EC3

Methodological Guideline for Explainability

L3.4.(1,2,3).3

contact@confiance-ai.fr | www.confiance.ai

Contributors

	Name	Organisation	Role
Responsible for the deliverable	Antonin Poché	IRT Saint Exupéry	Author, Lead of 3.27
Scientific responsible	Philippe Dejean	IRT Saint Exupéry	Lead of 3.15 and co-author
Co-authors	Mathilde Guillemot	Air Liquide	Industrial partner

Document Control

Revision	Date	Author	Commentary
V1.0	31/12/2022	Philippe Dejean	Batch 2 version
V2.0	21/11/2023	Antonin Poché	Batch 3 version
v2.1	05/12/2023	Antonin Poché	Batch 3 complete version

Contents

A. Introduction

A.1. Motivations

A.1.1 General introduction to trustworthy AI challenges

Trustworthiness in AI within critical systems (systems that can directly or indirectly affect human life and moral entities) is essential for its widespread adoption (by the industry, the decision makers, the general public, etc.) and poses the following significant challenges.

- First, how to design AI models, so that, by construction, they satisfy trustworthy properties (accuracy, robustness...).
- Secondly, how to characterize these AI models, for example to understand and explain their behavior and their adequacy to the operational domain.
- Then, how to implement and embed those AI models on hardware, by making them fit for the target without losing their trustworthy properties.
- Another question is, what methods of data engineering to apply in order to, among other topics, manage important volumes of data and adapt to the evolution of the operational domain.
- At system level, what verification and certification processes to consider specifically for AI-based systems.
- Finally, a federation of all these matters is necessary to build an end-to-end methodological approach, supported by a consistent engineering environment compatible with industrial practices.

These are the challenges, among others, that the Confiance.ai program addresses.

A.1.2 Motivations for explainability guidelines

In today's rapidly evolving landscape of artificial intelligence and machine learning, the importance of explainability cannot be overstated. As these technologies increasingly permeate various aspects of our lives, from healthcare and finance to autonomous vehicles and recommendation systems, the demand for transparent and interpretable AI systems is on the rise. Users, regulators, and stakeholders require insights into how decisions are made by these algorithms, making the need for explainability paramount.

However, navigating the vast terrain of explainability is no simple feat. There is a profusion of methods, tools, and solutions available (i.e. explainability libraries like Xplique, Captum, AIX360, or PairSaliency...), each with its own set of advantages, drawbacks, and trade-offs. The sheer diversity of these approaches underscores the complexity of the challenge in ensuring that AI and machine learning models can provide meaningful explanations for their predictions and decisions.

Choosing the right method or methods for achieving explainability is not only a technical decision but also a strategic one, deeply intertwined with the unique requirements and constraints of the target audience, the specific use case, and the broader context in which the AI system operates. What works for a medical diagnosis model may not be suitable for a recommendation system, and the expectations of regulators can differ significantly from those of end-users or business stakeholders.

This document is designed to be a comprehensive guide to help practitioners, researchers, and organizations navigate the intricate landscape of AI explainability. It will provide insights into the importance of explainability, highlight the multitude of available methods, and offer guidance on selecting the most suitable approach based on the intricacies of your specific situation.

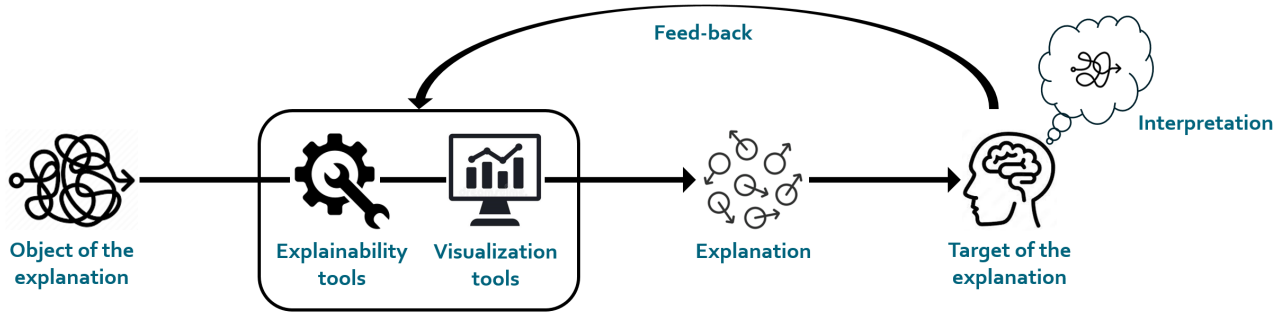

Figure A.1: Schema representing the process of explainability

A.2. How to use this document

To use this document, thus to apply the explainability guidelines, the reader should follow the steps described in the protocol Ch. B with the support of the taxonomy Ch. C. Nonetheless, reading this document can also give the reader a general view of the aspect of taking explainability into account in a AI system. Furthermore, the explainability taxonomy in Ch. C can give the reader an overview of the existing solutions either in the literature or packaged.

A.2.1 Glossary of terms and acronyms

Definitions were set by another document Delaborde et al. (2023), thus for consistency, we will use the same definitions:

- **Explainability** deals with the capability to provide the human with understandable and relevant information on how an AI application is coming to its result.
- **Interpretability** relates to the capability of an element representation (an object, a relation, a property...) to be associated with the mental model of a human being. It is a basic requirement for an explanation.
- *Comprehensibility* refers to the capability of an element representation (an object, a relation, a property...) to be understood by a person according to its level of expertise or background knowledge.

To complete those definitions, Fig. A.1 illustrates the position of the different entity in the explainability process. Then the explainability taxonomy Chap. C will cover more specific definitions.

A.2.2 Scope and organization of the document

The trustworthiness attributes treated by the present document are explainability and comprehensibility. The current guidelines guide the user through the different existing methods and libraries in explainability and how to choose the most adapted to his use case.
The two main contributions of this document are:

- The protocol on how to select the most adapted explainability methods presented in Ch. B.
- Then as a support to this protocol this document includes a taxonomy of explainability methods Ch. C, with a focus on the most mature methods in the literature, attributions methods, in Ch. D. The taxonomy includes a variety of tables to qualify which methods is pertinent for which target audience, which methods are available in which library, which library supports a particular use case.

Finally, this document explores possibilities of applying explainability in other paradigms in Ch. E.

A.2.3 Context with respect to other documents

This document is focused on explainability methods selection, thus for methods application, interpretability, and other aspects, it works in pair with other documents:

- Regarding methods application, the DEEL *Guidelines to explain machine learning algorithms* Vigouroux et al. (2023) describe in detail a scientific approach to apply and evaluate several kind of explainability methods (attributions, feature visualizations, and concepts).
- On a more atomic level, the detail on each method application is left to the different libraries and components. For example, Xplique provides unitary tutorial for each attribution method or tutorials for each task.
- For explanation analysis, the *Specification of the interpretability HMI components* Mader et al. (2023) provides specification for visualization and on the guiding toward interpretability. Furthermore, they work on the interpretability of methods parameters, and on comprehensibility of the explainability in general.
- The link can also be made with the *Analysis and capture protocol of use case holders needs and requirements* Arlotti and Heulot (2023) as they work on individuals vision of an AI systems, which includes their needs and constraints.

A.2.4 Target readers

Explainability methods should be applied by "ML-Algorithm Engineers", because it requires model understanding, therefore, the main target readers are "ML-Algorithm Engineers" tasked with the application of explainability. However, for interpretation, knowledge on the data is often required, thus "Data Engineers" may need to learn about the different explanation formats through the present guidelines. Finally, "Systems Engineers" can go through the description of the protocol to become familiar with explainability process and constraints.

A.2.5 Place in the end-to-end process

The guidelines described in this document are used in the model evaluation phase, in particular in the "Evaluation of ML model explainability" (see Robert et al. (2023) for more detail on the end-to-end process). However, it can also help the development of the ML model in the choice of the architecture, library... Indeed, if explainability is an important component for the AI system, taking into account explainability constraints into model development is pertinent. Either by choosing a model explainable by design or by taking into account the which kind of model is supported by the explainability methods of interest.
Furthermore, this document can also be used to understand the explainability problematic and the existing solutions.

A.2.6 Limitations and perspectives

The present guidelines can guide the reader on the choice of methods and libraries to use depending on the problem. However, there are some limitations which also represent potential improvements:

- The present guidelines do not describe how to apply the methods, neither the scientific procedure nor the code applications. The former is partly covered by the DEEL *Guidelines to explain machine learning algorithms* Vigouroux et al. (2023) and the latter by libraries' tutorials.
- The present guidelines do not cover all existing libraries or explanation formats as the number is too large.
- The protocol is in its first version, the feedback from industrial partners have not been taken into account yet.

B. Explainability protocol

The explainability protocol can be seen as guidelines on how to bring explainability into an AI system. The protocol can be divided in two parts:

- The method selection illustrated in Fig. B.1, this is the main focus of this document. This document extensively describes the selection of candidates and methods application and comparison are treated by a complementary document Vigouroux et al. (2023).
- The analysis of the explanation is less explored as it is covered by a complementary document Mader et al. (2023).

B.1. Select the explainability method

To select the method, we need to see which method can be applied, among those, which ones are interesting, then apply and compare them.

B.1.1 What do we have?

The first step in the method selection is to understand our context and problem to see which method or type of method can be applied, in some, answer to the question "What do we have?". This question can be divided in many sub-questions, the first ones are about the problem and the model:

- **What is the task?** Some task such as anomaly detection, object detection or segmentation need specific explainability methods.
- **What is the type of the data?** Depending on the type of data, some method may not be adapted, the vision task is overly represented in the literature and methods may not be adapted for time-series or text for example.
- **Do we have a model?** If we have an already trained model, then we will have to focus on post-hoc methods. Otherwise, we can either make an explainable by design model and jump to the What do we want section (B.1.2). Or, we can train a classic model and apply post-hoc explainability methods. Nevertheless, the method we want to apply may impact the kind of model we need and thus guide us in the architecture and/or library selection.
- **Do we have access to the weights?** When we have a model to explain, we may not have access to the weights, in this case, we are limited to black-box methods. Otherwise, we can take methods among black-box and white-box methods.
- **What is the architecture of the model?** Some explainability methods are designed for specific models architectures like CNN, LSTM or Transformers. In this case, knowing the architecture of the model is important.
- **In which library is it implemented?** Depending on which library the methods are implemented in, they may not be applicable to model from other libraries. It is particularly the case for white-box methods. We may have to limit ourselves to black-box methods. This includes Tensorflow and pytorch but also sklearn, jax and other libraries producing machine learning models.

Once we have described our model properties, we can now list the compatible methods:

- **What explainability methods do we have?** We may not want to implement explainability methods ourselves, therefore the easiest way it to use libraries. There exist many libraries and a large range have been benchmarked in the Confiance.ai project. Nonetheless, with the four following libraries, most of the implemented methods are covered: Xplique, Captum, AIX360, and PairSaliency. Those three libraries are available in the Confiance.ai environment.
- **Is the explainability method theoretically applicable to the model?** As aforementioned, some method are incompatible with some models, therefore, the number of available methods is constrained by their

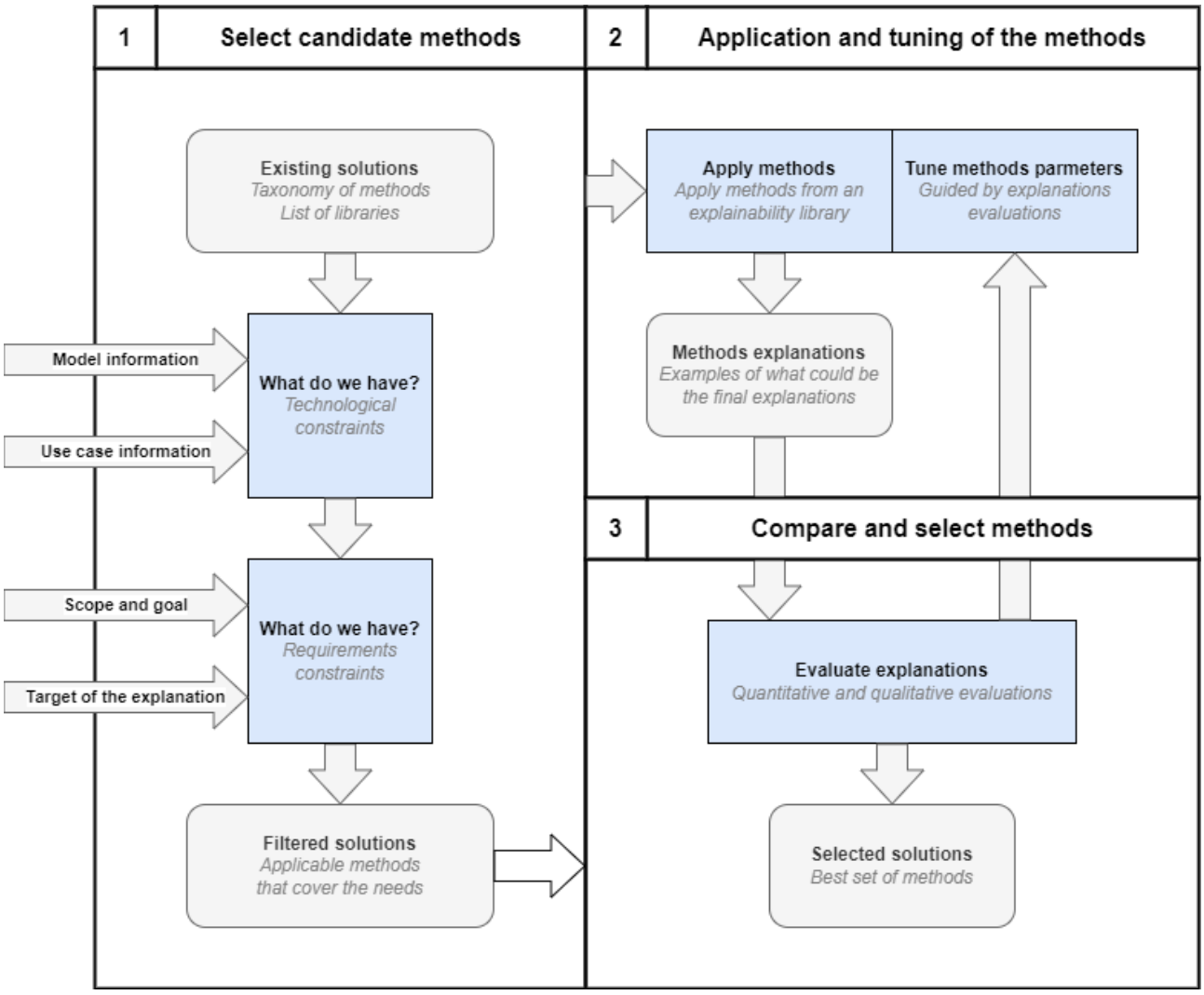

Figure B.1: Schema of explainability methods selection

compatibility. This question is not trivial, we can refer to Tab. C.2, Tab. C.3, Tab. C.4, and Tab. C.5, furthermore, the report Dejean et al. (2022) provides an in depth description and application of the available methods.

- **Is there an available tool supporting it?** Finally, even if the method is theoretically compatible, the available implementation may not support it. For example, suppose we have a tensorflow resnet model trained on imagenet and we want to apply the GradCAM method available on Captum. Yet, GradCAM is a white-box method using the gradient of the model and Captum is written in pytorch. As such, we cannot apply it to our model. See Tab. C.6 for libraries applicability.

We do not expect the readers applying this protocol to know all the different methods from all libraries. Therefore, we designed a method taxonomy (see Ch. C) with the questions "What do we have?" and "What do we want?" in mind to help in the protocol application. In particular, the table of libraries applicability Tab. C.6 provides a guide of which library may be applicable to the use case in hand. However, the announced coverage is for attribution methods.

B.1.2 What do we want?

The second step of the protocol is to analyze what kind of explanations do we need. We will see among the compatible and available methods which ones corresponds the most to our needs. In this step, we can also divide the problem between different sub-questions, there are questions regarding the object of the explanation, the target of the explanation and the explanation itself:

- **What is the object / scope of the explanation?** We may want to explain the model, the decisions and/or the data. Note that all explanations give information toward those three direction, but most of the time, there is a dominant one. (This is linked to the global / local taxonomy).
- **What are we aiming to do with the explanations?** We may want to detect a bad behavior, understand the decision process, have more confidence in our model or even discover new knowledge. Either way, this will impact the pertinence of the format of the explanation you will use and the required liability of the method.
- **Who is the target of the explanation?** We may need explainability in many context and for many different persons. The explanation may be for a data scientist designing the model, the domain expert who need to confirm the model pertinence, an end-user how need more confidence in the model or controller asserting model performance. They all have different means to interpret the explanation and it should be kept in mind.
- **What elements does the target have to interpret from the explanations?** Depending on the target, we may not want the same information to be understood from the explanations. Most of the time, we want explanation to be efficient, hence, it should focus on the pertinent points.
- **Which explanation format(s) do we prefer?** The four previous questions can guide us to select the most appropriate format of the explanation. More detail on the different formats we be given in the section C.2 in the taxonomy.

While applying this step, we may not know what we want precisely. In this case, we should not restrain ourselves and apply several kind of methods. Indeed, most of the time different explanation formats are complementary and will lead to a more complete comprehension of our model. We can see examples with Fig. B.2 and Fig. B.3 respectively from Chen et al. (2019) and Kenny and Keane (2021).

However, we may have an opposite behavior, this list may show us that what we want is not possible with the current methods. Either because it is too optimistic or because the libraries and the literature are not mature enough.

B.1.3 Application and tuning of the methods

The application and tuning step cannot be described here, each library and each method may work differently. Therefore, we should refer to the tutorials of the respective libraries, the general tutorials, the tutorials on the

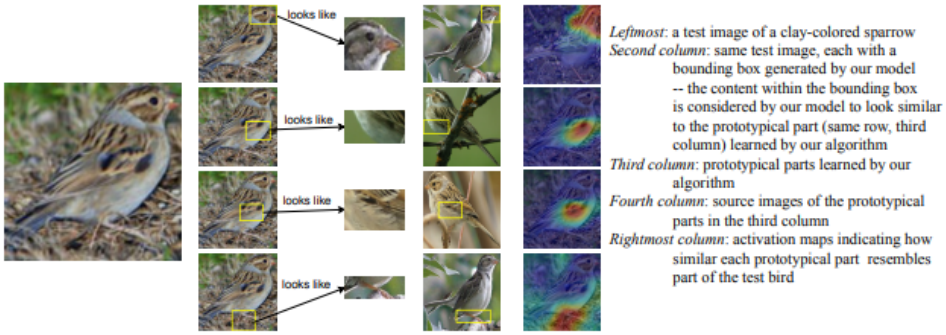

Figure 1: Image of a clay colored sparrow and how parts of it look like some learned prototypical parts of a clay colored sparrow used to classify the bird's species.

Figure B.2: Association of two explanation formats (prototypical parts and attributions) from Chen et al. (2019)

Fig. 7. *Requires Color* A CNN Misclassifies an Image of a "Rifle" as "Shovel": The nearest neighbors by themselves give some insight into the misclassification, but the FAMs improve the explanation as they show that the CNN has learned "Snowy-Background" as a primary feature in the identification of "Shovels" (n.b., the outlier in the nearest-neighbors is outlined in red).

Figure B.3: Association of two explanation formats (similar examples and attributions) from Kenny and Keane (2021)

task our model is solving, the tutorials on the data type we are treating before applying a method. Then, to tune a method, we should refer to the corresponding unit tutorial if it exists.
Nevertheless, guidelines for several explainability formats were proposed by DEEL Vigouroux et al. (2023) as they focus on Xplique's methods.

B.1.4 Comparison and selection of the methods

The final step before selecting the method is to compare them, you can either compare sets of settings for a method or methods between themselves. To do so, apart from our appreciations on which explanation we prefer (be careful of the confirmation bias), we should look at expected properties of an explanation. There are many properties that were presented in various papers and survey. We give for example the five from Fel and Vigouroux (2020):

- **Fidelity**: How representative of the model behavior is an explanation.
- **Stability**: If the explanation is robust to small perturbation (also depend on the model).
- **Comprehensibility**: How interpretable are the explanation (subjective).
- **Generalizability**: To which extend does the explanation truly reflects the underlying decision process.
- **Consistency**: How logically similar are explanation of two different predictors trained on the same task. (May not be wanted if predictors have different behaviors).

To evaluate how compliant is an explanation with respect the properties, we need metrics (apart from comprehensibility which is subjective). Those metrics may not be available in all libraries. But we should at least evaluate the fidelity. Nonetheless, we may encounter a trade-off, sometimes, comprehensibility and fidelity are opposed. There is no known solution for this problem, it is left to the user preference.

Through those metrics, we are supposed to select the best method to apply, but most of the time, methods formats are complementary. Selecting only one format would reduce the final comprehension. Hence, we suggest to keep a method from each format if you have several explanation formats.

B.2. Analysis through explanations

Once the method has been selected, we can generate pertinent explanations. However, an explanation in itself is most of the time unusable, we need to find a way to provide this information to a human in an intelligible way, this is the visualization part. Then, we need to provide to the target of the explanation the means to understand the visualization, this is the interpretability part.

B.2.1 Visualizations of the explanations

Visualization are often overlooked in the literature, therefore, the subject is not mature yet. Let's take the example of the most explored explanation format, the attributions (further detailed will be provided in Ch. C and Ch. D). Those methods generate for a given sample, attributions of each features' influence on the model decision. The associated visualization for images is a heatmap, sometimes superposed to the studied input image. The problem is that each method have a different definition of an attribution, some have negative and positive values while others gave values between 0 and 1 that sums to 1. This leads to a lot of questions: Should we use the same colors? Should take the absolute value? Should we normalize the values? Should we clip the values?...

In some, a visualization should provide unbiased information, it should aim toward comprehensibility but represent the whole explanation. We do not have guidelines to build visualizations yet, but it should be included in future work. Nonetheless, when building visualization, one should be careful that the visualization effectively carries the information of the explanation.
Further recommendations and guidance are provided in a complementary document Mader et al. (2023).

B.2.2 Interpretation of the explanations

The interpretation of an explanation depends on the person that will interpret it. We can be in three different cases:

- The person understands the explanation and have a correct interpretation.
- The person thinks he understood the explanation but he does not have a correct interpretation of it. Even if it is slightly different.
- The person knows he does not understand the explanations or have doubts.

The problematic case is the second one, we can assume that it is not possible to treat this problem after the explanation has been given. Otherwise, it would prevent a smooth operation. Therefore, we suggest to only show explanations to persons who have the means to interpret it. However, the interpretability field is not mature yet and we cannot provide guidelines to do so, it should be developed in future work.
Similarly to visualizations, more recommendations and guidance are provided in a complementary document Mader et al. (2023).

C. Taxonomy

Explainability is a large field that has been extensively explored in the literature, they are thus a large range of explainability methods. Therefore we need to take a step back on those methods through the taxonomy. However, as large as it is, explainability is also a young and not mature domain, as such the taxonomy we will present you is only one of the many possible taxonomies (Adadi and Berrada (2018); Carvalho et al. (2019); Das and Rad (2020); Belle and Papantonis (2020)). The taxonomy we propose will be oriented by the aforementioned protocol, thus it will be divided in two parts. A first part on methods prerequisites or applicability and a part on explanation formats or outputs. Our aim is that through the taxonomy, answers to the protocol will automatically select the compatible and pertinent explainability methods.

C.1. Methods prerequisites and applicability

The methods prerequisites correspond to the "What do we have?" part of the protocol (see section B.1.1). Each category corresponds to one or several questions in the protocol. Therefore, classifying methods through this taxonomy will greatly reduce the method selection time.

- **Specificity :**
  - **task-specificity** (What is the task?): A method may only be applicable to one machine learning task or a smaller group of tasks. (Classification regression, segmentation...)
  - **data-type-specificity** (What is the type of the data?): A method may only be applicable to one data-type or a smaller group of data-types. (Tabular data, time-series, images, videos, text...)
  - **architecture-specificity** (What is the architecture of the model?): A method may only be applicable to one model or a smaller group of models. In this case, we should precise to which architecture it is specific (i.e. CNN, LSTM...).
- **Necessary information :** (Do we have access to the weights?)
  - **Black-box** methods see the model as a predictor box, no further information are needed.
  - **White-box** methods need to access the weights or the gradient of a model.
- **Application Time: Intrinsic vs By-design vs Post-hoc:** (Do we have a model?)
  - **Intrinsic** methods need to be built at the same time as the model itself is built. Those methods often require to deeply understand the model and to adapt the method to the precise model structure.
  - **By-design** methods correspond to models that either generate explanation with the decision, are transparent models, have explainable part or an architecture that ease explanations generation.
  - **Post-hoc** methods are applied to a trained model, they allow much more flexibility on the model choice.

C.2. Explanations formats

The first part of the taxonomy is common or comparable between most proposed taxonomy. In the second part, we group methods together through the format of their explanations, then we deduce their scope and target audience. We will first present some explanation format examples (we cannot make an exhaustive list), then we will discuss the consequences of the format selection on the possible target and scope of the explanation.

C.2.1 Examples of formats

C.2.1.1 Attributions

Attribution methods (also known as features importance or features relevance) compute the influence of each feature in a decision, they are often represented as heat-maps. However, each method has its own mathematical

definition of an attribution, thus they are often over-interpreted and subject to confirmation-bias. (see Fig. C.1).

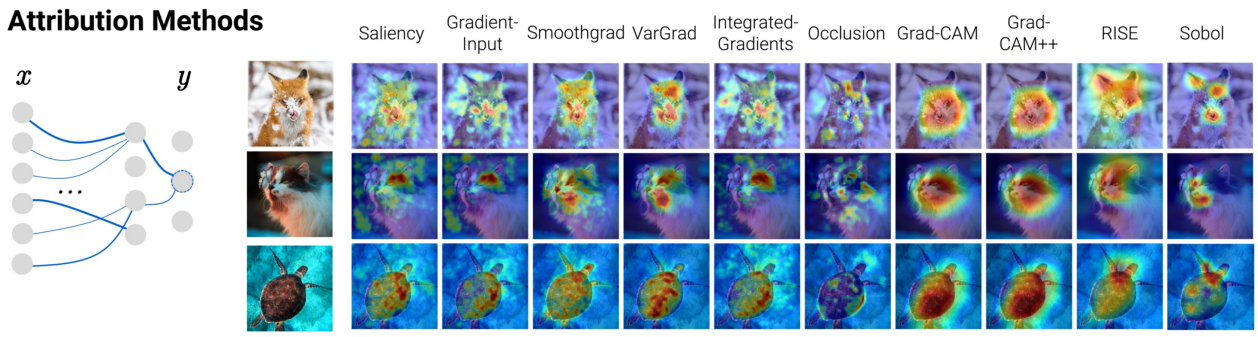

Figure C.1: Example of the attribution explanation format from the Xplique library

C.2.1.2 Example-based

Example-based can be seen as a group of formats which includes:

- **Case-based reasoning** methods use dataset elements or possible dataset element to make an explanations. We can find methods explaining a decision by showing what the model consider as the most similar examples to the studied sample (examples are extracted from the train set). (see Fig. C.2)
- **Influential instances** are elements which add an important influence during model's training. Hence for the prediction it tries to explain.
- **Counterfactual** will generate an element the closest possible to the studied sample but where the model make a different prediction.
- **Prototypes** are elements representing a class, a class often have several prototypes (it can be compared to centroids in clustering).

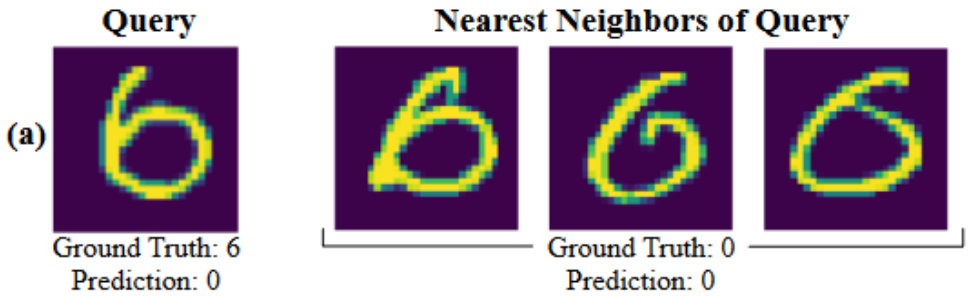

Figure C.2: Example of case-based explanation format with the COLE method from Kenny et Keane 2019 Kenny and Keane (2019)

C.2.1.3 Concepts

Concepts can be seen as sub-class of a sample, a class is represented by several sub-classes and a sub-class can appear in several classes. As an example, circle concept appear in cars for the wheels and in dogs for the eyes. Most of the methods use labeled concepts because generated concept are rarely interpretable. (see Fig. C.3)

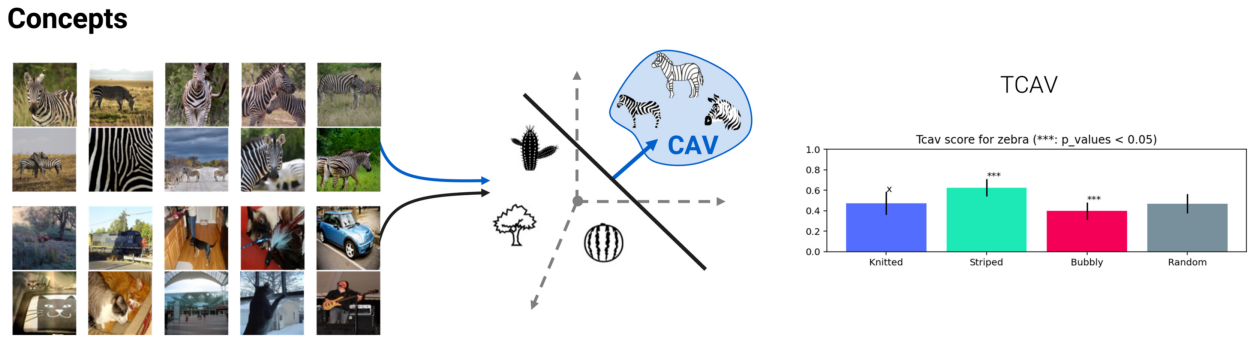

Figure C.3: Example of the concepts explanation format from the Xplique library

C.2.1.4 Dependencies

Dependencies or correlations, it is a coupling between two elements. We make the link between features or the link between features and labels. In both of those cases, the scope of the explanations is the data. While, when those methods make the link between features and decisions, the scope is naturally the decisions. Key methods are PDP (Hastie et al. (2009)) and ICE (Goldstein et al. (2015)).

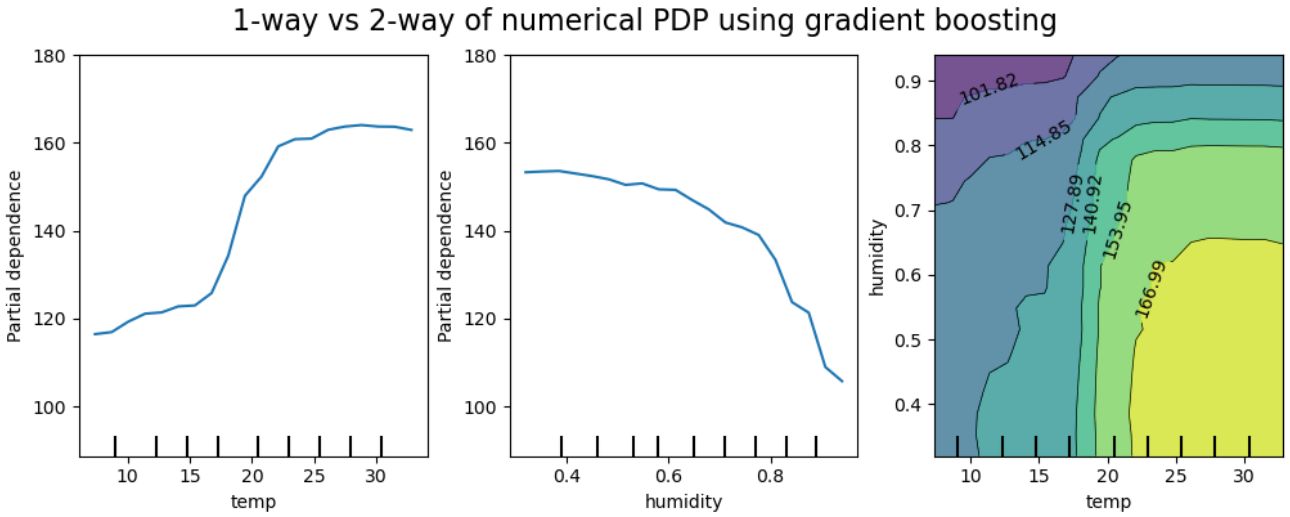

Figure C.4: Example of the dependencies format through the PDP method, visualization from scikit-learn.

C.2.1.5 Model surrogate

Model surrogate are small models, often considered transparent or interpretable. Their role is to represent a more complex model we would like to explain. Those simple models try to make the same prediction as the complex model (distillation). In this way when we understand the simple model behavior we can get intuition on the complex model behavior. Here again, those methods can be easily over-interpreted and are subject to confirmation bias. Furthermore, simple models need to make the trade-of between fidelity and comprehensibility (see section B.1.4 for definitions). In some cases, those simple model only approximate the complex model locally. (see Fig. C.5).

C.2.1.6 Rules

In the industry, most expert models are made of rules, thus if we can summarize the model decision process to a set of rules, it will be highly explainable. Nevertheless, the complexity of each rule and the number of rules

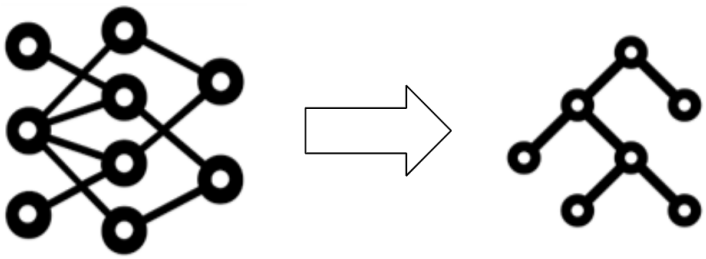

Figure C.5: Representation of the model surrogate explanation process

can hinder interpretability. This is another example of the trade-off between fidelity and comprehensibility of explanations.

C.2.1.7 Literal and oral explanations

Literal and oral phrases are used to convey explanations between humans, therefore, doing the same for artificial intelligence is natural and should be easier to accept for a human. Sridharan (2021) gives the example of an application to robots. We can illustrate it through a robot in a company and an employee who asks a question to the robot: "Why are you moving those boxes?" an AI could answer "I am looking for object A, object A is on shelf 1 but boxes are in front of shelf 1 thus I am moving those boxes." (see Fig. C.6).

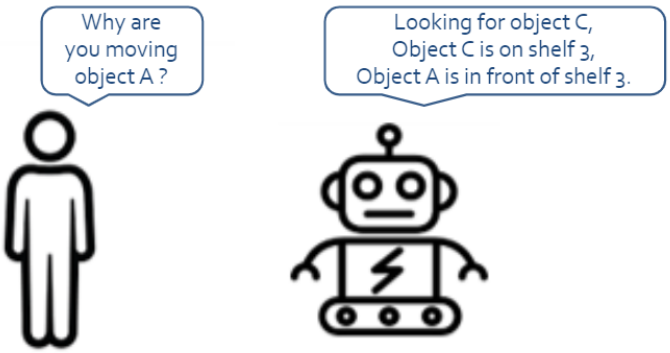

Figure C.6: Example of the literal and oral phrases explanation format

C.2.2 Formats selection

What we want to do with the explanations will restrict the explanations format we can use. Because, all the explanation formats cannot be used in the same way. Indeed, on the first hand, they do not convey information about the same elements (this also means that they are often complementary). On the second hand, the information they convey and the way they convey it may not be interpretable by all targets. Therefore, what we want to achieve through the explanations and the target of the explanations will guide our method selection.

C.2.2.1 The scope

The scope determines what an interpretability method aims at explaining. We determined three scopes, each method gives information on all scope but most of the time focuses on one.

- Methods explaining the **model** are known as **global** methods. They are methods that try to summarize the overall behavior of the model.

- Methods explaining the **decisions** are known as **local** methods. They focus on a single decision, and tries to unveil the process for this decision.
- Apart from those two groups, there are methods that aim to explain the **data**, it includes all data exploration. (This should not be confused with knowledge of model comprehension of the dataset).

C.2.2.2 The target of the explanation

When we produce an explanation, we do not want it to be limited to one person, we hope it to be for everyone. However, potential target audience may not have the same means to interpret an explanation, as knowledge on the model or on the data may be necessary. Thinking the explainability problem by specifying the target audience is key to avoid over-interpretation. We have no way to verify that a person got a coherent interpretation, hence we should present explanations only to persons with the means to interpret it. Means to interpret can also be provided together with the explanation, through guidelines for example.

We divide potential necessary knowledge in three categories and complement with a final constraint:

- **Dataset knowledge**: To understand an explanation, we often need to understand what the data represent. If instances from the train set are provided to the target to explain the behavior of the model, we expect the target be able to understand the data and make the link between them.
- **Model knowledge**: Some formats are based on the model structure and give insight on it, for example on the latent representation of the model. In those cases, without knowledge on the model, it is not possible to interpret the results.
- **General machine learning knowledge**: In some cases, an machine learning or statistical intuition is needed. Correlations for example cannot be interpreted without statistical knowledge.
- **Depend on the complexity**: Prior knowledge is not the only constraint to interpretability. A certain format may be trivial to interpret for simple explanations and impossible to interpret for too complex explanations. If we compare a one line rule to two pages of rules, we can see that the complexity of the explanation can hinder interpretability. The complexity can be the number of rules, the depth of the surrogate decision tree, the number concepts...

With those elements, we can take the example of some concrete targets examples to illustrate previous list:

- **End-user**: is the main user of the model. We cannot assume any prior knowledge from the end-user.
- **Domain expert**: has an extensive knowledge on the data, however, we cannot assume any knowledge on the model or in machine learning.
- **Data scientist**: has knowledge on the model and in machine learning.
- **Developer of the model**: has knowledge on the model and in machine learning by definition. But we can also assume knowledge on the data as it is pertinent for model development.
- **Regulator**: should have knowledge in machine learning in general.

Finally, some explanations formats give intuitions on the model but are not interpretable. Because they are by default too complex to be interpreted.

C.2.2.3 Correspondences

Once we listed the potential scopes and targets of an explanation, we can make the link between the formats and those elements. Hence, we propose Tab. C.1.
We previously said that most of the time, methods focus on one scope. However, in Tab. C.1, few formats seem to cover all scopes, prototypes for example. In fact, it depends on how we use the method: We could make an inference, compare the element to the closest prototypes and have a better understanding of the decision. More globally, we could look all the prototypes of each class and get intuition on the representation the model have of each class. Finally, if we are confident in our model, we can see the prototypes as information on the different classes and thus on the data.

Format	Scope	Prior knowledge			Depend on Complexity
		Dataset	Model	Machine learning	
(features) Attributions	Decisions			X	
Layers attributions	Decisions	X	X	X	X
Neurons attributions	Decisions	X	X	X	X
Example-based	Decisions	X			
Influential examples	Decisions	X	X	X	
Prototypes	All	X		X	
Counterfactuals	Decisions	X		X	
Concepts	All	X			X
Disentangled representation	Data	X	X	X	X
Model-surrogate	Model			X	X
Rules	Model				X
Literal or oral phrases	All				X

Table C.1: Correspondence between formats, scope and targets

C.3. Table of methods in each library and their classification

We described a taxonomy for explainability methods, however a taxonomy without classified methods cannot be used in the protocol. Therefore, we propose to classify all methods from the Xplique, Captum, AIX360, and PairSaliency libraries in the respective Tab. C.2, Tab. C.3, Tab. C.4, and Tab. C.5. Note that only **post-hoc** methods are provided and included in those tables because we did not study by-design methods in this work and they represent a negligible part of those libraries. Furthermore, the specificity correspond to the theoretical specificity and do not take into account the actual constraints linked to the implementation. Hence the model's library is not taken into account, therefore the model we want to explained should be compared to the method requirements in the documentation of the methods. Finally, an empty cell means that there is no limitation to the use of the method, for either, the model, the task or the data type. While a grey cell mean that the method is not concerned by the specificity, potentially because it does not explain the model.

C.3.1 Xplique

We can see in Tab. C.2 that the Xplique mainly focuses on attribution methods. The library proposes 16 attribution methods, with 6 black-box and 10 white-box methods, with close to zero specificity. The two other formats are more case specific.

C.3.2 Captum

The second table, (Tab. C.3), for methods from the Captum library also shows three different formats. With a particularity for the attributions, the difference between **features, layers, and neurons attributions**. The idea is to stop midway in the back propagation and return attribution either on the layer or the neurons. Naturally, not all methods are compatible and some may only be applicable to layers or neurons. Furthermore, those methods are by definition white-box as we stop in the middle of the neural network. Hence, also specific to neural network, but we will the specificity for the features attribution methods as they are the most used. There are respectively 16, 11, and 9 features, layers and neurons attribution methods. The format called **influential examples** is a case of example-based format where the training impact of a training sample toward the prediction on the studied sample is used to determine the similarity between samples.

Format	Necessary information	Name	Specificity		
			Task	Data-type	Architecture
Attributions	Black-box	HSIC		Image	
		Lime			
		Kernel-Shap			
		Occlusion			
		Rise			
		Sobol		Image	
	White-box	Deconv-Net			Differentiable
		ForGrad		Image	Differentiable
		Grad-CAM	Classification	Image	CNN
		Grad-CAM++	Classification	Image	CNN
		Gradient-Input			Differentiable
		Guided-Backprop			Differentiable
		Integrated-Gradient			Differentiable
		Saliency			Differentiable
		Smooth-Grad			Differentiable
		Square-Grad			Differentiable
		Var-Grad			Differentiable
Concepts	White-box	CAV	Classification	Labeled concepts	NN
		TCAV	Classification	Labeled concepts	NN
		Craft	Classification	Images	NN
Feature visualization	White-box	Feature visualization	Classification	Images	NN
		MaCo	Classification	Images	NN

Table C.2: Taxonomy of explainability methods from the Xplique library

C.3.3 AIX360

Tab. C.4 classifies methods from the AIX360 library. It shows the diversity of formats present in this library, but also highlights two main drawbacks of this library:

- The lack of applicability through the specificity of all methods.
- The lack of diversity in each format which prevents us from taking a step back and evaluate the explanations.

C.3.4 PairSaliency

Tab. C.5 classifies methods from the PairSaliency library. The library have a huge diversity of gradient-based methods and few interesting metrics. However, their focus on gradient-based can also be a drawback as they are more complex to apply. Indeed, their framework suppose that the user build himself wrappers calling the gradient of its model.

C.4. Table of libraries coverage of use case types

Format	Format precision			Necessary information	Name	Specificity		
	Features	Layers	Neurons			Task	Data-type	Architecture
Attributions	X			Black-box	Feature Ablation			
					Feature Permutation			
					KernelShap			
					Lime			
					Occlusion			
					Shapley Value Sampl.			
		X	X	White-box	Conductance			NN
	X		X		Deconvolution			CNN
	X	X	X		DeepLift			Differentiable
	X	X	X		DeepLiftShap			Differentiable
		X	X		Feature Ablation			NN
	X	X	X		GradientShap			Differentiable
	X	X			GradCAM		Image	CNN
	X		X		Guided Backprop			NN
	X				Input X Gradient			Differentiable
	X	X	X		Integrated Gradients			Differentiable
		X			Internal Influence			NN
		X			Layer Activation			NN
		X			Layer Grad. X Activ.			NN
	X	X			LRP			NN
			X		Neuron Gradient			NN
	X				Saliency			Differentiable
Concepts				White-box	TCAV	Classif.	Labeled concepts	NN
Influential Examples				White-box	SimilarityInfluence	?	?	?
					TracInCP			Differentiable
					TracInCPFast			Differentiable
					TracInCPFastRandProj			Differentiable

Table C.3: Taxonomy of explainability methods from the Captum library

Format	Necessary information	Name	Specificity		
			Task	Data-type	Architecture
Attributions	Black-box	Lime			
		SHAP			
	White-box	SHAP			Differentiable
Contrastive	White-box	CEM	Classification	Tabular	NN
		CEM-MAF	Classification	Image	NN
Disentangled representation	White-box	DIPVAE			
Distillation	White-box	Prof-Weight			NN
Literal phrases	White-box	TED		Labeled explanations	
Prototypes	White-box	Protodash		Tabular	
Rules	White-box	BRCG	Binary classif.		Boolean Rule Regression
		GLRM			Linear or Logistic Rule Regression

Table C.4: Taxonomy of explainability methods from the AIX360 library

Format	Necessary information	Name	Specificity		
			Task	Data-type	Architecture
Attributions	Black-box	Occlusion		Image	
	White-box	Blur IG			Differentiable
		Grad-CAM			Differentiable
		Guided Backpropagation			Differentiable
		Guided IG			Differentiable
		Integrated Gradients			Differentiable
		SmoothGrad			Differentiable
		Vanilla Gradients			Differentiable
		XRAI			Differentiable

Table C.5: Taxonomy of explainability methods from the PairSaliency library

Use case dimension	Detail	Library			
		Xplique	Captum	AIX360	PairSaliency
Model framework	Tensorflow	X	X	X	X
	Pytorch	X	X	?	X
	Sklearn	X			
Data type	Image	X	X	X	X
	Time Series	X		X	
	Tabular data	X	X	X	X
	NLP		X	X	
Tasks	Classification	X	X	X	X
	Regression	X	X	X	X
	Object Det.	X			
	Semantic Seg.	X	X		

Table C.6: Table of libraries applicability for attribution methods

D. Focus on attribution methods

Among all the available methods in the different libraries, attributions methods are by far the most represented. This also comes from their diversity in the literature and the maturity they acquired through time. These points justifies a particular focus on those methods. We will first present those methods more in detail, then discuss some general findings to ease parameters tuning or aim toward automation.

D.1. Deep understanding of methods

To comprehend attribution methods, we need to explore three points, the theoretical definition, the implementation and the empirical behavior. We will briefly see the definition of all methods or group of methods. However, there is no exhaustive list of attribution methods, and we could not review them all. Hence we will use the list of methods available in Xplique library as a reference. Furthermore, it will pertinent as there are specific tutorials corresponding to those methods which provides great intuition on how the methods work. However, other methods may fall in not described categories. The first division in attribution methods is between perturbation-based and back-propagation-based methods.

D.1.1 Perturbation-based methods

The perturbation-based methods take the model as a predictor box, they perturb the input and analyze the impact of those perturbations on the prediction of the model. Hence there are two points that may differ between perturbation methods: the way they generate perturbed inputs and the way they aggregate outputs difference to generate explanations. In any case, those methods do not need the weights of the model, just inferences, therefore, they are black-box methods. They have the advantage of a larger applicability, however, they require many inferences and can be costly.

D.1.1.1 The perturbations

We will focus our description on images, because they highlight the difference between perturbations. But for tabular data, all features are treated differently and not group in super-pixels, for time-series, we can group time-steps in windows but the principle is the same. Among the perturbation-based methods we can distinguish three types of perturbations:

- **Occlusion patch**: A patch moves across the image and set the corresponding pixels to a baseline value. One image is generated by patch and patches can overlap. (Methods: Occlusion)
- **Super-pixels segmentation**: The input image is divided in super-pixels and randomly set some super-pixels to a baseline value. The super-pixels are generated by classic image segmentation methods. One image is generated by mask of super-pixels. (Methods: Lime, KernelShap)
- **Grid division**: The image is divided through a grid then the cases of the grid are randomly set to a baseline value. One image is generated by grid mask. (Methods: Rise, Sobol, HSIC)

D.1.1.2 The aggregations

Once the perturbed inputs are generated we can compute the model predictions and compare them to the original prediction. The aggregation of those difference between the predictions on perturbed samples and the original prediction is the second part of those methods. We can distinguish three or four aggregations:

- **Mean**: Each feature should have been perturbed in at least one sample, hence for each feature we can take the mean of the prediction differences over the samples where the features was perturbed. (Methods: Occlusion)

- **Extrapolated mean**: It uses the same principles as previous aggregation, however, the explanation would have been too crude in this case, hence we can extrapolate the results to smooth it. (Methods: Rise)
- **Model surrogate**: We can also train a simple model to make the same prediction as our initial model on the perturbed samples and learn a local approximation of our model. The weights of this model are then used as explanations. (Methods: Lime, KernelShap)
- **Sensitivity analysis estimator**: Some recent methods are based on sensitivity analysis and combine a smart perturbation generation to a smart influence estimator. The attributions are computed in the grid dimension and extrapolated. (Methods: Sobol, HSIC)

D.1.2 Back-propagation-based methods

While perturbation-based methods process goes from the input to the output, the back-propagation methods process goes the other way around, from the outputs to the inputs. Those methods need an access to the weights to back-propagate information from the output to the inputs. Therefore, they are white-box methods, limited to differentiable models, nonetheless, they are often much faster. Most of those methods are based on the gradient, in fact, the gradient in itself is considered an explanation (this is the Saliency method). The methods can be divided in four categories:

- **Back-propagate through weights**: Attributions are back-propagated through the network by taking the inverse of each layer. (Methods: DeconvNet)
- **Simple gradient**: Computes the gradient one time, it may be on a slightly modified network or weighted by the input. (Methods: Saliency, Gradient-Input, GuidedBackprop)
- **Approximated gradient**: The gradient is too noisy, hence, some methods take a local approximation of this gradient. Several perturbed inputs are generated, inferences are made and the resulting gradient are aggregated. (Methods: SmoothGrad, SquareGrad, VarGrad, IntegratedGradient)
- **Gradient in feature space**: For some architectures, the gradient do not need to be back-propagated to the input space, it can be propagated to the feature space and the attributions are extrapolate to the input space. (Methods: Grad-CAM, Grad-CAM++)

D.1.3 Methods behavior

We only took the methods from the Xplique library, yet the methods diversity is striking. Naturally, the methods behaviors are also diverse and we cannot provide global guidelines. In the same sense, as each method computes the attributions in a different way, the resulting attributions do no represent the same things. We could say that each method is a new definition of attributions. Hence, neither can we give global guidelines on explanation interpretations, this work was initiated in parallel to this document Mader et al. (2023).

Nonetheless, some global remarks can be extracted, they are recommendations on what to do if a method fails or does not work. However, it is complicated to evaluate it, in simple terms, we could say that a method fails if the results are not comprehensible. Note that, a method could produce non comprehensible but faithful explanations. Other methods should also be evaluated as they provide point of comparison and non comprehensible results cannot be used. The different general guidelines are as follow:

- It does not make sense to try to explain a model with poor performances (not satisfying performances).
- If an explanation method fails, we should make sure we apply the method in the right way and that it is pertinent to apply it.
- If it still fails, we should look at the method's tutorial to use coherent parameters.
- If it still fails, we should try other methods:
  - If other methods succeed, then our initial method is at fault. We should look at metrics nonetheless, as the model may still be the problem (see section B.1.4).
  - Otherwise, there is a huge possibility that the model is not good.

D.2. Toward automation

Through the exploration of tutorials, we discovered that with some information, methods parameters possible values can be greatly reduced. For some parameters, the default value should be changed only in particular cases. Some parameters evolve by pair and fixing one greatly restrict the other one possibilities. Finally, for the number of samples, we can automatize its selection.

D.2.1 Fewer parameters

A method with fewer parameters is naturally easier to understand and to use, the user can focus his time on pertinent points. To reduce the number parameters, hence to automatically fix some parameters values we have several possibilities:

- Always maintain the default value. Some parameters are left open for very specific cases, but should not be modified most of the time. Those parameters should be modified only when other parameters tuning did not work.
- Select the value depending on the data type or compute the value directly from the data. In some cases, like the perturbation value, the value will depend on the data type and often, only one value make sense.
- Fix the value depending on another parameter's value. Often, there are pairs of parameters that should be modified together with a fix ratio, thus reducing the number of variables.
- Empirically automatize the value selection, the case of the number of sample in perturbation-based methods will be discussed in the next section.

D.2.2 Number of samples

The number of sample is used by most perturbation-based methods and by some gradient-based ones. Most of the time, this parameter is proportional to the computation cost of the method. Therefore, we want it to be as small as possible. However, it should be high enough for the method too converge.

To automatize this, we could create an algorithm that for gradually increasing values of this parameter, computes explanations and compare it to the previously computed explanation. If the distance between two consecutive explanations is under a given threshold, then we can consider that the method has converged and the corresponding value for the number of samples can be kept.

Nonetheless, each computation of the explanation can be costly depending on number of sample. Therefore, we propose to use an online mean: The explanations are computed on a fixed number of samples at each iteration and the new explanation is aggregated to the previous ones through a weighted mean. The newly aggregated explanation is then compared to the previously aggregated one, if the difference is small enough, then the method has converged. This process cannot be used for all methods. However, it should drastically decrease the time needed.

E. Explainability in other paradigms

As described in the protocol section B.1.2, there are many uses for explainability (we could use example-based explanation to detect mislabelling, as in figure C.2). In this chapter we will explore another use of explainability, the integration in other paradigms. In the Confiance project, we explored the possibilities of integration of the explainability in incremental dataset construction. It was divided in three main problematic: domain adaptation, active learning, and incremental learning. We will present in general how can the explainability be used before treating the three aforementioned paradigms more in detail.

E.1. Uses of explainability

Usually, explainability is used to gain confidence on a model. However, there are many others uses of explainability such as:

- **Knowledge discovery**: When a model is better than what we expect, understanding the model may lead to discover properties about our data we were unaware of.
- **Evaluate models**: Explainability can be use to complete performance metrics. For example when the model is faced with novelty, anomalies or corner cases data.
- **Compare models**: We can compare explanation of two different models on the same data, to understand why they produce different outputs or if they have the same way to provide a similar output.

In fact, we cannot evaluate models without baselines, an explanation on a given cannot be evaluated in itself. We have seen that there are metrics and users impression to evaluate explanation (see section B.1.4). Nonetheless, the scores from those metrics are not intelligible. Therefore, baselines are necessary, we can compare the scores between two explanations or directly compare two explanations.

E.2. Domain adaptation

Domain adaptation is a paradigm where we have a source and a target domain, which are both datasets of the same type of data, but the distribution may change. The aim is to obtain a model with the best performances on the target domain but we can only train it on the source domain. We may nonetheless change the source domain distribution to match the target domain distribution, there are many other possibilities but this is not the subject of this report. This is sometimes compared to transfer learning.

In this section we explore the possible uses of explainability in the domain adaptation paradigm. First of all, in domain adaptation, they define the naive model and the oracle model which are model only train on respectively the source and the target domain. Those two models represent great baselines for model evaluation and therefore to use explainability. Note that if we cannot see the difference between explanations from the naive and oracle model, explainability will be of no help.

The naive and oracle model provide us with respectively the lowest and the highest expected performances. Therefore, we can also get the worst and best expected explanations. Therefore, we can go further than performance metrics by comparing the explanations of different models on data from the target domain.

E.3. Active learning

Active learning is a paradigm where we have a dataset of labelled and unlabelled data, a model is initially trained on the labelled data. The aim is to find at each step, the best set of data to label to increase the model performances. It may be because data are costly to labelled or because we cannot train our model on all the data.

As previous paradigm, we are lucky to have have at least a baseline, the initial model, but if we have the labels, we could add a model trained on all data.

Nonetheless, there are more models and different data on which it can make sense to analyze explanations here. Let's say that the initial model is m_0 and the model trained on all data is m_n while m_i is the model trained with the initial data and the data added by the i different steps, $0 < i < n$. Lets also call the different parts of the datasets d_i while the initial dataset is d_0 and the total d. Note that if we cannot see the difference between explanations from m_0 and m_n, explainability will be of no help.

Through those definitions, with m_0, m_i, and m_n, we have three possibilities for a given data: the data has been seen by all three models, only by m_i and m_n or only by m_n. Therefore we can evaluate is adding the different datasets parts improves the explanations on already seen data, newly added data or unseen data. We could also see the evolution of the explanation for known data and unseen data while we gradually add dataset parts.

E.4. Incremental learning

In incremental learning, we also have an initial dataset and we add dataset parts step by step (like new classes for example). But we do not choose the parts and we cannot retrain on the previous parts (including the initial dataset).

We can use the same baselines and definitions as in active learning. However, a fourth possibility is added by dividing the "known by m_i and m_n" in two cases: the dataset is from d_i or the model is from a previously seen part.

F. Applications

F.1. Air Liquide - Cylinder Counting

F.1.1 What do we have?

- **What is the task?**
  The proposes task is an **object detection in crowded scenes**. Figure F.1 shows an example of the use case.
  The object to detect are cylinders from a top view of a transport truck. All cylinders are relatively similar, but there are some differences (in terms of size or cap), but in this preliminary work, there is no classification task and all cylinders are considered to be the same.
- **What is the type of the data?**
  Input are frames from a video.
  Output data are bounding box.
- **Do we have a model?** Yes
- **Do we have access to the weights?** Yes
- **What is the architecture of the model?** The model is a Yolov5s.
- **In which library is it implemented?** Pytorch

Once we have described our model properties, we can now list the compatible methods:

- **What explainability methods do we have?**
  According to the Table C.6, the only package integrated into the confiance.ai environment for object detection is Xplique.
  However, other methods not implemented in the packages mentioned above can be found in the literature (e.g. LRP for yolov5 Karasmanoglou et al. (2022))
- **Is the explainability method theoretically applicable to the model?**
  Inside confiance.ai environment, only XPlique library with all attribution methods.
- **Is there an available tool supporting it?** Yes

F.1.2 What do we want?

- **What is the object / scope of the explanation?**
  The main idea is to explain the decision of the algorithm. As a consequence, a local method must be chosen.

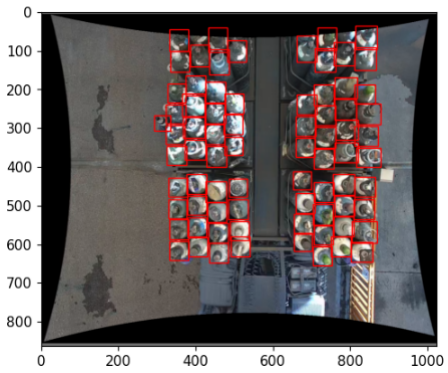

Figure F.1: Example of the result of object detection on one frame

- **What are we aiming to do with the explanations?**
  Two objectives are considered : detection of bad behaviour (false positive or false negatives) (1) and detect area where the confidence is lower (2).
- **Who is the target of the explanation?**
  The first explanation is useful for data scientist to better understand the model. The second one is intended for operators on fields to detect the areas that should be hand-processed.
- **What elements does the target have to interpret from the explanations?**
  In the first case, we seek to detect explanatory patterns that differ from the others. In the second case, we are interested in areas where the level of confidence is lowest.
- **Which explanation format(s) do we prefer?**
  There is no choice of format since only attribution methods are available. The only possible format is attribution.

F.1.3 Conclusion and feed-back

This questionnaire is easy to answer and paves the way for a selection tool for explainability method. The paragraph B makes it possible to understand most of the questions. However, two points could be deeper explored :

- the last three questions of the "What do we have ?" part seems to be redundant.
- the last question of the second part "What do we want ?" about the format of explanation is directly linked to the explainability method defined in the "What do we have ?" part. The question is redundant with the latter.

The two parts "What do we have ?" and "What do we want?" are very different. Indeed, the first one could be answered by anyone with access to the use case, however to answer the second required a deeper knowledge of the use case, its use and the industry.

- The first part is quite trivial, since all the answers can be found in the tables Tab. C.2, Tab. C.3, Tab. C.4, and Tab. C.5. However, the tables are sometimes difficult to read and cross-reference. To make it clearer and easier to use, a decision tree could be build based on the information contained in the different tables.
- The second part actually required a questionnaire and cannot be fully automated. However, some questions are not relevant depending on the answer to other questions (such as the format one when you have no choice). It would therefore be possible to skip certain questions or adapt there wording according to the answered given to previous questions.

As a conclusion, these methodological guidelines are a good starting point for choosing an explainability method adapted to specific needs. The first part "What do we have?" could be "automated" and replaced by a decision tree for instance. The second part "What do we want?" need to be enriched with different industrial use cases, to provide a broader list of possible answers to the various questions.

G. Conclusion

In this report, we have seen a protocol to bring explainability in a project, then a taxonomy that is adapted to this protocol. By classifying methods through this taxonomy we should be able to greatly decrease the method selection complexity. The taxonomy includes a variety of tables to qualify which methods is pertinent for which target audience, which methods are available in which library, which library supports a particular use case. This protocol - taxonomy pair also allowed us to take a step back on explainability methods and discover that among available methods, the attributions methods were over-represented. Hence, we provided further detail on those methods.

Finally, we described how can explainability be used in other paradigms. Explainability can be use to compare models and evaluate them if we have a baseline. Hence, in domain adaptation, active learning, and incremental learning, we were able to define pertinent model evaluation to complete performance metrics. Nonetheless, if the baseline models which should be the worst and the best references have comparable explanations, it is not pertinent to study the explanations. Maybe another explainability method can be applied.

Nonetheless, our work still possess some limitations, we will define them before presenting our perspectives.

G.1. Limitations

G.1.1 Protocol limitations

The protocol is limited to the selection of explainability methods. The guidelines on methods application and tuning depend on libraries Vigouroux et al. (2023). Regarding methods comparison, it also depend on available metrics in libraries. Then the analyzes part is in the scope of another document Mader et al. (2023).
Furthermore, we do not have now feed-back on the application of this protocol from industrial. However, our fear is that, for most use cases, the only available and applicable methods are attribution methods.

G.1.2 Taxonomy limitations

The taxonomy limitations are mainly about coverage, it cannot cover all existing kind methods in the literature, neither can it cover all libraries. Furthermore, libraries are bound to evolve, thus tables may become obsolete.

G.1.3 Current explainability limitations

Finally, the real limitations are the current available methods, indeed, for methods other than attribution methods, most use case do not have compatible implementations. Furthermore, most methods from the literature were not implemented in libraries.
Another limitation of the literature concerns the visualization and interpretation of the explainability methods. While some work have been initiated Mader et al. (2023), this problem is still not resolved.

G.2. Perspectives

Naturally, our perspective is first to tackle our limitations. As such, new explainability formats and libraries should be integrated. Then, we need to take into account feed-back from the industrial partners.
Finally, outside of this document, future work should develop and mature visualization and interpretation of explanations.

Bibliography

Adadi, A. and Berrada, M. (2018). Peeking inside the black-box: A survey on explainable artificial intelligence (xai). *IEEE Access*.

Arlotti, C. and Heulot, N. (2023). *Analysis and capture protocol of use case holders needs and requirements*. Confiance.ai.

Belle, V. and Papantonis, I. (2020). Principles and practice of explainable machine learning. *arXiv preprint*.

Carvalho, D. V., Pereira, E. M., and Cardoso, J. S. (2019). Machine learning interpretability: A survey on methods and metrics. *Electronics*.

Chen, C., Li, O., Tao, D., Barnett, A., Rudin, C., and Su, J. K. (2019). This looks like that: deep learning for interpretable image recognition. *Advances in neural information processing systems*, 32.

Das, A. and Rad, P. (2020). Opportunities and challenges in explainable artificial intelligence (xai): A survey. *arXiv preprint*.

Dejean, P., Vigouroux, D., Petit, A., Poché, A., Petiteau, D., Berhaut, P., Lecadre, S., Coppin, A., Gardet, C., and Allouche, T. (2022). *Project EC3, Action Sheet 15: Explainability benchmark v2 (Extension of batch 1 technical report) - Technical Report on use case Main document*. Confiance.ai.

Delaborde, A., Mattioli, J., Adjed, F., Amokrane-Ferka, K., Awadid, A., Gonzalez, M., and Sohier, H. (2023). *EC2 - Methodological Guideline for Trustworthy AI Assessment*. Confiance.ai.

Fel, T. and Vigouroux, D. (2020). Representativity and consistency measures for deep neural network explanations. *CoRR*, abs/2009.04521.

Goldstein, A., Kapelner, A., Bleich, J., and Pitkin, E. (2015). Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation. *journal of Computational and Graphical Statistics*, 24(1):44–65.

Hastie, T., Tibshirani, R., Friedman, J. H., and Friedman, J. H. (2009). *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer.

Karasmanoglou, A., Antonakakis, M., and Zervakis, M. (2022). Heatmap-based explanation of yolov5 object detection with layer-wise relevance propagation. pages 1–6.

Kenny, E. M. and Keane, M. T. (2019). Twin-systems to explain artificial neural networks using case-based reasoning: Comparative tests of feature-weighting methods in ann-cbr twins for xai. In *Twenty-Eighth International Joint Conferences on Artifical Intelligence (IJCAI), Macao, 10-16 August 2019*, pages 2708–2715.

Kenny, E. M. and Keane, M. T. (2021). Explaining deep learning using examples: Optimal feature weighting methods for twin systems using post-hoc, explanation-by-example in xai. *Knowledge-Based Systems*, 233:107530.

Mader, M.-A., Dejean, P., and TODO (2023). *Specification of the interpretability HMI components*. Confiance.ai.

Robert, B., Awadid, A., Langlois, B., Roux, X. L., and Proum, C.-M. (2023). *EC2 Project - Confiance.ai end-to-end approach for engineering trusted AI-based systems – v1*. Confiance.ai.

Sridharan, M. (2021). Toward explainable reasoning and learning in robotics. *not published*.

Vigouroux, D., Poché, A., Mussot, V., Glize, E., Ducoffe, M., Picard, A., Boisnard, F., Boumazouza, R., Fel, T., and Hervier, L. (2023). *Guidelines to explain machine learning algorithms*. DEEL.

Confiance ai

Titre : Guide méthodologique pour l'explicabilité

Mots Clés : Confiance, explicabilité, maturation, protocole, taxonomie

Ce rapport présente un protocol pour appliquer l'explicabilité dans un projet. Il guide principalement la sélection de la ou des méthodes adaptés au cas d'usage. Cela prend en compte les contraintes et les besoins du cas d'usage. De plus, ce document fournit une taxonomie des méthodes d'explicabilité, ainsi que des tables de ces méthodes dans les librairies. Ces éléments servent de support pour appliquer et comprendre le protocol.

Title: Methodological guidelines for explainability

Keywords: Trust, explainability, maturation, protocol, taxonomy

This report presents a protocol for implementing explainability in a project. It primarily guides the selection of suitable methods for the use case, taking into account its constraints and needs. Additionally, this document provides a taxonomy of explainability methods, along with tables of these methods within libraries. These elements serve as support for applying and understanding the protocol.

Our partners:

Air Liquide, AIRBUS, Atos, cea, Inria, NAVAL GROUP, GROUPE RENAULT, IRT Saint Exupéry, SAFRAN, sopra steria, SystemX, THALES Building a future we can all trust, Valeo, Investir l'Avenir