactions in MicroBooNE [9] and the adoption of boosted decision trees in Super-Kamiokande to improve multi-site tagging of electron neutrino-like events [10]. Although these examples fall within the same scope as our study, it is difficult to make a quantitative comparison due to differences in the underlying principles of neutrino detection and/or the involved neutrino sources. However, a recent study on the application of machine learning methods for background rejection in KamLAND geo-neutrino analysis has similar experimental conditions and reports a significant improvement in the signal-to-background ratio [11].

A common concern with the widespread adoption of ML methods is their perceived *black-box* nature. Many of these advanced algorithms lack transparency, making it difficult for researchers to understand the underlying mechanisms driving their predictions. This lack of interpretability can be a significant barrier, especially in scientific domains where a deep understanding of the processes involved is crucial. In this study, we address this question by focusing on the development of interpretable and explainable ML methods [12–14] for the specific task of reactor antineutrino event selection in a liquid scintillator detector.

Reactor antineutrinos have held a crucial role in the neutrino physics landscape since their very first detection [15]. The most common channel for their detection is the Inverse Beta Decay (IBD) reaction, where an electron antineutrino interacts with a proton and produces a positron and a neutron, primarily due to its substantial cross section with respect to other processes in the MeV energy range [16]. Many of the modern reactor experiments, such as KamLAND [17], Daya Bay [18], Double Chooz [19], and RENO [20], have adopted the Liquid Scintillator (LS) technology. This approach is based on using organic Hydrogen-rich materials that serve as both the proton target for the antineutrinos and the medium for detecting the outgoing positron. The resulting neutron can be captured by either isotopes present in the scintillator, such as Hydrogen itself or other elements like Carbon or Nitrogen, or by specifically loaded targets like gadolinium [18–20]. We focus on the Jiangmen Underground Neutrino Observatory (JUNO) [21,22], a multi-purpose and new generation LS experiment currently under construction in South China, largely exceeding its predecessors in size and expected performances. Its Central Detector (CD) is composed of a 20 kton liquid scintillator target housed within a 17.7-meter-radius spherical acrylic vessel and immersed in a 35 kton ultra-pure water pool. The CD is equipped with an advanced photo-detection system comprising 17612 20-inch photomultiplier tubes (PMTs) and 25600 3-inch PMTs, attached to a surrounding Stainless Steel (SS) structure. This configuration provides an extensive total photo-coverage of ∼78%, granting a photoelectron (PE) statistics of ∼1600 PEs at 1 MeV [23].

Due to the extremely small cross sections of neutrino weak interactions, neutrino events are inherently rare. For this reason, intensive efforts are dedicated to mitigating the backgrounds [24]. Efficient control of radiogenic contamination is achieved through meticulous detector design, careful selection of the employed materials, and strict radiopurity standards for the LS formula. Despite its underground location (1800 m.w.e.) and the expected high level of LS purification, the substantial size of the JUNO detector results in more significant background contamination than what is typically observed in smaller-scale experiments. As a result, performing an efficient event selection is of utmost importance in JUNO.

This paper is organized as follows: in Section 2, the IBD reaction mechanism and a cut-based benchmark selection strategy are presented, underlying the problem addressed with machine learning. Section 3 introduces the ML models used in the study and discusses goals of the interpretability analysis of an ML model. In Section 4, the data samples employed in the study are described. In Section 5, we discuss the machine learning approach in details, we describe how the ML models were trained and how their hyperparameters were optimized. The performance of the presented ML models is discussed in Section 6 and compared to the benchmark approach.[1] Section 7 is dedicated to the model's interpretability. Section 8 discusses model calibration and uncertainty estimation of the model. Finally, we conclude and summarize the work in Section 9.

2. Electron antineutrino detection and benchmark IBD selection

The selection rationale is driven by the characteristic pattern yielded by the IBD reaction, where an antineutrino $\overline{\nu}_e$ interacts with a proton in the Hydrogen-rich target medium, producing a positron and a neutron in the final state. The positron quickly deposits its kinetic energy through ionization and annihilates with an electron into two 511 keV photons, resulting in a *prompt* signal. Meanwhile, the neutron thermalizes in the detector and, after an average time of 220 μs, undergoes capture on either Hydrogen or Carbon present in the LS. The subsequent emission of a 2.22 MeV(∼99% of cases) or 4.95 MeV (∼1% of cases) gammas, respectively, generates a *delayed* signal. It is worth mentioning that neutrons can be captured on other isotopes, like ^{13}C, ^{14}N, forming a delayed signal at higher energies in approximately 0.01% of cases.

This double signature represents a powerful means to discriminate signal from backgrounds. The latter can be divided into two main groups. A correlated background consists of a pair of events induced by a single physics process, mimicking the prompt-delayed pattern induced by reactor antineutrino interactions (e.g., geoneutrinos and long-lived cosmogenic isotopes, ^{9}Li and ^{8}He, fall within this class [22]). On the other hand, uncorrelated backgrounds, often referred to as *accidental coincidences*, arise when two independent signals are detected within a short time window, even though they are not associated with the same interaction (i.e., they mimic the typical time signature of signal events). These coincidences are primarily attributed to radioactive contamination in the detector materials and surroundings. Some correlated backgrounds (e.g. geoneutrinos) are irreducible, others can be reduced through ad-hoc cuts (e.g. muon cuts for cosmogenic backgrounds), but their residual contamination is considered irreducible. Therefore the main task required of a selection algorithm is to distinguish between two classes: reactor antineutrino events and accidental coincidences.

As a benchmark for our ML model, we adapted the cut-based selection strategy from [25], which is based on a combination of cuts on the following six variables (or *features*): E_{prompt}, E_{delayed}, R_{prompt}, R_{delayed}, Δt, and ΔR. The quantities E_{prompt}, E_{delayed} represent the reconstructed energies, R_{prompt} and R_{delayed} are the radial components of the reconstructed vertices. Finally, Δt is the time interval between prompt and delayed signals, and ΔR is the Euclidean distance between the two vertices.

All the aforementioned variables are obtained through a stand-alone Monte Carlo (MC) simulation based on the most up-to-date data published by the JUNO collaboration. Specifically, the generated data includes the simulation of the detector's geometry and the particles' interactions inside the target material, e.g., physics processes such as light production and energy leakage, in order to produce data-like samples. Energies and vertices of the events are smeared with respect to their true values, following Ref. [25]. During the real data taking and analysis of

[1] Throughout this discussion, "benchmark" selection refers to the cut-based selection used in [25], which however does not necessarily reproduce JUNO official selection.

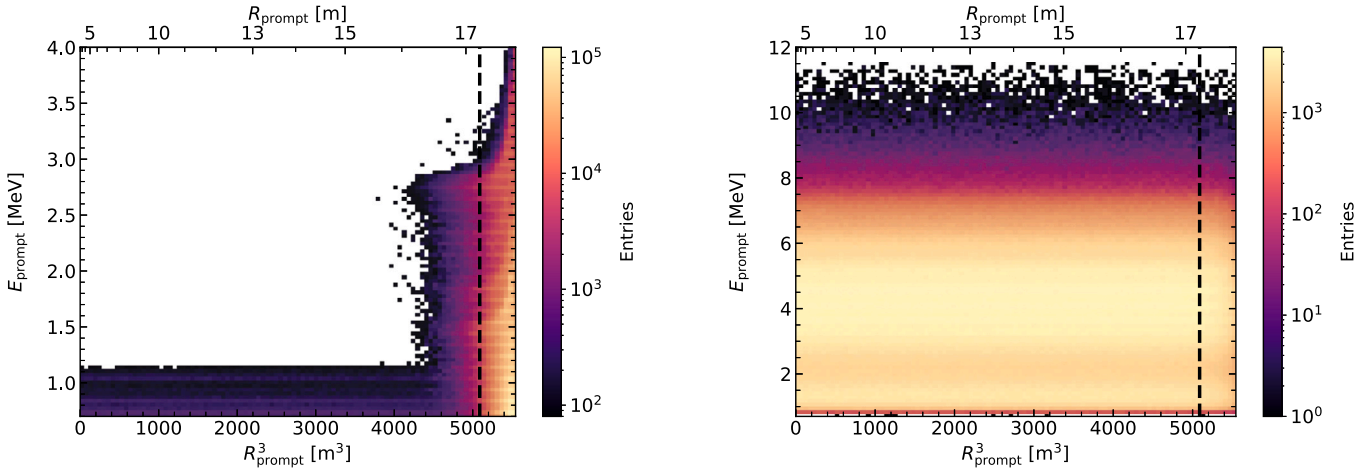


Fig. 1. Reconstructed energy as a function of volume (R^3) for radioactivity (left) and IBD events (right). The IBD prompt energy spectrum extends up to approximately 12 MeV, while radiogenic events dominate the low energy range. The FV cut is indicated by the dashed line. The secondary axis provides the linear scale.

the experiment, variables will be provided either by JUNO reconstruction algorithms or by the official JUNO simulation software [26] tuned on data. The cuts used for the benchmark approach are the following:

- **Fiducial volume (FV) cut:** prompt or delayed candidates are discarded if their vertices are reconstructed more than 17.2 m away from the detector center. This cut is implemented to mitigate the impact of the exponentially increasing radioactive background rate at the edges of the target volume.
- **Energy cut:** the energy windows are set as $E_{\text{prompt}} \in (0.7, 12.0)$ MeV for prompt events and $E_{\text{delayed}} \in (1.9, 2.5) \cup (4.4, 5.5)$ MeV for delayed events.
- **Time cut:** the surviving pairs are required to fall in a time coincidence window of 1 ms, corresponding to approximately 5× the neutron capture time.
- **Vertex cut:** the distance between prompt and delayed events vertices has to be smaller than 1.5 m, hence $\Delta R < 1.5$ m.

Moreover, during data taking, an additional muon veto cut will be used, according to the topology of track-like events, resulting in a reduction of fiducial volume for a given time interval. The current state-of-the-art muon veto strategy [25] yields a selection efficiency of 91.6% for IBD events. We will not consider this criterion in our discussion, but it can be applied at a second stage as a multiplying factor for both ML and cut-based selection approaches.

Hereinafter we define *efficiency* as the ratio between the amount of correctly classified IBD events and the total amount of IBDs in the dataset:

$$\text{efficiency} = \frac{N_{\text{IBD}}^{\text{tagged as IBD}}}{N_{\text{IBD}}} \tag{1}$$

This quantity corresponds, in the field of ML, to the *recall* metric, measuring a classification model's ability to correctly identify the positive class. On the other hand, *purity* is associated with the residual background contamination and is defined as the ratio of correctly identified signal pairs to the total number of tagged as IBD events, represented by the equation:

$$\text{purity} = \frac{N_{\text{IBD}}^{\text{tagged as IBD}}}{N^{\text{tagged as IBD}}} \tag{2}$$

The purity of a given sample is analogous to the ML *precision*, gauging how accurately a machine learning model predicts the positive class. The selection efficiency for each cut is determined by calculating the ratio of the number of events meeting the specific criterion to the total number of reconstructed events before the application of that particular cut. This step-wise application of cuts is viable for IBD events due to the almost uncorrelated nature of all features. In contrast, accidental coincidences require a simultaneous application of all selection criteria to capture the intrinsic (and significant) dependence among features. An example is reported in Fig. 1. (left for radiogenic events, right for IBD prompt candidates), where the event rate is shown as a function of R^3 and reconstructed energy. IBD events are uniformly distributed inside the CD since antineutrinos are homogeneously interacting in the detector. Contrariwise, a strong correlation exists between energy and radial distributions for radioactivity events. As a consequence, selection efficiency terms cannot be computed independently and then progressively combined. In the subsequent sections, we will outline how this challenge can be effectively tackled using ML techniques.

The FV cut results in a sensitive loss of statistics for IBD events, of the order of 8%, as it can be determined by strictly geometrical considerations. Thus, one of the goals of using a ML algorithm is to create a more flexible demarcation between the two classes of events. This flexibility would allow us to retain a greater number of signal events, maintaining the same, or potentially improved, purity level. Furthermore, we observe that the energy cut is effective in rejecting background events and preserves nearly all IBD candidates, while still offering limited room for improvement. Furthermore, the conventional "box-like" cut applied to the ΔR and Δt features ($\Delta R < 1.5$ m, $\Delta t < 1$ ms) is suboptimal, and analytical optimization becomes challenging when dealing with multi-dimensional PDFs. The single impact of the cuts on efficiency is discussed in detail in Section 6.1.

In addition, the accidental coincidences (R^3, energy) bi-dimensional distribution in Fig. 1 (left panel) suggests that cuts that depend on the vertices distance ΔR and energy can potentially help in distinguishing between the two classes. Indeed, IBD candidates are to a large extent uniformly distributed while background events show a non-trivial radial distribution at different energies. These specific examples underscore the potential of explainable ML techniques to (1) identify optimal cuts and (2) offer valuable insights into the relationships between features, empowering the analyzers with the capability to make informed decisions.

3. Problem statement

As explained in the preceding section, the benchmark selection strategy is based on the use of relatively basic cuts, which, while effective, fail to address the inherent correlations in high-dimensional data. On the other hand, machine learning methods are proven to be powerful tools to process high-dimensional data and to find underlying non-linear dependencies within it. In this study, we use a fully connected neural

network (FCNN) and Boosted Decision Trees (BDT) [27,28] as classifiers to distinguish between signal (reactor antineutrino) and background (random coincidence) events. BDT is a well-established approach for handling low-dimensional, high-level, tabular data [29] and we use it as a baseline ML model to serve as comparison with the neural network. One limitation of tree-based methods is that, due to the nature of the algorithms, the decision boundaries between classes are represented by non-smooth, step-like functions. While BDT can still be highly effective in separating signal from background, this characteristic makes it challenging to directly apply the learned decision profiles as optimized cuts for a cut-based selection approach. In contrast, FCNN naturally provides smooth and differentiable boundaries between the classes, making neural networks a more desirable and general approach for achieving the objectives of this study. The classifiers use as input the following ten features, complementing the cut-based selection set with angular information: E_{prompt}, E_{delayed}, R^3_{prompt}, R^3_{delayed}, $\cos(\theta_{\text{prompt}})$, $\cos(\theta_{\text{delayed}})$, φ_{prompt}, φ_{delayed}, ΔR, and Δt. Here, φ_{prompt} and φ_{delayed} are the azimuthal angles, θ_{prompt} and θ_{delayed} the zenith angles with respect to the vertical z axis. The choice of this particular set of features is driven by both the geometrical structure of the detector and the unique patterns observed in signal events. In brief, the variables R^3, φ, and $\cos\theta$ exhibit a uniform distribution for IBD candidates, whereas a more complex trend is expected for radiogenic background events. A detailed explanation of this particular aspect will be provided in Section 4. We acknowledge that using the prompt energy as an input feature for the model causes energy-dependent efficiency and purity estimations and this has to be properly taken into account at the level of subsequent analyses.

A central part of the study is a comprehensive analysis of an ML model's interpretability. This analysis has several goals:

1. Ensure trust in the model and its transparency by a deep understanding of the dependencies between features.
2. Achieve an understanding of decision boundaries between different classes, that can be provided by interpretable ML models in both visual and quantitative ways.
3. Optimize and fine-tune the cut-based selection criteria. An estimation of the importance of each feature within the selection task both at the level of the entire dataset and at the level of each individual event can potentially help in improving the efficiency of traditional event classification. This can be especially important during the initial phase of data-taking since event generators and reconstruction algorithms may not perform optimally. As a result, one can expect noticeable discrepancies between Monte Carlo simulations and real data. Training an ML model on inaccurate simulations could lead to unreliable performances, as the robustness of the algorithm to this discrepancy remains uncertain.

In the JUNO experiment, the tuning of detector response in the Monte Carlo simulation and the validation of reconstruction algorithms is planned to be performed using calibration data [30,31]. Once this tuning is completed, ML models can be retrained to achieve more reliable performance.

4. Data description

In this study, a standalone MC is used to generate data-like samples to test our selection strategy. In particular, two different datasets were prepared:

1. **IBD dataset**: it consists of 15M independent IBD pairs uniformly distributed in the full CD volume with radii up to 17.7 m. The energy distribution follows the expected oscillated spectrum of reactor electron antineutrinos. It is worth mentioning that the two mass ordering assumptions are equivalent for our purpose, hence we chose the normal ordering current global best fit values to build out our dataset [32].
2. **Accidentals dataset**: it consists of 15M pairs of different radioactive decays of all types, namely α, β, γ [24]. To prevent biases and ensure the model's generalization capability, the same amount of events is chosen to balance the IBD dataset. This number of accidental events corresponds to approximately 50 days of data collection. The radioactivity of the materials used in the construction of the detector represents one of the main sources of accidental background (see Table 1). These radioactive contaminants release energy through their decay processes, and they are categorized as *internal* if they are produced in the LS, or *external* [24] if they arise from other components of the detector, respectively. *Internal* radioactive events are uniformly distributed in the full CD volume with radii up to 17.7 m. *External* radioactive events are instead generated at the detector edges and radially decrease following an exponential distribution going towards the detector center, because of their interaction with the LS itself [24]. While the *internal* radioactivity is simulated as latitudinally and longitudinally homogeneous, the *external* contribution is simulated with an angular modulation due to the grid structure of the detector components. Both components contribute to the resulting accidentals dataset's features.

Table 1
Main radioactive contaminants and corresponding sources [24]: 1) LS, 2) acrylic sphere, 3) stainless steel (SS) structure, 4) PMTs glass, 5) water pool.

Location	Isotopes
LS	^{238}U, ^{232}Th, ^{40}K, ^{210}Pb, ^{14}C, ^{85}Kr
Acrylic	^{238}U, ^{232}Th, ^{40}K
SS	^{238}U, ^{232}Th, ^{40}K, ^{60}Co
Glass	^{238}U, ^{232}Th, ^{40}K, ^{208}Tl
Water pool	^{222}Rn

4.1. Data preparation

To build the feature table for both datasets, we iterate over all events in the sample, considering them as prompt candidates. Then, for each i-th prompt event, we select all j-th events (with $i < j$) occurring in a time window of 10 τ, where $\tau \simeq 220\,\mu$s is the mean neutron capture time. This particular choice is aimed at minimizing event loss, ensuring that the fraction of potential candidates to be excluded is less than 5×10^{-5}. Finally, we compute and store the relevant features for all possible (i, j) combinations within the specified time interval.[2] Afterward, the two feature tables are merged, assigning the corresponding class (IBD or accidental) to each event. Fig. 2 shows the distributions of the 10 features after all steps described above, for both IBD events (red) and accidental coincidences (blue).

Hereinafter, we analyze the feature distributions in detail:

- Accidental coincidences are not uniformly distributed within the LS and exhibit an exponential increase towards the edges of the detector, as expected. At the same time, it is worth noting that their radial coordinate (i.e., R^3_{prompt} and R^3_{delayed} in blue in the first two panels of Fig. 2) is approximately uniform in the target volume up

[2] The average number of combinations per event is approximately 0.5 for IBDs and approximately 0.1 for accidentals. We would like to emphasize that the 0.5 value for IBDs arises because, for each prompt event, a delayed event can always be found within 10τ, but the delayed event does not have a subsequent event to form a pair within the window, taking into account the very low event rate of IBD.

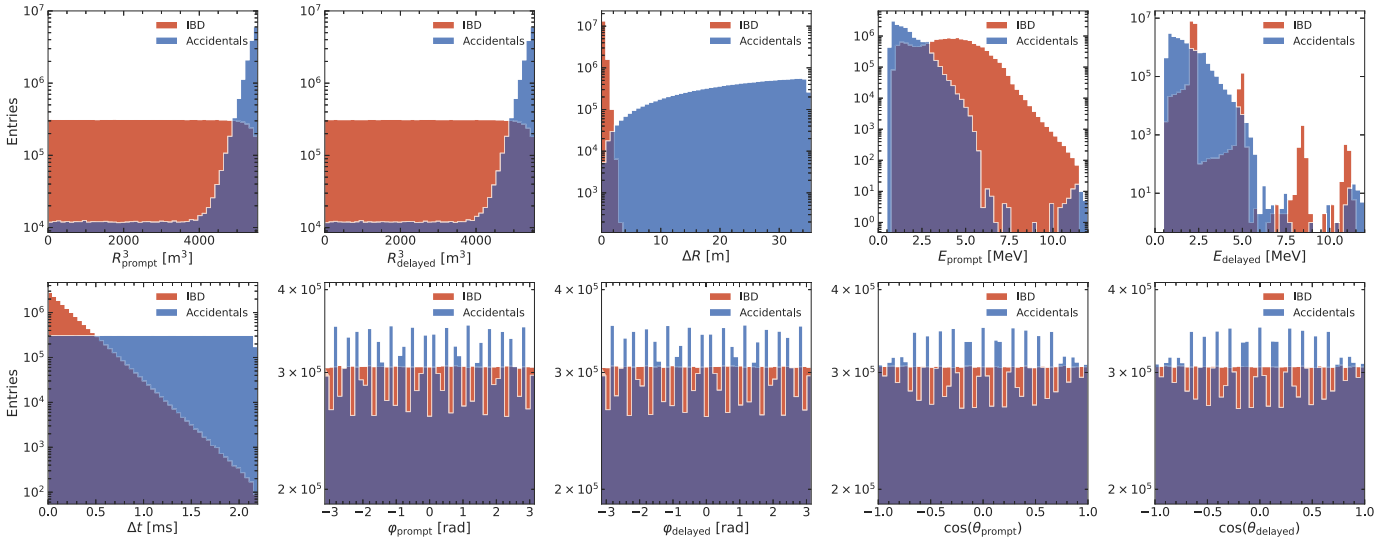


Fig. 2. Distributions of features for the dataset used to train and evaluate the ML model for both IBD (red) and accidental (blue) events.

to about 16 m ($\simeq 4000\,\text{m}^3$). IBD events are instead uniformly distributed inside the CD, as previously mentioned.

- The Euclidean distance ΔR between IBD prompt-delayed candidates' vertices is peaked at approximately ~0.2 meters and the distribution depends on the random walk process of the emitted neutrons. As for random coincidences, the ΔR distribution is shaped by the spatial distribution of radiogenic events within the CD.
- The energy distribution corresponds to the positron spectrum, and to the gammas emitted by neutron capture, for prompt and delayed IBD candidates, respectively. On the other hand, the radioactive decays of the primary contaminants determine the energy spectrum shape for accidental coincidences [24], which is the same for both prompt and delayed events, except for statistical fluctuations. This spectrum has a prominent peak at energies ~1 MeV, where the major contribution comes from ^{14}C and quenched α peaks, mainly from the ^{238}U/^{210}Pb chains [24], and it extends up to 5 MeV, at the end point of ^{208}Tl β decay.
- The Δt distribution for IBD events is an exponential decay with characteristic time related to neutron capture, while it is almost flat for accidentals. Specifically, the expected distribution is exponential with a long half-life determined by the event rate.
- IBD events exhibit a spherical symmetry, resulting in a uniform distribution for φ and $\cos\theta$. In contrast, radioactivity presents a distinctive non-uniformity due to contamination from the detector supporting structure, which are localized at fixed positions in φ and $\cos\theta$. This deviation from uniformity is noticeable at the edges of the detector: this effect can be seen in Fig. 2.

5. Machine learning approach

In the context of machine learning, our goal — selection of IBD events among accidental background — is a supervised classification problem. In supervised learning problems, a model considers input-target pairs and learns the mapping from input features to a target value (or so-called *label*). This learning process is based on using data samples with known input-target pairs. Depending on the type of target, one can define two types of supervised problems: classification problem (the target represents a discrete set of values) or regression problem (the target represents continuous values). More formally, let us have a set of pairs: $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_n, y_n) = \{\mathbf{x}_i, y_i\}_{i=1,...,n}$, where $\mathbf{x}_i \in \mathbb{R}^p$, $y_i \in \{0, 1\}$, p is the number of input features, and n is the amount of events in a data sample. The mapping from $\mathbf{x}$ to y is then defined by a function $h : y = h(\mathbf{x})$. Our classification task is then to find a model $f : \mathbb{R}^p \longrightarrow \mathbb{R}$ which is a function of both $\mathbf{x}$ and parameters ϕ. The set of parameters ϕ specify a relationship between an input and an output of a model. The discrepancy between the output and the target, between f and h, is quantified with a function called *loss* L. Training a model means to find a set of ϕ which make the model f best approximate the function h, so minimizes the loss function: $\hat{\phi} = \underset{\phi}{\text{argmin}}[L(\phi)]$.

The set of pairs $\{\mathbf{x}_i, y_i\}_{i=1,...,n}$ is called *training* dataset, i.e. the one used to directly train a model. Usually, to perform proper training and model evaluation procedures, one needs two additional datasets: a *validation* dataset and a *testing* dataset. The former is used to optimize the hyperparameters [3] of a model and to evaluate the performance *during* the training process. Conversely, the latter is used to test the performance of a model once it is *trained*. For our task, the 30M dataset is split into three parts with the following ratios: 20M events for training, 5M events for validation, and 5M events for testing. This choice is made to have enough data for training, tuning hyperparameters, and evaluating the final model's performance, while ensuring that each subset is representative of the overall dataset.

In this study, FCNN and BDT are used as a model f.

5.1. Boosted decision trees

BDT is a gradient boosting-based algorithm that combines multiple weak decision trees into a single strong model. The ensemble of BDT trees is built sequentially: each new tree is designed to correct the predictions made by the ensemble in the previous step. To avoid overfitting, trees usually have a relatively small maximum depth of less than 5. The boosting process reduces both bias and variance, improving overall prediction performance. As mentioned in Section 3, BDT is a powerful approach for low-dimensional, high-level, tabular input data and we use it as a baseline model for comparison with the FCNN.

Additionally, one of the advantages of BDT is its ability to provide an independent way to access global feature importance through the algorithm's construction. One method for this is given by the gain feature importance [33], which is calculated as the average reduction in the loss function when an input feature is used to split a node in a tree. We

[3] Hyperparameters — parameters of a model that define its structure and its learning process. Hyperparameters are set before training is started and cannot be adjusted during the training, unlike learnable parameters (e.g. weights in a neural network). An example of hyperparameters of a neural network could be the following parameters: number of layers, number of units in a layer, learning rate, etc.

Table 2
Hyperparameter search space for FCNN. Selected hyperparameters are highlighted in bold.

Hyperparameter	Search space and **selected hyperparameter**
Units in input layer	[16, 256]: **96**
Units in hidden layers	[16, 256]: **240**
Number of hidden layers	[1, 10]: **2**
Activation [35–38]	**ReLU**, Leaky ReLU, SiLU, PReLU, Tanh
Optimizer [39–41]	**Adam**, SGD, RMSprop
Learning rate	$[10^{-5}, 10^{-1}]$: $\mathbf{3.5 \cdot 10^{-4}}$
Scheduler type [42,43]	Exponential, ReduceOnPlateau, **CosineAnnealing**, None
Layer weights initialization [44,45]	xavier uniform, xavier normal, **orthogonal**, normal, uniform
Batch normalization [46]	True, **False**
Batch size	[128, 2048]: **1024**

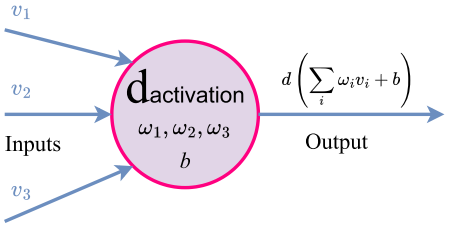


Fig. 3. The schematic view of a neuron — the basic component of a neural network.

use this metric as an additional cross-check for the feature importances computed for the FCNN model.

In this paper, we use the XGBoost [33] implementation of BDT as a robust and widely accepted framework for adopting the algorithm. To optimize the hyperparameters, we use the Random Sampler from the Optuna library [34]. The total number of trees is determined through the early stopping procedure with a *patience* parameter of 20, which indicates the maximum number of trees that can be added to the ensemble without any improvement on the validation dataset before the training process is stopped. The binary cross-entropy loss is used as the loss function. The optimized BDT model, with a maximum depth of **2**, a learning rate of **0.25**, and a total of **483** trees, is used further in the study.

5.2. Fully connected deep neural network

Fig. 3 shows the basic component of an FCNN, i.e., a *neuron* (or a *unit*). Neurons are connected with other neurons and the strength of their connection is defined by weights ω_i. These weights are adjusted during the training process to minimize the difference between the predicted and true outputs. Each neuron computes a weighted sum of inputs and then applies an *activation function*:

$$g(\mathbf{v}) = d\left(\sum_i \omega_i v_i + b\right),$$

where b is a bias, v_i are the inputs (usually, outputs of neurons of the previous layer, or the values of features in the case of the first layer), d is an activation function, and g is a neuron output. In order to build an FCNN model that is able to reproduce complex nonlinear dependencies in the data, the activation functions in the neurons must be nonlinear. Otherwise, in the case of their linearity, the entire neural network could be reduced to a linear mapping. There are many different nonlinear activation functions and more details about them can be found in Ref. [47].

In an FCNN, neurons are organized into layers, where each neuron in the layer is connected to all neurons from the previous one. Such a neural network can be divided into three main parts: the input layer, the hidden layers, and the output layer. The input layer receives features that describe a physical event, while the output layer gives the prediction of the model (in our case, the classification score from 0 to 1). Hidden layers allow the model to expand the space of functions that it is able to approximate. A wide variety of hyperparameters define a

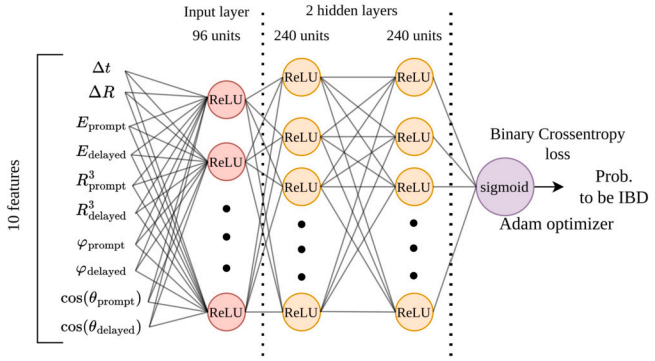


Fig. 4. Network architecture after the optimization procedure. The 10 features introduced in Section 3 are used as input for a fully connected neural network with 3 layers: the input layer with 96 neurons and 2 hidden layers of 240 neurons. As an activation function for the neurons, we use ReLU functions for all the layers except for the output one where with the sigmoid function is used. Binary cross-entropy [50] is used as a loss function, and Adam is used as an optimizer. The model consists of 84k trainable parameters. Being small and compact, the model can provide predictions for more than 1M events per second.

neural network and their optimization is an important part of building the final model. In this study, hyperparameter optimization is performed using the Tree-structured Parzen Estimator algorithm [48] from the Optuna library. Table 2 shows the hyperparameters search space and the selected hyperparameters. We use the PyTorch framework [49] to build and train the model. It takes approximately two days to perform hyperparameter optimization and one hour to train the selected model on a Nvidia A30 GPU. The binary cross-entropy loss [50] is used as a loss function and the sigmoid [47] is used as an activation function for the output layer. All input features were normalized with a standard score normalization. The training process is performed with an early stopping condition on the validation dataset with a *patience* of 20. Here, the patience parameter refers to the number of epochs the training process is allowed to continue without any improvement (on the validation dataset) before being stopped. Fig. 4 shows the optimized FCNN architecture and its main hyperparameters.

6. Results

One of the advantages of using a neural network (as well as a BDT) to select IBD events is the non-binary output of the model. By applying the sigmoid function to the raw output, the models produce a value between 0 and 1 that can be associated with the model's confidence score of an event being an IBD candidate. The models assign this score to each event. The threshold T in score above which an event is considered to be IBD is a tunable parameter. In absence of a prior physics requirement, one can choose a threshold based on balancing efficiency Eq. (1) and purity Eq. (2), maximizing the F1-score:

$$\text{F1-score} = 2 \cdot \frac{\text{purity} \cdot \text{efficiency}}{\text{purity} + \text{efficiency}} \tag{3}$$

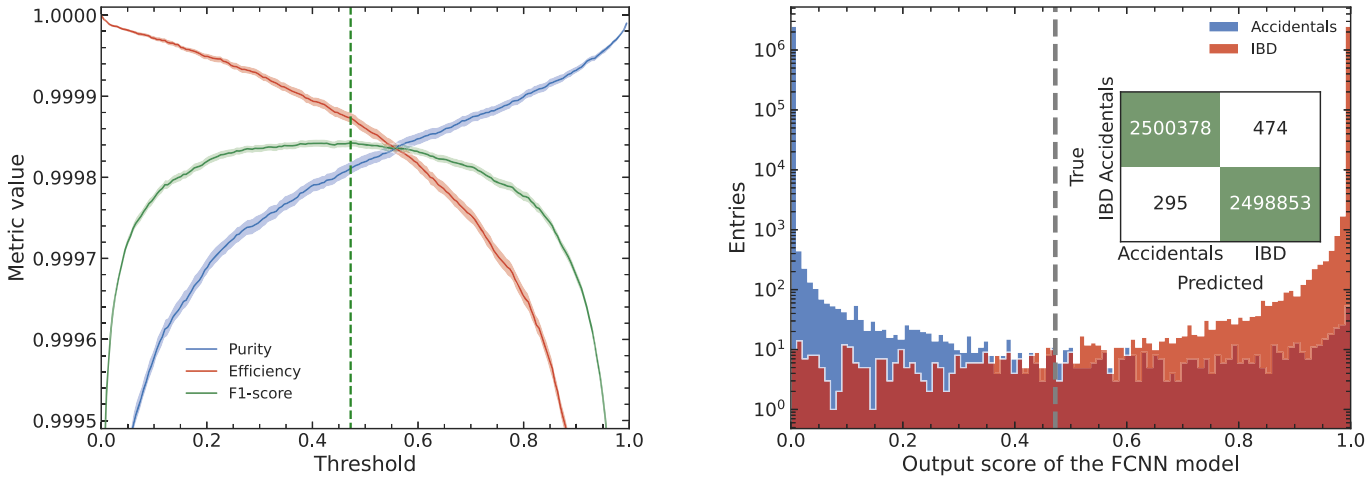


Fig. 5. Left: Medians of the metrics by solid lines and their standard deviations after the bootstrap procedure as a function of the threshold (T-value). The best threshold value is shown with the dashed line. Right: Output score provided by the FCNN model for events from the testing dataset. Most of the events are perfectly separated. The dashed line shows the best T-value. The inset plot represents a confusion matrix of the predictions.

The F1-score is the harmonic mean of purity and efficiency and helps us strike a balance between correctly identifying accidental events (purity) and not missing any IBD events (efficiency). Given the small number of misclassifications, we use the bootstrap technique with the validation dataset to provide a more robust estimation of the optimal threshold value. This technique can be used to evaluate the variability of a parameter by repeatedly sampling from a dataset with replacement. In the context of our application, we re-sample 200 times the entire validation dataset (5M events) and evaluate purity, efficiency, and F1-score metrics at various threshold values. As an example, we use a neural network (but closely the same result is obtained for the BDT) and to assess the model's performance across different threshold values, we vary T from 0 to 1 in a uniform grid of 200 points. The left panel of Fig. 5 reports the result of the evaluation procedure: the median values of purity, efficiency, and F1-score at different T-values are shown with the solid lines, while the corresponding standard deviation is represented by the shaded bandwidths. The best T-value, in terms of F1-score, is ∼0.47. Fig. 5 depicts the FCNN results for the testing dataset and the dashed line stands for the chosen threshold value. As Fig. 5 shows, by varying the T value, one may vary signal to background ratio (and so the efficiency and purity of the selection). This may also be important in various physics analyses. For example, in physics channels where efficiency is the most significant metric, the T threshold can be reduced to retain more IBD events, even if this results in degraded purity. On the other hand, where background hinders the estimation of the parameter of interest, it is important to balance the trade-off of these metrics. Using the optimized T obtained from the maximization of the F1-score as the threshold to assign the IBD class, we get the following metrics for the model: efficiency of 99.988%, purity of 99.981%, and F1-score 99.985%. This procedure for choosing a threshold serves as a generalized and agnostic method, and applicable with no specific requirement provided by a physics analysis. In the following subsection we instead introduce a physics-driven condition to fix a threshold T for the model.

6.1. ML selection and cut-based selection comparison

Table 3 presents a summary of the benchmark IBD selection applied to our dataset. It is worth noting that the outcome of this selection depends on the employed dataset, and may not reflect JUNO's official selection.

To evaluate the performances of the FCNN and BDT models and to compare it with the cut-based selection approach, we use efficiency as the main metric with an additional condition on the background level, i.e., the fraction of selected (classified as IBDs) accidentals with respect to the total number.

Table 3
Summary of the benchmark selection cuts and their single impact on IBD selection efficiency.

Selection Criterion	Efficiency (%)
All IBDs	100.0
FV cut	91.7
IBD Selection	97.1
Energy cut	98.7
Time cut	99.0
Vertex cut	99.4
Combined Selection	89.9

Since only an extremely small number of accidentals satisfy the selection criteria, a very large radioactive sample is required to achieve a quantitative assessment of the two approaches. Therefore, an additional dataset was prepared, consisting of 147.5 million accidental coincidence pairs, corresponding to more than 1 year of data collection [24]. Combined with the testing dataset, it consists of 152.5 million events, with 2.5 million events being IBD and the rest being accidentals. To ensure a comparison between the ML models and the cut-based selection, the models are required to achieve the same background level as the other approach, by choosing a specific threshold. Adhering to this requirement allows us to directly compare the efficiencies.

Table 4 shows the performances of the models and their comparison with the cut-based selection. Two different fiducial volumes are considered for the ML models: (1) the full target volume, $R < 17.7$ m, and (2) within the FV cut, i.e., $R < 17.2$ m. In the first case, thanks to greater flexibility and the ability to work with events at the detector edge, the ML models demonstrate a higher efficiency in tagging IBD events compared to the conventional approach, achieving an improvement of ∼9.1 (BDT) and ∼8.5 (FCNN) percentage points. At the same time, the background level is adjusted to the same value of $\sim 1.27 \times 10^{-6}$ in all the approaches. Moreover, even within the FV volume, the boosted decision trees and the neural network are able to increase the efficiency of tagging IBD events both by ∼1.7 percentage points. This increase comes from events which would be otherwise lost because of likely sub-optimal time and vertex cuts.

One approach to assess the influence of features on a model's output and their interconnections is the partial dependence plot (PDP) [51]. PDP computes the impact of a specific subset of features, typically one or two, by marginalizing (averaging) over all other features in a given

Table 4
The resulting efficiencies of the ML models and the comparison with the cut-based selection. Different background levels are used for comparison: equivalent (1×Bkg), doubled (2×Bkg), fivefold (5×Bkg), halved (0.5×Bkg), and five times lower (0.2×Bkg). For the ML approaches, different background levels are adjusted to the corresponding ones by changing the threshold value. Moreover, we consider two cases for the neural network-based selection and BDT-based selection: (1) the FV cut applied, and (2) the full target volume. The muon veto cut is not included because it yields the same effect in all approaches.

Approach	Volume	Efficiency				
		0.2×Bkg	0.5×Bkg	1×Bkg	2×Bkg	5×Bkg
BDT	Full detector volume: $R < 17.7$ m	98.38%	98.81%	**99.02%**	99.19%	99.39%
	$R < 17.2$ m	91.58%	91.62%	**91.63%**	91.64%	91.64%
FCNN	Full detector volume: $R < 17.7$ m	96.94%	97.79%	**98.40%**	98.82%	99.21%
	$R < 17.2$ m	91.53%	91.60%	**91.63%**	91.64%	91.64%
Cuts	$R < 17.2$ m	—	—	89.90%	—	—

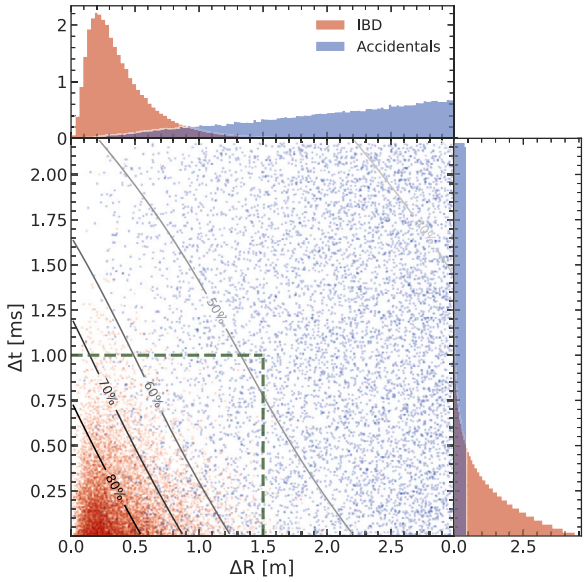


Fig. 6. An example of a partial dependence plot for the following two features: ΔR, Δt. Solid lines represent different FCNN's confidence levels that an event belongs to the *IBD* class. The green lines show the cuts selection criteria. Blue and red points show the events from the testing dataset.

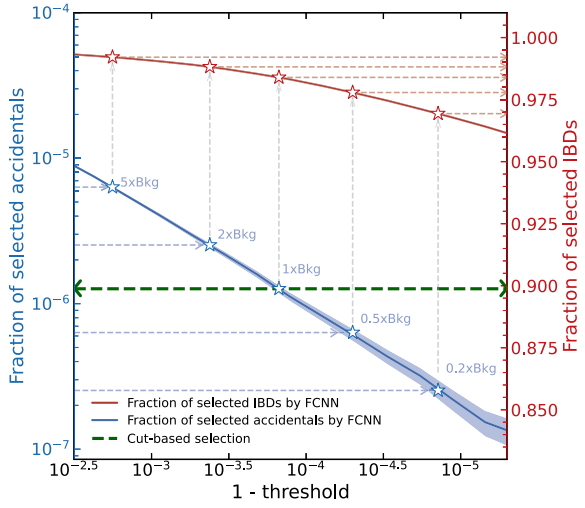


Fig. 7. Comparison of the FCNN model performances and the cut-based selection for different threshold values. The green dashed line depicts the cut-based selection performances. The red and the blue solid lines show the fraction of the selected IBDs (efficiency) and the fraction of the selected accidentals (background level), respectively. Here, *selected* is defined as *classified as IBDs*. By relaxing the threshold, the model can achieve higher efficiency but obtaining more accidental events. The blue dashed lines point to the star markers that indicate the considered background levels, namely, equivalent (1×Bkg), doubled (2×Bkg), fivefold (5×Bkg), halved (0.5×Bkg), and five times lower (0.2×Bkg). The red dashed lines show the corresponding efficiencies.

feature set. Assuming f represents a classifier, $\mathbf{x}_s$ denotes a set of feature values to be used in the evaluation of the PDP, $\mathbf{x}_c$ the remaining features, the partial dependence of $\mathbf{x}_s$ can be estimated as follows:

$$\hat{f}_s(\mathbf{x}_s) = \frac{1}{n}\sum_{i=1}^{n} f(\mathbf{x}_s, \mathbf{x}_{c,i}),$$

where n is the number of events sampled from the training dataset, and $\mathbf{x}_{c,i}$ are values of the corresponding features. Fig. 6 shows an example of a partial dependence plot for the FCNN model, where $\mathbf{x}_s = (\Delta R, \Delta t)$. Here, we use the PDP to illustrate the interconnection between ΔR and Δt and to compare the learned relationship with the cuts. As was mentioned in Section 3, in contrast to the box-like decision boundary ($\Delta R < 1.5$ m, $\Delta t < 1$ ms), the neural network is able to learn a smoother boundary between the two classes, and therefore to improve the efficiency.[4]

Furthermore, four additional scenarios are considered: a background level two times higher (2×Bkg), five times higher (5×Bkg), two times lower (0.5×Bkg), and five times lower (0.2×Bkg) than the one provided by the benchmark selection (1×Bkg). As mentioned earlier, for certain physics analyses, a relatively elevated background level may not be critical, while the additional signal events are important and vice versa. Table 4 shows the change in efficiencies for these four scenarios: an increase of ∼0.17 percentage points (BDT) and ∼0.42 percentage points (FCNN) for the 2xBkg, and an increase of ∼0.37 percentage points (BDT) and ∼0.81 percentage points (FCNN) for the 5xBkg. Additionally, there is a decrease of only ∼0.21 percentage points (BDT) and ∼0.61 percentage points (FCNN) when suppressing the background level by two times, as well as a decrease of ∼0.64 percentage points (BDT) and ∼1.46 percentage points (FCNN) when suppressing the background level by five times. Fig. 7 depicts the dependence of the fraction of selected IBDs (efficiency) and the fraction of selected accidentals (background level) on different thresholds. The FCNN model is used as an example, with similar observations applicable to the BDT model. The green line shows the cut-based selection performances. The red dashed lines represent the efficiencies of the FCNN model with different background level conditions (illustrated by the blue dashed lines and star markers). Therefore, in the 1×Bkg case, the difference between the green line and the red

[4] While BDT has learned an efficient separation boundary, its non-smooth, step-like nature makes it impractical for use as an optimized cut for the cut-based approach.

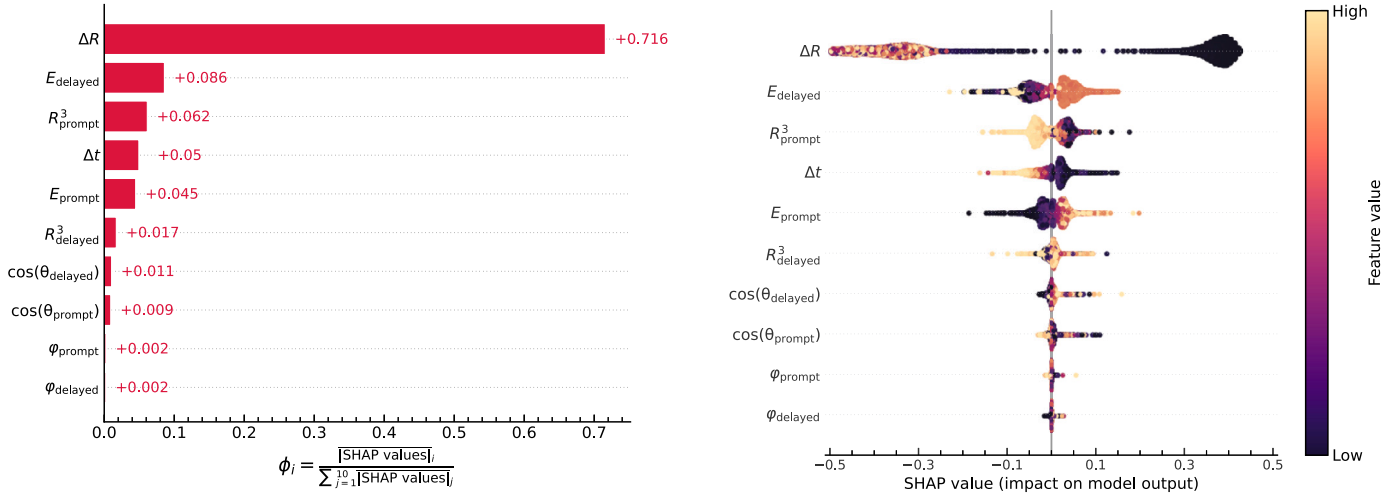


Fig. 8. SHAP-based interpretability plots for the FCNN model. The left one represents the global explanation that summarizes the impact of the features on the model's output. Notice that global explanations are normalized to 1, i.e., $\sum_i^{10} \phi_i = 1$, where i runs over the feature indexes. The right plot illustrates the local explanations for individual predictions and 20k events are shown. One event is represented by a point in each row, displaying the SHAP value associated with the corresponding feature. The density of SHAP values for a given feature represented by "clumps". The color provides the relative value of a feature: dark blue for low values and light blue for high values. Further details are provided in the text.

solid line indicates the increase in signal events statistics with respect to the cut-based selection.

7. Model's interpretability

The *black-box* nature of ML models can be overcome by employing constructs such as the Shapley values, introduced in the mid-20th century by Lloyd Shapley within the domain of cooperative game theory [52]. It stands as a measure to assess the importance of individual players within a coalition in reaching a common objective [13]. Conceptually, the Shapley value gauges the impact of a player by quantifying how the average outcome changes when that player is included in the game, as opposed to its absence. It also serves as a fairness criterion, ensuring that each participant gains at least as much as they would have independently. Therefore, it is a valuable tool in situations where contributions are unequal, yet cooperation among players is essential to achieve a collective payoff [13]. Mathematically, Shapley values provide a means to study the correlations between different variables. By considering all possible combinations of variables entering or leaving the game one can systematically evaluate their impact on the outcome. The main disadvantage lies in the fact that the exact calculation of Shapley values is challenging and requires extensive computation time [13]. In light of this challenge, for our study, we adopt the SHAP (SHapley Additive exPlanations) framework [53]. SHAP [54,55] introduces simplifications to address the computational challenges and speed up calculations while maintaining the interpretability and fairness of feature attributions [53–55]. Explanations provided by SHAP offer valuable insights into the contribution of individual features to model outcomes, facilitating a comprehensive understanding from both global and local perspectives. Indeed, this framework enables us to explore not only the overall importance of features across the entire dataset but also the specific influence of features on individual predictions. Although SHAP values are a model-agnostic method that can be applied to both FCNN and BDT, we use FCNN for further analysis, as it is our main approach and provides all the desired features together. However, it is important to acknowledge that SHAP values may not perfectly capture the complexity of non-linear models, especially in deep learning models.

7.1. Global and local explanations

Global explanations compute the summarized impact of each input feature on the model output. Thanks to this, the generalized importance of the features can be estimated. It helps answer questions such as, "What features are the most important for the model's predictions on average?". Formally, global explanations are averaged absolute SHAP values calculated based on a provided data sample. On the other hand, *local* explanations focus on the individual event and provide features' importance for a specific instance. It helps answer questions like, "Why did the model make this particular prediction for this specific data point?" We compute SHAP values for 20k events from the testing dataset. SHAP values can be both positive and negative, showing the impact on predictions with respect to the average value of the output variable (the labels 0 or 1 in a binary classification case). Since our datasets are balanced by construction, the mean value is equal to 0.5. Thus, positive SHAP values indicate the contribution of a feature to pushing the model's output towards the IBD class. On the other hand, negative SHAP values indicate a contribution towards the *accidentals* class.

The left part of Fig. 8 illustrates the global explanations for the FCNN model.[5] Each row corresponds to a feature and the bars' width represent the feature importance. The values are normalized to 1, so that $\sum_i^{10} \phi_i = 1$. The most important features are, in order, ΔR, E_{delayed}, R^3_{prompt}, and Δt. The impact of E_{prompt} and R^3_{delayed}, being smaller on average, helps in the selection of rarer cases. The same applies to the cosine theta features, which allow the model to correct the prediction, especially for values at their extremes. In contrast to $\cos(\theta)$, the azimuthal angles $\varphi_{\text{prompt}}, \varphi_{\text{delayed}}$ features have almost negligible importance. We would also like to highlight that the independent method for assessing global feature importances — gain feature importance for BDT — is mostly in agreement with the results obtained with SHAP with FCNN. Both methods identify the same three main clusters of features: ΔR, radii, energies and Δt, followed by the angles-related features.

Local explanations help to better understand why certain features are more or less important and in which cases. The right panel of Fig. 8 illustrates a set of SHAP values for each feature for 20k events taken from the testing dataset. One event is a point in each row, hence it is decomposed into ten points. The color represents the relative value of a feature: from dark blue (values are close to its minimum) to light blue (values are close to its maximum). The concentration of events on cer-

[5] It is worth noting that another independent method for assessing feature importance ranking — permutation importance [56] — provides nearly identical results.

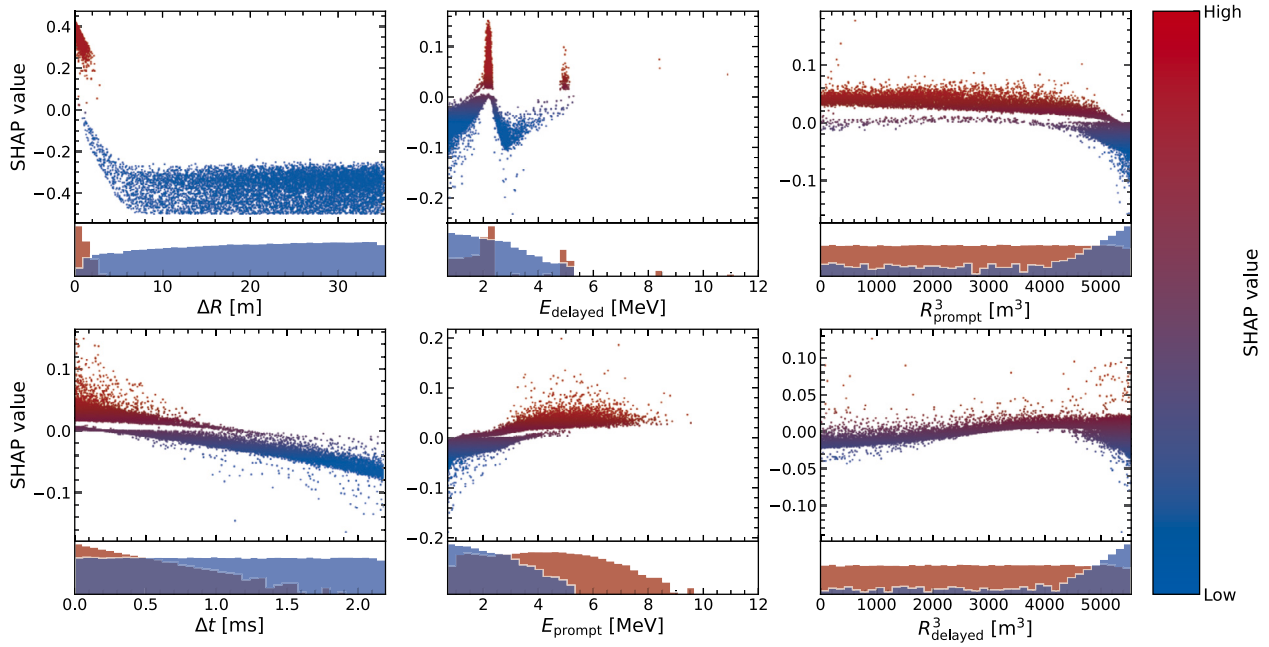


Fig. 9. Detailed local explanations (top panel) for top-6 features and their distributions (bottom panel). Color represents SHAP value: from blue (negative, more confident to be accidentals) to red (positive, more confident to be IBD). The normalization of the color scale is set to enhance the differences between the two classes in terms of SHAP values.

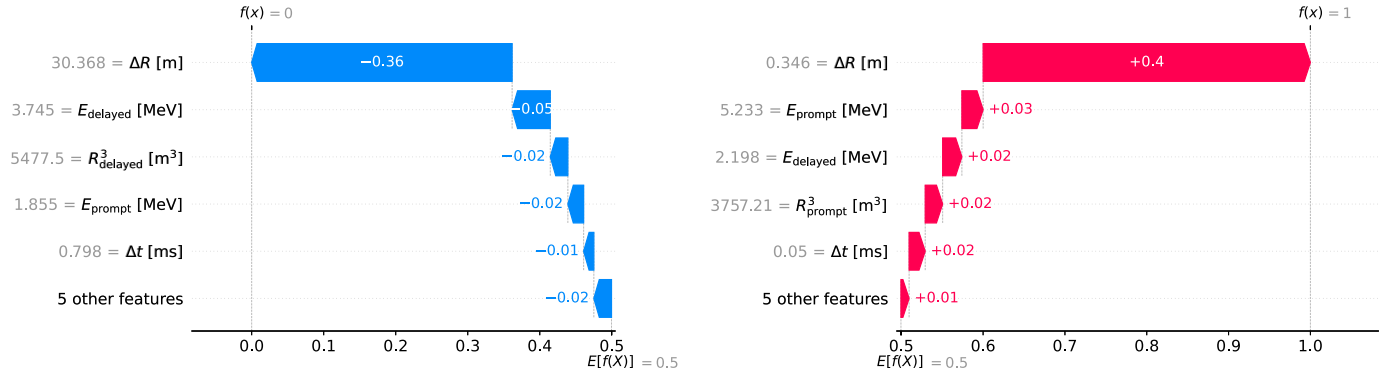
tain SHAP values is shown as "clumps". Regarding ΔR, there is a clear correlation between its value, (i.e., the color of a data point), and the corresponding SHAP value (i.e., the position on the horizontal axis): for events with smaller ΔR the model is more confident to assign the *IBD* class (positive SHAP values) than for events with large ΔR (negative SHAP values). The next most important feature is E_{delayed} because its distribution is different for the two classes: in particular, the clustered structure (related to the different gamma emission peaks) that we observe for IBD events is not present for accidentals. For IBD events, it is strictly related to the isotope that captures the IBD neutron: 2.2 MeV (^{1}H), 4.95 MeV (^{12}C), higher energies (^{13}C, ^{14}N). Thus, positive SHAP values are associated with events with these particular E_{delayed}. While the cut-based selection completely reject events with higher E_{delayed} energies, the FCNN model is able to preserve them, increasing efficiency. Another energy-related feature, E_{prompt}, has the following dependence: at small values, the model is more confident that the events belong to the *accidentals* class since this part of the energy spectrum is populated mainly by the ^{14}C isotope, having very few events associated with reactor antineutrinos. On the contrary, accidentals with higher energies are almost nonexistent and IBDs dominate, resulting in positive SHAP values.

Fig. 9 shows detailed explanations for the top-6 features: for each event, the top panel reports the feature value on the x-axis, with the corresponding SHAP value on the y-axis. The bottom panel shows histograms of the feature distributions, for accidental coincidences in blue and IBD pairs in red, as previously reported in Fig. 2. The colorbar shows the relative contribution for these features in terms of SHAP value, while also being a proxy for the y-axis values. These explanations are useful to visualize the relation between the feature distributions and their impact on the model's output. Bright red and blue regions correspond to features significantly pushing the model to the *IBD* and *accidentals* classes, respectively. On the other hand, purple-shaded areas provide little contribution to the model's output. For example, as was mentioned above, E_{delayed} explanations have a clear clustered structure. There are several clusters of positive SHAP values associated with the released gamma energies from neutron capture on different isotopes. The width of the cluster can be used as cut boundaries for the benchmark selection procedure. Fig. 9 also shows a clear clustering structure for ΔR and E_{prompt}. Indeed, for events with $\Delta R \lesssim 2$ m the model is more confident to assign the *IBD* class, while for events with $\Delta R \gtrsim 2$ m, FCNN is less confident, resulting in negative SHAP values. Concerning E_{prompt}, SHAP values follow the feature distributions and have an overlapping in the region of [1.5, 4] MeV. In this energy range, correlations with other features play a key role (mainly E_{delayed} and ΔR), allowing the model to distinguish IBDs from accidentals. The time-related feature Δt mostly pushes towards the *IBD* class when it is within several neutron capture times τ. On the other hand, in the case of highly separated in time events, they are considered more probable to be accidentals. Concerning the position-related features (R^3_{prompt}, R^3_{delayed}) in Fig. 9, their absolute values become more significant closer to the edge, increasing the confidence that an event is accidental. It is interesting to note that some events have a large positive SHAP value for $R^3_{\text{delayed}} > 17.65^3$ m^3. This is because, in the case of events at the very edge of the detector, the probability of gamma leakage becomes higher. In a liquid scintillator detector, gamma leakage refers to the energy loss caused by gamma rays not depositing all their energy within the detector volume. Even though the energy of the delayed event is much less than expected, the event is correctly classified as IBD according to the values of all other features, including information about proximity to the edge.

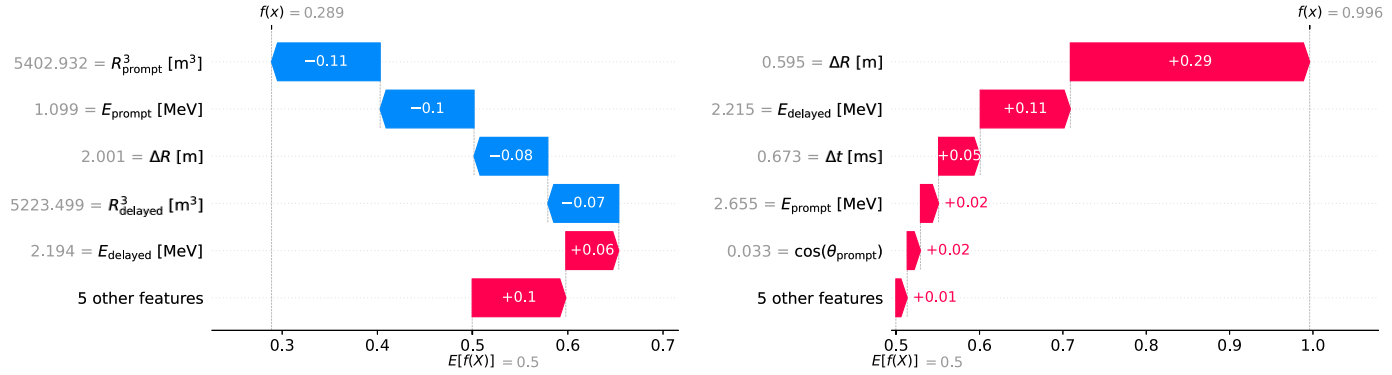
7.2. Special cases

Local explanations, which provide insights into how the model makes decisions for individual events, are an effective tool for debugging the model and identifying special cases. In order to do this, we employ the so-called *waterfall* plot [53], a visualization tool that conveys the impact of SHAP values on our model's output. The x-axis reports the expected value of the model output $E[f(x)]$: it starts from a baseline value, set at 0.5 in our case due to dataset balance,[6] and each subsequent row shows how each feature contributes to the overall prediction. The color indicates whether a specific feature pushes the prediction higher (red,

[6] If we randomly sample an event from the dataset, we have a 50% probability for it to be either an IBD or an accidental coincidence.



(a) SHAP interpretability plot of FCNN predictions for typical cases of accidentals (left) and IBD (right).



(b) SHAP interpretability plot for cases of wrong predictions of FCNN for true IBD (left) and true accidentals (right).

Fig. 10. SHAP explanations provided for particular cases of correct classifications (a) and misclassifications (b). Features are sorted based on the magnitude of their SHAP values, and the smallest magnitude features are clustered at the bottom of the plot.

i.e., more confident to be IBD) or lower (blue, thus more confident to be background) than the base value.

Focusing on specific cases, Fig. 10a reports SHAP values for events that were correctly classified, namely a typical accidental coincidence on the left panel and an IBD pair on the right side. The accidental event has a large ΔR of ∼30 m and E_{delayed} outside the energy ranges of neutron capture gammas. This combination of values already allows the model to designate this event as a random coincidence with ∼90% confidence. On the other hand, for the right event with $\Delta R = 0.35$ m, $E_{\text{prompt}} = 5.23$ MeV, and E_{delayed} ∼2.2 MeV the model assigned a ∼100% confidence score to be an IBD event.

Moreover, Fig. 11 shows an example of a correctly classified gamma leakage event that would be discarded by the cut-based selection because of the very low E_{delayed} and the FV cut. Contrariwise, the ML model is able to identify this kind of events thanks to the combination of all other features. The efficiency of classification of events with the gamma leakage effect is $\gtrsim$ 95%, using the threshold optimized based on F1-score maximization.

On the other hand, Fig. 10b illustrates cases when the model made a wrong prediction. The left one is a true IBD with an escaped gamma that was classified as an accidental event. Unlike the gamma leakage event presented above in Fig. 11, this event has a low classification score mostly because of the atypical high ΔR. The right panel of Fig. 10b shows a true accidental event that was classified as IBD. This misclassification was caused by the unlikely case of accidentals with an IBD-like combination of features.

8. Model calibration and uncertainty quantification

In this section, we discuss two important aspects of building a trustworthy and reliable machine learning model: model calibration and uncertainty estimation.

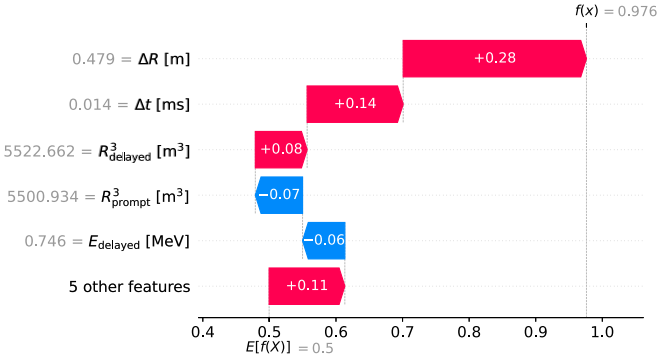


Fig. 11. An example of an IBD event with the gamma leakage effect: the gamma produced by neutron capture did not deposit its entire energy in the LS, but instead escaped the target. Despite the almost complete energy leakage (E_{delayed} of 0.746 MeV), the model classifies this as an IBD event with ∼100% confidence, based on the combination of other features.

Model calibration refers to the process of aligning the output scores predicted by a classifier to probabilities [57]. A well-calibrated model produces scores that can be directly interpreted as probabilities, allowing for straightforward use of the classifier's predictions in situations where a probabilistic interpretation is a desired and pivotal output. In these cases, the relationship between probability and classifier output score appears as a diagonal straight line with unit slope and zero intercept. Uncalibrated models may deviate from this straight line, associating high output score values with low probabilities (i.e., overconfident classifiers), or conversely associating low score values with high probabilities (i.e., underconfident), or combinations of the two. Model calibration is not always a fundamental requirement, but it greatly simplifies

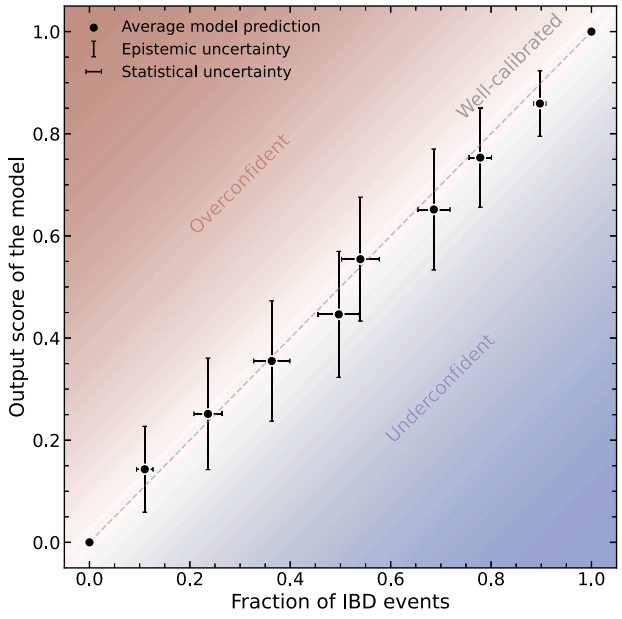


Fig. 12. Calibration plot for the FCNN model with the Monte Carlo dropout technique applied. The model is well-calibrated across the 5M events testing dataset. On the x-axis the fraction of IBD events in each bin (frequentist probability) and its statistical uncertainty are presented. The y-axis displays the average predictions and corresponding epistemic uncertainty provided by the MC dropout. The uncertainty estimation is done using 100 forward passes.

the interpretation of model decisions. Finally, calibration is not an intrinsic property of the model, but also depends on the reference dataset. Various techniques exist to calibrate uncalibrated models, should this be required during use [58].

The second aspect to address is uncertainty estimation. In the machine learning literature, uncertainty estimation is often divided into two types: (i) aleatoric and (ii) epistemic uncertainty [59]. Aleatoric uncertainty is related to the inherent randomness in the data itself, making it irreducible [59]. In contrast, epistemic uncertainty refers to the intrinsic model uncertainty, which is potentially reducible by providing additional information during the training of the model [59]. Understanding and quantifying uncertainty not only improves model interpretability but also provides a measure of confidence in its predictions. This is particularly important when dealing with areas of sparse data or when identifying out-of-distribution events, where predictions are likely to be less reliable. Several methods exist for quantifying the impact of model uncertainty [60]. An effective and relatively simple to implement method for estimating epistemic uncertainty is Monte Carlo (MC) dropout [61].

MC dropout is a technique that uses dropout [62] to approximate Bayesian inference during both model training and inference. In standard dropout, neurons of the network are randomly "dropped" (i.e., set to zero) during training with a probability p for each neuron, and then at inference stage, dropout is turned off. However, in MC dropout, dropout is kept active during inference as well: by performing multiple forward passes, it enables the model to make predictions under slightly different conditions (different set of active neurons) for each pass, allowing us to capture the variance in these predictions as a measure of uncertainty. Thus, MC dropout keeps the stochasticity at inference stage as well.

In this study, we applied dropout layers to our FCNN classifier after each fully connected layer, using a dropout probability of $p = 0.2$. For each event in our testing dataset, we performed 100 forward passes, recording the average prediction and its standard deviation. Fig. 12 presents a calibration plot showing both the epistemic uncertainty estimated using the MC dropout technique and the model's calibration. On the x-axis, we display the fraction of IBD events (frequentist probability) in each bin and its statistical uncertainty, while the y-axis shows the corresponding averaged predictions and their standard deviations, representing epistemic uncertainty as estimated by MC dropout. This plot demonstrates that our model is well-calibrated across the testing dataset, as the predictions closely align with the diagonal line representing perfect calibration. Additionally, the uncertainty (represented by the error bars) is minimal at the edges (where predictions are closer to 0 or 1), and increases in the middle of the range. This increase in uncertainty around predictions near 0.5 can be attributed to the sparser data points in this region, which makes the model less confident in its predictions.

Additionally, in the case of a realistic class imbalance (with significantly more accidentals than IBD events), the same model will show overconfidence: a smaller fraction of IBD events will correspond to the same output score. However, this discrepancy can be easily resolved with a simple scaling procedure using the following expression: $p = \frac{s}{1+(R-1)(1-s)}$, where R is the class imbalance ratio: $R = \frac{N_{\text{accidentals}}}{N_{\text{IBD}}}$, s represents the confidence scores provided by the model, p is the resulting probability. We would like to note that the rates can be estimated independently from data before running the ML selection.

9. Conclusions

In this study, we introduced a machine learning model, specifically a fully connected neural network, for event selection in a large liquid scintillator detector. Taking the JUNO experiment as a case study, we demonstrate that the presented ML model (i) is capable of matching the performance of a boosted decision trees based classifier and (ii) is able to learn a more flexible boundary between signal and background events compared to a cut-based selection criteria. This consequently leads to a ∼1.7 percentage points increase of efficiency within the fiducial volume. Moreover, the ML approach opens up the possibility to remove the strict fiducial volume cut, retaining a higher number of signal events, providing an improvement of ∼8.5 percentage points in efficiency. For both cases, the model keeps exactly the same background level as the cut-based selection. It also proves to be powerful in tagging events characterized by gamma leakage, that would otherwise be discarded by the cuts. Furthermore, we outline a systematic approach for preparing datasets and optimizing model hyperparameters. This methodology is not exclusive to JUNO but can be extended and applied to other liquid scintillator-based detectors, for any supervised learning tasks. A key aspect of our study involves interpretability analysis, aimed at investigating the decision-making process of the ML model and offering valuable insights into its behavior. This deepened understanding contributes to refining cut-based event selection strategies, ensuring the robustness of model predictions, both at the local (individual event) and global (across a set of events) levels. Part of the work has been devoted to model calibration and estimating the epistemic uncertainty of its predictions in order to corroborate the confidence in the network's decisions and outputs. In summary, our work underscores the potential of the ML approach to optimize event selection for inverse beta decay interactions in neutrino experiments. This flexibility proves particularly advantageous in striking a balance between purity and efficiency tailored to the physics channel of interest.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Arsenii Gavrikov reports financial support was provided by European Union. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

We are thankful to the JUNO reactor antineutrino physics working group for the support and advice provided during the drafting of this manuscript. We are also grateful to CloudVeneto for providing IT support and GPU resources, and to the IHEP computing center for providing access to their computing cluster. This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie Grant Agreement No. 101034319 and from the European Union – NextGenerationEU.

Data availability

Data will be made available on request.

References

[1] D. Bourilkov, Machine and deep learning applications in particle physics, Int. J. Mod. Phys. A 34 (2019) 1930019.
[2] M.D. Schwartz, Modern machine learning and particle physics, Harv. Data Sci. Rev. (2021).
[3] D. Rocco, M. Messier, E. Niner, G. Pawloski, P. Vahle, A convolutional neural network neutrino event classifier, J. Instrum. 11 (2016) P09001.
[4] J. Renner, et al., Background rejection in NEXT using deep neural networks, J. Instrum. 12 (2017) T01004.
[5] Z. Qian, et al., Vertex and energy reconstruction in JUNO with machine learning methods, Nucl. Instrum. Methods A 1010 (2021) 165527.
[6] A. Gavrikov, Y. Malyshkin, F. Ratnikov, Energy reconstruction for large liquid scintillator detectors with machine learning techniques: aggregated features approach, Eur. Phys. J. C 82 (2022) 1021.
[7] V. Chekalina, et al., Generative models for fast calorimeter simulation: the LHCb case, EPJ Web Conf. 214 (2019) 02034.
[8] MODE collaboration, Toward the end-to-end optimization of particle physics instruments with differentiable programming, Rev. Phys. 10 (2023) 100085.
[9] MicroBooNE collaboration, Reconstruction and selection of neutrino interactions in MicroBooNE using deep convolutional neural networks, https://doi.org/10.2172/2406059, July 2024.
[10] P. Fernandez Menendez, Atmospheric neutrino oscillations with Super-Kamiokande and prospects for SuperK-Gd, PoS ICRC2021 (2021) 008.
[11] T. Sakai, Advanced new tool for background rejection in KamLAND geo-neutrino analysis using machine learning methods, Neutrino24, https://doi.org/10.5281/zenodo.13122055, June 2024.
[12] V. Belle, I. Papantonis, Principles and practice of explainable machine learning, Front. Big Data (2021) 39.
[13] C. Molnar, Interpretable Machine Learning: A Guide for Making Black Box Models Explainable, Online, 2022.
[14] I. Arnquist, et al., Interpretable boosted-decision-tree analysis for the Majorana demonstrator, Phys. Rev. C 107 (2023) 014321.
[15] C.L. Cowan Jr, et al., Detection of the free neutrino: a confirmation, Science 124 (1956) 103.
[16] P. Vogel, J. Beacom, Angular distribution of neutron inverse beta decay, $\bar{\nu}_e + p \rightarrow e^+ + n$, Phys. Rev. D 60 (1999) 053003.
[17] K. Eguchi, et al., First results from KamLAND: evidence for reactor antineutrino disappearance, Phys. Rev. Lett. 90 (2003) 021802.
[18] F. An, et al., Observation of electron-antineutrino disappearance at Daya Bay, Phys. Rev. Lett. 108 (2012) 171803.
[19] Y. Abe, et al., Indication of reactor ν_e disappearance in the Double Chooz experiment, Phys. Rev. Lett. 108 (2012) 131801.
[20] J.K. Ahn, et al., Observation of reactor electron antineutrinos disappearance in the RENO experiment, Phys. Rev. Lett. 108 (2012) 191802.
[21] F. An, et al., Neutrino physics with JUNO, J. Phys. G, Nucl. Part. Phys. 43 (2016) 030401.
[22] A. Abusleme, et al., JUNO physics and detector, Prog. Part. Nucl. Phys. 123 (2021) 103927.
[23] M. Grassi, The JUNO experiment: status and prospects, NNN23, https://doi.org/10.5281/zenodo.10624758, Oct. 2023.
[24] A. Abusleme, et al., Radioactivity control strategy for the JUNO detector, J. High Energy Phys. 2021 (2021) 1.
[25] A. Abusleme, et al., Sub-percent precision measurement of neutrino oscillation parameters with JUNO, Chin. Phys. C 46 (2022) 123001.
[26] T. Lin, et al., Simulation software of the JUNO experiment, Eur. Phys. J. C 83 (2023) 382.
[27] J.H. Friedman, Stochastic gradient boosting, Comput. Stat. Data Anal. 38 (2002) 367.
[28] J.H. Friedman, Greedy function approximation: a gradient boosting machine, Ann. Stat. 29 (2001) 1189.
[29] D. McElfresh, et al., When do neural nets outperform boosted trees on tabular data?, arXiv:2305.02997, 2024.
[30] A. Abusleme, T. Adam, S. Ahmad, R. Ahmed, S. Aiello, M. Akram, et al., Calibration strategy of the JUNO experiment, J. High Energy Phys. 2021 (2021) 1.
[31] Arsenii Gavrikov, Andrea Serafini, NeuroMCT: fast Monte Carlo tuning with generative machine learning in the JUNO experiment, CHEP24, https://indico.cern.ch/event/1338689/contributions/6015882/, Oct. 2024.
[32] NuFIT 5.2, Three-neutrino fit based on data available in November 2022, http://www.nu-fit.org/, 2022.
[33] T. Chen, C. Guestrin, Xgboost: a scalable tree boosting system, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 785–794.
[34] T. Akiba, S. Sano, T. Yanase, T. Ohta, M. Koyama, Optuna: a next-generation hyperparameter optimization framework, arXiv:1907.10902, 2019.
[35] V. Nair, G.E. Hinton, Rectified linear units improve restricted Boltzmann machines, in: Proceedings of the 27th ICML Conference, ICML'10, 2010, pp. 807–814.
[36] A.L. Maas, et al., Rectifier nonlinearities improve neural network acoustic models, in: Proc. ICML 30, 2013, p. 3.
[37] D. Hendrycks, K. Gimpel, Gaussian error linear units GELUs, arXiv:1606.08415, 2023.
[38] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: surpassing human-level performance on imagenet classification, in: Proceedings of the IEEE International Conference on Computer Vision (ICCV), December 2015.
[39] D.P. Kingma, J. Ba, Adam: a method for stochastic optimization, arXiv:1412.6980, 2017.
[40] H. Robbins, S. Monro, A stochastic approximation method, Ann. Math. Stat. 22 (1951) 400.
[41] T. T., H. G., Lecture 6.5 - RMSprop: Divide the gradient by a running average of its recent magnitude, 2012.
[42] Z. Li, S. Arora, An exponential learning rate schedule for deep learning, arXiv:1910.07454, 2019.
[43] I. Loshchilov, F. Hutter, SGDR: stochastic gradient descent with warm restarts, arXiv:1608.03983, 2017.
[44] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: Proceedings of the 13th AISTATS, 2010, p. 249.
[45] A.M. Saxe, J.L. McClelland, S. Ganguli, Exact solutions to the nonlinear dynamics of learning in deep linear neural networks, arXiv:1312.6120, 2014.
[46] S. Ioffe, C. Szegedy, Batch normalization: accelerating deep network training by reducing internal covariate shift, arXiv:1502.03167, 2015.
[47] J. Lederer, Activation functions in artificial neural networks: a systematic overview, arXiv:2101.09957, 2021.
[48] J. Bergstra, R. Bardenet, Y. Bengio, B. Kégl, Algorithms for hyper-parameter optimization, Adv. Neural Inf. Process. Syst. 24 (2011).
[49] A. Paszke, et al., Pytorch: an imperative style, high-performance deep learning library, arXiv:1912.01703, 2019.
[50] I.J. Good, Rational decisions, J. R. Stat. Soc., Ser. B, Methodol. 14 (1952) 107.
[51] B.M. Greenwell, B.C. Boehmke, A.J. McCarthy, A simple and effective model-based variable importance measure, arXiv:1805.04755, 2018.
[52] L. Shapley, E. Artin, M. Morse, Quota solutions op n-person games.
[53] S.M. Lundberg, S.-I. Lee, A unified approach to interpreting model predictions, Adv. Neural Inf. Process. Syst. 30 (2017).
[54] S.M. Lundberg, et al., From local explanations to global understanding with explainable ai for trees, Nat. Mach. Intell. 2 (2020) 56.
[55] S.M. Lundberg, G.G. Erion, S.-I. Lee, Consistent individualized feature attribution for tree ensembles, Methods 5 (2018) 25.
[56] L. Breiman, Random forests, Mach. Learn. 45 (2001) 5.
[57] A.P. Dawid, The well-calibrated bayesian, J. Am. Stat. Assoc. 77 (1982) 605.
[58] C. Guo, G. Pleiss, Y. Sun, K.Q. Weinberger, On calibration of modern neural networks, in: Proceedings of the 34th International Conference on Machine Learning, vol. 70, 2017, p. 1321.
[59] E. Hüllermeier, W. Waegeman, Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods, Mach. Learn. 110 (2021) 457.
[60] M. Abdar, et al., A review of uncertainty quantification in deep learning: techniques, applications and challenges, Inf. Fusion 76 (2021) 243.
[61] Y. Gal, Z. Ghahramani, Dropout as a bayesian approximation: representing model uncertainty in deep learning, in: Proceedings of the 33rd International Conference on Machine Learning, vol. 48, 2016, p. 1050.
[62] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (2014) 1929.