Eberhard Karls Universität Tübingen

Master's thesis

Artificial Neural Networks and Machine Learning Algorithms for On-board Event Analysis of High Energy Observations

Christopher Weinert

prepared in the degree program
Astro & Particle Physics

Institut für Astronomie und Astrophysik Tübingen
High Energy Astrophysics Group

First supervisor: Prof. Dott. Andrea Santangelo
Second supervisor: Dr. Jan-Christoph Tenzer

Winter semester 2024/2025

Declaration of originality

Hereby I confirm, Christopher Weinert, Matr. Nr. 6274476, that this assignment is my own work and that I have only sought and used mentioned tools. I have clearly referenced in the text and the bibliography all sources used in the work (printed sources, internet or any other source), including verbatim citations or paraphrases. I am aware of the fact that plagiarism is an attempt to deceit which, in case of recurrence, can result in a loss of test authorization. Furthermore, I confirm that neither this work nor parts of it have been previously, or concurrently, used as an exam work – neither for other courses nor within other exam processes.

Place and date ____________________ Signature ____________________

Abstract

The rapid advancement of Artificial Intelligence (AI) has driven its exploration and integration into scientific research and industrial workflows. Among these endeavors is the i-RASE project, initiated by the Technical University of Denmark, which seeks to develop the first high energy radiation detector capable of real-time analysis using AI. The University of Tübingen contributes to this project by providing the detector's electronic system, specifically an FPGA designed to operate the trained AI algorithms.

This thesis investigates the inherent question in this project whether AI can effectively replace traditional methodologies in detector data analysis. A range of Machine Learning (ML) and Deep Learning (DL) methods were evaluated, including Random Forests, Convolutional Neural Networks (CNNs), and Long Short-Term Memory (LSTM) networks. Training and testing data were derived from experimental recordings of a high energy detector, and the models were tasked with evaluating the detector data and predicting its properties. These properties included the energy, radiation and interaction type of the incident particles, and the drift time of the charge carriers within the detector's semiconductor, and the subpixel resolution of the impact point on the pixelated surface of the detector.

This thesis provides a comprehensive insight into the design and training of the AI models. The fully trained models demonstrated promising and robust performance, particularly the DL approaches, in estimating the specified radiation properties. Future work will focus on integrating the optimal AI models onto the FPGA for operational testing.

Contents

1 Introduction **9**

2 Cd(Zn)Te detectors **11**

3 Artificial Intelligence **19**
3.1 Classification and Regression . . . 21
3.2 Machine Learning . . . 25
3.2.1 Decision Tree . . . 26
3.2.2 Random Forest . . . 28
3.2.3 Gradient Boosting Tree . . . 29
3.3 Deep Learning . . . 33
3.3.1 Loss function . . . 37
3.3.2 Model training . . . 40
3.3.3 Convolutional Neural Network . . . 43
3.3.4 Memory-type Networks . . . 45

4 Experimental framework & methodology **49**
4.1 High energy radiation recordings . . . 49
4.2 Task formulation and principle justification . . . 56
4.3 Models' architecture . . . 58
4.4 Energy . . . 62
4.5 Drift time . . . 69
4.6 Pixel pattern . . . 77
4.7 Subpixel resolution . . . 85

5 Conclusion **93**

6 Outlook **97**

1 Introduction

At the time of authoring this thesis, Artificial Intelligence (AI) has begun to massively influence our daily lives and has found a wide variety of applications. It has paved its way into science, engineering, information technologies, healthcare, telecommunication, education and even arts and entertainment. Machine Learning (ML) and/or Deep Learning (DL) – as subcategories of Artificial Intelligence – has set its place in our society with an on-going steady rise in popularity.

Autonomous driving, self-operating robots and machines, the understanding of sophistic language and being able to grasp its interrelations, the creation of high resolution videos and pictures out of nothingness, as well as determining anomalies and/or regularities in subject-specific highly-complex data sets are just a few examples of the captivating tasks AI is able to tackle.

The raison d'Être for ML or DL algorithms lies in the fact that at some point certain tasks become overly intricate for humans and conventional mathematical approaches lead to dissatisfactory results. In this scenario AI comes into action and tries to comprehend and subsequently unveil dependencies and patterns within the given assignment.

On account of the perpetual advancement of computational technology/hardware and the prosperous gathering of data through manifold branches, it is not coincidental that in this day and age the utilization of AI seems to rocket. It appears unexpected that the idea of the highly complex DL methods can be traced back to the 1980s [30], where CPUs and GPUs came not remotely close to the performance which we have today. Particularly, DL algorithms profit from these technological strides, due to their complex – and thus computationally expensive – structure.

With the rise of AI across various sectors of our modern society, it has also gained significant traction in science and industry. Numerous enterprises, universities and institutes have initiated projects to benchmark novel AI approaches against existing software solutions.

Computers with AI technology promising to boost the overall performance substantially [34], generative photo editing almost indistinguishable from a traditional photo shoot [64] or passing actual exams from law and business schools [41] are just few examples of the manifold tasks AI is able to handle proficiently.

The applications of AI span a vast array of fields, with the number of patents and research studies in this domain increasing at an unprecedented pace [48]. Among these initiatives is the i-RASE project, launched by the Technical University of Denmark (DTU). This project aims to develop a gamma radiation detector integrated with one or more artificial neural networks for real-time signal processing of incoming events. The envisioned applications of such a detector encompass medical imaging, industrial inspection, and space instrumentation [44].

One specific objective of this effort is to harness the capabilities of AI for the computation of certain features related to the incoming radiation. These features include the type of radiation and interaction, the energy of the charge carriers, and the three-dimensional impact position within the detector.

This endeavor naturally raises the fundamental question: *Can AI replace traditional methods in real-time detector data analysis?* Addressing this question is the central focus of this thesis.

For this purpose, data recorded by a CZT-based radiation detector were provided by one of the project partners. This data set was used to build and evaluate AI models that could eventually be integrated into such detectors for real-time operation. While the set was initially unlabeled, it still proved valuable for AI-driven data analysis. This thesis provides a detailed account of how the labeling problem was addressed, the AI models that were utilized, and the nature of the data involved.

The thesis begins by detailing the foundational principles of Cd(Zn)Te semiconductors as applied in radiation detectors, alongside an overview of the fundamental concepts of Artificial Intelligence. This introductory framework sets the stage for the experimental segment, which constitutes the core of the work. In this section, the radiation detector is characterized, the data set and its derived properties are analyzed, and the performance of the applied AI algorithms is systematically evaluated. The thesis concludes by assessing the feasibility of employing AI methods as alternatives to traditional analysis techniques and offering an outlook on potential future developments in the project.

2 Cd(Zn)Te detectors

Cadmium Zinc Telluride (Cd(Zn)Te) detectors, often referred to as CZT detectors, have proven to be invaluable tools in High Energy Astrophysics, enabling the detection of highly energetic photons originating from phenomena such as Black Hole accretion disks, compact mergers and Supernovae [4].

This thesis focuses on analyzing data recorded by a CZT detector. In the following, the fundamental principles of this type of detector are discussed, starting with the working principles of semiconductors. For additional details, readers may also consult [4].

The conductivity of a material is governed by its valence electrons. Elements naturally strive to achieve a fully occupied valence shell. Those with a significant deficiency of valence electrons in the outer shell tend to easily lose them, allowing for a high mobility of charge carriers, enabling electrical current, and are thus classified as conductors (e.g., metals). Conversely, elements with only a small deficit of valence electrons are more tightly bound to the atomic nucleus, making them poor conductors, or insulators (e.g., non-metallic elements such as sulfur). Semiconductors lie in-between, possessing a balanced ratio of free and bound electrons in their valence shell, enabling them to conduct electricity only under specific conditions.

Semiconductors come in the form of crystals and are typically characterized by two primary energy bands: the valence band, where electrons reside in their ground state and no conduction occurs, and the conduction band, where electrons move freely, allowing electrical conductivity. To transition from the valence band to the conduction band, electrons must acquire sufficient energy to overcome the energy gap between these bands, known as the band gap. This process is illustrated in Figure 1.

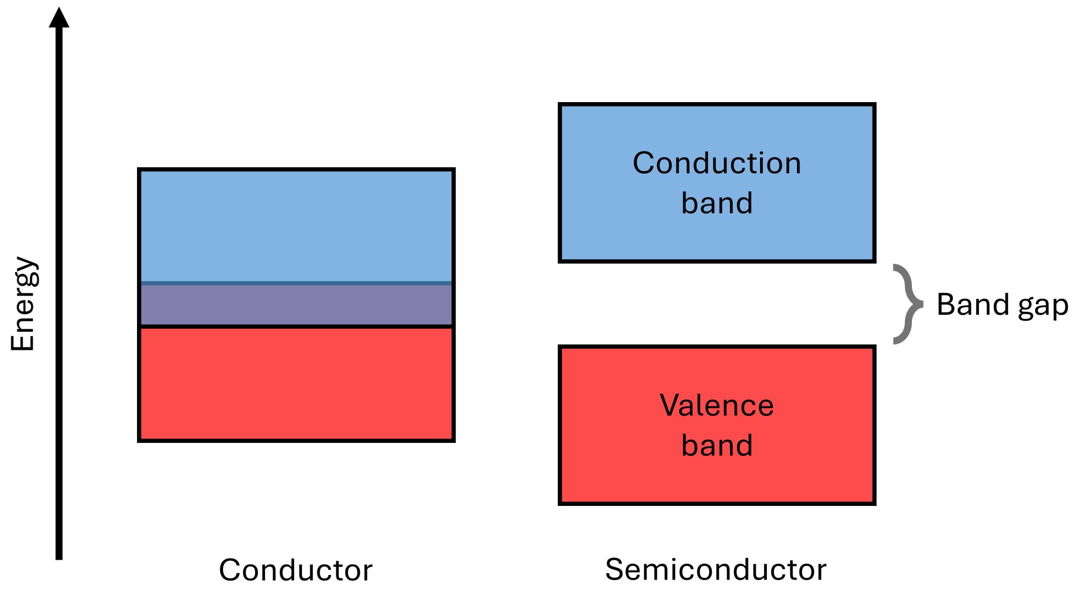


Figure 1: *In conductors, the valence and conduction bands overlap, allowing nearly all electrons to move freely and facilitate electrical current. In semiconductors, a band gap must be traversed for conduction to occur (adapted from [45]).*

Semiconductors are employed in detectors to capture high energy photonic radiation. When an incident photon interacts with the CZT crystal and its energy is sufficient to promote an electron from the valence band to the conduction band, electron-hole pairs are generated. The electron occupies now the conduction band, while the hole remains in the valence band. Subsequently, the hole manifests itself as an electron vacancy within the atomic lattice, effectively comprising a positive charge. The number of generated electron-hole pairs is proportional to the energy of the incident photon.

Under the influence of an applied electric field across the crystal, the negatively charged electrons migrate toward the anode, and the positively charged holes are drawn toward the cathode. The accumulation of these charges induces an electrical signal, which, as previously mentioned, is proportional to the energy of the incident photon. Using readout electronics, this charge signal is converted into a voltage pulse, which can be screened for human interpretation, enabling the determination of various properties of the incoming photon and, consequently, the astrophysical object emitting it [4][43]. This process is illustrated in Figure 2.

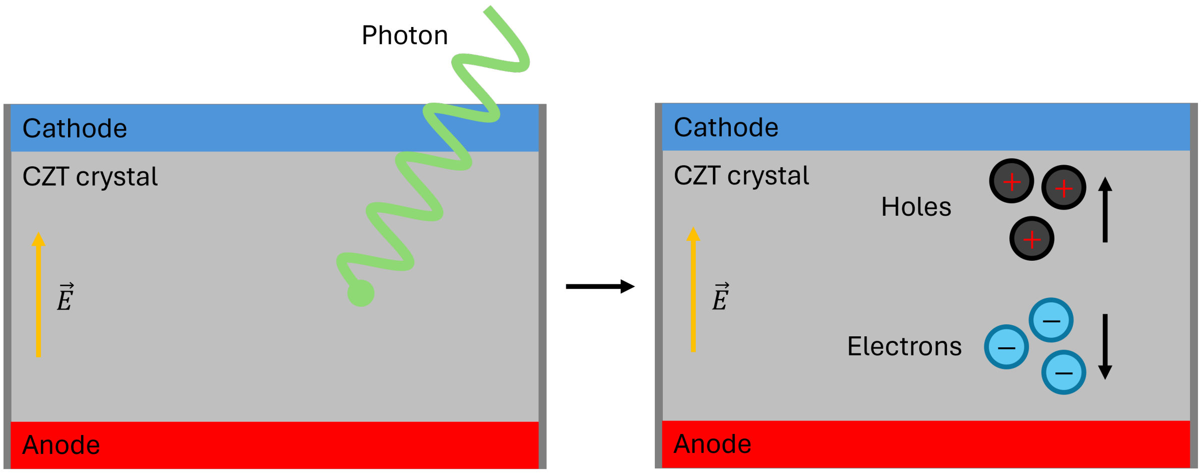


Figure 2: *Following the photon's interaction with the detector, electron-hole pairs are created and directed toward the anode and cathode, respectively (adapted from [4]).*

The detection mechanism outlined above can be summarized as follows: when a charge carrier deposits its energy within the semiconductor, it generates moving charges q (electron-hole pairs) within the semiconductor crystal. These moving charges induce a charge Q on the electrodes, which is then converted into an output signal by the detector's digital electronics. This process is governed by the fundamental equation:

$$Q = \oint_S \varepsilon \vec{E} \cdot d\vec{S} \tag{1}$$

where the induced charge Q is calculated over the surface $\vec{S}$ enclosing the electrode. $\vec{E}$ represents the electric field, oriented as normal vectors to the surface, while ε is the dielectric constant of the medium.

However, directly applying Equation (1) to calculate Q is computationally challenging because the electric field $\vec{E}$ varies with the position of q, necessitating its calculation for every event before using Equation (1) [25].

The Shockley-Ramo theorem – presented in Equation (2) – offers a significant simplification:

$$Q = -q\phi_0(\vec{x}) \tag{2}$$

This theorem states that the charge Q induced on an electrode can be calculated using q (explicitly considered as point charge) and the weighting potential $\phi_0(\vec{x})$, which is present at the charge's position $\vec{x}$. The weighting potential ϕ_0 is defined under specific conditions: the electrode of interest is held at a unit potential (1 Volt), all other electrodes are at ground potential (0 Volts), and any additional charges are absent. By reducing the problem to the calculation of ϕ_0, which depends only on the geometry of the electrodes, the otherwise complex computations described in Equation (1) are transformed into an electrostatic problem for each movement of the charge, making it significantly easier to solve [25][33][59]. A formal proof of the Shockley-Ramo theorem is available in [25].

The induced charges, computed using the Shockley-Ramo theorem, are directly proportional to the energy deposited in the recorded event. The energy can be expressed in terms of least significant bits (LSB) via the detector's digital electronics, as will be discussed in greater detail in Chapter 4.1.

When calculating the energy of an event, it is crucial to account for the type of radiation and interaction that occurred. In some cases, an electron may scatter and escape the detector, carrying away a portion of the energy. This effect must be carefully considered by analyzing the induced pixel pattern, which will be reviewed in more detail in Chapter 4.6.

Conversely, the measured energy can also provide valuable insights into the type of interaction that likely occurred within the detector. The primary objective of this thesis, viewed from an astrophysical perspective, is to detect high-energy photons originating from astrophysical sources and objects. Consequently, the primary focus of this chapter centers on photonic interaction, which are explained in detail in the following paragraphs.

Figure 3 shows the most probable photon interaction processes based on the energy and the atomic number Z (number of protons) of the detector material. CZT crystals, with an approximate atomic number of 50 [39], experience that photons with energies below 0.4 MeV predominantly undergo the photoelectric effect, while those in the range of 0.4 MeV to 5 MeV are primarily subject to Compton scattering. At energies above 5 MeV, pair production becomes the most probable interaction mechanism.

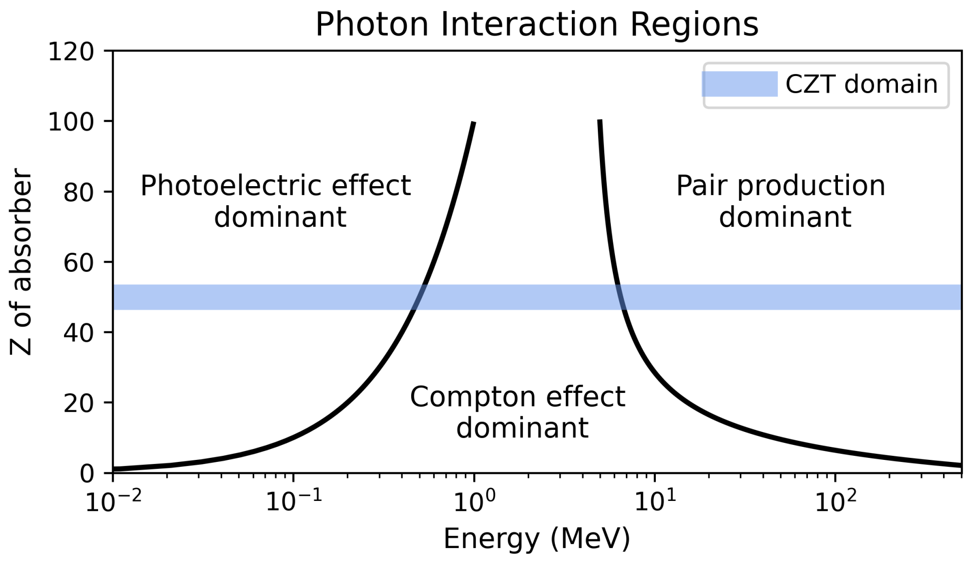


Figure 3: *The probability of photonic interactions depending on the energy and the atomic number (Z) of the absorber material. The plotted lines subdivide the predominant regions of each interaction by the energy content of the interaction and the Z of the material in which this interaction happened. The specific CZT domain is highlighted at* $Z = 50$ *(adapted from [80]).*

Photoelectric effect

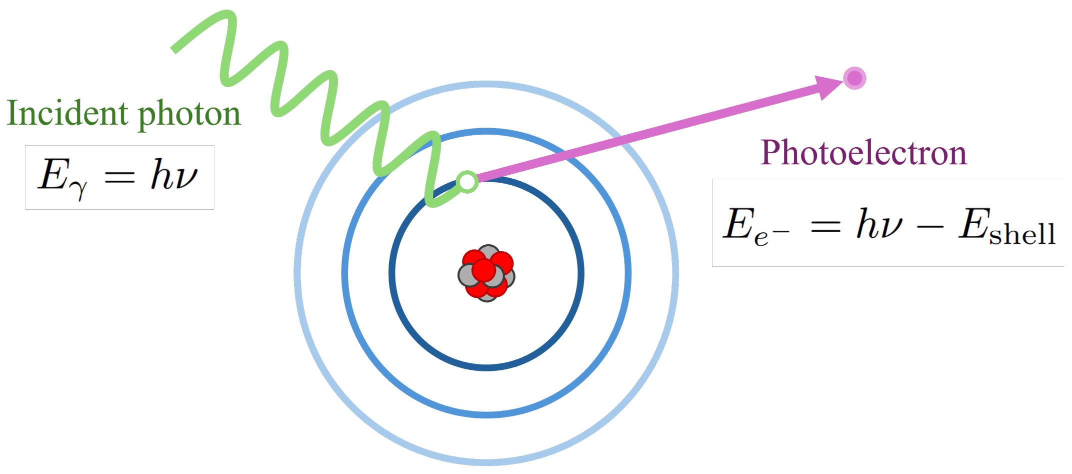


Figure 4: *Principle of the photoelectric effect: An incident photon interacts with the inner layers of an atom in the absorber, ejecting an electron, the so-called photoelectron. The energy of the ejected photoelectron is equal to the energy of the incident photon minus the binding energy of the electron's atom shell (adapted from [46]).*

The photoelectric effect (Figure 4) describes a process where an incident photon interacts with an atom of the absorber material, striking an electron in one of its inner shells. Notably, the full energy of the photon is passed on to this electron, which then becomes a photoelectron, ejected from the atom, with an energy equal to the photon's initial energy minus the electron's binding energy [76]. The ejected photoelectron can further interact with nearby atoms, potentially imparting enough energy for lifting their electrons into the conduction band, generating electron-hole pairs.

Compton scattering

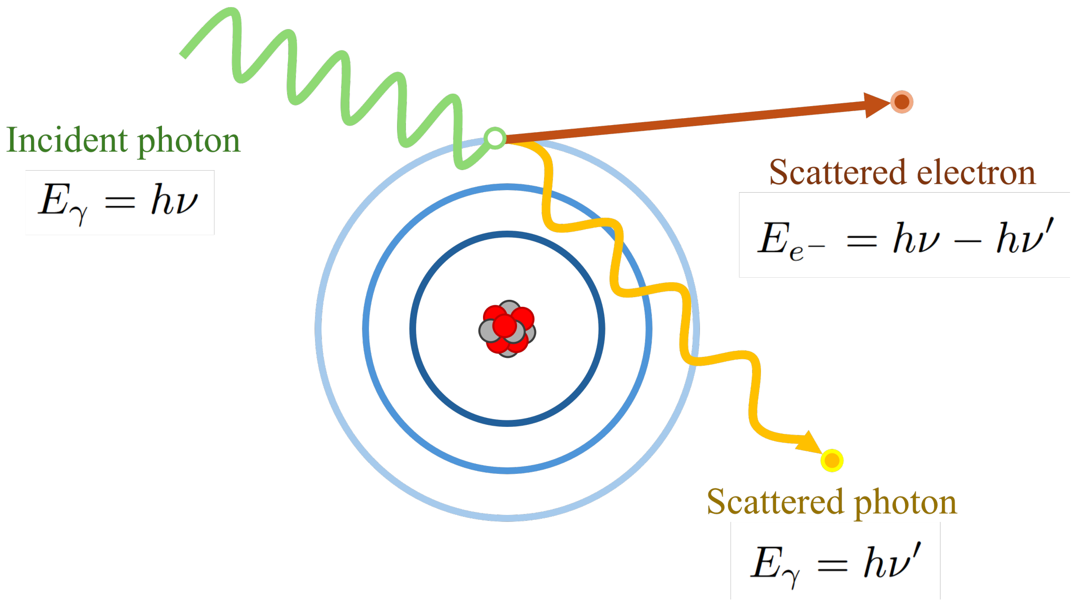


Figure 5: *Principle of Compton scattering. A photon interacts with an outer-shell electron, resulting in scattering. The electron is ejected with an energy equal to the difference between the incident photon energy and the scattered photon energy (adapted from [46]).*

Compton scattering is an inelastic scattering process in which an incident photon interacts with an electron, typically one in an outer shell, ejecting it from the atom. During this interaction, the photon changes direction at an angle φ and transfers a portion of its energy upon impact with the electron, which is then scattered at an angle ϕ. The electron gains kinetic energy equal to the difference between the photon's initial energy and its energy after scattering, potentially generating electron-hole pairs during its motion in the semiconductor (explanation supported by Figure 5) [8].

Pair production

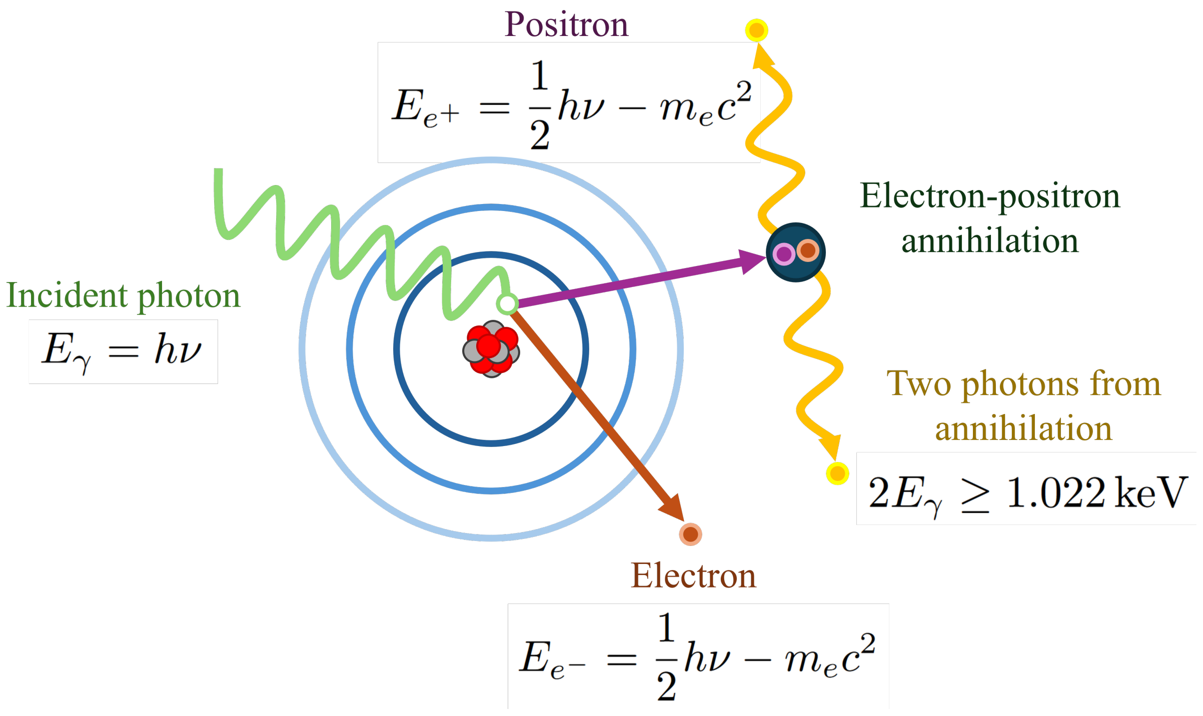

Positron
$E_{e+} = \frac{1}{2}h\nu - m_e c^2$
Electron-positron annihilation
Incident photon
$E_\gamma = h\nu$
Two photons from annihilation
$2E_\gamma \geq 1.022\,\text{keV}$
Electron
$E_{e-} = \frac{1}{2}h\nu - m_e c^2$

Figure 6: *Principle of pair production and the subsequent annihilation. The photon can undergo a spontaneous transformation into an electron and positron, carrying half of the energy of the incident photon reduced by the rest mass energy of the resulting electron (or positron). Eventually the positron annihilates with an electron of the absorber culminating in the creation of two photons (adapted from [46]).*

As it propagates through the crystal, a high-energy photon can spontaneously transform into an electron and a positron, if its energy exceeds the threshold defined by the combined rest mass energy of these two particles: $2m_e c^2$, corresponding to an energy amounting at least 1.022 MeV. The positron, as the electron's antiparticle, shares its properties – including mass – but carries an opposite charge [76]. Both particles travel through the semiconductor, generating electron-hole pairs along their paths. While the electron creates these pairs through repulsive Coulomb interactions, the positron engages in similar interactions but with attractive forces [59].

Once the positron has sufficiently slowed down (since lower energy increases the likelihood of the following process), it may encounter an electron, initiating an annihilation process. In this process, the electron and positron are converted into two photons. As illustrated in Figure 6, these photons are emitted in precisely opposite directions if the annihilation occurs at rest. Each photon therefore possesses the positron's rest mass energy of 511 keV and can subsequently interact with the detector, either through Compton scattering or the photoelectric effect.

Additionally, there is Rayleigh scattering, an interaction in which photons are elastically scattered, sustaining their initial energy. This type of interaction is less significant for gamma-ray spectroscopy [46].

Earlier in this chapter, it was highlighted that the primary focus of these studies lies in photons and their interactions. However, other types of radiation may originate from natural decay processes in the experiment's environment or – more relevant for astrophysics – from cosmic rays.

Cosmic rays consist of high-energy particles originating from various galactic and extragalactic sources across the universe. The majority of cosmic rays are protons, followed by helium nuclei, and also a small fraction of heavier nuclei. In gamma-ray spectroscopy, cosmic rays are regarded as background noise. Their secondary particles – generated through interactions with the earth's atmospheric particles, forming extensive particle showers – can obscure the actual target signal either by producing also gamma rays or charged particles that interact with the detector. For instance, cosmic-ray-induced negatively charged muons can ionize atoms in the detector, creating electron-hole pairs. Muons leave a distinctively long track pattern within the detector, due to its significant mass relative to the electron the deceleration (gradual energy loss) is elongated [9][46][75].

The pixelated surface of the detector is heavily influenced by the type of interaction and radiation associated with the incident event (cf. Chapter 4.6). The pixelated image can be further refined to enhance event classification and improve event imaging quality: A thorough analysis of the detector's response enables event classification with a finer subpixel precision surpassing the limitation of the resolution imposed by the manufacturer's pixel pitch. This can be accomplished for instance by examining the transient signals, particularly the amplitudes of neighboring pixels relative to the pixel with the highest energy content. If a neighboring pixel on one side exhibits a higher amplitude, it suggests that the resulting charge cloud is closer to the edge of that adjacent pixel [59]. The specific method applied in this work is elaborated in Chapter 4.7.

Beyond refining the spatial resolution in the x- and y-directions, the depth of interaction (the z-component) can also be incorporated into the analysis. This enables three-dimensional event imaging, facilitating a more detailed classification of events, including a profound investigation of the determination of the source location. A commonly used technique for this purpose is cathode depth sensing, which relies on the ratio of the anode and cathode signals. The anode signal is proportional to the induced charge, while the cathode signal is proportional to the induced charge multiplied by the depth of interaction. A comprehensive explanation of this method can be found in [29]. However, in this work, the provided detector (cf. Chapter 4.1) employs a distinct approach to determine the depth of interaction, which will be thoroughly described in Chapter 4.5.

The operational principles of a CZT detector, enabled by the unique composition of its crystal, are effectively utilized in astrophysical gamma-ray observations as well as in medical imaging contexts. This crystal, a compound semiconductor, is composed of multiple elemental constituents: cadmium (Cd), zinc (Zn), and tellurium (Te). The material composition is typically represented as $Cd_{1-x}Zn_xTe$, with $x = 0.1$ being the most common material cut [4].

The use of CZT crystals in detectors offers significant advantages for X-ray and γ-ray detection compared to older technologies, such as those employing cadmium telluride (CdTe) crystals. One of the major benefits is that CZT detectors can operate at room temperature due to their relatively wide band gap of approximately 1.57 eV (for $Cd_{0.9}Zn_{0.1}Te$). This eliminates the need for additional cooling devices, which would otherwise be required to reduce thermal noise [4][58]. The wide band gap ensures that, even at higher temperatures, thermal excitation of electrons across the gap is unlikely, thereby minimizing unwanted noise in the detection process.

Furthermore, the high atomic number Z of the material ($Z \approx 50$) enhances detection efficiency in the high energy regime [47]. This efficiency stems from the interactions of photons with the material, which include the photoelectric effect, Compton scattering, and pair production. Among these, the photoelectric effect is particularly favorable as it typically results in the complete deposition of the photon's energy within the detector material. The probability of detecting the photoelectric effect is proportional to Z^4, meaning materials with a higher atomic number are more likely to record this interaction.

In this section, the fundamental principles of semiconductors and the advantages of employing CZT detectors have been presented. The primary goal of this work is to evaluate the data recorded by these detectors using Artificial Intelligence. The theoretical foundations of these methods will be explored in detail in the subsequent chapter.

3 Artificial Intelligence

Artificial Intelligence aims to emulate intelligent behavior analogous to the cognitive functions a human would comprise. A diverse array of approaches has been developed to attain this intelligence, some of which are described below.

The terms Artificial Intelligence (AI), Machine Learning (ML) and Deep Learning (DL) are often used interchangeably, though their real relationships can be better understood with the aid of Figure 7.

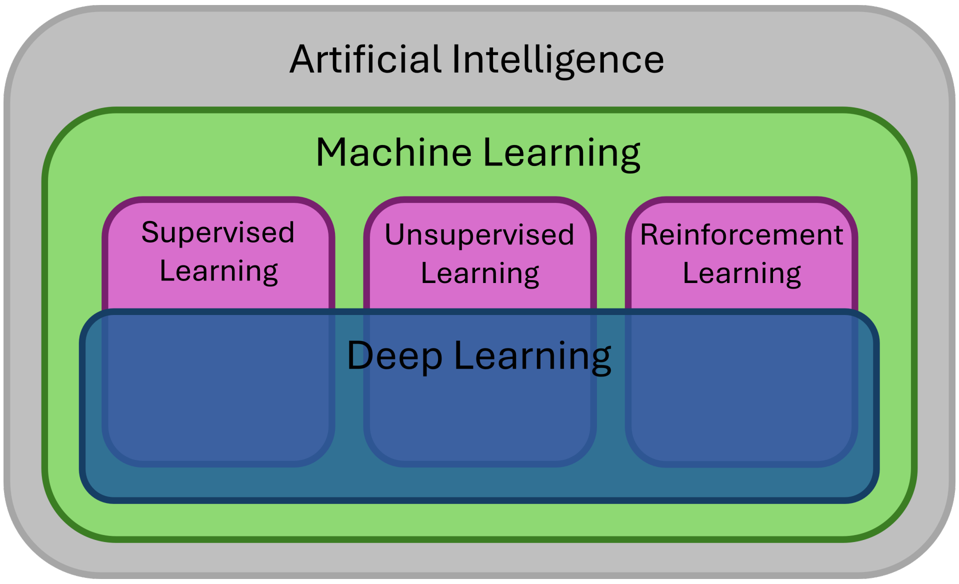


Figure 7: *The hierarchical relationship between Artificial Intelligence, Machine Learning, and Deep Learning, as well as their associated learning paradigms: Supervised Learning, Unsupervised Learning, and Reinforcement Learning. Machine Learning serves as a subset of AI, while Deep Learning, a further specialized field within Machine Learning, also utilizes these learning strategies to achieve its objectives (adapted from [60]).*

Machine Learning algorithms serve as universal function approximators, adjusting their parameters to align with the patterns and relationships in the input data – a process commonly referred to as training.

Moreover, the term Deep Neural Networks refers to algorithms designed to emulate the functional principles of the human brain. When these algorithms are now applied and trained to solve a specific task, they are encompassed within Deep Learning, a subset of Machine Learning [60].

Certain fields, such as robotics – the conceptualization and operation of machines resembling, inter alia, human motor abilities – fall under AI but not ML [24]. Similarly, some algorithms within ML do not belong to DL, examples of which are discussed in Chapter 3.2 of this thesis.

The fundamental question arises: How can algorithms learn autonomously? This is addressed through three general paradigms: Supervised Learning, Unsupervised Learning, and Reinforcement Learning.

Supervised Learning is best described as learning with solutions provided. In this approach, the training data is accompanied by corresponding labels. It is among the most widely used learning paradigms due to its versatile applications, including the work discussed in this thesis. The main challenge lies in obtaining labeled data, which is often time-intensive and reliant on expert input. In such cases, labels can alternatively be derived from deterministic simulations, traditional models and functions, or specially designed experiments where outcomes are known in advance.

In addition to the training data, a separate testing data set is reserved and excluded from the training process. Once the model is fully trained, its performance is evaluated on this unseen test data, serving as a measure of its ability to generalize – effectively "understanding" the task's objective and successfully applying learned knowledge to previously unencountered scenarios. Furthermore, a validation data set can be employed during the training process to assess the model on unseen data at various stages of the training, which helps to monitor the learning progress more proficiently.

Unsupervised Learning, by contrast, involves learning without predefined solutions. This learning type is typically employed to uncover underlying patterns, condense data into a simplified representation, or understand probability distributions to generate realistic synthetic examples [60]. A prime example of the latter is the Generative Adversarial Network (GAN), which can produce synthetic images as illustrated in Figure 8.

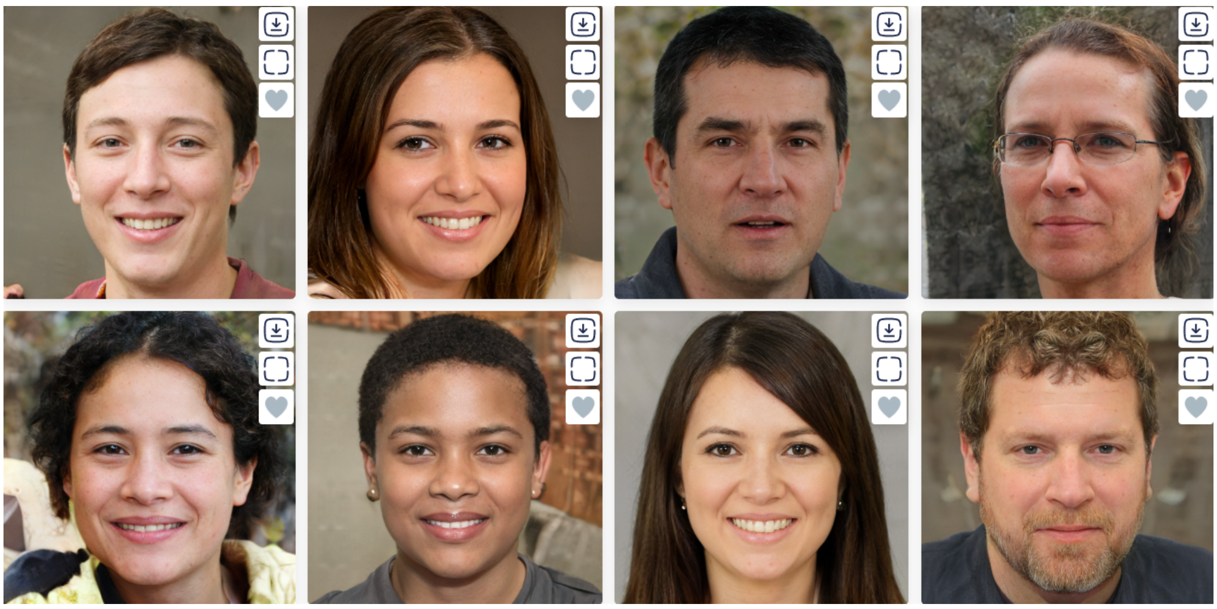


Figure 8: *Images generated by a GAN. None of these faces exist in reality, showcasing the capability of generating lifelike examples [74].*

GANs are a class of Deep Learning models consisting of two primary components: a generator and a discriminator network. Both networks learn the underlying features of the training data, such as the structural attributes of a face (e.g., the arrangement of eyes, nose, and mouth). The generator synthesizes modified versions of the training data and presents them to the discriminator, which evaluates whether the samples are real (from the training set) or generated. The GAN is considered fully trained when the discriminator can no longer reliably distinguish between the real and generated data [60].

Reinforcement Learning (RL) involves training an AI agent to interact with an environment through a sequence of actions. Each action changes the state of the environment and is associated with a reward. The AI's goal is to maximize cumulative rewards over time. A unique challenge in RL arises when the largest rewards are delayed, requiring the agent to recognize that short-term sacrifices can lead to long-term gains. During training, the AI must balance exploration – trying new actions – with exploitation – relying on known strategies that yield satisfactory results [60]. A chessboard serves as an ideal RL environment: the agent learns to evaluate the board's state and determine the optimal moves leading to victory. The delicate and desired exploration and exploitation in a RL training can be exemplified in the famous chess match between Stefan Levitsky and Frank Marshall, where the bold move "black queen to G3", initially appeared to be a huge sacrifice, yet, ultimately secured the game (Figure 9).

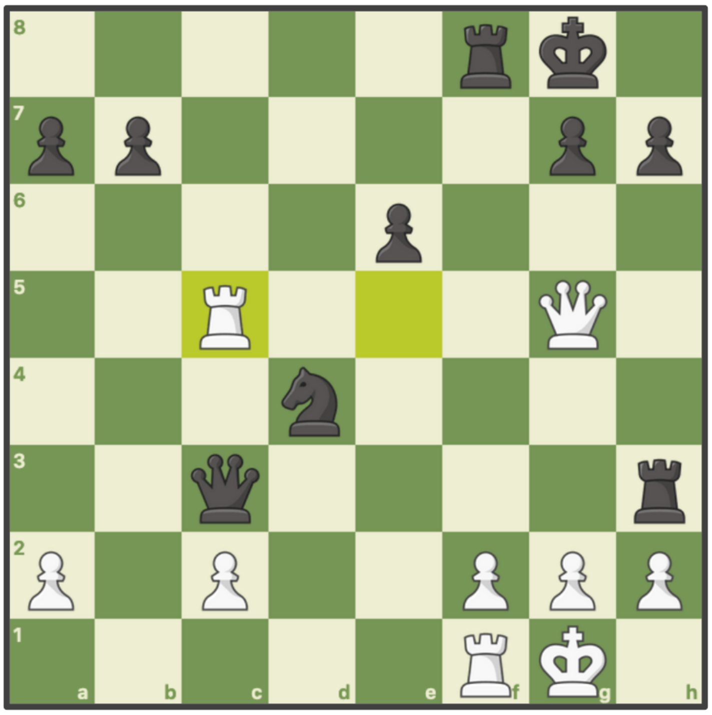


Figure 9: *Chess as a use case for RL highlights the importance of learning that seemingly unfavorable moves can ultimately lead to significant victories. This concept, which is intuitive for humans, is demonstrated by the match of Stefan Levitsky vs. Frank Marshall: The outstanding move ("black queen to G3") lead to checkmate [12].*

In this thesis, all models are developed within the framework of Supervised Learning. Consequently, this paradigm is the primary focus, and it is assumed throughout that labeled data is consistently available, unlike in Unsupervised Learning.

3.1 Classification and Regression

Classification refers to the process of assigning or predicting a label to a given sample. For example, consider a fruit basket containing a red, 10 g, spherical fruit with a radius of 10 mm labeled as "cherry". Here, the descriptive attributes – red, 10 g, spherical, and 10 mm – are referred to as a sample of the tabular data X, while the label "cherry" represents the target y.

Now, suppose the basket also contains an apple with characteristics such as red, 100 g, spherical, and 10 cm. A classification algorithm could be employed to automatically and correctly classify all the fruits in the basket. However, designing a classifier that solely relies on features like color or shape would fail to distinguish between the cherry and the apple, as both examples share similar attributes. In contrast, selecting the feature of weight would enable a clear distinction between them. Yet, if a banana weighing 100 g were introduced, weight alone would no longer suffice, necessitating the inclusion of additional features or a combination of features. This process of selecting and optimizing features is essentially the training phase of the model, where the algorithm identifies on how to solve the task with minimal error.

Different classifiers adopt various approaches to achieve this. For instance, one type of classifier may attempt to fit a decision boundary, such as a line or hyperplane, that separates different classes in the feature space, which can extend into high-dimensional spaces for more complex data. Another classifier might determine the class label based on the nearest neighbors in the feature space to the new sample, where the number and proximity of the neighbors influence the decision. Tree-based models, on the other hand, rely on a sequence of binary decisions structured as a decision tree to assign labels. A more detailed discussion on tree-based models is provided in Chapter 3.2.1.

An example is shown in Figure 10 of a fruit basket classifier as a linear decision boundary for distinguishing cherries from strawberries. The same problem arises as discussed before: A mere classification based on the color would fail, thus taking the combination of diameter and weight holds the desired classification.

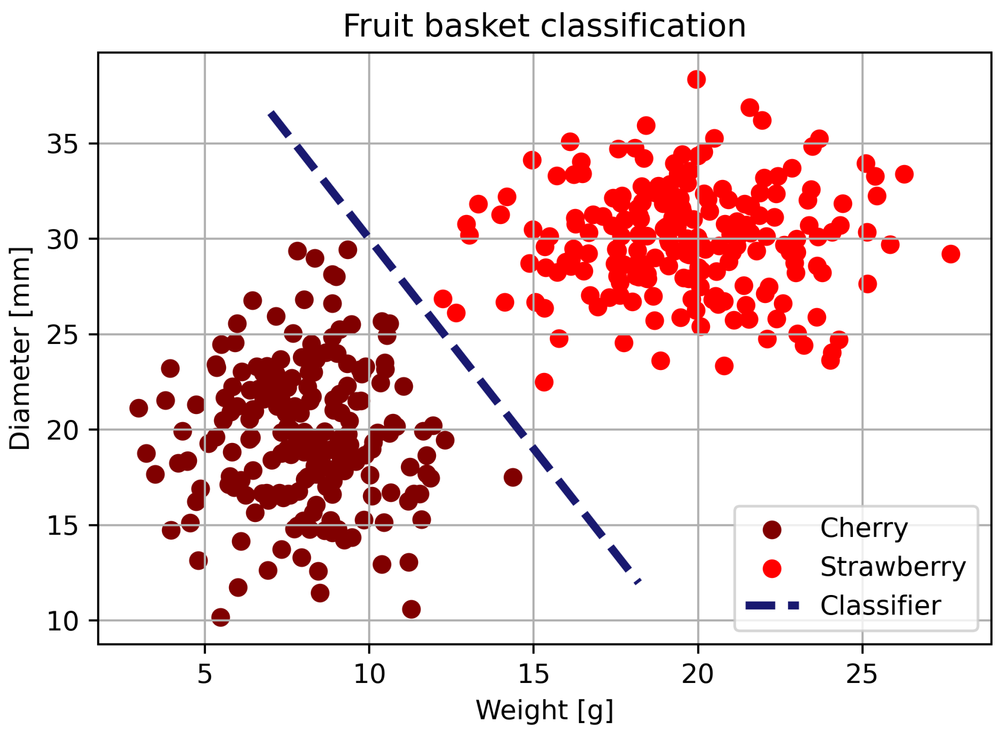


Figure 10: *An example illustrating how a classifier might operate, using features such as weight and diameter for linear separation. Color alone would not suffice for classification. Identifying the most distinguishing features is a task for machine learning algorithms. Here specifically weight and diameter for cherry-strawberry classification.*

In classification tasks, the target variable y is discrete. Conversely, in regression tasks, y is continuous. Both tasks involve finding a mapping from observations X to output y, and the better the algorithm performs, the more generalizable and reliable its predictions. The algorithms mentioned earlier, including tree-based models, also have counterparts optimized for regression tasks, which are discussed further in Chapter 3.2.1.

An example of a regression task is shown in Figure 11, which depicts the relationship between the age of chickens and their body weight.

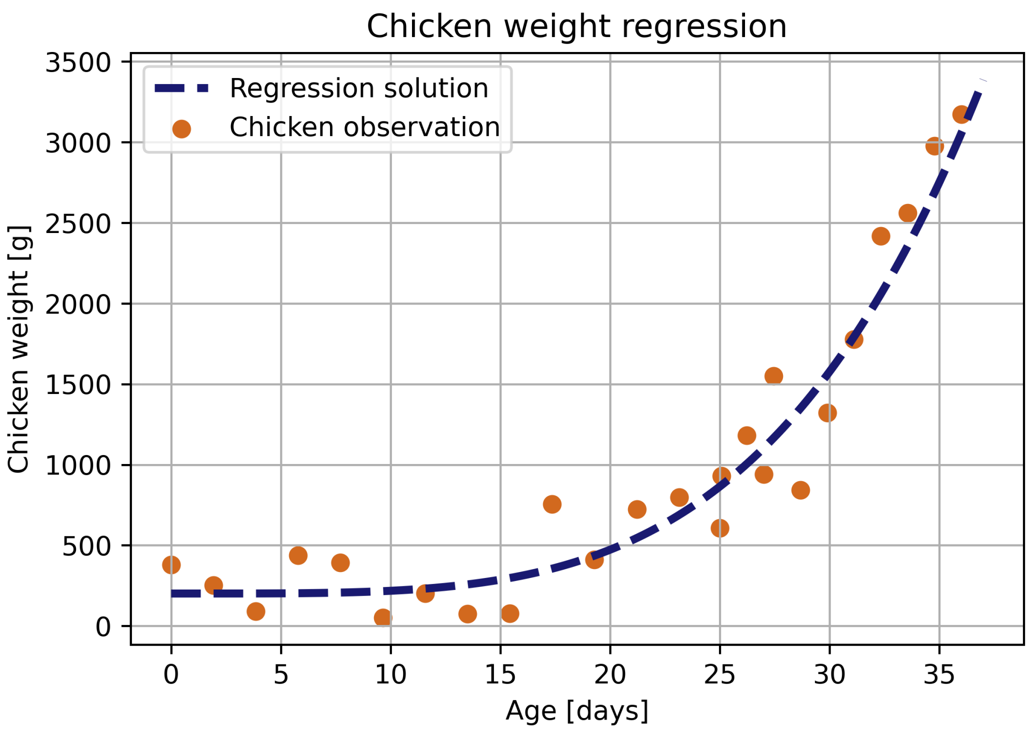
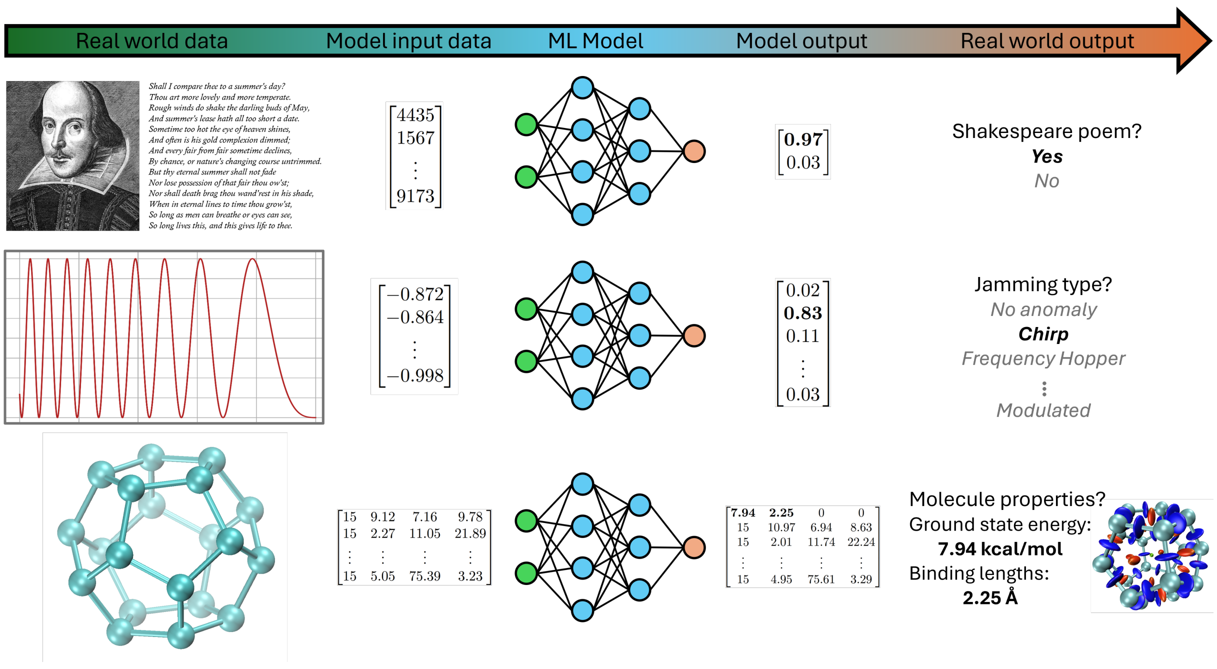


Figure 11: *An example of regression using chicken weights as a function of their age. Regression tasks aim to identify a functional relationship between observations and targets, enabling interpolation for unknown data points (adapted from [65]).*

In Figure 11, each dot represents a data point corresponding to an observed chicken. A well-trained regression model will generate a mapping similar to the blue curve, which interpolates the weight y for a given age X.

While the examples above involve relatively straightforward tasks, Machine Learning (ML), and particularly Deep Learning (DL), have demonstrated remarkable adaptability and success in solving highly intricate problems. Some advanced use cases are depicted in Figure 12.



Shall I compare thee to a summer's day?
Thou art more lovely and more temperate.
Rough winds do shake the darling buds of May,
And summer's lease hath all too short a date.
Sometime too hot the eye of heaven shines,
And often is his gold complexion dimmed;
And every fair from fair sometime declines,
By chance, or nature's changing course untrimmed.
But thy eternal summer shall not fade
Nor lose possession of that fair thou ow'st;
Nor shall death brag thou wand'rest in his shade,
When in eternal lines to time thou grow'st,
So long as men can breathe or eyes can see,
So long lives this, and this gives life to thee.

Figure 12: *Three possible applications well-suited for the use of DL, starting from the real world observations to the mapping for the model and based on that the model produces an output, which can be transformed to a real world explanation eventually. First row: Poet classification by learning the linguistic characteristics of the author to determine if randomly chosen poems can be considered as his work [84]. Second row: The GNSS or GPS signal that is received by positioning devices can be altered or omitted by jamming devices. The signal's waveform is fed to the network and the corresponding label would comprise the jamming type [19]. Third row: An AI can learn the properties of molecular structures. The model input is the approximate shape of the target molecule and the results include, inter alia, ground state energy, binding lengths and the optimized structure with the resulting Van der Waals forces [79].*

Figure 12 highlights three possible intricate AI-tasks. Training a neural network to understand the semantics of human language and even distinguish an author's characteristic writing style represents a highly complex task. Remarkably, this has already been achieved by advanced Large Language Models (LLMs), which can provide meaningful and contextually aware responses [62]. For instance, one practical application of such models lies in author identification, particularly in cases of cybercrime. Perpetrators of hate speech, fake news, cyberbullying, or online threats often conceal their identities. By training an AI on linguistic features, such as syntax, vocabulary, and writing style, the identity of an author can be inferred, allowing investigators to trace malicious content back to its source. As a demonstration, consider Shakespearean literature: its distinct characteristics, including verbal nuances and metrical patterns, make it well-suited for such classification tasks. The input data typically consists of tokenized text, which the AI processes to compute linguistic relations, with the output being the predicted author. This task falls under the category of binary classification whether the poem is written in a Shakespearean style or not [84].

Another challenging application of Deep Learning is the identification of jamming signals within GNSS systems, which encompass GPS and other global navigation technologies. In military contexts, jamming devices are employed to disrupt drone and radar systems. These devices generate a variety of interference patterns, such as chirps, frequency shifts, etc., each requiring a specific mitigation strategy. AI models can be trained to detect these jamming patterns and the jammer position by analyzing how the signal changes in response to the interference. The input to the model might be the waveform of the received signal, with the output being the classification of the interference type. This task exemplifies multi-class classification, where the AI learns to recognize distinct jamming types based on labeled training data [19].

An even more intricate example lies in predicting properties of molecular and crystal structures, a task at the frontier of computational chemistry and material science. Complex systems, such as proteins or molecular assemblies, require expensive computational methods, like Density Functional Theory (DFT), to approximate their properties, including ground-state energy, atomic positions, and binding lengths. These calculations often span weeks and the result is known to be a approximation to the true observed structures (depending on the theory level of the used DFT functions, it can be gauged on how good the approximation is in the end), making AI-based approaches a valuable alternative. By training models on approximate molecular structures and assigning the corresponding observed real-world properties stemming from data of well-studied molecules, AI can learn mappings that enable rapid and accurate predictions. This approach not only accelerates research but also enhances accuracy.

A striking achievement in this domain is AlphaFold, which earned the 2024 Nobel Prize in Chemistry for revolutionizing protein structure prediction [38]. The input to such AI models might include atomic arrangements, while the output could involve multivariate regression predicting multiple properties, such as Van der Waals forces, optimized positions, and energetic states [78][79].

These examples were shown to illustrate the manifold possibilities and the adaptability of ML and DL techniques in solving diverse and complex tasks.

In this thesis, the focus is on learning the relationship between the detector's response and the resulting event properties, including energy, radiation type, and interaction type through the application of AI. Despite the inherent complexity of the task, the very essentials of the DL and ML framework distill down to the simple functional form [23]:

$$f(X) = y \quad \Leftrightarrow \quad \mathrm{AI}(X) = y \tag{3}$$

where f represents the AI algorithm, X denotes the input data (which in principle can have arbitrary dimensions) and y is the desired output. This notation will be consistently used throughout this thesis.

The fundamental relation in Equation (3) underscores the central concept, though the specific methods for achieving the mapping vary significantly. The initial approach explored in this thesis involves tree-based algorithms, which fall under the umbrella of traditional Machine Learning methods.

3.2 Machine Learning

Machine Learning (ML) aims to fit a mapping function $f : X \to y$. The more diverse and extensive the training data, the more accurate and reliable the predictions of y are. For instance, in a binary classification task illustrated in Figure 10, the inclusion of a wide range of fruit sizes enables a more precise boundary for separating the classes. This, in turn, enhances the accuracy of future predictions. While this example demonstrates a spatially-driven binary classifier, tree-based algorithms employ a fundamentally different approach, as outlined below. The following chapters draw upon the foundational theories presented in [63], which delves into Decision Trees, [5], elaborating on the principles of Random Forests, and [20], which investigates Gradient Boosting Trees.

3.2.1 Decision Tree

Instead of utilizing spatial features for classification or regression tasks, Decision Trees operate through a sequence of binary questions (e.g., yes/no, greater/smaller), each observing a single property at a time [31][54]. While there exist variations of decision trees that account for multiple properties simultaneously [6], for simplicity, we will focus on Decision Trees observing only one feature per question.

A Decision Tree consists of a root node, internal nodes, and leaf nodes. Initially, all samples reside in the root node, where the first question is posed to split the data. If the data reaching a node becomes "pure" (i.e., contains only one class), the node is designated as a leaf node and is labeled with that class. Conversely, if two or more classes remain in a node, it becomes an internal node, requiring further splits until purity is achieved [31].

The selection of the splitting questions is critical. The optimal question is determined based on the information gained, i.e., highest possible number of samples have reached a leaf node ("highest possible" since in more intricate tasks one split criterion can not separate the entirety of the data set to each class). To measure the disorder in a node, entropy H is often employed and will be defined in the following [54].

Consider D_m as the total number of samples m reaching an arbitrary node (with $m = 1, 2, ..., M$), with $D_{m,j}$ representing the samples directed to outcome j describing the outcome (with $j = 1, 2, ..., J$) after a split, with f as the criterion for splitting the classes (it follows then always of course that $D_{m,j} \leq D_m$). The index k (with $k = 1, 2, ..., K$) indicates the class of a sample, so $D^k_{m,j}$ denotes the number of samples of class k in outcome j. The relative proportion of class k compared to all samples in outcome j is defined as $p^k_{m,j} = \frac{D^k_{m,j}}{D_{m,j}}$ [22].

The entropy, the measure of disorder, is defined via:

$$H_{m,j}(f) = -\sum_{k=1}^{K} p^k_{m,j} \log_2(p^k_{m,j}) \tag{4}$$

Another widely used disorder metric in Decision trees is the Gini index [22] (which is also used in this work), defined as:

$$G_{m,j}(f) = 1 - \sum_{k=1}^{K} \left(p^k_{m,j}\right)^2 \tag{5}$$

And to calculate the disorder in each tree level for all the inner nodes:

$$G_m(f) = -\sum_{j=1}^{J} \frac{D_{m,j}}{D_m} \, G_{m,j}(f) \tag{6}$$

The aim is to minimize the disorder at each internal node, while leaf nodes inherently have a disorder value of $G_m(f) = 0$ since they contain only one class.

The decision function f (split criterion) is evaluated for each feature, and the split yielding the lowest $G_m(f)$ is selected to optimize classification at that node. However, the decision tree algorithm follows a greedy strategy, selecting the optimal split at the current step without considering the global tree structure. This can result in suboptimal trees, where an alternative initial split might have produced a shallower tree and thus a faster classification. To illustrate this process, consider the fruit basket example:

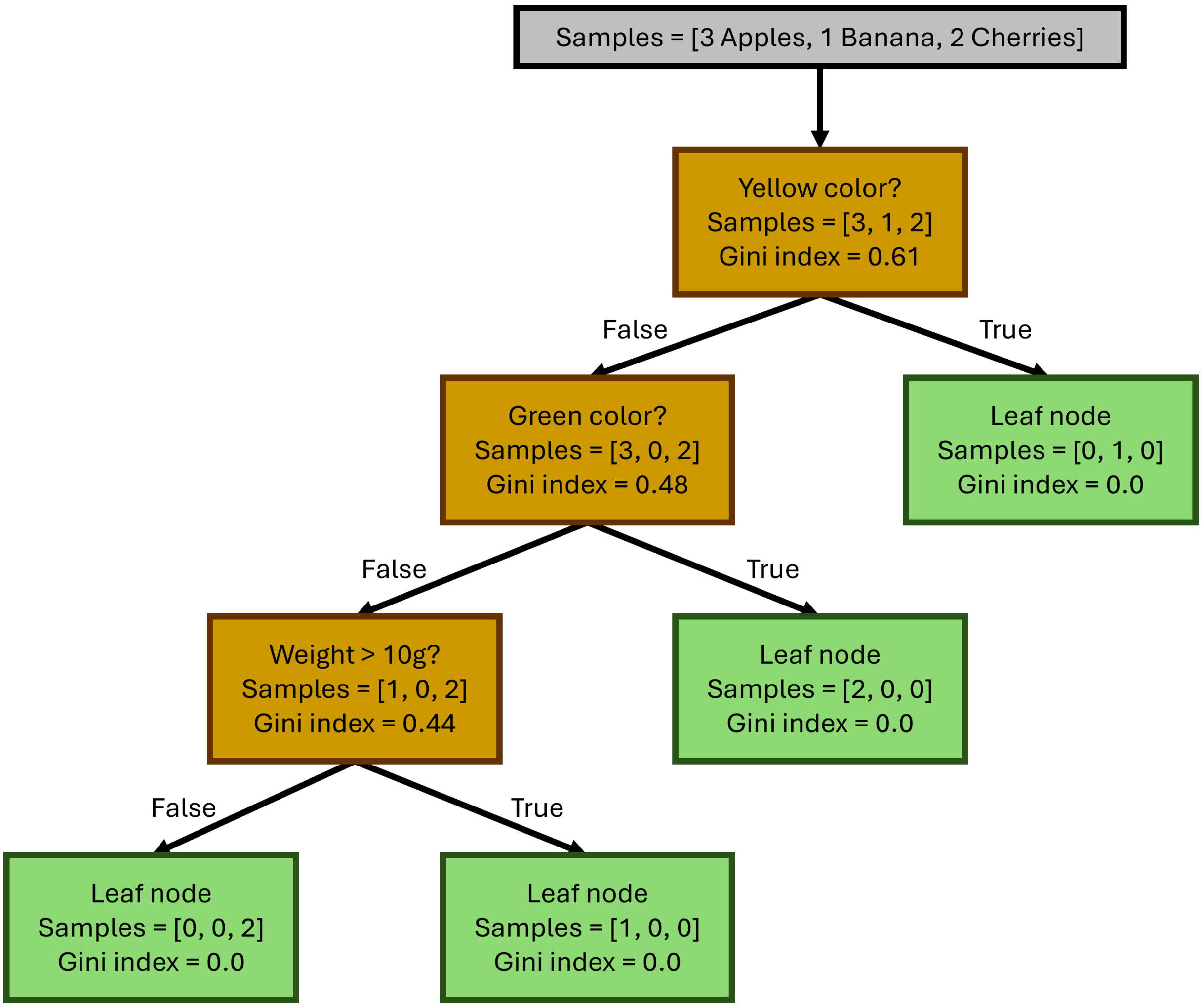


Figure 13: *Decision Tree training. The sequence of questions and corresponding Gini indices, calculated via Equation (6), are shown at each node. By the end of training, all fruits are classified into leaf nodes, enabling the tree to classify unknown fruit baskets – fully trained classifier.*

In this example, the fruit basket contains 3 apples, 1 banana, and 2 cherries. Initially, all samples are in the root node. The first split distinguishes the banana based on color (yellow), assigning it to a leaf node and significantly reducing the Gini index (Equation (6)). The remaining apples and cherries proceed to an internal node. A second question, whether the fruit is green, separates two green apples into a leaf node, leaving one red apple and the cherries. Since color is no longer a distinguishing feature, the next split uses weight to classify the remaining samples. Albeit this fruit basket being a simplified example – typically data sets contain more samples –, the same principles would also apply to more intricate tasks.

For regression tasks, Decision Tree regressors handle continuous features by evaluating potential splits across at all indices in the feature's range. In this case, a disorder metric would not be an expressive measure for a possible successful split. The classification metrics are substituted by e.g., the Mean Squared Error (MSE), a regression metric:

$$\mathrm{MSE} = \frac{1}{n} \sum_{i=1}^{n} \left(y_{\mathrm{pred},i} - y_{\mathrm{true},i} \right)^2 \tag{7}$$

The MSE emphasizes larger error values due to the square in the equation and can be used in the training as criterion. Whereas in the evaluation of training (and also testing) the Root Mean Squared Error (RMSE), simply the square root of MSE, is preferred for interpretability. Alternatively, the Mean Absolute Error (MAE) is suitable when all prediction errors, regardless of magnitude, are treated equally:

$$\mathrm{MAE} = \frac{1}{n} \sum_{i=1}^{n} \left| y_{\mathrm{pred},i} - y_{\mathrm{true},i} \right| \tag{8}$$

An adequate visualization of the Decision Tree regressor is shown in [3].

One drawback of decision trees, as previously mentioned, is their reliance on a greedy approach, which might result in a suboptimal tree structure. This shortcoming is mitigated by alternative algorithms explained in the following.

3.2.2 Random Forest

The Random Forest model is the regularization operation – enhancing the model's capability to generalize – applied to Decision Trees through an ensemble method [60]. The methodology addresses the greediness of the trees and moreover their sensitivity to small variances in the data. For instance, if a split occurs near the true variance boundary, it may lead to misclassification. Furthermore, Decision Trees alone struggle to model arbitrarily high complexity effectively [27].

The Random Forest concept involves training N distinct Decision Trees, each utilizing a random subset of the training data. As a result, the ensemble comprises a diverse collection of Decision Trees, each with its unique sequence of splitting criteria.

During prediction, the sample in question is evaluated by each Decision Tree in the ensemble. Since individual trees may produce varying classifications, a majority voting mechanism is employed to determine the final label. The class receiving the highest number of votes across the ensemble is selected as the prediction [27].

An illustration of the Random Forest classification process for a fruit basket is presented in Figure 14.

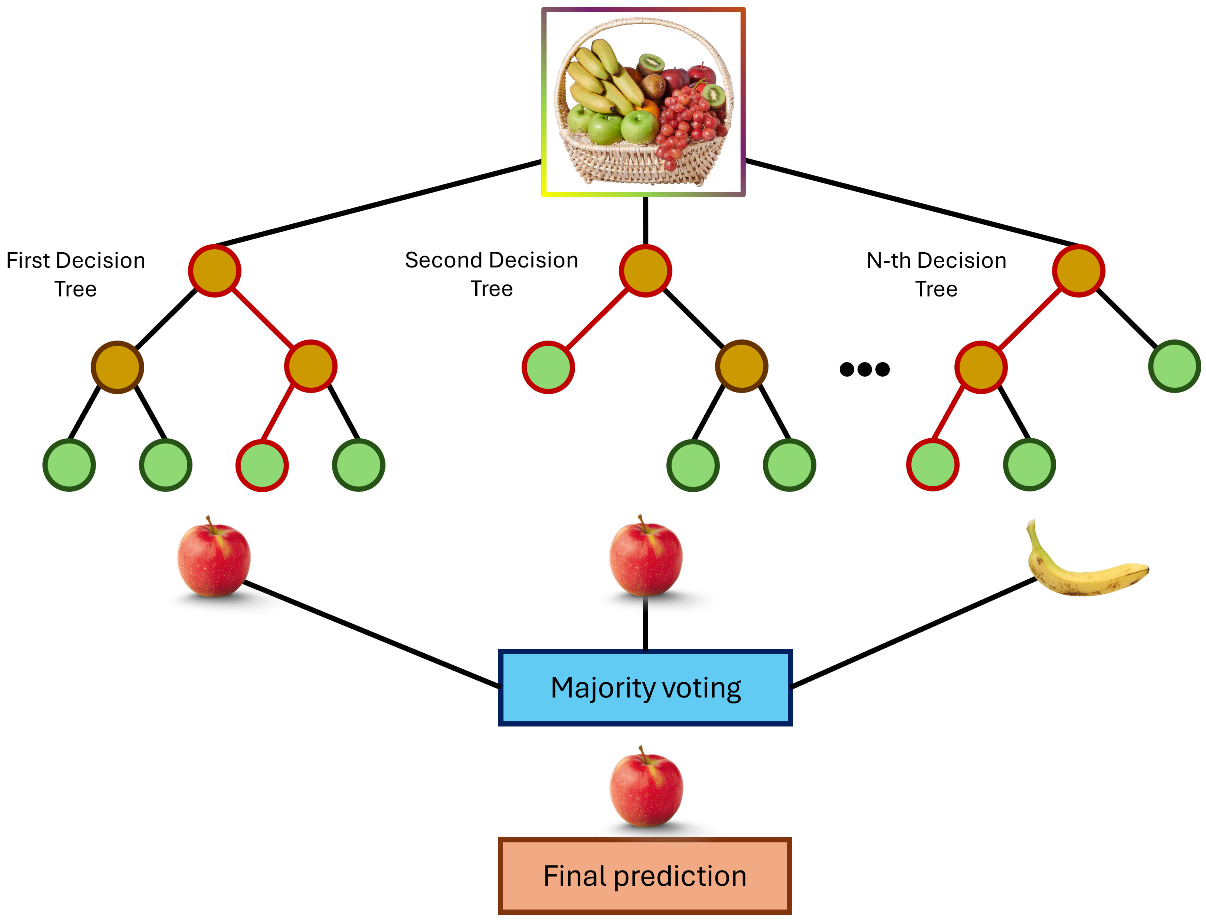


Figure 14: *Classification process on a Random Forest. A sample is classified by each of the N Decision Trees. For instance, one sample from the fruit basket might be identified as an apple by one tree and as a banana by another. The final decision is made using majority voting (adapted from [1]).*

In Figure 14, the Random Forest classification mechanism is depicted using a fruit basket as an illustrative example. The ensemble comprises N individually trained Decision Trees (cf. Chapter 3.2.1). Each tree independently processes the fruit sample and generates different predictions. The final classification is then determined through majority voting. For instance, if 89 trees classify the sample as an apple and 11 classify it as a banana, the ultimate prediction is apple.

3.2.3 Gradient Boosting Tree

Boosting refers to the process of iteratively building an ensemble of models, starting with a trivial base model that makes simplistic predictions. This is followed by training shallow Decision Trees (with a maximum depth typically ≤ 10) that focus on correcting the errors of the previous model. This procedure is repeated iteratively, where each subsequent model attempts to minimize the residual errors of the current ensemble, until either a predefined maximum number of trees is reached or the overall error ceases to improve.

The term "gradient" in Gradient Boosting reflects the optimization technique employed. Specifically, the model parameters are updated by following the descending gradient of the loss function, a fundamental concept in Deep Learning, which will be revisited in more detail in Chapter 3.3.2.

To illustrate the Gradient Boosting process, consider a regression example. The first step involves constructing a simple initial model for baseline predictions. For this example and without loss of generality the MSE (Equation (7)) is chosen as the loss function, simplified by replacing $1/n$ with $1/2$ for convenience in calculating gradients, as this alteration does not impact the eventual outcome. A logical initial prediction is the mean value of the target variable, calculated as follows:

$$F_0(X) = \arg\min_{\theta} \sum_{i=1}^{n} L(y_i, \theta) = y_{pred,0} \tag{9}$$

where F_0 denotes the initial model, X represents the input data, and $y_{pred,0}$ is the set of predictions. The notation $\arg\min$ of θ indicates: find a constant value θ that minimizes the loss $L(y_i, \theta)$ across all n observations y_i. For the MSE metric, this corresponds to predicting the mean of the target variable. An illustration of this initial prediction process is shown in Figure 15.

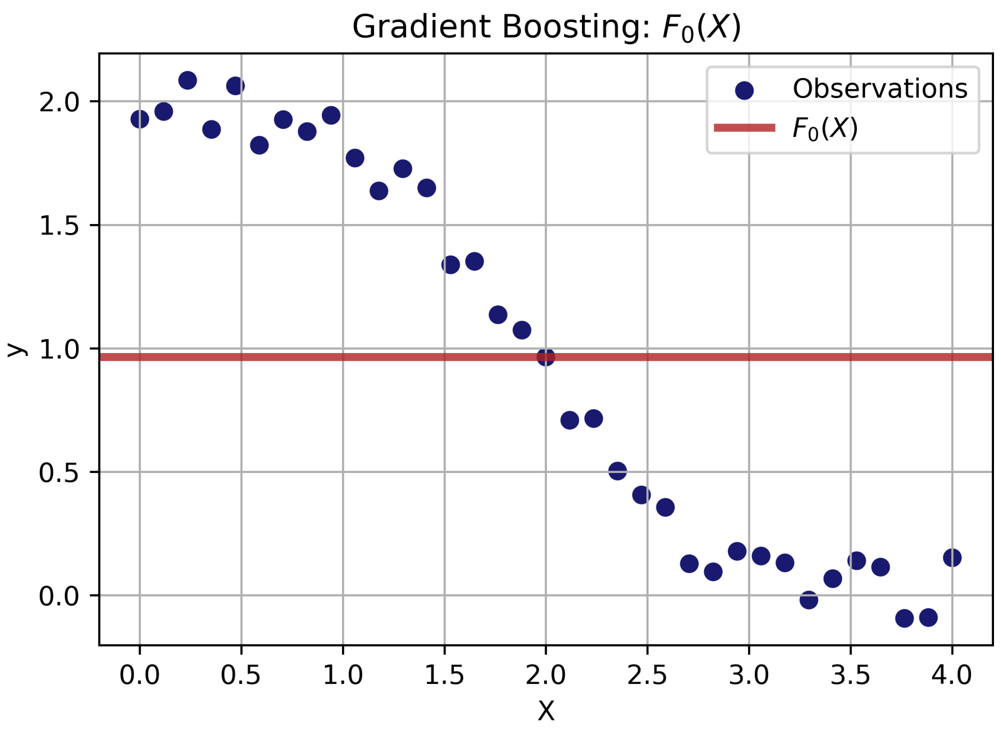


Figure 15: *Regression example illustrating the Gradient Boosting process. The base model F_0 predicts the mean of all observations.*

The next step involves calculating the gradients, which in this case are the residuals, representing the differences between the predicted values and the true targets as shown in the following equations:

$$
\begin{aligned}
r_{m,i} &= \frac{\partial L\big(y_i, F_m(X_i)\big)}{\partial F_m(X_i)} = \\
&= \frac{\partial}{\partial F_m(X_i)}\left[\frac{1}{2}\sum_{j=1}^{n}\big(F_m(X_j) - y_{\text{true},j}\big)^2\right] = \\
&= \frac{\partial}{\partial F_m(X_i)}\left[\frac{1}{2}(F_m(X_i) - y_{\text{true,i}})^2\right] = \\
&= F_m(X_i) - y_{\text{true,i}}
\end{aligned}
\tag{10}
$$

where $r_{m,i}$ is the residual of the i-th sample of the prediction of the m-th model. Inserted for the loss function $L\big(y_i, F_m(X_i)\big)$ is the slightly modified MSE from Equation (7). In Chapter 3.3.1 it will be seen that the MSE itself is a modification of the basic loss function Least Squares loss, discussed in Chapter 3.3.1. The explanation of the simplification from the second to the third line: the sum vanishes because the partial derivative is taken for the i-th sample, all the other terms become zero. An example of residual calculations for the initial model is illustrated in Figure 16.

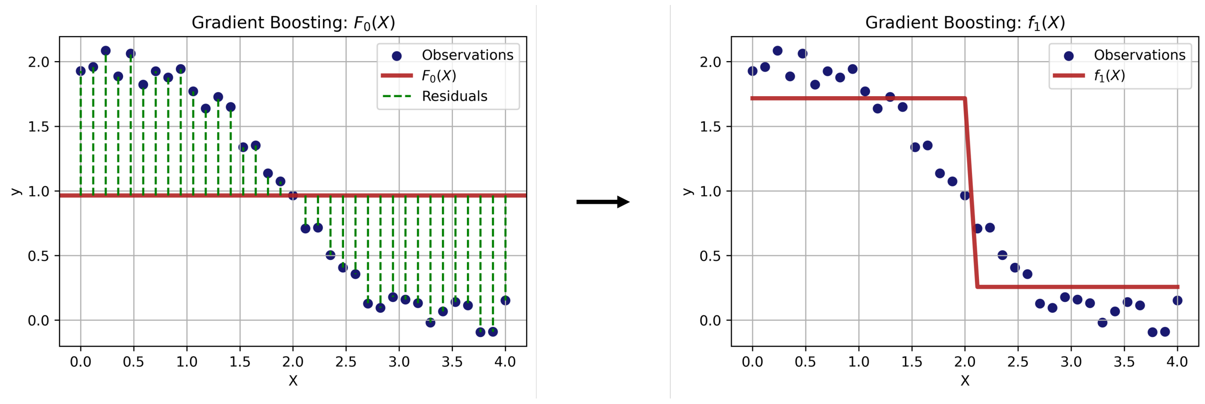
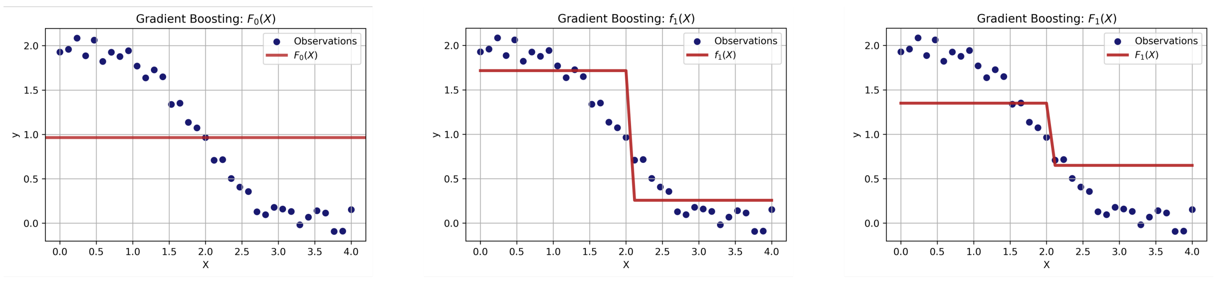


Figure 16: *Residuals are calculated to train a new weak learner f_1. In this case, the learner makes two splitting decisions.*

Based on the residuals, a new model f (typically a Decision Tree) is trained. This model, referred to as a weak learner in the context of Gradient Boosting, is then added to the ensemble.

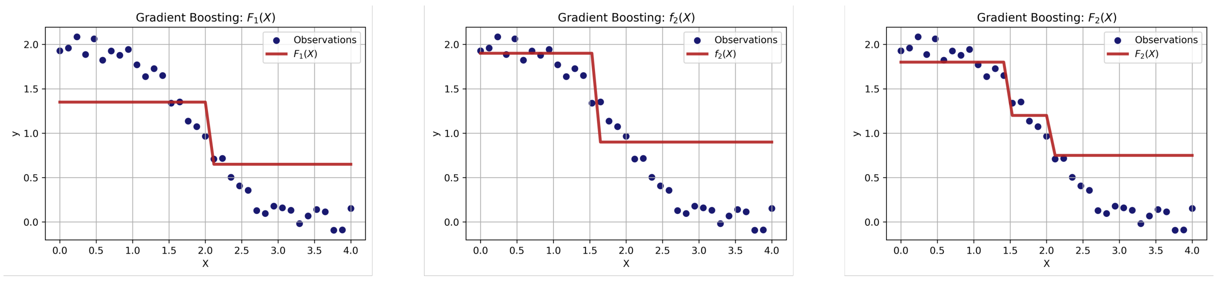
The resulting model, $F_0 + \nu f_1 = F_1$, incorporates the learning rate ν, which controls the influence of new learners. The impact of this step is illustrated in Figure 17.



$$F_0 \quad + \quad \nu f_1 \quad = \quad F_1$$

Figure 17: *A newly trained weak learner f_1 is added to the model F_0, with ν regulating the contribution of new information.*

The residuals of the updated ensemble F_1 are then calculated, and another weak learner f_2 is trained, further refining the model. This iterative process continues, as exemplified in Figure 18, until a desired accuracy is reached or the maximum number of trees is attained [14].



$$F_1 \quad + \quad \nu f_2 \quad = \quad F_2$$

Figure 18: *Adding additional weak learners progressively improves the model, resulting in finer divisions and better predictions.*

In practice, weak learners are trained with more sophistication than depicted here, and the classifier version of Gradient Boosting operates similarly to the regression case, with loss functions like Categorical Cross Entropy (softmax loss) replacing MSE. The softmax loss is discussed further in Chapter 3.3.

For this work, the Histogram-based Gradient Boosting Tree (Histo Boost) from the scikit-learn library was employed, as it is particularly suited for data sets containing more than 10,000 samples. The same working principles as described above apply for the Histo Boost with the exception that this algorithm bins the data into integer-valued histograms. This alteration significantly reduces the training time [67].

Additionally, the eXtreme Gradient Boosting Tree (XG Boost) was utilized. XG Boost distinguishes itself through a set of techniques like assigning low or even zero weights to weak learners which address less important features of the data. For example, in a fruit basket classification with apples and bananas, the "weight" feature might be down-weighted relative to "shape" or "color" [11][16][81]. Moreover, XG Boost accelerates training by enabling parallel processing of weak learners, reducing computation time [55].

In comparison to basic models such as Decision Trees and its ensemble method Random Forest, boosting algorithms excel by harnessing the flexibility and direct optimization capabilities of the gradient descent approach. This flexibility has been demonstrated as a significant advantage in numerous applications [26][55][69].

Adjustments for an efficient and effective gradient descent is also a main part of the following sections.

3.3 Deep Learning

Deep Learning aims to design a classifier or regressor model for a task akin to the human brain. The brain consists of neurons interconnected through synapses, which activate upon perceiving certain stimuli, memories, or images. The following chapters are informed by both the content and structure of [60], specifically up to, but excluding, Chapter 3.3.4 on Memory-Type Networks.

In accordance, a Deep Learning model (Neural Network) comprises units with adjustable parameters (neurons) interconnected by weighted paths (synapses). Each unit is equipped with non-linear activation functions, such as the Rectified Linear Unit (ReLU), which fires upon encountering specific stimuli. These principles are illustrated using a shallow network, as depicted in Figure 19.

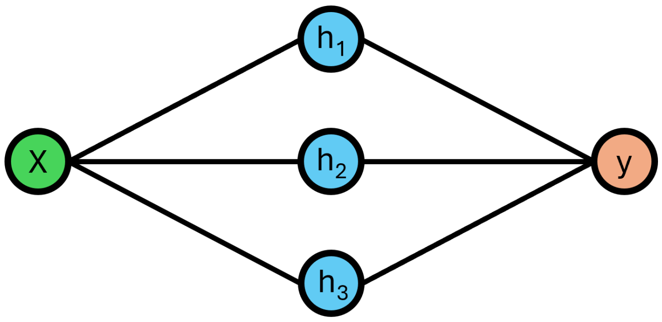


Figure 19: *Shallow networks consist of one hidden layer (containing the units), mapping the input X to the labels y. This example employs three hidden units h_i.*

The Figure 19 illustrates the functioning of a shallow neural network, applied in the following to a fictitious supervised regression task: The network processes scalar input and output values [23]. It consists of an input layer for the data X and an output layer predicting the labels y, connected by three hidden units h_i, governed by the following Equations (11):

$$\begin{aligned} h_1 &= a(\theta_{10} + \theta_{11} X) \\ h_2 &= a(\theta_{20} + \theta_{21} X) \\ h_3 &= a(\theta_{30} + \theta_{31} X) \end{aligned} \tag{11}$$

where a refers to a non-linear activation function, such as the ReLU function, defined as:

$$a(X) = \text{ReLU}(X) = \begin{cases} 0 & X < 0 \\ X & X \geq 0 \end{cases} \tag{12}$$

and the output/prediction y is calculated with the following equation pursuant to the initial basic Equation (3):

$$y = f(X, \Phi) = \phi_0 + \phi_1 h_1 + \phi_2 h_2 + \phi_3 h_3 \tag{13}$$

where Φ depicts a set of all the tuneable parameters in the model (cf. Figure 19), which, in this case includes: $\Phi = [\phi_0, \phi_1, \phi_2, \phi_3, \theta_{10}, \theta_{11}, \theta_{20}, \theta_{21}, \theta_{30}, \theta_{31}]$.

Details on training these parameters are elaborated in Chapter 3.3.2. For now, parameters are chosen arbitrarily to illustrate their role, as shown below:

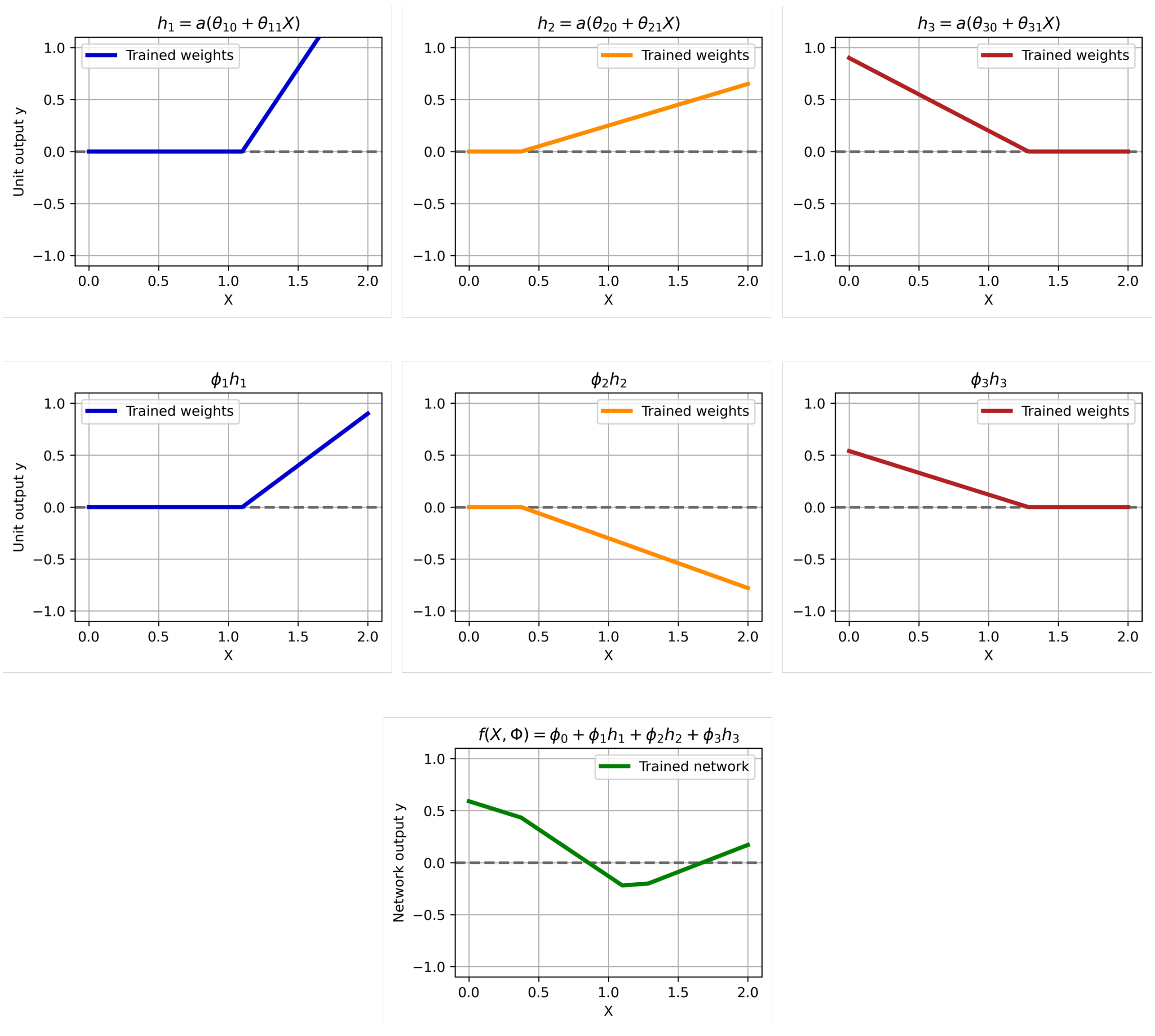


Figure 20: *Illustration of how fully trained neurons combine to generate the output. In the first row the ReLU function is applied. In the second row the weighting and in the third row the full network output.*

In Figure 20 the resulting prediction from adjusted network parameters is shown. As mentioned before, three neurons in the hidden layer (hidden units) are used, thus three different functions are assembled. The $\theta_{i,j}$ parameters establish the linear function in the figure and were chosen arbitrarily and serve as mere illustration.

The first row of Figure 20 shows the ReLU-applied parameters, which clip negative values, mimicking a neuron firing only after surpassing a threshold, akin to the human brain's processing of sensory input.

The second row highlights the weighting mechanism, which is critical. While the clipped functions approximate non-linear relationships, the weights determine the relevance and magnitude of each relationship of the units, influencing the final prediction. Negative weights are also possible, as seen in the example.

Finally, the third row demonstrates how the sum of the hidden units yields the network's prediction y for a given input X.

Although this example assumes scalar input of X and y, the same principles presented in Equations (11) and (13) also apply for higher-dimensional inputs and outputs. For X – given that the input layer of the network comprises the same dimensionality as X – Equation (11) is applied to each X value (e.g., X is a vector of three, then every vector component gets a parameter set of the hidden neurons h_1, h_2 and h_3).

The predictions can also generalize to a system of equations for arbitrary dimensions, as follows:

$$\begin{aligned} y_0 &= \phi_{00} + \sum_{d=1}^{D} \phi_{0d} h_d \\ y_1 &= \phi_{10} + \sum_{d=1}^{D} \phi_{1d} h_d \\ &\vdots \\ y_n &= \phi_{n0} + \sum_{d=1}^{D} \phi_{nd} h_d \end{aligned} \tag{14}$$

where D denotes the number of hidden units. The universal approximation theorem guarantees that, with a sufficient high number of hidden units D, the model can approximate any arbitrarily complex function. However, increasing the number of hidden units can be computationally expensive. Instead, depth is introduced by stacking hidden layers, resulting in Deep Neural Networks.

To derive the Deep Neural Networks' mechanisms, two examples of Figure 19 are concatenated:

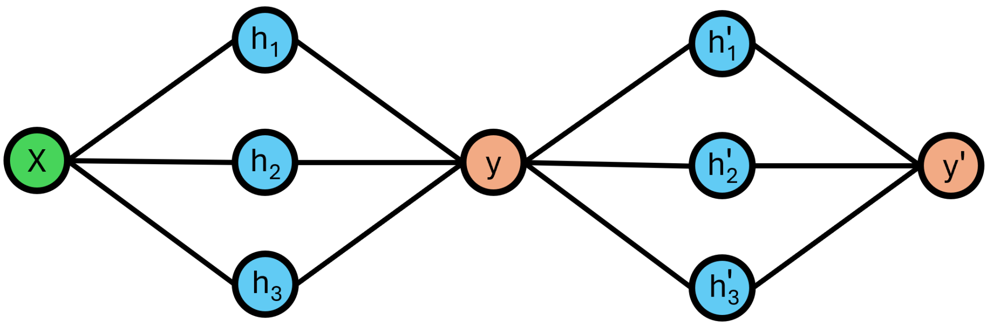


Figure 21: *Two shallow networks are combined. The output of the first network serves as the input to the second.*

The output of the first layer becomes the input to the hidden units $h_i^{'}$ of the second layer, so the resulting equations for these units in accordance to Equation (11) and the subsequent insertion of Equation (13) provide:

$$\begin{aligned}
h_1^{'} &= a(\theta_{10}^{'} + \theta_{11}^{'} y) = a(\theta_{10}^{'} + \theta_{11}^{'}\phi_0 + \theta_{11}^{'}\phi_1 h_1 + \theta_{11}^{'}\phi_2 h_2 + \theta_{11}^{'}\phi_3 h_3)\\
h_2^{'} &= a(\theta_{20}^{'} + \theta_{21}^{'} y) = a(\theta_{20}^{'} + \theta_{21}^{'}\phi_0 + \theta_{21}^{'}\phi_1 h_1 + \theta_{21}^{'}\phi_2 h_2 + \theta_{21}^{'}\phi_3 h_3)\\
h_3^{'} &= a(\theta_{30}^{'} + \theta_{31}^{'} y) = a(\theta_{30}^{'} + \theta_{31}^{'}\phi_0 + \theta_{31}^{'}\phi_1 h_1 + \theta_{31}^{'}\phi_2 h_2 + \theta_{31}^{'}\phi_3 h_3)
\end{aligned} \tag{15}$$

which can be concisely rewritten as

$$\begin{aligned}
h_1^{'} &= a(\psi_{10} + \psi_{11} h_1 + \psi_{12} h_2 + \psi_{13} h_3)\\
h_2^{'} &= a(\psi_{20} + \psi_{21} h_1 + \psi_{22} h_2 + \psi_{23} h_3)\\
h_3^{'} &= a(\psi_{30} + \psi_{31} h_1 + \psi_{32} h_2 + \psi_{33} h_3)
\end{aligned} \tag{16}$$

with $\psi_{i0} = \theta_{i0}^{'} + \theta_{i1}^{'}\phi_0$ and $\psi_{ij} = \theta_{ij}^{'}\phi_i$ for $j = 1, ..., n$. By examining Equations (16), it becomes clear that every unit of the second layer depends on weighted input of the preceding layer, creating a Deep Neural Network:

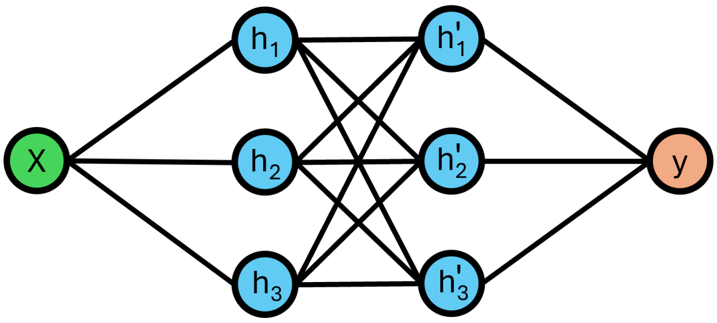


Figure 22: *Deep networks consist of multiple hidden layers. Their operation is derived from the same principles of combined shallow networks.*

The principles of stacking shallow networks to form Deep Neural Networks have been outlined. It is important to note that X and y as previously described, can also be chosen to have arbitrary dimensions, and the described equations remain consistent. Similarly, the number of hidden layers and their respective units can, in principle, also be selected arbitrarily high.

The advantage of deeper networks in approximating complex functions more efficiently stems from their ability to create more mapping regions through additional hidden layers. These layers add complexity to the combined outputs of the network, enabling it to capture intricate patterns in the data. As illustrated in Figure 20, a shallow network may define per hidden neuron only two regions: the slope and the clipped values. While deeper networks introduce more regions by taking the preceding layers' units into account, offering substantial flexibility to model complex tasks. A detailed analysis and formal proof of the efficiency of Deep Neural Networks can be found in [60].

Determining the optimal number of hidden layers and units for a network is inherently subjective, as there is no definitive recipe to ensure perfect convergence to the true relationship between X and y. This process, known as hyperparameter tuning, involves adjusting parameters that are not updated during training itself. The evaluation of whether a specific architecture is appropriate requires iterative testing and refinement/optimization, often relying heavily on the experience of the AI-user. This work also explores additional strategies to support training, which are discussed and applied in detail in Chapter 4.3.

With the fundamental operations of Deep Learning now established, it is essential to define a metric to evaluate their performance – a kind of measure for interpreting the network's effectiveness. This topic will be addressed in the following section.

3.3.1 Loss function

To evaluate the suitability of a model's architecture for a given task during training, a metric is required to monitor its performance.

An ML model aims to map X to y, essentially approximating the real-world function. The loss function $L(y_{\text{pred},i}, \Phi)$, evaluates how well the model performs by comparing its predictions with actual values. This function depends on the set of parameters Φ that the model uses to compute predictions.

To establish a meaningful criterion for assessing a model's performance, the idea is that the model predicts the conditional probability of y for given X, denoted as $Pr(y|X)$. In essence, the model is tasked with computing the parameters (e.g., mean and variance) of the probability distribution for y.

The distribution (highlighted in violet in Figure 23) is derived from the distribution parameters θ (:= mean and variance), calculated by the model $f(X_i, \Phi)$ describing the conditional probability $Pr(y|\theta)$. Thus, for a given input sample X_i the model should assign a high probability to the corresponding output y_i. Taking the product of all N possible X and y pairs results in the maximum likelihood criterion:

$$\begin{aligned}
\hat{\Phi} &= \arg\max_{\Phi} \left(\prod_{i=1}^{N} Pr(y_i|X_i) \right) \\
&= \arg\max_{\Phi} \left(\prod_{i=1}^{N} Pr(y_i|\theta_i) \right) \\
&= \arg\max_{\Phi} \left(\prod_{i=1}^{N} Pr\big(y_i|f(X_i, \Phi)\big) \right)
\end{aligned} \tag{17}$$

where $\hat{\Phi}$ is considered to be the set of parameters for the model, that maximizes the likelihood out of the observations. This formulation (Equation (17)) assumes the data to be independent and identically distributed.

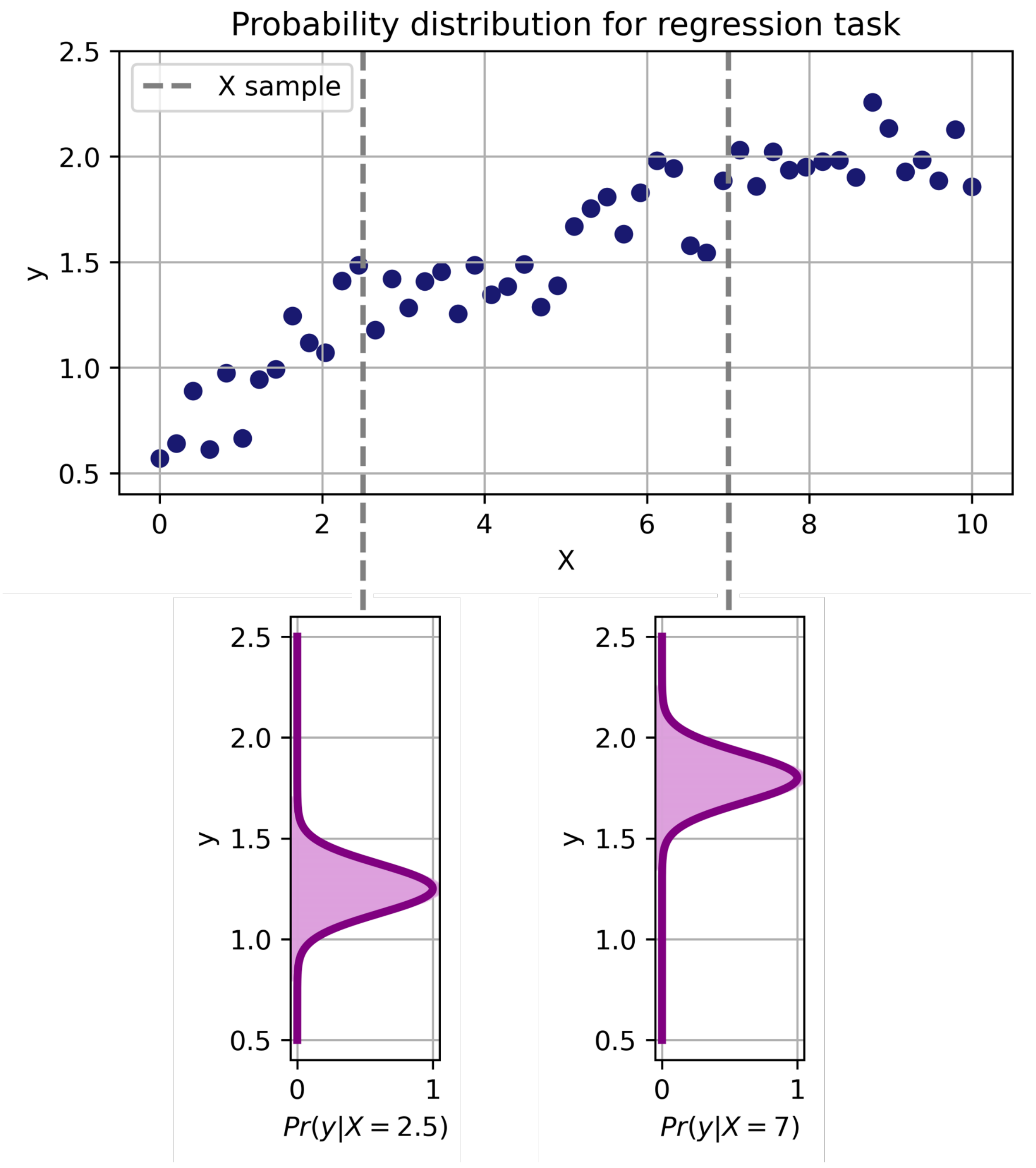


Figure 23: *A meaningful performance metric for training neural networks involves predicting a conditional probability and its parameters. For instance, in the plotted regression task, samples at* $X = 2.5$ *following a normal distribution with a mean of 1.25 and variance of 0.02.*

Due to the limitations of finite precision arithmetic in computational calculations, extremely small numbers (close to machine epsilon: difference between 1.0 and the next larger floating point number) can be problematic (e.g., the product of $Pr\big(y_i|f(X_i, \Phi)\big)$

might get very small for improbable events). Consequently, the logarithm of the likelihood is employed, preserving the characteristics of the probability while avoiding numerical issues. The logarithmic transformation also ensures that the maximum likelihood remains the maximum in the logarithmic domain, albeit with a different scaling.

In ML, it is conventional to minimize the loss function. Thus, the maximization of likelihood is transformed into a minimization problem by multiplying (-1). This leads to the negative log-likelihood criterion, expressed as the loss function $L(y_i, \theta)$:

$$\begin{aligned}\hat{\Phi} &= \arg\min_{\Phi} \Bigg(-\log\Bigg[\prod_{i=1}^{N} Pr\big(y_i | f(X_i, \Phi)\big)\Bigg]\Bigg) \\ &= \arg\min_{\Phi} \Bigg(-\sum_{i=1}^{N} \log\Big[Pr\big(y_i | f(X_i, \Phi)\big)\Big]\Bigg) \\ &= \arg\min_{\Phi} \big(L(y, \theta)\big)\end{aligned} \tag{18}$$

Applying Equation (18) to the standard distribution:

$$\begin{aligned}\hat{\Phi} &= \arg\min_{\Phi} \Bigg(-\sum_{i=1}^{N} \log\Bigg[\frac{1}{\sqrt{2\pi\sigma^2}} \exp\Bigg(-\frac{\big(y_{\text{true},i} - f(X_i, \Phi)\big)^2}{2\sigma^2}\Bigg)\Bigg]\Bigg) \\ &= \arg\min_{\Phi} \Bigg(-\sum_{i=1}^{N} \log\Bigg[\frac{1}{\sqrt{2\pi\sigma^2}}\Bigg] - \frac{\big(y_{\text{true},i} - f(X_i, \Phi)\big)^2}{2\sigma^2}\Bigg) \\ &= \arg\min_{\Phi} \Bigg(\sum_{i=1}^{N} \big(y_{\text{true},i} - f(X_i, \Phi)\big)^2\Bigg)\end{aligned} \tag{19}$$

where in the second line terms independent of Φ are omitted, and $1/(2\sigma^2)$ being a scaling factor, was disregarded, cumulating in the loss function Least Squares loss:

$$L(y_i, \theta) = \sum_{i=1}^{N} \big(y_{\text{true},i} - f(X_i, \Phi)\big)^2 \tag{20}$$

which is the consequence from the assumption that X is independent and identically distributed, where the predicted label corresponds to the mean calculated by the model, representing the most probable y-value (termed as y_{pred}). In Chapter 3.2.1 the MSE and MAE loss (Equations (7) and (8)) were presented, which are just modifications of the Least Squares loss (Equation (20)).

Above described loss functions are applied to regression tasks. A different loss function is needed for classification. Assuming that y_i belongs to one of K classes, where the parameters $\lambda_1, \lambda_2, ..., \lambda_K$ represent the probabilities associated with each class. The constraints require each parameter to lie within $[0, 1]$ and all the parameters to sum up to 1, ensuring a valid probability distribution.

These constraints are enforced by the softmax function (in the following equation X_K represents one sample with K outputs, indicating the class probabilities):

$$\text{softmax}_K(X) = \frac{\exp(X_K)}{\sum_{k=1}^{K} \exp(X_k)} \tag{21}$$

Generally, the AI-model predicts probabilities for the classes' affiliation for each X sample, which might not follow the constraints explained above, so Equation (21) is utilized to remedy these problems. The output of the softmax function is the normalized probability of one sample – i.e., K values representing each class probability, for illustration: A classification with three classes will display a prediction consisting of a vector of three normalized probability values indicating the sample's class affiliation.

The likelihood that X has the label y as class k, calculated by the model $f(X_i, \Phi)$ is given by:

$$Pr(y = k|X) = \text{softmax}_K\big(f(X_i, \Phi)\big) \tag{22}$$

Incorporating the softmax function into the loss function results in the following expression:

$$\begin{aligned} L(y_i, \theta) &= -\sum_{i=1}^{N} \log \Big[\text{softmax}_K\big(f(X_i, \Phi)\big)\Big] \\ &= -\sum_{i=1}^{N} \left(f_K(X_i, \Phi) - \log \left[\sum_{k=1}^{K} \exp\left(f_k(X_i, \Phi)\right) \right] \right) \end{aligned} \tag{23}$$

The predicted class label corresponds to the maximum value of the softmax output (Equation (23)), representing the class with the highest probability. This loss function, known as (multi-class) cross-entropy loss, reflects the principle of cross-entropy by quantifying the differences between two probability distributions. It is also commonly referred to as the softmax loss for its explicit use of the softmax function.

The loss functions discussed above, designed for both regression and classification tasks, serve as sensible metrics for evaluating model performance during training.

3.3.2 Model training

In the preceding chapter, loss functions were introduced as metrics for evaluating model performance. A logical progression involves reinforcing favorable outcomes – where the parameters Φ effectively reduce the loss – while discouraging unfavorable results – instances of high loss. Gradient descent methods are designed precisely to serve this purpose, but how is this descent actually defined? To illustrate, consider a linear regression model characterized by two parameters: the slope m and the offset c. While the focus here is regression, gradient descent is equally applicable to classification problems.

There are infinite possible combinations of the parameters m and c, but one specific pair exists – unknown a priori – that describes the relation between input X and output y best. This problem can be represented in a three-dimensional space where two dimensions correspond to m and c, and the third dimension represents the value of the loss function (e.g., the Mean Squared Error for regression). The collection of all possible parameter combinations forms the "loss landscape", with the global minimum representing the best possible solution.

Since regression tasks operate in continuous space, and the optimal parameters are unknown beforehand, m and c are iteratively refined. This is achieved by slightly adjusting the parameters to follow the direction of the descending gradient of the loss function – hence the term "gradient descent". Visually, this corresponds to the parameters progressively moving deeper into the valleys of the loss landscape, as depicted in Figure 24.

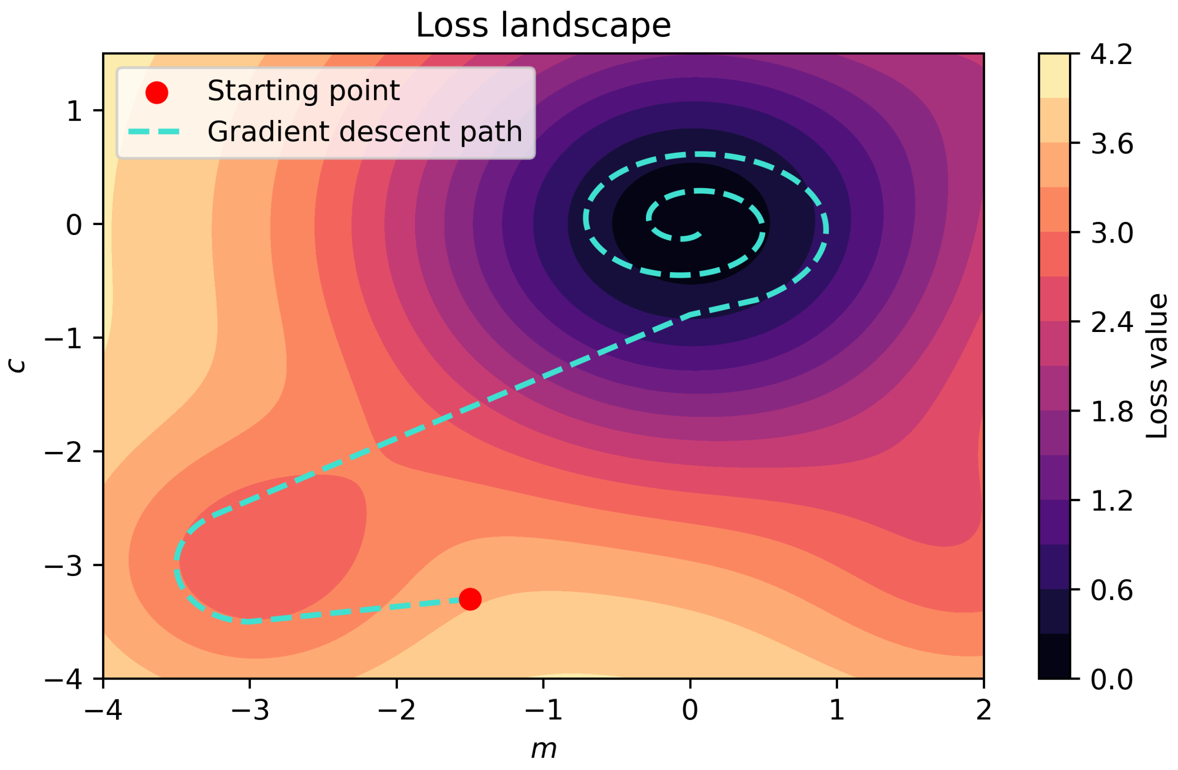


Figure 24: *Illustrative example of a gradient descent. The initial values of* $m = -1.5$ *and* $c = -3.3$ *lead to a descent in the local minimum* $(-3, -3)$*. However, the global minimum lies elsewhere. A proper gradient descent set up (i.e., Stochastic Gradient Descent which is explained in the following) ensures exploration of the loss landscape, eventually guiding the parameters toward the global minimum, as indicated by the spiral trajectory into the steepest valley.*

This process can be concisely expressed as:

$$\Phi_{\text{new}} = \Phi - \alpha \cdot \frac{\partial L}{\partial \Phi} \tag{24}$$

where the term $\frac{\partial L}{\partial \Phi}$ denotes the gradient of the loss with respect to each model parameter:

$$\frac{\partial L}{\partial \Phi} = \left[\frac{\partial L}{\partial \theta_0}, \frac{\partial L}{\partial \theta_1}, \dots, \frac{\partial L}{\partial \theta_M}, \frac{\partial L}{\partial \phi_0}, \frac{\partial L}{\partial \phi_1}, \dots, \frac{\partial L}{\partial \phi_N} \right] \tag{25}$$

These equations describe the update mechanism for refining model weights and parameters based on the descending loss, improving the approximation of the underlying relation between X and y. The parameter α, referred to as the learning rate, determines the step size, effectively regulating how much each gradient influences the new parameter values.

The loss landscape, as illustrated in Figure 24, may contain saddle points or local minima where gradient descent could stall. To mitigate this, Stochastic Gradient Descent (SGD) introduces noise during the optimization process by computing updates using randomly sampled subsets of X, referred to as batches. These "noisy" steps may increase the chance of escaping local minima and continuing the search for the global minimum. A complete pass through all batches constitutes an epoch.

Using a fixed step size for gradient descent can result in overshooting the minimum or unnecessary slow progress in flat regions of the loss landscape. Adaptive Moment Estimation (Adam) addresses these issues through the following equations:

$$\begin{aligned} m_{t+1} &= \beta \cdot m_t + (1-\beta) \cdot \frac{\partial L(\Phi_t)}{\partial \Phi} \\ v_{t+1} &= \gamma \cdot v_t + (1-\gamma) \cdot \left(\frac{\partial L(\Phi_t)}{\partial \Phi}\right)^2 \end{aligned} \tag{26}$$

$$\begin{aligned} \tilde{m}_{t+1} &= \frac{m_{t+1}}{1-\beta^{t+1}} \\ \tilde{v}_{t+1} &= \frac{v_{t+1}}{1-\gamma^{t+1}} \end{aligned} \tag{27}$$

which are incorporated into the parameter update as:

$$\Phi_{t+1} = \Phi_t - \alpha \cdot \frac{\tilde{m}_{t+1}}{\sqrt{\tilde{v}_{t+1}} + \epsilon} \tag{28}$$

The variable t represents the current optimization time step, whereas $t+1$ denotes the upcoming step. Equation (28) normalizes the gradients and also introduces the adaptive descending steps (the fraction in the equation): The m defines the overall direction of the step, while v scales with the squared gradient, consequently adjusting the step size: flat regions result in small gradient values in the denominator, which are getting squared, leading to an overall large step size, while steep loss areas cause relatively big values in the gradient, therefore, executing more cautious steps to prevent overshooting. The ϵ is a constant for avoiding division by zero.

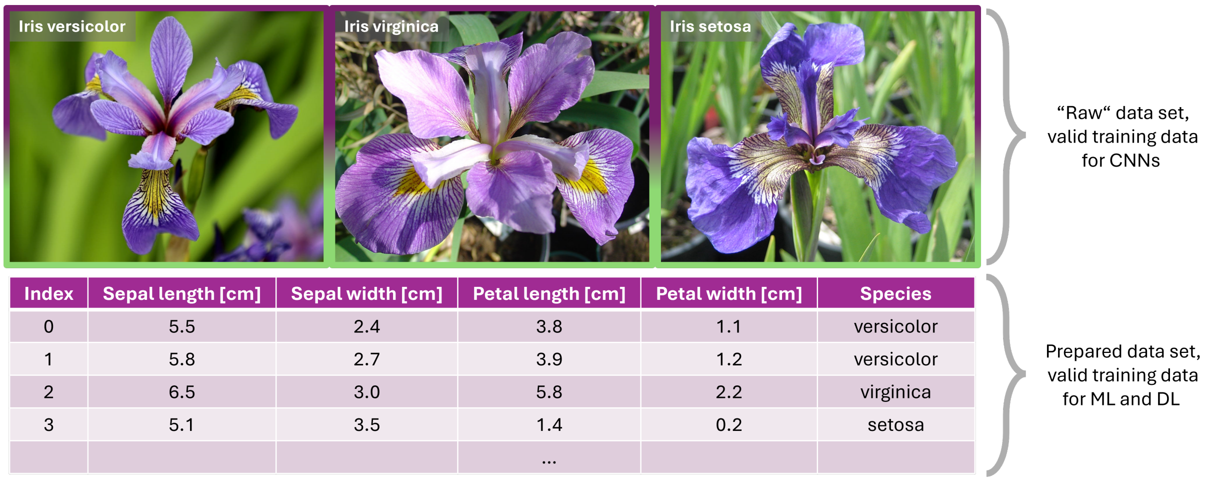
Additionally, the terms in Equation (26) with momentum parameters β and γ – ranging from $[0, 1[$ – provide control over the weighting of the directionality and scaling depending on the current time step. Having the momentum parameters in the denominator raised to the power of the current time step has the effect of generally bigger step sizes in the early stages of the descent and progressively finer steps as convergence nears. Adam is widely regarded as a highly effective optimizer for deep learning training.

A typical model, such as the Convolutional Neural Network (CNN) depicted in Figure 41 (to be presented in Chapter 4.3), requires the adjustment of approximately 10^6 parameters during training. Calculating the gradients for each parameter at every optimization step necessitates an efficient mechanism, such as the backpropagation algorithm. Furthermore, the initial parameter values must be carefully chosen: overly large initial gradients can lead to the exploding gradient problem, while excessively small gradients can cause vanishing gradients, both of which hinder the training process. For further details of the backpropagation and model parameter initialization, the reader is referred to [60] and [53].

With the foundational mechanisms now clarified, the focus will shift to specialized deep learning architectures.

3.3.3 Convolutional Neural Network

Convolutional Neural Networks (CNNs) are among the most widely employed deep learning architectures, finding applications across diverse domains such as image recognition and speech processing. Their principal advantage lies in the automated extraction of features through convolutional layers [77]. This process eliminates the need for manual feature engineering, as exemplified with the classic Iris data set [68]: Instead of requiring features such as petal and sepal dimensions to be explicitly provided, a CNN can infer the relevant features directly from raw labeled images of the iris species, as depicted in Figure 25.


Iris versicolor
Iris virginica
Iris setosa
"Raw" data set, valid training data for CNNs
Prepared data set, valid training data for ML and DL

Index	Sepal length [cm]	Sepal width [cm]	Petal length [cm]	Petal width [cm]	Species
0	5.5	2.4	3.8	1.1	versicolor
1	5.8	2.7	3.9	1.2	versicolor
2	6.5	3.0	5.8	2.2	virginica
3	5.1	3.5	1.4	0.2	setosa
			...		

Figure 25: *CNNs can process raw image data to automatically derive features, unlike the Iris data set (depicted below), where features such as sepal and petal length are manually recorded. With labeled images alone, CNNs can classify iris species.*

A convolutional layer consists of kernels, also known as filters, which are small matrices (e.g., 2×2, 3×3, or 5×5) with dimensions smaller than those of the input data. These kernels carry weights that are optimized during training, enabling them to extract meaningful features from the input. Although the focus here is on two-dimensional kernels for image data, one-dimensional and three-dimensional convolutional layers also exist, suitable for applications like time-series data and videos, respectively.

The convolution operation involves sliding the kernel across the input data matrix both vertically and horizontally, performing a dot product at each position to produce an output known as the feature map. Figure 26 illustrates this process.

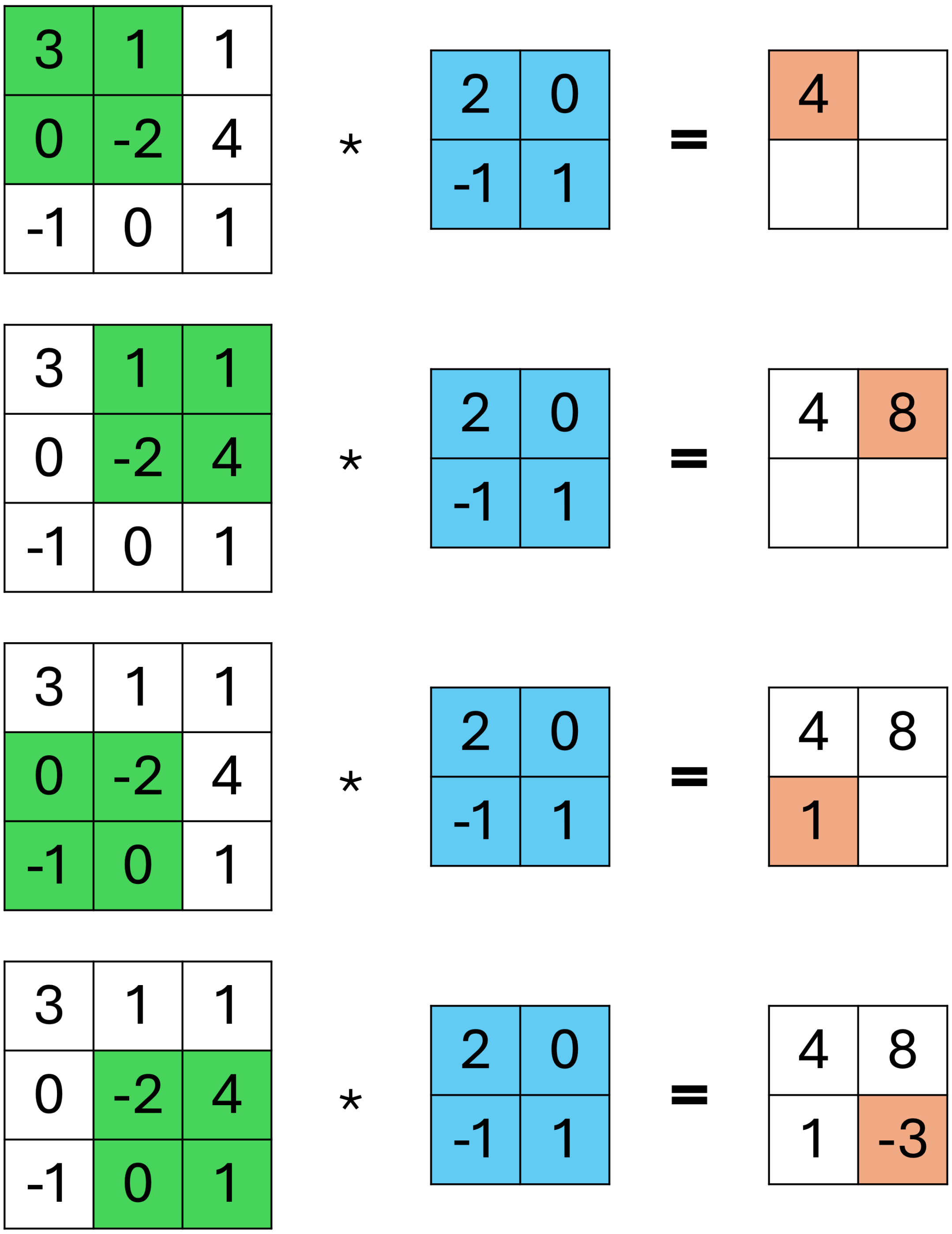


Figure 26: *Illustration of the convolution operation. The left matrices represent an example input (e.g., image pixels), the middle matrices are convolutional filters (e.g., 2×2 kernels), and the result is computed through dot products. For instance, for the top-left window, the operation computes* $3 \cdot 2 + 1 \cdot 0 + 0 \cdot (-1) + (-2) \cdot 1 = 4$*. This process allows a CNN to distill an image into its fundamental components (e.g., a house into a rectangle and a triangle).*

Additional parameters further refine the convolution process. For example, padding allows the kernel to slide beyond the boundaries of the input, resulting in larger feature maps. The stride parameter enables the kernel to skip rows or columns, yielding smaller feature maps. These parameters provide flexibility in controlling the dimensions and granularity of the extracted features.

Non-linear activation functions, such as ReLU (Rectified Linear Unit), are often applied after the convolution operation. ReLU introduces non-linearity, enabling the network to learn complex patterns by clipping negative values, as shown in Equation (12) and illustrated in Figure 20.

Following the convolutional layers, pooling layers are typically used to reduce the dimensions of the feature maps, thereby retaining only the most salient information. Figure 27 illustrates the pooling operation, where 2×2 windows are used for either average pooling (computing the average value in each window) or max pooling (selecting the maximum value in each window). This operation further reduces computational complexity and accelerates training.

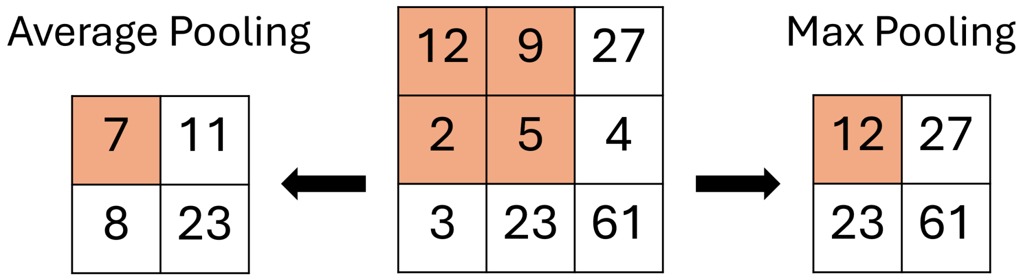


Figure 27: *Illustration of pooling layers. The input feature map is divided into 2×2 windows, and either average pooling or max pooling is applied. This operation reduces dimensionality while preserving critical information.*

The architecture of convolutional layers often includes multiple kernels, typically in powers of two. For instance, a common configuration might involve 64 filters, each of size 3×3 (which is generally denoted as convolutional layer of 3×3×64).

CNN architectures excel at distilling input data into their most fundamental features. For example, an image of a house can be reduced to geometric primitives such as a rectangle and a triangle, enabling the network to capture the essence of the input efficiently.

After the convolutional and pooling layers, a flattening layer converts the resulting feature maps into a one-dimensional vector. This vector serves as input to a fully connected dense network, similar to those described earlier in this work (Figure 22). The term "dense" in this context refers to the network's architecture, where each neuron is linked to all neurons in the preceding layer.

Convolutional Neural Networks have consistently proven to be one of the most reliable and effective deep learning architectures, as also demonstrated in this thesis. The following sections will discuss an additional type of a deep learning model.

3.3.4 Memory-type Networks

These types of networks were investigated at a relatively late stage of this work. Typically, such networks are employed for sequential data, including text, speech, video, and properties that evolve over time, due to their ability to retain characteristics across multiple sequences and samples. It was hypothesized that this capability could be particularly advantageous for tasks requiring the retention and propagation of information, such as the 11×11 pixel analysis (Chapter 4.6) or the comparison of cathode and anode signals (Chapter 4.5). However, while these networks demonstrated reasonable performance in one specific task, their overall performance was inferior to that of Convolutional Neural Networks (CNNs). Consequently, this chapter does not delve deeply into the details of their theory. For a more comprehensive understanding, the reader is encouraged to consult [66] and [28], which also provided the foundational literature for the subsequent explanations.

Recurrent Neural Networks (RNNs) are specialized architectures designed to utilize a hidden state for each unit carrying past training information, enabling the network to directly influence weight updates based on previous samples [66]. This is a key distinction from conventional dense networks.

Unfortunately, RNNs are particularly susceptible to the vanishing or exploding gradient problem due to their hidden state mechanism. During training, the gradient from the hidden states is multiplied with the loss gradient, potentially resulting in extremely small or excessively large values that exceed the floating-point precision limits of computational systems [28][49]. This issue can lead to catastrophic results, as it may have happened in the results displayed in Figure 70.

To address these challenges, Long Short-Term Memory (LSTM) networks were introduced.

LSTMs retain the fundamental concept of carrying information from previous samples into the current computation but refine it by allowing dynamic adjustment of what and how much information is utilized in each cell in the layer at each training step. This is achieved through the incorporation of various gates, which weight and scale the influence of prior samples on the current one [28]. An LSTM cell is depicted in Figure 28.

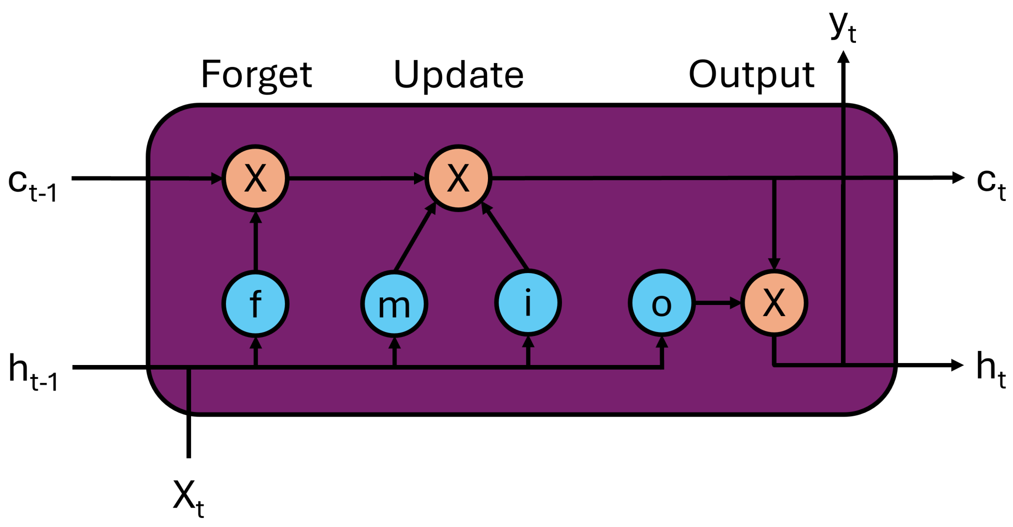


Figure 28: *An LSTM cell comprises gates that regulate the flow of information from previous calculations c_{t-1}, outputs h_{t-1}, and new input X_t. The retained information is updated within the cell for future computations, with updates representing a weighted combination of past predictions and new inputs (adapted from [49]).*

In Figure 28, c represents the memory cells, with $t-1$ denoting the output and sample from the preceding step. The hidden state h conveys additional information from past samples and also provides the label prediction y_t. Each gate in the LSTM serves a distinct purpose: The forget gate f determines how much of the retained information should be preserved. The memory cell m stores long-term information derived from the hidden states, effectively blending old and new memories (X_t). The input gate i regulates the extent to which new information is mixed with the content from the memory cell. Finally, The output gate o adjusts the amount of new information passed to the hidden state for subsequent LSTM cells [21]. All gate parameters are optimized during the training process.

A comprehensive view of how LSTM layers are composed and operate is presented in Figure 29. This figure illustrates the relationship between the input data X, the outputs y, and the internal structure of the LSTM [49]. Notably, if the c_i components are omitted, the diagram effectively represents the structure of a standard RNN.

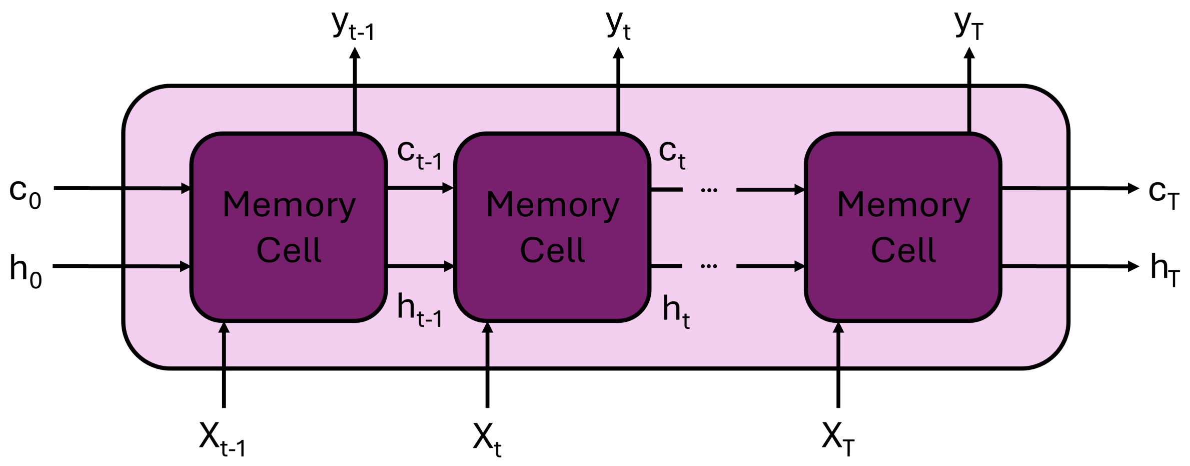


Figure 29: *A representation of an LSTM layer showing the arrangement of the cells relative to the input X and output y. By omitting the c_i components, this diagram also serves to illustrate the working principles of RNNs (adapted from [49]).*

The fundamentals of Machine Learning (ML) and Deep Learning (DL) have now been covered, including all models utilized in this work. In the subsequent chapters, the discussed methodologies will be applied to real-world data acquired from a high-energy detector.

4 Experimental framework & methodology

The objective of this thesis lies in the question whether AI can replace traditional methods in high energy detector data analysis. The working principles in AI where discussed in previous chapters, now these models are going to be applied.

To investigate the thesis query, the company IDEAS, an industrial partner in the i-RASE project, provided recordings from a CZT detector measurement campaign. This detector – featured in their product portfolio [32] – was calibrated using a Caesium-137 (Cs-137) source, which generated the received data.

4.1 High energy radiation recordings

The GDS-100 utilizes a CZT semiconductor for gamma-ray spectrometry capable of continuous sampling for providing the pulse shape of incoming radiation in a pixelated field of vision, which extend over a 11 by 11 array, enabling to localize the point of impact on a 2D plane.

The CZT crystal is located between one large cathode and 121 anodes, in accordance with the 11×11 pixel array design, upon which a strong electric field is applied, either at 2000 V or 3000 V. The recorded data were obtained under a high voltage of 2000 V. All the physics of a semiconductor introduced in Chapter 2 also apply here: When a charged particle hits the detector, electron-hole pairs are formed and drawn toward their respective electrodes. The drift of these charge carriers within the electric field induces a current, resulting in a rise in pulse amplitude, which is then recorded by charge-sensitive electronics. As previously mentioned, the signals are continuously sampled, but are digitized only when the amplitude exceeds a certain internal threshold.

The resulting waveform is described in units of least significant bits (LSB), a product of the detector's internal Analog-to-Digital conversion (ADC). LSB represents the number of distinct levels the detector's ADC unit can distinguish – in other words, it is a measure of the resolution of the detector [51]. An ADC with an n-bit resolution divides the detector's operational energy range into 2^n discrete energy levels. Consequently, a measurement in LSB units can be converted into a physical energy unit (e.g., electron volts) by

$$\text{value in LSB} \cdot \frac{\text{energy range detector in eV}}{2^n \text{ (with } n \text{ bit resolution)}} = \text{value in eV} \tag{29}$$

Amplitudes are sampled over 160 cell values and given in LSB units, with each cell corresponding to 20 nanoseconds (ns). This setup provides a total time window of 3200 ns for each signal's amplitude.

The measurement setup featured a Cs-137 source in the form of a disk, positioned directly above the detector. The detector was enclosed within an aluminum housing to shield it from extraneous light sources.

The data set of detector measurements contains information on the event time in GPS time format, x and y position on the 11×11 pixel grid and the aforementioned 160 cells. The Figures 30 and 31 illustrate examples of a single-event and a multi-event, respectively. In this thesis, an "event" is defined as all triggered signals occurring under the same timestamp. Each event consists of one cathode signal and one or more anode signals. If only one anode signal is recorded, the event is classified as a single-event, otherwise it is classified as a multi-event.

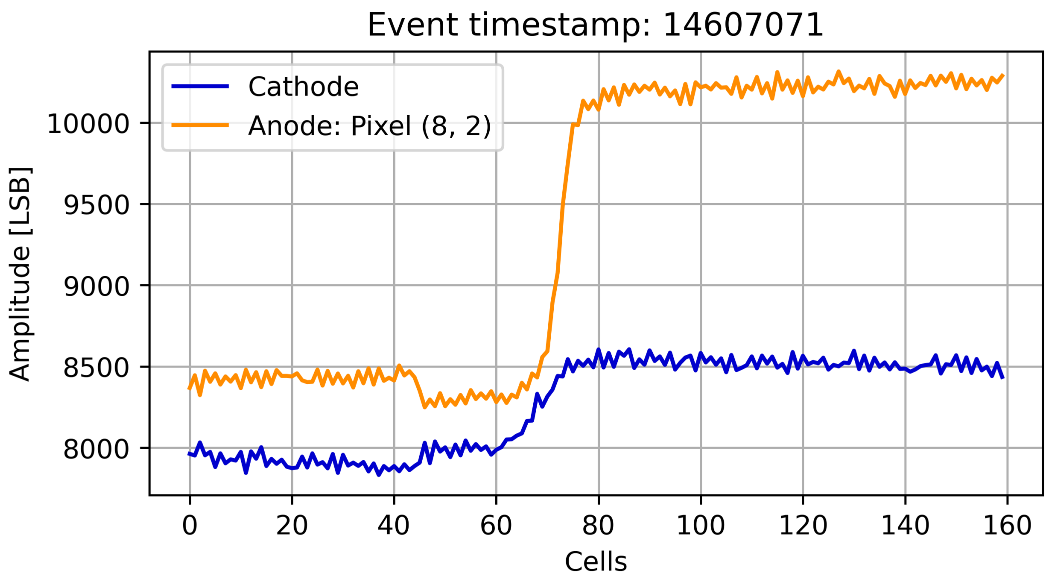


Figure 30: *Single-event without corrections from GDS-100 recordings. The resulting waveform of an interaction with the detector.*

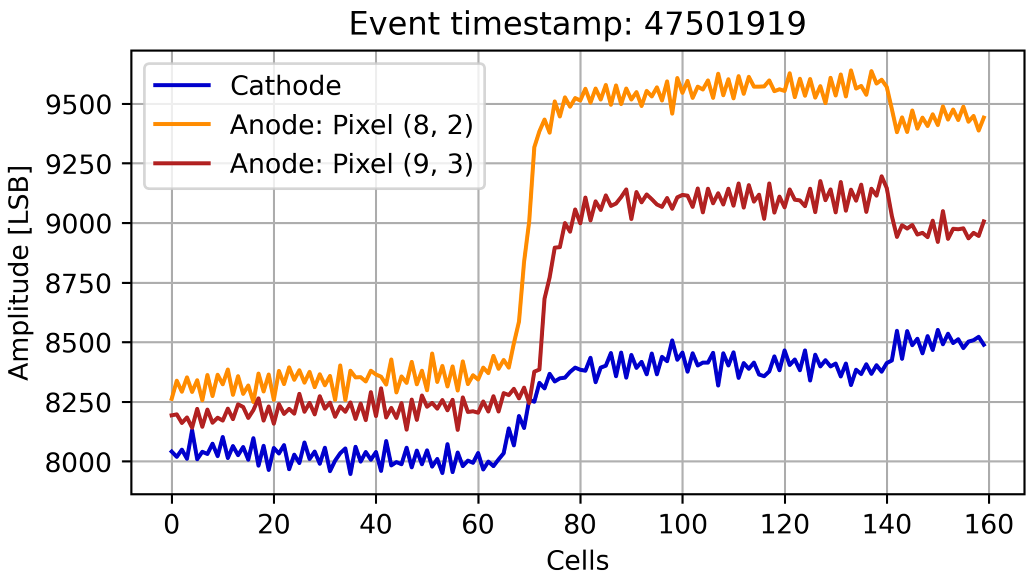


Figure 31: *Multi-event without corrections. These events involve more pixel activations resulting from various types of interactions like Compton scattering.*

Figures 30 and 31 reveal that, particularly in regions before and after the rising edge, the signal fluctuates slightly around the true value and displays a significant offset of approximately 8200 LSB. An offset obscures the true reference point (typically set to 0 LSB) of the detector's amplitude values, thereby potentially masking low-amplitude signals. These factors can lead to inaccuracies in the analysis and evaluation of the data. To address these problems, the manufacturer provided a manual detailing a cell-wise value adjustment and a baseline correction tailored to the specific recording campaign – collectively referred to as "pedestal correction". The resulting effects of all of these corrections applied to the waveforms in Figures 30 and 31 are illustrated in Figures 32 and 33.

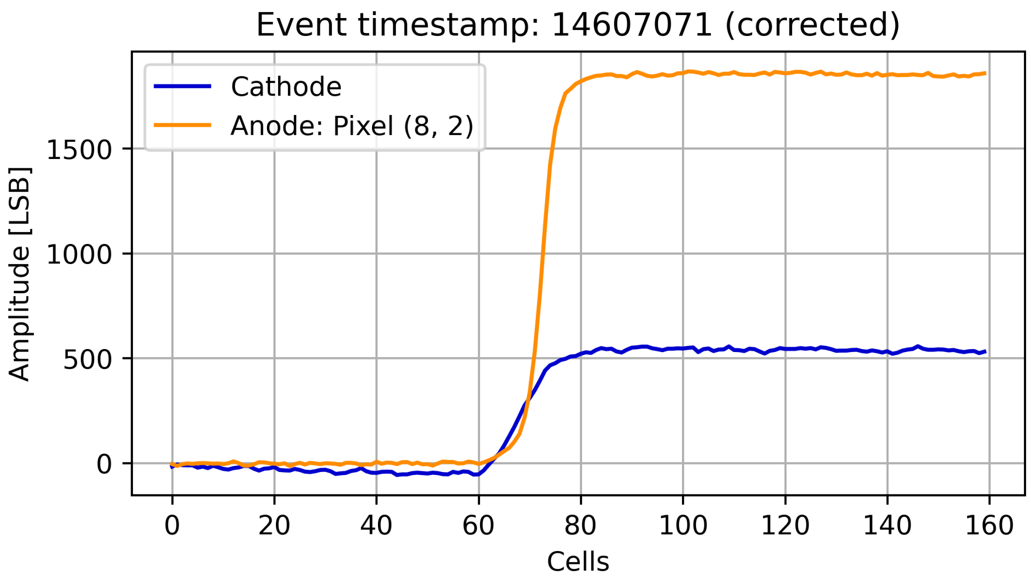


Figure 32: *Single-event pedestal-corrected. Plot of the 160 cell values. Corrections involve baseline correction and smoothening the waveform without veiling its initial characteristics.*

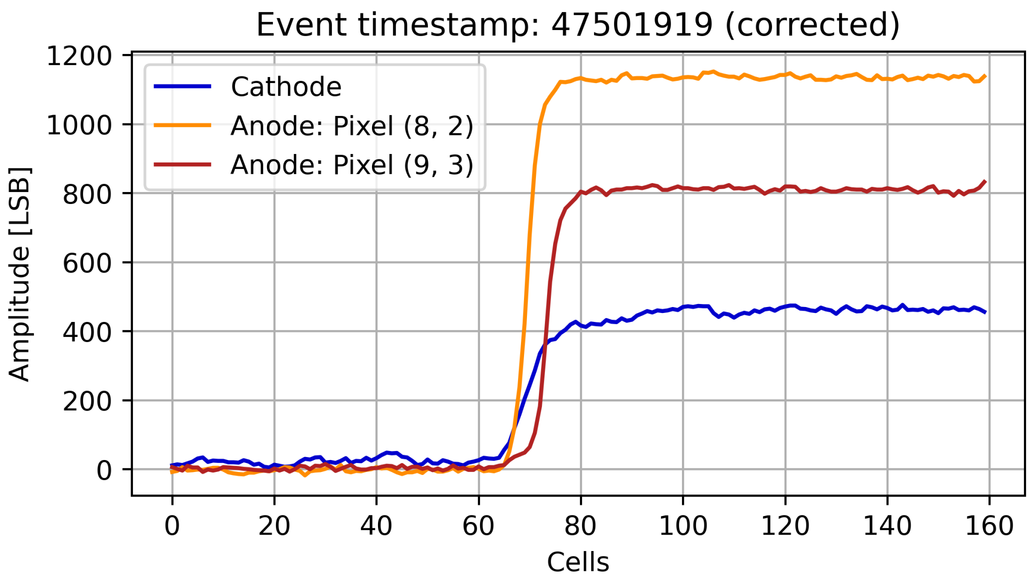


Figure 33: *Multi-event pedestal-corrected. Corrections smooth the waveform enabling finer property derivation.*

The data set which is used in this thesis as the AI training data comprises one million waveforms, which results in 460,392 events (i.e., different timestamps). The single-events comprise 75% of the total event count. Figure 34 shows the result of counting the anode signal from single-events for each individual pixel in the 11×11 array.

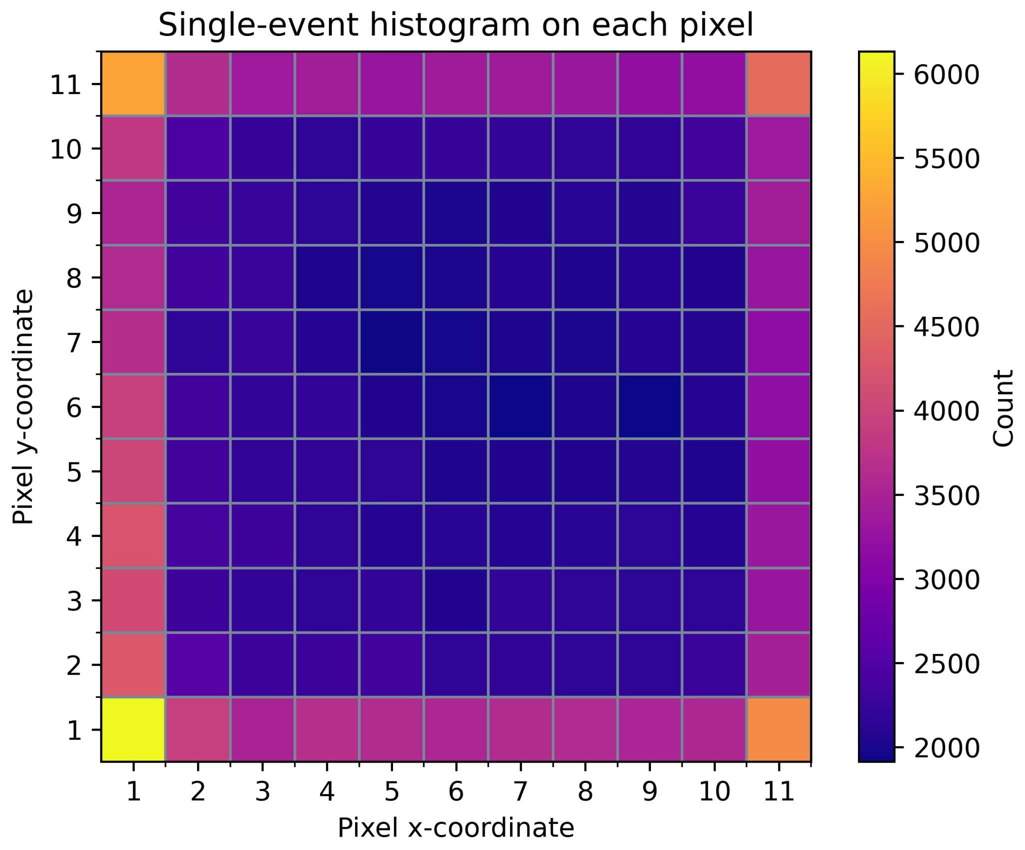


Figure 34: *The number of single-events detected at each pixel has been counted and visualized using a color bar to indicate their frequency. Distinct edge/corner effects are observable in the distribution.*

Figure 34 reveals a higher event count along the edges and, most prominently, at the corners of the pixel surface. This phenomenon can arise from several factors. One plausible explanation is the scattering of escaped radiation, which is reflected back onto the sides of the detector surface by the aluminum shielding. Additionally, experimental studies have shown that peculiar effects emerge near the detector edges. These include alterations in the weighting potential (introduced in Chapter 2) and deviations of the electric field lines near the edges, which can influence the dielectric constant. Such effects may disrupt an otherwise uniform illumination of the detector surface as it can be seen in Figure 34. For a detailed investigation into these phenomena, the reader is referred to [17].

In the following analysis, the energy content per triggered pixel was required. This was determined using pedestal-corrected data by calculating the difference between the mean value of the last 20 cells and the mean value of the first 20 cells, as seen in Figure 35. This procedure can also be found in the source code of the manufacturer's pedestal correction.

In this thesis, the analysis, training and testing were conducted with pedestal corrected observations.

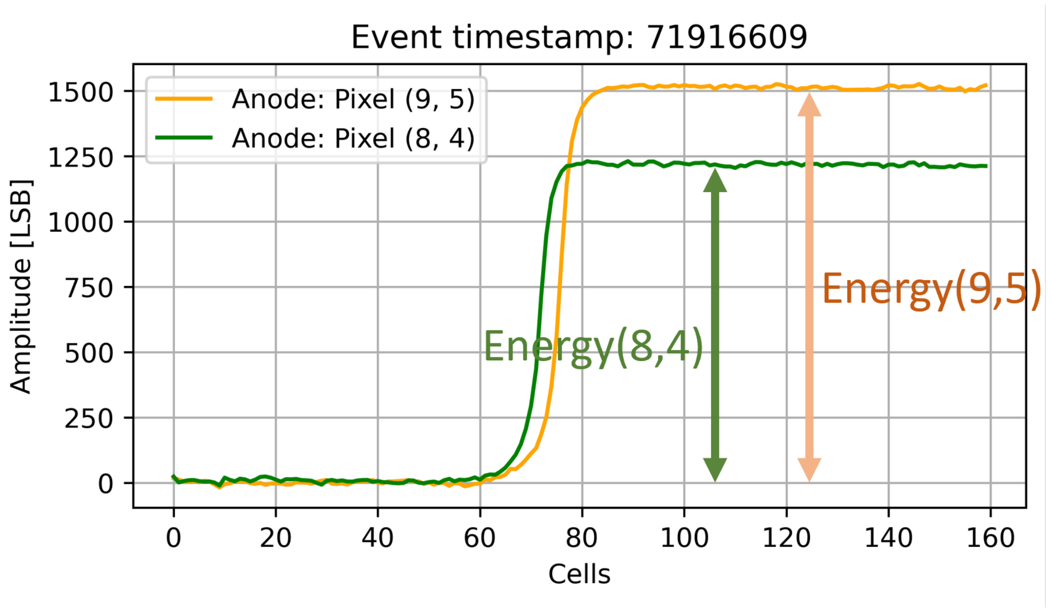


Figure 35: *Illustration of the principle used to calculate the energy deposited in the respective pixels based on visual examination of waveform amplitudes. For baseline-corrected signals, the peak amplitude of the waveform corresponds to the energy value.*

The energy prediction forms a key part of the thesis and is discussed in greater depth in Chapter 4.4.

An additional test was conducted to investigate whether the detector exhibits a directional preference for the movement of electrons generated from electron-hole pairs (Figure 36). This was examined by counting occurrences of all other triggered pixels relative to the pixel indicating the highest energy value. Such a phenomenon arises when incident photons deposit a sufficient amount of energy to the crystal, causing the released electrons to trigger additional electron-hole pairs along their paths. This cascading effect is better known as impact ionization.

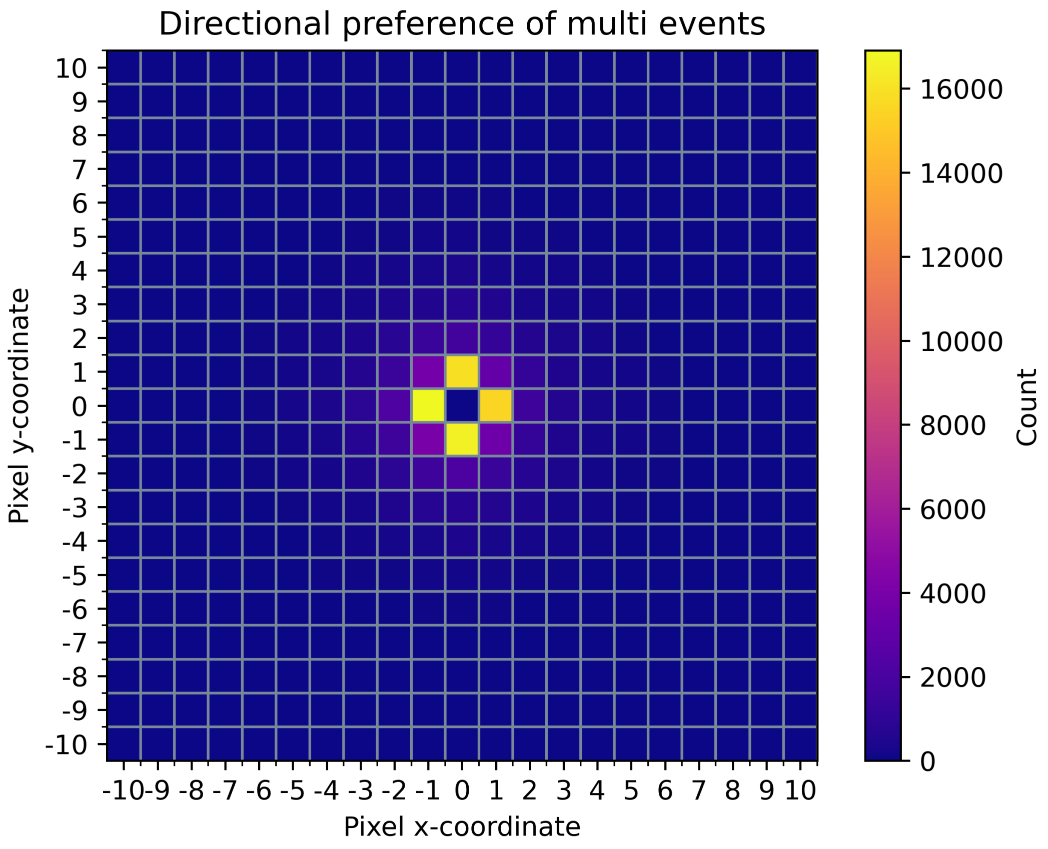


Figure 36: *For all multi-events in the data set the coordinates of all the triggered pixels were counted relative to the pixel exhibiting the highest total energy in an event, which is plotted always in* $(0, 0)$.

In Figure 36, the signal with the highest energy from a multi-event was centered at position $(0, 0)$, with the remaining signals plotted relative to this maximum. The resulting figure indicates that no directional preference is observed in the distribution of additional pixel activations.

A histogram of the recorded event energies over a sufficiently long time interval reveals a characteristic pattern of the source. The histogram displays distinctive peaks, which can be attributed to decay processes the source undergoes as it transitions to a more stable energy state. All of these steps describe essentially the procedure of spectroscopy. Figure 37 shows the histogram derived from all the available energies from single-events, confirming that the data indeed originates from a Cs-137 source, as noted in the introduction to this chapter.

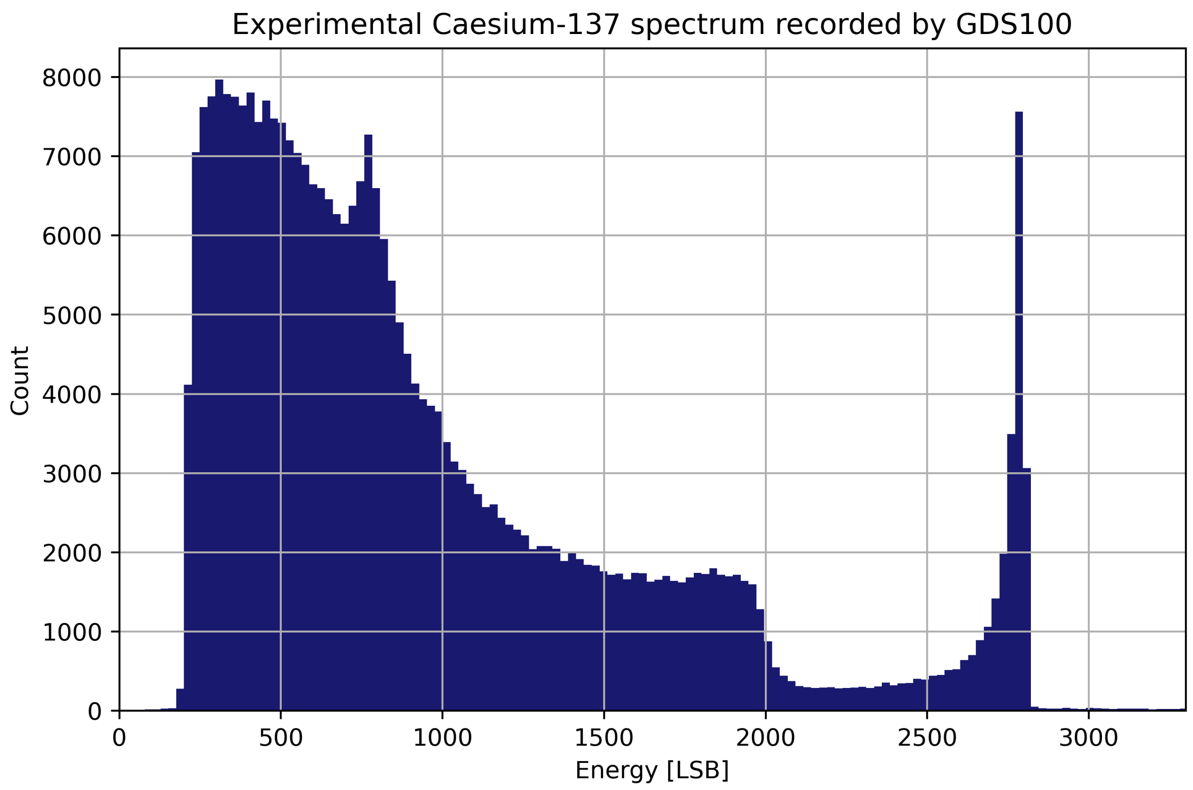


Figure 37: *Histogram of all single-event energies from the recordings. Counting the appearance of all available energies leads to a characteristic shape, which can be associated with the decay pattern of a specific element, here, Cs-137.*

A prominent feature in the spectrum is the peak near 2750 LSB (corresponds to a energy of 662 keV), which arises from the de-excitation of the meta stable state of Barium-137 – a product from the Cs-137 decay – via gamma photon emission. This distinct peak serves as a calibration marker for detectors operating in the gamma-ray band [50].

In addition, the cutoff at approximately 2000 LSB (480 keV) corresponds to the Compton edge. Gamma photons experiencing Compton scattering within the crystal may escape, transferring only a fraction of their initial energy to the recoiled electron. This scattering process produces a continuum of energies (hence the name Compton continuum), depending on the scattering angle, which spans roughly from 800 LSB (184 keV) to 2000 LSB (480 keV), as illustrated in Figure 37. The sharp edge at 2000 LSB (480 keV) is associated with an angle of 180°, representing the maximum recoil energy imparted to the scattered electron.

Lastly, the peak at around 800 LSB (184 keV), known as the backscatter peak, originates from photons which undergo Compton scattering outside the detector, such as the shielding walls surrounding the source, and re-enter the detector setup with reduced energy due to the scattering [7][73].

The analysis of the tests on the received data set clearly indicate that this data provides a solid and reliable foundation for probing whether ML/DL can replace traditional detector data analysis. This investigation aims specifically at certain event properties, the details of which are discussed in the following chapters.

4.2 Task formulation and principle justification

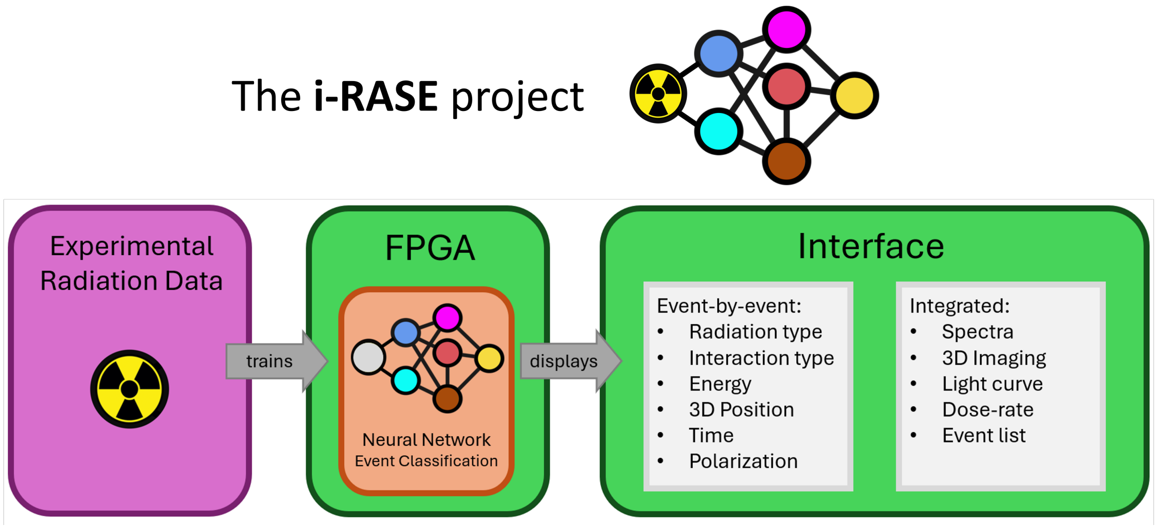


Figure 38: *The structure of the i-RASE project seen from this master's thesis. The neural networks trained from real radiation data will be assembled on an FPGA/micro-controller unit for providing event information from detected high energy events.*

Figure 38 outlines a schematic representation and an excerpt of selected tasks within the i-RASE project. The differently colored boxes indicate the assigned obligations among different project partners. Highlighted in green are the responsibilities designated to the University of Tübingen. The tasks include developing a Field Programmable Array (FPGA) capable of hosting an AI model that can provide real-time (almost instantaneous) predictions from incoming detector data, which include the properties listed in Figure 38 as interface output.

The listed properties are linked to characteristics of high energy interaction emitted by radiation sources. These particles serve as messengers, carrying unique features that the detector captures and interprets, ultimately aiming to provide an accurate description of the source.

FPGAs are a specialized type of integrated circuit that can be custom-programmed for executing specific hardware tasks. A unique advantage of FPGAs is their ability to perform tasks in parallel, which makes them particularly advantageous in applications where high-speed processing and low latency are critical.

To evaluate whether the FPGA meets its intended purpose, fully trained AI models must first be developed – this thesis focuses on that crucial step.

Ultimately, the FPGA and its subsequent interface should provide, for each detected event, the properties outlined in the right-hand box in Figure 38. After reviewing the available data, it was decided that the primary areas of investigation for this thesis would focus on predicting the interaction type, radiation type, energy and the 3D Position of the charged particle's impact of the event using Machine and Deep Learning models.

Initially, the data set was unlabeled, necessitating the development of conventional algorithms as a reference. This allows precisely the comparison that was initially intended: an assessment of AI models versus traditional approaches for calculating radiation-borne features originating from recordings from a high energy detector.

For each of these four categories, eight types of AI models were selected: four from Machine Learning and another four from Deep Learning.

The ML algorithms implemented consist of the Histogram-based Gradient Boosting Tree (Histo Boost), Extreme Gradient Boosting (XG Boost), Random Forest and Decision Tree. The DL models include both compact and a more elaborated CNN architectures, as well as a compact RNN and a LSTM network. In this thesis, "compact" models refer to networks with fewer than 100,000 tuneable parameters. A comprehensive description of the models is provided in Chapter 4.3.

Research has demonstrated that Random Forests and especially gradient boosting models are highly effective for tabular data [69].

In this project, however, the data used for model training presents a distinctive structure: Although the format is tabular, the 160 cells represent the evolution of the anode/cathode pulse shape in time, therefore, having a hybrid-like data structure – tabular (x, y) and sequential (160 cells) structures in one observation. This characteristic clearly justifies the use of Deep Learning to capture the diversified nature within the data, complementing, perhaps surpassing the Machine Learning approach.

Previous research has shown that CNNs achieve outstanding results across a variety of tasks, e.g., image classification/segmentation, natural language processing and even autonomous systems [2]. This success is likely due to their unique ability to extract features efficiently through the mathematical foundation of convolution, thus having relatively fast training times and a smaller storage size in comparison to other model types reaching similar testing accuracy. Their effectiveness will be examined in this thesis once again.

The rationale for incorporating these compact architectures is to assess whether a smaller network could effectively capture the data's underlying structure, potentially yielding robust predictions. Additionally, the compact models might be better suited for integrating the algorithm on the FPGA, where the overall storage size of the model is crucial. The same reasoning applies for the ML algorithms for including the Decision Tree – as solely one classifier of the ensemble method Random forest.

The decision to include RNNs – also LSTMs – in this project is grounded in their inherent ability to map sequential data to arbitrary outputs, a capability enabled by their structural design (cf. Chapter 3.3.4). Sequential data encompasses interrelated patterns that can be found in semantic and syntax of human language, each single frame of a movie scene or – here more relevant – variables evolving through time. The 160 cell values represent the temporal evolution of pulse amplitudes associated with each respective pixel in the detector, fitting naturally into the sequential modeling strengths of RNNs and LSTMs.

The foundational prerequisites have been established: the data is thoroughly understood, the objectives for training and prediction are clearly defined, and the models themselves have been selected and justified.

The following chapters of the thesis will explore finer details on how the models were designed and structured, the aforementioned properties of each event in detail and unveils on how the training data was composed for the AI models. Furthermore, the performance of each model will be presented.

4.3 Models' architecture

The architectures chosen for each model were generally consistent across all tasks. Chapter 4.2 explored the reasoning behind employing both compact and more advanced architectures. This dual approach also stemmed from an understanding that certain tasks involve relatively straightforward objectives, thus, a compact architecture may be efficient in capturing the underlying complexity of the tasks.

To ensure model compatibility, 2D convolutional layers were used for matrix-formatted training data in CNNs, while 1D convolutional layers were applied for vector data (e.g., energy, see Chapter 4.4). This configuration is essential for functional training execution, as mismatches between data dimensions and convolutional layer types would otherwise disrupt the operation.

Overall, all selected model architectures performed well. In particular, for DL, it was frequently evident that the underlying structure of each task could be effectively learned and retained, which minimized the need for architecture-specific adjustments across the objectives. Although alternative structures were initially tested, those that performed poorly were omitted.

After completing the training and test phase of the ML and DL algorithms and the best-performing models demonstrated a strong grasp of the underlying patterns, they need to undergo a lengthy process of fine-tuning and optimization. However, this step is only taken after all project partners reach a consensus on the definitive detector design, including model-task combinations, or the potential development of a task-unifying approach – a model which covers all the tasks at once. Such a unifying model would need to comprehend various aspects of each task, likely requiring a more complex architecture. Whether this complexity can be accommodated within an FPGA unit remains to be seen. As the project is still in its early stages, covering the fine-tuning part is beyond the scope of this thesis.

The ML algorithms used in this work were selected from the well-regarded scikit-learn suite, with all models chosen from tree-based approaches. As a baseline representative, the Decision Tree was included, alongside the more advanced ensemble method, Random Forest, configured with 100 estimators, i.e., 100 individual decision trees.

Additionally, two gradient boosting algorithms were employed: the Histogram-Based Gradient Boosting Tree (Histo Boost) and Extreme Gradient Boosting (XG Boost). These algorithms were left with default settings, as these configurations appeared to be well-suited to the task requirements.

For all regression tasks, the models were trained using the loss criterion of the Mean Squared Error (MSE, Equation (7)). For the classification task, the Decision Tree and Random Forest models used the Gini impurity criterion, while the Histo Boost and XG Boost models applied the softmax loss (cf. Chapter 3.3.1).

The Deep Learning (DL) algorithms used in this work were implemented using the well-established TensorFlow library. For all models addressing regression tasks, also the Mean Squared Error (MSE) was employed as the loss function, with Mean Absolute Error (MAE) recorded alongside as an additional descriptive metric. For the classification task, all models utilized the softmax loss. The data set was split into training and validation sets in a 4:1 ratio, with a batch size of 64 – a common choice in DL. The Adam optimizer, using its default learning rate of 0.001, was selected for gradient descent optimization.

Additional techniques, known as callbacks, were incorporated to further enhance model training and are available for further reference in [70]:

First, model weights were saved only when the validation set's MSE showed improvement. Monitoring validation loss serves as a safeguard against overfitting – the issue that the model parameters were exclusively fit to the training data, leading to the loss of generalization –, providing a realistic measure of model performance on unseen data. This validation loss was also monitored by other callbacks, described below.

One critical callback reduces the learning rate if no improvement in validation loss is observed for a specified period, which is simply termed as "Reduce Learning Rate on Plateau". This technique can help the model to reach deeper into the minimum of the loss landscape, as illustrated in Figure 39. In this process, the DL model explores the parameter space to locate the global minimum, representing the optimal parameter set (as discussed in Chapter 3.3.2). Gradient descent adjusts parameters by taking steps toward the steepest gradient, with the step size governed by the learning rate. In cases where the learning rate is too high, the model can overshoot the minimum, oscillating around it without converging. Reducing the learning rate shrinks the step size, allowing the model to reach deeper into the optimal region [87]. In this work, the learning rate was reduced by a factor of 0.1 after ten consecutive epochs of non-improvement in validation loss, with a minimum threshold set at 10^{-6}, beyond which further reductions slow the training progress significantly.

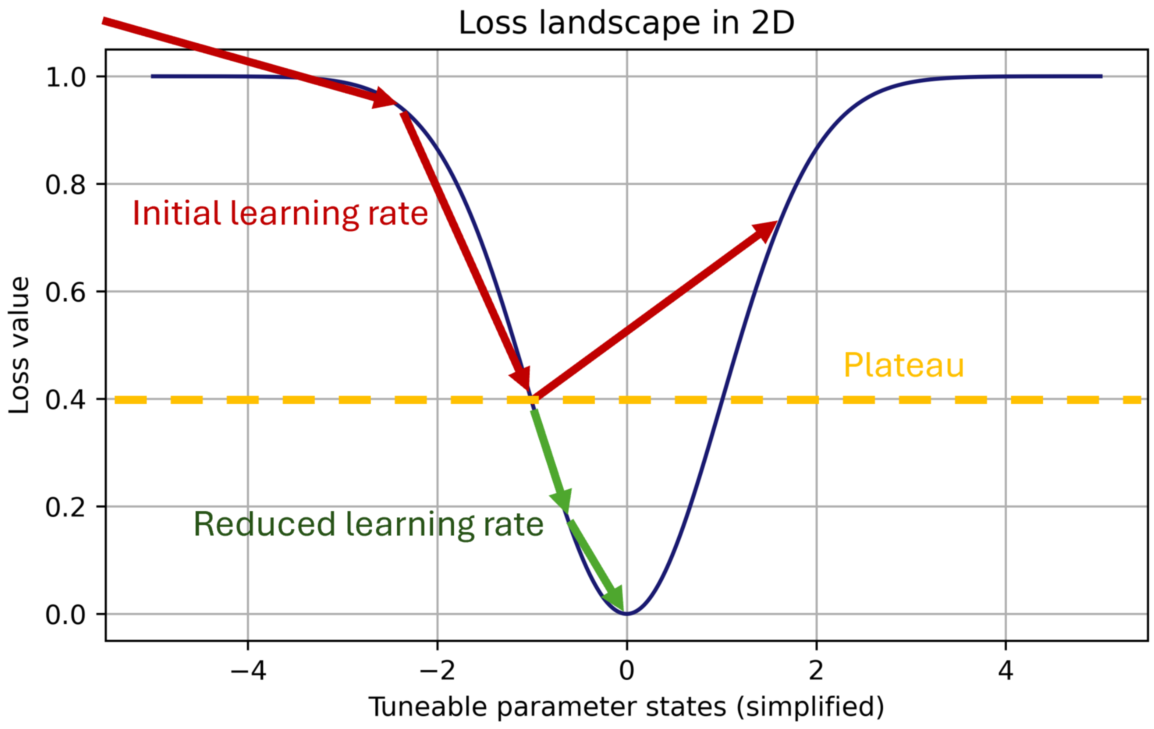


Figure 39: *Simplified visualization of reducing the learning rate on a plateau scenario. This plateau acts as a barrier that the learning rate (step size) cannot surpass, as its initially high value can lead to overshooting the minimum. Decreasing the learning rate during the learning process might lead to a more effective gradient descent.*

Another callback, Early Stopping, was used to increase training efficiency. While each model was set to train for up to 500 epochs, Early Stopping halts training when no further improvement in validation loss is detected. This prevents unnecessary computation time when the model has reached its suppositional optimal state. In this study, training ceased after 14 epochs without improvement.

Together, these augmentations resulted in improved performance and a more efficient use of training time.

The layers of the following models are visualized in the figures. The input layer of each model is tailored to the respective X described in each task section, while the output layer aligns with the target variable y. Where activation functions were applied, the ReLU function was used.

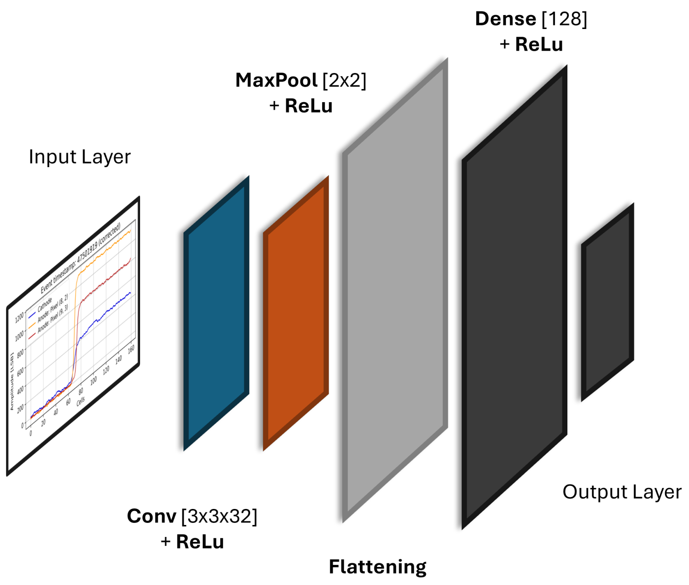


Figure 40: *Compact CNN architecture. Layers are symbolized through planes and the type is indicated through color. This architecture was chosen to try if this simple architecture can grasp less complex patterns in training.*

The architecture shown in Figure 40 is based on a basic CNN model from TensorFlow tutorials. This model demonstrates how even a simple constructed CNN can outperform other types of models in certain tasks [71][72].

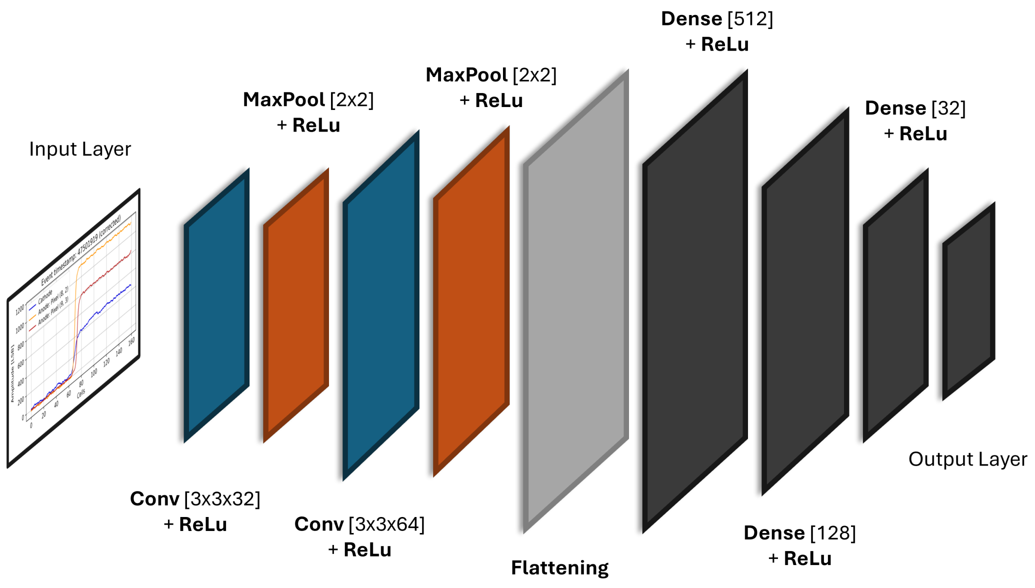


Figure 41: *CNN architecture. More complex structure for more intricate tasks. The first convolutional layers enable the feature extraction and the dense layers subsequently connecting the distilled features with each other.*

Figure 41 presents a common CNN architecture, used in many benchmark and actual applications [13][15][56][61][82][85]. This model yielded strong results in this work.

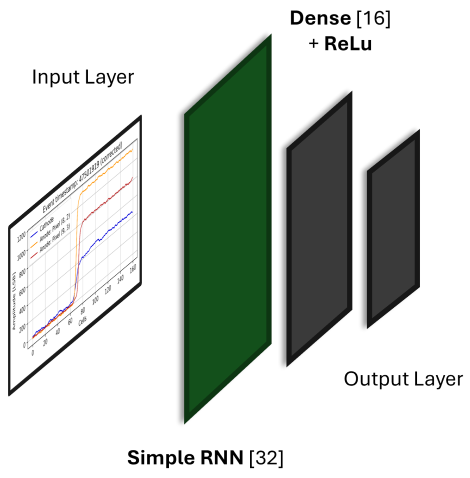


Figure 42: *Compact RNN structure. Only one RNN layer was chosen, due to high execution time in learning and to have a simple model in the memory-type models. The feature extraction is done by the RNN layer.*

The Recurrent Neural Network (RNN) shown here, in Figure 42, represents a compact, simplified application of this architecture, which serves as a lighter alternative to the more complex LSTM network depicted in Figure 43.

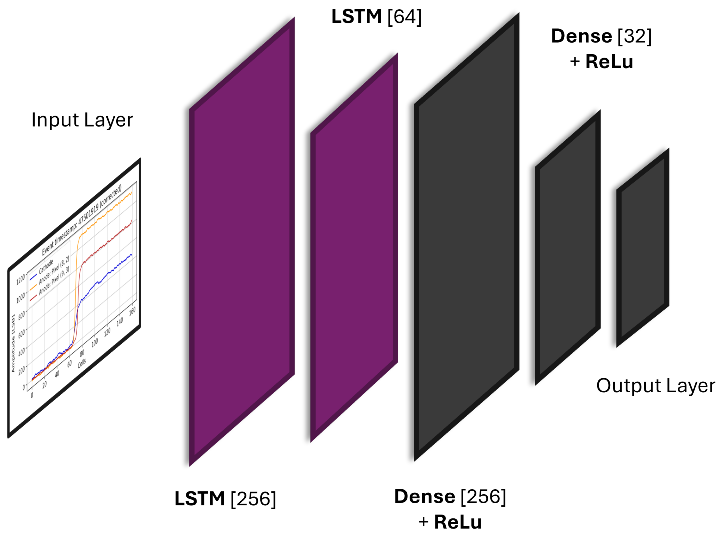


Figure 43: *LSTM network architecture. A architecture with more trainable units relative to the RNN. The advantages of the memory approach are harnessed here.*

Long Short-Term Memories (LSTMs), as a more advanced variant of RNNs (cf. Chapter 3.3.4), were also employed in a relatively large structure to heighten the effectiveness of this network type. Memory-based networks, such as LSTMs, were the most recent addition to this study, introduced following a test on the pixel pattern (cf. Chapter 4.6), where they demonstrated a notably high performance, even surpassing CNNs on this specific task. This motivated further testing of the memory networks on the other tasks.

Each of these architectures were tested and applied to the data recordings designated to predict the properties explained in the subsequent chapters.

4.4 Energy

A fundamental approach for calculating the total energy of a given observation was previously outlined in the explanation of the detector data analysis (cf. Chapter 4.1) and is illustrated in Figure 35.

The training data, X, consists solely of the 160 cell values derived from the anode data, where the pulse amplitude generated by the electrons is proportional to the incident energy (cf. Chapter 2). The corresponding energy value, measured in LSB units, serves as the label, y, leading ultimately to a feature vector of length 160 for X and a scalar quantity for y. Structuring X and y in this manner is a logical approach, closely aligned with the conventional functions used to calculate the total energy of the event.

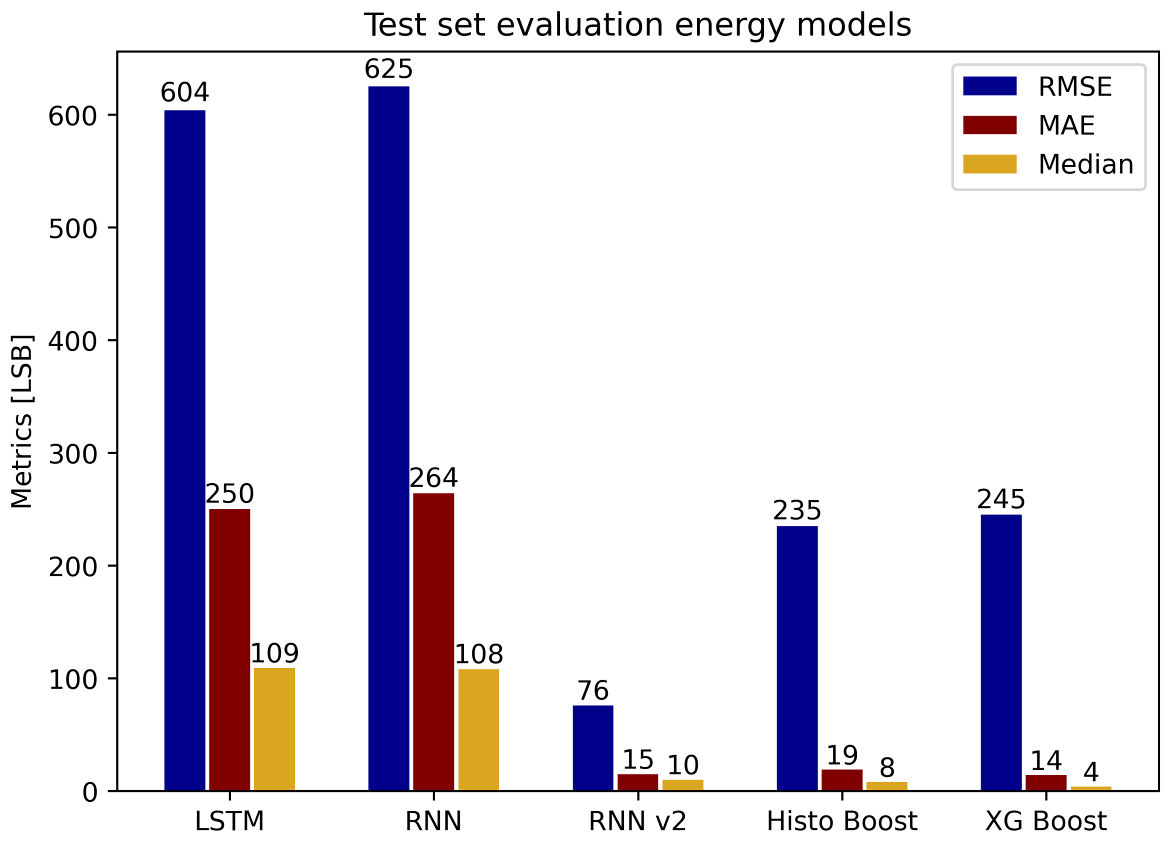


Figure 44: *The first half of the energy test metrics evaluates performance using three measures: Root Mean Squared Error (RMSE), Mean Squared Error (MSE), and median error, all expressed in the energy unit LSB. The results indicate that both the boosting algorithms and the differently compiled RNN (RNN v2) models were effectively trained.*

As the energy prediction represents a regression task, the primary metrics include the Root Mean Squared Error (RMSE), Mean Absolute Error (MAE) and median error.

As shown in Figures 44 and 45, nearly all models performed well, as the task was relatively straightforward. Among these, the CNNs outperformed the other models, with the compact CNN achieving slightly superior results. This suggests that simpler architectures can provide robust performance for tasks of lower complexity. One potential explanation for the success of the CNNs is the ability of recognizing edges (i.e., certain shapes and features) which is central to this task and in accordance to the strengths of CNN models.

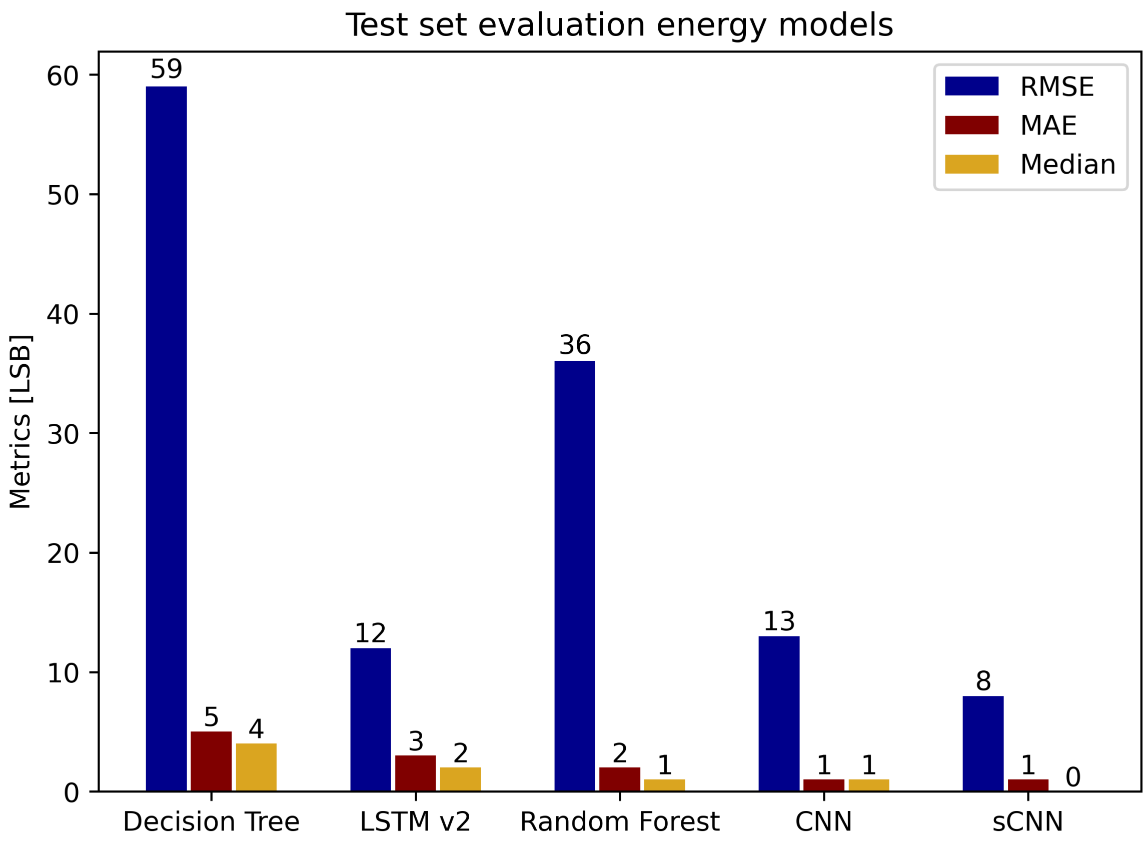


Figure 45: *The second half of the energy test metrics also include the three regression metrics: RMSE, MAE and median error, all reported in the LSB unit. The CNN models demonstrate great performance.*

Two variations were tested for the RNN and LSTM models. In the initial approach, X was structured as a 1×160 input, which did not perform well with memory type models. These models rely on retaining information across time steps, which is absent in a single amplitude input. However, restructuring the input as a 160×1 matrix – as it has been done in version 2 (v2) – significantly improved performance, especially for the LSTM, as the sequence of amplitude values could then be sustained across cells. Nevertheless, training time was prohibitively long for these models, since storing information across 160 time steps for a single event led to considerable slowdowns.

It is also noticeable that RMSE scores were generally higher than MAE, likely due to a moderate number of outliers that posed challenges for accurate predictions, as RMSE amplifies the penalty for larger errors. Despite this, the CNN models demonstrated robustness even in handling these outliers. In comparison, the ML models did not perform as well as DL models, given that the data format here diverges from traditional tabular structures.

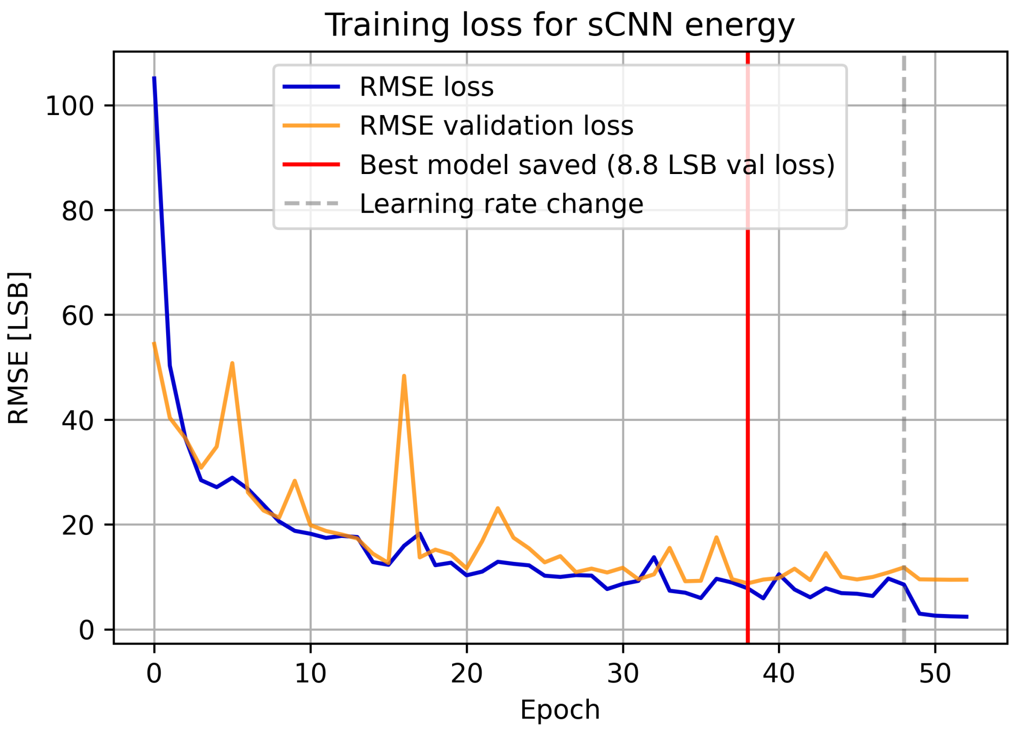


Figure 46: *Training history of the sCNN visualized. The evolution of RMSE loss in LSB is shown over the training epochs for both the training set (highlighted in blue) and validation set (highlighted in orange). The steadily declining trend of the loss indicates effective learning of the underlying data patterns.*

The training process involves optimizing the model parameters to minimize the loss function. As shown in Figure 46 the loss decreases steadily, reflecting the model's ability to grasp the underlying structure of the data and map it accurately to the desired output. Notably, the best model was saved prior to any learning rate adjustments, achieved using the initial (relatively high) learning rate. This suggests that the model quickly learned appropriate parameters, further indicating that the task is relatively easy to solve.

Apart from the RNN and LSTM, all other models exhibit a similar loss progression during training. One remarkable result can be observed in the training history of the RNN v2, as shown in Figure 47.

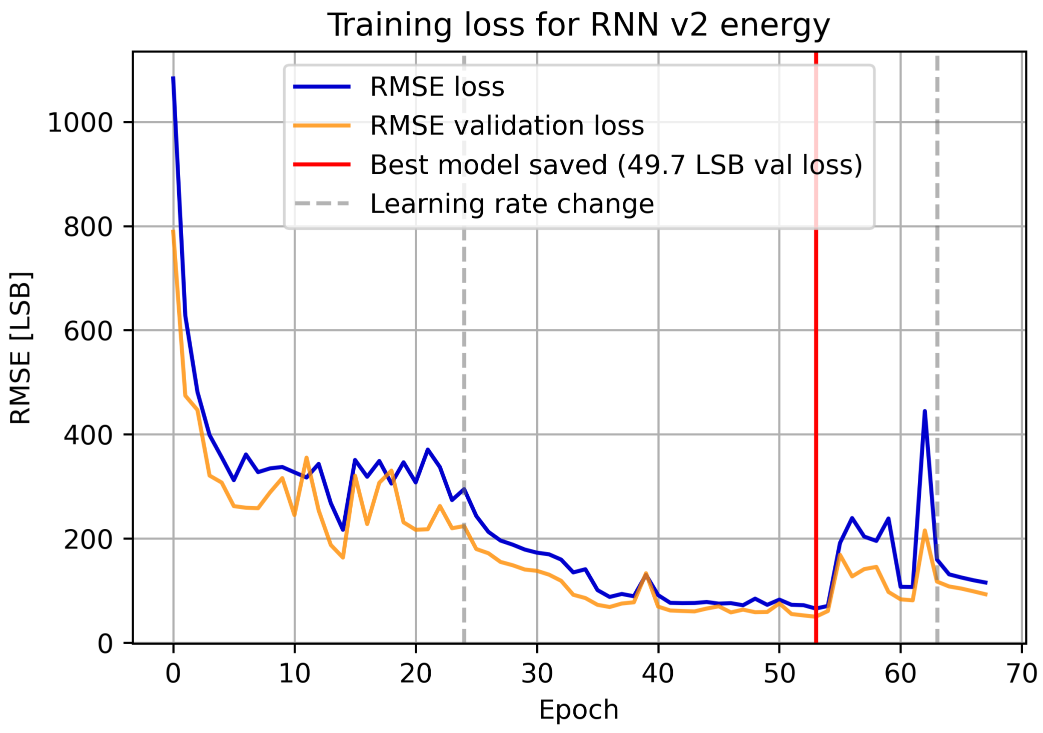


Figure 47: *Training history of RNN v2. The RMSE loss is plotted across training epochs. Following the saving of the best model, the loss increases, suggesting the optimization algorithm overshot the minimum.*

In Figure 47, the loss begins to rise after the model parameters were saved, indicating that the Adam optimizer's step size was too large. This caused the optimization process to overshoot the local minimum, effectively leaving the "minimum well" (as illustrated conceptually in Figure 39). After the learning rate was adjusted, the loss stabilized, but the early stopping mechanism terminated the training soon afterward.

The error distribution between predicted and true labels can be effectively visualized using histograms, as shown in Figure 48.

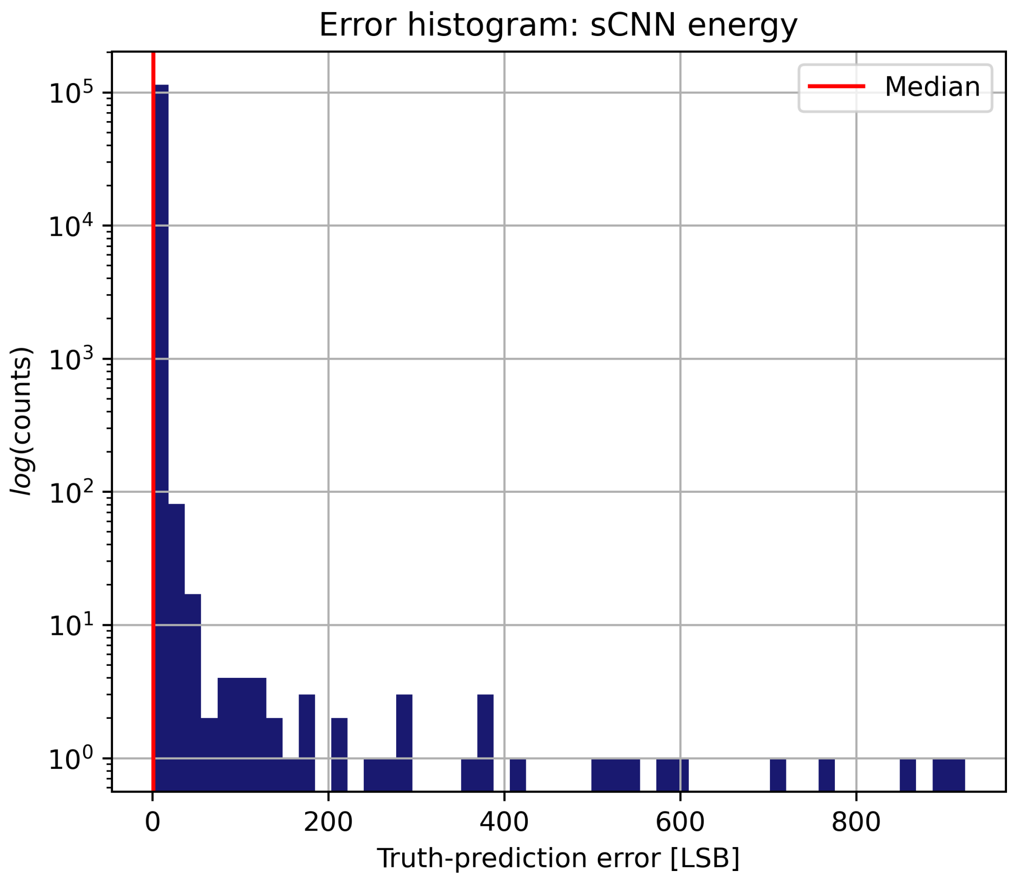


Figure 48: *Error histogram of the sCNN. The differences between true and predicted labels in LSB are counted and displayed on a logarithmic scale. The majority of errors are concentrated close to zero.*

To better illustrate the distribution, the counts are plotted on a logarithmic scale. The majority of errors are close to zero, consistent with the median being located at this position. Additionally, no significant outliers are observed in the error distribution.

Other models exhibit similar results, although their histograms may be slightly shifted toward higher error values.

For assessing the performance across the range of predicted and true values a regression scatter plot is advantageous.

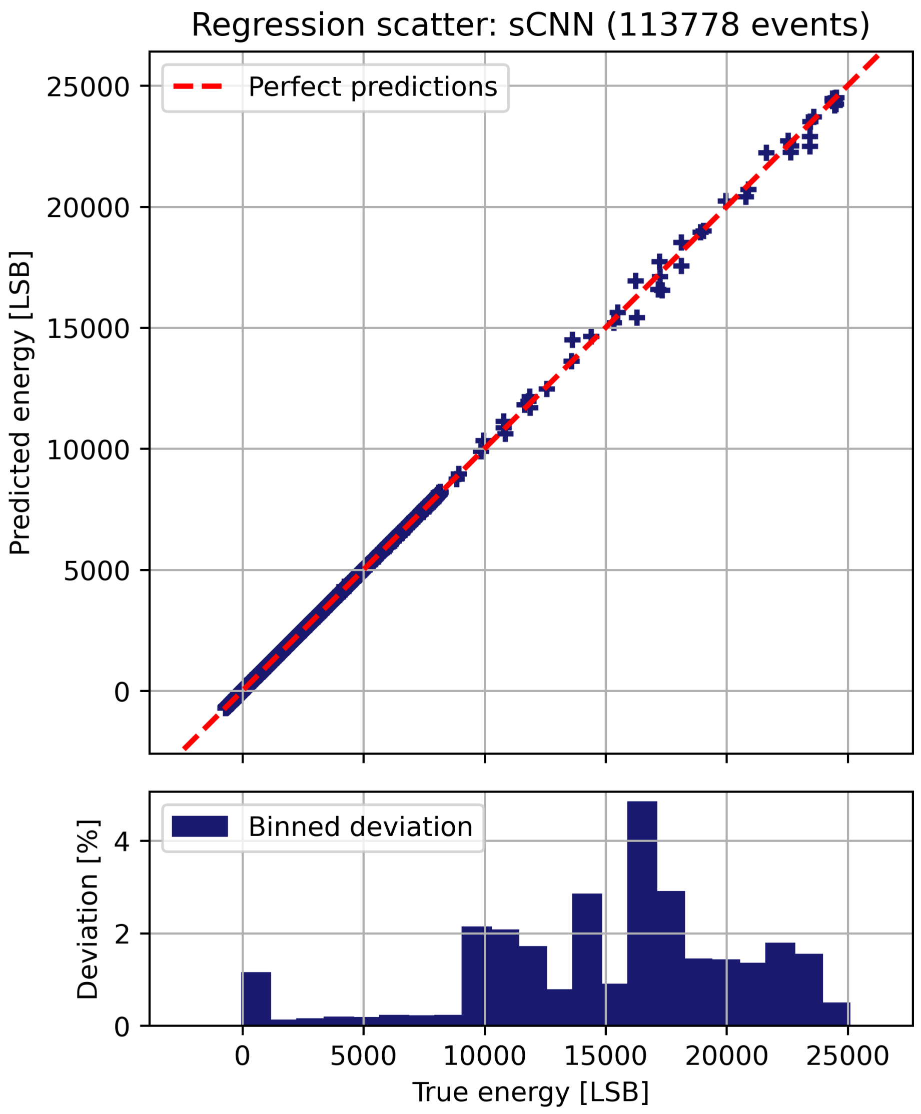


Figure 49: *Regression scatter plot of the sCNN. Predicted labels are plotted against true labels in the energy unit LSB, demonstrating robust performance across the entire value range. The deviation from the true value is overall relatively small, with a slight increase after 10,000 LSB.*

Figure 49 shows the predicted energy being plotted against their corresponding true values. The sCNN demonstrates consistent and reliable performance across the entire scale, indicating robust generalization.

In contrast, an intriguing result emerges in Figure 50 where the XG Boost model completely fails for values above 10,000 LSB, which amount to a comparatively small share of the samples, suggesting that the model suffered an overfit and could not generalize.

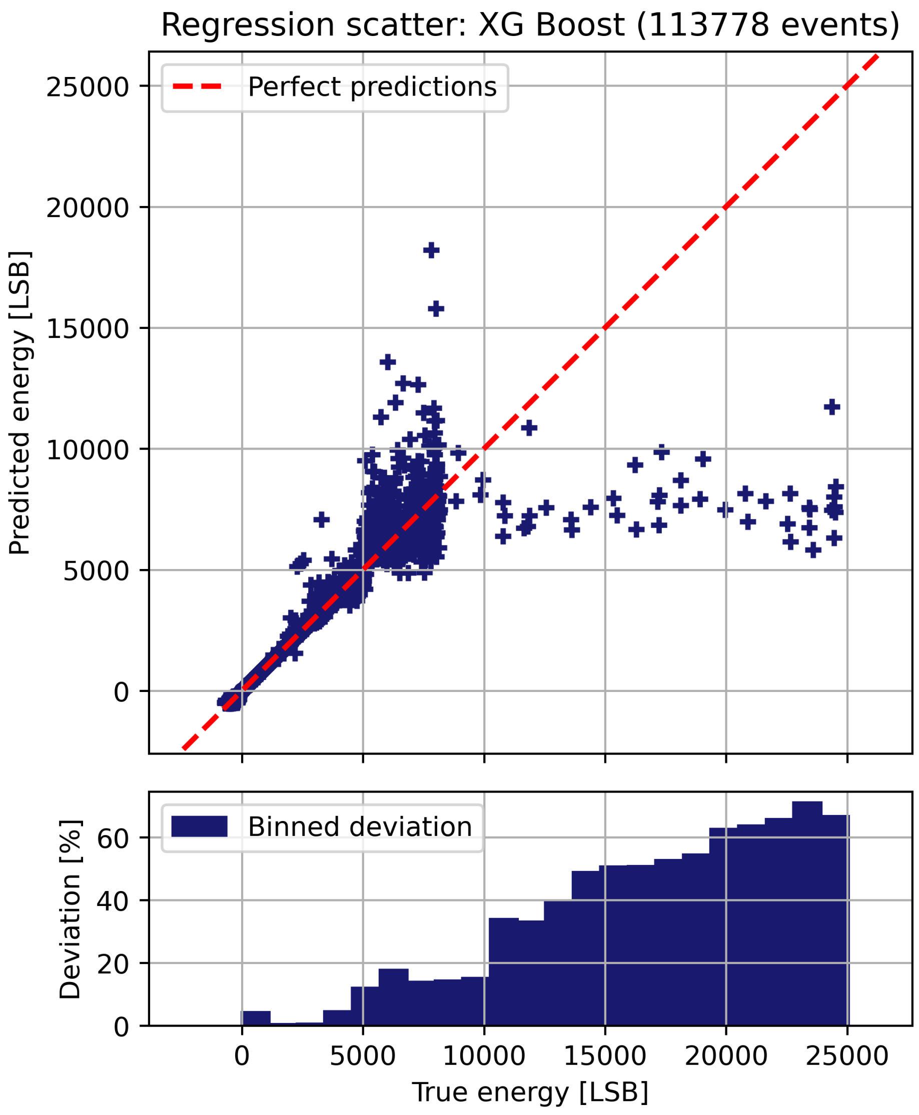


Figure 50: *XG Boost regression scatter plot. True labels are plotted on the x-axis, and predicted labels are on the y-axis. While predictions align closely with the diagonal for lower values, the model fails completely beyond 10,000 LSB, also supported by the deviation plot displaying more than 60% difference to the true value.*

4.5 Drift time

Chapter 4.2 mentioned the task of the AI algorithms for predicting the 3D impact position of a charged particle within the detector – that is, the specific point in which the particle interacts with the built-in crystal. The drift time allows the computation of the depth of interaction within the crystal – z coordinate, which describes the semiconductor's measure of depth –, whereas the x and y coordinates correspond to the position on the detector's pixelated surface.

The CZT detector manual provides guidance on determining the drift time, explaining that it begins when the cathode signal starts to rise and concludes once the anode signal reaches full saturation. This process has a physical basis: when a photon (or any other charged particle) enters the detector and interacts with the crystal and has sufficient high energy, electron-hole pairs are created – the valence electrons break from the atom (more detailed in Chapter 2). Located on opposite sides of the crystal, the cathode detects positively charged holes, causing its signal to rise. Meanwhile, the negatively charged electrons are drawn toward the anode, also inducing an increase in the anode's amplitude. As the last electrons arrive fully at the anode, the amplitude has reached its peak, hence the electrons' drift has been completed. Since the 160 recorded cell values represent the amplitude's evolution in time, they enable the calculation of the drift time.

In Figure 51 a visualization of the drift time with its associated anode and cathode waveform is shown. It follows the description given in the previous paragraph.

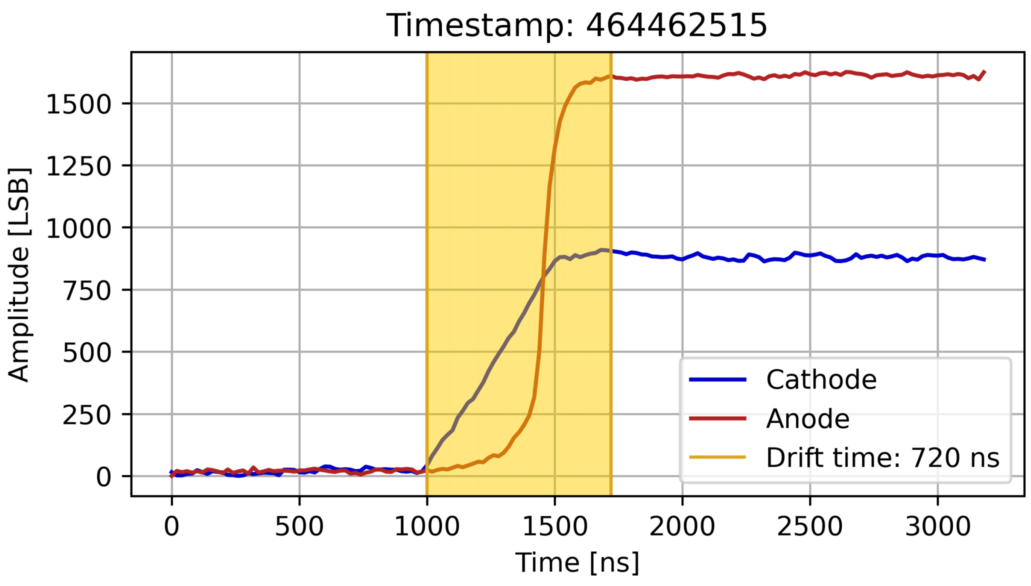


Figure 51: *Visualization of the drift time. The drift time is determined using the waveforms from the anode and cathode. It is calculated by measuring the time interval from the rising edge of the cathode signal to the saturation point of the anode signal, as charted on the time axis.*

The conventional method calculates for the cathode the time at which the signal begins to rise. Specifically, it computes the mean and the standard deviation for the first 30 cell values and marks a cell as anomalous if its value exceeds the mean plus one standard deviation, indicating a potential rise point. This classification is repeated for all 160 cells, which are then analyzed in reverse to locate the first non-anomalous cell after the waveform's peak is surpassed (assuring that the rising point is before the saturation).

For the anode signal, the function works in the opposite manner: the desired point (point of saturation) is where the amplitude starts to decrease as seen backwards in time, marking points as anomalous when their cell value is lower than the mean minus one standard deviation. Similarly, the saturation point is found where the first non-anomalous cell is located, after the global minimum of the signal is passed, seen forward in time.

Figures 52 (cathode) and 53 (anode) illustrate the working principle of the function described above.

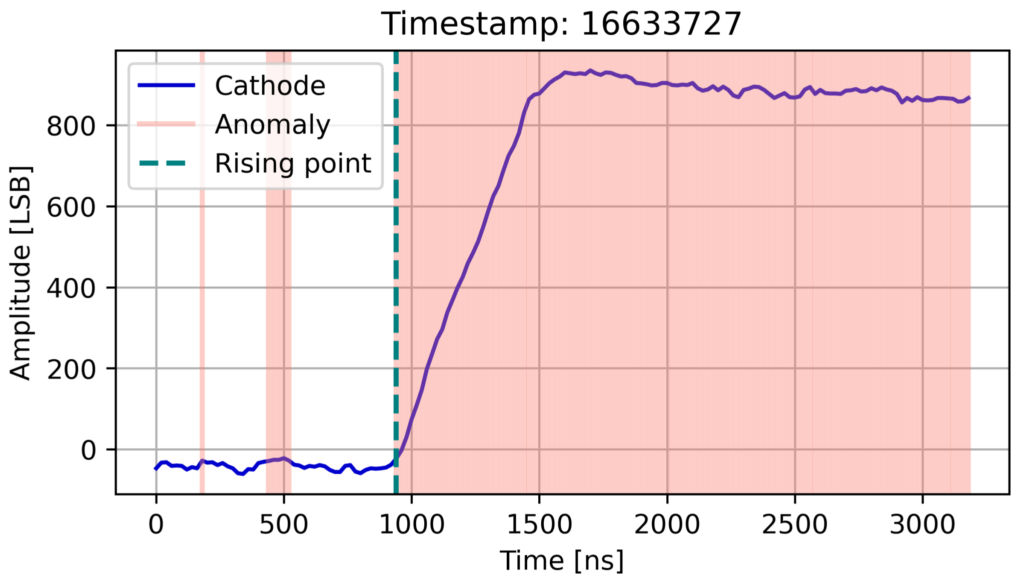


Figure 52: *Detection of anomalies in the cathode signal. A cell is flagged as anomalous if it exceeds the threshold defined by the mean plus one standard deviation of the first 30 cells. A sustained anomaly is a prerequisite for identifying the onset of the rising point.*

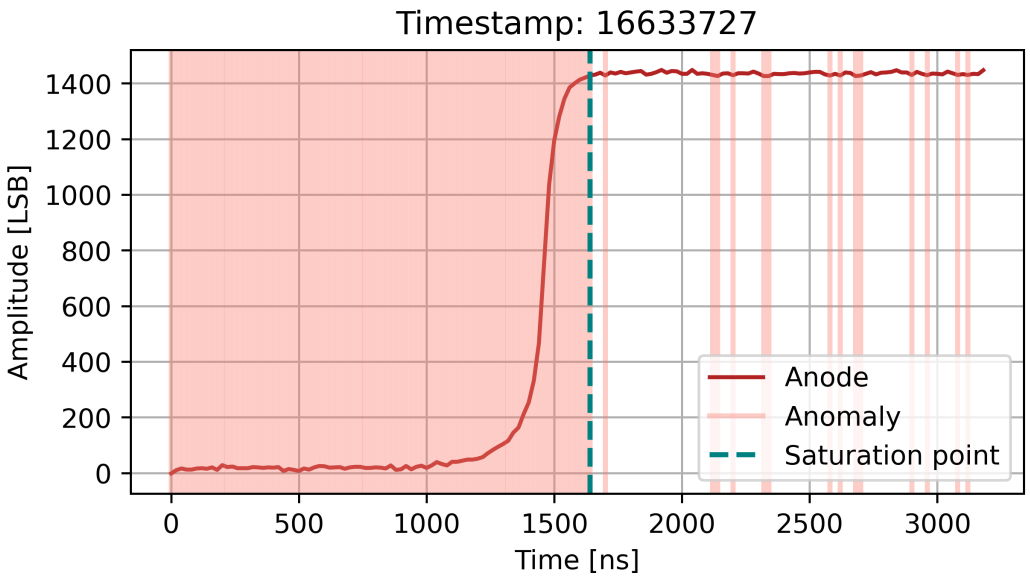


Figure 53: *Anomaly detection for the anode signal. The cell is considered anomalous if its value falls below the mean minus one standard deviation of the last 30 cells. Notably, the anomaly detection process for the anode operates inversely to that of the cathode.*

This approach reliably identifies adequate rise and saturation times for the vast majority of cathode and anode observations, respectively. Given the detector's event readout design, the minimal achievable time resolution is 20 ns (cf. Chapter 4.1). Events that exhibited negative drift times, resulting from atypical detector behavior, were excluded from the training data to maintain a more deterministic data set for the AI training.

The training data, X, is structured as a $2{\times}160$ matrix, containing 160 cells for both the anode and cathode signals, as these are necessary to determine the drift time. In particular, every anode was paired with its associated event-dependent cathode. The output, y, is a scalar quantity representing the drift time in ns.

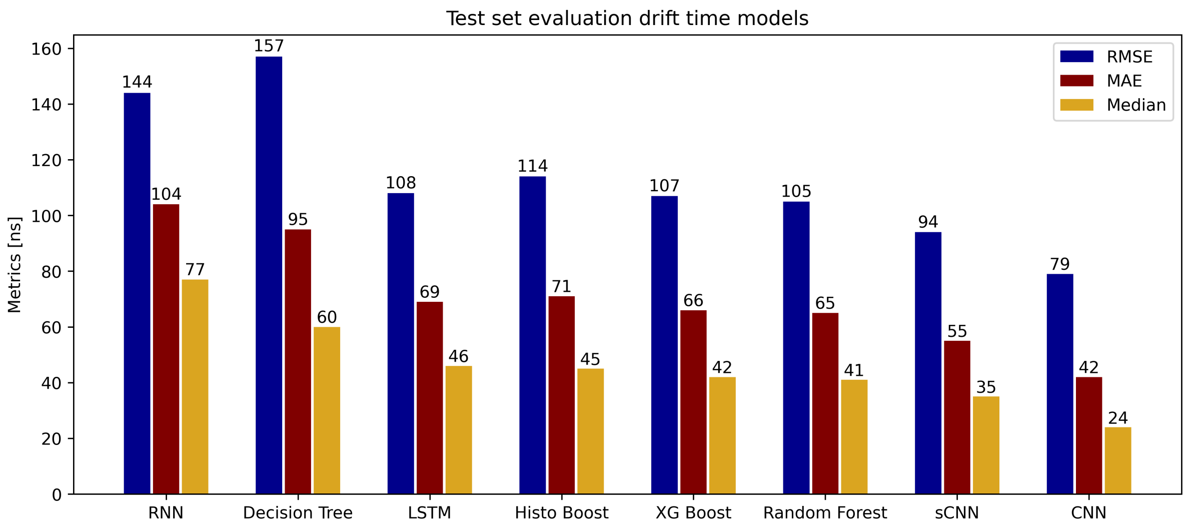


Figure 54: *Comparison of the drift time models' metrics. The RMSE, MAE and the median error in ns are plotted. The CNN models excel in this task.*

The same metrics used in Chapter 4.4 were also applied here as this is also a regression task. The CNN models again demonstrated superior performance, as the 2×160 matrix format resembles an image in which feature (e.g., shapes, edges) detection directly contributes to accurate predictions. However, this task is more complex than the energy prediction task. With the readout system's finest waveform resolution at 20 ns, the CNN achieved a precision in that order, as reflected by the median error.

Interestingly, the ML algorithms also performed well, despite needing to compare two waveforms. A potential explanation is that the binary decision process at each tree node may operate similarly to the rise/saturation detection utilized in the traditional function, where each cell is labeled as either an anomaly or not.

In contrast, the memory-based RNN and LSTM models underperformed compared to their counterparts. It was anticipated that these models might excel by directly transferring information from cathode to anode data, thus enabling a true comparison between the two. However, they did not match the performance of the CNN and ML models.

The training process of the CNN revealed a consistent downward trend in the RMSE loss' values, as illustrated in Figure 55. Notably, the adaptation of the learning rate significantly enhanced the training process. The best-performing model was saved after the learning rate was reduced twice. However, the large gap between the training and validation losses indicates that models saved during epochs where this gap is pronounced might suffer from overfitting. This suggests that, on unseen validation data, the model does not demonstrate substantial improvement. This serves as a compelling example of using validation loss as a critical metric for determining the optimal checkpoint for saving the model.

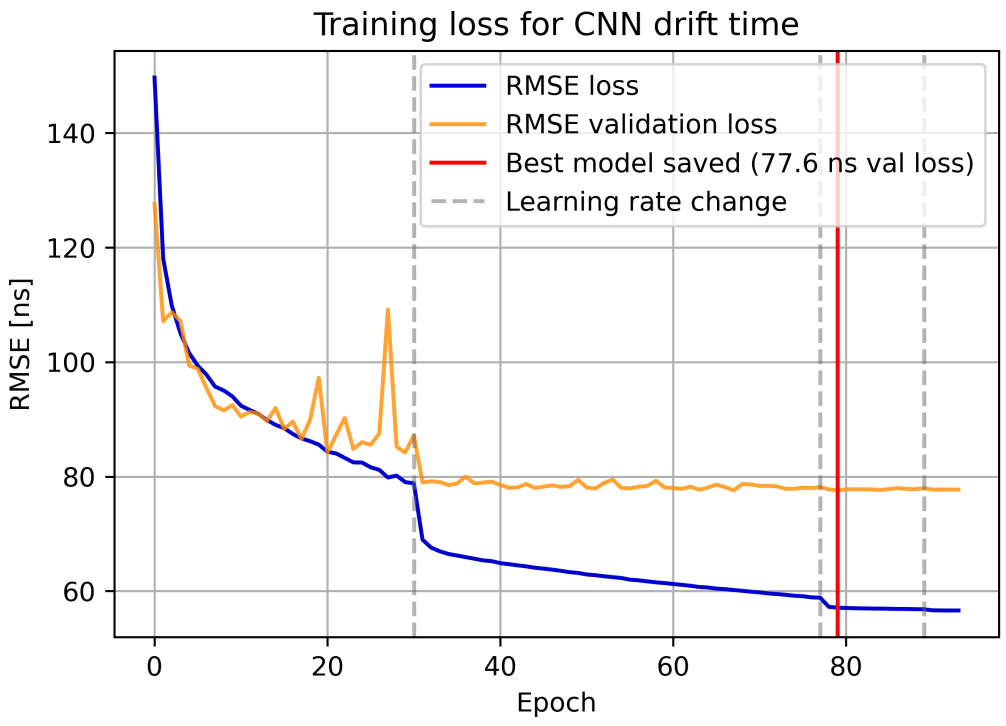


Figure 55: *CNNs training history. The RMSE losses are displayed. The loss functions indicate a steady fall. While the RMSE loss falls with every LR change, the validation loss remains at one point mostly constant, indicating an overfit. Since the saving of the best model weights depend on the validation loss, the danger of overfitting is largely defused.*

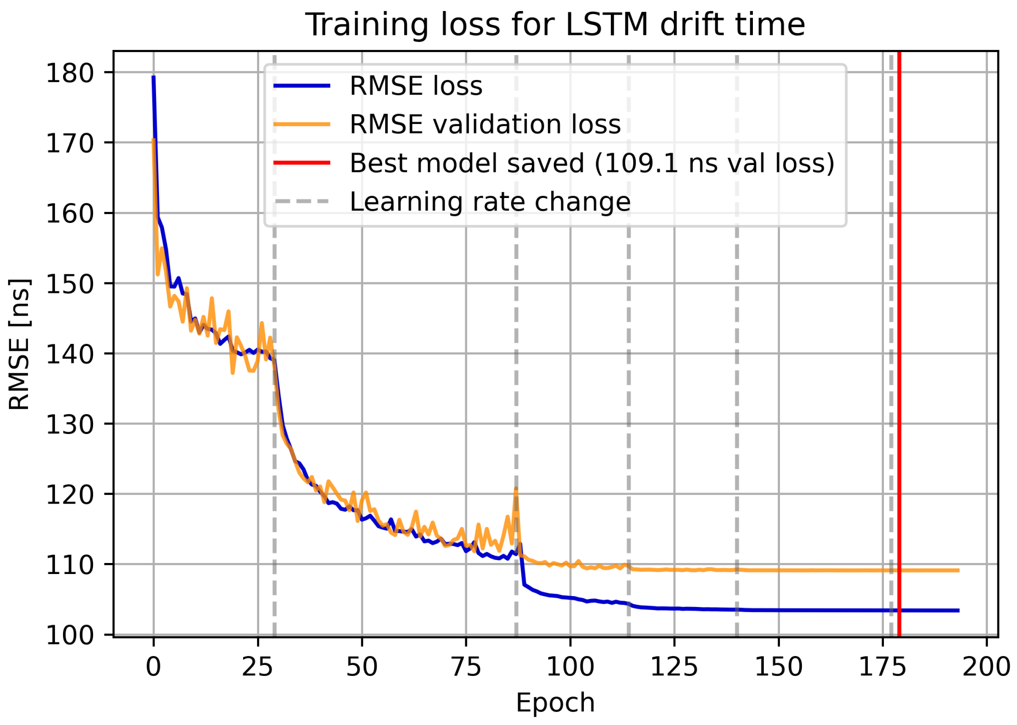


Figure 56: *The training history of the LSTM drift time. Remarkable impact of the LR change is visible here, continually reducing the loss value massively.*

Another strong example of the impact of learning rate changes is illustrated in Figure 56, showcasing the training history of the LSTM for drift time prediction. Each adjustment to the learning rate induces a significant reduction in the loss function, enabling a steeper descent and optimizing the model's overall performance.

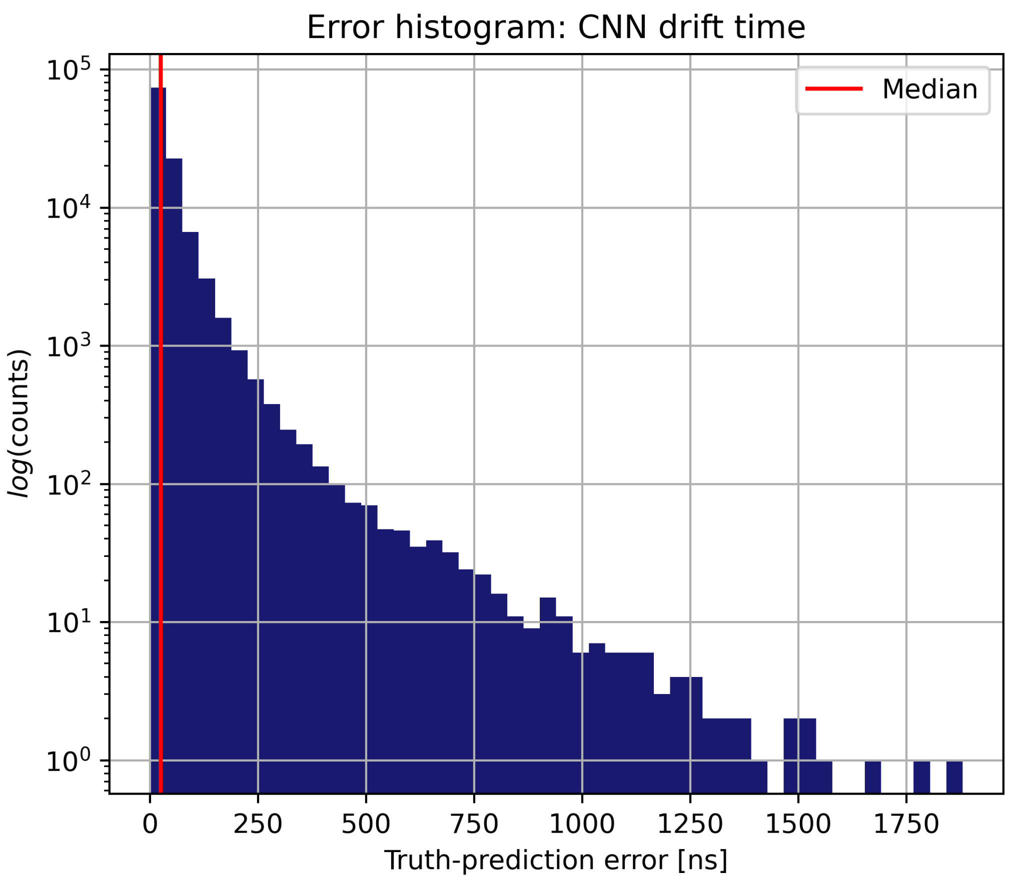


Figure 57: *Error histogram for the CNN. The occurrences of the prediction errors are counted. The histogram exhibits a sharp decline after the initial count, indicating a stable performance of the model.*

The distribution of prediction errors for the CNN model is shown in Figure 57. This histogram represents the frequency of the differences between true and predicted values. It exhibits a rapid decline in error occurrences after the initial peak, highlighting the model's stable performance.

To further investigate potential issues in specific drift time regions, a regression scatter plot of predicted versus true times is presented in Figure 58.

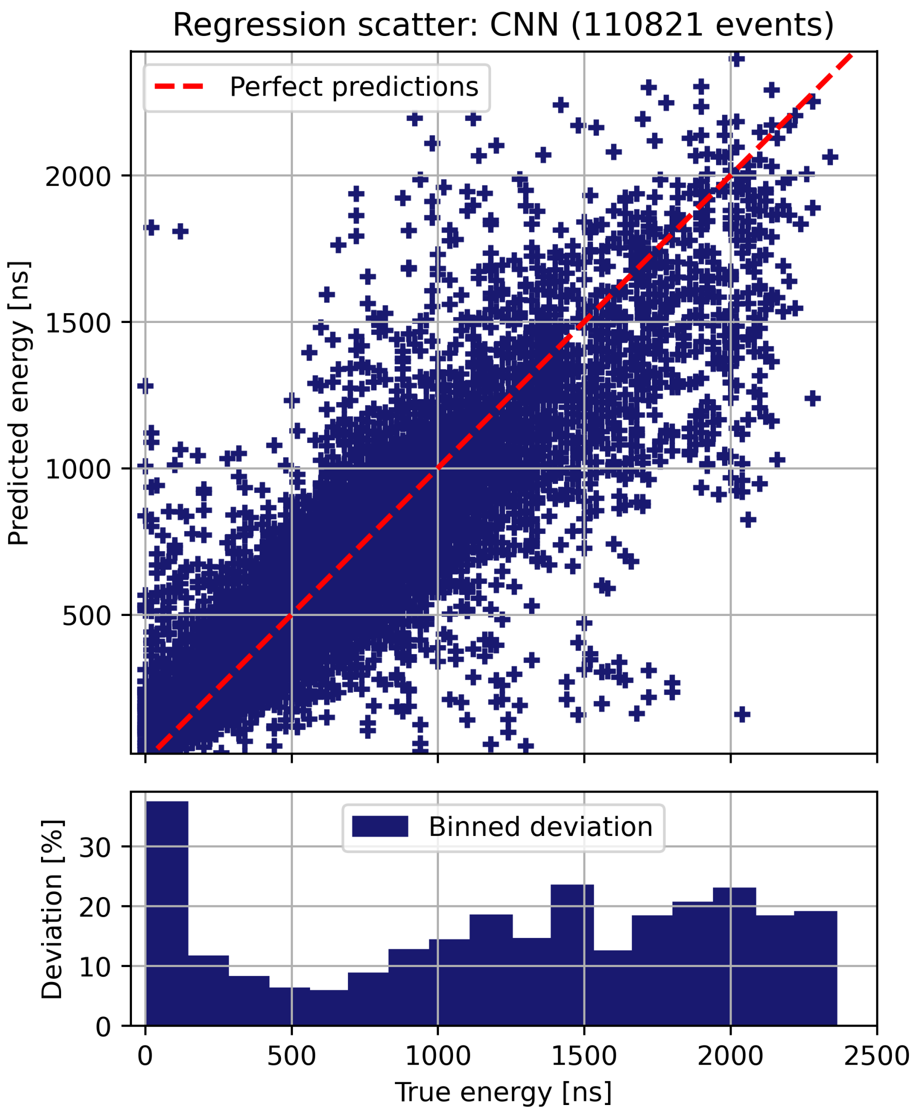


Figure 58: *Regression scatter plot of the CNN model. Predicted versus true times in ns. A very broad distribution of the analyzed samples can be seen, which commonly reflects a poor model performance.*

Figure 58 reveals a broad distribution of truth-prediction points around the perfect prediction line, indicating a poorer performance than what was gauged with the CNN before and also contradicts the findings from Figures 54 and 57, prompting additional analysis.

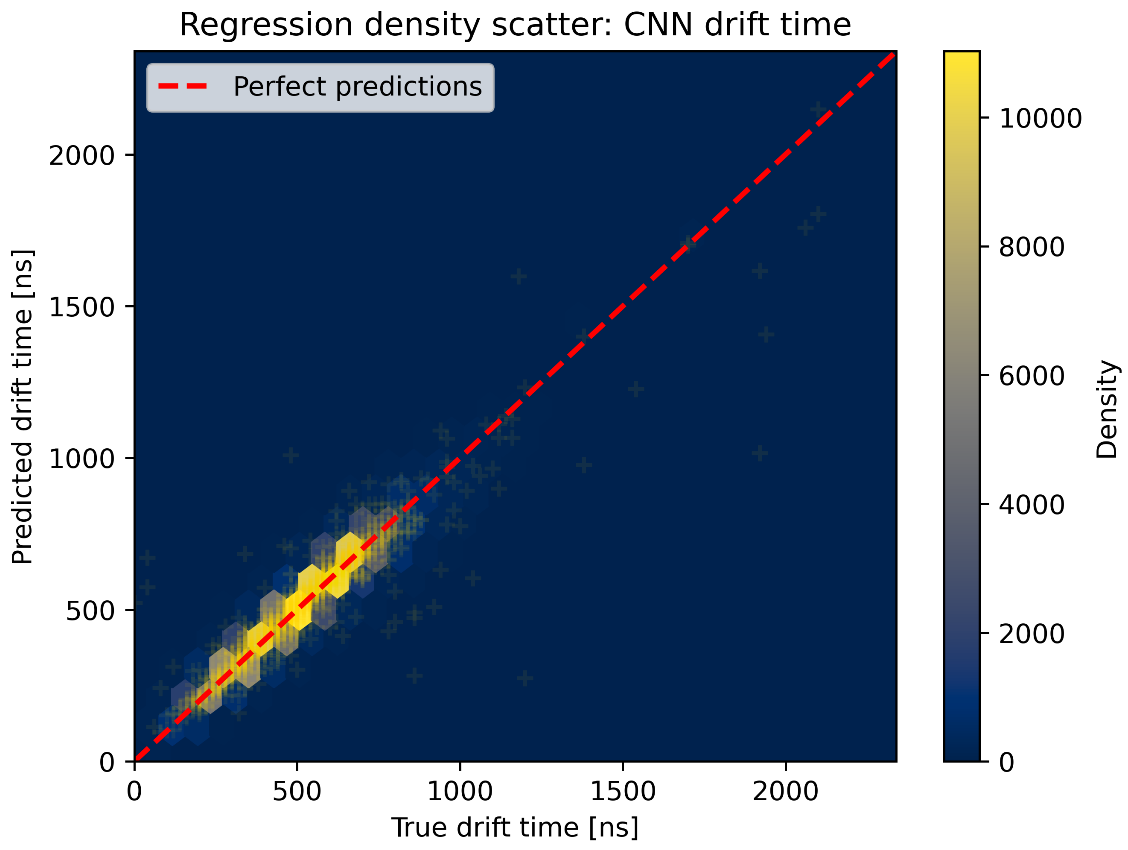


Figure 59: *Regression density "starry" plot for the CNN model. The density indicates the location of the cluster of points. It becomes now clear that the distribution of the points lies along the perfect prediction line, signaling a great model performance.*

The regression density "starry" plot in Figure 59, addresses this discrepancy by visualizing the density of true-prediction pairs. Unlike the scatter plot, this visualization highlights that the majority of the data points lie along the perfect prediction line, confirming strong model performance. The density-based approach resolves the ambiguity seen in the scatter plot in Figure 58, aligning with previous observations and providing a more nuanced understanding of the model's predictive accuracy.

While this discussion primarily focuses on the CNN model, other models produced similar results, albeit with slightly inferior performance and without any unusual behaviors warranting detailed discussion.

Once the drift time is predicted, the depth of interaction can be calculated using the electric field E as follows:

$$E = \frac{V}{d} \tag{30}$$

where V is the voltage and d the detector crystal's thickness. This leads to the calculation of the drift velocity v_d [83]:

$$v_d = \mu_e \cdot E \tag{31}$$

where μ_e is the electron mobility, a material-dependent constant, allowing the calculation of depth of interaction z as:

$$z = v_d \cdot t_d \tag{32}$$

where t_d is the predicted drift time.

The thickness of the semiconductor was specified as 10 mm in the manual, and for electron mobility μ_e the value 1350 cm^2 / (V $\cdot$ s) was used, a typical value for CZT-based detectors [86].

A decision was made not to train the AI models to directly predict depth of interaction due to the variability in applied voltage across different applications. If the detector's applied voltage differs from the settings used during model training, predictions of depth of interaction would likely be inaccurate. Predicting drift time alone avoids this issue, as it is independent of voltage, ensuring consistent results across different settings. Further tests may consolidate this statement.

4.6 Pixel pattern

Each interaction in the detector generates a certain pattern upon impacting on its pixelated grid. Furthermore, this pattern is also influenced by multiple factors alongside the type of particle (radiation), including event energy and the type of interaction occurring within the detector. These variables alone result in a diverse range of possible configurations, which might lead to similar pixel patterns.

In the received Cs-137 recordings, the majority of the events fall within the energy range of 0.1 MeV to 0.7 MeV (Cs-137 source, cf. Chapter 4.1 and Figure 37). Consequently, this data set is largely composed of interactions governed by the photoeffect and Compton scattering and less by pair production. In Chapter 2 the theory of these interactions are clarified. The subsequent sections focus on determining the resulting pixel patterns from the recorded interactions, beginning with an analysis of the probable pixel responses based on the type of photonic event interaction.

Inherent to the photoelectric effect's relatively low kinetic energy compared to other interaction types, the photoelectron travels a short distance before losing its energy, eventually ceasing to excite electrons for enabling to cross the band gap. As a result, a photoelectric event is highly localized, typically concentrating the full deposited energy within a single pixel – a feature that greatly benefits spectroscopy [46].

Compton scattering typically occurs at higher photon energy ranges compared to the photoelectric effect, enabling the recoiled photon to possibly undergo further interactions within the detector, either as Compton scattering again or photoelectric effect, creating a cluster of pixels (multi-event) with an diffuse energy continuum [57].

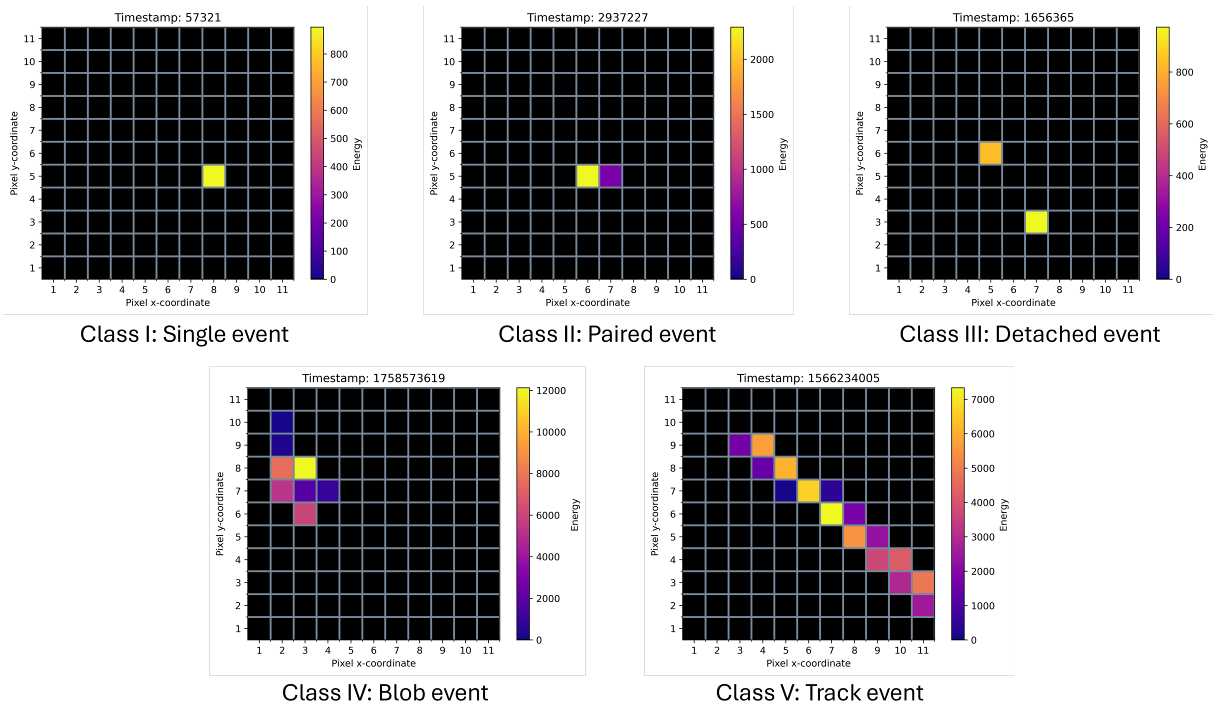
In spectroscopy, Compton scattering can present challenges, as the scattered photon may escape the detector. This differs from the photoelectric effect, where the photon's full energy is absorbed by the material. Such photon escape can result in an incomplete energy deposition, potentially leading to an inaccurate reading of the original photon's energy [46].

In the sequence of the pair production mechanism two charged particles – electron and positron – are generated capable of ionizing the crystal's atoms, followed by annihilation producing two photons that may also interact (e.g., via photoelectric effect and/or Compton scattering), the resulting pixel pattern can become complex [59]. It may form a cluster or even distinct, disjointed segments on the pixel canvas, depending on the interactions involved.

When photons interact through pair creation and annihilation, complications can arise due to the potential escape of the annihilation-borne photons from the detector, which may result in an inaccurate energy readout [46].

To assign particle types, their interaction, and energy to certain pixel patterns, an initial classification of the pixel patterns themselves is beneficial. After thorough analysis of the recordings and the insights of each interaction's pixel response outlined earlier, five distinct classes of pixel patterns have been defined, as illustrated in Figure 60.

- Class I represents events where only a single pixel is triggered
- Class II, referred to as paired events, covers instances where two adjacent pixels are activated, which may also include diagonal neighbors
- Class III, termed detached events, involves patterns with at least two pixels or pixel clusters that are not connected
- Class IV known as blob events, captures clusters of three or more pixels arranged in a rather spherical formation
- Class V, denoted to as track events, consists of patterns where three or more pixels form a more elongated structure



Class I: Single event

Class II: Paired event

Class III: Detached event

Class IV: Blob event

Class V: Track event

Figure 60: *All pixel pattern classes. Classes depend on the shape of the 11 by 11 pixel pattern.*

This classification serves as an initial step to differentiate between complex particle interactions based on pixel pattern characteristics.

To accurately label the pixel patterns, again a conventional function-based approach was used. For Classes I and II, the calculations are straightforward. However, for Class III, a breadth-first search (BFS) algorithm – commonly employed in graph theory for traversing nodes – was implemented.

The BFS algorithm operates as follows: starting from a designated source node, it identifies and marks all adjacent (neighboring) nodes. Once all reachable neighbors are marked, the algorithm proceeds to the next unmarked neighbor as the new source node, marking its neighbors in turn. This process repeats until there are no unmarked nodes remaining. The term "breadth-first" arises from the method's traversal pattern: nodes on the same hierarchical level are visited first, before moving to deeper levels. To illustrate this distinction, Figure 61 includes a comparison with the Depth-first search (DFS) algorithm, which follows a similar process but prioritizes vertical or deeper traversal paths [52]. Either algorithm would yield the same outcome in this case, as both ultimately explore all connected nodes.

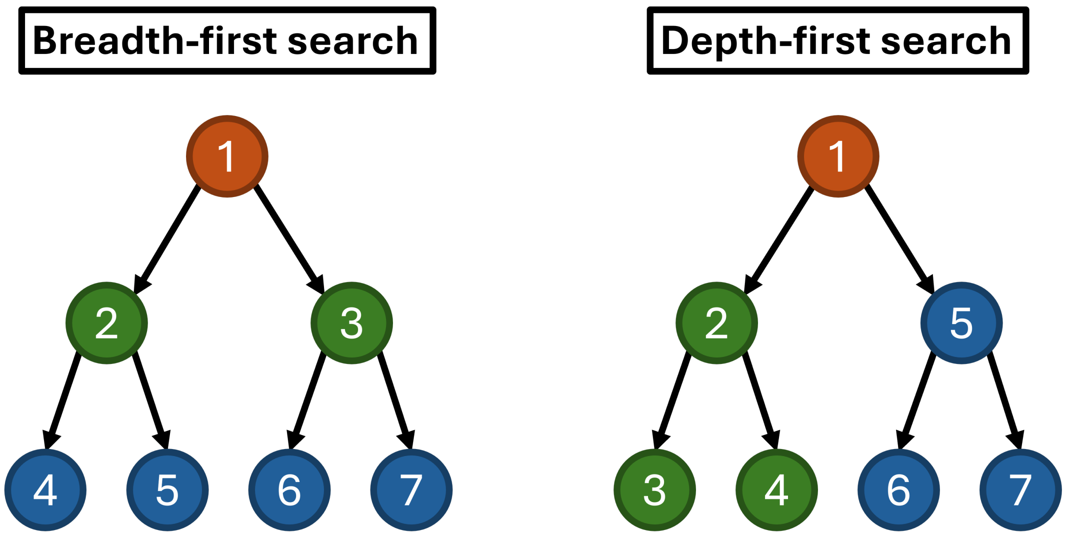


Figure 61: *Breadth-first search (BFS) and Depth-first search (DFS) working principles visualized. These represent two distinct strategies for traversing a graph. The numbers indicate the chronological order in which the nodes are visited. BFS explores all nodes at the same level before proceeding to deeper levels, while DFS prioritizes depth by exploring each path to its fullest extent before backtracking (adapted from [37]).*

Applying this method to the pixel pattern classification: each pattern is treated as a graph, where the pixel with the highest energy serves as the starting source node. The key insight here is that if, upon completing the BFS traversal, there remain unmarked pixels across the entire pixel canvas, then the pattern must be an instance of a detached event, Class III, as there are no neighboring pixels present on which the BFS algorithm could traverse further on, therefore, the unmarked pixels represent separate, unconnected pixel clusters. In its essence, the event is composed of two or more disjointed graphs.

If the pixel structure is identified as a consolidated shape containing more than two pixels, the next step is to determine whether it belongs to Class IV (blob) or Class V (track) patterns. To make this distinction, the ratio of the principal axes is calculated.

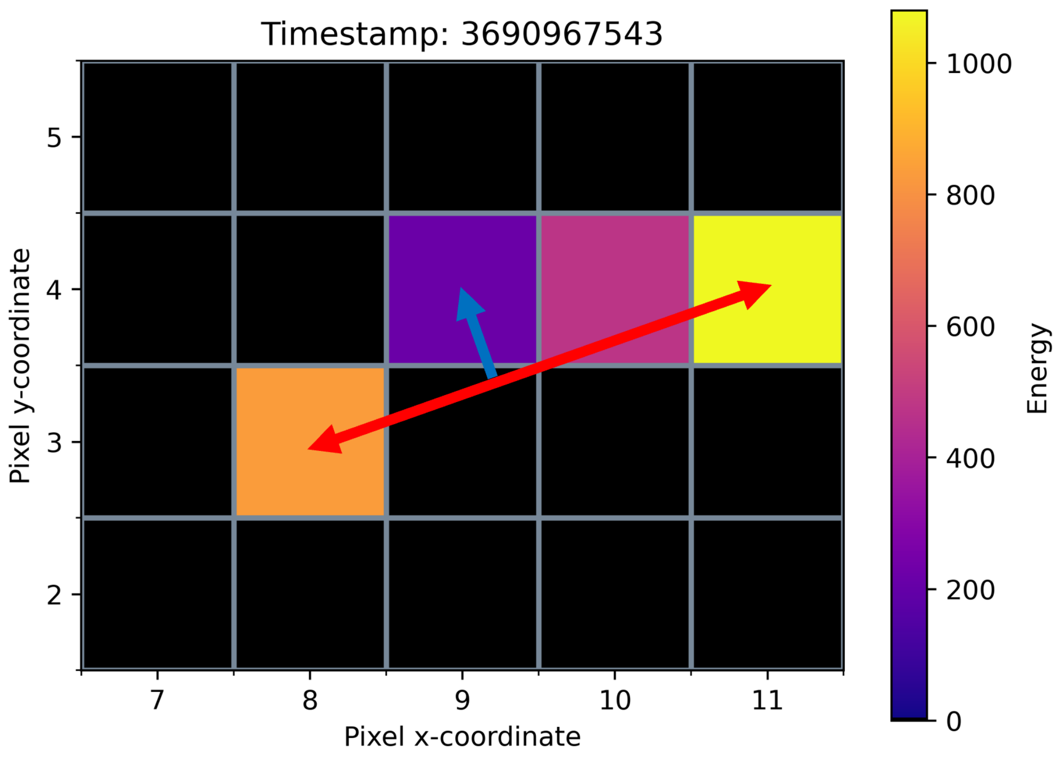


Figure 62: *The principle axes are drawn exemplarily on a pixel pattern. From the pixel with the most energy to the most far pixel reaches M_1 (first axis, highlighted in red), to that perpendicular lies M_2 (secondary axis, highlighted in blue) pointing to the furthest pixel possible.*

As illustrated in Figure 62, the method operates as follows: the primary axis, M_1, is defined as the line connecting the pixel with the highest energy to the most distant pixel within the pattern (highlighted in red). The secondary axis, M_2, represents the longest perpendicular distance from M_1 to any pixel in the structure (highlighted in blue).

To differentiate between blob and track patterns, a "shape threshold" c is introduced, based on the ratio of these principal axes as shown in Equation (33):

$$f(P,c) = \left\{ \begin{array}{ll} \text{Track} & \text{for } \frac{M_2}{M_1} \leq c \\ \text{Blob} & \text{for } \frac{M_2}{M_1} > c \end{array} \right\} \tag{33}$$

where P denotes the pixel matrix from the detector, with M_1 and M_2 calculated accordingly. This threshold effectively categorizes between more circular blob patterns and elongated track patterns. In this work, a threshold value of $c = 0.4$ was selected, as it proved to be a suitable criterion when visually assessing the patterns, as seen in Figures 63.

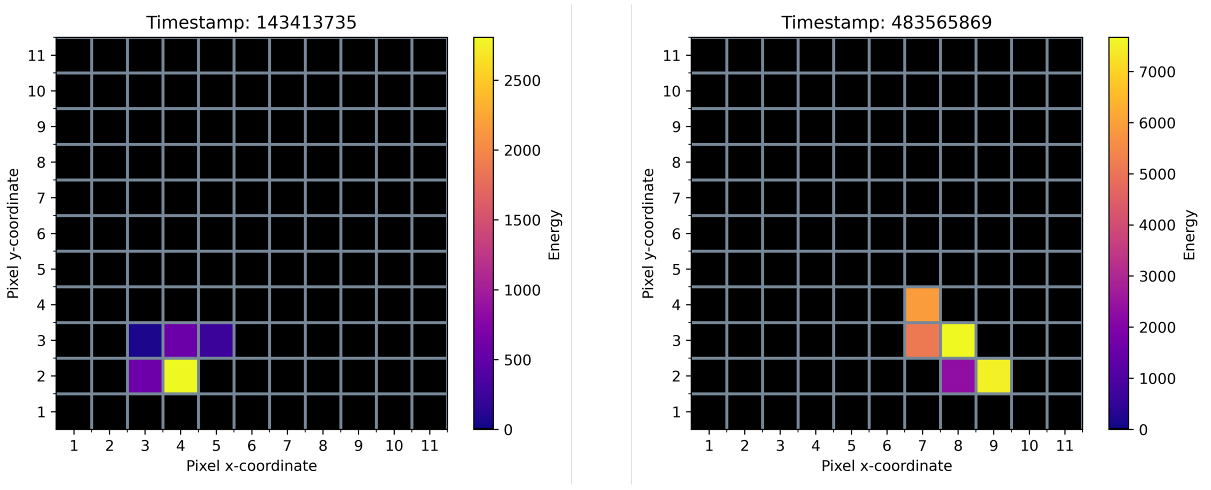


Figure 63: *(Left) One example of the blob class. (Right) Example of the track class. The blob exhibits a cluster-like shape, while the track displays an elongated structure.*

As the sole classification task in this thesis, y was discrete, representing the corresponding class types using the function described afore. Since Classes I and II are straightforward to classify, they were excluded from the training, thus, only Class III, IV and V remained as targets. The opportunity arose to assemble X as a matrix of the number of anode signals (cathode information was not necessary in this task) times the 160 amplitude values, harnessing the feature extraction capabilities inherent in DL algorithms (cf. Chapter 3.3.3 and Figure 25), for minimal – and foremost automatic – data preparation in advance.

Despite the initial promise, the results form this DL approach did not meet expectations, though the steadily declining loss function (up to a significant high threshold value eventually) indicated potential. A falling loss function, yet an unsatisfactory performance, might suggest that the task demands a substantially greater model complexity. However, due to the prior requirement that these models be deployable on an FPGA, with its storage constraints, an alternative approach was devised: The input data, X, was arranged as an 11×11 matrix (representing the pixel canvas) with the energy values mapped to their respective coordinates (untriggered pixels had a zero entry).

This arrangement not only improved execution time, but also elevated the performance of the DL algorithms. The CNN models benefited from grasping the pixel canvas as two-dimensional image, a format in which CNNs evidently excel. Similarly, RNN/LSTM models gained an advantage from the interdependencies along the row-by-row sequence. Both ML and DL models were trained on this setup.

Before discussing architecture and results, an additional challenge required attention.

Unbalanced classification

During training data preparation, it was observed that the data set exhibited significant class imbalance. Specifically, Class IV contained five times fewer samples, and Class V had fifteen times fewer samples, compared to the more common "detached" class. This imbalance can lead to biased weight adjustments in favor of the majority class, with the effect worsening as the disparity in sample counts increases.

To address this problem, several auxiliary means are available. A more intricate approach involves generating synthetic examples for the minority classes to supplement the data set. This method is challenging, as the underlying pattern of the minority class must be imitated and moreover altered in such a way that it closely simulates natural behavior emanating from a realistic probability distribution [10].

Alternatively, simpler techniques involve either reducing the majority class samples or duplicating the minority class samples to balance the data set, known as undersampling and oversampling, respectively [35]. In this thesis, oversampling was applied, randomly duplicating samples from Class IV and Class V until both of their sample count approximately matched that of Class III. This approach moderately improved the performance across all models.

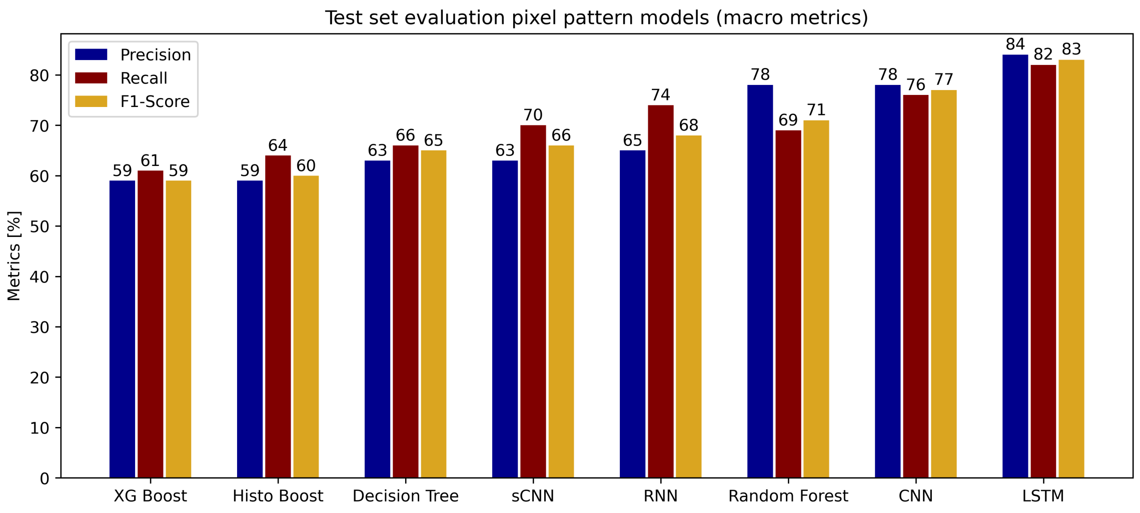


Figure 64: *Results of the pixel pattern task. Metrics such as Precision, Recall, and F1-Score (expressed as percentages) are presented to evaluate performance. The macro-average metrics indicate equal weighting across all classes. The DL models demonstrated robust results, with the Random Forest classifier performing remarkably well, exceeding expectations.*

Since this is a classification task, different metrics were used to evaluate performance, namely, Precision (positive predictive value),

$$\text{Precision} = \frac{\text{True positives}}{\text{True positives} + \text{False positives}} \tag{34}$$

Recall (sensitivity),

$$\text{Recall} = \frac{\text{True positives}}{\text{True positives} + \text{False negatives}} \tag{35}$$

and F1-Score (harmonic mean of precision and recall),

$$\text{F1-Score} = \frac{2 \cdot \text{True positives}}{2 \cdot \text{True positives} + \text{False positives} + \text{False negatives}} \tag{36}$$

The classes True positives, False negatives, etc., shall be exemplary described via track labels. A True positive occurs when a sample with a track label is correctly identified as such, while a False negative mislabels a track sample as non-track. Conversely, a

False positive labels a non-track sample as track, and a True negative correctly identifies non-track samples.

Precision is prioritized when the cost of a False positive is high, while Recall is preferred if missing a positive sample is costly. For example, in security applications, a mine detector would place more emphasis on Recall, as missing a positive case surely has more severe consequences than false positives.

In this work, however, both metrics are considered equally important, making the F1-Score the preferred measure as it balances Precision and Recall. Given the multi-class nature of this classification, for calculating the global metric (as seen in Figure 64), each class is treated as a binary classification problem (e.g., track vs. non-track), eventually calculating the harmonic mean across the three classes' metric result. To ensure equal importance for each class, macro averaging was used, giving each class the same weight.

The results presented in Figure 64 reaffirm superior performance of DL. As mentioned before, the LSTM network was included in this thesis, since it delivers here a strong result, due to the capacity of capturing long-term interdependencies, here operatively over 11 sequential arrays, resulting in a satisfactory recognition of certain pixel shapes.

The CNN model also performed well, which aligns with their effectiveness in shape detection – a core use case for CNNs. By contrast, tree-based algorithms underperformed as expected, given that image-like pixel canvases are not typical inputs for these models. However, the Random Forest achieved a moderate result, suggesting that the binary nature of the tree nodes might still isolate certain core features effectively, even in a rather unconventional setting.

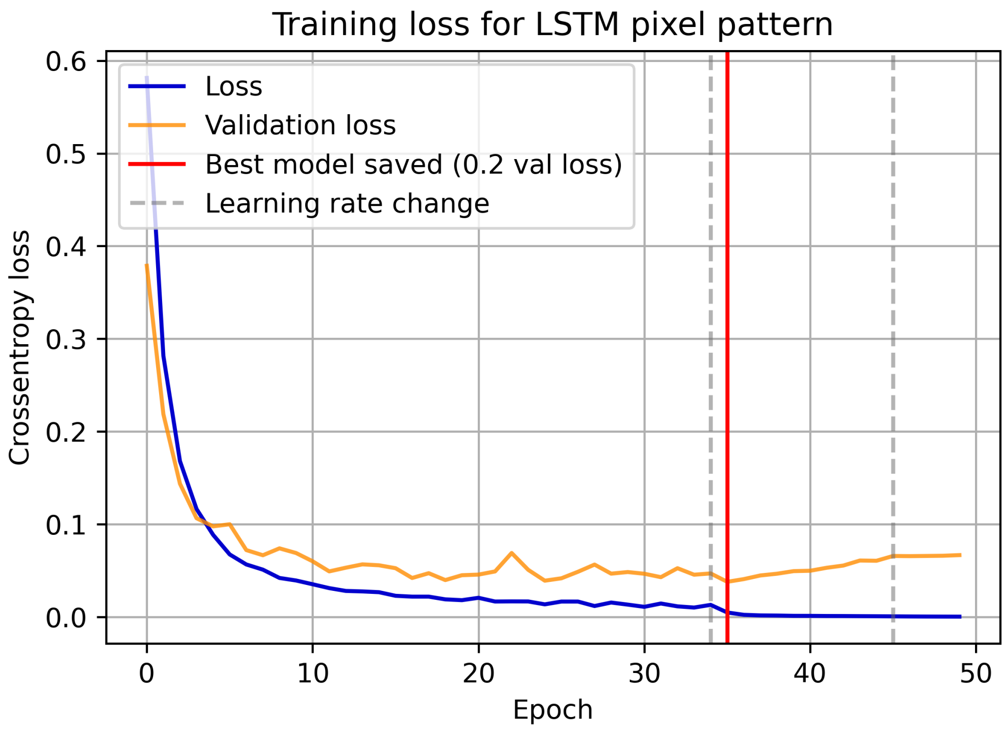


Figure 65: *LSTMs training history. The RMSE loss exhibits a steady decline, indicating effective learning. Notably, the training process concluded after approximately 50 epochs, which, upon test data evaluation, reflects the model's excellent performance.*

The training history of the best-performing model – the LSTM – depicted in Figure 65. Its training progression resembles that of the drift time CNN, with a continuous decline in the loss function. However, after one of the learning rate adjustments, signs of overfitting emerge. This issue is mitigated by linking the model-saving mechanism to performance on the validation data set, ensuring better generalization.

In classification tasks, it is crucial to evaluate the classes' performance individually, as poor predictions can often be traced to a single misunderstood class, even when other classes are well-handled. To assess this, the confusion matrix is used.

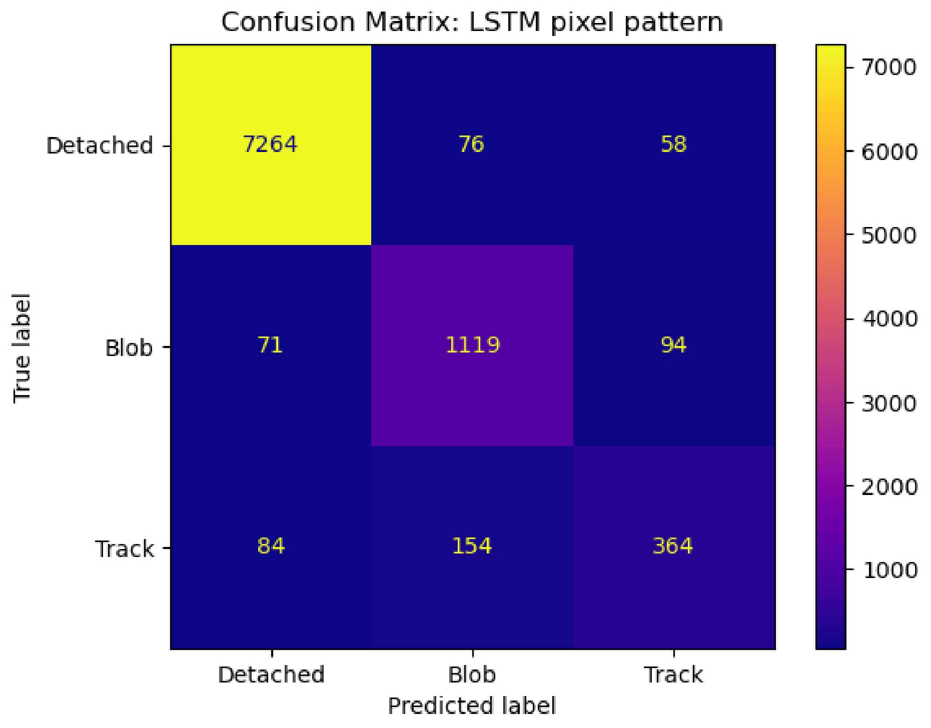


Figure 66: *Confusion matrix for the LSTM model. The true labels are plotted against the predicted labels. A well-performing model shows most samples aligned along the diagonal, where the predicted class matches the true class.*

For an effective model, the confusion matrix should exhibit a strong diagonal line, representing accurate predictions within each class. Any deviation from this diagonal indicates misclassifications. The confusion matrix functions as a discrete equivalent of the regression evaluation scatter plots discussed in Chapters 4.4 and 4.5.

The LSTM model demonstrates strong performance (Figure 66), with the majority of each class's samples falling along the diagonal, indicating accurate classification. The Detached class, in particular, was well-recognized by all models. This is noteworthy because, while distinguishing detached pixel patterns is visually intuitive for humans, it is algorithmically complex due to the variety of scenarios that must be accounted for – a challenge tackled here with a graph theory approach.

Some confusion is observed between the Track and Blob classes: a notable number of Track samples were misclassified as Blob. The complexity of distinguishing these two classes was evident even during the design of the conventional function, as some patterns could logically and visually belong to both categories. This ambiguity highlights the model's – and AI in general – human-like competence in assessing nuanced patterns.

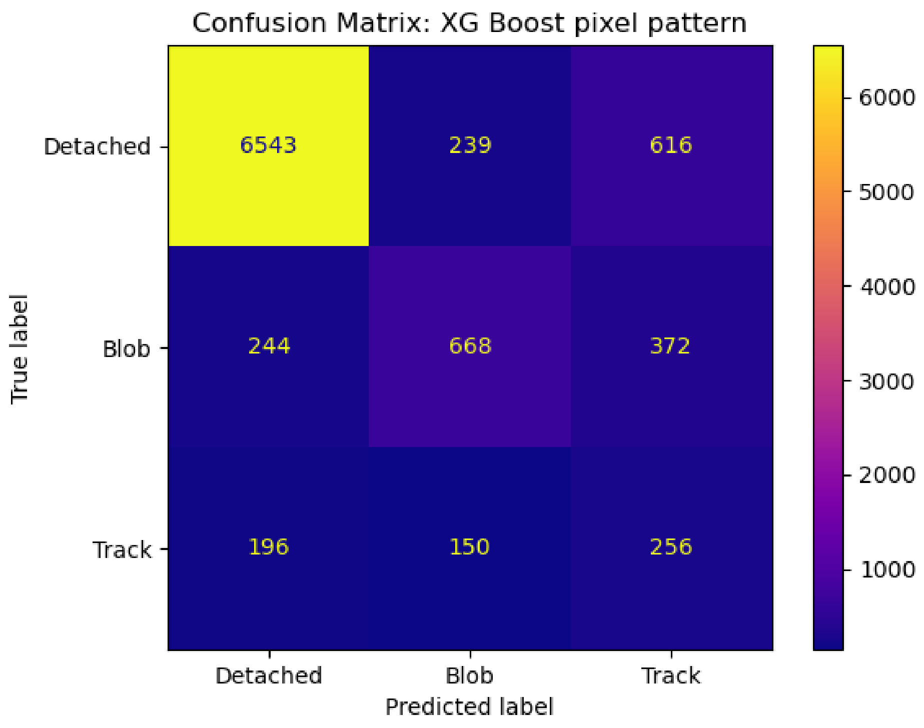


Figure 67: *Confusion matrix of the XG Boost model. This model performed poorly, as evident from the distribution of true track class samples distribute across all predicted labels. This pattern suggests the model relies on random guessing and fails to understand the underlying task.*

In Figure 67, a poor result is demonstrated by the XG Boost model. While it correctly classified the majority of Detached events, the Track class suffered from remarkable misclassification. This suggests that XG Boost struggled to grasp the characteristics of the Track class, performing only marginally better than random guessing for this category.

Again, in this classification task, the model does not provide a definitive determination of the particle or the interaction involved. Instead, it serves as a preliminary indicator, as additional data is essential for a conclusive prediction. For example, incorporating energy data can help further narrow down possible interaction types (cf. Figure 3). Pair production, for instance, is unlikely to produce a single-pixel event. Additionally, combining depth of interaction data with subpixel measurements enables modeling of the interaction in 3D, strengthening the foundation for gauging radiation and interaction type to greater accuracy.

4.7 Subpixel resolution

The GDS-100 detector uses pixels to describe the x and y coordinates of the particles' impact point. Specifically, an 11×11 matrix corresponding to a 22 mm × 22 mm detection area. However, it allows for even finer localization through subpixel resolution, aiding in the computation of 3D spatial positions within the detector.

The manufacturer's manual does not detail a specific approach for calculating subpixel precision, therefore, a naïve approach can be employed, using a center-of-mass (COM) calculation based on the energy deposited in each pixel. This method can be expressed as:

$$\vec{r}_{\mathrm{COM}} = \frac{\sum_{i=1}^{121} E_i \vec{r}_i}{\sum_{i=1}^{121} E_i} \tag{37}$$

where E_i is the energy deposited in each pixel and $\vec{r}$ represents the coordinates of pixel i for $i = 1, ..., 121$.

For this method, only certain multi-pixel events are relevant. Precisely, events where two to four adjacent pixels are triggered, as illustrated in Figure 68 (the configuration may also be rotated).

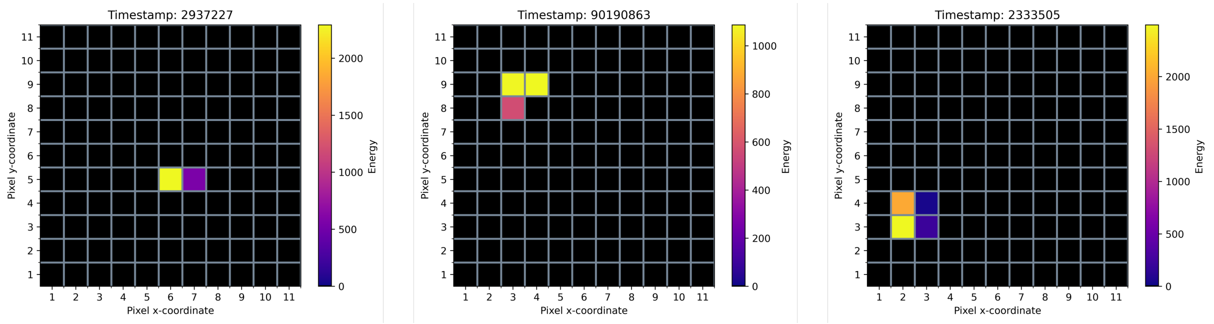


Figure 68: *Potential candidate configurations suitable for probing the subpixel task must adhere to the condition that all neighboring elements are adjacent.*

Since this approach offers limited room for interpretation, a complexity was introduced based on findings from [42]. In that study, researchers observed that multi-pixel events – similar to those investigated here (2, 3 and 4 pixel events) – are typically only triggered when a particle impact occurs near the pixel edge.

This insight was adopted here to introduce an artificial non-linearity to the problem, serving as a justification for the use of AI. The goal is not to impose a definitive subpixel calculation on the detector in deployment, but rather to demonstrate potential approaches.

The basic principle is as follows: after filtering the data set to isolate the desired events, the center of mass (COM) was computed. If the COM was found outside a specified radius, centered on the pixel with the highest energy deposit, the COM location was adjusted. The adjustment involved shifting the pixel along the line connecting the pixel's center to the COM, until it intersected with the pixel grid, as visually represented in Figure 69.

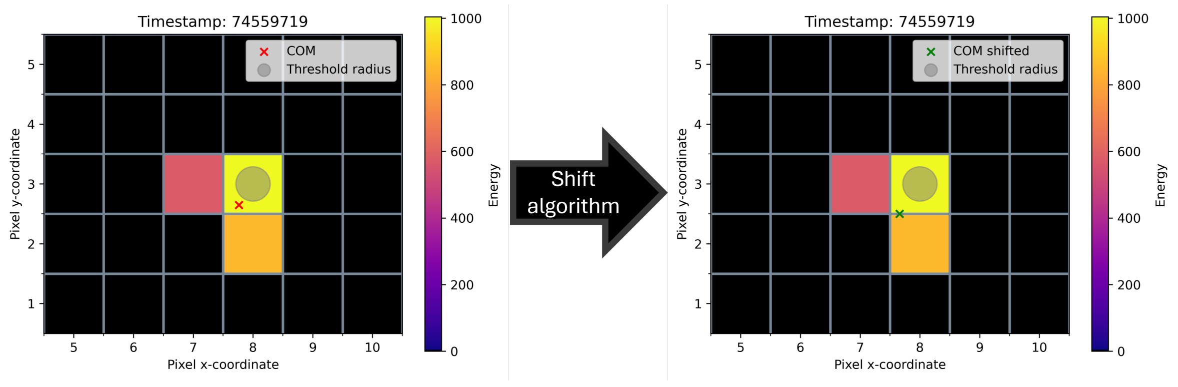


Figure 69: *Visualization of the shift algorithm demonstrating its operation: if the COM lies outside the defined threshold radius, it is shifted toward the pixel grid, otherwise, the COM remains in its original position.*

A radius of 0.4 mm (0.2 in unitless terms) was used in this study. However, this approach does conflict with the pixel pattern analysis, as previously shown that specific pixel patterns can be indicative of particular types of radiation and of interaction, in this case the investigated events one would refer to as Class IV "blob" events (e.g., from Compton scattering).

To resolve this conflict comprehensively, all the predictions stemming from the outcomes of both the pixel pattern and the subpixel resolution task are presented to the user. The final interpretation is thereby entrusted to the user's discernment. At heart, this thesis is intended as a rigorous starting point for further investigations into detector deployment fostered by Machine Learning.

The training set X for the AI models were compiled as 4×3 matrix, capturing the x and y positions as well as the energy values of up to four triggered pixels. For samples with fewer than four triggered pixels, zero-padding was applied to fill the remaining entries.

The decision to perform the feature extraction again manually, rather than allowing the DL algorithms to perform by themselves, follows the identical line of reasoning previously outlined in Chapter 4.6. The labels y are 2D vectors that represent the finer impact points on the 2D pixel grid.

Another mechanism is needed for preparing the training data effectively: Normalization. X is structured as tabular data, but the columns differ significantly in scale: x and y coordinates range from 1 to 11, while the energy values reaches into the thousands. These disparities complicate training, as the neural network's weights must adjust to this uneven distribution, which can slow down training or, in the worst case, degrade the overall performance [36].

To address this, the energy values were normalized using Min-Max Scaling, defined as:

$$X_{\text{norm}} = \frac{X - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}} \tag{38}$$

where X represents a sample, and X_{min} and X_{max} denote the global minimum and maximum values across all observations, respectively [36]. This normalized energy value was then multiplied by 11 to match the scale of the x and y coordinates.

Normalization was only applied to the DL training data set, whereas tree-based ML models can handle varying scales without normalization, as they split features into binary categories at each node, merely shifting the threshold without affecting overall results, unlike DL algorithms that rely on weight adjustments for feature scaling [18] (cf. Chapter 3.2.1).

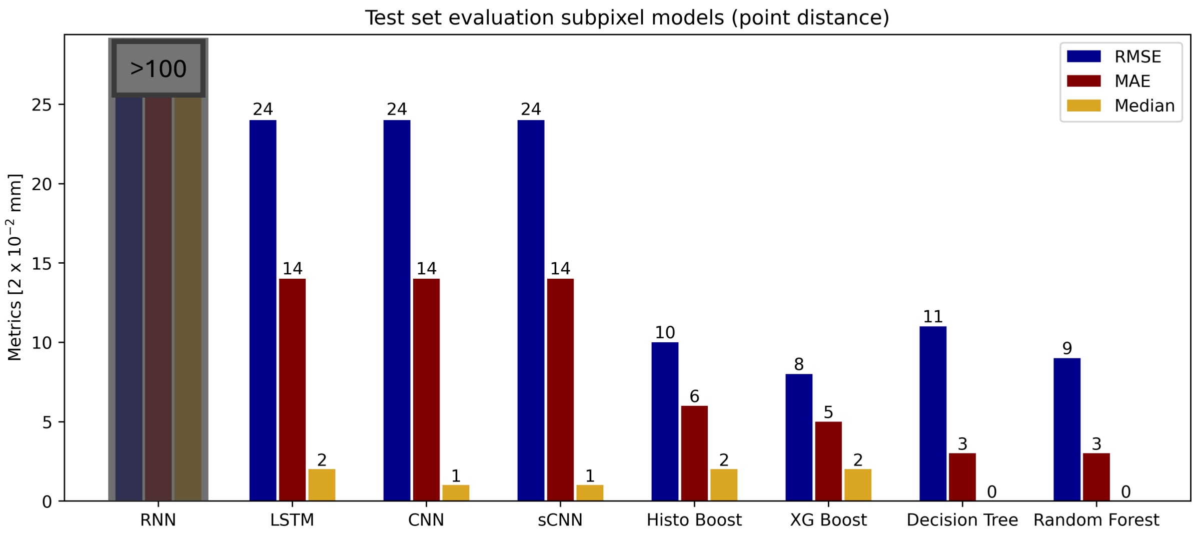


Figure 70: *The regression metrics for the subpixel resolution task, including RMSE, MAE and median error, are reported in units of* 2×10^{-2} *mm. This choice reflects the fact that the detector's spatial measurements are twice the scale of the unitless pixel array. The ML models demonstrate strong performance on this task, whereas the RNN fails to produce meaningful results.*

For the subpixel task, the evaluation required slight adjustment since, unlike other tasks, the target variable, y, was structured as a 2D vector. Thus, the error was measured as the Euclidean distance between predicted and true coordinates — the smaller the distance, the better the prediction. The resulting metrics are shown in Figure 70.

Since all testing was conducted on an 11×11 unitless grid, these results can be readily adapted to a 22 mm $\times$ 22 mm detector grid by multiplying the distance by a factor of 2, as indicated in the metrics' units in Figure 70. For visual clarity, the 11×11 grid was scaled by a factor of 100.

Not surprisingly, the Machine Learning models performed exceptionally well, given that X was structured as traditional tabular data – a format in which these models typically excel. All tree-based models yielded similar results, with a median error of approximately 1% of the grid size.

The Deep Learning models also performed comparatively well, however, they produced outliers, reflected in a relatively high RMSE compared to the median error. One notable exception was the RNN, which failed at this task, yielding errors that substantially exceeded the metrics observed across all other models. Conversely, the related LSTM model performed adequately, suggesting that the RNN's architecture may have been too simplistic for this task.

The considerably higher MAE compared to the median error raises an interesting question, which will be explored in the subsequent evaluations.

The training history for the DL models exhibit a peculiar behavior, as shown in Figure 71.

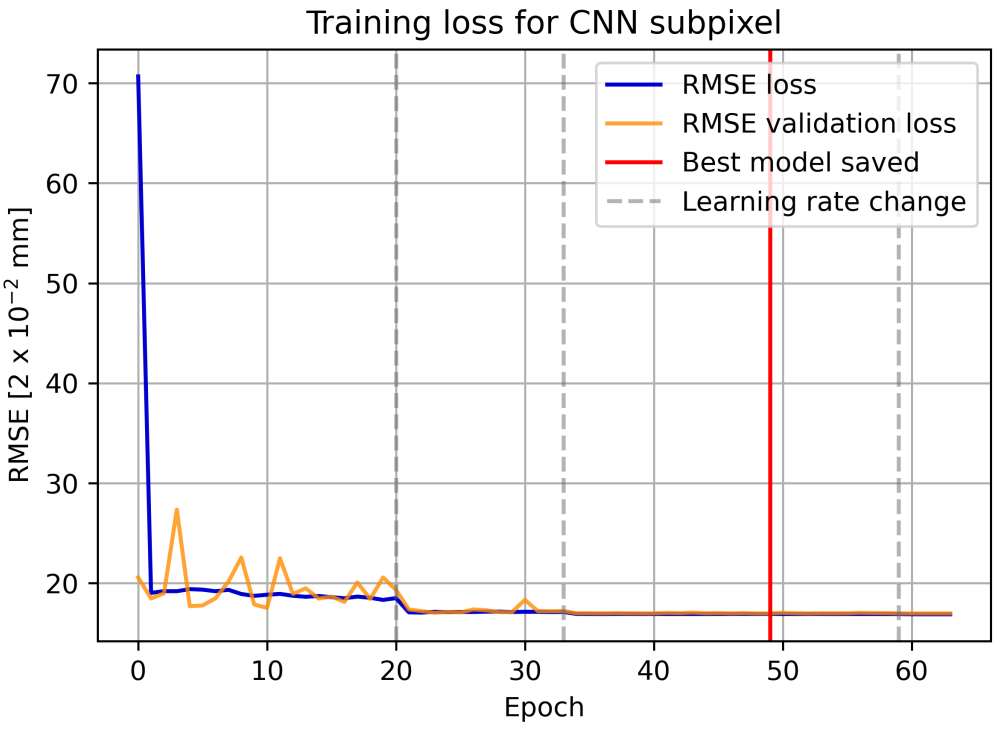


Figure 71: *Training history for the subpixel CNN. The loss' object is the RMSE metric. A sudden drop followed by a flat progression suggests issues in the training process. Also, the loss was only minimally affected by adjustments to the learning rate. This result indicates the need for modifications to the model architecture.*

The training loss does not decline smoothly, instead, it shows a sudden drop followed by a relatively flat progression. This behavior indicates that the model easily fits certain samples while struggling with others, which aligns with the compilation of the task: The ordinary samples correspond to simple COM calculations, while the more challenging cases involve the shifted COMs. To outperform the ML models, the DL architectures must be revised for further improvements.

In the discussion before a critical question arose on why the MAE and the RMSE are substantially higher than the median. This discrepancy can be explained by examining the error distribution displayed in Figure 72.

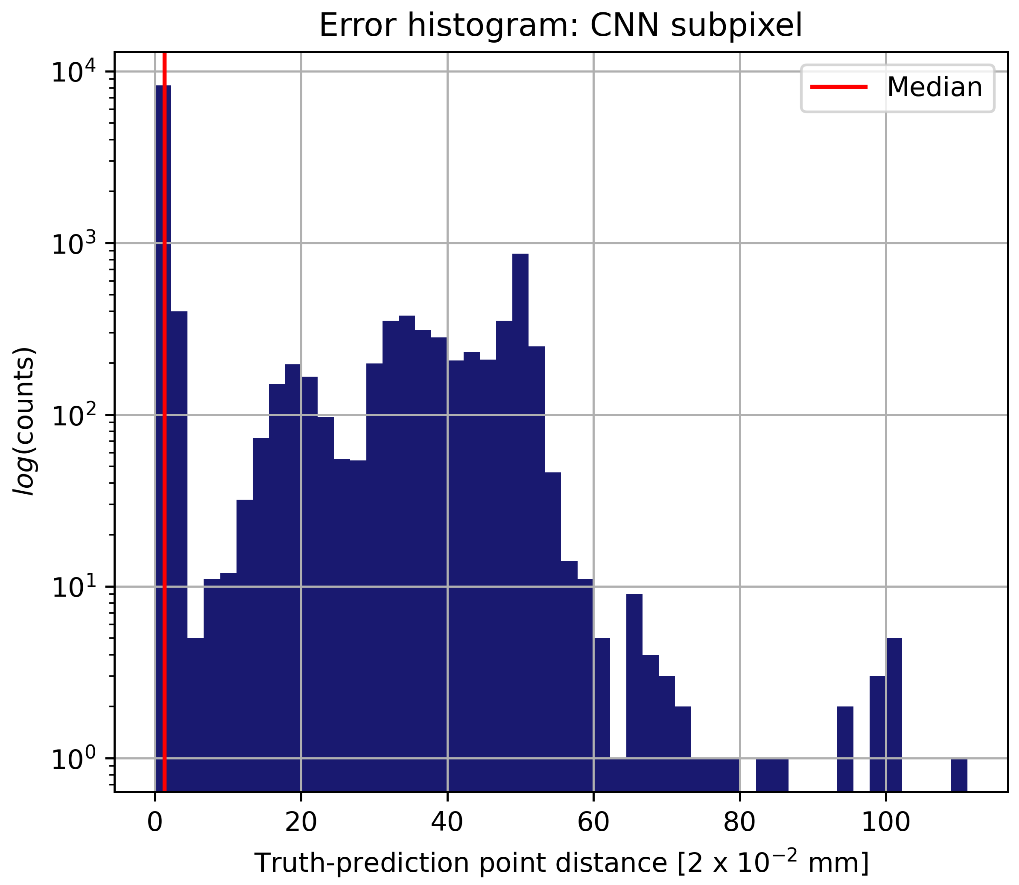


Figure 72: *Error histogram for the CNN. While most predictions are close to zero, a non-negligible proportion of outliers contributes to the higher RMSE and MAE values compared to the median error.*

As shown in Figure 72, the majority of predictions are concentrated near to zero. However, a moderate number of the predictions produced outliers, particularly those associated with shifted COMs (as will be assessed in Figure 74), in the range of 20 to 60 units of point distances. These outliers significantly inflate both the RMSE and MAE metrics, explaining the observed disparity with the median error.

In contrast, the Random Forest demonstrates exceptional performance, as reflected in the error histogram in Figure 73.

The histogram reveals that the majority of errors are close to zero, with a steep drop in counts at higher values. This distribution reflects the model's reliability, as corroborated by the metrics in Figure 70.

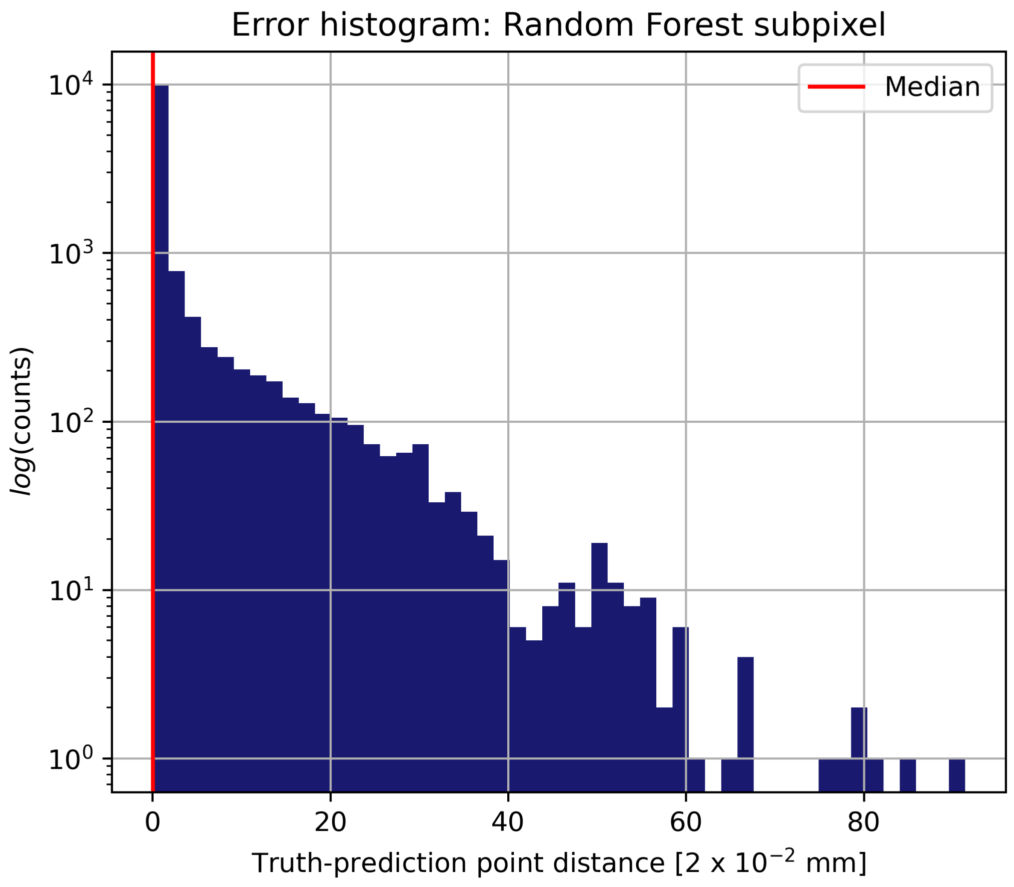


Figure 73: *Error histogram for the Random Forest. Most errors are concentrated near zero, indicating the model's strong performance on the test data set.*

To further analyze the Random Forest's performance, individual predictions are depicted in Figure 74. The visualization shows that problematic predictions primarily occur for shifted COM cases. Some are believed to be outside the threshold radius (an example is highlighted in magenta) while others exhibit grid misalignment (an example is highlighted in dark red). However, predictions for actual COM samples are accurate, indicated by the various shades of green samples.

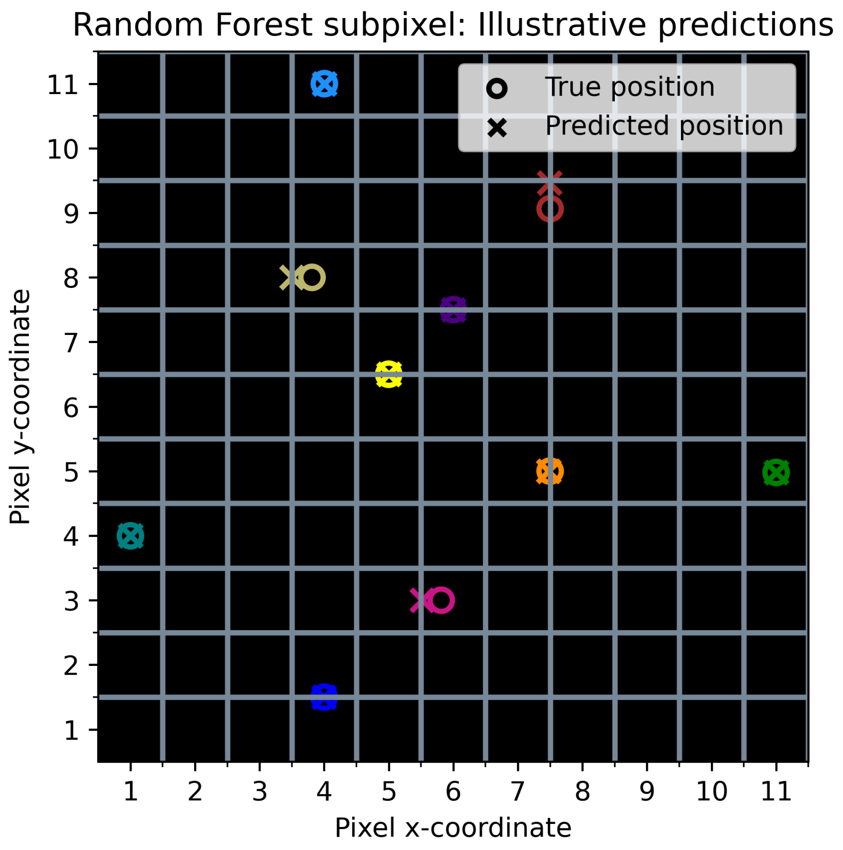


Figure 74: *Visualization of Random Forest predictions: Identical colors correspond to the same sample, with different markers – "X" and "O" – representing the predicted and true positions, respectively. Challenges arise primarily in cases involving shifted COM points, while predictions for pure COM samples remain robust.*

All the experimental tasks have been thoroughly detailed, and the results have been comprehensively discussed. What remains is to address the central question of the thesis.

5 Conclusion

The central question addressed in this work was whether AI-based algorithms can effectively replace and/or improve conventional methods for calculating event properties recorded by high-energy detectors. Firstly, a brief introduction to the fundamentals of semiconductors and AI were presented, followed by a detailed description of the experimental procedure. This began with an overview of the detector from which the high-energy recordings were obtained. The project's objectives and the role of this thesis within the broader scope of the i-RASE initiative were then elucidated. Subsequently, the specific tasks for benchmarking the AI models were defined and evaluated. These tasks included predicting event energy, drift time, and the resulting depth of interaction within the detector's semiconductor, classifying pixel patterns to estimate the type of radiation and interaction occurring within the detector, and determining the subpixel precision of incoming particles at the detector's surface.

Based on the presented results, the conclusion is a cautious yes. The AI models performed exceptionally well across the tasks, demonstrating resilience against noise and uncertainties inherent in experimental data. This success underscores the potential of AI to contribute meaningfully to detector data analysis, especially when coupled with experimental data that incorporates real-world complexities.

Insights of the usage of AI in this project

A significant advantage of utilizing an AI on experimental data lies in the models' ability to capture unforeseen or unknown external influences, offering a realistic foundation for AI training. Synthetic data and simulations often fail to account for all possible noise/uncertainty terms: location-dependent noise signatures, limited accuracy of the detector and its readout systems, and/or time-dependent changes of the properties within the data – e.g., Technetium-99m employed for diagnostic imaging across human organs possessing a half-life of six hours [40], which has to be considered in the detection process. Furthermore, it is difficult to gauge how much and what type of uncertainty to include in each of the events or whether these effects persist across different setups, i.e., different handling of the noise for indoor and outdoor measurements. Experimental data intertwines these uncertainties – drawn from an inherently real-world probability distribution –, making it invaluable for training AIs, which are then able to interconnect and balance all of the aforementioned problems into their predictions. Moreover, a data set consisting over one million real-world events also hints toward the usage of AI data analysis.

However, the usage of AI is only partially justified because the conventional methods, which generated the training and testing labels, often entail implicit biases that AI models inadvertently inherit. For instance, if the traditional functions struggle with rare or exotic samples, AI models trained on them are likely to inherit the same limitations.

Limitations of simulated data

A key concern in detector data analysis is the reliance on simulated data. AI trained solely on simulations risk in learning an idealized version of the underlying physics, divorced from the complex realities of experimental conditions. Such models are prone to failure in real-world applications. A potential remedy is to fine-tune simulation-trained models using experimental data, blending theoretical knowledge with empirical reality. Training exclusively on simulated data is less meaningful, the key benefit of using AI lies in surpassing conventional methods, not merely replicating them.

Understanding the role of labels

In this work, the exact accuracy of the labels was of secondary importance. Labels were generated by customized functions and serve as approximations to the ground truth, but the more captivating result in this work was to observe the AI's ability to grasp the underlying patterns and deriving meaningful predictions. For example, in the subpixel precision task (cf. Chapter 4.7), pixel-dependent variations for computing finer impact positions, although challenging to model explicitly, were reasonably addressed by the AI models: results demonstrated the models' ability to interpret experimental data and produce sound predictions. Having access to the true target labels would be ideal as it would enable the AI to directly compete in real-world applications. However, the demonstrated results represent a crucial and justified first step toward achieving this goal.

For illustration, the prediction of protein structures by the AlphaFold AI were compared to experimentally observed and examined structures archived in a reference database [38]. By imagining a parallel universe where the database contained altered protein structures due to a different environment, physics, lifeforms, etc., but inheriting the same level of complexity in the protein formation pattern, the AlphaFold's ability – understanding the patterns and relationships governing protein folding – would remain effective and would still result in the groundbreaking success in biological chemistry, which is determined through its model composition and AI design and is not directly dictated by the actual label value, rather the complexity they implicitly introduce.

This principle influenced the approach taken in this thesis. Artificial non-linearity was deliberately introduced to a COM calculation for the subpixel task to somewhat reflect the complexity of the problem, since no labels from the manufacturer were attached. However, for energy and drift time computations, which were guided by clear instructions in the detector manual, there was no need for further complications. The radiation and interaction type determination were established using insights from relevant scientific literature and discussions with field experts during i-RASE consortium meetings, ensuring the analysis was grounded in robust scientific understanding.

Future directions and recommendations

To address biases introduced by conventional labeling methods and enhance AI's applicability, future efforts should prioritize experimental setups that provide real and also diverse labels. For instance, subpixel precision and depth of interaction are also needed to determine the location of the radiation source more precisely. Both tasks enhance the detector's spatial resolution. These tasks could benefit from controlled experiments employing well-characterized radioactive sources, such as Cs-137, and by systematically varying the source-to-detector distance or introducing a mesh in front of the detector to alter its response, position labels could be derived directly from experimental conditions, reducing reliance on approximations. Each change of the source's position and mesh type is marked in the data set's time stamps with the corresponding position label.

For energy predictions the task is bound to the detector specific precision, but can be further refined through spectroscopy from additional well-understood sources, beside the Cs-137, akin to the process described in Figure 37.

More challenging tasks, such as radiation type and interaction classification, require careful consideration of background noise and external interference. Experimental setups isolating specific particle types or interactions, informed by literature and expert experience, could add valuable training data for AI models and leverage the result in these tasks.

Application and potential of AI

This study highlights AI's significant potential to bridge gaps in our understanding of detector data. In cases where the initial conditions, such as the anode and cathode signals contain rich information about the properties of the incident radiation of each recorded event, but the final outcomes are not explicitly available, AI can play a pivotal role. While these outcomes can be made observable through carefully designed experiments, as previously discussed, the intermediate processes often remain partially elusive. Here, AI serves as an effective surrogate, addressing complexities that human experts cannot fully decipher due to factors like undefinable noise, unexpected anomalies, or time-dependent artifacts in the data.

This represents one of the most compelling use cases for AI: stepping in where human understanding is limited – whether due to the inherent noise and variability in the data, unforeseen challenges, or the sheer volume of data that cannot be exhaustively analyzed by human effort. By learning from diverse experimental observations, AI models develop insights that can, in some instances, surpass human expertise [2]. This ability to generalize and adapt makes AI a powerful tool for tackling challenges where traditional approaches fall short.

Conclusively, the insights gained from this work represent a pivotal step forward in the i-RASE project, laying the foundation for further AI-driven advancements in detector data analysis. The results highlight AI's ability to decipher complex patterns in experimental data, providing a robust starting point for future investigations and hopefully kindles further advancements in this project.

6 Outlook

The thesis addressed the central question: *Can AI replace conventional methods in detector data analysis?* The answer was affirmative, albeit with minor limitations. An outlook on the i-RASE project's future directions is presented below.

A promising enhancement to the implemented AI algorithms would be the incorporation of uncertainty estimation, a field that provides a nuanced perspective on model predictions by assigning a confidence parameter to each output. This approach offers a significant advantage, as the model's predictions appear more transparent.

For instance, consider an airport security scenario where a Machine Learning model is 60% confident that a bag does not contain any prohibited items. Instead of simply outputting "no danger", the algorithm highlights its uncertainty, signaling that it encountered challenges in classifying the observation. Based on such confidence levels, handlers can take informed action if the model's certainty falls below a defined threshold, such as 90%. This capability offers users greater context and reliability compared to a binary true/false output.

In parallel, the requirements for the team at the University of Tübingen emphasize deploying neural networks on an FPGA, as detailed in Chapter 4.2. The networks tested in this work are well-suited for this purpose and could potentially operate on an FPGA. However, the final network solution will be provided by the Technical University of Denmark. At the time of writing, a key question remains unanswered: should a single network or multiple networks be utilized to predict event properties?

Deploying a single network for solving all the data analysis tasks risks reaching the FPGA's capacity limits. A viable strategy would involve determining the maximum network complexity that can be implemented on an FPGA while maintaining satisfactory performance and then to successively pruning the network's architecture until the optimal trade-off between accuracy and prediction pace is reached. A convolutional neural network (CNN) appears particularly promising, as its use of convolutional layers reduces the number of parameters (cf. Chapter 3.3.3). However, if multiple networks are to be deployed, storage and execution time considerations become critical. Each radiation-feature-specific network would need to be loaded individually, making it essential to avoid overly complex architectures that could overwhelm the AI-detector setup.

The i-RASE project is still in its early stages, yet initial results indicate great potential and a promising outlook. Notably, the timing coincides with a significant milestone in AI: the awarding of the Nobel Prize to Deep Learning pioneers John Hopfield and Geoffrey Hinton for their foundational contributions to neural network theory, which emulate human brain functionality. Geoffrey Hinton remarked, "Deep Learning is going to be able to do everything" [2], and with such optimism, I look forward to the future with great anticipation.

Acknowledgments

I would like to take this opportunity to express my sincere gratitude to everyone who supported me in the completion of this thesis.

First and foremost, I wish to thank my supervisor, Prof. Dott. Andrea Santangelo, for welcoming me into the High Energy Astrophysics group and for agreeing to evaluate my thesis.

I am also very grateful to Dr. Chris Tenzer for involving me in this project. I appreciate his unwavering support in addressing my questions and challenges, as well as his guidance in fostering a scientific approach to problem solving. His valuable and constructive feedback significantly enhanced the quality of my work. I am particularly thankful for the opportunity to participate in numerous international conferences under his mentorship, which profoundly shaped my personal and professional growth, leaving me with many enriching experiences.

Additionally, I also extend my profound thanks to our esteemed project partners at IDEAS for their provision of the measurement data set from a Cs-137 source on a CdZnTe detector, along with comprehensive documentation and classical analysis scripts in Python, which was instrumental to our progress. Their exceptional willingness to address our inquiries and provide support exceeded the customary expectations within the scope of such a project framework. I am particularly indebted to Sofia Godø, Tor Magnus Johansen, and Dirk Meier, whose efforts were pivotal in enabling the AI-driven analysis presented in this study.

I am particularly appreciative to the European Innovation Council (EIC) for their generous funding of the i-RASE project. As Europe's flagship innovation program, the EIC is dedicated to foster groundbreaking advancements by supporting only the most innovative proposals. We are deeply honored to have been selected by such a prestigious program.

I am equally grateful to the Prüfungsamt Mathematik und Physik, particularly Sandra Grahl, and to Sabine Lauer from the institute's administration, for their assistance with all administrative matters related to this thesis.

Lastly, I wish to express my heartfelt gratitude to my parents, whose unconditional support made both this thesis and my studies possible.

This master's thesis was conducted in the Astro & Particle Physics program at the Eberhard Karls Universität Tübingen, starting on April 22, 2024, and submitted on January 14, 2025.

Christopher Weinert
Tübingen, January 10, 2025

References

[1] AlmaBetter.com. *Random Forest Algorithm in Machine Learning.* Accessed: 09.12.2024. 2023. URL: https://www.almabetter.com/bytes/tutorials/data-science/random-forest.

[2] Laith Alzubaidi et al. "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions". In: *Journal of Big Data* 8 (Mar. 2021).

[3] Samy Baladram. *Decision Tree Regressor, Explained: A Visual Guide with Code Examples.* Towards Data Science, accessed: 29.11.2024. 2024. URL: https://towardsdatascience.com/decision-tree-regressor-explained-a-visual-guide-with-code-examples-fbd2836c3bef.

[4] Cosimo Bambi and Andrea Santangelo. *Handbook of X-ray and Gamma-ray Astrophysics.* 1st ed. Springer Singapore, 2024. ISBN: 9789811969591.

[5] Leo Breiman. "Random Forests". In: *Machine Learning* 45 (Oct. 2001), pp. 5–32. URL: https://doi.org/10.1023/A:1010933404324.

[6] Carla Brodley and Paul Utgoff. "Multivariate Decision Trees". In: *Kluwer Academic Publishers, Machine Learning* 19 (1995), pp. 45–77. URL: https://link.springer.com/content/pdf/10.1023/A:1022607123649.pdf.

[7] CAEN Educational. *Study of the 137Cs spectrum: the backscatter peak and X rays.* Accessed: 28.10.2024. 2012. URL: https://edu.caen.it/experiments/nuclear-physics/study-of-the-137cs-spectrum-the-backscatter-peak-and-x-rays/#:~:text=The%20result%20is%20an%20excess,of%20an%20X%2Dray)..

[8] Bradley W. Carroll and Dale A. Ostlie. *An Introduction to Modern Astrophysics.* 2nd ed. Edinburgh Gate, Harlow, Essex CM20 2JE: Pearson Education Limited, 2014. ISBN: 978-1-292-02293-2.

[9] CERN. *Cosmic rays: particles from outer space.* Accessed: 01.11.2024. 2024. URL: https://www.home.cern/science/physics/cosmic-rays-particles-outer-space.

[10] Nitesh V. Chawla et al. "SMOTE: Synthetic Minority Over-sampling Technique". In: *Journal of Artificial Intelligence Research* 16 (2002), pp. 321–357. URL: https://arxiv.org/pdf/1106.1813.

[11] Tianqi Chen and Carlos Guestrin. "XGBoost: A Scalable Tree Boosting System". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2016). https://doi.org/10.1145/2939672.2939785, pp. 785–794.

[12] ColinStapczynski. *The 10 Best Chess Moves Of All Time: Stefan Levitsky vs. Frank James Marshall.* chess.com, Accessed: 10.11.2024. 2022. URL: https://www.chess.com/article/view/best-chess-moves#shirov.

[13] Aymeric Damien. *Convolutional Neural Network Example.* Accessed: 09.11.2024. 2019. URL: https://github.com/aymericdamien/TensorFlow-Examples/blob/master/tensorflow_v2/notebooks/3_NeuralNetworks/convolutional_network.ipynb.

[14] DataMListic. *Gradient Boosting with Regression Trees Explained.* Accessed: 19.11.2024. 2023. URL: https://www.youtube.com/watch?v=lOwsMpdjxog.

[15] Dharmaraj. *Convolutional Neural Networks (CNN) — Architecture Explained.* Accessed: 09.11.2024. 2022. URL: https://medium.com/@draj0718/convolutional-neural-networks-cnn-architectures-explained-716fb197b243.

[16] DMLC XGBoost. *XGBoost Documentation.* Accessed: 20.11.2024. URL: https://xgboost.readthedocs.io/en/stable/index.html.

[17] Diane D. Duarte. "Edge effects in a pixelated CdTe radiation detector". PhD thesis. University of Surrey, 2016. URL: https://openresearch.surrey.ac.uk/esploro/outputs/doctoral/Edge-effects-in-a-pixelated-CdTe/99514506902346/filesAndLinks?index=0.

[18] Mario Filho. *Do Decision Trees Need Feature Scaling Or Normalization?* Forecastegy; Accessed: 07.11.2024. 2023. URL: https://forecastegy.com/posts/do-decision-trees-need-feature-scaling-or-normalization/.

[19] Fraunhofer Institute for Integrated Circuits IIS. *Development of an advanced interference detection and robustness capabilities system for GNSS signals (DARCY).* Accessed: 29.11.2024. 2020. URL: https://www.iis.fraunhofer.de/en/ff/lv/lok/proj/darcy.html.

[20] Jerome H. Friedman. "Greedy function approximation: A gradient boosting machine". In: *The Annals of Statistics* 29.5 (Oct. 2001), pp. 1189–1232. URL: https://doi.org/10.1214/aos/1013203451.

[21] Gaurav Singhal. *Introduction to LSTM Units in RNN.* Accessed: 01.12.2024. 2020. URL: https://www.pluralsight.com/resources/blog/guides/introduction-to-lstm-units-in-rnn.

[22] Gini Index: Decision Tree, Formula, Calculator, Gini Coefficient in Machine Learning. *Chainika Thakar and Shagufta Tahsildar.* Accessed: 15.11.2024. 2022. URL: https://blog.quantinsti.com/gini-index/.

[23] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning.* http://www.deeplearningbook.org. MIT Press, 2016.

[24] Google Cloud. *Künstliche Intelligenz (KI) und maschinelles Lernen (ML).* Accessed: 12th August 2024. URL: https://cloud.google.com/learn/artificial-intelligence-vsmachine-learning#differences-between-ai-and-ml.

[25] Zhong He. "Review of the Shockley–Ramo theorem and its application in semiconductor gamma-ray detectors". In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 1 (2001). An optional note, pp. 250–267. URL: https://www.sciencedirect.com/science/article/pii/S0168900201002236.

[26] Paul Hiemstra. *Strength in Numbers: Why Does a Gradient Boosting Machine Work So Well?* Medium, accessed: 20.11.2024. 2022. URL: https://towardsdatascience.com/strength-in-numbers-why-does-a-gradient-boosting-machine-work-so-well-8450d3b114c0.

[27] Tin Kam Ho. "Random decision forests". In: *Proceedings of 3rd International Conference on Document Analysis and Recognition.* 1 (1995), pp. 278–282. DOI: 10.1109/ICDAR.1995.598994..

[28] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (1997), pp. 1735–1780.

[29] JaeSub Hong et al. "Cathode depth sensing in CZT detectors". In: *SPIE* 5165 (2004). URL: http://dx.doi.org/10.1117/12.506216.

[30] JJ. Hopfield. "Neural networks and physical systems with emergent collective computational abilities". In: *The Proceedings of the National Academy of Sciences* 79.8 (Apr. 1982). https://doi.org/10.1073/pnas.79.8.2554.

[31] IBM. *What is a decision tree?* Accessed: 15.11.2024. URL: https://www.ibm.com/topics/decision-trees.

[32] IDEAS Integrated Detector Electronics AS. *GDS-100*. Accessed: 04.11.2024. URL: https://ideas.no/products/gds-100/.

[33] Indico, Berkeley Lab. *Ramo-Shockley Theorem*. Accessed: 22.12.2024. 2015. URL: https://indico.physics.lbl.gov/event/285/contributions/691/attachments/614/677/studentMeeting_Ramo_6_3_16.pdf.

[34] Intel. *The AI PC from Intel*. Accessed: 29.08.2024. URL: https://www.intel.com/content/www/us/en/products/docs%5Cprocessors/core-ultra/ai-pc.html#:~:text=AI%20PCs%20use%20artificial%20intelligence,tasks%20locally%20and%20more%20efficiently..

[35] Jason Brownlee. *A Gentle Introduction to Imbalanced Classification*. Accessed: 02.12.2024. 2020. URL: https://machinelearningmastery.com/what-is-imbalanced-classification/.

[36] Jason Brownlee. *How to use Data Scaling Improve Deep Learning Model Stability and Performance*. Accessed: 02.12.2024. 2020. URL: https://www.nb-data.com/p/is-it-necessary-for-feature-scaling.

[37] Vaidehi Joshi. *Breaking Down Breadth-First Search*. Medium; Accessed: 07.11.2024. 2017. URL: https://medium.com/basecs/breaking-down-breadth-first-search-cebe696709d9.

[38] John Jumper and Demis Hassabis. "Highly accurate protein structure prediction with AlphaFold". In: *Nature* 596 (Aug. 2021), pp. 583–589. URL: https://doi.org/10.1038/s41586-021-03819-2.

[39] I. Jung et al. "Detailed studies of pixelated CZT detectors grown with the modified horizontal Bridgman method". In: *Elsevier, Astroparticle Physics* 28 (Dec. 2007), pp. 397–408. URL: https://doi.org/10.1016/j.astropartphys.2007.08.004.

[40] Steven M. Kane et al. "Technetium-99m". In: *National Library of Medicine* (Feb. 2024). URL: https://www.ncbi.nlm.nih.gov/books/NBK559013/.

[41] Samantha Murphy Kelly. *ChatGPT passes exams from law and business schools*. CNN Business, accessed: 23.11.2024. 2023. URL: https://edition.cnn.com/2023/01/26/tech/chatgpt-passes-exams/index.html.

[42] Nils Kimmel et al. "The direct measurement of the signal charge behavior in pnCCDs with subpixel resolution". In: *Elsevier, Nuclear Intruments and Methods in Physics Research* 568 (2006), pp. 128–133. URL: https://doi.org/10.1016/j.nima.2006.07.017.

[43] Kromek. *What is CZT?* Accessed: 02.12.2024. 2024. URL: https://www.kromek.com/czt/#:~:text=When%20ionizing%20radiation%20interacts%20with,electrodes%20where%20they%20are%20collected..

[44] Irfan Kuvvetli et al. *i-RASE: Intelligent Radiation Sensor Readout Systems.* Technical University of Denmark, accessed: 23.11.2024. 2024. URL: https://orbit.dtu.dk/en/projects/i-rase-intelligent-radiation-sensor-readout-systems.

[45] Geir Kvam-Langelandsvik and Hans Roven. "Optimization of Electrical Conductivity in Screw Extruded Wires". PhD thesis. June 2017.

[46] Laboratoire National Henri Becquerel. *Interactions of photons with matter and shape of spectra in X/γ-ray spectrometry.* Accessed: 01.11.2024. URL: http://www.lnhb.fr/pdf/Photon_matter_interaction.pdf.

[47] Zhangwen Li et al. "Research on the Technological Progress of CZT Array Detectors". In: *Sensors* 24 (2024). URL: https://doi.org/10.3390/s24030725.

[48] Shana Lynch. *The State of AI in 9 Charts.* 29.08.2024. 2022. URL: https://hai.stanford.edu/news/state-ai-9-charts.

[49] MathWorks. *Long Short-Term Memory (LSTM).* Accessed: 25.11.2024. 2024. URL: https://de.mathworks.com/discovery/lstm.html.

[50] MAXIMUS.ENERGY. *The Rich Physics of Cs-137 Gamma Spectrum.* Accessed: 28.10.2024. 2020. URL: https://maximus.energy/index.php/2020/10/24/the-rich-physics-of-cs-137-gamma-spectrum/.

[51] Monolithic Power Systems. *Key Parameters Of ADCs.* Accessed: 27.10.2024. 2024. URL: https://www.monolithicpower.com/en/learning/mpscholar/analog-to-digital-converters/introduction-to-adcs/key-parameters-of-adcs?srsltid=AfmBOop17FXyH2sB0zdkEQUCOp6obrfyiCQGQiXzDmdj3dtpNsPFGnHm.

[52] Nick Troccoli. *CS 106X, Lecture 22: Graphs; BFS; DFS.* Accessed: 02.12.2024. URL: https://web.stanford.edu/class/archive/cs/cs106x/cs106x.1192/lectures/Lecture22/Lecture22.pdf.

[53] Michael A. Nielsen. *Neural Networks and Deep Learning.* Determination Press, 2015. URL: http://neuralnetworksanddeeplearning.com/index.html.

[54] Nils Nilsson. *Introduction to Machine Learning.* Stanford, CA 94305: Department of Computer Science, Stanford University, Nov. 1998. URL: https://ai.stanford.edu/~nilsson/mlbook.html.

[55] NVIDIA. *What is XGBoost?* Accessed: 20.11.2024. 2024. URL: https://www.nvidia.com/en-us/glossary/xgboost/.

[56] Keiron O'Shea and Ryan Nash. "An Introduction to Convolutional Neural Networks". In: *The name of the journal* (Dec. 2015). URL: https://doi.org/10.48550/arXiv.1511.08458.

[57] S. Howalt Owe et al. "Evaluation of a Compton camera concept using the 3D CdZnTe drift strip detectors". In: *Journal of Instrumentation* 14.1 (Jan. 2019). URL: https://dx.doi.org/10.1088/1748-0221/14/01/C01020.

[58] Beomjun Park et al. "Bandgap engineering of $\mathrm{Cd}_{1-x}\mathrm{Zn}_x\mathrm{Te}_{1-y}\mathrm{Se}_y$ ($0 < x < 0.27, 0 < y < 0.026$)". In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 1036 (2022). URL: https://doi.org/10.1016/j.nima.2022.166836.

[59] Matthew Petryk. "Algorithms and Electronics for Processing Data from Pixelated Semiconductor Gamma-Ray Detectors". PhD thesis. University of Michigan, 2023. URL: `https://cztlab.engin.umich.edu/wp-content/uploads/sites/187/2023/01/thesis_mpetryk.pdf`.

[60] Simon J.D. Prince. *Understanding Deep Learning.* The MIT Press, May 2024. ISBN: 978-0262048644.

[61] Purwono et al. "Understanding of Convolutional Neural Network (CNN): A Review". In: *International Journal of Robotics and Control Systems* 2.4 (2022). URL: `doi:10.1088/1757-899X/490/4/042022`.

[62] Partha Pratim Ray. "ChatGPT: A comprehensive review on background, applications, key challenges, bias, ethics, limitations and future scope". In: *Internet of Things and Cyber-Physical Systems* 3 (2023), pp. 121–154. URL: `https://doi.org/10.1016/j.iotcps.2023.04.003`.

[63] Lior Rokach and Oded Maimon. "Decision Trees". In: *The Data Mining and Knowledge Discovery Handbook* 6 (Jan. 2005), pp. 165–192. URL: `https://link.springer.com/chapter/10.1007/0-387-25465-X_9`.

[64] Samsung. *How to use Generative photo editing with Galaxy AI.* Accessed: 23.11.2024. 2024. URL: `https://www.samsung.com/latin_en/support/mobile-devices/how-to-use-generative-photo-editing-on-the-galaxy-s24/`.

[65] Estefania Sanchez-Rodriguez et al. "Changes with age (from 0 to 37 D) in tibiae bone mineralization, chemical composition and structural organization in broiler chickens". In: *Poultry Science* 98 (July 2019). DOI: `10.3382/ps/pez363`. URL: `https://www.researchgate.net/figure/Body-weight-and-bone-properties-as-a-function-of-chicken-age-A-Body-weight-B-tibia_fig1_334187731`.

[66] Robin M. Schmidt. "Recurrent Neural Networks (RNNs): A gentle Introduction and Overview". In: *arXiv* (Nov. 2019). Department of Computer Science, Eberhard-Karls-University Tübingen. URL: `https://arxiv.org/pdf/1912.05911`.

[67] scikit-learn. *Histogram-Based Gradient Boosting.* Accessed: 20.11.2024. URL: `https://scikit-learn.org/dev/modules/ensemble.html#histogram-based-gradient-boosting`.

[68] scikit-learn. *The Iris Dataset.* Accessed: 01.12.2024. 2024. URL: `https://scikit-learn.org/1.5/auto_examples/datasets/plot_iris_dataset.html`.

[69] Ravid Shwartz-Ziv and Amitai Armon. "Tabular Data: Deep Learning is Not All You Need". In: *arXiv* (2021). URL: `https://arxiv.org/abs/2106.03253`.

[70] TensorFlow. *Module:tf.keras.callbacks.* Accessed: 01.12.2024. 2024. URL: `https://www.tensorflow.org/api_docs/python/tf/keras/callbacks`.

[71] TensorFlow. *Tensorflow 2 quickstart for experts.* Accessed: 09.11.2024. 2024. URL: `https://www.tensorflow.org/tutorials/quickstart/advanced`.

[72] TensorFlow. *Time series forecasting.* Accessed: 09.11.2024. 2024. URL: `https://www.tensorflow.org/tutorials/structured_data/time_series#convolution_neural_network`.

[73] The University of Liverpool, Department of Physics. *Backscattered Peaks.* Accessed: 28.10.2024. 2000. URL: `https://ns.ph.liv.ac.uk/~ajb/radiometrics/gamma_radiation/interactions_surroundings/backscattered_peaks.html`.

[74] thispersonnotexist.org. *Generate Random Faces With AI, Random Face.* Accessed: 29.11.2024. 2024. URL: https://thispersonnotexist.org/.

[75] David J. Thompson and Alexander A. Moiseev. "Pair Production Detectors for Gamma-ray Astrophysics". In: *Springer Singapore* (Oct. 2024). Accessed: 01.11.2024. URL: https://ntrs.nasa.gov/api/citations/20220000717/downloads/Pair_Production_Chapter.pdf.

[76] Mark Thomson. *Modern Particle Physics.* University Printing House, Cambridge CB2 8BS, United Kingdom: Cambridge University Press, 2013. ISBN: 978-1-107-03426-6.

[77] ubiAI. *The Secret to Mastering Feature Extraction in Convolutional Neural Network.* Accessed: 01.12.2024. 2023. URL: https://ubiai.tools/the-secret-to-mastering-feature-extraction-in-convolutional-neural-network/.

[78] Christopher Weinert et al. "Effective encapsulation of noble and small gases and molecules by tubularene molecule". In: *Journal of Inorganic and General Chemistry* 650.3 (Dec. 2023).

[79] Christopher Weinert et al. "Selective Noble Gas Inclusion in Pentagon-Dodecahedral X_{20}-Cages". In: *Molecules* 28.15 (July 2023).

[80] Nicola Winch, Scott Watson, and James Hunter. "Modeling blur in various detector geometries for MeV radiography". In: *SPIE Medical Imaging* (Mar. 2017). URL: https://www.researchgate.net/figure/Dominant-physical-processes-of-photon-interaction-in-matter-as-a-function-of-photon_fig1_314711714.

[81] xgboosting.com. *XGBoosting Regularization Techniques.* Accessed: 20.11.2024. 2024. URL: https://xgboosting.com/xgboost-regularization-techniques/.

[82] Rikiya Yamashita et al. "Convolutional neural networks: an overview and application in radiology". In: *Springer with Insights Imaging* (2018), pp. 611–629. URL: https://insightsimaging.springeropen.com/articles/10.1007/s13244-018-0639-9.

[83] Jian Yang et al. "Direct measurement of electron drift parameters in pixelated cadmium zinc telluride semiconductor detectors". In: *Elsevier, Measurement* 220 (2023). URL: https://doi.org/10.1016/j.measurement.2023.113128.

[84] Nadezhda Yarushkina and Vadim Moshkin. "Application of Machine Learning Algorithms to Determine the Authorship of Text Fragments". In: *Advances in Intelligent Systems and Computing.* Springer Nature Switzerland AG, 2021, pp. 260–267.

[85] Liang Yu, Binbin Li, and Bin Jiao. "Research and Implementation of CNN Based on TensorFlow". In: *IOP Conf. Series: Materials Science and Engineering* 490 (2019). URL: https://iopscience.iop.org/article/10.1088/1757-899X/490/4/042022.

[86] H. Yücel, Ö. Birgül, and Ş. Çubukçu. "A novel approach in voltage transient technique for the measurement of electron mobility and mobility-lifetime product in CdZnTe detectors". In: *Elsevier, Nuclear Engineering and Technology* 51 (2019), pp. 731–737. URL: https://doi.org/10.1016/j.net.2018.12.024.

[87] Wei Zhang et al. "Loss Landscape Dependent Self-Adjusting Learning Rates in Decentralized Stochastic Gradient Descent". In: *arXiv* (2021). URL: https://arxiv.org/abs/2112.01433.