SAPIENZA
UNIVERSITÀ DI ROMA

A convergent and explainable machine learning framework for pattern recognition in the ATLAS experiment

Faculty of Information Engineering, Informatics, and Statistics
Master Course on Artificial Intelligence and Robotics

Leandro Maglianella
ID number 1792507

Advisor
Prof. Christian Napoli

Co-Advisors
Prof. Stefano Giagu
Prof. Simone Scardapane

Academic Year 2021/2022

Thesis defended on 20 January 2023
in front of a Board of Examiners composed by:

Prof. Leonardo Querzoni (chairman)
Prof. Irene Amerini
Prof. Emiliano Casalicchio
Prof. Antonio Cianfrani
Prof. Francesco Leotta
Prof. Christian Napoli
Prof. Simone Scardapane

A convergent and explainable machine learning framework for pattern recognition in the ATLAS experiment
Sapienza University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Author's email: maglianella.1792507@studenti.uniroma1.it

Dedicated to my loved ones

Abstract

Most modern Machine Learning (ML) approaches are commonly defined as "black-box" models. This means that the process they follow to perform decisions is neither immediately clear nor at all understandable by humans. Recently a new field of research came about to address these issues: it is Explainable Artificial Intelligence (xAI), whose goal is to implement methods capable of giving explanations for ML algorithms' predictions in such a way as to make their outcomes transparent and reliable for users. At the same time, it is often the case that distinct xAI methods may provide explanations that differ both from a quantitative and a qualitative standpoint. Moreover, this large variety makes it difficult even for domain experts to efficiently select and interpret their results. This thesis will consider and study these ML and xAI limitations employing a High-Energy Physics (HEP) use-case about pattern recognition of muonic motions in the ATLAS experiment. Five xAI methods based on three different approaches will be explored and tested in our scenario introducing some ad hoc metrics: their strengths and drawbacks will be considered and I will carry out a discussion about the convergent use of xAI algorithms in a real-world environment. The achievements of this work lie in the development of techniques suited for producing, from a model's decision, an array of variegate explanations that can be combined to reach an easily understandable result by both experts and beginners.

Acknowledgments

Ringrazio Lorenzo, che per merito del suo indispensabile lavoro ha permesso l'effettiva riuscita di questo e tanti altri progetti. Questo percorso accedemico sarebbe stato completamente diverso se non ci fossi stato e spero di continuare la nostra collaborazione anche nei futuri tempi a venire.

Ringrazio anche i supervisori di questa tesi, che con la loro esperienza e i loro consigli hanno guidato la sua ideazione e sviluppo: il Prof. Christian Napoli, il Prof. Stefano Giagu e il Prof. Simone Scardapane.

Ringrazio la mia famiglia, mia madre, mio padre e mio fratello, per la vostra presenza e il sostegno che mi avete sempre dimostrato nel corso della mia vita e durante questo percorso di studi.

Ringrazio tutti i miei amici, che mi hanno accompagnato in questo viaggio e mi hanno dato la forza di andare avanti anche nei momenti più difficili.

Voglio esprimere la mia gratitudine anche verso tutti coloro che, direttamente o indirettamente, hanno contribuito alla realizzazione di questo lavoro.

Infine, a Cecilia, grazie dell'Amore, il supporto e la complicità che ogni giorno mi doni. Grazie per credere in me.

Grazie di cuore a tutti voi,
Leandro.

Contents

1 Introduction **1**

2 Explainable Artificial Intelligence **5**

2.1 Method types partition 6

2.2 Explanation types partition 8

2.3 Knowledge Distillation 9

3 Related xAI techniques **11**

3.1 Saliency Maps Methods 11

3.1.1 Class and Regression Activation Map 12

3.1.2 Integrated Gradients 14

3.1.3 SmoothGrad 15

3.2 Intrinsically-interpretable models 17

3.2.1 Soft Decision Trees 19

3.3 Tracing Gradient Descent for data influence estimation 22

4 Foundations of High-Energy Physics **25**

4.1 The Standard Model 26

4.2 The European Council for Nuclear Research 29

4.2.1 The ATLAS detector 30

4.2.2 ATLAS's Trigger and Data Acquisition systems 32

5 Materials and tasks **35**

5.1 Main and only-noise datasets 35

5.2 Task metrics 40

6 Implementation **43**

6.1 Models 43

6.1.1 Convolutional Neural Networks 44

6.1.2 Convolutional Soft Decision Tree 45

6.2 xAI techniques adaptations 48

7 Results and Discussion **53**

7.1 Hardware 53

7.2 Regression stage 54

7.2.1 Regression main task 56

7.2.2 Regression sub-task 58

7.2.3 Regression alternative task 61

7.2.4 Regression of only-noise dataset 62

7.3 Explainability stage 64

7.3.1 Saliency Maps Methods 65

7.3.2 Intrinsically-interpretable models 75

7.3.3 Tracing Gradient Descent 84

8 Conclusions **93**

Bibliography **97**

Chapter 1

Introduction

The passage of the years and the scientific and technological advancements have made it very clear that Artificial Intelligence (AI) is quickly becoming more and more ubiquitous in any type of economic sector of our modern societies. As an example, we could consider the recent tremendous growth in global investment in AI: if in 2017 it amounted to about 10 billion US dollars, in 2021 this value has increased to about 80 billion US dollars (surpassing the estimated value, which was only 50 billion) and it is expected to skyrocket even more in the near future [1]. It is in fact true that AI is now widely and extensively used even in the so-called *critical domains* where important decisions are taken that could influence our everyday society. For instance, some of the areas tied to this definition are advertisement [36], transportation [25], healthcare, military [6, 56], finance and legal applications [1]. In contexts such as the ones just mentioned the used Machine Learning methods must be transparent and they must demonstrate to be reliable, meaning that it is required to give humans some way to understand the procedure that led to its outputs.

Unfortunately, however, too often this does not happen and we rely on unexplainable models to just get the highest possible performances neglecting their explainability: the interpretability and the accuracy of a model are usually inversely proportional. For instance, Deep Neural Networks (DNNs) usually outperform other types of ML methods concerning the most generally accepted evaluation metrics

(accuracy, precision, recall, etc.), but they hugely lack the interpretability aspects. Moreover, sometimes NNs can be very delicate architectures whose predictions can be deeply corrupted even by the most insignificant variation [49]. In contrast, intrinsically-explainable architectures often guarantee a clear comprehension of their decision-making routine but they are also usually unreliable because of their poor performances in the related task and therefore not applicable in the real world [63].

The desiderata expressed until now are managed to be solved by the application of **Explainable Artificial Intelligence** (xAI) approaches that can merge high performances and highly explainable capabilities to understand decisions made by ML algorithms [34]. Nevertheless, it is usually hard to produce explanations that fully satisfy our needs because of their subjective nature and, moreover, the currently available xAI tools are so heterogeneous that could confuse users even more. For this reason in this thesis, instead of limiting ourselves to one single xAI method, we built a set of techniques addressable to a varied range of end-users, including both experts with prior knowledge and beginners with no preparation in the study field: doing so, we introduce the concept of xAI *convergent* approaches, working together to building a more comprehensive explanation.

One of the main topics of xAI is **Knowledge Distillation** (KD) [30], which implements ways of creating a simplified predictive model from another one, usually more complex, used as a "teacher" from which knowledge can be extracted. The derived "student" model should be more compact and lighter but at the same time should not register a relevant decrease in performances with respect to its teacher. Lots of effective techniques for KD have been developed over the years, in this work, the focus is on how to transfer knowledge from a high-efficiency but non-interpretable DNN teacher to an explainable but also simplified model without loss of performance.

The studied domain concerns a **High-Energy Physics** (HEP) application to analyze specific parameters of **muons**, unstable subatomic particles radiated continuously on Earth's surface from the cosmos [3]. The ATLAS experiment of the Large Hadron Collider (LHC) at the European Organization for Nuclear Research,

or CERN, in Geneva [22] is particularly relevant in this study. There, muons are explored using accelerators and detectors capable to log their trajectories and record their associated characteristics, which are stored for possible following studies only if they pass some associated tests: for this reason, efficiently estimating particle parameters while being able to reject muonic events caused by noisy observations is critical. This, together with simultaneously providing understandable explanations for models' predictions, are the objective of this thesis.

The researches reported in this document come from a group work involving three different departments of Sapienza University: the Department of Computer, Control and Management Engineering (DIAG), the Department of Information Engineering, Electronics and Telecommunications (DIET), and the Department of Physics. Our collaboration generated five xAI methods based on three different approaches: I have worked on all of them and, to give a full view of our work, I will discuss all of them, anyway, it should be noted that this thesis officially focuses on the two methods of the last two approaches. Moreover, this research is related to the European project *"Multi-disciplinary Use Cases for Convergent new Approaches to AI explainability"* (MUCCA) on xAI in the CHIST-ERA context [20]: this project brings together researchers with heterogeneous backgrounds to implement and test ML explainability techniques relevant in the scientific world.

The thesis has the following structure: Chapter 2 introduces the concepts of xAI to understand the notions later expressed in Chapter 3, where the related techniques from literature used in this work are explained. Chapter 4 gives some basic knowledge on HEP and technical details about the detector where the data used in this research were produced. Chapter 5 goes deeper into the used datasets and describes the proposed tasks and the adopted evaluation metrics. Chapter 6 concerns the implementation of models, their experimental settings, and explainability techniques. Consequently, in Chapter 7 a complete description of experiments and results is provided, both regarding the regression and explainability stage. Finally, Chapter 8 summarizes the achievements of the project and proposes future works based on it.

Chapter 2

Explainable Artificial Intelligence

Explainable Artificial Intelligence is the recently rising branch of ML which tries to fulfill the needs of implementation transparency for critical application domains: nevertheless, its origins date back to 1970-1980 [43] and the term was first coined in 2004 [58]. No agreement has been yet reached on a formal definition for xAI, however, we can state that the general goal in this field of study consists in finding and developing techniques that can merge high performances and highly explainable capabilities for understanding the decisions made by ML algorithms. Therefore, a real xAI approach must contain some sort of system that allows humans to study and understand how an input is mathematically mapped to an output in the ML model.

Talking about xAI, a different topic must also be pointed out: in fact, xAI is not just implemented to support humans' understanding of the machine, but can also be seen as a way to solve a more socio-ethical issue regarding the responsibility and accountability of an autonomous rational agent for the outcomes it produces and suggests. In fact, for artificial intelligence to become "general" and to be eventually considered as an evolved entity, in the scenario of serious damages generated by its application in some critical domain, it must possess a transparent way with which

some party involved can be established guilty. Moreover, xAI is important not only to ensure human control of AI but also to comprehend more deeply the reasoning process modeled by a machine, enabling further improvements and discoveries.

Summarizing, xAI's primary goal is the formulation of ML predictive models that to the maximum extent possible can be described as *understandable*, *comprehensible*, *transparent*, *interpretable*, *intelligible*, *responsible*, *accountable*, *accurate* and *interactive* [1].

There exists a lot of explainability methods, differing from one another in a large variety of design choices: for this reason, a lot of different possible categorization ways to partition them have been proposed in literature [1, 23, 24, 50, 57, 59, 63]. Anyway, being xAI still an emerging field, it should be remarked that these classifications could neither be exclusive nor exhaustive. The following taxonomy attempt at giving a clear and exhaustive report of the most accepted categorization approaches, to introduce the explainability techniques later used in this thesis.

2.1 Method types partition

This first partitioning approach is done according to three considered criteria: *Interpretability Complexity*, *Interpretability Scoop* and *Model Dependency Level* [1].

- ***Interpretability Complexity*** describes how much the interpretation of a specific kind of model is difficult for humans. A model can be specifically designed to be explainable in an easy way, in this case, it is said to be *Intrinsic Interpretable* (for instance, Decision Trees fall into this class). On the contrary, models based on very complex architectures (for instance, NNs belong to this class) to give a predictive outcome are usually not rapidly comprehensible: thus they belong to the *Post-hoc* class, meaning that they cannot be directly explained but instead need some further processing to provide information about their choices. Moreover, it is easily understandable that models belonging to the second class usually result in higher performances at the expense of

explainability.

- ***Interpretability Scoop*** instead is about the degree of understanding given by a model's explanation. If it is capable of helping us directly comprehend the behavior of the entire model it is said to be a *Global* method. Instead, if it makes us only find out the reasons behind a single prediction given at a time it is defined as *Local*.

- ***Model Dependency Level***, finally, is about the extent of applicability of a specific explainability technique to different predictive models. An explanation method is *Model-specific* if it can only be applied to one kind of model class (or to a restricted specific set of classes). Clearly, *Intrinsic Interpretable* models belong by definition also to the *Model-specific* class. Alternatively, if an explanation method is not bound to some ML model and can be used indifferently every time, it is defined *Model-agnostic*.

Figure 2.1 summarizes method types partition approach.

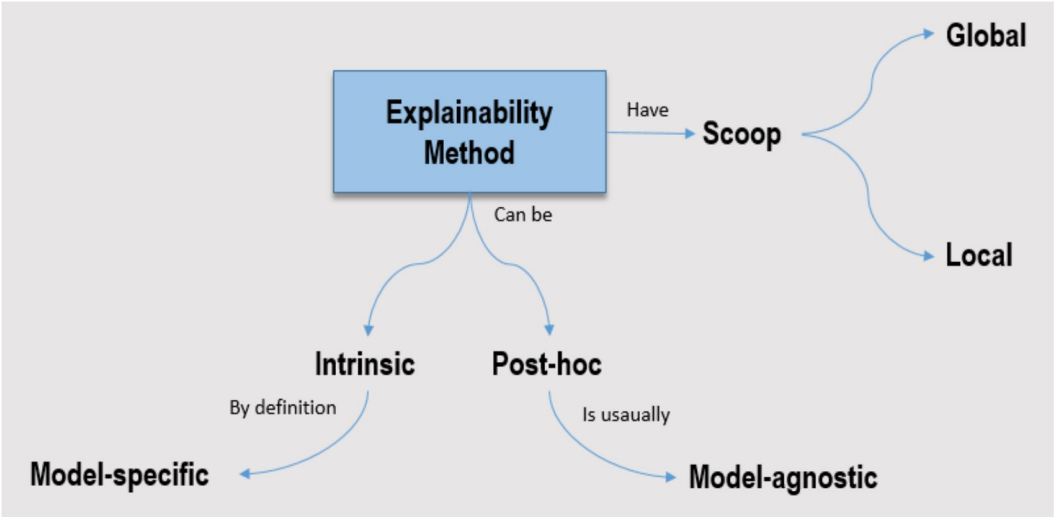


Figure 2.1. Explanability method partitions.

2.2 Explanation types partition

While the first partition considers more general attributes of the xAI methods, this second taxonomy is conducted based on the nature of the generated explanations. This approach includes a huge amount of classes, here only the four main ones will be described: *Features Attribution*, *Learned Features*, *Explanation by Examples* and *Counterfactuals*.

- ***Features Attribution*** methods try to give explanations by understanding which are the most relevant features in a certain given input sample for the prediction yielded by a ML architecture. In general, these methods work by computing an importance score for all the features of the sample to then emphasize the highest-scoring features. This is the most popular and investigated category: a very large number of its approaches are being researched and are part of the state-of-the-art (SOTA), for instance, *LIME* [51], *Kernel SHAP* [40] and *CAM* [64].

- ***Learned Features*** methods try to give explanations by extracting what are the characteristics of the samples of a dataset that a ML model has learned during its training process. In particular, for Neural Network architectures, it is usually investigated what are the input data point that can maximize the response/activation of the network's inner neurons. One of the main papers in this area [65] helped to understand more on a high level how CNNs work internally claiming that the first layers of these architectures learn simple figurative concepts while the last layers have the potential of merging them to form a more complex shape that can retrieve the learned figures. Moreover, they state that a model trained for a scene classification task would also be capable of recognizing and detecting features on several levels of abstraction (for instance, edges, textures, objects, and scenes) by extracting information directly from the intermediate layers and not only from the final one.

- ***Explanation by Examples*** methods try to give explanations in a more

"factual" manner. Furthermore, these approaches try to make the models' behavior transparent by giving, together with the prediction for an input, a set of learned samples from the training dataset that the model believes to be similar to the input. Often, this procedure is performed by implementing and slightly modifying some classical methods such as, for instance, K-means [39] and K-nearest neighbors [4].

- ***Counterfactuals*** methods try to give explanations by finding out what is the minimal set of features for input such that, if slightly modified, would result in a different prediction for that particular input. These approaches also focus on discovering only features that would make sense to modify (*actionable* edits) by studying particular properties between the input and the training dataset. Counterfactual candidate samples do not necessarily have to belong and be chosen from the training dataset, indeed some techniques have also been elaborated to use generative ML models to build them ad hoc [38].

2.3 Knowledge Distillation

As already briefly mentioned, KD methods consist in extracting knowledge from a teacher model to improve another student model (typical accuracy improvements in classification tasks quantifies in about +5% [30]). This process is done with the objective of creating a student model capable of performances comparable to or even better than the teacher's ones while simultaneously and significantly reducing both the dimension in memory and the computational effort: this very often corresponds by default to the creation of a more explainable model, reason why KD is also widely used as an xAI technique.

Knowledge Distillation can usually be classified based on its learning scheme, namely based on how the distillation process is performed. The three main sub-categories are *Offline Distillation*, *Online Distillation* and *Self-Distillation* [30].

- ***Offline Distillation*** consists in transferring the knowledge from a pre-trained

teacher model to a student model. This means that the whole training process is performed asynchronously in two separate steps where the teacher is firstly trained without considering the student, then the teacher's predictions or some features extracted by it are used to lead the student's training. This procedure is the most simple among the three and usually also the most effective, however, because of the teacher's training, it may also be the most time-consuming. Moreover, student models rely a lot on the teacher, therefore making its role extremely important.

- ***Online Distillation*** inherits most of its features from the previous class; the only concept that varies, in this case, is that the training is "online", namely it happens synchronously, and both the teacher and the student models are trained simultaneously (for this reason, it is not rare to use parallel computing strategies while dealing with Online Distillation).

- ***Self-Distillation***, finally, can be imagined as a special case of Online Distillation: the concept here is that the teacher and student now share the same architecture and the knowledge transfer is more likely done between different layers of the model or between different epochs of the training session. This is the most recent proposed technique among the three and, although it has been already tested in some researches, it is not fully tuned yet.

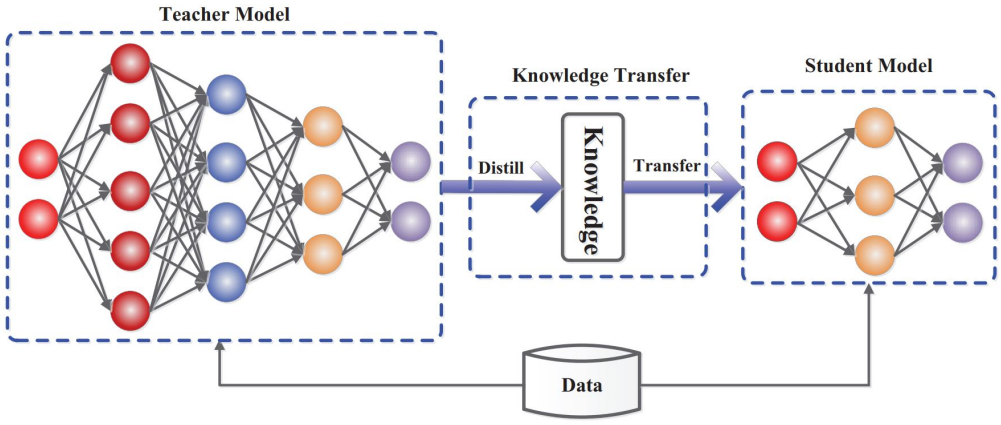


Figure 2.2. The classic KD framework.

Chapter 3

Related xAI techniques

Let us now, given the general definitions for xAI, explore more in particular but yet on a theoretical level the five different selected techniques implemented in this thesis. In the following, for each of them, I will outline their characteristics and design choices. Moreover, these techniques have been successively adapted to better address our specific task[1], therefore it is crucial to have some background knowledge related to them.

3.1 Saliency Maps Methods

Saliency Maps Methods are a specific kind of approach belonging to the post-hoc, local, and attribution-based xAI classes. In particular, the explanations given by them are images encoding pixels' importance for the output prediction, meaning that the most highlighted areas in the image show the main relevant regions causing the automatic decisions of the model.

Three out of the five implemented algorithms are from this group.

[1] All these variations will be elaborated in the Chapter 6.

3.1.1 Class and Regression Activation Map

The first algorithm comes from the *Class Activation Map* (CAM) family [64], later adapted to our regression task, namely relying on *Regression Activation Map* (RAM) [60].

CAMs, introduced by [64], are used to locally understand why the output of a Convolutional Neural Network (CNN) is produced by highlighting the most important discriminative regions of an image for its prediction[2]. These most important features of interest are generated as heatmaps using the feature maps of the model's last convolutional layer, located just before the final output layer. Figure 3.1 shows an example where the highlighted regions expose the relevant regions for a *"mouse"* prediction: we can see how effectively this kind of visual explanation can help users, in particular non-expert ones, giving an easy-to-read result that only depends on the quality of the highlighted pixels.

The key concept in the architectures implementing this method consists in the use of a *Global Averaging Pooling* (GAP) layer to connect the last convolutional layer and the output layer: doing so, each neuron in the GAP layer will contain the spatial average of the feature maps from the last convolutional layer, namely storing values expressing the neuron's involvement to the final prediction.

Taking into consideration a spatial coordinate (x, y) of an input image and a class c, $f_k(x, y)$ symbolizes the activation of unit k in the feature map f_k. It is used together with w_k^c, the weight to class c for unit k, to compute the class score S_c:

$$S_c = \sum_k w_k^c \sum_{x,y} f_k(x, y) = \sum_{x,y} \sum_k w_k^c f_k(x, y) = \sum_{x,y} M_c(x, y) \ . \tag{3.1}$$

We note that, in Equation 3.1, $\sum_{x,y} f_k(x, y)$ is the output of the GAP layer and M_c is the class activation map for class c, namely the $2D$ matrix containing $M_c(x, y)$ for each element (x, y).

S_c, which specifically quantifies how likely class c will be chosen by the model, is

[2]CAM is not obligatorily used only for images, it can also be used for other types of data, for instance, texts.

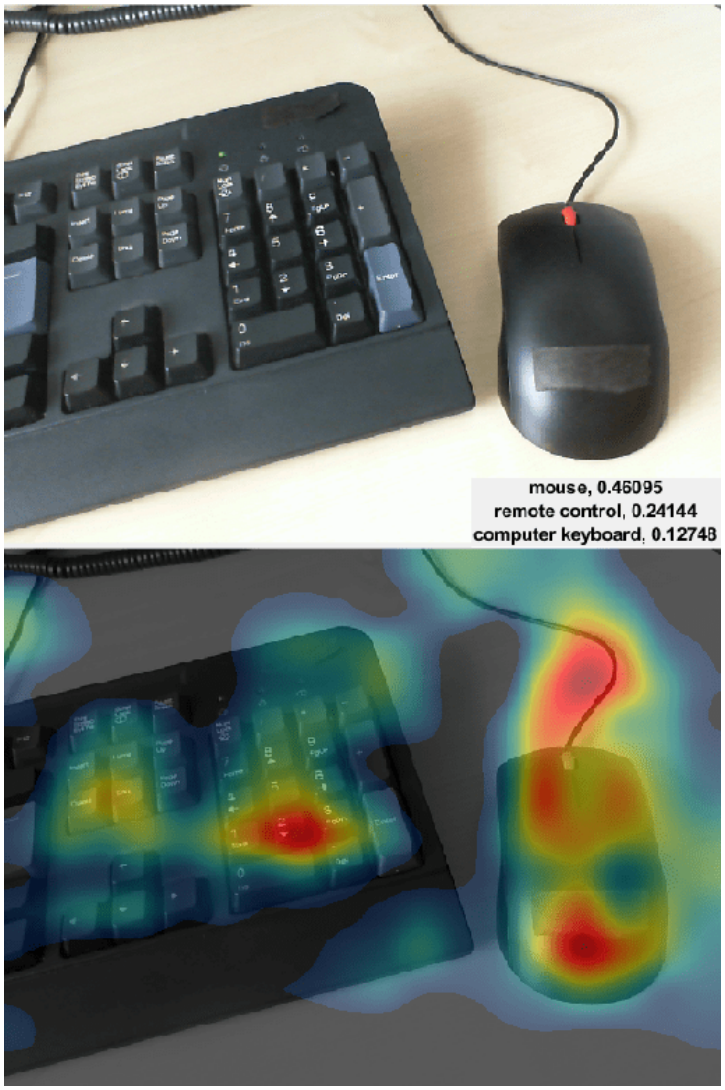


Figure 3.1. CAM example showing the regions of the input image contributing the most to the prediction.

then used as the parameter of a softmax function that simply outputs the probability to predict class c:

$$P_c = \frac{\exp(S_c)}{\sum_c \exp(S_c)} \ . \tag{3.2}$$

This method, although very simple to be applied, has some fundamental drawbacks that we had to face. For instance, to use an Activation Map approach, the considered model needs to have a specific final prediction head, therefore limiting the available usable architectures. Moreover, relying only on CNN's last convolutional layer for an explanation may lead to the loss of important information derivable from

previous layers. It also has to be noted that, instead of a GAP layer, it is possible to use a different component having a similar behavior (for instance, a *Global Max Pooling* layer).

RAMs [60] are slightly different modifications of the CAM idea: in particular, they basically share the same theoretical concepts and RAMs are also used to detect the relevant discriminative regions helpful to the model's prediction. The only trivial difference in the formulation of Equation 3.1 is the absence of the apex/subscript c from w_k^c, M_c and S_c since the concept of classes is no more present. Moreover, the final softmax activation function is not needed anymore and it is replaced by a linear function that directly gives S as the final output prediction.

3.1.2 Integrated Gradients

The second approach developed in this work is called *Integrated Gradient* (***IntGrad***) [55]. Being it a saliency-based method, similarly to Activation Maps, it produces as an explanation a heatmap representing the most important discriminative regions of an image. Anyway, the authors of this approach claim that the proposed method is able to overcome typical limitations of CAM-based algorithms by considering some properties that are usually neglected by them: *sensitivity* and *implementation invariance* [55].

The first property concerns situations where two different inputs diverge in only one feature (for instance, in only one pixel in study cases with images) but have different predictions: a method is *sensitive* if that specific feature is reported and properly highlighted in the generated heatmap due to its particularly heavy contribution.

The *implementation invariance* property is instead respected when, for every pair of *functionally equivalent* models, namely models that have different architectures but predict the same output for each equal input, the produced heatmaps are identical: the need of a specific model's prediction head described in the previous section is therefore overcome.

Integrated Gradients works by computing and accumulating the gradients at all points along a path between a *baseline* image x' and a target test image x. Clearly, both the baseline and the path to follow can be freely chosen: usually, the baseline is set to a black image (i.e., an image where all its pixels are zeroed) and the path is a straight line interpolating from x' to x. From these premises, we can define the following formula to compute the integrated gradient along the i^{th} dimension of x:

$$IntGrad_i(x) = (x_i - x'_i) \times \int_{\alpha=0}^{1} \frac{\partial F(x' + \alpha \times (x - x'))}{\partial x_i} d\alpha, \tag{3.3}$$

where F is a function representing a DNN, $F(x)$ is the prediction for x and $\frac{\partial F(x)}{\partial x_i}$ is its gradient along the i^{th} dimension. It is clear how the term α regulates the contribution of the original image in the interpolation. When dealing with gray-scaled or RGB images, α can be seen as a parameter that iteratively increases the intensity of the interpolated images as shown in Figure 3.2 [35].

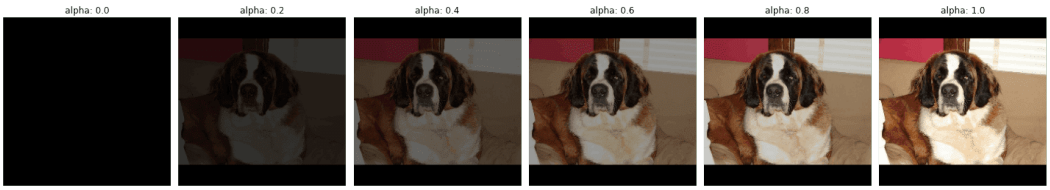


Figure 3.2. Example of the intensity interpolation process used by Integrated Gradients' algorithm.

Practically, the created set of images must have a limited cardinality m, thus the integral is easily approximated to an averaged summation as follows:

$$IntGradApprox_i(x) = (x_i - x'_i) \times \sum_{k=1}^{m} \frac{\partial F(x' + \frac{k}{m} \times (x - x'))}{\partial x_i} \times \frac{1}{m}. \tag{3.4}$$

3.1.3 SmoothGrad

The discussion of methods that exploit saliency maps is concluded including a further extension, ***SmoothGrad*** [54]: it at this point evident that its output is again a heatmap providing some level of understanding for the regression task. The core idea of this last approach is to create, given a test image of interest, new images

similar to the original one by adding random noise to it: these new n noisy samples can then be explained by means of any attribution method and the final result is obtained by averaging all the resulting heatmaps. The effects of this algorithm lie in general improvements in the visual coherence and sharpness of the generated explanation.

The paper [54] has demonstrated that, by choosing the correct amount of n pictures to average and of the noise percent to randomly add, the method is able to robustly eliminate intrinsic noise present in the original image from the final averaged heatmap. Figure 3.3 show the effects of a Gaussian noise addition to a sample image [35].

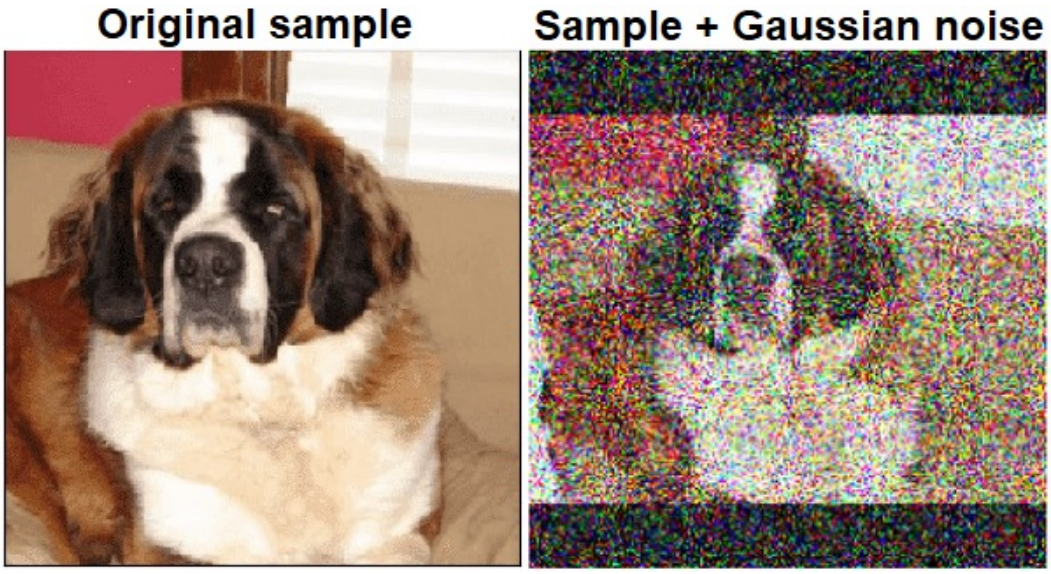


Figure 3.3. Visualization of a sample image together with a possible noisy version given in input to the SmoothGrad algorithm.

The noise removal effect of SmoothGrad is evident in Figure 3.4, where different approaches are compared. In particular for this work, we can see how overall it can get much cleaner and more accurate outcomes than IntGrad.

Mathematically, we obtain the final explanation map $\hat{M}_c(x)$ associated to a class c through the following equation:

$$\hat{M}_c(x) = \frac{1}{n}\sum_{1}^{n} M_c(x + \mathcal{N}(0, \sigma^2)), \tag{3.5}$$

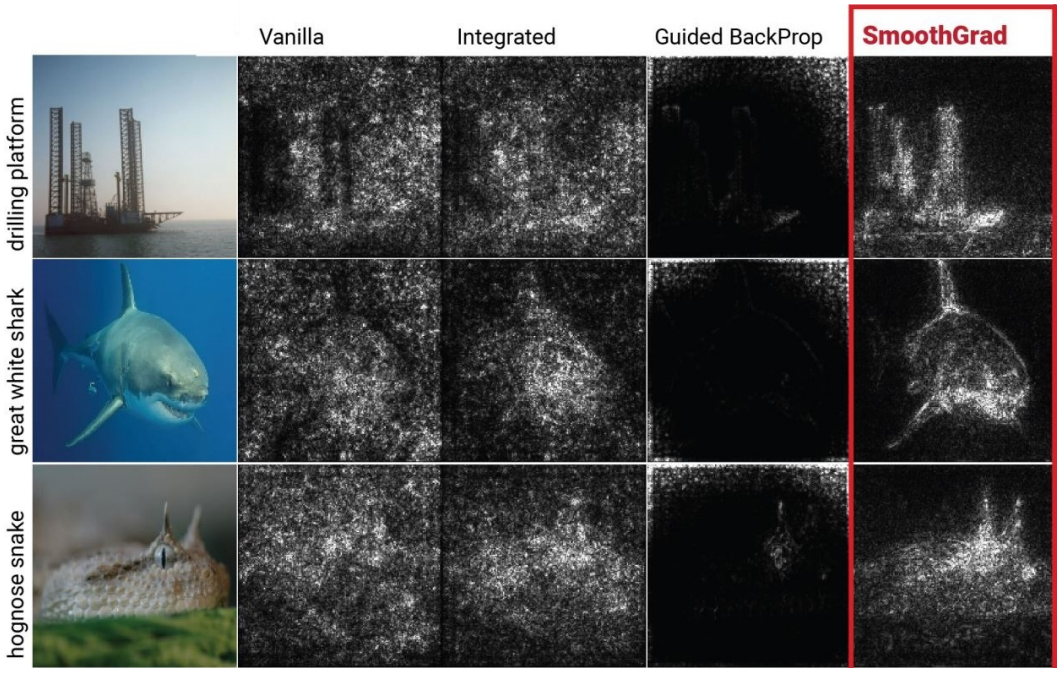


Figure 3.4. Example of the results obtained using some saliency map methods, among them Integrated Gradients alone and combined with SmoothGrad.

where $\mathcal{N}(0, \sigma^2)$ is a zero-mean Gaussian distribution with standard deviation σ that regulates the amount of random noise added in the pictures and M_c is a sensitivity map. As before, when approaching a regression task, the classes typical of a classification task are unnecessary and in Equation 3.5 we can merely delete the subscript c.

The mapping approach M can be chosen arbitrarily among the saliency-based methods: in this work, we will use a combination of SmoothGrad with the previously described method, IntGrad. This simply results in imposing $M(\cdot) := IntGrad(\cdot)$.

3.2 Intrinsically-interpretable models

Decision Trees [63], as earlier stated, are hierarchical models characterized by incredibly transparent explainability. Thanks to their hierarchical structure, it is always easy for users to determine why a decision is made at every level of the tree. A prediction given by this kind of model is determined by the leaf node reached at the end of its decision flow utilizing the logic rules previously learned for every inner

node. Indeed, when an input is given to a tree model, a particular route inside the tree is traveled based on the data's features. Figure 3.5 show an easy and instructive example of decision-making of a DT-based architecture: employing the three features *weather*, *humidity* and *wind* the model is capable of giving an output to a simple task of binary classification, namely deciding if recommending someone to go outside to play tennis. This straightforward example also allows us to underline again the logical explainability of such models.

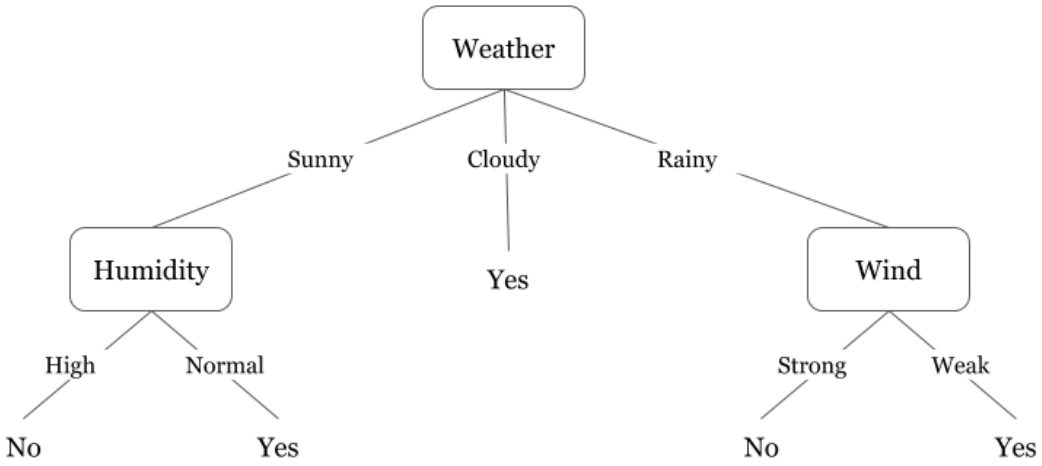


Figure 3.5. Toy Decision Tree. Using just a few features an output for the *PlayTennis* problem is efficiently reached.

Nevertheless, self-interpretable models are usually not able to provide reliable outcomes when the complexity of the ML task grows because they lack efficiency and are prone to overfitting [63].

An interesting and recent approach designed a particular architecture that can inherit the peculiar hierarchy of a DT and simultaneously exploit the effectiveness of a NN, creating an architecture called ***Neural Decision Tree*** (NDT) [5]. The basic idea of this structure is that each node in the tree can be modeled as a multi-layer perceptron (MLP) whose output is passed to a sigmoid function to determine which "branch" needs to be visited. This means that, for a tree formed by some MLPs fed with an input x, the output of a specific node i having trainable weights w_i and bias

b_i is defined:

$$out_i(x) = \sigma(w_i \times x + b_i). \tag{3.6}$$

Knowing that the sigmoid function σ maps any real value to the range $[0, 1]$, using the previous output we can simply determine the next step taken in the architecture from i as:

$$\begin{cases} \text{right, if } out_i(x) > 0.5 \\ \text{left, otherwise} \end{cases} \tag{3.7}$$

reaching a new node. This process is repeated for every crossed inner node at each depth level until a leaf node is reached, which is responsible for the final output of the NDT.

As for neural networks in general, further improvements of this model are achieved by staking more layers on top of the others, namely constructing a tree with a larger depth, defined ***Deep Neural Decision Tree*** (DNDT) [62]. DNDTs proved in [62] that they can achieve even better performances with respect to a DNN competitor, supporting the thesis of the benefits guaranteed by this deep and also interpretable structure.

However, the tree architecture was not exploited at its full potential yet and its development managed to provide us with an even more advanced model: Soft Decision Trees.

3.2.1 Soft Decision Trees

DTs and all of their variants discussed so far are characterized by a procedure that can be defined as "hard", meaning that, at each depth level of the tree, the decision flow proceeds to the left or the right child node when a splitting inner node is reached. This can imply a constraint in the exploration of other unvisited branches that can be good or even better than the current traversed branch concerning the ML task to solve.

To attenuate this limitation, the work by [33] has led to a similar structure but

with a different decision-making mechanism, called ***Soft Decision Tree*** (SDT). Even if the hierarchical architecture of the previously described NDT is preserved, its novel property of "softness" makes this model able to widely and deeply explore the whole expansion of the tree by visiting all the inner ad leaf nodes in a depth-first search fashion: every path from the root node to a leaf is associated with a probability and all the outputs of the leaves are considered with a contribution defined by the associated probabilities, that act as weights. This implication modifies Equation 3.6 such that, for each i-th inner node, the probability $p_i(x)$ of taking the leftmost (or rightmost, according to the desired design choice) is

$$p_i(x) = \sigma(w_i \times x + b_i). \tag{3.8}$$

Consequently, the probability associated with a leaf node l through a path $Path(l)$ is computed by considering all the probabilities of the inner nodes visited during the path. Such quantity can be mathematically expressed as follows [41]:

$$P_l(x) = \prod_{i \in Path(l)} p_i(x)^{l_i} \cdot (1 - p_i(x))^{1-l_i}, \tag{3.9}$$

where

$$l_i = \begin{cases} 1, \text{ if the next node is the left (or right) child of node } i \\ 0, \text{ otherwise} \end{cases}. \tag{3.10}$$

SDTs can obviously be applied in regression tasks, namely defining a ***Soft Decision Tree Regressor*** (SDTR) architecture. The regressive measures produced by such a model are computed, using the probabilities of Equation 3.9, as:

$$\hat{y} = \sum_{l \in LeafNodes} P_l(x) \cdot R_l, \tag{3.11}$$

where R_l defines the prediction that leaf l would give if consulted in a "hard"

fashion [41].

Moreover, it is significant to mention that in [41] a pre-processing normalization operation was applied, a notion that has been largely exploited in this project as well: given the original set of labels y, then its normalized version y' is defined as $y' = \frac{y-\mu}{\sigma}$, where μ and σ are, respectively, the mean and the standard deviation of y in the training set.

Moving on with the dissertation, the paper [28] triggered our interest in including a Knowledge Distillation approach in this work. Indeed, in [28], authors exploited and tested the KD notion leveraging a DNN as a teacher model for their SDT student in a classification task on the MNIST dataset, the large and very popular database of handwritten digits. Moreover, they also developed a tool to help the user visualize the internal decision-making path of the tree and evaluate its correct or wrong performed choices, an example is represented in Figure 3.6 [28]: as we will discuss later, in this thesis we assimilated the concept of this tool and re-proposed it in an enriched way.

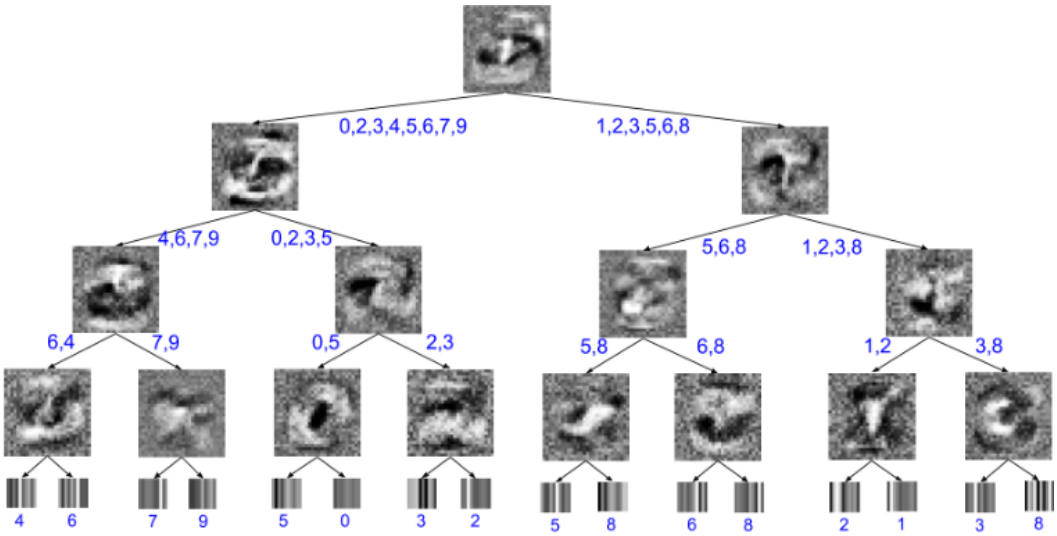


Figure 3.6. SDT application on MNIST: each node shows the perturbations of its associated MLP while the leaves communicate the classes' probability distributions.

It must be mentioned at this point that all the trees introduced until now, although having all the good intrinsic properties given by their hierarchical archi-

tectures, usually lack reliable performances when multi-dimensional data signals are involved. In fact, their effective usage is limited to some restricted applications only: they are extremely performing in use-cases where tabular data is utilized, but they usually report worse results when data are collected in different modalities like audio, images, and videos.

In order to extend the usage of Soft Decision Trees to such high-complexity domains, studies have further combined them with the convolution operation, which has experienced particular success with these kinds of data over the last decade of developments in the AI field. The work proposed by [2] has introduced the ***Convolutional Soft Decision Tree*** (ConvSDT), an extension of SDT with convolutional layers on top of the tree that are trained together with the weights of the MLP-nodes, and demonstrated how this kind of models could aim at being comparably efficient to CNNs even in use-cases normally entrusted to them.

3.3 Tracing Gradient Descent for data influence estimation

The last implemented approach tested for our use-case is a relatively new method based on the concept of *influence*, namely how much the prediction of a specific test sample is influenced by the train samples the model has been fitted on. The algorithm, denoted as ***Tracin Data Influence*** (TracIn) and developed in the work of [48], aims at estimating such influence by monitoring how the loss on the test sample to be explained changes at training time. The ideal version of this method is typically too onerous to be applied since it involves tracing all model's parameters and all used training points at each iteration of the training process. Moreover, usually, a model is fitted using a batch of data for every iteration, thus making it impossible to track back loss variations to some specific training sample.

Consequently, the authors have defined a first-order approximation combined with the usage of checkpoints, saved at training time, to recreate and simulate a

discretized evolution of the model's fitting. Let us expand on these two points.

The first point states that we can use a first-order approximation of the variations of the loss value of a test example instead of its actual value. The reason why this approach is accepted lies in the fact that the size of the rate controlling how the internal weights of a model are updated during training is generally very small. A loss function l can consequently be approximated as $l(w_{t+1}, z') = l(w_t, z') + \nabla l(w_t, z') \cdot (w_{t+1} - w_t) + O(\|w_{t+1} - w_t\|^2)$ where w are weights of the model at a certain time-step t and z' is a test sample.

The second notion of this approach concerns the use of the model's checkpoints saved at training time, enabling us to fully save and later check the evolution of the model. Once the training routine ends, TracIn can load offline and exploit k stored checkpoints having weights $w_{t_1}, ..., w_{t_k}$ to approximate the gradient descents involved in the process and estimate the influence of a training sample z for the prediction of some test point z'. Defining η_i as the step size between checkpoints $i-1$ and i, the formulated equation is:

$$TracInApprox(z, z') = \sum_{i=1}^{k} \eta_i \nabla l(w_{t_i}, z) \cdot \nabla l(w_{t_i}, z'). \tag{3.12}$$

Expanding the previous calculation Equation 3.12 to every single z composing the training set will result in a sort of influence ranking on z' where samples associated with positive values of influence are called *proponents*, since they support the process reducing the loss, in contrast to the *opponents* that are samples with negative values and increase the loss. The explanation of the test sample z' can be intuitively modeled by observing the proponents and opponents with the highest and lowest influence scores to understand the samples that have mostly affected the prediction both in a positive and negative aspect.

Even if the method is computationally heavy because it requires calculating the influence of each point in the whole train set, its implementation is very simple and easy to understand.

The authors of [48] provide an example of the application of TracIn on images using a neural network trained on the Imagenet dataset, the very large and famous dataset for visual object recognition: Figure 3.7 depicts their results where it is noticeable the relation between the sample images and their associated Proponents (correctly directed the prediction) and Opponents (highly-confusing pictures that are somehow still related to the input).

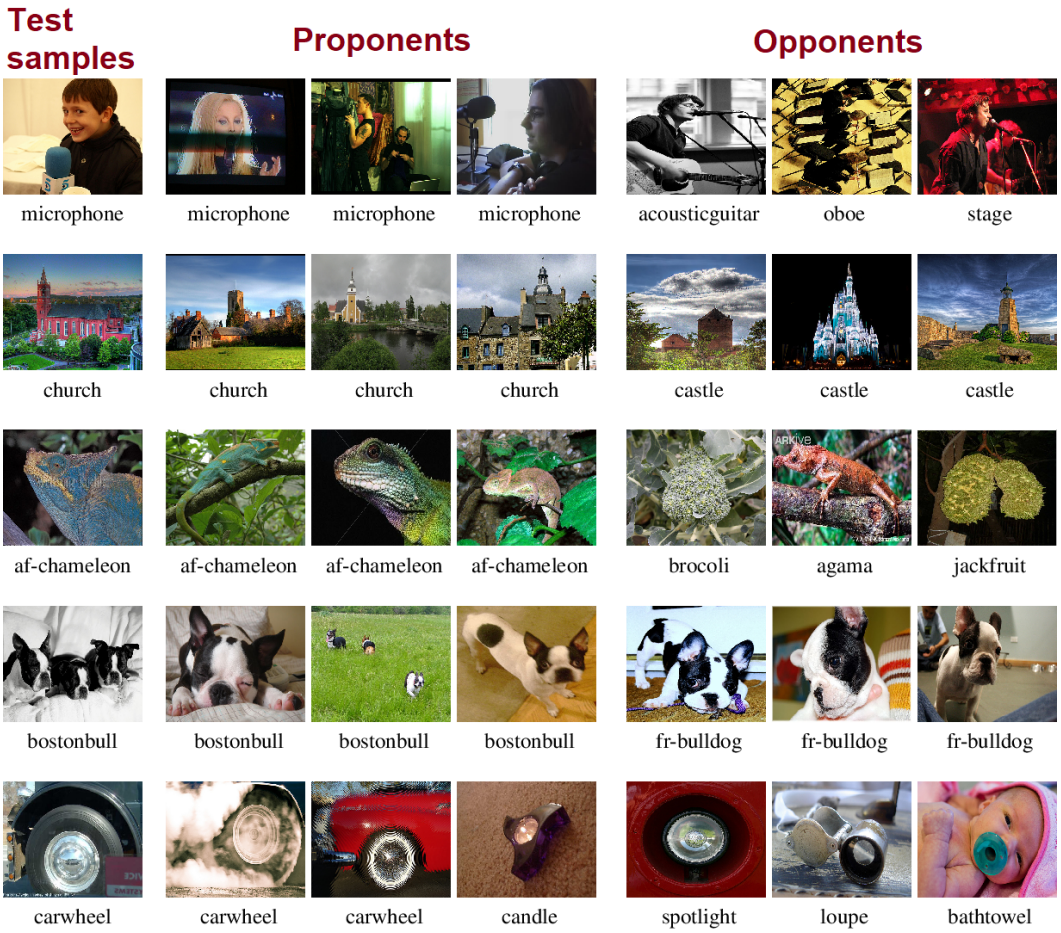


Figure 3.7. Example of results obtained by the Tracing Gradient algorithm.

Chapter 4

Foundations of High-Energy Physics

The use-case employed in this thesis comes from the HEP domain, one of the leading areas of study in modern Physics. In particular, HEP is generally viewed as part of the larger *Particle Physics* topic, which deals with the study of the elementary constituents of matter [46]. Researchers, to investigate and study these *pointlike* particles, strictly depend on the available technological probe instruments and their resolution[1]: the better we are able to look into the matter, the better we can understand it [46].

At the beginning of the last century, particle accelerators still had a very poor resolution of $\approx 10^{-12}$ meters (m) and protons and neutrons were considered elementary components of the universe. Anyway, the discovery of the electron and a remarkably great number of other particles, significantly pushed this field's advancements and the construction of high-energy accelerators achieving a current resolution of $\approx 10^{-16}$ m [46].

Another reason for the need for such high energies in experimental particle physics is that most of the elementary particles are extremely massive and therefore

[1]The resolution is defined as the minimum distance measure Δr such that, if two points are at a distance Δr apart, they can be determined as separate.

the energy $E = mc^2$ required to create them is thus substantial [46].

To summarize, to understand the existing interactions between the particles forming the universe we need machines that artificially create and control the necessary quantity of energy to let researchers perform experiments on them.

One last relevant topic in HEP is the definition of its measurement units, some of which are shown in Figure 4.1. While the generally known *International System of Units* utilizes meters, kilograms, and seconds for length, mass, and time, in the considered domain they are impractical: typical lengths are in the order of 10^{-15} m, masses 10^{-27} kg, times 10^{-25} s. Especially for later in this work, it is crucial to define one important unit: the Giga-Electron-Volt (GeV) equal to 1.602×10^{-10} J, which is the commonly used energy unit and, when considering the light speed $c = 1$ (usually done in HEP because of the frequent presence of c), it is also used as mass unit [46].

Quantity	High energy unit	Value in SI units
length	1 fm	10^{-15} m
energy	1 GeV $= 10^9$ eV	1.602×10^{-10} J
mass, E/c^2	1 GeV/c^2	1.78×10^{-27} kg
$\hbar = h/(2\pi)$	6.588×10^{-25} GeV s	1.055×10^{-34} J s
c	2.998×10^{23} fm s^{-1}	2.998×10^{8} m s^{-1}
$\hbar c$	0.1975 GeV fm	3.162×10^{-26} J m

Figure 4.1. Some of the most important High-Energy Physics units of measurement.

4.1 The Standard Model

The Standard Model of Elementary Particles (SM) is a theory that describes particles as relativistic quantum fields and their interactions under three of the four known fundamental forces: electromagnetic, strong, and weak interactions

are included while gravitational one is excluded, nevertheless, it can be considered negligible at collider energy scale [47, 52, 53].

The SM claims that matter is composed of two families of particles: *fermions*, which follow Fermi-Dirac statistics, and *bosons*, which obey Bose-Einstein statistics. Moreover, these elementary particles can be again broken down into specific elements, categorized as reported in Figure 4.2.

More particularly, the fermion family contains *quarks* and *leptons* [18, 61]: the former are responsible for making up protons and neutrons, famous parts of the atom generally studied in high school, while the latter are the elementary particles not subject to the *strong force*, one of the fundamental forces, which is responsible for binding infinitesimal structures. Fermions are the interacting particles of this model [61].

On the other hand, *bosons* transfer discrete amounts of energy and are, therefore, the particles that by being exchanged are appointed to generate the fundamental forces in the universe [18].

The most relevant particle for the use-case of this thesis is the ***muon***, an unstable subatomic particle belonging to the class of leptons (in green in Figure 4.2). Muons are as massive as approximately 200 electrons [11] while being also very highly energetic, their motion proceeds at roughly 99% of the light speed. They are generated from cosmic rays, high-energy beams that continuously hit our atmosphere. In fact, when a cosmic ray proton impacts atomic nuclei in the upper atmosphere, particles named *pions* are generated which rapidly decay into muons and muon neutrinos that naturally propagate down, towards the surface of the Earth [3]. Given the cause of their origin, muons are naturally the prevalent part among the cosmic radiation hitting Earth and they are subject to recently-started and interesting studies: in a minute, about 10000 muons reach every square meter of our planet's surface [11].

Although the SM has been capable of justifying almost every theoretical high-energy particle interaction empirically over the years, there are yet phenomena that

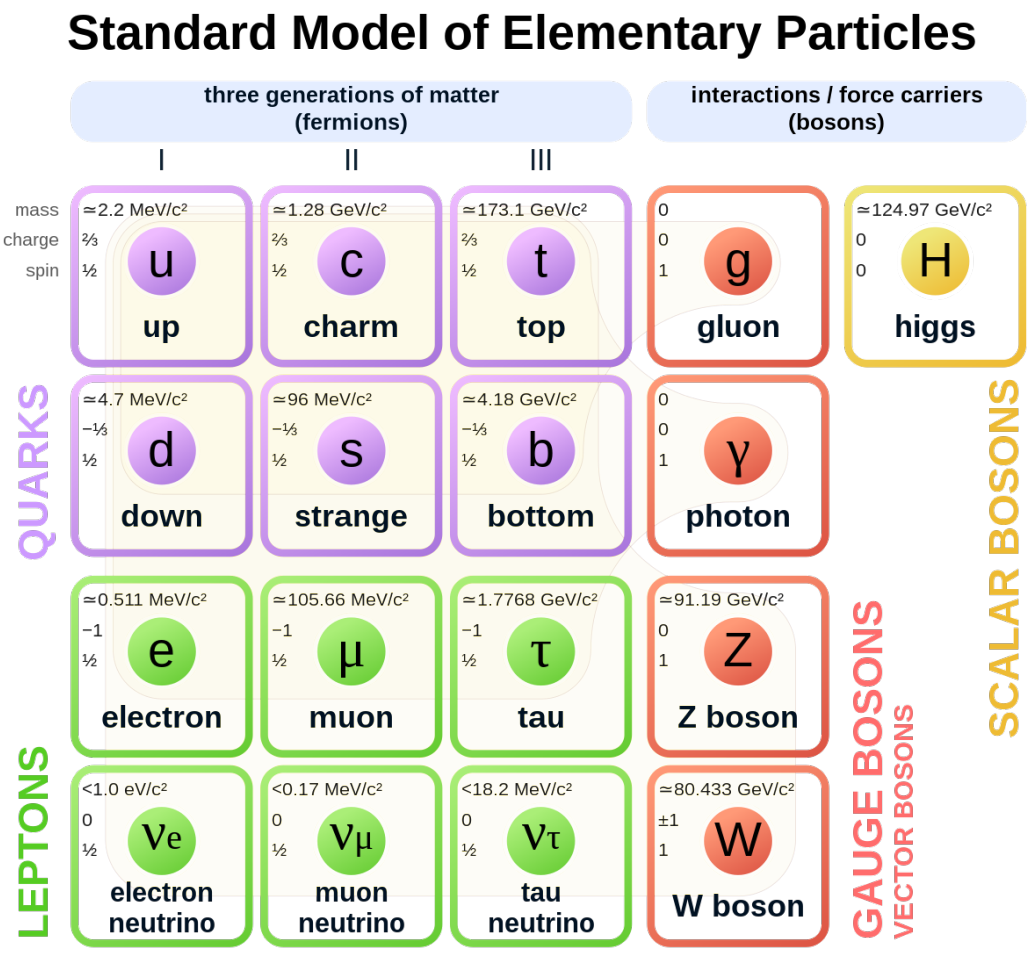


Figure 4.2. Elementary particles of the SM: fermions (on the left) are composed of quarks and leptons, bosons (on the right) are distinguishable in gauge and scalar bosons.

cannot be explained by it [18]. Indeed, it is estimated that it can describe only the 5% of the known universe, leaving unanswered important questions about the other hypothesized components of our reality: *dark matter* (27%) and *dark energy* (68%) [15].

As imaginable, this thesis will largely omit a real dissertation about all the remaining HEP characteristics: in the following, we will only focus on some more aspects strictly relevant to fully understanding the ML scenario in question.

4.2 The European Council for Nuclear Research

In December 1951, during an intergovernmental meeting of UNESCO in Paris, it was born the most important and famous research center for HEP: the *European Council for Nuclear Research* (CERN). It is located in Geneva and now, after more than 70 years and with 23 Member States, CERN is recognized as the most advanced physics center for research in the Particle Physics field [7].

One of the most recent and notorious contributions of CERN to humanity's understanding of reality date back to 4 July 2012 [18] when it was observed a new particle in the mass region around 126 GeV which is believed to be the *Higgs Boson* predicted by the SM: for the moment being, this discovery is still consistent and led François Englert and Peter Higgs to obtaining the Nobel prize in physics in 2013.

CERN is equipped with the world's largest and most complex scientific instruments [7], at CERN researchers can interact with *particle accelerators* and *detectors*: specifically, these highly technological machines are needed to experiment with high-energy particles. Their particular duties are, for accelerators, to boost beams of particles to increase their energy through the use of electric and magnetic fields and direct their motions to make them collide. Instead, detectors record the results of these collisions by tracking and storing specific parameters and properties of the particles for further studies: the muonic data used in this thesis is directly derived from the observations registered at CERN.

Moreover, it is interesting to mention that a new research infrastructure is being designed for construction: the *Future Circular Collider* (FCC), which hopefully will host in ≈ 2050 the next generation of HEP researches with the intent of reaching collision energies of 100 TeV (10^5 GeV), in the search for new physics [9]. Let us now analyze the current technology pertinent to this work.

4.2.1 The ATLAS detector

The dataset of muons' motion patterns used in this work, later described in Chapter 5, is originated in CERN's ***ATLAS*** detector, where more than 5500 people investigate and perform Particle Physics experiments [17]. As sketched in Figure 4.3, ATLAS is positioned along the path of the accelerator *Large Hadron Collider* (LHC), 100 meters below ground at one of its four interaction points: the remaining three interaction points host the three other detectors where different scientific matters are studied.

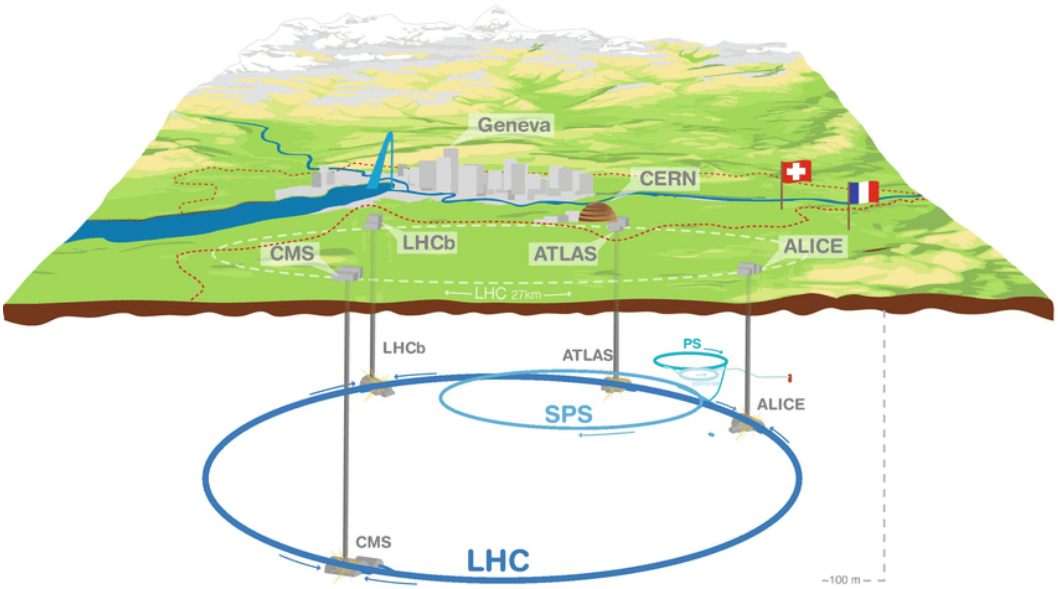


Figure 4.3. Illustration of LHC and its four interaction points: ATLAS, ALICE, CMS, and LHCb.

ATLAS, an acronym for "A Toroidal LHC ApparatuS", is the largest cylindrical particle detector in the world: it is 46 m-long with 25 m of diameter and reaches a weight of 7000 ton [8]. The ATLAS detector has a multi-layer structure in order to optimize particle detection, identification, and parameter measurements. Figure 4.4 is a graphical computer-generated image showing the vastness of how many sub-components make it up.

Among this complex system, the *Muon Spectrometer* is the specific area where muonic patterns are recorded: in fact, given their physical properties, they can pass

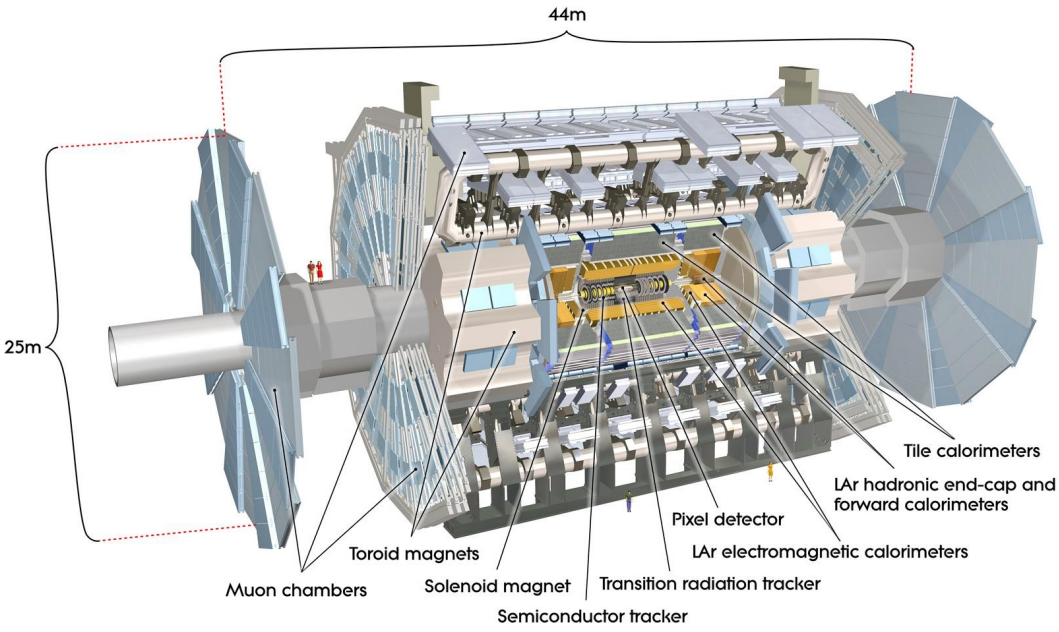


Figure 4.4. The ATLAS detector and its components.

through ATLAS's Inner Detector and Calorimeter undetected [12]. The spectrometer's arrangement is pictured in Figure 4.5 and features three portions: the barrel and two end-caps. In the barrel, muonic patterns are measured using special chambers positioned in three cylindrical layers around the beam axis. On the other hand, three layers of chambers are placed perpendicularly to the beam in the end-caps. In total, the spectrometer is formed by 4000 muon chambers with which muonic patterns can be observed.

Four different types of chambers are contained in the muon spectrometer, and the data used in this work comes from only one of them. Indeed, the *Resistive-plate chambers* (RPC) are the ones relevant for our measurements: they are fast gaseous detectors composed of two parallel plates, made of a very high resistivity plastic material, that act as a positive anode and a negative cathode [16]. Between the two plates, a thin gas volume is placed so that, when a muon passes through the chamber, electrons are knocked out of the atoms of the gas and used to compute the muon's properties [16]. RPCs combine a good spatial resolution with a time resolution of just one nanosecond [16]. Inside the chambers is present an electric

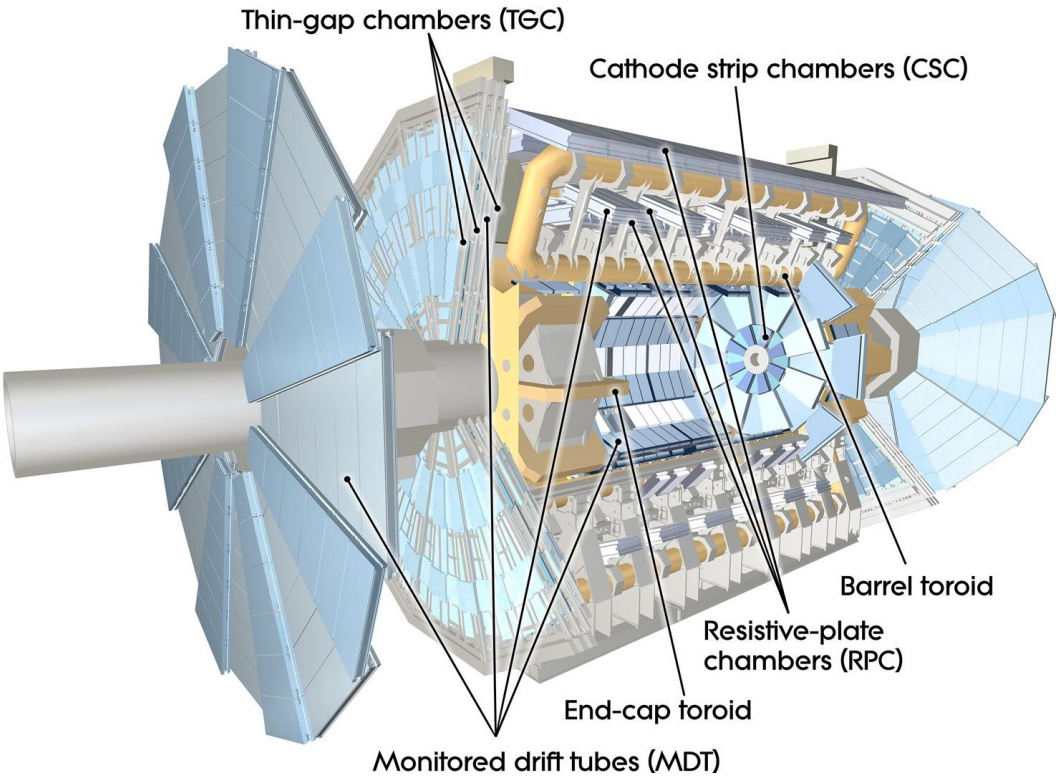


Figure 4.5. The ATLAS Muon Spectrometer.

field of 5000 V/mm [12].

4.2.2 ATLAS's Trigger and Data Acquisition systems

ATLAS has the spectacular capacity of observing up to 1.7 billion collisions per second which generate a data volume greater than 60 Terabytes per second [19]. However, very few of these recorded impacts will contain analyzable characteristics: it is hence needed an apparatus for incredibly fast decisions called *Trigger and Data Acquisition system* (TDAQ) is responsible to select only the collision events having some set momentum and reducing data to manageable amounts. The first level of the trigger has a decision time of 2.5 microseconds and passes approximately 100000 events per second to the second level, where more in-depth analysis is performed using 40000 CPUs [19]. In the end, only about 1000 events per second are selected for further studies.

Let us now introduce the main muonic measurements considered inside TDAQ

that will be later significant for this thesis: ***transverse momentum*** p_T and ***pseudorapidity*** η, effectively storing a lot of insights about muonic patterns in only two parameters.

For the following, we consider placing the origin of a $3D$ coordinate system at the interaction point of ATLAS such that the x-axis positively points towards the center of the LHC ring, the z-axis is directed as the beam of particles and the y-axis is pointed upwards [21]. Simultaneously, let us keep in mind that, for the same reference frame, polar coordinates can be adopted instead of Cartesian ones[2]: r, θ and ϕ, where r defines the distance from the origin of the system, θ is the angle from the z-axis and ϕ is again an angle from the x-y plane [21]. Figure 4.6 summarizes the defined coordinate systems.

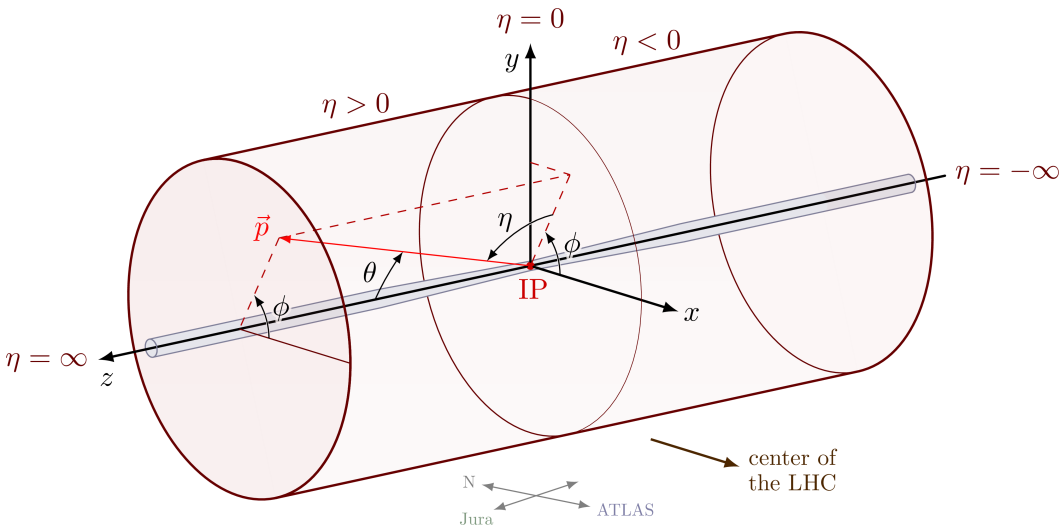


Figure 4.6. The ATLAS coordinate system.

Having established this reference system, we can define the following quantities:

- The **momentum** p is the first relevant measure for this work: given the mass m of the particle and its velocity vector v, it is defined as their product $p = m \cdot v$ and, considering our $3D$ coordinate system, it results in a vector made up of 3 components, $\boldsymbol{p} = (p_x,\ p_y,\ p_z)$. The **transverse momentum**

[2]Polar coordinate can be helpful in this case because, as mentioned before, ATLAS is a cylindrical detector.

p_T is a simple scalar quantity associated with p and can be computed as

$$p_T = \sqrt{p_x^2 + p_y^2} \, . \tag{4.1}$$

p_T is measured using the earlier-mentioned GeV energy unit. This metric is declared as "transverse" and given the subscript "$_T$" because it is computed with respect to the x-y plane [21].

- The other interesting quantity is called **pseudorapidity** η, and can be declared as [21]:

$$\eta = -\ln\left[\tan\left(\frac{\theta}{2}\right)\right] . \tag{4.2}$$

It is a dimensionless amount ranging in the interval $[-\infty, +\infty]$ and its value determines a non-linear relationship with the arrival angle of particles approaching the detector.

Chapter 5

Materials and tasks

Now that all the introductory theoretical topics have been assessed, we may begin the more practical discussion on the actual focus of this work. This thesis has two main goals, the first one is solving very accurately a regression task about predicting the measures associated with muonic patterns that have been collected by the ATLAS detector. These prediction labels are the previously mentioned transverse momentum p_T from Equation 4.1 and the pseudorapidity η[1] from Equation 4.2. The second, and I would also say the most important, task will be addressed in the next Chapter 6 and concerns the creation of explainability techniques to provide user-understandable explanations for models' predictions.

5.1 Main and only-noise datasets

As analyzed before, when a muonic event takes place in the RPC chambers of the detector, it may be accepted by the ATLAS's TDAQ system: in this case, the muon's trajectory is saved into an "image". We must now make a consideration: in the following, the term "image" will be used to define the items composing our dataset even if it is not properly correct. This is because our data are in reality expressed by complex signals that have been converted to make them easily visualizable into

[1]Actually, the pseudorapidity label is related to the η_{index} value, soon defined in the section about datasets.

a fixed $2D$ mesh that, after stabilization and calibration processes, manages of simulating the particle motion in an image.

The images we will take into consideration can be modeled as matrices of shape 9×384. Along the height of an image, there are 9 vertical pixels corresponding to the layers of the RPCs traveled by the muon there-in depicted. On the opposite, the 384 horizontal pixels across the width of the image outline η_{index}, a pseudorapidity variation. This quantity is a "pixelized" version obtained via a linear mapping of the real η and is designed such that the lowest lit pixel in the pattern reflects its value. The indexed η obtained by means of:

$$\eta_{index} = 384 \cdot \frac{|\eta| - \eta_{min}}{\eta_{max} - \eta_{min}} \ . \tag{5.1}$$

In Equation 5.1, η_{min} is set to 0 while $\eta_{max} = 0.95$. From now on, we will refer to the quantity $\eta_{index}/384$ as simply η.

Every image is associated with its own binary label $[p_T,\ \eta]$. These images are binary gray-scaled (black-and-white) pictures where 0-valued pixels are predominant and only a few pixels are lit with a value of 1, meaning that the associated sensors of the detector have registered a muon movement. Figure 5.1 shows an example of what an image from the dataset looks like. Unfortunately, in realistic settings it is implausible that only the actual trajectory of the muon is captured in the image but also some amount of random noise caused by electric or environmental radioactive artifacts generated by the high intensity of the particle beams of the LHC in the ATLAS cavern, radiating a neutron and photon gas causing some noisy wrong "hits" in the final image. Figure 5.2 depicts the actual perceived event related to the previous Figure 5.1, it is noticeable that both the relevant component with the actual pattern of the muon and the additional radioactive noise are present. It should be noted that, while every image of the dataset contains really randomly noised pixels, they are not completely incoherent, but evenly spread throughout the image: it is rare to find a lot of noise in a restricted area.

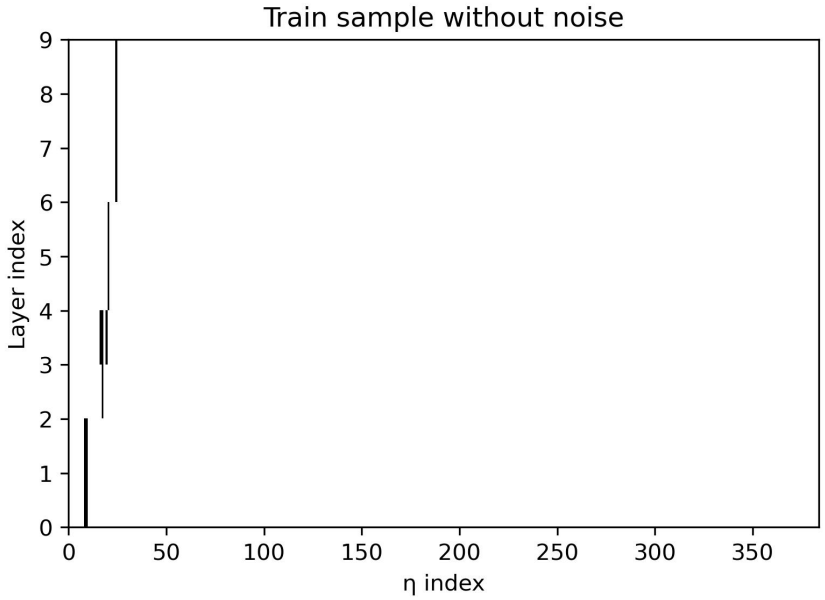


Figure 5.1. Sample image containing the muonic pattern.

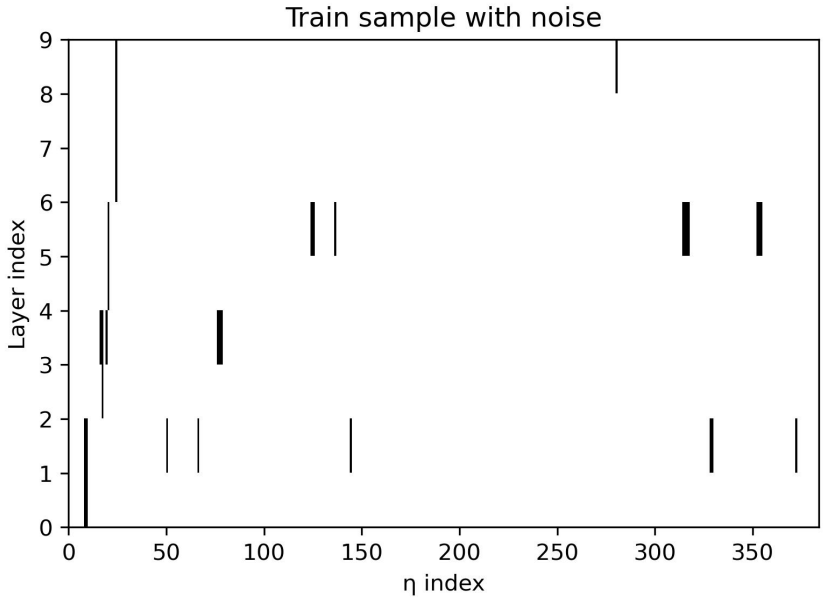


Figure 5.2. Sample image containing the muonic pattern and additional noise.

With reference to Figure 5.1, let us inspect what information can be extracted from a muonic pattern. Inside LHC, muons are affected by an electric field E and a magnetic field B: they, therefore, are affected by Lorentz's electromagnetic force[2].

[2] $F = q(E + v \times B)$, with q being the muon's electric charge and v its velocity.

Because of it, muons' trajectory is deviated in a curved fashion depending on their velocities: a particle with a sufficiently high velocity will be less influenced by this force than a slower one, resulting in having a less curved trajectory. This analysis lets us deduct that, if a muon performed a very straight path, it is likely that it is very fast and it consequently must have a high transverse momentum.

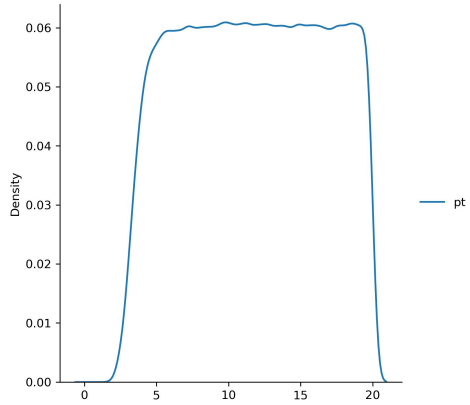
This thesis uses the dataset originated from [13]: from it, two datasets were extracted by us and they were made available at [10] and [14].

The first, and main, one [10] includes 945k instances. Each sample contains its ground-truth label and two images: the noisy one as well as the denoised one. However, it has to be noted that clearly our ML models were trained using only the noisy images supervised with the labels. The denoised ones were only later employed for support at the explanation phase, excluding them from the training steps.

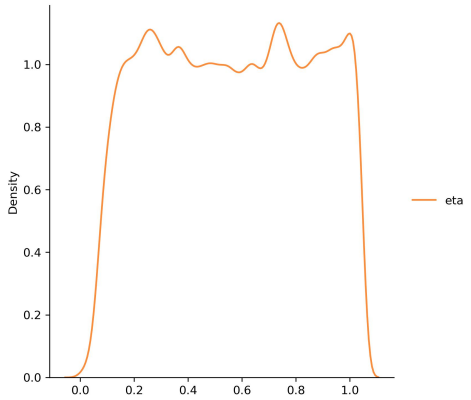
A 90-10 train-validation split has been employed on this dataset by its original authors [13], analyzing them we found out that the sections were specifically created to have a comparable data distribution among them as demonstrated by Figure 5.3 and Table 5.1. The figure depicts how many samples with certain p_T and η values are in existence for both datasets: we can notice the distributions' similarities confronting Figure 5.3a with Figure 5.3c and Figure 5.3b with Figure 5.3d. Instead, the table directly states the values of their statistical properties: means and standard deviations are literally the same, with a minimal difference of 0.01 for the *std* of p_T.

Throughout the following section, it will be explained that some p_T ranges are for us of greater interest, in fact, the main objective for the effectiveness of the trained model is based on the extent to which it is capable of selecting or rejecting muons in certain ranges of momentum values. In particular, being able to correctly predict patterns with a real $p_T > 15$ GeV is crucial for our task. However it should be noted that this segment is underpopulated compared to the rest of the dataset: in the training dataset, only 30.194% of the samples have a momentum higher than 15 GeV, while in the validation dataset only the 30.205%. Finally, it should be evident that images with $p_T < 2$ GeV are not present at all among the data because

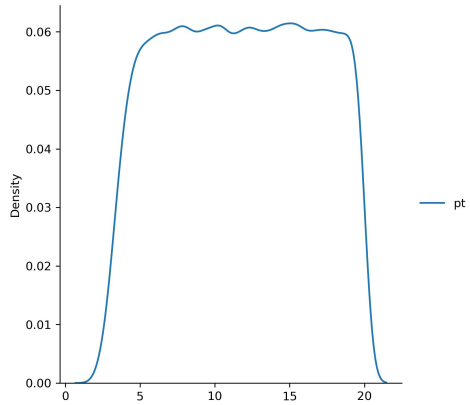
those muonic events are discarded by TDAQ's first-level controls because of their low amount of energy.



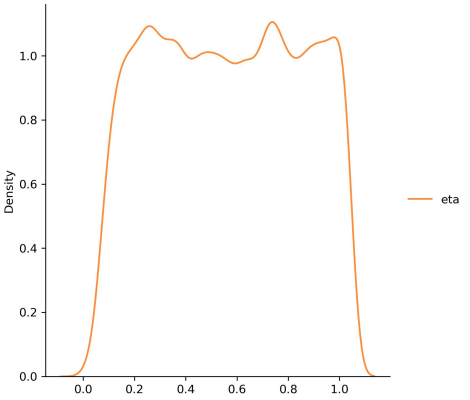
(a) Transverse momenta distribution in the training set.



(b) Pseudorapidity distribution in the training set.



(c) Transverse momenta distribution in the validation set.



(d) Pseudorapidity distribution in the validation set.

Figure 5.3. Data distribution of the main dataset.

The second dataset [14] was not employed during training, similarly to the denoised images of the main dataset, but exclusively in the explanation stage and it has only images where solely noise events were recorded. This dataset's properties are highlighted again in Table 5.1: although it contains the same amount of images as the first dataset, no correspondence between them has been detected. Moreover, given that these samples do not contain any muonic pattern, both their labels were simply assumed to be 0.00.

The purpose of this second body of data is dual: it can give knowledge both on

the regression and the explanation task. Through it, we could check the robustness of our models to images never seen at training time, with never observed patterns of particles and labels, and investigate the cases wrongly predicted to have a high momentum. Similarly, by analyzing their additional explanations we managed to understand even more the decision-making process of our architectures.

Set	#	Label	Mean	Std
Training	850003	p_T	11.71	4.81
		η	0.56	0.28
Validation	94445	p_T	11.71	4.80
		η	0.56	0.28
Only-noise	944448	p_T	0.00	0.00
		η	0.00	0.00

Table 5.1. Statistical quantities of momentum and pseudorapidity labels for the main dataset (subdivided in training and validation sets) and the only-noise dataset.

5.2 Task metrics

In order to evaluate the image regression task for HEP we have adopted and defined specific evaluation criteria for training and validation processes.

At training time, we required our models to minimize the prediction error both on p_T and η: we used the *Mean Absolute Error* (**MAE**) loss. Stating that y_i is the ground-truth $[p_T,\ \eta]$ and $\hat{y}_i$ is the predicted value $[\hat{p_T},\ \hat{\eta}]$ for the *i-th* sample, MAE can be computed for batches of data as:

$$MAE(y, \hat{y}) = \frac{\sum_{i=1}^{n} |y_i - \hat{y}_i|}{n}, \tag{5.2}$$

On the other hand, when computing this loss function for only a single sample, its formulation simply becomes $MAE(p_T, \eta, \hat{p}_T, \hat{\eta}) = |p_T - \hat{p}_T| + |\eta - \hat{\eta}|$, where again the "hat" symbol represents a quantity predicted by a model.

Since we are dealing with a HEP use-case, specific physics-related metrics have been proposed to evaluate our work and the performances of the networks developed

in the implementation. In particular, two measures to quantify the effectiveness of our trained models have been adopted and, we must notice, both of them have acted on p_T only. The definitions of these metrics are:

- **spread:** The average value of the difference between real and predicted momenta in the range $[7, 13]$ GeV, namely around the threshold of 10 GeV. Let us imagine that y_{p_T} and $\hat{y}_{p_T}$ are, as before, the vectors containing ground-truth and estimated p_T: in this case, η is not considered. From this, we can define a set of vector indices pointing to momenta belonging in the said range as $S = \{s \mid y_{p_T,s} \in [7, 13] \text{ GeV}\}$. The spread value is then found:

$$\text{spread}(y_{p_T}, \hat{y}_{p_T}) = \frac{\sum_{s \in S} |y_{p_T,s} - \hat{y}_{p_T,s}|}{\#(S)}, \tag{5.3}$$

  where $\#(\cdot)$ represents the cardinality of a set. This formulation is very similar to the previous MAE's definition: indeed, we can conceptually figure the spread as a MAE only applied on momenta and only, specifically, on momenta in the $[7, 13]$ GeV range. This quantity is optimized through its minimization.

- **plateau efficiency:** The "precision" metric for samples whose real momentum is above the 15 GeV threshold. We can again define two useful sets of vector indices: $E_{den} = \{e \mid y_{p_T,e} > 15 \text{ GeV}\}$, namely the indices of relevant muonic samples, and $E_{num} = \{e \mid y_{p_T,e} > 15 \text{ GeV} \wedge \hat{y}_{p_T,e} > 15 \text{ GeV}\}$, namely the subset of E_{den} containing samples that were also predicted with high momentum. In the following, I will refer to the plateau efficiency simply as **efficiency**, now merely definable as:

$$\text{efficiency}(E_{num}, E_{den}) = \frac{\#(E_{num})}{\#(E_{den})} \tag{5.4}$$

  This quantity is optimized through its maximization.

The efficiency metric testifies to the importance of correctly rejecting or selecting muonic events along the 15 GeV threshold. This separation can be conceptualized as

a typical binary classification sub-task where a muon can be correctly (or wrongly) selected or rejected. Using standard ML terminology for classification tasks, we can differentiate four predictive outcomes: True Positive (TP) and True Negative (TN), when a muon is correctly selected or rejected; on the opposite, when the model places a sample in the wrong sector, it can be a False Positive (FP) or a False Negative (FN), whether it wrongly accepted or rejected the muon. As usual, these four prediction sectors can be combined to be visualized in a confusion matrix, Figure 5.4 clarifies and summarizes the situation.

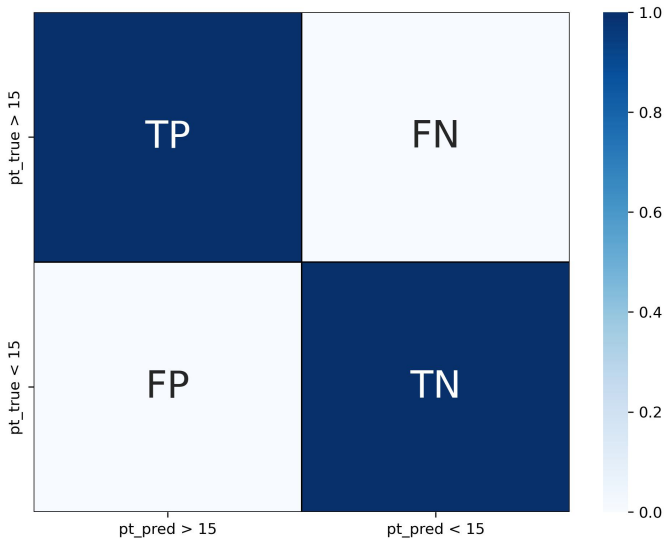


Figure 5.4. A confusion matrix example depicting the four classification areas.

Other than the precision metric expressed so far by the efficiency, using TP, TN, FP, and FN sets it is possible to compute different general evaluative quantities. For instance, this work will mention:

$$Precision = \frac{TP}{TP + FP}; \tag{5.5}$$

$$Recall = \frac{TP}{TP + FN}; \tag{5.6}$$

$$F1\text{-}score = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}; \tag{5.7}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}. \tag{5.8}$$

Chapter 6

Implementation

We can now begin the exposition on the pragmatic implementation details and, moreover, the best experimental settings deducted from some conducted experiments will now be outlined. Hereafter I will describe all the specifics of the selected ML models and the variations we applied to the earlier proposed explainability techniques to make them work the best in our application domain. As stated in previous chapters, this project uses and is based on a variegate array of current SOTA approaches to allow us to coordinately target a vaster category of end-users, from non-experts to researchers with knowledge in Particle Physics.

6.1 Models

The architectures provided by this thesis consist of two Convolutional Neural Networks and a Convolutional Soft Decision Tree:

- ***Attribution CNN***: It is the first model we implemented and, as understandable by its name, this network is associated with the three attribution-based saliency maps methods.
- ***TracIn CNN***: As mentioned in the following, this second architecture is very similar to the first one and it was trained to later be used with the TracIn approach.

- ***ConvSDT***: This is our Convolutional Soft Decision Tree model. The same architecture has been trained two times: when we will refer to the network produced through the KD training procedure we will specify it by defining it ***Distilled ConvSDT***.

6.1.1 Convolutional Neural Networks

CNNs were trivially designated for our application domain because, given the fundamental convolutional operation helping to preserve the context of the spatial structure of multi-dimensional data, they are particularly suited for datasets containing graphical inputs.

The *Tensorflow 2*-based CNNs used here are quite simple: after an *Input* layer having the same dimensions of our muonic images, namely (9×384), they are made up of a succession of four bidimensional convolutional layers. These layers have an increasing amount of output filters, respectively, of 16, 32, 64, and 128. Every layer uses a kernel of size (5×5), a *same* padding is employed and a $ReLU$ activation function is applied after each convolution.

As expected for the subsequent application of RAM, the output of the body of this architecture is passed to a Global Average Pooling (GAP) layer and, finally, the regressive output is given by the head of the model: here is where is located the only structural difference between Attribution CNN and TracIn CNN. The final component of Attribution CNN is, in fact, constituted of three *Dense* layers having respectively 1024, 512, and 2 units. On the opposite, the head of TracIn CNN is made up of only a single final *Dense* layer with 2 neuronal units; this design choice is motivated by intrinsic implementation requirements of the Gradient Tracing algorithm [48].

The penultimate and the third last layers of Attribution CNN use $ReLU$-s activation functions while its final one and the single layer of TracIn CNN's head use a *Linear* activation function.

Table 6.1 represents the structure of the Attribution CNN model and its related

trainable parameters, from this table the features of TracIn CNN are also easily deductible.

Layer	**Output shape**	**# params**
Input Layer	$(9, 384, 1)$	0
Conv2D	$(9, 384, 16)$	416
Conv2D	$(9, 384, 32)$	12832
Conv2D	$(9, 384, 64)$	51264
Conv2D	$(9, 384, 128)$	204928
GAP	(128)	0
Dense	(1024)	132096
Dense	(512)	524800
Dense	(2)	1026
Trainable params	**Non-trainable params**	**Total params**
927.362	0	927.362

Table 6.1. Architecture and number of parameters for each layer of the adopted Attribution CNN. The "Output shape" column refers to a single-image input.

During the training procedure, the Adam optimizer [37] has been used with a *learning rate* of 0.001 for the Attribution CNN, while the TracIn CNN was trained with a SGD optimizer [31], with a *learning rate* of 0.01. The reasons behind the choice of using a different optimizer lie in the fact that the approximated version of the TracIn algorithm is thought to work with SGD; using Adam as an optimizer, being it not properly based on gradient descent, would imply a slight modification in the approach [48]. The CNN models were trained for 30 epochs dividing the dataset into 32-sized batches. The MAE loss function from Equation 5.2 was adopted and the efficiency and spread metrics (Equation 5.3 and Equation 5.4) were constantly monitored: a callback function was used to save the model scoring the highest efficiency on the validation dataset.

6.1.2 Convolutional Soft Decision Tree

The second class of our networks, the intrinsically-interpretable ConvSDT, was designed using *PyTorch*. Namely, inspired by and taking up the work of [2, 28], we constructed a Soft Decision Tree hierarchical structure positioned right after a

sequence of convolutional layers. Table 6.2 shows the resulting architecture of this model and its related parameters.

Starting from the convolutional part, the input image is given to three layers with respectively 64, 32, and 16 filters with a kernel size of (3×3). Between each pair of consecutive layers, we introduced a *Batch Normalization-ReLU-Max Pooling* routine to gradually reduce the dimensionality of the input image. This is motivated by the fact that (Soft) Decision Trees have often reported a decrease in performances with high-dimensional data [63]; with convolution, we have exploited its advantages with images and have simultaneously facilitated the training of the tree thanks to such dimensionality reduction process: from a $9 \times 384 = 3456$-dim input vector without convolution, we attain to decrease and simplify it to a flattened 768-dim vector.

In the second phase, this obtained 768-dim vector enters the SDT. Its hierarchical structure is a priori built by defining a fixed depth d of the tree, namely this also implies the presence of $(2^d - 1)$ inner nodes and of 2^d leaf nodes. By imposing $d = 5$, the decision tree is then modeled with two *Linear* layers with, respectively, $(768 \times (2^5 - 1)) = (768 \times 31)$ and $(2^d \times 2) = (32 \times 2)$ input-output features. The "inner" layer is followed by a *Sigmoid* activation function to simulate the binary splitting flow of the tree, while the "leaf" layer is initialized with a normal distribution, as experimented in [41]. The output of the first layer is manipulated to compute the path probabilities to reach every leaf: these probabilities are then used to "weight" the predictions of the leaves, namely the output of the second and last layer [28, 33].

During the training procedure, the Adam optimizer [37] with a *learning rate* of 0.001 has been used for the ConvSDTs. The ConvSDTs were trained for 5 epochs using a batch size of 64. The MAE loss function from Equation 5.2 was adopted and the efficiency and spread metrics (Equation 5.3 and Equation 5.4) were constantly monitored: a callback function was used to save the model scoring the highest efficiency on the validation dataset.

As mentioned at the beginning of this chapter, we have also experimented a

Layer	Output shape	# params
Conv2D	$(64, 9, 384)$	640
BatchNorm2D	$(64, 9, 384)$	128
ReLU	$(64, 9, 384)$	0
MaxPool2D	$(64, 4, 192)$	0
Conv2D	$(32, 4, 192)$	18464
BatchNorm2D	$(32, 4, 192)$	64
ReLU	$(32, 4, 192)$	0
MaxPool2D	$(32, 2, 96)$	0
Conv2D	$(16, 2, 96)$	$4,624$
BatchNorm2D	$(16, 2, 96)$	32
ReLU	$(16, 2, 96)$	0
MaxPool2D	$(16, 1, 48)$	0
Flatten	(768)	0
Linear	(31)	23839
Sigmoid	(31)	0
Linear	(2)	64
Trainable params	Non-trainable params	Total params
47.855	0	47.855

Table 6.2. Architecture and number of parameters for each layer of the adopted ConvSDT. The "Output shape" column refers to a single-image input.

re-training from scratch exploiting a simple *Offline Distillation* approach [30]. The procedure, already addressed in [28] and re-proposed in this work, is straightforward: instead of training the soft tree as usual, namely by using the actual labels as ground-truth values, the same hierarchical model has been trained replacing labels with the predictions generated from a teacher model, here the previously trained Attribution CNN. The resulting model will be denoted as *Distilled ConvSDT* in the rest of the discussion to differentiate it from the 'plain' version of the decision tree.

Comparing Table 6.1 and Table 6.2, we can notice a very important aspect differentiating CNNs and ConvSDTs. Indeed, the complexity of the tree-based model is incredibly low if compared to the CNN: just observing the different number of parameters in the two tables, we notice how ConvSDT's amount is more or less 20 times smaller than the one of CNNs for attribution methods.

6.2 xAI techniques adaptations

Let us explore the fine-tunings we experimented with and finally applied to our approaches: it is anyway worth mentioning that the changes have not involved all five approaches, some of them were already applicable to the HEP domain.

These modifications were required because, as described in the previous Chapter 5, the images composing our dataset are extremely different from standard images and therefore we had to re-design a bit the SOTA techniques to deal with our muonic pattern recognition task in a more appropriate way.

- The first attribution approach, Regression Activation Map [60], has turned out to be already suitable for our scenario. Two heatmaps for each input sample image are produced by our implementation: one about the transverse momentum and the other for the pseudorapidity, highlighting the discriminative regions for both labels separately.

- The IntGrad algorithm required some adaptations. The interpolation procedure, which aims at creating a sequence of images from a baseline to the original picture to be explained, proved to be the major aspect to be addressed. More specifically, being our images black-and-white they cannot be interpolated through the generic pixel-intensity approach as in Figure 3.2. We created a dedicated interpolation routine to produce a sequence of images such that, starting from an empty baseline with all zero-ed pixels and ending with the original image, the in-betweens are realized by iteratively adding one lit 1-valued pixel for each new interpolation in a top-down fashion.

  For instance, given a noisy image composed of $n = 18$ lit pixels, the resulting interpolation will instantiate a set of $n + 1 = 19$ images, including the empty baseline: Figure 6.1 expresses this example.

- Concluding attribution approaches, SmoothGrad has also required a variation, specifically in the noise introduction routine. Indeed, the original distribution

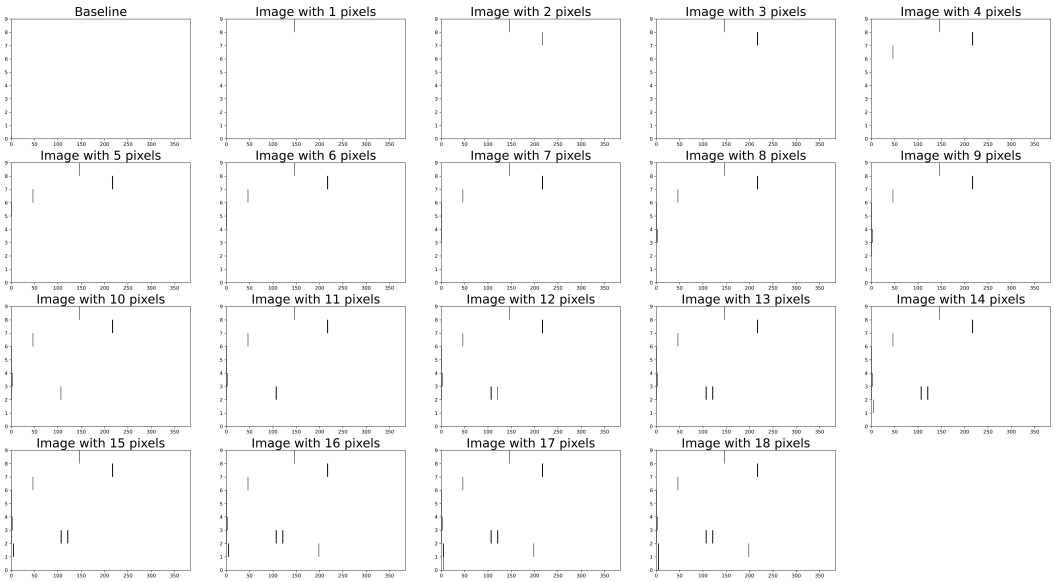


Figure 6.1. Example of an interpolation set. A collection of 19 images are generated from a sample with 18 active pixels.

used to regulate noisy addition is Gaussian (Equation 3.5) and it is therefore unsuitable for the binary images of this application. Considering the already present radioactive noises, we opted for replacing the Gaussian with a *salt-and-pepper* noise distribution to add 1s according to a predefined noise percentage σ. An example showing how this type of noise acts on our images is visible in Figure 6.2: it is evidently understandable how an increasing value of σ corresponds to a much more noisy image.

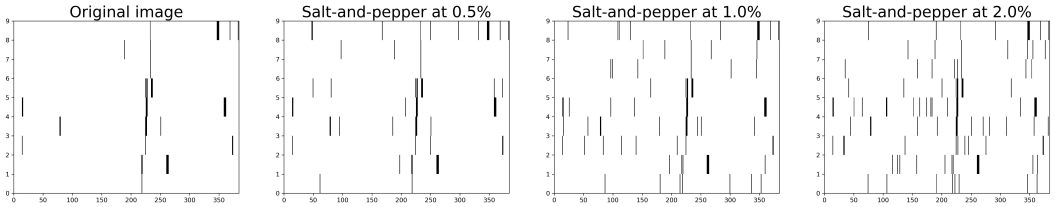


Figure 6.2. Examples of the effects of salt-and-pepper noise on an image using different noise percent $\sigma \in \{0.5\%, 1\%, 2\%\}$.

We carried out some tests to determine the best performing quantities used as attributes in this approach. The two hyper-parameters used by the method, the number of smoothing iterations (n) and the noise percent to add (σ), have in fact been chosen through a "visual inspection" of a Grid Search over

values for $n \in [10, 20, 50, 80, 100]$ and $\sigma \in [0.5\%, 1\%, 2\%, 5\%, 10\%]$. Evidence showed that setting $n = 100$ and $\sigma = 2\%$ represented, for our use-case, the best trade-off between the level of noise, computational cost, and quality of the explanatory results: Figure 6.3 displays the performed experiment which brought us to this conclusion.

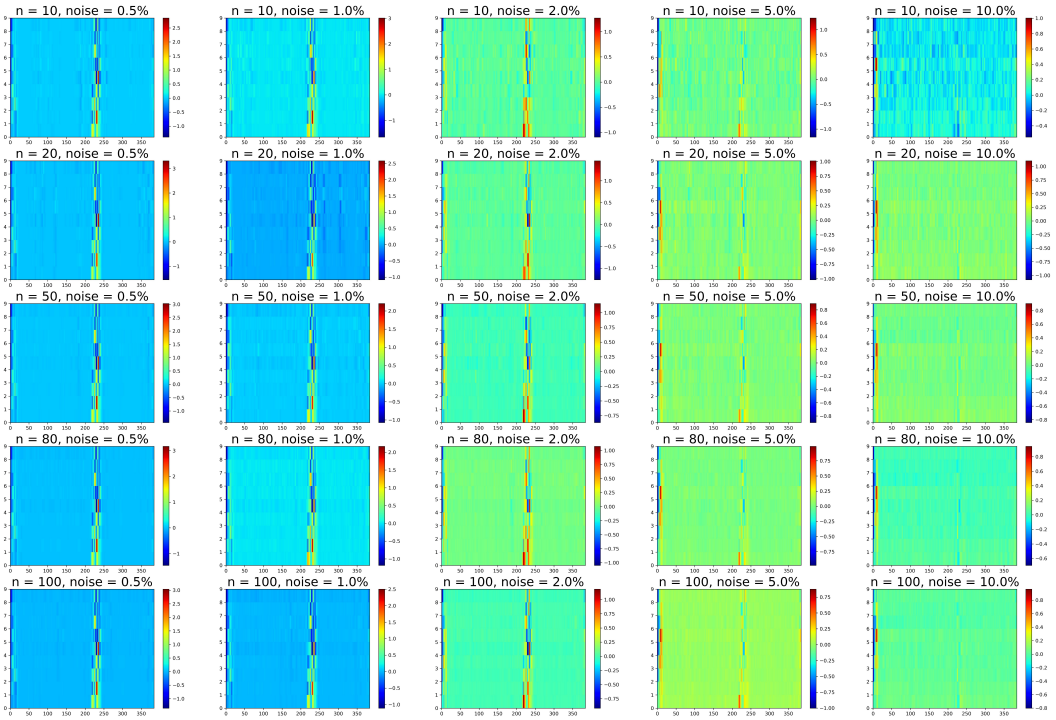


Figure 6.3. Example of the Grid Search used to tune SmoothGrad's hyper-parameters.

- ConvSDT did not require particular adaptations. This work provides a graphic visualization of their decision flow by re-implementing the typical hierarchical drawing of Decision Trees inspired by the Scikit-learn library [45] and adapting it to sketch the decisions of each node from the root to the leaves with their associated probabilities, similarly as done in [28] but with augmented visual information and insights relative to the model's functioning.

- Finally, the only contributions to the TracIn method concerned the choice of which checkpoints to save and use in the approximated version of the approach. The original paper [48] recommends selecting checkpoints having relevant loss

improvements with respect to the temporally adjacent epochs: the picked saved models have been chosen according to this criterion. After careful evaluations, we selected 3 models among the training ones, picked uniformly in the interval of the train epochs: the final choice consisted of checkpoints occurring at epochs 5, 15, and 25.

Chapter 7

Results and Discussion

In this final chapter, I will discuss the performance results achieved corresponding to both the regression and the explainability tasks.

7.1 Hardware

When it comes to training DNNs, and in particular CNNs, it is generally well known that Graphics Processing Units (GPUs) can achieve significantly faster executions in comparison to Central Processing Units (CPUs) because of their larger number of logical cores [44].

For this reason, at first, we utilized the GPU component while training both our architectures. Nevertheless, we noticed that a CPU-training for the ConvSDT model was faster than a GPU one, due to the peculiarity of its hierarchical architecture. Because of this, we decided to keep using the GPU only while operating with the CNN and, instead, to use the CPU for the ConvSDT.

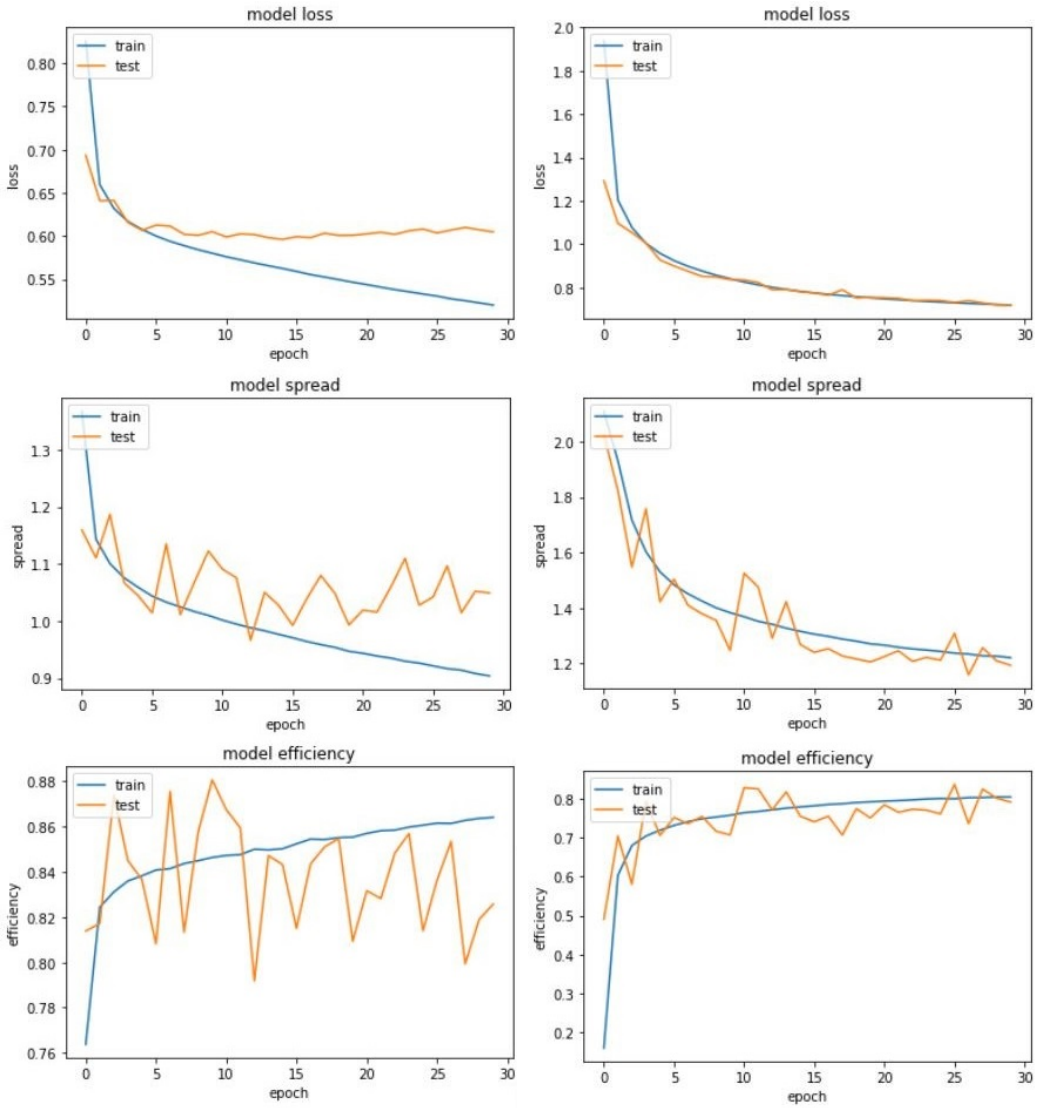
The hardware used to execute the experiments employs the following components:

- **GPU:** NVIDIA® GeForce® RTX 2070 SUPER™ 8GB GDDR6;
- **CPU:** Intel® Core™ i7-10875H Processor 2.3 GHz (16M Cache, up to 5.1 GHz, 8 cores);
- **RAM:** 16GB.

Exploiting this setup, a standard training epoch for the CNNs extended approximately around 9 minutes, while for the ConvSDTs 50 minutes.

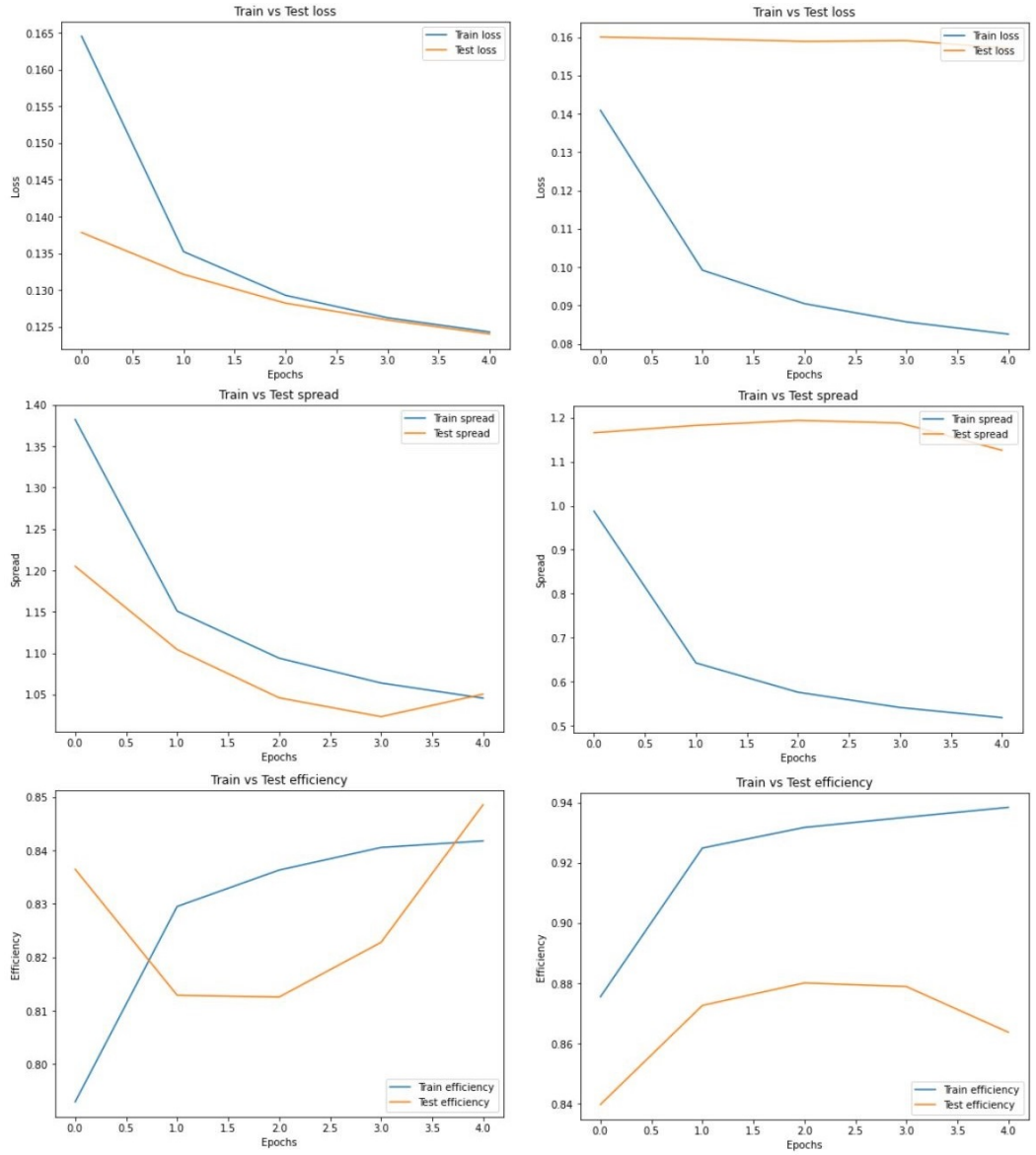
7.2 Regression stage

All the training processes have been executed successfully as we will analyze in this section. Moreover, as an additional monitoring system, these procedures have been graphically plotted in Figure 7.1.



(a) Attribution CNN's plots.

(b) TracIn CNN's plots.



(c) ConvSDT's plots.

(d) Distilled ConvSDT's plots.

Figure 7.1. Training plots for every model.

These plots show our metrics' values computed on both the training and validation sets and from them we can derive that the loss and spread functions consistently and correctly decreased while the efficiency increased.

7.2.1 Regression main task

After training, each generated model has been evaluated on the aforementioned validation dataset made up of ≈ 95k samples. The results for the regression task are measured through the MAE loss, the spread, and the efficiency.

Before starting the main discussion, I would quickly mention the other models, and their associated results, that did not excel enough in our tasks but that have been the base on which the currently discussed models were built. It is very important to show how our research, which lasted about one year, actually evolved: dozens of different networks were trained and tested before reaching the final results. Table 7.1 shows a selection of them and, in particular, it shows our experiments on ensembles of architectures (a topic that was not extensively commented on in the thesis).

Regarding the models' architectural structure, standard Decision Trees have been trained without enforcing a depth limit while every NDT and SDT share the same max d that has been later inherited by ConvSDT, i.e. $d = 5$: this imposed limit is only due to the computational capabilities of our hardware. Similarly, we could not fully explore the loss function possibilities for tree architectures because of the greater time complexity of *Mean Squared Error* (MSE) compared to MAE [29]. All the distilled networks were trained using the Attribution CNN as a teacher. Moreover, an ensemble of SDT was composed of 6 SDTs whereas a "big" ensemble had 15 of them.

Model	Spread	Efficiency
DecTree	3.4228	0.5236
Distilled DecTree	3.4249	0.6194
NDT	2.4248	0.6155
SDT	2.6006	0.5612
SDT Ensemble	2.1619	0.5819
Distilled SDT	2.7082	0.6392
Distilled SDT Ensemble	2.1593	0.6443
Big Distilled SDT Ensemble	2.0421	0.6521

Table 7.1. Evaluation Metrics for originally implemented models: rejected because of their low validation efficiency.

Let us begin the dissertation with the main results, collected in Table 7.2, containing the evaluations for all four tested models.

Model	Loss	Spread	Efficiency
Attribution CNN	0.6051	1.1227	0.8807
ConvSDT	**0.1240**	**1.0505**	0.8489
Distilled ConvSDT	0.1589	1.2688	**0.8864**
TracIn CNN	0.7315	1.3093	0.8367

Table 7.2. Evaluation Metrics for our models. Loss values are considerably lower for ConvSDTs due to the pre-processing label normalization operation.

- Loss: As explained in the caption of Table 7.2, we can compare this metric only between models belonging to the same architecture. As expected because of its better general performances, Attribution CNN recorded a substantially lower MAE than TracIn CNN. On the opposite, considering the ConvSDT architectures, this correspondence does not show again: it seems that Knowledge Distillation, while anyway helping the efficiency metric, gave instead no benefits to the loss.

- Spread: About the spread value, every model registered satisfactory achievements. The best result for this evaluative measure was obtained by ConvSDT with a validation spread of 1.0505. The real significance of this result can be truly understood by considering that the spread value can be hardly minimized under 1.0 GeV. The reasons behind this fact are that even the ATLAS detector itself, with which data has been collected, has an intrinsic resolution of approximately $1 - 2$ GeV in our interval of interest around 10 GeV of momentum. For this reason, it is confirmed the high capacity of our ML algorithms to predict optimal metrics despite the intrinsic physical limitations of our particle trajectory tracking task.

- Efficiency: Finally, we examine the efficiency value, the most important among the test quantities of this study. Again, Attribution CNN is certified to be one

of our most performing models, with an efficiency considerably higher than ConvSDT and TracIn CNN. However, the greatest remark for this part of the dissertation is about KD: the Distilled ConvSDT student was in fact able to reach and even surpass the Attribution CNN teacher, reporting a consistent increase of +3.75% (+0.57% with reference to the teacher), resulting in a final efficiency of 88.64% .

7.2.2 Regression sub-task

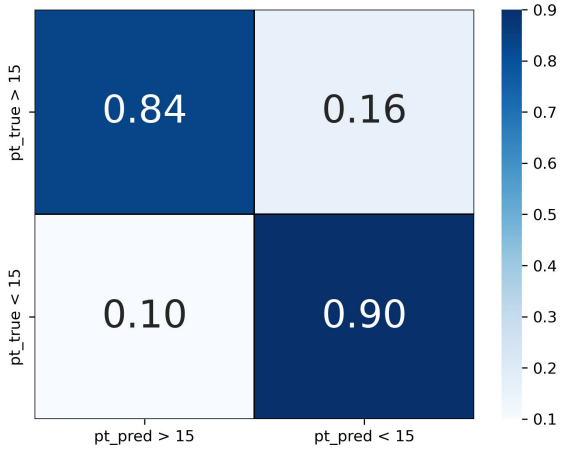
Focusing now on the secondary binary classification task, we can analyze the rate of correct selection/rejection of muonic patterns that the use of our models in the ATLAS's Trigger would produce. Three confusion matrices have been generated for this purpose, one for each model except for the *undistilled* ConvSDT, and they are visible in Figure 7.2. In particular, we decided to show them in a row-normalized fashion: each partition indicates the percentage of events belonging therein and, as previously defined, True Positives are placed in the upper left sector, True Negatives in the lower right, False Negatives are situated in the upper right and False Positives in the lower left.

Some observations that can be produced by looking at the confusion matrices are about the strengths of each specific architecture: the following can help users with their model's choice according to what is the quantity they prefer to minimize for their achievements. CNNs seem to minimize the number of FP events but at the price of increasing FN, while instead Distilled ConvSDT has the opposite effect, reducing the FN and increasing FP.

Anyway, even if there are these slight differences in the FN and FP amounts between the three models, it is significant to mention that they all are in the end very similar and moreover the selection/rejection rates are comparable. This demonstrates the robustness of all our results even despite considering that the data were greatly unbalanced providing fewer examples of positive events, namely images with $p_T > 15$ GeV.

(a) Attribution CNN.

(b) Distilled ConvSDT.



(c) TracIn CNN.

Figure 7.2. Row-normalized confusion matrices for the regression sub-task. They show the rates of correct/wrong selected/rejected events at the selected threshold of 15 GeV.

We also produced a more complete classification report, showed in Table 7.3, with the intent of evaluating the sub-task more profoundly[1]. Here, we can notice how the confusion matrices' rates are again found in the Recall metric for each model. The most relevant observation we can perform relates to the last statement I did before the introduction of this table: while the recall metrics are quite comparable among them, instead the precision and F1-score differ more clearly between selected/rejected events. This is undoubtedly because of the lower populousness of the class of muonic traces with high p_T (i.e. the "selected" class). Moreover, the table reiterates the previous statements showing that the most performing model is found to be the Attribution CNN with 90%, 89%, and 89% of, respectively, averaged Precision, Recall and F1-score, immediately followed by the Distilled ConvSDT.

Model		Precision	Recall	F1-score	Support
Attribution CNN	Selected	**0.79**	0.88	**0.83**	28527
	Rejected	**0.95**	**0.90**	**0.92**	65918
	Accuracy	-	-	**0.89**	94445
	Macro avg	**0.87**	**0.89**	**0.88**	94445
	Weighted avg	**0.90**	**0.89**	**0.89**	94445
Distilled ConvSDT	Selected	0.77	**0.89**	**0.83**	28527
	Rejected	**0.95**	0.88	0.91	65918
	Accuracy	-	-	**0.89**	94445
	Macro avg	0.86	**0.89**	0.87	94445
	Weighted avg	**0.90**	0.88	**0.89**	94445
TracIn CNN	Selected	0.78	0.84	0.81	28527
	Rejected	0.93	**0.90**	0.91	65918
	Accuracy	-	-	0.88	94445
	Macro avg	0.86	0.87	0.86	94445
	Weighted avg	0.88	0.88	0.88	94445

Table 7.3. Classification report for the sub-task. Metrics are displayed according to the two classes together with their means.

[1]The Macro average and the Weighted average shown in the table refer respectively to an arithmetical and a weighted mean.

7.2.3 Regression alternative task

Until now, our efficiency metric has been strictly related to the 15 GeV threshold: let us introduce some variations. For instance, we could consider the recent related work of [27] where an application was developed studying muonic patterns and focusing on generating real-time models through compression and distillation techniques (as we also did in this work). In particular, the main goal of the authors was to develop networks that could be efficiently executed on Field-Programmable Gate Arrays (FPGAs), a specific hardware architecture usually adopted in place of GPUs to perform heavy parallel computing with low power consumption [42]. Anyway, in this paper the definition of the efficiency metric is given with a little difference with respect to the one used in this thesis: the threshold of interest is in fact moved and reduced from 15 GeV to 10 GeV. Equation 5.4 can be accordingly adjusted into an alternative formulation where $E'_{num} = \{e \mid y_{p_T,e} > 10 \text{ GeV} \wedge \hat{y}_{p_T,e} > 10 \text{ GeV}\}$ and $E'_{den} = \{e \mid y_{p_T,e} > 10 \text{ GeV}\}$ and:

$$\text{efficiency}_{alt}(E'_{num}, E'_{den}) = \frac{\#(E'_{num})}{\#(E'_{den})} \tag{7.1}$$

Lowering the threshold in this way automatically results in a more relaxed task, and it is fascinating to notice the improvements in efficiency adopting its alternative version, grouped in Table 7.4. This supplementary set of results testifies how much the imposed threshold is relevant for the selection/rejection task: the quantity generally increases by about 10% for each model. The efficiency results stay consistent with the one of Table 7.2: namely, the "ranking" of networks stays the same even if the efficiency gap between the worst/best model is reduced to $\approx 1-2$. Anyway, the obtained values are incredible: the best model is again Distilled ConvSDT which scores a 96.17% efficiency demonstrating that, for some reason, the beneficial effects of KD are reduced to only an increase of +1.17%.

Model	$\text{Efficiency}_{\text{alt}}$
Attribution CNN	0.9579
ConvSDT	0.9500
Distilled ConvSDT	**0.9617**
TracIn CNN	0.9427

Table 7.4. Efficiency evaluation metric for its alternative definition: the threshold is moved to 10 GeV from the original 15 GeV.

7.2.4 Regression of only-noise dataset

We conclude the regression analysis considering the predictive capacities of our models on the second dataset, namely the set of images containing only noise hits and no muonic traces. These images have zeroed momentum (and pseudorapidity), therefore we decided to focus on the samples that our networks selected, namely the ones classified above our usual 15 GeV limit: the following describes the additional insights on our results we managed to extract from them.

Despite the considerable size of 944448 images of this dataset, the (wrong) selection rates achieved are expressed in Table 7.5.

Model	#	Wrong %
Attribution CNN	1389	0.1471
Distilled ConvSDT	2	0.0002
TracIn CNN	11	0.0012

Table 7.5. Selection rates of only-noise samples. The second column specifies how many of the 944448 images were selected and the third column the respective percentage.

These results are extremely remarkable: indeed, the requirement of the project demanded just for the fake rate to be $< 0.2\%$. This restriction was required because the fraction of erroneously selected only-noise events of the real-time muonic selection system of the ATLAS experiment corresponds exactly to this amount: at the LHC, collision events occur every 25 ns, equivalent to a frequency of 40 MHz. The 0.2% of this frequency corresponds to a rate of 80 kHz, which is precisely the maximum frequency of erroneous selections admitted.

This imposed threshold is more than perfectly satisfied by every model. Only Attribution CNN, which generally behaves really well, results to be the worst performing network in this sub-task, albeit within the required limit.

We should remind the readers that the first dataset used for training did not contain any only-noise image and, therefore, our models did not expect this kind of input. For this reason, this result is particularly positive. In the next section focusing on explainability, we will go back on these noisy images predicted with a high momentum to try to understand the causes of such erroneous predictions to then reduce them.

Summarizing this first set of results about the regression task, Attribution CNN and Distilled ConvSDT proved to be the best models providing some acceptable compromises between a minimized spread and a maximized efficiency. The plain version of the ConvSDT guarantees the most reliable spread but it lacks efficiency and, finally, TracIn CNN is the worst model among the four. However, these results are not unexpected if considering the architecture of the network: the single-layer regression head is strongly weakened compared to the head of Attribution CNN composed of three high-dimensional layers. ConvSDT-based architectures are in theory weakened as well: their number of parameters is almost 20 times smaller than the CNN and, about memory storage, a saved checkpoint of Attribution CNN takes 10.6 MB of memory space, more than 55 times bigger than the 196 KB needed for the ConvSDT checkpoint. Nevertheless, thanks to their design and KD, ConvSDTs perform perfectly making them preferable for real-time applications.

All the implemented networks have reported specific strengths for this HEP application, and they can be convergently considered to get some explanation of predictions already at this regression stage. All networks achieved more than satisfactory efficiency results, which become even better when considering the alternative definition of efficiency with the threshold at 10 GeV.

7.3 Explainability stage

After discussing the high-level results of the regression task through statistical measures, we can now begin the last important analysis about how effectively our models can provide explanations and justifications for their decisions: even if our networks empirically proved to give robust predictions, users do not have yet the possibility to fully understand the reasons behind how a muonic pattern with some random noisy hits can be translated by the ML algorithms in our two labels.

In this section, we will begin using our xAI instruments. However, it often happens that different xAI approaches may output different (or opposed) explanations and this discrepancy usually results in difficulties in grasping an official interpretation of the prediction. We tried to find strengths in this heterogeneity of approaches by extracting for our use-case some easy-to-understand and human-readable explanations from each one of them. Moreover, also the drawbacks related to this particular scenario will be handled. In a nutshell, we developed a framework for a convergent application of multiple xAI algorithms in a realistic context.

To stay true to this last statement, each of the proposed xAI techniques is implemented to address various categories of end-users and explanations will be provided in different formats. This section contains a lot of graphical representations of our generated and proposed explanations: inside them, we also show the denoised version for each considered sample. These images are here introduced only to highlight the position of pixels belonging to real particle trajectories and, as we already explained for Figure 5.2, we remark that the denoised images have never been part of the training stage.

The following discussion is subdivided into the three classes of related xAI techniques and it will primarily focus on the explanation's differences related to the TP/TN/FP/FN categories and the peculiarity of the ones addressing only-noise images. We mainly focus on the momentum label, nevertheless, we must notice that the pseudorapidities have been addressed and forecasted very accurately as well.

7.3.1 Saliency Maps Methods

Let us begin with the explainability results of our first architecture: Attribution CNN. The explainability study we performed focused on the set of attribution-based approaches (RAM, IntGrad, and SmoothGrad) which investigate the model's predictions by finding the most relevant discriminative regions present in some assessed sample image.

Every explanation we created for this first set displays, in addition to the real and predicted labels, ten images placed in two rows and five columns. Starting from the leftmost column, we observe:

- The denoised input image and the same image but with noise passed to the model to be predicted and explained;
- The feature heatmap outputted using the RAM method on the η label and its superimposition with the input image with noise;
- The feature heatmap outputted using the RAM method on the p_T label and its superimposition with the input image with noise;
- The feature heatmap outputted using the IntGrad method and its superimposition with the input image with noise;
- The feature heatmap outputted using the SmoothGrad method and its superimposition with the input image with noise.

Color bars have been placed next to heatmaps; they clarify, for each pixel, its relevance value, namely how much significance it had in the determination of the model's decisions. This attributed value can be either positive or negative and its actual importance only depends on its modulus. The background and other generic uninfluential areas are displayed with the color corresponding to the 0 value.

Looking at Figure 7.3, which reports two representative instances of TP and TN, we can start the analysis on the correctly estimated images.

The former is illustrated by Figure 7.3a and it consists of a pattern with high p_T characterized by an almost straight-lined pattern with $p_T \approx 19$ GeV and predicted $p_T \approx 18$ GeV. We can distinctly see that, in this case, the model selects, for every implemented xAI method, the right muonic pattern by highlighting only pixels that actually belong to the trace of muonic particles. Focusing again on the images in the second row of the explanation, we see that it simultaneously succeeds in discarding most of the noise, behaving as a sort of noise reducer. The most discriminative regions enhanced through our approaches in these images are indeed the ones containing the actual trajectory of the particle and not even one noisy pixel was selected meaning that Attribution CNN is perfectly ignoring artifacts present in the image and focusing on the correct areas where muon has passed through in the detector. In this way, we can safely state that the related predictions of the CNN are unobjectionably intended to be trustworthy.

Now, taking into consideration Figure 7.3b, depicting a TN pattern with real momentum of $\approx$ 14 GeV and predicted momentum of $\approx$ 13 GeV[2]. An analysis similar to the one of TP can be performed: the pixels with the most heat do not include pixels do not include background or noise sections but only the real path. Again, the user can be easily convinced of the capability of Attribution CNN in the selection/rejection task using the superimposed images as proof.

It is also important to notice how the comprehension of the interpretation is facilitated and the methods are potentially targeted to all types of end-users: scientists analyzing this xAI outcome do not need particular knowledge both in ML or in HEP, they have simply to observe which pixels are highlighted to understand the decisions' reasons.

On the opposite, in the wrong prediction (FP and FN) showed in Figure 7.4, we can see that for the wrong predictions the model fails in its denoising operation because it highlights additional and noised pixels that should be instead zeroed:

[2]These associated momenta are still high ones, nevertheless below the 15 GeV threshold. These kinds of images having a p_T near the imposed limit are the most relevant being the easiest to be misclassified.

consequently, the associated predictions are not so reliable as well.

Analyzing both the FP and FN instances (Figure 7.4a and Figure 7.4b), we can conclude that there are some strange muonic behaviors taking place there, probably due to malfunctioning in the spectrometer detection mechanism, making the trajectory extremely difficult to recognize. In fact, for the FP sample, the observations of layers 4, 5, and 6 of the RPC chambers are missing while, for the FN one, layers 3, 4, 7, 8, and 9 are not present. These data acquisition issues cause bad predictions for them: the first one, a low-momentum sample with $p_T \approx 9$ GeV, is estimated at $p_T \approx 19$ GeV; FN's momentum label is high, $p_T \approx 19$ GeV, but the model output $p_T \approx 7$ GeV.

Observing the related generated heatmaps, we can again find explanations for these wrong regression results: the model has in fact marked some adjacent background pixels with a high importance value, while clearly they should be instead zeroed. This is a very interesting property: it looks like Attribution CNN is trying to reconstruct the unrecorded parts of the trajectory by giving importance to pixels between partial observations, resulting in highlighting multiple vertical lines. However, doing so also increases the confusion in the image and causes erroneous output. Nevertheless, even if the produced prediction is not correct, the superimposed images show that the model is trying to reconstruct a path only building on the uncompleted real muonic traces, while still being able to completely discard the noises.

These two wrong-predicted samples tell us something very important. The ML models we developed are of course not perfect and they should not be blindly followed but, for these cases, our analysis demonstrated that the fallacy is not fully coming from a network's fault but mostly from the underlying data: for this reason, HEP researchers can still derive useful insights by observing the incorrectly generated tracks of important pixels. Moreover, correlating them with the correct patterns can give us a better understanding of the peculiarities of this use-case therefore potentially leading us to improvements both in the network and in the data.

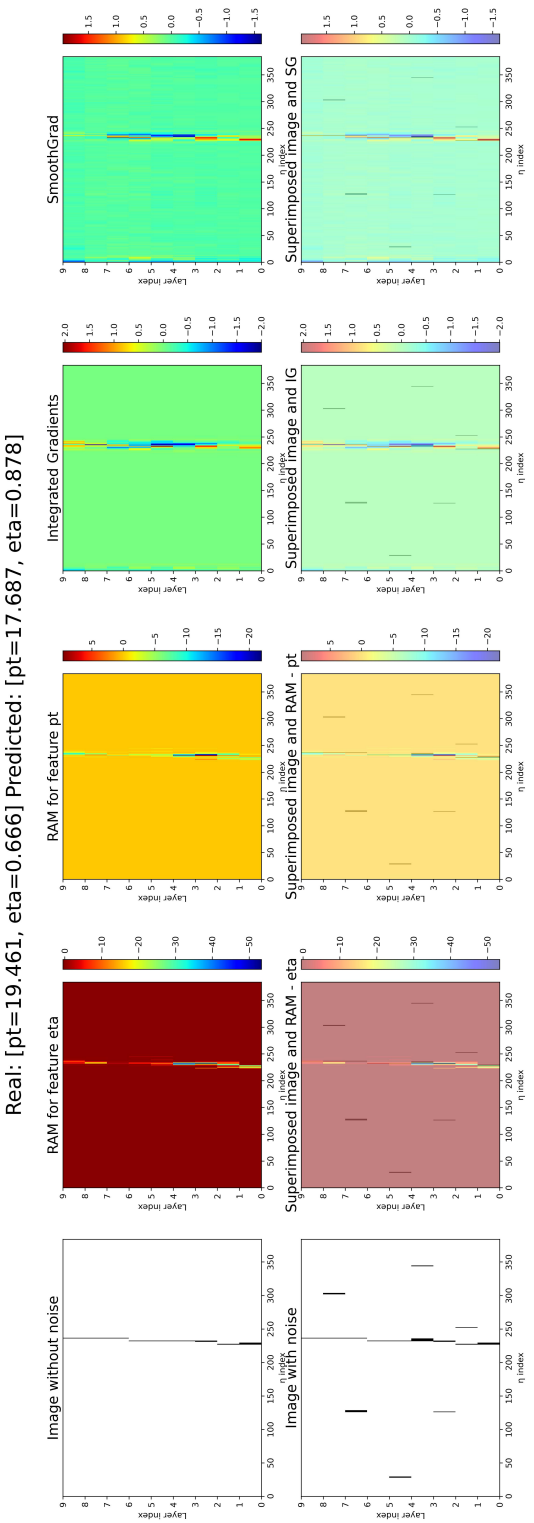
Switching to the test related to only-noise images (Figure 7.5), recalling that only-noise samples have 0 GeV as the label, we here propose two cases: when the model wrongly predicts a high momentum, $p_T > 15$ GeV, or a low one.

We get some interesting observations from the first case in Figure 7.5a: every attribution-based method believes that there is a real straight pattern along the left side, probably because of two noisy pixels nearby. This gives an explanation for both the predicted labels: the supposed pattern is an almost straight line, motivating the high predicted momentum of ≈ 16 GeV, and the line is along the left border, motivating the ≈ 0 outputted pseudorapidity.

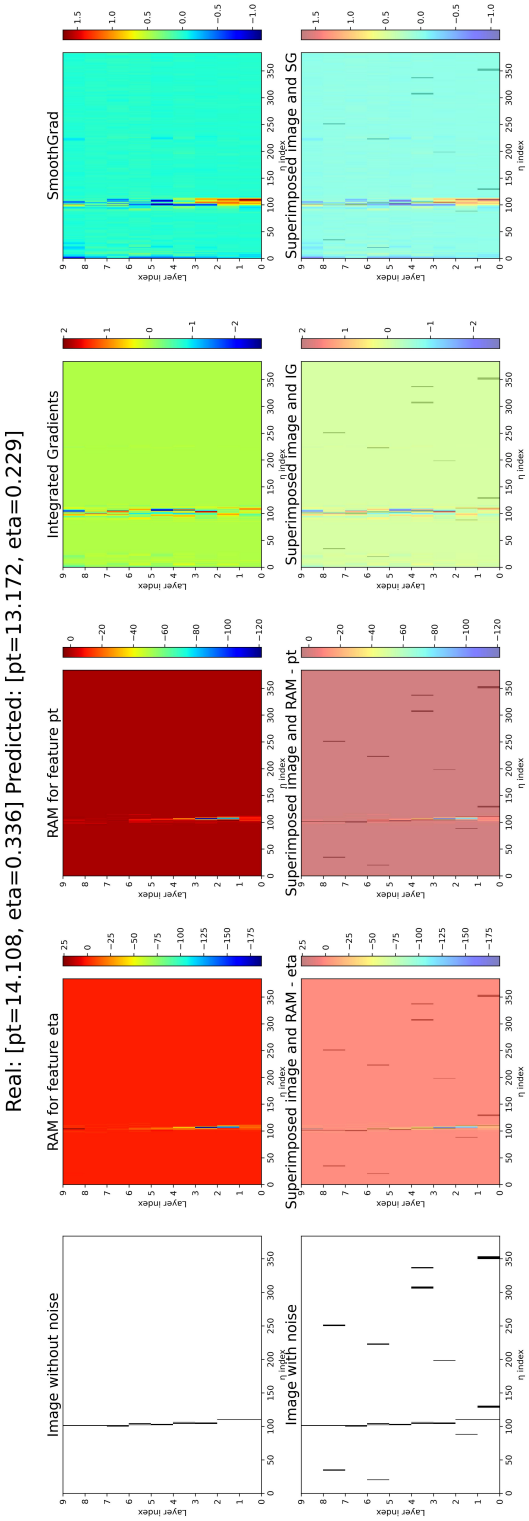
The second proposed sample in Figure 7.5b provides us once again some more knowledge. Here, the prediction of the image results in a $p_T \approx 2.4$ GeV, namely deriving that this is a properly rejected event. The explanation here is extremely eloquent: in fact, it shows that Attribution CNN is absolutely (and correctly) unable to find a muonic pattern. It just assigns minor importance to a small number of pixels in the image, but without generating a continuous line.

Generally speaking, all the resulting attribution-based explanations, although having slight differences among them, prove to be valid. Anyway, RAM's outputs can usually obtain muonic patterns thinner than IntGrad's and SmoothGrad's ones. However, SmoothGrad's explanations are, tautologically given its definition, an IntGrad improvement and turn out to be the best noise-reducing method.

Finally, we would like to point out that our model and our results appear to be both reliable and robust despite the work presented by [26]. This paper claim that, theoretically, no attribution-based method can guarantee high-quality explanations for models that are also achieving high performances in their related task: anyway, we analytically showed how it is not the case for our HEP use-case where these xAI techniques were capable to correctly enhance real patterns in correct predictions and to highlight additional perturbations in wrong decisions for a highly efficient DL network.

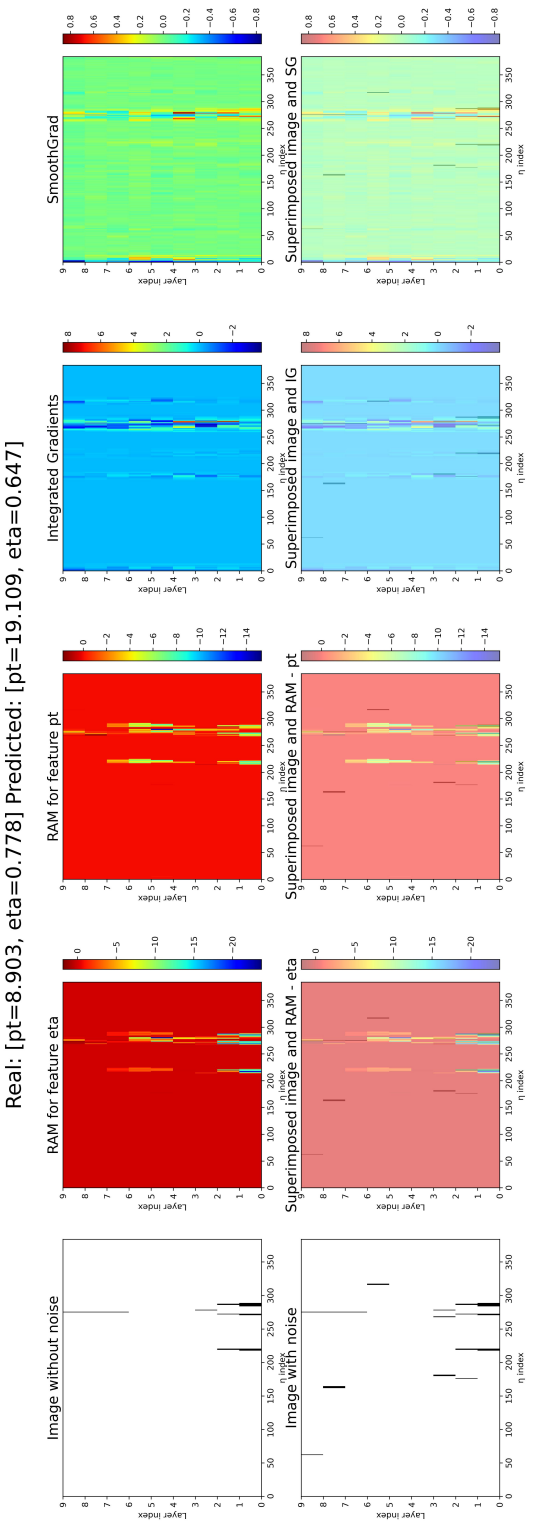


(a) True Positive example.

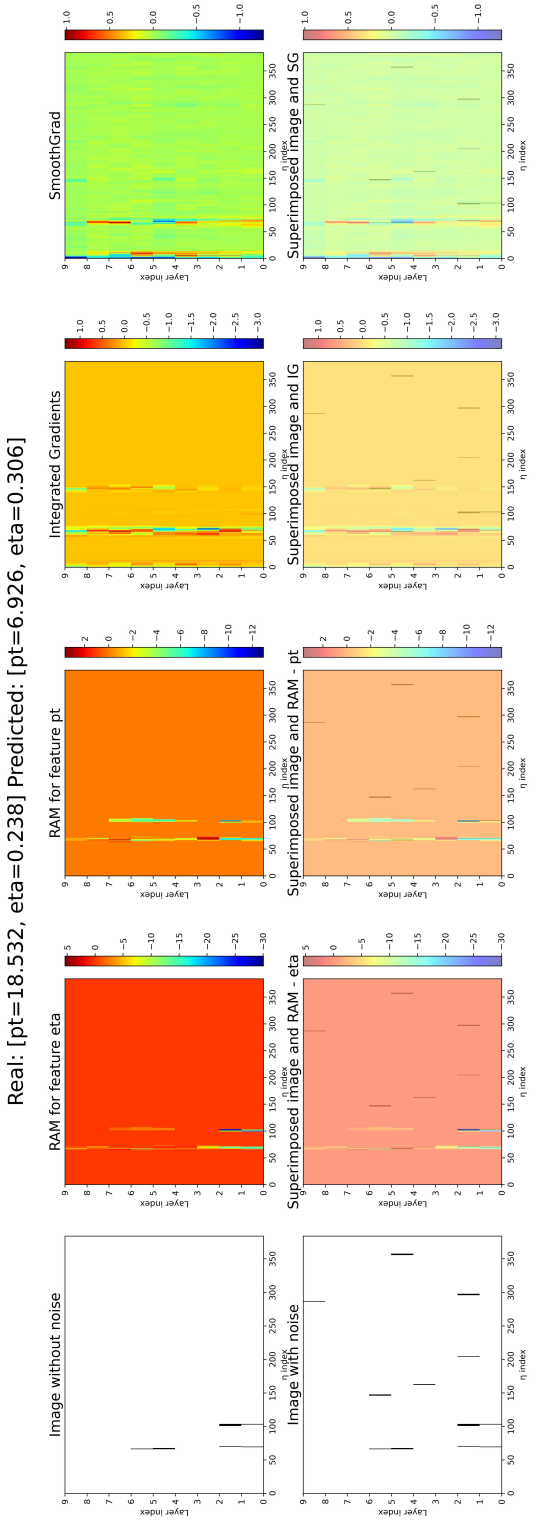


(b) True Negative example.

Figure 7.3. Attribution-based explanations for correctly predicted muonic patterns.

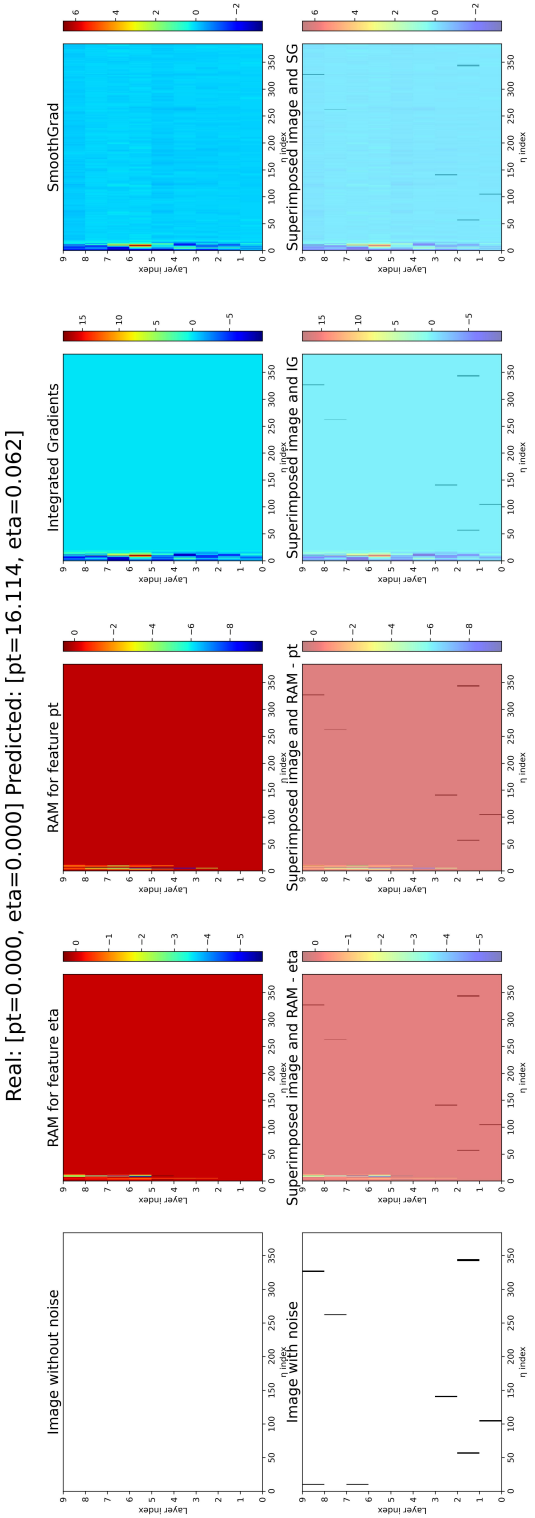


(a) False Positive example.

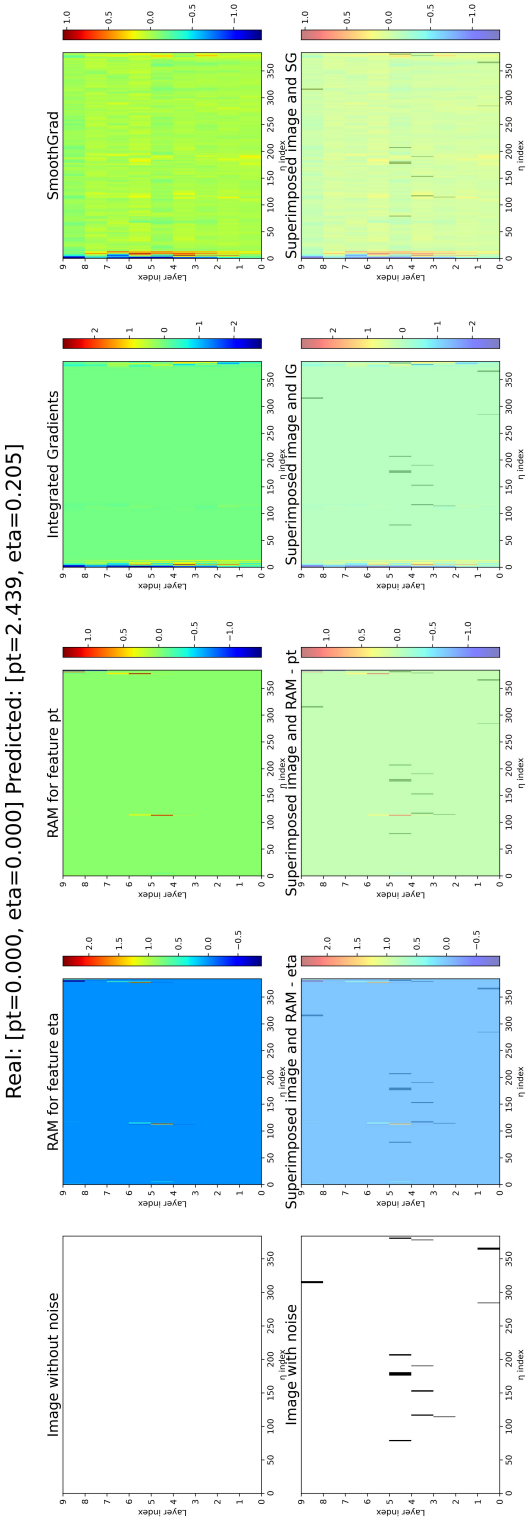


(b) False Negative example.

Figure 7.4. Attribution-based explanations for wrongly predicted muonic patterns.



(a) Only-noise example with high predicted momentum.



(b) Only-noise example with low predicted momentum.

Figure 7.5. Attribution-based explanations for only-noise muonic patterns.

7.3.2 Intrinsically-interpretable models

Moving to explanations derivable from the ConvSDT architecture, as is the regression stage discussion we will only consider results pulled out of the Distilled ConvSDT network. The interpretability provided by the hierarchical structure of the ConvSDT allows the user to easily and intuitively understand the model's decision flow. Our implementation generates a graphical visualization to render the explanation sustaining a prediction, for this we took inspiration from the work of [28] which produced the image previously shown in Figure 3.6.

Our visualization function portrays the phases of the process applied to the input image from its passage through the convolutional layers to its crossing along the depth levels of the tree. Each inner node is represented as a 2D grid showing the correlation between the kernel of the multi-layer perceptron and the input and is equipped with a color bar to qualify the positive and negative correlation influence of the node's pixels/weights. The leaf nodes, instead, are characterized by the probability of reaching them together with the associated predictions of p_T and η to be weighted accordingly. Moreover, there will exist a *maximum probability path*, marked in green, leading to the leaf with the highest and heaviest contribution and other $2^d - 1 = 2^5 - 1 = 31$ red paths with lower probabilities whose leaves' predictions will act as a refinement to the final prediction. In this way, the possibility to explain is given in a double way: leaves quantify the probability as confidence in predicting certain values rather than others, while inner nodes certify that such confidence is based on coherent pixels or areas of the correspondent correlation images.

Each explanation image in this category, other than the top-down visualization of the complete decisional mechanism of the tree, depicts in the top left the input image together with its denoised rendering, for the same reasons presented during the discussion of attribution methods' results. Then, also the feature maps of the three convolutional layers on top of the SDT are subsequently shown.

Some correctly and wrongly selected samples are respectively shown in Figure 7.6 and Figure 7.7.

Starting from the selected TP case, depicted in Figure 7.6a, the input has a straight pattern on the left side of the image with a high momentum of $p_T = 17.273$ GeV. Giving this image to Distilled ConvSDT, it correctly predicted $p_T \approx 18$ GeV: let us inspect the explanation. The model correctly identifies a strong 77%-maximum probability path leading to the most contributing leaf which outputs a momentum of 18.77 GeV and a $\eta = 0.23$, both already close to the ground-truths. The high probability of this leaf node suggests that the prediction is performed with high confidence around the label values therein contained. Anyway, exploiting the "softness" property of this architecture, a smaller contribution is still taken from the other leaves and weighted to their corresponding percentage to get a final momentum prediction of $p_T = 17.899$ GeV: in this way, the tree was able to come even closer to the real p_T value. For instance, the leaf node with id 21 with its percentage of $\approx 6\%$ is the second most influencing node: nevertheless, it has an associated 10 GeV, effectively lowering the value expressed by the maximum probability path.

Examining the kernel images of the inner nodes composing the maximum probability path (i.e. nodes with ids 0, 1, 3, 7, and 15), we can notice that they highlight a positive correlation near pixels that are close to the actual pattern of the muon in the input image, implying that the tree is able to detect the particle movement even in the high presence of noise such as in this test sample. Thanks to this and the maximum probability path notion, users can understand the correctness of the predictions produced by Distilled ConvSDT.

Observing the TN sample in Figure 7.6b, it contains a very curved pattern with a low real momentum of ≈ 4 GeV. The maximum probability path to the architecture's leaf has again a high confidence of 71% leading to a "hard" output of 2.4 GeV for momentum and of ≈ 1 for pseudorapidity. As before, the influences of the other leaves refine the prediction to $p_T \approx 3.5$ GeV and therefore makes it closer to the real label. Once again, inspecting the learned filters we see that the model was able to reconstruct the muonic pattern resulting in high-reliability decisions of Distilled ConvSDT.

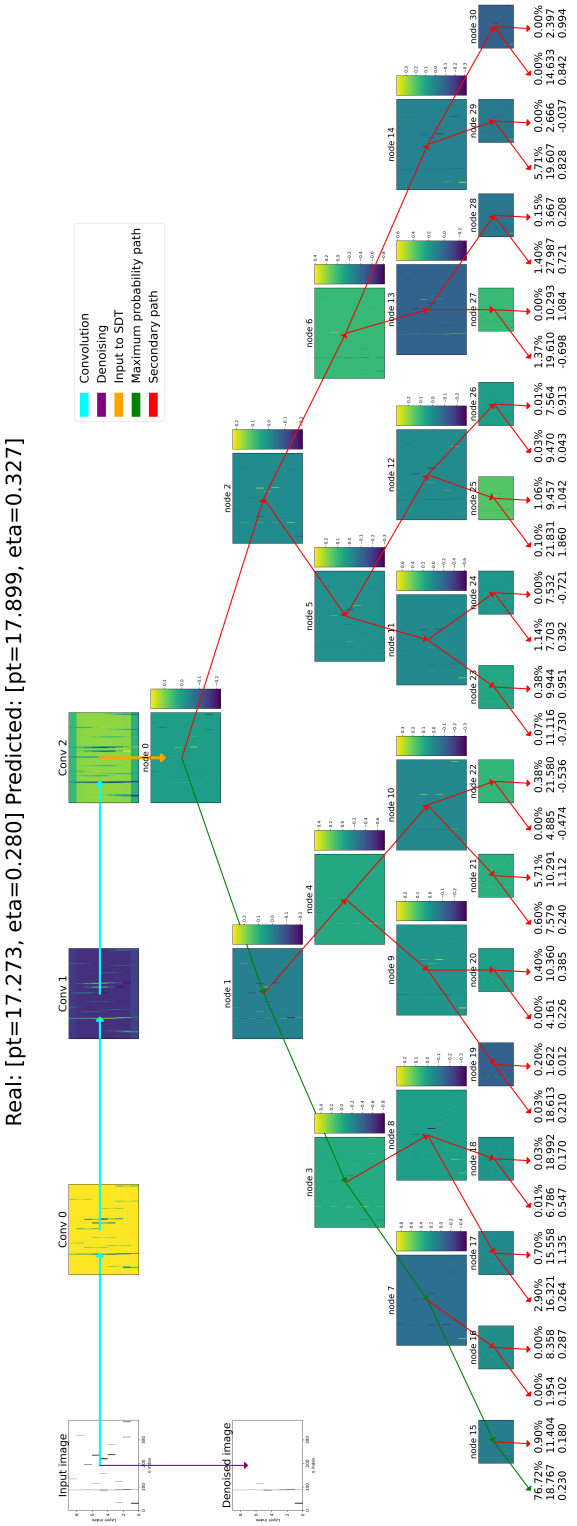
On the other hand, we can consider the wrong predictions pictured in Figure 7.7. The FP image in Figure 7.7a depicts a curved particle's trajectory, it has a $p_T \approx 6$ GeV and a predicted momentum of ≈ 17.5 GeV. The user can immediately visualize that the kernels of the inner nodes along the maximum probability path are positively correlated only in the occurrence of noise, while the actual trajectory of the particle is often negatively correlated. This means that the predictions of the associated leaf are mainly influenced by artifacts and not by the real muonic pattern: the predicted momentum reached in the leaf node is indeed 19.61 GeV which is way far from the true value. The percentage here associated with the maximum probability path is 54%, denoting how the tree model gave a wrong answer even if it had average confidence, and moreover, it could not be helped by the lower paths. As before in the saliency maps methods, the pattern of this image is partially lost between layers 1 and 4 of the RPC chambers, meaning that only the second half of the trajectory is present, leading to the error.

The FN sample in Figure 7.7b depicts a high-momentum pattern with ≈ 19 GeV but with a low 5 GeV-prediction. This is an even greater mistake by Distilled ConvSDT: the confidence is higher (62%) for a predicted 2.4 GeV and the filters highlight noises while the denoised image is not missing any relevant data. Therefore, in this case, the faulty prediction is solely imputable to imperfections of the model and the radioactive noise.

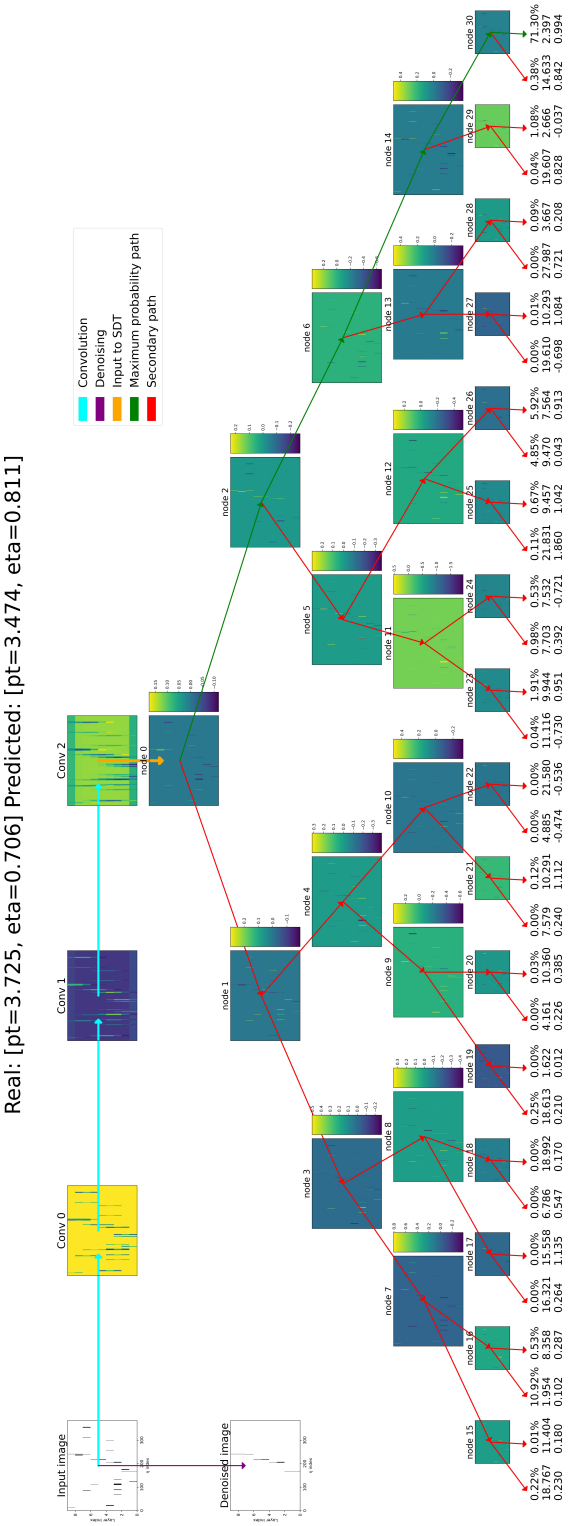
We have tested the explainability of this model with only-noise images as well. As stated in the last section, our tree-based model was able to reject an astonishing amount of these events resulting in an error rate of just 0.0002% (only 2 images). One of them has been selected to be here analyzed and it is shown in Figure 7.8: looking at it we can clearly perceive the great uncertainty in directing the decision path towards a specific value. Here the confidence of the maximum probability path is only 21% with several other leaves with probabilities of $\approx 10\%$: these leaves in their entirety are in contradiction with each other and are unable to help the prediction. All this large quantity of nodes with low probabilities certify great

confusion and uncertainty in individuating a leading path.

Finally, with the proposed evaluative tool to provide visual explanations for the ConvSDT, it is easy to interpret a prediction exploiting trees' hierarchical structure and we can safely state whether the model's decisions can be considered reliable exploiting all the information contained in the inner and leaf nodes of the tree. This transparent architecture helps to deduce additional insights in the model's functioning about both exact and erroneous predictions of the model and is aimed simultaneously at expert and non-expert users in HEP thanks to its wide-ranged comprehensibility where, as for attribution methods, they can directly look at the kernel filters that right away shows, at all levels of the architecture, the influential pixels.

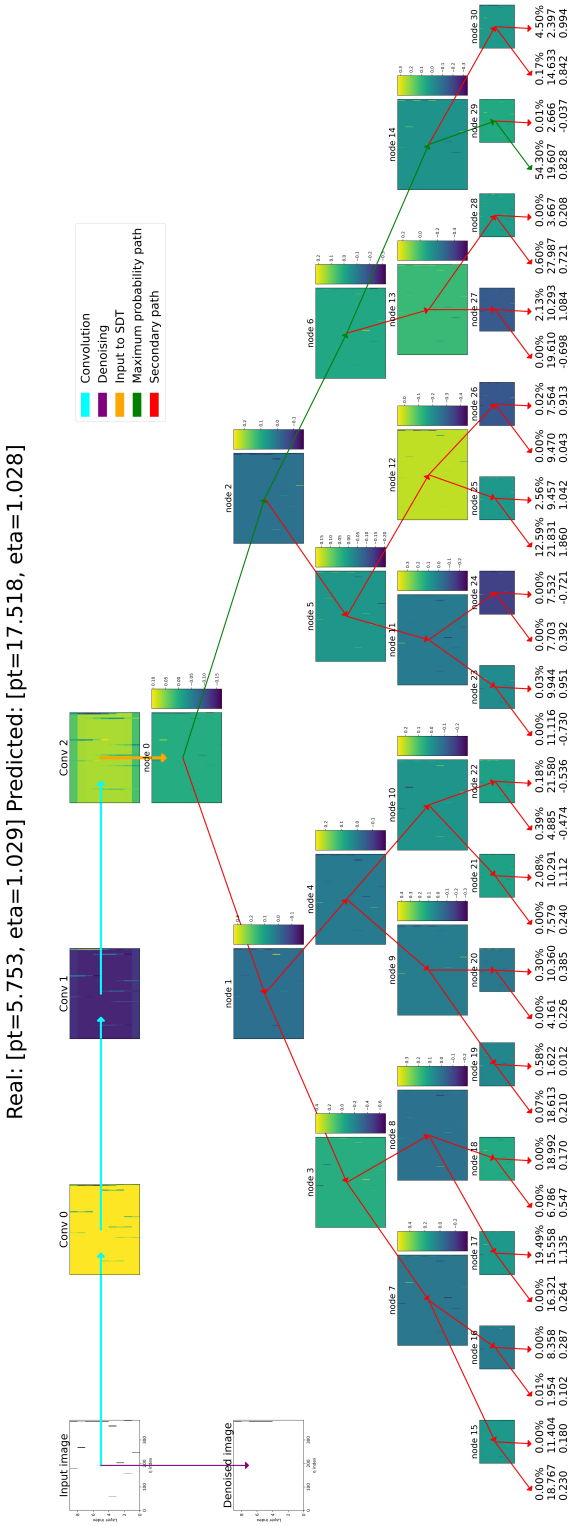


(a) True Positive example.

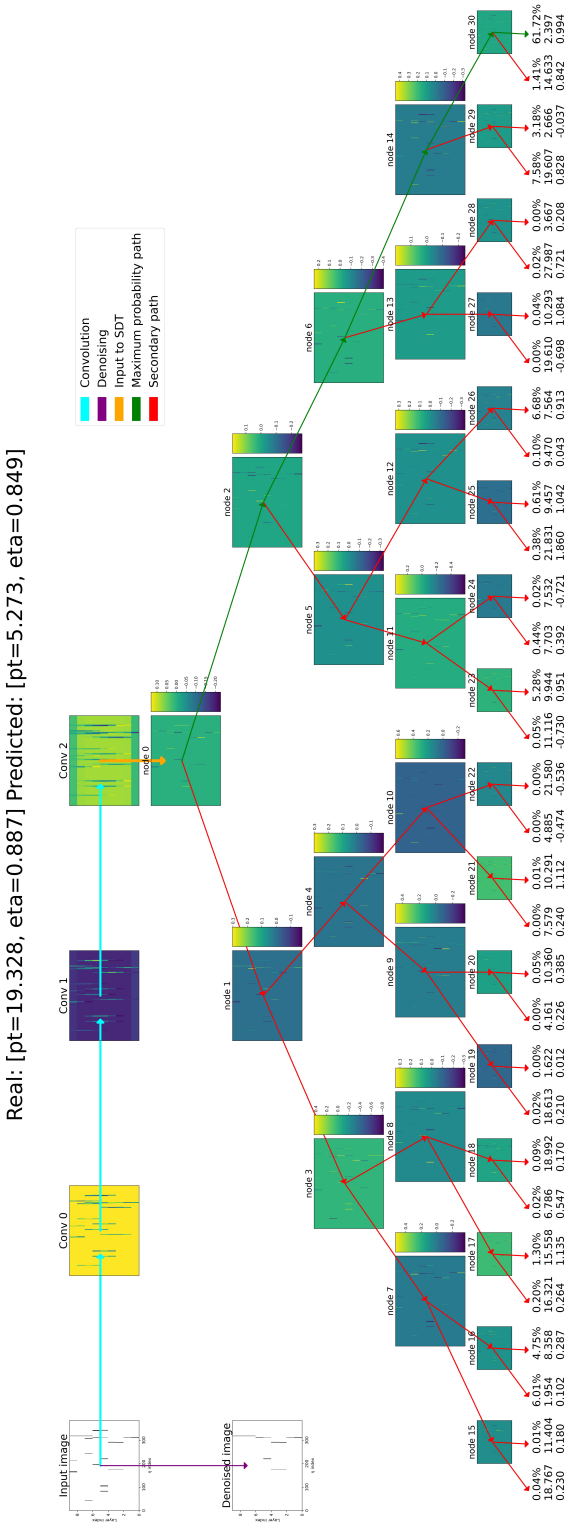


(b) True Negative example.

Figure 7.6. Decision tree-based explanations for correctly predicted muonic patterns.



(a) False Positive example.



(b) False Negative example.

Figure 7.7. Decision tree-based explanations for wrongly predicted muonic patterns.

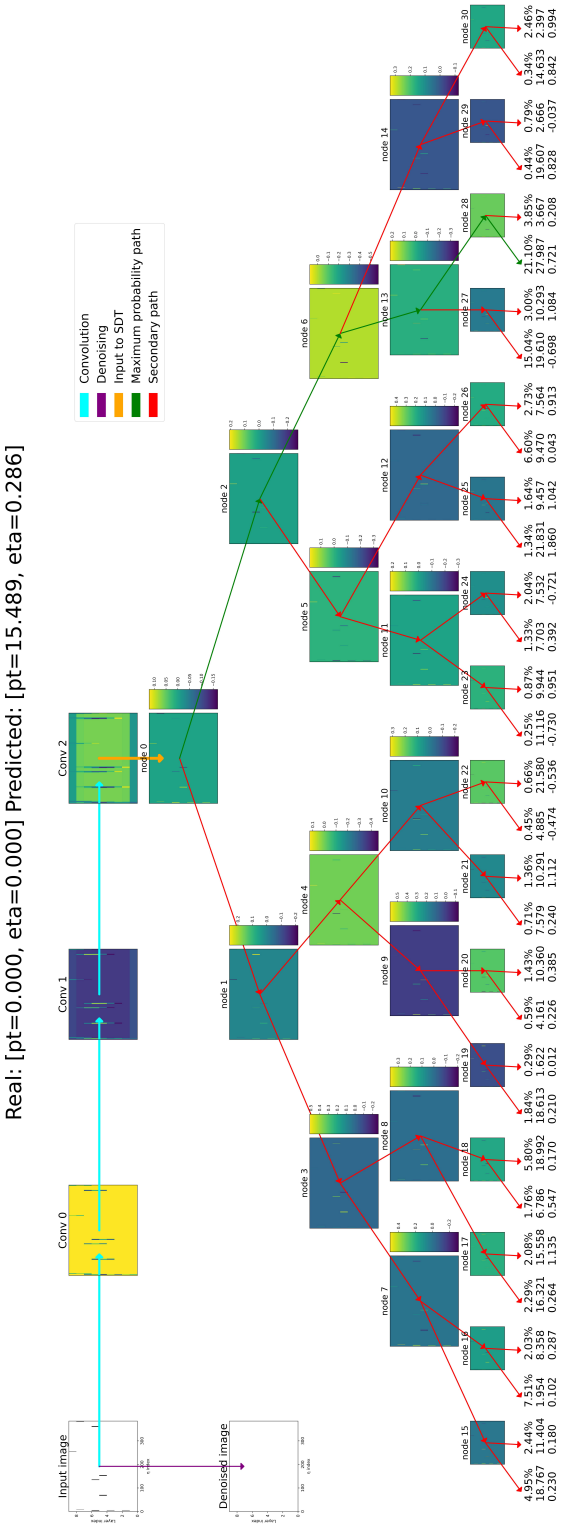


Figure 7.8. Decision tree-based explanation for only-noise muonic pattern.

7.3.3 Tracing Gradient Descent

The concluding part of this explainability dissertation is about our last xAI method, namely TracIn to estimate training data influence, applied to the final architecture: TracIn CNN. TracIn is based on the concepts of Proponents, namely positively influencing train samples that have reduced the loss at training time, and Opponents, instances that, vice versa, have increased the loss and were responsible for errors during the learning process.

In the following, each explanation aims at showing these two sets together with the image to be explained, its denoised version, and the real and predicted labels. In every image belonging to the two relevant sets, the user can also read a title containing its identification number and its level of influence in deciding the labels of the proposed input[3]. We decided to limit the analysis to the top 10 of both Proponents and Opponents to avoid presenting a messy explanation. Moreover, with the intent of bringing in their visualizations the most possible information, they are plotted so that the pixels representing noise are depicted in black and the actual pattern of the muon in red.

It is important to notice that the complete interpretability of the proposed plots can be fully derived only with prior knowledge about the specific use-case of HEP. We report that non-expert users may in fact be confused and may not adequately understand the outputs of this xAI technique. The explanatory power of this approach has hence more limitations than the other two, but anyway could assist them in some particularly difficult experiments.

Because of the just mentioned reason, we consulted HEP researchers to validate our results: fortunately, they were satisfied with our approach and reported that TracIn is able to individuate Proponents consistently. This method was indeed capable of, the majority of the time, yielding Proponents having reasonable labels with respect to the ones of the explaining image in cause and, more importantly,

[3]Clearly, a positive influence is assigned to Proponents while a negative influence is given to Opponents.

having a real pattern somehow resembling the input.

Unfortunately, we could not get the same validation in the case of Opponents. The feedback of the experts was in fact concerned with Opponents' unintelligibility and incoherence: we can state that, in some cases, we can find similarities between them and the sample but, anyway, this does not appear frequently enough to attest to the robustness of the method. With this in mind, we are going to proceed with the study of the explanations by only giving priority to Proponents; Opponents will be shown in the subsequent visual outputs but will not be discussed.

The analysis focuses once more on correctly predicted examples, visible in Figure 7.9, to then move on to the wrong ones (Figure 7.10).

The TP selected event, in Figure 7.9a, shows a trajectory on the left side with a high momentum of ≈ 16.3 GeV: TracIn CNN correctly selected it, assigning a prediction of 16.6 GeV. The TN sample (Figure 7.9b) instead has a more curved muonic pattern with a $p_T \approx 12$ GeV and a prediction of $p_T \approx 10$ GeV.

Analyzing these two cases, we can consistently notice how the Proponents of TP and the Opponents of TN (and vice versa) are respectively the same samples from the training dataset. While at first this could be considered a bit strange, we instead believe it is a natural implication of this work: the concepts of Proponents/Opponents and Positive/Negative are in a way symmetrical and it makes sense that samples helping the positive predictions could be simultaneously restraining the negative ones (and vice versa).

Focusing on Proponents only, the similarities between the denoised actual pattern of the test image and the muonic paths in red in the associated most influential train samples are evident: when the input has high momentum expressed as a straight line in the pattern, the majority of Proponents is characterized by a straight line path too, while, analogously, in the case on a curved trajectory implying a low p_T, also the muonic traces in the Opponents are mainly "crooked" and very similar to the curvature of the test image. In these two cases, we can clearly understand that the predictions are supported by a relevant amount of coherent and similar data

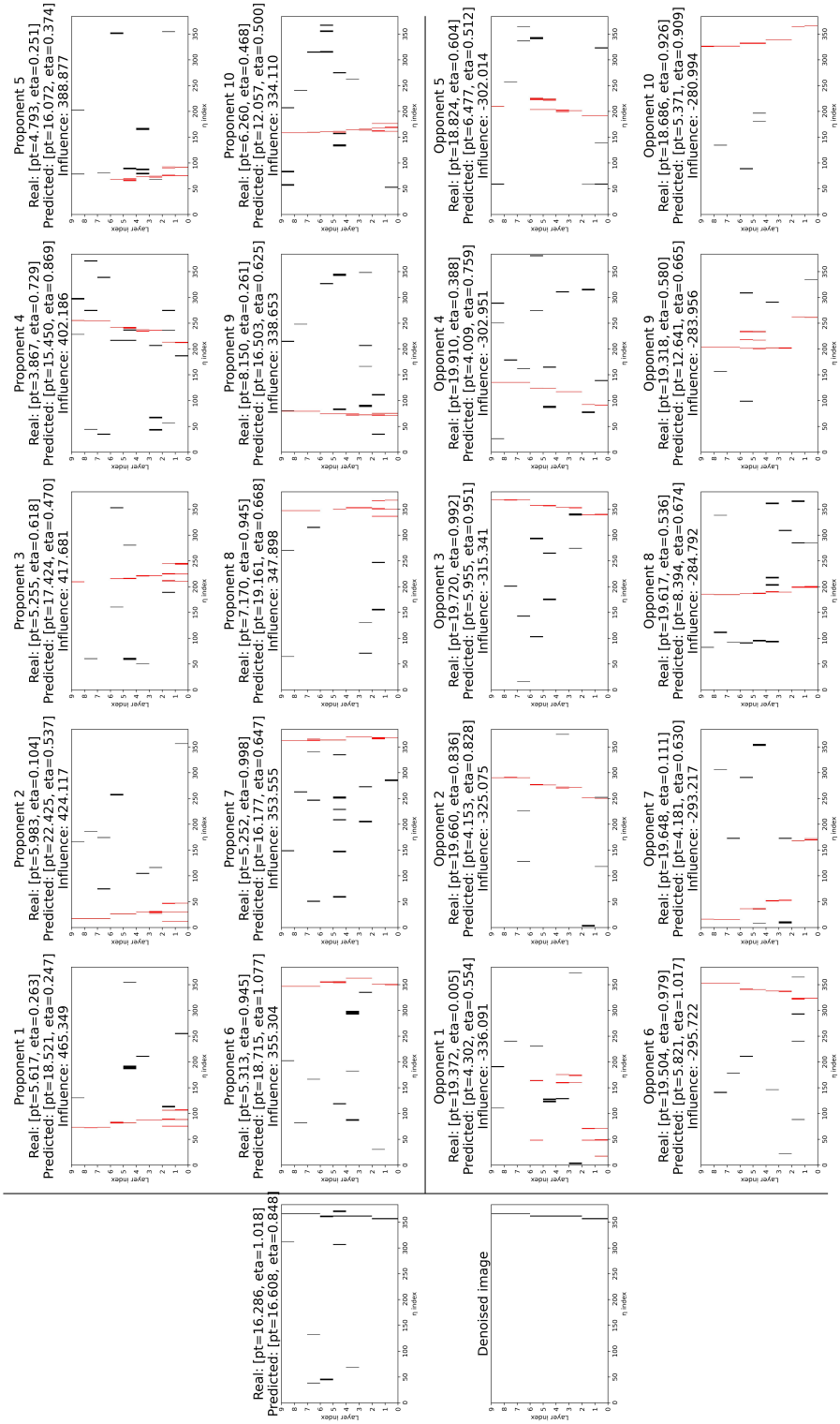
samples, resulting in the high reliability of our model's predictions.

The FP image in Figure 7.10a has a straight trajectory, which would suggest a high p_T, but is instead labeled with a p_T of ≈ 7 GeV: the prediction, as expected, assign it a value of 18 GeV. On the opposite, the FN sample (Figure 7.10b) has a high labeled p_T of ≈ 19 GeV but a prediction of 7.5 GeV: again the real path in the image is slightly curved and therefore the ground-truth is not immediately justified. As already happened before, both samples miss relevant pattern components: FP lacks the first two layers of muonic trajectory and FN is missing the last three, and again this could explain TracIn CNN's misclassification.

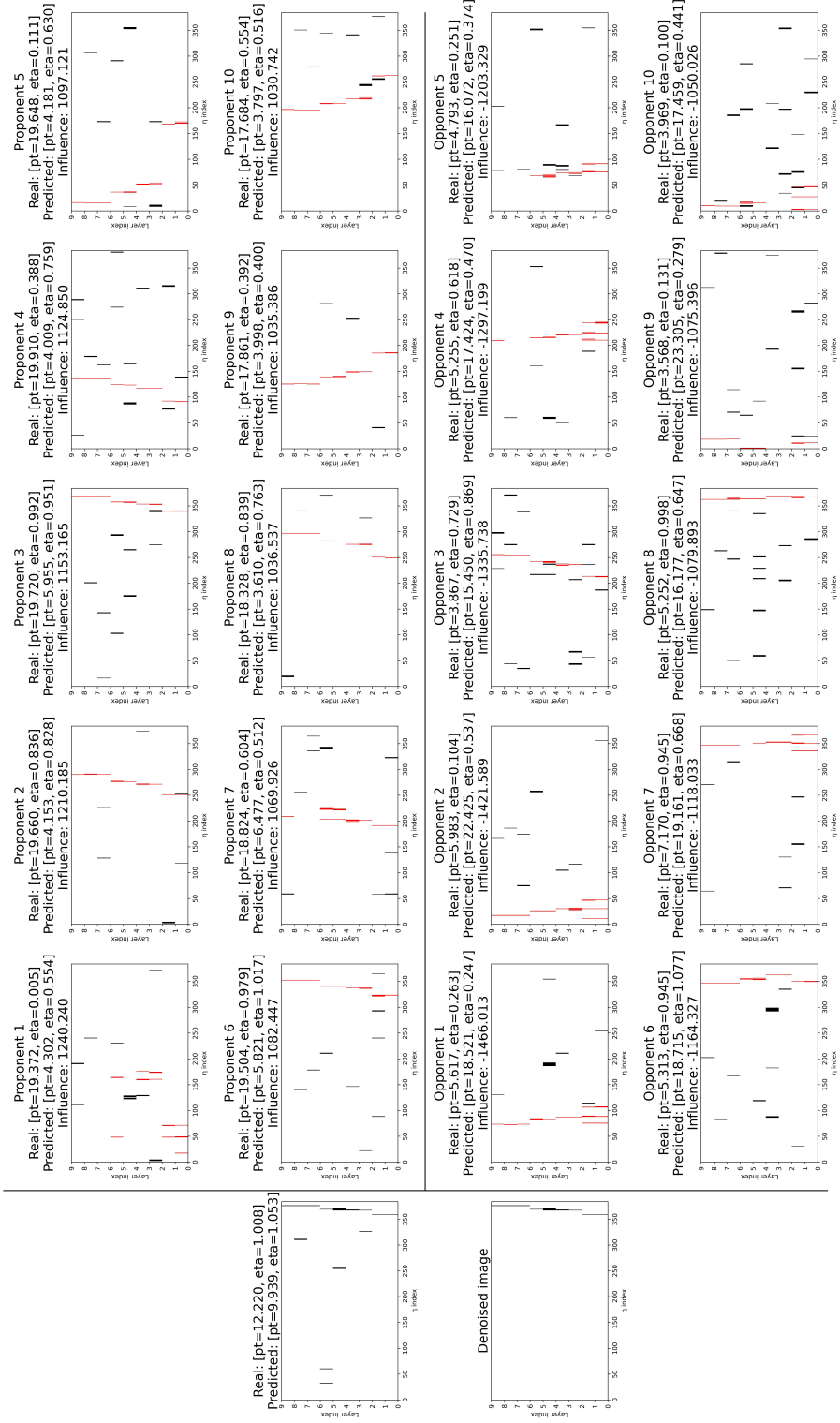
Anyway, when considering both these wrong decisions, the user observing the nature of the associated Proponents intuitively interprets it as a bad prediction because, even if they are similar to the test instance in input, they are responsible for the error by directing the p_T and η values away from their actual values.

One final observation about the usage of TracIn is related to the case with only-noise images. As in Distilled ConvSDT case, even this network was able to reject almost every event belonging to this category. Studying this small set is again our focus at this point in the analysis, to try to extract knowledge on how to enhance the noise rejection mechanism by individuating which fictitious pixels are mainly confused as an actual muonic trajectory. We can do this by applying the gradient tracing to an only-noise sample with a high predicted p_T. A sample outcome is depicted in Figure 7.11, it was decided to have a $p_T \approx 12.4$ and the figure effectively groups some of the Proponents that mostly confuse the model having as input an image with noisy data only.

An additional work I would like to mention for future experiments is addressed in [32], where the authors implement a method based on attention to individuate the most influential data samples for an image classification task, which directly recalls the definition of Proponents we have adopted in this work and inspired by the original paper introducing TracIn [48].

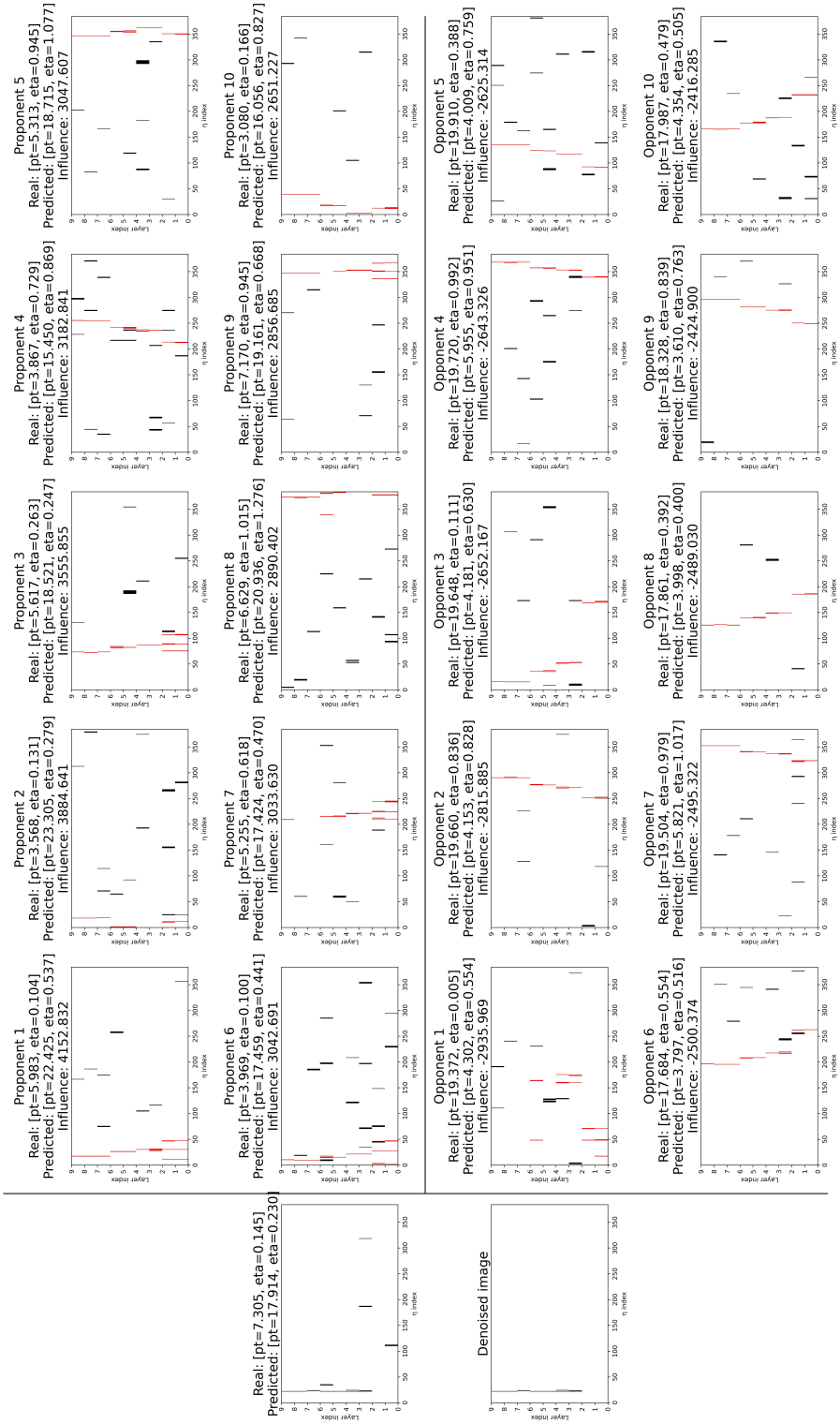


(a) True Positive example.

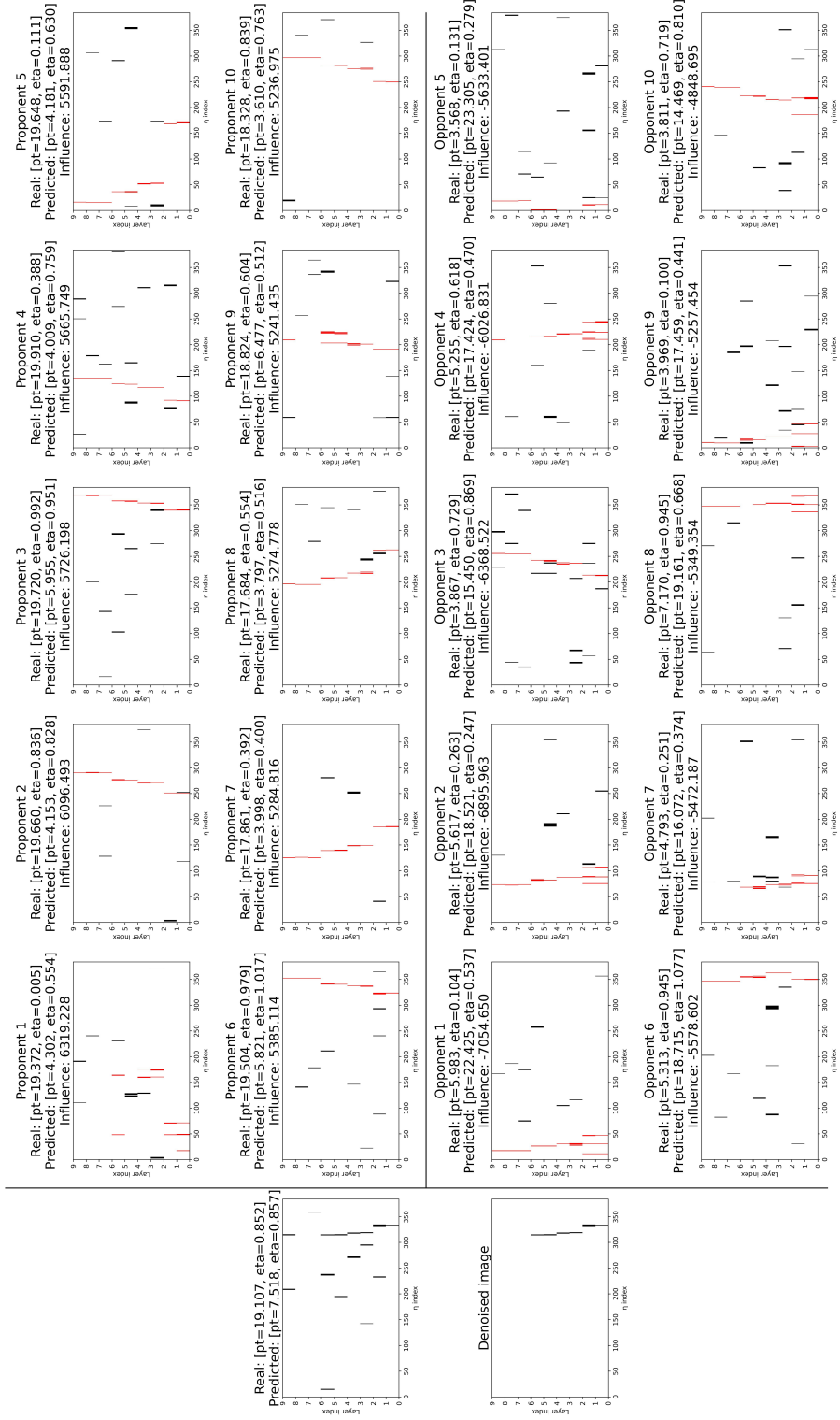


(b) True Negative example.

Figure 7.9. TracIn example-based explanations for correctly predicted muonic patterns.



(a) False Positive example.



(b) False Negative example.

Figure 7.10. TracIn example-based explanations for wrongly predicted muonic patterns.

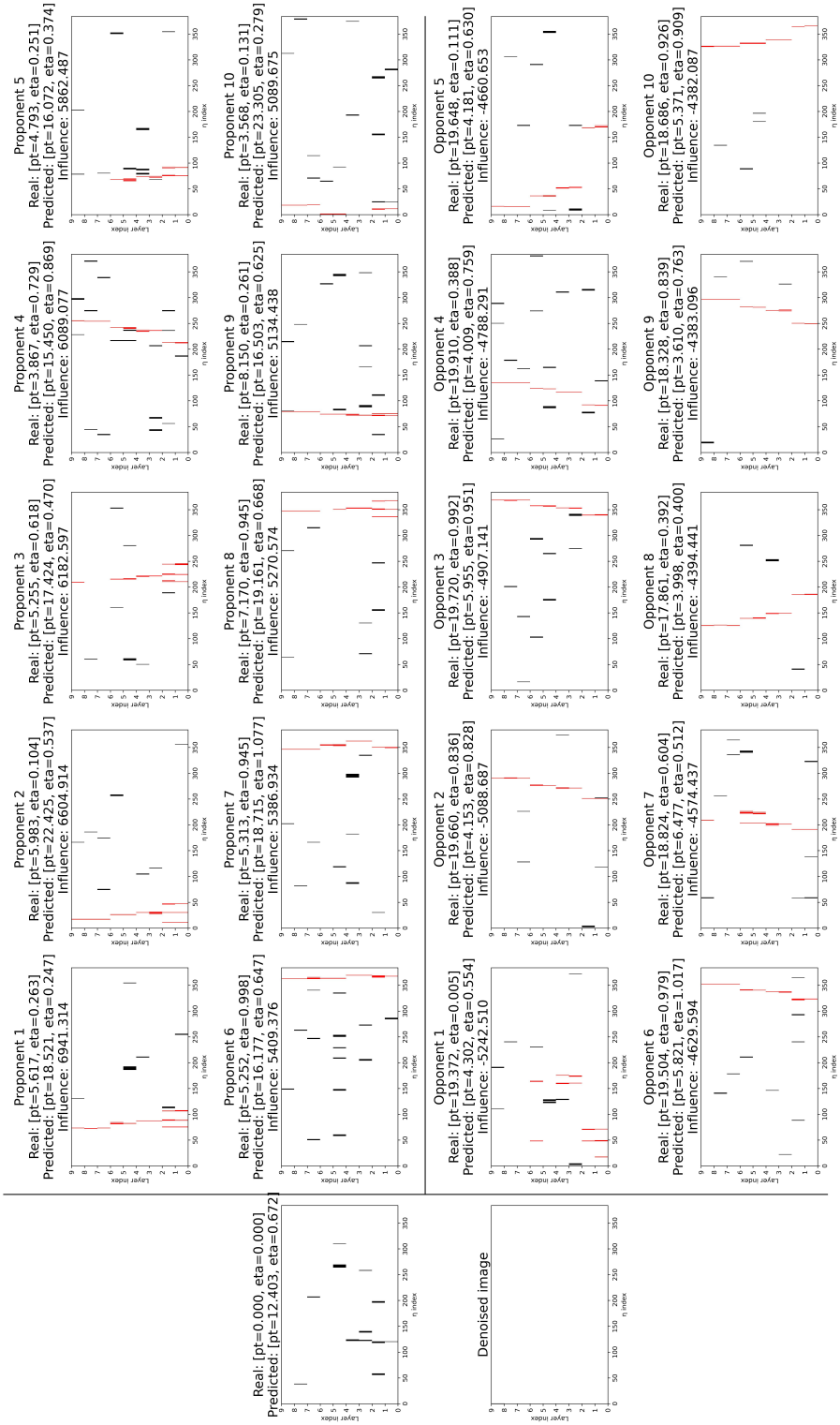


Figure 7.11. TracIn example-based explanation for only-noise muonic pattern.

Chapter 8

Conclusions

Finally, let us conclude and summarize the main topics of this thesis.

The proposed High Energy Physics use-case was successfully addressed: I gave a full description of the HEP foundations and their ingenious application at CERN and, specifically, in the ATLAS detector. A needed in-depth theoretical analysis on Explainable Artificial Intelligence and its techniques was provided as well: I mentioned the academic matters, studying them mathematically, but we also focalized the socio-economic and ethical topics related to xAI.

Through its implementation, this work has been extensively able to prove that, including and implementing well-advised xAI and KD directives, it is possible to re-create perfectly performing ML models which also guarantee predictions' explanations needed in critical application domains. We were able to re-create and extend some Attribution-based methods (Activation Map, Integrated Gradients, and SmoothGrad) for Convolutional Neural Networks. With this first model (defined as Attribution CNN), we guaranteed satisfactory performances with respect to the two physical metrics we used for evaluation (spread and efficiency) while simultaneously "peeking" inside the "black-box" architecture of our CNN by producing heatmap images containing the most significant discriminative regions for the regression assignment.

Moreover, focusing both on correct and wrong predictions, users are able to

detect whether the model's outcomes can be considered reliable or to diagnose its limitations by investigating the causes of errors and further optimizing the ML algorithm.

We can state that this first result would have been already fully adequate to solve the given task in an experimental setting: however, it is always important to remember that a model often becomes valuable only when it ensures its potential in being deployed in real-time and in real-world scenarios. In fact, in realistic decision support systems, a model must offer the possibility to be consulted in a short time and without the use of excessive computational resources[1].

This, together with the intrinsic explainability, has been one of the major reasons motivating the implementation of our second type of architecture, the Convolutional Soft Decision Tree. It is a hierarchical architecture inspired by the simpler Decision Tree and combined with the neuronal capabilities of a Deep Learning method. It is not only remarkably smaller with respect to the CNN, meaning lower required memory space and number of parameters, but also intrinsically-explainable thanks to the peculiarity of its structure.

We trained a simple ConvSDT and then we improved its performance in the regression task by exploiting a simple yet effective technique of Knowledge Distillation, through which we transferred the reliable capabilities of the CNN into such a simplified model without loss of performance: on the opposite, Distilled ConvSDT was capable of overcoming Attribution CNN in most of the metrics used for validation. We created an effective visualization tool to display its decision flow and provide exhaustive explanations for a requested input image. The augmented interpretability was enriched by extrapolating information directly from the nodes of the tree and rendering it in a human-readable fashion in order to be easily understandable even by non-expert users.

Finally, going back again to our non-interpretable CNN architecture, we have

[1]Here we are not only intending the classical computational resources (i.e. computational time and memory space), we should also remember that our CNN model is built to run on GPU while our ConvSDT uses a simple CPU. Nowadays it is still rare for ordinary machines (for instance, smartphones) to be equipped with dedicated graphical units.

also tested a different method to provide explainability based on estimating data influence by computing gradients and tracing their descent, TracIn: it was analyzed on a network defined as TracIn CNN. This method properly individuates which are the most influential images that have mainly affected predictions of a specific test image to explain. We have demonstrated the reliability of its outcomes both for correct and wrong predictions and we have also provided potential and future studies to exploit this technique in very-noisy domains and understand the regions of the image causing confusion.

At the end of our work, we were able to implement a large set of several xAI techniques based on different cutting-edge methods of this Artificial Intelligence field, capable to guarantee competitive performances on a real-world application. The strengths and drawbacks of each approach have been assessed thoroughly, highlighting the cases when errors were introduced by our models but also finding some flaws in the considered datasets. Most importantly, we provided effective ways to explain their predictions and decisions through a convergent and combined use of xAI methods.

Bibliography

[1] Pamina Adadi and Mohammed Berrada. Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI). *IEEE Access*, 6, oct 2018.

[2] Alper Ahmetoğlu, Ozan İrsoy, and Ethem Alpaydın. Convolutional Soft Decision Trees. In Věra Kůrková, Yannis Manolopoulos, Barbara Hammer, Lazaros Iliadis, and Ilias Maglogiannis, editors, *Artificial Neural Networks and Machine Learning – ICANN 2018*, pages 134–141, Cham, 2018. Springer International Publishing.

[3] J. Al-Khalili. *Paradox: The Nine Greatest Enigmas in Physics*. Broadway, 2012.

[4] N. S. Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992.

[5] Randall Balestriero. Neural Decision Trees, 2017.

[6] Marta Bistron and Zbigniew Piotrowski. Artificial intelligence applications in military systems and their influence on sense of security of citizens. *Electronics*, 10(7), 2021.

[7] CERN. About CERN. https://home.cern/about. Accessed: 2022-12-18.

[8] CERN. Detector & Technology. The ATLAS Detector. https://atlas.cern/Discover/Detector. Accessed: 2022-12-18.

[9] CERN. Future Circular Collider. https://home.cern/science/accelerators/future-circular-collider. Accessed: 2022-12-18.

[10] CERN. Main dataset. https://drive.google.com/file/d/1s6JZuJPAqUtnnvezDepApE4BMKo3FP4q/view?usp=sharing. Accessed: 2022-12-18.

[11] CERN. Muon definition. https://home.cern/tags/muon. Accessed: 2022-12-18.

[12] CERN. Muon Spectrometer. https://atlas.cern/Discover/Detector/Muon-Spectrometer. Accessed: 2022-12-18.

[13] CERN. Muon trigger data. https://gitlab.cern.ch/lsabetta/muontriggerdata. Accessed: 2022-12-18.

[14] CERN. Only-noise dataset. https://drive.google.com/file/d/1nKHHj6ZqbS1SGkH5OYpezbYujbRlXxsi/view?usp=sharing. Accessed: 2022-12-18.

[15] CERN. Physics. https://www.home.cern/science/physics. Accessed: 2022-12-18.

[16] CERN. Resistive Plate Chambers. https://cms.cern/detector/detecting-muons/resistive-plate-chambers. Accessed: 2022-12-18.

[17] CERN. The ATLAS Experiment. https://atlas.cern/about. Accessed: 2022-12-18.

[18] CERN. The Standard Model. https://www.home.cern/science/physics/standard-model. Accessed: 2022-12-18.

[19] CERN. Trigger and Data Acquisition. Selects events with distinguishing characteristics that make them interesting for physics analyses. https://atlas.cern/Discover/Detector/Trigger-DAQ. Accessed: 2022-12-18.

[20] CHIST-ERA. MUCCA - Multi-disciplinary Use Cases for Convergent new Approaches to AI explainability. https://www.chistera.eu/projects/mucca. Accessed: 2022-12-18.

[21] ATLAS collaboration. *ATLAS detector and physics performance: Technical Design Report, 1.* Technical design report. ATLAS. CERN, Geneva, 1999.

[22] The ATLAS Collaboration et al. The ATLAS experiment at the CERN large hadron collider. *Journal of Instrumentation*, 3(08):S08003–S08003, aug 2008.

[23] Roberto Confalonieri, Ludovik Coba, Benedikt Wagner, and Tarek R. Besold. A historical perspective of explainable Artificial Intelligence. *WIREs Data Mining and Knowledge Discovery*, 11(1):e1391, 2021.

[24] Arun Das and Paul Rad. Opportunities and Challenges in Explainable Artificial Intelligence (XAI): A Survey, 2020.

[25] European Parliament. Artificial intelligence in transport: Current and future developments, opportunities and challenges. https://www.europarl.europa.eu/thinktank/en/document/EPRS_BRI(2019)635609. Accessed: 2022-12-18.

[26] Hidde Fokkema, Rianne de Heide, and Tim van Erven. Attribution-based explanations that provide recourse cannot be robust, 2022.

[27] Francescato, Simone, Giagu, Stefano, Riti, Federica, Russo, Graziella, Sabetta, Luigi, and Tortonesi, Federico. Model compression and simplification pipelines for fast deep neural network inference in FPGAs in HEP. *Eur. Phys. J. C*, 81(11):969, 2021.

[28] Nicholas Frosst and Geoffrey Hinton. Distilling a Neural Network Into a Soft Decision Tree, 2017.

[29] GitHub. Use median instead of mean when constructing RandomForestRegressor. https://github.com/scikit-learn/scikit-learn/issues/9553#issuecomment-324484928. Accessed: 2022-12-18.

[30] Jianping Gou, Baosheng Yu, Stephen John Maybank, and Dacheng Tao. Knowledge distillation: A survey. *CoRR*, jun 2020.

[31] Shun ichi Amari. Backpropagation and stochastic gradient descent method. *Neurocomputing*, 5(4):185–196, 1993.

[32] Kazuki Irie, Róbert Csordás, and Jürgen Schmidhuber. The dual form of neural networks revisited: Connecting test time predictions to training patterns via spotlights of attention, 2022.

[33] Ozan Irsoy, Olcay Taner Yildiz, and Ethem Alpaydin. Soft decision trees. In *International Conference on Pattern Recognition*, 2012.

[34] Mir Riyanul Islam, Mobyen Uddin Ahmed, Shaibal Barua, and Shahina Begum. A systematic review of explainable artificial intelligence in terms of different application domains and tasks. *Applied Sciences*, 12(3), 2022.

[35] Kemal Erdem. XAI Methods - Integrated Gradients. https://erdem.pl/2022/04/xai-methods-integrated-gradients. Accessed: 2022-12-18.

[36] Jan Kietzmann, Jeannette Paschen, and Emily Treen. Artificial intelligence in advertising: How marketers can leverage artificial intelligence along the consumer journey. *Journal of Advertising Research*, 58:263–267, 09 2018.

[37] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[38] Shusen Liu, Bhavya Kailkhura, Donald Loveland, and Yong Han. Generative counterfactual introspection for explainable deep learning. *GlobalSIP*, 2019.

[39] Stuart P. Lloyd. Least squares quantization in pcm. *Information Theory, IEEE Transactions on 28.2*, pages 129–137, 1982.

[40] Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *CoRR*, abs/1705.07874, 2017.

[41] Haoran Luo, Fan Cheng, Heng Yu, and Yuqi Yi. SDTR: Soft Decision Tree Regressor for Tabular Data. *IEEE Access*, 9:55999–56011, 2021.

[42] Sparsh Mittal. A survey of FPGA-based accelerators for convolutional neural networks. *Neural Computing and Applications*, 32:1109–1139, 2018.

[43] Johanna Moore and William Swartout. Explanation in expert systems: A survey. 01 1989.

[44] Kyoung-Su Oh and Keechul Jung. Gpu implementation of neural networks. *Pattern Recognition*, 37(6):1311–1314, 2004.

[45] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[46] Donald H. Perkins. *Introduction to High Energy Physics*. Cambridge University Press, 4 edition, 2000.

[47] Antonio Pich. The standard model of electroweak interactions. *CERN-2012-001*, 2012.

[48] Garima Pruthi, Frederick Liu, Mukund Sundararajan, and Satyen Kale. Estimating Training Data Influence by Tracing Gradient Descent, 2020.

[49] Ali Rahimi. Machine learning has become alchemy, 2018.

[50] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why Should I Trust You?": Explaining the Predictions of Any Classifier, 2016.

[51] Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should I trust you?": Explaining the predictions of any classifier. *CoRR*, abs/1602.04938, 2016.

[52] Carlo Rovelli. *Seven brief lessons on physics.* Riverhead Books, New York, New York, First American edition, 2016.

[53] Carlo Rovelli. *Reality Is Not What It Seems. The Journey to Quantum Gravity.* Penguin Books, UK, 2017. OCLC: 1049123981.

[54] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. Smoothgrad: removing noise by adding noise, 2017.

[55] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic Attribution for Deep Networks, 2017.

[56] Peter Svenmarck, Linus Luotsinen, Mattias Nilsson, and Johan Schubert. Possibilities and challenges for artificial intelligence in military applications. In *Proceedings of NATO Big Data and Artificial Intelligence for Military Decision Making Specialists' Meeting*, 05 2018.

[57] Erico Tjoa and Cuntai Guan. A Survey on Explainable Artificial Intelligence (XAI): Toward Medical XAI. *IEEE Transactions on Neural Networks and Learning Systems*, 32(11):4793–4813, 2021.

[58] Michael van Lent, William Fisher, and Michael Mancuso. An explainable artificial intelligence system for small-unit tactical behavior. In *Proceedings of the 16th Conference on Innovative Applications of Artifical Intelligence*, IAAI'04, page 900–907. AAAI Press, 2004.

[59] Giulia Vilone and Luca Longo. Explainable Artificial Intelligence: a Systematic Review, 2020.

[60] Zhiguang Wang and Jianbo Yang. Diabetic Retinopathy Detection via Deep Convolutional Networks for Discriminative Localization and Visual Explanation, 2017.

[61] Steven Weinberg. A model of leptons. *Phys. Rev. Lett.*, 19:1264–1266, Nov 1967.

[62] Yongxin Yang, Irene Garcia Morillo, and Timothy M. Hospedales. Deep Neural Decision Trees, 2018.

[63] Yu Zhang, Peter Tiňo, Aleš Leonardis, and Ke Tang. A Survey on Neural Network Interpretability. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 5(5):726–742, 2021.

[64] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning Deep Features for Discriminative Localization, 2015.

[65] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Object detectors emerge in deep scene cnns. *ICLR 2015*, 2015.