A Practical Tutorial on Explainable AI Techniques

Adrien Bennetot[a,b,c], Ivan Donadello[d], Ayoub El Qadi[c,f], Mauro Dragoni[e], Thomas Frossard[f], Benedikt Wagner[h], Anna Saranti[i], Silvia Tulli[k,c], Maria Trocan[g], Raja Chatila[c], Andreas Holzinger[i,j], Artur d'Avila Garcez[h], Natalia Díaz-Rodríguez[a,l]

[a]*ENSTA, Institut Polytechnique Paris and INRIA Flowers Team, Palaiseau, France*
[b]*Segula Technologies, Parc d'activité de Pissaloup, Trappes, France*
[c]*Sorbonne Université, Paris, France*
[d]*Free University of Bozen-Bolzano, Italy*
[e]*Fondazione Bruno Kessler, Trento, Italy*
[f]*Tinubu Square, Issy-les-Moulineaux, France.*
[g]*Institut Supérieur d'Électronique de Paris (ISEP), Issy-les-Moulineaux, France.*
[h]*City University London, U.K.*
[i]*Medical University Graz, Austria*
[j]*Alberta Machine Intelligence Institute, Edmonton, Canada*
[k]*INESC-ID and Instituto Superior Técnico, Lisbon, Portugal*
[l]*Andalusian Research Institute in Data Science and Computational Intelligence (DaSCI), University of Granada, Spain*

Abstract

Last years have been characterized by an upsurge of opaque automatic decision support systems, such as Deep Neural Networks (DNNs). Although they have great generalization and prediction skills, their functioning does not allow obtaining detailed explanations of their behaviour. As opaque machine learning models are increasingly being employed to make important predictions in critical environments, the danger is to create and use decisions that are not justifiable or legitimate. Therefore, there is a general agreement on the importance of endowing machine learning models with explainability. The reason is that 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 tutorial is meant to be the go-to handbook for any audience with a computer science background aiming at getting intuitive insights of machine learning models, accompanied with straight, fast, and intuitive explanations out of the box. We believe that this article provide a valuable contribution for applying XAI techniques in their particular day-to-day models, datasets and use-cases. Figure 1 acts as a flowchart/map for the reader and should help him to find the ideal method to use according to his type of data. The reader will find a description of the proposed method as well as an example of use and a Python notebook that he can easily modify as he pleases in order to apply it to his own case of application.

Keywords: Explainable Artificial Intelligence, Machine Learning, Deep Learning, Interpretability, Shapley, Grad-CAM, Layer-wise Relevance Propagation, DiCE, Counterfactual explanations, TS4NLE, Neural-symbolic learning

1. Introduction

In systems based on machine learning, one has to consider that there will unavoidably be faulty system decisions. This is due to several reasons, originating from data (e.g., bias) or from the system's design (e.g., network structure, connectivity, optimization process, bugs, or code quality management, etc.). This leads to a given system performance in terms of accuracy, false negatives or false positives compared to ground truth. However, this is actually an unavoidable feature of DNNs, and not simply bugs in the system that can be corrected. An interesting approach is to consider this situation from a software engineering viewpoint.

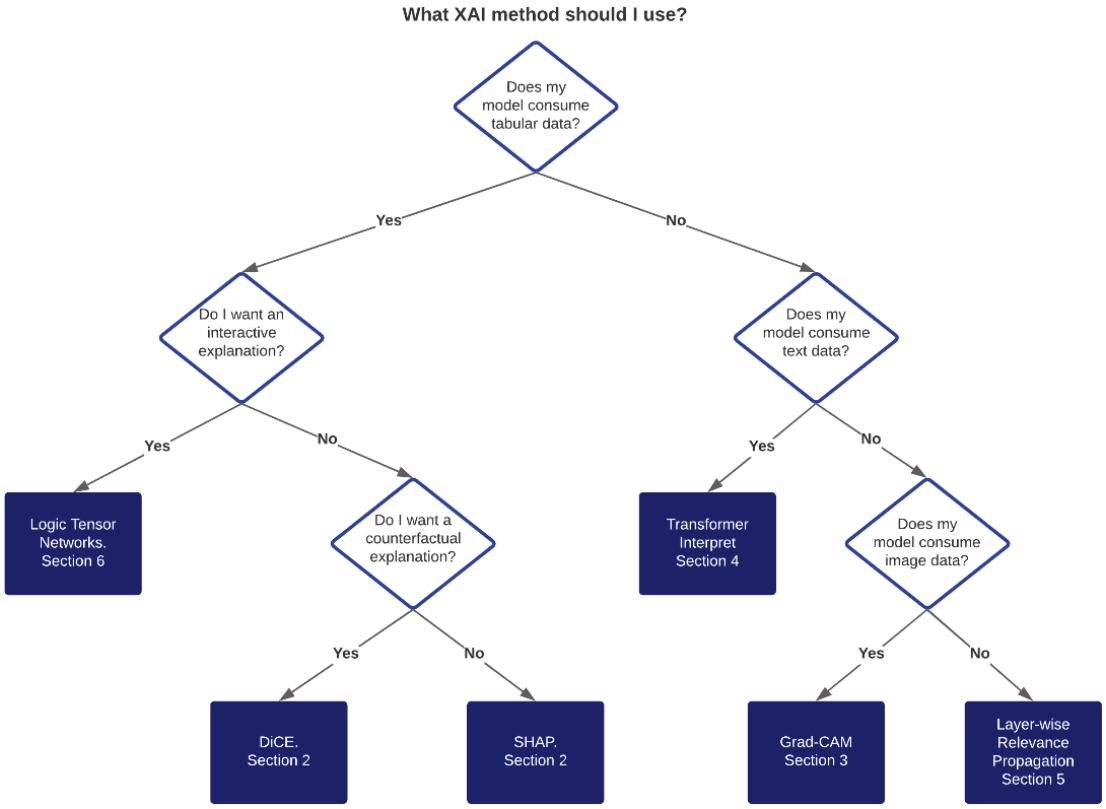

Figure 1: Map/Flowchart of the XAI methods described in the article. It is not meant to be exhaustive regarding all the existing XAI methods as it presents only the existing methods in the paper. Moreover, the Layer-wise relevance propagation method is recommended when the data are neither textual, tabular nor images, but it can be used on these type of data as well. Also, SHAP can be used on images but this particular use-case is not in the scope of this article.

One can consider that this result is a faulty system behavior, with respect to its intended objectives and specifications. Following principles of dependable and robust system design could insure a safer system operation. However this is based on proper tools for understanding the system's behavior and outputs among which explainability. This performance can be largely insufficient to trust the system and could lead to catastrophic outcomes in critical applications e.g., where human lives can be at stake. Therefore, there is a need for tools for understanding the system's behavior and outputs, i.e. the need for explainability.

Explainability refers to the details and reasons a model gives to make its functioning clear or easy to understand [3]. The literature makes a clear distinction among models that are interpretable by design and those that can be explained by means of external methods. EXplainable AI (XAI) techniques are increasingly being used by a wider audience and are starting to be applied in multiple fields in industry and in academia. As more and more techniques appear and as it is often complex to interpret or convert their explanatory elements into an actionable explanation, i.e., that either a developer or an expert can transform into an action to fix the model, we present a tutorial of some of the most used XAI methods producing explanations in common formats applied on models ingesting different formats of data (image, tabular, textual and graphs) and models using neural-symbolic computation. We also consider some user interface aspects to better interact with then non technical user, such as using neural-symbolic computation models to produce natural language explanations, and counterfactual explanations.

The goal of this tutorial is to serve as a practical and rapidly usable tool for any developer wishing to

Table 1: Table referencing the different XAI methods developed in the paper as well as the link to the associated *Google Colab*, which the user can use to apply the method to his own use-case. Note that certain methods can also be used for other data types than the one described in the table but only the data types treated in the tutorials are mentioned.

XAI Method	Data Type	Explanation Type	ML Task Explained	Dataset used
SHAP [20]	Tabular	Feature Importance	Credit RIsk Scoring	Zindi[1]
Original **SHAP** Repository: https://github.com/slundberg/shap				
Proposed **SHAP** Tutorial: https://colab.research.google.com/drive/1HuhpUAl4s9ZIs3yWHsAExApLxGAv3NuH?usp=sharing				
DiCE [23]	Tabular	Counterfactual	Credit Risk Scoring	Zindi
Original **DiCE** Repository: https://github.com/interpretml/DiCE				
Proposed **DiCE** Tutorial: https://colab.research.google.com/drive/1nUTTTfcCuxsnZmaJpfvLsxRB4FFaORVK?usp=sharing				
Transformer Interpret (TI) [25]	Textual	Feature Importance	Sentiment Analysis	MNLI[2]
Original **TI** Repository: https://github.com/cdpierse/transformers-interpret				
Proposed **TI** Tutorial: https://colab.research.google.com/drive/1XGGXUYNC1M_jlmQUV5dZB3HVdQRXeghd				
Grad-CAM [32]	Image	Visual	Image Classification	ImageNet
Original **Grad-CAM** Repository: https://keras.io/examples/vision/grad_cam/				
Proposed **Grad-CAM** Tutorial: https://colab.research.google.com/drive/1bA2Fg8TFbI5YyZyX3zyrPcT3TuxCLHEC?usp=sharing				
Layer-wise Relevance Propagation (LRP) [22]	Graph	Visual	Image Classification	GDC[3]
Original **LRP for Graphs** Repository: https://git.tu-berlin.de/thomas_schnake/demo_gnn_lrp				
Proposed **LRP for Graphs** Tutorial: https://colab.research.google.com/drive/166FYIwxblfrEltkYqY_jiJoAm9VLMweJ?usp=sharing				
Logic Tensor Networks (LTN) [33]	Textual	Interactive	Recidivism Prediction	COMPASS[4]
Original **LTN** Repository: https://github.com/logictensornetworks/logictensornetworks				
Proposed **LTN** Tutorial: https://colab.research.google.com/drive/1Ip9Yb9gVRSRqaBKY9gOpiWn9pq3LovWG?usp=sharing				
TS4NLE [12]	Textual	Natural Language	Explanation Rendering	UMLS[5]
Original **TS4NLE** Repository: https://github.com/ivanDonadello/TS4NLE				
Proposed **TS4NLE** Tutorial: https://colab.research.google.com/drive/1iCVSt7TFMruSzeg5DswLOzOR1n7xATbz				

obtain explanatory elements for his Deep Learning model. Contrarily to other tutorials that focus on some particular datatype like textual [6], we provide a broader view on XAI methods by addressing most of the data types and issues faced by users wishing to explain their Deep Learning models.

To this end, we describe an XAI method for each of the most common data types: tabular in Section 2, images in Section 3 and textual in Section 4 as well as a general method in Section 5. We accompany these techniques with a description of the neuro-symbolic methods used to make interactive explanations 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 are free and open source to be easily reused, adapted or taught and are presented with a systematic introduction of fundamental theories and common practices with practical use cases and suitability analysis for any application, task or data type, with concrete examples of use cases and interactive codes. Table 1 references the different methods and associated *Google Colab* described in the article so that the reader can quickly access the method that interests him.

2. XAI techniques for tabular data

Model-agnostic interpretation methods separate the explanations from the machine learning model. The flexibility in such methods lies in the ability to use the method to explain any machine learning model. We will explore two different post-hoc explanations methods for tabular data: SHapley Additive

exPlanations (SHAP) [20] and DIverse Counterfactual Explanations (DICE) [23]. We will apply both methods to a particular problem: credit scoring.

2.1. SHAP Analysis

In this section, we will focus on one of most used model-agnostic method, the SHapley Additive exPlanations (SHAP) and its use for tabular data.

SHAP is based on game theory, in particular on the Shapley values. The original framework was built in order to re-allocate in a rightful way the gain of a cooperative game among players. 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 (i.e., the model without the feature i) while ϕ_i corresponds to the contribution of feature i to the model prediction.

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

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. This difference is attributed to feature i.

Since it is computationally expensive to consider all possible sets of features S and average the difference in prediction due to the feature i, SHAP generates a random sampling of the possible sets of S to compute the average. This average represents the estimated feature importance. SHAP exhibits numerous desirable properties, such as singularity detection (i.e., if the feature is locally zero, the SHAP value is 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, so will be its SHAP value).

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 [8]. Since in the financial industry, and specially 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, 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".

Extreme Gradient Boosting (XGBoost) [8], 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 derived in strict regulations such as the European General Data Protection (GDPR) and Ethics guidelines for trustworthy AI. 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 tutorial, 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 [7]. 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 [27]). Finally we apply the SHAP framework to understand the main reasons why a 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, the model will predict that the loan will be amortized. Otherwise the customer will not be able to return the loan.

2.1.2. SHAP explanation visualization

The SHAP technique 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 show 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 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– to the probability of default (i.e., fail to pay back).

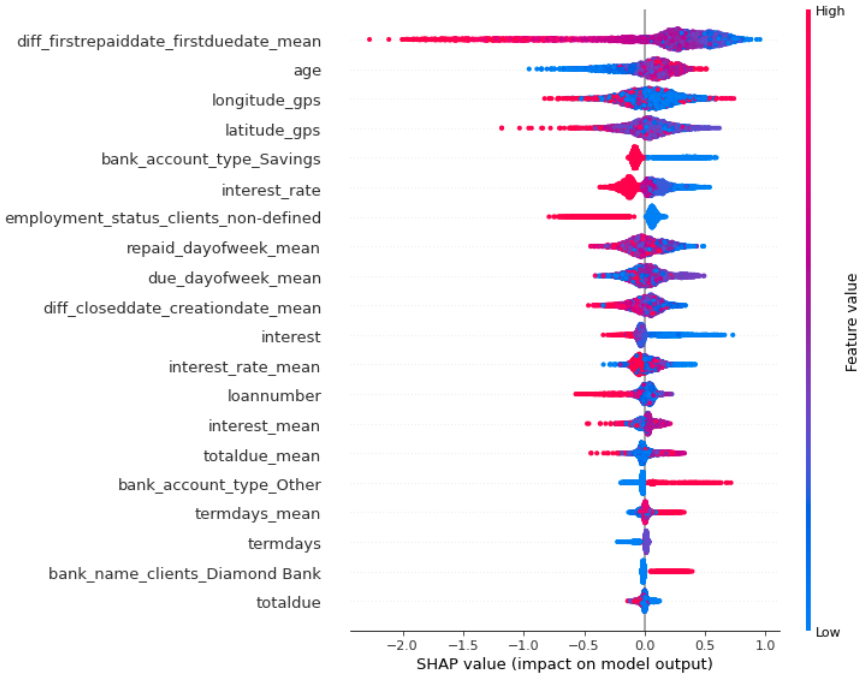

Figure 2: X-axis represents the SHAP value (i.e., contribution) for each feature of the loan default prediction model, while y-axis indicates the features, ranked by importance from top to bottom. Each point represents a data point (i.e., a single individual). Feature values are encoded using a color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

Contribution of each explanatory feature to the final prediction based on 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 Fig. 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[6].

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 [19] that allows to compute the exact SHAP values in a polynomial time faster by keeping track of the number of subsets S that flow into each node of the tree. Another problem is that the SHAPley value can change with the order of features selected, and thus, for an exact computation of SHAP values, all possible combination 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 the human expert explanation [27] or a background solid scientific model [21] in order to converge towards a commonly aligned criteria explanation. This is a crucial requirement for ML model adoption in applications involving critical decisions.

2.2. DiCE: Diverse Counterfactual Explanations

DiCE considers the problem of generating counterfactual explanations from a set of counterfactual (CF), 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 [23]. 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.

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

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 which is the d

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

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

C is the generated set of k CFs generated for the example x that minimizes the following function (as in [23]):

$$C(x) = arg\ min_{c_1,c_2,...,c_k} \frac{1}{k} \sum_{i=1}^{k} yloss(f(c_i), y) + \frac{\lambda_1}{k} \sum_{i=1}^{k} dist(c_i, x) - \lambda_2 dpp\_diversity(c_1, c_2, ..., c_k) \tag{5}$$

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. The diversity is represented by the term $dpp\_diversity = det(K)$ 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 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 consumers default

We will focus on the use case presented in subsection 2.1. 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 be denied. In this particular use case, the interest is in advising the customer (i.e., the stakeholder) what to do in order to get the credit. We are interested in generating counterfactual examples for the clients with denied loans[7].

In Fig. 3, we generate 3 counterfactual explanations for a given loan applicant. Initially the considered applicant has been denied the loan by the Random Forest [5] model. The CF examples show which features the applicant should change in order to get the loan approved. For example, if the applicant had a job and the amount of the loan was smaller, the loan would have been approved (bottom right point in the Fig.3).

2.2.2. 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 machine learning model. It equally allows to produce explanations that are easily conveyable not only to developers but also to a non technical audience. On the other hand, currently, the disadvantage of using DiCE is that works only for differentiable models, since it uses gradient descent for the optimization process.[8].

3. XAI techniques for images

Convolutional Neural Networks (CNNs) constitute the state-of-art models in all fundamental computer vision tasks (image classification, object detection, instance segmentation) as of 2021. They 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, this tutorial presents a technique that try to understand the decision process of a Convolutional Neural Network (CNN) by mapping back the model output into the input space to see

[7] DiCE tutorial 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

[8] DiCE original source: https://github.com/interpretml/DiCE. Guided DiCE tutorial and credit risk scoring example tutorial: https://colab.research.google.com/drive/1nUTTTfcCuxsnZmaJpfvLsxRB4FFaORVK?usp=sharing

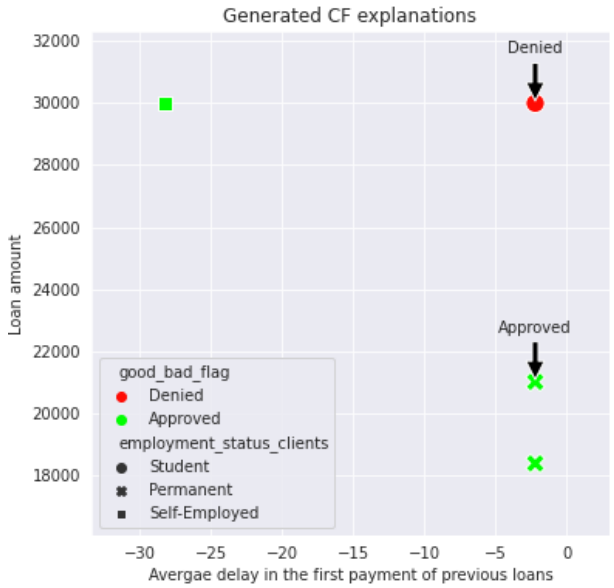

Figure 3: Counterfactual (CF) examples generated using DiCE [23]. Green points are cases where the loan applicant would be approved. We selected a set of variables that are the easiest for the customer to vary (i.e., employment status, the behaviour on the payment of previous loans and the loan amount demanded). We generate 3 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.

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.

3.1. Grad-CAM use case: image classification

Gradient-weighted Class Activation Mapping (Grad-CAM) [32] 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. In other words, the Grad-CAM technique makes it possible to know which part of the image contributed the most to the model's prediction.

In order to obtain the class-discriminative localization map $L^c_{Grad-CAM}$ for any class c, 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 forward activation maps with a Rectifier Linear Unit (ReLU) activation function. We usually normalize

the heatmap and color it to make it more visually interpretable[9].

$$L^{c}_{Grad-CAM} = ReLU(\sum_{k} \alpha^{c}_{k} A^{k}) \tag{7}$$

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 $ClassActivationMap \leftarrow Mean(Activations)$

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 Fig.4, we want the model to be able to detect the rabbit because it contains features typical of a rabbit and not because there is a grated carrot and a kitchen-like background since we want to be able to detect a rabbit even when there is no carrot nor any other typical background.

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 [13]. 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

[9]Grad-CAM tutorial available at: https://colab.research.google.com/drive/1bA2Fg8TFbI5YyZyX3zyrPcT3TuxCLHEC?usp=sharing refined from https://keras.io/examples/vision/grad_cam/

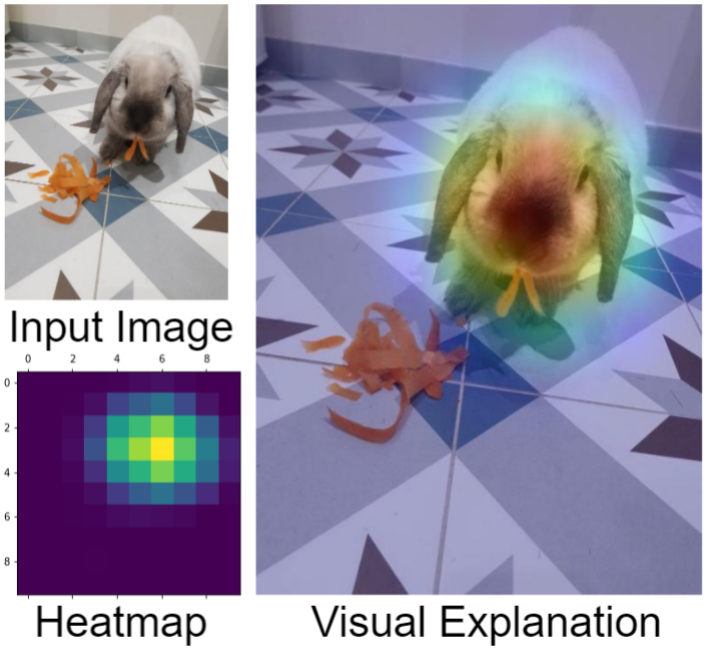

Figure 4: Superimposed visualization of Grad-CAM's heatmap and the input image, showing the model mostly used the center-right of the image (where the head of the rabbit is) in order to make its prediction.

which pixels. Guided Grad-CAM [31] 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 [1] as it was shown the relationship between *good* saliency and generalization performance is tenuous that improved generalization is not always accompanied by improved heatmaps [37].

4. XAI Techniques for Textual Data

In this section we investigate techniques to explain textual data. 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 language tasks. Examples of such tasks include, but are not limited to, tokenization, text classification, speech recognition, machine translation, image caption generation, machine translation, question answering system creation, and document summarization.

Due to the complexity of these NLP models, a strand of explainability research focuses on facilitating the identification of different features that contribute to their outputs [18]. Natural Language Processing models' attempt to extract information and insights from natural language data. Danilevsky et al. [10] clustered the XAI literature for NLP with respect to the type of explanation (i.e., post-hoc, self-explaining), and whether the information or justification for the model's prediction concerns a specific input (i.e., local), or the functioning of the model as a whole (i.e., global). Their taxonomy identified five main explanations techniques, namely: feature importance[38], surrogate model[28], example-driven[9], provenance-based[42], and declarative induction[26].

Among the existing 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 [36]. Transformer models are general-purpose architectures such as BERT and GPT-2.

The Transformer architecture works by weighing the influence of different parts of the input data, and aims at reducing sequential computation by relying entirely on the self-attention mechanism to compute a representation of its inputs and outputs [36]. In practice, the encoder represents the input as a set of key-value pairs, $(\mathcal{K}, \mathcal{V})$, of dimension dk and dv respectively. The decoder packs the previous output into a query $\mathcal{Q}$ of dimension m and obtains next output by mapping this query against the set of keys and values [40]. The matrix of outputs, also called score matrix, determines the importance a specific word has 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 and all keys (Eq. 8).

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

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 [40].

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

In the multi-head attention (Eq. 9), h corresponds to the parallel attention layers (i.e., heads) and the Ws are all learnable parameter matrices (i.e., $W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$ and and $W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$).

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

Bidirectional Encoder Representations from Transformers (BERT) [11] is a transformer-based machine learning technique for NLP pre-training developed by Google. The peculiarity of this technique is that applies bidirectional training to language modeling.

In contrast to directional models that read the text input sequentially (e.g., OpenAI GPT, ELMo), a bidirectional encoder processes the entire sequence of words at once. This way of processing words allows BERT to learn to unambiguously contextualize words based on both left and right words, and by repeating this process multiple times (i.e., multi-head), to learn different contexts between different pairs of words. Fig. 5 describes the BERT Architecture. To define the goal of the prediction, BERT makes use of two techniques: Masked Language Modeling (MLM), and Next Sequence Prediction (NSP). The former consists of substituting approximately the 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 giving 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 the Transformer models, there exist a number of interfaces for exploring the inner workings of transformer models. *Captum*[10] is a multi modal package for model interpretability built on PyTorch. Captum attribution algorithms can be grouped in three main categories; primary, layer and neuron attribution algorithms. Primary attribution algorithms allow us to attribute output predictions to model inputs. Layer attribution algorithms allow us to attribute output predictions to all neurons in the hidden layer. Neuron attribution algorithms allow us to attribute an internal, hidden neuron to the input of the model[18].

[10]Captum: https://github.com/pytorch/captum

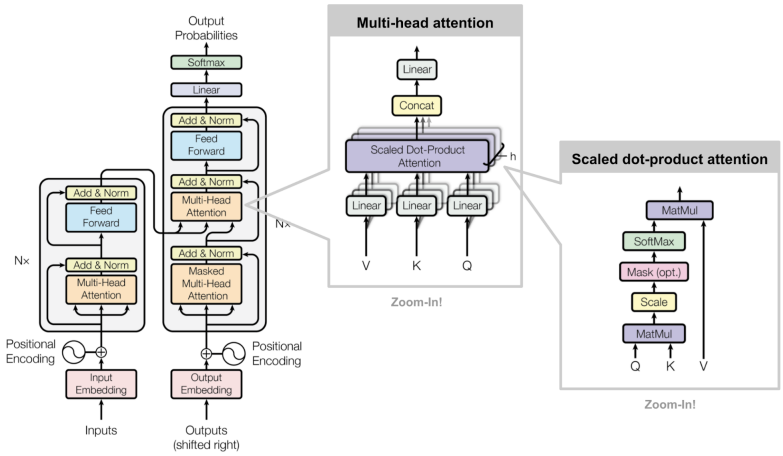

Figure 5: High-level overview of the BERT Transformer model: the input is a sequence of tokens embedded into vectors, the output is a sequence of vectors linked by the index to the input tokens. The encoder multi-head attention mechanism computes queries, keys and values from the encoder states. The encoder feed-forward network takes additional information from other tokens and integrates them in the model. The decoder masked multi-head attention mechanism computes queries, keys and values from the decoder states. The decoder multi-head attention mechanism looks at the source of the target tokens taking the queries from the decoder states and the keys and values from the encoder states. The decoder feed-forward network takes additional information from other tokens and integrates them in the model (original image from [36]).

4.2. Explaining Transformer Models with Transformer Interpret

Transformer-Interpret[11] is a dedicated tool for interpreting Transformer Models. The default attribution method used by Transformer-Interpret is Integrated Gradients (IG) [34].

IG's goal is 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 differents.

Early interpretability methods for neural networks assigned feature importance scores using gradients because computing the gradients of the input with regard to the output is *Implementation Invariant* but does not satisfy *Sensitivity* as a feature change does not necessarily yield a non-zero gradient for that feature. IG adds *Sensitivity* to gradients based methods by establishing a baseline (a reference input with an equiprobable prediction between the different classes) and compute the sum of all the gradients from this baseline to the input of interest (the one we try to explain). This allows to know how the change of value of a feature of the input leads to a change in the classification

A commonly text classification task is sentiment analysis. Sentiment analysis aims at detecting positive, neutral or negative sentiment in text. Transformer-Interpret help us identify and visualize how positive/negative attribution numbers associated to a word contributes positively/negatively towards the predicted class, thanks to IG, as shown in Fig 6.

.

[11] BERT Transformer Interpreter is available here https://colab.research.google.com/drive/12_Q1WI05oXMfG-B_GwyiLjyU-rxRyFdX?usp=sharing

Figure 6: Transformer Interpret to Analyse Sentiment Analysis thanks to Integrated Gradients. Picture from [25]

Legend: ■ Negative □ Neutral ■ Positive

True Label	Predicted Label	Attribution Label	Attribution Score	Word Importance
POSITIVE	**POSITIVE (1.00)**	**POSITIVE**	**2.02**	[CLS] i love you , i like you [SEP]

The Transformer-Interpret tool has the advantage of relying on two well documented packages and frameworks (e.g., Captum and HuggingFace Transformers). It provides simple methods to explain most common natural language processing tasks performed by Transformer models, such as sequence classification, zero-shot classification, and question answering.

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 [22]. The heatmap's data structure and size is the same as the input's and its highlighted parts denote the areas of the input that played the highest role in the classification. LRP is an XAI method that applies to several (any) 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. 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. 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 write:

$$f(\mathbf{x}) = f(\tilde{\mathbf{x}}) + \left(\frac{\partial f}{\partial \mathbf{x}} \Big|_{\mathbf{x}=\tilde{\mathbf{x}}} \right)^T (\mathbf{x} - \tilde{\mathbf{x}}) + \epsilon = 0 + \sum_p \underbrace{\frac{\partial f}{x_p} \Big|_{\mathbf{x}=\tilde{\mathbf{x}}} (x_p - \tilde{x}_p)}_{R_p(\mathbf{x})} + \epsilon \tag{10}$$

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 Eq. 10 as follows:

$$f(\mathbf{x}) = 0 + \sum_p \underbrace{\frac{\partial f}{x_p} \Big|_{\mathbf{x}=\tilde{\mathbf{x}}} (x_p - \tilde{x}_p)}_{R_p(\mathbf{x})} + \epsilon \tag{11}$$

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 Eq. 10:

$$\begin{aligned} &\textstyle\sum_j R_j = \left(\frac{\partial (\sum_j R_j)}{\partial \{x_i\}} \Big|_{\partial \{\tilde{x}_i\}} \right)^T (\{x_i\} - \{\tilde{x}_i\}) + \epsilon = \\ &\textstyle\sum_i \sum_j \frac{\partial R_j}{\partial x_i} \Big|_{\partial \{\tilde{x}_i\}} (x_i - \tilde{x}_i) + \epsilon \end{aligned} \tag{12}$$

$$\sum_j R_j = \left(\frac{\partial(\sum_j R_j)}{\partial\{x_i\}}\Big|_{\partial\{\tilde{x}_i\}} \right)^T (\{x_i\} - \{\tilde{x}_i\}) + \epsilon = \sum_i \sum_j \frac{\partial R_j}{\partial x_i}\Big|_{\partial\{\tilde{x}_i\}} (x_i - \tilde{x}_i) + \epsilon \quad (13)$$

One of the challenges of LRP is to find a neighbouring point $\tilde{\mathbf{x}}$ of $\mathbf{x}$, for which $f(\tilde{\mathbf{x}}) = 0$ (root point). A good root point is one that 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 gray/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 also a heatmapping method, Grad-CAM does not compute relevances 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 onto that architecture, whereas LRP has shown to be more universal.

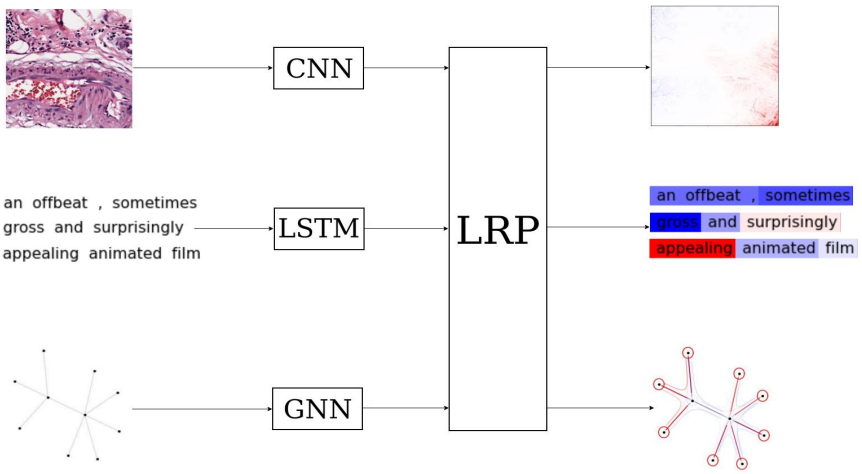

Figure 7: Layer-wise Relevance Propagation (LRP) applied onto various neural network architectures, each of them processing different types of data.

5.1. LRP applied onto Fully Connected (FC) Neural Network that solves a regression problem

In this task[12], 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

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

the i-th and j-th 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 $x_k = \sum_j x_j$.

Relevances of each neuron at each layer (indexed by i, j and k correspondingly) are computed by the following set of equations:

$$R_k = x_k = \sum_j x_j, \quad R_j = x_j = max(0, \sum_i x_i w_{ij} + b_j), \quad R_i = \sum_j \frac{w_{ij}^2}{\sum_{i'} w_{i'j}^2} R_j \tag{14}$$

where i' denotes all neurons of the input layer, including the i-th 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 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 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 a multiplication with a small weight) and not create a high activation. On the other hand, 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.

The backpropagation of relevance happens only once after the training is accomplished. Furthermore, two properties of LRP are used for verification of the computations with unit tests [29], namely positivity and conservation (Eq. 15) of relevance of the neurons at each layer:

$$\forall x, p : R_p(x) \geq 0, \quad \sum_i R_i = \sum_j R_j \tag{15}$$

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

5.2. Explaining three different CNN architectures performing image classification with LRP

Convolutional Neural Networks (CNNs) have shown to be very successful in various complex tasks including visual recognition and image classification tasks. There are several CNN architectures like VGGNet, GoogLeNet (Inception-v1), and ResNet that demonstrated remarkable performance achievements; Inception-v3 is the most popular architecture in digital pathology.

The VGGNet architecture is composed of small convolutional filters of size 3×3 and 2×2 pooling layers at specific points in the network. The use of small filters and simultaneously increased depth is what gives this architecture the capacity to decompose larger features into smaller ones and enable complex combinations. GoogLeNet is deeper than VGGNet; nevertheless, it has much fewer parameters. The novelty was the definition of an Inception module which contains several convolution filters of different sizes and their results are concatenated. The ResNet architecture introduced skip connections; by that means it can overcome the vanishing and exploding gradient problem and enable an even deeper network.

The benefits of visual explanations by computing high resolution heatmaps plays important role in the evaluation of the model from patch-level to the relevant level of cells.

The pixel contributions, visualized as heatmaps allow human experts to verify classification decisions made by machine learning algorithms. In this task, three state-of-the-art CNN architectures will be trained with whole-slide histopathology images, available on the Genomic Data Commons (GDC) Data Portal[13].

A classifier should base its decision on all different occurrences of the corresponding cell type to ensure accurate predictions on whole-slide images.

[13] GDC dataset https://portal.gdc.cancer.gov/, generated by The Cancer Genome Atlas (TCGA) Program http://cancergenome.nih.gov/ and publicly available. The data consists of 51 whole-slide histopathology images, from which 24 are lymph nodes and 27 esophagus whole-slide images (WSI).

Resizing those images into 224×224, to be an adequate input to the CNNs, contributes to loss of information and poor performance.

To achieve even higher performance, transfer learning with the use of pre-trained weights was used. Data augmentation operations applied on the training set, such as rotation, shifting, flipping, zooming, and shearing can increase the diversity of training data and lead to better generalization. The use of LRP on several already trained CNNs, with pre-trained weights and augmentation or not the observation, as well as the comparison of performances and heatmaps, shows ways to respond to how methods against overfitting help performance and interpretability, how do state-of-the-art CNN architectures process images, what are the differences between architectures w.r.t. performance and interpretability, what is the impact of bad performance on the XAI heatmaps, and how do heatmaps of misclassified samples look like.

5.3. Explaining an LSTM doing sentiment analysis with LRP

LRP can be applied on both LSTMs and bi-directional LSTMs [2] that process sequential data (one dimensional signals and text in the form of sequences). The LSTM architecture uses cells that contain several components (gates): the input signal s, gate g, forget gate f and accumulation neuron k. Each of them has a corresponding functionality. $s-1, g-1, f-1, k-1$ are the neurons in the LSTM cell that come immediately before them and represent the processing of the previous time step (the neuron of the previous layer is also denoted by j). The forward pass of the LSTM architecture is represented with the following set of equations:

$$z_s = \sum_j a_j w_{js}\ ,\ z_g = \sum_j a_j w_{jg}, \quad a_p = \tanh\big(z_s\big) \cdot \mathrm{sigm}\big(z_g\big), \quad a_k = a_f \cdot a_{k-1} + a_p \tag{16}$$

where w the weight between two components, and z the vector of all activations at a specified component. All those terms denote linear mappings, $tanh()$ and $sigm()$ are the hyperbolic tangent and sigmoid function correspondingly, whereas p is the product of s and g (second part of equation set 16). a_p represents the activation at the component p, a_j the activation at the neuron in layer j. The third part computes the activation in the accumulation neuron.

The LRP-ϵ rule computes the relevance of the neuron at level j with the use of the formula 17, where $\epsilon_s = \epsilon \cdot \mathrm{sign}\big(\sum_{0,j} a_j w_{js}\big)$ is a constant to avoid division by zero in case activations have a small value.

$$R_j = \sum_s \frac{a_j w_{js}}{\epsilon_s + \sum_{0,j} a_j w_{js}} R_s \tag{17}$$

Our sentiment analysis example task[14] consists of a five-class sentiment prediction of sentences from the Stanford Sentiment Treebank (SST) dataset[15]. The already trained LSTM network is run on two pre-selected sentences from the test set: one correctly classified and one misclassified. Our goal is not to compute the word relevances, but writing the code that will show how the removal of the N (user-specified) most relevant words influences the prediction of this class, where the influence is measured by the word relevance.

5.4. 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 processes non-grid structured data, therefore the filters cannot operate by the same means.

[14] Notebook on LRP for an LSTM trained on text data: https://colab.research.google.com/drive/1M7f2CI5Khyh0W1MbkQZUWScVjeQ2lvcp?usp=sharing

[15] SST Dataset https://nlp.stanford.edu/sentiment/index.html

One of the simplest architectures is called GCN (Graph Convolutional Network) [30]. The rules for aggregation and combination (Eq. 18) of the information lying in the features of the neighboring nodes and edges are:

$$\boldsymbol{Z}_t = \boldsymbol{\Lambda}\boldsymbol{H}_{t-1}, \quad \boldsymbol{H}_t = \rho(\boldsymbol{Z}_t\boldsymbol{W}_t) \tag{18}$$

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, etc. $\boldsymbol{\Lambda}H_{t-1}$ 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 [30] applies constraints (piecewise linear and positive homogeneity) to this nonlinearity (here ReLU is used). Eq. 18 is re-written and the GNN-LRP rule for computing the relevance $R_{jKL...}$ of neuron j after one has processed nodes K and L by all neurons k that gathered information from node K becomes:

$$R_{jKL...} = \sum_{k\in K} \frac{\lambda_{JK} h_j w^{\wedge}_{jk}}{\sum_J \sum_{j\in J} \lambda_{JK} h_j w^{\wedge}_{jk}} R_{kL...} \tag{19}$$

where K, L are elements of a walk on the input graph[16] 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 different visual details. λ_{JK} is the element of the Laplacian matrix $\boldsymbol{\Lambda}$ corresponding to the connection between nodes J and K, h_j is the activation of neuron j, $R_{kL...}$ is the relevance of neuron k after one has processed node L. The "..." indicate 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, use 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[17] is a slightly changed version of the researchers' original's[18]. The task is studying whether the 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 term of neurons) with the aim of obtaining a model output and output explanation that are 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 the 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

[16] A walk on the graph involves nodes that are processed by corresponding neurons at layers labeled with the same (non-capital) indexes.

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

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

will have high relevance, but we will not know which node exactly it is and which feature exactly it is. 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 in sorted order.

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 a symbolic level where reasoning is performed over abstract, discrete entities, following logical rules. A common goal of symbolic systems is to model reasoning, which inherently allows for better explainability. Neural networks, on the other hand, operate in the sub-symbolic (or *connectionist*) level. Individual neurons do not necessarily represent a readily recognisable concept or any discrete concept.

The integration between both levels can bridge low-level information processing –such as the one frequently encountered in perception and pattern recognition– with reasoning and explanation on a higher, more cognitive level of abstraction. Realising this integration facilitates a range of benefits, such as achieving representations that are abstract, re-usable, and general-purpose. Having these readily available could concretely tackle some of the pressing issues with current deep learning practices.

6.1. XAI in Neural-Symbolic AI

Explainability in neuro-symbolic systems has been traditionally approached by learning a set of symbolic rules, known as Knowledge Extraction, and evaluating how well it may approximate the behaviour of a complex neural network by measuring the percentage of matching predictions on a test set, referred to as fidelity [41, 35]

This is comparable to most contemporary explainability methods that are not powerful enough to guarantee the soundness and completeness of the explanation concerning the underlying model. Most metrics currently in place are lacking a reliable way of expressing this uncertainty. For instance,

The measured fidelity is supposed to be a good proxy of the closeness of the representation to the underlying model. However, this metric is limited in its capacity and ability to find semantically meaningful representations that allow for transparent reasoning, as it is solely optimizing for the resemblance of the explained model.

6.2. Framework for interactive explainability

In a more tightly integrated Neural-Symbolic system, XAI occurs in line with the neural-symbolic cycle. In this framework, we can query and revise the information and consolidate existing background knowledge. The system can utilize background knowledge to provide meaningful semantics for the explanations, facilitating human-machine interaction and achieving desired 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 [16], incorporated into the cycle as a constraint on the learning process. This enables an interactive integration of a desired behaviour, notably fairness constraints, by verifying and incorporating knowledge at each cycle, instead of (global or local) XAI serving only to produce a one-off description of a static system.

The neural-symbolic cycle can be seen as a common ground for communication and system interaction, allowing for a human-in-the-loop continual learning approach. This enables an interactive integration of a desired behaviour, notably fairness constraints, by verifying and incorporating knowledge at each cycle, instead of (global or local) XAI serving only to produce a one-off description of a static system.

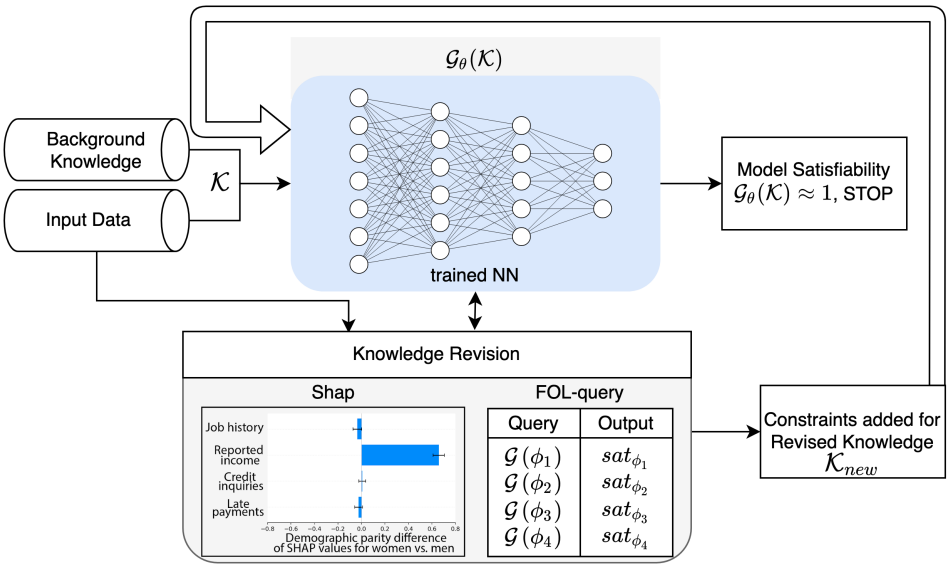

Figure 8: Illustration of the LTN pipeline for continual interactive learning: revision is carried out by querying a deep network interactively and learning continually, thus applying the neural-symbolic cycle multiple times. Explanations extracted from the network using, e.g., SHAP can highlight bias 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?*, by translating into FOL queries and checking their degree of satisfiability (*sat()*) (c.f. section 6.4). Subsequently, such desired queries can be added to the optimisation function. This process concludes once it has been shown to reduce bias at a subsequent SHAP explanation.

Symbolic knowledge representation extracted from the learning system at an adequate abstract level for communication with users should allow knowledge consolidation and targeted revision.

The following example will demonstrate how to use the Logic Tensor Networks (LTN) framework for explainable classification tasks and subsequently address some undesired model properties according to the pipeline in Figure 8. In it, we use the Shapley method, but any other XAI method could have been chosen, and an integrated logical Neural Network querying mechanism, to gain insights into the model during our knowledge revision process.

6.3. Logic Tensor Networks (LTN) for explainable model revision

The framework used in this approach and accompanying notebook[20] is LTN [33] and [4]. However, instead of treating the learning of the parameters from data and knowledge as a single process, we emphasise the dynamic and flexible 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 knowledge (FOL). We make the explanation approach iterative by saving the learned parametrisation at each cycle in our implementation, while changing the original LTN implementation from Neural Tensor Networks to a feed-forward Neural Network to demonstrate the agnostic nature of the approach.

Whereas many inherently neural-symbolic methods come with stringent architectural constraints on the model itself, this 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 behaviour, that is, observing the value of an output given a predefined input. The predictive model itself can be chosen independently, with the

LTN acting as an interface to provide the explanation of the model to the user in the form of targeted FOL queries.

Logic Tensor Networks [33, 4] 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 also learnable. Thus, the grounding of symbols depends on a set of parameters θ. With a choice of a multilayer perceptron to 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 aggregate of the formulas in the knowledge base $\mathcal{K}$. The aggregate of the entries in $\mathcal{K}$ can be achieved by simply averaging all terms using the mean. The general mean is used because it gives flexibility to the user to tweak the method of the aggregation (by specifying p), meaning the relative importance of smaller and larger values[19]. The objective function $\mathrm{Sat}_A(\mathcal{G}_\theta(\mathcal{K}))$is therefore the satisfiability of all formulas in $\mathcal{K}$ which are maximized by tweaking the model parameters

which is subject to an aggregation A of all formulas, e.g. the generalised mean (p-mean).In its simplest form of binary classification without constraints, $\mathcal{K}$ will consist of one term for positive examples in the dataset and one for negatives. Notice that the approach described in Figure 8 is model-agnostic. The core extension of regular neural network optimisation 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 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.

A query can explain AI systems by connecting different model outputs, aggregating inputs for summarising the behaviour of a system in specific domains or relating specific inputs with specific features against each other and the output.

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. $max(1 - a, b)$). Semantics of quantifiers are defined by aggregation functions. In the following example, we approximate the binary connectives using the product t-norm and the corresponding t-conorm which define our fuzzy logic. The universal quantifier is defined as the generalised mean, also referred to as p-mean.

[19] *p-mean*$(x_1, \ldots, x_n) = \left(\frac{1}{n}\sum_{i=1}^{n} x_i^p\right)^{\frac{1}{p}}$.

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* $\mathcal{K}$ **do**
  Initialize $\mathcal{G}_\theta(P)$ `// each` P `can be a multilayer perceptron or output neuron`
for *epoch* $<$ *num-epochs* **do**
  max sat $\mathcal{G}_\theta(\mathcal{K})$ `// optimize` θ `to achieve max satisfiability of` $\mathcal{K}$
while *Revision* **do** `// user-defined Boolean`
  for *each FOL-query* ϕ_q **do**
    Calculate $\mathcal{G}\left(\phi_q\right)$ `// query the network to obtain the truth-value of` ϕ_q
    if $\mathcal{G}\left(\phi_q\right) < t$ `//` t `in [0,1] is a user-defined minimum sat value`
    then
      Add ϕ_q to $\mathcal{K}_{new}$
  Apply XAI-method `// we use Shapley values`
  for *each predicate* P **do**
    Inquire $\mathcal{G}_\theta(P)$ `// query predicate-specific groundings`
    if $\mathcal{G}_\theta$ *has undesired property* $f(\mathcal{G}_\theta)$ **then** `// user-determined desiderata`
      Revise $f(\mathcal{G})$ to $\mathcal{K}_{new}$ `// method-dependent revision`
  if $\mathcal{K}_{new} \neq \emptyset$ **then**
    $\mathcal{K} \leftarrow \mathcal{K} \cup \mathcal{K}_{new}$ $\theta^* = \arg\max_{\theta \in \Theta} \mathcal{G}_\theta(\mathcal{K})$ `// re-train the network`

Algorithm 2 illustrates the steps we take to continuously refine $\mathcal{K}_{new}$ with a human-in-the-loop. The queries derive from questions 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 behaviour which may not be as apparent as the above common questions In Figure 8, XAI method SHAP reports a disparity in how the variable *reported income* is used by the ML system for men and women when predicting credit risk [39]. LTN queries 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[20]. This cycle can be repeated until the revision process delivers satisfactory results to the user, with respect to model performance and behaviour.

6.4. *Case Study on recidivism prediction with a Deep Neural Network within the LTN framework for explainable revision*

We demonstrate the method mentioned above using a well-known fairness-related *COMPASS* dataset for recidivism prediction, obtained from the UCI machine learning repository[20]. For additional examples using alternative datasets, as well as comparisons with alternative methods, we refer the reader to [39].

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 discretisation, we obtain answers to the question: *How prediction for equally sized groups for each protected and unprotected variables differ across different risk categories for re-offending?* We find out whether the model achieves material equivalence between black and white prisoners of the same risk category on aggregation. Querying such axioms reveals a low satisfiability level ($sat_{\phi_i} \approx 0.5$), showing that the model is picking up on significant disparities in groups of medium risk of recidivism. This means that when the model is uncertain, it will predict re-offending for the protected group more often than for the unprotected group. While on the extremes, meaning in groups where the risk of recidivism is very low or very high, there are no significant differences in recidivism prediction of the model for the protected and unprotected group, which is in line with the desired notion of *group fairness* [14].

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 [39] and the notebook[20]. 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 discretisation 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 discretisations 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 generalised mean, converging from a simple average towards the min value, gradually (with higher relative importance for lower values) [14].

Integrating XAI methods with neural-symbolic approaches allows us to learn about the undesired behaviour of a model and intervene to address discrepancies, which is ultimately an important goal of explainability. We have demonstrated an interactive model-agnostic method and an algorithm for fairness

[20] Method demonstration including data is accessible at https://colab.research.google.com/drive/1Ip9Yb9gVRSRqaBKY9gOpiWn9pq3LovWG?usp=sharing. The original LTN repository adapted for this method is: https://github.com/logictensornetworks/logictensornetworks

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)

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 understand 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 [13]. For example, if an explanation has to be provided on a specific device, 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 [3] 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 and 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 [12]. The generation of such natural language explanations can rely on pipelines which takes the structured explanation content as input and, through several steps, performs its linguistic realizationThe work in [12] injects in such a pipeline a template system that implements the text structuring phase of the pipeline. Figure 9 shows the explanation generation process starting from a SHAP analysis of a model output.

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. Beside 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 10. 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[21] knowledge base for extracting information about the nutritional disease risks connected with such a food category. Beside public knowledge, the explanation graph can be enriched with user 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) [12] 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 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[22] and linguistic realization of the pipeline are performed with standard natural language processing engines such as RosaeNLG[23].

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 the healthcare domain.

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.

[21] https://www.nlm.nih.gov/research/umls/index.html

[22] *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.

[23] https://rosaenlg.org/rosaenlg/3.0.0/index.html

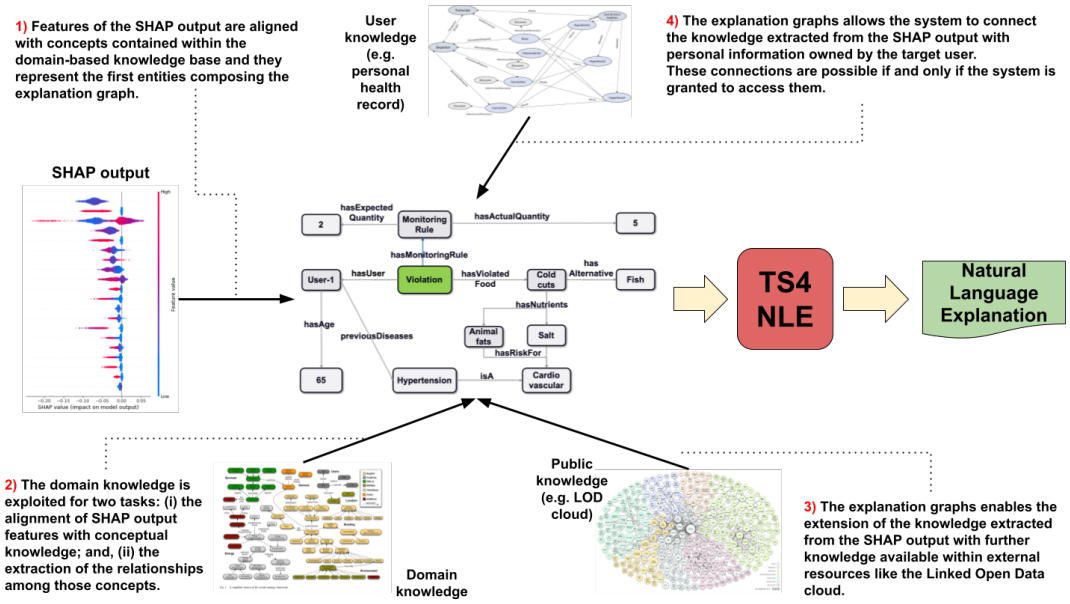

Figure 9: The images show the process of transforming a SHAP output into an explanation graph that is then transformed into its equivalent natural language explanation. Features contained within the SHAP output are transformed into concepts linked with a knowledge base related to the problem's domain. 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.

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 in order 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' lifestyle with respect to a standard notification of a bad lifestyle. A further tutorial of this use case is available online[24].

The explanation graph contains entities connected by relations encoding the rationale of the AI system decision. Fig. 10 contains the explanation graph for a 65 years 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 [24] and expanded taking into consideration additional strategies presented in [15]. 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

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

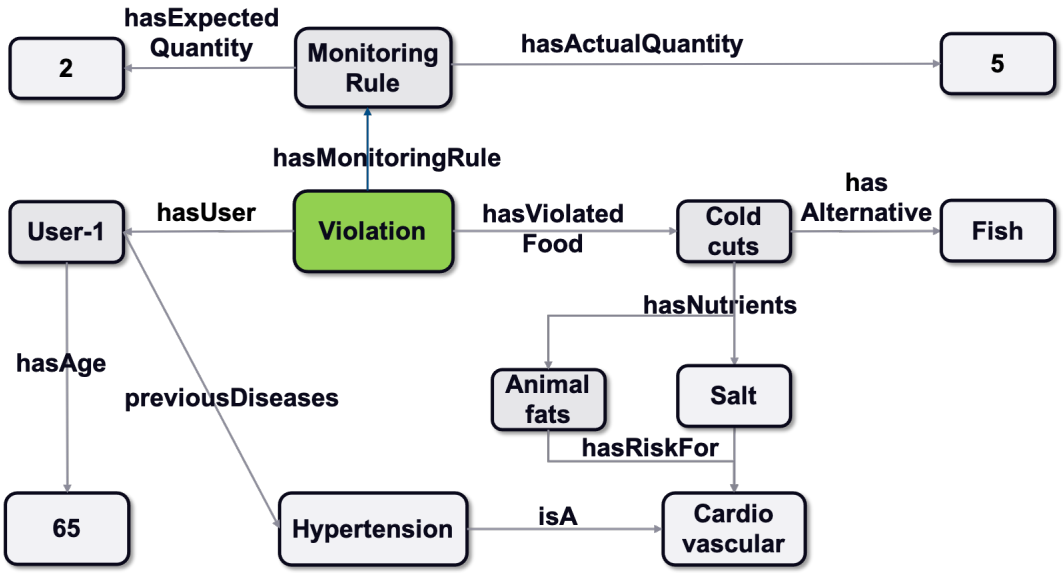

Figure 10: Explanation graph for users exceeding in cold cuts consumption in the diet & healthy lifestyle adherence application.

sentences that respect the grammatical rules of a desired language[25]. Below we describe the implemented strategies to automate the message generation, focusing also on linguistic choices. The template model together with an example for instantiating it, is represented in Figure 11.

Explanation Feedback: is the part of the message that informs the user about the not compliant behaviour, 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 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

[25]Current version of TS4NLE supports the generation of messages in English and Italian. In particular, Italian language requires a morphological engine (based on the open-source tool called morph-it[26]) to generate well-formed sentences starting from the constraints written in the template (e.g., tenses and subject consistency for verbs)

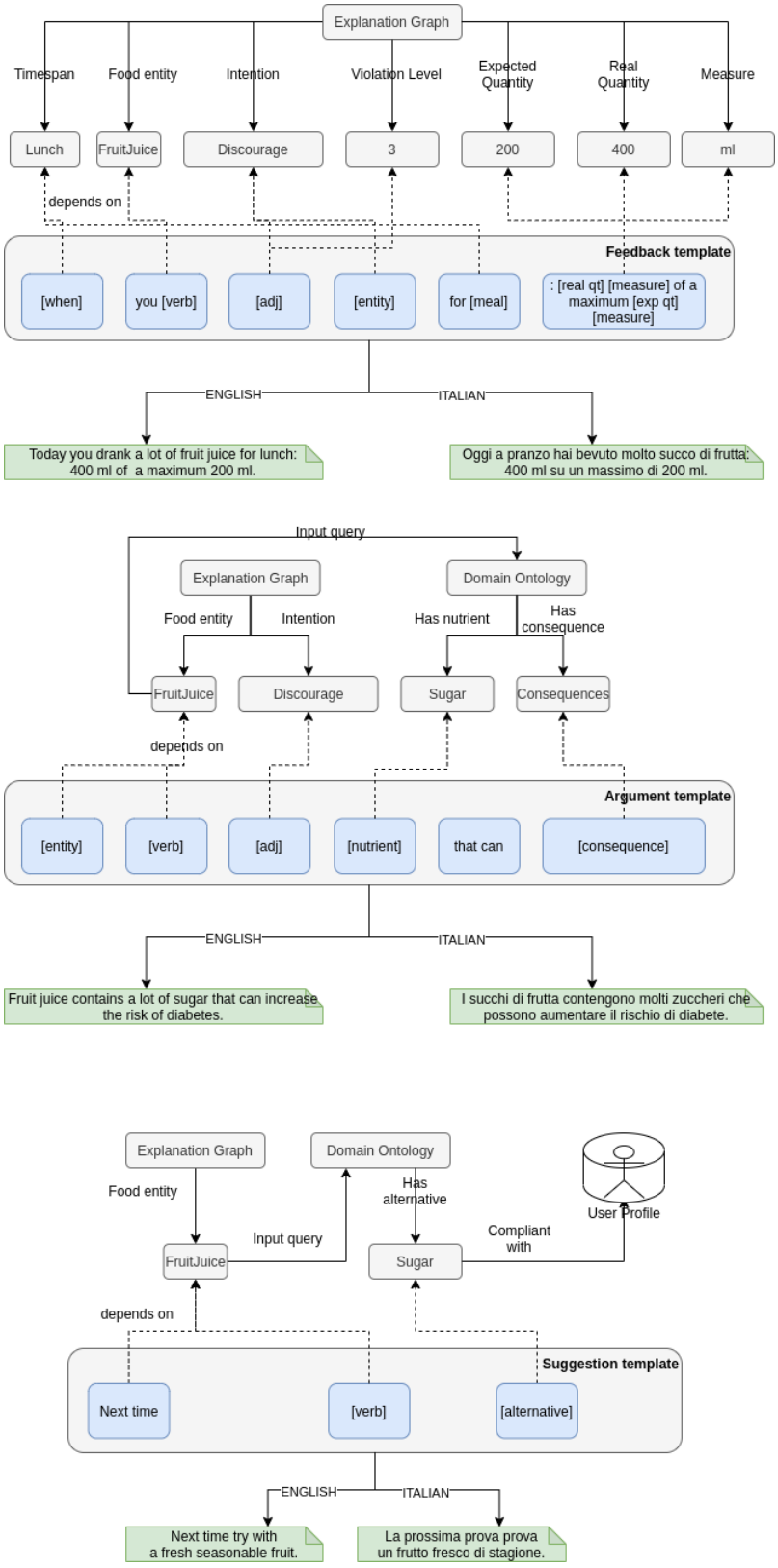

Figure 11: TS4NLE model (template and example of violation) for generating the text of an explanation. The top part represents the generation of the feedback. Choices on template and message chunks depend on the violation package. This holds also for both the argument and suggestion. Dashed lines represent a dependency relation. A template of type “informative” is used in the example. The middle part represents the generation of the argument, which is given as part of the explanation when violating diet restrictions. Finally, the bottom part represents the generation of the suggestion.

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. On the other hand, 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 in order 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[27] and the colaboratory (Colab notebook) session are online[28] 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 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

[27] https://github.com/ivanDonadello/TS4NLE
[28] https://colab.research.google.com/drive/1iCVSt7TFMruSzeg5DswLOzOR1n7xATbz

can be considered the major bottleneck of the TS4NLE approach. Such bottleneck can be mitigated by using machine learning with human-in-the-loop techniques to increase variability in the generated natural language explanations. This aspect will be further investigated as the main future direction of the TS4NLE approach.

8. Conclusion

We described a number of XAI techniques for extracting explanatory rationales from predictive models that make a particular decision for a particular input. These techniques were applied to a variety of use cases and models that incorporate different types of data of diverse nature. We walked through their implementation so that anyone can adapt them to a specific model and use case, with the ultimate goal of serving a didactic purpose. Explanations were aimed at both developer and decision maker audiences, and they are open and freely available for instructing and learning[29]. Future work should showcase more methods and how they deal with incomplete and multi-modal data [17].

9. Acknowledgements

This research was funded by the French ANRT industrial Cifre PhD contracts with SEGULA Technologies and with Tinubu Square. Parts of this work has received funding by the Austrian Science Fund (FWF), Project: P-32554. N. Díaz-Rodríguez is supported by the Spanish Government Juan de la Cierva Incorporación contract (IJC2019-039152-I). S. Tulli is supported by the European Union's Horizon 2020 research and innovation programme under grant agreement No 765955 (ANIMATAS Innovative Training Network).

References

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

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

[3] 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. Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information Fusion*, 2019.

[4] Samy Badreddine, Artur d'Avila Garcez, Luciano Serafini, and Michael Spranger. Logic Tensor Networks. *arXiv preprint arXiv:2012.13635*, 2020.

[5] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[6] Oana-Maria Camburu and Z. Akata. Natural-xai: Explainable ai with natural language explanations. In *ICML*, 2021.

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

[7] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, Jun 2002.

[8] Tianqi Chen and Carlos Guestrin. Xgboost. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Aug 2016.

[9] Danilo Croce, D. Rossini, and R. Basili. Auditing deep learning processes through kernel-based explanatory models. In *IIR*, 2019.

[10] Marina Danilevsky, Kun Qian, R. Aharonov, Yannis Katsis, B. Kawas, and P. Sen. A survey of the state of explainable ai for natural language processing. *AACL-IJCNLP 2020*, abs/2010.00711, 2020.

[11] J. Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, 2019.

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

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

[14] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. Fairness through awareness. In *ITCS 2012 - Innovations in Theoretical Computer Science Conference*, 2012.

[15] M. Guerini, O. Stock, and M. Zancanaro. A taxonomy of strategies for multimodal persuasive message generation. *Applied Artificial Intelligence Journal*, 21(2):99–136, 2007.

[16] Andreas Holzinger. Interactive machine learning for health informatics: When do we need the human-in-the-loop? *Brain Informatics*, 3(2):119–131, 2016.

[17] Andreas Holzinger, Bernd Malle, Anna Saranti, and Bastian Pfeifer. Towards multi-modal causability with graph neural networks enabling information fusion for explainable ai. *Information Fusion*, 71(7):28–37, 2021.

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

[19] Scott M. Lundberg, Gabriel G. Erion, and Su-In Lee. Consistent individualized feature attribution for tree ensembles, 2019.

[20] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, pages 4765–4774, 2017.

[21] Björn Lütjens, Brandon Leshchinskiy, Christian Requena-Mesa, Farrukh Chishtie, Natalia Díaz-Rodriguez, Océane Boulais, Aaron Piña, Dava Newman, Alexander Lavin, Yarin Gal, and Chedy Raïssi. Physics-informed GANs for Coastal Flood Visualization, 2020.

[22] Grégoire Montavon, Sebastian Lapuschkin, Alexander Binder, Wojciech Samek, and Klaus-Robert Müller. Explaining nonlinear classification decisions with deep taylor decomposition. *Pattern Recognition*, 65:211–222, 2017.

[23] Ramaravind K. Mothilal, Amit Sharma, and Chenhao Tan. Explaining machine learning classifiers through diverse counterfactual explanations. *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency.*, Jan 2020.

[24] H. op den Akker, M. Cabrita, R. op den Akker, V. M. Jones, and H.J. Hermens. Tailored motivational message generation: A model and practical framework for real-time physical activity coaching. *Journal of Biomedical Informatics*, 55:104–115, 2015.

[25] Charles Pierse. Transformers Interpret, 2 2021.

[26] Nicolas Pröllochs, S. Feuerriegel, and D. Neumann. Learning interpretable negation rules via weak supervision at document level: A reinforcement learning approach. In *NAACL-HLT*, 2019.

[27] Ayoub El Qadi, Natalia Díaz-Rodríguez, Maria Trocan, and Thomas Frossard. Explaining credit risk scoring through feature contribution alignment with expert risk analysts, 2021.

[28] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Model-agnostic interpretability of machine learning, 2016.

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

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

[31] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[32] Ramprasaath R Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, and Dhruv Batra. Grad-CAM: Why did you say that?, 2016.

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

[34] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3319–3328. JMLR. org, 2017.

[35] Joe Townsend, Theodoros Kasioumis, and Hiroya Inakoshi. Eric: Extracting relations inferred from convolutions, 2020.

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

[37] Joseph D Viviano, Becks Simpson, Francis Dutil, Yoshua Bengio, and Joseph Paul Cohen. Saliency is a possible red herring when diagnosing poor generalization. *International Conference on Learning Representations (ICLR) 20219*, 2019.

[38] Nikos Voskarides, E. Meij, M. Tsagkias, M. Rijke, and W. Weerkamp. Learning to explain entity relationships in knowledge graphs. In *ACL*, 2015.

[39] Benedikt Wagner and Artur S. D'Avila Garcez. Neural-Symbolic Integration for Fairness in AI. In *AAAI Spring Symposium AAAI-MAKE*, 2021.

[40] Lilian Weng. Attention? attention! *lilianweng.github.io/lil-log*, 2018.

[41] Adam White and Artur d'Avila Garcez. Measurable Counterfactual Local Explanations for Any Classifier. In *24th European Conference on Artificial Intelligence*, 2020.

[42] Amir Zadeh, Paul Pu Liang, Soujanya Poria, E. Cambria, and Louis-Philippe Morency. Multimodal language analysis in the wild: Cmu-mosei dataset and interpretable dynamic fusion graph. In *ACL*, 2018.