A Practical Tutorial on Explainable AI Techniques

ADRIEN BENNETOT, Sorbonne Université, Paris, France
IVAN DONADELLO, Free University of Bozen-Bolzano, Bolzano, Italy
AYOUB EL QADI EL HAOUARI, Sorbonne Universite, Paris, France and Tinubu Square, Paris, France
MAURO DRAGONI, Fondazione Bruno Kessler, Trento, Italy
THOMAS FROSSARD, Tinubu Square, Paris, France
BENEDIKT WAGNER, City University of London, London, United Kingdom of Great Britain and Northern Ireland
ANNA SARRANTI, University of Natural Resources and Life Sciences Vienna, Wien, Austria
SILVIA TULLI, Sorbonne Universite, Paris, France
MARIA TROCAN, Institut Supérieur d'Électronique de Paris (ISEP), Paris, France
RAJA CHATILA, Sorbonne Universite, Paris, France
ANDREAS HOLZINGER, University of Natural Resources and Life Sciences Vienna, Wien, Austria and Institute for Medical Informatics, Medical University Graz, Graz, Austria

This research was funded by the French ANRT industrial Cifre PhD contracts with SEGULA Technologies and with Tinubu Square. Holzinger received funding by the Austrian Science Fund (FWF), Project: P-32554. Díaz-Rodríguez was supported by Juan de la Cierva Incorporación grant IJC2019-039152-I funded by MCIN/AEI /10.13039/501100011033 by "ESF Investing in your future", a MSCA Postdoctoral Fellowship (Grant agreement ID 101059332), Google Research Scholar Program, and a 2022 Leonardo Grant for Researchers and Cultural Creators from BBVA Foundation (the Foundation takes no responsibility for the opinions, statements and contents of this project, which are entirely the responsibility of its authors). Tulli is supported by the European Union's Horizon 2020 research and innovation programme under grant agreement No 765955 (ANIMATAS Innovative Training Network). Holzinger is supported by the European Union's Horizon 2020 research and innovation programme under grant agreement No. 826078 (Feature Cloud). This publication reflects only the authors' view, and the European Commission is not responsible for any use that may be made of the information it contains. We acknowledge the support of the PNRR project INEST - Interconnected North-East Innovation Ecosystem (ECS00000043), under the NRRP MUR program funded by the NextGenerationEU. We acknowledge the support of the PNRR project FAIR - Future AI Research (PE00000013), under the NRRP MUR program funded by the NextGenerationEU .
Authors' Contact Information: Adrien Bennetot, Sorbonne Université, Paris, France; e-mail: adrien.bennetot.upmc@gmail.com; Ivan Donadello, Free University of Bozen-Bolzano, Bolzano, Trentino-Alto Adige, Italy; e-mail: ivan.donadello@unibz.it; Ayoub El Qadi El Haouari, Sorbonne Universite, Paris, Île-de-France, France and Tinubu Square, Paris, France; e-mail: ayoub.el_qadi_el_haouari@etu.sorbonne-universite.fr; Mauro Dragoni, Fondazione Bruno Kessler, Trento, Italy; e-mail: dragoni@fbk.eu; Thomas Frossard, Tinubu Square, Paris, France; e-mail: tfd@tinubu.com; Benedikt Wagner, City University of London, London, London, United Kingdom of Great Britain and Northern Ireland; e-mail: Benedikt.Wagner@city.ac.uk; Anna Sarranti, University of Natural Resources and Life Sciences Vienna, Wien, Wien, Austria; email: anna.saranti@boku.ac.at; Silvia Tulli, Sorbonne Universite, Paris, Île-de-France, France; e-mail: silvia.tulli@gaips.inesc-id.pt; Maria Trocan, Institut Supérieur d'Électronique de Paris (ISEP), Paris, France; e-mail: maria.trocan@isep.fr; Raja Chatila, Sorbonne Universite, Paris, Île-de-France, France; e-mail: raja.chatila@sorbonne-universite.fr; Andreas Holzinger, University of Natural Resources and Life Sciences Vienna, Wien, Wien, Austria and Institute for Medical Informatics, Medical University Graz, Graz, Steierrmark, Austria; e-mail: andreas.holzinger@medunigraz.at; Artur d'Avila Garcez, City University, London, London, United Kingdom of Great Britain and Northern Ireland; e-mail: a.garcez@city.ac.uk; Natalia Díaz-Rodríguez, University of Granada, Granada, Andalucía, Spain; e-mail: ndiaz@decsai.ugr.es .

ACM 0360-0300/2024/11-ART50
https://doi.org/10.1145/3670685

ARTUR D'AVILA GARCEZ, City University, London, United Kingdom of Great Britain and Northern Ireland
NATALIA DÍAZ-RODRÍGUEZ, University of Granada, Granada, Spain

The past years have been characterized by an upsurge in opaque automatic decision support systems, such as Deep Neural Networks (DNNs). Although DNNs have great generalization and prediction abilities, it is difficult to obtain detailed explanations for their behavior. As opaque Machine Learning models are increasingly being employed to make important predictions in critical domains, there is a danger of creating and using decisions that are not justifiable or legitimate. Therefore, there is a general agreement on the importance of endowing DNNs with explainability. EXplainable Artificial Intelligence (XAI) techniques can serve to verify and certify model outputs and enhance them with desirable notions such as trustworthiness, accountability, transparency, and fairness. This guide is intended to be the go-to handbook for anyone with a computer science background aiming to obtain an intuitive insight from Machine Learning models accompanied by explanations out-of-the-box. The article aims to rectify the lack of a practical XAI guide by applying XAI techniques, in particular, day-to-day models, datasets and use-cases. In each chapter, the reader will find a description of the proposed method as well as one or several examples of use with Python notebooks. These can be easily modified to be applied to specific applications. We also explain what the prerequisites are for using each technique, what the user will learn about them, and which tasks they are aimed at.

CCS Concepts: • **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability;

Additional Key Words and Phrases: Explainable artificial intelligence, machine learning, deep learning, interpretability, shapley, Grad-CAM, layer-wise relevance propagation, DiCE, counterfactual explanations, TS4NLE, neural-symbolic learning

ACM Reference Format:
Adrien Bennetot, Ivan Donadello, Ayoub El Qadi El Haouari, Mauro Dragoni, Thomas Frossard, Benedikt Wagner, Anna Sarranti, Silvia Tulli, Maria Trocan, Raja Chatila, Andreas Holzinger, Artur d'Avila Garcez, and Natalia Díaz-Rodríguez. 2024. A Practical Tutorial on Explainable AI Techniques. *ACM Comput. Surv.* 57, 2, Article 50 (November 2024), 44 pages. https://doi.org/10.1145/3670685

1 Introduction

In machine learning, it should be acknowledged that system decisions can be faulty, as there are factors intrinsic to machine learning systems that can lead to inadequate system behavior. This is due to several reasons, such as bias in the data or issues with system design (such as network structure, connectivity, optimization process, and code quality management). These factors can impact system performance in terms of accuracy, false negatives, or false positives compared to the ground truth. This is an inevitable feature of DNNs, rather than simply bugs in the system that can be corrected. System performance can be largely insufficient to promote trust in the system, and it could lead to catastrophic outcomes in critical applications where human lives may be at stake. An interesting approach is to consider this situation from a software engineering point of view. Following principles of dependable and robust system design might ensure safer system operation. Therefore, there is a need for tools to understand a machine learning system's behavior and outputs—i.e., the need for explainability. Explainability here refers to making explicit the details and reasons for a model outcome, to make its functioning clear or easy to understand [8].

Explainable AI (XAI) seeks to elucidate the internal workings of a machine learning model, aiming to offer clear explanations regarding the methods, procedures and outputs of the model in a manner comprehensible to users [1]. XAI techniques are increasingly being used by a wider audience and are starting to be applied in multiple fields in industry and academia. More and more techniques are appearing, and it can often be complex to interpret or convert their explanatory

elements into actionable explanations—i.e., explanations that developers or experts can transform into actions to fix the model. This article is a guide to some of the most commonly used XAI methods, producing explanations in common format applied to models ingesting different types of data (image, tabular, textual, and graphs) and models using neural-symbolic computation. We also consider some user interface aspects that seek to offer better interaction between the system and non-technical users, as in the case of neural-symbolic computation models producing natural language explanations and counterfactual explanations.

As black-box Machine Learning models are increasingly employed to make important predictions in critical contexts [16, 21, 59], the demand for transparency is increasing from various stakeholders in AI [72]. This is particularly true in the field of healthcare, where operations need to be trusted for ethical and safety reasons [65]. With this motivation, all techniques presented are accompanied by an example. Some techniques are also presented with a second application in a different domain than the first application to show how our tutorials can be easily adapted to different use cases.

The goal of this guide is to serve as a practical and readily usable tool for any developer wishing to obtain some explanatory elements for the behavior of their deep learning model. These can be used to complement efforts to regulate, audit, and govern AI systems [22]. Contrary to other guides that focus on a specific data type [17], we provide a broader view of XAI methods by addressing most of the data types and issues faced by users wishing to explain their models. Dividing the tutorial based on the user's input data type offers contextual relevance and should help with accessibility and facilitate practical application. Tailoring examples and methods to specific data modalities allows users to better understand unique challenges and nuances, hopefully promoting effective utilization in real-world scenarios. By providing detailed implementation guidance for each data type, this tutorial is also a practical resource, easing the transition from theory to application and offering a better learning experience.

In what follows, we describe an XAI method for each of the most common data types: tabular in Section 2, images in Section 3, and text in Section 4, as well as a general method in Section 5. We accompany these techniques with a description of the neural-symbolic methods used to make explanations interactive in Section 6 as well as a method to render XAI explanations through natural language generation in Section 7. Data, models, and Python *Google Colab* interactive notebooks were made available and are free and open source to be reused with ease, adapted, or used to consolidate learning or teaching. They are presented with a systematic introduction of the fundamental theories and common practices. Use cases and suitability analyses for each application, task, or data type are provided with concrete examples and interactive code. Table 1 summarizes the different methods presented here with their associated working demonstration (*Google Colab* notebooks) to facilitate the reader's exploration of the material. Figure 1 acts as a flowchart/map for the readers to help find a method to use according to their types of data.

2 XAI Techniques for Tabular Data

Model-agnostic XAI techniques are meant to be applied to any machine learning model and are applied after the model has been trained [18]. The flexibility of such methods lies in their ability to explain any machine learning model [75]. In this text, we will explore two different post hoc explanation methods for tabular data: **SHapley Additive exPlanations (SHAP)** [54] and **Diverse Counterfactual Explanations (DiCE)** [62]. We present these methods because they have important advantages in addition to being model-agnostic:

— SHAP has a solid theoretical foundation in game theory, which ensures the validity of the explanation by following mathematical axioms. It provides contrastive explanations, comparing the prediction with the average prediction. This method is suitable for users who

Table 1. The Different XAI Methods Explored in the Article with Associated Links to *Google Colab* Implementations, which Can Be Used to Apply the Methods to Your Own Use-case

XAI method	Datatype	Explanation kind	Task explained	Dataset used
SHAP [54]	Tabular	Feature Importance	Bipolar Disease Prediction	Simula Depresjon [36][1]
Original **SHAP** Repository: https://github.com/slundberg/shap				
SHAP practical case guide: https://colab.research.google.com/drive/1AxdhD-ZkZya57-ePk6Nqg0Z8P2eMu9XX?usp=sharing				
DiCE [62]	Tabular	Counterfactual	Bipolar Disease Prediction	Simula Depresjon [36]
Original **DiCE** Repository: https://github.com/interpretml/DiCE				
DiCE practical case guide: https://colab.research.google.com/drive/12jw91RouPBc9slFwB2OWRwYB6Ckviiv3?usp=sharing				
Transformers Interpret (TI) [68]	Textual	Feature Importance	Sentiment Analysis	MultiNLI corpus [105][2]
Original **TI** Repository: https://github.com/cdpierse/transformers-interpret				
TI practical case guide: https://colab.research.google.com/drive/1XGGXUYNC1M_jlmQUV5dZB3HVdQRXeghd				
Grad-CAM [85]	Image	Visual	Image Classification	TCGA and Target [37][3]
Original **Grad-CAM** Repository: https://keras.io/examples/vision/grad_cam/				
Grad-CAM practical case guide: https://colab.research.google.com/drive/1ZXznvG_G1Y-JyHX9a_x6yKrXHhMp6tpm?usp=sharing				
Layer-wise Relevance Propagation (LRP) [61]	Graph	Visual	Image Classification	TCGA and Target[37]
Original **LRP for Graphs** Repository: https://git.tu-berlin.de/thomas_schnake/demo_gnn_lrp				
LRP for Graphs practical case guide: https://colab.research.google.com/drive/166FYIwxblfrEltkYqY_jiJoAm9VLMweJ?usp=sharing				
Original **GCExplainer** Repository: https://github.com/CharlotteMagister/GCExplainer				
GCExplainer for Graphs practical case guide: https://colab.research.google.com/drive/16ayMlyDzNubxSkpIBHXDvbEgpB88wBBK?usp=sharing				
Logic Tensor Networks (LTN) [87]	Textual	Interactive	Violence Risk Prediction	COMPAS ProPublica[52][4]
Original **LTN** Repository: https://github.com/logictensornetworks/logictensornetworks				
LTN practical case guide: https://colab.research.google.com/drive/1Ip9Yb9gVRSRqaBKY9gOpiWn9pq3LovWG?usp=sharing				
TS4NLE [29]	Textual	Natural Language	Explanation Rendering	UMLS [13][5]
Original **TS4NLE** Repository: https://github.com/ivanDonadello/TS4NLE				
TS4NLE practical case guide: https://colab.research.google.com/drive/1iCVSt7TFMruSzeg5DswLOzOR1n7xATbz				

Note that certain methods can also be used for other data types than the one described in the table, including multimodal data, but only the data types included in this tutorial are mentioned. All guides are accessible via https://github.com/NataliaDiaz/XAI-tutorial.

need to be certain that the explanation provided for the behavior of their model is valid and reflects what has happened within the model.

— DiCE produces counterfactual explanations, one of the best explanation types to understand a model due to its similarity to the human cognitive process [95]. The explanation highlights the minimal changes in the input required to obtain a contrastive output. It allows for the production of explanations that are easily conveyable not only to developers but also to a non-technical audience. This method is perfect for users who want an explanation that is as meaningful and understandable as possible.

We apply these two methods one after the other to provide an understandable and easy-to-modify example of use. We use each method for two use cases, one in finance and one in the medical field, to emphasize their ease of use.

2.1 SHAP Analysis

In this section, we will discuss one of the most widely used model-agnostic methods, SHapley Additive exPlanations (SHAP) [54], and its application to tabular data. SHAP enables us to understand the contribution of each feature to the final prediction by testing the prediction obtained by experimenting with all possible combinations of the presence and absence of features.

To put it simply, SHAP is like trying to determine how much each person should pay on a restaurant bill by analyzing how much it would have cost if the person had not come to the

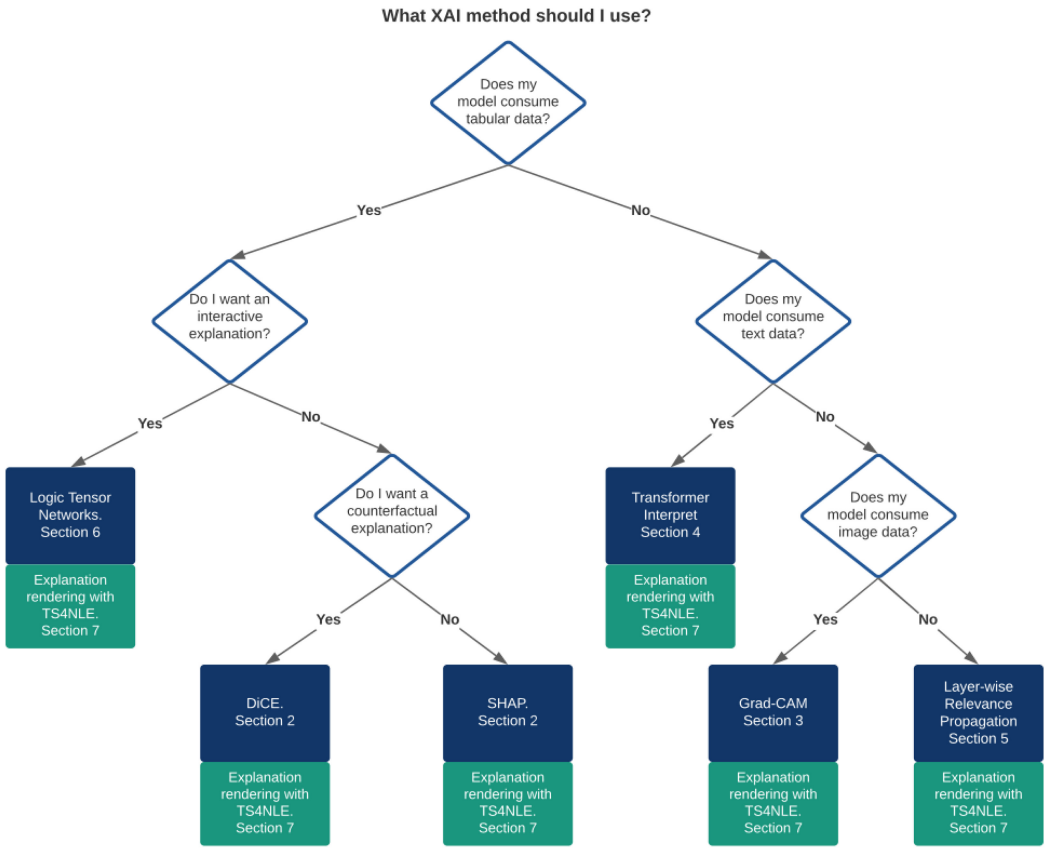

Fig. 1. Map/flowchart of the XAI methods described in the article. This is of course non-exhaustive regarding the existing XAI methods. It only refers to the methods discussed in this article. Moreover, the layer-wise relevance propagation method is recommended when the data are neither textual, tabular, nor images, but it can be used on these types of data as well. Also, SHAP and LTN can be used on images, but we include here only the data types for which Python notebooks were provided.

restaurant. SHAP is based on game theory, specifically Shapley values. Shapley values, originating from cooperative game theory, provide a fair and mathematically rigorous method for allocating the contribution of each participant in a cooperative setting by averaging the marginal contributions over all possible orders in which individuals could join coalitions [88]. SHAP decomposes the prediction of a model among all features involved by using an additive feature attribution analysis:

$$g(x') = \phi_0 + \sum_{i=1}^{M} \phi_i x'_i, \tag{1}$$

where $g(x')$ is the explanation model that matches the original model $f(x)$ when $x = h_x(x')$ and where $x' \in \{0,1\}^M$, M is the number of input features and $\phi_i \in \mathbb{R}$. ϕ_0 represents the baseline model (that is, the model without the feature i) while ϕ_i corresponds to the contribution of feature i to the model prediction:

$$\phi_i = \sum_{S \subseteq N \setminus i} \frac{|\ S\ |!(M - |S| - 1)!}{M!} \left[f_X(S \cup i) - f_X(S) \right], \tag{2}$$

where N is the set of all input features. The inner functioning of SHAP considers, for each feature i, two different models: $f_{S\cup\{i\}}(x)$ and $f_S(x)$. Then, it computes the difference in prediction between both models. Then, the difference in prediction between both models, averaged over all possible coalitions, is the credit attributed to feature i.

Due to the computational expense of considering all possible sets of features S and averaging the difference in prediction due to feature i, SHAP generates a random sample of possible sets of S to compute the average. This average is used to estimate feature importance. SHAP has several desirable properties, including singularity detection (i.e., if the feature is locally zero, then the SHAP value is also zero), local accuracy (i.e., for a specific input x, the explanation model matches the output of the model f for the simplified input x'), and consistency (i.e., if in a second model approximation with a different subset of features the contribution of the feature is higher, then the SHAP value will also be higher).

2.1.1 SHAP Use Case: Predicting Consumers Default. Credit Scoring refers to the problem of deciding whether or not to accept a consumer's loan application. Such models assist lenders and are based on different kinds of data (e.g., performance of consumer's previous loans, financial information, personal data). In this particular use case, we build a model based on **Extreme Gradient Boosting (XGBoost)** to assess the probability of default of the loan applicant [20]. Since in the financial industry, and especially in the credit scoring field, there is a need to understand the decision-making process, we apply SHAP to understand the reasons behind the model decision-making. XGBoost will predict a probability that subsequently will be mapped into two classes. If the probability of the assessed loan is greater or equal to 0.5, then the model will predict that the loan will be amortized. Otherwise, the customer will not be able to return the loan. This is an excellent case to provide a counterfactual explanation scenario: "If criterion x was different, then the loan would be granted."

XGBoost [20], an ensemble technique that combines tree models with Gradient Boosting, has rapidly gained interest in the credit scoring. The XGBoost works by sequentially adding weak learners (i.e., decision trees) to an ensemble, each one correcting its predecessor.

In the financial industry, and specially in the credit scoring field, there is a need to understand the decision-making process. Such needs have resulted in strict regulations, such as the European General Data Protection Regulation (GDPR) and ethics guidelines for trustworthy AI [5, 23, 26]. Different XAI techniques can tackle the explainability issue of black-box models when treating tabular data. More generally, most methods known as model-agnostic XAI techniques are applicable in this case. In this guide, we introduce SHAP, a technique that reflects the importance level contribution of each feature in the model to a given outcome.

We start by building the model. The data we dispose contains financial information about companies, which are divided into two classes. The majority class (about 99%) represents companies that do not incur into a default the year after. Since the dataset is highly imbalanced, we generate synthetic data using SMOTE [19]. Then, we construct the XGBoost model, which will output the probability that a company will be in default one year after. We compare the results of our model with the credit rating of Tinubu Square (see Reference [73]). Finally, we apply the SHAP framework to understand the main reasons why the model considers that a company is likely to suffer financial difficulties.

We build a loan default predictor, i.e., a model that predicts a probability that is mapped into two classes. If the probability of the assessed loan is greater or equal to 0.5, then the model will predict that the loan will be amortized. Otherwise, the customer will not be able to return the loan.

SHAP facilitates the understanding of the model by displaying what features have been the most relevant for the model and their impact in the final prediction. On one hand, the SHAP analysis

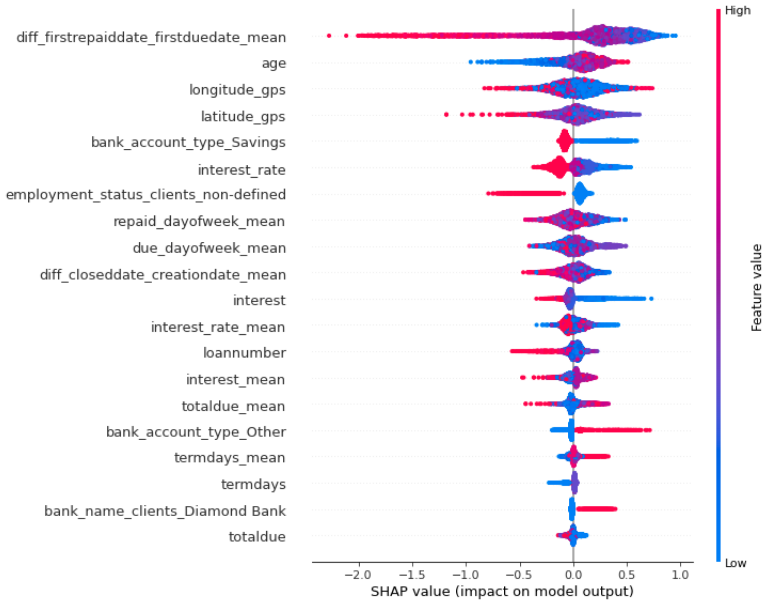

Fig. 2. X-axis represents the SHAP value (i.e., contribution) for each feature of the loan default prediction model, while y-axis indicates the features, ranked by importance from top to bottom. Each point represents a data point (i.e., a single individual). Feature values are encoded using a color gradient. Negative values in x-axis correspond to an increase in the probability of default.

shows us what variables influenced model's output the most. On the other hand, SHAP analysis does not explain how the magnitude of the different features affects the output of the model. This is mainly due the following reasons: highly imbalanced, considerable volume of missing data, and the scarce features used.

The contribution of each explanatory feature to the final prediction is based on a Shapley analysis of contribution decomposition for the default prediction. X axis represents the feature contribution value (negative values in x axis increase the probability of default and vice versa). Features are sorted according to the their relevance (i.e., SHAP average absolute value). In the studied case, Figure 2 shows at the top the most relevant features for the model when predicting the loan's default: the mean difference between the day the first payment due was paid and the day it had to be paid, the age of the borrower, and his localization (latitude and longitude). The analysis results are intuitive and show that large average delays in the payment of the first payment due increase the probability of default. The SHAP analysis also shows that younger borrowers are more likely to not repay the loan.[1]

2.1.2 SHAP Use Case: Predicting Bipolar Disease. Bipolar disorder, formerly called manic depression, is a mental health condition that causes extreme mood swings that include emotional highs (mania or hypomania) and lows (depression) that often come accompanied by different features (i.e., physical and psychological features). For this particular use case, we build a model based on XGBoost. Then, we apply SHAP to help psychiatrists understand the causes behind a potential patient tendency towards a mania or depression episode.

[1]SHAP guide online: https://colab.research.google.com/drive/1HuhpUAl4s9ZIs3yWHsAExApLxGAv3NuH?usp=sharing refined from https://github.com/slundberg/shap

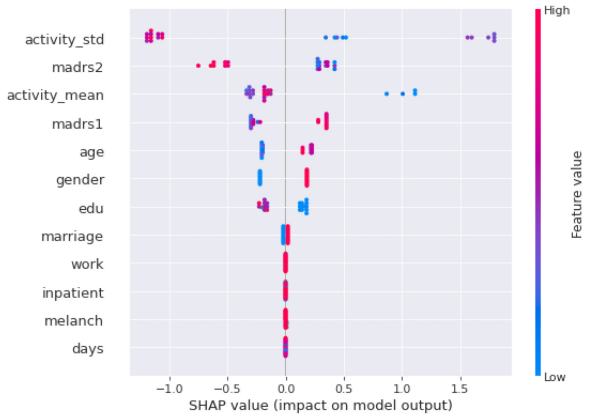

Fig. 3. X-axis represents the SHAP value (i.e., contribution) for each feature of the bipolar disease prediction model, while y-axis indicates the features, ranked by importance from top to bottom. Each point represents a data point (i.e., a single patient). Feature values are encoded using a color gradient. Negative values in x-axis correspond to an increase in the probability of being diagnosed with a bipolar disease.

As explained in the previous use case, the SHAP technique facilitates the understanding of the model by displaying what features had the most impact, i.e., contributed the most to the model prediction. It also reveals how the magnitude of the different features affects—positively or negatively—the probability of suffering a bipolar disease in the future.

In the studied case, Figure 3 shows at the top the most relevant features for the model when predicting a bipolar disease: the variation in the *activity* measurement, the **Montgomery Asberg Depression Rating Scale (MADRS)** at the moment the Actigraph was stopped (*madrs2*), the mean of the *activity* measurement, and the MADRS at the moment the actigraph was initiated (*madrs1*). The analysis results are intuitive and show that high values in the variation of the activity measurements (*activity_std*) decrease the probability of being diagnosed with a bipolar disease.[2]

2.1.3 SHAP Values Suitability Analysis: Pros and Cons. Among additive feature attribution methods, SHAP is the only possible consistent, locally accurate method that obeys the missingness property (i.e., a missing feature gets an attribution of zero). However, it is computationally expensive, since as the number of features increases, the number of possible combinations combinatorially explodes, leading to an expensive computation time. For tree-based models, there is a version of SHAP [53] that allows to compute the exact SHAP values faster, in polynomial time, by keeping track of the number of subsets S that flow into each node of the tree. Another problem is that the Shapley values can change with the order of features selected, and thus, for an exact computation of Shapley values, all possible combinations of subsets must be considered.

Generally, XAI techniques such as SHAP only focus on explaining the model's inner functioning. However, they do not compare the level of alignment of the ML model explanation with human expert interpretations (i.e., psychiatrists). Hence, there is a need for ML models in this domain to meet experts' criteria to allow trust. This is a crucial requirement for ML model adoption in critical decision-making.

[2]SHAP guide online: https://colab.research.google.com/drive/1AxdhD-ZkZya57-ePk6Nqg0Z8P2eMu9XX?usp=sharing adapted from https://github.com/slundberg/shap

2.2 DiCE: Diverse Counterfactual Explanations

Some XAI techniques for tabular data focus on explaining the model by measuring the features that impacted the most the prediction. However, there are methods that explain the model by providing information of feature-perturbed versions of the analyzed instance, e.g., References [77, 104]. These methods fall into the **counterfactual (CF)** explanations methods. In this section, we focus on **Diverse Counterfactual Explanations (DiCE)** [62]. Intuitively, we seek to know what would be the minimal change in the model's input that would make its output different, and we turn this search into an optimization problem.

DiCE considers the problem of generating counterfactual explanations from a set of counterfactual, i.e., alternative events, to a given model output.

This is set as an optimization problem. Ideally, the set of CF examples should balance the variety of the suggested CF instances (diversity) with the capability of the stakeholder to meet the suggested changes (proximity) proposed by the CF framework. Furthermore, the CF explanations need to be aligned with human experts' criteria.

We present term-by-term the elements of the function to minimize [62]. The first term to be encoded mathematically is the concept of diversity. Diversity is captured building on **determinantal point processes (DPP)**, a method for solving the subset selection problem with diversity constraints. A DPP is a mathematical model for random point patterns that exhibit repulsion between points. The probability distribution of the points is determined by the determinant of a kernel matrix, reflecting the pairwise interactions and ensuring diversity in selected subsets, making DPPs useful in applications such as diverse subset selection and recommendation systems [93].

$$dpp\_diversity = det(K), \tag{3}$$

where det is the computation of the determinant of matrix K with $K_{i,j} = \frac{1}{1+dist(c_i,c_j)}$, and $dist(c_i, c_j)$ is a distance metric. c_i represents each generated counterfactual explanation.

Proximity is quantified as the negative distance between the CF example's features and the original input's. For each generated CF (c_i), we compute the distance between the CF and the input x.

$$Proximity := -\frac{1}{k}\sum_{i=1}^{k} dist(c_i, x). \tag{4}$$

C is the generated set of k CFs generated for example x that minimizes the following function (as in Reference [62]):

$$C(x) = arg\ min_{c_1,c_2,\ldots,c_k} \frac{1}{k}\sum_{i=1}^{k} yloss(f(c_i), y) \tag{5}$$

$$+ \frac{\lambda_1}{k}\sum_{i=1}^{k} dist(c_i, x) - \lambda_2 dpp\_diversity(c_1, c_2, \ldots, c_k),$$

where $yloss()$ measures the distance between the output of the model for the CF generated $f(c_i)$ and the output we desire, i.e., the generated CF example. C is the set of k CFs that are close to example x, with a high diversity within the CF generated and for which the outcome of the model is as close as possible to the desired class. Both λ_1 and λ_2 are hyperparameters that balance the three parts of the loss function.

2.2.1 DiCE Use Case: Predicting Consumer's Default. We will focus on the use case presented in Section 2.1.1. As mentioned before, the credit scoring is used to assess the probability that a borrower will not repay his credit. These models are used to identify the cases where the loan must

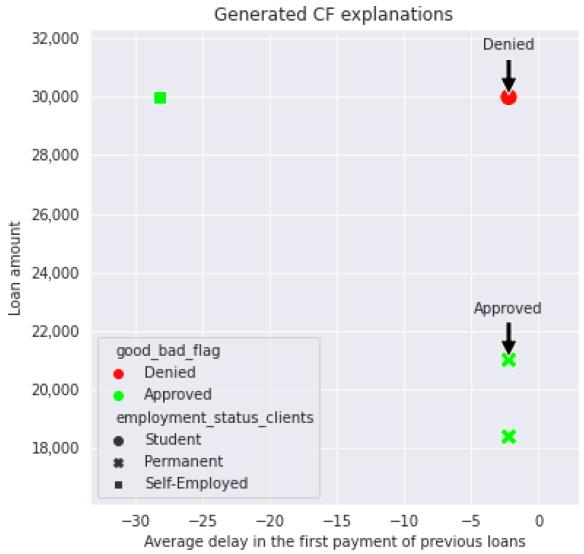

Fig. 4. Counterfactual examples generated using DiCE [62]. Green points are cases where the loan applicant would be approved. We selected a set of variables that are the easiest for the customer to vary (i.e., employment status, the behavior on the payment of previous loans, and the loan amount demanded). We generate three different CF examples for the consumer, but the amount of CFs to be generated can be decided by the user of DiCE or the applicant—if unsatisfied with previous CFs. However, the number of CF that can be generated is limited by the restrictions imposed to create new CF examples.

be denied. In this particular use case, the interest is in advising the customer (i.e., the stakeholder) what to do to get the credit. We are interested in generating counterfactual examples for the clients with denied loans.[3]

In Figure 4, we generate three counterfactual explanations for a given loan applicant. Initially, the considered applicant has been denied the loan by the Random Forest [15] model. The CF examples show which features the applicant should change to get the loan approved. For example, if the applicant had a job and the amount of the loan was smaller, then the loan would have been approved (bottom right point in the Figure 4).

2.2.2 DiCE Use Case: Predicting Bipolar Disease. In this particular use case, as presented in Section 2.1.2, there is a particular interest in providing psychiatrists with the criteria that would change the model output (i.e., a change in the diagnosis of bipolar disease)

In Table 2, we generate two counterfactual explanations for a given patient. The trained model (i.e., Random Forest) predicts a given patient has no bipolar disease. CF examples show which features should change to be diagnosed with bipolar disease. The example in Table 2 shows that if the patient was older, then the diagnosis (CF instance 2 in Table 2) would have been positive (i.e., Bipolar Disease).

2.2.3 DiCE Analysis: Pros and Cons. Advantages of using DiCE include the agnosticism of the method as a capability of generating high number of unique counterfactual explanations for any given ML model. It equally allows to produce explanations that are easily conveyable not only to developers but also to non-technical audiences. However, currently, the disadvantage of DiCE

[3]DiCE guide available at https://colab.research.google.com/drive/1nUTTTfcCuxsnZmaJpfvLsxRB4FFaORVK?usp=sharing. We implemented DiCE using the framework developed by Mothilal et al.: https://github.com/interpretml/DiCE

Table 2. Comparison between the Features that DiCE Modifies to Change the Model Outcome from 0 (No Bipolar Disease) to 1 (Bipolar Disease) for a Single Patient (i.e., Case)

Patient Data	Age	MADRS 1	MADRS 2	Unipolar Depressive Probability	Bipolar II Probability
Original Data	45–49	24	25	0.72	0.28
CF instance 1	**50–54**	24	**21.2**	0.36	0.64
CF instance 2	**65–69**	24	**20.5**	0.48	0.52

All other features remained fixed.

is that works only for differentiable models, since it uses gradient descent for the optimization process. Gradient descent is an iterative optimization algorithm used in machine learning to adjust the parameters of a model by computing and moving in the direction of the steepest decrease in a cost function [14].

3 XAI Techniques for Image Models

Convolutional Neural Networks (CNNs) and Vision Transformers constitute the state-of-art models in all fundamental computer vision tasks (image classification, object detection, instance segmentation). CNNs are built as a sequence of convolutional and pooling layers that automatically learn and entails extremely complex internal relations between features. At the end of the sequence, one or multiple fully connected layers are used to match the output features map into scores.

While some XAI techniques try to delve inside the network and interpret how the intermediate layers see the external world [92], this guide presents a technique that seeks to understand the decision process of a CNN by mapping back the model output to the input space to see which parts of the image were discriminative for the prediction. This choice is motivated by the simplicity offered by visual understanding of explanatory elements that can be relevant for a wide audience. Thus, there is no prerequisite to understand and use this kind of explanation, which can be useful to anyone who wants to know which pixels are relevant for an image classification. Moreover gradient-based methods are usually faster to compute than model-agnostic methods, which makes it very suitable for people with limited computing power or real-time requirements.

3.1 Grad-CAM Use Case: Image Classification

Gradient-weighted Class Activation Mapping (Grad-CAM) [85] uses the gradients (of any target concept) flowing into the final convolutional layer to produce a coarse localization map, highlighting the important regions in the image for predicting the concept. Intuitively, the Grad-CAM technique makes it possible to know which part of the image contributed the most to the model's prediction. The goal of Grad-CAM is to visualize where in an image a convolutional layer looks for a particular prediction. It examines which regions are active in the feature maps of the final convolutional layers to understand how the CNN makes judgments.

Grad-CAM offers visual justifications for CNN judgments. In contrast to other techniques, the gradient is back-propagated to the last convolutional layer rather than all the way back to the original picture to create a coarse localization map that highlights key areas of the original image. Grad-CAM assigns each neuron a relevance value for the prediction. Explained intuitively, the output of a Grad-CAM is an image of the same size as the input image of our neural network, in which each pixel is colored according to its importance in the final prediction. Different CNNs can be explained with Grad-CAM; it works for any network having using gradient descent.

To obtain the class-discriminative localization map $L^c_{Grad-CAM}$ for any class c, i.e., a visual representation that highlights the key regions in an input, such as an image, relevant for

ALGORITHM 1: Grad-CAM Algorithm: Computing a class activation map as output explanation for a given classified image

Require: Input Image I, Classifier C
Step 1: Isolate the last convolutional layer of model C
$LastLayer \leftarrow C.LastConvolutionalLayer$
Step 2: Create a model mapping the input image to the activations of the last layer
$ActivationMap \leftarrow LastConvModel(I, LastLayer)$
Step 3: Create a model mapping the activations of the last layer to the class predictions
$PredMap \leftarrow PredModel\ (LastLayer.Output, C.Output)$
Step 4: Compute activations of the last layer
$Activations \leftarrow ActivationMap(I)$
Step 5: Compute class predictions
$Predictions \leftarrow PredMap(Activations)$
Step 6: Compute the gradient of the top prediction
$TopPrediction \leftarrow Max(Predictions)$
$GradTopPrediction \leftarrow TopPrediction.Gradient$
Step 7: Multiply each channel in the activation map by the mean of the across the dimensions.
$PooledGradients \leftarrow Pool(GradTopPrediction)$
for $i \in Range(Activations)$ **do**
    $Activations[i] \leftarrow Activations[i] * PooledGradients[i]$
end for
Step 8: Return the heatmap for class C activation as the mean of the activation map:
return $ClassActivationMap \leftarrow Mean(Activations)$

predicting a specific class in tasks like object detection or image classification, Grad-CAM computes the gradient y^c of the score for class c with respect to the feature map activation A^k for feature map k of a convolutional layer. These gradients are global-average-pooled by summing feature map activations $A^k_{i,j}$ over the width i and height j of the activation map containing Z pixels to obtain the neuron importance α^c_k, defined as:

$$\alpha^c_k = \frac{1}{Z} \sum_i \sum_j \frac{\partial y^c}{\partial A^k_{i,j}}. \tag{6}$$

The output of Grad-CAM, heatmap $L^c_{Grad-CAM}$, is obtained by performing a weighted combination of activation maps, he output of intermediate layers in a neural network, with a **Rectifier Linear Unit (ReLU)** activation function. We usually normalize the heatmap and color it to make it more visually interpretable.[4]

$$L^c_{Grad-CAM} = ReLU\left(\sum_k \alpha^c_k A^k\right). \tag{7}$$

The goal of visualizing class activation maps in a CNN is to ensure that the model is taking a decision for the right reason and that it does not contain any inner bias due to learned spurious correlations or purposely misleading selected data. Taking as example Figure 5, a binary classifier was trained to classify pictures of lymphoid tissue and esophagus membrane. We want the model to be able to predict that the RGB input image represents an esophagus membrane, because it

[4]Grad-CAM guide available at: https://colab.research.google.com/drive/1ZXznvG_G1Y-JyHX9a_x6yKrXHhMp6tpm adapted to a medical use case from https://pyimagesearch.com/2020/03/09/ and https://colab.research.google.com/drive/1bA2Fg8TFbI5YyZyX3zyrPcT3TuxCLHEC?usp=sharing adapted to an additional domain from https://keras.io/examples/vision/grad_cam/

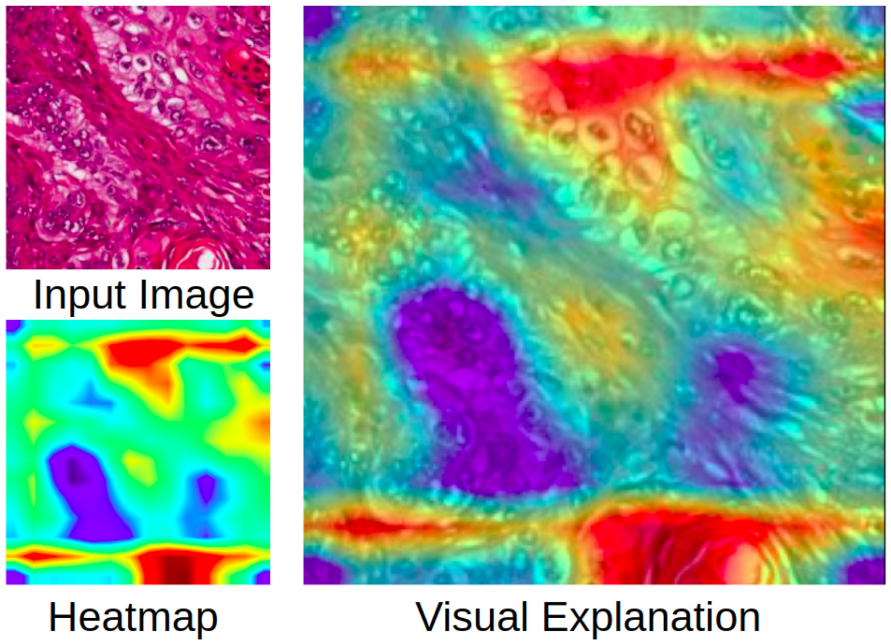

Fig. 5. Grad-CAM example on a medical image representing an esophagus membrane and classified as such by a binary classification model using VGG16 trained on TCGA (Cancer Genome Atlas).[5] Superimposed visualization of Grad-CAM's heatmap and the input image, showing the model mostly used the bottom and the top of the image to make its prediction.

contains features typical of an esophagus. An expert in the medical field would be able to verify that the hottest regions are the ones that should be used to make the prediction.

3.2 Grad-CAM Suitability Analysis: Pros and Cons

The Grad-CAM technique has the advantage of being easy to understand, as the explanation is visual and easy to implement for any gradient-based model, as it does not require to modify the model architecture. However, the interpretation of the heatmap is subjective and therefore induces a human bias, since the explanation does not come directly from the model but from an interpretation that the user makes, based on what it is able to recognize on the heatmap [30]. Also, the Grad-CAM is class-discriminative but lacks the ability to show fine-grained importance, as the heatmap is coarse and not in high-resolution. This means that it can determine globally which region contributed the most to detecting a certain class, but not precisely which pixels. Guided Grad-CAM [84] proposes a high-resolution class-discriminative visualization by combining Grad-CAM with existing fine-grained visualizations. Nevertheless, the validity of explanations obtained by saliency-based techniques can be misleading [3], as it was shown the relationship between "good" saliency and generalization performance is tenuous and that improved generalization is not always accompanied by improved heatmaps [97].

4 XAI Techniques for Language Models

Most of the information available worldwide is in text form, from legal documents to medical reports. A variety of deep learning models have been applied to improve and automate complex

[5] Available at the GDC Data Portal https://portal.gdc.cancer.gov/

language tasks. Examples of such tasks include, but are not limited to, tokenization, text classification, speech recognition, machine translation, and document summarizing.

Among the existing **Natural Language Processing (NLP)** models, we analyzed transformers models for two pivotal reasons: They rely on the attention mechanism (i.e., initially designed for neural machine translation), and they are exceptionally effective for common **natural language understanding (NLU)** and **natural language generation (NLG)** tasks [96]. Thus, this section of the tutorial focuses on a method for explaining the outputs of transformer models, specifically for NLP tasks. As the use of this method in this context is specific to transformers, we begin this section with a reminder of this architecture.

The Transformer architecture functions by assessing the significance of different parts of the input data. Initially, it deconstructs the sentence into individual words or parts of words (tokens). To grasp the context of each word—how each word is related to every other word in the sentence—the model examines all the words in a sentence simultaneously. Subsequently, the model employs the so-called *attention mechanism* to emphasize the parts of the sentence that are more relevant to understanding the current word. In text generation, the transformer leverages the context it has acquired to predict the next word in a sequence. This is done by calculating probabilities for numerous potential next words and selecting the one with the highest probability. Following this, it incorporates this new word into the sequence and repeats the process until a complete sentence or paragraph is generated.

When a sentence is decomposed into tokens, which can be words or parts of words, each token is converted into a numerical representation. These numerical representations are then used to produce three distinct types of vectors for every token: query vectors, key vectors, and value vectors. The query vector represents the token that the model is currently analyzing. Every token in the sentence is associated with a key vector. These key vectors are instrumental in determining the compatibility or relationship between the token under analysis (represented by the query vector) and all other tokens in the sentence. The model computes a score by comparing the query vector with each key vector, which indicates how much focus (or weight) should be placed on each token in the sentence in relation to the token being analyzed, i.e., attention score. Tokens with higher scores are considered more relevant to the context of the current token. After calculating the attention scores, the model utilizes the value vectors of each token. These value vectors contain the substantive information about each token. Based on the attention scores, the model calculates a weighted sum of these value vectors, thereby highlighting the information from tokens that are more pertinent.

In practice, the encoder represents the input as a set of key-value pairs, $(\mathcal{K}, \mathcal{V})$, with dimensions d_k and d_v, respectively. The decoder packages the previous output into a query Q of dimension m and obtains the next output by mapping this query against the set of keys and values [103]. The matrix of outputs, also known as the score matrix, determines the importance of a specific word with respect to other words.

The score matrix is the result of a scaled dot-product, where the weight assigned to each output is determined by the dot-product of the query with all keys (Equation (8)).

$$\text{Attention}(Q, \mathcal{K}, \mathcal{V}) = \text{softmax}\left(\frac{Q\mathcal{K}^\top}{\sqrt{dk}}\right)\mathcal{V}. \tag{8}$$

The attention mechanism repeats h times with different, learned linear projections of the queries, keys and values to dk, dk, and dv dimensions, respectively. The independent attention outputs of each learned projection are then concatenated and linearly transformed into the expected dimension [103].

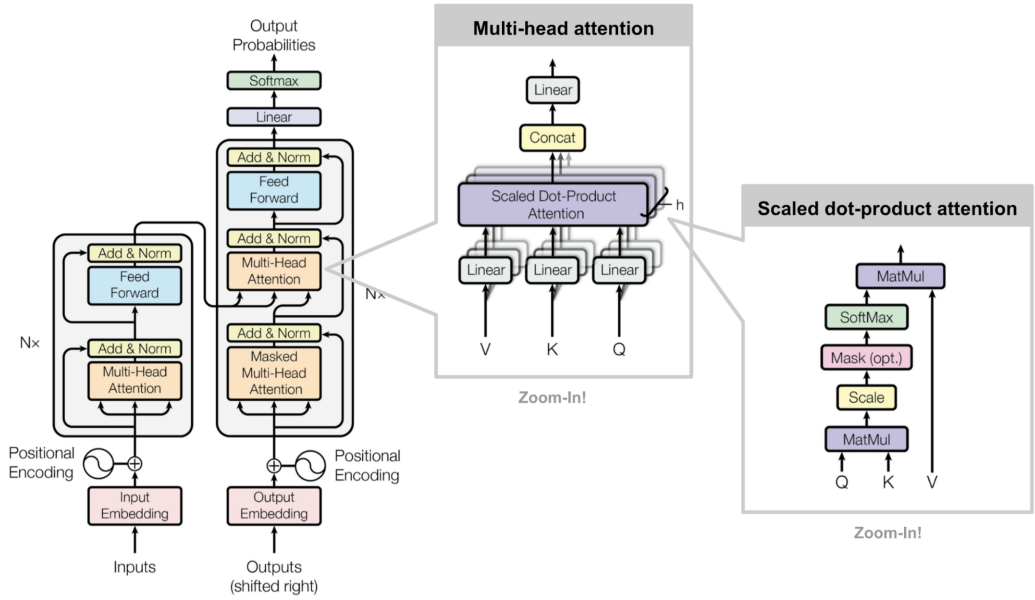

Fig. 6. High-level overview of the BERT Transformer model: The input comprises a sequence of tokens embedded into vectors, while the output consists of a sequence of vectors linked to the input tokens via index. The encoder employs a multi-head attention mechanism to compute queries, keys, and values from the encoder states. Additionally, the encoder feed-forward network incorporates information from other tokens to enhance model integration. However, the decoder utilizes a masked multi-head attention mechanism to compute queries, keys, and values from the decoder states. Furthermore, the decoder's multi-head attention mechanism examines the source of the target tokens, drawing queries from the decoder states and keys and values from the encoder states. Finally, the decoder feed-forward network incorporates additional token information, further enhancing model integration (original image from Reference [96]).

$$\begin{aligned} \text{MultiHead}(\mathcal{Q},\mathcal{K},\mathcal{V}) &= \text{Concat}\,(\text{head}_1,\ldots,\ \text{head}_{\text{h}})\,\mathcal{W}^O, \\ \text{where head}_{\text{i}} &= \text{Attention}\left(\mathcal{Q}\mathcal{W}_i^Q,\mathcal{K}\mathcal{W}_i^K,\mathcal{V}\mathcal{W}_i^V\right). \end{aligned} \tag{9}$$

In multi-head attention (Equation (9)), h represents the number of parallel attention layers (heads), and the $\mathcal{W}$s are learnable parameter matrices, specifically, $\mathcal{W}i^Q \in \mathbb{R}^{d\text{model}\times d_k}$, $\mathcal{W}i^K \in \mathbb{R}^{d\text{model}\times d_k}$, $\mathcal{W}i^V \in \mathbb{R}^{d\text{model}\times d_v}$, and $\mathcal{W}^O \in \mathbb{R}^{hd_v\times d_{\text{model}}}$.

4.1 An Example of Transformer Architecture: Bidirectional Encoder Representations from Transformers (BERT)

Bidirectional Encoder Representations from Transformers (BERT) [24] is a machine learning technique based on transformers for pre-training in NLP, developed by Google. The distinctiveness of this technique lies in its application of bidirectional training to understand the context of a word based on all of its surroundings (both left and right of the word) rather than just the words that precede it, as in traditional unidirectional language models (e.g., OpenAI GPT [74], ELMo [67]). This way of processing words allows BERT to learn to unambiguously contextualize words. By repeating this process multiple times (i.e., multi-head), BERT can learn different contexts between different pairs of words. Figure 6 describes the BERT architecture. To define the prediction goal, BERT utilizes two techniques: **Masked Language Modeling (MLM)**, inspired by the Cloze Procedure [91], and **Next Sequence Prediction (NSP)** [47]. The former

involves substituting approximately 15% of the tokens with a mask token and querying the model to predict the values of the masked tokens based on the surrounding words. The latter involves training the model by providing pairs of sentences as input to learn to predict whether the second sentence in the pair is the subsequent sentence in the original document.

Due to the increased attention received by Transformer models, there are numerous interfaces available for exploring their inner workings and interpreting their decisions, including those made by BERT.

Captum[6] is a comprehensive toolkit developed on top of PyTorch, designed to enhance model interpretability. At its core, it offers a range of attribution algorithms, which are techniques used to assign importance scores to different input features or components of a model. These scores help explain why a model made a particular prediction, thus improving our understanding of its decision-making process.

Attribution algorithms provided by Captum can be categorized into three main groups:

— Primary Attribution Algorithms: These algorithms focus on attributing the model's output predictions directly to its input features. By quantifying the importance of each input feature, they reveal which features are most influential in driving the model's decisions.
— Layer Attribution Algorithms: Layer attribution algorithms delve deeper into the model's architecture by attributing output predictions to individual neurons within a hidden layer. This allows for a more nuanced understanding of how different parts of the model contribute to specific predictions.
— Neuron Attribution Algorithms: Neuron attribution algorithms offer a granular analysis by attributing the influence of a single internal neuron to the model's input. This level of detail helps uncover the specific activations and transformations occurring within the model, providing deeper insights into its decision-making process [48].

4.2 Explaining Transformer Models with Transformer-Interpret

Transformer-Interpret is an efficient tool that relies on Captum and HuggingFace[7] pre-trained models. It offers user-friendly methods to explain most common natural language processing tasks performed by Transformer models, including sequence classification, zero-shot classification, and question answering:

— Sequence Classification: assigns a category to a whole text sequence based on overall content, like labeling texts with their sentiment.
— Zero-shot Classification: classifies texts into categories unseen during training, using the model's ability to generalize based on context and semantics.
— Question Answering: generates or identifies answers to questions from a given text, focusing on the model's comprehension and information retrieval capabilities.

Thanks to attribution methods, Transformer-Interpret elucidates the workings behind these tasks. The default attribution method utilized by Transformer-Interpret is **Integrated Gradients (IG)** [90], which visualizes the importance of input features in the model's predictions. To achieve this, IG computes the integral of gradients with respect to inputs along the path from a given baseline to the input.

The integrated gradient along the ith dimension for an input x and baseline x' is defined as follows:

$$IntegratedGrads_i(x) := \left(x_i - x_i'\right) \times \int_{\alpha=0}^{1} \frac{\partial F\left(x' + \alpha \times (x - x')\right)}{\partial x_i} d\alpha, \tag{10}$$

[6]Captum: https://github.com/pytorch/captum
[7]HugginFace: https://huggingface.co/

where $\frac{\partial F(x)}{\partial x_i}$ is the gradient $F(x)$ along ith dimension. α is the scaling coefficient.

IG's aims to satisfy two desirable axioms for an attribution mechanism:

— *Sensitivity.* If a modification in a feature's value leads to a change in the classification output, then that feature should have a non-zero attribution, as it means this feature must have played a role in the classification.
— *Implementation Invariance.* The attribution method result should not depend on the parameters of the neural network, i.e., two neural networks giving the same output for a certain input should have the same attribution even if their weights are different.

Early attempts at interpretability in neural networks relied on gradient-based feature importance scores, which, while implementation-invariant, often fell short of meeting the sensitivity criterion. A feature's modification does not always result in a non-zero gradient, potentially obscuring its actual influence on the model's output. Integrated Gradients addresses this limitation by establishing a baseline—a hypothetical input designed to represent an "uninformative" state from which any deviation is meaningful–and calculating the cumulative gradient effect from this baseline to the actual input. This cumulative effect elucidates how variations in a feature's value from the baseline contribute to changes in the output, thereby integrating sensitivity into gradient-based attribution.

This is fundamental to the methods used to explain Transformer tasks by Transformer-Interpret. For the purpose of this tutorial, we will concentrate on sentiment analysis and zero-shot classification, applying them to COVID-19 content on Twitter. Sentiment analysis is a type of sequence classification where the goal is to determine the sentiment expressed in a piece of text, such as positive, negative, or neutral. When applied to COVID-19 content on Twitter, sentiment analysis can help understand public opinion and emotions regarding the pandemic. Zero-shot classification is particularly useful for analyzing Twitter data, because it allows the classification of tweets into newly emerging categories without retraining the model. For example, as the pandemic evolves, new variants, symptoms, or treatments may become topics of discussion. Zero-shot classification can identify these new categories based on the context provided by the tweet, even if the model has never seen them before.

The sections that follow describe the functionality of two tools offered by Transformer-Interpret, which are useful for our analysis: the Sequence Classification Explainer, used for sentiment analysis, and the Zero-Shot Classification Explainer, used for identifying emerging categories or sentiments.

4.2.1 Sequence Classification Explainer. Consider a model designed for sentiment analysis, categorizing sentiments into three distinct labels: "positive" (0), "neutral" (1), and "negative" (2). Sequence Classification Explainer helps us specify the target sentiment by its index, such as index 2 for "negative" sentiment, to compute attributions.

The explainer provides insights into the model's decision-making process by analyzing different embedding types. By default, it examines word embeddings ("0"), focusing on the semantic meaning of words. Alternatively, it allows for the exploration of position embeddings ("1"), assessing the impact of word placement within the sequence on the model's classification.

4.2.2 Zero-Shot Classification Explainer. Zero-Shot Classification involves assigning labels to text examples without having seen examples of those labels during training. This task is particularly challenging, as it requires the model to understand and generalize from its training data to unseen categories.

Legend: Negative Neutral Positive

True Label	Predicted Label	Attribution Label	Attribution Score	Word Importance
health	health (0.37)	health	-3.28	[CLS] to stop the pan ##de ##mic it is important that everyone turns up for their shots .
sport	sport (0.17)	sport	-1.88	[CLS] to stop the pan ##de ##mic it is important that everyone turns up for their shots .
vaccine	vaccine (0.30)	vaccine	-3.29	[CLS] to stop the pan ##de ##mic it is important that everyone turns up for their shots .
guns	guns (0.16)	guns	1.74	[CLS] to stop the pan ##de ##mic it is important that everyone turns up for their shots .

Legend: Negative Neutral Positive

True Label	Predicted Label	Attribution Label	Attribution Score	Word Importance
health	health (0.37)	health	-3.28	[CLS] to stop the pan ##de ##mic it is important that everyone turns up for their shots .
sport	sport (0.17)	sport	-1.88	[CLS] to stop the pan ##de ##mic it is important that everyone turns up for their shots .
vaccine	vaccine (0.30)	vaccine	-3.29	[CLS] to stop the pan ##de ##mic it is important that everyone turns up for their shots .
guns	guns (0.16)	guns	1.74	[CLS] to stop the pan ##de ##mic it is important that everyone turns up for their shots .

Fig. 7. Zero-shot classifier on COVID-19 content from Twitter. Data from Reference [63].

The Zero-Shot Classification Explainer provides a way to understand how a model makes these predictions. It does so by returning a table with attributions for every label, showing the contribution of different parts of the input text to the assignment of each label.

For example, when analyzing tweets related to COVID-19 to categorize them into themes such as "public health advice," "misinformation," or "vaccine updates" without having been explicitly trained on these labels, the explainer aids in identifying which words or phrases are most influential in determining each category.

Figure 7 shows the application of the Zero-Shot Classification Explainer to categorize tweets related to COVID-19 into predefined labels. The True Label is the actual category label for the text. Predicted Label is the category label predicted by the model with a probability score in parentheses, and it represents the confidence of the model in its prediction. The Attribution Label is the label for which the attribution score is calculated. The Attribution Score is the numerical value showing the influence of the input text on the predicted label. Negative scores indicate that the text negatively influenced the prediction for that label, while positive scores suggest a positive influence. The Word Importance shows the text being analyzed with certain parts highlighted. The color coding (red for negative, green for positive, and white for neutral) is based on the attribution score and indicates the words or phrases that most significantly impacted the model's prediction for each label. The "CLS" token mentioned in the Word Importance column is a special token used in Transformer models like BERT, which stands for "classification" and is used as an aggregate representation of the input for classification tasks.

From the data shown in the table, we can infer that the model seems to be performing well, as the predicted labels match the true labels. The model is most confident about its prediction for the "health" and "vaccine" labels, as suggested by the higher absolute attribution scores (note that they are negative, which could be due to the model's design where perhaps a negative score indicates a stronger influence for a positive match). The phrase "it is important that everyone turns up for their shots" is a significant contributor to the classifications related to "health" and "vaccine," which makes intuitive sense. The word "shots" in the context of "guns" has a positive attribution score, indicating a different kind of influence on the classification compared to the health and vaccine contexts.

5 XAI Techniques to Explain Image, Text, and Graph Classification Models: Layer-wise Relevance Propagation (LRP)

Layer-wise Relevance Propagation (LRP) is a method that produces a heatmap for every input data sample [61]. The heatmap's data structure and size are the same as the input's, and its

highlighted parts denote the areas of the input that played the highest role in the classification, as Figure 8 shows. LRP is an XAI method that applies to several neural network architectures and thereby to several types of data that they can process. Each subsection next deals with a type of data, showing that LRP is flexible enough to handle different data modalities.

LRP's methodology is based on the Taylor expansion of a function $f(x)$ at point a, as expressed by Equation (11):

$$f(x) = f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \cdots \tag{11}$$

Provided that a neural network is computing a non-linear function $f(\mathbf{x})$ of its input $\mathbf{x}$, the function can be expanded near a root point $\tilde{\mathbf{x}}$. The higher-order terms can be considered negligible and represented by a constant ϵ.

$$f(\mathbf{x}) = f(\tilde{\mathbf{x}}) + \left(\frac{\partial f}{\partial \mathbf{x}}\bigg|_{\mathbf{x}=\tilde{\mathbf{x}}}\right)^T (\mathbf{x}-\tilde{\mathbf{x}}) + \epsilon = 0 + \sum_p \underbrace{\frac{\partial f}{x_p}\bigg|_{\mathbf{x}=\tilde{\mathbf{x}}} (x_p - \tilde{x}_p)}_{R_p(\mathbf{x})} + \epsilon. \tag{12}$$

Since $f(\tilde{\mathbf{x}}) = 0$, and assuming without loss of generality that $\tilde{\mathbf{x}}$ is an image composed by pixels p, one can re-write Equation (12) as follows:

$$f(\mathbf{x}) = 0 + \sum_p \underbrace{\frac{\partial f}{x_p}\bigg|_{\mathbf{x}=\tilde{\mathbf{x}}} (x_p - \tilde{x}_p)}_{R_p(\mathbf{x})} + \epsilon. \tag{13}$$

The goal of LRP is to redistribute the neural network output onto the input variables; i.e., the relevance R_j to lower-level relevances $\{R_i\}$. Starting from the output layer, one can restate Equation (12):

$$\begin{aligned} &\textstyle\sum_j R_j = \left(\frac{\partial(\sum_j R_j)}{\partial\{x_i\}}\Big|_{\partial\{\tilde{x}_i\}}\right)^T (\{x_i\} - \{\tilde{x}_i\}) + \epsilon = \\ &\textstyle\sum_i \sum_j \frac{\partial R_j}{\partial x_i}\Big|_{\partial\{\tilde{x}_i\}} (x_i - \tilde{x}_i) + \epsilon. \end{aligned} \tag{14}$$

$$\sum_j R_j = \left(\frac{\partial(\sum_j R_j)}{\partial\{x_i\}}\bigg|_{\partial\{\tilde{x}_i\}}\right)^T (\{x_i\} - \{\tilde{x}_i\}) + \epsilon \tag{15}$$

$$= \sum_i \sum_j \frac{\partial R_j}{\partial x_i}\bigg|_{\partial\{\tilde{x}_i\}} (x_i - \tilde{x}_i) + \epsilon. \tag{16}$$

One of the challenges of LRP is to find a neighboring point $\tilde{\mathbf{x}}$ of $\mathbf{x}$, for which $f(\tilde{\mathbf{x}}) = 0$ (root point). A good root point removes the elements of a datapoint $\mathbf{x}$ that cause $f(\mathbf{x})$ to be positive. For example, in the case of object detection and a classifier that discriminates between images containing an object and images that do not, an optimization method should look for a similar image that contains an object not recognizable from the classifier—hence, the output $f(\tilde{\mathbf{x}}) = 0$. Examples of such images are some that contain blur or have parts that are relevant for the recognition of the object replaced with grey/black (non-informative) pixels.

The main difference between Grad-CAM and LRP is that although both of them compute gradients, the first one computes the gradient concerning the feature maps activations of CNNs, whereas the second one does it also for other types of architectures, not necessarily CNNs, and it is done in a per-neuron basis. Although it also is a heatmapping method, Grad-CAM does not compute

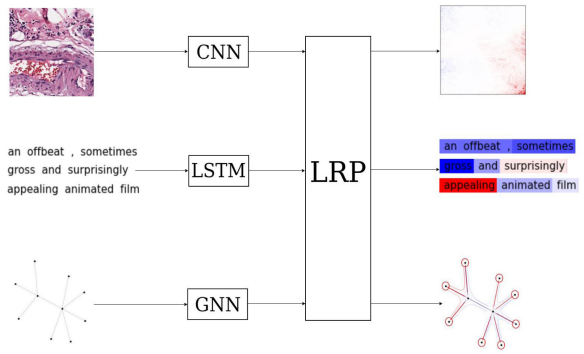

Fig. 8. Layer-wise Relevance Propagation (LRP) applied onto various neural network architectures, each of them processing different types of data.

relevance per se but rather tries to locate the part of the input image that is responsible for the predicted label. Grad-CAM is only applied to CNN architectures (and Graph Convolutional Neural Networks [71]), whereas LRP is more universal and has been customized/adapted for new major neural network architectures.

5.1 LRP Applied onto Fully Connected (FC) Neural Network that Solves a Regression Problem

In this task,[8] the interest is in computing the relevance at particular elements of a small, fully connected neural network. The network consists of only one input layer (its neurons are indexed by (i), one hidden (j) and one output layer (k). x_i represents the values of the input neurons, x_j the outputs of the hidden layer neurons, and x_k the outputs of the output layer neurons. The nonlinear function of the hidden layer is the ReLU, which is expressed by the equation $x_j = max(0, \sum_i x_i w_{ij} + b_j)$. w_{ij} are the weights between the ith and jth layer and b_j the bias (omitted in this task). The nonlinear function performed by the output layer is the sum pooling function, expressed by $x_k = \sum_j x_j$.

The fully connected neural network used in the first task is depicted in Figure 9. Relevance of each neuron at each layer (indexed by i, j, and k correspondingly) is computed by the following set of equations:

$$R_k = x_k = \sum_j x_j, \tag{17}$$

$$, \quad R_j = x_j = max\left(0, \sum_i x_i w_{ij} + b_j\right), \tag{18}$$

$$R_i = \sum_j \frac{w_{ij}^2}{\sum_{i'} w_{i'j}^2} R_j, \tag{19}$$

where i' denotes all neurons of the input layer, including the ith neuron. In this equation, i corresponds to one particular neuron in the input and i' is an index over all of them.

The relevance of the neuron in the output layer is completely specified by the sum of its inputs, since this is its functionality. The relevance of each neuron in layer j is derived by using Equations (13) and (16): $R_j = R_k(\tilde{\mathbf{x}}) + \frac{\partial R_k}{\partial x_j}\Big|_{\{\tilde{x}_j\}} \cdot (x_j - \tilde{x}_j) = x_j = max(0, \sum_i x_i w_{ij} + b_j)$. Since the ReLU

[8]Notebook on LRP for a Fully Connected (FC) Neural Network using synthetic data: https://colab.research.google.com/drive/1Md2Rz3Ff1r05zq98cYndiEqrhg-DGYPv?usp=sharing

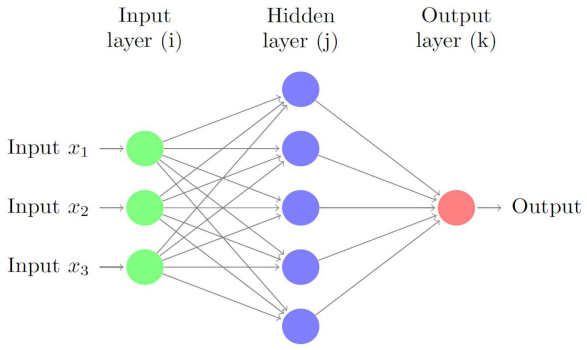

Fig. 9. Fully connected neural network used in the first task. It is composed of 3 neurons in the input layer, 5 neurons in the hidden layer, and 1 neuron in the output layer. The activation function that was used is the Rectified Linear Unit (ReLU).

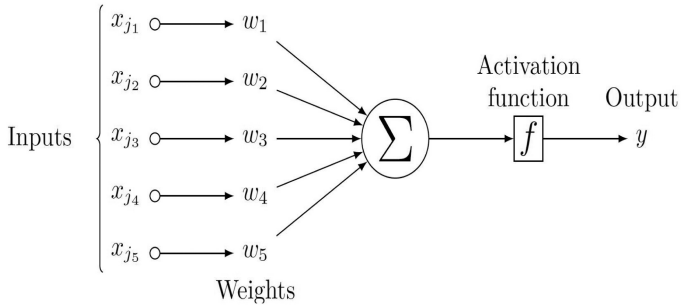

Fig. 10. The functionality of a neuron of a fully connected neural network with five inputs, five weights, the sum of their corresponding multiplication, and the application of a nonlinear activation function.

nonlinearity is used, it is ensured that $\{\forall j : \tilde{x_j} \geq 0\}$ and $\frac{\partial R_k}{\partial x_j} = \frac{\partial \sum_j x_j}{\partial x_j} = 1$. Therefore, the root point $\tilde{\mathbf{x}}$, for which $R_k(\tilde{\mathbf{x}}) = 0$ is $\tilde{\mathbf{x}} = 0$. The computation of the relevances R_i of each neuron in layer i has a derivation that is out of scope for this article. Nevertheless, it is important to note that the relevance is proportional to the squared weight of the connection—keeping in mind that weights can take both negative and positive values.

The relevance of the neurons of each layer (in general) is computed by using the relevances of neurons of the next one.

The overall goal of the task is to understand how the interplay between values of the input and the network weights define the computed relevance values. To better explain this, Figure 10 depicts the functionality of a neuron in a fully connected neural network. For convenience and without loss of generality, we can think that this is the neuron in the output layer k of the task (see also Figure 9). Each input x_{j1} to x_{j5} will be multiplied by a corresponding weight (w_1 to w_5), and then the sum of all those multiplications $x_{j1}w_1 + \cdots + x_{j5}w_5$ comprises the input of the nonlinear activation function. In this case, the **Rectified Linear Unit (ReLU)** was used.

For a fixed input, the exercises deal with two cases: The first one sets weights that are not randomly chosen. Thereby, positive weights multiplied by positive input values will generate a high positive value that in turn will consist of the input to the activation function. Those parts will be, in retrospect, the ones that will have higher relevance in general. On the contrary, weights near zero will "suppress" all highly positive or highly negative input values. The second case is quite the opposite of that: Random weights do not show any preference for input values. By those

means, one can compare the situation of a trained vs. not-trained neural network that has randomly initialized weights.

The overall goal of the task is to understand how the interplay between values of the input and network weights defines the computed relevance values. When the weights are randomly distributed, only the highest input values will manage to create a high activation (with few exceptions). When the weights are not randomly distributed, then some relatively high values might be suppressed (by multiplication with a small weight) and not create a high activation. However, some relatively small values, when multiplied with a high weight value, might induce a high activation. By those means, one can compare the situation of a trained vs. not trained neural network.

It is important to emphasize the difference between the backpropagation procedure, which occurs several times during the training of a neural network, and the backpropagation of relevance, which happens only once after the training is completed. Furthermore, two properties of LRP are used for verification of the computations with unit tests [81], namely, positivity and conservation (Equation (20)) of relevance of the neurons at each layer:

$$\forall x, p : R_p(x) \geq 0, \quad \sum_i R_i = \sum_j R_j, \tag{20}$$

where x is the input, p represents any neuron of the network, and layer i precedes layer j.

5.2 Explaining a GNN Performing Node Classification on Graphs with GNN-LRP

Graph Neural Networks (GNNs) perform three main types of tasks on graph datasets: node classification, link prediction, and graph classification. They can be thought of as an extension of **Convolutional Neural Networks (CNNs)** that process non-grid structured data, therefore the filters cannot operate by the same means.

One of the simplest architectures is called **GCN (Graph Convolutional Network)** [82]. The rules for aggregation and combination (Equation (21)) of the information lying in the features of the neighboring nodes and edges are:

$$\boldsymbol{Z}_t = \boldsymbol{\Lambda} \boldsymbol{H}_{t-1}, \quad \boldsymbol{H}_t = \rho(\boldsymbol{Z}_t \boldsymbol{W}_t), \tag{21}$$

where t denotes the layer, and $\boldsymbol{\Lambda}$ is the Laplacian matrix of the input graph, which can be a scene, protein interaction, social media, a knowledge graph, and so on; $\boldsymbol{\Lambda} H_{t-1}$ is the representation of the previous layer, $\boldsymbol{W}_t$ are the weights, and ρ is the non-linear activation function. The GNN-LRP method [82] applies constraints (piecewise linear and positive homogeneity) to this nonlinearity (here, ReLU is used). Equation (21) is re-written, and the GNN-LRP rule for computing the relevance $R_{jKL\ldots}$ of neuron j after one has processed nodes K and L by all neurons k that gathered information from node K becomes:

$$R_{jKL\ldots} = \sum_{k \in K} \frac{\lambda_{JK} h_j w^{\wedge}_{jk}}{\sum_J \sum_{j \in J} \lambda_{JK} h_j w^{\wedge}_{jk}} R_{kL\ldots}, \tag{22}$$

where K, L are elements (nodes or edges) of a walk on the input graph. A walk on the graph involves nodes that are processed by corresponding neurons at layers labeled processed by neurons with k, l indexes (capital letter subindexes represent nodes, while at the same time, they also denote all neurons with that corresponding non-capital index that process those nodes). Weight $w^{\wedge}_{jk}$ is a weighted sum (denoted by the $\wedge$) of the elements of matrix $\boldsymbol{W}_t$ that links neuron j to neuron k parameterized by a user-provided parameter *gamma*, which for different values, facilitates explanations with varying visual presentation. Typically, a range of *gamma* values is used where each of them corresponds to a different level of detail representation; the data scientist can try them individually for each example graph. For some values of gamma—and in relation to the sign of the

computed relevances—the importance (positive or negative)[9] will be attributed to a few elements, thereby being more "concentrated." For others, more neighbors of a really important element are considered relevant, too, making the presentation of relevances more "coarse." λ_{JK} is the element of the Laplacian matrix $\mathbf{\Lambda}$ corresponding to the connection between nodes J and K, h_j is the activation of neuron j, $R_{kL\ldots}$ is the relevance of neuron k after one has processed node L. The "..." indicates that the walk contains, in general, further nodes.

The task of graph classification contains three parts. At first, Barabasi-Albert graphs are created by the user. These are random scale-free networks that have a preferential attachment mechanism with user-defined growth factors. Nodes and edges can represent anything that can be approximated by a scale-free graph (a graph that has a degree distribution according to the power law). Examples of such graphs are citations and social networks. This growth factor will be the label for the prediction. The GNN that will be used for this prediction is defined in the second part, along with the corresponding functions for training and computing relevances. The third part deals with training the network, using a test set to compute its performance and then applying GNN-LRP to compute and display a plot showing the *walk's* relevances.

The most important goal of this task is to understand what the computational path of a GNN is, and its recursive nature, to comprehend how this leads naturally to the relevance of walks in the graph. Our notebook[10] is a slightly changed version of the researchers' original.[11] The task is studying whether the layer walk relevances are plausible and how they change if the length of the walk on the graph is changed, in juxtaposition with an adaptation of the GNN layer sizes (changing its amount and size in terms of neurons) to obtain a model output and output explanation that is consistent with the ground truth and the expectations of a domain expert.

For example, let us assume we have a scene graph of a medical image with cells and links. Cells are represented by nodes and have various features such as size, color, or shape. It might be that a classification lies on the relevance of the size of a particular node of the graph. Paths that contain this node are expected to have high relevance because of this node; nevertheless, this importance will be distributed to the rest of the elements of this path, although they might be not at all decisive/relevant on their own. To distill that the characteristics of this particular node were responsible, the user is required to compare several walks containing it and its neighboring elements. GNN-LRP might not be the most adequate XAI method for this use case.

One important challenge of all XAI methods is the explainability of misclassified samples. LRP does not at the moment provide a substantial and quantifiable benefit in comparison with other heatmapping methods; nevertheless, the perturbation analysis deals with misclassified samples in a more robust way than other methods, because the performance is influenced (i.e., drops) monotonically after the removal of the relevant elements (nodes or edges) in sorted order.

5.3 Explaining a GNN Performing Node Classification on Graphs with GCExplainer

The GCExplainer is an explanation method that can be applied to both node and graph classification-solving GNN architectures [57]. It has a different way of operating on each layer

[9]The relevance is positive on all elements of an input example that contribute to the prediction in a manner that increases the confidence of the neural network. For example, in a two-class classifier that discriminates between images of animals and cars, an input image containing parts of both classes will be eventually classified as one of the two, but the confidence of the classifier is not expected to be very strong. Assuming that the decision is the car, the areas containing animals should have negative relevance, because they lessen the prediction strength of the classifier. All areas that "speak against" the predicted class are expected to have negative relevance. If they were removed from the image, then they would increase the confidence of the prediction—which is the main principle of counterfactual explanations.

[10]Notebook on LRP for a GNN trained on graph data https://colab.research.google.com/drive/166FYIwxblfrEltkYqY_jiJoAm9VLMweJ?usp=sharing

[11]GNN-LRP: https://git.tu-berlin.de/thomas_schnake/demo_gnn_lrp

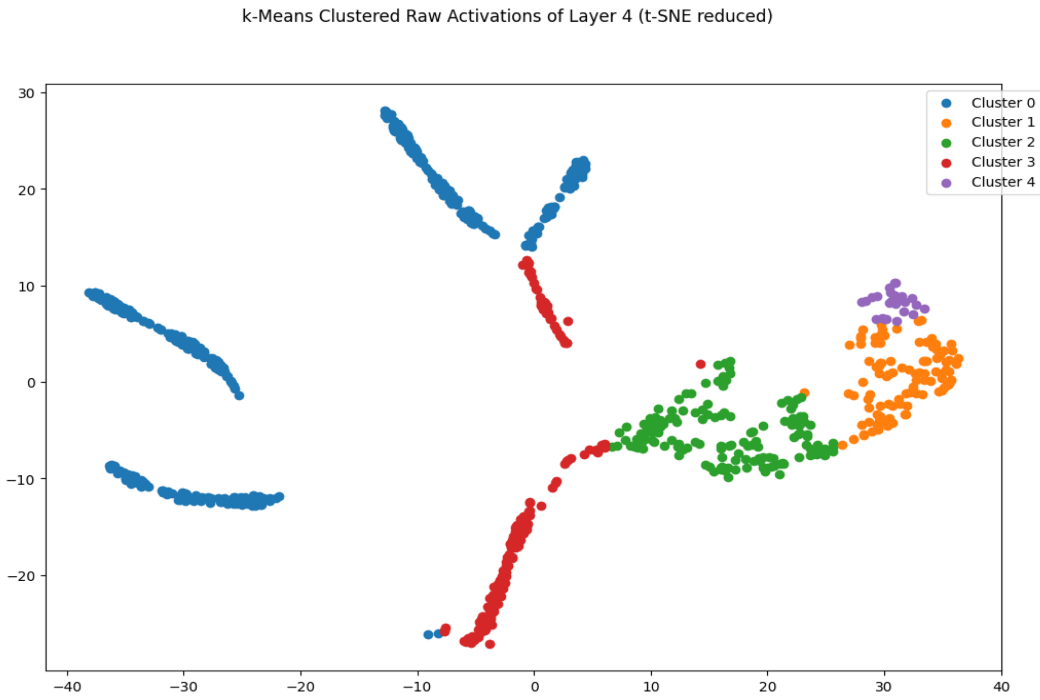

Fig. 11. Results of clustering of the activations of the fourth layer of a Graph Convolutional Neural Network (GCN) [46] with the use of t-distributed Stochastic Neighbor Embedding (t-SNE) clustering method. It is assumed that the number of clusters is $k = 5$ and each of the points in the scatter plot represents one node in the graph that is classified by the GCN.

of the GNN than GNN-RLP, since it follows the assumption that the activation of each layer contains information about concepts relevant to the task solution that are captured by the GNN. This draws a parallel with the way the learned filters of a CNN are formed through training with the ones on the lower layers trained to recognize basic visual structures, whereas the ones on the higher layers are sensitive to combinations of the basic visual structures [4, 50]. Therefore, an inspection of those outputs after training that led to a sufficiently good performance can reveal those concepts and most importantly distinguish them from each other.

Whereas GNN-LRP and GNNExplainer [108]—the first explanation method for GNNs—analyze single instances, the GCExplainer provides global and unsupervised explanations; a possibility that the aforementioned explanation methods can only achieve by aggregating the results of several local explanations. To compute global explanations from single instance explanations, the GCExplainer first applies a dimensionality reduction method—one of **Principal Component Analysis (PCA)** [66], **t-distributed Stochastic Neighbor Embedding (t-SNE)** [94], **Density-based spatial clustering of applications with noise (DBSCAN)** [34], or **Uniform Manifold Approximation and Projection (UMAP)** [58]—on 2 dimensions for the output of each layer of the GNN. This 2-D space can be inspected by the user and an estimation of the number k of potential clusters—each of which will represent a concept—can be made. What is more, the use of the mean Silhouette Coefficient [78] can also provide quantitative information about a good value for k.

After the clustering is finished, all nodes in each cluster as well as their neighborhoods can be visualized as in Figure 11 and Figure 12. The expectation is that all nodes in a cluster will have

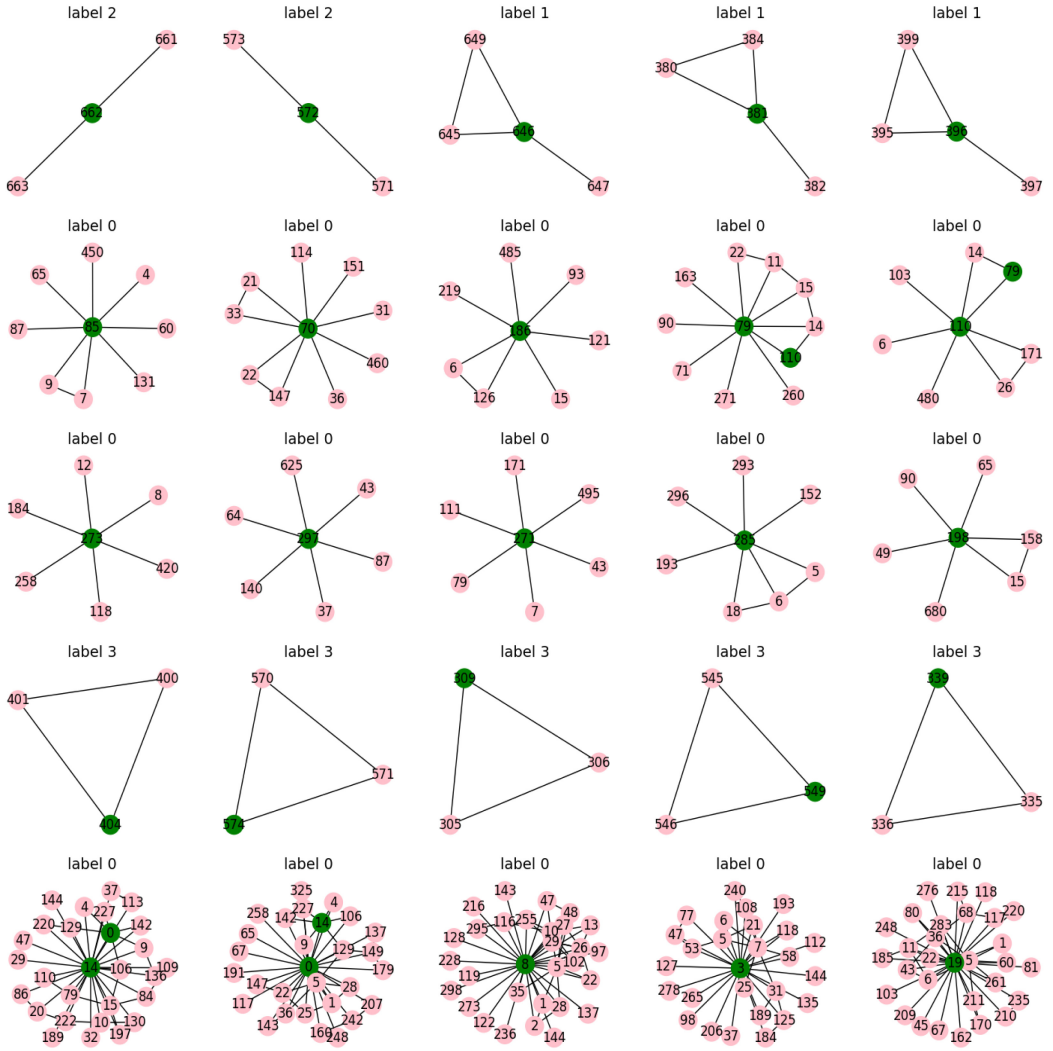

Fig. 12. Depiction of five nodes (in green) as well as their 1-hop neighborhoods (colored pink) that are nearest to the centroid of each cluster. Each row contains the nodes of one concept, and in each row, the leftmost element is the nearest to the cluster's centroid, the second is the second nearest, and so on. By visual observation and without measuring the Graph Edit Distance (GED) between the graphs in each row, one can see the similarity between all subgraphs of one concept. The labels (ground truth, not the predicted ones) of the nodes are also depicted above the subgraph.

similar node feature values and the structure of the graph "around" them will also not deviate substantially. The number of neighborhood hops that are adequate for the user to decide on the coherence of the concept is also an adaptive parameter represented by the variable n, and it can be that multiple concepts are grouped in the same cluster. The higher the n, the more complex the

concept appears to the user. Overall, this is a method that practically implements the Human-in-the-Loop principle [76], where the user adapts two values: the number of the clusters k according to his/her domain knowledge and a clustering metric and the size of the neighborhood through n. The more k increases, the more detailed the concept is and the more fine differences between the concepts appear.

The suitability of this explanation method can also be measured by XAI Quality metrics, as described in several recent research works [31, 83]. A discovered concept's purity can be measured by the **Graph Edit Distance (GED)** [2], which equals the number of operations necessary to transform one graph into another. This is an indicative metric for the graph structure diversity around nodes that are clustered in the same concept, and the higher the n, the less likely it is for this metric to have a value near zero; nevertheless, the features of the nodes' values should also be taken into account for their similarity. The concept completeness score [106] is computed through an individual **Machine Learning (ML)** model (not the original GNN) that performs classification on all inputs that were clustered as belonging to a particular concept. The higher the predictive performance of this classifier, the higher the completeness score; a **Decision Tree (DT)** having sufficiently high performance can be indicative of completeness and at the same time provide decision-making rules as a meta-explanation for reasoning about the coherence and separability of the clustered concepts.

In a further research step, the inventors of the GCExplainer incorporated it in an encompassing framework [45] that measured the overlap between human-expressed concepts and the ones discovered by the method through **Mutual Information (MI)** [56]. This process involves three concrete steps: (1) alignment of human and explainer concepts, (2) refinement of the concepts and parameters of the GCExplainer according to the degree of alignment in the first step, as represented by a **Confusion Matrix (CM)**, and (3) improvement of the concepts' purity and completeness. This process leads to an explanation where all discovered concepts are as pure and complete as possible, comprising a so-called interpretation representation of the dataset [45].

The exercise notebook[12] is a slightly changed version of the researchers' original.[13] Users can adapt the values for the parameters k (number of clusters) and n (number of hops around the explained node) with some additional information about the adequacy of the explanation (explained variance ratio for PCA and silhouette score of the clustering).

5.4 LRP Suitability Analysis: Pros and Cons

The existence of both positive and negative relevance values [7, 79] is one of the impactful properties of LRP. Considering the explanation of one input sample, let us say an image, a heatmap produced by LRP contains both positive and negative values, whereas a heatmap from another explainability method, for example, **Sensitivity Analysis (SA)** [10] only contains negative values. In practical terms, for SA there is no differentiation of which parts of the input enhance the certainty and confidence of a prediction, which ones are an indication of another class (if we are dealing with a classification problem), and which parts are rather neutral (ideally in images in the background). This comprises the forefront for counterfactual explanations [11] and the creation of Probabilistic Graphical Models [98] as explanations, since the sequential removal of positively important components of the input sample have been shown to lead to a monotonic decrease of performance in the case of a correct prediction—and the opposite phenomenon for a misclassified one.

[12]Notebook on GCExplainer for a GNN performing node classification on graph data: https://colab.research.google.com/drive/16ayMlyDzNubxSkpIBHXDvbEgpB88wBBK?usp=sharing

[13]GCExplainer: https://github.com/CharlotteMagister/GCExplainer

This gives the opportunity to domain experts that helped with the preparation of the dataset to change its characteristics in a semi-automated way, since they do not have the time to go through that many heatmaps to understand and improve the behavior of the model. Therefore, Semi-automated Spectral Relevance Analysis was invented [51]. This method was inspired by the detection of a so-called "Clever-Hans" effect in a high-performing neural network that was classifying images of the PASCAL VOC2007 dataset. The neural network learned that this tag was so indicative for accurate prediction of the class "horse" that even experiments with images with cars and this tag were classified as "horse." To be able to semi-automatically separate between explanations that classify an image as "horse" because it actually contains a horse and those that contain some artifact, one can cluster the LRP heatmaps. If the resulting clusters are relatively far from each other, then one can assume that the neural network has found different ways to classify those images and maybe some of them are Clever-Hans.

However, LRP is of great value to software developers and data scientists as well, because they can improve the neural network's architecture by pruning [107], data, feature, loss, and gradient augmentation [102], according to the results of this method. This has been shown to have great benefits in efficiency and explainability as well, and the fact that this is a method applied to different architectures for various types of data opens the path for actionable insights and **Actionable XAI (AxAI)** [80].

On the negative side, LRP is still a method for local explanations, even if the Semi-automated Spectral Relevance Analysis mentioned before has alleviated this fact to a large extent. Inevitably, as one can also see in the GNN-LRP notebook, a node or an edge that belongs to several walks will have a different relevance value w.r.t. the walk that the user is currently observing. Particularly if there are both negative and positive relevance values for one and the same element of the graph, the user has to think actively about the fraction of the neighborhood the walk represents and reason how the components of the graph contributed to potentially diverging values. This is an already studied problem between different XAI methods [49] but not an expected phenomenon in the same XAI method. It is important to remember that the GNN's training process uses all neighboring elements' information when updating the embeddings, therefore the existence of a well-defined and even synthetically generated ground-truth is not an integral solution for this method. However, the GCExplainer's results reflect the influence of the neighborhood in a more cohesive manner, since there is no need to distill the dynamics of relevance in neighboring elements. Nonetheless, both GNN-LRP and GCExplainer lack one important property that is present in the first explainer of GNNs, the GNNExplainer [108], which can identify the important features of nodes and edges (and not just the nodes and edges themselves). A user study that would compare those two as well as explainability methods for GNNs that create causal models (for the purposes of quantifying causability) like the PGMExplainer [98] constitutes valuable future work.

Furthermore, although perturbation analysis and counterfactual explanations are useful for explainability, after the removal or addition of elements in the input sample, the redistribution of relevances w.r.t. the prediction outcome is not always consistent, and there are no extensive user studies. For some architectures in image processing like ResNet [39] that contain residual connections, it has been reported in the GitHub forum that the resulting heatmaps are not as good as for other CNN architectures without residual connections. This is due to the LRP's backpropagation properties; nevertheless, researchers on LRP can extend their method to new architectures, and therefore the scientific community can expect improvements in those deficiencies.

6 Neural-symbolic AI for Interactive Explainability in Neural Networks

Neural-symbolic Learning and Reasoning seeks to integrate principles from neural network learning with logical reasoning. Symbolic systems operate on symbols with reasoning performed over

such abstract, discrete entities, following logical rules. In principle, this process in itself should allow for better explainability than distributed representations at the level of data. Neural networks operate at the sub-symbolic (or *connectionist*) level. Individual neurons do not necessarily represent an abstract entity or a readily recognizable concept.

The integration between both symbolic and sub-symbolic levels as promoted by neural-symbolic systems seeks to bridge data-driven information processing—frequently encountered in perception and pattern recognition—with reasoning and explanation at a higher level of abstraction. Realizing this integration should facilitate a range of benefits, such as achieving representations that are abstract, re-usable, and general-purpose. Having these representations readily available could concretely tackle some of the pressing issues with current deep learning practices.

6.1 XAI in Neural-symbolic AI

Explainability in neural-symbolic systems has been traditionally approached by learning a set of symbolic rules—known as Knowledge Extraction from trained neural networks—and evaluating how well the rules may approximate the behavior of the neural network by measuring the percentage of matching predictions on a test set, which is referred to as the *fidelity* of the extracted symbolic explanation to the neural network.

Most contemporary explainability methods are not powerful enough to guarantee the soundness and completeness of the explanation w.r.t. the underlying model, which is typically a very large neural network. Metrics such as fidelity lack a reliable way of expressing uncertainty associated with the lack of completeness results.

The measured fidelity is seen as a proxy for how close an explanation might be to the representation of the underlying model. By the same token, it is not a metric of the capacity to find semantically meaningful representations. For this, a more interactive approach is needed, as outlined in the caption of Figure 13.

6.2 Framework for Interactive Explainability

In a tightly integrated neural-symbolic system, XAI occurs as part of the neural-symbolic cycle. In this framework, we can query and revise information and consolidate existing background knowledge. The system can utilize background knowledge to provide meaningful semantics for the explanations, facilitating human-machine interaction and hopefully achieving the desired higher-level properties.

By applying the neural-symbolic cycle multiple times, partial symbolic descriptions of the knowledge encoded in the deep network can be checked and, through a human-in-the-loop approach [42], incorporated into the cycle as a constraint on the learning process. This enables an interactive integration of a desired behavior, notably fairness constraints, by validating and incorporating knowledge at each cycle, instead of (global or local) XAI serving only to produce a one-off description of a static system.

Therefore, the neural-symbolic cycle can be seen as a common ground for communication and system interaction. Symbolic knowledge representation extracted from the learning system at an adequate level of abstraction for communication with users should allow knowledge consolidation and targeted revision. The key challenge, therefore, is the efficient extraction of this abstract knowledge from very large networks.

We shall illustrate how the **Logic Tensor Network (LTN)** framework is used for explainable classification, subsequently addressing some undesired model properties according to the pipeline in Figure 13. In the example shown in the figure, we use the Shapley method, but any other XAI method could have been chosen. A logical neural network querying mechanism is then integrated into the process to produce insight into the model during the knowledge revision process.

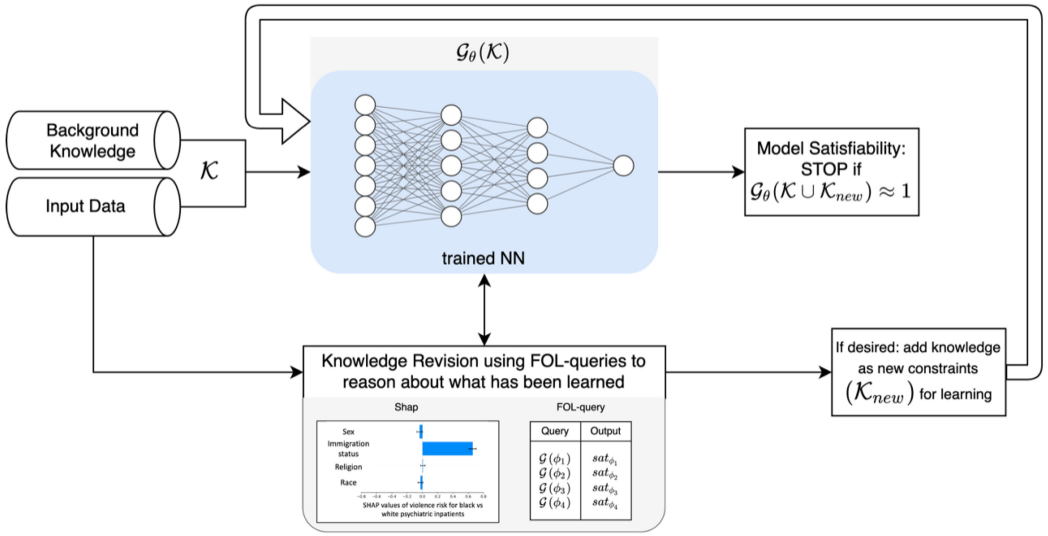

Fig. 13. Illustration of the LTN pipeline for continual interactive learning: Revision is carried out by querying a deep network interactively and learning continually, thus applying the neural-symbolic cycle multiple times. Explanations extracted from the network using, e.g., SHAP, can highlight bias or undesired properties in feature importance. Equally, querying the network in LTN-style shows the satisfiability of specific model properties, such as fairness constraints, which can subsequently be added to the knowledge base $\mathcal{K}$ for further training. Doing this, we can answer questions such as: *How does the model behave for a specific group of individuals compared to others?*, translating into FOL queries and checking their degree of satisfiability (*sat()*) (cf. Section 6.4). Subsequently, such desired queries can be added to the optimization function. This process concludes once it has been shown to reduce bias at a subsequent SHAP explanation.

6.3 Logic Tensor Networks (LTN) for Explainable Model Revision

The framework used in this approach and accompanying notebook[14] is LTN [9, 86, 87]. However, instead of treating the learning of the parameters from data and knowledge as a single process, we emphasize the dynamic and flexible XAI nature of the process of training from data, querying the trained model for knowledge, and adding knowledge in the form of constraints for further training, as part of a cycle whose stopping criteria are governed by a fairness metric. Furthermore, we focus on the core of the LTN approach: constraint-based learning from data and **first order logic (FOL)** knowledge. We make the explanation approach iterative by saving the learned parametrization at each cycle in our implementation. For simplicity, we also replace the original LTN implementation, which uses Neural Tensor Networks with a feed-forward Neural Network. This also demonstrates the agnostic nature of the approach: LTN-based XAI is independent of the network model.

Whereas many inherently neural-symbolic methods come with stringent architectural constraints on the model itself, our LTN adaptation is model-agnostic, since LTN as a framework solely requires the ability to query any deep network (or any ML model) for its behavior, that is, observing the value of an output given a pre-defined input. The predictive model itself can be chosen independently, with the LTN acting as an interface to provide an explanation of the model to the user in the form of targeted FOL queries.

Logic Tensor Networks [9, 87] implement a many-valued FOL language $\mathcal{L}$, which consists of a set of constants $\mathcal{C}$, variables $\mathcal{X}$, function symbols $\mathcal{F}$, and predicate symbols $\mathcal{P}$. Logical formulas in $\mathcal{L}$ allow to specify background knowledge related to the task at hand. The syntax in LTN is that of

FOL, with formulas consisting of predicate symbols and connectives for negations ($\neg$), conjunction, disjunction and implication ($\wedge, \vee, \rightarrow$), and quantifiers ($\forall, \exists$).

Learning in the LTN framework for explanation: LTN functions and predicates are learnable. Thus, the grounding of symbols depends on a set of parameters θ. With a choice of a multilayer perceptron as model, each logical predicate is represented by a feed-forward mapping, where σ denotes the sigmoid activation function, which ensures that predicate P is mapped from $\mathbb{R}^{mxn}$ to a truth-value in $[0, 1]$.

Since the grounding of a formula $\mathcal{G}_\theta(\phi)$ denotes the degree of truth of ϕ, one direct training signal is the degree of truth of the formulas in the knowledge base $\mathcal{K}$. The aggregate truth-value of all the formulas in $\mathcal{K}$ is computed by averaging all terms using the geometric mean, but alternative approaches are possible [9]. The objective function $\theta = \text{Sat}_A(\mathcal{G}_\theta(\mathcal{K}))$ is therefore the satisfiability of all formulas in $\mathcal{K}$, which is maximized by training the model parameters. In its simplest form of binary classification without constraints, $\mathcal{K}$ will consist of one term for positive examples in the dataset and one term for negative examples. In summary, the core extension applied to a regular neural network optimization enabled by LTN is that of querying with many-valued first-order logic and learning with knowledge base constraints.

Continuous querying for model understanding: LTN inference using first-order logic clauses is not only a post hoc explanation in the traditional sense. It allows that inference forms an integral part of an iterative process, allowing for incremental explanation through the distillation of knowledge guided by data. We achieve this by computing the value of a grounding $\mathcal{G}_\theta(\phi_q)$, given a trained network (set of parameters θ), for a user-defined query ϕ_q.

Specifically, we save and reinstate the learned parameters stored in the LTN implementation. This is done by storing the parameters θ resulting from each learning cycle. A query is any logical formula expressed in first-order logic. Queries are evaluated by calculating the grounding $\mathcal{G}$ of any formula whose predicates are already grounded in the multilayer perceptron or even by defining a predicate in terms of existing predicates. For example, the logical formula $\forall x : (A(x) \rightarrow B(x))$ can be evaluated by applying the values of x, obtained from the dataset, to the trained Neural Network, obtaining the values of output neurons A and B in (0,1) (corresponding to the truth-values of predicates A and B, respectively); and calculating the implication with the use of the Reichenbach-norm and aggregating for all x using the p-mean, as exemplified below.

A query can explain AI systems by connecting different model outputs, aggregating inputs for summarizing the behavior of a system in specific domains or relating specific inputs with specific features against each other and the output. In Reference [99], it is demonstrated that this framework can be extended to include intermediate representations, thus providing an understanding of how concepts are logically associated with particular output classes.

Logical formulas used for such explanations follow semantics for logical connectives that are defined according to fuzzy logic semantics: Conjunctions are approximated by t-norms (e.g., $min(a, b)$), disjunctions by t-conorms (e.g., $max(a, b)$), negation by fuzzy negation (e.g., $1 - a$), and implication by fuzzy implications (e.g., $max(1 - a, b)$). The universal quantifier is defined as the generalized mean, also referred to as p-mean.

Algorithm 2 illustrates the steps we take to continuously refine $\mathcal{K}_{new}$ with a human-in-the-loop. The queries are derived from questions that a user might have about the model's response: *How does the model behave for a specific group? How does the model behave for particular edge cases?* These questions can be translated into FOL-queries. Simultaneously, an XAI method further informs the user about possible undesired model behavior, which may not be as apparent as the above common questions. In Figure 13, XAI method SHAP reports a disparity in how the variable *immigrant status* is used by the model for black and white inpatients when predicting the risk of violence in a psychiatric hospitalization setting [89].

Algorithm 2: LTN-active learning cycle

Input: Dataset, Knowledge (in the form of FOL)
Output: Model satisfiability measured as overall sat-level

```
for each predicate P in K do
    Initialize G_θ(P)        // each P can be a multilayer perceptron or output neuron
for epoch < num-epochs do
    max sat G_θ(K)           // optimize θ to achieve max satisfiability of K
while Revision do                                   // user-defined Boolean
    for each FOL-query φ_q do
        Calculate G(φ_q)     // query the network to obtain the truth-value of φ_q
        if G(φ_q) < t        // t in [0,1] is a user-defined minimum sat value
         then
            Add φ_q to K_new
    Apply XAI-method                                // we use Shapley values
    for each predicate P do
        Inquire G_θ(P)                  // query predicate-specific groundings
        if G_θ has undesired property f(G_θ) then // user-determined desiderata
            Revise f(G) to K_new                    // method-dependent revision
    if K_new ≠ ∅ then
        K ← K ∪ K_new
        θ* = arg max_{θ∈Θ} G_θ(K)                   // re-train the network
```

The XAI technique SHAP is used together with LTN queries to highlight such findings and subsequently address them by adding knowledge to $\mathcal{K}_{new}$ and retraining, as will be illustrated in the next section and the accompanying notebook. This cycle can be repeated until the revision process delivers satisfactory results to the user with respect to model performance and behavior.

6.4 Case Study on Violence Reiteration Prediction in a Psychiatric Hospitalization Setting with a Deep Neural Network within the LTN Framework for Explainable Revision

We demonstrate the method mentioned above using a well-known fairness-related *COMPAS* dataset from ProPublica.[14] For additional examples using alternative datasets, as well as comparisons with alternative methods, we refer the reader to Reference [101]. We use a case study of violence risk prediction of inpatients in a psychiatric hospitalization, i.e., to predict whether inpatients tend to reiterate violent behavior or not during their stay [89].[15] Typical features of this case study encode sociodemographic information such as sex, race, immigration status, and so on. ML algorithms tend to generate false positives in presence of a protected group versus an unprotected group (black vs. white, immigrants vs. non-immigrants). In this case, a bias on such features may compromise fairness. The presented framework (based on XAI (SHAP)+LTN) shows how such *MLOps* pipeline can detect it and tackle it.

A trained network is queried to return the truth value associated to the predicate used for the classification task $\mathcal{G}(D(\mathcal{T}))$ for the entire training set $\mathcal{T}$. This will allow us to answer how the model treats similar individuals across protected and unprotected groups. Using quantile-based discretization, we obtain answers to the question: *How does prediction for equally sized groups for each protected and unprotected variable differ across different risk categories for violence?* We de-

[14]The demonstration of the method, the XAI + LTN pipeline (on which XAI techniques such as SHAP can be embedded for interactive explanations), and the data are accessible at https://github.com/benediktwagner/nesyxai/blob/master/experiments/fairness/1_LTN_fairness_tutorial.ipynb. The original LTN repository adapted for this method is: https://github.com/logictensornetworks/logictensornetworks

[15]https://informatics.bmj.com/content/bmjhci/29/1/e100459/DC1/embed/inline-supplementary-material-1.pdf?download=true

termine whether the model achieves parity between black and white prisoners in the same risk category on aggregate. Querying such axioms reveals a low level of satisfiability ($sat_{\phi_i} \approx 0.5$), suggesting that the model is learning undesirable disparities in groups with medium risk of violence. Thus, the model predicts bad behavior more often for the protected group than for the unprotected group of the same risk category. In groups where the risk of violence is very low or very high, there are no significant differences in violence prediction of the model for the protected and unprotected group, which is in line with the desired notion of *group fairness* [32].

We confirm the disparity between groups by calculating their Shapley values. Since the SHAP method uses the same units as the original model output, we can decompose the model output using SHAP and compute each feature's parity difference among protected and unprotected groups using their respective Shapley value.

We can subsequently revise $\mathcal{K}$ using the queries ϕ_i of the different groups as soft constraints and are able to revise the network to decrease the undesired disparities while retaining high accuracy as measured in Reference [101] and the notebook.[14] In this demonstration, only model outputs combined with protected attributes are used to inform the queries, as the focus is primarily querying the output concerning unfair treatment. A further query could answer how predictions differ across groups of a specific age in combination with protected attributes.

Any combination of features or even intermediate representations—such as feature activations in a CNN—as well as a combination of models are available and can be queried using the LTN framework through fuzzy logic. Furthermore, the latest iteration of the LTN framework allows for dynamic masking, which means that the explanation iterations and revision could be further automated within the LTN framework using custom masks. Such custom masks, for example, remove the necessity of manual discretization into parity groups by performing automatic grouping based on dynamically changing output logits.

In our example, however, the user can vary the number of user-defined queries and discretizations groups into different granularities. It is worth emphasizing the flexibility of such approach w.r.t. further queries and its potential use with alternative fairness constraint constructions. With the increasing complexity of models as well as fairness definitions, rich languages such as fuzzy FOL can be beneficial to adapt to regulatory and societal changes to notions of fairness. One example would be a simple adaptation of the value p in the aggregation using the p-mean. Using larger values for p, the fairness notion converges from *group fairness* towards *individual fairness*, as the generalized mean, converging from a simple average towards the *min* value, gradually (with higher relative importance for lower values).

Integrating XAI methods with neural-symbolic approaches allows us to learn about the undesired behavior of a model and intervene to address discrepancies. Effecting such intervention is ultimately the goal of the field of *AI alignment*, to which the techniques of interactive explainability have a major contribution to make, as advocated in Reference [100]. We have demonstrated an interactive model-agnostic method and an algorithm for fairness in healthcare and have shown how one can remove demographic disparities from trained neural networks by using a continual learning LTN-based framework.

7 Rendering XAI Explanations through a Template System for Natural Language Explanations (TS4NLE)

Methods presented during the previous sections provide good explanations. However, the way in which an explanation is presented is sometimes as important as the content of the explanation, because an explanation is only as good as its audience's understanding of it. Therefore, we present a method to render XAI Explanations through Natural Language Generation. This method is aimed at anyone who wants to display their explanation in the best possible way so it is perfectly

understandable by their user, whether they are a developer or an end-user with no knowledge of deep learning.

All methods discussed above provide outputs in a structured format that can be represented in a graph-like way. Such a format enables the design of different strategies for transforming the provided outputs into a representation that can be easily understood and consumed by the target user.

Explanations generated starting from structured formats such as the one mentioned above help users in better understanding the output of an AI system. A better understanding of this output allows users to increase the overall acceptability in the system. An explanation should not only be correct (i.e., mirroring the conceptual meaning of the output to explain), but also useful. An explanation is useful or actionable if and only if it is meaningful for the users targeted by the explanation and provides the rationale behind the output of the AI system [30]. For example, if an explanation has to be provided on a specific device, then such a device represents a constraint to be taken into account for deciding which is the most effective way for generating the explanation. Such explanation can be in natural language/vocal messages, visual diagrams, or even haptic feedback.

In this section, we focus on the generation of **Natural Language Explanations (NLE)**. Producing these carries a challenge, given the requirement of adopting proper language with respect to the targeted audience [8] and their context. Briefly, let us consider a sample scenario occurring within the healthcare domain, where patients suffering from diabetes are monitored by a virtual coaching system in charge of providing recommendations about healthy behaviors (i.e., diet and physical activities) based on what patients ate and which activities they did. The virtual coaching system interacts with both clinicians and patients, and when an undesired behavior is detected, it has to generate two different explanations: one for the clinician containing medical information linked with the detected undesired behavior—including also possible severe adverse consequences; and one for the patient, omitting some medical details and, possibly, including persuasive text inviting to correct the patient's behavior in the future. The end-to-end explanation generation process, from model output to an object usable by the target users, requires a building block in the middle supporting the rendering activity. Such rendering requires explanations having a formal representation with a logical language equipped with predicates for entities and relations. This formal representation can be directly represented as an *explanation graph* with entities/nodes and relations/arcs. It allows: (i) its own enhancement with other concepts from domain ontologies or Semantic Web resources and (ii) an easy rendering in many human-comprehensible formats. Such an explanation graph can be easily obtained from the XAI techniques explained above. For example, the explanatory features and the output class provided by SHAP can be regarded as the nodes of the explanation graph, whereas arcs are computed on the basis of the SHAP features values. SHAP's output is one of the possible inputs that the TS4NLE strategy can process. Indeed, TS4NLE is agnostic with respect to the type of model adopted by the ML system, since it can work with any approach providing an output that can be represented with a graph-like format. The *explanation graph* can also work as bridge for accessing different types of knowledge usable, for example, to enrich the content of natural language explanations by respecting privacy and ethical aspects connected with the knowledge to use.

Explanations require a starting formal (graph-like) representation to be easily rendered and personalized through natural language text [29]. The generation of such natural language explanations can rely on pipelines that take the structured explanation content as input and, through several steps, perform its linguistic realization The work in Reference [29] injects in such a pipeline a template system that implements the text structuring phase of the pipeline. Figure 14 shows the explanation generation process starting from a SHAP analysis of a model output.

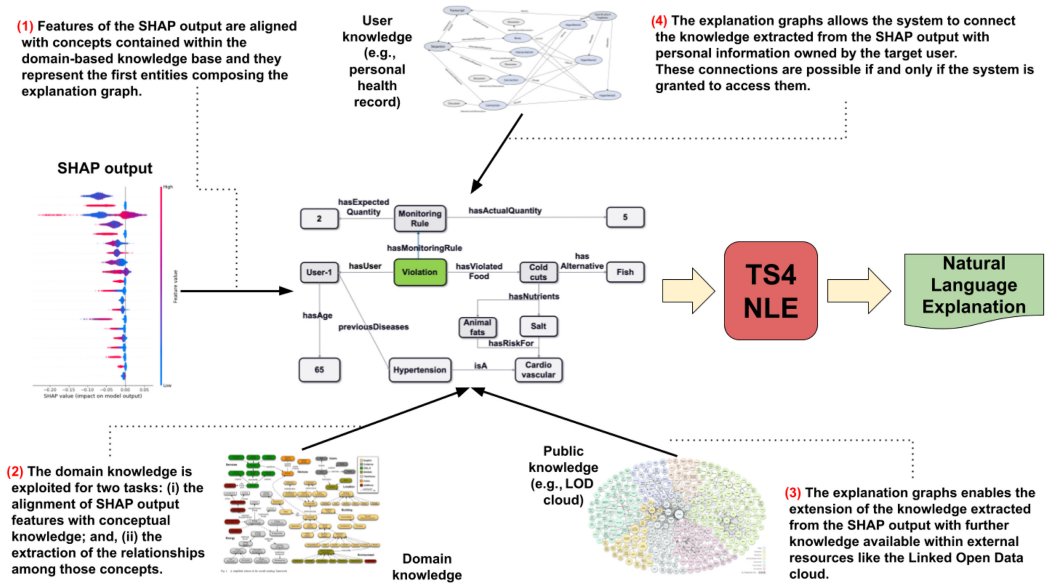

Fig. 14. The images show the process of transforming a SHAP output into an explanation graph that is then transformed into its equivalent natural language explanation. Features contained within the SHAP output are transformed into concepts linked with a knowledge base related to the problem's domain. Such a knowledge base is exploited also for extracting relationships between the detected concepts. This preliminary explanation graph can be enriched with further knowledge extracted from publicly available resources (e.g., the Linked Open Data cloud) as well as with private data (e.g., personal health records). Finally, the explanation graph, through the NLE rendering component, is transformed into a natural language explanation.

As mentioned above, generating natural language explanations starts from the creation of the explanation graph, since it provides a complete structured representation of the knowledge that has to be transferred to the target user. As first step, the features of the SHAP output are transformed into concepts of the explanation graph and they are, possibly, aligned with entities contained within the knowledge base related to the problem's domain. Such entities represent the first elements composing the explanation graph that can be used as collector for further knowledge exploited for creating the complete message. Besides the alignment of SHAP output features with the domain knowledge, such a knowledge base is exploited for extracting the relationships among the identified concepts. The extraction of such relationships is fundamental for completing the explanation graph as well as for supporting its transformation into its equivalent natural language representation. Once the alignment between the SHAP output and the domain knowledge has been completed, the preliminary explanation graph can be extended in two ways. First, public available knowledge can be linked to the preliminary explanation graph for completing the domain knowledge. Let us consider as example the explanation graph shown in Figure 15. Some medical information associated with the identified food category may not be contained in the domain knowledge integrated into the local system. Hence, by starting from the concept representing the food category, we may access, through the Linked Open Data cloud, the UMLS[16] knowledge base for extracting information about the nutritional disease risks connected with such a food category. Besides public knowledge, the explanation graph can be enriched with user

[16]Unified Medical Language System (UMLS): https://www.nlm.nih.gov/research/umls/index.html

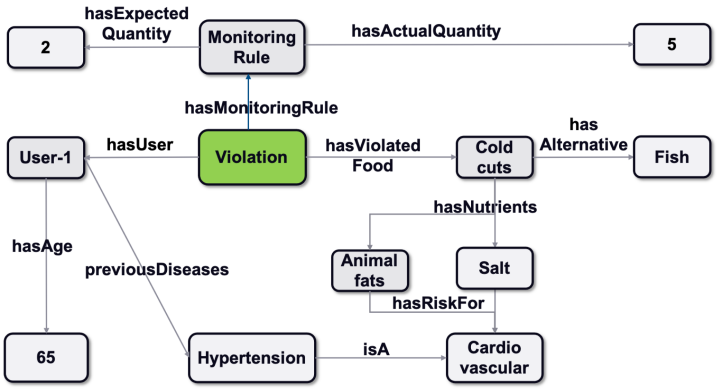

Fig. 15. Explanation graph for users exceeding in cold cuts consumption in the diet & healthy lifestyle adherence application.

information provided if and only if they are compliant with respect to possible privacy constraints. User information can be provided by knowledge bases as well as probabilistic models. Also, in this case, TS4NLE is agnostic with respect to the external source to exploit. In the use case we present below, TS4NLE relies on an external user-oriented knowledge base containing facts that TS4NLE can reason on for deciding which kind of linguistic strategy to adopt. Let us consider the healthcare domain use case. Here, information contained in the users' personal health record can be used for enriching the explanation graph with concepts by linking, for example, the negative effects of the over-consumption of a specific food category by users with potential nutritional diseases.

Finally, the created explanation graph can be rendered in a natural language form through a template system for natural language explanations (TS4NLE) [29] that leverages a **Natural Language Generation (NLG)** pipeline. Templates are formal grammars whose terminal symbols are a mixture of terms/data taken from the nodes/arcs of the explanation graph and from a domain knowledge base. Terms in the explanation graph encode the rationale behind the AI system decision, whereas the domain knowledge base encodes further terms that help the user's comprehension by: (i) enhancing the final rendered explanation with further information about the output and (ii) using terms or arguments that are tailored to that particular user and increase the comprehension of the explanation. Generally, this user model is previously given, in the form of an ontology or knowledge graph.

TS4NLE is structured as a decision tree where the first level contains high-level and generic templates that are progressively specialized and enriched according to the user's feature specified in the user model. Once templates are filled with non-terminal terms, the lexicalization[17] and linguistic realization of the pipeline are performed with standard natural language processing engines such as RosaeNLG.[18]

7.1 TS4NLE Use Case: Persuasive Message Generation for Healthy Lifestyle Adherence

In this subsection, we provide the description of a complete use case related to the generation of persuasive natural language explanation within a concrete case within the healthcare domain.

[17] *Lexicalization* is the process of choosing the right words (nouns, verbs, adjectives, and adverbs) that are required to express the information in the generated text; it is extremely important in NLG systems that produce texts in multiple languages. Thus, the template system chooses the right words for an explanation, making it tailored.
[18] https://rosaenlg.org/rosaenlg/3.0.0/index.html

Given as input a user lifestyle (obtained with a diet diary or a physical activity tracker), AI systems are able to classify the user behavior in classes ranging from *very good* to *very bad*. The explanation graph contains the reason for such a prediction and suggestions for reinforcing or changing the particular lifestyle. According to the user model (e.g., whether the user has to be encouraged or not, the users' barriers or capacities), the template system is explored to reach a leaf containing the right terms to fill the initial non-terminal symbols of the template. A user study regarding the Mediterranean diet states that such tailored explanations are more effective at changing users' lifestyles with respect to a standard notification of a bad lifestyle. A further guide of this use case is available online.[19]

The explanation graph contains entities connected by relations encoding the rationale of the AI system decision. Figure 15 contains the explanation graph for a 65-year-old user that consumes too much cold cuts. Such a graph is rendered with TS4NLE as: *"This week you consumed too much (5 portions of a maximum 2) cold cuts. Cold cuts contain animal fats and salt that can cause cardiovascular diseases. People over 60 years old are particularly at risk. Next time try with some fresh fish."*

The generation of the natural language explanation shown above is performed by TS4NLE by following the steps below. After the generation of the explanation graph, the *message composition* component of TS4NLE starts the generation of three textual messages for the feedback, the argument, and the suggestion, respectively. This is inspired by the work in Reference [64] and expanded, taking into consideration additional strategies presented in Reference [38]. These consist of several persuasion strategies that can be combined together to form a complex message. Each strategy is rendered through natural language text with a template. A template is formalized as a grammar whose terminal symbols are filled according to the data in the violation package and new information queried in the reference ontology. Once templates are filled, a sentence realizer (i.e., a producer of sentences from syntax or logical forms) generates natural language sentences that respect the grammatical rules of a desired language

Below, we describe the implemented strategies to automate the message generation, focusing also on linguistic choices.

Explanation Feedback: is the part of the message that informs the user about the not-compliant behavior, hereafter called "violation," with the goal that has been set up. Feedback is generated considering data included in the explanation graph starting from the violation object: The food entity of the violation will represent the object of the feedback, whereas the level of violation (e.g., deviation between food quantity expected and that actually taken by the user) is used to represent the severity of the incorrect behavior. The intention of the violation represents the fact that the user has consumed too much or not enough amount of a food entity. Feedback contains also information about the kind of meal (breakfast, lunch, dinner, or snack) to inform the user about the time span in which the violation was committed. From a linguistic point of view, choices in the feedback type are related to the verb and its tense: E.g., beverages imply use of the verb *to drink*, while for solid food, we use *to eat*. To increase the variety of the message, verbs *to consume* and *to intake* are also used. Past simple tense is used when violation is related to a specific moment (e.g., *You drank a lot of fruit juice for lunch*), while present continuous is used when the violation is related to a period of time of more days and the period is not yet ended (e.g., *You are drinking a lot of fruit juice this week*).

Explanation Argument: it is the part of the message informing users about possible consequences of a behavior. For example, in the case of diet recommendations, the *Argument* consists of two parts: (i) information about nutrients contained in the food intake that caused the violation

[19] https://horus-ai.fbk.eu/semex4all/

and (ii) information about consequences that nutrients have on human body and health. Consequences imply the positive or negative aspects of nutrients. In this case, TS4NLE uses the intention element contained in the selected violation package to identify the type of argument to generate. Let us consider the violation of our running example, where the monitoring rule limits the daily fruit juice drinking to less than 200 ml (a water glass), since it contains too much sugar. In the presence of an excess in juice consumption (translating to a discouraging intention) the argument is constituted by a statement with the negative consequences of this behavior on user health. On the contrary, the violation of a rule requiring the consumption of at least 200 gr of vegetables per day brings the system to generate an argument explaining the many advantages of getting nutrients contained in that food (an encouraging intention). In both cases, this information is stored within the explanation graph. Moreover, TS4NLE analyzes the message history to decide which property of the explanation graph to use in the *Argument* to generate a message content that depends on e.g., content sent in the past few days, ensuring a certain degree of variability. With respect to linguistic choices, the type of nutrients and their consequences influence the verb usage in the text. Finally, to emphasize different aspects of the detected violation, templates encode the use of appropriate parts of speech. For example, for stressing the negative aspects of the violated food constraint, the verb *contain* (nutrients) and *can cause* (for consequences) were used. However, positive aspects are highlighted by the verb phrase *is rich in* and verb *help* are used for nutrients and consequences, respectively.

Explanation Suggestion: This part represents an alternative behavior that TS4NLE delivers to the user to motivate him/her to change his/her lifestyle. Exploiting the information available within the explanation graph, and possibly collected from both public and private knowledge, TS4NLE generates a *post* suggestion to inform the user about the healthy behavior that he/she can adopt as alternative. To do that, the data contained in the explanation graph are not sufficient. TS4NLE performs additional meta-reasoning to identify the appropriate content that depends on (i) qualitative properties of the entities involved in the event; (ii) user profile; (iii) other specific violations; (iv) history of messages sent. Continuing with the running example, first TS4NLE queries the domain knowledge base through the reasoner to provide a list of alternative foods that are valid alternatives to the violated behavior (e.g., similar-taste relation, list of nutrients, consequences on user health). These alternatives are queried according to some constraints: (i) compliance with the user profile and (ii) compliance with other set-up goals. Regarding the first constraint, the reasoner will not return alternative foods that are not appropriate for the specific profile. Let us consider a vegetarian profile: The system does not suggest vegetarian users to consume fish as an alternative to meat, even if fish is an alternative to meat by considering only the nutrients. The second constraint is needed to avoid alternatives that could generate a contradiction with other healthy behavior rules. For example, the system will not propose cheese as alternative to meat if the user has the persuasion goal of cheese reduction. Finally, a control on message history is executed to avoid the suggestion of alternatives recently proposed. Regarding the linguistic aspect, the system uses appropriate verbs, such as *try* or *alternate*, to emphasize the alternative behavior. Both tools[20] and the colaboratory (Colab notebook) session are online[21] for freely creating new use cases using the TS4NLE approach.

7.2 TS4NLE Suitability Analysis: Pros and Cons

The use of explanation graphs is an intuitive and effective way for transforming meaningless model outputs into a comprehensive artifact that can be leveraged by targeted users. Explanation graphs

[20] https://github.com/ivanDonadello/TS4NLE

[21] https://colab.research.google.com/drive/1iCVSt7TFMruSzeg5DswLOzOR1n7xATbz

convey formal semantics that: (i) can be enriched with other knowledge sources publicly available on the web (e.g., Linked Open Data cloud) or privacy-protected (e.g., user profiles); (ii) allow rendering in different formats (e.g., natural language text or audio); and (iii) allow full control over the rendered explanations (i.e., the content of the explanations). Natural language rendering with a template-system allows full control on the explanations at the price of high effort in domain and user modeling by domain experts. This aspect can be considered the major bottleneck of the TS4NLE approach. Such bottleneck can be mitigated by using machine learning with human-in-the-loop towards interactive interpretable machine learning [43] to increase variability in the generated natural language explanations.

8 Limitations

In the landscape of XAI, one central limitation stems from the sheer diversity and abundance of explainability methods available. Our article provides insights into several widely used XAI techniques; however, it is crucial to acknowledge that the field is evolving rapidly, with new methods constantly emerging. The sheer number and diversity of these methods make it challenging to be exhaustive in our coverage. New techniques may offer unique advantages or address specific limitations, and thus, the landscape of XAI is ever-expanding.

Furthermore, the fast-paced evolution of the XAI domain introduces a dynamic aspect to the limitations. Some methods discussed in our guide may already be succeeded by more advanced approaches, or they might have given rise to tools and implementations that are simpler and more efficient. The field's progress may render certain explanations or techniques obsolete, emphasizing the need for ongoing exploration and adaptation. As a result, the shelf life of specific XAI methods may be limited, and developers should stay abreast of the latest advancements to ensure the relevancy and efficacy of their chosen explainability tools.

In essence, the limitations inherent in our guide stem not only from the impossibility of being exhaustive in the face of the myriad XAI methods but also from the dynamic nature of the field, where advancements can quickly outpace existing methodologies. Navigating this evolving landscape requires a commitment to staying informed about emerging techniques and understanding that the most effective XAI strategies may shift over time.

Another noteworthy limitation of the tutorials presented in our guide lies in the nature of the datasets used for demonstration purposes. While the provided tutorials offer a foundational understanding of XAI techniques, it is essential to recognize that they are primarily illustrated on simple datasets. Real-world applications often involve more complex and dynamic datasets, introducing additional challenges and intricacies not fully captured in these tutorial scenarios.

In particular, some real-world contexts demand the incorporation of real-time data, where the need for instantaneous decision-making is paramount. The tutorials, focused on elucidating XAI methods, may not fully encapsulate the complexities introduced by the dynamic and evolving nature of data streams in real-time applications. In scenarios such as these, where the temporal dimension is critical, the applicability and effectiveness of certain XAI techniques may require further exploration and adaptation.

Thus, while the tutorials serve as a valuable starting point, developers and practitioners should be cognizant of the potential disparities between the controlled environments of tutorial datasets and the nuanced challenges posed by real-world applications. Addressing the intricacies of complex, real-time data scenarios will necessitate additional considerations and potentially the exploration of more advanced XAI methods tailored to such dynamic contexts. This acknowledgement underscores the importance of further research and experimentation when applying XAI techniques in practical, real-world settings with more intricate data dynamics.

9 Conclusion

In conclusion, our exploration of **Explainable AI (XAI)** techniques has equipped developers, domain experts, and decision-makers with valuable tools to unravel the intricacies of neural-symbolic and machine learning models using tabular data, images, language, and graphs. By delving into the implementation details, we aimed to demystify the black-box nature of these models, fostering a didactic environment for users seeking clarity.

The explanations provided cater to a diverse audience, emphasizing accessibility and open sharing via the availability of the explanations and accompanying resources on our GitHub repository,[22] inviting continuous learning and collaboration.

Looking ahead, future endeavors should focus on expanding the repertoire of XAI methods, especially in handling challenges posed by incomplete and multi-modal data[41]. As the field evolves quickly [60], addressing these complexities becomes paramount for ensuring the applicability of XAI techniques across various domains. By staying at the forefront of innovation, we can collectively contribute to the responsible and effective governance of AI systems [6].

In essence, this guide serves as a resource, empowering users to navigate the nuanced landscape of XAI. As we strive for greater understanding and transparency in artificial intelligence, this work lays the groundwork for ongoing discussions, developments, and improvements in the realm of explainability.

Beyond the specific case studies outlined in our guide, XAI finds relevance in a diverse array of real-world applications, offering transparency and interpretability in decision-making processes. One notable sector is healthcare and medical AI [40], where the need for accurate and interpretable models is paramount. XAI can aid in understanding and justifying predictions made by complex medical models, ensuring healthcare professionals can trust and comprehend the reasoning behind critical diagnoses or treatment recommendations. For instance, XAI could be applied to interpret semantic features or object parts [12, 27], contributing to a medical image classification, providing insights for better-informed and less biased clinical decisions [25]. Other application of XAI is using linguistic summaries to aid translating the model explanation to domain experts in psychiatry [44]. Another domain where XAI can have a relevant impact is in climate change and earth sciences [55].

In the financial domain, XAI proves invaluable in risk assessment and fraud detection. Interpretable models can shed light on the factors influencing credit scoring [33], enabling financial institutions to explain lending decisions to customers and regulatory bodies. Moreover, in the context of fraud detection, XAI can elucidate the rationale behind flagged transactions, enhancing the transparency of automated fraud prevention systems.

In the field of autonomous vehicles, where safety is paramount, XAI can play a crucial role. Providing explanations for decisions made by self-driving cars ensures not only regulatory compliance [35] but also fosters public trust. For instance, an XAI system could explain why a vehicle made a specific maneuver or identification during an unforeseen event, contributing to safer and more accountable autonomous transportation systems.

In the realm of customer service and chatbots, XAI facilitates a more transparent and understandable interaction. Users can receive clear explanations for recommendations or decisions made by virtual assistants, enhancing the user experience and trust in automated systems. This is particularly relevant in industries such as e-commerce or online services where personalized recommendations play a significant role. One example can be insurance advisors [69] or any recommender system, inclusive and accessible applications [28, 70].

These examples highlight the wide-ranging applications of XAI beyond the presented case studies, emphasizing its importance in domains where transparency, accountability, and user trust are

[22] https://github.com/NataliaDiaz/XAI-tutorial

critical components of decision-making processes. As technology continues to advance, the need for interpretable AI will likely expand into even more diverse and complex real-world scenarios.

Acknowledgements

The BBVA Foundation takes no responsibility for the opinions, statements and contents of this project, which are entirely the responsibility of its authors. This publication reflects only the authors' view, and the European Commission is not responsible for any use that may be made of the information it contains.

References

[1] Saranya A. and Subhashini R. 2023. A systematic review of Explainable Artificial Intelligence models and applications: Recent developments and future trends. *Decis. Analyt. J.* 7 (2023), 100230. DOI: https://doi.org/10.1016/j.dajour.2023.100230

[2] Zeina Abu-Aisheh, Romain Raveaux, Jean-Yves Ramel, and Patrick Martineau. 2015. An exact graph edit distance algorithm for solving pattern recognition problems. In *Proceedings of the 4th International Conference on Pattern Recognition Applications and Methods*.

[3] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. 2018. Sanity checks for saliency maps. In *Proceedings of the Conference on Advances in Neural Information Processing Systems*. 9505–9515.

[4] Charu C. Aggarwal. 2018. Neural networks and deep learning. *Springer* 10, 978 (2018), 3.

[5] Sajid Ali, Tamer Abuhmed, Shaker El-Sappagh, Khan Muhammad, Jose M. Alonso-Moral, Roberto Confalonieri, Riccardo Guidotti, Javier Del Ser, Natalia Díaz-Rodríguez, and Francisco Herrera. 2023. Explainable Artificial Intelligence (XAI): What we know and what is left to attain trustworthy artificial intelligence. *Inf. Fusion* 99 (2023), 101805.

[6] Plamen P. Angelov, Eduardo Almeida Soares, Richard Jiang, Nicholas I. Arnold, and Peter M. Atkinson. 2021. Explainable artificial intelligence: An analytical review. *Wiley Interdiscip. Rev.: Data Min. Knowl. Discov.* 11 (2021). Retrieved from https://api.semanticscholar.org/CorpusID:236501382

[7] Leila Arras, Grégoire Montavon, Klaus-Robert Müller, and Wojciech Samek. 2017. Explaining recurrent neural network predictions in sentiment analysis. In *EMNLP'17 Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis (WASSA'17)*.

[8] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador Garcia, Sergio Gil-Lopez, Daniel Molina, Richard Benjamins, Raja Chatila, and Francisco Herrera. 2020. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Inf. Fusion* 58 (2020), 82–115. DOI: https://doi.org/10.1016/j.inffus.2019.12.012

[9] Samy Badreddine, Artur d'Avila Garcez, Luciano Serafini, and Michael Spranger. 2020. Logic tensor networks. *arXiv preprint arXiv:2012.13635* (2020).

[10] David Baehrens, Timon Schroeter, Stefan Harmeling, Motoaki Kawanabe, Katja Hansen, and Klaus-Robert Müller. 2010. How to explain individual classification decisions. *J. Mach. Learn. Res.* 11 (2010), 1803–1831.

[11] Jacqueline Michelle Metsch, Anna Saranti, Alessa Angerschmid, Bastian Pfeifer, Vanessa Klemt, Andreas Holzinger, and Anne-Christin Hauschild. 2024. CLARUS: An interactive explainable AI platform for manual counterfactuals in graph neural networks. *Journal of Biomedical Informatics* 150 (2024), 104600. https://doi.org/10.1016/j.jbi.2024.104600

[12] Adrien Bennetot, Gianni Franchi, Javier Del Ser, Raja Chatila, and Natalia Diaz-Rodriguez. 2022. Greybox XAI: A neural-symbolic learning framework to produce interpretable predictions for image classification. *Knowl.-based Syst.* 258 (2022), 109947.

[13] Olivier Bodenreider. 2004. The unified medical language system (UMLS): Integrating biomedical terminology. *Nucleic Acids Res.* 32, suppl_1 (2004), D267–D270.

[14] Stephen Boyd and Lieven Vandenberghe. 2004. *Convex Optimization*. Cambridge University Press. Retrieved from http://www.amazon.com/exec/obidos/redirect?tag=citeulike-20&path=ASIN/0521833787

[15] Leo Breiman. 2001. Random forests. *Mach. Learn.* 45, 1 (2001), 5–32. DOI: https://doi.org/10.1023/A:1010933404324

[16] Niklas Bussmann, Paolo Giudici, Dimitri Marinelli, and Jochen Papenbrock. 2020. Explainable AI in fintech risk management. *Front. Artif. Intell.* 3 (2020). DOI: https://doi.org/10.3389/frai.2020.00026

[17] Oana-Maria Camburu and Z. Akata. 2021. Natural-XAI: Explainable AI with natural language explanations. In *Proceedings of the International Conference on Machine Learning (ICML'21)*.

[18] Diogo Carvalho, Eduardo Pereira, and Jaime Cardoso. 2019. Machine learning interpretability: A survey on methods and metrics. *Electronics* 8 (07 2019), 832. DOI: https://doi.org/10.3390/electronics8080832

[19] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. 2002. SMOTE: Synthetic minority over-sampling technique. *J. Artif. Intell. Res.* 16 (June 2002), 321–357. DOI: https://doi.org/10.1613/jair.953

[20] Tianqi Chen and Carlos Guestrin. 2016. XGBoost. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. DOI: https://doi.org/10.1145/2939672.2939785

[21] Minh Dang and Tan N. Nguyen. 2023. Digital face manipulation creation and detection: A systematic review. *Electronics* 12, 16 (2023). DOI: https://doi.org/10.3390/electronics12163407

[22] Paul B de Laat. 2021. Companies committed to responsible AI: From principles towards implementation and regulation? *Philos. Technol.* 34, 4 (2021), 1135–1193.

[23] Javier Del Ser, Alejandro Barredo-Arrieta, Natalia Díaz-Rodríguez, Francisco Herrera, Anna Saranti, and Andreas Holzinger. 2024. On generating trustworthy counterfactual explanations. *Inf. Sci.* 655 (2024), 119898.

[24] J. Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT'19)*.

[25] Natalia Díaz-Rodríguez, Rūta Binkytė, Wafae Bakkali, Sannidhi Bookseller, Paola Tubaro, Andrius Bacevičius, Sami Zhioua, and Raja Chatila. 2023. Gender and sex bias in COVID-19 epidemiological data through the lens of causality. *Inf. Process. Manag.* 60, 3 (2023), 103276.

[26] Natalia Díaz-Rodríguez, Javier Del Ser, Mark Coeckelbergh, Marcos López de Prado, Enrique Herrera-Viedma, and Francisco Herrera. 2023. Connecting the dots in trustworthy artificial intelligence: From AI principles, ethics, and key requirements to responsible AI systems and regulation. *Inf. Fusion* 99 (2023), 101896.

[27] Natalia Díaz-Rodríguez, Alberto Lamas, Jules Sanchez, Gianni Franchi, Ivan Donadello, Siham Tabik, David Filliat, Policarpo Cruz, Rosana Montes, and Francisco Herrera. 2022. EXplainable Neural-Symbolic Learning (X-NeSyL) methodology to fuse deep learning representations with expert knowledge graphs: The MonuMAI cultural heritage use case. *Inf. Fusion* 79 (2022), 58–83.

[28] Natalia Díaz-Rodríguez and Galena Pisoni. 2020. Accessible cultural heritage through explainable artificial intelligence. In *Proceedings of the 28th ACM Conference on User Modeling, Adaptation and Personalization*. 317–324.

[29] Ivan Donadello, Mauro Dragoni, and Claudio Eccher. 2019. Persuasive explanation of reasoning inferences on dietary data. In *PROFILES/SEMEX@ISWC (CEUR Workshop Proceedings,* Vol. 2465*)*. CEUR-WS.org, 46–61.

[30] Derek Doran, Sarah Schulz, and Tarek R. Besold. 2017. What does explainable AI really mean? A new conceptualization of perspectives. In *CEx@AI*IA (CEUR Workshop Proceedings,* Vol. 2071*)*. CEUR-WS.org, 1–8.

[31] Emmanuel Doumard, Julien Aligon, Elodie Escriva, Jean-Baptiste Excoffier, Paul Monsarrat, and Chantal Soulé-Dupuy. 2023. A quantitative approach for the comparison of additive local explanation methods. *Inf. Syst.* 114 (2023), 102162.

[32] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. 2012. Fairness through awareness. In *Proceedings of the Innovations in Theoretical Computer Science Conference (ITCS'12)*. DOI: https://doi.org/10.1145/2090236.2090255

[33] Ayoub El-Qadi, Maria Trocan, Patricia Conde-Cespedes, Thomas Frossard, and Natalia Díaz-Rodríguez. 2023. Credit risk scoring using a data fusion approach. In *Proceedings of the International Conference on Computational Collective Intelligence*. Springer, 769–781.

[34] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'96)*. 226–231.

[35] David Fernandez-Llorca and Emilia Gómez. 2023. Trustworthy artificial intelligence requirements in the autonomous driving domain. *Computer* 56, 2 (2023), 29–39.

[36] Enrique Garcia-Ceja, Michael Riegler, Petter Jakobsen, Jim Tørresen, Tine Nordgreen, Ketil J. Oedegaard, and Ole Bernt Fasmer. 2018. Depresjon: A motor activity database of depression episodes in unipolar and bipolar patients. In *Proceedings of the 9th ACM on Multimedia Systems Conference (MMSys'18)*. ACM, New York, NY. DOI: https://doi.org/10.1145/3204949.3208125

[37] Robert L. Grossman, Allison P. Heath, Vincent Ferretti, Harold E. Varmus, Douglas R. Lowy, Warren A. Kibbe, and Louis M. Staudt. 2016. Toward a shared vision for cancer genomic data. *New Eng. J. Med.* 375, 12 (2016), 1109–1112.

[38] M. Guerini, O. Stock, and M. Zancanaro. 2007. A taxonomy of strategies for multimodal persuasive message generation. *Appl. Artif. Intell. J.* 21, 2 (2007), 99–136.

[39] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 770–778.

[40] Andreas Holzinger, Matthias Dehmer, Frank Emmert-Streib, Rita Cucchiara, Isabelle Augenstein, Javier Del Ser, Wojciech Samek, Igor Jurisica, and Natalia Díaz-Rodríguez. 2022. Information fusion as an integrative cross-cutting enabler to achieve robust, explainable, and trustworthy medical artificial intelligence. *Inf. Fusion* 79 (2022), 263–278.

[41] Andreas Holzinger, Bernd Malle, Anna Saranti, and Bastian Pfeifer. 2021. Towards multi-modal causability with graph neural networks enabling information fusion for explainable AI. *Inf. Fusion* 71, 7 (2021), 28–37. DOI: https://doi.org/10.1016/j.inffus.2021.01.008

[42] Andreas Holzinger, Anna Saranti, Anne-Christin Hauschild, Jacqueline Beinecke, Dominik Heider, Richard Roettger, Heimo Mueller, Jan Baumbach, and Bastian Pfeifer. 2023. Human-in-the-loop integration with domain-knowledge graphs for explainable federated deep learning. In *Lecture Notes in Computer Science (LNCS)* Vol. 14065. Springer, 45–64. DOI: https://doi.org/10.1007/978-3-031-40837-3_4

[43] Miroslav Hudec, Erika Minarikova, Radko Mesiar, Anna Saranti, and Andreas Holzinger. 2021. Classification by ordinal sums of conjunctive and disjunctive functions for explainable AI and interpretable machine learning solutions. *Knowl.-based Syst.* 220 (2021), 106916. DOI: https://doi.org/10.1016/j.knosys.2021.106916

[44] Katarzyna Kaczmarek-Majer, Gabriella Casalino, Giovanna Castellano, Monika Dominiak, Olgierd Hryniewicz, Olga Kamińska, Gennaro Vessio, and Natalia Díaz-Rodríguez. 2022. PLENARY: Explaining black-box models in natural language through fuzzy linguistic summaries. *Inf. Sci.* 614 (2022), 374–399.

[45] Dmitry Kazhdan, Botty Dimanov, Lucie Charlotte Magister, Pietro Barbiero, Mateja Jamnik, and Pietro Lio. 2023. GCI: A (G) raph (C) oncept (I) nterpretation Framework. *arXiv preprint arXiv:2302.04899* (2023).

[46] Thomas N. Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[47] Ryan Kiros, Yukun Zhu, R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler. 2015. Skip-thought vectors. In *Proceedings of the Conference on Neural Information Processing Systems (NIPS'15)*.

[48] Narine Kokhlikyan, Vivek Miglani, M. Martín, E. Wang, B. Alsallakh, Jonathan Reynolds, Alexander Melnikov, Natalia Kliushkina, Carlos Araya, Siqi Yan, and Orion Reblitz-Richardson. 2020. Captum: A unified and generic model interpretability library for PyTorch. *ArXiv* abs/2009.07896 (2020).

[49] Satyapriya Krishna, Tessa Han, Alex Gu, Javin Pombra, Shahin Jabbari, Steven Wu, and Himabindu Lakkaraju. 2022. The disagreement problem in explainable machine learning: A practitioner's perspective. *arXiv preprint arXiv:2202.01602* (2022).

[50] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet classification with deep convolutional neural networks. *Adv. Neural Inf. Process. Syst.* 25 (2012), 1097–1105.

[51] Sebastian Lapuschkin, Stephan Wäldchen, Alexander Binder, Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. 2019. Unmasking Clever Hans predictors and assessing what machines really learn. *Nat. Commun.* 10, 1 (2019), 1–8.

[52] Jeff Larson, Surya Mattu, Lauren Kirchner, and Julia Angwin. 2016. Data and analysis for "How we analyzed the COMPAS recidivism algorithm." Retrieved from https://github.com/propublica/compas-analysis

[53] Scott M. Lundberg, Gabriel G. Erion, and Su-In Lee. 2019. Consistent Individualized Feature Attribution for Tree Ensembles. arXiv:1802.03888 [cs.LG]

[54] Scott M. Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In *Proceedings of the Conference on Advances in Neural Information Processing Systems*. 4765–4774.

[55] Björn Lütjens, Brandon Leshchinskiy, Christian Requena-Mesa, Farrukh Chishtie, Natalia Díaz-Rodríguez, Océane Boulais, Aruna Sankaranarayanan, Aaron Piña, Yarin Gal, Chedy Raïssi, Alexander Lavin, and Dava Newman. 2021. Physically-consistent generative adversarial networks for coastal flood visualization. *arXiv preprint arXiv:2104.04785* (2021).

[56] David J. C. MacKay. 2003. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press.

[57] Lucie Charlotte Magister, Dmitry Kazhdan, Vikash Singh, and Pietro Liò. 2021. GCExplainer: Human-in-the-loop concept-based explanations for graph neural networks. *arXiv preprint arXiv:2107.11889* (2021).

[58] Leland McInnes, John Healy, and James Melville. 2018. UMAP: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426* (2018).

[59] Carlo Metta, Andrea Beretta, Riccardo Guidotti, Yuan Yin, Patrick Gallinari, Salvatore Rinzivillo, and Fosca Giannotti. 2021. Explainable deep image classifiers for skin lesion diagnosis. DOI: https://doi.org/10.48550/ARXIV.2111.11863

[60] Dang Minh, H. Xiang Wang, Y. Fen Li, and Tan N. Nguyen. 2022. Explainable artificial intelligence: A comprehensive review. *Artif. Intell. Rev.* 55, 5 (June 2022), 3503–3568. DOI: https://doi.org/10.1007/s10462-021-10088-y

[61] Grégoire Montavon, Sebastian Lapuschkin, Alexander Binder, Wojciech Samek, and Klaus-Robert Müller. 2017. Explaining nonlinear classification decisions with deep Taylor decomposition. *Pattern Recog.* 65 (2017), 211–222.

[62] Ramaravind K. Mothilal, Amit Sharma, and Chenhao Tan. 2020. Explaining machine learning classifiers through diverse counterfactual explanations. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*. DOI: https://doi.org/10.1145/3351095.3372850

[63] Martin Müller, Marcel Salathé, and Per E. Kummervold. 2020. COVID-Twitter-BERT: A natural language processing model to analyse COVID-19 content on Twitter. *arXiv preprint arXiv:2005.07503* (2020).

[64] H. op den Akker, M. Cabrita, R. op den Akker, V. M. Jones, and H. J. Hermens. 2015. Tailored motivational message generation: A model and practical framework for real-time physical activity coaching. *J. Biomed. Inform.* 55 (2015), 104–115.

[65] Urja Pawar, Donna O'Shea, Susan Rea, and Ruairi O'Reilly. 2020. Explainable AI in healthcare. DOI: https://doi.org/10.1109/CyberSA49311.2020.9139655

[66] Karl Pearson. 1901. LIII. On lines and planes of closest fit to systems of points in space. *London, Edinb. Dubl. Philos. Mag. J. Sci.* 2, 11 (1901), 559–572.

[67] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT'18)*.

[68] Charles Pierse. 2021. Transformers interpret version 0.5.2. *Github Repository* (2021). Retrieved from https://github.com/cdpierse/transformers-interpret

[69] Galena Pisoni and Natalia Díaz-Rodríguez. 2023. Responsible and human centric AI-based insurance advisors. *Inf. Process. Manag.* 60, 3 (2023), 103273.

[70] Galena Pisoni, Natalia Díaz-Rodríguez, Hannie Gijlers, and Linda Tonolli. 2021. Human-centred artificial intelligence for designing accessible cultural heritage. *Appl. Sci.* 11, 2 (2021), 870.

[71] Phillip E. Pope, Soheil Kolouri, Mohammad Rostami, Charles E. Martin, and Heiko Hoffmann. 2019. Explainability methods for graph convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 10772–10781.

[72] Alun Preece, Dan Harborne, Dave Braines, Richard Tomsett, and Supriyo Chakraborty. 2018. Stakeholders in Explainable AI. arXiv:1810.00184

[73] Ayoub El Qadi, Maria Trocan, Natalia Diaz-Rodriguez, and Thomas Frossard. 2023. Feature contribution alignment with expert knowledge for artificial intelligence credit scoring. *Signal, Image and Video Processing* 17, 2 (2023), 427–434.

[74] Alec Radford, Jeff Wu, R. Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 8 (2019), 9.

[75] Carl O. Retzlaff, Alessa Angerschmid, Anna Saranti, David Schneeberger, Richard Roettger, Heimo Mueller, and Andreas Holzinger. 2024. Post-hoc vs Ante-hoc explanations: XAI design guidelines for data scientists. *Cognit. Syst. Res.* 86, 8 (2024), 101243. DOI: https://doi.org/10.1016/j.cogsys.2024.101243

[76] Carl O. Retzlaff, Srijita Das, Christabel Wayllace, Payam Mousavi, Mohammad Afshari, Tianpei Yang, Anna Saranti, Alessa Angerschmid, Matthew E. Taylor, and Andreas Holzinger. 2024. Human-in-the-loop reinforcement learning: A survey and position on requirements, challenges, and opportunities. *J. Artif. Intell. Res.* 79, 1 (2024), 349–415. DOI: https://doi.org/10.1613/jair.1.15348

[77] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. Why should I trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1135–1144.

[78] Peter J. Rousseeuw. 1987. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *J. Computat. Appl. Math.* 20 (1987), 53–65.

[79] Wojciech Samek, Grégoire Montavon, Alexander Binder, Sebastian Lapuschkin, and Klaus-Robert Müller. 2016. Interpreting the predictions of complex ML models by layer-wise relevance propagation. *arXiv preprint arXiv:1611.08191* (2016).

[80] Anna Saranti, Miroslav Hudec, Erika Mináriková, Zdenko Takáč, Udo Großschedl, Christoph Koch, Bastian Pfeifer, Alessa Angerschmid, and Andreas Holzinger. 2022. Actionable explainable AI (AxAI): A practical example with aggregation functions for adaptive classification and textual explanations for interpretable machine learning. *Mach. Learn. Knowl. Extract.* 4, 4 (2022), 924–953.

[81] Anna Saranti, Behnam Taraghi, Martin Ebner, and Andreas Holzinger. 2020. Property-based testing for parameter learning of probabilistic graphical models. In *Proceedings of the International Cross-domain Conference for Machine Learning and Knowledge Extraction*. Springer, 499–515.

[82] T. Schnake, O. Eberle, J. Lederer, S. Nakajima, K. T. Schütt, K. R. Müller, and G. Montavon. 2020. Higher-order explanations of graph neural networks via relevant walks. *arXiv: 2006.03589* (2020).

[83] Gesina Schwalbe and Bettina Finzel. 2023. A comprehensive taxonomy for explainable artificial intelligence: A systematic survey of surveys on methods and concepts. *Data Min. Knowl. Discov.* (2023), 1–59.

[84] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. 2017. Grad-CAM: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE International Conference on Computer Vision*. 618–626.

[85] Ramprasaath R. Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, and Dhruv Batra. 2020. Grad-CAM: Visual explanations from deep networks via gradient-based localization. *Int J Comput Vis* 128 (2020), 336–359. https://doi.org/10.1007/s11263-019-01228-7

[86] Luciano Serafini, Artur d'Avila Garcez, Samy Badreddine, Ivan Donadello, Michael Spranger, and Federico Bianchi. 2021. Logic tensor networks: Theory and applications. In *Neuro-Symbolic Artificial Intelligence: The State of the Art*. IOS Press, 370–394.

[87] Luciano Serafini and Artur d'Avila Garcez. 2016. Logic tensor networks: Deep learning and logical reasoning from data and knowledge. *arXiv preprint arXiv:1606.04422* (2016).

[88] Lloyd S. Shapley. 1953. A value for n-person games. In *Contributions to the Theory of Games II*, Harold W. Kuhn and Albert W. Tucker (Eds.). Princeton University Press, Princeton, NJ, 307–317.

[89] Laura Sikstrom, Marta M. Maslej, Katrina Hui, Zoe Findlay, Daniel Z. Buchman, and Sean L. Hill. 2022. Conceptualising fairness: Three pillars for medical algorithms and health equity. *BMJ Health Care Inform.* 29, 1 (2022).

[90] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning*. JMLR. org, 3319–3328.

[91] W. L. Taylor. 1953. "Cloze Procedure": A new tool for measuring readability. *Journal. Mass Commun. Quart.* 30 (1953), 415–433.

[92] Joe Townsend, Esma Mansouri-Benssassi, Kwun Ho Ngan, and Artur d'Avila Garcez. 2023. Discovering visual concepts and rules in convolutional neural networks. In *Compendium of Neurosymbolic Artificial Intelligence*, Pascal Hitzler, Md. Kamruzzaman Sarker, and Aaron Eberhart (Eds.). *Frontiers in Artificial Intelligence and Applications*, Vol. 369. IOS Press, 337–372. DOI: https://doi.org/10.3233/FAIA230148

[93] Malyshev Vadim and Anatoly Vershik. 2002. *Asymptotic Combinatorics with Application to Mathematical Physics*. Publisher Springer Dordrecht. DOI: https://doi.org/10.1007/978-94-010-0575-3

[94] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *J. Mach. Learn. Res.* 9, 11 (2008).

[95] Nicole Van Hoeck, Patrick D. Watson, and Aron K. Barbey. 2015. Cognitive neuroscience of human counterfactual reasoning. *Front. Hum. Neurosci.* 9 (2015). DOI: https://doi.org/10.3389/fnhum.2015.00420

[96] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762* (2017).

[97] Joseph D. Viviano, Becks Simpson, Francis Dutil, Yoshua Bengio, and Joseph Paul Cohen. 2019. Saliency is a possible red herring when diagnosing poor generalization. In *Proceedings of the International Conference on Learning Representations (ICLR'19)*.

[98] Minh Vu and My T. Thai. 2020. PGM-Explainer: Probabilistic graphical model explanations for graph neural networks. *Adv. Neural Inf. Process. Syst.* 33 (2020), 12225–12235.

[99] Benedikt Wagner and Artur d'Avila Garcez. 2021. Neural-symbolic integration for interactive learning and conceptual grounding. In *NeurIPS, Workshop on Human and Machine Decisions*. Retrieved from https://arxiv.org/abs/2112.11805

[100] Benedikt Wagner and Artur d'Avila Garcez. 2024. A neurosymbolic approach to AI alignment. *Neurosymbolic AI* Retrieved from https://neurosymbolic-ai-journal.com/system/files/nai-paper-729.pdf

[101] Benedikt Wagner and Artur S. D'Avila Garcez. 2021. Neural-symbolic integration for fairness in AI. In *Proceedings of the AAAI Spring Symposium (AAAI-MAKE'21)*.

[102] Leander Weber, Sebastian Lapuschkin, Alexander Binder, and Wojciech Samek. 2023. Beyond explaining: Opportunities and challenges of XAI-based model improvement. *Inf. Fusion* 92 (2023), 154–176.

[103] Lilian Weng. 2018. Attention? Attention! Retrieved from http://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html

[104] Adam White and Artur d'Avila Garcez. 2020. Measurable counterfactual local explanations for any classifier. In *Proceedings of the 24th European Conference on Artificial Intelligence*. Retrieved from http://arxiv.org/abs/1908.03020

[105] Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics, 1112–1122. Retrieved from http://aclweb.org/anthology/N18-1101

[106] Chih-Kuan Yeh, Been Kim, Sercan Arik, Chun-Liang Li, Tomas Pfister, and Pradeep Ravikumar. 2020. On completeness-aware concept-based explanations in deep neural networks. *Adv. Neural Inf. Process. Syst.* 33 (2020), 20554–20565.

[107] Seul-Ki Yeom, Philipp Seegerer, Sebastian Lapuschkin, Alexander Binder, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek. 2021. Pruning by explaining: A novel criterion for deep neural network pruning. *Pattern Recog.* 115 (2021), 107899.

[108] Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. 2019. GNNExplainer: Generating explanations for graph neural networks. *Adv. Neural Inf. Process. Syst.* 32 (2019).

Received 8 March 2023; revised 22 March 2024; accepted 8 May 2024