UNIVERSITÀ DEGLI STUDI DI PADOVA

Dipartimento di Fisica e Astronomia "Galileo Galilei"

Corso di Laurea in Fisica

Tesi di Laurea

Ricostruzione dell'energia nell'esperimento JUNO con tecniche di Machine Learning

Energy reconstruction in the JUNO experiment with Machine Learning techniques

Relatore

Prof. Alberto Garfagnini

Correlatore

Arsenii Gavrikov

Laureando

Tommaso Ferracci

Anno Accademico 2022/2023

Abstract

The Jiangmen Underground Neutrino Observatory (JUNO) is a neutrino observatory currently under construction in South China. The JUNO central detector is a 35.4 m diameter sphere filled with 20 kt of liquid scintillator (LS). Thanks to an unprecedented effective energy resolution better than $3\%/\sqrt{E(\mathrm{MeV})}$, JUNO is expected to determine the neutrino mass hierarchy, one of the still open questions in neutrino physics. The interactions of rector anti-neutrinos in the detector happen via the so-called inverse beta decay (IBD), where an anti-neutrino interacts with a proton producing a positron and a neutron in the final state. The positron deposits its energy in the LS and annihilates into two gammas. The scintillation light produced after energy deposition is collected by photomultipliers (PMTs) and used for energy reconstruction of the neutrino interactions.

Because of its ability to automatically learn complex non-linear dependencies, Machine Learning (ML) represents a valid alternative to traditional energy reconstruction algorithms. In this work, based on recently published papers, 162 features are engineered from the information provided by individual PMTs to train three different ML models: boosted decision trees (BDT), a fully connected deep neural network (FCDNN) and a 1-dimensional convolutional neural network (1DCNN). All models satisfy the requirement on the effective energy resolution to determine the neutrino mass ordering, and FCDNN results in the best combination of performance and stability. In addition, possible future improvements to the 1DCNN architecture are outlined, based on its promising preliminary results.

Il Jiangmen Underground Neutrino Observatory (JUNO) è un osservatorio di neutrini attualmente in costruzione nel Sud della Cina. Il detector centrale di JUNO è una sfera da 35.4 m di diametro riempita con 20 kt di scintillatore liquido. Grazie a una risoluzione energetica efficace migliore di $3\%/\sqrt{E(\mathrm{MeV})}$ ci si aspetta che JUNO determini la gerarchia di massa dei neutrini, una delle domande ancora aperte nella fisica dei neutrini. L'interazione degli anti-neutrini da reattore all'interno del detector avviene tramite decadimento beta inverso (IBD), dove un anti-neutrino interagisce con un protone producendo un neutrone e un positrone. Il positrone deposita la sua energia nello scintillatore liquido e annichilisce in due gamma. La luce di scintillazione prodotta in seguito al deposito di energia viene raccolta da fotomoltiplicatori (PMTs) e utilizzata per la ricostruzione energetica delle interazioni di neutrini.

Per via della sua abilità di apprendere automaticamente dipendenze non-lineari complesse, il Machine Learning (ML) rappresenta una valida alternativa a algoritmi tradizionali per la ricostruzione dell'energia. In questo lavoro, basato su pubblicazioni recenti, 162 features vengono ingegnerizzate a partire dall'informazione fornita dai singoli PMTs con lo scopo di addestrare tre diversi modelli di ML: alberi di decisione (BDT), una rete neurale completamente connessa (FCDNN) e una rete neurale convoluzionale unidimensionale (1DCNN). Tutti i modelli soddisfano il requisito sulla risoluzione energetica efficace per determinare l'ordinamento di massa dei neutrini, e FCDNN fornisce la migliore combinazione di performance e stabilità. Vengono in aggiunta proposti suggerimenti su possibili futuri miglioramenti dell'architettura 1DCNN, a partire da risultati preliminari promettenti.

Contents

Chapter 1

Introduction

1.1 The JUNO Experiment

The Jiangmen Underground Neutrino Observatory (JUNO) [1, 2] is a large neutrino experiment currently under construction near Kaiping, in the region of Guangdong in South China. The commissioning of the main detector, followed by science runs, is expected to begin in 2024. The main purpose of the experiment is to establish the *neutrino mass ordering* (NMO).

It is known from particle physics that neutrinos exist in three different *flavors*: electronic ($|\nu_e\rangle$), muonic ($|\nu_\mu\rangle$) and tauonic ($|\nu_\tau\rangle$). While neutrinos were long believed to be mass-less particles, in 1998 the Super-Kamiokande Collaboration announced the first experimental evidence of neutrino oscillation [3] and, as a consequence, of non-zero neutrino mass. More precisely, each flavor eigenstate can be expressed as a quantum superposition of three different mass eigenstates ($|\nu_1\rangle$, $|\nu_2\rangle$ and $|\nu_3\rangle$), meaning that neutrinos oscillate between different flavors as they travel through space and time. The relationship between flavor and mass eigenstates is given by the Pontecorvo-Maki-Nakagawa-Sakata (PMNS) matrix U [4, 5]:

$$\begin{pmatrix} \nu_e \\ \nu_\mu \\ \nu_\tau \end{pmatrix} = \begin{pmatrix} U_{e1} & U_{e2} & U_{e3} \\ U_{\mu 1} & U_{\mu 2} & U_{\mu 3} \\ U_{\tau 1} & U_{\tau 2} & U_{\tau 3} \end{pmatrix} \begin{pmatrix} \nu_1 \\ \nu_2 \\ \nu_3 \end{pmatrix}$$

In the Standard Model of particle physics the PMNS matrix is unitary. It can in general be parameterized with 3 *mixing angles* (θ_{12}, θ_{13} and θ_{23}) and single phase angle δ_{CP} related to charge-parity violations [6].
While the mixing angles have already been measured with precisions at a level of few percents, δ_{CP} and the octant of θ_{23} (i.e., $\theta_{23} < \frac{\pi}{4}$ or $\theta_{23} > \frac{\pi}{4}$) are still unknown. JUNO's sensitivity the the NMO is independent of both these parameters, providing complementary information with respect to accelerator and atmospheric experiments which are instead reliant on the matter effects in neutrino oscillation.

The main sources of electron antineutrinos ($|\bar{\nu_e}\rangle$) for the JUNO experiment are two nuclear power plants located roughly 53 km away from the detector. Due to neutrino oscillations, their survival probability in vacuum [7] can be written as:

$$\begin{aligned} P_{\bar{\nu_e}\to\bar{\nu_e}} = 1 - \sin^2(2\theta_{13}) \left(\cos^2(\theta_{12}) \sin^2 \left(\Delta m^2_{31} \frac{L}{4E^\nu} \right) + \sin^2(\theta_{12}) \sin^2 \left(\Delta m^2_{32} \frac{L}{4E^\nu} \right) \right) \\ - \cos^4(\theta_{13}) \sin^2(2\theta_{12}) \sin^2 \left(\Delta m^2_{21} \frac{L}{4E^\nu} \right) \end{aligned}$$

where $\Delta m^2_{ij} = m^2_i - m^2_j$ is the mass-square-difference between eigenstates i and j, $L \approx 53\,\text{km}$ is the distance from the power plants and E^ν is the antineutrino energy. While Δm^2_{21} has been measured in previous experiments, there are two possibilities regarding the neutrino mass ordering:

- Normal Ordering (NO): $|\Delta m^2_{31}| > |\Delta m^2_{32}| \implies m_1 < m_2 < m_3$
- Inverted Ordering (IO): $|\Delta m^2_{31}| < |\Delta m^2_{32}| \implies m_3 < m_1 < m_2$

JUNO will measure oscillations driven by both small mass splitting (Δm^2_{21}) and large mass splitting (Δm^2_{31} and Δm^2_{32}), as shown in fig. 1.1. An unprecedented relative energy resolution of $\sigma_E/E = 3\%/\sqrt{E\,(\mathrm{MeV})}$ will be needed to identify the correct NMO in the small oscillation peaks.

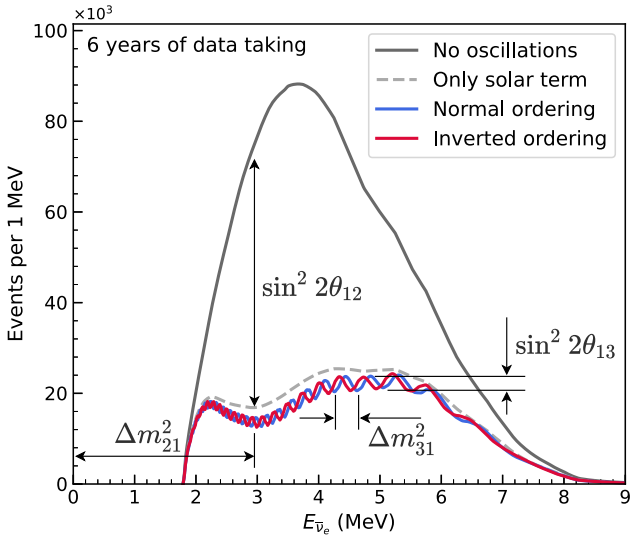


Figure 1.1: Expected reactor antineutrino energy spectrum with (black) and without (gray, blue and red) the effect of neutrino oscillation. The spectra are weighted by IBD cross-section [2].

The JUNO detector (fig. 1.2) consists of a Central Detector (CD), a water Cherenkov detector and a Top Tracker. The CD contains in a transparent 12-cm thick, 35.4 m in diameter acrylic sphere the primary antineutrino target: 20 kton of liquid scintillator. The acrylic sphere is surrounded by 17612 large 20-inch photomultiplier tubes (LPMTs) and 25600 small 3-inch photomultiplier tubes (SPMTs). The former provide 75.2% photocatode coverage, and the latter add an extra 2.7%. The ultra-pure water Cherenkov detector serves both as a passive shield against external radioactivity and neutrons from cosmic rays and as an active veto for cosmic muons (efficency $> 99.5\%$). It is equipped with 2400 LPMTs to detect Cherenkov light from muons. The veto system is completed by the Top Tracker, located at the top of the detector and capable of detecting muon tracks through three layers of plastic scintillator.

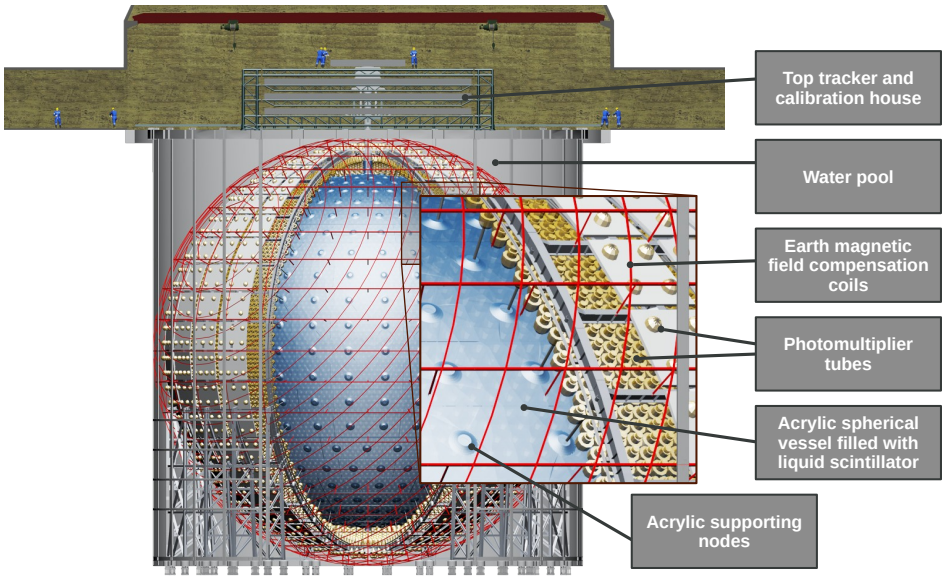


Figure 1.2: Schematic view of the JUNO detector [2].

1.2 Inverse Beta Decay

All reactors in the proximity of JUNO are pressurized water reactors where fissions of four main isotopes, ^{235}U, ^{238}U, ^{239}Pu and ^{241}Pu, account for $> 99.7\%$ of the antineutrinos. The antineutrino flux $\phi(E^\nu)$ is given by these four components weighted by the fission rate of the isotopes, which can be reconstructed using information provided by the nuclear power plants (fig. 1.3).

JUNO detects electron antineutrinos via the Inverse Beta Decay (IBD) channel:

$$\bar{\nu_e} + p \rightarrow e^+ + n.$$

The positron deposits its kinetic energy (0-8 MeV range) in the liquid scintillator, then it quickly annihilates with an electron into two 0.511-MeV gammas, providing a prompt signal. The deposited energy in the detector is the sum of the kinetic energy of the positron and the annihilation energy: $E_{dep} = E^{e^+} + 1.022\,\text{MeV}$. As the antineutrino transfers most of its energy to the positron, $E^\nu \approx E^{e^+} + 1.8\,\text{MeV}$.

The neutron scatters in the detector until it is captured on hydrogen (99%) or carbon (1%) nuclei, producing respectively 2.2 MeV and 4.9 MeV de-excitation gammas. This signal comes with an average delay of $\sim 200\,\mu$s, which allows for the separation of IBD events from background.

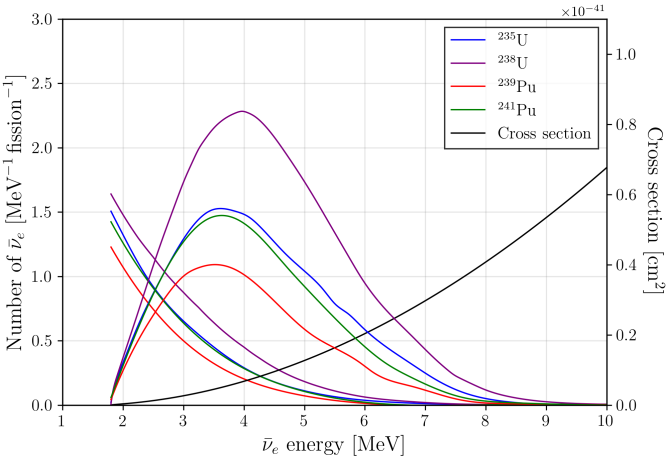


Figure 1.3: Antineutrino flux for the four main isotopes. IBD cross-section. Antineutrino energy spectrum (no oscillation) weighted by IBD cross-section [1].

The energy deposited in the detector can be measured through the scintillation mechanism: a material excited by ionizing radiation re-emits the absorbed energy in form of light that can be detected by PMTs. The active volume of the main detector is filled with Linear Alkyl Benzene (LAB), an organic solvent that emits 280 nm (UV) radiation upon γ excitation. As the signal requires a larger wavelength to be observable, wavelength shifter molecules like 2,5-diphenyloxazole (PPO), and 1,4-bis(2-methylstyryl)benzene (bis-MSB) [8] are needed for the energy to be re-emitted as visible photons. Finally, the produced light is collected by PMTs around the detector and converted to an electric signal which is then processed and digitized by the electronics.

Effects such as quenching in the liquid scintillator and Cherenkov radiation are responsible for complex non-linearities which affect energy reconstruction. The visible energy E_{vis}, reconstructed in this work, is intended as the scintillation part of the realized energy.

1.3 Energy Reconstruction

To determine the NMO at the level of 3 standard deviations in 6 years of data-taking, the JUNO detector requires an energy non-linearity uncertainty below 1% and an effective energy resolution better than $3\%/\sqrt{E(\mathrm{MeV})}$. To achieve this unprecedented resolution, stringent requirements must be enforced on the transparency of the scintillator and the detection efficiency of PMTs. A yield of over 1500 photoelectrons/MeV is obtained at the detector center in current simulations using nominal detector parameters [2]. This yield is position-dependent, and a calibration strategy based on the combined information provided by LPMTs and SPMTs is implemented to compensate for this non-uniformity in traditional reconstruction methods.

In general, the energy resolution [7] can be parameterized as:

$$\frac{\sigma_E}{E} = \sqrt{\left(\frac{a}{\sqrt{E}}\right)^2 + b^2 + \left(\frac{c}{E}\right)^2}$$

where a is mainly driven by the Poisson statistics of true accumulated charge on PMTs, b is independent of energy and accounts for positional non-uniformity, and c represents the contribution of dark noise. It was found numerically using mock neutrino energy spectra generated assuming different values of a, b and c that the JUNO requirement on the effective energy resolution can be rewritten in terms of a single parameter $\tilde{a}$ [7]:

$$\tilde{a} = \sqrt{(a)^2 + (1.6 \cdot b)^2 + \left(\frac{c}{1.6}\right)^2} < 3\%.$$

The charge and time information provided by PMTs can be used as input for various energy and vertex reconstruction algorithms. While many traditional approaches have been explored, from simply using charge-weighted PMTs positions to reconstruct the event vertex to complex likelihood methods, the underlying non-uniformities and non-linearities characterizing the experiment suggest a different possibility: Machine Learning.

Chapter 2

Machine Learning

2.1 Supervised Learning

The purpose of supervised learning is, given a large amount of input data for which the correct output is known, to build a model capable of mapping the input data in the corresponding output. This approach has experienced a rapid rise in popularity in High-Energy Physics (HEP), mostly due to the significant amount of labeled data produced in modern simulations. Several components are needed to successfully build, train and deploy a supervised learning algorithm:

- **Data**. A set $\{\mathbf{X}^{(i)}, y^{(i)}\}_{i=1,\dots,m}$, where for each *instance* i both the *features* matrix $\mathbf{X}^{(i)}$ containing all the information that will be used to make predictions and the true value of the *target* $y^{(i)}$ are known. As energy reconstruction is a regression task, y will be treated as a continuous variable.

- **Inductive Bias**. Training a Machine Learning algorithm equates to searching a hypothesis space $\mathbf{H}$ for the function $f : \mathbf{X} \to y$ that correctly maps the features in the target. The inductive bias represents the set of assumptions made about both data and the target function that make this search possible, and leads to the selection of the Machine Learning model to use.

- **Training**. While training algorithms are model-specific, in the common case of neural networks the selected model will depend on a set of *weights* $\mathbf{W}$, typically initialized randomly. The model is used to make predictions that are then compared to the true values of the target through a *loss function*, and the weights are updated in order to minimize the loss. The update rule is defined by an *optimizer*. In this process, each training sample passes through the model multiple times; each pass of the entire training dataset is defined as an *epoch*.

- **Validation**. The trained model is useful only if it is able to generalize well to data it has not yet seen. Failure to do so can be observed when the model either is too simple to fully capture the dependency of y on $\mathbf{X}$ (*underfitting*), or has learned by heart the training dataset, meaning that predictions on new data are influenced by random noise present in the training samples (*overfitting*). Because of these problems measures of performance on the training dataset are not reliable, and a validation dataset is needed to monitor the training process.
  In addition, Machine Learning models typically depend on various *hyperparameters* that can be tweaked based on the performance on validation data, in a process called *hyperparameter tuning*.

- **Testing**. As the choice of hyperparameters is optimized on the validation dataset, measures of performance on this dataset also become unreliable, especially after a lot of fine-tuning. An additional testing dataset is thus needed to provide a definitive measure of performance after which the model can no longer be tweaked.

2.2 Gradient Boosted Trees

Decision trees are a *white-box* Machine Learning model, meaning they make predictions by establishing a set of easily interpretable rules. The ID3 algorithm [9], typically used to generate decision trees, splits the dataset based on the feature that maximizes the information gain (or equivalently minimizes the entropy). It then reiterates on the different subsets using the remaining features until it either cannot decrease the entropy further or it reaches the maximum allowed depth for the tree, which is a hyperparameter set to prevent overfitting. This process defines a discrete set of branches leading from the root to a leaf-node, each representing a certain region $\mathbf{R_n}$ of the input space. In regression problems, given a new sample falling in $\mathbf{R_n}$, the algorithm assigns it the mean value of the target for all training samples in $\mathbf{R_n}$ as the predicted value. Notice how ID3 is a greedy algorithm: it searches for the best feature at the current iteration, generating a tree that is not guaranteed to be the global optimum. Indeed, the problem of learning the best decision tree is NP-complete [10].

Gradient boosting [11] is a popular algorithm in ensamble learning, and it works by sequentially adding new predictors to an ensamble, each one correcting its predecessor. In the context of regression problems, a decision tree is trained and used as base predictor. Then, a second tree is trained using as input the residual errors made by the first predictor, and this process is repeated until the ensamble grows to a predetermined size (although several regularization hyperparameters are available to prevent overfitting). Given a new sample at prediction time, the predictions of all trees are added (fig. 2.1). Gradient boosted models, in spite of their efficiency, are capable of achieving performances comparable to complex Deep Learning models on tabular data [12].

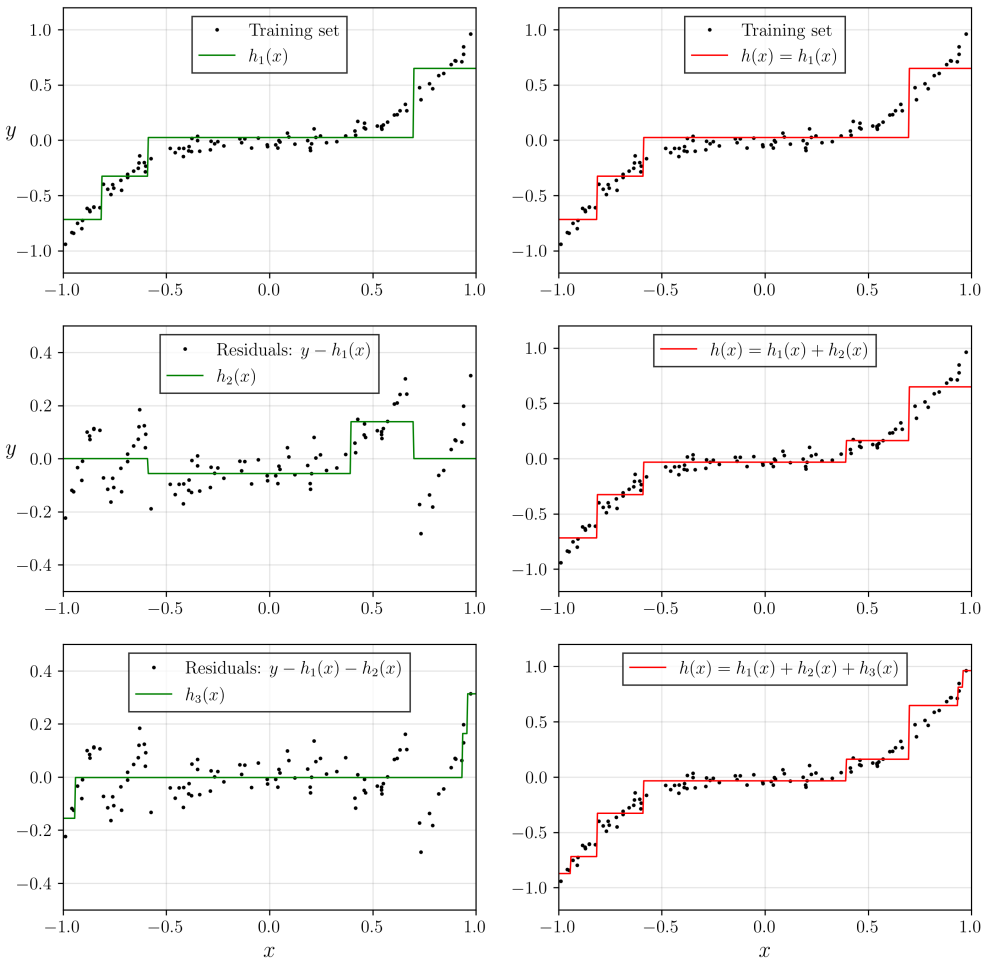
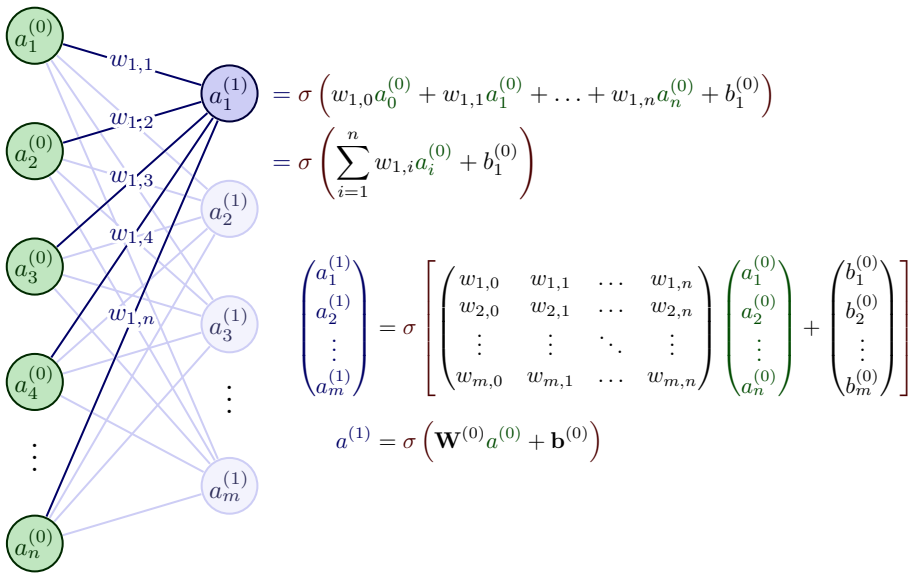


Figure 2.1: Depiction of gradient boosting for a simple regression problem. The first tree is trained normally (top left), then each subsequent tree is trained on the previous predictor's residuals (lower left). The right column displays the updated ensamble predictions.

2.3 Deep Neural Networks

Artificial neurons were first introduced in 1943 [13], and the simplest architecture for an artificial neural network (ANN), the perceptron, was invented in 1957 [14]. It is composed of one or more threshold logic units (TLU) organized in a single layer. Each TLU computes a linear combination of its inputs adding an offset, then it applies an activation function to the result:



where $a^{(0)}$ is the matrix of input features, $\mathbf{W}^{(0)}$ contains all the connection weights, $\mathbf{b}^{(0)}$ contains one bias term per unit and σ is the activation function, needed to introduce non-linearity. A fully connected deep neural network (FCDNN) is given by multiple perceptrons stacked on top of each other, creating several hidden layers before the final output layer, as seen in fig. 2.2.

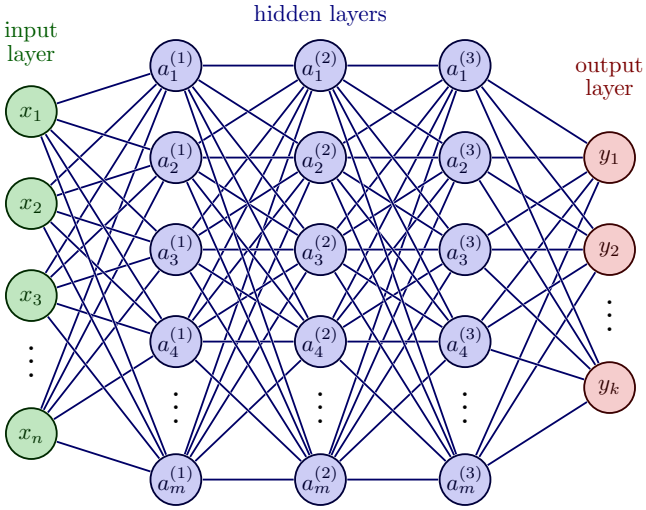


Figure 2.2: Fully connected deep neural network architecture.

Training of such complex architectures is made possible by a combination of reverse-mode automatic differentiation [15] and gradient descent. In the forward pass, the output of each layer is computed to measure the network's output error. Then the partial derivatives of the loss function with respect to each weight are estimated by working backward through the network and computing how much each connection contributed to the error in the following layer. Finally, the weights are updated by gradient descent.

Because of the inner complexities of these architectures there are many hyperparameters to tune: the number of hidden layers, the number of units per hidden layer, activation functions, weight initialization, optimizers, learning schedulers, etc.

2.4 Computing Details

In this work, ML models were trained on a virtual machine hosted on CloudVeneto and equipped with a NVIDIA T4 GPU. To benefit the most from the GPU acceleration provided by the CUDA hardware architecture, the following libraries for Python 3.10.6 were used:

- **XGBoost** [16] for tree-based models. It is a robust and efficient framework to train and deploy gradient boosted trees. GPUs are easily integrated via the `tree_method` parameter.
- **PyTorch** [17] for neural networks. It defines a class called Tensor (`torch.Tensor`) to store and operate on arrays of numbers. Tensors behave similarly to NumPy arrays, but they can easily be moved between the CPU and GPU and support highly optimized routines for linear algebra and gradient computation, making them particularly suited to train deep neural networks.

Chapter 3

Data Analysis

3.1 Data Description

The analyzed dataset has been generated using the full detector Monte Carlo method provided by the official JUNO software [18–20], which is based on the Geant4 framework [21, 22]. Each simulated event begins with the injection of a positron with given kinetic energy, while the neutron produced in the IBD interaction is not simulated. The dataset has then been split for the purposes of training and model evaluation:

- **Training dataset**: $2 \cdot 10^6$ events spread uniformly in the scintillator volume with isotropic angular distribution. The kinetic energy of the simulated positron has been sampled uniformly in the [0, 10] MeV range.
- **Testing dataset**: 14 subsets of $5 \cdot 10^4$ events each corresponding to the following values of the kinetic energies of the positron: [0, 0.1, 0.3, 0.6, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10] MeV. This allows for evaluation at different energies.

The following details should be noted:

- Both the information provided by 20-inch PMTs and 3-inch PMTs have been used for energy reconstruction. While the contribution of the latter to the total light collection is small, this serves as a useful comparison with previous studies [23] which considered only large PMTs.
- PMTs produce pulses spontaneously causing noise (dark current). Of the 17612 LPMTs, 25% are produced by Hamamatsu and 75% by NNVT, with dark current rate measured at 19.3 kHz and 49.3 kHz respectively [24]. This effect, together with other intrinsic noise in the electronics, has been simulated.
- For events happening near the edge of the detector, one or both photons produced in the electron-positron annihilation can escape without contributing to the light yield. In addition, the radioactive background is particularly strong at the detector's edge. To counter for these effects affecting energy reconstruction, a volume cut at 17.2 m has been applied, removing from the dataset all events in the outer 0.5 m layer of the detector.

3.2 Feature Engineering

Given a simulated event, both the accumulated charge and the first hit time (FHT) are known for each PMT (fig. 3.1). This results in 86424 information channels (many of which empty) that can be used for energy reconstruction. Training of neural networks on such a large number of features is computationally expensive, so the purpose of this study is to aggregate the available information in a small set of features that will be used as input to simpler Machine Learning models with the purpose of reconstructing the visible energy E_{vis}.

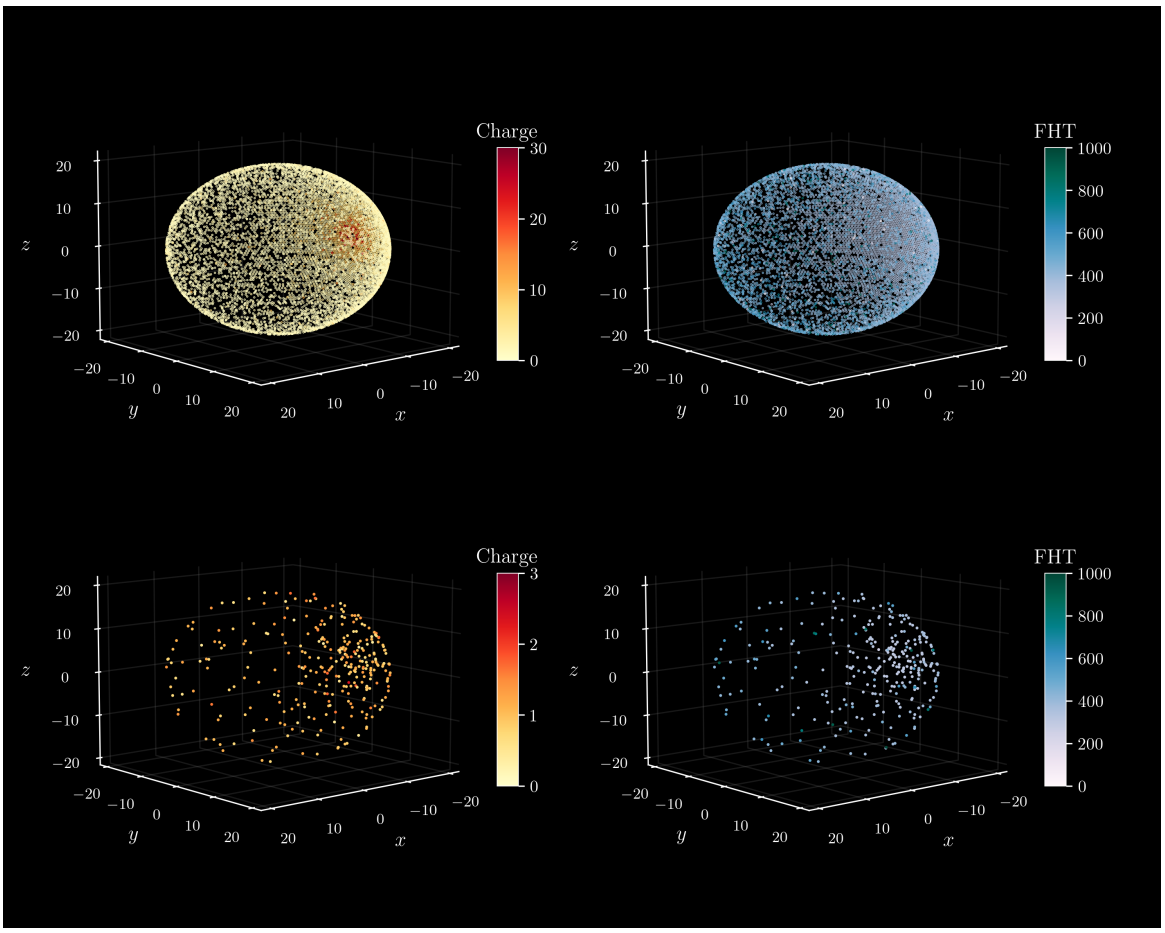


Figure 3.1: Accumulated charge in the trigger window and first hit time for the same event as seen by LPMTs (above) and SPMTs (below). Only fired PMTs are shown.

The following features have been engineered:

1. `AccumCharge`: sum of the charges accumulated by all PMTs in the trigger window. It is linearly correlated with E_{vis} and thus expected to be a powerful feature. It is also strongly position dependent: in the region $R \gtrsim 16$ m it decreases (fig 3.2) because of the total internal reflection, due to which photons with a large enough incident angle never reach the PMTs.

2. `nPMTs`: number of fired PMTs weighted by surface area of the PMT. It is expected to be similar to `AccumCharge` but its correlation with E_{vis} is slightly less linear.

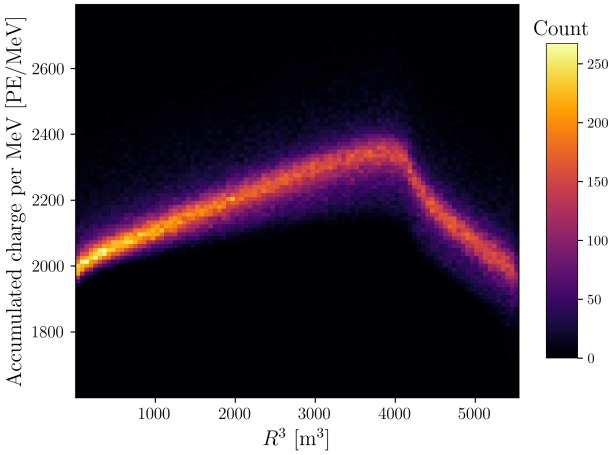


Figure 3.2: `AccumCharge` on PMTs per 1 MeV of deposited energy against event radius cubed.

3. Features aggregated given the charge and position of the PMTs:

   (a) Cartesian coordinates of the center of charge:

$$(x_{cc},\, y_{cc},\, z_{cc}) = \vec{r}_{cc} = \frac{\sum_{i=1}^{N_{LMPTs}} \vec{r}_{LPMT_i} \cdot n_{pe,i} + \sum_{i=1}^{N_{SMPTs}} \vec{r}_{SPMT_i} \cdot n_{pe,i}}{\sum_{i=1}^{N_{LMPTs}} n_{pe,i} + \sum_{i=1}^{N_{SMPTs}} n_{pe,i}}$$

   and its radial component $R_{cc} = |\vec{r}_{cc}|$. As JUNO's spherical symmetry is broken along the z axis because of the supporting structure which obstructs the installation of some PMTs in the bottom hemisphere, z_{cc} is expected to be particularly important.
   While these features are technically enough to characterize the center of charge, the following others are engineered to help ML models learn complex non-linear dependencies:

$$\theta_{cc} = \arctan \frac{\sqrt{x_{cc}^2 + y_{cc}^2}}{z_{cc}}, \quad \phi_{cc} = \arctan \frac{y_{cc}}{x_{cc}}, \quad J_{cc} = R_{cc}^2 \cdot \sin\theta_{cc}, \quad \rho_{cc} = \sqrt{x_{cc}^2 + y_{cc}^2}$$

$$\gamma_x^{cc} = \frac{x_{cc}}{\sqrt{y_{cc}^2 + z_{cc}^2}}, \quad \gamma_y^{cc} = \frac{y_{cc}}{\sqrt{x_{cc}^2 + z_{cc}^2}}, \quad \gamma_z^{cc} = \frac{z_{cc}}{\sqrt{x_{cc}^2 + y_{cc}^2}}$$

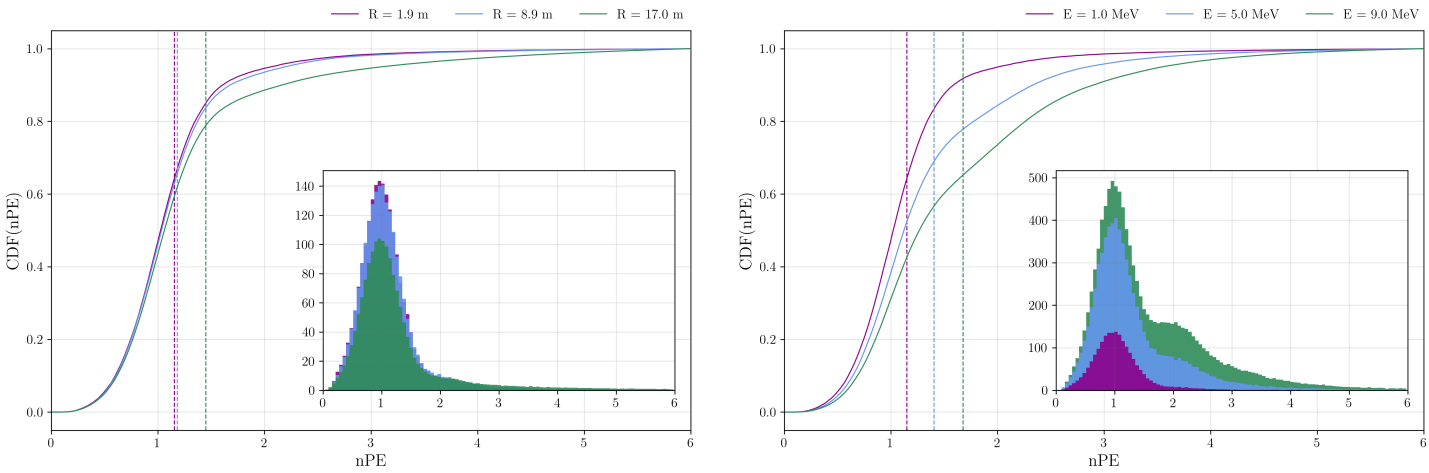


Figure 3.3: CDFs and PDFs of LPMTs charge distribution for events with $E_{kin} \approx 1$ MeV and different radii (left) and events roughly at the detector's center but with different energies (right). Dashed lines indicate mean values of the PDFs.

   (b) Charge distribution for LPMTs.
   As it can be seen from fig. 3.3, the shape of the charge distribution captures information about both the intensity of light emission (for higher energies the peak at two hits per PMT becomes more prominent) and the position (near the edge of the detector the distribution becomes more right-tailed). To characterize the distribution the following percentiles are used:

$$\{pe_{2\%},\, pe_{5\%},\, pe_{10\%},\, pe_{15\%},\, \ldots,\, pe_{90\%},\, pe_{95\%}\}$$

   in addition to the distribution's mean, standard deviation, skewness, kurtosis and entropy:

$$\{pe_{mean},\, pe_{std},\, pe_{skew},\, pe_{kurtosis},\, pe_{entropy}\}$$

   (c) Charge distribution for SPMTs.
   The same set of features described in (b) is engineered separately for SMPTs.

4. Features aggregated given the first hit time and position of the PMTs:

(a) Cartesian coordinates of the FHT center:

$$(x_{cht},\, y_{cht},\, z_{cht}) = \vec{r}_{cht} = \frac{\sum_{i=1}^{N_{LMPTs}} \frac{\vec{r}_{LPMT_i}}{t_{ht,i}+c} + \sum_{i=1}^{N_{SMPTs}} \frac{\vec{r}_{SPMT_i}}{t_{ht,i}+c}}{\sum_{i=1}^{N_{LMPTs}} \frac{1}{t_{ht,i}+c} + \sum_{i=1}^{N_{SMPTs}} \frac{1}{t_{ht,i}+c}}$$

and its radial component $R_{cht} = |\vec{r}_{cht}|$. Here c is a constant (50 ns) needed to avoid division by zero. As it can be seen from fig. 3.4, the correlation between the radial components of the center of charge and the center of first hit time is non-linear, especially for large event radii, thus bringing complementary information.

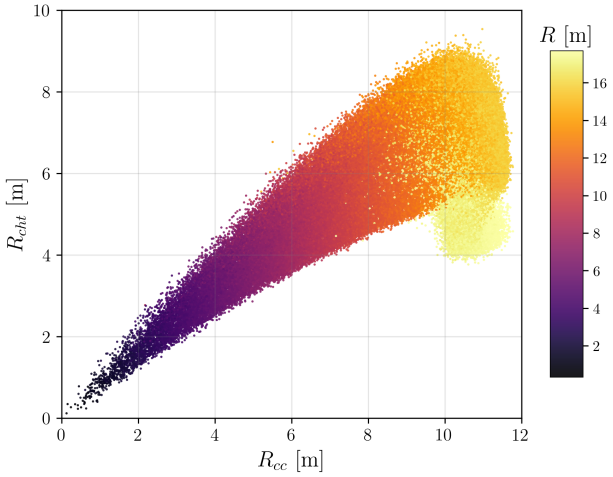


Figure 3.4: Correlation between R_{cc} and R_{cht}, colored by the radial component of the event vertex.

As it was done for the center of charge, additional features are engineered:

$$\theta_{cht} = \arctan\frac{\sqrt{x_{cht}^2 + y_{cht}^2}}{z_{cht}},\quad \phi_{cht} = \arctan\frac{y_{cht}}{x_{cht}},\quad J_{cht} = R_{cht}^2 \cdot \sin\theta_{cht},\quad \rho_{cht} = \sqrt{x_{cht}^2 + y_{cht}^2}$$

$$\gamma_x^{cht} = \frac{x_{cht}}{\sqrt{y_{cht}^2 + z_{cht}^2}},\quad \gamma_y^{cht} = \frac{y_{cht}}{\sqrt{x_{cht}^2 + z_{cht}^2}},\quad \gamma_z^{cht} = \frac{z_{cht}}{\sqrt{x_{cht}^2 + y_{cht}^2}}$$

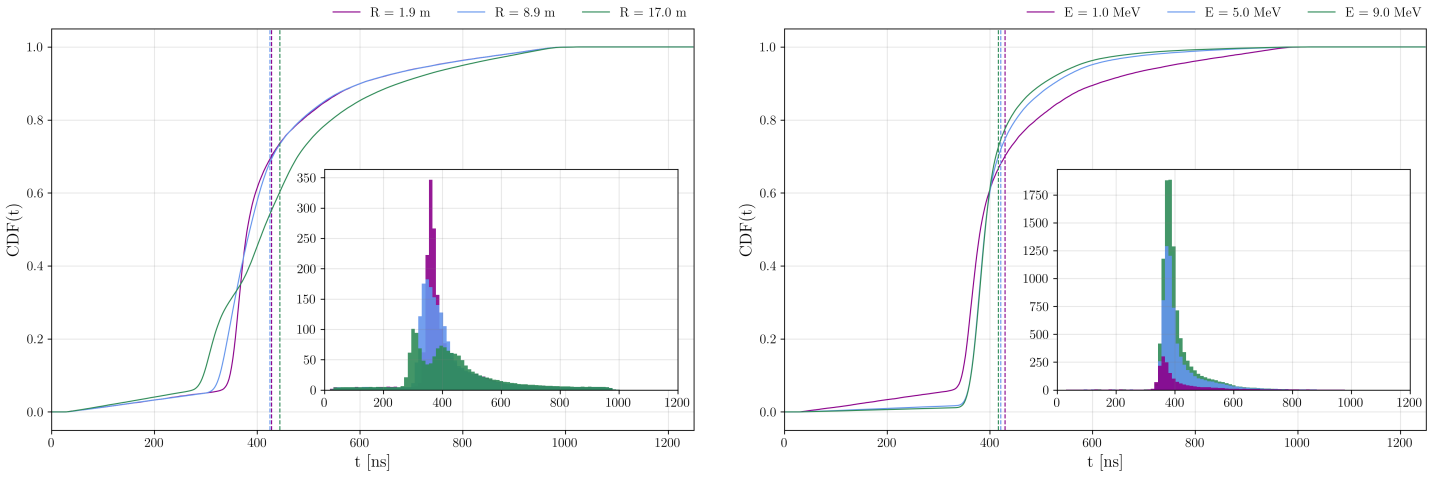


Figure 3.5: CDFs and PDFs of LPMTs FHT distribution for events with $E_{kin} \approx 1$ MeV and different radii (left) and events roughly at the detector's center but with different energies (right). Dashed lines indicate mean values of the PDFs.

(b) FHT distribution for LPMTs.
Once again there is dependence on both the energy and radial position (see fig 3.5). Near the detector's edge photons are detected earlier due to the lower time-of-flight correction and a depression in the distribution is symptomatic of the effects of internal reflections. At the detector's center distributions are similar in shape, differing mostly by the relative contribution from dark noise and annihilation gammas, which is higher at lower energies. We characterize the distribution with the same features used above:

$$\{ht_{2\%},\, ht_{5\%},\, ht_{10\%},\, ht_{15\%},\, \ldots,\, ht_{90\%},\, ht_{95\%}\}$$

$$\{ht_{mean},\, ht_{std},\, ht_{skew},\, ht_{kurtosis},\, ht_{entropy}\}$$

From fig. 3.6 it can be noted how the FHT distribution displays high variance in the bins leading to its peak. To account for this effect, differences between adjacent percentiles have also been introduced as features, as they are more robust to variance:

$$\{ht_{5\%-2\%},\, ht_{10\%-5\%},\, \ldots,\, ht_{95\%-90\%}\}$$

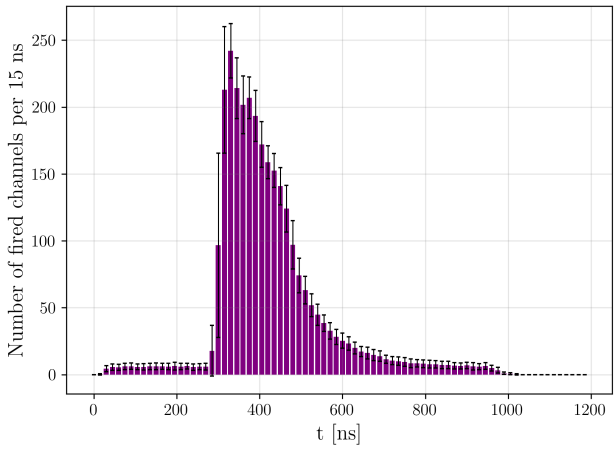


Figure 3.6: Average PDF for the FHT distribution using events with $E_{kin} \approx 2$ MeV and $R \approx 16$ m.

(c) FHT distribution for SPMTs.
The same set of features described in (b) is engineered separately for SMPTs.

In total 162 features have been engineered, drastically reducing the number of channels needed for energy reconstruction. Many of these features are highly correlated, so further reduction is possible via feature selection.

3.3 Feature Selection

The purpose of feature selection is to obtain the smallest subset of features capable of providing the same performance as when the model is trained on the entire set. As this step is computationally expensive, boosted decision trees have been chosen as the baseline model. The developed algorithm works as follows:

1. Firstly it trains a BDT using all the engineered features. To get a more reliable estimate of the model's performance on data it has not yet seen, 5-fold cross-validation is implemented. The performance on an additional hold-out dataset is monitored to stop training when the validation loss starts increasing (early stopping). The performance metric chosen is the mean absolute percentage error (MAPE):

$$\text{MAPE} = \frac{100\%}{N} \sum_{i=1}^{N} \left| \frac{y_i - \hat{y}_i}{y_i} \right|$$

as it is less forgiving than the root mean squared error (RMSE) when dealing with lower energies. Because of cross-validation, both the mean and standard deviation of the metric are known.

2. Then it loops over all the features, finds the one with the best performance on its own (unsurprisingly it is `AccumCharge`) and adds it to an empty list of features as its first element.

3. Lastly it loops over all the remaining features, finds the one providing the largest step-wise gain in performance and adds it to the list. The process is repeated recursively until the MAPE becomes statistically compatible with the value found when training on all features.

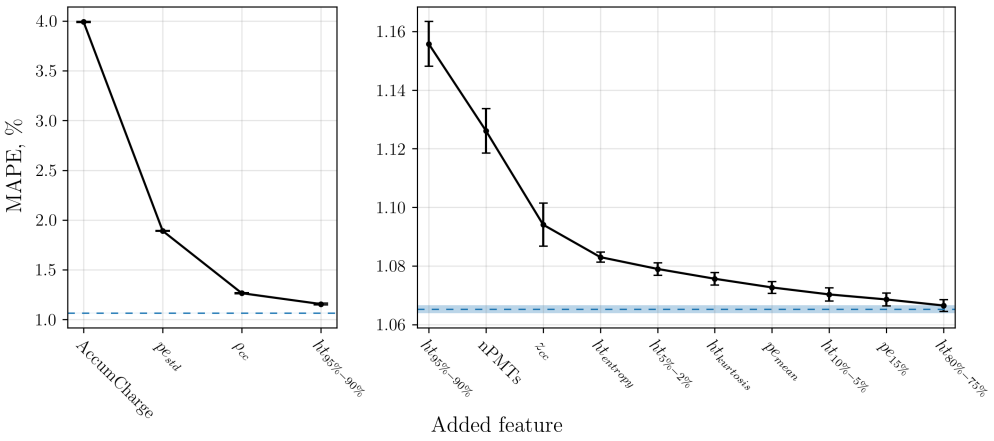


Figure 3.7: The feature selection process. In blue the MAPE when training on all features.

The following 13 features are selected (see fig. 3.7):

1. `AccumCharge`
2. pe_{std}
3. ρ_{cc}
4. $ht_{95\%-90\%}$
5. `nPMTs`
6. z_{cc}
7. $ht_{entropy}$
8. $ht_{5\%-2\%}$
9. $ht_{kurtosis}$
10. pe_{mean}
11. $ht_{10\%-5\%}$
12. $pe_{15\%}$
13. $ht_{80\%-75\%}$

Many features characterizing the shape of the charge and time profiles prove to be important, among which the entropy of the first hit time distribution, which was not present in previous studies [23]. As expected, none of the features regarding the charge and time profiles for SPMTs made it to the top, but SPMTs still affect `AccumCharge`, ρ_{cc}, `nPMTs` and z_{cc} and may thus provide useful information.

BDTs are trained setting the maximal tree depth to 8 and subsampling to 0.8 for regularization purposes. The learning rate was set to 0.08 and the early stopping condition to a patience of 5.

3.4 Hyperparameter Tuning

3.4.1 BDT

Now that the most important features have been selected, hyperparameter tuning is performed on BDTs trained with 5-fold cross-validation and early stopping on an additional hold-out dataset. The implementation of the Tree-Structured Parzen Estimator algorithm available in the Optuna library [25] is used to search for an optimum in the hyperparameter space (see fig. 3.8 and tab. 3.1).

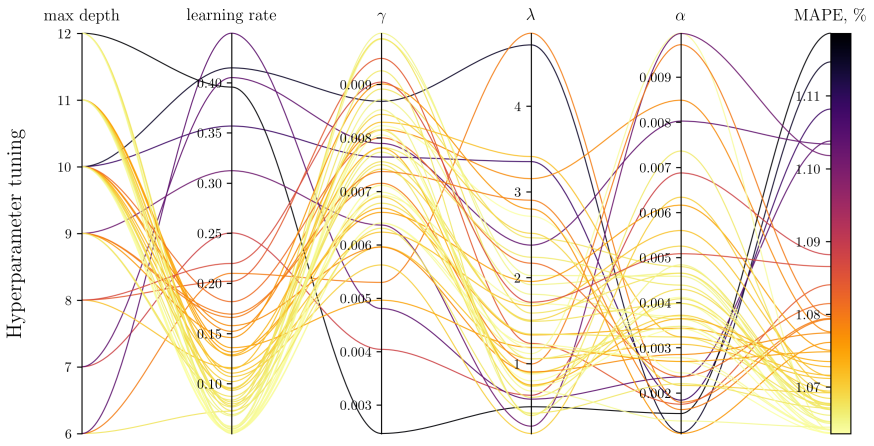


Figure 3.8: Visualization of 50 attempts in the hyperparameter search, colored by MAPE.

Hyperparameter	*Range*	*Optimum*
Max depth of trees	[6, 12]	**10**
Learning rate	[0.05, 0.5]	**0.05**
γ	[0.001, 0.01]	**0.008**
λ	[0.1, 5]	**0.7**
α	[0.001, 0.01]	**0.003**

Table 3.1: Hyperparameter search space and results.

The best model is finally trained and SHAP (SHapley Additive exPlanations) values [26] are computed to assess the contribution of each feature to the predictions of the model. As it can be seen from fig. 3.9, although the order is slightly different from that returned by the feature selection algorithm, the most important features are confirmed.

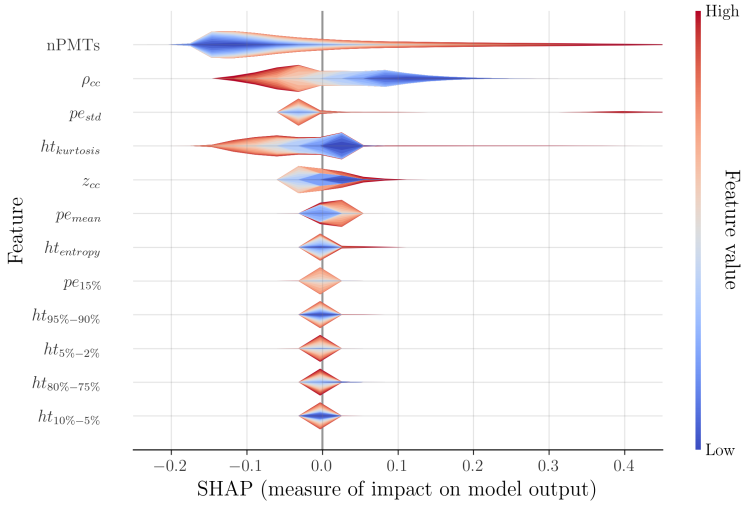


Figure 3.9: SHAP values distributions for all features. `AccumCharge` is not displayed as its impact is much larger (SHAP > 2).

3.4.2 FCDNN

Fully connected deep neural networks (FCDNN) can also be effective when working with tabular data, but they are generally more computationally expensive than tree-based models and require a lot of hyperparameter tuning. To compensate for that, a single validation set is used instead of a 5-fold cross-validation. In addition, the Median Pruner algorithm from the Optuna library [25] is

implemented to help with the Bayesian optimization of hyperparameters by discarding unpromising trials. Training is performed with an early stopping condition on the validation set with a patience of 50 epochs and with the batch size set to 1024.

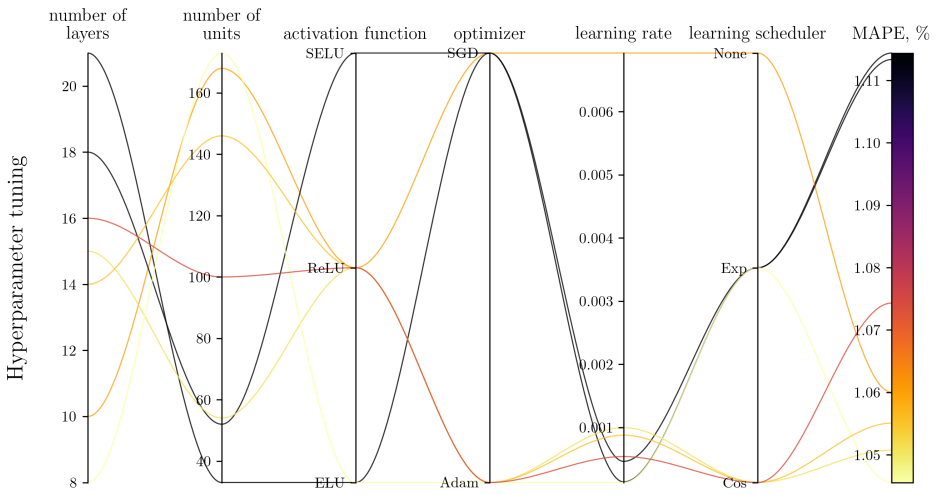


Figure 3.10: Visualization after 200 attempts (many are pruned) in the hyperparameter search, colored by MAPE.

Hyperparameter	*Range*	*Optimum*
Number of hidden layers	[6, 24]	**8**
Number of units in hidden layers	[1, 256]	**173**
Activation function	[ReLU, ELU, SELU]	**ELU**
Optimizer	[Adam, SGD, RMSprop]	**Adam**
Learning rate	[0.0001, 0.01]	**0.00013**
Learning scheduler	[None, Cos, Exp]	**Exp**

Table 3.2: Hyperparameter search space and results.

As it can be seen from fig. 3.10, the combination of Bayesian optimization and pruning allows for a very efficient search in the hyperparameter space, with only 7 trials needed to converge to an optimum. The best trial (see tab. 3.2) is trained until the early stopping condition triggers after 113 epochs, when both the MAPE and MSE curves have converged to similar values for the training and validation set, indicating a good balance in the bias/variance trade-off.

3.4.3 1DCNN

A more exotic approach consists in using a 1-dimensional convolutional neural network (1DCNN). CNNs are widely used in computer vision, as they are capable of automatically extracting through filters local features in image data which can then be fed to fully connected layers. Such networks are rarely used on tabular data as there are no locality characteristics (the ordering of columns is arbitrary, see fig. 3.11). However, a fully connected layer can be used to let the model learn on its own a spacial representation of the features on which the convolutional layers can then work on [12].

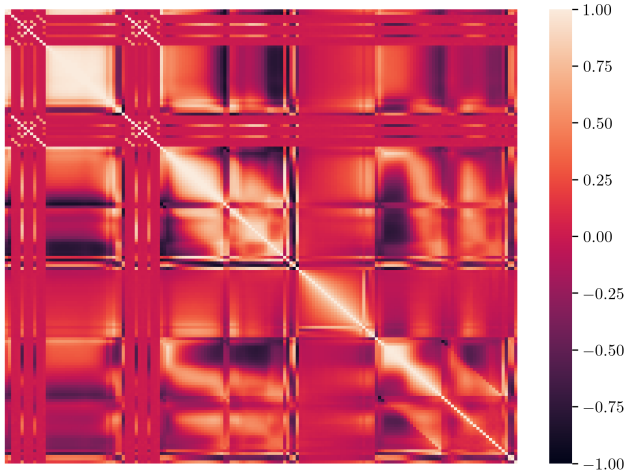


Figure 3.11: Correlation heatmap of the 162 engineered features.

As this model is significantly more complex than the previous ones, it benefits from using all the 162 features available and requires multiple regularization techniques to prevent overfitting and improve stability, including batch normalization, weight normalization and dropout layers. In addition, a shortcut is added to the architecture: after the second convolutional layer, propagation proceeds through either two consecutive convolutions or directly to the last pooling layer (see fig. 3.12). The shorter connection is clearly less prone to overfitting. Lastly, due to the computational complexity of the model the architecture is kept fixed and only hyperparameters related to the training process are tuned.

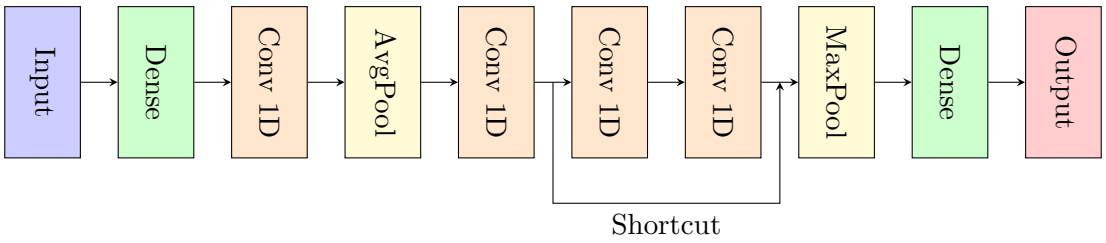


Figure 3.12: 1DCNN architecture.

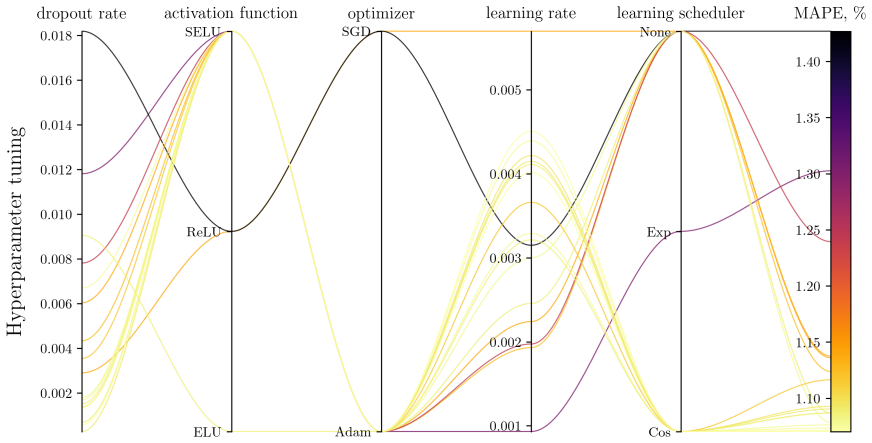


Figure 3.13: Visualization after 500 attempts (many are pruned) in the hyperparameter search, colored by MAPE.

Hyperparameter	*Range*	*Optimum*
Dropout rate	[0, 0.05]	**0.0006**
Activation function	[ReLU, ELU, SELU]	**SELU**
Optimizer	[Adam, SGD, RMSprop]	**Adam**
Learning rate	[0.0001, 0.01]	**0.005**
Learning scheduler	[None, Cos, Exp]	**Cos**

Table 3.3: Hyperparameter search space and results.

Fig. 3.13 displays Bayesian optimization in action: SELU [27] and Adam [28] are immediately identified as the best activation function and optimizer respectively, and following trials sample a smaller region of the hyperparameter space keeping these hyperparameters fixed.

The best trial (see tab. 3.3) is trained until the early stopping condition triggers after 97 epochs. It should be noted that both validation MAPE and validation MSE exhibit significant fluctuations during training, causing the early stopping condition to be unreliable in identifying the moment the model starts overfitting. This may be the cause of the deterioration in performance at higher energies discussed in 3.5. In addition, the asymptotic limit of the validation loss is lower than the corresponding limit for the training loss, although this can partly be attributed to the presence of dropout layers, which are not active at prediction time as they serve only for regularization purposes. Suggestions for future improvement are discussed in 4.

3.5 Results

The best models returned by the hyperparameter tuning process are now ready for evaluation. For every subset of the testing dataset, characterized by a distribution of events peaked around a specific value of the visible energy, a Gaussian fit of the distribution of the difference between true and predicted energy ($E_{vis} - E_{pred}$) is performed. Bias and resolution can then be computed based on the results of this fit:

$$\text{Bias, } \% = 100 \cdot \frac{\mu}{E_{vis}} \qquad \text{Resolution, } \% = 100 \cdot \frac{\sigma}{E_{vis}}$$

with μ and σ mean and standard deviation estimated by the Guassian fit. Subsets with kinetic energy of the simulated positron equal to 0 MeV and 10 MeV are excluded from the analysis, as ML models learn to expect $E_{kin} \in 0\ -10\,\text{MeV}$ causing distributions at the edge points to deviate from normality and exhibit systematically higher resolution (see fig. 3.14).

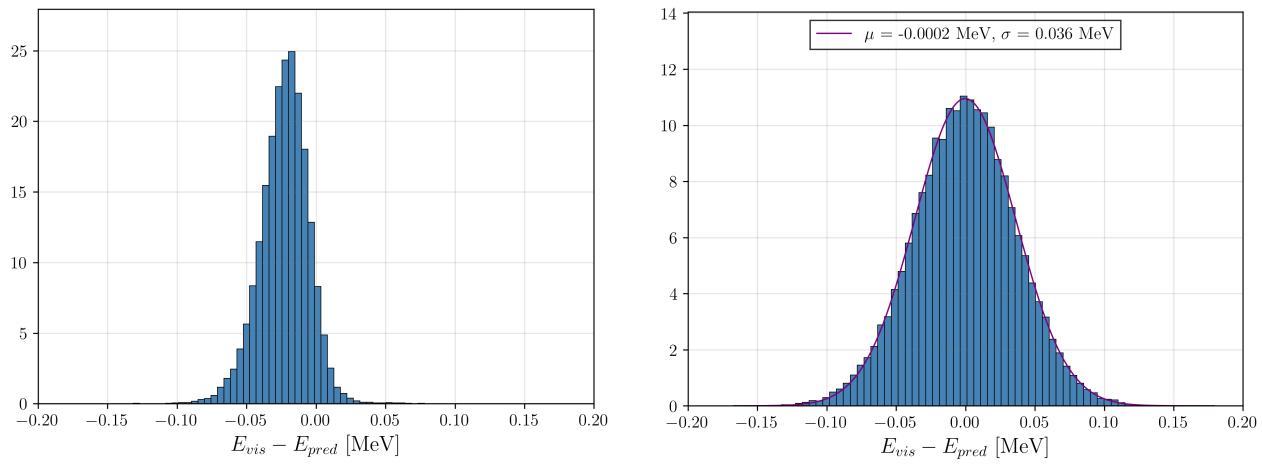


Figure 3.14: Examples of distributions of the difference between true and predicted energy for BDT. On the left $E_{kin} = 0\,\text{MeV}$ with the described edge effect, on the right $E_{kin} = 1\,\text{MeV}$.

.

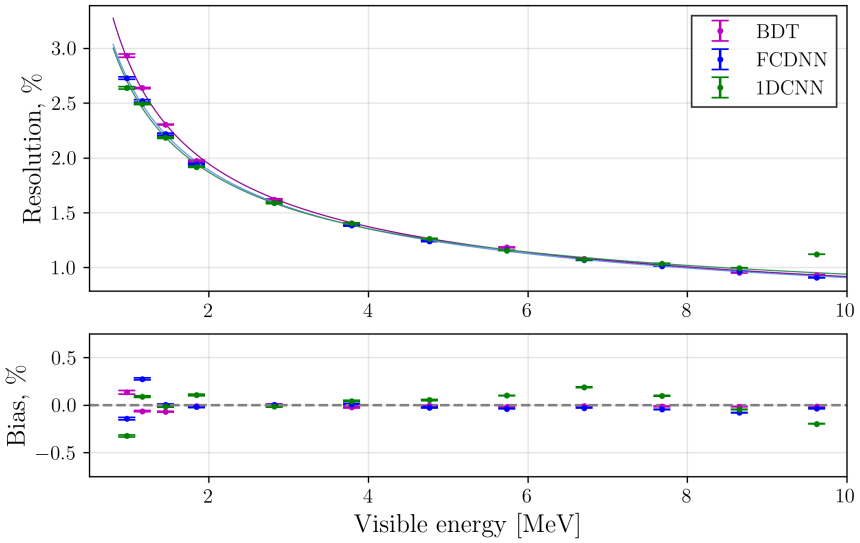


Figure 3.15: Percentage resolution (top) and bias (bottom) for BDT, FCDNN, 1DCNN on the different subsets of the testing dataset.

As it can be seen in fig. 3.15, excluding occasional outliers at lower energies, possibly still affected by the edge problem described above, the energy reconstruction bias for BDT and FCDNN is statistically compatible with zero. On the other hand, 1DCNN's reconstruction bias, while still small, presents a systematic trend at higher energies, possibly due to overfitting of the training dataset.

1DCNN has the lowest resolution at lower energies but deteriorates at higher energies with a significant outlier for $E_{kin} = 9\,\text{MeV}$, which has been excluded from the fit. FCDNN and BDT are more consistent, with the former performing better at nearly all energies.

Parameter \ *Model*	BDT	FCDNN	1DCNN
$a \pm \sigma_a$	2.524 ± 0.091	2.550 ± 0.038	2.470 ± 0.079
$b \pm \sigma_b$	0.433 ± 0.066	0.409 ± 0.028	0.519 ± 0.048
$c \pm \sigma_c$	1.28 ± 0.21	0.77 ± 0.16	0.87 ± 0.25
$\tilde{a} \pm \sigma_{\tilde{a}}$	2.737 ± 0.023	2.675 ± 0.011	2.661 ± 0.022

Table 3.4: Energy resolution parameterization fit results for BDT, FCDNN and 1DCNN.

Energy resolution is fitted for all models according to the parameterizations described in 1.3 (see tab. 3.4). 1DCNN exhibits the lowest effective resolution $\tilde{a}$, but because of the large error the estimate is statistically compatible with the value for FCDNN. What can be said with certainty is that neural networks achieve lower effective resolutions than BDT and all models satisfy the requirement to determine the neutrino mass ordering: $\tilde{a} < 3\%$.

Chapter 4

Conclusions

In this work, three different Machine Learning models have been trained on aggregated features extracted from Monte Carlo simulation data to reconstruct the visible energy in the JUNO detector: boosted decision trees (BDT), a fully connected deep neural network (FCDNN) and a 1-dimensional convolutional neural network (1DCNN). While all models satisfy the requirement on the effective energy resolution to determine the neutrino mass ordering ($\tilde{a} < 3\%$), FCDNN provides the best combination of performance and stability. On the other hand, decision trees train approximately 100 times faster and perform better on smaller datasets, making them useful for calibration on sparse data. Finally, 1DCNN achieves statistically compatible performance to FCDNN while training twice as fast and is the best candidate for future improvements:

- cross-validation can be implemented to offer more reliable estimates of performance on validation data and thus improve stability;
- the early stopping condition can be set on a trailing average of the validation loss, reducing the dependence on random noise in validation data;
- redundant features may prevent the model from learning a useful representation of the feature space. Principal Component Analysis (PCA) and other unsupervised learning algorithms could filter for a predetermined number of features to use, leading to faster training times and possibly better performance.

Bibliography

[1] F. An et al., "Neutrino Physics with JUNO", J. Phys. G **43**, 030401 (2016).

[2] A. Abusleme et al., "JUNO physics and detector", Prog. Part. Nucl. Phys. **123**, 103927 (2022).

[3] Y. Fukuda et al., "Evidence for oscillation of atmospheric neutrinos", Phys. Rev. Lett. **81**, 1562–1567 (1998).

[4] Z. Maki, M. Nakagawa, and S. Sakata, "Remarks on the unified model of elementary particles", Prog. Theor. Phys. **28**, 870–880 (1962).

[5] B. Pontecorvo, "Neutrino Experiments and the Problem of Conservation of Leptonic Charge", Zh. Eksp. Teor. Fiz. **53**, 1717–1725 (1967).

[6] R. L. Workman and Others, "Review of Particle Physics", PTEP **2022**, 083C01 (2022).

[7] A. Abusleme et al., "Calibration Strategy of the JUNO Experiment", JHEP **03**, 004 (2021).

[8] A. Abusleme et al., "Optimization of the JUNO liquid scintillator composition using a Daya Bay antineutrino detector", Nucl. Instrum. Meth. A **988**, 164823 (2021).

[9] J. R. Quinlan, "Induction of decision trees", Mach. Learn. **1**, 81–106 (1986).

[10] L. Hyafil and R. L. Rivest, "Constructing optimal binary decision trees is np-complete", Information Processing Letters **5**, 15–17 (1976).

[11] J. Friedman, "Greedy function approximation: a gradient boosting machine", The Annals of Statistics **29** (2000).

[12] R. Shwartz-Ziv and A. Armon, "Tabular data: deep learning is not all you need", Information Fusion **81**, 84–90 (2022).

[13] W. S. Mcculloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity", Journal of Symbolic Logic **9**, 49–50 (1943).

[14] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain.", Psychological review **65**, 386 (1958).

[15] S. Linnainmaa, "Taylor expansion of the accumulated rounding error", BIT **16**, 146–160 (1976).

[16] T. Chen and C. Guestrin, "XGBoost", in Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining (2016).

[17] A. Paszke et al., "Pytorch: an imperative style, high-performance deep learning library", in Proceedings of the 33rd international conference on neural information processing systems (2019).

[18] X. Huang et al., "Offline Data Processing Software for the JUNO Experiment", PoS, 1051 (2017).

[19] T. Lin et al., "The Application of SNiPER to the JUNO Simulation", J. Phys. Conf. Ser. **898**, edited by R. Mount and C. Tull, 042029 (2017).

[20] T. Lin et al., "Parallelized JUNO simulation software based on SNiPER", J. Phys. Conf. Ser. **1085**, 032048 (2018).

[21] S. Agostinelli et al., "GEANT4–a simulation toolkit", Nucl. Instrum. Meth. A **506**, 250–303 (2003).

[22] J. Allison et al., "Recent developments in Geant4", Nucl. Instrum. Meth. A **835**, 186–225 (2016).

[23] A. Gavrikov, Y. Malyshkin, and F. Ratnikov, "Energy reconstruction for large liquid scintillator detectors with machine learning techniques: aggregated features approach", Eur. Phys. J. C **82**, 1021 (2022).

[24] A. Abusleme et al., "Mass testing and characterization of 20-inch PMTs for JUNO", Eur. Phys. J. C **82**, 1168 (2022).

[25] T. Akiba et al., "Optuna: a next-generation hyperparameter optimization framework", in Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery and data mining, KDD'19 (2019), 2623–2631.

[26] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions", in Proceedings of the 31st international conference on neural information processing systems, NIPS'17 (2017), 4768–4777.

[27] G. Klambauer et al., "Self-normalizing neural networks", in Proceedings of the 31st international conference on neural information processing systems, NIPS'17 (2017), 972–981.

[28] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization", in Proceedings of the 3rd international conference on learning representations (2015).